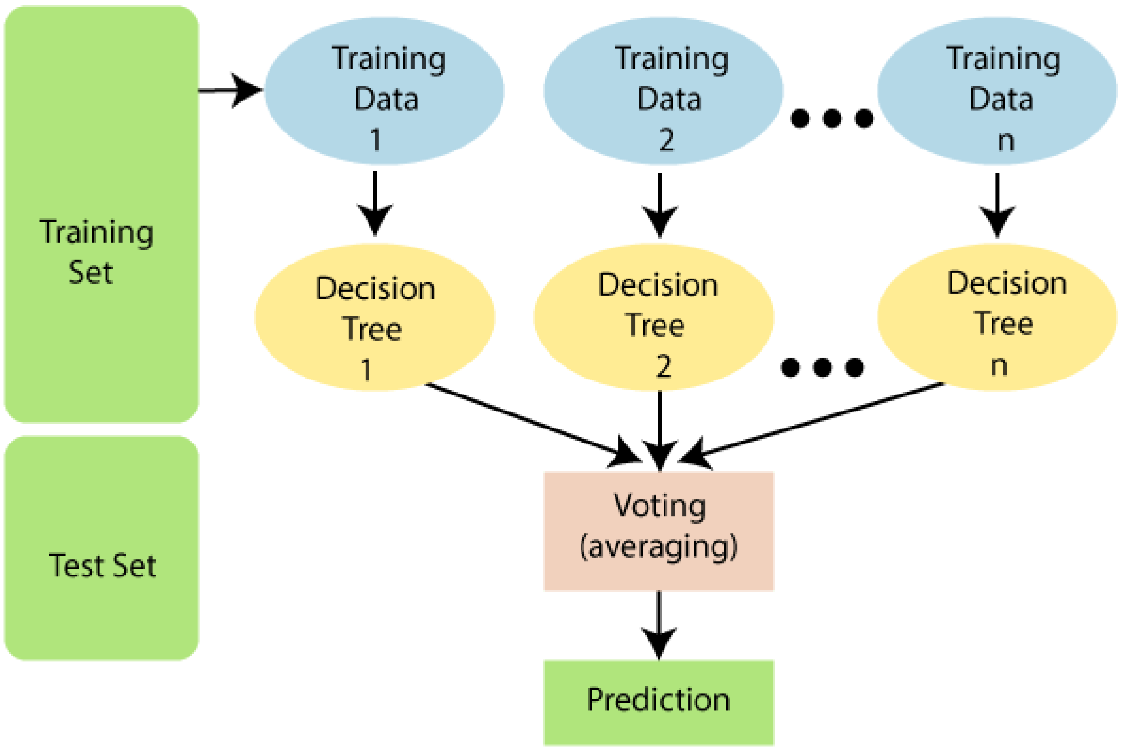
Random Forest

Random Forest is a trademark term for an ensemble of decision trees. In Random Forest, we've collection of decision trees (so known as "Forest"). To classify a new object based on attributes, each tree gives a classification and we say the tree "votes" for that class. The forest chooses the classification having the most votes (over all the trees in the forest).

"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.



- Random Forest is a trademark term for an ensemble of decision trees. In Random Forest, we've collection of decision trees (so known as "Forest").
- To classify a new object based on attributes, each tree gives a classification and we say the tree "votes" for that class. The forest chooses the classification having the most votes (over all the trees in the forest).

Each tree is planted & grown as follows:

- If the number of cases in the traning set is N, then sample of N cases is taken at random but with replacement. This sample will be the traning set for growing the tree.
- If there are M input variables, a number m<<M is specified such that at each node, m variables are selected at random out of the M and the best split on these m is used to split the node. The value of m is held constant during the forest growing.
- Each tree is grown to the largest extent possible. There is no pruning.

Assumptions for Random Forest

- Since the random forest combines multiple trees to predict the class of the dataset, it is possible that some decision trees may predict the correct output, while others may not. But together, all the trees predict the correct output. Therefore, below are two assumptions for a better Random forest classifier:
- There should be some actual values in the feature variable of the dataset so that the classifier can predict accurate results rather than a guessed result.
- The predictions from each tree must have very low correlations.

Below are some points that explain why we should use the Random Forest algorithm:

- It takes less training time as compared to other algorithms.
- It predicts output with high accuracy, even for the large dataset it runs efficiently.
- It can also maintain accuracy when a large proportion of data is missing.

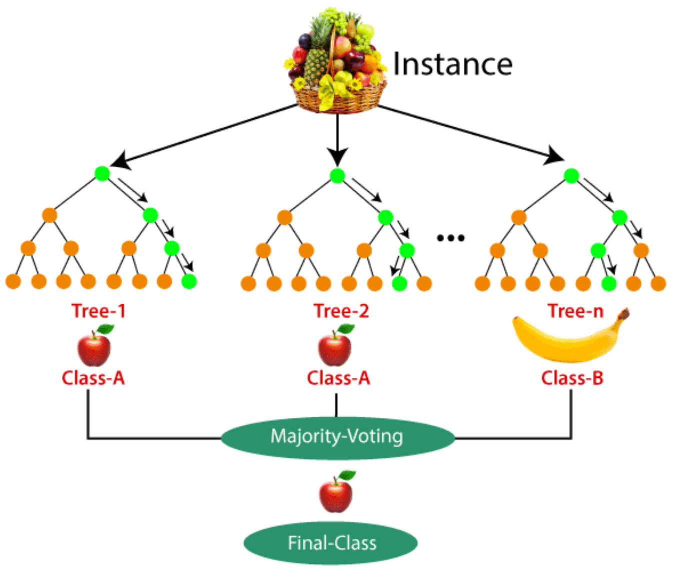
Random Forest works in two-phase first is to create the random forest by combining N decision tree, and second is to make predictions for each tree created in the first phase.

The Working process can be explained in the below steps and diagram:

- Step-1: Select random K data points from the training set.
- Step-2: Build the decision trees associated with the selected data points (Subsets).
- Step-3: Choose the number N for decision trees that you want to build.
- Step-4: Repeat Step 1 & 2.
- Step-5: For new data points, find the predictions of each decision tree, and assign the new data points to the category that wins the majority votes.

Example:

- Suppose there is a dataset that contains multiple fruit images. So, this dataset is given to the Random forest classifier. The dataset is divided into subsets and given to each decision tree. During the training phase, each decision tree produces a prediction result, and when a new data point occurs, then based on the majority of results, the Random Forest classifier predicts the final decision. Consider the below image:



There are mainly four sectors where Random forest mostly used:

- Banking: Banking sector mostly uses this algorithm for the identification of loan risk.
- Medicine: With the help of this algorithm, disease trends and risks of the disease can be identified.
- Land Use: We can identify the areas of similar land use by this algorithm.
- Marketing: Marketing trends can be identified using this algorithm.

Advantages of Random Forest

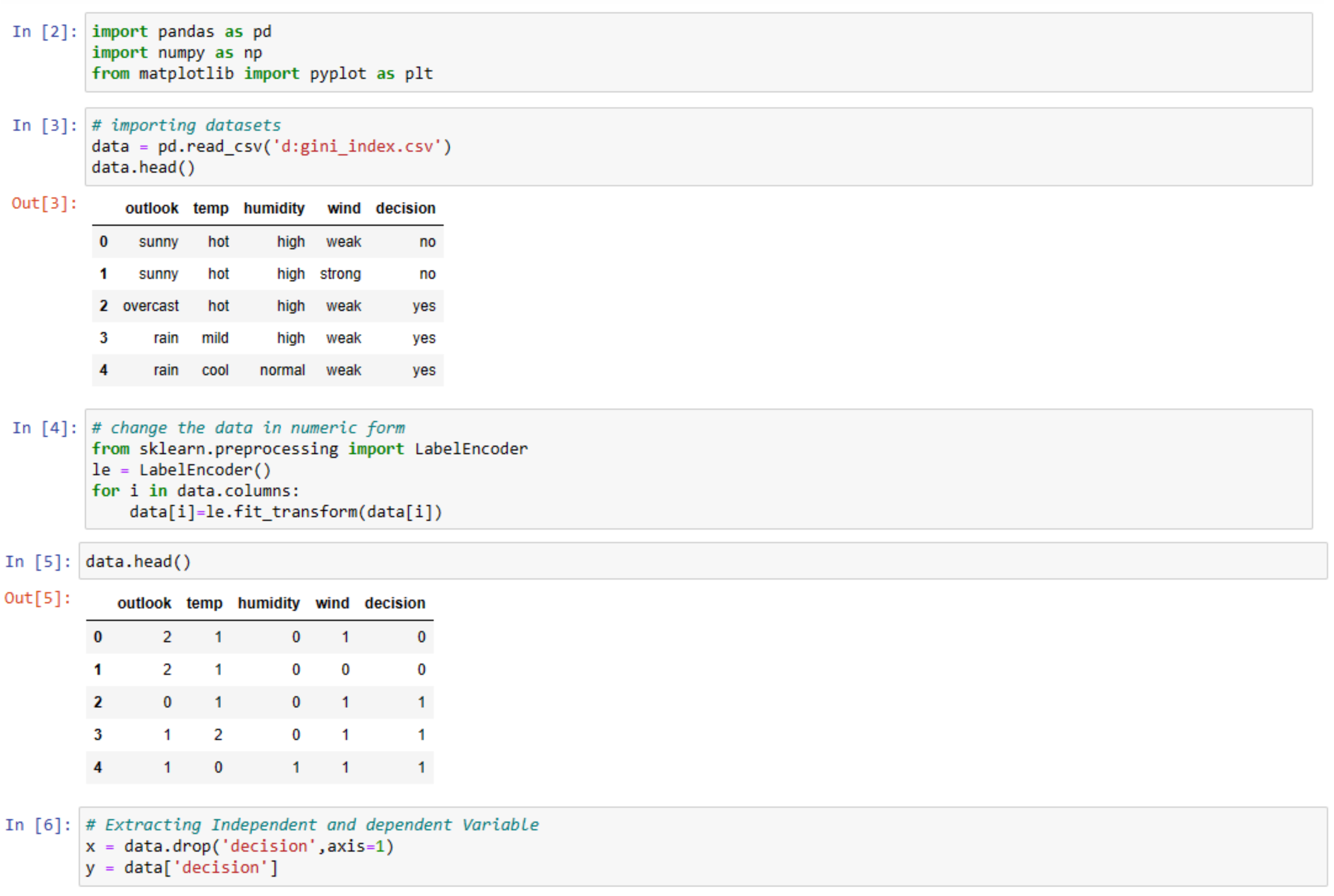
- Random Forest is capable of performing both Classification and Regression tasks.
- It is capable of handling large datasets with high dimensionality.
- It enhances the accuracy of the model and prevents the overfitting issue.

Disadvantages of Random Forest

- Although random forest can be used for both classification and regression tasks, it is not more suitable for Regression tasks.

Implementation Steps are given below:

- Data Pre-processing step
- Fitting the Random forest algorithm to the Training set
- Predicting the test result
- Test accuracy of the result (Creation of Confusion matrix)
- Visualizing the test set result.



```
In [2]: import pandas as pd
        import numpy as np
        from matplotlib import pyplot as plt
```

```
In [3]: # importing datasets
        data = pd.read_csv('d:gini_index.csv')
        data.head()
```

Out[3]:

	outlook	temp	humidity	wind	decision
0	sunny	hot	high	weak	no
1	sunny	hot	high	strong	no
2	overcast	hot	high	weak	yes
3	rain	mild	high	weak	yes
4	rain	cool	normal	weak	yes

```
In [4]: # change the data in numeric form
        from sklearn.preprocessing import LabelEncoder
        le = LabelEncoder()
        for i in data.columns:
            data[i]=le.fit_transform(data[i])
```

```
In [5]: data.head()
```

Out[5]:

	outlook	temp	humidity	wind	decision
0	2	1	0	1	0
1	2	1	0	0	0
2	0	1	0	1	1
3	1	2	0	1	1
4	1	0	1	1	1

```
In [6]: # Extracting Independent and dependent Variable
        x = data.drop('decision',axis=1)
        y = data['decision']
```

```
In [7]: # Splitting the dataset into training and test set.
        from sklearn.model_selection import train_test_split
        x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.20)
```

```
In [8]: # festure scaling
        from sklearn.preprocessing import StandardScaler
        st_x = StandardScaler()
        x_train = st_x.fit_transform(x_train)
        x_test = st_x.fit_transform(x_test)
```

```
In [9]: # Fitting Random Forest classifier to the training set
        from sklearn.ensemble import RandomForestClassifier
        model = RandomForestClassifier(n_estimators=10,criterion='entropy')
        model.fit(x_train,y_train)

Out[9]: RandomForestClassifier(criterion='entropy', n_estimators=10)
```

- n_estimators= The required number of trees in the Random Forest. The default value is 10. We can choose any number but need to take care of the overfitting issue.
- criterion= It is a function to analyze the accuracy of the split. Here we have taken "entropy" for the information gain.

```
In [10]: # Predicting the test set result
         y_pred = model.predict(x_test)
```

```
In [11]: # Creating the Confusion matrix
         from sklearn.metrics import classification_report, confusion_matrix
         print(confusion_matrix(y_test,y_pred))

         print(classification_report(y_test,y_pred))

[[0 1]
 [1 1]]
              precision    recall  f1-score   support

           0       0.00      0.00      0.00         1
           1       0.50      0.50      0.50         2

    accuracy                           0.33         3
   macro avg       0.25      0.25      0.25         3
weighted avg       0.33      0.33      0.33         3
```

Difference b/w Decission tree and Random Forest

Decision trees

1. Decision trees normally suffer from the problem of overfitting if it's allowed to grow without any control
2. A single decision tree is faster in computation.
3. When a data set with features is taken as input by a decision tree it will formulate some set of rules to do prediction.

Random Forest

1. Random forests are created from subsets of data and the final output is based on average or majority ranking and hence the problem of overfitting is taken care of.
2. It is comparatively slower.
3. Random forest randomly selects observations, builds a decision tree and the average result is taken. It doesn't use any set of formulas.

Following hyperparameters increases the predictive power:

1. **n_estimators –** number of trees the algorithm builds before averaging the predictions.
2. **max_features –** maximum number of features random forest considers splitting a node.
3. **mini_sample_leaf –** determines the minimum number of leaves required to split an internal node.

Following hyperparameters increases the speed:

1. **n_jobs** – it tells the engine how many processors it is allowed to use. If the value is 1, it can use only one processor but if the value is -1 there is no limit.
2. **random_state –** controls randomness of the sample. The model will always produce the same results if it has a definite value of random state and if it has been given the same hyperparameters and the same training data.
3. **oob_score –** OOB means out of the bag. It is a random forest cross-validation method. In this one-third of the sample is not used to train the data instead used to evaluate its performance. These samples are called out of bag samples.