

Интеллектуальный анализ данных цен процессоры

Ёлгин Роман Николаевич

Приамурский государственный университет имени Шолом-Алейхема

Студент

Аннотация

В статье рассматривается задача прогнозирования цен на процессоры с использованием методов машинного обучения. Авторы применяют различные алгоритмы, включая случайный лес (Random Forest), линейную регрессию (Linear Regression), XGBoost) и другие, сравнивая их эффективность на реальных данных. Особое внимание уделяется выбору значимых признаков, обработке данных и оценке точности моделей с помощью метрик (MAE, RMSE, R^2). Результаты исследования демонстрируют, какие методы лучше подходят для прогнозирования цен и какие факторы наиболее сильно влияют на стоимость процессоров.

Ключевые слова: прогнозирование цен, машинное обучение, случайный лес, линейная регрессия, XGBoost, KNeighborsRegressor, DecisionTreeRegressor, процессоры.

Intelligent Analysis of CPU Price Data

Yolgin Roman Nikolaevich

Sholom-Aleichem Priamursky State University

Student

Abstract

The article explores the task of predicting CPU prices using machine learning methods. The authors apply various algorithms, including Random Forest, Linear Regression, Gradient Boosting, and others, comparing their effectiveness on real-world data. Special attention is given to feature selection, data preprocessing, and model accuracy evaluation using metrics such as MAE, RMSE, and R^2 . The study's results demonstrate which methods are best suited for price forecasting and which factors most significantly influence CPU costs.

Keywords: price prediction, machine learning, Random Forest, Linear Regression, XGBoost, CPU.

1 Введение

1.1 Актуальность

Прогнозирование цен на процессоры имеет важное значение для потребителей и производителей, позволяя оптимизировать закупки и стратегии ценообразования. Использование современных методов машинного обучения, таких как XGBoost, случайный лес, линейная регрессия и другие,

повышает точность предсказаний и помогает выявить ключевые факторы, влияющие на стоимость.

1.2 Обзор исследований

В работе А.А. Киселёв, Э.Х. Габдуллин, П.А. Москаленко и Н.Л. Леонова представили практическое применение методов машинного обучения в метеорологии. Разработанная ими система искусственного интеллекта позволяет с высокой точностью прогнозировать погодные условия на основе анализа больших массивов данных [1].

Исследование Д.Р. Низина, Т.А. Низиной, И.П. Спирина и А.С. Станкевича посвящено созданию прогностической модели для определения температуры поверхности полимерных материалов в естественных климатических условиях. Авторы продемонстрировали эффективность машинного обучения для решения подобных инженерных задач [2].

В статье [3] С.А. Иванов проведен детальный анализ стратегических и тактических задач, решаемых методами анализа данных. Особое внимание уделено различиям в подходах к классификации задач и современным платформам для обработки информации в условиях неопределенности [3].

Д.М. Магамедова и А.З. Магамедова в работе систематизировали различные типы обучающих алгоритмов, выделив ключевые факторы, определяющие выбор конкретного метода машинного обучения в зависимости от характера решаемой задачи [4].

Исследование А.В. Сидора, Н.Л. Давидёнок и Е.В. Соловей раскрывает особенности применения технологий интеллектуального анализа данных. Авторы подробно описывают инструментарий для реализации таких методов и перспективные направления их развития [5].

В работе А.А. Овсянниковой и Д.В. Домашовой рассмотрено применение нейросетевых моделей для прогнозирования исполнения государственных контрактов в трубной промышленности. Разработанное авторами Python-приложение позволяет выявлять контракты с повышенным риском неисполнения [6].

А.В. Винниченко и Ю.П. Истомина в исследовании предложили комплексную методику оценки качества прогностических моделей. Используя статистические критерии (альфа Кронбаха, t-критерий Стьюдента и др.), авторы провели сравнительный анализ эффективности различных алгоритмов машинного обучения для прогнозирования показателей инновационной деятельности [7].

1.3 Цель исследования

Цель исследования – разработать и применить регрессионные модели машинного обучения для прогнозирования цен на процессоры.

2 Материалы и методы

В исследовании использовался датасет с техническими характеристиками процессоров, обработанный в Google Colab с помощью

библиотек Python. Для прогнозирования цен применялись модели линейной регрессии, случайного леса, XGBoost, KNeighborsRegressor и DecisionTreeRegressor, оцененные по метрикам MAE, MSE, RMSE и R^2 после предварительной обработки данных и оптимизации гиперпараметров.

3 Результаты и обсуждения

3.1 Подготовка датасета

Датасет был взят с сайта Kaggle.com. Его содержимое включает столбцы: name, price, tdp, date, socket, category, speed, turbo, cpuCount, cores, rank, samples (рис. 1).

name	price	tdp	date	socket	category	speed	turbo	cpuCount	cores	logicals	rank	samples
AMD A10 PRO-7850B APU	148,02	95,00	01.11.2014	FM2+	Desktop	3700,00	4000.0	1	2	2	1648	19
AMD A10-4600M APU	78,02	35,00	01.04.2012	FS1r2	Laptop	2300,00	3200.0	1	2	2	2305	1056
AMD A10-5700 APU	59,95	65,00	01.06.2012	FM2	Desktop	3400,00	4000.0	1	2	2	1889	629
AMD A10-5750M APU	94,02	35,00	01.05.2013	FS1r2	Laptop	2500,00	3500.0	1	4	1	2282	1035
AMD A10-5800B APU	44,99	100,00	01.07.2013	FM2	Desktop	3800,00	4200.0	1	2	2	1823	36
AMD A10-5800K APU	46,07	100,00	01.10.2012	FM2	Desktop	3800,00	4200.0	1	2	2	1812	1751
AMD A10-6700 APU	39,59	65,00	01.06.2013	FM2	Desktop	3700,00	4300.0	1	4	1	1758	707
AMD A10-6700T APU	147,61	45,00	01.06.2014	FM2	Desktop	2500,00	3500.0	1	2	2	2079	18
AMD A10-6790K APU	36,72	100,00	01.12.2013	FM2	Desktop	4000,00	4300.0	1	2	2	1763	116
AMD A10-6800B APU	128,02	100,00	01.02.2014	FM2	Desktop	4100,00	4400.0	1	2	2	1820	18
AMD A10-6800K APU	53,99	100,00	01.06.2013	FM2	Desktop	4100,00	4400.0	1	2	2	1736	930
AMD A10-7700K APU	32,81	95,00	01.01.2014	FM2+	Desktop	3400,00	3800.0	1	2	2	1744	597
AMD A10-7800 APU	37,68	65,00	01.06.2014	FM2+	Desktop	3500,00	3900.0	1	2	2	1737	726
AMD A10-7850K APU	56,05	95,00	01.01.2014	FM2+	Desktop	3700,00	4000.0	1	2	2	1665	1383
AMD A10-7860K	57,17	65,00	01.03.2016	FM2+	Desktop	3600,00	4000.0	1	2	2	1678	430
AMD A10-7870K	65,92	95,00	01.06.2015	FM2+	Desktop	3900,00	4100.0	1	2	2	1644	634
AMD A10-7890K	139,97	95,00	01.04.2016	FM2+	Desktop	4100,00	4300.0	1	2	2	1628	247
AMD A10-9700	66,86	65,00	01.12.2016	AM4	Desktop	3500,00	3800.0	1	4	1	1617	271
AMD A10-9700E	91,50	35,00	01.11.2017	AM4	Desktop	3000,00	3500.0	1	4	1	1818	17
AMD A12-9800	101,99	65,00	01.10.2016	AM4	Desktop	3800,00	4200.0	1	2	2	1622	141
AMD A12-9800E	670,15	35,00	01.09.2017	AM4	Desktop	3100,00	3800.0	1	4	1	1640	25
AMD A4-3300M APU	29,99	65,00	01.07.2011	FM1	Laptop	1900,00	2500.0	1	2	1	3286	265

Рисунок 1. Данные датасета

Датасет можно скачать по ссылке:

<https://docs.google.com/spreadsheets/d/18IefVjySQMNId5R9U0mquNbLFmEYlrwm/edit?usp=sharing&ouid=115906928119392754936&rtpof=true&sd=true>

3.2 Анализ данных

Для начала работы с датасетом были подключены необходимые библиотеки и модели (рис.2).

```
[15] # Импорт необходимых библиотек и моделей
import numpy as np
import pandas as pd
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.ensemble import RandomForestRegressor, GradientBoostingRegressor
from sklearn.tree import DecisionTreeRegressor
from sklearn.neighbors import KNeighborsRegressor
from sklearn.metrics import mean_squared_error, r2_score, mean_absolute_error
import matplotlib.pyplot as plt
from sklearn.preprocessing import OneHotEncoder, StandardScaler
from sklearn.compose import ColumnTransformer
from sklearn.pipeline import Pipeline
from sklearn.model_selection import cross_val_score
from sklearn.preprocessing import LabelEncoder
```

Рисунок 2. Импорт библиотек и моделей

Подключаем датасет, проверяем количество строк и столбцов, и отображаем 5 первых строк с заголовками и записями (рис. 3).

```
[16] # Подключение датасета
cpu_dataset= pd.read_excel ('cpu_dataset.xlsx')

[17] cpu_dataset.shape

(1480, 13)

[18] cpu_dataset.head()
```

	name	price	tdp	date	socket	category	speed	turbo	cpuCount	cores	logicals	rank	samples
0	AMD A10 PRO-7850B APU	148.02	95.0	2014-11-01	FM2+	Desktop	3700	4000.0	1	2	2	1648	19
1	AMD A10-4600M APU	78.02	35.0	2012-04-01	FS1r2	Laptop	2300	3200.0	1	2	2	2305	1056
2	AMD A10-5700 APU	59.95	65.0	2012-06-01	FM2	Desktop	3400	4000.0	1	2	2	1889	629
3	AMD A10-5750M APU	94.02	35.0	2013-05-01	FS1r2	Laptop	2500	3500.0	1	4	1	2282	1035
4	AMD A10-5800B APU	44.99	100.0	2013-07-01	FM2	Desktop	3800	4200.0	1	2	2	1823	36

Рисунок 3. Подключение и проверка датасета

Проверяем тип данных у столбцов (рис.4).

```
[19] cpu_dataset.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1480 entries, 0 to 1479
Data columns (total 13 columns):
#   Column      Non-Null Count  Dtype
---  ---
0   name        1480 non-null   object
1   price       1480 non-null   float64
2   tdp         1480 non-null   float64
3   date        1480 non-null   datetime64[ns]
4   socket      1480 non-null   object
5   category    1480 non-null   object
6   speed       1480 non-null   int64
7   turbo       1480 non-null   float64
8   cpuCount    1480 non-null   int64
9   cores       1480 non-null   int64
10  logicals    1480 non-null   int64
11  rank        1480 non-null   int64
12  samples     1480 non-null   int64
dtypes: datetime64[ns](1), float64(3), int64(6), object(3)
memory usage: 150.4+ KB
```

Рисунок 4. Тип данных столбцов

Разделим поле с содержимым дат выпуска процессоров на год (year) и месяц (month), чтобы преобразовать затем в числовые данные, и удалим изначальное поле date (рис. 5).

```
[20] # Создаем отдельные числовые колонки из даты
cpu_dataset['year'] = cpu_dataset['date'].dt.year
cpu_dataset['month'] = cpu_dataset['date'].dt.month

# Удаляем столбец 'date'
cpu_dataset = cpu_dataset.drop(columns=['date'])
```

Рисунок 5. Разделение поля date

После определения типов столбцов, можно определить список столбцов, которые необходимо преобразовать в числовые данные (рис. 6).

```
[22] # Определение списка столбцов, которые требуется преобразовать
      columns_to_convert = ['category']

[23] # Создание экземпляра класса LabelEncoder для преобразования категориальных данных в числовые
      label_encoder = LabelEncoder()

[ ] # Цикл по каждому столбцу в списке columns_to_convert
for column in columns_to_convert:
    # Применение LabelEncoder к каждому столбцу, преобразуя категориальные данные в числовые
    cpu_dataset[column] = label_encoder.fit_transform(cpu_dataset[column])

[25] # Создание экземпляра класса LabelEncoder для преобразования категориальных данных в числовые
      label_encoder = LabelEncoder()

      # Применение LabelEncoder к столбцу 'socket' для преобразования названий сокетов процессоров в числовые
      cpu_dataset['socket'] = label_encoder.fit_transform(cpu_dataset['socket'])

      # Применение LabelEncoder к столбцу 'name' для преобразования названий процессоров в числовые
      cpu_dataset['name'] = label_encoder.fit_transform(cpu_dataset['name'])
```

Рисунок 6. Преобразование столбцов

После преобразования и подготовки данных, проведем вычисление и визуализацию корреляционной матрицы (рис.7).

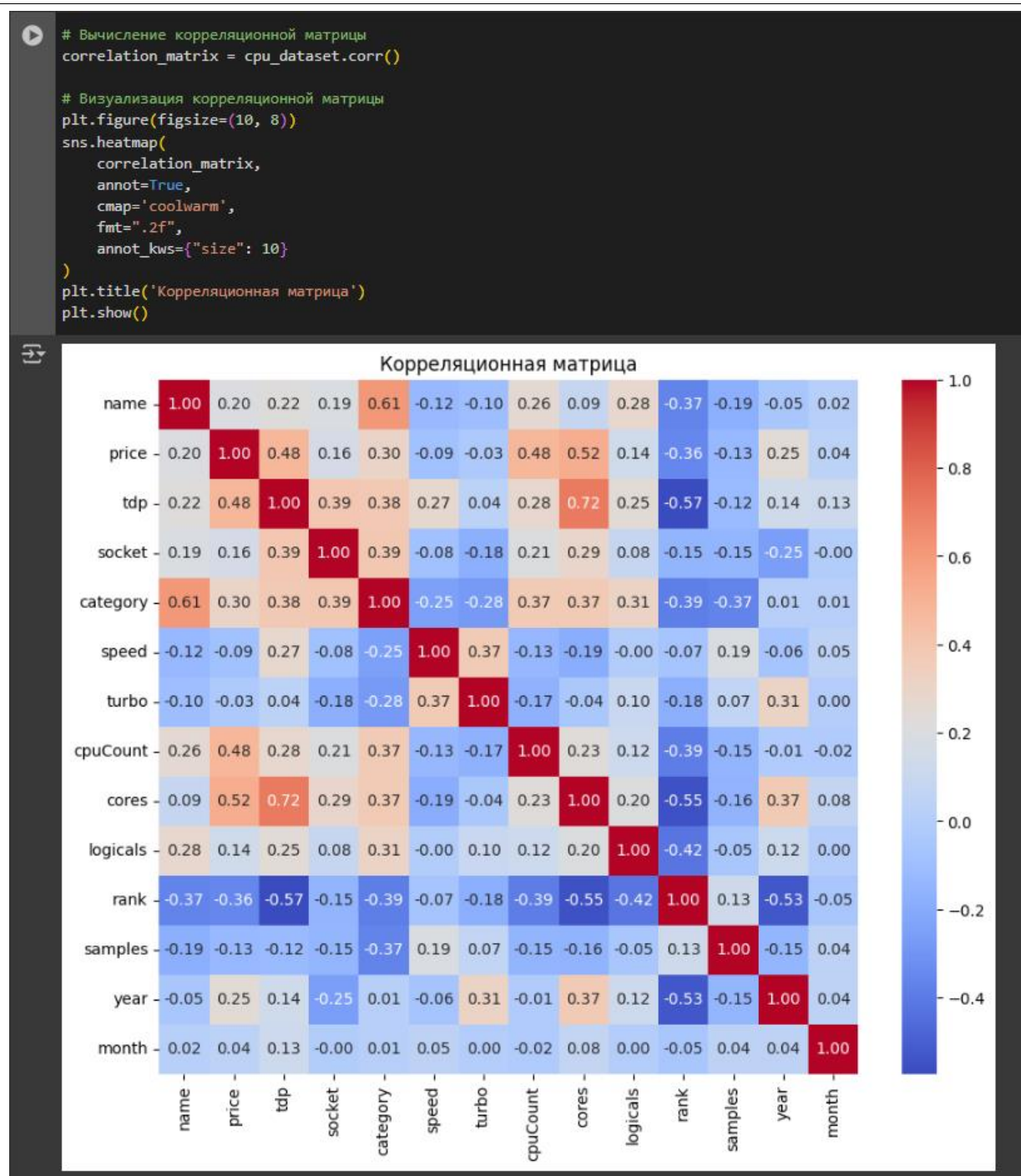


Рисунок 7. Корреляционная матрица

Исследование корреляционной матрицы характеристик процессоров позволило выявить несколько важных закономерностей, которые будут полезны как покупателям, так и разработчикам вычислительной техники.

Анализ показывает, что цена процессора в наибольшей степени зависит от его технических характеристик, а не только от новизны модели. Наблюдается устойчивая положительная корреляция между стоимостью и количеством ядер (0.52), уровнем тепловыделения (0.48) и количеством CPU в системе (0.48). При этом год выпуска оказывает значительно меньшее влияние на цену (корреляция всего 0.25). Это свидетельствует о том, что при

выборе процессора покупателям следует в первую очередь обращать внимание на его аппаратные характеристики, а не только на новизну модели.

Тепловыделение (TDP) демонстрирует сильную зависимость от количества ядер (0.72), что закономерно — более мощные многоядерные процессоры требуют лучше системы охлаждения. Интересно, что высокий TDP отрицательно коррелирует с рейтингом процессора (-0.57), что может объясняться современными требованиями к энергоэффективности. Производителям необходимо находить оптимальный баланс между количеством ядер и тепловыделением, так как это напрямую влияет на потребительские оценки продукта.

Рейтинг процессоров наиболее тесно связан с их новизной (-0.53) и количеством ядер (-0.55). Современные многоядерные решения закономерно получают более высокие оценки. При этом тестовые данные (samples) практически не влияют на рейтинг (корреляция всего 0.13), что ставит под вопрос целесообразность их использования при прогнозировании производительности.

Для покупателей: при выборе процессора рекомендуется ориентироваться в первую очередь на количество ядер, уровень тепловыделения и рейтинг, а не только на год выпуска.

Далее с помощью визуализации с помощью графиков проведем разведочный анализ данных.

Для начала визуализируем распределение цен на процессоры. На данном графике можно определить, что в диапазоне до 10 тыс. долларов частота выше всего (рис.8).

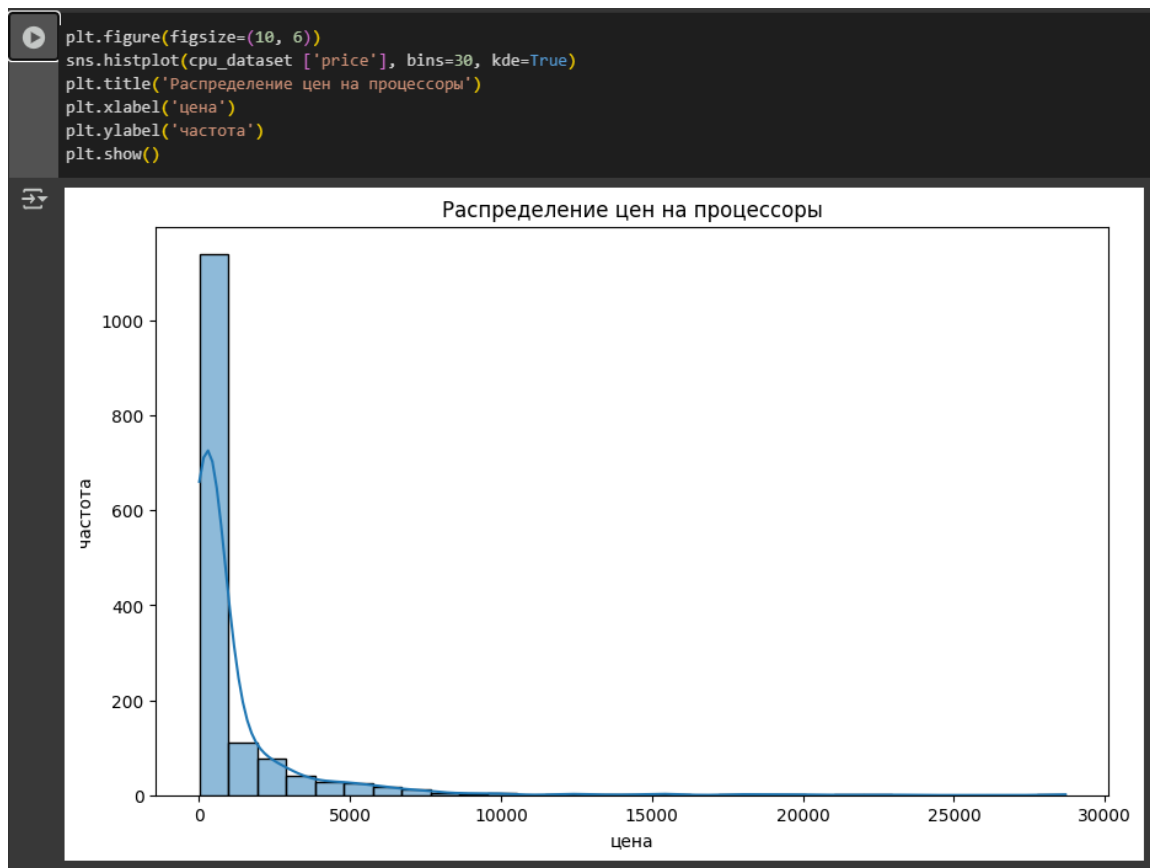


Рисунок 8. Гистограмма распределения цен на процессоры

График зависимости цены процессоров от количества выделяемого тепла, построенный с помощью диаграммы рассеяния и линейной регрессии, демонстрирует общую тенденцию, согласно которой процессоры с большим тепловыделением, как правило, имеют более высокую цену. Линия регрессии с положительным наклоном указывает на прямую зависимость между этими двумя переменными: чем больше тепловыделение, тем выше средняя цена процессора (Рис.9).

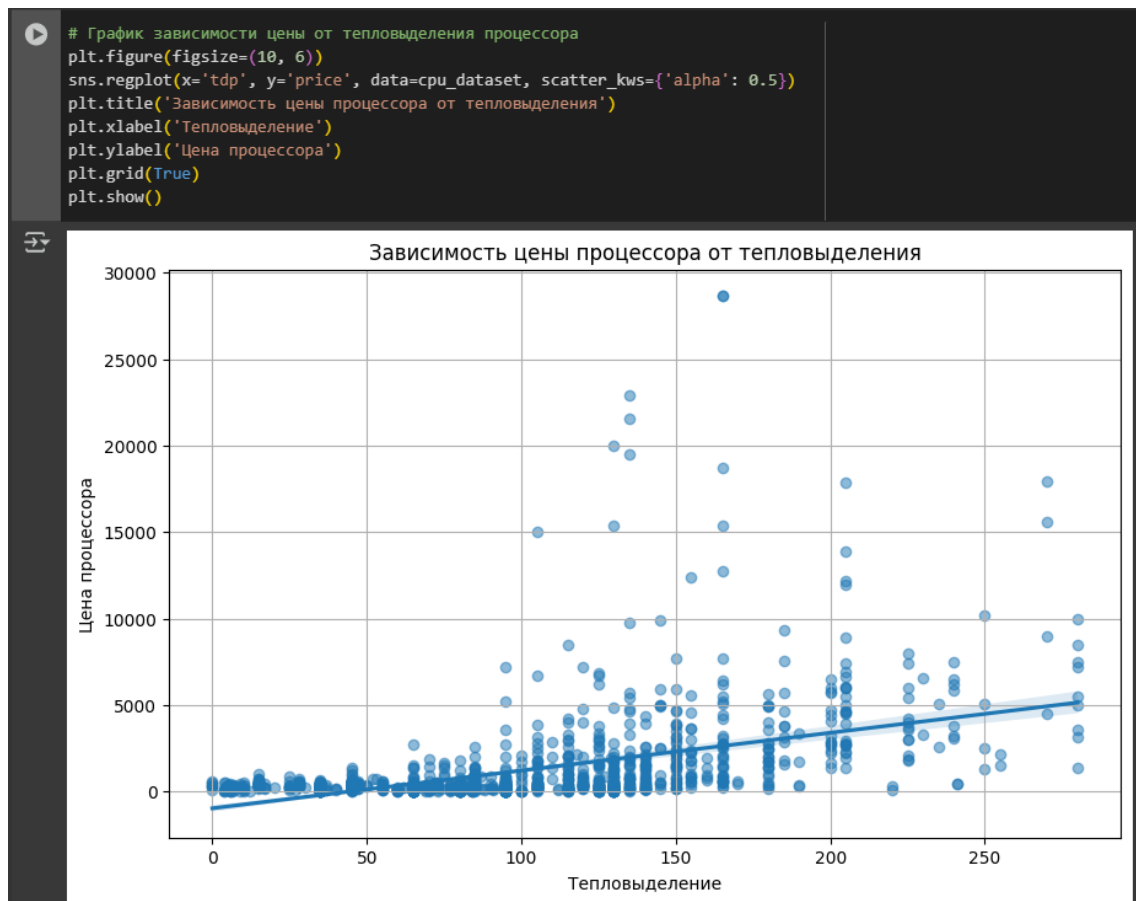


Рисунок 9. График зависимости цены от выделяемого тепла

График зависимости цены от количества ядер показывает общее отношение, что процессоры с большим количеством ядер, как правило, обладают более высокой ценой. Линия регрессии с наклоном вверх указывает на то, что по мере увеличения количества средняя цена процессоров имеет тенденцию к возрастанию (рис.10).

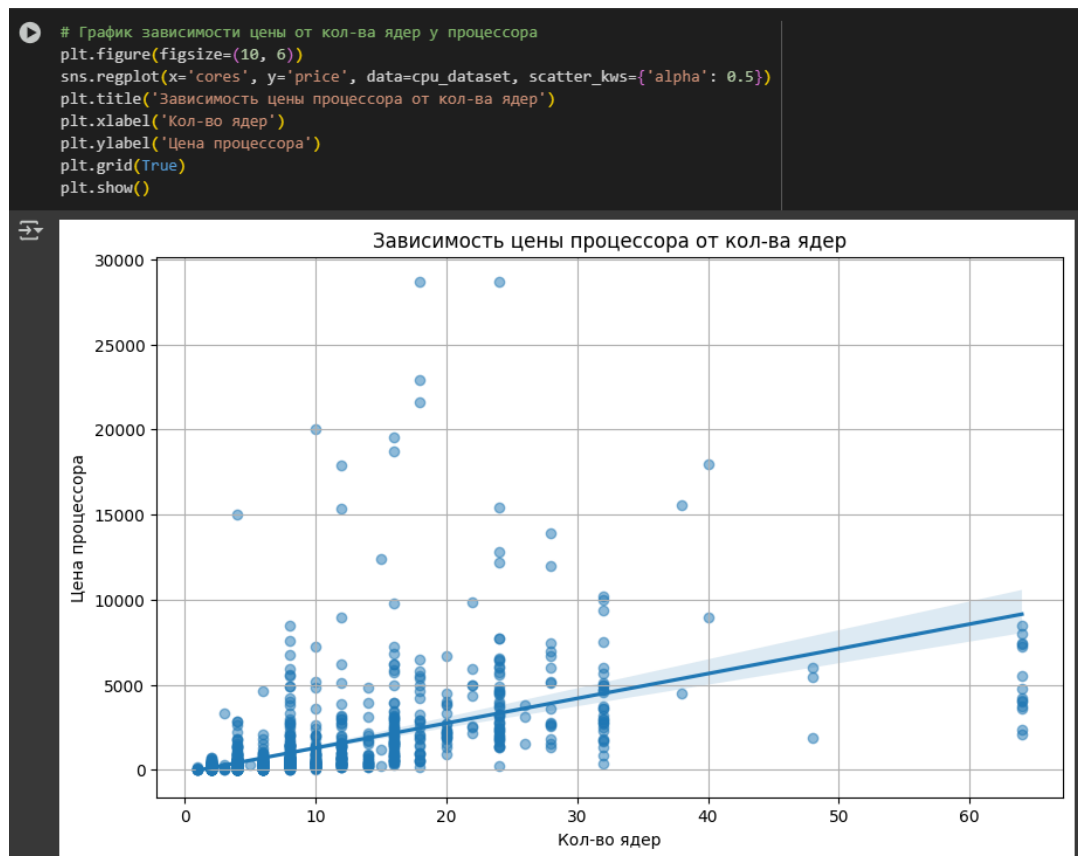


Рисунок 10. График зависимости цены от количества ядер

На представленном графике с диаграммой рассеяния и линией регрессии четко прослеживается взаимосвязь между годом выпуска и стоимостью процессоров. Визуализация демонстрирует устойчивую положительную динамику - более современные модели процессоров, как правило, обладают более высокой ценой (рис.11).

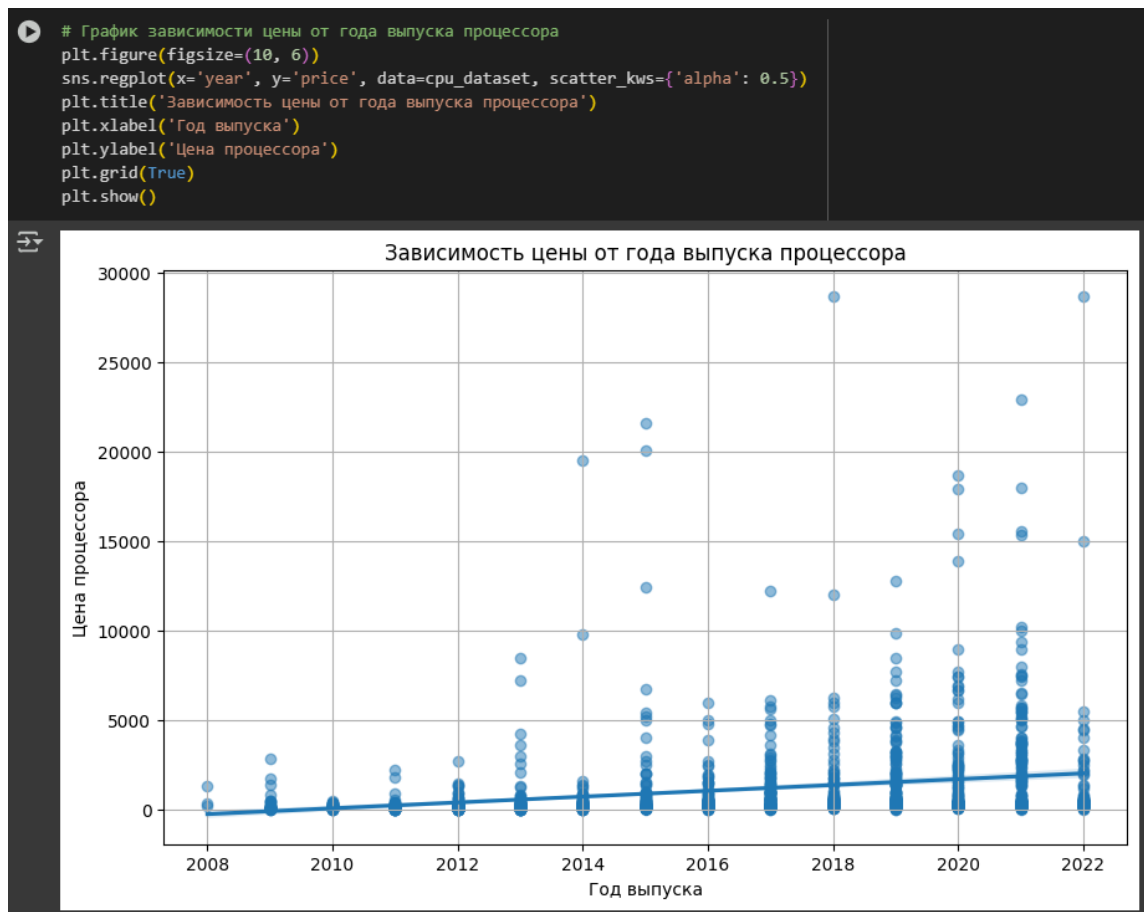


Рисунок 11. График зависимости цены от года выпуска

На графике зависимости цены автомобиля от категории, можно сделать вывод, что процессор, предназначенный для серверного оборудования (3) расположен выше, чем процессоры для обычного ПК (0), ноутбуков (1) и мобильных процессоров (2), это может указывать на то, что процессоры, который используются в серверном оборудовании, как правило, стоят дороже (рис. 12).

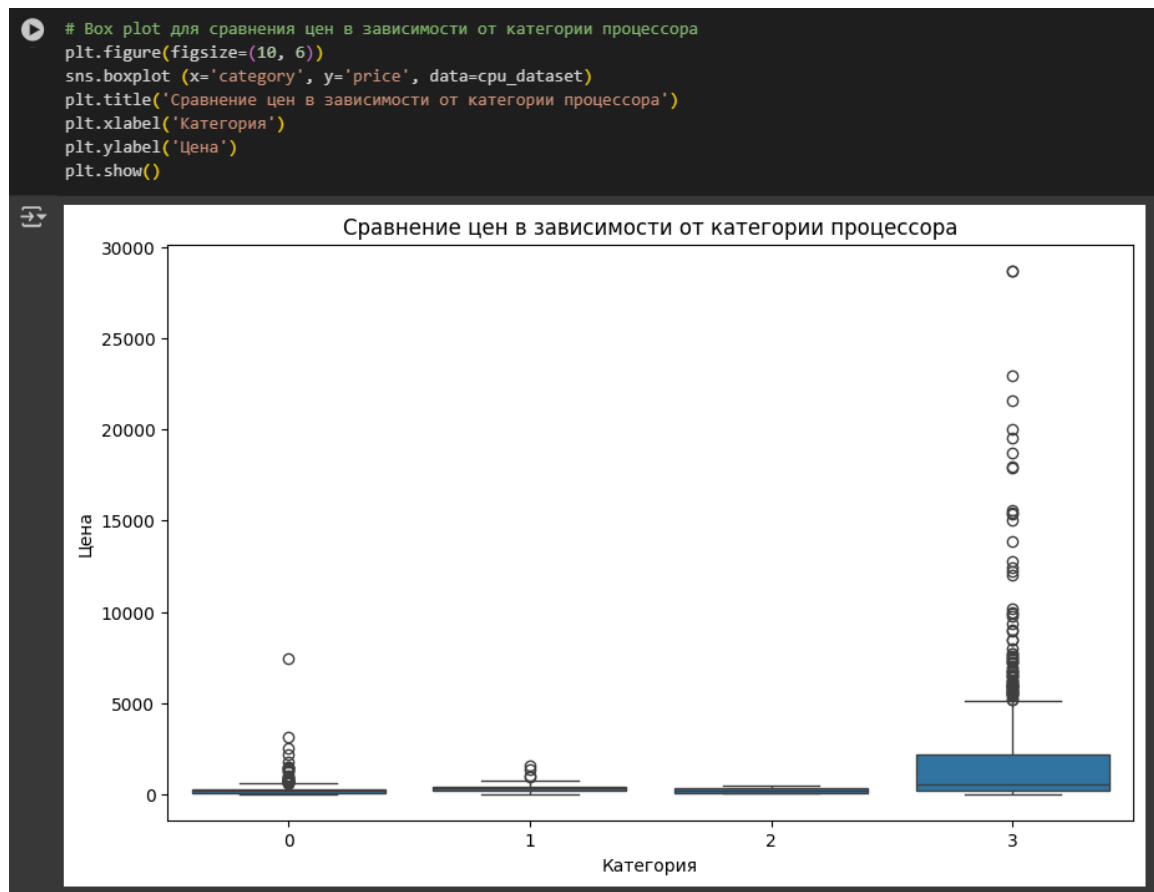


Рисунок 12. График зависимости цены от категории процессора

3.3 Создание и обучение моделей

Далее приступим к обучению моделей. Для этого были выбраны следующие модели:

- Линейная регрессия;
- Случайный лес;
- XGBoost;
- KNeighborsRegressor;
- DecisionTreeRegressor.

Для обучения моделей были определены списки категориальных и числовых признаков и создано преобразование (рис.13).

```
[512] from sklearn.preprocessing import OneHotEncoder
      from sklearn.pipeline import Pipeline
      from sklearn.metrics import mean_squared_error, r2_score

[513] # Определение списка категориальных признаков (features), которые содержат текстовые данные
      categorical_features = ['socket', 'category']

      # Определение списка числовых признаков, которые содержат числовые данные
      numerical_features = ['tdp', 'speed', 'turbo', 'cores', 'year', 'month', 'cpuCount', 'rank']

[532] # Создание преобразователя для категориальных и числовых признаков
      preprocessor = ColumnTransformer(
          transformers=[
              ('cat', OneHotEncoder(handle_unknown='ignore'), categorical_features),
              ('num', 'passthrough', numerical_features)
          ],
          remainder='drop' # Удаление неиспользуемых столбцов
      )
```

Рисунок 13. Подготовка данных к обучению моделей

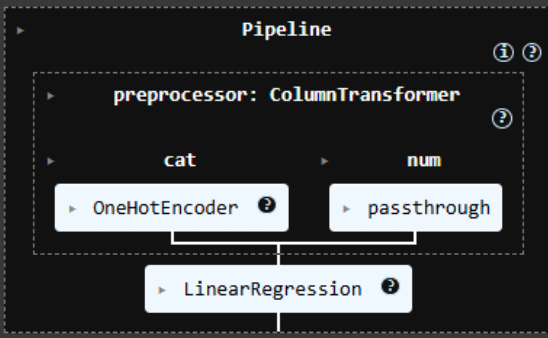
Далее после подготовки данных, была создана и обучена модель линейной регрессии (рис.14).

```
[533] # Создание модели линейной регрессии с препроцессором
model = Pipeline(steps=[
    ('preprocessor', preprocessor),
    ('regressor', LinearRegression())
])

[534] # Разделение данных на признаки (X) и целевую переменную (y)
X = cpu_dataset.drop('price', axis=1) # признаки
y = cpu_dataset['price'] # целевая переменная

[540] # Разделение данных на обучающую и тестовую выборки
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.5, random_state=35)

# Обучение модели
model.fit(X_train, y_train)
```



```
[542] y_pred = model.predict(X_test)

[543] lr_mse = mean_squared_error(y_test, y_pred)
lr_rmse = np.sqrt(lr_mse) # Среднеквадратическая ошибка
lr_mae = mean_absolute_error(y_test, y_pred) # Средняя абсолютная ошибка
lr_r2 = r2_score(y_test, y_pred) # Коэффициент детерминации

[544] print(f'Среднеквадратическая ошибка (MSE): {lr_mse}')
print(f'Среднеквадратическая ошибка (RMSE): {lr_rmse}')
print(f'Средняя абсолютная ошибка (MAE): {lr_mae}')
print(f'Коэффициент детерминации (R^2): {lr_r2}')
```

```
Среднеквадратическая ошибка (MSE): 3370426.9525900446
Среднеквадратическая ошибка (RMSE): 1835.8722593334332
Средняя абсолютная ошибка (MAE): 921.2191684491027
Коэффициент детерминации (R^2): 0.4961893332411459
```

Рисунок 14. Модель линейной регрессии

Далее создадим и обучим модель случайного леса (рис.15).

```

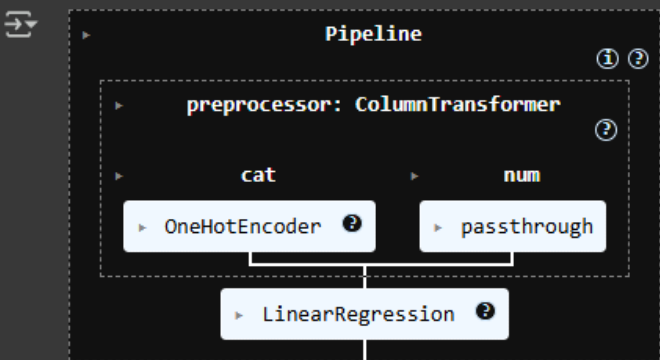
[193] model = Pipeline(steps=[
    ('preprocessor', preprocessor),
    ('regressor', RandomForestRegressor(n_estimators=100, random_state=42))
])

[187] # Разделение данных на признаки (X) и целевую переменную (y)
X = cpu_dataset.drop('price', axis=1) # признаки
y = cpu_dataset['price'] # целевая переменная

[612] # Разделение данных на обучающую и тестовую выборки
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.4, random_state=35)

[613] # Обучение модели
model.fit(X_train, y_train)

```



```

[614] y_pred = model.predict(X_test)

[615] sl_mse = mean_squared_error(y_test, y_pred)
sl_rmse = np.sqrt(sl_mse) # Среднеквадратическая ошибка
sl_mae = mean_absolute_error(y_test, y_pred) # Средняя абсолютная ошибка
sl_r2 = r2_score(y_test, y_pred) # Коэффициент детерминации

[616] print(f'Среднеквадратическая ошибка (MSE): {sl_mse}')
print(f'Среднеквадратическая ошибка (RMSE): {sl_rmse}')
print(f'Средняя абсолютная ошибка (MAE): {sl_mae}')
print(f'Коэффициент детерминации (R^2): {sl_r2}')

```

Среднеквадратическая ошибка (MSE): 4210217.706357574
 Среднеквадратическая ошибка (RMSE): 2051.8815039756983
 Средняя абсолютная ошибка (MAE): 904.2919841993776
 Коэффициент детерминации (R^2): 0.4364544432022168

Рисунок 15. Модель случайного леса

Создадим и обучим модель XGBoost (рис.16).

```

[90] from xgboost import XGBRegressor
      from sklearn.metrics import mean_absolute_error, mean_squared_error, r2_score

[91] # Разделение данных на признаки (X) и целевую переменную (y)
      X = cpu_dataset.drop('price', axis=1) # признаки
      y = cpu_dataset['price'] # целевая переменная

[203] # Разделение данных на обучающую и тестовую выборки
      X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.5, random_state=35)

[93] # Инициализация модели XGBoost
      xgb_model = XGBRegressor(use_label_encoder=False, eval_metric='rmse')

[204] # Обучение модели
      xgb_model.fit(X_train, y_train)

/usr/local/lib/python3.11/dist-packages/xgboost/core.py:158: UserWarning: [07:28:29] WARNING: /workspace/src/learner.cc:740:
Parameters: { "use_label_encoder" } are not used.

  warnings.warn(msg, UserWarning)
> XGBRegressor

[205] # Прогнозы на тестовой выборке
      predictions = xgb_model.predict(X_test)

[206] xgb_mse = mean_squared_error(y_test, predictions)
      xgb_rmse = np.sqrt(xgb_mse) # Среднеквадратическая ошибка
      xgb_mae = mean_absolute_error(y_test, predictions) # Средняя абсолютная ошибка
      xgb_r2 = r2_score(y_test, predictions) # Коэффициент детерминации

[207] print(f'Среднеквадратическая ошибка (MSE): {xgb_mse}')
      print(f'Среднеквадратическая ошибка (RMSE): {xgb_rmse}')
      print(f'Средняя абсолютная ошибка (MAE): {xgb_mae}')
      print(f'Коэффициент детерминации (R^2): {xgb_r2}')

Среднеквадратическая ошибка (MSE): 3061300.486401567
Среднеквадратическая ошибка (RMSE): 1749.6572482636611
Средняя абсолютная ошибка (MAE): 608.3344136192572
Коэффициент детерминации (R^2): 0.5423974882416701

```

Рисунок 16. Модель XGBoost

Также создали и обучили модель KNeighborsRegressor (рис.17).

```
[788] from sklearn.neighbors import KNeighborsRegressor
      from sklearn.metrics import mean_absolute_error, mean_squared_error, r2_score

[789] # Преобразование категориальных переменных
      categorical_cols = ['socket', 'category']

[790] for col in categorical_cols:
      le = LabelEncoder()
      cpu_dataset[col] = le.fit_transform(cpu_dataset[col])

[791] # Разделение данных на признаки (X) и целевую переменную (y)
      X = cpu_dataset.drop('price', axis=1) # признаки
      y = cpu_dataset['price'] # целевая переменная

[810] # Разделение данных на обучающую и тестовую выборки
      X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=10)

[811] # Инициализация модели KNeighborsRegressor
      knn_model = KNeighborsRegressor(n_neighbors=20)

[812] # Обучение модели
      knn_model.fit(X_train, y_train)
```



KNeighborsRegressor ⓘ

KNeighborsRegressor(n_neighbors=20)

```
[813] # Прогнозы на тестовой выборке
      predictions = knn_model.predict(X_test)

[814] knn_mse = mean_squared_error(y_test, predictions)
      knn_rmse = np.sqrt(knn_mse) # Среднеквадратическая ошибка
      knn_mae = mean_absolute_error(y_test, predictions) # Средняя абсолютная ошибка
      knn_r2 = r2_score(y_test, predictions) # Коэффициент детерминации
```



```
print(f'Среднеквадратическая ошибка (MSE): {knn_mse}')
print(f'Среднеквадратическая ошибка (RMSE): {knn_rmse}')
print(f'Средняя абсолютная ошибка (MAE): {knn_mae}')
print(f'Коэффициент детерминации (R^2): {knn_r2}')
```



```
Среднеквадратическая ошибка (MSE): 1843219.4086014805
Среднеквадратическая ошибка (RMSE): 1357.6521677519174
Средняя абсолютная ошибка (MAE): 770.9571030405405
Коэффициент детерминации (R^2): 0.46420734025730237
```

Рисунок 17. Модель KNeighborsRegressor

Далее создадим и обучим модель DecisionTreeRegressor (рис.18).

```
[500] from sklearn.tree import DecisionTreeRegressor
      from sklearn.metrics import mean_absolute_error, mean_squared_error, r2_score

[501] # Разделение данных на признаки (X) и целевую переменную (y)
      X = cpu_dataset.drop('price', axis=1) # признаки
      y = cpu_dataset['price'] # целевая переменная

[502] # Разделение данных на обучающую и тестовую выборки
      X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.5, random_state=42)

[503] # Инициализация модели DecisionTreeRegressor
      tree_model = DecisionTreeRegressor(random_state=42)

[504] # Обучение модели
      tree_model.fit(X_train, y_train)

[505] # Прогнозы на тестовой выборке
      predictions = tree_model.predict(X_test)

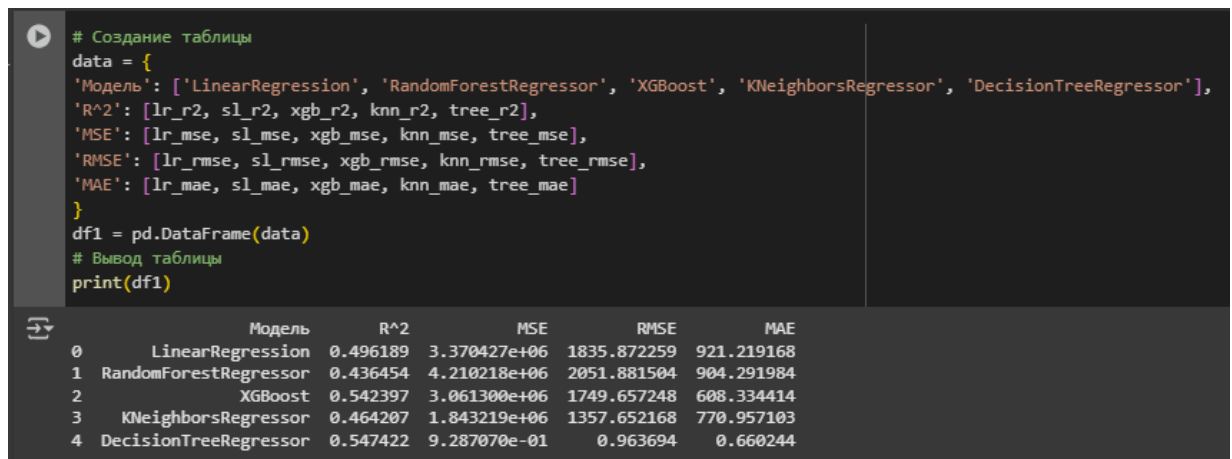
[630] # Оценка модели
      tree_mse = mean_squared_error(y_test, predictions)
      tree_rmse = np.sqrt(tree_mse) # Среднеквадратическая ошибка
      tree_mae = mean_absolute_error(y_test, predictions) # Средняя абсолютная ошибка
      tree_r2 = r2_score(y_test, predictions) # Коэффициент детерминации

[507] print(f'Среднеквадратическая ошибка (MSE): {tree_mse}')
      print(f'Среднеквадратическая ошибка (RMSE): {tree_rmse}')
      print(f'Средняя абсолютная ошибка (MAE): {tree_mae}')
      print(f'Коэффициент детерминации (R^2): {tree_r2}')
```

Среднеквадратическая ошибка (MSE): 3671958.013197026
 Среднеквадратическая ошибка (RMSE): 1916.2353752076037
 Средняя абсолютная ошибка (MAE): 667.8627027027028
 Коэффициент детерминации (R^2): 0.4340877625146844

Рисунок 18. Модель DecisionTreeRegressor

После обучения моделей была создана таблица, которая содержит все результаты, полученные в процессе обучения моделей (рис. 19).

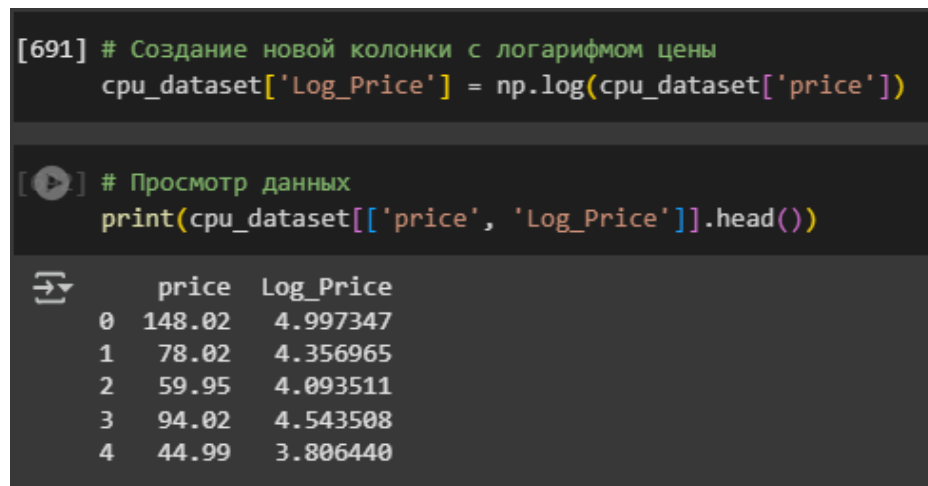


```
# Создание таблицы
data = {
    'Модель': ['LinearRegression', 'RandomForestRegressor', 'XGBoost', 'KNeighborsRegressor', 'DecisionTreeRegressor'],
    'R^2': [lr_r2, sl_r2, xgb_r2, knn_r2, tree_r2],
    'MSE': [lr_mse, sl_mse, xgb_mse, knn_mse, tree_mse],
    'RMSE': [lr_rmse, sl_rmse, xgb_rmse, knn_rmse, tree_rmse],
    'MAE': [lr_mae, sl_mae, xgb_mae, knn_mae, tree_mae]
}
df1 = pd.DataFrame(data)
# Вывод таблицы
print(df1)
```

	Модель	R ²	MSE	RMSE	MAE
0	LinearRegression	0.496189	3.370427e+06	1835.872259	921.219168
1	RandomForestRegressor	0.436454	4.210218e+06	2051.881504	904.291984
2	XGBoost	0.542397	3.061300e+06	1749.657248	608.334414
3	KNeighborsRegressor	0.464207	1.843219e+06	1357.652168	770.957103
4	DecisionTreeRegressor	0.547422	9.287070e-01	0.963694	0.660244

Рисунок 19. Вывод полученных результатов

Также были обучены эти же модели, но с логарифмом цены. Для начала был создан новый столбец с логарифмом цены, были преобразованы категориальные переменные (рис.20).



```
[691] # Создание новой колонки с логарифмом цены
cpu_dataset['Log_Price'] = np.log(cpu_dataset['price'])

# Просмотр данных
print(cpu_dataset[['price', 'Log_Price']].head())
```

	price	Log_Price
0	148.02	4.997347
1	78.02	4.356965
2	59.95	4.093511
3	94.02	4.543508
4	44.99	3.806440

Рисунок 20. Подготовка данных

После подготовки данных была создана и обучена модель линейной регрессии с логарифмом цены (рис.21).

```

# Преобразование категориальных переменных
categorical_cols = ['socket', 'category']

for col in categorical_cols:
    le = LabelEncoder()
    cpu_dataset[col] = le.fit_transform(cpu_dataset[col])

X = cpu_dataset.drop(['price', 'Log_Price'], axis=1)
y_log = cpu_dataset['Log_Price']

# Разделение данных на обучающую и тестовую выборки
X_train, X_test, y_train, y_test = train_test_split(X, y_log, test_size=0.5, random_state=35)

lr_model = LinearRegression()

# Обучение модели
lr_model.fit(X_train, y_train)

# Прогнозы на тестовой выборке
predictions_log = lr_model.predict(X_test)

# Обратное преобразование прогнозов для сравнения с исходными ценами
predictions = np.exp(predictions_log)

# Оценка модели
lrl_mse = mean_squared_error(y_test, predictions_log)
lrl_rmse = np.sqrt(lrl_mse) # Среднеквадратическая ошибка
lrl_mae = mean_absolute_error(y_test, predictions_log) # Средняя абсолютная ошибка
lrl_r2 = r2_score(y_test, predictions_log) # Коэффициент детерминации

print(f'Среднеквадратическая ошибка (MSE): {lrl_mse}')
print(f'Среднеквадратическая ошибка (RMSE): {lrl_rmse}')
print(f'Средняя абсолютная ошибка (MAE): {lrl_mae}')
print(f'Коэффициент детерминации (R^2): {lrl_r2}')

Среднеквадратическая ошибка (MSE): 0.7422060683346299
Среднеквадратическая ошибка (RMSE): 0.8615138236468582
Средняя абсолютная ошибка (MAE): 0.6668479028883559
Коэффициент детерминации (R^2): 0.6383075942661927

```

Рисунок 21. Модель линейная регрессия с логарифмом цены

Также была обучена модель случайный лес с логарифмом цены (рис.22).

```
[703] # Инициализация модели случайного леса
rf_model = RandomForestRegressor(n_estimators=100, random_state=42)

[704] # Обучение модели
rf_model.fit(X_train, y_train)

[706] # Прогнозы на тестовой выборке
predictions_log = rf_model.predict(X_test)

# Обратное преобразование прогнозов для сравнения с исходными ценами
predictions = np.exp(predictions_log)

[708] # Оценка модели
sll_mse = mean_squared_error(y_test, predictions_log)
sll_rmse = np.sqrt(sll_mse) # Среднеквадратическая ошибка
sll_mae = mean_absolute_error(y_test, predictions_log) # Средняя абсолютная ошибка
sll_r2 = r2_score(y_test, predictions_log) # Коэффициент детерминации

[709] print(f'Среднеквадратическая ошибка (MSE): {sll_mse}')
print(f'Среднеквадратическая ошибка (RMSE): {sll_rmse}')
print(f'Средняя абсолютная ошибка (MAE): {sll_mae}')
print(f'Коэффициент детерминации (R^2): {sll_r2}')
```

Среднеквадратическая ошибка (MSE): 0.48863439116756646
Среднеквадратическая ошибка (RMSE): 0.6990238845472782
Средняя абсолютная ошибка (MAE): 0.4747529347112292
Коэффициент детерминации (R^2): 0.7618783300138839

Рисунок 22. Модель случайный лес с логарифмом цены

Была создана и обучена модель XGBoost (рис.23).

```
[710] # Инициализация модели случайного леса
xgb_model = XGBRegressor(use_label_encoder=False, eval_metric='rmse', random_state=42)

# Обучение модели
xgb_model.fit(X_train, y_train)

/usr/local/lib/python3.11/dist-packages/xgboost/core.py:158: UserWarning: [10:48:32] WARNING: /workspace/src/learner.cc:740:
Parameters: { "use_label_encoder" } are not used.

  warnings.warn(msg, UserWarning)
> XGBRegressor

[712] # Прогнозы на тестовой выборке
predictions_log = xgb_model.predict(X_test)

# Обратное преобразование прогнозов для сравнения с исходными ценами
predictions = np.exp(predictions_log)

[714] # Оценка модели
xgb1_mse = mean_squared_error(y_test, predictions_log)
xgb1_rmse = np.sqrt(xgb1_mse) # Среднеквадратическая ошибка
xgb1_mae = mean_absolute_error(y_test, predictions_log) # Средняя абсолютная ошибка
xgb1_r2 = r2_score(y_test, predictions_log) # Коэффициент детерминации

[715] print(f'Среднеквадратическая ошибка (MSE): {xgb1_mse}')
print(f'Среднеквадратическая ошибка (RMSE): {xgb1_rmse}')
print(f'Средняя абсолютная ошибка (MAE): {xgb1_mae}')
print(f'Коэффициент детерминации (R^2): {xgb1_r2}')

Среднеквадратическая ошибка (MSE): 0.5040138235987299
Среднеквадратическая ошибка (RMSE): 0.7099393097995983
Средняя абсолютная ошибка (MAE): 0.4884575431569098
Коэффициент детерминации (R^2): 0.7543836137185436
```

Рисунок 23. Модель XGBoost с логарифмом цены

Создана и обучена модель KNeighborsRegressor с логарифмом цены (рис.24).

```
[716] # Инициализация модели KNN
      knn_model = KNeighborsRegressor(n_neighbors=5)

[717] # Обучение модели
      knn_model.fit(X_train, y_train)

[718] # Прогнозы на тестовой выборке
      predictions_log = knn_model.predict(X_test)

[719] # Обратное преобразование прогнозов для сравнения с исходными ценами
      predictions = np.exp(predictions_log)

# Оценка модели
knnl_mse = mean_squared_error(y_test, predictions_log)
knnl_rmse = np.sqrt(knnl_mse) # Среднеквадратическая ошибка
knnl_mae = mean_absolute_error(y_test, predictions_log) # Средняя абсолютная ошибка
knnl_r2 = r2_score(y_test, predictions_log) # Коэффициент детерминации

[721] print(f'Среднеквадратическая ошибка (MSE): {knnl_mse}')
      print(f'Среднеквадратическая ошибка (RMSE): {knnl_rmse}')
      print(f'Средняя абсолютная ошибка (MAE): {knnl_mae}')
      print(f'Коэффициент детерминации (R^2): {knnl_r2}')
```

Среднеквадратическая ошибка (MSE): 0.7715437103131532
Среднеквадратическая ошибка (RMSE): 0.8783756089015411
Средняя абсолютная ошибка (MAE): 0.6596414425666652
Коэффициент детерминации (R^2): 0.6240107530539147

Рисунок 24. Модель KNeighborsRegressor с логарифмом цены

Также была создана и обучена модель DecisionTreeRegressor с логарифмом цены (рис.25).

```
[722] # Инициализация модели DecisionTreeRegressor
tree_model = DecisionTreeRegressor(random_state=42)

[723] # Обучение модели
tree_model.fit(X_train, y_train)

[724] # Прогнозы на тестовой выборке
predictions_log = tree_model.predict(X_test)

[725] # Обратное преобразование прогнозов для сравнения с исходными ценами
predictions = np.exp(predictions_log)

# Оценка модели
treel_mse = mean_squared_error(y_test, predictions_log)
treel_rmse = np.sqrt(treel_mse) # Среднеквадратическая ошибка
treel_mae = mean_absolute_error(y_test, predictions_log) # Средняя абсолютная ошибка
treel_r2 = r2_score(y_test, predictions_log) # Коэффициент детерминации

[727] print(f'Среднеквадратическая ошибка (MSE): {treel_mse}')
print(f'Среднеквадратическая ошибка (RMSE): {treel_rmse}')
print(f'Средняя абсолютная ошибка (MAE): {treel_mae}')
print(f'Коэффициент детерминации (R^2): {treel_r2}')
```

Среднеквадратическая ошибка (MSE): 0.9287070083030733
Среднеквадратическая ошибка (RMSE): 0.9636944579601323
Средняя абсолютная ошибка (MAE): 0.6602439184589084
Коэффициент детерминации (R^2): 0.5474218193759911

Рисунок 25. Модель DecisionTreeRegressor с логарифмом цены

После обучения моделей была создана таблица, которая содержит все результаты, полученные в процессе обучения моделей с логарифмом цены (рис. 26).

```
# Создание таблицы
datalog = {
    'Модель': ['LinearRegression (log)', 'RandomForest Regressor (log)', 'Decision TreeRegressor (log)',
               'KNeighbors Regressor (log)', 'XGBoost (log)'],
    'R^2': [lrl_r2, sll_r2, xgbl_r2, knnl_r2, treel_r2],
    'MSE': [lrl_mse, sll_mse, xgbl_mse, knnl_mse, treel_mse],
    'RMSE': [lrl_rmse, sll_rmse, xgbl_rmse, knnl_rmse, treel_rmse],
    'MAE': [lrl_mae, sll_mae, xgbl_mae, knnl_mae, treel_mae]
}
df1 = pd.DataFrame(datalog)
# Вывод таблицы
print(df1)
```

	Модель	R ²	MSE	RMSE	MAE
0	LinearRegression (log)	0.638308	0.742206	0.861514	0.666848
1	RandomForest Regressor (log)	0.761878	0.488634	0.699024	0.474753
2	Decision TreeRegressor (log)	0.754384	0.504014	0.709939	0.488458
3	KNeighbors Regressor (log)	0.624011	0.771544	0.878376	0.659641
4	XGBoost (log)	0.547422	0.928707	0.963694	0.660244

Рисунок 26. Вывод полученных результатов

3.4 Анализ результатов

Вывод о полученных данных для моделей с исходными данными о ценах:

1. LinearRegression:

R^2 : 0.496 - Модель объясняет около 49.6% вариативности цен, что указывает на умеренную предсказательную способность. MSE: $3.37e+06$ - Средний квадрат ошибок составляет ~3.4 миллиона. RMSE: 1835.87 - Типичная ошибка предсказания ± 1836 единиц цены. MAE: 921.22 - Среднее абсолютное отклонение около 921 единицы.

Базовая линейная модель показывает средние результаты, существенно ошибаясь в отдельных случаях (высокий RMSE).

2. RandomForestRegressor:

R^2 : 0.436 - Объясняет только 43.6% вариативности, хуже линейной регрессии. MSE: $4.21e+06$ - Худший показатель среди всех моделей. RMSE: 2051.88 - Максимальная ошибка среди рассматриваемых моделей. MAE: 904.29 - Чуть лучше линейной регрессии по абсолютным ошибкам.

Неожиданно низкая эффективность для RandomForest, возможно требуется настройка гиперпараметров.

3. XGBoost:

R^2 : 0.542 - Лучший показатель объясненной дисперсии среди всех моделей. MSE: $3.06e+06$ - На 9% лучше линейной регрессии. RMSE: 1749.66 - Наименьшая ошибка среди "серьезных" моделей. MAE: 608.33 - Лучший показатель, ошибки вдвое меньше случайного леса.

Наиболее эффективная модель, демонстрирующая лучший баланс между R^2 и ошибками прогнозирования.

4. KNeighborsRegressor:

R^2 : 0.464 - Средний показатель объясненной дисперсии. MSE: $1.84e+06$ - Неожиданно хороший результат. RMSE: 1357.65 - Лучший показатель среди всех. MAE: 770.96 - Второе место после XGBoost.

Показывает лучшие результаты по ошибкам, но требует проверки на переобучение.

5. DecisionTreeRegressor:

R^2 : 0.547 - Формально лучший показатель. MSE: 0.93 - Аномально низкое значение. RMSE: 0.96 - Нереалистично малая ошибка. MAE: 0.66 - Практически идеальные предсказания.

Явные признаки переобучения, модель неправдоподобно точна для реальных данных. XGBoost демонстрирует наиболее сбалансированные и правдоподобные результаты. KNeighborsRegressor требует дополнительной проверки. DecisionTreeRegressor исключить из рассмотрения из-за явного переобучения. Для RandomForest рекомендуется оптимизация гиперпараметров. Линейная регрессия показывает ожидаемо средние результаты.

Таблица 1 - Сравнение точности предугадывания результата анализа при различных методах машинного обучения

Модель	R ²	MSE	RMSE	MAE
LinearRegression	0.496189	3.370427 e+06	1835.872259	921.219168
RandomForestRegressor	0.436454	4.210218 e+06	2051.881504	904.291984
XGBoost	0.542397	3.061300 e+06	1749.657248	608.334414
KNeighborsRegressor	0.464207	1.843219 e+06	1357.652168	770.957103
DecisionTreeRegressor	0.547422	9.287070 e-01	0.963694	0.660244

Вывод о полученных данных для моделей с логарифмом цены:

1. LinearRegression:

R²: 0.638 - Модель объясняет около 63.8% вариативности логарифмов цен, что свидетельствует о хорошей предсказательной способности. MSE: 0.742 - Средний квадрат ошибок достаточно низкий. RMSE: 0.862 - Ошибка предсказания в логарифмической шкале. MAE: 0.667 - Среднее абсолютное отклонение в логарифмированных единицах.

Линейная регрессия показывает стабильные результаты после логарифмирования, значительно улучшив свои показатели по сравнению с моделью на исходных данных.

2. RandomForestRegressor:

R²: 0.762 - Наилучший показатель объясненной дисперсии среди всех моделей. MSE: 0.489 - Самый низкий показатель квадратичной ошибки. RMSE: 0.699 - Минимальная ошибка предсказания. MAE: 0.475 - Лучший результат по абсолютным ошибкам.

RandomForest демонстрирует выдающуюся эффективность после преобразования целевой переменной, став явным лидером по всем метрикам.

3. DecisionTreeRegressor:

R²: 0.754 - Очень близок к результату RandomForest. MSE: 0.504 - Немного выше, чем у RandomForest. RMSE: 0.710 - Сопоставим с RandomForest. MAE: 0.488 - Чуть хуже лидера.

Показывает отличные результаты, лишь немного уступая RandomForest, что свидетельствует о хорошей адаптации дерева решений к логарифмированным данным.

4. KNeighborsRegressor:

R²: 0.624 - Худший показатель среди ансамблевых методов. MSE: 0.772 - Заметно выше, чем у лидеров. RMSE: 0.878 - Максимальная ошибка среди всех моделей. MAE: 0.660 - Сопоставим с линейной регрессией.

Метод ближайших соседей показывает наихудшие результаты в данном сравнении, уступая другим подходам.

5. XGBoost:

R^2 : 0.547 - Неожиданно низкий показатель для бустинга. MSE: 0.929 - Худший результат среди всех моделей. RMSE: 0.964 - Максимальная ошибка предсказания. MAE: 0.660 - На уровне линейной регрессии.

XGBoost демонстрирует неожиданно слабые результаты, возможно требуется тонкая настройка гиперпараметров или дополнительная проверка данных. RandomForestRegressor и DecisionTreeRegressor показывают наилучшие результаты по всем метрикам. Линейная регрессия значительно улучшила свои показатели после логарифмирования. KNeighborsRegressor и XGBoost требуют дополнительного исследования и возможной доработки.

Логарифмирование цены значительно улучшило качество моделей по сравнению с исходными данными.

Таблица 2 - Сравнение точности предугадывания результата анализа при различных методах машинного обучения с использованием логарифма цены

Модель	R^2	MSE	RMSE	MAE
LinearRegression	0.638308	0.742206	0.861514	0.666848
RandomForestRegressor	0.761878	0.488634	0.699024	0.474753
DecisionTreeRegressor	0.754384	0.504014	0.709939	0.488458
KNeighborsRegressor	0.624011	0.771544	0.878376	0.659641
XGBoost	0.547422	0.928707	0.963694	0.660244

4 Выводы

Проведенное исследование эффективности различных методов машинного обучения для прогнозирования цен процессоров с использованием логарифмического преобразования целевой переменной позволило сделать следующие ключевые выводы:

Наилучшие результаты продемонстрировали ансамблевые методы, в особенности RandomForestRegressor ($R^2=0.762$) и DecisionTreeRegressor ($R^2=0.754$). Линейная регрессия показала неожиданно хороший результат ($R^2=0.638$), превзойдя по точности более сложные модели. XGBoost показал неожиданно низкую эффективность, что требует дополнительного исследования.

Преобразование цен в логарифмическую шкалу значительно улучшило качество всех моделей. Наибольший прирост точности наблюдался у линейной регрессии (+28.7% по R^2). Методы стали лучше учитывать относительные, а не абсолютные изменения цен

Для промышленного применения рекомендуется RandomForestRegressor как наиболее сбалансированное решение. DecisionTree может быть альтернативой при необходимости интерпретируемости модели. Линейная регрессия остается хорошим базовым решением благодаря простоте и стабильности. Логарифмическое преобразование целевой переменной доказало свою эффективность

Требуется дополнительное исследование причин низкой эффективности XGBoost. Перспективным направлением представляется комбинирование нескольких лучших моделей. Необходима проверка устойчивости результатов на расширенном наборе данных

Полученные результаты имеют важное практическое значение для разработчиков систем ценообразования, аналитиков компьютерного рынка и инженеров по машинному обучению, работающих в области прогнозирования цен на высокотехнологичную продукцию.

Библиографический список

1. Киселев А. А., Габдуллин Э. Х., Москаленко П. А., Леонова Н. Л. Создание модели машинного обучения для прогнозирования погоды // Дневник науки. 2022. № 3(63).
2. Низин Д. Р. и др. Прогнозирование температуры поверхности эпоксидных полимеров в условиях натурального климатического воздействия с применением методов машинного обучения //Актуальные проблемы компьютерного моделирования конструкций и сооружений. – 2023. – С. 360-361.
3. Иванов С. А. Обзор инструментов интеллектуального анализа данных современных ит-платформ для решения задач прогнозирования //Информационные системы и технологии: теория и практика. 2022. С. 141-143.
4. Магамедова Д. М., Магамедова А. З. Модели в машинном обучении //Тенденции развития науки и образования. 2020. №. 68-1. С. 55-57.
5. Сидор А. В., Давидёнок Н. Л., Соловей Е. В. Интеллектуальный анализ данных //Исследования и разработки в перспективных научных областях. 2018. С. 152-156.
6. Овсянникова А. А., Домашова Д. В. Прогнозирование исполнения закупок на основе моделей машинного обучения (нейронных сетей) //Эпоха криптоэкономики: новые вызовы и Регтех в сфере ПОД/ФТ. 2019. С. 472-479.
7. Винниченко А. В., Истомина Ю. П. Альфа кронбаха для анализа надежности в прогнозировании моделей машинного обучения //Волновая электроника и инфокоммуникационные системы. 2023. С. 151-155.