ГУАП

КАФЕДРА № 41

ОТЧЕТ   
ЗАЩИЩЕН С ОЦЕНКОЙ

ПРЕПОДАВАТЕЛЬ

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ст. 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. № | 4216 |  |  |  | С.Д. Комолова |
|  |  |  | подпись, дата |  | инициалы, фамилия |

Санкт-Петербург 2024

**Цель лабораторной работы:** изучение алгоритмов и методов регрессии на практике.

**Индивидуальное задание:**

Вариант 1:

Задание 1: Простая линейная регрессия

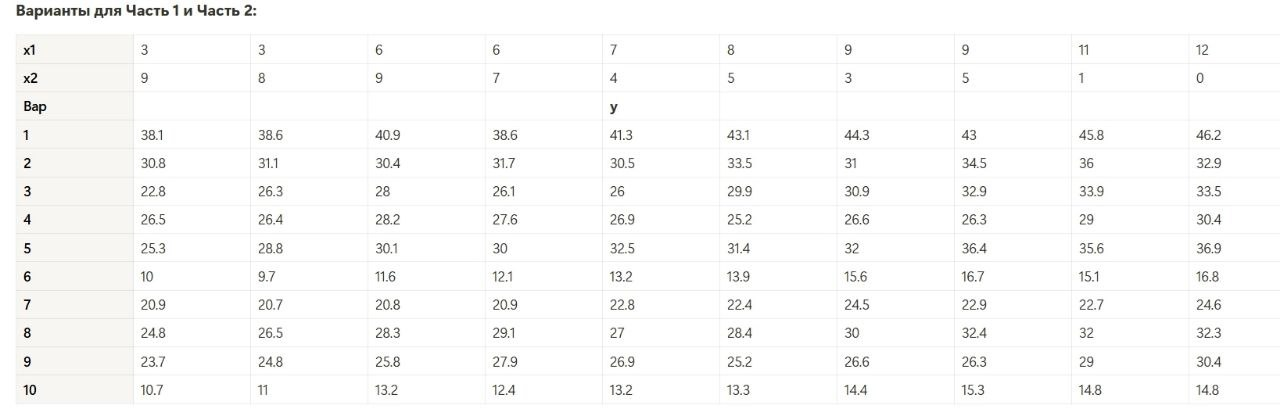


Рисунок 1 – Вариант 1 для задания 1

Задание 2: Полиномиальная регрессия

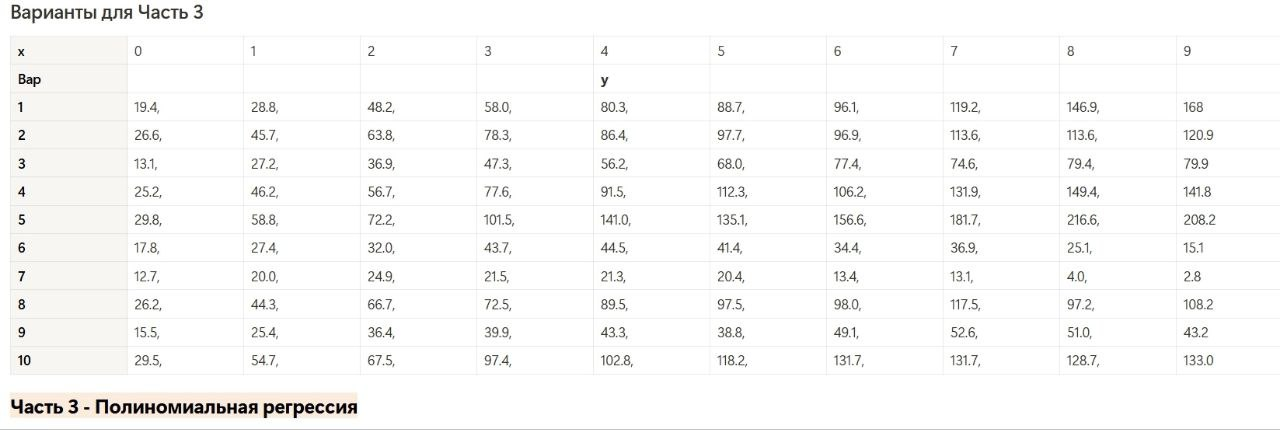


Рисунок 2 – Вариант 2 для задания 2

Задание 3: Решение задачи регрессии различными методами

**Ход работы:**

Ссылка на репозиторий github: <https://github.com/sufferix/data-analysis>

**Задание 1:**

Для начала работы были установлены необходимые для работы библиотеки

import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

import seaborn as sns

Для выполнения первого задания необходимо создать датафрейм из значений x1 и y

data = { "x1": [3, 3, 6, 6, 7, 8, 9, 9, 11, 12],

"y": [38.1, 38.6, 40.9, 38.6, 41.3, 43.1, 44.3, 43.0, 45.8, 46.2]}

df = pd.DataFrame(data)

Далее был импортирован метод LinearRegression из библиотеки sklearn, позволяющая составлять линейную регрессию из созданного датафрейма. Чтобы обучить модель, используется метод fit()

from sklearn.linear\_model import LinearRegression

X = df[["x1"]]

y = df["y"]

model = LinearRegression()

model.fit(X, y)

C помощью метода predict() составляются предсказанные значения на основе проведенного обучения

predictions = model.predict(X)

results\_df = pd.DataFrame({"True Values": y, "Predicted Values": predictions})

Чтобы оценить, насколько хорошо модель предсказывает значения, используются метрики качества регрессии. Основными метриками являются:

* Средняя квадратичная ошибка MSE
* Средняя абсолютная ошика MAE
* Среднеквадратичная ошибка RMSE
* Коэффициент детерминации R2

Из библиотеки sklearn были импортированы методы данных метрик: mean\_squared\_error, mean\_absolute\_error, r2\_score, которые в качестве параметров берут исходные значения y и предсказанные

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

mse = mean\_squared\_error(y, predictions)

mae = mean\_absolute\_error(y, predictions)

rmse = np.sqrt(mse)

r2 = r2\_score(y, predictions)

Также была посчитана формула самой линии регрессии, которая имеет вид y = ax + b, где a - это уровень наклона линии, а b - смещение по оси у Была выведена таблица исходных значений y и предсказанных, а также выведены значения всех метрик

a = model.coef\_[0]

b = model.intercept\_

display(results\_df)

print(f' MSE: {mse}\n MAE: {mae}\n RMSE: {rmse}\n R2: {r2}\n a: {a}\n b: {b}')

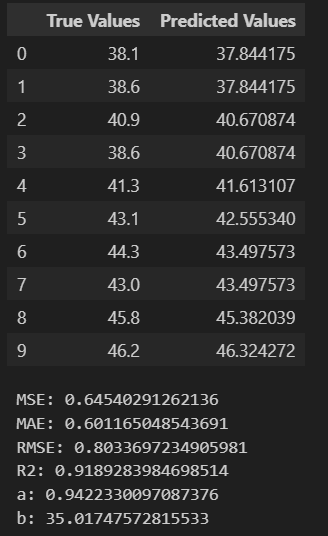


Рисунок 3 – Расчет метрик

По данной таблице можно сказать, что почти все предсказанные значения очень близки к исходным, на это также указывают маленькие значения метрик

Далее был выведен график типа scatter, на котором точками указаны истинные значения y, а линией - линия регрессии по предсказанным значениям

plt.figure(figsize=(10, 5))

plt.scatter(X, y, color="blue", label="True Values")

plt.plot(X, predictions, color="red", label="Regression Line")

plt.xlabel("x1")

plt.ylabel("y")

plt.title("График линейной регрессии")

plt.legend()

plt.show()

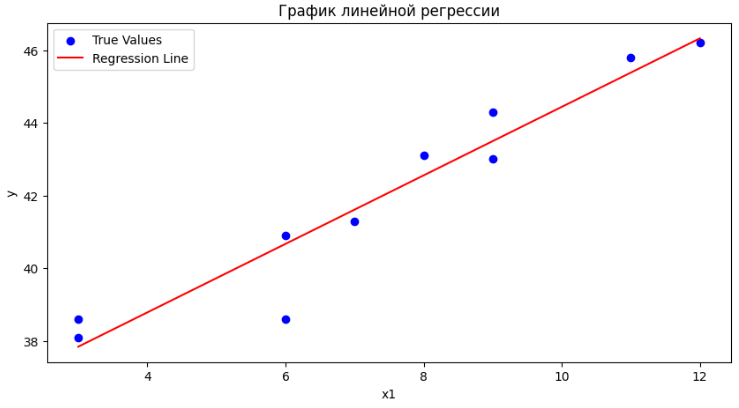


Рисунок 4 – График линейной регрессии

По данному графику можно сказать, что исходные точки находятся около линии регрессии, также наблюдается один выброс по отношению к остальным точкам

Далее был выведен график разницы значений исходных точек от предсказанных

plt.figure(figsize=(10, 5))

plt.plot(df.index, y - predictions, marker='o', linestyle='-', color="green")

plt.xlabel("Индексы")

plt.ylabel("Разница")

plt.title("График разницы значений исходных точек от предсказанных")

plt.show()

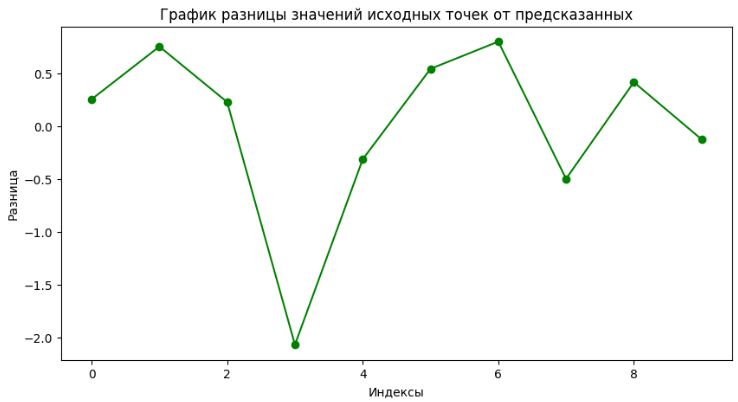


Рисунок 5 – График разницы значений исходных точек от предсказанных

По данному графику можно сказать, что модель выдавала предсказанные точки, которые могли быть как меньше исходного значения, так и больше, максимальная разница составляет -2

**Задание 2**

Был создан второй датафрейм из других значений

data2 = {

"x": [0, 1, 2, 3, 4, 5, 6, 7, 8, 9],

"y": [19.4, 28.8, 48.2, 58.0, 80.3, 88.7, 96.1, 119.2, 146.9, 168.0]

}

df2 = pd.DataFrame(data2)

X = df2[["x"]]

y = df2["y"]

Далее было проведено обучение модели для создания полиномиальной регрессии 2 и 3 степени. Для создания полиномиальной регрессии использовался метод PolynomialFeatures библиотеки sklearn. Также были рассчитаны метрики качества регрессии для данных графиков

from sklearn.preprocessing import PolynomialFeatures

degrees = [2, 3, 4, 5]

for degree in degrees:

poly = PolynomialFeatures(degree=degree)

X\_poly = poly.fit\_transform(X)

model = LinearRegression()

model.fit(X\_poly, y)

predictions = model.predict(X\_poly)

mse = mean\_squared\_error(y, predictions)

mae = mean\_absolute\_error(y, predictions)

rmse = np.sqrt(mse)

r2 = r2\_score(y, predictions)

a = model.coef\_[1]

b = model.intercept\_

plt.figure(figsize=(10, 5))

plt.scatter(X, y, color="blue", label="True Values")

plt.plot(X, predictions, color="red", label=f"Полиномиальная регрессия степени {degree}")

plt.xlabel("x")

plt.ylabel("y")

plt.title(f"Полиномиальная регрессия степени {degree}")

plt.legend()

plt.show()

print(f'Метрики для степени {degree}:\n MSE: {mse}\n MAE: {mae}\n RMSE: {rmse}\n R2: {r2}\n a: {a}\n b: {b}')

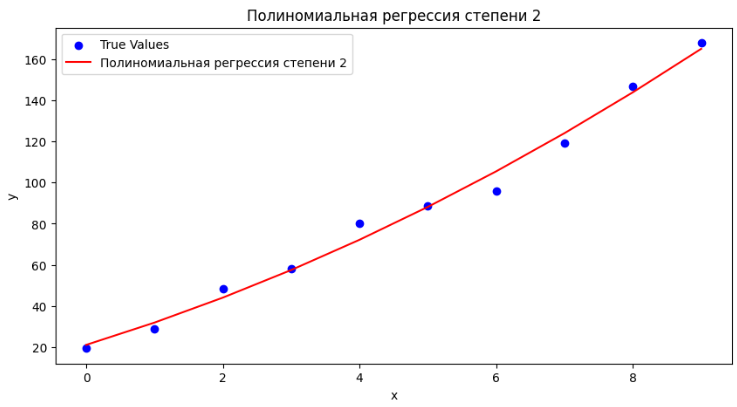


Рисунок 6 – Полиномиальная регрессия степени 2

Метрики для степени 2:

MSE: 22.49771363636362

MAE: 3.810181818181836

RMSE: 4.74317548024144

R2: 0.9895810075919272

a: 10.212045454545443

b: 21.04272727272736

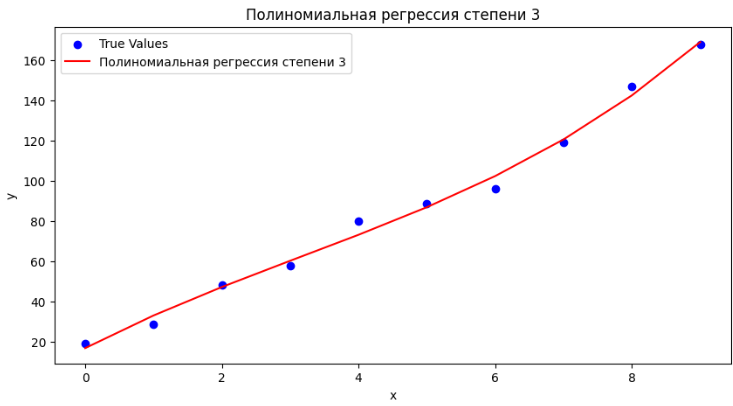


Рисунок 7 - Полиномиальная регрессия степени 3

Метрики для степени 3:

MSE: 14.643671561771558

MAE: 3.2072027972027413

RMSE: 3.8267050528844733

R2: 0.9932183196348539

a: 17.56315073815133

b: 17.02433566433544

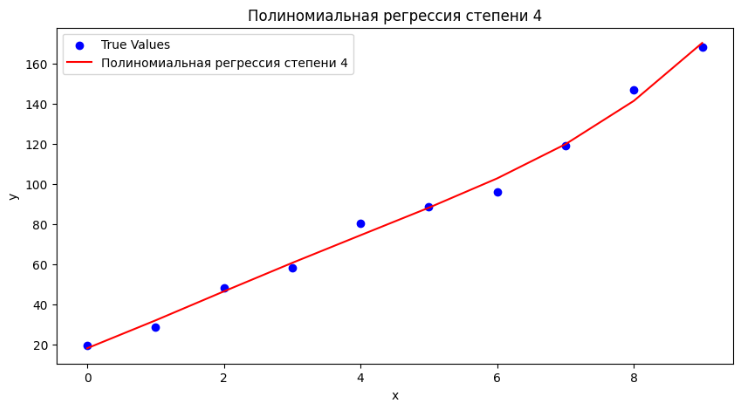


Рисунок 8 - Полиномиальная регрессия степени 4

Метрики для степени 4:

MSE: 13.67668939393938

MAE: 3.032762237762728

RMSE: 3.698200832018102

R2: 0.993666142023752

a: 13.202136752134612

b: 18.070979020979166

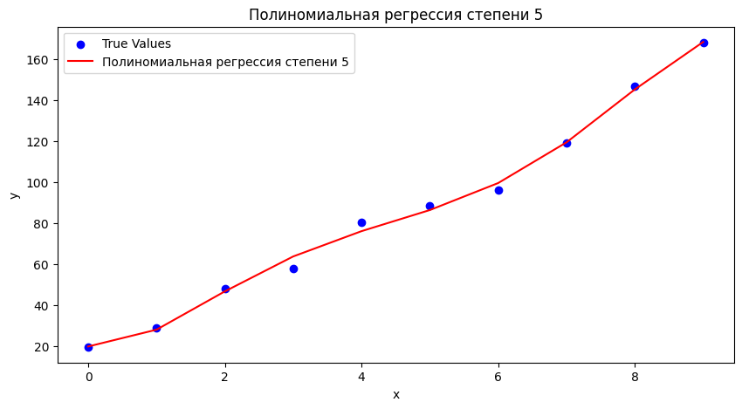


Рисунок 7 - Полиномиальная регрессия степени 5

Метрики для степени 5:

MSE: 7.3980111888112265

MAE: 2.0783496503491032

RMSE: 2.719928526415947

R2: 0.9965738819661001

a: -4.02899999998908

b: 19.77328671328341

По данным графикам и метрикам можно заметить, что полином 5-й степени показывает лучшие результаты, поскольку он лучше подстраивается под точки, так как чем выше степень полинома, тем больше модель способна учитывать локальные особенности, такие как резкие изменения и колебания значений. Также у полинома 5-й степени наблюдаются наиболее низкие значения ошибок и наиболее высокий R2 - 0,996, что указывает на лучшую точность среди остальных представленных полиномов.

**Задание 3**

Для выполнения третьего задания необходимо загрузить датасет carprice

df = pd.read\_csv('carprice.csv',sep=',')

df.head(5)

После вывода нескольких строк датасета можно увидеть его содержание. Всего в датасете 26 столбцов, большинство из которых являются числовыми.

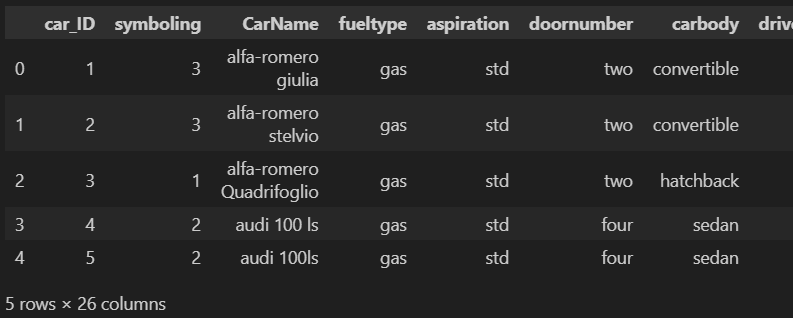


Рисунок 8 – Вывод первых 5 строк датасета

За целевую переменную был взять столбец price, так как он наиболее подходящий для выстраивания предсказаний значений. Была выведена гистограмма по данному столбцу

sns.histplot(df['price'], kde=True)

plt.title('Гистограмма стоимости автомобиля')

plt.show()

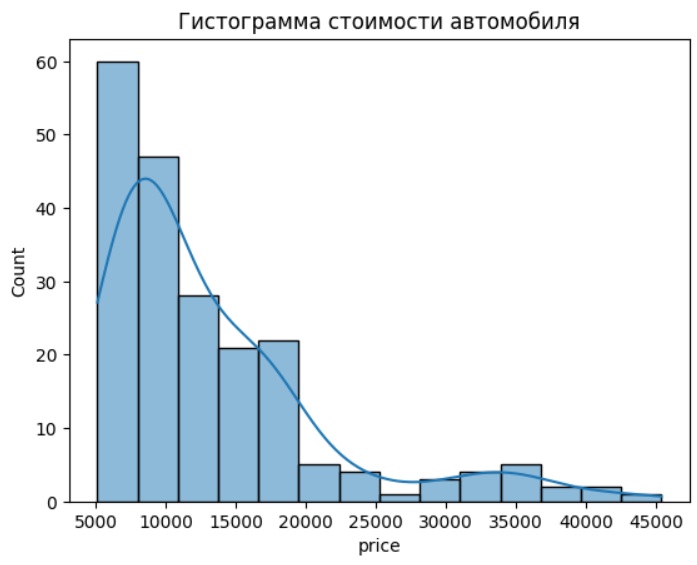


Рисунок 9 – Гистограмма стоимости автомобиля

По данной гистограмме можно понять, что больше всего в датасете наблюдается дешевые автомобили

Был также построен график вида boxplot

sns.boxplot(y=df['price'])

plt.title('Boxplot стоимости автомобиля')

plt.show()

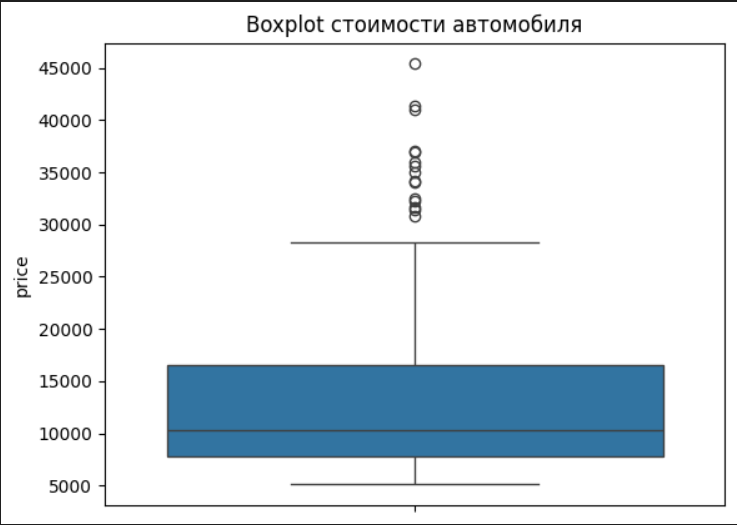


Рисунок 10 – Boxplot стоимости автомобиля

Основной диапазон стоимости автомобилей находится примерно между 5 000 и почти 30 000 денег, медиана стоимости автомобилей составляет 10 000 денег. Также выше 30 000 присутствуют выбросы, которые показывают, что в выборке не так много дорогих машин

Была построена матрица диаграмм

sns.pairplot(df.drop(columns=['car\_ID']))

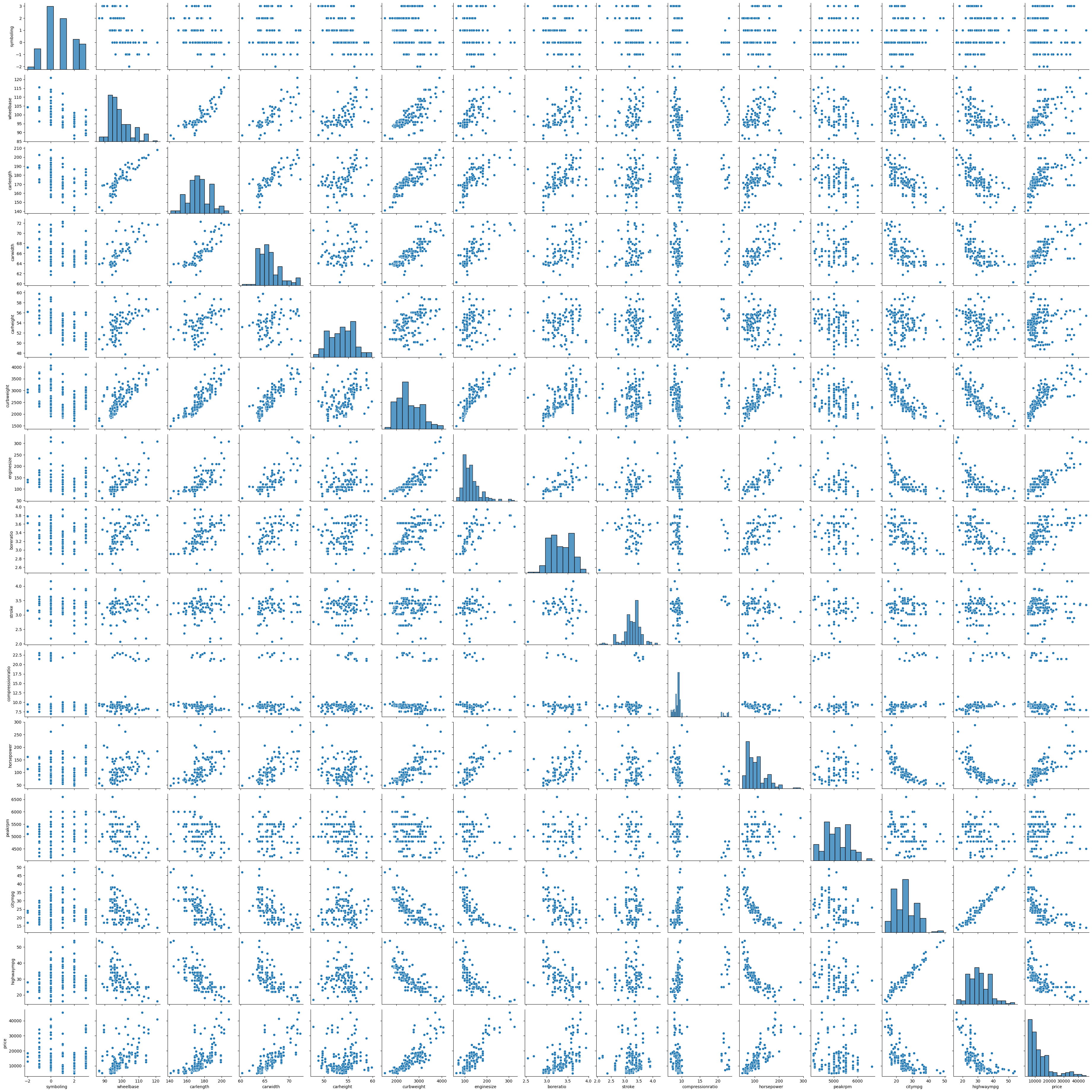


Рисунок 11 – Матрица диаграмм

По данной матрице можно сказать, что большинство столбцов имеют корреляцию друг с другом, присутствуют как прямые зависимости, так и обратные

Далее был изменен столбец doornumber с текстовых значений на числовые, так как в столбце 2-х дверная и 4-хдверная машины были помечены как two и four

set\_option('future.no\_silent\_downcasting', True)

df['doornumber'] = df['doornumber'].replace('two',2)

df['doornumber'] = df['doornumber'].replace('four',4)

После этого все текстовые столбцы были удалены, так как их перевод в числовые значения мог усложнить всю модель и повлиять на результаты.

Датафрейм был поделен на валидационную и на обучающую выборки. Для теста была взята четверть от всех значений

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

df\_numeric = df.select\_dtypes(exclude=['object'])

X = df\_numeric.drop(columns=['price'])

y = df\_numeric['price']

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size= 0.25, random\_state= 42)

Далее все числовые значения были нормализованы и приведены к одной единице измерения. Это необходимо для того, чтобы все признаки одинаково влияли на целевую переменную. Производится нормализация с помощью метода StandardScaler библиотеки sklearn

from sklearn.preprocessing import StandardScaler

scaler = StandardScaler()

X\_train\_scaled = scaler.fit\_transform(X\_train)

X\_test\_scaled = scaler.transform(X\_test)

Далее модель была обучена линейной регрессии на обучающей выборке

model = LinearRegression()

model.fit(X\_train\_scaled, y\_train)

Были рассчитаны метрики качества

predictions = model.predict(X\_test\_scaled)

mse = mean\_squared\_error(y\_test, predictions)

mae = mean\_absolute\_error(y\_test, predictions)

rmse = np.sqrt(mse)

r2 = r2\_score(y\_test, predictions)

a = model.coef\_[1]

b = model.intercept\_

print(f' MSE: {mse}\n MAE: {mae}\n RMSE: {rmse}\n R2: {r2}\n a: {a}\n b: {b}')

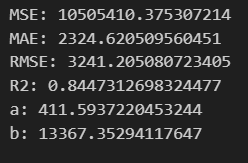


Рисунок 12 – Расчет метрик

Модель показывает достаточно хорошее качество предсказания R2 = 0.8447 и объясняет примерно 85% вариации в данных

Была выведена таблица исходных и предсказанных значений

results\_df2 = pd.DataFrame({"True Values": y\_test, "Predicted Values": predictions})

display(results\_df2)

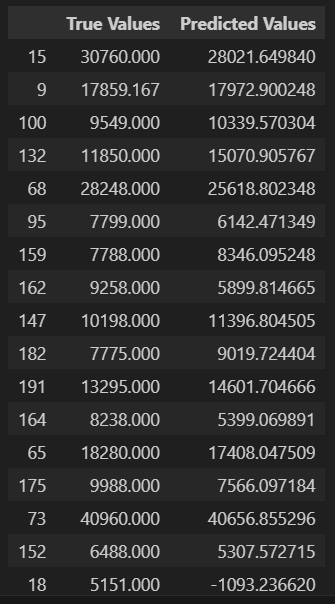


Рисунок 13 – Таблица исходных и предсказанных значений

По данной таблице можно увидеть, что большинство значений предсказано довольно близко к исходным значениям. Однако есть значения, которые сильно отличаются от исходных, например, 41315.000 и 27880.941

Была выведена таблица коэффициентов значимости каждого признака

feature\_importance\_df = pd.DataFrame({'Feature': X.columns, 'Coefficient': model.coef\_})

display(feature\_importance\_df)

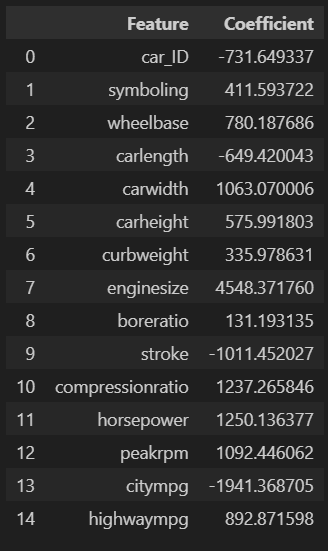


Рисунок 14 – Таблица коэффициентов значимости

По данной таблице можно сказать, что признаки enginesize, carwidth, compressionratio и horsepower оказывают наибольшее влияние на целевую переменную.

Для визуализации результатов был построен график, который сопоставляет исходные и предсказанные значения

plt.figure(figsize=(10, 5))

plt.plot(y\_test.values, label='True Values')

plt.plot(predictions, label='Predicted Values', linestyle='--')

plt.title('Исходные и предсказанные значения')

plt.legend()

plt.show()

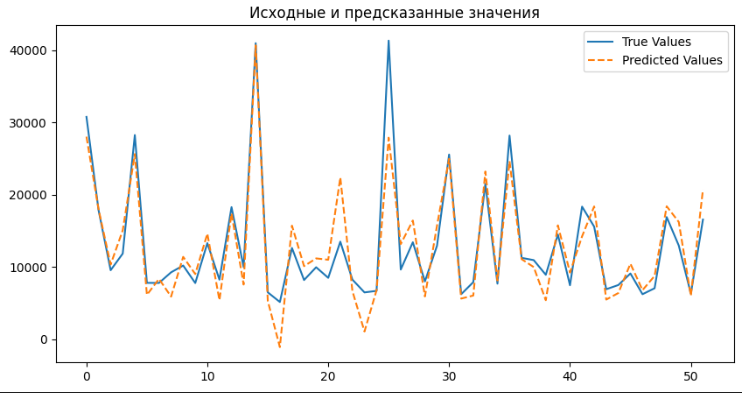


Рисунок 15 – Таблица исходных и предсказанных значений

По этому графику становится более понятно, что где-то модель предсказала значения с хорошей точностью, а где-то ошибалась. Но в большинстве своём модель выдавала значения, близкие к исходным

Далее были созданы модели, которые обучались построению регрессии методами k-ближайших соседей и дерева решений на основе обучающей выборке

from sklearn.neighbors import KNeighborsRegressor

knn\_model = KNeighborsRegressor(n\_neighbors=5)

knn\_model.fit(X\_train\_scaled, y\_train)

knn\_predictions = knn\_model.predict(X\_test\_scaled)

from sklearn.tree import DecisionTreeRegressor

tree\_model = DecisionTreeRegressor(random\_state=42)

tree\_model.fit(X\_train\_scaled, y\_train)

tree\_predictions = tree\_model.predict(X\_test\_scaled)

Далее были посчитаны метрики для обоих метода

mae\_knn = mean\_absolute\_error(y\_test, knn\_predictions),

mse\_knn = mean\_squared\_error(y\_test, knn\_predictions),

rmse\_knn = np.sqrt(mean\_squared\_error(y\_test, knn\_predictions)),

r2\_knn = r2\_score(y\_test, knn\_predictions)

mae\_tree = mean\_absolute\_error(y\_test, tree\_predictions),

mse\_tree = mean\_squared\_error(y\_test, tree\_predictions),

rmse\_tree = np.sqrt(mean\_squared\_error(y\_test, tree\_predictions)),

r2\_tree = r2\_score(y\_test, tree\_predictions)

print(f'K-Nearest Neighbors metrics:\n MSE: {mse\_knn}\n MAE: {mae\_knn}\n RMSE: {rmse\_knn}\n R2: {r2\_knn}')

print(f'Decision Tree:\n MSE: {mse\_tree}\n MAE: {mae\_tree}\n RMSE: {rmse\_tree}\n R2: {r2\_tree}')

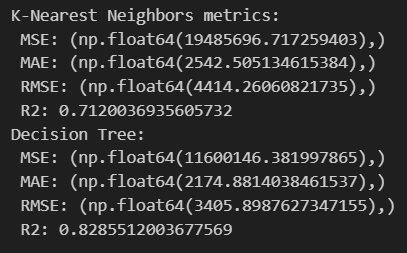


Рисунок 16 – Метрики методов KNN и дерева решений

Исходя из всех метрик, дерево решений показывает лучшие результаты по сравнению с методом K-ближайших соседей. Модель дерева решений имеет меньшие ошибки и более высокое значение R2, что указывает на её лучшую точность и умение предсказать значения

Для сравнения всех полученных результатов был составлен график сравнения исходных значений и предсказанных методами линейной регрессии, k-ближайших соседей и дерева решений

plt.figure(figsize=(12, 6))

plt.plot(y\_test.values, label="True Values", marker='o')

plt.plot(knn\_predictions, label="K-Nearest Neighbors Predictions", linestyle='--', marker='x')

plt.plot(tree\_predictions, label="Decision Tree Predictions", linestyle='--', marker='s')

plt.plot(predictions, label="Linear Regression Predictions", marker='+')

plt.ylabel("Price")

plt.title("Исходные и предсказанные значения по методу k-ближайших соседей и дереву решений")

plt.legend()

plt.show()

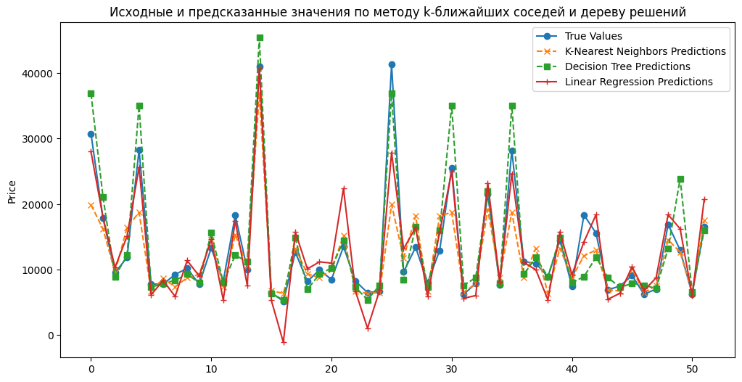


Рисунок 17 – Исходные и предсказанные значения по методам линейной регрессии, KNN и дерева решений

По данному графику можно сделать следующие выводы:

Дерево решений даёт наиболее точные предсказания для этой выборки, поскольку оно лучше улавливает резкие изменения, характерные для данных.

Линейная регрессия показывает худшие результаты для данных с высоким уровнем вариативности, поскольку сглаживает пики и не подходит для задач с выраженной нелинейной зависимостью.

K-ближайшие соседи также сглаживает данные и может быть менее точным, чем дерево решений, для задач с высокой вариативностью, хотя и работает лучше, чем линейная регрессия.

Вывод

В данной лабораторной работе был проведен регрессионный анализ некоторых наборов данных и датасета carprice.csv. В первом задании была обучена модель для проведения линейной регрессии. Также были посчитаны метрики MSE, MAE, RMSE и R2, которые определяют точность построения предсказанных точек, имеющие низкие значения, кроме R2 - это нормализованная метрика, которая составила 0.9189, что показывает хорошую тточность предсказания. Также был построен график разницы предсказанных точек от исходных, максимальная разница составила -2.

Во втором задании была обучена модель для проведения полиномиальной регрессии. Была создана функция, составляющая графики для регрессии 2, 3, 4, и 5 степени. Наиболее точные результаты показала полиномиальная регрессия 5 степени, метрики которой были меньше всех остальных, а R2 была наиболее приближена к единице (0,996).

В третьем задании необходимо было обработать датасет carprice.csv. За целевую переменную был взят столбец price, показывающий цену автомобиля. Были созданы гистограмма и boxplot распределения цен, которые дали понять, что в датасете больше всего машин с низкой или средней ценой. Датасет был разделен на обучающую и валидационные выборки (25% от всех значений). На основе валидационной выборке было проведено обучение линейной регресии. Метрики тестовой выборки имели большие значения, однако R2 составила 0,84, что хоть и позволяет работать с моделью дальше, но присутствует необходимость в доработке. Был составлен график предсказанных и исходных значений, было замечено, что где-то модель выдавала довольно близкие значения, а где-то значения сильно разнились.

Также модель была обучена таким методам регрессии как k-ближайших соседей и дерева решений, и был составлен график сравнения предсказанных значений по этим методам вместе с линейным с исходными значениями. Визуальное сравнение и сравнение всех метрик показало, что для данного случая более точные значения выдавал метод дерева решений