Breast Cancer Prediction

Antonio Caputo

26/09/2019

Introduction

This project is about one of the most important problem in today society: cancer disease prediction. With big data and Machine Learning growth in biomedical and healthcare communities, accurate analysis of medical data can predict early disease detection. Breast cancer is most common form of cancer in Women. It represents about 12% of all new cancer cases and 25% of all cancers in women. Here we build Machine Learning Models to predict the type of Breast Cancer (Malignant or Benign).

In order to build our model we use a study from US state of Wisconsin the Breast Cancer Wisconsin (Diagnostic) DataSet (https://archive.ics.uci.edu/ml/machine-learning-databases/breast-cancer-wisconsin/breast-cancer-wisconsin.data).The data used for this project was collected in 1993 by the University of Wisconsin and it is composed by the biopsy result of 569 patients in Wisconsin Hospital. 3-D digitized images of breast mass(lump) were used for extracting information. The dataset's features describe characteristics of the cell nuclei on the image. The features information are specified below:

- ID number
- Diagnosis (M = malignant, B = benign)

and then we have the 10 variables used for the prediction:

1. radius: mean of distances from center to points on the perimeter
2. texture: standard deviation of grey-scale values
3. perimeter
4. area: Number of pixels inside contour + ½ for pixels on perimeter
5. smoothness: local variation in radius lengths
6. compactness: perimeter^2 / area

7. concavity: severity of concave portions of the contour
8. concave points: number of concave portions of the contour
9. symmetry
10. fractal dimension: "coastline approximation"

```
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(data.table)) install.packages("data.table", repos = "http://cran.us.r-project.org")
if(!require(randomForest)) install.packages("randomForest", 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(corrplot)) install.packages("corrplot", repos = "http://cran.us.r-project.org")
if(!require(gam)) install.packages("gam", repos = "http://cran.us.r-project.org")


data <- read.csv("https://archive.ics.uci.edu/ml/machine-learning-databases/breast-cancer-wisconsin/wdb
colnames(data)<- c("id","diagnosis","radius_mean","texture_mean","perimeter_mean","area_mean","smoothnes
```

Then we have the mean, standard error and "worst" or largest (mean of the three largest values) of these features were computed for each image, resulting in 30 variables

This project will make a performance comparison between different machine learning algorithms in order to to assess the correctness in classifying data with respect to efficiency and effectiveness of each algorithm in terms of accuracy, precision, sensitivity and specificity, in order to find the best diagnosis.

Method/Analysis

Let's analize our data, dimension and type.

```
class(data)
```

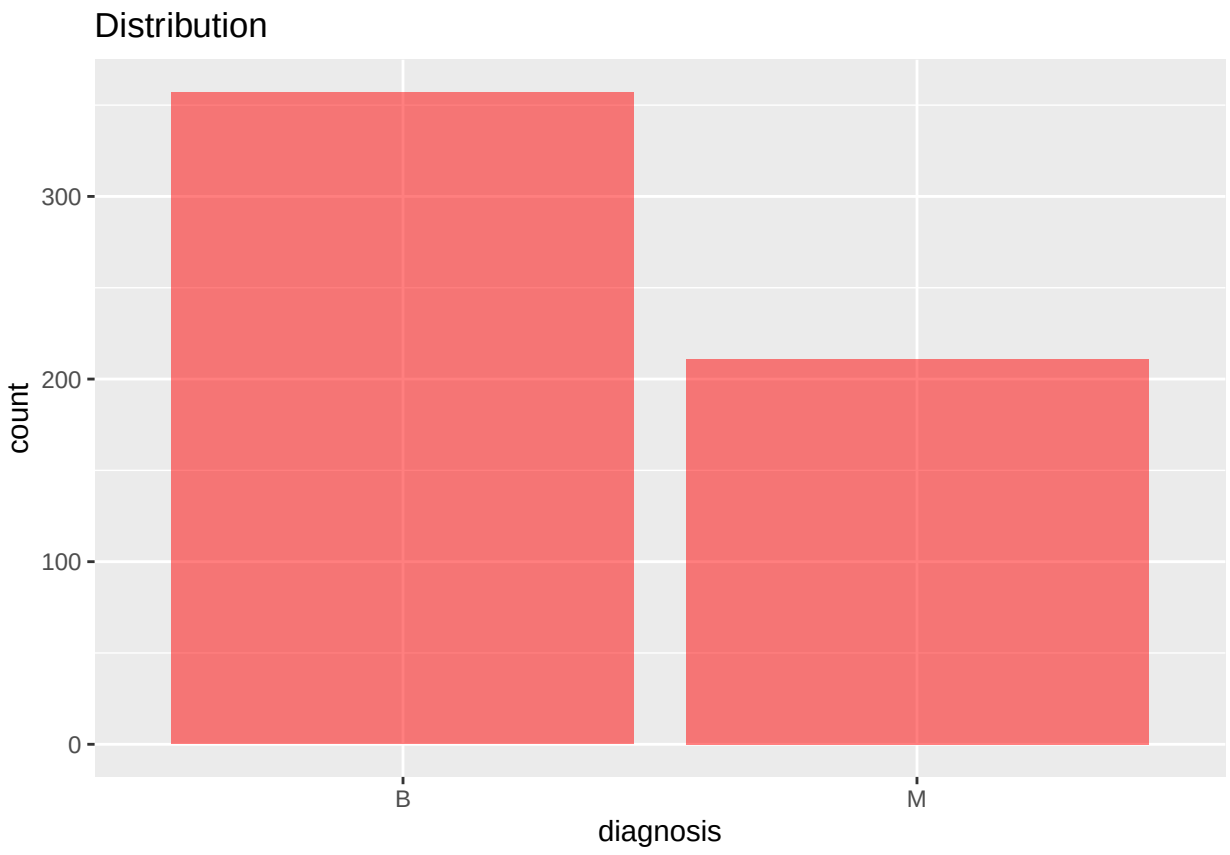
```
[1] "data.frame"
```

```
dim(data)
```

```
[1] 568  32
```

So we have 568 observations, where `211` are malignant and `357` are benign. So the distrubution of the observations is not balanced.

```
ggplot(data, aes(x=diagnosis))+geom_bar(fill="red",alpha=0.5)+labs(title="Distribution")
```

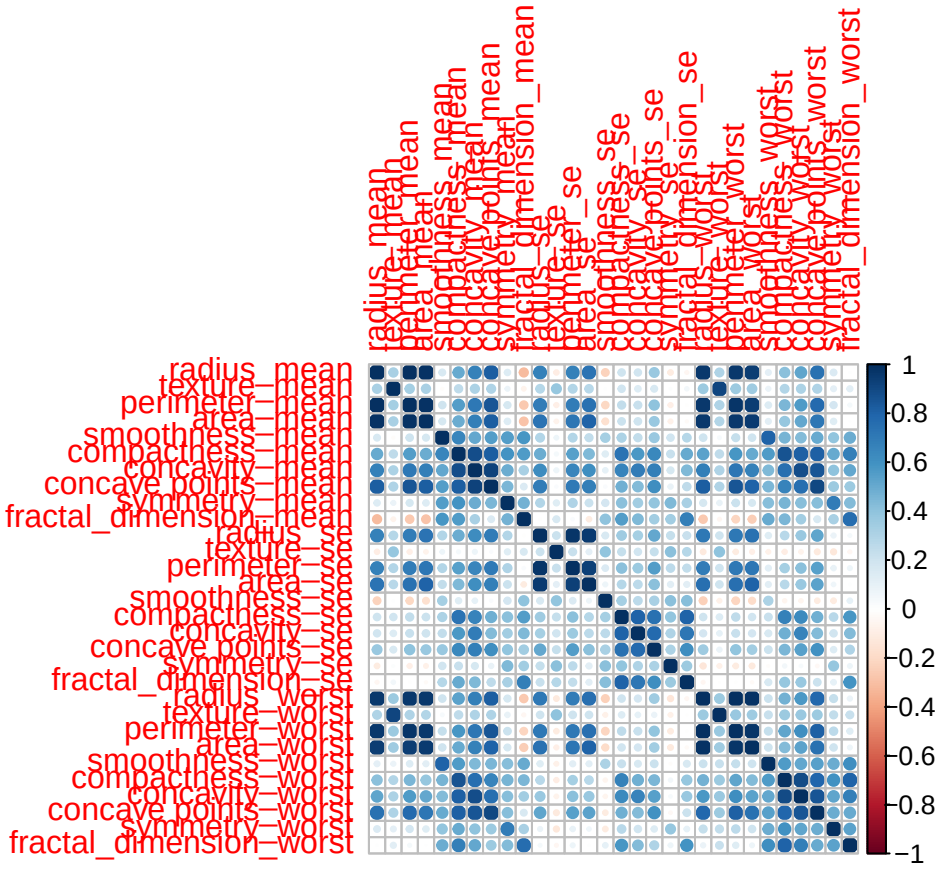


In our data set thera are 30 features, so we have to try to lighten it, in order to have a lighter model. What we can do is see if there are any features with high correlation that give us just redundant information.

So we build the correlation Matrix and find the features with correletion more than 0.9 and than we remove them from our data set.

We also can see a corrplot for better understand which column are more correlated:

```
correlationMatrix <- cor(data[,3:ncol(data)])


corrplot(correlationMatrix)
```




```
# find features that are highly corrected (corr >0.90)
high_correlation <- findCorrelation(correlationMatrix, cutoff=0.9)


# print indexes of highly correlated attributes
print(high_correlation)
```

```
 [1]  7  8 23 21  3 24  1 13 14  2
```

```
# Remove correlated variables

data_no_cor <- data %>% select(-high_correlation)
```

```r
#final data set

data <- cbind(diagnosis= data$diagnosis, data_no_cor)
```

In the end our attributes will be 22

Before we start our analysis, we have to do the last step, which consist of creatring the train and test set from our data. We split the dataset into Train (70%) and Test (30%)

```r
# split data in train and test set
set.seed(1, sample.kind="Rounding")

data_index <- createDataPartition(data$diagnosis, times=1, p=0.7, list = FALSE)
train_set <- data[data_index, ]
test_set <- data[-data_index, ]

fitControl <- trainControl(method="cv", number = 15, classProbs = TRUE,summaryFunction = twoClassSummary
```

Result

In this project we will now train different model to predict if the observation is Malignant or Benign, and we will compare them using different coefficent like accuracy, sensitivity, specificity, F1 score.

Logistic Regression Model

Logistic regression is a specific case of a set of generalized linear models. Below we can see the results of this model:

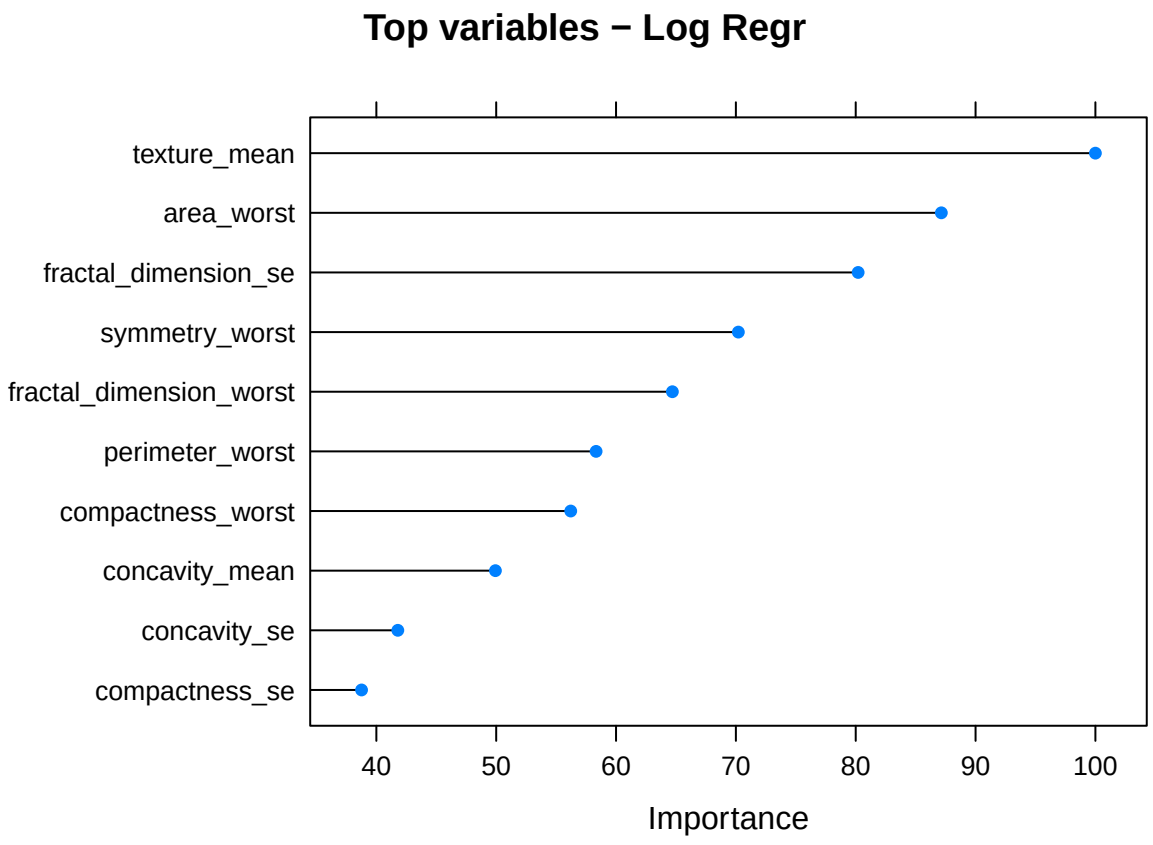
```r
logr_model<- train(diagnosis ~ . ,data= train_set, method= "glm",
                   preProcess=c("scale", "center"),metric="ROC", trControl= fitControl)

logr_pred<- predict(logr_model, test_set)

logr_confmatrix<- confusionMatrix(logr_pred, test_set$diagnosis, positive = "M")

logr_confmatrix
```

```
Confusion Matrix and Statistics

          Reference
Prediction   B   M
         B 105   7
         M   2  56

               Accuracy : 0.9471
                 95% CI : (0.9019, 0.9755)
    No Information Rate : 0.6294
    P-Value [Acc > NIR] : <2e-16

                  Kappa : 0.8846
```

```
 Mcnemar's Test P-Value : 0.1824

            Sensitivity : 0.8889
            Specificity : 0.9813
         Pos Pred Value : 0.9655
         Neg Pred Value : 0.9375
             Prevalence : 0.3706
         Detection Rate : 0.3294
   Detection Prevalence : 0.3412
      Balanced Accuracy : 0.9351

       'Positive' Class : M
```

And a plot of the most important variable:

```
plot(varImp(logr_model), top=10, main="Top variables - Log Regr")
```



k-Nearest Neighbor

With k-nearest neighbors (kNN) we estimate p(x1, x2) in a similar way to bin smoothing. However, as we will see, kNN is easier to adapt to multiple dimensions. First we define the distance between all observations based on the features. Then, for any point (x1, x2) for which we want an estimate of p(x1, x2), we look for the k nearest points to (x1, x2) and then take an average.

```r
#knn

knn_model<- train(diagnosis ~ . ,data= train_set, method= "knn",
                  preProcess=c("scale", "center"),metric="ROC", trControl= fitControl, tuneGrid = data.f

knn_pred<- predict(knn_model, test_set)

knn_confmatrix<- confusionMatrix(knn_pred, test_set$diagnosis, positive = "M")

knn_confmatrix
```

```
Confusion Matrix and Statistics

          Reference
Prediction   B   M
         B 103   2
         M   4  61

               Accuracy : 0.9647
                 95% CI : (0.9248, 0.9869)
    No Information Rate : 0.6294
    P-Value [Acc > NIR] : <2e-16

                  Kappa : 0.9248

 Mcnemar's Test P-Value : 0.6831

            Sensitivity : 0.9683
            Specificity : 0.9626
         Pos Pred Value : 0.9385
         Neg Pred Value : 0.9810
             Prevalence : 0.3706
         Detection Rate : 0.3588
   Detection Prevalence : 0.3824
      Balanced Accuracy : 0.9654

       'Positive' Class : M
```

Random Forest Model

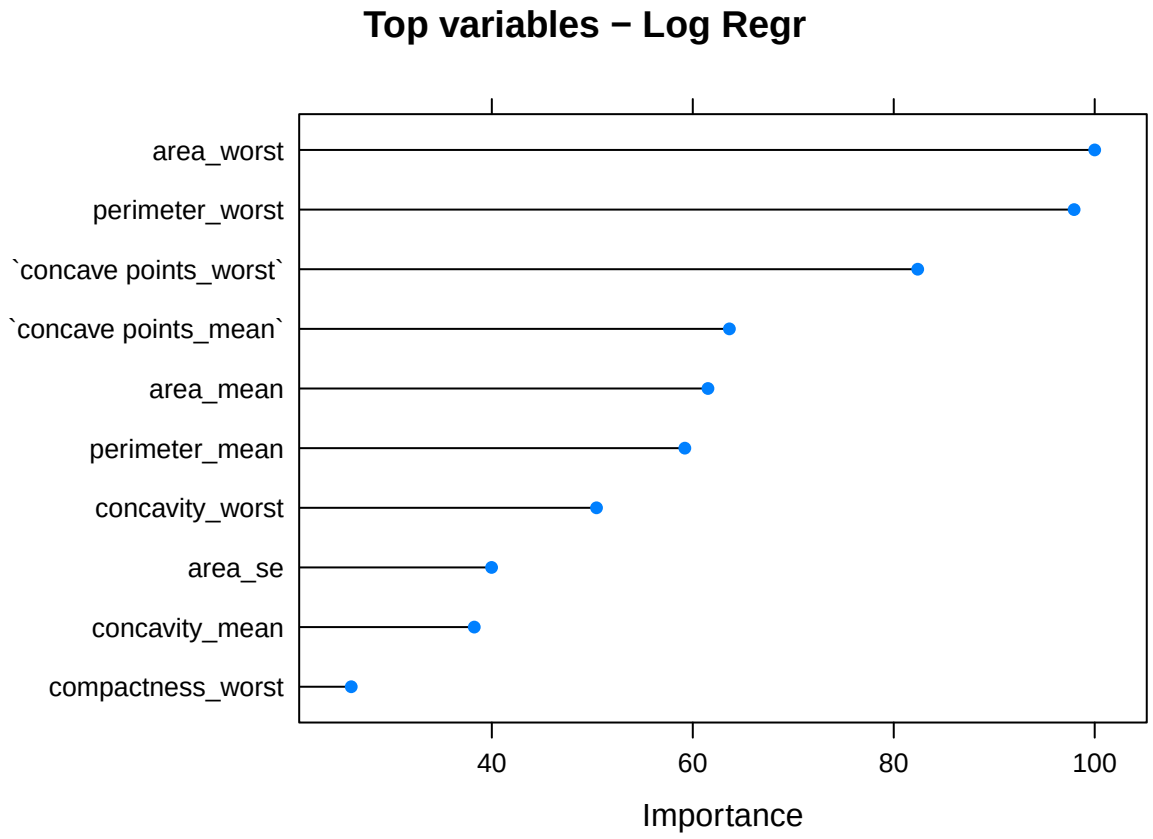
Random forest is a method based on decision trees. It split data into sub-samples, trains decision tree classifiers on each sub-sample and averages prediction of each classifier.

```r
rf_model<- train(diagnosis ~ . ,data= train_set, method= "rf",
                   preProcess=c("scale", "center"),metric="ROC", trControl= fitControl)

rf_pred<- predict(rf_model, test_set)

rf_confmatrix<- confusionMatrix(rf_pred, test_set$diagnosis, positive = "M")

rf_confmatrix
```

```
Confusion Matrix and Statistics

          Reference
Prediction   B   M
         B 103   1
         M   4  62

               Accuracy : 0.9706
                 95% CI : (0.9327, 0.9904)
    No Information Rate : 0.6294
    P-Value [Acc > NIR] : <2e-16

                  Kappa : 0.9376

 Mcnemar's Test P-Value : 0.3711

            Sensitivity : 0.9841
            Specificity : 0.9626
         Pos Pred Value : 0.9394
         Neg Pred Value : 0.9904
             Prevalence : 0.3706
         Detection Rate : 0.3647
   Detection Prevalence : 0.3882
      Balanced Accuracy : 0.9734

       'Positive' Class : M
```

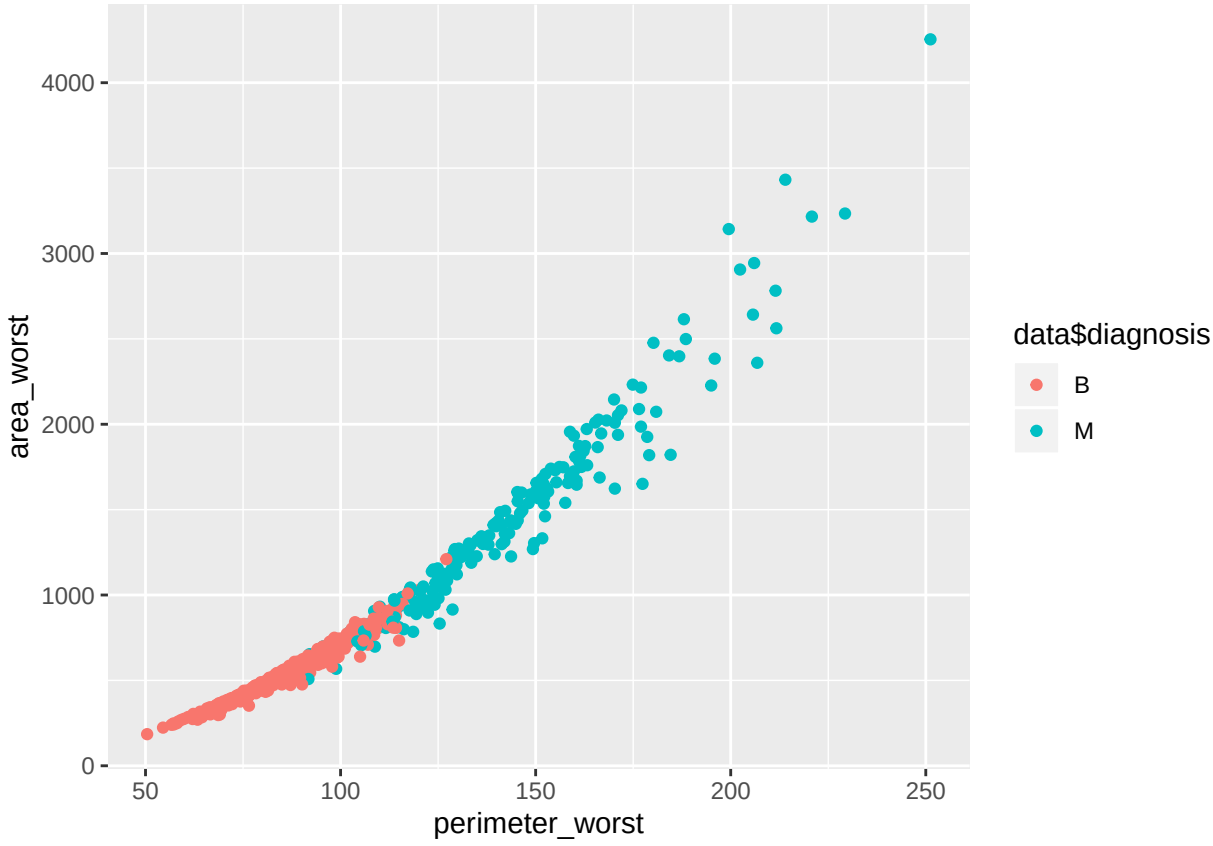
And a plot of the most important variable:

```r
plot(varImp(rf_model), top=10, main="Top variables - Log Regr")
```



We can see the plot with the two most important predictor.

```
data%>%ggplot(aes(perimeter_worst, area_worst, col=data$diagnosis))+ geom_point()
```



Quadratic Discriminant Analysis QDA

Quadratic Discriminant Analysis (QDA) is a version of Naive Bayes in which we assume that the distributions pX|Y =1(x) and pX|Y =0(x) are multivariate normal.

```
#qda

qda_model<- train(diagnosis ~ . ,data= train_set, method= "qda",
                  preProcess=c("scale", "center"))

qda_pred<- predict(qda_model, test_set)

qda_confmatrix<- confusionMatrix(qda_pred, test_set$diagnosis, positive = "M")

qda_confmatrix
```

```
Confusion Matrix and Statistics

          Reference
Prediction   B   M
         B 103   2
         M   4  61

               Accuracy : 0.9647
                 95% CI : (0.9248, 0.9869)
```

```
    No Information Rate : 0.6294
    P-Value [Acc > NIR] : <2e-16

                  Kappa : 0.9248

 Mcnemar's Test P-Value : 0.6831

            Sensitivity : 0.9683
            Specificity : 0.9626
         Pos Pred Value : 0.9385
         Neg Pred Value : 0.9810
             Prevalence : 0.3706
         Detection Rate : 0.3588
   Detection Prevalence : 0.3824
      Balanced Accuracy : 0.9654

       'Positive' Class : M
```

Conclusion

We trained 4 model: logistic regression, Random Forest, k-Nearest Neighbor and Quadratic Discriminant Analysis. Let's see now our results and difference. Below we have summary table for every model:

```
models<- list(Log_regr= logr_model, Random_Forest= rf_model, KNN= knn_model,QDA=qda_model)

confusionmatrix<- list(Log_regr= logr_confmatrix, Random_Forest= rf_confmatrix,
                       KNN= knn_confmatrix,QDA=qda_confmatrix)

confusionmatrix_results <- sapply(confusionmatrix, function(x) x$byClass)
confusionmatrix_results %>% knitr::kable()
```

	Log_regr	Random_Forest	KNN	QDA
Sensitivity	0.8888889	0.9841270	0.9682540	0.9682540
Specificity	0.9813084	0.9626168	0.9626168	0.9626168
Pos Pred Value	0.9655172	0.9393939	0.9384615	0.9384615
Neg Pred Value	0.9375000	0.9903846	0.9809524	0.9809524
Precision	0.9655172	0.9393939	0.9384615	0.9384615
Recall	0.8888889	0.9841270	0.9682540	0.9682540
F1	0.9256198	0.9612403	0.9531250	0.9531250
Prevalence	0.3705882	0.3705882	0.3705882	0.3705882
Detection Rate	0.3294118	0.3647059	0.3588235	0.3588235
Detection Prevalence	0.3411765	0.3882353	0.3823529	0.3823529
Balanced Accuracy	0.9350987	0.9733719	0.9654354	0.9654354

```
confusionmatrix_accuracy <- list(confusionmatrix$Log_regr$overall["Accuracy"],
                                 confusionmatrix$Random_Forest$overall["Accuracy"],
                                 confusionmatrix$KNN$overall["Accuracy"],
                                 confusionmatrix$QDA$overall["Accuracy"])
confusionmatrix_accuracy %>% knitr::kable()
```

	x		x		x		x
Accuracy	0.9470588	Accuracy	0.9705882	Accuracy	0.9647059	Accuracy	0.9647059

We can see that random forest model perfom better, Although in term of accuracy there isn't a lot of difference we notice more difference in term of sensitivity. In this particular kind of problem we are more interested and worried in predicting True Positives and so less False Negative that have a bigger social cost.