Cancer Details

Renato Festa

2023-03-10

Overview

The second phase of the Capstone Project in the HarvardX PH125.9x course is to choose a project in which I can use the knowledge acquired from all the previous topics to develop a machine learning script and a complete report on the dataset I choose to work with. The dataset I have chosen is called Cancer Details, which is available on Kaggle and was posted by ATHIRA G. This dataset contains information on cancer samples, including details such as radius, texture, perimeter, area, and others. In order to download the dataset automatically I needed to upload it in my github, from where the code can download it automatically.

Method

First, I will analyze the dataset to better understand its contents, and then I will begin building the code to improve the accuracy of the results. One of the columns in the dataset determines whether the cancer sample is malignant or benign. My objective is to predict whether a cancer sample is malignant or benign using only the measurements provided by the other columns. Therefore, in this case, I aim to achieve the highest possible accuracy.

Analysis

Downloading the dataset and first view

First, lets take a look at the dataset to get familiar with it.

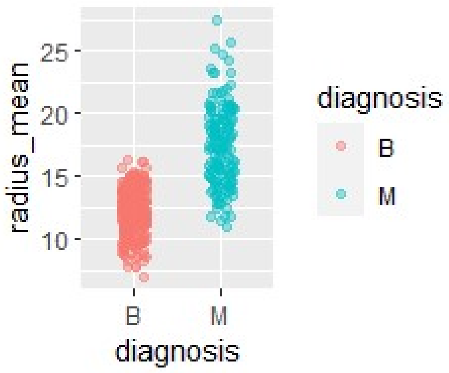
```
## 'data.frame':    569 obs. of  32 variables:
##  $ id                     : int  842302 842517 84300903 84348301 84358402
843786 844359 84458202 844981 84501001 ...
##  $ diagnosis              : Factor w/ 2 levels "B","M": 2 2 2 2 2 2 2 2 2
2 ...
##  $ radius_mean            : num  18 20.6 19.7 11.4 20.3 ...
##  $ texture_mean           : num  10.4 17.8 21.2 20.4 14.3 ...
##  $ perimeter_mean         : num  122.8 132.9 130 77.6 135.1 ...
##  $ area_mean              : num  1001 1326 1203 386 1297 ...
##  $ smoothness_mean        : num  0.1184 0.0847 0.1096 0.1425 0.1003 ...
##  $ compactness_mean       : num  0.2776 0.0786 0.1599 0.2839 0.1328 ...
##  $ concavity_mean         : num  0.3001 0.0869 0.1974 0.2414 0.198 ...
##  $ concave.points_mean    : num  0.1471 0.0702 0.1279 0.1052 0.1043 ...
##  $ symmetry_mean          : num  0.242 0.181 0.207 0.26 0.181 ...
##  $ fractal_dimension_mean : num  0.0787 0.0567 0.06 0.0974 0.0588 ...
##  $ radius_se              : num  1.095 0.543 0.746 0.496 0.757 ...
##  $ texture_se             : num  0.905 0.734 0.787 1.156 0.781 ...
##  $ perimeter_se           : num  8.59 3.4 4.58 3.44 5.44 ...
##  $ area_se                : num  153.4 74.1 94 27.2 94.4 ...
##  $ smoothness_se          : num  0.0064 0.00522 0.00615 0.00911 0.01149
...
##  $ compactness_se         : num  0.049 0.0131 0.0401 0.0746 0.0246 ...
##  $ concavity_se           : num  0.0537 0.0186 0.0383 0.0566 0.0569 ...
##  $ concave.points_se      : num  0.0159 0.0134 0.0206 0.0187 0.0188 ...
##  $ symmetry_se            : num  0.03 0.0139 0.0225 0.0596 0.0176 ...
##  $ fractal_dimension_se   : num  0.00619 0.00353 0.00457 0.00921 0.00511
...
##  $ radius_worst           : num  25.4 25 23.6 14.9 22.5 ...
##  $ texture_worst          : num  17.3 23.4 25.5 26.5 16.7 ...
##  $ perimeter_worst        : num  184.6 158.8 152.5 98.9 152.2 ...
##  $ area_worst             : num  2019 1956 1709 568 1575 ...
##  $ smoothness_worst       : num  0.162 0.124 0.144 0.21 0.137 ...
##  $ compactness_worst      : num  0.666 0.187 0.424 0.866 0.205 ...
##  $ concavity_worst        : num  0.712 0.242 0.45 0.687 0.4 ...
##  $ concave.points_worst   : num  0.265 0.186 0.243 0.258 0.163 ...
##  $ symmetry_worst         : num  0.46 0.275 0.361 0.664 0.236 ...
##  $ fractal_dimension_worst: num  0.1189 0.089 0.0876 0.173 0.0768 ...
```

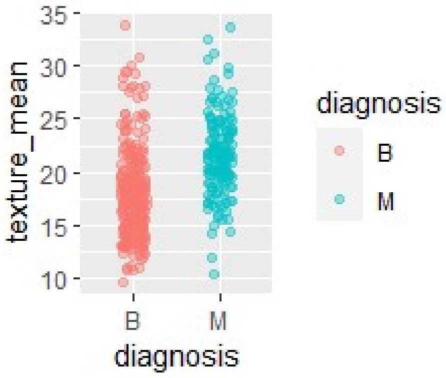
The dataset has 32 columns and 569 rows. The first column is the ID of the sample, and the second one is a factor indicating whether the cancer is malignant or benign. The other 30 columns contain data collected from each sample, which we will use to determine whether a cancer sample is malignant or not.

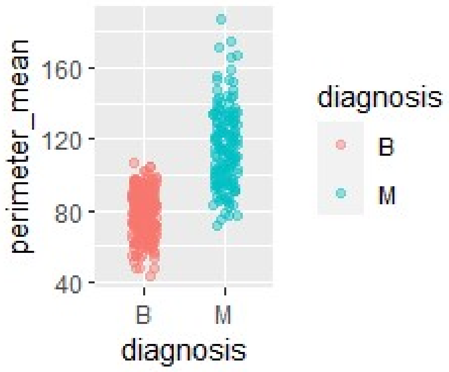
Next, I will split the cancerdetails dataset into a training set and a test set. I will use the training set to predict whether the data from the test set indicates a malignant cancer or not.

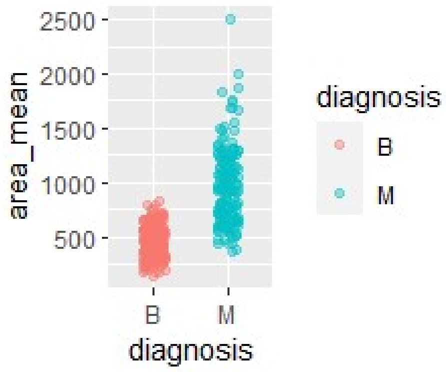
```r
library(caret)
set.seed(321)
test_index <- createDataPartition(cancerdetails$id, times = 1, p = 0.8, list
= FALSE)

train_set <- cancerdetails[test_index, ] #create training set
test_set <- cancerdetails[-test_index, ] #create test set
```

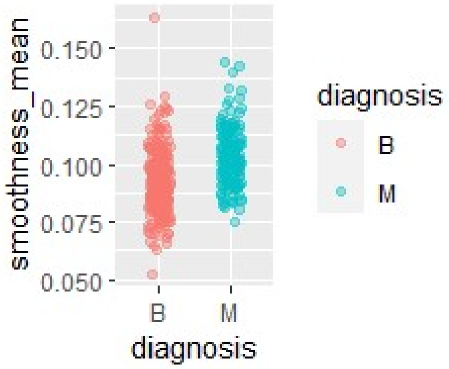
To compare the differences between the two categories of cancer for each column, I will use the training set to create plots. In these plots, I will be examining the differences between the 'M' and 'B' categories to find a better way to differentiate between them.

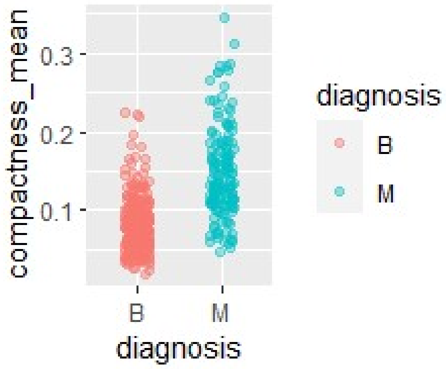

radius_mean
25
20
15
10
B
M
diagnosis
diagnosis
B
M

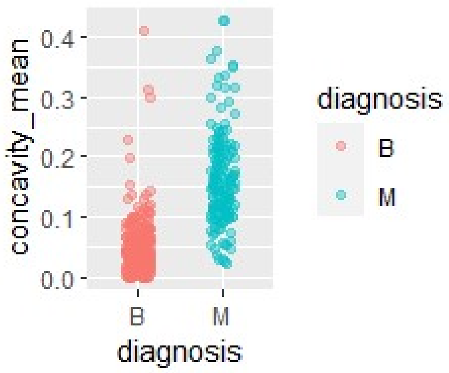

texture_mean
35
30
25
20
15
10
B
M
diagnosis
diagnosis
B
M

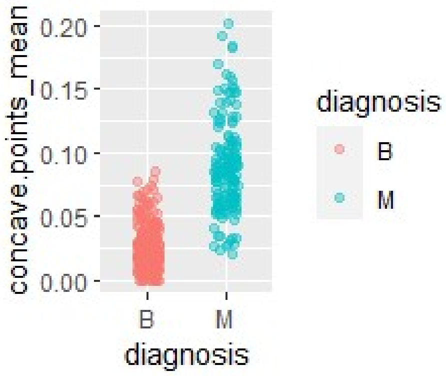

perimeter_mean
160
120
80
40
B
M
diagnosis
diagnosis
B
M

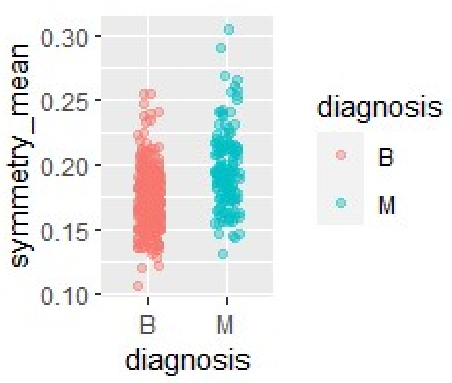

area_mean
2500
2000
1500
1000
500
B
M
diagnosis
diagnosis
B
M

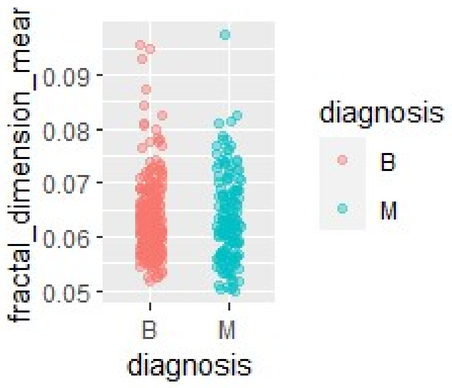

smoothness_mean
0.150
0.125
0.100
0.075
0.050
B
M
diagnosis
diagnosis
B
M

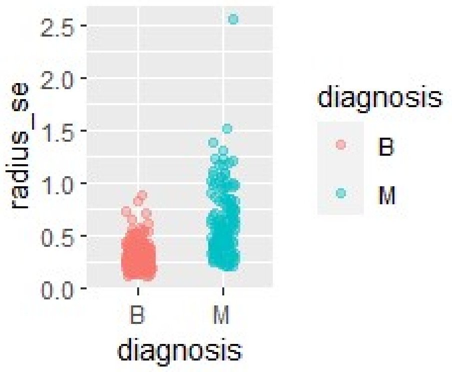

compactness_mean
0.3
0.2
0.1
B
M
diagnosis
diagnosis
B
M

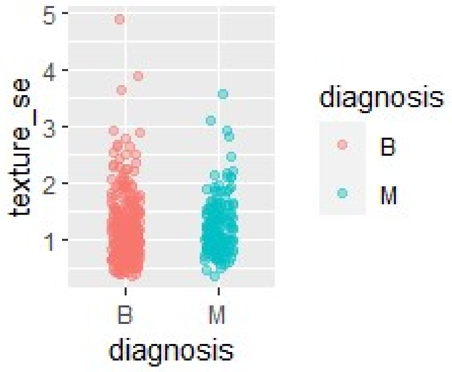

concavity_mean
0.4
0.3
0.2
0.1
0.0
B
M
diagnosis
diagnosis
B
M

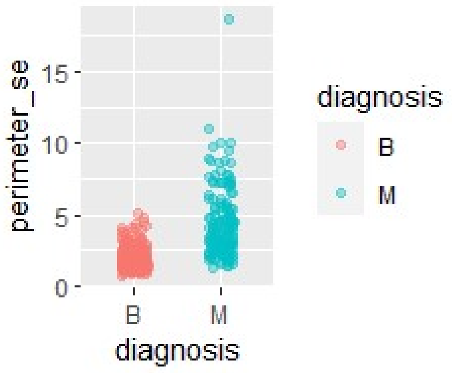

concave.points_mean
0.20
0.15
0.10
0.05
0.00
B
M
diagnosis
diagnosis
B
M

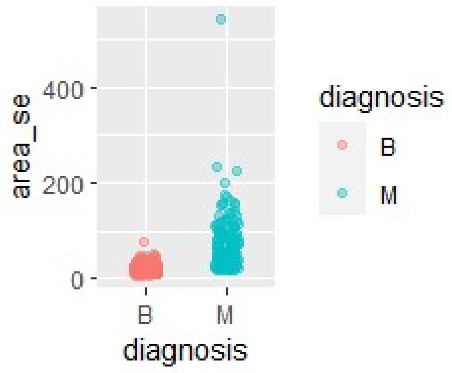

symmetry_mean
0.30
0.25
0.20
0.15
0.10
B
M
diagnosis
diagnosis
B
M

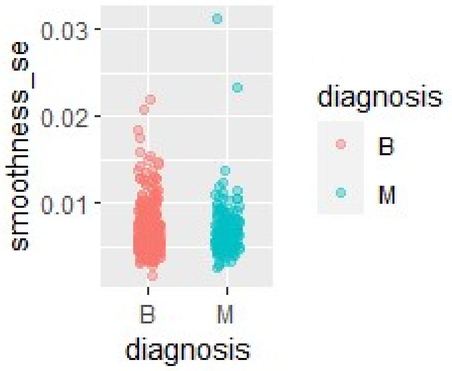

fractal_dimension_mear
0.09
0.08
0.07
0.06
0.05
B
M
diagnosis
diagnosis
B
M

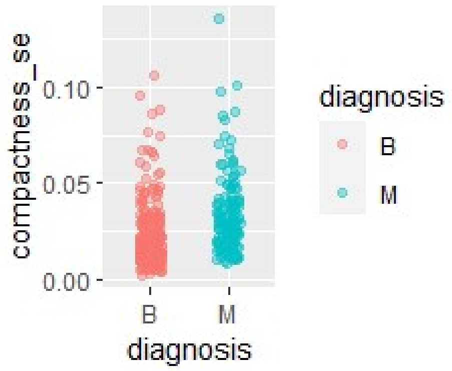

radius_se
2.5
2.0
1.5
1.0
0.5
0.0
B
M
diagnosis
diagnosis
B
M

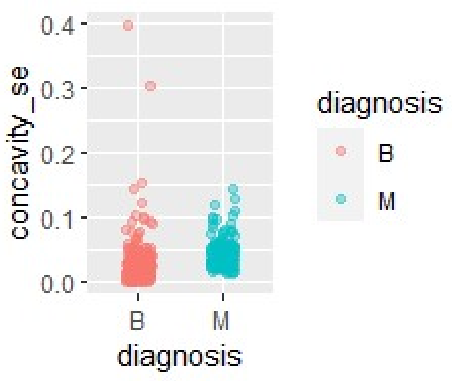

texture_se
5
4
3
2
1
B
M
diagnosis
diagnosis
B
M

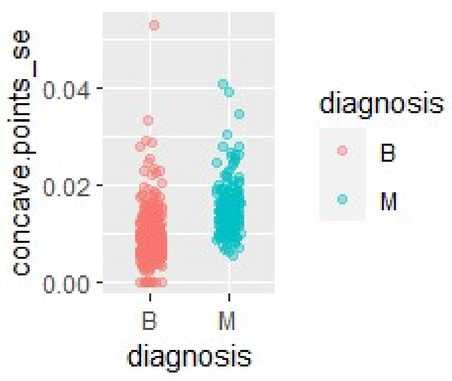

perimeter_se
15
10
5
0
B
M
diagnosis
diagnosis
B
M

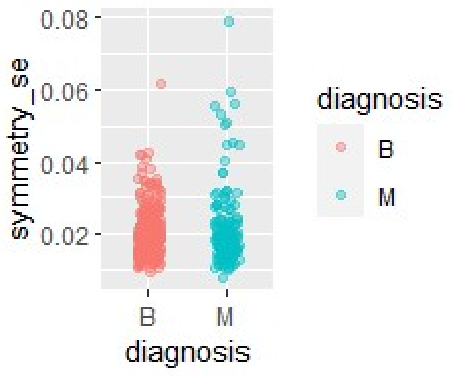

area_se
400
200
0
B
M
diagnosis
diagnosis
B
M

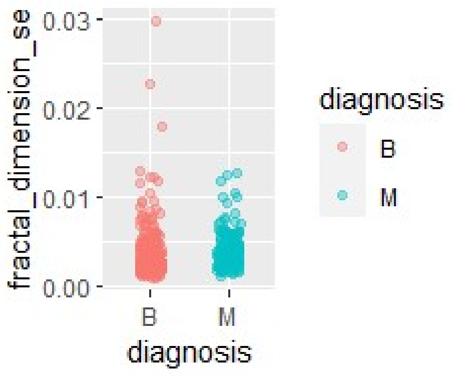

smoothness_se
0.03
0.02
0.01
B
M
diagnosis
diagnosis
B
M

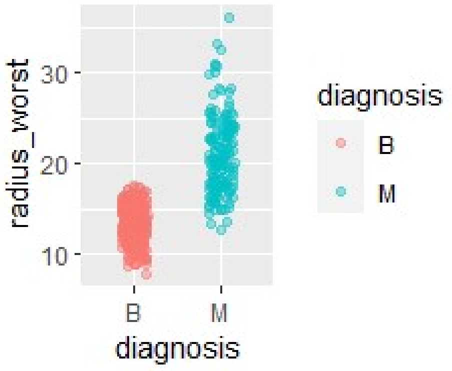

compactness_se
0.10
0.05
0.00
B
M
diagnosis
diagnosis
B
M

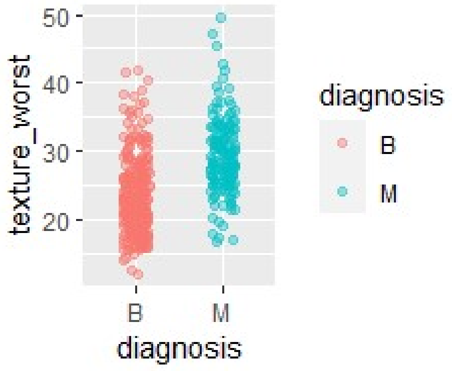

concavity_se
0.0
0.1
0.2
0.3
0.4
B
M
diagnosis
diagnosis
● B
● M

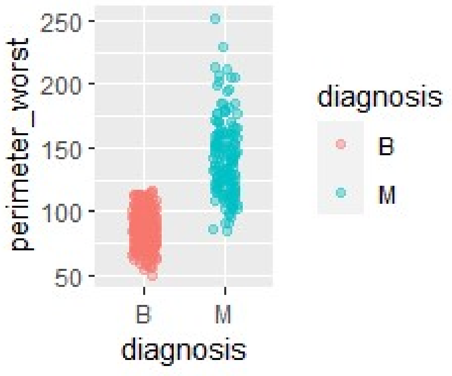

concave.points_se
0.00
0.02
0.04
B
M
diagnosis
diagnosis
● B
● M

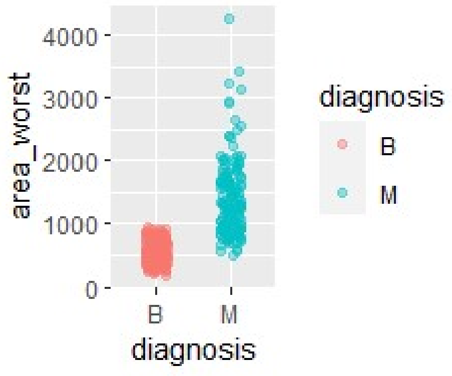

symmetry_se
0.02
0.04
0.06
0.08
B
M
diagnosis
diagnosis
● B
● M

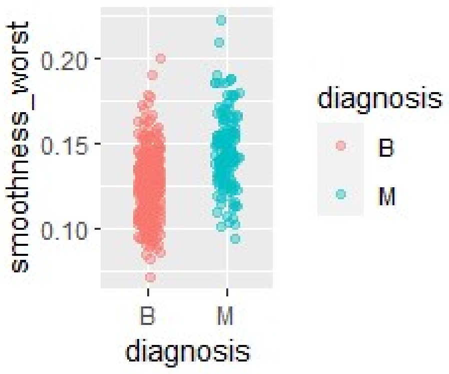

fractal_dimension_se
0.00
0.01
0.02
0.03
B
M
diagnosis
diagnosis
● B
● M

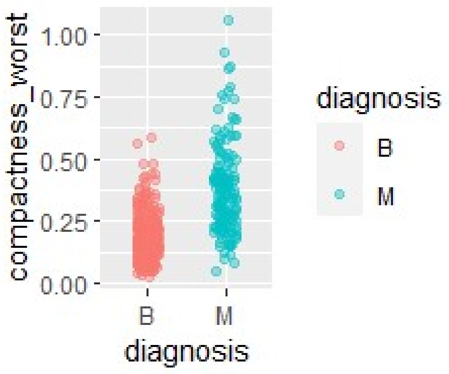

radius_worst
10
20
30
B
M
diagnosis
diagnosis
● B
● M

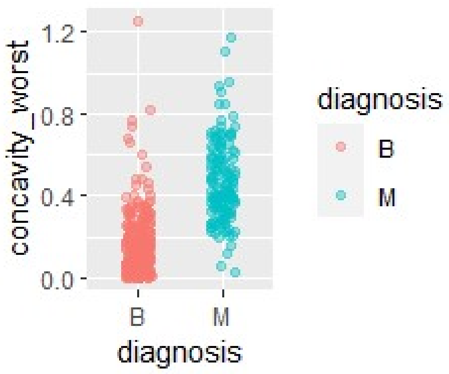

texture_worst
20
30
40
50
B
M
diagnosis
diagnosis
● B
● M

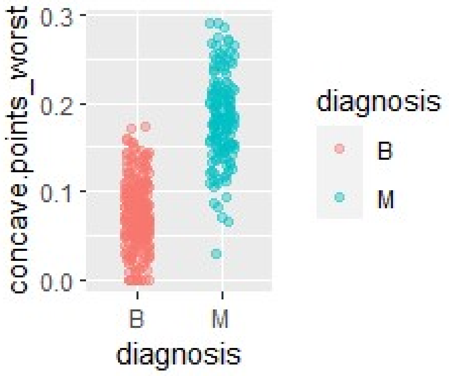

perimeter_worst
50
100
150
200
250
B
M
diagnosis
diagnosis
● B
● M

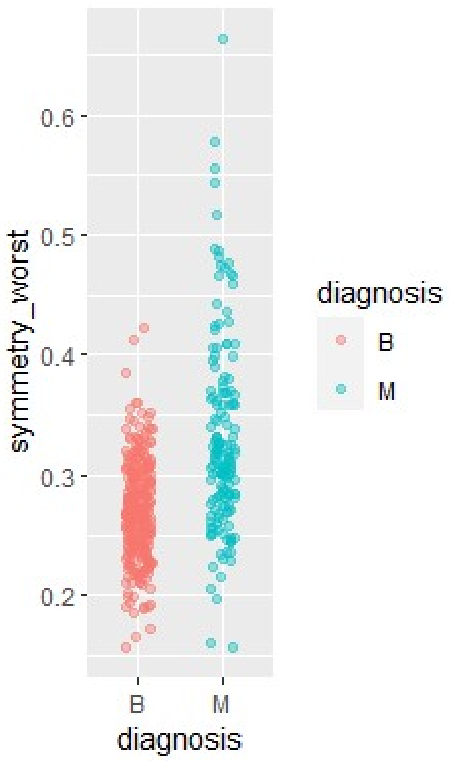

area_worst
0
1000
2000
3000
4000
B
M
diagnosis
diagnosis
● B
● M

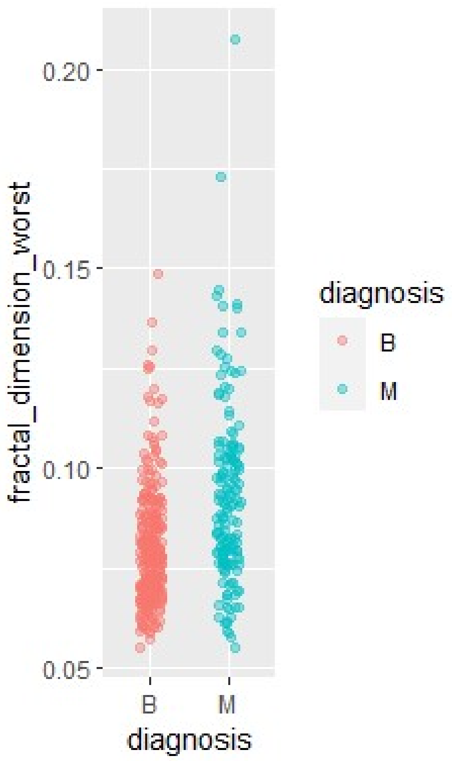

smoothness_worst
0.20
0.15
0.10
B
M
diagnosis
diagnosis
● B
● M

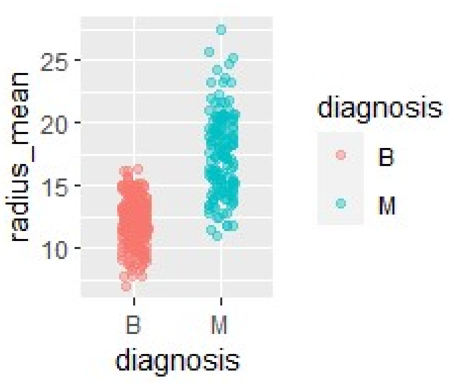

compactness_worst
1.00
0.75
0.50
0.25
0.00
B
M
diagnosis
diagnosis
● B
● M

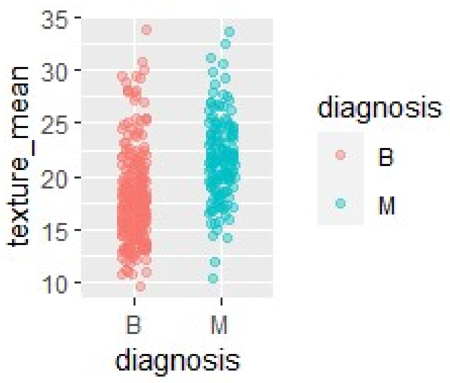

concavity_worst
1.2
0.8
0.4
0.0
B
M
diagnosis
diagnosis
● B
● M

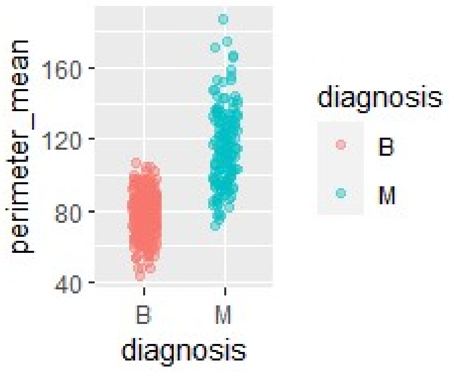

concave.points_worst
0.3
0.2
0.1
0.0
B
M
diagnosis
diagnosis
● B
● M

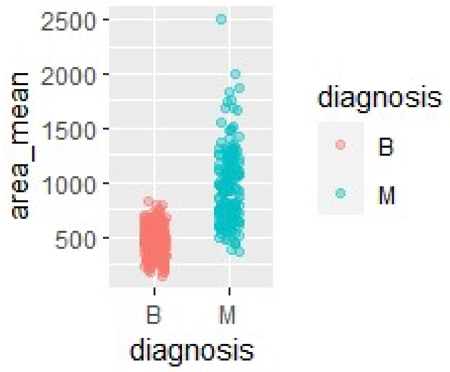

symmetry_worst
0.6
0.5
0.4
0.3
0.2
B
M
diagnosis
diagnosis
● B
● M


fractal_dimension_worst
0.20
0.15
0.10
0.05
B
M
diagnosis
diagnosis
● B
● M

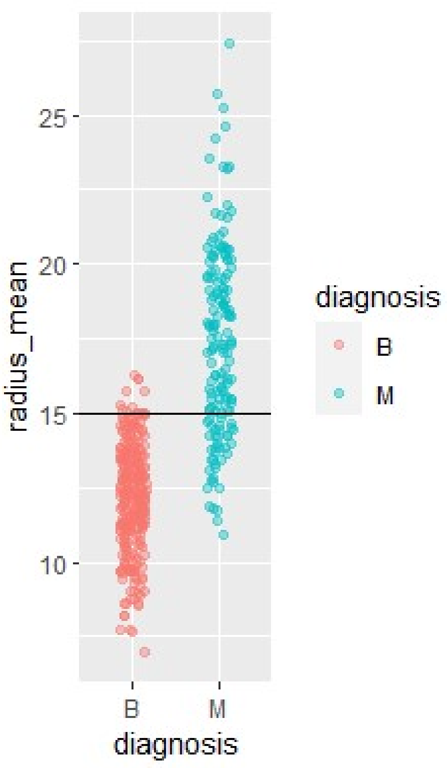


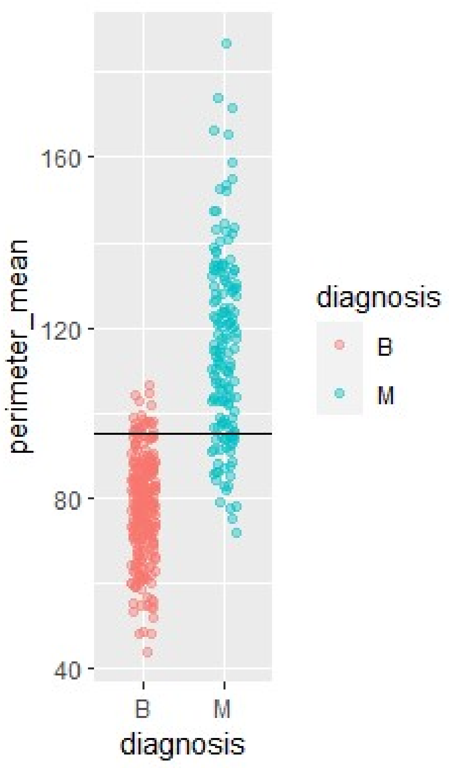


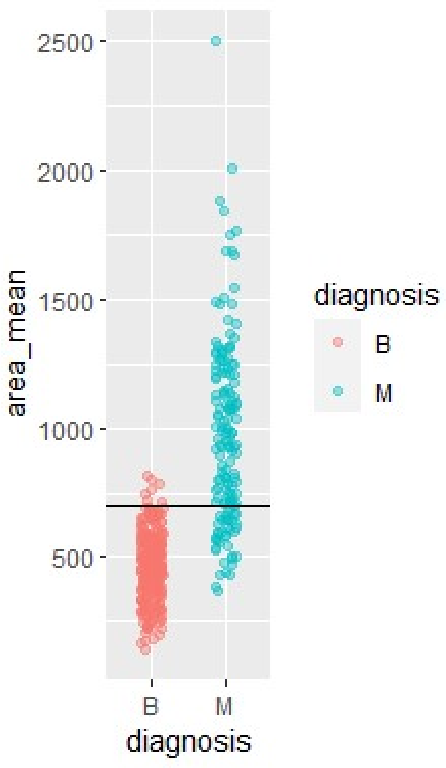


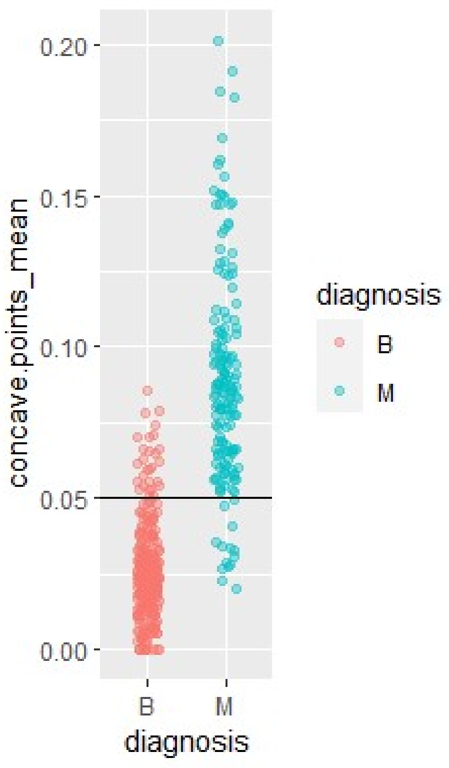


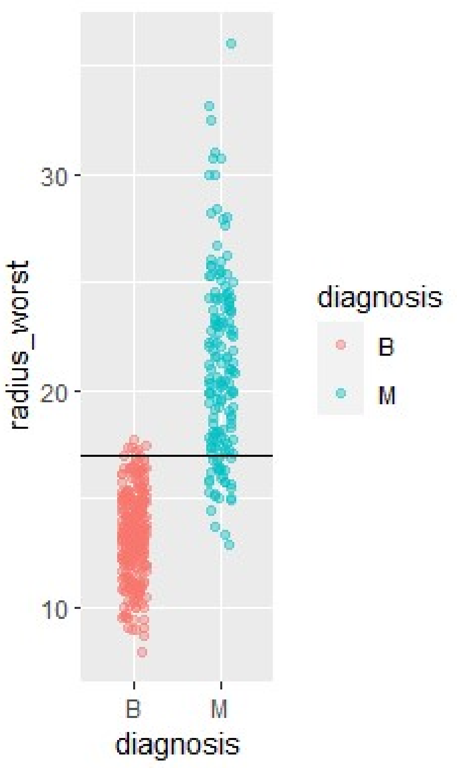
Upon examining the plots, we can see that in most cases, the malignant cancer samples have higher values. I can leverage this difference to create a code that can differentiate between the two types of cancer. However, since some plots show a more significant difference than others, I will only use the plots where the difference is clearer. Then, I will determine a cut-off line that can more accurately separate the samples. Here are the chosen plots and how the cut-off line will be established:

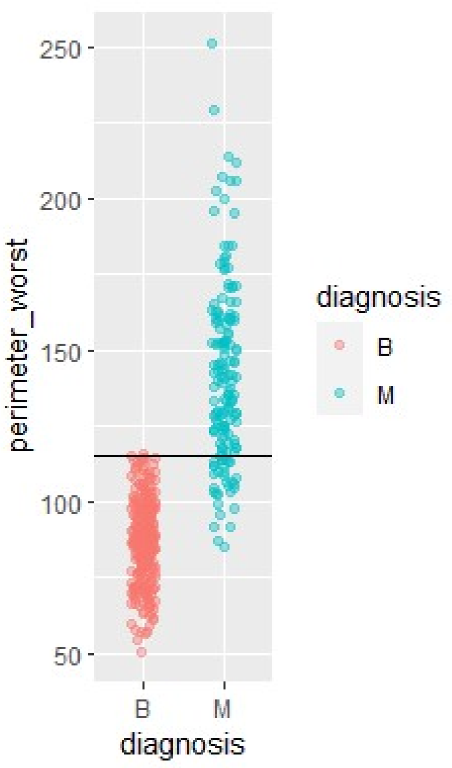

radius_mean
25
20
15
10
B
M
diagnosis
diagnosis
B
M

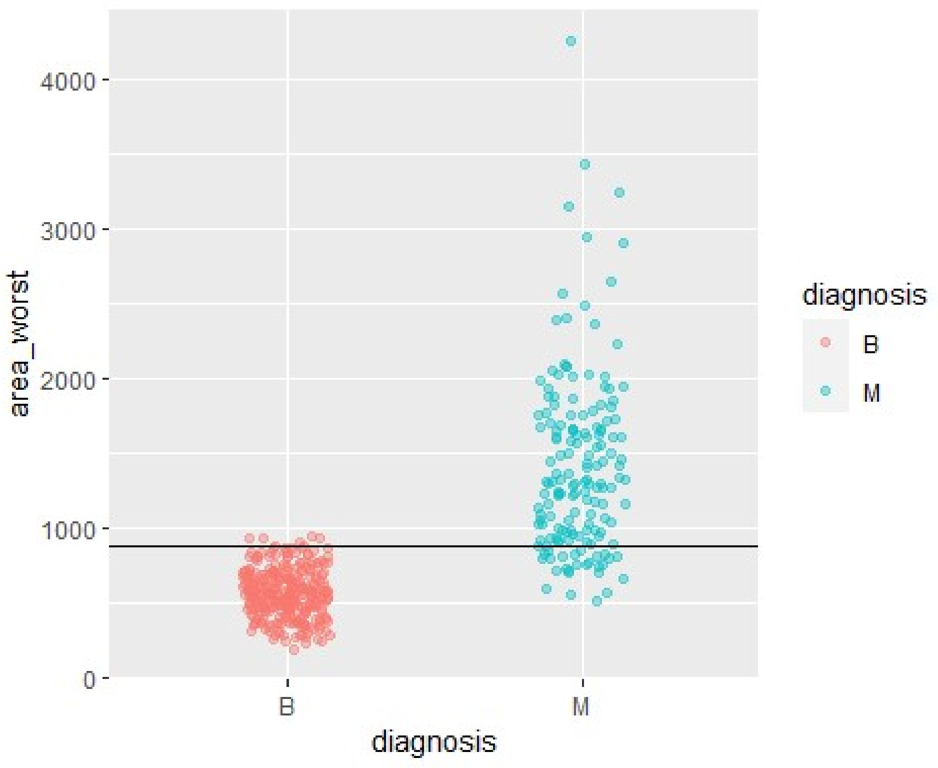

perimeter_mean
160
120
80
40
B
M
diagnosis
diagnosis
B
M


area_mean
2500
2000
1500
1000
500
B
M
diagnosis
diagnosis
B
M


concave.points_mean
0.20
0.15
0.10
0.05
0.00
B
M
diagnosis
diagnosis
B
M



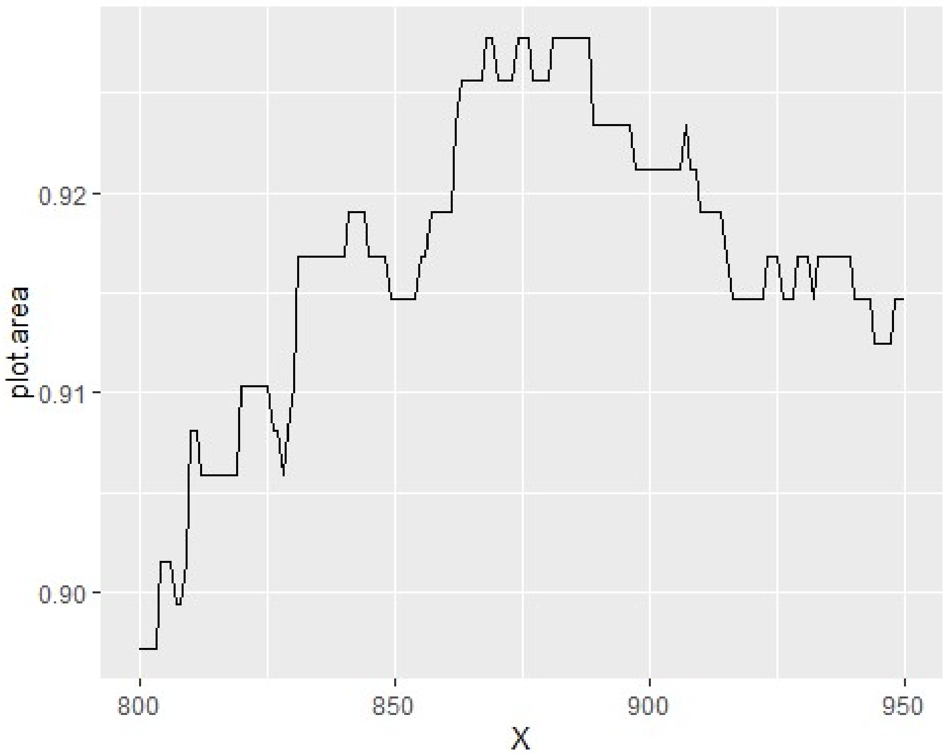




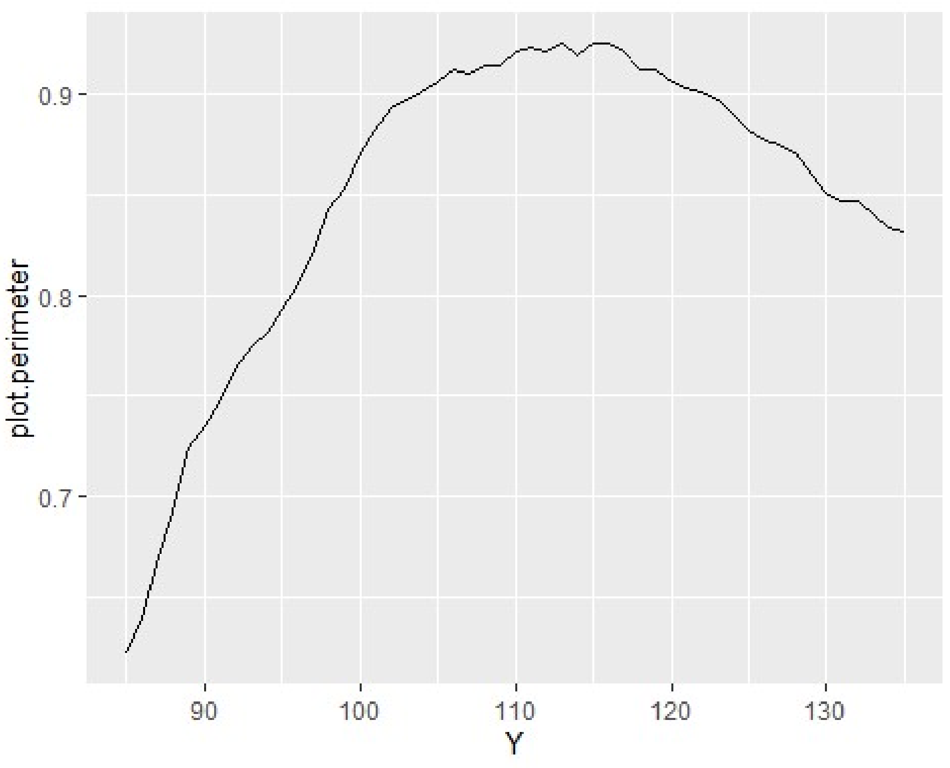
Using these cut-off lines, I will calculate the accuracy of my model if I decide to classify all points above it as 'M' and all below as 'B'

```
## [1] "The accuracy in the radius_mean plot is -"
## [2] "0.892778993435449"

## [1] "The accuracy in the perimeter_mean plot is -"
## [2] "0.87527352297593"

## [1] "The accuracy in the area_mean plot is -"
## [2] "0.897155361050328"

## [1] "The accuracy in the concave.points_mean plot is -"
## [2] "0.908096280087527"

## [1] "The accuracy in the radius_worst plot is -"
## [2] "0.923413566739606"

## [1] "The accuracy in the perimeter_worst plot is -"
## [2] "0.925601750547046"

## [1] "The accuracy in the area_wors plot is -"
## [2] "0.925601750547046"
```

The columns **'perimeter_worst'** and **'area_worst'** provide the best results. Now, I'm going to test other cuts on the same columns to see if we can achieve even higher accuracy.



```
## [1] "Cut in area_worst" "868"

## [1] "Accuracy in area_worst" "0.927789934354486"
```



```
## [1] "Cut in perimeter_worst" "113"

## [1] "Accuracy in perimeter_worst" "0.925601750547046"
```

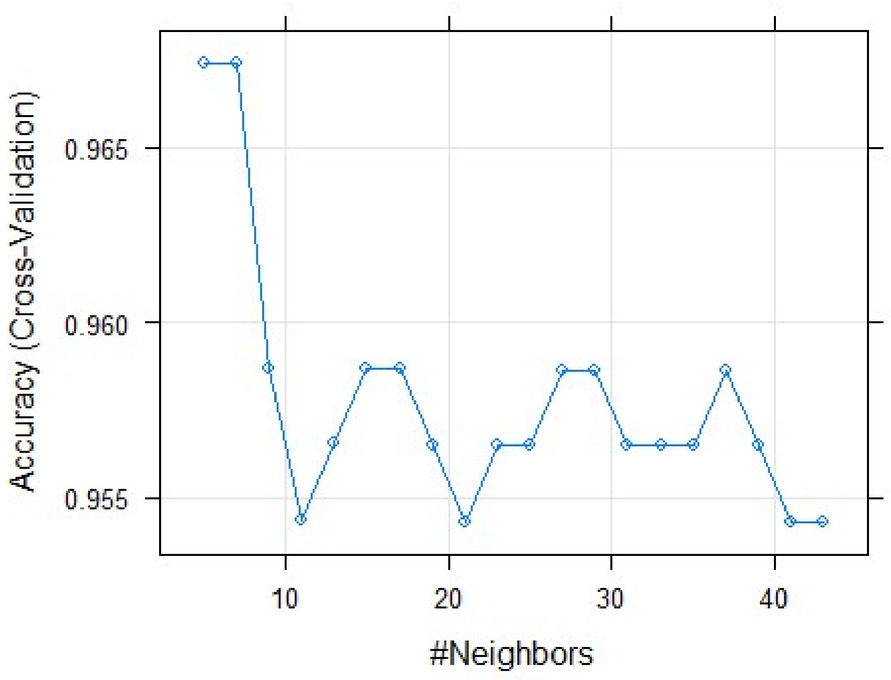
As the **area_worst** had the best result, I'm going to use it to predict if the cancer samples in to test_set is 'M' or 'B'.

```
resultarea_testset <- as.factor(ifelse(test_set$area_worst > 868, "M", "B"))
confusionMatrix(resultarea_testset, test_set$diagnosis, positive = "M")

## Confusion Matrix and Statistics
##
##           Reference
## Prediction  B  M
##          B 59  9
##          M  3 41
##
##                Accuracy : 0.8929
##                  95% CI : (0.8203, 0.9434)
##     No Information Rate : 0.5536
##     P-Value [Acc > NIR] : 6.723e-15
##
##                   Kappa : 0.7807
##
##  Mcnemar's Test P-Value : 0.1489
##
##             Sensitivity : 0.8200
##             Specificity : 0.9516
##          Pos Pred Value : 0.9318
##          Neg Pred Value : 0.8676
##              Prevalence : 0.4464
##          Detection Rate : 0.3661
##    Detection Prevalence : 0.3929
##       Balanced Accuracy : 0.8858
##
##        'Positive' Class : M
##
```

K-nearest neighbors

In the second phase of this report, I will use a more sophisticated approach to achieve better results. To do this, I have chosen to use the K-nearest neighbors (KNN) algorithm and cross-validation methods. IBM classifies KNN as “a non-parametric, supervised learning classifier that uses proximity to make classifications or predictions about the grouping of individual data points”.

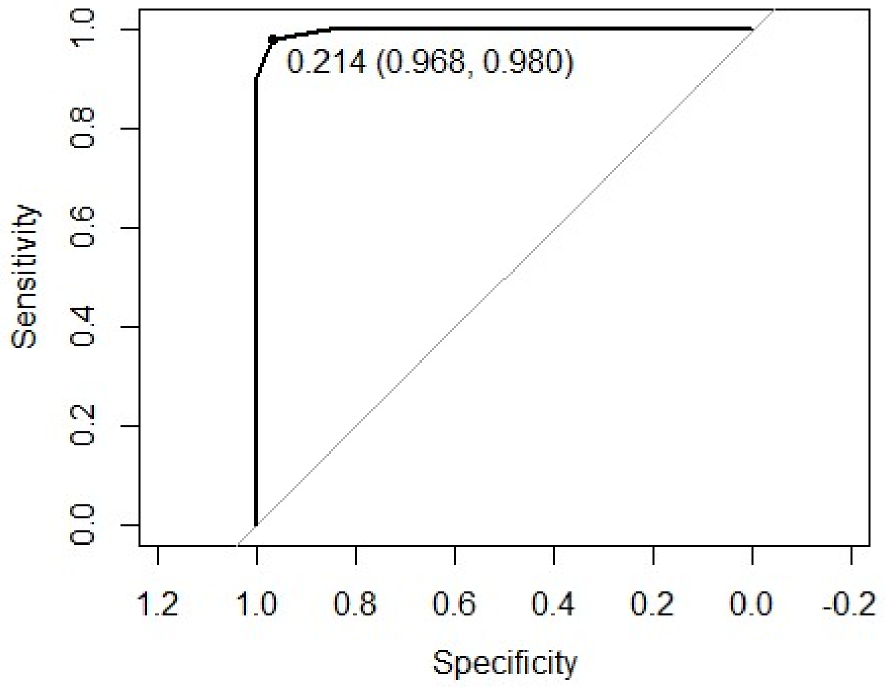
```
ctrl <- trainControl(method = "cv", #Here we choose the cross-validation method
                     number = 10)  #Here we decide in how many folds we will split the sample

knnFit <- train(diagnosis ~ .-id,
                method = "knn",
                preProcess = c("center","scale"),
                tuneLength = 20, #try 20 values for k
                trControl = ctrl,
                metric = "Accuracy",
                data = train_set)
knnFit$finalModel

## 7-nearest neighbor model
## Training set outcome distribution:
##
##   B   M
## 295 162
```



The 7-nearest neighbor model achieved the best result. Now let's use this result on the test data to calculate the accuracy of the predictions.

```
predknn <- predict(knnFit, test_set, type = "prob")
resultknn <- as.factor(ifelse(predknn[,2] > 0.5, "M", "B")) # If the chances
of a cancer is "M" is higher than 50% we are going to chose "M".

confusionMatrix(resultknn, test_set$diagnosis, positive = "M")

## Confusion Matrix and Statistics
##
##           Reference
## Prediction  B  M
##          B 62  5
##          M  0 45
##
##                Accuracy : 0.9554
##                  95% CI : (0.8989, 0.9853)
##     No Information Rate : 0.5536
##     P-Value [Acc > NIR] : < 2e-16
##
##                   Kappa : 0.9088
##
##  Mcnemar's Test P-Value : 0.07364
##
##             Sensitivity : 0.9000
##             Specificity : 1.0000
##          Pos Pred Value : 1.0000
##          Neg Pred Value : 0.9254
##              Prevalence : 0.4464
##          Detection Rate : 0.4018
##    Detection Prevalence : 0.4018
##       Balanced Accuracy : 0.9500
##
##        'Positive' Class : M
##
```

We have already exceeded the accuracy of the first model. However, we can still improve the results by using the **ROC curve**. Currently, the code predicts that if the chance of being 'M' is higher than 50%, it should choose 'M'. But now we are going to determine whether 50% is the best threshold for making predictions.



The plot shows that if the chance of the diagnosis being 'M' is higher than 0.214, it should select 'M'. Let's see if this actually improves the accuracy:

```
resultknn <- as.factor(ifelse(predknn[,2] > 0.214, "M", "B"))
confusionMatrix(resultknn, test_set$diagnosis, positive = "M")
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  B  M
##          B 60  1
##          M  2 49
##
```

```
##                Accuracy : 0.9732
##                  95% CI : (0.9237, 0.9944)
##     No Information Rate : 0.5536
##     P-Value [Acc > NIR] : <2e-16
##
##                   Kappa : 0.9459
##
##  Mcnemar's Test P-Value : 1
##
##             Sensitivity : 0.9800
##             Specificity : 0.9677
##          Pos Pred Value : 0.9608
##          Neg Pred Value : 0.9836
##              Prevalence : 0.4464
##          Detection Rate : 0.4375
##    Detection Prevalence : 0.4554
##       Balanced Accuracy : 0.9739
##
##        'Positive' Class : M
##
```

Results

```
## # A tibble: 2 × 2
##   Model_Type   Accuracy
##   <chr>        <chr>
## 1 Cut-off line 0.8929
## 2 K-NN         0.9732
```

The cut-off line model has a good accuracy of 89.29%, but the K-NN model turns to be a better model, reaching 97.32%.

Conclusion

The first model uses only the cut-off line from one column to evaluate if a cancer sample is malignant or benign. The problem with this model is that sometimes, in all the other columns, the sample has values that are compatible with its classification, but in that one column, the values have a bias that can be enough to cause the wrong classification.

One of the reasons why K-NN is considered the better model is that it uses the information from all the columns to predict in which category the sample belongs. Even if one column's value is out of the norm, the model can still fit the sample into the correct category. Furthermore, this model predicts the category based on other samples with similar values, which makes it more flexible and adjustable as we add more samples to the dataset. In the cut-off line model, we would have to recalculate the value to split the data and change the value manually.