Федеральное государственное автономное

образовательное учреждение высшего образования

«Пермский государственный национальный исследовательский университет» (ПГНИУ)

Региональный институт непрерывного образования (РИНО ПГНИУ)

Цифровая кафедра

Выпускная аттестационная (квалификационная) работа

по курсу профессиональной переподготовки

«Анализ данных и машинное обучение»

**Определение стоимости автомобиля посредством методов ИИ**

Разработчики проекта:

Конев Виктор Александрович

Черединова Ксения Михайловна

Ясырев Михаил Евгеньевич

Пермь, 2024

ОГЛАВЛЕНИЕ

[ПАСПОРТ ПРОЕКТА 3](#_Toc185613357)

[СОДЕРЖАНИЕ ПРОЕКТА 4](#_Toc185613358)

[Анализ проблемы исследования 4](#_Toc185613359)

[Реализация проекта 7](#_Toc185613360)

[Этап 1. Предобработка данных 7](#_Toc185613361)

[Этап 2. Исследовательский анализ данных (EDA) 8](#_Toc185613362)

[Этап 3. Построение и обучение моделей 14](#_Toc185613363)

[Этап 4. Визуализация результатов 16](#_Toc185613364)

[ЗАКЛЮЧЕНИЕ 20](#_Toc185613365)

[ПРИЛОЖЕНИЯ 22](#_Toc185613366)

**ПАСПОРТ ПРОЕКТА**

**Название проекта:**

Определение стоимости автомобиля посредством методов ИИ.

**Сведения об авторах:**

Конев Виктор Александрович, Черединова Ксения Михайловна, Ясырев Михаил Евгеньевич.

**Цель:**

Определить стоимость автомобиля с высокой точностью, используя методы искусственного интеллекта, чтобы помочь покупателям и продавцам справедливо оценивать транспортные средства.

**Задачи:**

1. Провести анализ данных об автомобилях, включающий их обработку, исследование корреляций, обучение различных моделей машинного обучения и выбор наилучшей из них.
2. Оптимизировать модели для достижения максимальной точности предсказания.

**Краткое описание проекта:**

Проект включает в себя три основных этапа: подготовка данных, исследовательский анализ и обучение моделей. На этапе предобработки данные кодируются, извлекаются полезные признаки из текстовых колонок, а числовые значения нормализуются. Исследовательский анализ позволяет понять распределение цены, выявить корреляции и ключевые признаки. Далее модели линейной регрессии, градиентного бустинга, случайного леса и метода ближайших соседей обучаются на подготовленных данных с подбором оптимальных гиперпараметров.

**Конкретные ожидаемые результаты:**

Основным результатом проекта станет модель, способная наиболее точно предсказывать стоимость автомобиля.

**СОДЕРЖАНИЕ ПРОЕКТА**

**Анализ проблемы исследования**

В современном автомобильном рынке оценка стоимости транспортных средств является одной из ключевых задач как для покупателей, так и для продавцов. Из-за большого разнообразия марок, моделей, комплектаций и состояний автомобилей определить справедливую цену становится сложной задачей. Часто субъективная оценка или недостаток данных приводят к неточным выводам, что создает риски для обеих сторон сделки.

Традиционные методы оценки, такие как использование прайс-листов или обращение к профессиональным оценщикам, часто не учитывают индивидуальные характеристики автомобиля или изменения рыночной ситуации. С другой стороны, использование методов искусственного интеллекта позволяет анализировать множество факторов одновременно и строить предсказания, основанные на больших объемах данных.

Основная проблема, решаемая в данном проекте, заключается в создании модели, способной с высокой точностью предсказывать стоимость автомобиля на основе данных о его характеристиках. Для этого необходимо использовать современные алгоритмы машинного обучения и подходы к обработке данных. Основное внимание уделяется выбору факторов, влияющих на стоимость, и обеспечению качества предсказаний с помощью оптимизации моделей.

Задача анализа стоимости автомобилей особенно актуальна в условиях быстро изменяющегося рынка и роста объемов данных. Использование машинного обучения в этой области открывает возможности для автоматизации процессов оценки и повышения их точности, что делает проект ценным с точки зрения практического применения.

**Цель:**

провести анализ данных об автомобилях и построить модель зависимости стоимости автомобиля от различных факторов, обеспечивающую точное прогнозирование цены.

**Задачи:**

1. Проанализировать проблему и обосновать ее актуальность.
2. Выполнить загрузку данных и их предобработку, включая обработку категориальных признаков, устранение пропусков и нормализацию числовых данных.
3. Провести исследовательский анализ данных: выявить ключевые закономерности, проверить распределение данных, провести корреляционный анализ и определить основные факторы, влияющие на стоимость.
4. Построить и обучить модели для предсказания стоимости, включая линейную регрессию, градиентный бустинг, случайный лес и метод ближайших соседей, с подбором оптимальных гиперпараметров.
5. Сравнить модели, оценить их качество, интерпретировать результаты и сделать вывод о достижении цели.

**Исходные данные**

В данной работе анализируется набор данных, содержащий информацию об автомобилях, их характеристиках и цене. Данные включают как технические параметры автомобилей (пробег, мощность, объем двигателя), так и их особенности оснащения (системы безопасности, функции комфорта и развлечений). Целью анализа является изучение факторов, влияющих на стоимость автомобилей, и построение модели, способной точно предсказать цену на основе предоставленных данных.

Список колонок анализируемого набора данных:

* make\_model – марка и модель автомобиля.
* body\_type – тип кузова.
* **price** – стоимость автомобиля (целевой показатель).
* vat – информация о налогообложении (например, подлежит ли НДС вычету).
* km – пробег автомобиля.
* Type – тип использования автомобиля (новый, подержанный и т.д.).
* Fuel – тип топлива (бензин, дизель, электричество и т.д.).
* Gears – количество передач в коробке передач.
* Comfort\_Convenience – оснащение, влияющее на комфорт и удобство.
* Entertainment\_Media – мультимедийные функции.
* Extras – дополнительные опции автомобиля.
* Safety\_Security – системы безопасности.
* age – возраст автомобиля в годах.
* Previous\_Owners – количество предыдущих владельцев.
* hp\_kW – мощность двигателя в киловаттах.
* Inspection\_new – наличие нового техосмотра.
* Paint\_Type – тип окраски автомобиля.
* Upholstery\_type – тип обивки салона.
* Gearing\_Type – тип коробки передач.
* Displacement\_cc – объем двигателя в кубических сантиметрах.
* Weight\_kg – вес автомобиля.
* Drive\_chain – тип привода (передний, задний, полный).
* cons\_comb – средний расход топлива на 100 км.

**Гипотеза исследования:**

Стоимость автомобиля зависит от его технических характеристик (пробега, возраста, мощности двигателя, объема двигателя), комплектации (наличия мультимедийных систем, функций комфорта и безопасности), типа кузова, а также типа топлива и коробки передач. Эти параметры могут быть использованы для построения модели линейной регрессии или других методов машинного обучения для предсказания цены автомобиля.

**Цель:**

Проверить, возможно ли с помощью предоставленных характеристик построить точную модель предсказания стоимости автомобиля.

**Подход к проверке гипотезы:**

На основе анализа корреляции определить, какие переменные имеют наибольшее влияние на стоимость автомобиля, и использовать их для построения модели. После предобработки данных и отбора значимых признаков обучить несколько моделей машинного обучения, включая линейную регрессию, и сравнить их точность.

**Реализация проекта**

### Этап 1. Предобработка данных

**Загрузка данных**

Для анализа был использован датасет, содержащий информацию об автомобилях, включая их характеристики, оснащение и стоимость.

path = kagglehub.dataset\_download("yaaryiitturan/auto-scout-car-price")

data = pd.read\_csv(path + '/final\_scout\_not\_dummy.csv')

**Обработка категориальных данных**

Для категориальных столбцов был применен метод One-Hot Encoding, что позволило преобразовать их в числовой формат.

categorical\_columns = ['make\_model', 'body\_type', 'Fuel', 'Gearing\_Type', 'Paint\_Type', 'Drive\_chain', 'vat', 'Type', 'Upholstery\_type']

data\_preprocessed = pd.get\_dummies(data\_preprocessed, columns=categorical\_columns, drop\_first=True)

**Извлечение количества опций из текстовых колонок**

Из текстовых колонок, содержащих список опций, был извлечен числовой признак — количество опций.

option\_columns = ['Comfort\_Convenience', 'Entertainment\_Media', 'Extras', 'Safety\_Security']

for col in option\_columns:

    data\_preprocessed[col + '\_count'] = data\_preprocessed[col].apply(lambda x: len(str(x).split(',')))

data\_preprocessed.drop(columns=option\_columns, inplace=True)

**Нормализация числовых данных**

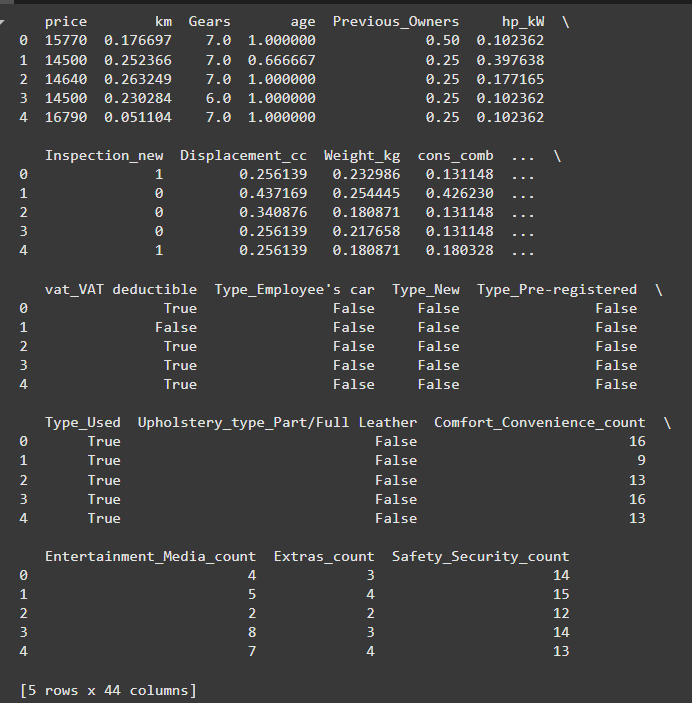
Все числовые признаки были приведены к единому масштабу с использованием метода Min-Max нормализации.

numeric\_columns = ['km', 'age', 'hp\_kW', 'Displacement\_cc', 'Weight\_kg', 'cons\_comb', 'Previous\_Owners']

scaler = MinMaxScaler()

data\_preprocessed[numeric\_columns] = scaler.fit\_transform(data\_preprocessed[numeric\_columns])

По итогу получаем такие данные

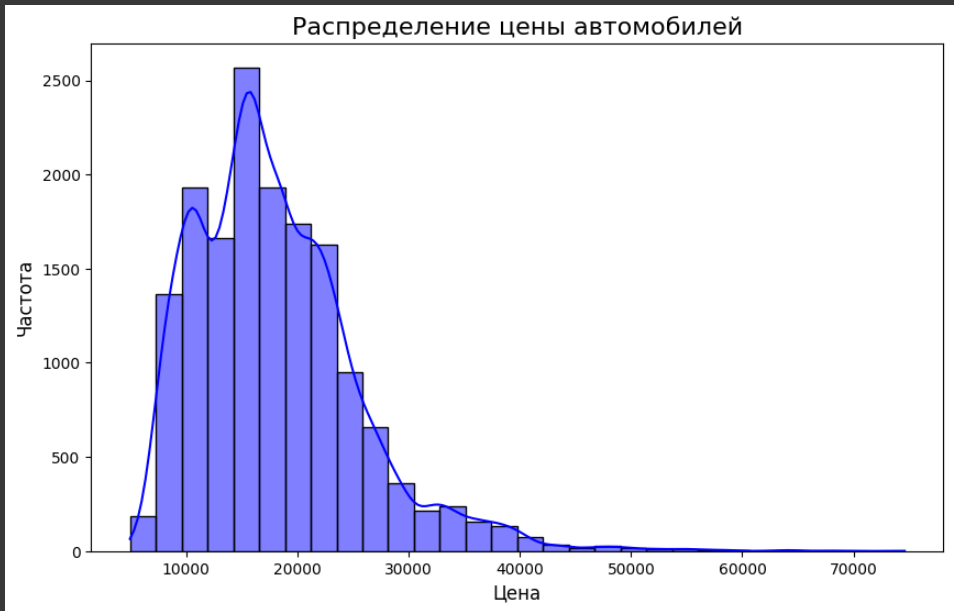
Рисунок 1 - Подготовленные данные

### Этап 2. Исследовательский анализ данных (EDA)

**Распределение целевой переменной**

Для анализа был построен график распределения цены автомобилей.

1. plt.figure(figsize=(10, 6))
2. sns.histplot(data\_preprocessed['price'], kde=True, bins=30, color='blue')
3. plt.title('Распределение цены автомобилей', fontsize=16)
4. plt.xlabel('Цена', fontsize=12)
5. plt.ylabel('Частота', fontsize=12)
6. plt.show()

Рисунок 2 - Распределение цены автомобилей

**Корреляционный анализ**

Для выявления взаимосвязей между признаками была построена корреляционная матрица.

# Корреляция всех признаков с целевым классом 'price'

corr\_with\_target = data\_preprocessed.corr()['price'].sort\_values(ascending=False)

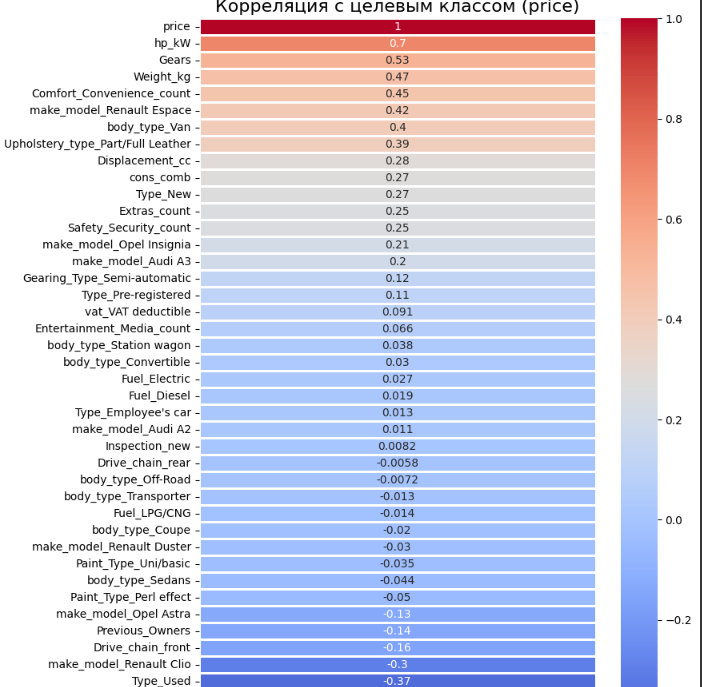
# Настройка фигуры и отображение тепловой карты с увеличенной высотой ячеек

plt.figure(figsize=(8, 12))  # Изменяем размер фигуры для увеличения высоты

sns.heatmap(corr\_with\_target.to\_frame(), cmap='coolwarm', annot=True, cbar=True, linewidths=1)

plt.title('Корреляция с целевым классом (price)', fontsize=16)

plt.show()

Рисунок 3 - Корреляция с целевым классом

**Оценка влияния признаков**

Построены графики зависимости стоимости от числовых признаков.

numeric\_features = ['km', 'age', 'hp\_kW', 'Displacement\_cc', 'Weight\_kg', 'cons\_comb', 'Previous\_Owners']

for feature in numeric\_features:

    plt.figure(figsize=(8, 5))

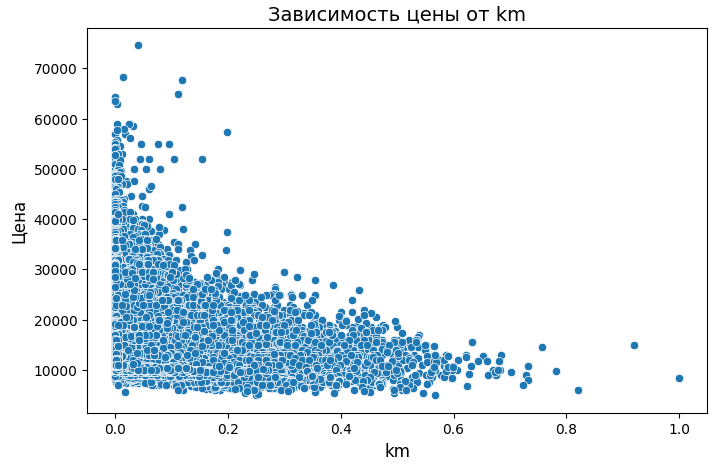
    sns.scatterplot(x=data\_preprocessed[feature], y=data\_preprocessed['price'])

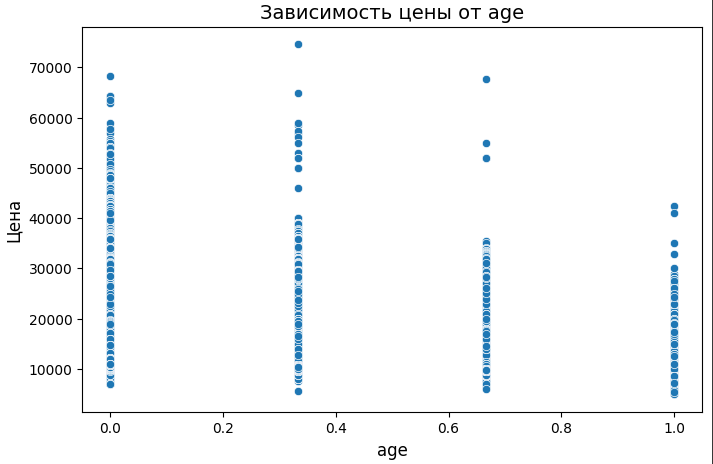
    plt.title(f'Зависимость цены от {feature}', fontsize=14)

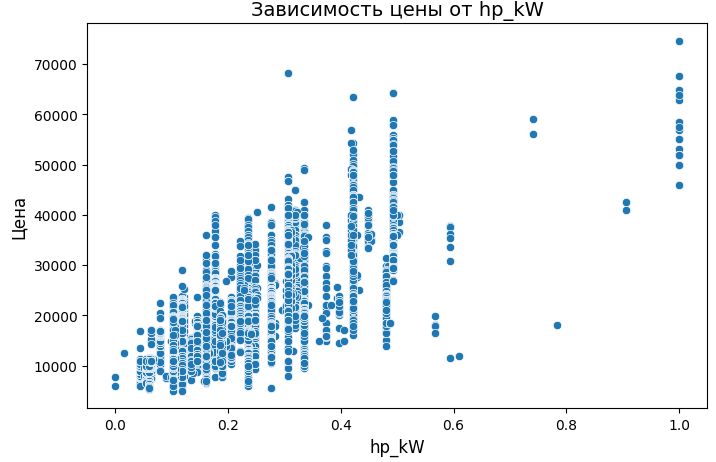
    plt.xlabel(feature, fontsize=12)

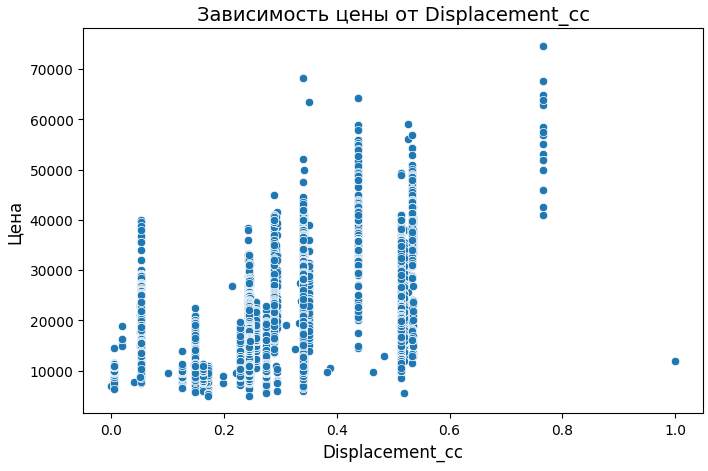
    plt.ylabel('Цена', fontsize=12)

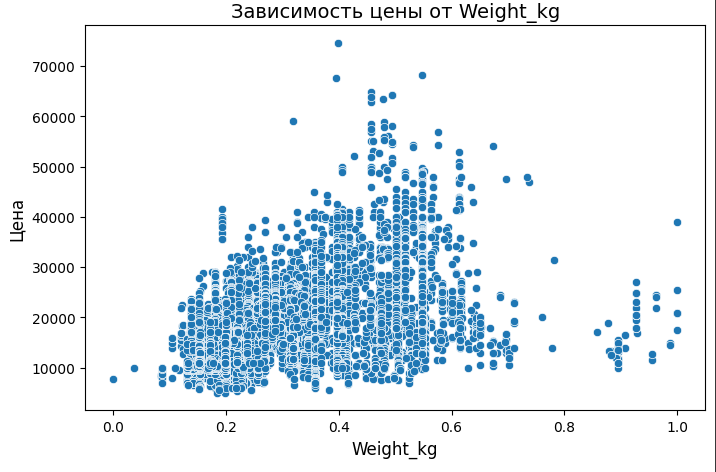
    plt.show()

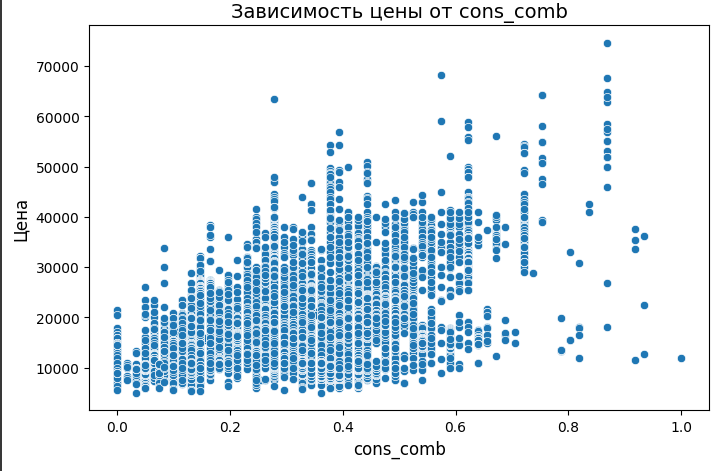
Рисунок 4 - Зависимость цены от km

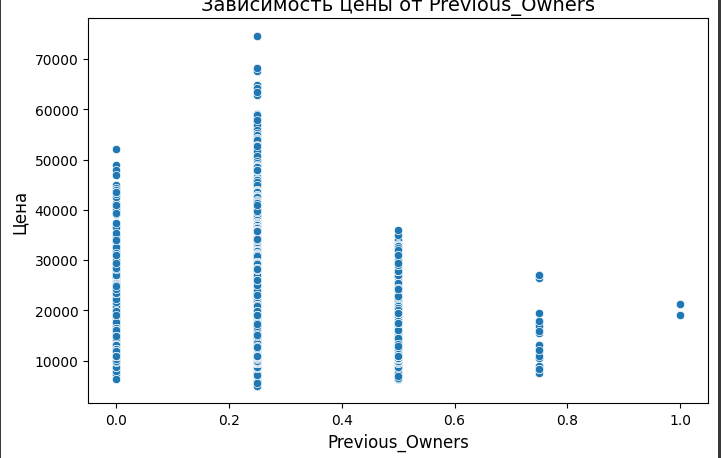
Рисунок 5 - Зависимость цены от age

Рисунок 6 Зависимость цены от hp\_kW

Рисунок 7 - Зависимость цены от Displacement\_cc

Рисунок 8 - Зависимость цены от Weight\_kg

Рисунок 9 - Зависимость цены от cons\_comb

Рисунок 10 - Зависимость цены от Previous\_Owners

### Этап 3. Построение и обучение моделей

**Подготовка данных для обучения**

Удалены признаки с низкой корреляцией, данные разделены на обучающую и тестовую выборки.

# Отбираем столбцы, у которых корреляция по модулю меньше 0.1

columns\_to\_drop = corr\_with\_target[abs(corr\_with\_target) < 0.1].index.tolist()

# Удаляем эти столбцы из DataFrame

data\_preprocessed = data\_preprocessed.drop(columns=columns\_to\_drop)

# Разделение данных на обучающую и тестовую выборки

X = data\_preprocessed.drop(columns=['price'])

y = data\_preprocessed['price']

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=42)

**Обучение моделей**

**Линейная регрессия**

lr\_model = LinearRegression()

lr\_model.fit(X\_train, y\_train)

y\_pred\_lr = lr\_model.predict(X\_test)

**Градиентный бустинг**

gb\_model = GradientBoostingRegressor(random\_state=42)

gb\_param\_grid = {

    'n\_estimators': [50, 100, 150],

    'learning\_rate': [0.01, 0.1, 0.2],

    'max\_depth': [3, 4, 5]

}

# Поиск гиперпараметров с GridSearchCV

gb\_grid\_search = GridSearchCV(gb\_model, gb\_param\_grid, cv=5, n\_jobs=-1, scoring='neg\_mean\_squared\_error')

gb\_grid\_search.fit(X\_train, y\_train)

best\_gb\_model = gb\_grid\_search.best\_estimator\_

y\_pred\_gb = best\_gb\_model.predict(X\_test)

**Случайный лес**

rf\_model = RandomForestRegressor(random\_state=42)

rf\_param\_grid = {

    'n\_estimators': [50, 100, 200],

    'max\_depth': [None, 10, 20, 30],

    'min\_samples\_split': [2, 5, 10],

    'min\_samples\_leaf': [1, 2, 4]

}

# Поиск гиперпараметров с GridSearchCV

rf\_grid\_search = GridSearchCV(rf\_model, rf\_param\_grid, cv=5, n\_jobs=-1, scoring='neg\_mean\_squared\_error')

rf\_grid\_search.fit(X\_train, y\_train)

best\_rf\_model = rf\_grid\_search.best\_estimator\_

y\_pred\_rf = best\_rf\_model.predict(X\_test)

**K-ближайшие соседи**

knn\_model = KNeighborsRegressor()

knn\_param\_grid = {

    'n\_neighbors': [3, 5, 7, 10],

    'weights': ['uniform', 'distance'],

    'algorithm': ['auto', 'ball\_tree', 'kd\_tree', 'brute']

}

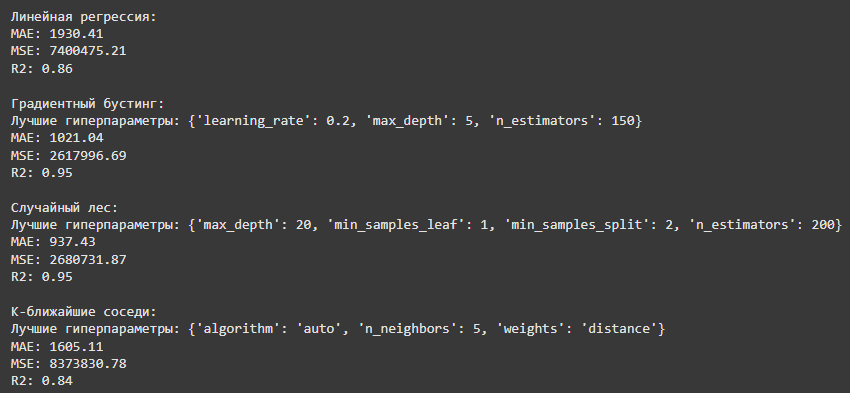
# Поиск гиперпараметров с GridSearchCV

knn\_grid\_search = GridSearchCV(knn\_model, knn\_param\_grid, cv=5, n\_jobs=-1, scoring='neg\_mean\_squared\_error')

knn\_grid\_search.fit(X\_train, y\_train)

best\_knn\_model = knn\_grid\_search.best\_estimator\_

y\_pred\_knn = best\_knn\_model.predict(X\_test)

****Рисунок 11 - Метрики для каждой из моделей

### Этап 4. Визуализация результатов

Для каждой модели построены графики зависимости реальных и предсказанных значений.

# 1. Линейная регрессия

plt.figure(figsize=(8, 6))

plt.scatter(y\_test, y\_pred\_lr, alpha=0.7, color='blue')

plt.plot([y\_test.min(), y\_test.max()], [y\_test.min(), y\_test.max()], 'k--', lw=2)

plt.title('Линейная регрессия: Реальные vs Предсказанные значения', fontsize=14)

plt.xlabel('Реальные значения', fontsize=12)

plt.ylabel('Предсказанные значения', fontsize=12)

plt.show()

# 2. Градиентный бустинг

plt.figure(figsize=(8, 6))

plt.scatter(y\_test, y\_pred\_gb, alpha=0.7, color='green')

plt.plot([y\_test.min(), y\_test.max()], [y\_test.min(), y\_test.max()], 'k--', lw=2)

plt.title('Градиентный бустинг: Реальные vs Предсказанные значения', fontsize=14)

plt.xlabel('Реальные значения', fontsize=12)

plt.ylabel('Предсказанные значения', fontsize=12)

plt.show()

# 3. Случайный лес

plt.figure(figsize=(8, 6))

plt.scatter(y\_test, y\_pred\_rf, alpha=0.7, color='orange')

plt.plot([y\_test.min(), y\_test.max()], [y\_test.min(), y\_test.max()], 'k--', lw=2)

plt.title('Случайный лес: Реальные vs Предсказанные значения', fontsize=14)

plt.xlabel('Реальные значения', fontsize=12)

plt.ylabel('Предсказанные значения', fontsize=12)

plt.show()

# 4. K-ближайшие соседи

plt.figure(figsize=(8, 6))

plt.scatter(y\_test, y\_pred\_knn, alpha=0.7, color='red')

plt.plot([y\_test.min(), y\_test.max()], [y\_test.min(), y\_test.max()], 'k--', lw=2)

plt.title('K-ближайшие соседи: Реальные vs Предсказанные значения', fontsize=14)

plt.xlabel('Реальные значения', fontsize=12)

plt.ylabel('Предсказанные значения', fontsize=12)

plt.show()

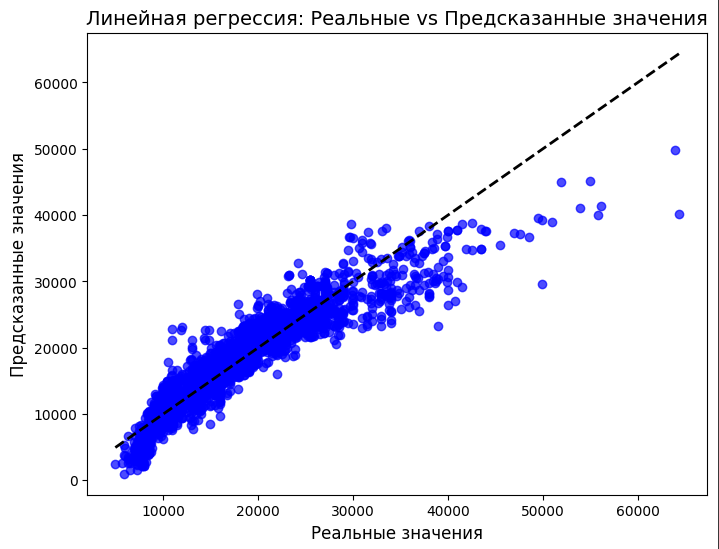
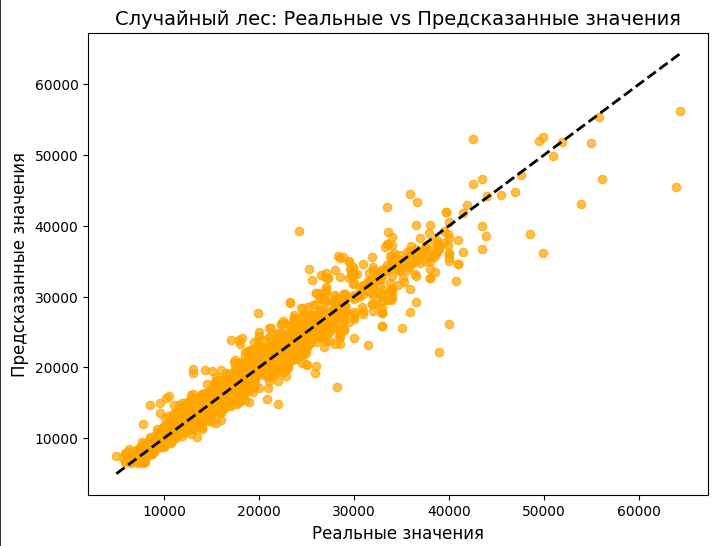
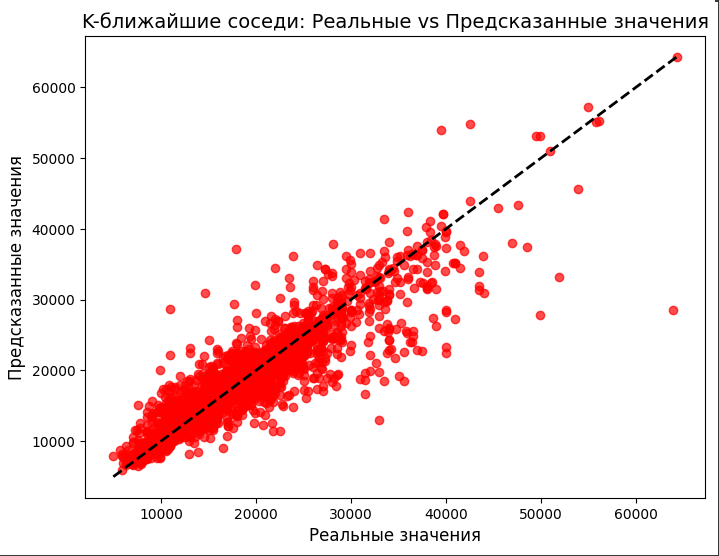
****

Рисунок 12 - График реальных и предсказанных значений для линейной регрессии

  
Рисунок 13 - График реальных и предсказанных значений для градиентного бустинга.

  
Рисунок 14 - График реальных и предсказанных значений для случайного леса.

  
Рисунок 15 - График реальных и предсказанных значений для K-ближайших соседей.

**ЗАКЛЮЧЕНИЕ**

В ходе работы была проведена тщательная обработка и анализ данных, что позволило построить модель, которая предсказывает стоимость автомобилей на основе различных факторов, таких как тип кузова, тип топлива, мощность, возраст и другие параметры. Выдвинутая гипотеза о том, что стоимость автомобилей может зависеть от этих факторов, была подтверждена с помощью построенных моделей.

Наибольшее влияние на цену автомобилей оказывают такие факторы, как мощность двигателя (hp\_kW), возраст машины (age) и пробег (km). Эти переменные обладают высокой корреляцией с ценой, что было подтверждено при анализе данных. Также важным фактором является тип топлива (Fuel), поскольку различные виды топлива влияют на эксплуатационные характеристики и, как следствие, на цену автомобиля.

При сравнении различных моделей линейной регрессии, градиентного бустинга, случайного леса и K-ближайших соседей было выявлено, что модели градиентного бустинга и случайного леса показали наилучшие результаты с коэффициентом детерминации (R²) 0,95. Эти модели обеспечили точные прогнозы, при этом случайный лес, несмотря на схожие результаты с градиентным бустингом, имел немного более высокие ошибки. Модель линейной регрессии также продемонстрировала хорошие результаты с R² 0,86, но её прогнозы были менее точными, чем у более сложных методов.

Модель градиентного бустинга показала наибольшую эффективность для данной задачи, что объясняется её способностью захватывать сложные зависимости между переменными. В то же время линейная регрессия, несмотря на свою простоту и скорость обучения, не смогла справиться с такой сложной задачей с такой же точностью.

Таким образом, поставленная цель была достигнута — выполнен анализ данных и построена модель для прогнозирования стоимости автомобилей с высокой точностью. Полученные результаты позволяют утверждать, что наиболее эффективным методом для прогнозирования является градиентный бустинг, а также случайный лес. Эти модели могут быть применены в реальных задачах, таких как оценка стоимости автомобилей для различных целей, включая продажу, страхование и кредитование.

В ходе работы были решены все поставленные задачи, включая обработку данных, построение и оценку моделей, а также интерпретацию результатов. Построенная модель может быть использована для дальнейших исследований и улучшений, таких как добавление новых переменных или применение других методов машинного обучения для повышения точности прогнозов.

**ПРИЛОЖЕНИЯ**

Приложение 1

Программный код

import os

import pandas as pd

import numpy as np

import seaborn as sns

import matplotlib.pyplot as plt

from sklearn.model\_selection import train\_test\_split

from sklearn.linear\_model import LinearRegression

from sklearn.ensemble import GradientBoostingRegressor

from sklearn.metrics import mean\_absolute\_error, mean\_squared\_error, r2\_score

from sklearn.ensemble import RandomForestRegressor

from sklearn.neighbors import KNeighborsRegressor

from sklearn.preprocessing import OneHotEncoder, MinMaxScaler

from sklearn.model\_selection import GridSearchCV

import kagglehub

path = kagglehub.dataset\_download("yaaryiitturan/auto-scout-car-price")

os.listdir(path)

data = pd.read\_csv(path + '/final\_scout\_not\_dummy.csv')

data.head()

data\_preprocessed = data.copy()

categorical\_columns = ['make\_model', 'body\_type', 'Fuel', 'Gearing\_Type', 'Paint\_Type', 'Drive\_chain', 'vat', 'Type', 'Upholstery\_type']

data\_preprocessed = pd.get\_dummies(data\_preprocessed, columns=categorical\_columns, drop\_first=True)

option\_columns = ['Comfort\_Convenience', 'Entertainment\_Media', 'Extras', 'Safety\_Security']

for col in option\_columns:

    data\_preprocessed[col + '\_count'] = data\_preprocessed[col].apply(lambda x: len(str(x).split(',')))

data\_preprocessed.drop(columns=option\_columns, inplace=True)

numeric\_columns = ['km', 'age', 'hp\_kW', 'Displacement\_cc', 'Weight\_kg', 'cons\_comb', 'Previous\_Owners']

scaler = MinMaxScaler()

data\_preprocessed[numeric\_columns] = scaler.fit\_transform(data\_preprocessed[numeric\_columns])

plt.figure(figsize=(10, 6))

sns.histplot(data\_preprocessed['price'], kde=True, bins=30, color='blue')

plt.title('Распределение цены автомобилей', fontsize=16)

plt.xlabel('Цена', fontsize=12)

plt.ylabel('Частота', fontsize=12)

plt.show()

corr\_with\_target = data\_preprocessed.corr()['price'].sort\_values(ascending=False)

plt.figure(figsize=(8, 12))

sns.heatmap(corr\_with\_target.to\_frame(), cmap='coolwarm', annot=True, cbar=True, linewidths=1)

plt.title('Корреляция с целевым классом (price)', fontsize=16)

plt.show()

# Отбираем столбцы, у которых корреляция по модулю меньше 0.1

columns\_to\_drop = corr\_with\_target[abs(corr\_with\_target) < 0.1].index.tolist()

# Удаляем эти столбцы из DataFrame

data\_preprocessed = data\_preprocessed.drop(columns=columns\_to\_drop)

numeric\_features = ['km', 'age', 'hp\_kW', 'Displacement\_cc', 'Weight\_kg', 'cons\_comb', 'Previous\_Owners']

# Определяем количество строк и столбцов

n\_cols = 3

n\_rows = (len(numeric\_features) + n\_cols - 1) // n\_cols  # Вычисляем необходимое количество строк

# Создаем фигуру и оси

fig, axes = plt.subplots(n\_rows, n\_cols, figsize=(14, 3 \* n\_rows))

axes = axes.flatten()  # Преобразуем массив осей в плоский список для удобства

# Рисуем графики

for i, feature in enumerate(numeric\_features):

    sns.scatterplot(ax=axes[i], x=data\_preprocessed[feature], y=data\_preprocessed['price'])

    axes[i].set\_title(f'Зависимость цены от {feature}', fontsize=14)

    axes[i].set\_xlabel(feature, fontsize=12)

    axes[i].set\_ylabel('Цена', fontsize=12)

# Удаляем пустые графики, если есть

for j in range(len(numeric\_features), len(axes)):

    fig.delaxes(axes[j])

plt.tight\_layout()

plt.show()

# Вспомогательная функция для подсчета и вывода метрик полученный моделей

def printModelScores(model, modelName, y\_pred, best\_params=None):

  print(f"{modelName}:")

  if best\_params is not None: print(f"Лучшие гиперпараметры: {best\_params}")

  print(f"MAE: {mean\_absolute\_error(y\_test, y\_pred):.2f}")

  print(f"MSE: {mean\_squared\_error(y\_test, y\_pred):.2f}")

  print(f"R2: {r2\_score(y\_test, y\_pred):.2f}\n")

X = data\_preprocessed.drop(columns=['price'])

y = data\_preprocessed['price']

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=42)

lr\_model = LinearRegression()

lr\_model.fit(X\_train, y\_train)

y\_pred\_lr = lr\_model.predict(X\_test)

# Метрики для линейной регрессии

printModelScores(lr\_model, 'Линейная регрессия', y\_pred\_lr)

gb\_model = GradientBoostingRegressor(random\_state=42)

gb\_param\_grid = {

    'n\_estimators': [50, 100, 150],

    'learning\_rate': [0.01, 0.1, 0.2],

    'max\_depth': [3, 4, 5]

}

# Поиск гиперпараметров с GridSearchCV

gb\_grid\_search = GridSearchCV(gb\_model, gb\_param\_grid, cv=5, n\_jobs=-1, scoring='neg\_mean\_squared\_error')

gb\_grid\_search.fit(X\_train, y\_train)

best\_gb\_model = gb\_grid\_search.best\_estimator\_

y\_pred\_gb = best\_gb\_model.predict(X\_test)

# Метрики для градиентного бустинга

printModelScores(gb\_model, 'Градиентный бустинг', y\_pred\_gb, gb\_grid\_search.best\_params\_)

rf\_model = RandomForestRegressor(random\_state=42)

rf\_param\_grid = {

    'n\_estimators': [50, 100, 200],

    'max\_depth': [None, 10, 20, 30],

    'min\_samples\_split': [2, 5, 10],

    'min\_samples\_leaf': [1, 2, 4]

}

# Поиск гиперпараметров с GridSearchCV

rf\_grid\_search = GridSearchCV(rf\_model, rf\_param\_grid, cv=5, n\_jobs=-1, scoring='neg\_mean\_squared\_error')

rf\_grid\_search.fit(X\_train, y\_train)

best\_rf\_model = rf\_grid\_search.best\_estimator\_

y\_pred\_rf = best\_rf\_model.predict(X\_test)

# Метрики для случайного леса

printModelScores(rf\_model, 'Случайный лес бустинг', y\_pred\_rf, rf\_grid\_search.best\_params\_)

knn\_model = KNeighborsRegressor()

knn\_param\_grid = {

    'n\_neighbors': [3, 5, 7, 10],

    'weights': ['uniform', 'distance'],

    'algorithm': ['auto', 'ball\_tree', 'kd\_tree', 'brute']

}

# Поиск гиперпараметров с GridSearchCV

knn\_grid\_search = GridSearchCV(knn\_model, knn\_param\_grid, cv=5, n\_jobs=-1, scoring='neg\_mean\_squared\_error')

knn\_grid\_search.fit(X\_train, y\_train)

best\_knn\_model = knn\_grid\_search.best\_estimator\_

y\_pred\_knn = best\_knn\_model.predict(X\_test)

# Метрики для K-ближайших соседей

printModelScores(rf\_model, 'K-ближайшие соседи', y\_pred\_knn, knn\_grid\_search.best\_params\_)

plt.figure(figsize=(8, 6))

plt.scatter(y\_test, y\_pred\_lr, alpha=0.7, color='blue')

plt.plot([y\_test.min(), y\_test.max()], [y\_test.min(), y\_test.max()], 'k--', lw=2)

plt.title('Линейная регрессия: Реальные vs Предсказанные значения', fontsize=14)

plt.xlabel('Реальные значения', fontsize=12)

plt.ylabel('Предсказанные значения', fontsize=12)

plt.show()

plt.figure(figsize=(8, 6))

plt.scatter(y\_test, y\_pred\_gb, alpha=0.7, color='green')

plt.plot([y\_test.min(), y\_test.max()], [y\_test.min(), y\_test.max()], 'k--', lw=2)

plt.title('Градиентный бустинг: Реальные vs Предсказанные значения', fontsize=14)

plt.xlabel('Реальные значения', fontsize=12)

plt.ylabel('Предсказанные значения', fontsize=12)

plt.show()

plt.figure(figsize=(8, 6))

plt.scatter(y\_test, y\_pred\_rf, alpha=0.7, color='orange')

plt.plot([y\_test.min(), y\_test.max()], [y\_test.min(), y\_test.max()], 'k--', lw=2)

plt.title('Случайный лес: Реальные vs Предсказанные значения', fontsize=14)

plt.xlabel('Реальные значения', fontsize=12)

plt.ylabel('Предсказанные значения', fontsize=12)

plt.show()

plt.figure(figsize=(8, 6))

plt.scatter(y\_test, y\_pred\_knn, alpha=0.7, color='red')

plt.plot([y\_test.min(), y\_test.max()], [y\_test.min(), y\_test.max()], 'k--', lw=2)

plt.title('K-ближайшие соседи: Реальные vs Предсказанные значения', fontsize=14)

plt.xlabel('Реальные значения', fontsize=12)

plt.ylabel('Предсказанные значения', fontsize=12)

plt.show()