**3.EXTRACT THE DATA FROM DATABASE USING PYTHON**

Install MySQL Driver

Python needs a MySQL driver to access the MySQL database.

In this tutorial we will use the driver "MySQL Connector".

We recommend that you use PIP to install "MySQL Connector".

PIP is most likely already installed in your Python environment.

Navigate your command line to the location of PIP, and type the following:

Download and install "MySQL Connector":

# pip install mysql-connector-python

"""

pip install mysql-connector-python

import mysql.connector as mysql

db = mysql.connect(

host = "localhost",

charset ='utf8',

user = "root",

passwd = "2009"

)

# create a database named "mydatabase":

cursor = db.cursor()

cursor.execute("CREATE DATABASE vgnt;")

cursor.execute("SHOW DATABASES")

for x in cursor:

print(x)

cursor.execute("USE vgnt")

cursor.execute("CREATE TABLE customers (name VARCHAR(255), address VARCHAR(255))")

sql = "INSERT INTO customers (name, address) VALUES (%s, %s)"

val = ("John", "Highway 21")

cursor.execute(sql, val)

db.commit()

print(cursor.rowcount, "record inserted.")

sql = "INSERT INTO customers (name, address) VALUES (%s, %s)"

val = [

('Peter', 'Lowstreet 4'),

('Amy', 'Apple st 652'),

('Hannah', 'Mountain 21'),

('Michael', 'Valley 345'),

('Sandy', 'Ocean blvd 2'),

('Betty', 'Green Grass 1'),

('Richard', 'Sky st 331'),

('Susan', 'One way 98'),

('Vicky', 'Yellow Garden 2'),

('Ben', 'Park Lane 38'),

('William', 'Central st 954'),

('Chuck', 'Main Road 989'),

('Viola', 'Sideway 1633')

]

cursor.executemany(sql, val)

db.commit()

print(cursor.rowcount, "was inserted.")

cursor.execute("select \* from customers")

for x in cursor:

print(x)

sql = "INSERT INTO customers (name, address) VALUES (%s, %s)"

val = ("Michelle", "Blue Village")

cursor.execute(sql, val)

db.commit()

print("1 record inserted, ID:", cursor.lastrowid)

# Select records where the address contains the word "way":

sql = "SELECT \* FROM customers WHERE address ='Park Lane 38'"

cursor.execute(sql)

myresult = cursor.fetchall()

for x in myresult:

print(x)

sql = "SELECT \* FROM customers WHERE address LIKE '%way%'"

cursor.execute(sql)

result = cursor.fetchall()

# 7. Implement naive baye's theorem to classify the English text

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

his 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

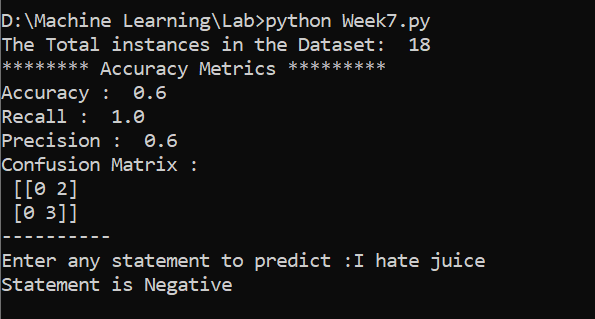
I love to dance,pos

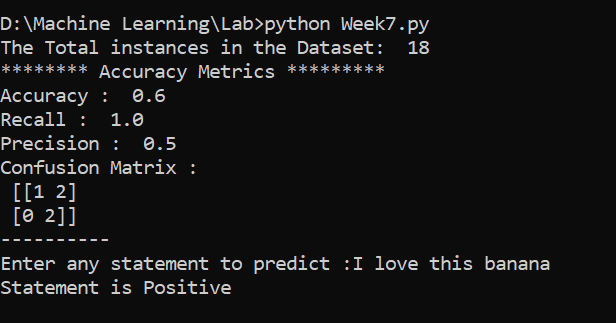
What a great holiday,pos

That is a bad locality to stay,neg

We will have good fun tomorrow,pos

I hate this food,neg





# 8.Implement an algorithm to demonstrate the significance of genetic algorithm

# genetic algorithm

def genetic\_algorithm(objective, n\_bits, n\_iter, n\_pop, r\_cross, r\_mut):

# initial population of random bitstring

pop = [randint(0, 2, n\_bits).tolist() for \_ in range(n\_pop)]

# keep track of best solution

best, best\_eval = 0, objective(pop[0])

# enumerate generations

for gen in range(n\_iter):

# evaluate all candidates in the population

scores = [objective(c) for c in pop]

# check for new best solution

for i in range(n\_pop):

if scores[i] < best\_eval:

best, best\_eval = pop[i], scores[i]

print(">%d, new best f(%s) = %.3f" % (gen,  pop[i], scores[i]))

# select parents

selected = [selection(pop, scores) for \_ in range(n\_pop)]

# create the next generation

children = list()

for i in range(0, n\_pop, 2):

# get selected parents in pairs

p1, p2 = selected[i], selected[i+1]

# crossover and mutation

for c in crossover(p1, p2, r\_cross):

# mutation

mutation(c, r\_mut)

# store for next generation

children.append(c)

# replace population

pop = children

return [best, best\_eval]

# 9. Implement the finite words classification system using Back-propagation algorithm

import pandas as pdfrom sklearn.model\_selection

import train\_test\_splitfrom sklearn.feature\_extraction.text

import CountVectorizerfrom sklearn.neural\_network

import MLPClassifierfrom 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 VectorsX = 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 modelclf.fit(Xtrain\_dims, Ytrain)

# to predict the test dataprediction = 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 statementtest\_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")

