Random Forests

Vitalii Novikov

Table of contents

Libraries

```
library <- function(...) {suppressPackageStartupMessages(base::library(...))}
if (!require(caret)) install.packages("caret"); library(caret)
```

```
Loading required package: caret
```

```
Loading required package: ggplot2
Loading required package: lattice
```

```r
if (!require(mlbench)) install.packages("mlbench"); library(mlbench)
```

```
Loading required package: mlbench
```

```r
if (!require(dplyr)) install.packages("dplyr"); library(dplyr)
```

```
Loading required package: dplyr


Attaching package: 'dplyr'


The following objects are masked from 'package:stats':

    filter, lag


The following objects are masked from 'package:base':

    intersect, setdiff, setequal, union
```

```r
library(plyr)
library(randomForest)
library(pROC)
```

Starting point

We would like to use Random Forests to predict the diagnosis (Benign or Malignant based on 10 features: - radius (mean of distances from center to points on the perimeter) - texture (standard deviation of gray-scale values) - perimeter - area - smoothness (local variation in radius lengths) - compactness (perimeter^2 / area - 1.0) - concavity (severity of concave portions of the contour) - concave points (number of concave portions of the contour) - symmetry - fractal dimension ("coastline approximation" - 1)

Data management

```r
dat <- read.csv("data.csv")
glimpse(dat)
```

```
Rows: 569
Columns: 33
$ id                      <int> 842302, 842517, 84300903, 84348301, 84358402, ~
$ diagnosis               <chr> "M", "M", "M", "M", "M", "M", "M", "M", "M", "~
$ radius_mean             <dbl> 17.990, 20.570, 19.690, 11.420, 20.290, 12.450~
$ texture_mean            <dbl> 10.38, 17.77, 21.25, 20.38, 14.34, 15.70, 19.9~
$ perimeter_mean          <dbl> 122.80, 132.90, 130.00, 77.58, 135.10, 82.57, ~
$ area_mean               <dbl> 1001.0, 1326.0, 1203.0, 386.1, 1297.0, 477.1, ~
$ smoothness_mean         <dbl> 0.11840, 0.08474, 0.10960, 0.14250, 0.10030, 0~
$ compactness_mean        <dbl> 0.27760, 0.07864, 0.15990, 0.28390, 0.13280, 0~
$ concavity_mean          <dbl> 0.30010, 0.08690, 0.19740, 0.24140, 0.19800, 0~
$ concave.points_mean     <dbl> 0.14710, 0.07017, 0.12790, 0.10520, 0.10430, 0~
$ symmetry_mean           <dbl> 0.2419, 0.1812, 0.2069, 0.2597, 0.1809, 0.2087~
$ fractal_dimension_mean  <dbl> 0.07871, 0.05667, 0.05999, 0.09744, 0.05883, 0~
$ radius_se               <dbl> 1.0950, 0.5435, 0.7456, 0.4956, 0.7572, 0.3345~
$ texture_se              <dbl> 0.9053, 0.7339, 0.7869, 1.1560, 0.7813, 0.8902~
$ perimeter_se            <dbl> 8.589, 3.398, 4.585, 3.445, 5.438, 2.217, 3.18~
$ area_se                 <dbl> 153.40, 74.08, 94.03, 27.23, 94.44, 27.19, 53.~
$ smoothness_se           <dbl> 0.006399, 0.005225, 0.006150, 0.009110, 0.0114~
$ compactness_se          <dbl> 0.049040, 0.013080, 0.040060, 0.074580, 0.0246~
$ concavity_se            <dbl> 0.05373, 0.01860, 0.03832, 0.05661, 0.05688, 0~
$ concave.points_se       <dbl> 0.015870, 0.013400, 0.020580, 0.018670, 0.0188~
$ symmetry_se             <dbl> 0.03003, 0.01389, 0.02250, 0.05963, 0.01756, 0~
$ fractal_dimension_se    <dbl> 0.006193, 0.003532, 0.004571, 0.009208, 0.0051~
$ radius_worst            <dbl> 25.38, 24.99, 23.57, 14.91, 22.54, 15.47, 22.8~
$ texture_worst           <dbl> 17.33, 23.41, 25.53, 26.50, 16.67, 23.75, 27.6~
$ perimeter_worst         <dbl> 184.60, 158.80, 152.50, 98.87, 152.20, 103.40,~
$ area_worst              <dbl> 2019.0, 1956.0, 1709.0, 567.7, 1575.0, 741.6, ~
$ smoothness_worst        <dbl> 0.1622, 0.1238, 0.1444, 0.2098, 0.1374, 0.1791~
$ compactness_worst       <dbl> 0.6656, 0.1866, 0.4245, 0.8663, 0.2050, 0.5249~
$ concavity_worst         <dbl> 0.71190, 0.24160, 0.45040, 0.68690, 0.40000, 0~
$ concave.points_worst    <dbl> 0.26540, 0.18600, 0.24300, 0.25750, 0.16250, 0~
$ symmetry_worst          <dbl> 0.4601, 0.2750, 0.3613, 0.6638, 0.2364, 0.3985~
$ fractal_dimension_worst <dbl> 0.11890, 0.08902, 0.08758, 0.17300, 0.07678, 0~
$ X                       <lgl> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA~
```

```r
nrow(dat)
```

```
[1] 569
```

This dataset include 569 observations. We will only use the "worst-features" for creating neural network.

Pick useful predictors

```r
clean_data <- as_tibble(dat[c(2,23:32)])
clean_data <- mutate(clean_data,
  diagnosis = case_match(diagnosis, "M" ~ "malignant", "B"~ "benign")
   ) %>% mutate(diagnosis = factor(diagnosis))
   # %>% mutate(diagnosis = relevel(diagnosis, ref = "malignant"))
#clean_data[['diagnosis']] <- as.factor(clean_data[['diagnosis']])
clean_data |> head(2)
```

```
# A tibble: 2 x 11
  diagnosis radius_worst texture_worst perimeter_worst area_worst
  <fct>            <dbl>         <dbl>           <dbl>      <dbl>
1 malignant         25.4          17.3            185.       2019
2 malignant         25.0          23.4            159.       1956
# i 6 more variables: smoothness_worst <dbl>, compactness_worst <dbl>,
#   concavity_worst <dbl>, concave.points_worst <dbl>, symmetry_worst <dbl>,
#   fractal_dimension_worst <dbl>
```

```r
clean_data |> apply(2, function(x) sum(is.na(x)))
```

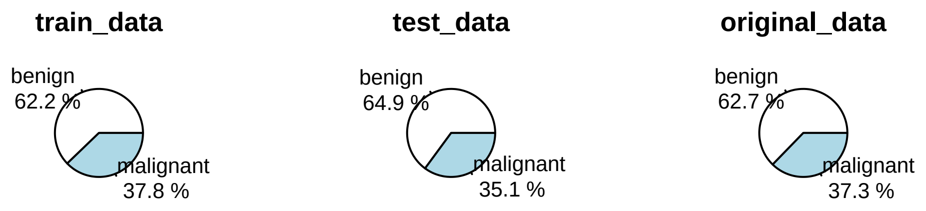
```
              diagnosis            radius_worst           texture_worst
                      0                       0                       0
        perimeter_worst              area_worst        smoothness_worst
                      0                       0                       0
      compactness_worst         concavity_worst    concave.points_worst
                      0                       0                       0
         symmetry_worst fractal_dimension_worst
                      0                       0
```

There is no missing data, good.

Stratified data set

Using 'sample' function

```
set.seed(230)
index <- sample(1:nrow(clean_data),round(0.8*nrow(clean_data)))
train_data <- clean_data[index,]
test_data <- clean_data[-index,]

pie_with_percentages <- function(diag_list, main = "Diagnosis", round = 1) {
  diag_table <- table(diag_list)
  diag_percentages <- prop.table(diag_table) * 100
  labels <- paste(names(diag_table), "\n", round(diag_percentages, round), "%")
  return(pie(diag_table, labels = labels, main = main))
}

layout(matrix(c(1,2,3), nrow = 1), respect = TRUE)
pie_with_percentages(train_data$diagnosis, main = "train_data")
pie_with_percentages(test_data$diagnosis, main = "test_data")
pie_with_percentages(clean_data$diagnosis, main = "original_data")
```

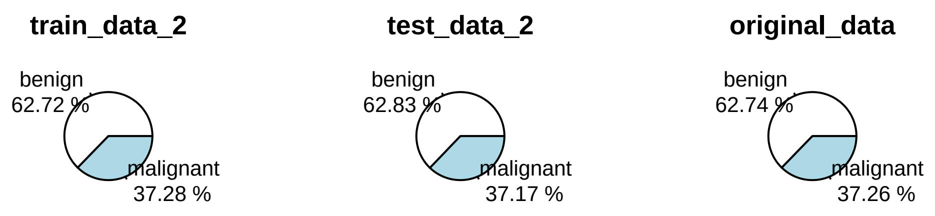


```r
layout(1)
```

The test data show slightly lowest proportion of malignant then both train and original data, but it is still very close to actual picture.

Using caret library

```r
set.seed(123)
index_2 <- createDataPartition(clean_data$diagnosis, p = 0.8, list = FALSE)

train_data_2 <- clean_data[index_2,]
test_data_2 <- clean_data[-index_2,]

layout(matrix(c(1,2,3), nrow = 1), respect = TRUE)
pie_with_percentages(train_data_2$diagnosis, main = "train_data_2", round = 2)
pie_with_percentages(test_data_2$diagnosis, main = "test_data_2", round = 2)
pie_with_percentages(clean_data$diagnosis, main = "original_data", round = 2)
```



In this case we got really stratified train and test split of the data, because the proportions of classes are almost the same in each dataset. For further steps we will use this split of the data.

```
trainD <- train_data_2
testD <- test_data_2

split_description <- t(rbind(
  data.frame("trainD" = nrow(trainD), "testD" = nrow(testD)),
  c(round(nrow(trainD)/nrow(clean_data)*100,2),
    round(nrow(testD)/nrow(clean_data)*100,2))
))
colnames(split_description) <- c("nrow","percentage (%)")
split_description
```

```
       nrow percentage (%)
trainD  456          80.14
testD   113          19.86
```

The trainD include 456 rows, which is 80.14% of the original dataset.

Normalize the data

All predictors have their own scales. We should perform min-max-normalization.

```
trainD |> summary()
```

```
     diagnosis     radius_worst    texture_worst   perimeter_worst 
 benign   :286   Min.   : 7.93   Min.   :12.02   Min.   : 50.41  
 malignant:170   1st Qu.:12.87   1st Qu.:21.05   1st Qu.: 83.73  
                 Median :14.91   Median :25.27   Median : 97.26  
                 Mean   :16.19   Mean   :25.65   Mean   :106.72  
                 3rd Qu.:18.38   3rd Qu.:29.91   3rd Qu.:124.15  
                 Max.   :36.04   Max.   :47.16   Max.   :251.20  
   area_worst      smoothness_worst  compactness_worst concavity_worst 
 Min.   : 185.2   Min.   :0.07117   Min.   :0.02729   Min.   :0.0000  
 1st Qu.: 509.4   1st Qu.:0.11650   1st Qu.:0.14010   1st Qu.:0.1081  
 Median : 684.0   Median :0.13145   Median :0.21165   Median :0.2112  
 Mean   : 875.0   Mean   :0.13234   Mean   :0.25314   Mean   :0.2691  
 3rd Qu.:1035.0   3rd Qu.:0.14600   3rd Qu.:0.33145   3rd Qu.:0.3814  
 Max.   :4254.0   Max.   :0.21840   Max.   :1.05800   Max.   :1.1700  
 concave.points_worst symmetry_worst   fractal_dimension_worst
 Min.   :0.00000      Min.   :0.1565   Min.   :0.05504        
 1st Qu.:0.06330      1st Qu.:0.2478   1st Qu.:0.07190        
 Median :0.09766      Median :0.2813   Median :0.08009
 Mean   :0.11275      Mean   :0.2881   Mean   :0.08430
 3rd Qu.:0.16025      3rd Qu.:0.3174   3rd Qu.:0.09192
 Max.   :0.29030      Max.   :0.6638   Max.   :0.20750
```

Scaled train dataset

```
maxs <- apply(trainD[-1], 2, max)
mins <- apply(trainD[-1], 2, min)

scaled_trainD <- scale(trainD[-1], center = mins, scale = maxs - mins) %>%
  cbind(trainD[1])

summary(scaled_trainD)
```

```
  radius_worst    texture_worst    perimeter_worst    area_worst
 Min.   :0.0000   Min.   :0.0000   Min.   :0.0000   Min.   :0.00000
 1st Qu.:0.1758   1st Qu.:0.2570   1st Qu.:0.1659   1st Qu.:0.07969
 Median :0.2483   Median :0.3769   Median :0.2333   Median :0.12258
 Mean   :0.2937   Mean   :0.3878   Mean   :0.2804   Mean   :0.16952
 3rd Qu.:0.3716   3rd Qu.:0.5092   3rd Qu.:0.3672   3rd Qu.:0.20886
 Max.   :1.0000   Max.   :1.0000   Max.   :1.0000   Max.   :1.00000
 smoothness_worst compactness_worst concavity_worst   concave.points_worst
 Min.   :0.0000   Min.   :0.0000    Min.   :0.00000   Min.   :0.0000
 1st Qu.:0.3079   1st Qu.:0.1094    1st Qu.:0.09237   1st Qu.:0.2181
 Median :0.4094   Median :0.1789    Median :0.18056   Median :0.3364
 Mean   :0.4155   Mean   :0.2191    Mean   :0.23002   Mean   :0.3884
 3rd Qu.:0.5083   3rd Qu.:0.2951    3rd Qu.:0.32598   3rd Qu.:0.5520
 Max.   :1.0000   Max.   :1.0000    Max.   :1.00000   Max.   :1.0000
 symmetry_worst   fractal_dimension_worst     diagnosis
 Min.   :0.0000   Min.   :0.0000          benign   :286
 1st Qu.:0.1799   1st Qu.:0.1106          malignant:170
 Median :0.2461   Median :0.1643
 Mean   :0.2595   Mean   :0.1919
 3rd Qu.:0.3172   3rd Qu.:0.2419
 Max.   :1.0000   Max.   :1.0000
```

Now we have all predictors in the same scale.

Scaled test dataset

```
maxs <- apply(testD[-1], 2, max)
mins <- apply(testD[-1], 2, min)

scaled_testD <- scale(testD[-1], center = mins, scale = maxs - mins) %>%
  cbind(testD[1])
```

Modeling part

First Random Forest

The initial model include 20 trees and has 3 variables randomly sampled as candidates at each split (floor(sqrt(10)) = 3).

```
set.seed(333)
rf_model <- randomForest(diagnosis ~ .,
                         data = scaled_trainD,
                         ntree = 20,
                         mtry = floor(sqrt(10))
                         )
```

```
predictions <- predict(rf_model, newdata = scaled_testD)
prob_predictions <- predict(rf_model, newdata = scaled_testD, type = "prob")[,2]
print(confusionMatrix(predictions, scaled_testD$diagnosis, mode = "prec_recall"))
```

```
Confusion Matrix and Statistics

           Reference
Prediction  benign malignant
  benign        67         1
  malignant      4        41

               Accuracy : 0.9558
                 95% CI : (0.8998, 0.9855)
    No Information Rate : 0.6283
    P-Value [Acc > NIR] : <2e-16

                  Kappa : 0.9066
```

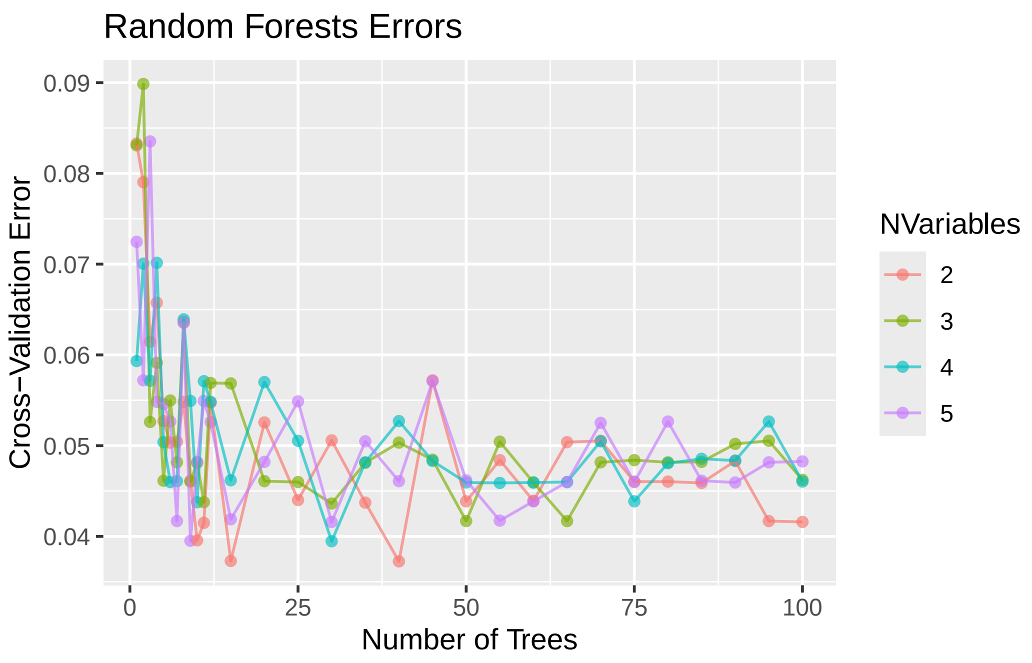
```
 Mcnemar's Test P-Value : 0.3711

              Precision : 0.9853
                 Recall : 0.9437
                     F1 : 0.9640
             Prevalence : 0.6283
         Detection Rate : 0.5929
   Detection Prevalence : 0.6018
      Balanced Accuracy : 0.9599

       'Positive' Class : benign
```

The initial model performed not so bad, just 5 observations were wrong classified. Therefore the Accuracy is about 96%.

Hyperparameters tuning

As Hyperparameters we have: 1) number of trees in the forest 2) number of variables randomly sampled as candidates at each split (it is recommended to take around sqrt(N_predictors), but it is interesting to explore how it affect the model)

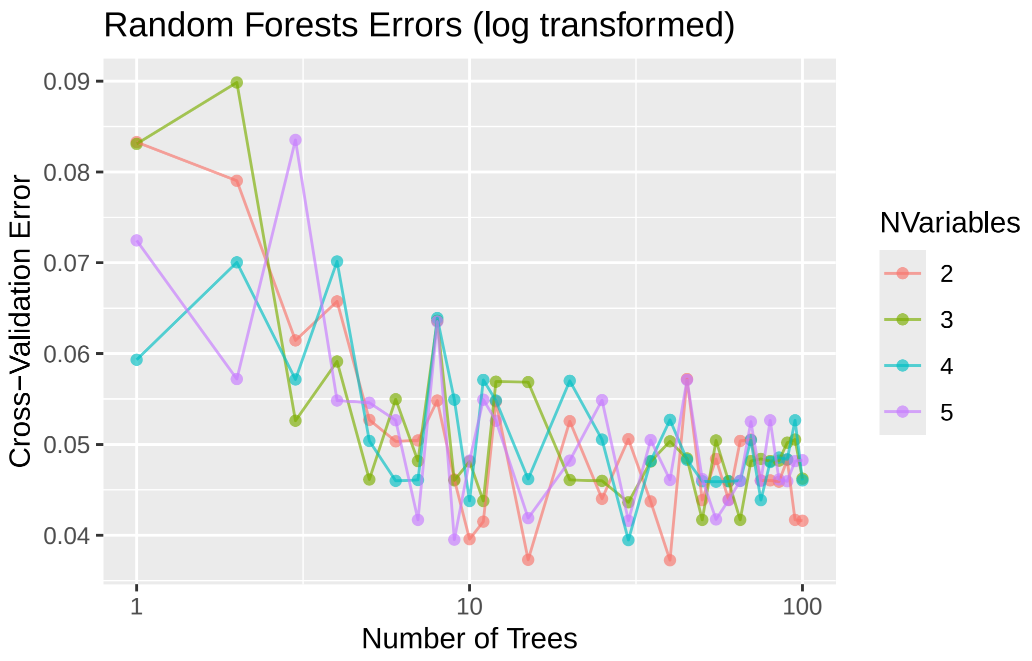
```r
set.seed(1)
n_trees <- c(1:12,seq(15, 100, by = 5))
error_df <- data.frame(Trees = numeric(), NVariables = factor(), Error = numeric())

control <- trainControl(method = "cv", number = 10, classProbs = TRUE, savePredictions = "fi

for (ntree in n_trees) {
  for (nv_count in 2:5) {## number of variables randomly sampled as candidates at each split
    rf_model <- train(diagnosis ~ ., data = scaled_trainD,
                      method = "rf",
                      trControl = control,
                      tuneGrid = data.frame(mtry = nv_count),
                      ntree = ntree)
    # Collect cross-validated errors
    error_df <- rbind(error_df,
                      data.frame(Trees = ntree,
                                 NVariables = as.character(nv_count),
                                 Error = 1 - max(rf_model$results$Accuracy)))
}}


ggplot(error_df, aes(x = Trees, y = Error,  color = NVariables)) +
  geom_point(alpha = 0.65) +
  geom_line(alpha = 0.65) +
  #scale_x_log10()+
  labs(title = "Random Forests Errors",
       x = "Number of Trees",
       y = "Cross-Validation Error")
```



There is no special patterns in differences between number of variables sampled as candidates at each split. May be "3" is a little bit more stable, but still the errors line fluctuates between 0.04 and 0.05.

To easier interpret the start of the plateau we should use log-transformed x-axis.

```
ggplot(error_df, aes(x = Trees, y = Error,  color = NVariables)) +
  geom_point(alpha = 0.65) +
  geom_line(alpha = 0.65) +
  scale_x_log10()+
  labs(title = "Random Forests Errors (log transformed)",
       x = "Number of Trees",
       y = "Cross-Validation Error")
```



The plateau begins around the number of trees equals 5, but there is a strange increase of errors for all models when number of trees equals 8.

When the model have 5 trees, the lowest error has the model with number of sampled variables = 3. Therefore, this model will be used in further steps.

Final Model

As described before, cross validation tell us that the appropriate number of trees is 5 and the appropriate number of sampled variables is 3.

```
set.seed(55)
final_model <- randomForest(diagnosis ~ .,
                          data = scaled_trainD,
                          ntree = 5,
                          mtry = 3)

predictions <- predict(final_model, newdata = scaled_testD)
prob_predictions <- predict(final_model, newdata = scaled_testD, type = "prob")[,2]
print(confusionMatrix(predictions, scaled_testD$diagnosis, mode = "prec_recall"))
```

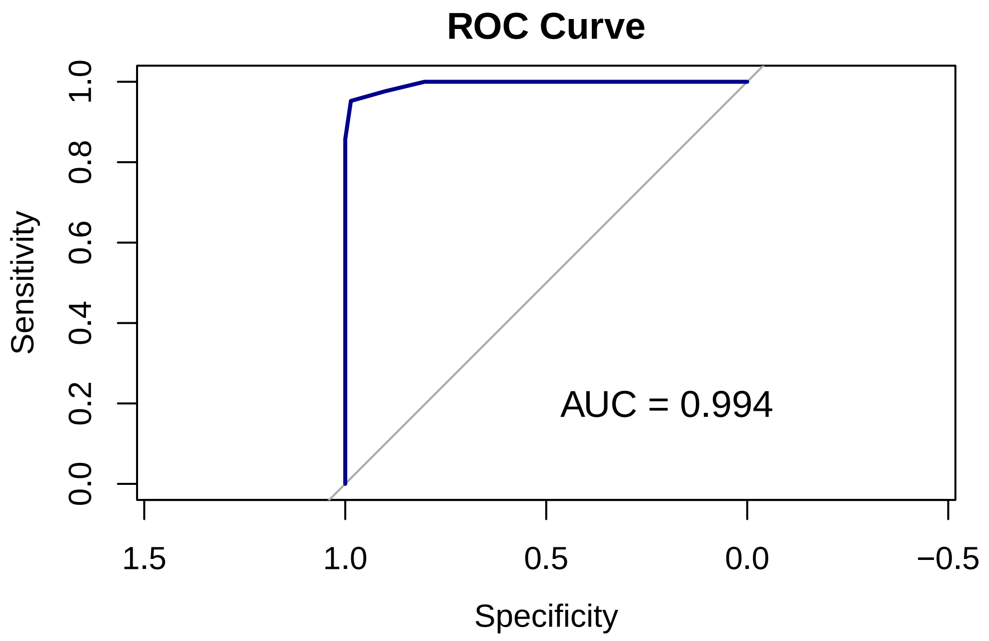
```
Confusion Matrix and Statistics

           Reference
Prediction  benign malignant
  benign        70         2
  malignant      1        40

               Accuracy : 0.9735
                 95% CI : (0.9244, 0.9945)
    No Information Rate : 0.6283
    P-Value [Acc > NIR] : <2e-16

                  Kappa : 0.9429

 Mcnemar's Test P-Value : 1

              Precision : 0.9722
                 Recall : 0.9859
                     F1 : 0.9790
             Prevalence : 0.6283
         Detection Rate : 0.6195
   Detection Prevalence : 0.6372
      Balanced Accuracy : 0.9691

       'Positive' Class : benign
```

The final model performed not bad, just 3 observations were wrong classified. Therefore the Balanced Accuracy around 97%. According to confusion matrix in 2 cases the malignant were classified as benign, which is bad.

```r
roc_obj <- roc(scaled_testD$diagnosis, prob_predictions,
               levels = c("benign", "malignant"))
```

```
Setting direction: controls < cases
```

```r
plot(roc_obj, col = "darkblue", main = "ROC Curve")
text(0.2, 0.2, labels = paste0("AUC = ", round(auc(roc_obj), 3)), cex = 1.2)
```



ROC Curve looks almost perfect with AUC = 0.994. There could be some improvements, because the TPR (sensitivity) become 1 at about 0.8 of specificity.

Possible improvements

The final model is already quite good in terms of detecting "malignant". If we want to detect all of the malignant items we can change the threshold to achieve the appropriate results, but it can lead to increase of False Positive predictions, which sometimes is ok. Also, the possible solution could be the "unsure" status for some range of probabilities. But for better calculating of percentages it may be necessary to use larger number of trees in the random forest.

And of course increasing of the number of observations in the initial dataset can help to train better model.