МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ ІМЕНІ ІГОРЯСІКОРСЬКОГО»
Факультет інформатики та обчислювальної техніки Кафедра
інформатики та програмної інженерії

Практикум №5

з курсу «Аналіз даних в інформаційнних системах» на тему:

«Регресивні моделі»

Викладач:
Ліхоузова Т.А.

Виконав:
студент 2 курсу групи
ІП-15 ФІОТ
Мєшков Андрій
Ігорович

Київ-2023

Практикум №5

Регресивні моделі

Мета роботи: ознайомитись з різновидами регресійних моделей.

Завдання:

Скачати потрібні дані.

Основне завдання

Завантажити дані про якість червоного вина

1. Дослідити дані, підготувати їх для побудови регресійної моделі
2. Розділити дані на навчальну та тестову вибірки
3. Побудувати декілька регресійних моделей для прогнозу якості вина (12 - quality). Використати лінійну одномірну та багатомірну регресію та поліноміальну регресію обраного вами виду (3-5 моделей)
4. Використовуючи тестову вибірку, з'ясувати яка з моделей краща

Додаткове завдання

Завантажити дані файлу Data4.csv

1. Дослідити дані, сказати чи є мультиколінеарність, побудувати діаграми розсіювання
2. Побудувати декілька регресійних моделей (використати лінійну регресію та поліноміальну регресію обраного вами виду)
3. Використовуючи тестову вибірку з файлу Data4t.csv, з'ясувати яка з моделей краща

Хід роботи:

Основне завдання:

Імпортуємо потрібні бібліотеки.

```
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error, r2_score
from sklearn.preprocessing import PolynomialFeatures
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline
```

Зчитаємо файл.

```
df = pd.read_csv('winequality-red.csv', sep=',', encoding='cp1252')
```

Проаналізуємо структуру.

```
df.info()
df.head()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1599 entries, 0 to 1598
Data columns (total 12 columns):
 #   Column                Non-Null Count  Dtype
---  ------                --------------  -----
 0   fixed acidity         1599 non-null   float64
 1   volatile acidity      1599 non-null   float64
 2   citric acid           1599 non-null   float64
 3   residual sugar        1599 non-null   float64
 4   chlorides             1599 non-null   float64
 5   free sulfur dioxide   1599 non-null   float64
 6   total sulfur dioxide  1599 non-null   float64
 7   density               1599 non-null   float64
 8   pH                    1599 non-null   float64
 9   sulphates             1599 non-null   float64
 10  alcohol               1599 non-null   float64
 11  quality               1599 non-null   int64
dtypes: float64(11), int64(1)
memory usage: 150.0 KB
```

	fixed acidity	volatile acidity	citric acid	residual sugar	chlorides	free sulfur dioxide	total sulfur dioxide	density	pH	sulphates	alcohol	quality
0	7.4	0.70	0.00	1.9	0.076	11.0	34.0	0.9978	3.51	0.56	9.4	5
1	7.8	0.88	0.00	2.6	0.098	25.0	67.0	0.9968	3.20	0.68	9.8	5
2	7.8	0.76	0.04	2.3	0.092	15.0	54.0	0.9970	3.26	0.65	9.8	5
3	11.2	0.28	0.56	1.9	0.075	17.0	60.0	0.9980	3.16	0.58	9.8	6
4	7.4	0.70	0.00	1.9	0.076	11.0	34.0	0.9978	3.51	0.56	9.4	5

Розділіть дані на навчальну та тестову вибірки.

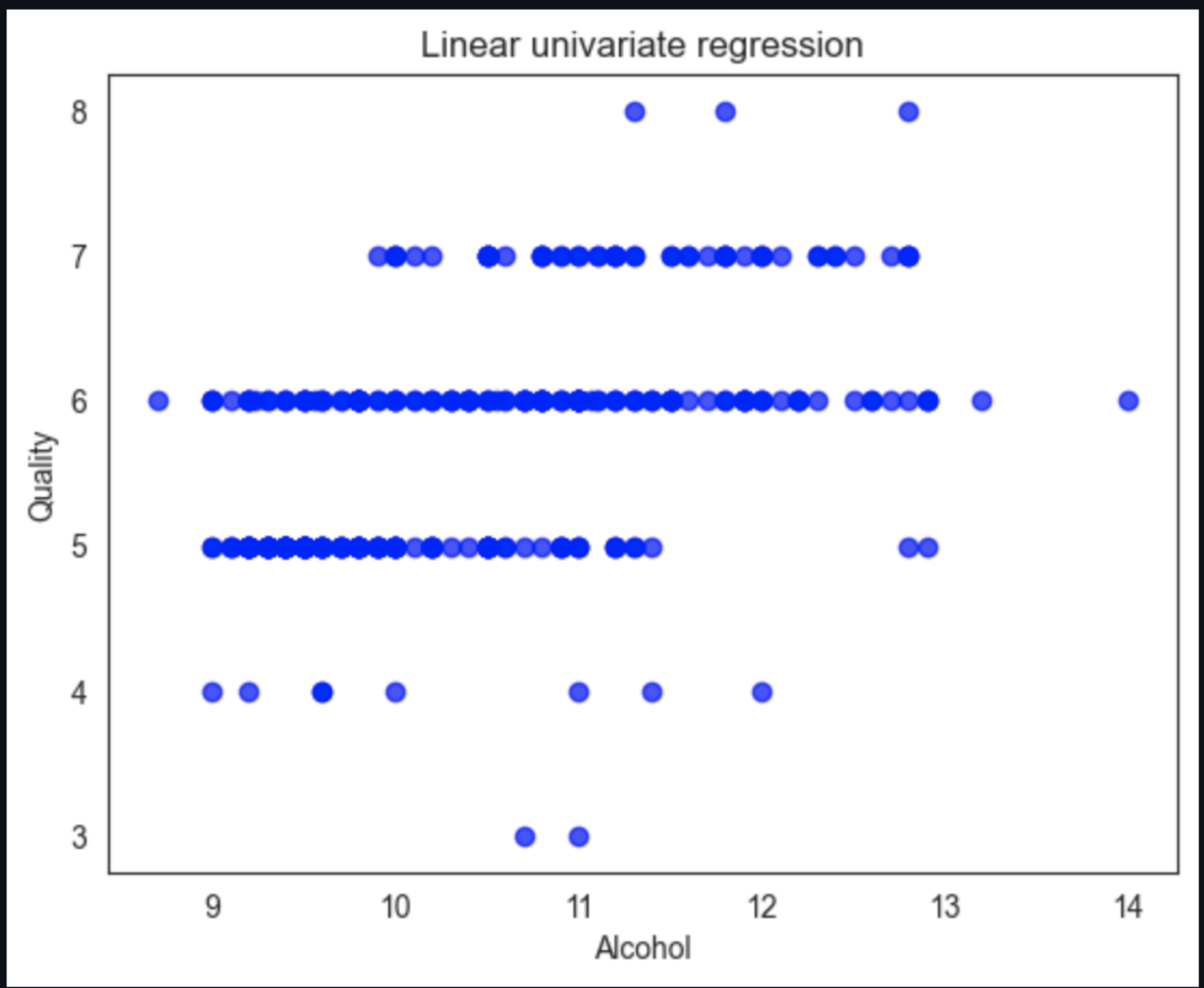
```
X = df.drop(['quality'], axis=1)
Y = df['quality']

X_train, X_test, y_train, y_test = train_test_split(X, Y, test_size=0.2)
```

Побудуємо моделі

```
# Linear univariate regression
model = LinearRegression()
# Навчання моделі
model.fit(X_train.iloc[:, -1:], Y_train)
# Прогнозування для тестової вибірки
y_pred = model.predict(X_test.iloc[:, -1:])
```

```
mse = mean_squared_error(Y_test, y_pred)
accuracy = model.score(X_test.iloc[:, -1:], Y_test)
print("Linear univariate regression:")
print("MSE = {:.4f}".format(mse), "\nAccuracy = {:.4f}".format(accuracy))

plt.xlabel('Alcohol')
plt.ylabel('Quality')
plt.title('Linear univariate regression')

plt.scatter(X_test.iloc[:, -1:], Y_test, alpha=0.7, color='blue')
plt.show()
```

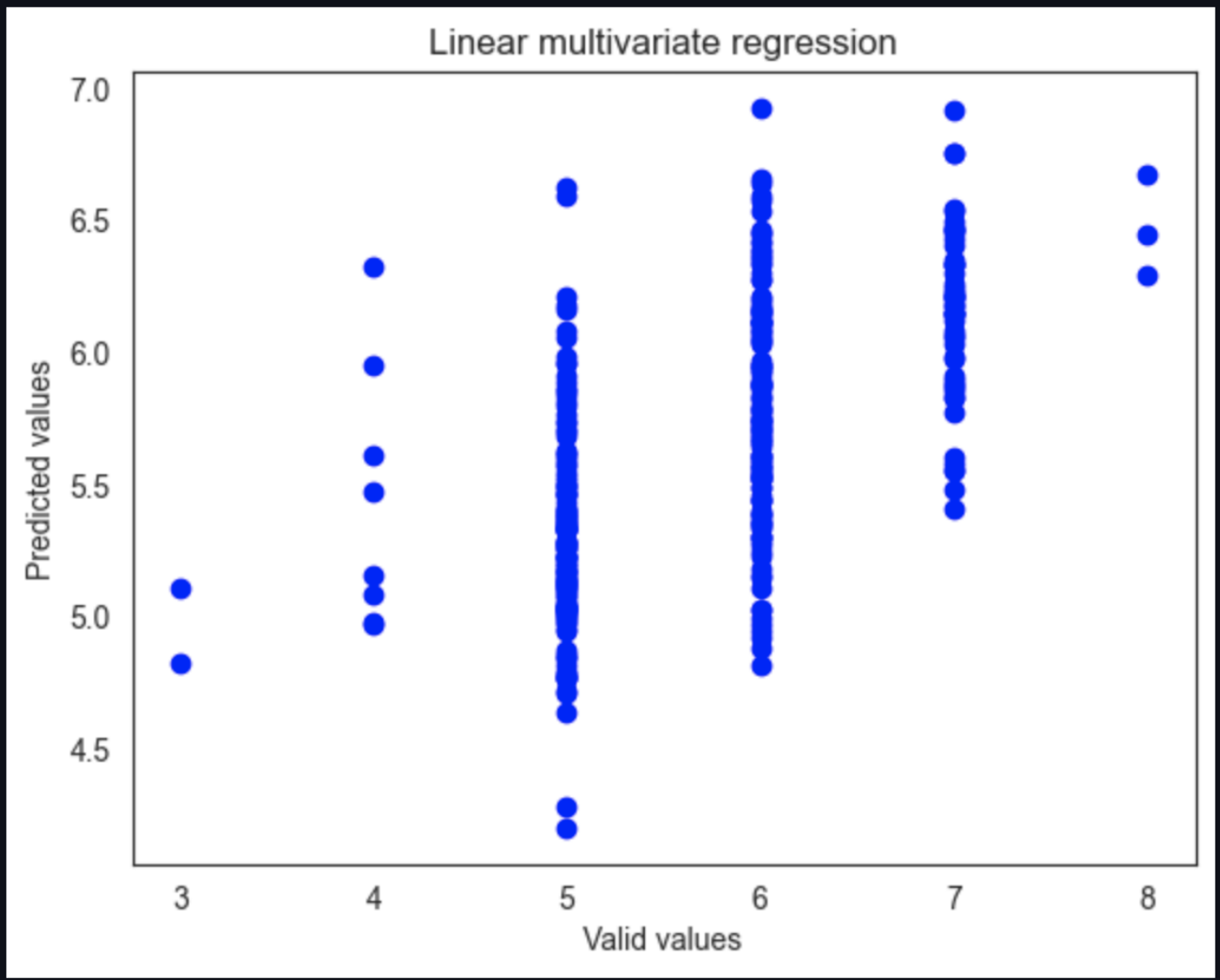
```
Linear univariate regression:
MSE = 0.5173
Accuracy = 0.2249
```




```
# Linear multivariate regression
model = LinearRegression()
# Навчання моделі
model.fit(X_train, Y_train)
```

```
# Прогнозування для тестової вибірки
y_pred = model.predict(X_test)

mse = mean_squared_error(Y_test, y_pred)
accuracy = model.score(X_test, Y_test)
print("Linear multivariate regression:")
print("MSE = {:.4f}".format(mse), "\nAccuracy = {:.4f}".format(accuracy))

plt.scatter(Y_test, y_pred, color='blue')
plt.xlabel('Valid values')
plt.ylabel('Predicted values')
plt.title('Linear multivariate regression')
plt.show()
```

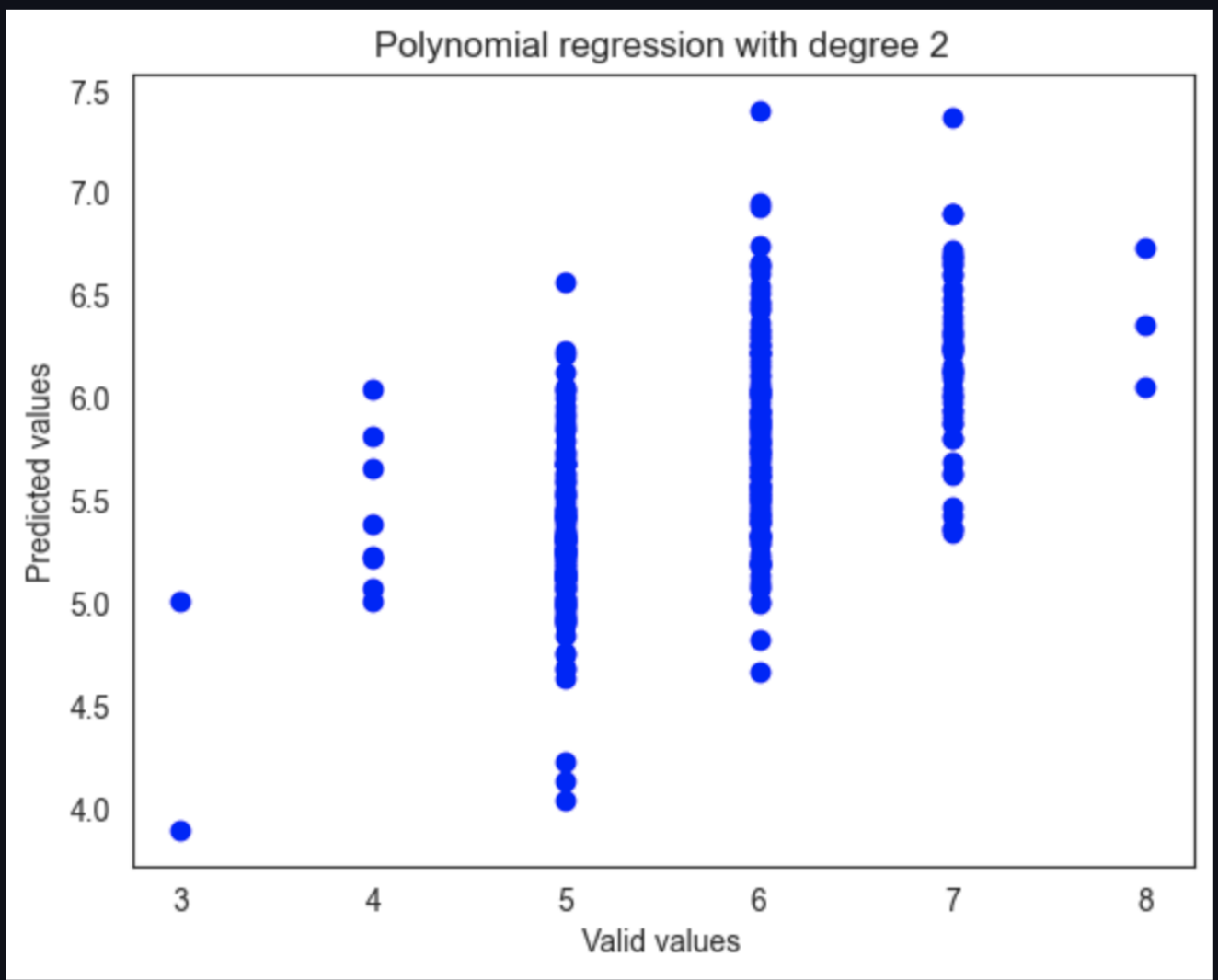
```
Linear multivariate regression:
MSE = 0.4384
Accuracy = 0.3431
```




```
# Polynomial regression with degree 2
poly = PolynomialFeatures(degree=2)

X_train_poly = poly.fit_transform(X_train)
X_test_poly = poly.transform(X_test)
```

```
model_poly = LinearRegression()
model_poly.fit(X_train_poly, Y_train)

y_pred = model_poly.predict(X_test_poly)

mse = mean_squared_error(Y_test, y_pred)
accuracy = model_poly.score(X_test_poly, Y_test)
print("Polynomial regression with degree 2:")
print("MSE = {:.4f}".format(mse), "\nAccuracy = {:.4f}".format(accuracy))

plt.scatter(Y_test, y_pred, color='blue')
plt.xlabel('Valid values')
plt.ylabel('Predicted values')
plt.title('Polynomial regression with degree 2')
plt.show()
```

```
Polynomial regression with degree 2:
MSE = 0.4390
Accuracy = 0.3421
```



Додаткове завдання:

Імпортуємо потрібні бібліотеки.

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.linear_model import LinearRegression
from sklearn.preprocessing import PolynomialFeatures
from sklearn.pipeline import make_pipeline
```

Завантажемо файл.

```
df = pd.read_csv('Data4.csv', sep=';', decimal=',', encoding='windows-1251').rename(columns={'Unnamed: 0': 'Country'})
```

Дослідимо дані

```
df.info()
print(df.head())
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 132 entries, 0 to 131
Data columns (total 7 columns):
 #   Column   Non-Null Count  Dtype  
---  ------   --------------  -----  
 0   Country  132 non-null    object 
 1   ISO      132 non-null    object 
 2   UA       132 non-null    object 
 3   Cql      132 non-null    float64
 4   Ie       132 non-null    float64
 5   Iec      132 non-null    float64
 6   Is       132 non-null    float64
dtypes: float64(4), object(3)
memory usage: 7.3+ KB
     Country  ISO         UA       Cql        Ie       Iec        Is
0    Albania  ALB    Албанія  0.973924  0.605348  0.538673  0.510113
1    Algeria  DZA      Алжир  0.782134  0.587219  0.348159  0.497986
2     Angola  AGO     Ангола  0.372344  0.274394  0.332117  0.346907
3  Argentina  ARG  Аргентина  0.883830  0.699685  0.281995  0.518820
4    Armenia  ARM   Вірменія  1.016499  0.718327  0.535648  0.486498
```

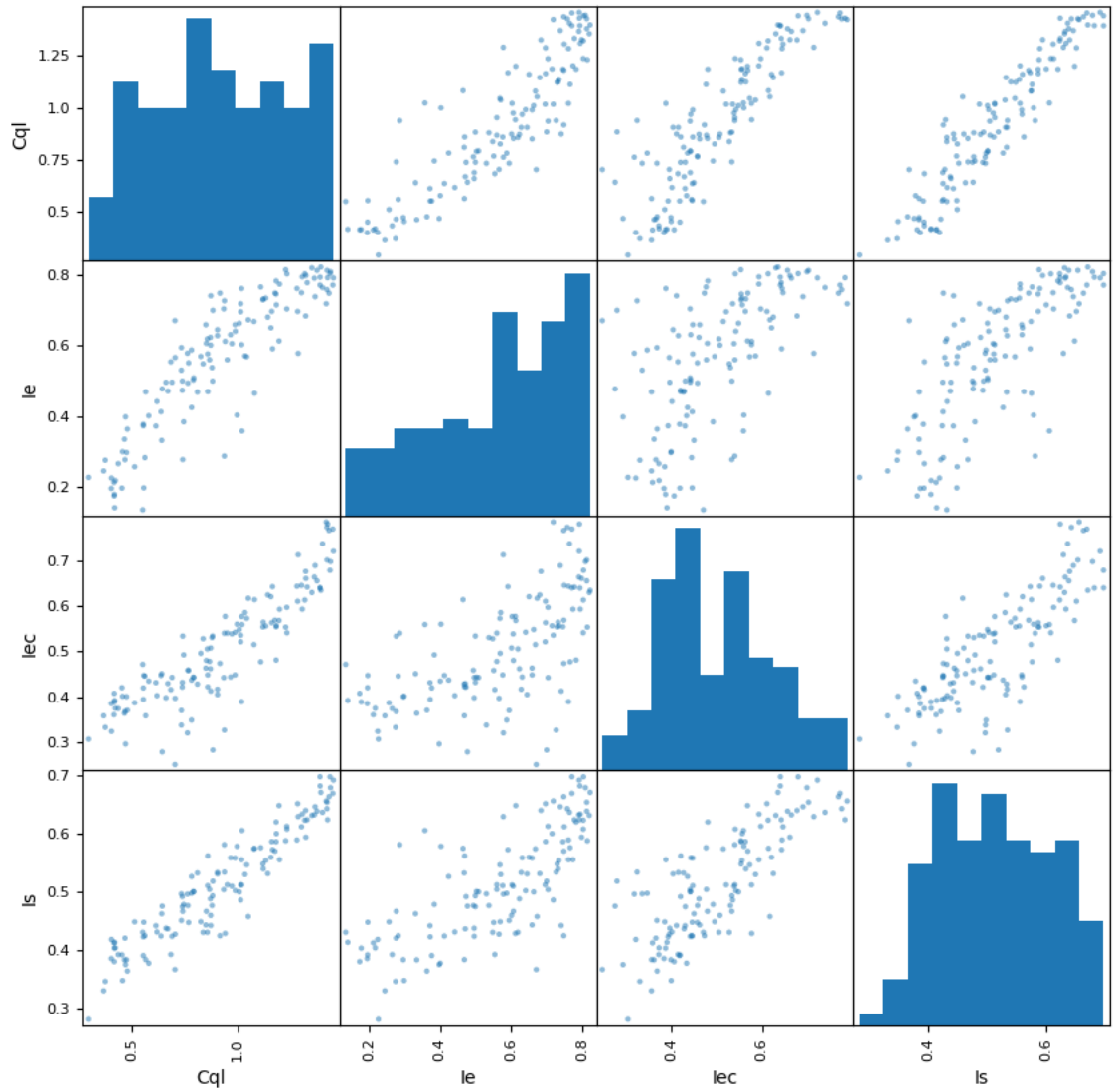
Перевіремо мультиколінеарність

```
df.corr()
```

	Cql	Ie	Iec	Is
Cql	1.000000	0.883664	0.875545	0.939172
Ie	0.883664	1.000000	0.619247	0.746320
Iec	0.875545	0.619247	1.000000	0.799211
Is	0.939172	0.746320	0.799211	1.000000

Побудуємо діаграми розсіювання.

```
pd.plotting.scatter_matrix(df, figsize=(10, 10))
plt.show()
```

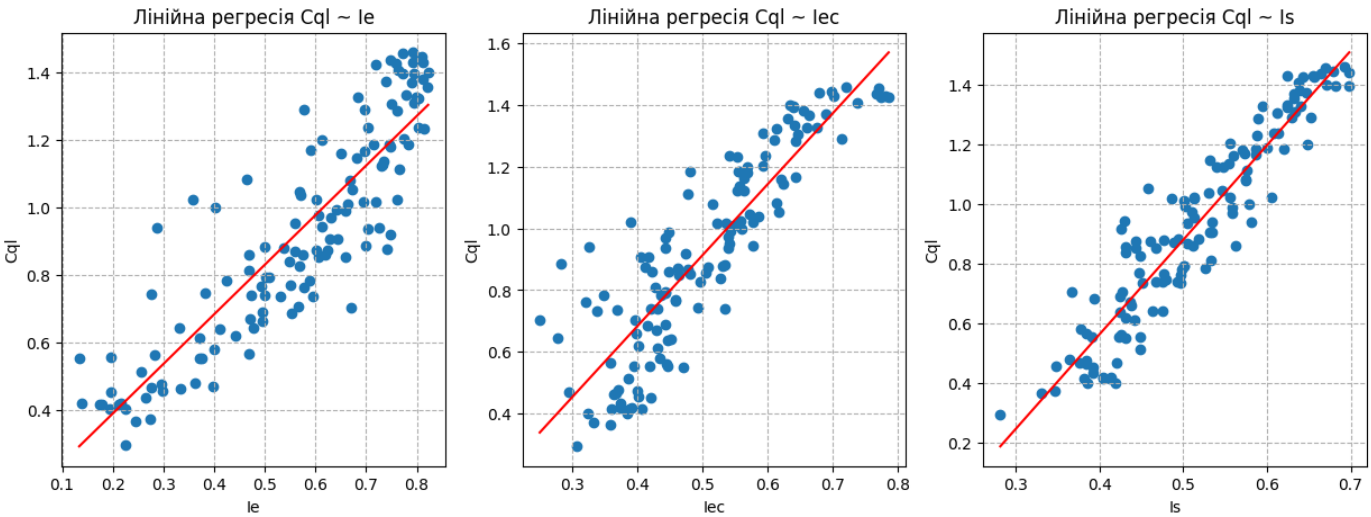


Побудуємо лінійні моделі залежно від Cql.

```
Y = df['Cql']

l_model1 = LinearRegression().fit(df[['Ie', 'Iec', 'Is']], Y)
l_model2 = LinearRegression().fit(df[['Iec', 'Is']], Y)
l_model3 = LinearRegression().fit(df[['Ie', 'Is']], Y)
l_model4 = LinearRegression().fit(df[['Ie', 'Iec']], Y)
l_model5 = LinearRegression().fit(df['Ie'].to_numpy().reshape(-1, 1), Y)
l_model6 = LinearRegression().fit(df['Iec'].to_numpy().reshape(-1, 1), Y)
l_model7 = LinearRegression().fit(df['Is'].to_numpy().reshape(-1, 1), Y)
```

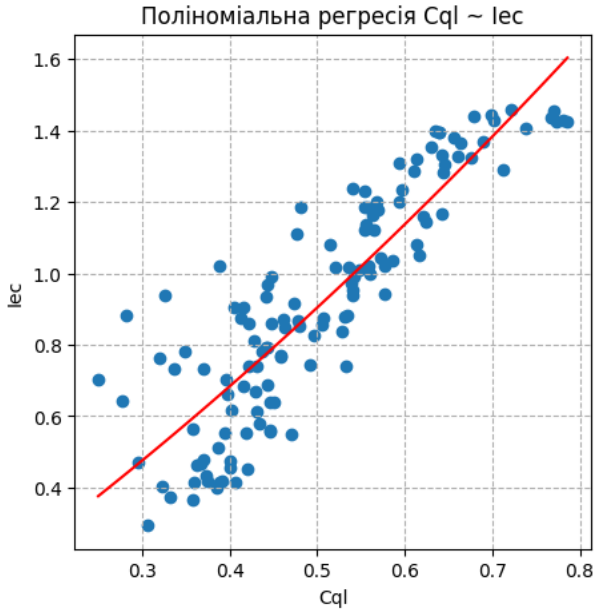
Побудуємо поліноміальні моделі
```
p_model1 = make_pipeline(PolynomialFeatures(degree=2), LinearRegression())
p_model1.fit(df[['Ie', 'Iec', 'Is']], Y)

p_model2 = make_pipeline(PolynomialFeatures(degree=2), LinearRegression())
p_model2.fit(df[['Iec', 'Is']], Y)

p_model3 = make_pipeline(PolynomialFeatures(degree=2), LinearRegression())
p_model3.fit(df[['Ie', 'Iec']], Y)

p_model4 = make_pipeline(PolynomialFeatures(degree=2), LinearRegression())
p_model4.fit(df['Iec'].to_numpy().reshape(-1, 1), Y)
```

Візуалізуємо однопредикторні лінійні моделі
```
linear_models = [l_model5, l_model6, l_model7]
params = ['Ie', 'Iec', 'Is']
params_values = []
y_pred = []

for i in range(len(params)):
    values = np.linspace(df[params[i]].min(), df[params[i]].max()).reshape(-1, 1)
    params_values.append(values)
    y_pred.append(linear_models[i].predict(values))
fig, axes = plt.subplots(1, 3, figsize=(15, 5))
for i in range(len(axes)):
    axes[i].set_title(f'Лінійна регресія Cql ~ {params[i]}')
    axes[i].set_xlabel(params[i])
    axes[i].set_ylabel('Cql')
    axes[i].grid(linestyle='--')
    axes[i].scatter(df[params[i]], Y)
    axes[i].plot(params_values[i], y_pred[i], color='red')
```



Візуалізуємо поліноміальну модель з одною змінною
```
X_pol_test = np.linspace(df['Iec'].min(), df['Iec'].max()).reshape(-1, 1)
Y_pol_pred = p_model4.predict(X_pol_test)

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

plt.title('Поліноміальна регресія Cql ~ Iec')
```

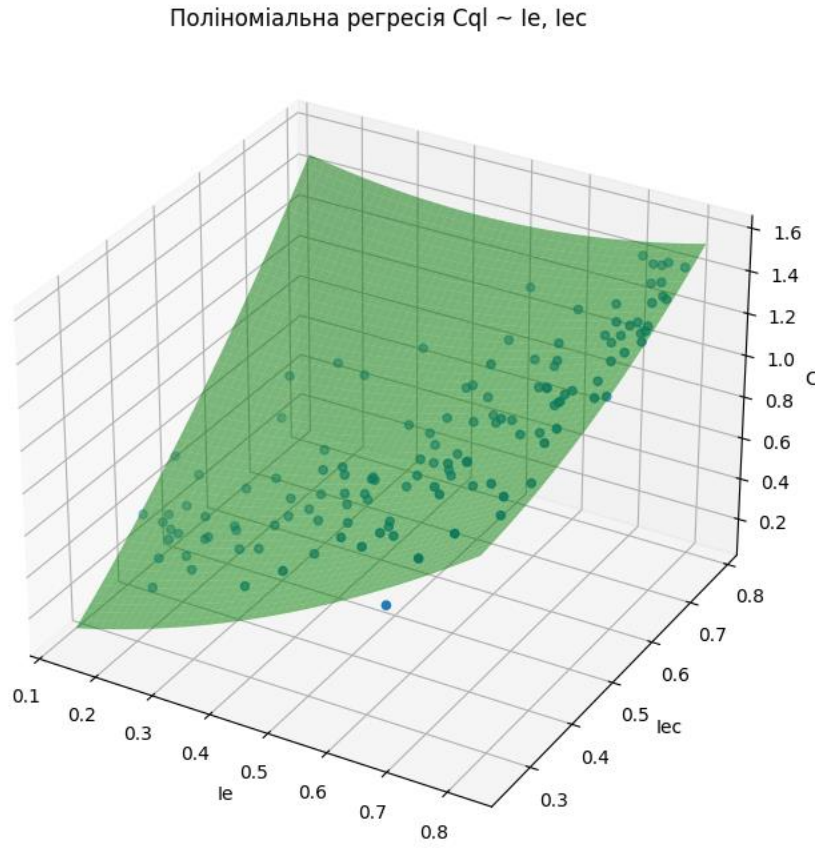
```
plt.xlabel('Cql')
plt.ylabel('Iec')
plt.grid(linestyle='--')

plt.scatter(df['Iec'], Y)
plt.plot(X_pol_test, Y_pol_pred, color='red')

plt.show()
```



Візуалізуємо поліноміальну модель з двома змінними

```
%matplotlib inline

fig = plt.figure(figsize=(8, 8))
ax = fig.add_subplot(111, projection='3d')

X_3d = params_values[0]
Y_3d = params_values[1]

XX, YY = np.meshgrid(X_3d, Y_3d)

Z = []
for i in range(len(Y_3d)):
    temp = []
    for j in range(len(X_3d)):
        temp.append(p_model3.predict(np.array([X_3d[j], Y_3d[i]]).T)[0])
    Z.append(temp)

Z = np.array(Z)

ax.set_title('Поліноміальна регресія Cql ~ Ie, Iec')
ax.set_xlabel('Ie')
ax.set_ylabel('Iec')
ax.set_zlabel('Cql')
ax.plot_surface(
    XX, YY,
    np.array(Z),
    color='green',
```

```
    alpha=0.5
)
ax.scatter(df['Ie'], df['Iec'], Y)

plt.show()
```



Зчитаємо тестові дані

```
df_test = pd.read_csv('Data4t.csv', encoding='windows-1251', sep=';',
decimal=',').rename(columns={'Unnamed: 0': 'Country'})

df_test
```

	Country	ISO	UA	Cql	Ie	Iec	Is
0	Togo	TGO	Того	0.453498	0.216806	0.368235	0.433951
1	Tunisia	TUN	Туніс	0.899462	0.659124	0.418256	0.514746
2	Turkey	TUR	Туреччина	0.859284	0.498840	0.509228	0.499453
3	Uganda	UGA	Уганда	0.571284	0.362946	0.448732	0.375726
4	Ukraine	UKR	Україна	0.802204	0.689164	0.303555	0.462744

Розраховуємо виходи кожної з моделей для тестового набору

```
test_predictions = []
names = ['l_model1' ,'l_model2' ,'l_model3' ,'l_model4' ,'l_model5' ,'l_model6' ,'l_model7' , 'p_model1',
'p_mode2', 'p_model3', 'p_model4']

test_predictions.append(l_model1.predict(df_test[['Ie', 'Iec', 'Is']]))
test_predictions.append(l_model2.predict(df_test[['Iec', 'Is']]))
test_predictions.append(l_model3.predict(df_test[['Ie', 'Is']]))
```

```
test_predictions.append(l_model4.predict(df_test[['Ie', 'Iec']]))
test_predictions.append(l_model5.predict(df_test['Ie'].to_numpy().reshape(-1, 1)))
test_predictions.append(l_model6.predict(df_test['Iec'].to_numpy().reshape(-1, 1)))
test_predictions.append(l_model7.predict(df_test['Is'].to_numpy().reshape(-1, 1)))
test_predictions.append(p_model1.predict(df_test[['Ie', 'Iec', 'Is']]))
test_predictions.append(p_model2.predict(df_test[['Iec', 'Is']]))
test_predictions.append(p_model3.predict(df_test[['Ie', 'Iec']]))
test_predictions.append(p_model4.predict(df_test['Iec'].to_numpy().reshape(-1, 1)))

test_predictions = np.array(test_predictions)
```

З'ясуємо, яка модель має найменше відхилення від справжніх значень виходів для тестової вибірки

```
best = np.sum((test_predictions - df_test['Cql'].to_numpy()) ** 2, axis=1).argmin()

print(f'The best solusion is {names[best]}')
```

```
The best solusion is p_model1
```

Висновок

За отриманими даними можна зробити висновок, що

- В основному завданні найкращою моделлю виявилась лінійна багатовимірна регресія з найменшим MSE.
- В додатковому завданні найкращою моделлю виявилась поліноміальна з трьома змінними з найменшим відхиленням від справжніх значень виходу для тестової вибірки.