**1. Write a program to perform different matrix operations on a 2D Matrix**

**CODE:**

print("Amrutha Biju")

print("21MCA003")

import numpy as np

A=np.array([ [2, 4],[5, 6] ])

B=np.array([ [9, 3],[9, 6] ])

print("Matrix addition ")

C=A+B

print(C)

print("Matrix Substraction ")

C=A-B

print(C)

print("Multiply the individual elements of matrix ")

C=np.multiply(A,B)

print(C)

print("Divide the elements of the matrices ")

C=np.divide(A,B)

print(C)

print("Matrix Multiplication" )

C=np.matmul(A,B)

print(C)

print("Display transpose of the matrix ")

C=np.transpose(C)

print(C)

print("Sum of diagonal element of matrix ")

C=np.diagonal(C)

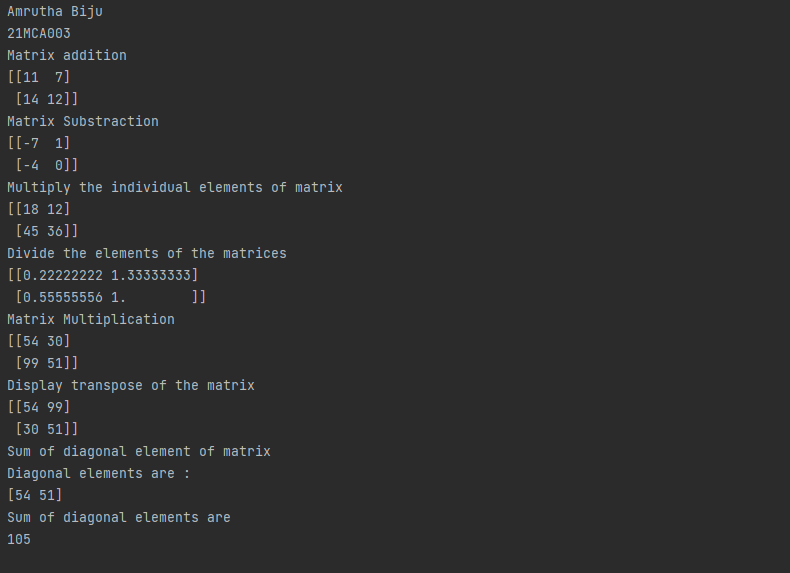
print("Diagonal elements are :")

print(C)

print("Sum of diagonal elements are ")

print(sum(C))

**OUTPUT**

****

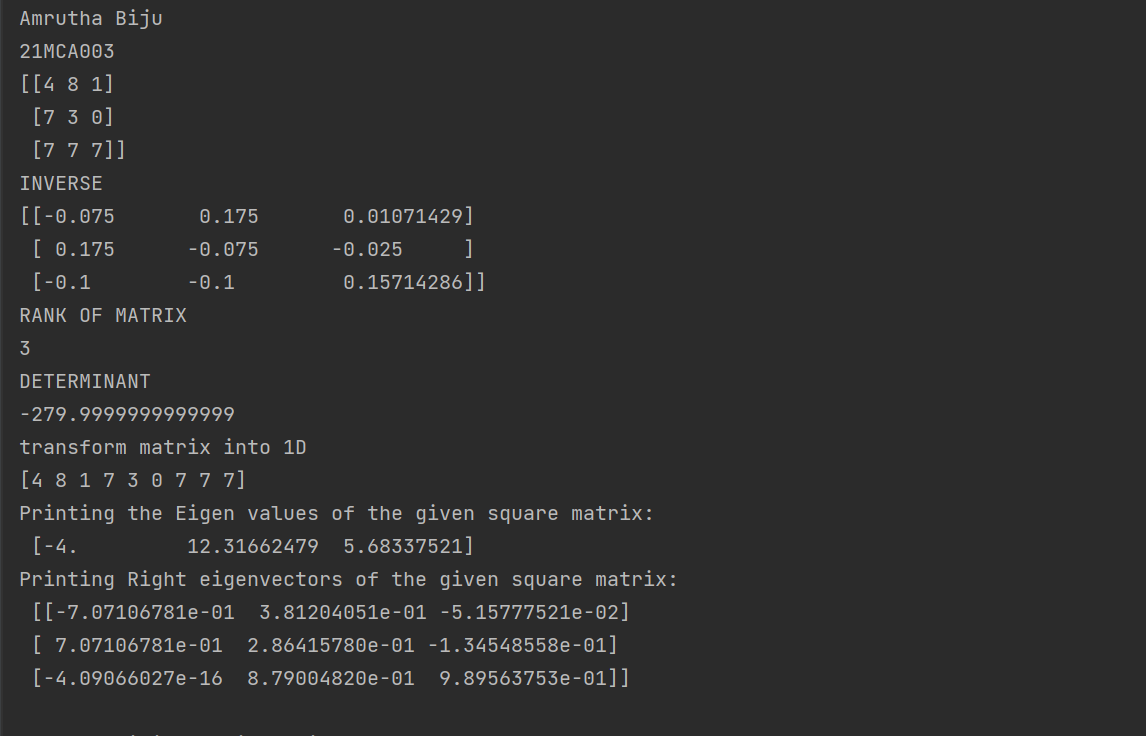
**2. Write a program to find the inverse, rank, determinant, Eigen values of a given matrix. Also transform the matrix to 1D array.**

**CODE:**

print("Amrutha Biju")  
print("21MCA003")  
  
import numpy as np  
m = np.random.randint(10, size=(3, 3))  
print(m)  
print("INVERSE")  
inverse=np.linalg.inv(m)  
print(inverse)  
print("RANK OF MATRIX")  
rank = np.linalg.matrix\_rank(m)  
print(rank)  
  
print("DETERMINANT")  
det=np.linalg.det(m)  
print(det)  
print("transform matrix into 1D")  
tmatrix = np.ravel(m)  
print(tmatrix)  
w, v = np.linalg.eig(m)

print("Printing the Eigen values of the given square matrix:\n",w)  
print("Printing Right eigenvectors of the given square matrix:\n",v)

**OUTPUT**

****

**3. Write a program to display the elements of the matrix X to different powers and identity matrix of a given matrix .Also create another matrix Y with same dimensions and display X2+2Y.**

**CODE:**

print("Amrutha Biju")

print("21mca003")

import numpy as np

A = np.array([ [1, 2, 3], [2, 2, 2], [3, 3, 3] ])

#B = np.array([ [3, 2, 1], [1, 2, 3], [1, 2, 3] ])

arrA = np.multiply(A,A)

print("The multiply od matrix is :")

print(arrA)

arrB = np.power(A, 3)

print("The power of each matrix is :")

print(arrB)

arrC = np.identity(3)

print("The identity matrix is :")

print(arrC)

arrD = np.power(A,3)

print("Power of each element of matrix is : ")

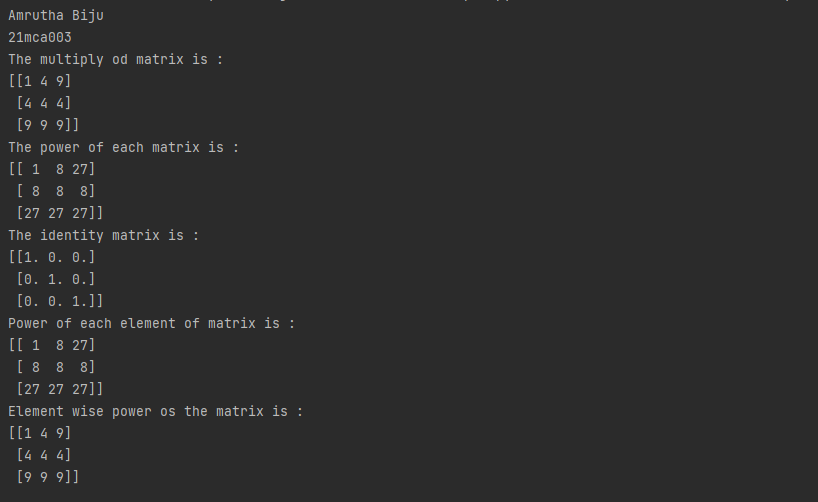
print(arrD)

arrE=np.power(A,2)

print("Element wise power os the matrix is :")

print(arrE)

**OUTPUT**



**4. Write a Program to display various elements of a give 4x4 matrix specifying appropriate indices.**

**CODE:**

print("Amrutha Biju")

print("21MCA003")

import numpy as np

X = np.array( [ [ 1, 6, 7, 4],

[ 5, 9, 2, 1],

[ 3, 8, 4, 6],

[ 2, 3, 6, 1] ] )

print("Original form")

print(X)

print("Excluding the first row")

print(X[1:,])

print("Alternate method for Excluding the first row")

num=np.delete(X,0,axis=0)

print(num)

print("Excluding last column")

print(X[:, :-1])

print("Display the elements of 1st and 2nd column in 2nd and 3rd row")

print(X[1:3,0:2])

print("Display the elements of 2nd and 3rd column")

print(X[:,[1,2]])

print("Display 2nd and 3rd element of 1st row")

print(X[0:1,1:3])

print("Display the elements from indices 4 to 10 in descending order")

flat\_array=X.flatten()

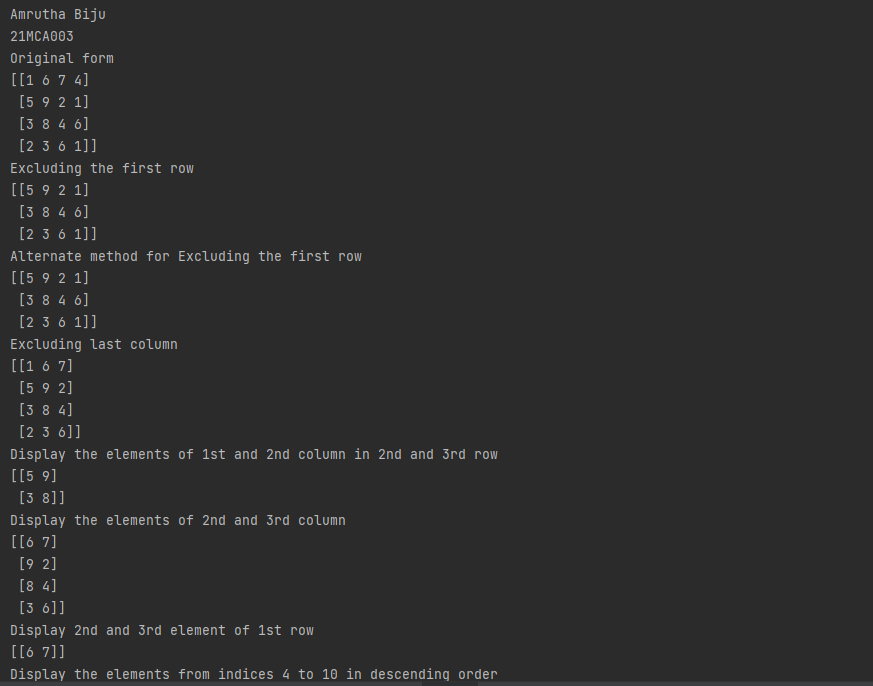
print(flat\_array)

new=sorted(flat\_array[-3:-10])

index=flat\_array[11:4:-1]

print(index)

**OUTPUT**

****

**5. Write a program to perform the SVD of a given matrix.**

**CODE:**

print("Amrutha Biju")

print("21mca003")

import numpy as np

A = np.array([[2, 1, -2],

[3, 0, 1],

[1, 1, -1]])

U, D, VT = np.linalg.svd(A)

print("Decomposed value of U :")

print(U)

print()

print("Decomposed value of D :")

print(D)

print()

print("Decomposed value of VT :")

print(VT)

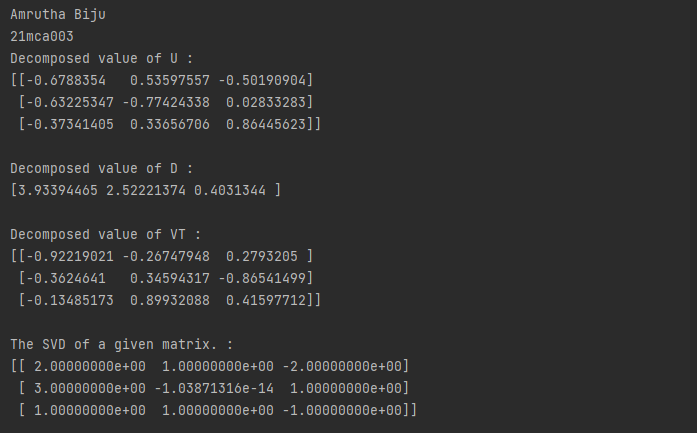
print()

A\_remake = (U @ np.diag(D) @ VT)

print("The SVD of a given matrix. :")

print(A\_remake)

**OUTPUT**

****

**6. Write a program to Solve systems of equations with numpy**

**CODE:**

print("Amrutha Biju")

print("21mca003")

import numpy as np

A = np.array([[2, 1, -2],

[3, 0, 1],

[1, 1, -1]])

b = np.array([[-3],

[5],

[-2]])

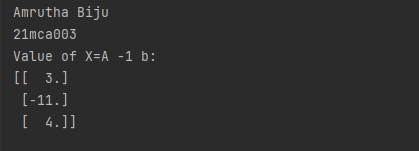
a=np.linalg.inv(A)

x= np.linalg.solve(a, b)

print("Value of X=A -1 b: ")

print(x)

**OUTPUT**

****

**7. Program to create a line graph with the specified style properties, given the information regarding the car details.**

**CODE:**

print("Amrutha Biju")

print("21mca003")

from matplotlib import pyplot as plt

import numpy as np

x = np.array([2001,2002,2003,2004,2005,2006,2007])

y = np.array([24000,22500,19700,17500,14500,10000,5800])

plt.plot(x,y)

plt.xlabel("year")

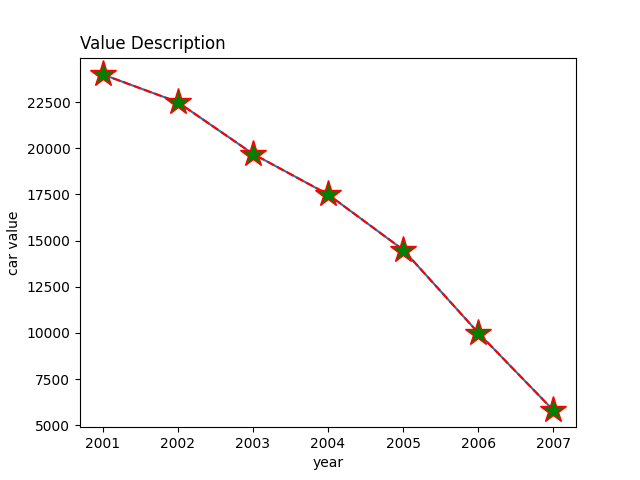
plt.ylabel("car value")

plt.title("Value Description",loc='left')

plt.plot(x,y,linestyle='dashed',color='r',marker='\*',markersize='20',markerfacecolor='green')

plt.show()

**OUTPUT**

****

**8. Program to represent the daily sales of the 2 items in a shop using line graph with grids and appropriate style properties.**

**CODE:**

print("Amrutha Biju")

print("21mca003")

import matplotlib.pyplot as plt

x = ['Mon','Tue','Wed','Thurs','Fri']

y = [300,450,150,400,650]

plt.subplot(2,1,1)

plt.plot(x,y,linestyle='dotted',color='cyan',marker='h',markerfacecolor='magenta',markeredgecolor='black')

plt.xlabel('Days of weak')

plt.ylabel('Sales of Drink')

plt.title("Sales Data1")

plt.grid(color='blue',linestyle=':')

x = ['Mon','Tue','Wed','Thurs','Fri']

y = [400,500,350,300,500]

plt.subplot(2,1,2)

plt.plot(x,y,linestyle='dashed',color='yellow',marker='D',markerfacecolor='green',markeredgecolor='red')

plt.xlabel('Days of weak')

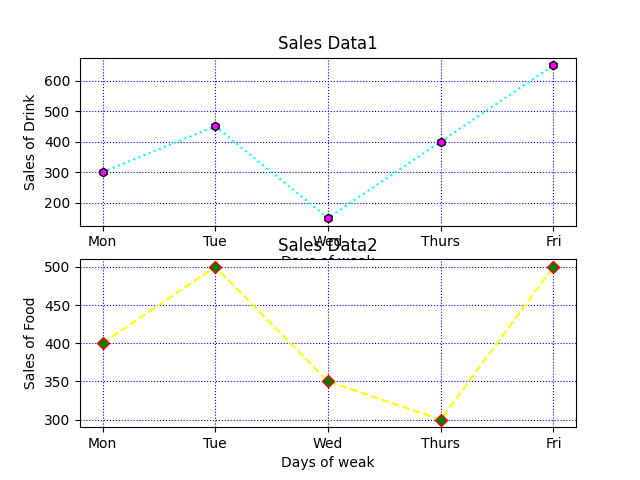
plt.ylabel('Sales of Food')

plt.title("Sales Data2")

plt.grid(color='blue',linestyle=':')

plt.show()

**OUTPUT**

****

**9. Program to create a scatter plot for the product details.**

**CODE:**

print("Amrutha Biju")

print("21mca003")

import matplotlib.pyplot as plt

x = ['Jan','Feb','Mar','Apr','May','Jun','July','Aug','Sep','Oct','Nov','Dec']

y = [173,153,195,147,120,144,148,109,174,130,172,131]

plt.title('Sales Data')

plt.xlabel('Month of Years',fontsize=18)

plt.scatter(x,y,color='pink')

x = ['Jan','Feb','Mar','Apr','May','Jun','July','Aug','Sep','Oct','Nov','Dec']

y = [189,189,105,112,173,109,151,197,174,145,177,161]

plt.scatter(x,y,color='yellow')

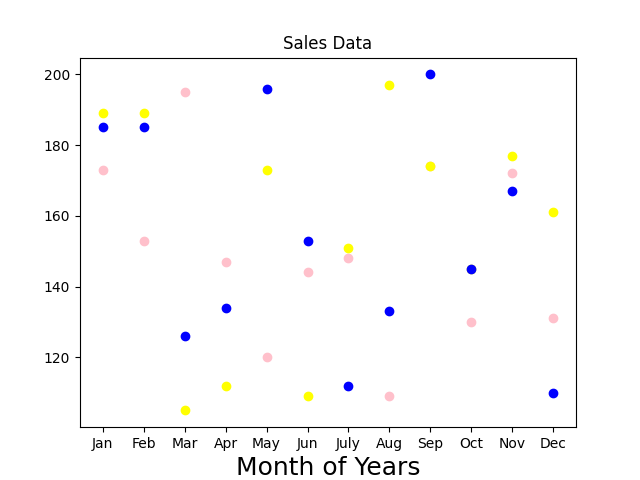
x = ['Jan','Feb','Mar','Apr','May','Jun','July','Aug','Sep','Oct','Nov','Dec']

y = [185,185,126,134,196,153,112,133,200,145,167,110]

plt.scatter(x,y,color='blue')

plt.show()

**OUTPUT**

****

**10. Program to create bar chart for given data regarding ‘Primary mode of transport’**

**CODE:**

print("Amrutha Biju")

print("21mca003")

import matplotlib.pyplot as plt

import numpy as np

data={'Walking': 29,'Cycling': 15,'Car':35,'Bus':18,'Train':3}

transport=list(data.keys())

frequency = list(data.values())

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

plt.bar(transport, frequency, color ='green', width = 0.1)

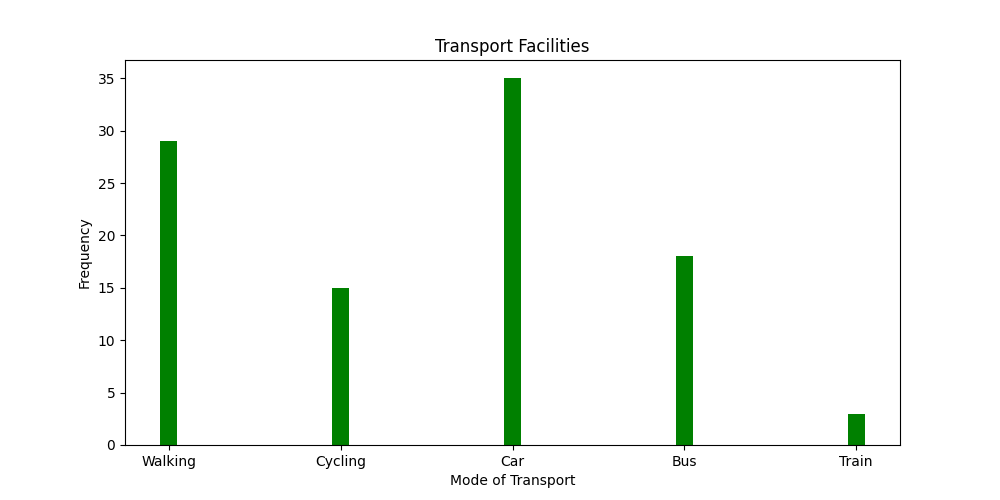
plt.xlabel("Mode of Transport")

plt.ylabel("Frequency")

plt.title("Transport Facilities")

plt.show()

**OUTPUT**

****

**11. Program to create histogram with bin size of 5 for the given data regarding height of cherry trees.**

**CODE:**

print("Amrutha Biju")

print("21mca003")

import matplotlib.pyplot as plt

import numpy as np

fig,ax = plt.subplots(1,1)

a=np.array([61, 63, 64, 66, 68, 69, 71, 71.5, 72, 72.5, 73, 73.5, 74, 74.5, 76, 76.2,

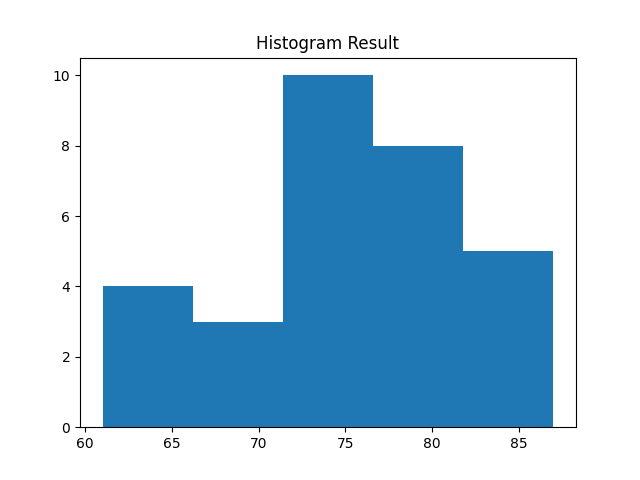
76.5, 77, 77.5, 78, 78.5, 79, 79.2, 80, 81, 82, 83, 84, 85, 87.])

plt.hist(a, bins =5)

plt.title("Histogram Result")

plt.show()

**OUTPUT**

****

**12. Write a program to implement KNN algorithm using iris data Set. Use different values for K and different values for text and training data.**

**CODE:**

print("Amrutha Biju")

print("21MCA003")

import pandas as pd

dataset=pd.read\_csv("iris.csv")

X = dataset.iloc[:,:1].values

y = dataset.iloc[:,4].values

print(X)

print(y)

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

X\_train,X\_test,y\_train,y\_test=train\_test\_split(X,y,test\_size=0.20)

from sklearn.neighbors import KNeighborsClassifier

classifier = KNeighborsClassifier(n\_neighbors=5)

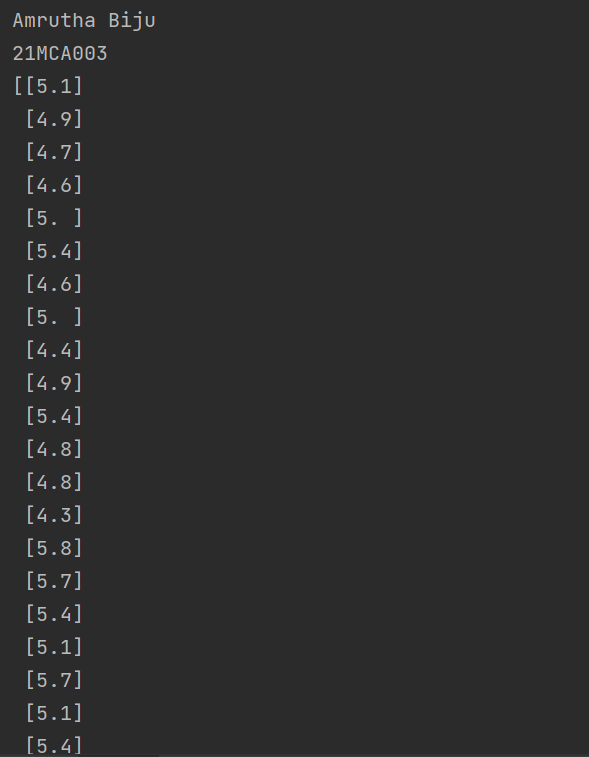
classifier.fit(X\_train,y\_train)

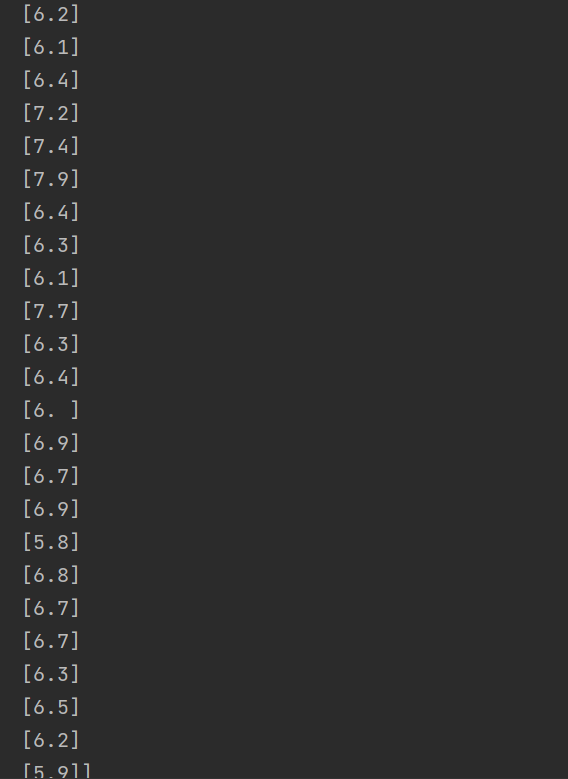
y\_pred = classifier.predict(X\_test)

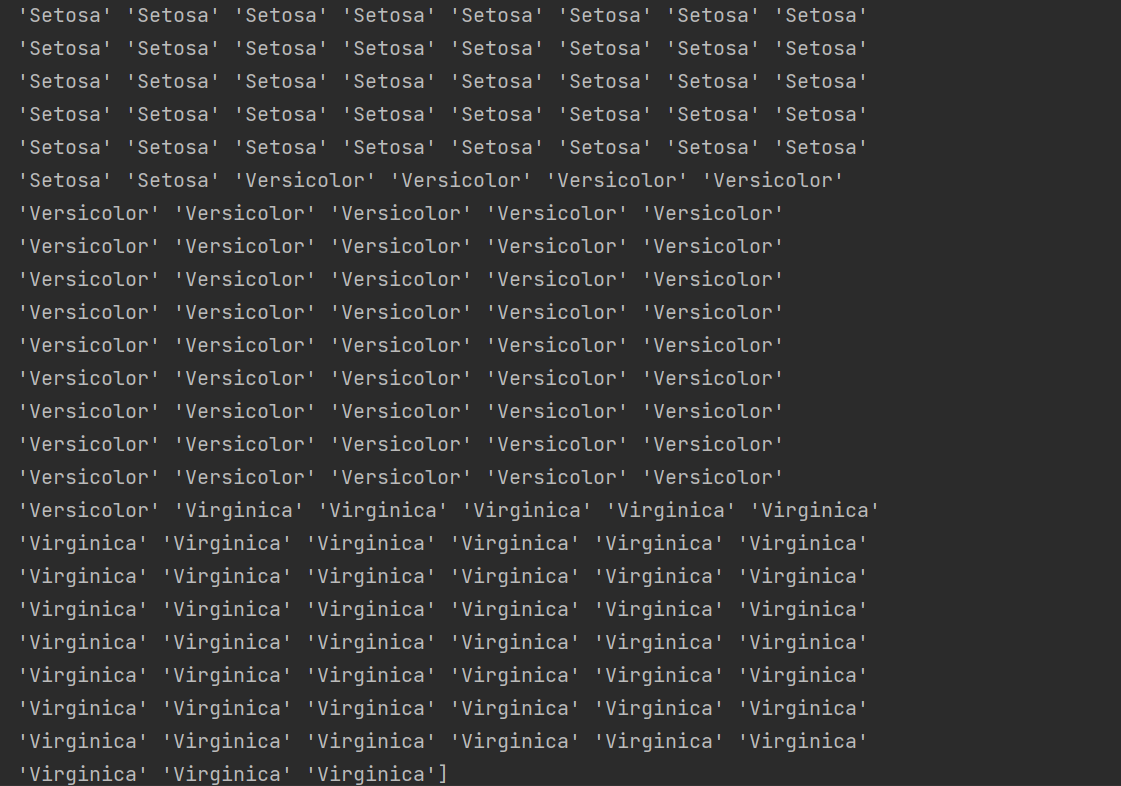
from sklearn.metrics import classification\_report,confusion\_matrix

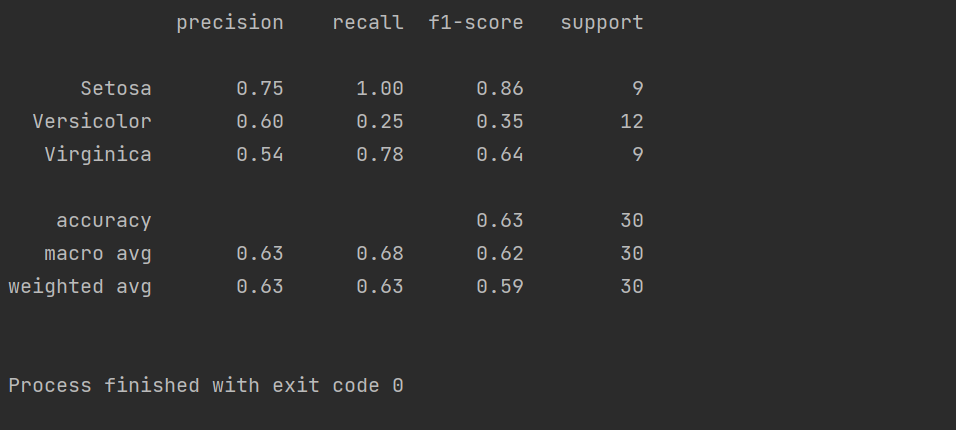
print(classification\_report(y\_test,y\_pred))

**OUTPUT**

****

****





**13. Write a program to implement naive bayes classification using different naive Bayes classification algorithms.**

**CODE:**

print("Amrutha Biju")

print("21MCA003")

import pandas as pd

dataset = pd.read\_csv('iris.csv')

X = dataset.iloc[:, :1].values

y = dataset.iloc[:, 4].values

print(X)

print(y)

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

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.20)

from sklearn.naive\_bayes import GaussianNB

classifier = GaussianNB()

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

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

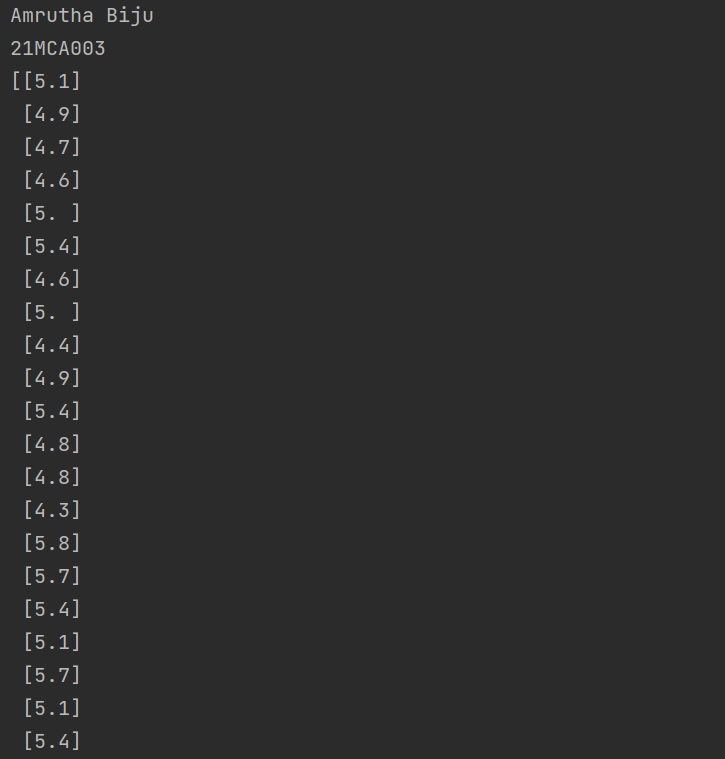
from sklearn.metrics import classification\_report,confusion\_matrix

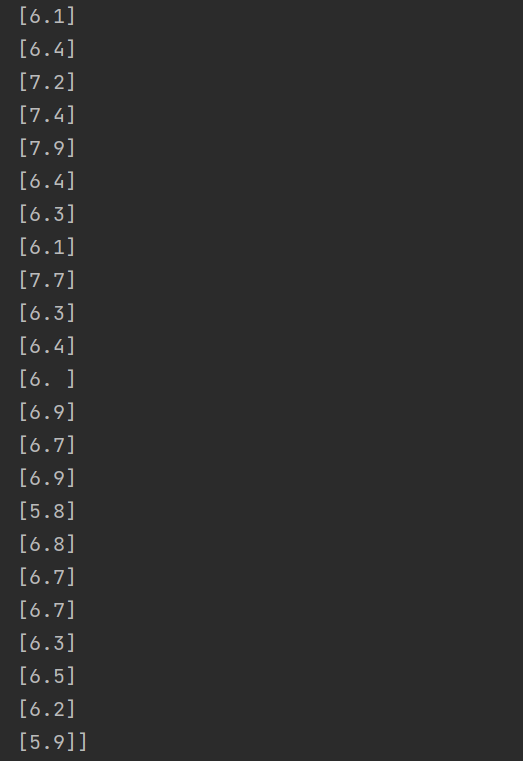
print(classification\_report(y\_test, y\_pred))

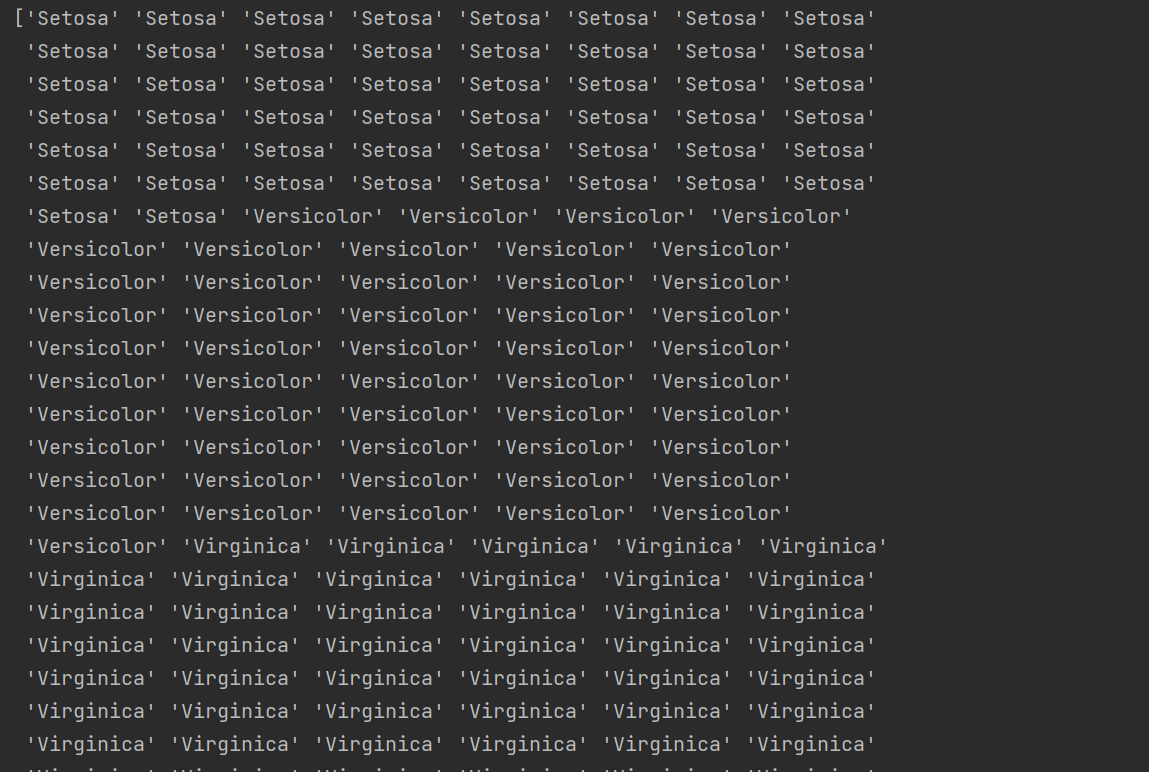
df = pd.DataFrame({'Real Values':y\_test, 'Predicted Values':y\_pred})

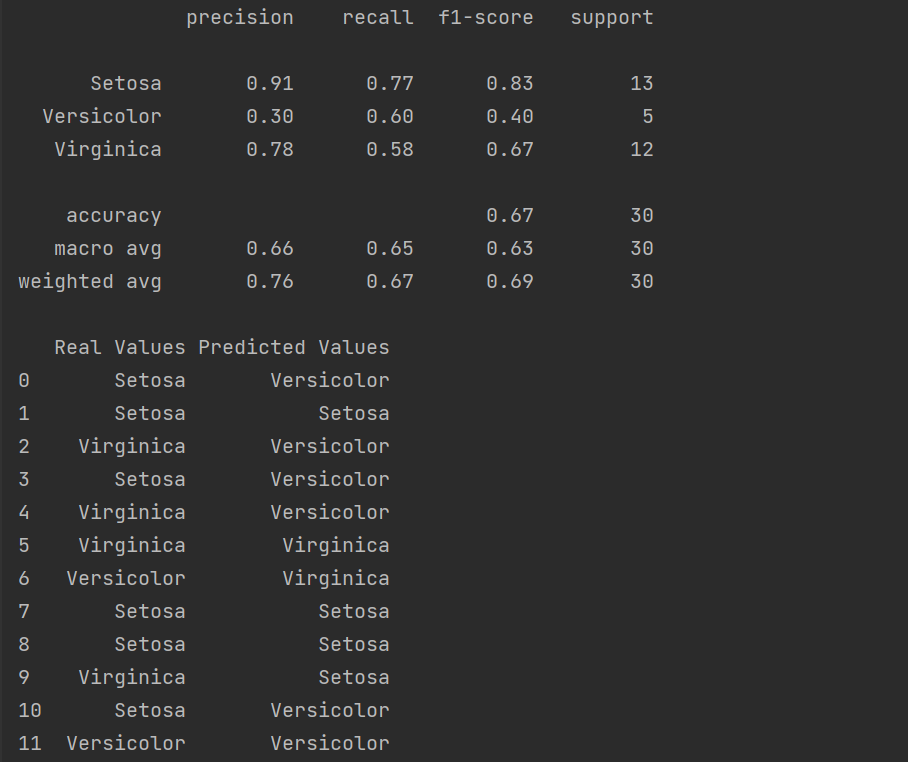
print(df)

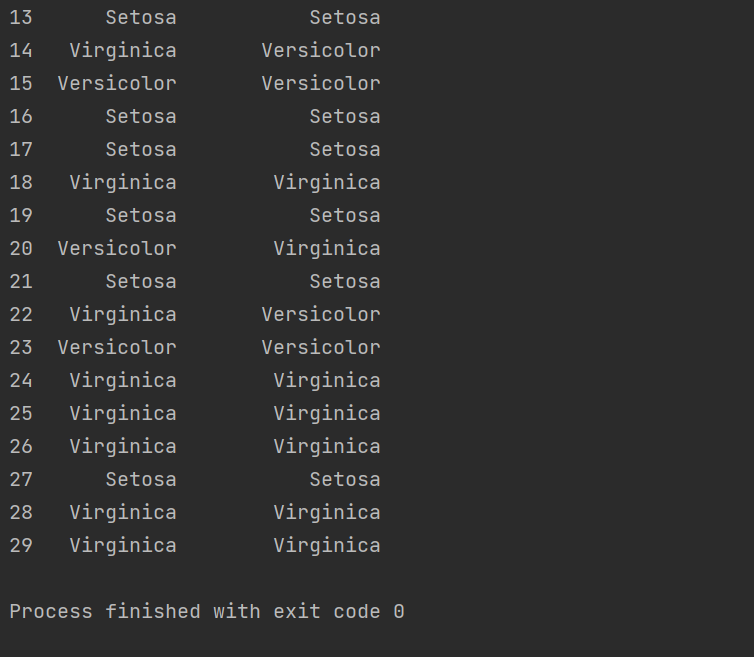
**OUTPUT**

****

****







**14. Write a program to implement decision tree algorithm using the given data set**

**CODE:**

print("Amrutha Biju")

print("21MCA003")

import numpy as np

import pandas as pd

import matplotlib.pyplot as plt

from sklearn import tree,metrics,model\_selection

data=pd.read\_csv('car.csv',names=['buying','main','doors','persons','lug\_boot','safety','class'])

data.head()

data.info()

data['class'],class\_names=pd.factorize(data['class'])

print(class\_names)

print(data['class'].unique())

data['buying'],\_ = pd.factorize(data['buying'])

data['main'],\_ = pd.factorize(data['main'])

data['doors'],\_ = pd.factorize(data['doors'])

data['persons'],\_ = pd.factorize(data['persons'])

data['lug\_boot'],\_ = pd.factorize(data['lug\_boot'])

data['safety'],\_ = pd.factorize(data['safety'])

data.head()

data.info()

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

y=data.iloc[:,-1]

x\_train,x\_test,y\_train,y\_test=model\_selection.train\_test\_split(x,y,test\_size=0.3,random\_state=0)

dtree=tree.DecisionTreeClassifier(criterion='entropy',max\_depth=3, random\_state=0)

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

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

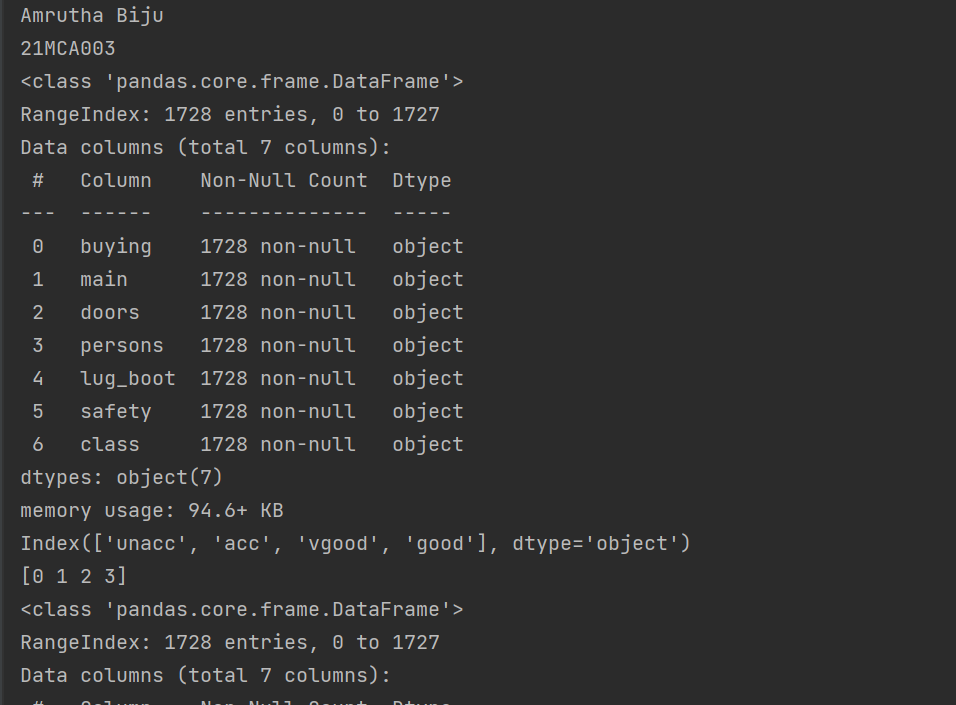
accuracy = metrics.accuracy\_score(y\_test,y\_pred)

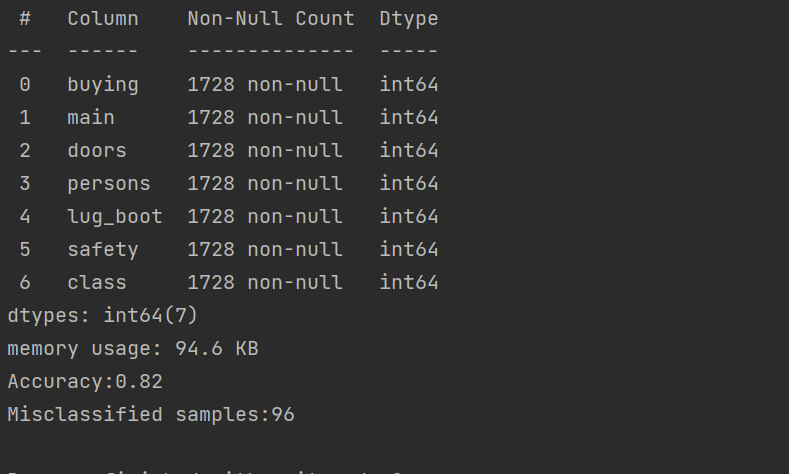
print('Accuracy:{:.2f}'.format(accuracy))

count\_misclassified = (y\_test != y\_pred).sum()

print('Misclassified samples:{}'.format(count\_misclassified))

**OUTPUT**

****

****

**15. Write a program to demonstrate Simple Linear Regression using given data set**

**CODE:**

import numpy as np

import pandas as pd

import matplotlib.pyplot as plt

student = pd.read\_csv('student\_scores.csv')

student.head()

x = student.iloc[:, :-1]

y = student.iloc[:, 1]

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

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

print(x\_train)

from sklearn.linear\_model import LinearRegression

regressor = LinearRegression()

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

print(regressor.intercept\_)

print(regressor.coef\_)

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

for (i, j) in zip(y\_test, y\_pred):

if i != j:

print("Actual value:", i, "predicted value:", j)

print("Number of mislabeled points from test data set", (y\_test != y\_pred).sum())

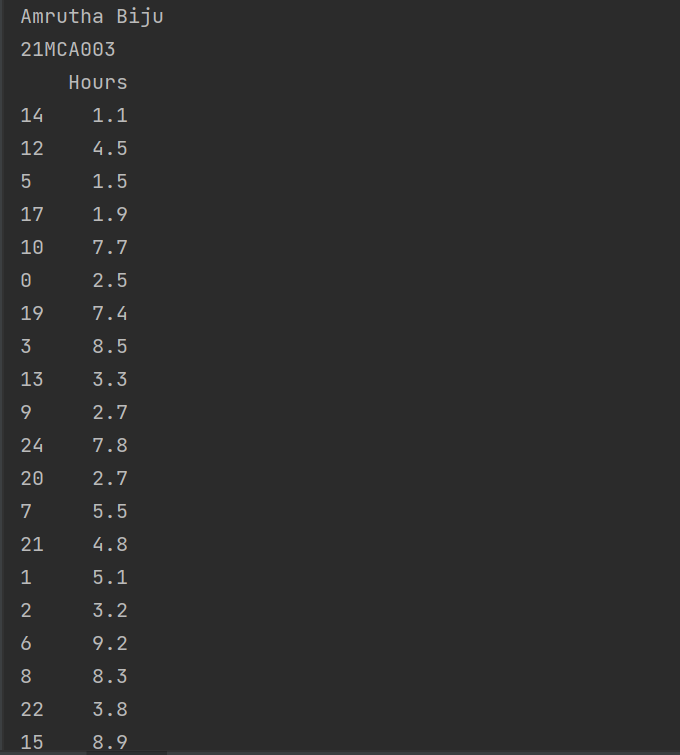
from sklearn import metrics

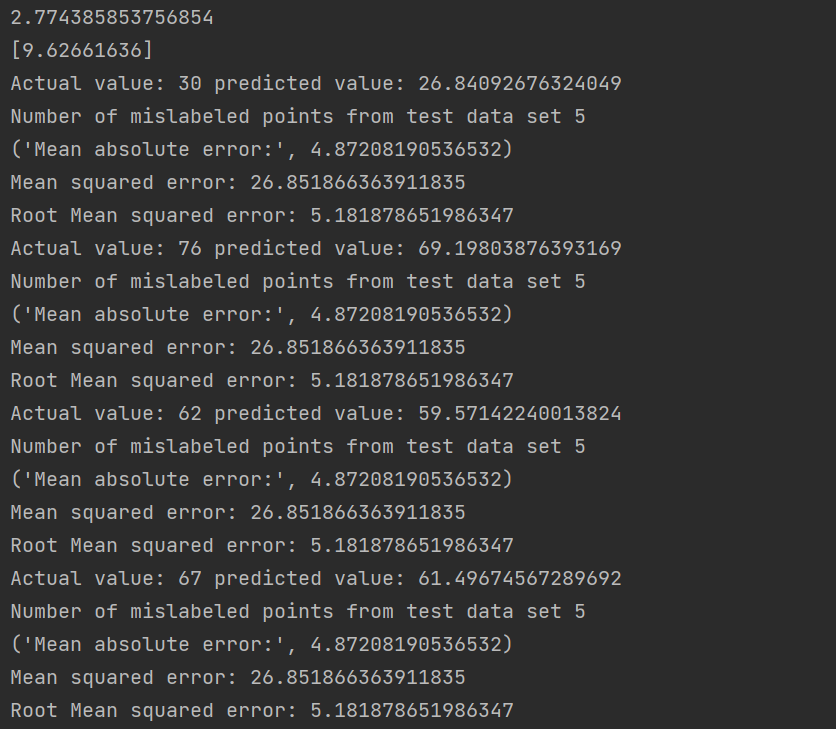
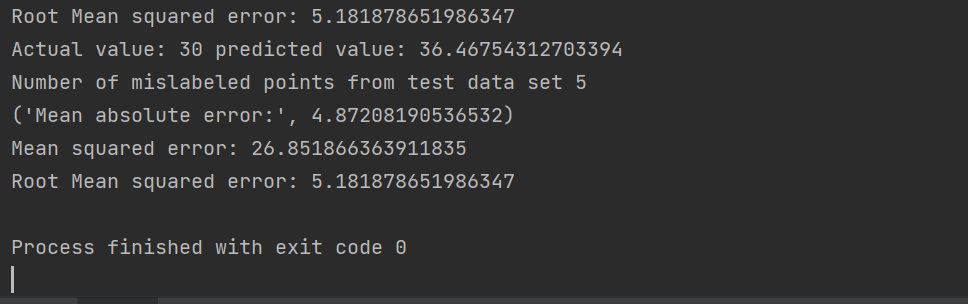
print(("Mean absolute error:", metrics.mean\_absolute\_error(y\_test, y\_pred)))

print("Mean squared error:", metrics.mean\_squared\_error(y\_test, y\_pred))

print("RootMeansquarederror:",np.sqrt(metrics.mean\_squared\_error(y\_test, y\_pred)))

**OUTPUT**

****



**16. Write a program to implement Multiple Linear Regression using appropriate data set**

**CODE:**

print("Amrutha Biju")

print("21MCA003")

import numpy as np

import pandas as pd

import matplotlib as plt

#import pd as pd

advertising=pd.read\_csv('Company\_data.csv')

advertising.head()

advertising.describe()

advertising.info()

x=advertising.iloc[:,:1]

print(x)

y=advertising.iloc[:,-1]

print(y)

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

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

print(x\_train)

from sklearn.linear\_model import LinearRegression

regressor = LinearRegression()

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

print(regressor.intercept\_)

print(regressor.coef\_)

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

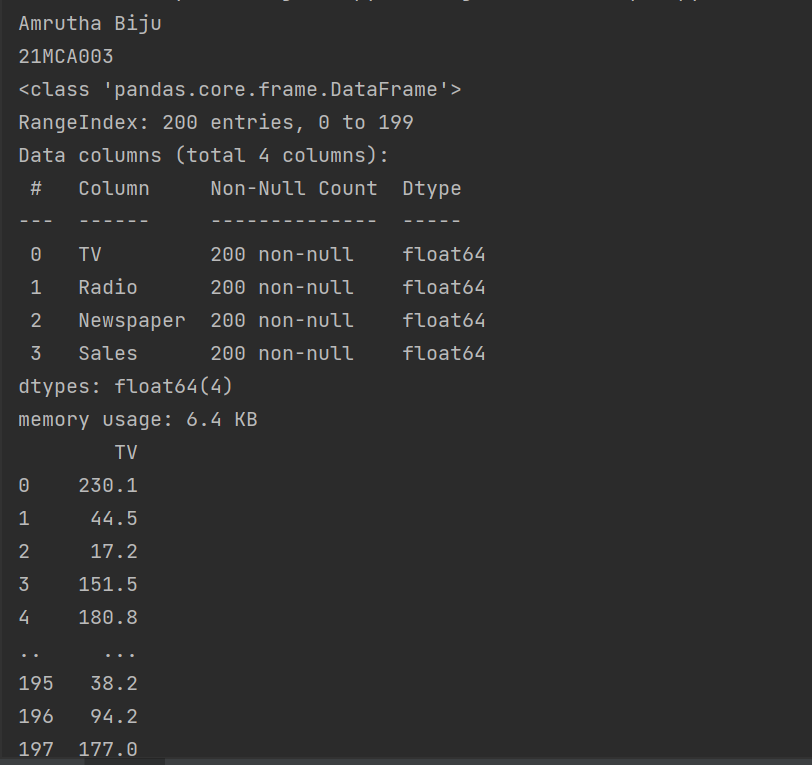
for(i,j) in zip(y\_test,y\_pred):

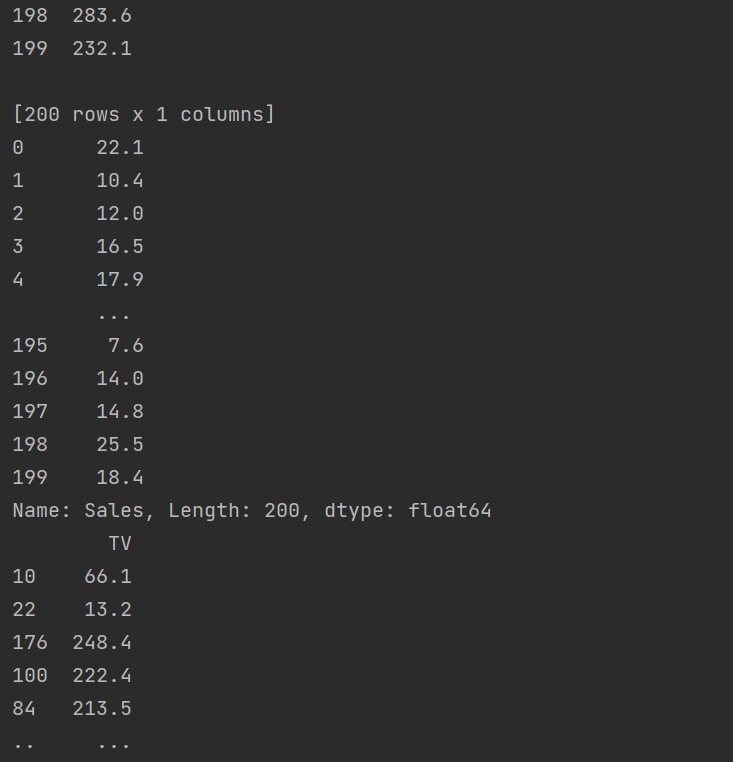
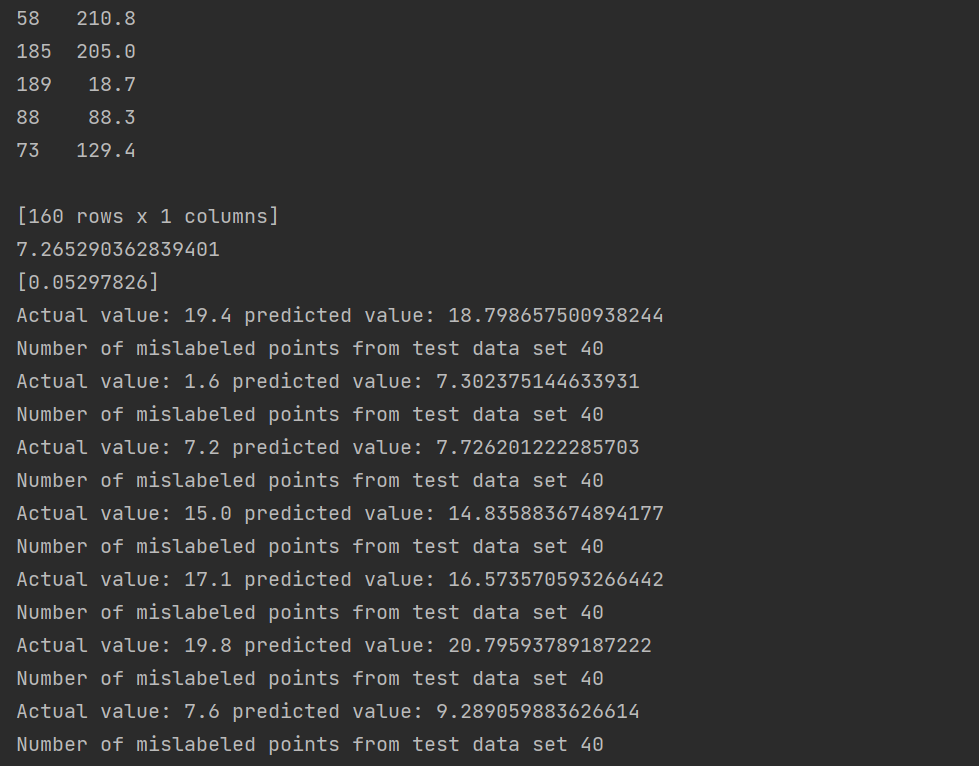
if i!=j:

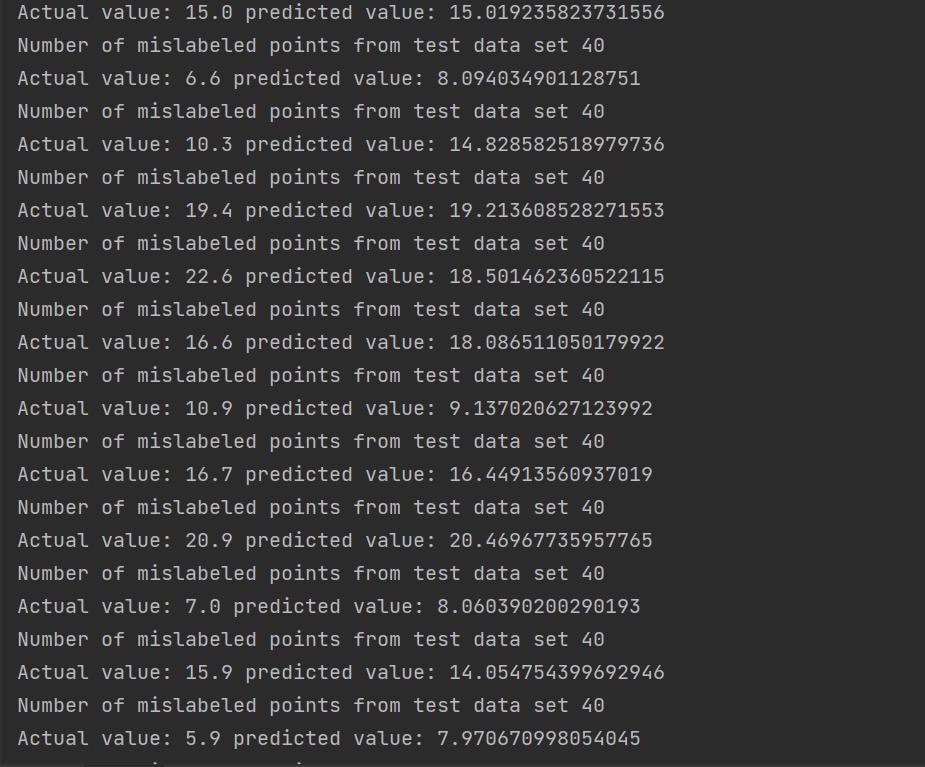
print("Actual value:",i,"predicted value:",j)

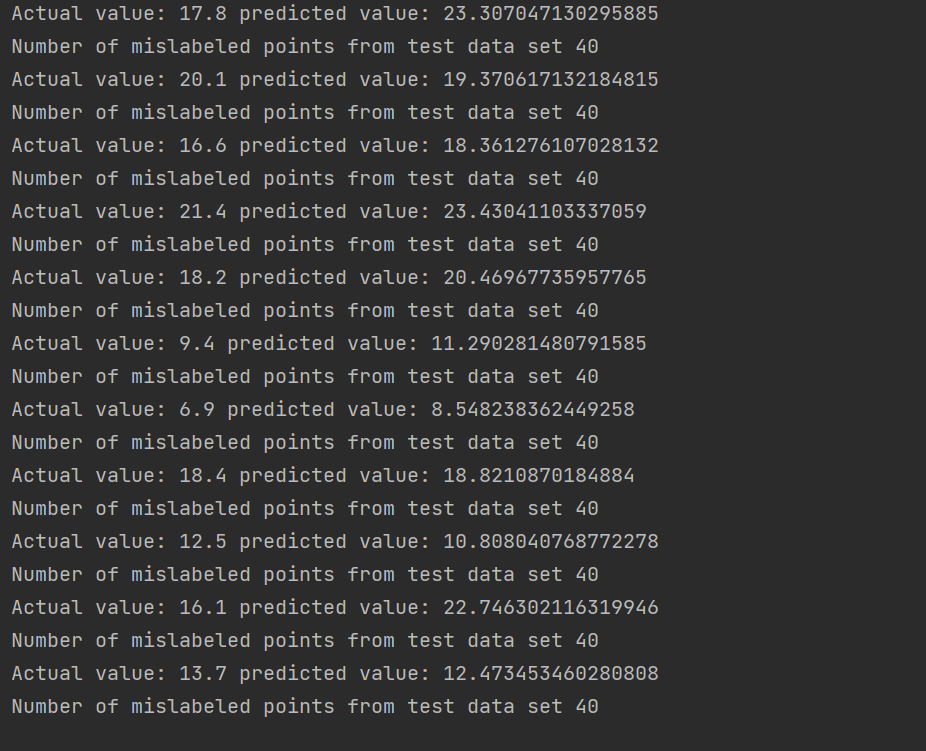
print("Numberofmislabeledpointsfromtestdata set",(y\_test!=y\_pred).sum())

**OUTPUT**

****

****





**17. Write a program to implement K –Means Clustering Algorithm with k=6. Create a scatter plot to visualize the same.**

**CODE:**

import pandas as pd

from matplotlib import pyplot as plt

from sklearn.cluster import KMeans

customers = pd.read\_csv('customer\_data.csv')

customers.head()

points = customers.iloc[:, 3:5].values

x = points[:, 0]

y = points[:, 1]

plt.scatter(x, y, s=50, alpha=0.7)

plt.xlabel('Annual Income (k$)')

plt.ylabel('Spending Score')

plt.show()

kmeans = KMeans(n\_clusters=6, random\_state=0)

kmeans.fit(points)

predicted\_cluster\_indexes = kmeans.predict(points)

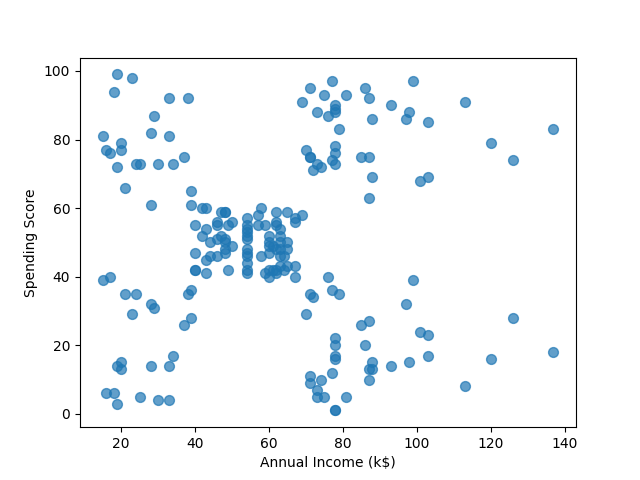
plt.scatter(x, y, c=predicted\_cluster\_indexes, s=50, alpha=0.7, cmap='viridis')

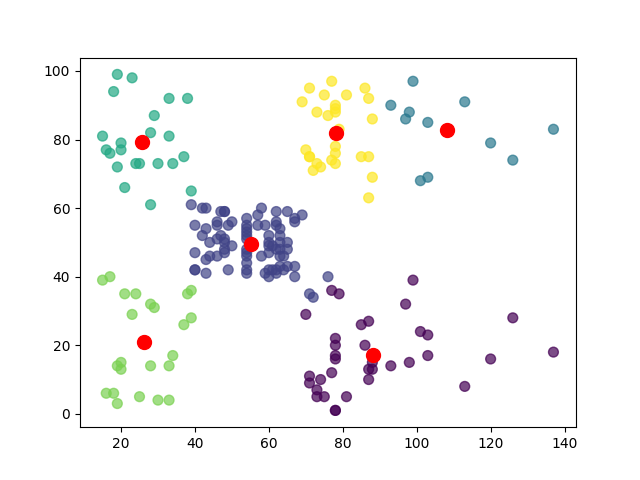
centers = kmeans.cluster\_centers\_

plt.scatter(centers[:, 0], centers[:, 1], c='red', s=100)

plt.show()

**OUTPUT**

****



**18. For given text:**

**1) perform word and sentence tokenization.**

**2) Remove the stop words from the given text**

**3) Perform Part of Speech tagging**

**4) create n-grams for different values of n=2,4.**

**CODE:**

print("Amrutha Biju")

print("21mca003")

import nltk

from nltk.corpus import stopwords

from nltk.tokenize import sent\_tokenize, word\_tokenize

text1 = "The data set given satisfies the requirement for model generation. This is used in Data Science Lab"

print("sentence tokenization:")

for i in sent\_tokenize(text1):

print(i)

print("word tokenization:")

for i in word\_tokenize(text1):

print(i)

text = word\_tokenize(text1)

print("parts of Speech:")

for i in nltk.pos\_tag(text):

print(i)

print("after removing stop words")

text = [word for word in text if word not in stopwords.words('english')]

print(text)

# 2 grams

print("2 grams are:")

temp = zip(\*[text[i:] for i in range(0, 2)])

ans = [' '.join(ngram) for ngram in temp]

print(ans)

# 4 grams

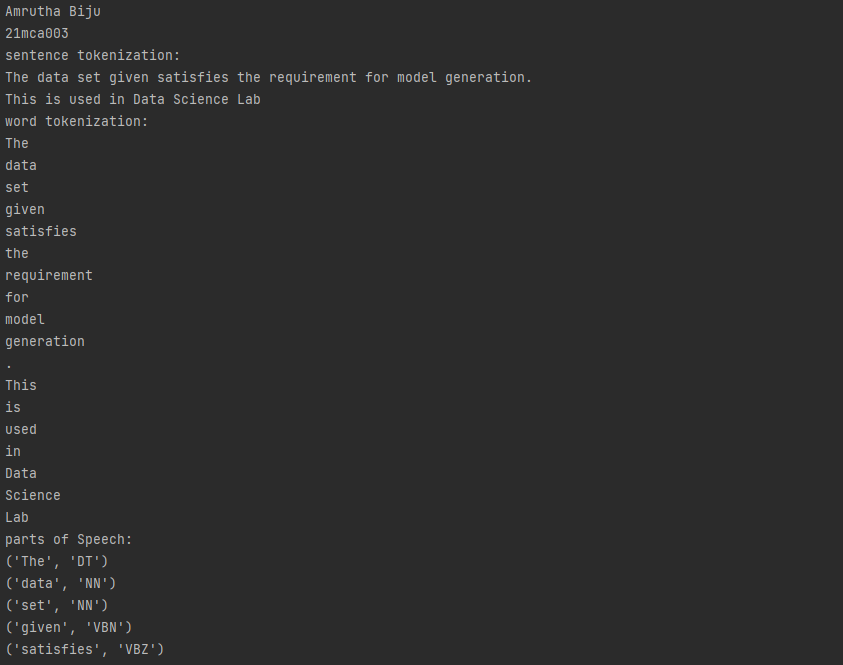
print("4 grams are:")

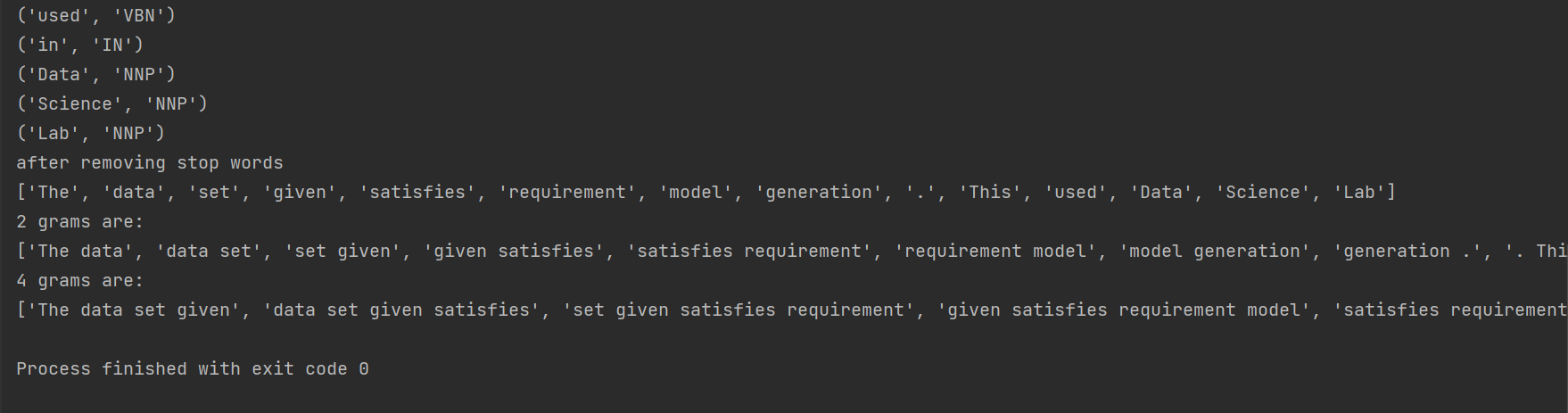
temp = zip(\*[text[i:] for i in range(0, 4)])

ans = [' '.join(ngram) for ngram in temp]

print(ans)

**OUTPUT**

****

****

**19. Write a program to perform chunking on given text by creating a chunk containing every word.**

**CODE:**

import nltk

nltk.download('averaged\_perceptron\_tagger')

from nltk.corpus import stopwords

sample\_text="Rama Killed Ravana to save Sita from Lanka.The Legend of Ramayan is the most popular Indian epic.A lot of Movies and serials have already been shot in several languages here in india based on the ramayan."

tokenized = nltk.sent\_tokenize(sample\_text)

for i in tokenized:

words = nltk.word\_tokenize(i)

tagged\_words=nltk.pos\_tag(words)

chunkGram=r"""chunk: {<.\*>+ }"""

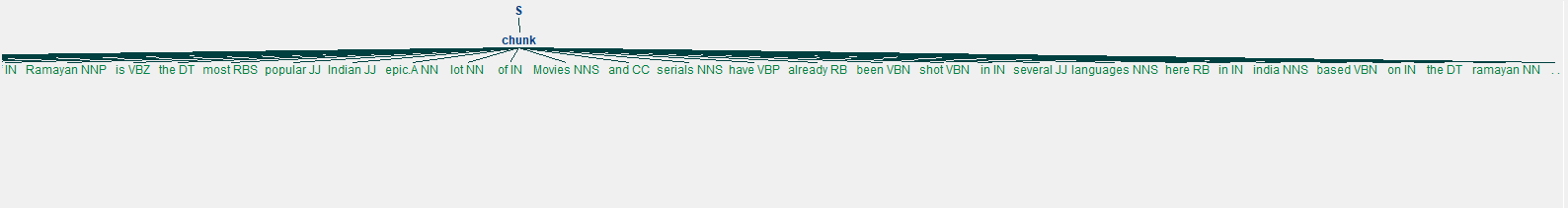
chunkParser=nltk.RegexpParser(chunkGram)

chunked=chunkParser.parse(tagged\_words)

print(chunked)

chunked.draw()

**OUTPUT**

****

**20. Write a program to create chunks using words in the given sentence -except Verbs(VB), determiner(DT) and propositions(IN)**

**CODE:**

import nltk

from nltk.corpus import stopwords

sample\_text="Rama Killed Ravana to save Sita from Lanka.The Legend of Ramayan is the most popular Indian epic.A lot of Movies and serials have already been shot in several languages here in india based on the ramayan."

tokenized=nltk.sent\_tokenize(sample\_text)

for i in tokenized:

words=nltk.word\_tokenize(i)

tagged\_words=nltk.pos\_tag(words)

chunkGram=r"""chunk: {<.\*>+}

}<VB.?|IN|DT|>{"""

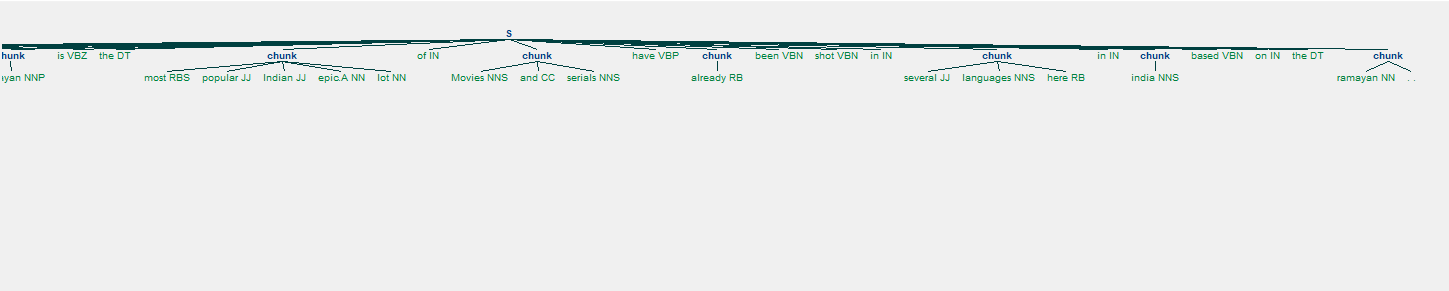
chunkParser=nltk.RegexpParser(chunkGram)

chunked=chunkParser.parse(tagged\_words)

print(chunked)

chunked.draw()

**OUTPUT**

****