УДК 004

Интеллектуальный анализ данных цен на легковые автомобили в Еврейской автономной области

Голубева Евгения Павловна
Приамурский государственный университет имени Шолом-Алейхема
Студент

Аннотация
Цель данной статьи – разработать и применить регрессионные модели машинного обучения для прогнозирования цен на легковые автомобили в Еврейской автономной области. Для разработки и применения регрессионной модели машинного обучения была использована интерактивная облачная среда Google Colaboratory и датасет с данными легковых автомобилей. С использованием Google Colaboratory был проведен комплексный анализ различных моделей регрессии с целью определения наиболее эффективного инструмента для предсказания цен на автомобили.
Ключевые слова: Google Colaboratory, библиотека, регрессионная модель.

Intelligent analysis of passenger car price data in the Jewish Autonomous Region

Golubeva Evgeniya Pavlovna
Sholom-Aleichem Priamursky State University
Student

Abstract
The purpose of this article is to develop and apply regression models of machine learning to predict the prices of passenger cars in the Jewish Autonomous Region. The interactive Google Colaboratory cloud environment and a dataset with passenger car data were used to develop and apply the regression model of machine learning. Using Google Colaboratory, a comprehensive analysis of various regression models was carried out in order to determine the most effective tool for predicting car prices.
Keywords: Google Colaboratory, library, regression model.

1 Введение

1.1 Актуальность

Каждый человек в своей жизни сталкивается с проблемой выбора, особенно в условиях перенасыщенного рынка, где огромное количество продукции затрудняет осознанный выбор. В случае с покупкой автомобиля, проблема усугубляется тем, что большинство покупателей не являются экспертами в данной предметной области.

Интеллектуальный анализ данных цен на автомобили помогает решить эту проблему, предоставляя потребителям и деловым структурам инструменты для понимания текущих рыночных тенденций, выявления факторов, влияющих на цены, и принятия обоснованных решений. Этот анализ не только упрощает процесс выбора для конечных пользователей, но и позволяет производителям и дистрибьюторам более точно устанавливать цены, что в свою очередь стимулирует конкурентоспособность и эффективность рынка автомобилей.

1.2 Обзор исследований

В.П. Невежин провел анализ продаж новых легковых автомобилей за последние годы и определил тенденции данного рынка [1]. Спрогнозировали цены на поддержанные машины с использованием машинного обучения Ю.В. Осипова и В.С. Бардина [2]. Е.А. Чепыгов статью посвятил решению современной задачи построения информационной системы для оценки в режиме реального времени стоимости транспортного средства [3]. Рассмотрел стратегические и тактические задачи, которые решаются с помощью анализа данных С.А. Иванов [4]. А.А. Овсянникова и Д.В. Домашова исследовали применение методов машинного обучения, включая нейронные сети, для прогнозирования результатов исполнения государственных контрактов в сфере трубной промышленности [5].

1.3 Цель исследования

Цель исследования - разработать и применить регрессионные модели машинного обучения для прогнозирования цен на легковые автомобили в Еврейской автономной области.

2 Материалы и методы

Для разработки и применения регрессионной модели машинного обучения для прогнозирования цен на легковые автомобили используется интерактивная облачная среда Google Colaboratory. Работа будет происходить на готовом наборе данных об ценах и характеристиках автомобилей:
https://docs.google.com/spreadsheets/d/1U4iAmEDIaadohFus-n7-fUIQhAj5gSlV/edit?usp=sharing&ouid=104272149632818699735&rtpof=true&sd=true

3 Результаты и обсуждения

Для начала работы были подключены необходимые библиотеки и модели (Рис.1).

```
# 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 1. Импорт библиотек и моделей

Подключаем датасет с данными, проверяем количество строк и столбцов, и отображаем 5 первых строк (Рис. 2).

```
[ ] # Подключение датасета
    car_dataset= pd.read_excel('Датасет.xlsx')

[ ] car_dataset.shape

(200, 14)

[ ] car_dataset.head()
```

	Car_brand	Car_Model	Price	Year	Power	Engine_capacity	Fuel	Additional_fuel	Transmission	Drive	Color	Mileage	Wheel	Body_type
0	Toyota	Corona Premio	470000	1986	260	2.0	бензин	нет	механика	передний	серый	300000	правый	Седан
1	Nissan	Teana	1190000	2010	182	2.5	бензин	нет	вариатор	передний	серый	260000	левый	Седан
2	Honda	Airwave	610000	2006	110	1.5	бензин	нет	вариатор	передний	зеленый	186000	правый	Универсал
3	Honda	Vezel	1430000	2014	132	1.5	бензин	гибрид	робот	передний	зеленый	175898	правый	SUV
4	Honda	Accord	1860000	2013	143	2.0	бензин	гибрид	вариатор	передний	черный	119000	правый	Седан

Рисунок 2. Подключение и проверка датасета

Проверяем тип данных столбцов (Рис.3).

```
car_dataset.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 200 entries, 0 to 199
Data columns (total 14 columns):
 #   Column           Non-Null Count  Dtype
---  ------           --------------  -----
 0   Car_brand        200 non-null    object
 1   Car_Model        200 non-null    object
 2   Price            200 non-null    int64
 3   Year             200 non-null    int64
 4   Power            200 non-null    int64
 5   Engine_capacity  200 non-null    float64
 6   Fuel             200 non-null    object
 7   Additional_fuel  200 non-null    object
 8   Transmission     200 non-null    object
 9   Drive            200 non-null    object
 10  Color            200 non-null    object
 11  Mileage          200 non-null    int64
 12  Wheel            200 non-null    object
 13  Body_type        200 non-null    object
dtypes: float64(1), int64(4), object(9)
memory usage: 22.0+ KB
```

Рисунок 3. Тип данных столбцов

После определения типов столбцов, можно выделить список столбцов, которые необходимо преобразовать в числовые данные (Рис. 4).

```
[ ]  # Определение списка столбцов, которые требуется преобразовать
     columns_to_convert = ['Fuel', 'Transmission', 'Drive', 'Additional_fuel', 'Wheel', 'Body_type']

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

Рисунок 4. Преобразование данных

Также необходимо преобразовать столбцы Car_brand, Car_Model и Color, для того чтобы вычислить корреляционную матрицу и визуализировать её в виде тепловой карты с помощью библиотеки Seaborn (Рис. 5).

```
[ ]  # 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
         car_dataset[column] = label_encoder.fit_transform(car_dataset[column])
```

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

     # Применение LabelEncoder к столбцу 'Car_brand' для преобразования названий брендов автомобилей в числовые коды
     car_dataset['Car_brand'] = label_encoder.fit_transform(car_dataset['Car_brand'])

     # Применение LabelEncoder к столбцу 'Car_Model' для преобразования названий моделей автомобилей в числовые коды
     car_dataset['Car_Model'] = label_encoder.fit_transform(car_dataset['Car_Model'])

     # Применение LabelEncoder к столбцу 'Color' для преобразования названий цветов автомобилей в числовые коды
     car_dataset['Color'] = label_encoder.fit_transform(car_dataset['Color'])
```

Рисунок 5. Преобразование столбцов

После преобразования данных, проведем вычисление и визуализацию корреляционной матрицы (Рис.6).

```
# Вычисление корреляционной матрицы
correlation_matrix =  car_dataset.corr()

# Визуализация корреляционной матрицы
plt.figure(figsize=(10, 8))
sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm', fmt=".2f", annot_kws={"size": 10})
plt.title('Корреляционная матрица')
plt.show()
```

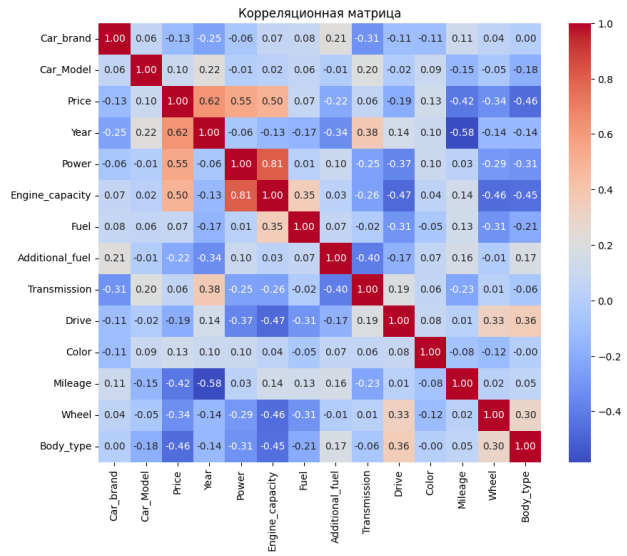


Рисунок 6. Корреляционная матрица

С помощью результатов корреляционной матрицы можно сделать следующие выводы:

Цена автомобилей положительно коррелирует с Годом выпуска (0.62), Мощностью (0.55) и Объемом двигателя (0.50), указывая на то, что более новые и мощные автомобили с большим двигателем дороже.

Год выпуска также коррелирует с Мощностью, подтверждая, что новые автомобили часто обладают лучшими техническими характеристиками.

Мощность сильно коррелирует с Объемом двигателя (0.81), что логично, учитывая, что объем двигателя влияет на мощность.

Пробег отрицательно коррелирует с Ценой (-0.42) и Годом выпуска (-0.58), что означает, что автомобили с меньшим пробегом и более новые стоят дороже.

Тип кузова и Колесная база отрицательно коррелируют с Ценой, указывая на возможное влияние этих параметров на стоимость.

Трансмиссия и Привод также коррелируют с некоторыми характеристиками, что важно учитывать при анализе цен.

Бренд и Модель имеют слабую корреляцию с другими переменными, что может указывать на их меньшее влияние на цену по сравнению с другими характеристиками.

Далее с помощью визуализации проведем разведочный анализ данных.

Для начало визуализируем распределение цен на автомобили. На данном графики можно определить, что в диапазоне до 1 млн.руб частота больше всего (Рис.7).

Рисунок 7. Гистограмма распределения цен на автомобили

График зависимости цены автомобилей от мощности двигателя, построенный с помощью диаграммы рассеяния и линейной регрессии, демонстрирует общую тенденцию, согласно которой автомобили с более высокой мощностью двигателя, как правило, имеют более высокую цену. Линия регрессии с положительным наклоном указывает на прямую зависимость между этими двумя переменными: чем больше мощность, тем выше средняя цена автомобиля (Рис.8).

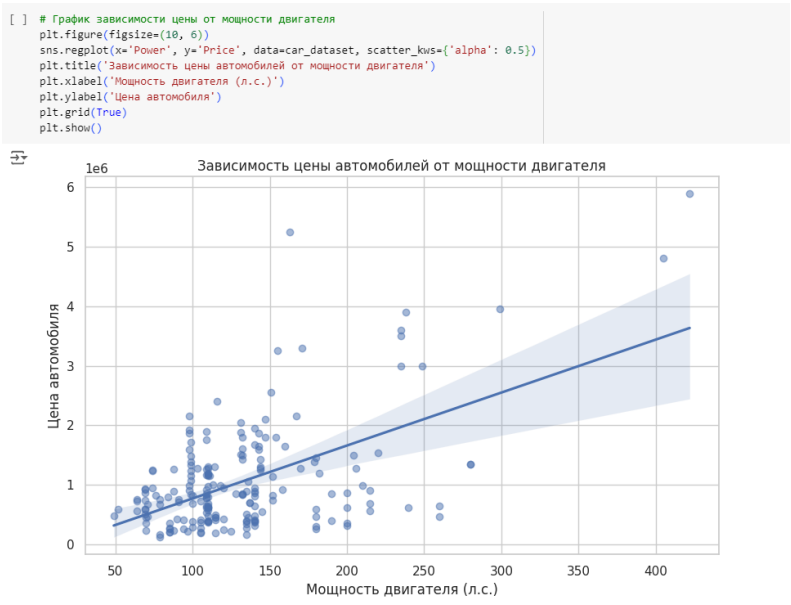


Рисунок 8. График зависимости цены автомобиля от мощности двигателя

График, построенный с использованием диаграммы рассеяния и линейной регрессии, показывает, что автомобили, выпущенные в более поздние годы, обычно имеют более высокую цену. Линия регрессии с восходящим наклоном указывает на то, что с течением времени средняя цена автомобилей имеет тенденцию к увеличению (Рис.9).

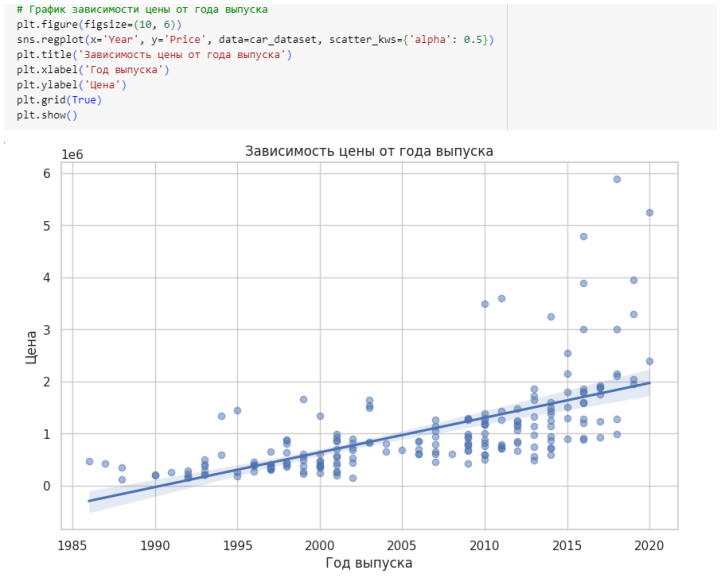


Рисунок 9. График зависимости цены автомобиля от года выпуска

График зависимости цены от объема двигателя отражает общее наблюдение, что автомобили с двигателями большего объема, как правило, обладают более высокой ценой. Линия регрессии с наклоном вверх указывает на то, что по мере увеличения объема двигателя средняя цена автомобилей имеет тенденцию к возрастанию (Рис.10).

```
# График зависимости цены от объема двигателя
plt.figure(figsize=(10, 6))
sns.regplot(x='Engine_capacity', y='Price', data=car_dataset, scatter_kws={'alpha': 0.5})
plt.title('Зависимость цены от объема двигателя')
plt.xlabel('Объем двигателя')
plt.ylabel('Цена')
plt.grid(True)
plt.show()
```

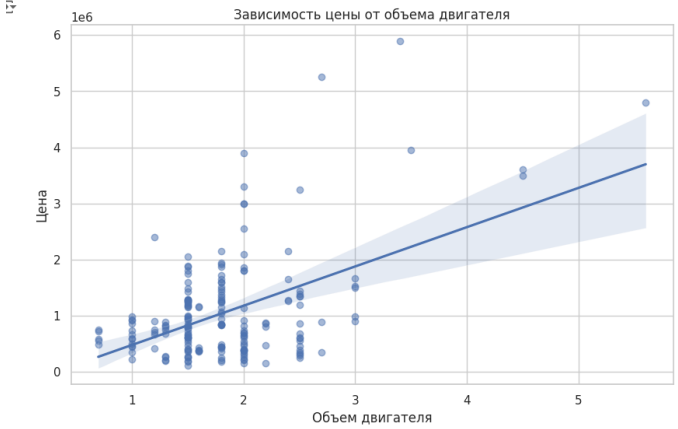


Рисунок 10. График зависимости цены автомобиля от объема двигателя

График зависимости цены от пробега демонстрирует общее наблюдение, что автомобили с высоким пробегом, как правило, продаются по более низким ценам. Линия регрессии с наклоном вниз указывает на то, что с возрастанием пробега средняя цена автомобиля имеет тенденцию к снижению (Рис.11).

```
# График зависимости цены от пробега
plt.figure(figsize=(10, 6))
sns.regplot(x='Mileage', y='Price', data=car_dataset, scatter_kws={'alpha': 0.5})
plt.title('Зависимость цены от пробега')
plt.xlabel('Пробег')
plt.ylabel('Цена')
plt.grid(True)
plt.show()
```

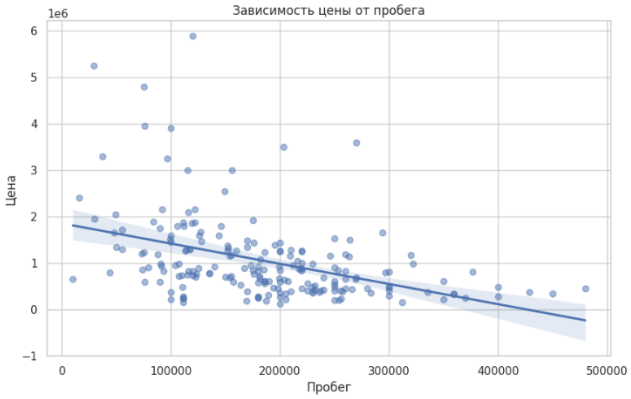


Рисунок 11. График зависимости цены автомобиля от пробега

На графике зависимости цены автомобиля от типа топлива, можно сделать вывод, что дизель (1) расположен выше, чем бензин (0), это может

указывать на то, что дизельные автомобили, как правило, стоят дороже (Рис. 12).

```
# Box plot для сравнения цен в зависимости от типа топлива
plt.figure(figsize=(10, 6))
sns.boxplot(x='Fuel', y='Price', data=car_dataset)
plt.title('Сравнение цен в зависимости от типа топлива')
plt.xlabel('Тип топлива')
plt.ylabel('Цена')
plt.show()
```

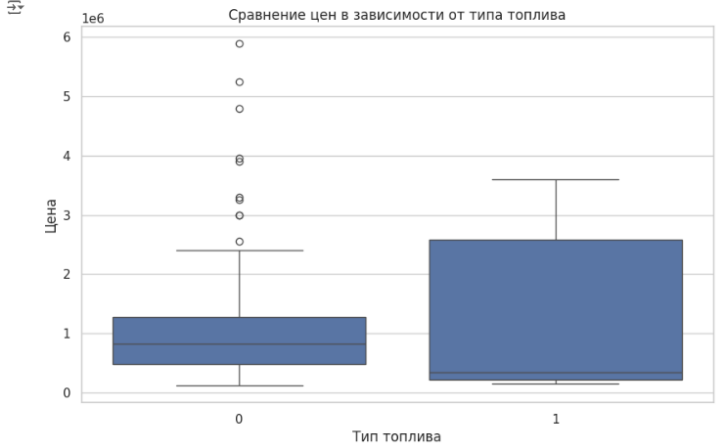


Рисунок 12. Box plot зависимости цены автомобиля от типа топлива

На графике зависимости цены автомобиля от трансмиссии, можно сделать вывод, что тип трансмиссии вариатор (1) расположен выше других, это может указывать на то, что автомобили с вариатором, как правило, стоят дороже, далее тип трансмиссии робот (3) расположен выше типов автомата (0) и механики (2), это говорит о том, что автомобиль с данной трансмиссией стоит дороже, чем автомобиль с типом трансмиссии автомат или механика. Ниже всего расположен тип трансмиссии механика (2), это значит, что автомобиль с механической коробкой передач имеет низкую цену (Рис. 13).

```
# Box plot для сравнения цен в зависимости от типа трансмиссии
plt.figure(figsize=(10, 6))
sns.boxplot(x='Transmission', y='Price', data=car_dataset)
plt.title('Сравнение цен в зависимости от типа трансмиссии')
plt.xlabel('Тип трансмиссии')
plt.ylabel('Цена')
plt.show()
```

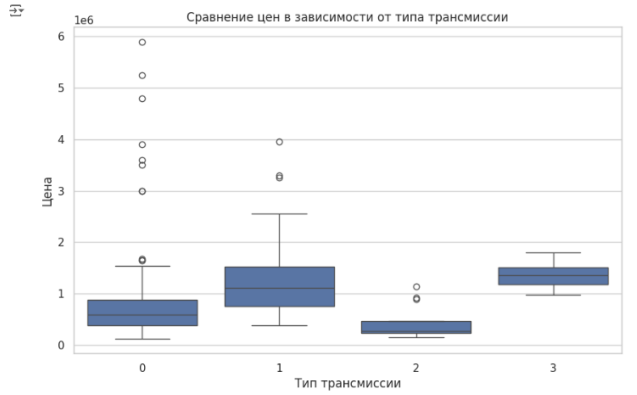


Рисунок 13. Box plot зависимости цены автомобиля от типа трансмиссии

Далее приступим к обучению моделей. Для обучения были выбраны 5 моделей:

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

Для обучения моделей были определены списки категориальных и числовых признаков и создано преобразование (Рис.14).

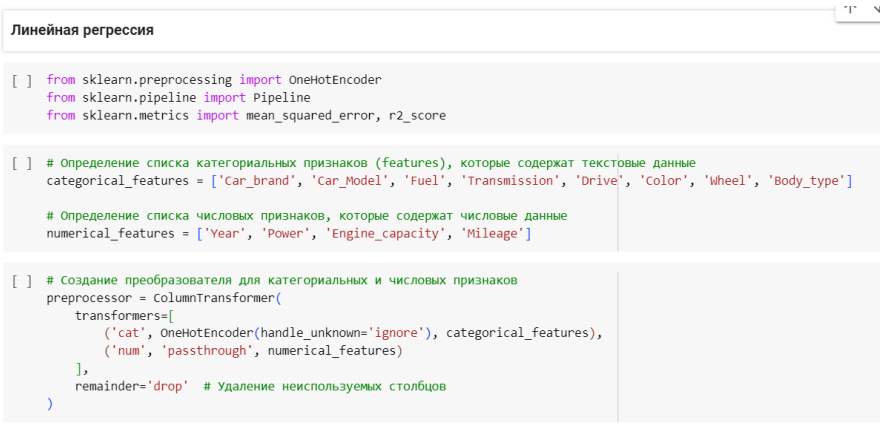


Рисунок 14. Подготовка данных к обучению моделей

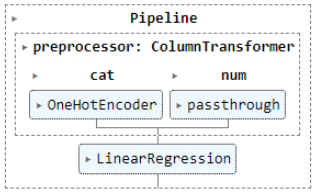
Далее после подготовки данных, была создана и обучена модель линейной регрессии (Рис.15).

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

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

[ ] X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

[ ] model.fit(X_train, y_train)
```




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

[ ] 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

[ ] 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): 133033065742.36787
Среднеквадратическая ошибка (RMSE): 364736.98159409047
Средняя абсолютная ошибка (MAE): 249741.83408852294
Коэффициент детерминации (R^2): 0.7432992352525023
```

Рисунок 15. Модель линейной регрессии

Далее создадим и обучим модель случайного леса (Рис.16).

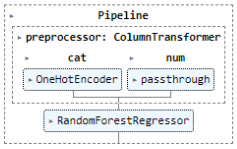
Случайный лес

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

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

[ ] # 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=42)

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




```
[ ] # Предсказание на тестовой выборке
    y_pred = model.predict(X_test)

[ ] # Оценка модели
    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

[ ] 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): 92843614274.91856
Среднеквадратическая ошибка (RMSE): 304702.5012613427
Средняя абсолютная ошибка (MAE): 172797.30675
Коэффициент детерминации (R^2): 0.8208488494698876
```

Рисунок 16. Модель случайного леса

Создадим и обучим модель XGBoost (Рис.17).

Рисунок 17. Модель XGBoost

Также создали и обучили модель KNeighborsRegressor (Рис.18).

Рисунок 18. Модель KNeighborsRegressor

Далее создадим и обучим модель DecisionTreeRegressor (Рис.19).

Рисунок 19. Модель DecisionTreeRegressor

После обучения моделей была создана таблица, которая содержит все результаты полученные в процессе обучения моделей (Рис. 20).

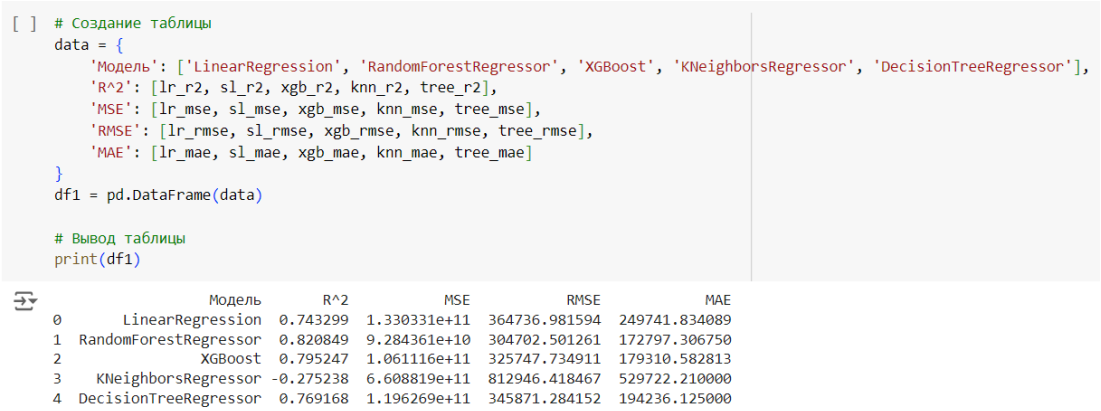


Рисунок 20. Вывод полученных результатов

Также были обучены 5 моделей с логарифмом цены. Для начала был создан новый столбец с логарифмом цены, были преобразованы категориальные переменные (Рис.21).

Линейная регрессия с логарифмом цены

```
[ ] # Создание новой колонки с логарифмом цены
    car_dataset['Log_Price'] = np.log(car_dataset['Price'])

[ ] # Просмотр данных
    print(car_dataset[['Price', 'Log_Price']].head())

      Price  Log_Price
0   470000  13.060488
1  1190000  13.989464
2   610000  13.321214
3  1430000  14.173185
4  1860000  14.436087

[ ] # Преобразование категориальных переменных
    categorical_cols = ['Car_brand', 'Car_Model', 'Fuel', 'Additional_fuel', 'Transmission', 'Drive', 'Color', 'Wheel', 'Body_type']

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

Рисунок 21. Подготовка данных

После подготовки данных была создана и обучена модель линейной регрессии с логарифмом цены (Рис.22).

```
X = car_dataset.drop(['Price', 'Log_Price'], axis=1)
y_log = car_dataset['Log_Price']

X_train, X_test, y_train, y_test = train_test_split(X, y_log, test_size=0.2, random_state=42)

# Инициализация модели линейной регрессии
lr_model = LinearRegression()

# Обучение модели
lr_model.fit(X_train, y_train)

▾ LinearRegression
LinearRegression()

# 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_mae = mean_absolute_error(y_test, predictions_log)
lrl_mse = mean_squared_error(y_test, predictions_log)
lrl_rmse = np.sqrt(lrl_mse)
lrl_r2 = r2_score(y_test, predictions_log)

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

Среднеквадратическая ошибка (RMSE): 0.31188325087954855
Среднеквадратическая ошибка (MSE): 0.0972711621791954
Средняя абсолютная ошибка (MAE): 0.24858490842343092
Коэффициент детерминации (R^2): 0.7988765398638653
```

Рисунок 22. Модель линейная регрессия с логарифмом цены

Также была обучена модель случайный лес с логарифмом цены (Рис.23).

Случайный лес с логарифмом цены

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

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

    RandomForestRegressor
    RandomForestRegressor(random_state=42)

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

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

    # Оценка модели
    sll_mae = mean_absolute_error(y_test, predictions_log)
    sll_mse = mean_squared_error(y_test, predictions_log)
    sll_rmse = np.sqrt(sll_mse)
    sll_r2 = r2_score(y_test, predictions_log)

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

    Среднеквадратическая ошибка (RMSE): 0.27393022974850334
    Среднеквадратическая ошибка (MSE): 0.07503777077006783
    Средняя абсолютная ошибка (MAE): 0.19999468182763472
    Коэффициент детерминации (R^2): 0.8448475811322625
```

Рисунок 23. Модель случайный лес с логарифмом цены

Была создана и обучена модель XGBoost (Рис.24).

XGBoost с логарифмом цены

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

    # Обучение модели
    xgb_model.fit(X_train, y_train)

    XGBRegressor

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

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

[ ] # Оценка модели
    xgbl_mae = mean_absolute_error(y_test, predictions_log)
    xgbl_mse = mean_squared_error(y_test, predictions_log)
    xgbl_rmse = np.sqrt(xgbl_mse)
    xgbl_r2 = r2_score(y_test, predictions_log)

[ ] print(f'Среднеквадратическая ошибка (RMSE): {xgbl_rmse}')
    print(f'Среднеквадратическая ошибка (MSE): {xgbl_mse}')
    print(f'Средняя абсолютная ошибка (MAE): {xgbl_mae}')
    print(f'Коэффициент детерминации (R^2): {xgbl_r2}')

    Среднеквадратическая ошибка (RMSE): 0.3538789638459603
    Среднеквадратическая ошибка (MSE): 0.12523032105269047
    Средняя абсолютная ошибка (MAE): 0.24231023006546923
    Коэффициент детерминации (R^2): 0.741066571840928
```

Рисунок 24. Модель XGBoost с логарифмом цены

Создана и обучена модель KNeighborsRegressor с логарифмом цены (Рис.25).

Рисунок 25. Модель KNeighborsRegressor с логарифмом цены

Также была создана и обучена модель DecisionTreeRegressor с логарифмом цены (Рис.26).

Рисунок 26. Модель DecisionTreeRegressor с логарифмом цены

После обучения моделей была создана таблица, которая содержит все результаты полученные в процессе обучения моделей с логарифмом цены (Рис. 27).

```
[ ] # Создание таблицы
    datalog = {
        'Модель': ['LinearRegression (log)', 'RandomForestRegressor (log)', 'DecisionTreeRegressor (log)', 'KNeighborsRegressor (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)  0.798877  0.097271  0.311883  0.248585
1  RandomForestRegressor (log)  0.844848  0.075038  0.273930  0.199995
2  DecisionTreeRegressor (log)  0.741067  0.125230  0.353879  0.242310
3    KNeighborsRegressor (log)  0.038378  0.465078  0.681966  0.528631
4               XGBoost (log)  0.597870  0.194486  0.441006  0.306905
```

Рисунок 27. Вывод полученных результатов

Выводы

В данной работе был проведен комплексный анализ различных моделей регрессии с целью определения наиболее эффективного инструмента для предсказания цен на автомобили. Исследование включало оценку моделей LinearRegression, RandomForestRegressor, DecisionTreeRegressor, KNeighborsRegressor и XGBoost, с использованием как исходных данных о ценах, так и данных с логарифмически преобразованной ценой.

Библиографический список

1. Невежин В. П. Прогнозирование стоимости легковых автомобилей среднего класса //Автомобильная промышленность. 2020. №. 6. С. 1-5.
2. Осипова, Ю. В. Прогнозирование цен на поддержанные машины с использованием машинного обучения / Ю. В. Осипова, В. С. Бардина // Первый шаг в большую науку: сборник статей II Международного научно-исследовательского конкурса, Пенза, 15 января 2024 года. Пенза: Наука и Просвещение (ИП Гуляев Г.Ю.), 2024. С. 17-20.
3. Чепыгов Е. А. Разработка интеллектуальной информационной системы для онлайн-оценки стоимости транспортного средства //Информационно-телекоммуникационные технологии и математическое моделирование высокотехнологичных систем. 2020. С. 218-222.
4. 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.
5. 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.