УНИВЕРСИТЕТ ИТМО

Факультет программной инженерии и компьютерной техники

Направление подготовки 09.03.01 Информатика и вычислительная техника

Дисциплина «Технологии нейросетевых вычислений»

Лабораторная работа №1

Вариант 1

Студенты
Мухаметгалеев Д.Т.
P34312
Афанасьев Д.О.
P34312

Преподаватель
Старобыховская А.А.

Санкт-Петербург, 2024 г.

Задание

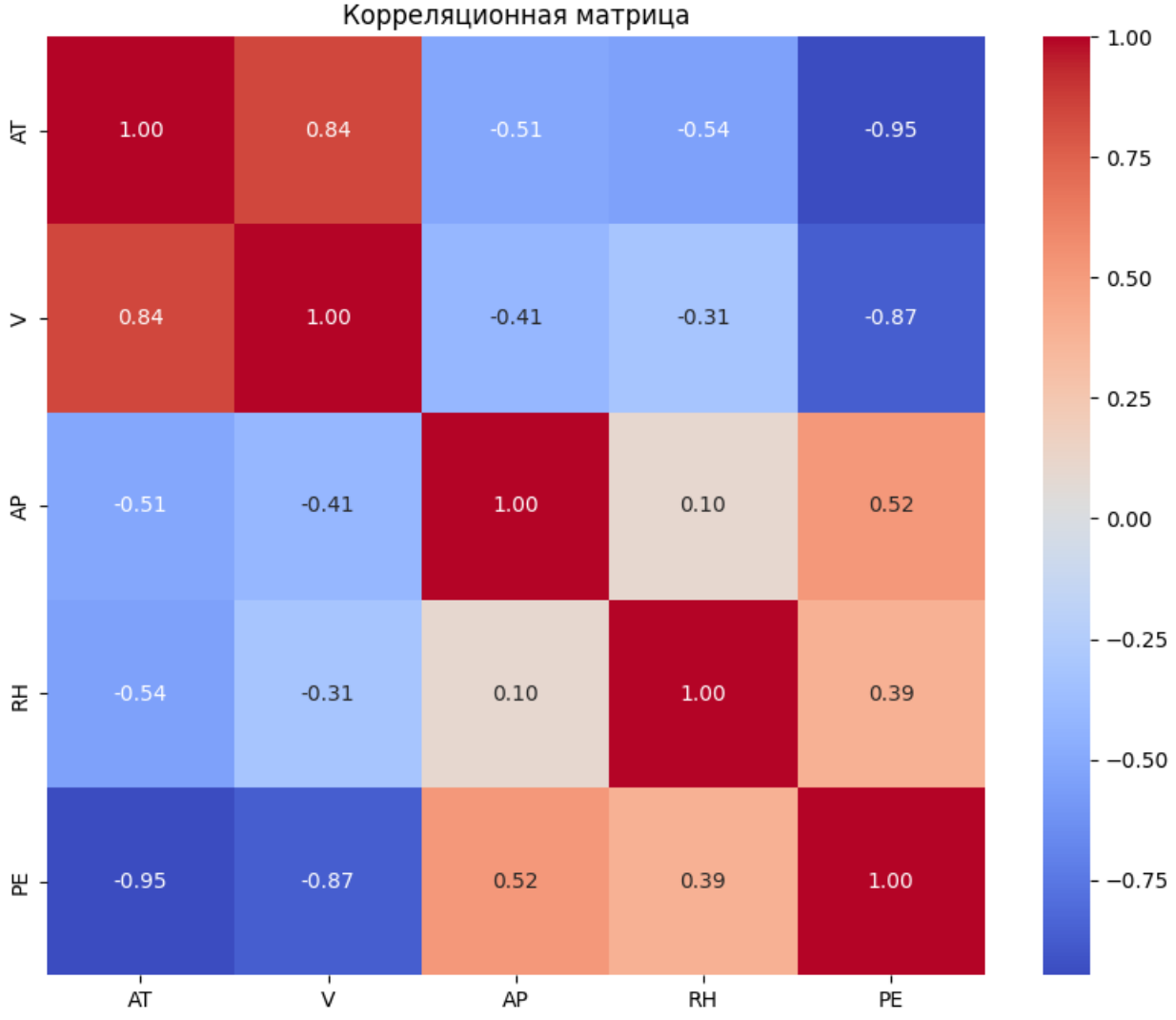
1. выбрать задачу из списка (сообщить об этом)
2. скачать dataset (см. задачу)
3. сделать анализ данных (не менее 8 графиков/расчетов)
4. сделать выводы по результатам исследования и описать потенциальные риски и их решения (не менее 3х)
5. подготовить данные к обучению

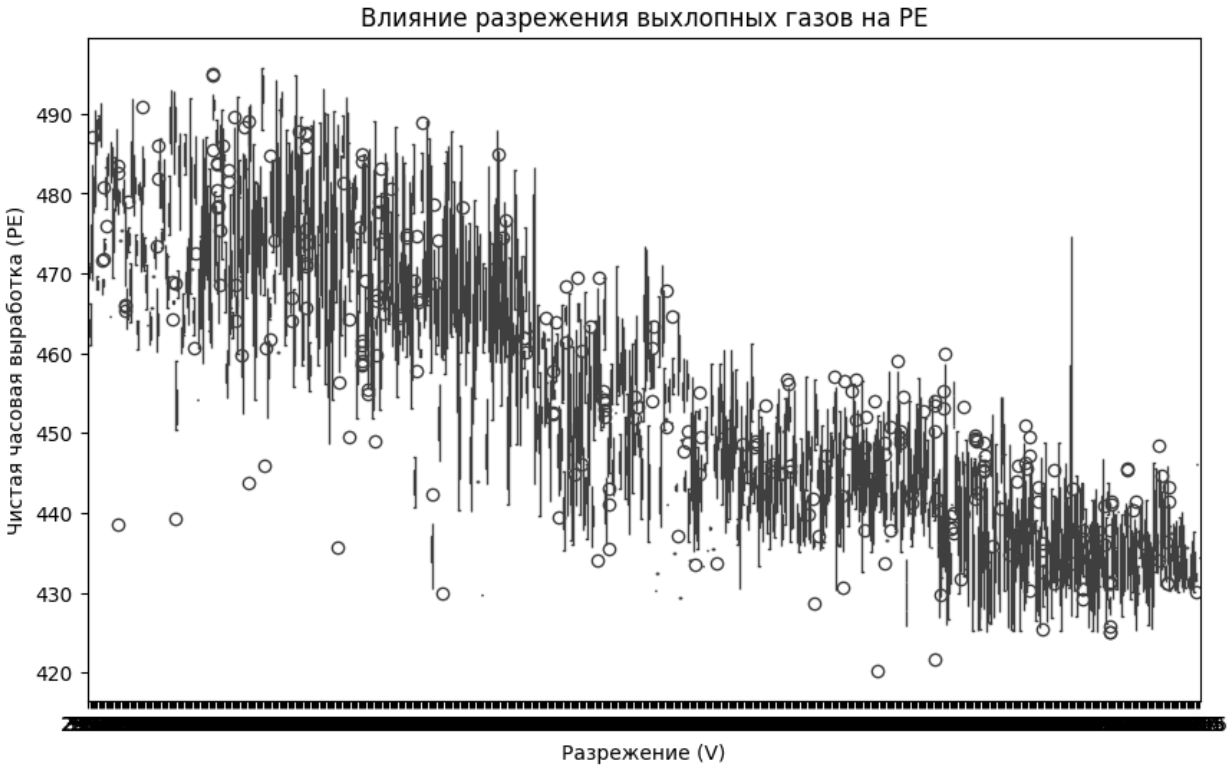
Анализ данных

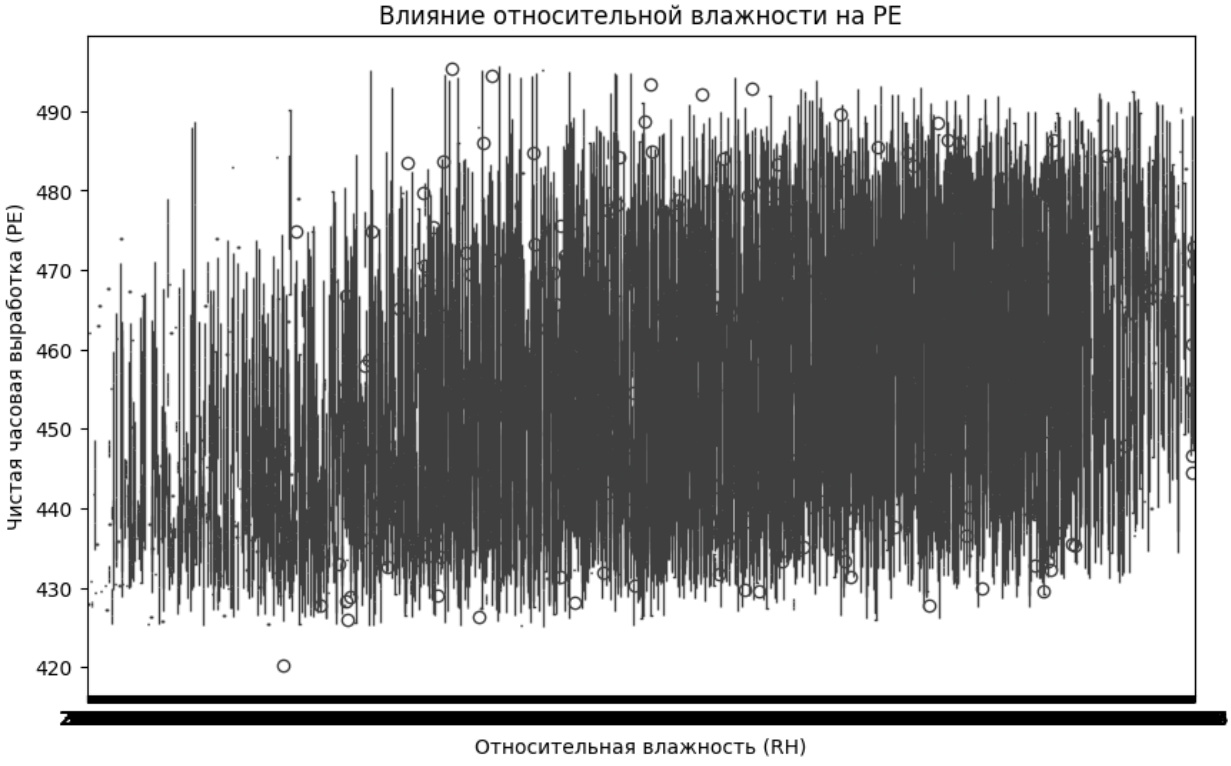
```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

data = pd.read_excel('Folds5x2_pp.ods', engine='odf')
print(data.head())
print(data.describe())

# 1. Гистограмма чистой часовой выработки электроэнергии (PE)
plt.figure(figsize=(10, 6))
sns.histplot(data['PE'], bins=30, kde=True, color='blue')
plt.title('Гистограмма чистой часовой выработки электроэнергии (PE)')
plt.xlabel('Чистая часовая выработка (PE)')
plt.ylabel('Частота')
plt.show()

# 2. Корреляционная матрица
correlation_matrix = data.corr()
plt.figure(figsize=(10, 8))
sns.heatmap(correlation_matrix, annot=True, fmt=".2f", cmap='coolwarm',
square=True)
plt.title('Корреляционная матрица')
plt.show()

# 3. Разброс PE по температуре
plt.figure(figsize=(10, 6))
sns.scatterplot(x='AT', y='PE', data=data, alpha=0.5)
plt.title('Разброс PE по температуре окружающей среды')
plt.xlabel('Температура (T)')
plt.ylabel('Чистая часовая выработка (PE)')
plt.show()

# 4. Разброс PE по давлению
plt.figure(figsize=(10, 6))
sns.scatterplot(x='AP', y='PE', data=data, alpha=0.5)
plt.title('Разброс PE по давлению окружающей среды')
plt.xlabel('Давление (AP)')
plt.ylabel('Чистая часовая выработка (PE)')
plt.show()

# 5. Влияние относительной влажности на PE
plt.figure(figsize=(10, 6))
sns.boxplot(x='RH', y='PE', data=data)
plt.title('Влияние относительной влажности на PE')
plt.xlabel('Относительная влажность (RH)')
plt.ylabel('Чистая часовая выработка (PE)')
plt.show()

# 6. Влияние разрежения на PE
```

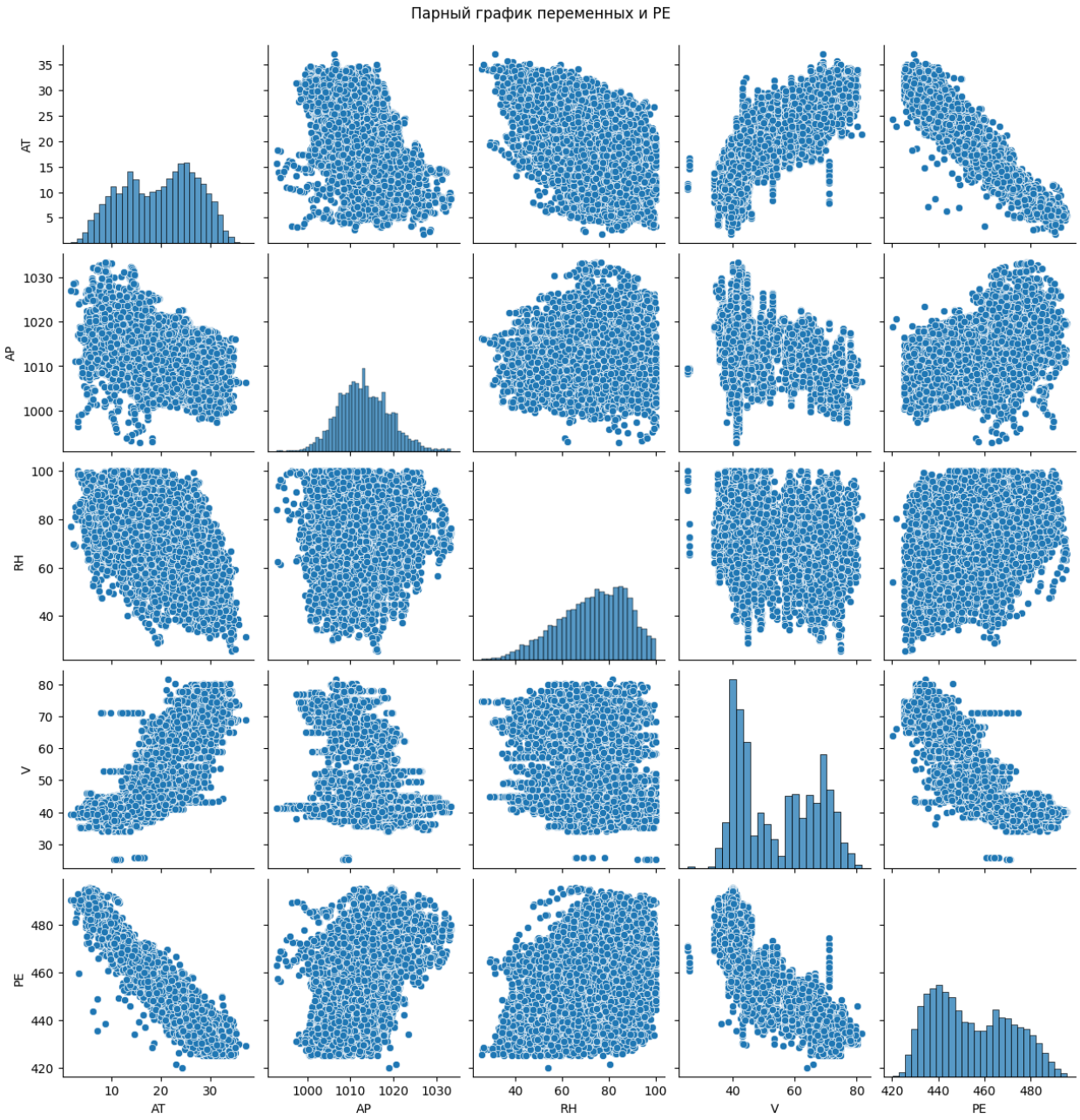
```
plt.figure(figsize=(10, 6))
sns.boxplot(x='V', y='PE', data=data)
plt.title('Влияние разрежения выхлопных газов на PE')
plt.xlabel('Разрежение (V)')
plt.ylabel('Чистая часовая выработка (PE)')
plt.show()

# 7. Средняя выработка электроэнергии по уровням температуры
temperature_bins = pd.cut(data['AT'], bins=10
avg_PE_by_temp = data.groupby(temperature_bins)['PE'].mean().reset_index()
plt.figure(figsize=(10, 6))
sns.barplot(x=avg_PE_by_temp['AT'].astype(str), y=avg_PE_by_temp['PE'],
palette='viridis')
plt.title('Средняя чистая выработка электроэнергии по уровням
температуры')
plt.xlabel('Температурные группы (T)')
plt.ylabel('Средняя чистая выработка (PE)')
plt.xticks(rotation=45)
plt.show()

# 8. Анализ всех переменных
sns.pairplot(data[['AT', 'AP', 'RH', 'V', 'PE']])
plt.suptitle('Парный график переменных и PE', y=1.02)
plt.show()
```


Гистограмма чистой часовой выработки электроэнергии (PE)
Частота
Чистая часовая выработка (PE)
0
100
200
300
400
500
600
420
430
440
450
460
470
480
490


Корреляционная матрица
AT
V
AP
RH
PE
1.00
0.84
-0.51
-0.54
-0.95
0.84
1.00
-0.41
-0.31
-0.87
-0.51
-0.41
1.00
0.10
0.52
-0.54
-0.31
0.10
1.00
0.39
-0.95
-0.87
0.52
0.39
1.00
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75


Влияние разрежения выхлопных газов на PE
Чистая часовая выработка (PE)
490
480
470
460
450
440
430
420
Разрежение (V)


Влияние относительной влажности на PE
Чистая часовая выработка (PE)
490
480
470
460
450
440
430
420
Относительная влажность (RH)



Вывод

- Посмотрев на корреляционную матрицу, можем заметить, что у нас присутствует существенная зависимость, следовательно данные хорошо детерминируют таргет
- Поскольку у нас сильная корреляция и антикорреляция, то простые модели будет работать довольно хорошо

Риски и их решения

1. Скорее всего представлены не все факторы от которых зависит выработка электроэнергии(например количество аварий и их влияние на мощность)
2. Датасет содержит данные только для одного типа электростанций и будет соответственно не актуален для других
3. Возможно, стоит учитывать и временной период этого датасета т.к. улучшение аппаратуры так же влияет на мощность электростанции

Подготовка данных

```
import numpy as np

from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
data = data.dropna()

X = data[['AT', 'AP', 'RH', 'V']]
y = data['PE']


# Нормализация
scaler = StandardScaler()
X = scaler.fit_transform(X)



X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
random_state=42)
```

Задание часть 2

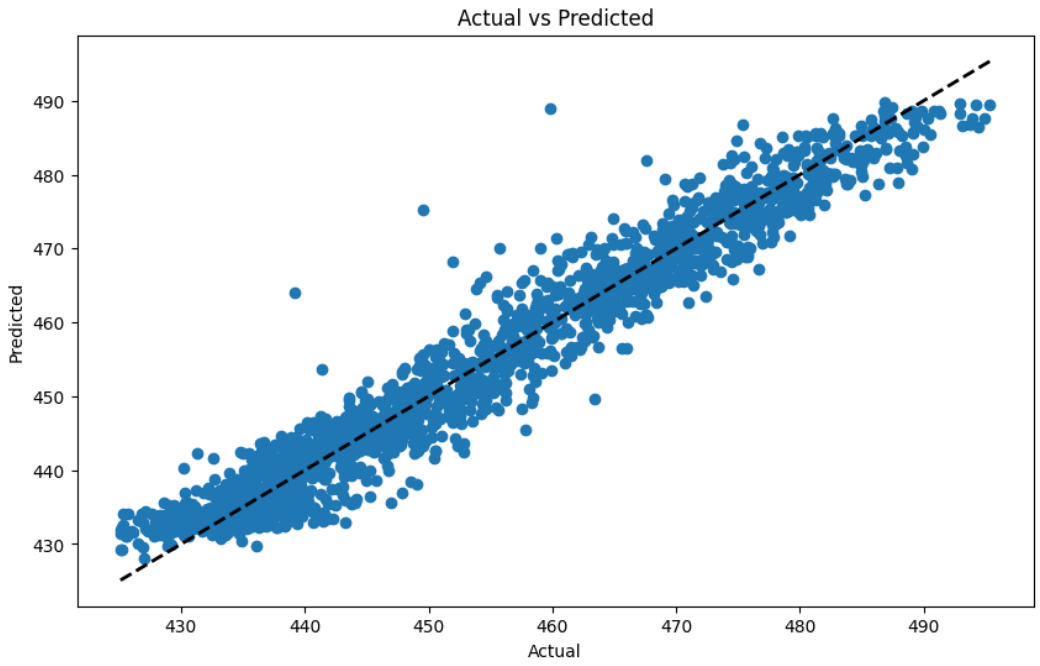
1. Подготовить к обучению
2. Выбрать сеть согласно задаче и обучить ее
    a. Каждый эксперимент должен быть сформулирован как гипотеза
    b. Должны быть добавлено отслеживание кривых и отрисовка графиков с помощью трекера
    c. Должны быть выведены примеры корректных предсказаний и ошибочных
    d. Должны быть рассчитаны метрики и проанализированы полученные результаты
3. Описать выводы + 3 новые гипотезы для повышения качества

Гипотезы

- На основе анализа, мы можем заметить, что данные очень простые и возможно справилась бы и ленейная регрессия
- Изменив базовые параметры мы получим более точные результаты

Обучение

```
from sklearn.model_selection import train_test_split
from sklearn.ensemble import GradientBoostingRegressor
from sklearn.metrics import mean_squared_error, r2_score
import matplotlib.pyplot as plt

# Обучение модели GradientBoostingRegressor
model = GradientBoostingRegressor()
model.fit(X_train, y_train)
# Предсказание на тестовом наборе
y_pred = model.predict(X_test)

# Расчет метрик
mse = mean_squared_error(y_test, y_pred)
r2 = r2_score(y_test, y_pred)

print(f'Mean Squared Error: {mse}')
print(f'R-squared: {r2}')

# Отрисовка графика сравнения предсказанных значений и реальных значений
plt.figure(figsize=(10, 6))
plt.scatter(y_test, y_pred)
plt.plot([y_test.min(), y_test.max()], [y_test.min(), y_test.max()], 'k--
', lw=2)
plt.xlabel('Actual')
plt.ylabel('Predicted')
plt.title('Actual vs Predicted')
plt.show()
```



Mean Squared Error: 14.662147479688228

R-squared: 0.9494509770626904

Линейная регрессия

```
from sklearn.linear_model import LinearRegression

model_lr = LinearRegression()
model_lr.fit(X_train, y_train)
y_pred_lr = model_lr.predict(X_test)
mse_lr = mean_squared_error(y_test, y_pred_lr)
r2_lr = r2_score(y_test, y_pred_lr)

# Сравнение точности моделей
print('Mean Squared Error GradientBoosting:', mse)
print('Mean Squared Error Linear Regression:', mse_lr)
print("GradientBoosting R-squared:", r2)
print("Linear Regression R-squared:", r2_lr)
```

Mean Squared Error GradientBoosting: 14.662147479688228

Mean Squared Error Linear Regression: 20.27370599968744

GradientBoosting R-squared: 0.9494509770626904

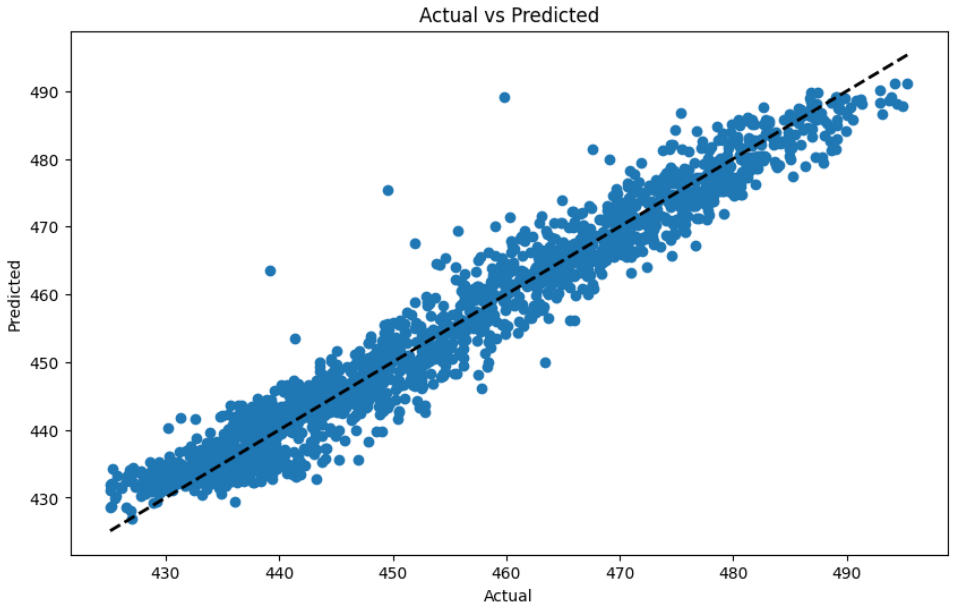
Linear Regression R-squared: 0.9301046431962188

```
# Сравнение предсказаний
results = pd.DataFrame({'Actual': y_test, 'GB_Predicted': y_pred,
'LR_Predicted': y_pred_lr})
print(results.head(10)) # Первые 10 строк для примера

plt.figure(figsize=(12, 6))
plt.plot(y_test - y_pred, label="Gradient Boosting")
plt.plot(y_test - y_pred_lr, label="Linear Regression")
plt.xlabel("Sample")
plt.ylabel("Error")
plt.title("Error Analysis")
plt.legend()
plt.show()
```

	Actual	GB_Predicted	LR_Predicted
2513	455.27	455.226126	455.680208
9411	436.31	436.476871	438.732122
8745	440.68	432.913263	434.164440
9085	434.40	435.880885	438.769546
4950	482.06	478.722749	479.888329
2755	436.07	437.412485	439.499474
563	452.48	448.609675	448.840102
5834	435.22	434.345794	434.809190
6850	432.93	434.150251	435.097364
4359	466.46	472.186991	472.947629

Tuned model

```
# Обучение модели GradientBoostingRegressor
tuned_model = GradientBoostingRegressor(n_estimators=150,
learning_rate=0.1, max_depth=3, min_samples_split=2, min_samples_leaf=1,
random_state=42)
tuned_model.fit(X_train, y_train)

# Предсказание на тестовом наборе
tuned_model_y_pred = tuned_model.predict(X_test)

# Расчет метрик
tuned_model_y_pred_mse = mean_squared_error(y_test, tuned_model_y_pred)
tuned_model_y_pred_r2 = r2_score(y_test, tuned_model_y_pred)

print(f'Mean Squared Error: {tuned_model_y_pred_mse}')
print(f'R-squared: {tuned_model_y_pred_r2}')


# Отрисовка графика сравнения предсказанных значений и реальных значений
plt.figure(figsize=(10, 6))
plt.scatter(y_test, tuned_model_y_pred)
plt.plot([y_test.min(), y_test.max()], [y_test.min(), y_test.max()], 'k--
', lw=2)
plt.xlabel('Actual')
plt.ylabel('Predicted')
plt.title('Actual vs Predicted')
plt.show()
```



Mean Squared Error: 13.732583757617288

R-squared: 0.9526557284794763