МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ

«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ ІМЕНІ ІГОРЯ СІКОРСЬКОГО»

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.A.

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](https://www.kaggle.com/uciml/red-wine-quality-cortez-et-al-2009)

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

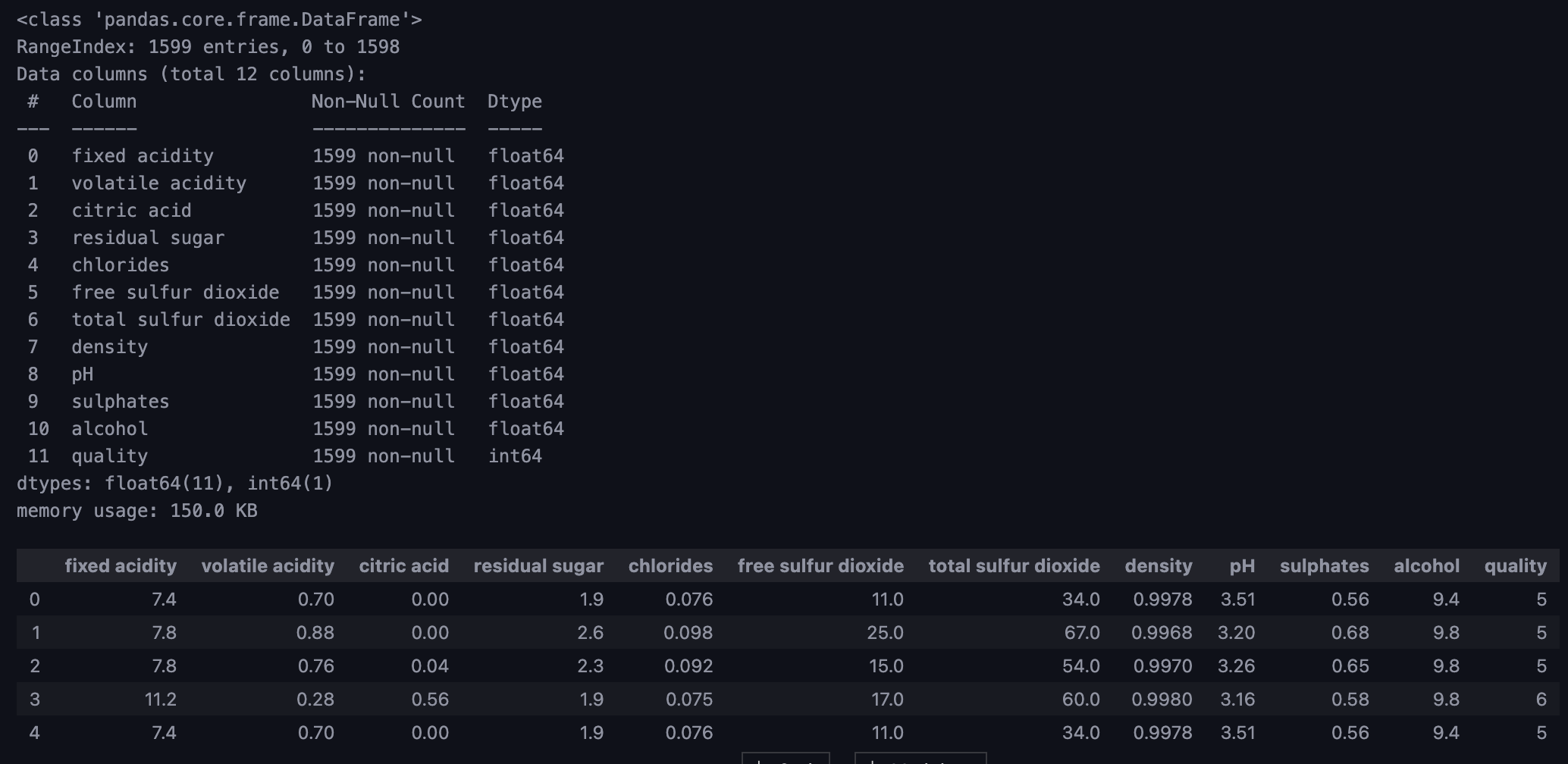
*Зчитаємо файл.*

df = pd.read\_csv('winequality-red.csv', sep=',', encoding='cp1252')

*Проаналізуємо структуру.*

df.info()

df.head()



*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))

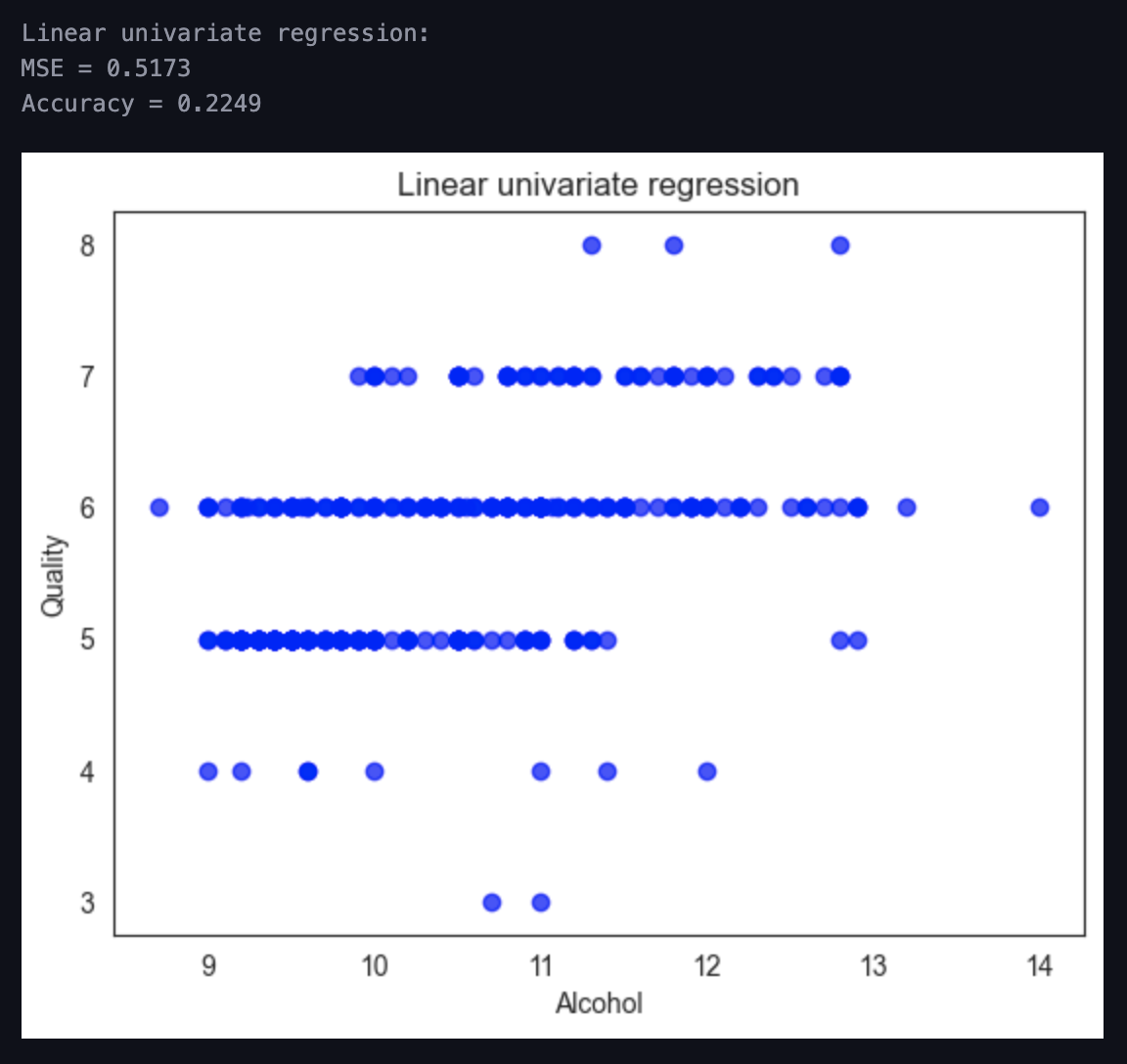
plt.xlabel('Alcohol')

plt.ylabel('Quality')

plt.title('Linear univariate regression')

plt.scatter(X\_test.iloc[:, -1:], Y\_test, alpha=0.7, color='blue')

plt.show()



*# Linear 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))

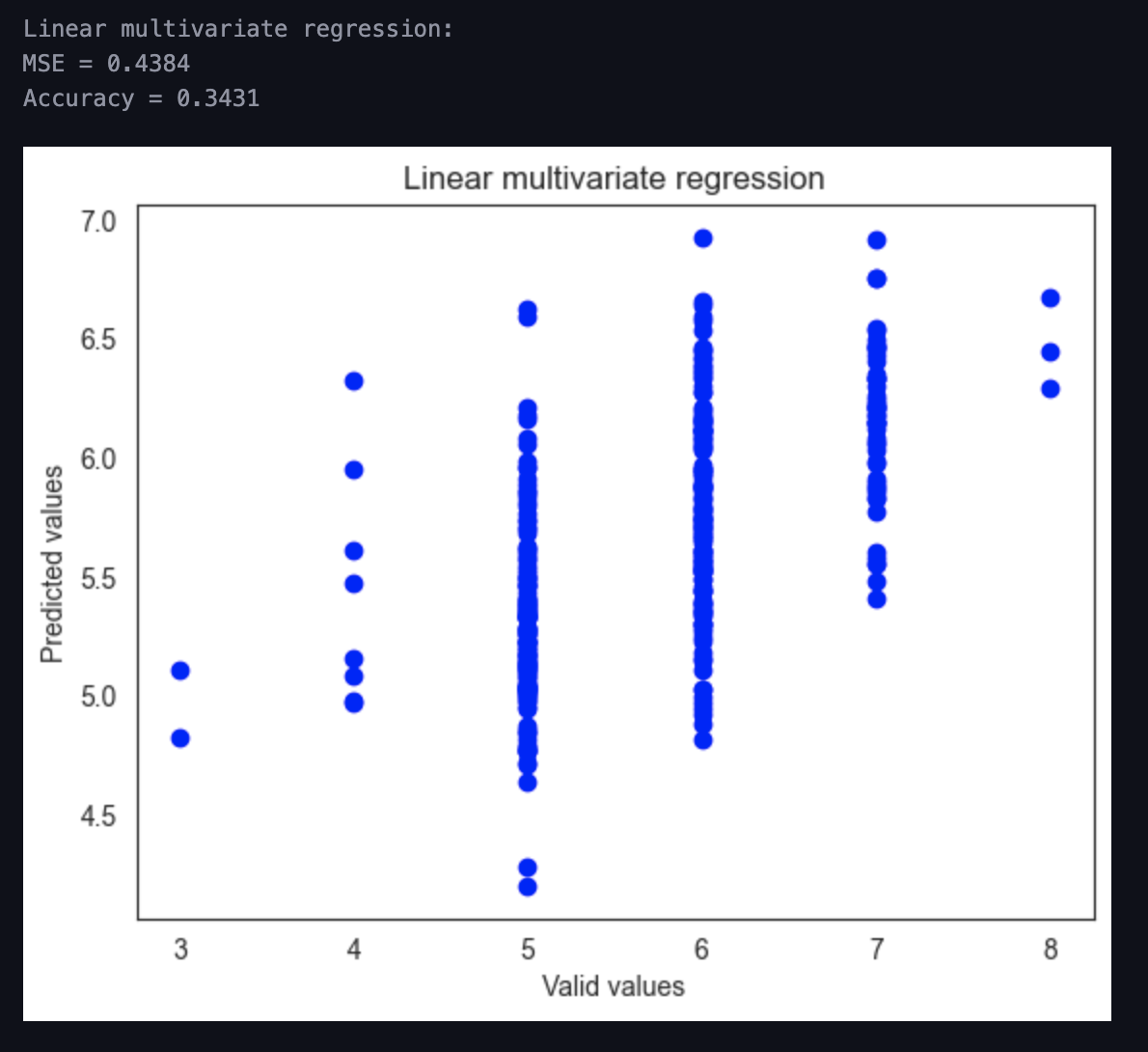
plt.scatter(Y\_test, y\_pred, color='blue')

plt.xlabel('Valid values')

plt.ylabel('Predicted values')

plt.title('Linear multivariate regression')

plt.show()



*# 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))

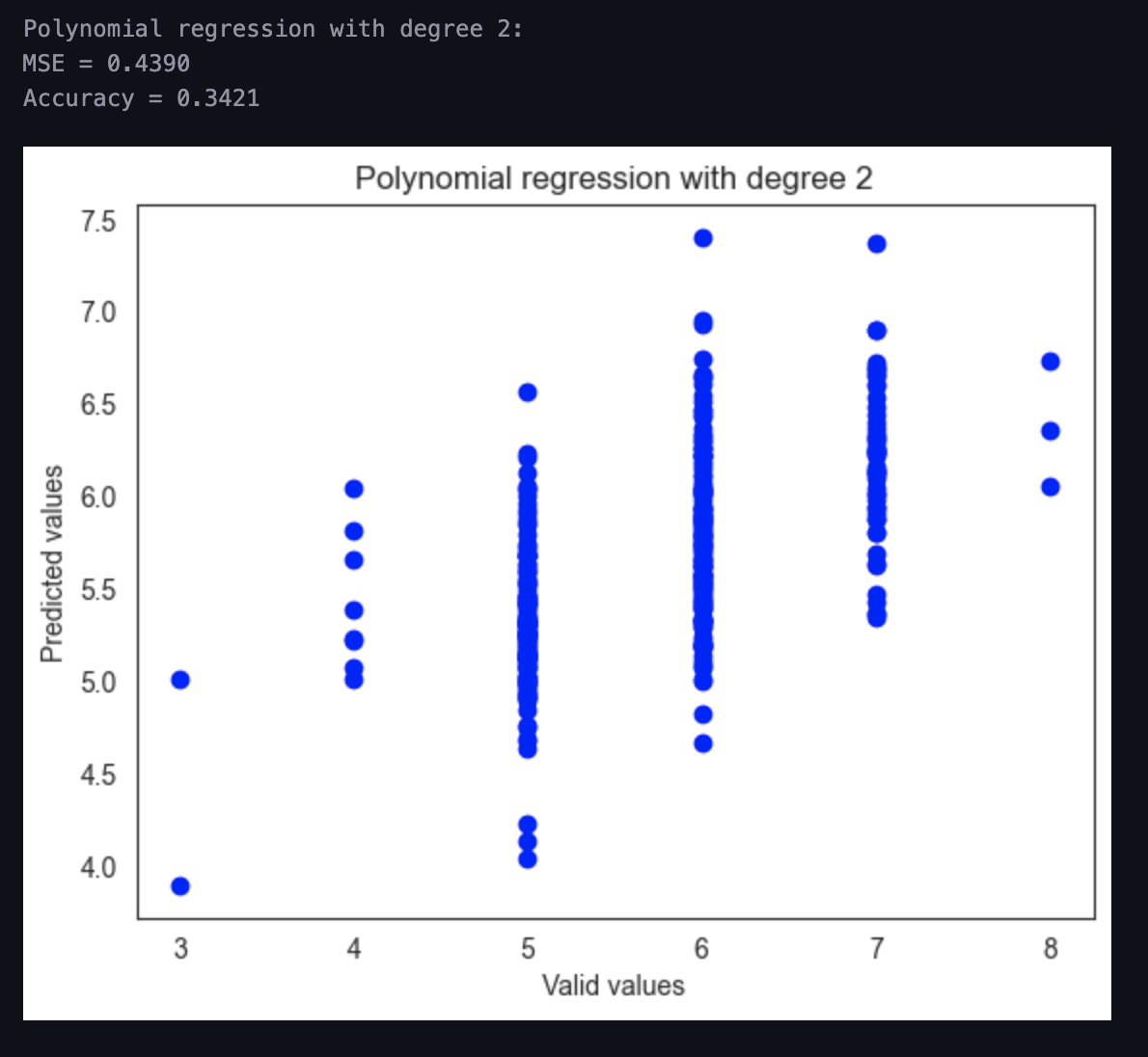
plt.scatter(Y\_test, y\_pred, color='blue')

plt.xlabel('Valid values')

plt.ylabel('Predicted values')

plt.title('Polynomial regression with degree 2')

plt.show()



**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

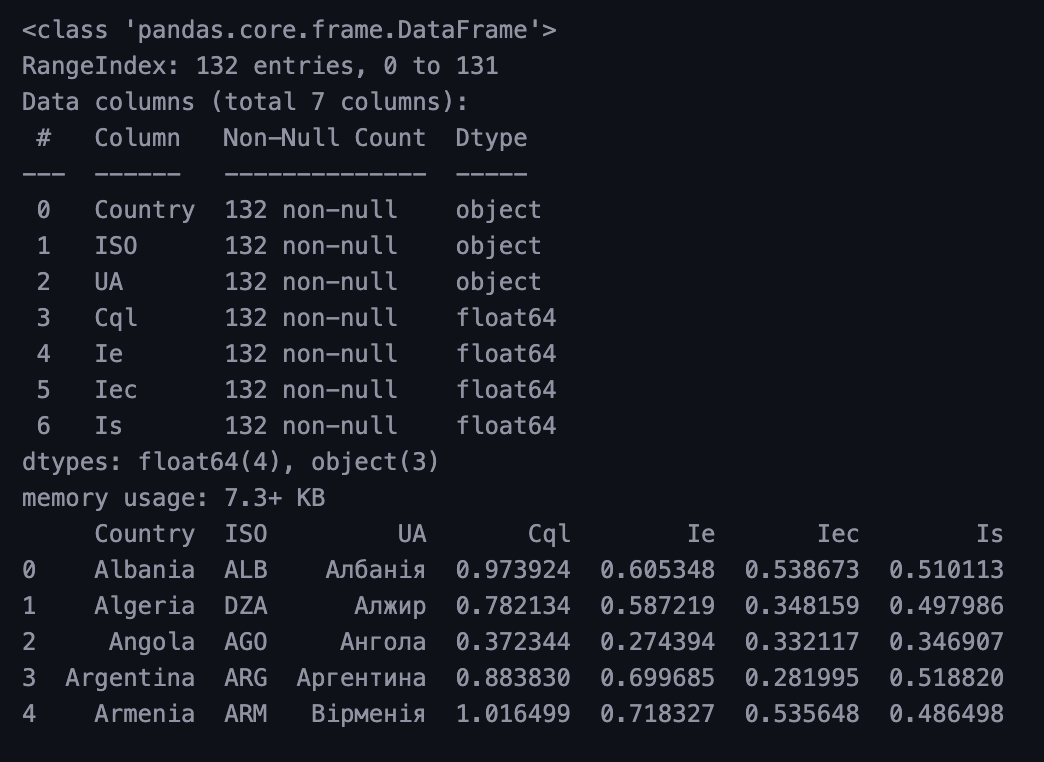
*Завантажемо файл.*

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

*Дослідимо дані*

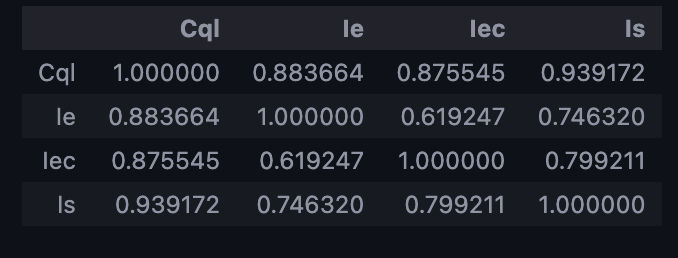
df.info()

print(df.head())



*Перевіремо мультиколінеарність*

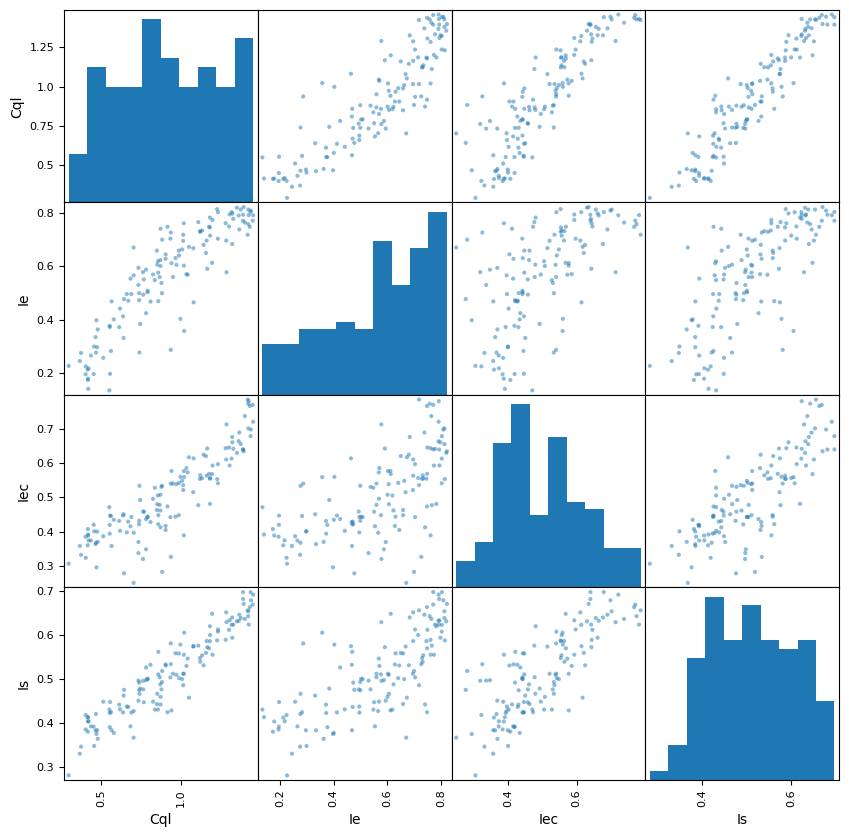
df.corr()



*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]}')

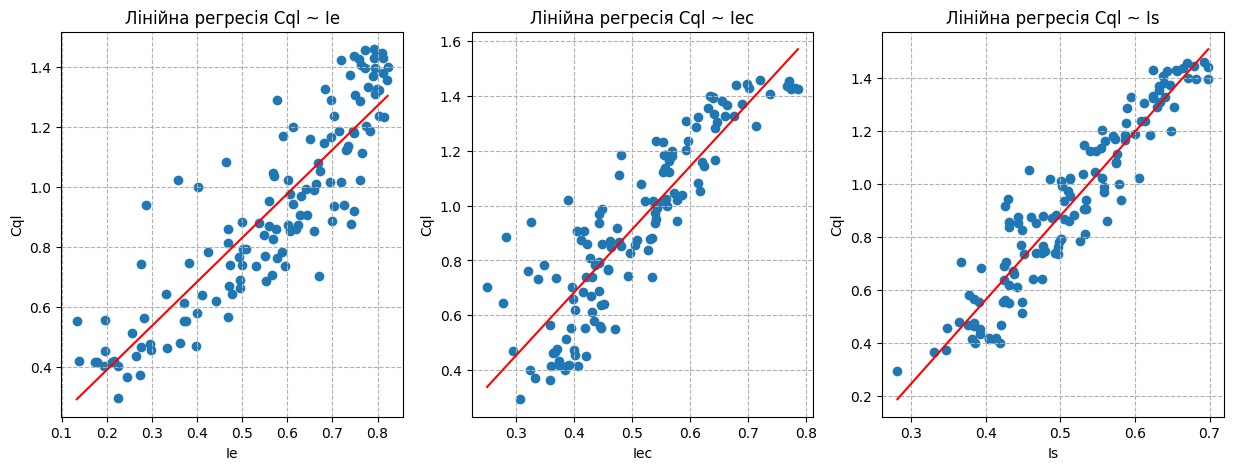
axes[i].set\_xlabel(params[i])

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')

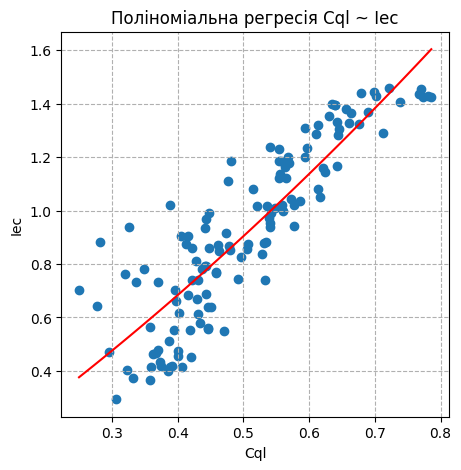
plt.ylabel('Iec')

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),

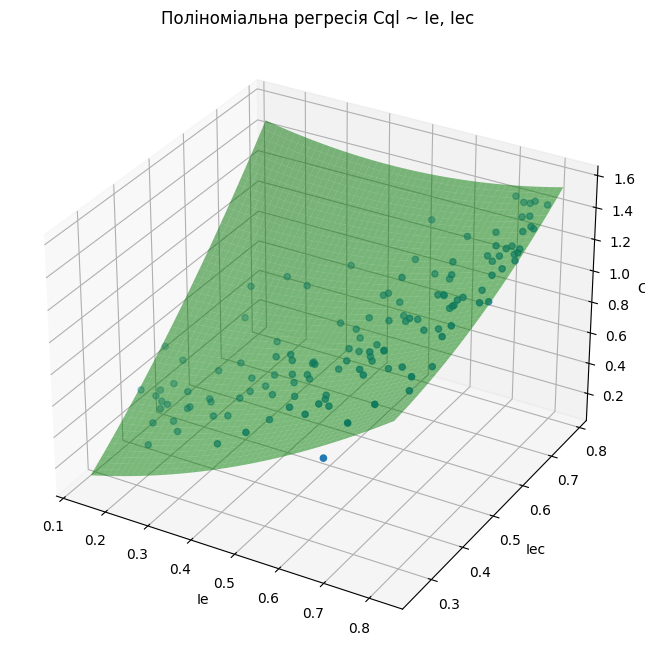
color='green',

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



*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]}')



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.