# Experiment No :1

**Aim :**

**CO1 :**

# Procedure:

# Output:

**Result:**

The program was executed and the result was successfully obtained. Thus CO1 was obtained.

# Experiment No: 2

**Aim :**

**CO1 :**

# Procedure:

# Output:

**Result:**

The program was executed and the result was successfully obtained. Thus CO1 was obtained.

# Experiment No :3

**Aim :**

**CO1 :**

# Procedure:

# Output:

**Result:**

The program was executed and the result was successfully obtained. Thus CO1 was obtained.

# Experiment No : 4

**Aim :**

**CO1 :**

# Procedure:

from sklearn.neighbors import KNeighborsClassifier

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

from sklearn.datasets import load\_iris

from sklearn.metrics import accuracy\_score

iris=load\_iris()

x=iris.data

y=iris.target

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

knn=KNeighborsClassifier(n\_neighbors=7)

knn.fit(x\_train,y\_train)

print(knn.predict(x\_test))

v=knn.predict(x\_test)

result=accuracy\_score(y\_test,v)

print("accuracy:",result)

# Output:

# 

**Result:**

The program was executed and the result was successfully obtained. Thus CO1 was obtained.

# Experiment No: 5

**Aim :**

**CO1 :**

# Procedure:

from sklearn.neighbors import KNeighborsClassifier

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

from sklearn.datasets import load\_digits

from sklearn.metrics import accuracy\_score

digits=load\_digits()

x=digits.data

y=digits.target

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

k=KNeighborsClassifier(n\_neighbors=10)

k.fit(x\_train,y\_train)

print(k.predict(x\_test))

v=k.predict(x\_test)

r=accuracy\_score(y\_test,v)

print("accuracy:",r)

# Output:

# 

**Result:**

The program was executed and the result was successfully obtained. Thus CO1 was obtained.

# Experiment No: 6

**Aim :**

**CO1 :**

# Procedure:

from sklearn.naive\_bayes import GaussianNB

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

from sklearn.datasets import load\_iris

from sklearn.metrics import accuracy\_score

iris=load\_iris()

x=iris.data

y=iris.target

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

clf=GaussianNB()

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

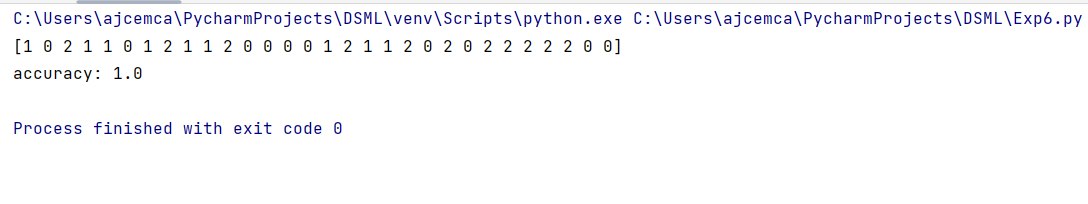
print(clf.predict(x\_test))

v=clf.predict(x\_test)

result=accuracy\_score(y\_test,v)

print("accuracy:",result)

# Output:



**Result:**

The program was executed and the result was successfully obtained. Thus CO1 was obtained.

# Experiment No: 7

**Aim :**

**CO1 :**

# Procedure:

from sklearn.naive\_bayes import GaussianNB

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

from sklearn.datasets import load\_breast\_cancer

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

bc=load\_breast\_cancer()

x=bc.data

y=bc.target

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

clf=GaussianNB()

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

print(clf.predict(x\_test))

v=clf.predict(x\_test)

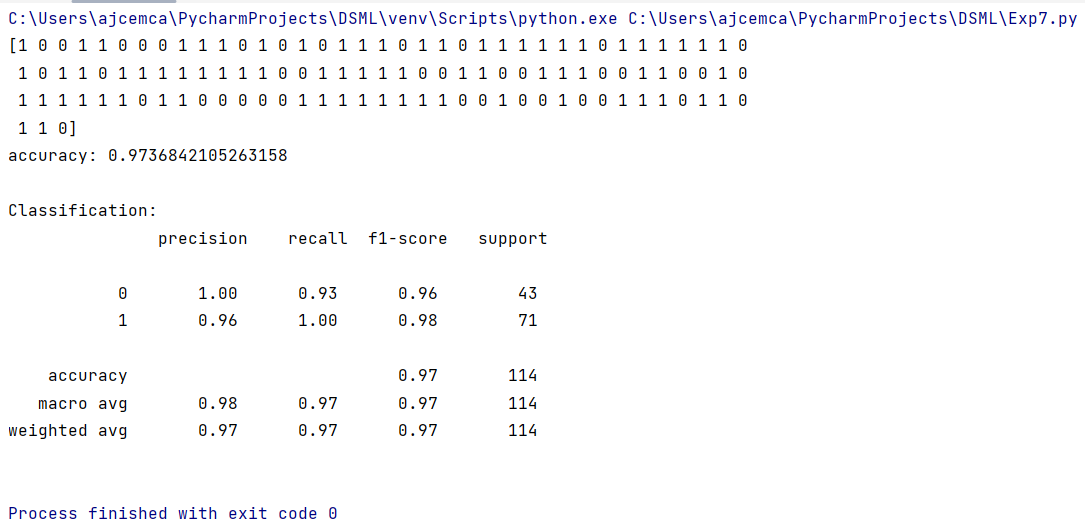
result=accuracy\_score(y\_test,v)

report=classification\_report(y\_test,v)

print("accuracy:",result)

print("\nClassification:\n",report)

# Output:



**Result:**

The program was executed and the result was successfully obtained. Thus CO1 was obtained.

# Experiment No: 8

**Aim :**

**CO1 :**

# Procedure:

from sklearn.tree import DecisionTreeClassifier

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

from sklearn.datasets import load\_iris

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

iris=load\_iris()

x=iris.data

y=iris.target

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

DT=DecisionTreeClassifier()

DT.fit(x\_train,y\_train)

print(DT.predict(x\_test))

v=DT.predict(x\_test)

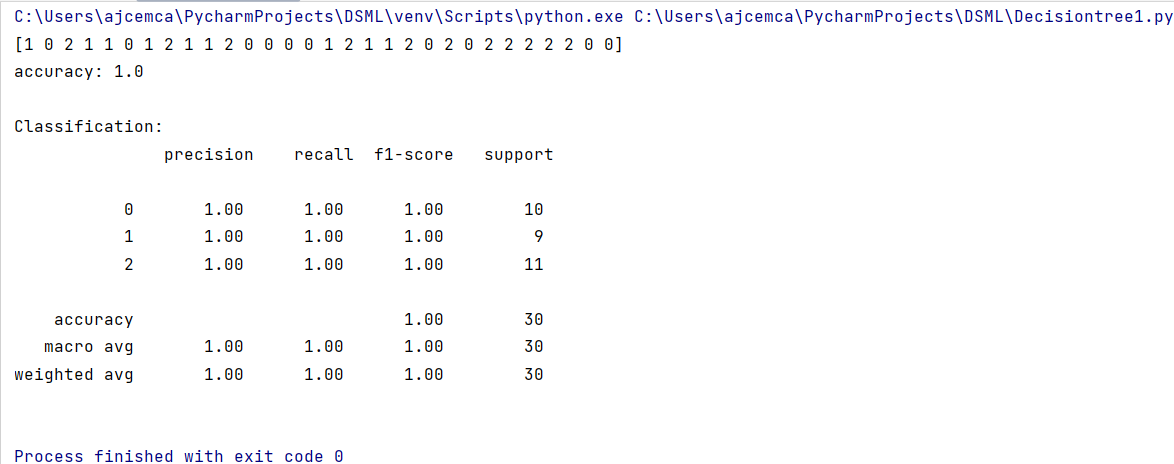
result=accuracy\_score(y\_test,v)

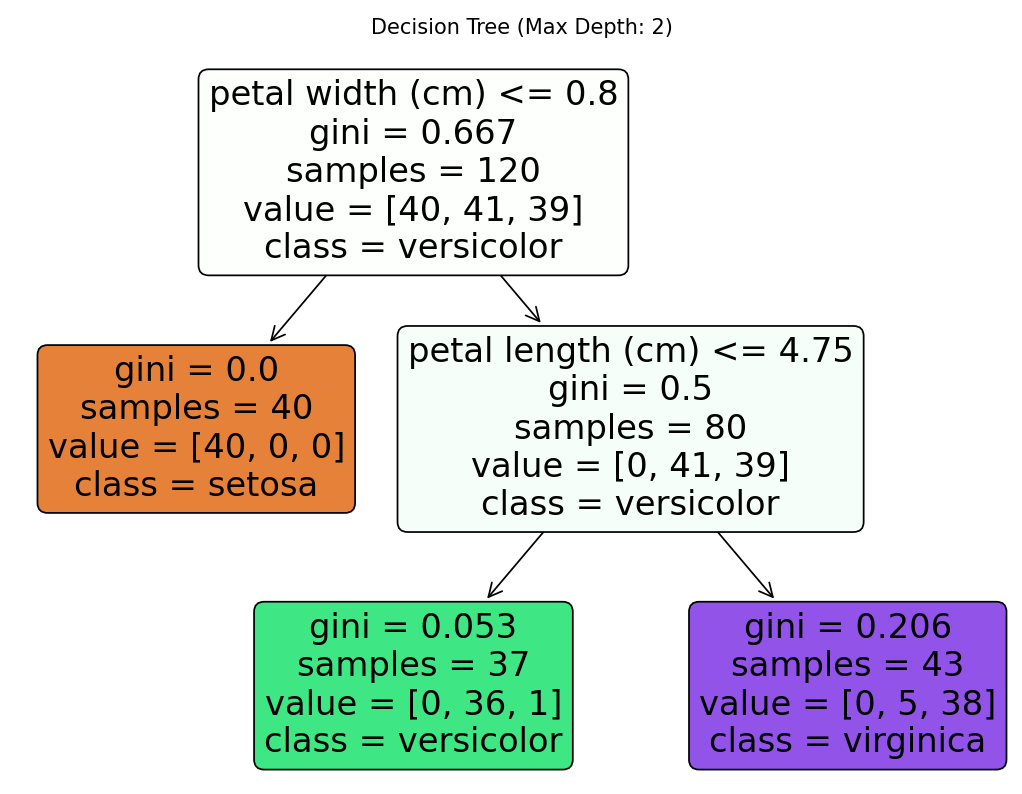
report=classification\_report(y\_test,v)

print("accuracy:",result)

print("\nClassification:\n",report)

# Output:

****

****

**Result:**

The program was executed and the result was successfully obtained. Thus CO1 was obtained.

# Experiment No: 9

**Aim :**

**CO1 :**

# Procedure:

from sklearn.tree import DecisionTreeClassifier

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

from sklearn.datasets import load\_breast\_cancer

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

bc=load\_breast\_cancer()

x=bc.data

y=bc.target

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

dt=DecisionTreeClassifier()

dt.fit(x\_train,y\_train)

print(dt.predict(x\_test))

v=dt.predict(x\_test)

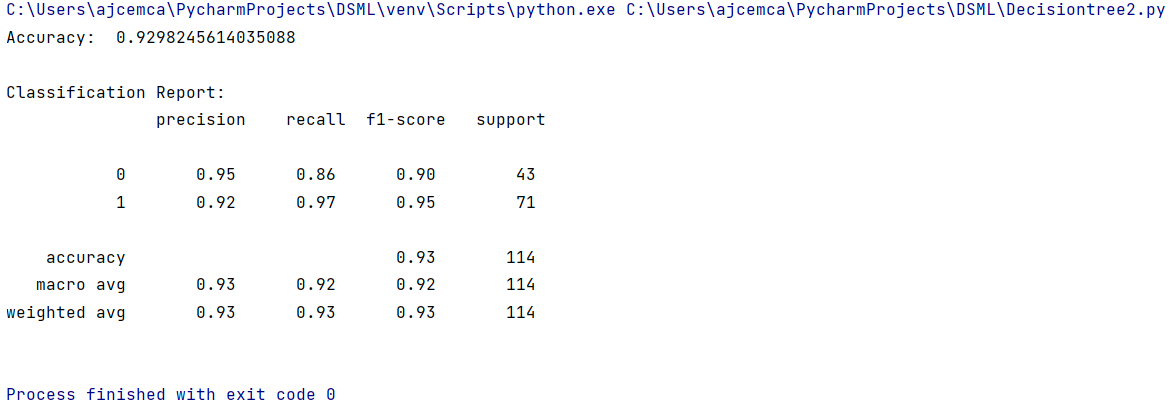
result=accuracy\_score(y\_test,v)

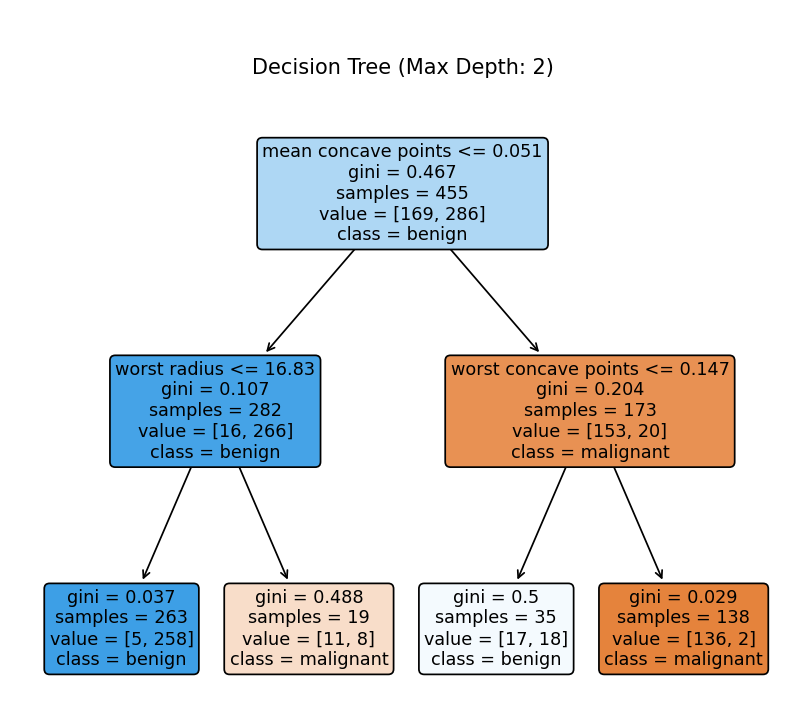
report=classification\_report(y\_test,v)

print("accuracy:",result)

print("\nClassification:\n",report)

# Output:

****

****

**Result:**

The program was executed and the result was successfully obtained. Thus CO1 was obtained.

# Experiment No: 10

**Aim :**

**CO1 :**

# Procedure:

# import pandas as pd

# from sklearn.datasets import fetch\_california\_housing

# from sklearn.linear\_model import LinearRegression

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

# from sklearn.metrics import mean\_squared\_error

# ch=fetch\_california\_housing()

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

# df['target']=ch.target

# 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)

# lr=LinearRegression()

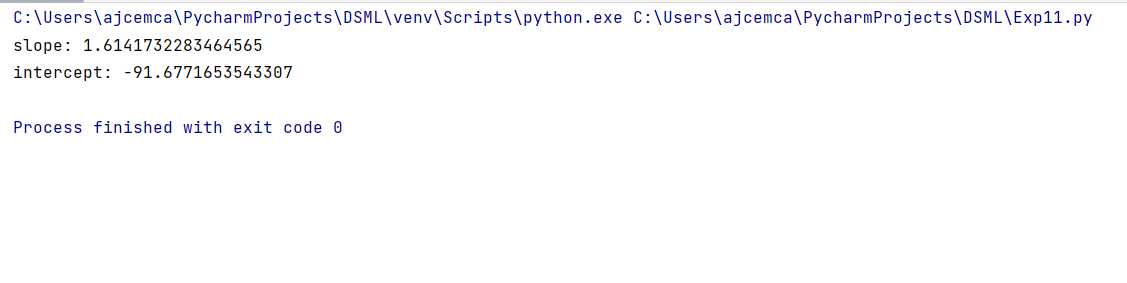
# lr.fit(x\_train,y\_train)

# v=lr.predict(x\_test)

# result=mean\_squared\_error(y\_test,v)

# print("Mean:",result)

# Output:



**Result:**

The program was executed and the result was successfully obtained. Thus CO1 was obtained.

# Experiment No: 11

**Aim :**

**CO1 :**

# Procedure:

import pandas as pd

import matplotlib.pyplot as plt

from sklearn.metrics import r2\_score

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

from sklearn.linear\_model import LinearRegression

data = pd.read\_csv('Salary\_Data.csv')

x = data['YearsExperience'].values.reshape(-1, 1)

y = data['Salary'].values

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

slr = LinearRegression()

slr.fit(x\_train, y\_train)

y\_pred = slr.predict(x\_test)

r2 = r2\_score(y\_test, y\_pred)

print("R-squared :", r2)

plt.scatter(x\_test,y\_test,color='black',label ='Data Points')

plt.plot(x\_test,y\_pred,color='blue',linewidth=3 ,label='Regression line')

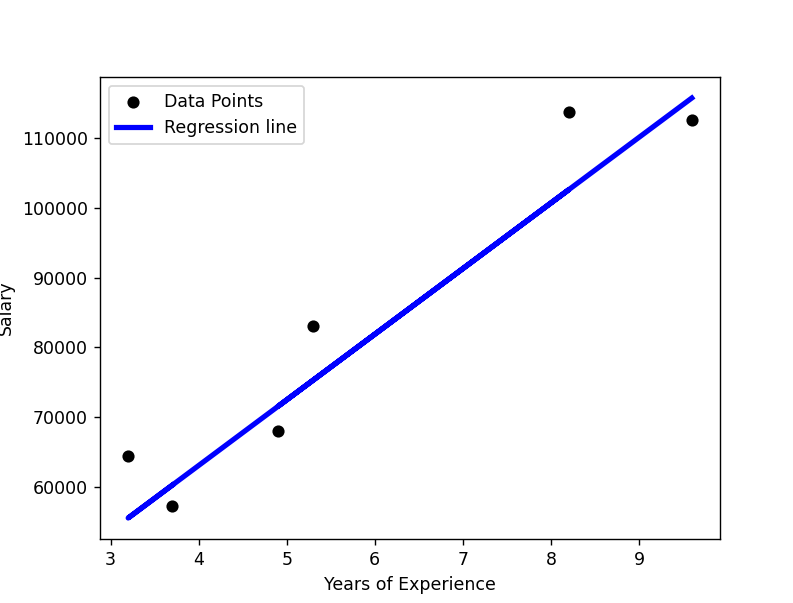
plt.xlabel('Years of Experience')

plt.ylabel('Salary')

plt.legend()

plt.show()

# Output:

****

**Result:**

The program was executed and the result was successfully obtained. Thus CO1 was obtained.

# Experiment No: 12

**Aim :**

**CO1 :**

# Procedure:

import pandas as pd

from sklearn.datasets import fetch\_california\_housing

from sklearn.linear\_model import LinearRegression

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

from sklearn.metrics import mean\_squared\_error

ch=fetch\_california\_housing()

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

df['target']=ch.target

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)

lr=LinearRegression()

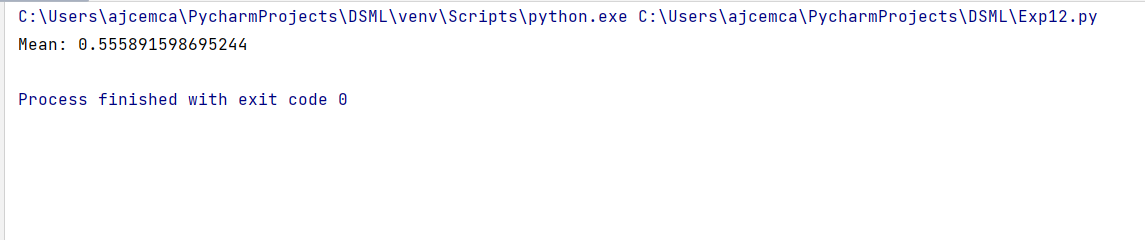
lr.fit(x\_train,y\_train)

v=lr.predict(x\_test)

result=mean\_squared\_error(y\_test,v)

print("Mean:",result)

# Output:



**Result:**

The program was executed and the result was successfully obtained. Thus CO1 was obtained.

# Experiment No: 13

**Aim :**

**CO1 :**

Use different packages and frameworks to implement text classification using SVM and clustering using k-means

# Procedure:

from sklearn.datasets import load\_iris

from sklearn.cluster import KMeans

import matplotlib.pyplot as plt

iris = load\_iris()

x = iris.data

y = iris.target

kmeans = KMeans(n\_clusters=3, random\_state=42)

kmeans.fit(x)

cluster\_labels = kmeans.labels\_

print(cluster\_labels)

centroids = kmeans.cluster\_centers\_

print(centroids)

plt.scatter(x[:, 0], x[:, 1], c=cluster\_labels, cmap='viridis', marker='o', edgecolor='black')

plt.scatter(centroids[:, 0], centroids[:, 1], marker='\*', s=200, c='red', label='centroid')

plt.xlabel(iris.feature\_names[0])

plt.ylabel(iris.feature\_names[1])

plt.title("Kmeans clustering of Dataset Iris")

plt.legend()

plt.show()

# 

# 

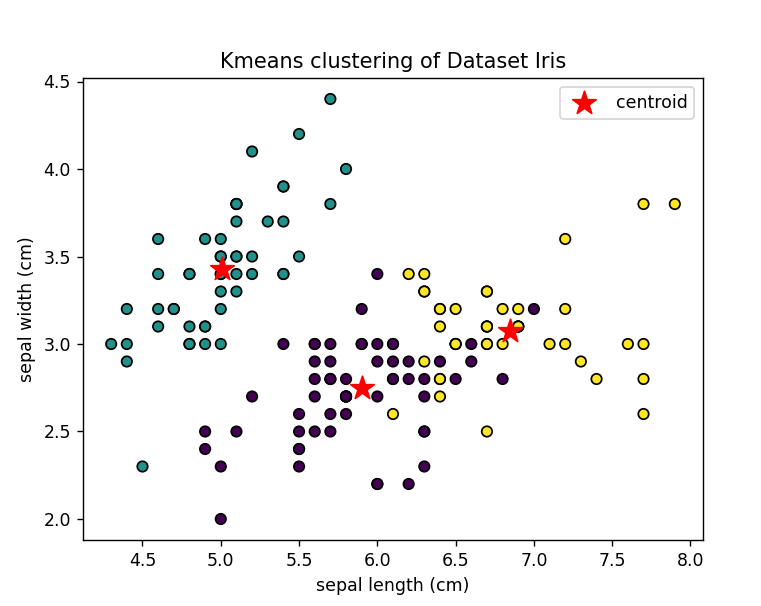
# 

# 

# 

# 

# Output:



**Result:**

The program was executed and the result was successfully obtained. Thus CO1 was obtained.

# Experiment No: 14

**Aim :**

**CO1 :**

Use different packages and frameworks to implement text classification using SVM and clustering using k-means

# Procedure:

# from sklearn.datasets import load\_breast\_cancer

# from sklearn.cluster import KMeans

# import matplotlib.pyplot as plt

# bc = load\_breast\_cancer()

# x = bc.data

# y = bc.target

# kmeans = KMeans(n\_clusters=3, random\_state=42)

# kmeans.fit(x)

# cluster\_labels = kmeans.labels\_

# print(cluster\_labels)

# centroids = kmeans.cluster\_centers\_

# print(centroids)

# plt.scatter(x[:, 0], x[:, 1], c=cluster\_labels, cmap='viridis', marker='o', edgecolor='black')

# plt.scatter(centroids[:, 0], centroids[:, 1], marker='\*', s=200, c='red', label='centroid')

# plt.xlabel(bc.feature\_names[0])

# plt.ylabel(bc.feature\_names[1])

# plt.title("Kmeans clustering of breast cancer Dataset ")

# plt.legend()

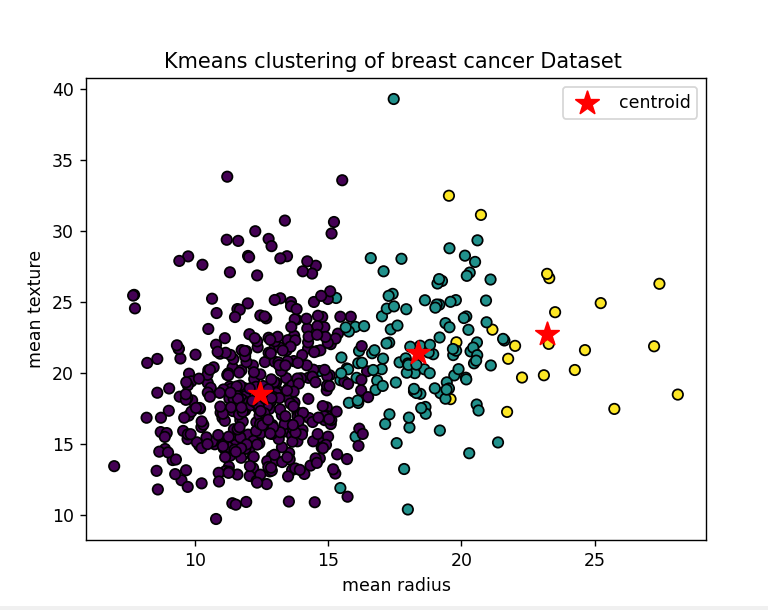
# plt.show()

# 

# 

# 

# Output:



**Result:**

The program was executed and the result was successfully obtained. Thus CO1 was obtained.

# Experiment No :15

**Aim :** Program to implement text classification using support vector machine

**CO3 :**

# Procedure:

from sklearn.datasets import fetch\_20newsgroups

from sklearn.feature\_extraction.text import TfidfVectorizer

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

from sklearn.svm import SVC

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

categories=['alt.atheism','soc.religion.christian','comp.graphics','sci.med']

twenty\_train=fetch\_20newsgroups(subset="train",categories=categories,shuffle=True,random\_state=42)

vectorizer=TfidfVectorizer()

x\_train\_tfidf=vectorizer.fit\_transform(twenty\_train.data)

y\_train=twenty\_train.target

x\_train,x\_test,y\_train,y\_test=train\_test\_split(x\_train\_tfidf,y\_train,test\_size=0.3,random\_state=42)

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

svm\_classifier.fit(x\_train,y\_train)

pred=svm\_classifier.predict(x\_test)

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

class\_report=classification\_report(y\_test,pred,target\_names=twenty\_train.target\_names)

print("Accuracy Score",accuracy\_score)

print("Classification Report\n",class\_report)

new\_data = [

"Computer graphics"

]

x\_new\_tfidf=vectorizer.transform(new\_data)

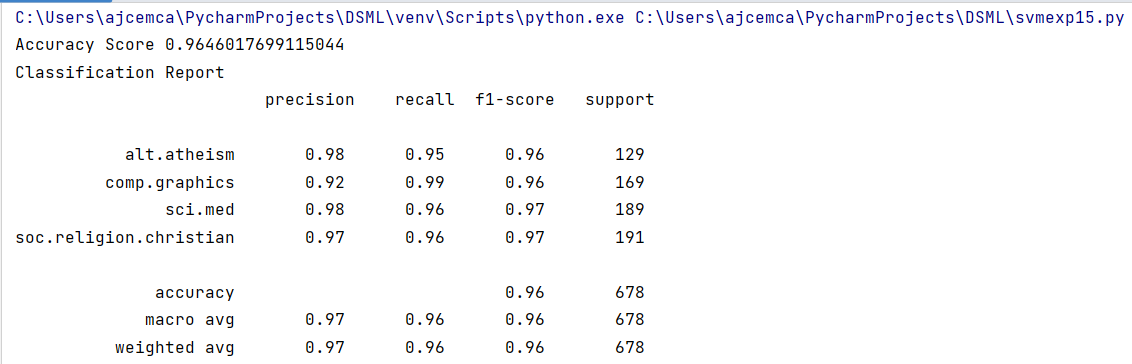
new\_pred=svm\_classifier.predict(x\_new\_tfidf)

for i,text in enumerate(new\_data):

predicted\_category=twenty\_train.target\_names[new\_pred[i]]

print("Predicted Category:",predicted\_category)

# Output:

****

**Result:**

The program was executed and the result was successfully obtained. Thus CO1 was obtained.