Prostate Cancer Prediction Project

Harvardx: PH125.9x Data Science

Frederico de Almeida Meirelles Palma

10 January, 2020

Contents

Chapter 1

Project Overview

The idea of this project is apply machine learning techniques learned in HarvardX: PH125.9x Data Science course choosing a publicity data.

Chapter 2

Introduction

The objective is to train a machine learning algorithm using the inputs in one subset (training) and then check in the validation set (testing). The chosed dataset is from Kaggle, the subject is prostate cancer and it can be found here: https://www.kaggle.com/sajidsaifi/prostate-cancer

2.1 Dataset

```
if(!require(tidyverse)) install.packages("tidyverse", repos = "http://cran.us.r-project.org")
if(!require(caret)) install.packages("caret", repos = "http://cran.us.r-project.org")
if(!require(dplyr)) install.packages("dplyr", repos = "http://cran.us.r-project.org")
if(!require(ggplot2)) install.packages("ggplot2", repos = "http://cran.us.r-project.org")
if(!require(funModeling)) install.packages("funModeling", repos = "http://cran.us.r-project.org")
if(!require(corrplot)) install.packages("recorrplotadr", repos = "http://cran.us.r-project.org")
if(!require(randomForest)) install.packages("randomForest", repos = "http://cran.us.r-project.org")
if(!require(nnet)) install.packages("nnet", repos = "http://cran.us.r-project.org")

# Prostate Cancer dataset:
# https://www.kaggle.com/sajidsaifi/prostate-cancer #source
# https://github.com/fredpalma/machine-learning/blob/master/prostateCancer/Prostate_Cancer.csv
# The file will be loaded from my github account

# The data
data <- read.csv("https://raw.githubusercontent.com/fredpalma/machine-learning/master/prostateCancer/Pr

library("ggplot2")
library("dplyr")
library("corrplot")
library("funModeling")
```

Chapter 3

Data Analysis

3.1 Data features

The dataset is in csv format and have the following features:

Attributes: 1. id 2. diagnosis_result (M = malignant, B = benign)

Features: 3. radius 4. texture 5. perimeter 6. area 7. smoothness 8. compactness 9. symmetry 10. fractal_dimension

3.2 Exploring the dataset

The dataset has 100 observations and 10 variables.

```
dim(data)
```

```
## [1] 100  10
```

Head of the dataset:

```
head(data)
```

```
##   id diagnosis_result radius texture perimeter area smoothness compactness
## 1  1                M     23      12       151  954      0.143       0.278
## 2  2                B      9      13       133 1326      0.143       0.079
## 3  3                M     21      27       130 1203      0.125       0.160
## 4  4                M     14      16        78  386      0.070       0.284
## 5  5                M      9      19       135 1297      0.141       0.133
## 6  6                B     25      25        83  477      0.128       0.170
##   symmetry fractal_dimension
## 1    0.242             0.079
## 2    0.181             0.057
## 3    0.207             0.060
## 4    0.260             0.097
## 5    0.181             0.059
## 6    0.209             0.076
```

Data Summary:

```
summary(data)
```

```
##        id         diagnosis_result     radius         texture
```

```
##  Min.   :  1.00   B:38            Min.   : 9.00   Min.   :11.00
##  1st Qu.: 25.75   M:62            1st Qu.:12.00   1st Qu.:14.00
##  Median : 50.50                   Median :17.00   Median :17.50
##  Mean   : 50.50                   Mean   :16.85   Mean   :18.23
##  3rd Qu.: 75.25                   3rd Qu.:21.00   3rd Qu.:22.25
##  Max.   :100.00                   Max.   :25.00   Max.   :27.00
##    perimeter          area          smoothness      compactness
##  Min.   : 52.00   Min.   : 202.0   Min.   :0.0700   Min.   :0.0380
##  1st Qu.: 82.50   1st Qu.: 476.8   1st Qu.:0.0935   1st Qu.:0.0805
##  Median : 94.00   Median : 644.0   Median :0.1020   Median :0.1185
##  Mean   : 96.78   Mean   : 702.9   Mean   :0.1027   Mean   :0.1267
##  3rd Qu.:114.25   3rd Qu.: 917.0   3rd Qu.:0.1120   3rd Qu.:0.1570
##  Max.   :172.00   Max.   :1878.0   Max.   :0.1430   Max.   :0.3450
##     symmetry      fractal_dimension
##  Min.   :0.1350   Min.   :0.05300
##  1st Qu.:0.1720   1st Qu.:0.05900
##  Median :0.1900   Median :0.06300
##  Mean   :0.1932   Mean   :0.06469
##  3rd Qu.:0.2090   3rd Qu.:0.06900
##  Max.   :0.3040   Max.   :0.09700
```

No missing values and the diagnosis results shows 62 malign and 38 benign cases.

```
map(data, function(.x) sum(is.na(.x)))
```

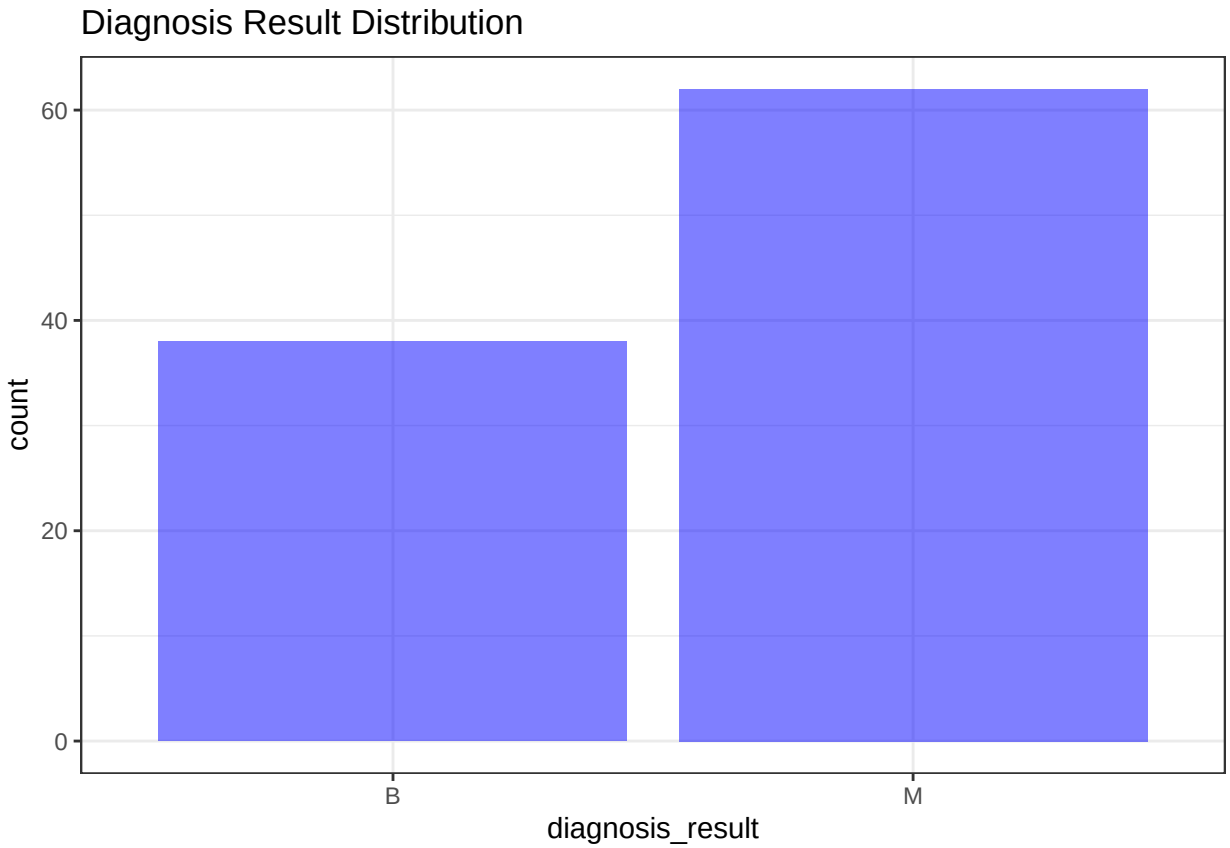
```
## $id
## [1] 0
##
## $diagnosis_result
## [1] 0
##
## $radius
## [1] 0
##
## $texture
## [1] 0
##
## $perimeter
## [1] 0
##
## $area
## [1] 0
##
## $smoothness
## [1] 0
##
## $compactness
## [1] 0
##
## $symmetry
## [1] 0
##
## $fractal_dimension
## [1] 0
```

Let's analise the proportions:

```
prop.table(table(data$diagnosis_result))
```

```
##
##    B    M
## 0.38 0.62
```

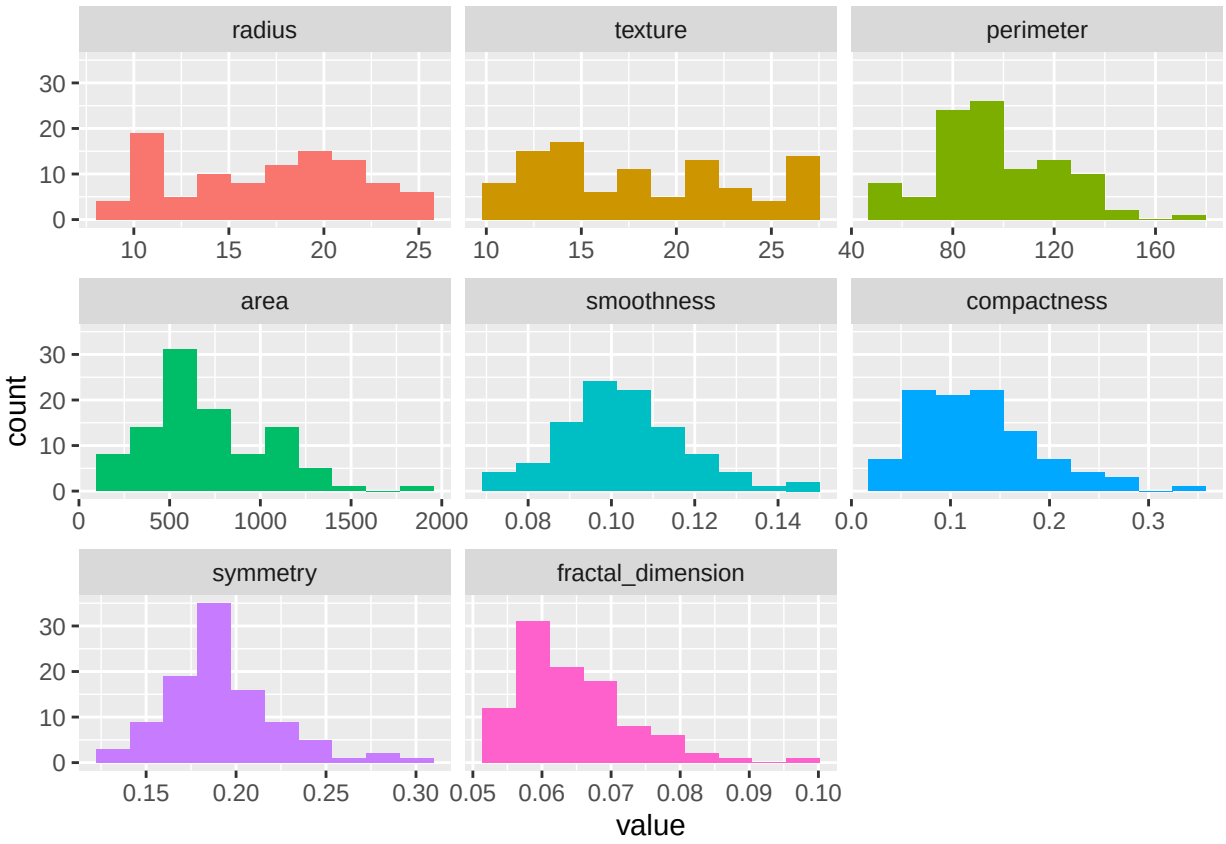
```
options(repr.plot.width=5, repr.plot.height=5)
ggplot(data, aes(x=diagnosis_result))+geom_bar(fill="blue",alpha=0.5)+theme_bw()+labs(title="Diagnosis
```



As we can see the diagnosis result is slightly unbalanced.

The most variables of the dataset are normally distributed as we can see:

```
plot_num(data %>% select(-id), bins=10)
```



Removing ID column from dataset:

```
data2 <- data %>%select(-id)
ncol(data2)
```

```
## [1] 9
```

Chapter 4

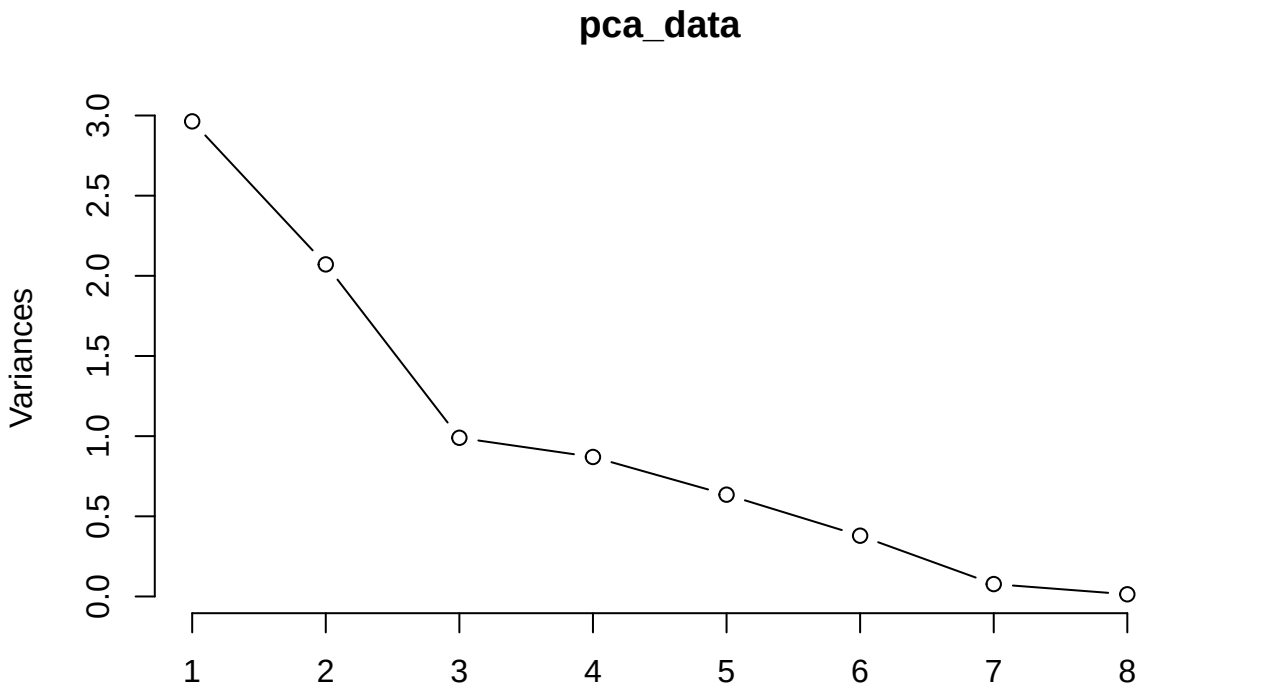
Model Development

4.1 PCA and LDA

4.1.1 Principal Component Analysis (PCA).

The objective to use Principal Component Analysis (PCA) is to reduce the dimensionality of a dataset and still mantains the most of the information of the original set.

```
pca_data <- prcomp(data2[,2:ncol(data2)], center = TRUE, scale = TRUE)
plot(pca_data, type="l")
```



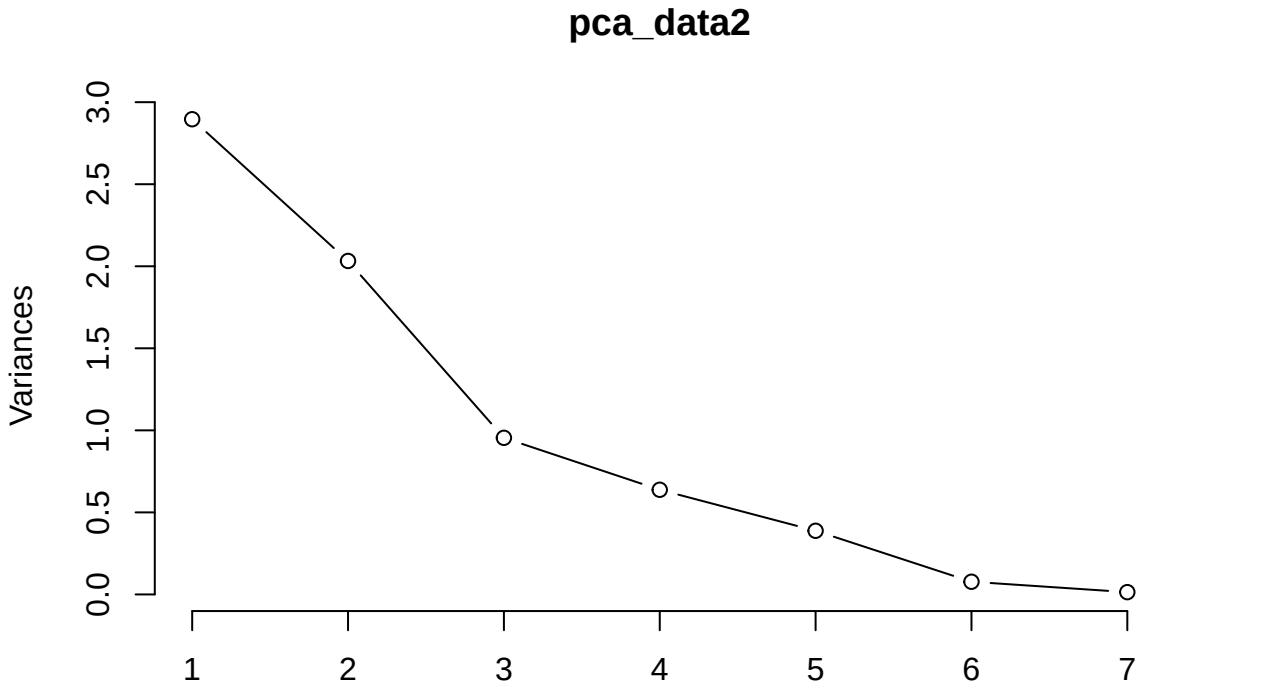

```
summary(pca_data)
```

```
## Importance of components:
##                           PC1    PC2    PC3    PC4     PC5     PC6     PC7
## Standard deviation     1.7214 1.4392 0.9950 0.9328 0.79687 0.61575 0.27816
## Proportion of Variance 0.3704 0.2589 0.1237 0.1088 0.07938 0.04739 0.00967
## Cumulative Proportion  0.3704 0.6293 0.7531 0.8618 0.94122 0.98861 0.99828
```

```
##                            PC8
## Standard deviation     0.11722
## Proportion of Variance 0.00172
## Cumulative Proportion  1.00000
```

As we can see, the two first components explains the 0.629 of the variance, using 5 principal components we have more than 0.94 and using 7 components we have more than 0.99 of the variance.

```
pca_data2 <- prcomp(data2[,3:ncol(data2)], center = TRUE, scale = TRUE)
plot(pca_data2, type="l")
```




```
summary(pca_data2)
```

```
## Importance of components:
##                           PC1    PC2    PC3     PC4     PC5     PC6    PC7
## Standard deviation     1.7018 1.4256 0.9770 0.79864 0.62287 0.27818 0.1183
## Proportion of Variance 0.4137 0.2903 0.1363 0.09112 0.05542 0.01105 0.0020
## Cumulative Proportion  0.4137 0.7040 0.8404 0.93152 0.98694 0.99800 1.0000
```

Using pca_data2, we have more than 98% of the variance using 5 Principal Components.

4.1.2 Linear Discriminant Analysis (LDA)

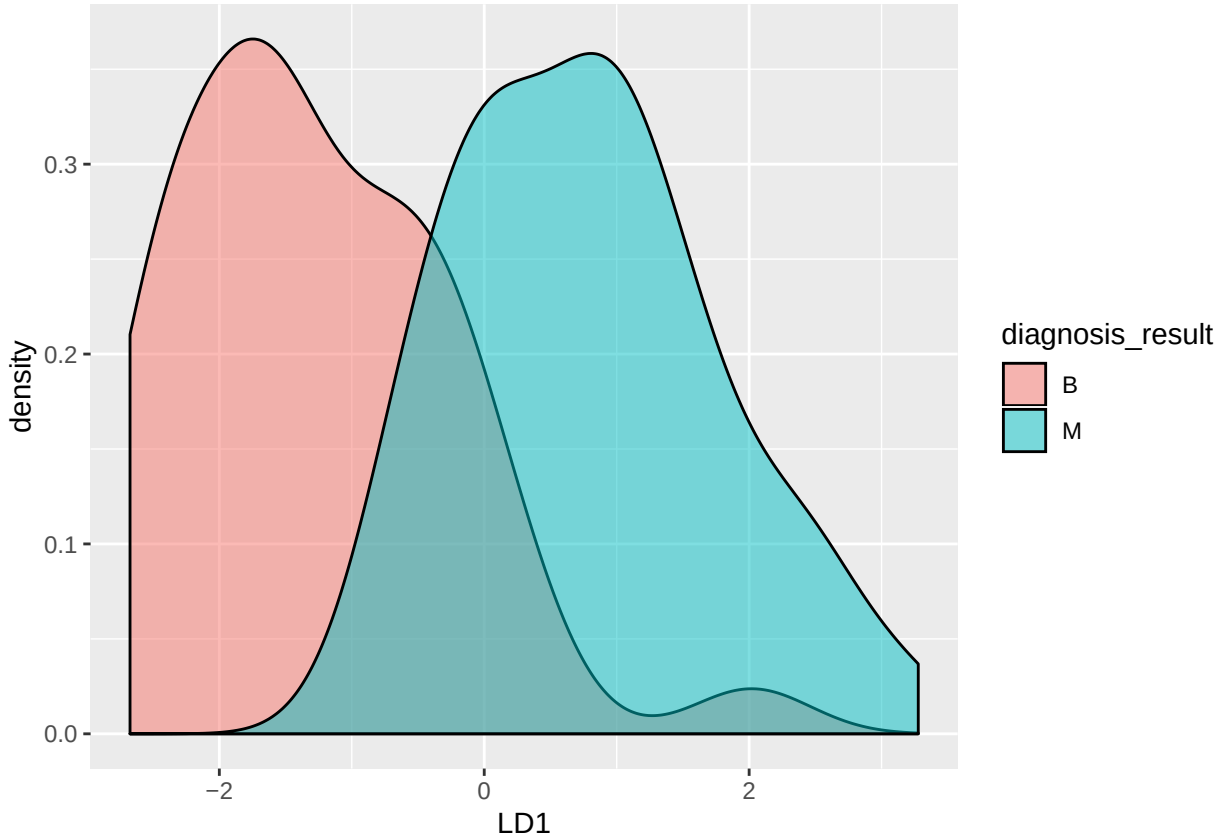
Another option is to use the Linear Discriminant Analysis (LDA) instead of PCA. LDA takes in consideration the different classes and could get better results.

```
lda_data <- MASS::lda(diagnosis_result~., data = data, center = TRUE, scale = TRUE)
lda_data
```

```
## Call:
## lda(diagnosis_result ~ ., data = data, center = TRUE, scale = TRUE)
##
## Prior probabilities of groups:
##    B    M
## 0.38 0.62
```

```
##
## Group means:
##         id   radius  texture perimeter     area smoothness compactness
## B 62.18421 17.94737 17.76316   78.5000 474.3421 0.09905263  0.08689474
## M 43.33871 16.17742 18.51613  107.9839 842.9516 0.10498387  0.15109677
##    symmetry fractal_dimension
## B 0.1840526        0.06460526
## M 0.1987581        0.06474194
##
## Coefficients of linear discriminants:
##                              LD1
## id                -1.317990e-02
## radius            -1.430760e-02
## texture            5.684457e-02
## perimeter          1.913916e-02
## area              -5.524215e-04
## smoothness        -8.083529e+00
## compactness        2.362519e+01
## symmetry          -5.483158e+00
## fractal_dimension -1.037989e+02
```

```
lda_predict <- predict(lda_data, data)$x %>% as.data.frame() %>% cbind(diagnosis_result=data$diagnosis_

ggplot(lda_predict, aes(x=LD1, fill=diagnosis_result)) + geom_density(alpha=0.5)
```



4.2 Creating training and testing datasets

Let's create the training and testing set using 80% to train and 20% to test then by building machine learning classification models with the objective it to predict whether is benign or malign.

```
#set.seed(3) # if using R 3.5 or earlier
set.seed(3, sample.kind = "Rounding")    # if using R 3.6 or later
data3 <- cbind (diagnosis_result=data$diagnosis_result, data2)
data_sampling_index <- createDataPartition(data$diagnosis, times=1, p=0.8, list = FALSE)
training <- data3[data_sampling_index, ]
testing <- data3[-data_sampling_index, ]


fitControl <- trainControl(method="cv",
                           number = 15,    #Either the number of folds or number of resampling iteratio
                           classProbs = TRUE,
                           summaryFunction = twoClassSummary)
```

4.3 Metrics description

Beforer apply some models follow the metrics description that we will use to compare them:

Accuracy is the number of correct predictions made divided by the total number of predictions made, multiplied by 100 to turn it into a percentage.

Precision or Positive Predictive Value (PPV) is the number of True Positives divided by the number of True Positives and False Positives. A low precision can also indicate a large number of False Positives.

Recall (Sensitivity) or True Positive Rate is the number of True Positives divided by the number of True Positives and the number of False Negatives.Recall can be thought of as a measure of a classifiers completeness. A low recall indicates many False Negatives.

F1 Score (F Score) or F Measure is the 2 x ((precision x recall) / (precision + recall)). The F1 score conveys the balance between the precision and the recall.

4.4 Naive Bayes Model

The Naive Bayesian classifier is based on applying Bayes' theorem with strong naive independence assumptions between the features.

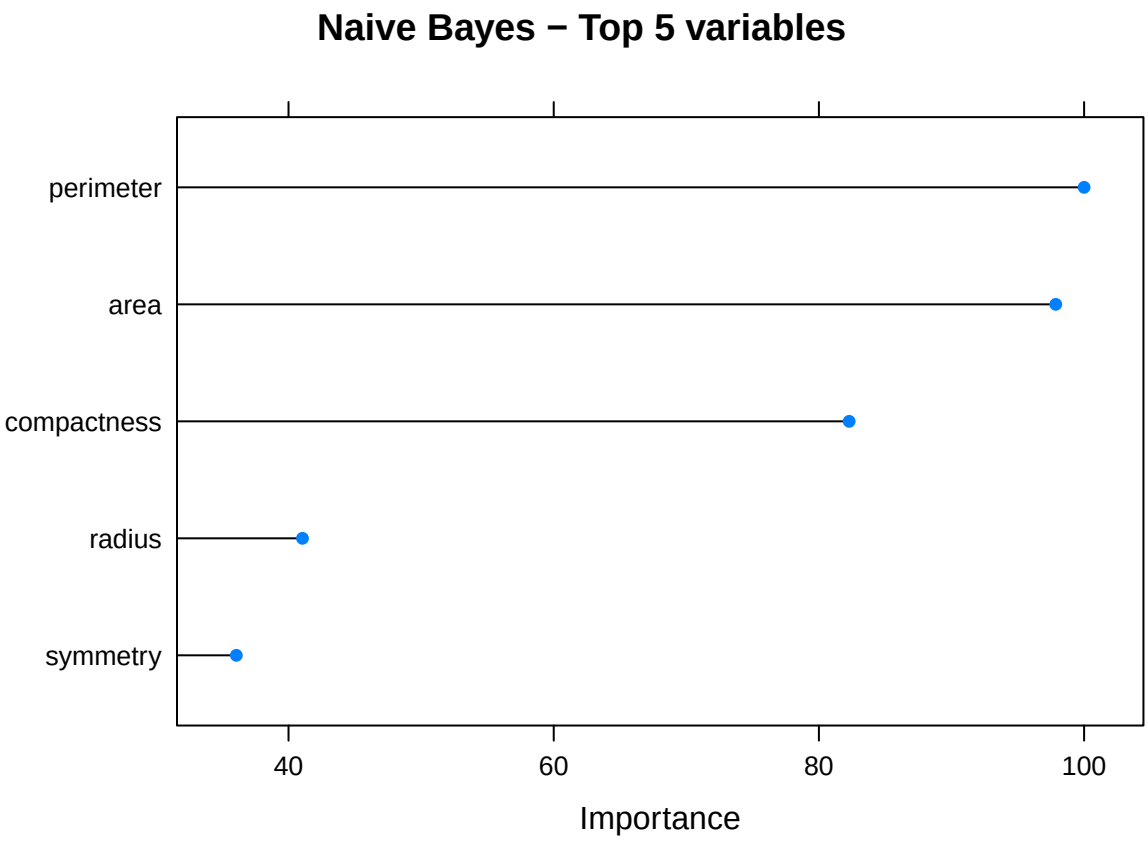
```
naive <- train(diagnosis_result~.,
                      training,
                      method="nb",
                      metric="ROC",
                      preProcess=c('center', 'scale'), #in order to normalize the data
                      trace=FALSE,
                      trControl=fitControl)

naive_pred <- predict(naive, testing)
confusionmatrix_naive <- confusionMatrix(naive_pred, testing$diagnosis_result, positive = "M")
confusionmatrix_naive
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  B  M
```

```
##          B  0  1
##          M  7 11
##
##                Accuracy : 0.5789
##                  95% CI : (0.335, 0.7975)
##     No Information Rate : 0.6316
##     P-Value [Acc > NIR] : 0.7650
##
##                   Kappa : -0.1014
##
##  Mcnemar's Test P-Value : 0.0771
##
##             Sensitivity : 0.9167
##             Specificity : 0.0000
##          Pos Pred Value : 0.6111
##          Neg Pred Value : 0.0000
##              Prevalence : 0.6316
##          Detection Rate : 0.5789
##    Detection Prevalence : 0.9474
##       Balanced Accuracy : 0.4583
##
##        'Positive' Class : M
##
```

The most important variables that permit the best prediction and contribute the most to the model are the following:

```
plot(varImp(naive), top=5, main="Naive Bayes - Top 5 variables")
```



4.5 Logistic Regression Model

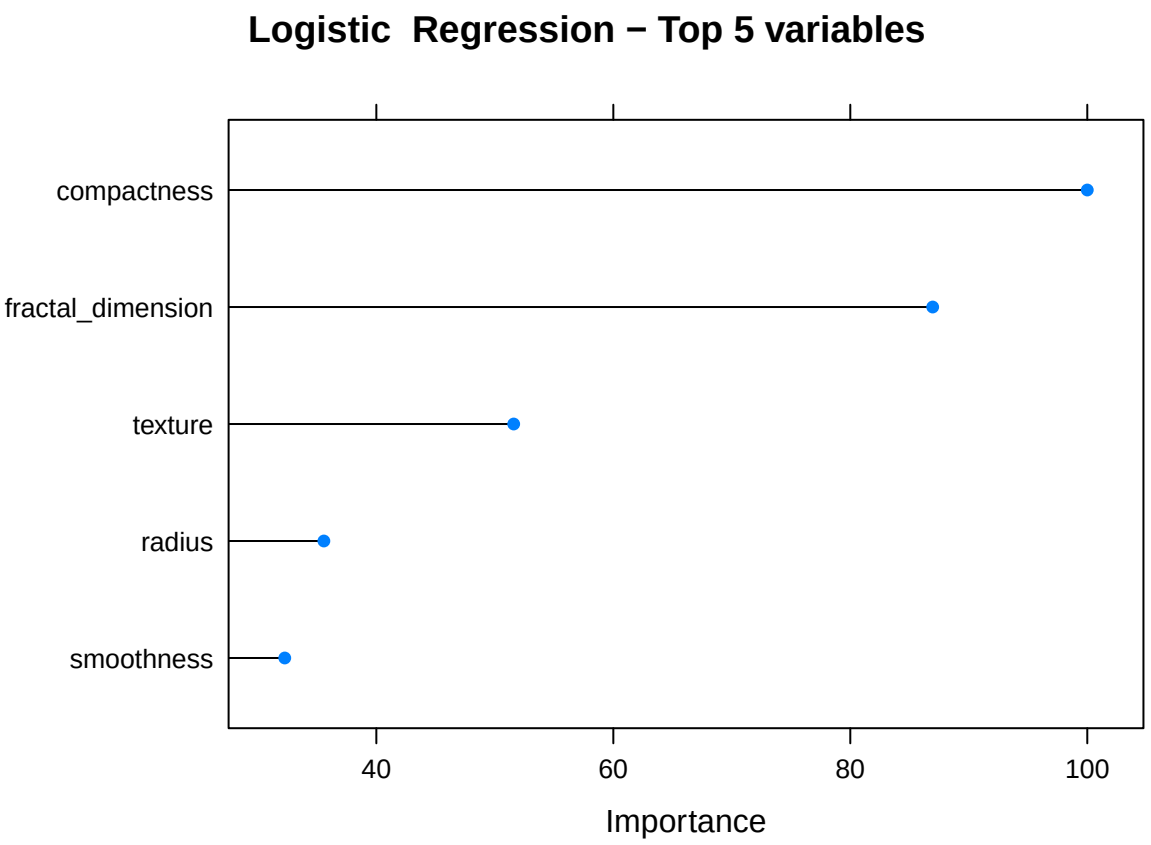
Logistic Regression is widly used for binary classification like (0,1). The binary logistic model is used to estimate the probability of a binary response based on one or more predictor (or independent) variables (features).

```
logreg<- train(diagnosis_result ~.,
               data = training, method = "glm",
                     metric = "ROC",
                     preProcess = c("scale", "center"),  # in order to normalize the data
                     trControl= fitControl)
logreg_pred<- predict(logreg, testing)

# Checking results
confusionmatrix_logreg <- confusionMatrix(logreg_pred, testing$diagnosis_result, positive = "M")
confusionmatrix_logreg
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  B  M
##          B  0  0
##          M  7 12
##
##                Accuracy : 0.6316
##                  95% CI : (0.3836, 0.8371)
##     No Information Rate : 0.6316
##     P-Value [Acc > NIR] : 0.60135
##
##                   Kappa : 0
##
##  Mcnemar's Test P-Value : 0.02334
##
##             Sensitivity : 1.0000
##             Specificity : 0.0000
##          Pos Pred Value : 0.6316
##          Neg Pred Value :    NaN
##              Prevalence : 0.6316
##          Detection Rate : 0.6316
##    Detection Prevalence : 1.0000
##       Balanced Accuracy : 0.5000
##
##        'Positive' Class : M
##
```

The most important variables that permit the best prediction and contribute the most to the model are the following:

```
plot(varImp(logreg), top=5, main=" Logistic  Regression - Top 5 variables")
```



4.6 Random Forest Model

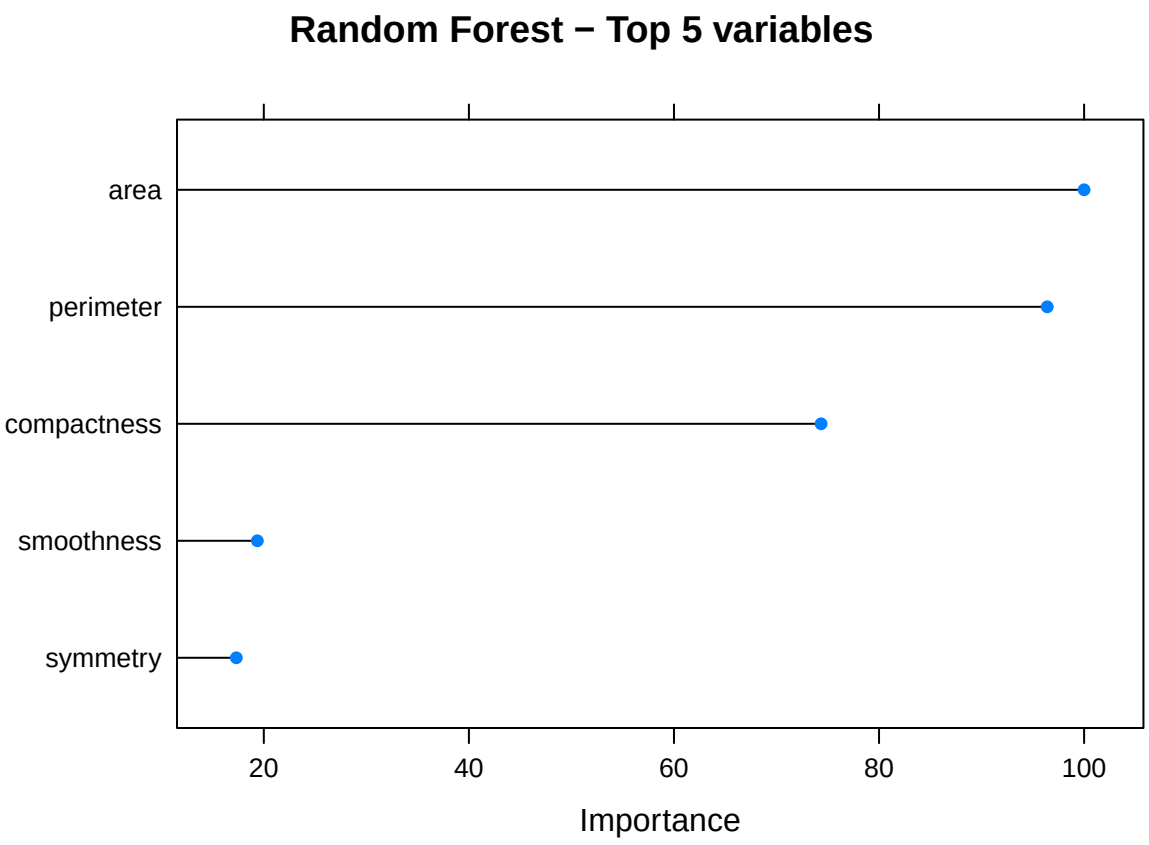
Random Forest is a supervised learning algorithm that builds multiple decision trees and merges them together to get a more accurate and stable prediction.

```
randomforest <- train(diagnosis_result~.,
                              training,
                              method="rf",
                              metric="ROC",
                              tuneLength=10,
                              tuneGrid = expand.grid(mtry = c(2, 3, 6)),
                              preProcess = c('center', 'scale'),
                              trControl=fitControl)

randomforest_pred <- predict(randomforest, testing)

confusionmatrix_randomforest <- confusionMatrix(randomforest_pred, testing$diagnosis_result, positive =
confusionmatrix_randomforest
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  B  M
##          B  1  2
##          M  6 10
##
##                Accuracy : 0.5789
```

```
##                  95% CI : (0.335, 0.7975)
##     No Information Rate : 0.6316
##     P-Value [Acc > NIR] : 0.7650
##
##                   Kappa : -0.027
##
##  Mcnemar's Test P-Value : 0.2888
##
##             Sensitivity : 0.8333
##             Specificity : 0.1429
##          Pos Pred Value : 0.6250
##          Neg Pred Value : 0.3333
##              Prevalence : 0.6316
##          Detection Rate : 0.5263
##    Detection Prevalence : 0.8421
##       Balanced Accuracy : 0.4881
##
##        'Positive' Class : M
##
```

```
plot(varImp(randomforest), top=5, main="Random Forest - Top 5 variables")
```



4.7 K Nearest Neighbors (KNN) Model

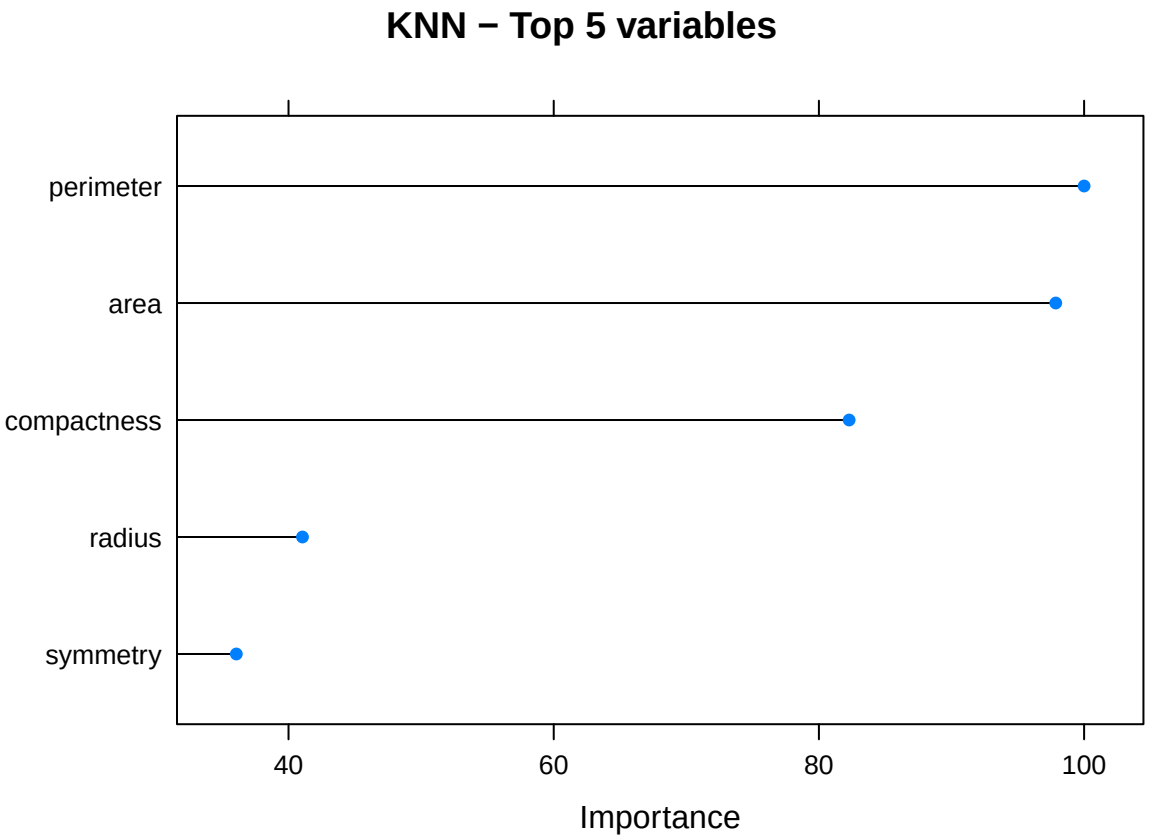
KNN (K-Nearest Neighbors) is a classifier algorithm where the learning is based "how similar" is a data from other. K nearest neighbors algorithm stores all available cases and classifies new cases based on a similarity measure.

```
knn <- train(diagnosis_result~.,
                   training,
                   method="knn",
                   metric="ROC",
                   preProcess = c('center', 'scale'),
                   tuneLength=10,
                   trControl=fitControl)

knn_pred <- predict(knn, testing)
confusionmatrix_knn <- confusionMatrix(knn_pred, testing$diagnosis_result, positive = "M")
confusionmatrix_knn
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  B  M
##          B  0  0
##          M  7 12
##
##                Accuracy : 0.6316
##                  95% CI : (0.3836, 0.8371)
##     No Information Rate : 0.6316
##     P-Value [Acc > NIR] : 0.60135
##
##                   Kappa : 0
##
##  Mcnemar's Test P-Value : 0.02334
##
##             Sensitivity : 1.0000
##             Specificity : 0.0000
##          Pos Pred Value : 0.6316
##          Neg Pred Value :    NaN
##              Prevalence : 0.6316
##          Detection Rate : 0.6316
##    Detection Prevalence : 1.0000
##       Balanced Accuracy : 0.5000
##
##        'Positive' Class : M
##
```

The most important variables that permit the best prediction and contribute the most to the model are the following:

```
plot(varImp(knn), top=5, main="KNN - Top 5 variables")
```



4.8 Neural Network with PCA Model

Artificial Neural Networks (NN) are a types of mathematical algorithms originating in the simulation of networks of biological neurons that ares designed to recognize patterns.

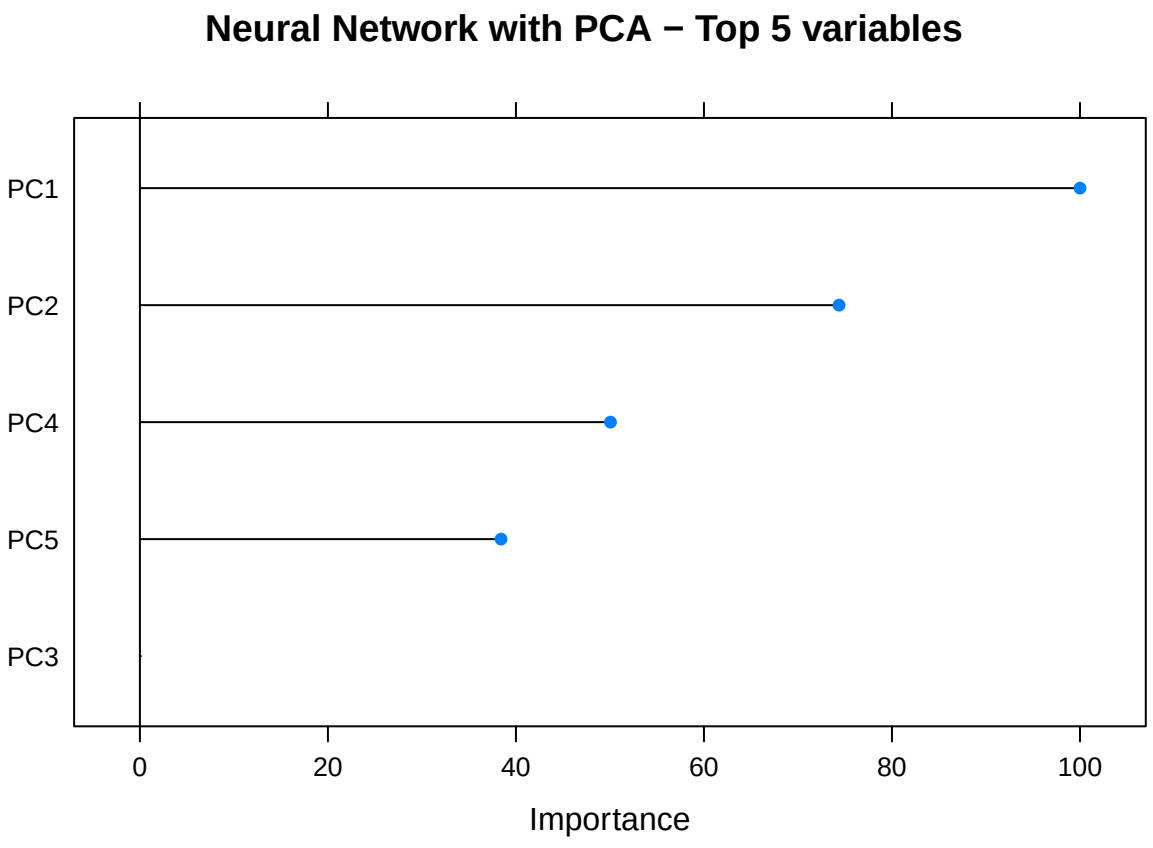
```
nn_pca <- train(diagnosis_result~.,
                        training,
                        method="nnet",
                        metric="ROC",
                        preProcess=c('center', 'scale', 'pca'),
                        tuneLength=10,
                        trace=FALSE,
                        trControl=fitControl)

nn_pca_pred <- predict(nn_pca, testing)
confusionmatrix_nn_pca <- confusionMatrix(nn_pca_pred, testing$diagnosis_result, positive = "M")
confusionmatrix_nn_pca
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  B  M
##          B  0  0
##          M  7 12
##
##                Accuracy : 0.6316
##                  95% CI : (0.3836, 0.8371)
```

```
##     No Information Rate : 0.6316
##     P-Value [Acc > NIR] : 0.60135
##
##                   Kappa : 0
##
##  Mcnemar's Test P-Value : 0.02334
##
##             Sensitivity : 1.0000
##             Specificity : 0.0000
##          Pos Pred Value : 0.6316
##          Neg Pred Value :    NaN
##              Prevalence : 0.6316
##          Detection Rate : 0.6316
##    Detection Prevalence : 1.0000
##       Balanced Accuracy : 0.5000
##
##        'Positive' Class : M
##
```

The most important variables that permit the best prediction and contribute the most to the model are the following:

```
plot(varImp(nn_pca), top=5, main="Neural Network with PCA - Top 5 variables")
```



4.9 Neural Network with LDA Model

Creating the training and testing set of LDA data:

```
training_lda <- lda_predict[data_sampling_index, ]
testing_lda <- lda_predict[-data_sampling_index, ]
```

```
nn_lda <- train(diagnosis_result~.,
                        training_lda,
                        method="nnet",
                        metric="ROC",
                        preProcess=c('center', 'scale'),
                        tuneLength=10,
                        trace=FALSE,
                        trControl=fitControl)

nn_lda_pred <- predict(nn_lda, testing_lda)
confusionmatrix_nn_lda <- confusionMatrix(nn_lda_pred, testing_lda$diagnosis_result, positive = "M")
confusionmatrix_nn_lda
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  B  M
##          B  6  0
##          M  1 12
##
##                Accuracy : 0.9474
##                  95% CI : (0.7397, 0.9987)
##     No Information Rate : 0.6316
##     P-Value [Acc > NIR] : 0.001951
##
##                   Kappa : 0.8834
##
##  Mcnemar's Test P-Value : 1.000000
##
##             Sensitivity : 1.0000
##             Specificity : 0.8571
##          Pos Pred Value : 0.9231
##          Neg Pred Value : 1.0000
##              Prevalence : 0.6316
##          Detection Rate : 0.6316
##    Detection Prevalence : 0.6842
##       Balanced Accuracy : 0.9286
##
##        'Positive' Class : M
##
```

Chapter 5

Results

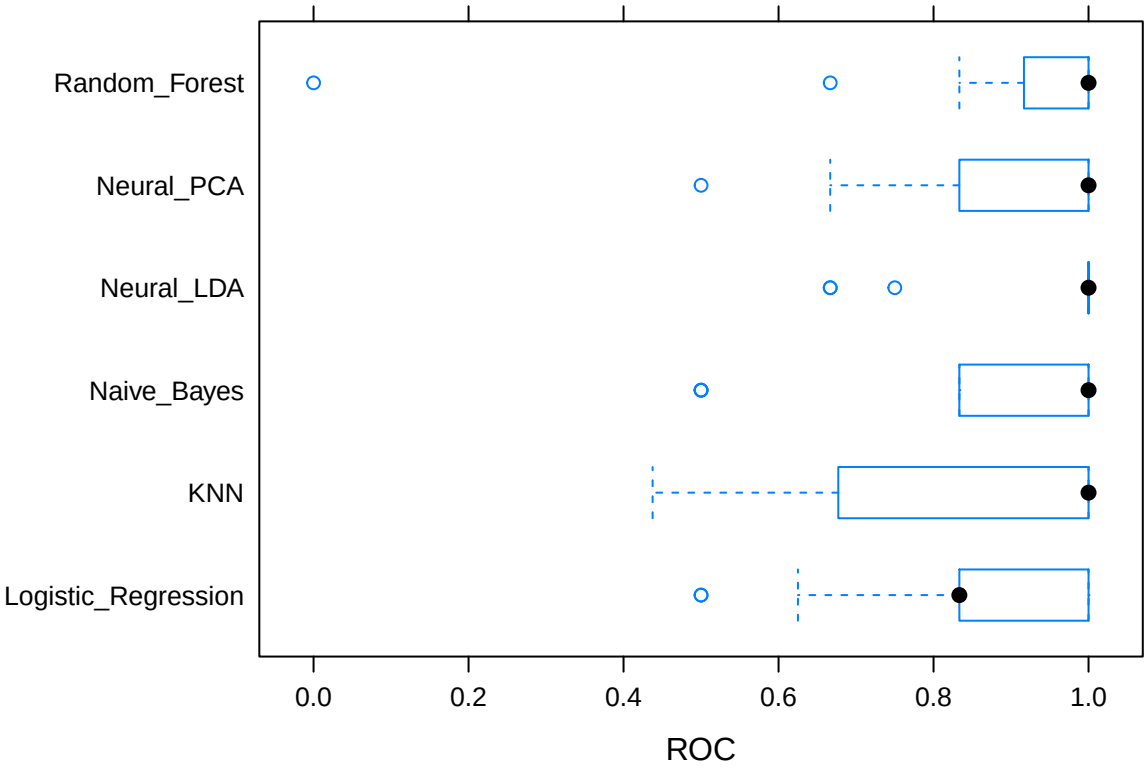
Now, we can compare and evaluate the results:

```
model_list <- list(Naive_Bayes=naive,
                   Logistic_Regression=logreg,
                   Random_Forest=randomforest,
                   KNN=knn,
                   Neural_PCA=nn_pca,
                   Neural_LDA=nn_lda)
model_results <- resamples(model_list)

summary(model_results)
```

```
##
## Call:
## summary.resamples(object = model_results)
##
## Models: Naive_Bayes, Logistic_Regression, Random_Forest, KNN, Neural_PCA, Neural_LDA
## Number of resamples: 15
##
## ROC
##                          Min.   1st Qu.    Median      Mean 3rd Qu. Max.
## Naive_Bayes         0.5000000 0.8333333 1.0000000 0.8611111       1    1
## Logistic_Regression 0.5000000 0.8333333 0.8333333 0.8444444       1    1
## Random_Forest       0.0000000 0.9166667 1.0000000 0.8888889       1    1
## KNN                 0.4375000 0.6770833 1.0000000 0.8444444       1    1
## Neural_PCA          0.5000000 0.8333333 1.0000000 0.8953704       1    1
## Neural_LDA          0.6666667 1.0000000 1.0000000 0.9388889       1    1
##                     NA's
## Naive_Bayes            0
## Logistic_Regression    0
## Random_Forest          0
## KNN                    0
## Neural_PCA             0
## Neural_LDA             0
##
## Sens
##                     Min. 1st Qu. Median      Mean 3rd Qu. Max. NA's
## Naive_Bayes          0.5     0.5    1.0 0.8000000       1    1    0
```

```
## Logistic_Regression  0.0     0.5    1.0 0.7333333       1    1    0
## Random_Forest        0.0     0.5    1.0 0.7333333       1    1    0
## KNN                  0.0     0.5    1.0 0.7111111       1    1    0
## Neural_PCA           0.5     0.5    0.5 0.7111111       1    1    0
## Neural_LDA           0.0     0.5    1.0 0.7777778       1    1    0
##
## Spec
##                            Min.   1st Qu. Median      Mean 3rd Qu. Max. NA's
## Naive_Bayes          0.3333333 0.6666667   0.75 0.7833333       1    1    0
## Logistic_Regression 0.3333333 0.7500000   1.00 0.8777778       1    1    0
## Random_Forest        0.3333333 0.6666667   1.00 0.8666667       1    1    0
## KNN                  0.6666667 1.0000000   1.00 0.9444444       1    1    0
## Neural_PCA           0.5000000 0.6666667   1.00 0.8777778       1    1    0
## Neural_LDA           0.3333333 0.7083333   1.00 0.8500000       1    1    0
```

```
bwplot(model_results, metric="ROC")
```



The Receiver Operating characteristic Curve is a graph showing the performance of a classification model at all classification thresholds metric measure the auc of the roc curve of each model. This metric is independent of any threshold. Prediction classes are obtained by default with a threshold of 0.5 which could not be the best with an unbalanced dataset.

```
confusionmatrix_list <- list(
  Naive_Bayes=confusionmatrix_naive,
  Logistic_Regression=confusionmatrix_logreg,
  Random_Forest=confusionmatrix_randomforest,
  KNN=confusionmatrix_knn,
  Neural_PCA=confusionmatrix_nn_pca,
  Neural_LDA=confusionmatrix_nn_lda)
confusionmatrix_list_results <- sapply(confusionmatrix_list, function(x) x$byClass)
confusionmatrix_list_results %>% knitr::kable()
```

	Naive_Bayes	Logistic_Regression	Random_Forest	KNN	Neural_PCA	Neural_LDA
Sensitivity	0.9166667	1.0000000	0.8333333	1.0000000	1.0000000	1.0000000
Specificity	0.0000000	0.0000000	0.1428571	0.0000000	0.0000000	0.8571429
Pos Pred Value	0.6111111	0.6315789	0.6250000	0.6315789	0.6315789	0.9230769
Neg Pred Value	0.0000000	NaN	0.3333333	NaN	NaN	1.0000000
Precision	0.6111111	0.6315789	0.6250000	0.6315789	0.6315789	0.9230769
Recall	0.9166667	1.0000000	0.8333333	1.0000000	1.0000000	1.0000000
F1	0.7333333	0.7741935	0.7142857	0.7741935	0.7741935	0.9600000
Prevalence	0.6315789	0.6315789	0.6315789	0.6315789	0.6315789	0.6315789
Detection Rate	0.5789474	0.6315789	0.5263158	0.6315789	0.6315789	0.6315789
Detection Prevalence	0.9473684	1.0000000	0.8421053	1.0000000	1.0000000	0.6842105
Balanced Accuracy	0.4583333	0.5000000	0.4880952	0.5000000	0.5000000	0.9285714

Chapter 6

Discussion

Analysing the metrics results, the best results in Sensitivity and Detection Prevalence is KNN Model, in Prevalence is Logistic Regression and the Model that has more best metrics results in Specificity, Positivie Prediction Value, Precision, Recall, F1 score, Detection Rate and Balanced Accuracy is Neural Network with Linear Discriminant Analysis Model.

```
confusionmatrix_results_max <- apply(confusionmatrix_list_results, 1, which.is.max)

metrics_report <- data.frame(metric=names(confusionmatrix_results_max),
                             best_model=colnames(confusionmatrix_list_results)[confusionmatrix_results_ma
                             value=mapply(function(x,y) {confusionmatrix_list_results[x,y]},
                                          names(confusionmatrix_results_max),
                                          confusionmatrix_results_max))
rownames(metrics_report) <- NULL
metrics_report
```

```
##                 metric          best_model     value
## 1          Sensitivity                 KNN 1.0000000
## 2          Specificity          Neural_LDA 0.8571429
## 3       Pos Pred Value          Neural_LDA 0.9230769
## 4       Neg Pred Value                <NA>        NA
## 5            Precision          Neural_LDA 0.9230769
## 6               Recall          Neural_LDA 1.0000000
## 7                   F1          Neural_LDA 0.9600000
## 8           Prevalence Logistic_Regression 0.6315789
## 9       Detection Rate          Neural_LDA 0.6315789
## 10 Detection Prevalence          Neural_PCA 1.0000000
## 11    Balanced Accuracy          Neural_LDA 0.9285714
```

Chapter 7

Conclusion

We analised a group of machine learning models and we choose Neural Network with Linear Discriminant Analysis (LDA) Model with high Specificity, Positive Prediction Value, Precision, Recall, F1 score, Detection Rate and Balanced Accuracy for the dataset analised.

Chapter 8

System Information

```
## [1] "Operating System and R:"
##                _
## platform       x86_64-pc-linux-gnu
## arch           x86_64
## os             linux-gnu
## system         x86_64, linux-gnu
## status
## major          3
## minor          6.2
## year           2019
## month          12
## day            12
## svn rev        77560
## language       R
## version.string R version 3.6.2 (2019-12-12)
## nickname       Dark and Stormy Night
```