Practical No. 6

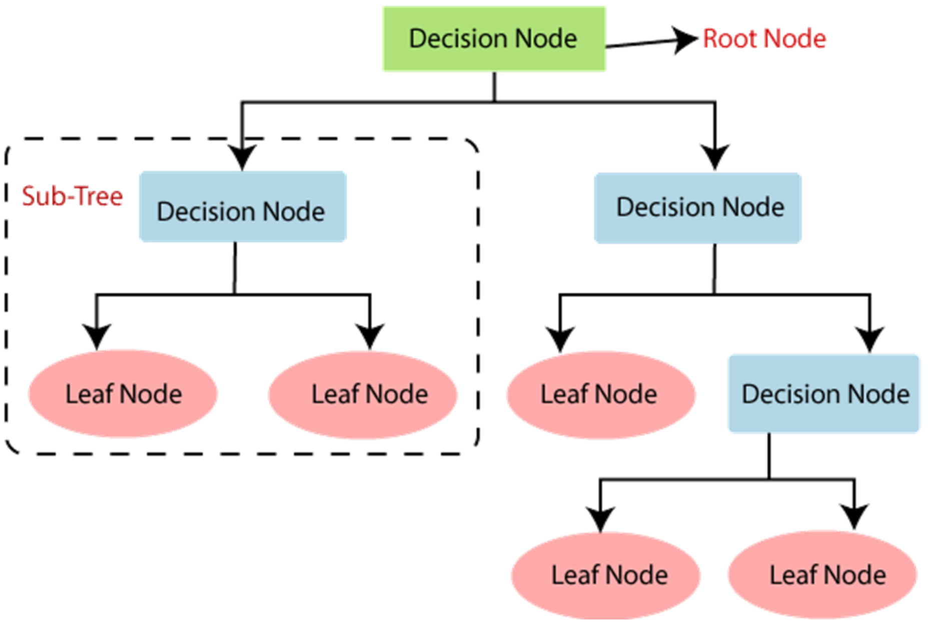
Title: Implementation of Bagging Algorithm: Decision Tree, Random Forest

Aim: Understanding basics of Bagging Algorithm: Decision Tree, Random Forest

Introduction:

Decision Tree

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.

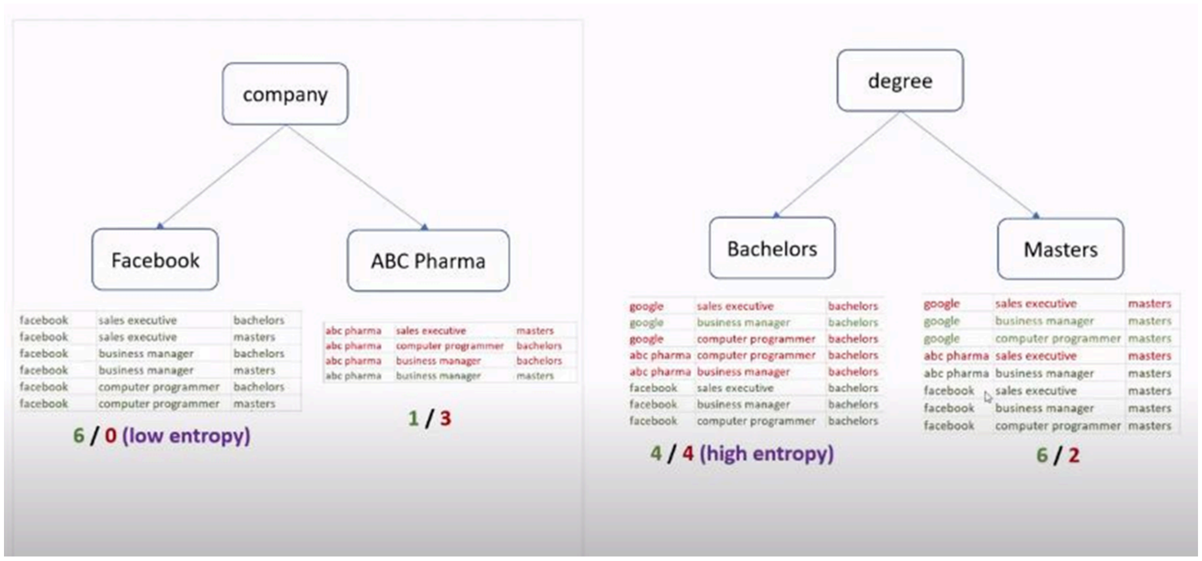


In a Decision tree, there are two nodes, which are the Decision Node and Leaf Node. Decision nodes are used to make any decision and have multiple branches, whereas Leaf nodes are the output of those decisions and do not contain any further branches.

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



facebook	sales executive	bachelors
facebook	sales executive	masters
facebook	business manager	bachelors
facebook	business manager	masters
facebook	computer programmer	bachelors
facebook	computer programmer	masters

6 / 0 (low entropy)

abc pharma	sales executive	masters
abc pharma	computer programmer	bachelors
abc pharma	business manager	bachelors
abc pharma	business manager	masters

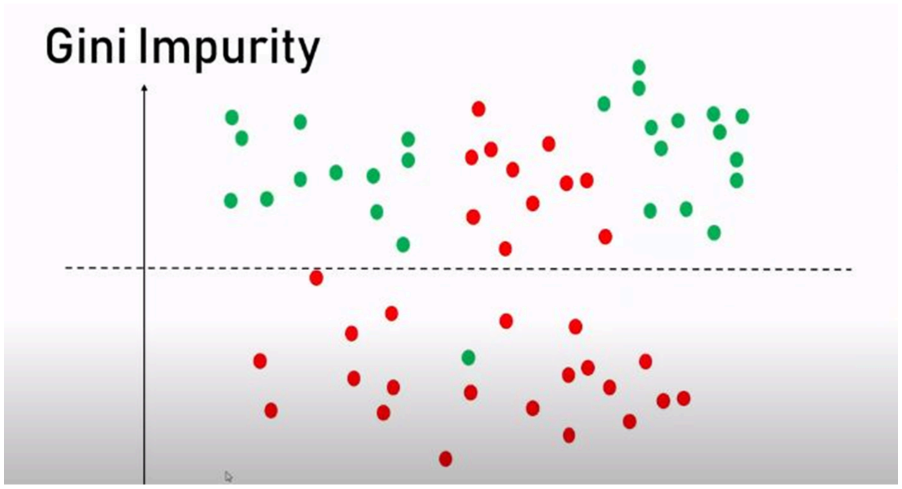
1 / 3

google	sales executive	bachelors
google	business manager	bachelors
google	computer programmer	bachelors
abc pharma	computer programmer	bachelors
abc pharma	business manager	bachelors
facebook	sales executive	bachelors
facebook	business manager	bachelors
facebook	computer programmer	bachelors

4 / 4 (high entropy)

google	sales executive	masters
google	business manager	masters
google	computer programmer	masters
abc pharma	sales executive	masters
abc pharma	business manager	masters
facebook	sales executive	masters
facebook	business manager	masters
facebook	computer programmer	masters

6 / 2

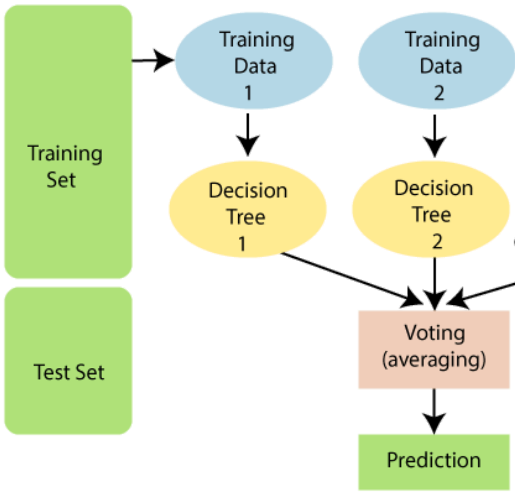


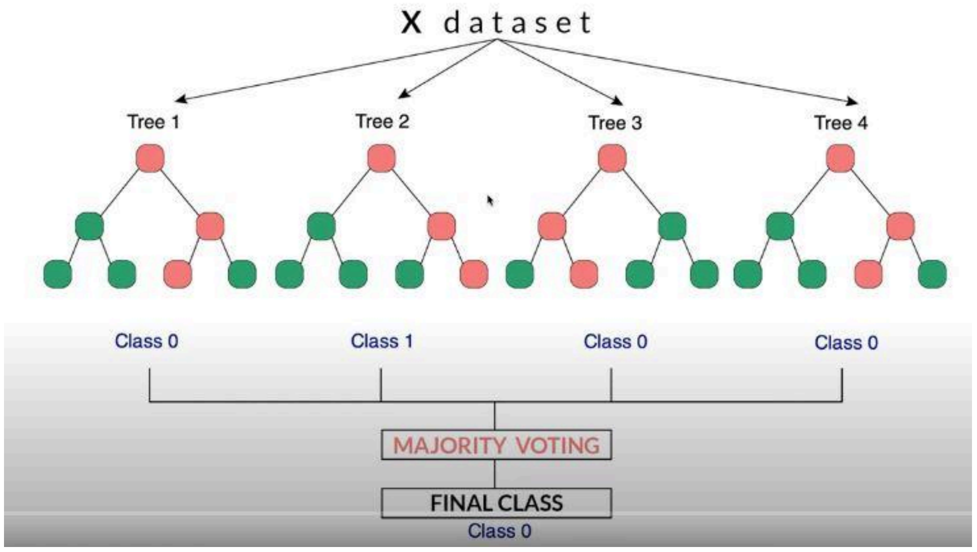
Introduction: Random Forest Algorithm

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

Implementation:
Program:

```
#Bagging Algorithm:Decision Tree
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
14	facebook	computer programmer	bachelors
15	facebook	computer programmer	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
15	facebook	computer programmer	masters	1	1	1

```
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
13	1	0	1
14	1	1	0
15	1	1	1

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

```
▾ DecisionTreeClassifier
DecisionTreeClassifier()
```

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

```
/usr/local/lib/python3.10/dist-packages/sklearn/base.py:439: UserWarning: X does not have valid feature names, but DecisionTreeClas
  warnings.warn(
array([0])
```

```
#Random Forest Algorithm
import numpy as np
import pandas as pd
from matplotlib import pyplot as plt

df = pd.read_csv("https://gist.githubusercontent.com/curran/a08a1080b88344b0c8a7/raw/0e7a9b0a5d2
df
```

	sepal_length	sepal_width	petal_length	petal_width	species
0	5.1	3.5	1.4	0.2	setosa
1	4.9	3.0	1.4	0.2	setosa
2	4.7	3.2	1.3	0.2	setosa
3	4.6	3.1	1.5	0.2	setosa
4	5.0	3.6	1.4	0.2	setosa
...	...	...	...	...	...
145	6.7	3.0	5.2	2.3	virginica
146	6.3	2.5	5.0	1.9	virginica
147	6.5	3.0	5.2	2.0	virginica
148	6.2	3.4	5.4	2.3	virginica
149	5.9	3.0	5.1	1.8	virginica

150 rows × 5 columns

```
from sklearn.model_selection import train_test_split
X_train,X_test,y_train,y_test=train_test_split(df.drop(['species'],axis='columns'),df[['specie
from sklearn.ensemble import RandomForestClassifier
RFC=RandomForestClassifier()
RFC.fit(X_train,y_train)
```

```
/usr/local/lib/python3.11/dist-packages/sklearn/base.py:1389: DataConversionWarning:
  return fit_method(estimator, *args, **kwargs)
```

```
▾ RandomForestClassifier
RandomForestClassifier()
```

```
RFC.predict([[4.6,3.1,1.5,0.2]])
```