Geekbrains

**Реализация автоматизированной системы** **для анализа изменения**

**качества продукции с использованием модели лингвистического**

**анализа**

Программа: Искусственный интеллект. Специалист

Специализация: Разработка

Андрей Комарницкий, Татьяна Лукьянова

Улан-Удэ, Москва

2025

Оглавление

[Введение 2](#_Toc199495052)

[Глава 1 Автоматизированная система анализа отзывов 7](#_Toc199495053)

[1.1 Основные компоненты, графическое представление 7](#_Toc199495054)

[Глава 2 Обучение модели 11](#_Toc199495055)

[2.1 Описание датасета 11](#_Toc199495056)

[2.2 Разметка данных 12](#_Toc199495057)

[2.3 Обучение модели 18](#_Toc199495058)

[Глава 3 Практическая часть 22](#_Toc199495059)

[3.1 Сбор данных 24](#_Toc199495060)

[3.2 Подготовка датасета 26](#_Toc199495061)

[3.3 Сохранение данных в БД 27](#_Toc199495062)

[3.4 Обучение модели 30](#_Toc199495063)

[3.5 Предсказание результатов 38](#_Toc199495064)

[3.6 Формирование отчетов 40](#_Toc199495065)

[3.7 Описание DAG 41](#_Toc199495066)

[Выводы и рекомендации 43](#_Toc199495067)

[Заключение 46](#_Toc199495068)

[Список используемой литературы 47](#_Toc199495069)

[Приложения 48](#_Toc199495070)

# Введение

***«****Разум, однажды расширивший свои границы,*

*никогда не вернётся в прежние»*

Альберт Эйнштейн

Эта мысль великого ученого в наши дни, на наш взгляд, приобрела более глубокий смысл, потому как под «разумом» можно понимать не только человеческую способность к осмыслению и развитию, но и искусственный интеллект, которому тоже присущи данные качества. Если продолжать поиск аналогий, то можно отметить, что искусственный интеллект так же недостаточно хорошо изучен, как и человеческий разум. По-прежнему остается много вопросов и направлений для исследования. Но, если вопросы познания человеческого разума наиболее актуальны и интересны определенному кругу ученых, то изучением возможностей искусственного интеллекта, а также его применением в повседневной жизни заинтересована более широкая аудитория, которая постоянно пополняется. Почему же развитие искусственного интеллекта сегодня развивается так стремительно? Для ответа на данный вопрос стоит заглянуть в историю и понять, какие факторы повлияли на это.

Первые шаги в области изучения искусственного интеллекта были совершены еще в середине XX века. Одними из первых ученых, занимавшихся исследованием в данной области, были Алан Тьюринг и Джон Маккарти. Алан Тьюринг в 1950 г. предложил «тест Тьюринга», который должен был определить, может ли машина мыслить. Джон Маккарти впервые на конференции в Дартмутском колледже предложил использовать термин «искусственный интеллект». В те же годы были созданы первые программы, которые могли выполнять элементарные задачи, такие, например, как игра в шахматы или решение простых логических задач.

В следующие четыре десятилетия были созданы программы, которые могли имитировать диалог с человеком, первые нейронные сети и эффективные алгоритмы машинного обучения. Одним из первых значимых событий в области развития ИИ стала победа шахматного компьютера в 1997 году над чемпионом мира.

Но, наиболее стремительное развитие в области искусственного интеллекта, происходит в последние несколько лет, начиная с 2016 г, когда была основана компания «OpenAI», которая разработала высокоэффективные системы генеративного ИИ для работы с языковыми моделями, которые сегодня широко используются во всем мире. Основным фактором для скачка развития ИИ послужил прогресс в области технологических решений, применение графических процессоров и появление облачных сервисов. Постоянное совершенствование аппаратного обеспечения позволяет сократить время на обработку данных, наличие облачных сервисов предоставляет возможность сэкономить затраты и воспользоваться огромными мощностями для обучения моделей.

На данный момент многие компании-лидеры в области IT из разных стран вносят существенный вклад в развитие технологий и алгоритмов, создаются новые нейронные сети и развиваются уже существующие. Одним из положительных моментов этого стремительного прогресса является то, что многие технологические решения, построенные с применением искусственного интеллекта, являются доступными для использования в повседневной жизни (в готовом виде или и как инструмент для создания собственных нейронных сетей и их обучения для решения различных задач).

Учитывая столь высокую популярность искусственного интеллекта и стремительный рост количества компаний, применяющих его в своей деятельности, получение образования по данному направлению является очень перспективным и важным для специалистов, которые хотят обладать навыками использования инновационных технологий и успешно их применять.

При выборе темы для данной работы мы ставили перед собой цель взять не слишком сложную предметную область, но реализовать проект, который решает реальную задачу, полезную для бизнеса розничной компании, а также охватить как можно больше инструментов, которые были изучены в рамках учебной программы, чтобы закрепить знания. При реализации проекта были использованы современные инструменты и технологии.

Тема нашего проекта: «Реализация автоматизированной системы для анализа изменения качества продукции с использованием модели лингвистического анализа».

Актуальность данной темы обусловлена тем, что в рамках проекта реализовано решение для одного из самого востребованного направления в области розничной торговли – продажи продуктов питания, в том числе готовой еды с предоставлением услуги доставки товаров. Для исследования была выбрана сеть магазинов здорового питания, в составе которых отсутствуют консерванты и усилители вкуса. Популяризация здорового образа жизни, физической активности и правильного питания задает положительную тенденцию для развития бизнеса в сфере доставки продуктов и готовых блюд. Рост количества компаний, производящих и реализующих аналогичные товары и услуги, а следовательно, и рост конкуренции, обязывают компании пристально следить за качеством своей продукции и своевременно реагировать на изменение ее качества, что еще раз подтверждает суждение о том, что тема нашей работы является актуальной и востребованной.

Целью проекта является создание системы контроля качества продукции на основании отзывов клиентов, с минимальными затратами человеческих ресурсов, финансов и времени на реализацию и развитие программного продукта.

Ценность идеи проекта заключается в том, что реализованная система является универсальным решением для решения задач подобного рода и может быть применена для аналогичных компаний с минимальными доработками функционала.

Основная задача, на решение которой направлен наш проект – это непрерывное отслеживание изменения качества продукции с целью своевременного реагирования и принятия соответствующих мер для устранения негативных последствий для бизнеса.

Объектом для исследования и сбора информации была выбрана российская сеть розничных магазинов продуктов здорового питания «ВкусВилл». Это динамично развивающаяся компания, основанная в 2009 году в Москве. На сегодняшний день функционирует более двух тысяч магазинов в 170 городах России. С 2024 года компания также запустила доставку товаров в ОАЭ и страны ближнего зарубежья - Казахстан, Беларусь, Киргизию, Армению и Таджикистан. Одним из аргументов в пользу выбора именно этой компании стало большое количество отзывов, число которых ежедневно пополняется, а также достаточно широкий ассортимент товаров, потому что наличие актуальных и разнообразных данных является одним из основных факторов для создания качественной ИИ-модели с высоким уровнем предсказаний.

Для реализации проекта были использованы следующие инструменты и технологии:

* Apache Airflow (построение и выполнение ETL-процессов);
* PostgreSQL (хранение данных в БД и построение отчетов);
* pgAdmin (клиент для управления БД);
* Windows Subsystem for Linux (WSL) (для запуска Linux-приложений в ОС Windows)
* Python (создание кода приложения);
* Docker (контейнеризация приложений);
* Selenium (библиотека для автоматизации взаимодействия с веб-браузером через программный код);
* Hugging Face (ИИ-инструмент для обучения модели);
* RuBERT-tiny2, Label Studio (разметка данных);
* razdel (Natasha), pymorphy2, NLTK (лингвистические модули для анализа текста).
* библиотеки Python для работы с датасетами, обучения модели и графического представления результатов (графиков и диаграмм);
* Draw.io (вспомогательный инструмент для визуализации предлагаемых решений в команде);
* VsCode, PyCharm (редакторы кода);
* Jupyter Notebook (анализ и обработка данных, обучение ИИ-модели, визуализация результатов);
* Облачный сервис cloud.mail.ru (хранение и обмен файлами в распределенной команде).

Состав команды: Андрей Комарницкий, Татьяна Лукьянова

# Глава 1 Автоматизированная система анализа отзывов

## 1.1 Основные компоненты, графическое представление

Рассмотрим более подробно структуру и компоненты, из которых состоит система для анализа отзывов пользователей, схематично она представлена на рисунке 1:



Рисунок 1 – Структура автоматизированной системы для анализа

изменения качества продукции

Верхнеуровнево система представлена пятью компонентами:

1. *Сайт компании с отзывами* – это источник для получения данных, на основании которых сначала будет обучаться модель и, в дальнейшем – выполнять предсказания.
2. *Паук (spider)* – инструмент веб-скраппинга, модуль для автоматизированного извлечения данных с веб-сайта и сохранения полученной информации в удобном для анализа формате.
3. *СУБД –* система для управления базой данных, данный модуль необходим для возможности сохранения полученных данных и их управлением.
4. *Файловое хранилище* – наличие данного модуля обусловлено необходимостью временного сохранения файлов, которые в дальнейшем будут обработаны и сохранены в базу данных.
5. *DAG (Airflow)* (Directed Acyclic Graph в Apache Airflow) – это направленный ациклический граф, описывающий процессы обработки данных (пайплайны). Представляет собой набор задач с отношениями и зависимостями друг от друга. DAG позволяет Airflow отслеживать зависимости между задачами и запускать пайплайны в правильном порядке.

Главным модулем, который производит запуск других модулей и обеспечивает их взаимодействие, является *DAG.* Основные этапы, из которых складывается работа системы:

1. *Выполнение сбора данных*. Паук по расписанию собирает свежие комментарии с сайта для заданных категорий продуктов, выполняет их предварительную обработку (очистка лишних символов, приведение типов данных) и сохраняет данные для каждой категории в отдельный файл в формате \*.csv в файловое хранилище.
2. *Подготовка датасета*. Скрипт, написанный на языке программирования Python, выполняет объединение файлов разных категорий продуктов в один файл в формате \*.csv.
3. *Загрузка датасета в БД*. Подготовленный на предыдущем этапе датасет загружается в промежуточную таблицу в базу данных. Состав колонок таблицы повторяет структуру файла \*.csv.
4. *Разнесение данных в итоговые таблицы БД.* Данные из промежуточной таблицы разносятся в итоговые таблицы, которые спроектированы в соответствии с требованиями к таблицам реляционных БД. При вставке данных производятся действия по приведению типов, заполнению недостающих записей в таблицы нормативно-справочной информации.
5. *Удаление файлов.* После загрузки данных в БД, файлы удаляются из хранилища, с целью экономии дискового пространства и избежания повторной загрузки данных.
6. *Предсказание результатов.* Данные, собранные и подготовленные на предыдущих этапах, передаются в ранее обученную модель для дообучения и получения предсказаний. После вычисления значения целевой переменной, результаты сохраняются в БД.
7. *Построение отчетов.* В системе предусмотрена возможность формирования отчета об изменении качества продукции, который строится с настроенной периодичностью по заданным параметрам (период, категории товаров) и направляется на электронную почту, адресатам, указанным в списке для рассылки.

На Рисунке 2 представлена диаграмма потоков данных (DFD) (в нотации Йордона-Де Марко):



Рисунок 2 – Диаграмма потоков данных (концептуальный уровень)

На диаграмме представлены только этапы целевой модели, этапы разметки данных, а также само обучение модели будет рассмотрено отдельно, в рамках следующей главы.

# Глава 2 Обучение модели

В рамках данной работы наибольший интерес для исследования представляет функционал, касающийся разметки данных и обучения модели, потому что именно эти разделы напрямую связаны с применением алгоритмов искусственного интеллекта. Рассмотрим каждый из этапов более подробно.

## 2.1 Описание датасета

Датасет содержит следующие данные:

* author – автор отзыва;
* comment – отзыв о покупке
* date – дата покупки и номер дисконтной карты
* url – веб–адрес продукта (товара)
* product – информация о продукте (название)
* stars – количество звезд (оценка товара от 1 до 5)
* price – стоимость товара на момент покупки
* currency – валюта
* weight – единица измерения товара
* category – категория товара

Пропуски и выбросы отсутствуют. Размер датасета составляет 4387 записей. Для тестирования модели использовался датасет размером 690 записей.

## 2.2 Разметка данных

Разметка данных для машинного обучения – это процесс структурирования и аннотации необработанных данных, приведение их к виду, подходящему для обучения моделей и алгоритмов. Другими словами, это процесс добавления меток (tag, label) в сырые данные, чтобы показать модели машинного обучения целевые атрибуты (ответы), которые она должна предсказывать. Метки представляют собой описательные элементы, которые указывают, чем является отдельный элемент данных, чтобы модель могла учиться на примере. Таким образом, размеченные данные выделяют признаки (характеристики) данных, чтобы помочь модели анализировать информацию и идентифицировать паттерны в исторических данных для выполнения точных предсказаний на новых, относительно близких входящих данных.

Разметка данных – это многоэтапный процесс, он включает такие шаги как определение задачи, которую решает модель, создание инструкций для разметки, разметку данных с помощью инструментов автоматизации или вручную, а также контроль качества.

Разметка данных является одним из важнейших этапов подготовки данных для машинного обучения с учителем, результат точности предсказания модели напрямую зависит от того, насколько качественно была выполнена разметка данных, на которых впоследствии обучалась модель.

Для обеспечения более высокого уровня предсказаний модели при создании данного проекта мы сравнили результаты трех выполненных вариантов разметки данных:

1. с помощью нейросети *RuBERT-tiny2*, без предварительной обработки целевого признака (текст комментария покупателя с сайта в исходном виде);
2. с использованием в качестве целевой переменной значения оценки товара (число от 1 до 5 - количество «звездочек», которое пользователь установил на сайте при оценке товара);
3. с помощью нейросети *RuBERT-tiny2*, с предварительной обработкой целевого признака (текст комментария покупателя с сайта в исходном виде). Предварительная обработка признака была выполнена путем токенизации, морфологического анализа, лемматизации и удалением стоп-слов.

Давайте рассмотрим подробнее каждое из действий предварительной обработки данных.

*Токенизация* - разделение письменного языка на предложения-компоненты, более мелкие единицы. Основная идея, лежащая в основе токенизации, - сегментация. Для текстовых данных это обычно означает разбиение предложений на слова, подслова или даже отдельные символы на основе заранее заданных правил или изученных шаблонов. Обычные методы включают в себя разделение текста на основе пробелов и знаков препинания. Однако часто требуются более продвинутые методы, особенно для работы с большими словарями или словами, не встречавшимися во время обучения. Такие методы, как Byte Pair Encoding (BPE) или WordPiece, разбивают слова на более мелкие подслова. Они часто используются в больших языковых моделях (LLM), таких как BERT и GPT-4, чтобы эффективно управлять размером словарного запаса и изящно обрабатывать неизвестные слова. Выбор стратегии токенизации может существенно повлиять на производительность модели и эффективность вычислений.

*Морфологический анализ –* это этап предварительной обработки данных, основная цель которого заключается в том, чтобы разбить слова на их составные части, такие как корни, приставки и суффиксы, и понять их роль и значения. Одной из задач морфологического анализа является снятие морфологической неоднозначности (омонимии).

*Лемматизация* – это процесс приведения слова к его базовой, начальной форме, или лемме. Лемма — это форма слова, которая представляет его в словаре и имеет общее значение с иными формами. Например, леммой для слова «бежать» будет «бег», для слова «говорит» — «говорить»

*Удаление стоп-слов* – это процесс фильтрации общеупотребительных слов, которые обычно несут мало смысловой нагрузки, из обрабатываемого текста. К стоп-словам относятся, например, предлоги, союзы, частицы и некоторые местоимения. Стоп-слова удаляют с целью сокращения объёма данных для анализа и фокусировании на значимых словах. Это повышает точность и эффективность обработки текста.

Для разметки данных первыми двумя способами (*RuBERT-tiny2*, без предварительной обработки целевого признака и разметка на основании целевой переменной оценки товара) были написаны две пользовательские функции, одна из которых размечает данные с использованием нейросети RuBRT-tiny, а вторая – на основе данных оценки по следующему принципу: от 1-3 - негативный отзыв, 4,5 - позитивный отзыв.

Для разметки данных третьим способом были использованы лингвистические модули:

1. *Модуль razdel (Natasha)*

Библиотека Natasha имеет множество функций включая токенизацию, морфологический анализ, лемматизацию, синтаксический анализ и извлечение именованных сущностей. Токенизация и сегментация текста на предложения и токены в Natasha осуществляются с помощью встроенной библиотеки Razdel.

1. *Морфологический анализатор pymorphy2*

PyMorphy2 также предоставляет функции для морфологического анализа текстов, PyMorphy2 полностью открыта и развивается сообществом. Библиотека использует словари OpenCorpora для анализа и может работать с русским. PyMorphy2 также предлагает API.

1. *Модуль NLTK*

NLTK – это свободно распространяемая библиотека Python, разработанная для работы с человеческим языком. Это комплексный набор инструментов, предназначенный для символьной и статистической обработки естественного языка. Она предоставляет легкий доступ к более чем 50 корпусам текстов и лексическим ресурсам, таким как WordNet, а также набор библиотек для классификации, токенизации, стемминга, метки частей речи, синтаксического анализа и семантического рассуждения.

Помимо этого, при создании стандартного русскоязычного словаря стоп-слов из него была исключена частица «не», т.к. в комментариях пользователей часто встречались грамматические ошибки при написании слова «невкусно» и различных его форм. Также дополнительно используется стоп-слова из внешнего локального словаря.

Перед выполнением токенизации во всех текстовых комментариях были удалены лишние символы (Emoji, переводы строк и прочие символы).

После выполнения разметки тремя способами был проведен визуальный анализ результатов при помощи гистограмм. Результат анализа показал, что при разметке данных по оценке товара количество негативных и позитивных отзывов имеют примерно одинаковые значения, однако этот результат отличается от тех, которые были получены при использовании нейросети *RuBERT-*tiny2 на модифицированных и неизмененных данных, где количество отрицательных отзывов превышает количество положительных*.*

Для того, чтобы определить, какая из целевых переменных наиболее точно отражает реальную оценку товара, было решено провести дополнительное исследование: сделать несколько срезов и случайным образом выбрать 5 наборов по 100 значений, затем разметить их вручную при помощи инструмента для разметки Label Studio и снова построить гистограммы для визуального анализа получившихся результатов.

После сопоставления результатов ручной разметки данных и трех целевых переменных, которые были получены разными способами разметки, описанными выше, мы выявили, что наилучший результат показала разметка данных на основе значения оценки товара (количество «звездочек»), в процентном соотношении количество совпадений этой переменной с результатами ручной разметки составило 92,2%. Таким образом, для обучения модели было принято решение взять данную целевую переменную. Но, следует отметить, что нейросеть RuBRT-tiny2 также показала высокую точность, 88,4% значений совпали с результатами ручной разметки данных.

На Рисунке 3 представлены гистограммы целевых переменных (sentiment – ручная разметка данных, target\_by\_RuBRT-tiny\_no\_processing – с использованием нейросети и без предварительной обработки признака, target\_by\_RuBRT-tiny\_processing – с использованием нейросети и предварительной обработкой признака, target\_by\_stars - с использованием в качестве целевой переменной значения оценки товара):

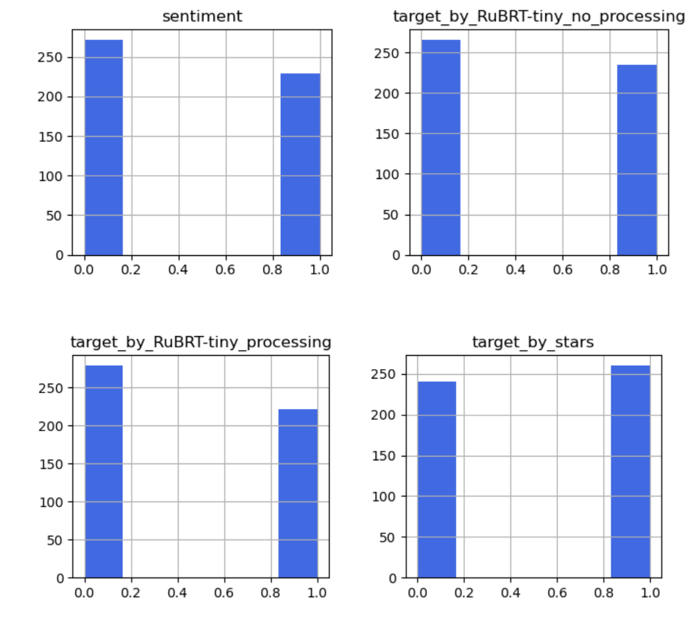


Рисунок 3 – Гистограммы целевых переменных

## 2.3 Обучение модели

На этапе обучения модели были выполнены следующие шаги:

1. Подготовка данных для обучения (проверка на пропуски, удаление образовавшихся в процессе лемматизации пропусков)
2. Извлечение признаков из текстовых данных с использованием TF-IDF *Term Frequency-Inverse Document Frequency (TF-IDF)* — это один из наиболее распространенных и мощных методов для извлечения признаков из текстовых данных. TF-IDF вычисляет важность каждого слова в документе относительно количества его употреблений в данном документе и во всей коллекции текстов. Этот метод позволяет выделить ключевые слова и понять, какие слова имеют больший вес для определенного документа в контексте всей коллекции. *TF (Частота термина)* обозначает, насколько часто определенное слово появляется в данном документе. Таким образом, TF измеряет важность слова в контексте отдельного документа. *IDF (Обратная частота документа)* измеряет, насколько уникально слово является по всей коллекции документов. Слова, которые появляются в большинстве документов, имеют низкое IDF, так как они не вносят большой информационной ценности. TF-IDF предоставляет несколько ключевых преимуществ:
3. Учет важности слов: TF-IDF учитывает как частоту слова в документе, так и его общую редкость по всей коллекции. Таким образом, он помогает выделять ключевые слова, которые часто встречаются в данном документе, но не слишком распространены в остальных.
4. Устранение шума: Слова, которые встречаются в большинстве документов (стоп-слова), имеют низкий IDF и, следовательно, низкий общий вес TF-IDF. Это позволяет устранить шум и фокусироваться на более важных словах.

Ограничения метода и ситуации, в которых TF-IDF может быть неэффективен:

1. Отсутствие семантической информации: TF-IDF не учитывает семантические связи между словами, что может привести к ограниченной способности понимания смысла текста.
2. Чувствительность к длине документа: Длинные документы могут иметь более высокие значения TF, даже если ключевые слова встречаются реже. В таких случаях, TF-IDF может недооценить важность конкретных слов.
3. Нормализация по длине документа. Когда мы вычисляем TF-IDF для слов в документах разной длины, длинные документы могут иметь более высокие значения TF, что может повлиять на точность результатов. Чтобы избежать этой проблемы, значения TF-IDF нормализуют по длине документа.
4. Преобразование в разреженную матрицу. На выходе мы получили достаточно большую матрицу (более 20 тыс. признаков).
5. Применение к полученным векторам TF-IDF линейных методов понижения размерности PCA и нелинейных t-SNE, UMAP.

PCA (Principal Component Analysis) – это анализ главных компонент, уменьшает размерность набора данных, максимизируя при этом дисперсию интерпретации каждого основного компонента.

1. Визуализация полученного результата с применением алгоритмов t-SNE (стохастическое вложение соседей с t-распределением) и UMAP (Uniform Manifold Approximation and Projection).

*t-SNE* – это алгоритм, используемый для упрощения и визуализации сложных данных. Он достигает этой цели путем сравнения сходства между точками данных в исходном пространстве высокой размерности и пространстве низкой размерности. Затем он создает распределение вероятностей для представления этих сходств, стремясь сделать их как можно более похожими. Алгоритм итеративно корректирует положение точек данных в низкоразмерном пространстве до тех пор, пока распределение не станет максимально близким.

*Uniform Manifold Approximation and Projection* *(UMAP)* является более мощным алгоритмом. Он также изучает нелинейные отображения для поддержания целостности кластера, и это происходит быстрее. Кроме того, по сравнению с t-SNE, UMAP, как правило, лучше справляется с поддержанием глобальной структуры данных. В этом контексте глобальная структура относится к «степени близости» между похожими типами вин, в то время как локальная структура относится к степени кластеризации одного и того же типа вина в уменьшенном по размерам пространстве.

1. Разделение датасета на данные для обучения и тестирования, размер тестовой выборки равен 35%.
2. Обучение нескольких моделей (со стандартными параметрами):

* Логистическая регрессия (LogisticRegression);
* Случайный лес (RandomForestClassifier);
* Повышение градиента (GradientBoostingClassifier);
* Экстремальный градиентный бустинг (XGBClassifier).

1. Анализ результатов. На данном этапе были получены показатели, свидетельствующие о том, алгоритм понижения размерности t-SNE с моделями RandomForestClassifier, GradientBoostingClassifier, XGBClassifier показал наилучшие результаты.
2. Подбор гиперпараметров. Все модели машинного обучения имеют набор аргументов, которые должны быть заданы специалистом. Единственный способ найти наиболее подходящие гиперпараметры для своего датасета – это метод проб и ошибок, что и является основной концепцией оптимизации гиперпараметров. Для подбора лучших гиперпараметров модели мы использовали библиотеку GridSearchCV sklearn.
3. Обучение модели с улучшенными гиперпараметрами.
4. Построение матрицы ошибок для анализа эффективности предсказаний модели. Она показывает соотношение между предсказанными и фактическими значениями, выявляет не только количество правильных прогнозов, но и типы ошибок, которые совершает модель. Для бинарной классификации матрица ошибок представляет собой таблицу из двух строк и двух столбцов. Строки соответствуют фактическим классам, столбцы – предсказанным. Элементы матрицы обозначают:

- True Positive (TP) — истинно положительные примеры (модель правильно предсказала позитивный класс);

- True Negative (TN) – истинно отрицательные примеры (модель правильно предсказала негативный класс);

- False Positive (FP) — ложноположительные примеры (модель ошибочно предсказала позитивный класс);

- False Negative (FN) — ложноотрицательные примеры (модель ошибочно предсказала негативный класс).

1. Получение кривых обучения. Графики кривых обучения представляют из себя зависимости доли ошибки от количества примеров тренировочной выборки.

# Глава 3 Практическая часть

В данной главе мы рассмотрим более подробно каждый из этапов, представленных на Рисунке 2 в первой главе в разделе 1.1, с указанием фрагментов кода и пояснением логики работы, а также кратко опишем порядок работы над проектом. Т.к. данный проект разрабатывался в команде из двух человек, то функции были разделены, каждый из участников реализовывал свою часть функционала, а затем все компоненты соединялись, проводилось общее тестирование исправление ошибок и доработка. Из основных шагов, которые были выполнены, можно выделить:

1. выбор темы проекта, анализ возможных направлений для исследования;
2. выбор инструментов;
3. распределение функций в команде;
4. планирование работ;
5. настройка рабочих мест (установка необходимого ПО, библиотек и др.);
6. проектирование;
7. разработка (написание скриптов для БД, DAG);
8. обучение модели, анализ результатов, поиск путей улучшения качества предсказаний модели (несколько итераций);
9. тестирование;
10. поиск идей для улучшения реализованной системы.

Отдельным пунктом можно выделить разрешение конфликтов совместимости версий библиотек на этапе сборки компонентов нашей системы воедино. Для их устранения мы использовали информацию из официальной документации компаний-авторов используемого программного обеспечения, а также ответы экспертов на аналогичные вопросы из открытых интернет-источников ([stackoverflow.com](file:///C:\Users\Tatyana%20Lukyanova\Downloads\stackoverflow.com), [habr.com](https://habr.com/) и др.).

* 1. **Сбор данных**

Для сбора данных реализован паук (script\_pars\_vkus\_vill.py), который собирает данные об отзывах пользователей с сайта <https://vkusvill.ru/>, для заданной категории товара, например, для получения отзывов о товарах из категории «Овощи, фрукты, ягоды, зелень», в запросе используется url <https://vkusvill.ru/goods/ovoshchi-frukty-yagody-zelen/>.

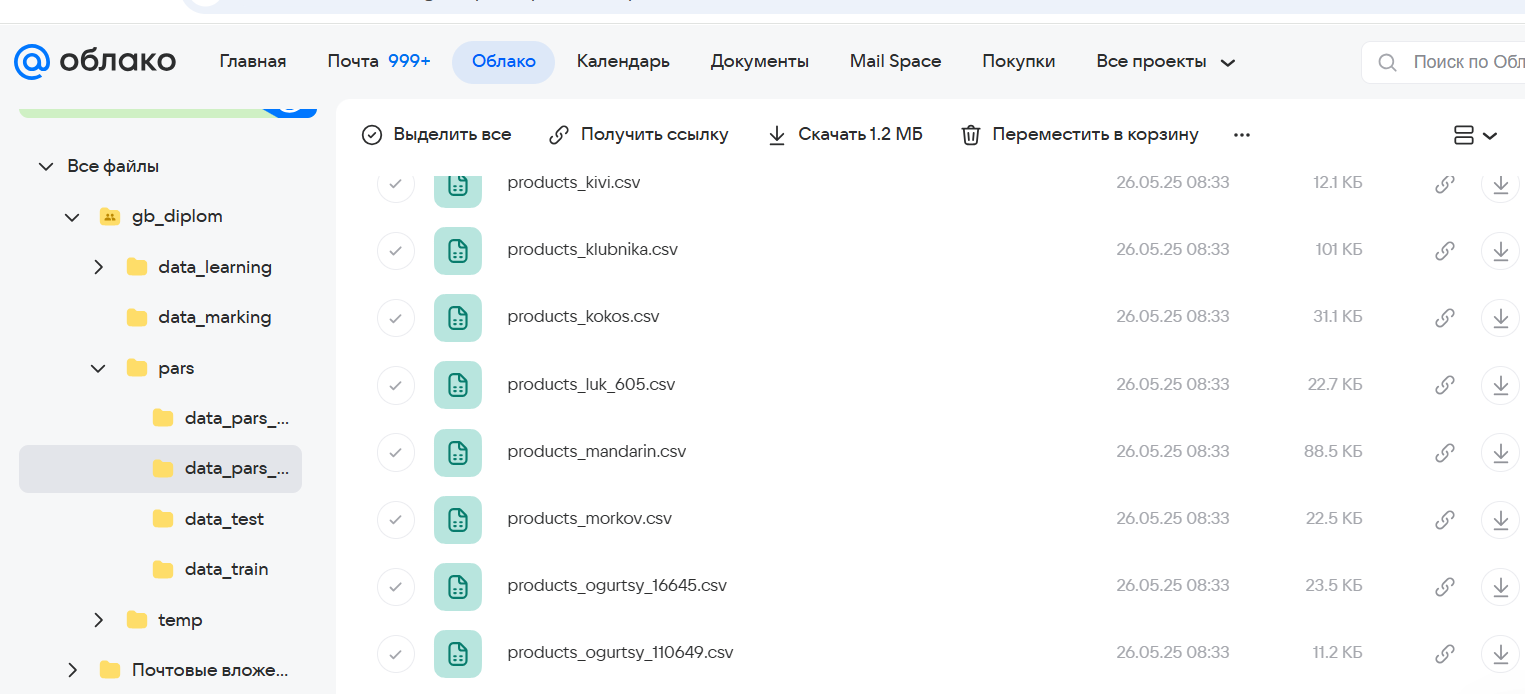
Паук переходит по указанной ссылке (открывается браузер), далее, проходит в цикле по всем продуктам категории, для каждого из них проматывает страницу вниз, нажимает кнопку «отзывы», считывает указанное количество отзывов, сохраняет считанные данные в файл в формате \*.csv, имя файла формируется по шаблону «'products\_' + <наименование продукта> + '\_' + <префикс> + '.csv'». Файлы для каждой категории сохраняются в каталог dat\_pars\_train, на Рисунке 4 представлен пример содержимого каталога с файлами, которые были сформированы при сборе пауком комментариев с сайта:

Рисунок 4 – каталог с файлами комментариев, собранных пауком с сайта

Перед сохранением данных в файл, они подвергаются предварительной обработке: очистке от лишних символов, форматированию.

Фрагмент кода, где происходит извлечение данных со страницы сайта, представлен на Рисунке 5:



Рисунок 5 – фрагмент извлечения данных с сайта

* 1. **Подготовка датасета**

Далее, при помощи библиотеки Pandas, все файлы, созданные на первом этапе, объединяются в один файл с именем "vkus\_vils\_products.csv" в формате \*.csv.

Фрагмент кода для объединения файлов представлен на Рисунке 6:



Рисунок 6 – фрагмент кода для объединения файлов

* 1. **Сохранение данных в БД**

На следующем этапе происходит чтение содержимого файла "vkus\_vils\_products.csv" и сохранение его в базу данных, в промежуточную таблицу "comments", состав колонок таблицы идентичен составу колонок загружаемого файла. При вставке записей в таблицу "comments" происходит преобразование даты из текстового формата, в котором дата считывается с сайта, в тип "date".

После того, как данные загрузились в таблицу "comments", они разносятся в итоговые таблицы: "categories", "products", "reviews".

После переноса данных из таблицы "comments", происходит ее очистка.

На Рисунке 7 представлена ER-диаграмма таблиц:

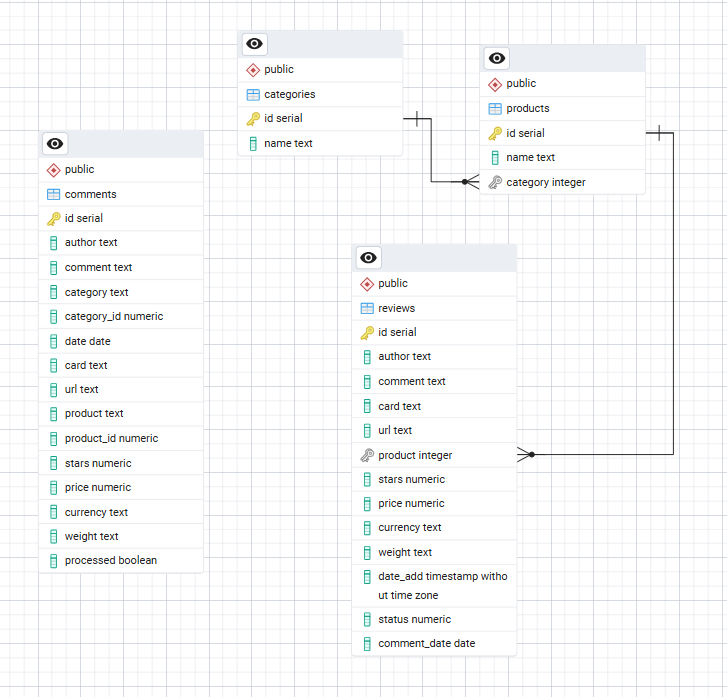


Рисунок 7 – ER-диаграмма таблиц

Далее представлен листинг кода с сохранением данных в БД (Рисунки 8, 9):

Рисунок 8 – Сохранение данных в промежуточную таблицу.

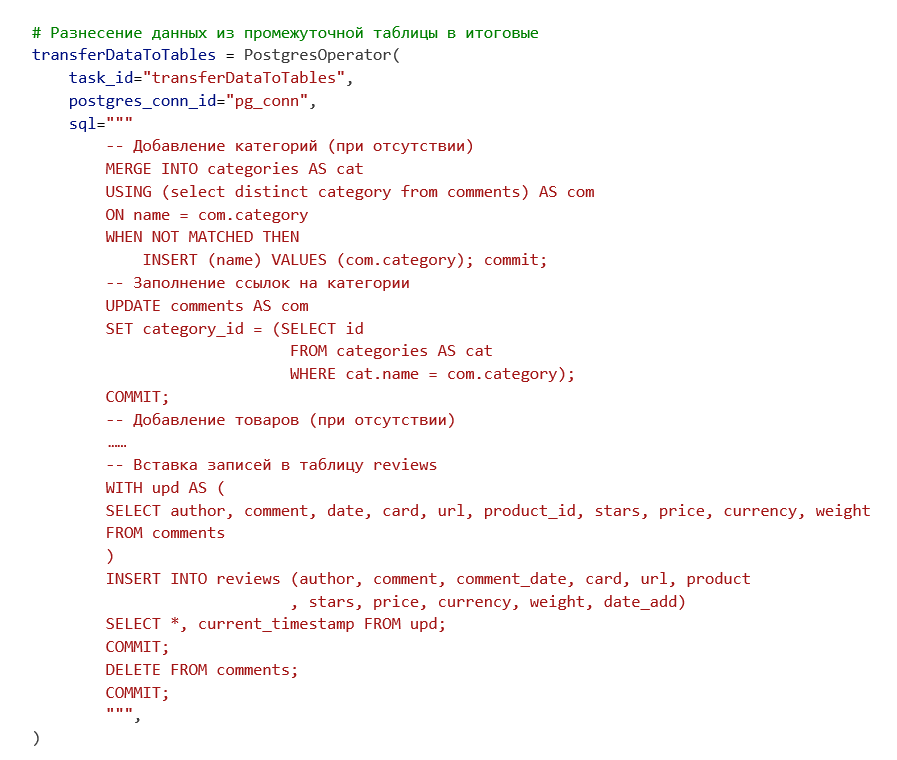
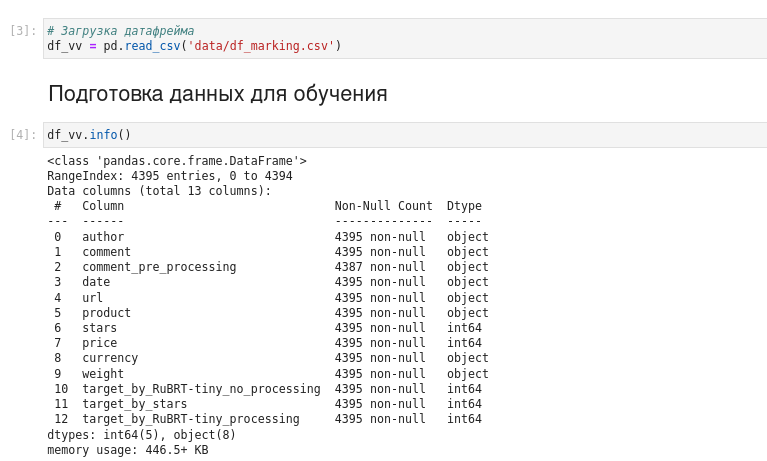
****

Рисунок 9 – Сохранение данных в итоговые таблицы

* 1. **Обучение модели**

Данный этап не отображен на диаграмме потоков данных, т.к. он был выполнен единожды, а в созданной в рамках данного проекта автоматизированной системе используется только его результат – сформированная модель машинного обучения, однако, этот этап был самым объемным по количеству затраченных часов на исследование и его реализацию. Рассмотрим его более подробно.

Загрузим размеченный датасет df\_marking.csv

Рисунок 10 – Загрузка данных

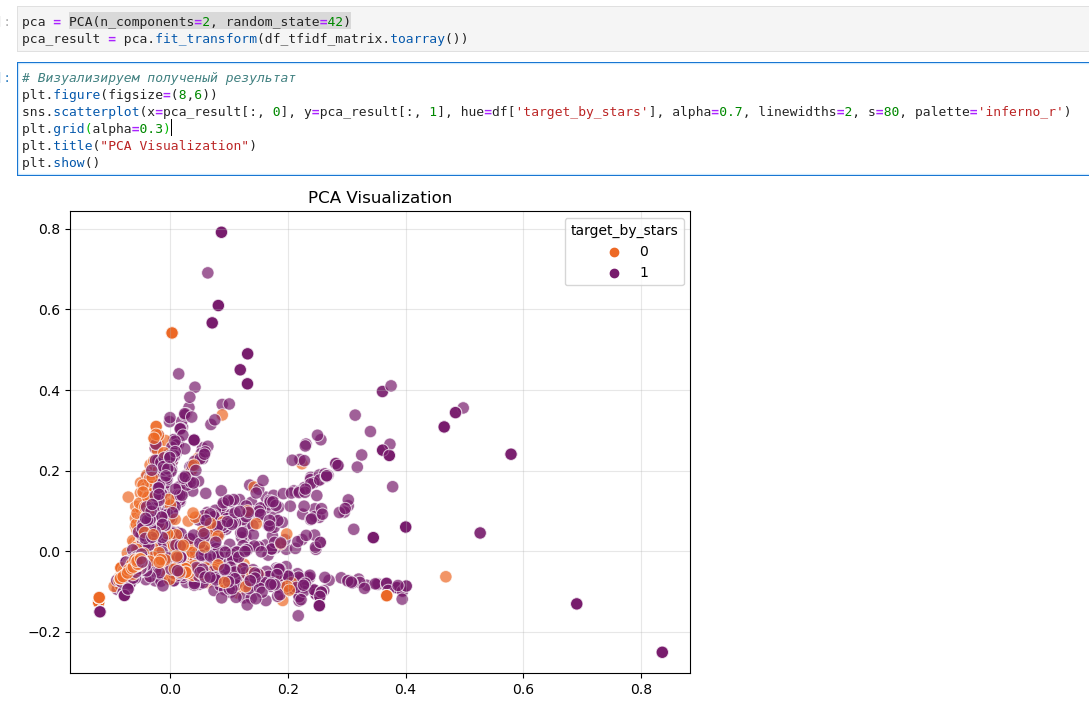
Для извлечения новых признаков, применим TfidfVectorizer к колонке “comment\_pre\_processing” с параметрами (ngram\_range=(1, 2), sublinear\_tf=True. Таким образом мы увеличим семантические отношения между словами, которые идут вместе, а также делая вес TF менее линейно растущим, что может быть полезным для учета насыщенности текста ключевыми словами.

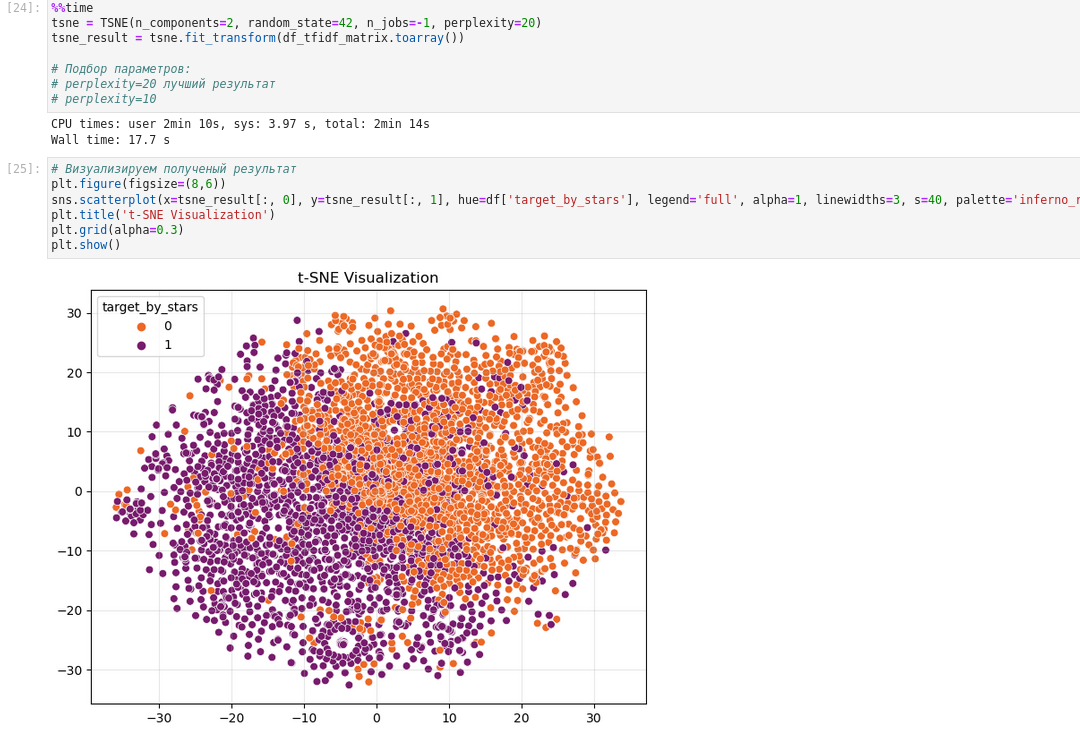
Рисунок 11 – Получение новых признаков (TfidfVectorizer)

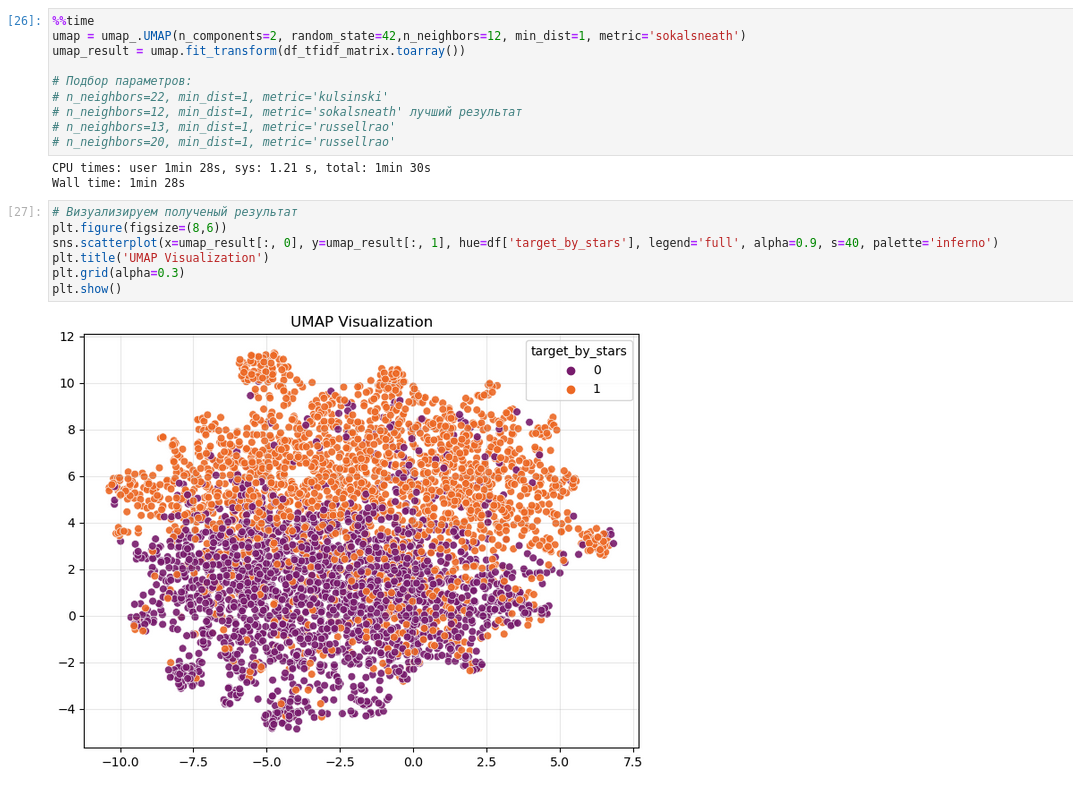
Когда мы вычисляем TF-IDF для слов в документах разной длины, длинные документы могут иметь более высокие значения TF, что может повлиять на точность результатов. Чтобы избежать этой проблемы, нормализуем значения TF-IDF по длине документа normalize с параметрами (norm='l2') .

При работе с большими объемами данных матрица TF-IDF может стать очень большой и занимать много памяти. Для оптимизации используем разреженные матрицы, которые хранят только ненулевые значения. На выходе мы получили достаточно большую матрицу с более 20 000 признаками.

Для дальнейшей работы и визуализации в векторном пространстве, понизим размерность матрицы до 2 компонент применив методы понижения размерности PCA, t-SNE, UMAP.

Рисунок 12 – Алгоритм PCA (n\_components=2)

Рисунок 13 – Алгоритм t-SNE (n\_components=2, perplexity=20)

Рисунок 14 – Алгоритм UMAP (n\_components=2, n\_neighbors=12, min\_dist=1, metric='sokalsneath')

Нелинейные алгоритмы понижения размерности t-SNE (t-Distributed Stochastic Neighbor Embedding) и UMAP (Uniform Manifold Approximation and Projection) наиболее точно разделили наши данные.

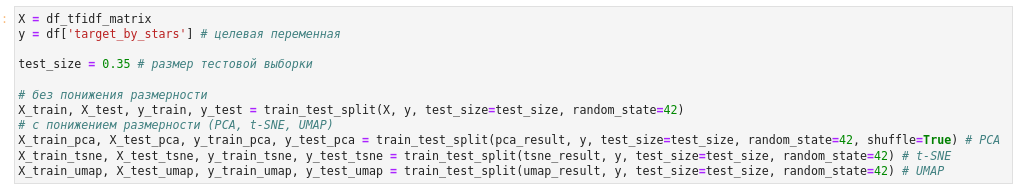
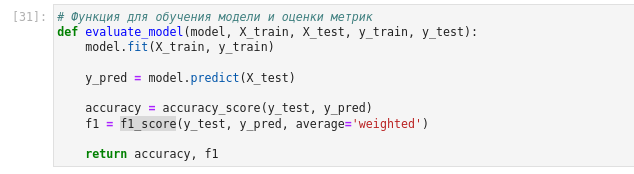
Далее разделим датасет на данные для обучения и тестирования. Для тестовой части зададим размер 35% от датасета.

Рисунок 15 – Разделение датасета на данные для обучения и тестирования

Для обучения моделей и оценки метрик напишем функцию evaluate\_model, которая принимает на вход модель и разделенные данные на обучающие и валидационные выборки. На выходе данная функция возвращает метрики accuracy и f1\_score.

Рисунок 16 – Функция evaluate\_model

В качестве обучающих алгоритмов возьмем:

* LogisticRegression,
* RandomForestClassifier,
* GradientBoostingClassifier,
* XGBClassifier

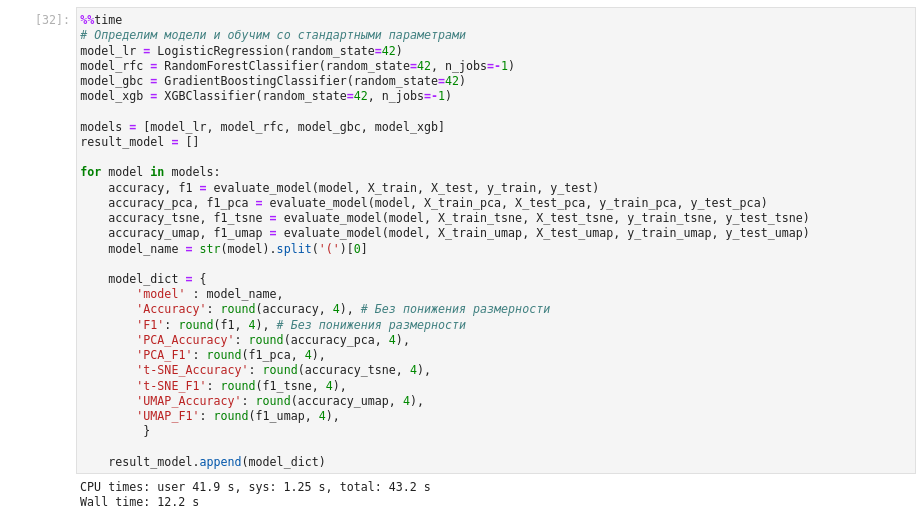
Определим модели и обучим со стандартными параметрами

Рисунок 17 – Обучение со стандартными параметрами

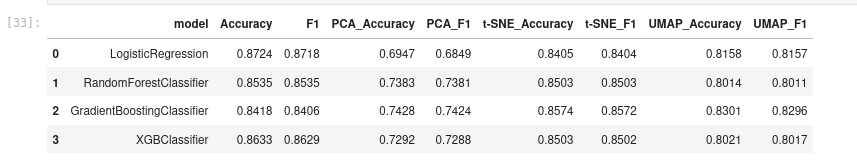
Посмотрим на результаты обучения моделей

Рисунок 18 – Вывод результатов обучения

Далее подберем гиперпараметры для достижения наилучших метрик с помощью GridSearchCV, а также протестируем выполнив кроссвалидацию Кfold.

Рисунок 19 – Подбор гиперпараметров

Посмотрим на результаты

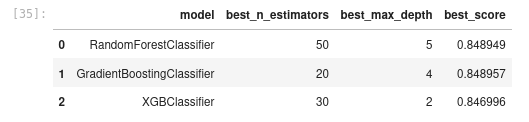


Рисунок 20 – Вывод результатов подбора гиперпараметров

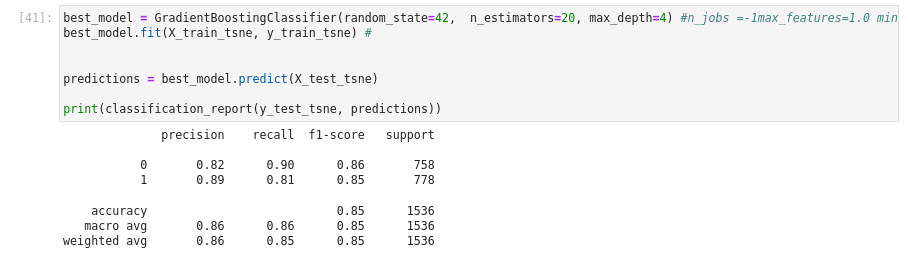
В качестве основного алгоритма выберем GradientBoostingClassifier и обучим его с полученными гиперпараметрами.

Рисунок 21 – Обучение модели и вывод результатов.

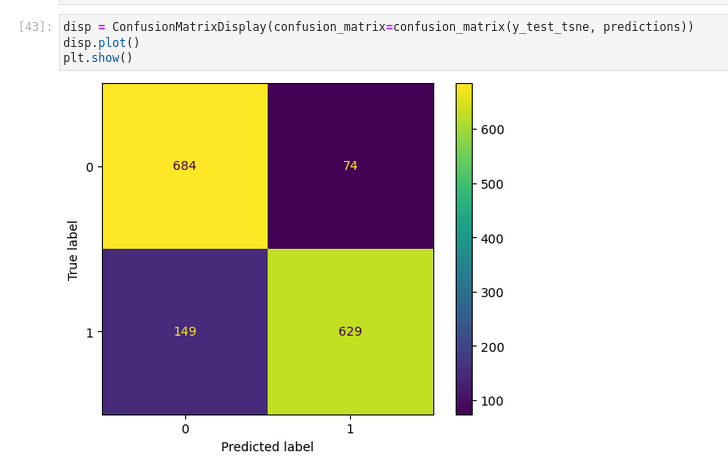
Выведем матрицу ошибок

Рисунок 22 – Матрица ошибок

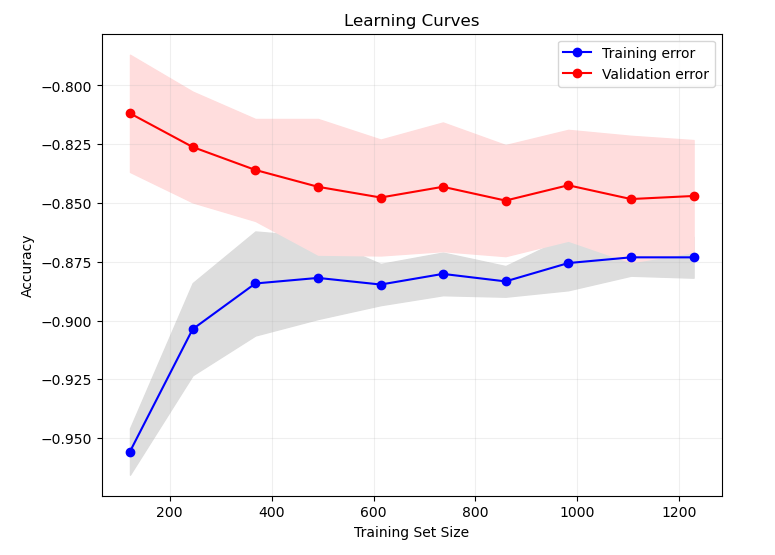
Построим кривые обучения

Рисунок 23 – Кривые обучения

Загрузим полученную модель и протестируем на новых (тестовых данных):

Рисунок 24 – Метрики предсказаний модели

На тестовых данных, метрики стали хуже.

* 1. **Предсказание результатов**

После того, как модель обучена и готова к использованию, она сохраняется в файл с расширением \*.pkl. На этапе предсказания результатов запускается bash-оператор, который выполняет скрипт, написанный на языке Python (predict\_results.py).

Основная логика, которая написана в этом скрипте представляет следующую последовательность действий:

1. подключение к базе данных;
2. создание датафрейма на основе данных из БД (берутся комментарии, которые были считаны пауком с сайта и ранее не обрабатывались, значение колонки “status” для таких записей будет иметь значение null);
3. извлечение признаков из текстовых данных с использованием TF-IDF (в сформированном датасете создается временная колонка, в которую записывается текст комментария, далее для нее выполняется метод Term Frequency-Inverse Document Frequency (TF-IDF), то есть вычисляется важность каждого слова в документе относительно количества его употреблений в данном документе и во всей коллекции текстов. Это позволяет выделить ключевые слова и понять, какие слова имеют больший вес;
4. нормализация матрицы TF-IDF по длине документа и преобразование ее в разреженную матрицу. Этот шаг необходим для ускорения вычислений, так как при использовании метода TF-IDF для представления текста, где большинство слов не встречаются, соответствующие элементы матрицы равны нулю. Преобразование в разреженную матрицу позволяет оптимизировать хранение и обработку таких данных.
5. загрузка модели машинного обучения из файла;
6. выполнение предсказаний модели для созданного датасета;
7. обновление значения колонки "status" в БД значениями целевой переменной из датафрейма, поиск соответствующей записи в БД происходит по ее идентификатору (значение id записи присутствует в датафрейме, который загружается из БД на первом шаге).
   1. **Формирование отчетов**

Формирование отчетов для системы анализа отзывов клиентов является важной составляющей, т.к. именно представление информации в удобном для восприятия виде (таблица или диаграмма), позволяет специалисту по качеству быстро обрабатывать большой объем информации и принимать соответствующие меры при необходимости. В рамках данного проекта, формирование отчетов не было автоматизировано, существует возможность построения отчета путем выбора данных из таблиц в базе по заданным параметрам (период, категория, товар). Но, учитывая, что при создании проекта мы использовали Apache Airflow, который позволяет гибко настраивать ETL-процессы и использовать много готовых библиотек с открытым кодом, автоматизация функций построения отчетов и их рассылки, может быть выполнена в короткие сроки и с минимальными затратами ресурсов на написание кода и выполнение настроек. В дальнейшем, при развитии системы, возможно реализовать автоматическое построение отчетов и их отправку на электронную почту адресатам, заданным в списке для рассылки.

* 1. **Описание DAG**

Как уже было сказано выше, при создании данного проекта мы использовали Apache Airflow. Это мощный инструмент для создания, выполнения, мониторинга и управления потоками операций по обработке данных, основными преимуществами которого являются гибкость, расширяемость, простота использования, наличие большого количества плагинов и возможность интегрирования с различными источниками данных, такими как MySQL, PostgreSQL, Apache Hive, Apache Spark, Apache Hadoop и многими другими. Среди недостатков Apache Airflow можно отметить сложность развертывания, ограничение на использование разных языков программирования (для написания операторов доступен только язык Python), но для реализации данного проекта эти факторы являются несущественными.

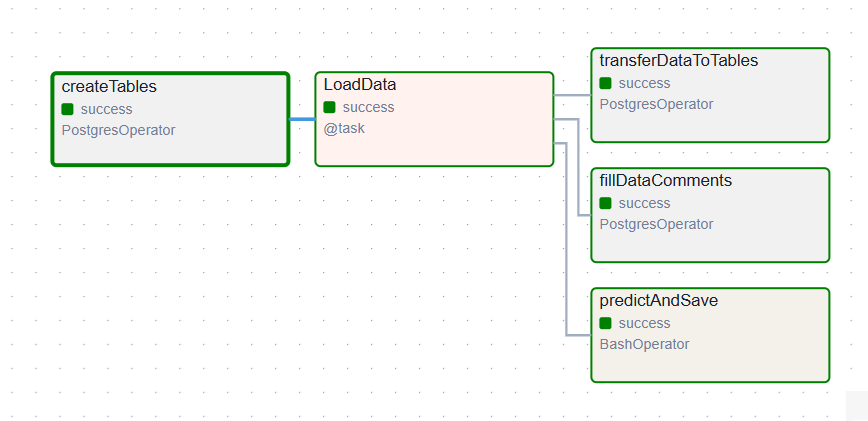
 Для настройки автоматизированного выполнения шагов, описанных в предыдущих разделах, мы реализовали DAG, граф представлен на Рисунке 25:

Рисунок 25 – Структура DAG

Рассмотрим подробнее каждый из компонентов.

1. *createTables* – это оператор для создания таблиц в БД, если они не были созданы ранее. Так как мы используем базу данных PostgreSQL, то и для работы с данными вызываем соответствующий оператор – PostgresOperator.
2. *LoadData –* это задача, которая выполняет чтение данных из файла в формате \*.csv, создает соединение с базой данных и копирует содержимое файла в промежуточную таблицу “comments”.
3. *fillDataComments –* это PostgresOperator, который осуществляет нормализацию данных, добавленных на предыдущем шаге (приведение даты к правильному формату из текстового вида, удаление лишних символов).
4. *transferDataToTables –* это PostgresOperator, предназначенный для разнесения данных из промежуточной таблицы “comments” в итоговые таблицы.
5. *predictAndSave –* это BashOperator (операто, который используется для запуска простых bash комманд)*.* Оператор запускает скрипт, написанный на языке Python, этот скрипт выполняет предсказание результатов с помощью ранее обученной модели и сохраняет результаты в базу данных.

Листинг кода DAG представлен в Приложении 2.

# Выводы и рекомендации

В процессе работы по обучению модели были применены алгоритмы логистической регрессии (LogisticRegression), случайного леса (RandomForestClassifier), повышения градиента (GradientBoostingClassifier) и экстремального градиентного бустинга (XGBClassifier). Алгоритмы обучались на тренировочной выборке размером 3070 наблюдений и тестировалась на валидационной выборке размером 1317 наблюдений. Выборка имела сбалансированный вид (0-класс 2207, 1-класс 2180). Также была подготовлена тестовая выборка из 690 наблюдений (0-класс 281, 1-класс 409). Так как выборки почти сбалансированы, использовалась метрика accuracy\_score.

К алгоритмам были подобраны оптимальные наборы гиперпараметров с помощью GridSearchCV, а также тестирование на кросс-валидации Kfold. В качестве основного алгоритма был выбран GradientBoostingClassifier.

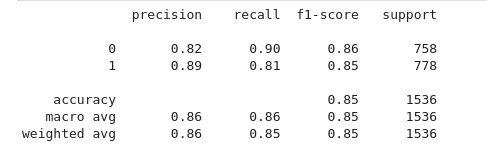
По результатам обучения были получены следующие метрики (Рисунки 26, 27):

Рисунок 26 – Метрики качества предсказания модели

Матрица ошибок и кривые обучения представлены в разделе про обучение модели на рисунках 22, 23.

Метрики качества предсказания модели имеют достаточно хороший уровень, но ошибка алгоритма на валидационной выборке близка к выходу на плато. Скорее всего, добавляя данные мы не сможем повысить качество классификации. Кривая может быть зашумлена (иметь большой разброс точек).

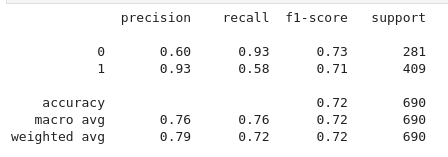
Метрики качества модели на новых данных (тестовых) представлены на Рисунке 27:

Рисунок 27 – Метрики качества предсказания на тестовых данных

Метрики качества модели на новых данных заметно просели по сравнению с валидационной выборкой, что свидетельствует об уменьшении обобщающей способности модели. Как предполагалось выше, скорее всего это связано с большой зашумленностью в данных.

**Рекомендации по дальнейшему улучшению системы**

1. Перенос модели машинного обучения в облачную систему «web-сервер Flask». Flask имеет множество расширений, которые могут добавить необходимые функции. Например, Flask-SQLAlchemy для работы с базами данных, Flask-Login для аутентификации пользователей и Flask-Migrate для управления миграциями базы данных. Данное решение позволит забыть о конфликтах в работе импортируемых библиотек, а также отслеживать качество предсказаний и своевременно до обучать модель.
2. Автоматизация этапа формирования отчетов, реализация возможности почтовой рассылки отчетов адресатам, заданным в настройках.

Для повышения качества классификации возможно рассмотреть следующие варианты:

1. Расширение словаря «стоп-слов» для удаления слов из отзывов, не имеющих смысловой нагрузки, для оценки модели, например: названия продукта, страна производитель продукта, имена, слова о способе доставки, названия поставщиков и пр..

2. Обработка отзывов для «слипшихся слов» и «абракадабры».

3. Ограничения длины отзыва до 10-15 слов. Во время анализа отзывов была подсчитана минимальная длина отзыва — 1 слово, максимальная — 76 слов, среднее значение — 7. Большая длина отзыва негативно сказывается на качестве классификатора, так как добавляет много «шума» в данные для модели во время обучения.

4. Тестирование других алгоритмов машинного обучения таких как:

* *Support Vector Machines*. Основная идея метода заключается в переводе исходных векторов в пространство более высокой размерности и поиск разделяющей гиперплоскости с наибольшим зазором в этом пространстве. Две параллельных гиперплоскости строятся по обеим сторонам гиперплоскости, разделяющей классы. Разделяющей гиперплоскостью будет гиперплоскость, обеспечивающая наибольшее расстояние до двух параллельных гиперплоскостей. Алгоритм основан на допущении, что чем больше разница или расстояние между этими параллельными гиперплоскостями, тем меньше будет средняя ошибка классификатора.
* CatBoost — открытая программная библиотека, разработанная компанией «Яндекс» и реализующая уникальный патентованный алгоритм построения моделей машинного обучения, использующий одну из оригинальных схем градиентного бустинга.

# Заключение

В рамках данного проекта была реализована система для анализа изменения качества продукции с использованием модели лингвистического анализа и инструментов для автоматизации ETL-процессов.

В процессе работы над проектом мы получили ценный практический опыт использования современных технологий, изучили ряд специализированных библиотек, предназначенных для решения задач машинного обучения, овладели навыками работы с датасетами и инструментами визуализации для анализа и обработки данных, попробовали реализовать программное решение для реальной бизнес-задачи. Учитывая, что этот проект является результатом работы распределенной команды из двух человек, можно отметить, что мы также получили опыт командной работы и ряд преимуществ такого формата работы над проектом. Например, при выполнении исследовательской части работы, каждый из нас изучал теоретические основы, искал варианты реализации, а затем, мы вместе обсуждали каждый из них, приводили аргументы в пользу того или иного решения, и, по итогу обсуждений, делали выбор.

Подводя итоги, можем сделать вывод, что цели, которые мы ставили перед собой при реализации данного проекта, успешно достигнуты.

# Список используемой литературы

1. Книга: Пол Бэрри. Изучаем программирование на Python 2-е издание., Москва, Эксмо, 2017.
2. Книга: Йоав Голдберг. Нейросетевые методы в обработке естественного языка, Москва, ДМК Пресс, 2019.
3. Статья: Куцев Роман @kucev, <https://habr.com/ru/articles/678524/>, 2022
4. Статья: artem @badcasedaily1 [https://habr.com/ru/companies/otus/articles /808435/](https://habr.com/ru/companies/otus/articles%20/808435/), 2024.
5. Статья: [@nvpushkarskiy2](https://habr.com/ru/users/nvpushkarskiy2/), <https://habr.com/ru/companies/Voximplant/articles/446738/>, 2019.
6. Статья: artem @badcasedaily1 <https://habr.com/ru/companies/otus/articles> /787116/ , 2024.
7. Статья: k0rsakov, <https://habr.com/ru/articles/722688/>, 2023
8. Статья: Vinayak Sharma, <https://habr.com/ru/companies/otus/articles/596071/>, 2021.
9. Статья: [Astrononer](https://www.astronomer.io/ebooks/dags-definitive-guide/), <https://habr.com/ru/articles/682384/>, 2022
10. Статья: Анна Маврина, <https://blog.skillfactory.ru/vektorizatsiya-teksta-v-nlp-ot-slov-k-chislam/>, 2025.
11. Статья: Павел Богомолов, <https://blog.skillfactory.ru/kak-rabotaet-tehnologiya-raspoznavaniya-rechi/>, 2025.
12. Статья: Анастасия Саврова, <https://blog.skillfactory.ru/modeli-bert-dlya-mashinnogo-obucheniya-gaid-dlya-nachinayuschih/>, 2024.
13. Статья: Мария Жарова, <https://blog.skillfactory.ru/predvaritelnaya-obrabotka-dannyh-v-mashinnom-obuchenii/>, 2025.

# Приложения

Приложение 1 - Листинг кода для этапа предсказания результатов

# Подключение к БД

def connect(params\_dic):

    """ Connect to the PostgreSQL database server """

    conn = None

    try:

        print('Connecting to the PostgreSQL database...')

        conn = psycopg2.connect(\*\*params\_dic)

    except (Exception, psycopg2.DatabaseError) as error:

        print(error)

        sys.exit(1)

    print("Connection successful")

    return conn

# Создание датафрейма на основе данных из БД

def postgresql\_to\_dataframe(conn, select\_query, column\_names):

    """ Tranform a SELECT query into a pandas dataframe """

    cursor = conn.cursor()

    try:

        cursor.execute(select\_query)

    except (Exception, psycopg2.DatabaseError) as error:

        print("Error: %s" % error)

        cursor.close()

        return 1

    # Naturally we get a list of tupples

    tupples = cursor.fetchall()

    cursor.close()

    # We just need to turn it into a pandas dataframe

    df = pd.DataFrame(tupples, columns=column\_names)

    return df

#  Подключение к БД

conn = connect(param\_dic)

# Список колонок таблицы

column\_names = ["id", "author", "comment", "card", "url", "product", "stars", "price", "currency", "weight", "date\_add", "status"]

# Выполнение запроса

df = postgresql\_to\_dataframe(conn, "select \* from reviews where status is null", column\_names)

# Предварительная обработка

df['target'] = df['stars'].apply(get\_label\_target) # Создаем колонку с целевой переменной "target\_by\_stars"

df['comment\_pre\_processing'] = df['comment'].apply(lambda x: clean\_text(x)) # очищенными от лишних данных

df['comment\_pre\_processing'] = df['comment\_pre\_processing'].apply(lambda x: lemmatization(x)) # применим функцию lemmatization

df['comment\_pre\_processing'] = df['comment\_pre\_processing'].apply(lambda x: ' '.join(x)) # приведем знаячения к строкам

df.dropna(subset=['comment\_pre\_processing'], inplace=True) # удалим пропуски в процессе лемматизации (при их наличии)

tfidf\_vectorizer = TfidfVectorizer(ngram\_range=(1, 2), sublinear\_tf=True, max\_features=None) # Применение TF-IDF к признаку 'comment\_pre\_processing'

df\_tfidf\_matrix = tfidf\_vectorizer.fit\_transform(df['comment\_pre\_processing'])

# Нормализация матрицы TF-IDF по длине документа

df\_tfidf\_matrix\_normalized = normalize(df\_tfidf\_matrix, norm='l2', axis=1)

# Преобразование в разреженную матрицу

sparse\_matrix = csr\_matrix(df\_tfidf\_matrix\_normalized)

# Понижение размерности TSNE

tsne = TSNE(n\_components=2, random\_state=42, n\_jobs=-1, perplexity=20)

tsne\_result = tsne.fit\_transform(df\_tfidf\_matrix.toarray())

# загружаем модель

clf = load('/home/lukyanova/airflow/dags/models/best\_model.pkl')

# предсказание результатов

pred = clf.predict(tsne\_result)

# перед записью в БД удаляем 'comment\_pre\_processing'

df.drop(['comment\_pre\_processing', 'target'], axis=1, inplace=True)

df['status'] = pred # добавляем колонку с предсказаниями 'status'

print("conn.cursor()")

# Обновление значения колонки "status" в БД значениями соответствующей колонки из датафрейма

cur = conn.cursor()

print("for index, row in df.iterrows()")

for index, row in df.iterrows():

    sql = 'update reviews set status = %s where id = %s'

    cur.execute(sql, (row['status'], row['id']))

conn.commit()

Приложение 2 – Листинг кода DAG

import pandas as pd

import datetime

import pendulum

from airflow.models.param import Param

import os

import requests

from airflow.decorators import dag, task

from airflow.providers.postgres.hooks.postgres import PostgresHook

from airflow.providers.postgres.operators.postgres import PostgresOperator

import psycopg2

import re

from datetime import datetime as dt

import logging

# from airflow.operators import BashOperator

from airflow.operators.bash import BashOperator

import sys

import pickle

import interpretable\_tsne

sys.path.append('/home/lukyanova/airflow/dags')

# 1. Создание dag

@dag(

    dag\_id="saveCommentsToDB",

    schedule\_interval="0 0 \* \* \*",

    start\_date=pendulum.datetime(2023, 1, 1, tz="UTC"),

    catchup=False,

    dagrun\_timeout=datetime.timedelta(minutes=60),

    )

def processData():

    # Создание в БД (при отсутствии) таблиц "Категории", "Товары", "Комментарии"

    createTables = PostgresOperator(

        task\_id="createTables",

        postgres\_conn\_id="pg\_conn",

        sql="""

            CREATE TABLE IF NOT EXISTS comments (

                "id" SERIAL PRIMARY KEY,

                "author"    TEXT,

                "comment"   TEXT,

                "category"  TEXT,

                "category\_id" NUMERIC,

                "date"      DATE,

                "card"      TEXT,

                "url"   TEXT,

                "product"   TEXT,

                "product\_id" NUMERIC,

                "stars" NUMERIC,

                "price" NUMERIC,

                "currency" TEXT,

                "weight"    TEXT,

                "processed" BOOLEAN

            );

          """,

    )

    @task

    def LoadData():

        # 2. Чтение данных из файлов

        try:

            # Чтение данных из файла

            df\_comments = pd.read\_csv("/home/lukyanova/airflow/dags/data/vkus\_vils\_products.csv", sep=',')

            #df\_comments = pd.read\_csv("/home/lukyanova/airflow/dags/data/test.csv", sep=',')

            df\_comments['card'] = df\_comments['date']

            df\_comments['processed'] = "0"

            # Создание соединения

            postgres\_hook = PostgresHook(postgres\_conn\_id="pg\_conn")

            conn = postgres\_hook.get\_conn()

            cur = conn.cursor()

            # print(f'df\_count = {df}')

            # Вставка данных в БД

            with conn.cursor() as c:

                psycopg2.extras.execute\_values(

                    cur=c,

                    sql="""

                        INSERT INTO comments

                        (author, comment, category, card, url, product, stars, price, currency, weight, processed)

                        VALUES %s;

                        """,

                    argslist=df\_comments.to\_dict(orient="records"),

                    template="""

                        (

                            %(author)s, %(comment)s, %(category)s, %(card)s, %(url)s,

                            %(product)s, %(stars)s, %(price)s,

                            %(currency)s, %(weight)s, %(processed)s

                        )

                        """

                )

            conn.commit()

            return 0

        except Exception as e:

            print(f'!!!! exception {e}')

            return 1

    # Нормализация данных

    fillDataComments = PostgresOperator(

        task\_id="fillDataComments",

        postgres\_conn\_id="pg\_conn",

        sql="""

            UPDATE comments SET weight = replace(weight, '/', '') WHERE date is null; commit;

            UPDATE comments SET date = TO\_DATE(trim(replace(replace(replace(replace(replace(replace(replace(replace(replace(replace(replace(replace(replace(SUBSTRING(card, 1, length(card) - STRPOS(card, 'Карта'))

                                                , 'янв', '01') , 'фев', '02')

                                                , 'мар', '03'), 'апр', '04')

                                                , 'мая', '05'), 'июн', '06')

                                                , 'июл', '07'), 'авг', '08')

                                                , 'сен', '09'), 'окт', '10')

                                                , 'ноя', '11'), 'окт', '12'), ' ', '-')), 'dd-mm-yyyy')

            WHERE date is null; commit;

            UPDATE comments SET card = SUBSTRING(card, STRPOS(card, 'Карта') + 7)  WHERE date is not null; commit;

            """,

    )

    # Разнесение данных из промежуточной таблицы в итоговые

    transferDataToTables = PostgresOperator(

        task\_id="transferDataToTables",

        postgres\_conn\_id="pg\_conn",

        sql="""

            -- Добавление категорий (при отсутствии)

            MERGE INTO categories AS cat

            USING (select distinct category from comments) AS com

            ON name = com.category

            WHEN NOT MATCHED THEN

                INSERT (name) VALUES (com.category); commit;

            -- Заполнение ссылок на категории

            UPDATE comments AS com

            SET category\_id = (SELECT id

                                FROM categories AS cat

                                WHERE cat.name = com.category);

            COMMIT;

            -- Добавление товаров (при отсутствии)

            MERGE INTO products AS prod

            USING (select distinct product, category\_id from comments) AS com

            ON name = com.product

            AND category = com.category\_id

            WHEN NOT MATCHED THEN

                INSERT (name, category) VALUES (com.product, com.category\_id);

            COMMIT;

            -- Заполнение ссылок на продукты

            UPDATE comments AS com

            SET product\_id = (SELECT id

                                FROM products AS prod

                                WHERE prod.name = com.product

                                AND prod.category = com.category\_id);

            COMMIT;

            -- Вставка записей в таблицу reviews

            WITH upd AS (

            SELECT author, comment, date, card, url, product\_id, stars, price, currency, weight

            FROM comments

            )

            INSERT INTO reviews (author, comment, comment\_date, card, url, product

                                , stars, price, currency, weight, date\_add)

            SELECT \*, current\_timestamp FROM upd;

            COMMIT;

            DELETE FROM comments;

            COMMIT;

            """,

    )

    # Запуск модели на добавленных данных

    predictAndSave = BashOperator(

    task\_id='predictAndSave',

    bash\_command='python /home/lukyanova/airflow/dags/predict\_results.py')

    print(f'Success predict and save')

    [createTables] >> LoadData() >> [fillDataComments, transferDataToTables, predictAndSave]

# 4. Запуск dag

processData()