**Experiment No.: 1**

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

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

**Procedure**:

import numpy as np

def input\_matrix(ourmatrix):

r = int(input(f"Enter the no of rows for {ourmatrix}:"))

c = int(input(f"Enter the no of columns for {ourmatrix}:"))

matrix=[]

print("Enter the elements:")

for i in range(r):

r=[]

for j in range(c):

elements=int(input(f"enter the element at row{i+1},colomn{j+1}"))

r.append(elements)

matrix.append(r)

return np.array(matrix)

matrix1=input\_matrix("matrix1")

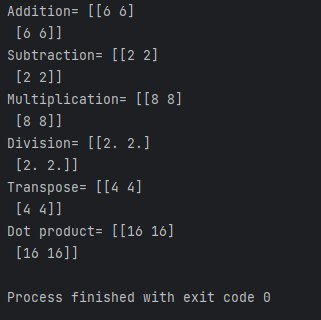
input\_matrix(matrix1)

matrix2=input\_matrix("matrix2")

input\_matrix(matrix2)

**Output Screenshot**





**Result:**

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

**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=")

print(U)

print("S=")

print(np.diag(S))

print("VT=")

print(VT)

reconstructed\_matrix= np.dot (U, np.dot(np.diag(S), VT))

print("Reconstructed matrix=")

print(reconstructed\_matrix)

**Output Screenshot**



**Result:**

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

**Experiment No.: 3**

**Aim:** Program to perform data visualisation using the python library matplotlib.

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

**Procedure**:

import matplotlib.pyplot as plt

categories = ['Category A','Category B','Category C','Category D']

values = [15,30,22,40]

plt.bar(categories,values,color="pink")

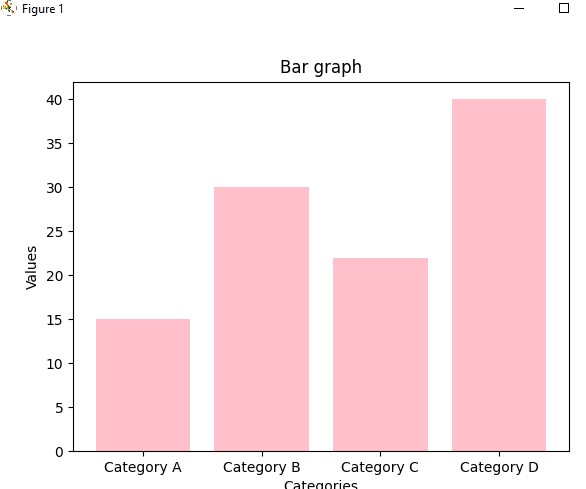
plt.xlabel("Categories")

plt.ylabel("Values")

plt.title("Bar graph")

plt.show()

**Output Screenshot**



**Result:**

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

**Experiment No.: 4**

**Aim:** Program to implement KNN classification using any standard dataset available in the public domain and find the accuracy of the 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)

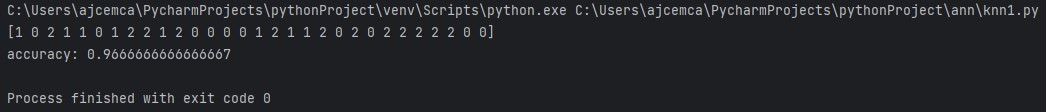
print(knn.predict(x\_test))

V=knn.predict(x\_test)

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

print("accuracy:",result)

**Output Screenshot**



**Result:**

The program was executed and the result was successfully obtained. Thus, CO2 was obtained.

**Experiment No.: 5**

**Aim:** Program to implement KNN classification using any standard dataset available in the public domain and find the accuracy of the 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=7)

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

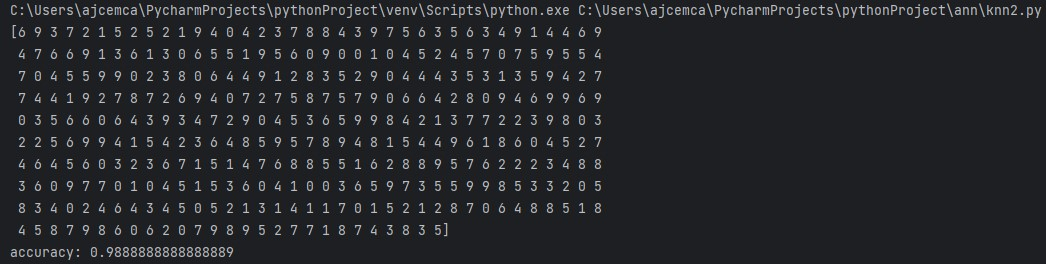
print(knn.predict(x\_test))

V=knn.predict(x\_test)

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

print("accuracy:",result)

**Output Screenshot**



**Result:**

The program was executed and the result was successfully obtained. Thus, CO2 was obtained.

**Experiment No.: 6**

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

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

**Procedure**:

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

from sklearn.datasets import load\_iris

from sklearn.naive\_bayes import GaussianNB

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)

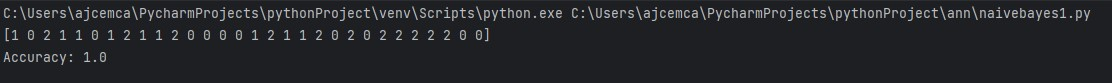
print(clf.predict(x\_test))

V=clf.predict(x\_test)

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

print("Accuracy:",result)

**Output Screenshot**



**Result:**

The program was executed and the result was successfully obtained. Thus, CO2 was obtained.

**Experiment No.: 7**

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

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

**Procedure**:

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

from sklearn.datasets import load\_breast\_cancer

from sklearn.naive\_bayes import GaussianNB

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

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)

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)

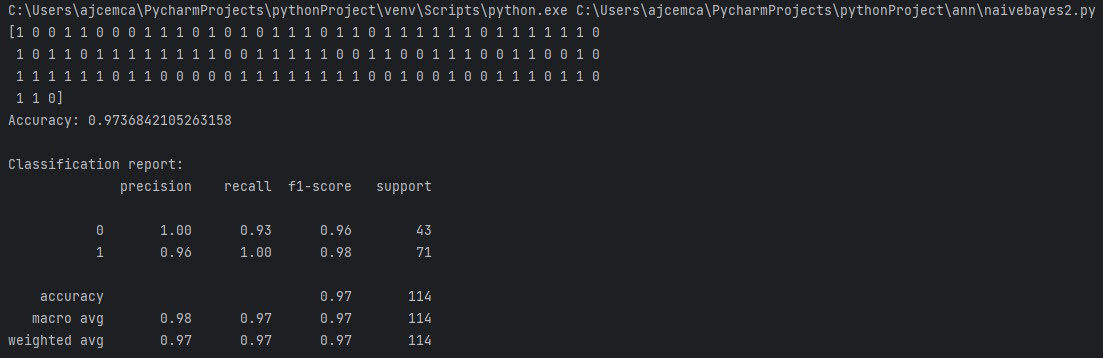
print()

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

print("Classification report:")

print(report)

**Output Screenshot**



**Result:**

The program was executed and the result was successfully obtained. Thus, CO2 was obtained.

**Experiment No.: 8**

**Aim:** Given 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 numpy as np

from sklearn.linear\_model import LinearRegression

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])

regressor = LinearRegression()

regressor.fit(x, y)

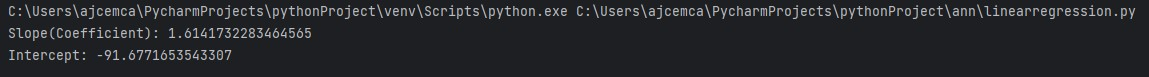
slope = regressor.coef\_[0]

intercept = regressor.intercept\_

print(f"Slope(Coefficient): {slope}")

print(f"Intercept: {intercept}")

**Output Screenshot**



**Result:**

The program was executed and the result was successfully obtained. Thus, CO2 was obtained.

**Experiment No.: 9**

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

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

**Procedure**:

import pandas as pd

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

from sklearn.linear\_model import LinearRegression

from sklearn.metrics import r2\_score

import matplotlib.pyplot as plt

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)

regressor = LinearRegression()

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

print(regressor.predict(x\_test))

V = regressor.predict(x\_train)

r2 = r2\_score(y\_train, V)

print("\nR squared:", r2)

plt.scatter(x\_test, y\_test, color="black", label="data points")

plt.plot(x\_train, V, color="blue", linewidth=3, label="Regression line")

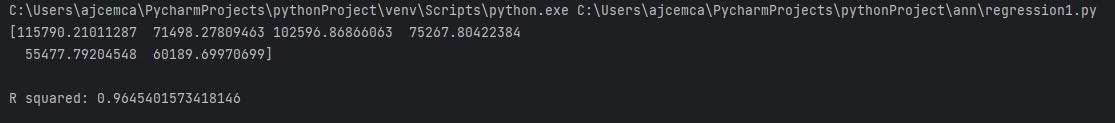
plt.xlabel("Years of Experience")

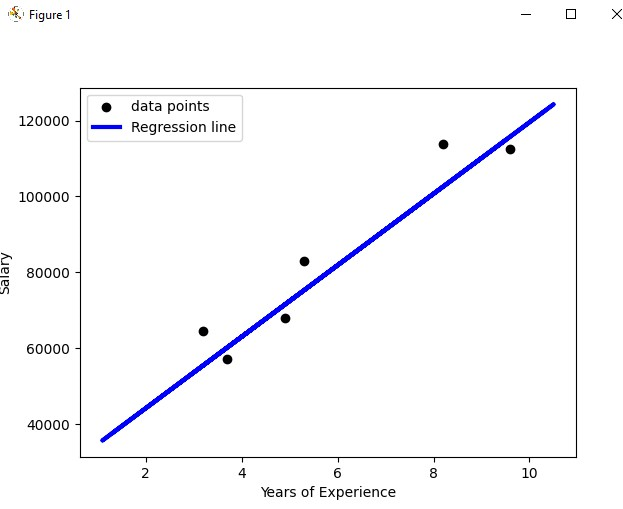
plt.ylabel("Salary")

plt.legend()

plt.show()

**Output Screenshot**





**Result:**

The program was executed and the result was successfully obtained. Thus, CO2 was obtained.

**Experiment No.: 10**

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

**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.model\_selection import train\_test\_split

from sklearn.linear\_model import LinearRegression

from sklearn.metrics import mean\_squared\_error

california\_housing = fetch\_california\_housing()

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

df["Target"] = california\_housing.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)

regressor = LinearRegression()

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

v = regressor.predict(x\_test)

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

print("Mean squared error: ", mse)

**Output Screenshot**



**Result:**

The program was executed and the result was successfully obtained. Thus, CO2 was obtained.

**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.datasets import load\_iris

from sklearn import tree

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

from sklearn.tree import DecisionTreeClassifier

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

import matplotlib.pyplot as plt

iris=load\_iris()

x=iris.data

y=iris.target

max\_depth=3

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

clf=DecisionTreeClassifier(max\_depth=max\_depth)

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

plt.figure(figsize=(15,10))

tree.plot\_tree(clf,filled=True,feature\_names=iris.feature\_names)

plt.title("Decision tree")

plt.show()

print(clf.predict(x\_test))

V=clf.predict(x\_test)

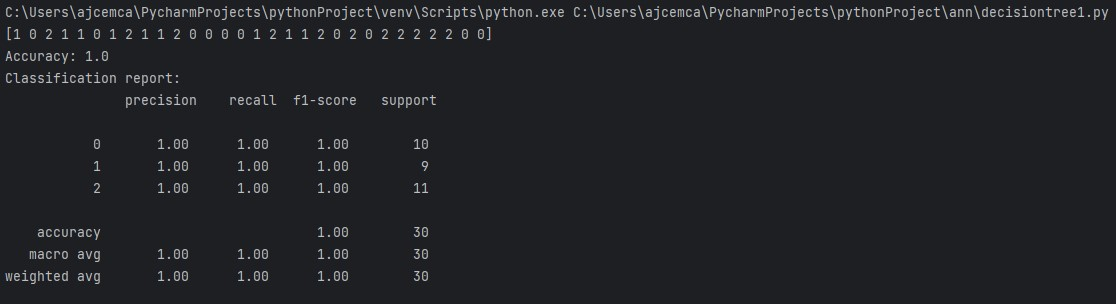
result=accuracy\_score(y\_test, V)

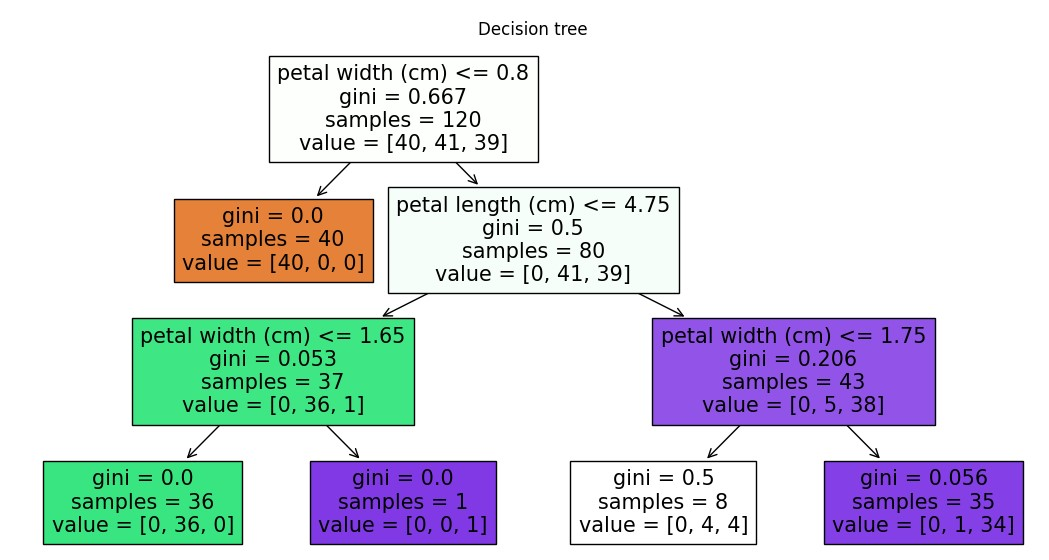
report=classification\_report(y\_test,V)

print("Accuracy:",result)

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

**Output Screenshot**





**Result:**

The program was executed and the result was successfully obtained. Thus, CO3 was obtained.

**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.datasets import load\_breast\_cancer

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

from sklearn.tree import DecisionTreeClassifier

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

from sklearn import tree

import matplotlib.pyplot as plt

data=load\_breast\_cancer()

x=data.data

y=data.target

max\_depth=3

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

clf=DecisionTreeClassifier(max\_depth=max\_depth)

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

plt.figure(figsize=(15,10))

tree.plot\_tree(clf,filled=True,feature\_names=data.feature\_names)

plt.title("Decision tree")

plt.show()

print(clf.predict(x\_test))

V=clf.predict(x\_test)

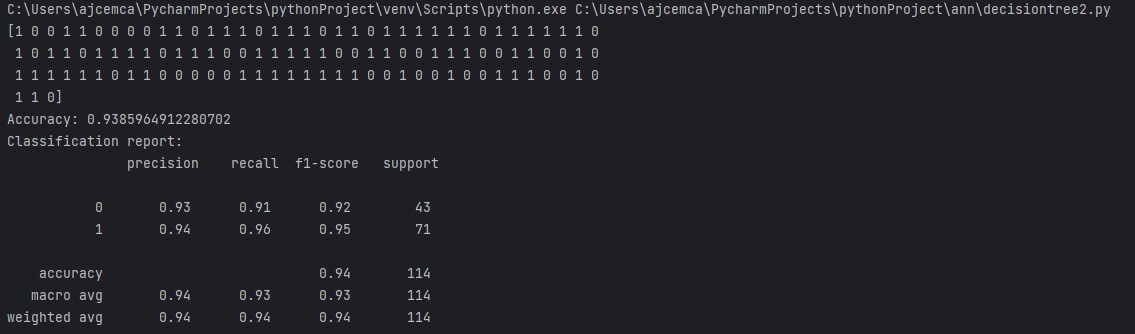
result=accuracy\_score(y\_test, V)

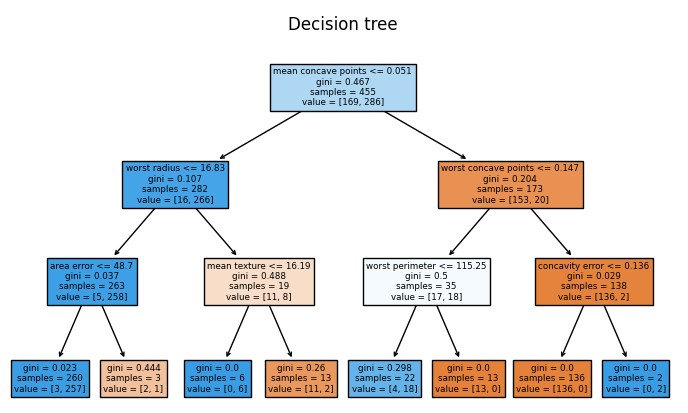
report=classification\_report(y\_test,V)

print("Accuracy:",result)

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

**Output Screenshot**





**Result:**

The program was executed and the result was successfully obtained. Thus, CO3 was obtained.

**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="Blues", marker="o", edgecolors="black")

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

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

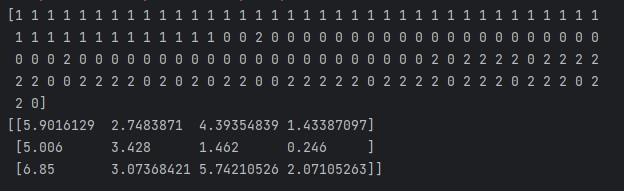
plt.ylabel(iris.feature\_names[1])

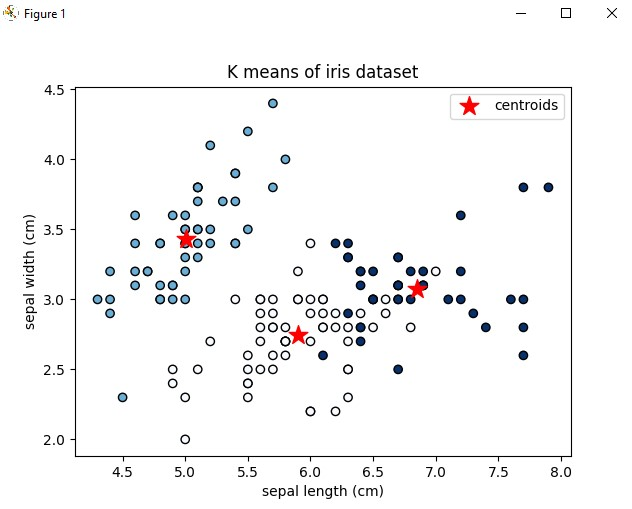
plt.title("K means of iris dataset")

plt.legend()

plt.show()

**Output Screenshot**





**Result:**

The program was executed and the result was successfully obtained. Thus, CO3 was obtained.

**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

data = load\_breast\_cancer()

x = data.data

y = data.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="Blues", marker="o", edgecolors="black")

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

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

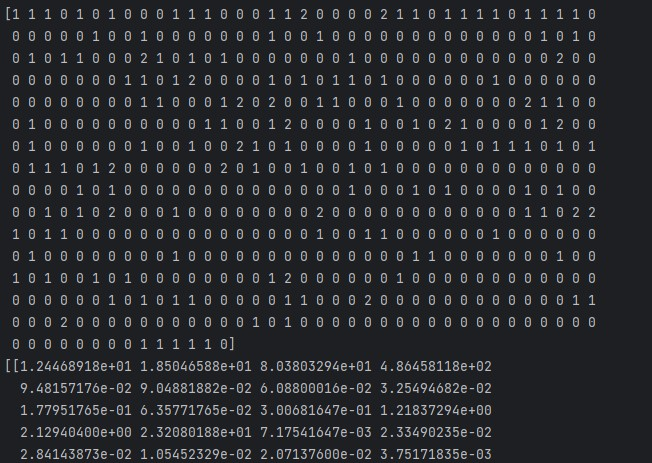
plt.ylabel(data.feature\_names[1])

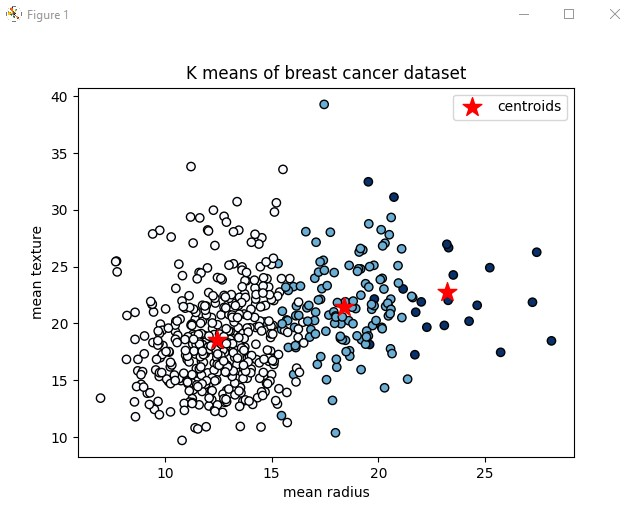
plt.title("K means of breast cancer dataset")

plt.legend()

plt.show()

**Output Screenshot**





**Result:**

The program was executed and the result was successfully obtained. Thus, CO3 was obtained.

**Experiment No.: 15**

**Aim:** Program to implement text 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 accuracy\_score,classification\_report

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)

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

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

report = classification\_report(y\_test,prediction,target\_names=twenty\_train.target\_names)

print("Accuracy = ",accuracy)

print("Classification report =")

print(report)

new\_data=[

"I have a question about computer graphics"

]

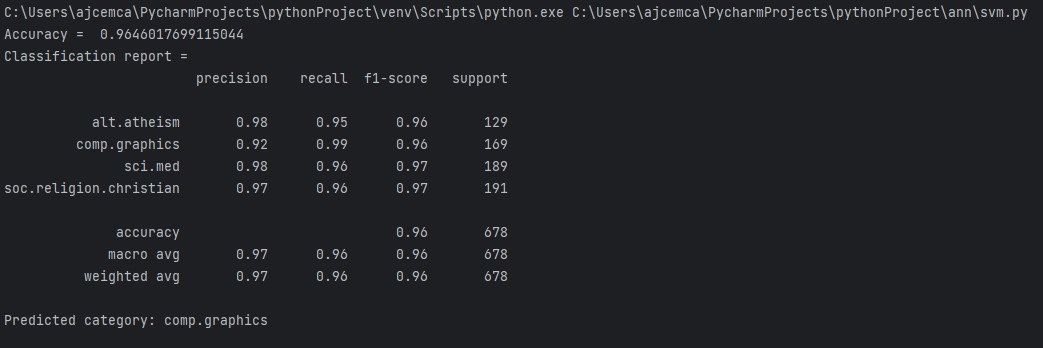
x\_new=vectorizer.transform(new\_data)

newprediction=svm\_classifier.predict(x\_new)

predicted\_category=twenty\_train.target\_names[newprediction[0]]

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

**Output Screenshot**



**Result:**

The program was executed and the result was successfully obtained. Thus, CO3 was obtained.

**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

# Load the MNIST dataset

(X\_train, y\_train), (X\_test, y\_test) =mnist.load\_data()

# Normalize pixel values to be between 0 and 1

X\_train = X\_train / 255.0

X\_test = X\_test / 255.0

# Flatten the images (convert 28x28 images to 1D vectors)

X\_train = X\_train.reshape(-1, 28 \* 28)

print(X\_train)

X\_test = X\_test.reshape(-1, 28 \* 28)

print(X\_train)

# One-hot encode the target labels

y\_train = to\_categorical(y\_train)

y\_test = to\_categorical(y\_test)

print(y\_test)

# Create a simple feedforward neural network model

model=Sequential([

Dense(128, activation='relu', input\_shape=(28 \* 28,)),

Dense(68, activation='relu'),

Dense(10, activation='softmax')

])

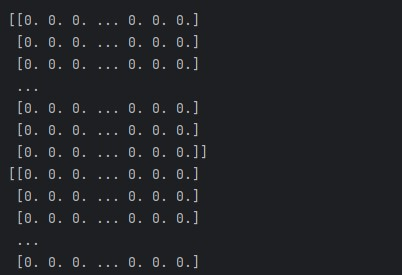
model.compile(optimizer='adam', loss='categorical\_crossentropy', metrics=['accuracy'])

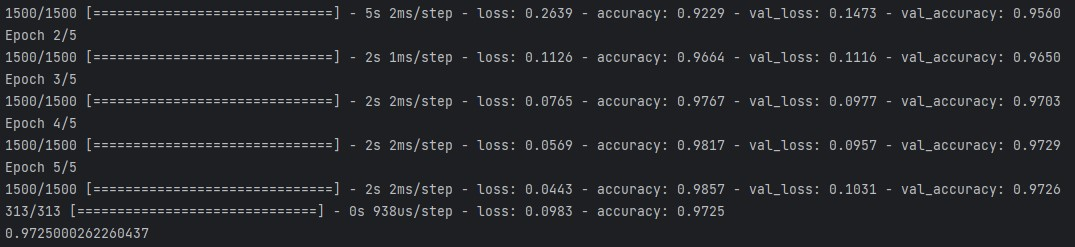
model.fit(X\_train,y\_train, epochs=5 , batch\_size=32, validation\_split=0.2)

loss, accuracy= model.evaluate(X\_test,y\_test)

print(accuracy)

**Output Screenshot**





**Result:**

The program was executed and the result was successfully obtained. Thus, CO4 was obtained.

**Experiment No.: 17**

**Aim:** Program to implement a simple web crawler and scrapping web pages.

**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)

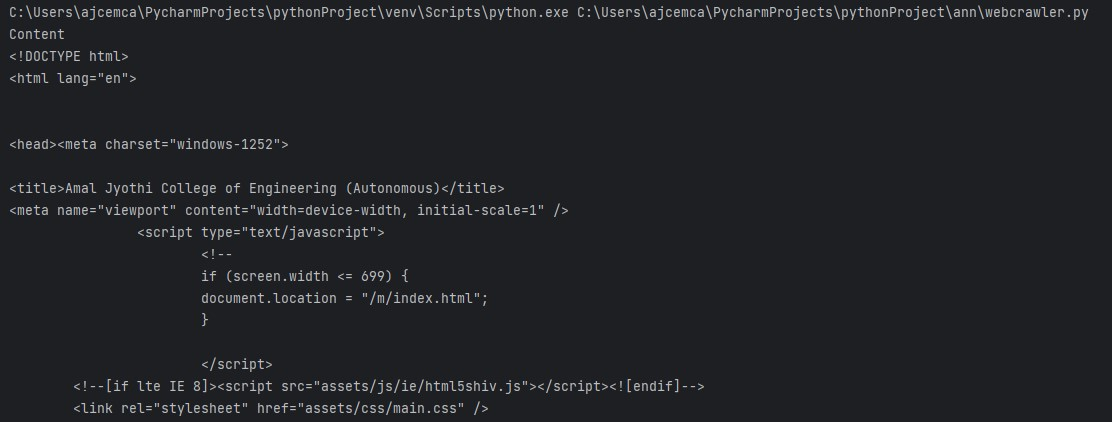
else:

print("Failed to fetch the page. Status\_code:", response.status\_code)

url\_to\_scrap = 'http://ajce.in'

simple\_scraper(url\_to\_scrap)

**Output Screenshot**



**Result:**

The program was executed and the result was successfully obtained. Thus, CO5 was obtained.

**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())

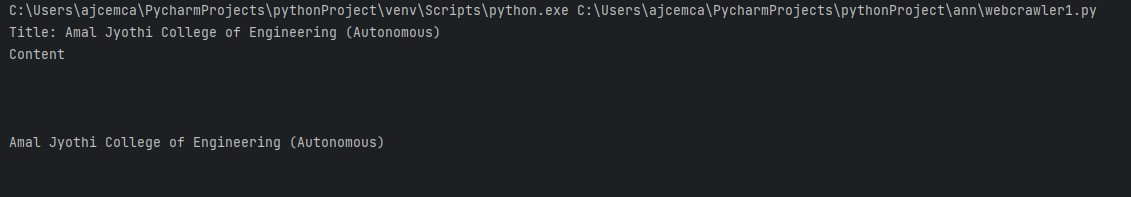
else:

print("Failed to fetch the page. Status\_code:", response.status\_code)

url\_to\_scrap = 'http://ajce.in'

simple\_scraper\_with\_bs(url\_to\_scrap)

**Output Screenshot**



**Result:**

The program was executed and the result was successfully obtained. Thus, CO5 was obtained.

**Experiment No.: 19**

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

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

**Procedure**:

import nltk

nltk.download('punkt')

nltk.download('brown')

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("\nN-gram Analysis(Bigram with Smoothening)")

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>}"

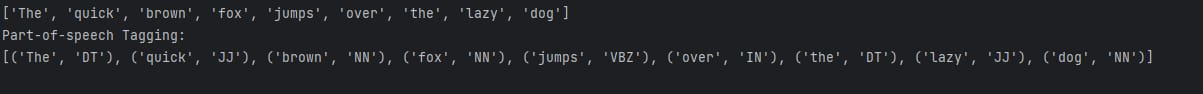
cp = RegexpParser(grammar)

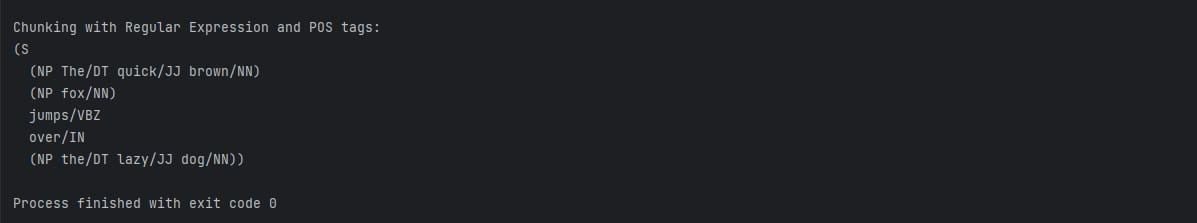
result = cp.parse(tagged\_sentence)

print("\nChunking with Regular Expression and POS tags")

print(result)

**Output Screenshot**





**Result:**

The program was executed and the result was successfully obtained. Thus, CO5 was obtained.