МИНИСТЕРСТВО СЕЛЬСКОГО ХОЗЯЙСТВА РОССИЙСКОЙ ФЕДЕРАЦИИ
ФГБОУ ВО РОССИЙСКИЙ ГОСУДАРСТВЕННЫЙ АГРАРНЫЙ УНИВЕРСИТЕТ – МСХА ИМЕНИ К.А. ТИМИРЯЗЕВА

Посвящается 100-летнему юбилею
Института экономики и управления АПК
Российского государственного
аграрного университета –
МСХА имени К.А. Тимирязева

Зинченко А.П., Уколова А.В., Демичев В.В., Тихонова А.В., Романцева Ю.Н., Харитонова А.Е., Быков Д.В., Ульянкин А.Е., Невзоров А.С., Сидоров А.И., Титов А.Д.

ЦИФРОВЫЕ ТЕХНОЛОГИИ АНАЛИЗА ДАННЫХ В СЕЛЬСКОМ ХОЗЯЙСТВЕ

Монография

Москва
Издательство «Научный консультант»
2022

УДК 004:311
ББК 65.05, 65.32
Ц75

Рецензенты:

профессор кафедры прикладной информатики Федерального государственного бюджетного образовательного учреждения высшего образования «Российский государственный аграрный университет – МСХА имени К.А. Тимирязева», доктор экономических наук **Худякова Е.В.**

ведущий научный сотрудник Всероссийского института аграрных проблем и нформатики имени А.А. Никонова, кандидат физико-математических наук **Сальников С.Г.**

Ц 75 **Цифровые технологии анализа данных в сельском хозяйстве:** монография / А.П. Зинченко, А.В. Уколова, В.В. Демичев, А.В. Тихонова, Ю.Н. Романцева, А.Е. Харитонова, Д.В. Быков, А.Е. Ульянкин, А.С. Невзоров, А.И. Сидоров, А.Д. Титов. – М.: Издательство «Научный консультант». – 2022. – 260 с.

ISBN 978-5-907477-96-4

В монографии представлены исследования преподавателей, аспирантов и магистров кафедры статистики и кибернетики ФГБОУ ВО РГАУ-МСХА имени К.А. Тимирязева в области цифровых технологий анализа агроэкономических данных.

Монография может представлять интерес для преподавателей, научных работников и сотрудников органов управления агропромышленным комплексом, специалистов сельского хозяйства, аспирантов и студентов, обучающихся по направлению «Информационные системы и технологии».

УДК 004:311
ББК 65.05, 65.32

ISBN 978-5-907477-96-4

СОДЕРЖАНИЕ

Предисловие 6

Глава 1 Методический подход к автоматизации и разработке модулей статистического анализа средствами языка программирования Python 9

1.1 Изучение предметной области 9

1.2 Алгоритм автоматизации статистического анализа 14

1.3 Реализации модулей статистического анализа средствами языка программирования Python 20

Библиографический список к главе 1 33

Глава 2 Разработка методики типизации сельскохозяйственных организаций на основе результатов сельскохозяйственных переписей с использованием языка программирования R 36

2.1 Особенности проведения и сводки итогов сельскохозяйственных переписей в России, США и Европе по типам предприятий 36

2.2 Разработка типологии сельскохозяйственных организации на основе результатов ВСХП–2016 в Липецкой области 43

2.3 Разработка информационной системы для автоматизации процесса типизации сельскохозяйственных организаций 48

Библиографический список к главе 2 56

Глава 3 Разработка информационной системы учета и обработки данных с поддержкой проведения статистического анализа на C++ 61

3.1 Теоретические основы проведения корреляционно-регрессионного анализа 61
3.2 Теоретические основы построения искусственных нейронных сетей 66
3.3 Разработка информационной системы 75
3.4 Разработка модуля взаимодействия с базами данных 79
3.5 Разработка модуля проведения корреляционно-регрессионного анализа 86
3.6 Разработка модуля построения нейронных сетей 96
3.7 Демонстрация работы модуля проведения корреляционно-регрессионного анализа 113
3.8 Демонстрация работы модуля построения нейронных сетей 120
Библиографический список к главе 3 125
Глава 4 Разработка мобильного приложения для сканирования молочной продукции 128
4.1 Технология разработки мобильного приложения 128
4.2 Исследование стоимости разработки мобильного приложения в России 139
4.3 Разработка мобильного приложения «Торру» 148
Библиографический список к главе 4 176
Глава 5 Системный подход к анализу больших данных АПК с использованием методов Data Science 179
5.1 Data Science как современный тренд развития IT в АПК 179
5.2 Применение Data Science в агропромышленном комплексе 185

5.3 Обоснование выбора технических средств авторской информационной системы для АПК 191
5.4 Описание и архитектура информационной системы 199
Библиографический список к главе 5 209
Глава 6 Разработка модуля информационной системы распознавания образов 211
6.1 Построение нейронных сетей для работы с изображениями ... 212
6.2 Функциональные модели разрабатываемой системы 219
6.3 Инфологическое проектирование системы 221
6.4 Разработка модуля информационной системы распознавания образов 224
6.6 Разработка инструкции пользования информационной системой 232
Библиографический список к главе 6 238
Приложения 242

ПРЕДИСЛОВИЕ

Цифровая трансформация ключевых отраслей экономики и повышение эффективности труда входят в перечень национальных целей развития России до 2030 года, определенных Указом Президента Российской Федерации № 474 от 21 июля 2020 г., ранее – Указом от 7 мая 2018 г. № 204, в рамках исполнения которых Правительством Российской Федерации сформирована и реализуется национальная программа «Цифровая экономика Российской Федерации». Цифровая трансформация с учетом текущих внешнеполитических и экономических рисков является одной из целей «Стратегии развития агропромышленного и рыбохозяйственного комплексов Российской Федерации на период до 2030 года», утвержденной Правительством Российской Федерации от 8 сентября 2022 г. № 2567-р.

Цифровизация сельского хозяйства базируется на современных способах производства сельскохозяйственной продукции и продовольствия с использованием цифровых технологий (интернет вещей, робототехника, искусственный интеллект, анализ больших данных, электронная коммерция и др.), обеспечивающих рост производительности труда и снижение затрат производства.

Подход к цифровой трансформации сельского хозяйства представлен в ведомственном проекте «Цифровое сельское хозяйство» в 2019 году, завершение которого планируется к 2024 году. В нем ставится ряд задач, который необходимо выполнить за данный период, например:

– создание и внедрение национальной платформы цифрового государственного управления сельским хозяйством «Цифровое сельское хозяйство» (ЦСХ). Все данные об объектах сельскохозяйственных ресурсов (земли сельскохозяйственного назначения, рабочий и продуктивный скот, сельскохозяйственная техника), сельскохозяйственном сырье и готовой продукции оцифрованы и включены в цифровую платформу ЦСХ;

– разработка системы сбора, хранения и обработки данных об объектах сельскохозяйственных ресурсов, сельскохозяйственном сырье и готовой продукции;

– разработка системы интеллектуального анализа данных и прогнозирования ЦСХ на основе технологий Advancedanalytics, DataDiscovery, DataMining, MachineLearning и искусственного интеллекта и др.

Цифровизация – это социально-экономическая трансформация, которую вызовет массовое внедрение и усвоение новых технологий создания, обработки, анализа и передачи информации. Сельское хозяйство становится одним из основных потребителей цифровых технологий. Генерация большого объема данных разнообразными датчиками в теплицах, полях, фермах и других производственных площадках особенно актуализирует применение цифровых технологий анализа агроэкономических данных. Широкое применение инструментов и методов Data Science, машинного обучения, нейронных сетей в конечном счете позволит существенно повысить эффективность принимаемых управленческих решений в АПК.

В монографии представлены результаты исследований по разработке информационных систем для автоматизации анализа агроэкономических данных, проводимых в рамках научно-исследовательской работы и подготовки выпускных квалификационных работ магистров по направлению «Информационные системы и технологии» в соответствии с программой развития Федерального государственного бюджетного образовательного учреждения высшего образования «Российский государственный аграрный университет - МСХА имени К.А. Тимирязева» на 2021-2030 годы.

Глава 1 «Методический подход к автоматизации и разработке модулей статистического анализа средствами языка программирования Python» написана член-корреспондентом РАН, доктором экономических наук Зинченко А.П. и кандидатом экономически наук, доцентом В.В. Демичевым.

Глава 2 «Разработка методики типизации сельскохозяйственных организаций на основе результатов сельскохозяйственных переписей с использованием языка программирования R» подготовлена кандидатом экономических наук, доцентом А.В. Уколовой и асситентом кафедры статистики и кибернетики А.Е. Ульянкиным при финансовой поддержке внутриуниверситетского конкурса «Аспирантский научный

контракт» в рамках программы развития Университета в соответствии с программой стратегического академического лидерства «Приоритет-2030».

Третья глава «Разработка информационной системы учета и обработки данных с поддержкой проведения статистического анализа на C++» написана кандидатом экономических наук, доцентом А.В. Уколовой и асситентом кафедры статистики и кибернетики Д.В. Быковым.

Над четвертой главой «Разработка мобильного приложения для сканирования молочной продукции» работали кандидат экономических наук, доцент Ю.Н. Романцева и асситент кафедры статистики и кибернетики А.С. Невзоров.

Пятую главу «Системный подход к анализу больших данных АПК с использованием методов Data Science» подготовили кандидат экономических наук, доцент А.В. Тихонова и магистрант А.И. Сидоров.

Шестая глава «Разработка модуля информационной системы распознавания лиц» написана кандидатом экономических наук, доцентом А.Е. Харитоновой и магистрантом 2022 года выпуска А.Д. Титовым.

Монография может представлять интерес для преподавателей, научных работников и сотрудников органов управления агропромышленным комплексом, специалистов сельского хозяйства, аспирантов и студентов.

Глава 1 Методический подход к автоматизации и разработке модулей статистического анализа средствами языка программирования Python

Автоматизация статистического анализа может быть условно разделена на три основных направления. Первое предполагает тщательную проработку **предметной области** исследования, которая, в свою очередь, включает в себя изучение понятия, системы показателей, методов и инструментов реализации автоматизации статистического анализа данного явления или процесса. Второе направление включает в себя планирование процесса автоматизации на основе соответствующего **алгоритма автоматизации статистического анализа**. Третье направление заключается в **реализации выбранными инструментами модулей статистического анализа**. В настоящей главе методический подход к автоматизации будет рассмотрен на примере статистического анализа эффективности сельского хозяйства.

1.1 Изучение предметной области

После конкретного определения изучаемого понятия, например, эффективности сельского хозяйства, следует представить систему статистических показателей, всесторонне отражающих исследуемое явление. В настоящей главе автоматизация статистического анализа проводиться на основе следующей системы абсолютных и относительных показателей условий функционирования и эффективности сельского хозяйства:

1) Площадь пашни, используемая предприятиями, организациями и гражданами, занимающимися сельскохозяйственным производством (тыс. га);
2) Площадь сельскохозяйственных угодий, используемых предприятиями, организациями и гражданами, занимающимися сельскохозяйственным производством (тыс. га);
3) Посевные площади всех сельскохозяйственных культур (тыс. га);
4) Продукция сельского хозяйства (млн. руб.);
5) Продукция животноводства и растениеводства (млн. руб.);
6) Численность занятых в сельском хозяйстве (чел.);
7) Объем субсидий (млн. руб.);

8) Инвестиции в основной капитал АПК (без учета лесного хозяйства) (млн. руб.);

9) Капитальные вложения за счет федерального бюджета и бюджетов субъектов РФ (без учета лесного хозяйства) (млн. руб.);

10) Удельный вес убыточных организаций (%);

11) Средний балл продуктивности климата;

12) Рентабельность производства с учетом и без учета субсидий (%);

13) Прибыль до налогообложения по всей деятельности сельскохозяйственных организаций, включая субсидии из бюджетов (млн. руб.);

14) Урожайность зерновых и зернобобовых, в весе после доработки (ц/га убранной площади);

15) Надой молока в расчете на 1 корову в сельскохозяйственных организациях (кг);

16) Внесение минеральных удобрений на один гектар посева сельскохозяйственных культур в сельскохозяйственных организациях (кг);

17) Расход кормов в расчете на одну условную голову КРС в сельскохозяйственных организациях (ц кормовых единиц);

18) Энергетические мощности в СХО, в расчете на 1 работника (л. с.);

19) Отношение субсидий в регионе к среднему по совокупности регионов;

20) Средний объем субсидий на 1 руб. стоимости продукции сельского хозяйства (руб.);

21) Удельный вес федерального бюджета в общем объеме инвестиций (%);

22) Удельный вес животноводства в производстве продукции сельского хозяйства (%);

23) Средняя рентабельность производства без учета субсидий (%);

24) Средняя рентабельность производства с учетом субсидий (%);

25) Рентабельность субсидий (%);

Следующим важным шагом изучения предметной области является описание автоматизируемых методов статистического анализа. В случае с эффективностью сельского хозяйства, согласно

изученной предметной области (или в соответствии с техническом заданием), статистический анализ должен включать анализ динамический рядов показателей эффективности сельского хозяйства, кластерный анализ, а также факторный анализ на основе регрессии (Рисунок 1.1).

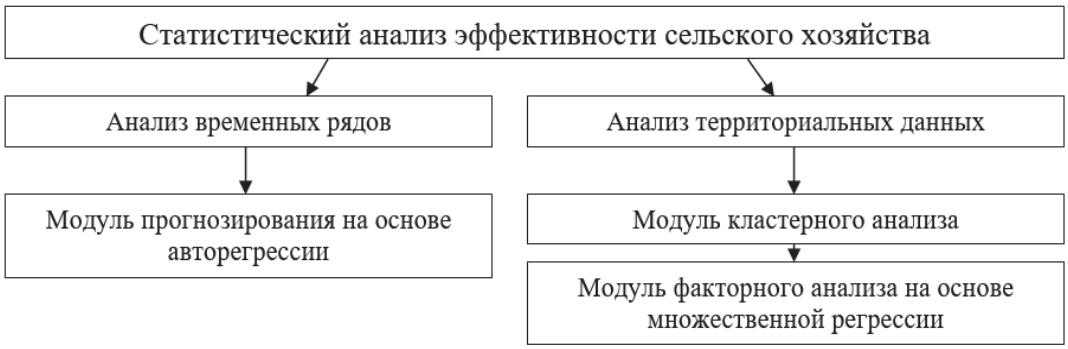


Рисунок 1.1 – Содержание статистического анализа эффективности сельского хозяйства

В нашем исследовании при апробации модуля прогнозирования будет использована автокорреляция одномерных временных рядов, то есть использована одна количественная переменная Yt – урожайность зерновых. При реализации модулей кластерного и регрессионного анализа будет применена система показателей эффективности сельского хозяйства, которая будет указана далее.

При изучении и обосновании выбора инструментов реализации следует подробно рассмотреть наиболее популярные и эффективные платформы, пакеты и языки программирования для статистического анализа.

В автоматизации статистического анализа ключевую роль играет создание базы данных необходимой для статистического анализа. Для реализации базы данных могут быть использованы такие системы как SQL Server Manager Studio, PostgreSQL, MySQL и другие. Представленные СУБД являются одними из самых популярных систем управления базами данных.

На следующем шаге изучения предметной области необходимо рассмотреть возможные языки программирования для реализации модулей автоматизации и отображения результатов статистического анализа. Наиболее популярными и удобными языками программирования для анализа данных являются Python и R. Оба языка используются по-разному. У Python более широкая

область применения. R – это язык, разработанный для наукоемких задач и анализа данных. Ранее R использовался прежде всего в научных исследованиях, однако сейчас он быстро распространяется на корпоративный рынок ввиду растущей популярности Big Data. Оба языка программирования имеют широкий спектр библиотек для реализации статистического анализа.

R – это язык программирования и программная среда для статистического анализа, отчетности для представления графики. Основным преимуществом использования R является то, что его можно использовать для реализации статистических концепций, таких как линейное и нелинейное моделирование, анализ временных рядов и кластеризации [3].

R – это интерпретируемый язык, поэтому каждая строка читается интерпретатором одна за другой. Это язык программирования высокого уровня. В нём имеется интерпретатор командной строки, и программист может напрямую вводить команды в командной строке. RStudio является общей интегрированной средой разработки (IDE), облегчающая программирование на R [2, 3].

В свою очередь особенности языка программирования Python заключаются в следующем:

1) Python легкий для обучения: относительно мало ключевых слов, простая структура и четко определенных синтаксис.

2) Python легко читаемый: блоки кода выделяются при помощи отступов, что совместно с ключевыми словами, взятыми из английского языка, значительно облегчают чтение кода.

3) Python легкий в обслуживании: простота обслуживания кода, написанного на этом языке.

4) Python имеет широкую кросс-платформенную стандартную библиотеку.

5) Есть возможность "на лету" тестировать нужные участки кода благодаря наличию интерактивного режима.

6) Python без проблем запускается на разных платформах, при этом сохраняет одинаковый интерфейс, независимо от того на каком компьютере работает пользователь, что характеризует его портативность.

7) При необходимости в Python можно внедрять низкоуровневые модули написанные на иных языках

программирования для наиболее гибкого решения поставленных задач.

8) Python имеет возможность работать с базами данных. В стандартной библиотеке можно найти модули для работы с большинством коммерческих баз данных.

9) На Python возможно создание GUI (Графического интерфейса пользователя) приложений, которые будут работать независимо от типа операционной системы.

Конечно, Python также имеет ряд свои недостатков. Одним из главных недостатков является его относительно низкая скорость выполнения [16]. Python является языком с полной динамической типизацией, автоматическим управлением памятью. Если на первый взгляд это может казаться преимуществом, то при разработке программ с повышенным требованием к эффективности, Python может значительно проигрывать по скорости своим статическим братьям (C/C++, Java, Go).

Второй недостаток Python связан с ростом кодовой базы. Следить за типом передаваемых друг другу данных бывает очень сложно, отсюда появляются проблемы. Для решения такого рода проблем динамические языки используют type annotations, проектов mypy по статическому анализу кода и так далее. Это же в свою очередь накладывает негативный оттенок на эстетическую сторону кода.

У R и Python много общего, так, например оба языка поддерживают объектно-ориентированное, императивное и процедурное проектирование. Также они оба являются интерпретируемыми языками. Могут быть использованы в разработке алгоритмов и являются языками программирования высокого уровня. Оба являются бесплатными и с открытым исходным кодом. Могут быть интегрированы с базами данных, такими как SQL [21], MySQL и Oracle [7]. Оба поддерживают разные файлы, такие как файлы CSV, Excel, XML JSON. И Python, и R просты в использование и изучение.

При этом различия в деталях определяют специфику каждого языка. R – это язык программирования и программная среда для статистических вычислений, графического представления и отчетности. Python – это интерпретируемый язык программирования общего назначения высокого уровня [17]. R

поддерживает такие структуры данных, как векторы, списки, матрицы, массивы, коэффициенты и фреймы данных [18]. В то время как Python поддерживает структуру данных, такую как списки, словари и кортежи [22]. Общая интегрированная среда разработки для программирования на языке R – это RStudio [19]. Для Python распространенно интегрированной средой разработки является Spyder, PyCharm, Eclipse и другие.

Ключевое различие между R и Python заключается в том, что R является статистически ориентированным языком программирования, в то время как Python является языком программирования общего назначения.

Таким образом, для дальнейшей реализации модулей в нашем примере будет использован язык программирования Python, который также имеет преимущества быстрой интеграции с базой данных.

1.2 Алгоритм автоматизации статистического анализа

После подробного рассмотрения предметной области, например, понятия, системы статистических показателей и методов статистического анализа эффективности сельского хозяйства необходимо итеративно следовать основным этапам разработки модулей и автоматизации статистического анализа. Структурированный и алгоритмизированный подход к автоматизации статистического анализа обеспечивает повышение его эффективности и прозрачности для всех заинтересованных в разработке сторон. Кроме того, поэтапное описание процесса автоматизации отдельных процессов анализа, позволяет работать над ним проектно, в команде. Рассмотрим основные этапы алгоритма автоматизации и разработки модулей статистического анализа (Приложение А).

1. Процесс автоматизации начинается с поставновки цели исследования. На этом этапе необходимо четко понимать «что» необходимо сделать, «как» и «почему». Этот этап является важным моментом для разработки, в конечном итоге, всех предполагаемых модулей анализа. Цель для нашего примера может быть сформулирована следующим образом: разработка модулей автоматизации статистического анализа эффективности сельского хозяйства.

2. На втором этапе осуществляется сбор данных. Проведение качественного исследования требует сбора данных из всех доступных для авторов исследования источников. На этом этапе данные формируются в таблицах Excel. Источники данных, применяемые в данном исследовании, могут быть представлены следующим образом (Рисунок 1.2).

Рисунок 1.2 – Источники статистической информации для автоматизации анализа

В результате второго этапа вы получите абсолютные и относительные показатели, которые еще необходимо будет дорабатывать и преобразовывать. В данных должны быть выявлены и устранены всевозможные ошибки. Данные из разных источников соединяются и преобразуются.

3. Подготовка и исследование данных. На этом этапе данные из первоначальной формы преобразуются в данные, которые можно использовать в анализе, моделировании, визуализации. Рассчитываются требуемые средние и относительные показатели, строятся диаграммы рассеяния, позволяющие осуществить первоначальное исследование данных.

4. Проектирование и реализация базы данных. Этап непосредственного проектирования автоматизированной системы,

в том числе базы данных. Этап подразумевает создание технического задания, построение UML-диаграмм [9], которые позволяют графически описать существующие взаимосвязи при автоматизации анализа (предполагается построение диаграммы деятельности, вариантов использования, состояний, последовательности) [23]. Непосредственное создание базы данных в среде Microsoft SQL Manager Studio или других СУБД.

5. Пятый этап – моделирование данных. Это самый важный момент, так как программирование на языке Python, с использованием библиотек Pandas, numpy и других, позволяет проводить кластеризацию регионов, по достаточно большому количеству показателей, рассчитывать средние по каждому кластеру, а также формировать все необходимые элементы корреляционно регрессионного анализа и других методов, в том числе методов машинного обучения.

Разработанные модули могут быть применены не только для регионов, но также для любых объектов наблюдения, требующих прогнозирования, кластеризации и регрессионного анализа [5].

Если говорить о методах кластеризации, регрессии и авторегрессии, то все они являются методами машинного обучения, которое, в свою очередь, существенно повысило возможности статистического анализа [6].

Создание базы данных позволяет накапливать данные предметной области, интегрировать их со средствами обработки данных или выгружать для проведения дальнейшего анализа на языке программирования Python. В качестве среды разработки, в данном примере, выбрана Spyder 4, как интегрированная среда разработки для научного программирования на языке Python [14].

Python – многоцелевой высокоуровневый язык программирования. Его дизайн позволяет писать хорошо читаемый код [8]. Он позволяет использовать эффективные высокоуровневые структуры данных и предлагает простой, но эффективный подход к объектно-ориентированному программированию. Сочетание изящного синтаксиса, динамической типизации в интерпретируемом языке делает Python идеальным языком для написания сценариев и ускоренной разработки приложений в различных сферах и на большинстве платформ [4, 8].

Python является интерпретируемым, то есть исходный код на Python не компилируется в машинный код, а выполняется непосредственно с помощью специальной программы-интерпретатора.

Также это интерактивный язык. Это означает, что возможность писать код прямо в оболочке интерпретатора и вводить новые команды по мере выполнения предыдущих.

При этом Python является объектно-ориентированным языком программирования. Он поддерживает принципы ООП, которые подразумевают инкапсуляцию кода в особые структуры, именуемые объектами [4].

В Python анализ данных реализуется с помощью имеющихся библиотек. Библиотека языка программирования – это просто набор модулей и функций, которые облегчают некоторые специфические операции с использованием этого языка программирования.

В Python существует множество библиотек для программирования, в том числе и для анализа данных – это pandas, numpy, matplotlib, plotly, scikit learn, seaborn, statmodels, tkinter и другие.

NumPy позволяет очень эффективно обрабатывать многомерные массивы. Многие другие библиотеки построены на NumPy, и без неё было бы невозможно использовать pandas, Matplotlib, SciPy или scikit-learn – именно поэтому она занимает первое место в списке. Также в ней есть несколько хорошо реализованных методов, например, функция random, которая гораздо качественнее модуля случайных чисел из стандартной библиотеки. Функция polyfit отлично подходит для простых задач по прогнозной аналитике, например, по линейной или полиномиальной регрессии [11].

Библиотека Pandas позволяет работать с двухмерными таблицами, такие, как в SQL и Excel, поскольку изначально в Python нет такой возможности. Pandas поддерживает все самые популярные форматы хранения данных: csv, excel, sql, буфер обмена, html и другие. Эта высокоуровневая библиотека позволяет строить сводные таблицы, выделять колонки, использовать фильтры по параметрам, выполнять группировку по параметрам, запускать функции (сложение, нахождение медианы, среднего,

минимального, максимального значений), объединять таблицы и многое другое. В pandas можно создавать и многомерные таблицы.

Matplotlib – самая популярная Python-библиотека для визуализации данных, что позволяет представлять их в наглядном виде, изучать более подробно, чем это можно сделать в обычном формате, и наглядно изложить другим людям. Она не так проста в использовании, но с помощью 4-5 наиболее распространённых блоков кода для простых линейных диаграмм и точечных графиков можно научиться создавать их очень быстро.

Для машинного обучения и построения прогнозной аналитики, наиболее подходящая библиотека – scikit-learn. Она содержит ряд методов, охватывающих всё, что может понадобиться в течение первых нескольких лет в карьере аналитика данных: алгоритмы классификации и регрессии, кластеризацию, валидацию и выбор моделей. Также её можно применять для уменьшения размерности данных и выделения признаков.

Машинное обучение в scikit-learn позволяет импортировать правильные модули и запускать метод подбора модели. Сложнее вычистить, отформатировать и подготовить данные, а также подобрать оптимальные входные значения и модели. Поэтому прежде, чем взяться за scikit-learn, нужно, во-первых, отработать навыки работы с Python и pandas, чтобы научиться качественно подготавливать данные, а во-вторых, освоить теорию и математическую основу различных моделей прогнозирования и классификации, чтобы понимать, что происходит с данными при их применении.

Библиотека SciPy включает средства для обработки числовых последовательностей, лежащих в основе моделей машинного обучения: интеграции, экстраполяции, оптимизации и других. Как и в случае с NumPy, чаще всего используется не сама SciPy, а упомянутая выше библиотека scikit-learn, которая во многом опирается на неё. SciPy является полезной библиотекой, поскольку содержит в себе ключевые математические методы для выполнения сложных процессов машинного обучения в scikit-learn.

Seaborn – это библиотека визуализации данных Python, основанная на matplotlib. Он предоставляет высокоуровневый интерфейс для рисования привлекательной и информативной статистической графики. Seaborn содержит более адекватные

дефолтные настройки оформления графиков. Также в библиотеке есть достаточно сложные типы визуализации, которые в matplotlib потребовали бы большого количество кода.

Библиотека Statsmodels предоставляет инструменты для статистического моделирования. Он основан на экосистеме SciPy и требует от нее поддержки данных в виде массивов NumPy. Он предоставляет набор методов статистического тестирования и моделирования, а также инструменты, предназначенные для анализа временных рядов, что также может быть использовано для прогнозирования.

В данном исследовании мы будем использовать такие библиотеки, как Pandas, Numpy, Seaborn и Statsmodels.api для модуля корреляционно-регрессионного анализа. И библиотеки Pandas, Matplotlib, Numpy и SciPy для модуля кластерного анализа. Библиотека streamlit применяется для создания веб-приложения автоматизированного анализа данных.

Streamlit – библиотека Python для отображения результатов анализа данных в веб-приложении с возможностью упрощения дальнейшего пользования разработанными модулями анализа и автоматизации последнего при изменении или обновлении исходных данных [20].

6. Автоматизация и отображение результатов статистического анализа. Streamlit – библиотека Python для отображения результатов анализа данных в веб-приложении с возможностью упрощения дальнейшего пользования разработанными модулями анализа и автоматизации последнего при изменении или обновлении исходных данных [20].

Возможности данной библиотеки позволяют создавать многостраничное веб-приложение с функциями редактирования текста, отображения графиков, таблиц, виджетов и других объектов. В данной приложение могут быть загружены не только графики, но и изображения, видео, аудиозаписи.

Веб-приложение на streamlit может быть персонализировано с возможностью установления имя пользователя и пароля.

Streamlit имеет усиленные возможности по изменению шрифтов, цвета страницы, боковой панели и ее компонентов, применения встроенных компонентов визуализации и анализа данных, расширенные возможности компонентов API.

Посредством Streamlit Cloud созданное веб-приложение может быть выложено для совместной работы над проектом в облаке, что существенно повышает функциональность и производительность дальнейшего развития этого приложения. Таким образом, streamlit позволяет быстро и легко автоматизировать статистический анализ и применение методов машинного обучения. Изменяя, например, исходные данные вы можете достаточно быстро реализовать все компоненты необходимого анализа. В данном исследовании создано веб-приложение для отображения авторегрессии, кластерного и регрессионного анализа. Приступим, собственно, к разработке модулей статистического анализа.

1.3 Реализации модулей статистического анализа средствами языка программирования Python

Разработка модуля прогнозирования. Результативной переменной является показатель урожайность зерновых культур (ц/га), как один из ключевых показателей эффективности использования сельскохозяйственных угодий за период с 1991 г. по 2020 гг. Данный модуль может быть использован также при прогнозировании других показателей как на уровне страны или региона, так и на уровне отдельного предприятия или структурного подразделения. Рассмотрим подробно данный модуль, а реализация других модулей будет представлена в приложении В и Г.

Загрузим данные по урожайности в интерактивную среду разработки на языке Python – Spyder 3.7:

```
#Импортируем библиотеки для загрузки данных и анализа данных
import pandas as pd
#Загрузим данные в виде переменной data
data = pd.read_csv("yield.csv",";")
#Выведем первые 5 значений
data.head()
print(data.head())
```

В итоге в среду разработки будут загружены данные по урожайности за представленный период времени. Далее построим линейный график для характеристики тенденции исследуемого показателя:

```
    #Импортируем библиотеки для построения
линейного графика
    import matplotlib.pyplot as plt
    # Зададим переменные для построения графика
    t = data['year']
    s = data['yield']
    #Создадим область графика и оси
    fig, ax = plt.subplots()
    ax.plot(t, s)
    #Зададим название графика, осей и сетку
основной области
    ax.set(xlabel='Годы', ylabel='Урожайность',
           title='Урожайность зерновых в России,
ц/га')
    ax.grid()
    #Сохраним рисунок в требуемом формате и
выведем рисунок
    fig.savefig("test.png")
    plt.show()
```

Результатом запуска кода программы будет следующий линейный график, отражающий динамику урожайности зерновых (Рисунок 1.3).

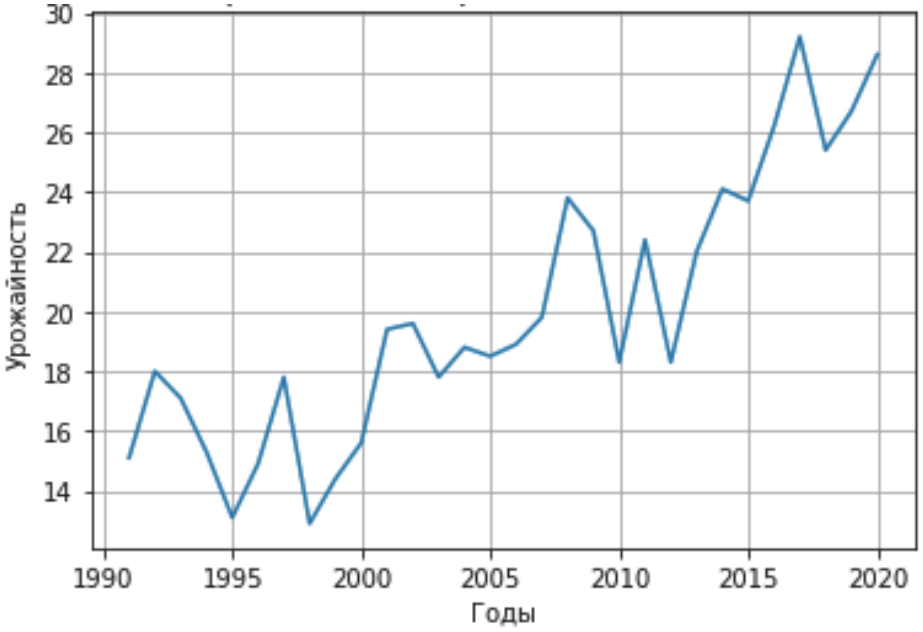


Рисунок 1.3 – Динамика урожайности зерновых в России в 1991-2020 гг.

По данным рисунка 1.3 можем предположить наличие трендовой (T), циклической (S) и случайной компонент (E). Наличие трендовой компоненты говорит о возможности прогнозирования значений с применением модели авторегрессии.

Для оценки наличия автокорреляции необходимо построить диаграмму рассеяния между урожайностью в одном году (Yt) и урожайности в следующем периоде (Yt+1).

Реализовать диаграмму рассеяния с лаговыми переменными возможно на основе использования функции lag_plot(). Воспользуемся данной функцией для построения диаграммы рассеяния с лаговыми переменными по данным об урожайности зерновых в России:

```
#Импортируем функцию и библиотеку для построения диаграммы
from pandas.plotting import lag_plot
from matplotlib import pyplot
#Построим диаграмму
lag_plot(data['yield'])
pyplot.show()
```

В результате получим диаграмму рассеяния, которая отражает зависимость урожайности в периоде t+1 от урожайности в периоде t (Рисунок 1.4).

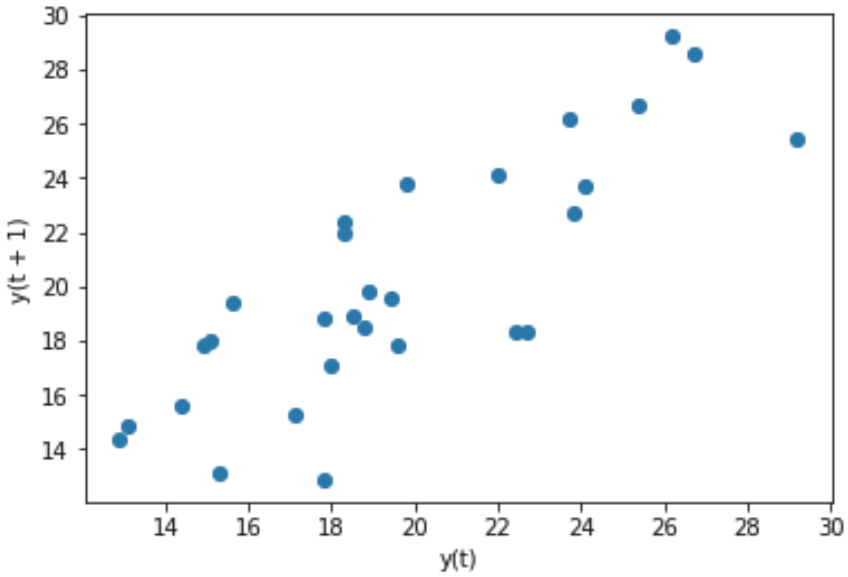


Рисунок 1.4 – Зависимость урожайности в периоде t+1 от периода t

По данным представленного рисунка можем предположить, что между представленными переменными существует прямая тесная взаимосвязь. Подобная диаграмма может быть построена для лаговых переменных любого порядка.

Более точно определить взаимосвязь между переменными можно на основе коэффициента корреляции. Рассчитать коэффициенты корреляции можно на основе функции corr():

```
#Импортируем необходимые функции и библиотеки
from pandas import DataFrame
from pandas import concat
#Преобразуем данные и зададим переменные для построения корреляционной матрицы
values = DataFrame(data["yield"].values)
dataframe = concat([values.shift(1), values], axis=1)
dataframe.columns = ['t-1', 't+1']
#Построим матрицу и выведем результат
result = dataframe.corr()
print(result)
```

При выпадении значений коэффициента из пределов от -0,5 до 0,5 корреляция принимается как высокая. Построенная матрица указывает на высокую взаимосвязь между наблюдением t -1 и t + 1 (Таблица 1.1).

Таблица 1.1 – Матрица коэффициентов корреляции

	t-1	t+1
t-1	1	0.820524
t+1	0.820524	1

Данный метод является достаточно простым и универсальным, однако, подходит только для переменных с лагом равным 1. Рассмотрим расширенную версию указанного подхода.

На основе построения графика может быть отражена взаимосвязь переменных с большей величиной лага. Для этого может быть применена библиотека Pandas и ее функция autocorrelation_plot():

```
#Импортируем функцию autocorrelation_plot и построим график
from pandas.plotting import autocorrelation_plot
```

```
autocorrelation_plot(data['yield'])
pyplot.show()
```

В результате получаем график автокорреляционной функции, которые позволяет определить статистически достоверный лаг для построения модели авторегрессии с точностью 95 и 99% (Рисунок 1.5).

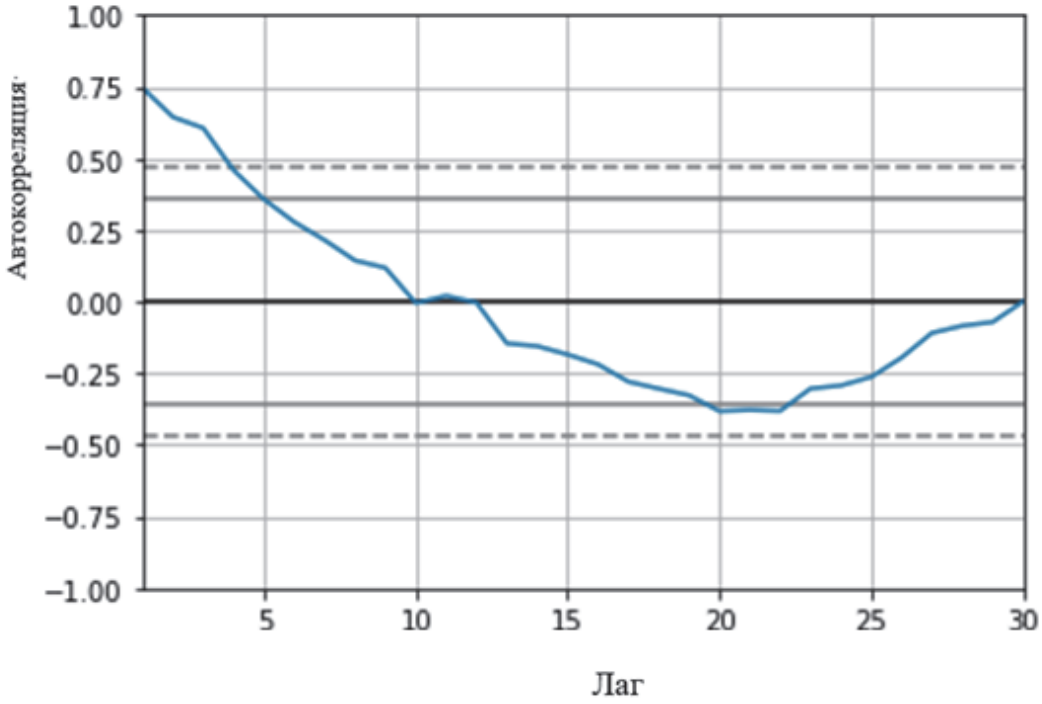


Рисунок 1.5 – Автокорреляционная функция для урожайности (функция autocorrelation_plot())

Запуск кода позволяет вывести график колебаний коэффициента корреляции, позволяющий сделать вывод о кардинальных изменениях величины коэффициента примерно каждые 10 лет (таким образом, подтверждается цикличность урожайности). Также по представленному рисунку можно сделать вывод с вероятностью 99%, что для данного временного ряда максимально возможным и статистически значимым является лаг равный 3, с вероятностью 95% лаг равный 4.

Автокорреляционная функция может быть также реализована в виде столбиковой диаграммы с указанием достоверности величины лага (максимально возможный лаг зададим равный 30). Для этого может быть использована функция plot_acf():

```
#Импортируем функцию plot_acf и построим столбиковую диаграмму
```

```
    from statsmodels.graphics.tsaplots import
plot_acf
    plot_acf(data["yield"], lags=25)
    pyplot.show()
```

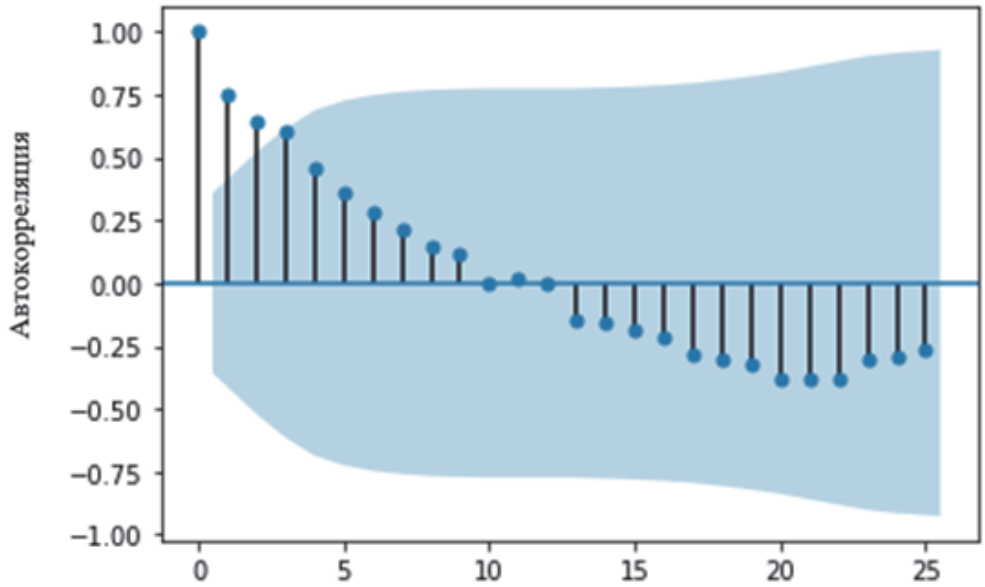


Рисунок 1.6 – Автокорреляционная функция для урожайности (функция plot_acf())

Построенная автокорреляционная функция подтверждает ранее сделанные выводы о цикличности ряда равной 10 годам и статистическую достоверность лага равного 3, либо 4 с разным уровнем достоверности.

Рассмотренная часть модуля позволяет анализировать автокорреляцию временных рядов. Далее необходимо разработать модуль, отвечающий за моделирование авторегрессии и непосредственно прогнозирование уровней ряда.

Библиотека statmodels позволяет построить модель регрессии, где можно задать конкретную величину лага и тренировать существующую модель при наличии обновления данных. В нашем случае максимальный лаг равен 7 (так как мы имеем 30 наблюдений).

Мы можем создать модель на основе функции AutoReg() и тренировать ее, вызывая функцию fit(). Это возвращает объект AutoRegResults.

Построив модель однажды, мы можем использовать модель для построения прогноза, просто вызывая функцию predict(), чтобы тренировать ее на новых данных:

```
    # Импортируем библиотеки и создадим модель
авторегрессии
    from statsmodels.tsa.ar_model import AutoReg
    from          sklearn.metrics          import
mean_squared_error
    from math import sqrt
    # Сформируем лаговые данные
    X = data['yield'].values
    train, test = X[1:len(X)-7], X[len(X)-7:]
    # Построим модель авторегрессии
    model = AutoReg(train, lags=7)
    model_fit = model.fit()
    print('Коэффициенты  авторегрессии:  %s'  %
model_fit.params)
    # Построим прогноз
    predictions                                =
model_fit.predict(start=len(train),
end=len(train)+len(test)-1, dynamic=True)
    for i in range(len(predictions)):
        print('прогнозируемые=%f, ожидаемые=%f' %
(predictions[i], test[i]))
    rmse     =     sqrt(mean_squared_error(test,
predictions))
    print('RMSE: %.3f' % rmse)
    # Выведем график прогноза
    pyplot.plot(test)
    pyplot.plot(predictions, color='red')
    plt.xlabel('Номер  прогнозируемого  периода
(года)')
    plt.ylabel('Урожайность')
    pyplot.show()
```

Весь код разработанного модуля прогнозирования представлен в приложении А. Разработанный фрагмент модуля позволяет нам построить прогноз на 7 лет (Рисунок 1.7).

```
Коэффициенты авторегрессии: [ 5.72321435  0.3299833   0.03176015 -0.10755712  0.04309956  0.36264285
 -0.18751391  0.29228236]
прогнозируемые=5.723214, ожидаемые=24.100000
прогнозируемые=16.643756, ожидаемые=23.700000
прогнозируемые=21.146116, ожидаемые=26.200000
прогнозируемые=21.347126, ожидаемые=29.200000
прогнозируемые=22.989290, ожидаемые=25.400000
прогнозируемые=15.729151, ожидаемые=26.700000
прогнозируемые=21.651837, ожидаемые=28.600000
RMSE: 9.630

In [22]:
```

Рисунок 1.7 – Вывод модели авторегрессии и прогнозируемых значений урожайности

Таким образом, модель авторегрессии примет следующий вид:

$$Yt = 5{,}72 + 0{,}329Y_{t-1} + 0{,}032Y_{t-2} - 0{,}011Y_{t-3} + 0{,}043Y_{t-4} + 0{,}363Y_{t-5} - 0{,}188Y_{t-6} + 0{,}292Y_{t-7} \quad (1.1)$$

Данная модель статистически достоверна на уровне значимости 5%. Используя данную модель, отметим, что прогноз урожайности на 2022 г. составит 29,9 ц/га.

На основе данной модели также может быть построен график прогноза урожайности на 7 лет.

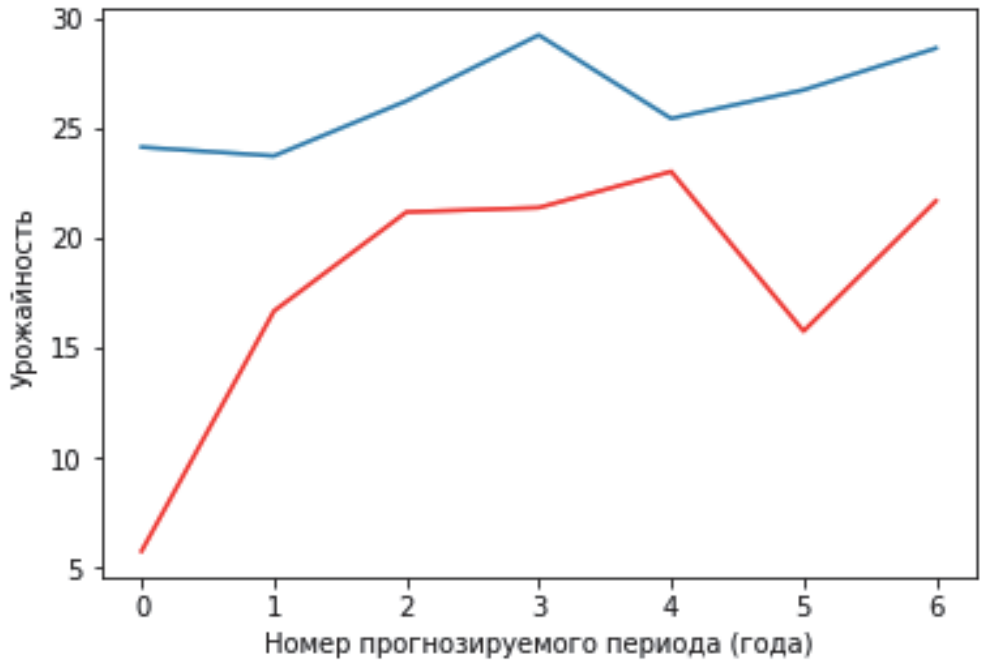


Рисунок 1.8 – Прогноз урожайности зерновых в России на 7 лет

Например, красным цветом обозначено прогнозируемое значение, а синим ожидаемое (фактическое значение ряда). Последнее отличается величиной ошибки, тогда как прогнозируемое построено точно по модели авторегрессии.

Разработанный модуль авторегрессионного анализа может быть использовать для анализа временных рядов и прогнозирования урожайности и других показателей эффективности сельского хозяйства на уровне страны, регионов и отдельных предприятий или подразделений, включая прогнозирование на основе больших данных [15].

6. Отображение и автоматизация. На шестом этапе процесса автоматизации необходимо разработать веб-приложение для дальнейшей автоматизации и отображения результатов статистического анализа. Здесь также необходимо продемонстрировать полученные результаты и сделать выводы, в том числе касательно наиболее важных факторов эффективности сельского хозяйства.

В таком виде результаты могут быть представлены заказчику и пользователю. Конечно, результаты могут быть представлены как в виде презентации, так и в виде научно-исследовательского отчета. Однако, создание веб-приложения позволяет автоматизировать производимый анализ, с возможностью его дальнейшего развития и углубления. Таким образом, модули анализа могут быть применены в другом проекте или задействованы в рабочем процессе при изменении или обновлении набора данных.

Автоматизация статистического анализа эффективности сельского хозяйства может быть использована как инструмент статистического анализа любых сложных явлений и процессов. Специалист может использовать разработанные модули анализа обращаясь к разработанному веб-приложению и подгружая новые данные для анализа. Обновление и дополнение данных может привести к необходимости повторения анализа. По этой причине важным моментом является автоматизация моделей и всего анализа.

Прогнозирование урожайности, соответственно, так же может быть выведено на уровень пользовательского интерфейса. Исключительно подгружая данные, пользователь можете осуществить прогнозирование показателя по модели авторегрессии, кластерный анализ и корреляционно-регрессионный анализ любого

из представленных явлений. В этом заключается отображение и автоматизация результатов статистического анализа.

Как отмечалось ранее библиотека streamlit широко применяется в создании простых веб-приложений визуализации данных и отображения моделей машинного обучения. Установка библиотеки осуществляется на основе стандартной процедуры установки библиотек Python pip install. После установления библиотеки мы можем открыть веб-приложение в браузере при помощи команды «streamleat hello». Данную команду необходимо ввести в терминале Anaconda Prompt, что позволит открыть в веб-браузере шаблон приложения.

Открыв приложение, далее могут вносится всевозможные изменения, которые требует проект. Внести изменения в название на первой странице приложения. Итак, запустим приложение и попробуем поменять название на главной странице и затем запустить приложение через терминал:

```
    #Импортируем библиотеки для загрузки данных и анализа данных
    import streamlit as st
    import pandas as pd
    #название первой страницы приложения
    st.write("""
    # Веб-приложение "Статистический анализ".
    Авторегрессия, кластерный и регрессионный анализ!
    """)
```

Следующим шагом зададим левую колонку в приложении и сразу укажем ее название. Кроме того, пропишем поле выбора в левой колонке, где укажем вид анализа: авторегрессия, кластерный, регрессия:

```
    #зададим левую колонку в приложении
    st.sidebar.header('Вид и параметры анализа')
    #вид анализа
    option = st.sidebar.selectbox('Выбрать вид анализа:', ('Авторегрессия', 'Кластерный', 'Регрессия'))
```

После обновления приложения появится левая колонка в приложении с новым названием и возможностью выбора проводимого анализа.

Дальнейшие действия должны быть направлены на имплементацию модели в приложение. Пример реализации веб-приложения и имплементации модели авторегрессии в данное веб-приложение представлен в приложении Д.

Запустив приложение, мы сможем увидеть титульную страницу с названием «Статистический анализ» (Рисунок 1.8).

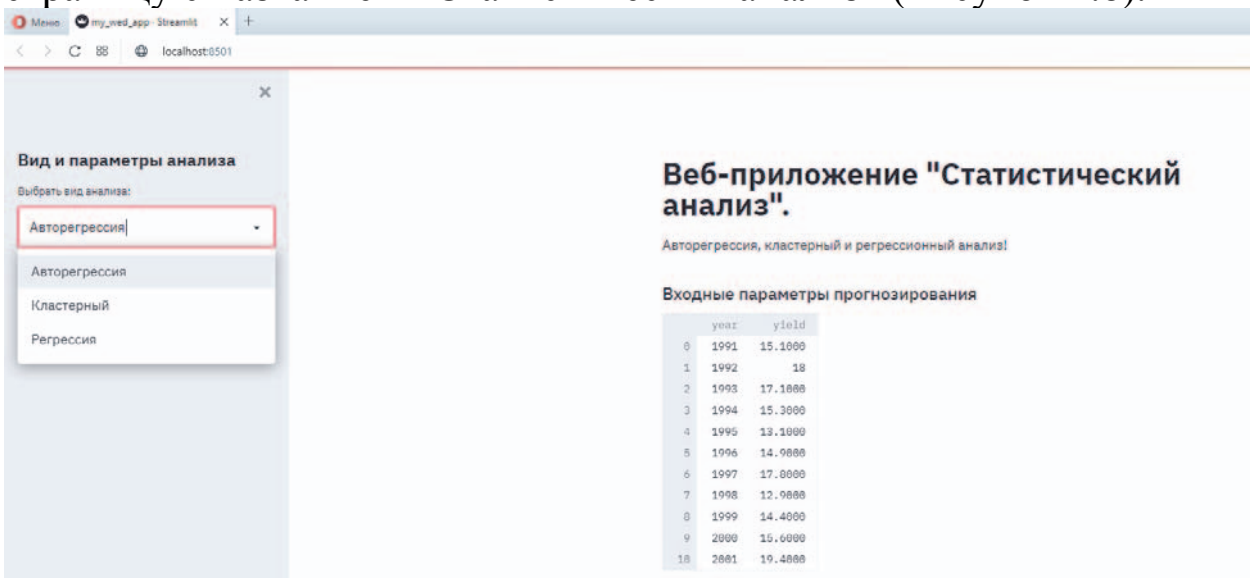


Рисунок 1.8 – Титульная страница веб-приложения «Статистический анализ»

На титульной странице указаны исходные данные для построения прогноза урожайности зерновых на основе авторегрессии, а также представлена левая панель, где может быть выбран вид проводимого анализа.

Все результаты реализованного ранее модуля авторегрессионного анализа с целью прогнозирования урожайности могут быть прописаны в настоящем приложении:

```
    #пропишем вывод линейного графика
    st.subheader('График динамики показателя')
    st.line_chart(data['yield'])
    #Матрица коэффициентов корреляции
    st.subheader('Матрица коэффициентов
корреляции')
    st.write(result)
```

```
    #пропишем название таблицы и зададим значения
выводимых параметров в приложении
    st.subheader('Параметры                модели
авторегрессии')
    st.write('Коэффициенты авторегрессии: %s' %
model_fit.params)
    #зададим прогнозируемые и ожидаемые значения
    st.subheader('Прогнозируемые   и   ожидаемые
значения')
    for i in range(len(predictions)):
    st.write('прогнозируемые=%f, ожидаемые=%f' %
(predictions[i], test[i]))
    #выведем точность модели
    st.subheader('Точность модели')
    st.write(rmse)
```

Данный код позволит отобразить результаты прогнозирования на основе авторегрессии. Например, график динамики урожайности, диаграмму рассеяния, матрицу парных коэффициентов корреляции, автокорреляционную функцию, параметры модели авторегрессии, прогнозируемые и ожидаемые значения и их график, точность построенной модели (Рисунок 1.9).

Для того, чтобы открыть приложение в браузере его необходимо запустить используя следующую команду в терминале (Anaconda Prompt), предварительно указав путь к папке, где сохранен код Python: streamlit run имя файла.py. В результате в терминале появится ссылка на приложение (url-ссылка), которую можно открыть в браузере. Кроме того, данное приложение может быть сохранено в формате html-документа и запущено при необходимости.

В качестве заключения стоит отметить, что в данное веб-приложение может быть внесено много улучшений, особенно в части разработки раздела предобработки данных, расширения и добавления модулей статистического, эконометрического, многомерного анализа, применения методов машинного обучения.

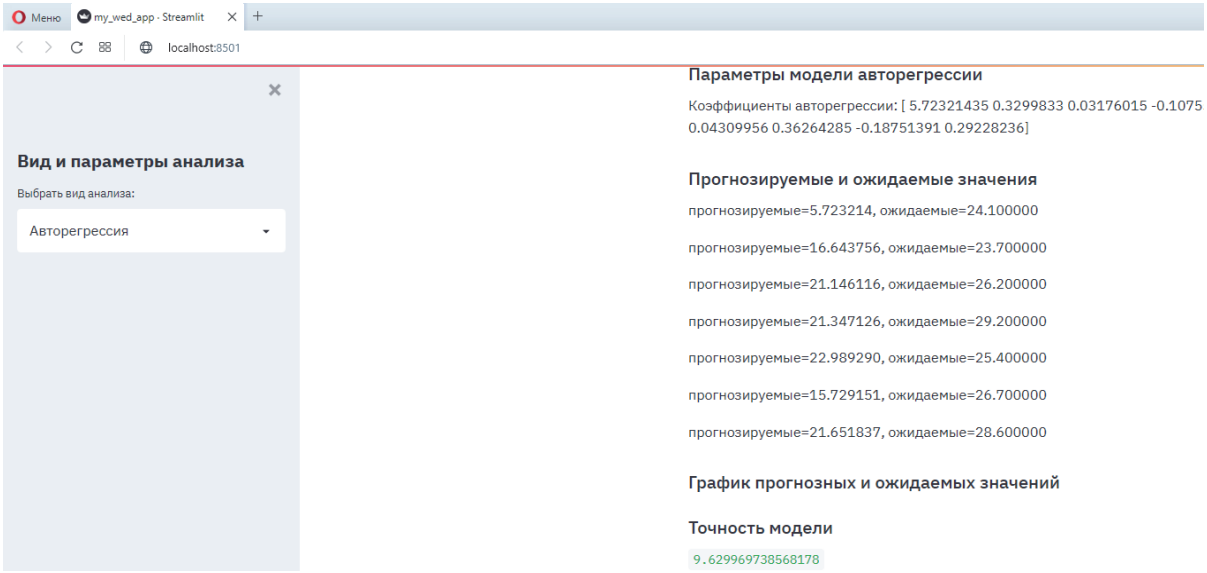


Рисунок 1.9 – Отображение параметров, прогнозируемых значений и точности модели авторегрессии в веб-приложении

7. *Тестирование.* На заключительном этапе проводиться тестирование разработанных модулей. Здесь главным является выбор стратегии тестирования (стратегии «белого» и «черного» ящиков), разработка тестов для отдельных задач, сравнение и оценка различных методов тестирования и их возможностей. Без проведения тестирование дальнейшее развитие модулей автоматизации является затруднительным. Так для модуля регрессионного анализа может быть выполнен структурный контроль.

Стратегия «белого ящика» предполагает также подготовку тестов для проверки методами покрытия операторов, решений и условий. Однако, если в коде отсутствуют циклы, условия и функции, то данные методы могут быть опущены. Стратегия «белого ящика» проверяет внутреннюю структуру программы, ее переходы, решения и логические выражения из условий. Может использоваться для небольших программ или их частей, потому что даже в программе средней сложности количество повторяющихся маршрутов слишком велико для всестороннего тестирования маршрутов.

Тестирование модуля программы отображения и автоматизации результатов статистического анализа может быть проведено методами стратегии «черного ящика». Формулировка разрабатываемого продукта: написать модуль статистического анализа, который позволяет вычислить: построить график динамики показателя, диаграмму рассеяния, корреляционную

матрицу, авторегрессионную модель, прогнозируемые значения и показатель качества модели. В данном случае могут быть применены методы эквивалентного разбиения, анализа граничных значений, тестирование на предположение об ошибке.

Стратегия «черного ящика» хорошо подходит для больших программ и их тестирования без углубления в структуру. Метод эквивалентного разбиения дает возможность с помощью нескольких тестов проверить работу программы при вводе значений из определенных областей. Анализ граничных значений тестирует работу на границах этих областей. С помощью метода предположения об ошибке можно найти такие случаи, которые могут быть не учтены при проектировании.

При использовании стратегии «черного» ящика нет необходимости разбираться в способе реализации программы и в том, как она устроена на более низком уровне. Но может быть пропущено множество ошибок. Одни могут себя совсем не проявлять во внешней работе программы, а для выявления других требуется сделать определённую последовательность действий, тестирование которой может производиться по методу предположения об ошибке, но для этого необходима хорошая интуиция. Таким образом, проведенное тестирование на основе стратегий «белого ящика» и «черного ящика» позволят проверить работоспособность разрабатываемых модулей статистического анализа.

Представленный алгоритм автоматизации статистического анализа носит итеративный характер, что подразумевает периодическое возвращение к каждому из этапов и их корректировка. В тоже время, соблюдение указанных пунктов, повышает эффективность и результативность полученных выводов. Описанный процесс позволяет иметь выверенный план исследования, проработку предмета, проблемы, которую необходимо анализировать, а также представление об ожидаемых результатах, еще до непосредственного проведения анализа.

Библиографический список к главе 1

1. Агропромышленный комплекс России в 2008-2021 гг.: статистический сборник / Министерство сельского хозяйства

Российской Федерации. – М.: ФГБНУ «Росинформагротех», 2008-2022. – Текст: непосредственный.

2. Буре, В.М. Методы прикладной статистики в R и Excel: Учебное пособие / В.М. Буре, Е.М. Парилина, А.А. Седаков. – 3-е изд., стер. – СПб.: Издательство «Лань», 2019. – 152 с.: ил. – Текст: непосредственный.

3. В чем разница между R и Python. – URL: https://raznisa.ru/raznica-mezhdu-r-i-python/ (Дата обращения: 15.04.2022). – Текст: электронный.

4. Введение в Python. – URL: http://pythonicway.com/python-overview (дата обращения: 18.05.2022). – Текст: электронный.

5. Демичев, В.В. Автоматизация статистического анализа данных отрасли растениеводства сельскохозяйственного предприятия / В.В. Демичев, Н.П. Дроздов // Научные труды КубГТУ. – 2020. – №6. – С. 11-20. – Текст: непосредственный.

6. Дэви, С. Основы Data Science и Big Data. Python и наука о данных / С. Дэви, М. Арно, А. Мохамед. – СПб.: Питер, 2018. – 336 с. – Текст: непосредственный.

7. Зайцев, П. MySQL по максимуму / П. Зайцев, В. Ткаченко, Б. Шварц. – Издательство Питер, 2018. – 864 с. – Текст: непосредственный.

8. Любанович, Б. Простой Python. Современный стиль программирования / Б. Любанович. – СПб.: Питер, 2019. – 480 с.: ил. – Текст: непосредственный.

9. Мацяшек, Л.А. Анализ и проектирование информационных систем с помощью UML 2.0 / Л.А. Мацяшек. – Издательство Вильямс, 2016. – 816 с. – Текст: непосредственный.

10. Нестратова, А. А. Автоматизация статистического анализа эффективности государственной поддержки сельского хозяйства Российской Федерации: направление подготовки 09.04.02 «Информационные системы и технологии»: магистер. дис. / Нестратова Анастасия Андреевна; ФГБОУ ВО РГАУ-МСХА имени К.А. Тимирязева. – М., 2020. – 117 с. – Текст: непосредственный.

11. Пять ключевых библиотек и пакетов для анализа данных на Python. – URL: https://techrocks.ru/2018/07/22/5-key-libraries-and-packets-for-data-analysis-in-python/ (дата обращения: 21.05.2022). – Текст: электронный.

12. Тринадцать способов настроить визуализацию матрицы корреляции. – URL: https://datastart.ru/blog/read/seaborn-heatmaps-

13-sposobov-nastroit-vizualizaciyu-matricy-korrelyacii (дата обращения: 30.04.2022). – Текст: электронный.

13. Федеральная служба государственной статистики. – URL: https://www.gks.ru/ (Дата обращения: 23.11.2021). – Текст: электронный.

14. Anaconda. – URL: https://www.anaconda.com/products/individual (дата обращения: 14.09.2021). – Текст: электронный.

15. Chan, J. SQL: Learn SQL (using MySQL) in One Day and Learn It Well. SQL for Beginners with Hands-on Project / J. Chan. – Book 5. – LCF Publishing, 2018. – 166 p. – Текст: непосредственный.

16. Deitel, P.J. Intro to Python for Computer Science and Data Science: Learning to Program with AI, Big Data and The Cloud / P.J. Deitel, D. Harvey. – 1 edition. – Pearson, 2019. – 880 p. – Текст: непосредственный.

17. Guttag, J.V. Introduction to Computation and Programming Using Python, second edition: With Application to Understanding Data / J.V. Guttag. – second edition edition. – The MIT Press, 2016. – 472 p. – Текст: непосредственный.

18. Haslwanter, T. An Introduction to Statistics with Python: With Applications in the Life Sciences (Statistics and Computing) / T. Haslwanter. – 1st ed. – Springer, 2016 – 278 p. – Текст: непосредственный.

19. Matloff, N. Probability and Statistics for Data Science: Math + R + Data (Chapman & Hall/CRC Data Science Series) / N. Matloff. – 1 edition. – Chapman and Hall/CRC, 2019. – 444 p. – Текст: непосредственный.

20. Streamlit. Официальный сайт библиотеки Python streamlit. – URL: https://streamlit.io/(дата обращения: 23.05. 2022). – Текст: электронный.

21. Taylor, A.G. SQL All-in-One For Dummies (For Dummies (Computer/Tech)) / A.G. Taylor. – 3 edition. – For Dummies, 2019. – 768 p. – Текст: непосредственный.

22. Turner, C. Python for Data Analysis: The Ultimate Beginner's Guide to Learn The Basics Of Data Analysis, Pandas and Python Statistics. Discover Programming Language and Machine Learning / C. Turner. – Independently published, 2020. – 164 p. – Текст: непосредственный.

Глава 2 Разработка методики типизации сельскохозяйственных организаций на основе результатов сельскохозяйственных переписей с использованием языка программирования R

2.1 Особенности проведения и сводки итогов сельскохозяйственных переписей в России, США и Европе по типам предприятий

Всероссийская сельскохозяйственная перепись, в соответствии с Федеральным законом от 21 июля 2005 г. № 108-ФЗ «О Всероссийской сельскохозяйственной переписи» и постановлением Правительства Российской Федерации от 10 апреля 2013 г. № 316 «Об организации Всероссийской сельскохозяйственной переписи 2016 года», была проведена с 1 июля по 15 августа 2016 г., кроме отдаленных и труднодоступных территорий, где транспортное сообщение затруднено, там перепись проходила с 15 сентября по 15 ноября 2016 г. [3].

Результаты ВСХП-2016 были опубликованы в 8 томах, где выходные данные были представлены в таблицах размером от двух до четырёх страниц принятого формата А4. На сайте Росстата все тома переписи доступны в электронном виде в файлах формата pdf, что существенно затрудняет работу с данными. Например, для расчёта относительного показателя «Среднегодовая численность работников в расчете на 100 га сельскохозяйственных угодий по малым сельскохозяйственным предприятиям» необходимо сравнить таблицу 36 второго тома с таблицей 4 третьего тома.

Программа Всероссийской сельскохозяйственной переписи 2016 г. разрабатывалась в соответствии с Программой Всемирной сельскохозяйственной переписи ФАО ООН, а также с учетом опыта проведения Всероссийской сельскохозяйственной переписи 2006 г. В программу переписи включены показатели о характеристике объектов сельскохозяйственной переписи и занятости в сельском хозяйстве, наличии и использовании земельных ресурсов, структуре земель по землепользователям; посевных площадях и многолетних насаждений по широкому перечню сельскохозяйственных культур, о поголовье сельскохозяйственных животных по видам; о характеристике трудовых ресурсов, производственной инфраструктуре, технических средствах и

технологиях в разрезе категорий сельскохозяйственных производителей [12].

В соответствии с утвержденной методологией (приказ Росстата от 29.02.2016 г. № 101) личные подсобные и другие индивидуальные хозяйства граждан в городских поселениях (городских округах) обследовались с применением выборочного метода статистического наблюдения. В связи с этим сводные данные по этой категории хозяйств представлены только по объектам, попавшим в выборочную совокупность [10].

Объекты исследования переписи:

- сельскохозяйственные организации;
- микропредприятия;
- крестьянские (фермерские) хозяйства;
- индивидуальные предприниматели;
- подсобные сельскохозяйственные предприятия несельскохозяйственных организаций;
- граждане, имеющие ЛПХ, ИЖС, не входящие в объединения в городских поселениях;
- граждане, имеющие ЛПХ, ИЖС, не входящие в объединения в сельских поселениях;
- садоводческие, огороднические, животноводческие и дачные некоммерческие объединения граждан.

Особенностью Всероссийской сельскохозяйственной переписи 2016 г. стало применение современных технологий при сборе сведений от респондентов. Впервые при опросе граждан, ведущих личное подсобное хозяйство, использовались планшетные компьютеры. Это позволило существенно повысить качество информации и ускорить процесс сбора и обработки данных.

Также новшеством при проведении переписи в 2016 г. стало предоставление сельскохозяйственным организациям, крестьянским (фермерским) хозяйствам и индивидуальным предпринимателям возможности сообщать сведения по программе переписного листа в электронном виде через систему web-сбора Росстата.

В 2021 г. с 1 по 30 августа в России впервые прошла сельскохозяйственная микроперепись (СХМП-2021). Было установлено, что микропереписи будут проводить между всероссийскими сельскохозяйственными переписями не позднее,

чем через 5 лет после очередной (последняя проводилась в 2016 г.) [20].

Из постановления Правительства РФ следует, что СХМП-2021 проведена в отношении отдельных субъектов сельскохозяйственной переписи на основе выборки не менее 30% объектов сельскохозяйственной переписи и включит в себя сведения о размерах посевных площадей по группам культур, площадей, занятых плодовыми многолетними насаждениями, и поголовье сельскохозяйственных животных.

Росстат определен ответственным за подготовку и проведение сельскохозяйственной микропереписи, обработку полученных сведений, подведение итогов наблюдения и их опубликование. Приказом Росстата от 30 ноября 2020 г. №741 была утверждена методология проведения СХМП-2021. А уже приказом от 18 декабря 2020 г. был утвержден календарный график мероприятия по подготовке и проведению сельскохозяйственной микропереписи 2021 г., автоматизированной обработке, подведению итогов и их официальной публикации, из которого следует, что предварительные итоги СХМП-2021 должны быть подведены в IV квартале 2021 г., окончательные – до конца I квартала 2023 г. [13].

В Российской Федерации не разработана официальная типология сельскохозяйственных предприятий, как это сделано в США и Европейском союзе. Результаты Всероссийской сельскохозяйственной переписи 2016 года были подведены по типам сельскохозяйственных организаций, выделенных в соответсвии с федеральным законом «О развитии малого и среднего предпринимательства в Российской Федерации» от 24 июля 2007 года № 209-ФЗ. Были выделены микропредприятия, малые и не относящиеся к субъектам малого предпринимательства, то есть крупные и средние организации попали в одну последнюю группу. Верхние границы нормативов отнесения к группам в ФЗ № 209 явно завышены для сельского хозяйства, что неизбежно приводит к смешению разных типов сельскохозяйственных производителей.

Перепись сельского хозяйства в Европе. Европейская типология ферм. Базовое обследование структуры фермерских хозяйств, сокращенно FSS, также известное как обследование структуры аграрных хозяйств, проводится всеми государствами-

членами Европейского союза (ЕС). FSS проводится последовательно на всей территории ЕС с использованием общей методологии на регулярной основе и, таким образом, обеспечивает сопоставимые и репрезентативные статистические данные по странам и времени на региональном уровне (вплоть до уровня NUTS 3). Каждые 3–4 года FSS проводится в форме выборочного обследования, а раз в десять лет – переписи [24].

Последняя перепись в Европе проводилась в 2009/2010 г., последующие годы – выборочные обследования. В соответствии с регламентом следующая перепись должна была проводиться в 2020 г., однако в связи с возникновением новой коронавирусной инфекции и последовавшими за этим ограничения в странах, Всемирной организацией FAO было рекомендовано перенести проведение национальных переписей на год позже. Таким образом, из 62 стран, участвующих во Всемирной переписи сельского хозяйства, 45 стран были вынуждены отложить ее проведение. В последующем FAO было рекомендовано проводить переписи с использованием информационно-коммуникационных средств для минимизации рисков распространения COVID-19.

Основной единицей, лежащей в основе FSS, является аграрное хозяйство. FSS охватывал все аграрные хозяйства, которые соответствуют минимальным требованиям, установленным применимым законодательством [28].

Обследование структуры аграрных хозяйств, также известное как обследование структуры хозяйств (FSS), помогает оценить сельскохозяйственную ситуацию в ЕС, отслеживать тенденции и изменения в структуре аграрных хозяйств, а также моделировать влияние внешних событий или политики [30].

По отдельным отраслям стандартизированный выпуск (SO) определяется в расчете на единицу площади или голову животных как произведение продуктивности на цену производителя, которая не учитывает НДС и другие налоги на продукты и прямые выплаты фермерам. SO определяется как средняя величина за пятилетний базисный период. Валовой стандартизированный выпуск предприятия как его общая рыночная производительность определяется статистическими ведомствами в рамках

трехгодичных обследований структуры сельского хозяйства как сумма произведений стандартизированного выпуска в расчете на единицу площади/голову животных и общей площади/поголовья животных данного предприятия. В соответствии со статьей 5 Регламента ЕС № 1242/2008 под стандартизованным выпуском понимается стандартизированная стоимость валовой продукции.

Классификация хозяйств построена на структуре стандартизированного валового выпуска, она имеет несколько уровней иерархии, отдельные производственные направления (53 класса) объединяются в 20 классов основного производственного направления, которые в свою очередь укрупняются и образуют 9 классов общего производственного направления:

1. Специализирующиеся на растениеводстве;
2. Специализирующиеся на садоводстве;
3. Специализирующиеся на выращивании многолетних насаждениях;
4. Специализирующиеся на животноводстве;
5. Специализирующиеся на разведении свиней и птицы;
6. Смешанное растениеводство;
7. Смешанное животноводство;
8. Смешанное растениеводство-животноводство;
9. Хозяйства, не относящиеся ни к одной категории [28].

Второй метод классификации сельскохозяйственных предприятий – по экономическим размерам предприятия – основан на величине стандартного валового выпуска, который измеряет производственный потенциал предприятия и дает приблизительную оценку его возможным доходам (Таблица 2.1).

До 2007 г. в обследовании структуры фермерских хозяйств (FSS) и в сети данных бухгалтерского учета фермерских хозяйств (FADN) использовался стандартизованный маржинальный доход (SGM) для классификации аграрных хозяйств по типу фермерских хозяйств и экономическим размерам (Решение Комиссии (ЕЕС) № 377/1985). Начиная с FSS 2010, эта классификация использует стандартный валовой выпуск (SO) вместо SGM в соответствии с Регламентом Комиссии (ЕС) № 1242/2008, который заменен Регламентом, делегированным Комиссией (ЕС) № 1198/2014 для FSS 2016 и более поздних версий [17].

Таблица 2.1 – Классификация предприятий по экономическим размерам

Класс	Граница, тыс. евро
I	Менее 2
II	От 2 до 4
III	От 4 до 8
IV	От 8 до 15
V	От 15 до 25
VI	От 25 до 50
VII	От 50 до 100
VIII	От 100 до 250
IX	От 250 до 500
X	От 500 до 750
XI	От 750 до 1000
XII	От 1000 до 1500
XIII	От 1500 до 3000
XIV	3000 и более

Принципы обеих концепций SGM и SO одинаковы, отличается только способ их расчета:

SGM = SO (Выход продукции)+ Прямые платежи – Затраты (2.1)

Перепись сельского хозяйства в США. Типология ферм. В 2017 г. проведена 29-я федеральная перепись сельского хозяйства. В отличие от России, перепись уже пятый раз проводится Национальной Службой сельскохозяйственной статистики (НАСС) Министерства сельского хозяйства США (USDA), а в течение 156 лет (1840-1996 гг.) за нее отвечало Бюро переписи населения Министерства торговли США. Закон об ассигнованиях 1997 г. содержал положение, согласно которому ответственность за проведение переписи сельского хозяйства возлагалась на НАСС.

В 1976 г. Конгресс санкционировал проведение переписей сельского хозяйства в 1978 и 1982 гг., чтобы базовый год совпал с другими экономическими переписями. Эта корректировка сроков установила 5-летний период проведения переписи, в годы, заканчивающиеся 2 и 7 [27].

Типология ферм в первую очередь фокусируется на «семейной ферме» или любой ферме, где большая часть бизнеса принадлежит производителю и лицам, связанным с производителем, включая родственников, которые не живут в домашнем хозяйстве производителя. Министерство сельского

хозяйства США (USDA) определяет ферму как любое место, из которого в течение данного года было произведено и продано (или обычно было бы продано) более 1000 долларов сельскохозяйственной продукции.

Типология классифицирует все фермы на уникальные группы, в 2017 г. она претерпела изменения (Рисунок 2.1). Семейные фермы классифицируются на основе валового денежного дохода фермы (GCFI). GCFI включает в себя продажи сельскохозяйственных производителей сельскохозяйственных культур и скота, сборы за поставку товаров по производственным контрактам, государственные платежи и доходы, связанные с фермерскими хозяйствами [23].

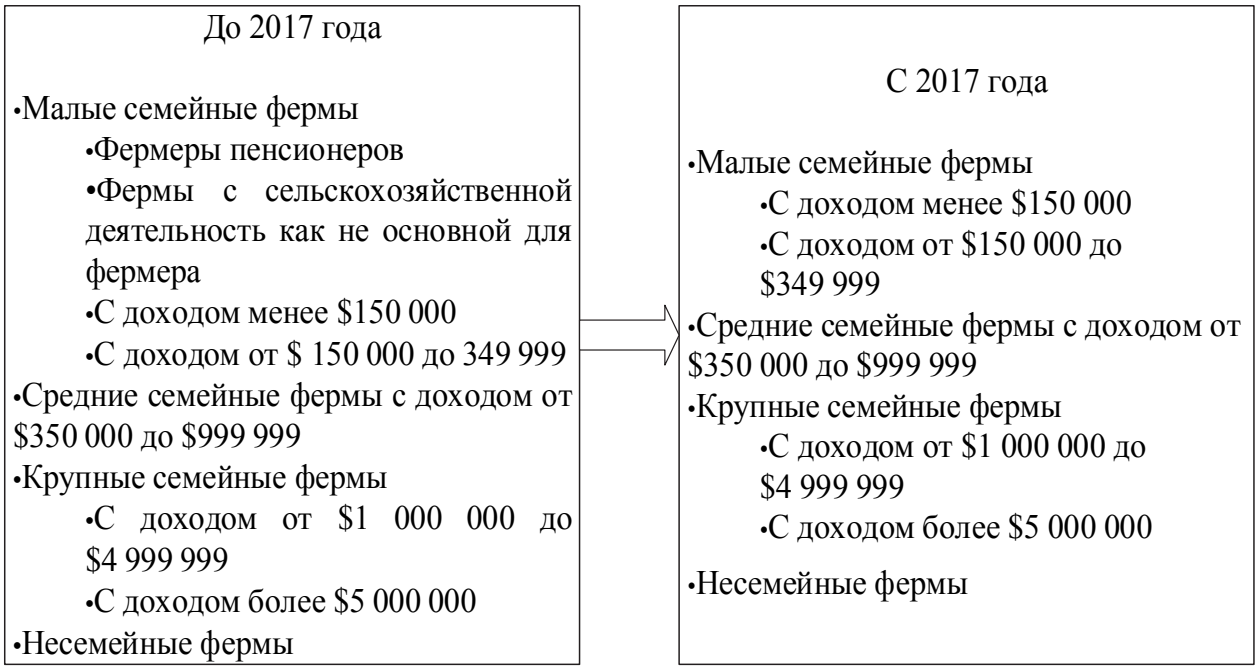


Рисунок 2.1 – Изменение типологии семейных ферм после переписи 2017 г.

Система показателей, которой характеризуется типология, содержит 29 разделов (землепользование, производство скота, птицы, занятость и др.). Представляемые результаты дают наиболее обширную характеристику сельскохозяйственных производителей в Соединенных Штатах, что позволяет дать их полную количественную и качественную оценку, в отличие от России. Результаты последней Всероссийской сельскохозяйственной переписи 2016 г. (ВСХП-2016) содержат только лишь информацию

о ресурсном и производственном потенциале предприятий. Согласно Федеральному закону «О развитии малого и среднего предпринимательства в Российской Федерации», сельскохозяйственные организации в России делятся на малые, средние и крупные (для крестьянских (фермерских) хозяйстве и личных подсобных хозяйств типология отсутствует), однако пороговые значения для этих типов сильно завышены, что относит большинство организаций к малым.

2.2 Разработка типологии сельскохозяйственных организации на основе результатов ВСХП – 2016 в Липецкой области

Для выделения типов сельскохозяйственных организаций по результатам переписи можно использовать различные методы многомерного анализа, одним из которых является кластерный анализ. Кластерный анализ может быть реализован с использованием метода k-средних, но при этом число кластеров должно быть уже известно.

Метод k-средних включает следующие этапы:

1) назначается количество групп (k), на которые должны быть разделены данные. K объектов исходного набора случайным образом выбираются в качестве начальных центров кластеров;

2) каждому наблюдению присваивается номер группы на основе ближайшего центроида, т.е. на основе наименьшего евклидова расстояния между объектом и точкой C_k;

3) пересчитываются координаты центроидов μ_k всех k кластеров и вычисляются внутригрупповые разбросы:

$$W(C_k) = \sum\left(x_i - \mu_k\right)^2 ; \quad (2.2)$$

4) общий внутригрупповой разброс

$$W_{total} = \sum W(C_k) \quad (2.3)$$

минимизируется, для чего шаги 2 и 3 повторяются до тех пор, пока назначения групп не перестанут меняться или не будет достигнуто указанное количество итераций [5].

Для апробации методики были выбраны сельскохозяйственные организации Липецкой области, так как она является типичным представителем регионов с активно развивающимся сельским хозяйством в настоящее время. В области работают крупные современные сельскохозяйственные

организации такие, как ООО «Лебедянь молоко», ООО «Авангард Агро Липецк», свиноводческий комплекс «Черкизово» и др. [15].

По размерам ресурсного потенциала Липецкая область занимает 23 место в рейтинге регионов, по размеру валовой добавленной стоимости сельского хозяйства – 22 место, а по доли валовой добавленной стоимости в валовом региональном продукте – 23 (Рисунок 2.2) [4].

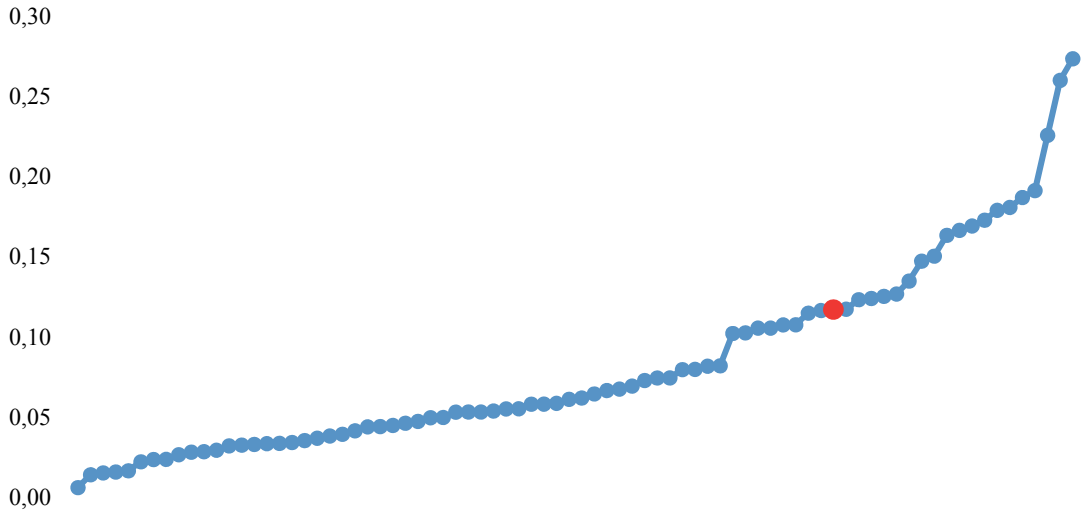


Рисунок 2.2 – Место Липецкой области в ранжированном ряду доли ВДС в ВРП Региона, 2020 г.

Результаты Всероссийской сельскохозяйственной переписи 2016 г. содержат большой набор данных для оценки размеров и специализации сельхозорганизаций, КФХ и ЛПХ: среднегодовая численность работников, площадь сельскохозяйственные угодий (в разрезе по их видам), посевные площади сельскохозяйственных культур, поголовье животных, наличие сельскохозяйственной техники, а также информация о кредитах и государственных субсидиях [16].

Для анализа и разработки новой типологии использовались следующие факторы:

1. Среднегодовая численность работников, чел.
2. Площадь сельскохозяйственных угодий, га.
3. Поголовье сельскохозяйственных животных, условные головы.
4. Наличие сельскохозяйственной техники, шт.

Проблема определения необходимого количества кластеров может быть решена на основе анализа дендрограммы объединения,

в Липецкой области выделяются три кластера сельскохозяйственных организаций (Рисунок 2.3).

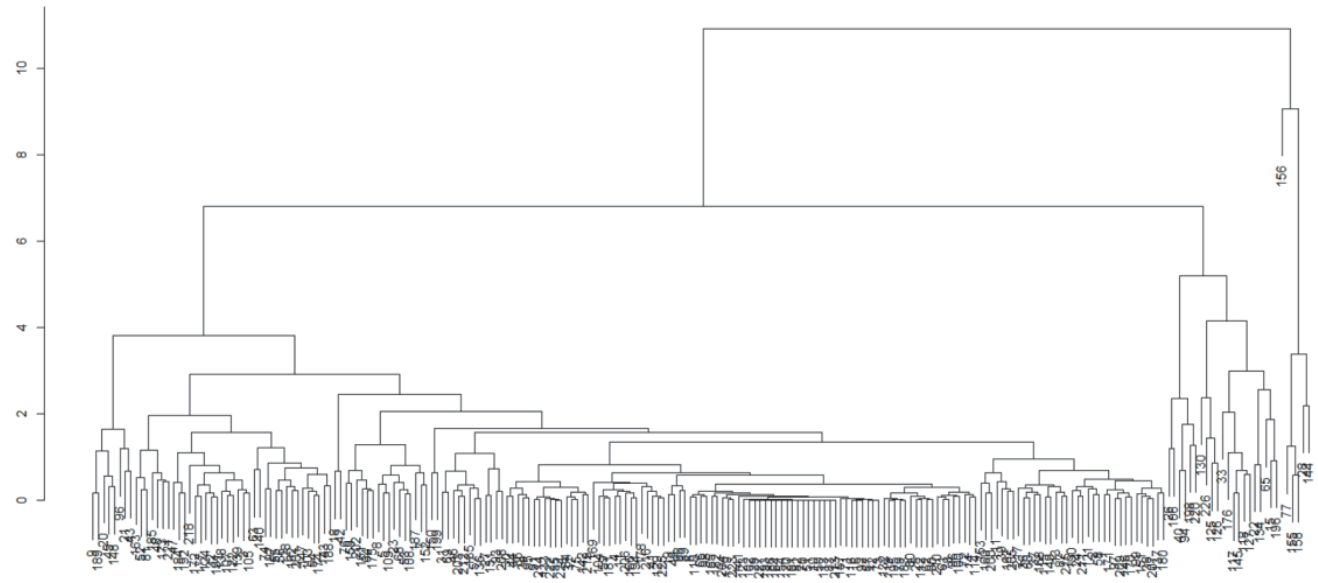


Рисунок 2.3 – Дендрограмма объединения

Однако, рассчитав суммы квадратов расстояний внутри кластеров и отобразив их на графике, можно выделить и 4 кластера (Рисунок 2.4).

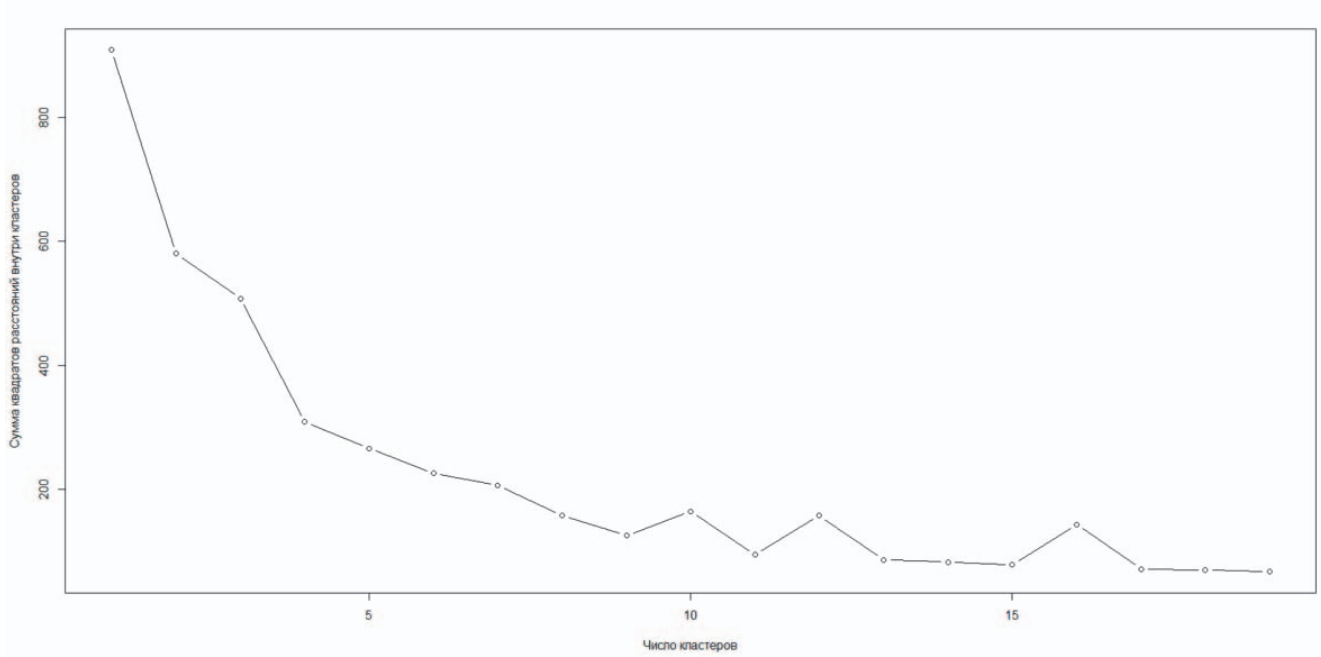


Рисунок 2.4 – Определение количества кластеров

При разбиении совокупности на 4 кластера различия по всем факторным признакам являются существенными по результатам дисперсионного анализа (Таблица 2.2), при выделении только 3

кластеров различия по среднегодовой численности работников не достоверны. Было принято решение выделить 4 кластера организаций методом k-средних.

Таблица 2.2 – Результаты дисперсионного анализа

Показатель (в расчете на 1 организацию)	3 Кластера	4 Кластера
Среднегодовая численность работников, чел.	0,99	0,0004
Площадь сельскохозяйственных угодий, га	$< 2 \cdot 10^{-16}$	$7{,}26 \cdot 10^{-8}$
Условное поголовье животных, гол.	$< 2 \cdot 10^{-16}$	$2{,}85 \cdot 10^{-5}$
Наличие сельскохозяйственной техники, шт.	$1{,}74 \cdot 10^{-4}$	0,0008

Показатели размеров и интенсификации организаций каждого из 4 кластеров (типов) приведены в таблицах 2.3 и 2.4.

Таблица 2.2 – Показатели размеров (в расчете на 1 организацию)

Показатели	Типы организаций			
	1	2	3	4
Число организаций	151	52	14	10
Среднегодовая численность работников, чел.	48	133	283	359
Площадь сельскохозяйственных угодий, га	2112	9721	31118	304
Из них: пашня	1830	8711	27273	284
сенокосы	111	279	723	-
пастбища	135	641	2621	21
многолетние насаждения	30	70	148	-
Условное поголовье, гол	636	755	560	26828
Поголовье животных, гол.:				
крупный рогатый скот	179	690	538	151
свиньи	1421	338	-	29329
овцы	3	110	320	22
птица	9153	-	-	879314
Наличие сельскохозяйственной техники, ед.	179	22	538	152
Доля организаций, получавших:				
кредитные средства	0,34	0,62	0,86	0,80
субсидии за счет средств федерального бюджета	0,63	0,90	0,86	1,00

Классификация была проведена по совокупности из 227 предприятий. Выделены 4 типа сельскохозяйственных организаций: 1 – малые, 2 – средние, 3 – крупные предприятия, специализирующие на растениеводстве и молочном скотоводстве, и 4 – крупные предприятия со специализацией на отраслях интенсивного животноводства (свиноводство и птицеводство). Дисперсионный анализ подтвердил существенность различий между выделенными типами (кластерами) на уровне значимости менее 0,1% по всем признакам, на основе которых произведена классификация. В 3 кластер вошло 14, в 4 – 10 организаций, возможно агрохолдингов. Площадь сельскохозяйственных угодий в расчете на 1 организацию 3 типа превышает 30 тысяч га, в 4 кластере поголовье свиней в расчете на 1 организацию достигает 5 тыс. голов, птицы – свыше 20 тыс. голов. Во 2 кластер вошли средние по размерам ресурсов предприятия: площадь сельскохозяйственных угодий в расчете на 1 предприятие составляет почти 10 тыс. га, поголовье крупного рогатого скота – около 700 голов, что свидетельствует о такой же специализации, как у предприятий 3 типа. Две трети сельскохозяйственных организаций Липецкой области являются малыми по размерам: со средней численностью работников около 50 человек, площадью сельскохозяйственных угодий – около 2000 га, у данного типа предприятий ограничен доступ к кредитным ресурсам и государственной поддержке. Только 30% малых организаций получали кредиты и 60% – субсидии за счет средств федерального бюджета, по сравнению с 70 и 90 % соответственно по крупным и средним организациям.

С увеличением размеров организаций среднегодовая численность работников в расчете на 100 га сельскохозяйственных угодий уменьшается, а обеспеченность техников растет, что свидетельствует о замене живого труда овеществленным и о более высокой производительности труда на крупных предприятиях (Таблица 2.3).

Таблица 2.3 – Показатели интенсификации организаций (в расчете на 100 га сельскохозяйственных угодий)

Показатель	Номер кластера			
	1	2	3	4
Среднегодовая численность работников, чел.	2,2	1,3	0,9	0,4
Условное поголовье, гол	0,3	7,7	1,7	3592,5
Поголовье животных, гол:				
крупный рогатый скот	0,0	7,1	1,7	20,2
свиньи	0,6	3,4	-	3927,4
овцы	0,0	1,1	1,0	2,9
птица	4,3	-	-	117750,5
Обеспеченность сельскохозяйственной техникой, ед.	0,0	0,2	0,2	0,8
Коэффициент распаханности	0,87	0,90	0,88	0,97

Показатели плотности поголовья подтверждают сделанные ранее выводы о специализации типов организаций.

2.3 Разработка информационной системы для автоматизации процесса типизации сельскохозяйственных организаций

Для удобства работы с любой программой необходима разработка интерфейса, создание которого происходило с использованием библиотеки Shiny – это пакет языка R для создания десктопных и веб-приложений.

Процессу проектирования информационной системы всегда предшествует ее визуальное моделирование.

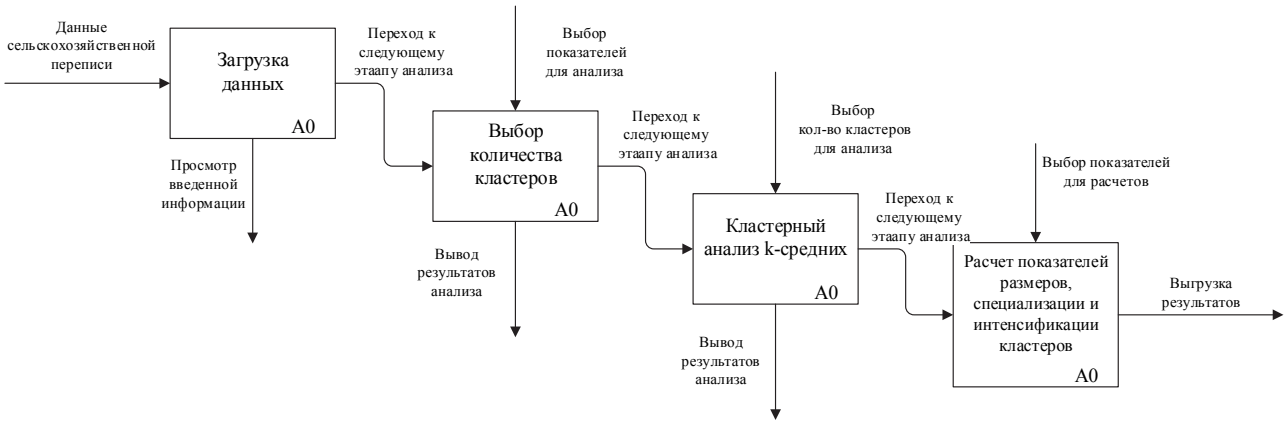


Рисунок 2.7 – Диаграмма первого уровня декомпозиции

Методология IDEF0, составляющей частью которой являются набор правил и процедур, предназначенных для создания функциональной модели объекта какой-либо предметной области, позволяет представить исчерпывающую информацию об движении информационных поток в разрабатываемой системе. На рисунке 2.7

представлена диаграмма в соответствии с нотацией IDEF0, описывающая движение информационных потоков, этапов работы информационной системы и уровни ее декомпозиции.

Диаграмма IDEF3 (Рисунок 2.8) используется для описания взаимодействия информационных потоков функции, где каждый уровень декомпозиции завершается «перекрёстком», который регламентирует следующие действие. Например, следующее действие не может быть совершено, пока не будут выполнены все предыдущие или переход к следующему действию возможен, если было выполнено хотя бы одно действие из предыдущих, или переход к следующему действию возможен только в том случае, если должно быть выполнено только одно действие из предыдущих и т.д.

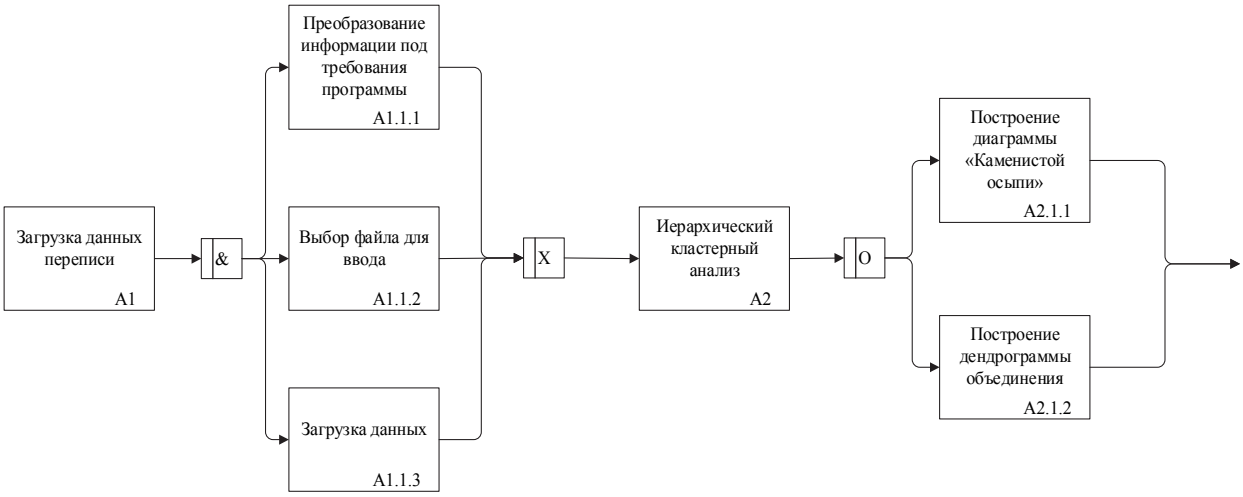


Рисунок 2.8 – Модель IDEF3

Операции первого уровня декомпозиции регламентируются следующими требованиями: все процессы должны быть запущены, но для перехода к следующему уровню должен быть завершен хотя бы один процесс. Операции второго уровня декомпозиции регламентируются так: хотя бы один из процессов должен быть запущен и хотя бы один – завершен.

Принцип работы системы можно увидеть на схеме (Рисунок 2.9).

В разрабатываемой системе анализ происходит в четыре этапа.

Первый этап – подготовительный. Необходимо подготовить входную информацию к требованиям системы.

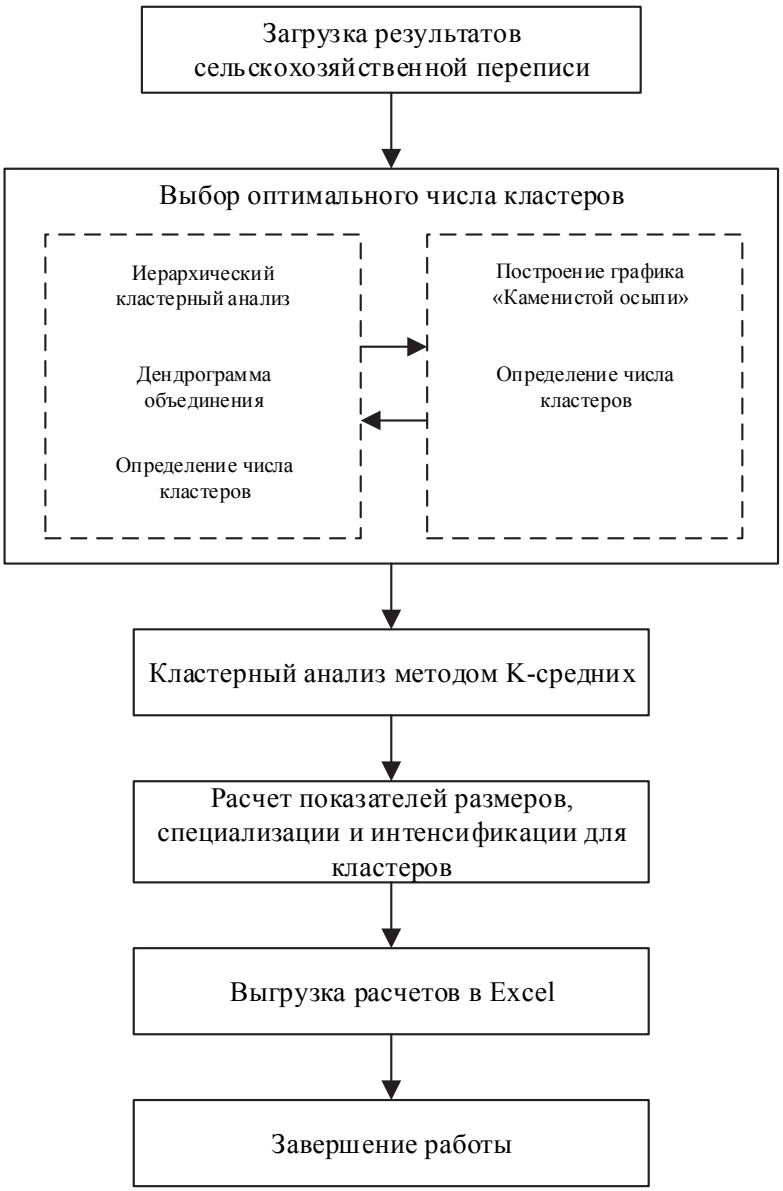


Рисунок 2.9 – Схема алгоритма автоматизации процесса типизации сельскохозяйственных организаций

Второй этап – выбор количества кластеров путем построения графика «Каменистой осыпи» или проведением иерархического кластерного анализа с выведением дендрограммы объединения. Результатом иерархического анализа является дендрограмма или схема объединения/разъединения (Рисунок 2.10).

Третий этап – проведение кластерного анализа методом k-средних.

И наконец, последний, четвёртый шаг – расчет показателей размеров, специализации и интенсификации для каждого кластера с последующей выгрузкой в Excel.

В результате реализации идеи получаем приложение, которое имеет следующий интерфейс, представленный на рисунке 2.11.

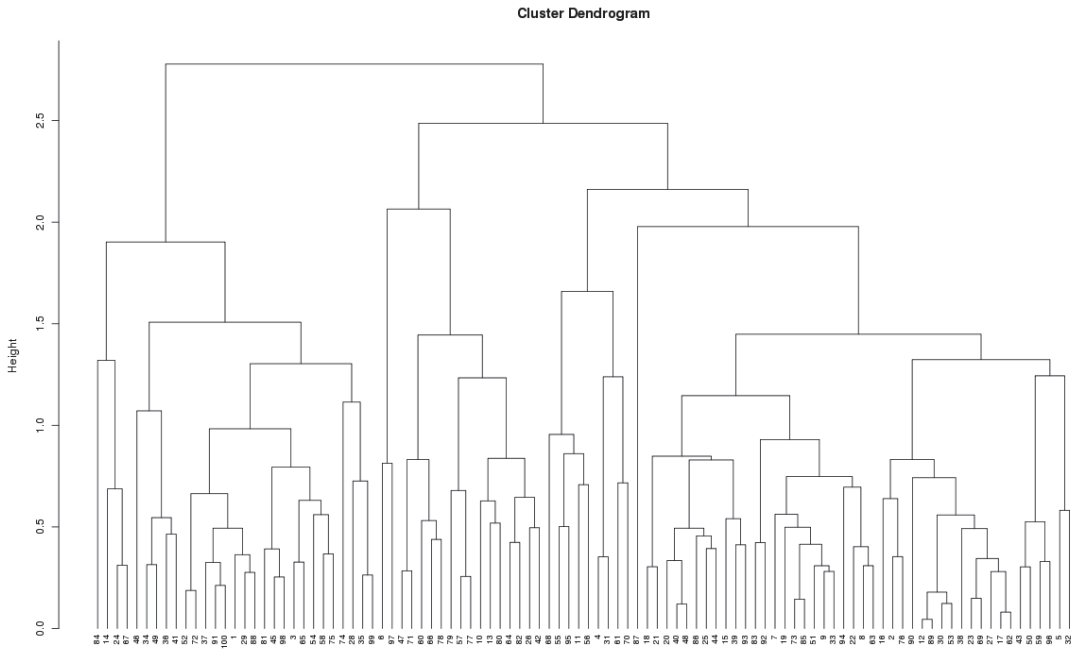


Рисунок 2.10 – Дендрограмма объединения

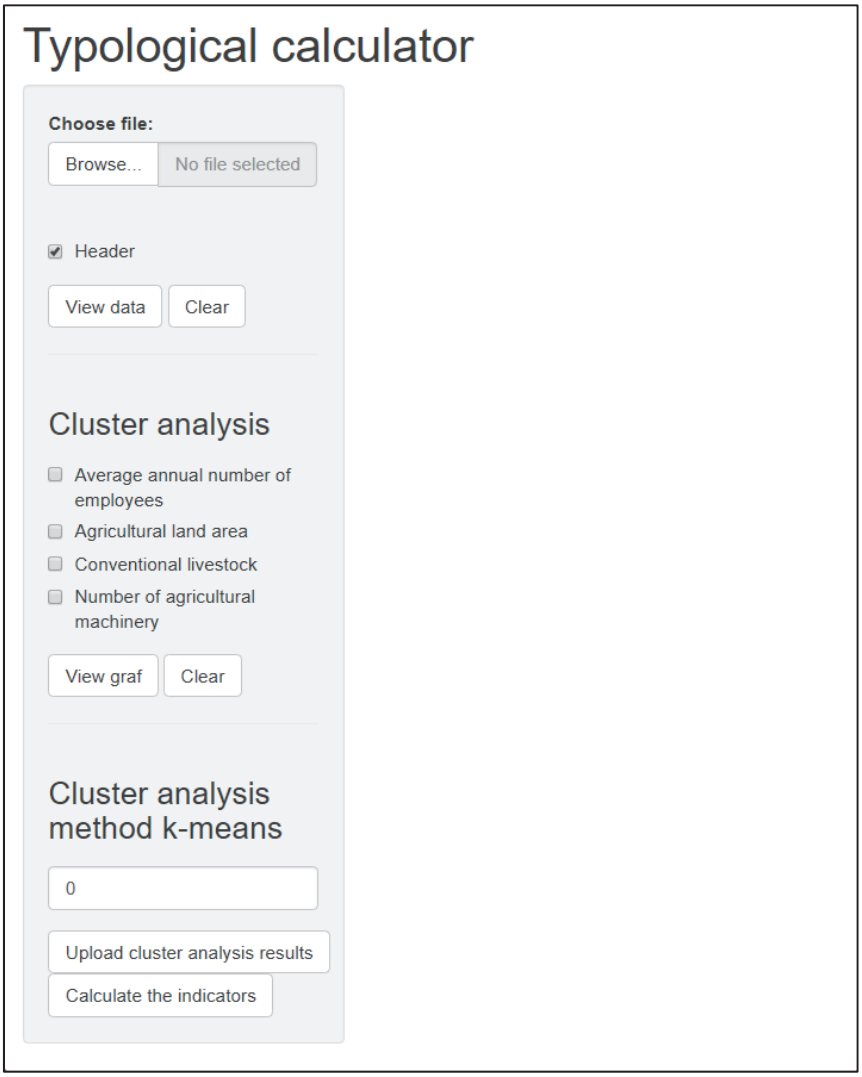


Рисунок 2.113 – Интерфейс приложения

Интерфейс приложения состоит из двух областей, боковая панель (слева) предлагает способы анализа, а основная панель (справа) служит для вывода результатов.

Так при загрузке исходных данных открывается окно выбора файла, после чего выбранный файл загружается, и его можно вывести на экран соответствующей кнопкой (Рисунок 2.12) и также после просмотра его можно убрать.

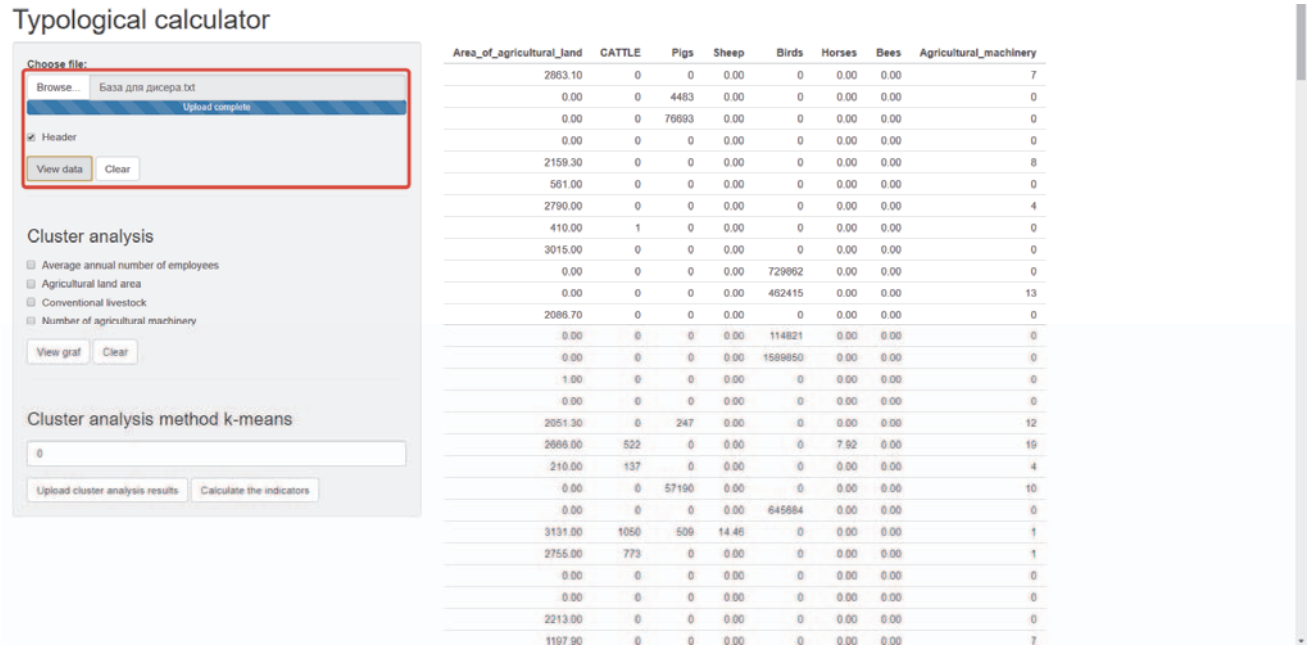


Рисунок 2.12 – Загрузка данных

После загрузки данных можно выполнить первый шаг анализа, а именно, определиться с количеством кластеров (Рисунок 2.13).

В соответствующей части меню необходимо выбрать показатели, которые будут участвовать в проведении анализа, а затем вывести результаты, построенную дендрограмму объединения и график «Каменистой осыпи»

После того, как определили оптимальное количество кластеров, можно проводить кластеризацию методом k-средних (Рисунок 2.14).

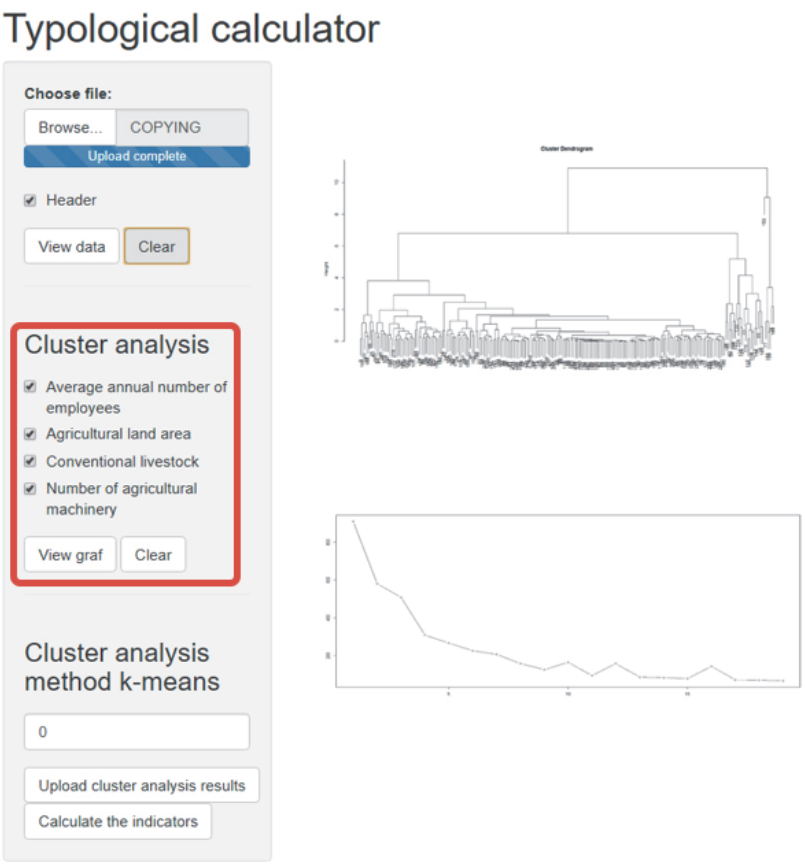


Рисунок 2.13 – Результаты проведение иерархического кластерного анализа

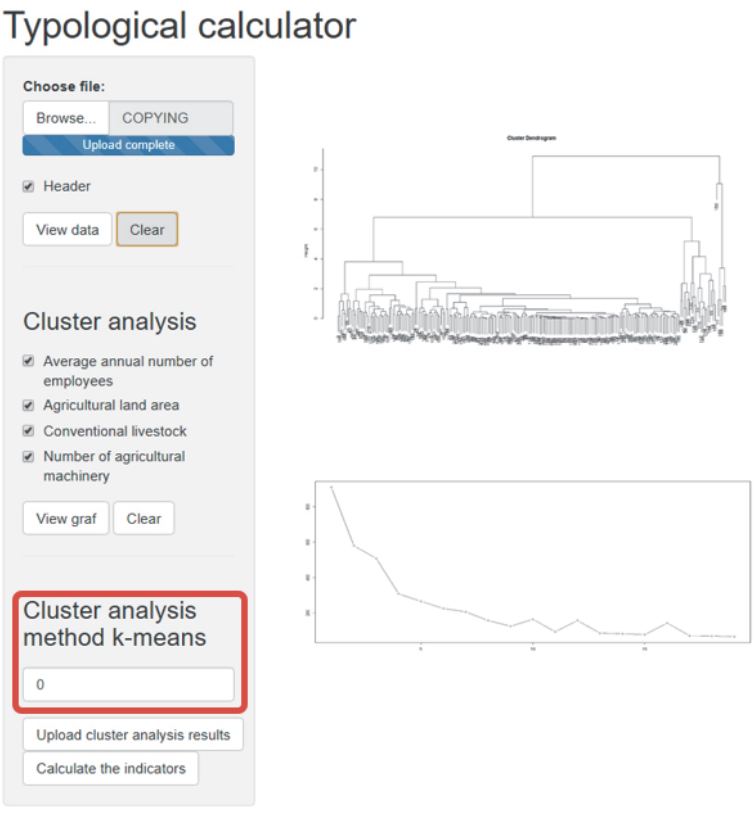


Рисунок 2.14 – Результаты кластеризации методом k-средних

И наконец, для выделенных кластеров рассчитываются показатели размеров, специализации и интенсификации, и выгружаются в Excel (Рисунок 2.15).

Показатели	Типы организаций			
	1	2	3	4
Количество организаций	151	52	14	10
Среднегодовая численность работников, чел.	48	133	283	359
Площадь сельскохозяйственных угодий, га	2112	9721	31118	747
Из которых: пахотные земли	1830	8711	27273	726
сенокосы	111	279	723	-
пастбища	135	641	2621	21
многолетние насаждения	30	70	148	-
Условное поголовье скота, голов	636	755	560	26828
Численность животных, голов				
крупный рогатый скот	179	690	538	151
свиньи	1421	338	-	29329
овцы	3	110	320	22
домашняя птица	9153	-	-	879314
Количество сельскохозяйственной техники, единиц	7,60	21,70	55,90	5,70
Процент организаций, получающих:				
кредитные средства	0,34	0,62	0,86	0,8
субсидии из федерального бюджета	0,63	0,9	0,86	1

Показатели	Типы организаций			
	1	2	3	4
Среднегодовая численность работников, чел.	2,3	1,4	0,9	48,1
Условное поголовье скота, голов	30,1	7,7	1,7	3592,5
Численность животных, голов				
крупный рогатый скот	8,5	7,1	1,7	20,2
свиньи	67,3	3,4	-	3927,4
овцы	0,1	1,1	1,0	2,9
домашняя птица	433,4	-	-	117750,5
Наличие сельскохозяйственной техники, единиц	0,36	0,22	0,18	0,76
Коэффициент распаханности	0,87	0,90	0,88	0,97

Рисунок 2.15 – Выгрузка показателей размеров, специализации и интенсификации

Пакет Shiny предлагает большое количество виджетов [29], они располагаются на боковой панели, например, виджет с выбором нескольких вариантов, как в иерархическом кластерном анализе, описывается кодом, представленном на рисунке 2.16. Виджет включает в себя следующие атрибуты: название, которое будет использоваться для связи с этим виджетом, надпись, будет видна в самом приложении над виджетом, сами элементы выбора и выбранный элемент. Кроме этих атрибутов есть еще некоторые, с помощью которых можно настроить виджет более детально [31].

```
checkboxGroupInput ("indicators",
            label = h3("Cluster analysis"),
            choices = list("Average annual number of employees" = 1, "Agricultural land area" = 2, "Conventional livestock" = 3, "Number of agricultural
            selected = 0
           ),
```

Рисунок 2.16 – Программирование виджета

Аналогично настраиваются все остальные виджеты. Весь основной анализ происходит в серверной части приложения, где описываются статистические методы обработки информации. Например, иерархический кластерный анализ представлен на рисунке 2.17, в зависимости от того, какая группа показателей

выбрана, делается выборка, затем проводится сам анализ по указанному методу и строится дендрограмма.

```
if (vibor == 1 ) ({
  library(cluster)
  country <- tab[seq(1,nrow(tab),3),]
  country.dist <- daisy(country[,2:8])
  country.h <- hclust(country.dist, method="ward")
  plot(country.h, labels=abbreviate(country[,1],1, method="both.sides"))
})
```

Рисунок 2.17 – Программирование иерархического кластерного анализа

Листинг процедуры кластерного анализа методом k-средних представлен на рисунке 2.18, он проводится по всем группам показателей, с учетом количества центров выбранным пользователем.

```
dat1<-as.data.frame(scale(tab[, 2:29]))
clust<-kmeans(dat1[, 2:8], centers = as.numeric(vibor), iter.max = 10, nstart = 20)
```

Рисунок 2.18 – Программирование кластерного анализа методом k-средних

Выводы. В рамках данного исследования была решена задача разработки новой типологии сельскохозяйственных организаций Липецкой области на основе результатов Всероссийской сельскохозяйственной переписи 2016. Были получены следующие результаты:

– были рассмотрены теоретические основы проведения сельскохозяйственной переписи в России и за рубежом (США, Европа), выявлены недостатки в методологии определения типов сельскохозяйственных организаций;

– разработан методический подход, позволяющий более точно определить типы сельскохозяйственных организаций Липецкой области и включающий в себя следующие этапы: определение системы показателей, проведение кластерного анализа методом k-средних, оценка качества модели посредством дисперсионного анализа, формирование таблиц, характеризующих типы сельскохозяйственных организаций по размерам, специализации и интенсификации производства, описание кластеров;

– обоснован выбор средства реализации анализа. Язык высокоуровневого программирования R соответствует требованиям

к доступности, достаточному набору функций для статистического анализа данных и возможности визуализации информационной системы;

– обоснован выбор региона (Липецкой области), как наиболее типичного с развивающимся сельским хозяйством и высокой долей валовой добавленной стоимости сельского хозяйства в валовом региональном продукте;

– проведена типизация сельскохозяйственных организаций Липецкой области на основе результатов Всероссийской сельскохозяйственной переписи 2016, были выделены 4 кластера: 1 – малые, 2 – средние, 3 – крупные предприятия, специализирующие на растениеводстве и молочном скотоводстве, и 4 – крупные предприятия со специализацией на отраслях интенсивного животноводства (свиноводство и птицеводство). Агрохолдинги специализируются на разведении сельскохозяйственной птицы и свиней, крупные предприятия имеют смешанный тип, с большим поголовьем КРС и площадью пашни, средние организации, также смешанные по типу, с поголовьем КРС и высоким коэффициентом распаханности, малые организации характеризуются большим поголовьем КРС, свиней и птицы, а также площадью сельскохозяйственных угодий.

– результаты кластерного анализа были признаны значимыми на основе дисперсионного анализа, и данная методика может быть применена в будущем при разработке типологии сельскохозяйственных организаций в целом по России.

Библиографический список к главе 2

1. Вендров, А. М. Проектирование программного обеспечения экономических информационных систем. 2-е изд., перераб. и доп. – М.: Финансы и статистика, 2015. – 180 с. – Текст: непосредственный.

2. Всероссийская сельскохозяйственная перепись 2016 года. Предварительные итоги: Статистический бюллетень / Федеральная служба государственной статистики. – М: ИИЦ «Статистика России», 2016. – 70 с. – Текст: непосредственный.

3. Гатаулина, Е.А. Сравнительный анализ аграрных структур России и США / Е.А. Гатаулина // Отечественные записки. – 2012. – №6 (51). – с. 134-158. – Текст: непосредственный.

4. Зинченко А.П. Региональная и муниципальная статистика: практикум / А.П. Зинченко, В.В. Демичев. – М.: РГАУ-МСХА имени К.А. Тимирязева, 2016. – 88 с. – Текст: непосредственный.

5. Зинченко, А.П. Математическая статистика / А.П. Зинченко, М.В. Кагирова, Ю.Н. Романцева, О.Б. Тарасова, А.В. Уколова, Т.Ф. Хромова, А.Е. Шибалкин: учебник. – М.: Изд-во РГАУ-МСХА им. К.А. Тимирязева, 2018. – 199 с. – Текст: непосредственный.

6. Кабаков, Р. И. R в действии. Анализ и визуализация данных в программе R / пер. с англ. П. А. Волновой. – М.: ДМК Пресс, 2014. – 588 с. – Текст: непосредственный.

7. Кагирова, М.В. Пути совершенствования информационного обеспечения анализа деятельность сельскохозяйственных производителей / М.В. Кагирова, Е.С. Коломеева // Международный технико-экономический журнал. – 2013. – №6. – с. 34-38. – Текст: непосредственный.

8. Кагирова, М.В. Статистический анализ развития цифровой экономики России / М.В. Кагирова // Экономика сельского хозяйства России. – 2019. – №3. – с. 27-29– Текст: непосредственный.

9. Кагирова, М.В. Статистическое изучение региональных особенностей производства продукции животноводства в РФ / М.В. Кагирова // Сборник докладов I Открытого российского статистического конгресса. Российская ассоциация статистиков; Федеральная служба государственной статистики РФ; Новосибирский государственный университет экономики и управления "НИНХ", 2016. – с. 425-430. – Текст: непосредственный.

10. О Всероссийской сельскохозяйственной переписи: Федеральный закон от 21.07.2005 N 108-ФЗ. – Доступ из справочно-правовой системы «Гарант». – URL: https://base.garant.ru/12141167/ (дата обращения 15.11.2022). – Текст: электронный.

11. О развитии малого и среднего предпринимательства в Российской Федерации: Федеральный закон от 24.07.2007 N 209-

ФЗ. – Доступ из справочно-правовой системы «Гарант». – URL: https://base.garant.ru/12154854/ (дата обращения 15.11.2022). – Текст: электронный.

12. Об организации Всероссийской сельскохозяйственной переписи 2016 года: Постановление Правительства РФ от 10.04.2013 N 316 (ред. от 06.03.2015). – Доступ из справочно-правовой системы «Гарант». – URL: https://base.garant.ru/70358376/ (дата обращения 15.11.2022). – Текст: электронный.

13. Об утверждении основных методологических и организационных положений по подготовке и проведению сельскохозяйственной микропереписи 2021 года: Приказ Росстата от 30 ноября 2020 года №74. – Доступ из справочно-правовой системы «Гарант». –URL: https://base.garant.ru/74985581/

14. Петриков, А. В. Всероссийская сельскохозяйственная перепись 2016 года как источник предложений по совершенствованию аграрной и сельской политики и стимулированию развития различных категорий хозяйств / А.В. Петриков // Московский экономический журнал. – 2018. – №3. – 258 с. – Текст: непосредственный.

15. Пинская, М.Р. Государственная финансовая поддержка АПК: налоговые и бюджетные инструменты: монография / М.Р. Пинская, А.В. Тихонова. – М.: Издательский Дом «Инфра-М», 2016. –178 с. – Текст: непосредственный.

16. Проблемы экономической и сельскохозяйственной статистики: монография / А.П. Зинченко, В.М. Баутин, А.Д. Думнов, Н.А. Эльдяева, А.В. Уколова, Ю.Н. Романцева, Е.С. Кованова, А.В. Тихонова, А.Е. Харитонова, Д.В. Дзюба, А.В. Сергеев. – Иркутск: Изд-во ООО «Мегапринт», 2017. – 161 с. – Текст: непосредственный.

17. Регламент Европейского Парламента и Европейского Совета №1166/2008 о структурных обследованиях в сельском хозяйстве и обследовании методов сельскохозяйственного производства. – Доступ из справочно-правовой системы «Гарант». – URL: https://base.garant.ru/71003992/ (дата обращения 15.11.2022). – Текст: электронный.

18. Романцева, Ю.Н. Анализ технической обеспеченности сельскохозяйственных производителей в России / Ю.Н. Романцева

// Экономика сельского хозяйства России. – 2019. – №3. – 19-24 с. – Текст: непосредственный.

19. Романцева, Ю.Н. Основные тенденции развития сельского хозяйства России в условиях цифровой трансформации / Ю.Н. Романцева // Сборник Материалов I Международной научно-практической конференции по проблемам развития аграрной экономики, 2020. – 535-538 с. – Текст: непосредственный.

20. Сельскохозяйственная микроперепись. Оперативные итоги: Федеральная служба государственной статистики. – URL: https://rosstat.gov.ru/storage/mediabank/oper_itogi_SXMP_2021.pdf (дата обращения 25.09.2022). – Текст: электронный.

21. Современные проблемы статистики сельского хозяйства и окружающей природной среды: Монография / А.П. Зинченко, В.М. Баутин, А.Д. Думнов, С.А. Скачкова, А.В. Уколова, М.В. Кагирова, и др. – М.: Изд-во РГАУ – МСХА, 2016. – 198 с. – Текст: непосредственный.

22. Тихонова, А.В. Государственная финансовая поддержка сельскохозяйственной кооперации: куда смотреть - на запад или "под ноги"? / А.В. Тихонова // Бухучет в сельском хозяйстве, 2016. - № 2. - С. 51-64.

23. Уколова, А.В. Сводка и анализ данных сельскохозяйственных переписей: опыт США / А.В. Уколова // Материалы I Открытого российского статистического конгресса, 2015. – с. 173-174. – Текст: непосредственный.

24. Уколова, А.В. Типизация сельскохозяйственных предприятий Германии / А.В. Уколова // Доклады ТСХА: сб. статей. – Вып. 291. – Ч. IV. – М.: Изд-во РГАУ-МСХА, 2019. – с. 344-349. – Текст: непосредственный.

25. Харитонова, А.Е. Дифференциация регионов по показателям эколого-экономического состояния и развития сельского хозяйства / А.Е. Харитонова // Вопросы статистики, 2018. – Т. 25. – № 10. – с. 37-46. – Текст: непосредственный.

26. Харитонова, А.Е. Статистический анализ и прогнозирование с использованием пакетов прикладных статистических программ: практикум / А.Е. Харитонова. М.: Изд-во РГАУ-МСХА им. К.А. Тимирязева, 2015. – 155 с. – Текст: непосредственный.

27. Census of agriculture//Farm typology / National Agricultural Statistics Service. – Volume 2. – Special Studies. – Part 10. – p. 706. – Текст: непосредственный.

28. Glossary: Agricultural holding. – URL: https://ec.europa.eu/eurostat/statistics-explained/index.php?title=Glossary:Agricultural_holding (дата обращения 23.10.2022). – Текст: электронный.

29. Shiny for RStudio. – URL: https://shiny.rstudio.com/ (дата обращения 03.09.2022). – Текст: электронный.

30. Small and large farms in the EU - statistics from the farm structure survey. – URL: https://ec.europa.eu/eurostat/statistics-explained/index.php?title=Small_and_ large_farms_in_the_EU_-_statistics_from_the_farm_structure_survey &oldid=357625 (дата обращения 17.09.2022). – Текст: электронный.

31. The R Manuals. – URL: Режим доступа: https://cran.r-project.org/ (дата обращения 17.10.2022). – Текст: электронный.

Глава 3 Разработка информационной системы учета и обработки данных с поддержкой проведения статистического анализа на C++

В современном мире цифровые технологии внедряются во все сферы человеческой жизни. Широко применяются различные информационные системы, в том числе с целью сокращения временных затрат на решение задач по учету и анализу данных больших объемов.

В данной главе рассмотрен вопрос разработки информационной системы (ИС) учета и обработки данных с поддержкой проведения статистического анализа. Указанная ИС состоит из трех модулей: модуль подключения к любой базе данных MS SQL Server для возможности работать с данными из разных баз данных; модуль проведения корреляционно-регрессионного анализа для оценки связей между переменными и проведения прогноза; модуль построения нейронных сетей для проведения классификации.

Используемые в исследовании программные средства: Microsoft Excel, Microsoft Visio, Microsoft SQL Server 2019, Qt Creator (C++). Для разработки базы данных информационной системы используется язык T-SQL, среда разработки Microsoft SQL Server 2019, пользовательского интерфейса информационной системы – язык программирования C++, библиотека Qt для языка C++, среда разработки Qt Creator.

Теоретическая значимость исследования заключается в разработке подходов и алгоритмов, позволивших реализовать информационную систему учета и анализа данных, практическая значимость – в возможности использовать разработанную информационную систему для целей учета и анализа данных.

3.1 Теоретические основы проведения корреляционно-регрессионного анализа

Для оценки параметров множественной регрессии используем метод обратной матрицы [4]. Исходные данные представим в виде двух матриц Y и X. Матрица Y будет состоять из одного столбца, матрица X из $n + 1$ столбцов, где n – число факторов, причем первый столбец матрицы X – единичный вектор.

Множественное линейное уравнение регрессии имеет вид:

$$\tilde{y}_x = b_0 + b_1 x_1 + \cdots + b_i x_i \,, \tag{3.1}$$

где b_0 – параметр условное начало,
b_i – параметры чистой регрессии при независимой переменной x_i.

Параметры (коэффициенты) множественного линейного уравнения в виде вектора-столбца B находятся по формуле:

$$B = (X'X)^{-1}X'Y \,, \tag{3.2}$$

где X' – транспонированная матрица X,
$X'X$ – скалярное произведение матриц X' и X,
$(X'X)^{-1}$ – обратная матрица матрицы $X'X$.

По полученным коэффициентам, подставив их в множественное линейное уравнение, найдем прогнозные значения y, представив их в виде матрицы $\tilde{Y}$. Ошибки регрессии в виде матрицы E найдем по формуле:

$$E = Y - \tilde{Y} \tag{3.3}$$

Найдем объемы вариации (Q_{total} – общий, Q_{error} – остаточный, $Q_{regr.}$ – по регрессии):

$$Q_{total} = (Y - \bar{y})'(Y - \bar{y}) \tag{3.4}$$
$$Q_{error} = E'E \tag{3.5}$$
$$Q_{regr.} = Q_{total} - Q_{error} \tag{3.6}$$

Найдем множественный коэффициент детерминации:

$$R^2 = \frac{Q_{regr.}}{Q_{total}} = 1 - \frac{Q_{error}}{Q_{total}} \tag{3.7}$$

Проверка достоверности уравнения в целом проводится на основе дисперсионного анализа по критерию F-Фишера [10]. Найдем степени свободы [5]:

$$df_{total} = n - 1 \,, \tag{3.8}$$

где n – число наблюдений.

$$df_{error} = n - m - 1 \,, \tag{3.9}$$

где m – число факторов.

$$df_{regr.} = m \tag{3.10}$$

Далее найдем фактическое значение критерия F-Фишера:

$$F = \frac{R^2}{1 - R^2} \cdot \frac{df_{error}}{df_{regr.}} \tag{3.11}$$

Фактическое значение используется в качестве параметра в функции плотности распределения значений F-Фишера – интегральной (кумулятивной) функции распределения:

$$F(x; df_{regr.}; df_{error}) = I\left(\frac{df_{regr.} \cdot x}{df_{error} + df_{regr.} \cdot x}; \frac{1}{2} df_{regr.}; \frac{1}{2} df_{error}\right), \tag{3.12}$$

где I – регуляризованная неполная бета-функция:

$$I(z; a; b) = \frac{B(z; a; b)}{B(a; b)}, \tag{3.13}$$

где $B(z; a; b)$ – неполная бета-функция,
$B(a; b)$ – бета-функция.

Неполная бета-функция и бета функция находятся по следующим формулам соответственно:

$$B(z; a; b) = \int_0^z t^{a-1}(1-t)^{b-1} dt \tag{3.14}$$

$$B(a; b) = \frac{\Gamma(a)\Gamma(b)}{\Gamma(a+b)}, \tag{3.15}$$

где Γ – гамма-функция:

$$\Gamma(x) = \int_0^\infty t^{x-1} e^{-t} dt \tag{3.16}$$

Если исследователем принят уровень критической значимости 5%, то значение функции плотности распределения меньшее 0,05 свидетельствует о значимости модели в целом.

Проведем t-тест для определения качества найденных коэффициентов модели. Найдем среднее квадратическое отклонение остатка:

$$S_e = \sqrt{\frac{Q_{error}}{df_{error}}} \tag{3.17}$$

Найдем средние ошибки для коэффициентов b_i:

$$m_{b_i} = S_e\sqrt{([X'X]^{-1})_{ii}}\,, \tag{3.18}$$

где $([X'X]^{-1})_{ii}$ – ii-й элемент в главной диагонали матрицы $X'X$.

Фактические значения критерия t-Стьюдента определим по формуле:

$$t_i = \left|\frac{b_i}{m_{b_i}}\right| \tag{3.19}$$

Фактическое значение используется в качестве параметра в функции плотности распределения значений t-Стьюдента – интегральной (кумулятивной) функции распределения:

$$F(x; df_{error}) = \frac{1}{2} + \frac{1}{2}\left[I\left(1; \frac{df_{error}}{2}; \frac{1}{2}\right) - I\left(\frac{df_{error}}{df_{error} + x^2}; \frac{df_{error}}{2}; \frac{1}{2}\right)\right] sgn(x), \tag{3.20}$$

где *sgn* – кусочно-постоянная функция:

$$sgn(x) = \begin{Bmatrix} 1, & x > 0 \\ 0, & x = 0 \\ -1, & x < 0 \end{Bmatrix} \tag{3.21}$$

Для получения критического значения критерия t-Стьюдента применяют обратную интегральную (кумулятивную) функцию распределения.

$$F^{-1}(x; df_{error}) = \sqrt{df_{error}\left(\frac{1}{I^{-1}\left(\mathrm{If}\left[x < \frac{1}{2}, 2x, 2(1-x)\right]; \frac{df_{error}}{2}; \frac{1}{2}\right)}\right)} sgn(x - \frac{1}{2}) \tag{3.22}$$

Значение функции плотности распределения меньшее 0,05 свидетельствует о значимости коэффициента модели при уровне значимости 5%.

Расчет вышеуказанных интегральных функций, таких как неполная бета-функция, гамма-функция, обратная интегральная функция распределения t-Стьюдента реализуется с помощью алгоритмов, предложенных в книге [20].

Для сравнения влияния на зависимую переменную различных объясняющих переменных, когда последние выражаются разными единицами измерения, используют стандартизированные коэффициенты регрессии (бета-коэффициенты), коэффициенты эластичности, а также коэффициенты раздельной детерминации [5]:

$$\beta_i = b_i \frac{\sigma_{x_i}}{\sigma_y}, \tag{3.23}$$

где σ_y – среднее квадратическое отклонение:

$$\sigma_y = \sqrt{\frac{(y - \bar{y})^2}{n}}. \tag{3.24}$$

$$Э_i = b_i \frac{\bar{x}_i}{\bar{y}}. \tag{3.25}$$

$$d_i^2 = \beta_i \cdot r_{x_i y}, \tag{3.26}$$

где $r_{x_i y}$ – парный коэффициент корреляции:

$$r_{yx} = \frac{\overline{yx} - \bar{y} \cdot \bar{x}}{\sqrt{\overline{y^2} - \bar{y}^2} \cdot \sqrt{\overline{x^2} - \bar{x}^2}} = \frac{\overline{yx} - \bar{y} \cdot \bar{x}}{\sigma_y \cdot \sigma_x}. \tag{3.27}$$

При построении прогноза рассчитывают точечную и интервальную оценки прогноза.

Точечная оценка прогноза для генеральной совокупности заключается в расчете средней ошибки прогноза:

$$m_{y_i} = S_e \sqrt{(1 + X_n [X'X]^{-1}) X_n'}, \tag{3.28}$$

где X_n – матрица-строка значений факторов при шаге прогнозирования n.

Для интервальной оценки прогноза рассчитывают нижнюю и верхнюю границу прогноза:

$$\tilde{y}_i - m_{y_i} t_{\alpha;df\_error} \leq \dot{y}_i \leq \tilde{y}_i + m_{y_i} t_{\alpha;df\_error}\,, \qquad (3.29)$$

где $\tilde{y}_i$ – смоделированное значение зависимой переменной y,
$t_{\alpha;df\_error}$ – критическое значение распределения критерия t-Стьюдента при уровне значимости α и числе степеней свободы df_error,
$\dot{y}_i$ – фактическое значение распределения критерия t-Стьюдента в генеральной совокупности.

Для повышения качества анализа данных исходные данные стандартизируют, что позволяет уменьшить разброс значений переменных. При стандартизации данных с помощью логарифмирования используют следующую формулу:

$$x_{stand.} = log_e x\,, \qquad (3.30)$$

где $x_{stand.}$ – стандартизированное значение переменной,
x – исходное значение переменной.

Также стандартизировать данные можно путем расчета нормированного отклонения:

$$x_{stand.} = \frac{x - \bar{x}}{\sigma_x}\,, \qquad (3.31)$$

где $\bar{x}$ – среднее значение переменной,
σ_x – стандартное отклонение переменной.

3.2 Теоретические основы построения искусственных нейронных сетей

Одним из основных способов применения разработанной информационной системы является построение искусственных нейронных сетей (ИНС) для классификации объектов по ряду признаков, представленных числовыми данными в табличном виде.

Построение искусственных нейронных сетей как нейронных моделей мозга входит в одно из базовых направлений развития искусственного интеллекта [1, с. 112]. Благодаря нахождению скрытых причинно-следственных связей в массивах данных,

нейронные сети способны решать интеллектуальные задачи, для которых не существует формального алгоритма. К таким задачам относят, например, обучение, распознавание сложных образов и т.д. [6].

Чубукова И.А. дает следующее определение искусственным нейронным сетям: «Искусственные нейронные сети представляют собой модели биологических нейронных сетей мозга, в которых нейроны имитируются относительно простыми, часто однотипными, элементами (искусственными нейронами)» [9].

ИНС в наиболее общем случае классифицируют на статические и динамические. Статические ИНС не имеют обратных связей (к таковым можно отнести следующие типы нейронных сетей: персептрон, сети прямого распространения – Feedforward Neural Network, сети радиально-базисных функций – RBF-сети, адаптивный линейный нейрон – ADALINE). Динамические (рекуррентные) ИНС имеют обратные связи (например, сети Кохонена, сети Хопфилда, сети Хемминга, сети Элмана, сети адаптивного резонанса – ART-сети) [2, с. 20].

Наличие обратных связей подразумевает то, что выходной сигнал элемента системы оказывает влияние на входной сигнал этого элемента системы. То есть входные сигналы могут усиливаться сигналами внутри системы через выходные сигналы [8, с. 52].

Нейронные сети применяются для прогнозирования, распознавания образов в условиях зашумленной входной информации, адаптивного управления, оптимизации [1].

В настоящий момент опубликовано множество исследований, связанных с применением нейронных сетей для решения различных практических задач. Например, в исследовании [16] модель сети прямого распространения с одним скрытым слоем на основе гипербазисной функции (Hyper basis function network) позволяет оценивает качество городской среды, обучаясь на большем наборе данных о загрязнении воздуха в США.

В работе [14] RBF-сеть применяется для обнаружения стадий созревания бананов, а ее эффективность сравнивается с многослойным персептроном. Авторы приходят к выводу, что RBF-сеть реализует более быструю и эффективную категоризацию стадий созревания бананов, а также имеет низкие вычислительные затраты по сравнению с многослойным персептроном.

Распознавание рукописных цифр с использованием базы данных MNIST на основе глубоких сверточных нейронных сетей (Deep Convolutional Neural Networks – DCNN) и самоорганизующихся карт Кохонена (Self-organizing Maps) изучается в работе [13]. В ней авторы разрабатывают новую модель нейронный сети – глубокие сверточные самоорганизующиеся карты (Deep Convolutional Self-organizing Maps – DCSOM) с обучением без учителя для инвариантного представления изображения из немаркированных визуальных данных.

Самоорганизующиеся карты Кохонена применяются для решения задачи кластеризации. Например, в исследовании [15] при оценке пространственного и временного распространения COVID-19 в Бразилии по числу случаев заболевания и числу смертей в регионах, штатах и городах страны, кластеризация посредством SOM позволила сгруппировать города, штаты и регионы в соответствии со случаями заболевания коронавирусом, что способствует выявлению степени эффективности применяемой стратегии борьбы с распространением вируса в этих городах, штатах и регионах.

Прогнозирование скорости инфильтрации очищенных сточных вод с помощью нейронной сети Элмана и многослойного персепетрона продемонстрировано в [11]. ART-сети используют для прогнозирования изменения инженерных объектов, чтобы предвидеть их повреждение [19]. Алгоритмы кластеризации на основе теории адаптивного резонанса исследуются в [18].

Нейронная сеть может быть представлена направленным графом с взвешенными связями, в котором искусственные нейроны являются вершинами, а синаптические связи – дугами:

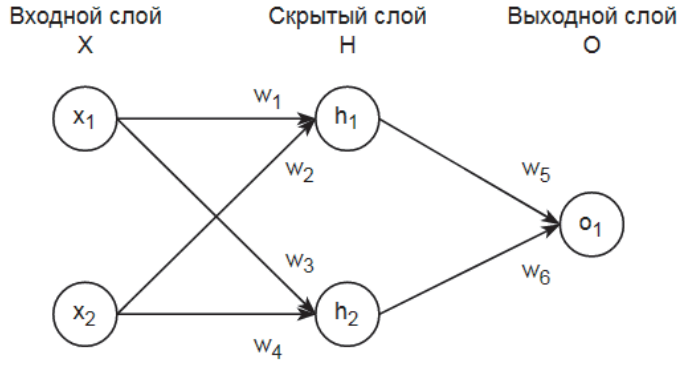


Рисунок 3.1 – Пример трехслойной нейронной сети

Нейронная сеть состоит из входного слоя X, некоторого числа скрытых слоев H_i выходного слоя O. В примере на рисунке 1 нейронная сеть содержит только один скрытый слой H.

Каждый слой нейронной сети состоит из нейронов (или узлов). Нейроны разных слоев последовательно связаны друг с другом линиями связи, через которые проходят сигналы.

При передаче сигнала с одного нейрона на другой, значение сигнала корректируется весовым коэффициентом w. Изменение значений весовых коэффициентов и подразумевает обучение нейронной сети.

Помимо этого, нейрон, получающий на вход сигнал, изменяет его, применяя функцию активации к значению входящего сигнала. Поэтому исходящий сигнал нейрона будет отличаться от входящего.

Функция активации – это функция, которая получает входной сигнал и генерирует выходной сигнал с учетом порогового значения [7]. Примером может служить ступенчатая функция: если x превышает определенный порог, то y примет значение 1, иначе 0.

В качестве функции активации обычно используют сигмоиду, которую иногда называют также логистической функцией:

$$y = \frac{1}{1 + e^{-x}}, \tag{3.32}$$

где y – выходной сигнал нейрона,
x – сумма входных сигналов, поступающих в нейрон.

По сравнению со ступенчатой функцией, она не имеет острых углов, то есть ее значения могут быть не только 0 или 1, а находится в определенном диапазоне. Для получения выходного сигнала нейрона также используются и другие S-образные функции.

Значения сигмоидальной функции находятся в диапазоне от -1 до 1. Поэтому для улучшения эффективности нейронной сети имеет смысл проводить стандартизацию данных, в том числе исключая выбросы.

Под выбросом понимают значение переменной, которое существенно отличается от других значений. Анализ выбросов в вероятностных, линейных, информационно-теоретических и других моделях данных подробно изучается в работе Charu C. Aggarwal

[12]. Методы и алгоритмы обнаружения выбросов рассматриваются в работе N. N. R. Ranga Suri [21].

В расчетах при построении нейронной сети также применяется производная функции сигмоиды, которая равна:

$$'y = y \cdot (1 - y) \tag{3.33}$$

Значение входящего сигнала, поступающего в нейрон, представляет собой сумму исходящих сигналов, помноженную на весовые коэффициенты соответствующих линий связи:

$$input = \sum (x_i \cdot w_i)$$

Для нашего примера значение входящего сигнала для нейрона h_1 будет равно:

$$h_{1(input)} = x_1 \cdot w_1 + x_2 \cdot w_2.$$

Значение исходящего сигнала для нейрона h_1 будет равно:

$$h_{1(output)} = f(h_{1(input)}),$$

где $f(h_{1(input)})$ – функция активации, применяемая к входящему сигналу $h_{1(input)}$

Для построения и обучения нейронных сетей используются матрицы и матричные операции.

Допустим, нам необходимо получить значения результирующего признака *y*. Имеющиеся значения признака *y* мы можем представить в виде матрицы *Y*.

Введем условные обозначения матриц:

Y – значения результирующего признака (в виде вектора-столбца),

X – значения факторов (в виде матрицы, где каждая строка содержит значения соответствующего фактора),

H – значения входящих сигналов скрытого слоя,

$H_{sigmoid}$ – значения исходящих сигналов скрытого слоя,

O – значения входящих сигналов выходного слоя,

$O_{sigmoid}$ – значения исходящих сигналов выходного слоя,

W_{ih} – весовые коэффициенты между входным (input) и скрытым (hidden) слоем,

W_{ho} – весовые коэффициенты между скрытым (hidden) и выходным (output) слоем.

Алгоритм построения нейронной сети сводится к тому, что необходимо сначала найти входной сигнал скрытого слоя H через скалярное произведение матриц (формула 3.34), затем применить к этому сигналу функцию активации и получить выходной сигнал скрытого слоя H. После этого, необходимо по тому же принципу найти входной сигнал выходного слоя O (формула 3.35) и выходной сигнала выходного слоя O.

$$H = W_{ih} \cdot X \tag{3.34}$$

Найдем входной сигнал скрытого слоя H для следующего фрагмента нейронной сети (рисунок 3.2).

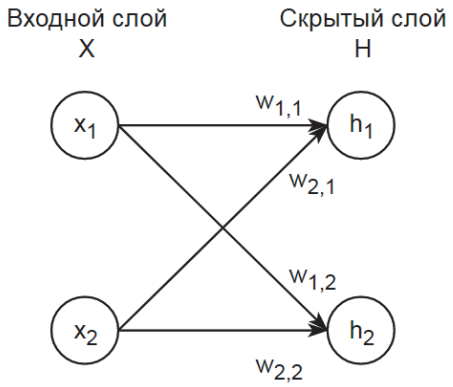


Рисунок 3.2 – Фрагмент нейронной сети с входным и скрытым слоем

Здесь каждый весовой коэффициент имеет два индекса: первый индекс указывает на номер нейрона входного слоя, второй индекс – на номер нейрона скрытого слоя. Например, весовой коэффициент $w_{1,2}$ влияет на связь между первым нейроном входного слоя и вторым нейроном скрытого слоя.

Логично было бы представить матрицу входных сигналов X и матрицу весов W_{ih} следующим образом:

$$X = \begin{pmatrix} x_1 \\ x_2 \end{pmatrix}; \; W_{ih} = \begin{pmatrix} w_{1,1} & w_{1,2} \\ w_{2,1} & w_{2,2} \end{pmatrix}$$

Однако мы не можем произвести скалярное умножение этих матриц, так как число столбцов X (1) не равно числу строк W_{ih} (2). Поэтому поменяем матрицы местами:

$$W_{ih} = \begin{pmatrix} w_{1,1} & w_{1,2} \\ w_{2,1} & w_{2,2} \end{pmatrix}; \; X = \begin{pmatrix} x_1 \\ x_2 \end{pmatrix}$$

Теперь число столбцов W_{ih} (2) равно числу строк X (2) и мы можем найти скалярное произведение данных матриц.

Однако в этом случае первый элемент результирующей матрицы будет находиться следующим образом:

$$h_1 = w_{1,1} \cdot x_1 + w_{1,2} \cdot x_2.$$

Посмотрев на рисунок 3.2, мы знаем, что первый элемент результирующей матрицы должен быть найден так:

$$h_1 = w_{1,1} \cdot x_1 + w_{2,1} \cdot x_2.$$

То есть вес при x_2 должен быть не $w_{1,2}$, а $w_{2,1}$. В связи с этим, нам необходимо транспонировать матрицу весовых коэффициентов, чтобы входные сигналы корректировались соответствующими им весовыми коэффициентами:

$$W_{ih} = \begin{pmatrix} w_{1,1} & w_{2,1} \\ w_{1,2} & w_{2,2} \end{pmatrix}; \; X = \begin{pmatrix} x_1 \\ x_2 \end{pmatrix}$$

Теперь, через скалярное произведение данных матриц, мы можем получить результирующую матрицу H:

$$W_{ih} \cdot X = \begin{pmatrix} w_{1,1} & w_{2,1} \\ w_{1,2} & w_{2,2} \end{pmatrix} \cdot \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = \begin{pmatrix} w_{1,1} \cdot x_1 + w_{2,1} \cdot x_2 \\ w_{1,2} \cdot x_1 + w_{2,2} \cdot x_2 \end{pmatrix} = \begin{pmatrix} h_1 \\ h_2 \end{pmatrix} = H$$

Чтобы рассчитать исходящие сигналы нейронов скрытого слоя H, необходимо применить функцию активации к значениям входящих сигналов:

$$H_{sigmoid} = f(H) \tag{3.35}$$

То есть:

$$H_{sigmoid} = \begin{pmatrix} f(h_1) \\ f(h_2) \end{pmatrix}$$

Найдем входящие сигналы выходного слоя:

$$O = W_{ho} \cdot H_{sigmoid} \tag{3.36}$$

$$O = W_{ho} \cdot H_{sigmoid} = (w_1 \quad w_2) \cdot \begin{pmatrix} f(h_1) \\ f(h_2) \end{pmatrix}$$

В результате скалярного произведения получим матрицу размером 1x1, содержащую одно значение:

$$O = w_1 \cdot (f(h_1) + w_2 \cdot (f(h_2) = (o_1)$$

Применив функцию активации, получим выходной сигнал выходного слоя, то есть выходной сигнал всей нейронной сети:

$$O_{sigmoid} = f(O) \tag{3.37}$$

$$O_{sigmoid} = f(O) = f(o_1)$$

Для обучения нейронной сети необходимо скорректировать все ее весовые коэффициенты во всех слоях. Для этого требуется рассчитать ошибки на каждом слое.

Найдя выходной сигнал слоя O можно приступить к обучению сети методом обратного распространения ошибки.

Найдем ошибки выходного слоя (ошибки сети):

$$E_o = Y^T - O \ , \tag{3.38}$$

где Y – матрица (вектор-столбец) фактических значений результирующего признака,

O – матрица (вектор-строка) значений результирующего признака, полученных на основе нейронной сети.

Нахождение ошибок выходного слоя не является проблемой, в отличие от – ошибок в скрытых слоях.

Метод обратного распространения ошибки – метод распространения ошибки сети между нейронами в обратном направлении (от выхода ко входу), пропорционально весовым

коэффициентам нейронов. Данный метод применяется с целью расчета ошибок в скрытых слоях. Ошибка сети распространяется на нейроны в обратном направлении. Например, нейрон выходного слоя связан с 2 нейронами скрытого слоя, причем веса у этих связей одинаковы. Тогда ошибка этого выходного нейрона распространится на 2 указанных нейрона поровну (1/2 ошибки распространится на первый нейрон, 1/2 ошибки – на второй нейрон).

Если имеется та же ситуация, за исключением того, что веса разные (например, 0,5 и 0,7), то на первый нейрон распространится соответствующая доля ошибки 0,5 / (0,5 + 0,7) = 0,4166 (41,66% ошибки), а на второй нейрон – 0,7 / (0,5 + 0,7) = 0,5833 (58,33% ошибки). Так как второй нейрон имеет связь более сильную, то он больше влияет на выходной сигнал сети, поэтому на него распространяется большая доля ошибки.

В указанных дробях знаменатель можно опустить, так как он служит лишь для масштабирования значений весов. То есть во втором случае на первый нейрон распространится 0,2 ошибки, а на второй нейрон – 0,3 ошибки. Такое допущение позволит исключить сложные вычисления и применить матричную алгебру для расчета ошибок в скрытых слоях, не повлияв на результат обучения.

Найдем ошибки скрытого слоя путем взвешивания ошибок выходного слоя на весовые коэффициенты:

$$E_h = W^T{}_{ho} \cdot E_o \ , \tag{3.39}$$

где $W^T{}_{ho}$ – транспонированная матрица весов W_{ho}.

Таким образом, для каждого слоя мы можем рассчитать соответствующие ошибки. При этом число скрытых слоев может быть произвольным, процедура нахождения ошибок останется прежней.

Изменение ошибки E при изменении веса W_{ho} представляется в виде выражения:

$$\frac{dE}{dW_{ho}} = -E_o * f'(O) \cdot H^T \ , \tag{3.40}$$

где $f'(O)$ – матрица производных от функции активации для входных сигналов выходного слоя *O*,
H^T – транспонированная матрица выходных сигналов скрытого слоя H.

Изменение ошибки *E* при изменении веса W_{ih} равно:

$$\frac{dE}{dW_{ih}} = -E_h * f'(H) \cdot X^T \ , \qquad (3.41)$$

где $f'(H)$ – матрица производных от функции активации для входных сигналов скрытого слоя *H*,
X^T – транспонированная матрица выходных сигналов входного слоя *X*.

В формулах (3.40), (3.41) символом «*» обозначается обычное поэлементное умножение матриц, а символом «·» – скалярное произведение матриц.

Формулы для нахождения новых значений весовых коэффициентов принимают вид:

$$новые\ W_{ho} = старые\ W_{ho} - \alpha \cdot \frac{dE}{dW_{ho}} \ , \qquad (3.42)$$

$$новые\ W_{ih} = старые\ W_{ih} - \alpha \cdot \frac{dE}{dW_{ih}} \ , \qquad (3.43)$$

где – коэффициент обучения (темп обучения), то есть множитель, сглаживающий величину изменений во избежание перескоков через минимум функции ошибки [7].

3.3 Разработка информационной системы

Информационная система разрабатывалась на языке программирования C++ с помощью библиотеки Qt в среде разработки Qt Creator [3]. Разработанная информационная система имеет следующие функциональные возможности:

1. Защищенный вход в систему.
2. Подключение к базе данных Microsoft SQL Server.
   2.1. Импорт данных из всех таблиц базы данных, кроме таблиц псевдонимов.
   2.2. Создание таблиц псевдонимов на сервере базы данных: псевдонимы таблиц, псевдонимы столбцов.

2.3. Отображение в ИС таблиц псевдонимов на общей таблице псевдонимов.
2.4. Использование псевдонимов
3. Создание и редактирование таблиц, включая связи.
3.1. Конструктор создания таблицы.
3.2. Удаление таблицы
3.3. Редактирование свойств таблицы
3.4. Редактирование данных в таблице.
3.4.1. Ввод, изменение, удаление записей из таблицы.
3.4.2. Подстановка значений из другой таблицы (взаимодействие таблиц, сложный ввод).
4. Импорт данных из файла в базу данных.
4.1. Выбор файла.
4.2. Создание и заполнение таблицы в базе данных.
5. Поиск значений в таблице.
6. Сортировка данных.
7. Стандартизация данных.
7.1. Логарифмирование.
7.2. Нормированное отклонение.
7.3. Приведение к диапазону значений от 0 до 1.
8. Корреляционно-регрессионный анализ данных.
8.1. Корреляция.
8.2. Регрессия.
8.3. Оценка значимости уравнения регрессии (F-тест).
8.4. Оценка значимости параметров уравнения регрессии (t-тест).
8.5. Стандартизированные коэффициенты регрессии.
9. Прогноз.
9.1. Прогноз на текущий временной промежуток (моделирование значений).
9.2. Прогноз на будущий временной промежуток.
9.2.1. Прогноз результативной переменной при изменении одного фактора.
9.2.2. Прогноз факторов на основе линейных трендов.
9.2.3. Прогноз результативной переменной по спрогнозированным факторам.
10. Построение графиков.
10.1. График исходных данных.

10.2. График смоделированных значений результативной переменной.
10.3. График прогноза результативной переменной при изменении одного фактора.
10.4. График прогноза факторов на основе линейных трендов.
10.5. График прогноза результативной переменной по спрогнозированным факторам.
11. Сохранение графиков.
11.1. Выбор параметров сохранения.
11.2. Предварительный просмотр графика.
12. Построение нейронных сетей.

В соответствии с определенными функциональными возможностями, структура разрабатываемой ИС была представлена в виде диаграммы классов. Программный код информационной системы включает 15 основных классов (рисунок 3.3):

– 12 классов форм (или окон), так как для каждой формы создается свой класс:

1) главная форма MainWindow;
2) форма входа в систему LoginForm;
3) форма создания таблицы CreateTableForm;
4) форма удаления таблицы DeleteTableForm;
5) форма изменения свойств таблицы UpdateTableForm;
6) форма псевдонимов для таблиц и столбцов AliasesForm;
7) форма импорта данных ImportForm;
8) форма проведения корреляционно-регрессионного анализа Analysis_CorrRegrForm;
9) форма сохранения графика PlotSavingForm;
10) форма отображения графика в отдельной форме PlotShowingForm;
11) подстановочная форма SubstitutionForm;
12) форма построения нейронных сетей NeuralNetworkForm.

– 1 импортированный класс сторонних разработчиков для построения графиков QCustomPlot.

– 2 остальных класса, не привязанных к формам: главный класс Main и класс операций с матрицами MatrixOp.

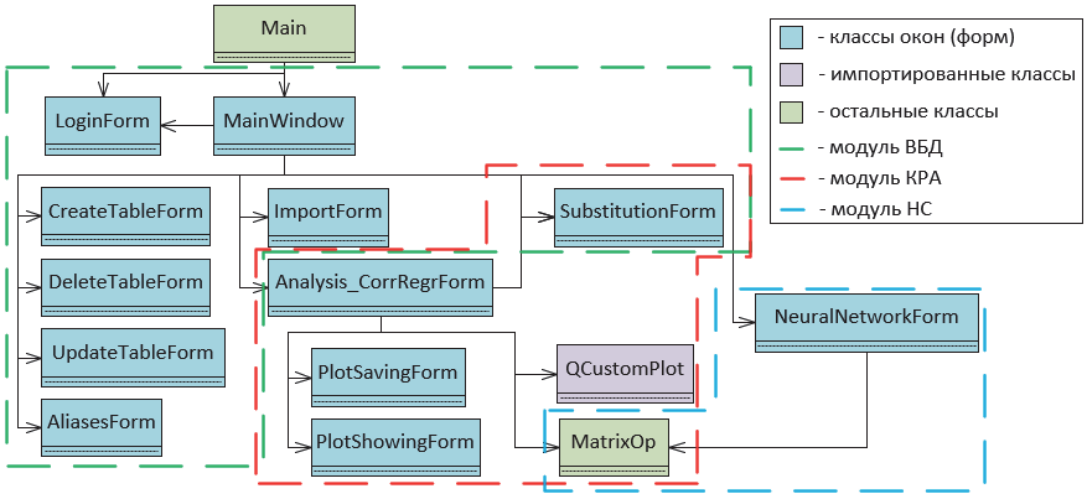


Рисунок 3.3 – Диаграмма классов

Программный код информационной системы условно можно разделить на три части, каждая из которых реализует определенный функционал. Назовем эти части «модулями»:

1. Модуль взаимодействия с базами данных.
2. Модуль проведения корреляционно-регрессионного анализа.
3. Модуль построения нейронных сетей.

1. Модуль взаимодействия с базами данных реализовывает следующий функционал:

- подключение к базе данных Microsoft SQL Server,
- взаимодействия с базой данных.

2. Модуль корреляционно-регрессионного анализа разработан для оценки связей между переменными и построения прогноза. Данный модуль реализует следующий функционал:

- считывание данных из таблицы базы данных,
- считывание данных из файла,
- проведение корреляционно-регрессионного анализа,
- построение прогноза,
- визуализация исходных данных, результатов анализа и прогноза.

3. Модуль построения нейронных сетей разработан для проведения классификации и реализует следующий функционал:

- считывание данных из файла,
- построение нейронной сети,
- использование нейронной сети для классификации.

3.4 Разработка модуля взаимодействия с базами данных

В модуль взаимодействия с базами данных входят следующие 8 классов: LoginForm, MainWindow, CreateTableForm, DeleteTableForm, UpdateTableForm, AliasesForm, ImportForm, SubstitutionForm.

Класс LoginForm позволяет подключаться к конкретной базе данных в соответствии с четырьмя параметрами, вводимыми пользователем: адрес базы данных, имя базы данных, имя пользователя, пароль.

Форма LoginForm, помимо четырех вышеуказанных полей, содержит кнопку «Войти» и элемент checkBox «Показать пароль».

При нажатии на кнопку «Войти» происходит попытка подключения к базе данных, при этом используется информация из всех четырех полей, заполненных пользователем.

Если подключение не удалось установить, то экземпляр класса LoginForm производит сигнал для объекта MainWindow: emit sendDB_Connection_bool(0). Это означает, что объект MainWindow должен вызвать определенный метод, в нашем случае он имеет имя getDB_Connection_bool(bool value). Переменная value данного метода примет значение, передающееся из объекта класса LoginForm, то есть 0. Таким образом, передача данных из объекта одного класса в объект другого (из одного окна приложения в другое окно) осуществляется с помощью сигнала и метода, выполняющегося при подаче сигнала.

Для связи метода с сигналом записывают следующую команду: connect(pLoginForm, SIGNAL(sendDB_Connection_bool(bool)), this, SLOT(getDB_Connection_bool(bool))). Данная команда записывается в файле mainwindow.cpp, который содержит реализацию класса MainWindow, так как именно объект формы MainWindow открывает большинство других форм. В указанной команде параметр pLoginForm является указателем на класс LoginForm, а параметр this – ключевым словом, указывающим на текущий класс, то есть на класс MainWindow. Листинг, содержащий команды для открытия формы LoginForm и установки связи сигнала и метода, может иметь следующий вид:

```
//Создание указателя на форму LoginForm
LoginForm *pLoginForm;
```

```
//Метод соединения с базой данных (выполняется
автоматически при запуске //приложения)
void MainWindow::connectToDB()
{
    pLoginForm = new LoginForm();
    pLoginForm -> show();
    connect(pLoginForm,
    SIGNAL(sendDB_Connection_bool(bool)), this,
    SLOT(getDB_Connection_bool(bool)));
    …
}
```

В нашем случае команда `LoginForm* pLoginForm` записана в отдельном заголовочном файле mainwindow.h, содержащем описания методов и переменных класса `MainWindow`. Остальные команды записаны в файле mainwindow.cpp.

Таким образом, в методе `getDB_Connection_bool(bool value)` класса `MainWindow` происходит сохранение передающегося из класса `LoginForm` значения value в глобальную переменную `DB_Connection_bool` класса `MainWindow`. Это нужно для того, чтобы избежать ошибки при попытке считывания информации о базе данных. Если значение `DB_Connection_bool` равно 0, то команды, приводящие к считыванию информации о базе данных не будут выполняться. Такие команды содержатся в методе `on_tb_select_triggered()` класса `MainWindow`, вызывающимся при успешном подключении к базе данных или при нажатии на кнопку «Обновить» на форме `MainWindow`.

Если подключение удалось установить, то объект класса `LoginForm` перед тем, как закрыть форму `LoginForm`, передает в объект класса `MainWindow` информацию, производя 3 сигнала (методы с ключевым словом `emit`):

```
//Подключение к базе данных
if (db.open())
{
    //Передача информации о базе данных
    emit sendDB_Connection_bool(1);
    emit sendDB_Name(&db, ui -> lineEdit_2 ->
    text());
    emit signal_ConnectionEstablished();
```

```
    this -> close();
}
```

С помощью первых двух сигналов передается значение 1 для переменной DB_Connection_bool и имя базы данных из поля lineEdit_2 в переменную DB_Name. Третий сигнал вызывает 3 метода класса MainWindow:

- метод on_tb_select_triggered(), вызывающийся при нажатии на кнопку «Обновить» (вывод данных),
- метод получения информации о столбцах со свойством IDENTITY getDB_ColsIDENTITY(),
- метод получения названий первичных ключей getDB_ColsPKeysAndUNIQUE().

Метод getDB_ColsIDENTITY() класса MainWindow с помощью SQL-запроса к базе данных запрашивает информацию о столбцах со свойством IDENTITY. Столбцы с таким свойством заполняются автоматически, а не пользователем вручную, то есть имеется некоторое начальное значение и инкремент, увеличивающий это значение. Таким образом, полная запись атрибута выглядит следующим образом: IDENTITY(seed, increment). Обычно данное свойство имеют столбцы-идентификаторы.

Полученная информация сохраняется в виде матрицы DB_ColsIDENTITY, имеющей 4 столбца: имя таблицы, имя столбца, начальное значение, инкремент.

Информацию о столбцах со свойством IDENTITY необходимо иметь, чтобы не заполнять ячейку в таком столбце, то есть данная информация будет необходима при добавлении новой строки в таблицу, а также при изменении строки.

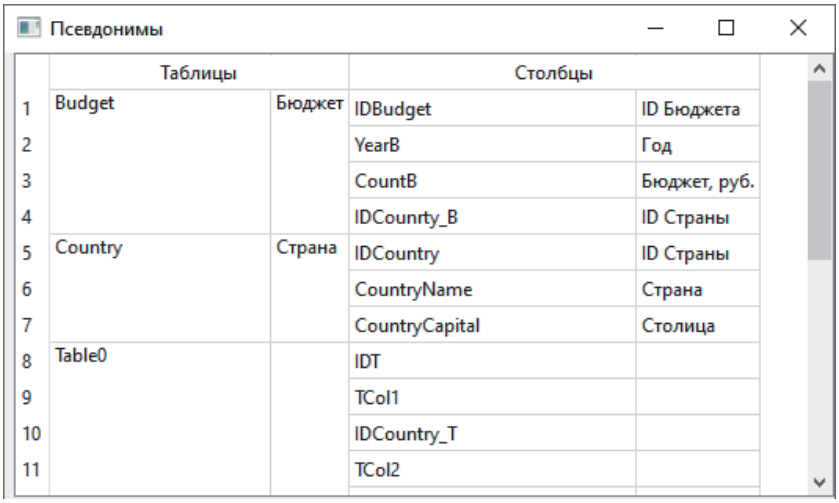
Метод getDB_ColsPKeysAndUNIQUE() по аналогии с предыдущим методом получает информацию о столбцах со свойством PRIMARY KEY, FOREIGN KEY или UNIQUE. Информация сохраняется в матрицу, имеющую 3 столбца: имя таблицы, имя столбца, имя свойства. Для простоты, имя свойства для вышеуказанных свойств будет PK, FK или UQ.

Метод on_tb_select_triggered(), вызывающийся при нажатии на кнопку «Обновить», реализует большое число операций. Он получает информацию о внешних ключах таблиц, сохраняя ее в матрицу qmodel_ColsFKeys, состоящую из 6

столбцов: имя таблицы, имя столбца, имя связанной таблицы, столбец связанной таблицы, тип столбца связанной таблицы (PK, FK или UQ), имя ограничения (уникальное имя, генерирующееся СУБД или администратором базы данных при создании таблицы).

Далее метод получает информацию о таблицах, столбцах, определяет начальные индексы для таблицы в массиве столбцов, получает информацию о псевдонимах таблиц и столбцов (если псевдонимы, то есть имена таблиц и столбцов, которые отображаются в приложении, установлены) с помощью следующих методов: `getDB_TablesInfo()`, `getDB_ColsInfo()`, `getDB_TablesStartIndexes()`, `getAliases()`.

Таблица псевдонимов имеет следующий вид (рисунок 3.4):



	Таблицы		Столбцы	
1	Budget	Бюджет	IDBudget	ID Бюджета
2			YearB	Год
3			CountB	Бюджет, руб.
4			IDCounrty_B	ID Страны
5	Country	Страна	IDCountry	ID Страны
6			CountryName	Страна
7			CountryCapital	Столица
8	Table0		IDT	
9			TCol1	
10			IDCountry_T	
11			TCol2	

Рисунок 3.4 – Форма «Псевдонимы», отображающая названия таблиц и столбцов, а также их псевдонимы (если таковые имеются)

Затем с помощью метода `insertNewNamesInAliasesTables()` записываются новые псевдонимы в таблицу псевдонимов таблиц `TablesAliases` и в таблицу псевдонимов столбцов `ColumnsAliases`, если две эти таблицы созданы в базе данных. Создание таблиц псевдонимов происходит с помощью метода `createAliasesTables()`, который вызывается в результате ответа пользователя в режиме диалогового окна, появляющегося при нажатии на кнопку «Меню – Псевдонимы» и в случае, если таблицы псевдонимов не созданы в базе данных.

Далее в методе `on_tb_select_triggered()` вызывается метод `get_colsIndexes_FKeys()` с целью получения индексов столбцов внешних ключей для каждой таблицы.

В методе `get_colsIndexes_FKeys()` создается матрица `colsIndexes_FKeys`, хранящая индексы столбцов – внешних ключей по каждой таблицы, то есть число строк данной матрицы равно числу таблиц в базе данных, а каждая строка *i* имеет число ячеек, равное числу столбцов – внешних ключей для таблицы *i*. Если в таблице *i* нет столбцов – внешних ключей, то строка *i* в матрице `colsIndexes_FKeys` содержит 0 ячеек. Таким образом, матрицу `colsIndexes_FKeys` сложно представить в виде стандартной таблицы, имеющей определенное число строк и столбцов, так как число столбцов в каждой строке может быть любым.

Также в методе `get_colsIndexes_FKeys()` создается матрица `tablesReferences_FKeys`, состоящая из 4 столбцов: индекс таблицы, индекс столбца – внешнего ключа, индекс таблицы столбца – внешнего ключа, индекс столбца внутри таблицы столбца – внешнего ключа (то есть индекс первичного или уникального ключа в связанной таблицы, который является столбцом – внешним ключом в исходной таблицы). По содержанию матрица `tablesReferences_FKeys` напоминает матрицу `qmodel_ColsFKeys`, однако в отличие от второй, первая содержит не имена, а индексы, с которыми удобно работать при реализации подстановочной таблицы.

Метод `on_tb_select_triggered()` после вызова вышеуказанных методов, в цикле с числом итераций равным числу таблиц в базе данных вызывает метод `getDataFromTable()`, считывающий данные из таблицы и загружающий их в созданную таблицу информационной системы.

Класс `MainWindow` также содержит методы для работы с таблицами базы данных: добавление строки, изменение строки, удаление строки или сразу несколько выделенных строк. При добавлении или изменении ячейки столбца – внешнего ключа открывается форма `SubstitutionForm`, содержащая подстановочную таблицу. Выбор любой ячейки подстановочной таблицы автоматически приводит к выделению всей строки подстановочной таблицы, а в ячейке исходной таблицы меняется

значение на значение первичного или уникального ключа подстановочной таблицы в выделенной строке.

Для отображения подстановочной формы вызывается метод `showSubstitution(QModelIndex qmindex)`, принимающий в качестве параметра переменную `qmindex` с типом `QModelIndex`. Класс `QModelIndex` позволяет хранить сложный индекс элемента `QModel`, представляющего собой матрицу данных (модель данных), использующуюся в качестве источника данных для таблиц `QTableView` в Qt. Хранение сложного индекса подразумевает хранение индекса строки и индекса столбца одновременно, то есть для хранения индексов строки и столбца ячейки таблицы можно использовать одну переменную `QModelIndex` вместо двух – `int`, хранящих эти индексы. В данном методе происходит определение индекса открытой пользователем таблицы, индекса последней выделенной ячейки и значение в этой ячейки.

Далее, если столбец имеет свойство внешнего ключа, то есть содержится в матрице colsIndexes_FKeys, то с помощью матрицы `tablesReferences_FKeys` определяется таблица, на которую ссылается столбец – внешний ключ. После чего объекту класса `SubstitutionForm` передается модель данных для подстановочной таблицы, значение в последней выделенной ячейке и индекс столбца в подстановочной таблице, по которому осуществляется связь между таблицами. Наконец, происходит открытие подстановочной таблицы. Если пользователь изменяет значение в ячейке столбца – внешнего ключа, то в открытой подстановочной таблице изначально автоматически выделяется строка с текущим индексом столбца – внешнего ключа, который записан в изначальной таблице. Если столбец не имеет свойство внешнего ключа, то подстановочная таблица не открывается, а в случае, если она была открыта – закрывается.

Класс `CreateTableForm` обеспечивает возможность создавать новые таблицы в базе данных. Форма `CreateTableForm` открывается с помощью объекта класса `MainWindow` при нажатии пользователем на кнопку «+» в области, где расположены таблицы. Кнопка «+» является пустой вкладкой (элемент `tab`) элемента `tabWidget`, и в отличие от всех остальных вкладок, не содержит таблицу (рисунок 3.5).

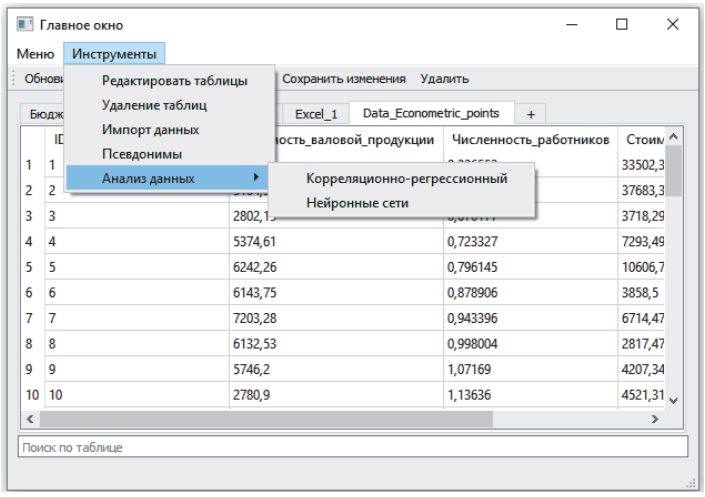


Рисунок 3.5 – Форма «Главное окно» после подключения к базе данных. Закрытие формы подключения к базе данных

При открытии формы `CreateTableForm` в класс `CreateTableForm` из класса `MainWindow` передается массив имен таблиц в базе данных и матрица столбцов-ключей `DB_ColsPKeysAndUNIQUE`. Сама форма `CreateTableForm` имеет 4 области элементов: «Настройки таблиц», «Код запроса», «Столбцы таблицы», «Дополнительные ограничения столбцов» (рисунок 3.6).

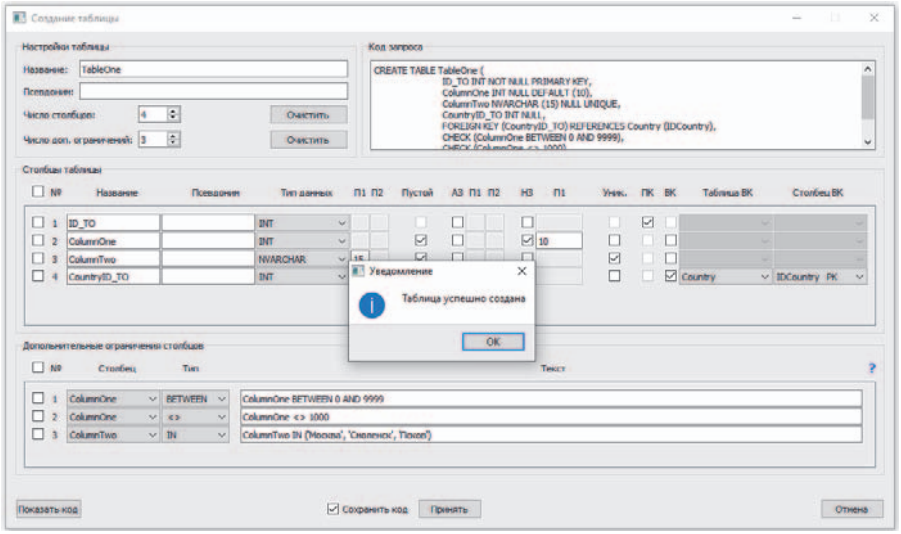


Рисунок 3.6 – Форма «Создание таблицы». Установка названия новой таблицы, числа столбцов, дополнительных ограничений столбцов. Настройка столбцов и дополнительных ограничений столбцов. Нажатие на кнопку «Принять»

Область «Настройки таблиц» позволяет пользователю задать имя таблицы, псевдоним таблицы, число столбцов, число дополнительных ограничений для столбцов.

Область «Код запроса» показывает сгенерированный код на языке T-SQL в соответствии с информацией из трех остальных областей формы.

Область «Столбцы таблицы» предназначена для настройки столбцов будущей таблицы. В ней пользователь имеет возможность задать название столбца, псевдоним столбца, тип данных столбца, а также указать такие свойства столбца, как допущение пустых значений (`NULL`), автоматическое заполнение (`IDENTITY`), наличие начального значения (`DEFAULT`), наличие только уникальных значений (`UNIQUE`), свойство первичного ключа (`PRIMARY KEY`) и внешнего ключа (`FOREIGN KEY`), название таблицы внешнего ключа, название столбца внешнего ключа в таблице внешнего ключа.

Область «Дополнительные ограничения столбцов» позволяет устанавливать дополнительные ограничения для столбцов из области «Столбцы таблицы». Под дополнительными ограничениями подразумеваются ограничения на допустимые значения столбца. К числу дополнительных ограничений можно отнести, например, такие ограничения, как: равенство определенному значению, границы допустимых значений (`BETWEEN`), вхождение значения в набор значений (`IN`), соответствие значения шаблону (`LIKE`).

3.5 Разработка модуля проведения корреляционно-регрессионного анализа

Модуль КРА разработан для выявления связей между переменными, построения прогноза и визуализации данных. Он состоит из 6 классов, главным из которых является непосредственно класс проведения КРА `Analysis_CorrRegrForm`.

Для реализации множественного КРА на основе метода обратной матрицы необходимо иметь класс операций (методов) с матрицами, который можно было бы использовать для проведения всех необходимых расчетов. Такой класс также будет необходим при построении нейронных сетей.

Разработанный класс операций с матрицами `MatrixOp` состоит из методов, принимающих в качестве параметров матрицы

в виде двумерных векторов и некоторые другие параметры, например, вещественное число, на которое нужно умножить матрицу. Также методы имеют один дополнительный параметр – указатель на матрицу-результат, благодаря чему можно передавать пустую матрицу в метод в качестве параметра и получать заполненную матрицу в результате выполнения метода.

Матрица-результат при первом вызове метода будет пустой, в самом методе она заполняется в зависимости от того, какой метод вызван. При повторном вызове метода старая матрица-результат очищается и заполняется вновь. Очистка происходит в результате проверки, является ли пустой матрица-результат.

Разработанный класс операций с матрицами содержит 33 метода, основные из которых следующие: транспонирование, скалярное произведение, расчет обратной матрицы; а также другие вспомогательные методы, например, метод расчета регрессии по коэффициентам, оценки значимости уравнения и параметров, прогноза, построения графиков и т.д.

Форма проведения корреляционно-регрессионного анализа данных `Analysis_CorrRegrForm` открывается при нажатии на кнопку «Инструменты – Анализ данных – Корреляционно-регрессионный анализ» на форме `MainWindow`. При этом, если подключение к базе данных установлено, объект класса `MainWindow` передает объекту класса `Analysis_CorrRegrForm` следующую информацию о таблицах базы данных: массив имен таблиц в базе данных, массив чисел столбцов (количество столбцов) в каждой таблице, массив имен столбцов, массив индексов начальных столбцов для каждой таблицы в общем массиве столбцов, массив чисел строк (количество строк) в каждой таблицы.

Форма `Analysis_CorrRegrForm` состоит из 4 областей элементов: «Параметры модели», «Выбор переменных», «Прогноз», «Результат анализа»; а также имеет кнопки, не входящие ни в одну из областей: «Сохранить график», «Открыть файл», «Закрыть файл», «Открыть таблицу», «Анализ», «Отмена» (рисунок 3.7).

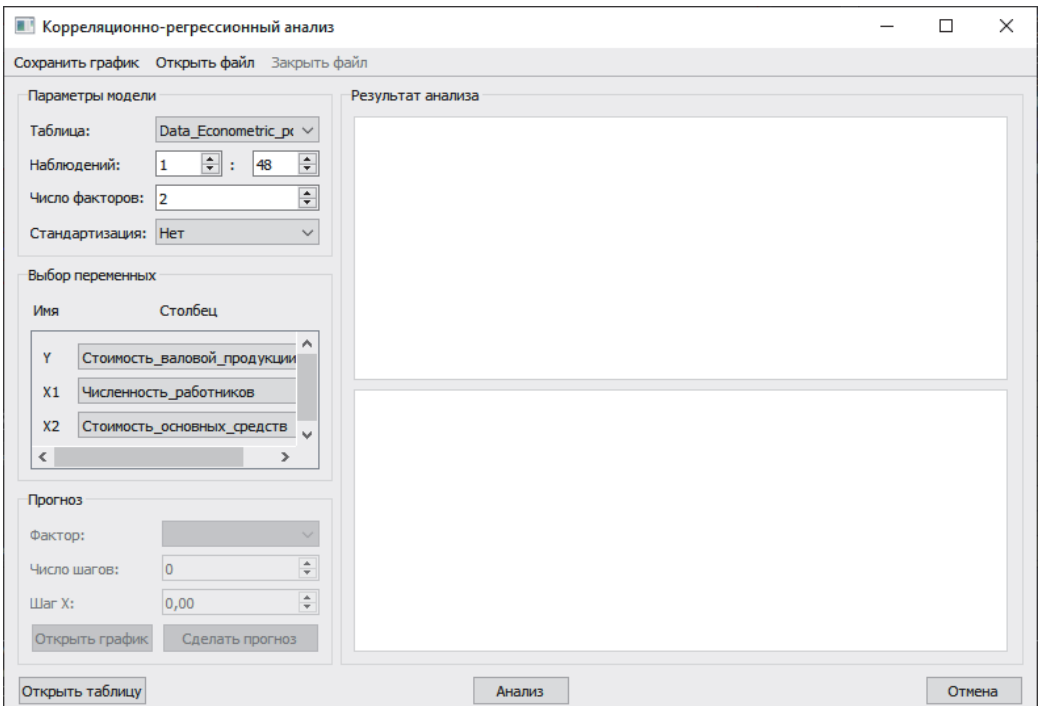


Рисунок 3.7 – Форма «Корреляционно-регрессионный анализ» при открытии. Автоматическая установка параметров анализа: выбор таблицы, интервала наблюдений, числа факторов, стандартизации, переменных

В области «Параметры модели» можно выбрать таблицу из базы данных, интервал наблюдений, число факторов, вид стандартизации. В области «Выбор переменных» выбираются переменные для анализа. В области «Прогноз» устанавливаются параметры прогноза. В области «Результат анализа» отображаются результаты анализа: таблицы и графики.

Проведение анализа данных возможно как на основе данных из таблицы базы данных, так и на основе данных из файла Excel. Для проведения анализа данных можно следовать следующем этапам.

1. Выбор источника данных. На этом этапе пользователь выбирает таблицу из базы данных, либо открывает файл с исходными данными, первая строка которого содержит имена столбцов. После выбора источника данных все параметры анализа данных устанавливаются автоматически, поэтому пользователь может сразу перейти к этапу 4, то есть нажать на кнопку «Анализ» для получения результатов анализа выбранных исходных данных.

2. Установка параметров анализа данных. Пользователь может установить интервал наблюдений, число факторов, вид

стандартизации исходных данных (отсутствие стандартизации, логарифмирование, нормированное отклонение). Если в качестве стандартизации выбрано логарифмирование, то нули в исходном наборе данных заменяются на единицы, так как $log_a b$ существует при $a, b > 0$.

3. Выбор переменных для анализа. На данном этапе пользователь определяет, какой столбец из набора исходных данных относится к зависимой переменной *Y*, а какие столбцы – к независимым переменным X_i, то есть к факторам.

4. Проведение корреляционно-регрессионного анализа. Для проведения корреляционно-регрессионного анализа пользователь должен нажать на кнопку «Анализ», после чего в области «Результат анализа» сформируется 6 таблиц и 2 графика:

- таблица исходных данных «ИД»;
- таблица модели «Модель», имеющая только выбранные пользователем столбцы из всех столбцов «ИД»;
- таблица «Корреляция», представляющая собой тепловую карту парных коэффициентов корреляции между столбцами таблицы «Модель»;
- таблица «F-тест», содержащая следующие значения: коэффициент детерминации *R squared*, коэффициент корреляции *R*, фактическое значение критерия F-Фишера, значения функции плотности распределения F-Фишера, *SE*, число наблюдений, число факторов, числа единиц свободы и *SS*;
- таблица «t-тест», содержащая следующие значения по каждой переменной, то есть по каждому столбцу таблицы «Модель»: коэффициенты линейного уравнения регрессии *Coeffs.*, стандартные ошибки *SE*, фактические значения критерия t-Стьюдента *t*, значения функции плотности распределения t-Стьюдента *t prob*., критические значения критерия t-Стьюдента *t crit*., суммы *Sum.*, средние значения *Mean*, дисперсии *Var.*, а также стандартизированные коэффициенты регрессии (бета-коэффициенты *Beta*, коэффициенты эластичности *Elasticity*, коэффициенты детерминации *d squared*);
- таблица смоделированных значений переменной Y или прогноза на текущий временной промежуток (present) «Прогноз P.», содержащая: фактические значения *Y*, спрогнозированные значения *Y Y_forecst*, значения нижней границы прогноза

Y_lowerBound, значения верхней границы прогноза *Y_upperBound*, средние ошибки прогноза *ME_forecast*, значения факторов X_i;

- график «Исходные данные», отображающий данные таблицы «Модель»;
- график «Прогноз по всем и.д.», отображающий первые 4 столбца таблицы «Прогноз Р.».

5. Построение прогноза. После создания модели регрессии, данную модель можно использовать для построения прогноза. Чтобы построить прогноз, в области элементов «Прогноз» пользователю, перед тем как нажать на кнопку «Сделать прогноз», достаточно установить следующие параметры прогноза: выбрать один из факторов, установить число шагов и значение шага, в соответствии с которым выбранный фактор будет изменяться.

Таким образом, пользователь задает значения факторных переменных и получает спрогнозированные значения исследуемой зависимой переменной *Y*, причем значения всех остальных факторов принимаются равными их средним значениям и остаются неизменными на каждом новом шаге. Помимо этого, строится также второй прогноз *Y*. Во втором прогнозе учитывается только число шагов, тогда как выбранный пользователем фактор и значение шага не учитываются.

Второй прогноз основывается на линейных трендах всех факторов. Сначала строятся модели парной линейной регрессии. В каждой такой модели в качестве зависимой переменной принимается фактор X_i (где i –порядковый номер фактора), в качестве независимой переменной – переменная, содержащая порядковые номера наблюдений. Число таких моделей равняется числу факторов в модели множественной линейной регрессии, построенной на этапе 4.

По построенным моделям парной линейной регрессии формируются прогнозы для факторов X_i на заданное ранее пользователем число шагов. После получения прогнозных значений факторов X_i, формируется прогноз уже основной зависимой переменной *Y* с помощью, построенной на этапе 4 модели множественной линейной регрессии.

Результаты вышеописанных двух прогнозов отображаются в области элементов «Результат анализа» путем создания 4 новых таблицы, 2 новых графиков и изменения 1 старого графика:

- таблица «Прогноз F.» отображает прогноз независимой переменной *Y* на определенное число шагов при изменении одного из факторов X_i на значения шага, при учете, что значения остальных факторов будут равны средним значениям и не будут меняться. Данная таблица содержит такие столбцы, как: порядковый номер шага *Step*, спрогнозированные значения *Y Y_forecst*, значения нижней границы прогноза *Y_lowerBound*, значения верхней границы прогноза *Y_upperBound*, средние ошибки прогноза *ME_forecast*, значения факторов X_i.
- таблица «Лин. уравнения» содержит коэффициенты парных линейных уравнений регрессии, в которых зависимая переменная представляет собой одну из переменных таблицы «Модель», а независимая – порядковый номер наблюдения. Также данная таблица содержит значение парного коэффициента детерминации по каждому уравнению для того, чтобы пользователь мог оценить качество каждой модели парной линейной регрессии;
- таблица «Лин. тренды» отображает спрогнозированные значения каждой переменной на заданное число шагов на основе моделей парных линейных регрессий из таблицы «Лин. уравнения»;
- таблица «Прогноз F. (X тренды)» отображает прогноз независимой переменной *Y* на определенное число шагов при спрогнозированных значениях факторов X_i на основе линейных трендов (таблица «Лин. тренды») и содержит столбцы: спрогнозированные значения *Y Y forecst*, значения нижней границы прогноза *Y lb*, значения верхней границы прогноза *Y ub*, средние ошибки прогноза *ME*, спрогнозированные значения факторов X_i.;
- график «Прогноз факторов» отображает значения исходных данных из таблицы «Модель», а также спрогнозированные значения факторов X_i из таблицы «Лин. тренды»;
- график «Прогноз Y по спрогноз. X» отображает первые 4 столбца из таблицы смоделированных значения *Y* «Прогноз P.», а также прогноз *Y* по линейным трендам факторов X_i из таблицы «Прогноз F. (X тренды)»;
- график «Прогноз по всем и.д.» видоизменяется: помимо отображения первых 4 столбцов из таблицы смоделированных

значений *Y* «Прогноз P.», добавляется отображения прогноза *Y* при изменении одного из факторов X_i из таблицы «Прогноз F.»;

6. Сохранения графиков. На данном этапе пользователь может сохранить графики на компьютер в виде изображения. Для сохранения выбранного графика необходимо нажать на кнопку «Сохранить график», после чего откроется форма `PlotSavingForm`, с помощью которой можно выбрать папку для сохранения изображения, задать имя рисунка, добавить заголовок графика, названия осей графика, выбрать размер сохраняемого изображения или задать произвольный размер. Также имеется возможность предварительного просмотра графика, реализуемая с помощью кнопки «Предпросмотр».

При нажатии на кнопку «Предпросмотр» формы `PlotSavingForm`, происходит копирование сохраняемого графика и открытие копии на отдельной форме `PlotShowingForm`, при этом из класса `PlotSavingForm` в класс `PlotShowingForm` передается информация о заголовке и названиях осей, поэтому копия сохраняемого графика дополняется заголовком и названиями осей.

Открыть график в отдельном окне также можно с формы `Analysis_CorrRegrForm`, нажав на кнопку «Открыть график». Помимо графика, имеется возможность открыть в отдельном окне и таблицу с исходными данными, нажав на кнопку «Открыть таблицу». Для этой цели используется форма `SubstitionForm`, в класс которой из класса `Analysis_CorrRegrForm` передается модель данных `mainModel`.

Модель данных `mainModel` заполняется данными из выбранной пользователем таблицы базы данных, либо, если открыт файл, то `mainModel` заполняется данными из открытого файла.

Главным методом класса `Analysis_CorrRegrForm` является метод `on_pushButton_Analysis_clicked()` – это метод, вызывающийся при нажатии на кнопку «Анализ». В данном методе происходит заполнение модели данных `mainModel` с помощью метода `getTableDataModel()`, после чего данные из `mainModel` сохраняются в две матрицы: матрица исходных данных `initialDataTable`, матрица данных для модели `regrModelDataTable`, содержащая только столбцы-переменные, выбранные пользователем в области «Выбор переменных».

Далее с помощью методов класса операций с матрицами `MatrixOp` происходит расчет регрессии, корреляции, проводится F-тест и t-тест, рассчитываются стандартизированные коэффициенты регрессии. Все основные расчеты представляются в виде таблиц, по данным которых строятся графики:

```
//Кнопка "Анализ"
void
Analysis_CorrRegrForm::on_pushButton_Analysis_clicked()
{
    MatrixOp matrixOp;

    //Очистка матриц векторов
    if (resultTable_parametersQuality.size() > 0)

matrixOp.clearMatrix(&resultTable_parametersQuality);
    if (firstAnalysis == 0)
    {

matrixOp.clearMatrix(&resultTable_parametersQuality);
        tabIdx_lastSelected = ui->tabWidget_Result->currentIndex();
    }

    ui->groupBox_Forecast->setEnabled(1);
    analysisCounter = analysisCounter + 1;

    //Получение модели данных таблицы
    getTableDataModel();

    //Заполнение двухмерного массива данными из модели
    fill_initialDataTable();
    fill_regrModelDataTable();
    factorsCount = varsCount - 1;
    observationsCount = Y_table.size();
```

```
    //Число степеней свободы
    QVector<int> df_vec =
getDegreesOfFreedom(observationsCount,
factorsCount);
    df_total = df_vec[0];
    df_regr = df_vec[1];
    df_error = df_vec[2];
    tablesCount_tabRes = 4;

declareTablesForTabWidget_tabRes(tablesCount_tab
Res); //Объявление 4 таблиц

    //Регрессия
    matrixOp.transposeMatrix(X_1_table,
&X_transpose_table);

    QVector<QVector<double>> XtX;

matrixOp.scalarProduct_matrices(X_transpose_tabl
e, X_1_table, &XtX);
    matrixOp.invertibleMatrix(XtX, &XtX_inv);

    QVector<QVector<double>> XtY;

matrixOp.scalarProduct_matrices(X_transpose_tabl
e, Y_table, &XtY);
    matrixOp.scalarProduct_matrices(XtX_inv,
XtY, &B_table);

    QVector<QVector<double>> Y_regr;
    computeRegression(X_table, B_table,
&Y_regr);

    QVector<QVector<double>> E_table;
    matrixOp.matrixDifference(Y_table, Y_regr,
&E_table);

    QVector<QVector<double>> Et_table;
```

```
    matrixOp.transposeMatrix(E_table,
&Et_table);

    QVector<QVector<double>> Y_minus_mean_table;
    double Y_mean =
matrixOp.matrixCol_Mean(Y_table);
    matrixOp.matrixMinusValue(Y_table,
&Y_minus_mean_table, Y_mean);
    matrixOp.transposeMatrix(Y_minus_mean_table,
&Y_minus_mean_table);

    QVector<QVector<double>> Q_table;

matrixOp.scalarProduct_matrices(Y_minus_mean_tab
le, Y_table, &Q_table);
    Q_total = Q_table[0][0];

    QVector<QVector<double>> Qe_table;
    matrixOp.scalarProduct_matrices(Et_table,
E_table, &Qe_table);
    Q_error = Qe_table[0][0];
    Q_regr = Q_total - Q_error;

    QStringList rows_names_YX;
    rows_names_YX.append("Y");
    for (int i = 1; i <
initialDataTable[0].size(); i++)
        rows_names_YX.append("X" +
QString::number(i));


createTableInTabWidget_tabRes(initialDataTable,
"ИД", 0, rows_names_YX);

createTableInTabWidget_tabRes(regrModelDataTable
, "Модель", 0, rows_names_YX);

    double R_squared = Q_regr / Q_total;
    double R = sqrt(R_squared);
```

```
    double F_fact = (R_squared / (1 -
R_squared)) * (double(df_error) /
double(df_regr));
    double F_prob =
function_Fisher_CumulativeDistribution(F_fact,
df_regr, df_error);
    SE = sqrt(Q_error / df_error);
    …
}
```

Расчет регрессии осуществляется следующим образом. Сначала происходит объявление вектора векторов или двумерного вектора с типом данных double для хранения вещественных чисел с плавающей точкой. Например, объявление транспонированной матрицы X с названием `X_transpose_table` осуществляется посредством команды: `QVector<QVector<double>> X_transpose_table` (в листинге выше данная команда не отражена).

Затем вызывается один из методов класса операций с матрицами `MatrxiOp`, в который передается объявленная матрица. В методе происходит проверка, является ли передаваемая матрица пустой, то есть, содержит ли она какое-то количество строк и столбцов. Если матрица пустая, то происходит описание матрицы: установка числа строк и столбцов. При самом первом проведение КРА матрица `X_transpose_table` будет пустой. Если матрица не пустая, то происходит очистка матрицы, заключающаяся в удалении строк и столбцов. После очистки следует описание матрицы и ее заполнение соответствующими значениями.

Например, для получения заполненной транспонированной матрицы X `X_transpose_table` вызывается метод `transposeMatrix()` экземпляра класса `MatrixOp`, в который передается два параметра (исходная матрица, матрица-результат на заполнение): `matrixOp.transposeMatrix(X_1_table, &X_transpose_table)`.

3.6 Разработка модуля построения нейронных сетей

Класс построения нейронных сетей `NeuralNetworksForm` содержит 43 метода. Все методы можно условно разделить на 5 групп. Группа «Кнопки и считывание файла» содержит 10 методов,

группа «Матрицы» – 13, группа «Нейронные сети» – 5, группа «Создание нейронной сети. Кнопки» – 7, «Создание нейронной сети. Сигналы» – 6 методов. Оставшиеся 2 метода не входят ни в одну группу, так как они создаются автоматически в среде разработки Qt Creator.

Форма выглядит `NeuralNetworksForm` следующим образом (рисунок 3.8):

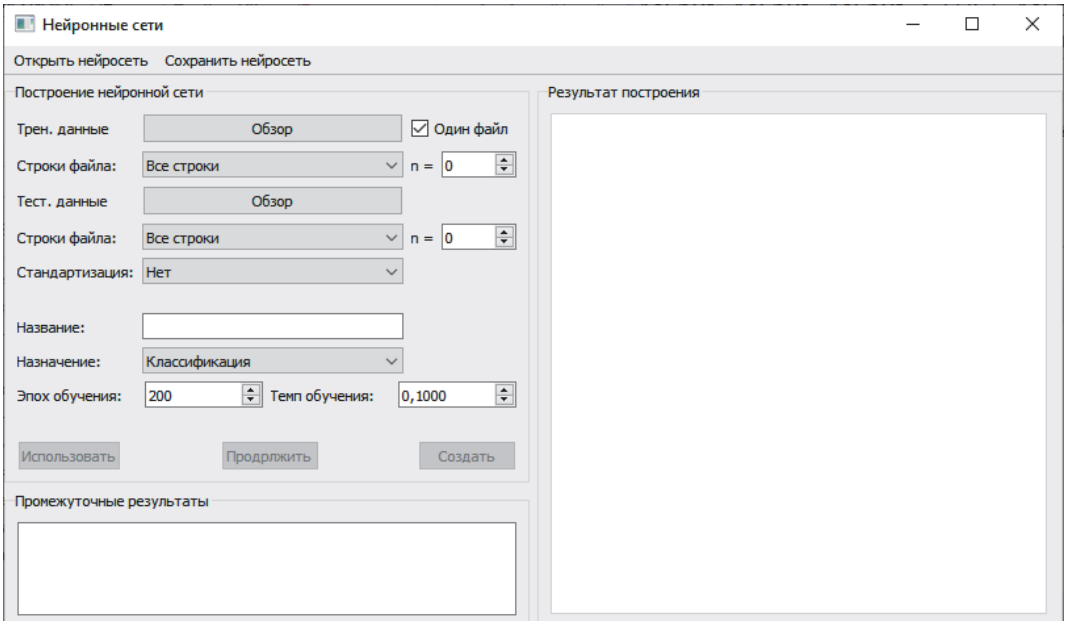


Рисунок 3.8 – Форма «Нейронные сети» при открытии

В первых строках программы, до методов, подключаются библиотеки. В файле neuralnetworksform.cpp происходит подключение следующих библиотек и файлов:

```
#include "neuralnetworksform.h"
#include "ui_neuralnetworksform.h"
#include <fstream>      //Для работы с файлами
#include <sstream>      //Для разбиения строки
#include <cmath>        //Для математических
функций
#include <windows.h>    //Для скрытия консоли
#include <regex>        //Для замены подстроки в
строке
#include <experimental/filesystem> //Для
создания директории
#include <QFileDialog>  //Для выбора директории
файла
#include <QDebug>       //Для вывода ошибок
```

```
#include <QHeaderView>  //Для
setSectionResizeMode в tableView
#include <QScrollBar>   //Для параметров
scrollBar в tableView
#include <QDateTime>    //Для определения даты и
времени
```

Далее идут 2 автоматически созданных метода. В методе открытия формы `NeuralNetworksForm` задаются настройки различных элементов формы. Метод закрытия формы `NeuralNetworksForm` удаляет указатель на область пользовательского интерфейса.

Рассмотрим методы группы 1 «Кнопки и считывание файла».

1.1 Метод `on_tb_open_triggered` выполняется при нажатии на кнопку «Открыть нейросеть» и позволяет открывать файл с сохраненными результатами обучения нейронной сети методом 1.2 `get_filePath` и считывать из него всю информацию с помощью методов 1.3 `get_fileRowsCount_fileColsCount` и 1.4 `readFile_savedNN`.

1.2 Метод получения пути к файлу `get_filePath` принимает указатель на переменную `filePath` типа `QString` и изменяет ее значение в результате открытия диалогового окна выбора директории, в котором пользователь выбирает файл. Путь к выбранному файлу сохраняется в качестве значения переменной `filePath`:

```
//Метод получения пути к файлу
void NeuralNetworksForm::get_filePath(QString
*filePath)
{
    *filePath =
    QFileDialog::getOpenFileName(this, "Выбрать
    путь к файлу", "C://",
    "All files(*.*);;Excel files(*.xl*)");
}
```

1.3 Метод получения числа строк и столбцов в файле `get_fileRowsCount_fileColsCount` принимает 3 параметра: строковую переменную пути к файлу `QString filePath`,

указатель на переменную числа строк `int *fileRowsCount`, указатель на переменную числа столбцов `int *fileColsCount`.

В методе происходит считывание файла построчно с помощью функции C++ `std::getline`, благодаря чему с помощью счетчика определяется число строк в файле и присваивается переменной `fileRowsCount`.

Число столбцов определяется так же с помощью счетчика. Сначала считывается вторая строка файла (во избежание ошибок при наличии строки названий столбцов). Далее определяется символ разделителя (например, пробел или точка с запятой) и производится посимвольное сравнение второй строки файла с символом разделителя. При каждом найденным разделителе счетчик увеличивается и в итоге определяется число столбцов в файле, которое присваивается переменной `fileColsCount`.

1.4 Метод чтения файла исходных данных `readFile_dataIn` принимает в качестве параметра переменную `filePath`, содержащую путь к файлу. В методе описываются три матрицы, представляющие собой двумерные векторы или векторы векторов: матрица исходных данных `dataIn`, матрица факторов `X`, матрица результирующей переменной `Y`. Файл с исходными данными считывается построчно, однако после каждой считанной строки, данная строка разбивается на элементы в соответствии с разделителем, и каждый элемент записывается в вышеупомянутые матрицы.

Также описывается матрица стандартизированных исходных данных `dataIn_stand` и матрица стандартизированных значений результирующего признака `Y_stand`, если выбран соответствующий вариант стандартизации в элементе `comboBox_standardization`. Для стандартизации Y в целях классификации применяется метод `standardizeMatrixForClassification_Y`.

1.5 Метод чтения файла сохраненных результаты обучения нейросети `readFile_savedNN` так же, как и предыдущий метод, принимает в качестве параметра переменную `filePath`. Метод считывает некоторые параметры сохраненной нейронной сети (число групп при классификации, число входных нейронов, число скрытых нейронов, число выходных нейронов), а также матрицы весов. Перед считыванием матриц весовых коэффициентов из

файла, в методе описывается двумерная матрица весов входного-скрытого слоев W_ih, а также двумерная матрица весов скрытого-выходного слоев W_ho. Далее эти матрицы заполняются соответствующими значениями из файла. После этого считываются все строки файла для их дальнейшего отображения в элементе tabWidget на вкладке «Сводка».

1.6 Метод чтения файла, содержащего значения факторов readFile_X, принимает в качестве параметра переменные filePath, число строк rows и число столбцов cols. В методе описывается двумерная матрица X и заполняется значениями из файла. После этого применяется метод для объявления таблиц и вкладок tabWidget declareTablesForTabWidget_resNN, создается таблица с матрицей X, а в элементе tabWidget создается вкладка «X» через метод makeTableInTabWidget_resNN, куда помещается созданная ранее таблица X. Если нейронная сеть уже была создана или был открыт файл сохраненных результатов обучения, то помимо вкладки «X» создаются вкладки «W_ih», «W_ho», а также вкладка «Сводка».

1.7 Метод объявления таблиц для tabWidget declareTablesForTabWidget_resNN принимает в качестве параметра число объявляемых таблиц tablescount. В методе создается одномерный массив типа QStandardItemModel размером tablescount, содержащий экземпляры моделей данных QStandardItemModel для таблиц. Также создается одномерный массив типа QTableView размером tablescount, содержащий экземпляры таблиц QTableView. Если нейронная сеть уже была создана или был загружен файл сохраненных результатов обучения нейросети, то указанные экземпляры и сами массивы удаляются перед созданием новых. Это делается для очистки выделенной памяти под экземпляры и массивы:

```
//Метод объявление таблиц для tabWidget.
Результаты нейросети
void
NeuralNetworksForm::declareTablesForTabWidget_resNN(int tablescount)
{
    //Удаление старых моделей и таблиц (очистка)
```

```
    if (firstStart_makeNN == 0 or
firstStart_readFile_savedNN == 0)
    {
        for (int i = 0; i < tablesCount_resNN;
i++)
        {
            qSIM_models_resNN[i] -> clear();
//Удаление моделей
            delete qSIM_models_resNN[i];
//Удаление моделей
            delete qTV_tables_resNN[i];
//Удаление таблиц
        }
        delete [] qSIM_models_resNN;
//Удаление массива моделей
        delete [] qTV_tables_resNN;
//Удаление массива таблиц

        ui->tabWidget->clear();
    }

    //Создание моделей для таблиц
    qSIM_models_resNN = new QStandardItemModel*
[tablescount]; //Создание массива моделей для
таблиц
    qTV_tables_resNN = new QTableView*
[tablescount];          //Создание массива
таблиц
    for (int i = 0; i < tablescount; i++)
    {
        qSIM_models_resNN[i] = new
QStandardItemModel;  //Создание моделей
        qTV_tables_resNN[i] = new QTableView;
//Создание таблиц
    }

    tabIdxCounter_tables_resNN = 0; //Обновление
счетчика
    tablesCount_resNN = tablescount;
}
```

1.8 Метод создания таблицы и размещения ее на элементе tabWidget makeTableInTabWidget_resNN принимает 4 параметра (последние 2 –необязательные): двумерную матрицу данных для таблицы matrix1, название вкладки tabTitle, точность данных (для округления) precision, список названий столбцов в таблице colsNames. В методе происходит заполнение экземпляра модели QStandardItemModel данными из матрицы matrix1, установка модели в качестве источника данных для экземпляра таблицы QTableView, добавление новой вкладки на элементе tabWidget, содержащей таблицу:

```
//Метод создания таблицы и размещения ее на
tabWidget
void
NeuralNetworksForm::makeTableInTabWidget_resNN(Q
Vector<QVector<double>> matrix1, QString
tabTitle, int precision, QStringList colsNames)
{
    int n = matrix1.size();      //Строк в
matrix1
    int m = matrix1[0].size();  //Столбцов в
matrix1
    int tabIdx = tabIdxCounter_tables_resNN;
    tabIdxCounter_tables_resNN += 1;

    //Представление матрицы в виде модели данных
    qSIM_models_resNN[tabIdx]->setRowCount(n);
    qSIM_models_resNN[tabIdx]-
>setColumnCount(m);

    if (precision == 0) //Если точность не
указана
    {
        for (int i = 0; i < n; i++)
        {
            for (int j = 0; j < m; j++)
                qSIM_models_resNN[tabIdx]-
>setData(qSIM_models_resNN[tabIdx]->index(i, j),
matrix1[i][j]);
```

```
        }
    }
    else  //Если точность указана
    {
        for (int i = 0; i < n; i++)
        {
            for (int j = 0; j < m; j++)
                qSIM_models_resNN[tabIdx]-
>setData(qSIM_models_resNN[tabIdx]->index(i, j),
QString::number(matrix1[i][j], 'f', precision));
        }
    }

    if (colsNames[0] != nullptr) //Если список
названий столбцов указан
    {
        for (int i = 0; i < colsNames.size();
i++)
            qSIM_models_resNN[tabIdx] ->
setHeaderData(i, Qt::Horizontal, colsNames[i]);
    }

    //Установка модели данных в таблицу
    qTV_tables_resNN[tabIdx] ->
setModel(qSIM_models_resNN[tabIdx]);
    qTV_tables_resNN[tabIdx] ->
horizontalHeader() ->
setSectionResizeMode(QHeaderView::ResizeToConten
ts);
    qTV_tables_resNN[tabIdx] -> verticalHeader()
->
setSectionResizeMode(QHeaderView::ResizeToConten
ts);

    //Установка таблицы на tabWidget
    ui -> tabWidget ->
addTab(qTV_tables_resNN[tabIdx], tabTitle);
```

```
    ui -> tabWidget -> setTabToolTip(tabIdx,
"Строк:\t    " + QString::number(n) +
"\nСтолбцов: " + QString::number(m));
}
```

1.9 Метод создания списка названий столбцов для таблицы X `get_XNames` создает список названий столбцов X1, X2, X3 и т.д. в зависимости от числа столбцов в матрице факторов X.

1.10 Метод стандартизации исходных данных `standardizeDataIn` создает матрицу и заполняет ее стандартизированными значениями переменной Y в диапазоне [0; 1] с помощью метода 2.10 `standardizeMatrixForClassification_Y`.

Методы группы «Матрицы» выполняют специфические операции с матрицами. Такие операции связаны с нейронными сетями, поэтому они не реализованы в классе операций с матрицами `MatrixOp`.

К методам группы «Матрицы» относятся: функция активации (сигмоида) нейронной сети 2.1 `sigmoidFunction`, функция активации (сигмоида) нейронной сети для матрицы 2.2 `sigmoidFunction_matrix`, производная от функции активации (сигмоиды) 2.3 `deriv_sigmoidFunction`, производная от функции активации (сигмоиды) для матрицы 2.4 `deriv_sigmoidFunction_matrix`, метод расчета ошибки прогноза для двух векторов 2.5 `errorFunction`, метод расчета ошибки прогноза для двух матриц 2.6 `errorFunction_matrix`, метод расчета среднеквадратической ошибки прогноза для двух матриц 2.7 `getMSE_matrix`, метод расчета средней относительной ошибки прогноза (в %) для двух матриц 2.8 `getMAPE_matrix` и другие методы.

2.9 Метод округления прогнозных значений нейросети для классификации `roundMatrixForClassification` имеет 3 параметра: матрицу для округления `matrix_for_round`, матрицу-эталон `matrix_etalon`, указатель на матрицу-результат `res_matrix`. В методе определяется число групп и номера групп по данным из матрицы-эталона `matrix_etalon`, то есть `matrix_etalon` содержит уникальные номера групп в виде целых чисел. Далее значения матрицы `matrix_for_round`, которая содержит выходные сигналы нейронной сети в виде вещественных

чисел в диапазоне [0; 1] и число элементов которой равно числу объектов в наборе исходных данных, округляются и приводятся к целочисленным значениям в соответствии со значениями матрицы-эталона matrix_etalon.

Например, если число групп равно 3, то матрица matrix_etalon содержит числа 0, 1, 2. Пусть матрица matrix_for_round содержит результаты применения нейронной сети, тогда значения в данной матрице будут находиться в диапазоне от 0 до 1. Пограничные значения, в соответствии с которыми происходит разделение на группы, для матрицы matrix_for_round будут равны: 1/3 = 0,33 для группы 0, 0,33 · 2 = 0,66 для группы 1, 0,33 · 3 = 1 для группы 2. Теперь мы можем все значения, которые меньше 0,33 отнести к группе 0, все значения, меньшие 0,66 – к группе 1, а все оставшиеся значения будут меньше 1 и больше 0,66, и будут отнесены к последней группе 2.

2.10 Метод стандартизации матрицы Y для классификации standardizeMatrixForClassification_Y по аналогии с предыдущим методом имеет также 3 параметра: матрицу для стандартизации matrix_for_stand, матрицу-эталон matrix_etalon, указатель на матрицу-результат res_matrix. Здесь происходит создание матрицы стандартизированных номеров групп со значениями в диапазоне [0; 1]. Например, для групп 0; 1; 2 стандартизированные номера групп будут 0; 0,5; 1, для групп 0; 1; 2; 3 – 0; 0,33; 0,66; 1. Далее исходный номер группы в матрице matrix_for_stand представляется в стандартизированном виде и записывается в матрицу res_matrix. Таким образом, значения исходной матрицы Y стандартизируются и записываются в матрицу Y_stand, которая будет использоваться для обучения нейронной сети:

```
//Метод стандартизации Y для классификации
(получение Y станд.)
void
NeuralNetworksForm::standardizeMatrixForClassifi
cation_Y(QVector<QVector<double>>
matrix_for_stand, QVector<QVector<double>>
matrix_etalon, QVector<QVector<double>>*
res_matrix)
```

```
{
    int n = matrix_for_stand.size();     //Строк
в matrix_for_stand
    int m = matrix_for_stand[0].size();
//Столбцов в matrix_for_stand

    if (res_matrix->size() > 0) //Перезаполнение
матрицы
        matrixOp.clearMatrix(res_matrix);

    //Описание матрицы-результата
    res_matrix -> reserve(n);
    for (int i = 0; i < n; i++)
        {res_matrix->append(QVector<double>());
(*res_matrix)[i].reserve(m);}

    //Определение числа групп в Y
    QVector<int> groups_sort;
    if (groupsCount_classificationNN == 0)
    {
        QVector<int> groups;
        int val;
        for (int i = 0; i <
matrix_etalon.size(); i++)
        {
            for (int j = 0; j <
matrix_etalon[0].size(); j++)
            {
                val =
static_cast<int>(matrix_etalon[i][j]);
                if (!groups.contains(val))
                {groups.append(val);
groups_sort.append(val);}
            }
        }
        groupsCount_classificationNN =
groups_sort.size(); //Сохранение числа групп
    }
    else
```

```
    {
        for (int i = 0; i <
groupsCount_classificationNN; i++)
            groups_sort.append(i);
    }

    //Упорядочивание групп
    for (int k = 0; k < groups_sort.size() - k;
k++)
    {
        for (int i = 0; i < groups_sort.size() -
1; i++)
        {
            if (groups_sort[i] > groups_sort[i +
1])
                qSwap(groups_sort[i],
groups_sort[i + 1]);
        }
    }

    //Создание вектора стандартизированных
номеров групп  (для 2 групп: 0, 1; для 3 групп:
0, 0.5, 1; для 4 групп: 0, 0.33, 0.66, 1; для 5
групп: 0, 0.25, 0.5, 0.75, 1)
    QVector<double> groups_sort_stand;
    double step = double(1) /
(double(groupsCount_classificationNN) - 1);

    ui->plainTextEdit->appendPlainText("step " +
QString::number(step));


groups_sort_stand.reserve(groupsCount_classifica
tionNN);
    for (int i = 0; i <
groupsCount_classificationNN; i++)
    {
        groups_sort_stand.append(step * i);
    }
```

```
    //Заполнение матрицы-результата
    double group = 0;
    for (int i = 0; i < n; i++)
    {
        for (int j = 0; j < m; j++)
        {
            for (int g = 0; g <
groupsCount_classificationNN; g++)
            {
                if (matrix_for_stand[i][j] ==
groups_sort[g])
                    group =
groups_sort_stand[g];
            }
            (*res_matrix)[i].append(group);
        }
    }
}
```

2.11 Метод расчета уровня соответствия двух матриц `getMatchingLvlForPredict_matrix` имеет 2 параметра: матрицу `Y` и матрицу `Y_pred`. Метод рассчитывает, сколько значений одинакового для двух матриц индекса из всех значений одной матрицы совпали со значениями другой матрицы. Далее рассчитывается процент совпадений или точность.

2.12 Метод представления матрицы в виде одной переменной `QString matrixToQString` принимает в качестве параметра матрицу и преобразует ее в строковую переменную, расставляя разделители и знаки переноса строки. Данный метод применяется при сохранении матриц весов в отдельный файл.

2.13 Метод отображения матрицы `showMatrixVec` также принимает в качестве параметра матрицу, преобразует ее значения в текстовый формат и построчно отображает на элементе `plainTextEdit` в области «Промежуточные результаты».

3.1 Группа методов «Нейронные сети» начинается с метода создания нейросети `makeNN`, принимающего в качестве параметров 7 значений: матрицу значений результирующего признака `Y_data`, число входных узлов (или нейронов) `inputnodes`, число скрытых узлов `hiddennodes`, число выходных узлов `outputnodes`, число

эпох обучения `training_epochs`, значение темпа обучения `learning_rate`, логическую переменную для запрета создания новой нейросети и продолжения обучения старой `continueTrainNN`, по умолчанию равную 0.

В методе определяется время построения нейросети с помощью метода C++ `std::chrono::system_clock::now()`, создаются границы начальных значений матриц весов `W_ih` и `W_ho`. Далее матрицы весов описываются и заполняются случайными числами, входящими в установленные ранее границы начальных значений.

После этого создаются 19 матриц, каждая из которых нужна для получения итоговых результатов обучения нейронной сети. В их число, во-первых, входят транспонированные матрицы факторов и результирующего признака: `X_transpose` и `Y_tanspose` (2 матрицы).

Во-вторых, матрицы результата нейросети и ошибок: матрица `H` для входящих сигналов нейронов скрытого слоя, матрица `H_sigmoid` для исходящих сигналов нейронов скрытого слоя *H*, матрица `O` для входящих сигналов нейронов выходного слоя *O*, матрица `O_sigmoid` для исходящих сигналов нейронов выходного слоя, матрица `E_o` для ошибок выходного слоя, транспонированная матрица `W_ho_t` для весов скрытого слоя, матрица `E_h` для ошибок скрытого слоя *H* (7 матриц).

В-третьих, матрицы для обновления весов скрытого-выходного слоев `W_ho`: матрица `O_deriv_sigmoid` для значений производной функции активации на выходном слое, первая матрица `W_ho_delta_1` для промежуточных расчетов изменения `W_ho`, транспонированная матрица `H_sigmoid_t` для исходящих сигналов узлов скрытого слоя, вторая матрица `W_ho_delta_2` для промежуточных расчетов изменения `W_ho`, матрица `W_ho_delta` для значений, на которые изменятся веса `W_ho` (5 матриц).

В-четвертых, матрицы для обновления весов входного-скрытого слоев `W_ih`: матрица `H_deriv_sigmoid` для значений производной функции активации на скрытом слое, первая матрица `W_ih_delta_1` для промежуточных расчетов изменения `W_ih`, вторая матрица `W_ih_delta_2` для промежуточных расчетов изменения `W_ih`, матрица `W_ih_delta` для значений, на которые изменятся веса `W_ih` (4 матрицы).

В-пятых, матрица для расчета среднеквадратической ошибки, представляющая собой транспонированную матрицу выходных сигналов O_sigmoid_t.

Далее идет этап обучения, где в цикле с числом итераций, равным числу эпох обучения, рассчитываются результаты нейронной сети и ошибки, происходит обновление весов W_ho и W_ih, а также на каждой пятидесятой эпохе обучения рассчитывается среднеквадратическая и относительная ошибки прогноза, которые отображаются в области «Промежуточные результаты».

В конце метода происходит создание таблиц на элементе tabWidget в области «Результат построения» с помощью методов 1.8 declareTablesForTabWidget_resNN и 1.9 makeTableInTabWidget_resNN.

3.2 Метод построения прогноза по обученной модели нейросети useNN_feedforward принимает 2 параметра: матрицу факторов X_test и указатель на матрицу-результат Y_pred_NN. В методе по рассчитанным ранее весовым коэффициентам W_ih и W_ho происходит расчет значений результирующей переменной по значениям факторов X_test:

```
//Метод построения прогноза по обученной модели
нейросети
void
NeuralNetworksForm::useNN_feedforward(QVector<QV
ector<double>> X_test, QVector<QVector<double>>*
Y_pred_NN)
{
    int n = X_test.size();       //Строк в X_test
    int m = 1;                   //Столбцов в
Y_pred

    if (Y_pred_NN->size() > 0) //Если Y_pred_NN
не пустая. Случай перезаполнения. Очистка
матрицы
        matrixOp.clearMatrix(Y_pred_NN);

    //Описание матрицы-результата
    Y_pred_NN -> reserve(n);
```

```
    for (int i = 0; i < n; i++)
        {Y_pred_NN->append(QVector<double>());
(*Y_pred_NN)[i].reserve(1);}

    //Расчет прогноза
    QVector<QVector<double>> H;   //Матрица
сигналов скрытого слоя H
    QVector<QVector<double>> O;   //Матрица
сигналов выходного слоя O
    QVector<QVector<double>> X_transpose;

    for (int i = 0; i < n; i++)
    {
        H.append(QVector<double>());
        O.append(QVector<double>());

        H[i].reserve(m);
        O[i].reserve(m);
    }

    matrixOp.transposeMatrix(X_test,
&X_transpose);

    matrixOp.scalarProduct_matrices(W_ih,
X_transpose, &H); //Расчет входных сигналов
скрытого слоя H
    sigmoidFunction_matrix(H, &H);  //Расчет
выходных сигналов скрытого слоя H
    matrixOp.scalarProduct_matrices(W_ho, H,
&O);   //Расчет входных сигналов выходногоо слоя
O
    sigmoidFunction_matrix(O, &O);
//Расчет выходных сигналов выходногоо слоя O

    matrixOp.transposeMatrix(O, &O);

    //Заполнение матрицы-результата
    for (int i = 0; i < n; i++)
    {
```

```
            for (int j = 0; j < m; j++)
                (*Y_pred_NN)[i].append(O[i][j]);
        }
}
```

3.3 Метод сохранения результатов обучения нейронной сети `saveNN` берет данные из вкладки «Сводка» в области «Результат построения» и сохраняет эти данные в отдельный файл.

3.4 Метод для кнопки «Сохранить НС» вызывает метод 3.3 `saveNN`.

Группа методов «Создание нейронной сети. Кнопки» содержит 7 методов для кнопок в области «Построение нейронной сети». В данную группу входят такие методы, как: 4.1 метод установки изначальных параметров области «Построение нейронной сети» `clearComponents_groupBox_makeNN`, 4.2 метод кнопки «Создать» `on_pushButton_makeNewNN_clicked`, 4.3 метод кнопки «Обзор» (тренировочные данные) `on_pushButton_openFile_trainData_clicked`, 4.4 метод при нажатии на кнопку «Обзор» `method__on_pushButton_openFile_trainData_clicked`, 4.5 метод кнопки «Обзор» (тестовые данные) `on_pushButton_openFile_testData_clicked`, 4.6 метод кнопки «Продолжить» `on_pushButton_continueTrainNN_clicked`, 4.7 метод кнопки «Использовать» `on_pushButton_useNN_clicked`.

Группа методов «Создание нейронной сети. Сигналы» состоит из 6 методов, которые вызываются при нажатии пользователем на тот или иной элемент в области «Построение нейронной сети».

В состав последней группы методов входят 6 методов: 5.1 метод при сигнале изменения `comboBox_rows_trainData void on_comboBox_rows_ trainData_activated`, 5.2 метод при сигнале изменения `comboBox_rows_testData on_comboBox_rows_testData_activated`, 5.3 метод при сигнале изменения `checkBox_testAndTrainInOneFile on_checkBox_testAndTrainInOneFile_stateC hanged`, 5.4 метод при сигнале изменения числа строк в `spinBox_rows_trainData on_spinBox_rows_trainData_valueChanged`, 5.5 метод при сигнале изменения числа строк в `spinBox_rows_testData on_spinBox_rows_testData_valueChanged`, 5.6 сигнал выбора вкладки `tabWidget on_tabWidget_tabBarClicked`.

Класс `NeuralNetworkForm` содержит ряд глобальный переменных, например, двумерный динамический массив `QStandardItemModel **qSIM_models_resNN`, содержащий экземпляры класса `QStandardItemModel`, которые являются моделями данных для таблиц. Сами таблицы или экземпляры класса `QTableView` содержаться в массиве `QTableView **qTV_tables_resNN`.

Помимо этого, в число глобальных переменных также входят матрицы данных или векторы векторов `dataIn`, `dataIn_stand`, `X`, `Y`, `Y_stand`, `W_ih`, `W_ho`. Данные матрицы имеют тип `QVector<QVector<double>>`.

3.7 Демонстрация работы модуля проведения корреляционно-регрессионного анализа

Проведем корреляционно-регрессионный анализ двух наборов данных. Анализ набора исходных данных №1 отражен на рисунках 3.9 – 3.18, а анализ набора исходных данных №2 – на рисунках Е.1 – Е.9 приложения Е.

Набор исходных данных №1 состоит из результативной переменной Y и двух факторов X_1 и X_2, и содержит 48 наблюдений (таблица 3.1). Набор исходных данных №1 хранится в базе данных в виде таблицы «Data_Econometric_points».

Таблица 3.1 – Набор исходных данных №1 для корреляционно-регрессионного анализа

№ предприятия п/п	Стоимость валовой продукции сельского хозяйства в расчете на 100 га сельскохозяйственных угодий, тыс. руб. (Y)	Численность работников, занятых в сельскохозяйственном производстве в расчете на 100 га сельскохозяйственных угодий, чел. (X1)	Стоимость основных средств производства в расчете на 100 га сельскохозяйственных угодий, тыс. руб. (X2)
1	4750,9	0,2	33502,3
2	5104,6	0,5	37683,3
3	2802,2	0,7	3718,3
4	5374,6	0,7	7293,5
5	6242,3	0,8	10606,7
…	…	…	…
44	8318,1	4,6	25235,7
45	4245,8	4,8	6031,3
46	8318,9	5,3	11846,3
47	12542,5	5,6	52346,0
48	13027,8	6,8	22700,5

На форме «Корреляционно-регрессионный анализ» («КРА») в поле «Таблица:» выберем таблицу «Data_Econometric_points». Все остальные параметры установятся автоматически, поэтому для анализа данных нажмем на кнопку «Анализ». В качестве результатов анализа сформируются таблицы и графики, в том числе графики переменных во вкладке «Исходные данные» (рисунок 3.9).

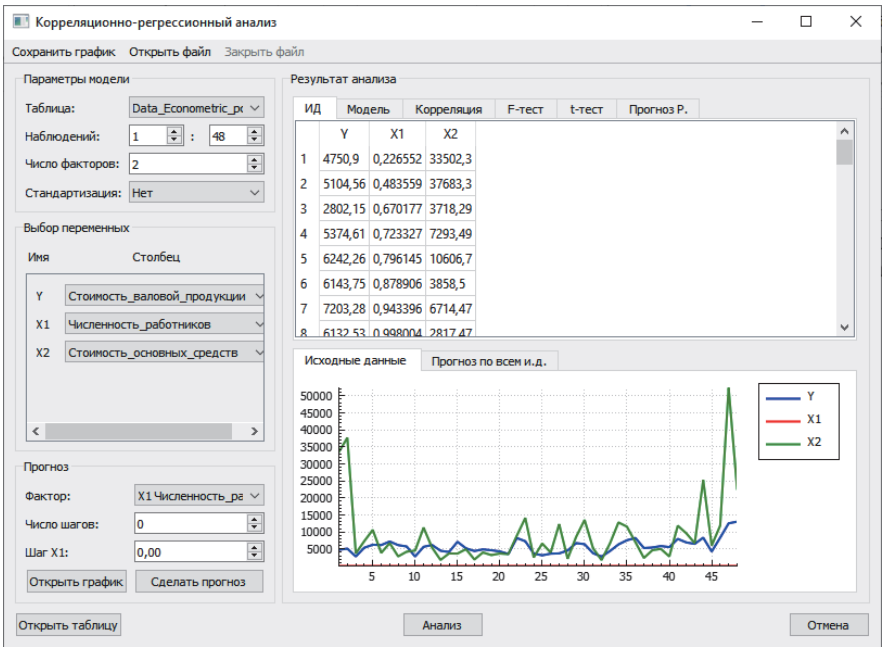


Рисунок 3.9 – Форма «КРА». Вкладка с таблицей «ИД» (исходные данные), вкладка с графиком «Исходные данные»

Во вкладке «Модель» отображается таблица со значениями выбранных для анализа переменных. Так как были выбраны все три переменные, то указанная таблица соответствует таблице вкладки «ИД», содержащей исходные данные по всем переменным, то есть весь набор данных (рисунок 3.10).

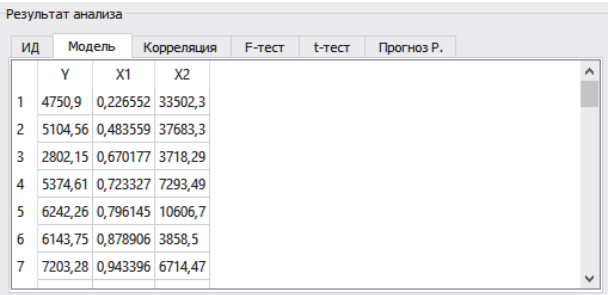


Рисунок 3.10 – Форма «КРА». Вкладка с таблицей «Модель»

Во вкладке «Корреляция» мы видим тепловую карту корреляции в виде таблицы с цветными ячейками, в которых

содержаться значения парных коэффициентов корреляции между переменными. Например, коэффициент корреляции между Y и X_1 равен 0,60, что говорит о средней тесноте связи переменных (рисунок 3.11).

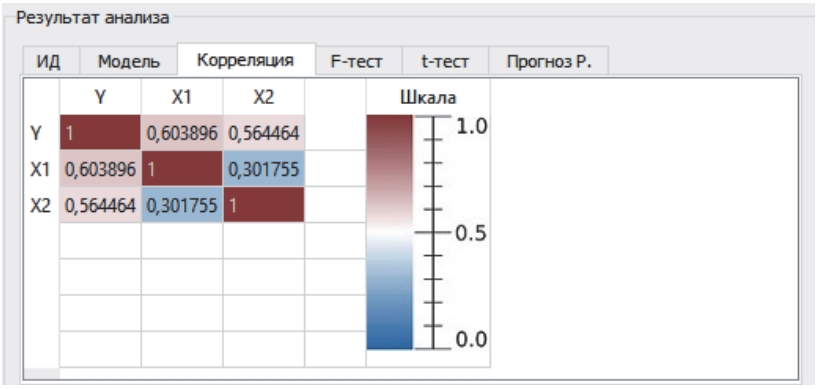


Рисунок 3.11 – Форма «КРА». Вкладка с таблицей «Корреляция»

Во вкладке «F-тест» представлены результаты дисперсионного анализа или теста на статистическую значимость модели множественной регрессии в целом. Здесь мы можем видеть рассчитанные по модели значения таких статистических показателей, как множественный коэффициент детерминации («R squared»), множественный коэффициент корреляции («R»), фактическое значение критерия F-Фишера («F»), его значимость «F prob.» и др. (рисунок 3.12).

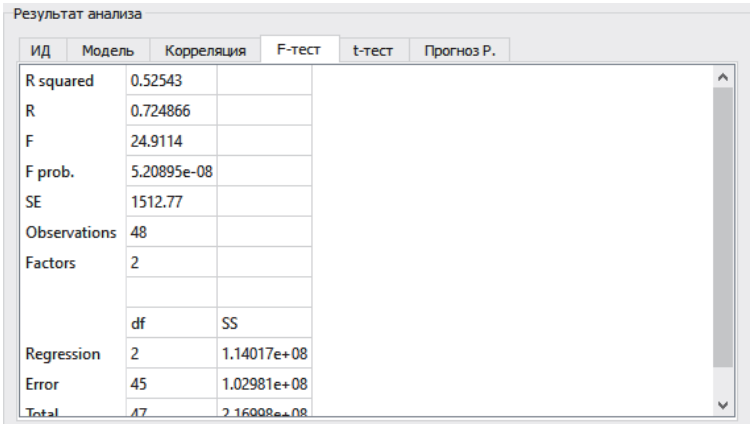
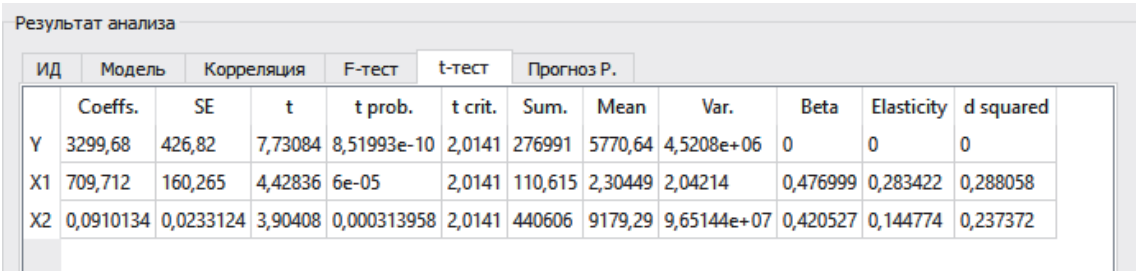


Рисунок 3.12 – Форма «КРА». Вкладка с таблицей «F-тест»

Во вкладке «t-тест» представлены результаты теста на статистическую значимость отдельных параметром модели множественной регрессии, основанного на критерии t-Стьюдента.

В табличном виде представлены оценки параметров или коэффициентов регрессии («Coeffs.»), фактическое значение критерия t-Стьюдента для каждой оценки («t»), его значимость («t prob.») и др. (рисунок 3.13).

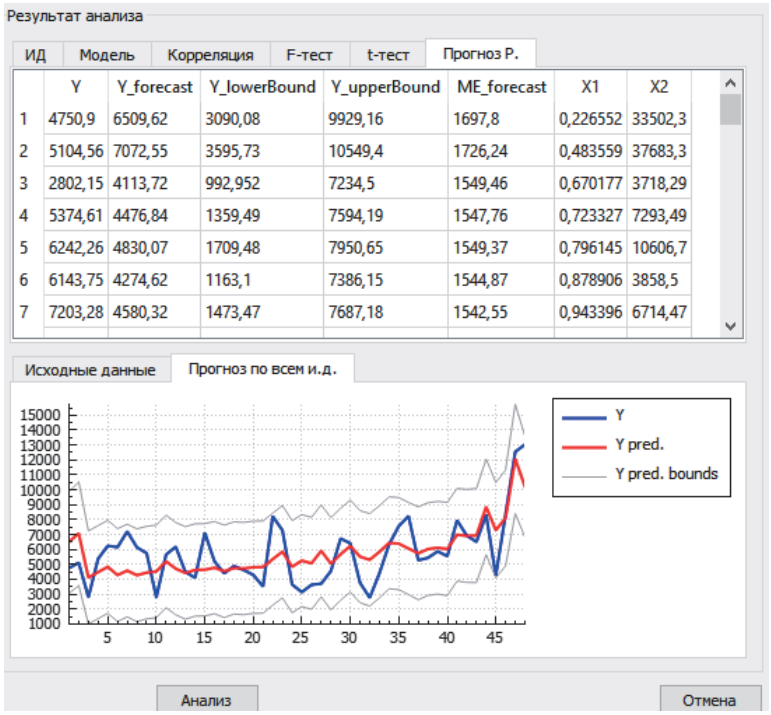

Результат анализа

ИД | Модель | Корреляция | F-тест | t-тест | Прогноз Р.

	Coeffs.	SE	t	t prob.	t crit.	Sum.	Mean	Var.	Beta	Elasticity	d squared
Y	3299,68	426,82	7,73084	8,51993e-10	2,0141	276991	5770,64	4,5208e+06	0	0	0
X1	709,712	160,265	4,42836	6e-05	2,0141	110,615	2,30449	2,04214	0,476999	0,283422	0,288058
X2	0,0910134	0,0233124	3,90408	0,000313958	2,0141	440606	9179,29	9,65144e+07	0,420527	0,144774	0,237372

Рисунок 3.13 – Форма «КРА». Вкладка с таблицей «t-тест»

Во вкладке «Прогноз Р.» находится таблица с расчетными значениями для построения прогноза на текущий временной промежуток или на текущие наблюдения (рисунок 3.14). Данный прогноз представляет собой смоделированные значения переменной *Y* (столбец «Y-forecast») по уравнению регрессии. Вкладка «Прогноз по всем и.д.» содержит график, визуализирующий основные столбцы таблицы.


Результат анализа

ИД | Модель | Корреляция | F-тест | t-тест | Прогноз Р.

	Y	Y_forecast	Y_lowerBound	Y_upperBound	ME_forecast	X1	X2
1	4750,9	6509,62	3090,08	9929,16	1697,8	0,226552	33502,3
2	5104,56	7072,55	3595,73	10549,4	1726,24	0,483559	37683,3
3	2802,15	4113,72	992,952	7234,5	1549,46	0,670177	3718,29
4	5374,61	4476,84	1359,49	7594,19	1547,76	0,723327	7293,49
5	6242,26	4830,07	1709,48	7950,65	1549,37	0,796145	10606,7
6	6143,75	4274,62	1163,1	7386,15	1544,87	0,878906	3858,5
7	7203,28	4580,32	1473,47	7687,18	1542,55	0,943396	6714,47

Исходные данные | Прогноз по всем и.д.

Анализ | Отмена

Рисунок 3.14 – Форма «КРА». Вкладка с таблицей «Прогноз Р.», вкладка с графиком «Прогноз по всем и.д.»

Сделаем прогноз для 10 предприятий, предположив, что фактор X1 для каждого нового предприятия будет увеличиваться на единицу. Для этого установим соответствующие параметры в области «Прогноз» и нажмем кнопку «Сделать прогноз» (рисунок 3.15). После этого будут созданы дополнительные таблицы и графики, а график на вкладке «Прогноз по всем и.д.» дополнится прогнозом, изображенным пунктирной линией.

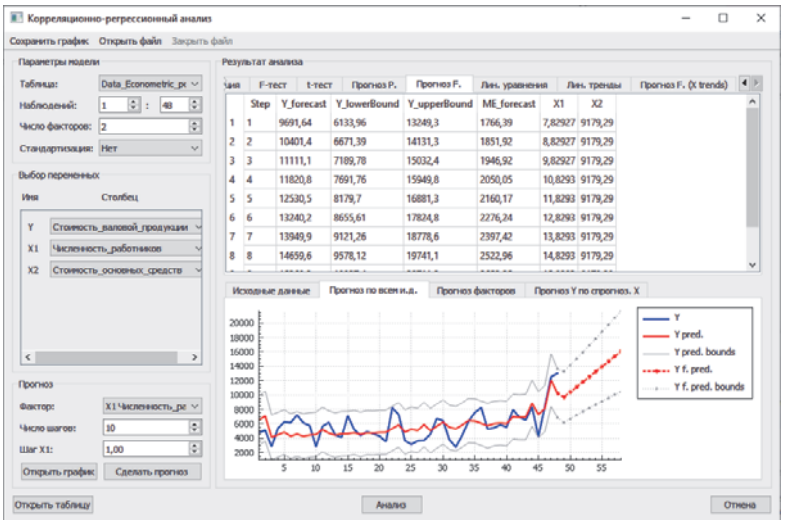


Рисунок 3.15 – Форма «КРА». Нажатие на кнопку «Сделать прогноз». Вкладка с таблицей «Прогноз F.», обновленная вкладка с графиком «Прогноз по всем и. д.»

Вкладка с таблицей «Лин. уравнения» содержит таблицу коэффициентов детерминации и оценок коэффициентов линейного уравнения каждой переменной с фактором времени (рисунок 3.16).

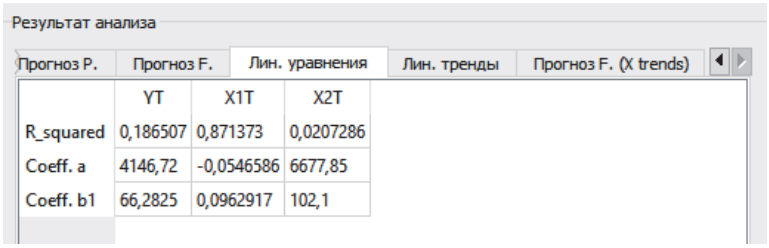


Рисунок 3.16 – Форма «КРА». Вкладка с таблицей «Лин. уравнения»

С помощью данных линейных уравнений строятся тренды для каждой переменной, включенной в модель (рисунок 3.17).

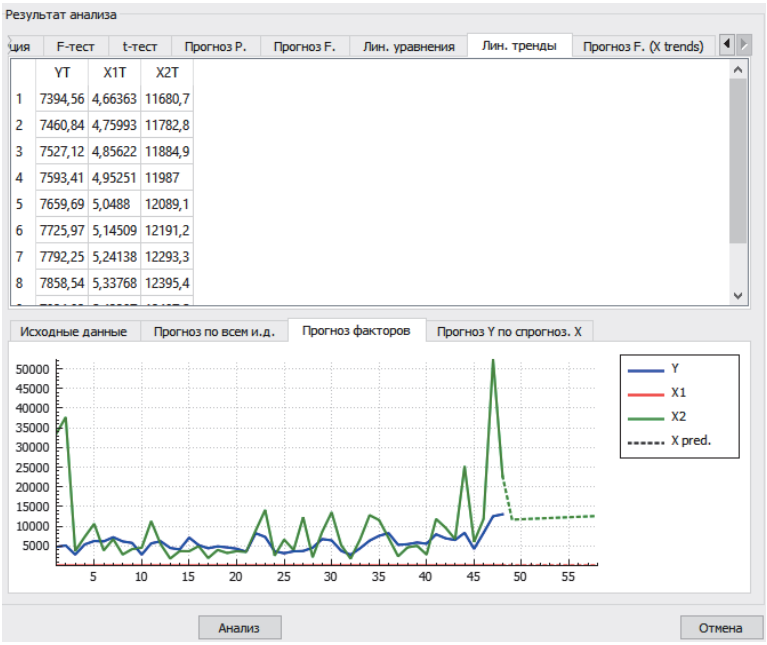


Рисунок 3.17 – Форма «КРА». Вкладка с таблицей «Лин. тренды», вкладка с графиком «Прогноз факторов», отображающим прогноз факторов на 10 шагов вперед на основе линейных трендов

На основе линейных трендов факторных переменных строится второй прогноз, который учитывает только число шагов, установленное в области «Прогноз» (рисунок 3.18).

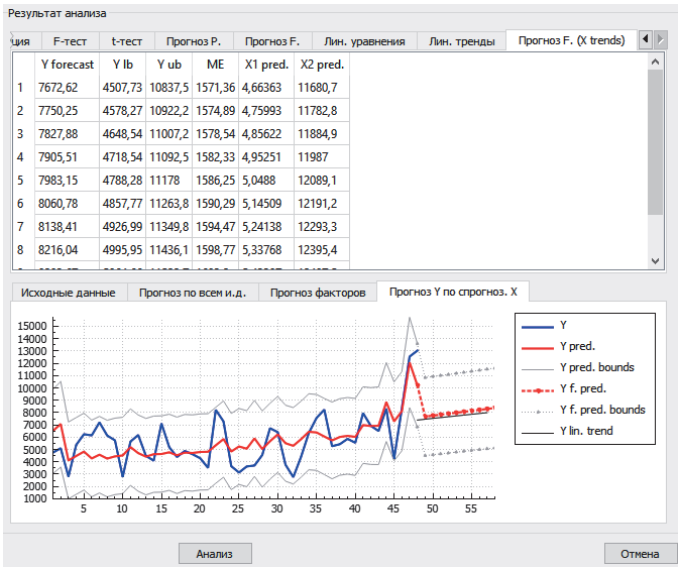


Рисунок 3.18 – Форма «КРА». Вкладка с таблицей «Прогноз F. (X trends)», вкладка с графиком «Прогноз Y по спрогнозированным X», отображающим прогноз результативной переменной Y на 10 шагов вперед на основе спрогнозированных по линейным трендам факторов

Результаты корреляционно-регрессионного анализа набора исходных данных №1 (рисунки 3.12, 3.13) совпадают с результатами такого же анализа, проведенного в Microsoft Excel (рисунок 3.19).

ВЫВОД ИТОГОВ

Регрессионная статистика

Множеств	0,724865926
R-квадрат	0,525430611
Нормиро	0,504338638
Стандарт	1512,766267
Наблюде	48

Дисперсионный анализ

	df	*SS*	*MS*	*F*	*ачимость F*
Регресси	2	114017582,1	57008791,03	24,91140186	5,21E-08
Остаток	45	102980780,1	2288461,779		
Итого	47	216998362,1			

	Коэффициент	*Стандартная ошибка*	*t-статистика*	*P-Значение*	*ижние 95%*	*рхние 95%*
Y-пересе	3299,675143	426,8201641	7,730832376	8,52E-10	2440,015	4159,335
Перемен	709,7123633	160,2653045	4,428359372	5,99997E-05	386,9215	1032,503
Перемен	0,091013516	0,023312376	3,904085701	0,000313954	0,04406	0,137967

Рисунок 3.19 – Результаты корреляционно-регрессионного анализа набора исходных данных №1 в Excel

Набор исходных данных №2 содержится в файле Excel (рисунок 3.20) и отражает результат сельскохозяйственной деятельности тепличного комбината, а также различные факторы, влияющие на результат.

	A	B	C	D	E	F	G	H	I	J	K
1	Валовой с	Относит в	% развери	Метеобо	Метеобо	Средняя т	масса мат	Влажност	ЕС мата,м	Интенсив	Естеств осв
2	850	53,33215	72,21205	22,2324	22,1876	22,20535	0	0	0,361215	37,40972	206,6423
3	4206	51,4308	76,53795	24,58986	24,51708	24,55049	10,65406	29,32903	0,322778	23,89563	169,9622
4	5031	60,1841	52,61354	23,02396	22,82653	22,92306	38,29198	100	0,1	71,77778	348,1429
5	5533	68,10375	69,72222	23,13753	22,96802	23,04872	42,39649	100	0,1	26,35549	245,4789
6	6140	70,38698	26,78024	22,53566	22,48503	22,50507	39,28944	100	0,073333	112,5521	510,6104
7	4964	70,67528	0	22,57663	22,35934	22,46622	35,65764	99,91056	0,068681	23,97712	312,9182
8	4483	70,56646	0	23,01104	22,85215	22,92465	33,83677	96,24278	0,1	49,38538	236,9665
9	3393	69,18674	0	22,71674	22,58101	22,64865	32,5616	92,44559	0,1	61,27885	332,195
10	4163	68,20774	0	22,44278	22,32688	22,38215	32,00215	90,78111	0,1	39,03601	278,7638
11	8250	66,22208	9,491667	23,14274	22,91372	23,02781	30,90701	87,52059	0,1	93,33028	446,4759
12	11689	64,88528	23,9976	22,66104	22,51986	22,58458	30,80493	87,20403	0,1	100,7412	605,0255
13	13458	69,12222	8,76184	22,95135	22,80903	22,8749	30,71938	86,95566	0,1	163,0117	880,0536
14	13288	70,41688	0	23,08399	22,93983	23,0142	30,58552	86,54701	0,100347	155,5134	952,9559

Sheet1

Рисунок 3.20 – Набор исходных данных №2 для корреляционно-регрессионного анализа

Набор исходных данных №2 состоит из 134 наблюдений по 11 показателям:

Y – Валовой сбор огурцов, кг.

X_1 – Относительная влажность воздуха, %.

X_2 – Развернутость шторы, %.

X_3 – Температура в метеобоксе 1, ^oC.

X_4 – Температура в метеобоксе 2, °С.

X_5 – Средняя температура воздуха в климатической зоне, °С.

X_6 – Масса мата, кг.

X_7 – Влажность мата, %.

X_8 – Электропроводность воды в мате, мСм.

X_9 – Интенсивность освещения, Вт/м2.

X_{10} – Естественная освещенность, Дж/см2.

Анализ набора исходных данных №2 с демонстрацией дополнительных возможностей представлен в приложении Е.

3.8 Демонстрация работы модуля построения нейронных сетей

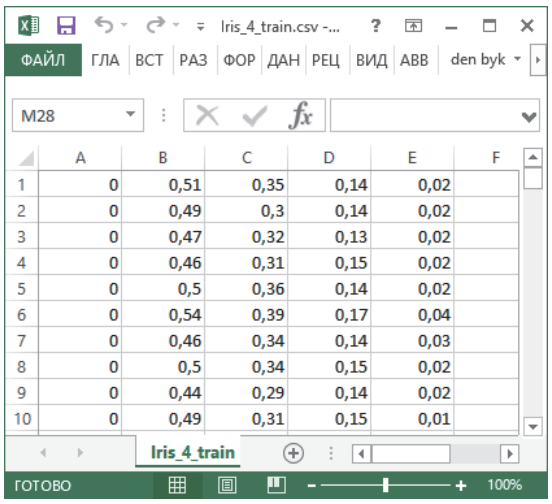
Модуль построения нейронных сетей позволяет строить, сохранять и использовать построенные нейронные для проведения классификации с помощью формы «Нейронные сети» (рисунок 3.8).

Проведем классификацию для набора данных Iris Data Set [17], содержащим 150 наблюдений для 3 видов растений iris – setosa, versicolor, virginica. К каждому виду растения относится 50 наблюдений. Каждое наблюдение имеет 5 признаков: длина чашелистика, ширина чашелистика, длина лепестка, ширина лепестка, вид растения.

Подготовим исходные данные для нейронной сети. Для этого изменим названия видов растения, присвоив номер 0 виду setosa, 1 – versicolor, 2 – virginica. Помимо этого, переместим столбец видов с конца в начало, чтобы данный столбец был первым. Наконец, все данные, кроме вида, разделим на 10, чтобы значения признаков входили в диапазон от 0 до 1.

Удалим по 2 последних наблюдения для каждого вида и сохраним их в отдельный файл, чтобы проверить точность классификации обученной нейронной сети. В данном файле удалим столбец вида, так как нейронная сеть должна будет сама определить вид растения.

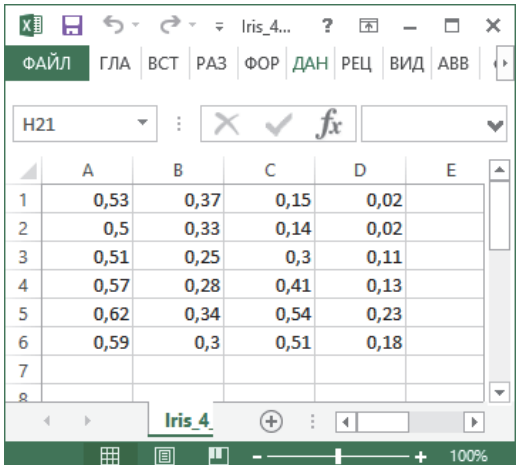
В результате получим следующий файл исходных данных для обучения, содержащий 144 наблюдения (рисунок 3.21).



	A	B	C	D	E	F
1	0	0,51	0,35	0,14	0,02	
2	0	0,49	0,3	0,14	0,02	
3	0	0,47	0,32	0,13	0,02	
4	0	0,46	0,31	0,15	0,02	
5	0	0,5	0,36	0,14	0,02	
6	0	0,54	0,39	0,17	0,04	
7	0	0,46	0,34	0,14	0,03	
8	0	0,5	0,34	0,15	0,02	
9	0	0,44	0,29	0,14	0,02	
10	0	0,49	0,31	0,15	0,01	

Рисунок 3.21 – Подготовленные исходные данные для обучения нейронной сети

Данные, на основе которых будет использоваться обученная нейронная сеть, представлены на рисунке 3.32. Если нейронная сеть проводит классификацию корректно, то результат для приведенных на рисунке наблюдений будет следующим: 0, 0, 1, 1, 2, 2.



	A	B	C	D	E
1	0,53	0,37	0,15	0,02	
2	0,5	0,33	0,14	0,02	
3	0,51	0,25	0,3	0,11	
4	0,57	0,28	0,41	0,13	
5	0,62	0,34	0,54	0,23	
6	0,59	0,3	0,51	0,18	
7					

Рисунок 3.22 – Подготовленные факторные данные для использования обученной нейронной сети

На форме «Нейронные сети» нажмем на кнопку «Обзор» и выберем файл для обучения нейронной сети (тренировочные данные). Кнопка «Обзор» изменит свое название на название выбранного файла. Так как по умолчанию стоит отметка на элементе «Один файл», то второй файл с тестовыми данными будет таким же и отдельно выбирать его не придется. Нажмем на кнопку «Создать» для создания и обучения нейронной сети. Результаты представлены на рисунках 3.23, 3.24.

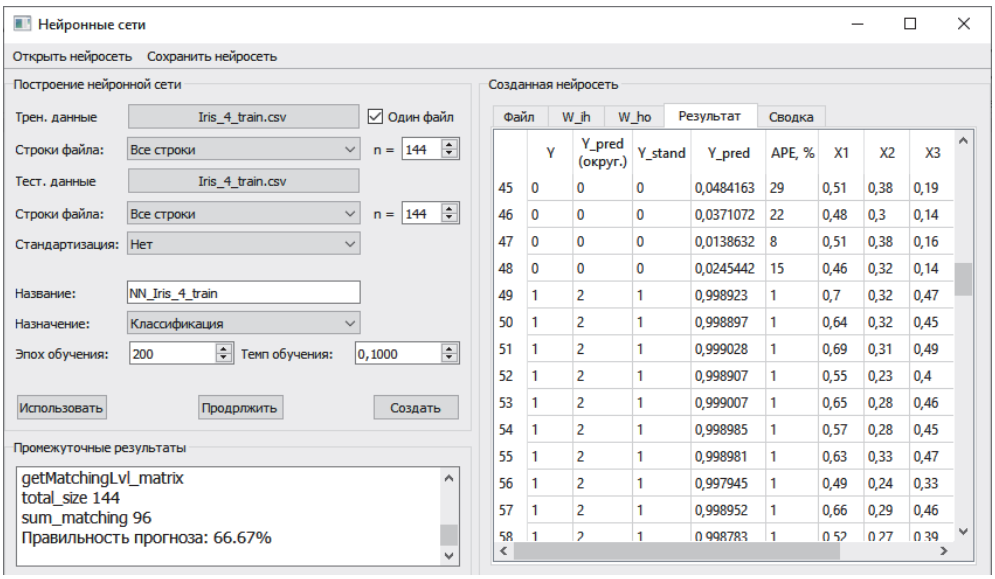


Рисунок 3.23 – Область «Созданная нейросеть». Вкладка «Результат», содержащая фактические значения результирующего признака Y и значения, полученные по модели нейронной сети («Y_pred (округ.)» и «Y_pred»)

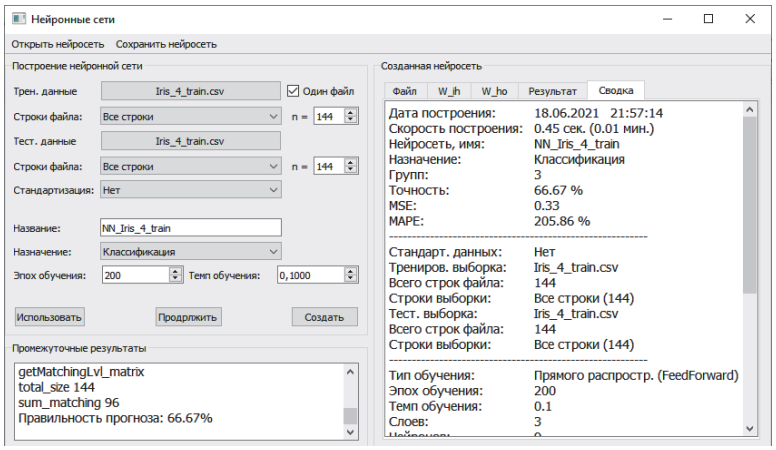


Рисунок 3.24 – Область «Созданная нейросеть». Вкладка «Сводка», содержащая основную информацию про построенную нейронную сеть

Примечание: все элементы в области «Построение нейронный сети» выставлены по умолчанию, кроме кнопки «Обзор» (или «Iris_4_train.csv»).

Мы провели обучение нейронной сети за 200 эпох обучения с темпом обучения 0,1. Точность построенной нейронной сети составила 66,67 %, то есть из всех значений Y только 66,67 % отнесены к правильному классу.

Теперь используем данную нейронную сеть, чтобы провести классификацию интересующих нас растений и понять, к какому

виду они относятся, предполагая, что мы не знаем этой информации.

Нажмем на кнопку «Использовать» и выберем файл с факторными данными. Область «Создание нейросети» автоматически сменит название на «Результат использования нейросети» (рисунок 3.25).

Результат использования нейронной сети отражен в столбце «Y_pred», который содержит значения результирующего признака, то есть вида растения. Столбец «Y_pred» имеет диапазон значений от 0 до 1 в соответствии с особенностью построенной нейронной сети, поэтому значения данного столбца округляются и приводятся к значениям 0, 1 или 2, в соответствии с видом растения. Столбец с округленными значениями называется «Y_pred (округ.)».

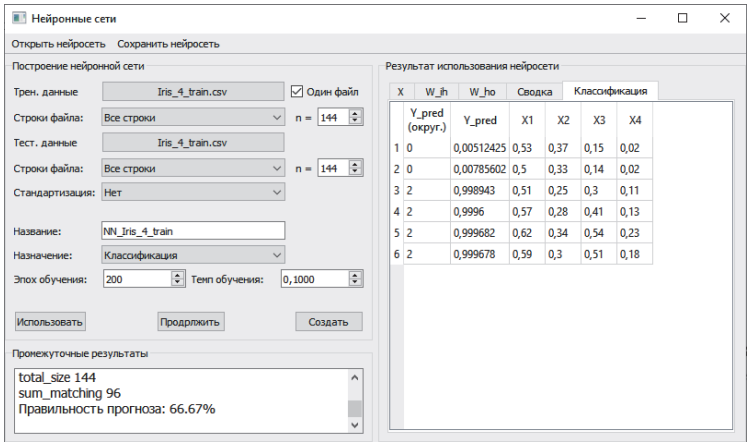


Рисунок 3.25 – Область «Результат использования нейросети». Вкладка «Классификация», содержащая результат применения нейронной сети для классификации объектов по загруженным ранее факторам X

Итак, в результате использования нейронной сети мы классифицировали 6 наблюдений по следующим группам: 0, 0, 2, 2, 2, 2. Мы знаем, что верная классификация будет следующая: 0, 0, 1, 1, 2, 2. То есть, можно заключить, что нейронная сеть классифицирует объекты недостаточно качественно: из 6 объектов только 4 были классифицированы верно.

Можно попытаться обучить нейронную сеть еще лучше, добившись более высокой точности классификации. Для этого имеет смысл увеличить число эпох обучения и изменять значение темпа обучения, пока вновь построенная нейронная сеть не покажет более высокую точность. Однако существенного

увеличения качества классификации можно добиться путем стандартизации столбца Y (рисунки 3.26, 3.27).

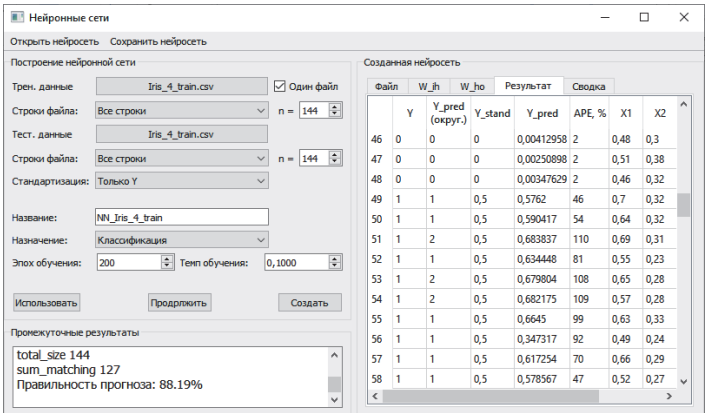


Рисунок 3.26– Построение новой нейронной сети, выбрав опцию стандартизации столбца Y. Открытие вкладки «Результат»

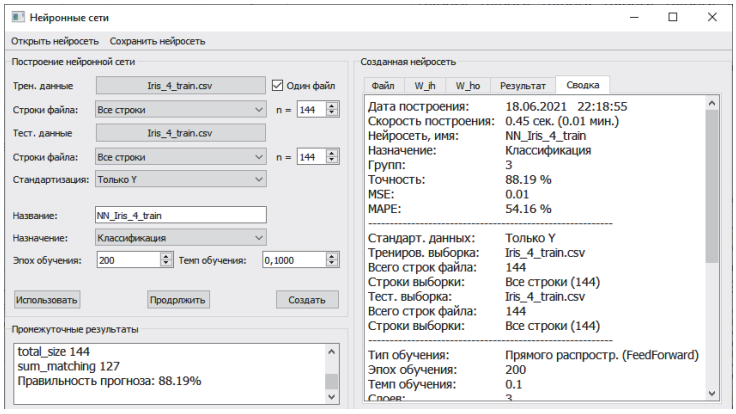


Рисунок 3.27 – Построение новой нейронной сети, выбрав опцию стандартизации столбца Y. Открытие вкладки «Сводка»

По рисунку 3.27 мы можем видеть, что качество классификации увеличилось до 88,19%.

Выводы. В данной главе изложены теоретические аспекты проведения корреляционно-регрессионного анализа и построения искусственных нейронных сетей, а также освещен процесс разработки информационной системы, позволяющей реализовывать функции учета и анализа данных.

Модуль взаимодействия с базами данных позволяет подключаться к различным базам данных Microsoft SQL Server, обеспечивает возможности создания новых таблиц или изменения свойств уже созданных, удаления таблиц, импорта данных из файла в базу данных, изменения данных в таблице.

Модуль проведения корреляционно-регрессионного анализа обеспечивает быстрый и простой для пользователя анализ данных, расположенных как в таблице базы данных, так и в файле. В результате анализа создается тепловая карта парных коэффициентов корреляции переменных, формируется модель множественной линейной регрессии, проводится оценка качества модели и ее параметров, строятся прогнозы: прогноз зависимой переменной при изменении одного из факторов, прогноз зависимой переменной при спрогнозированных значениях всех факторов на основе их линейных трендов. Рассчитанные данные распределяются по десяти таблицам, образуя результаты корреляционно-регрессионного анализа. Данные из пяти таких таблиц визуализируются в виде графиков, для которых разработаны инструменты сохранения и открытия в отдельном окне.

Модуль построения нейронных сетей предоставляет возможность конструировать и обучать нейронные сети, устанавливая такие параметры, как число эпох обучения, темп обучения, стандартизация исходных данных. Построенную нейронную сеть имеется возможность сохранить и загрузить для использования на определенном наборе проверочных данных. С ее помощью можно достаточно точно классифицировать значения результативной переменной по значениям факторных переменных.

Библиографический список к главе 3

1. Боровская, Е. В. Основы искусственного интеллекта: учебное пособие / Е. В. Боровская, Н. А. Давыдова. – 4-е изд., электрон. – М.: Лаборатория знаний, 2020. – 130 с. – Текст: электронный.

2. Бураков, М.В. Нейронные сети и нейроконтроллеры: учеб. пособие / М.В. Бураков. – СПб.: ГУАП, 2013. – 284 с. – Текст: непосредственный.

3. Быков Д.В. Разработка информационной системы учета и обработки данных с поддержкой проведения статистического анализа на C++ / Д.В. Быков, А.В. Уколова // Электронный сетевой политематический журнал «Научные труды КубГТУ». – 2022. – № 4. – С. 32-45. – URL: https://elibrary.ru/item.asp?id =49370090. – Текст: электронный.

4. Зинченко, А.П. Математическая статистика: учебник / А. П. Зинченко, М.В. Кагирова, Ю.Н. Романцева [и др.]; под ред. А.П. Зинченко и А.В. Уколовой. – Москва: РГАУ-МСХА имени К. А. Тимирязева, 2018. – 199 с. – Текст: непосредственный.

5. Кремер, Н.Ш. Эконометрика: учебник и практикум для вузов / Н.Ш. Кремер, Б.А. Путко; под редакцией Н.Ш. Кремера. – 4-е изд., испр. и доп. – Москва: Юрайт, 2021. – 308 с. – URL: https://urait.ru/bcode/468442. – Текст: электронный

6. Остроух, А. В. Системы искусственного интеллекта: монография / А. В. Остроух, Н. Е. Суркова. – 2-е изд., стер. – Санкт-Петербург: Лань, 2021. – 228 с. – URL: https://e.lanbook.com/book/176662. – Текст: электронный

7. Рашид, Т. Создаем нейронную сеть: Пер. с англ. – СПб.: ООО «Альфа-книга», 2017. – 272 с. – Текст: непосредственный.

8. Хайкин, С. Нейронные сети: полный курс, 2-е издание: Пер. с анг. – М.: Издательский дом «Вильямс», 2006. – 1104 с. – Текст: непосредственный.

9. Чубукова, И.А. Data Mining: учебное пособие / Чубукова И.А. – Москва: Интуит НОУ, 2016. – 470 с. – URL: https://book.ru/book/917500. – Текст: электронный.

10. Шайкина, Е.В. Эконометрика. Курс лекций: Учеб. пособие / Е.В. Шайкина, А.В. Уколова. – М.: РГАУ-МСХА, 2009. – 119 с. – Текст: непосредственный.

11. Abdalrahman, G. Modeling the infiltration rate of wastewater infiltration basins considering water quality parameters using different artificial neural network techniques / G. Abdalrahman, Sai Hin Lai, P. Kumar, et al. // Engineering Applications of Computational Fluid Mechanics. – 2022. – vol. 16 (1). – pp. 397-421. – URL: https://doi.org/10.1080/19942060.2021.2019126. – Текст: электронный.

12. Aggarwal, C.C. Outlier Analysis / C.C. Aggarwal. – Springer International Publishing AG, 2017. – 465 с. – Текст: непосредственный.

13. Aly, S. Deep Convolutional Self-Organizing Map Network for Robust Handwritten Digit Recognition / S. Aly, S. Almotairi // IEEE Access. – 2020. – vol. 8. – pp. 107035-107045. – URL: https://doi.org/10.1109/ACCESS.2020.3000829.– Текст: электронный.

14. Bonini Neto, A. Radial Base Neural Network For The Detection of Banana Maturation Stages: Perceptron Multilayer Network Comparison / A. Bonini Neto, V. Ferreira da Silva Fávaro, W. Prado Leão dos Santos et al. // Revista Brasileira De Engenharia De Biossistemas. – 2022. – vol. 16. – URL: https://doi.org/10.18011/bioeng.2022.v16.1175. – Текст: электронный

15. Galvan, D. The Spread of the COVID-19 Outbreak in Brazil: An Overview by Kohonen Self-Organizing Map Networks / D. Galvan, E. Luciane, C. Hágata, and A. Carlos Conte□Junior // Medicina. – 2021. – vol. 57. – URL: https://doi.org/10.3390/medicina57030235. – Текст: электронный

16. Haldorai, A. Canonical Correlation Analysis Based Hyper Basis Feedforward Neural Network Classification for Urban Sustainability / A. Haldorai, A. Ramu // Neural Process Letters. – 2021. – vol. 53. – pp. 2385-2401. – URL: https://doi.org/10.1007/s11063-020-10327-3. – Текст: электронный

17. Iris Data Set // archive.ics.uci.edu – репозиторий машинного обучения, содержащий наборы баз данных, генераторы данных. – URL: https://archive.ics.uci.edu/ml/datasets/Iris. – Текст: электронный

18. Masuyama, N. Adaptive Resonance Theory-Based Topological Clustering With a Divisive Hierarchical Structure Capable of Continual Learning / N. Masuyama, N. Amako, Y. Yamada et al. // IEEE Access. – 2022. – vol. 10. – pp. 68042-68056. – Текст: электронный. – URL: https://doi.org/10.1109/ACCESS.2022.3186479

19. Mrówczyńska, M. Measurement Data Processing with the Use of Art Networks / M. Mrówczyńska, J. Sztubecki // Civil and Environmental Engineering Reports. – 2018. – Vol. 28. – Issue 2. – P. 186-195– URL: https://doi.org/10.2478/ceer-2018-0029. – Текст: электронный.

20. Press, W.H. Numerical recipes: The Art of Scientific Computing, Third Edition / W.H. Press, B.P. Flannery, S.A. Teukolsky, W.T. Vetterling. – Cambridge University Press, 2007. – 1235 p. – Загл. с титул. экрана. – Текст: непосредственный.

21. Ranga Suri N. N. R. Outlier Detection: Techniques and Applications : A Data Mining Perspective / N. N. R. Ranga Suri, Narasimha Murty M., G. Athithan. – Springer, 2019. – 238 c. – Текст: непосредственный.

Глава 4 Разработка мобильного приложения для сканирования молочной продукции

В настоящее время увеличение использования населением высокопроизводительных мобильных устройств в качестве основного средства получения информации и запроса услуг через интернет приобретает устойчивую тенденцию. Переход от традиционных персональных компьютеров к мобильным устройствам (смартфонам и планшетам) в качестве средства доступа к услугам вынудил предприятия, различающихся по организационно-правовым формам и размерам, адаптировать мобильные каналы для своих существующих приложений. Параллельно с этим растет спрос на новые виды приложений, которые могут использовать уникальные характеристики мобильных устройств. Хотя разработка пользовательского интерфейса мобильного приложения для существующего бизнес-приложения, безусловно, имеет смысл, пользователи мобильных приложений ожидают большего от своего мобильного опыта. Это проявляется в постоянно растущем спросе на разработку мобильных приложений на рынке.

Разработка мобильных приложений стремительно развивается. От розничной торговли, телекоммуникаций и электронной коммерции до страхования, здравоохранения и правительства, организации в различных отраслях должны соответствовать ожиданиям пользователей в отношении удобных способов проведения транзакций и доступа к информации в режиме реального времени. Сегодня мобильные устройства и мобильные приложения, раскрывающие их ценность, являются наиболее популярным способом подключения людей и компаний к Интернету. Чтобы оставаться актуальными, быстро реагирующими и успешными, организациям необходимо разрабатывать мобильные приложения, которые требуются их клиентам, партнерам и сотрудникам.

4.1 Технология разработки мобильного приложения

История развития мобильных приложений. История мобильных приложений начинается с простейших первых

приложений – календарей, калькуляторов и игр, разработанных в среде Java.

3 апреля 1973 г. Мартин Купер из Motorola впервые позвонил по мобильному телефону доктору Джоэлу С. Энгелю из Bell Labs. В течение следующих двух десятилетий исследователи активно готовили мобильные приложения для этого устройства. Отдел исследований и разработок IBM Simon разработал первое мобильное приложение для смартфонов в 1993 г., ровно через два десятилетия после того, как был сделан первый звонок. Он имел такие функции, как адресная книга, календарь, мировое время и калькулятор.

В 2002 г. был выпущен смартфон (Blackberry), который стал одним из главных достижений в области разработки мобильных приложений, подчеркивающим значимость Blackberry Limited, также известной как Research in Motion Limited (RIM). Именно это привело к интеграции концепции, известной как беспроводная электронная почта.

В начале 90-х выпущено первое из узнаваемых приложений EPOC и разработанная Psion для портативных компьютеров. 16-битная система EPOC могла запускать дневники и базы данных, электронные таблицы и текстовые процессоры. Но будущие модели были способны работать с 32-разрядной операционной системой и были интегрированы с 2 МБ ОЗУ, что позволяло пользователям добавлять дополнительные приложения через свои пакеты программного обеспечения.

Разработанные Palm Inc. в 1996 г. Palm OS были в основном предназначены для персональных цифровых помощников и были известны как Garnet OS. Они имели графический пользовательский интерфейс с сенсорным экраном, а также ряд основных и других сторонних приложений, которые были запрограммированы на C/C++. Позже в качестве расширения для них были представлены браузеры протокола беспроводных приложений (WAP).

Язык разметки WAP Forum для беспроводных сетей был специально разработан для устройств, зависящих от XML, поддерживал протоколы беспроводных приложений и вследствие своей простоты мог работать на низкой пропускной способности мобильных сетей того времени, преодолевая существенные

требования к вычислительной мощности языка разметки гипертекста HTML.

Следующий этап развития был тесно связан с Java ME, J2ME или JME, а позже с персонализированным Java, который получил широкую популярность и существует в области программирования до сих пор.

Профиль мобильного информационного устройства, который поставляется с подмножеством конфигураций, включая подключенную ограниченную конфигурацию устройства, использовался для устройств, реализованных профилей. Кроме того, CLDC могла работать на устройствах с памятью от 160 КБ до 512 КБ и поставлялась с библиотеками класса Java, которые могут работать с виртуальными машинами.

Развитие операционных систем далее было связано с ОС Symbian, которая была разработана Symbian Ltd, совместным предприятием Ericsson, Motorola, Nokia и PSION и представляла собой усовершенствованную версию ОС PSION EPOC. До 2008 г. предприятие широко используемую операционную систему, способную одновременно работать на 250 миллионах устройств. Nokia продолжала работать над импровизацией ОС Symbian, и мы обнаружим, что эта платформа S60 была реализована на разных телефонах Nokia, включая Samsung и LG [3].

В настоящее время смартфоны и iPhone, которые мы ежедневно используем, эволюционировали настолько, что без приложений (социальные сети, банковское дело, здоровье и фитнес, игры, путешествия и отдых, покупки, новости и т.п.), количество которых исчисляется миллионами, человек не представляет своей жизни.

Основные подходы к проектированию мобильного приложения. Разработка мобильных приложений – это процесс создания программных приложений, которые работают на мобильном устройстве. Обычное мобильное приложение использует сетевое подключение для работы с удаленными вычислительными ресурсами. Следовательно, процесс мобильной разработки включает в себя создание устанавливаемых пакетов программного обеспечения (код, двоичные файлы, активы и т.д.), реализацию серверных служб, таких как доступ к данным с

помощью API, и тестирование приложения на целевых устройствах.

На современном рынке смартфонов есть две доминирующие платформы. Одной из них является платформа iOS от Apple Inc. Платформа iOS – это операционная система, на которой работает популярная линейка смартфонов Apple iPhone. Второй – Android от Google. Операционная система Android используется не только устройствами Google, но и многими другими производителями для создания собственных смартфонов и других интеллектуальных устройств [20].

Хотя между этими двумя платформами есть некоторое сходство при создании приложений, разработка для iOS и разработка для Android предполагают использование разных комплектов разработки программного обеспечения (SDK) и разных наборов инструментов разработки. В то время как Apple использует iOS исключительно для своих устройств, Google предоставляет Android другим компаниям при условии, что они отвечают определенным требованиям, например, включают определенные приложения Google на поставляемые ими устройства. Разработчики могут создавать приложения для сотен миллионов устройств, ориентируясь на обе эти платформы [6].

Существует четыре основных подхода к разработке мобильных приложений.

1) Нативные мобильные приложения – написаны на языке программирования и на платформах, предоставленных владельцем платформы, и работают непосредственно в операционной системе устройства, такой как iOS и Android.

2) Кроссплатформенные мобильные приложения – могут быть написаны на различных языках программирования и платформах, но они компилируются в нативное приложение, работающее непосредственно в операционной системе устройства.

3) Гибридные мобильные приложения – создаются с использованием стандартных веб-технологий, таких как JavaScript, CSS и HTML5, и поставляются в виде установочных пакетов приложений. В отличие от нативных приложений, гибридные приложения работают с «веб-контейнером», который обеспечивает среду выполнения браузера и мост для нативных API-интерфейсов устройств через Apache Cordova.

4) Прогрессивные веб-приложения (PWA) – предлагают альтернативный подход к традиционной разработке мобильных приложений, пропуская интеграцию в магазин приложений и установку. PWA – это веб-приложения, которые используют набор возможностей браузера, таких как работа в автономном режиме, запуск фонового процесса и добавление ссылки на главный экран устройства, чтобы обеспечить взаимодействие с пользователем, подобное приложению.

Каждый из этих подходов к разработке мобильных приложений имеет свои преимущества и недостатки. Выбирая правильный подход к разработке своих проектов, разработчики учитывают желаемый пользовательский опыт, вычислительные ресурсы и встроенные функции, необходимые для приложения, бюджет разработки, сроки и ресурсы, доступные для поддержки приложения.

Одна из проблем разработки нативных мобильных приложений заключается в том, что она требует узкоспециализированного набора навыков. Несмотря на то, что существуют большие и активные сообщества разработчиков для C и Java – языковых семейств, которые в основном используются для нативной разработки – меньше разработчиков, которые хорошо разбираются в версиях этих языков для конкретных платформ и соответствующих IDE. На самом деле, квалифицированные разработчики нативных приложений пользуются таким спросом, что многие компании с трудом нанимают и удерживают их в штате, и вместо этого им часто приходится прибегать к помощи сторонних дизайнеров и разработчиков для создания своих приложений для них [16].

Гибридные приложения позволяют разработчикам использовать веб-технологии HTML5/CSS/JavaScript, а затем инкапсулировать эти веб-приложения в контейнер, который позволяет веб-приложению действовать как собственное приложение на устройстве. Поскольку гибридные мобильные приложения – это просто веб-приложения, работающие во встроенной среде браузера, большую часть кода из веб-приложения можно использовать для создания мобильного приложения. Поскольку производительность рендеринга и времени выполнения мобильных браузеров постоянно растет, гибридная разработка

является жизнеспособной альтернативой для веб-разработчиков, которые хотят быстро создавать мобильные приложения.

Точно так же PWA пишутся с использованием традиционных технологий программирования веб-приложений, обычно включающих некоторые варианты JavaScript, HTML5 и CSS, и первоначально доступны через браузер на устройстве или компьютере.

Большинство кроссплатформенных фреймворков, таких как React Native и Native Script, предоставляют собственные компоненты для работы с кроссплатформенным кодом, в то время как некоторые другие, такие как Flutter и Xamarin, компилируют кроссплатформенный код в собственный код для повышения производительности [27].

Существует два взаимосвязанных основных компонента мобильного приложения:

1) «Внешняя часть» (Frontend), которое находится на мобильном устройстве,

2) «Внутренняя часть» (Backend) службы, которые поддерживают мобильную внешнюю часть.

В первые дни эры современных приложений для смартфонов мобильные приложения прошли ту же эволюцию, что и первые веб-сайты. Сначала приложения и сайты были полностью замкнуты в себе и действовали как не более чем статическая реклама бренда, компании, продукта или услуги.

Однако по мере улучшения возможностей подключения и сети приложения стали все больше подключаться к источникам данных и информации, которые находились за пределами самого приложения, и приложения становились все более динамичными, поскольку они могли обновлять свой пользовательский интерфейс и контент данными, полученными по сети, от запросов к источникам данных [5].

В результате мобильные интерфейсные приложения все больше полагаются на серверные службы и интегрируются с ними, которые предоставляют данные для потребления через мобильный интерфейс. Такие данные могут включать, например, информацию о продукте для приложений электронной коммерции или информацию о рейсах для приложений для путешествий и бронирования. Для мобильной игры данные могут включать новые

уровни или испытания, а также результаты или аватары других игроков.

Взаимодействие Frontend с Backend происходит следующим образом – мобильный интерфейс получает данные от серверной части с помощью различных сервисных вызовов, таких как API. В некоторых случаях эти API могут принадлежать и управляться одной и той же организацией, разрабатывающей мобильное приложение. В других случаях API может контролироваться третьей стороной, а доступ к мобильному приложению предоставляется на коммерческой основе. Например, разработчик может получать социальные сети или рекламный контент, звоня в службы СМИ или рекламной компании. В этом случае разработчику может потребоваться подписать контракт, чтобы получить учетные данные и ключ, который предоставляет доступ к API и определяет, как этот разработчик может его использовать, сколько это будет стоить или как часто он может вызываться, или сколько данных можно запросить за какой период времени [11].

Мобильный интерфейс - это визуальная и интерактивная часть приложения, с которой взаимодействует пользователь. Обычно оно находится на устройстве или, по крайней мере, есть значок, представляющий приложение, которое отображается на главном экране или закреплено в каталоге приложений устройства. Приложение можно загрузить из магазина приложений платформы, загрузить непосредственно на устройство или открыть через браузер устройства, как в случае с PWA.

Рабочий процесс фронтенд-разработки выглядит следующим образом – когда разработчик говорит, что он программист мобильных приложений, он чаще всего имеет в виду интерфейсную часть приложения и хорошо владеет языками и технологиями, которые используются для создания этого внешнего приложения.

В зависимости от размера команды, создающей приложение, в проектировании и разработке внешнего интерфейса мобильного приложения может участвовать много разных людей. Размер команды может варьироваться от одного разработчика, который делает все, что связано с созданием приложения, до десятков, сотен и более человек со специальными навыками.

Например, могут быть специальные креативные/графические дизайнеры, отвечающие за создание визуальных элементов

приложений, таких как значки, фон, цвета, темы и другие части приложения. Команда также может иметь пользовательский опыт и дизайн пользовательского интерфейса, которые работают над комплектацией компонентов, как они взаимодействуют друг с другом и пользователем.

Интеграция с серверной частью также имеет значительную долю работы над мобильным приложением. Независимо от размера команды критическим элементом усилий по разработке является построение логики приложения, которая отвечает за выполнение сетевых вызовов к серверным службам, извлечение данных и обновление данных в серверных системах с помощью новой сгенерированной информации. из приложения.

Доступ к этим серверным службам обычно осуществляется через различные интерфейсы прикладного программирования, наиболее известные как API. Существуют различные типы API, такие как REST и GraphQL, а также множество способов и стилей доступа к ним. Хотя некоторые внутренние API-интерфейсы служб доступны непосредственно приложению через вызовы на самой платформе, многие специализированные службы необходимо интегрировать в приложение с помощью комплекта разработки программного обеспечения, широко известного как SDK. После добавления SDK в приложение через среду разработки приложение может использовать API, определенные в SDK [13].

Примером серверной службы для мобильного интерфейса может быть база данных, содержащая информацию, используемую в приложении. Для прямого доступа к базе данных разработчик мобильных устройств должен знать сетевое расположение базы данных, протокол доступа к базе данных, учетные данные для аутентификации и авторизации доступа к данным, а также конкретные команды базы данных, необходимые для извлечения необходимых данных.

В качестве альтернативы разработчик может использовать специализированный API при взаимодействии с базой данных; разработчику может потребоваться знать только параметры, необходимые в вызове метода, чтобы получить или обновить необходимую информацию. В некоторых случаях разработчик мобильных приложений может разработать эти API самостоятельно

или использовать определение API, предоставленное ему владельцем/оператором внутреннего ресурса.

Как правило, REST API используется для взаимодействия с источниками данных в облаке, такими как облачная база данных. API GraphQL также является еще одним вариантом для разработчиков, поскольку он упрощает работу с серверными данными в мобильном приложении. GraphQL обеспечивает поддержку запросов через единую конечную точку API и схему данных, которую можно использовать для создания и простого расширения моделей данных, используемых в приложении [4].

Серверная часть мобильного приложения работает независимо от того, какая интерфейсная платформа или методология разработки используются, для предоставления высококачественных мобильных приложений, которые радуют и удерживают пользователей, требуются надежные серверные службы.

Учитывая критическую важность серверного обслуживания для успеха мобильного приложения, разработчики должны принять несколько важных архитектурных решений. Эти решения включают в себя то, какие службы разработчики должны создавать сами или какие сторонние службы использовать, а затем надо ли запускать и поддерживать свои собственные службы или использовать преимущества стороннего обслуживания.

Исходя из всего перечисленного можно сделать вывод, чтобы повысить производительность и эффективность разработчиков, необходимо создавать свои собственные сервисы только в том случае, если они тесно связаны с предметной областью приложения и воплощать уникальную интеллектуальную собственность. Кроме того, даже для сервисов, которые разработчики создают сами, почти всегда следует использовать облачные сервисы для создания и обслуживания своей серверной инфраструктуры.

Сканирование Data matrix. 2D-коды, такие как матрица данных и QR-коды, используются почти во всех отраслях промышленности для обмена информацией о деталях или продуктах, на которые они нанесены. Например, коды сохраняют URL-адрес веб-сайта. В случае продуктов такие коды сканируются на каждом этапе производства и процесса доставки, чтобы отслеживать их и сохранять ценную информацию в базе данных [21].

Матричный код данных (Data matrix) – это двумерный код, состоящий из черных и белых ячеек, которые обычно располагаются в виде квадратного узора (хотя существуют и прямоугольные узоры). Количество строк и столбцов увеличивается с увеличением объема информации, хранящейся в коде, который ограничен 2335 буквенно-цифровыми символами. L-образная форма, следующая за его границами, – это шаблон поиска, который используется сканерами для распознавания и считывания кода. Использование матричных кодов данных стандартизировано благодаря международному стандарту ISO/IEC 16022.

QR-код – или код быстрого ответа – представляет собой двухмерный код, состоящий из черных ячеек, расположенных в виде квадратной сетки на белом фоне. Он может хранить до 4296 буквенно-цифровых символов. Максимальное количество символов в основном определяется количеством строк и столбцов. Его искатель состоит из трех квадратных структур по углам, что облегчает его распознавание. QR-коды одинаково используются во всем мире благодаря международному стандарту ISO/IEC 18004.

Data Matrix и QR-коды являются двухмерными штрих-кодами, которые могут нести всю ту же информацию, что и традиционные линейные или «одномерные» штрих-коды. GS1 – глобальная организация по стандартизации – разрабатывает и поддерживает глобальные стандарты для различных типов штрих-кодов, включая коды Data Matrix и QR-коды. Это означает, что коды Data Matrix и QR-коды могут содержать все идентификационные ключи GS1, включая GTIN (глобальный номер предмета торговли) – уникальный, международно-признанный номер, используемый для идентификации продуктов.

Оба кода могут использоваться для хранения информации, включая даты истечения срока годности товара, серийные номера и номер партии, а также могут содержать URL-адреса для «расширения» упаковки продукта, направляя потребителей на внешний веб-сайт с полезной информацией, такой как аллергены и соответствие образу жизни, или рецепт и инструкции по применению.

Кроме того, оба кода имеют возможности исправления ошибок – это означает, что, в отличие от стандартного одномерного штрих-кода, данные все еще можно считать, даже если код

частично удален или поврежден. Возможности исправления ошибок для обоих кодов достигаются за счет использования алгоритма исправления ошибок Рида-Соломона – математического метода исправления ошибок, который добавляет резервные данные в код.

Однако между двумя кодами есть небольшие различия, которые делают их более подходящими для разных приложений [22].

Коды Data Matrix физически меньше QR-кодов, обеспечивая высокую плотность данных при очень небольшом размере, что делает их идеальным решением для маркировки отдельных частей продукта в условиях ограниченного пространства. Коды Data Matrix также являются единственными 2D-кодами, одобренными GS1 для регулируемых предметов медицинского назначения; они также являются типом кода по умолчанию для автомобильных и аэрокосмических приложений.

Для сравнения, QR-коды больше и могут содержать больше данных, чем коды Data Matrix. Кроме того, хотя коды Data Matrix способны кодировать информацию только в числовых и буквенно-цифровых символах, поскольку QR-коды были изобретены в Японии, они также могут включать кандзи и другие многобайтовые наборы символов, что делает их подходящими для использования с неевропейскими кодами.

И QR-коды, и коды Data Matrix являются общедоступными кодами; это означает, что они не требуют авторских отчислений, поэтому вам не нужно платить за лицензию на их использование. Опубликованные спецификации и требования к печати для каждого кода можно получить в Международной организации по стандартизации (ISO).

Несмотря на их различия, оба кода могут использоваться очень похожими способами – оба предоставляют возможности для кодирования большего количества данных, чем традиционный штрих-код 1D, и могут использоваться для предоставления дополнительной информации для внутренней прослеживаемости и в целях потребительского маркетинга.

На практике, однако, коды Data Matrix чаще всего используются для внутренней идентификации продукта и приложений по борьбе с контрафактной продукцией, в то время как

QR-коды стали стандартным форматом для большинства приложений, ориентированных на потребителя.

Код матрицы данных используют, для:

– отслеживания цепочки поставок, включая прямую маркировку деталей, например, электронных компонентов;

– защиты от подделок посредством сериализации, например, в фармацевтической упаковке.

QR код предоставляет:

– дополнительную информации о продукте;

– инструкцию по применению и рецепты;

– социальный обмен;

– автоматическую привязку для заказа запчастей и оформления гарантии;

– информацию об акциях, конкурсах и геймификации.

Для разработки мобильного приложения с применением сканирования кода будем отталкиваться от Data Matrix и разрабатывать технологию способную его считывать, так как на продуктах молочной продукции, приложение для которой мы будем реализовывать, присутствует такой код [9].

4.2 Исследование стоимости разработки мобильного приложения в России

Оценка стоимости разработки мобильного приложения. Учитывая масштаб распространения смартфонов необходимость разработки и использования мобильного приложения не вызывает вопросов. Наличие и продвижение приложения ведет к повышению продаж, расширению клиентской базы, поддержанию постоянной связи с клиентами, сбору информации об аудитории, позволяющую корректировать стратегию развития предприятия.

Разработка приложения - сложная задача, требующая значительных затрат времени, бюджета и усилий. А когда сам бизнес основан на мобильном приложении, стоимость и время разработки становятся решающим фактором успеха. Более того, существует огромная разница в ценах на разработку мобильных приложений в зависимости от географии, платформ, сложности приложения и этапа разработки.

Индустрия разработки мобильных приложений - одна из самых быстрорастущих в мире. Инновации и технологии, такие как

искусственный интеллект, блокчейн и т.д., стимулируют рост этого сегмента. Компании используют эти технологии, разрабатывая мобильные приложения либо через своих разработчиков, либо через аутсорсинг. Основная проблема при разработке мобильных приложений – это стоимость.

Простые вычисления с затраченным временем и средней почасовой оплатой (1500 рублей в час), полученной в результате сбора данных, означают, что средние затраты на разработку приложения будут следующими:

– простое приложение стоит до 500 тыс. рублей с минимальными жизнеспособными функциями;

– сложное приложение стоит от 500 тыс. до 1млн рублей с популярными потребительскими функциями;

– расширенное приложение с передовыми функциями стоит более 1млн. рублей.

Конечно, это оценочные расчеты. И в процессе разработки есть много движущихся частей. Каждая из этих переменных обсуждается и учитывается в следующей разбивке затрат на разработку приложения.

Время, затраченное на каждый из этапов процесса разработки, разное. Простые расчеты на основе исследования компании Infoshell показывают, что среднее время разработки приложения характеризуется следующими цифрами:

– создание простого приложения занимает от 200 до 400 рабочих часов с минимальными жизнеспособными функциями;

– создание сложного приложения с популярными потребительскими функциями занимает от 400 до 800 рабочих часов;

– создание расширенного приложения с передовыми функциями занимает от 800 рабочих часов.

Для расчета рассматриваются стандартные практики - одновременная работа над двумя этапами разработки и рабочее время по 6-8 часов в день, 5 дней в неделю.

Платформа во многом влияет на стоимость разработки приложений. Стоимость разработки приложения для Android, iOS или гибрида рассчитывается примерно одинаково. Если есть цель создать приложение для какой-то одной платформы, разницы в стоимости разработки нет. Но если появляется необходимость

создать приложение для нескольких платформ, придется платить за каждую отдельную платформу [26].

Предприятиям необходимо учитывать такие факторы, как доля рынка Android и iOS, целевая аудитория, разнообразие устройств, требования ОС, сервисы приложений и, самое главное, техническая реализация. Для разработки приложений для этих платформ требуются разные языки программирования, используются разные инструменты разработки и разные API.

Чтобы улучшить взаимодействие с пользователем, должны быть разработаны собственные приложения для конкретных руководств и аппаратного обеспечения устройства. Таким образом, разработка таких приложений требует интенсивного программирования и большого количества времени.

Приблизительная оценка стоимости разработки приложения для Android или iOS (2500 рублей в час) по отдельности дает:

- простое приложение - от 500 тыс. до 1 млн. рублей;
- более сложное приложение - от 1 млн. до 2 млн. рублей;
- расширенное приложение - от 2 млн. рублей.

Нативные приложения могут дорого обойтись, если вы разрабатываете их как для Android, так и для iOS. Если цель приложения – охватить большую аудиторию, хорошим выбором будет кроссплатформенная или гибридная разработка приложений. Одно гибридное приложение предназначено для работы на различных устройствах, поэтому для разработки требуется только одно приложение.

Стоимость разработки приложений имеет большую разницу по географическому показателю. Наиболее популярными странами для разработки мобильных приложений являются США, Канада, Южная Америка, Великобритания, Украина, Индия, Юго-Восточная Азия и Австралия. И почасовые ставки, взимаемые разработчиками приложений, значительно различаются в этих странах [29]. Таким образом, местонахождение разработчиков также влияет на общую стоимость разработки мобильного приложения. Необходимо разумно рассмотреть местоположение в соответствии с доступным бюджетом для создания приложения, и наоборот.

Как правило, цены не обязательно гарантируют качество разработки вашего приложения. Иногда дешевые затраты на

разработку могут сэкономить вам деньги, в то время как дорогие затраты связаны с именем компании.

Для большинства компаний планирование разработки приложений является неотъемлемой частью процесса. Цель состоит в том, чтобы четко понять каждый аспект мобильного приложения, которое компании собираются разработать. В некотором смысле планирование всего проекта позволяет избежать дорогостоящих изменений содержания проекта.

Этап планирования включает:

- исследование рынка и бизнес-анализ;
- формулировка требований;
- определение масштаба проекта;
- продумывание функций и целей приложения;
- выбор стека технологий;
- планирование времени и расходов.

Проектирование мобильного приложения необходимо как для представления его внешнего вида, так и его внутреннего наполнения. Большинство компаний-разработчиков прилагают усилия для исследования и проектирования пользовательского опыта.

Стоимость разработки приложения включает:

- исследование и анализ пользователей;
- информационную архитектуру;
- навигацию по приложениям;
- каркас.

По данным компаний, на разработку пользовательского интерфейса приложения уходит в среднем от 90 до 140 часов (или до 270 часов). И, таким образом, он может стоить от 200 тыс. до 350 тыс. рублей, при этом расходы доходят до 600 тыс. рублей за дизайн, оптимизированный для пользователей.

Затраты на разработку пользовательского интерфейса приложения включают:

- фирменные стили – логотипы, иконки, цвета;
- интерфейс приложения – визуальные эффекты, контент;
- визуализированные проекты;
- окончательный дизайн/прототип.

Для разработки пользовательского интерфейса приложения в среднем требуется от 40 до 100 часов или до 400 часов для превосходных визуальных эффектов. При средней стоимости разработки затраты на проектирование составляют от 100 тыс. до 250 тыс. рублей и могут доходить до 1 млн. рублей.

Проектирование UX – это непрерывный процесс, который во многом зависит от брендинга, бизнес-целей и потребностей, сложности проекта, количества экранов и функциональности.

Стоимость разработки пользовательских функций приложения, как было отмечено выше, зависит от его сложности, что определяет большую часть времени и затрат на разработку приложения. Пользовательские функции, которые должны быть разработаны, во многом усложняют приложение. И поэтому нельзя дать однозначного ответа, пока не будет принято решение о наборе функций [2].

Таким образом, необходимо получить общее представление разработчиков приложений, чтобы указать время, необходимое для разработки трех наборов пользовательских функций – простых, сложных и расширенных. Каждый набор включает в себя некоторые из самых популярных пользовательских функций, которые обычно есть в большинстве приложений.

Приблизительное время и стоимость разработки приложения для каждого набора функций:

Простые пользовательские функции - от 70 до 105 часов от 175 до 262 тыс. рублей.

- Войти (электронная почта/социальные сети);
- Профиль пользователя (создание и редактирование);
- Чат/Сообщения;
- Загрузка файла;
- Приборная доска;
- Поиск;
- Собственные функции устройства.

Сложные пользовательские функции от 100 до 500 часов от 250 тыс. до 1 млн. 250 тыс. рублей.

- Вовлечение пользователей (социальные сети, электронная почта, SMS и т.д.);
- Лента активности;
- Камера/Галерея;

– Электронная коммерция — каталоги, заказы, тележки и т.д.;
– Геолокация;
– Связь в один клик;
– Интеграция платежей;
– Интеграция со сканером QR/штрих-кода;
– Потоковое аудио/видео;
– Синхронизация на всех устройствах;
– Настройка конфиденциальности пользователя.

Представленные цифры призваны дать общее представление о средней стоимости разработки приложения. На самом деле одно приложение будет включать в себя несколько функций и не обязательно из одного и того же набора или даже сложности, как упоминалось выше.

Проще говоря, даже одна функция может стать сложной в зависимости от требований приложения и увеличить стоимость разработки. Каждая из этих функций требует индивидуальной разработки. Таким образом, общая стоимость приложения на самом деле зависит от конкретных функций, которые должны быть включены, и от их характера.

Стоимость разработки инфраструктуры приложений подразумевает технологию, гибкость, масштабирование, сервисы и все другие технические решения, которые требуются мобильному приложению, обеспечиваются исключительно его инфраструктурой. Таким образом, разработка инфраструктуры приложения имеет одну из самых высоких из статей затрат, уступая только затратам на расширенные пользовательские и административные функции.

Ниже приведены минимально необходимые элементы инфраструктуры приложения, на которых основаны затраты на разработку:

– Настройка базы данных;
– Решения для хранения данных;
– Сторонние API-интеграции;
– Шифрование данных и безопасность приложений;
– Масштабируемость приложения.

Поддержка и техническое обслуживание, о которых часто забывают, требуют значительных инвестиций, чтобы поддерживать динамику бизнеса приложений. И большинство девелоперских фирм включают стоимость обслуживания (на некоторый период) в контракт [30].

Некоторые из мероприятий по поддержке и обслуживанию, которые включают компании, следующие:

- Плата за инфраструктуру и сторонние API;
- Постоянное исправление ошибок и обновления приложений;
- Оптимизация кода;
- Стабильность и производительность приложения;
- Аварийное обслуживание;
- Масштабирование приложения;
- Разработка новых функций;
- Обновление до последних версий ОС.

Часто это непрерывный процесс технического обслуживания длится гораздо дольше, чем время разработки приложения. Обычно компании берут в среднем 25% стоимости разработки в год. Это означает то, что, если разработка сложного приложения стоит 962 500 рублей, можно предположить, что обслуживание приложения будет стоить около 240 625 рублей в год.

Экономическая эффективность. Важнейшим этапом плана развития деятельности компании является обоснование экономической эффективности проекта. Актуальность, предлагаемой разработки требует полного обоснования. Эффективность системы заключается в том, что ее свойства выполняют поставленную цель в определенных условиях использования и качества. Показатели эффективности характеризуют степень приспособленности системы к выполнению поставленных перед нею задач и являются обобщающими показателями разработки мобильного приложения.

После разработки проекта мобильного приложения сканера молочной продукции для потребителя «Торру» необходимо дать оценку экономической эффективности. В данной работе эффективность следует рассматривать как сопоставление результатов использования информационной системы с затратами на ее ведение и эксплуатацию. Далее приведены качественные и

количественные методы оценки эффективности при внедрении программно-аналитического обеспечения. Для проведения оценки экономической эффективности необходимо провести расчет затрат на реализацию проекта, а также оценить ожидаемый эффект [18].

Качественные показатели эффективности будут характеризоваться следующим параметрами:

- удобное дополнение в виде простого мобильного приложения. Потенциальный покупатель молочной продукции, который пользуется мобильным устройством, может отсканировать Data matrix-код для получения информации о продукте;
- приложение содержит информацию о товарах, интересующих потенциального потребителя;
- имеется простая авторизация пользователей;
- для администратора есть возможность вывода статистики;
- существование у компании собственного проекта по разработке мобильного приложения делает ее более современной и популярной;
- с помощью приложения привлекается число новых клиентов.

Значимыми обобщающими показателями являются показатели экономического эффективности системы, которые характеризуют целесообразность произведенных на создание и функционирование системы затрат.

В представленном случае, показатели экономической эффективности проекта мобильного приложения характеризуют целесообразность произведенных на его создание и функционирования затрат. Эти показатели должны сопоставлять затраты и результаты: затраты на разработку и создание программного обеспечения, а также текущие затраты на ее эксплуатацию, говоря о результат, имеется ввиду – прибыль, получаемая в результате использования системы.

Экономическая эффективность характеризует отношение результатов, представленными величиной прибыли к величине суммарных затрат на создание и эксплуатацию системы. Тем не менее, в качестве показателя экономической целесообразности создания системы очень часто выступает показатель экономического эффекта, количественно равный прибыли за вычетом нормы прибыли с произведенных капитальных затрат.

Таким образом, в качестве показателей экономической эффективности обычно используются:

1) годовой экономический эффект:

$$Э = Эг - Ен \times К, \quad (4.1)$$

где *Эг* – годовая (прибыль), получаемая при использовании ИС, руб.,

Ен – нормативный коэффициент эффективности капитальных вложений;

К – единовременные (капитальные) затраты (вложения) на создание ИС.

2) коэффициент экономической эффективности капитальных вложений:

$$Е = Эг \div К, \quad (4.2)$$

где *Эг* – годовая (прибыль), получаемая при использовании ИС, руб.;

К – единовременные (капитальные) затраты (вложения) на создание ИС.

3) срок окупаемости капитальных вложений (в годах):

$$Т = К \div Эг, \quad (4.3)$$

где *К* – единовременные (капитальные) затраты (вложения) на создание ИС;

Эг – годовая (прибыль), получаемая при использовании ИС, руб.

Коэффициент *Ен* должен характеризовать средний уровень эффективности капитальных вложений в хозяйство страны, и при рыночной экономике он должен быть не меньше ставки банковского кредита. Если использовать названные показатели в качестве критерия для принятия решения о целесообразности создания ИС, то они должны быть следующими:

$$Э > 0, Е > Ен, Т > 1 \div Ен, \quad (4.4)$$

Следующим шагом проведем анализ затрат на разработку и эксплуатацию проекта мобильного приложения «Торру» для сканирования молочной продукции.

Основными статьями расходов выступают:

- Образование юридического лица;
- ЗП сотрудников с отчислениями;

– Аренда помещения;
– Покупка основных средств;
– Расходные материалы (оборотные средства);
– Затраты на рекламу и командировки;
– Прочие затраты.

Затраты на разработку подразумевают следующие пункты:

1. Анализ требований, проектирование, аналитика, спецификация – необходимо для того, чтобы охарактеризовать конечную цель проекта.

2. Кодирование – может происходить параллельно со следующим этапом разработки – тестированием программного обеспечения для приведения приложения в представляемый вид при проектировании.

3. Тестирование и отладка, запуск приложения, внедрение или вывод созданного ПО на проектную мощность [15].

4.3 Разработка мобильного приложения «Торру»

Выбор языка программирования. Для разработки мобильного приложения, необходимо выбрать язык программирования. Сравнение и анализ основных из них позволит выбрать оптимальный для разработки приложения и внедрения в него необходимых функций сканирования «Data Matrix» кода и других.

Для на начала рассмотрим базовые понятия разработки веб-сервисов. Frontend – это разработка пользовательского интерфейса и функций, которые работают на клиентской стороне веб-сайта или приложения. Это всё, что видит пользователь, открывая веб-страницу, и с чем он взаимодействует [17].

Backend разработка – это набор аппаратно-программных средств, при помощи которых реализована логика работы приложения. Попросту говоря, это то, что скрыт от глаз Backend пользователя и происходит вне его браузера и компьютера.

Многие языки программирования, которые используются для Backend-разработки, пользуются большим спросом на рынке:

– Python – один из самых популярных вариантов Backend - программирования. Он относительно новый и имеет огромную библиотечную поддержку;

– PHP – существует на рынке давно и широко используется даже сегодня. Например, изначально Facebook имеет свой существенный Backend, разработанный на PHP;

– JavaScript – благодаря мощным веб-фреймворкам, таким как NodeJS, JavaScript завоевал огромную долю рынка и стал одним из самых популярных языков внутреннего программирования;

Прежде чем проводить сравнение, необходимо сначала перечислить основные положения для сравнения, которые существенно повлияют на выбор веб-фреймворка:

– простота обучения – это один из самых важных параметров, чтобы решить, какой веб-фреймворк следует использовать. Если язык программирования трудно выучить, нет смысла тратить на него время. Сегодня время разработки для всех практических целей важнее времени изучения;

– поддержка для пользователя – возможность обратиться за помощью при написании кода к более продвинутым специалистам онлайн на форумах. Если какой-то конкретный язык программирования недостаточно известен, и им владеют мало людей, которые могут помочь с решением различных проблем, то лучше подумать выборе другого языка программирования;

– материалы для изучения так же, как и поддержка пользователя, важна, чтобы язык или любой другой способ программирования имел достаточно материалов для изучения, доступных для разработчиков, чтобы разобраться и понять нюансы;

– стоимость является также важным критерием для выбора, поскольку большинство программ не являются бесплатными. Это может существенно повлиять на выбор для использования в организациях с ограниченным бюджетом. Поэтому для достижения определенных целей в разработке, далеко не в последнюю очередь, необходимо задумываться о средствах, которые потребуется вложить в реализацию данного направления. В целом, в наше время, это нужно не только тем, у кого заработок основан на данной сфере, но и в любой другой фирме для дальнейшего развития;

– поддержка библиотек необходима для облегчения разработки программного обеспечения. Если язык программирования широко используется, будет больше разработчиков, которые будут разрабатывать библиотеки для

конкретного языка. В результате изучение и использование становится еще проще;

– скорость, которая позволяет поддерживать высокую пропускную способность и низкую задержку, которые могут потребовать серверные приложения;

– разнообразие веб-фреймворков является преимуществом, чтобы язык программирования обеспечивал хорошо спроектированные фреймворки веб-разработки, которые просты в использовании и создают удобные приложения;

– настройка определяет выбор языка программирования, который должен также зависеть от доступных инструментов отладки, доступных для языка. Отсутствие хороших инструментов для настройки означает, что разработчики будут тратить больше времени на настройку, что, по сути, не является самым продуктивным использованием времени [7].

Python – выпущен в 1991 г. и разработан Гвидо ван Россумом. Python является языком функционального программирования высокого уровня. Python находит свое применение в веб-разработке, разработке программного обеспечения и т.д. Язык предлагает множество функций, таких как работа на разных платформах, работа в системе интерпретатора, его легко кодировать, так как его синтаксис похож на английский язык (Таблица 4.1).

PHP – широко используемый язык сценариев с открытым исходным кодом. PHP способен генерировать динамический контент страницы, может создавать, открывать, читать, писать, удалять и закрывать файлы на сервере. Он может собирать данные формы, а также может отправлять и получать файлы cookie. PHP также дружественный к базе данных и может добавлять, удалять, изменять данные в базе данных. Он также может быть использован для контроля доступа пользователей и для шифрования данных.

Java – строго типизированный объектно-ориентированный язык программирования. Достоинством подобного способа выполнения программ является полная независимость байт-кода от операционной системы и оборудования, что позволяет выполнять Java-приложения на любом устройстве, для которого существует соответствующая виртуальная машина.

Подводя итог, можно сказать, что для разработки мобильного приложения больше подойдет Python, так как данный язык более направлен на Backend с высокой производительностью.

Среда разработки. Среда разработки – это набор процедур и инструментов для разработки, тестирования и отладки приложения или программы. Другими словами, среда разработки программного обеспечения и веб-разработки – это рабочая среда для разработчиков, в которой можно вносить изменения, не нарушая ничего в реальной среде. Интегрированная среда разработки часто используются как инструмент программирования, чтобы помочь разработчику [14].

Таблица 4.1 – Сравнение Python, PHP и Java

Аспекты	Python	PHP	Java
Операционная система	Любая	Windows, Linux, Mac OS X	Любая
Материалы для изучения	Материалы предоставлены на его официальных сайтах и других форумах	PHP также соответствует Python и большим объемом информации	Имеет крупную базу для изучения
Поддержка для пользователя	Обширная	Обширная	Обширная
Поддержка библиотек	Python обладает исключительно хорошо развитой библиотечной поддержкой практически для всех типов приложений	PHP отстает, но Packagist (репозиторий пакетов PHP) является сильной основой, поддерживающей PHP	Имеет относительно небольшое количество библиотек
Подключение к базе данных	Интеграция базы данных не так сильна, как PHP.	Он обеспечивает доступ к более чем 20 различным базам данных	Разнообразие баз данных не велико
Производительность	Python имеет высокую производительность	PHP имеет в несколько раз выше производительность, чем Python	Средняя
Разнообразие веб-фреймворков	Django, Flask, Pylons, Pyramid	Codeigniter, Zend, Laravel, Symfony	Spring Framewor, PrimeFaces
Настройка	Python предоставляет мощный отладчик под названием PDB (Python Debugger)	PHP предоставляет пакет XDebug для отладки	Имеет большое количество настроек

NetBeans IDE – свободная интегрированная среда разработки приложений (IDE) на языках программирования Java, Python, PHP, JavaScript, C, C++, Ада [3] и ряда других. Последние версии NetBeans IDE поддерживают рефакторинг, профилирование, выделение синтаксических конструкций цветом, автодополнение набираемых конструкций на лету и множество предопределённых шаблонов кода.

Visual Studio Code – редактор исходного кода, разработанный Microsoft для Windows, Linux и macOS. Позиционируется как «лёгкий» редактор кода для кроссплатформенной разработки веб и облачных приложений.

Atom – бесплатный текстовый редактор с открытым исходным кодом для Linux, macOS, Windows с поддержкой плагинов, написанных на Node.js (Таблица 4.2).

Таблица 4.2 – Сравнение Atom и VS Code

Аспекты	Atom	VS Code	NetBeans
Поддержка языков программирования	HTML, CSS, JS, PHP	HTML, CSS, JS, PHP	Java,Python, PHP, JavaScript, C, C++, Ада
Комфортный интерфейс	Да	Да	Да
Производительность	Заметно ниже, чем VS Code	Грамотная оптимизация расширений значительно повышает производительность	Высокая
Экосистема	Большое количество расширений с простой установкой	Большое количество расширений с простой установкой	Большое количество расширений с простой установкой
Настройка	Настройки самого редактора полностью отделены от расширений	Большое количество расширений и кастомизаций предполагают большое количество конфигурационных настроек	Большой выбор настроек

Так как мы уже выбрали язык программирования Python для разработки приложения, необходима среда разработки, где можно использовать Python [8].

Подводя итог всему выше сказанному, проанализировав данные среды программирования, остановлюсь на VS Code, по причине простой настройки и высокой оптимизации, что повышает производительность работы приложения.

Проектирование мобильного приложения. Unified Modeling Language (UML) представляет собой стандартный визуальный язык моделирования предназначен для использования:

- моделирования бизнеса и аналогичных процессов,
- анализа, проектирование и внедрение программных систем.

UML – это общий язык для бизнес-аналитиков, архитекторов и разработчиков программного обеспечения, используемый для описания, определения, проектирования и документирования существующих или новых бизнес-процессов, структуры и поведения артефактов программных систем.

В UML используются следующие виды диаграмм:

- Диаграмма деятельности,
- Диаграмма вариантов использования,
- Диаграмма классов,
- Диаграмма состояний,
- Диаграмма компонентов,
- Диаграмма последовательности,
- Диаграмма объектов,
- Диаграмма развертывания

Диаграмма вариантов использования показывает, как пользователи, отображаемые в виде фигурок, называемых «актерами», взаимодействуют с системой. Этот тип UML-диаграммы должен представлять собой общий обзор взаимоотношений между участниками и системами (Рисунок 4.1).

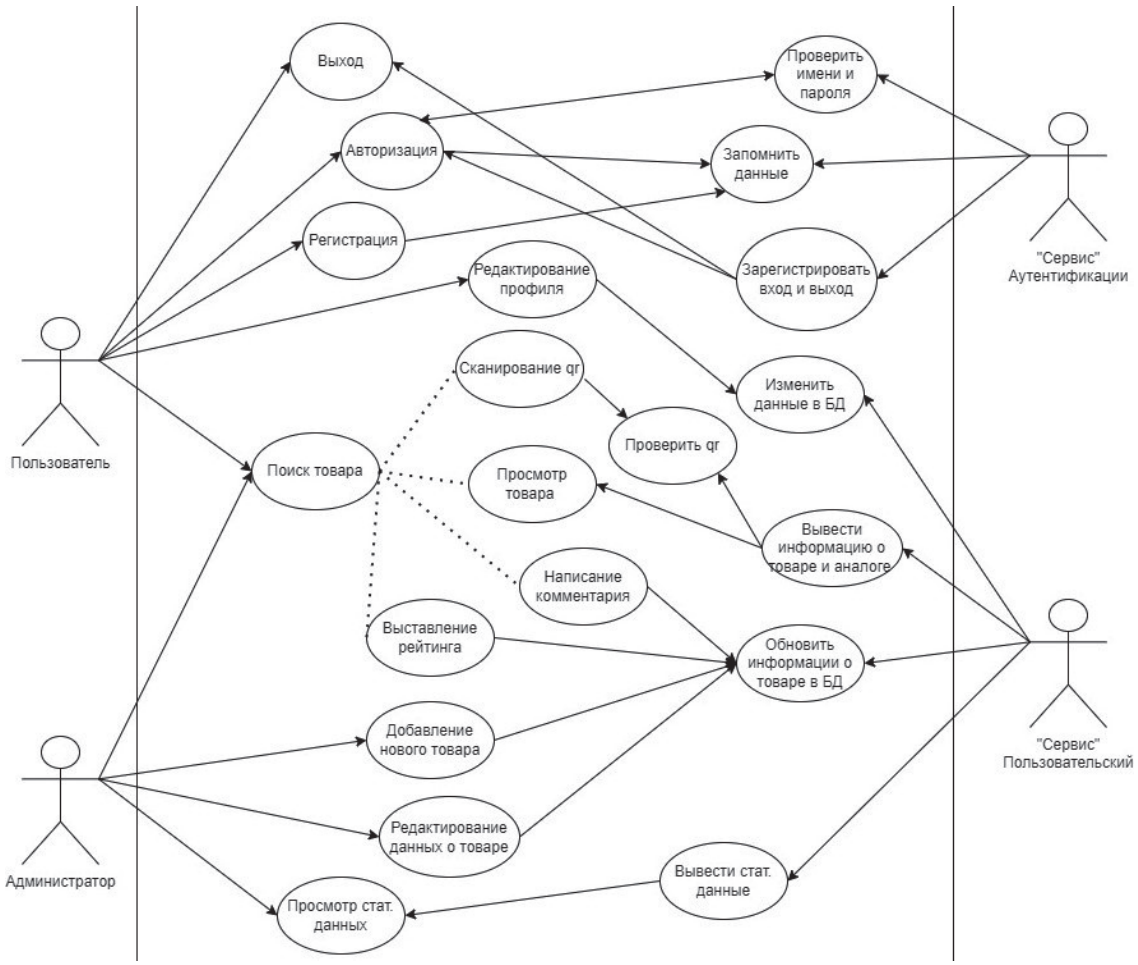


Рисунок 4.1 – Диаграмма вариантов использования

Поскольку большая часть программного обеспечения основана на объектно-ориентированном программировании, где разработчики определяют типы функций, которые можно использовать, диаграммы классов являются наиболее часто используемым типом диаграмм UML. Диаграммы классов показывают статическую структуру системы, включая классы, их атрибуты и поведение, а также отношения между каждым классом.

Класс представлен прямоугольником, который содержит три отсека, сложенных вертикально - верхний отсек содержит имя класса и является обязательным, но нижние два отсека содержат подробную информацию об атрибутах класса и операциях класса или его поведении (Рисунок 4.2).

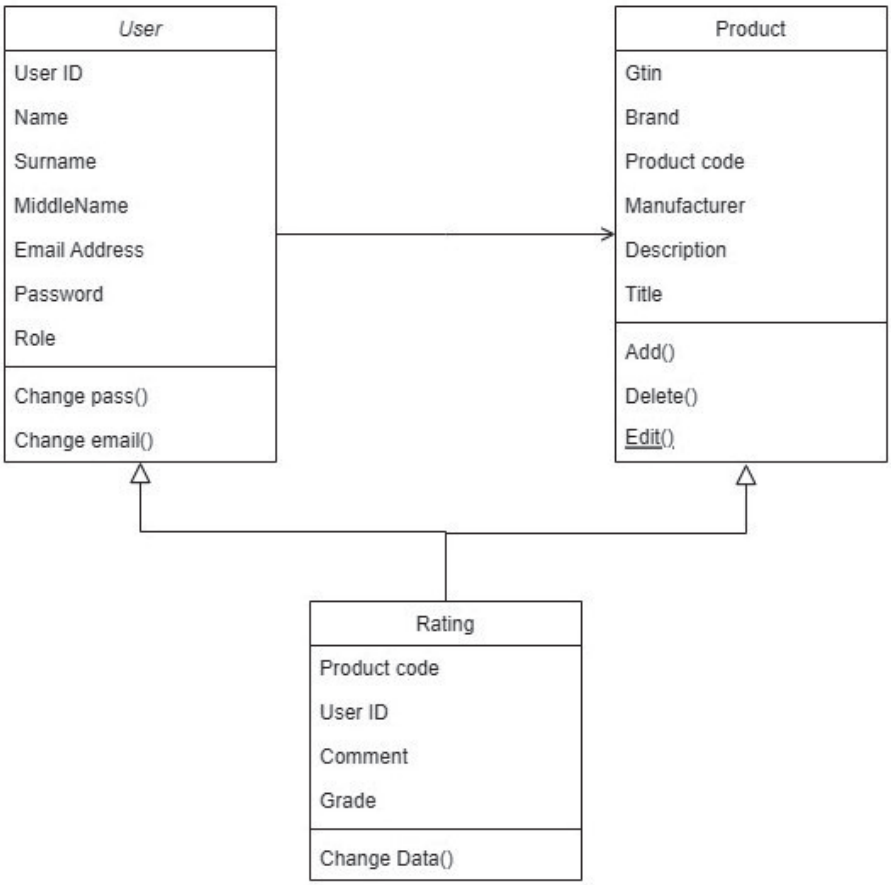


Рисунок 4.2 – Диаграмма классов

Диаграммы деятельности визуализируют шаги, выполняемые в сценарии использования – действия могут быть последовательными, разветвленными или параллельными (Рисунок 4.3). Этот тип UML-диаграммы используется для демонстрации динамического поведения системы, но он также может быть полезен при моделировании бизнес-процессов. Также диаграммы являются графическим представлением рабочих процессов поэтапных действий и действий с поддержкой выбора, итерации и параллелизма. В Unified Modeling Language диаграммы деятельности предназначены для моделирования как вычислительных, так и организационных процессов (т. е. рабочих процессов), а также потоков данных, пересекающихся с соответствующими действиями. Хотя диаграммы действий в основном показывают общий поток управления, они также могут включать элементы, показывающие поток данных между действиями через одно или несколько хранилищ данных.

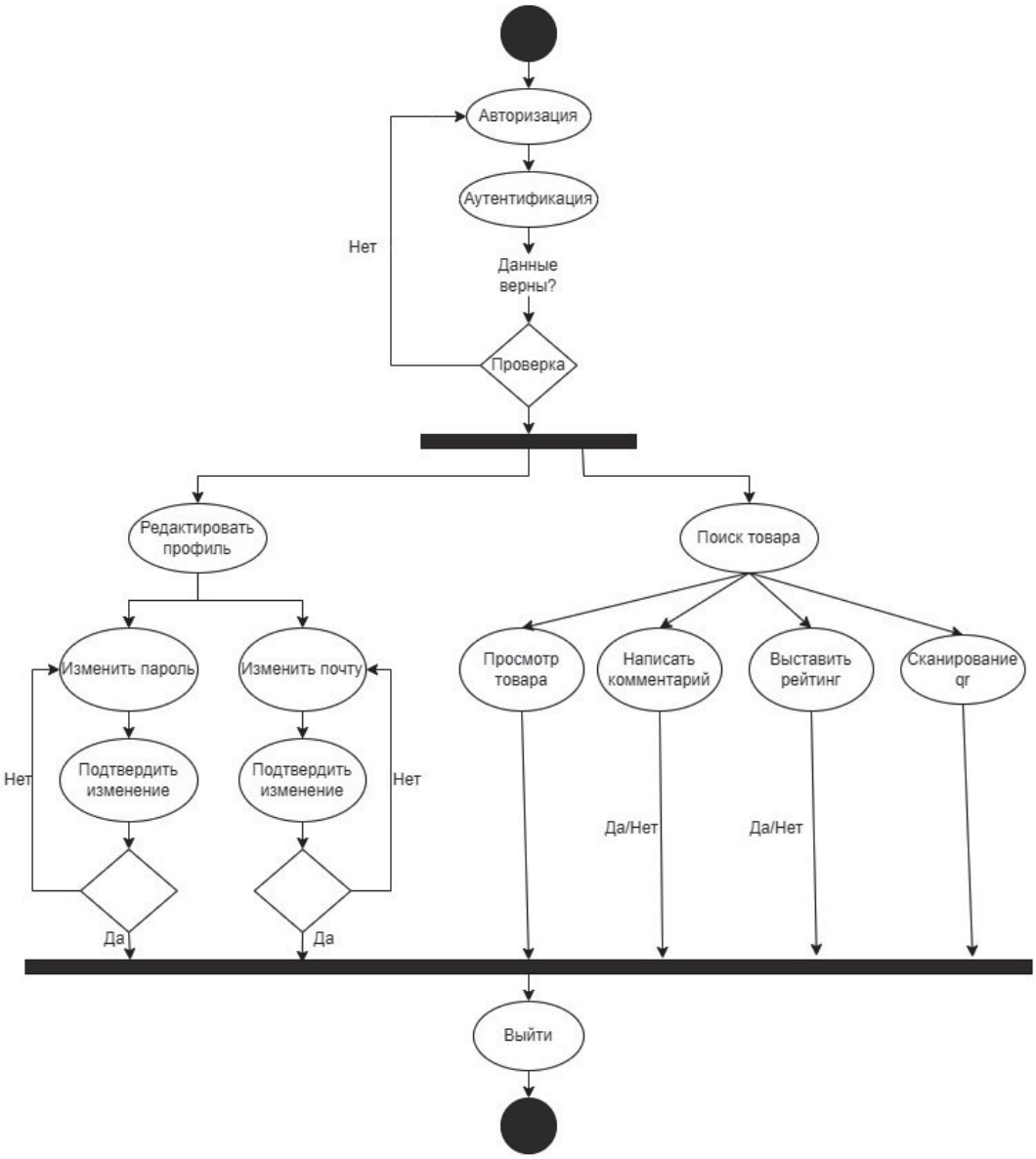


Рисунок 4.3 – Диаграмма деятельности

Диаграммы состояний изображают состояния и переходы (Рисунок 4.4). Состояние относится к различным комбинациям информации, которую может содержать объект, и эта диаграмма UML может визуализировать все возможные состояния и способ перехода объекта из одного состояния в другое.

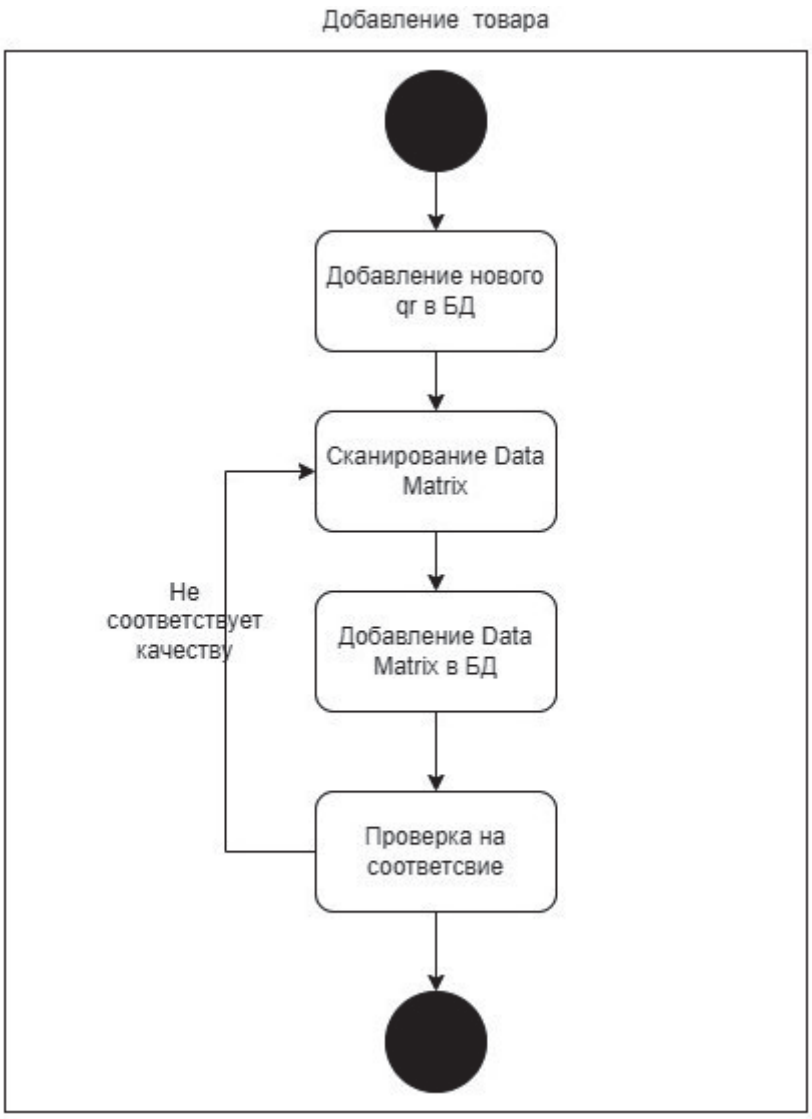


Рисунок 4.4 – Диаграмма состояний

Диаграмма компонентов, по существу, более специализированная версия класса правил схема-то же обозначения применяются для обоих (Рисунок 4.5). Диаграмма компонентов разбивает сложную систему на более мелкие компоненты и визуализирует отношения между этими компонентами.

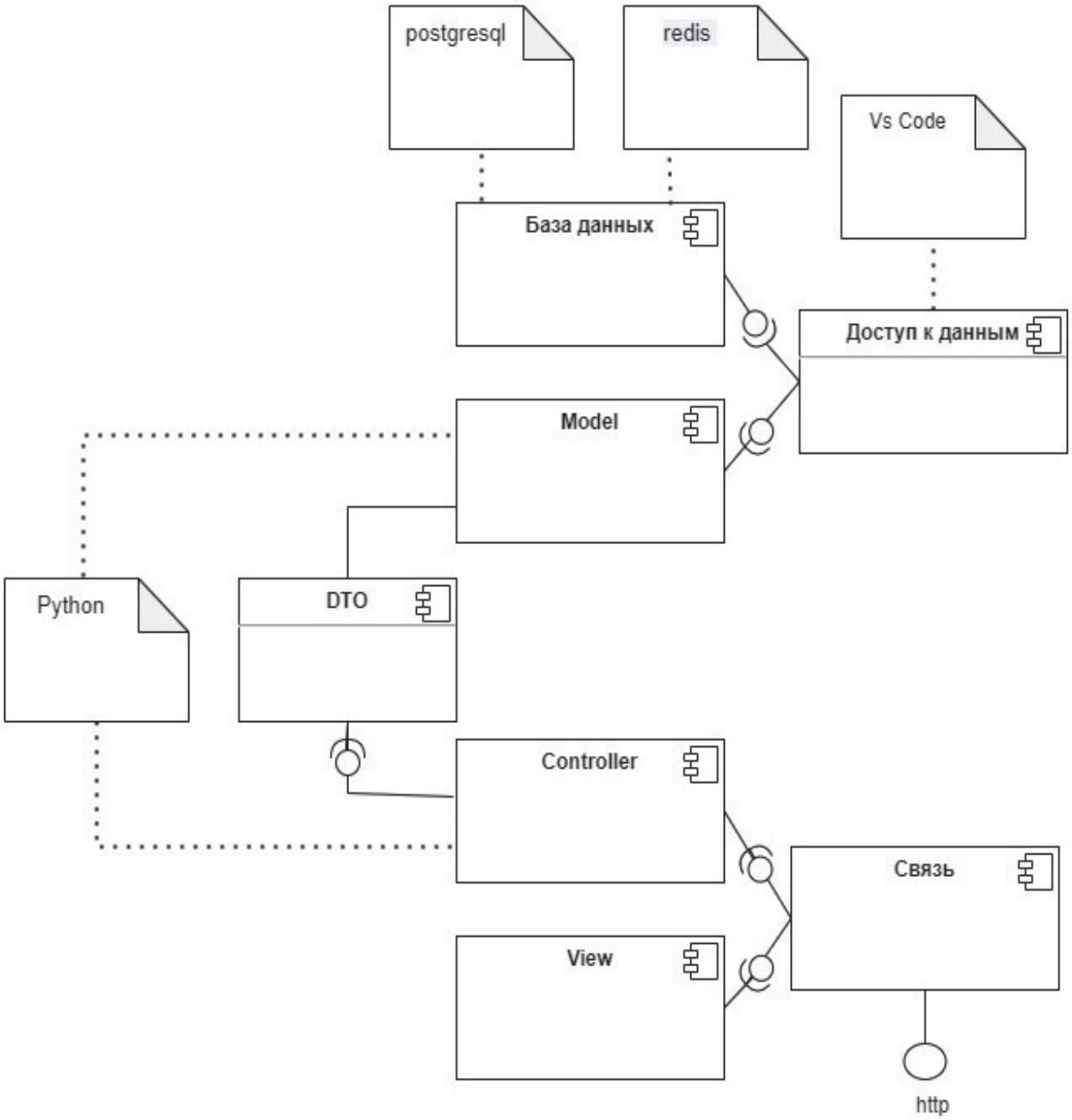


Рисунок 4.5 – Диаграмма компонентов

Выбор веб-сервера. Apache HTTP Server, является свободным и открытым исходным кодом кроссплатформенный веб-сервер/программное обеспечение, выпущенное в соответствии с условиями лицензии Apache License 2.0. Apache разрабатывается и поддерживается открытым сообществом разработчиков под эгидой Apache Software Foundation (Таблица 4.3).

Nginx позиционируется производителем как простой, быстрый и надёжный сервер, не перегруженный функциями. Применение nginx целесообразно прежде всего для статических веб-сайтов и как обратного прокси-сервера перед динамическими.

IIS (Internet Information Services) – большой набор серверов для нескольких служб Интернета от компании Microsoft. Основным компонентом IIS является веб-сервер, который позволяет размещать в Интернете сайты. IIS поддерживает протоколы HTTP, HTTPS, FTP, POP3, SMTP, NNTP [3].

Таблица 4.3 – Сравнение Apache, Nginx, IIS

Аспекты	Apache	Nginx	IIS
Поддержка ОС	Windows,MacOS X,BSD,Linux,Solaris,VMS	Windows,MacOS X,BSD,Linux,Solaris	Windows
Поддержка ASP.NET	–	–	+
Виртуальное управление	+	+	+
Консоль администрирования	+	–	+
Поддержка HTTPS	+	+	+
Поддержка IPV6	+	+	+
Поддержка FastCGI	+	+	+

Подводя итог всему выше сказанному, проанализировав данную таблицу выберем Nginx, так как обладает открытым исходным кодом, имеет поддержку большего количества операционных систем.

Выбор базы данных. База данных представляет собой организованную совокупность данных, как правило, хранятся и доступ в электронном виде с компьютерной системой. Там, где базы данных более сложны, они часто разрабатываются с использованием формальных методов проектирования и моделирования.

Система управления базами данных (СУБД) – это программное обеспечение, которое взаимодействует с конечными пользователями, приложениями и самой базой данных для сбора и анализа данных. Программное обеспечение СУБД дополнительно включает в себя основные средства, предоставляемые для администрирования базы данных. Общая сумма базы данных, СУБД и связанных приложений может называться «системой базы данных». Часто термин «база данных» также используется для произвольной ссылки на любую из СУБД, систему базы данных или приложение, связанное с базой данных [12].

Для разработки приложения подойдут реляционные базы данных. Эти базы данных классифицируются по набору таблиц, в которые данные попадают в заранее определенную категорию. Таблица состоит из строк и столбцов, где столбец содержит запись для данных для конкретной категории, а строки содержат экземпляр для этих данных, определенных в соответствии с категорией. Язык структурированных запросов (SQL) – это стандартный пользовательский и прикладной программный интерфейс для реляционной базы данных.

Существуют различные простые операции, которые можно применять над таблицей, что облегчает расширение этих баз данных, объединение двух баз данных с общим отношением и изменение всех существующих приложений.

Рассмотрим и сравним основные базы данных, использующиеся на рынке, выберем лучшую.

SQLite – это встроенная файловая СУБД, которая не требует установки или настройки. Это, в свою очередь, означает, что приложение не запускается под отдельным серверным процессом, который необходимо запустить, остановить или настроить. Эта безсерверная архитектура обеспечивает совместимость базы данных.

Полная база данных SQL содержится в одном файле на диске, и все операции чтения и записи происходят непосредственно в этом файле на диске. Поскольку данные напрямую записываются обратно в файл на диске, SQLite придерживается свойств ACID для защиты транзакций от сбоев выделения памяти и ошибок ввода-вывода диска, которые могут возникнуть в результате неожиданных сбоев системы или сбоев питания [2].

MySQL – одна из самых популярных и широко распространенных СУБД с открытым исходным кодом (Таблица 4.4). В отличие от SQLite, он использует архитектуру сервер / клиент, которая состоит из многопоточного сервера SQL. Эта многопоточная природа MySQL обеспечивает большую производительность, поскольку потоки ядра могут легко использовать несколько процессоров. База данных написана на C и C ++ и поддерживает различные платформы, такие как

операционные системы Windows Server и дистрибутивы Linux, такие как RHEL 7 и Ubuntu. Он также придерживается системы ACID для обеспечения согласованности транзакций и предоставляет различные Connectors и API, такие как C, C ++, Java, PHP и т. д.

PostgreSQL – это СУБД с открытым исходным кодом, в которой особое внимание уделяется расширяемости и соответствию стандартам. Как и MySQL, PostgreSQL использует модель базы данных клиент / сервер, а процесс сервера, который обрабатывает связь с клиентом, управляет файлами базы данных и операциями, известен как процесс postgres.

Чтобы упростить этот выбор, мы составили список сравнений SQLite, MySQL и PostgreSQL, которые могут помочь вам принять решение.

Непосредственно для разработки мобильного приложения нам больше подойдет PostgreSQL, так как больше остальных подходит для разработки веб-сайтов, веб-приложений, поэтому остановимся на этой базе данных.

База для кэширования данных. Для повышения скорости работы приложения, повысить производительность операций ввода-вывода в секунду для многих рабочих нагрузок необходимо выбрать базу данных для кэширования. Сравним базы данных, которые предназначены для конкретной функции.

Redis (расшифровывается как Remote Dictionary Server) – это быстрое хранилище данных типа «ключ-значение» в памяти с открытым исходным кодом. Redis Redis указан как *«с открытым исходным кодом (лицензия BSD), хранилище структуры данных в памяти, используемое в качестве базы данных, кэша и брокера сообщений».* Чтобы немного расширить это, Redis — это способ хранения пар ключ-значение в наборе различных типов данных, таких как списки, наборы и хэши. Redis хранит эти данные в памяти, а это означает, что он очень быстро возвращает данные по запросу. Эта скорость делает его идеальным в качестве кэша для вашего приложения, где вам нужно запрашивать и возвращать данные, а скорость является важным фактором.

Таблица 4.4 – Сравнение баз данных

	SQLite	MySQL	PostgreSQL
Архитектура	Файловая (встроенная)	Клиент/сервер	Клиент/сервер
Серверная ОС	Нет(без сервера)	FreeBSD,Linux, OSX,Solaris,Windows	FreeBSD,HP-UX,Linux, NetBSD,OpenBSD, OSX,Solaris, Unix, Windows
Транзакционная последовательность	ACID	ACID	ACID
Копирование	Нет	Мастер-Ведомая Репликация, Мастер-Мастер Репликация	Мастер-Ведомая Репликация
Язык программирования (базовый код)	C	C, C ++	C
Поддерживаемые языки программирования	ActionScript, Ada, Basic, C, C #, C ++, D, Delphi, Forth, Fortran, Haskell, Java, JavaScript, Lisp, Lua, MATLAB, PHP, PL / SQL	C, C ++, Delphi, Perl, Java, Lua, .NET, Node.js, PHP	NET, C, C ++, Delphi, Java, Perl, PHP, Python, Tcl
Издание	Сообщество с возможностью профессиональной поддержки	Сообщество (бесплатно), Стандарт и Предприятие	Сообщество с возможностью коммерческой поддержки
Популярные варианты использования	Сайты с низким средним трафиком	Веб-сайты, веб-приложения, стек LAMP, приложения на основе OLTP	Аналитика, интеллектуальный анализ данных
DBaaS	Нет	База данных Azure для MySQL , Amazon RDS для MySQL , Google Cloud SQL для MySQL	База данных Azure для PostgreSQL , Amazon RDS для PostgreSQL , Google Cloud SQL для PostgreSQL

Memcached описывает Memcached как «*бесплатную высокопроизводительную систему кэширования объектов с распределенной памятью с открытым исходным кодом*». Как и Redis, Memcached – это способ с открытым исходным кодом для хранения пар ключ-значение в памяти, что означает, что данные извлекаются очень быстро. Это делает Memcached еще одним способом возврата данных, где важна скорость. Memcached также является многопоточным, а это означает, что могут быть некоторые

улучшения производительности, когда ваше приложение может использовать несколько ядер.

Создадим сравнительную таблицу, чтобы наглядно отобразить плюсы и минусы баз данных (Таблица 4.5).

Таблица 4.5 – Сравнение баз данных для кэширования

	Memcached	Redis
Задержка менее миллисекунды	+	+
Простота использования	+	+
Разделение данных	+	+
Поддержка языков программирования	+	+
Расширенные структуры данных	-	+
Многопоточная архитектура	+	-
Snapshots	-	+
Репликация	-	+
Кластеризация	-	+

В первую очередь нужно пояснить некоторые понятия. Snapshots означает файл *резервной копии базы данных Redis*, который позволяет сделать полный снимок текущего состояния Redis. RDB – это компактный способ хранения текущего состояния Redis в файле, который можно перенести в другое место (то есть вне площадки для аварийного восстановления). RDB работает, создавая дочерний процесс экземпляра Redis и позволяя дочернему процессу завершить резервное копирование. Это означает, что родительский процесс Redis не занят созданием RDB-файла и записью на диск, что позволяет ему выполнять обычную работу Redis. RDB также хорошо работает с большими наборами данных в журнале AOF, поскольку ему не нужно воспроизводить экземпляр Redis построчно [12].

Репликация – это способ копирования данных из одного экземпляра в другой с созданием *реплики*. Цель состоит в том, чтобы обеспечить хранение дубликатов данных в другом экземпляре на случай, если главный экземпляр выйдет из строя. Redis изначально поддерживает репликацию и может выполнять репликацию в режиме мастер-последователь. Экземпляры Redis также могут иметь более одной реплики для дополнительной избыточности. Memcached не поддерживает репликацию без стороннего программного обеспечения.

Кластеризация – это способ обеспечения высокой доступности службы путем создания нескольких экземпляров и их объединения в

кластер. Создание кластеров иногда может быть сложным в настройке и управлении, если программное обеспечение не предназначено для этого. В качестве решения для масштабируемого кэширования Redis предлагает Redis Cluster, который обеспечивает функции кластеризации для Redis. Есть много преимуществ использования Redis в кластере, а не только в качестве отдельного экземпляра:

Списки Redis – это способ хранения неупорядоченных данных, к которым затем можно получить доступ по индексу. Redis реализует то, что известно, как *связанный список*, который представляет собой несколько иной способ реализации списков, чем *массив.* Это означает, что Redis чрезвычайно быстр и последователен при добавлении или удалении значения из начала или конца списка

Благодаря данным преимуществам выберем Redis для разработки мобильного приложения.

Проектирование базы данных. В данном пункте рассмотрены логическая и физическая схема базы данных корпоративного портала (Рисунок 4.6 и 4.7).

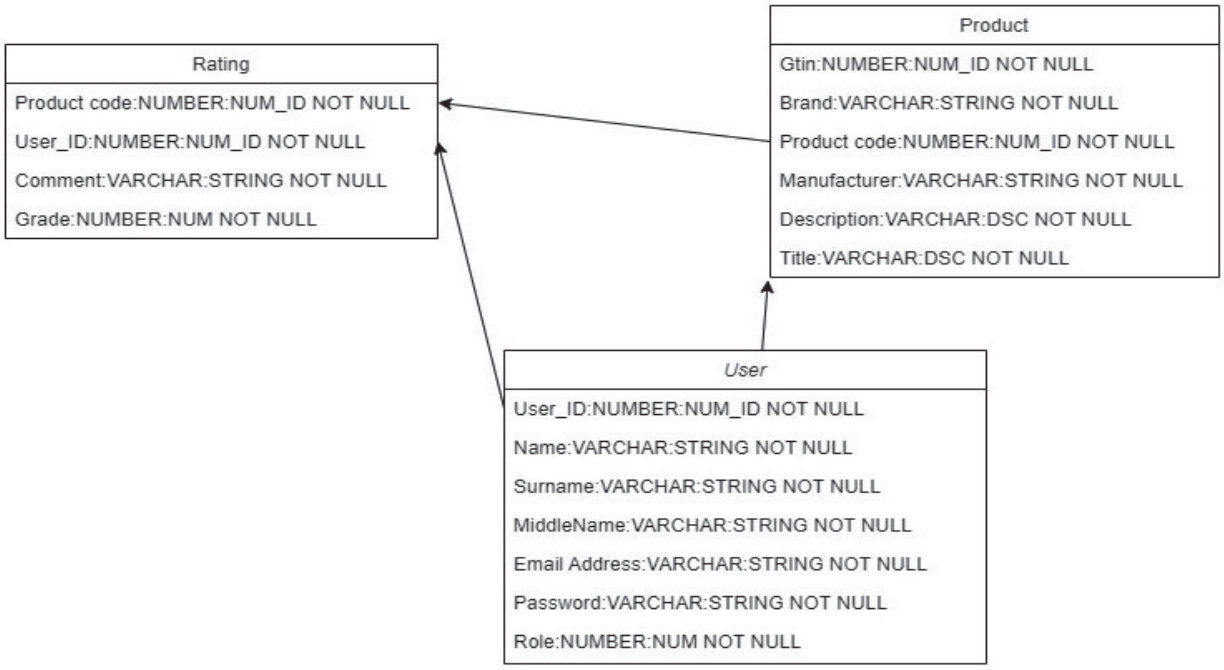


Рисунок 4.6 – Физическая схема базы данных

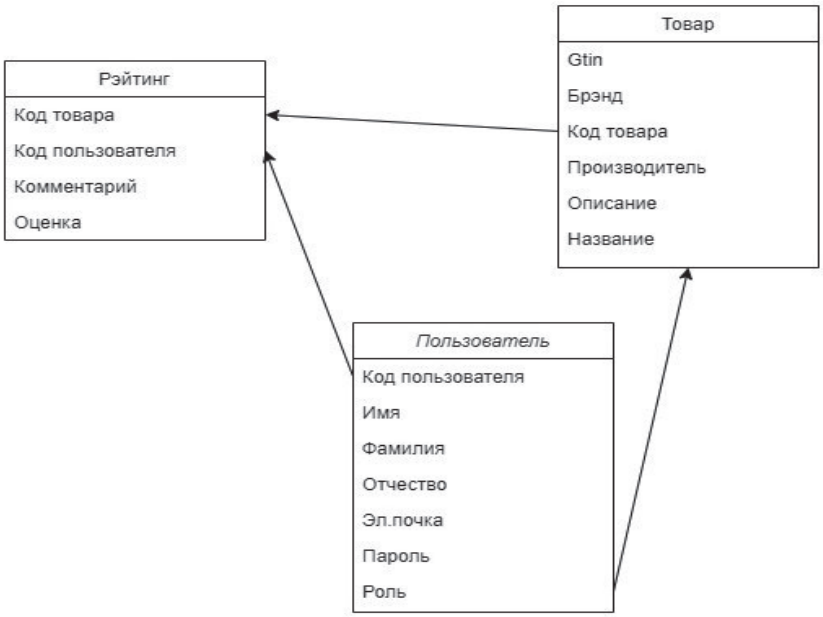


Рисунок 4.7 – Логическая схема базы данных

На физическом уровне атрибуты базы данных отображают типы данных логической модели, а на логическом уровне данные представляются так, как называются, в сущности.

Разработка макета страниц. В первую очередь создадим файл с отображением информации на главной станице (Рисунок 4.8).

```
import React from 'react';
import Comment from '../components/Comment';
import GoodCard from '../components/GoodCard';

const Home = ({companyName, productName, brandName, amount, consist, author, time, rate, comment }) => {
    return (
        <div className="container max-w-md flex flex-col justify-between items-center mx-auto py-4 px-2">
            <form class="flex items-center w-full">
                <label for="simple-search" class="sr-only">Search</label>
                <div class="relative w-full">
                    <div class="flex absolute inset-y-0 left-0 items-center pl-3 pointer-events-none">
                        <svg class="w-5 h-5 text-gray-500 dark:text-gray-400" fill="currentColor" viewBox="0 0 20 20" xmlns="http://www.w3.org/2000/svg"><path fill-rule="ever
                    </div>
                    <input type="text" id="simple-search" class="bg-gray-50 border border-gray-300 text-gray-900 text-sm rounded-lg focus:ring-blue-500 focus:border-bl
                </div>
                <button type="submit" class="p-2.5 ml-2 text-sm font-medium text-white bg-blue-700 rounded-lg border border-blue-700 hover:bg-blue-800 focus:ring-4 focus
                    <svg class="w-5 h-5" fill="none" stroke="currentColor" viewBox="0 0 24 24" xmlns="http://www.w3.org/2000/svg"><path stroke-linecap="round" stroke-linejoin="ro
                </button>
                <button type="submit" class="p-2.5 ml-2 text-sm font-medium text-white bg-blue-700 rounded-lg border border-blue-700 hover:bg-blue-800 focus:ring-4 focus
                    <svg class="w-5 h-5" fill="none" stroke="currentColor" viewBox="0 0 24 24" xmlns="http://www.w3.org/2000/svg"><path stroke-linecap="round" stroke-linejoin="ro
                </button>
            </form>
            <GoodCard
                companyName={companyName}
                productName={productName}
                brandName={brandName}
                amount={amount}
                consist={consist}
            />
            <form class="flex items-center w-full">
                <button id="dropdown-button" data-dropdown-toggle="dropdown" class="flex-shrink-0 z-10 inline-flex items-center py-2.5 px-4 text-sm font-medium text-center text
                <div id="dropdown" class="hidden z-10 w-44 bg-white rounded divide-y divide-gray-100 shadow dark:bg-gray-700">
                    <ul class="py-1 text-sm text-gray-700 dark:text-gray-200" aria-labelledby="dropdown-button">
                        <li>
                            <a href="#" class="block py-2 px-4 hover:bg-gray-100 dark:hover:bg-gray-600 dark:hover:text-white">1</a>
                        </li>
                        <li>
                            <a href="#" class="block py-2 px-4 hover:bg-gray-100 dark:hover:bg-gray-600 dark:hover:text-white">2</a>
```

Рисунок 4.8 – Главная страница

Добавляем необходимые кнопки, а также задаем им и тексту размер и цвет (Рисунок 4.9 и 4.10). Определенные функции и адреса будем задавать позже.

Рисунок 4.9 – Подвал главной страницы

Рисунок 4.10 – Шапка страницы

Отдельное подключение функции шапки позволяет, не включать лишний код на каждую страницу приложения.

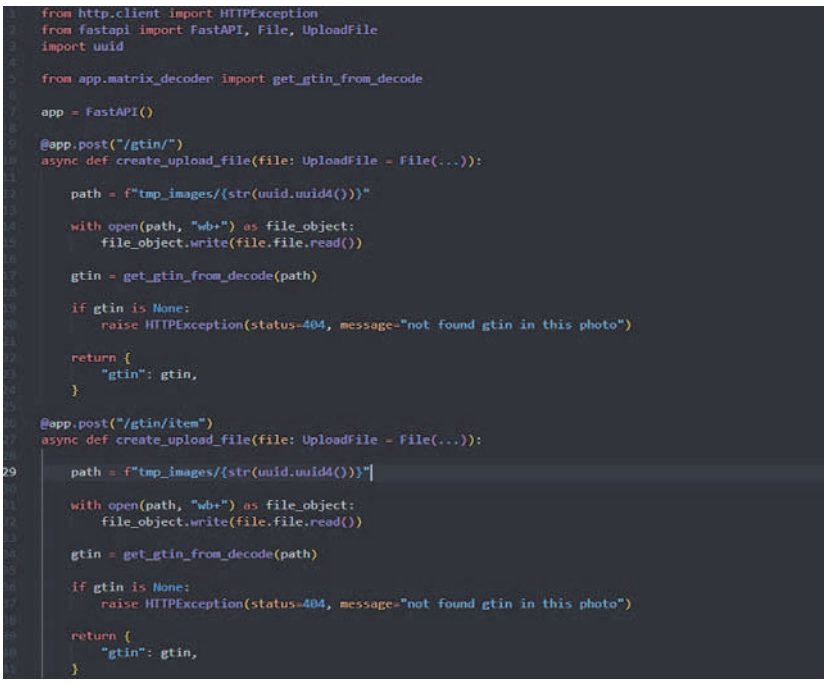
Далее добавляем форму регистрации и авторизации. Данные для регистрирования в приложении: Имя, фамилия, дата рождения, телефонный номер, электронная почта, пароль, подтверждение пароля (Рисунок 4.11).

Рисунок 4.11 – Страница регистрации

Для авторизации оставляем только электронную почту и пароль (Рисунок 4.12). Подметим, что цвет кнопок приходится добавлять каждый раз.

Рисунок 4.12 – Код авторизации

Следующим этапом будет создание модулей основных функций приложения. В первую очередь необходим backend-модуль, который анализирует фотографию data matrix-кода и выдает информацию о товаре (Рисунок 4.13).



```python
from http.client import HTTPException
from fastapi import FastAPI, File, UploadFile
import uuid

from app.matrix_decoder import get_gtin_from_decode

app = FastAPI()

@app.post("/gtin/")
async def create_upload_file(file: UploadFile = File(...)):

    path = f"tmp_images/{str(uuid.uuid4())}"

    with open(path, "wb+") as file_object:
        file_object.write(file.file.read())

    gtin = get_gtin_from_decode(path)

    if gtin is None:
        raise HTTPException(status=404, message="not found gtin in this photo")

    return {
        "gtin": gtin,
    }

@app.post("/gtin/item")
async def create_upload_file(file: UploadFile = File(...)):

    path = f"tmp_images/{str(uuid.uuid4())}"

    with open(path, "wb+") as file_object:
        file_object.write(file.file.read())

    gtin = get_gtin_from_decode(path)

    if gtin is None:
        raise HTTPException(status=404, message="not found gtin in this photo")

    return {
        "gtin": gtin,
    }
```

Рисунок 4.13 – Модуль сканирования

Далее создаем модуль, который по gtin (номер товара) достает информацию с официального сайта (Рисунок 4.14).

```python
import requests
from bs4 import BeautifulSoup

from .model import Item

GOODS_URL = "https://srs.gs1ru.org/id/gtin/"

def get_item_info(gtin: str):
    url = GOODS_URL + gtin

    r = requests.get(url)

    parsed_html = BeautifulSoup(r.text, "html.parser")

    company_name = parsed_html.select_one('.product-card__header-company').text.strip()
    product_name = parsed_html.select_one('.product-card__header-product-name').text.strip()

    product_table = parsed_html.find_all('table')[1]
    brand_name = product_table.find_all('tr')[2].find_all('span')[1]
    amount = product_table.find_all('tr')[3].find_all('span')[1].text
    consist = product_table.find_all('tr')[4].find_all('span')[1].text

    return Item(
        company=company_name,
        product=product_name,
        brand=brand_name,
        amount=amount,
        consist=consist,
    )
```

Рисунок 4.14 – Модуль Gtin

Перед отправкой данных на сервер важно убедиться, что все обязательные поля, например, при регистрации, заполнены правильно, то есть в корректном формате, поэтому модуль для валидации форм крайне важная часть в приложении (Рисунок 4.15).

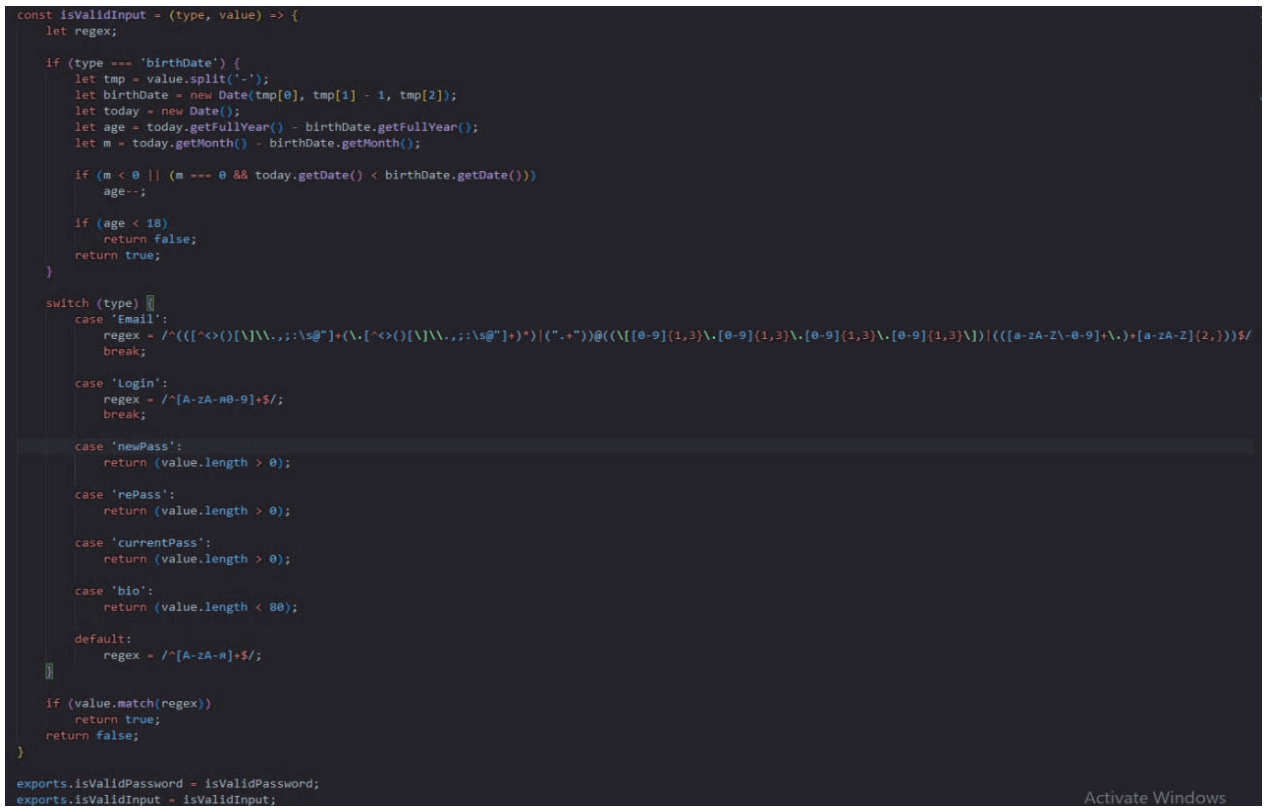
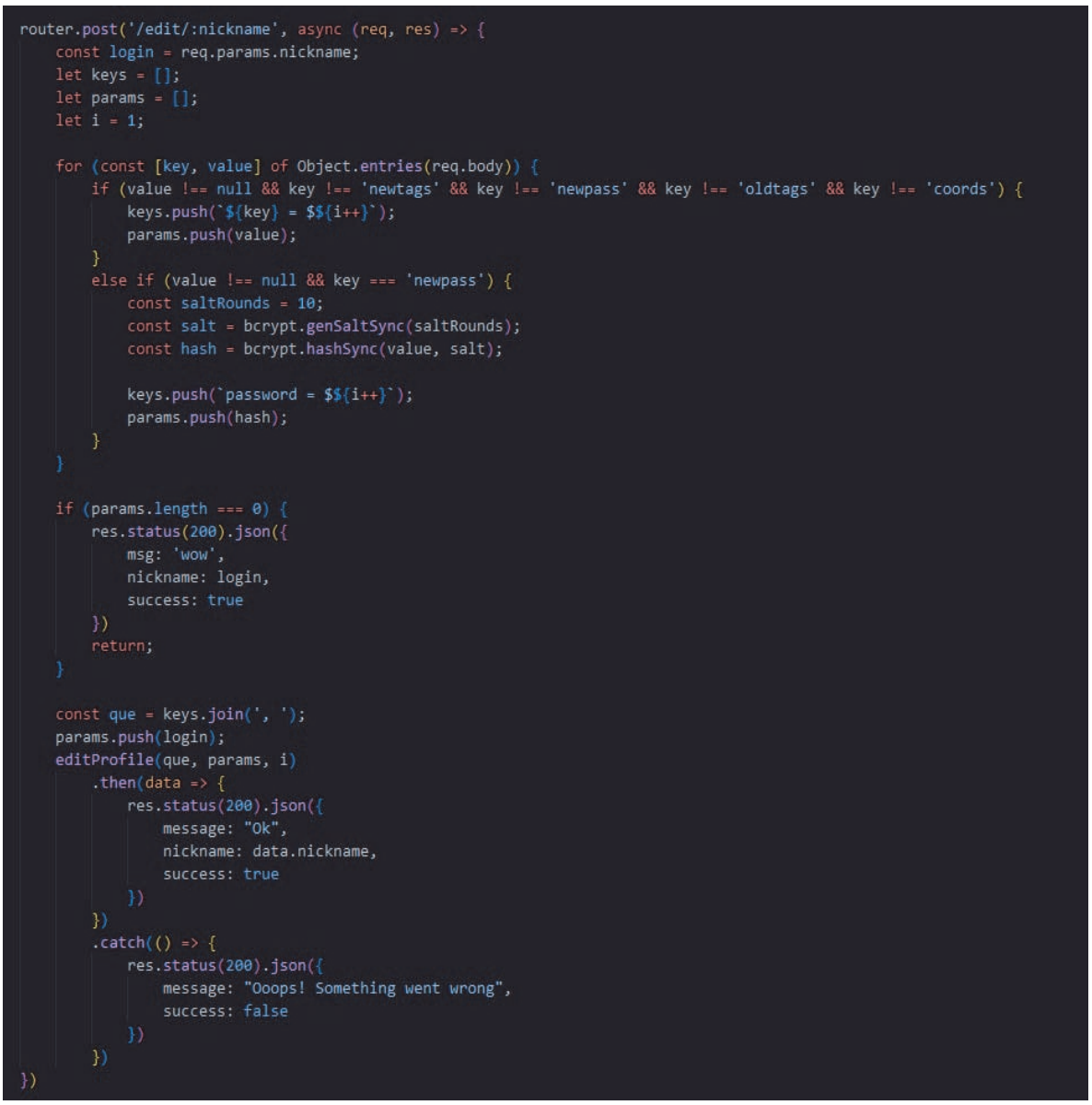


Рисунок 4.15 – Валидация форм

Для того чтобы появилась возможность изменения пароля, смены почты, телефонного номера пропишем необходимые функции (Рисунок 4.16).

Для смены пароля необходимо для начала ввести старый пароль, и для его проверки создаем функцию, которая обращается в базе данных с конкретным запросом проверки пароля.



```
router.post('/edit/:nickname', async (req, res) => {
    const login = req.params.nickname;
    let keys = [];
    let params = [];
    let i = 1;

    for (const [key, value] of Object.entries(req.body)) {
        if (value !== null && key !== 'newtags' && key !== 'newpass' && key !== 'oldtags' && key !== 'coords') {
            keys.push(`${key} = $${i++}`);
            params.push(value);
        }
        else if (value !== null && key === 'newpass') {
            const saltRounds = 10;
            const salt = bcrypt.genSaltSync(saltRounds);
            const hash = bcrypt.hashSync(value, salt);

            keys.push(`password = $${i++}`);
            params.push(hash);
        }
    }

    if (params.length === 0) {
        res.status(200).json({
            msg: 'wow',
            nickname: login,
            success: true
        })
        return;
    }

    const que = keys.join(', ');
    params.push(login);
    editProfile(que, params, i)
        .then(data => {
            res.status(200).json({
                message: "Ok",
                nickname: data.nickname,
                success: true
            })
        })
        .catch(() => {
            res.status(200).json({
                message: "Ooops! Something went wrong",
                success: false
            })
        })
})
```

Рисунок 4.16 – Смена данных в личном кабинете

В нашем случае необходимо выделить сотрудников, которые будут добавлять, редактировать информацию о товарах, а также собирать необходимую статистику для дальнейшего использования (Рисунок 4.17).

```python
async def get_current_user(token: str = Depends(oauth2_scheme)):
    credentials_exception = HTTPException(
        status_code=status.HTTP_401_UNAUTHORIZED,
        detail="Could not validate credentials",
        headers={"WWW-Authenticate": "Bearer"},
    )
    try:
        payload = jwt.decode(token, SECRET_KEY, algorithms=[ALGORITHM])
        username: str = payload.get("sub")
        if username is None:
            raise credentials_exception
        token_data = TokenData(username=username)
    except JWTError:
        raise credentials_exception
    user = get_user(db, username=token_data.username)
    if user is None:
        raise credentials_exception
    return user


async def get_current_active_user(current_user: User = Depends(get_current_user)):
    if current_user.disabled:
        raise HTTPException(status_code=400, detail="Inactive user")
    return current_user
```

Рисунок 4.17 – Проверка доступа

Для вывода статистики необходимо собрать данные, которые представим в виде графика, что позволяет провести анализ посещения определенного товара или определенного продукта в сравнении с другими. (Рисунок 4.18)

```jsx
import React from 'react';

import {
    XYPlot,
    XAxis,
    YAxis,
    VerticalGridLines,
    HorizontalGridLines,
    VerticalBarSeries,
} from 'react-vis';

const Stat = ({greenData }) => {
    return (
        <div className="container max-w-md flex flex-col justify-between items-center mx-auto py-8 px-2">
            <XYPlot margin={{ top: 150 }} xType="ordinal" width={800} height={600} xDistance={100}>
                <VerticalGridLines />
                <HorizontalGridLines />
                <XAxis top={30} tickLabelAngle={-90} />
                <YAxis />
                <VerticalBarSeries className="vertical-bar-series-example" data={greenData} />
            </XYPlot>
        </div>
    )
}

export default Stat;
```

Рисунок 4.18 – Вывод статистики

Реализация мобильного приложения. При запуске приложения в первую очередь необходимо зарегистрироваться. Перечень данных для регистрации такой: имя, фамилия, дата рождения, номер телефона, электронная почта, пароль, подтверждение пароля. Этот список идентичен тому, что было прописано в коде страницы. Каждый из этих пунктов можно отредактировать в личном кабинете (Рисунок 4.19).

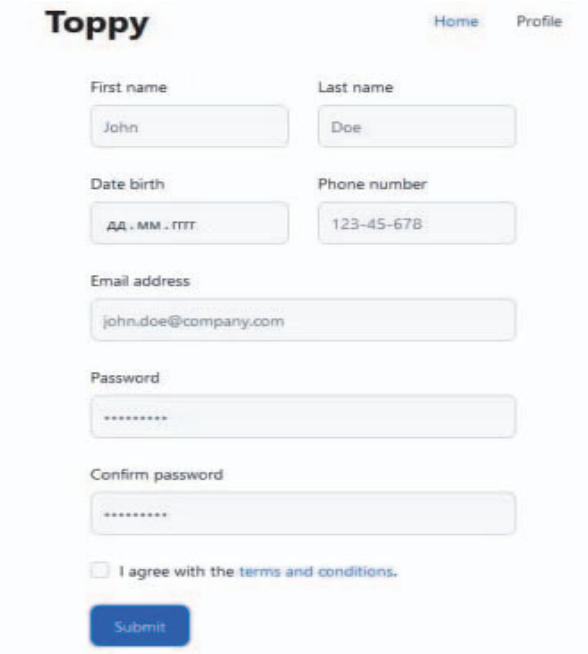


Рисунок 4.19 – Страница регистрации

Повторное посещение приложения подразумевает аутентификацию. Для этого необходимо ввести электронную почту и пароль, к тому же есть возможность запомнить данные, которые введены пользователем (Рисунок 4.20).

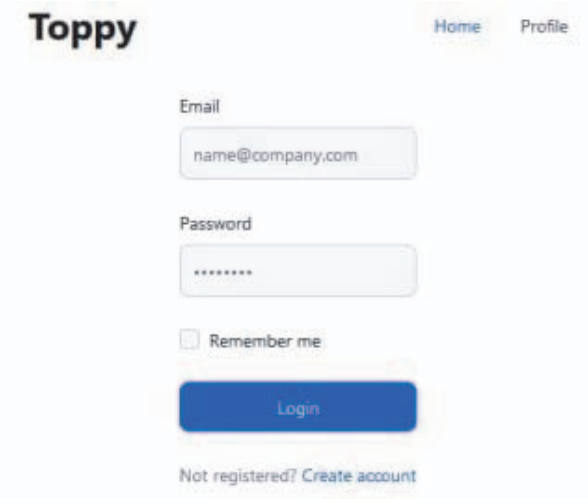


Рисунок 4.20 – Страница авторизации

Отображение нужного товара после сканирования или поиска этого товара в списке, позволяет пользователю получить имеющуюся информацию о продукте, оставить комментарий и поставить оценку (Рисунок 4.21).

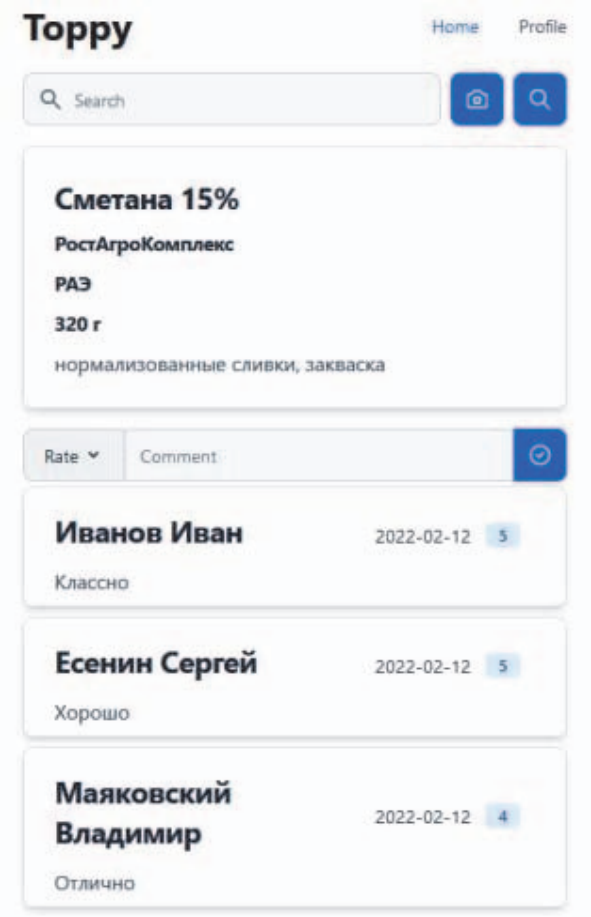


Рисунок 4.21 – Отзыв о товаре

Модуль проверки доступа необходим для возможности вывода определенной статистики. Администратор может выбрать данные, которые ему нужны и отобразить из в виде графика (Рисунок 4.22).

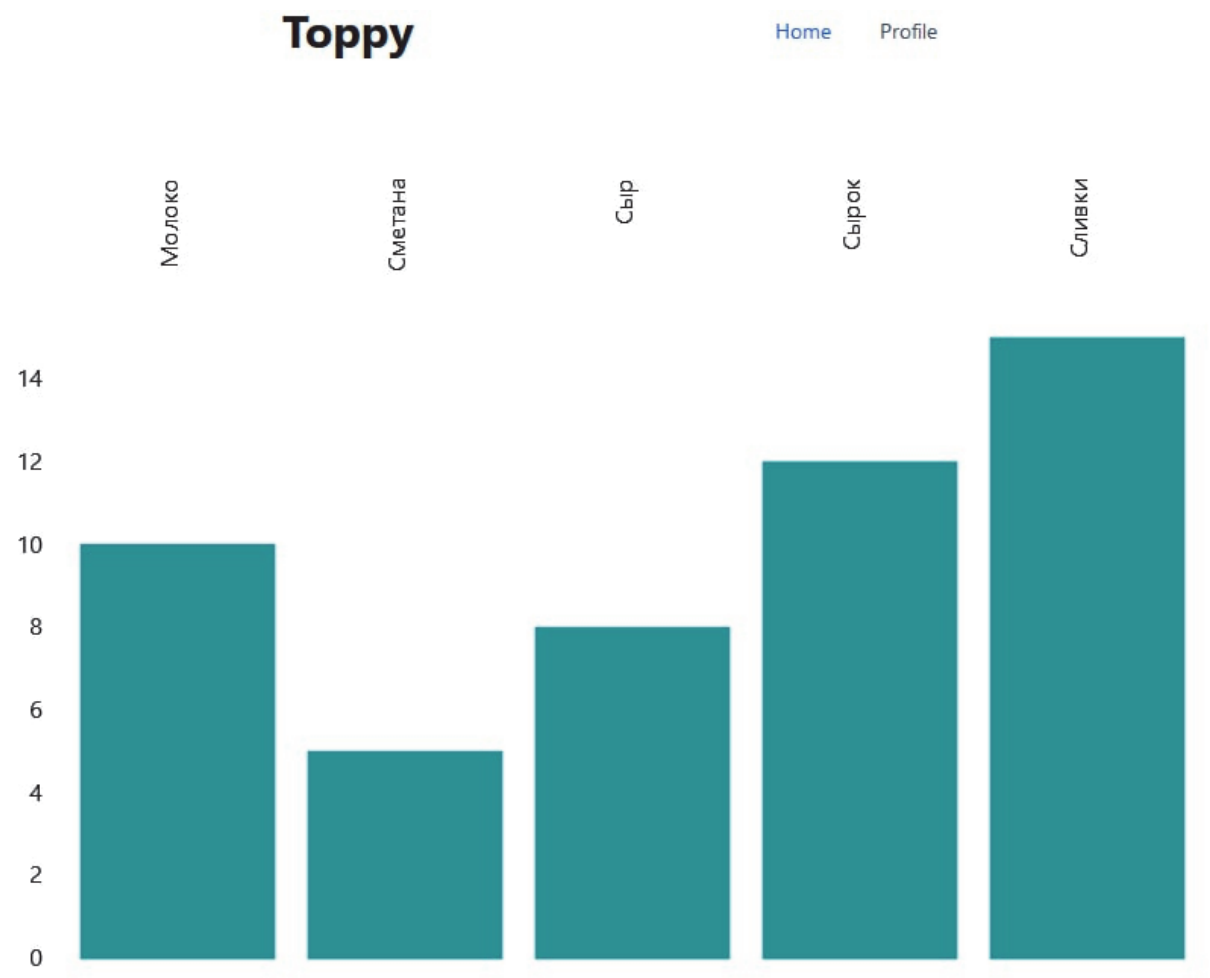


Рисунок 4.22 – Статистика по посещению

Основная цель такой функции – это наглядное отображение данных о товарах, что позволяет в дальнейшем проводить анализ спроса на продукцию.

Расчет экономической эффективности от разработки мобильного приложения. Расчет экономической эффективности проекта мобильного приложения начинается с учета капитальных затрат (*К*) на создание и дальнейшего внедрение представительства, которые составляют 188 тыс. руб. Предполагаемая годовая чистая прибыль (*Эг*), получаемая при использовании web-представительства за первый год эксплуатации, равна:

50 тыс. руб. х 12 мес. = 600 тыс. руб. (4.5)

Примем нормативный коэффициент эффективности капитальных вложений (*Ен*) за 0,14.

Годовой экономический эффект (*Э*) составит:

600 тыс. руб. - 188 тыс. руб. х 0,14 = 573,7 тыс. руб. (4.6)

Коэффициент экономической эффективности капитальных вложений (*Е*) будет равен:

600 тыс. руб. / 188 = 3,2, (4.7)

что превышает размер нормативного коэффициента эффективности капитальных вложений.

Срок окупаемости капитальных вложений (*Т*) равен:

188 тыс. руб. : 50 тыс. руб. = 3,8 месяца (4.8)

Все три критерия соответствуют поставленным условиям (*Э>0, Е>Ен, Т>1/Ен*), следовательно, разработка проекта мобильного приложения является эффективной и оправданной.

Таким образом, на основе анализа основных подходов к проектированию мобильного приложения и технологии сканирования с выделением наиболее оптимальных с точки зрения простоты разработки, всестороннего исследования инструментов разработки с построением UML-диаграмм спроектировано мобильное приложение.

Разработанное мобильное приложение-сканер молочной продукции позволяет потребителям получать сведения о покупаемом товаре, производителям и продавцам анализировать потребительские предпочтения при проведении маркетинговых исследований;

Начиная с изучения теоретических основ и до окончания разработки, были выделены главные преимущества использования нашего мобильного приложения. В отличие от более дорогих существующих приложений на рынке конкретное приложение не требует серьезной доработки и затрат на пользование, за исключением масштабирования и повышения трафика. Таким образом, в ходе проведённых исследований была подтверждена актуальность разработки мобильного приложения.

Проведенная оценка экономической эффективности разработки мобильного приложения и затрат на его внедрение и продвижение позволила отметить преимущества приложения и обоснованность его использования для различных категорий пользователей.

Разработанное приложение «Торру» для сканирования молочной продукции внедрено в деятельность ООО ТД «Партнер»

и применяется в маркетинговых исследованиях анализа потребительских предпочтений для определения приоритетных направлений реализации продукции ООО.

Библиографический список к главе 4

1. Архипова, М.Ю. Анализ международной практики внедрения цифровизации в агропромышленный комплекс национальных и наднациональных экономик, на примере стран с традиционно развитым сельским хозяйством: аналитические материалы / М.Ю. Архипова, М.В. Кагирова, А.В. Уколова, Ю.Н. Романцева, А.Е. Харитонова, В.В. Демичев. – Москва: Издательство «Научный консультант», 2021. – 118 с. – Текст: непосредственный.

2. Аникеев, С.В. Разработка приложений баз данных в Delphi: Самоучитель / С.В. Аникеев, А.В. Маркин. – М.: ДИАЛОГ-МИФИ, 2013. – 160 с. – Текст: непосредственный.

3. Афоничкин, А.И. Разработка бизнес-приложений в экономике на базе MS Excel. / А.И. Афоничкин. – М.: ДИАЛОГ-МИФИ, 2003. – 416 с. – Текст: непосредственный.

4. Болотнов, А.М. Разработка программных приложений в среде BlackBox: Учебное пособие / А.М. Болотнов. – СПб.: Лань, 2018. – 144 с. – Текст: непосредственный.

5. Васильев, А.Е. Микроконтроллеры. Разработка встраиваемых приложений / А.Е. Васильев. – СПб.: BHV, 2012. – 207 с.– Текст: непосредственный.

6. Гецманн, П. Разработка приложений для Windows Phone. Архитектура, фреймворки, API / П. Гецманн. – СПб.: BHV, 2014. – 880 с. – Текст: непосредственный.

7. Гринберг, М. Разработка веб-приложений с использованием Flask на языке Python / М. Гринберг. – М.: ДМК, 2016. – 272 с. – Текст: непосредственный.

8. Есенин, С.А. DirectX и Delphi: разработка графических и мультимедийных приложений / С.А. Есенин. – СПб.: BHV, 2006. – 512 с. – Текст: непосредственный.

9. Заяц, А.М. Проектирование и разработка WEB-приложений. Введение в frontend и backend разработку на JavaScript и node.js: Учебное пособие / А.М. Заяц, Н.П. Васильев. – СПб.: Лань, 2019. – 120 с. – Текст: непосредственный.

10. Кагирова, М.В. Статистический анализ развития цифровой экономики в России / М.В. Кагирова // Экономика сельского хозяйства России. – 2019. – № 3. – С. 27-29. – Текст: непосредственный.

11. Колисниченко, Д.Н. PHP и MySQL. Разработка веб-приложений. Профессиональное программирование / Д.Н. Колисниченко. – СПб.: BHV, 2015. – 592 с. – Текст: непосредственный.

12. Крелль, Б.Е. Windows Mobile. Разработка приложений для КПК. / Б.Е. Крелль. – М.: ДМК Пресс, 2009. – 352 с. – Текст: непосредственный.

13. Машнин, Т.С. Eclipse: разработка RCP-, Web-, Ajax- и Android-приложений на Java / Т.С. Машнин. -– СПб.: BHV, 2013. – 384 с. – Текст: непосредственный.

14. Машнин, Т.С. JavaFX 2.0: разработка RIA-приложений / Т.С. Машнин. – СПб.: BHV, 2012. – 320 с. – Текст: непосредственный.

15. Нортроп, Т. Разработка защищенных приложений на Visual Basic .NET и Visual C# .NET: Учебный курс Microsoft / Т. Нортроп. – М.: Русская редакция, 2007. – 688 с. – Текст: непосредственный.

16. Полещук, В.И. AutoCAD 2004: разработка приложений и адаптация. / В.И. Полещук. – СПб.: BHV, 2004. – 624 с. – Текст: непосредственный.

17. Прохоренок, Н.А. Python 3 и PyQt. Разработка приложений / Н.А. Прохоренок. – СПб.: БХВ-Петербург, 2013. – 704 с. – Текст: непосредственный.

18. Пугачев, С.В. Разработка приложений для Windows 8 на языке C# / С.В. Пугачев, А.М. Шериев, К.А. Кичинский. – СПб.: БХВ-Петербург, 2013. – 416 с. – Текст: непосредственный.

19. Романцева, Ю.Н. Анализ технической обеспеченности сельскохозяйственных производителей в России / Ю.Н. Романцева // Экономика сельского хозяйства России. – 2019. – № 3. – С. 19-24. – Текст: непосредственный.

20. Рудикова, Л.В. Базы данных. Разработка приложений / Л.В. Рудикова. – СПб.: BHV, 2006. – 496 с. – Текст: непосредственный.

21. Соколова, Ю.С. Разработка приложений в среде Delphi. Ч. 1. Общие приемы программирования / Ю.С. Соколова, С.Ю. Жулева. – М.: ГЛТ, 2011. – 144 с. – Текст: непосредственный.

22. Соколова, Ю.С. Разработка приложений в среде Delphi. Ч.2.Компоненты и их использование / Ю.С. Соколова, С.Ю. Жулева. – М.: Горячая линия -Телеком, 2011. – 144 с. – Текст: непосредственный.

23. Ульянкин, А.Е. Автоматизация анализа международных эколого-экономических систем на основе технологий машинного обучения А.Е. Ульянкин // Электронный сетевой политематический журнал «Научные труды КУБГТУ». – 2020. – № 6. – с.74-83. – Текст: непосредственный.

24. Харитонова, А.Е. Анализ результатов Всероссийской сельскохозяйственной переписи с использованием методов машинного обучения / А.Е. Харитонова, А.А. Сундупей, С.А. Скачкова // Бухучет в сельском хозяйстве. – 2020. – № 12. – С. 41-48. – Текст: непосредственный.

25. Харитонова, А.Е. Статистическое исследование эколого-экономических систем сельского хозяйства / Дисс. на соиск. степени канд. экон. наук / Новосибирск: Новосибирский государственный университет экономики и управления "НИНХ", 2016. – 223 с. – Текст: непосредственный.

26. Эспозито, Д. Разработка современных веб-приложений: анализ предметных областей и технологий / Д. Эспозито. – М.: Вильямс И.Д., 2017. – 464 с. – Текст: непосредственный.

27. Chen, Y. Crowd-Sourcing Mobile Platform for Textbook Selling and Exchange Using Information Retrieval. / Y Chen, Y Sun, F Zhang // A In Proceedings of the 2019 Conference CS & IT Conference. – India: AIRCC Publishing Corporation: Tamil Nadu, 2019. – Текст: непосредственный.

28. Kagirova, M.V. Essential principles of working with statistical information: учебное пособие / M.V. Kagirova. – М: Изд-во РГАУ-МСХА им. К.А. Тимирязева, 2016. – 81 с. – Текст: непосредственный.

29. Malaperdas, G. Digitization in Archival Material Conservation Processes / Eur. J. Eng. Technol. Res, 2021. – № 6. – с. 30–32. – Текст: непосредственный.

30. Reis, A. Using online artificial vision services to assist the blind-an assessment of microsoft cognitive services and google cloud vision. / A. Reis, D. Paulino, V. Filipe, J. Barroso // In Trends and Advances in Information Systems and Technologies. – Germany: Springer, 2018. – p. 174–184. – Текст: непосредственный.

Глава 5 Системный подход к анализу больших данных АПК с использованием методов Data Science

В последнее время большие потоки данных стали повсеместными из-за того, что ряд приложений генерируют огромное количество данных с большой скоростью. Это затруднило применение существующих инструментов, технологий, методов и приемов анализа данных непосредственно к потокам больших данных из-за присущих им динамических характеристик.

Однако в условиях гиперболического роста информации область науки о данных все еще находится в зачаточном состоянии. Это область, которая возникла на пересечении нескольких дисциплин – статистики, бизнес-аналитики и информатики, подвержена бурной эволюции, изменчива и нестабильна.

В связи с чем, важно четко сформулировать системный подход к принятию лучших решений на основе анализа больших данных с применением Data Science, чтобы он мог быть методически применен к новым проблемам. Таким образом, все выше сказанное отмечает актуальность настоящего исследования.

5.1 Data Science как современный тренд развития IT в АПК

1. Одним из актуальных направлений применения Data Science в IT сфере АПК является анализ данных урожая в сельском хозяйстве. Управление урожаем проделало длинный путь от того, когда сельскохозяйственные производители, как правило, полагались на опыт и предложения от коллег - фермеров для понимания, какие зерновые культуры становятся лучшими в течение определенных времен года и других специфических особенностей.

Однако на настоящий момент сельхозпроизводители, могут участвовать в чем-то, названном биоразведкой, включающей определение того, какие растения облагают потенциальными активными или функциональными преимуществами для определенных рынков. После идентификационного процесса биохимики и другие специалисты выращивают тысячи вариантов тех растений через обычные технологии размножения.

Сегодняшнее программное обеспечение позволяет аналитикам выявлять тенденции в огромных объемах данных, задачи по

растениеводству, описанные выше, могут выполняться с большей эффективностью и меньшим количеством ошибок. Измеряя характеристики сельскохозяйственных полей и почв. и имея информацию о типах почв и прогнозируемом уровне осадков, можно делать прогнозы продуктивности и давать рекомендации, которые помогут фермерам выращивать более крупный урожай с теми же размерами полей. Это самая значительная возможность, которую наука о данных предоставит сельскому хозяйству. Данные для обработки и анализа могут быть собраны из двух основных источников (но не ограничиваясь ими). Первым источником является беспроводная сеть датчиков почвы (см. Рисунок). Специальная базовая станция автоматически собирает данные с этих датчиков и отправляет их на сервер для хранения и обработки. Второй источник - беспилотный летательный аппарат (беспилотник) для воздушной съемки специальной камерой (данные изображения можно обрабатывать и анализировать). Стоит отметить, что эти данные также можно получить с помощью спутников (спутниковых снимков) [1].

Анализ данных в сельском хозяйстве с использованием беспилотников предоставляет дополнительные возможности, такие как:

- создание электронных карт полей;
- проведение оперативного мониторинга состояния посевов;
- оценка всхожести культур;
- прогноз урожайности;
- проведение экологического мониторинга сельскохозяйственных земель.

Во время исследования в Университете штата Айова ученые использовали машинное обучение, чтобы отсортировать варианты семян, сохраненные в банках генов во всем мире и определить, который будет самым полезным для разводчиков, пытающихся произвести превосходящие версии. Прогнозы урожая, сгенерированные анализом данных, были на 76 процентов точны в исследовании, где ученые начали с 962 записей в базе данных и сузили их до 200 с помощью Data Sceince.

2. Аналитика данных в сельском хозяйстве обеспечивает лучшие оценки степени риска. Дата центры и методы аналитики

могут сделать фермеров более осведомленными в вопросе оценки степени риска.

Бразильская компания под названием AgroTools лидирует в этом вопросе, и показывает потенциальные возможности. Компания - клиент платформы Google Cloud и специализируется на обработке огромных баз данных спутниковых снимков для мониторинга имущественного распространения на участке земли, которая является столь же большим как Италия, Дания и Франция вместе. Одна из целей этих ежедневных проверок состоит в том, чтобы подтвердить, что производители сырья в системе поставок зависят от стабильных источников и продуктов.

AgroTools ежегодно выполняет больше чем 200 000 операций анализа данных на 1,151 слоях информации. Из-за технологий, таких как беспилотники и гаджетов Интернета вещей (IoT), дело идёт к коллекции огромной информации, суммарный объем данных AgroTools мог достигнуть 1,600 петабайтов в течение последних нескольких лет.

Интенсивные требования к данным компании сделали Google Cloud умным выбором удовлетворить её потребности. Особенно, если AgroTools нужно масштабироваться в зависимости от увеличения информационного потока или других тенденций.

3. Большие данные датчиков IoT могут изменить уход за домашним скотом. Болезни животных могут быстро распространяться в стаде из сотен или тысяч коров. Во многих случаях болезни, вызванные несколькими коровами, передаются десяткам других, прежде чем фермеры осознают проблему. К счастью, существует несколько IoT-гаджетов для предотвращения этой и других проблем.

Некоторые из них следят за фертильностью, что может быть особенно выгодно на объектах, где фермеры сильно зависят от успешного разведения животных. Другие уведомляют фермеров, когда коровы находятся в периодах высокой продуктивности молока. Исходя из того, что показывают данные, работники фермы могут делать такие вещи, как добавление типа зерна, которое способствует лактации, в режим кормления животных.

Датчики также собирают данные о поведенческих отклонениях. Поскольку эти различия могут быть первым признаком тяжелой болезни, информация помогает фермерам

активно бороться с возможными проблемами со здоровьем, изолируя коров, которые могут быть больны. Поскольку эти датчики обычно получают данные непрерывно и могут использоваться на фермах с тысячами коров, легко понять, почему центры обработки данных помогают специалистам сельского хозяйства собирать и получать информацию.

4. Аналитика данных может улучшить управление сельскохозяйственной цепочкой поставок. Сельскохозяйственные цепочки поставок представляют ряд проблем для фермеров и дистрибьюторов. В отличие от большинства товаров, пищевые продукты являются скоропортящимися и могут представлять опасность для здоровья, если с ними неправильно обращаться во время транспортировки.

Аналитика данных может значительно улучшить способ доставки этих продуктов с фермерских полей на рынки по всему миру. Дистрибьюторы смогут выявить неэффективность своих цепочек поставок, чтобы помочь сельскохозяйственной продукции быстрее и с меньшими затратами добраться до места назначения. Розничные продавцы могут использовать данные о продажах и запасах, а также информацию, которую они собрали о поведении покупателей, чтобы минимизировать потери и избыточные запасы, оставаясь на шаг впереди потребностей рынка.

Большие данные могут раскрыть потенциал городского сельского хозяйства. Городское сельское хозяйство, как с точки зрения людей, выращивающих продукты питания на приусадебных участках или в садах на крыше, стало популярной тенденцией во всем мире в последние десятилетия. Несмотря на то, что не следует сбрасывать со счетов возможность поставлять выращенные на месте культуры городскому населению, у городского сельского хозяйства есть много преимуществ, которые выходят за рамки простого обеспечения продовольствием местных общин. Это оказывает положительное воздействие на окружающую среду с точки зрения уменьшения стока от сильных дождевых осадков и улучшения качества воздуха. Работа, которая должна быть направлена на посадку и поддержание этих участков, также помогает укрепить социальные связи общества.

Воспользовавшись анализом больших данных, городские фермеры могут повысить эффективность и максимально

использовать потенциал доступного для них ограниченного пространства. Некоторые исследователи полагают, что в результате этих усилий в конечном итоге может быть получено до 180 миллионов метрических тонн продовольствия в год, что составляет около десяти процентов мирового производства бобовых, корнеплодов и клубней, а также овощных культур.

5. Сельское хозяйство и сельское хозяйство также могут повысить эффективность центров обработки данных.

Рассмотрим пример компании Apple. Когда Apple начала планировать дата-центр в Дании, она хотела творчески использовать биологические отходы близлежащих ферм. Компания заключила партнерское соглашение с Орхусским университетом для заключения многолетнего соглашения, целью которого было выяснить способы преобразования биогаза, изготовленного из соломы и навоза, в электричество. Одна из предложенных идей заключалась в подаче сельскохозяйственных отходов в варочный котел для производства метана. Это решение, как сообщается, обеспечит питание центра обработки данных, а также превратит часть отходов в удобрения, которые фермеры могут использовать. В другом случае дата-центр, управляемый Google на Тайване, надеется рассчитывать на то, что местные специалисты по аквакультуре рассчитывают на то, что они позволят ему устанавливать солнечные панели на опорах внутри рыбоводных прудов. Аквакультура тесно связана с сельским хозяйством, и поскольку эта договоренность направлена на максимизацию эффективности землепользования при соблюдении местной экологии, будущие соглашения, подобные этому, могут также принести пользу сельскому хозяйству. Оба этих случая показывают, как представители центров обработки данных могут сотрудничать с фермерами для определения путей достижения экологически чистых энергетических целей [7].

В то время как аналитика больших данных и сельское хозяйство могут выглядеть так, как будто они принадлежат разным векам, последние тенденции показывают, что они могут успешно функционировать в едином направлении. Поскольку изменение климата продолжает оказывать влияние на мировые поставки продовольствия, фермеры смогут использовать технологии для преодоления этих проблем. Принимая во внимание, что сельское

хозяйство исторически извлекало выгоду из технологических инноваций, принятие анализа больших данных однажды может рассматриваться как следствие механизации сельского хозяйства [5].

Одной из основных проблем внедрения методов науки о данных в сельском хозяйстве является высокая стоимость оборудования для сбора данных. Внедрение новых технологий часто сопряжено с определенным риском. Внедрение модифицированных сельскохозяйственных машин и беспроводных сенсорных систем требует значительных финансовых вложений. Кроме того, важно понимать срок окупаемости инвестиций. Вторая проблема - законодательство. Политическая воля часто необходима для внедрения технологий в сельскохозяйственной отрасли на государственном уровне. В то же время решение этого вопроса является интересом государства. Знание реальных площадей и структуры сельскохозяйственных угодий позволит узнать, сколько использованных и заброшенных земель в стране, и выработать политику по их эффективному использованию. Кроме того, должны быть четко определены критерии нецелевого использования земель сельскохозяйственного назначения и нарушения земельного законодательства. Дополнительным аспектом являются юридические ограничения полетов дронов. Вождение дрона требует некоторой подготовки. В некоторых странах беспилотники должны быть зарегистрированы, и специальная лицензия должна быть получена. Основными проблемами использования дронов являются безопасность, защита собранной информации и нюансы страхования.

Третья проблема - нехватка специалистов по аграрным технологиям, способных внедрять новые технологии в сельском хозяйстве. Задача совершенствования образовательных программ не менее важна. Программы должны быть направлены не только на приобретение знаний, но и на приобретение профессиональных навыков. Самая большая проблема для использования науки о данных в сельском хозяйстве – это качество данных, качество методов обработки и формы представления результатов. Чтобы решить эту проблему, отрасли необходимо создать более совершенные датчики и камеры, а также разработать высокоавтоматизированные машины и интуитивно понятное

программное обеспечение, которое требует минимального обучения для работы с ним [4].

5.2 Применение Data Science в агропромышленном комплексе

Мировая популяция, как ожидают, достигнет 9,3 миллиарда человек к 2050 г., сейчас же она составляет 7,3 миллиарда. Организация Объединенных Наций (ООН) по вопросам продовольствия и сельского хозяйства (ФАО) предсказывает рост сельского хозяйства, которое будет увеличено на 70% для обслуживания запланированного спроса. Существует большая потребность увеличить производство зерновых культур с ограниченными имеющимися ресурсами, такими как земля, вода и удобрения для удовлетворения потребностей этой увеличивающейся популяции.

Интеллектуальное сельское хозяйство является одним из решений для удовлетворения растущего спроса на продовольствие при одновременном соблюдении требований устойчивого развития. В умном сельском хозяйстве возрастает роль информации. Информация о погодных условиях, почвах, болезнях, насекомых, семенах, удобрениях и др. представляет собой важный вклад в экономическое и устойчивое развитие этого сектора. Интеллектуальное управление состоит из сбора, передачи, выбора и анализа данных. Поскольку объем сельскохозяйственных данных значительно увеличивается, необходимы надежные аналитические методы, способные обрабатывать и анализировать большие объемы данных для получения более надежной информации и гораздо более точных прогнозов [3].

Сельское хозяйство, как определено Лансом Донни, развивалось от мелкомасштабной трудоемкой стадии (до 1920-х) к промышленной стадии (с 1920 до 2010) с улучшением тяжелого машиностроения и семеноведения, и это прокладывает свой путь к третьей стадии или «АПК 3.0», которое подразумевает систему принятия решений на основе данных, полученных из внешних источников. Ланс Донни утверждает, что эта система в сельском хозяйстве обеспечивает более высокую производительность, устойчивость методов и даже помогает предоставить прозрачность потребителям, желающим знать больше об их еде.

Одним из актуальных трендов является использование науки данных (Data science). Данный факт подтверждается передовым зарубежным опытом. К примеру, наука о данных и большие данные стали бумом в последние 5 лет в Китае. Дистанционное зондирование – это одна из самых важных областей, использующих большие данные. Сельское хозяйство является одной из наиболее важных и популярных областей применения дистанционного зондирования и науки о данных. За последнее десятилетие в Китае и во всем мире произошли быстрые изменения в области сельскохозяйственного дистанционного зондирования и науки о данных. Достигнут существенный прогресс в области количественной инверсии сельскохозяйственных культур и экологических параметров с помощью дистанционного зондирования. В качестве примеров применения дистанционного зондирования в земледелии широко представлены классификация и картирование сельскохозяйственных культур, мониторинг роста сельскохозяйственных культур и оценка урожайности сельскохозяйственных культур. В Китае также широко применяется система мониторинга сельского хозяйства на основе дистанционного зондирования.

Другой пример зарубежного опыта – Вьетнам. В молочном производстве наука о данных используется для регулирования качества производства молока у коров, где каждая корова помечена с чипом RFID. Процесс доения автоматизирован с датчиками в вымени, которые могут обнаружить воспаления в молочных железах коровы. Если воспаление будет обнаружено, машина остановит процесс доения, и корова будет отмечена и проверена. В AfiMilk подобный чип присоединен к ногам каждой козы, который отслеживает ее движение. Если коза не двигается в течение длительного периода, или если она покажет неоднозначные показания качества сна, она будет проверена на болезнь.

Наука о данных изменяет способ, которым фермеры и сельскохозяйственные производители принимают решения. Современная технология позволила собрать данные почвы, воды и полезных ископаемых от ферм, и сохранить их в централизованной системе, обычно известной как Интернет вещей (IoT). IoT обращается к идее подключить устройства, передающие данные к Интернету так, чтобы они могли разделить и обмениваться

данными независимо. Такие данные могут быть объединены с данными из внешних источников, таких как спутники, метеорологические станции и даже данные соседних ферм для формирования большего объема. Аналитика данных может использоваться в накопленной большой части для получения информации, которая может использоваться фермерами для оптимизации их сельского хозяйства. Фермеры могут таким образом принять управленческие решения в области сельского хозяйства с помощью получаемой информации по всему производственному циклу: от планирования, посадки, сбора урожая до продвижения его на рынок.

Data Science базируется на двух столпах (Рисунок 5.1).

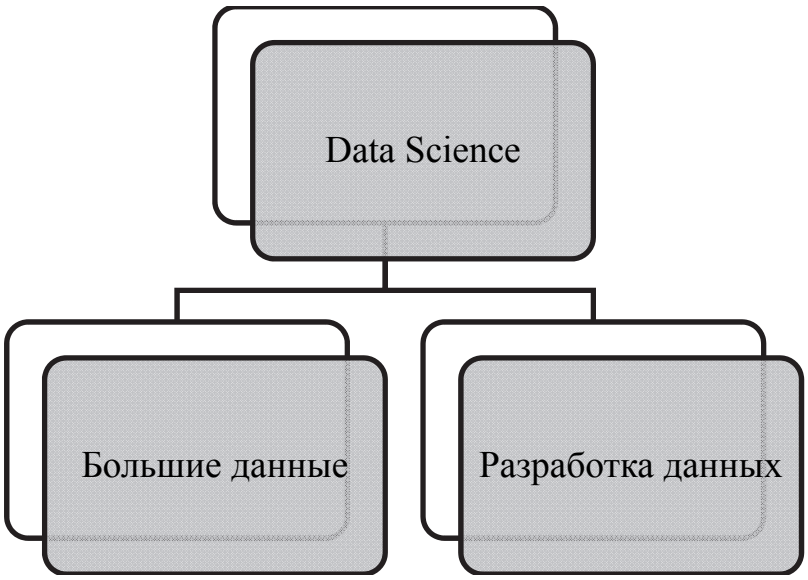


Рисунок 5.1 – Архитектоника Data Science

1. Первый столп состоит из данных, где особое внимание уделяется большим данным, хотя Data Science также имеет потенциал для получения новых представлений из наборов данных, которые не всегда соответствуют трем V.

2. Второй компонент – это процесс разработки, состоящий из цикла этапов, необходимых для перехода от исходных данных и получения знаний. Что касается процессов, то за основу фазы цикла больших данных используются: сбор данных, анализ данных, интеграция и слияние данных, моделирование и визуализация.

Стоит отметить, что в первый столп Data Science в области сельского хозяйства формируется не только в рамках получения информации в рамках применения нанотехнологий, датчиков,

систем зондирования и т.п., но и при проведении статистического наблюдения, в частности статистических переписей.

Элементы второго столпа в области сельского хозяйства более подробно представлены на рисунке 5.2.

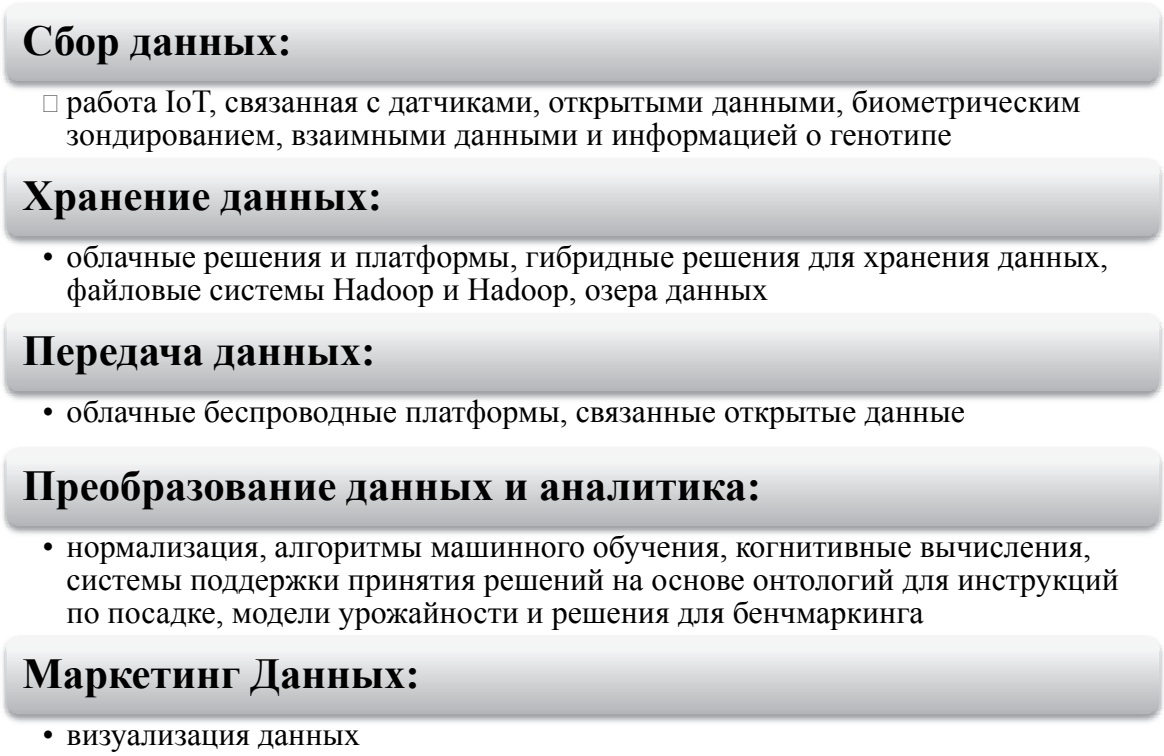


Рисунок 5.2 – Структура Data Science в агропромышленном комплексе

Приведем некоторые примеры использования Data Science в агропромышленном комплексе. В частности, использование методов науки о данных может применяться фермерами для помощи им в принятии сложных управленческих решений по следующим вопросам: что они собираются вырастить, где и когда; как выбрать ирригацию, удобрению и пестициды, которые будут использоваться. Фермеры могут собрать данные из других источников и воспользоваться аналитикой данных для получения понимания по будущим решениям. Данные, полученные при помощи датчиков в сельском хозяйстве, таких как датчик питательных веществ почвы, влагосодержания, воздухопроницаемость и т.д. (также называемые как локализованные данные), могут использоваться отдельно, или с данными из внешних источников как температура и ливень для получения различных типов информации [2].

Кроме того, доступные данные от новых технологий, таких как спектроскопы, могут использоваться в фермах для идентификации качества почвы и готовности и качества культур. Принцип спектроскопии передает различные длины волны света через объект так, чтобы различные свойства, такие как температура, масса, яркость и состав могли быть получены. Так, данные о спектроскопии могут использоваться для автоматизации некоторых процессов в сельском хозяйстве. Кроме того, данные могут использоваться в моделях для прогнозирования объемов производства.

Другой пример, возможности применения методов Data Science для налаживания отношений между сельскохозяйственными товаропроизводителями и потребителя продукции. В частности, данные могут позволить потребителям определить молекулярный состав потребляемых продуктов. Данные из датчиков и спектроскопов, обнаруживающих химический состав в воздухе, почве и зерновых культурах, могут быть использованы потребителями, применившими к ним интеллектуальный анализ. Это поможет построить доверительные отношения среди обеспокоенных клиентов и увеличить потребительскую приверженность к определенного рода продуктам [6].

Одна из главных проблем, препятствующая внедрению науки данных, – это то, что сельхозтоваропроизводители не всегда приветствует подобного рода изменения. Фермеры очень скептически относятся к изменению их методов ведения сельского хозяйства, потому что это является очень дорогостоящим и несет определенные риски. Переход к цифровым методам сельского хозяйства также требует больших инвестиций, которые могут обеспечить только крупные аграрные производители. Крупные компании могут получить субсидии относительно более быстро, по сравнению с мелкими фермерами. ЛПХ и небольшие фермеры не всегда в состоянии внедрить цифровое сельское хозяйство, в ряде случаев, они могут быть неспособны к работе с данными по причине низкой квалификации. Таким образом, еще одна проблема внедрения таких систем – то, что они будут использоваться только крупными предприятиями, что существенно влияет на конкурентоспособность на рынке.

Вторая часть проблем относится к любым типам систем, основанных на данных: ее сбор, очистка, хранение и ее распределение через безопасные каналы. Проблема с данными в сельском хозяйстве состоит в том, что они поступают в различных формах и временных интервалах. Например, данные с датчика концентрации водородных ионов почвы могут быть взяты в любой момент времени, но такие данные как проверка зрелости фруктов, которая должна быть собрана вручную, не могут быть получены так легко. Таким образом, необходимо преобразовать такие данные и соединить их так, чтобы они соответствовали определенной структуре данных. Данные должны быть оценены по совместимости, возможности многократного использования, полноценности, применимости, уместности и эффективности так, чтобы это могло использоваться в аналитике. Фермеры должны также делиться своими локальными данными с другими фермерами так, чтобы все они могли использовать эти данные для аналитики. Так же данные должны быть разделены с доверенными людьми, или же они могут ввести недобросовестную конкуренцию.

Цифровое преобразование в сельском хозяйстве вызвало разновидность инноваций в существующем мире. Одной из таких является MyCrop, интеллектуальная и самообучающаяся система реального времени, работающая с данными местоположения каждого фермерского хозяйства, погодой и данными об урожае. Система использует большие данные, машинное обучение и смартфоны, чтобы предоставлять информацию, экспертные знания и ресурсы мелким фермерам.

Движение мира к цифровому сельскому хозяйству сопровождается получением значительных инвестиций. Многочисленные научные исследования продолжают максимизировать эффективность ферм. Слияние новых технологий позволит получить маленьким фермам гораздо более крупные урожаи и их производительность сравниться с крупными предприятиями. Новые тенденции в сельском хозяйстве с использованием науки о данных и аналитики коренным образом изменят производительность сельского хозяйства, а само производство товаров стонет гораздо стабильнее и качественнее.

5.3 Обоснование выбора технических средств авторской информационной системы для АПК

Управление и анализ крупномасштабных наборов данных обычно связываются с термином Big Data (Большие данные). Согласно Begoli и Horey, Big Data – практика сбора и обработки больших наборов данных, систем, и алгоритмы, предназначенные для анализа этих крупных наборов данных. Большие данные относятся к большим разнородным объемам автономных источников с распределенным и децентрализованным управлением, которые исследуют данные и определяют отношения между этими данными. Из-за распределенной обработки, включающей много узлов, необходимо, чтобы управление данными имело высокую отказоустойчивость [10].

В частности, следует сравнивать различные технологии Больших данных и платформы аналитики согласно следующим категориям: обработка (streaming и batch), хранилище данных, интеграция данных, аналитика, управление и мониторинг. Однако они не обращаются к большой аналитике платформы данных. Как правило, перед выбором правильной технологии или платформы пользователь, или организация исследует потребности в приложениях/алгоритмах и то, что может обеспечить каждая технология/платформа.

Для исследования были выбраны главные особенности YARN/Hadoop, Spark, Flink и Hyracks/ASTERISK.

Во многих аспектах больших данных Apache Hadoop стал фактическим стандартом для выгрузки и получения доступа к данным и вычислительным ресурсам. Hadoop - масштабируемый open-source вычислительный фреймворк, который позволяет разделение процессов вычисления через многие хост-серверы, являющиеся не обязательно высокопроизводительными компьютерами. У Hadoop есть два главных компонента: приложение MapReduce и распределенная файловая система (DFS) под названием HDFS – Hadoop Distributed FileSystem. Преимущества Hadoop, главным образом, заключаются в его высокой гибкости, масштабируемости, низкая стоимость, надежность и эффективность при обработке большого объема структурированных и неструктурированных данных, а также функционал распределенных вычислений, который разбивается на

отдельные подпроцессы (job). В последних версиях Hadoop появился YARN – Yet Another Resource Negotiator, архитектура которого, по сути, представляет их себя визуальную оболочку, которая позволяет визуализировать инфраструктуру кластеров, выполнять делегации и имеет функционал запланированных вычислений [12].

Apache Spark является универсальным приложением для распределенной обработки данных. Он имеет программную модель, подобную MapReduce, но расширяет его с абстракцией совместного использования данных под названием Resilient Distributed Datasets или RDDs. Используя это расширение, Spark может получить большие возможности в оптимизации нагрузки при обработке данных при запросах SQL, стриминг, машинное обучение и обработку графов. Spark был также разработан, чтобы преодолеть дисковые ограничения ввода/вывода и улучшить работу более старых систем. Главная особенность Spark - способность выполнять вычисления в памяти. Это позволяет данным кэшироваться в памяти, таким образом исключая верхнее ограничение для повторяющихся задач на диске YARN [8].

Apache Flink является opensource приложение пакетной и потоковой структурной обработки для распределенных и высокопроизводительных приложений проекта. Он основан на философии, что много классов приложений обработки данных, включая аналитику в реальном времени, непрерывные потоки данных, историческую обработку данных и повторяющиеся алгоритмы, могут быть выражены и выполнены как потоковые отказоустойчивые пакеты данных. Flink может работать как абсолютно независимая структура поверх HDFS и YARN. Это усиливает хранение в памяти для увеличения скорости вычислений. Главные инновации Flink по сравнению с предыдущими технологиями Больших данных: распределенное время вычисления потока данных, которое использует конвейерное потоковое выполнение вычислений для потоков и пакетов, установка контрольных точек; нативная итеративная обработка; сложные семантики окон, поддержка беспорядочной обработки.

Hyracks/ASTERIX представляет собой программную платформу с многораздельными параллелями, предназначенную для выполнения ресурсоемких вычислений в больших кластерах

без совместного использования ресурсов. Hyracks включает в себя набор операторов, которые можно использовать для сборки заданий, обработки данных без необходимости написания кода Map и Reduce. Кроме того, он также предоставляет Yarn-совместимый слой для запуска существующих заданий MapReduce. Hyracks представляет собой масштабируемую систему управления информацией, которая поддерживает хранение, запросы и анализ больших коллекций полуструктурированных вложенных объектов данных. Hyracks обеспечивает повышение производительности по сравнению с MapReduce благодаря более гибкой пользовательской модели, а также является более эффективной реализацией, чем Hadoop для заданий MapReduce, для различных случаев использования данных. Hyracks также увеличивает производительность устранения неисправностей после Hadoop, используя менее пессимистичный подход к обработке ошибок [9].

Компании, используя технологии Больших данных, обычно сталкиваются с такими проблемами как: хранение разнородных источников как структурированых, неструктурированных и полуструктурированные данных; потребность получить базу знаний от больших и разнородных наборов данных, не только применяя вопросы SQL, но также и выполняя сложные алгоритмы машинного обучения или графовых вычислений; непрерывное получение потоков данных, которые должны непрерывно обрабатываться в порядке миллисекунд для аналитики в (почти) реальном времени.

1. Пакетная (Batch) обработка данных. Этот вид обработки данных тесно связан с долгосрочным вычислением на большом объеме данных, всех сразу, за цикл. Обычно он выполняется в задачах ETL (Extract, Transform and Load - извлечение, преобразование и загрузка), агрегирование данных, обучение и обновление моделей машинного обучения. Hadoop был широко распространен в пакетной обработке благодаря своей реализации MapReduce для распределения обработки данных в вычислительном кластере со многими узлами. Hyracks также выполняет пакетную обработку данных. Тем не менее, Spark стал основным принятым механизмом для обработки больших данных различными компаниями, поскольку он обеспечивает быструю

обработку данных в памяти, которая преодолевает издержки чтения и записи Hadoop.

2. Потоковая (Streaming) обработка. При потоковой обработке данные обрабатываются, а результаты формируются строго в течение определенных временных ограничений (часто порядка миллисекунд, а иногда и микросекунд в зависимости от приложения и требований пользователя). Например, Spark Streaming получает потоки входных данных в реальном времени и делит данные на микропакеты, которые обрабатываются механизмом Spark и используются для генерации окончательного потока результатов в пакетах. Микропакетирование позволяет обрабатывать поток как последовательность небольших пакетов или порций данных. Однако это может привести к значительным дополнительным затруднениям в виде задач планирования. С другой стороны, Flink может предоставить все преимущества буферизации без каких-либо затрат на планирование задач. Flink также может хорошо работать в сценариях в реальном времени или почти в реальном времени, когда данные из кластера должны быть доступны практически в один и тот же момент их генерации.

3. Универсальное хранилище. HDFS может хранить разнообразное сочетание структурированных, неструктурированных и полуструктурированных данных. Hyracks может использовать данные из HDFS, а также предоставляет хранилище данных AsterixDB для приема, хранения, индексирования, запроса и анализа больших объемов данных с использованием гибкой модели данных (ADM). Spark поддерживает интеграцию с широким спектром файловых систем, включая HDFS, файловую систему MapR, Cassandra, Amazon S3 или реализацию специального решения. Flink позволяет интегрировать гетерогенные наборы данных, начиная от строго структурных реляционных данных, неструктурированных текстовых данных и полуструктурированных данных. Он также работает с HDFS и подключается к различным другим системам хранения данных. Flink и Spark не предоставляют первичного решения для хранения.

4. Аналитика данных. YARN / Hadoop активно поддерживает несколько проектов верхнего уровня для создания инструментов разработки и управления потоком данных и их обработки, таких

как Giraph, Pig, Hive, Mahout и HBase. Spark также поддерживает широкий спектр приложений, включая ETL, машинное обучение (MLib), потоковую обработку (Spark Streaming) и вычисления графов (GraphX). Стек Flink предлагает библиотеки с высокоуровневыми API для различных вариантов использования: обработка сложных событий (CEP), машинное обучение (FlinkML) и Graph Analytics (Gelly). Программный стек системы Hyracks также состоит из различных интерфейсов для аналитики, таких как SQL (Hivesterix), XQuery (Apache VXQuery) и Graph (Pregelix).

Даже Hyracks может эффективно выполнять сложные распределенные операции потока данных и выражать полные реляционные алгебры. Он также предоставляет API-интерфейсы низкого уровня и требует, чтобы эксперт по машинному обучению сформировал свои алгоритмы в качестве операторов потока данных [11].

В таблице 5.1 указана суммарная итоговая информация, описанная выше. Она дает краткое сравнение возможностей технологий больших данных.

Таблица 5.1 – Сравнение технологий Big Data

Категория \Технология	**Hadoop**	**Spark**	**Flink**	**Hyracks**
Тип обработки	Batch	Mini-batch	Streaming, Batch	Batch
Хранилище	HDFS	-	-	AsterixDB
Анализ данных	SQL,ML, Graph	ETL,ML, Graph	ML,CEP, Graph	SQL, XQuery, Graph

5. Платформы больших данных. Платформа больших данных представляет собой экосистему услуг и технологий, которая должна выполнять анализ объемных, сложных и динамических данных. Таким образом, расширение аппаратной платформы становится неизбежным, и выбор правильных аппаратных / программных технологий становится решающим решением, если требования пользователя должны быть удовлетворены в разумные сроки. Недавно появился набор платформ Big Data, в том числе Big Data Europe (BDE), Hortonworks 2 и Cloudera3.

5.1. Платформа BDE разработала вычислительную инфраструктуру для обработки больших объемов данных в различных форматах. Она отвечает требованиям упрощения использования, упрощения развертывания, управления

неоднородностью и улучшения масштабируемости, а также облегчает выполнение и интеграцию сред и данных Big Data, таких как Hadoop, Spark, Flink и многих других. Авторы решили использовать Docker в качестве методологии упаковки и развертывания, а также эффективного управления разнообразными аппаратными ресурсами наряду с меняющимися требованиями к программному обеспечению. BDE позволяет выполнять различные задачи потока больших данных, такие как передача сообщений (через Kafka, Flume), хранение (через Hive, Cassandra), анализ (через Spark, Flink) или публикация (через GeoTriples). Кроме того, платформа с открытым исходным кодом и полностью бесплатна.

5.2. Hortonworks Data Platform (HDP) — это современная архитектура данных с открытым исходным кодом, которая обеспечивает немедленную выгоду за счет сокращения затрат на хранение, поскольку она интегрирует YARN в свой центр обработки данных, и за счет оптимизации затрат на Enterprise Data Warehouse за счет разгрузки недорогих вычислительных задач, таких как ETL на YARN. YARN позволяет HDP интегрировать все механизмы обработки данных в сообществе и коммерческой экосистеме для предоставления согласованных общих услуг и ресурсов на платформе. Ambari — это интуитивно понятный веб-интерфейс и надежный REST API, который делает управление HDP более простым, последовательным и безопасным. Кроме того, HDP — это комплексное решение, предлагающее не только обработку данных и управление, но и возможности предприятия, соответствующие требованиям предприятия, охватывающим вопросы безопасности, управления, и операции.

5.3. Cloudera была первой компанией, которая разработала и распространила программное обеспечение на основе Apache Hadoop и сделала анализ данных на больших данных более удобным и доступным для всех, кто заинтересован. Он интегрирует Hadoop с более чем десятком других важных проектов с открытым исходным кодом. Cloudera создала функционально продвинутую систему, которая помогает выполнять сквозные рабочие процессы больших данных. Различные проекты составляют экосистему Cloudera для различных задач больших данных: потоковая обработка (через Spark), передача сообщений (через Kafka, Flume), хранение (через Accumulo, Hive, Pig, Hbase), анализ (через Flink, Impala), поиск (через Cloudera Search) или предоставление

расширяемого и производительного веб-интерфейса для пользователей (через HUE).

Ряд повторяющихся проблем, с которыми обычно сталкиваются организации, которые могут стать более сложными при работе с большими данными:

– интегрировать различные источники больших данных и предоставлять прозрачное представление пользователям;

– управлять и защищать активы данных организации, чтобы гарантировать в целом понятные, правильные, полные и защищенные корпоративные данные;

– осуществлять мониторинг данных, ресурсов и приложений для анализа и оценки работоспособности и производительности всей системы.

Каждую проблему можно объединить в одну из следующих категорий проблем: интеграция данных, управление данными и службы мониторинга.

1. Интеграция данных включает в себя объединение данных из разных источников и предоставление пользователям единого представления о них. HDP сотрудничает с Talend, мощным и универсальным решением с открытым исходным кодом для интеграции больших данных, которое изначально поддерживает Hadoop, включая коннекторы для HDFS, Hbase, Pig, Sqoop и Hive, без необходимости писать код. Talend также поддерживает Cloudera Navigator. Другой альтернативой для HDP является Oracle Data Integrator (ODI). Пользователь может создать поток от источников к целям различных технологий, включая реляционные базы данных, приложения, таблицы XML, JSON, Hive, файлы Hbase, HDFS и т. Д. Платформа BDE идет дальше, чем HDP и Cloudera, и включает в себя Semantic Data Lake – репозиторий, предназначенный для обработки и анализа наборов данных в их исходных форматах – под названием Ontario. Онтарио создает семантический слой поверх озера данных, который отвечает за отображение данных в существующие семантические словари / онтологии. Успешный процесс картирования, называемый семантическим лифтингом, обеспечивает просмотр всей информации. Таким образом, данные можно извлекать, запрашивать или анализировать из разнородных источников в озере, как если бы они были в едином формате с использованием языка запросов высокого уровня. Другим важным компонентом

является Semagrow, система обработки запросов SPARQL, которая объединяет несколько удаленных конечных точек.

2. Управление данными – это система прав принятия решений и ответственности за процессы, связанные с информацией, которые выполняются в соответствии с согласованными моделями, которые описывают, кто может выполнять какие действия с какой информацией и когда, при каких обстоятельствах, с использованием каких методов расширяет это определение путем включения политик, касающихся оптимизация, конфиденциальность и монетизация больших данных. Управление системами больших данных может быть сложным. Последовательная защита наборов данных в нескольких репозиториях может быть чрезвычайно подвержена ошибкам. Компонент Cloudera Navigator Data Management – это полностью интегрированный инструмент управления данными и безопасности для Hadoop, который был разработан для удовлетворения требований глобальных предприятий по обеспечению соответствия, управления данными и аудита. HDP использует Apache Atlas и Apache Ranger, которые сочетают классификацию данных с применением политики безопасности. Apache Atlas был создан как часть инициативы Hadoop Data Governance, и он дает возможность просматривать межкомпонентную линию, обеспечивая полный обзор перемещения данных через некоторые механизмы синтаксического анализа, такие как Apache Storm, Kafka, Falcon и Hive. Apache Ranger обеспечивает централизованное управление безопасностью для Hadoop. Интегрируя Atlas и Ranger, HDP позволяет компаниям внедрять динамические политики доступа во время выполнения, которые активно предотвращают нарушения. BDE не слишком углубляется в управление данными, поскольку не занимается такими вопросами, как конфиденциальность данных, обмен ими и права.

Таблица 5.2 – Сравнение платформ Big Data

Категория \Платформа	HDP	BDE	Cloudera
Интеграция данных	Talend, ODI	Ontario, Semagrow	Talend
Управление данными	Atlas, Ranger	No support	Cloudera Navigator
Мониторинг	Ambari	Promethues, ELK Stack	Cloudera Manager

3. Мониторинг — это процесс предварительного анализа и оценки того, что было отслежено (данные, ресурсы или приложения). Программное обеспечение для мониторинга помогает измерять и отслеживать данные, обычно используя информационные панели, оповещения и отчеты. Cloudera Manager предоставляет множество функций для мониторинга работоспособности и производительности компонентов кластера (хостов, демонов служб), а также требований к производительности и ресурсам для заданий, выполняемых в кластерах. BDE различает мониторинг ресурсов и мониторинг состояния. Первый позволяет отслеживать работоспособность сервера или компонента в платформе (использование ЦП, использование памяти, сетевой ввод-вывод и использование диска), а второй дает представление о состоянии конкретного приложения. Для мониторинга ресурсов на платформе BDE могут быть полезны инструменты Docker Stats, cAdvisor, Prometheus, InfluxDB и Grafana. Для мониторинга состояния BDE поддерживает встроенную запись в докер и стек ELK. Как часть HDP, Apache Ambari позволяет планировать, устанавливать и безопасно настраивать кластеры компьютеров, упрощая текущее обслуживание и управление кластерами.

В таблице 5.2 представлено краткое сравнение всех платформ, представленных выше.

5.4 Описание и архитектура информационной системы

Разрабатываемая авторами информационная система должна обеспечивать полноценный функционал работы с большими данными ВСХП, а именно – скорость работы с данными, масштабируемость, возможность анализа и поиска данных в реальном времени. При этом содержание системы (границы системы) предполагает наличие следующих элементов:

- настраиваемый модуль кластеризации данных;
- модуль интеграции основного хранилища данных с внешними базами данных;
- модуль мониторинга кластеров;
- интерфейс взаимодействия с хранилищем данных для анализа;
- модуль «песочницы» для быстрого и наглядного доступа к данным;

– функционал экспорта данных в различные форматы (csv, json и т.п.).

Ниже в таблице 5.3 представлены требования к информационной системе.

Таблица 5.3 - Требования к информационной системе

№ п/п	Вид требования	Описание
1	Системные требования	Минимум 3 системы на ОС Linux Red Hat CentOS. Каждая из систем имеет минимум 16 ГБ RAM и HDD/SSD объемом 1 ТБ при этом дальнейших рост хранилища подразумевает масштабирование систем (увеличение их количества, добавление RAM и увеличение объема постоянной памяти). Так же каждая система должна иметь процессор с тактовой частотой минимум 3 ГГц и максимально возможно большим объемом кеш-памяти (влияет на скорость выполнения циклических операций расчетов).
2	Требования к удобству системы	Система должна быть максимально совместима, «дружественна», необходимо постараться максимально автоматизировать все операции в работе с системой.
3	Требования к производительности	Система должна быть с высоким показателем отказоустойчивости, иметь высокую скорость выполнения вычислительных операций и обеспечивать быструю и качественную интеграцию данных из внешних источников в основное хранилище.
4	Интерфейс (взаимодействие) системы	Для взаимодействия с системой требуется использовать один из общепринятых стандартов запросов, в том числе не исключена возможность использования документно-ориентированных баз данных для взаимодействия. Так как документно-ориентированная СУБД допускает гораздо большую вложенность и сложность структуры данных, чем остальные СУБД.
5	Требования к безопасности	В системе должна быть предусмотрена авторизация пользователей, делегация полномочии пользователей, система защиты от угроз, таких как DDOS атака, потеря данных, несанкционированный доступ, аппаратная угроза.
6	Требования к обслуживанию системы на протяжении ее жизненного цикла	Система должна предоставлять удобный функционал для обслуживания системы администраторами, обновления компонентов, мониторинг системы, масштабирование, а также резервное копирование отчетов анализа данных.

Для устранения недостатков было принято решение разработать систему по технологии Hadoop Big Data с использованием нативных фреймворков данной технологии. Система управления базами данными будет построена на фреймворке Apache Hive, он позволяет выполнять запросы,

агрегировать и анализировать данные, хранящиеся в Hadoop. Также использовано массово-параллельное приложение Apache Impala для интерактивного выполнения SQL запросов к данным кластера Hadoop.

Для распределенной обработки неструктурированных и слабоструктурированных данных применён фреймворк Apache Spark, который работает в парадигме резидентных вычислений (обработка данных в оперативной памяти) – это обеспечивает быстрый многократный доступ к данным и отлично подходит для машинного обучения. С целью обеспечения функционала планировки вычислительных задач использована система Apache Hadoop YARN, так же YARN имеет встроенный функционал мониторинга выполнения вычислений. В том числе использована система Oozie, которая предоставляет принципиальную возможность объединения нескольких последовательно выполняемых заданий в одну логическую единицу работы.

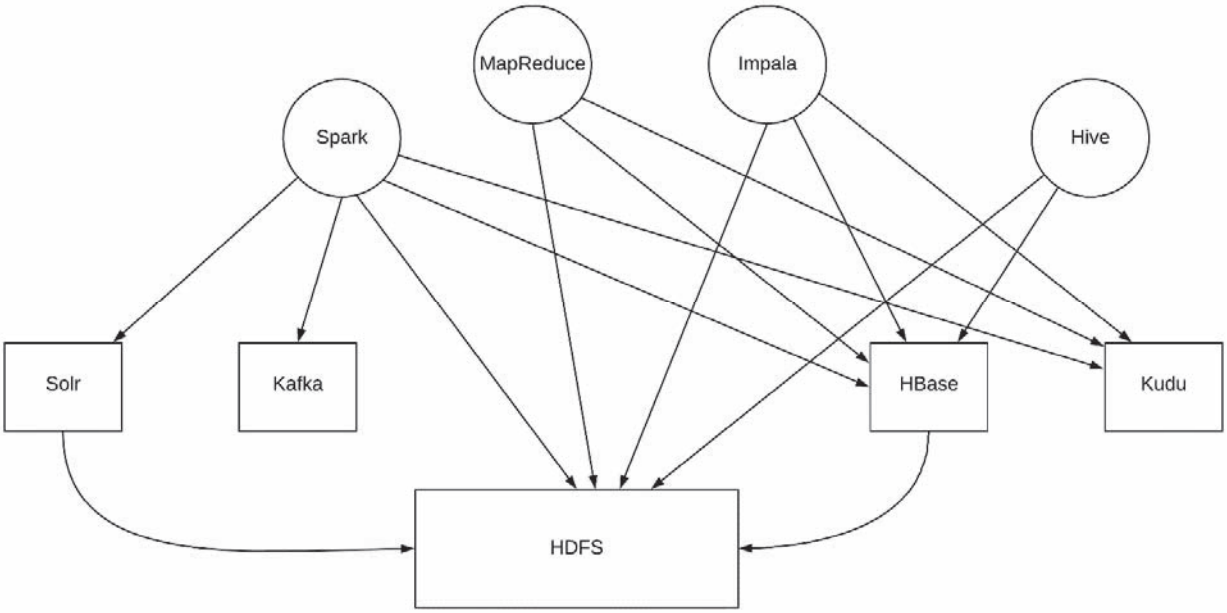


Рисунок 5.3 – Взаимодействие компонентов на передающем уровне

Для обеспечения быстрого поиска уже проанализированных данных использована поисковая платформа Solr за счет нереляционного хранения данных, что идеально для разрабатываемой ИС. Так же применена система Apache Kafka, которая позволяет делать масштабирование информационной системы и автоматизировать, и оптимизировать эту задачу. Для мониторинга системных компонентов применена служба под

названием Apache ZooKeeper, которая предоставляет информацию о конфигурации системы и данных о синхронизации, которую ZooKeeper получается через JavaAPI. Кроме того, в Hadoop применяется файловая система HDFS и для её работоспособности и обеспечения отказоустойчивости применяется технология HBase – нереляционная СУБД, которая поддерживает сжатие операций в памяти, фильтр Блума (определения принадлежности элемента к множеству).

Результат работы созданной информационной системы представим в виде контрольного примера анализа данных Всероссийской Сельскохозяйственной Переписи 2016 г. Его задача – проанализировать данные переписи по личным подсобным хозяйствам Российской Федерации (ЛПХ), а именно:

1. разделить все ЛПХ по округам Российской Федерации;
2. разделить все ЛПХ по регионам внутри каждого округа;
3. осуществить группировку ЛПХ по цели производства сельскохозяйственной продукции: 1 группа – это ЛПХ, которые производят продукцию с целью самообеспечения себя продовольствием, 2 группа – ЛПХ, использующие подсобное производство как дополнительный источник дохода, и 3 группа – ЛПХ, использующие подсобное производство как основной источник дохода;
4. каждую группу следует разбить на ещё 2 подгруппы – ЛПХ, которые реализуют сельскохозяйственную продукцию и которые не реализуют её.

География размещения ЛПХ по округам представлена на рисунке 5.4.

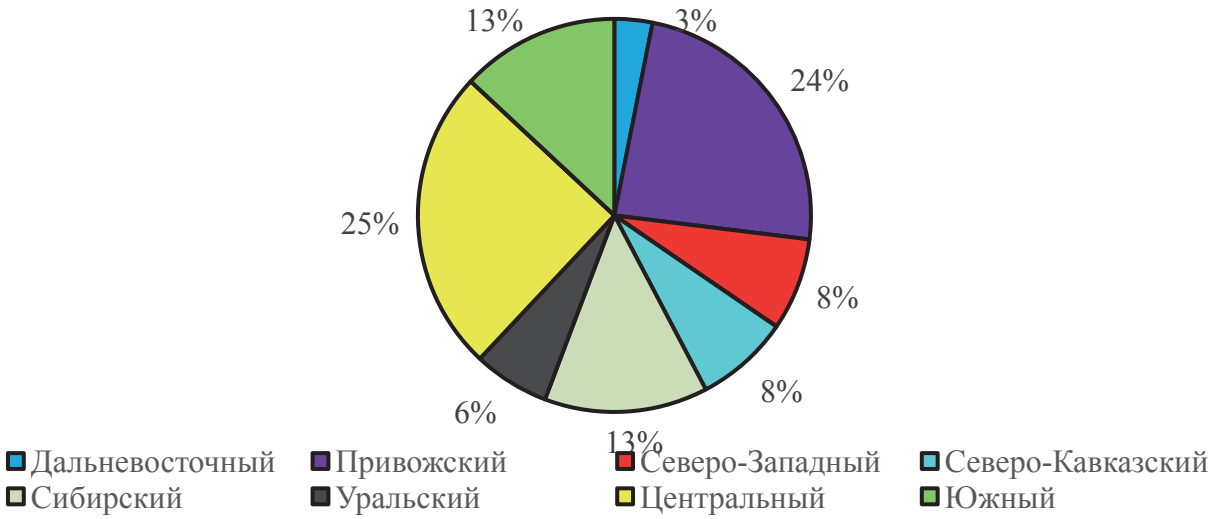


Рисунок 5.4 – География размещения ЛПХ по федеральным округам

Как видно по данным рисунка 5.4, наибольшее количество ЛПХ расположено в Центральной федеральном округе, второе место занимают Сибирский и Южный федеральные округа. При этом незначительная доля последнего может быть объяснена тем, что для благоприятных для ведения сельского хозяйства регионов, оно является основой предпринимательской деятельности и приводит к формированию предпринимательских форм регистрации бизнеса – ИП, КФХ, организации. После обработки представленных больших данных, можно сделать расчеты и провести соответствующий анализ (таблица 5.4).

Проведенный анализ позволил выявить зависимость между уровнем технической обеспеченности личных подсобных хозяйств и степенью их товарности. Так в личных подсобных хозяйствах, для которых оно является основным источником дохода и которые реализуют свою продукцию, количество тракторов, приходящихся в расчете на 1 ЛПХ, составляет 3,26, в то время как для других категорий личных подсобных хозяйств обеспеченность техникой существенно ниже. Кроме того, товарные личные подсобные хозяйства существенно крупнее по своему размеру, в частности, площадь посевов на один ЛПХ в товарных хозяйствах, реализующих продукцию, выше более чем в 10 раз по сравнению с аналогичным показателем по ЛПХ, которые ведут хозяйство чисто для самообеспечения продовольствием. Также в таких хозяйствах выше процент распаханности почвы.

Кроме того, анализ позволил выявить еще одну существенную зависимость, в частности, так как ранее неоднократно экономистами было доказано, что производство продукции животноводства является низкорентабельным видом деятельности, то основная масса крупнорогатого скота сконцентрирована в группах нетоварных личных подсобных хозяйств, в то время как товарные личные подсобные хозяйства специализируются на производстве продукции растениеводства.

Таблица 5.4 – Статистическая оценка деятельности ЛПХ России

Показатель\ Группа	Самообеспечение продовольствием		Дополнительный источник денежных средств		Основной источник денежных средств		Неопределенные	
	Реализующие	Нереализующие	Реализующие	Нереализующие	Реализующие	Нереализующие	Реализующие	Нереализующие
Количество ЛПХ в группе	2 379 164,00	9 643 679,00	117 447,00	18 351,00	12 962,00	1 960,00	1 971 409,00	4 317 495,00
Количество проживающих лиц на 1 ЛПХ, человек/ЛПХ	2,08	2,08	2,13	2,17	3,26	2,50	2,62	0,44
Общая площадь, кв. м	20 152 191 935,00	38 981 473 297,00	2 305 635 612,00	363 631 508,00	755 578 192,00	59 208 611,00	39 188 257 595,00	24 242 958 688,00
Общая площадь на одно ЛПХ, кв. м/ЛПХ	8 470,28	4 042,18	19 631,29	19 815,35	58 291,79	30 208,48	19 878,30	5 615,05
Общая площадь посевов группы, кв. м	4 558 350 950,00	7 982 718 099,00	856 618 166,00	189 600 356,00	197 070 441,00	34 306 840,00	8 271 795 629,00	347 517 021,00
Площадь посевов на один ЛПХ, кв. м/ЛПХ	1 915,95	827,77	7 293,66	10 331,88	15 203,71	17 503,49	4 195,88	80,49
Процент распаханности площади группы, %	22,62	20,48	37,15	52,14	26,08	57,94	21,11	1,43
Площадь под зерно-бобовые всего, кв. м	850 900 946,00	1 022 304 646,00	256 757 514,00	72 028 458,00	64 546 235,00	12 308 319,00	2 327 285 476,00	155 051 642,00
Площадь под картофель всего, кв. м	1 864 167 697,00	4 313 876 017,00	113 900 009,00	25 940 000,00	25 178 257,00	2 621 193,00	2 224 268 165,00	66 385 948,00
Скот крупный рогатый – всего, голов	1 630 513,00	1 814 819,00	217 224,00	14 748,00	77 922,00	3 012,00	4 204 801,00	81 082,00

Показатель\ Группа	Самообеспечение продовольствием		Дополнительный источник денежных средств		Основной источник денежных средств		Неопределенные	
	Реализующие	Нереализующие	Реализующие	Нереализующие	Реализующие	Нереализующие	Реализующие	Нереализующие
Скот молочный крупный рогатый - всего, голов	1 466 308,00	1 628 451,00	179 607,00	12 714,00	48 426,00	2 258,00	3 639 224,00	67 193,00
Скот мясной крупный рогатый - всего, голов	163 315,00	184 217,00	36 954,00	1 934,00	29 167,00	748,00	564 044,00	13 125,00
Тракторы всего, штук	119 945,00	205 883,00	6 645,00	541,00	1 510,00	90,00	190 882,00	6 814,00
Тракторов на один ЛПХ, штук/ЛПХ	0,05	0,02	0,06	0,03	0,12	0,05	0,10	0,00
Средний процент реализации: Сахарная свекла (кроме кормовой), %	0,14	-	0,08	-	0,25	-	0,14	-
Средний процент реализации: Овощи, %	16,07	-	10,46	-	17,45	-	14,37	-
Средний процент реализации: Плоды, %	11,16	-	4,41	-	4,57	-	8,40	-
Средний процент реализации: Ягоды, %	10,94	-	5,05	-	4,72	-	8,45	-

Показатель\ Группа	Самообеспечение продовольствием		Дополнительный источник денежных средств		Основной источник денежных средств		Неопределенные	
	Реализующие	Нереализующие	Реализующие	Нереализующие	Реализующие	Нереализующие	Реализующие	Нереализующие
Средний процент реализации: Виноград, %	1,07	-	0,53	-	0,48	-	0,85	-
Средний процент реализации: Скот и птица, %	5,66	-	9,72	-	26,12	-	13,39	-
Средний процент реализации: Молоко всех видов, %	5,08	-	12,49	-	27,06	-	15,27	-
Средний процент реализации: Яйцо, %	5,45	-	4,77	-	9,29	-	7,48	-
Средний процент реализации: Мед, %	0,93	-	2,13	-	4,51	-	2,64	-
Средний процент реализации: Шерсть, %	1,16	-	1,87	-	5,17	-	3,02	-
Средний процент реализации: Шкурки пушных зверей, %	0,06	-	0,08	-	0,08	-	0,21	-

Показатель\ Группа	Самообеспечение продовольствием		Дополнительный источник денежных средств		Основной источник денежных средств		Неопределенные	
	Реализующие	Нереализующие	Реализующие	Нереализующие	Реализующие	Нереализующие	Реализующие	Нереализующие
Средний процент реализации: Кормовые культуры, %	6,16	-	30,70	-	30,70	-	7,06	-
Средний процент реализации: Рассада, %	0,18	-	0,61	-	0,61	-	0,80	-

Обращает на себя внимание и тот факт, что более 6 млн. личных подсобных хозяйств отнесли себя к категориям неопределившихся, то есть они не отнесли себя ни к товарным, ни к полутоварным, ни к нетоварным личным подсобным хозяйствам. Что еще более важно, в категории товарных личных подсобных хозяйств, которые определили ЛПХ как основной источник дохода – около 2000 личных подсобных хозяйств по Российской Федерации ответили на вопрос «Реализуют ли они сельскохозяйственную продукцию?» отрицательно, что говорит о несопоставимости результатов проведенного обследования. Региональный анализ данных 2000 личных подсобных хозяйств показал, что более четверти из них (более 500) были зарегистрированы в Республике Дагестан.

По этой причине нами был проанализирован переписной лист личных подсобных хозяйств, определено, что его необходимо упростить, а также целесообразно вопросы относительно того, к какой группе относится личное подсобное хозяйство и реализует или не реализует ли оно продукцию, сделать контрольными и взаимоисключающими, что позволит повысить эффективность проведения статистического наблюдения.

В действующем варианте переписного листа вопрос «Укажите цель производства сельскохозяйственной продукции» содержится в разделе 1, вопрос же «Продаете ли Вы (включая переданную родственникам, другим людям) сельскохозяйственную продукцию?» содержится в 7 разделе переписного листа и не связан с предыдущими разделами.

Выводы. Цель данного раздела монографии заключалась в разработке автоматизированной системы анализа больших данных, сформированных по данным Всероссийской сельскохозяйственной переписи 2016 г., с использованием методов Data Science. В результате выполнения работы выявлены особенности формирования Big Data и Data Science в АПК, проведен анализ архитектур построения систем хранения и анализа больших данных, а также выполнен анализ больших данных Всероссийской сельскохозяйственной переписи. Показано, что на начальных этапах целесообразно использовать дистрибутивы для развертывания рассматриваемых систем с учетом пакетной или потоковой обработки данных. Так же был развернут графический

интерфейс для системы хранилища и обработки данных, ускоряющий в разы взаимодействие с системой и упрощающий управление компонентами информационную систему.

Анализ данных показал достоверную статистику ЛПХ, были обнаружены ошибки в сборе данных при переписи и в целом выявлены недостатки текущего способа хранения, обработки и предоставления данных переписи Росстатом. Был проанализирован переписной лист личных подсобных хозяйств, определено, что его необходимо упростить, а также целесообразно вопросы относительно того, к какой группе относится личное подсобное хозяйство и реализует или не реализует ли оно продукцию, сделать контрольными и взаимоисключающими, что позволит повысить эффективность проведения статистического наблюдения.

Исследованный комплекс технологий, алгоритмов и их реализаций на языке Python показал свою работоспособность и возможность его применения при дальнейших исследованиях в области больших данных.

Библиографический список к главе 5

1. Воронин, Е.А. Кластерные технологии и их применение в АПК / Е.А. Воронин // Вестник Федерального государственного образовательного учреждения высшего профессионального образования «Московский государственный агроинженерный университет имени В.П. Горячкина». – 2008. – с. 25. – Текст: непосредственный.

2. Цифровая экономика Российской Федерации: Информационные материалы о национальной программе. – URL: http://static.government.ru/media/files/3b1AsVA1v3VziZip5VzAY8RTcLEbdCct.pdf (дата обращения 07.07.2022). – Текст: непосредственный.

3. Кеникстул, В.И. Формирование системы регионального управления сельским хозяйством / Е.А. Воронин, В.И. Еремеев, Н.И. Жуков, В.Н. Микляева. – 2014. – 81 с. – Текст: непосредственный.

4. Размещение сельскохозяйственного производства по территории и категориям хозяйств в Российской Федерации (экономико-статистический анализ): монография / Ю.Н. Романцева.

– М.: Изд-во РГАУ-МСХА имени К.А. Тимирязева, 2010. – 172 с. – Текст: непосредственный.

5. Современные проблемы информационного, учетного и финансового обеспечения устойчивого развития АПК: монография / И.В. Харчева, А.В. Уколова, Л.В. Постникова, И.В. Макунина и др. – Изда-тельство РГАУ-МСХА, 2015. – 163 с. – Текст: непосредственный.

6. Экономика предприятия (организации) АПК: Учебник / Р.Г. Ахметов, А.В. Голубев, Р.С. Гайсин, А.Е. Шибалкин и др. – М.: Изд-во РГАУ-МСХА имени К.А. Тимирязева, 2013. – 618 с. – Текст: непосредственный.

7. Analytics Comes of Age. – URL: https://www.mckinsey.com/~/media/McKinsey/Business%20Functions/McKinsey%20Analytics/Our%20Insights /Analytics%20comes%20of%20age/Analytics-comes-of-age.ashx (дата обращения 05.07.2022). – Текст: электронный.

8. Apache HBase – Apache HBase™ Home. – URL: https://hbase.apache.org/ (дата обращения 05.07.2022). – Текст: электронный.

9. Big Data Analytics Landscape 2020. – URL: https://www.learnbigdatatools.com/big-data-analytics-landscape-2020 (дата обращения 05.07.2022). – Текст: электронный.

10. Cloudera CDH. – URL: https://www.cloudera.com/products/open-source/apache-hadoop/key-cdh-components.html (дата обращения 05.07.2022). – Текст: электронный.

11. 10 Charts That Will Change Your Perspective Of Big Data's Growth – URL: https://www.forbes.com/sites/louiscolumbus/2018/05/23/10-charts-that-will-change-your-perspective-of-big-datas-growth/#432b11a29268 (дата обращения 05.07.2022). – Текст: электронный.

12. Hadoop. – URL: https://ru.wikipedia.org/wiki/Hadoop (дата обращения 05.07.2022). – Текст: электронный.

Глава 6 Разработка модуля информационной системы распознавания образов

Последние годы в мире популярны исследования и разработка в области построения информационных систем (ИС) с использованием средств машинного обучения. Такие системы активно используются для распознавания лиц на улицах города, контроля дорожного трафика, выписка штрафов автомобилистам и т.п.

Все эти системы узкоспециализированы и разрабатываются для работы в масштабах предприятий или города. На рынке либо невозможно, либо сложно найти такую систему для частного использования.

Для учебных заведений существует ряд проблем в проведении контроля посещения и составления списка присутствующих на онлайн-экзаменах или онлайн-занятиях. Так, например, в ФГБОУ ВО РГАУ-МСХА имени К.А. Тимирязева, при проведении вступительных экзаменов в формате онлайн (webinar-конференция, к примеру), руководителю необходимо попросить каждого присутствующего показать свой документ, определяющий личность в камеру, чтобы сверить со списком, подавших заявления. Сложность заключается в том, что у абитуриента может быть плохое качество камеры, из-за чего документ является нечитаемым, что затрудняет описанную процедуру. Помимо этого, учитывая сложность, данное мероприятие занимается достаточно времени, поскольку необходимо поочередно зафиксировать каждого.

В связи с этим особую актуальность приобретает вопрос автоматизации и адаптации этого процесса посредством разработки информационной системы на базе нейронной сети, которая способна самостоятельно без показа документов идентифицировать личность присутствующих, используя базу знаний и сравнивая ее с видеопотоком.

Используемые в исследовании программные средства: Microsoft Word и Visio, StarUML, PyCharm (Python 3.6).

Теоретическая значимость разработки заключается в том, что в работе систематизированы возможные алгоритмы обработки информации нейронными сетями. Практическая значимость разработки заключается в возможности использования

разработанной информационной системы в целях проведения контроля присутствующих на занятиях, онлайн-экзаменах и прочих мероприятиях в любом аграрном вузе страны.

6.1 Построение нейронных сетей для работы с изображениями

Искусственные нейронные сети (ИНС) — это математические модели организации реальных биологических нейронных сетей (БНС). Но в отличие от математических моделей БНС, ИНС не требует точное описание всех химических и физических процессов.

Основываясь на строении нервной клетки человека, была предложена модель математического нейрона (Рисунок 6.1).

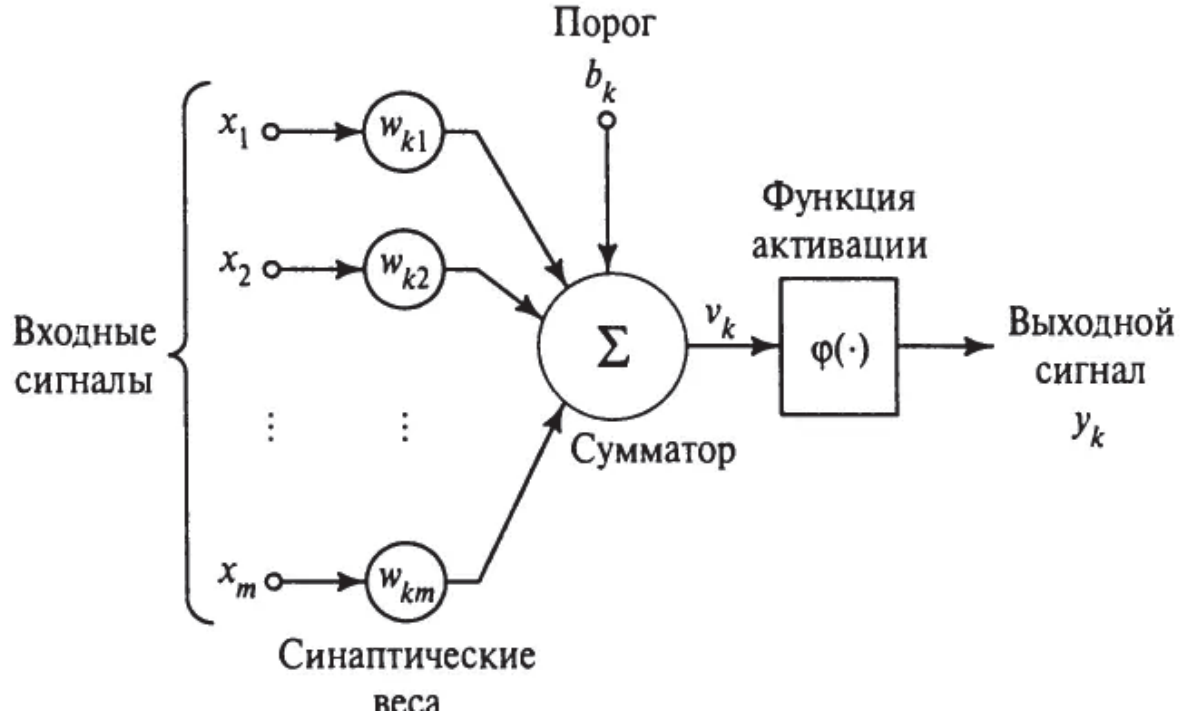


Рисунок 6.1 – Строение математического нейрона

1. X – входной вектор параметров.
2. W – вектор весов (матрица весов), представляет собой некоторое количество числовых значений, которые могут быть изменены в процессе обучения сети.
3. Сумматор – представляет собой функцию, которая суммирует все значения, входящие в нейронную сеть после умножения их на соответствующие определенные веса.
4. Функция активации нейрона – принимает определенное значение, которое принимается ввиду пришедшего результата от сумматора.

5. В конце нейроны, куда на один из множества их собственных входов подается значение с выхода данного нейрона [7].

Из этих минимальных структурных единиц собирают классические искусственные нейронные сети (Рисунок 6.2).

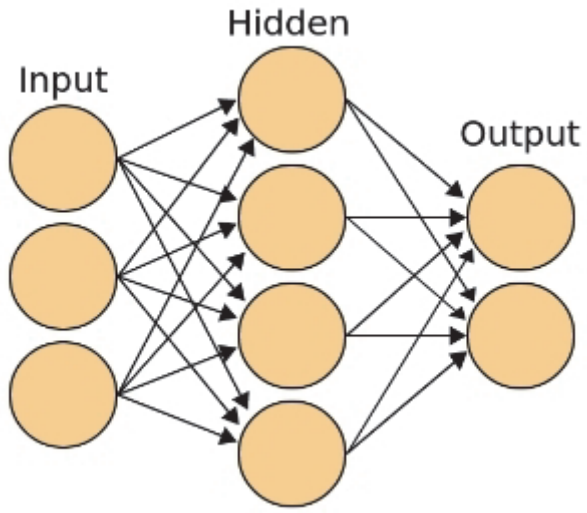


Рисунок 6.2 – Классическая топология нейросети

Определяют следующие слои нейронных сетей:

1. Входной (рецепторный) – слой, на который подается исходная информация и преобразуется в вид, потребляемый сетью.

2. Ассоциативный (скрытый) – структура, способная к запоминанию примеров, нахождению сложных корреляций и нелинейных зависимостей, к построению абстракций и обобщений. Часто ассоциативный слой в нейронной сети состоит из некоторого количества слоев разного назначения.

3. Выходной слой – это слой, каждый нейрон которого отвечает за конкретный класс. Выход этого слоя можно интерпретировать как функцию распределения вероятности принадлежности объекта разным классам [9].

Именно благодаря наличию скрытых ассоциативных слоев, искусственная нейронная сеть способна строить гипотезы, основанные на нахождении сложных зависимостей.

Например, для сверточных нейросетей, построенные для обработки изображений, на вход подаются числовые значения пикселей от 0 до 255, а на выходе стоят нейроны, каждый из

которых соответствует определенному классу классификации (кошка, собака, человек и т. д.) [3].

Для работы с картинками необходимо учитывать, что каждое изображение состоит из пикселей, которые получаются в результате наложения друг на друга 3 цветовых каналов RGB: Red (Красный), Green (Зеленый), Blue (Синий). Каждый пиксель имеет значения в зависимости от оттенка цвета (R, G, B) (Рисунок 6.3).

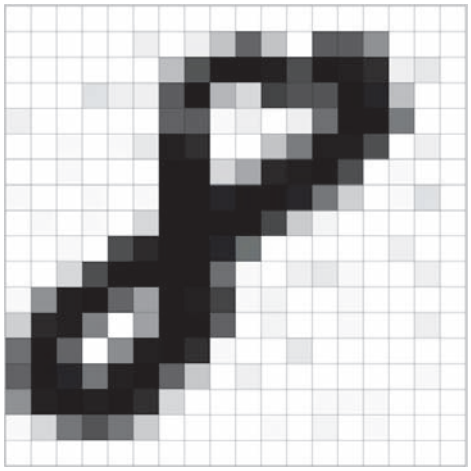


Рисунок 6.3 – Представление изображения в виде пикселей

В результате, используя библиотеку NumPy возможно преобразовать любое изображение в трехмерный массив данных, в котором содержатся значения от 0 до 255 (Рисунок 6.4).

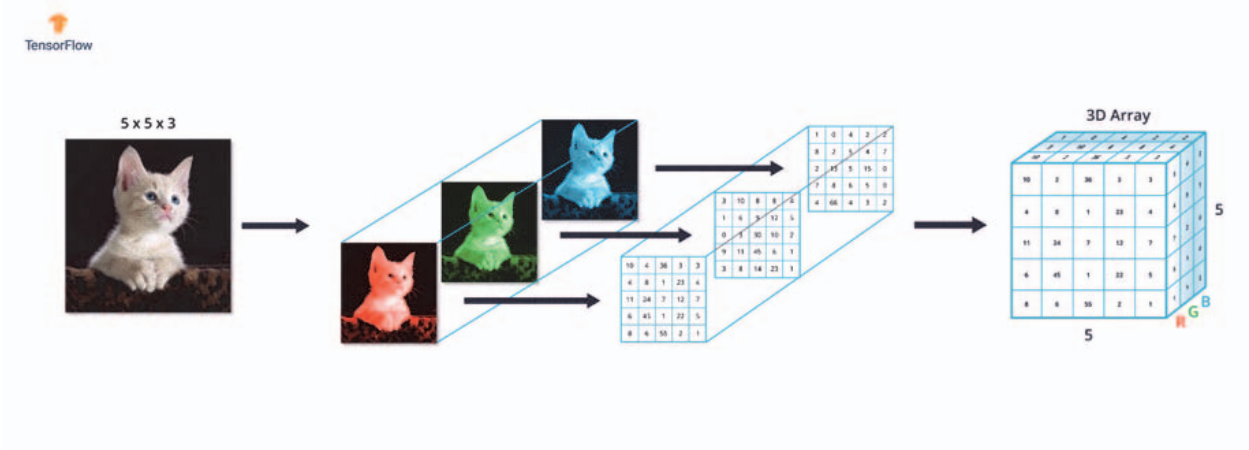


Рисунок 6.4 – Преобразование изображения в числовой массив

Для снижения потребления ресурсов компьютера и времени обработки массива нейронной сетью, каждое числовое значение нормализуется таким образом, чтобы оно было в промежутке от 0

до 1. Это возможно с помощью функции Standartize(), доступной из библиотеки Keras, используя модуль генерации данных.

После преобразования изображения в массив и подачи его в СНС, этот массив определенным образом исследуется и преобразовывается.

Все действия проходят под действием функций из библиотек TensorFlow и Keras [12, 24, 20, 19, 5, 25].

После прохождения входного слоя, массив данных попадает в скрытые слои нейронной сети, которых может быть большое количество и разных видов. Самыми интересными являются слои свертки и пулинга.

При обучении сетей без сверточных слоев, возникает проблема, что такая сеть не может идентифицировать нужный объект на изображении, если он находится не в центре экрана. Она считает, что это совершенно новый объект.

Во избежание этого, используются сверточные слои, которые подобно «скользящему окну» разбивают изображения на пересекающиеся фрагменты. Каждый из них загружается в НС с одним нюансом: для каждого фрагмента используются идентичные веса нейронной сети, т.е. каждый фрагмент обрабатывается одинаково. Если нейросеть в каком-либо участке улавливает оригинальный признак, то такой участок помечается сетью [20].

Если данную операцию объяснять более научным языком, то для свертки используется сверточное ядро, которое может иметь размеры 3x3, 7x7. Операция свертки находит и определяет определенный признак на изображении, который может проявляться в виде перехода от определенного светлого значения признака к определенному темному значению признака. Такой признак будет индивидуален для конкретного класса, он же и будет использоваться для классификации. Чем больше отличительных особенностей будет содержать класс, тем точнее сеть будет выделять его в массе изображений. Итогом является слой свертки или карта признака (Рисунок 6.5) [25].

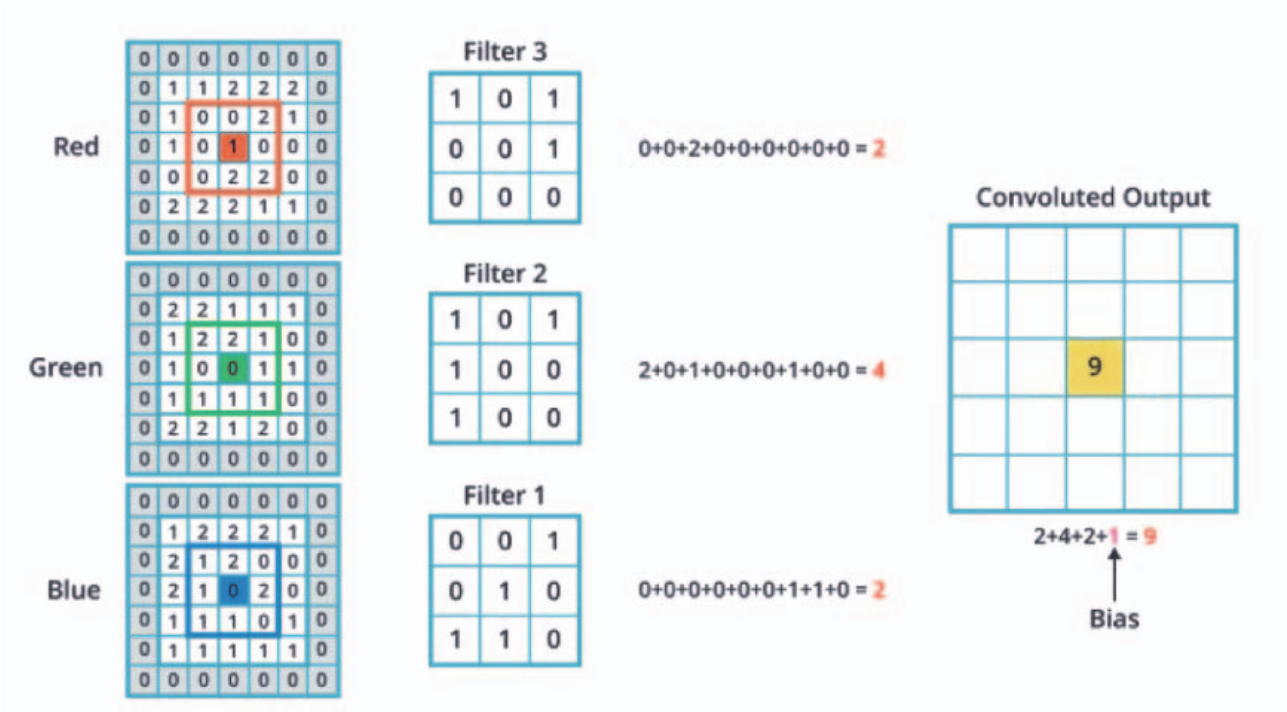


Рисунок 6.5 – Операция свертки

Выходной информацией сверточного слоя является массив данных, который содержит части исходного изображения, которые нейронная сеть считает наиболее значимыми в классификации. Но он слишком громоздкий и несет слишком много лишней информации.

Для уменьшения его размера придумали операцию пулинга, Которая являет собой сжатие изображения или слоя свёртки, используя максимальное или среднее значение. Вследствие этого, группа пикселей (обычно размера 2×2) уплотняется до одного пикселя [13].

Фактически, происходит расширение области захвата и обработки ядром свертки в два раза. Таким образом, оно уже обрабатывает не мелкие детали изображения, а более крупные. Так же пулингом объединяются карты признаков (полученные сверткой) в более абстрактные признаки, уже не пиксели, а черточки и т.д.

Получившийся сжатый массив поступает в полносвязный слой, который выступает в качестве полносвязной нейронной сети. На выходе нейронной сети, изображению присваивается класс, основывая на обработанной информации (Рисунок 6.6).

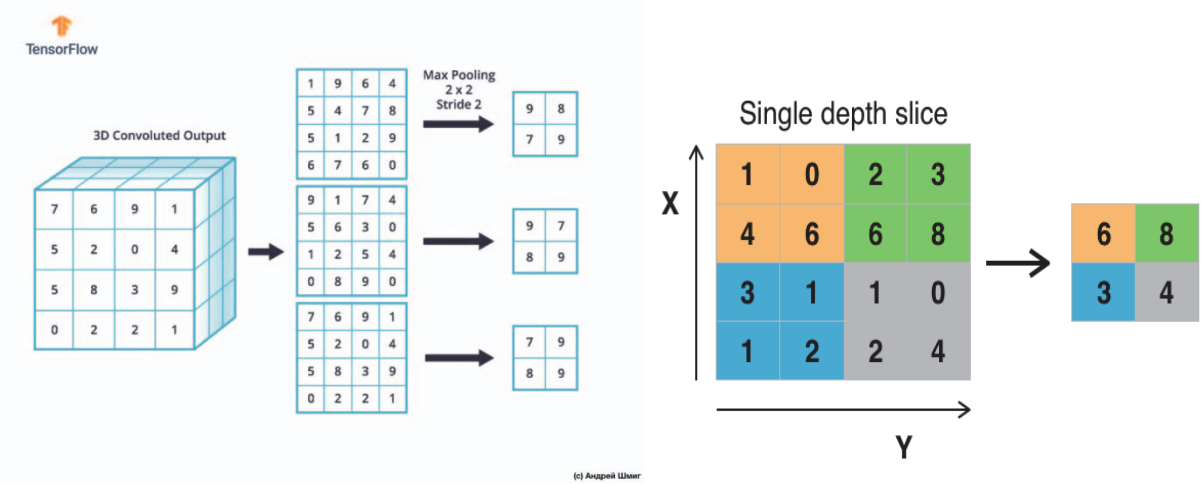


Рисунок 6.6 – Операция пулинга

Эти слои возможно сочетать и складывать между собой столько раз, сколько потребуется для того, чтобы исследователя устроил результат работы в конкретной области применения.

Основополагающим принципом описанного процесса служит начало извлечения крупных признаков с большого изображения и постепенное его уменьшения для исследования мелких признаков, пока результатом не станет единственный вариант исхода [17].

После обучения каждой модели нейронной сети, необходимо оценивать ее качество, дабы понимать, как хорошо, спроектированная модель справляется со своей задачей.

Прежде всего проводится оценка точности модели, которая показывает, на сколько вероятен верных прогноз модели. Она проходит путем подачи в нейронную сеть отдельно отобранных данных, не участвующих в обучении.

При оценке качества модели, внимание обращается на 3 параметра:

- точность на тренировочном и валидационном наборах данных;
- потери на тренировочном и валидационном наборах данных;
- переобучение модели.

Все эти параметры оцениваются путем изучения графиков, которые выводятся с помощью библиотеки Matplotlyb (Рисунок 6.7).

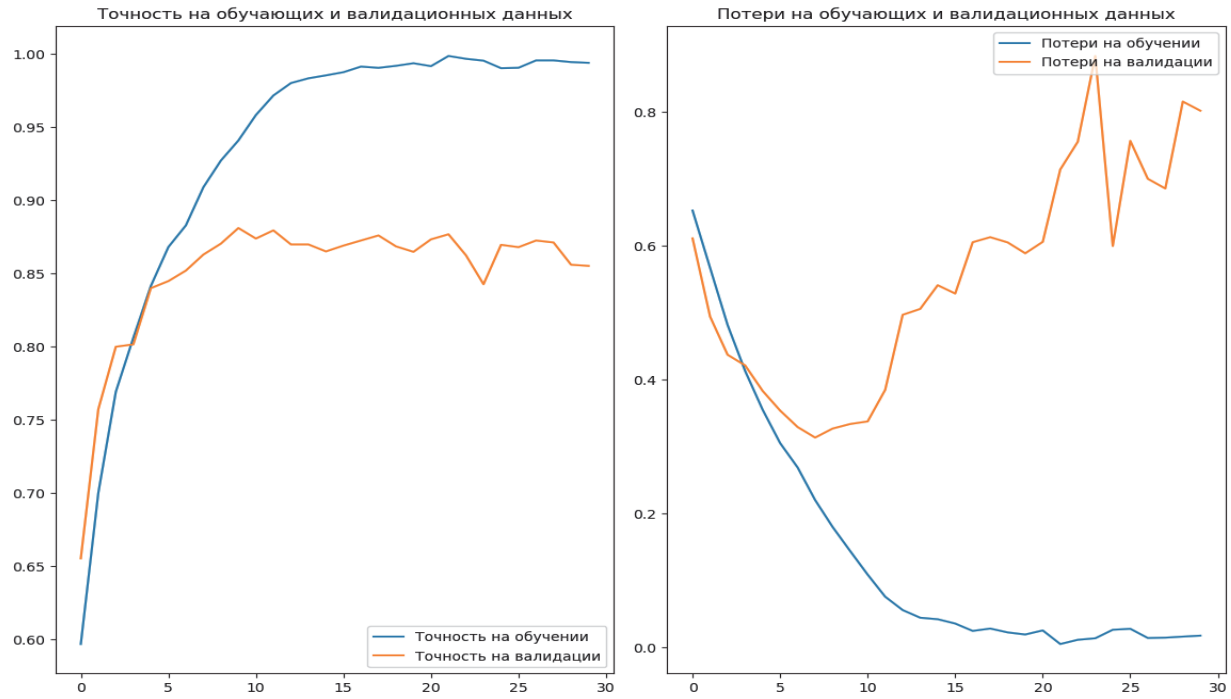


Рисунок 6.7 – Точность и потери при обучении модели

Исследуя эти графики, видно, что примерно на 5 итерации обучения, они начинаются расходиться, что является сигналом к тому, что модель начинает переобучаться, т.е. НС начинает запоминать обучающие данные.

Существуют некоторые методы для того, чтобы избежать переобучения модели и повышения ее точности, при применении которых возможно получить результат при 100 итерациях, представленный ниже.

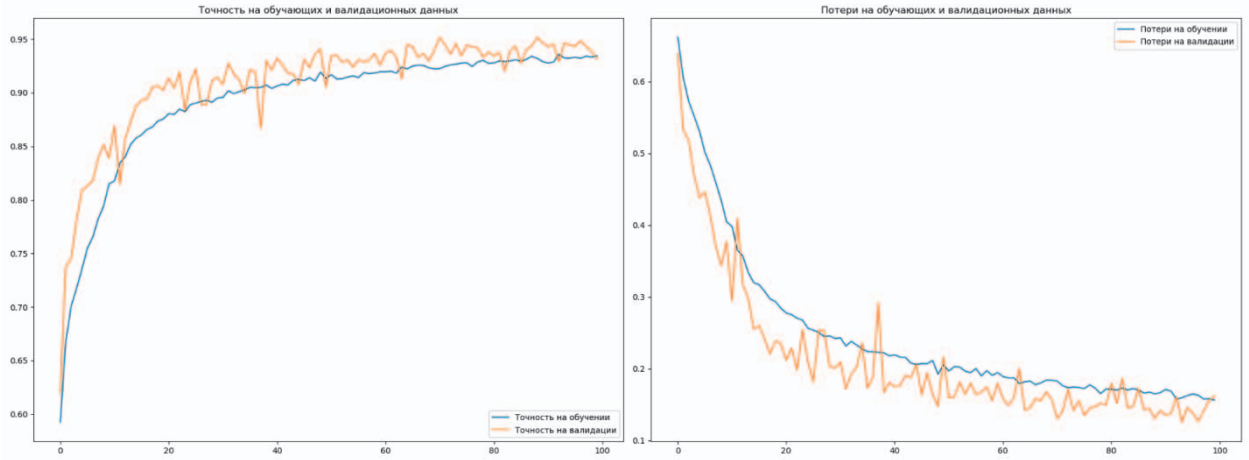


Рисунок 6.8 – Новый результат обучения модели

Исследуя графики можно задуматься о том, чтобы продолжать обучение модели, но экспериментальным путем было определено,

что наиболее оптимальный вариант – взять веса модели при обучении на 25-ой итерации. В этом состоянии модель имеет точность прогноза 90,78%.

6.2 Функциональные модели разрабатываемой системы

IDEF0 (Integrated Definition Function Modeling) – методология функционального моделирования. Основой IDEF0 методологии является понятие «блок», который отображает какую-либо бизнес-функцию. Все стороны блока имеют различные роли:

- левая сторона имеет значение «входа»;
- правая – «выхода»;
- верхняя – «управления»;
- нижняя – «механизма».

Между функциями взаимодействия представляются в виде дуги, которая отображает поток данных, поступающий с одной функции на другую. В зависимости от того, с какой стороной блока связан поток, его называют соответственно "входным", "выходным", "управляющим".

В IDEF0 реализованы три базовых принципа моделирования процессов:

- принцип функциональной декомпозиции;
- принцип ограничения сложности;
- принцип контекста.

Принцип функциональной декомпозиции – это способ моделирования типовой ситуации, когда любое действие, операция, функция могут быть декомпозированы на более простые действия, операции, функции. Другими словами, сложная бизнес-функция может быть представлена в виде совокупности элементарных функций. Представляя функции графически, в виде блоков, можно как бы заглянуть внутрь блока и детально рассмотреть ее структуру и состав.

Важную роль при проектировании программного продукта уделяется контекстным диаграммам. Эта диаграмма показывает взаимодействие программы с объектами, которые ее окружают, то есть внешними объектами [4].

В данной работе контекстная диаграмма представлена:

- одной работой – обработка медиа информации нейронной сетью;
- управляющее воздействие – выбранный пользователем режим работы;
- входным объектом – исходная медиаинформация;
- механизмом осуществления работы – рабочая станция пользователя, пользователь;
- выходным объектом – результат обработки информации.

Контекстная диаграмма имеет номер А-0 и представлена на рисунке 10.

Рисунок 6.9 – Диаграмма А-0 в нотациях IDEF0

Процесс описания программы в целом с разбиением ее на крупные фрагменты называется функциональной декомпозицией, а диаграммы, описываемые в каждом фрагменте, называются диаграммами декомпозиции.

После декомпозиции контекстной диаграммы проводится декомпозиция каждого большого фрагмента программы на более мелкие и так до тех пор, пока не будет достигнут нужный уровень согласно [4].

Декомпозиция контекстной диаграммы представлена на рисунке 11. Она имеет номер А-0 и состоит из следующих основных работ, которые осуществляются при работе с разрабатываемым программным продуктом:

- запуск системы;
- выбор режима работы системы;
- обработка информации.

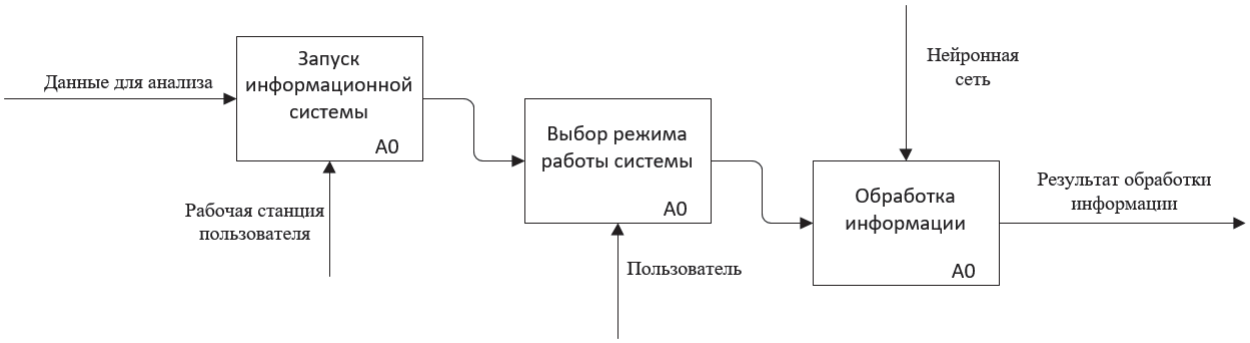


Рисунок 6.10 – Диаграмма декомпозиции A-0 в нотации IDEF0

В декомпозиции контекстной диаграммы представлены основные работы, осуществляемые пользователем при работе с разрабатываемым программным продуктом.

6.3 Инфологическое проектирование системы

Прежде, чем приступить к написанию кода программы, необходимо разработать инфологические модели, на базе которых и будет создаваться приложение.

Исходя из use case диаграммы, разрабатываемое ПО предоставляет возможности пользователю:

- взаимодействие с веб-страницей, разрабатываемого ПО, через которую осуществляется доступ ко всем возможностям продукта;
- загрузки необходимых медиафайлов для работы (изображений и видеозаписей);
- сохранение результатов работы в форматы, предполагаемые выбранной процедурой работы.

Стоит отметить, что выполнение операций, связанных с сохранением и загрузкой информации, пользователю помогает проводник Windows. Т.е. эти операции проходят с использованием проводника операционной системы на рабочей станции.

Диаграмма последовательности (sequence diagram) позволяет отследить, какие действия необходимо совершить пользователем

для реализации необходимого варианта использования (Рисунок 6.11).

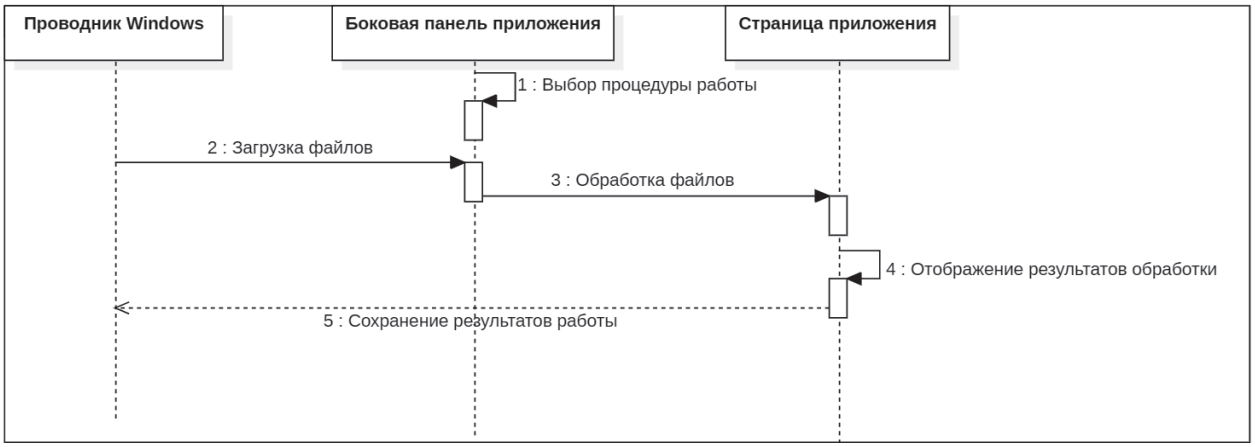


Рисунок 6.11 – Диаграмма последовательности в нотациях UML

Опираясь на нее, прежде всего пользователю для работы необходимо выбрать необходимую процедуру для работы, после в зависимости от процедуры, загрузить файл(ы), только после чего начинается работа и отображение результатов. Цепочку закрывает сохранение результатов работы.

Одной из самых важных диаграмм UML является – диаграмма состояний, которая показывает, в каких состояниях может находится программа и процесс смены ее состояний за счет наступления определенных событий (Рисунок 6.12).

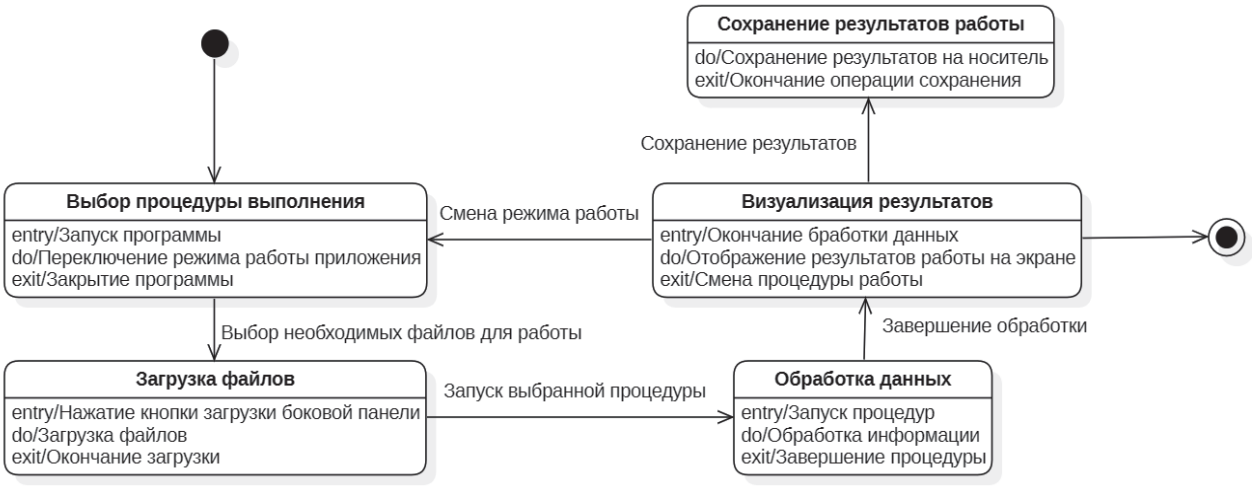


Рисунок 6.12 – Диаграмма состояний в нотациях UML

Диаграмма иллюстрирует, что программа может находиться в 5-ти состояниях:

1. При запуске приложения отрывается веб-страница, на боковой панели которой находится выпадающий список, означающий ожидание выбора необходимого режима работы.

2. Состояние загрузки файлов, в которое программа переходит при нажатии соответствующей кнопки.

3. Состояние обработки данных, в которое программа переходит при нажатии соответствующей кнопки. В этом состоянии происходят вычисления и обработка исходных данных в зависимости от выбранного режима работы.

4. Состояние визуализации данных, в которое программа переходит после окончания обработки данных. В этом состоянии на странице приложения отображаются результаты прошлого состояния.

5. Состояние сохранения результатов, в которое программа переходит либо по нажатию кнопки после обработки данных, либо после окончания визуализации результатов работы.

Одной из наиболее важных диаграмм инфологического моделирования является – диаграмма классов (class diagram). Диаграмма классов определяет типы классов системы и различного рода статические связи, которые существуют между ними. На них изображаются также атрибуты классов, операции классов и ограничения, которые накладываются на связи между классами (Рисунок 6.13).

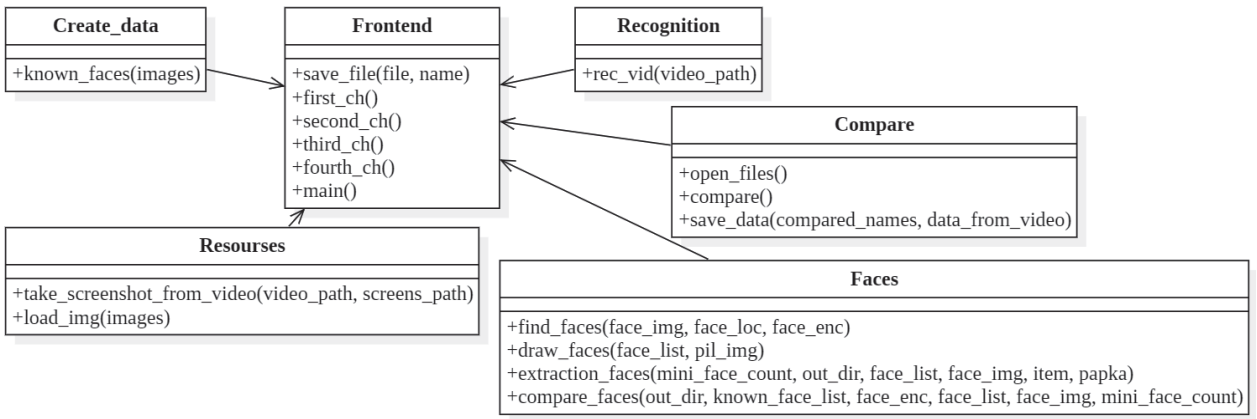


Рисунок 6.13 – Диаграмма классов в нотациях UML

На рисунке выше изображено 6 классов:

1. Frontend – представляет собой исполняемый файл кода, функции которого направлены на визуализацию работы приложения и связывает в себе все имеющиеся классы. Помимо

этого, является точкой входа для программы (Entry point). Представляет собой адрес в оперативной памяти, с которого начинается выполнение программы.

2. Create_data – представляет собой исполняемый файл, который подключается к главному классу программы. Содержит единственную функцию, которая перебирает загруженные изображения людей и выявляет признаки каждого человека, после чего результат сохраняет в файл для дальнейшего использования идентификации личности.

3. Resourses – исполняемый файл, подключающийся к главному классу. Содержит 2 функции, которые направлены на обработку загруженного видео, создание из него скриншотов и выявления признаков людей на них для дальнейшего использования в классификации личностей.

4. Faces – исполняемый файл, подключающийся к главному классу. Служит для произведения манипуляций с лицами. Такие манипуляции, как:

- нахождение всех лиц на изображении;
- обводка рамкой этих лиц;
- вырезка этих лиц для дальнейшего использования;
- сравнение лиц между собой.

5. Compare – исполняемый файл, подключающийся к главному классу. Его специфика в том, что внутренние функции направлены на сопоставление кодировок лиц с фотографий известных людей и кодировок изображений из видео для создание общей базы знаний с кодировками лиц и соответствующих им имен.

6. Recognition – исполняемый файл, подключающийся к главному классу. Выполняет важную функцию, направленную на идентификацию личностей на видео, используя подготовленную ранее базу знаний.

6.4 Разработка модуля информационной системы распознавания образов

Python предлагает большое количество библиотек с открытым кодом как для проектирования нейронных сетей, так и для анализа больших данных и построения интерфейса программы для визуализации работы.

В основе проекта используются следующие библиотеки:

• Face_Recognition – создание и настройка нейронных сетей, направленных на распознавание лиц людей;

• OpenCV – работа с изображениями и видео;

• Streamlit – создание визуальной оболочки программы в виде веб-страницы.

Помимо этих библиотек использовались вспомогательные:

• Os – работа с файловой системой операционной сети;

• Pathlib – работа с дирректориями операционной системы;

• Time – позволяет отслеживать операции во времени;

• NumPy – работа с массивами чисел;

• Pickle – сохранение и чтение файлов соответствующего формата;

• Pillow – чтение и запись изображений.

1. Визуальная оболочка программы (класс Frontend).

Является главным файлом, к которому подключаются все оставшиеся. Выступает их связующим звеном для обеспечения взаимодействия пользователя с программой. Точкой входа является функция «main()» (Рисунок 6.14).

```
111  def main():
112      st.set_page_config(page_title='Идентификация лиц', layout='wide')
113      st.title("Идентификация лиц")
114      st.sidebar.title('Режимы работы')
115
116      choose = st.sidebar.selectbox("Выберите необходимую операцию", ("выбрать...",
117                                                                       "1. Обработка известных фото",
118                                                                       "2. Извлечение признаков из видео",
119                                                                       "3. Подготовка базы данных",
120                                                                       "4. Идентификация лиц"))
121
122
123      if choose == "1. Обработка известных фото":
124          first_ch()
125      elif choose == "2. Извлечение признаков из видео":
126          second_ch()
127      elif choose == "3. Подготовка базы данных":
128          third_ch()
129      elif choose == "4. Идентификация лиц":
130          fourth_ch()
```

Рисунок 6.14 – Функция «main()

После инициализации функции в строке 111, обозначаются основные элемента интерфейса, такие как название страницы, заголовок, заголовок боковой панели и выпадающий список с выбором режима работы на боковой панели (строки 112-120).

Заканчивается функция рядом условий, прохождение которых зависит от выбора режима работы в выпадающем списке. В зависимости от выполненного условия, запускается функция соответствующего режима работы. Каждая из этих функций запускает свой исполняемый файл, в которых ведется работа.

2. Класс «create_data».

Содержит в себе единственную функцию, в которой обрабатываются изображения с людьми, имена которых известны и кодировки записываются в файл.

Прежде всего, в ней инициализируется шкала прогресса, окно разбивается на 3 колонки и инициализируются пустые списки, в которые позже будут записаны кодировки и имена людей с фото (Рисунок 6.15).

```
5    def known_faces(images):
6        progress_bar = st.progress(0)
7        col0, col1, col2 = st.columns((1, 1, 1))
8        side = 0
9
10       knownEncodings = []
11       knownNames = []
```

Рисунок 6.15 – Инициализация элементов интерфейса

Оставшийся код функции находится в цикле, в котором перебираются все загруженные изображения по порядку. Прежде всего программа выводит изображение на экран, чтобы пользователь понимал, с каким изображением программа в данный момент работает и правильно ли она его загрузила:

col0.caption(f'Изображение: {j+1}/{len(images)}: "{name}"')

col0.image(images[j], width=200)

После чего изображение преобразуется в массив чисел. Этот массив обрабатывается нейронной сетью, которая распознает лица на изображении и возвращает их координаты. Затем используя координаты найденных лиц, нейронка рассчитывает их кодировку:

face_img = face_recognition.load_image_file(images[j])

face_loc = face_recognition.face_locations(face_img, model='cnn')

face_enc = face_recognition.face_encodings(face_img,
known_face_locations=face_loc, num_jitters=100)

В итоге, полученная информация записывается в списки, которые были инициализированы в начале функции:

for encoding in face_enc:
knownEncodings.append(encoding)
knownNames.append(name)

3. Класс «Faces».

Содержит в себе функции, связанные с манипуляцией с лицами на изображениях. Первая функция «find_faces» – как следует из названия, находит лица на изображениях (Рисунок 6.16).

```
def find_faces(face_img, face_loc, face_enc):
    face_img1 = face_img
    face_list =[]
    face_enc_list = []
    enc_count = 0

    for face_location in face_loc:
        top, right, bottom, left = face_location
        face_img = face_img[top:bottom, left:right]

        face_list.append([top, bottom, left, right])
        face_enc_list.append(face_enc[enc_count])

        enc_count += 1
    return face_list, face_img1
```

Рисунок. 6.16 – Первая функция класса «Faces»

Функция принимает изображение с лицами, их координаты и кодировки, после чего инициализируются списки, в которые записываются координаты и кодировки для дальнейшего использования.

Вторая функция класса «draw_faces» (Рисунок 6.17).

```
21  def draw_faces(face_list, pil_img):
22
23      draw = ImageDraw.Draw(pil_img)
24      for i in range(len(face_list)):
25          top, bottom, left, right = face_list[i]
26          draw.rectangle(((left, top), (right, bottom)), outline="green", width=5)
27
28      del draw
29      return pil_img.show()
```

Рисунок 6.17 – Вторая функция класса «Faces»

На вход получает записанные координаты и изображение. Используя библиотеку Pillow, на изображении рисуются зеленые квадраты вокруг лиц (строки 24-26).

Третья функция «extraction_faces» (Рисунок 6.18).

```
31  def extraction_faces(mini_face_count, out_dir, face_list, face_img, item, papka):
32      if item == True:
33          for i in range(len(face_list)):
34              top, bottom, left, right = face_list[i]
35              draw_mini_face = face_img[top-40:bottom+20, left-20:right+20]
36              draw_face = Image.fromarray(draw_mini_face)
37              draw_face.save(f'{out_dir}/{i}_person/{mini_face_count}_face_img.jpg')
38              mini_face_count += 1
39
40      elif item == False:
41          top, bottom, left, right = face_list
42          draw_mini_face = face_img[top-40:bottom+20, left-20:right+20]
43          draw_face = Image.fromarray(draw_mini_face)
44          os.mkdir(f'{out_dir}/{papka-1}_person')
45          draw_face.save(f'{out_dir}/{papka-1}_person/{mini_face_count}_face_img.jpg')
46          mini_face_count += 1
47
48      else:
49          top, bottom, left, right = face_list
50          draw_mini_face = face_img[top-40:bottom+20, left-20:right+20]
51          draw_face = Image.fromarray(draw_mini_face)
52          draw_face.save(f'{out_dir}/{item-2}_person/{mini_face_count}_face_img.jpg')
53          mini_face_count += 1
54
55      return mini_face_count
```

Рисунок 6.18 – Третья функция класса «Faces»

В зависимости от значений, передаваемых в функцию, выстраивается определенная логика, по которой программа из обрабатываемого изображения копирует лица людей в

определенные папки в файловой системе. Важно, что в каждой из папок содержатся лица только одного человека (Рисунок 6.19).

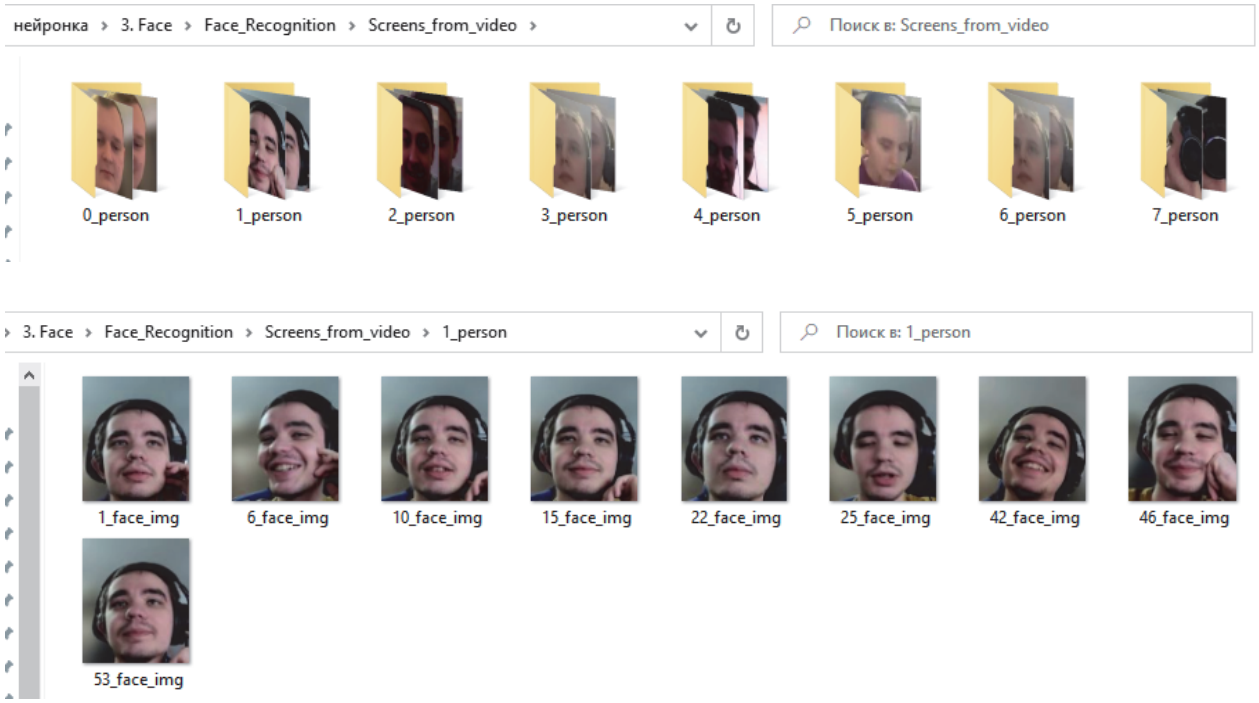


Рисунок 6.19 – Сохранение лиц в файловую систему

Можно видеть, что в директории создаются папки с порядковыми номерами, в которых расположены изображения похожих лиц.

Последняя функция «compare_faces».

Сравнивает между собой все кодировки, чтобы каждая из них соответствовала определенному человеку (Рисунок 6.20).

```
64  def compare_faces(out_dir, known_face_list, face_enc, face_list, face_img, mini_face_count):
65
66      for enc_id in range(0, len(face_enc)):
67          ins = False
68          for item in range(0, len(known_face_list)):
69              result = face_recognition.compare_faces(known_face_list[item], face_enc[enc_id], tolerance=0.45)
70
71              if any(result):
72                  known_face_list[item].append(face_enc[enc_id])
73                  mini_face_count = extraction_faces(mini_face_count, out_dir, face_list[enc_id], face_img, item+2, False)
74                  ins = True
75
76          if ins == False:
77              known_face_list.append([face_enc[enc_id]])
78              mini_face_count = extraction_faces(mini_face_count, out_dir, face_list[enc_id], face_img, False, (len(known_face_list)))
79      return mini_face_count
```

Рисунок 6.20 – Четвертая функция класса «Faces»

В цикл поступает список с кодировками на изображении (строка 66). После чего каждая кодировка сравнивается с уже имеющимися в сохраненном списке. Высчитывается нейронной

сетью евклидово расстояние между ними, которое по умолчанию и чаще всего используется равно 0.6. Но поскольку некоторые люди могут быть похожи друг на друга, было принято решение снизить значение до 0.45 (строка 69). Благодаря этому, в каждой папке сохраняются лица только одного и того же человека. Но из-за строгого критерия, список составляется из большего количества людей, нежели предполагается, потому что нейронная сеть считает разными людьми, если человек смотрит камеру и в профиль.

Эта проблема решается на стадии подготовки итоговой базы знаний.

4. Класс «Compare».

Состоит из 3 функций, первая из которых служит для загрузки и чтения сохраненных ранее кодировок.

Вторая функция «compare» служит для сравнения кодировок двух файлов (Рисунок 6.21).

```
16  def compare():
17      time.sleep(2)
18      compared_names = []
19      data_from_video, names, names_encs = open_files()
20      st.success('[SUCCESS] Данные загружены!\n')
21
22      for i in range(len(data_from_video)):
23          compared_names.append([])
24
25      st.info('[INFO] Инициализация сопоставления кодировок...')
26      for enc_id in range(0, len(names_encs)):
27          for item in range(0, len(data_from_video)):
28              result = face_recognition.compare_faces(data_from_video[item], names_encs[enc_id], tolerance=0.5)
29
30              if any(result):
31                  data_from_video[item].append(names_encs[enc_id])
32                  compared_names[item] = [names[enc_id]]
33
34      return compared_names, data_from_video
```

Рисунок 6.21 – Вторая функция класса «Compare»

Прежде всего инициализируется список, в который будут записаны имена людей, которые уже нейронная сеть сравнила, после чего запускается первая функция класса (стоки 18-19).

В двойном цикле сравниваются кодировки разных списков уже используя евклидово расстояние, равное 0.5 (строки 26-32). Функция возвращает список имен и список кодировок.

Третья функция «save_data». Помимо того, что объединяет файлы кодировок, дополняет файл с известными кодировками значениями из файла видео. При этом, все лишние кодировки удаляются.

5. Класс «Recognition».

Заключительный файл программы, который состоит из единственной функции «rec_vid», которой поступает путь до выбранной видео записи.

При запуске, на странице приложения размечаются элементы – шкала прогресса, место отображение кадра, место отображения номера кадра и кнопки завершения работы:

```
def rec_vid(video_path):
    progress_bar = st.progress(0)
    image_loc = st.empty()
    inf_loc = st.empty()
    close = st.button('Завершить и сохранить видео')
```

После этого подгружается база знаний:

```
data = pickle.loads(open("Face_Recognition/encodings/known_persons", "rb").read())
known_face_names = data['names']
data = data['encodings']
```

Далее инициализируется процесс обработки видео и записи нового (Рисунок 6.22).

```
output = cv2.VideoWriter('Face_Recognition/encodings/output_video.mp4', cv2.VideoWriter_fourcc(* 'mp4v'), 30, frame_size)

while True:
    ret, frame = cap.read()
    if frame_id % 2 == 0:
        faces = face_recognition.face_locations(frame, model='cnn')
        face_enc = face_recognition.face_encodings(frame, known_face_locations=faces)
        for face_encoding, face_location in zip(face_enc, faces):
            for item in range(len(data)):
                face_distances = face_recognition.face_distance(data[item], face_encoding)
                best_match_index = np.argmin(face_distances)
                if face_distances[best_match_index] < 0.5:

                    name = known_face_names[item]

                    left = face_location[3] - 40
                    top = face_location[0] - 40
                    right = face_location[1] + 40
                    bottom = face_location[2] + 40

                    color = [0, 255, 0]
                    cv2.rectangle(frame, (left, top), (right, bottom), color)

                    cv2.rectangle(frame, (left, bottom), (right, bottom + 23), (0, 255, 0), cv2.FILLED)
                    cv2.putText(frame, name[0], (left, bottom + 20), cv2.FONT_HERSHEY_COMPLEX, 0.7, (255, 255, 255), 2)
```

Рисунок 6.23 – Алгоритм обработки видео

Цикл означает, что пока идет чтение видео, алгоритм ниже будет работать. Первым делом, записывается номер и кадр видео

(строка 28). После каждый второй кадр обрабатывается нейронной сетью. Как и в прошлых примерах, на кадре находятся координаты лиц, с них снимаются кодировки и сравниваются с кодировками в базе знаний, используя евклидово расстояние, равное 0.5 (строки 30-34). Наилучший результат отображается на экране.

Для этого на изображение наносятся увеличенные рамки вокруг лиц, под которыми располагаются имя и фамилия человека (строки 40-49).

6.6 Разработка инструкции пользования информационной системой

При запуске программы, открывается веб-страница с боковой панелью приложения (Рисунок 6.24).

Рисунок 6.24 – Стартовая страница программы

На боковой панели расположен выпадающий список с выбором режима работы программы:

- обработка известных фото;
- извлечение признаков из видео;
- подготовка базы данных;
- идентификация лиц (Рисунок 6.25).

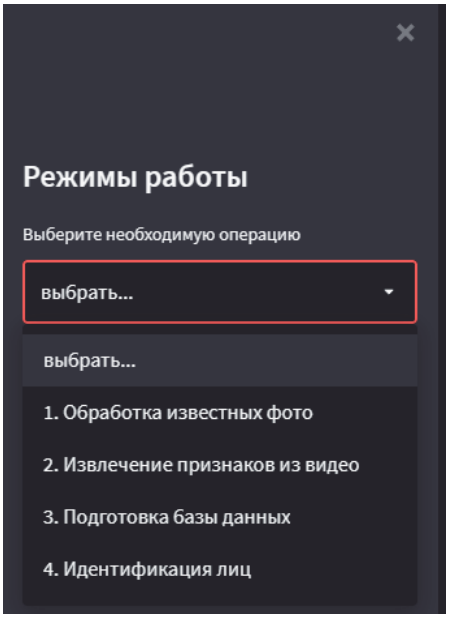


Рисунок 6.25 – Выбор режима работы программы

Первый режим позволяет обработать изображения с лицами известных людей, которые в дальнейшем используются для идентификации.

Второй режим позволяет обработать записанное видео и декодировать лица на нем.

Третий режим совмещает 2 кодировки, созданные ранее, что позволяет создать внушительную базу знаний для более точной финальной идентификации личностей.

Последний режим позволяет идентифицировать личности на видео, используя созданную базу знаний.

Выбрав первый режим работы программы, пользователю необходимо загрузить изображение, с помощью соответствующей кнопки и взаимодействуя с проводником операционной системы (Рисунок 6.26).

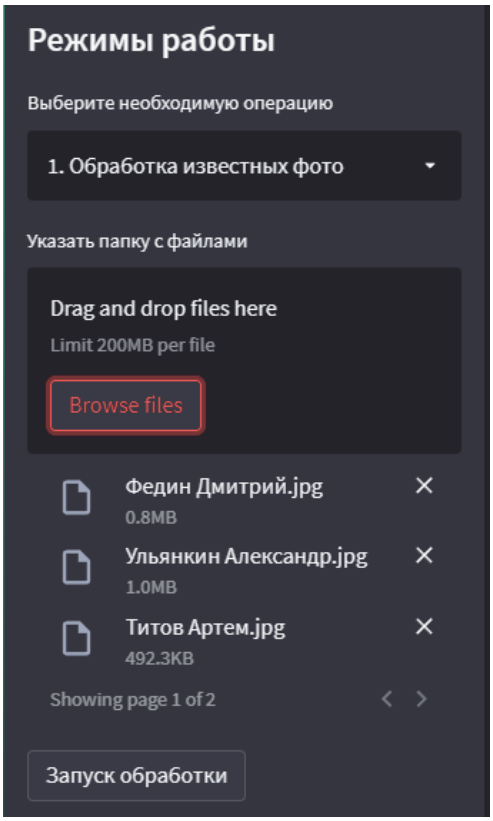


Рисунок 6.26 – Загрузка изображений для первого режима работы

После загрузки, отображается кнопка запуска работы, по нажатию которой, программа обрабатывает изображения, записывая кодировки в файл с присвоением имен из названий изображений. Процесс обработки и ход выполнения отображается на странице приложения (Рисунок 6.27).

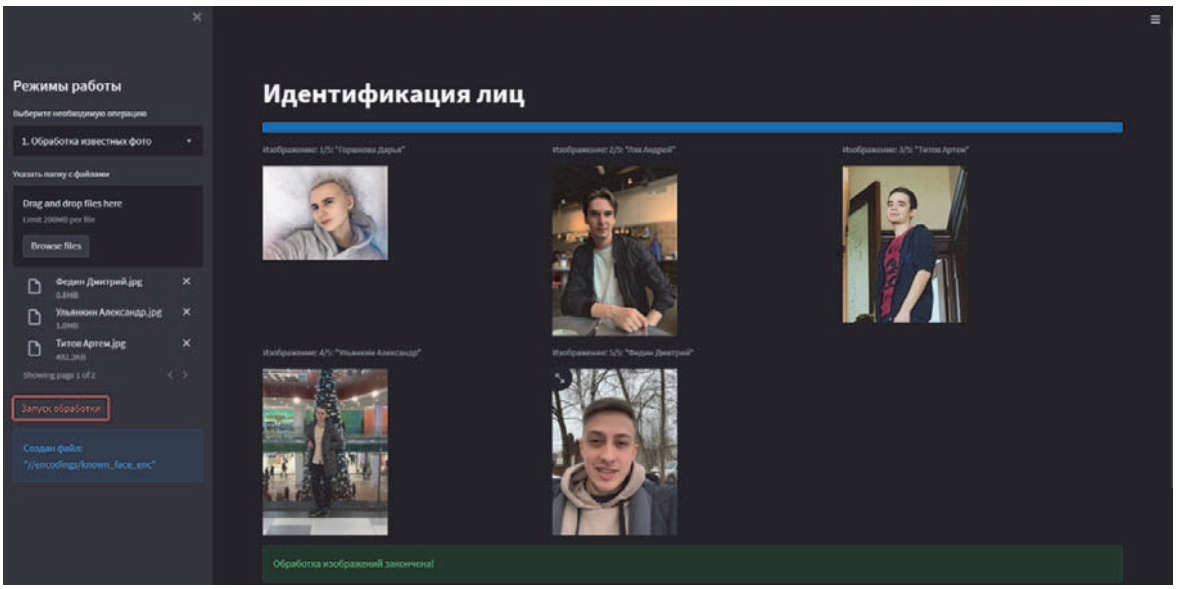


Рисунок 6.28 – Процесс обработки изображений

Под заголовком программы отображается шкала прогресса выполнения, под которой последовательно отображаются уже декодированные изображения. По окончанию работы программа сигнализирует зеленой полосой с надписью: «Обработка изображений закончена». После этого на боковой панели появляется информационной сообщение, знаменующее путь, по которому был сформирован файл с кодировками.

Выбрав второй режим работы, пользователю необходимо определиться с необходимой процедурой работы:

- создание скриншотов из видео;
- извлечение признаков из скриншотов (Рисунок 6.29).

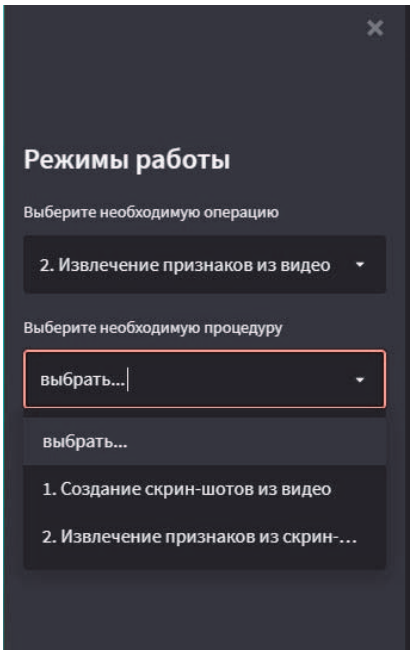


Рисунок 6.29 – Выбор процедуры

Как следует из названия, первая процедура подразумевает загрузка видео и извлечения из него до 12 скриншотов, которые производятся каждые 3 секунды. В процессе на экране они показываются с присвоением порядкового номера каждому (Рисунок 6.30).

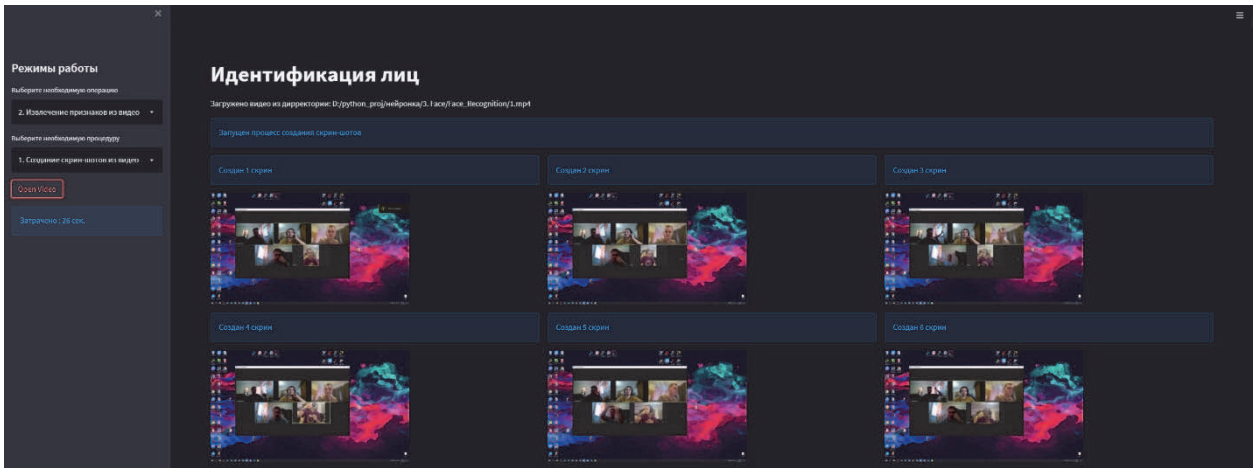


Рисунок 6.30 – Создание скриншотов из видео

Для второй процедуры необходимо загрузить сохраненные скриншоты, после чего по нажатию на соответствующую кнопку, программа извлекает признаки с каждого изображения (Рисунок 6.31).

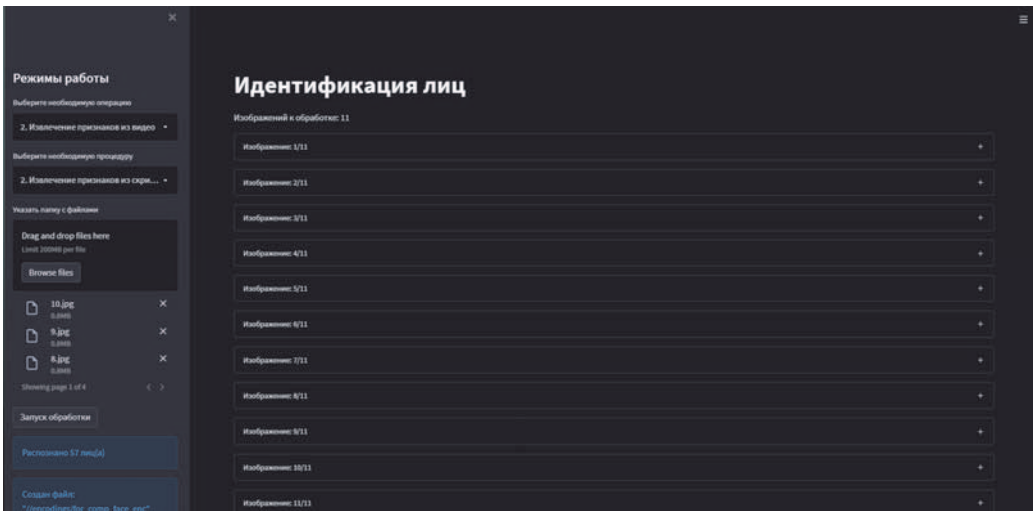


Рисунок 6.31 – Извлечение признаков из скриншотов

В ходе выполнения на экране отображаются контейнеры со скринами по порядку (Рисунок 6.32). По завершению работы, на боковой панели показывается 2 информационных сообщения:

- количество распознанных лиц (общее количество со всех изображений);

- путь к сохраненному файлу с кодировками лиц.

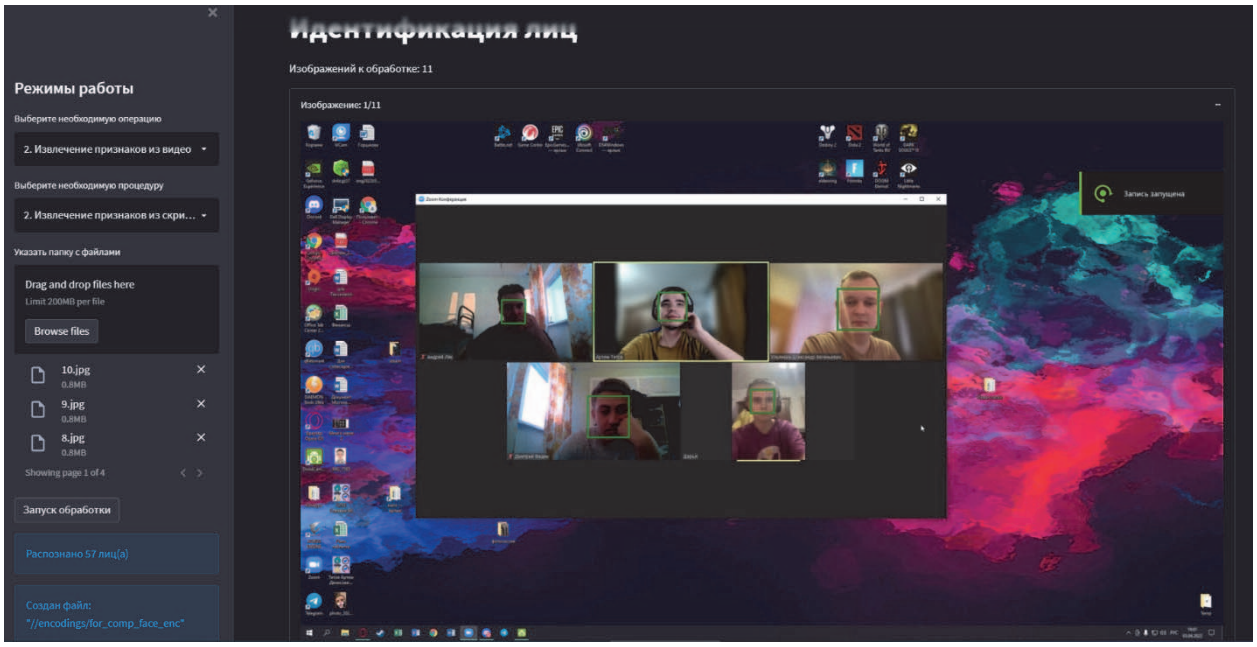


Рисунок 6.32 – Контейнер с изображением

Каждый контейнер содержит обработанное изображение, на котором можно видеть по зеленым рамкам, где находятся распознанные и декодированные лица.

Выбрав третий режим работы, пользователю достаточно лишь нажать на кнопку для запуска работы. Программа автоматически берет уже подготовленные два файла с кодировками оттуда, куда они были сохранены и проводит логические вычисления (Рисунок 6.33).

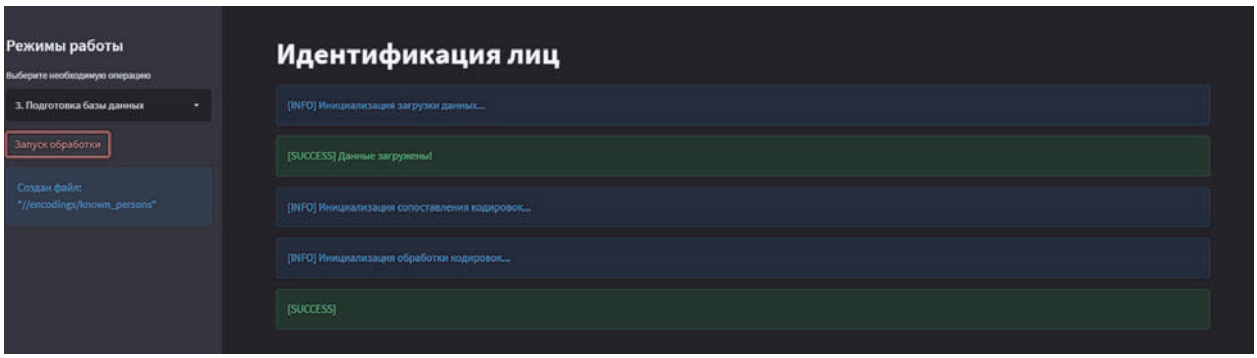


Рисунок 6.33 – Выполнение третьего режима работы

В ходе выполнения на экране отображаются информационные сообщения, показывая пользователю, на какой стадии выполнения находится программа. По завершению, на боковой панели отображается путь к файлу с итоговой базой знаний.

Последний режим работы, который использует ранее созданную базу знаний. Для начала работы, пользователь должен загрузить видео, на котором необходимо идентифицировать людей (Рисунок 6.34).

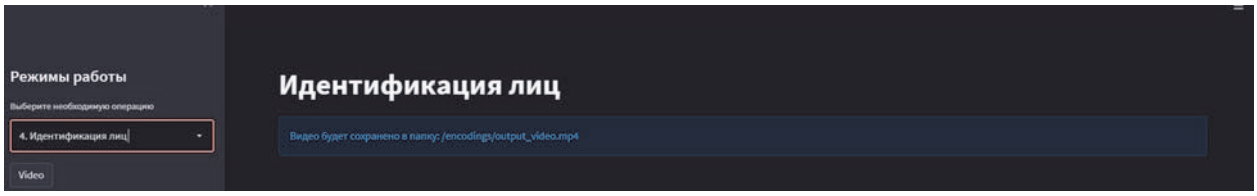


Рисунок 6.34 – Начало работы с 4 режимом

После выбора видео, начинается процесс работы, который возможно остановить, нажав кнопку (Рисунок 6.35).

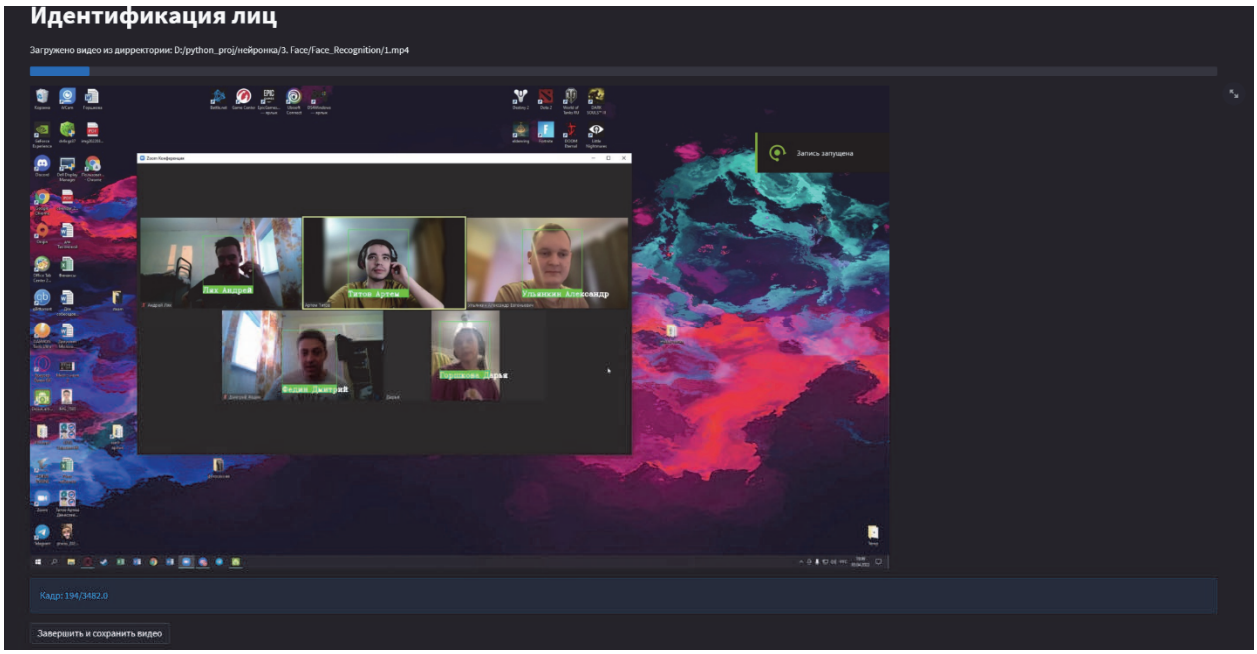


Рисунок 6.35 – Процесс идентификации лиц

Под заголовком программы отображается путь к видео, которое выбрал пользователь. Под ним расположена шкала прогресса. Под ней отображается каждый второй кадр видео с уже распознанными лицами.

Лица обводятся рамкой, под которой пишется присвоенное имя с фамилией человека. Ниже в информационном сообщении отображен порядковый номер отображаемого кадра и кнопка остановки и сохранения видео с идентификацией.

Процесс обработки достаточно долгий даже на производительной рабочей станции, поэтому чтобы не дожидаться обработки всего видео, ниже расположена кнопка, по нажатию

которой процесс будет остановлен, а уже обработанные кадры будут сохранены в видео запись с 30 кадрами в секунду. Таким образом, обработки 120 кадров вполне достаточно для того, чтобы понимать, кто на изображении.

Выводы. С целью упрощения процесса верификации присутствующих на онлайн конференции для аграрных вузов страны была разработана и построена информационная система с использованием искусственного интеллекта.

В процессе планирования разработки, созданы функциональные и инфологические модели проектирования. Они подробно иллюстрируют работу и возможности ИС.

Система состоит из шести связанных между собой модулей. Первый модуль позволяет обработать изображения с лицами известных людей, которые в дальнейшем используются для идентификации. Второй модуль позволяет обработать записанное видео и декодировать лица на нем. Третий модуль совмещает 2 кодировки, созданные ранее, что позволяет создать внушительную базу знаний для более точной финальной идентификации личностей. Четвертый модуль позволяет идентифицировать личности на видео, используя созданную базу знаний. Последний модуль является графической оболочкой информационной системы, которая связывает все предыдущие модули для реализации взаимодействия работы с пользователем системы.

В работе проведено сравнение с имеющимися системами распознавания лиц и оценка эффективности разработки.

Библиографический список к главе 6

1. Быстренина, И.Е. Информационное обеспечение агропромышленного комплекса / И.Е. Быстренина // Кормопроизводство, 2015. – №5. – с. 8 - 12. – Текст: непосредственный.

2. Вандер, Дж.Плас. Python для сложных задач. Наука о данных и машинное обучение / Дж.Плас Вандер. – Питер, 2018. – 426 с. – Текст: непосредственный.

3. Гелиг, А. Введение в математическую теорию обучаемых распознающих систем и нейронных сетей / А.Х. Гелиг, А.С.

Матвеев. - М.: Издательство СПбГУ, 2018. – 224 с. – Текст: непосрдественный.

4. Долганов, О. И. Моделирование бизнес-процессов: учебник и практикум для академического бакалавриата / О.И. Долганов, Е.В. Виноградова, А.М. Лобанова, М.: Юрайт, 2016. – 127 с. – Текст: непосредственный.

5. Жерон О. Прикладное машинное обучение с помощью Scikit-Learn и TensorFlow. Концепции, инструменты и техники для создания интеллектуальных систем / О. О. Жерон. – Вильямс, 2018. – 688 с. – Текст: непосредственный.

6. Землянский, А.А. Управление информационными ресурсами в научно-исследовательской работе: учебное пособие / А.А. Землянский, И.Е. Быстренина. – М.: Издательско-торговая корпорация «Дашков и Ко», 2020. – 110 с. – Текст: непосредственный.

7. Круглов, В. В. Искусственные нейронные сети. Теория и практика / В.В. Круглов, В.В. Борисов. – М.: Горячая линия - Телеком, 2019. – 382 с. – Текст: непосредственный.

8. Латыпова, Р. Нейронные сети / Р. Латыпова. – М.: LAP Lambert Academic Publishing, 2018. – 422 с. – Текст: непосредственный.

9. Машинное обучение. Как обучить сверточные нейронные сети на Python (TensorFlow Eager API). – URL: https://www.machinelearningmastery.ru/ convolutional-neural-networks-an-introduction-tensorflow-eager-api-7e99614a2879/ (дата обращения: 16.10.2021). – Текст: электронный.

10. Мюллер, А. Введение в машинное обучение с помощью Python. Руководство для специалистов по работе с данными / А. Мюллер, С. Гвидо. – Вильямс, 2017. – 480 с. – Текст: непосредственный.

11. Рашка, С. Python и машинное обучение. Машинное и глубокое обучение с использованием Python, scikit-learn и TensorFlow / С. Рашка, В. Мирджалили. – Вильямс, 2019. – 337 с. – Текст: непосредественный.

12. Русскоязычная документация по Keras. – URL: https://ru-keras.com/home/ (дата обращения 05.02.2022). – Текст: электронный.

13. Рутковская, Д. Нейронные сети, генетические алгоритмы и нечеткие системы / Д. Рутковская, М. Пилиньский, Л. Рутковский. – М.: Горячая линия - Телеком, 2018. – 384 с. – Текст: непосрдетсвенный.

14. Ульянкин, А.Е. Автоматизация анализа международных эколого-экономических систем на основе технологий машинного обучения А.Е. Ульянкин // Электронный сетевой политематический журнал «Научные труды КУБГТУ». – 2020. – № 6. – с.74-83. – Текст: непосредственный.

15. Учет и планирование рабочего времени сотрудников организации: разработка информационных систем: монография / И.Е. Быстренина, Т.С. Белоярская, И.В. Макунина, А.В. Миронцева, А.А. Шилова, А.А. Ласточкина, В.В. Антонова. - М.: Издательство «Научный консультант», 2019. – 148 с. – Текст: непосредственный.

16. Хайкин, С. Нейронные сети: полный курс / С. Хайкин. – М.: Диалектика, 2019. – 1104 с. – Текст: непосредственный.

17. Шолле, Ф. Глубокое обучение на Python / Ф. Шолле. – Питер, 2018. – 590 с. – Текст: непосредственный.

18. Benfort, B. Applied Text Analysis with Python. / B. Benfort, T. Ojeda, R. Bilbro // Enabling Language-Aware Data Products with Machine Learning. – O'Reilly Media, 2018. – p. 264. – Текст: непосредственный.

19. Geron, Au. Hands-On Machine Learning with Scikit-Learn and TensorFlow: Concepts, Tools, and Techniques to Build Intelligent Systems. –O'Reilly Media, 2017. – p. 332. – Текст: непосредственный.

20. Gibson, A. Deep Learning / A. Gibson, J. Patterson. – O'Reilly Media, 2017. – p. 187. – Тексu: непосредственный.

21. Goodfellow, I. Deep Learning. / I. Goodfellow, Y, Bengio, A. Courville. – MIT Press, 2016. – p. 291. – Текст: непосредственный.

22. Haykin, S. Neural Networks and Learning Machines / S. Haykin. – Pearson, 2018. – p. 260. – Текст: непосредственный.

23. How to Train Convolutional Neural Networks in Python (TensorFlow Eager API). – URL: https://towardsdatascience.com /convolutional-neural-networks-an-introduction-tensorflow-eager-api-7e99614a2879 (дата обращения: 08.05.2022). – Текст: электронный.

24. Machine Learning with scikit-learn. – URL: http://amueller.github.io/sklearn_tutorial/#/ (дата обращения 15.02.2022). – Текст: электронный.

25. Rashka, S. Python Machine Learning: Machine Learning and Deep Learning with Python, scikit-learn, and TensorFlow / S. Rashka, V. Mirjalili. – Packt Publishing, 2017. – p. 124. – Текст: непосредственный.

26. Streamlit Documentation. – URL: https://docs.streamlit.io/library/api-reference/write-magic/st.write (дата обращения: 23.05.2022). – Текст: электронный.

Приложения

Приложение А

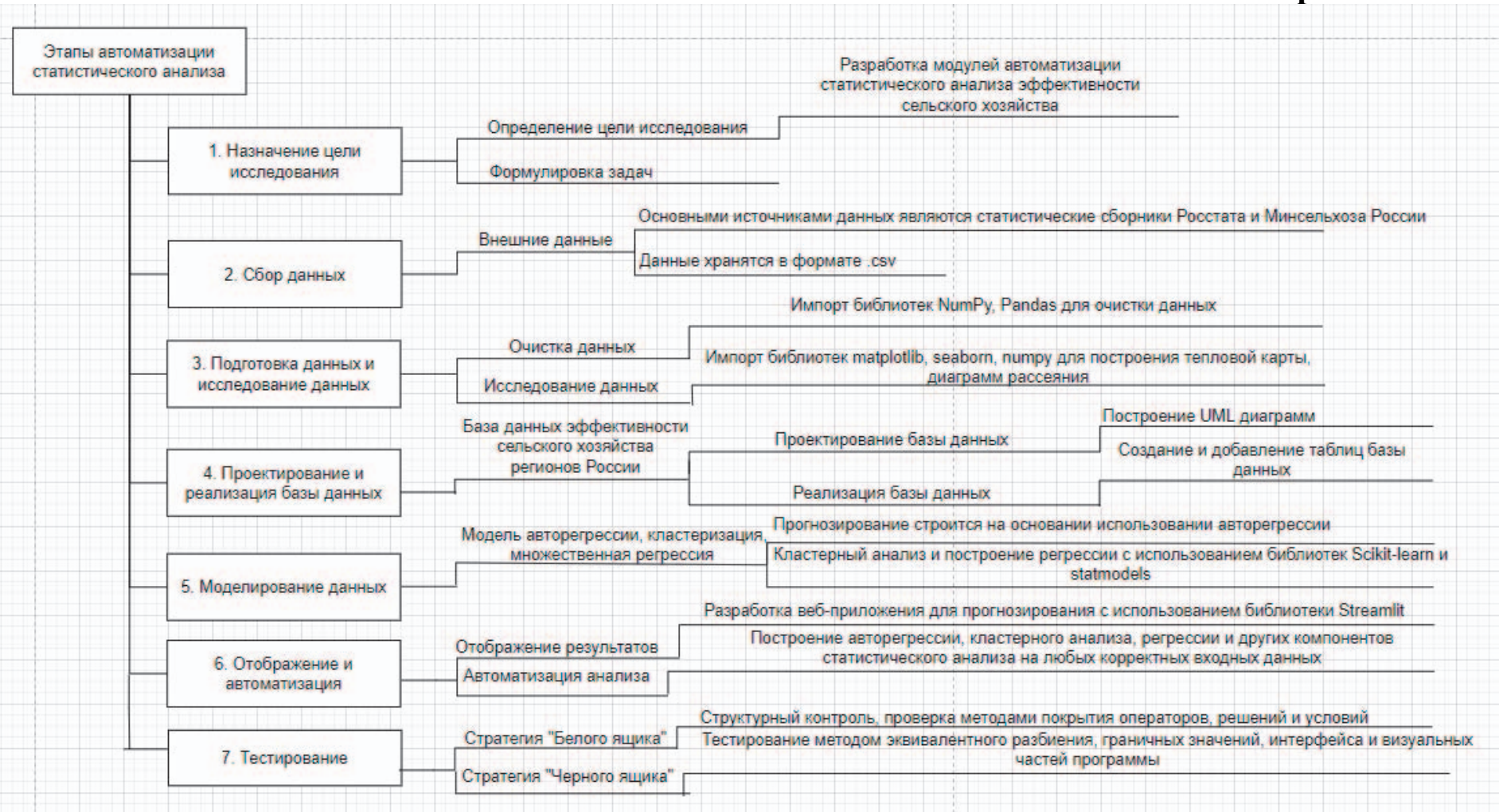


Рисунок А.1 – Алгоритм автоматизации статистического анализа

Приложение Б

Листинг модуля прогнозирования

```
    #Импортируем библиотеки для загрузки данных и
анализа данных
    import pandas as pd

    #Загрузим данные в виде переменной data
    data = pd.read_csv("yield.csv",";")
    #Выведем первые 5 значений
    data.head()
    print(data.head())

    #Импортируем библиотеки для построения
линейного графика
    import matplotlib.pyplot as plt

    # Зададим переменные для построения графика
    t = data['year']
    s = data['yield']

    #Создадим область графика и оси
    fig, ax = plt.subplots()
    ax.plot(t, s)

    #Зададим название графика, осей и сетку
основной области
    ax.set(xlabel='Годы', ylabel='Урожайность',
           title='Урожайность зерновых в России,
ц/га')
    ax.grid()

    #Сохраним рисунок в требуемом формате и
выведем рисунок
    fig.savefig("test.png")
    plt.show()
```

```
    #Импортируем функцию и библиотеку для
построения диаграммы
    from pandas.plotting import lag_plot
    from matplotlib import pyplot

    #Построим диаграмму
    lag_plot(data['yield'])
    pyplot.show()

    #Импортируем необходимые функции и библиотеки
    from pandas import DataFrame
    from pandas import concat

    #Преобразуем данные и зададим переменные для
построения корерляционной матрицы
    values = DataFrame(data["yield"].values)
    dataframe = concat([values.shift(1), values],
axis=1)
    dataframe.columns = ['t-1', 't+1']
```

```
    #Построим матрицу и выведем результат
    result = dataframe.corr()
    print(result)

    #Импортируем функцию autocorrelation_plot и
построим график
    from          pandas.plotting          import
autocorrelation_plot
    autocorrelation_plot(data['yield'])
    pyplot.show()

    #Импортируем функцию plot_acf и построим
столбиковую диаграмму
    from   statsmodels.graphics.tsaplots   import
plot_acf
    plot_acf(data["yield"], lags=25)
    pyplot.show()

    # Импортируем библиотеки и создадим модель
авторегрессии
    from statsmodels.tsa.ar_model import AutoReg
    from          sklearn.metrics          import
mean_squared_error
    from math import sqrt

    # Сформируем лаговые данные
    X = data['yield'].values
    train, test = X[1:len(X)-7], X[len(X)-7:]

    # Построим модель авторегрессии
    model = AutoReg(train, lags=7)
    model_fit = model.fit()
    print('Коэффициенты   авторегрессии:   %s'   %
model_fit.params)

    # Построим прогноз
```

```
    predictions                                    =
model_fit.predict(start=len(train),
end=len(train)+len(test)-1, dynamic=True)
    for i in range(len(predictions)):
        print('прогнозируемые=%f, ожидаемые=%f' %
(predictions[i], test[i]))
    rmse     =     sqrt(mean_squared_error(test,
predictions))
    print('RMSE: %.3f' % rmse)

    # Выведем график прогноза
    pyplot.plot(test)
    pyplot.plot(predictions, color='red')
    plt.xlabel('Номер   прогнозируемого   периода
(года)')
    plt.ylabel('Урожайность')
    pyplot.show()
```

Приложение В

Листинг модуля кластерного анализа

```
    #Установим библиотеки и загрузим данные
    from pandas import read_csv
    import matplotlib.pyplot as plt
    import numpy as np
    from scipy.spatial import distance as
sci_distance
    from sklearn import cluster as sk_cluster
    data = read_csv('Clusters.csv',';')
    data.head()

    #определение количества кластеров
    cdata = data
    K = range(1, 20)
    KM =
(sk_cluster.KMeans(n_clusters=k).fit(cdata) for
k in K)
    centroids = (k.cluster_centers_ for k in KM)
    D_k = (sci_distance.cdist(cdata, cent,
'euclidean') for cent in centroids)
    dist = (np.min(D, axis=1) for D in D_k)
    avgWithinSS = [sum(d) / cdata.shape[0] for d
in dist]
    plt.plot(K, avgWithinSS, 'b*-')
    plt.grid(True)
    plt.xlabel('Количество кластеров')
    plt.ylabel('Средняя сумма квадратов в
кластере')
    plt.show()

    #Кластерный анализ методом k-средних
    n_clusters = 5#определенное ранее количество
кластеров
    means_cluster =
sk_cluster.KMeans(n_clusters=n_clusters,
random_state=111)
```

```
    columns = ["X11", "X12", "X13", "X14", "X15",
"X16", "X17", "X18", "X19"]
    est = means_cluster.fit(data[columns])
    clusters = est.labels_
    data['cluster'] = clusters

    #Загрузим требуемые библиотеки и выведем
информацию о кластерах
    import pandas as pd
    import tkinter as tk
    from tkinter import filedialog
    for c in range(n_clusters):
        cluster_members=data[data['cluster']   ==
c][:]
        print('Cluster{}(n={}):'.format(c,
len(cluster_members)))
    print(data.groupby(['cluster']).mean())

    #сохраним средние по кластерам в excel
    a = data.groupby(['cluster']).mean()
    df = pd.DataFrame(a)
    root= tk.Tk()
```

```
    c1 = tk.Canvas(root, width = 300, height =
300, bg = 'lightsteelblue2', relief = 'raised')
    c1.pack()

    def exportExcel ():
        global df
        export_file_path                          =
filedialog.asksaveasfilename(defaultextension='.
xlsx')
        df.to_excel  (export_file_path,  index  =
False, header=True)
    saveAsButtonExcel  =  tk.Button(text='Export
Excel',     command=exportExcel,     bg='green',
fg='white', font=('helvetica', 12, 'bold'))
    c1.create_window(150,                        150,
window=saveAsButtonExcel)
    root.mainloop()
```

Приложение Г

Листинг модуля регрессионного анализа

```
    #Загрузим csv-файл с данными для анализа
    from pandas import read_csv
    data = read_csv('Analysis.csv',';')
    data.head()

    #Построим матрицу корреляции в виде тепловой
карты
    import seaborn as sns
    import numpy as np
    matrix = np.triu(data.corr())
    sns.heatmap(data.corr(),          annot=True,
mask=matrix, fmt='.1g')

    import matplotlib.pyplot as plt
    #определим переменные y и x
    x = data['X6']
    y = data['Y1']
    # Точечная диаграмма (рассеияния)
    xyplot = plt.figure()
    plt.scatter(x,
     y,
     color = 'darkgreen')
    plt.title('Зависимость   рентабельность   от
удельного веса животноводства')
    plt.xlabel('Удельный вес животноводства')
    plt.ylabel('Рентабельность, %')
    plt.show()

    #Построение модели регрессии
    import statsmodels.api as sm
    #определим переменные x и y
    x1 = data[['X3', 'X6', 'X8']]
    y1 = data['Y1']
    x1 = sm.add_constant(x1)
```

```
    # Построим модель на основе метода наименьших
квадратов
    model = sm.OLS(y1, x1).fit()
    predictions = model.predict(x1)
    # Сформируем вывод
    model.summary()
    print(model.summary())
```

Листинг веб-приложения

```
    #Импортируем библиотеки для загрузки данных и
анализа данных
    import streamlit as st
    import pandas as pd
    #название первой страницы приложения
    st.write("""
    # Веб-приложение "Статистический анализ".
    Авторегрессия, кластерный и регрессионный
анализ!
    """)
    #зададим левую колонку в приложении
    st.sidebar.header('Вид и параметры анализа')
    #вид анализа
    option = st.sidebar.selectbox('Выбрать вид
анализа:', ('Авторегрессия', 'Кластерный',
'Регрессия'))
    #Загрузим данные в виде переменной data
    data = pd.read_csv("yield.csv",";")
    #зададим переменную, содержащую входные
значения
    df = data
    #пропишем название параметров на лицевой
страничке приложения и выведем значения этих
параметров
    st.subheader('Входные параметры
прогнозирования')
    st.write(df)
    #Импортируем библиотеки для построения
линейного графика
    import matplotlib.pyplot as plt
    # Зададим переменные для построения графика
    t = data['year']
    s = data['yield']
    #Создадим область графика и оси
    fig, ax = plt.subplots()
    ax.plot(t, s)
```

```
    #Зададим название графика, осей и сетку
основной области
    ax.set(xlabel='Годы', ylabel='Урожайность',
           title='Урожайность зерновых, ц/га')
    ax.grid()
    #Сохраним рисунок в требуемом формате и
выведем рисунок
    fig.savefig("test.png")
    plt.show()
    #Импортируем функцию и библиотеку для
построения диаграммы
    from pandas.plotting import lag_plot
    from matplotlib import pyplot
    #Построим диаграмму
    lag_plot(data['yield'])
    pyplot.show()
    #Импортируем необходимые функции и библиотеки
    from pandas import DataFrame
    from pandas import concat
    #Преобразуем данные и зададим переменные для
построения корреляционной матрицы
    values = DataFrame(data["yield"].values)
    dataframe = concat([values.shift(1), values],
axis=1)
    dataframe.columns = ['t-1', 't+1']
    #Построим матрицу и выведем результат
    result = dataframe.corr()
    #Импортируем функцию autocorrelation_plot и
построим график
    from          pandas.plotting          import
autocorrelation_plot
    autocorrelation_plot(data['yield'])
    pyplot.show()
    #Импортируем функцию plot_acf и построим
столбиковую диаграмму
    from   statsmodels.graphics.tsaplots   import
plot_acf
    plot_acf(data["yield"], lags=25)
    pyplot.show()
```

```
    # Импортируем библиотеки и создадим модель
авторегрессии
    from statsmodels.tsa.ar_model import AutoReg
    from          sklearn.metrics          import
mean_squared_error
    from math import sqrt
    # Сформируем лаговые данные
    X = data['yield'].values
    train, test = X[1:len(X)-7], X[len(X)-7:]
    # Построим модель авторегрессии
    model = AutoReg(train, lags=7)
    model_fit = model.fit()
    # Построим прогноз
    predictions                                  =
model_fit.predict(start=len(train),
end=len(train)+len(test)-1, dynamic=False)
    for i in range(len(predictions)):
        print('прогнозируемые=%f, ожидаемые=%f' %
(predictions[i], test[i]))
    rmse     =     sqrt(mean_squared_error(test,
predictions))
    print('RMSE: %.3f' % rmse)
    # Выведем график прогноза
    pyplot.plot(test)
    pyplot.plot(predictions, color='red')
    plt.xlabel('Номер   прогнозируемого   периода
(года)')
    plt.ylabel('Урожайность')
    pyplot.show()
    #пропишем вывод линейного графика
    st.subheader('График динамики показателя')
    st.line_chart(data['milk'])
    #пропишем вывод диаграммы рассеяния
    st.subheader('Диаграмма рассеяния')
    st.write()
    #Матрица коэффициентов корреляции
    st.subheader('Матрица           коэффициентов
корреляции')
    st.write(result)
```

```
    #Автокорреляционная функция
    st.subheader('Автокорреляционная функция')
    st.write(pyplot.show())
    #пропишем название таблицы и зададим значения
выводимых параметров в приложении
    st.subheader('Параметры                модели
авторегрессии')
    st.write('Коэффициенты авторегрессии: %s' %
model_fit.params)
    #зададим прогнозируемые и ожидаемые значения
    st.subheader('Прогнозируемые   и   ожидаемые
значения')
    for i in range(len(predictions)):
        st.write('прогнозируемые=%f,
ожидаемые=%f' % (predictions[i], test[i]))
    # Выведем график прогноза
    st.subheader('График прогнозных и ожидаемых
значений')
    st.write()
    #выведем точность модели
    st.subheader('Точность модели')
    st.write(rmse)
```

Демонстрация работы модуля проведения корреляционно-регрессионного анализа

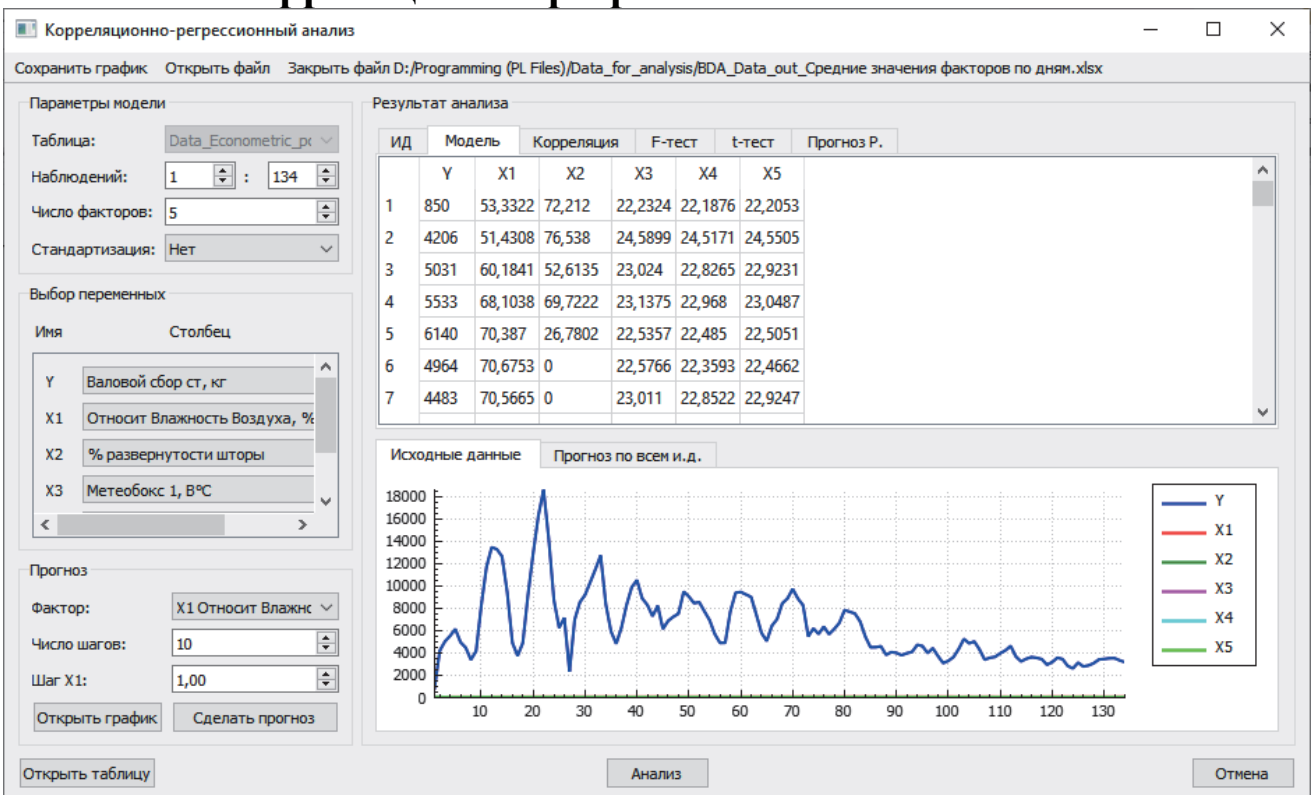


Рисунок Е.1 – Форма «КРА». Нажатие на кнопку «Анализ», проведение корреляционно-регрессионного анализа по первым пяти факторам набора данных из файла Microsoft Excel

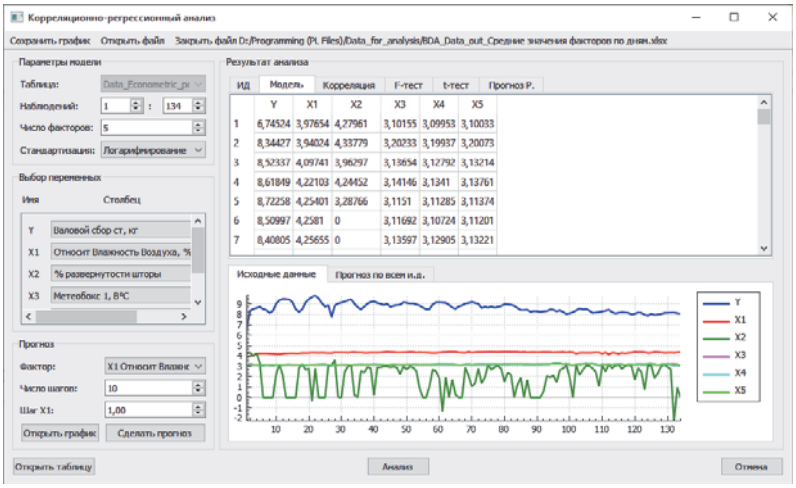


Рисунок Е.2 – Форма «КРА». Выбор стандартизации «Логарифмирование», нажатие на кнопку «Анализ», проведение нового корреляционно-регрессионного анализа по стандартизированным исходным данным

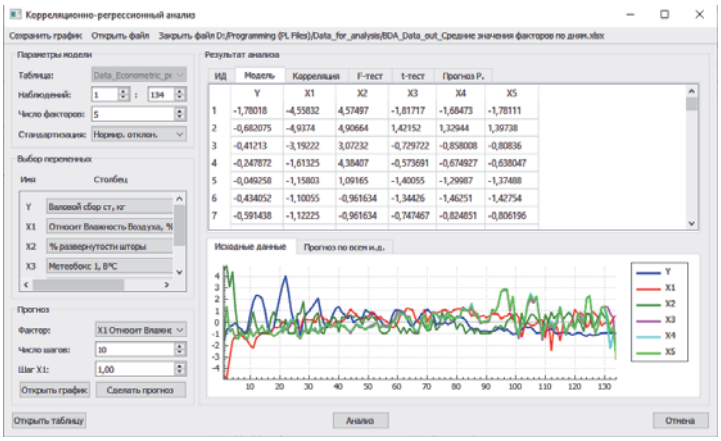
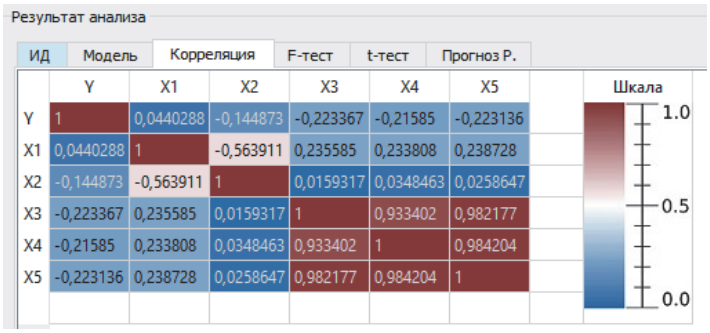


Рисунок Е.3 – Форма «КРА». Выбор стандартизации «Нормированное отклонение», нажатие на кнопку «Анализ», проведение нового корреляционно-регрессионного анализа по стандартизированным исходным данным



	Y	X1	X2	X3	X4	X5
Y	1	0,0440288	-0,144873	-0,223367	-0,21585	-0,223136
X1	0,0440288	1	-0,563911	0,235585	0,233808	0,238728
X2	-0,144873	-0,563911	1	0,0159317	0,0348463	0,0258647
X3	-0,223367	0,235585	0,0159317	1	0,933402	0,982177
X4	-0,21585	0,233808	0,0348463	0,933402	1	0,984204
X5	-0,223136	0,238728	0,0258647	0,982177	0,984204	1

Рисунок Е.4 – Форма «КРА». Результаты корреляционно-регрессионного анализа по стандартизированным исходным данным. Вкладка таблицы «Корреляция»

	YT	X1T	X2T	X3T	X4T	X5T
1	-1,01569	0,692078	-0,250816	0,821455	0,763995	0,8054
2	-1,03074	0,702331	-0,254532	0,833624	0,775313	0,817332
3	-1,04579	0,712584	-0,258247	0,845794	0,786632	0,829264
4	-1,06084	0,722837	-0,261963	0,857964	0,79795	0,841196
5	-1,07588	0,73309	-0,265679	0,870134	0,809268	0,853128
6	-1,09093	0,743343	-0,269395	0,882303	0,820587	0,86506
7	-1,10598	0,753596	-0,273111	0,894473	0,831905	0,876991

Рисунок Е.5 – Форма «КРА». Результаты корреляционно-регрессионного анализа по стандартизированным исходным данным. Вкладка таблицы «Лин. тренды», вкладка графика «Прогноз факторов»

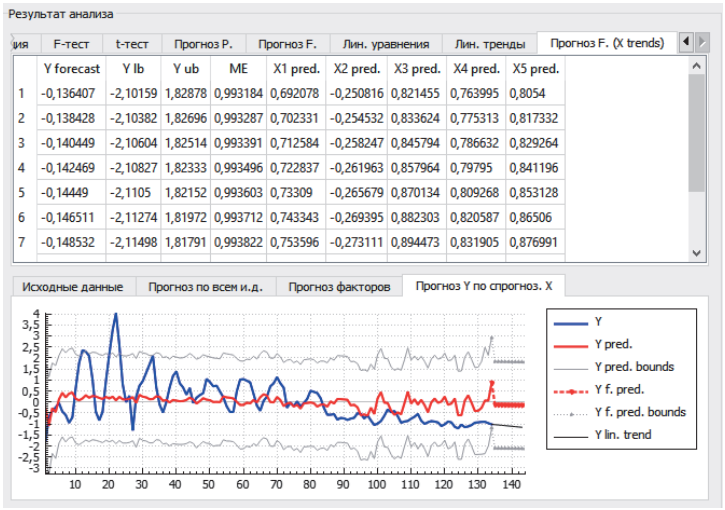


Рисунок Е.6 – Форма «КРА». Результаты корреляционно-регрессионного анализа по стандартизированным исходным данным. Вкладка таблицы «Прогноз F (X trends)», вкладка графика «Прогноз факторов»

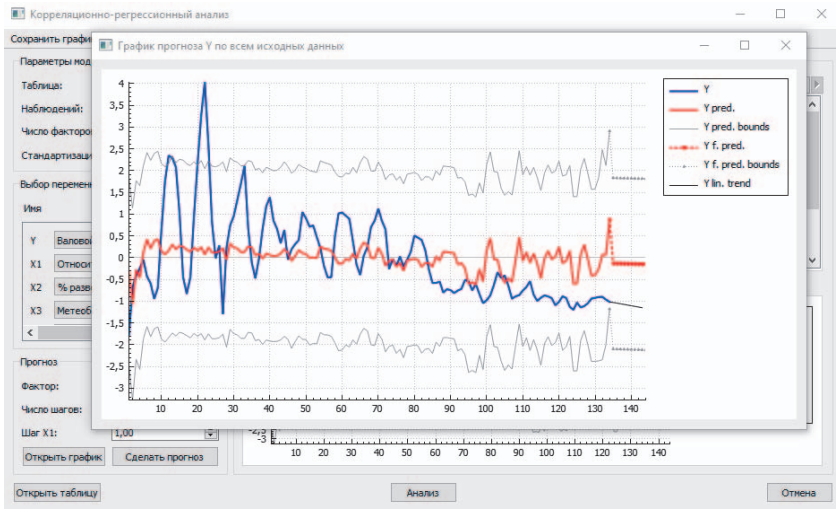


Рисунок Е.7 – Форма «КРА». Нажатие на кнопку «Открыть график», открытие графика в отдельном окне

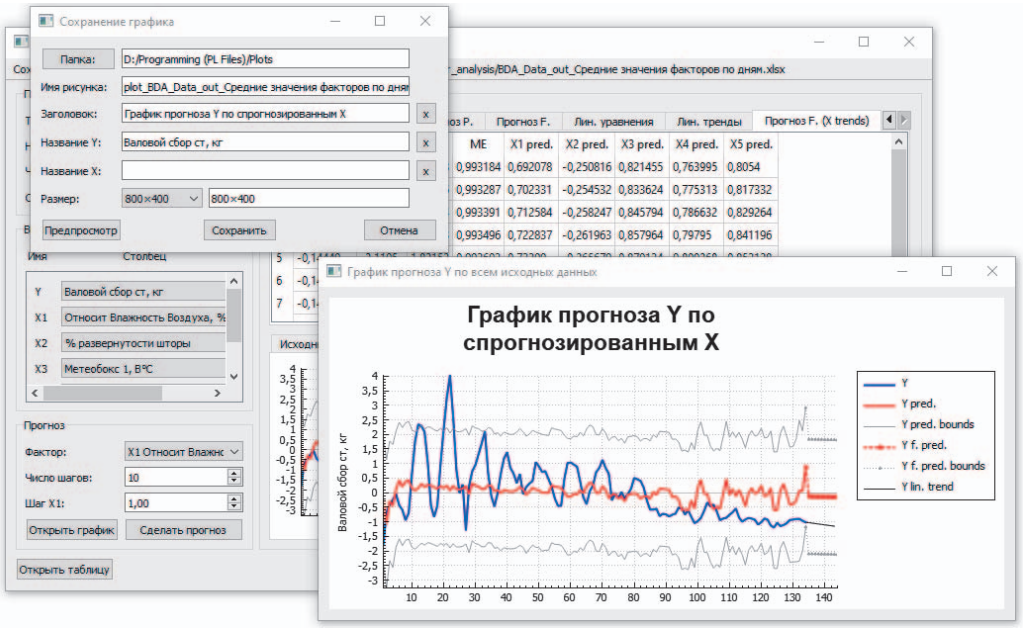


Рисунок Е.8 – Форма «КРА». Нажатие на кнопку «Сохранить график», открытие формы сохранения графика с автоматически заполненными полями. Нажатие на кнопку «Предпросмотр», открытие графика в отдельном окне с заданными параметрами

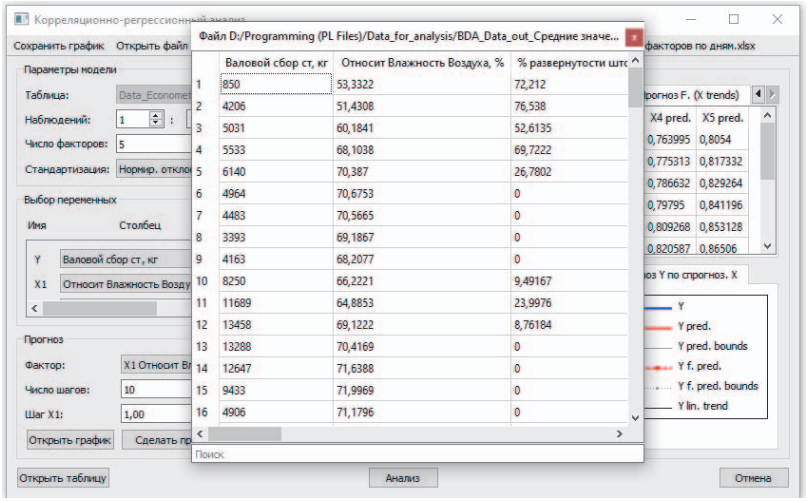


Рисунок Е.9 – Форма «КРА». Нажатие на кнопку «Открыть таблицу», открытие формы с таблицей исходных данных

Научное издание

Зинченко А.П., Уколова А.В., Демичев В.В., Тихонова А.В., Романцева Ю.Н., Харитонова А.Е., Быков Д.В., Ульянкин А.Е., Невзоров А.С., Сидоров А.И., Титов А.Д.

ЦИФРОВЫЕ ТЕХНОЛОГИИ АНАЛИЗА ДАННЫХ В СЕЛЬСКОМ ХОЗЯЙСТВЕ

монография

Сдано в набор 24.11.2022. Подп. в печ. 01.12.2022.
Формат 60×88/16. Бумага офсетная.
Усл.печ.л. 16,3 Тираж 500 экз.

Издательство «Научный консультант» предлагает авторам: издание рецензируемых сборников трудов научных конференций; печать монографий, методической и иной литературы

ISBN 978-5-907477-96-4

Издательство Научный консультант
123007, г. Москва, Хорошевское ш., 35к2, офис 508.
Тел.: +7 (926) 609-32-93, +7 (499) 195-60-77 www.n-ko.ru keyneslab@gmail.com