ГУАП

КАФЕДРА № 41

ОТЧЕТ
ЗАЩИЩЕН С ОЦЕНКОЙ

ПРЕПОДАВАТЕЛЬ

старший преподаватель		В. В. Боженко
должность, уч. степень, звание	подпись, дата	инициалы, фамилия

ОТЧЕТ О ЛАБОРАТОРНОЙ РАБОТЕ №3

РЕГРЕССИОННЫЙ АНАЛИЗ

по курсу: ВВЕДЕНИЕ В АНАЛИЗ ДАННЫХ

РАБОТУ ВЫПОЛНИЛ

СТУДЕНТ ГР. № 4117		Д. С. Николаев
	подпись, дата	инициалы, фамилия

Санкт-Петербург 2023

Цель работы

Изучение алгоритмов и методов регрессии на практике

Индивидуальное задание

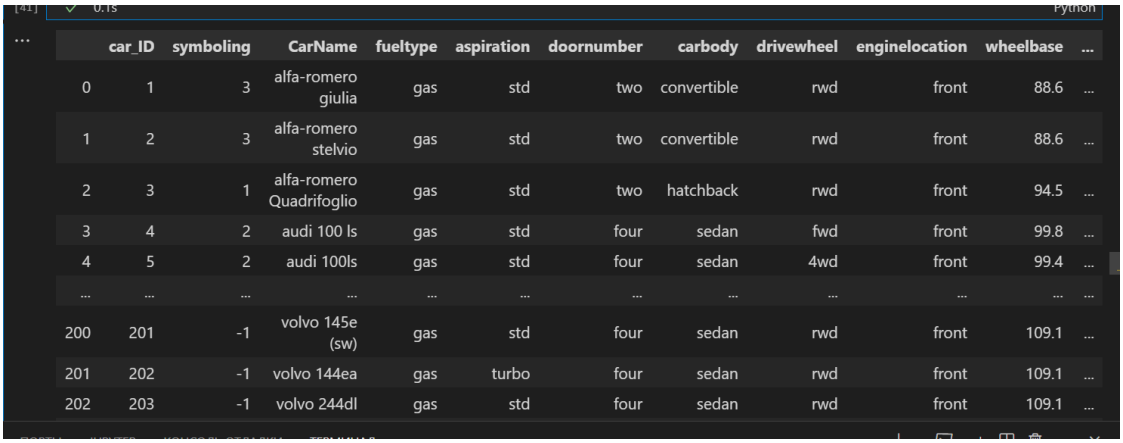
Вариант 7.

Ход работы

Ссылка на GitHub.

Данный отчет содержит часть 4 - решение задачи регрессии различными методами и дополнительные задания.

Для начала был загружен набор данных `car_price.csv` из файлов к выполнению лабораторной работы. Код - `df = pd.read_csv('car_price.csv')`. Результат продемонстрирован на рисунке 1.



	car_ID	symboling	CarName	fueltype	aspiration	doornumber	carbody	drivewheel	enginelocation	wheelbase	...
0	1	3	alfa-romero giulia	gas	std	two	convertible	rwd	front	88.6	...
1	2	3	alfa-romero stelvio	gas	std	two	convertible	rwd	front	88.6	...
2	3	1	alfa-romero Quadrifoglio	gas	std	two	hatchback	rwd	front	94.5	...
3	4	2	audi 100 ls	gas	std	four	sedan	fwd	front	99.8	...
4	5	2	audi 100ls	gas	std	four	sedan	4wd	front	99.4	...
...	...	...	...	...	...	...	...	...	...	...	...
200	201	-1	volvo 145e (sw)	gas	std	four	sedan	rwd	front	109.1	...
201	202	-1	volvo 144ea	gas	turbo	four	sedan	rwd	front	109.1	...
202	203	-1	volvo 244dl	gas	std	four	sedan	rwd	front	109.1	...

Рисунок 1 – Загрузка данных

В качестве целевой переменной был выбран столбец `price`, так как именно цена наиболее интересна в качестве прогнозирования, цена автомобиля является ключевым показателем, который интересует потенциальных покупателей и продавцов.

Далее для выбранной целевой переменной `price` были построены гистограмма и boxplot. С помощью данных графиков можно увидеть какая цена встречается чаще всего. Код гистограммы продемонстрирован на листинге 1, результат на рисунке 2, а код boxplot на листинге 2 и рисунке 3.

Листинг 1 – Код построения гистограммы для `price`

```
plt.figure(figsize= (12, 6))
plt.hist(df['price'], bins=30, color='skyblue', alpha=0.7)
plt.title('Гистограмма целевой переменной')
plt.xlabel('Значения целевой переменной')
plt.ylabel('Частота')
```

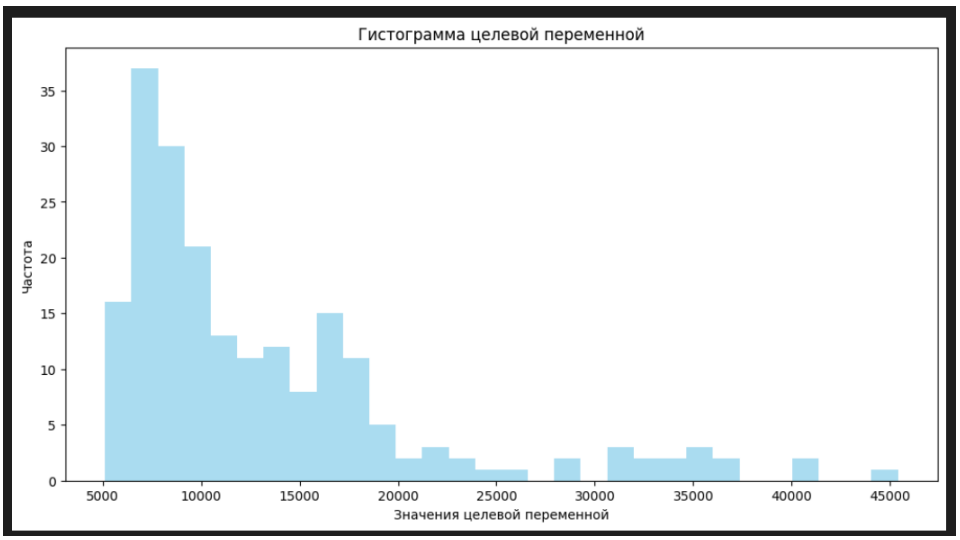


Рисунок 2 – Гистограмма для `price`

Листинг 2 – Код построения boxplot для `price`

```
plt.boxplot(df['price'], vert=False)
plt.title('Boxplot целевой переменной')
plt.xlabel('Значения целевой переменной')
plt.ylabel('Цена')
```

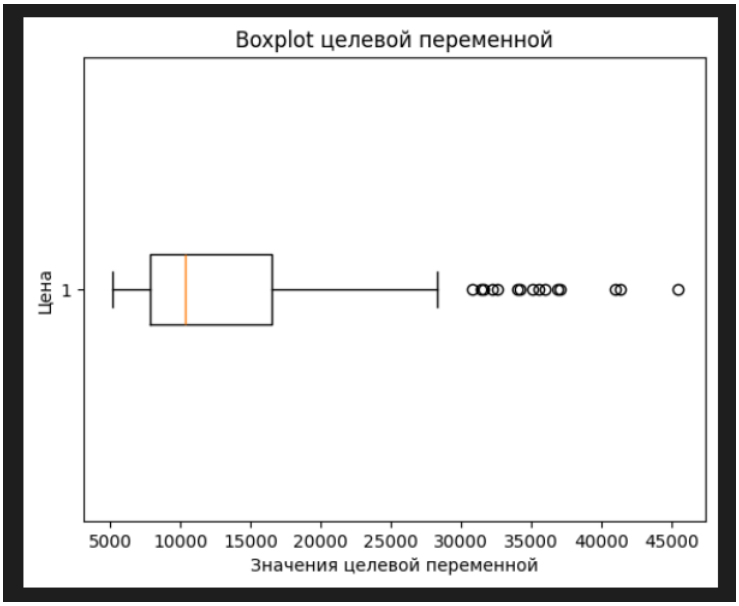


Рисунок 3 – График boxplot для `price`

По данным графикам можно сказать, что чаще всего машины стоят в диапазоне от 5.000 до 15.000. После значения 25.000 появляются выпадающие точки.

Далее была построена матрица диаграмм рассеивания для 'horsepower', 'citympg', 'price', 'enginesize' по 'price'. Для этого используются навыки из предыдущих лабораторных работ и библиотека `seaborn`. Код построения матрицы диаграммы рассеивания -

`sns.pairplot(df[['horsepower', 'citympg', 'price', 'enginesize']], hue = 'price')`. Сам график продемонстрирован на рисунке 4.

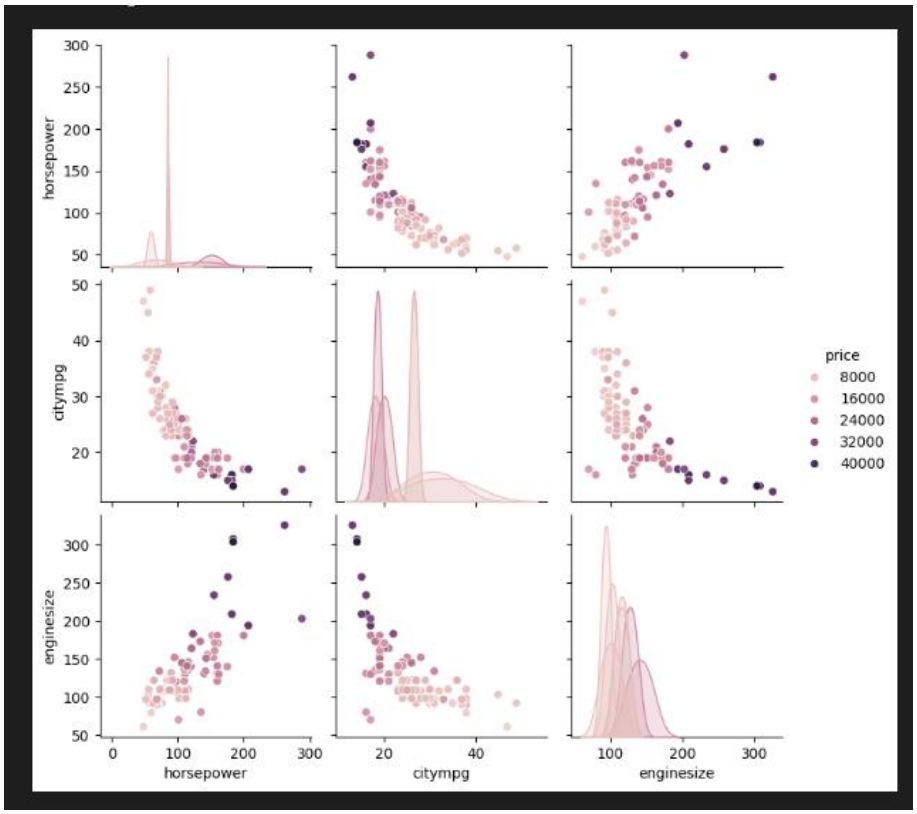


Рисунок 4 – График матриц диаграмм рассеивания

Из графиков видно, что на цену линейно влияет `horsepower` и `enginesize`, то есть при увеличении данных переменных растет и цена.

Далее для деления данных на обучающие и валидационные сначала были выделены числовые столбцы (Листинг 3).

Листинг 3 – Код создания датафрейма из числовых столбцов

```
df_param = df[['symboling', 'wheelbase', 'enginesize', 'boreratio',
'stroke', 'compressionratio', 'horsepower', 'peakrpm', 'citympg',
'highwaympg']]

df_param
```

Числовой столбец `car_ID` не брался, так как никак не влияет на цену, а только лишь нумерует данные.

Для деления данных была использована функция `train_test_split`, в которой параметр `random_state` позволяет получать повторяемые результаты при каждом запуске модели, а сама же модель делится 3 к 1, где 3 - обучающие данные. Код представлен на листинге 4.

Листинг 4 – Деление данных на валидационные и тренировочные

```
X = df_param
y = df['price']
X_train, X_valid, y_train, y_valid = train_test_split(X,
                                                      y,
                                                      test_size=0.25,
                                                      random_state=0)
```

После была проведена нормализация данных через `fit_transform` и была обучена модель линейной регрессии с помощью `LinearRegression`. Для тренировки модели используется метод `fit`, который принимает тренировочные данные модели. С помощью же метода `pridict` происходит вычисление предсказанных точек в соответствии с моделью. Код продемонстрирован на листинге 5.

Листинг 5 – Код обучения линейной модели

```
from sklearn.preprocessing import StandardScaler
sc = StandardScaler()
X_train = sc.fit_transform(X_train)
X_valid = sc.fit_transform(X_valid)
model = LinearRegression()
model.fit(X_train, y_train)
y_predictions = model.predict(X_valid)
```

Также был осуществлен подбор оптимальных параметров с помощью `GridSearchCV` для линейной регрессии, где `parametrs` - словарь с названиями гиперпараметрамов и всеми наборами значений, которые могут передаваться в `LinearRegression()`. `fit_intercept` - логическое значение, указывающее, следует ли подбирать интерсепт, `copy_X` - логическое значение, указывающее, следует ли копировать данные, `n_jobs` - количество рабочих потоков для выполнения задачи и `positive` - логическое значение, указывающее, должны ли предсказанные значения быть положительными (Листинг 6).

Листинг 6 – Код подбора гиперпараметров для линейной регрессии

```
from sklearn.model_selection import GridSearchCV
parametrs = {
    'fit_intercept': [True, False],
    'copy_X': [True, False],
    'n_jobs': list(range(101)),
    'positive': [True, False]
}
grid = GridSearchCV(model, parametrs, scoring="neg_mean_squared_error",
cv=5)
grid.fit(X_train, y_train)
grid.best_params_
```

Исходя из вывода, можно сказать, что в `LinearRegression` лучше всего передавать параметры {'copy_X': True, 'fit_intercept': True, 'n_jobs': 0, 'positive': True}. После чего была заново создана линейная регрессия с другими гиперпараметрами (Листинг 7).

Листинг 7 – Код создания линейной регрессии с полученными гиперпараметрами

```
model = LinearRegression()
model.set_params(copy_X=True, fit_intercept=True, n_jobs=0, positive=True)
model.fit(X_train, y_train)
y_predictions = model.predict(X_valid)
```

Также для подробного анализа были выведены метрики MSE, MAE, RMSR и R2 (Листинг 8, Рисунок 5).

Листинг 8 – Код расчета метрик

```
mae = mean_absolute_error(y_valid, y_predictions)
mse = mean_squared_error(y_valid, y_predictions)
rmse = np.sqrt(mse)
r2 = r2_score(y_valid, y_predictions)
print("Mean Absolute Error:", mae)
print("Mean Squared Error:", mse)
print("Root Mean Squared Error:", rmse)
print("R-squared (R2) Score:", r2)
```

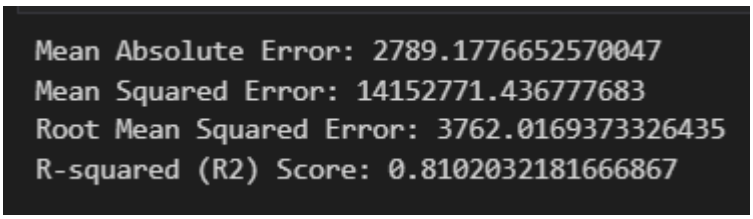
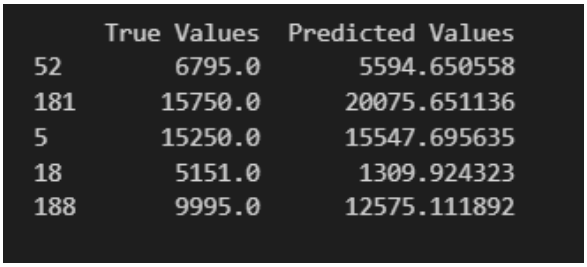


Рисунок 5 – Результат расчета метрик

MAE имеет значение примерно 2789, что указывает среднею разницу между предсказанными и реальными значениями, учитывая, что цена от 5000 до 45000, то значение не слишком большое. MSE и RMSE же имеют большее отклонение, что указывает на наличие больших ошибок в некоторых прогнозах. R2 равное 0.81 указывает на то, что около 81% дисперсии в цене автомобилей может быть объяснено моделью, получается, что модель обладает неплохой объясняющей способностью.

Далее был создан датафрейм с истинными и предсказанными значениями. Код создания датафрейма `results_df = pd.DataFrame({'True Values': y_valid, 'Predicted Values': y_predictions})`, результат продемонстрирован на рисунке 6.

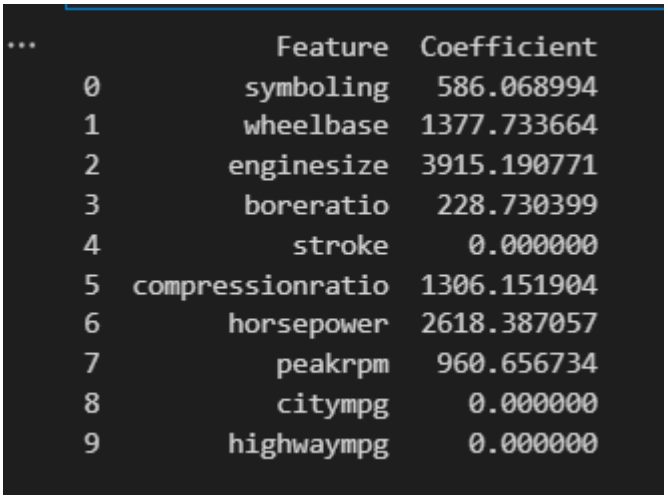


	True Values	Predicted Values
52	6795.0	5594.650558
181	15750.0	20075.651136
5	15250.0	15547.695635
18	5151.0	1309.924323
188	9995.0	12575.111892

Рисунок 6 – Истинные и предсказанные значения

Видно, что есть предсказанные данные, которые почти совпадают с реальными, а есть и наоборот, которые сильно отличаются, что говорит о разбросе данных в целом.

Далее был создан датафрейм с признаками и значением коэффициентов для каждого признака. Код вычисления коэффициентов - `coefficients_df = pd.DataFrame({'Feature': df_param.columns, 'Coefficient': model.coef_})`. Результат продемонстрирован на рисунке 7.



	Feature	Coefficient
0	symboling	586.068994
1	wheelbase	1377.733664
2	enginesize	3915.190771
3	boreratio	228.730399
4	stroke	0.000000
5	compressionratio	1306.151904
6	horsepower	2618.387057
7	peakrpm	960.656734
8	citympg	0.000000
9	highwaympg	0.000000

Рисунок 7 – Коэффициенты линейной регрессии

Положительные коэффициенты указывают на то, что при увеличении данного параметра увеличивается и цена, коэффициенты равные 0, говорят о том, что они никак не влияют на данную модель. Это связано с тем, что были переданы гиперпараметры, которые запрещают предсказывать отрицательные значения. Можно сделать вывод, что машины, имеющие больший размер двигателя больше влияют на его стоимость.

Далее была выполнена визуализация - отображено на графике фактическое и предсказанное значение (Листинг 9, Рисунок 8).

Листинг 9 – Построение графика фактических и предсказанных значений

```
mae = mean_absolute_error(y_valid, y_predictions)
mse = mean_squared_error(y_valid, y_predictions)
rmse = np.sqrt(mse)
r2 = r2_score(y_valid, y_predictions)
print("Mean Absolute Error:", mae)
print("Mean Squared Error:", mse)
print("Root Mean Squared Error:", rmse)
print("R-squared (R2) Score:", r2)
```

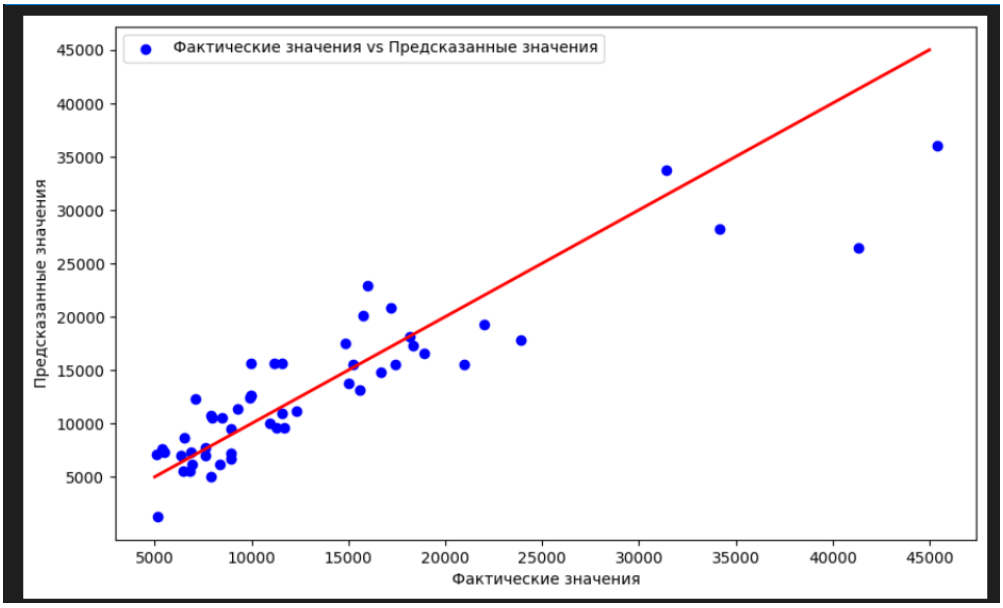


Рисунок 8 – График фактических и предсказанных значений

Из графика можно сделать вывод, что предсказанное значение зачастую выше истинного, но все же близко к нему.

Также была реализована регрессия методом k-ближайших соседей и деревом решений, затем были подсчитаны их метрики. Для подсчета регрессии использовались библиотеки `KNeighborsRegressor` и `DecisionTreeRegressor`. Описание кода продемонстрировано в комментариях (Листинг 10, Рисунок 9).

```
K-ближайших соседей:
Mean Absolute Error: 2468.481407692308
Mean Squared Error: 16281710.699425982
Root Mean Squared Error: 4035.060185353619
R-squared (R2) Score: 0.7816529216700429

Дерево решений:
Mean Absolute Error: 2193.576923076923
Mean Squared Error: 10549067.553418802
Root Mean Squared Error: 3247.932812331376
R-squared (R2) Score: 0.8585309540307979
```

Рисунок 9 – Вывод метрик для k-ближайших соседей и дерева решений

Листинг 10 – Код подсчета метрик для метода k-ближайших соседей и дерева решений

```
from sklearn.neighbors import KNeighborsRegressor

from sklearn.tree import DecisionTreeRegressor

# Инициализация моделей

knn_model = KNeighborsRegressor(n_neighbors=5)# Пример, можно выбрать
другое количество соседей

dt_model = DecisionTreeRegressor()

# Обучение моделей

knn_model.fit(X_train, y_train)

dt_model.fit(X_train, y_train)

# Предсказания для K-ближайших соседей и дерева решений

knn_predictions = knn_model.predict(X_valid)

dt_predictions = dt_model.predict(X_valid)

# Расчет метрик для K-ближайших соседей и дерева решений

knn_mae = mean_absolute_error(y_valid, knn_predictions)

knn_mse = mean_squared_error(y_valid, knn_predictions)

knn_rmse = np.sqrt(knn_mse)

knn_r2 = r2_score(y_valid, knn_predictions)

dt_mae = mean_absolute_error(y_valid, dt_predictions)

dt_mse = mean_squared_error(y_valid, dt_predictions)

dt_rmse = np.sqrt(dt_mse)

dt_r2 = r2_score(y_valid, dt_predictions)

print("K-ближайших соседей:")

print("Mean Absolute Error:", knn_mae)

print("Mean Squared Error:", knn_mse)

print("Root Mean Squared Error:", knn_rmse)

print("R-squared (R2) Score:", knn_r2)

print("\nДерево решений:")

print("Mean Absolute Error:", dt_mae)

print("Mean Squared Error:", dt_mse)

print("Root Mean Squared Error:", dt_rmse)

print("R-squared (R2) Score:", dt_r2)
```

По данным результатам можно сказать, что метод дерева решений справился лучше, так как R2 ближе к 1, а MSE, RMSE и MAE меньше.

Также был построен график фактического и предсказанного значения для регрессия методом k-ближайших соседей, деревом решений и линейной (Листинг 11, Рисунок 10).

Листинг 11 – Код построения графика разных моделей

```
value = [5000, 10000, 15000, 20000, 25000, 30000, 35000, 40000, 45000]
plt.figure(figsize=(12, 6))
plt.plot(value, value, color='black', linewidth=2)
plt.scatter(y_valid, y_predictions, c='blue', label='Линейная регрессия')
plt.xlabel('Фактические значения')
plt.ylabel('Предсказанные значения')
plt.scatter(y_valid, knn_predictions, c='green', label='K-ближайших
соседей')
plt.scatter(y_valid, dt_predictions, c='red', label='Дерево решений')
plt.xlabel('Фактические значения')
plt.ylabel('Предсказанные значения')
plt.legend()
plt.xticks(value)
plt.yticks(value)
plt.show()
```

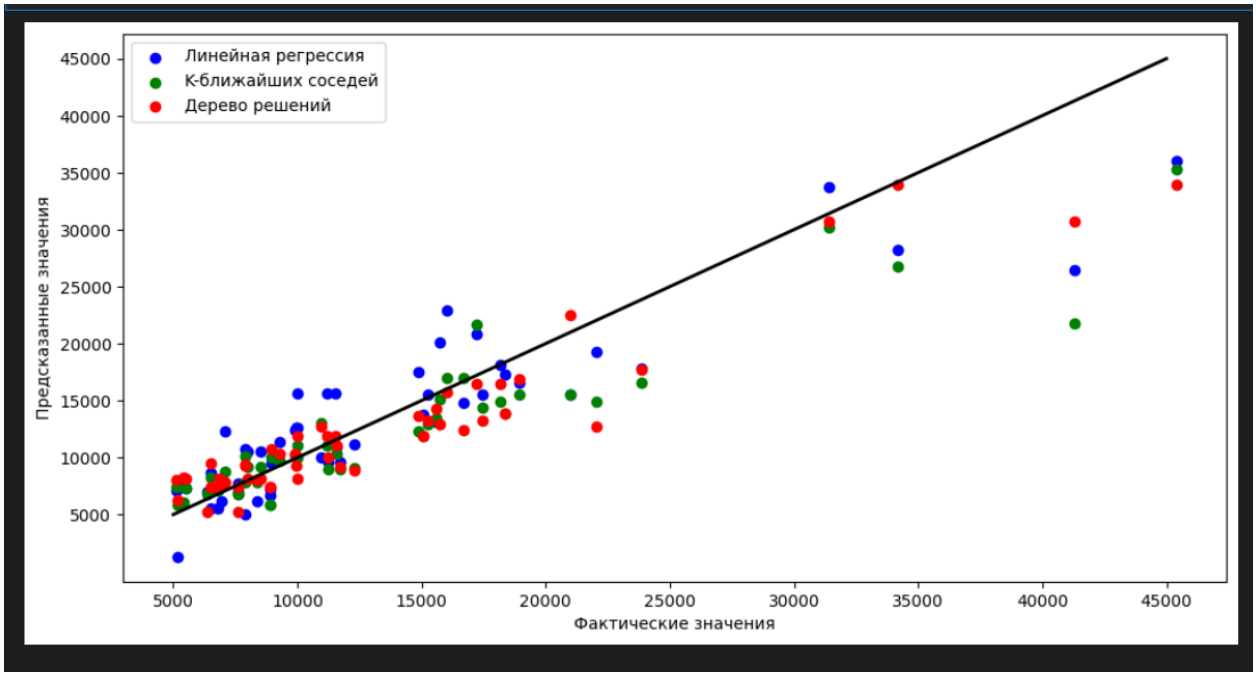


Рисунок 10 – График фактический и предсказанных точек для разных моделей

График также подтверждает, что метод дерево решений предсказанные значения ближе к истинным.

1. Метод линейной регрессии: Линейная регрессия подходит для первоначального анализа данных, но может быть недостаточно гибкой для сложных зависимостей.

2. Метод ближайших соседей: Хороший метод для нелинейных зависимостей, но требует аккуратного подбора параметров.

3. Метод дерева решений: Дает хорошие результаты в данном случае. Важно контролировать глубину дерева и другие параметры, чтобы избежать переобучения.

Целесообразно использовать метод дерева решений, так как он обладает хорошей способностью к адаптации к нелинейным зависимостям и показывает лучшие результаты на данном наборе данных.

Далее были выполнены дополнительные задания.

Задание 1 - Из 4 части убрать выпадающие точки и сравнить модели. Была проделана такая же работа, что и в 4 части, но значения `price` брались больше 25.000, так как именно после 25.000 начинались выпадающие точки. Код представлен в Приложении А, а результат на рисунке 11.

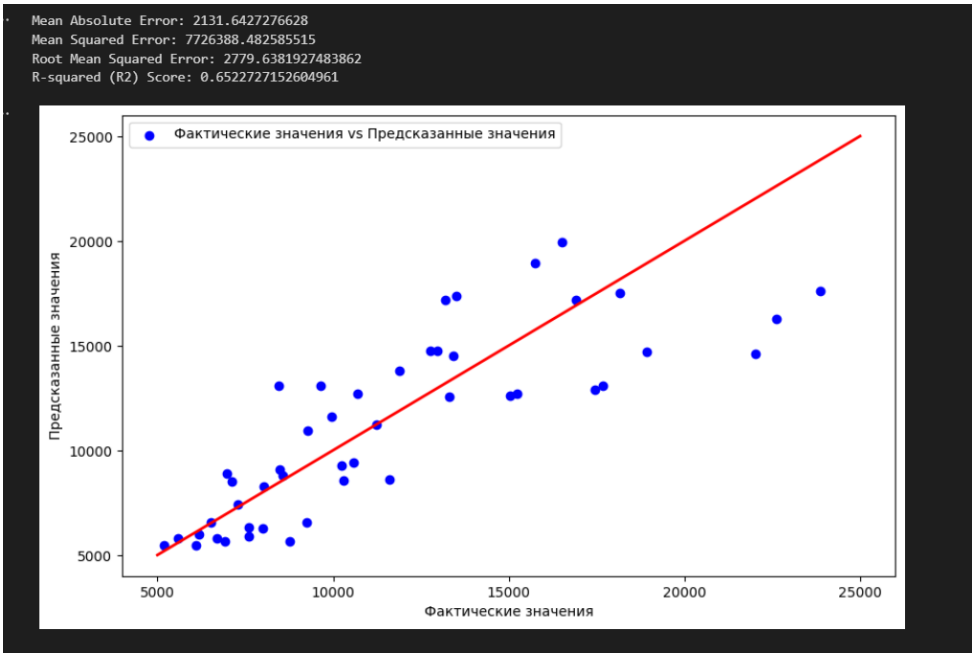


Рисунок 11 – Модель без выпадающих точек

Сравнив метрики, было выделено, что MAE, MSE и RMSE уменьшились, что говорит о том, что данные стали ближе к фактическим, среднее отклонение предсказанных точек уменьшилось. Однако R2 стало меньше, а чем меньше R2, тем меньше объясняющей способностью модели. Что в принципе и ожидалось, так как были убраны большие значения, следовательно вторая модель не сможет их точно объяснить или предсказать.

Задание 2 - Выполнить визуализацию важности признаков для линейной регрессии. Оставить в модели несколько наиболее важных признаков и заново обучить её. Сравнить

качество модели, в которой были все признаки и качество модели с наиболее важными.

Для начала был построен график, на котором можно будет выделить наиболее важные коэффициенты (Листинг 12, Рисунок 12).

Листинг 12 – Построение графика коэффициентов

```
pd.DataFrame({"Feature":df_param.columns,"Coefficients":model.coef_.flatten
().T})

plt.barh(df_param.columns, model.coef_.flatten())
```

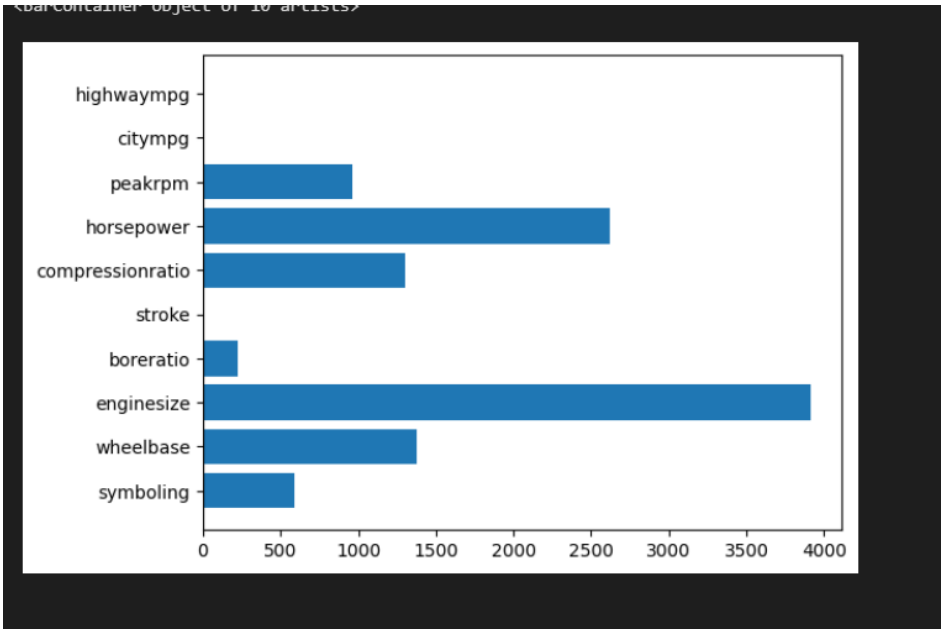


Рисунок 12 – График коэффициентов

Были выделены несколько наиболее важных признаков - `enginesize` и `hoursepower`. По ним уже были посчитаны метрики и выведен график. Код продемонстрирован в Приложении Б, а результат на рисунке 13.

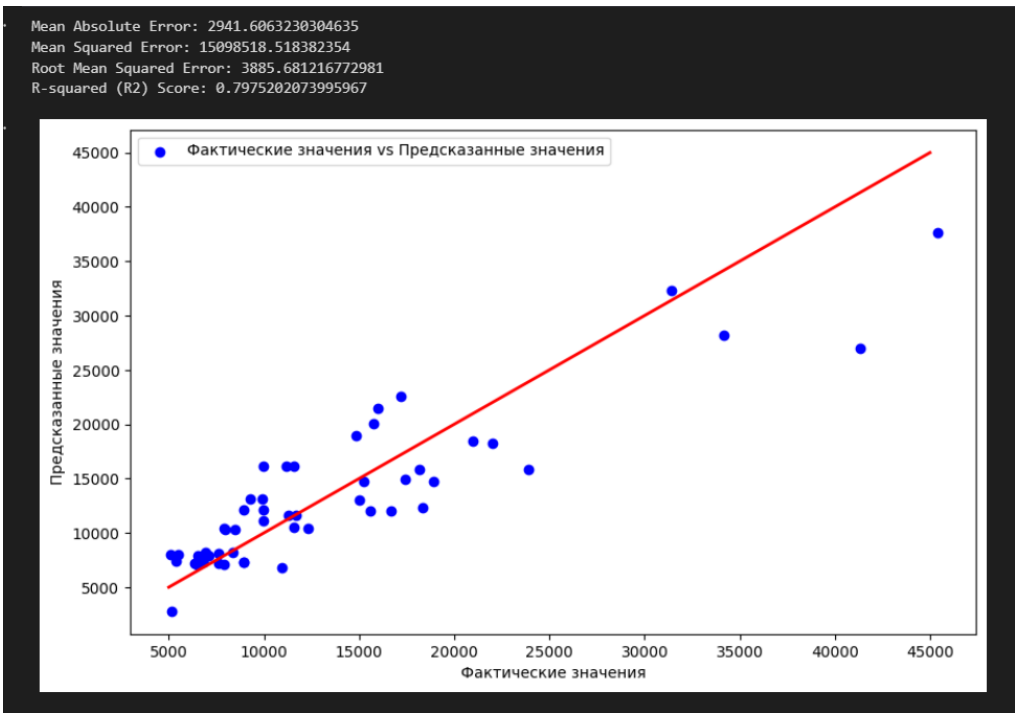


Рисунок 13 – Модель по важным коэффициентам

Анализируя данные, можно сказать, что модель стала справляться хуже. Это может быть связано с тем, что при удалении признаков, теряется часть информации о данных. В целом значения отличаются не на много, добавление большего количества признаков, лишь не на много улучшает модель.

Вывод

В данной работе был проведен обширный анализ данных с использованием различных статистических и машинного обучения методов. В первых трех частях, были выполнены задания, позволяющие научится различным методам машинного обучения. В последней же части были пременены навыки на обширном наборе данных. А именно:

1. Предобработка данных: Первоначально был загружен и проанализирован набор данных car_price.csv. После этого была выбрана целевая переменная - price, которую следовало предсказать. Проведен анализ гистограммы и boxplot, что позволило понять распределение цен и выявить возможные выбросы в данных.

2. Визуализация и анализ взаимосвязей: Была построена матрица диаграмм рассеивания для нескольких признаков относительно цены. Анализируя эти графики, было выявлено, что horsepower и enginesize имеют линейную зависимость с ценой, что подтверждает их важность как предсказательных признаков.

3. Подготовка данных и обучение модели: Данные были разделены на обучающую и тестовую выборки, а затем стандартизированы с использованием StandardScaler. Затем была применена модель линейной регрессии с оптимизацией гиперпараметров через GridSearchCV. Результаты указывают на то, что модель имеет хорошую способность объяснения вариации цен автомобилей, с коэффициентом детерминации R2, близким к 0.83.

4. Сравнение с другими моделями: Дополнительно были реализованы регрессия методом k-ближайших соседей и деревом решений. Сравнение моделей показало, что дерево решений дает более точные прогнозы, что подчеркивает его эффективность в данной задаче.

5. Анализ важности признаков: С помощью коэффициентов регрессии была оценена важность признаков. Horsepower, enginesize, citympg, и highwaympg оказались наиболее важными признаками, влияющими на цену автомобиля.

Работа продемонстрировала важность анализа данных и выбора правильных признаков для построения точных моделей прогнозирования. Использование метода дерева решений в данной задаче позволило достичь наилучших результатов. Однако, всегда стоит продолжать эксперименты с различными моделями и параметрами, чтобы улучшить точность прогнозов в будущем.

Приложения

Приложение А – Код построения модели без выпадающих точек

```
df2 = df.query('price < 25000')

df_param2 = df2[['symboling', 'wheelbase', 'enginesize', 'boreratio',
'stroke', 'compressionratio', 'horsepower', 'peakrpm', 'citympg',
'highwaympg']]

X2 = df_param2

y2 = df2['price']

X_train2, X_valid2, y_train2, y_valid2 = train_test_split(X2,
                                                          y2,
                                                          test_size=0.25,
                                                          random_state=0)

X_train2 = sc.fit_transform(X_train2)

X_valid2 = sc.fit_transform(X_valid2)

model2 = LinearRegression()

model2.set_params(copy_X=True, fit_intercept=True, n_jobs=0, positive=True)

model2.fit(X_train2, y_train2)

y_predictions2 = model2.predict(X_valid2)

mae = mean_absolute_error(y_valid2, y_predictions2)

mse = mean_squared_error(y_valid2, y_predictions2)

rmse = np.sqrt(mse)

r2 = r2_score(y_valid2, y_predictions2)

print("Mean Absolute Error:", mae)

print("Mean Squared Error:", mse)

print("Root Mean Squared Error:", rmse)

print("R-squared (R2) Score:", 2)

value2 = [5000, 10000, 15000, 20000, 25000]

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

plt.plot(value2, value2, color='red', linewidth=2)

plt.scatter(y_valid2, y_predictions2, c='blue', label='Фактические значения
vs Предсказанные значения')

plt.xlabel('Фактические значения')

plt.ylabel('Предсказанные значения')

plt.xticks(value2)

plt.yticks(value2)

plt.legend()

plt.show()
```

Приложение Б – Код построения модели по наилучшим коэффициентам

```
df_param3 = df[['enginesize', 'horsepower']]
X3 = df_param3
y3 = df['price']
X_train3, X_valid3, y_train3, y_valid3 = train_test_split(X3,
                                                          y3,
                                                          test_size=0.25,
                                                          random_state=0)
X_train3 = sc.fit_transform(X_train3)
X_valid3 = sc.fit_transform(X_valid3)
model3 = LinearRegression()
model3.set_params(copy_X=True, fit_intercept=True, n_jobs=0, positive=True)
model3.fit(X_train3, y_train3)
y_predictions3 = model3.predict(X_valid3)
mae = mean_absolute_error(y_valid3, y_predictions3)
mse = mean_squared_error(y_valid3, y_predictions3)
rmse = np.sqrt(mse)
r2 = r2_score(y_valid3, y_predictions3)
print("Mean Absolute Error:", mae)
print("Mean Squared Error:", mse)
print("Root Mean Squared Error:", rmse)
print("R-squared (R2) Score:", r2)
value3 = [5000, 10000, 15000, 20000, 25000, 30000, 35000, 40000, 45000]
plt.figure(figsize=(10, 6))
plt.plot(value3, value3, color='red', linewidth=2)
plt.scatter(y_valid3, y_predictions3, c='blue', label='Фактические значения
vs Предсказанные значения')
plt.xlabel('Фактические значения')
plt.ylabel('Предсказанные значения')
plt.xticks(value3)
plt.yticks(value3)
plt.legend()
plt.show()
```