

Применение машинного обучения для прогнозирования цен на ноутбуки: сравнительный анализ моделей

Лапушкина Елена Павловна

Приамурский государственный университет им. Шолом-Алейхема

Студент

Аннотация

Целью исследования является применение различных методов машинного обучения для построения модели с помощью прогнозирования цен на ноутбуки. В данной статье приведён процесс обучения системы предсказывающей цены ноутбуков на основании характеристик собранного датасета сети магазинов DNS ритейл, с помощью парсинга. В результате исследования выбрана модель и переменные, обеспечивающие лучшие результаты для текущей задачи. Сравнение моделей произведено на основании основных метрик: R2, MSE.

Ключевые слова: Модель обучения, прогнозирование, парсинг, датасет.

The use of machine learning to predict laptop prices: a comparative analysis of models

Lapushkina Elena Pavlovna

Sholom-Aleichem Priamursky State University

Student

Abstract

The purpose of the study is to apply various machine learning methods to build a model for predicting laptop prices. This article describes the process of training a system that predicts laptop prices based on characteristics using a created dataset. As a result of the research, a model and variables were selected that provide the best results for the current task. The models were compared based on the main metrics: R2, MSE.

Keywords: Learning model, forecasting, parsing, dataset.

1 Введение

1.1 Актуальность

В современном мире объёмы данных растут в геометрической прогрессии, и умение извлекать ценную информацию становится все более критическим для успешного функционирования бизнеса. Одной из важнейших задач в этом контексте является анализ и понимание факторов, влияющих на ценообразование товаров и услуг.

1.2 Обзор исследований

А.А. Маковеев, М.И. Сафонов, А.Г. Ковалёв, К.А. Федченко в своем исследовании рассмотрели разработку системы для агрегации наиболее выгодных предложений по покупке автозапчастей. На данном этапе было проведено проектирование и создание концепта будущей системы [1]. В.А. Кононов в данной статье описал исследование по разбору и описанию программного кода, выполняющего парсинг сайтов с выводом информации в удобном для пользователя виде. Программный код представлен на языке программирования Python [2]. Т.С. Волокитина исследовала преимущества о возможности Google Colab, для изучения технологий машинного обучения и нейронных сетей. Предоставила теоретическую справку и анализ возможностей Google Colab [3]. В.Н. Клячкин, В.Н. Ковальногов, В.В. Шеркунов, Х. Мохамед в своей научной работе посвятили исследованию анализа влияния различных факторов на эффективность работы горелочных устройств, оцениваемую по температуре дымовых газов, с применением методов машинного обучения [4]. А.В. Дагаев рассмотрел, как анализ развития языков программирования позволяет упростить и ускорить процесс анализа данных. [5].

1.3 Цель исследования

Целью исследования является применение различных методов машинного обучения для построения модели прогнозирования цен на ноутбуки.

Материалы и методы

Для реализации был использован язык программирования Python, сервис Google Colab. Данные для датасета были взяты с официального сайта DNS.

Ссылка на датасет:
<https://docs.google.com/spreadsheets/d/1hr87pheAovKvz9pUnWwT2f-bXrDMgf-T/edit?usp=sharing&ouid=110343077311761014179&rtpof=true&sd=true>

Ссылка на официальный сайт DNS: https://www.dns-shop.ru/?utm_referrer=https%3A%2F%2Fwww.dns-shop.ru%2Fproduct%2F127c5bcb7f0aed20%2Fumnaa-kolonka-novaa-andeksstancia-mini-cernyj%2F

Ссылка на ноутбук:
<https://colab.research.google.com/drive/1KD6nXf9m4hrBiFWxGxkFX5y8mOZeU1L1?usp=sharing>

3 Результаты и дискуссии

Для сбора необходимой информации о товаре, нужно написать простой парсер, который будет считывать данные с сайта и отправлять их в соответствующую таблицу Excel. Это делается для того, чтобы ускорить процесс сбора данных. Создать новый проект в PyCharm, далее нужно

импортировать нужные библиотеки, с которыми предстоит работать: pandas, selenium и Webdriver Manager (рис. 1).

```
1 import pandas as pd
2 from selenium import webdriver
3 from selenium.webdriver.edge.options import Options
4 from webdriver_manager.microsoft import EdgeChromiumDriverManager
5 from selenium.webdriver.common.by import By
6 from selenium.webdriver.edge.service import Service
7 from selenium.webdriver.support.ui import WebDriverWait
8 from selenium.webdriver.support import expected_conditions as EC
9 import re
10
```

Рисунок 1 –Создание проекта в PyCharm

Далее необходимо указать URL адрес главного сайта для парсинга. Чтобы парсить сайт и получать с него именно те данные, которые необходимо, следует перейти и посмотреть интересные строки, в которых могут содержаться символы. Данные символы могут мешать дальнейшему анализу данных, поэтому необходимо понять, какие символы нужно очистить чтобы данные были одинаковые (т.е. пробелы, “#” и тд.) (рис. 2).

```
13 def clean_genre_name(genre): 1 usage
14     return genre.replace("#", "").strip()
15
```

Рисунок 2- Очистка от ненужных символов

Следующий шаг, необходимо перейти непосредственно к написанию кода для браузера, который будет подключаться для удаленной работы на странице, также задать ему параметр «headless», для работы без графического интерфейса. Чтобы знать с какими ошибками можно столкнуться, рекомендуется делать отладку (рис. 2).

```
23 def parse_manga_list(url): 1 usage
24     print(f"Шаг 1: Запуск браузера и подключение к странице {url}")
25
26     options = Options()
27     options.add_argument("--headless")
28     driver_path = EdgeChromiumDriverManager().install()
29     service = Service(driver_path)
30     driver = webdriver.Edge(service=service, options=options)
31
```

Рисунок 5- Инициализация браузера

Теперь необходимо, чтобы парсер нашел всю мангу на главной странице, для этого ранее была использована функция WebDriverWait, у

которого было выставлено значение «10», то есть когда появятся элементы с определенным классом, то парсер не начнет работать дальше. манга будет искаться по элементу item-grid, если ничего не будет найдено на странице, то в отладке появится сообщение: «манга не найдена на странице», а если манга будет найдена, то при каждой найденной, появятся сообщения: «Найдено <количество манги> манга на странице <номер страницы> (рис.3).

```

35     mangas = driver.find_elements(By.CLASS_NAME, value: "item-grid")
36     if not mangas:
37         print("Манга не найдена на странице.")
38         driver.quit()
39         return
40     print(f"Найдено {len(mangas)} манг на странице.")
41     manga_list = []
42

```

Рисунок 3- Поиск общего количества манги на первой странице

Стоит отметить, что те элементы, которые указаны в коде, были получены из элемента в панели разработчика в браузере, чтобы в него попасть нужно нажать сочетание клавиш «Ctrl»+«Shift»+«I». В коде поиск XPath осуществляется по блокам <div> определенных строк, которые будут указаны в тексте (рис. 4).

```

... <span data-target="views-counter.output">15.26K</span> == $0

```

Рисунок 4- Извлечение текста из элемента браузера

Следующим шагом нужно выполнить проверку файла, в который будут сохраняться данные, если список не пуст, то данные сохраняются, если пуст – выводится сообщение об ошибке. Задается имя файла, который будет сохранен и сообщение отладчика «Данные сохранены в файл 'manga_list.xlsx'». В конце происходит закрытие браузера и вывод данной информации. В самом конце прописывается запуск скрипта (рис. 5).

```

116     if manga_list:
117         df = pd.DataFrame(manga_list)
118         df.to_excel( excel_writer: "manga_list.xlsx", index=False)
119         print("Данные сохранены в файл 'manga_list.xlsx'")
120     else:
121         print("Не удалось найти мангу.")
122
123     print("Закрытие браузера")
124     driver.quit()
125
126 f __name__ == "__main__":
127     parse_manga_list(BASE_URL)

```

Рисунок 5- Сохранение манги в файл Excel и запуск главного скрипта

Далее загрузить получившийся датасет в ноутбук Google Colab, и провести этапы исследования (рис. 6).

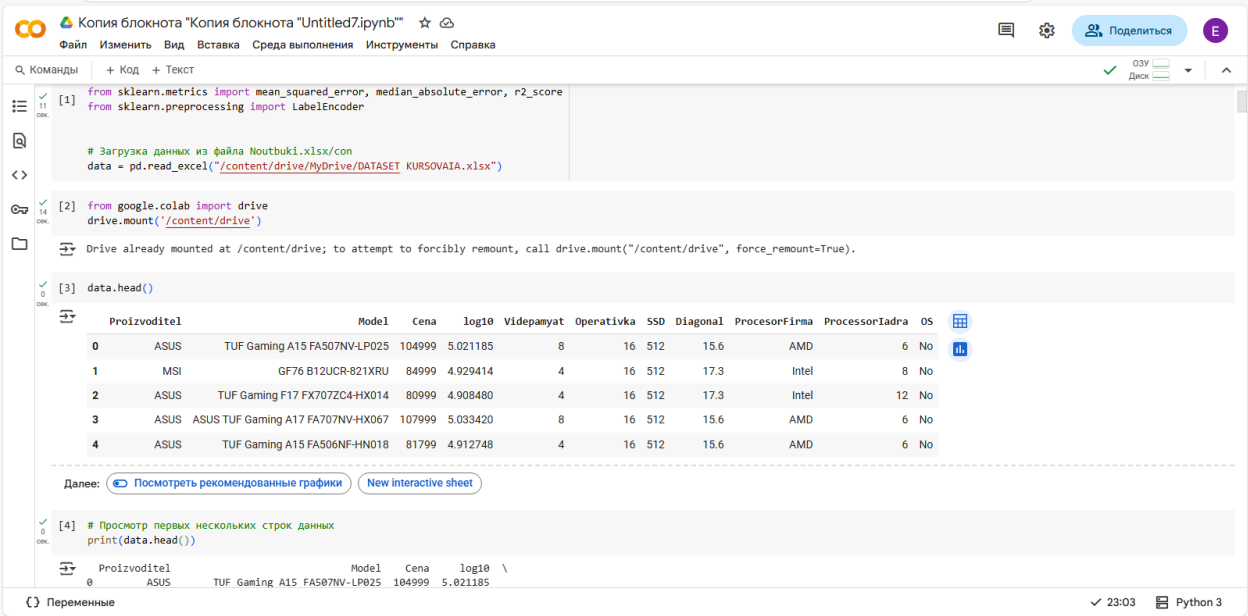


Рисунок 6- Загрузка данных

Далее с помощью описания кода провести анализ данных (рис. 7).

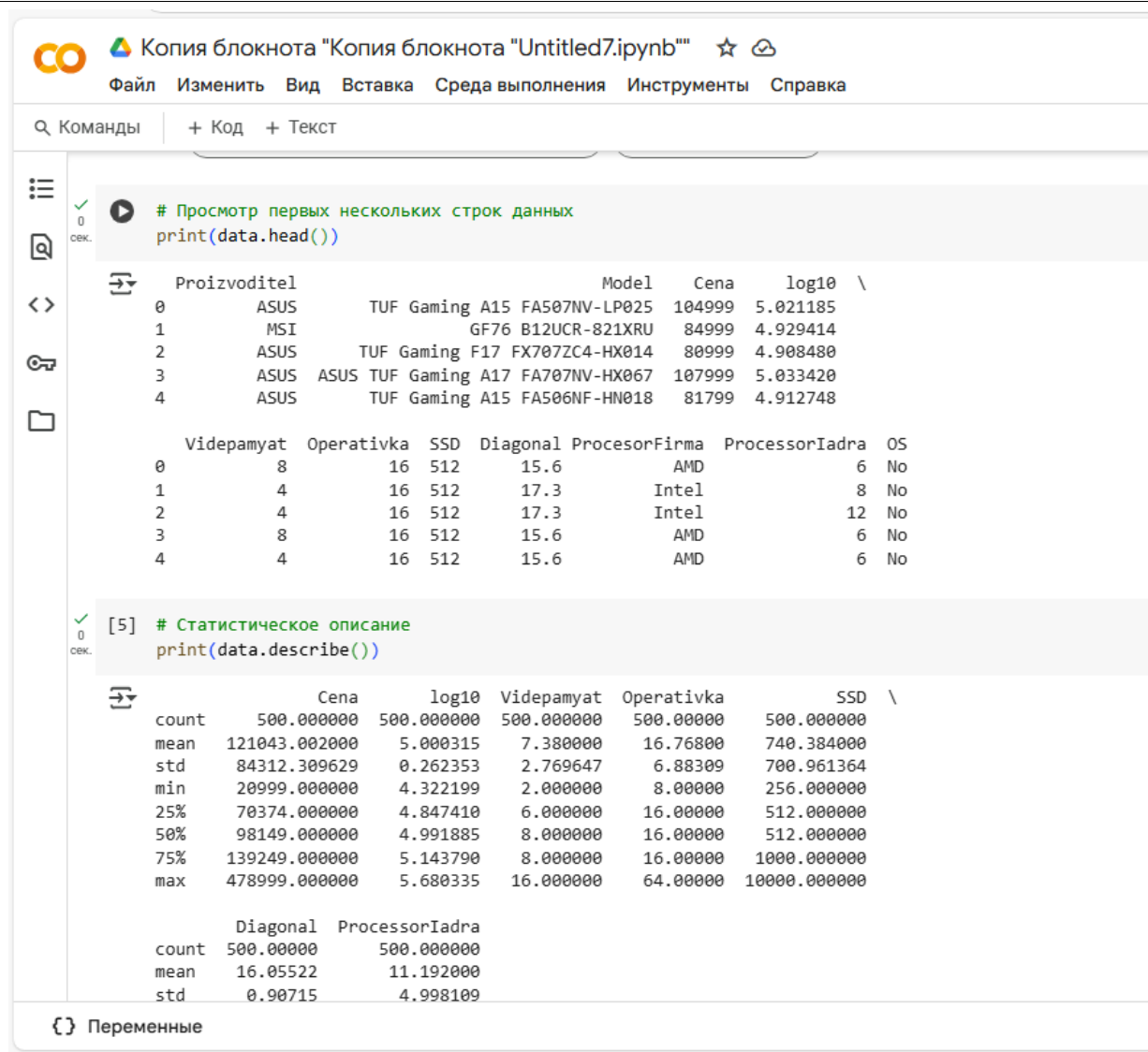


Рисунок 7- Анализ данных

Далее с помощью описания кода изобразить графически количественное распределение ноутбуков по цене (рис.8).

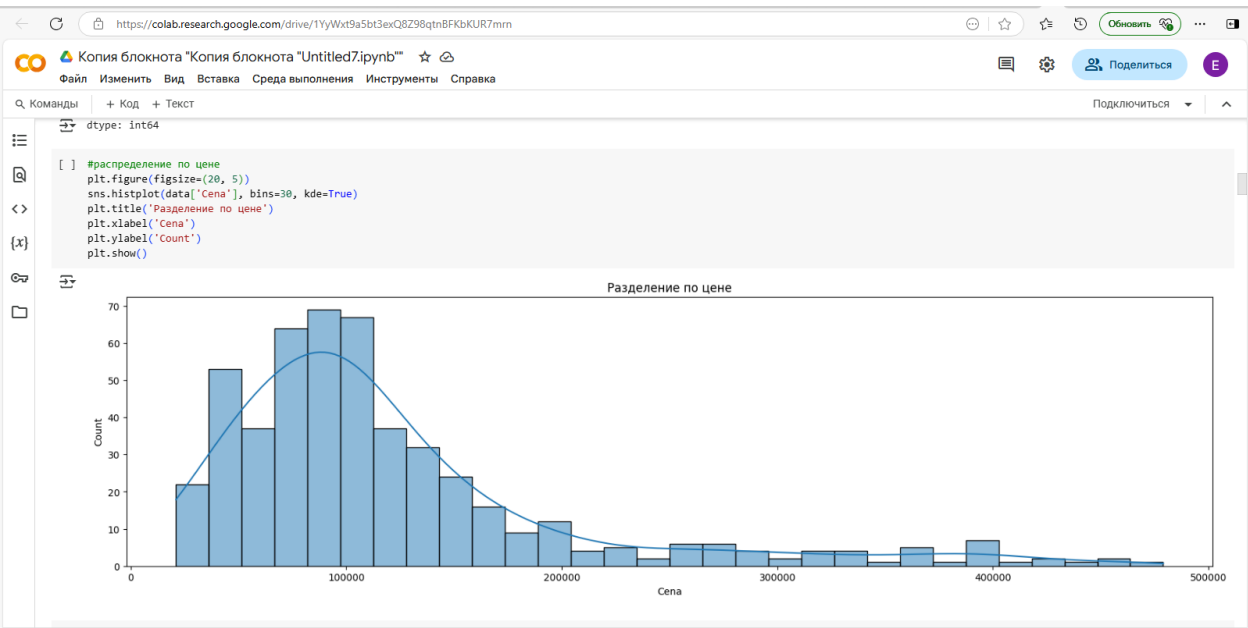


Рисунок 8- Обзор разделения по цене

С помощью описания кода отобразить распределение по остальным категориям (рис.9), (рис. 10), (рис. 11), (рис. 12), (рис. 13), (рис. 14), (рис. 15), (рис. 16), (рис. 17), (рис.18), (рис. 19), (рис. 20).

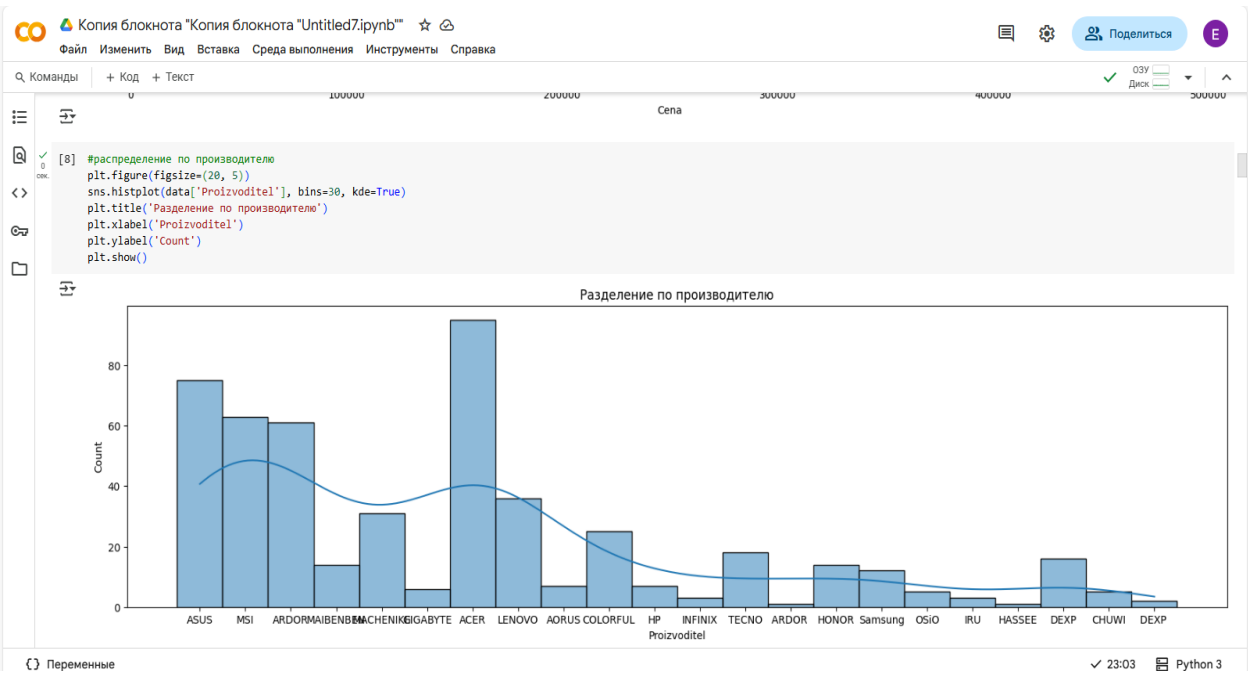


Рисунок 9- Обзор разделения по производителю

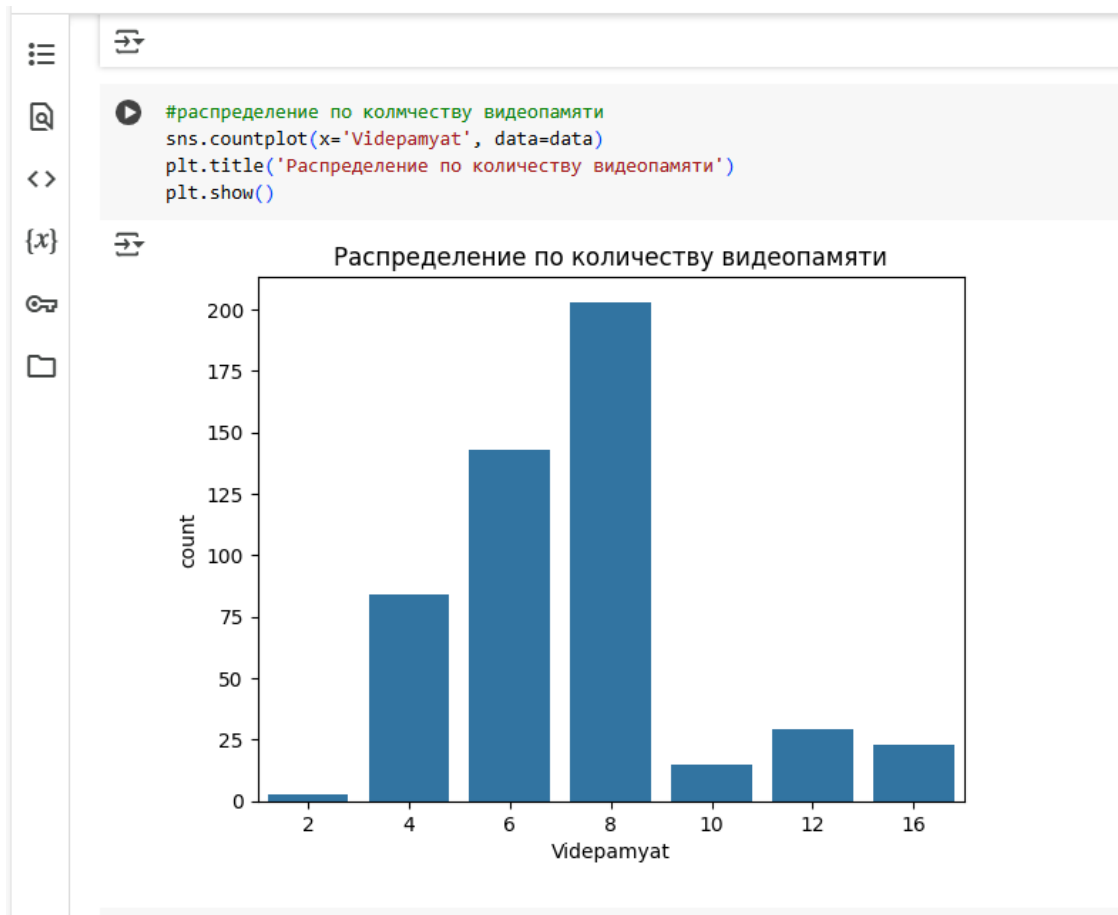


Рисунок 10- Обзор распределения по количеству видеопамяти

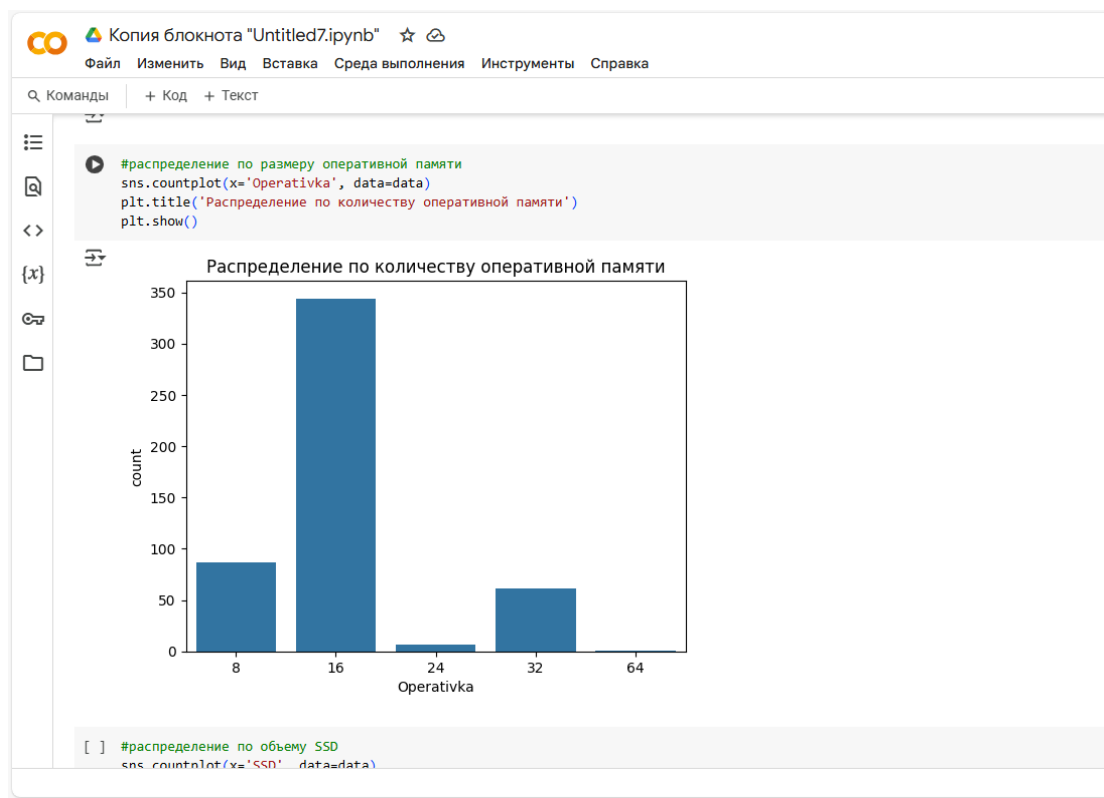


Рисунок 11- Обзор распределения по количеству оперативной памяти

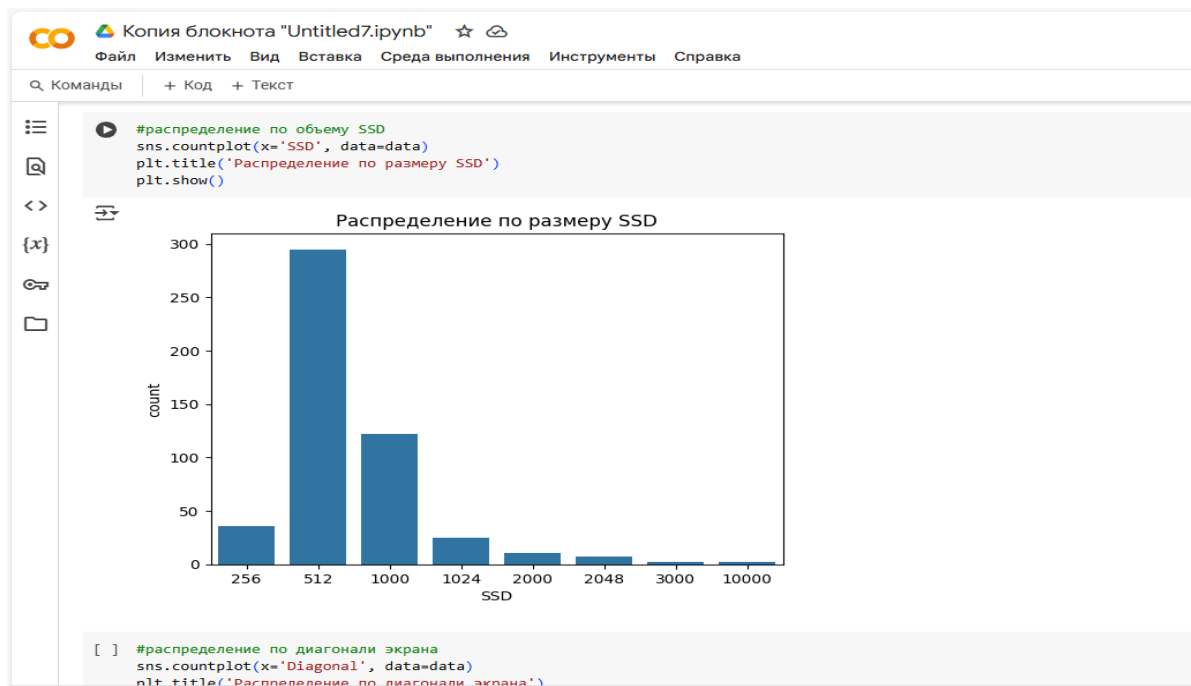


Рисунок 12- Обзор распределения по размеру SSD

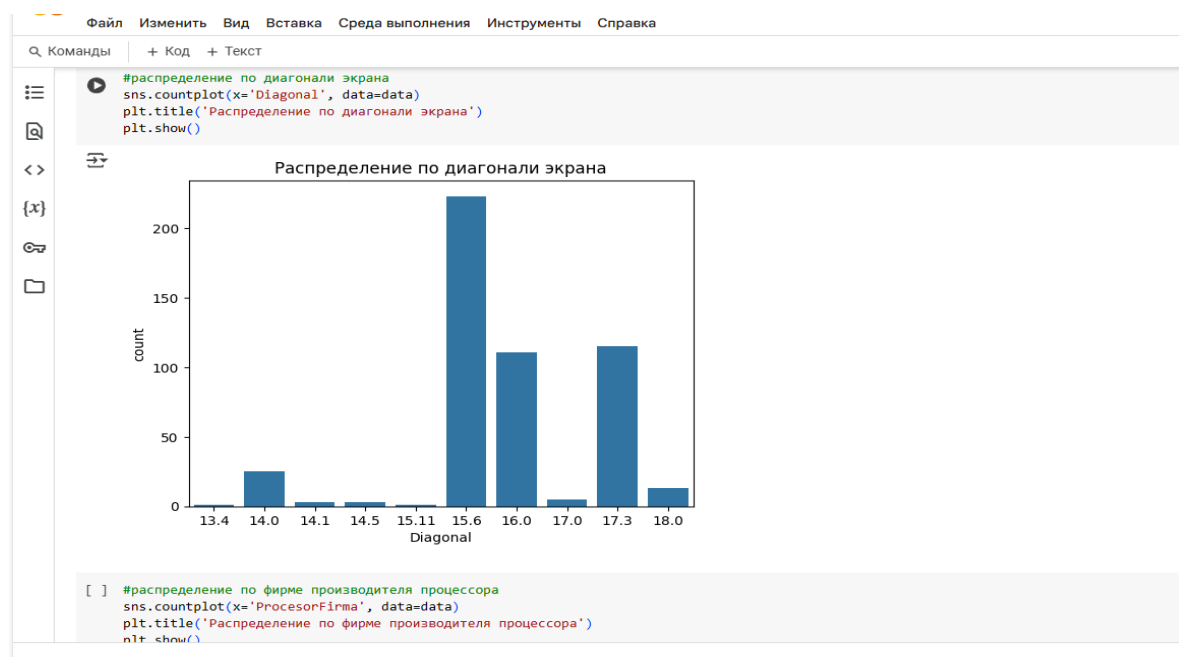


Рисунок 13- Обзор распределения по диагонали экрана

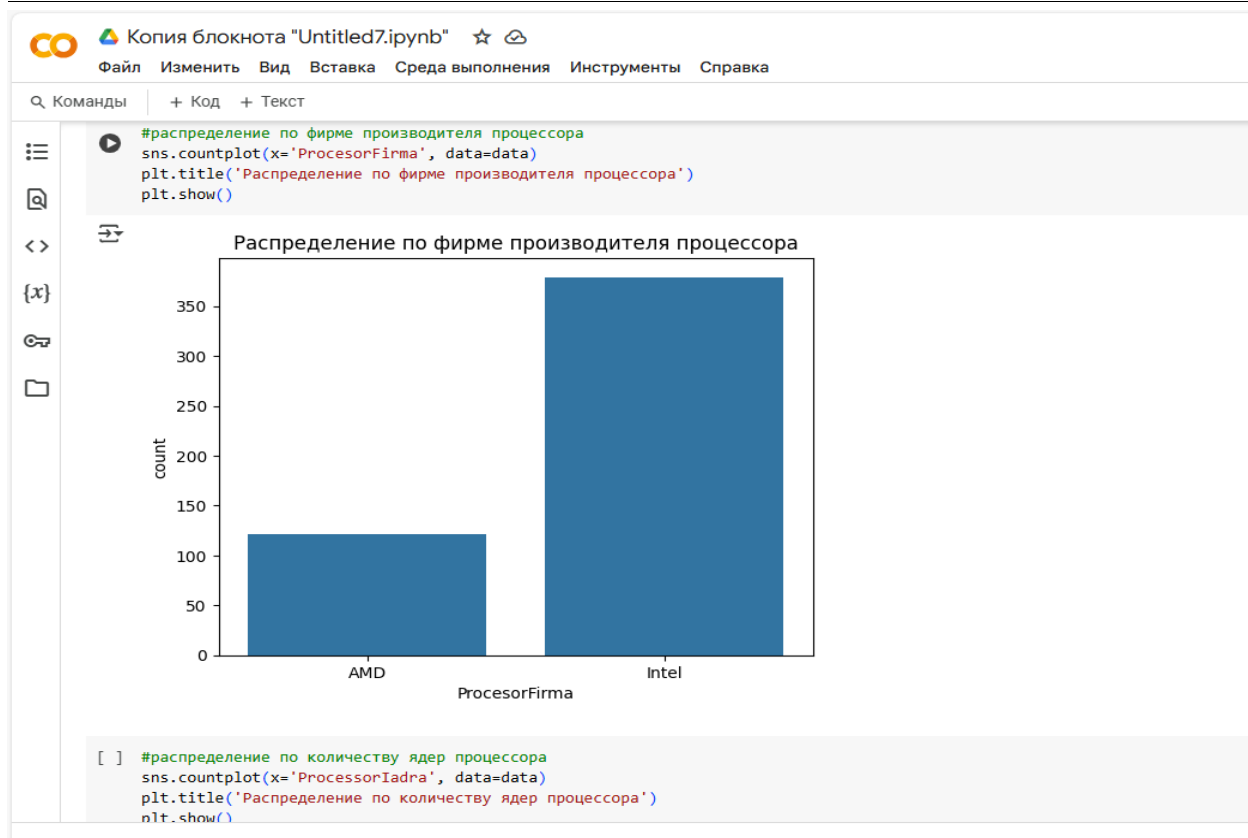


Рисунок 14- Обзор распределения по фирме производителя процессора

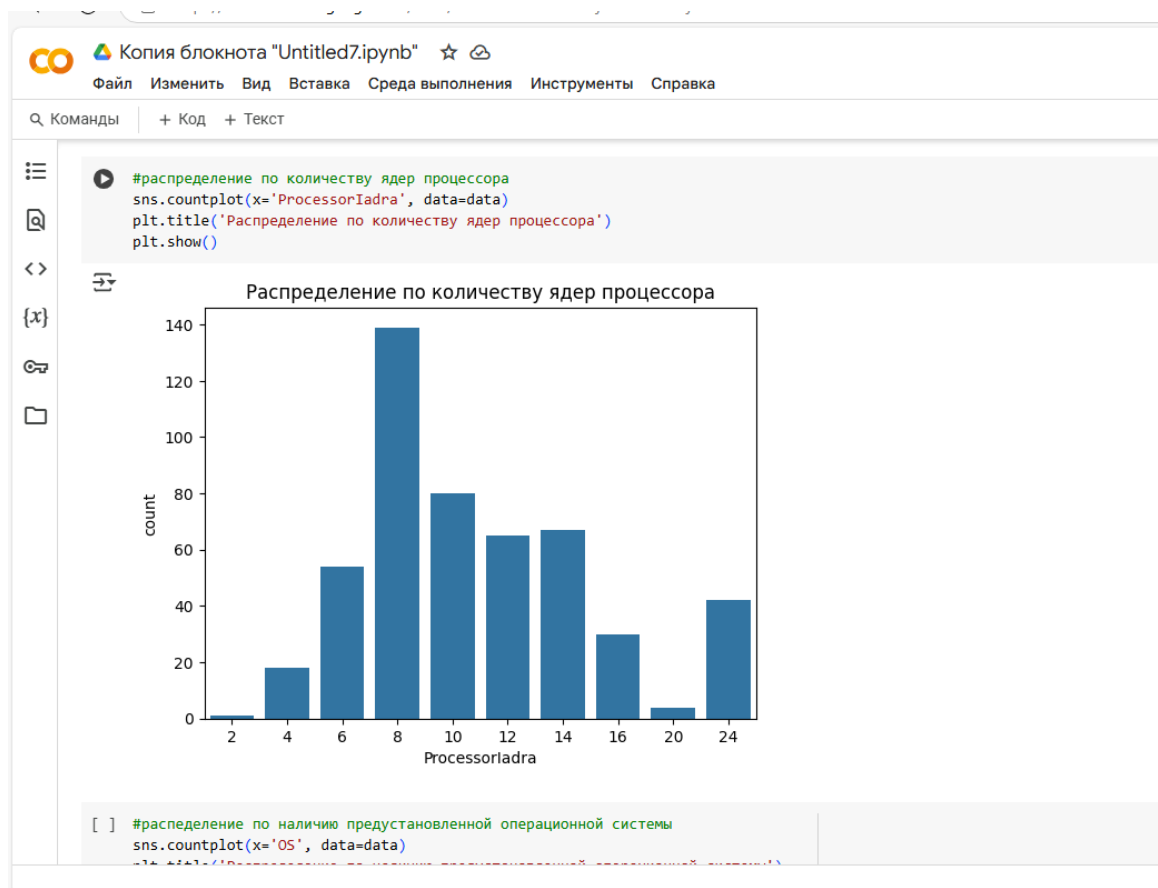


Рисунок 15- Обзор по количеству ядер процессора

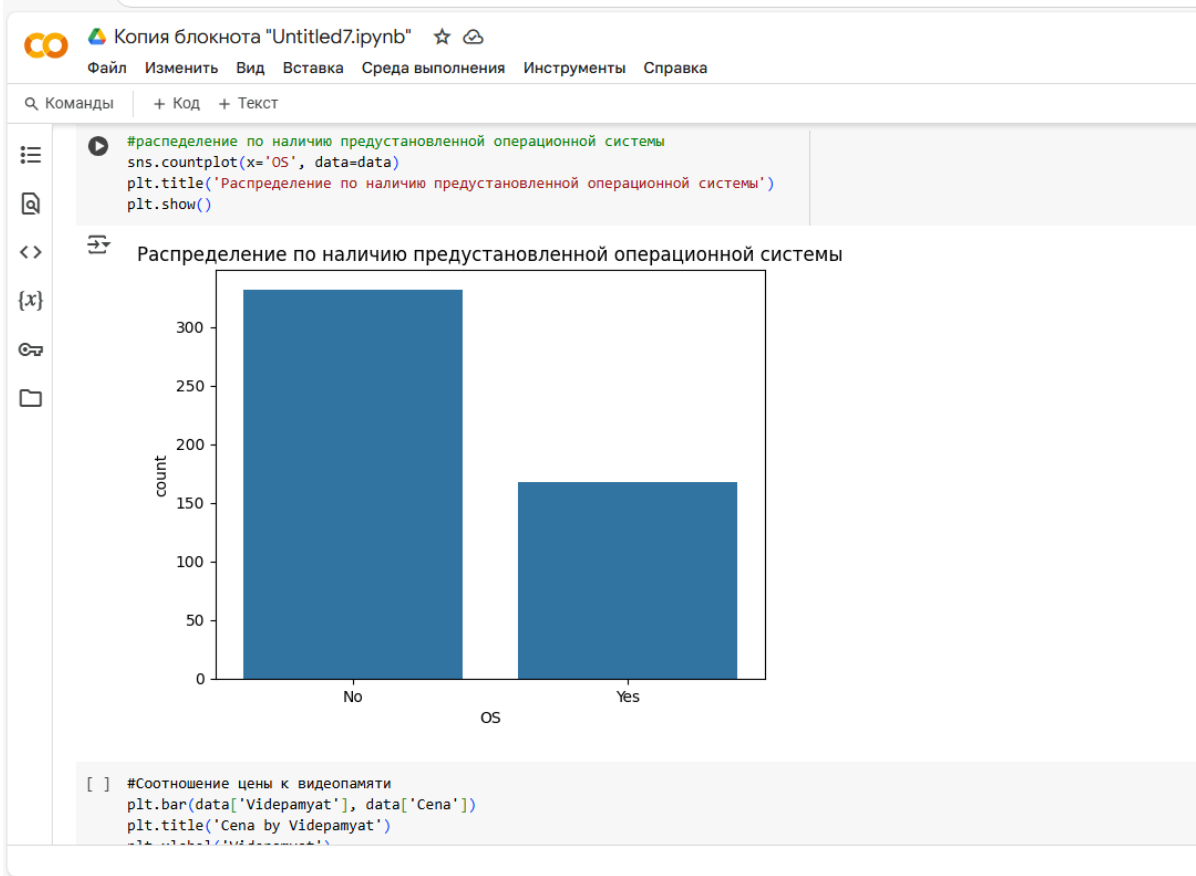


Рисунок 16- Обзор распределения по наличию предустановленной операционной системы

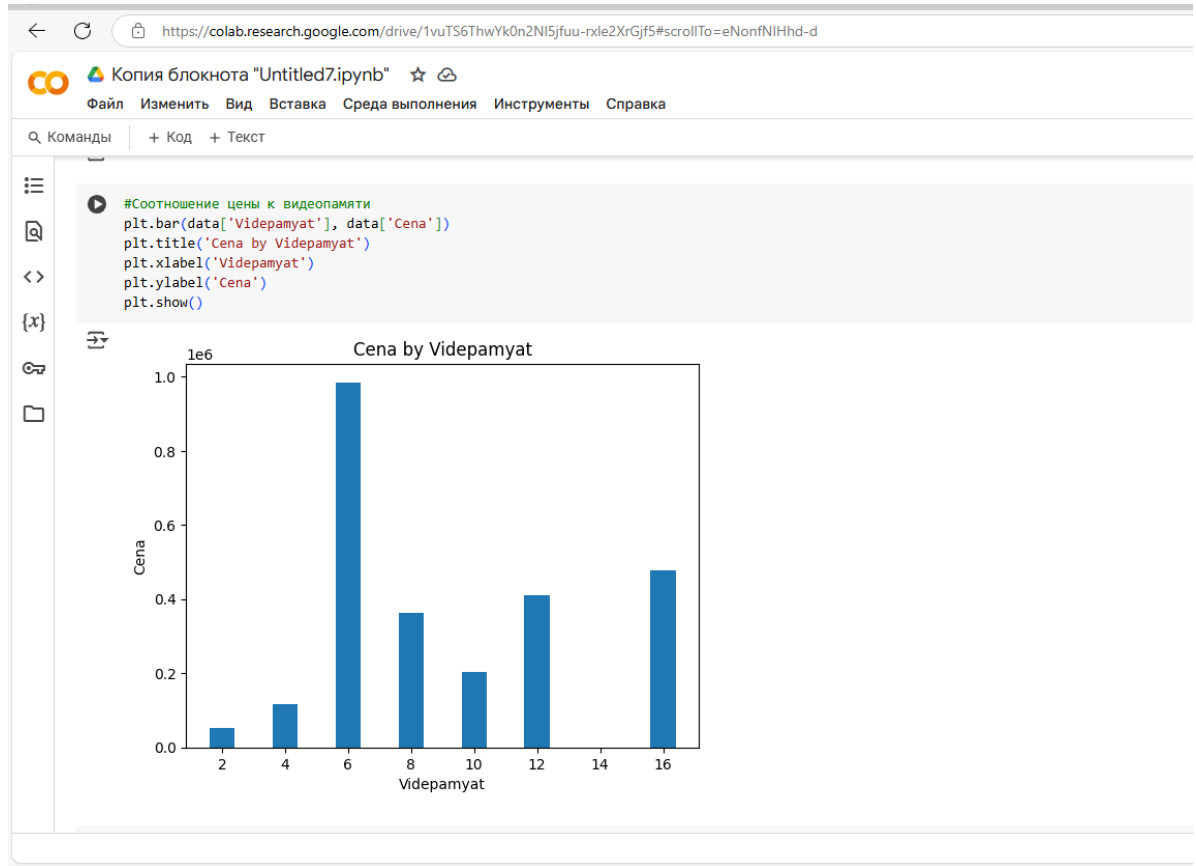


Рисунок 17- Обзор соотношения цены к видеопамяти

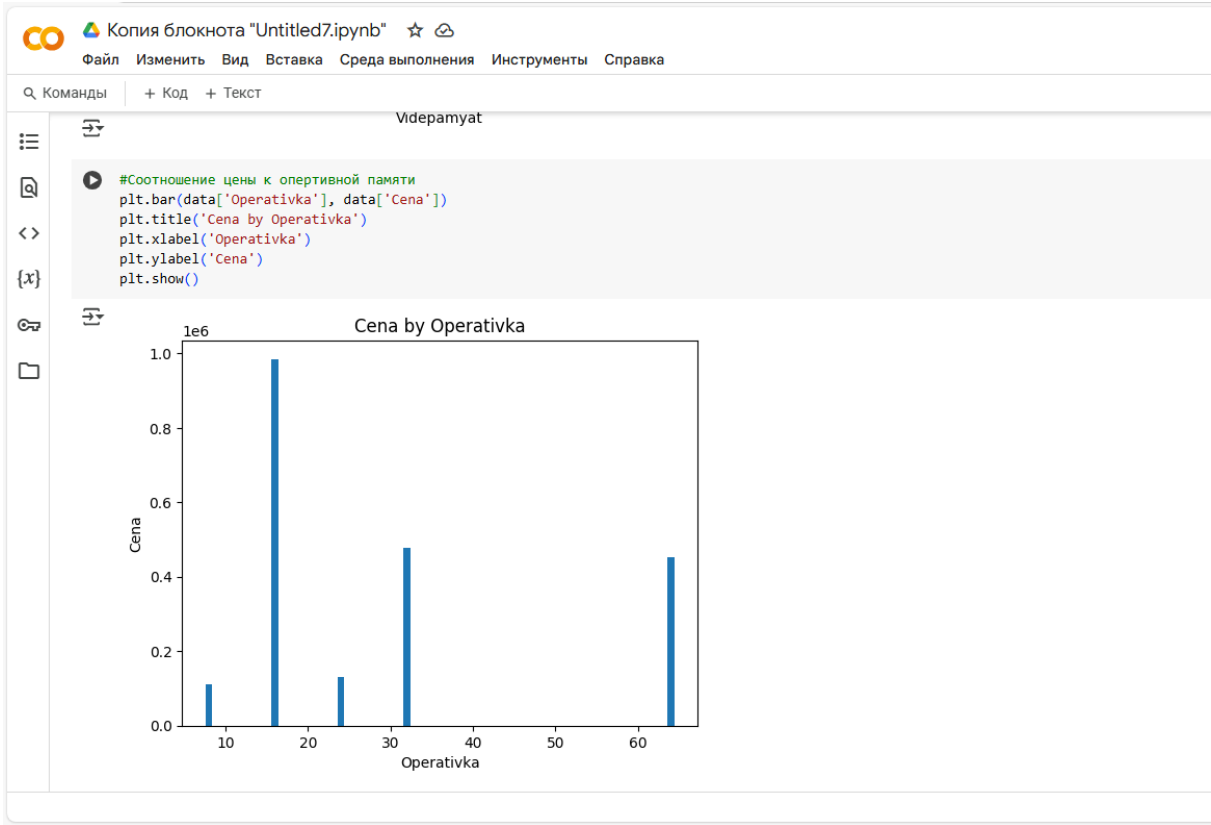


Рисунок 18- Обзор соотношения цены к оперативной памяти



Рисунок 19- Обзор соотношения цены к диагонали экрана

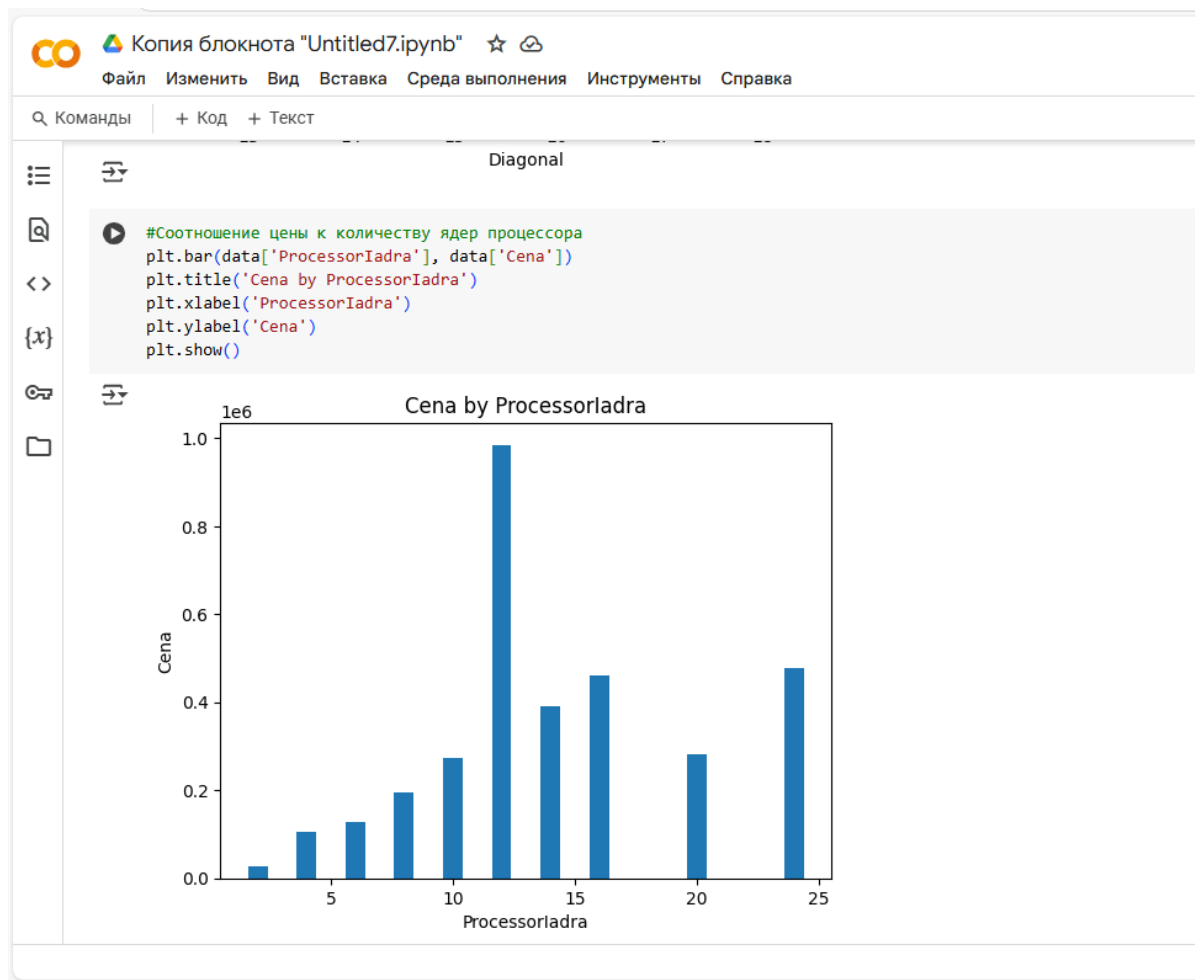


Рисунок 20- Обзор соотношения цены к количеству ядер процессора

Далее построить матрицу корреляции для того чтобы, выявить взаимосвязи между различными переменными набора данных. Это поможет определить, какие факторы могут оказывать влияние друг на друга, что в свою очередь может быть полезно для дальнейшего анализа, прогнозирования или построения моделей. Понимание корреляции между переменными также может указать на возможные зависимости и помочь в принятии обоснованных решений при интерпретации данных (рис.21).

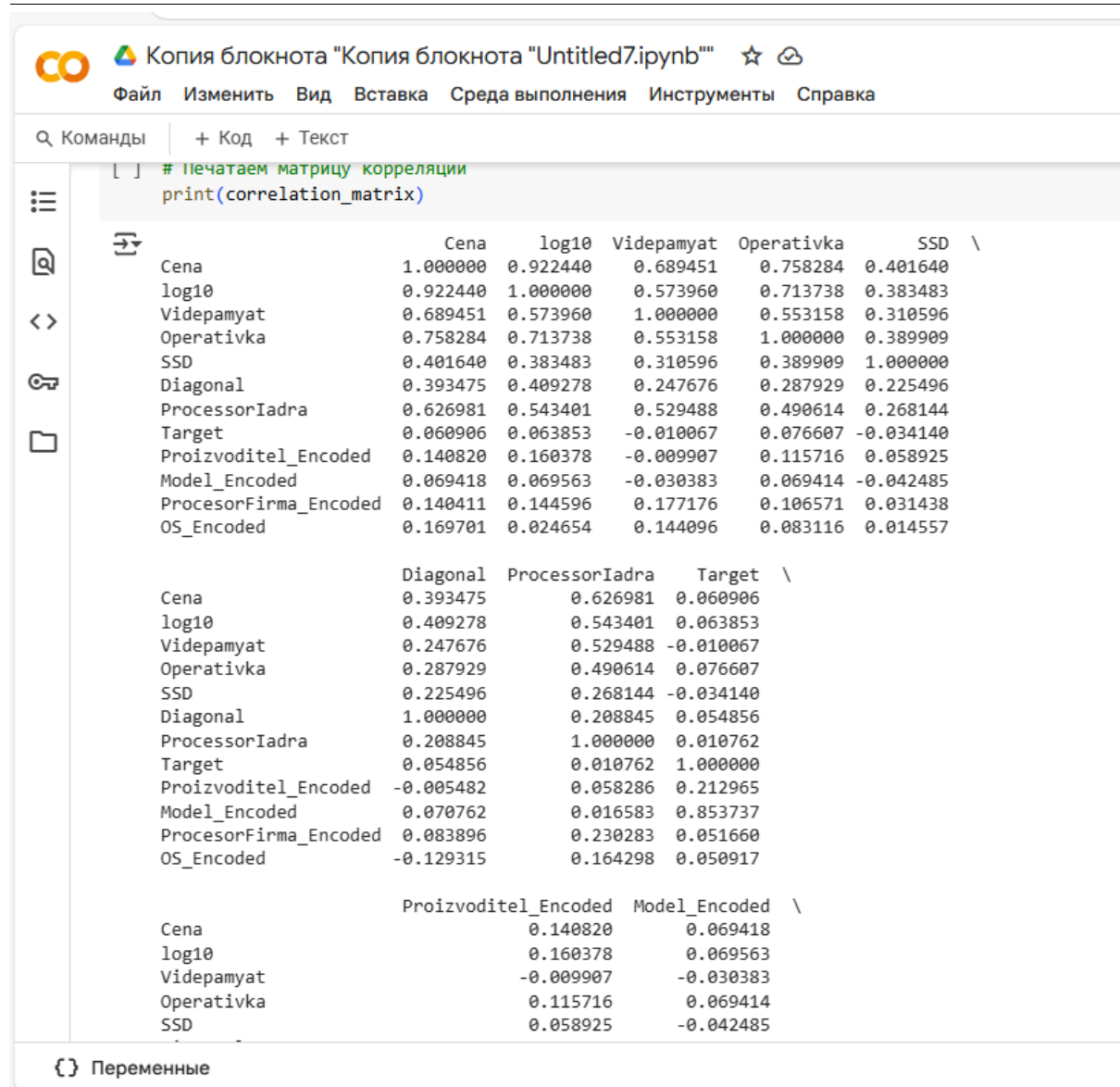


Рисунок 21- Матрица корреляции

В данном датасет, некоторые колонки имеют категориальное значение, применено Target Encoding, для их кодирования в числовые значения (рис. 22).

```

# Применяем Target Encoding для 'Proizvoditel'
mean_proizvoditel = df.groupby('Proizvoditel')['PopularityCoefficient'].mean().to_dict()
df['Proizvoditel_Encoded'] = df['Proizvoditel'].map(mean_proizvoditel).fillna(0)

# Применяем Target Encoding для 'Model'
mean_model = df.groupby('Model')['PopularityCoefficient'].mean().to_dict()
df['Model_Encoded'] = df['Model'].map(mean_model).fillna(0)

# Применяем Target Encoding для 'ProcesorFirma'
mean_procesorfirma = df.groupby('ProcesorFirma')['PopularityCoefficient'].mean().to_dict()
df['ProcesorFirma_Encoded'] = df['ProcesorFirma'].map(mean_procesorfirma).fillna(0)

# Применяем Target Encoding для 'OS'
mean_os = df.groupby('OS')['PopularityCoefficient'].mean().to_dict()
df['OS_Encoded'] = df['OS'].map(mean_os).fillna(0)

# Создаем новую колонку 'efficiency', используя закодированные значения
df['efficiency'] = df[['Proizvoditel_Encoded', 'Model_Encoded', 'ProcesorFirma_Encoded', 'OS_Encoded']].mean(axis=1)

# Проверка наличия колонки 'efficiency' в DataFrame
print(df[['Proizvoditel_Encoded', 'Model_Encoded', 'ProcesorFirma_Encoded', 'OS_Encoded', 'efficiency']].head())

# Вычисление корреляционной матрицы
correlation_matrix = df[['Proizvoditel_Encoded', 'Model_Encoded', 'ProcesorFirma_Encoded', 'OS_Encoded', 'efficiency',
                          'PopularityCoefficient', 'Cena', 'log10', 'Videpamyat', 'Operativka', 'SSD', 'Diagonal',
                          'ProcessorIadra']].corr()

# Построение матрицы корреляции
plt.figure(figsize=(12, 10))
sns.heatmap(correlation_matrix, annot=True, fmt=".2f", cmap="coolwarm", square=True)
plt.title('Матрица корреляции закодированных переменных и других характеристик', fontsize=16)

```

Рисунок 22- Кодирование категориальных данных методом Target Encoding

Далее построить тепловую карту, строится она для визуализации матрицы данных. Она помогает быстро идентифицировать сильные и слабые взаимосвязи, а также закономерности и аномалии в данных. Тепловую карту нужно строить на основе числовых данных, матрицы корреляций, это позволит зрительно охватить сложные зависимости и взаимодействия (рис. 23).

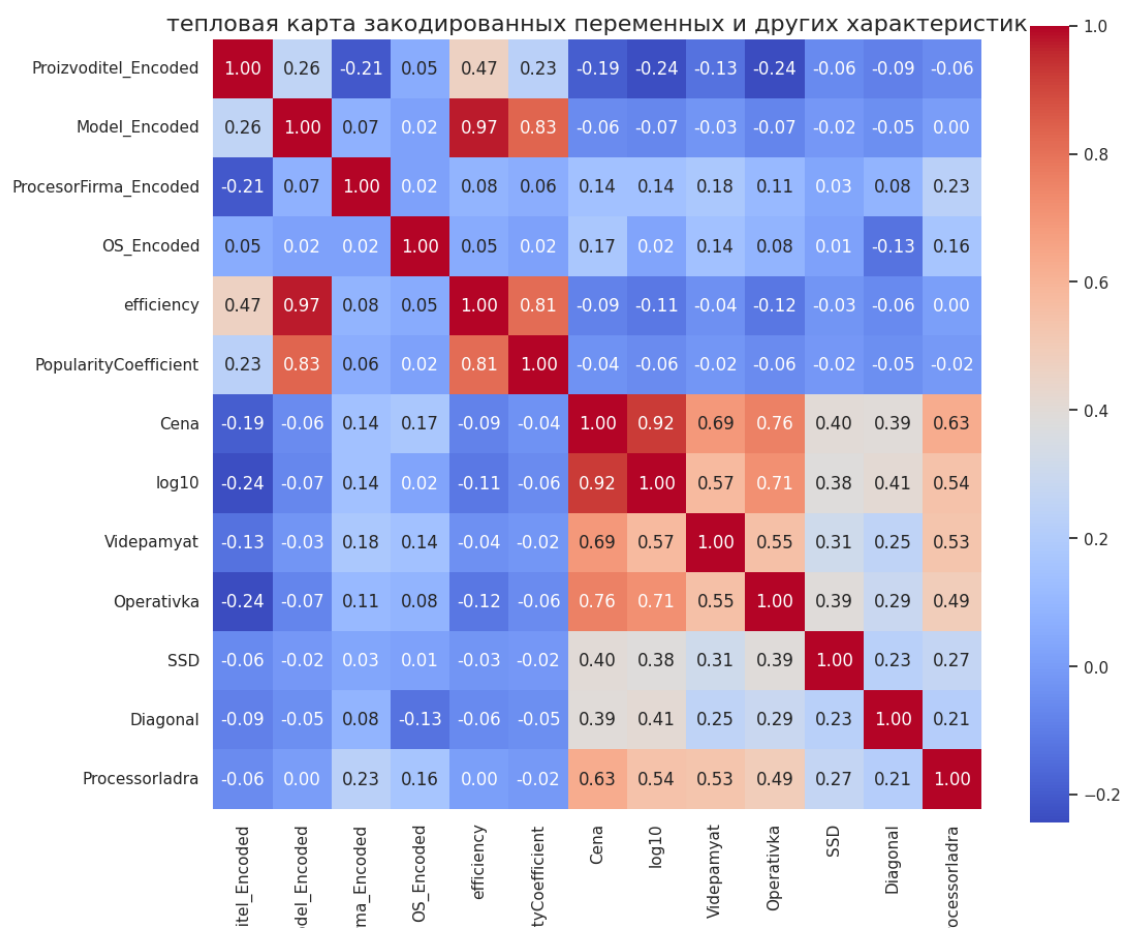
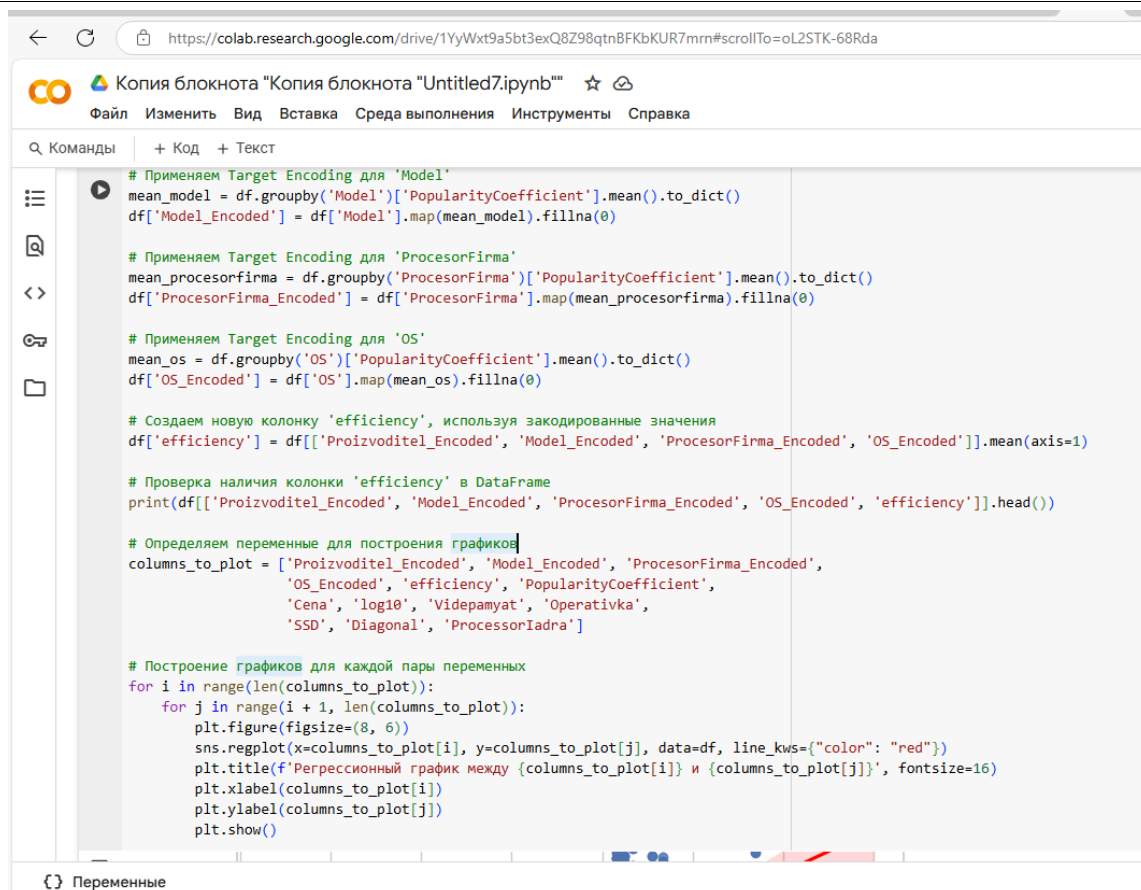


Рисунок 23- Тепловая карта

В приведенном ниже коде используется библиотека Seaborn для построения графиков, и Matplotlib для их настройки и отображения, построены регрессионные графики для каждой пары переменных из списка `columns_to_plot`, такой визуально покажет зависимость одной переменной от другой, и позволит оценить взаимосвязь между ними (рис. 24).



```

# Применяем Target Encoding для 'Model'
mean_model = df.groupby('Model')['PopularityCoefficient'].mean().to_dict()
df['Model_Encoded'] = df['Model'].map(mean_model).fillna(0)

# Применяем Target Encoding для 'ProcessorFirma'
mean_processorfirma = df.groupby('ProcessorFirma')['PopularityCoefficient'].mean().to_dict()
df['ProcessorFirma_Encoded'] = df['ProcessorFirma'].map(mean_processorfirma).fillna(0)

# Применяем Target Encoding для 'OS'
mean_os = df.groupby('OS')['PopularityCoefficient'].mean().to_dict()
df['OS_Encoded'] = df['OS'].map(mean_os).fillna(0)

# Создаем новую колонку 'efficiency', используя закодированные значения
df['efficiency'] = df[['Proizvoditel_Encoded', 'Model_Encoded', 'ProcessorFirma_Encoded', 'OS_Encoded']].mean(axis=1)

# Проверка наличия колонки 'efficiency' в DataFrame
print(df[['Proizvoditel_Encoded', 'Model_Encoded', 'ProcessorFirma_Encoded', 'OS_Encoded', 'efficiency']].head())

# Определяем переменные для построения графиков
columns_to_plot = ['Proizvoditel_Encoded', 'Model_Encoded', 'ProcessorFirma_Encoded',
                  'OS_Encoded', 'efficiency', 'PopularityCoefficient',
                  'Cena', 'log10', 'Videpamyat', 'Operativka',
                  'SSD', 'Diagonal', 'ProcessorIadra']

# Построение графиков для каждой пары переменных
for i in range(len(columns_to_plot)):
    for j in range(i + 1, len(columns_to_plot)):
        plt.figure(figsize=(8, 6))
        sns.regplot(x=columns_to_plot[i], y=columns_to_plot[j], data=df, line_kws={"color": "red"})
        plt.title(f'Регрессионный график между {columns_to_plot[i]} и {columns_to_plot[j]}, fontsize=16)
        plt.xlabel(columns_to_plot[i])
        plt.ylabel(columns_to_plot[j])
        plt.show()

```

Рисунок 24- Обзор кода библиотеки Seaborn

Отобразить графики с максимальной зависимостью (рис. 25), (рис. 26), (рис. 27), (рис. 28), (рис. 29), (рис. 30), (рис. 31), (рис. 32).

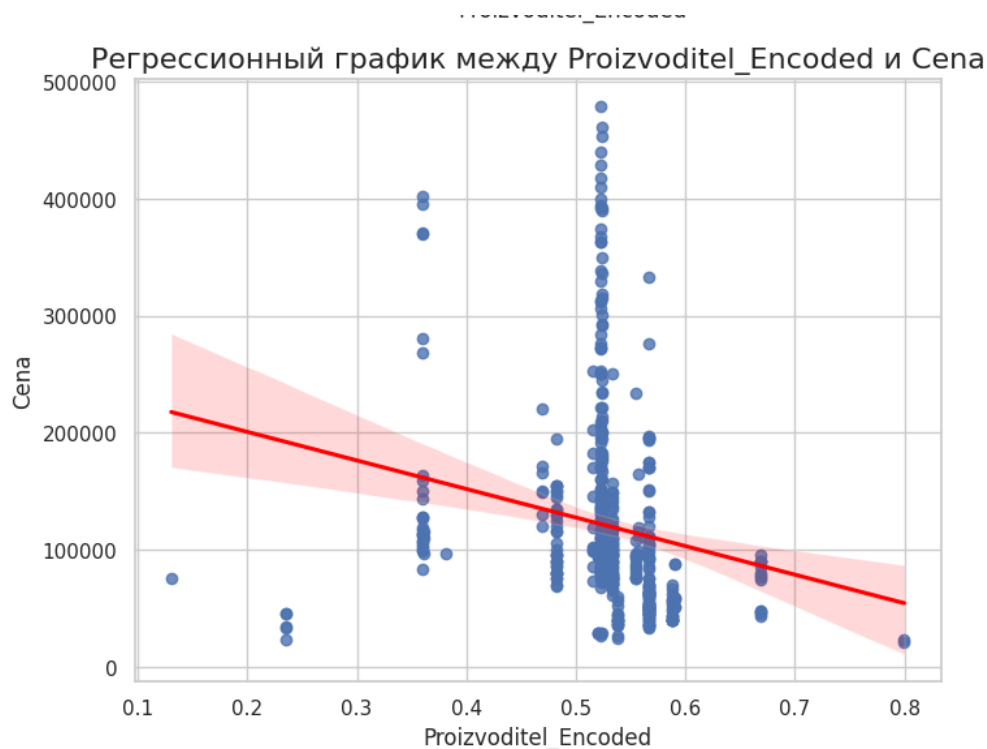


Рисунок 25- Обзор графика зависимости производителя от цены

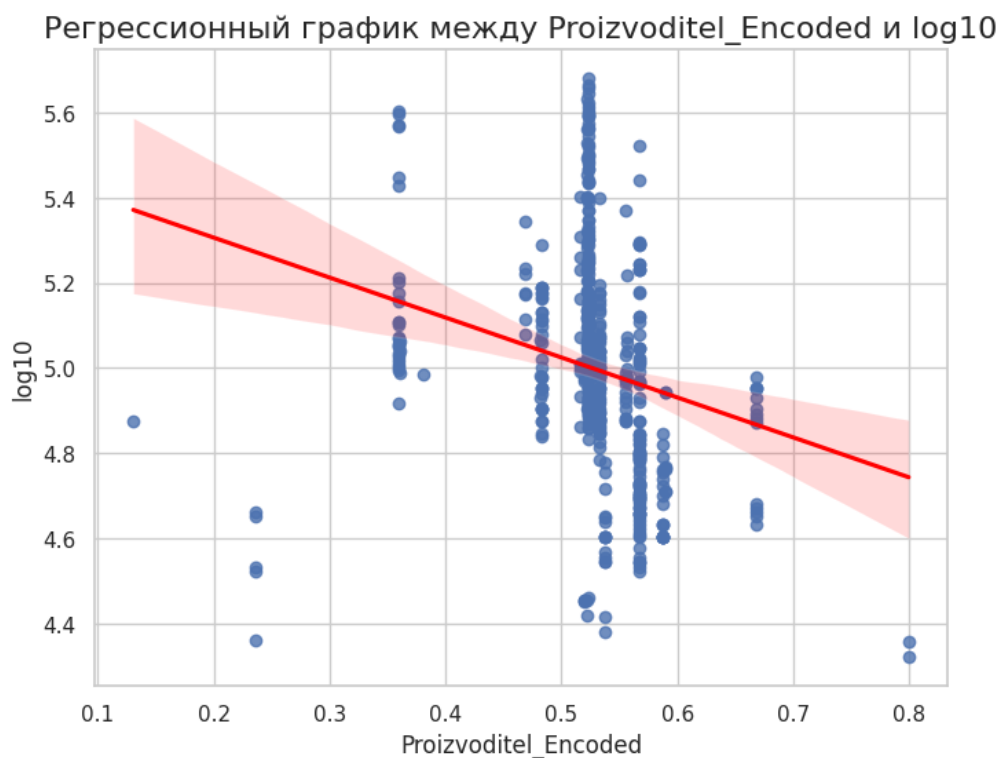


Рисунок 26- Обзор графика зависимости производителя от средней цены



Рисунок 27- Обзор графика зависимости производителя от программного обеспечения

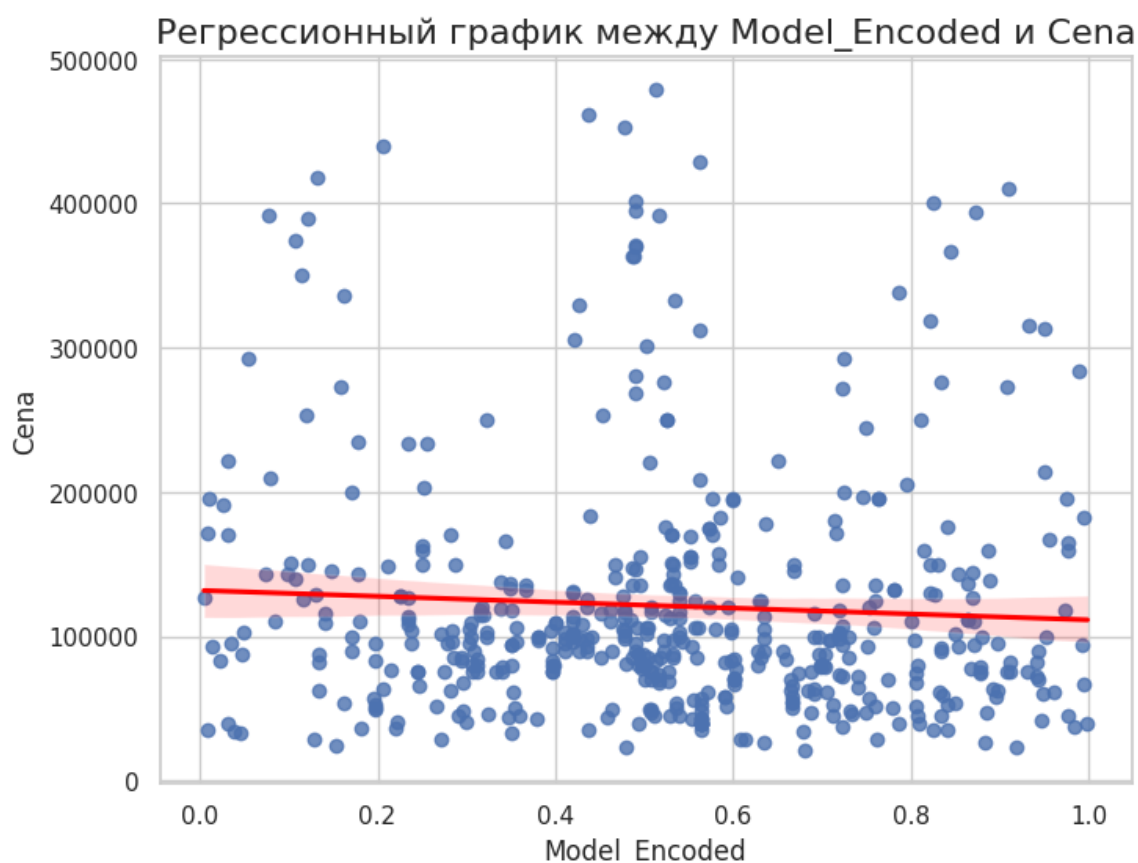


Рисунок 28- Обзор графика зависимости цены от модели

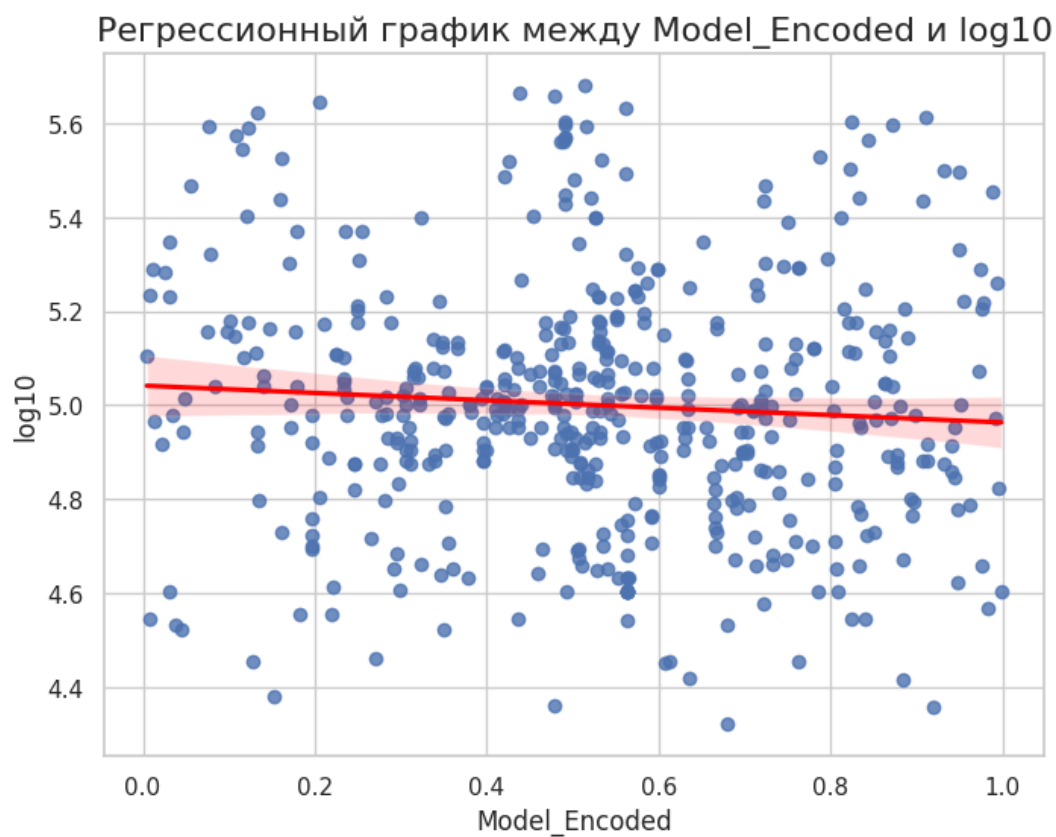


Рисунок 29- Обзор графика зависимости среднего значения цены от модели

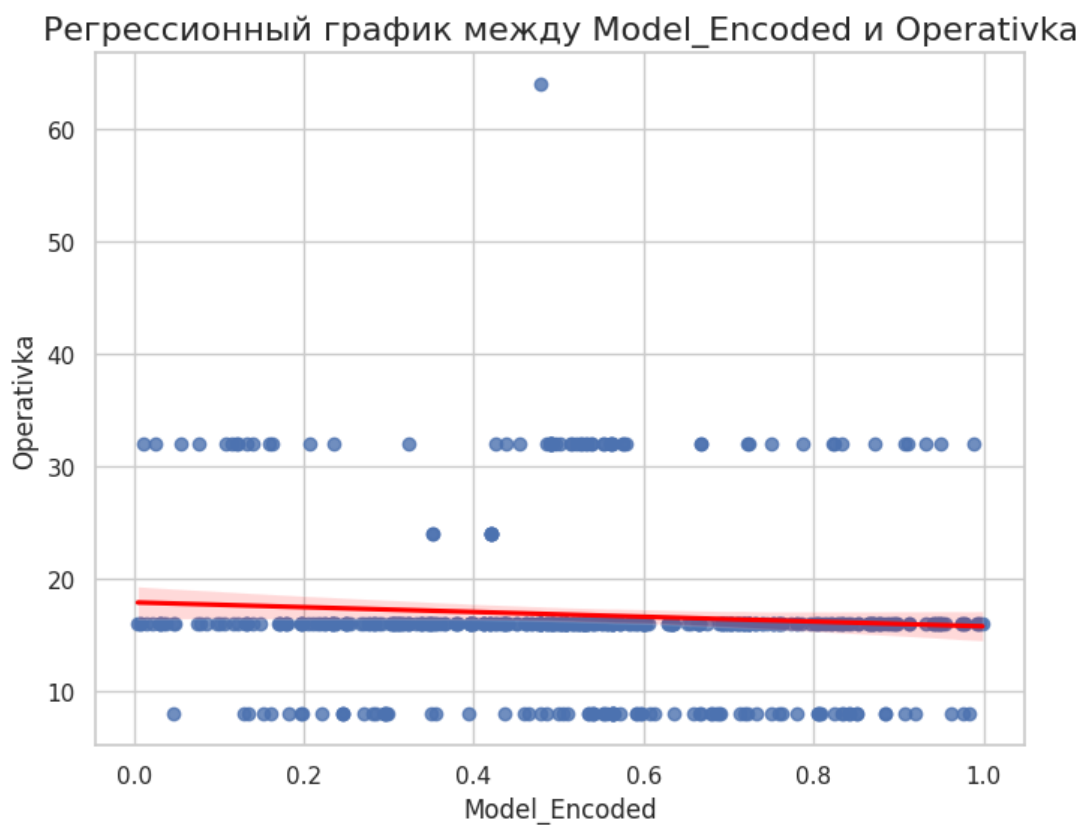


Рисунок 30– Обзор графика зависимости оперативной памяти от модели

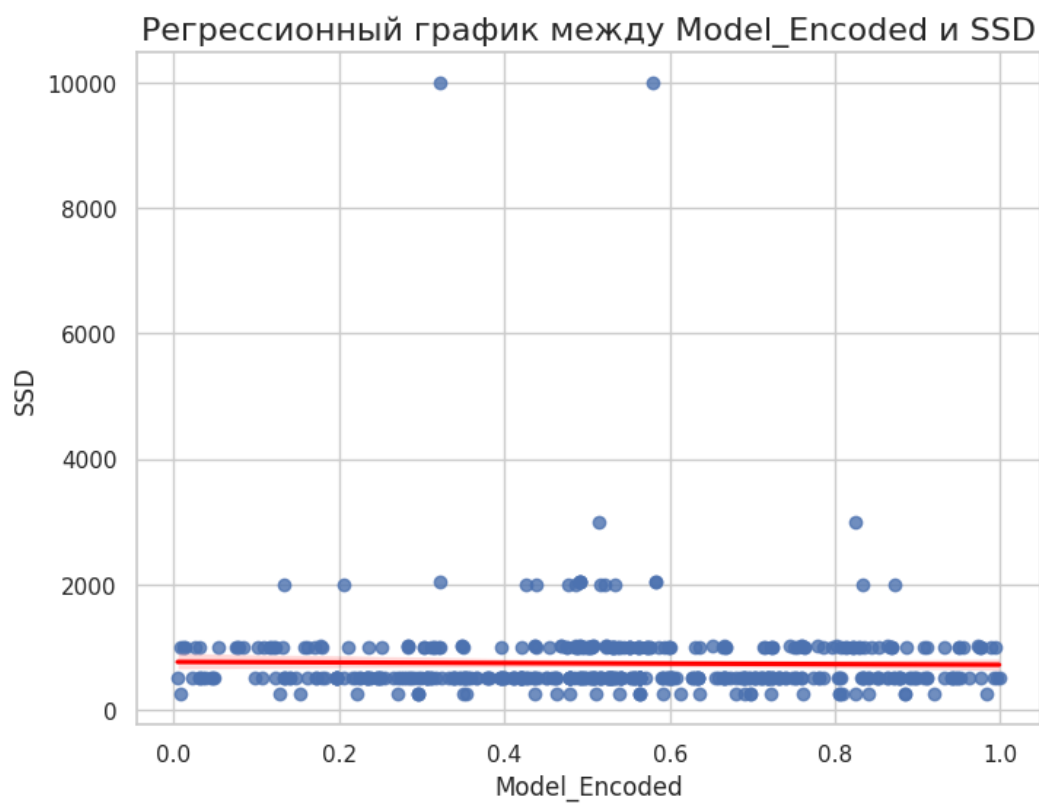


Рисунок 31- Обзор графика зависимости программного обеспечения от модели



Рисунок 32- Обзор графика зависимости процессора от цены

В процессе кодирования, для каждой уникальной категории в определённой переменной колонке `proizvoditel`, `model`, `procesorFirma`, и `os`, созданы новые колонки (фичи) `efficiency` и `popularityCoefficient`, они представляют среднее значение всех закодированных переменных. Это помогает захватить взаимосвязь между несколькими признаками и целевой переменной, что, в свою очередь, способствует повышению производительности моделей машинного обучения. Такой подход необходим для улучшения способности модели улавливать основные закономерности в данных и позволяет значительно повысить её качество (рис. 33), (рис. 34), (рис. 35), (рис.36), (рис. 37).

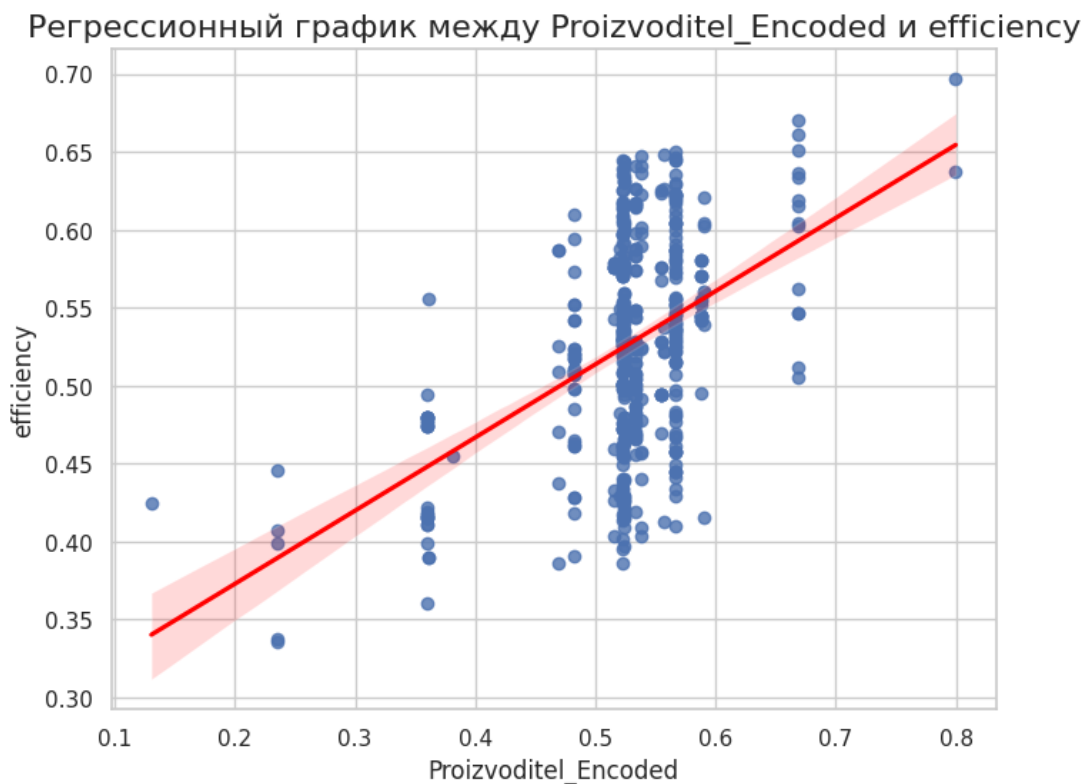


Рисунок 33- Обзор графика зависимости производителя от эффективности

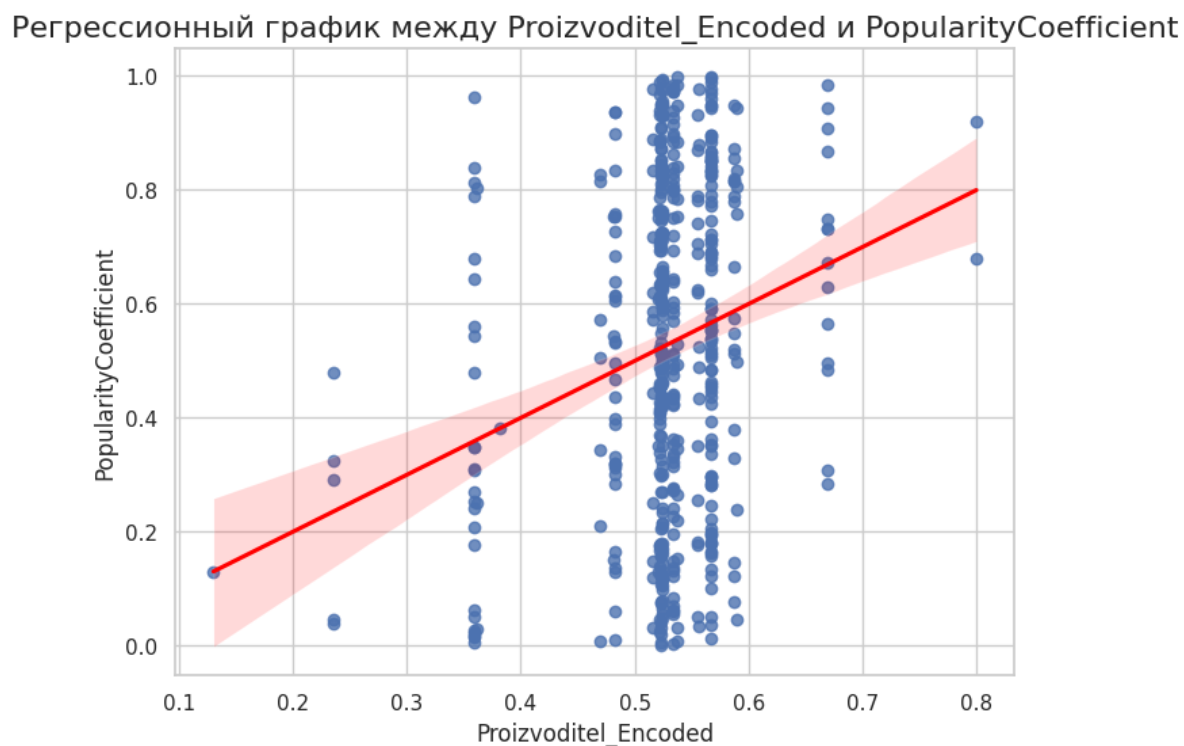


Рисунок 34- Обзор графика зависимости производителя от популярный коэффициент

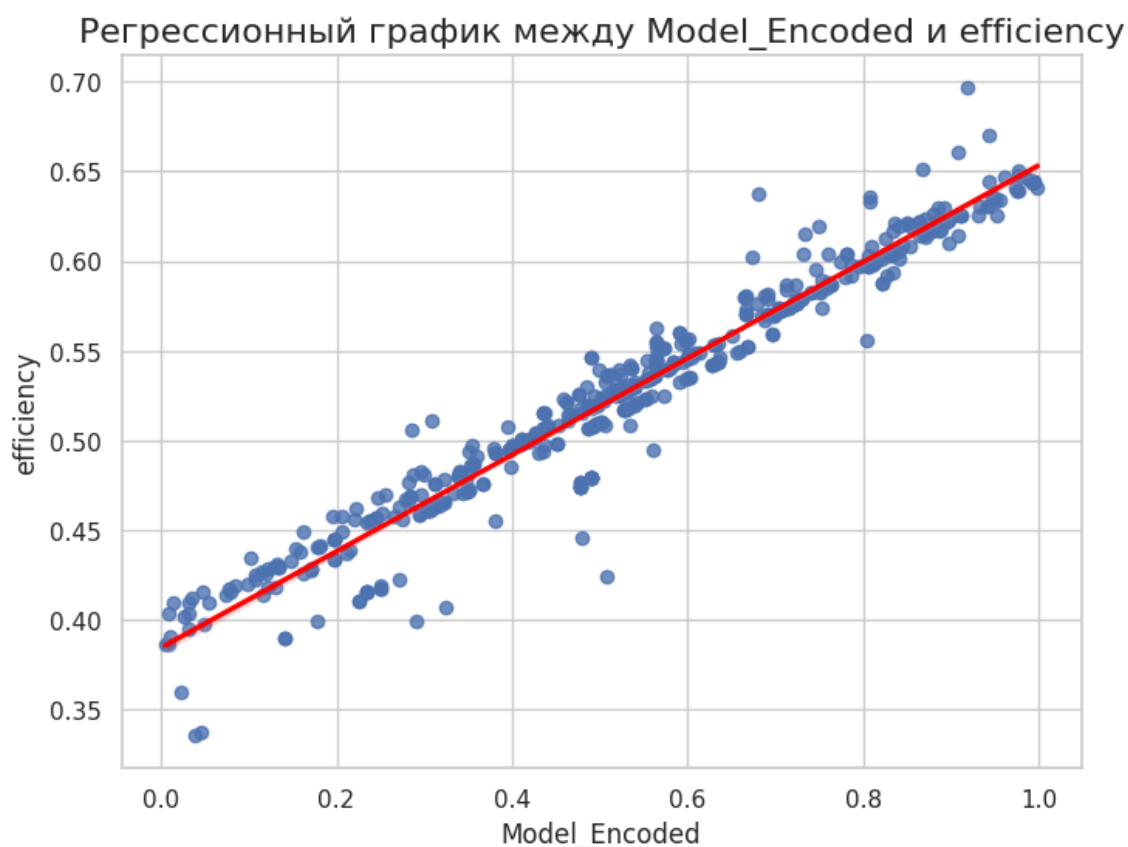


Рисунок 35- Обзор графика зависимости модели от эффективности

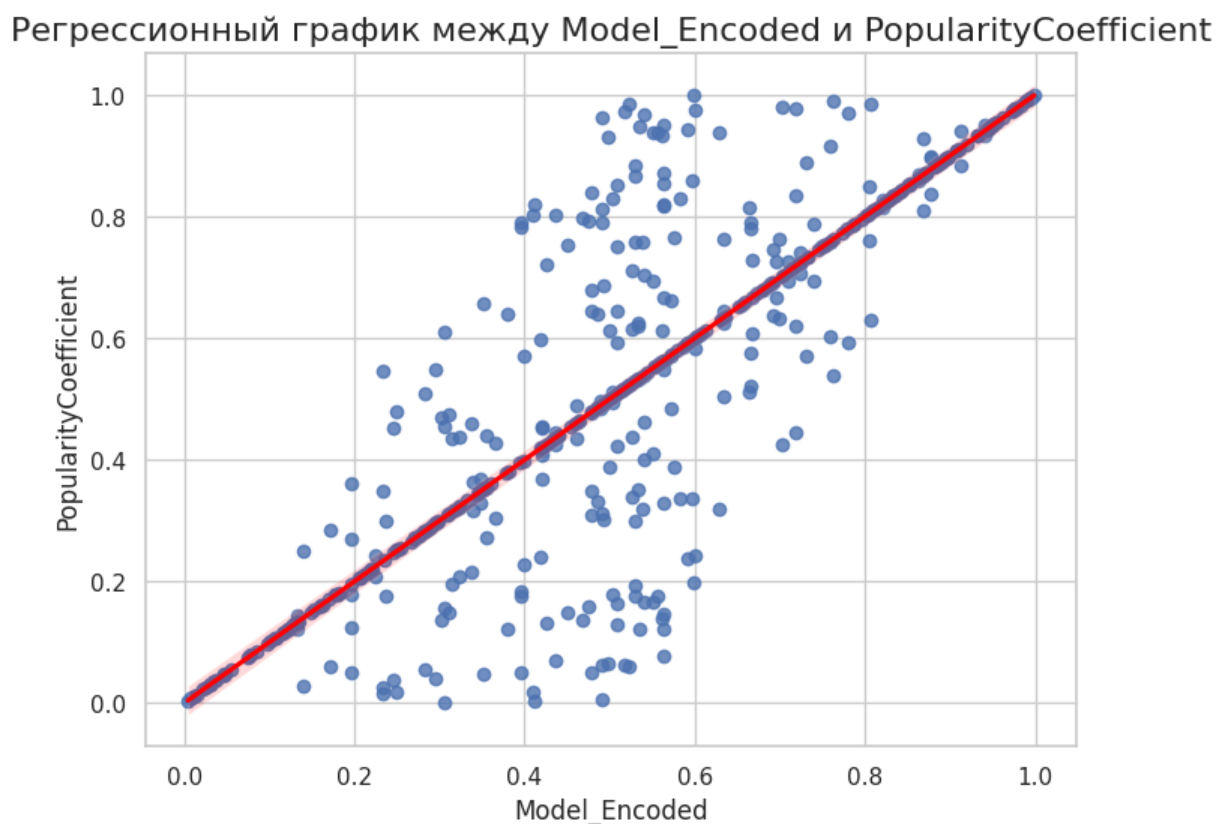


Рисунок 36- Обзор кода графика зависимости модели от популярный коэффициент

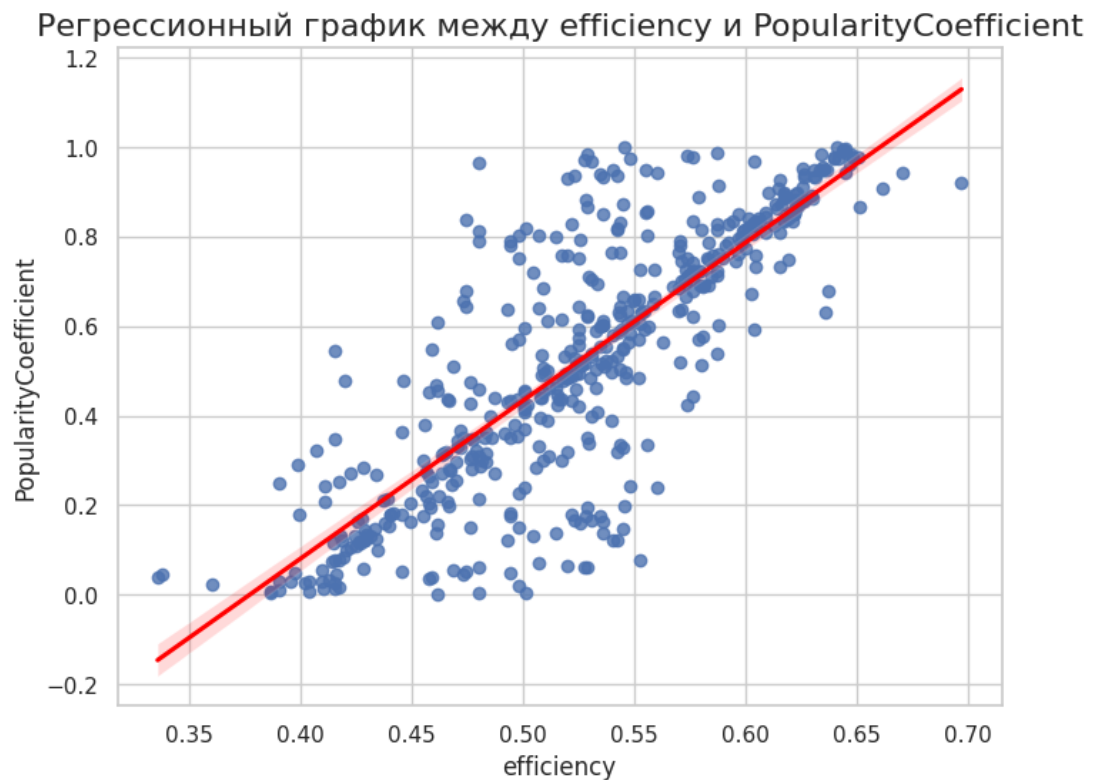


Рисунок 37- Обзор графика зависимости популярный коэффициент от эффективности

Выполнить обучение и оценку моделей, а также вычисление и вывод метрик (рис. 38).

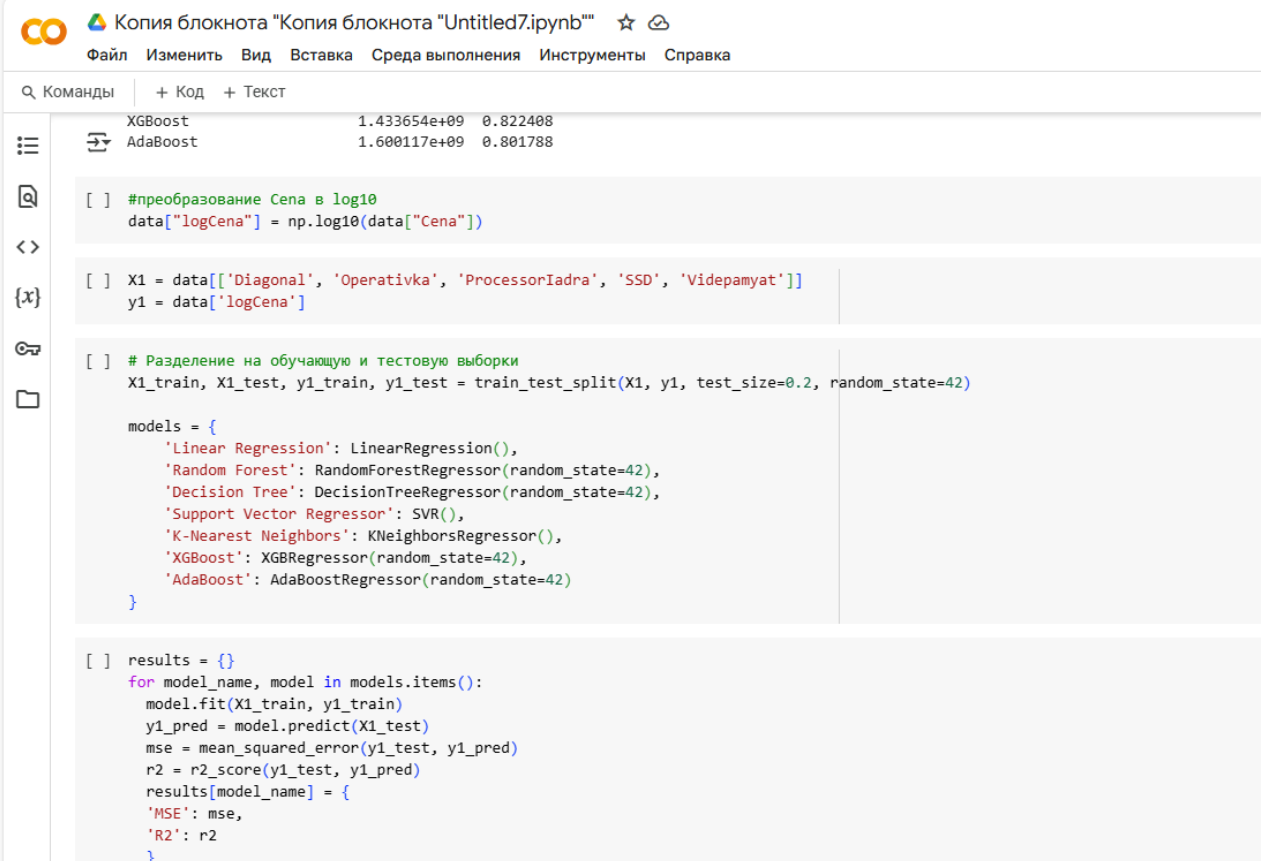
```
[23] results = {}
for model_name, model in models.items():
    model.fit(X_train, y_train)
    y_pred = model.predict(X_test)
    mse = mean_squared_error(y_test, y_pred)
    r2 = r2_score(y_test, y_pred)
    results[model_name] = {
        'MSE': mse,
        'R2': r2
    }
results_df = pd.DataFrame(results).T
print("Results:")
print(results_df)
```

```
Results:
              MSE      R2
Linear Regression  2.089726e+09  0.741139
Random Forest    1.412010e+09  0.825090
Decision Tree     1.815617e+09  0.775093
Support Vector Regressor  9.167508e+09 -0.135610
K-Nearest Neighbors  1.851159e+09  0.770691
XGBoost          1.433654e+09  0.822408
AdaBoost         1.600117e+09  0.801788
```

```
[31] #преобразование Cena в log10
data["logCena"] = np.log10(data["Cena"])
```

Рисунок 38- Обзор метрик MSE, R2

Из полученных данных следует, что модель Random Forest показала наиболее точный результат по метрике R2. Но для того, чтобы точно убедиться в надежности данной модели, необходимо провести еще одно исследование с выводом метрик, преобразовав перед этим категорию «Cena» с использованием $\log_{10}$. Провести разделение на обучающую и тестовую выборки (Рис.39).



Копия блокнота "Копия блокнота "Untitled7.ipynb""

Файл Изменить Вид Вставка Среда выполнения Инструменты Справка

Команды + Код + Текст

	XGBoost		1.433654e+09	0.822408
	AdaBoost		1.600117e+09	0.801788

```
[ ] #преобразование Cena в log10
data["logCena"] = np.log10(data["Cena"])

[ ] X1 = data[['Diagonal', 'Operativka', 'ProcessorIadra', 'SSD', 'Videpamyat']]
y1 = data['logCena']

[ ] # Разделение на обучающую и тестовую выборки
X1_train, X1_test, y1_train, y1_test = train_test_split(X1, y1, test_size=0.2, random_state=42)

models = {
    'Linear Regression': LinearRegression(),
    'Random Forest': RandomForestRegressor(random_state=42),
    'Decision Tree': DecisionTreeRegressor(random_state=42),
    'Support Vector Regressor': SVR(),
    'K-Nearest Neighbors': KNeighborsRegressor(),
    'XGBoost': XGBRegressor(random_state=42),
    'AdaBoost': AdaBoostRegressor(random_state=42)
}

[ ] results = {}
for model_name, model in models.items():
    model.fit(X1_train, y1_train)
    y1_pred = model.predict(X1_test)
    mse = mean_squared_error(y1_test, y1_pred)
    r2 = r2_score(y1_test, y1_pred)
    results[model_name] = {
        'MSE': mse,
        'R2': r2
    }
```

Рисунок 39 – Преобразование в logCena

Далее выполнить вывод метрик с новой переменной (рис.40).

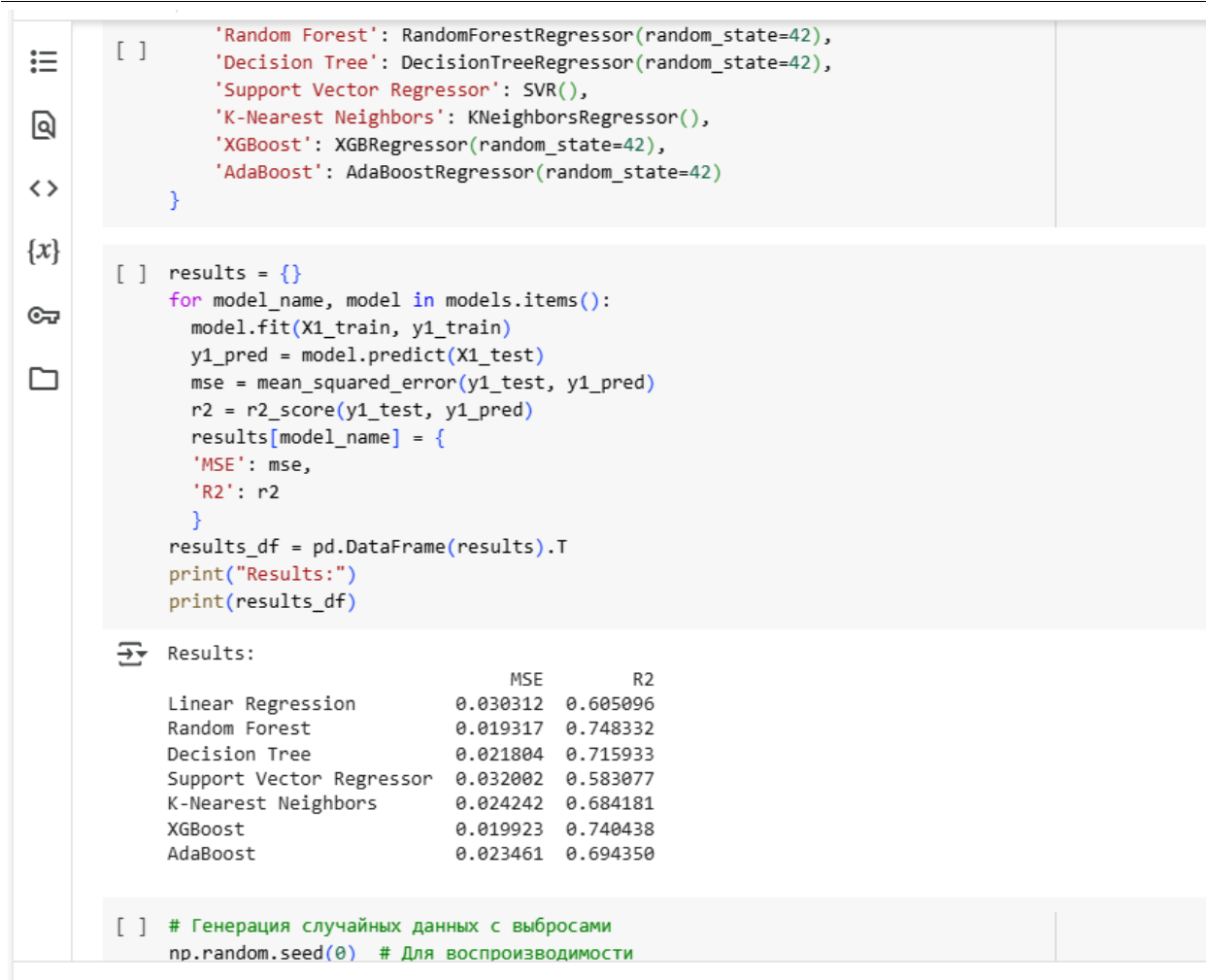


Рисунок 40 – Вывод метрик

На основании данных вывести итоговую таблицу (Таблица 1)

Таблица 1. Результаты

Метод	Степень разброса Получившихся моделей (%) Обычная цена	Степень разброса Получившихся моделей (%) Цена лог10
Linear Regression	74,2%	60,5%
Random Forest	82,6%	74,9%
Decision Tree	77,6%	71,6%
Support Vector Regressor	-13,6%	58,4%
K- Nearest Neighbors	77,1%	68,5%
XGBoost	82,3%	74,1%
AdaBoost	80,2%	69,5%

4 Выводы

В результате проведенного анализа и оценки двух основных метрик, можно сделать вывод, что модель Random Forest показала наивысшие результаты в прогнозировании обычной цены на ноутбуки. Этот результат подчеркивает эффективность данной модели в задачах, связанных с предсказанием цен, и открывают перспективы для дальнейших исследований в этой области.

Библиографический список

1. Маковеев А.А., Сафонов М.И., Ковалёв А.Г., Федченко К.А. Парсинг-искусство извлечения данных // Электронные средства и системы управления. Материалы докладов международной научно- практической конференции. 2024. №1-1. С. 281-284. URL: <https://www.elibrary.ru/item.asp?id=80306600>
2. Кононов В.А. Парсинг сайтов. Парсер на языке Python // Интернаука. 2020. №29. С. 9-10. URL: <https://www.elibrary.ru/item.asp?id=43868703>
3. Волокитина Т.С. Анализ возможностей Google Colab // Современные научные исследования и инновации. 2020. №12. С. 1. URL: <https://www.elibrary.ru/item.asp?id=44477003>
4. Клячкин В.Н., Ковальногов В.Н., Шеркунов В.В., Хуссейн Мохамед Сравнительный анализ методов построения математических моделей функционирования объекта с применением машинного обучения // Программные продукты и системы. 2023. №2. С. 189-195. URL: <https://www.elibrary.ru/item.asp?id=54262529>
5. Дагаев А.В. Методы анализа данных в машинном обучении // Экономический потенциал и корпоративная социальная ответственность. 2024. С. 31-39. URL: <https://www.elibrary.ru/item.asp?id=68621474>