

## **Machine Learning Lab Manual**

1. 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. (Ans: 15%)

### **ALGORITHM:**

**Step 1:** Calculate probability for each word in a text and filter the words which have a probability less than threshold probability. Words with probability less than threshold probability are irrelevant.

**Step 2:** Then for each word in the dictionary, create a probability of that word being in insincere questions and its probability insincere questions. Then finding the conditional probability to use in naive Bayes classifier.

**Step 3:** Prediction using conditional probabilities.

**Step 4:** End.

### **PROGRAM:**

```
PFIA=float(input("Enter probability that it is Friday and that a student is absent="))
PF=float(input(" probability that it is Friday="))
PABF=PFIA / PF
print("probability that a student is absent given that today is Friday using conditional probabilities=",PABF)
```

### **OUTPUT:**

```
Enter probability that it is Friday and that a student is absent= 0.03
probability that it is Friday= 0.2
probability that a student is absent given that today is Friday using conditional probabilities=
0.15
```

2. Extract the data from database using python

### **ALGORITHM:**

Step 1: Connect to MySQL from Python  
Step 2: Define a SQL SELECT Query  
Step 3: Get Cursor Object from Connection  
Step 4: Execute the SELECT query using execute() method  
Step 5: Extract all rows from a result  
Step 6: Iterate each row  
Step 7: Close the cursor object and database connection object  
Step 8: End.

## **PROCEDURE**

### **CREATING A DATABASE IN MYSQL AS FOLLOWS:**

```
CREATE DATABASE myDB;
```

```
SHOW DATABASES;
```

```
USE myDB
```

```
CREATE TABLE MyGuests (id INT, name VARCHAR(20), email VARCHAR(20));
```

```
SHOW TABLES;
```

```
INSERT INTO MyGuests (id,name,email) VALUES(1,"sairam","xyz@abc.com");
```

```
...
```

```
SELECT * FROM authors;
```

We need to install mysql-connector to connect Python with MySQL. You can use the below command to

install this in your system.

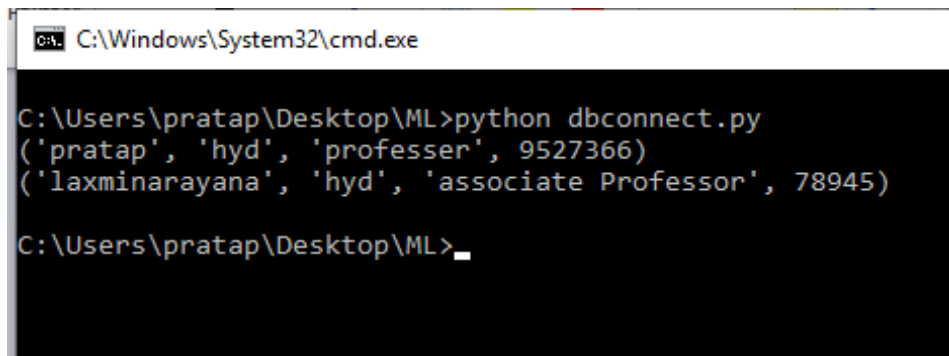
**pip install mysql-connector-python-rf**

### **PYTHON SOURCE CODE:**

```
import mysql.connector
mydb = mysql.connector.connect(
    host="localhost",
    user="root",
    password="",
    database="myDB"
)
mycursor = mydb.cursor()
mycursor.execute("SELECT * FROM MyGuests")
myresult = mycursor.fetchall()
for x in myresult:
    print(x)
```

### **OUTPUT:**

```
INSERT INTO 'myguests' ('Name','Address','Designation','Contact') VALUES  
( 'pratap','hyd','professor',9874563210),  
( 'laxminarayana','hyd','associate profesoe',7895423570)
```



A screenshot of a Windows command prompt window. The title bar shows 'C:\Windows\System32\cmd.exe'. The command prompt shows the following text:

```
C:\Users\pratap\Desktop\ML>python dbconnect.py  
( 'pratap', 'hyd', 'professer', 9527366)  
( 'laxminarayana', 'hyd', 'associate Professor', 78945)  
  
C:\Users\pratap\Desktop\ML>_
```

3. Implement k-nearest neighbours classification using python  
**ALGORITHM:**

Step 1: Load the data

Step 2: Initialize the value of k

Step 3: For getting the predicted class, iterate from 1 to total number of training data points

- i) Calculate the distance between test data and each row of training data. Here we will use Euclidean distance as our distance metric since it's the most popular method. The other metrics that can be used are Chebyshev, cosine, etc.
- ii) Sort the calculated distances in ascending order based on distance values. Get top k rows from the sorted array
- iii) Get the most frequent class of these rows i.e. Get the labels of the selected K entries
- iv) Return the predicted class ☐ If regression, return the mean of the K labels ☐ If classification, return the mode of the K labels
  - If regression, return the mean of the K labels
  - If classification, return the mode of the K labels

Step 4: End.

## PROGRAM

```
import numpy as np
from sklearn import datasets
iris = datasets.load_iris()
data = iris.data
labels = iris.target
for i in [0, 79, 99, 101]:

    print(f"index: {i:3}, features: {data[i]}, label: {labels[i]}")
    np.random.seed(42)
    indices = np.random.permutation(len(data))
    n_training_samples = 12
    learn_data = data[indices[:-n_training_samples]]
    learn_labels = labels[indices[:-n_training_samples]]
    test_data = data[indices[-n_training_samples:]]
    test_labels = labels[indices[-n_training_samples:]]
    print("The first samples of our learn set:")
    print(f"{'index':7s}{'data':20s}{'label':3s}")
    for i in range(5):

        print(f"{'i':4d} {learn_data[i]} {learn_labels[i]:3}")
        print("The first samples of our test set:")
        print(f"{'index':7s}{'data':20s}{'label':3s}")
        for i in range(5):

            print(f"{'i':4d} {learn_data[i]} {learn_labels[i]:3}")
```

**#The following code is only necessary to visualize the data of our learnset**

```
import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d import Axes3D
colours = ("r", "b")
X = []
for iclass in range(3):
```

```

X.append([], [], [])
for i in range(len(learn_data)):
    if learn_labels[i] == iclass:
        X[iclass][0].append(learn_data[i][0])
        X[iclass][1].append(learn_data[i][1])
        X[iclass][2].append(sum(learn_data[i][2:]))
    colours = ("r", "g", "y")
    fig = plt.figure()
    ax = fig.add_subplot(111, projection='3d')
    for iclass in range(3):
        ax.scatter(X[iclass][0], X[iclass][1], X[iclass][2], c=colours[iclass])
    plt.show()

```

```

def distance(instance1, instance2):
    """ Calculates the Euclidean distance between two instances"""
    return np.linalg.norm(np.subtract(instance1, instance2))

```

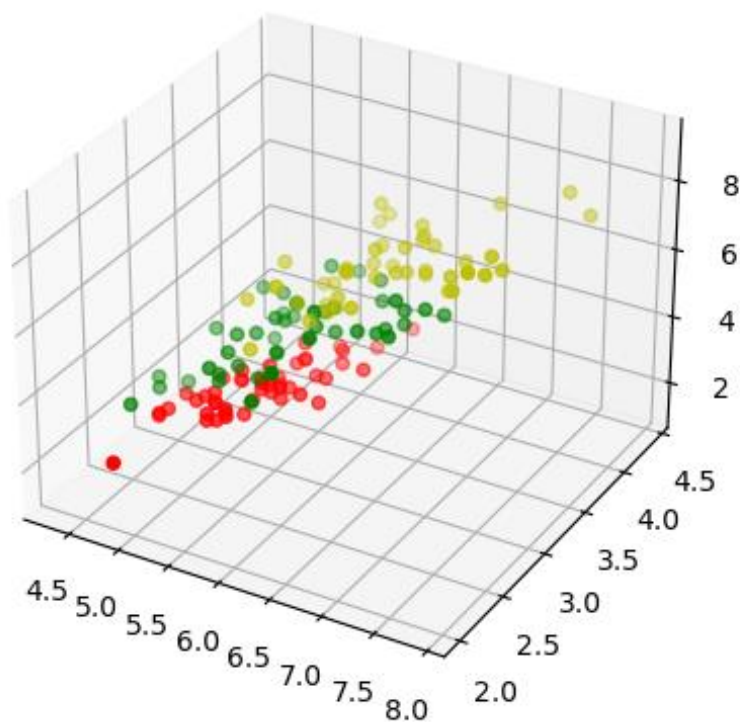
```

def get_neighbors(training_set, labels, test_instance, k, distance):
    """
    get_neighbors calculates a list of the k nearest neighbors of an instance 'test_instance'.
    The function returns a list of k 3-tuples. Each 3-tuple consists of (index, dist, label)
    """
    distances = []
    for index in range(len(training_set)):
        dist = distance(test_instance, training_set[index])
        distances.append((training_set[index], dist, labels[index]))
    distances.sort(key=lambda x: x[1])
    neighbors = distances[:k]
    return neighbors
for i in range(5):
    neighbors = get_neighbors(learn_data, learn_labels, test_data[i], 3, distance=distance)
    print("Index: ", i, "\n",
          "Testset Data: ", test_data[i], "\n",
          "Testset Label: ", test_labels[i], "\n",
          "Neighbors: ", neighbors, "\n")

```

**OUTPUT:**

```
(base) dohathi@dohathi-Compaq-15-Notebook-PC:~/ML_LAB$ python KNN.py
index: 0, features: [5.1 3.5 1.4 0.2], label: 0
index: 79, features: [5.7 2.6 3.5 1. ], label: 1
index: 99, features: [5.7 2.8 4.1 1.3], label: 1
index: 101, features: [5.8 2.7 5.1 1.9], label: 2
The first samples of our learn set:
index  data                      label
0      [6.1 2.8 4.7 1.2]          1
1      [5.7 3.8 1.7 0.3]          0
2      [7.7 2.6 6.9 2.3]          2
3      [6.  2.9 4.5 1.5]          1
4      [6.8 2.8 4.8 1.4]          1
The first samples of our test set:
index  data                      label
0      [6.1 2.8 4.7 1.2]          1
1      [5.7 3.8 1.7 0.3]          0
2      [7.7 2.6 6.9 2.3]          2
3      [6.  2.9 4.5 1.5]          1
4      [6.8 2.8 4.8 1.4]          1
```



```

Index:      2
Testset Data: [6.3 2.3 4.4 1.3]
Testset Label: 1
Neighbors: [(array([6.2, 2.2, 4.5, 1.5]), 0.26457513110645864, 1),
(array([6.3, 2.5, 4.9, 1.5]), 0.574456264653803, 1), (array([6. , 2.2, 4.
. , 1. ]), 0.5916079783099617, 1)]

Index:      3
Testset Data: [6.4 2.9 4.3 1.3]
Testset Label: 1
Neighbors: [(array([6.2, 2.9, 4.3, 1.3]), 0.200000000000000018, 1),
(array([6.6, 3. , 4.4, 1.4]), 0.2645751311064587, 1), (array([6.6, 2.9,
4.6, 1.3]), 0.3605551275463984, 1)]

Index:      4
Testset Data: [5.6 2.8 4.9 2. ]
Testset Label: 2
Neighbors: [(array([5.8, 2.7, 5.1, 1.9]), 0.31622776601683755, 2),
(array([5.8, 2.7, 5.1, 1.9]), 0.31622776601683755, 2), (array([5.7, 2.5,
5. , 2. ]), 0.33166247903553986, 2)]

```

4. Given the following data, which specify classifications for nine combinations of VAR1

and VAR2 predict a classification for a case where VAR1=0.906 and VAR2=0.606, using the result of k means clustering with 3 means (i.e., 3 centroids)

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

====> 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

```
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]

```

5. 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?

----> 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

$$P(S | MR) = P(S \cap MR) / P(MR)$$

Based on the given problem statement,

$$P(S \cap MR) = \text{The number of MedRisk with Single records} / \text{total number of Records} \\ = 2 / 10 = 0.2$$

and

$$P(MR) = \text{The number of records with MedRisk} / \text{total number of Records} \\ = 3 / 10 = 0.3$$

Then, the Conditional probability of single given medRisk

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

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
```

6. Implement linear regression using python.

**ALGORITHM:**

Step 1: Create Database for Linear Regression  
Step 2: Finding Hypothesis of Linear Regression  
Step 3: Training a Linear Regression model  
Step 4: Evaluating the model  
Step 5: Scikit-learn implementation  
Step 6: End

**# Importing Necessary Libraries**

```
import numpy as np
import matplotlib.pyplot as plt
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error, r2_score

# generate random data-set

np.random.seed(0)
x = np.random.rand(100, 1) #Generate a 2-D array with 100 rows, each row containing 1
random numbers:
y = 2 + 3 * x + np.random.rand(100, 1)
regression_model = LinearRegression() # Model initialization
regression_model.fit(x, y) # Fit the data(train the model)
y_predicted = regression_model.predict(x) # Predict

# model evaluation

rmse = mean_squared_error(y, y_predicted)
r2 = r2_score(y, y_predicted)

# printing values

print('Slope:', regression_model.coef_)
print('Intercept:', regression_model.intercept_)
print('Root mean squared error: ', rmse)
print('R2 score: ', r2)

# plotting values # data points

plt.scatter(x, y, s=10)
plt.xlabel('x-Values from 0-1')
plt.ylabel('y-values from 2-5')

# predicted values

plt.plot(x, y_predicted, color='r')
plt.show()
```

OUTPUT:

