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 №6

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

Викладач:

Ліхоузова Т.A.

Виконав:

студент 2 курсу групи ІП-15 ФІОТ  
Мєшков Андрій Ігорович

Київ-2023

**Практикум №6**

**Кластеризація та класифікація**

**Мета роботи**: ознайомитись з

* методами класифікації та кластеризації;
* моделями, що використовують дерева прийняття рішень;
* інструментами факторного аналізу методом головних компонент та методом найбільшої подібності.

**Завдання:**

Скачати потрібні дані.

**Основне завдання**

Для даних по титаніку titanic.csv побудувати модель, в якій можна визначити, чи виживе пасажир, заповнивши решту параметрів. Використати декілька методів. Порівняти результати.

**Додаткове завдання**

Використовуючи файл Data2.csv

1. визначити, який регіон домінує в кластерах по ВВП на душу населення та щільності населення
2. вивести частотні гістограми всіх показників файла Data2.csv, використовуючи цикл
3. створити функцію, яка на вхід отримує два набори даних, перевіряє чи є лінійна залежність та виводить True чи False (будемо розуміти під «є лінійна залежність», якщо коефіцієнт кореляції по модулю більше 0,8)

**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.preprocessing *import* StandardScaler

*from* sklearn.linear\_model *import* LogisticRegression

*from* sklearn.ensemble *import* AdaBoostClassifier

*from* sklearn.tree *import* DecisionTreeClassifier

*from* sklearn.ensemble *import* RandomForestClassifier

*from* sklearn.metrics *import* accuracy\_score

*from* sklearn.model\_selection *import* cross\_val\_score

*import* matplotlib.pyplot *as* plt

*import* matplotlib.patches *as* mpatches

*import* seaborn *as* sns

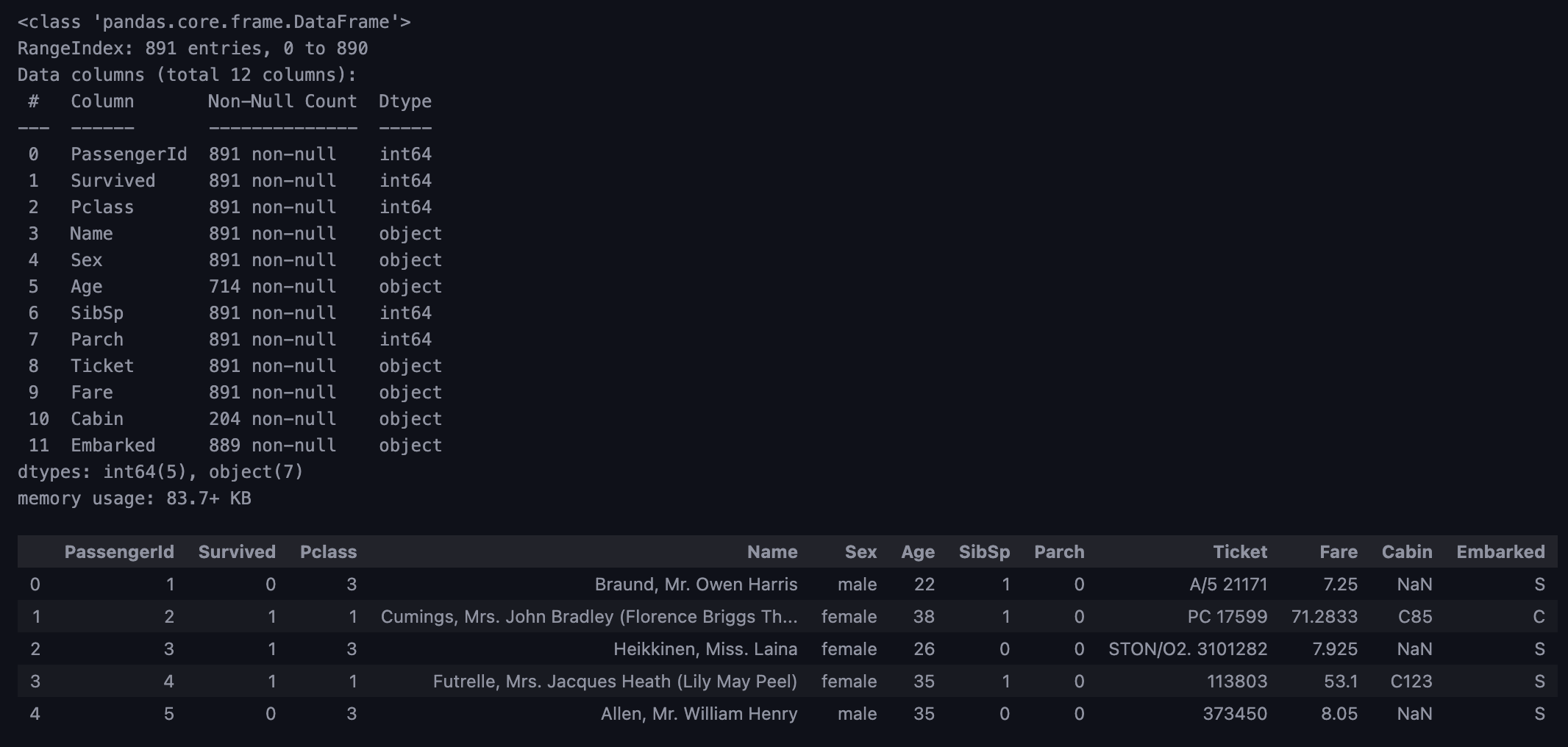
*Зчитаємо файл.*

df = pd.read\_csv('titanic.csv', sep=',', decimal=',', encoding='windows-1251')

*Проаналізуємо структуру.*

df.info()

df.head()



*Підготовка та дослідження даних*

df = df.drop(columns=['PassengerId', 'Name'])

df['Pclass'] = df['Pclass'].astype(str)

*Розділіть дані на навчальну та тестову вибірки.*

df\_train, df\_test = train\_test\_split(

df,

test\_size=0.2,

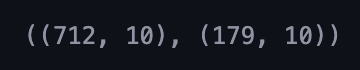
random\_state=1

)

df\_train.head()



df\_train.shape, df\_test.shape



*Кількість виживших та загиблих відповідно в навчальному сеті*

df\_train['Survived'].sum()

df\_train['Survived'].count() - df\_train['Survived'].sum()

*Дослідимо пропущені значення*

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

value\_is = mpatches.Patch(color='purple', label='Value')

value\_not = mpatches.Patch(color='aqua', label='No value')

plt.title('Heatmap missing values')

plt.legend(handles=[value\_is, value\_not], bbox\_to\_anchor=(1, 1), loc='upper left')

colours = ['purple', 'aqua']

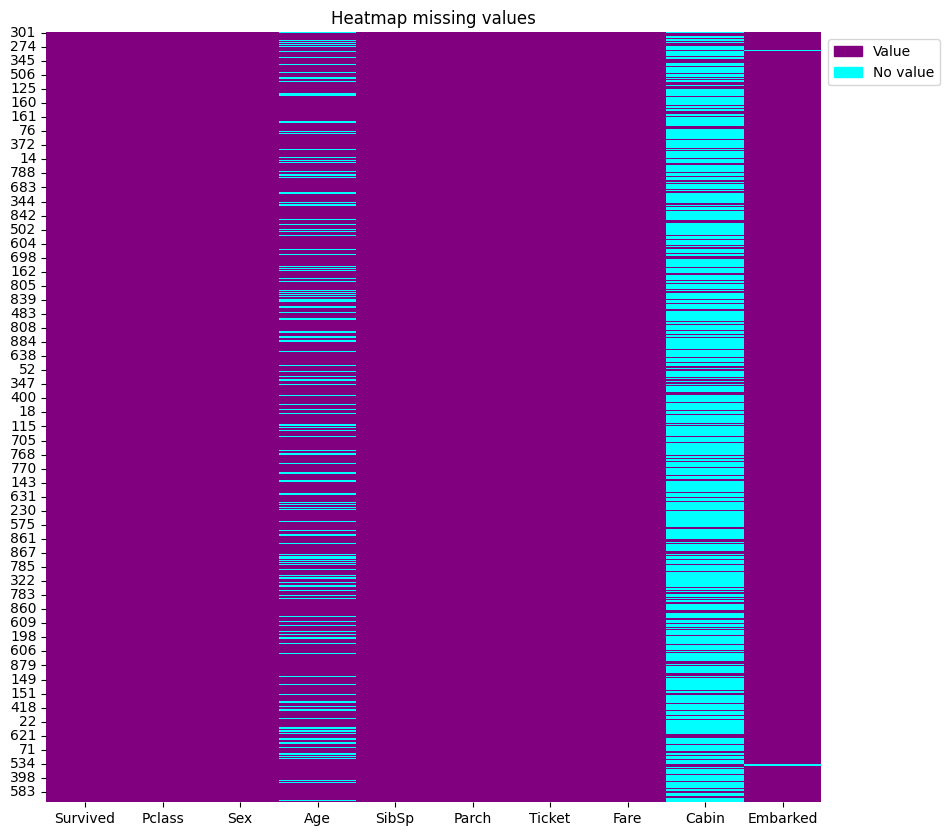
sns.heatmap(

df\_train.isna(), cbar=False,

cmap=sns.color\_palette(colours),

)

plt.show()



total = df.isnull().sum().sort\_values(ascending=False)

percent = (df.isna().mean() \* 100).sort\_values(ascending=False)

missing\_data = pd.concat([total, percent], axis=1, keys=['Total', 'Percent'])

missing\_data.head()



*Видалимо непотрібні дані та скорегуємо потрібні*

df\_train = df\_train.drop(columns=['Cabin', 'Ticket'])

df\_test = df\_test.drop(columns=['Cabin', 'Ticket'])

df\_train = df\_train.fillna(df\_train.mean())

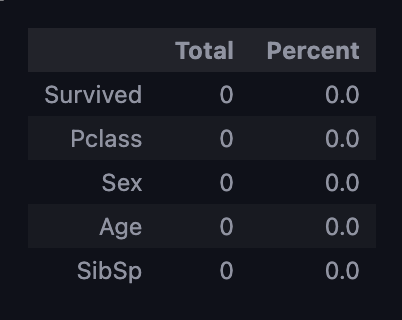
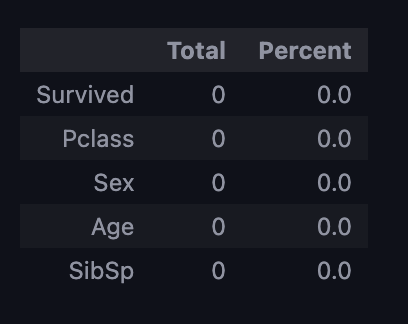
df\_test = df\_test.fillna(df\_test.mean())

df\_train['Embarked'] = df\_train['Embarked'].fillna(df\_train['Embarked'].mode()[0])

df\_test['Embarked'] = df\_test['Embarked'].fillna(df\_train['Embarked'].mode()[0])

df\_train['Age'] = df\_train['Age'].fillna(df\_train['Age'].mode()[0])

df\_test['Age'] = df\_test['Age'].fillna(df\_train['Age'].mode()[0])

**

*Кодуємо категоріальні значення*

all\_features = pd.concat([df\_train, df\_test]).reset\_index(drop=True)

all\_features = pd.get\_dummies(all\_features)

df\_train = all\_features.iloc[:df\_train.shape[0], :]

df\_test = all\_features.iloc[df\_train.shape[0]:, :]

*Розділіть дані на навчальні та тестові набори*

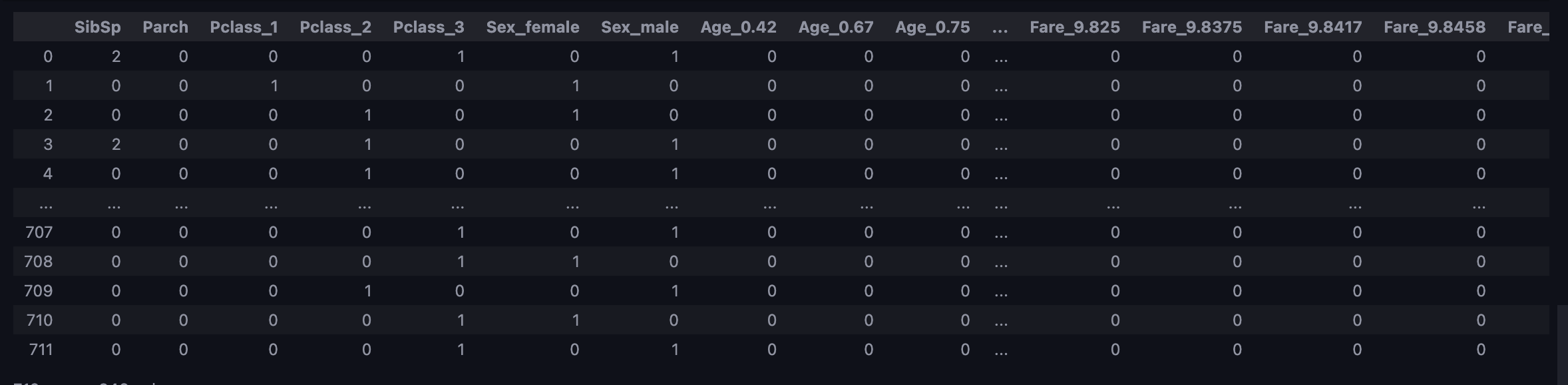
X\_train = df\_train.drop(columns='Survived')

y\_train = df\_train['Survived']

X\_test = df\_test.drop(columns='Survived')

y\_test = df\_test['Survived']

X\_train

**

*Побудуємо моделі*

models = [

('Logistic Regression', LogisticRegression()),

('Decision Tree', DecisionTreeClassifier(max\_depth=3, random\_state=1)),

('Random Forest', RandomForestClassifier(max\_depth=5)),

('AdaBoost Classifier', AdaBoostClassifier(learning\_rate=0.3))

]

i = 0

best = ''

*for* name, model *in* models:

*# Train model*

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

scores = cross\_val\_score(model, X\_train, y\_train, cv=5)

*# Predict on test data*

y\_pred = model.predict(X\_test)

*# Evaluate model performance*

accuracy = accuracy\_score(y\_test, y\_pred)

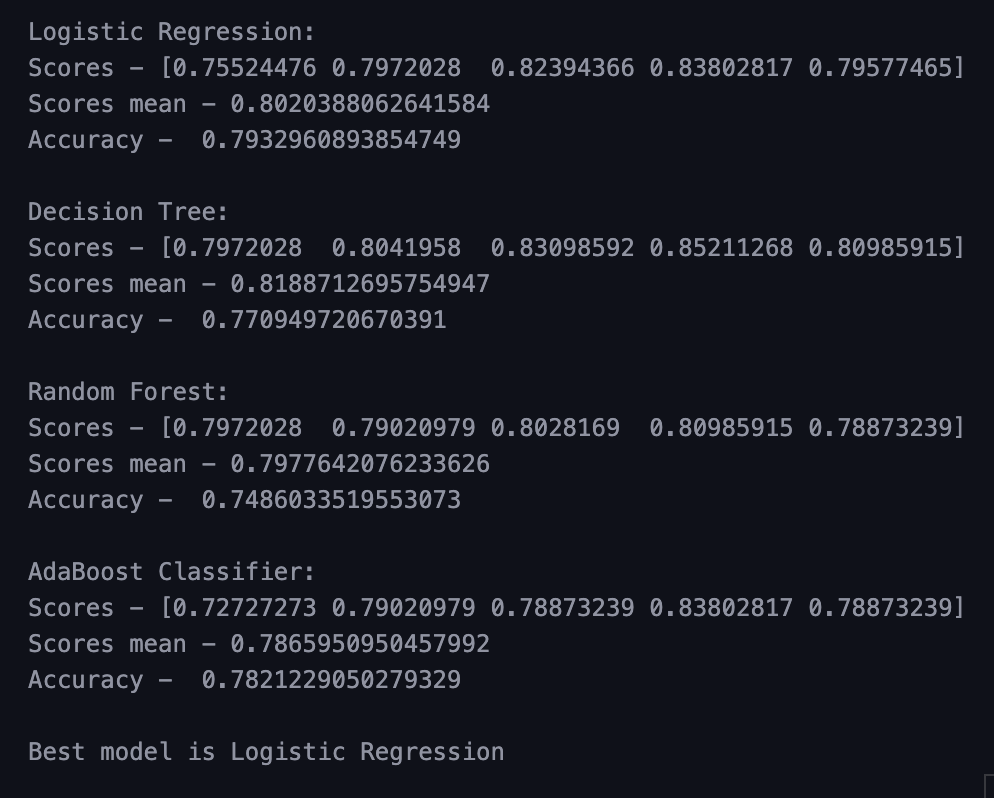
print(f'{name}: \nScores - {scores} \nScores mean - {scores.mean()} \nAccuracy - {accuracy:}\n')

*if* accuracy>i:

i = accuracy

best = name

print(f'Best model is {best}')



**Додаткове завдання:**

*Імпортуємо потрібні бібліотеки.*

*import* pandas *as* pd

*import* matplotlib.pyplot *as* plt

*from* sklearn.cluster *import* KMeans

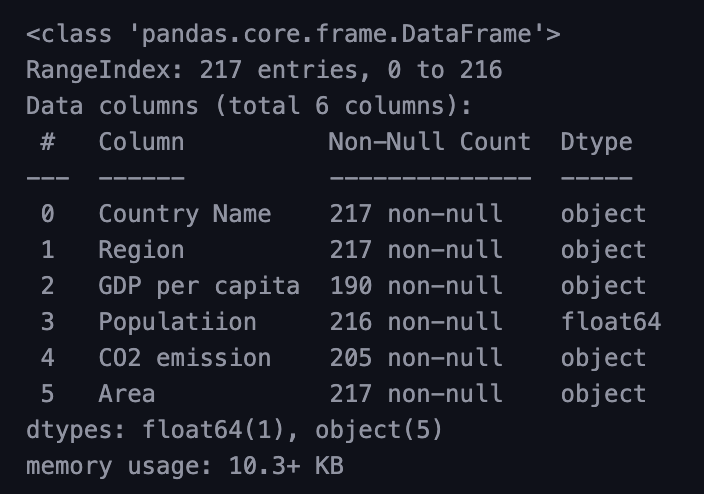
*import* plotly.express *as* px

*Завантажемо файл.*

df = pd.read\_csv('Data2.csv', sep=';', encoding='cp1252')

*Дослідимо дані*

df.info()



*Зкорегуємо дані*

df.rename(columns={"Populatiion": "Population"}, inplace=True)

df['Area'] = df['Area'].str.replace(',', '.').astype(float)

df["GDP per capita"] = df["GDP per capita"].str.replace(',', '.').astype(float)

df["CO2 emission"] = df["CO2 emission"].str.replace(',', '.').astype(float)

fix\_gdp = df[df['GDP per capita'] < 0]

area\_gdp = df[df['Area'] < 0]

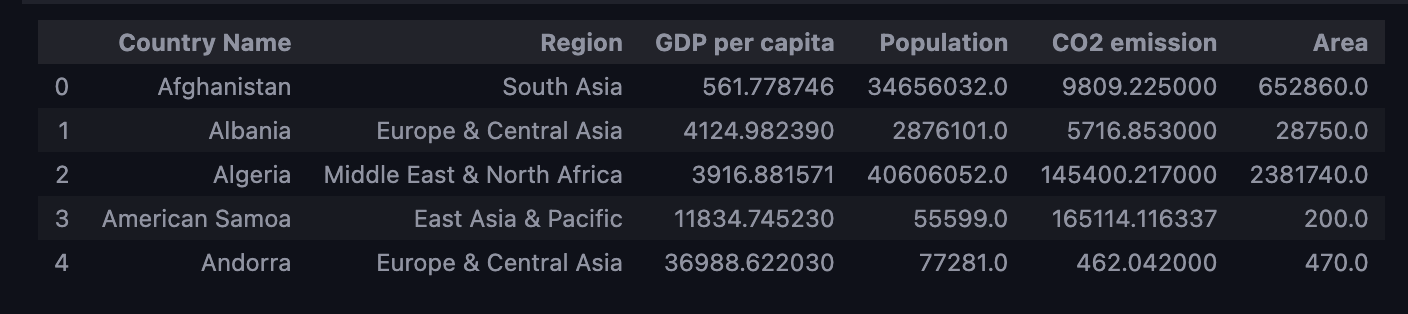
fix\_gdp['GDP per capita'] \*= -1

area\_gdp['Area'] \*= -1

df[df['GDP per capita'] < 0] = fix\_gdp

df[df['Area'] < 0] = area\_gdp

df = df.fillna(df.mean())

**

*Створимо щільність населення*

df['Population density'] = df['Population'] / df['Area']

*Визначимо, який регіон домінує в кластерах по ВВП на душу населення та щільності населення*

*# Selecting the features for clustering*

X = df[['GDP per capita', 'Population density']]

km\_kwargs = {

'init': 'random',

'n\_clusters': 4,

'n\_init': 10,

'max\_iter': 300,

'random\_state': 42,

}

*# Using KMeans clustering algorithm to cluster the data*

km = KMeans(\*\*km\_kwargs)

km.fit(X)

*# Adding the predicted cluster labels to the original data*

df['Cluster'] = km.labels\_

*# Grouping the data by region and cluster and calculating the mean for each group*

region\_cluster\_means = df.groupby(['Region', 'Cluster']).mean()

*# Sorting the data by GDP per capita and population density*

sorted\_data = region\_cluster\_means.sort\_values(['GDP per capita', 'Population density'], ascending=False)

*# Displaying the dominant region for GDP per capita and population density clusters*

print('\n\n')

print("Dominant region for GDP per capita cluster: ", sorted\_data.loc[sorted\_data['GDP per capita'].idxmax()].name[0])

print("Dominant region for population density cluster: ", sorted\_data.loc[sorted\_data['Population density'].idxmax()].name[0])

print('\n\n')

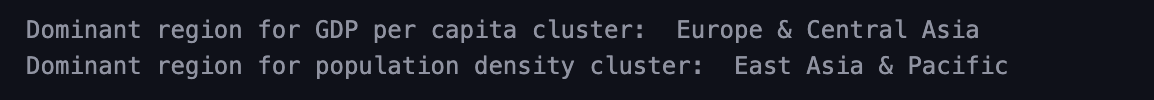
**

fig = px.scatter(

df, x='GDP per capita', y='Population density', color=km.labels\_,

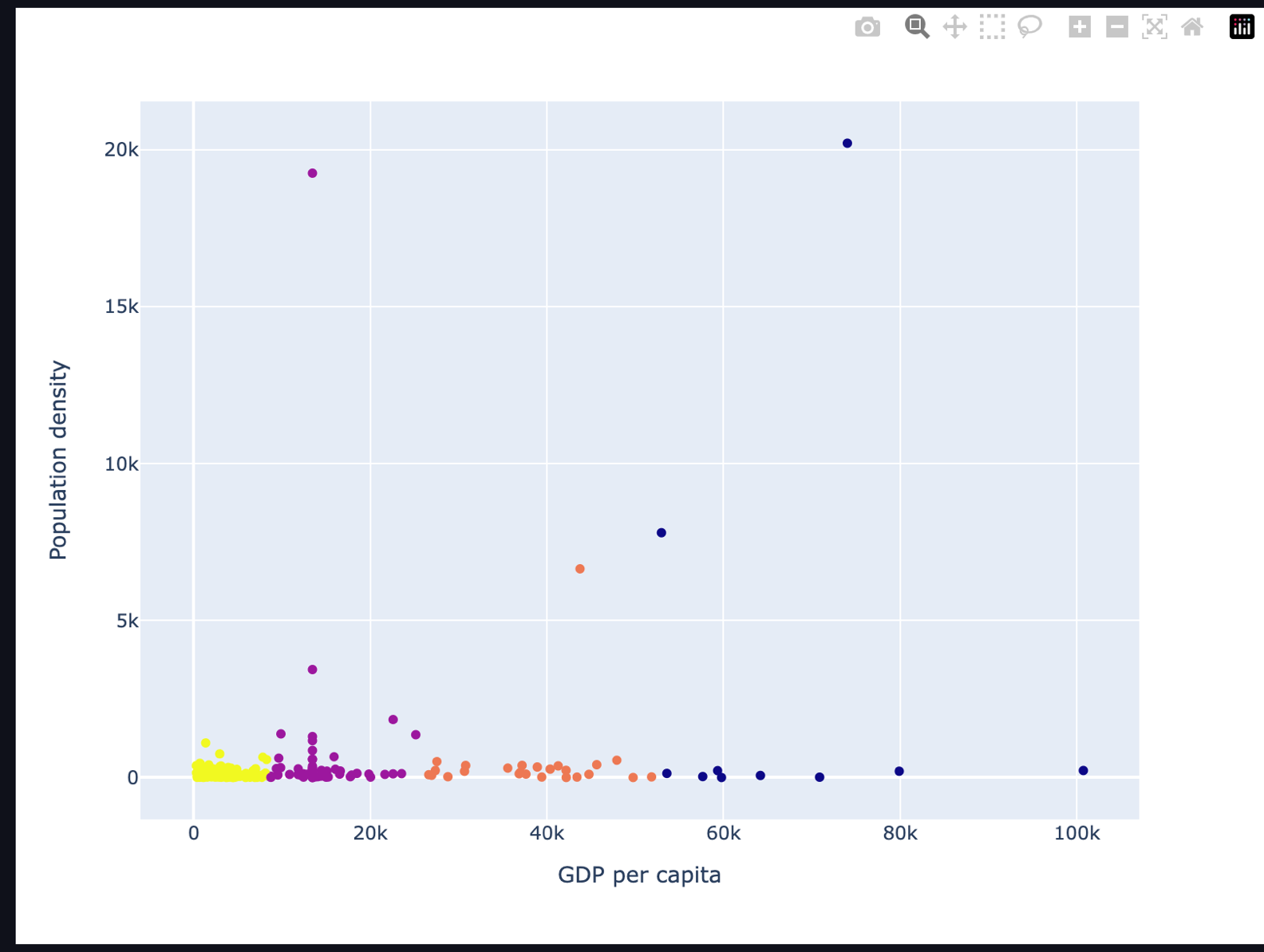
hover\_data=['Country Name', 'Region'],

width=800, height=600

)

fig.update(layout\_coloraxis\_showscale=False)

fig.show()



*Виведемо частотні гістограми всіх показників файла Data2.csv, використовуючи цикл*

fig, axes = plt.subplots(2, 3, figsize=(15, 10))

labels = df.columns[2:]

*for* i *in* range(len(labels)):

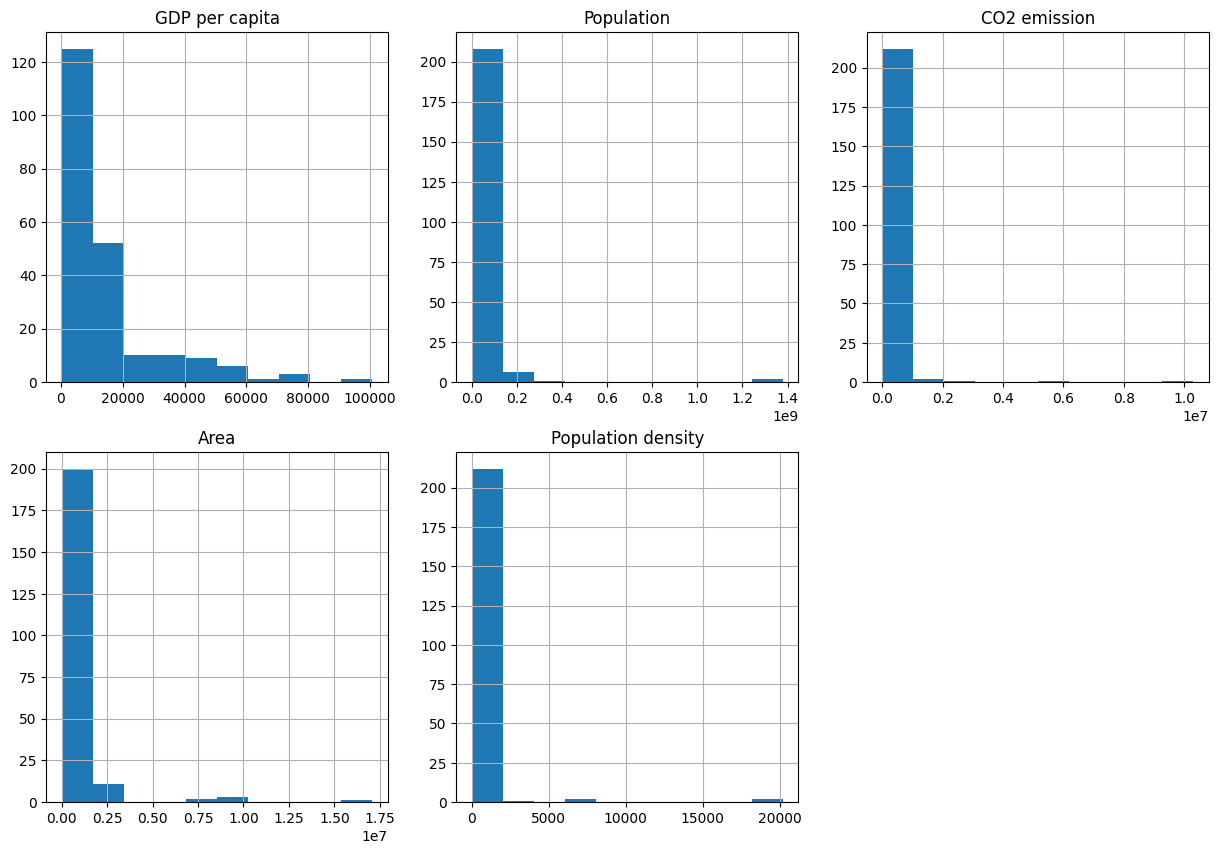
ax\_i = (i // 3, i % 3)

axes[ax\_i].set\_title(labels[i])

axes[ax\_i].grid('-')

axes[ax\_i].hist(df[labels[i]])

fig.delaxes(axes[1][2])



*Створимо функцію.*

*import* numpy *as* np

*from* scipy.stats *import* pearsonr

def linear\_relationship(x, y):

*# Calculating the correlation coefficient and p-value*

corr, p\_value = pearsonr(x, y)

print(f'Correlation coefficient: {corr}')

*# Checking if the absolute value of the correlation coefficient is greater than 0.8*

*if* abs(corr) > 0.8:

*return* True

*else*:

*return* False

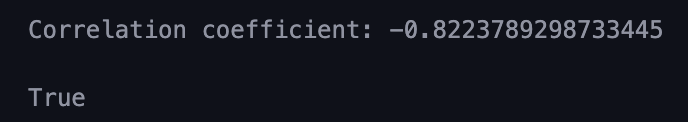
*# 1000 random integers between 0 and 50*

x = np.random.randint(0, 50, 1000)

*# Negative Correlation with some noise*

y = -x + np.random.normal(0, 10, 1000)

linear\_relationship(x, y)



Висновок

За отриманими даними можна зробити висновок, що

* В основному завданні найкращою моделлю виявилась LogisticRegression. Проте ефективність моделей можна покращити, краще обробивши дані та підібравши кращі гіперпараметри моделей.
* В додатковому завданні домінуючий регіон за ВВП - Europe & Central Asia, а за Щільністю населення - East Asia & Pacific.