Experiment No:1

Aim: Program to perform matrix operations. Use numpy as the python library and perform the operation using built in functions.

CO1: Use different python packages to perform numerical calculations, statistical computations and data visualization.

Procedure:

```
import numpy as np
rows1 = int(input("Enter the number of rows for matrix 1: "))
cols1 = int(input("Enter the number of columns for matrix 1: "))
matrix1=np.empty((rows1,cols1), dtype=float)
print("Enter the value of the first matrix:")
for i in range(rows1):
for j in range(cols1):
matrix1[i][j]=float(input(f"Enter element at position ({i + 1}, {j + 1}): "))
rows2 = int(input("Enter the number of rows for matrix 2: "))
cols2 = int(input("Enter the number of columns for matrix 2: "))
matrix2=np.empty((rows2,cols2), dtype=float)
print("Enter the value of the second matrix:")
for i in range(rows2):
for j in range(cols2):
matrix2[i][j]=float(input(f"Enter element at position ({i + 1}, {j + 1}): "))
print("\nMatrix 1:")
print(matrix1)
print("\nMatrix 2:")
print(matrix2)
sum=np.add(matrix1,matrix2)
print("sum:", sum)
prod=np.multiply(matrix1,matrix2)
print("product:", prod)
sub=np.subtract(matrix1,matrix2)
```

print("subtract:", sub)

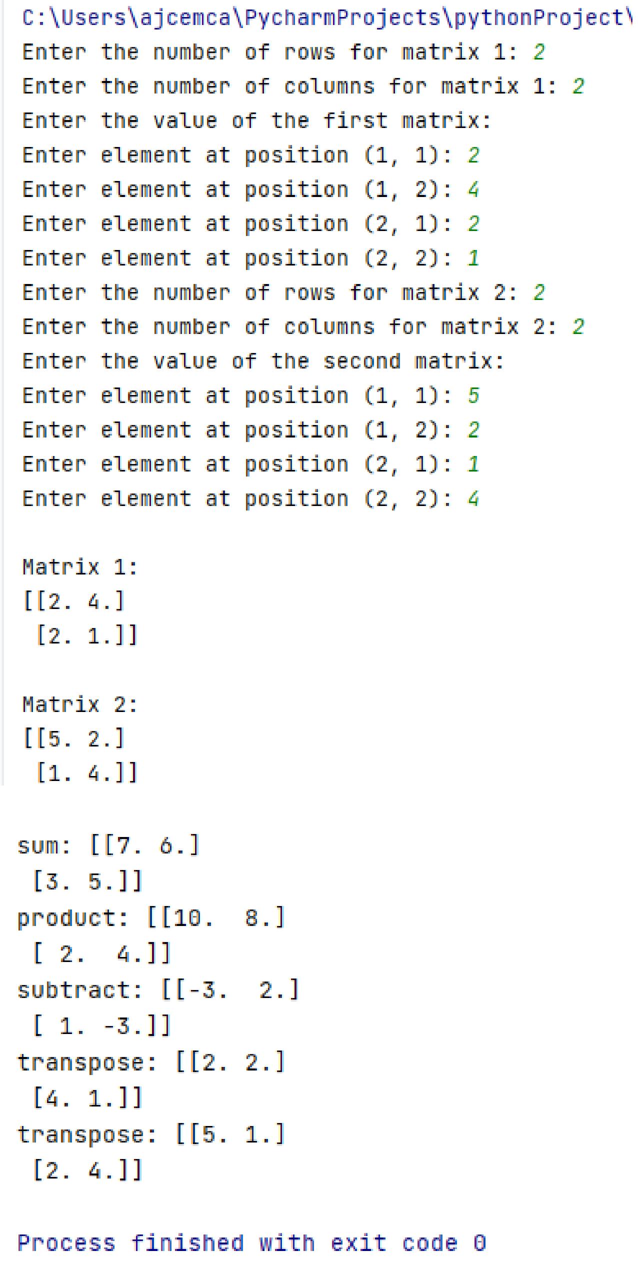
trp1=np.transpose(matrix1)

trp2=np.transpose(matrix2)

print("transpose:", trp1)

print("transpose:", trp2)

Output Screenshot:



```
C:\Users\ajcemca\PycharmProjects\pythonProject\
Enter the number of rows for matrix 1: 2
Enter the number of columns for matrix 1: 2
Enter the value of the first matrix:
Enter element at position (1, 1): 2
Enter element at position (1, 2): 4
Enter element at position (2, 1): 2
Enter element at position (2, 2): 1
Enter the number of rows for matrix 2: 2
Enter the number of columns for matrix 2: 2
Enter the value of the second matrix:
Enter element at position (1, 1): 5
Enter element at position (1, 2): 2
Enter element at position (2, 1): 1
Enter element at position (2, 2): 4

Matrix 1:
[[2. 4.]
 [2. 1.]]

Matrix 2:
[[5. 2.]
 [1. 4.]]

sum: [[7. 6.]
 [3. 5.]]
product: [[10.  8.]
 [ 2.  4.]]
subtract: [[-3.  2.]
 [ 1. -3.]]
transpose: [[2. 2.]
 [4. 1.]]
transpose: [[5. 1.]
 [2. 4.]]

Process finished with exit code 0
```

Result: The program was executed successfully and the output was obtained. Thus, CO1 has been attained.

Experiment No:2

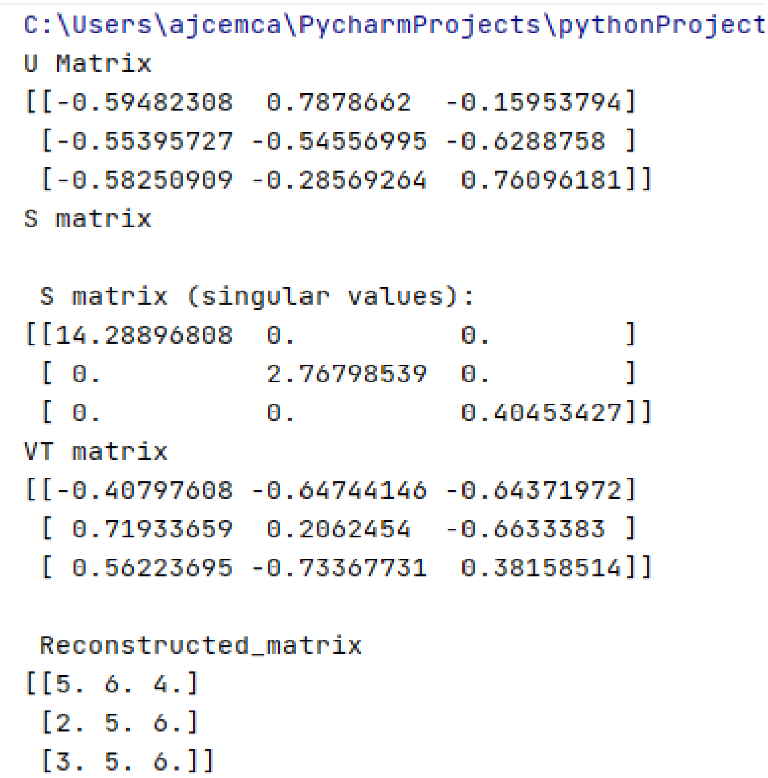
Aim: Program to perform single value decomposition using numpy.

CO1: Use different python packages to perform numerical calculations, statistical computations and data visualization.

Procedure:

```
import numpy as np
matrix = np.array([[5, 6, 4], [ 2 , 5, 6], [ 3 , 5 , 6]])
u, s, vt = np.linalg.svd(matrix)
print("u matrix")
print(u)
print("s matrix:")
print(np.diag(s))
print("vt:")
print(vt)
reconstructed_matrix = np.dot(u, np.dot(np.diag(s), vt))
print("Original Matrix:")
print(reconstructed_matrix)
```

Output Screenshot:



```
C:\Users\ajcemca\PycharmProjects\pythonProject
U Matrix
[[-0.59482308  0.7878662  -0.15953794]
 [-0.55395727 -0.54556995 -0.6288758 ]
 [-0.58250909 -0.28569264  0.76096181]]
S matrix

 S matrix (singular values):
[[14.28896808  0.          0.        ]
 [ 0.          2.76798539  0.        ]
 [ 0.          0.          0.40453427]]
VT matrix
[[-0.40797608 -0.64744146 -0.64371972]
 [ 0.71933659  0.2062454  -0.6633383 ]
 [ 0.56223695 -0.73367731  0.38158514]]

 Reconstructed_matrix
[[5. 6. 4.]
 [2. 5. 6.]
 [3. 5. 6.]]
```

Result: The program was executed successfully and the output was obtained. Thus, CO1 has been attained.

Experiment No:3

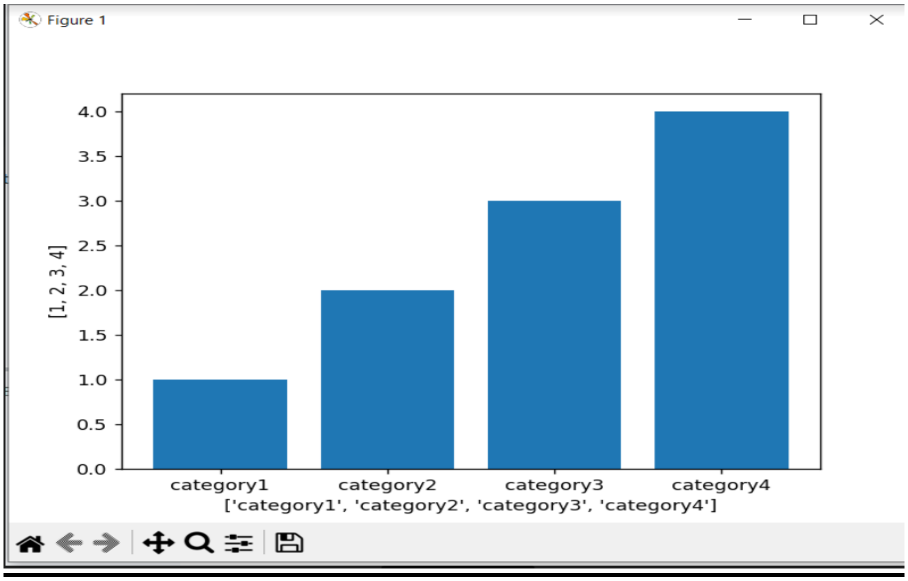
Aim: Program to perform data visualisation using python library matplotlib

CO1: Use different python packages to perform numerical calculations, statistical computations and data visualization.

Procedure:

```
import matplotlib.pyplot as plt

category = ["Category 1", "Category 2", "Category 3"]

values = [10, 15, 7]

plt.bar(category, values, color='skyblue')

plt.xlabel('Categories')

plt.ylabel('Values')

plt.title('Bar Chart Example')

plt.show()
```

Output Screenshot:



Result: The program was executed successfully and the output was obtained. Thus, CO1 has been attained.

Experiment No:4

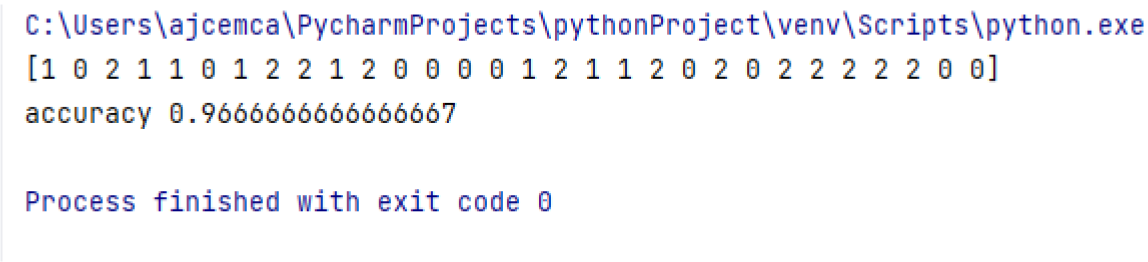
Aim: Program to implement KNN classification using any standard dataset available in the public domain and find the accuracy of algorithm. (Iris Dataset)

CO2: Use different packages and frameworks to implement regression and classification algorithms.

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)  # trying to fit maximum data
print(knn.predict(x_test))
v=knn.predict(x_test)   # result of model prediction
result=accuracy_score(y_test,v) # comparing both prediction
print("accuracy",result)
```

Output Screenshot:



```
C:\Users\ajcemca\PycharmProjects\pythonProject\venv\Scripts\python.exe
[1 0 2 1 1 0 1 2 2 1 2 0 0 0 0 1 2 1 1 2 0 2 0 2 2 2 2 2 0 0]
accuracy 0.9666666666666667

Process finished with exit code 0
```

Result: The program was executed successfully and the output was obtained. Thus, CO2 has been attained.

Experiment No:5

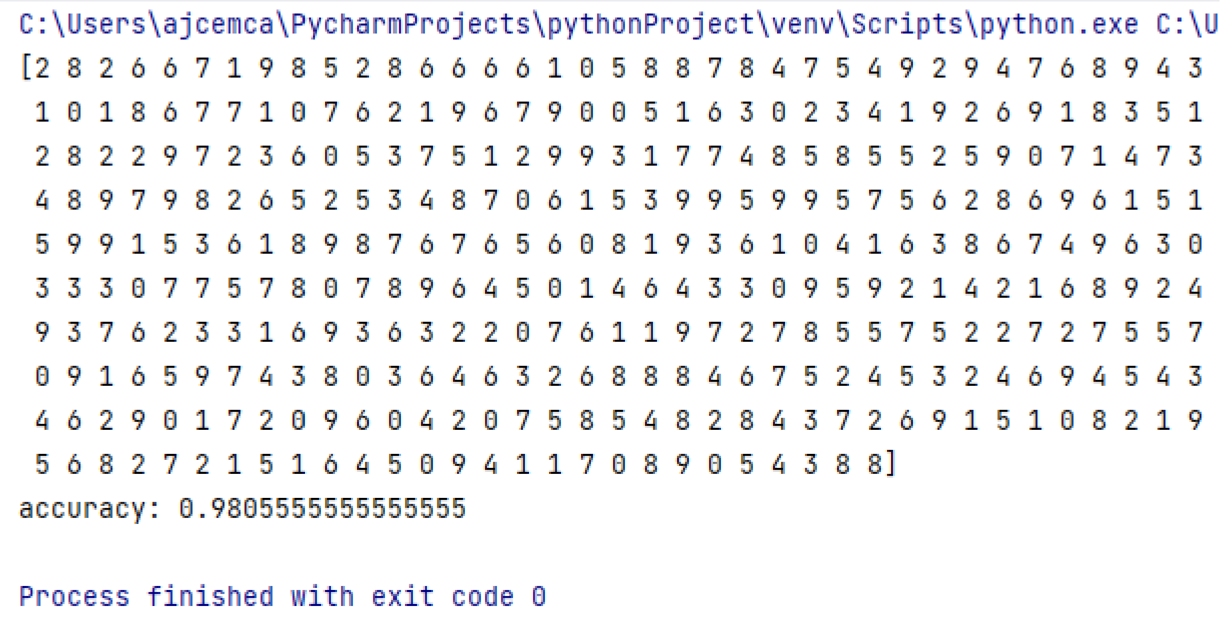
Aim: Program to implement KNN classification using any standard dataset available in the public domain and find the accuracy of algorithm. (Load Digits)

CO2: Use different packages and frameworks to implement regression and classification algorithms.

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)

knn=KNeighborsClassifier(n_neighbors=9)

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 Screenshot:



```
C:\Users\ajcemca\PycharmProjects\pythonProject\venv\Scripts\python.exe C:\U
[2 8 2 6 6 7 1 9 8 5 2 8 6 6 6 6 1 0 5 8 8 7 8 4 7 5 4 9 2 9 4 7 6 8 9 4 3
 1 0 1 8 6 7 7 1 0 7 6 2 1 9 6 7 9 0 0 5 1 6 3 0 2 3 4 1 9 2 6 9 1 8 3 5 1
 2 8 2 2 9 7 2 3 6 0 5 3 7 5 1 2 9 9 3 1 7 7 4 8 5 8 5 5 2 5 9 0 7 1 4 7 3
 4 8 9 7 9 8 2 6 5 2 5 3 4 8 7 0 6 1 5 3 9 9 5 9 9 5 7 5 6 2 8 6 9 6 1 5 1
 5 9 9 1 5 3 6 1 8 9 8 7 6 7 6 5 6 0 8 1 9 3 6 1 0 4 1 6 3 8 6 7 4 9 6 3 0
 3 3 3 0 7 7 5 7 8 0 7 8 9 6 4 5 0 1 4 6 4 3 3 0 9 5 9 2 1 4 2 1 6 8 9 2 4
 9 3 7 6 2 3 3 1 6 9 3 6 3 2 2 0 7 6 1 1 9 7 2 7 8 5 5 7 5 2 2 7 2 7 5 5 7
 0 9 1 6 5 9 7 4 3 8 0 3 6 4 6 3 2 6 8 8 8 4 6 7 5 2 4 5 3 2 4 6 9 4 5 4 3
 4 6 2 9 0 1 7 2 0 9 6 0 4 2 0 7 5 8 5 4 8 2 8 4 3 7 2 6 9 1 5 1 0 8 2 1 9
 5 6 8 2 7 2 1 5 1 6 4 5 0 9 4 1 1 7 0 8 9 0 5 4 3 8 8]
accuracy: 0.9805555555555555

Process finished with exit code 0
```

Result: The program was executed successfully and the output was obtained. Thus, CO2 has been attained.

Experiment No:6

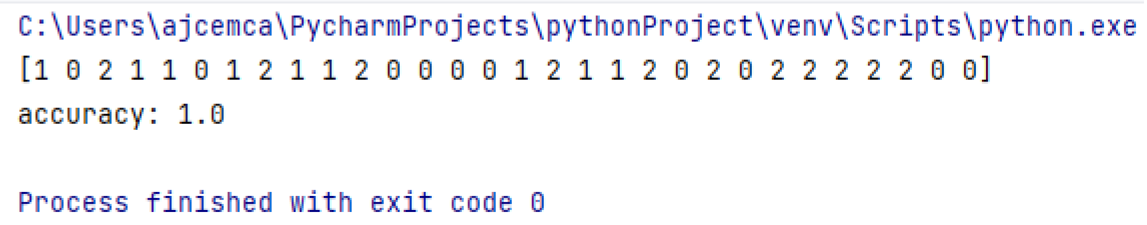
Aim: Program to implement Naïve Bayes Algorithm using any standard dataset available in the public domain and find the accuracy of algorithm (Iris Dataset).

CO2: Use different packages and frameworks to implement regression and classification algorithms.

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)
nb = GaussianNB()
nb.fit(x_train,y_train)
print(nb.predict(x_test))
v=nb.predict(x_test)
result=accuracy_score(y_test,v)
print("Accuracy::",result)
```

Output Screenshot:



```
C:\Users\ajcemca\PycharmProjects\pythonProject\venv\Scripts\python.exe
[1 0 2 1 1 0 1 2 1 1 2 0 0 0 0 1 2 1 1 2 0 2 0 2 2 2 2 2 0 0]
accuracy: 1.0

Process finished with exit code 0
```

Result: The program was executed successfully and the output was obtained. Thus, CO2 has been attained.

Experiment No:7

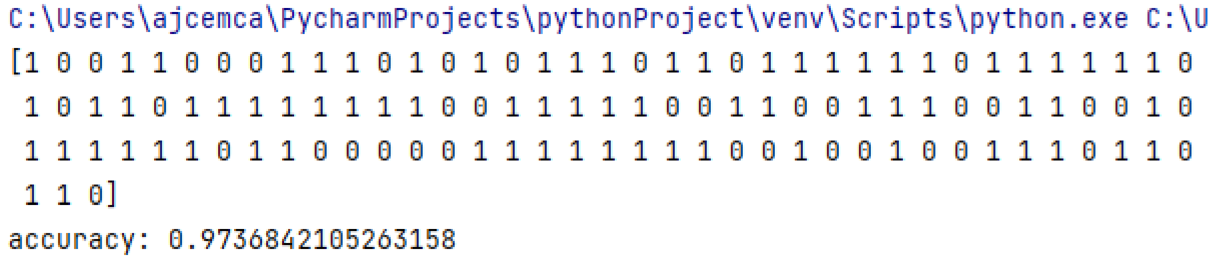
Aim: Program to implement Naïve Bayes Algorithm using any standard dataset available in the public domain and find the accuracy of algorithm (Breast Cancer Dataset)

CO2: Use different packages and frameworks to implement regression and classification algorithms.

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
from sklearn.datasets import load_breast_cancer
data=load_breast_cancer()
x=data.data
y=data.target
x_train,x_test,y_train,y_test = train_test_split(x,y,test_size=0.2, random_state=42)
nb=GaussianNB()
nb.fit(x_train,y_train)
print(nb.predict(x_test))
v=nb.predict(x_test)
result=accuracy_score(y_test,v)
print("Accuracy::",result)
```

Output Screenshot:



```
C:\Users\ajcemca\PycharmProjects\pythonProject\venv\Scripts\python.exe C:\U
[1 0 0 1 1 0 0 0 1 1 1 0 1 0 1 0 1 1 1 0 1 1 0 1 1 1 1 1 1 0 1 1 1 1 1 1 0
 1 0 1 1 0 1 1 1 1 1 1 1 1 0 0 1 1 1 1 1 0 0 1 1 0 0 1 1 1 0 0 1 1 0 0 1 0
 1 1 1 1 1 1 0 1 1 0 0 0 0 0 1 1 1 1 1 1 1 1 0 0 1 0 0 1 0 0 1 1 1 0 1 1 0
 1 1 0]
accuracy: 0.9736842105263158
```

Result: The program was executed successfully and the output was obtained. Thus, CO2 has been attained.

Experiment No:8

Aim: Give one dimensional dataset represented with numpy array. Write a program to calculate slope and intercept

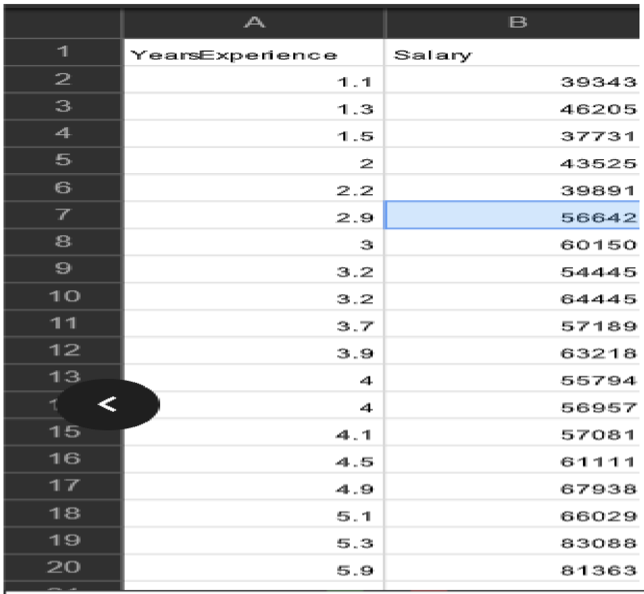
CO2: Use different packages and frameworks to implement regression and classification algorithms.

Procedure:

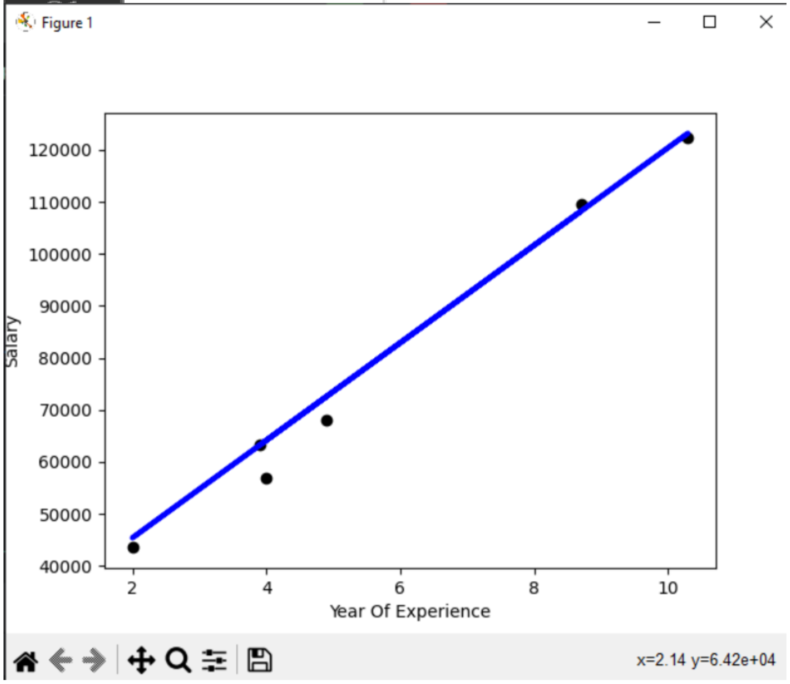
```
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import r2_score
data= pd.read_csv('Salary_Data.csv')
x=data['YearsExperience'].values.reshape(-1,1)
y=data['Salary']
x_train,x_test,y_train,y_test= train_test_split(x, y, test_size=0.2, random_state=47)
sr=LinearRegression()
sr.fit(x_train,y_train)
y_pred=sr.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('Year Of Experience')
plt.ylabel('Salary')
```

plt.show()

Output Screenshot:



	A	B
1	YearsExperience	Salary
2	1.1	39343
3	1.3	46205
4	1.5	37731
5	2	43525
6	2.2	39891
7	2.9	56642
8	3	60150
9	3.2	54445
10	3.2	64445
11	3.7	57189
12	3.9	63218
13	4	55794
14	4	56957
15	4.1	57081
16	4.5	61111
17	4.9	67938
18	5.1	66029
19	5.3	83088
20	5.9	81363



Result: The program was executed successfully and the output was obtained. Thus, CO2 has been attained.

Experiment No:9

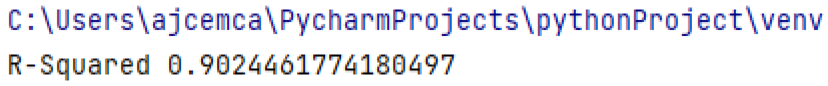
Aim: Program to implement simple linear regression using any standard dataset available in the public domain and find r2 score.

CO2: Use different packages and frameworks to implement regression and classification algorithms.

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 Screenshot:



```
C:\Users\ajcemca\PycharmProjects\pythonProject\venv
R-Squared 0.9024461774180497
```

Result: The program was executed successfully and the output was obtained. Thus, CO2 has been attained.

Experiment No:10

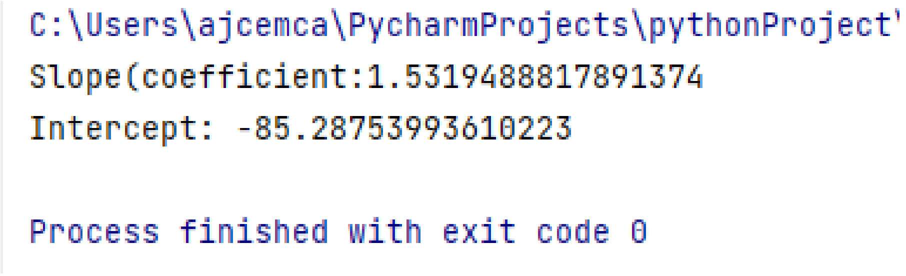
Aim: Program to implement linear and multiple regression techniques using any standard dataset available in the public domain and evaluate its performance

CO2: Use different packages and frameworks to implement regression and classification algorithms.

Procedure:

```
import numpy as np
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error
x=np.array([64,75,68,73,78,82,76,85,71,88]).reshape(-1,1)
y=np.array([17,27,15,24,39,44,30,48,19,47])
model=LinearRegression()
model.fit(x,y)
b=model.coef_[0]
a=model.intercept_
print(f"slope : {b}")
print(f"intercept : {a}")
```

Output Screenshot:



```
C:\Users\ajcemca\PycharmProjects\pythonProject'
Slope(coefficient:1.5319488817891374
Intercept: -85.28753993610223

Process finished with exit code 0
```

Result: The program was executed successfully and the output was obtained. Thus, CO2 has been attained.

Experiment No:11

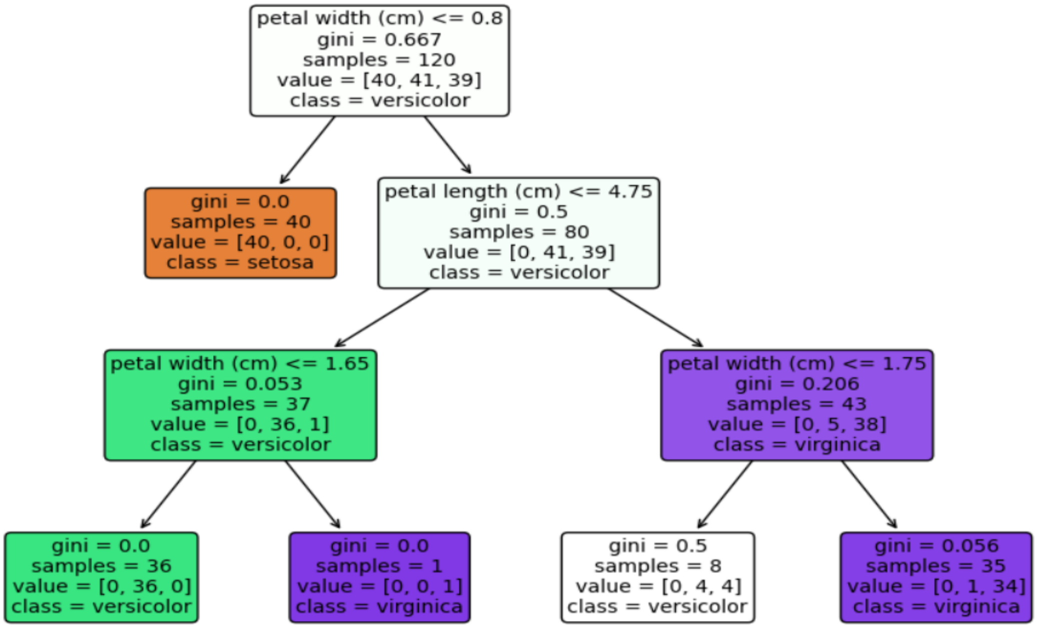
Aim: Program to implement decision trees using any standard dataset available in the public domain and find the accuracy of the algorithm.(Iris Dataset)

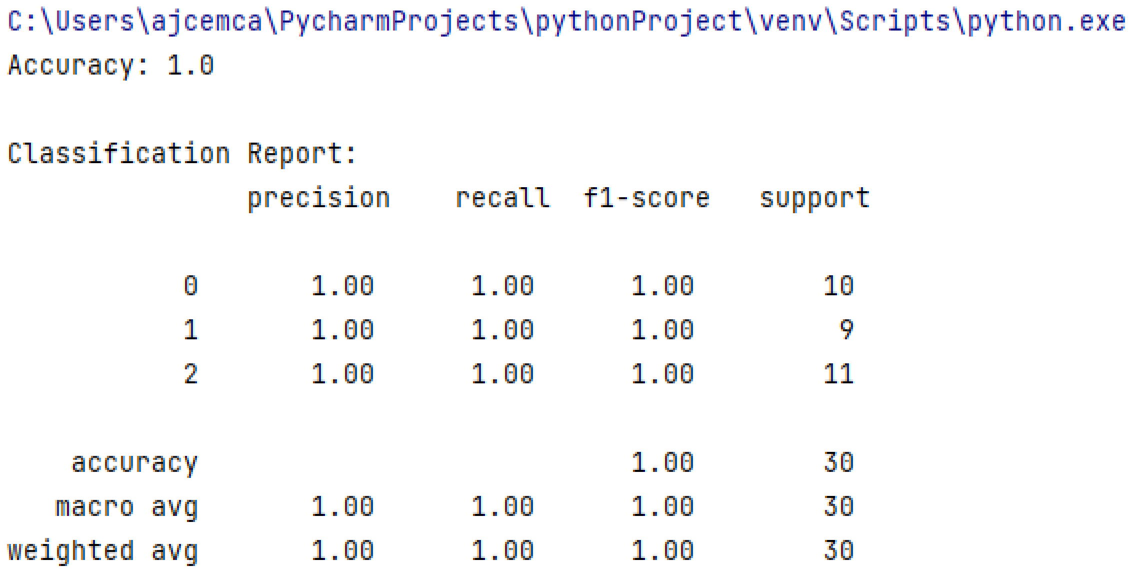
CO3: Use different packages and frameworks to implement text classification using SVM and clustering using k-means.

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
from sklearn.tree import plot_tree
import matplotlib.pyplot as plt
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(max_depth=3)
dt.fit(x_train,y_train)
print(dt.predict(x_test))
v=dt.predict(x_test)
report=classification_report(y_test,v)
result=accuracy_score(y_test,v)
print("Accuracy::",result)
print("\nClassification Report::\n",report)
plt.figure("DECISION TREE",figsize=(10, 10))
plot_tree(dt,filled=True,feature_names=iris.feature_names,class_names=iris.target_names,
rounded=True)
plt.show()
```

Output Screenshot:





```
C:\Users\ajcemca\PycharmProjects\pythonProject\venv\Scripts\python.exe
Accuracy: 1.0

Classification Report:
               precision    recall  f1-score   support

           0       1.00      1.00      1.00        10
           1       1.00      1.00      1.00         9
           2       1.00      1.00      1.00        11

    accuracy                           1.00        30
   macro avg       1.00      1.00      1.00        30
weighted avg       1.00      1.00      1.00        30
```

Result: The program was executed successfully and the output was obtained. Thus, CO3 has been attained.

Experiment No:12

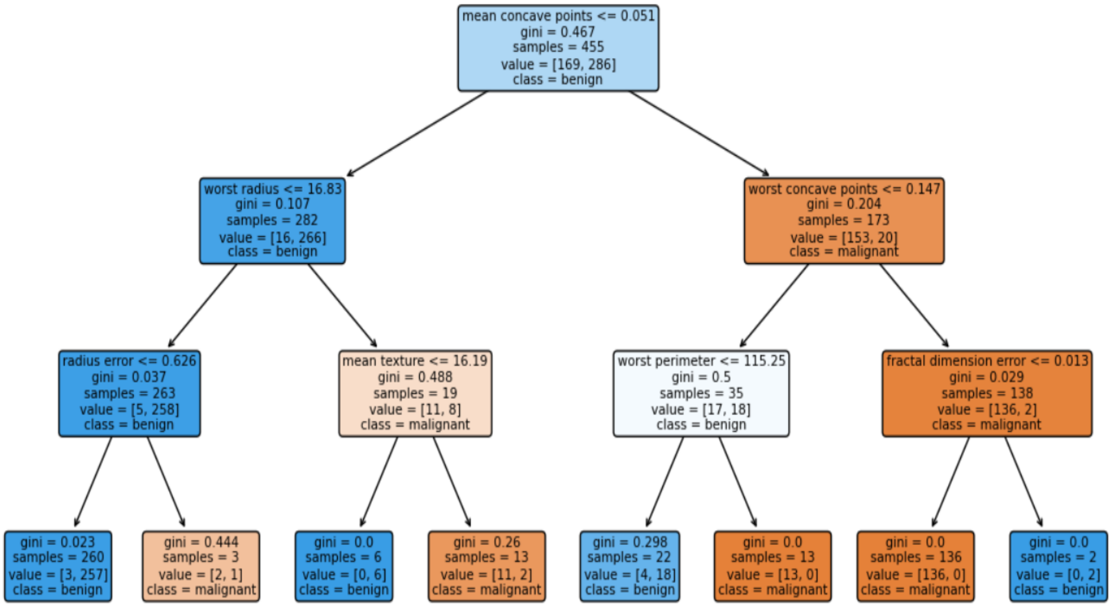
Aim: Program to implement decision trees using any standard dataset available in the public domain and find the accuracy of the algorithm. (Breast Cancer Dataset)

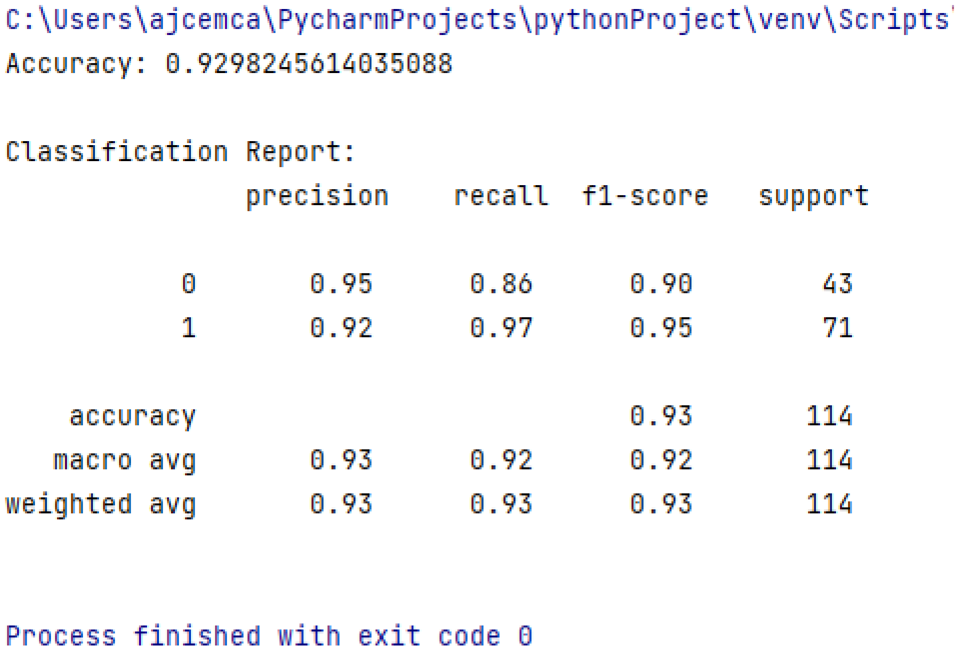
CO3: Use different packages and frameworks to implement text classification using SVM and clustering using k-means.

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
from sklearn.tree import plot_tree
import matplotlib.pyplot as plt
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(max_depth=3)
dt.fit(x_train, y_train)
v = dt.predict(x_test)
report = classification_report(y_test, v)
result = accuracy_score(y_test, v)
print("Accuracy:", result)
print("\nClassification Report:\n", report)
plt.figure("DECISION TREE",figsize=(20, 10))
plot_tree(dt, filled=True, feature_names=bc.feature_names, class_names=bc.target_names, rounded=True)
plt.show()
```

Output Screenshot:





```
C:\Users\ajcemca\PycharmProjects\pythonProject\venv\Scripts'
Accuracy: 0.9298245614035088

Classification Report:
              precision    recall  f1-score   support

           0       0.95      0.86      0.90        43
           1       0.92      0.97      0.95        71

    accuracy                           0.93       114
   macro avg       0.93      0.92      0.92       114
weighted avg       0.93      0.93      0.93       114


Process finished with exit code 0
```

Result: The program was executed successfully and the output was obtained. Thus, CO3 has been attained.

Experiment No:13

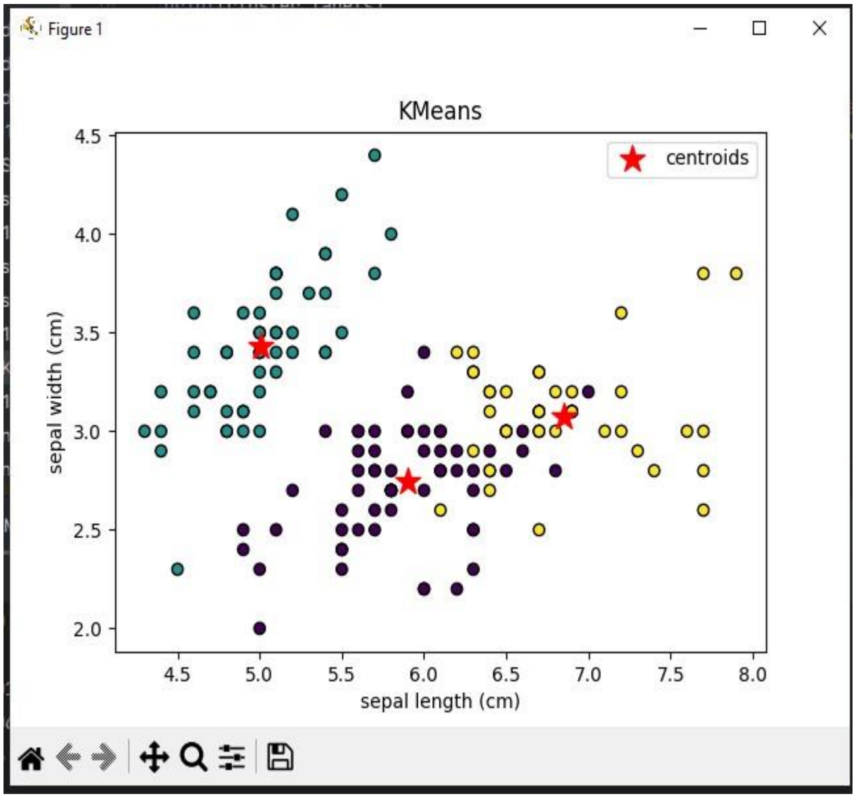
Aim: Program to implement k-means clustering technique using any standard dataset available in the public domain. (Iris Dataset)

CO3: 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='centroids')

plt.xlabel(iris.feature_names[0])

plt.ylabel(iris.feature_names[1])

plt.title('KMeans')

plt.legend()
plt.show()
```

Output Screenshot:



Result: The program was executed successfully and the output was obtained. Thus, CO3 has been attained.

Experiment No:14

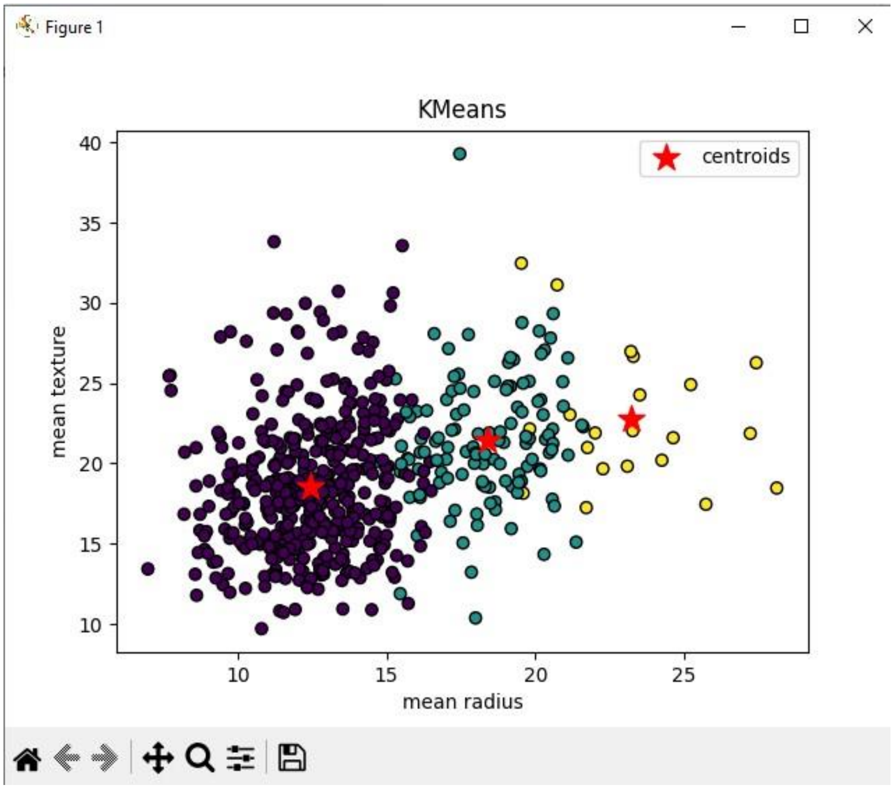
Aim: Program to implement k-means clustering technique using any standard dataset available in the public domain. (Breast Cancer Dataset)

CO3: 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='centroids')
plt.xlabel(bc.feature_names[0])
plt.ylabel(bc.feature_names[1])
plt.title('KMeans')
plt.legend()
plt.show()
```

Output Screenshot:



Result: The program was executed successfully and the output was obtained. Thus, CO3 has been attained.

Experiment No:15

Aim: Program to implement test classification using support vector machine.

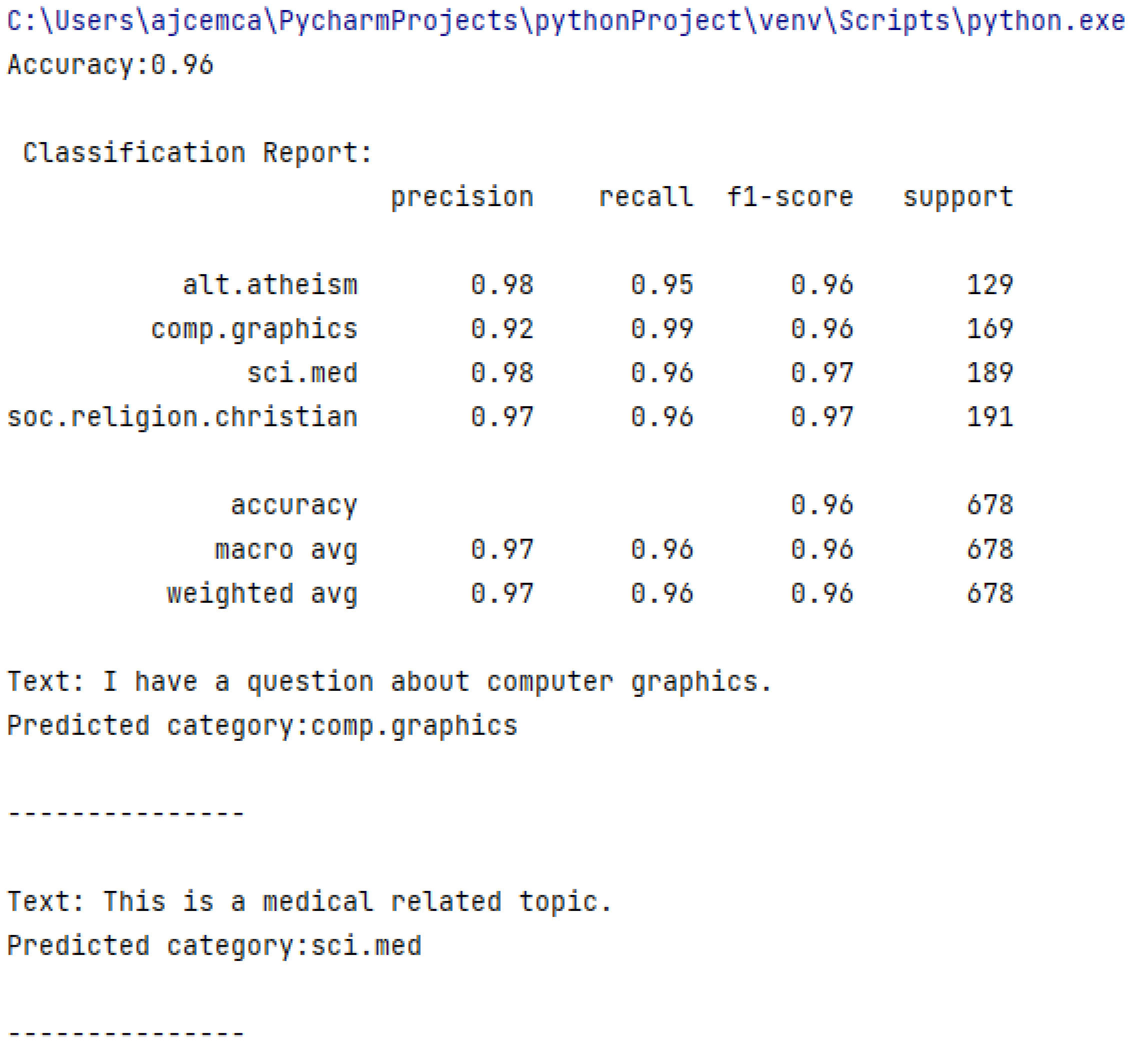
CO3: Use different packages and frameworks to implement text classification using SVM and clustering using k-means.

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)

print(X_train_tfidf)

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)

predictions = svm_classifier.predict(X_test)

accuracy = accuracy_score(y_test, predictions)
```

```
class_report = classification_report(y_test, predictions,
target_names=twenty_train.target_names)

print("Accuracy:", accuracy)

print("\nClassification Report:", class_report)

new_data = [

     "I have a question about computer graphics.",
    "This is a medical related topic.",
]

X_new_tfidf = vectorizer.transform(new_data)

new_predictions = svm_classifier.predict(X_new_tfidf)

for i,text in enumerate(new_data):

    print("Text",text)

    predicted_category=twenty_train.target_names[new_predictions[i]]

    print("Predicted category:",predicted_category)

    print("-------------")
```

Output Screenshot:



```
C:\Users\ajcemca\PycharmProjects\pythonProject\venv\Scripts\python.exe
Accuracy:0.96

 Classification Report:
                        precision    recall  f1-score   support

           alt.atheism       0.98      0.95      0.96       129
         comp.graphics       0.92      0.99      0.96       169
               sci.med       0.98      0.96      0.97       189
soc.religion.christian       0.97      0.96      0.97       191

              accuracy                           0.96       678
             macro avg       0.97      0.96      0.96       678
          weighted avg       0.97      0.96      0.96       678

Text: I have a question about computer graphics.
Predicted category:comp.graphics

---------------

Text: This is a medical related topic.
Predicted category:sci.med

---------------
```

Result: The program was executed successfully and the output was obtained. Thus, CO3 has been attained.

Experiment No:16

Aim: Program on artificial neural network to classify images from any standard dataset in the public domain using Keras framework.

CO4: Implement convolutional neural network algorithm using Keras framework.

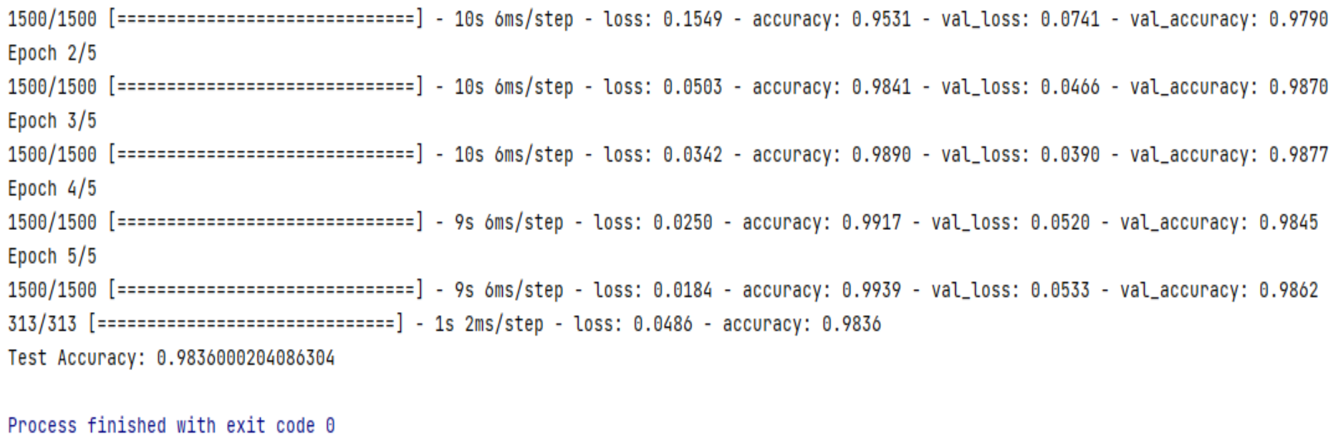
Procedure:

```
import tensorflow as tf
from tensorflow.keras.datasets import mnist
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense,Flatten
from tensorflow.keras.utils import to_categorical
(X_train,y_train),(X_test,y_test) = mnist.load_data()
X_train = X_train/255.0
X_test = X_test/255.0
X_train = X_train.reshape(-1,28*28)
print(X_train)
X_test=X_test.reshape(-1,28*28)
print(X_test)
y_train=to_categorical(y_train)
y_test=to_categorical(y_test)
print(y_test)
model=Sequential([
    Dense(128, activation='relu', input_shape=(28 * 28,)),
    Dense(64,activation='relu'),
    Dense(10,activation='softmax')
])
model.compile(optimizer='adam',loss='categorical_crossentropy',metrics=['accuracy'])
model.fit(X_train,y_train,epochs=5,batch_size=32,validation_split=0.2)
```

loss,accuracy=model.evaluate(X_test,y_test)

print(f"Test_Accuracy:",accuracy)

Output Screenshot:



```
1500/1500 [==============================] - 10s 6ms/step - loss: 0.1549 - accuracy: 0.9531 - val_loss: 0.0741 - val_accuracy: 0.9790
Epoch 2/5
1500/1500 [==============================] - 10s 6ms/step - loss: 0.0503 - accuracy: 0.9841 - val_loss: 0.0466 - val_accuracy: 0.9870
Epoch 3/5
1500/1500 [==============================] - 10s 6ms/step - loss: 0.0342 - accuracy: 0.9890 - val_loss: 0.0390 - val_accuracy: 0.9877
Epoch 4/5
1500/1500 [==============================] - 9s 6ms/step - loss: 0.0250 - accuracy: 0.9917 - val_loss: 0.0520 - val_accuracy: 0.9845
Epoch 5/5
1500/1500 [==============================] - 9s 6ms/step - loss: 0.0184 - accuracy: 0.9939 - val_loss: 0.0533 - val_accuracy: 0.9862
313/313 [==============================] - 1s 2ms/step - loss: 0.0486 - accuracy: 0.9836
Test Accuracy: 0.9836000204086304

Process finished with exit code 0
```

Result: The program was executed successfully and the output was obtained. Thus, CO4 has been attained.

Experiment No:17

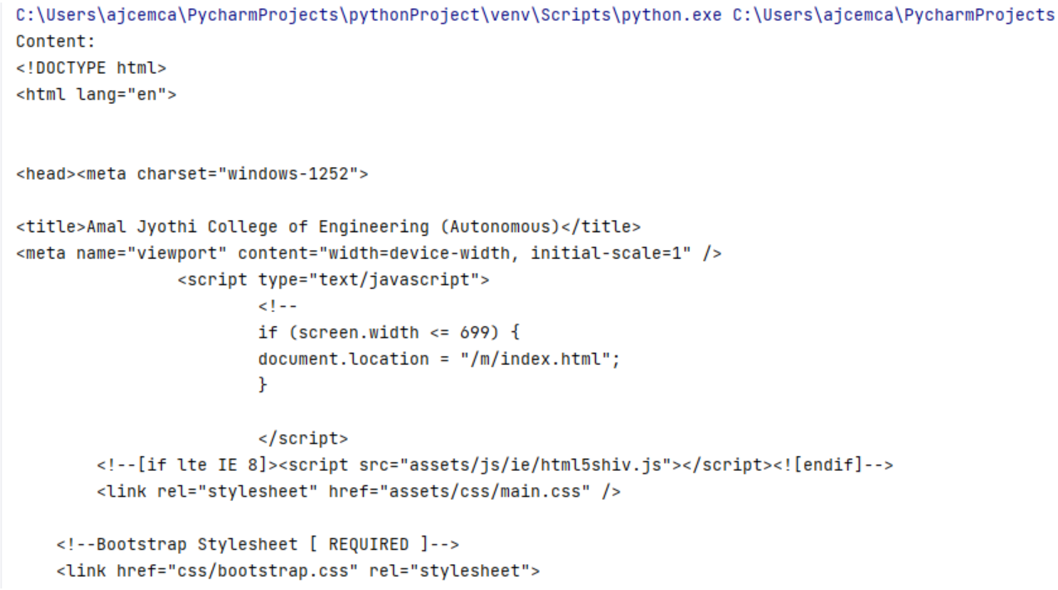
Aim: Program to implement a simple web crawler using requests library

CO5: Implement programs for web data mining and natural language processing using NLTK.

Procedure:

```
import requests
def simple_scraper(url):
    response = requests.get(url)
    if response.status_code == 200:
        print("Content:")
        print(response.text)
    else:
        print("Failed to fetch the page.Status code:", response.status_code)
url_to_scrape = "https://ajce.in"
simple_scraper(url_to_scrape)
```

Output Screenshot:



```
C:\Users\ajcemca\PycharmProjects\pythonProject\venv\Scripts\python.exe C:\Users\ajcemca\PycharmProjects
Content:
<!DOCTYPE html>
<html lang="en">


<head><meta charset="windows-1252">

<title>Amal Jyothi College of Engineering (Autonomous)</title>
<meta name="viewport" content="width=device-width, initial-scale=1" />
                <script type="text/javascript">
                        <!--
                        if (screen.width <= 699) {
                        document.location = "/m/index.html";
                        }

                        </script>
        <!--[if lte IE 8]><script src="assets/js/ie/html5shiv.js"></script><![endif]-->
        <link rel="stylesheet" href="assets/css/main.css" />

    <!--Bootstrap Stylesheet [ REQUIRED ]-->
    <link href="css/bootstrap.css" rel="stylesheet">
```

Result: The program was executed successfully and the output was obtained. Thus, CO5 has been attained.

Experiment No:18

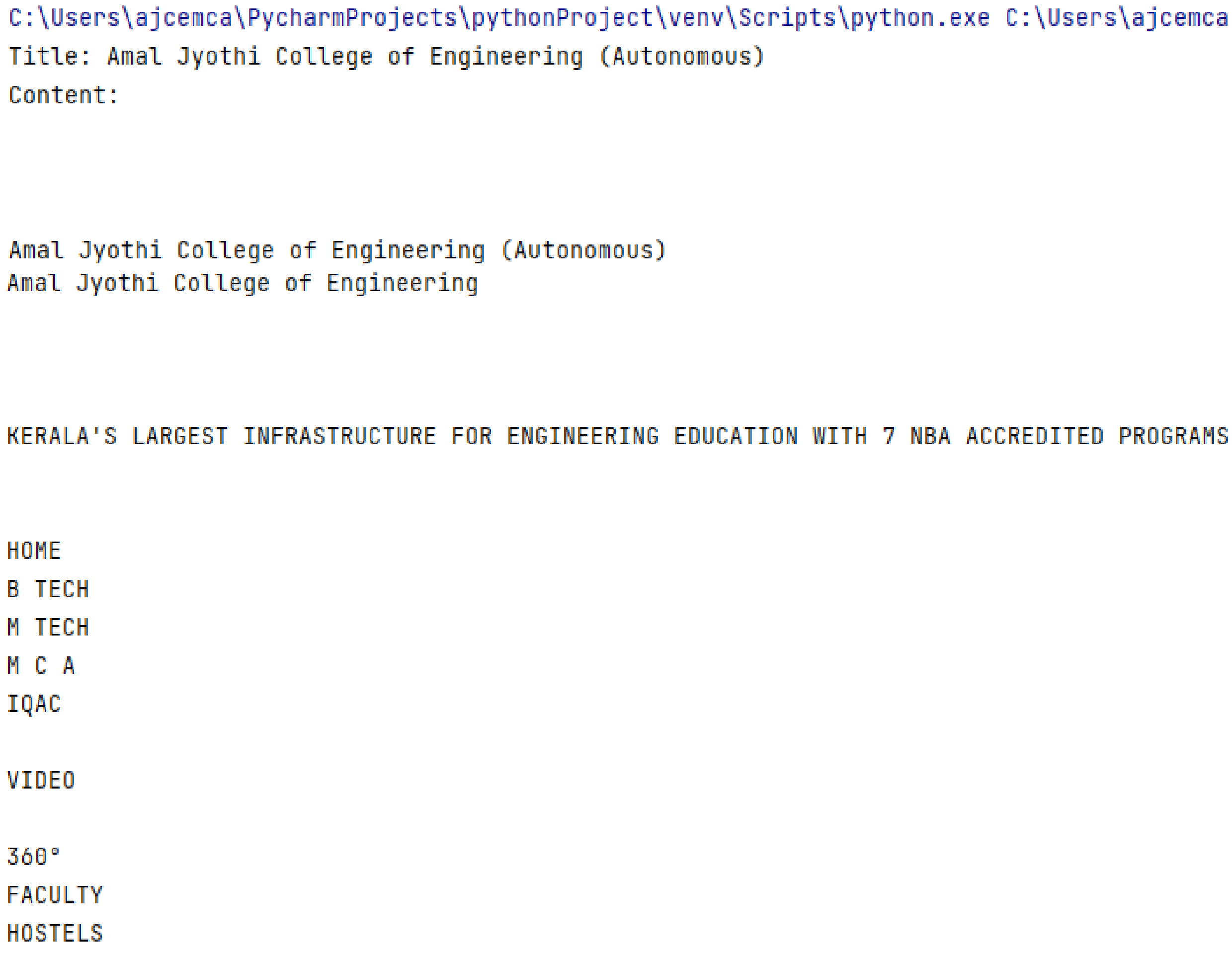
Aim: Program to implement a simple web crawler and parse the content using BeautifulSoup.

CO5: Implement programs for web data mining and natural language processing using NLTK

Procedure:

```
import requests
from bs4 import BeautifulSoup
def simple_scraper_with_bs(url):
    response = requests.get(url)
    if response.status_code == 200:
        soup = BeautifulSoup(response.content, 'html.parser')
        print("Title:", soup.title.string)
        print("Content:")
        print(soup.get_text())
    else:
        print("Failed to fetch the page.Status code:", response.status_code)
url_to_scrape = "https://ajce.in"
simple_scraper_with_bs(url_to_scrape)
```

Output Screenshot:



```
C:\Users\ajcemca\PycharmProjects\pythonProject\venv\Scripts\python.exe C:\Users\ajcemca
Title: Amal Jyothi College of Engineering (Autonomous)
Content:




Amal Jyothi College of Engineering (Autonomous)
Amal Jyothi College of Engineering




KERALA'S LARGEST INFRASTRUCTURE FOR ENGINEERING EDUCATION WITH 7 NBA ACCREDITED PROGRAMS



HOME
B TECH
M TECH
M C A
IQAC

VIDEO

360°
FACULTY
HOSTELS
```

Result: The program was executed successfully and the output was obtained. Thus, CO5 has been attained.

Experiment No:19

Aim: Implement problems on natural language processing – Part of Speech tagging, N-gram &amp; smoothening and Chunking using NLTK

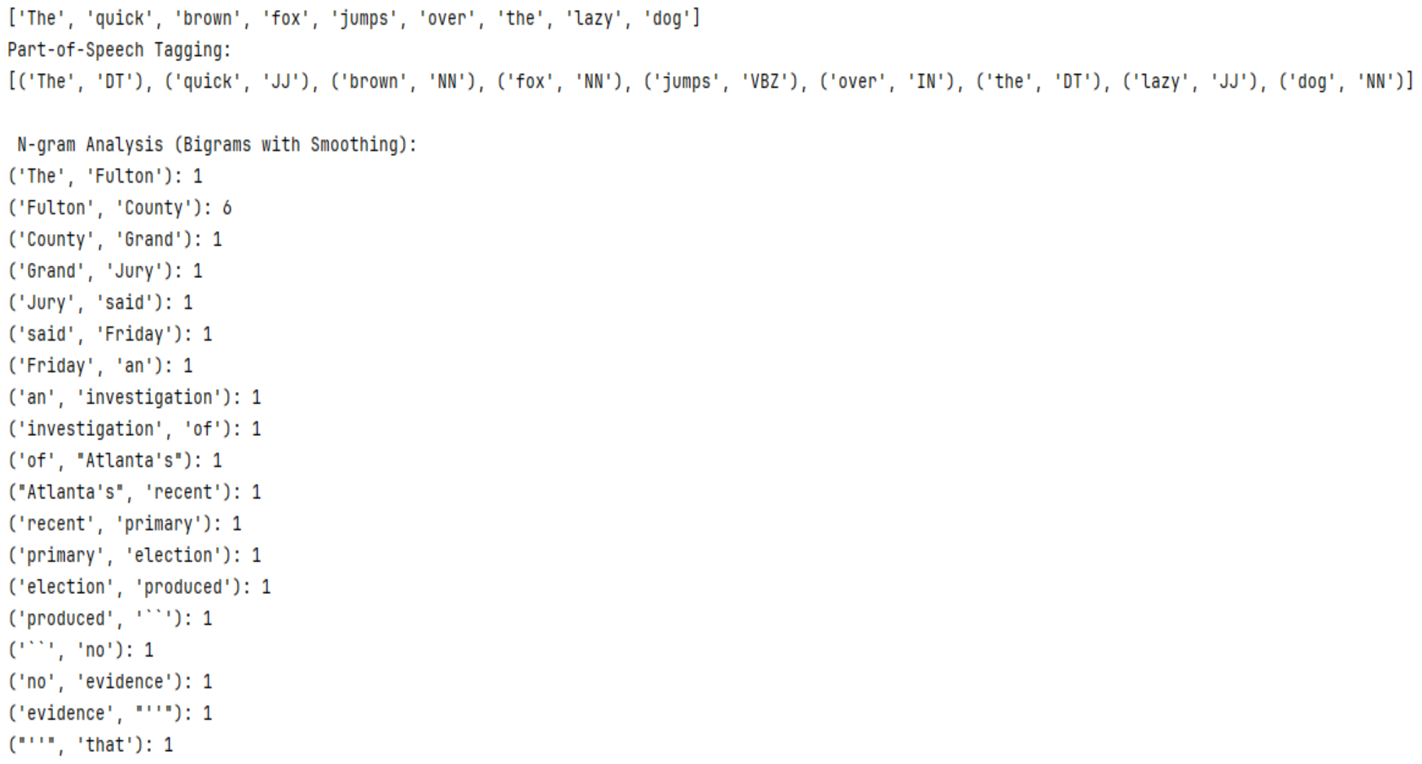
CO5: Implement programs for web data mining and natural language processing using NLTK.

Procedure:

```
import nltk
nltk.download('brown')
nltk.download('punkt')
nltk.download('averaged_perceptron_tagger')
from nltk.tokenize import word_tokenize
from nltk.util import ngrams
from nltk.corpus import brown
from nltk.chunk import RegexpParser
sentence = "The quick brown fox jumps over the lazy dog"
tokens = word_tokenize(sentence)
print(tokens)
pos_tags = nltk.pos_tag(tokens)
print("Part-of-Speech Tagging: ")
print(pos_tags)
text = brown.words(categories='news')[:1000]
bigrams = list(ngrams(text, 2))
freq_dist = nltk.FreqDist(bigrams)
print("\n N-gram Analysis (Bigrams with Smoothing): ")
```

```
for bigram in bigrams:

print(f"{bigram}: {freq_dist[bigram]}")

tagged_sentence = nltk.pos_tag(word_tokenize("The quick brown fox jumps over the lazy dog"))

grammar = r"NP: {<DT>?<JJ>*<NN>}"

cp = RegexpParser(grammar)

result = cp.parse(tagged_sentence)

print("\n Chunking with Regular Expressions and POS tags: ")

print(result)
```

Output Screenshot:



```
['The', 'quick', 'brown', 'fox', 'jumps', 'over', 'the', 'lazy', 'dog']
Part-of-Speech Tagging:
[('The', 'DT'), ('quick', 'JJ'), ('brown', 'NN'), ('fox', 'NN'), ('jumps', 'VBZ'), ('over', 'IN'), ('the', 'DT'), ('lazy', 'JJ'), ('dog', 'NN')]

 N-gram Analysis (Bigrams with Smoothing):
('The', 'Fulton'): 1
('Fulton', 'County'): 6
('County', 'Grand'): 1
('Grand', 'Jury'): 1
('Jury', 'said'): 1
('said', 'Friday'): 1
('Friday', 'an'): 1
('an', 'investigation'): 1
('investigation', 'of'): 1
('of', "Atlanta's"): 1
("Atlanta's", 'recent'): 1
('recent', 'primary'): 1
('primary', 'election'): 1
('election', 'produced'): 1
('produced', '``'): 1
('``', 'no'): 1
('no', 'evidence'): 1
('evidence', "''"): 1
("''", 'that'): 1
```

Result: The program was executed successfully and the output was obtained. Thus, CO5 has been attained.