Aim:

The probability that it is Friday and that a student is absent is 3%. Since there are 5 school days in a week, the probability that it is Friday is 20%. What is theprobability that a student is absent given that today is Friday? Apply Baye's rule in python to get the result. (Ans: 15%)

Source Code:

Week1.py

```
''' Aim : The probability that it is Friday and that a student is absent is 3 %. Since there are 5 school days in a week, the probability that it is Friday is 20 %. What is the probability that a student is absent given that today is Friday? Apply Baye's rule in python to get the result.


================================
Explanation:
================================
        F : Friday
        A : Absent

        Based on the given problem statement,

        The probability that it is Friday and that a student is absent is 3%
        i.e
        P(A ∩ F)= 3% = 3 / 100 = 0.03

        and

        The probability that it is Friday is 20%
        i.e

        P(F)=20% = 20/100 = 0.2

        Then,

        The probability that a student is absent given that today is Friday
        P(A | F)

        By the definition of Baye's rule( conditional probability ), we have

        P(A | F) = P(A ∩ F) / P(F)


==============================
Source Code :
==============================

'''

# The probability that it is Friday and that a student is absent is 3%
pAF=0.03
print("The probability that it is Friday and that a student is absent :",pAF)
# The probability that it is Friday is 20%
```

```
pF=0.2
print("The probability that it is Friday : ",pF)
# The probability that a student is absent given that today is Friday
pResult=(pAF/pF)
# Display the Result
print("The probability that a student is absent given that today is Friday :
",pResult * 100,"%")
```

Output:

```
D:\Machine Learning\Lab>python Week1.py
The probability that it is Friday and that a student is absent : 0.03
The probability that it is Friday :  0.2
The probability that a student is absent given that today is Friday :  15.0 %
```

Aim:

Extract the data from database using python

Source Code:

Week2.py

```
'''Aim:  Extract the data from database using python

================================
Explanation:
================================

===> First You need to Create a Table (students) in Mysql Database (SampleDB)

---> Open Command prompt and then execute the following command to enter into
MySQL prompt.

--> mysql -u root -p


And then, you need to execute the following commands at MySQL prompt to create
table in the database.

--> create database SampleDB;

--> use SampleDB;

--> CREATE TABLE students (sid VARCHAR(10),sname VARCHAR(10),age int);

--> INSERT INTO students VALUES('s521','Jhon Bob',23);
--> INSERT INTO students VALUES('s522','Dilly',22);
--> INSERT INTO students VALUES('s523','Kenney',25);
--> INSERT INTO students VALUES('s524','Herny',26);


===> Next,Open Command propmt and then execute the following command to install
mysql.connector package to connect with mysql database through python.

--> pip install mysql.connector (Windows)
--> sudo apt-get install mysql.connector (linux)

================================
Source Code :
================================. '''

import mysql.connector
# Create the connection object
myconn = mysql.connector.connect(host = "localhost", user = "root",passwd =
"",database="SampleDB")
# Creating the cursor object
cur = myconn.cursor()
# Executing the query
cur.execute("select * from students")
# Fetching the rows from the cursor object
result = cur.fetchall()
print("Student Details are :")
# Printing the result
for x in result:
```

```
        print(x);
# Commit the transaction
myconn.commit()
# Close the connection
myconn.close()
```

Output:

```
C:\xampp\mysql\bin>mysql -u root -p
Enter password:
Welcome to the MariaDB monitor.  Commands end with ; or \g.
Your MariaDB connection id is 9
Server version: 10.4.18-MariaDB mariadb.org binary distribution

Copyright (c) 2000, 2018, Oracle, MariaDB Corporation Ab and others.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

MariaDB [(none)]> create database SampleDB;
Query OK, 1 row affected (0.046 sec)

MariaDB [(none)]> use SampleDB;
Database changed
MariaDB [SampleDB]> CREATE TABLE students (sid VARCHAR(10),sname VARCHAR(10),age int);
Query OK, 0 rows affected (0.285 sec)

MariaDB [SampleDB]> INSERT INTO students VALUES('s521','Jhon Bob',23);
Query OK, 1 row affected (0.066 sec)

MariaDB [SampleDB]> INSERT INTO students VALUES('s522','Dilly',22);
Query OK, 1 row affected (0.061 sec)

MariaDB [SampleDB]> INSERT INTO students VALUES('s523','Kenney',25);
Query OK, 1 row affected (0.029 sec)

MariaDB [SampleDB]> INSERT INTO students VALUES('s524','Herny',26);
Query OK, 1 row affected (0.040 sec)

MariaDB [SampleDB]>
```

```
D:\Machine Learning\Lab>python Week2.py
Student Details are :
('s521', 'Jhon Bob', 23)
('s522', 'Dilly', 22)
('s523', 'Kenney', 25)
('s524', 'Herny', 26)
```

Aim:

Implement k-nearest neighbours classification using python

Source Code:

Week3.py

```
''' Aim : Implement k-nearest neighbours classification using python.


================================
Explanation:
================================
===> To run this program you need to install the sklearn Module

===> Open Command propmt and then execute the following command to install
sklearn Module

---> pip install scikit-learn

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

In this program, we are going to use iris dataset.And this dataset Split into
training(70%) and test set(30%).

The iris dataset conatins the following features

---> sepal length (cm)
---> sepal width (cm)
---> petal length (cm)
---> petal width (cm)

The Sample data in iris dataset format is  [5.4 3.4 1.7 0.2]

Where   5.4 ---> sepal length (cm)
                 3.4 ---> sepal width (cm)
                 1.7 ---> petal length (cm)
                 0.2 ---> petal width (cm)


==============================
Source Code :
==============================

'''

# Import necessary modules
from sklearn.neighbors import KNeighborsClassifier
from sklearn.model_selection import train_test_split
from sklearn.datasets import load_iris
import random

# Loading data
data_iris = load_iris()
# To get list of target names
label_target = data_iris.target_names
print()
print("Sample Data from Iris Dataset")
print("*"*30)
```

```
# to display the sample data from the iris dataset
for i in range(10):
        rn = random.randint(0,120)
        print(data_iris.data[rn],"===>",label_target[data_iris.target[rn]])

# Create feature and target arrays
X = data_iris.data
y = data_iris.target

# Split into training and test set
X_train, X_test, y_train, y_test = train_test_split(
                        X, y, test_size = 0.3, random_state=1)

print("The Training dataset length: ",len(X_train))
print("The Testing dataset length: ",len(X_test))
try:
        nn = int(input("Enter number of neighbors :"))
        knn = KNeighborsClassifier(nn)

        knn.fit(X_train, y_train)
        # to display the score
        print("The Score is :",knn.score(X_test, y_test))
        # To get test data from the user
        test_data = input("Enter Test Data :").split(",")
        for i in range(len(test_data)):
                test_data[i] = float(test_data[i])
        print()

        v = knn.predict([test_data])
        print("Predicted output is :",label_target[v])
except:
        print("Please supply valid input......")
```

Output:

```
D:\Machine Learning\Lab>python Week3.py

Sample Data from Iris Dataset
*****************************
[6.8 2.8 4.8 1.4] ===> versicolor
[5.8 2.7 3.9 1.2] ===> versicolor
[6.6 2.9 4.6 1.3] ===> versicolor
[5.1 3.8 1.6 0.2] ===> setosa
[4.9 2.5 4.5 1.7] ===> virginica
[7.3 2.9 6.3 1.8] ===> virginica
[4.9 3.1 1.5 0.1] ===> setosa
[4.5 2.3 1.3 0.3] ===> setosa
[7.6 3.  6.6 2.1] ===> virginica
[6.6 3.  4.4 1.4] ===> versicolor
The Training dataset length:  105
The Testing dataset length:  45
Enter number of neighbors :10
The Score is : 0.9777777777777777
Enter Test Data :6.2,2.6,3.4,0.6

Predicted output is : ['versicolor']
```

```
D:\Machine Learning\Lab>python Week3.py

Sample Data from Iris Dataset
*****************************
[6.1 2.8 4.7 1.2] ===> versicolor
[4.9 3.1 1.5 0.2] ===> setosa
[5.4 3.9 1.3 0.4] ===> setosa
[4.9 2.4 3.3 1. ] ===> versicolor
[6.4 3.2 5.3 2.3] ===> virginica
[6.3 3.3 6.  2.5] ===> virginica
[6.5 3.  5.8 2.2] ===> virginica
[5.6 2.7 4.2 1.3] ===> versicolor
[5.1 2.5 3.  1.1] ===> versicolor
[4.4 3.  1.3 0.2] ===> setosa
The Training dataset length:  105
The Testing dataset length:  45
Enter number of neighbors :10
The Score is : 0.9777777777777777
Enter Test Data :5.0,3.3,1.4,0.3

Predicted output is : ['setosa']
```

Aim:

Given the following data, which specify classifications for nine ombinationsof VAR1 and VAR2 predict a classification for a case where VAR1=0.906and VAR2=0.606, using the result of k-means clustering with 3 means (i.e., 3centroids)

Source Code:

Week4.py

```
'''Aim: Given the following data, which specify classifications for nine
ombinationsof VAR1 and VAR2 predict a classification for a case where
VAR1=0.906and VAR2=0.606, using the result of k-means clustering with 3 means
(i.e., 3centroids)

================================
Explanation:
================================
===> To run this program you need to install the sklearn Module

===> Open Command propmt and then execute the following command to install
sklearn Module

---> pip install scikit-learn

In this program, we are going to use the following data

VAR1 VAR2 CLASS
1.713 1.586 0
0.180 1.786 1
0.353 1.240 1
0.940 1.566 0
1.486 0.759 1
1.266 1.106 0
1.540 0.419 1
0.459 1.799 1
0.773 0.186 1

And, we need apply k-means clustering with 3 means (i.e., 3 centroids)


Finally, you need to predict the class for the VAR1=0.906 and VAR2=0.606

==============================
Source Code :
==============================
'''
from sklearn.cluster import KMeans
import numpy as np
X = np.array([[1.713,1.586], [0.180,1.786], [0.353,1.240],
[0.940,1.566], [1.486,0.759], [1.266,1.106],[1.540,0.419],[0.459,1.799],
[0.773,0.186]])
y=np.array([0,1,1,0,1,0,1,1,1])
kmeans = KMeans(n_clusters=3, random_state=0).fit(X,y)
print("The input data is ")
print("VAR1 \t VAR2 \t CLASS")
i=0
```

```
for val in X:
        print(val[0],"\t",val[1],"\t",y[i])
        i+=1
print("="*20)
# To get test data from the user
print("The Test data to predict ")
test_data = []
VAR1 = float(input("Enter Value for VAR1 :"))
VAR2 = float(input("Enter Value for VAR2 :"))
test_data.append(VAR1)
test_data.append(VAR2)
print("="*20)
print("The predicted Class is : ",kmeans.predict([test_data]))
```

Output:

```
D:\Machine Learning\Lab>python Week4.py
The input data is
VAR1     VAR2    CLASS
1.713    1.586   0
0.18     1.786   1
0.353    1.24    1
0.94     1.566   0
1.486    0.759   1
1.266    1.106   0
1.54     0.419   1
0.459    1.799   1
0.773    0.186   1
====================
The Test data to predict
Enter Value for VAR1 :0.906
Enter Value for VAR2 :0.606
====================
The predicted Class is :  [0]
```

Aim:

The following training examples map descriptions of individuals onto high, medium and low credit-worthiness.Input attributes are (from left to right) income, recreation, job, status, age-group, home-owner. Find the unconditional probability of 'golf' and the conditional probability of 'single' given 'medRisk' in the dataset

Source Code:

Week5.py

```
'''Aim:The following training examples map descriptions of individuals onto
high, medium and low credit-worthiness.

medium skiing design single twenties no -> highRisk
high golf trading married forties yes -> lowRisk
low speedway transport married thirties yes -> medRisk
medium football banking single thirties yes -> lowRisk
high flying media married fifties yes -> highRisk
low football security single twenties no -> medRisk
medium golf media single thirties yes -> medRisk
medium golf transport married forties yes -> lowRisk
high skiing banking single thirties yes -> highRisk
low golf unemployed married forties yes -> highRisk

Input attributes are (from left to right) income, recreation, job, status, age-
group, home-owner. Find the unconditional probability of 'golf' and the
conditional probability of 'single' given 'medRisk' in the dataset

================================
Explanation:
================================

In the given data set,

----> The total number of records are 10.

----> The number of records which contains 'golf' are 4.

----> Then, the Unconditional probability of golf :

                        =  The number of records which contains 'golf' / total
number of records
                        = 4 / 10
                        = 0.4

*****************************

To find the Conditional probability of single given medRisk,

---> S : single
---> MR : medRisk

---> By the definition of Baye's rule( conditional probability ), we have

        P(S | MR) = P(S ∩ MR) / P(MR)
```

```
        Based on the given problem statement,

    P(S ∩ MR) = The number of MedRisk with Single records / total number of
Records
                          = 2 / 10 = 0.2
    and

    P(MR) = The number of records with MedRisk /total number of Records
                  = 3 / 10 = 0.3

        Then, the Conditional probability of single given medRisk

        P(S | MR) = 0.2 / 0.3
                          = 0.66666

================================
Source Code :
================================
'''
total_Records=10
numGolfRecords=4
unConditionalprobGolf=numGolfRecords / total_Records
print("Unconditional probability of golf: ={}".format(unConditionalprobGolf))
#conditional probability of 'single' given 'medRisk'
numMedRiskSingle=2
numMedRisk=3
probMedRiskSingle=numMedRiskSingle/total_Records
probMedRisk=numMedRisk/total_Records
conditionalProb=(probMedRiskSingle/probMedRisk)
print("Conditional probability of single given medRisk: =
{}".format(conditionalProb))
```

Output:

```
D:\Machine Learning\Lab>python Week5.py
Unconditional probability of golf: =0.4
Conditional probability of single given medRisk: = 0.6666666666666667
```

Aim:

Implement linear regression using python

Source Code:

Week6.py

```
''' Implement linear regression using python

================================
Explanation:
================================

===> To run this program you need to install the pandas Module

---> pandas Module is used to read csv files

===> To install, Open Command propmt and then execute the following command

---> pip install pandas


And, then you need to install the matplotlib Module

---> matplotlib Module is used to plot the graphs

===> To install, Open Command propmt and then execute the following command

---> pip install matplotlib

Finally, you need to create dataset called "Age_Income.csv" file.

==============================
Source Code :
==============================

'''
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
# To read data from Age_Income.csv file
dataFrame = pd.read_csv('Age_Income.csv')
# To place data in to age and income vectors
age = dataFrame['Age']
income = dataFrame['Income']

# number of points
num = np.size(age)
# To find the mean of age and income vector
mean_age = np.mean(age)
mean_income = np.mean(income)

# calculating cross-deviation and deviation about age
CD_ageincome = np.sum(income*age) - num*mean_income*mean_age
CD_ageage = np.sum(age*age) - num*mean_age*mean_age

# calculating regression coefficients
b1 = CD_ageincome / CD_ageage
b0 = mean_income - b1*mean_age
```

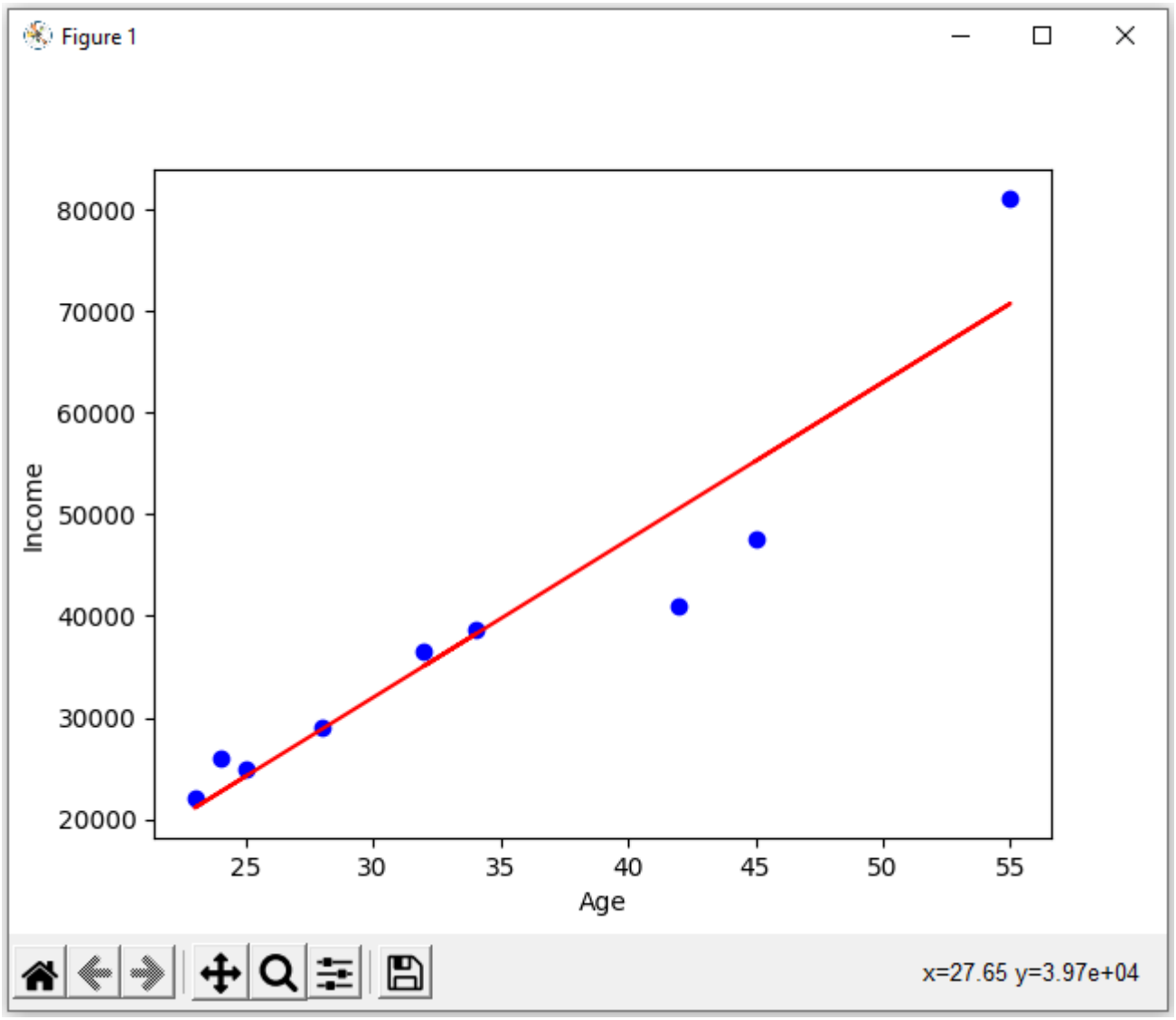
```
# to display coefficients
print("Estimated Coefficients :")
print("b0 = ",b0,"\nb1 = ",b1)
# To plot the actual points as scatter plot
plt.scatter(age, income, color = "b",marker = "o")
# TO predict response vector
response_Vec = b0 + b1*age
# To plot the regression line
plt.plot(age, response_Vec, color = "r")
# Placing labels
plt.xlabel('Age')
plt.ylabel('Income')
# To display plot
plt.show()
```

Age_Income.csv

```
Age,Income
25,25000
23,22000
24,26000
28,29000
34,38600
32,36500
42,41000
55,81000
45,47500
```

Output:

```
D:\Machine Learning\Lab>python Week6.py
Estimated Coefficients :
b0 =  -14560.45016077166
b1 =  1550.7923748277433
```



Aim:

Implement naive baye's theorem to classify the English text

Source Code:

Week7.py

```
''' Implement linear regression using python

================================
Explanation:
================================

===> To run this program you need to install the pandas Module

---> pandas Module is used to read csv files

===> To install, Open Command propmt and then execute the following command

---> pip install pandas


And, then you need to install the sklearn Module

===> Open Command propmt and then execute the following command to install
sklearn Module

---> pip install scikit-learn

Finally, you need to create dataset called "Statements_data.csv" file.

==============================
Source Code :
==============================

'''
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.naive_bayes import MultinomialNB
from sklearn.metrics import accuracy_score, confusion_matrix, precision_score,
recall_score

msglbl_data = pd.read_csv('Statements_data.csv', names=['Message', 'Label'])
print("The Total instances in the Dataset: ", msglbl_data.shape[0])
msglbl_data['labelnum'] = msglbl_data.Label.map({'pos': 1, 'neg': 0})
# place the data in X and Y Vectors
X = msglbl_data["Message"]
Y = msglbl_data.labelnum
# to split the data into train se and test set
Xtrain, Xtest, Ytrain, Ytest = train_test_split(X, Y)

count_vect = CountVectorizer()
Xtrain_dims = count_vect.fit_transform(Xtrain)
Xtest_dims = count_vect.transform(Xtest)
df =
pd.DataFrame(Xtrain_dims.toarray(),columns=count_vect.get_feature_names_out())
clf = MultinomialNB()
# to fit the train data into model
```

```
clf.fit(Xtrain_dims, Ytrain)
# to predict the test data
prediction = clf.predict(Xtest_dims)
print('******** Accuracy Metrics *********')
print('Accuracy : ', accuracy_score(Ytest, prediction))
print('Recall : ', recall_score(Ytest, prediction))
print('Precision : ',precision_score(Ytest, prediction))
print('Confusion Matrix : \n', confusion_matrix(Ytest, prediction))
print(10*"-")
# to predict the input statement
test_stmt = [input("Enter any statement to predict :")]
test_dims = count_vect.transform(test_stmt)
pred = clf.predict(test_dims)
for stmt,lbl in zip(test_stmt,pred):
        if lbl == 1:
                print("Statement is Positive")
        else:
                print("Statement is Negative")
```

Statements_data.csv

```
This is very good place,pos
I like this biryani,pos
I feel very happy,pos
This is my best work,pos
I do not like this restaurant,neg
I am tired of this stuff,neg
I can't deal with this,neg
What an idea it is,pos
My place is horrible,neg
This is an awesome place,pos
I do not like the taste of this juice,neg
I love to sing,pos
I am sick and tired,neg
I love to dance,pos
What a great holiday,pos
That is a bad locality to stay,neg
We will have good fun tomorrow,pos
I hate this food,neg
```

Output:

```
D:\Machine Learning\Lab>python Week7.py
The Total instances in the Dataset:  18
******** Accuracy Metrics *********
Accuracy :  0.6
Recall :  1.0
Precision :  0.6
Confusion Matrix :
 [[0 2]
 [0 3]]
----------
Enter any statement to predict :I hate juice
Statement is Negative
```

```
D:\Machine Learning\Lab>python Week7.py
The Total instances in the Dataset:  18
******** Accuracy Metrics *********
Accuracy :  0.6
Recall :  1.0
Precision :  0.5
Confusion Matrix :
 [[1 2]
 [0 2]]
----------
Enter any statement to predict :I love this banana
Statement is Positive
```

```
D:\Machine Learning\Lab>python Week7.py
The Total instances in the Dataset:  18
******** Accuracy Metrics *********
Accuracy :  0.8
Recall :  1.0
Precision :  0.75
Confusion Matrix :
 [[1 1]
 [0 3]]
----------
Enter any statement to predict :I do not like this place
Statement is Negative
```

Aim:

Implement the finite words classification system using Back-propagation algorithm

Source Code:

Week9.py

```
''' Implement the finite words classification system using Back-propagation
algorithm

==================================
Explanation:
==================================

===> To run this program you need to install the pandas Module

---> pandas Module is used to read csv files

===> To install, Open Command propmt and then execute the following command

---> pip install pandas


And, then you need to install the sklearn Module

===> Open Command propmt and then execute the following command to install
sklearn Module

---> pip install scikit-learn

===> Open Command propmt and then execute the following command to install
sklearn-neuralnetwork Module

---> pip install scikit-neuralnetwork


Finally, you need to create dataset called "Statements_data.csv" file.

==============================
Source Code :
==============================

'''
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.neural_network import MLPClassifier
from sklearn.metrics import accuracy_score, confusion_matrix, precision_score,
recall_score

msglbl_data = pd.read_csv('Statements_data.csv', names=['Message', 'Label'])
print("The Total instances in the Dataset: ", msglbl_data.shape[0])
msglbl_data['labelnum'] = msglbl_data.Label.map({'pos': 1, 'neg': 0})
# place the data in X and Y Vectors
X = msglbl_data["Message"]
Y = msglbl_data.labelnum
# to split the data into train se and test set
Xtrain, Xtest, Ytrain, Ytest = train_test_split(X, Y)
```

```
count_vect = CountVectorizer()
Xtrain_dims = count_vect.fit_transform(Xtrain)
Xtest_dims = count_vect.transform(Xtest)
df =
pd.DataFrame(Xtrain_dims.toarray(),columns=count_vect.get_feature_names_out())
clf = MLPClassifier(solver='lbfgs', alpha=1e-5,hidden_layer_sizes=(5, 2),
random_state=1)
# to fit the train data into model
clf.fit(Xtrain_dims, Ytrain)
# to predict the test data
prediction = clf.predict(Xtest_dims)
print('******** Accuracy Metrics *********')
print('Accuracy : ', accuracy_score(Ytest, prediction))
print('Recall : ', recall_score(Ytest, prediction))
print('Precision : ',precision_score(Ytest, prediction))
print('Confusion Matrix : \n', confusion_matrix(Ytest, prediction))
print(10*"-")
# to predict the input statement
test_stmt = [input("Enter any statement to predict :")]
test_dims = count_vect.transform(test_stmt)
pred = clf.predict(test_dims)
for stmt,lbl in zip(test_stmt,pred):
        if lbl == 1:
                print("Statement is Positive")
        else:
                print("Statement is Negative")
```

Statements_data.csv

```
This is very good place,pos
I like this biryani,pos
I feel very happy,pos
This is my best work,pos
I do not like this restaurant,neg
I am tired of this stuff,neg
I can't deal with this,neg
What an idea it is,pos
My place is horrible,neg
This is an awesome place,pos
I do not like the taste of this juice,neg
I love to sing,pos
I am sick and tired,neg
I love to dance,pos
What a great holiday,pos
That is a bad locality to stay,neg
We will have good fun tomorrow,pos
I hate this food,neg
```

Output:

```
D:\Machine Learning\Lab>python Week9.py
The Total instances in the Dataset:  18
******** Accuracy Metrics *********
Accuracy :  0.4
Recall :  0.6666666666666666
Precision :  0.5
Confusion Matrix :
 [[0 2]
 [1 2]]
----------
Enter any statement to predict :I love biryani
Statement is Positive
```

```
D:\Machine Learning\Lab>python Week9.py
The Total instances in the Dataset:  18
******** Accuracy Metrics *********
Accuracy :  0.8
Recall :  0.6666666666666666
Precision :  1.0
Confusion Matrix :
 [[2 0]
 [1 2]]
----------
Enter any statement to predict :i do not like summer
Statement is Negative
```