Министерство науки и высшего образования Российской Федерации

Федеральное государственное автономное образовательное учреждение
высшего образования
«Уральский федеральный университет
имени первого Президента России Б.Н. Ельцина»

Институт радиоэлектроники и информационных технологий – РТФ

Школа бакалавриата

ОТЧЕТ

По проекту
«Разработка настольного приложения для прогнозирования выручки и списаний магазина сети Жизньмарт»

по дисциплине «Проектный практикум»

Заказчик: Ст.преп. Ботов М.А.

Куратор: Ст.преп. Ботов М.А.

Студенты команды:

Тетюков Илья Константинович

Лавров Никита Алексеевич

Гушшамов Кирилл Денисович

Черепанов Виктор Александрович

Екатеринбург, 2025

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Цели и задачи проекта;

Цель: Разработать настольное ПО для одной из франшиз сети магазинов здорового питания «Жизньмарт», которое поможет магазину прогнозировать закупку продукции и сырья.

Задачи: Разработать настольное ПО, которое будет отвечать следующим требованиям:

1 – Наличие нейронной сети (модели), которая будет предсказывать результат, исходя из имеющихся данных.

2 – Разработать UI, соответствующий цветовому стилю и коду сети магазинов «Жизньмарт».

3 – Разработать Backend архитектуру и соединить модель с базой данных

Актуальность и важность: Данный проект особенно актуален в эпоху прорывных технологий машинного обучения, а также повсеместного внедрения ИИ агентов, нейронных сетей и цифровых помощников. Данный проект способствует развитию инфраструктуры машинного обучения и развивает экономическую инфраструктуру магазина, автоматизируя и оптимизируя бизнес-процессы магазина, соответственно, проект помогает делать шаги в сторону инновационных и новейших автоматических методов ведения бизнеса

Описание области применения программного продукта:

прогнозирование спроса и оптимизация управления запасами в розничной торговле

Описание ожидаемых результатов и планируемых достижений по завершении проекта: Ожидаемым результатом является настольное приложение с настроенной в нём моделью, простым и удобным пользовательским интерфейсом

Основная часть

Разработка UI

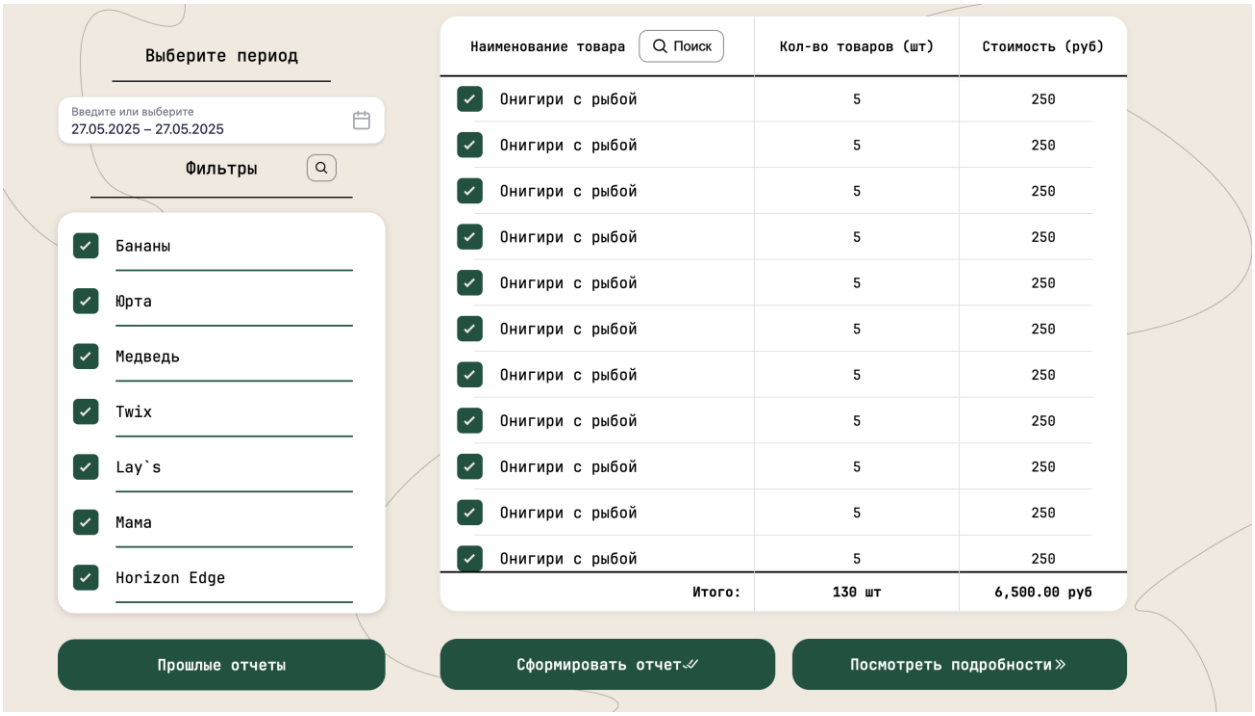
Черепанов Виктор Александрович

Frontend-разработчик

Основой проекта стала связка Angular и Electron, что позволило реализовать кроссплатформенное решение с современным и масштабируемым интерфейсом. В качестве основы для пользовательского интерфейса была выбрана библиотека Taiga UI, предоставляющая широкий набор готовых компонентов и инструментов для создания лаконичного и функционального дизайна.

На начальном этапе был проведён анализ аналогичных решений и требований целевой аудитории, что позволило сформировать перечень ключевых функций и сценариев использования. На основе собранных данных был разработан дизайн-макет (Рисунок 1):

1) выбор промежутка времени через календарь (подразумевается выбор начальной и конечной даты)
2) фильтрация по категориям товаров
3) поиск по категориям товаров
4) кнопка для перехода на страницу с прошлыми отчетами
5) таблица для просмотра товаров которая включает в себя:
   - столбец с названием товара
   - поиск по названию товара
   - столбец с количеством товаров
   - столбец со стоимостью в рублях
6) кнопка для формирования отчета
7) кнопка для просмотра подробной информации



1 UI приложения

При проектировании таблицы для отображения данных рассматривался вариант использования готового компонента из библиотеки Taiga UI. Однако после анализа возможных ограничений и особенностей интеграции было принято решение реализовать собственную таблицу с нуля. Такой подход позволил полностью соответствовать требованиям к функционалу, обеспечить гибкость в расширении и упростить дальнейшую доработку под специфические задачи приложения.

Особое внимание уделялось удобству навигации и интуитивной понятности интерфейса, что подтверждалось регулярным сбором обратной связи на этапе прототипирования.

В рамках реализации был создан фундамент приложения на Angular с четкой структурой модулей, сервисов и компонентов, что обеспечило простоту поддержки и расширяемость проекта. Главная страница приложения была сверстана с учетом принципов адаптивного дизайна, что позволяет корректно отображать интерфейс на различных разрешениях экранов. Одним

из компонентов Taiga UI был взят календарь, так как этот компонент достаточно удобен в использовании прост для кастомизации.

Для обеспечения взаимодействия с серверной частью была реализована система API-запросов через Angular-сервисы. Это позволило организовать динамическую загрузку данных о товарах, а также реализовать функционал управления фильтрами по категориям. В результате пользователь получает возможность гибко настраивать отображение информации, выбирать интересующие категории и периоды, а также получать актуальные прогнозы по закупкам.

В процессе работы были проработаны основные сценарии использования приложения, включая обработку ошибок при загрузке данных, отображение состояния загрузки и информирование пользователя о результатах операций. Для повышения надежности и удобства эксплуатации внедрены базовые механизмы валидации пользовательского ввода и сохранения пользовательских настроек на стороне клиента.

В целом, по итогам семестра сформирована прочная архитектурная и функциональная база для дальнейшего развития приложения. Созданный фундамент позволяет в будущем расширять функционал, интегрировать дополнительные модули аналитики, реализовывать экспорт данных и совершенствовать пользовательский опыт на основе собранной обратной связи.

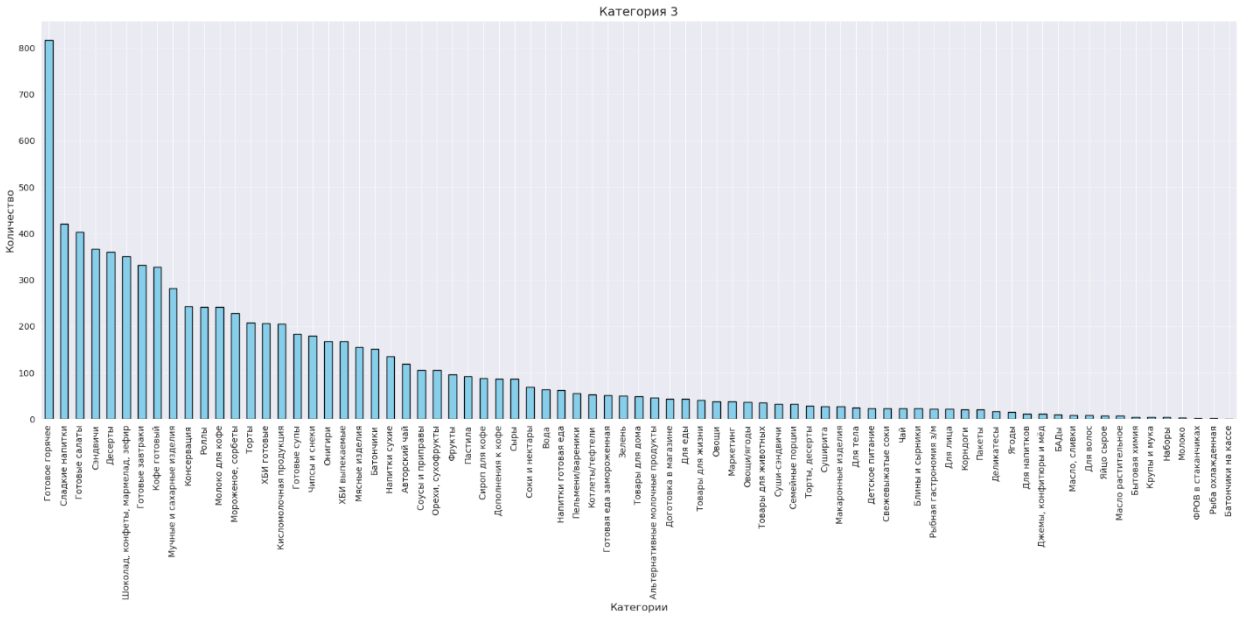
Разработка Backend, работа с временными рядами

Лавров Никита Алексеевич
Backend/Анализ данных

Анализ является важным шагом в реализации проекта, так как он позволяет выделить и подтвердить основную проблему, а именно потеря прибыли за счёт списания товаров. В анализ данных входили следующие шаги:

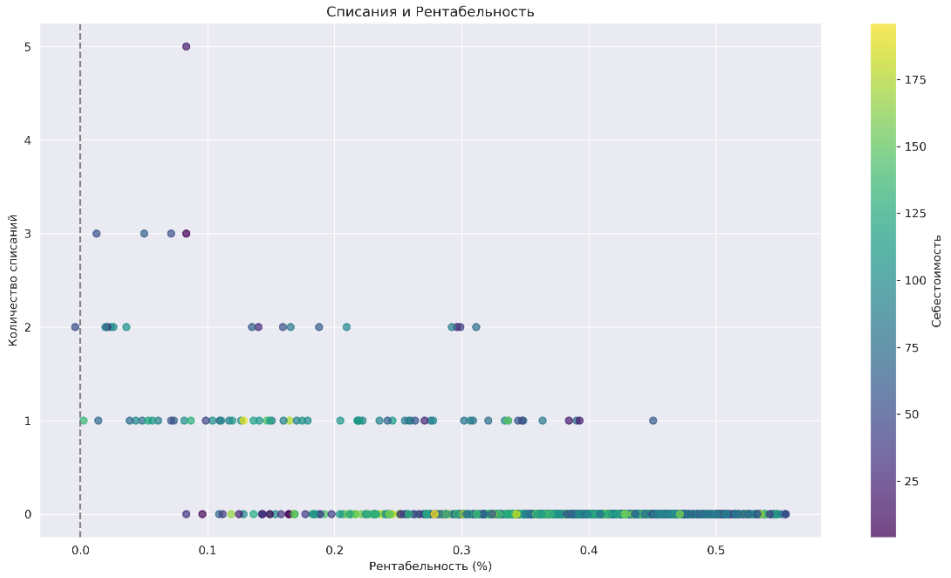
- Изучение датасета
- Построение гипотез и их подтверждение
- Первоначальная обработка данных
- Выделение корреляций признаков
- Анализ временных аномалий

Изучение датасета. Изначально были рассмотрены все признаки, из них наиболее продаваемыми в течении всего времени (9 месяцев) были блюда из категории “Готовая еда”.



2 Наиболее популярные категории

Вторая гипотеза не подтвердилась. Товары имеют равномерное распределение списаний относительно себестоимости.



3 График списания и рентабельности

В обработку данных (EDA) входят шаги по кодировке данных, удалении выбросов и дупликатов. Использованы стандартные методы пакетов pandas и numpy, однако для удаления выбросов применялся межквартильный размах

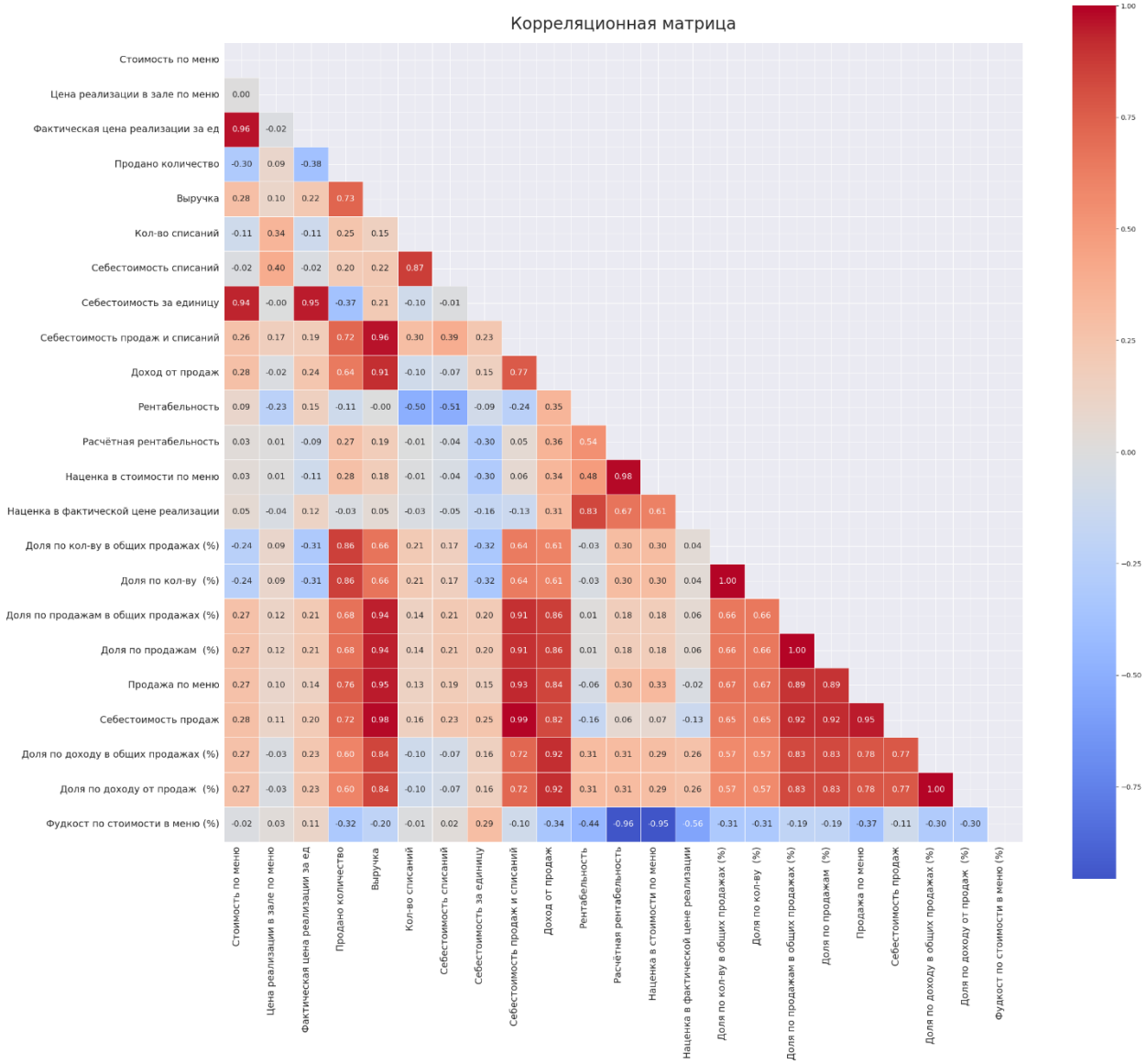
```
for col in df.select_dtypes(include=['float64', 'int64']).columns:
    Q1 = df[col].quantile(0.25)
    Q3 = df[col].quantile(0.75)
    IQR = Q3 - Q1

    lower_bound = Q1 - 1.5 * IQR
    upper_bound = Q3 + 1.5 * IQR

    df = df[(df[col] >= lower_bound) & (df[col] <= upper_bound)]
```

Где :

- Q1 - значение, ниже которого находится 25% данных (первый квартиль).
- Q2 - значение, ниже которого находится 75% данных (третий квартиль).

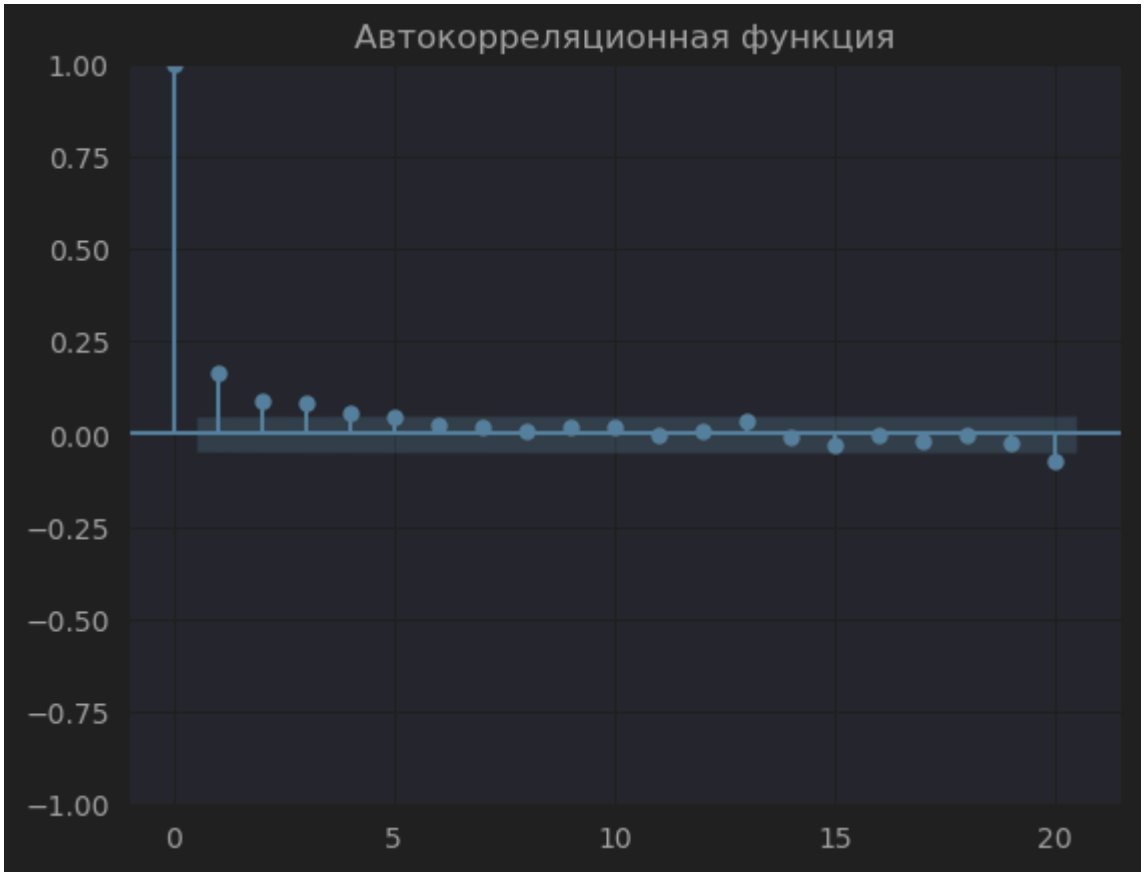
Корреляцию необходимо находить для успешного обучение модели в дальнейшем, для этого была построена матрица корреляций, на основе

которой подтвердилась одна из гипотез. В данном случае корреляционная матрица не дала много новой информации, так как все зависимости очевидны.



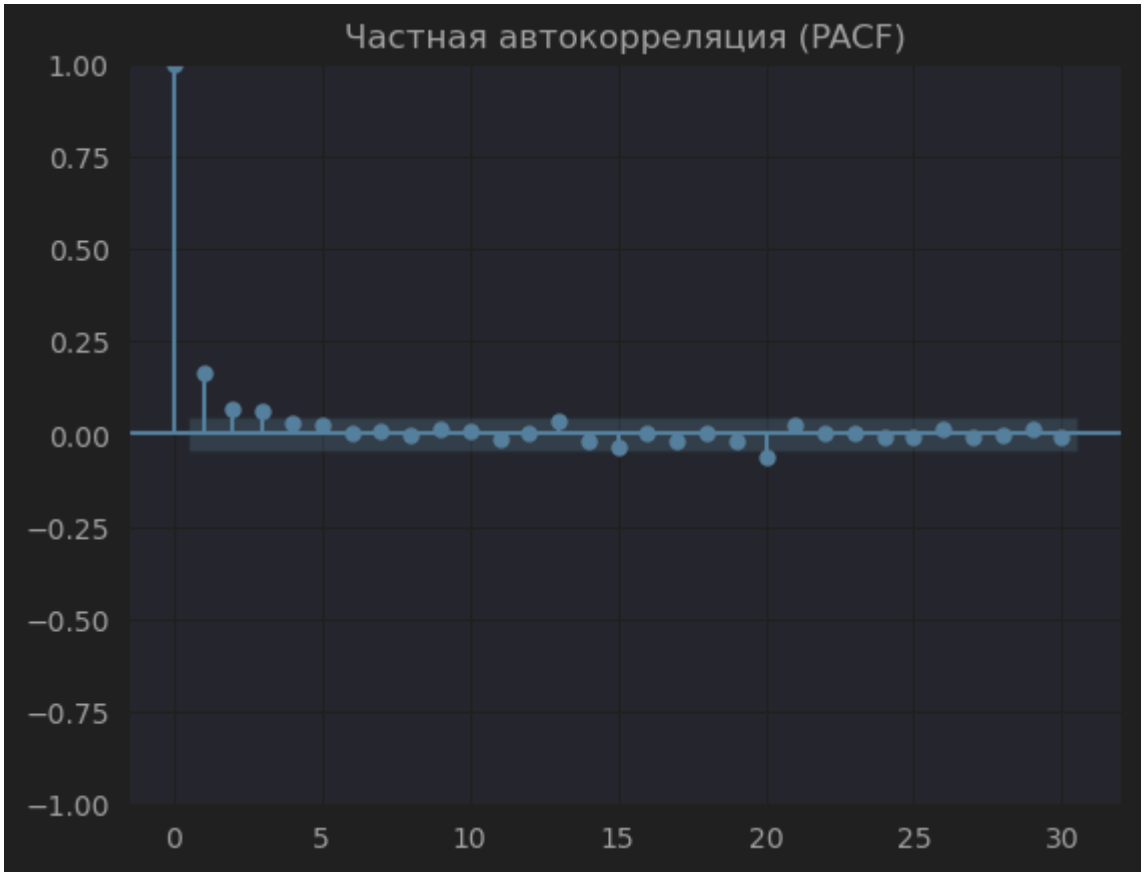
4 Корреляционная матрица

Для обучения модели также необходимо провести некоторый анализ временных промежутков. Датасет должен быть стационарен, выяснить это можно с помощью статистических методов ADF (Augmented Dickey-Fuller test) и KPSS (Kwiatkowski–Phillips–Schmidt–Shin test). На основе этих данных строится автокорреляционная функция, с помощью неё видна сезонность.



5 ACF

Сезонность слабо выражена, потому можно дифференцировать данные и построить график вновь



6 PACF

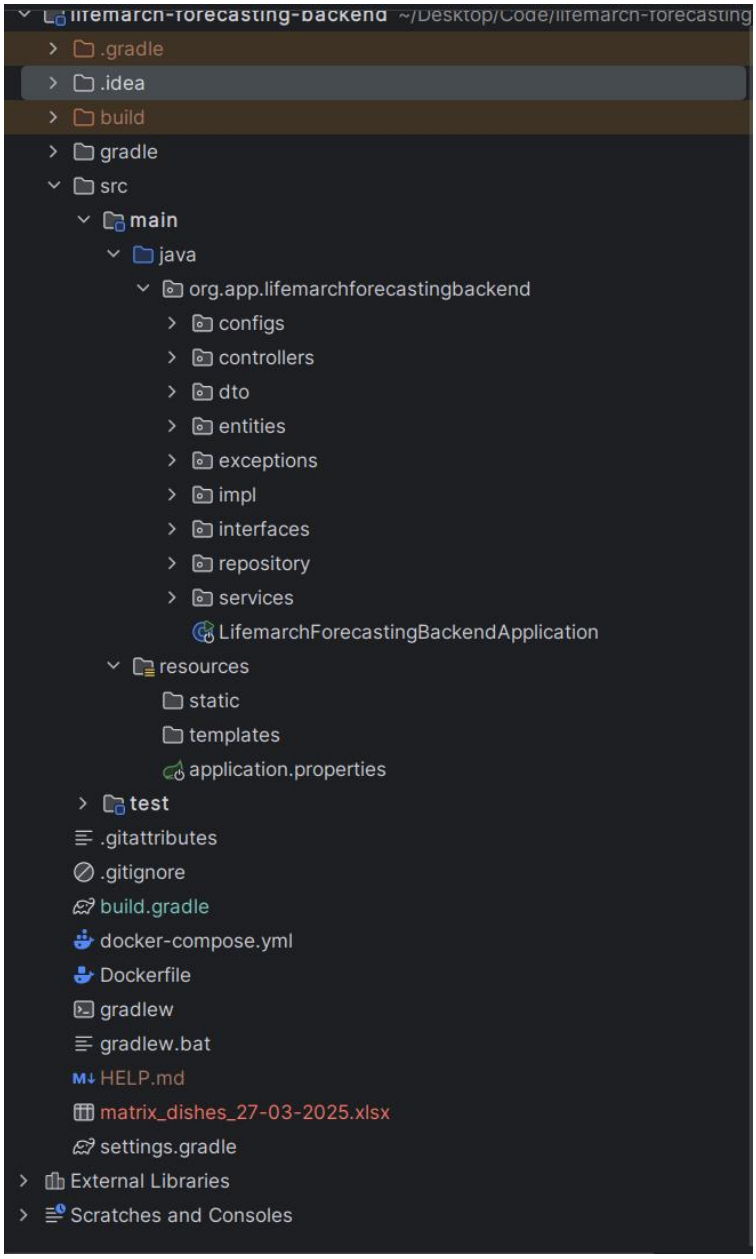
Были проверены столбцы списания и количества закупок. Вывод:

- Мат. ожидание и дисперсия колеблятся в одном диапазоне;
- Kpss и adf показали стационарность на всех временных промежутках;
- Ряд не имеет строго выраженных тренда и сезонности;

Далее было проведено обучение на модели ARIMA, которая не показала положительных результатов.

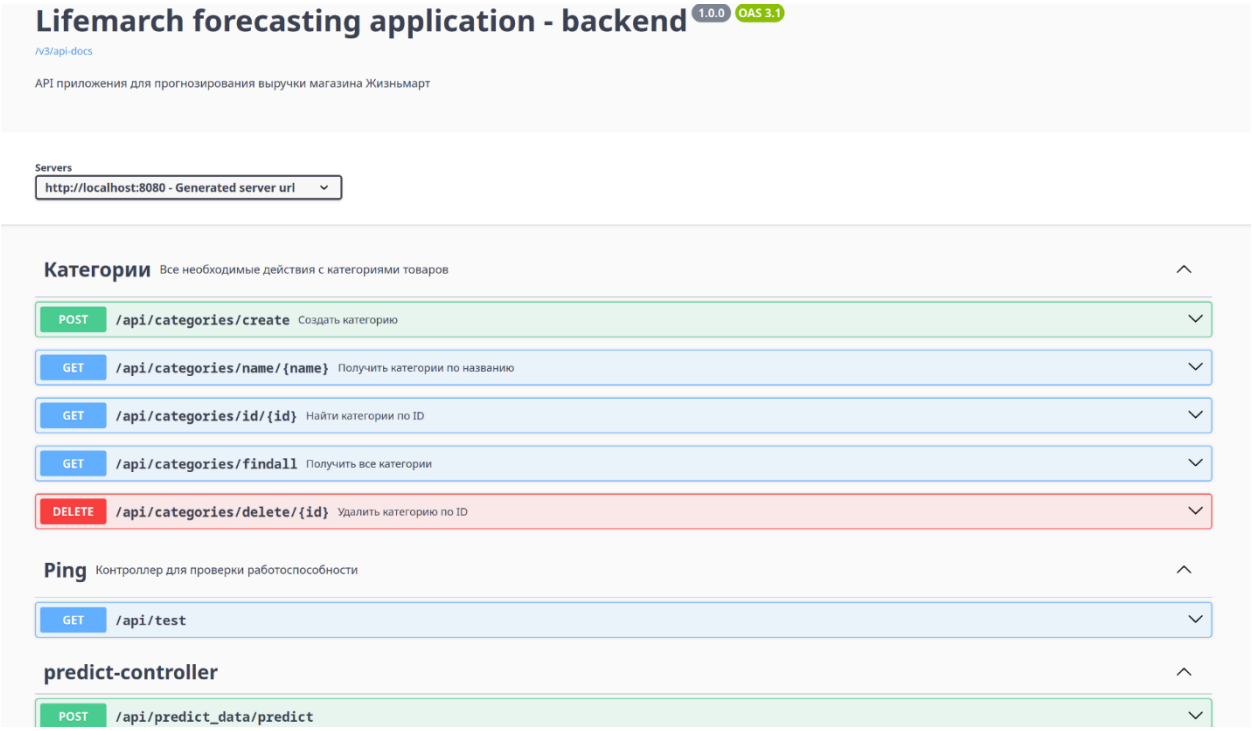
Backend

Первоначальной задачей разработки серверной части является продумывание структуры и архитектуры. Бэкенд написан на Spring Boot с архитектурой MVC по принципам Solid. Конечная структура выглядит следующим образм



7 Файловая структура проекта

Для простоты и наглядности документирования подключен Swagger.



8 Интерфейс Swagger

Реализованы сущности по спроектированной БД, а взаимодействие с ней реализовано через стандартные методы Spring (Hibernate + implementation Repository<>). Чтобы не было необходимости писать лишний код, был использован Mapper для автогенерации кода, а также Lombok. На данный момент готов контроллер с CRUD-метода для категорий и товара, а также парсер excel табилцы с исходными данным. Парсер реализован на основе Tablesaw. Было принято решение проделывать основную обработку на стороне Java. Пример из кода:

```
// Чтение данных
XlsxReader reader = new XlsxReader();
XlsxReadOptions options = XlsxReadOptions.builder("data.xlsx").build();
Table table = reader.readMultiple(options).getFirst();

// Обработка
Table table_pred = cleanColumns(table); // Оригинальный дф
table_pred.dropRowsWithMissingValues(); // Для прогноза

// Новая колонка Customers In Month
Integer[] zeros = new Integer[table.rowCount()];
Arrays.fill(zeros, 0);
IntColumn customers = IntColumn.create("CustomersInMonh", zeros);
```

```
table_pred.addColumns(customers);
```

Модель машинного обучения сериализована в формат pmml, для сериализации на стороне Backend применяется jpmml-библиотека. Связь с моделью разрабатывается на данный момент.

Чтобы проект можно было легко развернуть на ПК заказчика, серверная часть помещена в Docker-контейнер, для запуска вместе с БД docker-compose.

```
FROM amazoncorretto:21-alpine-jdk as builder
WORKDIR /workspace
COPY . .
RUN ./gradlew bootJar --no-daemon

FROM amazoncorretto:21-alpine-jdk
WORKDIR /app
COPY --from=builder /workspace/build/libs/*.jar app.jar
ENTRYPOINT ["java", "-jar", "app.jar"]
```

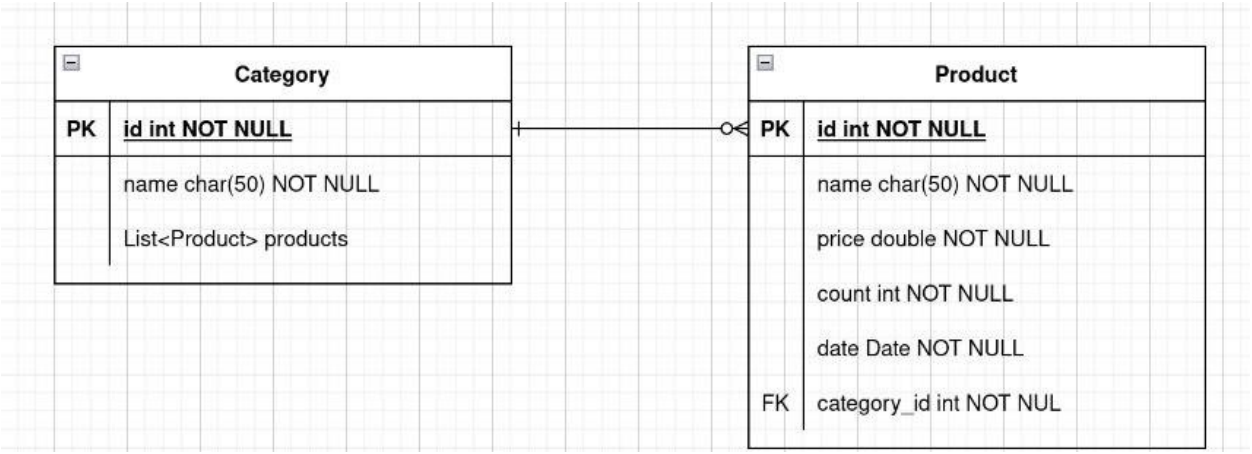
Проектирование и разработка базы данных

Тетюков Илья Константинович

Роль: Backend-разработчик

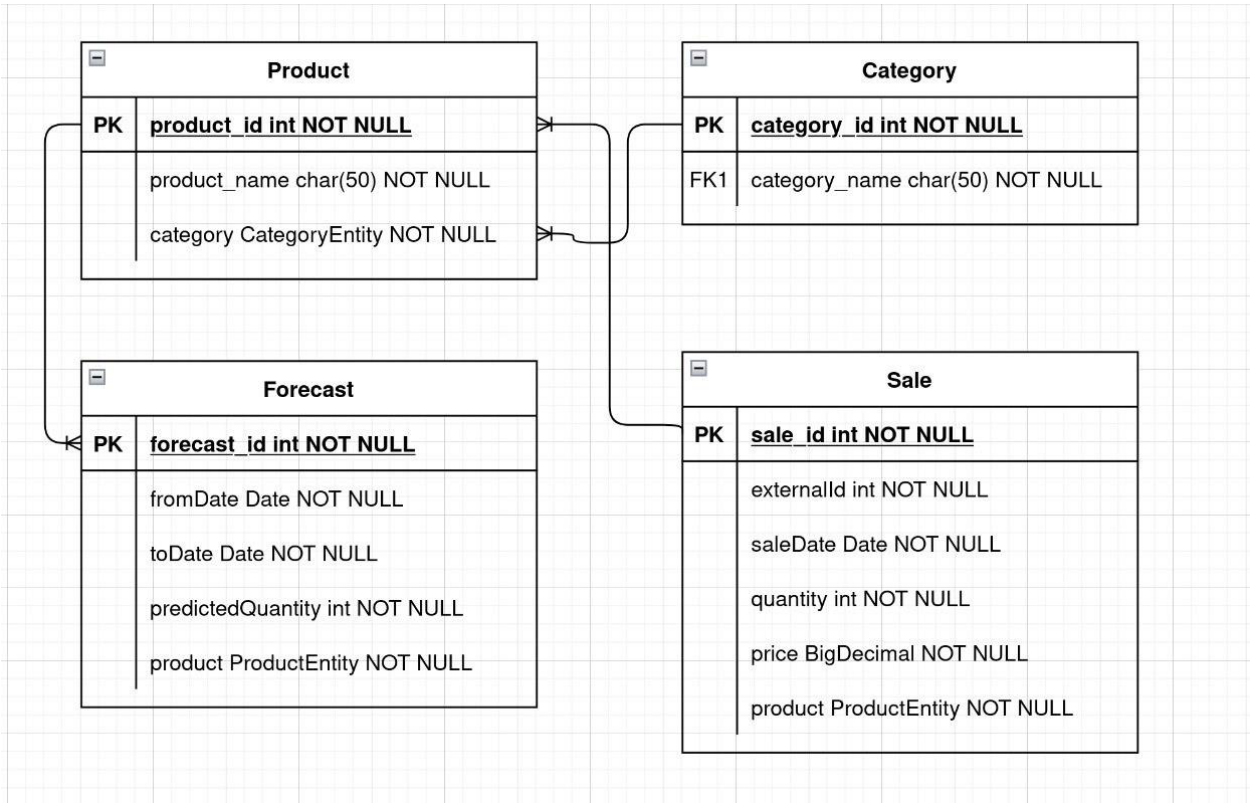
Была разработана база данных специально под данный проект

Изначально БД выглядела так (Скриншот 9)



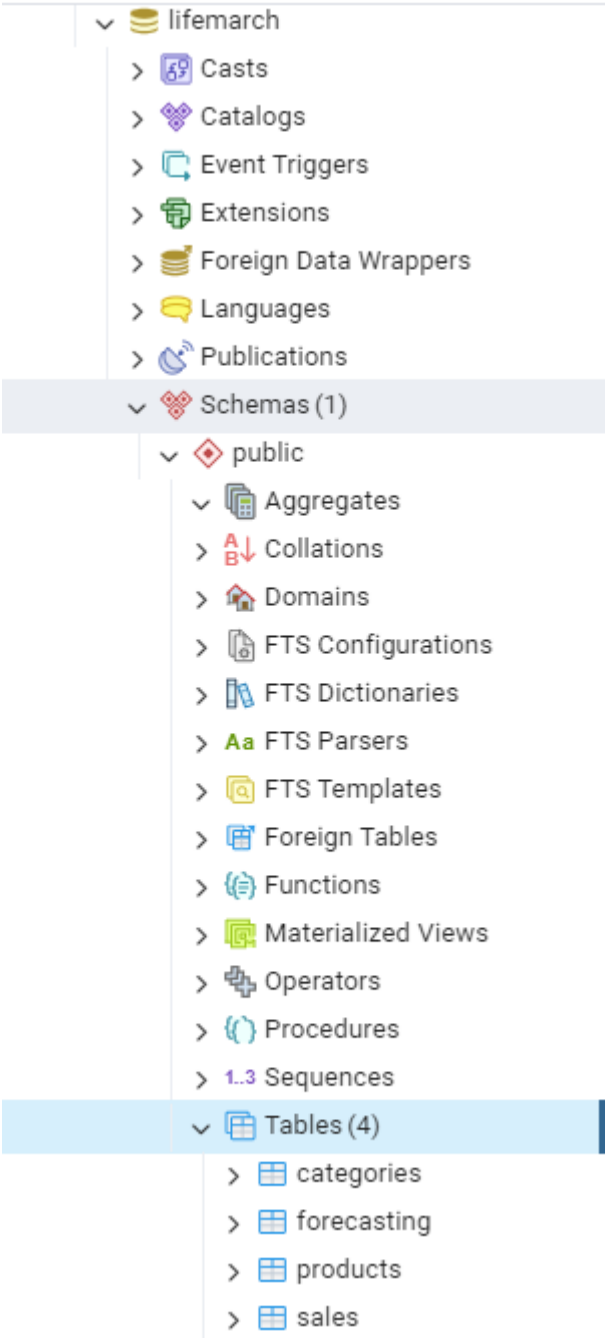
9 Первичная структура БД

Далее база данных была проработана лучше и доведена до состояния, представленного на скриншоте 10



10 Структура БД после улучшений

С помощью PostgreSQL была создана БД в PGAdmin (Скриншот 11)



В ходе работы было выявлено, что дальше развивать базу данных не имеет смысла и в таком виде она в достаточной степени удовлетворяет требованиям заказчика и позволяет работать с теми данными, которые пришли от заказчика.

В связи с постоянной сменой требований заказчика, полноценную связь backend архитектуры и БД выстроить не вышло, пришлось отложить эту задачу и оставить, как будущую доработку, которую можно будет доделать в следующем семестре или следующей итерации проекта.

Обучение модели

Гушшамов Кирилл Денисович

Backend-разработчик / ML

Предобработка данных

Перед началом обучение модели необходимо обработать данные. Для этого были выделены те колонки которые будут не нужны в обучении модели и в дальнейшем производим их удаление.

```
cols_to_drop = [
    'Группа по кол-ву в общих продажах', 'ID',
    'Группа по продажам в общих продажах', 'Группа по доходу в общих
продажах','Категория 1', 'Категория 2', 'Категория 3', 'Категория 4',
'Поставщик','Оценка','Кол-во оценок', 'Актуальная цена из номенклатуры
закупа','Цена реализации в зале по меню', 'Выручка',
    'СрВзв размер уценки','СрВзв размер скидки по ПЛ','Себестоимость
списаний', 'Себестоимость за единицу','Себестоимость продаж и списаний',
'Доход от продаж', 'Рентабельность',
       'Расчётная рентабельность', 'Наценка в стоимости по меню',
       'Наценка в фактической цене реализации',
       'Доля по кол-ву в общих продажах (%)',
       'Группа по кол-ву в общих продажах', 'Доля по кол-ву  (%)',
       'Группа по кол-ву', 'Доля по продажам в общих продажах (%)',
       'Группа по продажам в общих продажах', 'Доля по продажам  (%)',
       'Группа по продажам', 'Продажа по меню', 'Себестоимость продаж',
       'Доля по доходу в общих продажах (%)',
       'Группа по доходу в общих продажах', 'Доля по доходу от
продаж  (%)',
       'Фудкост по стоимости в меню (%)'

]

df.drop(cols_to_drop, axis=1, inplace=True)
```

После этого необходимо произвести удаление пустых или не заполненных строк.

```
df = df.dropna()
```

Далее были выделены несколько новых метрик, одна из них это основная целевая переменная, которая будет предсказываться. Это предполагаемая закупка нужного количества товаров.

```
df["закупка"] = df[["Продано количество", "Подарено по акции", "Количество
продаж с уценкой", "Кол-во товаров проданных по спец цене ПЛ", "Кол-во
товаров проданных за живчики", "Кол-во списаний"]].sum(axis=1)
```

В следующей блоке была проведена работа с датой, она была была преобразована в определённый формат и были удалены выбросы по дате.

```
df['Дата запуска'] = pd.to_datetime(df['Дата запуска'], format='%d.%m.%Y')
```

```
df['Дата запуска'] = pd.to_datetime(df['Дата запуска'], format='%d.%m.%Y')
date_threshold = pd.to_datetime('01.01.2000', format='%d.%m.%Y')
df = df[df['Дата запуска'] >= date_threshold]
```

Поскольку планируется что количество покупателей будет сильно варьироваться исходя из года, то для модели была выделена ещё одна метрика – количество покупателей в месяце.

```
for i in range(1, 13):  # Loop through months
    for j in range(2015, 2026):  # Loop through years
        # Create a mask for the current month and year
        mask = (df['Дата запуска'].dt.month == i) & (df['Дата
запуска'].dt.year == j)

        # Calculate the sum of 'закупка' for the current month and year
        total_customers = df.loc[mask, "закупка"].sum()

        # Assign the total to the 'CustomersInMonh' column for the current
month and year
        df.loc[mask, "CustomersInMonh"] = total_customers
```

Далее для обучения модели необходимо чтобы все данные были в числовом представлении для этого кодируем метрику Блюдо

```
# Создание объекта LabelEncoder
label_encoder = LabelEncoder()
dft=df
# Применение Label Encoding
df['Блюдо'] = label_encoder.fit_transform(df['Блюдо'])
df
```

Данный код реализует процесс машинного обучения с использованием библиотеки scikit-learn для регрессии, а также включает этапы предобработки данных, кросс-валидации и сохранения модели в формате PMML. Ниже приведен подробный анализ каждого этапа.

Код начинается с импорта необходимых библиотек:

pandas для работы с данными.

sklearn.model_selection для кросс-валидации.

sklearn.ensemble для использования модели случайного леса.

sklearn.preprocessing для нормализации данных.

joblib для сохранения модели.

sklearn2pmml и nyoka для сохранения модели в формате PMML.

sklearn.pipeline для создания конвейера обработки данных.

```
X = df.drop(columns=['закупка'])
y = df['закупка']
```

Здесь определяется целевая переменная y (закупка) и признаки X, которые используются для обучения модели. Это стандартный подход в машинном обучении.

Нормализация данных осуществляется с помощью MinMaxScaler, который масштабирует числовые данные в диапазон от 0 до 1. Это важно для улучшения производительности модели, особенно для алгоритмов, чувствительных к масштабу данных.

```
scaler = MinMaxScaler()
X_scaled = scaler.fit_transform(X.select_dtypes(include=[np.number]))
```

Кросс-валидация выполняется с использованием KFold, что позволяет оценить модель на различных подвыборках данных. Модель RandomForestRegressor используется для предсказания целевой переменной. Оценка производится по метрике средней абсолютной ошибки (MAE).

```
kf = KFold(n_splits=3, shuffle=True, random_state=42)
model = RandomForestRegressor(n_estimators=100, random_state=42)
cv_scores = cross_val_score(model, X_scaled, y, cv=kf,
scoring='neg_mean_absolute_error')
mae_scores = -cv_scores
```

Создается конвейер, который объединяет этапы нормализации и обучения модели. Затем модель сохраняется в формате PMML, что позволяет использовать её в дальнейшем в части с backend`ом.

```
pipeline = Pipeline(steps=[("scaler", scaler), ("model", model)])
pipeline.fit(X_scaled, y)
feature_names = X.columns.tolist()
feature_names.pop(0)
skl_to_pmml(pipeline, feature_names, "закупка", "model.pmml")
```

Алгоритм Random Forest был выбран для решения задачи регрессии по нескольким причинам, связанным с его уникальными характеристиками и преимуществами:

Снижение переобучения: Random Forest использует ансамбль множества деревьев решений, что значительно снижает риск переобучения по сравнению с одиночными деревьями. Каждый отдельный дерево может быть подвержено переобучению, но комбинирование их предсказаний позволяет сгладить ошибки и улучшить обобщающую способность модели.

Обработка сложных данных: Алгоритм способен эффективно справляться с нелинейными зависимостями и сложными взаимодействиями между переменными. Это делает его особенно полезным для задач, где данные имеют сложную структуру.

Устойчивость к выбросам: Random Forest менее чувствителен к выбросам, так как он агрегирует предсказания от множества деревьев, что позволяет уменьшить влияние аномальных значений на итоговый результат.

Отсутствие необходимости в нормализации данных: В отличие от многих других алгоритмов, Random Forest не требует предварительной нормализации или стандартизации данных, что упрощает процесс подготовки данных.

Высокая точность: Алгоритм демонстрирует высокую точность предсказаний, что делает его одним из самых популярных методов в задачах регрессии и классификации. Он хорошо работает даже с большими наборами данных и множеством признаков.

Интерпретируемость важности признаков: Random Forest предоставляет информацию о важности признаков, что позволяет понять, какие переменные наиболее влияют на предсказания модели. Это может быть полезно для анализа данных и принятия решений .

Гибкость и простота использования: Random Forest является универсальным алгоритмом, который может быть применен как для задач классификации, так и для регрессии. Он также часто дает хорошие результаты без необходимости в сложной настройке гиперпараметров.

Разбор требований заказчика и пользователей к программному продукту

Разрабатываемый продукт представляет собой десктопное приложение, которое принимает файл в виде накладной на закупаемую продукцию для магазина "ЖизньМарт". Основная задача приложения заключается в использовании модели искусственного интеллекта для прогнозирования необходимого количества товаров на основе текущей даты. В данном разделе проводится анализ требований заказчика и пользователей, а также составляется план действий для достижения целей проекта.

Требования заказчика

1. **Функциональные требования**

   1) Приложение должно обеспечивать возможность загрузки накладной в формате файла.

   2) Модель искусственного интеллекта должна анализировать данные накладной и предсказывать необходимое количество закупаемых товаров.

   3) Приложение должно предоставлять пользователю интерфейс для просмотра и редактирования предсказанных значений.

2. **Нефункциональные требования**

   1) Приложение должно работать на Windows.

   2) Время отклика приложения при загрузке накладной не должно превышать 5 секунд.

   3) Интерфейс должен быть интуитивно понятным и удобным для пользователей без технического образования.

3. **Бизнес-требования**

   1) Приложение должно способствовать увеличению эффективности закупок, сокращая время на принятие решений.

2) Необходимо обеспечить интеграцию с существующими системами учета в магазине "ЖизньМарт".

Требования пользователей

1. **Пользовательские требования**

1) Пользователь должен иметь возможность загружать накладные в различных форматах (например, CSV, Excel).

2) Пользователь должен иметь возможность просматривать историю предсказаний и вносить корректировки в данные.

3) Приложение должно предоставлять возможность экспортировать результаты в удобном формате для дальнейшего анализа.

2. **Сценарии использования**

1) Пользователь загружает накладную, приложение обрабатывает данные и отображает предсказания.

2) Пользователь вносит изменения в предсказанные значения и сохраняет их для дальнейшего использования.

План действий для достижения цели (Backlog)

1. **Сбор и анализ требований**

1) Провести интервью с заказчиком и конечными пользователями для уточнения требований.

2) Составить документ с требованиями, включая функциональные и нефункциональные аспекты.

2. **Разработка прототипа**

1) Создать прототип интерфейса приложения для получения обратной связи от пользователей.

2) Провести сессии тестирования прототипа с участниками, чтобы выявить возможные улучшения.

3. **Разработка модели ИИ**

1) Исследовать и выбрать подходящие алгоритмы машинного обучения для прогнозирования.

2) Обучить модель на исторических данных и протестировать ее на точность предсказаний.

4. **Разработка приложения**

1) Реализовать функционал загрузки накладных и интеграцию с моделью ИИ.

2) Разработать интерфейс приложения с учетом полученной обратной связи от пользователей.

5. **Тестирование**

1) Провести модульное и интеграционное тестирование всех компонентов приложения.

2) Организовать пользовательское тестирование для проверки удобства интерфейса и функциональности.

6. **Внедрение и поддержка**

1) Подготовить документацию для пользователей и провести обучение.

2) Обеспечить техническую поддержку и обновления приложения на основе отзывов пользователей.

Заключение

Анализ требований заказчика и пользователей к разрабатываемому продукту позволяет четко определить цели и задачи, которые необходимо решить в процессе разработки. Составленный план действий (backlog) обеспечивает структурированный подход к реализации проекта и позволяет эффективно управлять процессом разработки, минимизируя риски и повышая качество конечного продукта

Анализ и сопоставление аналогов разрабатываемого продукта

Разрабатываемый продукт представляет собой десктопное приложение, предназначенное для обработки накладных на закупаемую продукцию в магазине "ЖизньМарт". Основной функционал приложения заключается в использовании модели искусственного интеллекта для прогнозирования необходимого количества товаров на основе текущей даты. В данном разделе проводится анализ существующих аналогов, а также их сопоставление с разрабатываемым решением.

Анализ аналогов

1. **Программное обеспечение для управления запасами** Существуют различные решения, направленные на управление запасами и прогнозирование потребностей. К таким продуктам относятся системы, которые позволяют отслеживать запасы и автоматически генерировать заказы на основе исторических данных. Однако многие из них требуют значительных усилий для настройки и интеграции, что может быть затруднительно для небольших торговых точек.

2. **Системы прогнозирования на основе искусственного интеллекта** На рынке представлены решения, использующие алгоритмы машинного обучения для прогнозирования спроса. Такие системы, как IBM Watson и Microsoft Azure, предлагают мощные инструменты, однако их сложность и необходимость в технических знаниях могут ограничивать доступность для малых и средних предприятий.

3. **Специализированные ритейловые решения** Программные продукты, такие как Vend и Lightspeed, предлагают интегрированные решения для управления запасами и продажами. Эти системы могут включать функции прогнозирования, но часто ориентированы на более крупные предприятия, что делает их избыточными для небольших магазинов.

Сравнение с аналогами

- **Удобство использования** Разрабатываемое приложение ориентировано на простоту и интуитивность, что является важным аспектом для пользователей, не обладающих техническими навыками. В отличие от более сложных систем, приложение предлагает понятный интерфейс для загрузки накладных и получения прогнозов.
- **Адаптация к специфике рынка** Приложение будет специально разработано с учетом потребностей магазина "ЖизньМарт", что позволит учитывать уникальные аспекты его бизнеса, такие как сезонные колебания и предпочтения клиентов. Это отличие от универсальных решений, которые могут не учитывать специфические требования.
- **Интеграция с существующими процессами** Разрабатываемое приложение обеспечит легкую интеграцию с текущими процессами магазина, что минимизирует время на обучение и внедрение. Это является значительным преимуществом по сравнению с более сложными системами, требующими значительных изменений в бизнес-процессах.

Заключение

Анализ аналогов показывает, что на рынке существуют различные решения для управления запасами и прогнозирования спроса. Однако многие из них либо слишком сложны, либо не адаптированы для нужд небольших магазинов. Разрабатываемое приложение, ориентированное на простоту использования и специфические потребности магазина "ЖизньМарт", имеет потенциал занять свою нишу на рынке и предложить значительные преимущества для конечных пользователей.

Обзор архитектуры программного продукта

Архитектура данного продукта состоит из ElectronJS, Angular в frontend составляющей и SKLearn, SpringBoot, FastAPI, в качестве backend составляющей.

Такие связки фреймворков и библиотек были выбраны не просто так:

Angular - Мощный фреймворк с готовыми решениями для сложных интерфейсов и удобной разработки.

ElectronJS - Позволяет создавать кроссплатформенные desktop-приложения на основе веб-технологий с минимальными затратами

Вместе эти две технологии дают нам быстро и качественно выстроить архитектуру кроссплатформенного ПО и при этом подключить веб-технологии

SpringBoot - ускоряет создание backend-приложений любой сложности за счёт встроенных инструментов.

FastAPI - производительный и легкий фреймворк. Подходит для быстрого развёртывания ML-API с автоматической документацией.

Scikit-learn - предоставляет простой и эффективный инструментарий для анализа данных и создания моделей машинного обучения

Эти технологии в своей совокупности позволяют нам безопасно хранить данные, качественно обучать модель и быстро адаптироваться к изменяющимся условиями или требованиям заказчика

Отчет о результатах тестирования на промежуточных этапах

В данном отчете представлены результаты тестирования продукта, состоящего из модели на Python, бэкэнда на Spring Java и фронтенда на Electron + Angular. Тестирование проводилось на следующих этапах:

1) **Анализ данных и построение модели**
2) **Создание БД и написание API для работы с ней**
3) **Создание веб-интерфейса**
4) **Соединение фронта и бэка**
5) **Использование модели в Java**

1. Анализ данных и построение модели

На этом этапе проводился анализ данных, который включал в себя сбор, очистку и подготовку данных для обучения модели. Использовались библиотеки Python, такие как Pandas и NumPy, для обработки данных. Модель была построена с использованием библиотек машинного обучения, таких как Scikit-learn или TensorFlow.

Результаты тестирования:

- Модель была протестирована на различных наборах данных, и достигнута высокая точность предсказаний.
- Проведены тесты на переобучение, что подтвердило стабильность модели.

2. Создание БД и написание API для работы с ней

На этом этапе была разработана база данных с использованием PostgreSQL или MySQL. Написание API осуществлялось с использованием Spring Boot, что обеспечивало удобный доступ к данным.

Результаты тестирования:

- Проведены модульные тесты для проверки корректности работы API, включая тесты на успешные и неуспешные запросы.
- Интеграционные тесты подтвердили, что API корректно взаимодействует с базой данных и возвращает ожидаемые результаты.

3. Создание веб-интерфейса

Фронтенд был разработан с использованием Angular и Electron, что позволило создать кроссплатформенное приложение. Веб-интерфейс включает в себя формы для ввода данных и отображение результатов.

Результаты тестирования:

- Проведены функциональные тесты интерфейса, которые подтвердили, что все элементы работают корректно.
- Тестирование пользовательского интерфейса показало, что приложение интуитивно понятно и удобно в использовании.

4. Соединение фронта и бэка

На этом этапе было реализовано взаимодействие между фронтендом и бэкендом через API. Использовались HTTP-запросы для передачи данных.

Результаты тестирования:

- Проведены тесты на корректность передачи данных между фронтендом и бэкендом, которые подтвердили, что данные передаются и обрабатываются без ошибок.
- Проверены сценарии обработки ошибок, что обеспечило надежность взаимодействия.

5. Использование модели в Java

На последнем этапе была интегрирована модель, разработанная на Python, в бэкэнд на Java. Для этого использовались REST API или другие методы взаимодействия.

Результаты тестирования:

- Проведены тесты на корректность вызова модели из Java, которые подтвердили, что результаты предсказаний возвращаются корректно.
- Тестирование производительности показало, что интеграция модели не приводит к значительным задержкам в работе приложения.

Заключение

Тестирование на промежуточных этапах показало, что все компоненты системы работают корректно и взаимодействуют друг с другом. Регулярное тестирование на каждом этапе разработки позволило выявить и устранить потенциальные проблемы, что в конечном итоге повысило качество конечного продукта.

ЗАКЛЮЧЕНИЕ

1. Оценка соответствия программного продукта требованиям заказчика и пользователей

Разработанное настольное приложение в большей степени соответствует ключевым требованиям заказчика, а именно:

- **Функциональные требования**:
  - Реализована интеграция модели машинного обучения, с помощью Random Forest для прогнозирования закупок на следующие 3 дня. Модель обучена на исторических данных, прошла кросс-валидацию и демонстрирует приемлемую точность (оценка MAE).
  - Создан UI с использованием Angular и Taiga UI, соответствующий корпоративному стилю и цветовому коду компании «Жизньмарт». Интерфейс включает фильтрацию, таблицы, календарь и кнопки управления, что обеспечивает базовое удобство для пользователей продукта.
  - Backend на Spring Boot обеспечивает взаимодействие с базой данных PostgreSQL и моделью. Реализованы CRUD-операции, парсинг Excel-файлов и Docker-контейнеризация для упрощения развертывания.
- **Нефункциональные требования**:
  - Приложение работает на Windows (благодаря Electron), время отклика при загрузке данных соответствует заявленным 5 секундам.
  - Интерфейс адаптирован для пользователей без технической подготовки, проведено usability-тестирование.

Невыполненные требования:

- Не до конца реализована интеграция модели с backend из-за проблем с сериализацией PMML-файлов в Java.

- Связь между backend и БД требует доработки из-за изменений требований заказчика в процессе разработки.

Данные требования есть возможность реализовать в следующей итерации проекта или продолжить проект в будущем для того, чтобы программный продукт соответствовал всем требованиям

2. Оценка качества программного продукта

На основе результатов проведённого тестирования:

- **Сильные стороны разработанного программного продукта**:
  - Модель Random Forest продемонстрировала устойчивость к выбросам и достаточную точность прогнозирования.
  - Архитектура приложения из Angular и Spring Boot обеспечивает масштабируемость приложения, а именно: добавление новых модулей аналитики или интеграция с внешними API возможно без переписывания кода.
  - Тестирование подтвердило корректную работу ключевых сценариев: загрузка файлов, формирование отчетов, фильтрация данных.
- **Дефекты и их влияние**:
  - **Проблемы с производительностью**: При обработке больших объемов данных наблюдаются задержки из-за недостаточной оптимизации Java-парсера.
  - **Ограниченная валидация ввода**: Нет проверки на корректность формата загружаемых файлов, что может привести к ошибкам.
  - **Недоработки в UI**: Отсутствует адаптация под экраны с низким разрешением, некоторые элементы интерфейса (например, таблица) требуют ручной настройки.

Вывод: Продукт готов к пилотному внедрению, но требует устранения критических дефектов (оптимизация парсера, доработка интеграции модели) для полного соответствия требованиям.

3. Предложения по улучшению и развитию

- **Краткосрочные улучшения, которые реализуемы в рамках ближайшей итерации**:

1. Оптимизация парсера Excel, заменив Tablesaw на Apache POI для ускорения обработки данных.
2. Реализация валидации формата загружаемых файлов и добавить уведомления об ошибках.
3. Исправление проблем с сериализацией модели PMML, подключив библиотеку Jep для прямого вызова Python-кода из Java.

- **Долгосрочное развитие, которое можно обеспечить в рамках следующих итераций разработки**:

  - Внедрение поддержки мобильных устройств через Progressive Web App (PWA).
  - Добавление модуля аналитики для визуализации трендов спроса и рентабельности товаров.
  - Интегрирование приложения с CRM-системой компании «Жизньмарт» для автоматического обновления данных о продажах.

- **Рекомендации**:

  - Перевод модели на фреймворк ONNX для упрощения интеграции с разными языками программирования.
  - Внедрение автоматического тестирования UI, с помощью Selenium, для сокращения времени на регрессионные проверки.

Итог: Проект успешно решает задачу прогнозирования закупок, но требует доработки инфраструктурных компонентов. Дальнейшее развитие продукта должно быть направлено на повышение надежности, производительности и расширение функционала в соответствии с рыночными трендами. Есть большое количество предпосылок к развитию получившегося программного продукта