

Минобрнауки России  
Федеральное государственное бюджетное образовательное учреждение  
высшего образования  
«Волгоградский государственный технический университет»

Факультет Электроники и вычислительной техники

Кафедра Электронно-вычислительные машины и системы

**ПОЯСНИТЕЛЬНАЯ ЗАПИСКА**  
**к курсовой работе (проекту)**

по дисциплине Системы обработки больших данных

на тему: Исследование данных по отзывам на фильмы и ТВ шоу с  
использованием фреймворка Apache Spark

Студент Маттиев Рустам Алимжонович  
(фамилия, имя, отчество)

Группа САПР-1.1

Руководитель работы (проекта) \_\_\_\_\_ П.Д. Кравченя  
(подпись и дата подписания) (инициалы и фамилия)

Члены комиссии:

_____ (подпись и дата подписания)	_____ (инициалы и фамилия)
_____ (подпись и дата подписания)	_____ (инициалы и фамилия)
_____ (подпись и дата подписания)	_____ (инициалы и фамилия)

Нормоконтроллер \_\_\_\_\_ П.Д. Кравченя  
(подпись и дата подписания) (инициалы и фамилия)

Волгоград 2026

Факультет    Электроники и вычислительной техники

Направление (специальность)    Информатика и вычислительная техника

Кафедра    Электронно-вычислительные машины и системы

Дисциплина    Системы обработки больших данных

Утверждаю  
Зав. кафедрой \_\_\_\_\_ А.Е. Андреев

«        »                                          20        г.

Студент Маттиев Рустам Алимжонович  
(фамилия, имя, отчество)

# 1. Тема: Исследование данных по отзывам на фильмы и ТВ шоу с использованием фреймворка Apache Spark

3. Содержание расчётно-пояснительной записки: РАЗВЕДОЧНЫЙ АНАЛИЗ ДАННЫХ С ИСПОЛЬЗОВАНИЕМ PYSPARK;  
МАШИННОЕ ОБУЧЕНИЕ НА БОЛЬШИХ ДАННЫХ.

Руководитель работы (проекта) \_\_\_\_\_ П.Д. Кравченя  
(подпись и дата подписания) (инициалы и фамилия)

Задание принял к исполнению \_\_\_\_\_ Р.А. Маттиев  
(подпись и дата подписания) (инициалы и фамилия)

# СОДЕРЖАНИЕ

ВВЕДЕНИЕ . . . . .	4
1 РАЗВЕДОЧНЫЙ АНАЛИЗ ДАННЫХ С ИСПОЛЬЗОВАНИЕМ PYSPARK . . . . .	5
1.1 Постановка задачи разведочного анализа . . . . .	5
1.2 Описание исходного датасета . . . . .	6
1.3 Определение пропущенных значений и преобразование данных	10
1.4 Анализ распределений, выбросов и категориальных признаков .	14
1.5 Выводы . . . . .	19
2 МАШИННОЕ ОБУЧЕНИЕ НА БОЛЬШИХ ДАННЫХ . . . . .	20
2.1 Задача регрессии . . . . .	20
2.1.1 Постановка задачи регрессии . . . . .	20
2.1.2 Решение задачи регрессии . . . . .	22
2.1.3 Анализ полученных результатов регрессии . . . . .	23
2.2 Задача классификации с использованием RandomForestClassifier	24
2.2.1 Постановка задачи классификации . . . . .	24
2.2.2 Решение задачи классификации . . . . .	25
2.2.3 Анализ полученных результатов классификации . . . . .	28
2.3 Выводы . . . . .	29
ЗАКЛЮЧЕНИЕ . . . . .	30
СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ . . . . .	31
ПРИЛОЖЕНИЕ А Изменение признаков . . . . .	32
ПРИЛОЖЕНИЕ Б Построение графика зависимости значения функции ошибки от номера итерации . . . . .	34
ПРИЛОЖЕНИЕ В Построение графиков ROC и PR кривых . . . . .	35

## ВВЕДЕНИЕ

Актуальность представленной курсовой связана с растущей необходимостью в создании высокоэффективных методов машинного обучения, предназначенных для анализа крупных массивов данных [1–3] и получения значимой информации из контента, генерируемого пользователями [4]. Особое значение при этом имеют комплексные решения, которые интегрируют этапы подготовки данных и последующего построения прогнозных моделей в рамках распределенных вычислительных сред.

Таким образом, целью курсовой работы является изучение и практическая реализация полного процесса машинного обучения на примере объемных данных об отзывах на фильмы и ТВ шоу с применением платформы Apache Spark [5; 6].

Для достижения поставленной цели выдвинуты следующие задачи:

1. Загрузка и первичное исследование структуры данных из распределенной файловой системы HDFS;
2. Выполнение базовых преобразований и очистки данных для подготовки к машинному обучению;
3. Применение алгоритмов машинного обучения на больших данных;
4. Анализ и оценка качества построенных прогностических моделей;
5. Визуализация результатов и подготовка выводов по исследованию.

В первом разделе рассмотрена более подробно постановка задачи и проведен обзор современных методов машинного обучения на больших данных. Во втором разделе описана методика предварительной обработки данных и выполнена верификация качества очистки. В третьем разделе представлены результаты применения алгоритмов машинного обучения, а также анализ эффективности построенных моделей. В заключении работы сформулированы общие выводы по результатам проведенного исследования.

# 1 РАЗВЕДОЧНЫЙ АНАЛИЗ ДАННЫХ С ИСПОЛЬЗОВАНИЕМ PYSPARK

## 1.1 Постановка задачи разведочного анализа

В рамках главы на основе наборов данных о пользовательских рецензиях на фильмы и ТВ шоу требуется провести EDA с использованием Apache Spark, чтобы оценить структуру и качество данных, проанализировать распределения ключевых признаков и их взаимосвязи, исследовать текстовые отзывы и подготовить визуализации для формирования выводов.

Исследование строится на принципах распределенных вычислений, что дает возможность эффективно обрабатывать миллионы записей [5; 7] и проводить анализ в условиях ограничений по вычислительным ресурсам и времени. Применение PySpark обеспечивает необходимую масштабируемость, а взаимодействие с распределенной файловой системой HDFS — удобную работу с крупными объемами данных.

Основные задачи разведочного анализа заключаются в следующем:

- загрузка данных из распределённой файловой системы HDFS и формирование единого датафрейма;
- исследование структуры, схемы и качества данных;
- выявление пропусков, дубликатов и некорректных значений;
- преобразование типов данных и создание производных признаков;
- предварительная оценка распределений количественных признаков и анализа категориальных данных;
- подготовка очищенного и структурированного набора данных для последующего применения алгоритмов машинного обучения.

Результатом данного этапа является построение целостного представления о данных и формирование корректной основы для дальнейших шагов анализа и моделирования. Это критически важная стадия подготовки данных, которая напрямую влияет на точность, надежность и интерпретируемость результатов будущих моделей.

## 1.2 Описание исходного датасета

В работе используется датасет «Amazon Review Data», доступный на платформе jmscauley [7]. Набор данных представляет собой файл json строк, который был конвертирован в csv файл. Объем данных составляет более 8,5 миллионов строк. Загруженные данные изначально имеют строковые типы, разнообразные форматы идентификаторов фильмов и ТВ шоу и включают большое количество текстовых полей.

Датасет включает следующие ключевые признаки (таблица 1):

Таблица 1 – Описание признаков датасета

Признак	Описание
asin	Идентификатор товара
image	Наличие изображения
overall	Общая оценка отзыва
reviewText	Текст отзыва
reviewTime	Дата отзыва
reviewerID	Идентификатор рецензента
reviewerName	Имя рецензента
style_Color	Стиль цвета текста
style_Format	Стиль формата текста
style_Shape	Стиль формы
style_Size	Стиль размера
summary	Краткое изложение
unixReviewTime	Дата отзыва в Unix формате
verified	Пройдена ли верификация
vote	Количество голосов

Чтение и демонстрация исходных данных на рисунке 1 производилось с помощью следующего кода:

```
path = "hdfs://hadoop-namenode:9820/data/Movies_and_TV.csv"
df = (spark.read.format("csv")
      .option("header", "true")
```

```
.load(path)

)

df.limit(10).toPandas()
```

	asin	image	overall	reviewText	reviewTime	reviewerID	reviewerName	style_Color:	style_Format:	style_Shape:	style_Size:	summary	unixReviewTime	verified	vote
0	0001527665	None	5.0	really happy they got evangelised ... spoiler a...	03 11, 2013	A3478QRKQDOPQ2	jacki	None	VHS Tape	None	None	great	1362960000	True	None
1	0001527665	None	5.0	Having lived in West New Guinea (Papua) during...	02 18, 2013	A2VHSG6TZHU1OB	Ken P	None	Amazon Video	None	None	Realistic and Accurate	1361145600	True	3
2	0001527665	None	5.0	Excellent look into contextualizing the Gospel...	01 17, 2013	A23EJWOW1TLENE	Reina Berumen	None	Amazon Video	None	None	Peace Child	1358380800	False	None
3	0001527665	None	5.0	"More than anything, I've been challenged to f...	01 10, 2013	A1KM9FNEJ8Q171	N Coyle	None	Amazon Video	None	None	Culturally relevant ways to share the love of ...	1357776000	True	None
4	0001527665	None	4.0	This is a great movie for a missionary going i...	12 26, 2012	A38LY2SSHVHRYB	Jodie Vesely	None	Amazon Video	None	None	Good Movie! Great for cross-cultural missionar...	1356480000	True	None
5	0001527665	None	5.0	This movie was in ENGLISH...it was a great su...	11 16, 2012	AHTYUW2H1276L	lilsistah21	None	Amazon Video	None	None	Great....	1353024000	True	3
6	0001527665	None	5.0	This is a fascinating true story, well acted b...	07 15, 2012	A3M3HCZLXW0YLF	Carla	None	Amazon Video	None	None	A remarkable true story, told in English (cont...	1342310400	True	4
7	0001527665	None	1.0	This DVD appears to be in German. It is not in...	09 3, 2010	A1OMHX76O2NC6V	Richard Ellis	None	Amazon Video	None	None	Peace Child DVD	1283472000	True	None
8	0001527665	None	1.0	This movie is not in English although the titl...	05 7, 2010	A3O8OZ41IK6O1M	Adam	None	Amazon Video	None	None	Not in English	1273190400	True	None

Рисунок 1 – Данные из датасета

Исследование структуры исходных данных было проведено в несколько этапов. На первом этапе с использованием команды `df.printSchema()` был выполнен анализ типов данных каждого поля, что позволило не только определить их формальную принадлежность (строковый, числовой, булевый тип), но и выявить потенциальную неоднородность в форматах хранения информации. Полученная схема данных стала основой для последующих решений по их преобразованию и очистке. Далее, в ходе визуального анализа выборки строк с помощью методов `df.show()` и `df.head()`, было принято решение об исключении ряда столбцов, несущих минимальную смысловую нагрузку для целей данного исследования. В частности, были удалены столбцы, содержащие стилевые метки (styles), поскольку они не имели прямого отношения к целевым задачам анализа текстовых рецензий и атрибутов контента. Финальный набор

признаков, отобранных для дальнейшего анализа и моделирования, представлен на схеме на рисунке 2.

```
[10]: df.printSchema()

root
 |-- Id: string (nullable = true)
 |-- Title: string (nullable = true)
 |-- authors: string (nullable = true)
 |-- publisher: string (nullable = true)
 |-- publishedDate: string (nullable = true)
 |-- categories: string (nullable = true)
 |-- review/score: string (nullable = true)
 |-- review/text: string (nullable = true)
 |-- review/summary: string (nullable = true)
 |-- review/helpfulness: string (nullable = true)
 |-- review/time: string (nullable = true)
 |-- User_id: string (nullable = true)
 |-- profileName: string (nullable = true)
```

Рисунок 2 – Структура таблицы после загрузки данных

Более детальное изучение содержимого выполнялось уже на этапе разведочного анализа при помощи выборочного просмотра записей (`df.show()`, `df.limit().toPandas()`), анализа уникальных значений и регулярных выражений. Эти методы позволили установить, что в каждом из столбцов может встречаться значение, которое явно не должно находиться в этом столбце, что может свидетельствовать о смещении строк в датасете. Поэтому далее будет уделено особое внимание очистке строк.

Параметры Spark-сессии были настроены с учётом объёма данных [1; 2]: увеличены объёмы памяти драйвера и исполнителей. Это обеспечивает стабильную работу при чтении и трансформации больших датафреймов. На рисунке 3 показана конфигурация `SparkSession`.

В конфиге, указанном ниже, последние строчки указывают на то, что используется `hadoop`:

```
def create_spark_configuration() -> SparkConf:
    user_name = os.getenv("USER")
    conf = SparkConf()
    conf.setAppName("Lab_1_DC")
    conf.setMaster("local[*]")
    conf.set("spark.executor.memory", "12g")
    conf.set("spark.executor.cores", "6")
    conf.set("spark.executor.instances", "1")
    conf.set("spark.driver.memory", "4g")
    conf.set("spark.driver.cores", "1")
    conf.set("spark.sql.catalog.spark_catalog.type", "hadoop")
    conf.set(
        "spark.sql.catalog.spark_catalog.warehouse",
        f"hdfs:///b"
    )
    return conf
```

```
[5]: spark = SparkSession.builder.config(conf=conf).getOrCreate()
     spark
```

[5]: **SparkSession - in-memory**

**SparkContext**

[Spark UI](#)

<b>Version</b>	v3.5.0
<b>Master</b>	local[*]
<b>AppName</b>	Lab_1_DC

Рисунок 3 – Демонстрация работы сессии Spark

Также для работы с датасетом, он был предварительно загружен в hdfs. Обзор директории представлен на рисунке 4.

Show  entries Search:

<input type="checkbox"/>	Permission	Owner	Group	Size	Last Modified	Replication	Block Size	Name	
<input type="checkbox"/>	<a href="#">-rw-r--r--</a>	<a href="#">jovyan</a>	<a href="#">supergroup</a>	3.54 GB	Nov 22 11:55	<a href="#">3</a>	128 MB	<a href="#">Movies_and_TV.csv</a>	
<input type="checkbox"/>	<a href="#">drwxr-xr-x</a>	<a href="#">jovyan</a>	<a href="#">supergroup</a>	0 B	Dec 14 12:04	<a href="#">0</a>	0 B	<a href="#">mattiev_database</a>	

Showing 1 to 2 of 2 entries Previous **1** Next

Рисунок 4 – Директория с данными для работы

### 1.3 Определение пропущенных значений и преобразование данных

Для систематической оценки полноты данных была реализована функция `analyze_missing_values()`, выполняющая постатейный подсчет NULL-значений и расчет их доли в каждом столбце набора данных. Данный этап позволяет выявить столбцы с критичным уровнем пропусков для выбора корректной стратегии их обработки:

```
def analyze_missing_values(data: DataFrame) -> None:
    print("=== АНАЛИЗ ПРОПУЩЕННЫХ ЗНАЧЕНИЙ ===")
    total_count = data.count()
    for column in data.columns:
        null_count = data.filter(col(column).isNull()).count()
        null_percentage = (null_count / total_count) * 100
        print(f"{column}: {null_count} пропусков "
              f"({null_percentage:.2f}%)")
```

Основная обработка выполнена в функции `handle_null_values()`. Примененные стратегии включают:

- Заполнение пропусков в строковых полях: Все столбцы строкового типа (`string`, `varchar`) с пропусками были заполнены значением "Unknown". Это обеспечивает консистентность категориальных и текстовых метаданных:

```
cleaned_data = data
for column in string_columns:
    null_count = (
        cleaned_data
        .filter(col(column).isNull())
        .count()
    )
    if null_count > 0:
        cleaned_data = cleaned_data.fillna({column: "Unknown"})
    print(
```

```

        f"Заменено NULL в {column}: "
        f"{null_count} значений"
    )

```

- Восстановление временных меток: Пропуски в поле `reviewTime` были заполнены на основе преобразования Unix-времени из поля `unixReviewTime` с помощью функции `from_unixtime()`. Это устраняет несоответствия в форматах дат:

```

if null_reviewtime_count > 0:
    cleaned_data = cleaned_data.withColumn(
        "reviewTime",
        when(
            (col("reviewTime").isNull()
             & col("unixReviewTime").isNotNull()),
            from_unixtime(col("unixReviewTime"), "yyyy-MM-dd")
        ).otherwise(col("reviewTime"))
    )

```

- Валидация и фильтрация по шаблонам (регулярным выражениям): Функция `clean_data_by_patterns()` обеспечивает глубокую очистку, удаляя строки, не соответствующие бизнес-правилам:

- `asin`: Должен состоять ровно из 10 алфавитно-цифровых символов.

```

cleaned_data = cleaned_data.filter(
    col("asin").rlike(patterns["asin"]) &
    (length(col("asin")) == 10)
)

```

- `reviewerID`: Не может быть NULL и должен содержать только буквы и цифры (до 50 символов).

```

cleaned_data = cleaned_data.filter(
    col("reviewerID").isNotNull() &
    col("reviewerID").rlike(patterns["reviewerID"])
)

```

- reviewerName: Если не NULL, должен соответствовать шаблону, допускающему буквы, цифры, пробелы и основные пунктуационные знаки.

```
cleaned_data = cleaned_data.filter(  
    col("reviewerName").isNull() |  
    col("reviewerName")  
        .rlike(patterns["reviewerName"]))  
)
```

- overall: Оценка должна быть в валидном диапазоне от 1.0 до 5.0.

```
cleaned_data = cleaned_data.filter(  
    col("overall").isNotNull() &  
    col("overall").between(1.0, 5.0)  
)
```

- unixReviewTime: Должен быть неотрицательным числом, представляющим дату не ранее 2000 года.

```
cleaned_data = cleaned_data.filter(  
    col("unixReviewTime").isNotNull() &  
    col("unixReviewTime").rlike(patterns["unixReviewTime"]) &  
    (col("unixReviewTime") >= 946684800)  
)
```

- reviewText: Если текст отзыва присутствует, его длина должна быть не менее 10 символов для содержательной аналитики.

```
cleaned_data = cleaned_data.filter(  
    col("reviewText").isNull() |  
    (length(col("reviewText")) >= 10)  
)
```

Для обеспечения уникальности записей были выполнены следующие шаги:

- Выявление дубликатов: Функция `analyze_duplicates()` подсчитывает как полные дубликаты строк, так и смысловые дубликаты отзывов (комбинация reviewerID, asin и unixReviewTime):

```

full_duplicates = (data
                    .groupBy(data.columns)
                    .count()
                    .filter(col("count") > 1))
print(f"Полные дубликаты: {full_duplicates.count()}")
review_duplicates = (data
                     .groupBy("reviewerID", "asin", "unixReviewTime")
                     .count()
                     .filter(col("count") > 1))
print(f"Дубликаты отзывов (reviewerID + asin "
      f"+ unixReviewTime): "
      f"{review_duplicates.count()}")

```

– Дедупликация: Функция `remove_duplicates()` последовательно:

1. Удаляет абсолютно идентичные строки `dropDuplicates()`:

```
cleaned_data = data.dropDuplicates()
```

2. Для оставшихся смысловых дубликатов отзывов оставляет только самую раннюю запись (на основе сортировки по `unixReviewTime`), используя оконную функцию `row_number()`. Это позволяет сохранить историческую целостность данных:

```

window_spec = Window.partitionBy(
    "reviewerID", "asin", "unixReviewTime"
).orderBy("unixReviewTime")

cleaned_data = cleaned_data.withColumn(
    "row_num", row_number().over(window_spec)
)

cleaned_data = cleaned_data.filter(
    col("row_num") == 1
).drop("row_num")

```

```
final_count = cleaned_data.count()
removed_count = initial_count - final_count
```

Результаты после выполнения преобразований продемонстрированы на рисунках 5 и 6.

```
root
|-- asin: string (nullable = true)
|-- reviewerID: string (nullable = true)
|-- reviewerName: string (nullable = true)
|-- reviewText: string (nullable = true)
|-- overall: float (nullable = true)
|-- summary: string (nullable = true)
|-- unixReviewTime: integer (nullable = true)
|-- reviewTime: string (nullable = true)
```

Рисунок 5 – Структура таблицы после обработки данных

	asin	reviewerID	reviewerName	reviewText	overall	summary	unixReviewTime	reviewTime
0	B005HS4CUIY	A0011668339FAW9XGHO	Mindy Graves	misleading would not play on our DVD player ...	1.0	misleading would not play on our DVD player re...	1490140800	2017-03-22
1	B001J710YS	A0018632VUVKRGSYBEAT	cierra	This product was amazing! It came in record ti...	5.0	Loved This	1353801600	2012-11-25
2	6301753534	A00222906VX8GH7X6J6B	Miguel A. Martinez	good price	5.0	Five Stars	1427846400	2015-01-04
3	B01COCKFOM	A0022400EY2L9WUJ22QRQ	Amazon Customer	I loved it great movie	5.0	Gotta see it	1474588800	2016-09-23
4	B00AEJM61I	A00849873KP522T77IL8	karen stephens	great movie to watch with all the lights of on...	5.0	Five Stars	1443139200	2015-09-25
5	B00EXDW1WA	A01008638Q2KKEWA1SAJ	Karen T.	Hilarious!!!!	5.0	Five Stars	1468540800	2016-07-15
6	0788812467	A0145698H8AAVMRXLBD8	cdy7626	We had this on VHS. Now needed it on DVD. Su...	5.0	Sure its been out a long time (new technology ...	1418515200	2014-12-14
7	B0000E32X3	A014922008WD7KG3O5XR	Mike McGrew	All Seasons of Frasier worth watching	5.0	Good solid entertainment	1463788800	2016-05-21
8	B0001NBNJ8	A014922008WD7KG3O5XR	Mike McGrew	I think Frasier like fine wine just got better...	5.0	Just gets better	1464566400	2016-05-30
9	B001GCUO5M	A0156038BA2FABM6OVXN	Rob Stidham	One of my all-time favorite movie's. Looks gre...	5.0	RUN	1456185600	2016-02-23

Рисунок 6 – Фрагмент данных после обработки

## 1.4 Анализ распределений, выбросов и категориальных признаков

Для количественного признака `overall` были построены гистограмма и круговая диаграмма с использованием Spark и последующей визуализацией в библиотеках Matplotlib.

Полученные результаты показывают:

- оценки пользователей смещены в сторону высоких значений (4–5) (рис. 7). Гистограмма построена с помощью следующего кода, где переменная `ratings` — список числовых значений оценок (от 1.0 до 5.0) из распределения, а `counts` — список количеств (частот) для каждой соответствующей оценки из распределения:

```

ratings = [float(row['overall']) for row in ratings_distribution]
counts = [row['count'] for row in ratings_distribution]

plt.figure(figsize=(10, 6))
plt.bar(ratings, counts, color='skyblue', edgecolor='black')
plt.title('Распределение оценок продуктов', fontsize=16)
plt.xlabel('Оценка', fontsize=12)
plt.ylabel('Количество отзывов', fontsize=12)
plt.xticks(ratings)
plt.grid(True, alpha=0.3)
plt.tight_layout()
plt.show()

```

- оценки пользователей в процентном соотношении на 81.1% состоят из положительных, что в будущем может отразиться на обучении (рис. 8). Круговая диаграмма была построена с помощью следующего кода, где переменная `positive_reviews` — количество положительных отзывов, `negative_reviews` — количество негативных отзывов, `neutral_reviews` — количество нейтральных отзывов:

```

positive_reviews = df_final.filter(col("overall") >= 4.0).count()
negative_reviews = df_final.filter(col("overall") <= 2.0).count()
neutral_reviews = df_final.filter((col("overall") > 2.0) & (col(

sentiments = ['Положительные', 'Нейтральные', 'Отрицательные']
sentiment_counts = [positive_reviews, neutral_reviews, negative_
colors = ['green', 'orange', 'red']

plt.figure(figsize=(8, 6))
plt.pie(sentiment_counts, labels=sentiments, colors=colors, autop
plt.title('Распределение отзывов по тональности', fontsize=16)
plt.axis('equal')
plt.tight_layout()
plt.show()

```

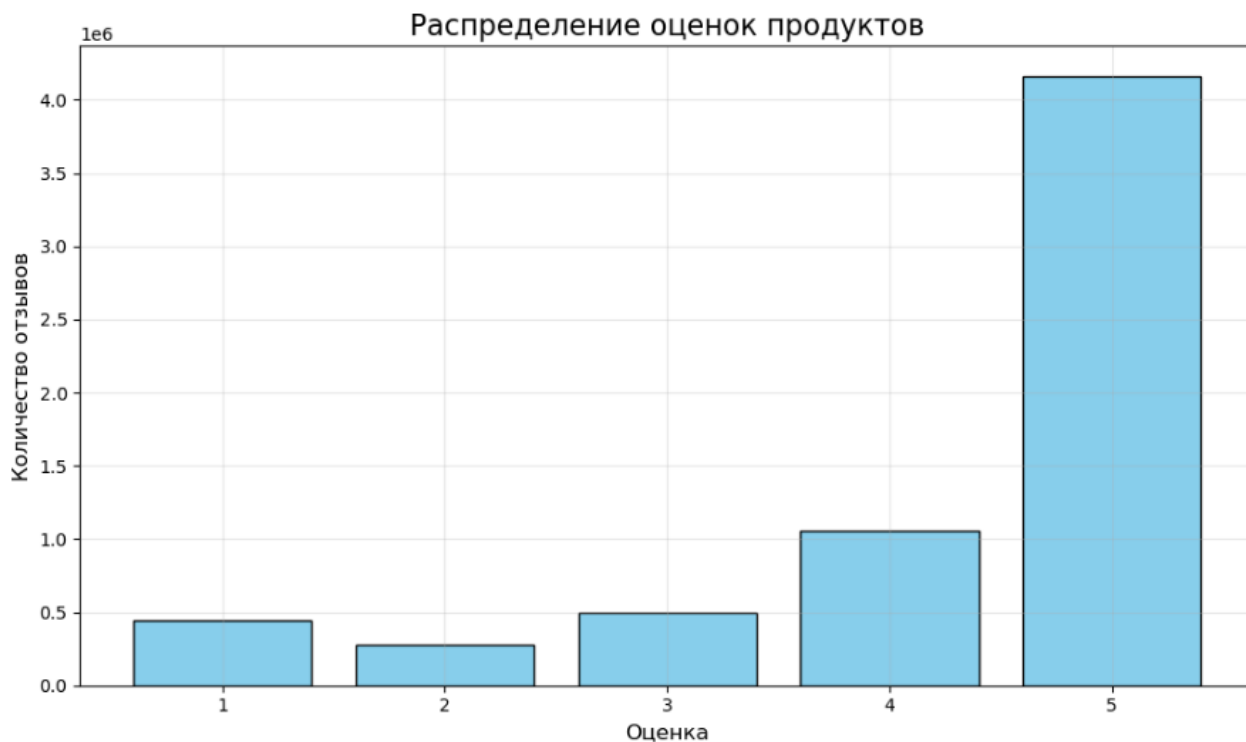


Рисунок 7 – Гистограмма распределения для overall



Рисунок 8 – Круговая диаграмма распределения для overall

На рисунке 9 продемонстрировано количество NULL и UNKNOWN значений у reviewerName, которые были получены с помощью следующего кода:

```
reviewername_null_count = (df_final.filter(col("reviewerName"))
```

```

        .isNull())
        .count())
reviewername_unknown_count = (df_final
                               .filter(col("reviewerName")== "Unknown")
                               .count())

print(f"NULL значений: {reviewername_null_count}")
print(f"Значений 'Unknown': {reviewername_unknown_count}")
print(f"Процент 'Unknown': "
      f"{reviewername_unknown_count/total_rows*100:.2f}%\n")

```

2. ПРОВЕРКА NULL И 'UNKNOWN' ЗНАЧЕНИЙ:  
 NULL значений: 0  
 Значений 'Unknown': 593  
 Процент 'Unknown': 0.01%

Рисунок 9 – Пример NULL и UNKNOWN значений у reviewerName

Для текстового признака `summary` была проанализирована частота встречаемости слов качественной оценки (рис. 10). Гистограмма была построена с помощью следующего кода, где `keywords_list` — список ключевых слов, обнаруженных в `summary`, `counts_list` — список частот (сколько раз каждое слово встречается):

```

keywords_list, counts_list, percentages_list = zip(*keyword_analysis)

plt.figure(figsize=(12, 6))
plt.bar(keywords_list, counts_list, color=['green' if kw in ['good', 'great', 'excellent'] else 'red'])
plt.title('Частота ключевых слов в summary', fontsize=16)
plt.xlabel('Ключевое слово', fontsize=12)
plt.ylabel('Количество', fontsize=12)
plt.xticks(rotation=45)
plt.tight_layout()
plt.show()

```

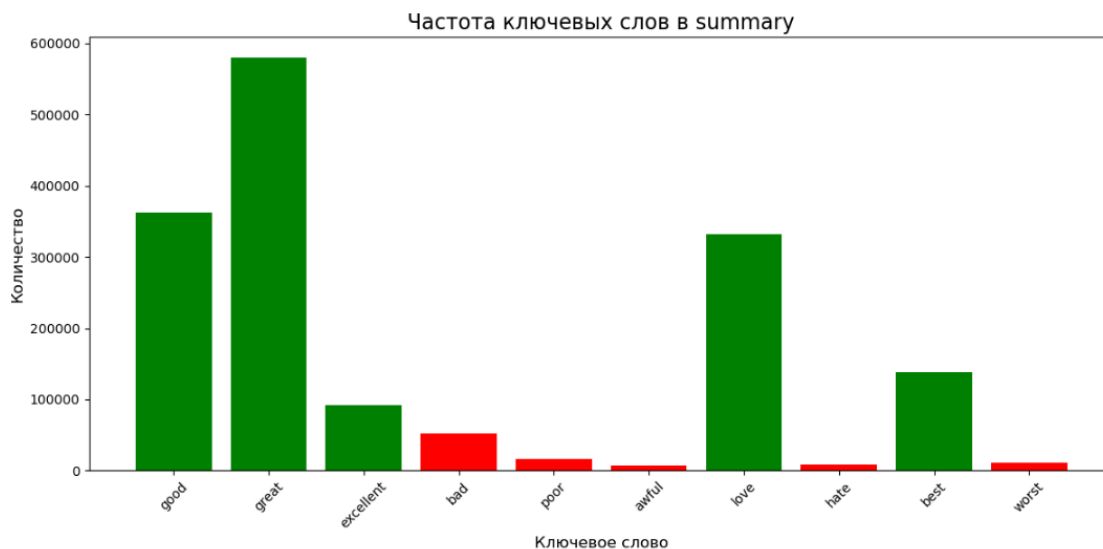


Рисунок 10 – Частоты для summary

Была построена корреляционная матрица числовых признаков. На основе которой уже можно сделать выводы по тому, как признаки коррелируют, например, с overall (рис. 11). Корреляционная матрица была построена с помощью следующего кода:

```
numeric_columns = ["overall", "text_length",
                   "summary_length", "review_year"]
correlation_matrix = {}
for col1 in numeric_columns:
    correlation_matrix[col1] = {}
    for col2 in numeric_columns:
        corr_value = df_correlation.select(corr(col1, col2))
                               .collect()[0][0]
        correlation_matrix[col1][col2] = corr_value
corr_matrix_pd = pd.DataFrame(correlation_matrix)
plt.figure(figsize=(10, 8))
sns.heatmap(corr_matrix_pd, annot=True, cmap='coolwarm', center=0,
            square=True, fmt='.3f', linewidths=0.5)
plt.title('Корреляционная матрица числовых признаков')
plt.tight_layout()
plt.show()
```

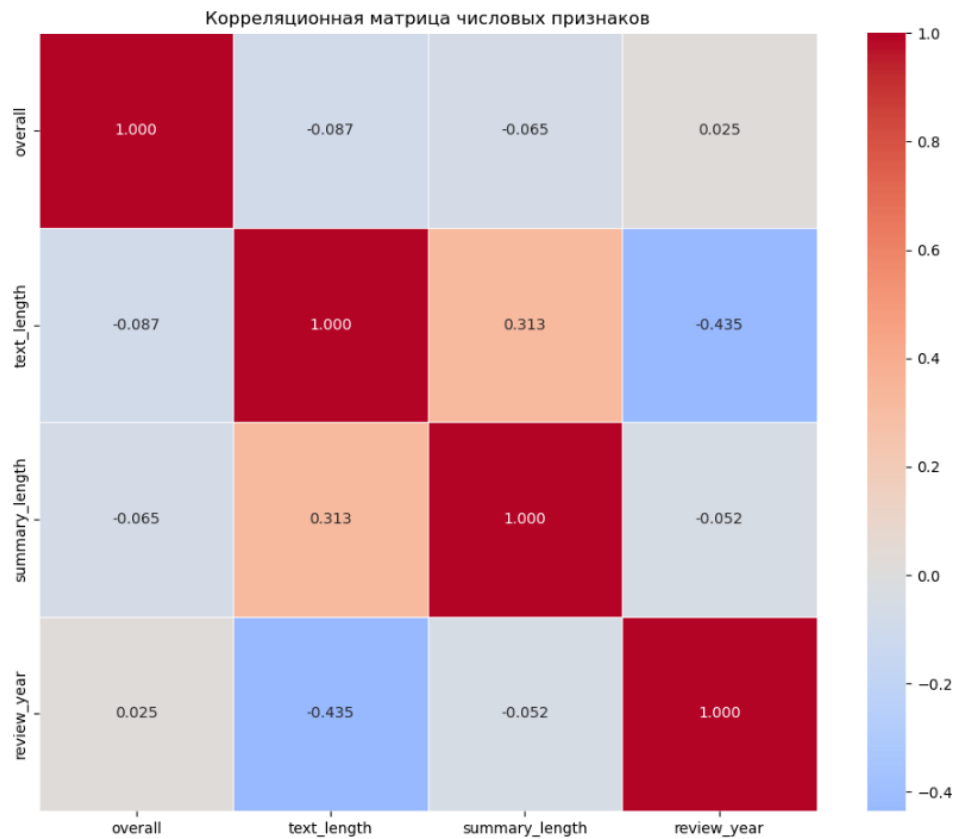


Рисунок 11 – Корреляционная матрица числовых признаков

Анализ корреляционной матрицы показал, что текущие признаки слабо коррелируют между собой.

## 1.5 Выводы

В ходе проведённого разведочного анализа был сформирован целостный и очищенный набор данных, готовый для применения алгоритмов машинного обучения. Основными результатами являются:

- выполнена загрузка и объединение данных из HDFS средствами Apache Spark;
- исследована структура данных, выявлены и обработаны пропуски и аномалии, преобразованы числовые признаки;
- сформированы новые признаки, повышающие информативность данных;
- проведён анализ распределений.

## 2 МАШИННОЕ ОБУЧЕНИЕ НА БОЛЬШИХ ДАННЫХ

В данной главе рассматриваются методы построения и оценки моделей машинного обучения в распределённой среде Apache Spark [1–3; 6]. Работа включает решение двух задач: прогнозирования числовой оценки отзыва (регрессия) и классификации полезности пользовательских отзывов. Все вычисления выполнялись с использованием фреймворка Apache Spark и библиотеки Spark ML [7; 8], обеспечивающих обработку данных объёмом более шести миллионов записей.

### 2.1 Задача регрессии

#### 2.1.1 Постановка задачи регрессии

Необходимо построить модель линейной регрессии для предсказания оценки рейтинга отзывов на фильмы и ТВ шоу (overall) на основе доступных признаков. Цель: найти линейную зависимость между признаками и целевой переменной, минимизируя ошибку предсказания. Качество модели оценивается метриками RMSE и  $R^2$  на тестовой выборке. Модель должна объяснить, какие факторы влияют на полезность отзывов и насколько сильно.

Признаки для моделирования были разделены на три группы по типу данных.

Бинарные признаки представлены двумя флагами:

- `isLongReview` — является ли отзыв длинным (длина текста превышает 100 символов);
- `isDetailedSummary` — содержит ли краткое описание (summary) развернутый текст (более 10 символов).

Числовые признаки включают метрики, связанные со временем, структурой и эмоциональной окраской отзывов:

- Временные атрибуты: метка времени в Unix-формате (`unixReviewTime`), год (`reviewYear`), месяц (`reviewMonth`) и день недели (`reviewDayOfWeek`) публикации отзыва.
- Структурные характеристики текста: длина полного текста отзыва (`reviewTextLength`), длина краткого описания (`summaryLength`), а также количество слов в отзыве (`reviewWordCount`).
- Меры эмоциональной окраски: количество позитивных (`positiveCount`) и негативных (`negativeCount`) слов, их соотношение (`sentimentRatio`) и количество восклицательных знаков как индикатор эмоциональности (`exclamationCount`).

Категориальные признаки в данной модели не используются, так как все значимые атрибуты были либо преобразованы в числовой формат, либо представлены в виде бинарных флагов.

После проведения разведочного анализа были добавлены новые признаки, которые должны улучшить обучаемость модели за счет большего влияния на переменную `overall`: числовые признаки `positiveCount`, `negativeCount`, `sentimentRatio`, `exclamationCount`. Их добавление в тестовую и обучающую выборки были реализованы с помощью одной функции. См. Приложение А.

	asin	reviewerID	reviewerName	reviewText	overall	summary	unixReviewTime	reviewTime	reviewWordCount	positiveCount	negativeCount	sentimentRatio	summaryLength	exclamationCount	reviewYear	re
0	0001526863	A20I3BVXGLBFXP	Judy K. Pilewski	My grandchildren love this DVD, even as dated ...	5.0	My grandchildren love this DVD	1.465776e+09	2016-06-13	23	1	0	10.0	30	0	2016	
1	0001527665	A38LY2SSHVHR8B	Jodie Vesely	This is a great movie for a missionary going i...	4.0	Good Movie! Great for cross-cultural missionar...	1.356480e+09	2012-12-26	31	1	0	10.0	50	0	2012	
2	0005000009	A33R55FMQ2PDH4	Connie	Excellent video and I thoroughly enjoyed it ve...	5.0	Where Jesus Walked	1.397434e+09	2014-04-14	26	1	0	10.0	18	0	2014	
3	0005000009	A30L9JYLV8TIWN	bullett	not what I was looking for.	1.0	One Star	1.429661e+09	2015-04-22	6	0	0	1.0	8	0	2015	
4	0005000009	ABRIR6R7IYMYX	carmen	Very nice movie	5.0	Five Stars	1.457568e+09	2016-10-03	3	1	0	10.0	10	0	2016	
5	0005019281	A18GZL78C27HOC	T. Clark	Great classic, the Fonzi rules	5.0	Five Stars	1.482538e+09	2016-12-24	5	1	0	10.0	10	0	2016	
6	0005019281	A18LQQ6KUK9IF6	Klaus2017	Not the best acting. The story is a classic so...	3.0	Great story not the best acting.	1.387325e+09	2013-12-18	34	1	1	1.0	32	0	2013	
7	0005019281	A1BOK08CB8RVSKD	Laura D	Have been looking for the Henry Winkler 'An Am...	5.0	Finally I Found It!	1.389053e+09	2014-07-01	22	0	0	1.0	19	1	2014	
8	0005019281	A1QZKM8M6866I8	Amazon Customer	While Winkler's make up job is dated and not r...	5.0	A nice version of Christmas carol - winkler do...	1.481242e+09	2016-09-12	35	1	0	10.0	59	0	2016	
9	0005019281	A15A7NXJUV1YOJZ	Thomas A. Kelly	"I had a copy of the original DVD but ufortuna...	5.0	A GREAT VERSION OF A CLASSIC.	1.389226e+09	2014-09-01	48	1	0	10.0	29	0	2014	

Рисунок 12 – Фрагмент датафрейма с новыми данными

Для оценки качества модели использовались метрики RMSE (Root Mean Square Error) и  $R^2$  (коэффициент детерминации).

### 2.1.2 Решение задачи регрессии

Построение модели линейной регрессии начиналось с подготовки данных: датасет загружался из HDFS в формате Parquet, после чего из него выделялся небольшой сэмпл для последующей потоковой обработки. Основная выборка разделялась на тренировочную и тестовую части в соотношении 80/20:

```
train_df, test_df = df.randomSplit([0.8, 0.2])
```

Далее выполнялась предобработка признаков. Числовые — приводились к вектору и нормализовывались через MinMaxScaler. Все признаки объединялись в единый вектор посредством VectorAssembler.

На основе этих этапов формировался конвейер Spark ML, включающий индексацию, кодирование, масштабирование признаков и модель линейной регрессии:

```
stages = []
vector_num_assembler = VectorAssembler(inputCols=numeric_features,
                                         outputCol="numeric_vector")
stages.append(vector_num_assembler)
numeric_scaler = MinMaxScaler(inputCol="numeric_vector",
                              outputCol="numeric_vector_scaled")
stages.append(numeric_scaler)
feature_cols = ["numeric_vector_scaled"] + binary_features
vector_all_assembler = VectorAssembler(inputCols=feature_cols,
                                         outputCol="features")
stages.append(vector_all_assembler)
linear_regression = LinearRegression(featuresCol="features",
                                     labelCol=label_col,
                                     predictionCol="prediction",
                                     standardization=False,
```

```

maxIter=max_iter)

stages.append(linear_regression)

pipeline = Pipeline(stages=stages)

```

Для неё использовались параметры  $\text{maxIter}=15$ ,  $\text{regParam}=0.01$  и отключённая стандартная нормализация ( $\text{standardization}=\text{False}$ ).

Оптимизация модели выполнялась с помощью 3-кратной кросс-валидации. Наилучшие результаты показала конфигурация с  $\text{regParam}=0.01$  и  $\text{elasticNetParam}=1.0$ , что соответствует L1 регуляризации.

### 2.1.3 Анализ полученных результатов регрессии

Модель была протестирована на отложенной тестовой выборке. Получены следующие значения метрик, а также график зависимости значения функции ошибки от номера итерации (см. рис. 13 и Приложение Б):

- $\text{RMSE} = 1.0718$ ;
- $R^2 = 0.2026$ .

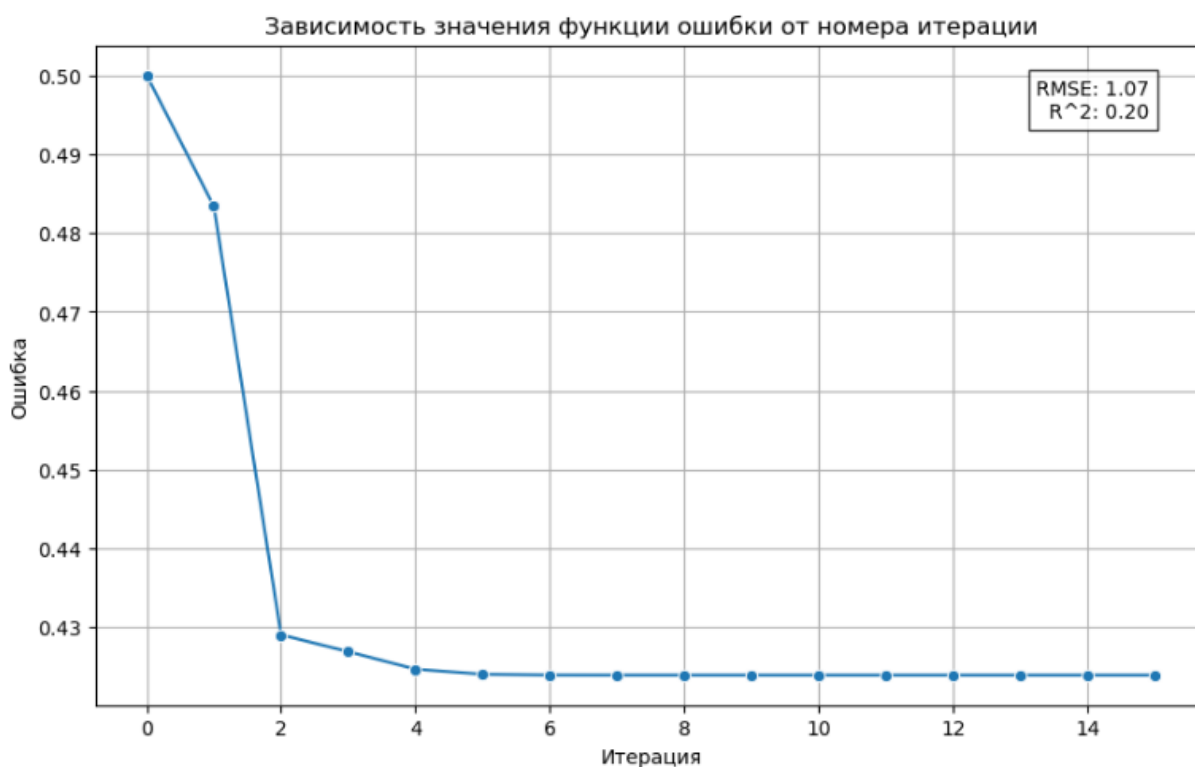


Рисунок 13 – График сходимости: изменение значения функции ошибки по итерациям

Низкое значение  $R^2$  свидетельствует о слабой объясняющей способности модели. Основные причины:

- слабая корреляция признаков с целевой переменной;
- возможные нелинейные зависимости, не учитываемые линейной моделью.

## 2.2 Задача классификации с использованием RandomForestClassifier

### 2.2.1 Постановка задачи классификации

Вторая часть работы посвящена построению модели бинарной классификации, определяющей полезность отзыва (is\_helpful). Целевой переменной является бинарный индикатор полезности.

Требуется построить классификатор на основе Random Forest, предсказывающий полезность отзыва по доступным данным. Необходимо проанализировать работу модели на валидационной выборке, определить оптимальный порог принятия решения и представить модель, которая, гарантируя обнаружение не менее 60% всех полезных отзывов ( $\text{Recall} \geq 0.60$ ), обеспечивает при этом наивысшую возможную долю верных предсказаний среди всех отмеченных как полезные (Precision).

Признак overall был исключен из-за сильной корреляции с целевой переменной (0.8986), что предотвращает утечку данных (см. рис. 14). Пример кода для расчета корреляции признаков:

```
correlation_check = df.select(
    F.corr("overall", "is_helpful").alias("corr_overall")
```

```
Корреляция признаков с целевой переменной:
Overall rating: 0.8986
Review text length: -0.0811
Summary length: -0.0643
Word count: -0.0835
Has Long Review: -0.0683
Is Detailed Summary: -0.1485
Has Long Review: -0.0683
```

Рисунок 14 – Корреляции признаков с целевой переменной

### 2.2.2 Решение задачи классификации

В задаче классификации данные также загружались из HDFS. Потом разбивались на выборки (рис. 15):

```
train_ratio = 0.8
train_df, test_df = df.randomSplit([train_ratio, 1 - train_ratio],
```

```
+-----+-----+
|is_helpful| count|
+-----+-----+
|          1|4176436|
|          0| 975813|
+-----+-----+
```

Рисунок 15 – Кол-во экземпляров классов в обучающей выборке

Видно, что классы находятся в дисбалансе, поэтому необходим oversampling, чтобы разбить данные примерно поровну. Для этого используем функцию oversample:

```
def oversample(data: DataFrame, column: str) -> DataFrame:
    pos = data.filter(F.col(column) == 1)
    neg = data.filter(F.col(column) == 0)

    total_pos = pos.count()
    total_neg = neg.count()

    if total_neg < total_pos:
        num_duplicates = total_pos - total_neg

        oversampled_neg = neg.withColumn(
            "dummy",
            F.explode(
                F.array_repeat(F.lit(1),
```

```

num_duplicates // total_neg + 1)
    )
    ).drop("dummy")

    balanced_df = pos.union(oversampled_neg)
else:
    balanced_df = data

return balanced_df

```

```

+-----+-----+
|is_helpful| count|
+-----+-----+
|          1|4176436|
|          0|3903252|
+-----+-----+

```

```

Баланс классов после oversampling:
Класс 1: 4176436 записей (51.7%)
Класс 0: 3903252 записей (48.3%)

```

Рисунок 16 – Кол-во экземпляров классов после балансировки

На этапе предобработки категориальные признаки преобразовывались с помощью `StringIndexer`, после чего все признаки объединялись в общий вектор, необходимый для обучения модели. Нормализация числовых признаков не проводилась, так как алгоритм случайного леса устойчив к масштабу данных.

Процесс обучения реализовывался через конвейер, включающий этапы подготовки данных и модель `RandomForestClassifier`:

```

string_indexer = StringIndexer(inputCols=categorical_features,
                                outputCols = [
                                    f"{feature}_index"
                                    for feature in categorical_features
                                ],
                                handleInvalid="keep",
                                stringOrderType="frequencyDesc")

```

```

all_features = (
    [f"{feature}_index" for feature in categorical_features]
    + binary_features
    + numeric_features
)

vector_assembler = VectorAssembler(
    inputCols=all_features,
    outputCol="features"
)

rf_classifier = RandomForestClassifier(
    featuresCol="features",
    labelCol=label_col,
    predictionCol="prediction",
    probabilityCol="probability",
    rawPredictionCol="rawPrediction",
    numTrees=100,
    maxDepth=max_depth,
    maxBins=32,
    minInstancesPerNode=1,
    subsamplingRate=1.0,
    featureSubsetStrategy="auto",
    impurity="gini",
    seed=42
)

pipeline = Pipeline(stages=[
    string_indexer, vector_assembler, rf_classifier
])

```

Для начального варианта использовались параметры numTrees=40, maxDepth=10 и subsamplingRate=0.8.

Оптимизация гиперпараметров выполнялась с использованием 3-кратной кросс-валидации. Наилучшие результаты были достигнуты при  $\text{maxDepth}=4$ ,  $\text{numTrees}=40$  и  $\text{subsamplingRate}=0.8$ .

### 2.2.3 Анализ полученных результатов классификации

Модель классификации продемонстрировала стабильные результаты: точность (Precision) составила 0.86, полнота (Recall) — 0.95, F1-мера — 0.90, а общая точность классификации достигла 0.83, а также получены графики зависимости значения функции ошибки от номера итерации (см. рис. 17 и 18 и Приложение В):

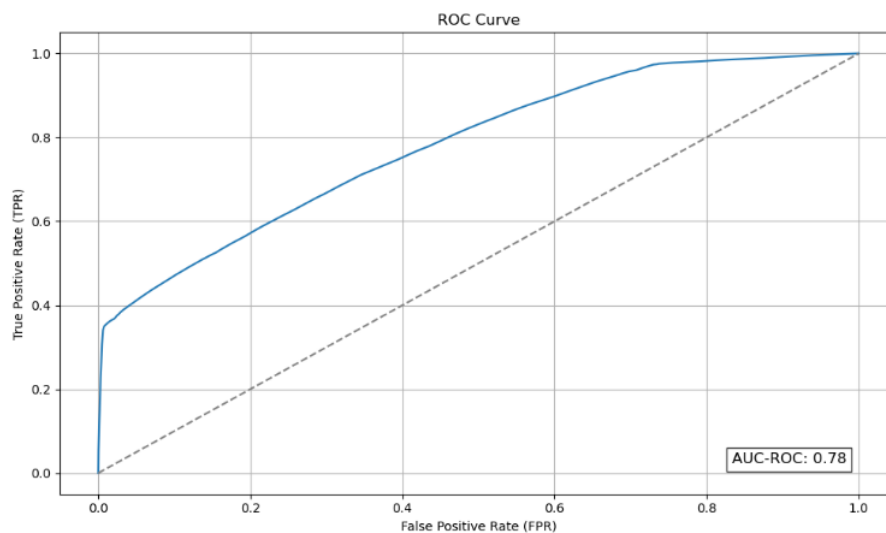


Рисунок 17 – ROC-кривая и значение AUC-ROC

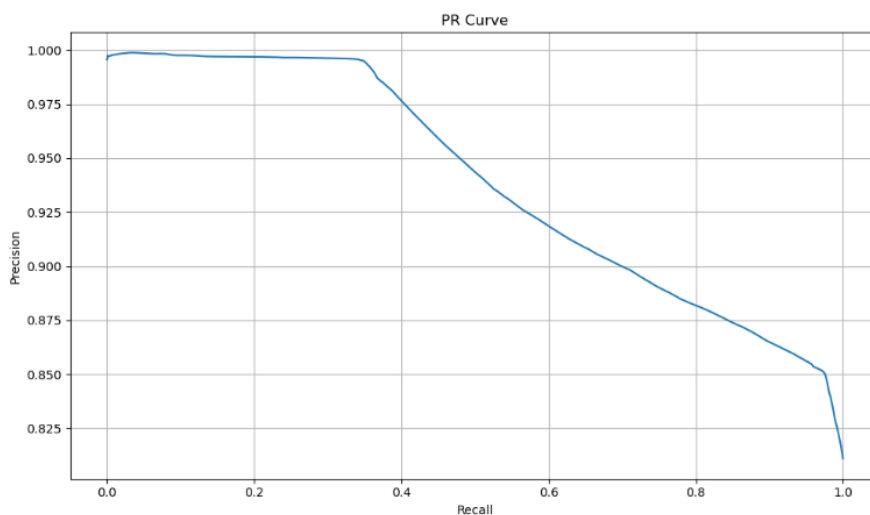


Рисунок 18 – PR-кривая

Построенная ROC-кривая показала AUC-ROC, равный 0.78, что свидетельствует о достаточной способности модели различать классы. Для выполнения требования полноты не ниже 60% был подобран порог вероятности 0.48. При таком пороге точность увеличилась до 0.92, полнота составила 0.60, F1-мера — 0.73, а значение Accuracy — 0.66.

Матрица ошибок для выбранного порога показывает следующие значения: 629 879 объектов были корректно классифицированы как положительные, 186 973 — как отрицательные; при этом 56 347 наблюдений были ошибочно отнесены к положительному классу, а 414 095 — пропущены моделью.

Таблица 2 – Матрица ошибок и метрики

TP: 629879	FP: 56347
FN: 414095	TN: 186973
precision	0.918
recall	0.603
f1	0.728
accuracy	0.634

Анализ важности признаков показал, что наибольший вклад в формирование решения оказывают длина текста отзыва, оценка пользователя, время публикации и длина заголовка. Именно эти признаки в большей степени определяют вероятность того, что отзыв будет признан полезным.

## 2.3 Выводы

В рамках работы были решены две задачи машинного обучения. Линейная регрессия показала ограниченную эффективность: низкое значение  $R^2$  (0.20) подтверждает, что текущие признаки слабо коррелируют. Напротив, задача классификации оказалась более успешной: модель достигла AUC-ROC = 0.78,  $F1 = 0.64$  и выполнила требуемый уровень полноты ( $Recall \geq 60\%$ ).

## ЗАКЛЮЧЕНИЕ

В рамках курсовой работы был успешно осуществлен полный цикл обработки больших данных и построения моделей машинного обучения на примере набора данных «Movies and TV» с применением платформы Apache Spark. В процессе исследования выполнен разведочный анализ, который охватил загрузку, очистку и трансформацию информации из двух исходных наборов. Проведена работа с пропущенными значениями, сформированы дополнительные признаки, изучены распределения и аномалии, что в итоге позволило подготовить качественный датасет для дальнейшего моделирования.

Для решения поставленных задач были реализованы и протестированы две модели машинного обучения. Линейная регрессия для прогнозирования пользовательских оценок показала устойчивую сходимость, однако низкое значение  $R^2 = 0,20$  указывает на необходимость использования более информативных признаков. Модель Random Forest для классификации полезности отзывов продемонстрировала хорошее качество ( $AUC-ROC = 0,78$ ) и выполнила поставленное условие: полнота (Recall) не ниже 60% при точности (Precision) 92%.

Применение распределённых вычислений на платформе Apache Spark подтвердило свою эффективность при работе с большими объёмами данных, обеспечив масштабируемость и высокую производительность на всех этапах проекта.

Перспективы дальнейшего развития работы включают применение методов обработки естественного языка (NLP) для извлечения семантических признаков из текстов отзывов, использование более сложных моделей, добавление временных и сезонных признаков.

## СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Изучаем Spark: молниеносный анализ данных / Карау, Х. [и др.]. — ДМК Пресс, 2015. — 304 с. — ISBN 978-5-97060-274-6.
2. Изучаем Spark: Быстрый анализ данных / Дамджи, Дж. [и др.]. — ДМК Пресс, 2020. — 450 с. — ISBN 978-5-93700-102-8.
3. *Чемберс, Б., Захария, М.* Spark: Полное руководство. — Питер, 2018. — 598 с. — ISBN 978-5-4461-0599-5.
4. *Koirala, Roshan.* Exploratory Data Analysis with pySpark. — 2020. — URL: [https://github.com/roshankoirala/pySpark\\_tutorial/blob/master/Exploratory\\_data\\_analysis\\_with\\_pySpark.ipynb](https://github.com/roshankoirala/pySpark_tutorial/blob/master/Exploratory_data_analysis_with_pySpark.ipynb) (visited on 09/19/2022).
5. *The Apache Software Foundation.* Официальный сайт Apache Spark. — 2022. — URL: <https://spark.apache.org/> (visited on 09/19/2022).
6. Spark SQL: Relational Data Processing in Spark / Armbrust, Michael [et al.] // Proceedings of the ACM SIGMOD International Conference on Management of Data. — 2015. — С. 1383–1394. — DOI: 10.1145/2723372.2742797.
7. *jmcauley.* Movies and TV. — 2018. — URL: [https://jmcauley.ucsd.edu/data/amazon\\_v2/index.html](https://jmcauley.ucsd.edu/data/amazon_v2/index.html) (visited on 09/15/2025).
8. *Tekdoğan, T., Çakmak, A.* Benchmarking Apache Spark and Hadoop MapReduce on Big Data Classification // 2021 5th International Conference on Cloud and Big Data Computing. — ACM, 2022. — 15–20. — DOI: 10.1145/3481646.3481649.

## ПРИЛОЖЕНИЕ А

### Изменение признаков

```
def add_features(df):
    return (df
            .withColumn("reviewTextLength",
                        length("reviewText"))
            .withColumn("summaryLength",
                        length("summary"))
            .withColumn("isLongReview",
                        when(col("reviewTextLength") > 100, 1)
                        .otherwise(0))
            .withColumn("isDetailedSummary",
                        when(col("summaryLength") > 10, 1)
                        .otherwise(0))
            .withColumn("reviewYear",
                        year("reviewTime"))
            .withColumn("reviewMonth",
                        month("reviewTime"))
            .withColumn("reviewDayOfWeek",
                        dayofweek("reviewTime"))
            .withColumn("reviewWordCount",
                        udf(lambda x: len(x.split())
                            if x and isinstance(x, str) else 0,
                            IntegerType())("reviewText"))
            .withColumn("positiveCount",
                        udf(lambda x: sum(1 for word in positive_words
                            if word in x.lower())
                            if x and isinstance(x, str) else 0,
                            IntegerType())("reviewText"))
            .withColumn("negativeCount",
                        udf(lambda x: sum(1 for word in negative_words
                            if word in x.lower())
                            if x and isinstance(x, str) else 0,
```

```

        IntegerType())("reviewText"))
.withColumn("sentimentRatio",
    when(col("negativeCount") == 0,
        when(col("positiveCount") > 0, 10)
        .otherwise(1))
    .otherwise(col("positiveCount") /
        col("negativeCount")))
.withColumn("exclamationCount",
    udf(lambda x: x.count('!'))
        if x and isinstance(x, str) else 0,
    IntegerType())("reviewText"))

```

## ПРИЛОЖЕНИЕ Б

### Построение графика зависимости значения функции ошибки от номера итерации

```
def plot_training_summary(cv_model: CrossValidatorModel) -> None:
    best_model = cv_model.bestModel

    training_summary = best_model.stages[-1].summary

    objective_history = training_summary.objectiveHistory
    print(objective_history)

    plt.figure(figsize=(10, 6))
    sns.lineplot(x=range(len(objective_history)),
                  y=objective_history,
                  marker='o')
    plt.xlabel('Итерация')
    plt.ylabel('Ошибка')
    plt.title(
        "Зависимость значения функции ошибки от номера итерации"
    )

    rmse = training_summary.rootMeanSquaredError
    r2 = training_summary.r2

    plt.text(0.95, 0.95, f"RMSE: {rmse:.2f}\nR^2: {r2:.2f}",
              transform=plt.gca().transAxes, ha='right', va='top',
              bbox=dict(facecolor='white', alpha=0.8), zorder=5)
    plt.grid()

    plt.show()
```

## ПРИЛОЖЕНИЕ В

### Построение графиков ROC и PR кривых

```
def plot_roc_pr_curves(pd_data: pd.DataFrame,
                        auc_roc: float) -> None:
    fig, axes = plt.subplots(1, 2, figsize=(20, 6))

    sns.lineplot(x="FPR", y="TPR", data=pd_data, ax=axes[0])
    sns.lineplot(x=[0, 1], y=[0, 1],
                  color="gray", linestyle="--", ax=axes[0])
    axes[0].set_xlabel("False Positive Rate (FPR)")
    axes[0].set_ylabel("True Positive Rate (TPR)")
    axes[0].set_title("ROC Curve")
    axes[0].grid()

    axes[0].text(0.94, 0.06, f'AUC-ROC: {auc_roc:.2f}',
                 transform=axes[0].transAxes,
                 fontsize=12, verticalalignment='bottom',
                 horizontalalignment='right',
                 bbox=dict(facecolor='white',
                           alpha=0.8), zorder=5)

    sns.lineplot(x="TPR", y="Precision",
                  data=pd_data, ax=axes[1])
    axes[1].set_xlabel("Recall")
    axes[1].set_ylabel("Precision")
    axes[1].set_title("PR Curve")
    axes[1].grid()

    plt.tight_layout()
    plt.show()
```