Machine Learning

2022-12-22

Random Forest

Random forests or random decision forests is an ensemble [1] 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 decision forests correct for decision trees' habit of overfitting to their training. Random forests generally outperform decision trees.

(This part considers the random forest classifier) Random forests are frequently used as "blackbox" [2] models in businesses, as they generate reasonable predictions across a wide range of data while requiring little configuration.

The fundamental concept behind random forest is a simple but powerful one — the wisdom of crowds. In data science speak, the reason that the random forest model works so well is:

A large number of relatively uncorrelated models (trees) operating as a committee will outperform any of the individual constituent models. Each tree in a random forest can pick only from a random subset of features. This forces even more variation amongst the trees in the model and ultimately results in lower correlation across trees and more diversification. So in our random forest, we end up with trees that are not only trained on different sets of data (thanks to bagging) but also use different features to make decisions.

How does it work?

Random forest algorithms have three main **hyperparameters**, which need to be set before training. These include **node size**, **the number of trees**, and **the number of features sampled**. From there, the random forest classifier can be used to solve for regression or classification problems.

The following steps explain the working of a random forest

- 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.

The random forest algorithm is made up of a collection of decision trees, and each tree in the ensemble is comprised of a data sample drawn from a training set with replacement, called the **bootstrap sample**. Of that training sample, one-third of it is set aside as test data, known as the out-of-bag (oob) sample, which we'll come back to later.

[1] In statistics and machine learning, ensemble methods use multiple learning algorithms to obtain better predictive performance than could be obtained from any of the constituent learning algorithms alone, also Ensemble learning is a widely-used and preferred machine learning technique in which multiple individual models, often called base models, are combined to produce an effective optimal prediction model. The Random Forest algorithm is an example of ensemble learning.

[2] A black box model is a system using inputs and outputs to create useful information, without any knowledge of its internal workings.

Bagging, also known as bootstrap aggregation, is the ensemble learning method that is commonly used to reduce variance within a noisy dataset. In bagging, a random sample of data in a training set is selected with replacement—meaning that the individual data points can be chosen more than once.

Bootstrapping is the method of randomly creating samples of data out of a population with replacement to estimate a population parameter.

In R it is done as follows;

```
library(pacman)
p_load(caTools, randomForest, readxl, dplyr)

pima <- read.csv(("E:/Documents/R-Studio Programms/Machine Learning/pima.csv"))
head(pima)
```

```
  Pregnancies Glucose Blood.Pressure Skin.Thickness Insulin  BMI
1           6     148             72             35       0 33.6
2           1      85             66             29       0 26.6
3           8     183             64              0      NA 23.3
4           1      89             66             23      94 28.1
5           0     137             40             35     168 43.1
6           5     116             74              0       0 25.6
  Diabetes.Pedigree Age Outcome
1             0.627  50       1
2             0.351  31       0
3             0.672  32       1
4             0.167  21       0
5             2.288  33       1
6             0.201  30       0
```

```
pima <- pima|>
  mutate(Outcome = factor(Outcome, levels = c(0,1), labels = c("Diabetic", "Not Diabetic")))|>
  na.omit()

glimpse(pima)
```

```
Rows: 752
Columns: 9
$ Pregnancies       <int> 6, 1, 1, 0, 5, 3, 2, 4, 10, 1, 5, 0, 7, 1, 1, 3, 8, ~
$ Glucose           <int> 148, 85, 89, 137, 116, 78, 197, 110, 139, 189, 166, ~
$ Blood.Pressure    <int> 72, 66, 66, 40, 74, 50, 70, 92, 80, 60, 72, 84, 74, ~
$ Skin.Thickness    <int> 35, 29, 23, 35, 0, 32, 45, 0, 0, 23, 19, 47, 0, 38, ~
$ Insulin           <int> 0, 0, 94, 168, 0, 88, 543, 0, 0, 846, 175, 230, 0, 8~
$ BMI               <dbl> 33.6, 26.6, 28.1, 43.1, 25.6, 31.0, 30.5, 37.6, 27.1~
$ Diabetes.Pedigree <dbl> 0.627, 0.351, 0.167, 2.288, 0.201, 0.248, 0.158, 0.1~
$ Age               <int> 50, 31, 21, 33, 30, 26, 53, 30, 57, 59, 51, 31, 31, ~
$ Outcome           <fct> Not Diabetic, Diabetic, Diabetic, Not Diabetic, Diab~
```

```
#Splitting the data into training anf testing data set
set.seed(123)
split.size = .7
sample.size = floor(split.size * nrow(pima))
```

```
indi <- sample(seq_len(nrow(pima)), size = sample.size)

Train <- pima[indi,]
Test <- pima[-indi,]
glimpse(Train)
```

```
Rows: 526
Columns: 9
$ Pregnancies       <int> 1, 7, 11, 8, 7, 3, 3, 6, 1, 1, 5, 2, 6, 2, 5, 1, 5, ~
$ Glucose           <int> 95, 114, 135, 100, 184, 170, 113, 119, 193, 103, 144~
$ Blood.Pressure    <int> 82, 76, 0, 74, 84, 64, 50, 50, 50, 30, 82, 58, 96, 6~
$ Skin.Thickness    <int> 25, 17, 0, 40, 33, 37, 10, 22, 16, 38, 26, 17, 0, 13~
$ Insulin           <int> 180, 110, 0, 215, 0, 225, 85, 176, 375, 83, 285, 265~
$ BMI               <dbl> 35.0, 23.8, 52.3, 39.4, 35.5, 34.5, 29.5, 27.1, 25.9~
$ Diabetes.Pedigree <dbl> 0.233, 0.466, 0.578, 0.661, 0.355, 0.356, 0.626, 1.3~
$ Age               <int> 43, 31, 40, 43, 41, 30, 25, 33, 24, 33, 58, 23, 30, ~
$ Outcome           <fct> Not Diabetic, Diabetic, Not Diabetic, Not Diabetic, ~
```

```
# Fitting a random Forest
set.seed(123)
classifier_RF <- randomForest(x =Train[-9], y = as.factor(Train$Outcome), ntree = 500 )
y_pred <- predict(classifier_RF, newdata = Test[-9])
```

The confusion matrix and the results are as follows;

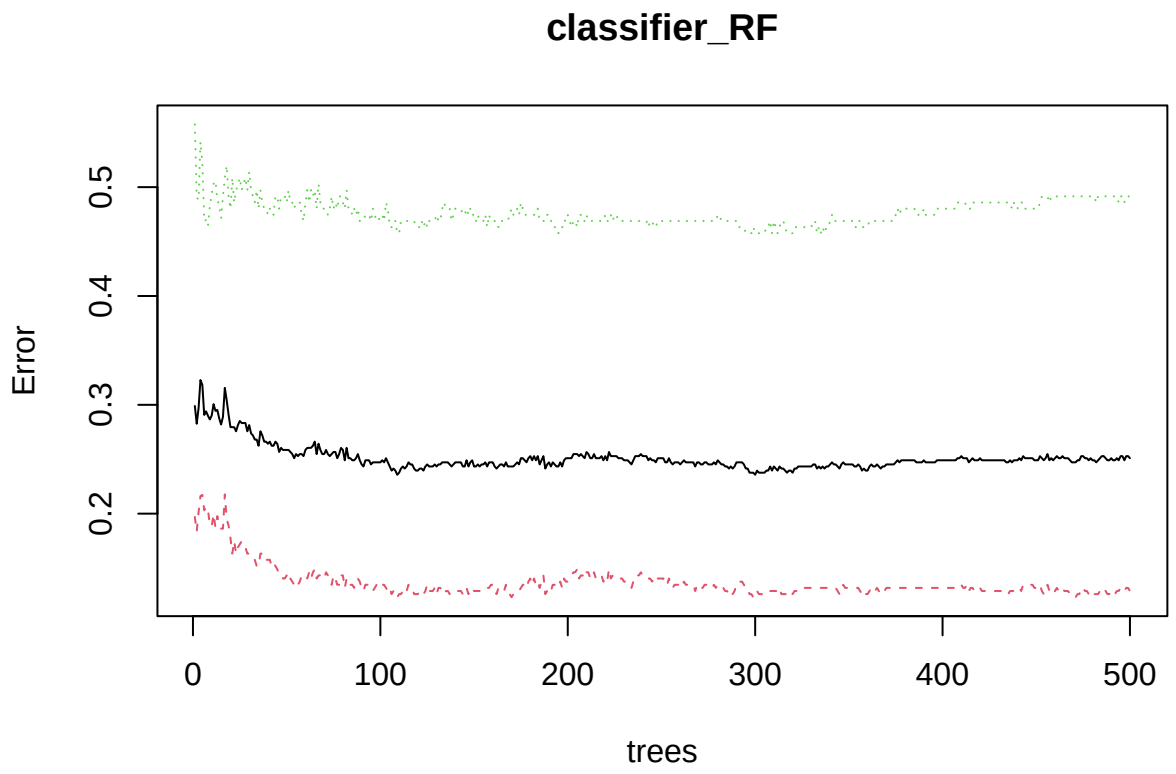
```
classifier_RF
```

```
Call:
 randomForest(x = Train[-9], y = as.factor(Train$Outcome), ntree = 500)
               Type of random forest: classification
                     Number of trees: 500
No. of variables tried at each split: 2

        OOB estimate of  error rate: 25.1%
Confusion matrix:
             Diabetic Not Diabetic class.error
Diabetic          304           45   0.1289398
Not Diabetic       87           90   0.4915254
```

The plot is as follows

```
plot(classifier_RF)
```



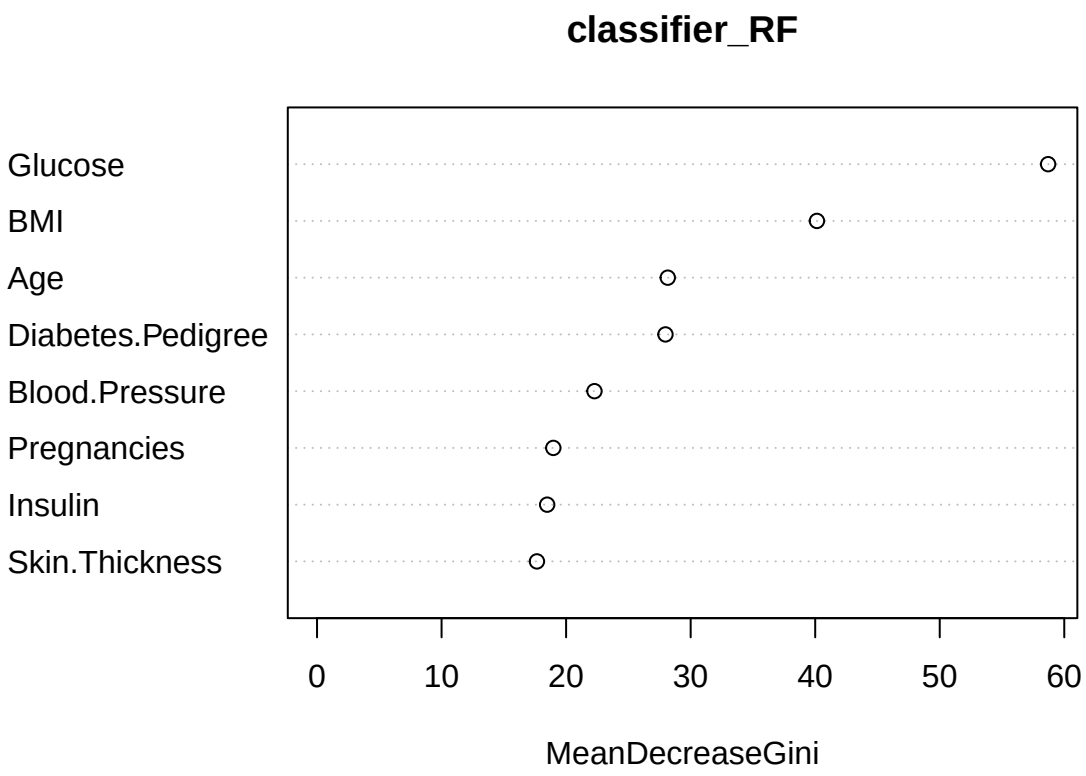
From the plot, the error reduces as the number of trees increases. The important features are;

```
importance(classifier_RF)
```

```
                  MeanDecreaseGini
Pregnancies               18.97238
Glucose                   58.70261
Blood.Pressure            22.27591
Skin.Thickness            17.65588
Insulin                   18.47985
BMI                       40.13914
Diabetes.Pedigree         27.97661
Age                       28.16141
```

The most important or significant feature is Glucose followed by BMI. The least important feature is skin thickness, as can be seen in the plot below;

```
varImpPlot(classifier_RF)
```



The feature or variable importance describes which features are relevant. The higher the value the more important the feature. Feature or variable importance tells which features of our data are most helpful towards our goal, which can be both regression and Classification.

Consider; The Turkish president thinks that high interest rates cause inflation, contrary to the traditional economic approach. For this reason, he dismissed two central bank chiefs within a year. And yes, unfortunately, the central bank officials have limited independence doing their job in Turkey contrary to the rest of the world.

In order to check that we have to model inflation rates with some variables. The most common view of the economic authorities is that the variables affecting the rates are currency exchange rates, and CDS(credit default swap). Of course, we will also add the funding rates variable, the president mentioned, to the model to compare with the other explanatory variables.

Because the variables can be highly correlated with each other, we will prefer the random forest model. This algorithm also has a built-in function to compute the feature importance.

The data I am going tou use has the following variables;

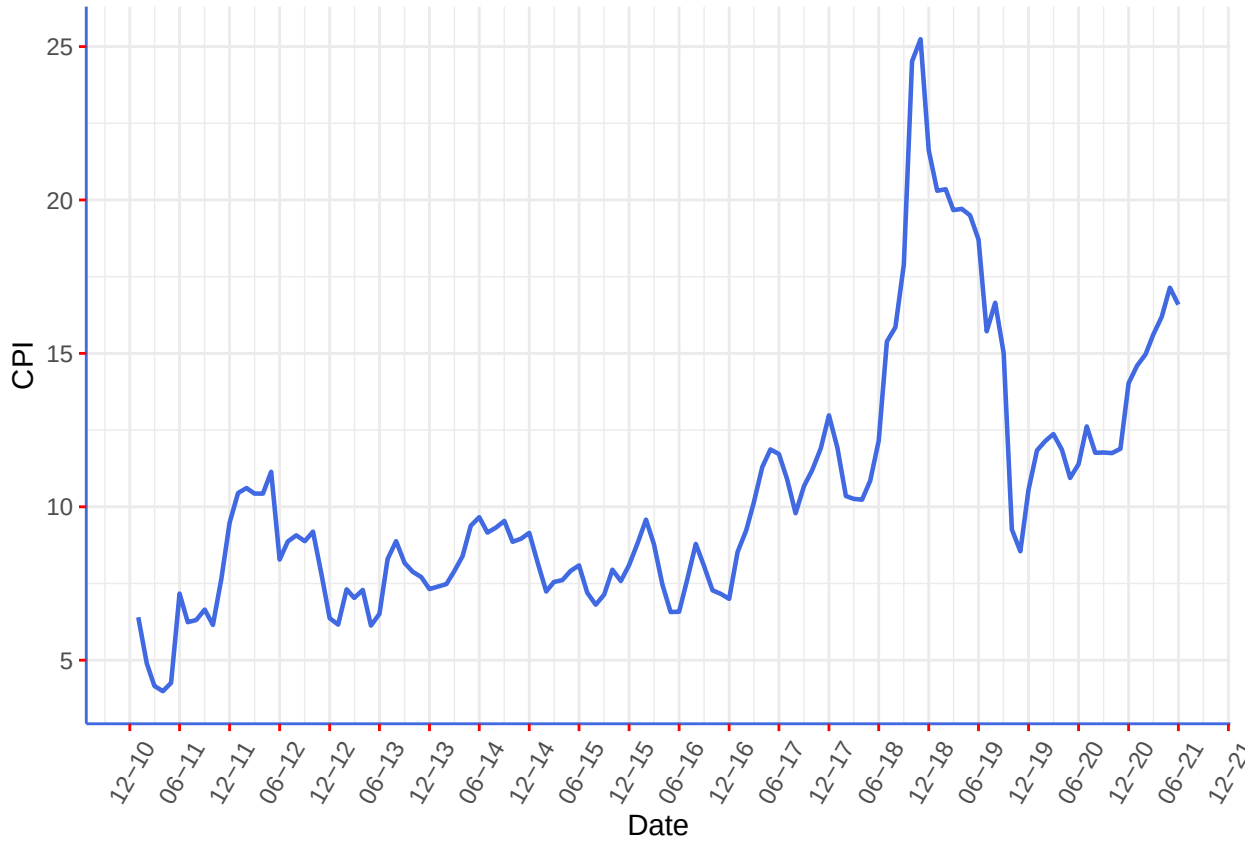
- cpi: The annual consumer price index. It is also called the inflation rate. This is our target variable.
- funding_rate: The one-week repo rate, which determined by the Turkish Central Bank. It is also called the political rate.
- exchange_rate: The currency exchange rates between Turkish Liras and American dollars.
- CDS: The credit defaults swap. It kind of measures the investment risk of a country or company. It is mostly affected by foreign policy developments in Turkey. Although the most popular CDS is 5year, I will take CDS of 1 year USD in this article.

```r
# Importing the data
library(ggplot2)
library(lubridate)
cpi <- read_excel("E:/Documents/R-Studio Programms/Machine Learning/cpi.xlsx")
cpi <- cpi |>
  na.omit()
head(cpi)
```

```
# A tibble: 6 x 5
  date    funding_rate   CPI exchange_rate   CDS
  <chr>          <dbl> <dbl>         <dbl> <dbl>
1 2011-01         6.25  6.4           1.60  62
2 2011-02         6.25  4.9           1.60  62.9
3 2011-03         6.25  4.16          1.55  51.1
4 2011-04         6.25  3.99          1.52  43.0
5 2011-05         6.25  4.26          1.59  61.6
6 2011-06         6.25  7.17          1.62  71.2
```

```r
cpi <- cpi |>
  mutate(date = as.yearmon(date))

# Visualization
cpi|>
  ggplot(aes(x = as.Date(date))) +
  geom_line(aes(y = CPI), col = "royalblue", linewidth = .8)+
  scale_x_date(date_labels = "%m-%y",date_breaks = "6 month") +
  theme_minimal() +
  xlab("Date")+
  theme(axis.line = element_line(color = "royalblue"),axis.text.x = element_text(angle = 60, vjust = 0.
```



Thus the random forest regression [3] is done as follow;

```
cpimodel <- randomForest(x = cpi[-c(1,3)], y = cpi$CPI, ntree = 500, importance = TRUE)
cpimodel
```

```
Call:
 randomForest(x = cpi[-c(1, 3)], y = cpi$CPI, ntree = 500, importance = TRUE)
               Type of random forest: regression
                     Number of trees: 500
No. of variables tried at each split: 1

          Mean of squared residuals: 1.597869
                    % Var explained: 90.61
```

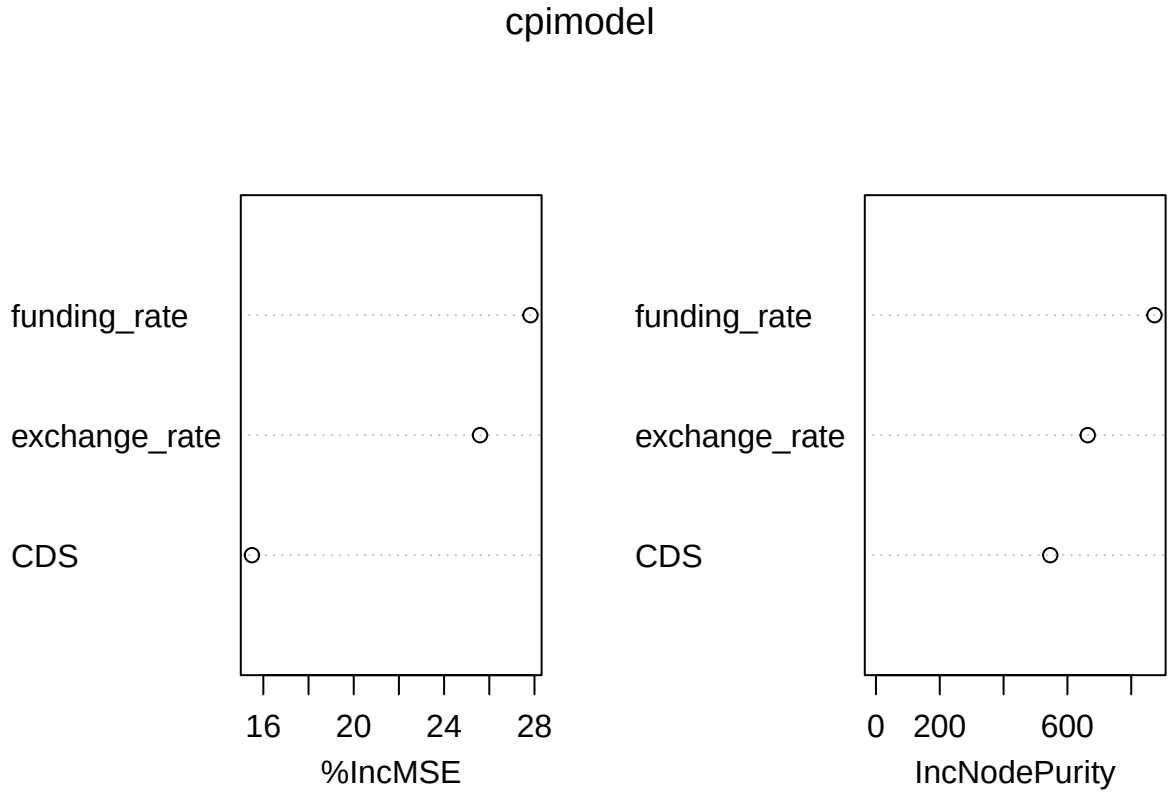
Below is the feature importance plot

```
importance(cpimodel)
```

```
              %IncMSE IncNodePurity
funding_rate  27.82155      872.8025
exchange_rate 25.58946      663.6904
CDS           15.49543      546.2964
```

[3] I the target variable is categorical, then R automatically does a random forest classifier and if the target variable is continous, R does a random forest regressor

```
varImpPlot(cpimodel)
```



When we examine the charts above, we can clearly see that the funding and exchange rates have similar effects on the model and, CDS has significant importance in the behavior of the CPI. So the theory of the president seems to fall short of what he claims.