Machine Learning with Python

Machine Learning Algorithms - **RANDOM FOREST**

Prof. Shibdas Dutta,

Associate Professor,

DCG DATA CORE SYSTEMS INDIA PVT LTD

Kolkata

Machine Learning Algorithms – Classification Algo- RANDOM FOREST

Introduction - RANDOM FOREST

As the name suggests, the Random forest is a "forest" of trees! i.e Decision Trees.

A random forest is a tree-based machine learning algorithm that randomly selects specific features to build multiple decision trees.

The random forest then combines the output of individual decision trees to generate the final output.

Decision trees involve the greedy selection to the best split point from the dataset at each step.
We can use random forest for classification as well as regression problems.

If the total number of column in the training dataset is denoted by p :

We take sqrt(p) number of columns for classification
For regression, we take a p/3 number of columns.

WHEN TO USE RANDOM FOREST ?

When we focus on accuracy rather than interpretation

If you want better accuracy on the unexpected validation dataset

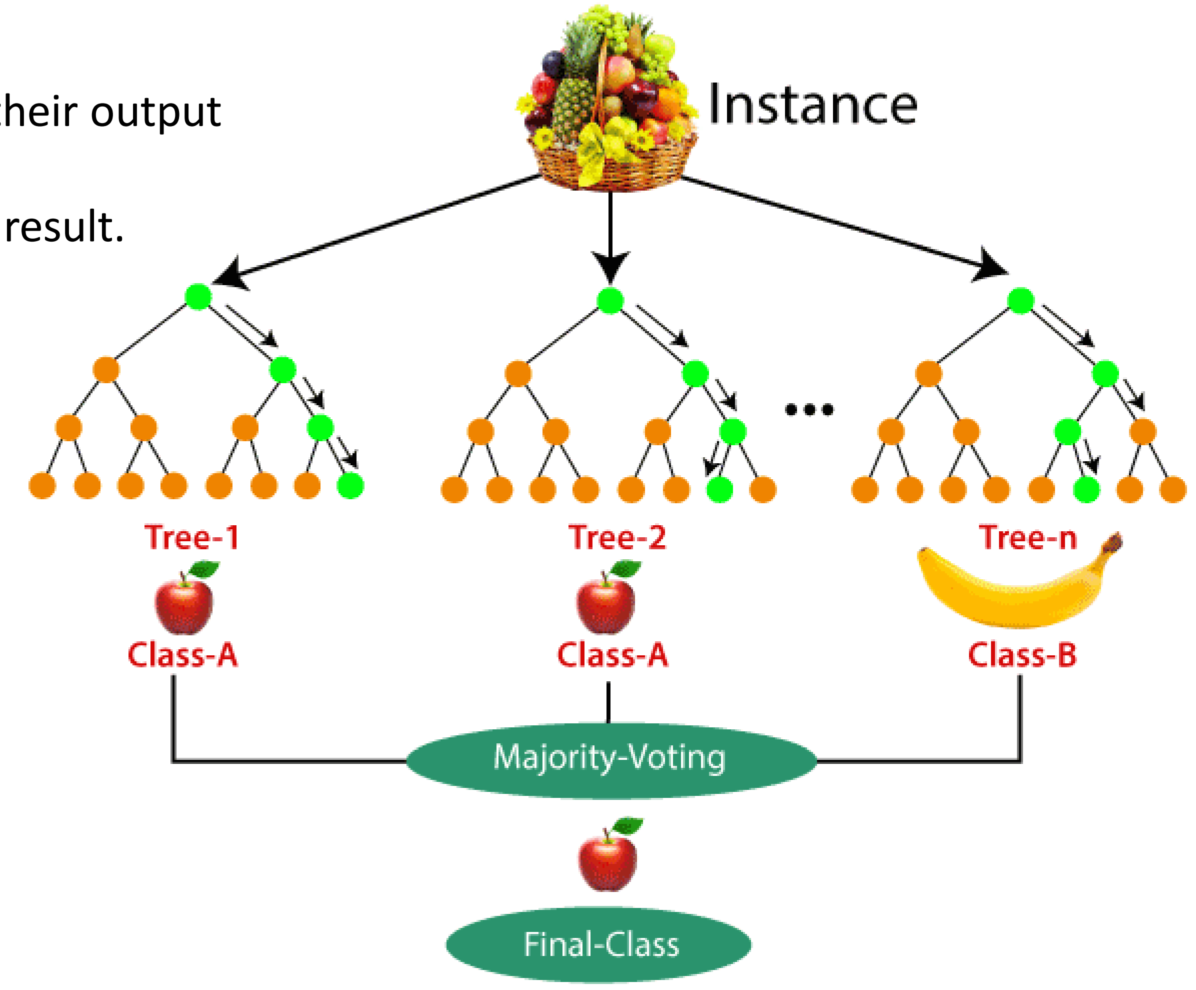
HOW TO USE RANDOM FOREST ?

Select random samples from a given dataset

Construct decision trees from every sample and obtain their output

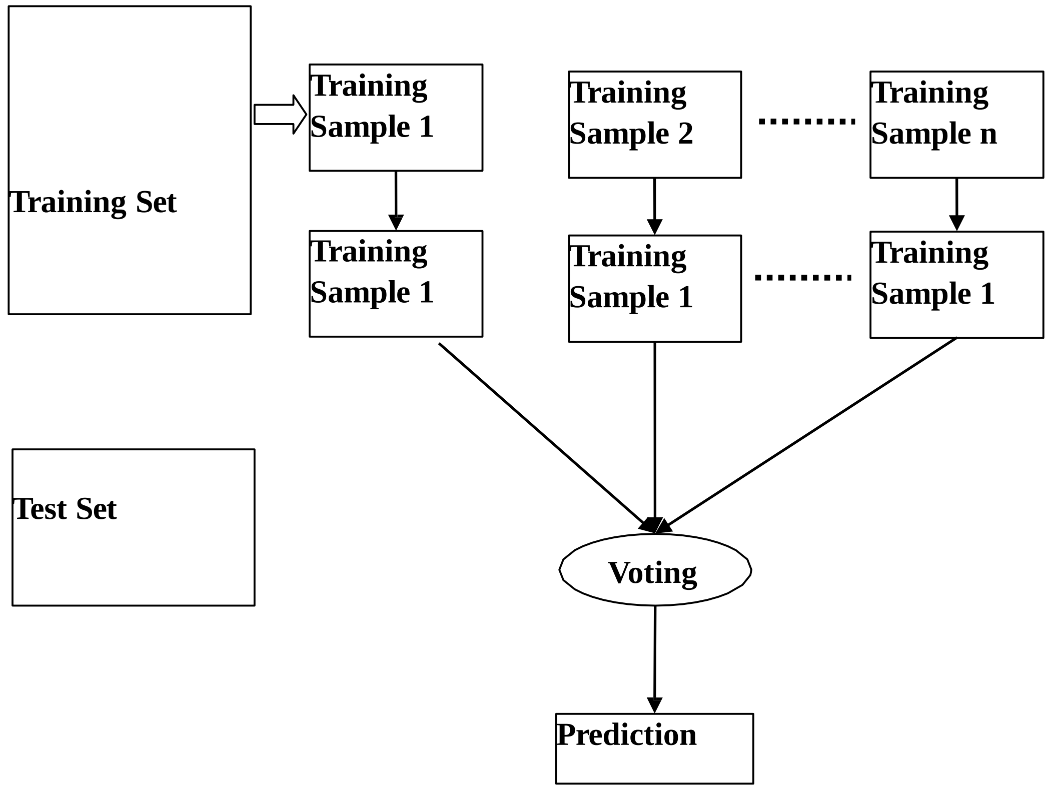
Perform a vote for each predicted result.

Most voted prediction is selected as the final prediction result.



Random Forest

The following diagram will illustrate its working:

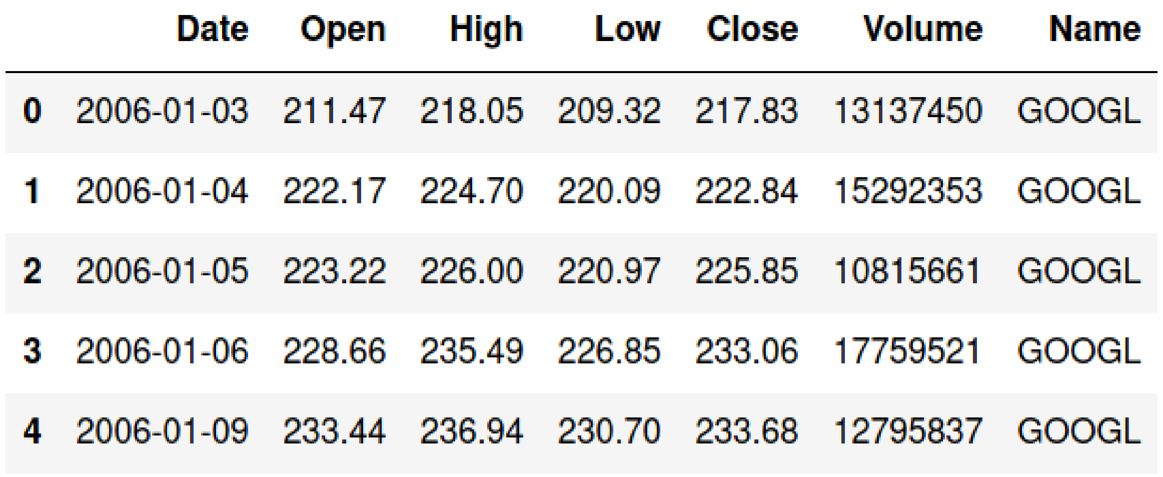


STOCK PREDICTION USING RANDOM FOREST-EXAMPLE

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

# Import the model we are using
from sklearn.ensemble import RandomForestRegressor

data = pd.read_csv('data.csv')
data.head()
```



	Date	Open	High	Low	Close	Volume	Name
0	2006-01-03	211.47	218.05	209.32	217.83	13137450	GOOGL
1	2006-01-04	222.17	224.70	220.09	222.84	15292353	GOOGL
2	2006-01-05	223.22	226.00	220.97	225.85	10815661	GOOGL
3	2006-01-06	228.66	235.49	226.85	233.06	17759521	GOOGL
4	2006-01-09	233.44	236.94	230.70	233.68	12795837	GOOGL

Here, we will be using the dataset (available below) which contains seven columns namely date, open, high, low, close, volume, and name of the company.
Here in this case google is the only company we have used.
Open refers to the time at which people can begin trading on a particular exchange.
Low represents a lower price point for a stock or index.
High refers to a market milestone in which a stock or index reaches a greater price point than previously for a particular time period.
Close simply refers to the time at which a stock exchange closes to trading.
Volume refers to the number of shares of stock traded during a particular time period, normally measured in average daily trading volume.

```
abc=[]
for i in range(len(data)):
    abc.append(data['Date'][i].split('-'))
    data['Date'][i] = ''.join(abc[i])
```

Using above dataset, we are now trying to predict the 'Close' Value based on all attributes. Let's split the data into train and test dataset.

```
#These are the labels: They describe what the stock price was over a period.
X_1 = data.drop('Close',axis=1)
Y_1 = data['Close']

# Using Skicit-learn to split data into training and testing sets

from sklearn.model_selection import train_test_split
X_train_1, X_test_1, y_train_1, y_test_1 = train_test_split(X_1, Y_1, test_size=0.33, random_state=42)
```

Now, let’s instantiate the model and train the model on training dataset:

```
rfg = RandomForestRegressor(n_estimators= 10, random_state=42)
rfg.fit(X_train_1,y_train_1)

pd.concat([pd.Series(rfg.predict(X_test_1)), y_test_1.reset_index(
drop=True)], axis=1)
```

Let’s find out the features on the basis of their importance by calculating numerical feature importances

```
# Saving feature names for later use
feature_list = list(X_1.columns)
print(feature_list)

# Get numerical feature importances
importances = list(rfg.feature_importances_)

# List of tuples with variable and importance
feature_importances = [(feature, round(importance, 2)) for feature, importance in zip(feature_list, importances)]

# Sort the feature importances by most important first
feature_importances = sorted(feature_importances, key = lambda x: x[1], reverse = True)

# Print out the feature and importances
[print('Variable: {:20} Importance: {}'.format(*pair)) for pair in feature_importances];
```

```
rfg.score(X_test_1, y_test_1)
```

We are getting an accuracy of ~99% while predicting. We then display the original value and the predicted Values.

Prediction

```
pd.concat([pd.Series(rfg.predict(X_test_1)), y_test_1.reset_index(drop=True)], axis=1)
```

ADVANTAGES OF RANDOM FOREST

It reduces overfitting as it yields prediction based on majority voting.
Random forest can be used for classification as well as regression.
It works well on a large range of datasets.
Random forest provides better accuracy on unseen data and even if some data is missing
Data normalization isn’t required as it is a rule-based approach

DISADVANTAGES

Random forest requires much more computational power and memory space to build numerous decision trees.
Due to the ensemble of decision trees, it also suffers interpretability and fails to determine the significance of each variable.
Random forests can be less intuitive for a large collection of decision trees.
Using bagging techniques, Random forest makes trees only which are dependent on each other. Bagging might provide similar predictions in each tree as the same greedy algorithm is used to create each decision tree. Hence, it is likely to be using the same or very similar split points in each tree which mitigates the variance originally sought.

Thank You