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

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 использованием искусственного интеллекта для помощи в оценке транспортных средств как продавцам, так и покупателям.

Задачи:

1. Провести анализ данных, включая исследование корреляции, провести предобработку.
2. Обучить различные модели машинного обучения и выбрать из них лучшую.
3. Подобрать оптимальные гиперпараметры для достижения лучших предсказаний моделями.

Краткое описание проекта:

В проекте можно выделить три основных этапа – подготовка данных, исследовательский анализ и обучение моделей. Этап подготовки данных включает в себя предобработку, в коде которой числовые признаки нормализуются, категориальные данные кодируются, а признаки с перечислением опций заменяются на количество этих опций. Из исследовательского анализа можно понять распределение цены, проверять корреляцию между признаками и целевой переменной, выявить основные значимые признаки и отбросить ненужные. Далее обучаем модели линейной регрессии, градиентного бустинга, случайного леса и метода ближайших соседей на подготовленных данных с подбором оптимальных гиперпараметров.

Конкретные ожидаемые результаты:

Основной результат проекта - модель, способная наиболее точно предсказать стоимость автомобиля.

СОДЕРЖАНИЕ ПРОЕКТА

Анализ проблемы исследования

На современном рынке автомобилей оценка стоимости транспортного средства может быть довольно большой проблемой, как для продавцов, так и для самих покупателей. Причиной этому является большое количество марок, моделей, комплектаций и состояний транспортных средств, в связи с чем определение справедливой стоимости становится сложной задачей. Зачастую субъективная оценка или недостаток данных приводят к неточным выводам и это создает риски для покупателя и продавца.

Классические методы оценки, например как просмотр прайс-листов или обращение к оценщикам, часто могут не учитывать индивидуальные особенности автомобиля или изменения в рыночной ситуации. С другой же стороны, используя методы искусственного интеллекта можно анализировать множество факторов одновременно и строить предсказания на основе больших объемов данных.

Основной проблемой, решаемой этим проектом, является создание модели, которая будет способна с высокой точностью предсказывать стоимость транспортного средства на основе данных о его характеристиках. Для достижения такого результата необходимо использовать современные алгоритмы машинного обучения и подходы для обработки данных. Особенное внимание уделяется выбору признаков, влияющих на стоимость и обеспечение качества предсказаний с помощью оптимизации моделей.

Анализ стоимости автомобилей особенно актуален в условиях изменчивого рынка и роста объема данных. Используя машинное обучение, открываются большие возможности для автоматизации различных процессов оценки с повышением точности, что делает данный проект ценным для практического применения.

Цель:

Проанализировать данные автомобилей и смоделировать зависимости стоимости от различных факторов.

Задачи:

1. Анализ проблемы и обоснование её актуальности.
2. Загрузка данных и предобработка (обработка категориальных данных, устранение пропусков, нормализация числовых данных)

3. Исследовательский анализ данных с выявлением ключевых закономерностей, проверка распределения. проведение корреляционного анализа и определение наиболее влиятельных признаков.
4. Построение и обучение моделей (линейная регрессия, градиентный бустинг, случайный лес и метод ближайших соседей) с подбором гиперпараметров.
5. Сравнение моделей, оценка их качества и объяснение результатов с выводами о достижении цели.

Исходные данные

В работе используется набор данных с информацией о транспортных средствах, их характеристиках и стоимости. Данные включает как технических параметры, например пробег, мощности и объем двигателя, так и особенности автомобильного оснащения, такие как системы безопасности, функции комфорта и развлечений. Целью анализа является изучение факторов, которые влияют на цену транспортного средства и построение модели, которая способна точно предсказать стоимость на основе предоставленных данных.

Список колонок набора данных:

- make_model – марка и модель автомобиля.
- body_type – тип кузова.
- **price** – стоимость автомобиля (целевой признак).
- vat – информация о налогообложении.
- 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 транспортного средства зависит от его технических характеристик, комплектации, типа кузова, а также типа топлива и коробки передач. С использованием этих параметров можно построить модели машинного обучения для точного предсказания цены автомобиля.

Цель:

Проверить возможность использования предоставленных характеристик для построения точной модели предсказания стоимость транспортного средства.

Подход к проверке гипотезы:

На основе корреляционного анализа определить переменные, которые имеют наибольшее влияние на стоимость автомобиля и использовать их для построения модели. Обучить несколько моделей машинного обучения 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

```
    price        km  Gears       age  Previous_Owners     hp_kW  \
0   15770  0.176697    7.0  1.000000             0.50  0.102362
1   14500  0.252366    7.0  0.666667             0.25  0.397638
2   14640  0.263249    7.0  1.000000             0.25  0.177165
3   14500  0.230284    6.0  1.000000             0.25  0.102362
4   16790  0.051104    7.0  1.000000             0.25  0.102362

    Inspection_new  Displacement_cc  Weight_kg  cons_comb  ...  \
0                1         0.256139   0.232986   0.131148  ...
1                0         0.437169   0.254445   0.426230  ...
2                0         0.340876   0.180871   0.131148  ...
3                0         0.256139   0.217658   0.131148  ...
4                1         0.256139   0.180871   0.180328  ...

    vat_VAT deductible  Type_Employee's car  Type_New  Type_Pre-registered  \
0                 True                False     False                False
1                False                False     False                False
2                 True                False     False                False
3                 True                False     False                False
4                 True                False     False                False

    Type_Used  Upholstery_type_Part/Full Leather  Comfort_Convenience_count  \
0        True                              False                         16
1        True                              False                          9
2        True                              False                         13
3        True                              False                         16
4        True                              False                         13

    Entertainment_Media_count  Extras_count  Safety_Security_count
0                           4             3                     14
1                           5             4                     15
2                           2             2                     12
3                           8             3                     14
4                           7             4                     13

[5 rows x 44 columns]
```

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()
```

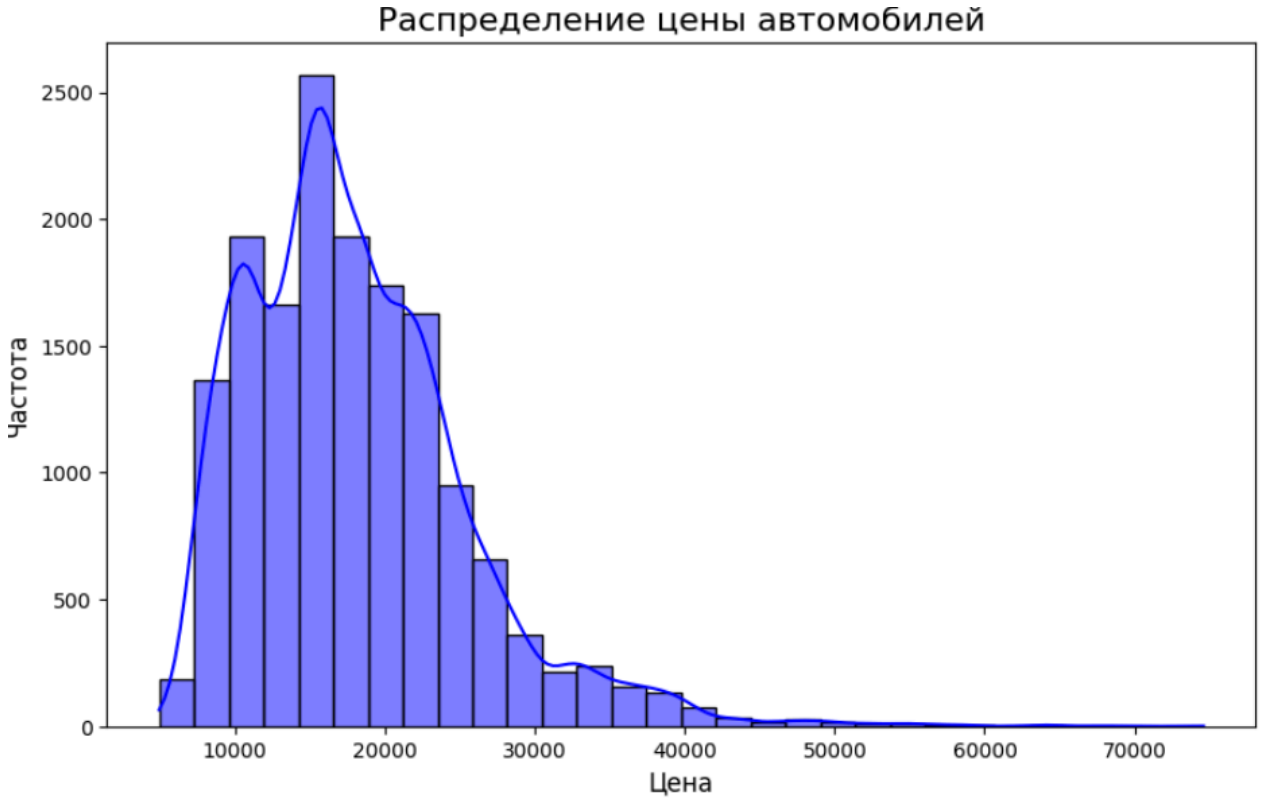

Рисунок 2 - Распределение цены автомобилей

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()
```

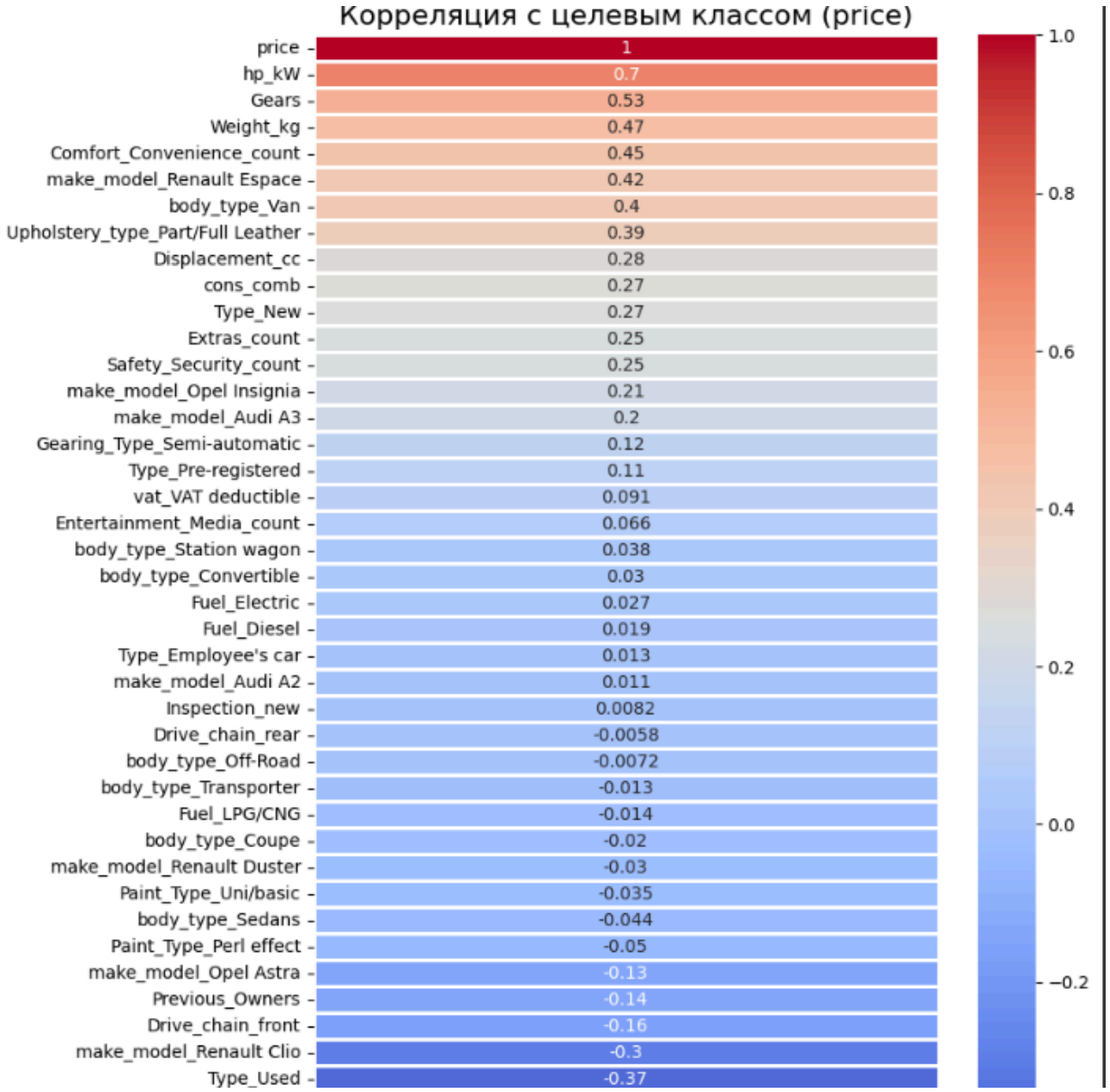

Рисунок 3 - Корреляция с целевым классом

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()
```

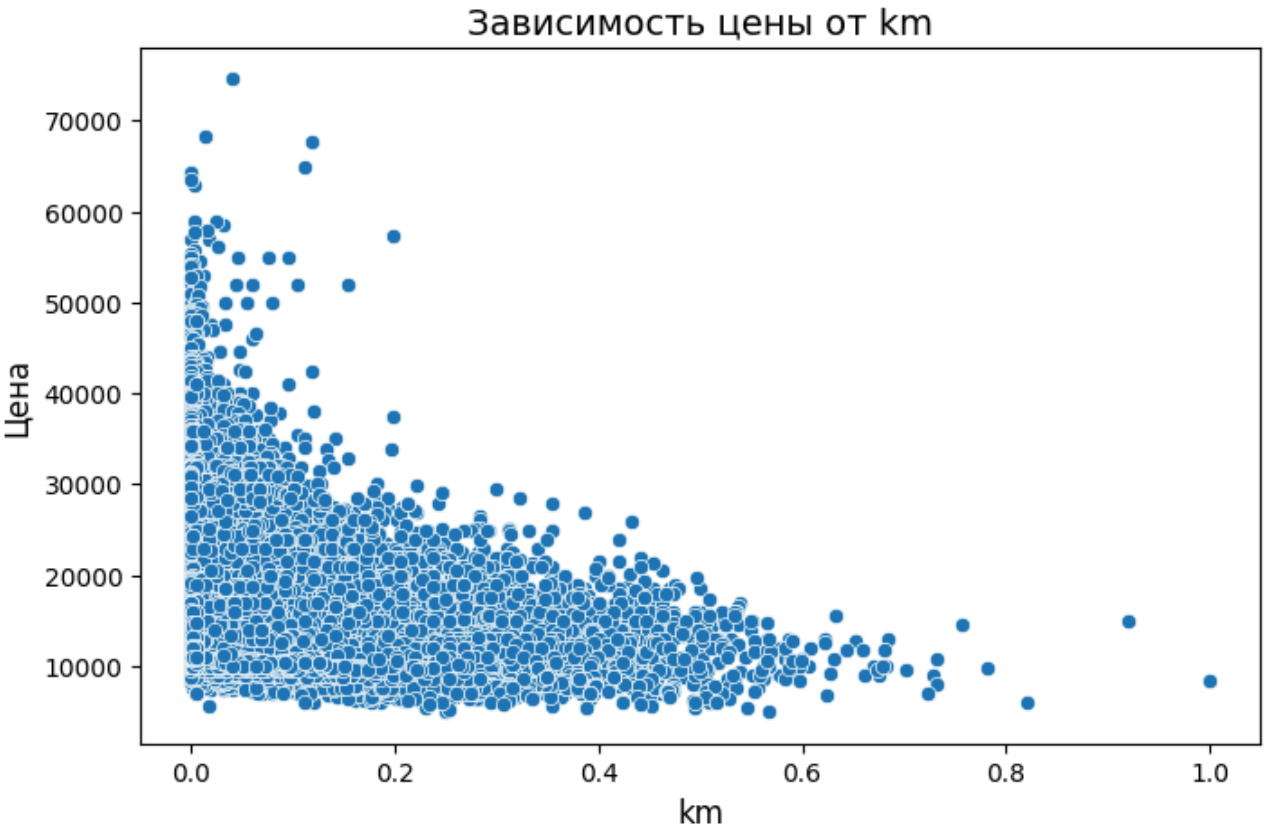

Рисунок 4 - Зависимость цены от km

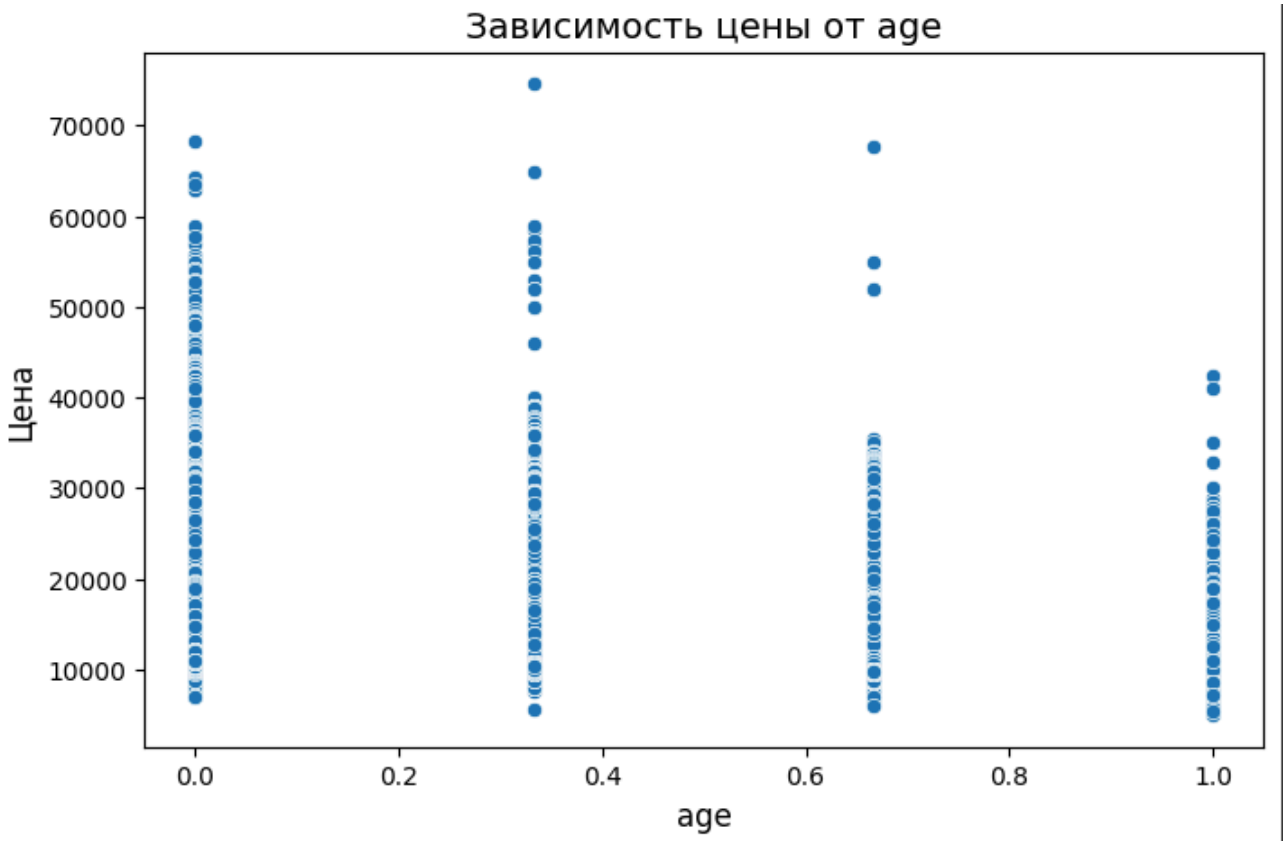

Рисунок 5 - Зависимость цены от age

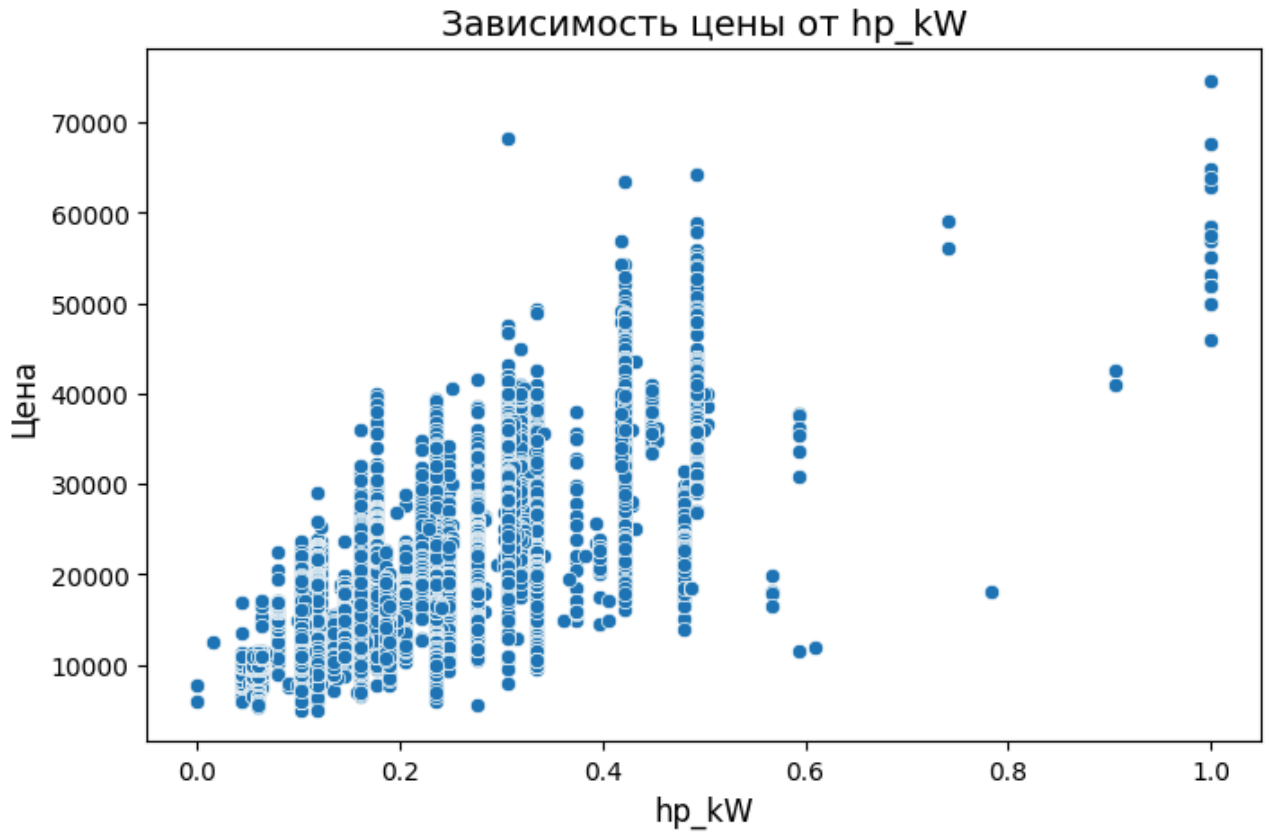

Рисунок 6 Зависимость цены от hp_kW

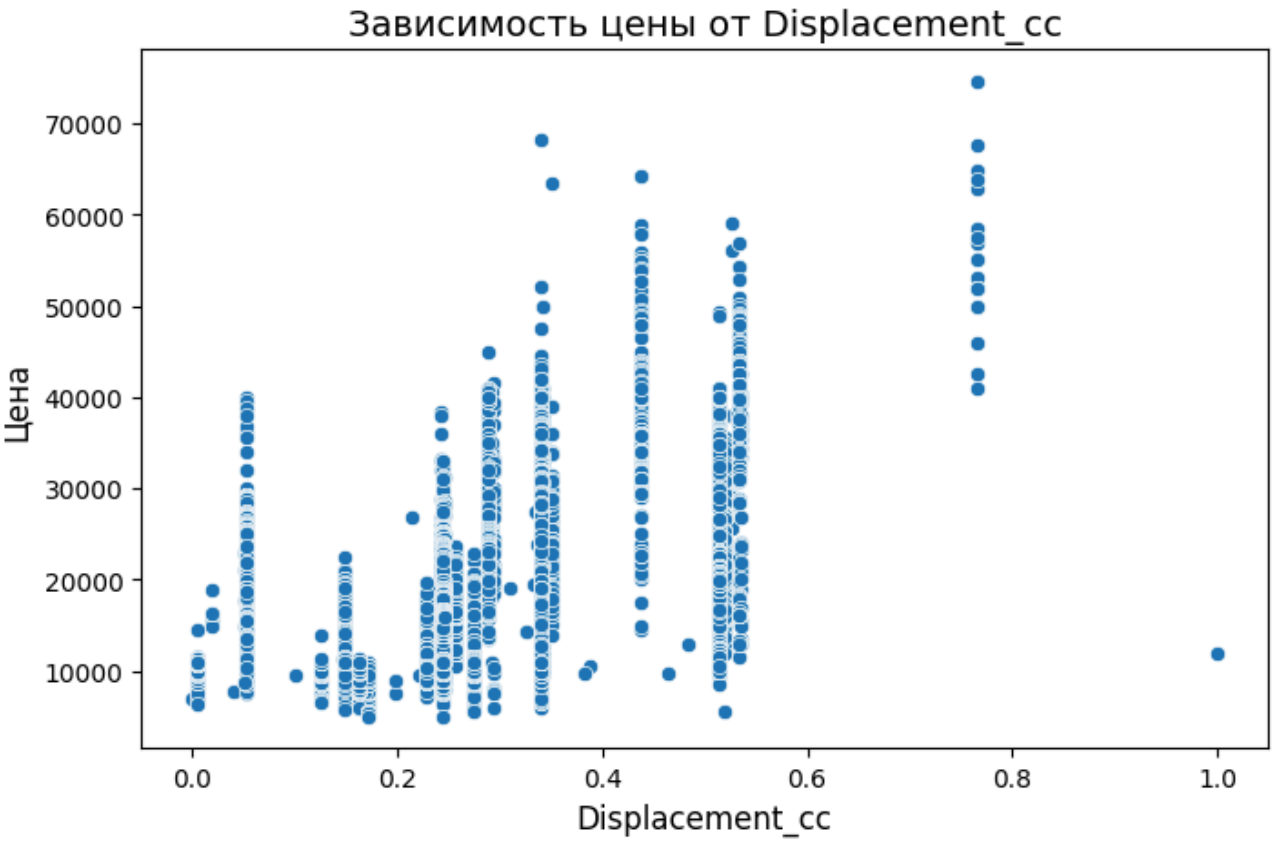

Рисунок 7 - Зависимость цены от Displacement_cc

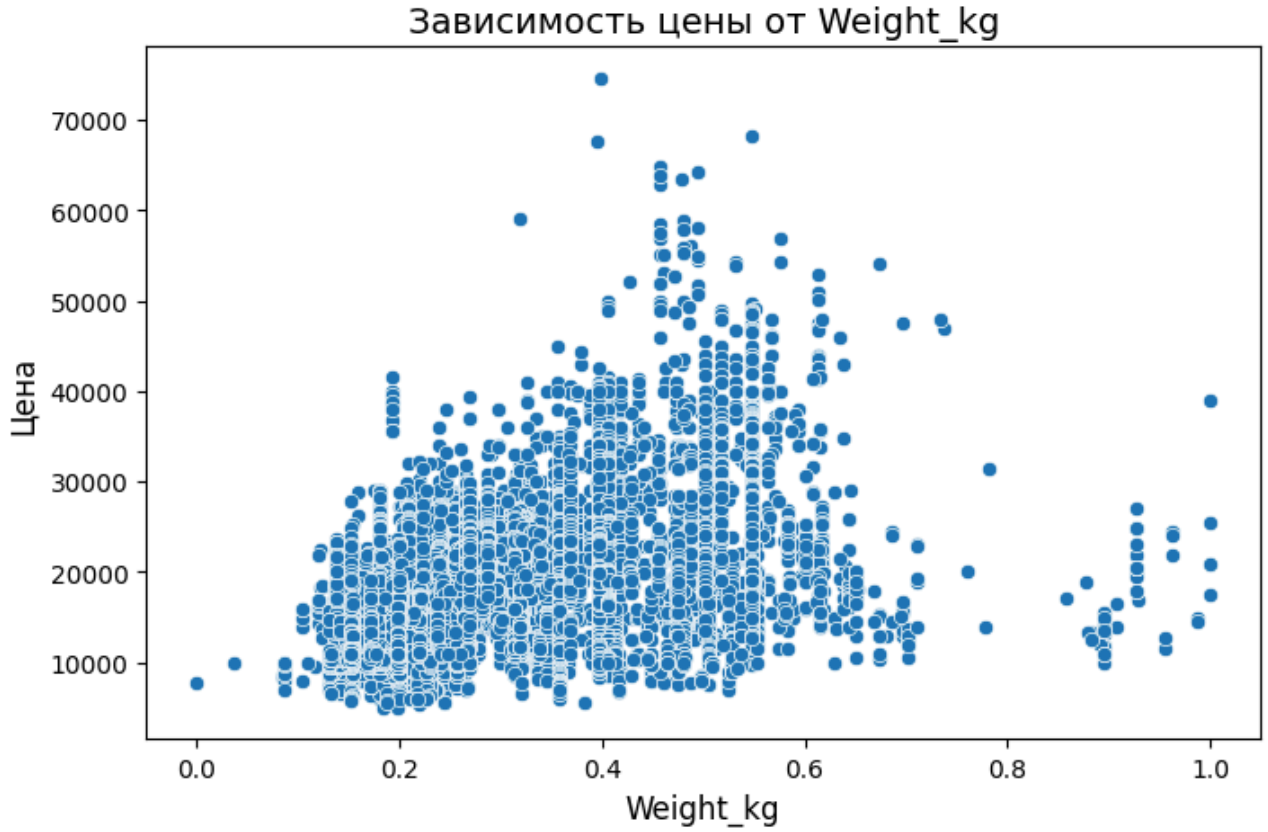

Рисунок 8 - Зависимость цены от Weight_kg

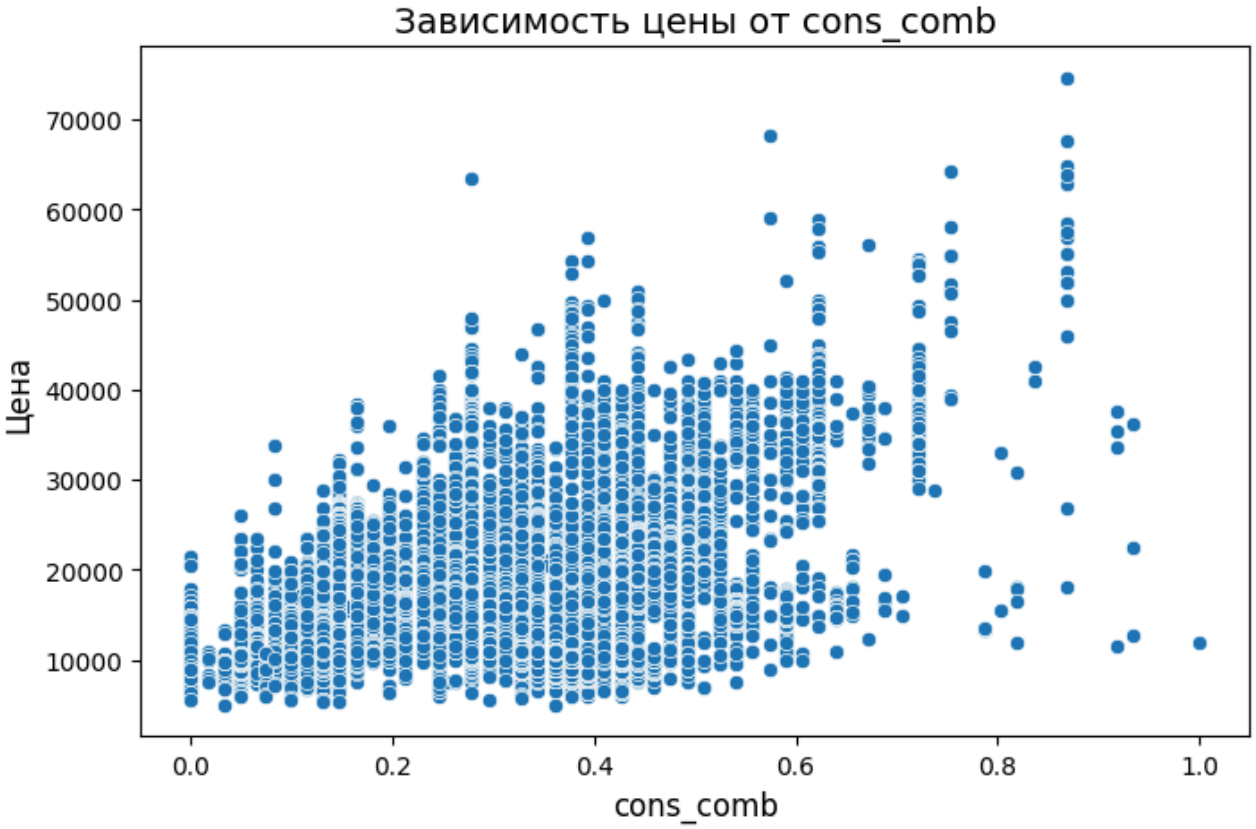

Рисунок 9 - Зависимость цены от cons_comb

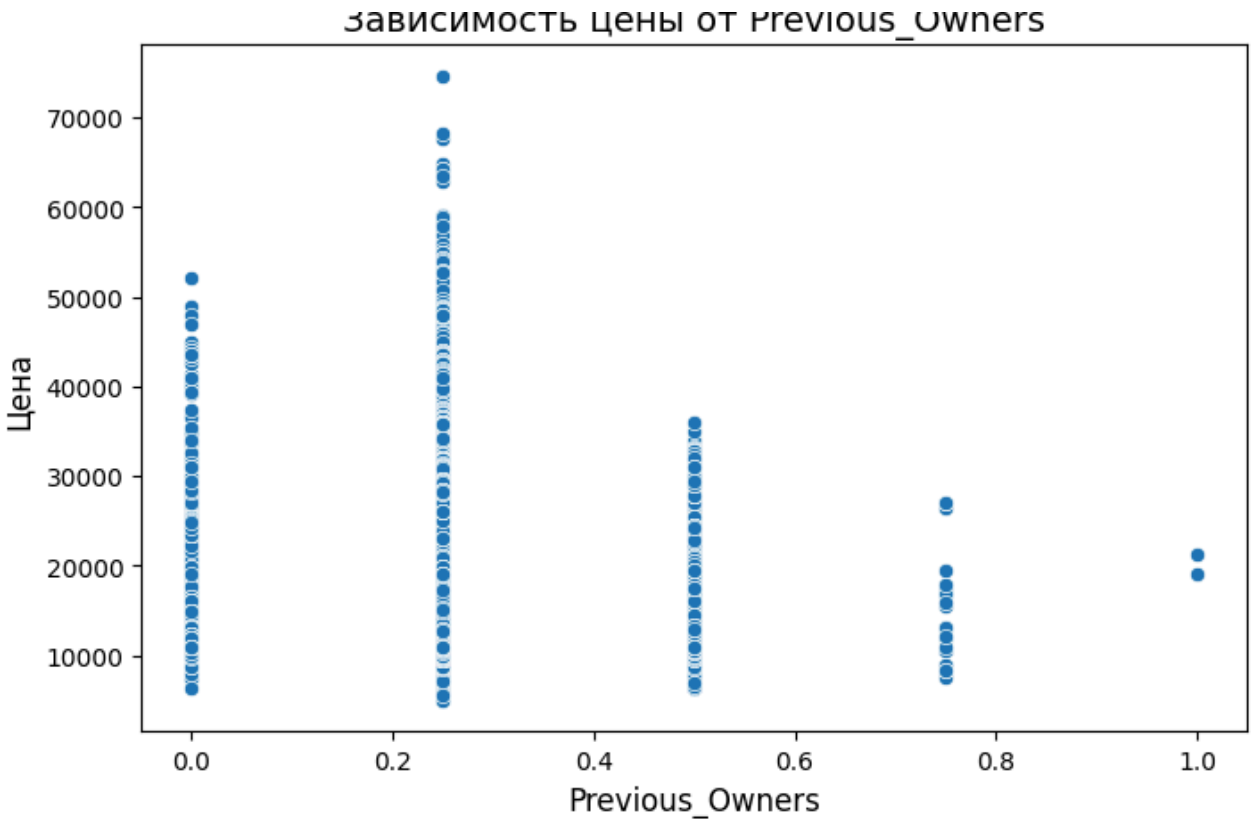

Рисунок 10 - Зависимость цены от Previous_Owners

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)
```

```
Линейная регрессия:
MAE: 1930.41
MSE: 7400475.21
R2: 0.86

Градиентный бустинг:
Лучшие гиперпараметры: {'learning_rate': 0.2, 'max_depth': 5, 'n_estimators': 150}
MAE: 1021.04
MSE: 2617996.69
R2: 0.95

Случайный лес:
Лучшие гиперпараметры: {'max_depth': 20, 'min_samples_leaf': 1, 'min_samples_split': 2, 'n_estimators': 200}
MAE: 937.43
MSE: 2680731.87
R2: 0.95

K-ближайшие соседи:
Лучшие гиперпараметры: {'algorithm': 'auto', 'n_neighbors': 5, 'weights': 'distance'}
MAE: 1605.11
MSE: 8373830.78
R2: 0.84
```

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()
```

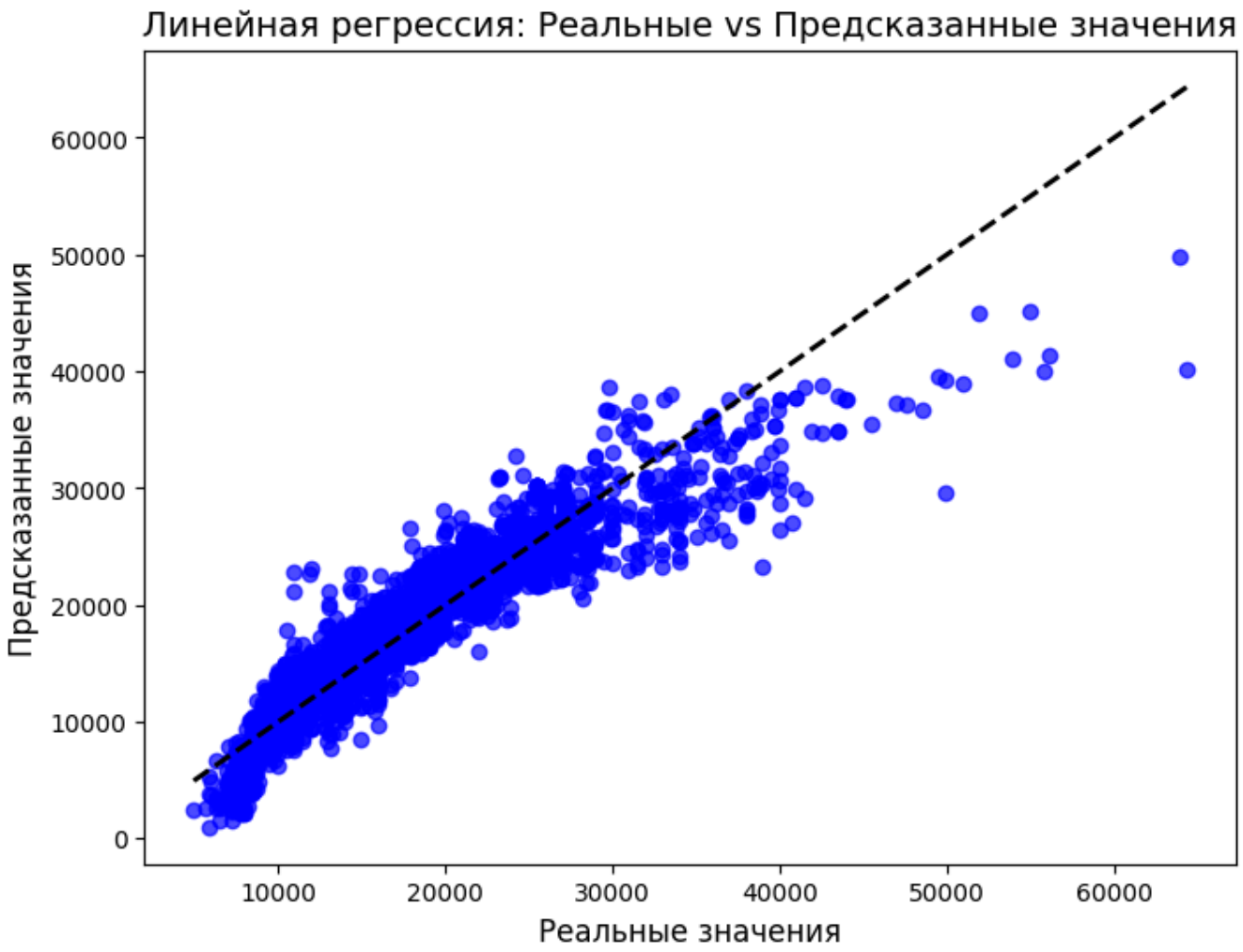

Рисунок 12 - График реальных и предсказанных значений для линейной регрессии

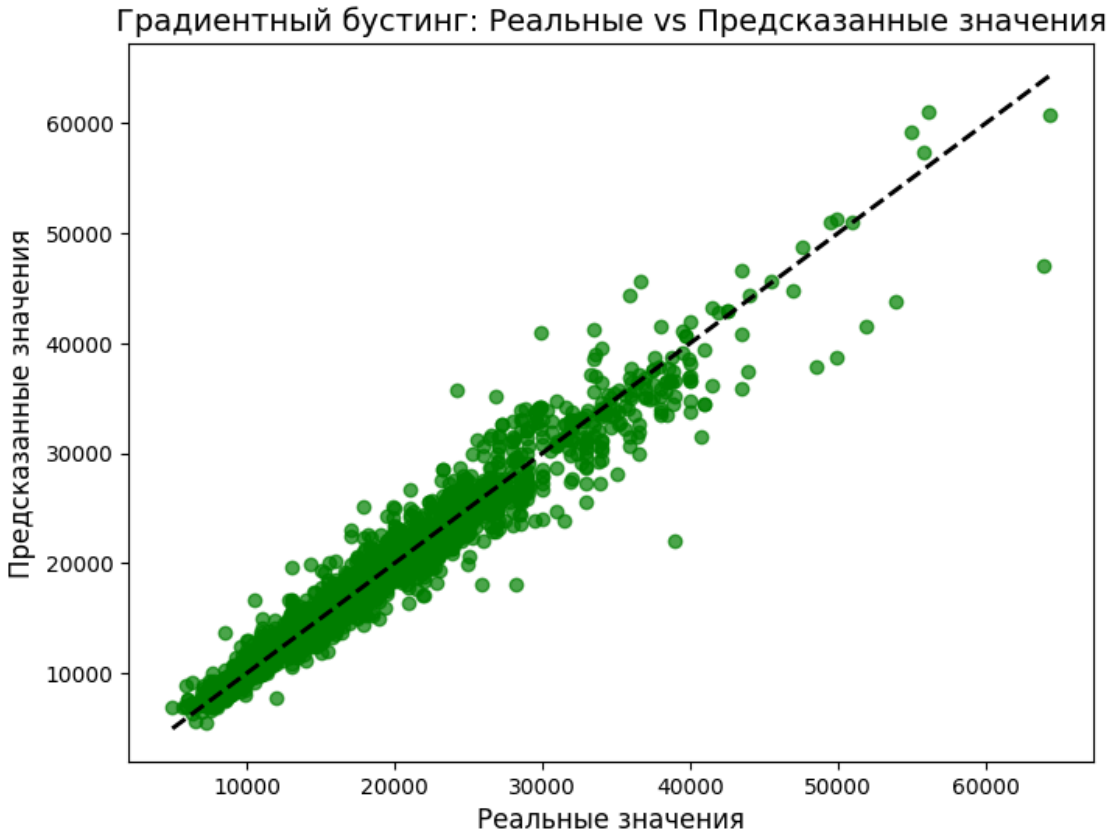

Рисунок 13 - График реальных и предсказанных значений для градиентного бустинга.

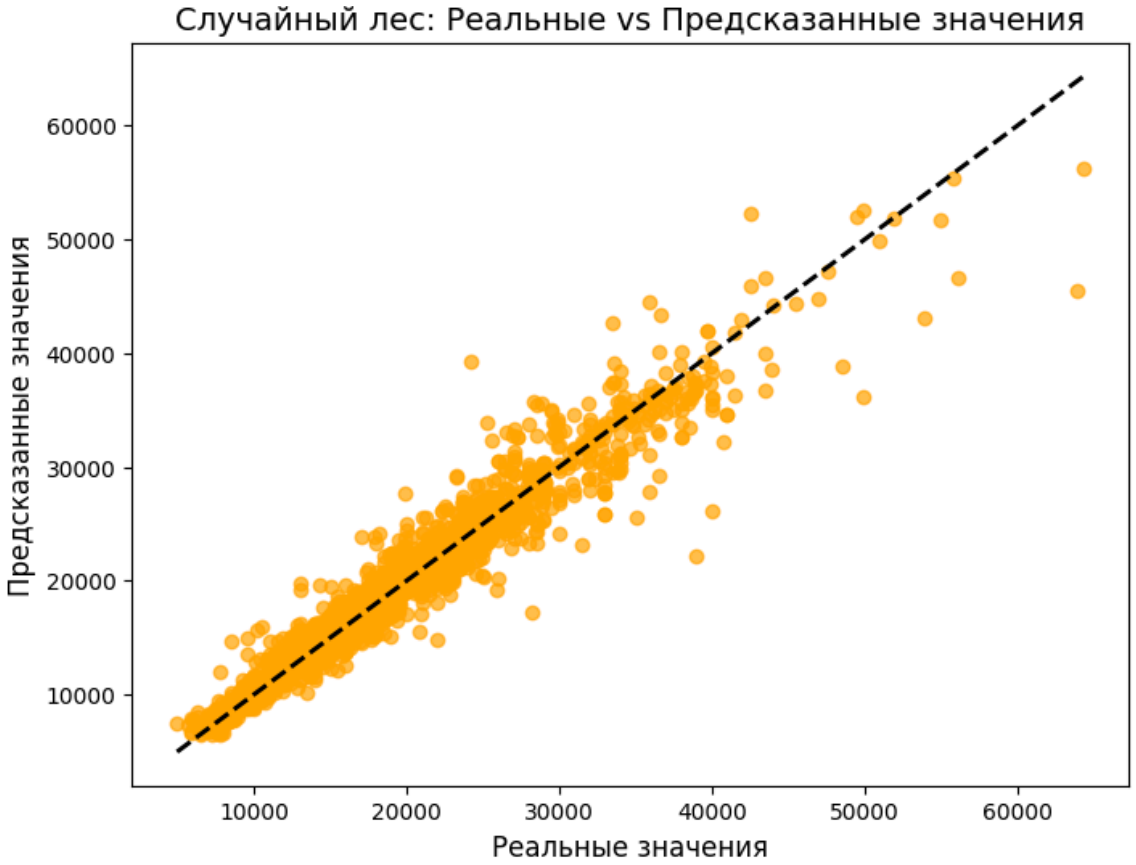

Рисунок 14 - График реальных и предсказанных значений для случайного леса.

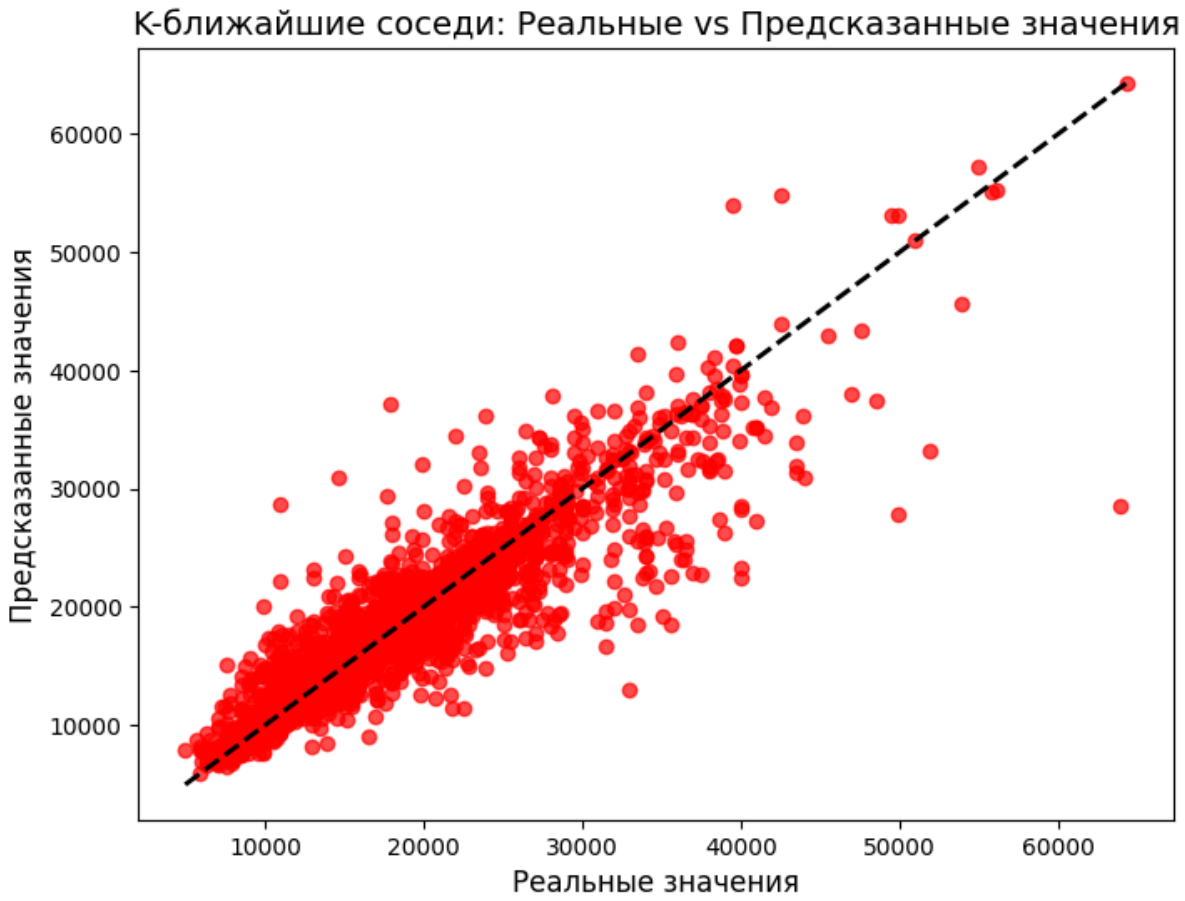

Рисунок 15 - График реальных и предсказанных значений для K-ближайших соседей.

ЗАКЛЮЧЕНИЕ

Таким образом, проведенный анализ данных позволил построить модель, предсказывающую стоимость автомобилей на основании различных признаков. Выдвинутая гипотеза о том, что стоимость автомобилей может зависеть от этих факторов, была подтверждена с помощью построенных моделей.

Наиболее значимое влияние на цену автомобилей оказывают следующие признаки: мощность двигателя (hp_kW), возраст машины (age) и пробег (km). Эти факторы обладают высокой корреляцией с ценой, что было подтверждено при анализе данных.

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 (R^2) 0,95, но случайный лес, несмотря на схожие результаты с градиентным бустингом, имеет более высокие ошибки. Модель линейной регрессии также показала хорошие результаты с R^2 0,86, но её прогнозы были менее точными.

Для модели градиентного бустинга оказалась характерна наибольшая эффективность для конкретно этой задачи, это связано с тем, что она способна захватывать сложные зависимости между переменными, линейная же регрессия хоть и проста и быстра в обучении, не смогла справиться с данной задачей с такой же точностью.

Таким образом, итоговая цель была достигнута, был выполнен анализ данных, после чего построена модель для прогнозирования стоимости автомобилей с достаточно высокой точностью. Результаты, которые были получены в ходе работы позволяют утверждать, что самыми эффективными методами из используемых для прогнозирования является градиентный бустинг и случайный лес. Эти модели могут использоваться в настоящих задачах оценки стоимости автомобилей для разных целей, вроде продажи, страхования или кредитования.

По итогу работы были решены все поставленные задачи, такие как: обработка данных, построение и оценка моделей, а также интерпретация результатов. Полученная итоговая модель может быть использована для дальнейших исследований

и улучшений, например, 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 К-ближайших соседей
printModelScores(rf_model, 'К-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()
```