Random forest Algorithm in Machine learning: An Overview

By Great Learning Team / Updated on Jun 13, 2023 / 47251

Introduction to Random Forest Algorithm

In the field of data analytics, every algorithm has a price. But if we consider the overall scenario, then a maximum of the business problem has a classification task. It becomes quite difficult to intuitively know what to adopt considering the nature of the data. Random Forests have various applications across domains such as finance, healthcare, marketing, and more. They are widely used for tasks like fraud detection, customer churn prediction, image classification, and stock market forecasting.

But today we will be discussing one of the top classifier techniques, which is the most trusted by data experts and that is Random Forest Classifier. Random Forest also has a regression algorithm technique which will be covered here.

If you want to learn in-depth, do check out our random forest course for free at Great Learning Academy. Understanding the importance of tree-based classifiers, this course has been curated on tree-based classifiers which will help you understand decision trees, random forests, and how to implement them in Python.

The word 'Forest' in the term suggests that it will contain a lot of trees. The algorithm contains a bundle of decision trees to make a classification and it is also considered a saving technique when it comes to overfitting of a decision tree model. A decision tree model hc high variance and low bias which can give us pretty unstable output unlike the commonly

adopted logistic regression, which has high bias and low variance. That is the only point when Random Forest comes to the rescue. But before discussing Random Forest in detail, let's take a quick look at the tree concept.

"A decision tree is a classification as well as a regression technique. It works great when it comes to taking decisions on data by creating branches from a root, which are essentially the conditions present in the data, and providing an output known as a leaf."

*For more details, we have a comprehensive article on different topic on **Decision Tree** for you to read.*

In the real world, a forest is a combination of trees and in the machine learning world, a Random forest is a combination /ensemble of Decision Trees.

So, let us understand what a decision tree is before we combine it to create a forest.

Imagine you are going to make a major expense, say buy a car. assuming you would want to get the best model that fits your budget, you would not just walk into a showroom and walk out rather drive out with your car. Is it that so?

So, Let's assume you want to buy a car for 4 adults and 2 children, you prefer an SUV with maximum fuel efficiency, you prefer a little luxury like good speakers, sunroof, cosy seating and say you have shortlisted models A and B.

Model A is recommended by your friend X because the speakers are good, and the fuel efficiency is the best.

Model B is recommended by your friend Y because it has 6 comfortable seats, speakers are good and the sunroof is good, the fuel efficiency is low, but he feels the other features convince her that it is the best.

Model B is recommended by your friend Z as well because it has 6 comfortable seats, speakers are better and the sunroof is good, the fuel efficiency is good in her rating.

It is very likely that you would go with Model B as you have majority voting to this model from your friends. Your friends have voted considering the features of their choice and a decision model based on their own logic.

Imagine your friends X, Y, Z as decision trees, you created a random forest with few decision trees and based on the outcomes, you chose the one that was recommended by the majority.

This is how a classifier Random forest works.

What is Random Forest?

Definition from Wikipedia

Random forests or random decision forests are an ensemble learning method for classification, regression and other tasks that operates by constructing a multitude of decision trees at training time. For classification tasks, the output of the random forest is the class selected by most trees. For regression tasks, the mean or average prediction of the individual trees is returned.

Random Forest Features

Some interesting facts about Random Forests – Features

- Accuracy of Random forest is generally very high
- Its efficiency is particularly Notable in Large Data sets
- Provides an estimate of important variables in classification
- Forests Generated can be saved and reused
- Unlike other models It does nt overfit with more features

How random forest works?

Let's Get it working

A random forest is a collection of Decision Trees, Each Tree independently makes a prediction, the values are then averaged (Regression) / Max voted (Classification) to arrive at the final value.

The strength of this model lies in creating different trees with different sub-features from the features. The Features selected for each tree is Random, so the trees do not get deep and are focused only on the set of features.

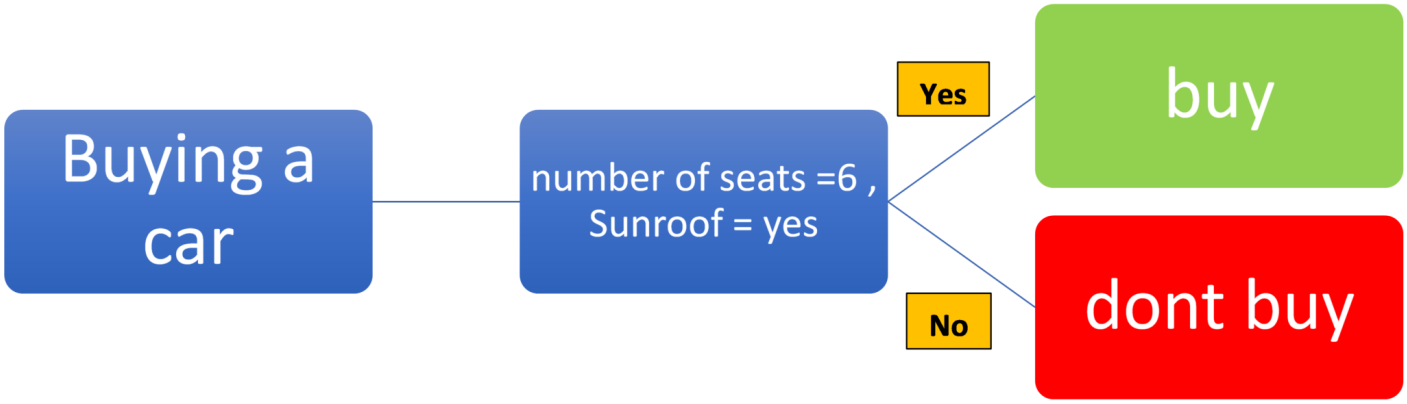
Finally, when they are put together, we create an ensemble of Decision Trees that provides a well-learned prediction.

An Illustration on building a Random Forest

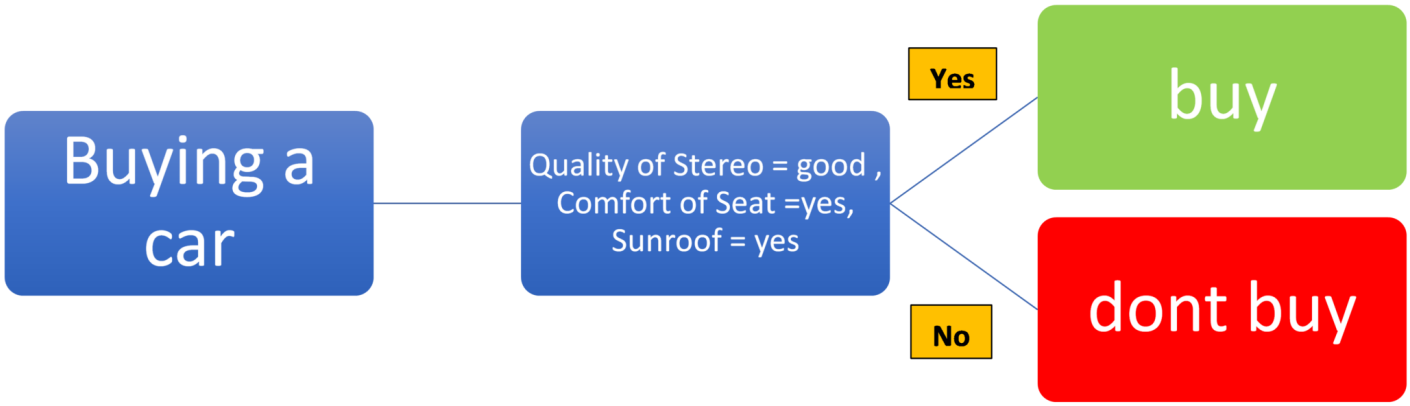
Let us now build a Random Forest Model for say buying a car

One of the decision trees could be checking for features such as Number of Seats and Sunroof availability and deciding yes or no

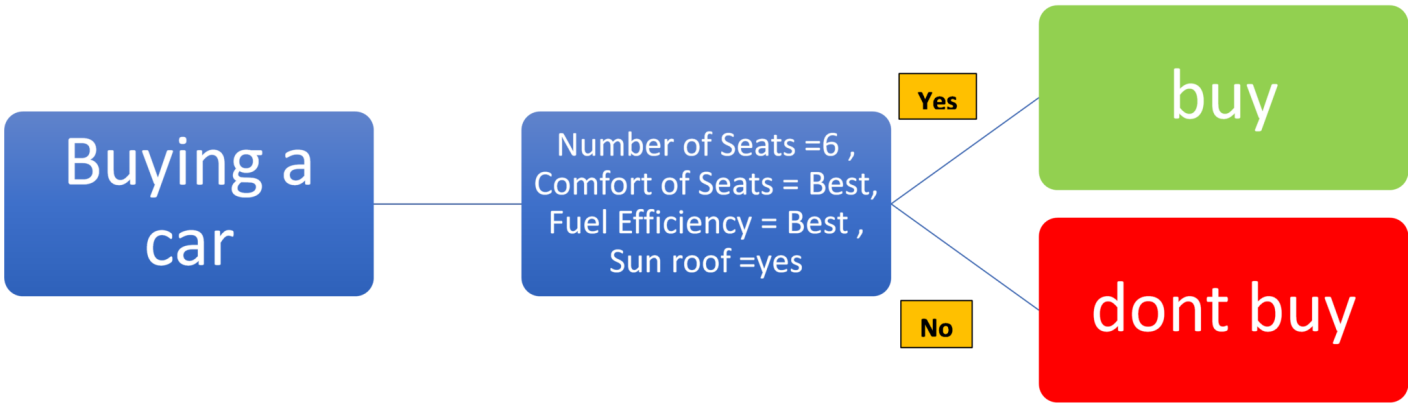
Here the decision tree considers the number of seat parameters to be greater than 6 as the buyer prefers an SUV and prefers a car with a sunroof. The tree would provide the highest value for the model that satisfies both the criteria and would rate it lesser if either of the parameters is not met and rate it lowest if both the parameters are No. Let us see an illustration of the same below:



Another decision tree could be checking for features such as Quality of Stereo, Comfort of Seats and Sunroof availability and decide yes or no. This would also rate the model based on the outcome of these parameters and decide yes or no depending upon the criteria met. The same has been illustrated below.



Another decision tree could be checking for features such as Number of Seats, Comfort of Seats, Fuel Efficiency and Sunroof availability and decide yes or no. The decision Tree for the same is given below.



Each of the decision Tree may give you a Yes or No based on the data set. Each of the trees are independent and our decision using a decision tree would purely depend on the features that particular tree looks upon. If a decision tree considers all the features, the depth of the tree would keep increasing causing an over fit model.

A more efficient way would be to combine these decision Trees and create an ultimate Decision maker based on the output from each tree. That would be a random forest

Once we receive the output from every decision tree, we use the majority vote taken to arrive at the decision. To use this as a regression model, we would take an average of the values.

Let us see how a random forest would look for the above scenario.

The data for each tree is selected using a method called bagging which selects a random set of data points from the data set for each tree. The data selected can be used again (with replacement) or kept aside (without replacement). Each tree would randomly pick the features based on the subset of Data provided. This randomness provides the possibility of finding the feature importance, the feature that influences in the majority of the decision trees would be the feature of maximum importance.

Now once the trees are built with a subset of data and their own set of features, each tree would independently execute to provide its decision. This decision will be a yes or No in the case of classification.

There will then be an ensemble of the trees created using methods such as stacking that would help reduce classification errors. The final output is decided by the max vote method for classification.

Let us see an illustration of the same below.

Each of the decision tree would independently decide based on its own subset of data and features, so the results would not be similar. Assuming the Decision Tree1 suggests ‘Buy’, Decision Tree 2 Suggests ‘Don’t Buy’ and Decision Tree 3 suggests ‘Buy’, then the max vote would be for Buy and the result from Random Forest would be to ‘Buy’

Each tree would have 3 major nodes

- Root Node

- Leaf Node
- Decision Node

The node where the final decision is made is called 'Leaf Node ', The function to decide is made in the 'Decision Node', the 'Root Node' is where the data is stored.

Please note that the features selected will be random and may repeat across trees, this increases the efficiency and compensates for missing data. While splitting a node, only a subset of features is taken into consideration and the best feature among this subset is used for splitting, this diversity results in a better efficiency.

When we create a Random forest Machine Learning model, the decision trees are created based on random subset of features and the trees are split further and further. The entropy or the information gained is an important parameter used to decide the tree split. When the branches are created, total entropy of the subbranches should be less than the entropy of the Parent Node. If the entropy drops, information gained also drops, which is a criterion used to stop further split of the tree. You can learn more with the help of a random forest machine learning course.

How does it differ from the Decision Tree?

A decision tree offers a single path and considers all the features at once. So, this may create deeper trees making the model over fit. A Random forest creates multiple trees with random features, the trees are not very deep.

Providing an option of Ensemble of the decision trees also maximizes the efficiency as it averages the result, providing generalized results.

While a decision tree structure largely depends on the training data and may change drastically even for a slight change in the training data, the random selection of features provides little deviation in terms of structure change with change in data. With the addition of Technique such as Bagging for selection of data, this can be further minimized.

Having said that, the storage and computational capacities required are more for Random Forests than a decision tree.

In summary, Random Forest provides much better accuracy and efficiency than a decision tree, this comes at a cost of storage and computational power.

Let's Regularize through Hyperparameters

Hyper parameters help us to have a certain degree of control over the model to ensure better efficiency, some of the commonly tuned hyperparameters are below.

N_estimators = This parameter helps us to determine the number of Trees in the Forest, higher the number, we create a more robust aggregate model, but that would cost more computational power.

max_depth = This parameter restricts the number of levels of each tree. Creating more levels increases the possibility of considering more features in each tree. A deep tree would create an overfit model, but in Random forest this would be overcome as we would ensemble at the end.

max_features -This parameter helps us restrict the maximum number of features to be considered at every tree. This is one of the vital parameters in deciding the efficiency. Generally, a Grid search with CV would be performed with various values for this parameter to arrive at the ideal value.

bootstrap = This would help us decide the method used for sampling data points, should it be with or without replacement.

max_samples – This decides the percentage of data that should be used from the training data for training. This parameter is generally not touched, as the samples that are not used for training (out of bag data) can be used for evaluating the forest and it is preferred to use the entire training data set for training the forest.

Real World Random Forests

Being a Machine Learning model that can be used for both classification and Prediction, combined with good efficiency, this is a popular model in various arenas.

Random Forest can be applied to any data set with multi-dimensions, so it is a popular choice when it comes to identifying customer loyalty in Retail, predicting stock prices in Finance, recommending products to customers even identifying the right composition of chemicals in the Manufacturing industry.

With its ability to do both prediction and classification, it produces better efficiency than most of the classical models in most of the arenas.

Real-Time Use cases

Random Forest has been the go-to Model for Price Prediction, Fraud Detection in Financial statements, Various Research papers published in these areas recommend Random Forest as the best accuracy producing model. (Ref1, 2)

Random Forest Model has proved to provide good accuracy in predicting disease based on the features (Ref-3)

The Random Forest model has been used to detect Parkinson-related lesions within the midbrain in 3D transcranial ultrasound. This was developed by training the model to understand the organ arrangement, size, shape from prior knowledge and the leaf nodes predict the organ class and spatial location. With this, it provides improved class predictability (Ref 4)

Moreover, a random forest technique has the capability to focus both on observations and variables of training data for developing individual decision trees and take maximum voting for classification and the total average for regression problems respectively. It also uses a bagging technique that takes observations in a random manner and selects all columns which are incapable of representing significant variables at the root for all decision trees. In this manner, a random forest makes trees only which are dependent on each other by penalising accuracy. We have a thumb rule which can be implemented for selecting sub-samples from observations using random forest. If we consider 2/3 of observations for training data and p be the number of columns then

1. For classification, we take sqrt(p) number of columns
2. For regression, we take p/3 number of columns.

The above thumb rule can be tuned in case you like increasing the accuracy of the model.

Let us interpret both bagging and random forest technique where we draw two samples, one in blue and another in pink.

From the above diagram, we can see that the Bagging technique has selected a few observations but all columns. On the other hand, Random Forest selected a few observations and a few columns to create uncorrelated individual trees.

A sample idea of a random forest classifier is given below

The above diagram gives us an idea of how each tree has grown and the variation of the depth of trees as per sample selected but in the end process, voting is performed for final classification. Also, averaging is performed when we deal with the regression problem.

Classifier Vs. Regressor

A random forest classifier works with data having discrete labels or better known as class.

Example- A patient is suffering from cancer or not, a person is eligible for a loan or not, et

A random forest regressor works with data having a numeric or continuous output and they cannot be defined by classes.

Example- the price of houses, milk production of cows, the gross income of companies, etc.

Advantages and Disadvantages of Random Forest

1. It reduces overfitting in decision trees and helps to improve the accuracy
2. It is flexible to both classification and regression problems
3. It works well with both categorical and continuous values
4. It automates missing values present in the data
5. Normalising of data is not required as it uses a rule-based approach.

However, despite these advantages, a random forest algorithm also has some drawbacks.

1. It requires much computational power as well as resources as it builds numerous trees to combine their outputs.
2. It also requires much time for training as it combines a lot of decision trees to determine the class.
3. Due to the ensemble of decision trees, it also suffers interpretability and fails to determine the significance of each variable.

Applications of Random Forest

Banking Sector

Banking analysis requires a lot of effort as it contains a high risk of profit and loss. Customer analysis is one of the most used studies adopted in banking sectors. Problems such as loan default chance of a customer or for detecting any fraud transaction, random forest can be a great choice.

The above representation is a tree which decides whether a customer is eligible for loan credit based on conditions such as account balance, duration of credit, payment status, etc.

Healthcare Sectors

In pharmaceutical industries, random forest can be used to identify the potential of a certain medicine or the composition of chemicals required for medicines. It can also be used in hospitals to identify the diseases suffered by a patient, risk of cancer in a patient, and mc other diseases where early analysis and research play a crucial role.

Credit Card Fraud Detection

Applying Random Forest with Python and R

We will perform case studies in Python and R for both Random forest regression and Classification techniques.

Random Forest Regression in Python

For regression, we will be dealing with data which contains salaries of employees based on their position. We will use this to predict the salary of an employee based on his position.

Let us take care of the libraries and the data:

```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
df = pd.read_csv('Salaries.csv')
df.head()
```

Position	Level	Salary
Business Analyst	1	45000
Junior Consultant	2	50000
Senior Consultant	3	60000
Manager	4	80000
Country Manager	5	110000
Region Manager	6	150000
Partner	7	200000
Senior Partner	8	300000
C-level	9	500000
CEO	10	1000000

```
X =df.iloc[:, 1:2].values
y =df.iloc[:, 2].values
```

As the dataset is very small we won't perform any splitting. We will proceed directly to fitt the data.

```
from sklearn.ensemble import RandomForestRegressor
model = RandomForestRegressor(n_estimators = 10, random_state = 0)
model.fit(X, y)
```

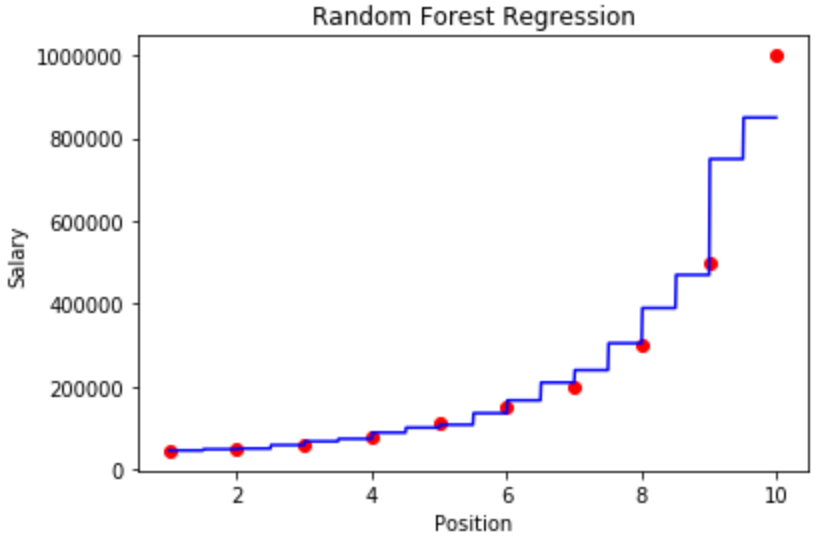
Did you notice that we have made just 10 trees by putting n_estimators=10? It is up to you to play around with the number of trees. As it is a small dataset, 10 trees are enough.

Now we will predict the salary of a person who has a level of 6.5

```
y_pred =model.predict([[6.5]])
```

After prediction, we can see that the employee must get a salary of 167000 after reaching a level of 6.5. Let us visualise to interpret it in a better way.

```
X_grid_data = np.arange(min(X), max(X), 0.01)
X_grid_data = X_grid.reshape((len(X_grid_data), 1))
plt.scatter(X, y, color = 'red')
plt.plot(X_grid_data,model.predict(X_grid_data), color = 'blue')
plt.title('Random Forest Regression’)
plt.xlabel('Position')
plt.ylabel('Salary')
plt.show()
```



Random Forest Regression in R

Now we will be doing the same model in R and see how it creates an impact in predictior

We will first import the dataset:

```
df = read.csv('Position_Salaries.csv')
df = df[2:3]
```

Level	Salary
1	45000
2	50000
3	60000
4	80000
5	110000
6	150000
7	200000
8	300000
9	500000
10	1000000

In R too, we won't perform splitting as the data is too small. We will use the entire data for training and make an individual prediction as we did in Python

We will use the 'randomForest' library. In case you did not install the package, the below code will help you out.

```
install.packages('randomForest')
library(randomForest)
set.seed(1234)
```

The seed function will help you get the same result that we got during training and testing.

```
model= randomForest(x = df[-2],
                    y = df$Salary,
                    ntree = 500)
```

Now we will predict the salary of a level 6.5 employee and see how much it differs from the one predicted using Python.

```
y_prediction = predict(model, data.frame(Level = 6.5))
```

As we see, the prediction gives a salary of 160908 but in Python, we got a prediction of 167000. It completely depends on the data analyst to decide which algorithm works better. We are done with the prediction. Now it's time to visualise the data

```
install.packages('ggplot2')
library(ggplot2)
x_grid_data = seq(min(df$Level), max(df$Level), 0.01)
ggplot()+geom_point(aes(x = df$Level, y = df$Salary),colour = 'red')
+geom_line(aes(x = x_grid_data, y = predict(model, newdata =
data.frame(Level = x_grid_data))),colour = 'blue') +ggtitle('Truth or
Bluff (Random Forest Regression)') +  xlab('Level') + ylab('Salary')
```

So this is for regression using R. Now let us quickly move to the classification part to see how Random Forest works.

Random Forest Classifier in Python

For classification, we will use Social Networking Ads data which contains information about the product purchased based on age and salary of a person. Let us import the libraries

```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
```

Now let us see the dataset:

```
df = pd.read_csv('Social_Network_Ads.csv')
df
```

User ID	Gender	Age	EstimatedSalary	Purchased
15624510	Male	19	19000	0
15810944	Male	35	20000	0
15668575	Female	26	43000	0
15603246	Female	27	57000	0
15804002	Male	19	76000	0
15728773	Male	27	58000	0
15598044	Female	27	84000	0
15694829	Female	32	150000	1
15600575	Male	25	33000	0
15727311	Female	35	65000	0
15570769	Female	26	80000	0
15606274	Female	26	52000	0
15746139	Male	20	86000	0
15704987	Male	32	18000	0

For your information, the dataset contains 400 rows and 5 columns.

```
X = df.iloc[:, [2, 3]].values
y = df.iloc[:, 4].values
```

Now we will split the data for training and testing. We will take 75% for training and rest for testing.

```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size =
0.25, random_state = 0)
```

Now we will standardise the data using StandardScaler from sklearn library.

```
from sklearn.preprocessing import StandardScaler
sc = StandardScaler()
X_train = sc.fit_transform(X_train)
X_test = sc.transform(X_test)
```

After scaling, let us see the head of the data now.

0	1
-0.804802	0.504964
-0.0125441	-0.567782
-0.309641	0.157046
-0.804802	0.273019
-0.309641	-0.567782
-1.1019	-1.43758
-0.70577	-1.58254
-0.210609	2.15757
-1.99319	-0.0459058
0.878746	-0.770734
-0.804802	-0.596776
-1.00287	-0.422817

Now it's time to fit our model.

```
from sklearn.ensemble import RandomForestClassifier
model = RandomForestClassifier(n_estimators = 10, criterion = 'entropy',
random_state = 0)
model.fit(X_train, y_train)
```

We have made 10 trees and used criterion as 'entropy ' as it is used to decrease the impurity in the data. You can increase the number of trees if you wish but we are keeping it limited to 10 for now.
Now the fitting is over. We will predict the test data.

```
y_prediction = model.predict(X_test)
```

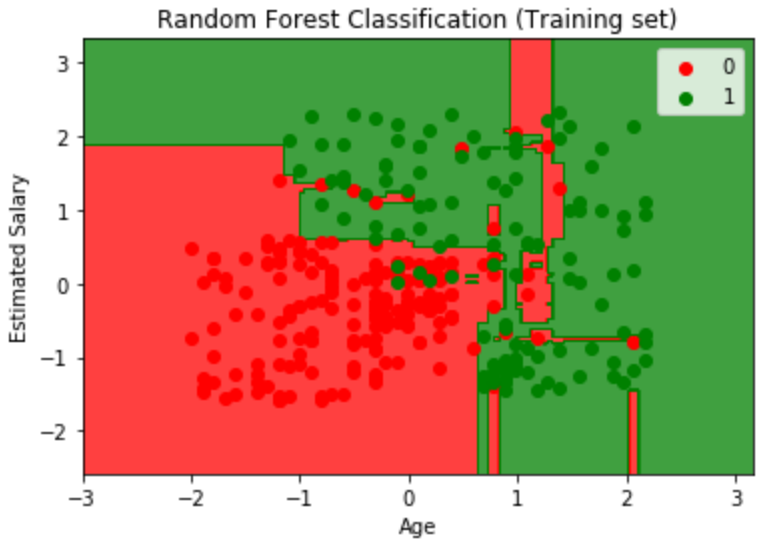
After prediction, we can evaluate by confusion matrix and see how good our model performs.

```
from sklearn.metrics import confusion_matrix
conf_mat = confusion_matrix(y_test, y_prediction)
```

0	1
63	5
4	28

Great. As we see, our model is doing well as the rate of misclassification is very less which is interesting. Now let us visualise our training result.

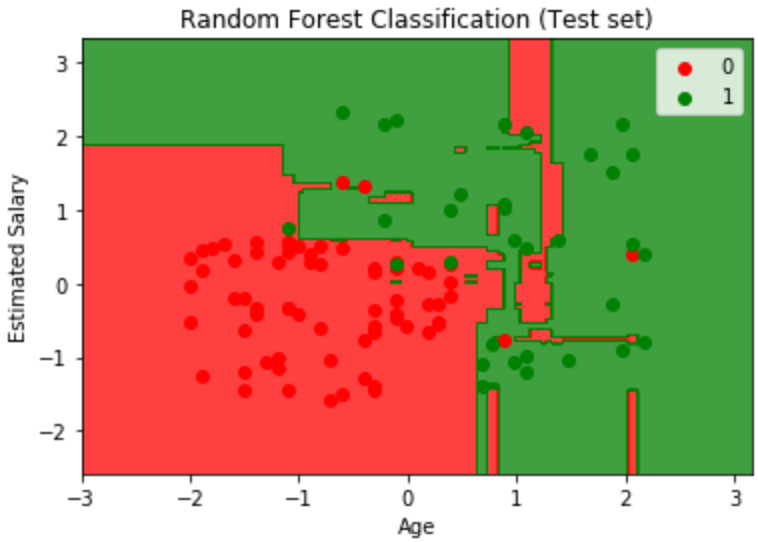
```
from matplotlib.colors import ListedColormap
X_set, y_set = X_train, y_train
X1, X2 = np.meshgrid(np.arange(start = X_set[:, 0].min() - 1, stop =
X_set[:, 0].max() + 1, step = 0.01),np.arange(start = X_set[:, 1].min() -
1, stop = X_set[:, 1].max() + 1, step = 0.01))
plt.contourf(X1,X2,model.predict(np.array([X1.ravel(),
X2.ravel()]).T).reshape(X1.shape),alpha = 0.75, cmap =
ListedColormap(('red', 'green')))
plt.xlim(X1.min(), X1.max())
plt.ylim(X2.min(), X2.max())
for i, j in enumerate(np.unique(y_set)):
    plt.scatter(X_set[y_set == j, 0], X_set[y_set == j, 1],
                c = ListedColormap(('red', 'green'))(i), label = j)
plt.title('Random Forest Classification (Training set)')
plt.xlabel('Age')
plt.ylabel('Salary')
plt.legend()
plt.show()
```



Now let us visualise test result in the same way.

```
from matplotlib.colors import ListedColormap
X_set, y_set = X_test, y_test
X1, X2 = np.meshgrid(np.arange(start = X_set[:, 0].min() - 1, stop =
X_set[:, 0].max() + 1, step = 0.01),np.arange(start = X_set[:, 1].min() -
```

```
1, stop = X_set[:, 1].max() + 1, step = 0.01))
plt.contourf(X1,X2,model.predict(np.array([X1.ravel(),
X2.ravel()]).T).reshape(X1.shape),alpha=0.75,cmap= ListedColormap(('red',
'green')))
plt.xlim(X1.min(), X1.max())
plt.ylim(X2.min(), X2.max())
for i, j in enumerate(np.unique(y_set)):
    plt.scatter(X_set[y_set == j, 0], X_set[y_set == j, 1],
                c = ListedColormap(('red', 'green'))(i), label = j)
plt.title('Random Forest Classification (Test set)')
plt.xlabel('Age')
plt.ylabel('Estimated Salary')
plt.legend()
plt.show()
```



So that's for now. We will move to perform the same model in R.

Random Forest Classifier in R

Let us import the dataset and check the head of the data

```
df = read.csv('SocialNetwork_Ads.csv')
df = df[3:5]
```

Now in R, we need to change the class to factor. So we need further encoding.

```
df$Purchased = factor(df$Purchased, levels = c(0, 1))
```

Now we will split the data and see the result. The splitting ratio will be the same as we did in Python.

```
install.packages('caTools')
library(caTools)
set.seed(123)
split_data = sample.split(df$Purchased, SplitRatio = 0.75)
training_set = subset(df, split_data == TRUE)
test_set = subset(df, split_data == FALSE)
```

Also, we will perform the standardisation of the data and see how it performs while testing.

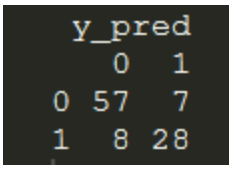
```
training_set[-3] = scale(training_set[-3])
test_set[-3] = scale(test_set[-3])
```

Now we fit the model using the built-in library 'randomForest' provided by R.

```
install.packages('randomForest')
library(randomForest)
set.seed(123)
model= randomForest(x = training_set[-3],
                          y = training_set$Purchased,
                          ntree = 10)
```

We set the number of trees to 10 to see how it performs. We can set any number of trees to improve accuracy.


```
conf_mat = table(test_set[, 3], y_prediction)
conf_mat
```



	y_pred	
	0	1
0	57	7
1	8	28

As we see the model underperforms compared to Python as the rate of misclassification is high.

Now let us interpret our result using visualisation. We will be using ElemStatLearn method for smooth visualisation.

```
library(ElemStatLearn)
train_set = training_set
X1 = seq(min(train_set [, 1]) - 1, max(train_set [, 1]) + 1, by = 0.01)
X2 = seq(min(train_set [, 2]) - 1, max(train_set [, 2]) + 1, by = 0.01)
grid_set = expand.grid(X1, X2)
colnames(grid_set) = c('Age', 'EstimatedSalary')
y_grid = predict(model, grid_set)
plot(set[, -3],
     main = 'Random Forest Classification (Training set)',
     xlab = 'Age', ylab = 'Estimated Salary',
     xlim = range(X1), ylim = range(X2))
contour(X1, X2, matrix(as.numeric(y_grid), length(X1), length(X2)), add =
TRUE)
points(grid_set, pch = '.', col = ifelse(y_grid == 1, 'springgreen3',
'tomato'))
points(train_set, pch = 21, bg = ifelse(train_set [, 3] == 1, 'green4',
'red3'))
```

The model works fine as it is evident from the visualisation of training data. Now let us see how it performs with the test data.

```
library(ElemStatLearn)
testset = test_set
X1 = seq(min(testset [, 1]) - 1, max(testset [, 1]) + 1, by = 0.01)
X2 = seq(min(testset [, 2]) - 1, max testset [, 2]) + 1, by = 0.01)
grid_set = expand.grid(X1, X2)
colnames(grid_set) = c('Age', 'EstimatedSalary')
y_grid = predict(model, grid_set)
plot(set[, -3], main = 'Random Forest Classification (Test set)',
     xlab = 'Age', ylab = 'Estimated Salary',
     xlim = range(X1), ylim = range(X2))
contour(X1, X2, matrix(as.numeric(y_grid), length(X1), length(X2)), add =
TRUE)
points(grid_set, pch = '.', col = ifelse(y_grid == 1, 'springgreen3',
'tomato'))
points(testset, pch = 21, bg = ifelse(testset [, 3] == 1, 'green4',
'red3'))
```

That's it for now. The test data just worked fine as expected.

Inference

Random Forest works well when we are trying to avoid overfitting from building a decision tree. Also, it works fine when the data mostly contain categorical variables. Other algorithms like logistic regression can outperform when it comes to numeric variables but when it comes to making a decision based on conditions, the random forest is the best choice. It completely depends on the analyst to play around with the parameters to improve accuracy. There is often less chance of overfitting as it uses a rule-based approach. But yet again, it depends on the data and the analyst to choose the best algorithm. Random Forest is a very popular Machine Learning Model as it provides good efficiency, the decision making used is very similar to human thinking. The ability to understand the feature importance helps us explain to the model though it is more of a black-box model. The efficiency provided and almost impossible to overfit are the great advantages of this model. This can literally be used in any industry and the research papers published are evidence of the efficacy of this simple yet great model.

If you wish to learn more about the Random Forest or other Machine Learning algorithms, upskill with Great Learning's PG Program in Machine Learning.

Great Learning Team

Great Learning's Blog covers the latest developments and innovations in technology that can be leveraged to build rewarding careers. You'll find career guides, tech tutorials and industry news to keep yourself updated with the fast-changing world of tech and business.

Recommended for you

Artificial Intelligence Tutorial for Beginners in 2024 | Learn AI Tutorial from Experts

Top 20 Applications of Deep Learning in 2024 Across Industries

Basics of building an Artificial Intelligence Chatbot – 2024

IOT Career Opportunities: Ultimate Guide 2024

Top Machine Learning Projects in 2024

10 Hottest Artificial Intelligence (AI) Technologies in 2024

Leave a Comment

Your email address will not be published. Required fields are marked *

Type here..

Name*

Email*

Website

Save my name, email, and website in this browser for the next time I comment.

I'm not a robot

reCAPTCHA
Privacy - Terms

Post Comment »

Free Courses

Python for Machine Learning

Data Science Foundations

Deep Learning with Python

Introduction to Cyber Security

Introduction to Digital Marketing

Java Programming

View More →

Blog Categories

Data Science

Artificial Intelligence

Career

Cybersecurity

Full Stack Development

Study Abroad

Study In USA

Popular Courses

PGP In Data Science and Business Analytics

PGP In Artificial Intelligence And Machine Learning

PGP In Management

PGP In Cloud Computing

Software Engineering Course

PGP In Digital Marketing

View More →

Salary Blogs

Salary Calculator

Data Architect Salary

Cloud Engineer Salary

Software Engineer Salary

Product Manager Salary

Interview Questions

Java Interview Questions

Python Interview Questions

SQL Interview Questions

Selenium Interview Questions

Machine Learning Interview Questions

NLP Interview Questions

View More →