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 поведения пользователя в электронной коммерции с использованием фреймворка Apache Spark

Студент Челядинов Дмитрий Владимирович
(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

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

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.

ЗАДАНИЕ
на курсовую работу (проект)

Студент Челядинов Дмитрий Владимирович
(фамилия, имя, отчество)

Группа САПР-1.1

1. Тема: Исследование данных поведения пользователя в электронной коммерции с использованием фреймворка Apache Spark

Утверждена приказом от «___» _____ 20 ___ г., № _____.

2. Срок представления работы (проекта) к защите «___» _____ 20 ___ г.

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

4. Перечень графического материала: _____

5. Дата выдачи задания «___» _____ 20 ___ г.

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

Задание принял к исполнению _____ Д.В. Челядинов
(подпись и дата подписания) (инициалы и фамилия)

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Актуальность данной курсовой работы обусловлена возрастающей потребностью в разработке эффективных методов машинного обучения для обработки больших данных [1–3] и извлечения полезных знаний из пользовательского контента [4]. Особую значимость приобретают комплексные подходы, сочетающие предварительную обработку данных и построение прогностических моделей на распределенных вычислительных платформах.

В связи с этим целью данной курсовой работы является исследование и реализация полного цикла машинного обучения на больших данных о поведении пользователя на рынке электронной коммерции с использованием фреймворка 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

Задачей данной главы является проведение комплексного EDA большого набора данных о поведении пользователей электронной коммерции с использованием возможностей фреймворка Apache Spark.

Для исследования применяются распределённые вычисления, что позволяет эффективно обрабатывать миллионы записей [5; 7] и анализировать данные в условиях ограничений по памяти и времени. Использование PySpark обеспечивает масштабируемость, а интеграция с HDFS — удобство работы с большими объёмами 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.

1.2 Описание исходного датасета

В работе используется датасет «eCommerce behavior data from multi category store», доступный на платформе Kaggle [8]. Набор данных состоит

из одного датасета — 2019-Nov.csv. В дальнейшем его название изменится на dataset.csv.

Датафрейм включает следующие ключевые признаки (таблица 1):

Таблица 1 – Описание признаков датасета

Признак	Описание
event_time	Время, когда произошло событие (UTC).
event_type	Вид события.
product_id	Идентификатор продукта.
category_id	Идентификатор категории продукта.
category_code	Таксономия категории товара (кодовое название).
brand	Строка с названием бренда.
price	Цена продукта.
user_id	Идентификатор пользователя.

Совокупный объём данных составляет более 67 миллионов строк. Загруженные данные изначально имеют строковые типы и разнородные форматы идентификаторов (см. рис. 1). Чтение и демонстрация исходных данных производилось с помощью следующего кода:

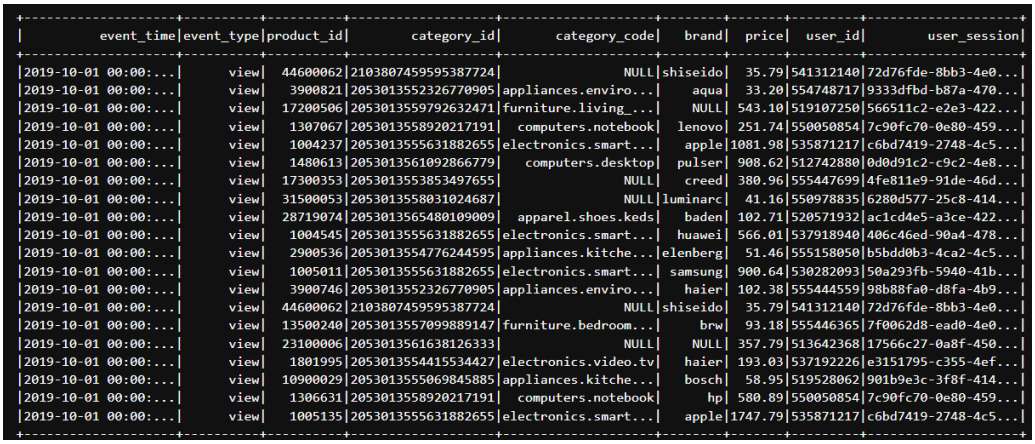
```
path = "hdfs://namenode:9000/user/dchel/dataset.csv"
df = (spark.read.format("csv")
      .option("header", "true")
      .load(path)
)
df.show()
```

Данный фрагмент показывает, что:

– исходные данные находятся в HDFS;

– происходит чтение csv файла dataset.csv;

– происходит вывод первых 20 строк датафрейма в консоль.



event_time	event_type	product_id	category_id	category_code	brand	price	user_id	user_session
2019-10-01 00:00:...	view	44600062	2103807459595387724	NULL	shiseido	35.79	541312140	72d76fde-8bb3-4e0...
2019-10-01 00:00:...	view	3900821	2053013552326770905	appliances.enviro...	aqua	33.20	554748717	9333dfbd-b87a-470...
2019-10-01 00:00:...	view	17200506	2053013559792632471	furniture.living_...	NULL	543.10	519107250	566511c2-e2e3-422...
2019-10-01 00:00:...	view	1307067	2053013558920217191	computers.notebook	lenovo	251.74	550050854	7c90fc70-0e80-459...
2019-10-01 00:00:...	view	1004237	2053013555631882655	electronics.smart...	apple	1081.98	535871217	c6bd7419-2748-4c5...
2019-10-01 00:00:...	view	1480613	2053013561092866779	computers.desktop	pulser	908.62	512742880	0d0d91c2-c9c2-4e8...
2019-10-01 00:00:...	view	17300353	2053013553853497655	NULL	creed	380.96	555447699	4fe811e9-91de-46d...
2019-10-01 00:00:...	view	31500053	2053013558031024687	NULL	luminarc	41.16	550978835	6280d577-25c8-414...
2019-10-01 00:00:...	view	28719074	2053013565480109009	apparel.shoes.keds	baden	102.71	520571932	ac1cd4e5-a3ce-422...
2019-10-01 00:00:...	view	1004545	2053013555631882655	electronics.smart...	huawei	566.01	537918940	406c46ed-90a4-478...
2019-10-01 00:00:...	view	2900536	2053013554776244595	appliances.kitche...	elenberg	51.46	555158050	b5bdd0b3-4ca2-4c5...
2019-10-01 00:00:...	view	1005011	2053013555631882655	electronics.smart...	samsung	900.64	530282093	50a293fb-5940-41b...
2019-10-01 00:00:...	view	3900746	2053013552326770905	appliances.enviro...	haier	102.38	555444559	98b88fa0-d8fa-4b9...
2019-10-01 00:00:...	view	44600062	2103807459595387724	NULL	shiseido	35.79	541312140	72d76fde-8bb3-4e0...
2019-10-01 00:00:...	view	13500240	2053013557099889147	furniture.bedroom...	brw	93.18	555446365	7f0062d8-ead0-4e0...
2019-10-01 00:00:...	view	23100006	2053013561638126333	NULL	NULL	357.79	513642368	17566c27-0a8f-450...
2019-10-01 00:00:...	view	1801995	2053013554415534427	electronics.video.tv	haier	193.03	537192226	e3151795-c355-4ef...
2019-10-01 00:00:...	view	10900029	2053013555069845885	appliances.kitche...	bosch	58.95	519528062	901b9e3c-3f8f-414...
2019-10-01 00:00:...	view	1306631	2053013558920217191	computers.notebook	hp	580.89	550050854	7c90fc70-0e80-459...
2019-10-01 00:00:...	view	1005135	2053013555631882655	electronics.smart...	apple	1747.79	535871217	c6bd7419-2748-4c5...

Рисунок 1 – Данные из датасета

В ходе анализа полей датасета, с помощью команды `df.select("event_type", "product_id", "category_id", "category_code", "brand", "price")` были выбраны следующие поля: `event_type`, `product_id`, `category_id`, `category_code`, `brand`, `price` (см. рис. 2).

event_type	product_id	category_id	category_code	brand	price
view	44600062	2103807459595387724	NULL	shiseido	35.79
view	3900821	2053013552326770905	appliances.enviro...	aqua	33.20
view	17200506	2053013559792632471	furniture.living_...	NULL	543.10
view	1307067	2053013558920217191	computers.notebook	lenovo	251.74
view	1004237	2053013555631882655	electronics.smart...	apple	1081.98
view	1480613	2053013561092866779	computers.desktop	pulser	908.62
view	17300353	2053013553853497655	NULL	creed	380.96
view	31500053	2053013558031024687	NULL	luminarc	41.16
view	28719074	2053013565480109009	apparel.shoes.keds	baden	102.71
view	1004545	2053013555631882655	electronics.smart...	huawei	566.01
view	2900536	2053013554776244595	appliances.kitche...	elenberg	51.46
view	1005011	2053013555631882655	electronics.smart...	samsung	900.64
view	3900746	2053013552326770905	appliances.enviro...	haier	102.38
view	44600062	2103807459595387724	NULL	shiseido	35.79
view	13500240	2053013557099889147	furniture.bedroom...	brw	93.18
view	23100006	2053013561638126333	NULL	NULL	357.79
view	1801995	2053013554415534427	electronics.video.tv	haier	193.03
view	10900029	2053013555069845885	appliances.kitche...	bosch	58.95
view	1306631	2053013558920217191	computers.notebook	hp	580.89
view	1005135	2053013555631882655	electronics.smart...	apple	1747.79

Рисунок 2 – Данные из датасета после select

Структура данных была изучена с использованием команды `df.printSchema()`, что позволило определить типы полей и выявить их потенциальную неоднородность. Так, поля `event_type`, `product_id`, `category_id`, `category_code`, `brand`, `price` и текстовые поля загружаются как строки, что указывает на возможное наличие разнородных форматов данных. Структура представлена на рисунке (см. рис. 3).

```
root
 |-- event_type: string (nullable = true)
 |-- product_id: string (nullable = true)
 |-- category_id: string (nullable = true)
 |-- category_code: string (nullable = true)
 |-- brand: string (nullable = true)
 |-- price: string (nullable = true)
```

Рисунок 3 – Структура таблицы после загрузки данных

Более детальное изучение содержимого выполнялось уже на этапе разведочного анализа при помощи выборочного просмотра записей `df.show()` (см. рис. 4), анализа уникальных значений и регулярных выражений.

event_type	product_id	category_id	category_code	brand	price
view	1004767	NULL	[electronics, sma...	samsung	250.82
view	1307135	NULL	[computers, noteb...	hp	262.3
view	1005186	NULL	[electronics, sma...	samsung	771.94
cart	1801911	NULL	[electronics, vid...	samsung	1001.0
view	10900003	NULL	[appliances, kitc...	polaris	40.85
cart	2702050	NULL	[appliances, kitc...	lg	489.96
view	1004249	NULL	[electronics, sma...	apple	766.76
view	6200358	NULL	[appliances, envi...	almacom	43.76
view	1004856	NULL	[electronics, sma...	samsung	131.64
view	3601376	NULL	[appliances, kitc...	candy	254.81
view	1004856	NULL	[electronics, sma...	samsung	131.64
view	7900194	NULL	[furniture, kitch...	joie	73.1
view	21401211	NULL	[electronics, clo...	casio	78.5
view	1801750	NULL	[electronics, vid...	harper	127.93
view	1801995	NULL	[electronics, vid...	haier	193.03
view	1004767	NULL	[electronics, sma...	samsung	250.82
view	14701668	NULL	[furniture, livin...	brw	43.5
cart	1004659	NULL	[electronics, sma...	samsung	729.75
cart	1004833	NULL	[electronics, sma...	samsung	172.19
view	6301982	NULL	[appliances, kitc...	dauscher	15.42

Рисунок 4 – Выборочный просмотра записей

Эти методы позволили установить, что:

– числовые идентификаторы различной длины (от 6 до 8 цифр);

– только одно значение view во всех строках выборки;

– множество значений NULL;

– значительный разброс цен (от бюджетных товаров до премиальных);

– числовые значения с двумя десятичными знаками.

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 5 показана конфигурация SparkSession.

В конфиге, указанном ниже, строчки указывают на то, что используется spark:

```
def create_spark_configuration() -> SparkConf:
    user_name = "dchel"

    conf = SparkConf()
    conf.setAppName("Lab 1")
    conf.setMaster("local[*]")
    conf.set("spark.submit.deployMode", "client")
    conf.set("spark.executor.memory", "12g")
    conf.set("spark.executor.cores", "8")
    conf.set("spark.executor.instances", "2")
    conf.set("spark.driver.memory", "4g")
    conf.set("spark.driver.cores", "2")
    conf.set("spark.sql.catalog.spark_catalog.type",
     "hadoop")
    conf.set("spark.sql.catalog.spark_catalog.warehouse",
     f"hdfs:///user/{user_name}")
    conf.set("spark.sql.catalog.spark_catalog.io-impl",
     "org.apache.iceberg.hadoop.HadoopFileIO")

    return conf
```

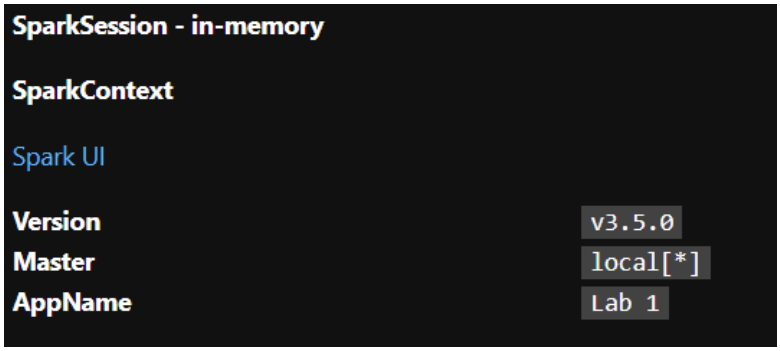


Рисунок 5 – Демонстрация работы сессии Spark

Также для работы с датасетом, он был предварительно загружен в hdfs. Обзор директории представлен на рисунке 6.

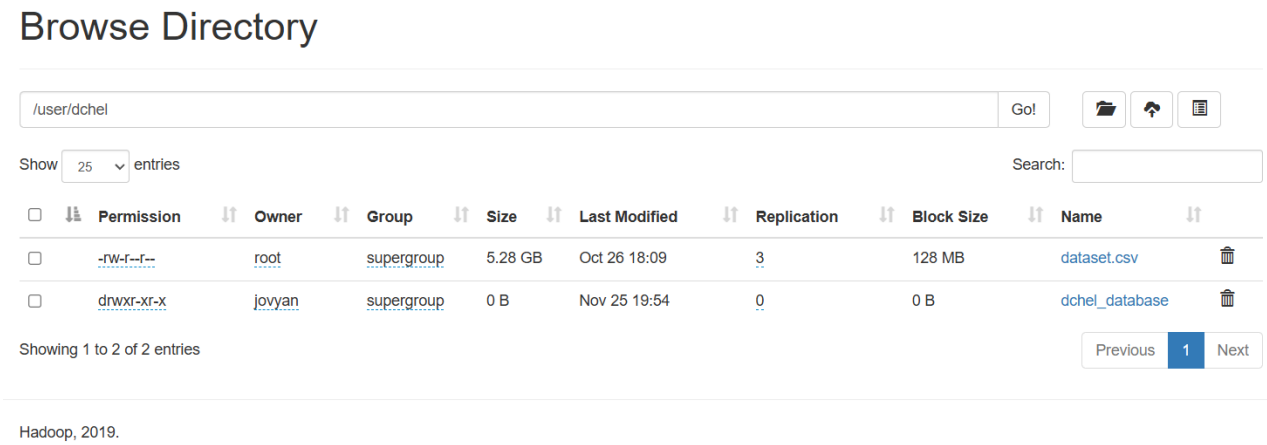


Рисунок 6 – Директория с данными для работы

1.3 Определение пропущенных значений и преобразование данных

Анализ полноты данных показал наличие пропусков в ряде столбцов, преимущественно в числовых полях. Для оценки количества пропусков была использована служебная функция, выполняющая подсчёт NULL-значений:

```
def count_nulls(data: DataFrame,
                column_name: str) -> None:
    null_counts = data.select(
        sum(col(column_name).isNull().cast("int"))
    ).collect()[0][0]
```

```
not_null_counts = data.select(
    sum(col(column_name).isNotNull().cast("int"))
).collect()[0][0]

print(f"Число колонок с NULL: {null_counts} "
      f"({100 * null_counts / (null_counts +
      not_null_counts):.2f}%)")
```

Были применены следующие стратегии:

– удалены строки содержащие NULL-значения в столбцах category_code и brand командой data.dropna(subset=["category_code", "brand"]);
– текстовый столбец category_id был удален с помощью команды df.drop("category_id").

После очистки структура данных была расширена и дополнена новыми признаками. В частности, выполнено преобразование типов:

– численный перевод идентификаторов продуктов (product_id);
– перевод кодов категории в массив строк (category_code);
– численный перевод цены (price).

Дополнительно созданы следующие производные признаки:

– массив кодов категорий, полученный путём разбиения строки с использованием split;
– признак содержания вида продукта contains_appliances, contains_computers, contains_electronics, contains_kitchen, contains_smartphone;
– булевый признак дороговизны продукта is_expensive;
– булевый признак бюджетного продукта is_budget;
– булевый признак среднебюджетного продукта is_mid_range;
– кол-во категорий, которые охватывают продукт category_count;
– булевый признак просмотра продукта is_view;
– булевый признак добавление продукта в корзину is_cart;

– булевый признак покупки продукта `is_purchase`;

– класс продукта `price_range`;

– класс продукта в численном формате `price_range_numeric`.

Результаты продемонстрированы на рисунках 7 и 8.

```
root
 |-- event_type: string (nullable = true)
 |-- product_id: integer (nullable = true)
 |-- brand: string (nullable = true)
 |-- price: double (nullable = true)
 |-- contains_appliances: boolean (nullable = true)
 |-- contains_computers: boolean (nullable = true)
 |-- contains_electronics: boolean (nullable = true)
 |-- contains_kitchen: boolean (nullable = true)
 |-- contains_smartphone: boolean (nullable = true)
 |-- is_expensive: integer (nullable = false)
 |-- is_budget: integer (nullable = false)
 |-- is_mid_range: integer (nullable = false)
 |-- category_count: integer (nullable = true)
 |-- is_purchase: integer (nullable = false)
 |-- is_view: integer (nullable = false)
 |-- is_cart: integer (nullable = false)
 |-- price_range: string (nullable = false)
 |-- price_range_numeric: integer (nullable = false)
```

Рисунок 7 – Структура таблицы после обработки данных

	event_type	product_id	brand	price	contains_appliances	contains_computers	contains_electronics	contains_kitchen	contains_smartphone	is_expensive	is_budget	is_mid_range	category_count	is_purchase	is_view
0	cart	1002042	samsung	77.139999	False	False	True	False	True	0	0	1	2	0	0
1	cart	1002524	apple	513.450012	False	False	True	False	True	1	0	0	2	0	0
2	cart	1002524	apple	513.469971	False	False	True	False	True	1	0	0	2	0	0
3	cart	1002524	apple	531.409973	False	False	True	False	True	1	0	0	2	0	0
4	cart	1002524	apple	533.260010	False	False	True	False	True	1	0	0	2	0	0
5	cart	1002524	apple	540.270020	False	False	True	False	True	1	0	0	2	0	0
6	cart	1002536	apple	576.570007	False	False	True	False	True	1	0	0	2	0	0
7	cart	1002542	apple	488.790009	False	False	True	False	True	1	0	0	2	0	0
8	cart	1002544	apple	460.070007	False	False	True	False	True	1	0	0	2	0	0
9	cart	1002544	apple	460.309998	False	False	True	False	True	1	0	0	2	0	0

Рисунок 8 – Фрагмент данных после обработки

Все преобразования были объединены в функцию, позволяющую повторно применять трансформации к датафрейму. Фрагмент кода вынесен в Приложение А.

1.4 Анализ распределений, выбросов и категориальных признаков

Для количественного признака `price`, была построена гистограмма с использованием Spark и последующей визуализацией в библиотеках Seaborn и Matplotlib (см. Приложение Б)

Полученные результаты показывают:

– Стоимость продуктов пользователей смещена в сторону невысоких значений (мода 125–150) (см. рис. 9);
– коэффициент полезности характеризуется бимодальным распределением, что отражает различия между поведениями пользователей (см. рис. 9).

Рисунок 9 – Гистограмма распределения для price

На рисунках 10, 11 продемонстрировано распределение аномальных значений у price.

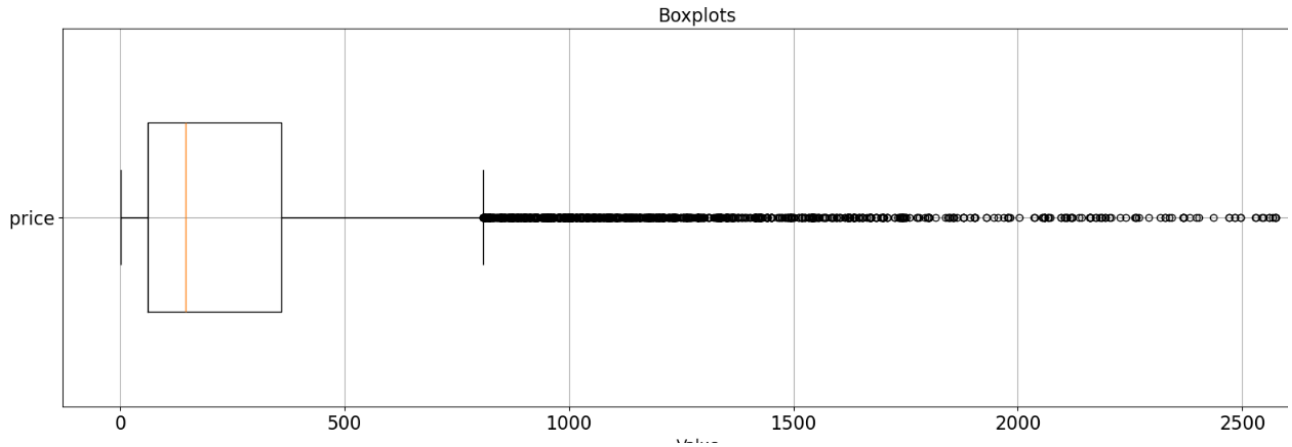


Рисунок 10 – Пример аномалий у price

Минимальное значение:	0.88
Среднее значение:	301.55
Среднеквадратичное отклонение:	390.91
Первый квартиль:	61.52
Медиана:	146.08
Третий квартиль:	360.11
Максимальное значение:	2574.07

Рисунок 11 – Расчетные значения у price

Для категориальных признаков `category_code`, `brand` были проанализированы частоты встречаемости. Самым популярным значением для поля:

– `brand` стало samsung (рис. 12);

– `category_code` стало electronics (рис. 13).

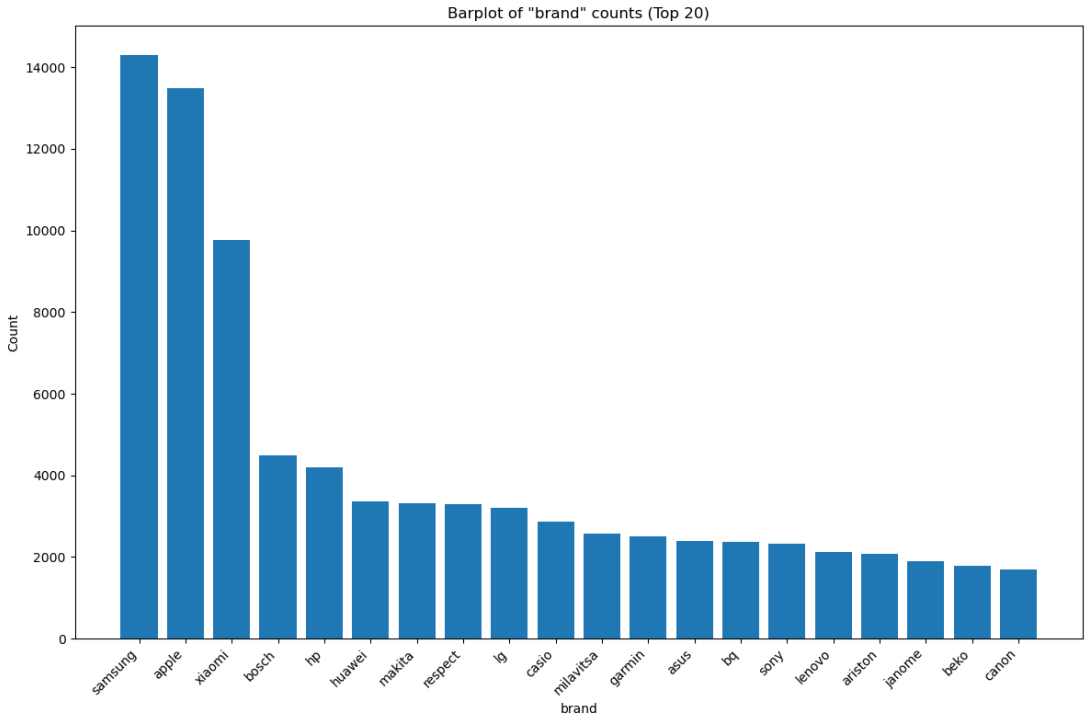


Рисунок 12 – Частоты для brand

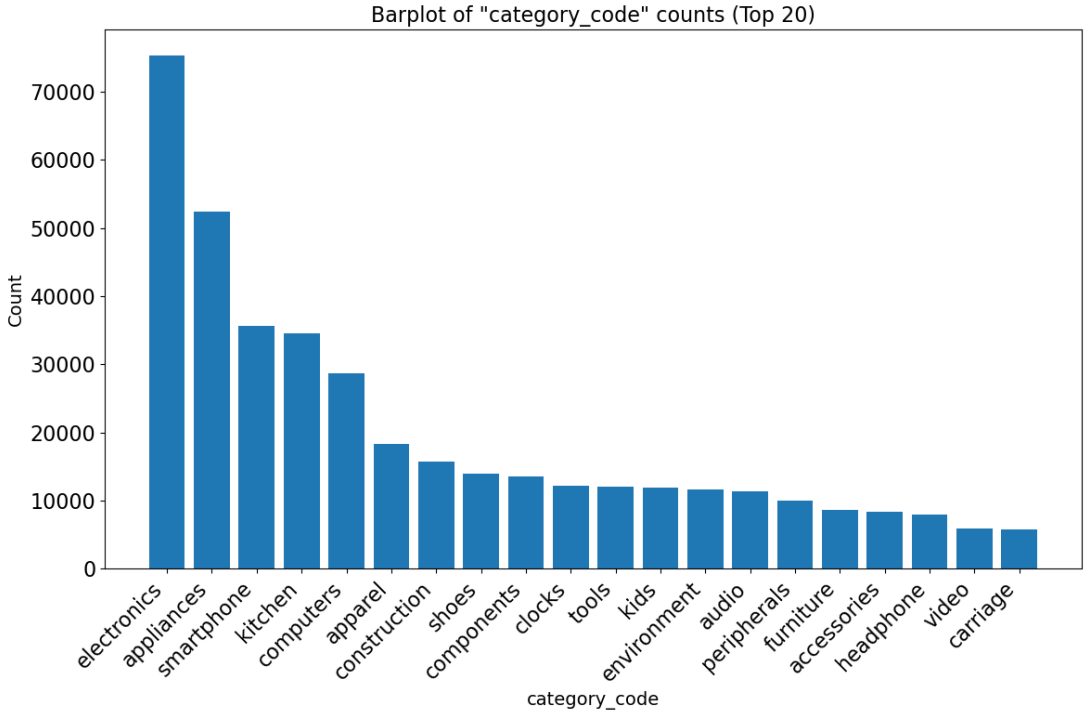


Рисунок 13 – Частоты для category_code

Проверка дубликатов показала, что некоторые товары имеют повторяющиеся записи — это связано с тем, что каждая запись соответствует отдельному пользовательскому действию. Такие дубликаты являются ожидаемыми и отражают структуру исходного набора данных. Поэтому для всего датасета была проведена дедубликация по `product_id`, `event_type` и `price` полям командой `df.dropDuplicates(["product_id", "event_type", "price"])`.

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;
- нормализованы текстовые поля и преобразованы числовые, бинарные признаки;
- сформированы новые признаки, повышающие информативность данных;
- проведён анализ распределений и выбросов в количественных характеристиках;
- выявлены особенности набора данных, связанные с дублированием записей по идентификатору продукта.

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 цены товара (регрессия) и классификации бюджетного класса продукта. Все вычисления выполнялись с использованием фреймворка Apache Spark и библиотеки Spark ML [8; 9], обеспечивающих обработку данных объёмом около трёх миллионов записей.

2.1 Задача регрессии

2.1.1 Постановка задачи регрессии

Необходимо построить модель GBT регрессии для предсказания цены продукта (price) на основе доступных признаков. Цель: найти нелинейную зависимость между признаками и целевой переменной, минимизируя ошибку предсказания. Требуется использовать RMSE и R^2 для оценки качества обучения модели. Модель должна объяснить, какие факторы влияют на цену товара и насколько сильно.

На основе следующих признаков:

```
binary_features = [
    "is_expensive",
    "is_budget",
    "is_mid_range",
    "is_purchase",
    "is_view",
    "is_cart",
    "contains_appliances",
    "contains_computers",
    "contains_electronics",
    "contains_kitchen",
```

```
    "contains_smartphone"
]

numeric_features = ["category_count"]

categorical_features = ["brand", "event_type",
                        "price_range", "price_range_numeric"]
```

проводилось обучение модели.

Из анализа были исключены признаки, не влияющие на целевую переменную или потенциально приводящие к переобучению: идентификаторы (`category_id`, `event_time`, `user_id`, `user_session`) (см. рис. 14-16).

event_time	event_type	product_id	category_id	category_code	brand	price	user_id	user_session
2019-10-01 00:00:...	view	44600062	2103807459595387724	NULL	shiseido	35.79	541312140	72d76fde-8bb3-4e0...
2019-10-01 00:00:...	view	3900821	2053013552326770905	appliances.enviro...	aqua	33.20	554748717	9333dfbd-b87a-470...
2019-10-01 00:00:...	view	17200506	2053013559792632471	furniture.living_...	NULL	543.10	519107250	566511c2-e2e3-422...
2019-10-01 00:00:...	view	1307067	2053013558920217191	computers.notebook	lenovo	251.74	550050854	7c90fc70-0e80-459...
2019-10-01 00:00:...	view	1004237	2053013555631882655	electronics.smart...	apple	1081.98	535871217	c6bd7419-2748-4c5...
2019-10-01 00:00:...	view	1480613	2053013561092866779	computers.desktop	pulser	908.62	512742880	0d0d91c2-c9c2-4e8...
2019-10-01 00:00:...	view	17300353	2053013553853497655	NULL	creed	380.96	555447699	4fe811e9-91de-46d...
2019-10-01 00:00:...	view	31500053	2053013558031024687	NULL	luminarc	41.16	550978835	6280d577-25c8-414...
2019-10-01 00:00:...	view	28719074	2053013565480109009	apparel.shoes.keds	baden	102.71	520571932	ac1cd4e5-a3ce-422...
2019-10-01 00:00:...	view	1004545	2053013555631882655	electronics.smart...	huawei	566.01	537918940	406c46ed-90a4-478...
2019-10-01 00:00:...	view	2900536	2053013554776244595	appliances.kitche...	elenberg	51.46	555158050	b5bdd0b3-4ca2-4c5...
2019-10-01 00:00:...	view	1005011	2053013555631882655	electronics.smart...	samsung	900.64	530282093	50a293fb-5940-41b...
2019-10-01 00:00:...	view	3900746	2053013552326770905	appliances.enviro...	haier	102.38	555444559	98b88fa0-d8fa-4b9...
2019-10-01 00:00:...	view	44600062	2103807459595387724	NULL	shiseido	35.79	541312140	72d76fde-8bb3-4e0...
2019-10-01 00:00:...	view	13500240	2053013557099889147	furniture.bedroom...	brw	93.18	555446365	7f0062d8-ead0-4e0...
2019-10-01 00:00:...	view	23100006	2053013561638126333	NULL	NULL	357.79	513642368	17566c27-0a8f-450...
2019-10-01 00:00:...	view	1801995	2053013554415534427	electronics.video.tv	haier	193.03	537192226	e3151795-c355-4ef...
2019-10-01 00:00:...	view	10900029	2053013555069845885	appliances.kitche...	bosch	58.95	519528062	901b9e3c-3f8f-414...
2019-10-01 00:00:...	view	1306631	2053013558920217191	computers.notebook	hp	580.89	550050854	7c90fc70-0e80-459...
2019-10-01 00:00:...	view	1005135	2053013555631882655	electronics.smart...	apple	1747.79	535871217	c6bd7419-2748-4c5...

Рисунок 14 – Фрагмент датафрейма с исходными данными

	event_type	product_id	brand	price	contains_appliances	contains_computers	contains_electronics	contains_kitchen	contains_smartphone	is_expensive	is_budget	is_mid_range	category_count	is_purchase	is_view
0	cart	1002042	samsung	77.139999	False	False	True	False	True	0	0	1	2	0	0
1	cart	1002524	apple	513.450012	False	False	True	False	True	1	0	0	2	0	0
2	cart	1002524	apple	513.469971	False	False	True	False	True	1	0	0	2	0	0
3	cart	1002524	apple	531.409973	False	False	True	False	True	1	0	0	2	0	0
4	cart	1002524	apple	533.260010	False	False	True	False	True	1	0	0	2	0	0
5	cart	1002524	apple	540.270020	False	False	True	False	True	1	0	0	2	0	0
6	cart	1002536	apple	576.570007	False	False	True	False	True	1	0	0	2	0	0
7	cart	1002542	apple	488.790009	False	False	True	False	True	1	0	0	2	0	0
8	cart	1002544	apple	460.070007	False	False	True	False	True	1	0	0	2	0	0
9	cart	1002544	apple	460.309998	False	False	True	False	True	1	0	0	2	0	0

Рисунок 15 – Фрагмент датафрейма с обработанными данными

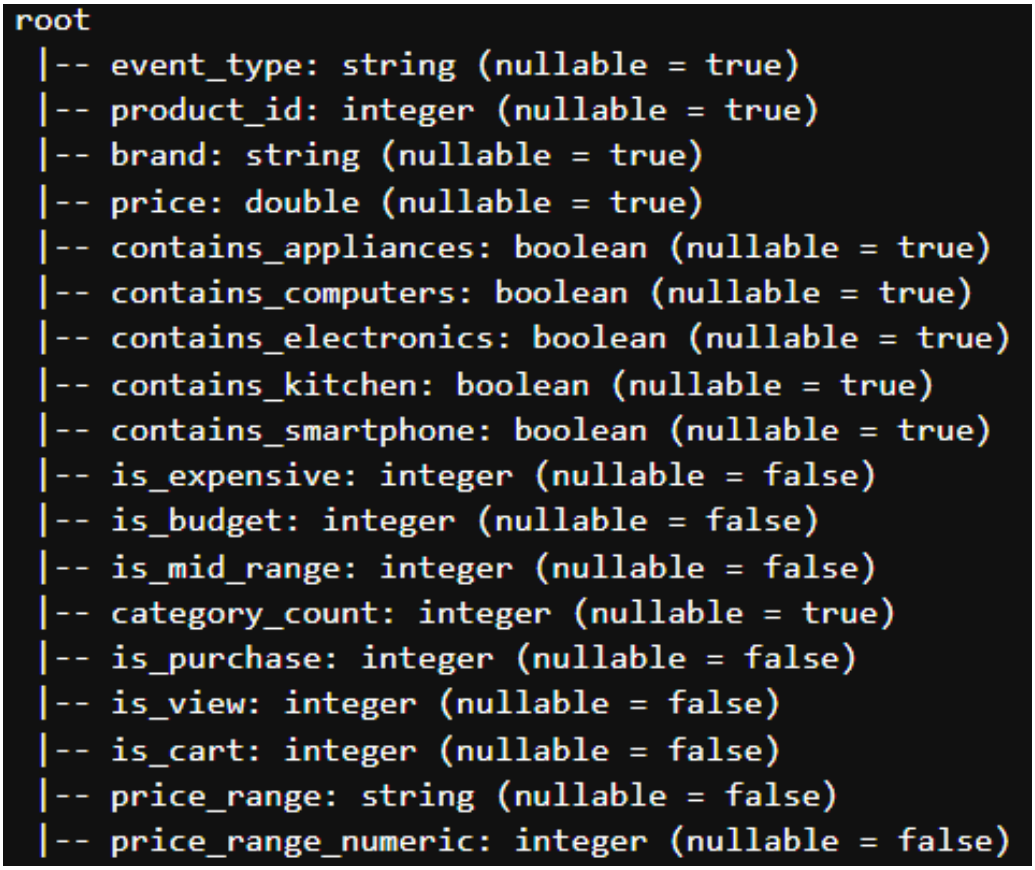


Рисунок 16 – Схема данных

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 (см. рис. 19).

```
spark.read.parquet("hdfs://namenode:9000/user/dchel/dchel_datab
ase/eCommerce_clear_data")
```

Далее выполнялась предобработка признаков. Категориальные параметры преобразовывались с помощью StringIndexer. Все признаки объединялись в единый вектор посредством VectorAssembler.

На основе этих этапов формировался конвейер Spark ML, включающий индексацию, кодирование, масштабирование признаков и модель GBT регрессии. Для неё использовались параметры maxIter=100, maxDepth=5, regParam=0.01.

Оптимизация модели выполнялась с помощью 2-кратной кросс-валидации. Наилучшие результаты показала конфигурация с regParam=0.1, maxIter=100 и maxDepth=5 (см. рис. 17):

```
CrossValidator(estimator=pipeline,
               estimatorParamMaps=param_grid,
               evaluator=cv_evaluator,
               numFolds=2,
               parallelism=4)
```

```
Параметры лучшей модели GBTRegressor:
maxDepth: 5
maxIter: 100
stepSize: 0.1
```

Рисунок 17 – Конфигурация лучшей GBT модели

2.1.3 Анализ полученных результатов регрессии

Модель была протестирована на отложенной тестовой выборке. Получены следующие значения метрик (см. рис. 18):

– RMSE = 65.9345;

– R^2 = 0.7862.

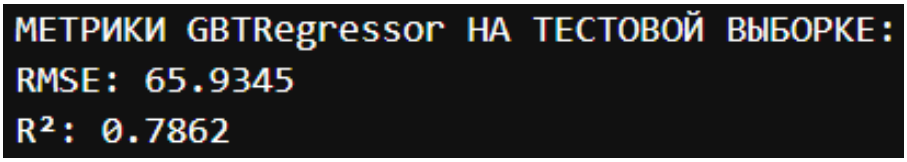


Рисунок 18 – RMSE и R^2 метрики

Высокое значение R^2 свидетельствует о сильной объясняющей способности модели.

2.2 Задача классификации с использованием LogisticRegression

2.2.1 Постановка задачи классификации

Вторая часть работы посвящена построению модели многоклассовой классификации, определяющей бюджетный класс (price_range_numeric). Целевой переменной является числовой индикатор бюджетной категории.

Требуется построить классификатор на основе Logistic Regression, предсказывающий бюджетный класс 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 (Recall ≥ 0.60), обеспечивает при этом наивысшую возможную долю верных предсказаний среди всех отмеченных как полезные (Precision).

2.2.2 Решение задачи классификации

В задаче классификации данные также загружались из HDFS. Потом производилось разделение тестовую и обучающую выборки (19).

```
# Обновим датафреймы train и test с новыми признаками
train_df, test_df = df.randomSplit([0.8, 0.2], seed=42)
print(train_df.count())
print(test_df.count())
```

```
151676
37625
```

Рисунок 19 – Объём выборок и кол-во экземпляров классов

На этапе предобработки категориальные признаки преобразовывались с помощью StringIndexer, после чего все признаки объединялись в общий вектор, необходимый для обучения модели. Нормализация числовых признаков проводилась с помощью StandardScaler.

Процесс обучения реализовывался через конвейер, включающий этапы подготовки данных и модель LogisticRegression. Для начального варианта использовались параметры maxIter=100, regParam=0.01, elasticNetParam=0.0 и family=”multinomial”.

Оптимизация гиперпараметров выполнялась с использованием 2-кратной кросс-валидации. Наилучшие результаты были достигнуты при maxIter=100, regParam=0.01, elasticNetParam=1.0 (см. рис. 20):

```
param_grid = ParamGridBuilder() \
    .addGrid(lr_model.regParam, [0.01, 0.1, 1.0]) \
    .addGrid(lr_model.elasticNetParam, [0.0, 0.5, 1.0]) \
    .addGrid(lr_model.maxIter, [10, 100]) \
    .build()
CrossValidator(estimator=pipeline,
               estimatorParamMaps=param_grid,
               evaluator=cv_evaluator,
               numFolds=2,
               parallelism=4)
```

```
Параметры лучшей модели LogisticRegressor:
regParam: 0.01
elasticNetParam: 1.0
maxIter: 100
tol: 1e-06
fitIntercept: True
family: multinomial
```

Рисунок 20 – Конфигурация лучшей LR модели

2.2.3 Анализ полученных результатов классификации

Модель классификации продемонстрировала стабильные результаты: точность (Precision) составила 0.97, полнота (Recall) — 0.968, F1-мера — 0.967, а общая точность классификации достигла 0.977 (см. рис. 21).

```
Test Metrics: {'accuracy': 0.9677376491535784, 'weightedPrecision': 0.9691029432450163, 'weightedRecall': 0.9677376491535784, 'f1':
0.9674194453559453}
```

Рисунок 21 – Метрики LR модели

Матрица ошибок для выбранного порога показывает следующие значения: более 12 000 объектов были корректно классифицированы, около 2 000 — некорректно (см. рис. 22 и Приложение В).

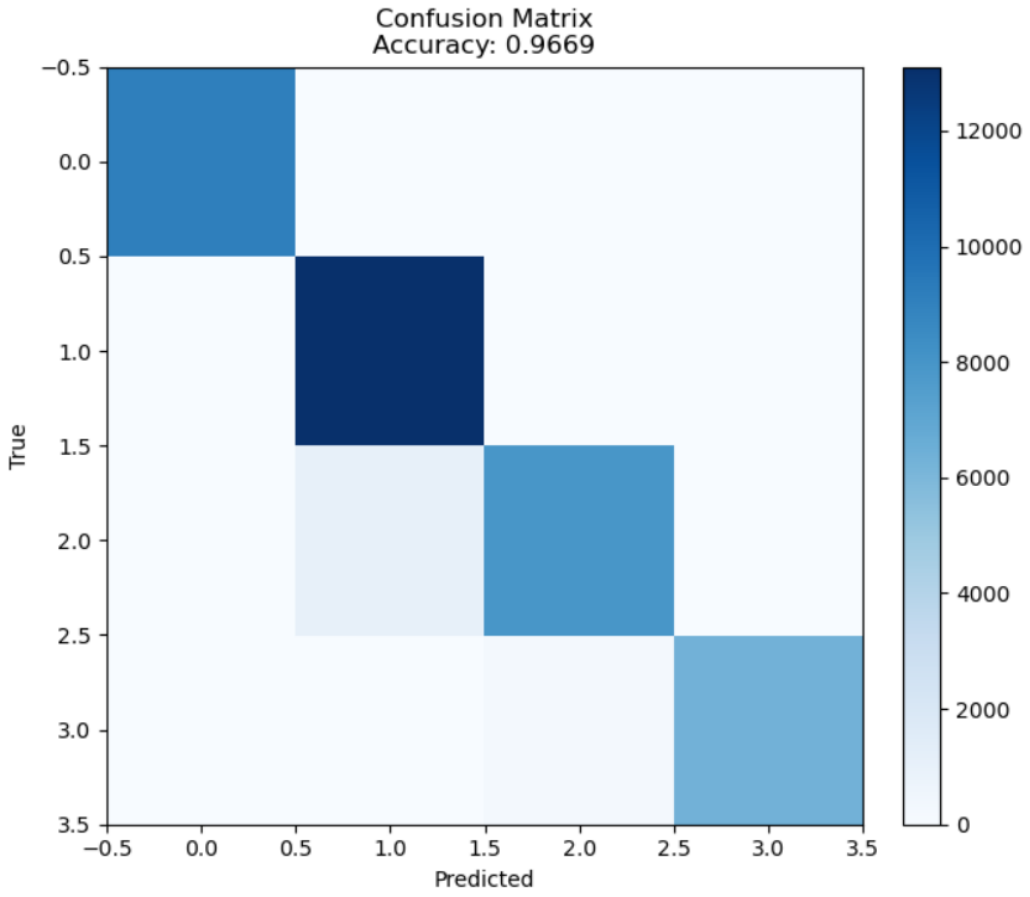


Рисунок 22 – Матрица ошибок

График распределения кол-ва верно предсказанных классификаций показывает следующее (см. Приложение Г):

Рисунок 23 – Кол-во верно предсказанных классификаций

2.3 Выводы

В рамках работы были решены две задачи машинного обучения. GBT регрессия показала эффективность: значение R^2 (0.78) подтверждает, что признаков достаточно для точного прогнозирования цены продуктов. Задача классификации оказалась более успешной: модель достигла accurancy = 0.96, F1 = 0.96 и выполнила требуемый уровень полноты (Recall ≥ 60%).

В перспективе дальнейшее развитие связано с использованием текстовых признаков [10] (TF-IDF, Word2Vec, BERT), внедрением нейронных сетей и современных ансамблей, а также сокращением размерности категориальных признаков. Полученные результаты демонстрируют эффективность Spark ML при анализе продуктов электронной коммерции в условиях больших данных.

ЗАКЛЮЧЕНИЕ

В ходе выполнения курсовой работы был реализован полный цикл анализа больших данных и машинного обучения на примере датасета «eCommerce behavior data from multi category store» с использованием фреймворка Apache Spark. В рамках исследования проведён разведочный анализ данных, включающий загрузку, очистку и преобразование информации источника. Были обработаны пропущенные значения, созданы новые признаки, проанализированы распределения и выбросы, что обеспечило подготовку качественного набора данных для последующего моделирования.

Для решения поставленных задач были реализованы и протестированы две модели машинного обучения. GBT регрессия для прогнозирования цены товара показала устойчивую сходимость, высокое значение $R^2 = 0{,}78$. Модель логистической регрессии для классификации бюджетного класса продуктов продемонстрировала хорошее качество (accurancy = 0,96) и выполнила поставленное условие: полнота (Recall) не ниже 60% при точности (Precision) 68%.

Сравнительный анализ задач показал, что бинарные признаки, такие как is_expensive или is_budget, эффективнее используются для прогнозирования цены на продукт. 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:

- расширение анализируемых пользовательских действий для построения полного воронки конверсии и расчета CTR (Click-Through Rate);
- обогащение данных профилями пользователей с применением RFM-анализа (Recency, Frequency, Monetary) и сегментацией по предпочтениям;
- внедрение временных и сезонных признаков для учета суточных, недельных и праздничных паттернов активности;

– разработку рекомендательных систем на основе контентной фильтрации (content-based) и коллаборативной фильтрации (collaborative filtering);
– анализ путей пользователей (Customer Journey) с применением марковских цепей для моделирования переходов между категориями;
– реализацию динамического ценообразования на основе анализа эластичности спроса и конкурентной среды;
– восстановление и структурирование иерархии категорий товаров для кросс-категорийного анализа;
– прогнозирование спроса с использованием моделей временных рядов (ARIMA/SARIMA, Prophet) и нейронных сетей (LSTM);
– проведение A/B-тестов для оптимизации алгоритмов рекомендаций и персонализации интерфейса;
– детекцию аномалий и мошеннической активности через анализ паттернов поведения пользователей.

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 (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. *Уайт, Т.* Hadoop: Подробное руководство. — 3-е изд. — Питер, 2013. — 672 с. — ISBN 978-5-496-00299-9.
8. *Kaggle*. eCommerce behavior data from multi category store. — 2019. — URL: https://www.kaggle.com/datasets/mkechinov/ecommerce-behavior-data-from-multi-category-store (visited on 12/16/2025).
9. *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.

10. Lakehouse: A New Generation of Open Platforms that Unify Data Warehousing and Advanced Analytics / Zaharia, Matei [et al.] // CIDR 2021. — 2021. — URL: https://www.cidrdb.org/cidr2021/papers/cidr2021_paper17.pdf (visited on 09/12/2024).

ПРИЛОЖЕНИЕ А

Исходный код обработки данных

```
def transform_dataframe(data: DataFrame) -> DataFrame:
    df = df.select(
        "event_type", "product_id", "category_id",
        "category_code", "brand", "price"
    )
    data = data.withColumn("category_id", col("category_id")
        .cast("Integer"))
    data = data.withColumn("product_id", col("product_id")
        .cast("Integer"))
    data = data.withColumn("price", col("price")
        .cast("Float"))
    # Преобразуем строку в массив строк
    data = data.withColumn("category_code",
                           split(col("category_code"), r"\.")
                          )
    data = data.dropna(subset=["category_code", "brand"])
    df = df.withColumn("is_expensive", when(col("price") > 200
        , 1).otherwise(0))
    df = df.withColumn("is_budget", when(col("price") < 50, 1)
        .otherwise(0))
    df = df.withColumn("is_mid_range", when((col("price")
        >= 50) & (col("price") <= 200), 1).otherwise(0))
    df = df.withColumn("category_count",
                   col("contains_appliances").cast("int") +
                   col("contains_computers").cast("int") +
                   col("contains_electronics").cast("int") +
                   col("contains_kitchen").cast("int") +
                   col("contains_smartphone").cast("int"))
    df = df.withColumn("is_purchase", when(col("event_type")
        == "purchase", 1).otherwise(0))
    df = df.withColumn("is_view", when(col("event_type")
```

```
    == "view", 1).otherwise(0))
df = df.withColumn("is_cart", when(col("event_type")
    == "cart", 1).otherwise(0))
df = df.withColumn("price_range",
                when(col("price") < 50, "budget")
                .when(col("price") < 150, "affordable")
                .when(col("price") < 300, "premium")
                .otherwise("luxury"))

# Создаем числовое представление для price_range
df = df.withColumn("price_range_numeric",
                 when(col("price_range") == "budget", 1)
                 .when(col("price_range") == "affordable"
                     , 2)
                 .when(col("price_range") == "premium", 3)
                 .otherwise(4))
return df
```

ПРИЛОЖЕНИЕ Б

Исходный код гистограммы распределения

```
def plot_quant_distribution(data: DataFrame,
                            column: str,
                            num_bins: int = 200) -> None:
    try:
        # Получаем min и max отдельными запросами
        min_value = data.selectExpr(f"min({column})")
            .collect()[0][0]
        max_value = data.selectExpr(f"max({column})")
            .collect()[0][0]

        print(f" Диапазон значений {column}:
            {min_value:.2f} - {max_value:.2f}")

        bin_size = (max_value - min_value) / num_bins

        data_with_bin = data.selectExpr(
            "*",
            f"floor(({column} - {min_value}) / {bin_size})
                as bin"
        ).filter(f"bin < {num_bins}")

        bin_counts = data_with_bin.groupBy("bin").count()
            .orderBy("bin")

        bin_counts_list = bin_counts.collect()

        bin_indices = []
        bin_values = []

        for row in bin_counts_list:
            bin_indices.append(row['bin'])
```

```
            bin_values.append(row['count'])

        bin_centers = [min_value + (bin_idx + 0.5) * bin_size
            for bin_idx in bin_indices]

        plt.figure(figsize=(20, 6))
        plt.bar(bin_centers, bin_values, width=bin_size * 0.8,
                alpha=0.7, color='skyblue', edgecolor='navy',
                linewidth=0.5)
        plt.xlabel("Value", fontsize=20)
        plt.ylabel("Count", fontsize=20)
        plt.title(f"Распределение количественного
            признака \"{column}\"")
        plt.grid(True, alpha=0.3)

        plt.gca().yaxis.set_major_formatter(
            plt.FuncFormatter(lambda x, p: f'{x:,.0f}'))

        plt.tight_layout()
        plt.show()

    except Exception as e:
        print(f"□ Ошибка при построении гистограммы: {e}")
        import traceback
        traceback.print_exc()
```

ПРИЛОЖЕНИЕ В

Исходный код матрицы ошибок

```
def plot_lr_training_metrics(cv_model, test_data,
    label_col="label"):
    test_predictions = cv_model.transform(test_data)

    evaluator_accuracy = MulticlassClassificationEvaluator(
        labelCol=label_col,
        predictionCol="prediction",
        metricName="accuracy"
    )

    evaluator_f1 = MulticlassClassificationEvaluator(
        labelCol=label_col,
        predictionCol="prediction",
        metricName="f1"
    )

    test_accuracy = evaluator_accuracy.evaluate(test_predictions)
    test_f1 = evaluator_f1.evaluate(test_predictions)

    print(f"Accuracy: {test_accuracy:.4f}")
    print(f"F1-Score: {test_f1:.4f}")

    from sklearn.metrics import confusion_matrix,
        ConfusionMatrixDisplay

    y_true = test_predictions.select(label_col)
        .rdd.flatMap(lambda x: x).collect()
    y_pred = test_predictions.select("prediction")
        .rdd.flatMap(lambda x: x).collect()

    cm = confusion_matrix(y_true, y_pred)
```

```
fig, (ax1, ax2) = plt.subplots(1, 2, figsize=(15, 6))

im = ax1.imshow(cm, cmap='Blues')
plt.colorbar(im, ax=ax1, fraction=0.046, pad=0.04)
ax1.set_title(f'Матрица ошибок: {test_accuracy:.4f}')
ax1.set_xlabel('Predicted')
ax1.set_ylabel('True')

ax2.bar(range(len(cm)), [cm[i, i] for i in range(len(cm))])
ax2.set_title('Правильные предсказания по классам')
ax2.set_xlabel('Класс')
ax2.set_ylabel('Количество')
ax2.grid(axis='y', alpha=0.3)

plt.tight_layout()
plt.show()

return test_predictions
```

ПРИЛОЖЕНИЕ Г

Исходный код распределения категориального признака

```
def plot_cat_distribution(data,
                          column_name: str,
                          top_n: int = 20) -> None:
    column_type = dict(data.dtypes)[column_name]

    if column_type == 'array<string>':
        categories = (
            data
            .select(explode(col(column_name))
            .alias(column_name))
            .groupBy(column_name)
            .count()
            .orderBy("count", ascending=False)
        )
    else:
        categories = (
            data
            .groupBy(column_name)
            .count()
            .orderBy("count", ascending=False)
        )

    total_categories = categories.count()
    print(f"Количество категорий признака {column_name}:
        {total_categories}")

    categories_list = categories.limit(top_n).collect()

    category_names = []
    category_counts = []
```

```
for row in categories_list:
    category_names.append(row[column_name])
    category_counts.append(row['count'])

plt.figure(figsize=(12, 8))
plt.bar(category_names, category_counts)
plt.title(f"Barplot of \"{column_name}\"
    counts (Top {top_n})")
plt.xlabel(column_name)
plt.ylabel("Count")
plt.xticks(rotation=45, ha='right')
plt.tight_layout()
plt.show()

if total_categories > top_n:
    print(f"Показаны топ-{top_n}
        категорий из {total_categories}")
```