МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ

РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное автономное образовательное учреждение  
высшего образования

«Южно-Уральский государственный университет

(национальный исследовательский университет)»

Высшая школа электроники и компьютерных наук

Кафедра системного программирования

**ОТЧЕТ**

**по производственной практике**

**(научно-исследовательская работа)**

|  |  |
| --- | --- |
|  | Выполнил, студент группы КЭ-403  Разуев Г.А.  Научный руководитель,  доц.каф.СП  Долганина Н.Ю.  Руководитель практики:  доц.каф.СП, к.ф.-м.н.\_\_  Турлакова С.У.\_\_\_\_\_\_\_  Дата: \_\_\_\_\_\_\_\_\_\_\_\_\_  Оценка: \_\_\_\_\_\_\_\_\_\_\_\_\_ |

Челябинск, 2025 г.

Министерство науки и высшего образования Российской Федерации

Южно-Уральский государственный университет

Кафедра системного программирования

УТВЕРЖДАЮ

Зав. кафедрой

системного программирования

\_\_\_\_\_\_\_\_\_\_\_\_\_ Л.Б. Соколинский

**ЗАДАНИЕ**

**на производственную практику**

**(научно-исследовательскую работу)**

1. **Тема работы**

Разработка модулей обработки данных, кластеризации пользователей и анализа продуктовых метрик и воронок библиотеки для продуктовой аналитики.

1. **Исходные данные к работе**
   1. Lean Analytics: Use Data to Build a Better Startup Faster [Электронный ресурс] URL: https://www.oreilly.com/library/view/lean-analytics/ (дата обращения: 03.02.2025 г.).
   2. Scikit Learn Documentation. [Электронный ресурс] URL: https://scikit-learn.org/stable/index.html (дата обращения: 03.02.2025 г.).
   3. Pandas Documentation. [Электронный ресурс] URL: https://pandas.pydata.org/docs/ (дата обращения: 03.02.2025 г.).
2. **Перечень подлежащих разработке вопросов**
   1. Разработать модуль обработки данных.
   2. Разработать модуль кластеризации пользователей.
   3. Разработать модуль анализа продуктовых метрик и воронок.
3. **Сроки**

Дата выдачи задания: 3 февраля 2025 г.

Срок сдачи законченной работы: 28 февраля 2025 г.

**Руководитель практики со стороны ЮУрГУ:**

Доцент кафедры СП,к.ф.-м.н. \_\_\_\_\_\_\_\_\_\_\_\_\_ Турлакова С.У.

подпись ФИО ответственного

**Научный руководитель практики:**

Доцент кафедры СП \_\_\_\_\_\_\_\_\_\_\_\_\_ Долганина Н.Ю.

подпись ФИО научного руководителя

**Задание принял к исполнению:**

\_\_\_\_\_\_\_\_\_\_\_\_\_ Разуев Г.А.

подпись ФИО студента

**ОГЛАВЛЕНИЕ**

[ВВЕДЕНИЕ 6](#_Toc168239191)

[1. АНАЛИЗ ПРЕДМЕТНОЙ ОБЛАСТИ 8](#_Toc168239192)

[1.1. Предметная область проекта 8](#_Toc168239193)

[1.2. Обзор аналогов 9](#_Toc168239194)

[1.3. Анализ существующих решений 13](#_Toc168239195)

[1.4. Заключение 14](#_Toc168239196)

[2. ПРОЕКТИРОВАНИЕ 15](#_Toc168239197)

[2.1. Анализ требований к системе 15](#_Toc168239198)

[2.1.1. Функциональные требования 15](#_Toc168239199)

[2.1.2. Нефункциональные требования 15](#_Toc168239200)

[2.2. Диаграмма вариантов использования 16](#_Toc168239201)

[2.3. Проектирование интерфейса 17](#_Toc168239202)

[2.4. Проектирование базы данных 18](#_Toc168239203)

[2.5. Архитектура приложения 20](#_Toc168239204)

[3. РЕАЛИЗАЦИЯ 23](#_Toc168239205)

[3.1. Реализация алгоритма формирования портфолио 23](#_Toc168239206)

[3.1.1. Сбор и предобработка данных 23](#_Toc168239207)

[3.1.2. Расчет целевой метрики 27](#_Toc168239208)

[3.1.3. Обучение и анализ результатов различных моделей машинного обучения 29](#_Toc168239209)

[3.1.4. Формирования ансамбля моделей 35](#_Toc168239210)

[3.1.5. Тестирование разработанного алгоритма 37](#_Toc168239211)

[3.2. Реализация серверной части приложения 39](#_Toc168239212)

[3.2.1. Создание проекта 39](#_Toc168239213)

[3.2.2. База данных 41](#_Toc168239214)

[3.2.3. Маршрутизатор 42](#_Toc168239215)

[3.2.4. Реализация компонентов, отвечающих за обновление данных 44](#_Toc168239216)

3.2.5. Реализация компонентов, отвечающих за расписание обновлений 46

[3.3. Реализация клиентской части приложения 47](#_Toc168239218)

[3.3.1. Организация исходного кода 47](#_Toc168239219)

[3.3.2. Взаимодействие с сервером 48](#_Toc168239220)

[4. ТЕСТИРОВАНИЕ 50](#_Toc168239221)

[4.1.1. Кроссбраузерное тестирование 50](#_Toc168239222)

[4.1.2. Функциональное тестирование 51](#_Toc168239223)

[ЗАКЛЮЧЕНИЕ 53](#_Toc168239224)

[ЛИТЕРАТУРА 54](#_Toc168239225)

[ПРИЛОЖЕНИЯ 56](#_Toc168239226)

[Приложение А. Листинги исходного кода 56](#_Toc168239227)

[Приложение Б. Скриншоты реализованного веб приложения 58](#_Toc168239228)

ВВЕДЕНИЕ

**Актуальность**

Современный рынок IT-продуктов и сервисов характеризуется высокой конкуренцией между компаниями за каждого пользователя. В такой привлечение новых пользователей и удержание существующих становятся критически важными задачами для бизнеса. Одним из ключевых инструментов, помогающих достичь этих целей, является анализ пользовательского опыта.

Разработка библиотеки для анализа действий пользователей в IT-продукте предоставляет аналитикам, и, как следствие, бизнесу мощный инструмент для глубокого понимания потребностей своих клиентов. Этот инструмент позволяет исследовать поведение пользователей, выявлять проблемные места в продукте и оптимизировать пользовательский опыт. В результате, продукт становится более удобным, функциональным и привлекательным, что в свою очередь способствует увеличению удовлетворенности и лояльности пользователей.

Одним из ключевых преимуществ использования такой библиотеки является ее способность значительно сократить время работы аналитиков над исследованиями. Вместо того чтобы вручную проделывать огромное количество манипуляций с большими наборами данных, аналитики могут использовать автоматизированные инструменты для быстрого и точного анализа. Это позволяет им сосредоточиться на более сложных задачах, таких как разработка стратегий улучшения продукта и принятие решений на основе данных.

Таким образом разрабатываемый продукт способен значительно сократить затраты времени аналитиков, позволив им сосредоточиться на более глобальных задачах, при этом предоставить исчерпывающие данные для принятия бизнес-решений на их основе. В итоге использование разрабатываемого инструмента позволяют компании улучшать свои продукты и услуги, более точно прогнозировать потребности клиентов, оптимизировать маркетинговые кампании и предлагать персонализированные предложения. Это позволит им не только удерживать существующих клиентов, но и привлекать новых, что способствует росту целевых показателей бизнеса и развитию компании.

**Постановка задачи**

Целью выпускной квалификационной работы является разработка библиотеки для исследования пользовательского опыта. Для достижения поставленной цели необходимо решить следующие задачи:

1. провести анализ литературы;
2. спроектировать библиотеку;
3. разработать библиотеку;
4. осуществить тестирование библиотеки;
5. написать документацию библиотеки;
6. опубликовать библиотеку на GitHub.

**Структура и содержание работы**

Работа состоит из введения, четырех глав, заключения, списка литературы и двух приложений. Объем работы составляет 59 страниц, объем списка литературы – 15 источников.

# АНАЛИЗ ПРЕДМЕТНОЙ ОБЛАСТИ

## Предметная область проекта

В данной выпускной квалификационной работе предметная область охватывает концепции и методологии, связанные с анализом пользовательского опыта в контексте информационных технологий. С учетом высокой конкуренции на рынке IT-продуктов и сервисов, исследование фактического поведения пользователей становится критически важным для принятия обоснованных бизнес-решений.

Пользовательский опыт (UX) — это восприятие и реакция пользователя на взаимодействие с продуктом или сервисом. Это включает в себя не только функциональные аспекты, такие как удобство и доступность, но и эмоциональные, связанные с ощущениями и удовлетворенностью пользователя.

## Обзор аналогов

Существует достаточно много инструментов для частичной автоматизации анализа данных, это обусловлено тем, что любой бизнес имеет потребность в проведении исследований, которые послужат основанием для принятия глобальных решений, а любой аналитик заинтересован в оптимизации собственной деятельности. Однако большая часть этих инструментов предоставляют достаточно скудный функционал, автоматизирую только узкий спектр задач. Далее будут рассмотрены 3 аналога, каждый из которых является популярной библиотекой, которая оптимизирует широкий круг задач по анализу данных.

Первым рассматриваемым аналогом является библиотека «Sweetviz» [1]. это библиотека Python с открытым исходным кодом, которая упрощает процесс разведочного анализа данных, создавая красивые визуализации и сводные отчёты всего несколькими строками кода. Она позволяет быстро получить представление о наборе данных, сравнить наборы данных и проанализировать целевые значения. Скриншот примера отчета, составленного библиотекой представлен на рисунке 1.

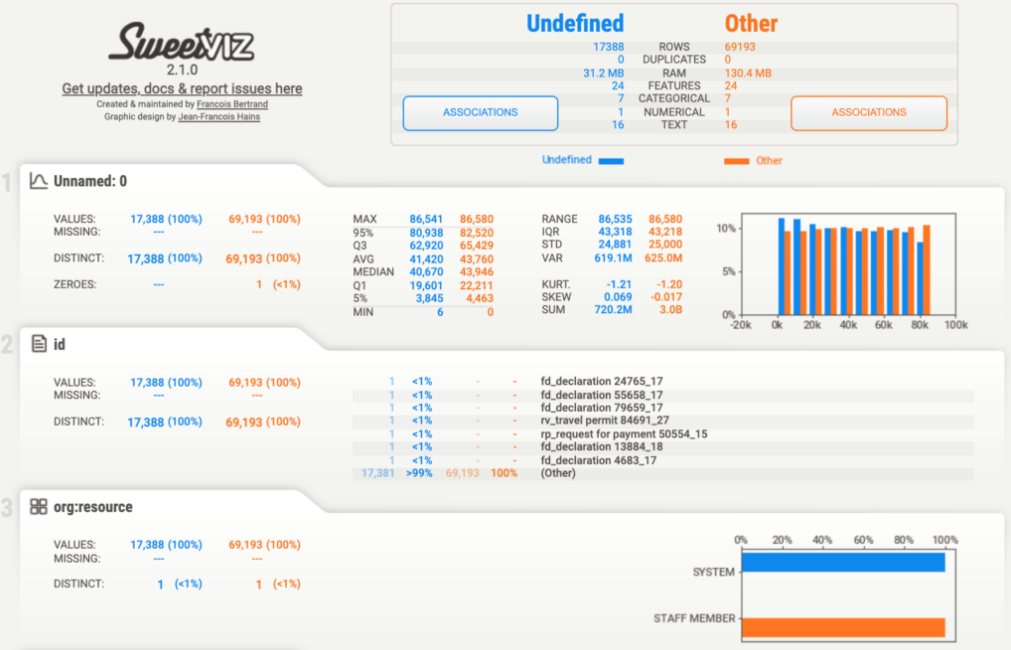


Рисунок 1 – Скриншот отчета «Sweetviz»

Основные выявленные достоинства и недостатки библиотеки приведены в таблице 1.

Таблица 1 – Преимущества и недостатки «Sweetviz»

|  |  |
| --- | --- |
| Преимущества | Недостатки |
| Возможность интерактивного исследования отчета при помощи интуитивно понятного интерфейса | Отсутствие возможности детальной настройки отчета |
| Возможность проводить первичное исследование, выявление зависимостей и аномалий | Отсутствие возможности предобрабатывать данные |
| Возможность получения широкого функционала с использованием очень небольшого объема кода | Отсутствие инструментов конкретно для анализа действий пользователя |
| Возможность делиться интерактивным отчетом с заказчиками анализа |  |

Следующим рассматриваемым аналогом является «KLib» [2]. это библиотека Python, которая предоставляет возможности автоматического разведочного анализа данных (EDA) и профилирования данных. Она предлагает различные функции и визуализации для быстрого изучения и анализа наборов данных. Скриншот отчета, составленного данной библиотекой представлен на рисунке 2.

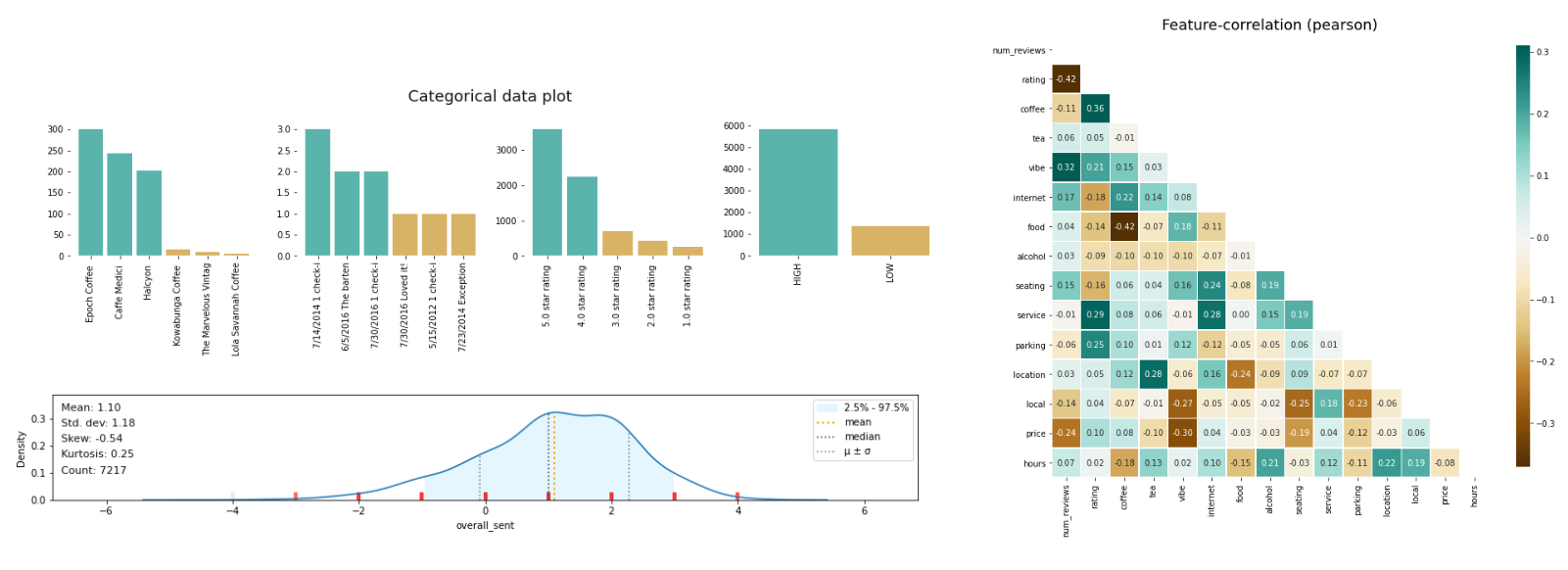


Рисунок 2 – Скриншот отчета «KLib»

Основные выявленные достоинства и недостатки сервиса приведены в таблице 2.

Таблица 2 – Достоинства и недостатки «KLib»

|  |  |
| --- | --- |
| **Достоинства** | **Недостатки** |
| Возможность составления отчета при помощи минимального объема кода | Отсутствие возможности детальной настройки отчета |
| Возможность проводить первичное исследование, выявление зависимостей и аномалий | Отсутствие возможности предобрабатывать данные |
|  | Отсутствие инструментов конкретно для анализа действий пользователя |

Последним и наиболее релевантным рассматриваемым аналогом является «Retentioneering» [3]. Это библиотека Python, которая значительно упрощает анализ потоков действий пользователей, пользовательских путей и журналов событий. Может использоваться для изучения поведения пользователей, сегментации пользователей и формирования гипотез о том, что побуждает пользователей к желаемым действиям или к отказу от продукта. Скриншот описания данной библиотеки представлен на рисунке 3.

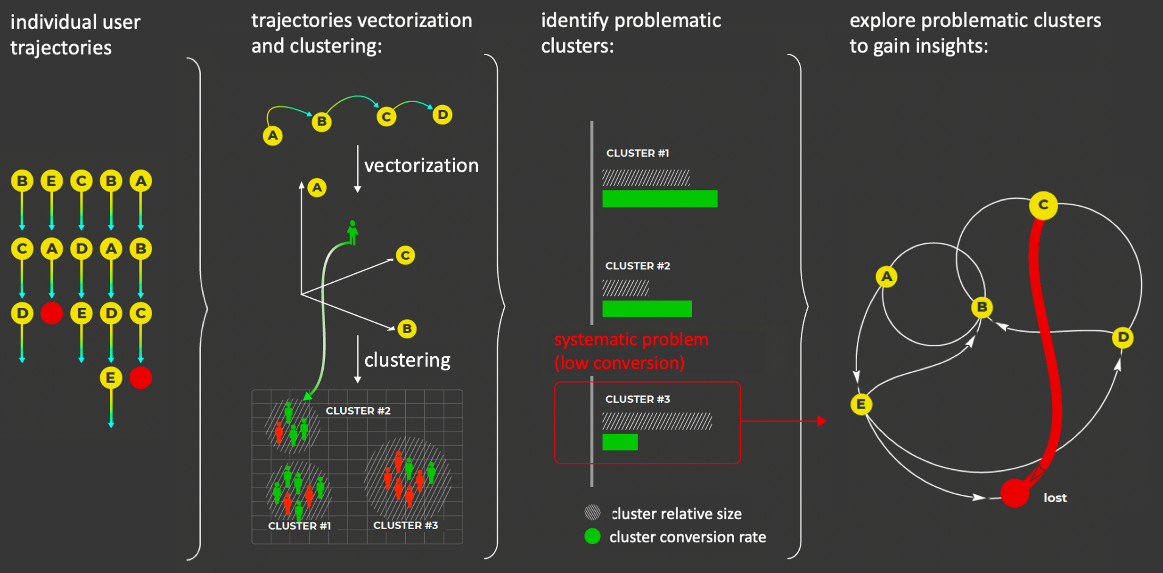


Рисунок 3 – Скриншот описания библиотеки «Retentioneering»

Основные выявленные достоинства и недостатки библиотеки приведены в таблице 3.

Таблица 3 – Достоинства и недостатки «Retentioneering»

|  |  |
| --- | --- |
| **Достоинства** | **Недостатки** |
| Возможность исследовать последовательности действий пользователей | Отсутствие возможности исследования отдельных сессий пользователя, только всего их пути в приложении |
| Возможность проводить первичное исследование, выявление зависимостей и аномалий | Отсутствие возможности получить легко интерпретируемую информацию о популярности пользовательских сценариев |
| Возможность получения широкого функционала с использованием очень небольшого объема кода | Отсутствие инструментов для выявления аномалий в метриках |
| Возможность предобрабатывать данные | Отсутствие инструментов для расчета популярных продуктовых метрик |
| Возможность выявлять сегменты пользователей |  |

Обзор аналогов показал, что большая часть существующих инструментов оптимизируют только первичное исследование данных и выявление поверхностных закономерностей, не предоставляя функционал для исследования конкретно пользовательского опыта. А единственная библиотека, предназначенная именно для анализа действий пользователя в продукте, имеет ряд ограничений. Однако анализ также выявил ряд присущих всем преимуществ, которые должны быть и в разрабатываемом продукте, среди них: возможность проводить первичное исследование, выявление зависимостей и аномалий, возможность предобработки данных, возможность получения широкого функционала с использованием небольшого объема кода. Среди недостатков, которые стоит предусмотреть: отсутствие возможности детальной настройки исследования, отсутствие инструментов для расчета популярных продуктовых метрик.

Таким образом для того, чтобы разрабатываемая библиотека решала проблему оптимизации широкого спектра задач по исследованию пользовательского опыта необходимо учесть все достоинства и недостатки аналогов.

## Анализ существующих решений

В качестве основы для разрабатываемого продукта будут использованы python библиотеки для работы с данными, являющиеся наиболее часто используемыми инструментами в области науки о данных:

1. «NumPy» – это библиотека для научных вычислений. Она используется для выполнения математических операций над большими объемами числовых данных, таких как линейная алгебра, матричные операции и работа с векторами.
2. «SciPy» – Эта библиотека включает модули для оптимизации, интегрирования, интерполяции, статистики, обработки сигналов и изображений, а также многие другие специализированные задачи..
3. «Pandas» – одна из самых популярных библиотек для анализа и манипуляции табличными данными.
4. «StatsModels» – библиотека для статистической модели и оценок. Отличается поддержкой широкого спектра статистических моделей и методов.
5. «Matplotlib» – основная библиотека для построения двумерных графиков в Python. Она поддерживает различные типы графиков, включая линии, гистограммы, круговые диаграммы, тепловые карты и многое другое.
6. «Seaborn» – построенная поверх Matplotlib библиотека для создания статичных и интерактивных графиков. Она ориентирована на более эстетичное представление данных и имеет встроенную поддержку различных типов статистических графиков.
7. «Plotly» – мощная библиотека для интерактивной визуализации данных. Поддерживает создание как статичных, так и динамических графиков через веб-интерфейсы.
8. «Scikit-Learn» – это библиотека для машинного обучения, включающая широкий спектр алгоритмов классификации, регрессии, кластеризации и снижения размерности. Также содержит инструменты для предобработки данных и оценки моделей

Для разработки будет использована среда программирования PyCharm 2019.3.3 Community Edition.

## Заключение

В ходе анализа предметной области был проведен обзор аналогов, в ходе которого были выявлены недостатки, которые нужно учесть и достоинства, которые необходимо воссоздать в разрабатываемом продукте.

Также был проведен анализ существующих решений, в результате которого, для реализации библиотеки было принято решение использовать библиотеки python для работы с данными, расчета статистических показателей и машинного обучения, а среду программирования PyCharm для разработки.

# ПРОЕКТИРОВАНИЕ

## Анализ требований к системе

После рассмотрения предметной области и обзора доступных средств разработки были выявлены функциональные и нефункциональные требования, которые должны быть учтены при разработке библиотеки.

### Функциональные требования

Функциональные требования необходимы для определения процессов, действий и операций, которые способна выполнить система. Функциональные требования используются для выражения поведения системы и определения ее функциональности. Исходя из назначения и анализа аналогов, разрабатываемая система должна соответствовать требованиям, которые представлены ниже.

1. Пользователь должен иметь возможность подготовить данные для исследования, т.е. при необходимости отсортировать события в логическом порядке, добавить разбиение на сессии, разметить существующие и добавить новые события и т.д.;
2. Библиотека должна предоставлять широкий спектр инструментов для анализа действий пользователей, среди которых визуализации последовательностей действий, расчет популярности пользовательских сценариев и т.д.;
3. Библиотека должна предоставлять инструменты для анализа продуктовых метрик, как наиболее популярных, так и пользовательских;
4. Библиотека должна предоставлять инструменты для анализа продуктовых воронок;
5. Библиотека должна предоставлять инструменты для сегментации пользователей на основе их действий и дополнительной информации.

### Нефункциональные требования

Нефункциональные требования служат для описания условий и качества разрабатываемой среды с целью создания системы с приемлемым качеством. В ходе рассмотрения инструментов для реализации системы, представленных в первой главе, а также из задач, которые решает разрабатываемая система, к ней были сформированы следующие нефункциональные требования.

1. Инструменты для манипуляций с данными – библиотеки Pandas и NumPy;
2. инструменты для визуализации данных – библиотеки Matplotlib, seaborn, Plotly;
3. инструменты для расчета статистических показателей – библиотеки StatsModels и SciPy;
4. Инструменты для машинного обучения – библиотека Scikit-learn

## Диаграмма вариантов использования

Для составления диаграммы вариантов использования был использован язык графического описания UML. На рисунке 4 представлена составленная диаграмма.

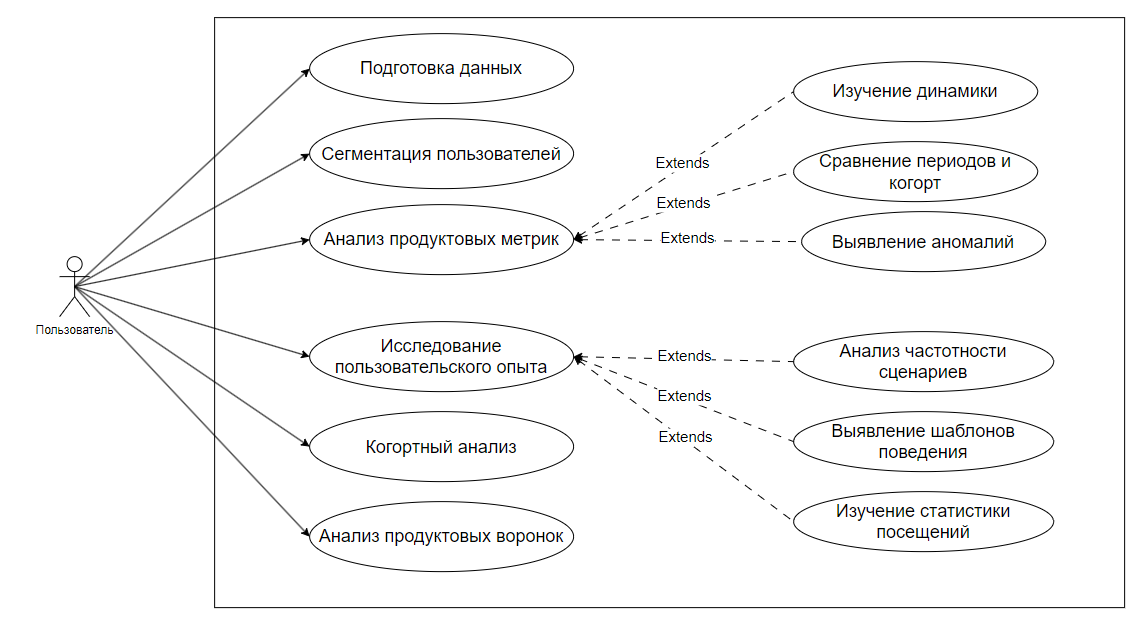


Рисунок 4 – Диаграмма вариантов использования

В разрабатываемой библиотеке основным актером, взаимодействующим с системой, является «Пользователь библиотеки» – разработчик, использующий библиотеку для анализа поведения пользователей в IT продукте.

## Архитектура библиотеки

Первый и один из самых ответственных этапов проектирования – создание архитектуры классов хранения данных, т.к. их будут использовать все остальные классы библиотеки. На рисунке 5 представлена диаграмма классов хранения данных.

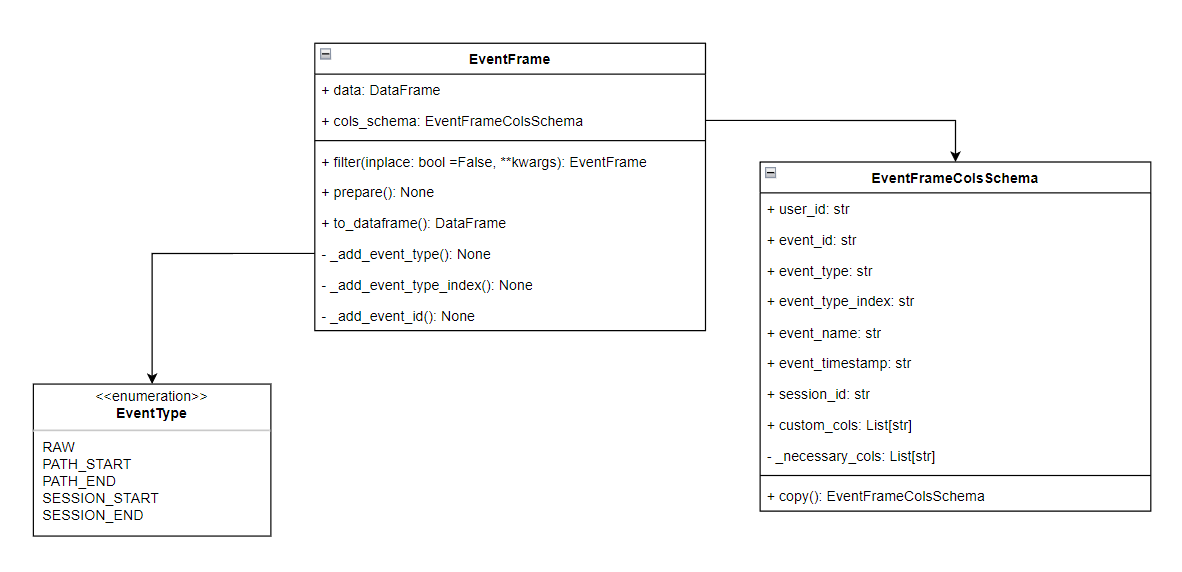


Рисунок – диаграмма классов хранения данных

Основной класс хранения данных – «EventFrame», он хранит сами данные в виде pandas.DataFrame, схему названий колонок в виде экземпляра класса «EventFrameColsSchema» и определяет несколько методов по преобразованию данных для дальнейшего их использования инструментами библиотеки. Класс «EventFrameColsSchema» предназначен для того, чтобы использовать данные с любыми названиями колонок, т.к. он хранит схему соответствия оригинальному названию колонки и ее смысловому содержанию.

Второй этап проектирования сосредотачивается на классах для обработки данных, который является не менее важным компонентом библиотеки. На рисунке 6 представлена диаграмма классов, отвечающих за предварительную обработку данных.

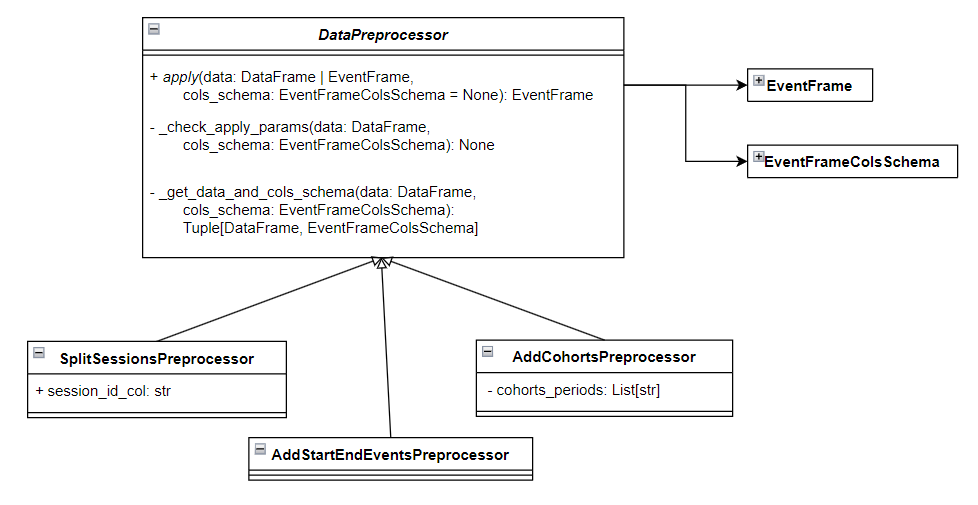
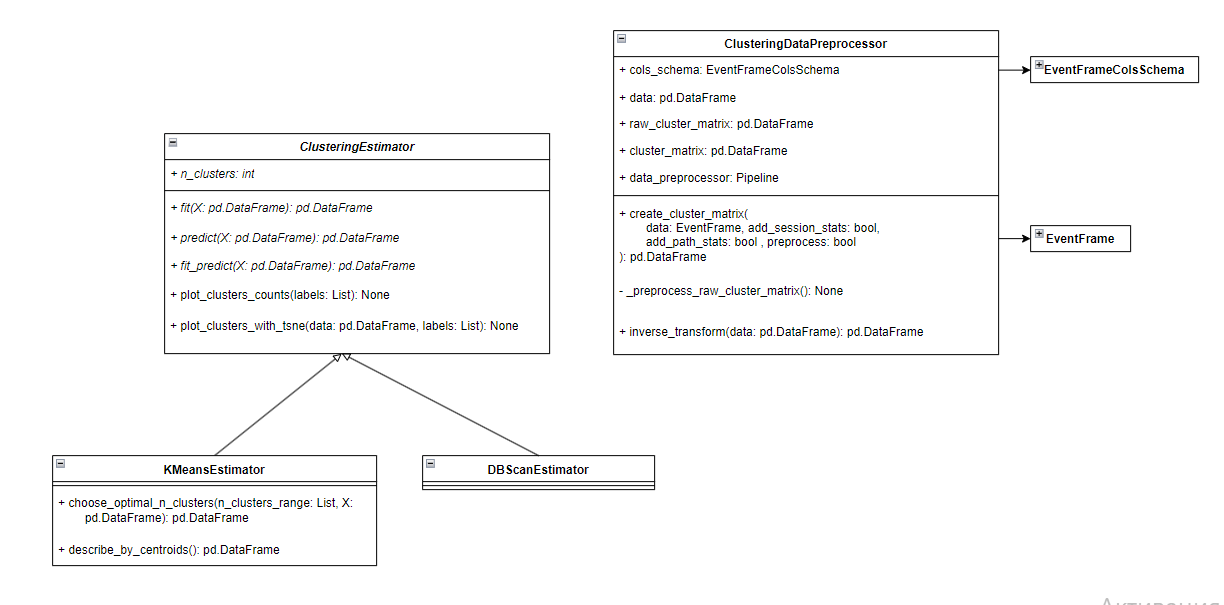


Рисунок - диаграмма классов, отвечающих за предварительную обработку данных

Абстрактным классом для обработки данных является «DataPreprocessor», который содержит абстрактный метод «apply» для применения различных предобработок к данным, хранящимся в классе EventFrame. Класс «DataPreprocessor» также включает в себя несколько ключевых приватных методов методов, обеспечивающие проверку параметров и получение данных в структурированном виде. Также на диаграмме представлены несколько конкретных реализаций абстрактного класса, каждый из которых предназначен для решения конкретной задачи предобработки.

Важно предоставить пользователю возможность решать задачи кластеризации как встроенными методами библиотеки, так и сторонними алгоритмами, поэтому при проектировании важно разделить функционал на независящие программные элементы. На рисунке НОМЕР\_РИСУНКА представлена диаграмма классов, отвечающих за кластеризацию



Подпись

Класс «ClusteringDataPreprocessor» используется исключительно для подготовки данных и извлечений из них статистик, а также приведения данных к исходному масштабу после проведения кластерного анализа. Этот класс никак не участвует в процессе обучения моделей для кластеризации, что позволяет пользователю выбирать средства кластеризации. Библиотека в свою очередь предлагает пользователю несколько таких средств – наследников абстрактного класса «ClusteringEstimator», каждый из которых реализует предоставляет набор методов для обучения моделей и предсказания меток кластеров.

Данная архитектура классов для обработки данных гарантирует гибкость и модульность, позволяя легко добавлять новые методы обработки и адаптировать библиотеку под изменяющиеся требования пользователя.

# РЕАЛИЗАЦИЯ

## Реализация алгоритма формирования портфолио

Алгоритм должен имитировать фундаментальный анализ показателей компаний, и на основании его результатов принимать решение о составе оптимального инвестиционного портфолио. Суть фундаментального анализа ценных бумаг заключается в оценке честной стоимости акций и выявление недооцененных бумаг с целью их приобретения. Честная цена акции - это цена, рассчитанная с учетом экономических факторов, которые влияют на стоимость бумаги: размера прибыли, состояния активов предприятия, рыночных перспектив. Иными словами, это та цена, на которую могли бы договориться продавец и покупатель, полностью осведомленные о положении компании на рынке.

Основная цель алгоритма – предсказывать изменение честной стоимости ценных бумаг после очередного выхода фундаментальных отчетов компаний. На сновании предсказанной честной стоимости алгоритм должен выявлять недооценённые компании и включать их в портфолио. Реализация алгоритма будет содержать следующие шаги:

1. Сбор и предобработка данных;
2. Расчет целевой метрики;
3. Обучение и анализ результатов различных моделей машинного обучения;
4. Формирование ансамбля лучших моделей машинного обучения;
5. Проведение исторических тестов инвестиционной стратегии разработанного алгоритма.

Далее будут подробно описаны каждый из приведенных шагов.

### Сбор и предобработка данных

Для реализации алгоритма будут использоваться акции крупных американских компаний, капитализация которых превышает 8 миллиардов долларов. Такой подход обусловлен тем, что показатели таких компаний менее волатильны, а цена акций в меньшей степени подвержена манипуляциям на рынке. Благодаря этим факторам большая часть изменений стоимости ценных бумаг будет обусловлена именно действиями и успехами компании, а не действиями инвесторов, преследующих собственные цели, а значит изменения цен можно с большей уверенностью использовать как целевую метрику для реализуемого алгоритма.

Для загрузки фундаментальных отчетов компаний будет использован API сервиса Yahoo Finance, который бесплатно предоставляет данные, преобразованные к единому формату.

Поскольку алгоритм должен осуществлять фундаментальный анализ показателей, из отчетов необходимо извлечь признаки, которыми руководствуются при проведении соответствующего типа анализа. Извлекаемые признаки приведены в списке ниже:

1. Ключевые показатели успешности компании, среди которых доход компании, чистая, валовая и операционная прибыль, дебетовая и кредитовая задолженности Компани, суммарная стоимость активов;
2. Основные мультипликаторы компаний, с которые могут свидетельствовать об успешности компании и об ее переоцененности или неоцененности. Примеры мультипликаторов: отношение чистой прибыли к выручке, отношение величины обязательств к акционерному капиталу и т.д.;
3. Расчет абсолютных и относительных изменений ключевых показателей относительно предыдущих периодов. Это очень важный параметр, который демонстрирует динамику положения дел в компании, а также дает возможность сделать вывод о том, насколько успешна деятельность компании в конкретном квартале относительно предыдущих. Помимо отчетов алгоритм должен принимать во внимание эконометрические показатели динамики стоимости конкретной бумаги. В первую очередь необходимо рассчитать прибыльность или убыточность бумаги, затем применить агрегирующие функции к временному ряду цен акции такие как минимум для выяснения максимальной просадки, стандартное отклонение для определения волатильности, расчет количества пиков, геометрическое среднее и т.д.

Отдельное внимание стоит уделить расчету коэффициентов Sharpe ratio и Sorting ratio, которые являются важными индикаторами для инвесторов, а значит и алгоритм должен на них ориентироваться.

Необходимость использования Sharp ratio обусловлена пониманием инвесторов того, что хорошо, когда доходности высокие и стабильные (низкая дисперсия) и плохо, когда в один день они могут получить +10%, а в следующей день -20%. Sharp ratio демонстрирует отношение средней доходности бумаги к стандартному отклонению динамики цен и рассчитывается на основе временных рядов прошлой доходности. Так если есть n периодов (месяцев, лет), то обозначив Rt доходность инвестиций в период t, отношение Шарпа будет рассчитываться по формуле (1)

|  |  |  |
| --- | --- | --- |
|  |  | (1) |
|  |  | (2) |

Отношение Шарпа измеряет риск стандартным отклонением. В формуле выборочного стандартного отклонения одинаково учитываются неожиданные отклонения от средней доходности R вверх и вниз. Между тем, большинство инвесторов ничего не имеют против случайных всплесков необычно высокой доходности, но очень переживают, когда доходность оказывается хуже ожиданий. Чтобы учесть эту особенность, можно использовать отношение Сортино (Sortino ratio). Для этого нам понадобится дополнительный параметр — минимально приемлемая доходность (minimum acceptable return, MAR). Этот уровень доходности может быть, как константой (например, 0%), так и переменной величиной (например, без рисковой ставки). В контексте данной работы величину MAR будем считать равной нолю, тогда Sortino ratio будет рассчитываться по формуле (3)

|  |  |  |
| --- | --- | --- |
|  |  | (3) |
|  |  | (4) |

Также необходимо рассчитать показатели, позволяющие сравнить динамику цен конкретной акции с динамикой всего рынка в целом, за который в контексте данной работы будет принят индекс S&P 500, включающий в себя акции 500 наиболее крупных компаний. Необходимость обусловлена тем, что сам по себе рынок иногда растет, а иногда падает, и прибыльность конкретной бумаги может быть обусловлена только ростом рынка. Для оценки связанности динамики конкретной бумаги с динамикой рынка будет применена модель оценки капитальных активов (CAPM), по сути эта модель демонстрирует взаимосвязь временного ряда цен конкретной акции от динамики всего рынка и описывается формулой (5)

|  |  |  |
| --- | --- | --- |
|  |  | (5) |

где – доходность конкретной бумаги;

– доходность всего рынка;

– коэффициент, демонстрирующий степень взаимосвязи доходности акции от доходности рынка.

– константа, демонстрирующая дополнительную доходность бумаги по отношению к рынку;

Со строгим определением и доказательством модели CAPM можно ознакомиться в книге Джона Линтнера «Оценка рискованных активов и выбор рискованных инвестиций в портфели акций и капитальные бюджеты» [10]. В ходе работы коэффициенты будут рассчитываться путем построения линейной регрессии, в которой независимой переменной будет выступать динамика цен рынка, а зависимой динамика цен конкретной бумаги.

Для нормализации численных данных будет использоваться алгоритм StandardScaler из библиотеки Scikit-Learn, которых рассчитывает нормализованное значение числа x по как разность числа x и среднего значения выборки, нормализованную на стандартное отклонение в выборке.

Для нормализации категориальных данных будет использоваться алгоритм OneHotEncoder из библиотеки Scikit-Learn, основывающийся на создании бинарных признаков, которые показывают принадлежность к уникальному значению.

Поскольку алгоритм будет представлять из себя ансамбль разных моделей машинного обучения, каждая из которых может по-разному относиться к наличию категориальных данных (например, CatBoost успешно использует категориальные признаки для предсказаний, а LightGBM умеет работать только с численными) были реализованы инструмент, преобразующие все данные исключительно к числовому типу, и инструмент, нормализующий числовые признаки, но оставляющий категориальные нетронутыми.

### Расчет целевой метрики

Главная цель алгоритма – предсказывать то, как должна измениться цена акции конкретной компании по сравнению с предыдущим кварталом после выхода очередных фундаментальных отчетов, а значит в качестве целевой метрики необходимо брать изменение стоимости акции по сравнению с предыдущим отчетным периодом, однако брать фиксированную цену в день выхода отчетов может оказаться не очень корректно, т.к. в день выпуска отчетов цена может сильно колебаться.

Изменения цены, вызванные именно отчетами компании должны длиться несколько дней непосредственно после выпуска отчетов, дальнейшие колебания будут уже скорее обусловлены действиями компании и отдельных инвесторов, поэтому в качестве цены, которая будет учитываться целевой метрикой следует взять среднее значение от показателей цены в ближайшие 10 дней после выпуска фундаментальных отчетов.

Также уместно нормализовать изменение цены (рассчитать относительное изменение) это значимо улучшит распределение целевой метрики, приблизив его к нормальному, отмасштабирует все значения целевой переменной, а также позволит использовать в том числе линейные модели в ансамбле. Таким образом целевая переменная будет рассчитываться по формуле (6)

|  |  |  |
| --- | --- | --- |
|  |  | (6) |
|  |  | (7) |

где - усредненая стоимость акции в текущий отчетный период;

- стоимость акции в конкретный день.

### Обучение и анализ результатов различных моделей машинного обучения

Процесс обучения моделей необходимо начинать с анализа распределения целевой переменой. В данном случае целевой переменной является относительное изменение капитализации крупной компании в связи с выходом новых отчетов, поэтому уместно предположить, что это изменение не может быть очень большим, а если компания существенно выросла или упала в цене, то это скорее обусловлено сторонними факторами, а не вышедшими отчетами, поэтому такие измерения у которых значение относительного изменения превысило 0.5 будут считаться выбросами и должны быть удалены из выборки. Исходное распределение и распределение после удаления выбросов из данных будут представлены на рисунке 9.

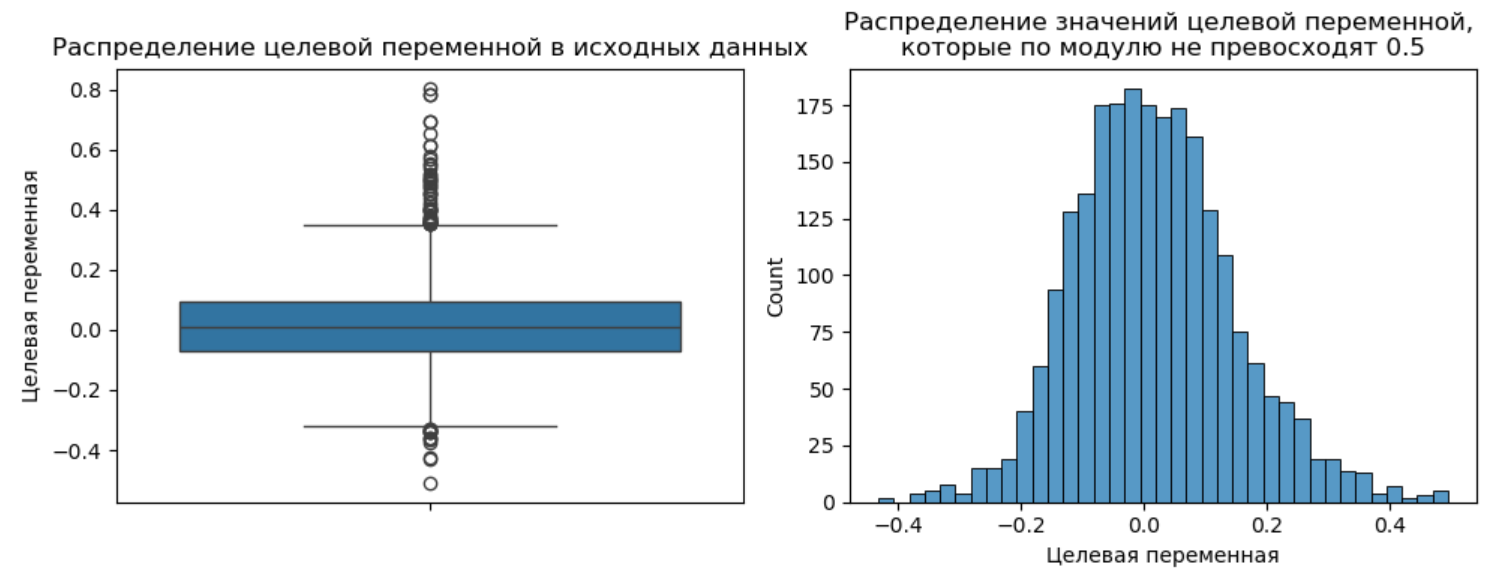


Рисунок 9 – Распределение целевой переменной

Далее будут применены несколько различных моделей машинного обучения, для каждой из которых будет осуществлен подбор параметров и выявление лучшего результата. Учитывая специфику задачи крайне важно не только достичь хороших показателей точности, но еще и избежать переобучения, потому что переобучение будет значить что алгоритм будет пригоден для работы только над теми данными, на которых он обучался, т.е. в будущем его предсказания почти гарантированно окажутся некорректными. Для контроля переобучения для каждого алгоритма будут строиться вариационные кривые. После выявления оптимальных параметров каждой модели будут посчитаны значения различных метрик точности для дальнейшего сравнения моделей между собой. В качестве основной метрики для построения вариационных кривых и подбора параметров будет использована среднеквадратичная ошибка, которая рассчитывается по формуле (8)

|  |  |  |
| --- | --- | --- |
|  |  | (8) |

Где – истинное значение предсказываемой величины;

– предсказанное значение;

– количество измерений в выборке.

Помимо этого, для более детального сравнения результатов работы моделей будет использован коэффициент детерминации, значение которого рассчитывается по формуле (9):

|  |  |  |
| --- | --- | --- |
|  |  | (9) |

где – истинное значение предсказываемой величины;

– предсказанное значение;

– количество измерений в выборке;

– среднее значение предсказываемой величины.

И абсолютная средняя ошибка, значение которой рассчитывается по формуле (10)

|  |  |  |
| --- | --- | --- |
|  |  | (10) |

где – истинное значение предсказываемой величины;

– предсказанное значение;

– количество измерений в выборке.

Данные доступны лишь за ограниченных период времени, в связи с чем их не очень много и в данном случае будет не уместно разбивать набор данных на тренировочный, валидационный и тестовый, поэтому данные будут разбиты только на тренировочный и тестовый. Следующий нюанс обучения моделей заключается в том, что, предсказывая изменение на текущий квартал, модель не должна знать про то, как в аналогичном квартале (и в будущем) изменились капитализации других компаний. Например, если вдруг случился мощный кризис, на фоне всеобщей паники капитализации всех компаний попадали даже при хороших отчетах. И модель по косвенным признакам поняла, что в данный момент кризис и занизила предсказания. Но во время инференса такой информации не будет, поэтому необходимо делать time-series валидацию. Идея которой заключается в том, что модели сначала обучаются на информации из хронологически первых отчетов, тестируются на следующих, потом обучаются на хронологически первых двух партиях отчетов, тестируя на третьей и т.д.

Ниже описаны причины выбора каждого из используемых алгоритмов и сам процесс их обучения.

Первый из используемых алгоритмов – лассо регрессия, которая несмотря на свою простоту имеет свойство выявлять только значимые независимые переменные, обнуляя коэффициенты всех остальных, что может быть полезно в процессе имитации фундаментального анализа ценных бумаг. Для выбора наилучшей модели была поведена оптимизация параметров алгоритма путем их перебора по сетке, а затем была построена валидационная кривая, отражающая зависимость показателей метрики точности от значений параметра , который отвечает за регуляризацию параметров независимых переменных. Валидационная кривая представлен на рисунке 10.

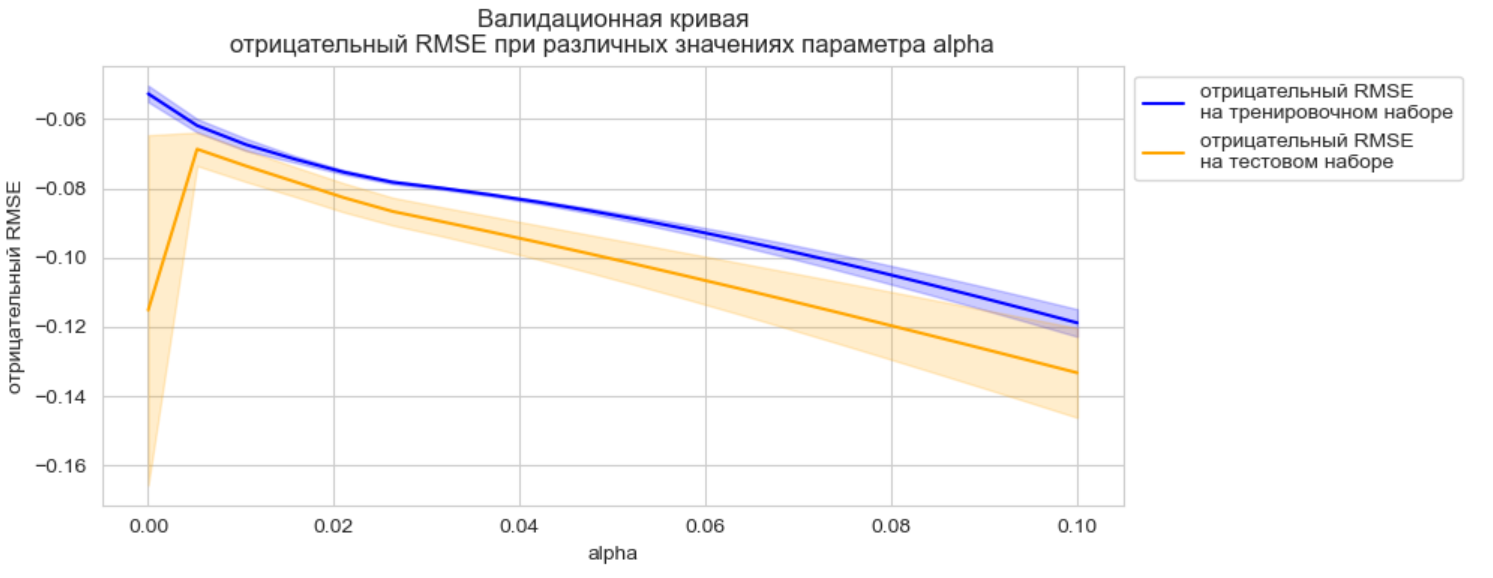


Рисунок 10 – Валидационная кривая линейной регресии

Следующий алгоритм – случайный лес, т.к. долгое время считалось что этот алгоритм почти не способен к переобучению. Это свойство очень уместно в контексте решения данной задачи. Для выбора наилучшей модели был поведен подбор оптимального значения параметра , который отвечает за максимальную глубину каждого из обучаемых решающих доревев, затем построена валидационная кривая, отражающая зависимость показателей метрики точности от значений параметра , который отвечает за количество обучаемых решающих деревьев. Валидационная кривая представлен на рисунке 11.

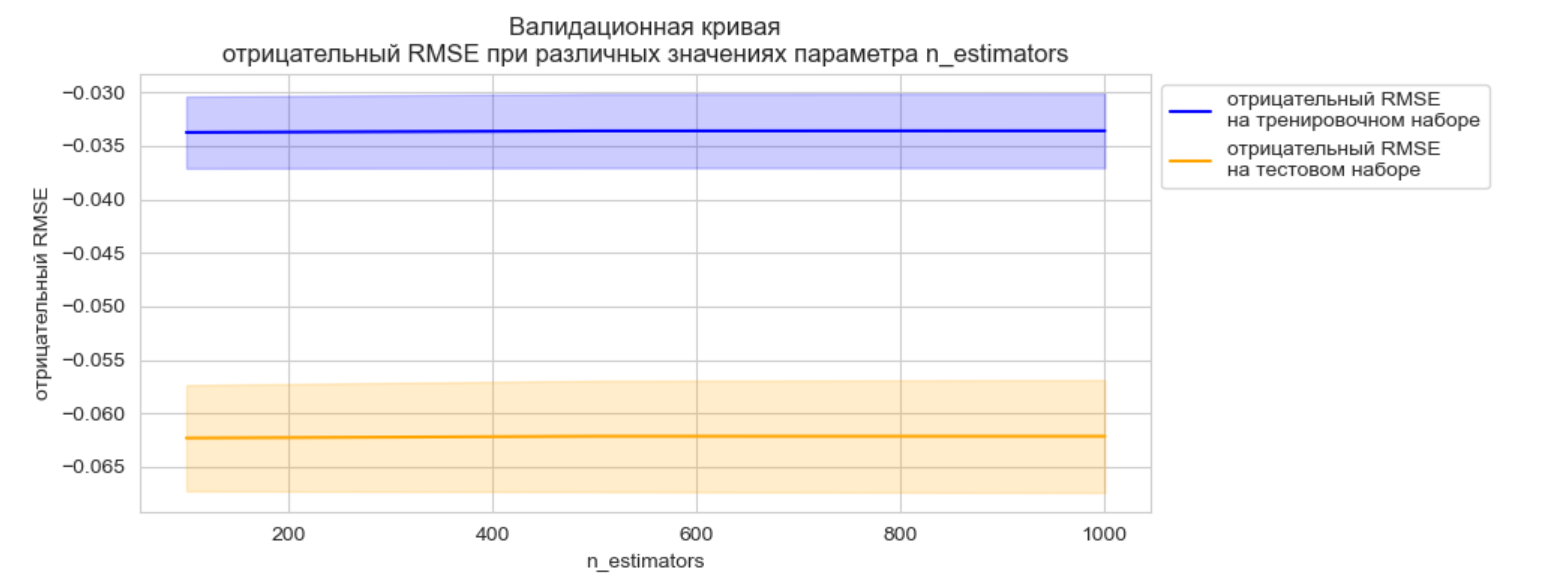


Рисунок 11 – Валидационная кривая алгоритма случайного леса

Третий алгоритм – LightGBM, который является модификацией градиентного бустинга и на данный момент считается одним из лучших алгоритмов машинного обучения по точности и скорости обучения. Для оптимизации параметров этого алгоритма будет применен поэтапный перебор по сетке различных комбинаций параметров, т.е. на первом этапе будет подобран оптимальный , определяющий чувствительность весов независимых переменных к ошибкам предсказаний модели, на следующем этапе будет подобрана оптимальная комбинация параметров , которые отвечают за регуляризацию. На третьем этапе будет подобрана оптимальная комбинация параметров , которые регулируют сложность каждого отдельного решающего дерева, на четвертом этапе будет подобрана оптимальная комбинация параметров , которые регулируют количество данных, которое получает каждое отдельное дерево решений для обучения. Последним шагом стало построение валидационной кривой, отражающей зависимость показателей метрики точности от значений параметра , который отвечает за количество обучаемых решающих деревьев. Валидационная кривая представлен на рисунке 12.

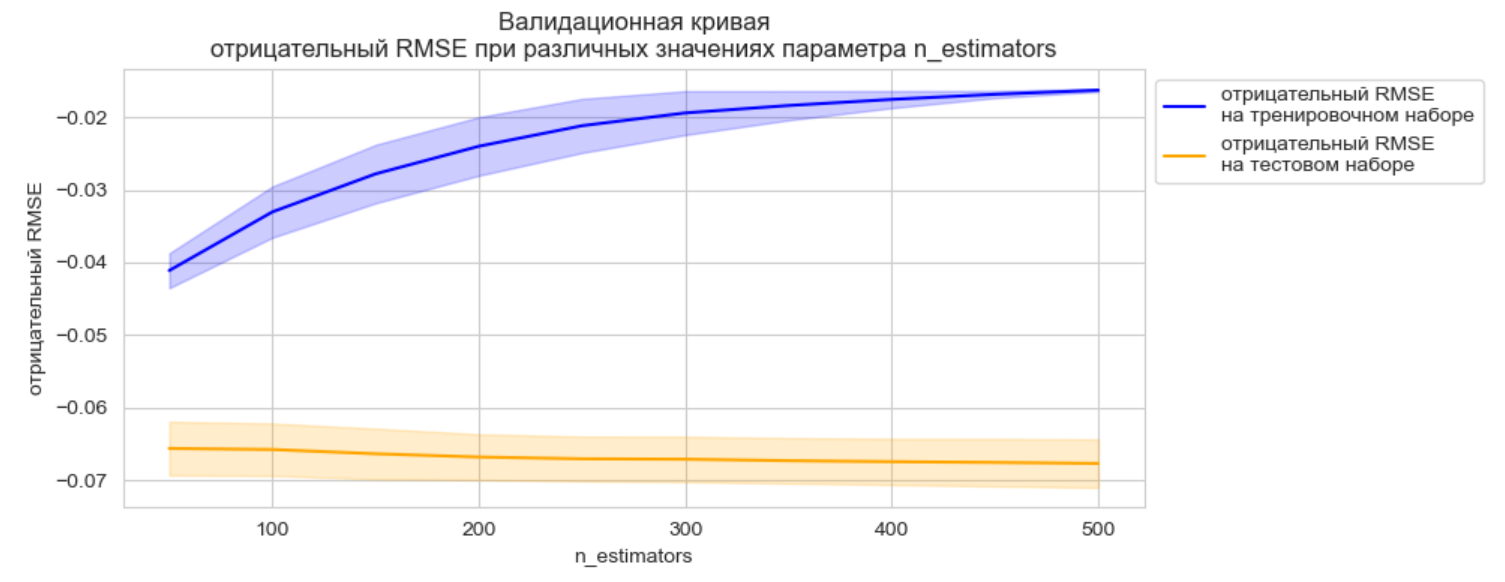


Рисунок 12 – Валидационная кривая алгоритма LightGBM

Последний используемый алгоритм – CatBoost, который также является модификацией градиентного бустинга, но еще и умеет работать с категориальными данными, применяя кодирование переменных средним значением по каждой категории, что может оказаться весьма полезным учитывая, что важной категориальной переменной является сектор экономики. Методика оптимизации параметров аналогична той, которая была применена для LightGBM. Валидационная кривая представлен на рисунке 13.

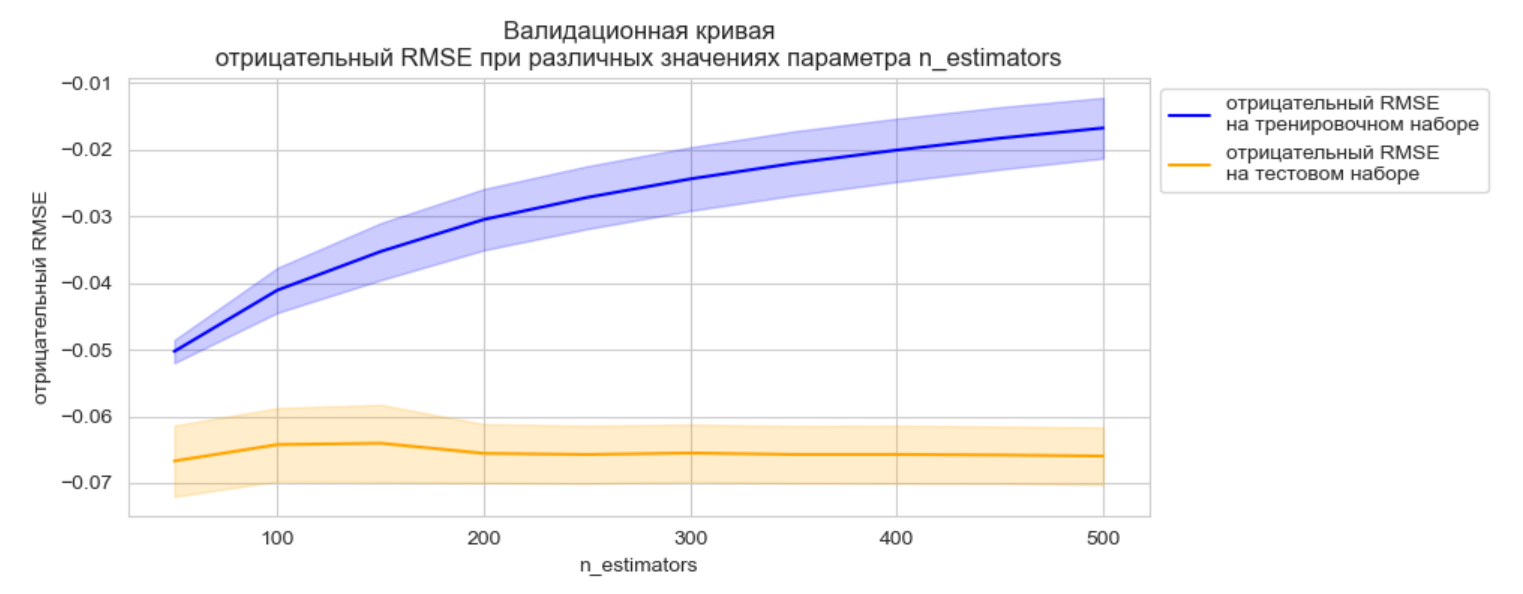


Рисунок 13 – Валидационная кривая алгоритма CatBoost

Все показатели качества моделей представлены в таблице 4.

Таблица 4 – Показатели качества моделей

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | Показатели на обучении | | | Показатели на тесте | | |
| Метрика  Модель | RMSE | R2 | MAE | RMSE | R2 | MAE |
| Lasso | 0.058499 | 0.793204 | 0.045396 | 0.063886 | 0.781666 | 0.045472 |
| Случайный лес | 0.029222 | 0.948400 | 0.023216 | 0.065279 | 0.772043 | 0.046085 |
| LightGBM | 0.018440 | 0.979453 | 0.013704 | 0.076637 | 0.685813 | 0.053431 |
| CatBoost | 0.015416 | 0.986143 | 0.012335 | 0.055922 | 0.777302 | 0.043145 |

В данной задаче почти по всем показателям лучше всех себя показывает CatBoost, однако показатели всех моделей приемлемые и все модели будут включены в итоговый ансамбль.

### Формирования ансамбля моделей

Одна модель может вести быть нестабильной при предсказаниях или дообучении на новых данных. В связи с этим и с целью улучшения итогового качества предсказаний было принято решение использовать ансамбль моделей. Для составления ансамбля существует несколько техник, однако в данном случае в связи с малым количеством данных использовать беггинг (обучение разных моделей на разных поднаборах данных) было бы некорректно. Поэтому было использовано смешивание - это метод ансамблевого машинного обучения, который использует модель машинного обучения, чтобы узнать, как наилучшим образом объединить прогнозы из нескольких моделей - участников ансамбля. Архитектура пакетированной модели содержит две или более базовых моделей, часто называемых моделями нулевого уровня, а метамодель, которая объединяет прогнозы базовых моделей, – моделью первого уровня. Метамодель обучается на основе прогнозов, сделанных базовыми моделями.

В качестве итогового алгоритма будет разработан стек-ансамбль, где базовые модели - это модели машинного обучения, обучение которых было рассмотрено в пункте 3.1.3, а метамодель - это линейная модель, которая «смешивает» прогнозы базовых моделей. В данном случае в качестве метамодели будет использована линейная регрессия.

Ансамбль реализован в соответствии со стандартным интерфейсом всех алгоритмов машинного обучения т.е. должен реализовывать методы fit и predict, и должен предоставлять возможность подбора оптимальных параметров для метамодели. Помимо этого, важно учесть, что некоторые базовые алгоритмы могут работать с разными наборами данных и ансамбль должен это учитывать. Код разработанного ансамбля представлен в листинге 1, в приложении А.

Ансамбль призван улучшать качество итоговых предсказаний, а значит его показатели целевых метрик качества должны быть лучше, чем у любой отдельно взятой модели. Показатели качества всех моделей на тестовом наборе данных в сравнении с ансамблем приведены в таблице 5.

Таблица 5 – Показатели качества всех моделей на тестовом наборе данных

|  |  |  |  |
| --- | --- | --- | --- |
| Модель | R2 | RMSE | MAE |
| Lasso | 0.554967 | 0.065454 | 0.049650 |
| Случайный лес | 0.788325 | 0.053751 | 0.039708 |
| LightGBM | 0.785850 | 0.055925 | 0.041067 |
| CatBoost | 0.795142 | 0.054324 | 0.040078 |
| Ансамбль | 0.811879 | 0.053625 | 0.039574 |

Ансамбль действительно показал себя лучше всех моделей по всем показателям, однако случайный лес очень близок к нему по точности, что еще раз подтверждает тот факт, что случайный лес почти не переобучается и его показатели на незнакомых наборах данных будут почти идентичны тем, которые он демонстрировал на тренировочных наборах. Однако ансамбль стабильнее, поэтому несмотря на очень схожие показатели точности в качестве итогового алгоритма будет использован ансамбль.

### Тестирование разработанного алгоритма

Для того чтобы узнать будет ли на практике подобная стратегия приносить прибыль, необходимо провести тестирование на исторических данных. Его суть заключается в том, что мы имитируем реальную работу алгоритма, предоставляя ему только данные за определенный период, не давая ему заглянуть в будущее. Алгоритм на основе этих данных составляет портфолио, а потом мы по известной истории цен акций узнаем принесло ли собранное портфолио прибыль. Помимо размера прибыли на конец рассматриваемого периода необходимо учитывать еще ряд эконометрических показателей, основные из которых максимальная просадка цен и волатильность стоимости.

Такому алгоритму мало быть прибыльным, его использование несет в себя ряд рисков, учитывая что для потенциальных клиентов приложения он будет черным ящиком, поэтому чтобы создать у пользователей мотивацию следовать рекомендациям необходимо продемонстрировать что риск оправдан и вознаграждение за риск больше чем у более безопасных вариантов, поэтому динамика стоимости портфолио будет сравниваться с динамикой индекса S&P 500, включающего в себя 500 крупнейших американских компаний. Вложения в данный индекс многие инвестры считают безопасным способом преумножить свой капитал, соответственно у большинства из них будет возникать вопрос о целесообразности вложений в более рисковый актив, чтобы на этот вопрос ответить необходимо продемонстрировать что при сопоставимом уровне рисков (волатильности цен и их просадок) выигрыш инвестора значительно выше при ведении рекомендованного портфолио. На рисунке 14 представлен график динамики стоимости портфолио.

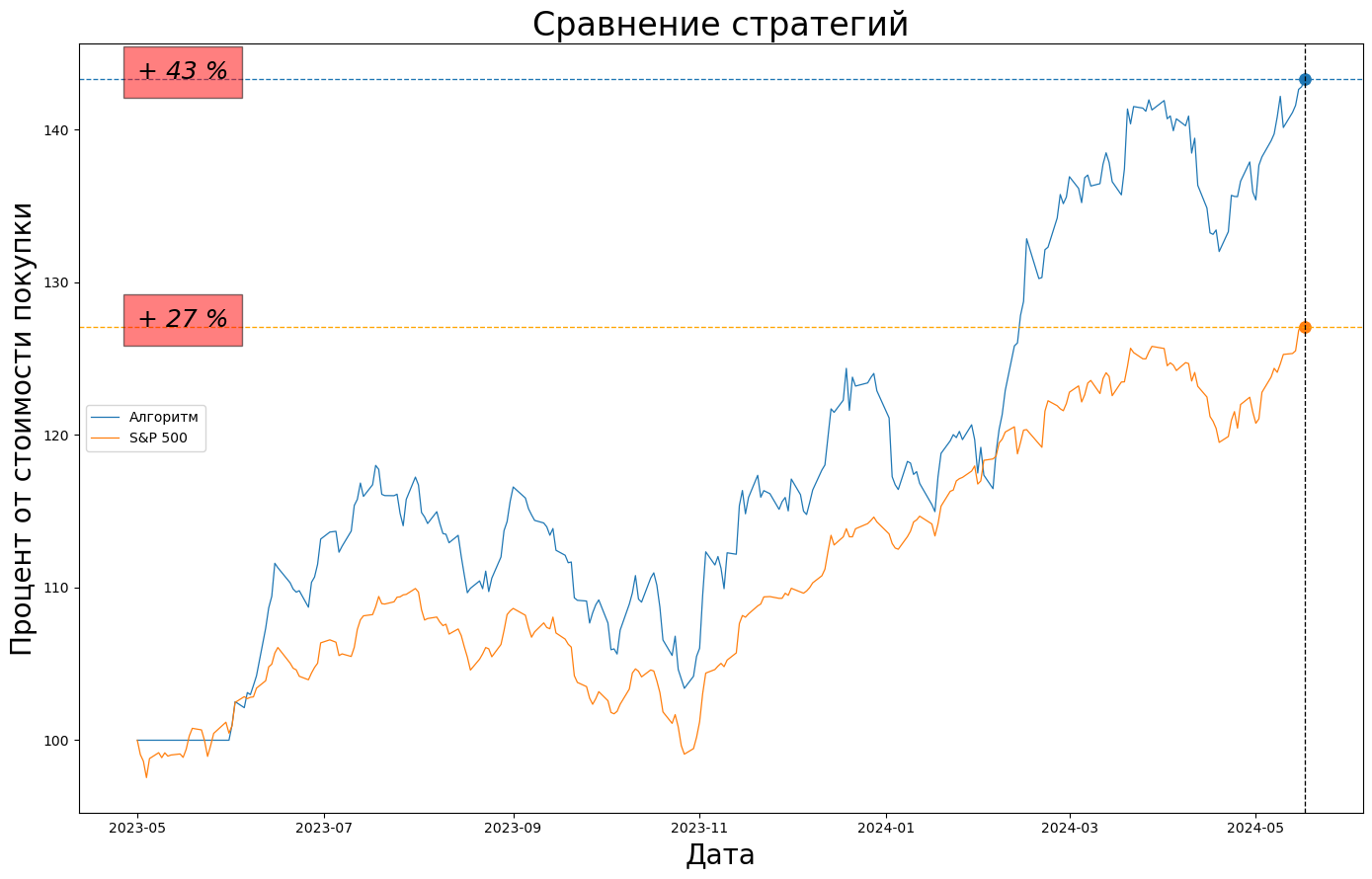


Рисунок 14 – Динамика стоимости портфолио

На протяжении большей части расматриваемого периода (вплоть до февраля 2024) динамика стоимости индекса и рекомендованного портфолио имеет очень высокую корреляцию, при этом стоимость портфолио в каждый отдельный момент выше в среднем на 8%. В феврале 2024 индекс догоняет алгоритм по стоимости, однако это обусловлено лишь тем, что алгоритм на тот момент только что осуществил перебалансировку. Спустя очень небольшое время после этого обновления состава стоимость портфеля за несколько дней возрастает почти на 20%. Начиная с этого момента динамики цен вновь приобретают очевидную корреляцию, однако стоимость портфолио, составленного алгоритмом теперь выше в среднем на 16%. В итоге к концу рассматриваемого периода алгоритм оказывается выгоднее на 16%.

## Реализация серверной части приложения

### Создание проекта

Помимо исходного кода проект должен содержать множество побочных файлов, таких как файлы для системы контроля версий git, файлы для системы автоматизации развёртывания и управления приложением docker и так далее. Поэтому сам исходный код будет помещен в директорию «src» основного проекта. Более того клиентская и серверная части приложения будут реализованы отдельно поэтому внутри директории «src» были созданы еще две директории «backend» и «frontend». В данном разделе речь пойдет о создании серверной части приложения с использованием веб-фреймворка Django внутри директории «backend».

После установки, веб-фреймворк Django предоставляет возможность создания проекта. Для этого необходимо выполнить команду «django-admin startproject smart\_invest», где smart\_invest это название нового проекта. Проект создается в директории, из которой была вызвана данная команда.

Далее, внутри проекта были созданы приложения. Приложение – это полностью или частично независимая единица функциональности проекта. Весь проект может быть реализован внутри одного приложения или же разбит на множество функциональных частей выполняющие разные задачи. В данном случае было создано два приложения: «main\_app» и «smart\_invest». Первое отвечает за функциональную составляющую всех страниц веб приложения, а второе - за глобальные настройки сервиса и первичную маршрутизацию.

Помимо обработки пользовательских запросов перед веб приложением стоит задача поддержания информации в актуальном состоянии, для выполнения этой задачи понадобится ряд инструментов, среди которых компоненты, которые отвечают за загрузку новостей, загрузку финансовых данных, обновление рекомендованных портфолио и компоненты, управляющие всеми перечисленными инструментами по расписанию. Для реализации всех перечисленных программных компонентов была создана отдельная директория с названием «utils», внутри которой были созданы несколько пакетов: «updating», в котором будут располагаться все компоненты для обновления различной информации; «scheduling», в котором будут располагаться компоненты по управлению обновлениями в соответствии с расписанием. А также папка «data», в которой будут храниться некоторые финансовые данные компаний.

Для подготовки данных для отправки клиенту будут реализованы отдельные компоненты, которые будут размещены в директории «utils/api\_loading».

Для установки сторонних пакетов и дальнейшей корректной работы приложения на других машинах было создано виртуальное окружение «smart\_invest\_env».

Итоговая файловая структура проекта представлена на рисунке 15.



Рисунок 15 – Файловая структура проекта

### База данных

По умолчанию в Django используется база данных SQLite3 [11]. Она создается при первом обращении к ней. SQLite3 – это автономный, работающий без сервера, транзакционный механизм базы данных SQL.

Согласно описанным выше функциональным требованиями, можно прийти к выводу, что использование SQLite вполне достаточно для решения поставленных задач.

Структура базы данных в Django создается с помощью моделей. Модель – это класс, который описывает одну таблицу в базе данных, в том числе и ее поля. Каждый экземпляр класса является отдельной записью в таблице. С его помощью можно получать, изменять и определять новые значения полей таблицы.

Модели описываются внутри файла «models.py». Для вебсайта, согласно спроектированной структуры базы данных, изображенной на рисунке 6, были созданы модели для таблиц «Strategy», «PortfolioComposition» и «News», которые содержат информацию о параметрах стратегий, составе портфолио новостях соответственно. Определение всех моделей представлено в листинге 1.

Листинг 1 – Определение моделей базы данных

class News(models.Model):  
 original\_id = models.CharField(max\_length=100)  
 title = models.CharField(max\_length=200)  
 author = models.CharField(max\_length=200)  
 article\_url = models.URLField()  
 image\_url = models.URLField()  
 description = models.CharField(max\_length=500)  
 published\_utc = models.DateTimeField()  
 tickers = models.TextField()  
 publisher\_name = models.CharField(max\_length=100)  
  
class Strategy(models.Model):  
 name = models.CharField(max\_length=50)  
 features\_file = models.FilePathField()  
 target\_smoother\_file = models.FilePathField()  
 to\_num\_prep\_file = models.FilePathField()  
 to\_num\_and\_cat\_prep\_file = models.FilePathField()  
 predictor\_file = models.FilePathField()  
 criterion = models.CharField(max\_length=30)  
 target\_n\_days = models.IntegerField()  
 tickers\_from\_sector = models.IntegerField()  
  
class PortfolioComposition(models.Model):  
 strategy\_id = models.ForeignKey(Strategy, on\_delete=models.CASCADE)  
 stock\_name = models.CharField(max\_length=50)  
 part = models.FloatField()creation\_date = models.DateTimeField()  
 sector = models.CharField(max\_length=50)

После определения моделей необходимо создать миграции [12]. Миграции – это сгенерированный код на основе моделей, который определяет на языке Python структуру базы данных, ее таблицы, поля, связи и т.д. Для создания миграций в командной строке была использована команда «manage.py makemigrations». При последующем применении миграций командой «manage.py migrate» уже выполняются непосредственно команды в виде SQL кода для конкретной базы данных.

### Маршрутизатор

Маршрутизатор – набор шаблонных правил связывающие URL адреса с определенными контроллерами.

Контроллер – программный код, который запускается при обращении по конкретному URL адресу.

Процесс получения пользователем содержимого веб-страницы заключается в следующем. При обращении на определенный URL адрес веб-сайта маршрутизатор следует определённым в нем шаблонным путям и передает управление соответствующему контроллеру, который в свою очередь отдает в качестве ответа данные, с использованием которых будет сформирован итоговый вид страницы на клиентской стороне веб приложения.

Шаблоны первичной маршрутизации находятся в файле «urls.py» в конфигурационной директории проекта «src/backend/smart\_invest». Его содержимое представлено в листинге 2.

Листинг 2 – Шаблоны первичной маршрутизации

urlpatterns = [  
 path('', include('main\_app.urls'))  
]

В списке «urlpatterns» хранятся все маршруты проекта. Каждый маршрут записывается в виде результата выполнения функции «path()», где в качестве первого параметра указывается шаблон пути, а в качестве второго может указываться список маршрутов уровня приложения.

Этот файл будет использован для будущего масштабирования системы и ее расширения, а пока все маршруты обрабатываются файлом «main\_app.urls». Его содержимое представлено в листинге 3.

Листинг 3 – Шаблоны маршрутизации приложения «main\_app»

urlpatterns = [  
 path('', views.index),  
 path("news/<int:limit>/", views.get\_news),  
 path("main\_tickers\_data/", views.get\_main\_tickers\_data),  
 path("incorrect\_rated\_shares", views.get\_incorrect\_rated\_shares),  
 path("ticker\_history/<slug:ticker>/", views.get\_ticker\_history),  
 path("strategies\_info/", views.get\_strategies\_info),  
 path("actual\_portfolio/", views.get\_actual\_portfolio),  
]

В файле маршрутизации приложения «main» определены маршруты для всех основных URL адресов веб-сайта, а также указаны соответствующие контроллеры, которые обрабатывают их. Контроллеры хранятся в файле «views.py».

Каждый метод, реализованный в этом файле по сути лишь оборачивает в нужный для отправки формат данные, которые были получены при помощи класса «ApiLoader», который реализует методы загрузки и предобработки данных для дальнейшей отправки их клиенту. Код класса «ApiLoader» представлен в листинге 2 приложения А.

### Реализация компонентов, отвечающих за обновление данных

Компоненты, отвечающие за обновление данных расположены в папке «utils/updating», существует три категории данных, которые необходимо поддерживать в актуальном состоянии: новости, рекомендованное портфолио и различные статистики, такие как актуальные котировки, истории цен акций на рынке и т.д. В связи с этим внутри данной директории было создано еще три пакета «news\_updating», «portfolio\_updating» и «stat\_updating», каждый для обновления своей категории данных.

Внутри пакетов помимо побочных компонентов реализованы соответственно классы «NewsUpdater», «PortfolioUpdater» и «Statupdater», экземпляры которых будут использованы компонентом, реализующим обновление по расписанию. Ниже будет описан функционал каждого из классов.

Класс «NewsUpdater» осуществляет обновление новостей из мира трейдинга. Он в свою очередь использует наследников интерфейса «NewsLoader» для загрузки новостей и наследников интерфейса «NewsPreprocessor» для обработки новостей и приведение их к необходимому формату (извлечение нужных полей, частичный перевод содержания на русский язык).

Класс «StatUpdater» осуществляет обновление статистики инвестиционных стратегий. Он загружает актуальные данные котировок акций при помощи наследников интерфейса «FinanceDataLoader», и потом, на основании этих данных проводит для каждой инвестиционной стратегии, параметры которой находятся в базе данных бэк-тест на исторических данных. Функционал проведения данных тестов реализован в классе «BackTest».

Класс «PortfolioUpdater» обновляет портфолио для каждой инвестиционной стратегии, параметры которой содержатся в базе данных, для обновления ему необходимо выполнить несколько шагов:

1. Загрузить фундаментальные отчеты компаний при помощи наследников интерфейса «DataLoader»;
2. Сформировать набор данных на основе отчетов и временных рядов стоимостей ценных бумаг при помощи класса «DataFrameBuilder»;
3. Обработать сформированный набор данных (масштабировать числовые данные и закодировать категориальные) при помощи наследников интерфейса «DataPreprocessor»;
4. Составить новое портфолио при помощи класса «PortfolioRebalancer».

На рисунке 16 представлена диаграмма последовательности, демонстрирующая работу класса «PortfolioUpdater» по обновлению актуальных портфолио.

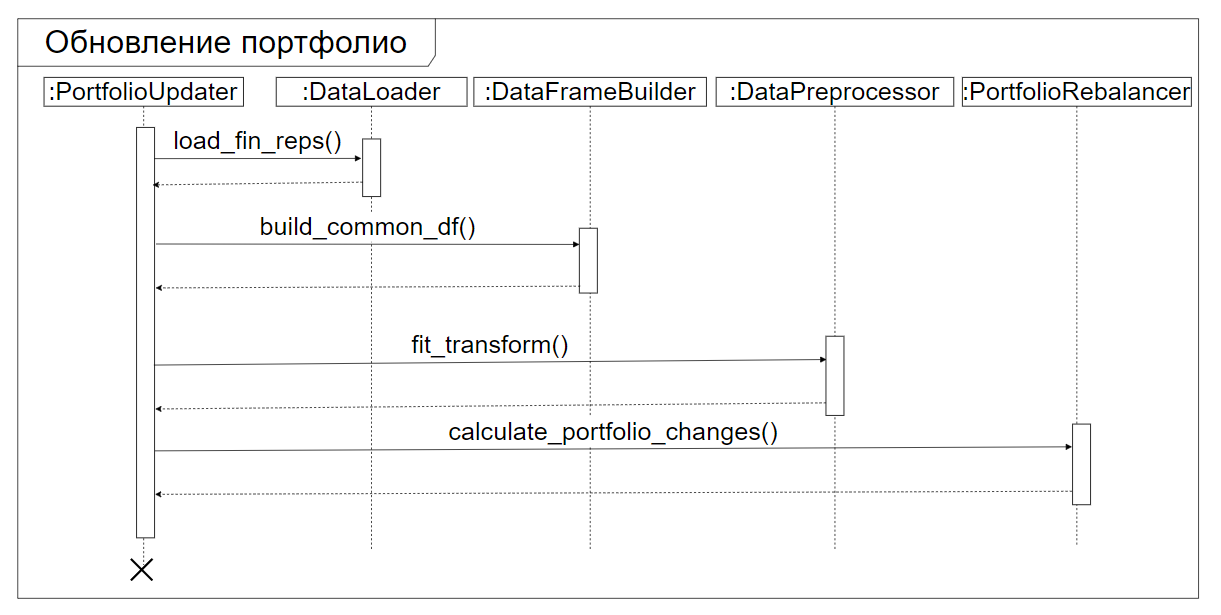


Рисунок 16 – Диаграмма последовательности обновления портфолио

### Реализация компонентов, отвечающих за расписание обновлений

За расписание обновлений отвечает модуль «scheduler» в директории «utils/scheduling». Для составления расписания используется класс «BackgroundScheduler» модуля «apscheduler.schedulers.background», встроенного в фреймворк Django.

Для запуска обновлений по расписанию был реализован метод «start», который представлен в листинге 4.

Листинг 4 – Функция запуска обновлений

def start():

news\_updater, stat\_updater, portfolio\_updater = init\_updaters()

scheduler = BackgroundScheduler()

scheduler.add\_job(news\_updater.update, 'cron', day='\*', hour=20)

scheduler.add\_job(stat\_updater.update, 'cron', day='\*', hour=20)

scheduler.add\_job(portfolio\_updater.update, 'cron', month='\*', day='1st mon', hour=20)

scheduler.start()

Новости из мира трейдинга и статистические данные будут обновляться ежедневно в 20 часов. А Вот портфолио будет обновляться в первый понедельник каждого месяца.

## Реализация клиентской части приложения

Современное веб-приложение требует эффективной и удобной клиентской части для обеспечения удовлетворения потребностей пользователей. В данном разделе рассматривается процесс разработки клиентской стороны веб-приложения с использованием веб фреймворка React. Скриншоты реализованного приложения представлены в приложении Б.

### Организация исходного кода

Исходный код проекта организован в два основных раздела – страницы и компоненты, реализация которых находится соответственно в директориях «frontend/src/components» и «frontend/src/views».

Компоненты позволяют разделить пользовательский интерфейс на независимые, повторно используемые части и работать с каждой из частей отдельно. Они принимают произвольные входные данные (называемые «props» или свойствами) и возвращают React-элементы, описывающие, что должно появиться на экране. При анализе макета и проектировании клиентской части приложения были выявлены компоненты, необходимые для реализации. Среди основных компонентов, которые были разработаны в рамках данного проекта, можно выделить:

1. Кнопка, компонент, обеспечивающий пользователю возможность выполнения определенного действия при нажатии;
2. Таблица, компонент для отображения табличных данных в удобном для пользователя виде;
3. Новостной блок, компонент, позволяющий отображать информацию о посте или статье с возможностью перехода к оригинальному посту;
4. Карусель, компонент, представляющий собой интерактивный элемент пользовательского интерфейса, который позволяет отображать серию изображений или контента в виде слайдов, которые автоматически или вручную переключаются.

Страницы приложения состоят из компонентов, которые являются основными строительными блоками интерфейса. Исходный код страниц формируется на основе переиспользуемых компонентов, которые могут быть использованы с различными параметрами. Это позволяет создавать гибкие и динамичные интерфейсы для пользователей, а также сокращает объем кода и упрощает его поддержку и масштабирование.

Для повышения надежности и читаемости кода в процессе разработки был использован TypeScript [13]. TypeScript предоставляет статическую типизацию, что позволяет выявлять ошибки на ранних этапах разработки и облегчает понимание структуры приложения.

Для сборки исходного кода в исполняемый браузерный файл был выбран сборщик Vite [14]. Vite был выбран из-за своей высокой производительности, компактного размера собранного файла и повышенной безопасности. Этот инструмент обеспечивает быструю сборку проекта и оптимизирует загрузку веб-приложения для конечного пользователя.

### Взаимодействие с сервером

Для обмена данными с сервером и отправки HTTP-запросов была использована библиотека axios [15]. Axios – это JavaScript-библиотека для выполнения HTTP-запросов из браузера или среды выполнения Node.js. Axios предоставляет удобные методы для работы с AJAX-запросами, обеспечивая эффективное взаимодействие клиентской части приложения с сервером.

Основными преимуществами данного инструмента, благодаря которым выбор был сделан в пользу него послужили:

1. Простота использования. Axios предоставляет интуитивно понятный API, что делает его отличным выбором для разработчиков, даже если они только начинают работать с HTTP-запросами. Он предлагает методы для отправки различных типов запросов, а также обработки ответов и ошибок;
2. Поддержка запросов в браузере. Axios может использоваться как в браузере, так и на сервере. Таким образом, есть возможность использовать один и тот же код для выполнения HTTP-запросов как на стороне клиента, так и на стороне сервера приложения;
3. Поддержка Promise (Промис). Axios использует Promise для обработки асинхронных операций, которые связаны с отправкой и получением HTTP-запросов. Благодаря этому есть возможность писать чистый и более структурированный код без колбэк-функций.

# ТЕСТИРОВАНИЕ

Для проверки работоспособности веб-приложения, необходимо провести тестирование. Чтобы проверить корректную работу веб-приложения в разных браузерах было проведено кроссбраузерное тестирование. Чтобы проверить корректную работу веб-приложения и обеспечения соответствия заявленному функционалу было выполнено функциональное тестирование, направленное на проверку соответствия работы приложения всем указанным требованиям.

### Кроссбраузерное тестирование

Тестирование проводилось путем посещения разных наиболее популярных браузеров и выполнением нескольких сценариев вручную. Результаты тестирования представлены в таблице 6.

Таблица 6 – Результаты кроссбраузерного тестирования

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| № теста | Название теста | Шаги | Ожидаемый результат | Тест пройден |
| 1 | Проверка корректности отображения веб сайта в веб-браузере Edge | Заход на сайт при помощи браузера Edge и выполнение нескольких сценариев | Страницы веб-сайта соответствуют прототипу при отображении в веб-браузере Edge | Да |
| 2 | Проверка корректности отображения веб сайта в веб-браузере Chrome | Заход на сайт при помощи браузера Chrome и выполнение нескольких сценариев | Страницы веб-сайта соответствуют прототипу при отображении в веб-браузере Chrome | Да |
| 3 | Проверка корректности отображения веб сайта в веб-браузере Firefox | Заход на сайт при помощи браузера Edge и выполнение нескольких сценариев | Страницы веб-сайта соответствуют прототипу при отображении в веб-браузере Firefox | Да |
| 4 | Проверка корректности отображения веб сайта в веб-браузере Yandex | Заход на сайт при помощи браузера Yandex и выполнение нескольких сценариев | Страницы веб-сайта соответствуют прототипу при отображении в веб-браузере Yandex | Да |

### Функциональное тестирование

Результаты тестирования представлены в таблице 7.

Таблица 7 – Результаты функционального тестирования

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| № теста | Название теста | Шаги | Ожидаемый результат | Тест пройден |
| 1 | Проверка корректности навигации на главной странице | Заход на главную страницу сайта и последовательное нажатие на кнопки «Сравнить с честной стоимостью» и «Стратегия» | После нажатия на кнопку «Сравнить с честной стоимость» должен осуществляться переход на страницу скринера акций. После нажатия на кнопку «Стратегия» должен осуществляться переход на страницу готового портфолио. | Да |
| 2 | Проверка корректности отображения карусели новостей на главной странице | Заход на главную страницу сайта и переключение новостей в карусели | Новости переключаются, новость, которая в результате переключения становится главной увеличивается в масштабе | Да |
| 3 | Проверка интерактивности графика на главной странице | Заход на главную страницу сайта и наведение курсора на разные точки графика | При наведении на конкретную точку графика должна отображаться надпись, содержащая значение выбранной точки | Да |
| 4 | Проверка корректности навигационной панели | Осуществление переходов на другие страницы сайта при помощи навигационной панели | При нажатии на кнопку навигационной панели происходит переход в соответствующий раздел | Да |
| 5 | Проверка поиска существующих тикеров на странице скринера акций | Введение существующий тикеров акций | При нажатии на кнопку «Сравнить» появляются два интерактивных графика, один из которых отражает динамику рыночной стоимости акций, а второй динамику честной стоимости. | Да |

Окончание таблицы 7

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 6 | Проверка поиска не существующих тикеров на странице скринера акций | Введение не существующий тикеров акций сразу после захода на страницу и после введения корректного названия тикера | При нажатии на кнопку «Сравнить» в обоих случаях должна отображаться надпись «Акция с указанным именем не найдена» | Да |
| 7 | Проверка интерактивность графика на странице готового портфолио | Переход на страницу готового портфолио и наведение на круговой график состава портфолио | При наведении на сектор графика должны отображаться название сектора экономики и суммарный вес акций из этого сектора | Да |
| 8 | Проверка отображения состава портфолио для каждого сектора | Переход на страницу готового портфолио и просмотр акций, включенных в портфолио для каждого сектора | При нажатии на состав конкретного сегмента должна появляться таблица, где каждому тиккеру сопоставлен вес в портфолио | Да |
| 9 | Проверка корректности ссылок на странице новостей | Переход на страницу новостей трейдинга и переход на оригинальную страницу какой-нибудь новости | При нажатии на заголовок новости в новой вкладке должна открываться страница оригинальной новости | Да |
| 10 | Проверка динамичности страницы новостей | Переход на страницу новостей и постепенное листание страницы до конца | При достижении определенного количества просмотренных новостей страница должна подгружать следующую партию без обновления страницы | Да |

**Вывод**

В данной главе было проведено функциональное и кроссбраузерное тестирование клиент-серверного приложения. Все тесты показали, что система функционирует согласно требованиям.

ЗАКЛЮЧЕНИЕ

В соответствии с целями данной работы был проведен обзор научной литературы и существующих аналогов, который подтвердил актуальность поставленной задачи. На основе исторических финансовых данных был разработан алгоритм составления оптимального инвестиционного портфолио.

Разработано веб-приложение, которое предоставляет пользователям набор важных для инвестирования инструментов, а также рекомендует стратегию инвестирования. Для достижения поставленной цели решены следующие задачи:

1. проведен анализ литературы;
2. разработан алгоритм, составляющий оптимальный портфель ценных бумаг;
3. спроектировано веб приложение;
4. реализовано веб приложение;
5. протестировано веб приложение.

Основной задачей данной работы являлась разработка алгоритма предсказания честной стоимости бумаг и составления на основе этих предсказаний инвестиционного портфолио. Созданный алгоритм показал высокую эффективность и точность, однако существует ряд направлений для его дальнейшего усовершенствования. В будущем планируется провести дополнительные исследования и тестирования алгоритма с целью улучшения его предсказательной способности и уменьшения возможных рисков.

ЛИТЕРАТУРА

1. «Seeking alpha» [Электронный ресурс] URL: [https://seekingalpha.com/](https://seekingalpha.com/%20) (дата обращения 04.03.2024).
2. «Investing.com» [Электронный ресурс] URL: <https://www.investing.com/> (дата обращения 04.03.2024).
3. «MorningStar» [Электронный ресурс] URL: [https://www.morningstar.com/](https://www.morningstar.com/%20) (дата обращения 04.03.2024).
4. API «YahooFinance» [Электронный ресурс] URL:   
   https://python-yahoofinance.readthedocs.io/ (дата обращения 04.03.2024).
5. API «Polygon.io» [Электронный ресурс] URL:   
   [https://polygon.io/](https://polygon.io/%20%20)  (дата обращения 04.03.2024).
6. «Scikit-Learn» [Электронный ресурс] URL:   
   [https://scikit-learn.org/](https://scikit-learn.org/%20) (дата обращения 04.03.2024).
7. Django Documentation [Электронный ресурс] URL: [https://docs.djangoproject.com/](https://docs.djangoproject.com/%20) (дата обращения: 14.02.2024 г.).
8. Django Documentation [Электронный ресурс] URL:  [https://react.dev/](%20https://react.dev/) (дата обращения: 14.02.2024 г.).
9. Figma [Электронный ресурс] URL: https://www.figma.com/ (дата обращения: 14.02.2024 г.).
10. John Lintner. “The valuation of risk assets and the selection of risky investments in stock portfolios and capital budgets”. In: The Review of Economics and Statistics 47.1 (1965).
11. Django Documentation [Электронный ресурс] URL: https://www.sqlite.org/docs (дата обращения: 14.02.2024 г.).
12. Дронов В.А. Django 3.0. Практика создания веб-сайтов на Python. – СПб.: БХВ-Петербург, 2021. – 704 с.: ил. – (Профессиональное программирование).
13. TypeScript documentation [Электронный ресурс] URL: [https://www.typescriptlang.org/docs/](https://react.dev/) (дата обращения: 14.02.2024 г.).
14. Vite documentation [Электронный ресурс] URL:   
    [https://vitejs.dev](https://react.dev/) (дата обращения: 14.02.2024 г.).
15. Axios documentation [Электронный ресурс] URL:  
     [https://axios-http.com/docs/intro](https://react.dev/) (дата обращения: 14.02.2024 г.).

ПРИЛОЖЕНИЯ

## Приложение А. Листинги исходного кода

Листинг 1 – Класс «BlendingEnsemble»

class BlendingEnsemble(FairRatesPredictor):

“””

BlendingEnsemble class implements algorithm of blending some base ML models   
 ‘’’

def \_\_init\_\_(self, estimators, meta\_model):

if not isinstance(estimators, list) and all((isinstance(estimator, tuple) for estimator in estimators)):

raise ValueError(

"'estimators' param should be in format: [(estimator, type\_of\_dataset), (estimator, type\_of\_dataset)]\n" +

"Where possible types of datasets are: 'cat' and 'num'.")

self.estimators = estimators

self.meta\_model = meta\_model

def fit(self, num\_dataset, cat\_dataset=None, X=None, y=None):

self.check\_datasets(num\_dataset, cat\_dataset)

meta\_X = []

for est, dataset in self.fit\_generator(num\_dataset, cat\_dataset):

X\_train, y\_train, X\_test, y\_test = dataset

est.fit(X\_train, y\_train)

predictions = est.predict(X\_test)

meta\_X.append(predictions.reshape(-1, 1))

meta\_X = hstack(meta\_X)

self.meta\_model.fit(meta\_X, y\_test)

def predict(self, num\_X, cat\_X=None):

meta\_X = []

for est, X in self.predict\_generator(num\_X, cat\_X):

predictions = est.predict(X)

meta\_X.append(predictions.reshape(-1, 1))

meta\_X = hstack(meta\_X)

return self.meta\_model.predict(meta\_X)

Листинг 2 – Класс «ApiLoader»

class ApiLoader:

def \_\_init\_\_(self):

with open(settings.FINANCE\_DATA\_DIRECTORY + '\\ticker\_to\_sector.json', 'r') as tickers\_file:

self.ticker\_to\_sector = json.loads(tickers\_file.read())

@staticmethod

def get\_prices\_in\_percents(prices: Series):

prices = prices.copy()

first\_price = prices[0]

prices = prices.apply(lambda price: price / first\_price \* 100)

return prices

def load\_news(self, limit=100):

values\_list = ['title', 'author', 'article\_url', 'image\_url', 'description', 'published\_utc', 'publisher\_name']

news = News.objects.order\_by('published\_utc')[:limit].values(\*values\_list)

json\_result = {

'news': list(news)

}

return json\_result

def load\_main\_tickers\_data(self):

main\_tickers\_data = read\_csv(settings.QUOTES\_DIRECTORY + '\\main\_tickers\_quotes.csv')

main\_tickers\_data\_json = main\_tickers\_data.to\_json(orient='index')

json\_result = {

'main\_tickers\_data': main\_tickers\_data\_json

}

return json\_result

def load\_incorrect\_rated\_shares(self):

overrated\_shares = read\_csv(settings.QUOTES\_DIRECTORY + '\\overrated\_shares.csv')

underrated\_shares = read\_csv(settings.QUOTES\_DIRECTORY + '\\underrated\_shares.csv')

overrated\_shares\_json = overrated\_shares.to\_json(orient='index')

underrated\_shares\_json = underrated\_shares.to\_json(orient='index')

json\_result = {'overrated\_shares': overrated\_shares\_json}

return json\_result

## Приложение Б. Скриншоты реализованного веб приложения

На рисунке 1 представлен скриншот главной страницы реализованного приложения.

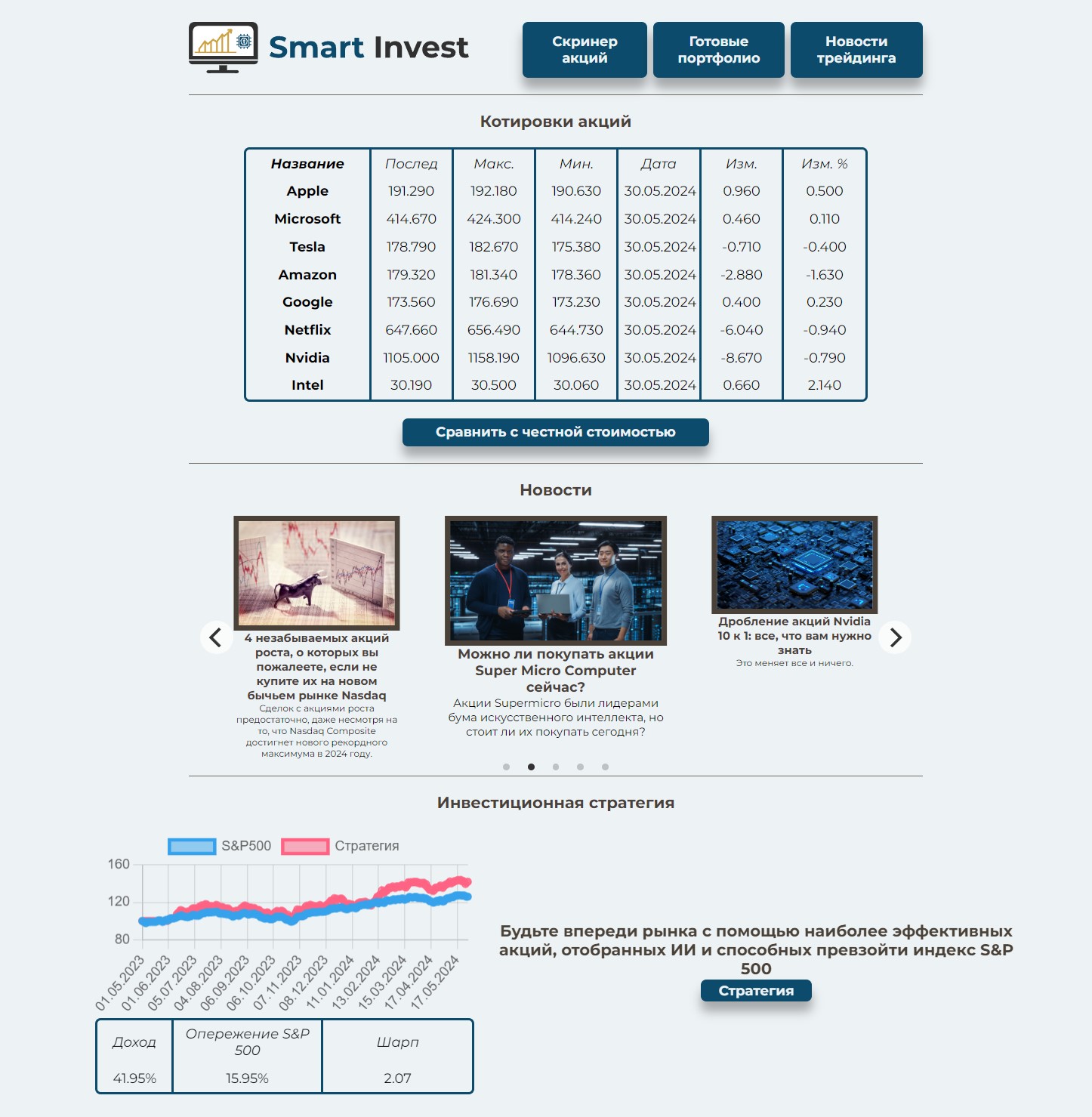


Рисунок 1 – Скриншот главной страницы приложения

На рисунке 2 представлен скриншот страницы рекомендованного портфолио реализованного приложения.

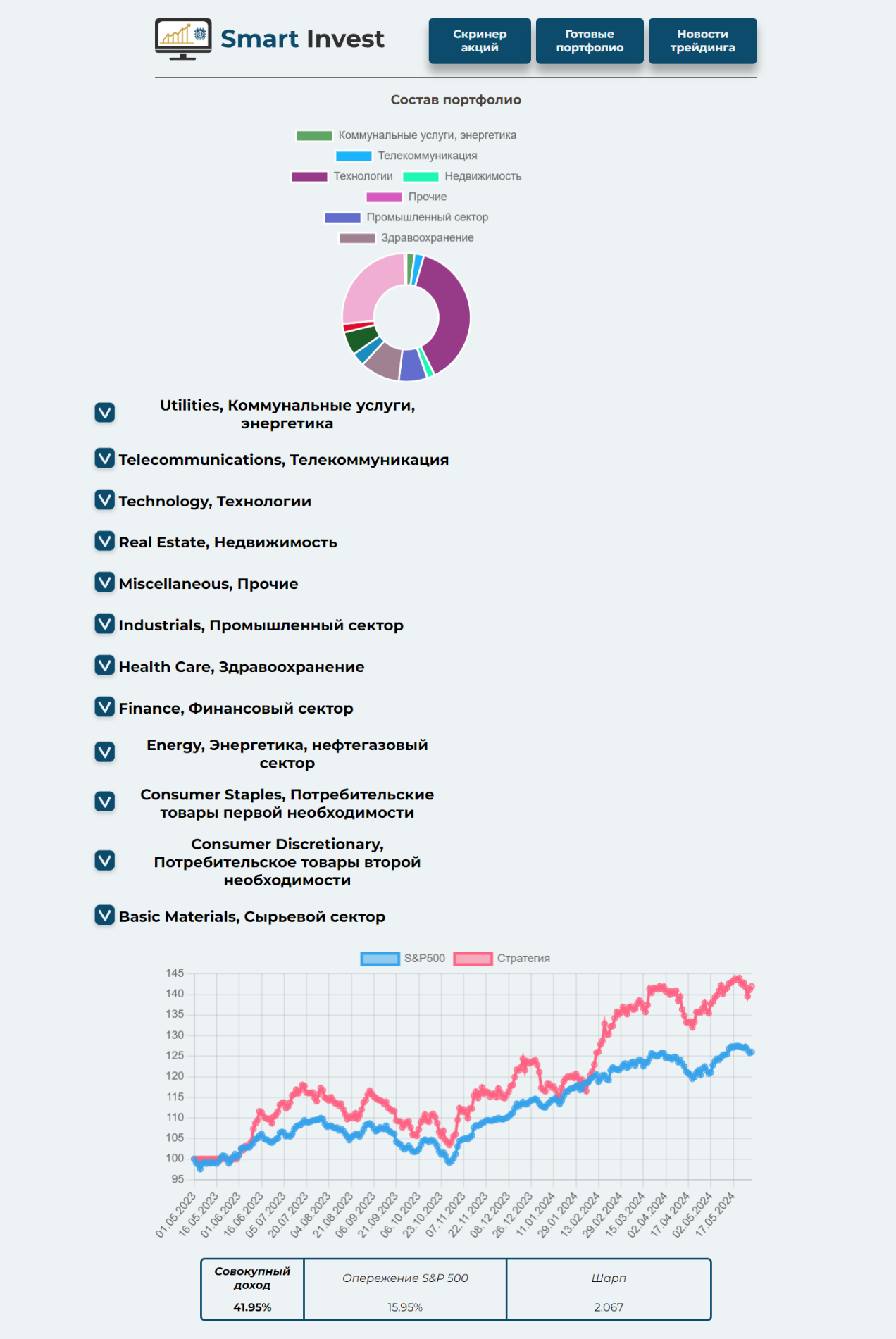


Рисунок 2 –Скриншот страницы рекомендованного портфолио