Random Forest - Regression

Thursday, January 19, 2017 11:43 AM

Intuition

Random forest is a type of Ensemble learning.
Ensemble learning is when you take multiple algorithms or the same algorithm multiple times and we put them together to make something which is much more powerful than the original.

STEP 1: Pick at random K data points from the Training set.

STEP 2: Build the Decision Tree associated to these K data points.

STEP 3: Choose the number Ntree of trees you want to build and repeat STEPS 1 & 2

STEP 4: For a new data point, make each one of your Ntree trees predict the value of Y to for the data point in question, and assign the new data point the average across all of the predicted Y values.

We construct like 500 trees and we take the average of the predictions of all trees. This technique is synonymous to wisdom of masses.

Python

Data pre-processing

```
# Random Forest Regression
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd

dataset = pd.read_csv('Position_Salaries.csv')
X = dataset.iloc[:, 1:2].values
y = dataset.iloc[:, 2].values
```

Building the Random Forest Model

Parameters for Random Forest

n_estimators

: integer, optional (default=10)

The number of trees in the forest.

max_features

: int, float, string or None, optional (default="auto")

The number of features to consider when looking for the best split:

- If int, then consider max_features features at each split.
- If float, then max_features is a percentage and int(max_features * n_features) features are considered at each split.
- If "auto", then max_features=n_features.
- If "sqrt", then max_features=sqrt(n_features).
- If "log2", then max_features=log2(n_features).
- If None, then max_features=n_features.

random_state

: int, RandomState instance or None, optional (default=None)

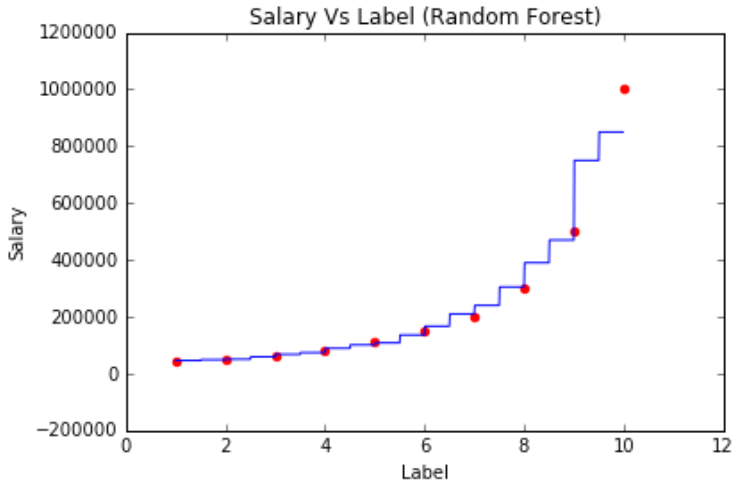
If int, random_state is the seed used by the random number generator; If RandomState instance, random_state is the random number generator; If None, the random number generator is the RandomState instance used by np.random.

```
# Random Forest Model
from sklearn.ensemble import RandomForestRegressor
regressor = RandomForestRegressor(n_estimators=10, random_state=0)
regressor.fit(X, y)
```

Plotting the model

```
X_grid = np.arange(min(X), max(X), 0.01)
X_grid = X_grid.reshape((len(X_grid), 1))
```

```
plt.scatter(X, y, color = 'red')
plt.plot(X_grid, regressor.predict(X_grid))
plt.title('Salary Vs Label (Random Forest)')
plt.xlabel('Label')
plt.ylabel('Salary')
plt.show()
```



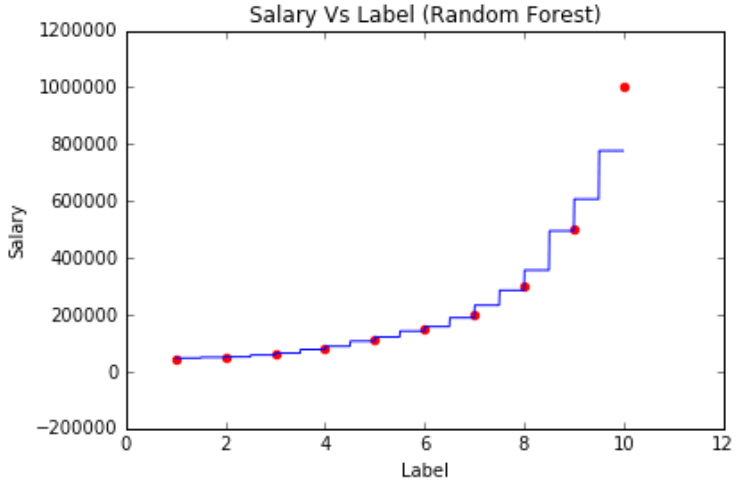
Prediction for a new point

```
# Predict salary of a new employee
value = regressor.predict(np.array([[6.5]]))
print(value)
```

Result - 167000

Changing and tweaking parameters leads to a better fitting model

```
# Random Forest Model
from sklearn.ensemble import RandomForestRegressor
regressor = RandomForestRegressor(n_estimators=100, random_state=0,
                                  max_features= 'sqrt')
regressor.fit(X, y)
```



Prediction for a new employee using this model

```
# Predict salary of a new employee
value = regressor.predict(np.array([[6.5]]))
print(value)
```

Result - 158300

R

Data pre-processing

```
# Import the Data set
dataset = read.csv('Position_Salaries.csv')
dataset = dataset[2:3]
```

Now, we fit the RandomForest Regression model to the dataset. In order to do that, we need to install package 'randomForest'.

```
Install.packages('randomForest')
```

Once we install it, we import the package by using the command

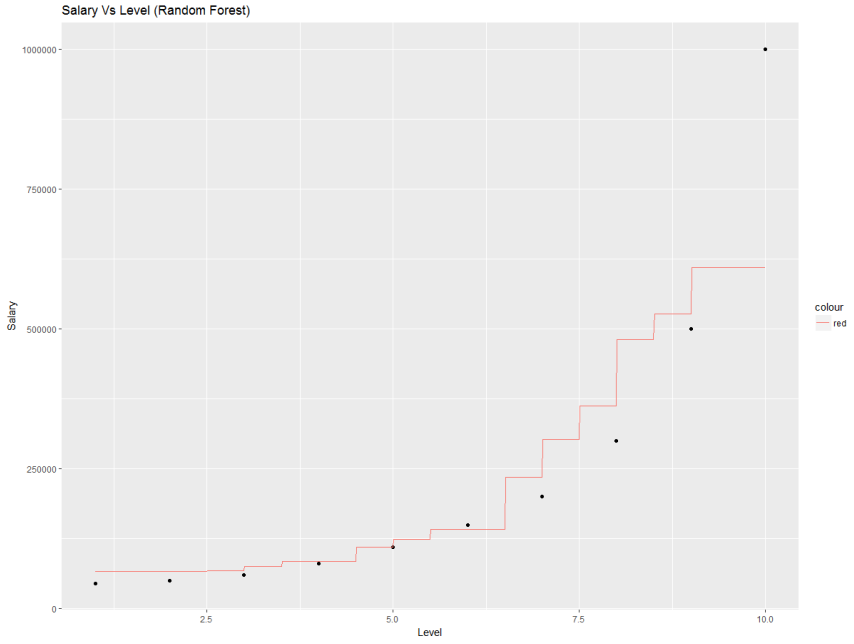
```
Library(randomForest)
```

We use the randomForest function inside this package in order to build a model. We can provide parameters like ntree for the model which determines the number of trees in random forest.

```
# Fitting the Random Forest Regression model to the dataset
library(randomForest)
set.seed(1234)
regressor = randomForest(formula = Salary ~ ., data = dataset, ntree = 10)
```

We finally plot the model as follows

```
# Plotting
library(ggplot2)
X_grid = seq(min(dataset$Level), max(dataset$Level), 0.01)
ggplot() + geom_point(aes(dataset$Level, dataset$Salary)) +
  geom_line(aes(X_grid, predict(regressor, data.frame(Level = X_grid)), color = 'red')) +
  ggtitle('Salary Vs Level (Random Forest)') + xlab('Level') + ylab('Salary')
```



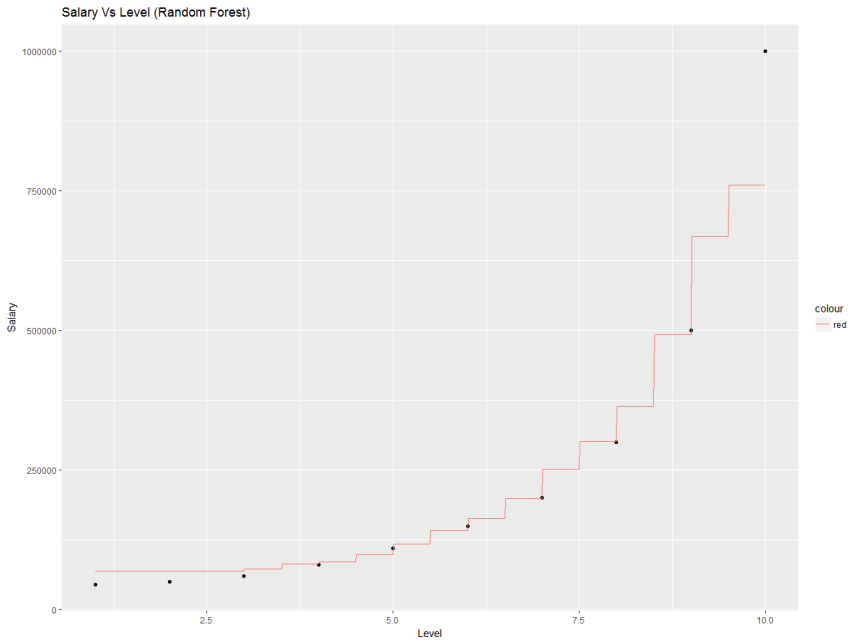
For a new data point wherein Level = 6.5, the predicted salary by this model is

```
# Predicting the salary of a new employee
prediction = predict(regressor, data.frame(Level = 6.5))
```

Result - 141733

We can play around with other parameters of the model. Ex - Increase the ntree to 100 in order to get better estimates

```
regressor = randomForest(formula = Salary ~ ., data = dataset, ntree = 100)
```



The prediction for the new data point is now

```
# Predicting the salary of a new employee
prediction = predict(regressor, data.frame(Level = 6.5))
```

Result - 163545