

Date:24/11/2021

Program no:1

Aim : Perform all matrix operations using python (Using numpy)

Program

```
import numpy

array1 = []

n=int(input("Enter the array Size"))

for i in range(n):

    array1.append(int(input("Enter thr first array elements : ")))

array1=numpy.array(array1)

print(numpy.floor(array1))

array2=[]

for i in range(n):

    array2.append(int(input("Enter the second array elemensts : ")))

array2=numpy.array(array2)

print(numpy.floor(array2))

print("Array Addition ")

print(numpy.add(array1,array2))

print("Array Substraction ")

print(numpy.subtract(array1,array2))

print("Array Multiplication")

print(numpy.multiply(array1,array2))

print("Array Division")

print(numpy.divide(array1,array2))
```

```

print("Array Dot")

print(numpy.dot(array1,array2))

print("Array Squareroot")

print(numpy.sqrt(array1))

print("Array Summation of array1 ")

print(numpy.sum(array1))

print("Array Transpose of array1")

print(array1.T)

```

Output

```

C:\Users\ajcemca\AppData\Local\Programs\Python\Python39\python.exe C:/Users/ajcemca/PycharmProjects/Sam/numpymatrix.py
Enter the array Size 3
Enter thr first array elements : 4
Enter thr first array elements : 6
Enter thr first array elements : 10
[ 4.  6. 10.]
Enter the second array elemensts : 2
Enter the second array elemensts : 3
Enter the second array elemensts : 5
[2. 3. 5.]
Array Addition
[ 6  9 15]
Array Substraction
[2 3 5]
Array Multiplication
[ 8 18 50]
Array Division
[2. 2. 2.]
Array Dot
76
Array Squareroot
[2.          2.44948974 3.16227766]
Array Summation of array1
20
Array Transpose of array1
[ 4  6 10]

```

Date:01/12/2021

Program no : 2

Aim: Perform SVD (Singular Value Decomposition) Using python.

Program

```
from numpy import array

from scipy.linalg import svd

A1= array([[2,13,3],[5,9,10],[8,7,3],[26,18,30],[22,31,45]])

print(A1)

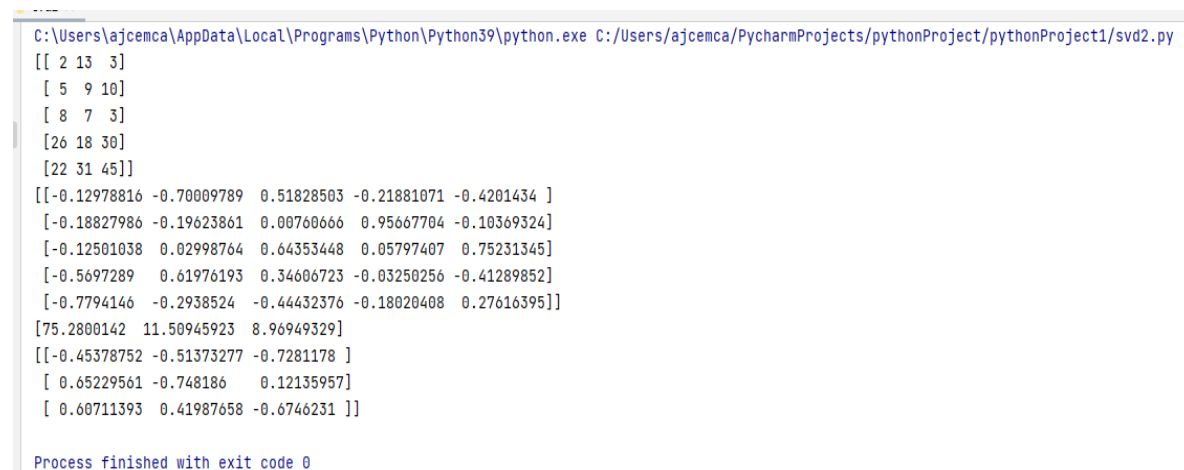
a,b,c=svd(A1)

print(a)

print(b)

print(c)
```

Output



```
C:\Users\ajcemca\AppData\Local\Programs\Python\Python39\python.exe C:/Users/ajcemca/PycharmProjects/pythonProject/pythonProject1/svd2.py
[[ 2 13  3]
 [ 5  9 10]
 [ 8  7  3]
 [26 18 30]
 [22 31 45]]
[[-0.12978816 -0.70009789  0.51828503 -0.21881071 -0.4201434 ]
 [-0.18827986 -0.19623861  0.00760666  0.95667704 -0.10369324]
 [-0.12501038  0.02998764  0.64353448  0.05797407  0.75231345]
 [-0.5697289   0.61976193  0.34606723 -0.03250256 -0.41289852]
 [-0.7794146  -0.2938524  -0.44432376 -0.18020408  0.27616395]]
[75.2800142  11.50945923  8.96949329]
[[-0.45378752 -0.51373277 -0.7281178 ]
 [ 0.65229561 -0.748186   0.12135957]
 [ 0.60711393  0.41987658 -0.6746231 ]]
Process finished with exit code 0
```

Date:01/12/2021

Program no : 3

Aim : Program to implement K-NN classification using any standard dataset available in public domain and find the accuracy of the algorithm.

Program

```

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

irisData=load_iris()

a=irisData.data

b=irisData.target

a_train,a_test,b_train,b_test=train_test_split(a,b,test_size=0.6,random_state=10)

knn=KNeighborsClassifier(n_neighbors=3)

knn.fit(a_train,b_train)

print(knn.predict(a_test))

x=knn.predict(a_test)

z=accuracy_score(b_test,x)

print(z)

```

Output

```

C:\Users\ajcemca\AppData\Local\Programs\Python\Python39\python.exe C:/Users/ajcemca/PycharmProjects/pythonProject/pythonProject1/knn.py
[1 2 0 1 0 1 2 1 0 1 1 2 1 0 0 2 2 0 0 0 2 2 2 0 1 0 1 1 1 2 1 1 2 2 0 2
 2 2 2 0 0 1 0 2 0 1 2 2 2 1 2 1 1 1 0 0 1 0 2 0 0 1 1 2 0 2 0 1 2 0 2 2 2
 2 2 0 1 2 1 0 2 1 1 0 0 0 1 2 2]
0.9333333333333333

Process finished with exit code 0

```

Date:01/12/2021

Program no : 4

Aim : Program to implement K-NN classification using any random dataset without using inbuilt packages.

Program

```
from math import sqrt

def euclidean_distance(row1,row2):

    distance = 0.0

    for i in range(len(row1) - 1):

        distance += (row1[i] - row2[i] )**2

    return sqrt(distance)


def get_neighbors(train,test_row, num_neighbors):

    distances = list()

    for train_row in train:

        dist =euclidean_distance(test_row, train_row)

        distances.append((train_row,dist))

        distances.sort(key=lambda tup:tup[1])

    neighbors = list()

    for i in range(num_neighbors):

        neighbors.append(distances[i][0])

    return neighbors


def predict_classification(train,test_row, num_neighbors):

    neighbors = get_neighbors(train, test_row, num_neighbors)
```

```

output_values = [row[-1] for row in neighbors]

prediction = max(set(output_values), key=output_values.count)

return prediction

```

```

dataset = [[2.7810836, 2.550537003,0],
           [1.465458936,2.64785645,0],
           [3.56789536,4.568555858,0],
           [1.468956556,3.1464756654,0],
           [5.135663212,2.621254545,0],
           [6.2545449552,5.1436870564,1],
           [8.4365631212,7.56655252636,1],
           [2.146589696,5.66655665555,1],
           [3.4664565252,5.46558866,1],
           [5.895525255,3.46565858,1]]

```

```

prediction = predict_classification(dataset,dataset[0], 5)

print('expected %d, Got %d. ' % (dataset[0][-1], prediction))

```

Output

```

C:\Users\ajcemca\AppData\Local\Programs\Python\Python39\python.exe C:/Users/ajcemca/PycharmProjects/pythonProject/pythonProject1/knn_func.py
expected 0, Got 0.

Process finished with exit code 0

```

Date:08/12/2021

Program no : 5

Aim : Program to implement Naïve Bayes algorithm classification using any standard dataset available in the public domain and find the accuracy of the algorithm

Program

```
import numpy as np

import pandas as pd

import matplotlib.pyplot as plt

dataset = pd.read_csv('Social_Network_Ads.csv')

x = dataset.iloc[:, [2,3]].values

y = dataset.iloc[:, -1].values

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,random_state = 20)

from sklearn.preprocessing import StandardScaler

sc=StandardScaler()

x_train = sc.fit_transform(x_train)

x_test = sc.transform(x_test)

print(x_train)

print(x_test)

from sklearn.naive_bayes import GaussianNB

classifier = GaussianNB()

classifier.fit(x_train,y_train)

y_pred = classifier.predict(x_test)

print(y_pred)

from sklearn.metrics import confusion_matrix, accuracy_score
```

```
ac = accuracy_score(y_test,y_pred)

cm = confusion_matrix(y_test,y_pred)

print(ac)

print(cm)
```

Output

```
C:\Users\ajcemca\AppData\Local\Programs\Python\Python39\python.exe C:/Users/ajcemca/PycharmProjects/pythonProject/pythonProject1/naivebayers.py
[[-0.80276277  0.44295604]
 [-0.70800656  1.43671337]
 [-0.23422551 -0.5508013 ]
 [ 0.90284902  1.16568865]
 [-1.0870314   0.4730699 ]
 [-0.89751898 -1.09285075]
 [-0.51849414  0.95489164]
 [ 1.47138628  0.41284218]
 [-1.46605624  0.38272832]
 [-1.75032487 -1.48433091]
 [-0.42373793 -1.12296461]
 [ 0.99760523 -1.00250917]
 [-0.23422551 -1.24342004]
 [ 0.90284902 -1.36387548]
 [ 0.52382418  1.82819354]
 [-1.65556866 -0.97239532]
 [ 0.14479933  0.20204517]
 [ 0.05004312 -0.5508013 ]
 [-0.61325035  0.17193131]
 [-0.32898172 -0.76159831]
 [-0.1394693  1.49694109]
 [ 1.37663007 -1.42418310]

[ 0.52382418  1.31625794]
[-1.84508108  0.53329761]
[ 1.94516733  2.27990141]
[-0.1394693  -1.06273689]
[-1.37130003 -1.45421705]
[ 2.03992354  1.85830739]
[-0.80276277 -0.76159831]]
[[0 1 0 0 0 0 1 0 1 1 0 1 1 0 0 0 0 0 1 0 0 0 0 0 1 0 0 1 0 0 0 0 0 1 0 1 0
 0 1 0 0 0 1 1 1 0 0 0 1 1 0 1 1 0 0 1 0 0 1 1 1 0 0 0 0 0 1 0 0 0 0 1 0 0 1
 0 1 0 0 1 0]
0.8875
[[44  1]
 [ 8 27]]

Process finished with exit code 0
```


Date:08/01/2022

Program no : 6

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

Program

```
import numpy as np

from sklearn.linear_model import LinearRegression

x=np.array([5,12,25,35,45,55]).reshape((-1,1))

y=np.array([5,20,16,32,22,38])

print(x)

model=LinearRegression()

model.fit(x,y)

r_sq=model.score(x,y)

print('coefficent of determination',r_sq)

print("intercept",model.intercept_)

print("slope",model.coef_)

y_pred=model.predict(x)

print('predicted value',y_pred)

import matplotlib.pyplot as plt

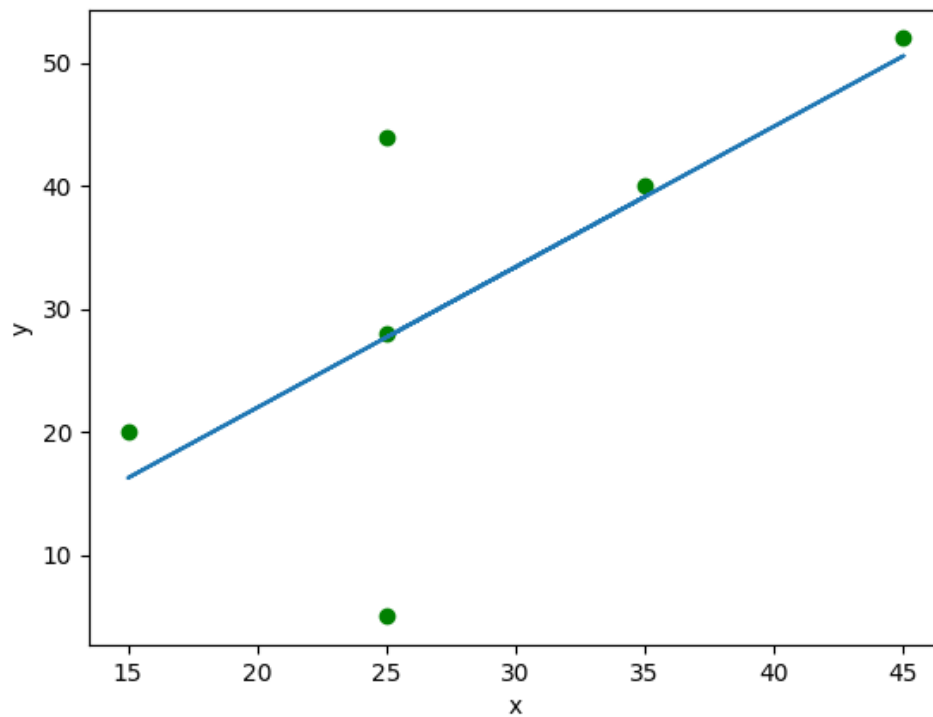
plt.scatter(x,y)

plt.plot(x,y_pred)
```

Output

```
C:\Users\ajcemca\AppData\Local\Programs\Python\Python39\python.exe C:/Users/ajcemca/PycharmProjects/pythonProject/pythonProjec
[[ 5]
 [15]
 [25]
 [35]
 [45]
 [55]]
[ 5 20 14 32 22 38]
Coefficient of determination : 0.7158756137479542
Intercept : 5.633333333333329
Slope : [0.54]
Predicting Responce : [ 8.33333333 13.73333333 19.13333333 24.53333333 29.93333333 35.33333333]

Process finished with exit code 0
|
```



Date:15/01/2022

Program no : 7

Aim: Program to implement Linear and Multiple regression techniques using any standard dataset available in public domain and evaluate its performance

Program

```
import numpy as np

import matplotlib.pyplot as plt

def estimate_coef(x, y):

    n = np.size(x)

    m_x = np.mean(x)

    m_y = np.mean(y)

    SS_xy = np.sum(y*x) - n*m_y*m_x

    SS_xx = np.sum(x*x) - n*m_x*m_x

    b_1 = SS_xy / SS_xx

    b_0 = m_y - b_1*m_x

    return (b_0, b_1)

def plot_regression_line(x, y, b):

    plt.scatter(x, y, color = "m",

               marker = "o", s = 30)

    y_pred = b[0] + b[1]*x

    plt.plot(x, y_pred, color = "g")
```

```
plt.xlabel('x')
```

```
plt.ylabel('y')
```

```
plt.show()
```

```
def main():
```

```
    x = np.array([0, 1, 2, 3, 4, 5, 6, 7, 8, 9])
```

```
    y = np.array([1, 3, 2, 5, 7, 8, 8, 9, 10, 12])
```

```
    b = estimate_coef(x, y)
```

```
    print("Estimated coefficients:\nb_0 = {} \
```

```
        \nb_1 = {}".format(b[0], b[1]))
```

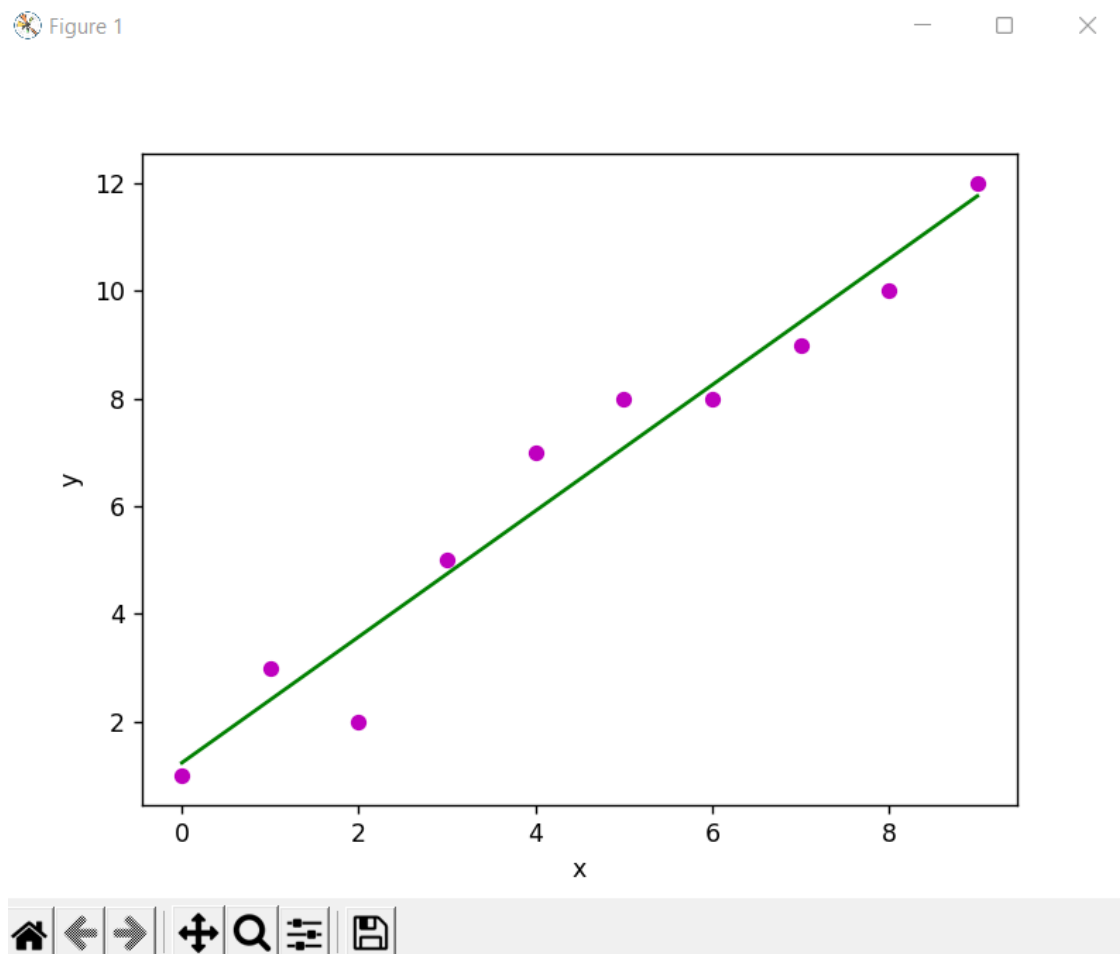
```
    plot_regression_line(x, y, b)
```

```
if __name__ == "__main__":
```

```
    main()
```

Output

```
C:\Users\ajcemca\AppData\Local\Programs\Python\Python39\python.exe
Estimated Coefficients :
    b_0 =2.0
    b_1 =1.0
|
```



Date:15/01/2022**Program no: 8**

Aim: Program to implement Linear and Multiple regression techniques using cars dataset available in public domain and evaluate its performance.

Program

```
import pandas

df=pandas.read_csv("cars.csv")

x=df[['Weight','Volume']]

y=df['CO2']

from sklearn import linear_model

regr=linear_model.LinearRegression()

regr.fit(x,y)

predictedco2=regr.predict([[2300,1300]])

print(predictedco2)
```

Output

```
C:\Users\ajcemca\PycharmProjects\pythonProject\venv\Scripts\python.exe C:/Users
[107.2087328]
|
Process finished with exit code 0
```

Date:15/01/2022**Program : 9**

Aim: Program to implement multiple linear regression techniques using Boston dataset available in the public domain and evaluate its performance and plotting graph.

Program

```
import matplotlib.pyplot as plt

from sklearn import datasets,linear_model,metrics

boston=datasets.load_boston()

x=boston.data

y=boston.target

from sklearn.model_selection import train_test_split

x_train,x_test,y_train,y_test=train_test_split( x,y,test_size=0.4,random_state=1)

reg=linear_model.LinearRegression()

reg.fit(x_train,y_train)

pre=reg.predict(x_test)

print("Prediction : ",pre)

print('Coefficients: ',reg.coef_)

print('Variance Score:{ }'.format(reg.score(x_test,y_test)))
```

Output

```
Prediction [23.67134888 21.74030058 18.99758813 25.48956674 33.81448    31.89278927
 10.78199818 11.76948128 41.56280351 25.3473233 24.372034 14.11450302
 12.40701286 27.20234921 23.43737274 32.84466914 23.60842764 32.59259513
 23.85004238 19.7314527 20.45440867 27.35630269 12.76780542 25.66660897
 29.72807157 22.15357852 30.19079303 16.04994832 28.39033945 14.35377707
 15.2366864 38.21896987 13.04273791 27.6805403 26.50783112 25.62615724
 25.38748819 6.18021821 23.83088365 39.44655193 17.18940017 35.30999805
 24.69974917 8.64460222 19.30224429 24.60568703 27.33660161 8.98033395
 23.69649295 19.01882711 29.95786303 24.28410874 25.0816296 21.47294993
 28.49968875 8.70205993 24.15032296 20.61644917 14.02028429 24.78978901
 28.82338698 25.38068056 23.46418288 22.310027 18.46758923 31.59220845
 15.36426251 20.47409342 30.44748162 36.1446885 32.21339274 16.96687201
 24.11087371 19.05568856 22.82152205 10.98287887 6.17771783 9.24239709
 15.64598693 20.63845804 33.63461858 24.08390253 32.35812922 26.61566412
 26.31571754 21.9784763 32.83918789 13.11679982 22.05714618 26.07450531
 20.03929014 30.62756585 31.36584349 27.00634194 31.2587809 10.26099249
 32.51975394 14.547433 29.30997208 24.76376686 20.66108026 21.37869093
 22.68461713 34.77810512 43.99628487 15.70849322 17.52320688 27.80934545
 20.75472888 17.49201269 18.03966614 35.70078381 12.76119622 38.97422809
 7.68925202 19.4931839 41.32259643 28.48332245 39.14616263 13.84700912
 23.11796205 21.25534719 26.62699789 37.98747407 23.72747822 30.68819329

-----
22.4086301 14.21437607 31.65769069 15.32061751 17.74804611 9.69417308
24.26986084 22.15086613 10.01444493 39.73186967 22.97980241 11.67661997
21.06713213 1.50437144 19.73119136 17.42220953 20.10684724 27.64309473
35.94944206 18.39398149 13.3431956 22.96380391 16.99239185]
Coefficeint : [-1.18523476e-01 3.79842025e-02 5.01458926e-03 3.46020042e+00
-1.38245151e+01 3.77221309e+00 -2.63157836e-02 -1.40793915e+00
 2.64788312e-01 -1.08549788e-02 -7.87130516e-01 1.00651827e-02
-4.66957118e-01]
Variance scores : 0.7465776075198429

Process finished with exit code 0
```


Date:15/01/2022

Program no:10

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

Program

```
import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

import seaborn as sns

from sklearn.preprocessing import LabelEncoder

from sklearn.model_selection import train_test_split

from sklearn.tree import DecisionTreeClassifier

from sklearn.metrics import classification_report, confusion_matrix

from sklearn.tree import plot_tree


df = sns.load_dataset('iris')

print(df.head())

print(df.info())

df.isnull().any() #return value true if any fields are null otherwise false.boolean value

print(df.shape)

sns.pairplot(data=df, hue = 'species')

plt.savefig("pne.png")


sns.heatmap(df.corr())

plt.savefig("one.png")
```

```
target=df['species']

df1 = df.copy()

df1 = df1.drop('species',axis=1)

print(df1.shape)

print(df1.head())

X = df1

print(target)


le = LabelEncoder()

target = le.fit_transform(target)

print(target)

y = target

X_train, X_test, Y_train, Y_test = train_test_split(X ,y , test_size = 0.2, random_state =
42)

print("Training split input- ",X_train.shape)

print("Training split input- ",X_test.shape)


dtree=DecisionTreeClassifier()

dtree.fit(X_train,Y_train)


print('Decision Tree Classifier Created')

y_pred =dtree.predict(X_test)

print("classification report - \n", classification_report(Y_test,y_pred))

cm = confusion_matrix(Y_test, y_pred)

plt.figure(figsize=(5,5))
```

```
sns.heatmap(data=cm,linewidths=.5, annot = True,square = True, cmap = 'Blues')
```

```
plt.ylabel('Actual label')
```

```
plt.xlabel('predicted label')
```

```
all_sample_title = 'Accuracy Score: {0}'.format(dtree.score(X_test, Y_test))
```

```
plt.title(all_sample_title, size = 15)
```

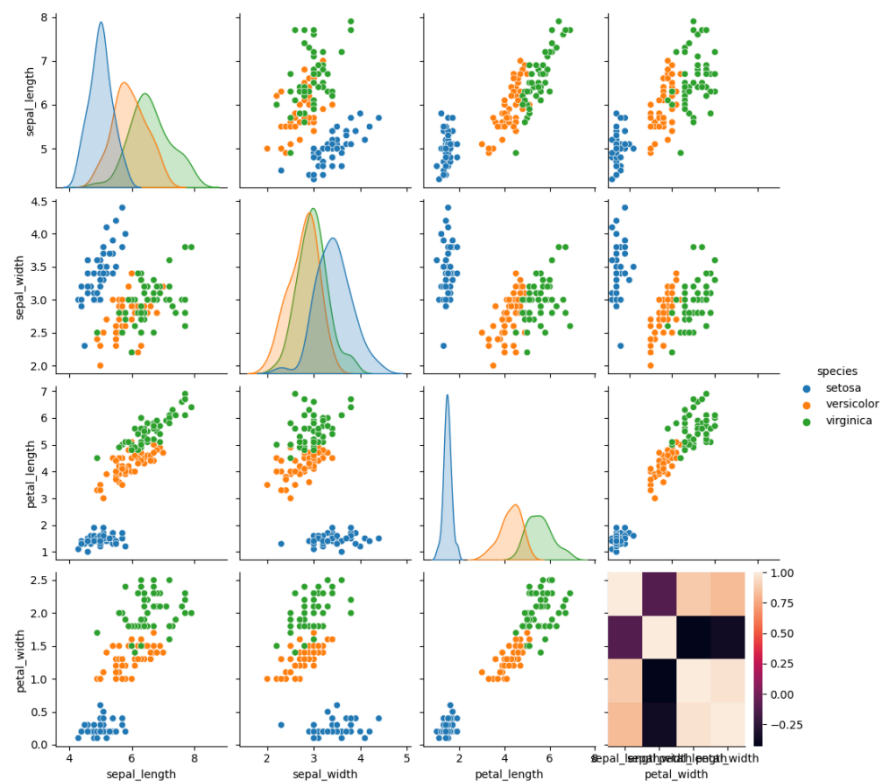
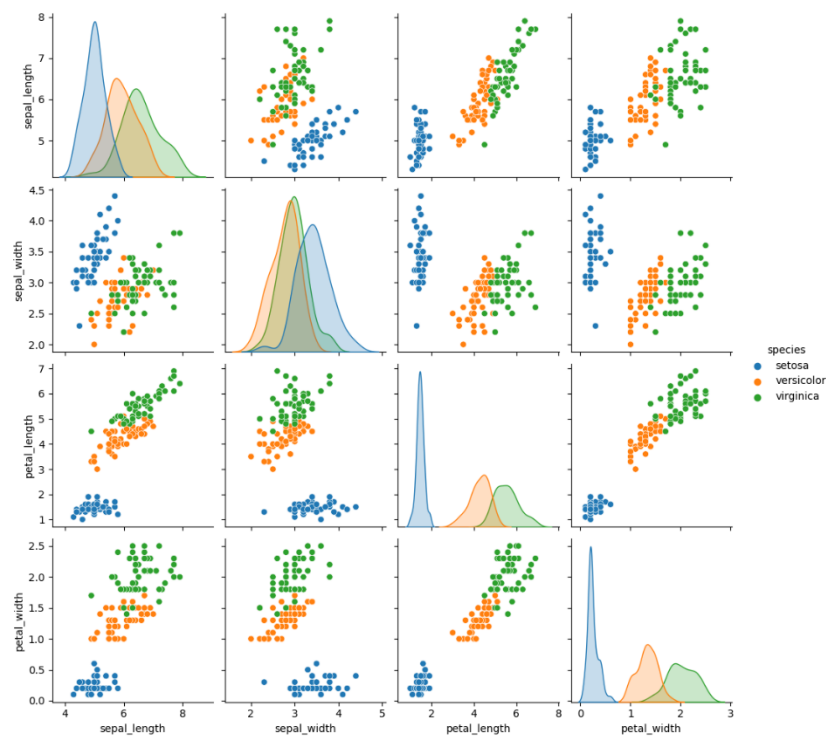
```
plt.savefig("two.png")
```

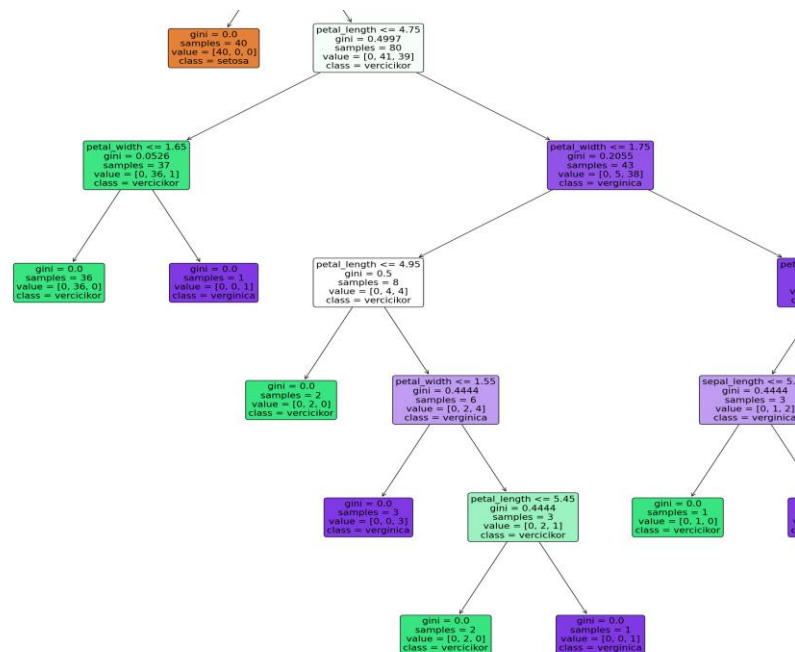
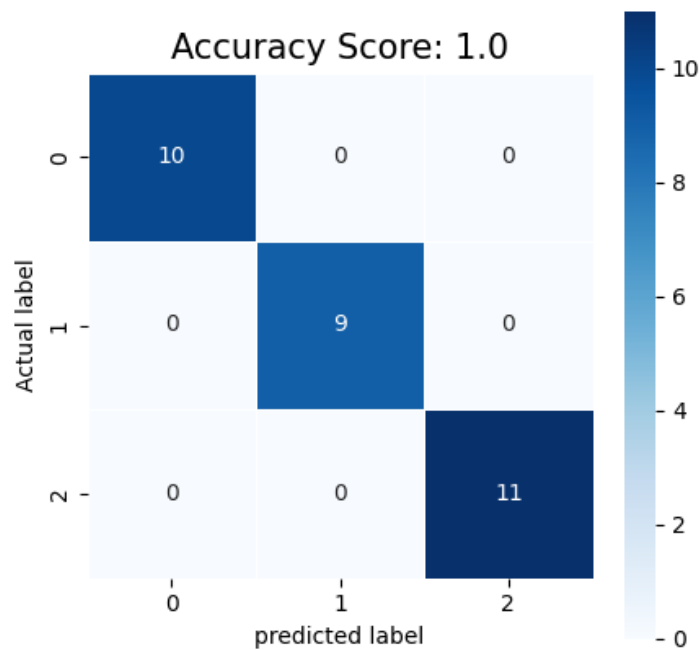
```
plt.figure(figsize = (20,20))
```

```
dec_tree = plot_tree(decision_tree=dtree, feature_names = df1.columns,class_names =  
["setosa", "verginica"],filled = True, precision =4 ,rounded =True)
```

```
plt.savefig("three.png")
```

Output





20MCA241 Data Science Lab

Date:05/01/2022**Program no:11**

Aim: Program to implement K-Means clustering technique using any standard dataset available in the public domain.

Program

```
import numpy as nm

import matplotlib.pyplot as mtp

import pandas as pd

dataset = pd.read_csv('Mall_Customers.csv')

x = dataset.iloc[:,[3, 4]].values

print(x)

from sklearn.cluster import KMeans

wcss_list = []

for i in range(1, 11):

    kmeans = KMeans(n_clusters=i, init='k-means++', random_state=42)

    kmeans.fit(x)

    wcss_list.append(kmeans.inertia_)

mtp.plot(range(1, 11), wcss_list)

mtp.title('Elbow method graph')

mtp.xlabel('Number of clusters(k)')

mtp.ylabel('wcss_list')

mtp.show()

kmeans = KMeans(n_clusters=5, init='k-means++', random_state=42)

y_predict = kmeans.fit_predict(x)
```

```
print(y_predict)

#visualising the cluster

mtp.scatter(x[y_predict == 0,0], x[y_predict == 0,1], s=100, c= 'blue', label =
'cluster1')

mtp.scatter(x[y_predict == 1,0], x[y_predict == 1,1], s=100, c= 'green', label =
'cluster2')

mtp.scatter(x[y_predict == 2,0], x[y_predict == 2,1], s=100, c= 'red', label = 'cluster3')

mtp.scatter(x[y_predict == 3,0], x[y_predict == 3,1], s=100, c= 'cyan', label =
'cluster4')

mtp.scatter(x[y_predict == 4,0], x[y_predict == 4,1], s=100, c= 'yellow', label =
'cluster5')

mtp.scatter(kmeans.cluster_centers_[:, 0], kmeans.cluster_centers_[:, 1], s=100,
c='black', label='Centroids')

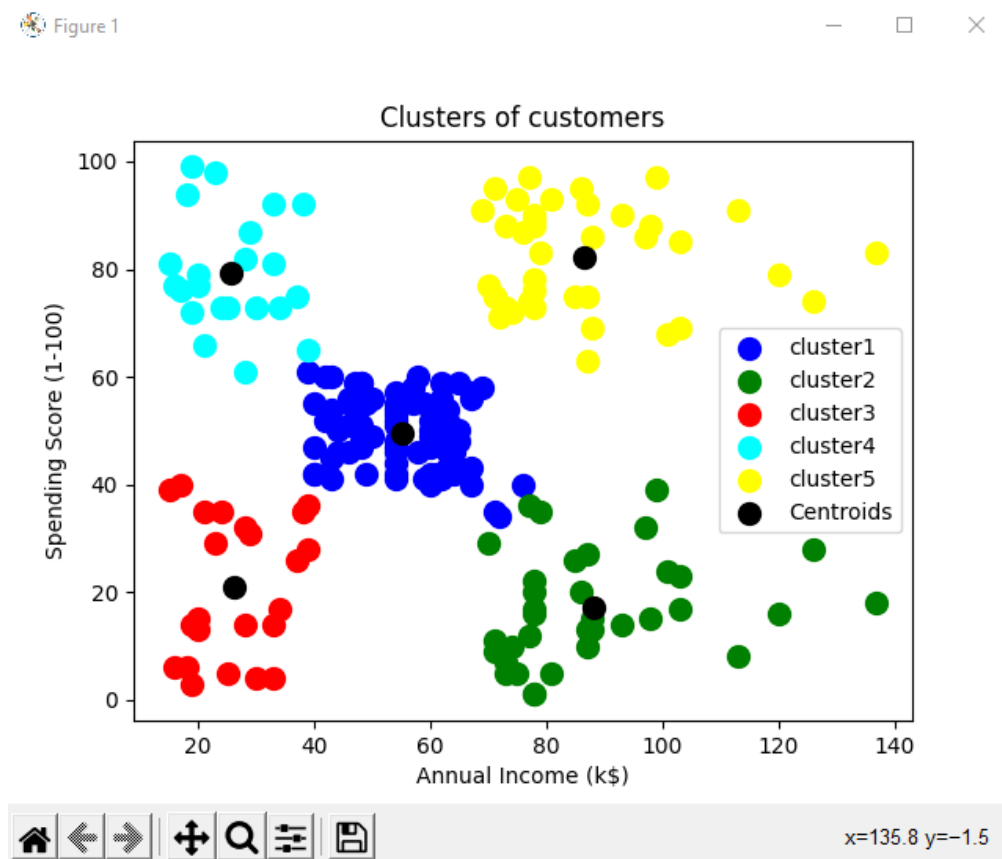
mtp.title('Clusters of customers')

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

mtp.ylabel('Spending Score (1-100)')

mtp.legend()

mtp.show()
```

Date:05/01/2022**Program no:12**

Aim: Program to implement K-Means clustering technique using any standard dataset available in the public domain.

Program

```
import numpy as nm

import matplotlib.pyplot as mtp

import pandas as pd

import sklearn

dataset =
pd.read_csv('world_country_and_usa_states_latitude_and_longitude_values.csv')

x = dataset.iloc[:, [1,2]].values

print(x)

from sklearn.cluster import KMeans

wcss = [] # empty array

for i in range(1, 11):

    kmeans = KMeans(n_clusters=i, init='k-means++', random_state=42)

    kmeans.fit(x)

    wcss.append(kmeans.inertia_)

mtp.plot(range(1, 11), wcss)

mtp.xlabel('number of clusters')

mtp.ylabel('wcss')

mtp.show()
```

```

kmeans = KMeans(n_clusters=5, init="k-means++", random_state=42)

y_kmeans = kmeans.fit_predict(x)

mtp.scatter(x[y_kmeans==0,0], x[y_kmeans==0,1], s=80, c='red', label = 'Cluster 1')
mtp.scatter(x[y_kmeans==1,0], x[y_kmeans==1,1], s=80, c='blue', label = 'Cluster 2')
mtp.scatter(x[y_kmeans==2,0], x[y_kmeans==2,1], s=80, c='green', label = 'Cluster 3')

mtp.scatter(kmeans.cluster_centers_[0], kmeans.cluster_centers_[1], s = 100, c =
'black', label = 'Centroids')

mtp.legend()

mtp.show()

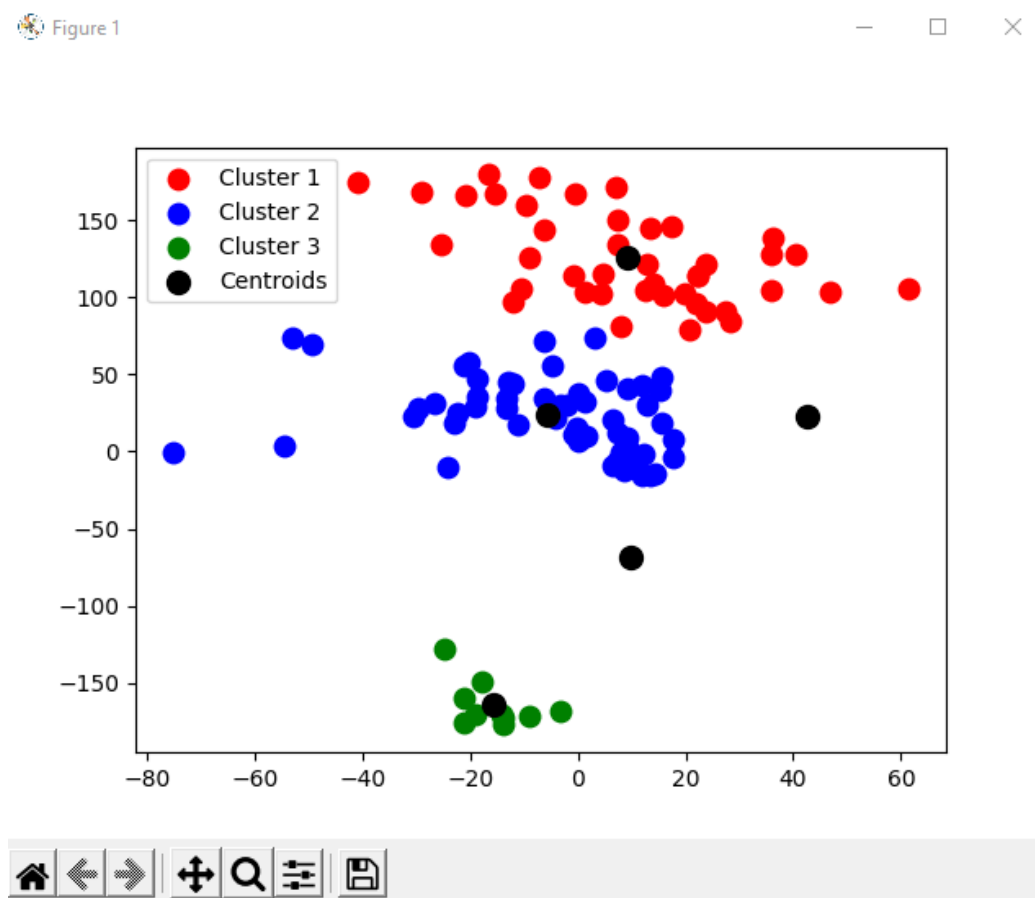
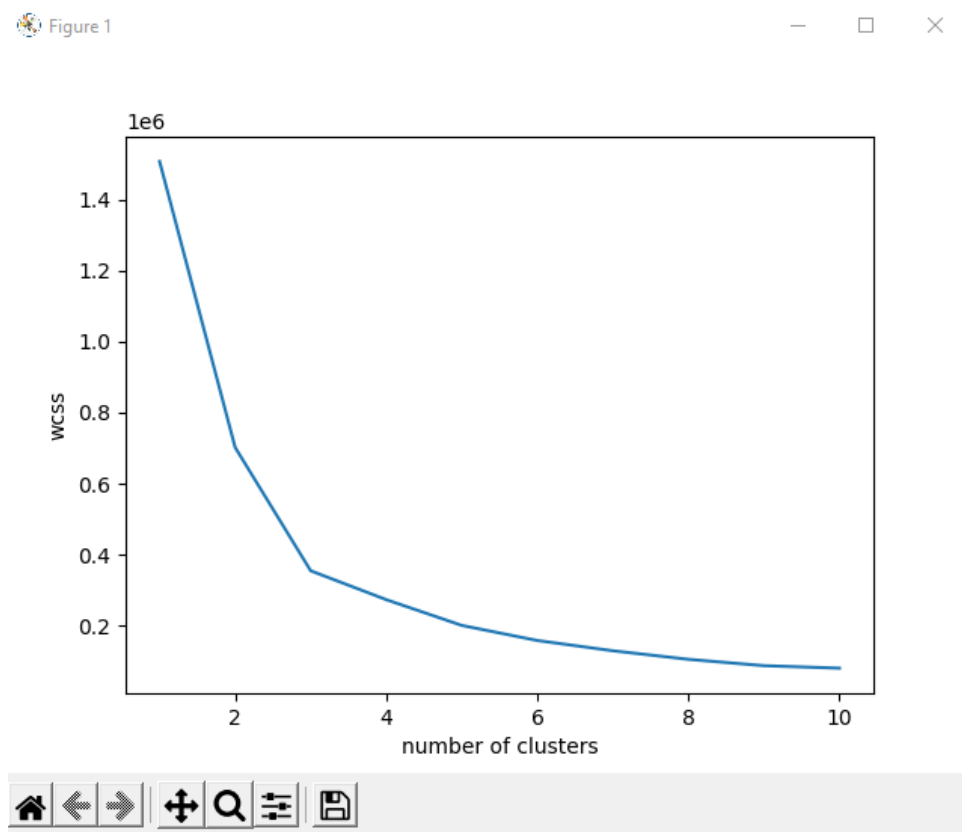
```

Output

```

C:\Users\mca\PycharmProjects\svd\venv\Scripts\python.exe
[[ 4.25462450e+01  1.60155400e+00]
 [ 2.34240760e+01  5.38478180e+01]
 [ 3.39391100e+01  6.77099530e+01]
 [ 1.70608160e+01 -6.17964280e+01]
 [ 1.82205540e+01 -6.30686150e+01]
 [ 4.11533320e+01  2.01683310e+01]
 [ 4.00690990e+01  4.50381890e+01]
 [ 1.22260790e+01 -6.90600870e+01]
 [-1.12026920e+01  1.78738870e+01]
 [-7.52509730e+01 -7.13890000e-02]
 [-3.84160970e+01 -6.36166720e+01]
 [-1.42709720e+01 -1.70132217e+02]
 [ 4.75162310e+01  1.45500720e+01]
 [-2.52743980e+01  1.33775136e+02]
 [ 1.25211100e+01 -6.99683380e+01]
 [ 4.01431050e+01  4.75769270e+01]
 [ 4.39158860e+01  1.76790760e+01]
 [ 1.31938870e+01 -5.95431980e+01]
 [ 2.36849940e+01  9.03563310e+01]

```



Date:02/02/2022

Program no:13

Aim: Programs on convolutional neural network to classify images from any standard dataset in the public domain

Program

```
import numpy as np

import pandas as pd

import matplotlib.pyplot as plt

import tensorflow as tf

from tensorflow import keras

np.random.seed(42)

fashion_mnist = keras.datasets.fashion_mnist

(x_train, y_train), (x_test, y_test) = fashion_mnist.load_data()

print(x_train.shape, x_test.shape)

x_train = x_train/255.0

x_test = x_test/255.0

plt.imshow(x_train[1], cmap='binary')

plt.show()

np.unique(y_test)

class_names = ['T-shirt/Top', 'Trouser', 'Pullover', 'Dress', 'Coat', 'Sandal', 'Shirt',
'Sneaker', 'Bag', 'Ankle Boot']

n_rows = 5

n_cols = 10
```

```

plt.figure(figsize=(n_cols * 1.4, n_rows * 1.6))

for row in range(n_rows):

    for col in range(n_cols):

        index = n_cols * row + col

        plt.subplot(n_rows, n_cols, index+1)

        plt.imshow(x_train[index], cmap='binary', interpolation='nearest')

        plt.axis('off')

        plt.title(class_names[y_train[index]])

plt.show()

model_CNN = keras.models.Sequential()

model_CNN.add(keras.layers.Conv2D(filters=32, kernel_size=7, padding='same',
activation='relu', input_shape=[28, 28, 1]))

model_CNN.add(keras.layers.MaxPooling2D(pool_size=2))

model_CNN.add(keras.layers.Conv2D(filters=64, kernel_size=3, padding='same',
activation='relu'))

model_CNN.add(keras.layers.MaxPooling2D(pool_size=2))

model_CNN.add(keras.layers.Conv2D(filters=32, kernel_size=3, padding='same',
activation='relu'))

model_CNN.add(keras.layers.MaxPooling2D(pool_size=2))

model_CNN.summary()

model_CNN.add(keras.layers.Flatten())

model_CNN.add(keras.layers.Dense(units=128, activation='relu'))

model_CNN.add(keras.layers.Dense(units=64, activation='relu'))

```

```

model_CNN.add(keras.layers.Dense(units=10, activation='softmax'))

model_CNN.summary()

model_CNN.compile(loss='sparse_categorical_crossentropy', optimizer='adam',
metrics=['accuracy'])

x_train = x_train[..., np.newaxis]
x_test = x_test[..., np.newaxis]

history_CNN = model_CNN.fit(x_train, y_train, epochs=2, validation_split=0.1)

pd.DataFrame(history_CNN.history).plot()

plt.grid(True)

plt.xlabel('epochs')

plt.ylabel('loss/accuracy')

plt.title('Training and validation plot')

plt.show()

test_loss, test_accuracy = model_CNN.evaluate(x_test, y_test)

print('Test Loss: {}', 'Test Accuracy: {}'.format(test_loss, test_accuracy))

```

Output

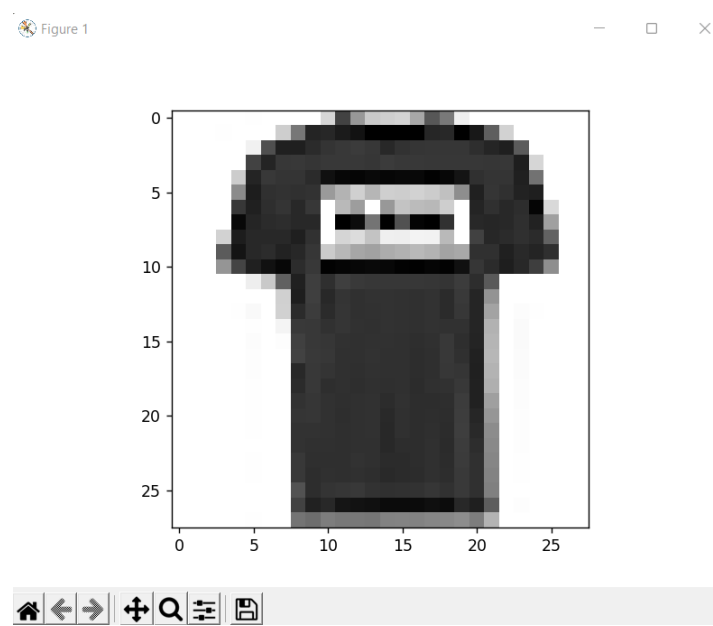
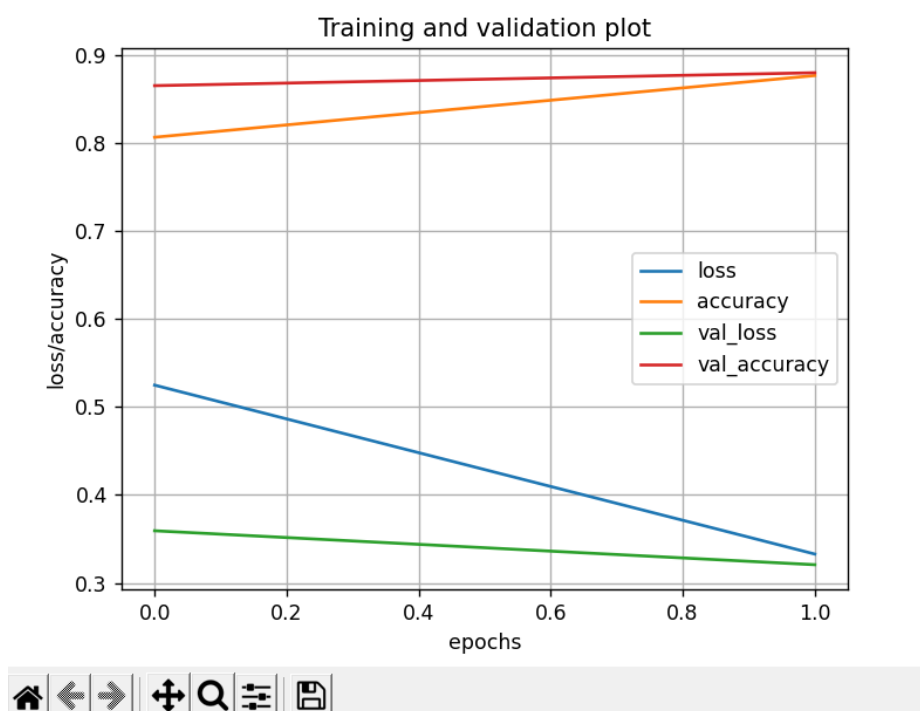


Figure 1



Figure 1



```

Model: "sequential"
-----
Layer (type)                Output Shape                Param #
-----
conv2d (Conv2D)              (None, 28, 28, 32)         1600
max_pooling2d (MaxPooling2D) (None, 14, 14, 32)         0
conv2d_1 (Conv2D)            (None, 14, 14, 64)         18496
max_pooling2d_1 (MaxPooling2D) (None, 7, 7, 64)         0
conv2d_2 (Conv2D)            (None, 7, 7, 32)           18464
max_pooling2d_2 (MaxPooling2D) (None, 3, 3, 32)         0

Total params: 38,560
Trainable params: 38,560
Non-trainable params: 0
-----
Model: "sequential"
-----
Layer (type)                Output Shape                Param #
-----
conv2d (Conv2D)              (None, 28, 28, 32)         1600
max_pooling2d (MaxPooling2D) (None, 14, 14, 32)         0

```

```

=====
Total params: 84,458
Trainable params: 84,458
Non-trainable params: 0
-----
Epoch 1/2
1688/1688 [=====] - 57s 33ms/step - loss: 0.5248 - accuracy: 0.8064 - val_loss: 0.3593 - val_accuracy: 0.8650
Epoch 2/2
1688/1688 [=====] - 52s 31ms/step - loss: 0.3329 - accuracy: 0.8765 - val_loss: 0.3207 - val_accuracy: 0.8797
313/313 [=====] - 4s 12ms/step - loss: 0.3302 - accuracy: 0.8794
Test Loss: {} Test Accuracy: 0.3301725685596466

Process finished with exit code 0

```

Date:16/02/2022

Program no:14

Aim: Program to implement a simple web crawler using python.

Program

```
import requests

import lxml

from bs4 import BeautifulSoup

url = "https://www.rottentomatoes.com/top/bestofrt/"

headers = {

    'User-Agent':'Mozilla/5.0 (Windows NT 6.1; WOW64) AppleWebKit/537.36
(KHTML, like Gecko) Chrome/63.0.3239.132 Safari/537.36 OPR/50.0.2762.58
(Edition Yx 01)'

}

f = requests.get(url, headers = headers)

movies_lst = []

soup = BeautifulSoup(f.content, 'html.parser')

movies = soup.find('table', {

    'class':'table'

}).find_all('a')

print(movies)

num = 0

for anchor in movies:

    urls = 'https://www.rottentomatoes.com' + anchor['href']

    movies_lst.append(urls)

print(movies_lst)
```

```

num += 1

movie_url = urls

movie_f = requests.get(movie_url, headers=headers)

movie_soup = BeautifulSoup(movie_f.content, 'lxml')

movie_content = movie_soup.find('div', {

    'class':'movie_synopsis clamp clamp-6 js-clamp'

})

print(num, urls, '\n', 'Movie:' + anchor.string.strip())

print('Movie info:' + movie_content.string.strip())

```

Output

```

"C:\Users\ajcemca\Desktop\my pgms\venv\Scripts\python.exe" "C:/Users/ajcemca/Desktop/my pgms/venv/simple web crawler.py"
[<a class="unstyled articleLink" href="/m/it_happened_one_night">
    It Happened One Night (1934)</a>, <a class="unstyled articleLink" href="/m/citizen_kane">
    Citizen Kane (1941)</a>, <a class="unstyled articleLink" href="/m/the_wizard_of_oz_1939">
    The Wizard of Oz (1939)</a>, <a class="unstyled articleLink" href="/m/modern_times">
    Modern Times (1936)</a>, <a class="unstyled articleLink" href="/m/black_panther_2018">
    Black Panther (2018)</a>, <a class="unstyled articleLink" href="/m/parasite_2019">
    Parasite (Gisaengchung) (2019)</a>, <a class="unstyled articleLink" href="/m/avengers_endgame">
    Avengers: Endgame (2019)</a>, <a class="unstyled articleLink" href="/m/1003707-casablanca">
    Casablanca (1942)</a>, <a class="unstyled articleLink" href="/m/knives_out">
    Knives Out (2019)</a>, <a class="unstyled articleLink" href="/m/us_2019">
    The (2019)</a>, <a class="unstyled articleLink" href="/m/beatles_a_hard_days_night">
    The Beatles: A Hard Day's Night (1964)</a>, <a class="unstyled articleLink" href="/m/widows_2018">
    Widows (2018)</a>, <a class="unstyled articleLink" href="/m/never_rarely_sometimes_always">
    Never Rarely Sometimes Always (2020)</a>, <a class="unstyled articleLink" href="/m/baby_driver">
    Baby Driver (2017)</a>, <a class="unstyled articleLink" href="/m/spider_man_homecoming">
    Spider-Man: Homecoming (2017)</a>, <a class="unstyled articleLink" href="/m/godfather_part_ii">
    The Godfather, Part II (1974)</a>, <a class="unstyled articleLink" href="/m/the_battle_of_algiers">
    The Battle of Algiers (La Battaglia di Algeri) (1967)</a>]
['https://www.rottentomatoes.com/m/it_happened_one_night', 'https://www.rottentomatoes.com/m/citizen_kane', 'https://www.rottentomatoes.com/m/the_wizard_of_oz',
1 https://www.rottentomatoes.com/m/the_battle_of_algiers /n Movie:The Battle of Algiers (La Battaglia di Algeri) (1967)
Movie info:Paratrooper commander Colonel Mathieu (Jean Martin), a former French Resistance fighter during World War II, is sent to 1950s Algeria to reinforce

```

Date:16/02/2022

Program no:15**Aim:** Program to implement a simple web crawler using python.**Program**

```

import requests
import lxml
from bs4 import BeautifulSoup
url = "https://www.rottentomatoes.com/top/bestofrt/"
headers = {
    'User-Agent':'Mozilla/5.0 (Windows NT 6.1; WOW64) AppleWebKit/537.36
(KHTML, like Gecko) Chrome/63.0.3239.132 Safari/537.36 OPR/50.0.2762.58
(Edition Yx 01)'
}
f = requests.get(url, headers = headers)
movies_lst = []
soup = BeautifulSoup(f.content, 'html.parser')
movies = soup.find('table', {
    'class':'table'
}).find_all('a')
print(movies)
num = 0
for anchor in movies:
    urls = 'https://www.rottentomatoes.com' + anchor['href']
    movies_lst.append(urls)
print(movies_lst)
num += 1
movie_url = urls
movie_f = requests.get(movie_url, headers=headers)
movie_soup = BeautifulSoup(movie_f.content, 'lxml')
movie_content = movie_soup.find('div', {
    'class':'movie_synopsis clamp clamp-6 js-clamp'
})

```

```
print(num, urls, '\n', 'Movie:' + anchor.string.strip())
print('Movie info:' + movie_content.string.strip())
```

Output

```
War for the Planet of the Apes (2017)/a>, <a class="unstyled articleLink" href="/m/paddington2">
Paddington 2 (2018)</a>, <a class="unstyled articleLink" href="/m/beatles_a_hard_days_night">
A Hard Day's Night (1964)</a>, <a class="unstyled articleLink" href="/m/widows_2018">
Widows (2018)</a>, <a class="unstyled articleLink" href="/m/never_rarely_sometimes_always">
Never Rarely Sometimes Always (2020)</a>, <a class="unstyled articleLink" href="/m/baby_driver">
Baby Driver (2017)</a>, <a class="unstyled articleLink" href="/m/spider_man_homecoming">
Spider-Man: Homecoming (2017)</a>, <a class="unstyled articleLink" href="/m/godfather_part_ii">
The Godfather, Part II (1974)</a>, <a class="unstyled articleLink" href="/m/the_battle_of_algiers">
The Battle of Algiers (La Battaglia di Algeri) (1967)</a>]
['https://www.rottentomatoes.com/m/it_happened_one_night', 'https://www.rottentomatoes.com/m/citizen_kane', 'https://www.rottentomatoes.com/m/the_wizard_of_oz
1 https://www.rottentomatoes.com/m/the_battle_of_algiers /n Movie:The Battle of Algiers (La Battaglia di Algeri) (1967)
Movie info:Paratrooper commander Colonel Mathieu (Jean Martin), a former French Resistance fighter during World War II, is sent to 1950s Algeria to reinforce
```

Date:16/02/2022**Program no:16****Aim:** Program to implement scrap of any website.**Program**

```
import csv

import requests

from bs4 import BeautifulSoup

url="http://www.values.com/inspirational-quotes"

r=requests.get(url)

print("Content:")

print(r.content)

print("Prettify:")

soup=BeautifulSoup(r.content,'lxml')

print(soup.prettify())

quotes=[]

table=soup.find('div',attrs={'id':'all_quotes'})

for row in table.find_all('div',attrs={'class':'col-6 col-lg-3 text-center margin-30px-bottom sm-margin-30px-top'}):

    quote={ }

    quote['theme']=row.h5.text

    quote['url']=row.a['href']

    quote['img']=row.img['src']

    quote['lines']=row.img['alt'].split("#")[0]
```

```

quote['author']=row.img['alt'].split("#")[1]

quotes.append(quote)

filename='inspiration_quotation.csv'

with open(filename,'w',newline='') as f:

    w=csv.DictWriter(f,['theme','url','img','lines','author'])

    w.writeheader()

    for quote in quotes:

        w.writerow(quote)

```

Output

```

"C:\Users\ajcemca\Desktop\my pgms\venv\Scripts\python.exe" "C:/Users/ajcemca/Desktop/my pgms/transpose.py"
Content:
b'<!DOCTYPE html>\n<html class="no-js" dir="ltr" lang="en-US">\n    <head>\n        <title>Inspirational Quotes - Motivational Quotes - Leadership Quotes | Pas
Prettify:
<!DOCTYPE html>
<html class="no-js" dir="ltr" lang="en-US">
  <head>
    <title>
      Inspirational Quotes - Motivational Quotes - Leadership Quotes | PassItOn.com
    </title>
    <meta charset="utf-8"/>
    <meta content="text/html; charset=utf-8" http-equiv="content-type"/>
    <meta content="IE=edge" http-equiv="X-UA-Compatible"/>
    <meta content="width=device-width,initial-scale=1.0" name="viewport"/>

```

```

theme,url,img,lines,author
LOVE,/inspirational-quotes/7444-where-there-is-love-there-is-life,https://assets.passiton.com/quotes/quote_artwork/74
LOVE,/inspirational-quotes/7439-at-the-touch-of-love-everyone-becomes-a-poet,https://assets.passiton.com/quotes/quote
FRIENDSHIP,/inspirational-quotes/8304-a-friend-may-be-waiting-behind-a-stranger-s-face,https://assets.passiton.com/qu
FRIENDSHIP,/inspirational-quotes/3331-whenever-we-are-it-is-our-friends-that-make,https://assets.passiton.com/quotes/
FRIENDSHIP,/inspirational-quotes/8303-find-a-group-of-people-who-challenge-and,https://assets.passiton.com/quotes/qu
FRIENDSHIP,/inspirational-quotes/8302-there-s-not-a-word-yet-for-old-friends-who-ve,https://assets.passiton.com/quote
FRIENDSHIP,/inspirational-quotes/7435-there-are-good-ships-and-wood-ships-ships-that,https://assets.passiton.com/quot
PERSISTENCE,/inspirational-quotes/6377-at-211-degrees-water-is-hot-at-212-degrees,https://assets.passiton.com/quotes/
PERSISTENCE,/inspirational-quotes/8301-the-key-of-persistence-opens-all-doors-closed,https://assets.passiton.com/qu

```


Date:16/02/2022

Program no:17

Aim: Program for Natural Language Processing which performs n-grams.

Program

```
def generate_ngrams(text,WordsToCombine):  
    words=text.split()  
    output=[]  
    for i in range(len(words)-WordsToCombine+1):  
        output.append(words[i:i + WordsToCombine])  
    return output  
  
x=generate_ngrams(text="this is a good book to study",WordsToCombine=3)  
  
print(x)
```

Output

```
"C:\Users\ajcemca\Desktop\my pgms\venv\Scripts\python.exe" "C:/Users/ajcemca/Desktop/my pgms/venv/ngrams.py"  
[['this', 'is', 'a'], ['is', 'a', 'good'], ['a', 'good', 'book'], ['good', 'book', 'to'], ['book', 'to', 'study']]  
  
Process finished with exit code 0
```

Date:16/02/2022

Program no:18

Aim: Program for Natural Language Processing which performs n-grams (Using in built functions).

Program

```
import nltk

from nltk.util import ngrams

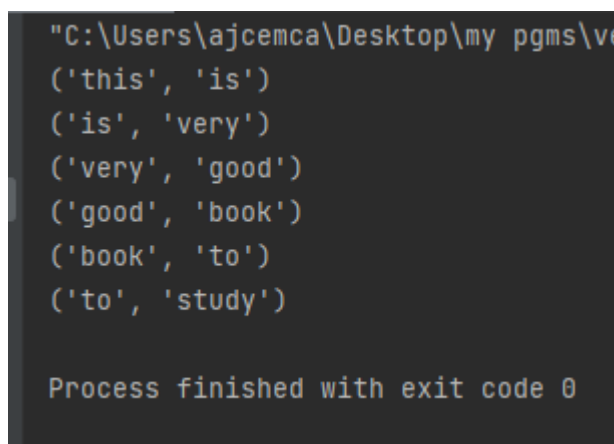
samplText="this is very good book to study"

Ngrams=ngrams(sequence=nltk.wordpunct_tokenize(samplText),n=2)

for grams in Ngrams:

    print(grams)
```

Output:



```
"C:\Users\ajcemca\Desktop\my pgms\ve
('this', 'is')
('is', 'very')
('very', 'good')
('good', 'book')
('book', 'to')
('to', 'study')

Process finished with exit code 0
```

Date:16/02/2022

Program no:19

Aim: Program for Natural Language Processing which performs speech tagging.

Program

```
import nltk

from nltk.corpus import stopwords

nltk.download('stopwords')

nltk.download('punkt')

nltk.download('averaged_perceptron_tagger')

from nltk.tokenize import word_tokenize, sent_tokenize

stop_words = set(stopwords.words('english'))

txt = "Sukanya, Rajib and Naba are my good friends. " \
      "Sukanya is getting married next year. " \
      "Marriage is a big step in one's life." \
      "It is both exciting and frightening. " \
      "But friendship is a sacred bond between people." \
      "It is a special kind of love between us. " \
      "Many of you must have tried searching for a friend " \
      "but never found the right one."

tokenized = sent_tokenize(txt)

for i in tokenized:

    wordsList = nltk.word_tokenize(i)

    wordsList = [w for w in wordsList if not w in stop_words]+

    tagged = nltk.pos_tag(wordsList)
```

```
print(tagged)
```

Output:

```
[('Sukanya', 'NNP'), (',', ','), ('Rajib', 'NNP'), ('Naba', 'NNP'), ('good', 'JJ'), ('friends', 'NNS'), ('.', '.')]
[('Sukanya', 'NNP'), ('getting', 'VBG'), ('married', 'VBN'), ('next', 'JJ'), ('year', 'NN'), ('.', '.')]
[('Marriage', 'NN'), ('big', 'JJ'), ('step', 'NN'), ('one', 'CD'), ('', ''), ('life.It', 'NN'), ('exciting', 'VBG'), ('frightening', 'NN'), ('.', '.')]
[('But', 'CC'), ('friendship', 'NN'), ('sacred', 'VBD'), ('bond', 'NN'), ('people.It', 'NN'), ('special', 'JJ'), ('kind', 'NN'), ('love', 'VB'), ('us', 'PRP'), ('.', '.')]
[('Many', 'JJ'), ('must', 'MD'), ('tried', 'VB'), ('searching', 'VBG'), ('friend', 'NN'), ('never', 'RB'), ('found', 'VBD'), ('right', 'JJ'), ('one', 'CD'), ('.', '.')]

Process finished with exit code 0
```

Date:16/02/2022

Program no:20**Aim: Program for Natural Language Processing which performs Chunking.****Program**

```
import nltk

nltk.download('punkt')

new = "The big cat ate the little mouse who was after the fresh cheese"

new_tokens = nltk.word_tokenize(new)

print(new_tokens)

new_tag = nltk.pos_tag(new_tokens)

[print(new_tag)]

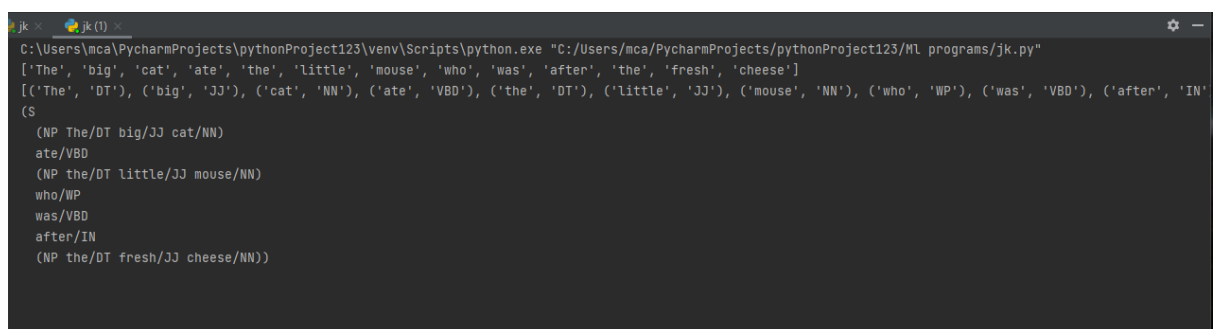
grammar=r"NP: {<DT>?<JJ>*<NN>}"

chunkParser = nltk.RegexpParser(grammar)

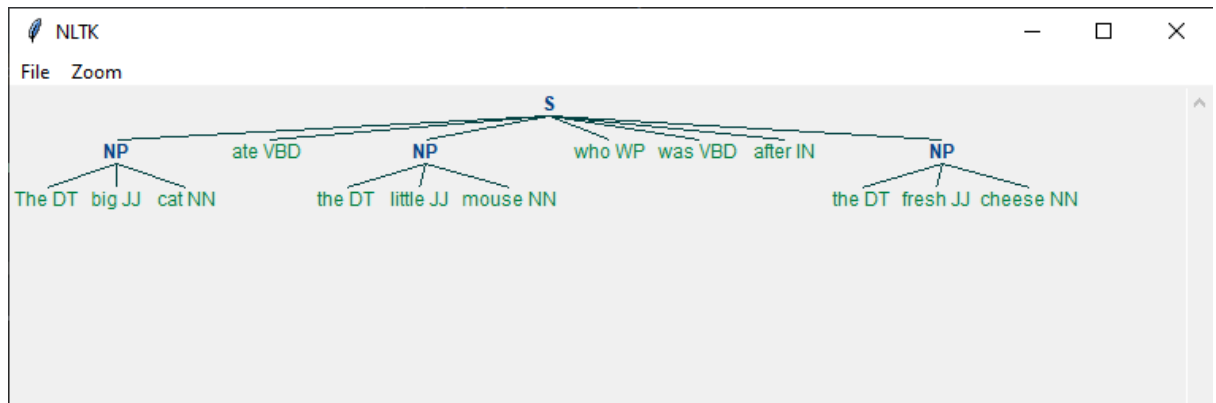
chunked=chunkParser.parse(new_tag)

print(chunked)

chunked.draw()
```

Output

```
C:\Users\mca\PycharmProjects\pythonProject123\venv\Scripts\python.exe "C:/Users/mca/PycharmProjects/pythonProject123/HL programs/jk.py"
['The', 'big', 'cat', 'ate', 'the', 'little', 'mouse', 'who', 'was', 'after', 'the', 'fresh', 'cheese']
[('The', 'DT'), ('big', 'JJ'), ('cat', 'NN'), ('ate', 'VBD'), ('the', 'DT'), ('little', 'JJ'), ('mouse', 'NN'), ('who', 'WP'), ('was', 'VBD'), ('after', 'IN'), ('the', 'DT'), ('fresh', 'JJ'), ('cheese', 'NN')]
(S
  (NP The/DT big/JJ cat/NN)
  ate/VBD
  (NP the/DT little/JJ mouse/NN)
  who/WP
  was/VBD
  after/IN
  (NP the/DT fresh/JJ cheese/NN))
```



Date: 23-02-2022

PROGRAM NO : 21

Aim: Write a python program for natural program language processing with chunking.

Program:

```
import nltk
#nltk.download('averaged_perception_tagger')
sample_text="""
Rama killed Ravana to save Sita from Lanka.The legend of the Ramayan is the most
popular Indian epic.
A lot of movies and serials have already been shot in several languages here in India
based Ramayana"""

tokenized=nltk.sent_tokenize(sample_text)
for i in tokenized:
    words=nltk.word_tokenize(i)
    tagged_words=nltk.pos_tag(words)
    chunkGram=r"""VB: { }"""
    chunkParser=nltk.RegexpParser(chunkGram)
    chunked=chunkParser.parse(tagged_words)
    print(chunked)
    chunked.draw()
```

Output

```
C:\Programming\Python39\python.exe "C:/Users/asifk/PycharmProjects/ML/23-02-2022/chunking 2.py"
(S
  Rama/NNP
  killed/VBD
  Ravana/NNP
  to/TO
  save/VB
  Sita/NNP
  from/IN
  Lanka.The/NNP
  legend/NN
  of/IN
  the/DT
  Ramayan/NNP
  is/VBZ
  the/DT
  most/RBS
  popular/JJ
  Indian/JJ
  epic/NN
  ./.)
```

