Keshav Memorial Institute of Technology
(Approved by AICTE, Affiliated to JNTUH & Accredited by NBA)
Narayanguda, Hyderabad, Telangana 500029
(Autonomous)

B.Tech. III Year SEM II Syllabus (R18 Regulations)

COMPUTER SCIENCE AND ENGINEERING (MACHINE LEARNING)

CS604PC MACHINE LEARNING LAB MANUAL

III YEAR B.TECH CSE II-SEM

L	T	P	C
0	0	3	1.5

Course Objectives:

- The objective of this lab is to get an overview of the various machine learning techniques and can able to demonstrate them using python.

Course Outcomes: After learning the student must be able to

- understand complexity of Machine Learning algorithms and their limitations;
- understand modern notions in data analysis-oriented computing;
- . be capable of confidently applying common Machine Learning algorithms in practice and implementing their own;
- Be capable of performing experiments in Machine Learning using real-world data.

LIST OF EXPERIMENTS:

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 Bayes rule in python to get the result. (Ans: 15%)

2. Extract the data from database using python

3. Implement k-nearest neighbours classification using python

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

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?

6. Implement linear regression using python.

7. Implement Naïve Bayes theorem to classify the English text

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

AIM: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.

THEORY:

Bayes' Theorem is a way of finding a probability when we know certain other probabilities.

The formula is:

$$P(A|B) = \frac{P(A)\,P(B|A)}{P(B)}$$

Which tells us: how often A happens *given that B happens*, written **P(A|B)**,
When we know: how often B happens *given that A happens*, written **P(B|A)**
and how likely A is on its own, written **P(A)**
and how likely B is on its own, written **P(B)**

Statement of theorem [edit]

Bayes' theorem is stated mathematically as the following equation:[3]

$$P(A \mid B) = \frac{P(B \mid A)P(A)}{P(B)}$$

where A and B are events and $P(B) \neq 0$.

- $P(A \mid B)$ is a conditional probability: the probability of event A occurring given that B is true. It is also called the posterior probability of A given B.
- $P(B \mid A)$ is also a conditional probability: the probability of event B occurring given that A is true. It can also be interpreted as the likelihood of A given a fixed B because $P(B \mid A) = L(A \mid B)$.
- $P(A)$ and $P(B)$ are the probabilities of observing A and B respectively without any given conditions; they are known as the marginal probability or prior probability.
- A and B must be different events.

Proof [edit]

For events [edit]

Bayes' theorem may be derived from the definition of conditional probability:

$$P(A \mid B) = \frac{P(A \cap B)}{P(B)}, \text{ if } P(B) \neq 0,$$
$$P(B \mid A) = \frac{P(B \cap A)}{P(A)}, \text{ if } P(A) \neq 0,$$

where $P(A \cap B)$ is the joint probability of both A and B being true. Because

$$P(B \cap A) = P(A \cap B),$$
$$\Rightarrow P(A \cap B) = P(A \mid B)P(B) = P(B \mid A)P(A)$$
$$\Rightarrow P(A \mid B) = \frac{P(B \mid A)P(A)}{P(B)}, \text{ if } P(B) \neq 0.$$

SOURCE CODE:

```
# calculate the probability  that a student is absent given that today is Friday

# calculate P(A|F) given P(F), P(F^A)
def bayes_theorem(p_f, p_f_and_a):
p_a_given_f = (p_f_and_a) / p_f   return
p_a_given_f
# P(F) p_f = 0.2
 # P(F^A)
p_f_and_a = 0.03
# calculate P(A|F)
result = bayes_theorem(p_f, p_f_and_a)
# summarize
print('P(A|F) = %.3f%%' % (result * 100))
```

OUTPUT:

```
P(A|F) = 15.000%
```

AIM: 2. Extract the data from database using python.

THEORY:

SQLite Database

It is a disk-based database that does not need any separate server process and installation of anything on your computer.

You even don't need to install any package in your Python distribution.

The package sqlite3 is part of the standard library since Python 2.5. We will use this sqlite3 package to create our own database.

What is a database?

How to create a database table using SQLite database?

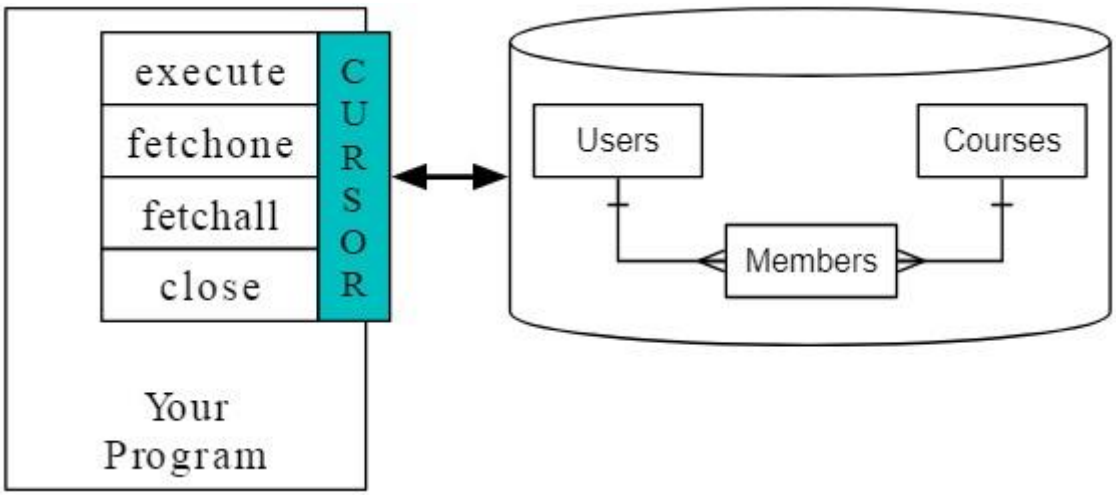
How to insert data into a database table using SQLite database?

What are the methods to retrieve data from the database table using SQLite database?

hat is a database? A database is a file that is organized for storing data. Data in the database is arranged in the form of tables, rows, and columns.

There are many different database systems available such as Oracle, MySQL, Microsoft SQL Server, PostgreSQL, and SQLite. Each of these databases is used for a variety of purposes. This tutorial discusses how to use SQLite in Python, as it is inbuilt in Python and easy to use.

What are the Functions available in SQLite? The following diagram shows the functions available in SQLite database.



First, we need to connect to the database using connect() function. connect() function accepts one parameter that is the name of the database. If the database is already present in the system then connect() function open the database. Otherwise, the connect() function creates a new database Once the database is created using connect() function. The database looks like a file to our program. We open the normal file using an open() function and then perform read and write operations. Similarly, the cursor() function is used to open the database.

After the database is opened using cursor() function, we can perform different operations on the database. Like, we can create a new table, add new data into the table. We can update the data in the table and delete the rows of the table. Also, we can drop the entire table. Finally, we can fetch the content of the table using fetchone() or fetchall() function and display it to the user.

How to create a database table using an SQLite database? In the following fragment of Python code, we have established the connection to the database using connect() function. Then the database was opened using cursor() function. Later, we have created a table by executing a CRETE TABLE query using the execute() function. Finally, we have closed the connection using the close() function.

How to insert data into database table? The next Fragment of Python code shows how to insert data into database.

How to retrieve data from the database table using SQLite? The data from the database is extracted using a SELECT query. Then three methods are used to demonstrate how to display the results to the user. First, The return value of the SELECT query was stored into a variable, data in this example. Then the one row from the data is extracted using the for loop and displayed to the user. Second, the fetchone() function is used to extract the first rows from the cursor and displayed. Third, fetchall() function is used to extract all the rows from the cursor and displayed.

SOURCE CODE:

```
#write your python program here by reading the problem statement import
sqlite3
 conn = sqlite3.connect('StudentInfo') cur = conn.cursor()
cur.execute('DROP TABLE IF EXISTS student')
cur.execute('CREATE TABLE student (name TEXT, rno INTEGER)')
 cur.execute('INSERT INTO student (name, rno) VALUES (?, ?)',('Mahesh', 45))
cur.execute('INSERT INTO student (name, rno) VALUES (?, ?)',('Rahul', 30))
cur.execute('SELECT * FROM student')
#print ("Data in the table using fetchall() function:")
print (cur.fetchall())     conn.close()
```

OUTPUT: [('Mahesh', 45), ('Rahul', 30)]

AIM:3.Implement k-nearest neighbours classification using python.

THEORY:

K-Nearest Neighbor Algorithm

Training algorithm:

For each training example (x, f (x)), add the example to the list training examples
Classification algorithm:

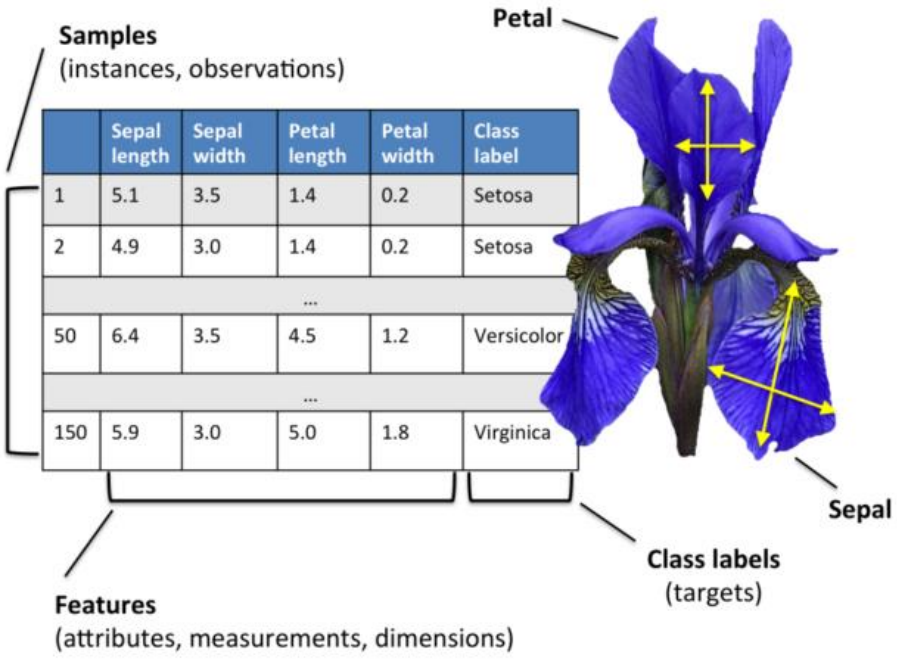
Given a query instance xq to be classified,

Let x1 . . .xk denote the k instances from training examples that are nearest to xq

Return $\hat{f}(x_q) \leftarrow \frac{\sum_{i=1}^{k} f(x_i)}{k}$ Where, f(xi) function to calculate the mean value of the k nearest training examples.

Data Set:

Iris Plants Dataset: Dataset contains 150 instances (50 in each of three classes)
Number of Attributes: 4 numeric, predictive attributes and the Class.



	Sepal length	Sepal width	Petal length	Petal width	Class label
1	5.1	3.5	1.4	0.2	Setosa
2	4.9	3.0	1.4	0.2	Setosa
...					
50	6.4	3.5	4.5	1.2	Versicolor
...					
150	5.9	3.0	5.0	1.8	Virginica

	sepal-length	sepal-width	petal-length	petal-width	Class
0	5.1	3.5	1.4	0.2	Iris-setosa
1	4.9	3.0	1.4	0.2	Iris-setosa
2	4.7	3.2	1.3	0.2	Iris-setosa
3	4.6	3.1	1.5	0.2	Iris-setosa
4	5.0	3.6	1.4	0.2	Iris-setosa

SOURCE CODE:

```
from sklearn.datasets import load_iris
iris = load_iris()
#print("Feature Names:",iris.feature_names,"Iris Data:",iris.data,"Target
Names:",iris.target_names,"Target:",iris.target)
from sklearn.model_selection import train_test_split
```

```
X_train, X_test, y_train, y_test = train_test_split( iris.data, iris.target, test_size =
.25,random_state=42)
from sklearn.neighbors import KNeighborsClassifier
clf = KNeighborsClassifier() clf.fit(X_train, y_train)
print(" Accuracy=",clf.score(X_test, y_test))
print("Predicted Data")
print(clf.predict(X_test))
prediction=clf.predict(X_test)
print("Test data :")
print(y_test) diff=prediction-y_test
print("Result is ") print(diff)
print('Total no of samples misclassied =', sum(abs(diff)))
```

OUTPUT:

```
Accuracy= 1.0
Predicted Data
[1 0 2 1 1 0 1 2 1 1 2 0 0 0 0 1 2 1 1 2 0 2 0 2 2 2 2 2 0 0 0 0 1 0 0 2
1
 0]                                                                    Test
data :
[1 0 2 1 1 0 1 2 1 1 2 0 0 0 0 1 2 1 1 2 0 2 0 2 2 2 2 2 0 0 0 0 1 0 0 2
1
 0]
Result is
[0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0
 0]
Total no of samples misclassied = 0
```

AIM: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 kmeans 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

THEORY:

Introduction to K-means Clustering:

k-means is one of the simplest unsupervised learning algorithms that solve the clustering problems. The procedure follows a **simple** and easy way to classify a given data set through a certain number of clusters (assume **k** clusters). The main idea is to define **k** centers, one for each cluster.

K-Means is applied in:

The *K*-means clustering algorithm is used to find groups which have not been explicitly labeled in the data and to find patterns and make better decisions.. Once the algorithm has been run and the groups are defined, any new data can be easily assigned to the most relevant group.

Customer Profiling:

market segmentation,

computer vision

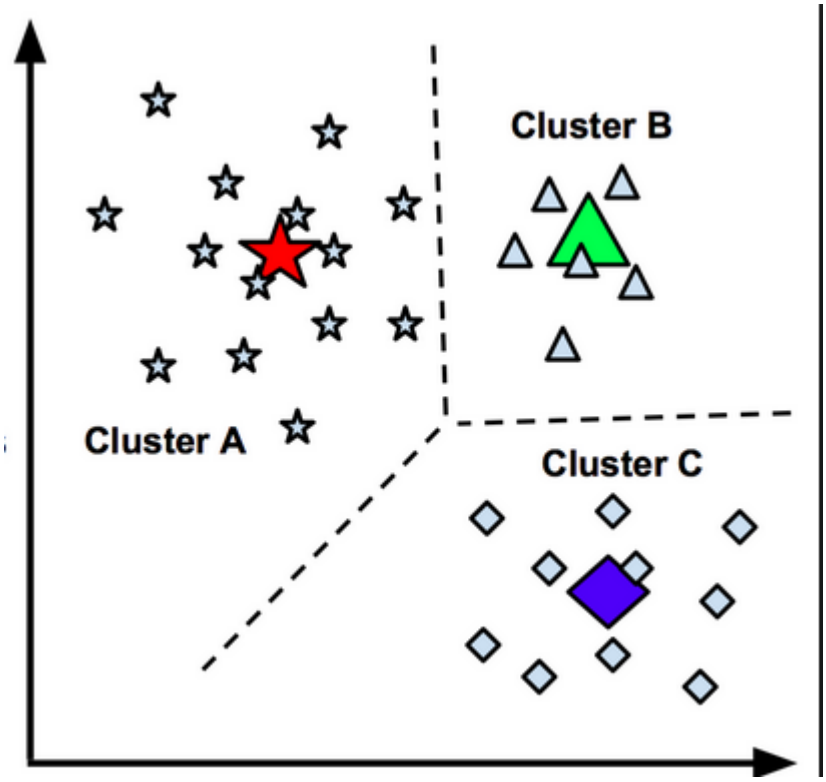
Geo-statistics

Astronomy

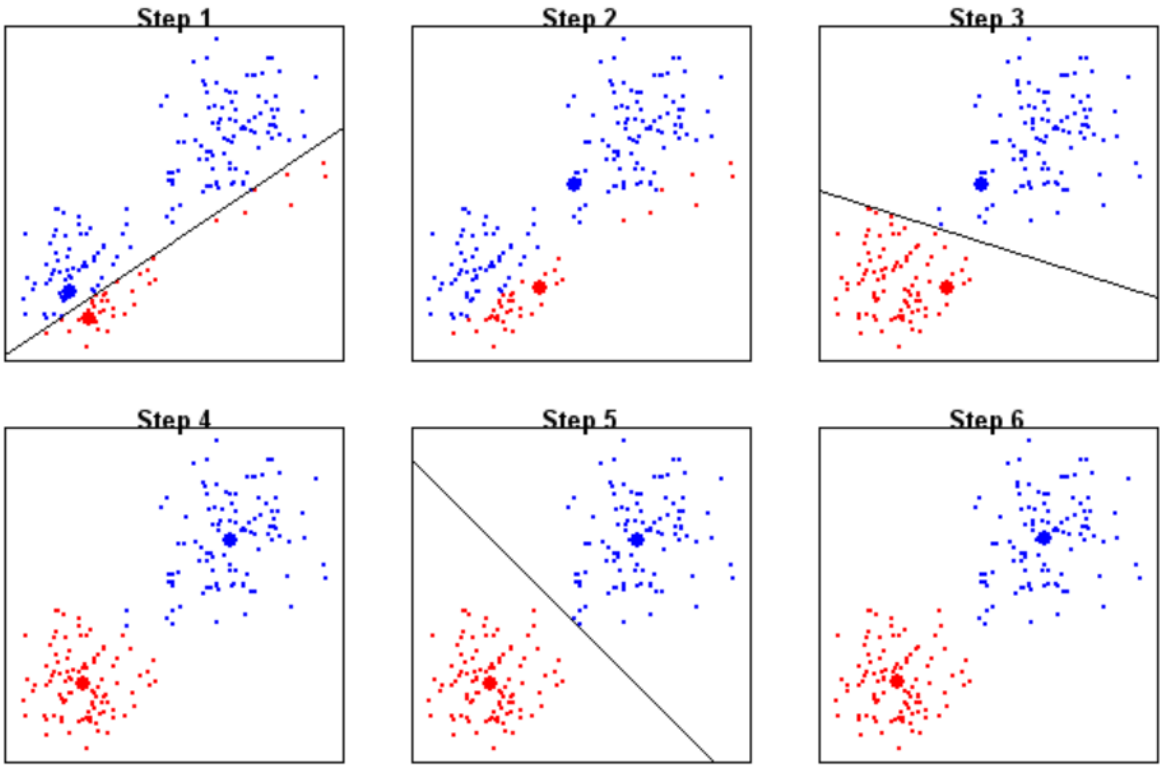
Algorithm:

To start with k-means algorithm, you first have to randomly initialize points called the cluster centroids (**K**). K-means is an **iterative algorithm** and it does two steps:

1. **Cluster assignment** 2. **Move centroid step**.



1. **Cluster assignment**
the algorithm goes through each of the data points and depending on which cluster is closer, It assigns the data points to one of the three cluster centroids.
2. Move centroid
Here, K-means moves the centroids to the average of the points in a cluster. In other words, the algorithm calculates the average of all the points in a cluster and moves the centroid to that average location.
This process is repeated until there is no change in the clusters (or possibly until some other stopping condition is met). K is chosen randomly or by giving specific initial starting points by the user.



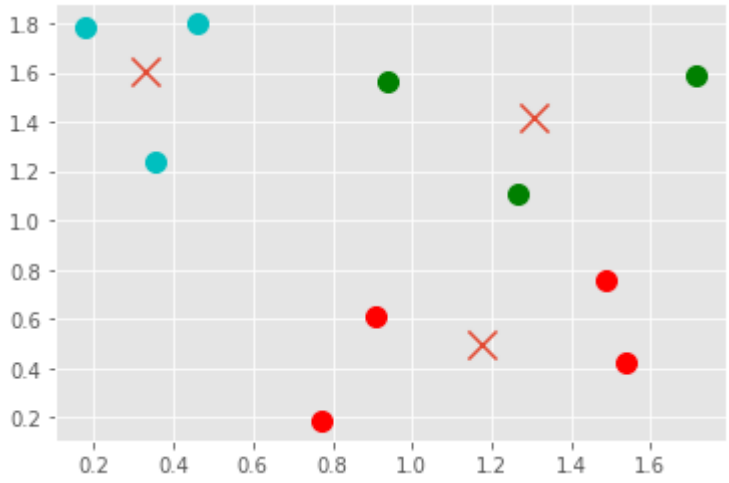
K-Means at work

SOURCE CODE:

```
import numpy as np
from sklearn.cluster
 import KMeans
X2=
np.array([[1.713,1.586],[0.18,1.786],[0.353,1.24],[0.94,1.566],[1.486,0.759],[1.266,1.106],[1.5
4,0.419],[0.459,1.799],[0.773,0.186],[0.906,0.606]])
#print(X2)
kmeans = KMeans(n_clusters=3,random_state=13) kmeans.fit(X2)
centroid = kmeans.cluster_centers_
labels = kmeans.labels_
print(centroid)
print(labels)
 for i in range(len(X2)):
   print ("coordinate:" , X2[i], "label:", labels[i])
```

OUTPUT:

```
[[1.30633333 1.41933333]
 [1.17625    0.4925    ]
 [0.33066667 1.60833333]]                                    [0
2 2 0 1 0 1 2 1 1]
coordinate: [1.713 1.586] label: 0
coordinate: [0.18  1.786] label: 2
coordinate: [0.353 1.24 ] label: 2
coordinate: [0.94  1.566] label: 0
coordinate: [1.486 0.759] label: 1
coordinate: [1.266 1.106] label: 0
coordinate: [1.54  0.419] label: 1
coordinate: [0.459 1.799] label: 2
coordinate: [0.773 0.186] label: 1
coordinate: [0.906 0.606] label: 1
```



AIM: 5.Find the unconditional probability of `golf' and the conditional probability of `single' given `medRisk' in the following dataset?

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.

THEORY:

CONDITIONAL PROBABILITY WITH EXAMPLES:

As the name suggests, Conditional Probability is the probability of an event under some given condition. And based on the condition our sample space reduces to the conditional element.

For example, find the probability of a person subscribing for the insurance given that he has taken the house loan. Here sample space is restricted to the persons who have taken house loan.

To understand Conditional probability, it is recommended to have an understanding of probability basics like Mutually Exclusive and Independent Events, Joint, Union and Marginal Probabilities and Probability vs Statistics etc.

In probability theory, conditional probability is a measure of the probability of an event occurring given that another event has occurred. If the event of interest is A and the event B is known or assumed to have occurred, "the conditional probability of A given B", or "the probability of A under the condition B", is usually written as P(A | B), or sometimes PB(A) or P(A / B)

Now the question may come like why use conditional probability and what is its significance in Data Science?

Let's take a real-life example. Probability of selling a TV on a given normal day may be only 30%. But if we consider that given day is Diwali, then there are much more chances of selling a TV. The conditional Probability of selling a TV on a day given that Day is Diwali might be 70%. We can represent those probabilities as P(TV sell on a random day) = 30%. P(TV sell given that today is Diwali) = 70%.

So Conditional Probability helps Data Scientists to get better results from the given data set and for Machine Learning Engineers, it helps in building more accurate models for predictions.

SOURCE CODE:

```
import pandas as pd
df=pd.read_csv("https://raw.githubusercontent.com/KMITDS/CS601PC/main/individual.csv")
#print(df.head(10))
# we need to find P(recreation=golf) and P(status=single/credit-worthiness=medRisk)
# Pr(status='single'| credit-worthiness='medRisk') = Pr(status='single' &
creditworthiness='medRisk') / Pr(credit-worthiness='medRisk') df[(df.status=='single')
& (df.cw=='medRisk')] df.columns df[(df.cw=='medRisk')]
print("P(recreation=golf)")  print(df[(df.recreation=='golf')].shape[0]/df.shape[0])
print(" P(status=single/credit-worthiness=medRisk)")
print(df[(df.status=='single') & (df.cw=='medRisk')].shape[0]/df[(df.cw=='medRisk')].shape[0])
```

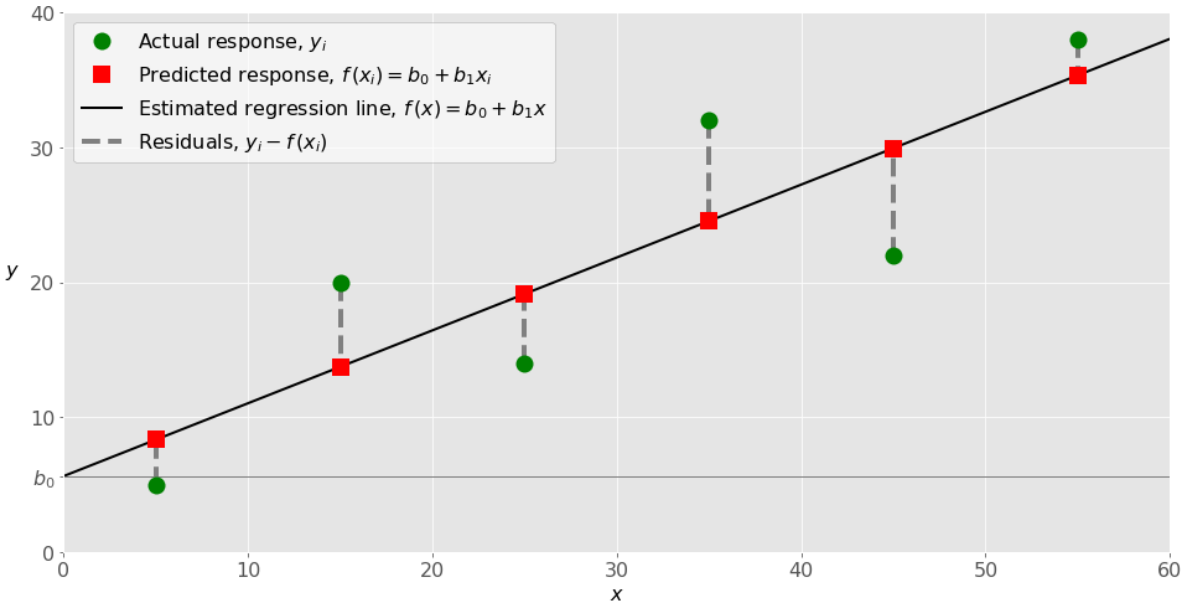
OUTPUT:

```
P(recreation=golf)
0.4
 P(status=single/credit-
worthiness=medRisk)
0.6666666666666666
```

AIM:6. Implement linear regression using python.

THEORY:

The following figure illustrates simple linear regression:



Python Packages for Linear Regression

The package **NumPy** is a fundamental Python scientific package that allows many high-performance operations on single- and multi-dimensional arrays.

It also offers many mathematical routines. Of course, it's open source.

The package **scikit-learn** is a widely used Python library for machine learning, built on top of NumPy and some other packages.

It provides the means for preprocessing data, reducing dimensionality, implementing regression, classification, clustering, and more.

Like NumPy, scikit-learn is also open source.

Simple Linear Regression With scikit-learn

Let's start with the simplest case, which is simple linear regression.

There are five basic steps when you're implementing linear regression:

1.Import the packages and classes you need.

2.Provide data to work with and eventually do appropriate transformations.

3.Create a regression model and fit it with existing data.

4.Check the results of model fitting to know whether the model is satisfactory.

5.Apply the model for predictions.

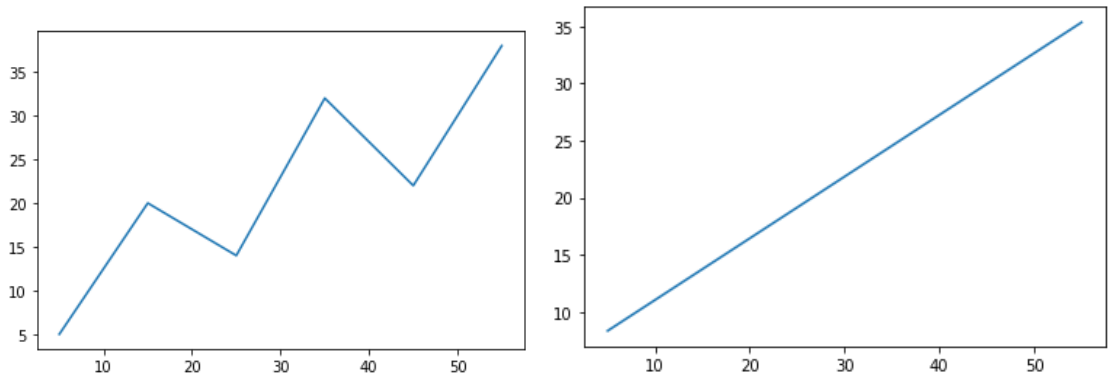
These steps are more or less general for most of the regression approaches and implementations.

SOURCE CODE:

```
import numpy as np
from sklearn.linear_model import LinearRegression
x = np.array([5, 15, 25, 35, 45, 55]).reshape((-1, 1))
y = np.array([5, 20, 14, 32, 22, 38]) model =
LinearRegression().fit(x, y) r_sq = model.score(x,y)
print('coefficient of determination:', r_sq)
print('intercept:', model.intercept_)
print('slope:', model.coef_)
 y_pred = model.predict(x)
print('predicted response:', y_pred, sep='\n')
```

OUTPUT:

```
coefficient of determination:
0.715875613747954
intercept: 5.633333333333333
slope: [0.54]
predicted response:
[ 8.33333333 13.73333333 19.13333333 24.53333333 29.93333333 35.33333333]
```



AIM:7. Implement Naïve Bayes theorem to classify the English text

THEORY:

LEARN_NAIVE_BAYES_TEXT (Examples, V)

Examples is a set of text documents along with their target values. V is the set of all possible target values. This function learns the probability terms $P(w_k|v_j)$, describing the probability that a randomly drawn word from a document in class v_j will be the English word w_k. It also learns the class prior probabilities $P(v_j)$.

1. *collect all words, punctuation, and other tokens that occur in Examples*
   - *Vocabulary* ← **c** the set of all distinct words and other tokens occurring in any text document from *Examples*

2. *calculate the required $P(v_j)$ and $P(w_k|v_j)$ probability terms*

   - For each target value v_j in V do
     - $docs_j$ ← the subset of documents from *Examples* for which the target value is *vj*
     - $P(v_j)$ ← $|docs_j| / |\text{Examples}|$
     - $Text_j$ ← a single document created by concatenating all members of $docs_j$
     - n ← total number of distinct word positions in $Text_j$
     - for each word w_k in *Vocabulary*
       - n_k ← number of times word $\boldsymbol{w_k}$ occurs in $Text_j$
       - $P(w_k|v_j)$ ← $(n_k + 1) / (n + |Vocabulary|)$

CLASSIFY_NAIVE_BAYES_TEXT (Doc)

Return the estimated target value for the document Doc. a_i denotes the word found in the i^{th} position within Doc.

- *positions* ← all word positions in *Doc* that contain tokens found in *Vocabulary*
- Return V_{NB}, where

$$v_{NB} = \underset{v_j \in V}{\text{argmax}} P(v_j) \prod_{i \in positions} P(a_i|v_j)$$

Data set:

	Text Documents	Label
1	I love this sandwich	pos
2	This is an amazing place	pos
3	I feel very good about these beers	pos
4	This is my best work	pos
5	What an awesome view	pos
6	I do not like this restaurant	neg
7	I am tired of this stuff	neg
8	I can't deal with this	neg
9	He is my sworn enemy	neg
10	My boss is horrible	neg
11	This is an awesome place	pos
12	I do not like the taste of this juice	neg
13	I love to dance	pos
14	I am sick and tired of this place	neg
15	What a great holiday	pos
16	That is a bad locality to stay	neg
17	We will have good fun tomorrow	pos
18	I went to my enemy's house today	neg

Basic knowledge

Confusion Matrix

		Actual	
		Positive	Negative
Predicted	Positive	True Positive	False Positive
	Negative	False Negative	True Negative

True positives: data points labelled as positive that are actually positive
False positives: data points labelled as positive that are actually negative
True negatives: data points labelled as negative that are actually negative
False negatives: data points labelled as negative that are actually positive

$$\text{Recall} = \frac{True\ Positive}{True\ Positive + False\ Negative}$$

$$= \frac{True\ Positive}{Total\ Actual\ Positive}$$

		Actual	
		Positive	Negative
Predicted	Positive	True Positive	False Positive
	Negative	False Negative	True Negative

$$\text{Precision} = \frac{True\ Positive}{True\ Positive + False\ Positive}$$

$$= \frac{True\ Positive}{Total\ Predicted\ Positive}$$

		Actual	
		Positive	Negative
Predicted	Positive	True Positive	False Positive
	Negative	False Negative	True Negative

Example:

		Actual	
		Positive	Negative
Predicted	Positive	1 TP	3 FP
	Negative	0 FN	1 TN

$$\text{Precision} = \frac{TP}{TP + FP} = \frac{1}{1+3} = 0.25$$

$$\text{Recall} = \frac{TP}{TP + FN} = \frac{1}{1+0} = 1$$

Accuracy: how often is the classifier correct?

$$\text{Accuracy} = \frac{\text{TP} + \text{TN}}{\text{Total}} = \frac{1+1}{5} = 0.4$$

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 import metrics
msg=pd.read_csv('https://raw.githubusercontent.com/KMITDS/CS601PC/main/naivetext.csv'
,names=['message','label']) print('The
dimensions of the dataset',msg.shape)
msg['labelnum']=msg.label.map({'pos':1,'neg':0})
X=msg.message
y=msg.labelnum
#splitting the dataset into train and test data
xtrain,xtest,ytrain,ytest=train_test_split(X,y,random_state=42)
print ('\n the total number of Training Data :',ytrain.shape)
print ('\n the total number of Test Data :',ytest.shape)
#output the words or Tokens in the text documents
```

```
cv = CountVectorizer() xtrain_dtm = cv.fit_transform(xtrain)
xtest_dtm=cv.transform(xtest)
#print('\n The words or Tokens in the text documents \n')
#print(cv.get_feature_names())
df=pd.DataFrame(xtrain_dtm.toarray(),columns=cv.get_feature_names())
 # Training Naive Bayes (NB) classifier on training data.
clf = MultinomialNB().fit(xtrain_dtm,ytrain)
predicted = clf.predict(xtest_dtm)
#printing accuracy, Confusion matrix, Precision and Recall
print('\n Accuracy of the classifier is',metrics.accuracy_score(ytest,predicted))
print('\n Confusion matrix')
print(metrics.confusion_matrix(ytest,predicted))
print('\n The value of Precision', metrics.precision_score(ytest,predicted))
print('\n The value of Recall', metrics.recall_score(ytest,predicted))
```

OUTPUT:

```
The dimensions of the dataset (18, 2)
the total number of Training Data : (13,)
the total number of Test Data : (5,)
Accuracy of the classifier is 0.6
Confusion matrix
[[2 0]
 [2 1]]
The value of Precision 1.0
The value of Recall 0.3333333333333333
```

AIM: 8. Build an ANN by implementing the Back-propagation algorithm.
THEORY:

BACKPROPAGATION Algorithm

BACKPROPAGATION (*training_example*, η, n_{in}, n_{out}, n_{hidden})

Each training example is a pair of the form $(\vec{x}, \vec{t})$, *where* $(\vec{x})$ *is the vector of network input values,* $(\vec{t})$ *and is the vector of target network output values.*
η *is the learning rate (e.g., .05).* n_{in} *is the number of network inputs,* n_{hidden} *the number of units in the hidden layer, and* n_{out} *the number of output units.*
The input from unit i into unit j is denoted x_{ji}, *and the weight from unit i to unit j is denoted* w_{ji}

- Create a feed-forward network with n_i inputs, n_{hidden} hidden units, and n_{out} output units.
- Initialize all network weights to small random numbers
- Until the termination condition is met, Do
  - For each $(\vec{x}, \vec{t})$, in training examples, Do

    Propagate the input forward through the network:
    1. Input the instance $\vec{x}$ to the network and compute the output o_u of every unit u in the network.

    Propagate the errors backward through the network:

2. For each network output unit k, calculate its error term δ_k

$$\delta_k \leftarrow o_k(1 - o_k)(t_k - o_k)$$

3. For each hidden unit h, calculate its error term δ_h

$$\delta_h \leftarrow o_h(1 - o_h) \sum_{k \in outputs} w_{h,k}\delta_k$$

4. Update each network weight w_{ji}

$$w_{ji} \leftarrow w_{ji} + \Delta w_{ji}$$

Where

$$\Delta w_{ji} = \eta \delta_j x_{i,j}$$

Training Examples:

Example	Sleep	Study	Expected % in Exams
1	2	9	92
2	1	5	86
3	3	6	89

Normalize the input

Example	Sleep	Study	Expected % in Exams
1	2/3 = 0.66666667	9/9 = 1	0.92
2	1/3 = 0.33333333	5/9 = 0.55555556	0.86
3	3/3 = 1	6/9 = 0.66666667	0.89

SOURCE CODE:

```
import numpy as np
X = np.array(([2, 9], [1, 5], [3, 6]), dtype=float)
 # two inputs [sleep,study] y = np.array(([92], [86], [89]), dtype=float)
# one output [expected % in exams]
X = X/np.amax(X,axis=0)
#maximum of X array longitudinally
y = y/100
#Sigmoid Function
def sigmoid (x): return 1/(1 + np.exp(-x))
 #Derivative of Sigmoid Function
 def derivatives_sigmoid(x): return x * (1 - x)
 #Variable initialization
 epoch=5 #Setting training iterations
lr=0.1 #Setting learning rate
 np.random.seed(1)
inputlayer_neurons = 2 #number of features in data set
 hiddenlayer_neurons = 3 #number of hidden layers neurons
 output_neurons = 1 #number of neurons at output layer
#weight and bias initialization
wh=np.random.uniform(size=(inputlayer_neurons,hiddenlayer_neurons))
bh=np.random.uniform(size=(1,hiddenlayer_neurons))
wout=np.random.uniform(size=(hiddenlayer_neurons,output_neurons))
bout=np.random.uniform(size=(1,output_neurons))
 #draws a random range of numbers uniformly of dim x*y
```

```
 for i in range(epoch):
 #Forward Propagation
 hinp1=np.dot(X,wh)
 hinp=hinp1 + bh
hlayer_act = sigmoid(hinp)
outinp1=np.dot(hlayer_act,wout)
 outinp= outinp1+bout
 output = sigmoid(outinp)
 #Backpropagation
 EO = y-output
 outgrad = derivatives_sigmoid(output)
 d_output = EO * outgrad
EH = d_output.dot(wout.T)
 hiddengrad = derivatives_sigmoid(hlayer_act)
#how much hidden layer wts contributed to error
d_hiddenlayer = EH * hiddengrad
wout += hlayer_act.T.dot(d_output) *lr
 # dotproduct of nextlayererror and currentlayerop
wh += X.T.dot(d_hiddenlayer) *lr
 print("Input: \n" + str(X))
 print("Actual Output: \n" + str(y))
 print("Predicted Output: \n" ,output)
```

OUTPUT:

```
Input:
 [[0.66666667 1. ]
 [0.33333333 0.55555556]
 [1. 0.66666667]]
 Actual Output:
 [[0.92]
 [0.86]
 [0.89]]
 Predicted Output:
 [[0.79107772]
[0.77932172]
 [0.79358328]
```

VIVA Questions :

1. What is machine learning?
2. Define supervised learning
3. Define unsupervised learning
4. Define semi supervised learning
5. Define reinforcement learning
6. What do you mean by hypotheses?
7. What is classification?
8. What is clustering?
9. Define precision, accuracy and recall
10. Define entropy
11. Define regression
12. How Knn is different from k-means clustering
13. What is concept learning?
14. Define specific boundary and general boundary
15. Define target function
16. Define decision tree
17. What is ANN
18. Explain gradient descent approximation
19. State Bayes theorem
20. Define Bayesian belief networks
21. Differentiate hard and soft clustering
22. Define variance
23. What is inductive machine learning?
24. Why K nearest neighbor algorithm is lazy learning algorithm
25. Why naïve Bayes is naïve
26. Mention classification algorithms
27. Define pruning
28. Differentiate Clustering and classification
29. Mention clustering algorithms
30. Define Bias
31. What is learning rate? Why it is needed