УДК 004

Прогнозирование стоимости смартфонов с ОС Android

Матвеева Алёна Сергеевна
Приамурский государственный университет имени Шолом-Алейхема
Студент

Аннотация
Целью исследования является применение методов искусственного интеллекта для прогнозирования стоимости смартфонов с операционной системой Android. Для реализации использовалась свободно распространяемая платформа GoogleColab. Практическая значимость данной работы заключается в возможности применения разработанной модели прогнозирования стоимости смартфонов с операционной системой Android для оптимизации ценообразования, анализа рыночной конкуренции и принятия бизнес-решений в сфере продаж и маркетинга мобильных устройств.
Ключевыеслова: Python, датасет, модель.

Forecasting the cost of smartphones Android OS

MatveevaAlyonaSergeevna
Sholom-Aleichem Priamursky State University
Student

Abstract
The purpose of the study is to use artificial intelligence methods to predict the cost of smartphones with the Android operating system. The freely distributed GoogleColab platform was used for implementation. The practical significance of this work lies in the possibility of using the developed model of forecasting the cost of smartphones with the Android operating system to optimize pricing, analyze market competition and make business decisions in the field of sales and marketing of mobile devices.
Keywords: Python, dataset, model.

1 Введение

1.1 Актуальность

В современном мире рынок мобильных устройств, особенно смартфонов на базе операционной системы Android, продолжает демонстрировать динамичное развитие и значительное влияние на повседневную жизнь людей. Прогнозирование стоимости смартфонов становится все более важной задачей для бизнеса, потребителей и исследователей. С учетом быстрого темпа выпуска новых моделей, разнообразия функционала и конкуренции между производителями, точное

прогнозирование цен на смартфоны становится ключевым фактором успешной стратегии продаж и покупки.

Использование методов искусственного интеллекта, таких как линейная регрессия, случайный лес, XGBoost, дерево решений и метод ближайших соседей, для прогнозирования стоимости смартфонов с ОС Android представляет собой актуальную исследовательскую задачу, поскольку позволяет не только предсказать цены на основе имеющихся данных, но и выявить наиболее эффективные методы для данной конкретной задачи.

1.2Обзор исследований

Исследованиями в данной теме занимались следующие авторы. Б.В.Овчаренко, С.В. Сурмило в своей статье представили программную реализацию прогнозирования пожаров с использованием различных моделей, в том числе адаптивных моделей прогнозирования [1]. Реализовали возможность экспорта и импорта данных из пакета программ MicrosoftOffice.

Д. Кано, Ю. Накано провели исследование, в котором заключается повышение эффективности прогнозирования энергопотребления [2]. Модуль прогнозирования энергопотребления выполнен с возможностью прогнозировать энергопотребление запланированного маршрута движения на основе скорректированного значения сопротивления воздуха.

В.К. Сай, М.В. Щербаков рассмотрели прогнозирование отказов сложных много объектных систем на основе комбинации нейросетей: пути повышения точности прогнозирования [3]. В работе предложена гибридная нейросетевая модель с двумя выходами на основе сверточных нейронных сетей (convolutionalneuralnetwork, CNN) и сетей долгой краткосрочной памяти (longshort-termmemory; LSTM). Сети CNN используются для извлечения пространственных свойств из многомерных сенсорных данных, а сети LSTM - для темпорального моделирования долговременных зависимостей.

Ю.С. Попков изучил основные понятия и алгоритмы машинного обучения и его рандомизированной версии [4]. Последняя является эффективным инструментом решения задач классификации и прогнозирования в условиях неопределенности. Исследуются основные свойства процедур рандомизированного машинного обучения и приводятся примеры решения прикладных задач.

В статье Р.Г. Галимова рассматриваются основы алгоритмов машинного обучения, в частности алгоритмы обучения с учителем [5]. Рассмотрены алгоритмы линейной регрессии, логистической регрессии, деревьев решений. Для каждого из алгоритмов представлен программный код на языках R и Python.

В статье И. Полонников рассматриваются тенденции цифрового маркетинга и машинного обучения для обработки данных в современном бизнесе [6]. Автор определяет скорость анализа и принятия решений, основанных на данных этого анализа, как определяющий фактор рыночной

конкуренции. А также описывает методы анализа и интеллектуального моделирования методов машинного обучения и A/B тестирования как перспективные маркетинговые инструменты.

Е. Ульянихин рассматривал применение алгоритмов машинного обучения в области информационных технологий [7]. Определены важные аспекты эффективного машинного обучения. Мы анализируем, как современные информационные технологии связаны с концепцией искусственного интеллекта.

В статье В.А. Фадеев, Ш.В. Зайдуллин и А.Ф. Надеев рассматривается применимость некоторых традиционных моделей прогнозирования временных рядов для временной динамики частоты отказов в настройке радиопередатчика EPS [8]. Были рассмотрены две основные проблемы упреждающего управления сетью: прогнозирование регулярной части временных рядов и прогнозирование выбросов. Для прогнозирования регулярной части использовались экспоненциальное сглаживание Холта-Уинтерса, усиление экстремального градиента (XGBoost), регрессия опорных векторов (SVR), динамическая линейная модель Python (PyDLM) и сезонная авторегрессионная интегрированная скользящая средняя (SARIMAX).

Н. Ломакин, А. Кулачинская, С. Наумова, М. Ибрагим, Е. Федоровская и И. Ломакин изучили построение когнитивной модели, использовав язык Python в облачной среде Collab [9]. Средняя абсолютная ошибка тестового набора для ML-модели случайного леса (DecisionTreeRegressor) составила 414,67, что на 61% ниже, чем для модели линейной регрессии (LinearRegression), средняя абсолютная ошибка которой составила 667,65.

В работе Д. Парбат и М. Чакраборти используется регрессионная модель опорных векторов для прогнозирования общего числа смертей, выздоровевших случаев, совокупного числа подтвержденных случаев и количества ежедневных случаев [10]. Предлагаемая методология основана на прогнозировании значений с использованием регрессионной модели опорных векторов с радиальной базисной функцией в качестве ядра и 10% доверительным интервалом для подгонки кривой.

1.3Цель исследования

Целью данного исследования является применение методов искусственного интеллекта для прогнозирования стоимости смартфонов с операционной системой Android. Для достижения этой цели были поставлены следующие задачи:

1) Сбор и анализ данных о характеристиках смартфонов с ОС Android и их ценах.
2) Подготовка данных для применения методов искусственного интеллекта, включая предобработку и очистку данных.
3) Применение методов линейной регрессии, случайного леса, XGBoost, дерева решений и метода ближайших соседей для прогнозирования стоимости смартфонов.

4) Оценка точности прогнозов и выявление факторов, влияющих на стоимость смартфонов с ОС Android.

5) Сравнительный анализ эффективности каждого метода и выбор наиболее подходящими моделями для предсказания цены.

2 Материалы и методы

В данном исследовании используется платформа GoogleColab для написания кода на языке программировании Python.

В работе был собран датасет данных в файле «Датасет Смартфоны.xlsx», о характеристиках смартфонов с ОС Android и их стоимостью с сайта DNS г. Биробиджан [11]. Датасет содержит 200 позиций смартфонов с характеристиками: бренд, модель, размер экрана, камера, ОЗУ (объём оперативной памяти), ОВЗ (объём встроенной памяти), ядра, частота, NFC, ёмкость аккумулятора, год релиза, цена и ссылка на сайтDNS для каждого смартфона.

Методы исследования включают сбор и анализ данных о характеристиках смартфонов, применение методов линейной регрессии, случайного леса, XGBoost, дерева решений и метода ближайших соседей для прогнозирования стоимости смартфонов, а также сравнительный анализ эффективности каждого метода с целью выбора, наиболее подходящего для данной задачи.

3 Результаты

Для работы с данными нужно импортировать основные библиотеки [12]:

```
import pandas as pd
importnumpyas np
importmatplotlib.pyplotasplt
fromsklearn.ensembleimportRandomForestRegressor
fromsklearn.linear_modelimportLinearRegression
fromsklearn.preprocessingimportStandardScaler
fromsklearn.model_selectionimporttrain_test_split
fromsklearn.metricsimport r2_score, mean_squared_error,
mean_absolute_error
fromsklearn.treeimportDecisionTreeRegressor
fromsklearn.neighborsimportKNeighborsRegressor
fromsklearnimport tree
importxgboostasxgb
```

Подключаем файл «Датасет Смартфоны.xlsx» с данными и отображаем таблицу с первыми пятью позициями для просмотра (Рис.1).

```
df = pd.read_excel('Датасет Смартфоны.xlsx')
df.head()
```

	Бренд	Модель	Размер_экрана	Камера	ОЗУ	ОВЗ	Ядра	Частота	NFC	Ёмкость_аккумулятор	Год_релиза	Цена	ссылка
0	POCO	C50	6.52	8	2	64	8	2.2	нет	5000	2023	4999	https://www.dns-shop.ru/product/d4c6a71bfe93ed...
1	Xiaomi	A2	6.52	8	3	64	8	2.2	нет	5000	2023	5999	https://www.dns-shop.ru/product/a9e4f148f46eed...
2	Infinix	SMART 8	6.56	13	3	64	8	1.6	нет	5000	2023	5999	https://www.dns-shop.ru/product/1c9ab3fa8a5eed...
3	realme	Note 50	6.70	13	3	64	8	1.8	нет	5000	2024	5999	https://www.dns-shop.ru/product/96bcc288b34ded...
4	realme	Note 50	6.70	13	4	128	8	1.8	нет	5000	2024	7499	https://www.dns-shop.ru/product/10b61286b34eed...

Рисунок 1 - Датасет Смартфоны

Чтобы посмотреть типы данных каждого столбца, нужно прописать df.info() (Рис.2).

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 200 entries, 0 to 199
Data columns (total 13 columns):
 #   Column               Non-Null Count  Dtype
---  ------               --------------  -----
 0   Бренд                200 non-null    object
 1   Модель               200 non-null    object
 2   Размер_экрана        200 non-null    float64
 3   Камера               200 non-null    int64
 4   ОЗУ                  200 non-null    int64
 5   ОВЗ                  200 non-null    int64
 6   Ядра                 200 non-null    int64
 7   Частота              200 non-null    float64
 8   NFC                  200 non-null    object
 9   Ёмкость_аккумулятор  200 non-null    int64
 10  Год_релиза           200 non-null    int64
 11  Цена                 200 non-null    int64
 12  ссылка               200 non-null    object
dtypes: float64(2), int64(7), object(4)
memory usage: 20.4+ KB
```

Рисунок 2 – Типы данных

Так как для исследования столбец «ссылка» не нужны, удаляем его из таблицы:

```
df = df.drop('ссылка', axis=1)
```

Изменим тип данных столбцу «NFC» на категориальные переменные:

```
df = df.astype({'NFC' : 'category' })
df.NFC = df.NFC.cat.rename_categories(new_categories =
                                      {'есть': '1', 'нет': '0'})
```

Посмотрим на уникальные значения из столбца «Бренд»:

```
df.Бренд.unique()
```

Вывод результата:

```
array(['POCO', 'Xiaomi', 'Infinix ', 'realme', 'HUAWEI', 'Tecno', 'ZTE',
       'Itel', 'HONOR', 'Samsung', 'Google Pixel'], dtype=object)
```

Отсортируем категории столбца «Бренд» по количеству в убывающем порядке с помощью команды df.Бренд.value_counts() и получим результаты (Рис.3).

```
Бренд
Samsung         39
Xiaomi          36
POCO            25
Tecno           23
HONOR           19
realme          16
Infinix         14
HUAWEI          14
Google Pixel    11
Itel             2
ZTE              1
Name: count, dtype: int64
```

Рисунок 3 – Уникальные значения из столбца «Бренд»

Код np.unique(df.Бренд, return_counts=True) возвращает уникальные значения из столбца «Бренд» в DataFramedf и их частоту встречаемости (Рис.4).

```
(array(['Google Pixel', 'HONOR', 'HUAWEI', 'Infinix ', 'Itel', 'POCO',
        'Samsung', 'Tecno', 'Xiaomi', 'ZTE', 'realme'], dtype=object),
 array([11, 19, 14, 14,  2, 25, 39, 23, 36,  1, 16]))
```

Рисунок 4 – уникальные значения из столбца "Бренд" и их частота

Код df.Бренд.value_counts().count() возвращает количество уникальных значений в столбце «Бренд» в DataFramedf. Вывод результата: 11.

Код np.unique(df.Камера, return_counts=True) возвращает уникальные значения из столбца «Камера» в DataFramedf и их частоту встречаемости. Код df.Камера.value_counts().count() возвращает количество уникальных значений в столбце «Камера» в DataFramedf (Рис.5).

```
[ ] np.unique(df.Камера, return_counts = True)

(array([  8,  12,  13,  16,  48,  50,  54,  64, 100, 108, 180, 200]),
 array([ 2,  1, 10,  3, 12, 93,  1, 27,  3, 33,  2, 13]))

[ ] # посмотрим на общее количество уникальных категорий
    df.Камера.value_counts().count()

12
```

Рисунок 5 – Уникальные значения из столбца «Камера» и их частота

Код np.unique(df.Ядра, return_counts=True) возвращает уникальные значения из столбца «Ядра» в DataFramedf и их частоту встречаемости. Код df.Ядра.value_counts().count() возвращает количество уникальных значений в столбце «Ядра» в DataFramedf (Рис.6).

```
[ ] np.unique(df.Ядра, return_counts = True)

(array([ 8,  9, 10]), array([189,   9,   2]))

[ ] # посмотрим на общее количество уникальных категорий
    df.Ядра.value_counts().count()

3
```

Рисунок 6 – Уникальные значения из столбца «Ядра» и их частота

Код np.unique(df.ОЗУ, return_counts=True) возвращает уникальные значения из столбца «ОЗУ» в DataFramedf и их частоту встречаемости. Код df.ОЗУ.value_counts().count() возвращает количество уникальных значений в столбце «ОЗУ» в DataFramedf (Рис.7).

```
[ ] np.unique(df.ОЗУ, return_counts = True)

(array([ 2,  3,  4,  6,  8, 12, 16]), array([ 1,  6, 29, 25, 97, 39,  3]))

[ ] # посмотрим на общее количество уникальных категорий
    df.ОЗУ.value_counts().count()

7
```

Рисунок 7 – Уникальные значения из столбца «ОЗУ» и их частота

Код np.unique(df.ОВЗ, return_counts=True) возвращает уникальные значения из столбца «ОВЗ» в DataFramedf и их частоту встречаемости. Код df.ОВЗ.value_counts().count() возвращает количество уникальных значений в столбце «ОВЗ» в DataFramedf (Рис.8).

```
[ ] np.unique(df.ОВЗ, return_counts = True)

(array([  32,   64,  128,  256,  512, 1024]), array([ 1, 15, 77, 83, 22,  2]))

[ ] # посмотрим на общее количество уникальных категорий
    df.ОВЗ.value_counts().count()

6
```

Рисунок 8 – Уникальные значения из столбца «ОВЗ» и их частота

Преобразования столбцов «Бренд», «Камера», «Ядра», «ОЗУ», «ОВЗ» в тип данных – категория и общая информации о DataFramedf (Рис.9):

```
    df = df.astype({'Бренд' : 'category', 'Камера' : 'category', 'Ядра'
: 'category', 'ОЗУ' : 'category', 'ОВЗ' : 'category' })
    df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 200 entries, 0 to 199
Data columns (total 12 columns):
 #   Column               Non-Null Count  Dtype
---  ------               --------------  -----
 0   Бренд                200 non-null    category
 1   Модель               200 non-null    object
 2   Размер_экрана        200 non-null    float64
 3   Камера               200 non-null    category
 4   ОЗУ                  200 non-null    category
 5   ОВЗ                  200 non-null    category
 6   Ядра                 200 non-null    category
 7   Частота              200 non-null    float64
 8   NFC                  200 non-null    category
 9   Ёмкость_аккумулятор  200 non-null    int64
 10  Год_релиза           200 non-null    int64
 11  Цена                 200 non-null    int64
dtypes: category(6), float64(2), int64(3), object(1)
memory usage: 12.2+ KB
```

Рисунок 9 – Изменённые типы данных

Были построены диаграммы для анализа важности данных. Для начала была рассмотрена столбчатая диаграмма, как цена зависит от бренда смартфонов (Рис.10):

```
df.plot(x='Бренд', y='Цена', kind='bar', figsize=(25, 6))
plt.title('Цены по брендам')
plt.xlabel('Бренд')
plt.ylabel('Цена')
plt.show()
```

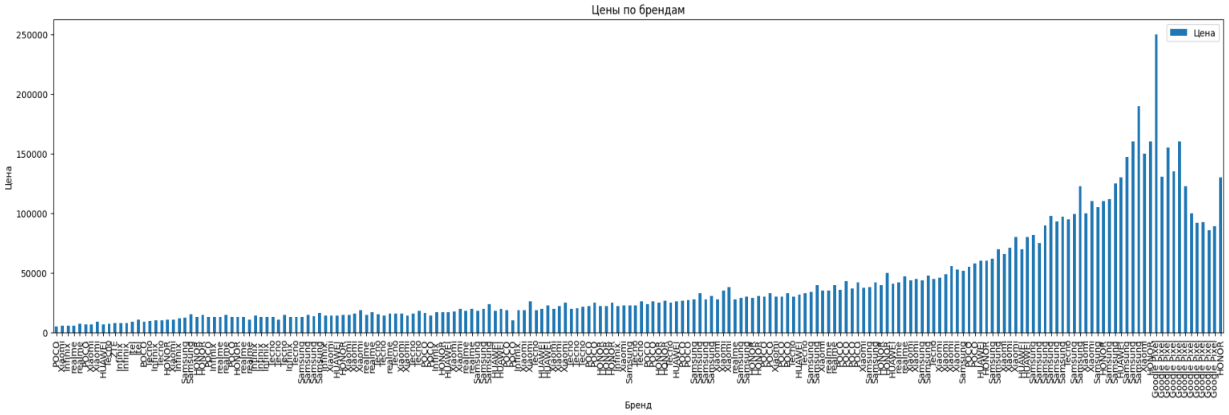

Рисунок 10 – Столбчатая диаграмма «Цены по брендам»

По даннойдиаграмме можно сделать вывод о том, как цены на смартфоны с ОС Android распределяются в зависимости от их бренда. Из диаграммы видно, что некоторые бренды имеют более высокие цены, в то время как другие предлагают устройства по более низким ценам.

Так же посмотрим на диаграммы категориальных переменных «Камера», «ОЗУ», «ОВЗ» и «Ядра» (Рис.11):

```
fig, axs = plt.subplots(2, 2, figsize=(15, 10))
df['Камера'].value_counts().plot(kind='bar', ax=axs[0, 0])
axs[0, 0].set_title('Камера')
```

```
    df['ОЗУ'].value_counts().plot(kind='bar', ax=axs[0, 1])
    axs[0, 1].set_title('ОЗУ')
    df['ОВЗ'].value_counts().plot(kind='bar', ax=axs[1, 0])
    axs[1, 0].set_title('ОВЗ')
    df['Ядра'].value_counts().plot(kind='bar', ax=axs[1, 1])
    axs[1, 1].set_title('Ядра')
    plt.tight_layout()
    plt.show()
```

Предоставленный код строит 4 столбчатые диаграммы для анализа важности данных о камере, объеме оперативной памяти (ОЗУ), объеме встроенной памяти (ОВЗ) и количестве ядер процессора.

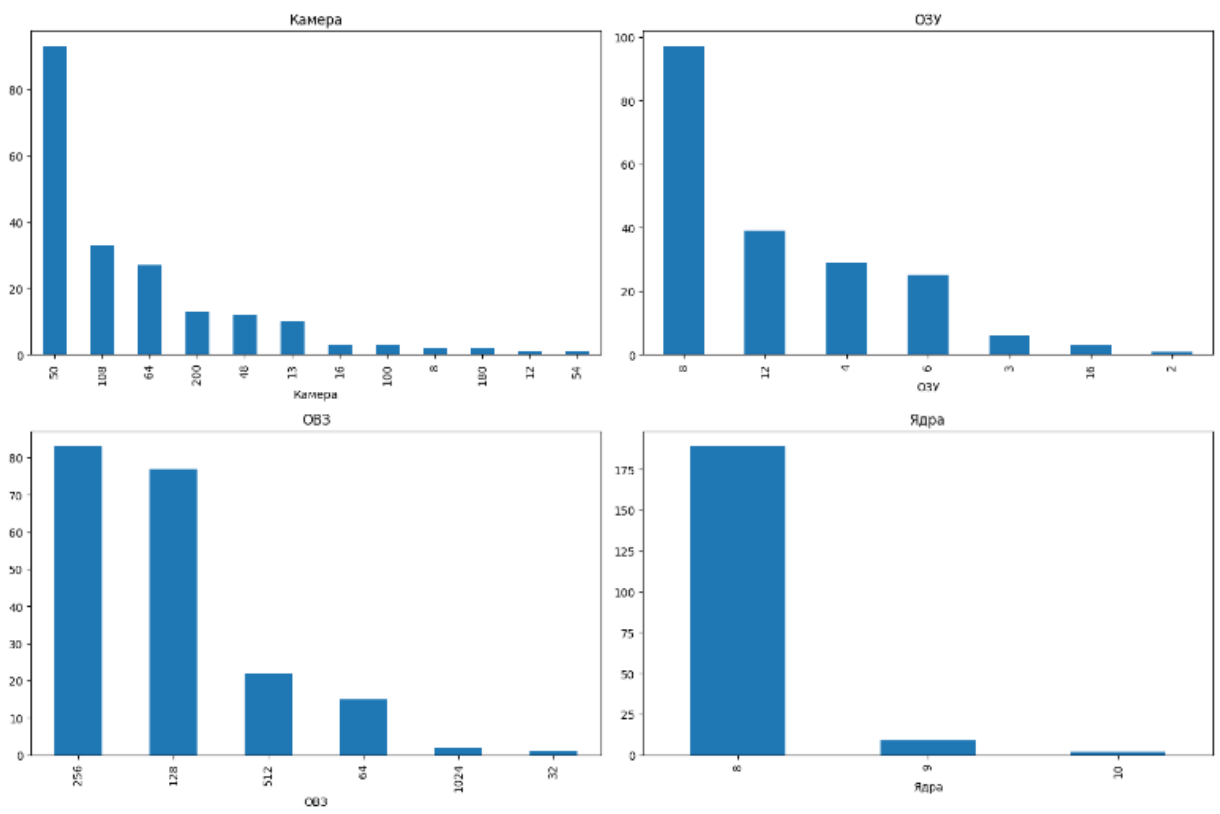

Рисунок 11 – Диаграммы по категориальным переменным

Код создает 4диаграммы рассеивания для анализа важности данных относительно цены смартфонов. Каждый график показывает взаимосвязь между определенной характеристикой (размер экрана, частота процессора, ёмкость аккумулятора, год релиза) и ценой смартфона, при этом используются разные цвета для обозначения года релиза (Рис.12):

```
    fig, axs = plt.subplots(2, 2, figsize=(15, 10))
    # Рассеивание с разными цветами и легендой
    scatter = axs[0, 0].scatter(df['Размер_экрана'], df['Цена'],
c=df['Год_релиза'], cmap='viridis')
    axs[0, 0].set_title('Размер_экрана vs Цена')
    scatter = axs[0, 1].scatter(df['Частота'], df['Цена'],
c=df['Год_релиза'], cmap='viridis')
    axs[0, 1].set_title('Частота vs Цена')
    scatter = axs[1, 0].scatter(df['Ёмкость_аккумулятор'], df['Цена'],
c=df['Год_релиза'], cmap='viridis')
    axs[1, 0].set_title('Ёмкость_аккумулятор vs Цена')
```

```
        scatter = axs[1, 1].scatter(df['Год_релиза'], df['Цена'],
c=df['Год_релиза'], cmap='viridis')
        axs[1, 1].set_title('Год_релиза vs Цена')
        # Добавлениелегенды
        fig.colorbar(scatter, ax=axs, orientation='vertical')
```

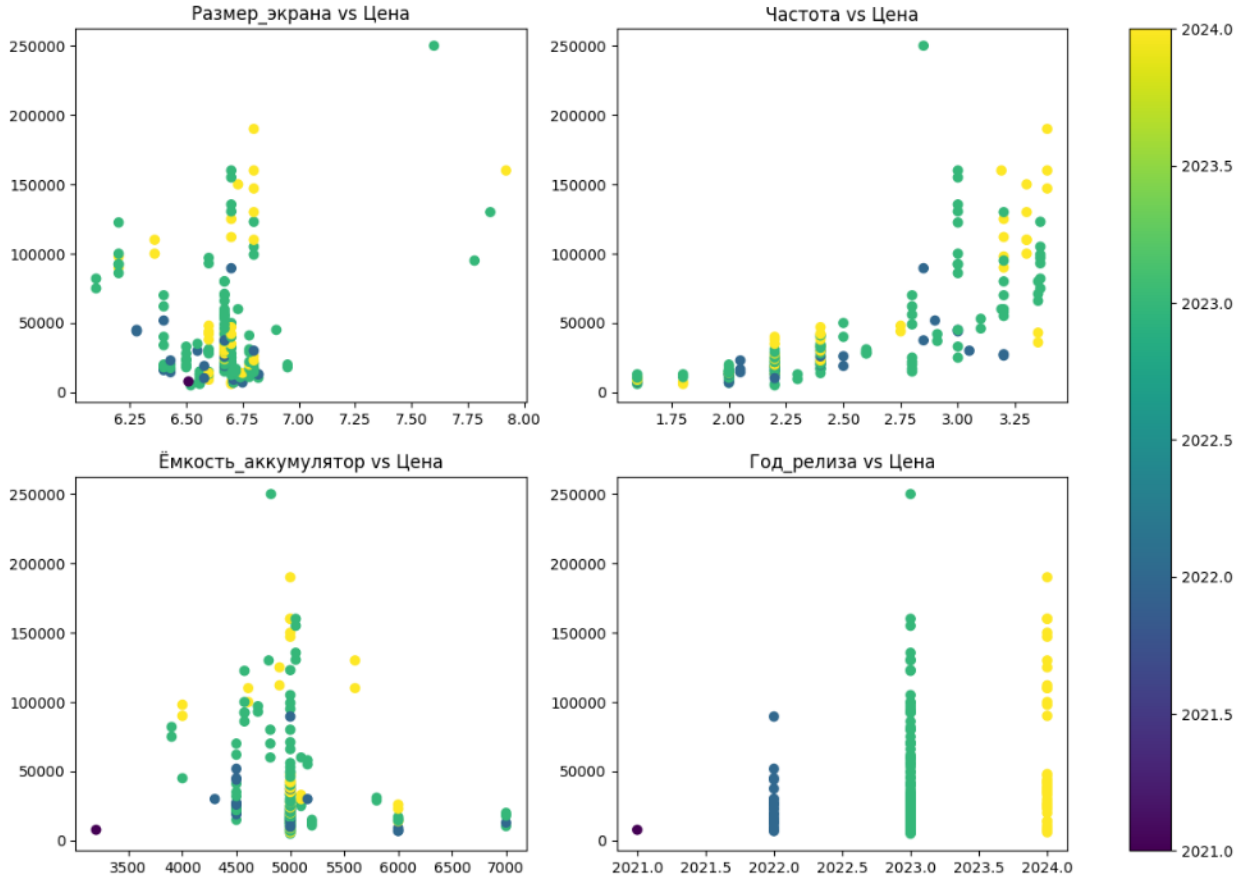

Рисунок 12 – Диаграммы рассеивания

Чтобы создать корреляционную матрицу нужно преобразовать категориальные данные в числовой формат (Рис.13):

```
        fromsklearn.preprocessingimportLabelEncoder
        # Кодирование категориальных переменных
        label_encoder = LabelEncoder()
        df['Бренд'] = label_encoder.fit_transform(df['Бренд'])
        df['Камера'] = label_encoder.fit_transform(df['Камера'])
        df['ОЗУ'] = label_encoder.fit_transform(df['ОЗУ'])
        df['ОВЗ'] = label_encoder.fit_transform(df['ОВЗ'])
        df['Ядра'] = label_encoder.fit_transform(df['Ядра'])
        df['NFC'] = label_encoder.fit_transform(df['NFC'])
        # Создание корреляционной матрицы
        correlation_matrix = df.corr()
        # Визуализация корреляционной матрицы
        plt.figure(figsize=(12, 8))
        sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm',
fmt='.2f')
        plt.title('Correlation Matrix')
        plt.show()
```

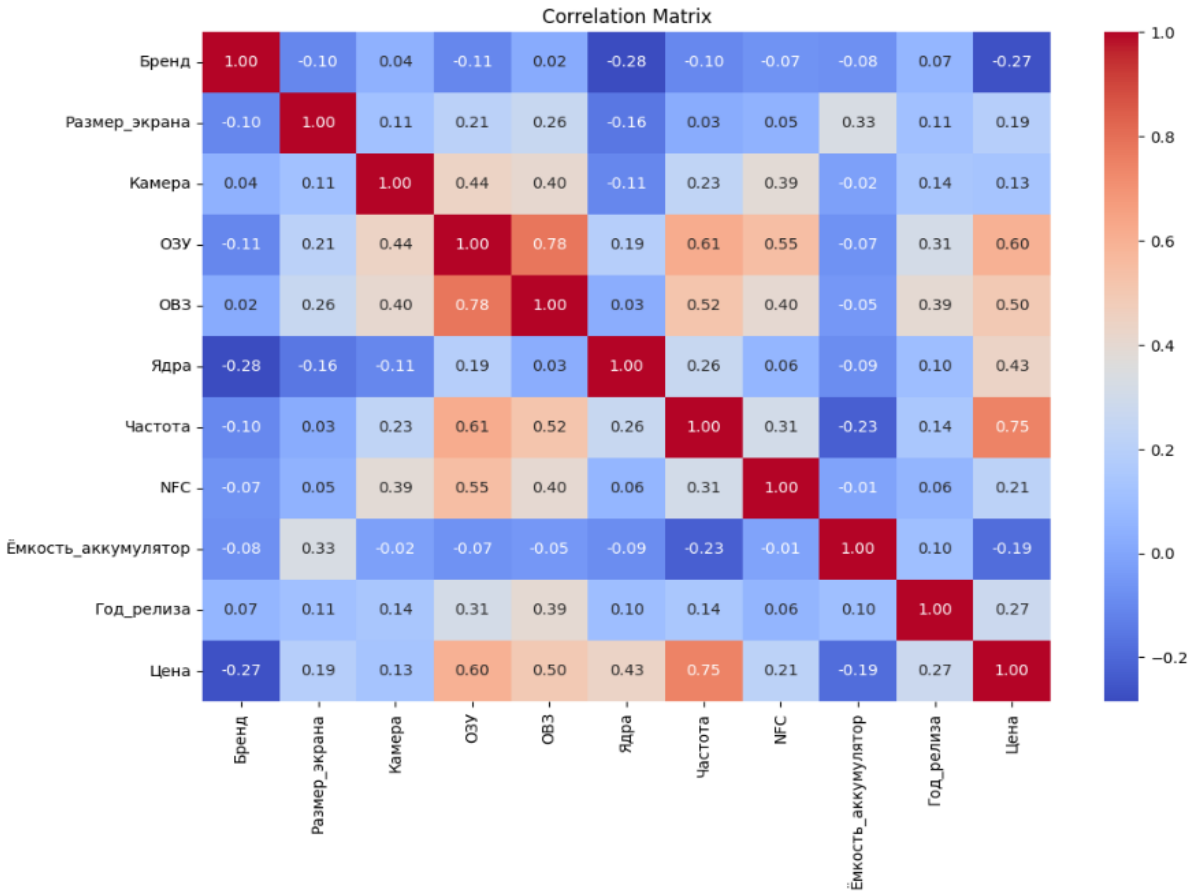

Рисунок 13 – Корреляционная матрица

Данные, представленные в корреляционной матрице, позволяют оценить степень взаимосвязи между различными переменными в наборе данных. В данном случае, коэффициент корреляции может принимать значения от -1 до 1, где 1 означает положительную линейную корреляцию, -1 - отрицательную линейную корреляцию, а 0 - отсутствие линейной корреляции.

Выводы по данным:

«Размер_экрана» имеет положительную корреляцию с различными характеристиками устройства, такими как ОЗУ, ОВЗ, частота процессора и цена. Это может указывать на то, что устройства с большим размером экрана чаще имеют более высокие технические характеристики и цену.

«ОЗУ» и «ОВЗ» также имеют сильную положительную корреляцию между собой, что логично, так как они оба относятся к памяти устройства.

«Частота» имеет высокую положительную корреляцию с ценой, что может указывать на то, что устройства с более высокой частотой процессора имеют более высокую цену.

Создаем модель «Линейная регрессия» для предсказания значения «Цена» на основе остальных данных:

```
fromsklearn.preprocessingimportOneHotEncoder
fromsklearn.composeimportmake_column_transformer
fromsklearn.pipelineimportmake_pipeline
# Определение признаков (X) и целевой переменной (y)
X = df.drop('Цена', axis=1)
```

```
    y = df['Цена']
    # Разделение данных на обучающий и тестовый наборы
    X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
    # Создание и обучение модели линейной регрессии с учетом категориальных переменных
    preprocessor = make_column_transformer((OneHotEncoder(), ['Бренд', 'Камера', 'ОЗУ', 'ОВЗ', 'Ядра', 'NFC']), remainder='passthrough')
    model = make_pipeline(preprocessor, LinearRegression())
    model.fit(X_train, y_train)
    # Предсказание цен на тестовом наборе
    y_pred = model.predict(X_test)
    # Оценка качества модели
    lr_r2 = r2_score(y_test, y_pred)
    lr_mse = mean_squared_error(y_test, y_pred)
    lr_rmse = mean_squared_error(y_test, y_pred, squared=False)
    lr_mae = mean_absolute_error(y_test, y_pred)
    print('Коэффициент детерминации (R^2):', lr_r2)
    print('Среднеквадратичная ошибка (MSE):', lr_mse)
    print('Корень среднеквадратичной ошибки (RMSE):', lr_rmse)
    print('Средняя абсолютная ошибка (MAE):', lr_mae)
```

Вывод результатов:

```
Коэффициент детерминации (R^2): 0.5703767738563323
Среднеквадратичная ошибка (MSE): 514728150.2569191
Корень среднеквадратичной ошибки (RMSE): 22687.621079719203
Средняя абсолютная ошибка (MAE): 15876.658933882369
```

Создание и обучение модели «Случайный лес» с учетом категориальных переменных:

```
    categorical_cols = [col for col inX.columnsif X[col].dtype == 'category']
    preprocessor = make_column_transformer((OneHotEncoder(), categorical_cols), remainder='passthrough')
    f_model = make_pipeline(preprocessor, RandomForestRegressor())
    f_model.fit(X_train, y_train)
    # Предсказание цен на тестовом наборе
    y_pred = f_model.predict(X_test)
    # Оценка качества модели
    f_r2 = r2_score(y_test, y_pred)
    f_mse = mean_squared_error(y_test, y_pred)
    f_rmse = mean_squared_error(y_test, y_pred, squared=False)
    f_mae = mean_absolute_error(y_test, y_pred)
    print('Коэффициент детерминации (R^2):', f_r2)
    print('Среднеквадратичная ошибка (MSE):', f_mse)
    print('Корень среднеквадратичной ошибки (RMSE):', f_rmse)
    print('Средняя абсолютная ошибка (MAE):', f_mae)
```

Вывод результатов:

```
Коэффициент детерминации (R^2): 0.8464332981010898
Среднеквадратичная ошибка (MSE): 183987037.01146913
Корень среднеквадратичной ошибки (RMSE): 13564.182135737825
Средняя абсолютная ошибка (MAE): 8718.298849206349
```

Создание и обучение модели «XGBoost» с учетом категориальных переменных:

```
    fromxgboostimportXGBRegressor
```

```
    categorical_cols = [col for col inX.columnsif X[col].dtype == 'category']
    preprocessor = make_column_transformer((OneHotEncoder(), categorical_cols), remainder='passthrough')
    xgb_model = make_pipeline(preprocessor, XGBRegressor())
    xgb_model.fit(X_train, y_train)
    # Предсказание цен на тестовом наборе
    y_pred = xgb_model.predict(X_test)
    # Оценка качества модели
    xgb_r2 = r2_score(y_test, y_pred)
    xgb_mse = mean_squared_error(y_test, y_pred)
    xgb_rmse = mean_squared_error(y_test, y_pred, squared=False)
    xgb_mae = mean_absolute_error(y_test, y_pred)
    print('Коэффициент детерминации (R^2):', xgb_r2)
    print('Среднеквадратичная ошибка (MSE):', xgb_mse)
    print('Корень среднеквадратичной ошибки (RMSE):', xgb_rmse)
    print('Средняя абсолютная ошибка (MAE):', xgb_mae)
```

Вывод результатов:

```
Коэффициент детерминации (R^2): 0.8709585608153281
Среднеквадратичная ошибка (MSE): 154603515.95564193
Корень среднеквадратичной ошибки (RMSE): 12433.966219820686
Средняя абсолютная ошибка (MAE): 7363.213696289063
```

Создание и обучение модели «Дерево решений»с учетом категориальных переменных:

```
    # СозданиеиобучениемоделиDecisionTreeRegressor
    categorical_cols = [col for col inX.columnsif X[col].dtype == 'category']
    preprocessor = make_column_transformer((OneHotEncoder(), categorical_cols), remainder='passthrough')
    dt_model = make_pipeline(preprocessor, DecisionTreeRegressor())
    dt_model.fit(X_train, y_train)
    # Предсказание цен на тестовом наборе
    y_pred = dt_model.predict(X_test)
    # Оценка качества модели
    dt_r2 = r2_score(y_test, y_pred)
    dt_mse = mean_squared_error(y_test, y_pred)
    dt_rmse = mean_squared_error(y_test, y_pred, squared=False)  # RMSE
    dt_mae = mean_absolute_error(y_test, y_pred)
    print('Коэффициент детерминации (R^2):', dt_r2)
    print('Среднеквадратичная ошибка (MSE):', dt_mse)
    print('Корень среднеквадратичной ошибки (RMSE):', dt_rmse)
    print('Средняя абсолютная ошибка (MAE):', dt_mae)
```

Вывод результатов:

```
Коэффициент детерминации (R^2): 0.8145453354166459
Среднеквадратичная ошибка (MSE): 222191750.0
Корень среднеквадратичной ошибки (RMSE): 14906.097745553663
Средняя абсолютная ошибка (MAE): 9082.5
```

Создание и обучение модели «Метод ближайших соседей» с учетом категориальных переменных:

```
    categorical_cols = [col for col inX.columnsif X[col].dtype == 'category']
    preprocessor = make_column_transformer((OneHotEncoder(), categorical_cols), remainder='passthrough')
    model = make_pipeline(preprocessor, KNeighborsRegressor())
    model.fit(X_train, y_train)
    # Предсказание цен на тестовом наборе данных
    y_pred = model.predict(X_test)
    # Вычислениеметрик
```

```
KN_r2 = r2_score(y_test, y_pred)
KN_mse = mean_squared_error(y_test, y_pred)
KN_rmse = np.sqrt(KN_mse)
KN_mae = mean_absolute_error(y_test, y_pred)
print(f'R^2: {KN_r2}')
print(f'MSE: {KN_mse}')
print(f'RMSE: {KN_rmse}')
print(f'MAE: {KN_mae}')
```

Вывод результатов:

```
R^2: 0.20841534706850562
MSE: 948391240.0
RMSE: 30795.96142353734
MAE: 18736.0
```

Создание таблицы сравнения методов с полученными данными:

```
data = {'Модель': ['LinearRegression', 'RandomForestRegressor', 'XGBoost', 'DecisionTreeRegressor', 'KNeighborsRegressor'],
        'R^2': [lr_r2, f_r2, xgb_r2, dt_r2, KN_r2],
        'MSE': [lr_mse, f_mse, xgb_mse, dt_mse, KN_mse],
        'RMSE': [lr_rmse, f_rmse, xgb_rmse, dt_rmse, KN_rmse],
        'MAE': [lr_mae, f_mae, xgb_mae, dt_mae, KN_mae]}
df1 = pd.DataFrame(data)
# Выводтаблицы
print(df1)
```

Таблица 1 – Сравнение методов с полученными данными.

Модель	R^2	MSE	RMSE	MAE
LinearRegression	0.570377	5.147282e+08	22687.621080	15876.658934
RandomForestRegressor	0.846433	1.839870e+08	13564.182136	8718.298849
XGBoost	0.870959	1.546035e+08	12433.966220	7363.213696
DecisionTreeRegressor	0.814545	2.221918e+08	14906.097746	9082.500000
KNeighborsRegressor	0.208415	9.483912e+08	30795.961424	18736.000000

Из таблицы 1 по полученным данным, можно сделать следующие выводы:

- **XGBoost**демонстрирует наилучшую производительность среди всех моделей, имея наивысший коэффициент детерминации (R^2) и наименьшие значения среднеквадратичной ошибки (MSE), корня среднеквадратичной ошибки (RMSE) и средней абсолютной ошибки (MAE). Это указывает на то, что XGBoost лучше всего соответствует данным и делает более точные прогнозы.
- **RandomForest** также показывает хорошие результаты, с высоким значением R^2 и относительно низкими значениями MSE, RMSE и MAE. Это указывает на эффективность случайного леса в предсказании целевой переменной.
- **DecisionTree** также демонстрирует хорошие результаты, с высоким значением R^2, но немного более высокими значениями MSE, RMSE и MAE по сравнению с XGBoost и RandomForest.
- **LinearRegression** показывает более скромные результаты по сравнению с XGBoost, RandomForest и DecisionTree, с более низким значением R^2 и более высокими значениями ошибок.

- **KNeighborsRegressor**демонстрирует наихудшую производительность среди всех моделей, имея наименьший коэффициент детерминации (R^2) и наибольшие значения ошибок.

Исходя из этих результатов, можно сделать вывод, что **XGBoost** и **RandomForest** являются наиболее подходящими моделями для данной задачи предсказания цен, в то время как **KNeighborsRegressor** показывает наихудшие результаты.

Создаем модель «Линейная регрессия» для предсказания значения «Цена» с применением натурального логарифма на основе остальных данных и используем дополнительный метод GridSearch:

```
fromsklearn.composeimportColumnTransformer
fromsklearn.preprocessingimportStandardScaler, OneHotEncoder
fromsklearn.pipelineimport Pipeline
fromsklearn.model_selectionimporttrain_test_split, GridSearchCV
# Применение натурального логарифма к целевой переменной
y = np.log(y)      # y = df['Цена']
# Разделение данных на обучающий и тестовый наборы
X_train,   X_test,   y_train,   y_test   =   train_test_split(X,   y,
test_size=0.2, random_state=42)
#  Создание  и  обучение  модели  линейной  регрессии  с  учетом
категориальных переменных и нормализацией
preprocessor = ColumnTransformer(
    transformers=[
            ('num',  StandardScaler(),  ['Частота'  ,  'Размер_экрана',
'Год_релиза']), # числовыестолбцы
          ('cat', OneHotEncoder(), ['Бренд', 'Камера', 'ОЗУ', 'ОВЗ',
'Ядра', 'NFC'])  # категориальныестолбцы
    ], remainder='passthrough')
l_model = Pipeline(steps=[('preprocessor', preprocessor),
                          ('regressor', LinearRegression())])
# ОпределениесеткипараметровдляGridSearch
param_grid   =   {'regressor__fit_intercept':   [True,   False]}   #
ПараметрдляучетапересечениЯ
# СозданиеобъектаGridSearchCV
grid_search = GridSearchCV(l_model, param_grid, cv=5)
# ОбучениемоделисиспользованиемGridSearch
grid_search.fit(X_train, y_train)
# Получение лучших параметров и оценка качества модели
best_params = grid_search.best_params_
best_model = grid_search.best_estimator_
y_pred = best_model.predict(X_test)
# Оценкакачествамодели
lrl_r2 = best_model.score(X_test, y_test)
lrl_mse = mean_squared_error(y_test, y_pred)
lrl_rmse = mean_squared_error(y_test, y_pred, squared=False)
lrl_mae = mean_absolute_error(y_test, y_pred)
print('Лучшиепараметры:', best_params)
print('Коэффициент детерминации (R^2):', lrl_r2)
print('Среднеквадратичная ошибка (MSE):', lrl_mse)
print('Корень среднеквадратичной ошибки (RMSE):', lrl_rmse)
print('Средняя абсолютная ошибка (MAE):', lrl_mae)
```

Вывод результатов:

```
Лучшие параметры: {'regressor__fit_intercept': True}
Коэффициент детерминации (R^2): 0.8441440981380995
Среднеквадратичная ошибка (MSE): 0.09347511672380124
Корень среднеквадратичной ошибки (RMSE): 0.3057370058134953
Средняя абсолютная ошибка (MAE): 0.239787100895549
```

Создаем модель «Случайный лес» с применением натурального логарифма и использованием дополнительного метода GridSearch:

```
rf_model = Pipeline(steps=[('preprocessor', preprocessor),
                           ('regressor', RandomForestRegressor())])
# Определениесеткипараметровдля GridSearch
param_grid1 = {
              'regressor__n_estimators':    [100,    200,    300],    #
Параметрдляколичествадеревьеввлесу
          'regressor__max_depth':  [None,  5,  10,  15]}#  Параметр  для
максимальной глубины деревьев
# СозданиеобъектаGridSearchCV
grid_search1 = GridSearchCV(rf_model, param_grid1, cv=5)
# Обучениемоделисиспользованием GridSearch
grid_search1.fit(X_train, y_train)
# Получение лучших параметров и оценка качества модели
best_params = grid_search1.best_params_
best_model1 = grid_search1.best_estimator_
y_pred = best_model1.predict(X_test)
# Оценкакачествамодели
rf_r2 = best_model1.score(X_test, y_test)
rf_mse = mean_squared_error(y_test, y_pred)
rf_rmse = mean_squared_error(y_test, y_pred, squared=False)
rf_mae = mean_absolute_error(y_test, y_pred)
print('Лучшиепараметры:', best_params)
print('Коэффициент детерминации (R^2):', rf_r2)
print('Среднеквадратичная ошибка (MSE):', rf_mse)
print('Корень среднеквадратичной ошибки (RMSE):', rf_rmse)
print('Средняя абсолютная ошибка (MAE):', rf_mae)
```

Вывод результатов:

```
Лучшие параметры: {'regressor__max_depth': None,
'regressor__n_estimators': 100}
Коэффициент детерминации (R^2): 0.8847269901658708
Среднеквадратичная ошибка (MSE): 0.06913538672983131
Корень среднеквадратичной ошибки (RMSE): 0.2629360886790387
Средняя абсолютная ошибка (MAE): 0.21230248701139048
```

Создание и обучение модели «XGBoost» с применением натурального логарифма, нормализации данных и с учетом категориальных переменных:

```
# Удаление пропущенных значений из данных
X = X.dropna()
y = y.dropna()
# Применение натурального логарифма к целевой переменной
y = np.log(y)
# Разделение данных на обучающий и тестовый наборы
X_train,   X_test,   y_train,   y_test   =   train_test_split(X,   y,
test_size=0.2, random_state=42)
# Создание и обучение модели XGBoost с учетом категориальных
переменных и нормализацией
preprocessor = ColumnTransformer(
    transformers=[
            ('num', StandardScaler(), ['Частота', 'Размер_экрана',
'Год_релиза']),
          ('cat', OneHotEncoder(), ['Бренд', 'Камера', 'ОЗУ', 'ОВЗ',
'Ядра', 'NFC'])
     ], remainder='passthrough')

xl_model = Pipeline(steps=[('preprocessor', preprocessor),
                           ('regressor', XGBRegressor())])
# Обучениемодели
xl_model.fit(X_train, y_train)
```

```
# Предсказание на тестовом наборе
y_pred = xl_model.predict(X_test)
# Оценка качества модели
xl_r2 = xl_model.score(X_test, y_test)
xl_mse = mean_squared_error(y_test, y_pred)
xl_rmse = mean_squared_error(y_test, y_pred, squared=False)
xl_mae = mean_absolute_error(y_test, y_pred)
print('Коэффициент детерминации (R^2):', xl_r2)
print('Среднеквадратичная ошибка (MSE):', xl_mse)
print('Корень среднеквадратичной ошибки (RMSE):', xl_rmse)
print('Средняя абсолютная ошибка (MAE):', xl_mae)
```

Вывод результатов:

```
Коэффициент детерминации (R^2): 0.8694652347135328
Среднеквадратичная ошибка (MSE): 0.0007283046530116154
Корень среднеквадратичной ошибки (RMSE): 0.02698712013186319
Средняя абсолютная ошибка (MAE): 0.021445386686712253
```

Создание и обучение модели «Дерево решений» с применением натурального логарифма, нормализации данных и с учетом категориальных переменных:

```
dl_model = Pipeline(steps=[('preprocessor', preprocessor),
                           ('regressor', DecisionTreeRegressor())])
# Обучениемодели
dl_model.fit(X_train, y_train)
# Предсказание на тестовом наборе
y_pred = dl_model.predict(X_test)
# Оценка качества модели
dl_r2 = dl_model.score(X_test, y_test)
dl_mse = mean_squared_error(y_test, y_pred)
dl_rmse = mean_squared_error(y_test, y_pred, squared=False)
dl_mae = mean_absolute_error(y_test, y_pred)
print('Коэффициент детерминации (R^2):', dl_r2)
print('Среднеквадратичная ошибка (MSE):', dl_mse)
print('Корень среднеквадратичной ошибки (RMSE):', dl_rmse)
print('Средняя абсолютная ошибка (MAE):', dl_mae)
```

Вывод результатов:

```
Коэффициент детерминации (R^2): 0.7841183310187337
Среднеквадратичная ошибка (MSE): 0.0012044885029204521
Корень среднеквадратичной ошибки (RMSE): 0.03470574164198846
Средняя абсолютная ошибка (MAE): 0.027079007555482138
```

Создание и обучение модели «Метод ближайших соседей» с применением натурального логарифма, нормализации данных и с учетом категориальных переменных:

```
kl_model = Pipeline(steps=[('preprocessor', preprocessor),
                           ('regressor', KNeighborsRegressor())])
# Обучениемодели
kl_model.fit(X_train, y_train)
# Предсказание на тестовом наборе
y_pred = kl_model.predict(X_test)
# Оценкакачествамодели
kl_r2 = kl_model.score(X_test, y_test)
kl_mse = mean_squared_error(y_test, y_pred)
kl_rmse = mean_squared_error(y_test, y_pred, squared=False)
kl_mae = mean_absolute_error(y_test, y_pred)
print('Коэффициент детерминации (R^2):', kl_r2)
print('Среднеквадратичная ошибка (MSE):', kl_mse)
print('Корень среднеквадратичной ошибки (RMSE):', kl_rmse)
print('Средняя абсолютная ошибка (MAE):', kl_mae)
```

Вывод результатов:

```
Коэффициент детерминации (R^2): 0.7163052746808678
Среднеквадратичная ошибка (MSE): 0.001582844141415838
Корень среднеквадратичной ошибки (RMSE): 0.03978497381444203
Средняя абсолютная ошибка (MAE): 0.0322641625986679
```

Создание таблицы сравнения методовс применением натурального логарифмапо полученным данным:

```
    data1 = {'Модель': ['LinearRegression (log)', 'RandomForestRegressor (log)', 'XGBoost (log)', 'DecisionTreeRegressor (log)', 'KNeighborsRegressor (log)'],
            'R^2': [lrl_r2, rf_r2, xl_r2, dl_r2, kl_r2],
            'MSE': [lrl_mse, rf_mse, xl_mse, dl_mse, kl_mse],
            'RMSE': [lrl_rmse, rf_rmse, xl_rmse, dl_rmse, kl_rmse],
            'MAE': [lrl_mae, rf_mae, xl_mae, dl_mae, kl_mae]}
    df2 = pd.DataFrame(data1)
    # Выводтаблицы
    print(df2)
```

Таблица 2 – Сравнение методов с применением натурального логарифма.

Модель	R^2	MSE	RMSE	MAE
LinearRegression(log)	0.844144	0.093475	0.305737	0.239787
RandomForestRegressor(log)	0.884727	0.069135	0.262936	0.212302
XGBoost(log)	0.869465	0.000728	0.026987	0.021445
DecisionTreeRegressor(log)	0.784118	0.001204	0.034706	0.027079
KNeighborsRegressor(log)	0.716305	0.001583	0.039785	0.032264

Основываясь на показателях производительности различных моделей, можно сделать следующие выводы:

Модель линейной регрессии достигла значения R^2, равного 0,844, что указывает на то, что она объясняет примерно 84,4% дисперсии целевой переменной. Однако прогнозы модели привели к относительно высокой среднеквадратичной ошибке (MSE) в 0,0935 и среднеквадратичной ошибке (RMSE) в 0,3057, что предполагает заметный уровень отклонения от фактических значений. Средняя абсолютная ошибка (MAE), равная 0,2398, означает среднюю абсолютную разницу между прогнозируемыми и фактическими значениями.

Модель случайного леса продемонстрировала сильную способность объяснять дисперсию целевой переменной со значением R^2, равным 0,8847. Более низкий MSE в 0,0691 по сравнению с моделью линейной регрессии предполагает, что прогнозы модели меньше отличаются от фактических значений. RMSE, равный 0,2629, указывает на меньшее среднее расстояние между прогнозируемыми и фактическими значениями. Значение MAE, равное 0,223, ниже, чем у модели линейной регрессии, что означает меньшую среднюю абсолютную разницу между прогнозируемыми и фактическими значениями.

Модель XGBoost достигла значения R^2, равного 0,8695, что указывает на высокую степень объяснения дисперсии, аналогичную модели случайного леса. Чрезвычайно низкий MSE, равный 0,0007, предполагает,

что прогнозы модели имеют очень малое отклонение от фактических значений. RMSE, равный 0,0270, значительно ниже, чем как в моделях линейной регрессии, так и в моделях случайного леса, что указывает на меньшее среднее расстояние между прогнозируемыми и фактическими значениями. Значение MAE 0,0214 является самым низким среди моделей, что означает наименьшую среднюю абсолютную разницу между прогнозируемыми и фактическими значениями.

Модель дерева решений объясняет примерно 78,4% дисперсии целевой переменной, на что указывает значение R^2, равное 0,7841. Прогнозы модели привели к относительно низкому MSE 0,0012 и умеренному RMSE 0,0347, что означает умеренное среднее расстояние между прогнозируемыми и фактическими значениями. Значение MAE 0,0271 является относительно низким, указывая на умеренную среднюю абсолютную разницу между прогнозируемыми и фактическими значениями.

Модель K-ближайших соседей объясняет примерно 71,6% дисперсии целевой переменной со значением R^2, равным 0,7163. Прогнозы модели привели к относительно низкому MSE 0,0016 и умеренному RMSE 0,0398, что означает умеренное среднее расстояние между прогнозируемыми и фактическими значениями. Значение MAE, равное 0,0323, означает умеренную среднюю абсолютную разницу между прогнозируемыми и фактическими значениями.

В целом, использование логарифма значительно улучшило производительность всех моделей, с **RandomForestRegressor (log)** и **XGBoost (log)** выделяющимися на фоне остальных моделей.

Из проведенного сравнения двух таблиц можно сделать следующие выводы:

Без использования логарифма:

Модели LinearRegression, DecisionTreeRegressor и KNeighborsRegressor показали более низкие значения коэффициента детерминации (R^2) и более высокие значения среднеквадратической ошибки (MSE), корня из среднеквадратической ошибки (RMSE) и средней абсолютной ошибки (MAE) по сравнению с моделями RandomForestRegressor и XGBoost. Модель XGBoost показала наилучшую общую производительность с самым высоким значением R^2 и самыми низкими значениями RMSE и MAE.

С использованием логарифма:

При использовании логарифма все модели показали значительное улучшение производительности по сравнению с предыдущими результатами. Модель RandomForestRegressor (log) выделяется с наивысшим значением R^2 и наименьшими значениями MSE, RMSE и MAE.

По результатам исследования лучшей моделью стала RandomForestRegressor с использованием логарифма.

4 Выводы

В данной статье были выполнены все поставленные задачи. Был проведён сбор и анализ данных о характеристиках смартфонов с ОС Android

и их ценах на сайте DNS г. Биробиджан. Обработаны данные для применения методов искусственного интеллекта. Были использованы модели методов линейной регрессии, случайного леса, XGBoost, дерева решений и метода ближайших соседей для прогнозирования стоимости смартфонов. Проведена оценка точности прогнозов и выявлены факторы, влияющие на стоимость смартфонов с ОС Android.

В ходе исследования был проведен сравнительный анализ эффективности каждого метода и выбор наиболее подходящих моделей для предсказания цены. Модели LinearRegression, DecisionTreeRegressor и KNeighborsRegressor показали более низкие значения коэффициента детерминации (R^2) и более высокие значения среднеквадратической ошибки (MSE), корня из среднеквадратической ошибки (RMSE) и средней абсолютной ошибки (MAE) по сравнению с моделями RandomForestRegressor и XGBoost. Модель XGBoost показала наилучшую общую производительность с самым высоким значением R^2 и самыми низкими значениями RMSE и MAE.

При использовании логарифма все модели показали значительное улучшение производительности по сравнению с предыдущими результатами. В частности, модель RandomForestRegressor (log) показывает наивысший коэффициент детерминации (R^2) и наименьшие значения среднеквадратичной ошибки (MSE), корня из среднеквадратичной ошибки (RMSE) и средней абсолютной ошибки (MAE). Модель XGBoost (log) также продемонстрировала высокую точность прогнозов с минимальными значениями ошибок. Однако, необходимо учитывать особенности рынка смартфонов и возможные изменения во времени, что требует дальнейшего исследования и анализа. В целом, данная тема представляет интерес для дальнейших исследований в области методов искусственного интеллекта и их применения для прогнозирования стоимости смартфонов с ОС Android.

Библиографический список

1. Овчаренко Б.В., Сурмило С.В. Программная реализация прогнозирования пожаров на основе адаптивных моделей прогнозирования // Пожарная и техносферная безопасность: проблемы и пути совершенствования. 2021. № 3 (10). С. 296-301.
2. Кано Д., Накано Ю. Устройство для прогнозирования энергопотребления и способ для прогнозирования энергопотребления // Патент на изобретение RU 2639713, 22.12.2017. Заявка № 2015148146 от 28.03.2014.
3. Сай В.К., Щербаков М.В. Прогнозирование отказов сложных много объектных систем на основе комбинации нейросетей: пути повышения точности прогнозирования // Прикаспийский журнал: управление и высокие технологии. 2020. № 1 (49). С. 49-60.
4. Попков Ю.С. Машинное обучение и рандомизированное машинное обучение: сходство и различие // В сборнике: Системный анализ и информационные технологии САИТ-2019. Труды Восьмой

международной конференции. 2019. С. 10-25.
5. Галимов Р.Г. Основы алгоритмов машинного обучения - обучение с учителем // Аллея науки. 2017. Т. 1. № 14. С. 810-817.
6. Polonnikov I. Prospect of the development of the A/B testing based on machine learning // В сборнике: Languages in professional communication. 2020. С. 609-613.
7. Ulyanikhin E. Machine learning in information security field // В сборнике: Languages in professional communication. 2020. С. 704-707.
8. Fadeev V.A., ZaidullinSh.V., Nadeev A.F. Investigation of the bayesian and non-bayesian time series forecasting frameworks in application to OSS systems of the LTE/LTE-a and 5G mobile networks // T-Comm. 2022. Т. 16. № 4. С. 52-60.
9. Lomakin N., Kulachinskaya A., Naumova S., Ibrahim M., Fedorovskaya E., Lomakin I. Modelling profits forecasts for the russian banking sector using random forest and regression algorithm // Sustainable Development and Engineering Economics. 2023. № 3 (9). С. 8-20
10. Parbat D., Chakraborty M. A python-based support vector regression model for prediction of COVID19 cases in INDIA // Chaos, Solitons & Fractals. 2020. Т. 138. С. 109942.
11. Данные URL:https://docs.google.com/spreadsheets/d/1wb2ihpc7CBKBoA6KtHyq4rUq4t2FarYz/edit?usp=sharing&ouid=102441220387432308502&rtpof=true&sd=true
12. Данные URL:https://colab.research.google.com/drive/1PjlWw6PiOkYN20pAV2Wkgwbs5v9tYZNK?usp=sharing