**Тема:** Класичні задачі машинного навчання та методи їх вирішення.

**Мета:** Ознайомитися з основними задачами машинного навчання (ML), навчитися застосовувати різні алгоритми для їх вирішення, провести порівняльний аналіз ефективності методів.

**Завдання:**

1. **Підготовка середовища та встановлення необхідних бібліотек**

pip install numpy pandas matplotlib seaborn scikit-learn

import numpy as np

import pandas as pd

import matplotlib.pyplot as plt

import seaborn as sns

from sklearn.model\_selection import train\_test\_split

from sklearn.metrics import classification\_report, confusion\_matrix

1. **Задача класифікації:**

**Виконайте завантаження даних**  
  
from sklearn.datasets import load\_iris

data = load\_iris()

df = pd.DataFrame(data.data, columns=data.feature\_names)

df['target'] = data.target

print(df.head())

**Проведіть розподіл даних на тренувальну та тестову вибірки**  
  
X = df.drop('target', axis=1)

y = df['target']

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=42)

**Запустіть процес навчання моделей**

from sklearn.linear\_model import LogisticRegression

from sklearn.neighbors import KNeighborsClassifier

from sklearn.tree import DecisionTreeClassifier

# Ініціалізація моделей

models = {

'Logistic Regression': LogisticRegression(),

'K-Nearest Neighbors': KNeighborsClassifier(),

'Decision Tree': DecisionTreeClassifier()

}

# Навчання та оцінка моделей

for name, model in models.items():

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

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

print(f"Model: {name}")

print(classification\_report(y\_test, y\_pred))

**Виконайте візуалізацію матриці плутанини (confusion matrix)**

model = DecisionTreeClassifier()

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

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

cm = confusion\_matrix(y\_test, y\_pred)

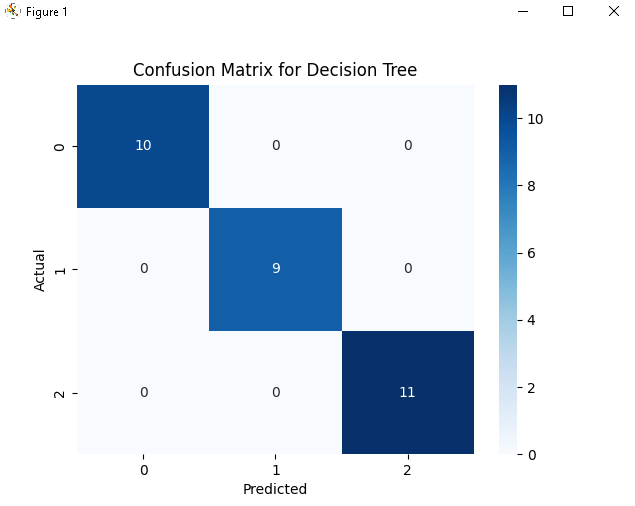
sns.heatmap(cm, annot=True, fmt='d', cmap='Blues')

plt.xlabel('Predicted')

plt.ylabel('Actual')

plt.title('Confusion Matrix for Decision Tree')

plt.show()



1. **Задача регресії:**

**Виконайте завантаження даних**  
  
from sklearn.datasets import fetch\_california\_housing

data = fetch\_california\_housing()

df = pd.DataFrame(data.data, columns=data.feature\_names)

df['target'] = data.target

print(df.head())

**Проведіть розподіл даних на тренувальну та тестову вибірки (аналогічно до попереднього завдання).**

**Навчання моделей**

from sklearn.linear\_model import LinearRegression

from sklearn.tree import DecisionTreeRegressor

from sklearn.ensemble import RandomForestRegressor

from sklearn.metrics import mean\_absolute\_error, mean\_squared\_error

models = {

'Linear Regression': LinearRegression(),

'Decision Tree': DecisionTreeRegressor(),

'Random Forest': RandomForestRegressor()

}

for name, model in models.items():

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

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

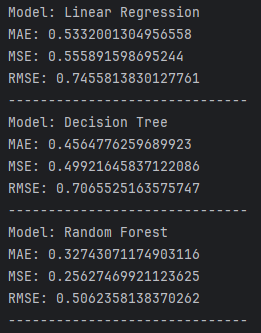
print(f"Model: {name}")

print('MAE:', mean\_absolute\_error(y\_test, y\_pred))

print('MSE:', mean\_squared\_error(y\_test, y\_pred))

print('RMSE:', np.sqrt(mean\_squared\_error(y\_test, y\_pred)))

print('-'\*30)



**Виконайте візуалізацію результатів регресії.**

1. **Зниження розмірності  
     
   Застосуйте алгоритм PCA для візуалізації багатовимірних даних у 2D-просторі**

from sklearn.decomposition import PCA

pca = PCA(n\_components=2)

reduced\_data = pca.fit\_transform(X)

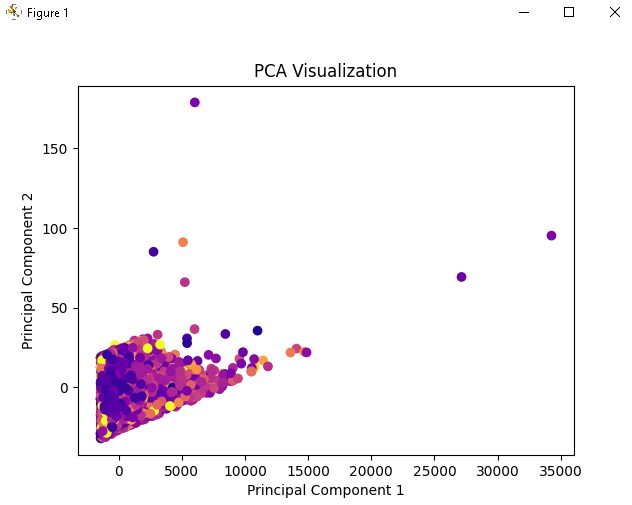
plt.scatter(reduced\_data[:, 0], reduced\_data[:, 1], c=df['KMeans Cluster'], cmap='plasma')

plt.xlabel('Principal Component 1')

plt.ylabel('Principal Component 2')

plt.title('PCA Visualization')

plt.show()

****

**Звіт про практичну роботу:**

Додайте вихідний код розв’язань задач завдань ваш репозиторій на GitHub та підготуйте Google Docs із описом завдань та скріншотами виконання програм та посиланнями на репозиторій.