Ensembling Learning: Boosting and ensembling learning mean when you take multiple algorithms or the same algorithm multiple times and you put them together to make something much more powerful than the original.

Working of Random Forest Algorithm

Before understanding the working of the random forest algorithm in machine learning, we must look into the ensemble learning technique. ***Ensemble*** simply means combining multiple models. Thus a collection of models is used to make predictions rather than an individual model.

Ensemble uses two types of methods:

1. **Bagging**– It creates a different training subset from sample training data with replacement & the final output is based on majority voting. For example, Random Forest.

2. **Boosting**– It combines weak learners into strong learners by creating sequential models such that the final model has the highest accuracy. For example, ADA BOOST, XG BOOST.

As mentioned earlier, Random forest works on the Bagging principle. Now let's dive in and understand bagging in detail.

Bagging

Bagging, also known as ***Bootstrap Aggregation*****,** is the ensemble technique used by random forest. Bagging chooses a random sample/random subset

from the entire data set. Hence each model is generated from the samples (Bootstrap Samples) provided by the Original Data with replacement known as ***row sampling***. This step of row sampling with replacement is called ***bootstrap***. Now each model is trained independently, which generates results. The final output is based on majority voting after combining the results of all models. This step which involves combining all the results and generating output based on majority voting, is known as ***aggregation***.

Random forest is just a team of decision trees. The final prediction of the random forest is simply the average of the different predictions of all the different decision trees.

Here is the 4-step way of the Random Forest

Steps Involved in Random Forest Algorithm

Step 1: In the Random forest model, a subset of data points and a subset of features is selected for constructing each decision tree. Simply put, n random records and m features are taken from the data set having k number of records.

Step 2: Individual decision trees are constructed for each sample.

Step 3: Each decision tree will generate an output.

Step 4: Final output is considered based on ***Majority Voting or Averaging*** for Classification and regression, respectively.

Important Hyperparameters in Random Forest

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

Hyperparameters to Increase the Predictive Power

n_estimators: Number of trees the algorithm builds before averaging the predictions.

max_features: Maximum number of features random forest considers splitting a node.

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

criterion: How to split the node in each tree? (Entropy/Gini impurity/Log Loss,mse)

max_leaf_nodes: Maximum leaf nodes in each tree

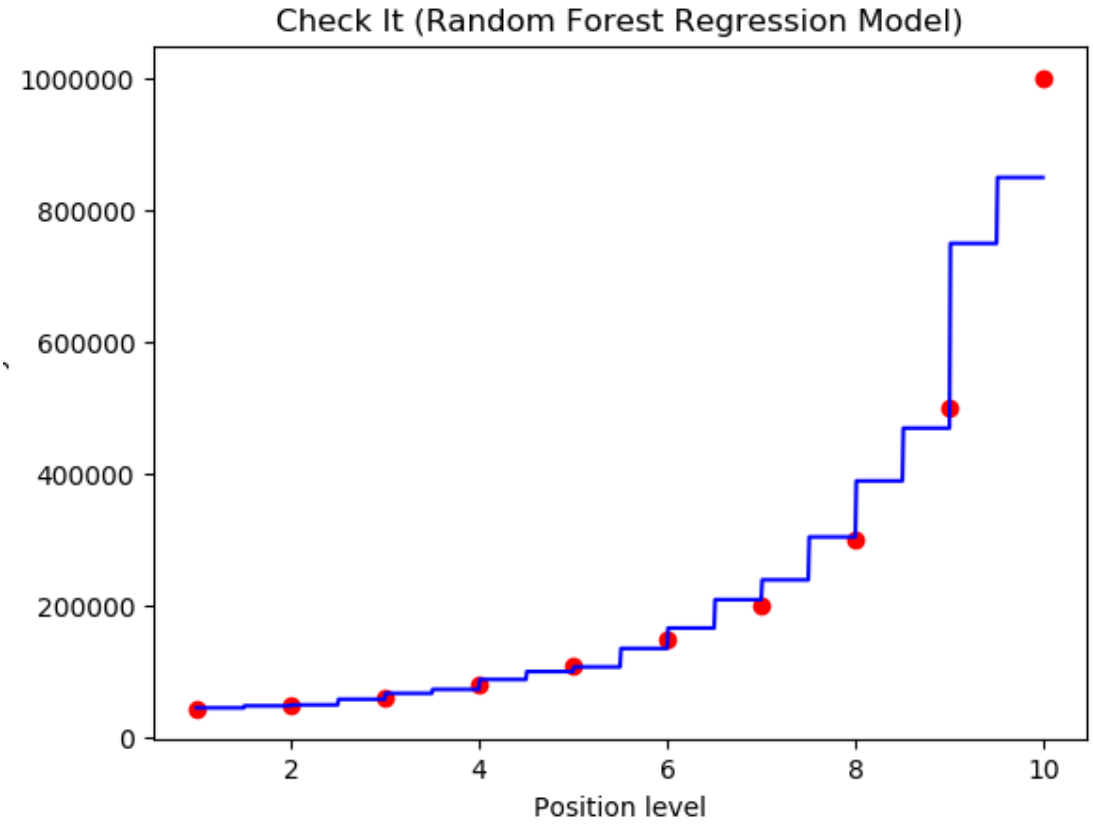
Hyperparameters to Increase the Speed

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.

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 has been given the same hyperparameters and training data.

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.

```
#1 Importing the librariesimport numpy as np
import matplotlib.pyplot as plt
import pandas as pd#2 Importing the datasetdataset =
pd.read_csv('Position_Salaries.csv')
X = dataset.iloc[:, 1:2].values
y = dataset.iloc[:, 2].values# Splitting the dataset into the
Training set and Test set#3 Fitting the Random Forest Regression
Model to the dataset
# Create RF regressor herefrom sklearn.ensemble import
RandomForestRegressor#Put 10 for the n_estimators argument.
n_estimators mean the number #of trees in the forest.regressor =
RandomForestRegressor(n_estimators=10, random_state=0)
regressor.fit(X,y)#4 Visualising the Regression results (for
higher resolution and #smoother curve)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), color = 'blue')
plt.title('Check It (Random Forest Regression Model)')
plt.xlabel('Position level')
plt.ylabel('Salary')
plt.show()
```



There are more steps in this result if we compare it with the Decision Tree Regression.

```
#5 Predicting a new resulty_pred = regressor.predict(5.5)
```

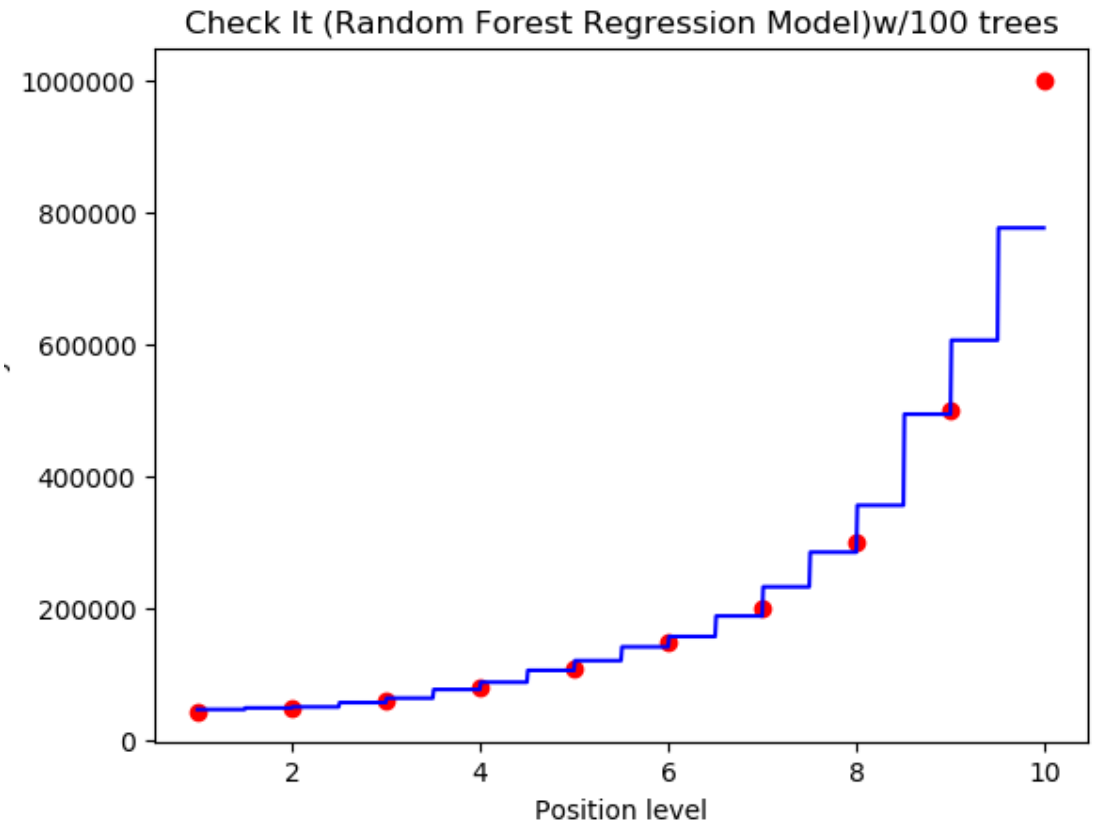
Output: y_pred: 108000

- **Let's increase the trees in the forest to 100 and check it out.**

```
# Fitting the Random Forest Regression Model to the dataset
# Create RF regressor herefrom sklearn.ensemble import
RandomForestRegressor#Put 100 for the n_estimators argument.
n_estimators mean the number #of trees in the forest.regressor =
RandomForestRegressor(n_estimators=100, random_state=0)
regressor.fit(X,y)
```

- **Look at the plot for the model with 100 trees.**

```
#Visualising the Regression results (for higher resolution and
#smoother curve)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), color = 'blue')
plt.title('Check It (Random Forest Regression Model)w/100
trees')
plt.xlabel('Position level')
plt.ylabel('Salary')
plt.show()
```



The steps of the graph don't increase 10 times as the number of trees in the forest. But the prediction will be better. Let's predict the result of the same variable.

```
# Predicting a new result
y_pred = regressor.predict(5.5)
```

Output: y_pred(5.5) = 121800 (It is a better prediction)

Let's increase the trees in the forest to 300 and check it out.

```
#3 Fitting the Random Forest Regression Model to the dataset
# Create RF regressor herefrom sklearn.ensemble import
RandomForestRegressor#Put 300 for the n_estimators argument.
n_estimators mean the number #of trees in the forest.regressor =
RandomForestRegressor(n_estimators=300, random_state=0)
regressor.fit(X,y)#4 Predicting a new result
y_pred = regressor.predict(5.5)
```

Output: y_pred (5.5)= 120233.33 (That's a great prediction and near to the observed values)