**Question 1**

1. Implement Naive Bayes Classifier on Iris Dataset  
2. Implement Decision Tree on Iris Dataset  
3. Find accuracy for different splitting points (training size = 0.5, 0.6, 0.7, 0.8, 0.9)  
4. Plot on a graph and compare

**Source Code**

**Naive Bayes Classifier**

import numpy as np

import matplotlib.pyplot as plt

from sklearn.datasets import load\_iris

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

from sklearn.naive\_bayes import GaussianNB

from sklearn.metrics import accuracy\_score, classification\_report

from sklearn.decomposition import PCA

from matplotlib.colors import ListedColormap

def plot\_decision\_regions(X, y, classifier, resolution=0.02):

markers = ('s', 'x', 'o')

colors = ('red', 'blue', 'lightgreen')

cmap = ListedColormap(colors[:len(np.unique(y))])

x1\_min, x1\_max = X[:, 0].min() - 1, X[:, 0].max() + 1

x2\_min, x2\_max = X[:, 1].min() - 1, X[:, 1].max() + 1

xx1, xx2 = np.meshgrid(np.arange(x1\_min, x1\_max, resolution),

np.arange(x2\_min, x2\_max, resolution))

Z = classifier.predict(np.array([xx1.ravel(), xx2.ravel()]).T)

Z = Z.reshape(xx1.shape)

plt.contourf(xx1, xx2, Z, alpha=0.3, cmap=cmap)

for idx, cl in enumerate(np.unique(y)):

plt.scatter(x=X[y == cl, 0], y=X[y == cl, 1],

alpha=0.8, c=[cmap(idx)], marker=markers[idx],

label=f'Class {cl}')

def naive\_bayes(t\_size):

iris = load\_iris()

X = iris.data # Features: sepal length, sepal width, petal length, petal width

y = iris.target # Target: species (0=setosa, 1=versicolor, 2=virginica)

feature\_names = iris.feature\_names

class\_names = iris.target\_names

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

clf = GaussianNB()

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

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

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

print(f"Accuracy: {accuracy:.2f}\n")

print("Classification Report:")

print(classification\_report(y\_test, y\_pred, target\_names=class\_names))

pca = PCA(n\_components=2)

X\_train\_pca = pca.fit\_transform(X\_train)

X\_test\_pca = pca.transform(X\_test)

clf.fit(X\_train\_pca, y\_train)

plt.figure(figsize=(8, 6))

plot\_decision\_regions(X\_train\_pca, y\_train, classifier=clf)

plt.xlabel('Principal Component 1')

plt.ylabel('Principal Component 2')

plt.legend()

plt.title('Naive Bayes Decision Boundaries (PCA-transformed)')

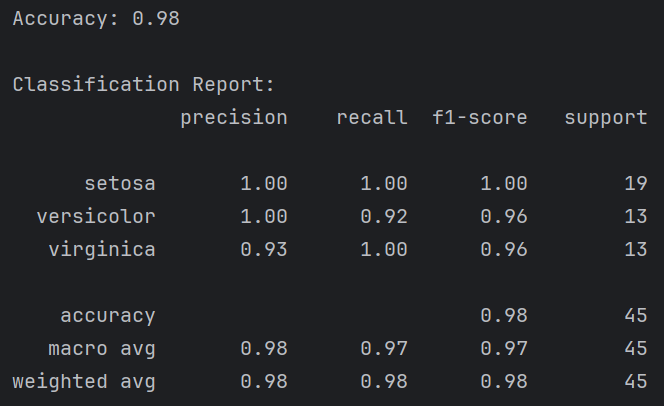
plt.show()

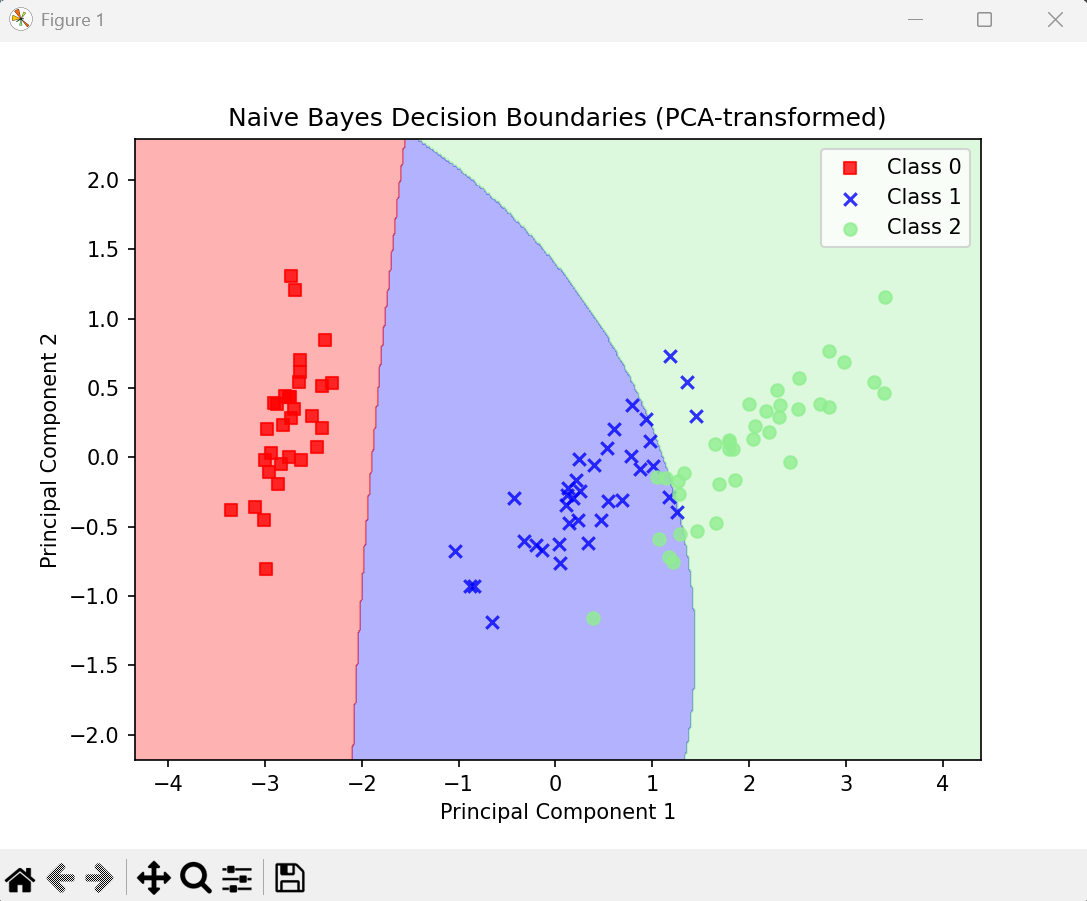
return accuracy

if \_\_name\_\_ == '\_\_main\_\_':

acc = naive\_bayes(0.7)

**Output**

*Terminal*

*Visualization*

**Decision Tree**

from sklearn.datasets import load\_iris

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

from sklearn.tree import DecisionTreeClassifier, plot\_tree

from sklearn.metrics import accuracy\_score, classification\_report

import matplotlib.pyplot as plt

def visualize\_tree(clf, feature\_names, class\_names):

plt.figure(figsize=(10, 5))

plot\_tree(

clf,

feature\_names=feature\_names,

class\_names=class\_names,

filled=True,

rounded=True,

fontsize=6,

)

plt.title("Decision Tree for Iris Species Classification", fontsize=16)

plt.show()

def dec\_tree(train\_size):

iris = load\_iris()

X = iris.data # Features: sepal length, sepal width, petal length, petal width

y = iris.target # Target: species (0=setosa, 1=versicolor, 2=virginica)

feature\_names = iris.feature\_names

class\_names = iris.target\_names

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=1 - train\_size, random\_state=42)

clf = DecisionTreeClassifier(random\_state=42)

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

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

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

print(f"Accuracy: {accuracy:.2f}\n")

print("Classification Report:")

print(classification\_report(y\_test, y\_pred, target\_names=class\_names))

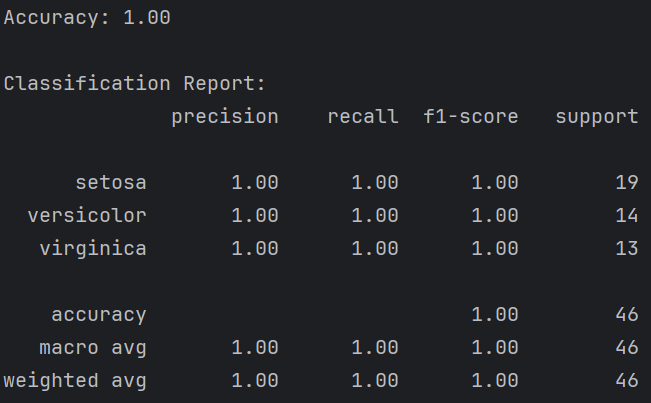
visualize\_tree(clf, feature\_names, class\_names)

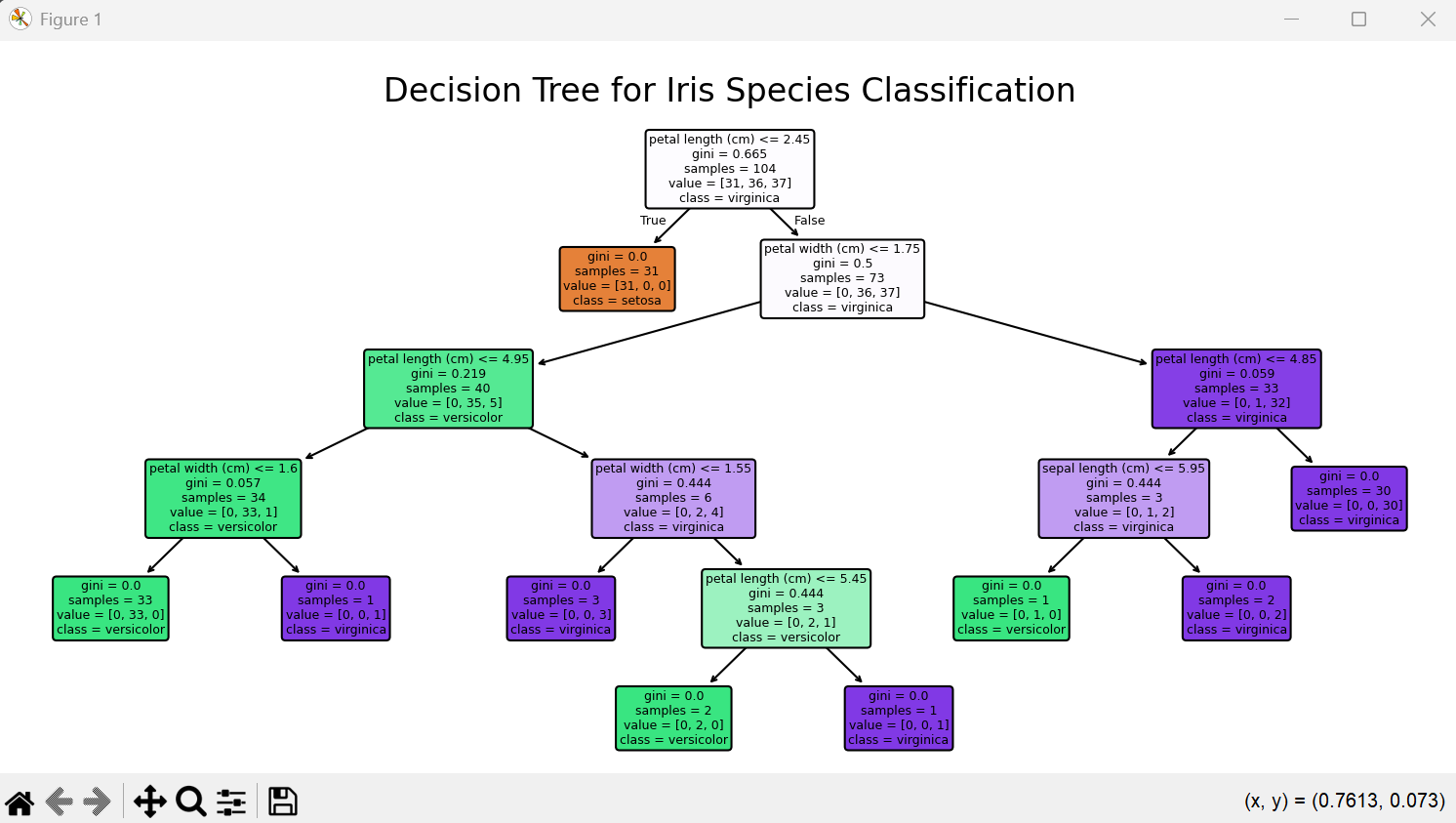
return accuracy

if \_\_name\_\_ == '\_\_main\_\_':

acc = dec\_tree(0.7)

**Output**

*Terminal*

*Visualization*

**Comparison**

from decision\_tree\_iris import dec\_tree

from naive\_bayes\_classifier\_iris import naive\_bayes

import pandas as pd

import matplotlib.pyplot as plt

def comp():

accuracy\_dec = {}

accuracy\_nai = {}

for i in range(5, 10):

accuracy\_dec.update({i/10:dec\_tree(i/10)})

accuracy\_nai.update({i/10:naive\_bayes(i/10)})

df = pd.DataFrame({

'Key': accuracy\_dec.keys(),

'Decision Tree': accuracy\_dec.values(),

'Naive Bayes': accuracy\_nai.values()

})

print (df)

plt.figure(figsize=(8, 4))

plt.plot(df['Key'], df['Decision Tree'], marker='o', label='Decision Tree', linewidth=2)

plt.plot(df['Key'], df['Naive Bayes'], marker='s', label='Naive Bayes', linewidth=2)

plt.title('Model Accuracy Comparison', fontsize=14)

plt.xlabel('Training Size', fontsize=12)

plt.ylabel('Accuracy', fontsize=12)

plt.xticks(df['Key'])

plt.ylim(0.85, 1.01)

plt.grid(True, linestyle='--', alpha=0.7)

plt.legend()

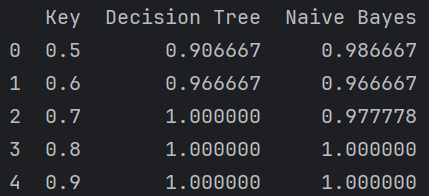
plt.tight\_layout()

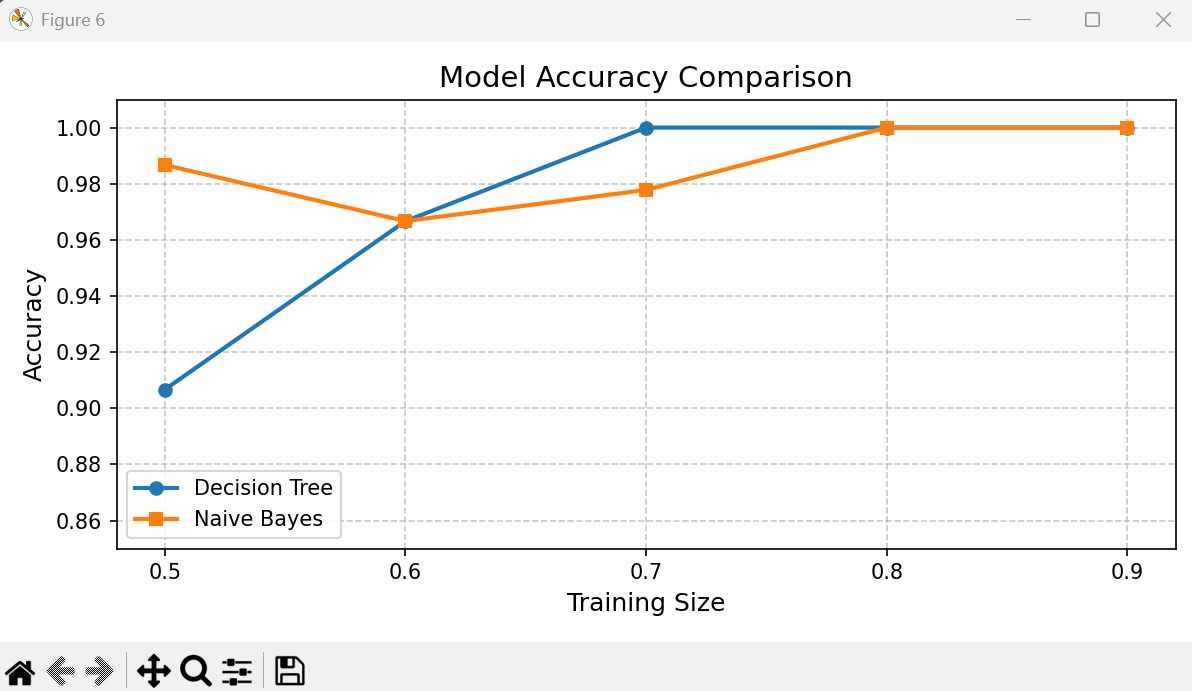
plt.show()

if \_\_name\_\_ == '\_\_main\_\_':

comp()

**Output**

*Terminal*

*Visualization*