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 №6

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

«Кластеризація та класифікація»

Викладач:
Ліхоузова Т.А.

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()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 891 entries, 0 to 890
Data columns (total 12 columns):
 #   Column       Non-Null Count  Dtype
---  ------       --------------  -----
 0   PassengerId  891 non-null    int64
 1   Survived     891 non-null    int64
 2   Pclass       891 non-null    int64
 3   Name         891 non-null    object
 4   Sex          891 non-null    object
 5   Age          714 non-null    object
 6   SibSp        891 non-null    int64
 7   Parch        891 non-null    int64
 8   Ticket       891 non-null    object
 9   Fare         891 non-null    object
 10  Cabin        204 non-null    object
 11  Embarked     889 non-null    object
dtypes: int64(5), object(7)
memory usage: 83.7+ KB
```

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
0	1	0	3	Braund, Mr. Owen Harris	male	22	1	0	A/5 21171	7.25	NaN	S
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38	1	0	PC 17599	71.2833	C85	C
2	3	1	3	Heikkinen, Miss. Laina	female	26	0	0	STON/O2. 3101282	7.925	NaN	S
3	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35	1	0	113803	53.1	C123	S
4	5	0	3	Allen, Mr. William Henry	male	35	0	0	373450	8.05	NaN	S

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()

	Survived	Pclass	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
301	1	3	male	NaN	2	0	367226	23.25	NaN	Q
309	1	1	female	30	0	0	PC 17485	56.9292	E36	C
516	1	2	female	34	0	0	C.A. 34260	10.5	F33	S
120	0	2	male	21	2	0	S.O.C. 14879	73.5	NaN	S
570	1	2	male	62	0	0	S.W./PP 752	10.5	NaN	S

df_train.shape, df_test.shape

```
((712, 10), (179, 10))
```

Кількість вижившиx та загиблих відповідно в навчальному сеті
df_train['Survived'].sum()
df_train['Survived'].count() - df_train['Survived'].sum()

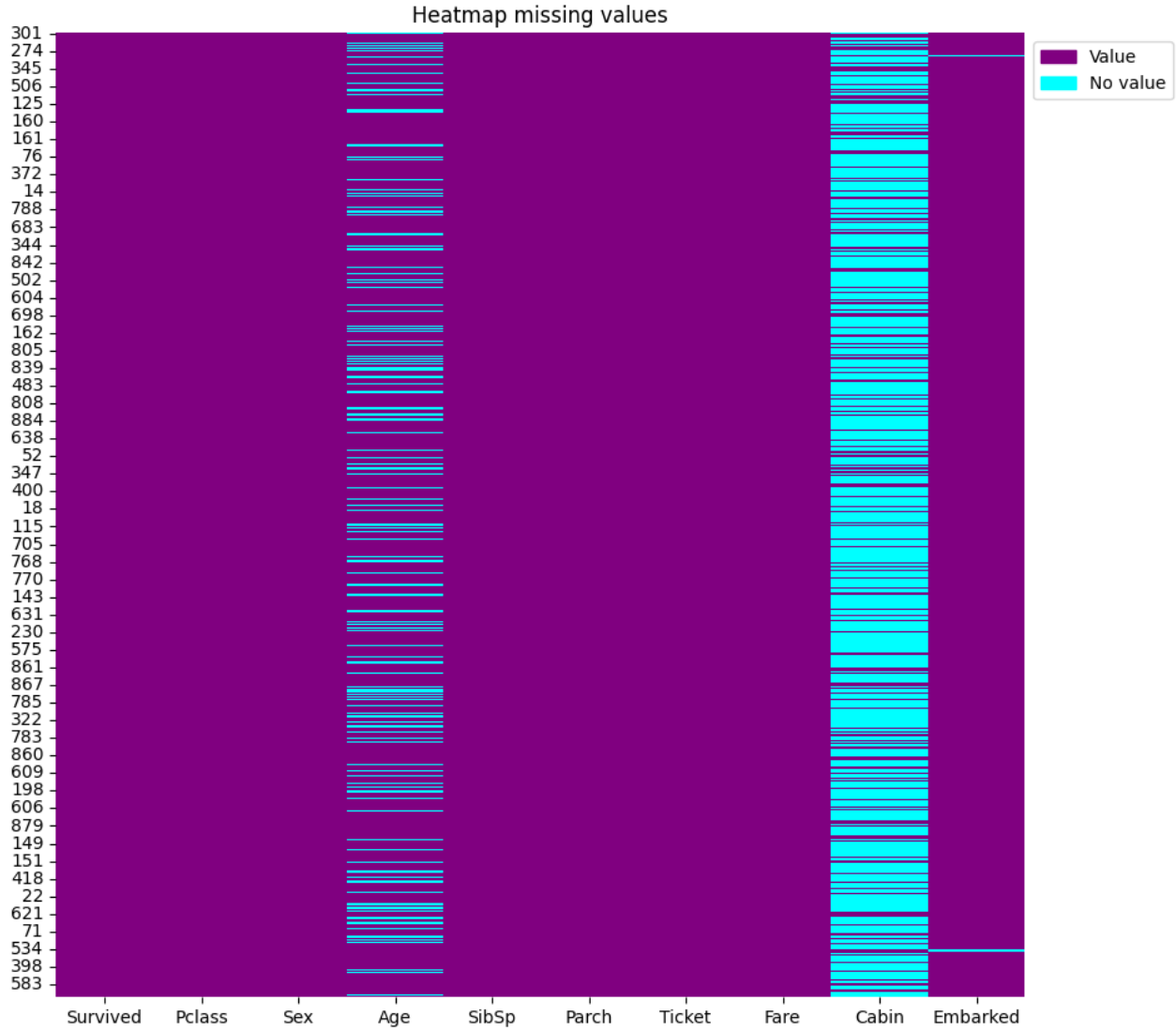
Дослідимо пропущені значення
plt.figure(figsize=(10, 10))

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()
```

	Total	Percent
Cabin	687	77.104377
Age	177	19.865320
Embarked	2	0.224467
Survived	0	0.000000
Pclass	0	0.000000

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])
```

	Total	Percent
Survived	0	0.0
Pclass	0	0.0
Sex	0	0.0
Age	0	0.0
SibSp	0	0.0

	Total	Percent
Survived	0	0.0
Pclass	0	0.0
Sex	0	0.0
Age	0	0.0
SibSp	0	0.0

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
```

	SibSp	Parch	Pclass_1	Pclass_2	Pclass_3	Sex_female	Sex_male	Age_0.42	Age_0.67	Age_0.75	...	Fare_9.825	Fare_9.8375	Fare_9.8417	Fare_9.8458	Fare_
0	2	0	0	0	1	0	1	0	0	0	...	0	0	0	0	
1	0	0	1	0	0	1	0	0	0	0	...	0	0	0	0	
2	0	0	0	1	0	1	0	0	0	0	...	0	0	0	0	
3	2	0	0	1	0	0	1	0	0	0	...	0	0	0	0	
4	0	0	0	1	0	0	1	0	0	0	...	0	0	0	0	
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	
707	0	0	0	0	1	0	1	0	0	0	...	0	0	0	0	
708	0	0	0	0	1	1	0	0	0	0	...	0	0	0	0	
709	0	0	0	1	0	0	1	0	0	0	...	0	0	0	0	
710	0	0	0	0	1	1	0	0	0	0	...	0	0	0	0	
711	0	0	0	0	1	0	1	0	0	0	...	0	0	0	0	

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}')
```

```
Logistic Regression:
Scores - [0.75524476 0.7972028  0.82394366 0.83802817 0.79577465]
Scores mean - 0.8020388062641584
Accuracy -  0.7932960893854749

Decision Tree:
Scores - [0.7972028  0.8041958  0.83098592 0.85211268 0.80985915]
Scores mean - 0.8188712695754947
Accuracy -  0.770949720670391

Random Forest:
Scores - [0.7972028  0.79020979 0.8028169  0.80985915 0.78873239]
Scores mean - 0.7977642076233626
Accuracy -  0.7486033519553073

AdaBoost Classifier:
Scores - [0.72727273 0.79020979 0.78873239 0.83802817 0.78873239]
Scores mean - 0.7865950950457992
Accuracy -  0.7821229050279329

Best model is Logistic Regression
```

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()

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 217 entries, 0 to 216
Data columns (total 6 columns):
 #   Column          Non-Null Count  Dtype
---  ------          --------------  -----
 0   Country Name    217 non-null    object
 1   Region          217 non-null    object
 2   GDP per capita  190 non-null    object
 3   Populatiion     216 non-null    float64
 4   CO2 emission    205 non-null    object
 5   Area            217 non-null    object
dtypes: float64(1), object(5)
memory usage: 10.3+ KB
```

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())

	Country Name	Region	GDP per capita	Population	CO2 emission	Area
0	Afghanistan	South Asia	561.778746	34656032.0	9809.225000	652860.0
1	Albania	Europe & Central Asia	4124.982390	2876101.0	5716.853000	28750.0
2	Algeria	Middle East & North Africa	3916.881571	40606052.0	145400.217000	2381740.0
3	American Samoa	East Asia & Pacific	11834.745230	55599.0	165114.116337	200.0
4	Andorra	Europe & Central Asia	36988.622030	77281.0	462.042000	470.0

Створимо щільність населення
df['Population density'] = df['Population'] / df['Area']

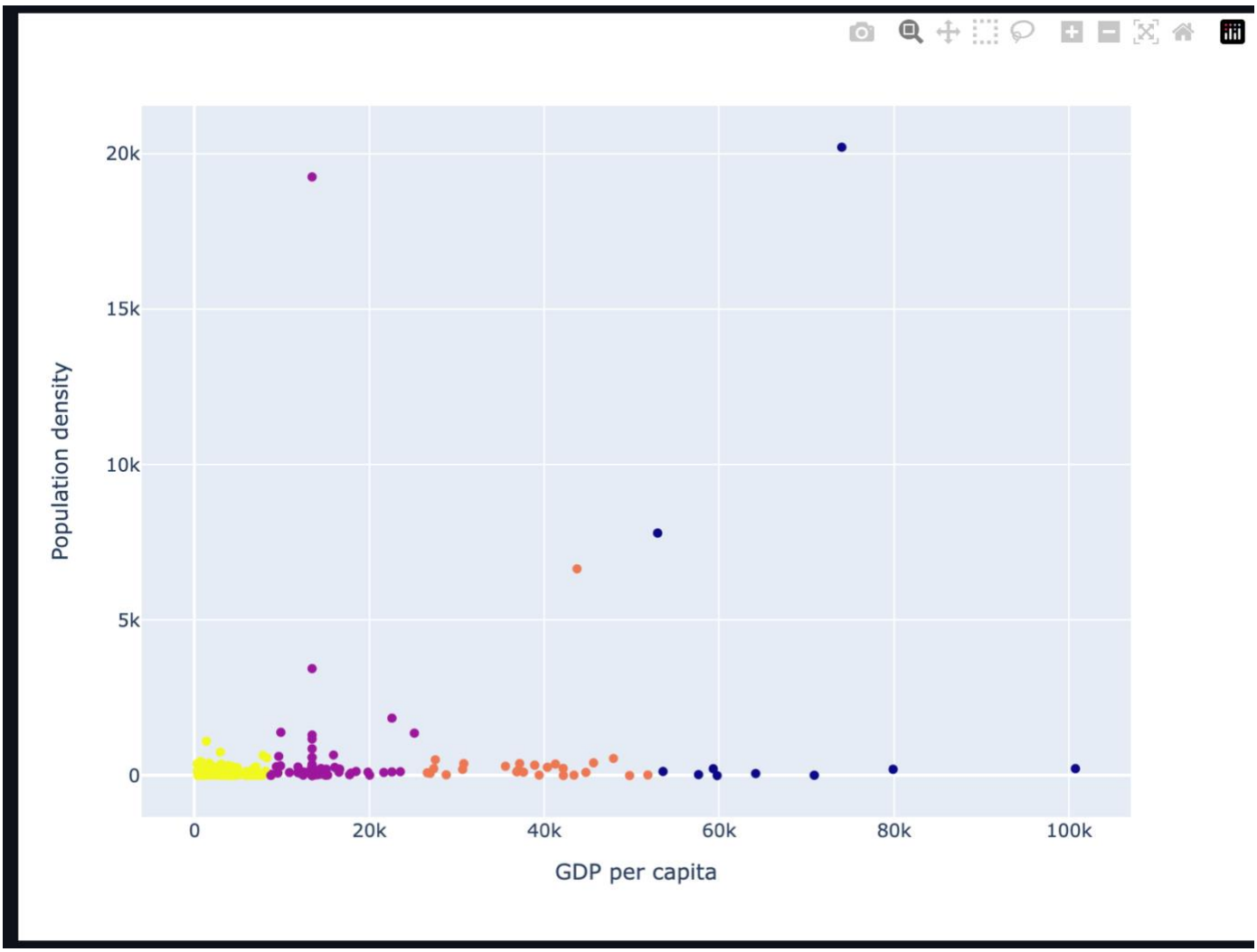
Визначимо, який регіон домінує в кластерах по ВВП на душу населення та щільності населення
 # Selecting the features for clustering
X = df[['GDP per capita', 'Population density']]

```
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')
```

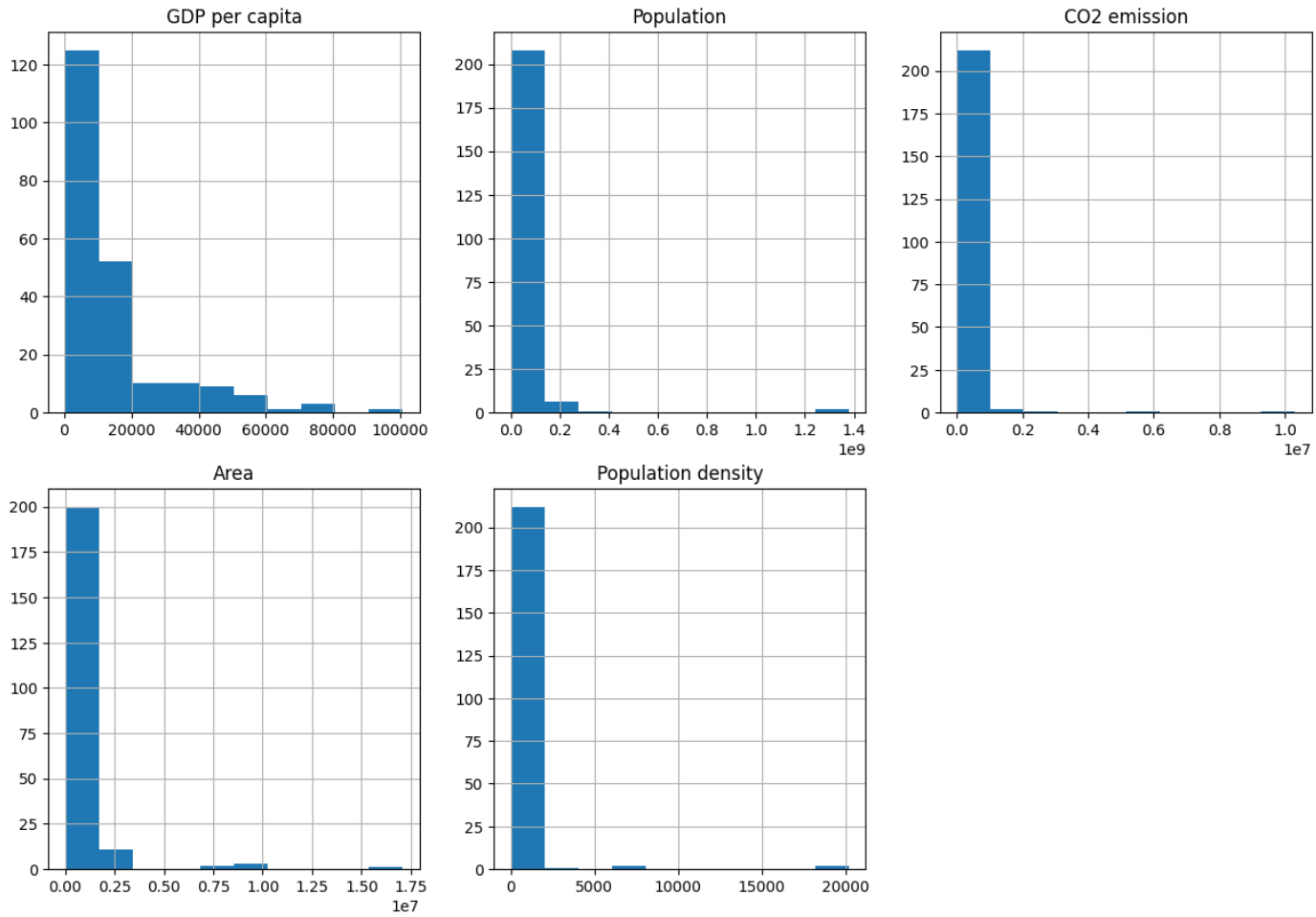
```
Dominant region for GDP per capita cluster:  Europe & Central Asia
Dominant region for population density cluster:  East Asia & Pacific
```

```
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)
```

```
Correlation coefficient: -0.8223789298733445

True
```

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.