Practical No.1

Title: Implementation of Logic programming

Aim: Understanding basics of prolog and implementation of prolog to solve tower of hanoi problem, water jug problem, tic-tac-toe problem and 8- Puzzle Problem.

Introduction:

What is Prolog :-

Prolog (programming in logic) is one of the most widely used programming languages in artificial intelligence research. As opposed to imperative languages such as C or Java (the latter of which also happens to be object-oriented) it is a declarative programming language. That means, when implementing the solution to a problem, instead of specifying how to achieve a certain goal in a certain situation, we specify what the situation (rules and facts) and the goal (query) are and let the Prolog interpreter derive the solution for us. Prolog is very useful in some problem areas, such as artificial intelligence, natural language processing, databases, . . . , but pretty useless in others, such as graphics or numerical algorithms.

Example : -

```
male(james1).
male(charles1).
male(charles2).
male(james2).
male(george1).
```

```
female(catherine).
female(elizabeth).
female(sophia).
parent(charles1, james1).
parent(elizabeth, james1).
parent(charles2, charles1).
parent(catherine, charles1).
parent(james2, charles1).
parent(sophia, elizabeth).
parent(george1, sophia).

mother(M,X):- parent(X,M),female(M),write(M),write(' is Mother of '),write(X),nl.

father(F,X):- parent(X,F),male(F).

sibling(S1,S2):- parent(S1,X), parent(S2,X).

grandfather(G,X) :- parent(Y,G),parent(X,Y).
```

Exercise - Write a prolog program to solve Tower of hanoi Problem

Implementation:

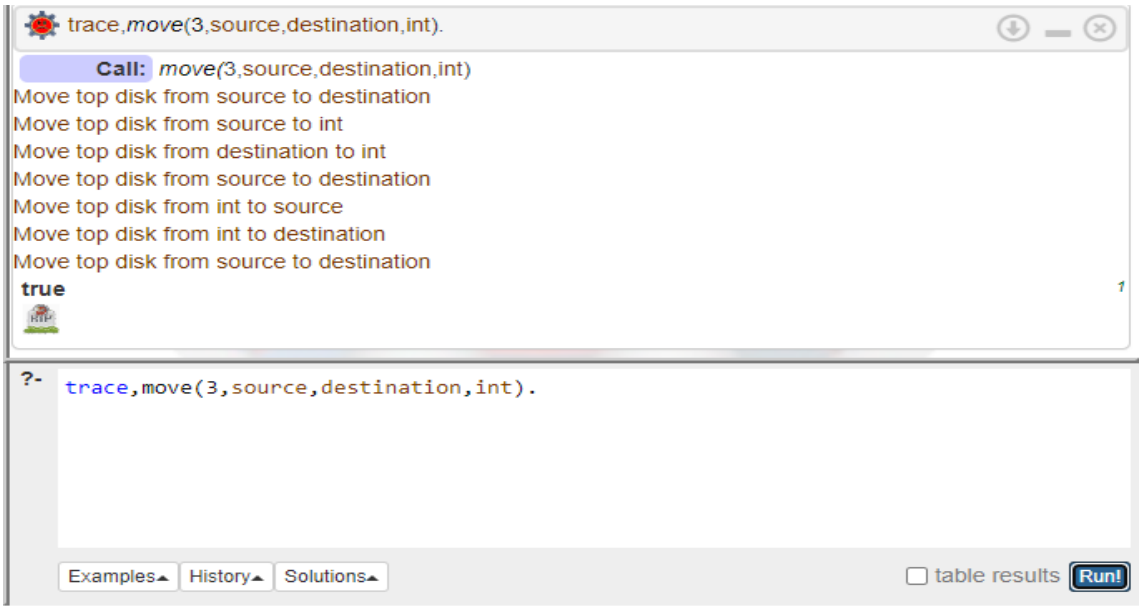
Program:

Q1). Write a prolog program to solve Tower of hanoi Problem.

```
move(1,X,Y,_) :-
```

```
    write('Move top disk from '),
    write(X),
    write(' to '),
    write(Y),
    nl.
move(N,X,Y,Z) :-
    N>1,
    M is N-1,
    move(M,X,Z,Y),
    move(1,X,Y,_),
    move(M,Z,Y,X).
```

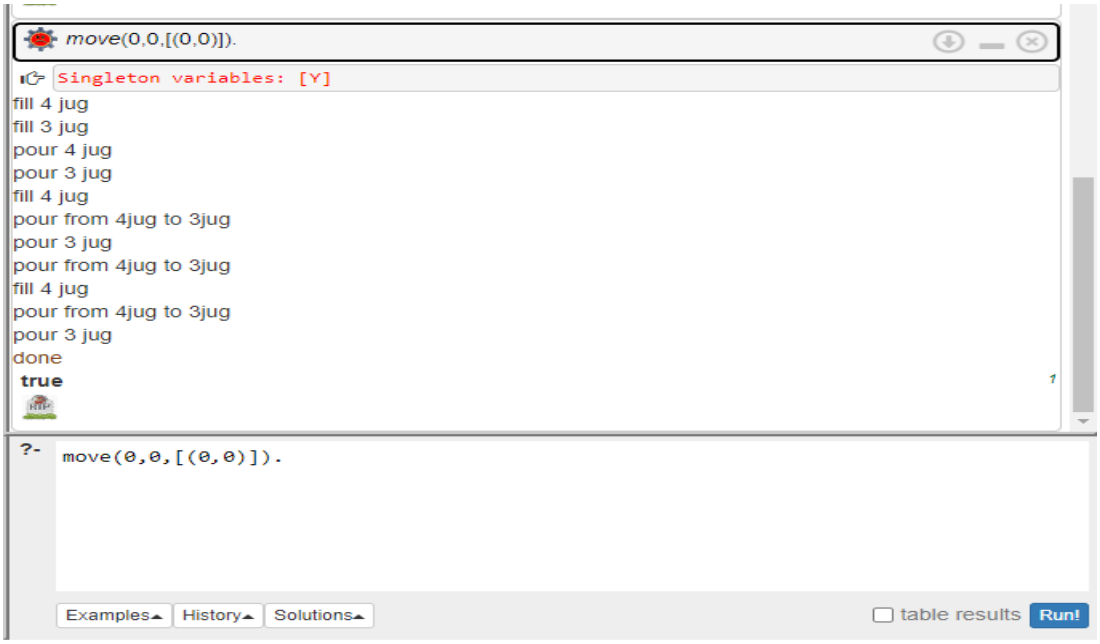
Output:



Q2). Write a prolog program to solve Water Jug Problem.

```
member(X,[X|_]).

member(X,[Y|Z]):-member(X,Z).


move(X,Y,_):-X=:=2,Y=:=0,write('done'),!.

move(X,Y,Z):-X<4,\+member((4,Y),Z),write("fill 4 jug"),nl,move(4,Y,[(4,Y)|Z]).

move(X,Y,Z):-Y<3,\+member((X,3),Z),write("fill 3 jug"),nl,move(X,3,[(X,3)|z]).

move(X,Y,Z):-X>0,\+member((0,Y),Z),write("pour 4 jug"),nl,move(0,Y,[(0,Y)|Z]).

move(X,Y,Z):-Y>0,\+member((X,0),Z),write("pour 3 jug"),nl,move(X,0,[(X,0)|Z]).

move(X,Y,Z):-P is X+Y,P>=4,Y>0,K is 4-X,M is Y-K,\+member((4,M),Z),write("pour from
3jug to 4jug"),nl,move(4,M,[(4,M)|Z]).

move(X,Y,Z):-P is X+Y,P>=3,X>0,K is 3-Y,M is X-K,\+member((M,3),Z),write("pour from
4jug to 3jug"),nl,move(M,3,[(M,3)|Z]).

move(X,Y,Z):-K is X+Y,K<4,Y>0,\+member((K,0),Z),write("pour from 3jug to
4jug"),nl,move(K,0,[(K,0)|Z]).

move(X,Y,Z):-K is X+Y,K<3,X>0,\+member((0,K),Z),write("pour from 4jug to
3jug"),nl,move(0,K,[(0,K)|Z]).
```

Output:

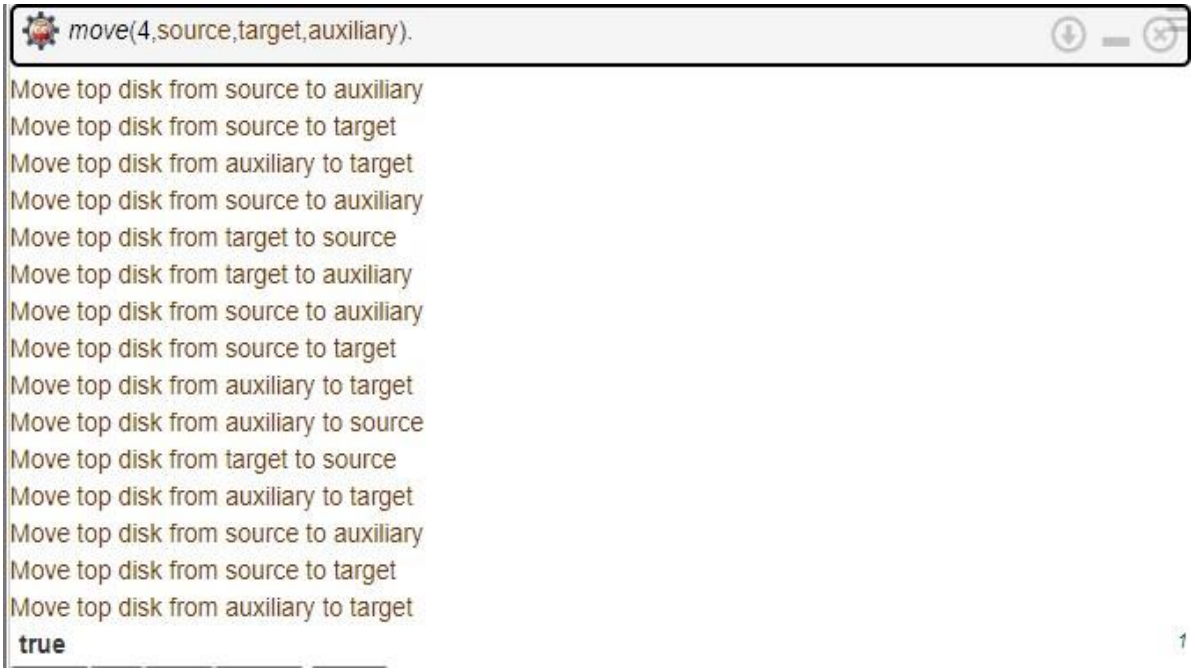


Q3. **Write a prolog program to solve tic-tac-toe Problem.**

```
move(1,X,Y,_) :-
  write('Move top disk from '), write(X), write(' to '), write(Y), nl.
move(N,X,Y,Z) :-
  N>1,
  M is N-1,
```

```
  move(M,X,Z,Y),
  move(1,X,Y,_),
  move(M,Z,Y,X).
```

Output:



Q4). Write a prolog program to solve 8- Puzzle Problem.

```
test(Plan):-
   write('Initial state:'),nl,
   Init= [at(tile4,1), at(tile3,2), at(tile8,3), at(empty,4), at(tile2,5), at(tile6,6), at(tile5,7), at(tile1,8), at(tile7,9)],
   write_sol(Init),
```

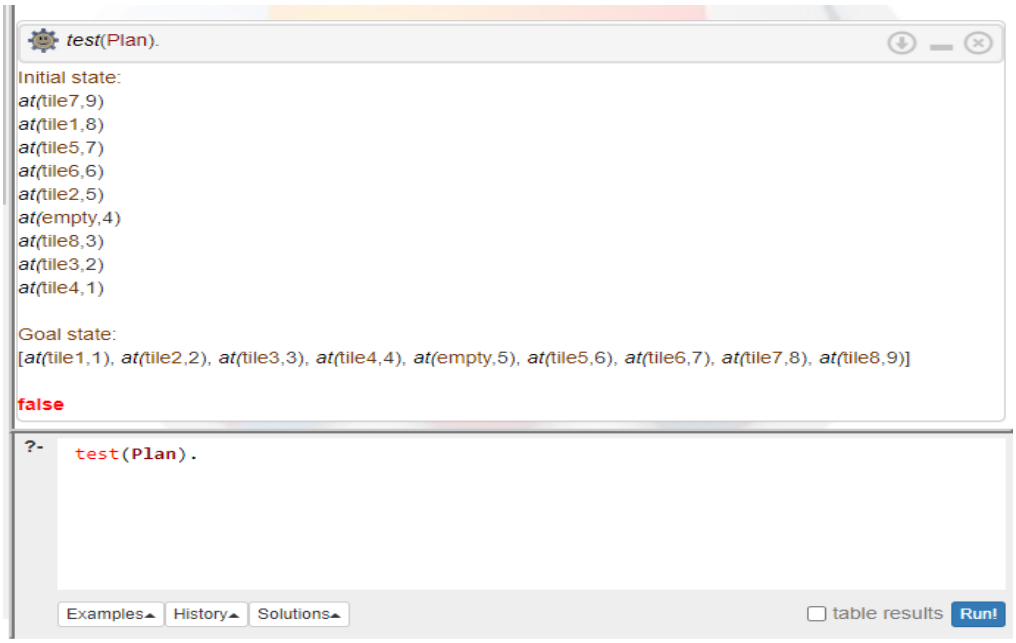
```
   Goal= [at(tile1,1), at(tile2,2), at(tile3,3), at(tile4,4), at(empty,5), at(tile5,6), at(tile6,7),
at(tile7,8), at(tile8,9)],
   nl,write('Goal state:'),nl,
   write(Goal),nl,nl,
   solve(Init,Goal,Plan).

solve(State, Goal, Plan):-
   solve(State, Goal, [], Plan).

%Determines whether Current and Destination tiles are a valid move.
is_movable(X1,Y1) :- (1 is X1 - Y1) ; (-1 is X1 - Y1) ; (3 is X1 - Y1) ; (-3 is X1 - Y1).

% This predicate produces the plan. Once the Goal list is a subset
% of the current State the plan is complete and it is written to
% the screen using write_sol/1.

solve(State, Goal, Plan, Plan):-
   is_subset(Goal, State), nl,
   write_sol(Plan).

solve(State, Goal, Sofar, Plan):-
```

```
   act(Action, Preconditions, Delete, Add),
   is_subset(Preconditions, State),
   \+ member(Action, Sofar),
   delete_list(Delete, State, Remainder),
   append(Add, Remainder, NewState),
   solve(NewState, Goal, [Action|Sofar], Plan).

% The problem has three operators.
% 1st arg = name
% 2nd arg = preconditions
% 3rd arg = delete list
% 4th arg = add list.

% Tile can move to new position only if the destination tile is empty & Manhattan distance
= 1
act(move(X,Y,Z),
   [at(X,Y), at(empty,Z), is_movable(Y,Z)],
   [at(X,Y), at(empty,Z)],
   [at(X,Z), at(empty,Y)]).

% Utility predicates.
```

```
% Check is first list is a subset of the second

is_subset([H|T], Set):-
   member(H, Set),
   is_subset(T, Set).
is_subset([], _).

% Remove all elements of 1st list from second to create third.

delete_list([H|T], Curstate, Newstate):-
   remove(H, Curstate, Remainder),
   delete_list(T, Remainder, Newstate).
delete_list([], Curstate, Curstate).

remove(X, [X|T], T).
remove(X, [H|T], [H|R]):-
   remove(X, T, R).

write_sol([]).
write_sol([H|T]):-
   write_sol(T),
```

```
    write(H), nl.

append([H|T], L1, [H|L2]):-
    append(T, L1, L2).
append([], L, L).

member(X, [X|_]).
member(X, [_|T]):-
    member(X, T).
```

Output:



Practical No. 2

Title:Introduction to Python Programming and python libraries

Aim: Understanding basics of python programming and python libraries like numpy, pandas and matplotlib

Introduction:

What is Python :-

Python is a popular programming language. It was created by Guido van Rossum, and released in 1991. It was designed with an emphasis on code readability, and its syntax allows programmers to express their concepts in fewer lines of code.

Python is a programming language that lets you work quickly and integrate systems more efficiently.

Python libraries for Machine Learning

Machine Learning, as the name suggests, is the science of programming a computer by which they are able to learn from different kinds of data. A more general definition given by Arthur Samuel is – "Machine Learning is the field of study that gives computers the ability to learn without being explicitly programmed." They are typically used to solve various types of life problems.

In the older days, people used to perform Machine Learning tasks by manually coding all the algorithms and mathematical and statistical formulas. This made the processing time consuming, tedious and inefficient. But in the modern days, it has become very much easy and efficient compared to the olden days with various python libraries, frameworks, and modules. Today, Python is one of the most popular programming languages for this task and it has replaced many languages in the industry, one of the reasons is its vast collection of libraries. Python libraries that used in Machine Learning are:

Numpy

NumPy is a very popular python library for large multi-dimensional array and matrix processing, with the help of a large collection of high-level mathematical functions. It is very useful for fundamental scientific computations in Machine Learning. It is particularly useful for linear algebra, Fourier transform, and random number capabilities. High-end libraries like TensorFlow use NumPy internally for manipulation of Tensors.

Pandas

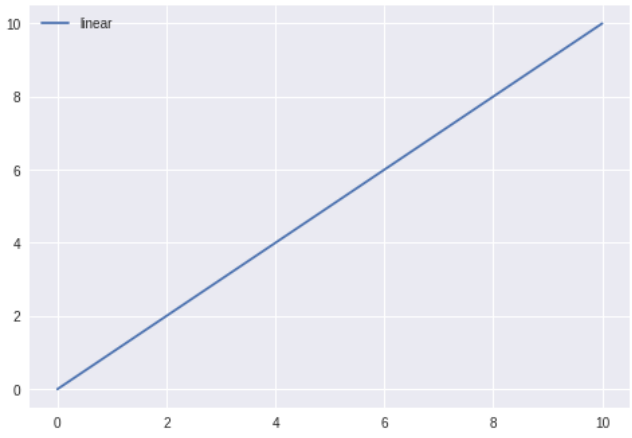
Pandas is a popular Python library for data analysis. It is not directly related to Machine Learning. As we know that the dataset must be prepared before training. In this case, Pandas comes handy as it was developed specifically for data extraction and preparation. It provides high-level data structures and a wide variety of tools for data analysis. It provides many inbuilt methods for grouping, combining and filtering data.

Matplotlib

Matplotlib is a very popular Python library for data visualization. Like Pandas, it is not directly related to Machine Learning. It particularly comes in handy when a programmer wants to visualize the patterns in the data. It is a 2D plotting library used for creating 2D graphs and plots. A module named pyplot makes it easy for programmers for plotting as it provides features to control line styles, font properties, formatting axes, etc. It provides various kinds of graphs and plots for data visualization, viz., histogram, error charts, bar charts, etc,

Example : -

```
import matplotlib.pyplot as plt

import numpy as np

x = np.linspace(0, 10, 100)

plt.plot(x, x, label ='linear')

plt.legend()

plt.show()
```



Exercise -

1. **Create Pandas Dataframe to enter employee database which includes Sr No, Name, Mobile No, City**
2. **Use Iris Dataset from Github and perform all basic operations on Iris Dataset**
3. **Use matplotlib to plot bar chart using dictionary**
4. **Use matplotlib to scatter graph and histogram**

Implementation:

Program:

1. Create Pandas Dataframe to enter employee database which includes Sr No, Name, Mobile No, City

```
[2] import pandas as pd
    d1=pd.DataFrame({"Sr_No" :[1,2,3,4],
                     "Name" :["Rahul","Akansha","Sayali","Shuklesh"],
                     "Phone_No" :[7231345678,1234567892,1234234567,8912674532],
                     "City" :["Ratnagiri","Mumbai","Pune","Satara"]})
    d1
```

	Sr_No	Name	Phone_No	City
0	1	Rahul	7231345678	Ratnagiri
1	2	Akansha	1234567892	Mumbai
2	3	Sayali	1234234567	Pune
3	4	Shuklesh	8912674532	Satara

2. Use Iris Dataset from Github and perform all basic operations on Iris Dataset

```
import pandas as pd

# download iris data and read it into a dataframe
url = 'http://archive.ics.uci.edu/ml/machine-learning-databases/iris/iris.data'
df = pd.read_csv(url, names=['sepal_length', 'sepal_width', 'petal_length', 'petal_width', 'class'])
df.head()
```

	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

-

```
df.tail()
```

	sepal_length	sepal_width	petal_length	petal_width	class
145	6.7	3.0	5.2	2.3	Iris-virginica
146	6.3	2.5	5.0	1.9	Iris-virginica
147	6.5	3.0	5.2	2.0	Iris-virginica
148	6.2	3.4	5.4	2.3	Iris-virginica
149	5.9	3.0	5.1	1.8	Iris-virginica

```
[ ]  df.describe()
```

	sepal_length	sepal_width	petal_length	petal_width
count	150.000000	150.000000	150.000000	150.000000
mean	5.843333	3.054000	3.758667	1.198667
std	0.828066	0.433594	1.764420	0.763161
min	4.300000	2.000000	1.000000	0.100000
25%	5.100000	2.800000	1.600000	0.300000
50%	5.800000	3.000000	4.350000	1.300000
75%	6.400000	3.300000	5.100000	1.800000
max	7.900000	4.400000	6.900000	2.500000

```
df.shape
```

```
(150, 5)
```

```
df.drop("class",axis=1)
```

	sepal_length	sepal_width	petal_length	petal_width
0	5.1	3.5	1.4	0.2
1	4.9	3.0	1.4	0.2
2	4.7	3.2	1.3	0.2
3	4.6	3.1	1.5	0.2
4	5.0	3.6	1.4	0.2
...	...	...	...	...
145	6.7	3.0	5.2	2.3
146	6.3	2.5	5.0	1.9
147	6.5	3.0	5.2	2.0
148	6.2	3.4	5.4	2.3

```
df.iloc[2:3]
```

	sepal_length	sepal_width	petal_length	petal_width	class
2	4.7	3.2	1.3	0.2	Iris-setosa

```
df.loc[0:3,"petal_length"]
```

```
0    1.4
1    1.4
2    1.3
3    1.5
Name: petal_length, dtype: float64
```

```
df.max()
```

```
sepal_length               7.9
sepal_width                4.4
petal_length               6.9
petal_width                2.5
class           Iris-virginica
dtype: object
```

```
def half(x):
  return x/2

df["sepal_length"].apply(half)
```

```
0      2.55
1      2.45
2      2.35
3      2.30
4      2.50
       ...
145    3.35
146    3.15
147    3.25
148    3.10
149    2.95
Name: sepal_length, Length: 150, dtype: float64
```

```
df["class"].value_counts()
```

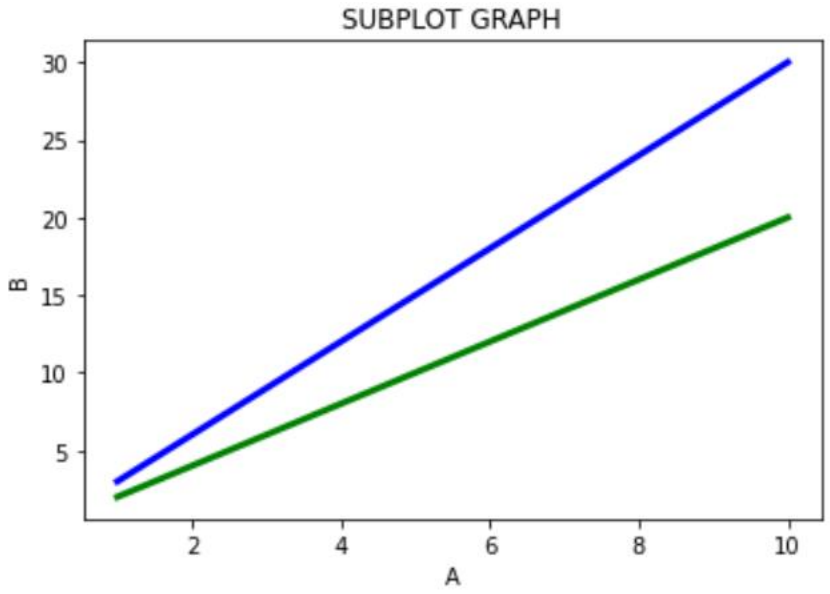
```
Iris-setosa        50
Iris-versicolor    50
Iris-virginica     50
Name: class, dtype: int64
```

```
df.sort_values(by="sepal_length")
```

	sepal_length	sepal_width	petal_length	petal_width	class
13	4.3	3.0	1.1	0.1	Iris-setosa
42	4.4	3.2	1.3	0.2	Iris-setosa
38	4.4	3.0	1.3	0.2	Iris-setosa
8	4.4	2.9	1.4	0.2	Iris-setosa
41	4.5	2.3	1.3	0.3	Iris-setosa
...	...	...	...	...	...
122	7.7	2.8	6.7	2.0	Iris-virginica
118	7.7	2.6	6.9	2.3	Iris-virginica
117	7.7	3.8	6.7	2.2	Iris-virginica

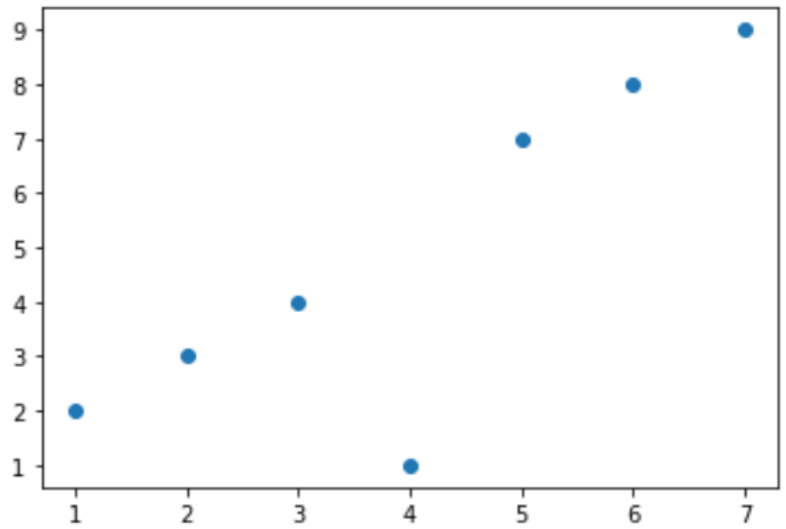
3. Use matplotlib to plot bar chart using dictionary

```
] x=np.array([1,2,3,4,5,6,7,8,9,10])
  y=x*2
  y1=x*3
  plt.plot(x,y,color='green',linewidth=3)
  plt.plot(x,y1,color='blue',linewidth=3)
  plt.title("SUBPLOT GRAPH")
  plt.xlabel("A")
  plt.ylabel("B")
  plt.show()
```

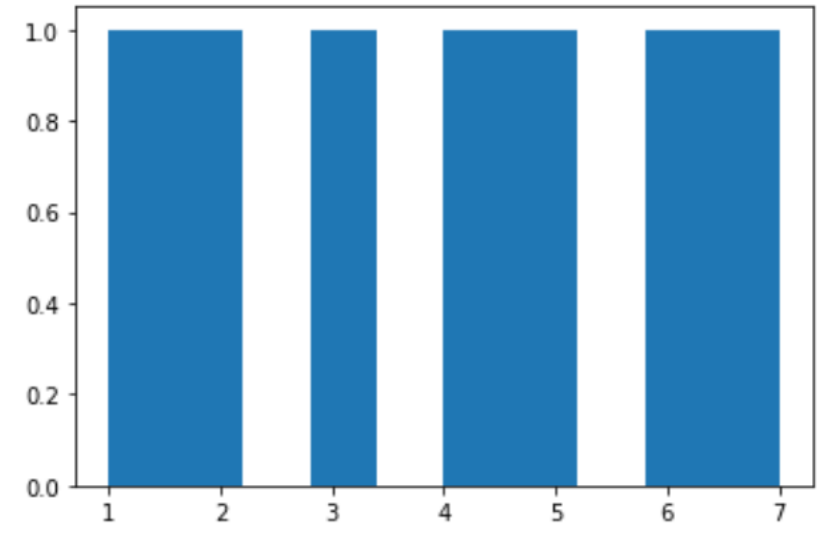


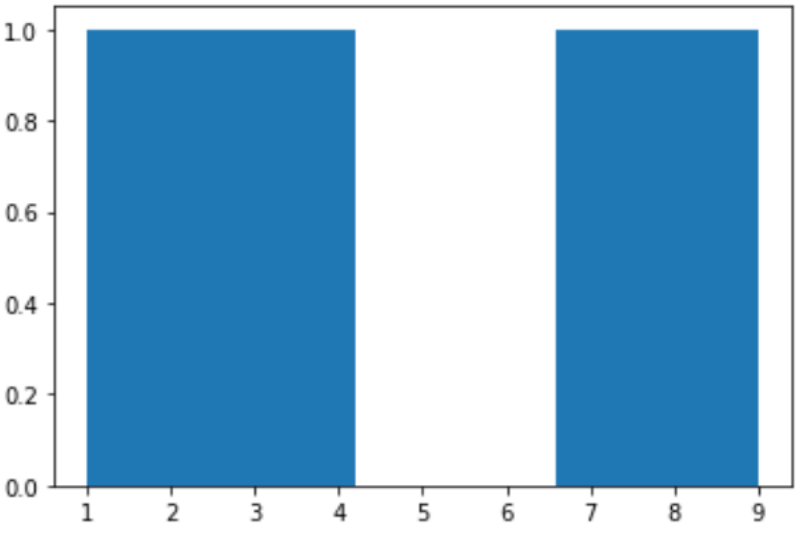
4. Use matplotlib to scatter graph and histogram.

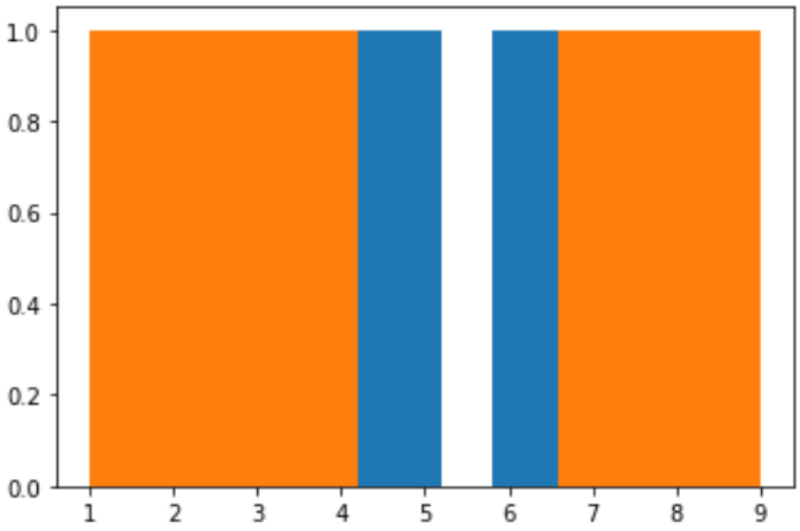
```
x=[1,2,3,4,5,6,7]
y=[2,3,4,1,7,8,9]
plt.scatter(x,y)
plt.show()
```



```
plt.hist(x)
plt.show()
plt.hist(y)
plt.show()
plt.hist(x)
plt.hist(y)
plt.show()
```






1.0
0.8
0.6
0.4
0.2
0.0
1
2
3
4
5
6
7
8
9

Practical No. 3

Title: Implementation of Linear Regression, Logistic regression, KNN- classification.

Aim: Understanding basics of Linear Regression, Logistic regression and KNN-classification.

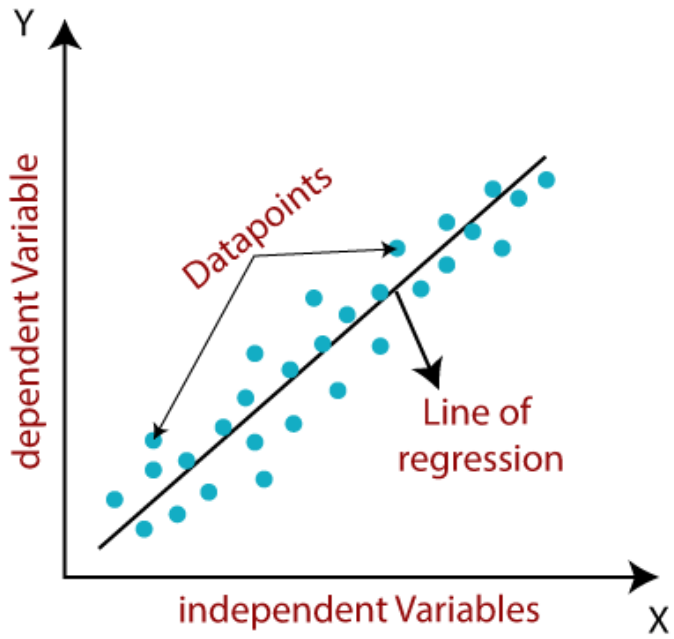
Introduction:

Linear Regression

Linear Regression is a machine learning algorithm based on **supervised learning**. It performs a **regression task**. Regression models a target prediction value based on independent variables. It is mostly used for finding out the relationship between variables and forecasting. Different regression models differ based on – the

relationshibetweedependent and independent variables they are considering, and the number of independent variables getting used.

Linear regression algorithm shows a linear relationship between a dependent (y) and one or more independent (y) variables, hence called linear regression. Since linear regression shows the linear relationship, which means it finds how the value of the dependent variable is changing according to the value of the independent variable.



The linear regression model provides a sloped straight line representing the relationship between the variables.

Mathematically, we can represent a linear regression as:

y= mx + b

Here,

Y= Dependent Variable (Target Variable)

X= Independent Variable (predictor Variable)

b= intercept of the line (Gives an additional degree of freedom)

m= Linear regression coefficient (scale factor to each input value).

The values for x and y variables are training datasets for Linear Regression model representation.

Finding the best fit line:

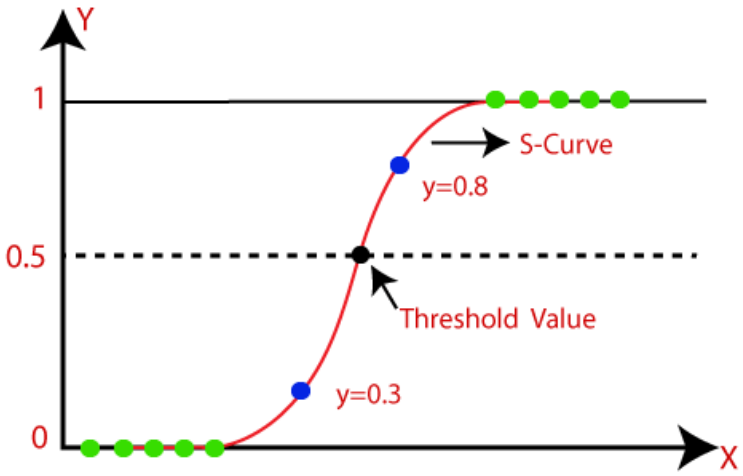
When working with linear regression, our main goal is to find the best fit line that means the error between predicted values and actual values should be minimized. The best fit line will have the least error.

The different values for weights or the coefficient of lines (a_0, a_1) gives a different line of regression, so we need to calculate the best values for a_0 and a_1 to find the best fit line, so to calculate this we use cost function.

Logistic Regression in Machine Learning

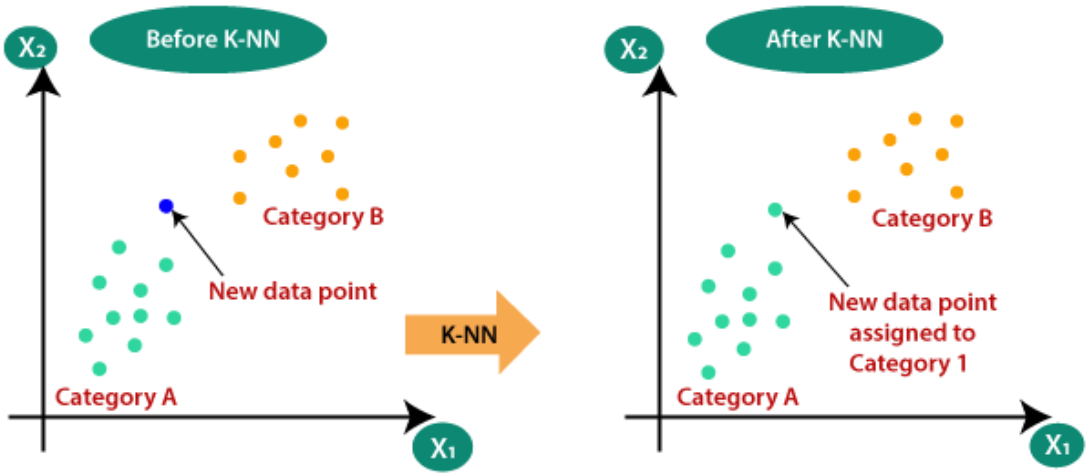
- Logistic regression is one of the most popular Machine Learning algorithms, which comes under the Supervised Learning technique. It is used for predicting the categorical dependent variable using a given set of independent variables.
- Logistic regression predicts the output of a categorical dependent variable. Therefore the outcome must be a categorical or discrete value. It can be either Yes or No, 0 or 1, true or False, etc. but instead of giving the exact value as 0 and 1, **it gives the probabilistic values which lie between 0 and 1**.
- Logistic Regression is much similar to the Linear Regression except that how they are used. Linear Regression is used for solving Regression problems, whereas **Logistic regression is used for solving the classification problems**.
- In Logistic regression, instead of fitting a regression line, we fit an "S" shaped logistic function, which predicts two maximum values (0 or 1).
- The curve from the logistic function indicates the likelihood of something such as whether the cells are cancerous or not, a mouse is obese or not based on its weight, etc.
- Logistic Regression is a significant machine learning algorithm because it has the ability to provide probabilities and classify new data using continuous and discrete datasets.

- Logistic Regression can be used to classify the observations using different types of data and can easily determine the most effective variables used for the classification. The below image is showing the logistic function:



K-Nearest Neighbor(KNN) Algorithm for Machine Learning

Suppose there are two categories, i.e., Category A and Category B, and we have a new data point x1, so this data point will lie in which of these categories. To solve this type of problem, we need a K-NN algorithm. With the help of K-NN, we can easily identify the category or class of a particular dataset. Consider the below diagram:

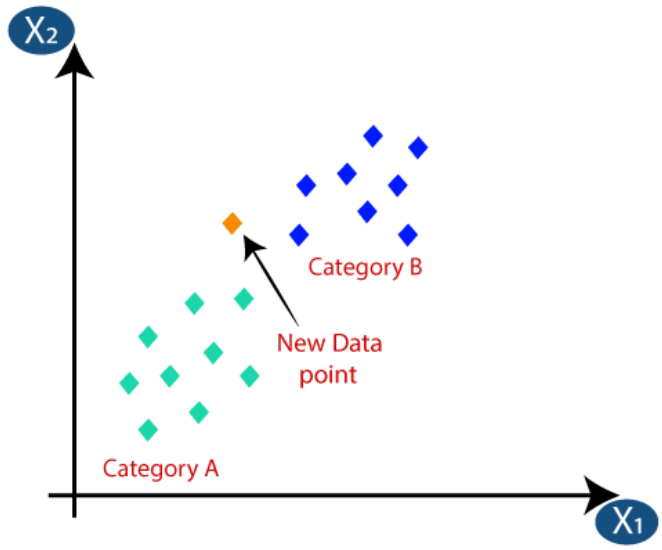


How does K-NN work?

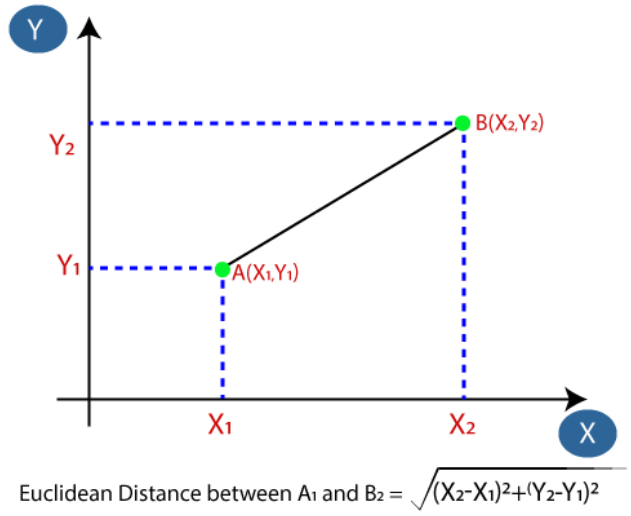
The K-NN working can be explained on the basis of the below algorithm:

- **Step-1:** Select the number K of the neighbors
- **Step-2:** Calculate the Euclidean distance of **K number of neighbors**
- **Step-3:** Take the K nearest neighbors as per the calculated Euclidean distance.
- **Step-4:** Among these k neighbors, count the number of the data points in each category.
- **Step-5:** Assign the new data points to that category for which the number of the neighbor is maximum.
- **Step-6:** Our model is ready.

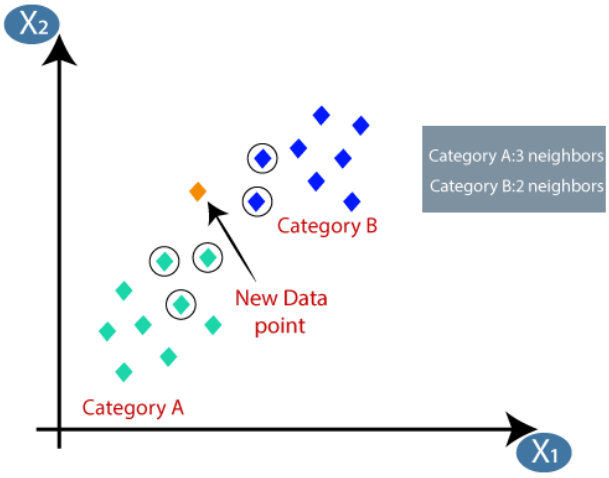
Suppose we have a new data point and we need to put it in the required category. Consider the below image:



- Firstly, we will choose the number of neighbors, so we will choose the k=5.
- Next, we will calculate the **Euclidean distance** between the data points. The Euclidean distance is the distance between two points, which we have already studied in geometry. It can be calculated as:



- By calculating the Euclidean distance we got the nearest neighbors, as three nearest neighbors in category A and two nearest neighbors in category B. Consider the below image:



- As we can see the 3 nearest neighbors are from category A, hence this new data point must belong to category A.

Exercise -

1. Predict Canada's per capita income in 2020. There is an exercise folder here on github at the same level as this notebook, download that and you will find the canada_per_capita_income.csv file. Using this build a regression model and predict the per capita income of canadian citizens in year 2020

Implementation:

```
url='https://raw.githubusercontent.com/codebasics/py/master/ML/1_linear_reg/Exercise/canada_per_capita_income.csv'

ar1=pd.read_csv(url)
ar1
```

	year	per capita income (US$)
0	1970	3399.299037
1	1971	3768.297935
2	1972	4251.175484
3	1973	4804.463248
4	1974	5576.514583
5	1975	5998.144346
6	1976	7062.131392
7	1977	7100.126170
8	1978	7247.967035
9	1979	7602.912681
10	1980	8355.968120
11	1981	9434.390652
12	1982	9619.438377
13	1983	10416.536590

```
rg=linear_model.LinearRegression()
ar1 = ar1.rename(columns={'per capita income (US$)': 'per_capita_income'})
rg.fit(ar1[['year']],ar1.per_capita_income)
```

```
LinearRegression()
```

```
rg.predict([[2020]])
```

```
/usr/local/lib/python3.7/dist-
  "X does not have valid featu
array([41288.69409442])
```

2. Logistic regression

Implementation:

```
#logitsic regression- Binary classification

import pandas as pd
import numpy as np
from sklearn import linear_model
import matplotlib.pyplot as plt
```

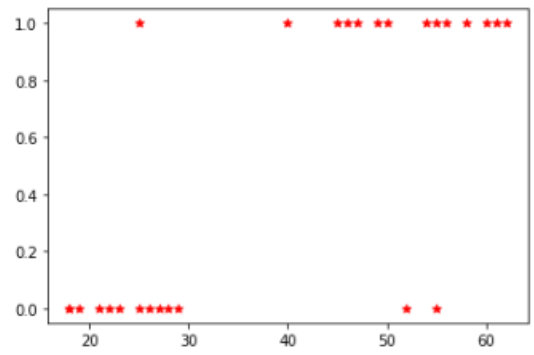
```
url='http://raw.githubusercontent.com/WamanParulekar/AIML/main/lic.csv'
lic = pd.read_csv(url)
lic
```

	age	lic_member
0	22	0
1	25	0
2	47	1
3	52	0
4	28	0
5	27	0
6	29	0
7	49	1
8	55	1
9	25	1
10	58	1
11	19	0
12	46	1
13	56	1
14	55	0

```
plt.scatter(lic.age,lic.lic_member,marker="*",color='r')
```

```
1 plt.scatter(lic.age,lic.lic_member,marker="*",color='r')
```

```
<matplotlib.collections.PathCollection at 0x7ff88bf72ed0>
```




```
from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test = train_test_split(lic[['age']],lic.lic_member, train_size=0.8)
```

```
len(x_test)
```

```
6
```

```
len(y_test)
```

```
6
```

```
from sklearn.linear_model import LogisticRegression
lr = LogisticRegression()
lr.fit(x_train , y_train)
```

```
LogisticRegression()
```

```
lr.predict(x_test)
```

```
array([1, 1, 1, 0, 0, 0])
```

```
lic
```

	age	lic_member
0	22	0
1	25	0
2	47	1
3	52	0
4	28	0
5	27	0
6	29	0
7	49	1
8	55	1
9	25	1
10	58	1
11	19	0
12	46	1
13	56	1
14	55	0
15	60	1

```
lr.score(x_test,y_test)
 0.8333333333333334

lr.predict([[30]])


 /usr/local/lib/python3.7/dist-packages/sklearn/bas
   "X does not have valid feature names, but"
 array([0])
```

3. Using K nearest neighbors classification predict type of flower given 'sepal_length', 'sepal_width', 'petal_length', 'petal_width' = 4.8,3.0,1.5,0.3

Implementation:

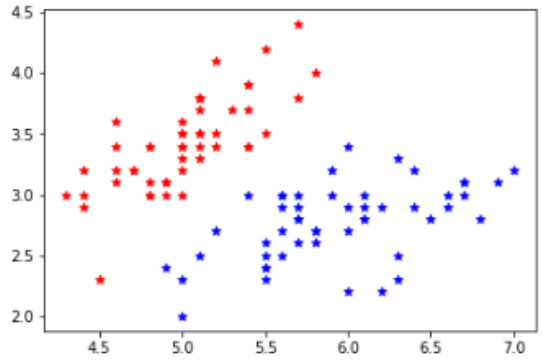
```
import pandas as pd
import numpy as np
from sklearn import linear_model
import matplotlib.pyplot as plt

url = 'http://archive.ics.uci.edu/ml/machine-learning-
databases/iris/iris.data'
iris = pd.read_csv(url, names=['sepal_length', 'sepal_width', 'petal
_length', 'petal_width', 'class'])

iris.head()
```

	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

```
iris1=iris[:50]
iris2=iris[50:100]
iris3=iris[100:]
```

```
plt.scatter(iris1.sepal_length,iris1.sepal_width,marker="*",color='r
')
plt.scatter(iris2.sepal_length,iris2.sepal_width,marker="*",color='b
')
```

```
<matplotlib.collections.PathCollection at 0x7f9183da4550>
```




```
from sklearn.model_selection import train_test_split
x=iris.drop('class',axis=1)
y=iris[['class']]
x_train,x_test,y_train,y_test=train_test_split(x,y,train_size=0.8)
from sklearn.neighbors import KNeighborsClassifier
knn=KNeighborsClassifier()
knn.fit(x_train,y_train)
```

```
/usr/local/lib/python3.7/dis
  return self._fit(X, y)
KNeighborsClassifier()
```

```
knn.predict([[4.8,3.0,1.5,0.3]])
```

```
/usr/local/lib/python3.7/dist-packages/sklearn/b
  "X does not have valid feature names, but"
array(['Iris-setosa'], dtype=object)
```

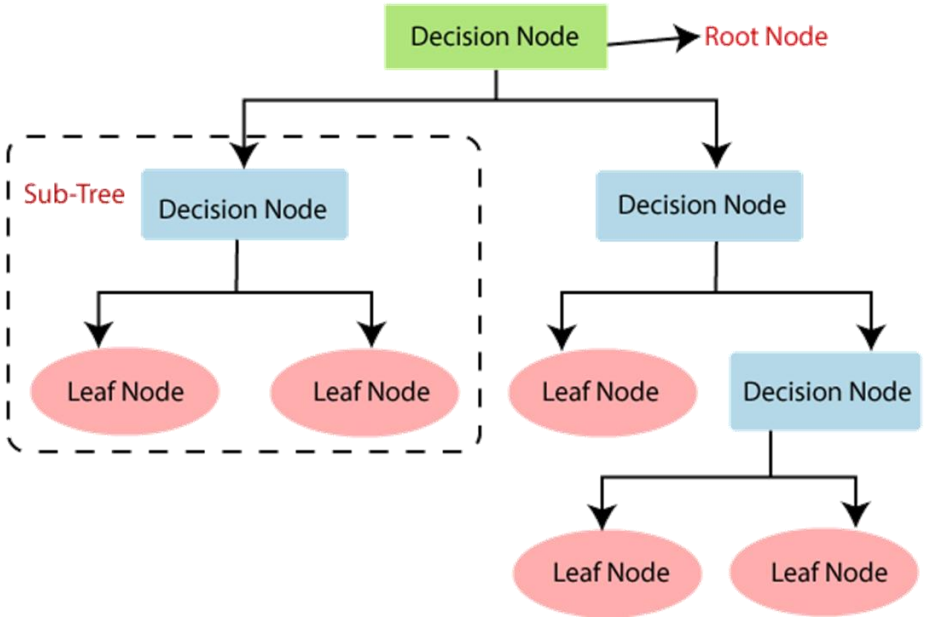
Practical No. 4

Title: Implementation of Bagging Algorithm: Decision Tree

Aim: Understanding basics of Bagging Algorithm: Decision Tree

Introduction:

Decision Tree is a Supervised learning technique that can be used for both classification and Regression problems, but mostly it is preferred for solving Classification problems. where internal nodes represent the features of a dataset, branches represent the decision rules and each leaf node represents the outcome.



In a Decision tree, there are two nodes, which are the Decision Node and Leaf Node. Decision nodes are used to make any decision and have multiple branches, whereas Leaf nodes are the output of those decisions and do not contain any further branches.

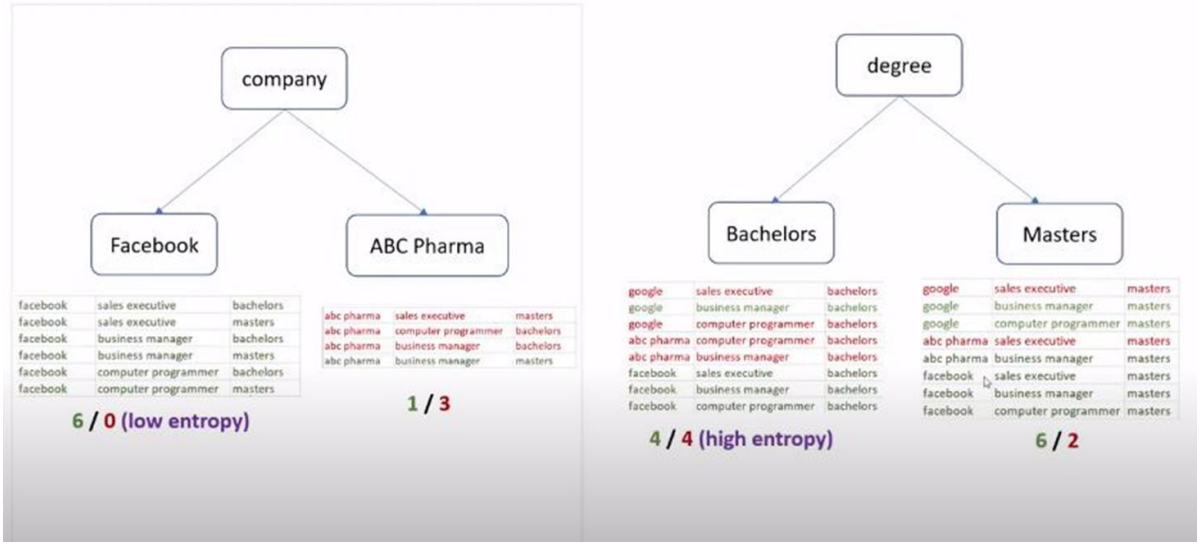
Step-1: Begin the tree with the root node, says S, which contains the complete dataset.
Step-2: Find the best attribute in the dataset using Attribute Selection Measure (ASM).
Step-3: Divide the S into subsets that contain possible values for the best attributes.
Step-4: Generate the decision tree node, which contains the best attribute.
Step-5: Recursively make new decision trees using the subsets of the dataset created in step -3. Continue this process until a stage is reached where you cannot further classify the nodes and called the final node as a leaf node.

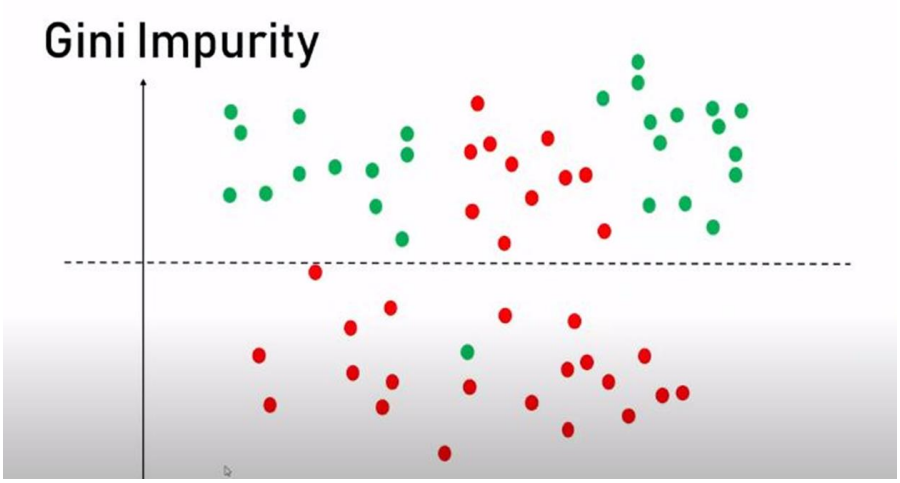
Attribute Selection Measures

While implementing a Decision tree, the main issue arises that how to select the best attribute for the root node and for sub-nodes. So, to solve such problems there is a technique which is called as **Attribute selection measure or ASM.**

By this measurement, we can easily select the best attribute for the nodes of the tree.
There are two popular techniques for ASM, which are:

Information Gain
Gini Index





Exercise -

1. Implement given dataset to predict salary range of computer programmer from google having bachelor degree / master's degree

Implementation:

```
import pandas as pd
url='https://raw.githubusercontent.com/codebasics/py/master/ML/9_decision_tree/salaries.csv'
df=pd.read_csv(url)
df
```

	company	job	degree	salary_more_then_100k
0	google	sales executive	bachelors	0
1	google	sales executive	masters	0
2	google	business manager	bachelors	1
3	google	business manager	masters	1
4	google	computer programmer	bachelors	0
5	google	computer programmer	masters	1
6	abc pharma	sales executive	masters	0
7	abc pharma	computer programmer	bachelors	0
8	abc pharma	business manager	bachelors	0
9	abc pharma	business manager	masters	1
10	facebook	sales executive	bachelors	1
11	facebook	sales executive	masters	1
12	facebook	business manager	bachelors	1
13	facebook	business manager	masters	1
14	facebook	computer programmer	bachelors	1
15	facebook	computer programmer	masters	1

```
inp=df.drop(['salary_more_then_100k'], axis="columns")
inp
```

	company	job	degree
0	google	sales executive	bachelors
1	google	sales executive	masters
2	google	business manager	bachelors
3	google	business manager	masters
4	google	computer programmer	bachelors
5	google	computer programmer	masters
6	abc pharma	sales executive	masters
7	abc pharma	computer programmer	bachelors
8	abc pharma	business manager	bachelors
9	abc pharma	business manager	masters
10	facebook	sales executive	bachelors
11	facebook	sales executive	masters
12	facebook	business manager	bachelors
13	facebook	business manager	masters

```
trgt=df['salary_more_then_100k']
trgt
```

```
0     0
1     0
2     1
3     1
4     0
5     1
6     0
7     0
8     0
9     1
10    1
11    1
12    1
13    1
14    1
15    1
Name: salary_more_then_100k, dtype: int64
```

```
from sklearn.preprocessing import LabelEncoder
lbl_company=LabelEncoder()
lbl_job=LabelEncoder()
lbl_degree=LabelEncoder()
```

```
inp['company_new'] = lbl_company.fit_transform(inp['company'])
inp['job_new'] = lbl_company.fit_transform(inp['job'])
inp['degree_new'] = lbl_company.fit_transform(inp['degree'])
inp
```

	company	job	degree	company_new	job_new	degree_new
0	google	sales executive	bachelors	2	2	0
1	google	sales executive	masters	2	2	1
2	google	business manager	bachelors	2	0	0
3	google	business manager	masters	2	0	1
4	google	computer programmer	bachelors	2	1	0
5	google	computer programmer	masters	2	1	1
6	abc pharma	sales executive	masters	0	2	1
7	abc pharma	computer programmer	bachelors	0	1	0
8	abc pharma	business manager	bachelors	0	0	0
9	abc pharma	business manager	masters	0	0	1
10	facebook	sales executive	bachelors	1	2	0
11	facebook	sales executive	masters	1	2	1
12	facebook	business manager	bachelors	1	0	0
13	facebook	business manager	masters	1	0	1
14	facebook	computer programmer	bachelors	1	1	0

```
inp_new = inp.drop(['company', 'job', 'degree'], axis='columns')

inp_new
```

	company_new	job_new	degree_new
0	2	2	0
1	2	2	1
2	2	0	0
3	2	0	1
4	2	1	0
5	2	1	1
6	0	2	1
7	0	1	0
8	0	0	0
9	0	0	1
10	1	2	0
11	1	2	1
12	1	0	0

```
from sklearn import tree
tree_model = tree.DecisionTreeClassifier()
tree_model.fit(inp_new, trgt)
```

```
DecisionTreeClassifier()
```

```
tree_model.predict([[2,1,0]])
```

```
/usr/local/lib/python3.7/dist-packages/sklearn/base.py:451:
  "X does not have valid feature names, but"
array([0])
```

```
tree_model.predict([[2,1,1]])
```

```
/usr/local/lib/python3.7/dist-packages/skl
  "X does not have valid feature names, bu
array([1])
```

Practical No. 5

Title: Implementation of Bagging Algorithm: Random Forest

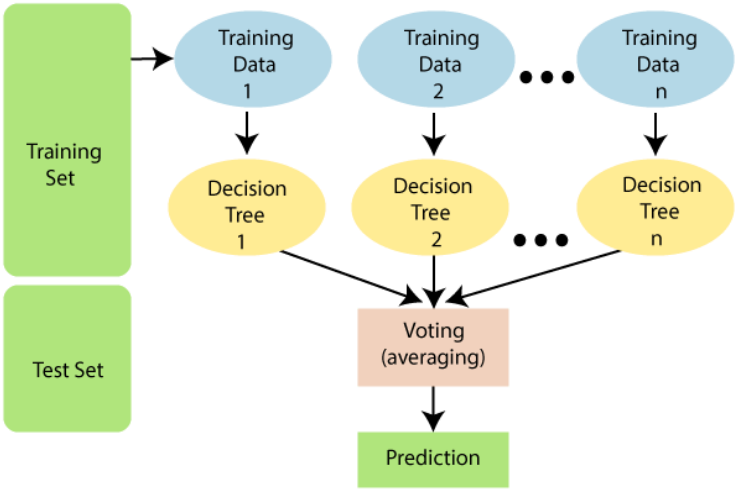
Aim: Understanding basics of Bagging Algorithm: Random Forest

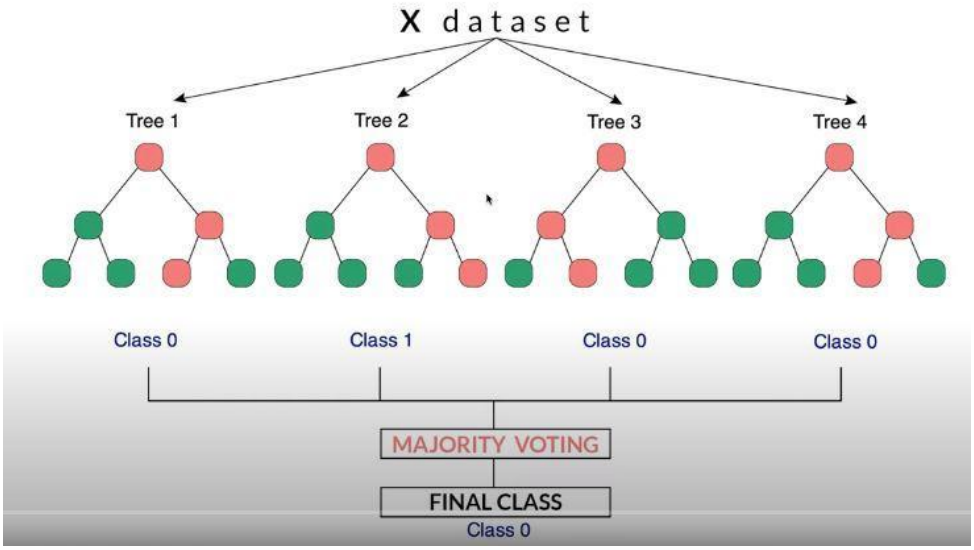
Introduction: Random Forest Algorithm

Random Forest is a popular machine learning algorithm that belongs to the supervised learning technique. It can be used for both Classification and Regression problems in ML. It is based on the concept of ensemble learning, which is a process of combining multiple classifiers to solve a complex problem and to improve the performance of the model.

As the name suggests, "Random Forest is a classifier that contains a number of decision trees on various subsets of the given dataset and takes the average to improve the predictive accuracy of that dataset." Instead of relying on one decision tree, the random forest takes the prediction from each tree and based on the majority votes of predictions, and it predicts the final output.

The greater number of trees in the forest leads to higher accuracy and prevents the problem of overfitting. The below diagram explains the working of the Random Forest algorithm:





Exercise -

1. Using iris dataset properties like sepal length, sepal width, petal length and petal width, predict the class of flower.

Implementation:

Program:

1.Using iris dataset properties like sepal length, sepal width, petal length and petal width, predict the class of flower.

```
from sklearn import tree
tree_model = tree.DecisionTreeRegressor()
tree_model.fit(inp_new,trgt)
```

```
DecisionTreeRegressor()
```

```
tree_model.predict([[2,1,1]])
```

```
/usr/local/lib/python3.7/dist-packages/sklearn/base.py:451: UserWarning: X does not have valid feature names, but DecisionTreeRegressor was fitt
  "X does not have valid feature names, but"
array([1.])
```

```
import pandas as pd
import numpy as np
url = 'http://archive.ics.uci.edu/ml/machine-learning-databases/iris/iris.data'
iris = pd.read_csv(url, names=['sepal_length', 'sepal_width', 'petal_length', 'petal_width', 'cla
ss'])
```

```
iris.head()
```

	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

```
from sklearn.model_selection import train_test_split
X_train,X_test, y_train, y_test = train_test_split(iris.drop(['class'],axis='columns'),iris[['class'
]],train_size=0.8)
```

```
from sklearn.ensemble import RandomForestClassifier
RFC = RandomForestClassifier()
RFC.fit(X_train,y_train)
```

```
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:3: DataConversionWarning: A column-vector y was passed when a 1d array was expected
  This is separate from the ipykernel package so we can avoid doing imports until
RandomForestClassifier()
```

RFC.predict([[4.8,3.0,1.5,0.3]])

```
/usr/local/lib/python3.7/dist-packages/sklearn/base.py:451: UserWarning: X does not have valid feature names, but RandomForestClassifier was fit
  "X does not have valid feature names, but"
array(['Iris-setosa'], dtype=object)
```

Practical No. 6

Title: Implementation of Support Vector Machine or SVM

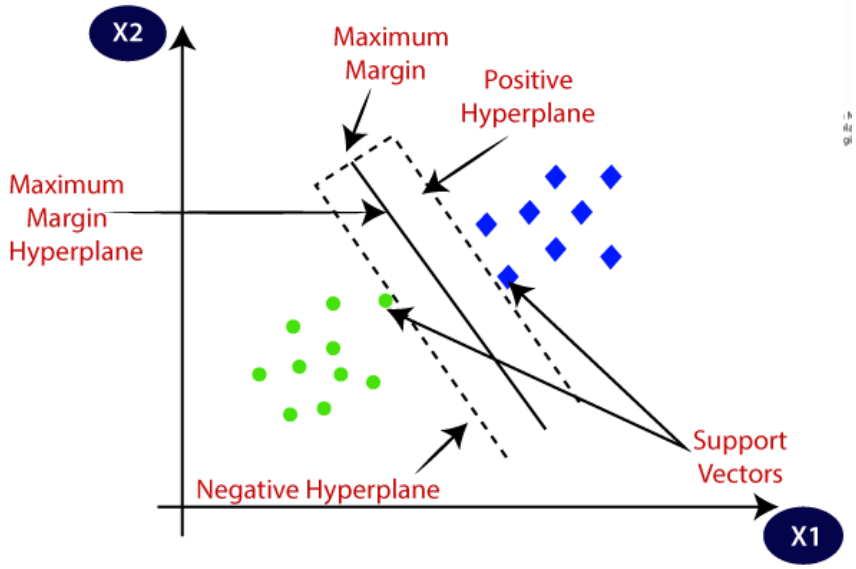
Aim: Understanding basics of Bagging Algorithm: Random Forest

Introduction:

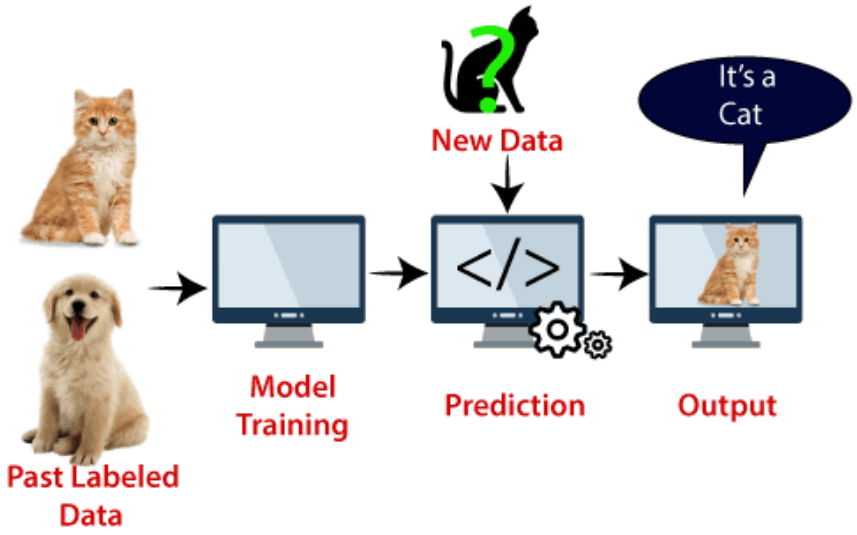
Support Vector Machine or SVM is one of the most popular Supervised Learning algorithms, which is used for Classification as well as Regression problems. However, primarily, it is used for Classification problems in Machine Learning.

The goal of the SVM algorithm is to create the best line or decision boundary that can segregate n-dimensional space into classes so that we can easily put the new data point in the correct category in the future. This best decision boundary is called a hyperplane.

SVM chooses the extreme points/vectors that help in creating the hyperplane. These extreme cases are called support vectors, and hence the algorithm is termed as Support Vector Machine. Consider the below diagram in which there are two different categories that are classified using a decision boundary or hyperplane:



Example: SVM can be understood with the example that we have used in the KNN classifier. Suppose we see a strange cat that also has some features of dogs, so if we want a model that can accurately identify whether it is a cat or dog, so such a model can be created by using the SVM algorithm. We will first train our model with lots of images of cats and dogs so that it can learn about different features of cats and dogs, and then we test it with this strange creature. So as support vector creates a decision boundary between these two data (cat and dog) and choose extreme cases (support vectors), it will see the extreme case of cat and dog. On the basis of the support vectors, it will classify it as a cat. Consider the below diagram:



SVM algorithm can be used for Face detection, image classification, text categorization, etc.

Types of SVM

SVM can be of two types:

- Linear SVM: Linear SVM is used for linearly separable data, which means if a dataset can be classified into two classes by using a single straight line, then such data is termed as linearly separable data, and classifier is used called as Linear SVM classifier.

- Non-linear SVM: Non-Linear SVM is used for non-linearly separated data, which means if a dataset cannot be classified by using a straight line, then such data is termed as non-linear data and classifier used is called as Non-linear SVM classifier.

Hyperplane and Support Vectors in the SVM algorithm:

Hyperplane: There can be multiple lines/decision boundaries to segregate the classes in n-dimensional space, but we need to find out the best decision boundary that helps to classify the data points. This best boundary is known as the hyperplane of SVM.

The dimensions of the hyperplane depend on the features present in the dataset, which means if there are 2 features (as shown in image), then hyperplane will be a straight line. And if there are 3 features, then hyperplane will be a 2-dimension plane.

We always create a hyperplane that has a maximum margin, which means the maximum distance between the data points.

Support Vectors:

The data points or vectors that are the closest to the hyperplane and which affect the position of the hyperplane are termed as Support Vector. Since these vectors support the hyperplane, hence called a Support vector.

Exercise -

Support Vector Machine Tutorial Using Python Sklearn

Implementation:

Program:

```
import pandas as pd

import numpy as np
```

```
url = 'http://archive.ics.uci.edu/ml/machine-learning-databases/iris/iris.data'

iris = pd.read_csv(url, names=['sepal_length', 'sepal_width', 'petal_length', 'petal_width', 'class'])
```

```
iris.head()
```

	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

```
from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(iris.drop(['class'],axis='columns'),iris[['class']],train_size=0.8)

len(X_train)

from sklearn.svm import SVC

model = SVC()

model.fit(X_train, y_train)
```

Output:

```
model.predict([[4.8,3.0,1.5,0.3]])

array(['Iris-setosa'], dtype=object)
```

Practical No. 7

Title: Implementation of K-Means clustering algorithm.

Aim: Understanding basics of K-Means clustering algorithm.

Introduction:

K-Means Clustering Algorithm

K-Means Clustering is an unsupervised learning algorithm that is used to solve the clustering problems in machine learning or data science. In this topic, we will learn what is K-means clustering algorithm, how the algorithm works, along with the Python implementation of k-means clustering.

What is K-Means Algorithm?

K-Means Clustering is an Unsupervised Learning algorithm, which groups the unlabeled dataset into different clusters. Here K defines the number of pre-defined clusters that need to be created in the process, as if K=2, there will be two clusters, and for K=3, there will be three clusters, and so on.

It allows us to cluster the data into different groups and a convenient way to discover the categories of groups in the unlabeled dataset on its own without the need for any training.

It is a centroid-based algorithm, where each cluster is associated with a centroid. The main aim of this algorithm is to minimize the sum of distances between the data point and their corresponding clusters.

The algorithm takes the unlabeled dataset as input, divides the dataset into k-number of clusters, and repeats the process until it does not find the best clusters. The value of k should be predetermined in this algorithm.

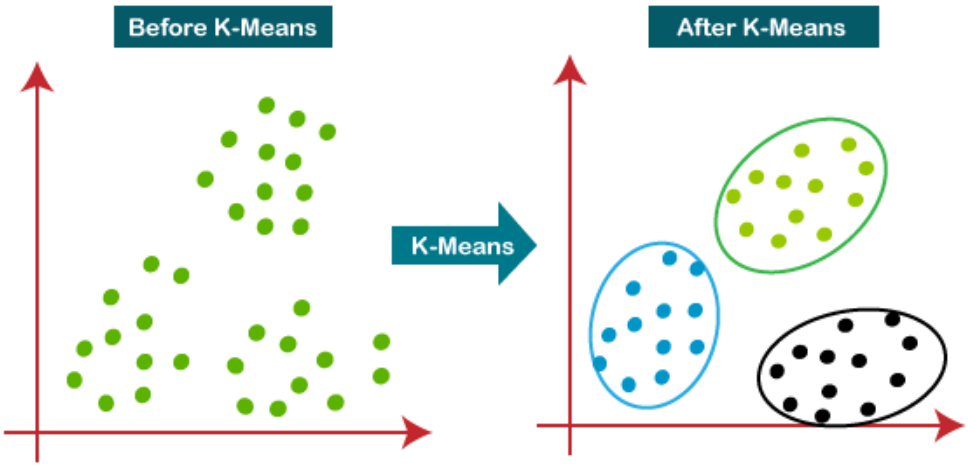
The k-means clustering algorithm mainly performs two tasks:

- Determines the best value for K center points or centroids by an iterative process.

- Assigns each data point to its closest k-center. Those data points which are near to the particular k-center, create a cluster.

Hence each cluster has datapoints with some commonalities, and it is away from other clusters.

The below diagram explains the working of the K-means Clustering Algorithm:



How does the K-Means Algorithm Work?

The working of the K-Means algorithm is explained in the below steps:

Step-1: Select the number K to decide the number of clusters.

Step-2: Select random K points or centroids. (It can be other from the input dataset).

Step-3: Assign each data point to their closest centroid, which will form the predefined K clusters.

Step-4: Calculate the variance and place a new centroid of each cluster.

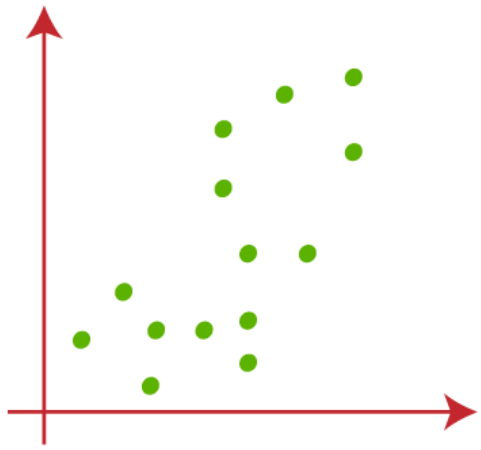
Step-5: Repeat the third steps, which means reassign each datapoint to the new closest centroid of each cluster.

Step-6: If any reassignment occurs, then go to step-4 else go to FINISH.

Step-7: The model is ready.

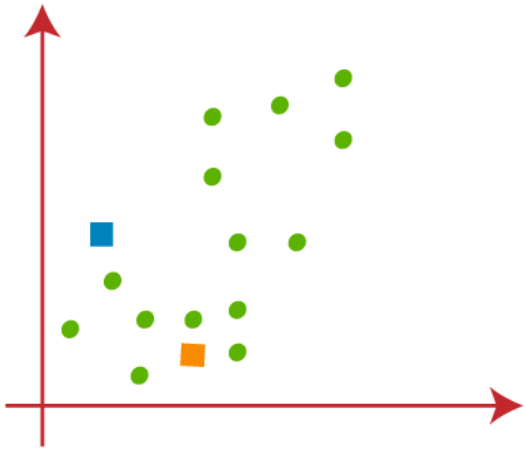
Let's understand the above steps by considering the visual plots:

Suppose we have two variables M1 and M2. The x-y axis scatter plot of these two variables is given below:

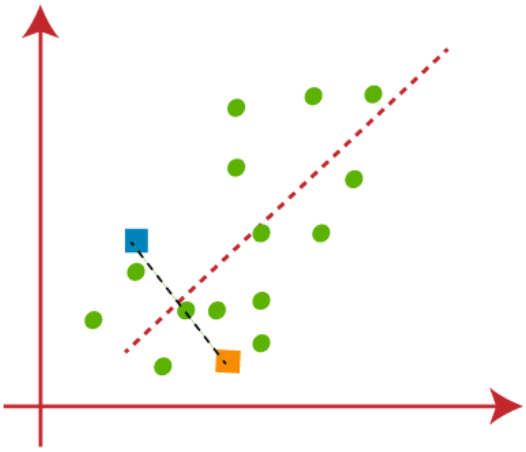


- Let's take number k of clusters, i.e., K=2, to identify the dataset and to put them into different clusters. It means here we will try to group these datasets into two different clusters.
  - We need to choose some random k points or centroid to form the cluster. These points can be either the points from the dataset or any other point. So, here we are selecting the below two points as k points, which are not the part of our dataset.

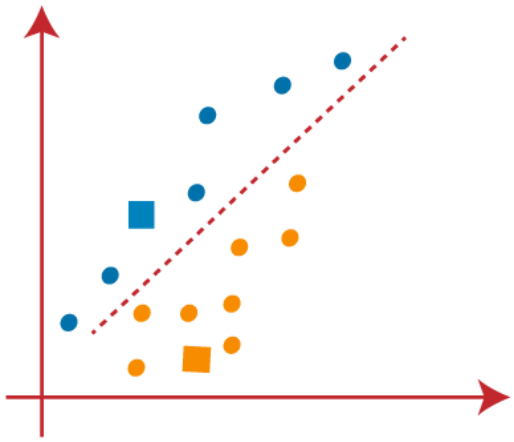
Consider the below image:



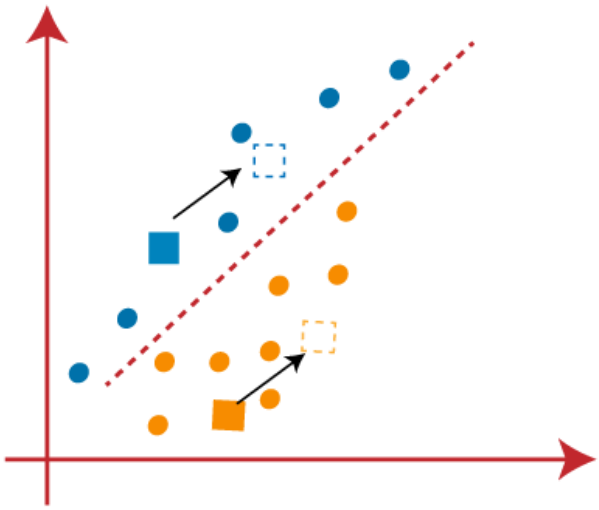
- Now we will assign each data point of the scatter plot to its closest K-point or centroid. We will compute it by applying some mathematics that we have studied to calculate the distance between two points. So, we will draw a median between both the centroids. Consider the below image:



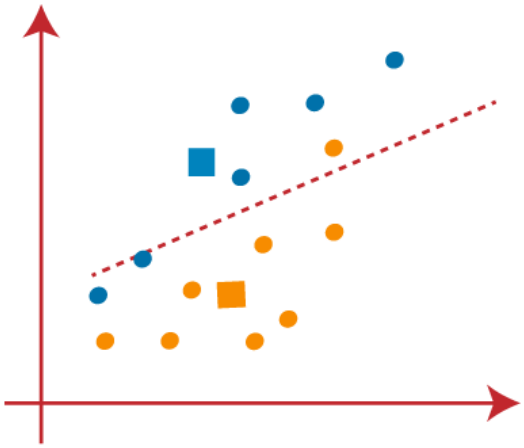
From the above image, it is clear that points left side of the line is near to the K1 or blue centroid, and points to the right of the line are close to the yellow centroid. Let's color them as blue and yellow for clear visualization.



- As we need to find the closest cluster, so we will repeat the process by choosing **a new centroid**. To choose the new centroids, we will compute the center of gravity of these centroids, and will find new centroids as below:



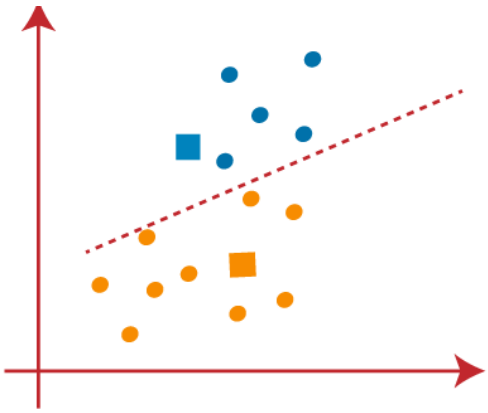
- Next, we will reassign each datapoint to the new centroid. For this, we will repeat the same process of finding a median line. The median will be like below image:



From the above image, we can see, one yellow point is on the left side of the line, and two blue points are right to the line. So, these three points will be assigned to new centroids.

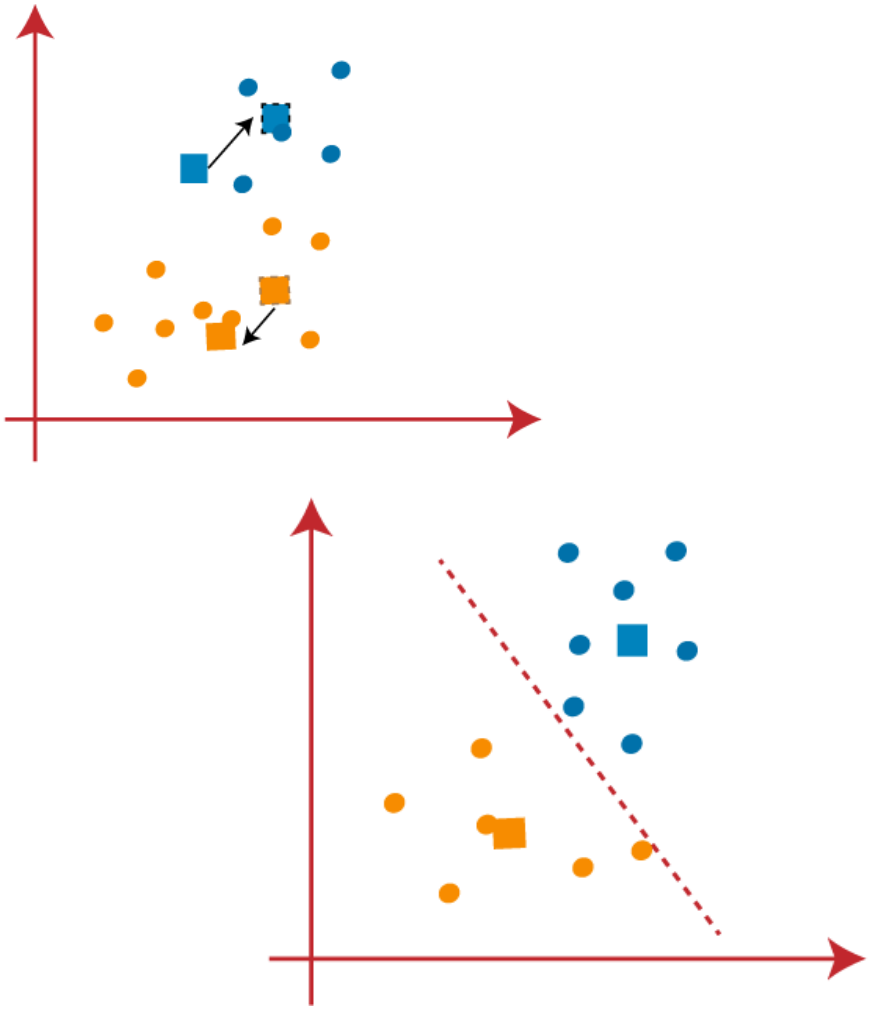
As reassignment has taken place, so we will again go to the step-4, which is finding new centroids or K-points.

- We will repeat the process by finding the center of gravity of centroids, so the new

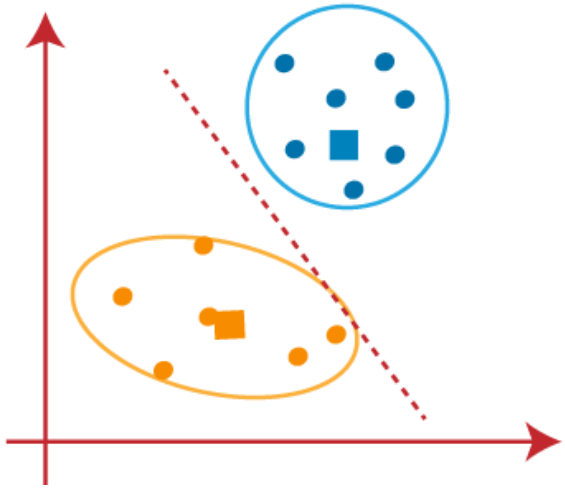


centroids will be as shown in the below image:

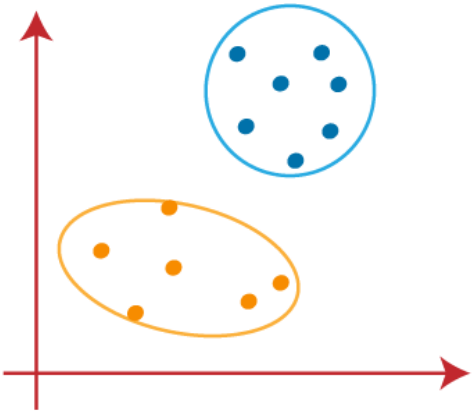
As we got the new centroids so again will draw the median line and reassign the data points. So, the image will be:



- We can see in the above image; there are no dissimilar data points on either side of the line, which means our model is formed. Consider the below image:



As our model is ready, so we can now remove the assumed centroids, and the two final clusters will be as shown in the below image:



How to choose the value of "K number of clusters" in K-means Clustering?

The performance of the K-means clustering algorithm depends upon highly efficient clusters that it forms. But choosing the optimal number of clusters is a big task. There are some different ways to find the optimal number of clusters, but here we are discussing the most appropriate method to find the number of clusters or value of K. The method is given below:

Elbow Method

The Elbow method is one of the most popular ways to find the optimal number of clusters. This method uses the concept of WCSS value. **WCSS** stands for **Within Cluster Sum of Squares**, which defines the total variations within a cluster.

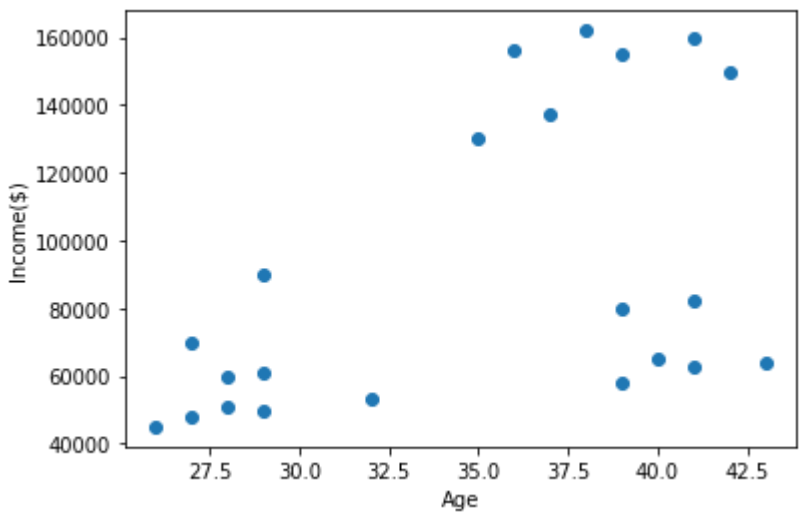
Exercise -

Implement Clustering With K Means

Implementation & Output:

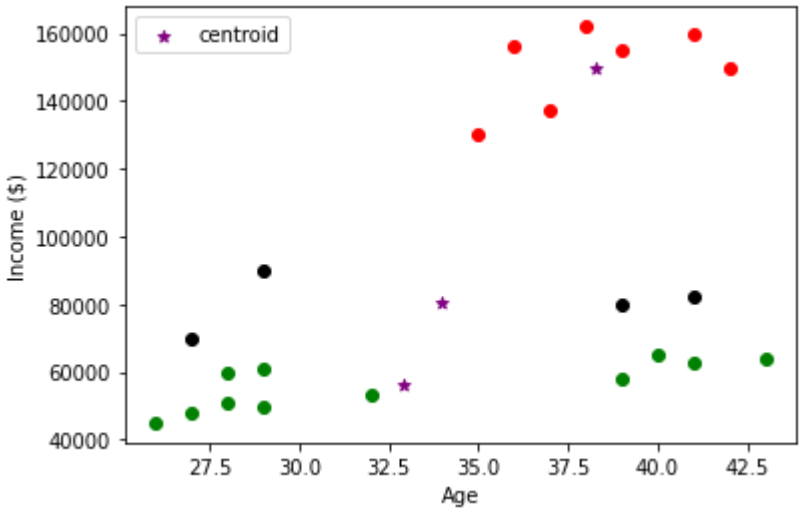
Program:

```
from sklearn.cluster import KMeans

import pandas as pd

from sklearn.preprocessing import MinMaxScaler

from matplotlib import pyplot as plt


url='https://raw.githubusercontent.com/codebasics/py/master/ML/13_kmeans/income.csv'

df = pd.read_csv(url)

df.head()


plt.scatter(df.Age,df['Income($)'])

plt.xlabel('Age')

plt.ylabel('Income($)')
```



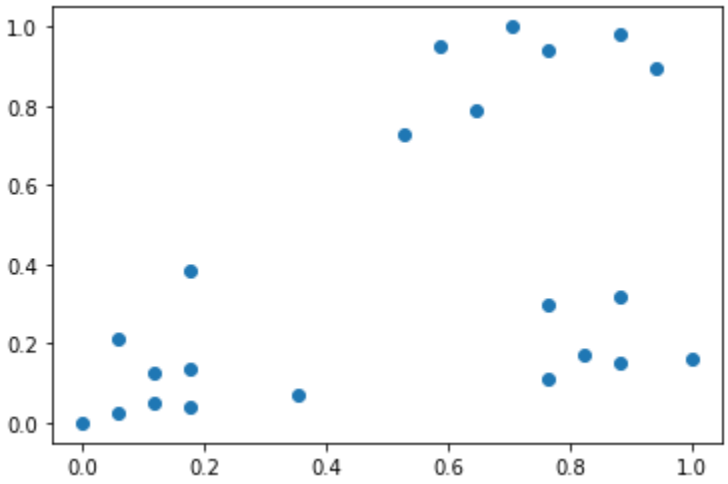

```
km = KMeans(n_clusters=3)
y_predicted = km.fit_predict(df[['Age','Income($)']])
y_predicted

df['cluster']=y_predicted
df.head()

km.cluster_centers_

df1 = df[df.cluster==0]
df2 = df[df.cluster==1]
df3 = df[df.cluster==2]
plt.scatter(df1.Age,df1['Income($)'],color='green')
plt.scatter(df2.Age,df2['Income($)'],color='red')
plt.scatter(df3.Age,df3['Income($)'],color='black')
```

```
plt.scatter(km.cluster_centers_[:,0],km.cluster_centers_[:,1],color='purple',marker='*',label
='centroid')

plt.xlabel('Age')

plt.ylabel('Income ($)')

plt.legend()
```



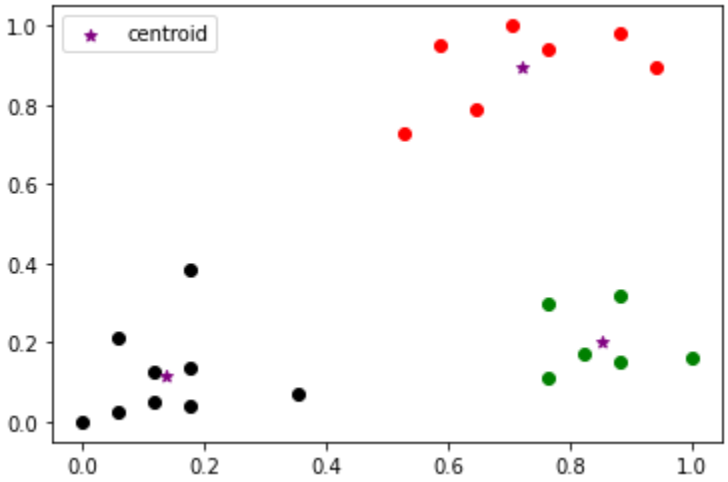

```
scaler = MinMaxScaler()


scaler.fit(df[['Income($)']])

df['Income($)'] = scaler.transform(df[['Income($)']])


scaler.fit(df[['Age']])

df['Age'] = scaler.transform(df[['Age']])


df.head()
```

```
plt.scatter(df.Age,df['Income($)'])
```




```
km = KMeans(n_clusters=3)
y_predicted = km.fit_predict(df[['Age','Income($)']])
y_predicted

df['cluster']=y_predicted
df.head()

km.cluster_centers_

df1 = df[df.cluster==0]
df2 = df[df.cluster==1]
df3 = df[df.cluster==2]
plt.scatter(df1.Age,df1['Income($)'],color='green')
```

```
plt.scatter(df2.Age,df2['Income($)'],color='red')

plt.scatter(df3.Age,df3['Income($)'],color='black')

plt.scatter(km.cluster_centers_[:,0],km.cluster_centers_[:,1],color='purple',marker='*',label
='centroid')

plt.legend()
```



Practical No.8

```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
from matplotlib import style
style.use('fivethirtyeight')
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import AdaBoostClassifier
from sklearn.preprocessing import LabelEncoder
from sklearn.model_selection import cross_validate
import scipy.stats as sps


# Load in the data and define the column labels


dataset = pd.read_csv('/home/tushar/AIML/mushrooms.csv',header=None)
dataset = dataset.sample(frac=1)
dataset.columns = ['target','cap-shape','cap-surface','cap-color','bruises','odor','gill-
attachment','gill-spacing',
        'gill-size','gill-color','stalk-shape','stalk-root','stalk-surface-above-ring','stalk-surface-below-
ring',
        'stalk-color-above-ring',
        'stalk-color-below-ring','veil-type','veil-color','ring-number','ring-type','spore-print-
color','population',
        'habitat']

print(dataset)

# Encode the feature values from strings to integers since the sklearn DecisionTreeClassifier only
takes numerical values
for label in dataset.columns:
   dataset[label] = LabelEncoder().fit(dataset[label]).transform(dataset[label])

print(dataset.info())

Tree_model = DecisionTreeClassifier(criterion="entropy",max_depth=1)


X = dataset.drop('target',axis=1)
Y = dataset['target'].where(dataset['target']==1,-1)



AdaBoost = AdaBoostClassifier(n_estimators=400,learning_rate=1,algorithm='SAMME')
```

AdaBoost.fit(X,Y)

prediction = AdaBoost.score(X,Y)

print('The accuracy is: ',prediction*100,'%')

Output:

```
~/AIML python3 boost_algo.py
     target cap-shape cap-surface cap-color bruises odor  ... veil-color ring-number ring-type spore-print-color population habitat
3483      e         f           f         e       t    n  ...          w           o         p                 k          y       d
1504      e         x           f         g       f    n  ...          w           o         e                 k          a       g
2198      e         x           f         g       t    n  ...          w           o         p                 k          y       d
3086      e         k           f         w       f    n  ...          w           t         p                 w          s       g
3866      p         x           f         w       f    c  ...          w           o         p                 k          v       d
...     ...       ...         ...       ...     ...  ...  ...        ...         ...       ...               ...        ...     ...
1320      e         x           s         w       f    n  ...          w           o         e                 k          s       g
1794      e         f           f         n       f    n  ...          w           o         e                 n          s       g
5313      p         x           y         n       f    s  ...          w           o         e                 w          v       p
5181      p         x           s         b       t    f  ...          w           o         p                 h          v       u
5448      p         x           s         g       t    f  ...          w           o         p                 h          s       u

[8125 rows x 23 columns]
<class 'pandas.core.frame.DataFrame'>
Int64Index: 8125 entries, 3483 to 5448
```

```
Data columns (total 23 columns):
 #   Column                    Non-Null Count  Dtype
---  ------                    --------------  -----
 0   target                    8125 non-null   int64
 1   cap-shape                 8125 non-null   int64
 2   cap-surface               8125 non-null   int64
 3   cap-color                 8125 non-null   int64
 4   bruises                   8125 non-null   int64
 5   odor                      8125 non-null   int64
 6   gill-attachment           8125 non-null   int64
 7   gill-spacing              8125 non-null   int64
 8   gill-size                 8125 non-null   int64
 9   gill-color                8125 non-null   int64
 10  stalk-shape               8125 non-null   int64
 11  stalk-root                8125 non-null   int64
 12  stalk-surface-above-ring  8125 non-null   int64
 13  stalk-surface-below-ring  8125 non-null   int64
 14  stalk-color-above-ring    8125 non-null   int64
 15  stalk-color-below-ring    8125 non-null   int64
 16  veil-type                 8125 non-null   int64
 17  veil-color                8125 non-null   int64
 18  ring-number               8125 non-null   int64
 19  ring-type                 8125 non-null   int64
 20  spore-print-color         8125 non-null   int64
 21  population                8125 non-null   int64
 22  habitat                   8125 non-null   int64
dtypes: int64(23)
```

```
memory usage: 1.5 MB
None
The accuracy is:  100.0 %
```