**Question**

Implement SVM and compare it with all the other models using the following datasets:  
1. Diabetes

2. Vertebral

**Source Code**

**Diabetes Dataset**

import pandas as pd

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

from sklearn.preprocessing import StandardScaler

from sklearn.tree import DecisionTreeClassifier

from sklearn.neighbors import KNeighborsClassifier

from sklearn.linear\_model import SGDClassifier

from sklearn.naive\_bayes import GaussianNB

from sklearn.svm import SVC

from sklearn.metrics import accuracy\_score, precision\_score, recall\_score, f1\_score

data = pd.read\_csv('dataset/diabetes.csv')

y = data.iloc[:, -1]

X = data.iloc[:, :-1]

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

scaler = StandardScaler()

X\_train\_scaled = scaler.fit\_transform(X\_train)

X\_test\_scaled = scaler.transform(X\_test)

def decision\_tree():

model = DecisionTreeClassifier()

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

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

return y\_pred

def kNN\_classifier():

model = KNeighborsClassifier(n\_neighbors=5)

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

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

return y\_pred

def mini\_batch\_gradient\_descent():

model = SGDClassifier(loss='log\_loss', max\_iter=1000, tol=1e-3)

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

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

return y\_pred

def naive\_bayes\_classifier():

model = GaussianNB()

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

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

return y\_pred

def svm():

svm\_model = SVC(kernel='linear', random\_state=42)

svm\_model.fit(X\_train, y\_train)

y\_pred = svm\_model.predict(X\_test)

return y\_pred

def evaluate(y\_pred, model):

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

precision = precision\_score(y\_test, y\_pred, average='binary')

recall = recall\_score(y\_test, y\_pred, average='binary')

f1 = f1\_score(y\_test, y\_pred, average='binary')

print(f'Model: {model}')

print(f'Accuracy: {accuracy:.4f}')

print(f'Precision: {precision:.4f}')

print(f'Recall: {recall:.4f}')

print(f'F1 Score: {f1:.4f}')

print()

def main():

y\_pred = decision\_tree()

evaluate(y\_pred, 'Decision Tree')

y\_pred = kNN\_classifier()

evaluate(y\_pred, 'kNN Classifier')

y\_pred = mini\_batch\_gradient\_descent()

evaluate(y\_pred, 'Mini-Batch Gradient Descent')

y\_pred = naive\_bayes\_classifier()

evaluate(y\_pred, 'Naive Bayes Classifier')

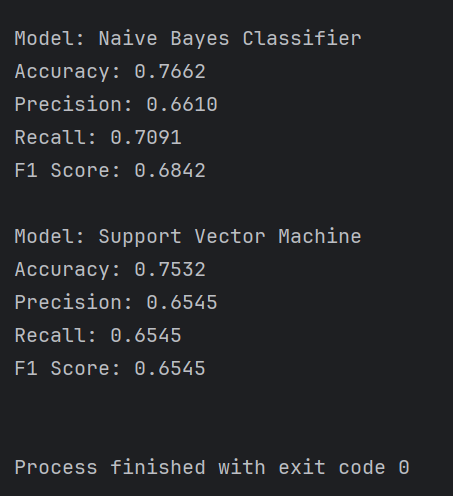
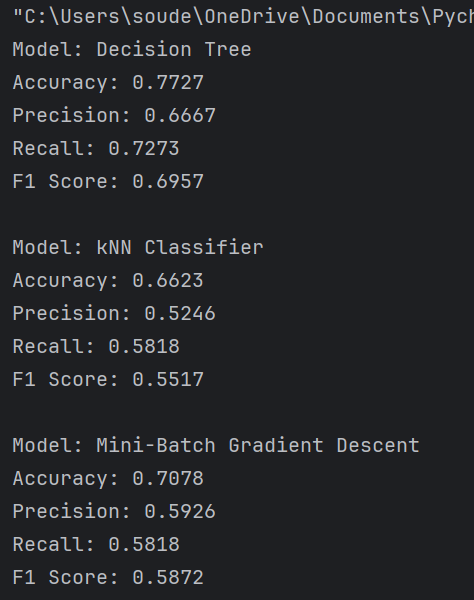
y\_pred = svm()

evaluate(y\_pred, 'Support Vector Machine')

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

main()

**Output**

*Terminal*

**Vertebral Dataset**

import pandas as pd

from sklearn.linear\_model import SGDClassifier

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

from sklearn.naive\_bayes import GaussianNB

from sklearn.neighbors import KNeighborsClassifier

from sklearn.preprocessing import LabelEncoder

from sklearn.svm import SVC

from sklearn.tree import DecisionTreeClassifier

from sklearn.metrics import accuracy\_score, precision\_score, recall\_score, f1\_score

df = pd.read\_csv('dataset/vertebral.csv')

X = df.drop('class', axis=1)

y = df['class']

le = LabelEncoder()

y = le.fit\_transform(y)

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

def decision\_tree():

model = DecisionTreeClassifier()

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

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

return y\_pred

def kNN\_classifier():

model = KNeighborsClassifier(n\_neighbors=5)

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

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

return y\_pred

def mini\_batch\_gradient\_descent():

model = SGDClassifier(loss='log\_loss', max\_iter=1000, tol=1e-3)

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

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

return y\_pred

def naive\_bayes\_classifier():

model = GaussianNB()

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

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

return y\_pred

def svm():

svm\_model = SVC(kernel='linear', random\_state=42)

svm\_model.fit(X\_train, y\_train)

y\_pred = svm\_model.predict(X\_test)

return y\_pred

def evaluate(y\_pred, model):

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

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

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

f1 = f1\_score(y\_test, y\_pred)

print(f"Model: {model}")

print(f"Accuracy: {accuracy:.4f}")

print(f"Precision: {precision:.4f}")

print(f"Recall: {recall:.4f}")

print(f"F1 Score: {f1:.4f}")

print()

def main():

y\_pred = decision\_tree()

evaluate(y\_pred, 'Decision Tree')

y\_pred = kNN\_classifier()

evaluate(y\_pred, 'kNN Classifier')

y\_pred = mini\_batch\_gradient\_descent()

evaluate(y\_pred, 'Mini-Batch Gradient Descent')

y\_pred = naive\_bayes\_classifier()

evaluate(y\_pred, 'Naive Bayes Classifier')

y\_pred = svm()

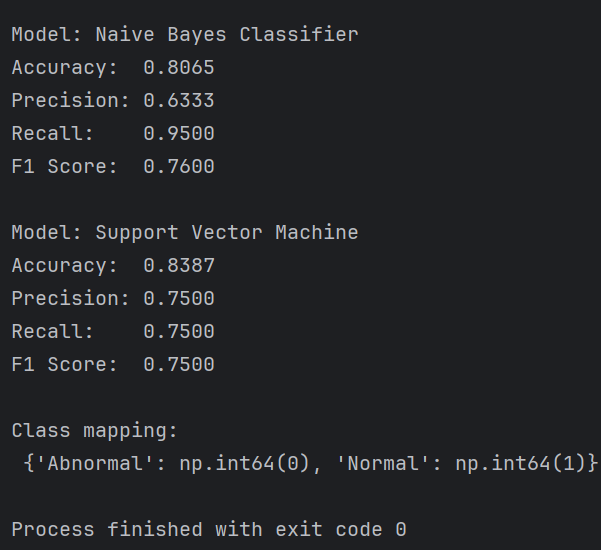
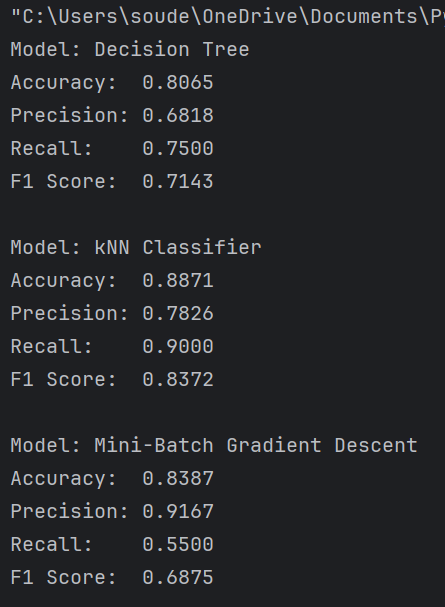
evaluate(y\_pred, 'Support Vector Machine')

print("Class mapping:", dict(zip(le.classes\_, le.transform(le.classes\_))))

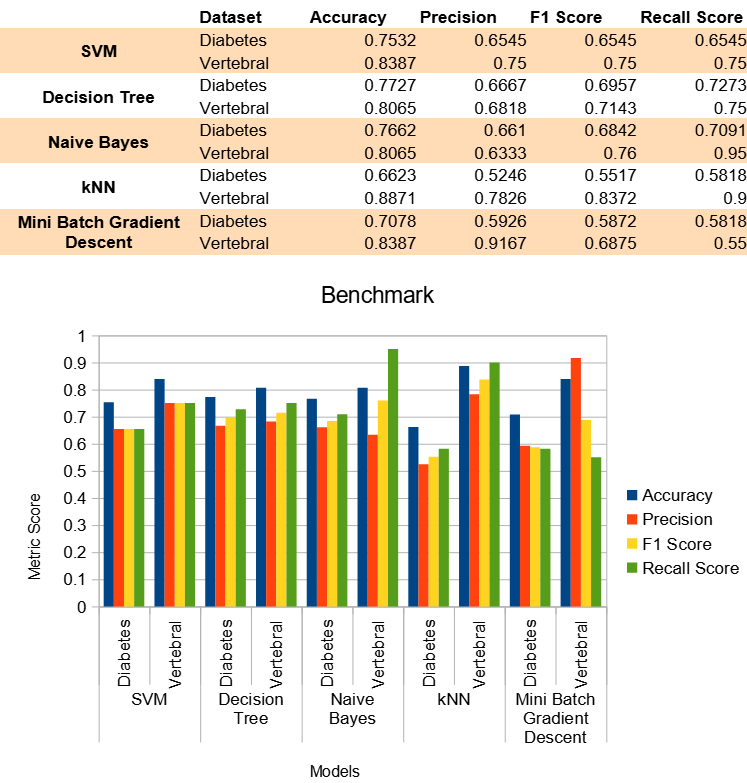
if \_\_name\_\_ == '\_\_main\_\_':

main()

**Output**

*Terminal*

**Comparison**

*Visualization*