CANCER TYPE DETECTION

William C.C. Data Science Profesional

Introduction

Project Overview

This project is developed in order to obtain the harvardX professional certification title. The main idea of the project is to have a Cancer dataset, which has 32 characteristics and the idea is to be able to train each of the algorithms seen in classes to be able to predict the type of cancer Benign (B), Malignant (M).

Despite the fact that the code is very easy to visualize and understand, each of the developed pazos will be explained. The code has been divided into the following parts:

1. Project initialization

2. Dataset exploration1

3. Building machine learning algorithms

4. Overview of results

Project initialization

The data set used is hosted at the address such, which has 32 characteristics to take into account of which the first characteristic is not significant since it is used simply with an index; The second characteristic is diagnostic and it is the characteristic that will be used in the output, that is, for training. For the training two classes Benigno (B) and Maligno (M) will be taken into account, classes that are located in the Diagnosis column.

The columns of the data are listed below:

1- id

2- diagnosis

3- radius_mean

4- texture_mean

5- perimeter_mean

6- area_mean

7- smoothness_mean

8- compactness_mean

9- concavity_mean

10- concave.points_mean

11- symmetry_mean

12- fractal_dimension_mean

13- radius_se

14- texture_se

15- perimeter_se

16- area_se

17- smoothness_se

18- compactness_se

19- concavity_se

20- concave.points_se

21- symmetry_se

22- fractal_dimension_se

23- radius_worst

24- texture_worst

25- perimeter_worst

26- area_worst

27- smoothness_worst

28- compactness_worst

29- concavity_worst

30- concave.points_worst

31- symmetry_worst

32- fractal_dimension_worst

This dataset has 569 observations.

The objective of this work is to predict the type of cancer that a person has according to its characteristics.

Part 1 - Project initialization Install libraries

```
## Loading required package: tidyverse

## -- Attaching packages -----------------------------------------------------------------------------------
-------- tidyverse 1.3.0 --

## v ggplot2 3.2.1     v purrr   0.3.3
## v tibble  2.1.3     v dplyr   0.8.4
## v tidyr   1.0.2     v stringr 1.4.0
## v readr   1.3.1     v forcats 0.4.0

## -- Conflicts ------------------------------------------------------------------------------------------ t
idyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()

## Loading required package: caret

## Loading required package: lattice

##
## Attaching package: 'caret'

## The following object is masked from 'package:purrr':
##
##     lift

## Loading required package: data.table

##
## Attaching package: 'data.table'

## The following objects are masked from 'package:dplyr':
##
##     between, first, last

## The following object is masked from 'package:purrr':
##
##     transpose

## Loading required package: ranger

## Warning: package 'ranger' was built under R version 3.6.3

## Loading required package: naivebayes

## naivebayes 0.9.6 loaded

##
## Attaching package: 'naivebayes'

## The following object is masked from 'package:data.table':
##
##     tables
```

```
## Loading required package: kernlab

##
## Attaching package: 'kernlab'

## The following object is masked from 'package:purrr':
##
##     cross

## The following object is masked from 'package:ggplot2':
##
##     alpha

## Loading required package: ggthemes

## Loading required package: knitr

## Warning: package 'knitr' was built under R version 3.6.3

## Loading required package: e1071
```

Now that the dataset is loaded, we proceed to analyze the information.

Methods and data analysis

This project will use the following algorithms:

1. Generalized Linear Model (method = "glm")
2. k-Nearest Neighbors (method = "knn")
3. Random Forest (method = "ranger", additional library "ranger")
4. Naive Bayes (method = "naive_bayes", additional library "naivebayes")
5. Support Vector Machines with Polynomial Kernel (method = "svmPoly", additional library "kernlab")

At the end we will give the score of each of the algorithms to identify which one behaves best.

Overview of the dataset

```
##         id diagnosis radius_mean texture_mean perimeter_mean area_mean
## 1   842302         M       17.99        10.38         122.80    1001.0
## 2   842517         M       20.57        17.77         132.90    1326.0
## 3 84300903         M       19.69        21.25         130.00    1203.0
## 4 84348301         M       11.42        20.38          77.58     386.1
## 5 84358402         M       20.29        14.34         135.10    1297.0
## 6   843786         M       12.45        15.70          82.57     477.1
##   smoothness_mean compactness_mean concavity_mean concave.points_mean
## 1         0.11840          0.27760         0.3001             0.14710
## 2         0.08474          0.07864         0.0869             0.07017
## 3         0.10960          0.15990         0.1974             0.12790
## 4         0.14250          0.28390         0.2414             0.10520
```

```
## 5        0.10030          0.13280         0.1980            0.10430
## 6        0.12780          0.17000         0.1578            0.08089
##   symmetry_mean fractal_dimension_mean radius_se texture_se perimeter_se
## 1        0.2419                 0.07871    1.0950     0.9053        8.589
## 2        0.1812                 0.05667    0.5435     0.7339        3.398
## 3        0.2069                 0.05999    0.7456     0.7869        4.585
## 4        0.2597                 0.09744    0.4956     1.1560        3.445
## 5        0.1809                 0.05883    0.7572     0.7813        5.438
## 6        0.2087                 0.07613    0.3345     0.8902        2.217
##   area_se smoothness_se compactness_se concavity_se concave.points_se
## 1  153.40      0.006399        0.04904      0.05373           0.01587
## 2   74.08      0.005225        0.01308      0.01860           0.01340
## 3   94.03      0.006150        0.04006      0.03832           0.02058
## 4   27.23      0.009110        0.07458      0.05661           0.01867
## 5   94.44      0.011490        0.02461      0.05688           0.01885
## 6   27.19      0.007510        0.03345      0.03672           0.01137
##   symmetry_se fractal_dimension_se radius_worst texture_worst perimeter_worst
## 1     0.03003             0.006193        25.38         17.33          184.60
## 2     0.01389             0.003532        24.99         23.41          158.80
## 3     0.02250             0.004571        23.57         25.53          152.50
## 4     0.05963             0.009208        14.91         26.50           98.87
## 5     0.01756             0.005115        22.54         16.67          152.20
## 6     0.02165             0.005082        15.47         23.75          103.40
##   area_worst smoothness_worst compactness_worst concavity_worst
## 1     2019.0           0.1622            0.6656          0.7119
## 2     1956.0           0.1238            0.1866          0.2416
## 3     1709.0           0.1444            0.4245          0.4504
## 4      567.7           0.2098            0.8663          0.6869
## 5     1575.0           0.1374            0.2050          0.4000
## 6      741.6           0.1791            0.5249          0.5355
##   concave.points_worst symmetry_worst fractal_dimension_worst
## 1               0.2654         0.4601                 0.11890
## 2               0.1860         0.2750                 0.08902
## 3               0.2430         0.3613                 0.08758
## 4               0.2575         0.6638                 0.17300
## 5               0.1625         0.2364                 0.07678
## 6               0.1741         0.3985                 0.12440
```

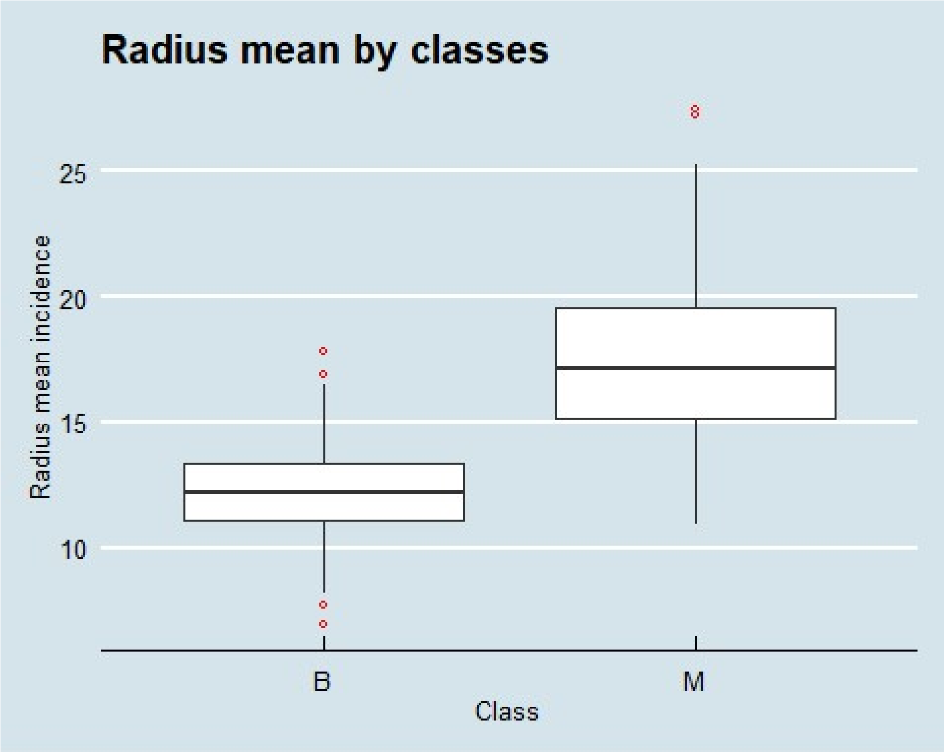
Summary of our dataset.

```
##        id            diagnosis  radius_mean      texture_mean
##  Min.   :     8915   B:267     Min.   : 6.981   Min.   : 9.71
##  1st Qu.:   869751   M:159     1st Qu.:11.623   1st Qu.:16.15
##  Median :   905511             Median :13.320   Median :18.84
##  Mean   : 29931248             Mean   :14.106   Mean   :19.38
##  3rd Qu.:  8712619             3rd Qu.:15.832   3rd Qu.:21.89
##  Max.   :911320501             Max.   :27.420   Max.   :39.28
##  perimeter_mean     area_mean      smoothness_mean   compactness_mean
##  Min.   : 43.79   Min.   : 143.5   Min.   :0.05263   Min.   :0.01938
##  1st Qu.: 74.72   1st Qu.: 413.3   1st Qu.:0.08590   1st Qu.:0.06313
##  Median : 86.04   Median : 546.2   Median :0.09587   Median :0.09453
```

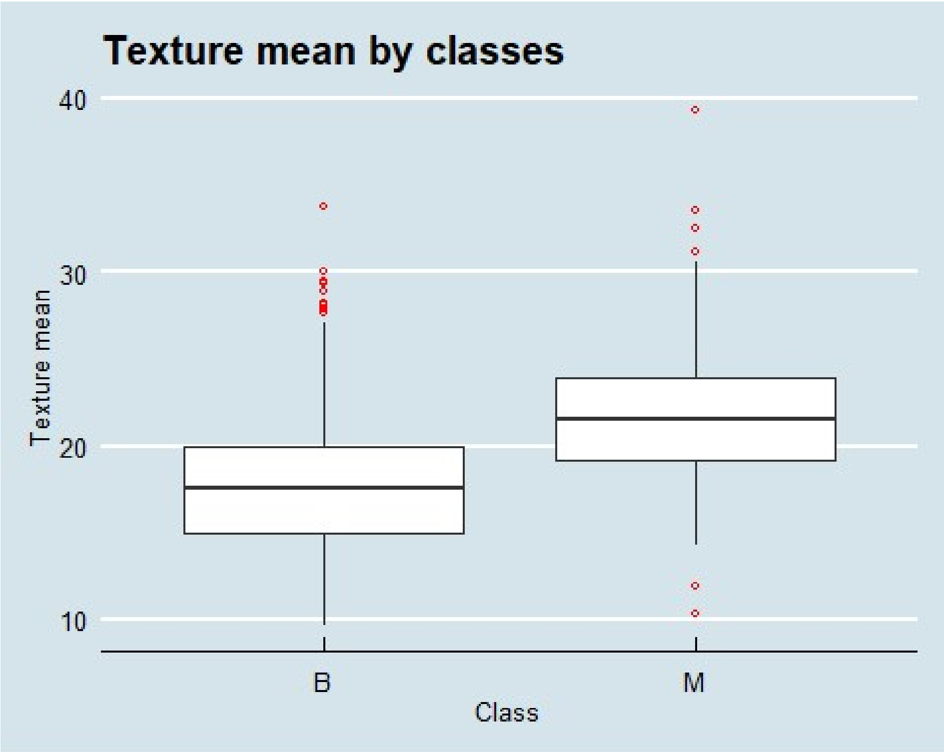
```
##  Mean   : 91.82   Mean   : 651.9   Mean   :0.09622   Mean   :0.10398
##  3rd Qu.:104.25   3rd Qu.: 787.0   3rd Qu.:0.10510   3rd Qu.:0.12962
##  Max.   :186.90   Max.   :2501.0   Max.   :0.16340   Max.   :0.34540
##  concavity_mean    concave.points_mean symmetry_mean    fractal_dimension_mean
##  Min.   :0.00000   Min.   :0.00000     Min.   :0.1060   Min.   :0.04996
##  1st Qu.:0.02924   1st Qu.:0.02032     1st Qu.:0.1618   1st Qu.:0.05769
##  Median :0.06155   Median :0.03338     Median :0.1787   Median :0.06132
##  Mean   :0.08875   Mean   :0.04893     Mean   :0.1810   Mean   :0.06274
##  3rd Qu.:0.12878   3rd Qu.:0.07385     3rd Qu.:0.1958   3rd Qu.:0.06607
##  Max.   :0.42680   Max.   :0.20120     Max.   :0.3040   Max.   :0.09744
##    radius_se        texture_se      perimeter_se       area_se
##  Min.   :0.1115   Min.   :0.3602   Min.   : 0.757   Min.   :  6.802
##  1st Qu.:0.2298   1st Qu.:0.8289   1st Qu.: 1.603   1st Qu.: 17.688
##  Median :0.3144   Median :1.0810   Median : 2.251   Median : 23.875
##  Mean   :0.4031   Mean   :1.2228   Mean   : 2.848   Mean   : 40.165
##  3rd Qu.:0.4858   3rd Qu.:1.4715   3rd Qu.: 3.359   3rd Qu.: 46.410
##  Max.   :2.5470   Max.   :4.8850   Max.   :18.650   Max.   :542.200
##  smoothness_se      compactness_se      concavity_se     concave.points_se
##  Min.   :0.001713   Min.   :0.002252   Min.   :0.00000   Min.   :0.000000
##  1st Qu.:0.005213   1st Qu.:0.012982   1st Qu.:0.01501   1st Qu.:0.007586
##  Median :0.006329   Median :0.020460   Median :0.02517   Median :0.010870
##  Mean   :0.007019   Mean   :0.025540   Mean   :0.03160   Mean   :0.011799
##  3rd Qu.:0.008107   3rd Qu.:0.032835   3rd Qu.:0.04296   3rd Qu.:0.014698
##  Max.   :0.031130   Max.   :0.106400   Max.   :0.30380   Max.   :0.040900
##   symmetry_se       fractal_dimension_se  radius_worst   texture_worst
##  Min.   :0.007882   Min.   :0.0008948    Min.   : 7.93   Min.   :12.02
##  1st Qu.:0.015003   1st Qu.:0.0022125    1st Qu.:12.92   1st Qu.:21.32
##  Median :0.018710   Median :0.0031260    Median :14.91   Median :25.43
##  Mean   :0.020348   Mean   :0.0037794    Mean   :16.25   Mean   :25.83
##  3rd Qu.:0.022927   3rd Qu.:0.0045473    3rd Qu.:18.80   3rd Qu.:30.07
##  Max.   :0.078950   Max.   :0.0228600    Max.   :36.04   Max.   :49.54
##  perimeter_worst    area_worst     smoothness_worst  compactness_worst
##  Min.   : 50.41   Min.   : 185.2   Min.   :0.07117   Min.   :0.02729
##  1st Qu.: 83.77   1st Qu.: 511.0   1st Qu.:0.11540   1st Qu.:0.14735
##  Median : 97.18   Median : 682.5   Median :0.13145   Median :0.21165
##  Mean   :107.15   Mean   : 878.9   Mean   :0.13229   Mean   :0.25482
##  3rd Qu.:126.60   3rd Qu.:1087.0   3rd Qu.:0.14583   3rd Qu.:0.33860
##  Max.   :251.20   Max.   :4254.0   Max.   :0.22260   Max.   :1.05800
##  concavity_worst  concave.points_worst symmetry_worst   fractal_dimension_worst
##  Min.   :0.0000   Min.   :0.00000      Min.   :0.1565   Min.   :0.05504
##  1st Qu.:0.1090   1st Qu.:0.06498      1st Qu.:0.2477   1st Qu.:0.07117
##  Median :0.2290   Median :0.09860      Median :0.2817   Median :0.08002
##  Mean   :0.2734   Mean   :0.11506      Mean   :0.2902   Mean   :0.08416
##  3rd Qu.:0.3872   3rd Qu.:0.16510      3rd Qu.:0.3186   3rd Qu.:0.09259
##  Max.   :1.2520   Max.   :0.29100      Max.   :0.6638   Max.   :0.20750
```

This dataset from this library is clean and we do not need to pre-process to perform our analysis. Next we will begin to explore some variables of our dataset, to observe the behavior of each one of the variables:

Radius mean by classes chart

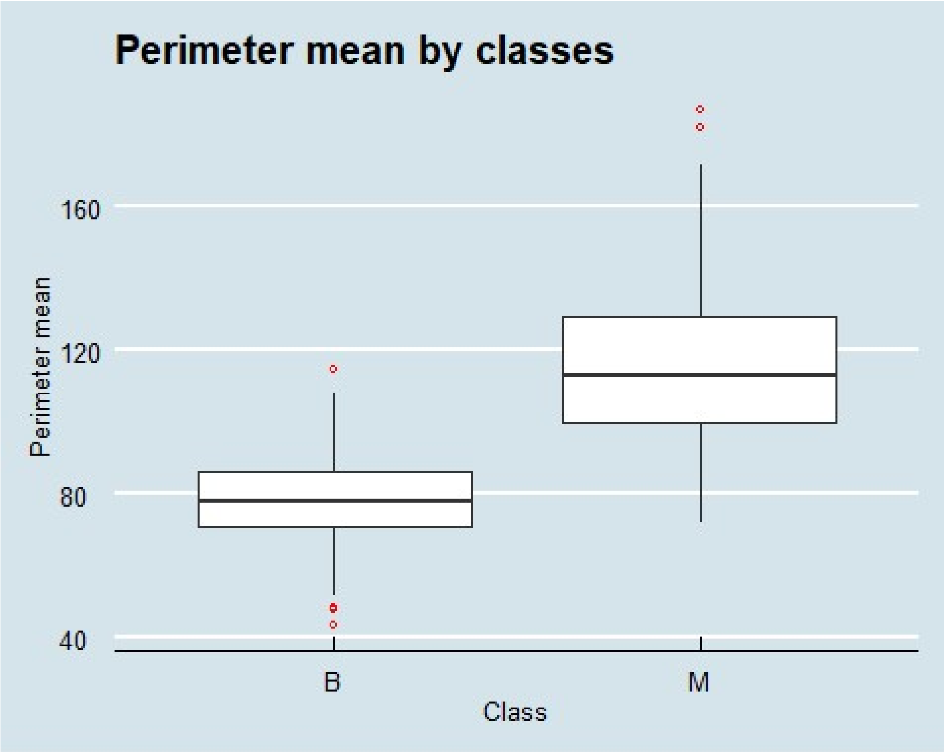


Texture mean by classes chart:



Perimeter mean by classes chart:

```
train %>%
  ggplot(aes(x=diagnosis, y=train[,5])) +
  geom_boxplot(outlier.colour="red",
               outlier.shape=1,
               outlier.size=1) +
  xlab('Class') +
  ylab('Perimeter mean') +
  ggtitle('Perimeter mean by classes') +
  theme_economist()
```

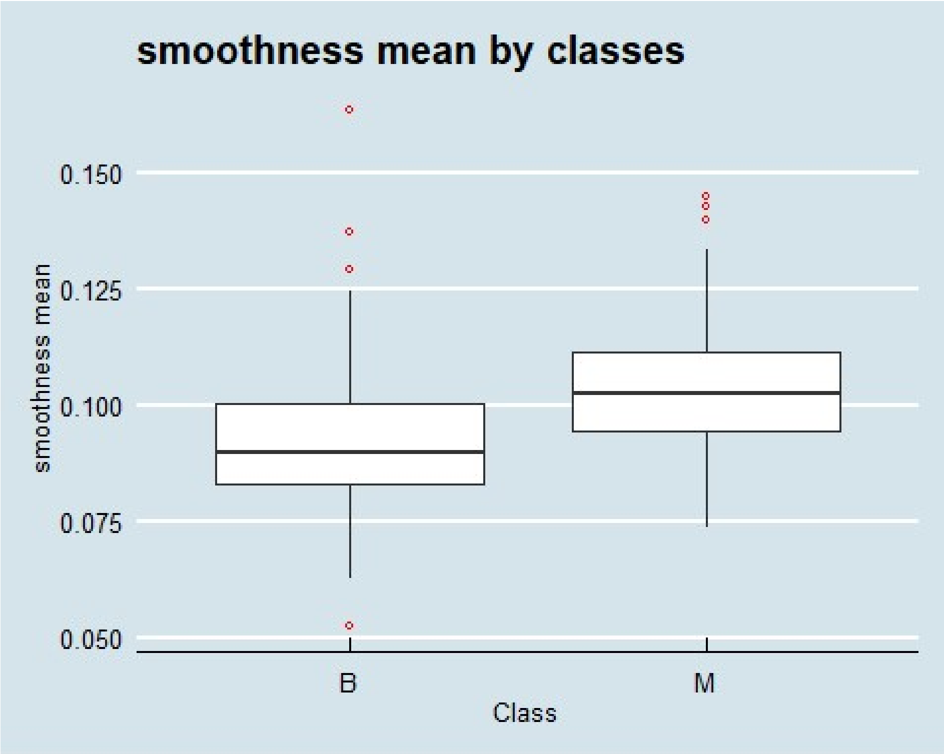


Area mean by classes chart:

```
train %>%
  ggplot(aes(x=diagnosis, y=train[,6])) +
  geom_boxplot(outlier.colour="red",
               outlier.shape=1,
               outlier.size=1) +
  xlab('Class') +
  ylab('Area mean') +
  ggtitle('Area mean by classes') +
  theme_economist()
```

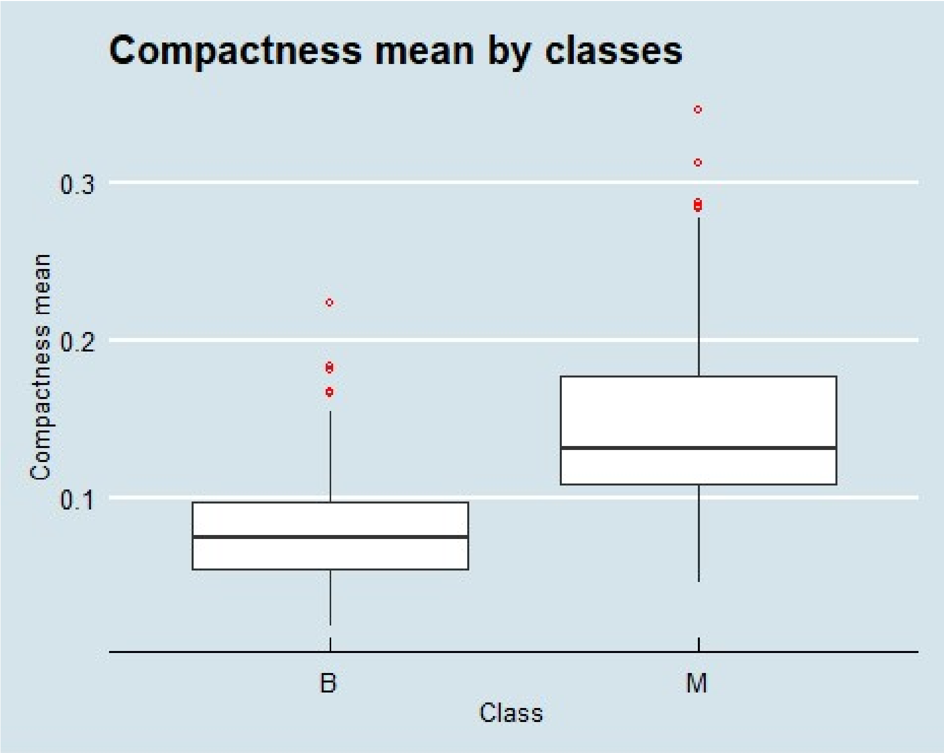
smoothness mean by classes chart:

```
train %>%
  ggplot(aes(x=diagnosis, y=train[,7])) +
  geom_boxplot(outlier.colour="red",
             outlier.shape=1,
             outlier.size=1) +
  xlab('Class') +
  ylab('smoothness mean') +
  ggtitle('smoothness mean by classes') +
  theme_economist()
```



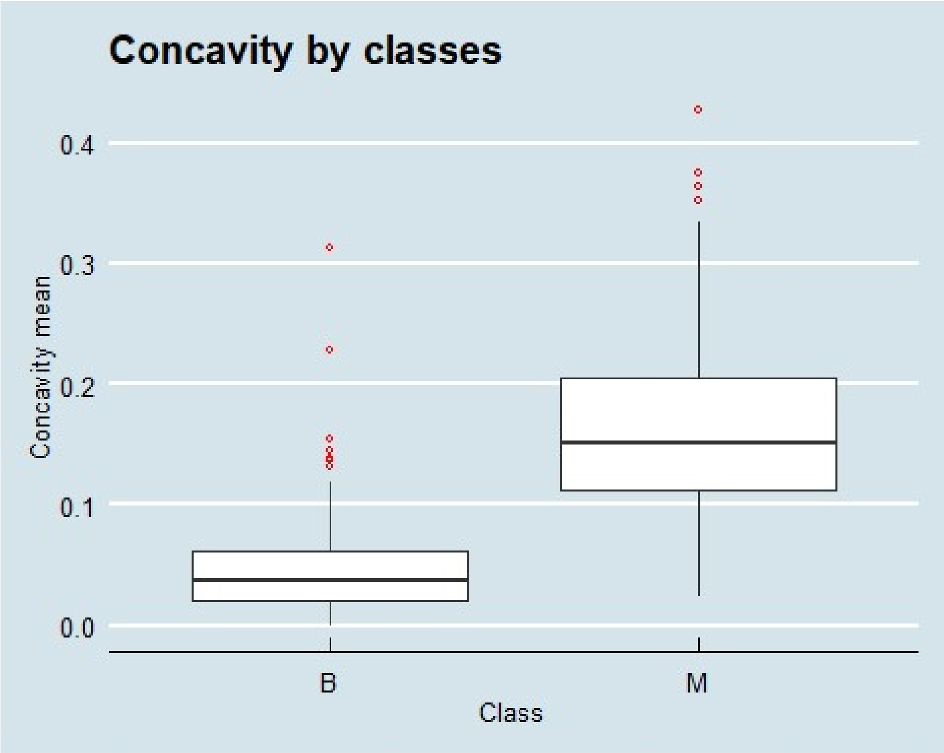
Compactness mean by classes chart:

```
train %>%
  ggplot(aes(x=diagnosis, y=train[,8])) +
  geom_boxplot(outlier.colour="red",
              outlier.shape=1,
              outlier.size=1) +
  xlab('Class') +
  ylab('Compactness mean') +
  ggtitle('Compactness mean by classes') +
  theme_economist()
```



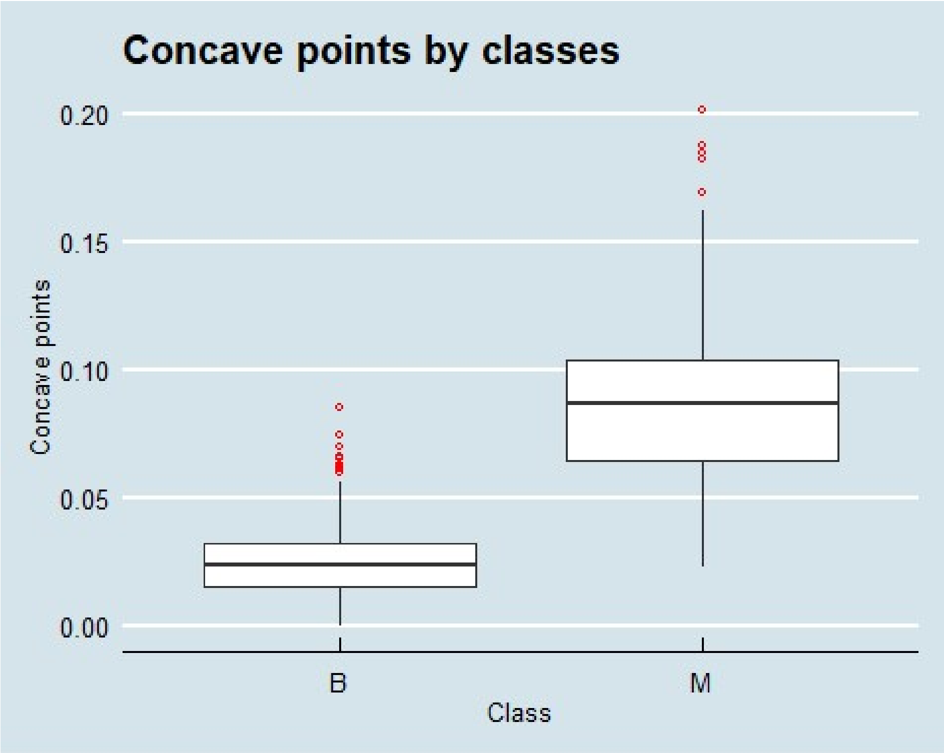
Concavity mean by classes chart:

```
train %>%
  ggplot(aes(x=diagnosis, y=train[,9])) +
  geom_boxplot(outlier.colour="red",
               outlier.shape=1,
               outlier.size=1) +
  xlab('Class') +
  ylab('Concavity mean') +
  ggtitle('Concavity by classes') +
  theme_economist()
```



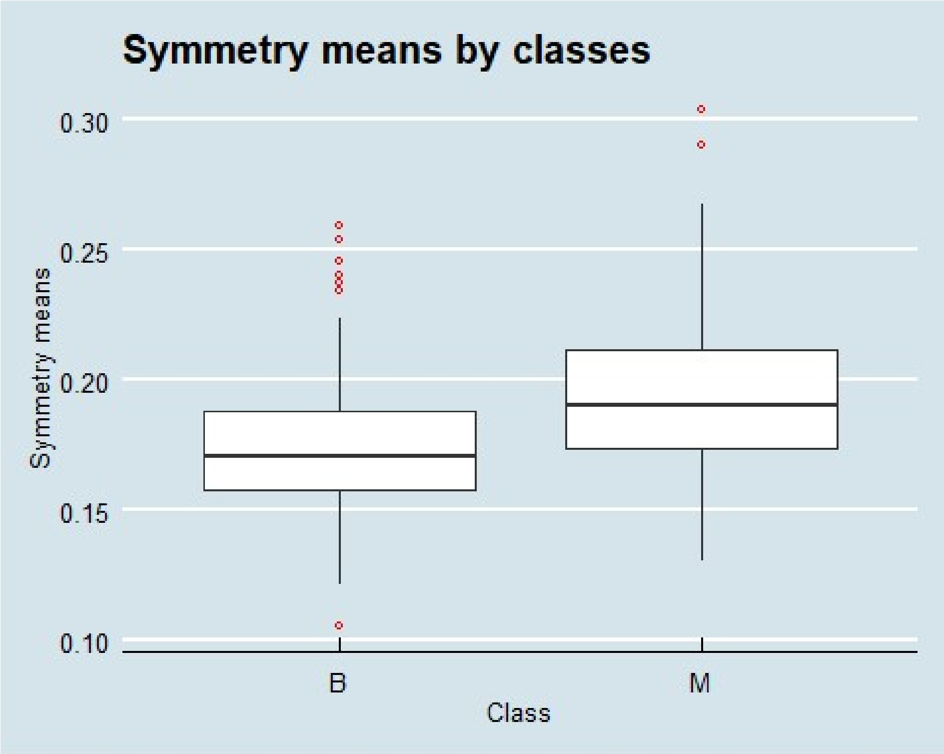
Concave points by classes chart:

```
train %>%
  ggplot(aes(x=diagnosis, y=train[,10])) +
  geom_boxplot(outlier.colour="red",
               outlier.shape=1,
               outlier.size=1) +
  xlab('Class') +
  ylab('Concave points') +
  ggtitle('Concave points by classes') +
  theme_economist()
```



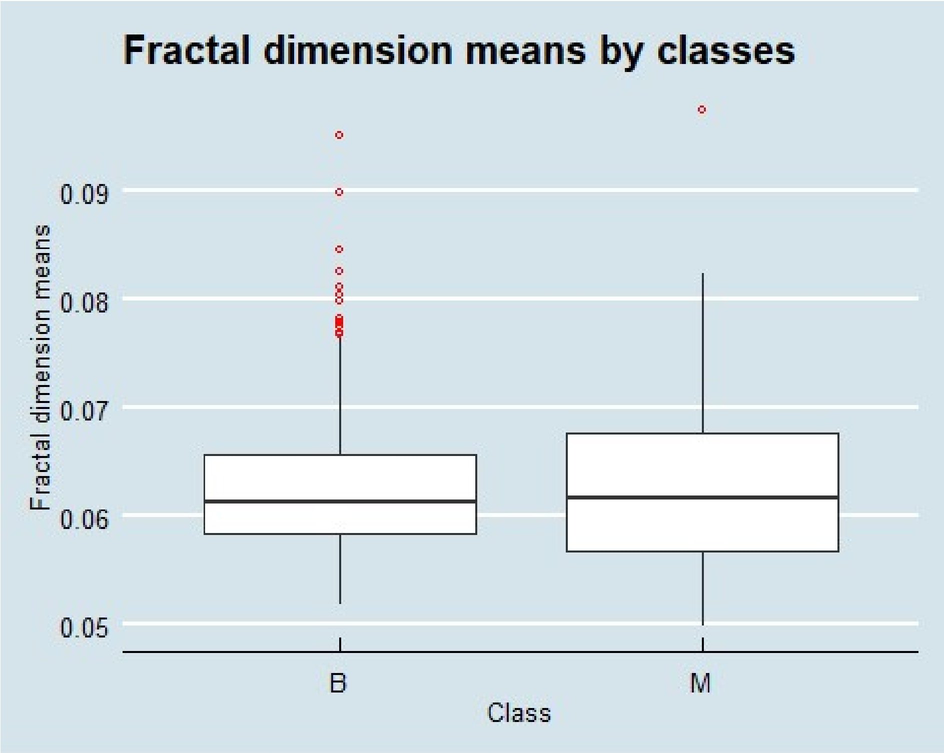
Symmetry means by classes chart:

```
train %>%
  ggplot(aes(x=diagnosis, y=train[,11])) +
  geom_boxplot(outlier.colour="red",
               outlier.shape=1,
               outlier.size=1) +
  xlab('Class') +
  ylab('Symmetry means') +
  ggtitle('Symmetry means by classes') +
  theme_economist()
```



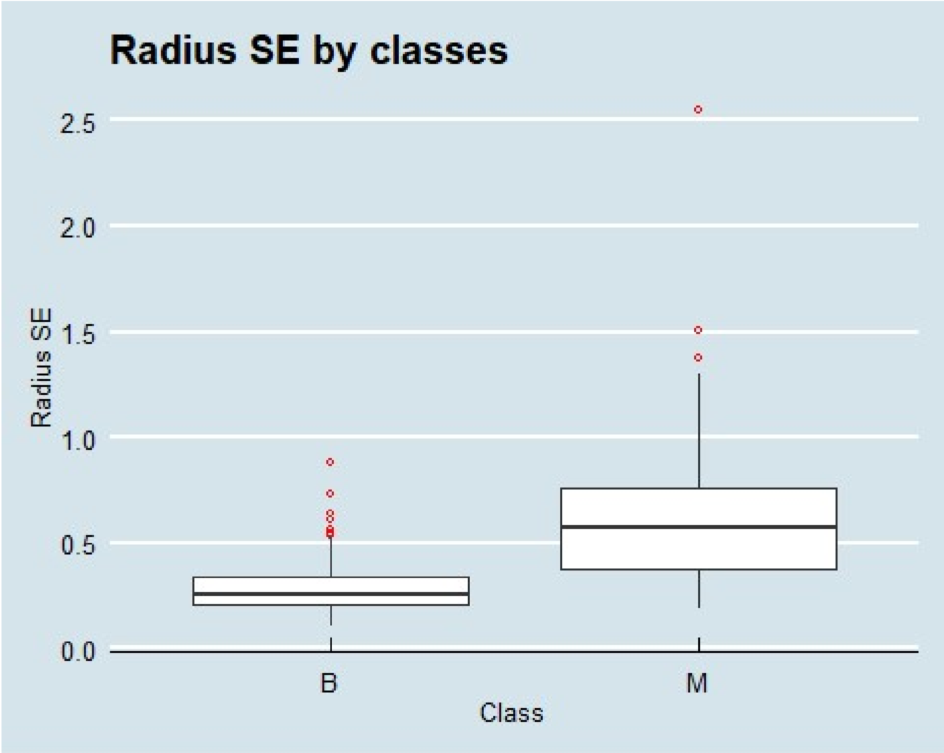
Fractal dimension means by classes chart:

```
train %>%
  ggplot(aes(x=diagnosis, y=train[,12])) +
  geom_boxplot(outlier.colour="red",
             outlier.shape=1,
             outlier.size=1) +
  xlab('Class') +
  ylab('Fractal dimension means') +
  ggtitle('Fractal dimension means by classes') +
  theme_economist()
```



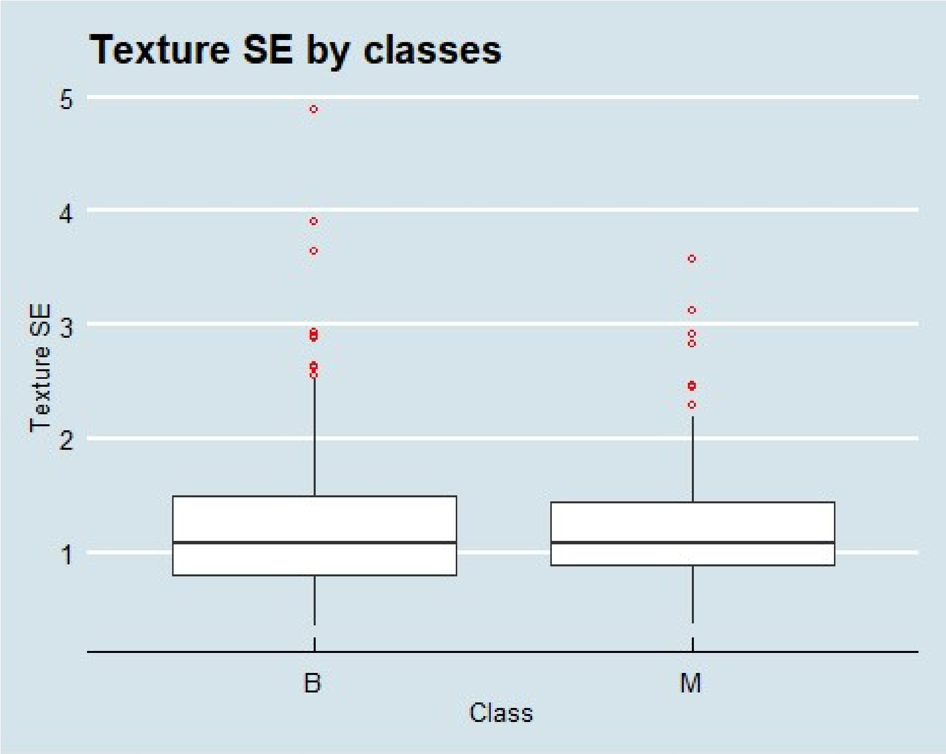
Radius SE by classes chart:

```
train %>%
  ggplot(aes(x=diagnosis, y=train[,13])) +
  geom_boxplot(outlier.colour="red",
               outlier.shape=1,
               outlier.size=1) +
  xlab('Class') +
  ylab('Radius SE') +
  ggtitle('Radius SE by classes') +
  theme_economist()
```



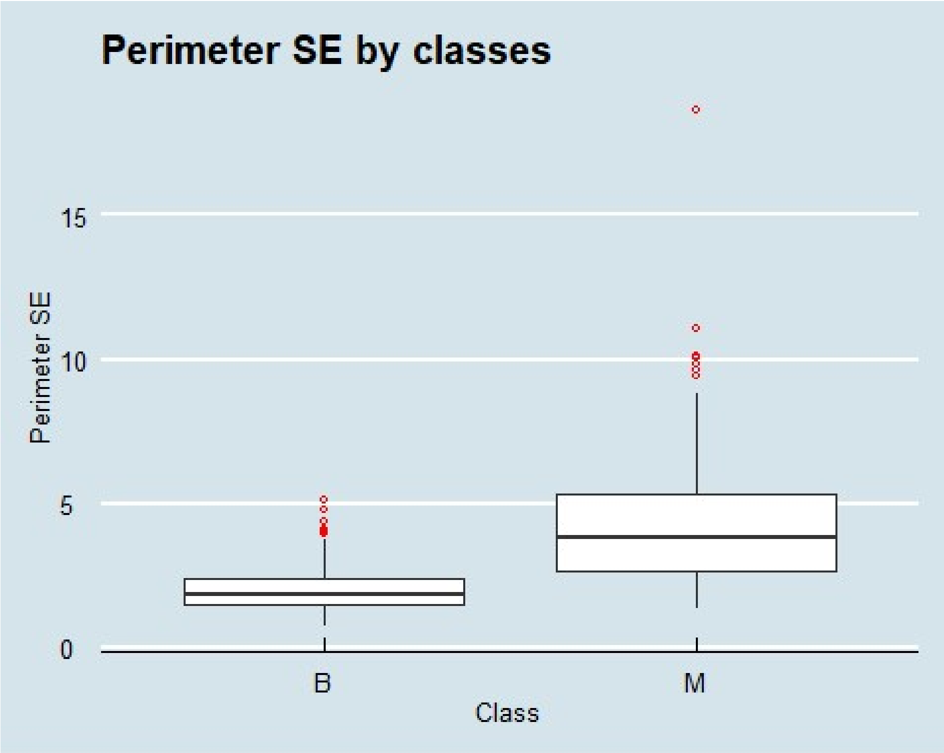
Texture SE by classes chart:

```
train %>%
  ggplot(aes(x=diagnosis, y=train[,14])) +
  geom_boxplot(outlier.colour="red",
             outlier.shape=1,
             outlier.size=1) +
  xlab('Class') +
  ylab('Texture SE') +
  ggtitle('Texture SE by classes') +
  theme_economist()
```



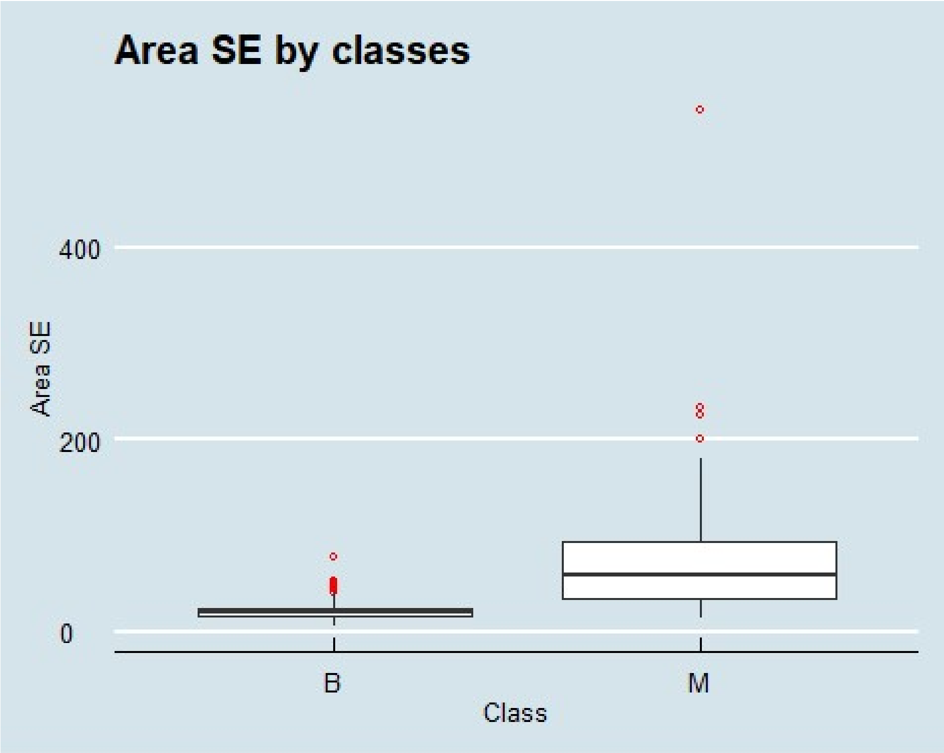
Perimeter SE by classes chart:

```
train %>%
  ggplot(aes(x=diagnosis, y=train[,15])) +
  geom_boxplot(outlier.colour="red",
               outlier.shape=1,
               outlier.size=1) +
  xlab('Class') +
  ylab('Perimeter SE') +
  ggtitle('Perimeter SE by classes') +
  theme_economist()
```



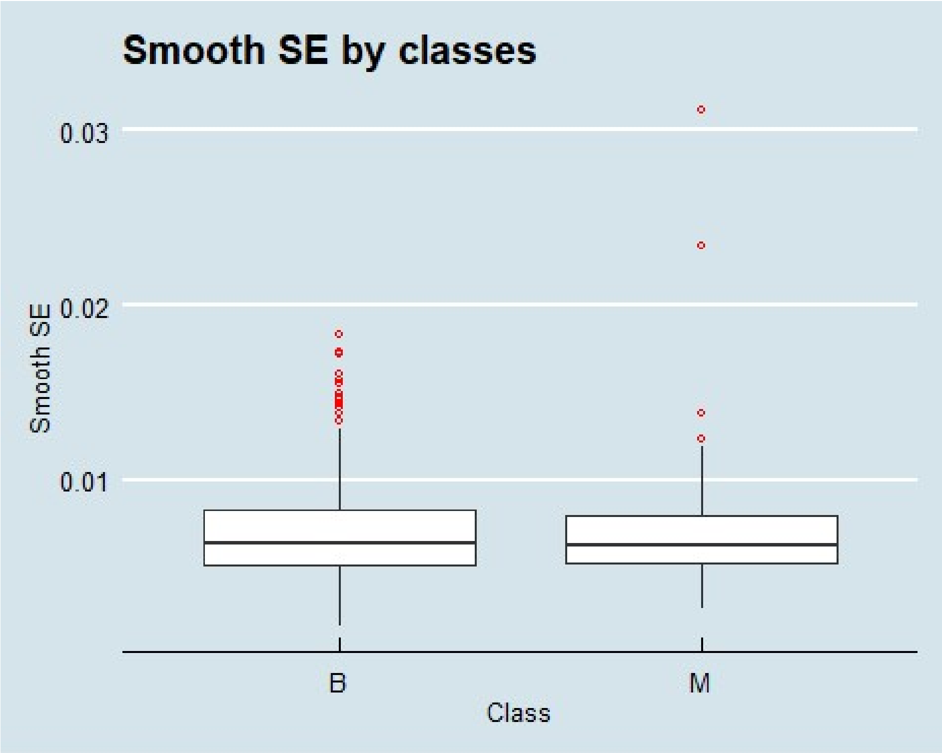
Area SE by classes chart:

```
train %>%
  ggplot(aes(x=diagnosis, y=train[,16])) +
  geom_boxplot(outlier.colour="red",
               outlier.shape=1,
               outlier.size=1) +
  xlab('Class') +
  ylab('Area SE') +
  ggtitle('Area SE by classes') +
  theme_economist()
```



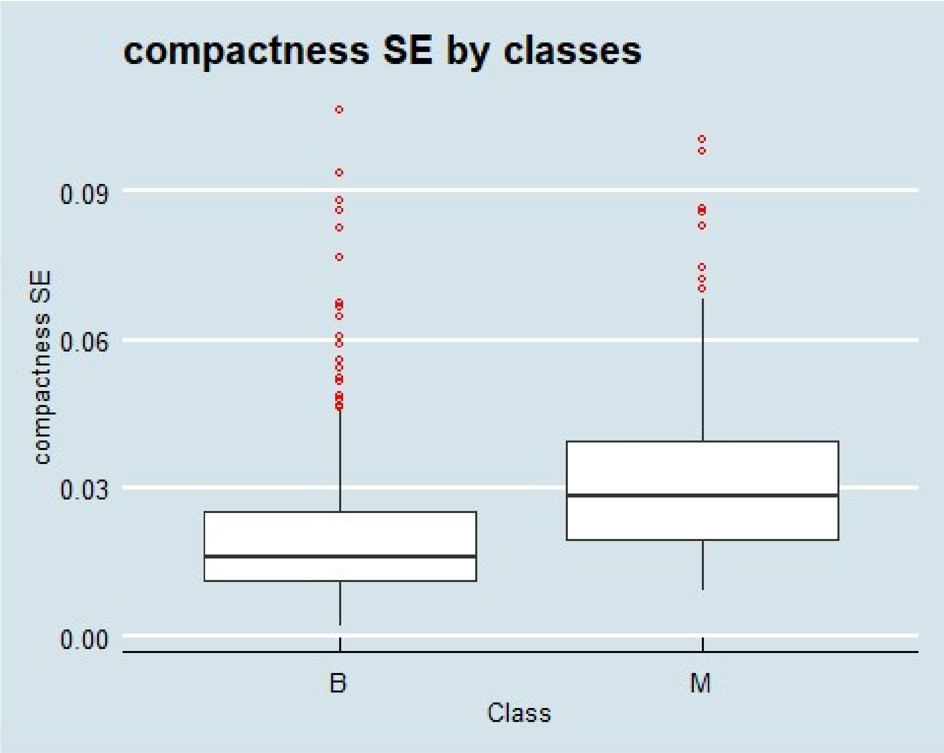
Smooth SE by classes chart:

```
train %>%
  ggplot(aes(x=diagnosis, y=train[,17])) +
  geom_boxplot(outlier.colour="red",
             outlier.shape=1,
             outlier.size=1) +
  xlab('Class') +
  ylab('Smooth SE') +
  ggtitle('Smooth SE by classes') +
  theme_economist()
```



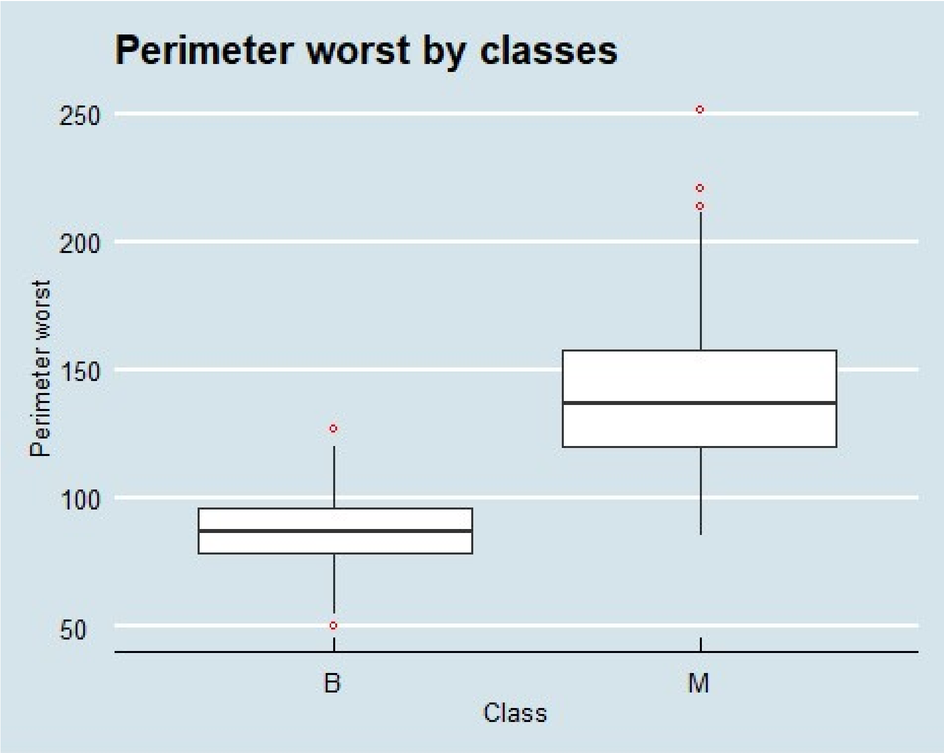
Compactness SE by classes chart:

```
train %>%
  ggplot(aes(x=diagnosis, y=train[,18])) +
  geom_boxplot(outlier.colour="red",
               outlier.shape=1,
               outlier.size=1) +
  xlab('Class') +
  ylab('compactness SE') +
  ggtitle('compactness SE by classes') +
  theme_economist()
```



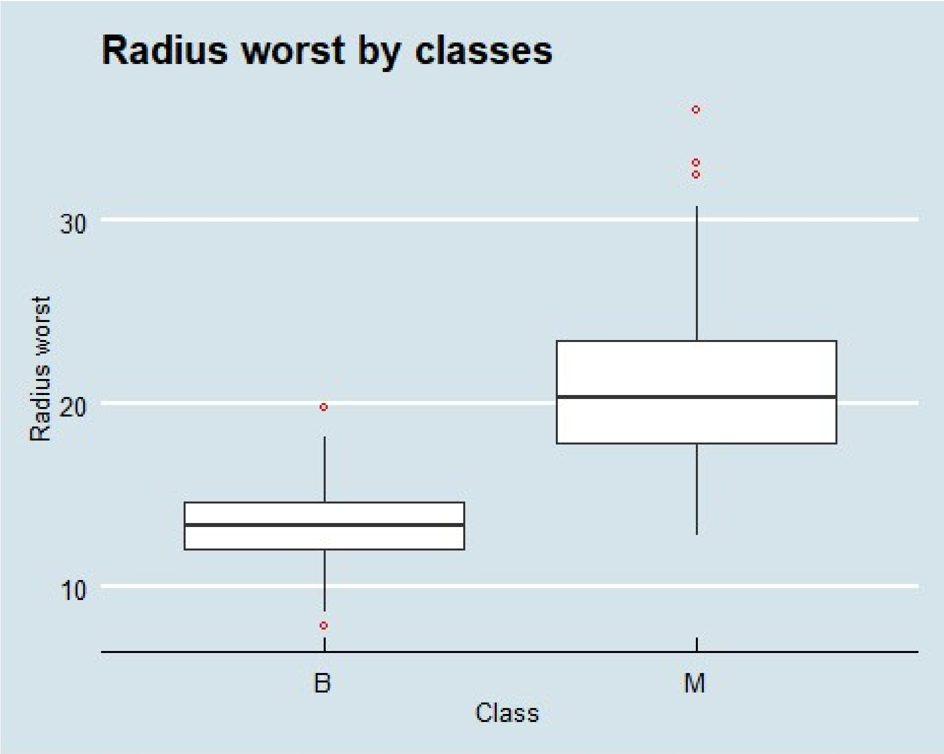
Perimeter worst by classes chart:

```
train %>%
  ggplot(aes(x=diagnosis, y=train[,25])) +
  geom_boxplot(outlier.colour="red",
               outlier.shape=1,
               outlier.size=1) +
  xlab('Class') +
  ylab('Perimeter worst') +
  ggtitle('Perimeter worst by classes') +
  theme_economist()
```



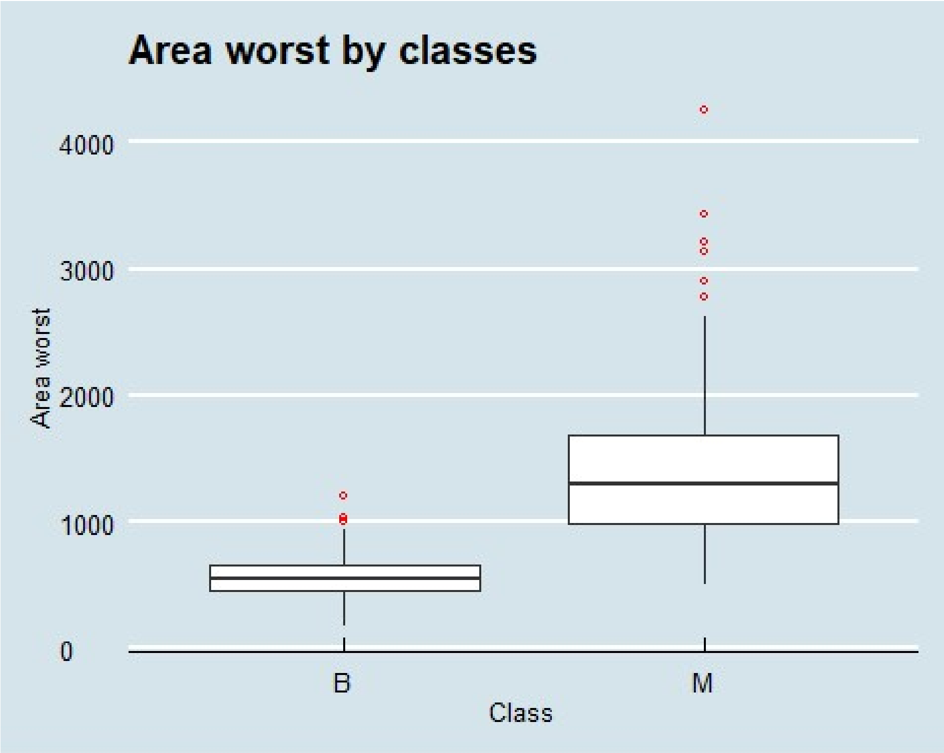
Radius worst by classes chart:

```
train %>%
  ggplot(aes(x=diagnosis, y=train[,23])) +
  geom_boxplot(outlier.colour="red",
             outlier.shape=1,
             outlier.size=1) +
  xlab('Class') +
  ylab('Radius worst') +
  ggtitle('Radius worst by classes') +
  theme_economist()
```



Area worst by classes chart:

```
train %>%
  ggplot(aes(x=diagnosis, y=train[,26])) +
  geom_boxplot(outlier.colour="red",
              outlier.shape=1,
              outlier.size=1) +
  xlab('Class') +
  ylab('Area worst') +
  ggtitle('Area worst by classes') +
  theme_economist()
```



Based on the graphs, we see that perimeter_worst and radius_worst have good presentations.

Correlation table for our predictors:

```
Cor_Tab <- cor(train[,3:32])

formatC(Cor_Tab, digits = 2, format = 'f', big.mark = ',') %>% knitr::kable()
```

From the table, we can see that some numbers seem to be highly correlated.

Defining arguments

Before we start, we have to define the parameters that are going to be passed into our functions (x and y for both train and test subsets).

```
# define train and test sets

Train_prediction <- as.matrix(train[,3:32]) #Train_Predictors
Train_class <- train$diagnosis #Train_Class
Test_prediction <- as.matrix(test[,3:32]) #Test_Predictors
Test_class <- test$diagnosis #Test_Class

CrlF <- trainControl(method="cv",         #Control the computational nuances of thetrainfunction
                     number = 15,    #Either the number of folds or number of resampling iterations
                     classProbs = TRUE,
                     summaryFunction = twoClassSummary)
```

Generalized Linear Model (GLM)

From a mathematical perspective, the basis of much of the evidence statistics found in the Model Linear (ML) general or classic. Its importance lies in its structure, we suppose, it reflects the explanatory elements of a phenomenon through relationships probabilistic functions between variables. The Generalized Linear Model (MLG), which we deal with in this work, is the natural extension of the Model Linear classic.

Alg 1 - Generalized Linear Model

```
set.seed(1, sample.kind="Rounding")
# if using R 3.5 or earlier, use `set.seed(1)` instead

train_glm <- train(x = Train_prediction, y = Train_class, method = "glm",
                   metric = "ROC",
                   preProcess = c("scale", "center"),  # in order to normalize the data
                   trControl= CrlF)

y_hat_glm <- predict(train_glm, Test_prediction)

m1acc <- confusionMatrix(y_hat_glm, Test_class)$overall['Accuracy']

cat('Model accuracy is equal to ',
    formatC(m1acc, digits = 5, format = 'f', big.mark = ','), '. ',
    'Below is the confusion matrix:', sep = "")

## Model accuracy is equal to 0.90909. Below is the confusion matrix:

confusionMatrix(y_hat_glm, Test_class)$table

##           Reference
## Prediction  B  M
##          B 84  7
##          M  6 46
```

Let’s move to other algorithms to see if we can improve accuracy.

k-Nearest Neighbors (KNN)

K-Nearest-Neighbor is an instance-based algorithm of supervised type of Machine Learning. It can vary to classify new samples (discrete values) or to predict (regression, continuous values). Being a simple method, it is ideal to enter the world of Machine Learning. It is basically used to classify values by searching for the “most similar” data points (by proximity) learned in the training stage and making guesses of new points based on that classification.

```
set.seed(1, sample.kind="Rounding")
# if using R 3.5 or earlier, use `set.seed(1)` instead

train_knn <- train(x = Train_prediction, y = Train_class, method="knn",
```

```
                  metric="ROC",
                  preProcess = c('center', 'scale'),
                  tuneLength=10, #The tuneLength parameter tells the algorithm to try different default v
alues for the main parameter
                  #In this case we used 10 default values
                  trControl=CrlF)

y_hat_knn <- predict(train_knn, Test_prediction)

m2acc <- confusionMatrix(y_hat_knn, Test_class)$overall['Accuracy']

cat('Model accuracy is equal to ',
    formatC(m2acc, digits = 5, format = 'f', big.mark = ','), '. ',
    'Below is the confusion matrix:', sep = "")

## Model accuracy is equal to 0.95804. Below is the confusion matrix:

confusionMatrix(y_hat_knn, Test_class)$table

##           Reference
## Prediction  B  M
##          B 90  6
##          M  0 47

# Shows optimal k from finalModel.
cat('Optimal k is ',
    formatC(train_knn$finalModel$k, digits = 0,
            format = 'f', big.mark = ','),
    sep = "")

## Optimal k is 13
```

Random Forest (RF)

Random Forest Regression is a supervised classification algorithm. As its name suggests, this algorithm creates the forest with multiple trees. In general, the more trees there are in the forest, the more robust the forest will be. Similarly, in the random forest classifier, the greater the number of trees in the forest, the greater the precision.

Once each decision tree has been calculated, the results of each of them are averaged and with this the prediction of the problem is obtained. The advantages of this algorithm are the following:

- It can solve both types of problems, that is, classification and regression, and it makes a decent estimate on both fronts.
- One of the most striking benefits is the power to handle large amounts of data with higher dimensionality.
- It can handle thousands of input variables and identify the most significant variables, making it considered one of the dimensionality reduction methods.
- Furthermore, the model shows the importance of the variable, which can be a very useful feature.

- It has an effective method of estimating missing data and maintains accuracy when a large proportion of the data is missing.

```
set.seed(1, sample.kind="Rounding")
# if using R 3.5 or earlier, use `set.seed(1)` instead

# Setting tuning parameters for the train function
grid <- expand.grid(mtry=seq(1, 5, 1),
                    splitrule = c("extratrees", "gini"),
                    min.node.size = seq(1, 50, 10))

train_rf <- train(x = Train_prediction, y = Train_class, method="nnet",
                  metric="ROC",
                  preProcess=c('center', 'scale', 'pca'),
                  tuneLength=10,
                  trace=FALSE,
                  trControl=CrlF)

y_hat_rf <- predict(train_rf, Test_prediction)

m3acc <- confusionMatrix(y_hat_rf, Test_class)$overall['Accuracy']

cat('Model accuracy is equal to ',
    formatC(m3acc, digits = 5, format = 'f', big.mark = ','), '. ',
    'Below is the confusion matrix:', sep = "")

## Model accuracy is equal to 0.95804. Below is the confusion matrix:

confusionMatrix(y_hat_rf, Test_class)$table

##           Reference
## Prediction  B  M
##          B 87  3
##          M  3 50
```

Naive Bayes (NB)

Naive Bayes models are a special class of Automatic Learning, or Machine Learning, classification algorithms, as we will refer from now on. They are based on a statistical classification technique called “Bayes’ theorem”. These models are called “Naive” algorithms, or “Innocents” in Spanish. They assume that the predictor variables are independent of each other. In other words, that the presence of a certain characteristic in a data set is not related at all to the presence of any other characteristic. They provide an easy way to build models with very good behavior due to their simplicity.

```
set.seed(1, sample.kind="Rounding")
# if using R 3.5 or earlier, use `set.seed(1)` instead

# Setting train control argument for the train function
control <- trainControl(method = "cv", number = 10, p = .9)
```

```
train_nb <- train(x = Train_prediction, y = Train_class, method="nb",
                  metric="ROC",
                  preProcess=c('center', 'scale'), #in order to normalize de data
                  trace=FALSE,
                  trControl=CrlF)

y_hat_nb <- predict(train_nb, Test_prediction)

m4acc <- confusionMatrix(y_hat_nb, Test_class)$overall['Accuracy']

cat('Model accuracy is equal to ',
    formatC(m4acc, digits = 5, format = 'f', big.mark = ','), '. ',
    'Below is the confusion matrix:', sep = "")

## Model accuracy is equal to 0.92308. Below is the confusion matrix:

confusionMatrix(y_hat_nb, Test_class)$table

##           Reference
## Prediction  B  M
##          B 84  5
##          M  6 48
```

Support Vector Machines with Polynomial Kernel (SVM)

They are also known by the acronym SVM for its acronym in English (Support Vector Machines). They can be used for both regression and classification.

Conceptually, SVMs are easier to explain for classification problems.

The techniques of Vector Support Machine (SVM) for some strange reason is one of the algorithms that arouse the most interest and above all the one that I think generates the most over estimation. It is possible that I have not used enough the family of algorithms to recognize so much their value.

Not more than a year ago, a friend in a proposal suggested using this family of algorithms without checking whether it was really convenient to use them for the type of data that would be analyzed. This friend's justification was that SVM were the "best" algorithms, which is false, you cannot rate a family of algorithms as the best, you need to test several against the data to determine which ones work best with them.

```
set.seed(1, sample.kind="Rounding")
# if using R 3.5 or earlier, use `set.seed(1)` instead

train_svm <- train(x = Train_prediction, y = Train_class, method = "svmPoly")

y_hat_svm <- predict(train_svm, Test_prediction)

m5acc <- confusionMatrix(y_hat_svm, Test_class)$overall['Accuracy']

cat('Model accuracy is equal to ',
```

```
  formatC(m5acc, digits = 5, format = 'f', big.mark = ','), '. ',
  'Below is the confusion matrix:', sep = "")
```

```
## Model accuracy is equal to 0.95804. Below is the confusion matrix:
```

```
confusionMatrix(y_hat_svm, Test_class)$table
```

```
##           Reference
## Prediction  B  M
##          B 89  5
##          M  1 48
```

Results

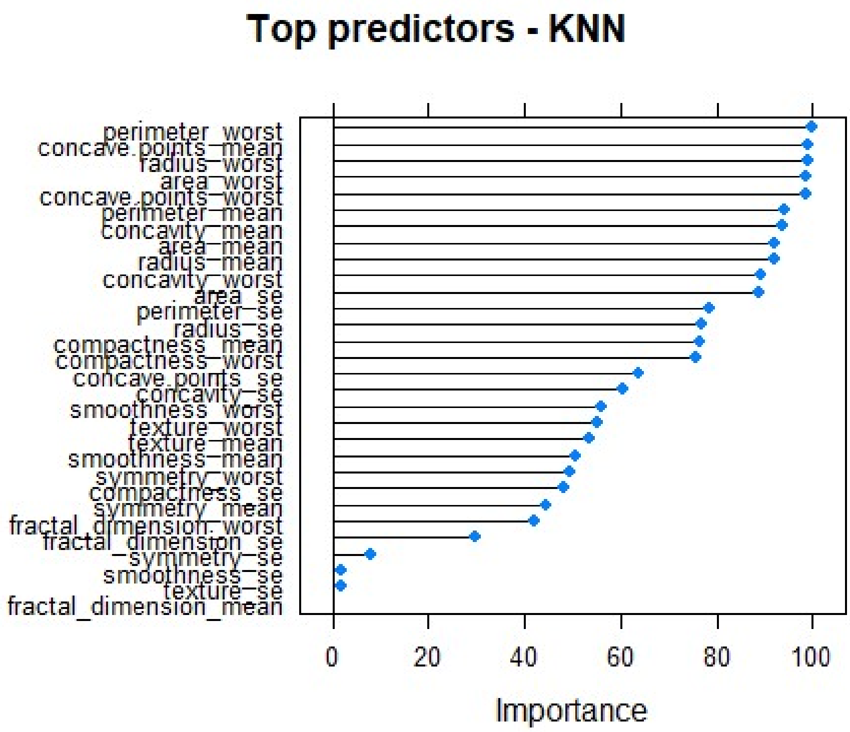
Below we show the results

```
models <- c('1. Generalized Linear Model',
            '2. k-Nearest Neighbors',
            '3. Random Forest',
            '4. Naive Bayes',
            '5. Support Vector Machines with Polynomial Kernel')

acc_values <- formatC(c(m1acc, m2acc, m3acc, m4acc, m5acc), digits = 5, big.mark=",")

acc_table  <- data.frame(Accuracy = acc_values, row.names = models)

acc_table %>% knitr::kable()
```

	Accuracy
1. Generalized Linear Model	0.90909
2. k-Nearest Neighbors	0.95804
3. Random Forest	0.95804
4. Naive Bayes	0.92308
5. Support Vector Machines with Polynomial Kernel	0.95804

Now we see which variables are better correlated with the output.

```
plot(varImp(train_knn), top=30, main="Top predictors - KNN")
```



Perimeter_worst is the variable with the highest correlation at the output

Conclusion

We analyzed the characteristics of cancer from a set of patient data available on github. We review the key predictors and then start building ML models.

The methods used for this analysis were:

1. Generalized linear model
2. Nearest K-Neighbors
3. Random Forest
4. Naive Bayes
5. Support of vector machines with polynomial nucleus

Models were tested in a test set. All models have good, relatively high precision.

We may consider increasing the size of the dataset to improve our results. And use other methods or combination of each other to improve our models.

Sources

Referenced websites:

1. https://raw.githubusercontent.com/gmineo/Breast-Cancer-Prediction-Project/master/

2. https://www.sciencedirect.com/topics/nursing-and-health-professions/pelvic-incidence

3. https://www.orthobullets.com/spine/2038/adult-isthmic-spondylolisthesis

4. https://en.wikipedia.org/wiki/Pelvic_tilt

5. https://datasetsearch.research.google.com/search?query=cancer%20dataset&docid=lqkM7t0bmGplzzTuAAAAAA%3D%3D

6. https://pdfs.semanticscholar.org/306d/2c889f7783e1b2944c9c684fc7342c77d206.pdf

7. https://datasetsearch.research.google.com/search?query=cancer%20dataset&docid=Fj%2BIDVyi5Wdm3sS7AAAAAA%3D%3D

Other sources used:

1. Introduction to Data Science - Data Analysis and Prediction Algorithms with R by Rafael A. Irizarry (https://rafalab.github.io/dsbook/)

2. Caret library (https://topepo.github.io/caret/)

3. towards data science (https://towardsdatascience.com)