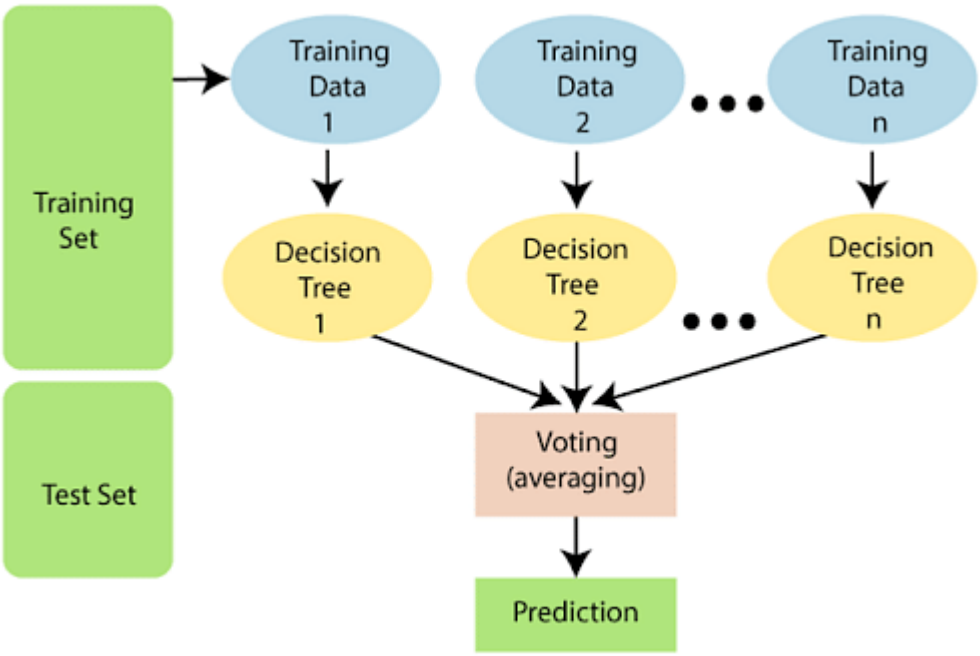
Random Forest Algorithm

A Random Forest Algorithm is a supervised machine learning algorithm that is extremely popular and is used for Classification and Regression problems in Machine Learning. We know that a forest comprises numerous trees, and the more trees more it will be robust. Similarly, the greater the number of trees in a Random Forest Algorithm, the higher its accuracy and problem-solving ability. Random Forest is a classifier that contains several decision trees on various subsets of the given dataset and takes the average to improve the predictive accuracy of that dataset. It is based on the concept of ensemble learning which is a process of combining multiple classifiers to solve a complex problem and improve the performance of the model.

Working of Random Forest Algorithm



The following steps explain the working Random Forest Algorithm:

Step 1: Select random samples from a given data or training set.

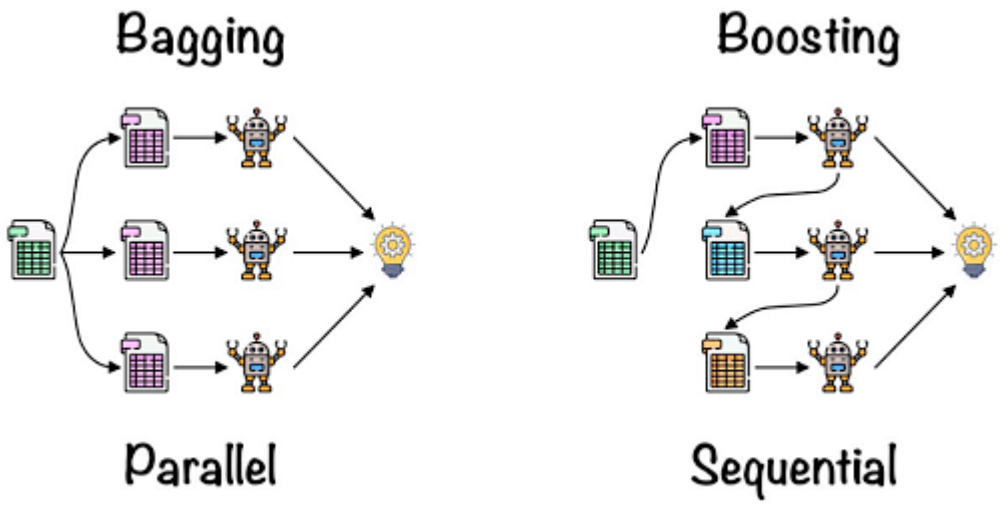
Step 2: This algorithm will construct a decision tree for every training data.

Step 3: Voting will take place by averaging the decision tree.

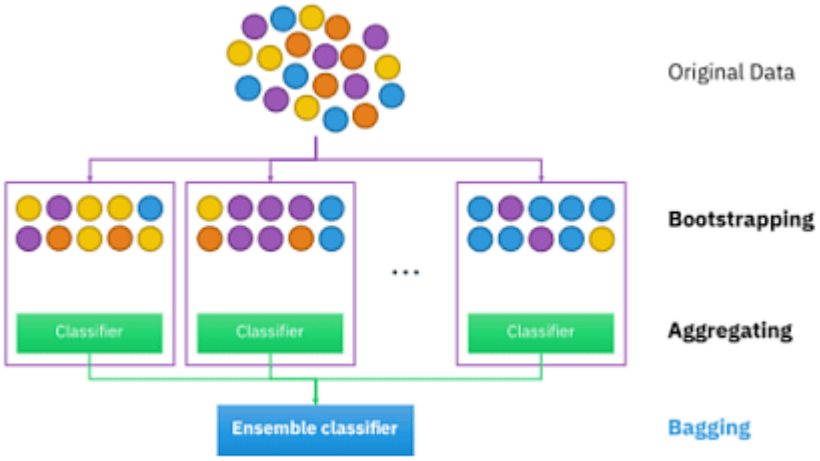
Step 4: Finally, select the most voted prediction result as the final prediction result.

This combination of multiple models is called Ensemble. Ensemble uses two methods:

1. Bagging: Creating a different training subset from sample training data with replacement is called Bagging. The final output is based on majority voting.
2. Boosting: Combing weak learners into strong learners by creating sequential models such that the final model has the highest accuracy is called Boosting. Example: ADA BOOST, XG BOOST.



Bagging: From the principle mentioned above, we can understand Random Forest uses the Bagging code. Now, let us understand this concept in detail. Bagging is also known as Bootstrap Aggregation used by random forest. The process begins with any original random data. After arranging, it is organised into samples known as Bootstrap Sample. This process is known as Bootstrapping. Further, the models are trained individually, yielding different results known as Aggregation. In the last step, all the results are combined, and the generated output is based on majority voting. This step is known as Bagging and is done using an Ensemble Classifier.



Essential Features of Random Forest

- Miscellany: Each tree has a unique attribute, variety and features concerning other trees. Not all trees are the same.
- Immune to the curse of dimensionality: Since a tree is a conceptual idea, it requires no features to be considered. Hence, the feature space is reduced.
- Parallelization: We can fully use the CPU to build random forests since each tree is created autonomously from different data and features.

- Train-Test split: In a Random Forest, we don't have to differentiate the data for train and test because the decision tree never sees 30% of the data.
- Stability: The final result is based on Bagging, meaning the result is based on majority voting or average.

Difference between Decision Tree and Random Forest

Decision Trees	Random Forest
• They usually suffer from the problem of overfitting if it's allowed to grow without any control.	• Since they are created from subsets of data and the final output is based on average or majority ranking, the problem of overfitting doesn't happen here.
• A single decision tree is comparatively faster in computation.	• It is slower.
• They use a particular set of rules when a data set with features are taken as input.	• Random Forest randomly selects observations, builds a decision tree and then the result is obtained based on majority voting. No formulas are required here.

Why Use a Random Forest Algorithm?

There are a lot of benefits to using Random Forest Algorithm, but one of the main advantages is that it reduces the risk of overfitting and the required training time. Additionally, it offers a high level of accuracy. Random Forest algorithm runs efficiently in large databases and produces highly accurate predictions by estimating missing data.

Important Hyperparameters

Hyperparameters are used in random forests to either enhance the performance and predictive power of models or to make the model faster.

The following hyperparameters are used to enhance the predictive power:

- n_estimators: Number of trees built by the algorithm before averaging the products.
- max_features: Maximum number of features random forest uses before considering splitting a node.

- mini_sample_leaf: Determines the minimum number of leaves required to split an internal node.

The following hyperparameters are used to increase the speed of the model:

- n_jobs: Conveys to the engine how many processors are allowed to use. If the value is 1, it can use only one processor, but if the value is -1,, there is no limit.
- 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.
- oob_score: OOB (Out Of the Bag) is a random forest cross-validation method. In this, one-third of the sample is not used to train the data but to evaluate its performance.

Important Terms to Know

There are different ways that the Random Forest algorithm makes data decisions, and consequently, there are some important related terms to know. Some of these terms include:

- Entropy

It is a measure of randomness or unpredictability in the data set.

- Information Gain

A measure of the decrease in the entropy after the data set is split is the information gain.

- Leaf Node

A leaf node is a node that carries the classification or the decision.

- Decision Node

A node that has two or more branches.

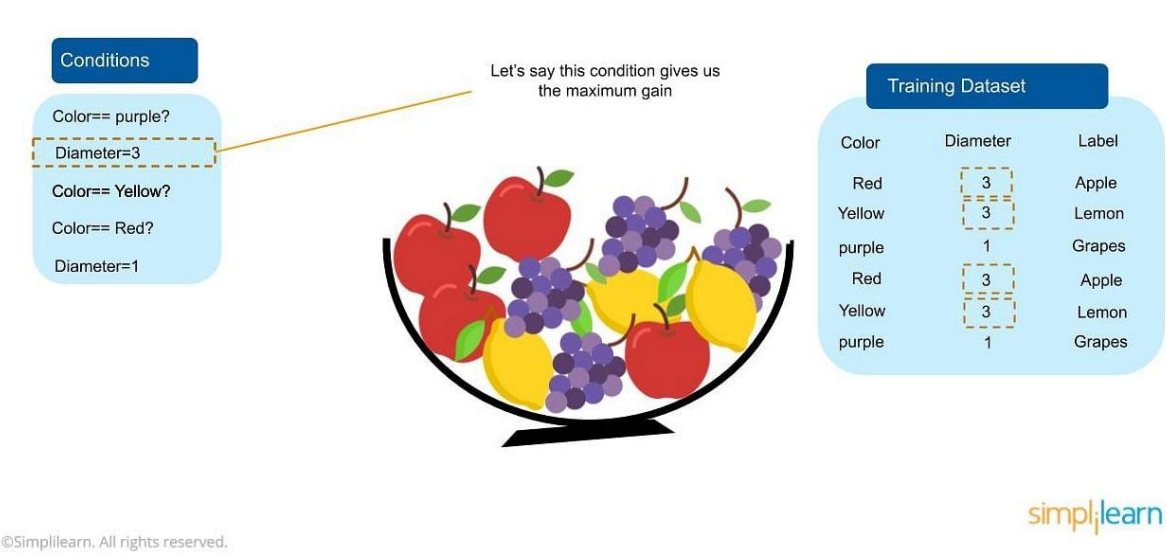
- Root Node

The root node is the topmost decision node, which is where you have all of your data.

Now that you have looked at the various important terms to better understand the random forest algorithm, let us next look at a case example.

Case Example

Let's say we want to classify the different types of fruits in a bowl based on various features, but the bowl is cluttered with a lot of options. You would create a training dataset that contains information about the fruit, including colors, diameters, and specific labels (i.e., apple, grapes, etc.) You would then need to split the data by sorting out the smallest piece so that you can split it in the biggest way possible. You might want to start by splitting your fruits by diameter and then by color. You would want to keep splitting until that particular node no longer needs it, and you can predict a specific fruit with 100 percent accuracy.



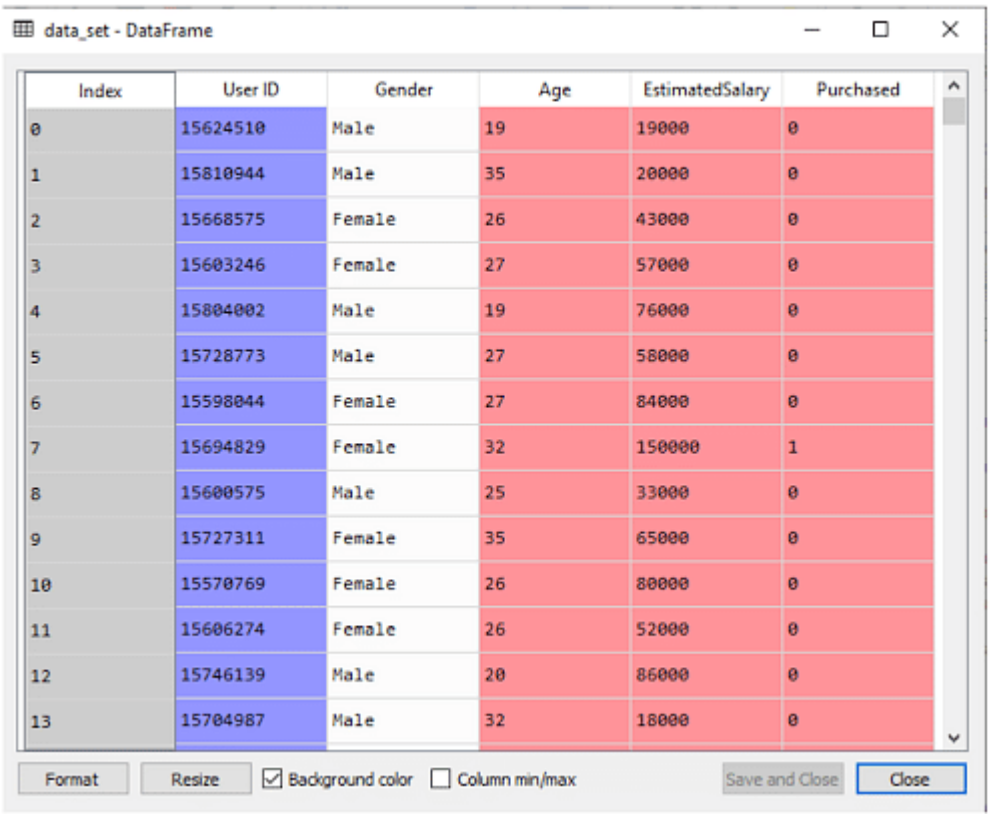
Below is a case example using Python

Coding in Python – Random Forest

1. Data Pre-Processing Step: The following is the code for the pre-processing step-

```
# importing libraries
import numpy as nm
import matplotlib.pyplot as mtp
import pandas as pd

#importing datasets
data_set= pd.read_csv('user_data.csv')

#Extracting Independent and dependent Variable
x= data_set.iloc[:, [2,3]].values
y= data_set.iloc[:, 4].values

# 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=

#feature Scaling
from sklearn.preprocessing import StandardScaler
st_x= StandardScaler()
x_train= st_x.fit_transform(x_train)
x_test= st_x.transform(x_test)
```

We have processed the data when we have loaded the dataset:



Index	User ID	Gender	Age	EstimatedSalary	Purchased
0	15624510	Male	19	19000	0
1	15810944	Male	35	20000	0
2	15668575	Female	26	43000	0
3	15603246	Female	27	57000	0
4	15804002	Male	19	76000	0
5	15728773	Male	27	58000	0
6	15598044	Female	27	84000	0
7	15694829	Female	32	150000	1
8	15600575	Male	25	33000	0
9	15727311	Female	35	65000	0
10	15570769	Female	26	80000	0
11	15606274	Female	26	52000	0
12	15746139	Male	20	86000	0
13	15704987	Male	32	18000	0

2. Fitting the Random Forest Algorithm: Now, we will fit the Random Forest Algorithm in the training set. To do that, we will import RandomForestClassifier class from the sklearn. Ensemble library.

```
#Fitting Decision Tree classifier to the training set
from sklearn.ensemble import RandomForestClassifier
classifier= RandomForestClassifier(n_estimators= 10, criterion="entropy")
classifier.fit(x_train, y_train)
```

Here, the classifier object takes the following parameters:

1. n_estimators: The required number of trees in the Random Forest. The default value is 10.
2. criterion: It is a function to analyse the accuracy of the split.

Output:

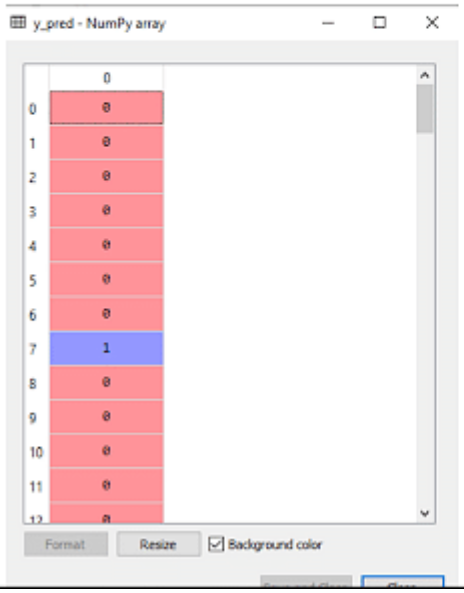
```
RandomForestClassifier(bootstrap=True, class_weight=None, criterion='entropy',
                       max_depth=None, max_features='auto', max_leaf_nodes=None,
                       min_impurity_decrease=0.0, min_impurity_split=None,
                       min_samples_leaf=1, min_samples_split=2,
                       min_weight_fraction_leaf=0.0, n_estimators=10,
                       n_jobs=None, oob_score=False, random_state=None,
                       verbose=0, warm_start=False)
```

3. Predicting the Test Set result:

```
#Predicting the test set result
y_pred= classifier.predict(x_test)
```

Output:

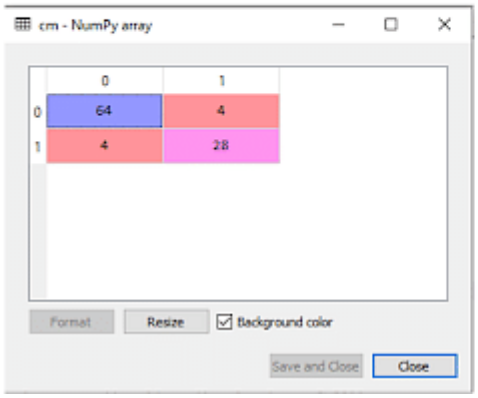
The prediction vector is given as:



4. Creating the Confusion Matrix

```
#Creating the Confusion matrix
from sklearn.metrics import confusion_matrix
cm= confusion_matrix(y_test, y_pred)
```

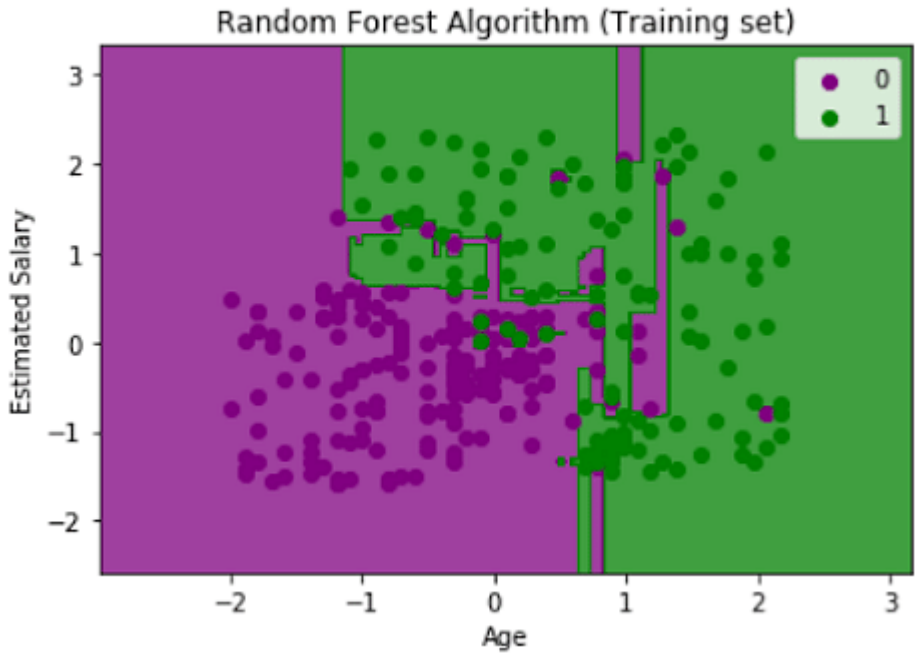
Output:



5. Visualizing the training Set result

```
from matplotlib.colors import ListedColormap
x_set, y_set = x_train, y_train
x1, x2 = nm.meshgrid(nm.arange(start = x_set[:, 0].min() - 1, stop = x_set[:, 0].max() + 1, step  =0.01),
nm.arange(start = x_set[:, 1].min() - 1, stop = x_set[:, 1].max() + 1, step = 0.01))
mtp.contourf(x1, x2, classifier.predict(nm.array([x1.ravel(), x2.ravel()]).T).reshape(x1.shape),
alpha = 0.75, cmap = ListedColormap(('purple','green' )))
mtp.xlim(x1.min(), x1.max())
mtp.ylim(x2.min(), x2.max())
for i, j in enumerate(nm.unique(y_set)):
    mtp.scatter(x_set[y_set == j, 0], x_set[y_set == j, 1],
        c = ListedColormap(('purple', 'green'))(i), label = j)
mtp.title('Random Forest Algorithm (Training set)')
mtp.xlabel('Age')
mtp.ylabel('Estimated Salary')
mtp.legend()
mtp.show()
```

Output:



6. Visualizing the Test Set Result

```
#Visulaizing the test set result
from matplotlib.colors import ListedColormap
x_set, y_set = x_test, y_test
x1, x2 = nm.meshgrid(nm.arange(start = x_set[:, 0].min() - 1, stop = x_set[:, 0].max() + 1, step  =0.01),
nm.arange(start = x_set[:, 1].min() - 1, stop = x_set[:, 1].max() + 1, step = 0.01))
mtp.contourf(x1, x2, classifier.predict(nm.array([x1.ravel(), x2.ravel()]).T).reshape(x1.shape),
alpha = 0.75, cmap = ListedColormap(('purple','green' )))
mtp.xlim(x1.min(), x1.max())
mtp.ylim(x2.min(), x2.max())
for i, j in enumerate(nm.unique(y_set)):
    mtp.scatter(x_set[y_set == j, 0], x_set[y_set == j, 1],
        c = ListedColormap(('purple', 'green'))(i), label = j)
mtp.title('Random Forest Algorithm(Test set)')
mtp.xlabel('Age')
mtp.ylabel('Estimated Salary')
mtp.legend()
mtp.show()
```

Applications of Random Forest

Some of the applications of Random Forest Algorithm are listed below:

1. Banking: It predicts a loan applicant's solvency. This helps lending institutions make a good decision on whether to give the customer loan or not. They are also being used to detect fraudsters.
2. Health Care: Health professionals use random forest systems to diagnose patients. Patients are diagnosed by assessing their previous medical history. Past medical records are reviewed to establish the proper dosage for the patients.
3. Stock Market: Financial analysts use it to identify potential markets for stocks. It also enables them to remember the behaviour of stocks.
4. E-Commerce: Through this system, e-commerce vendors can predict the preference of customers based on past consumption behaviour.

When to Avoid Using Random Forests?

Random Forests Algorithms are not ideal in the following situations:

1. Extrapolation: Random Forest regression is not ideal in the extrapolation of data. Unlike linear regression, which uses existing observations to estimate values beyond the observation range.
2. Sparse Data: Random Forest does not produce good results when the data is sparse. In this case, the subject of features and bootstrapped sample will have an invariant space. This will lead to unproductive spills, which will affect the outcome.

Advantages of Random Forest Algorithm

- Can perform both Regression and classification tasks.
- Produces good predictions that can be understood easily.

- Can handle large data sets efficiently.
- Provides a higher level of accuracy in predicting outcomes over the decision algorithm.

Disadvantages of Random Forest Algorithm

- While using a Random Forest Algorithm, more resources are required for computation.
- It Consumes more time compared to the decision tree algorithm.
- Less intuitive when we have an extensive collection of decision trees.
- Extremely complex and requires more computational resources.