УДК 004

Модели машинного обучения для датасета Автомобили

Бушманова Татьяна Сергеевна
Приамурский государственный университет имени Шолом-Алейхема
Студент

Аннотация
В данной статье приведён процесс обучения системы, предсказывающей цену автомобилей на основании характеристик при помощи собственного датасета, на 200 записей. В результате исследования выбрана модель и переменные, обеспечивающие лучшие результаты для текущей задачи. Сравнение моделей произведено на основании основных метрик: R2, RMSE, MAE, MSE.
Ключевые слова: Регрессионная модель, машинное обучение, Colab, датасет.

Machine learning models for the Car dataset

Bushmanova Tatyana Sergeevna
Sholom-Aleichem Priamursky State University
Student

Abstract
This article describes the process of training a system that predicts the price of a monitor based on characteristics using its own dataset, for 200 records. 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, RMSE, MAE, MSE
Keyword: Regression model, machine learning, Colab, dataset**.**

1 Введение

1.1 Актуальность

В данной статье будет рассмотрен процесс машинного обучения для предсказания цены на автомобили. Для предсказания цены автомобиля на основе его характеристик можно использовать методы машинного обучения, такие как линейная регрессия, случайные леса или градиентный бустинг.

Прежде всего необходимо провести подготовительный анализ данных, включающий в себя очистку и предварительную обработку данных, такую как заполнение пропущенных значений, масштабирование признаков и кодирование категориальных переменных.

Далее можно разделить данные на обучающий и тестовый наборы, обучить выбранную модель на обучающих данных, произвести тюнинг для

улучшения качества предсказаний, и, наконец, оценить производительность модели на тестовом наборе данных.

1.2 Обзор исследований

К.А. Найденова и О.А. Невзорова рассматривают современные методы машинного обучения, применяемые в задачах обработки естественного языка [1].

Е.М. Аксютина и Ю.С. Белов в своей статье рассматривают основные архитектуры, применяемые для анализа больших данных, их особенности и ограничения, а также выявляются требования к методам машинного обучения, выполнение которых позволит применять их при анализе больших данных [2].

В.И. Донской провел сравнительный анализ различных определений обучаемости, рассмотрены необходимые и достаточные условия обучаемости, указаны границы применимости VC теории [3].

Б.И. Гельцер, М.М. Циванюк представили анализ научной литературы по результатам использования методов машинного обучения [4].

В.С. Старостин цифровая трансформация маркетинга в эпоху машинного интеллекта [5].

1.3 Цель исследования

Цель исследования - разработка модели машинного обучения для предсказания цены автомобиля на основе его характеристик.

2 Материалы и методы

Для реализации приложения используется язык программирования Python, и средство для разработки colaboratory. Датасет создан самостоятельно.

https://colab.research.google.com/drive/1vg7jnoPjPkNAbdYVHRuekYdrr-L3wY74?usp=sharing –ссылка на colab.

https://docs.google.com/document/d/1yW-UTVNDcbH_wDYd5ZANsbwnhV1QVEN_fmGSY4-Eyw/edit?usp=sharing – на датасет.

3 Результаты и обсуждения

Делаем импорт необходимых библиотек для загрузки библиотек и модулей, которые предоставляют инструменты для работы с данными (рис. 1). Код на рисунке отвечает за импорт необходимых библиотек и модулей, построения модели линейной регрессии и оценки ее качества. Библиотека pandas используется для работы с табличными данными, numpy - для выполнения численных вычислений, а scikit-learn предоставляет инструменты для машинного обучения и оценки моделей. Импортированные функции и метрики оценки моделей используются для разделения данных на обучающий и тестовый наборы, создания модели линейной регрессии и вычисления метрик ее качества, таких как среднеквадратичная ошибка,

средняя абсолютная ошибка и коэффициент детерминации. Библиотека matplotlib используется для визуализации данных.

```
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error, mean_absolute_error, r2_score
import matplotlib.pyplot as plt
```

Рисунок 1 – Импорт необходимых библиотек

Далее выполняем загрузку данных из датасета carr.xlsx с использованием библиотеки pandas (рис.2). Функция используется для чтения данных из Excel-файла в формате DataFrame. После загрузки данных происходит проверка корректности загрузки и вывод первых нескольких строк данных с помощью метода. Это позволяет быстро ознакомиться с структурой данных и убедиться, что они были успешно загружены.

```
# Загрузка данных из файла carr.xlsx
data = pd.read_excel('carr.xlsx')

# Проверка загрузки и вывод первых нескольких строк данных
print(data.head())
```

```
     name     model transmission   color  Year of release engine_type  \
0  Subaru   Outback    automatic  silver             2010    gasoline
1  Subaru   Outback    automatic    blue             2002    gasoline
2  Subaru  Forester    automatic     red             2001    gasoline
3    LADA     Vesta    automatic    grey             2017    gasoline
4    LADA    Largus   mechanical  silver             2014    gasoline

   Dvigatel   the body     cena     log10
0       2.5  universal  1800000  6.255273
1       3.0  universal   450000  5.653213
2       2.0        suv   760000  5.880814
3       1.6      sedan   930000  5.968483
4       1.6  universal   720000  5.857332
```

Рисунок 2 – Загрузка данных из датасета

Затем выполняем анализ данных. Сначала вычисляются основные статистические показатели для каждого числового столбца с помощью метода, который выводит среднее значение, стандартное отклонение, минимальное и максимальное значения, квартили и т. д. Затем создаются гистограммы для каждого числового столбца. Параметр «figsize» определяет размеры создаваемой фигуры. После этого вызывается функция для отображения гистограмм. Это помогает понять распределение данных и выявить возможные аномалии или особенности в данных.

```
# Статистика о данных
print(data.describe())

# Графики и распределения
data.hist(figsize=(10, 10))
plt.show()
```

```
       Year of release    Dvigatel          cena       log10
count       199.000000  198.000000  1.990000e+02  199.000000
mean       2006.502513    2.151515  1.182394e+06    5.903293
std           8.510762    0.833665  1.521136e+06    0.370547
min        1972.000000    1.000000  8.000000e+04    4.903090
25%        2001.000000    1.600000  4.900000e+05    5.690196
50%        2008.000000    2.000000  8.200000e+05    5.913814
75%        2013.000000    2.500000  1.335000e+06    6.125332
max        2023.000000    5.600000  1.600000e+07    7.204120
```

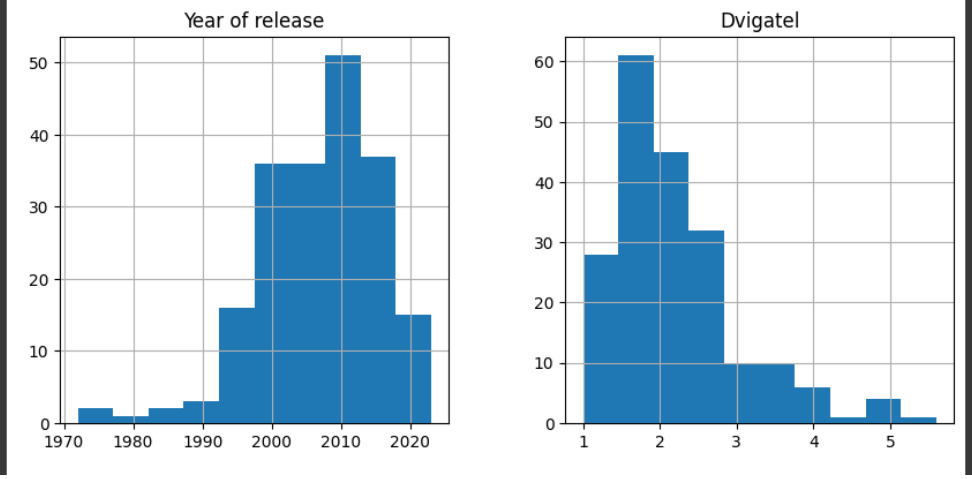

Рисунок 3 – Анализ данных

Необходимо выполнить предобработку данных для моделирования (рис. 4). Код кодирует категориальные переменные, выбирает признаки и целевую переменную, а затем логарифмирует целевую переменную для улучшения ее распределения, что может быть полезно для моделей, таких как линейная регрессия.

```
# Кодирование категориальных переменных
data_encoded = pd.get_dummies(data)

# Выбор признаков и целевой переменной
X = data_encoded[['Year of release']]
y = data_encoded['cena']

# Логарифмирование целевой переменной (для линейной регрессии)
y_log = np.log10(y)
```

Рисунок 4 – Предобработка данных для моделирования

Выполним разделение данных на обучающий и тестовый наборы. Параметр test_size = 0.2 указывает, что 20% данных будут использоваться для тестирования, а оставшиеся 80% - для обучения модели. Создаем модель линейной регрессии и обучаем ее на обучающих данных. После выполнения этого кода, модель будет готова для предсказаний на новых данных (рис. 5).

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

# Создание и обучение модели
model = LinearRegression()
model.fit(X_train, y_train)

LinearRegression
LinearRegression()
```

Рисунок 5 – Обучение модели

Выполним предсказание целевой переменной на тестовом наборе данных с использованием обученной модели. После получения прогнозов, вычисляются различные метрики оценки качества модели, такие как среднеквадратичная ошибка (MSE), средняя абсолютная ошибка (MAE), корень из среднеквадратичной ошибки (RMSE) и коэффициент детерминации (R2 score). Затем значения этих метрик выводятся на экран. Эти метрики позволяют оценить точность и эффективность модели на тестовых данных (рис. 6).

```
# Предсказание на тестовом наборе
y_pred = model.predict(X_test)

# Метрики оценки модели
mse = mean_squared_error(y_test, y_pred)
mae = mean_absolute_error(y_test, y_pred)
rmse = np.sqrt(mse)
r2 = r2_score(y_test, y_pred)

print("MSE:", mse)
print("MAE:", mae)
print("RMSE:", rmse)
print("R2 score:", r2)

MSE: 0.04194030378333637
MAE: 0.15757435159818203
RMSE: 0.20479331967458406
R2 score: 0.5820495982226115
```

Рисунок 6 – Вывод метрик

Выполним предсказания с двумя фичами. В данном случае создадим новый датафрейм, который содержит только два выбранных признака: год выпуска автомобиля и тип двигателя. Выполняем предсказание целевой переменной на тестовом наборе данных с использованием модели, построенной на двух выбранных признаках. Затем вычислим различные

метрики оценки качества модели с использованием только этих двух признаков (рис. 7).

```
# Предсказание на тестовом наборе для модели с двумя фичами
y_pred2 = model2.predict(X_test2)

# Метрики оценки модели с двумя фичами
mse2 = mean_squared_error(y_test2, y_pred2)
mae2 = mean_absolute_error(y_test2, y_pred2)
rmse2 = np.sqrt(mse2)
r2_2 = r2_score(y_test2, y_pred2)

print("MSE (две фичи):", mse2)
print("MAE (две фичи):", mae2)
print("RMSE (две фичи):", rmse2)
print("R2 score (две фичи):", r2_2)
```

```
MSE (две фичи): 0.04292756162597193
MAE (две фичи): 0.15960935065647935
RMSE (две фичи): 0.20718967548112027
R2 score (две фичи): 0.5722112142633766
```

Рисунок 7 – Предсказание с двумя фичами

Попробуем использовать для прогнозирования и другие модели. Создадим модель случайного леса для решения задачи. Импортируем соответствующий класс. Затем создаем экземпляр модели с заданным параметром, который обеспечивает воспроизводимость результатов. Далее обучаем модель и выводим метрики (рис. 8).

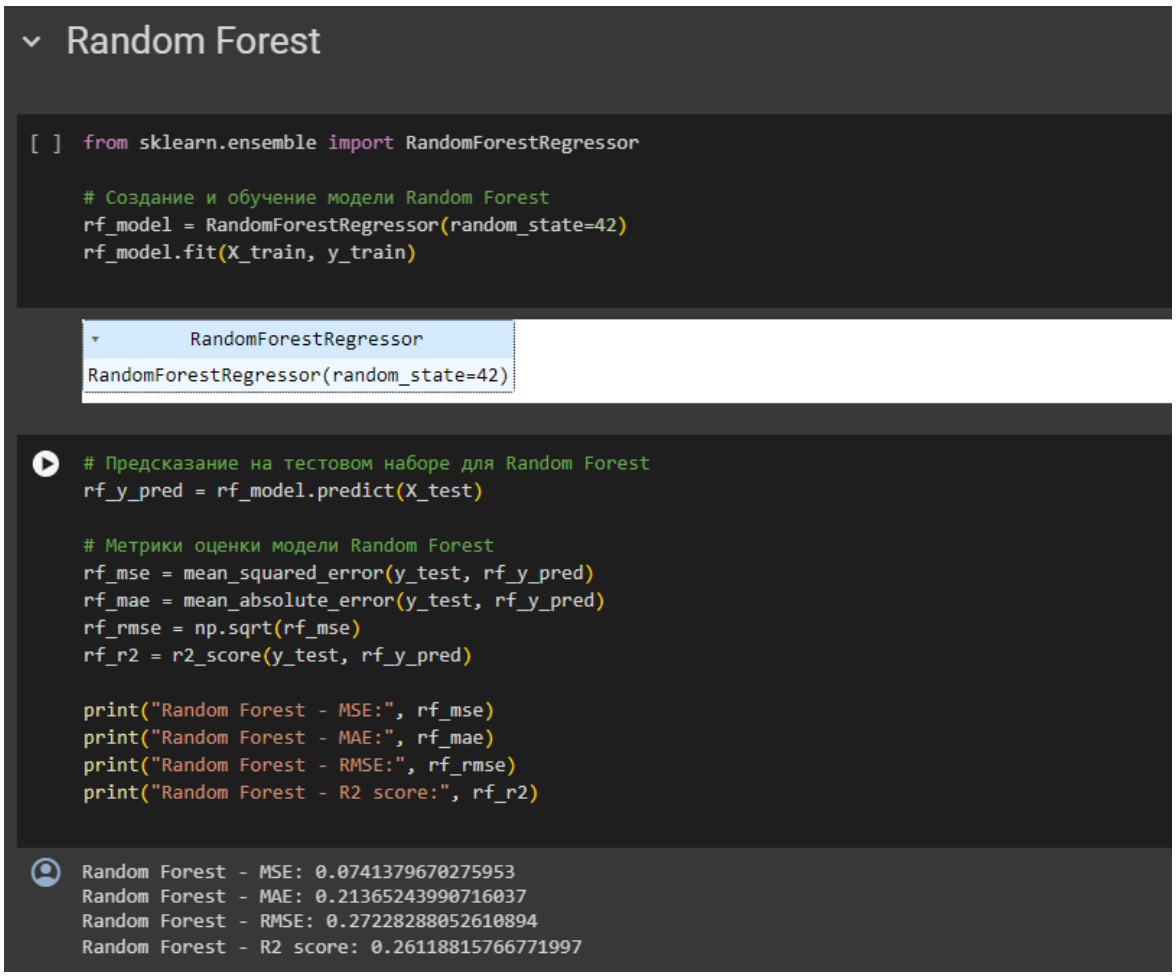

Рисунок 8 – Использование модели случайного леса

Далее код создает и обучает модель дерева решений для задачи регрессии. Затем выполняется предсказание целевой переменной на тестовом наборе данных, а затем вычисляются и выводятся метрики оценки качества модели, такие как MSE, MAE, RMSE и R2 (рис. 9).

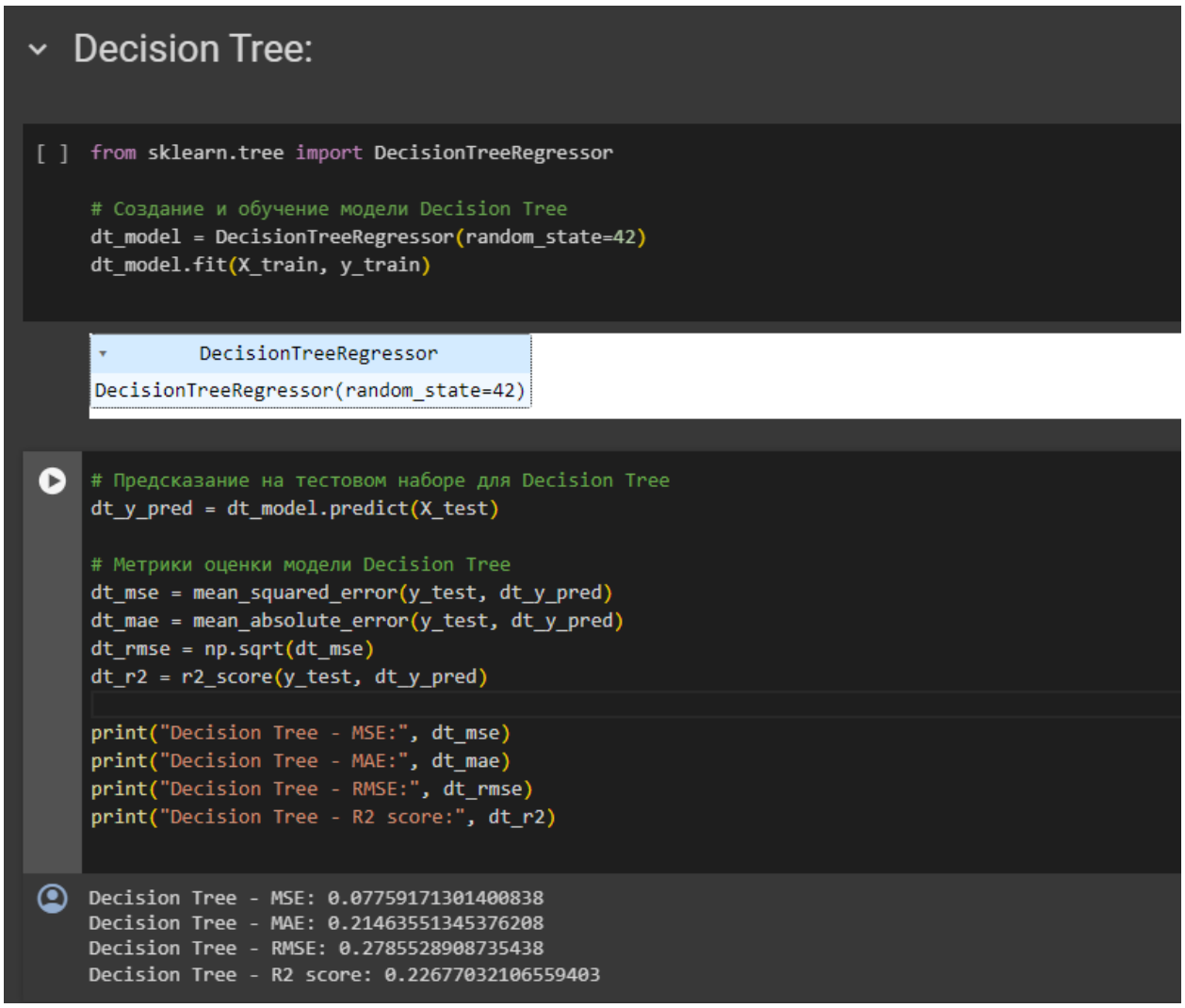

Рисунок 9 – Использование модели дерева решений

Также используем модель k-ближайших соседей (KNN) для задачи регрессии. Код выполняет предсказание целевой переменной на тестовом наборе данных, и вычисляются метрики оценки качества модели. Полученные значения метрик выводятся на экран для оценки эффективности модели KNN на тестовых данных (рис. 10).

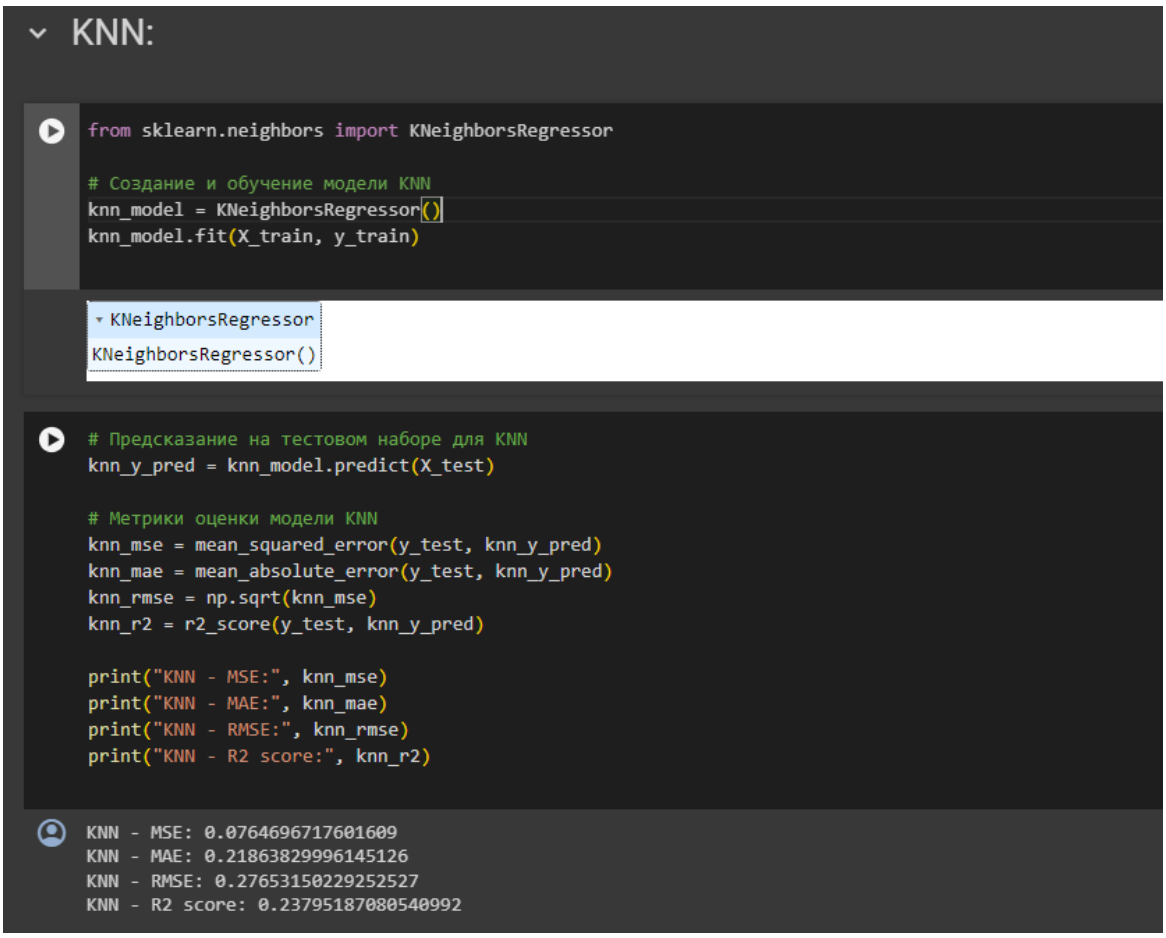

Рисунок 10 – Использование модели k-ближайших соседей

Создадим и обучим последнюю модель XGBoost. Затем выполняется предсказание целевой переменной на тестовом наборе данных, и вычисляются метрики оценки качества модели. Полученные значения метрик выводятся на экран для оценки эффективности модели XGBoost на тестовых данных.

XGBoost:

```
from xgboost import XGBRegressor

# Создание и обучение модели XGBoost
xgb_model = XGBRegressor(random_state=42)
xgb_model.fit(X_train, y_train)
```

```
▸ XGBRegressor
```

```
[ ] # Предсказание на тестовом наборе для XGBoost
    xgb_y_pred = xgb_model.predict(X_test)

    # Метрики оценки модели XGBoost
    xgb_mse = mean_squared_error(y_test, xgb_y_pred)
    xgb_mae = mean_absolute_error(y_test, xgb_y_pred)
    xgb_rmse = np.sqrt(xgb_mse)
    xgb_r2 = r2_score(y_test, xgb_y_pred)

    print("XGBoost - MSE:", xgb_mse)
    print("XGBoost - MAE:", xgb_mae)
    print("XGBoost - RMSE:", xgb_rmse)
    print("XGBoost - R2 score:", xgb_r2)
```

```
XGBoost - MSE: 0.07741793504630061
XGBoost - MAE: 0.21441964946051
XGBoost - RMSE: 0.2782407860941681
XGBoost - R2 score: 0.22850208180339726
```

Рисунок 11 – Использование модели XGBoost

Построение моделей с прологарифмированной ценой.

С другими фичами

```
[ ] # Добавление других фичей
    X3 = data_encoded[['Year of release', 'engine_type_gasoline', 'transmission_automatic']]

    # Логарифмирование целевой переменной
    y_log = data_encoded['log10']
```

```
[ ] # Разделение на train и test
    X_train3, X_test3, y_train3, y_test3 = train_test_split(X3, y_log, test_size=0.2, random_state=42)
```

```
[ ] # Создание и обучение модели Random Forest
    rf_model3 = RandomForestRegressor(random_state=42)
    rf_model3.fit(X_train3, y_train3)
```

```
▾ RandomForestRegressor
RandomForestRegressor(random_state=42)
```

Рисунок 12. Пробуем с другими фичами

```
[ ]  # Предсказание на тестовом наборе для Random Forest
     rf_y_pred3 = rf_model3.predict(X_test3)

     # Метрики оценки модели Random Forest
     rf_mse3 = mean_squared_error(y_test3, rf_y_pred3)
     rf_mae3 = mean_absolute_error(y_test3, rf_y_pred3)
     rf_rmse3 = np.sqrt(rf_mse3)
     rf_r23 = r2_score(y_test3, rf_y_pred3)

     print("Random Forest - MSE:", rf_mse3)
     print("Random Forest - MAE:", rf_mae3)
     print("Random Forest - RMSE:", rf_rmse3)
     print("Random Forest - R2 score:", rf_r23)
```

```
Random Forest - MSE: 0.06401756835351638
Random Forest - MAE: 0.18438170136834003
Random Forest - RMSE: 0.25301693293832406
Random Forest - R2 score: 0.36204161597746365
```

```
[ ]  # Создание и обучение модели Decision Tree
     dt_model3 = DecisionTreeRegressor(random_state=42)
     dt_model3.fit(X_train3, y_train3)
```

```
DecisionTreeRegressor
DecisionTreeRegressor(random_state=42)
```

```
[ ]  # Предсказание на тестовом наборе для Decision Tree
     dt_y_pred3 = dt_model3.predict(X_test3)

     # Метрики оценки модели Decision Tree
     dt_mse3 = mean_squared_error(y_test3, dt_y_pred3)
     dt_mae3 = mean_absolute_error(y_test3, dt_y_pred3)
     dt_rmse3 = np.sqrt(dt_mse3)
     dt_r23 = r2_score(y_test3, dt_y_pred3)
```

Рисунок 13. Предсказание, метрики Random Forest

```
print("Decision Tree - MSE:", dt_mse3)
print("Decision Tree - MAE:", dt_mae3)
print("Decision Tree - RMSE:", dt_rmse3)
print("Decision Tree - R2 score:", dt_r23)
```

```
Decision Tree - MSE: 0.07086133811507966
Decision Tree - MAE: 0.20573232361354177
Decision Tree - RMSE: 0.2661979303358305
Decision Tree - R2 score: 0.293840957783151
```

```
[ ] # Создание и обучение модели KNN
knn_model3 = KNeighborsRegressor()
knn_model3.fit(X_train3, y_train3)
```

```
▾ KNeighborsRegressor
KNeighborsRegressor()
```

```
[ ] # Предсказание на тестовом наборе для KNN
knn_y_pred3 = knn_model3.predict(X_test3)

# Метрики оценки модели KNN
knn_mse3 = mean_squared_error(y_test3, knn_y_pred3)
knn_mae3 = mean_absolute_error(y_test3, knn_y_pred3)
knn_rmse3 = np.sqrt(knn_mse3)
knn_r23 = r2_score(y_test3, knn_y_pred3)

print("KNN - MSE:", knn_mse3)
print("KNN - MAE:", knn_mae3)
print("KNN - RMSE:", knn_rmse3)
print("KNN - R2 score:", knn_r23)
```

```
KNN - MSE: 0.06689625753775882
```

Рисунок 14. Предсказание и метрики KNN

```
KNN - MSE: 0.06689625753775882
KNN - MAE: 0.17950401136500005
KNN - RMSE: 0.25864310842889054
KNN - R2 score: 0.33335442982973174
```

```
[ ]  # Создание и обучение модели XGBoost
     xgb_model3 = XGBRegressor(random_state=42)
     xgb_model3.fit(X_train3, y_train3)
```

```
XGBRegressor(base_score=None, booster=None, callbacks=None,
             colsample_bylevel=None, colsample_bynode=None,
             colsample_bytree=None, device=None, early_stopping_rounds=None,
             enable_categorical=False, eval_metric=None, feature_types=None,
             gamma=None, grow_policy=None, importance_type=None,
             interaction_constraints=None, learning_rate=None, max_bin=None,
             max_cat_threshold=None, max_cat_to_onehot=None,
             max_delta_step=None, max_depth=None, max_leaves=None,
             min_child_weight=None, missing=nan, monotone_constraints=None,
             multi_strategy=None, n_estimators=None, n_jobs=None,
             num_parallel_tree=None, random_state=42, ...)
```

```
[ ]  # Предсказание на тестовом наборе для XGBoost
     xgb_y_pred3 = xgb_model3.predict(X_test3)

     # Метрики оценки модели XGBoost
     xgb_mse3 = mean_squared_error(y_test3, xgb_y_pred3)
     xgb_mae3 = mean_absolute_error(y_test3, xgb_y_pred3)
     xgb_rmse3 = np.sqrt(xgb_mse3)
     xgb_r23 = r2_score(y_test3, xgb_y_pred3)

     print("XGBoost - MSE:", xgb_mse3)
     print("XGBoost - MAE:", xgb_mae3)
     print("XGBoost - RMSE:", xgb_rmse3)
     print("XGBoost - R2 score:", xgb_r23)
```

```
XGBoost - MSE: 0.06738253874799972
XGBoost - MAE: 0.1936822821531006
XGBoost - RMSE: 0.2595814684217649
XGBoost - R2 score: 0.32850846046468696
```

Рисунок 15. Предсказание и метрики XGBoost

Random Forest	
MSE	0.06401756835351638
MAE	0.18438170136834003
RMSE	0.25301693293832406
R2	0.36204161597746365

Decision Tree	
MSE	0.07086133811507966
MAE	0.20573232361354177
RMSE	0.2661979303358305
R2	0.293840957783151

KNN	
MSE	0.06689625753775882
MAE	0.17950401136500005
RMSE	0.25864310842889054
R2	0.33335442982973174

XGBoost	
MSE	0.06738253874799972
MAE	0.1936822821531006
RMSE	0.2595814684217649
R2	0.32850846046468696

Рисунок 16. Результаты моделирования

В данном случае модель Random Forest демонстрирует лучшую производительность с точки зрения всех оценочных метрик, таких как среднеквадратичная ошибка (MSE), средняя абсолютная ошибка (MAE), корень из среднеквадратичной ошибки (RMSE) и коэффициент детерминации (R2), что указывает на более точные предсказания цен по сравнению с другими рассмотренными моделями.

Выводы

В результате исследования были изучены основы обучения искусственного интеллекта для предсказания цены продукта.

Для обучения были использован собственный набор данных, содержащие цены, характеристики и другие данные автомобилей, продающихся на сайте Drom. Были проанализированы результаты обучения на различных наборах данных и проведены эксперименты с различными конфигурациями моделей.

В результате экспериментов**,** можно сделать вывод о том, что модель Random Forest наиболее эффективна для предсказания цены.

Библиографический список

1. Найденова К. А., Невзорова О. А. Машинное обучение в задачах обработки естественного языка: обзор современного состояния исследований //Ученые записки Казанского университета. Серия Физико-математические науки. 2008. Т. 150. № 4. С. 5-24.
2. Аксютина Е. М., Белов Ю. С. Обзор архитектур и методов машинного обучения для анализа больших данных //Электронный журнал: наука, техника и образование. 2016. № 1. С. 134-141.

3. Донской В. И. Машинное обучение и обучаемость: сравнительный обзор // Intellectual Archive. 2012. №933. С. 19.
4. Гельцер Б.И., Циванюк М.М. Рублев В.Ю. Методы машинного обучения как инструмент диагностических прогностических исследований // Российский кардиологический журнал. 2020. № 12. С. 164-171.
5. Старостин В.С. Трансформация маркетинговых технологий в эпоху машинного интеллекта. //Вестник университета. 2018. № 1. С. 28-34.