HarvardX: PH125.9x Data Science Capstone Project / Choose Your Own

(Breast Cancer Prediction)

Gokhan USTA

07 January, 2021

1. Introduction

The purpose of breast tumor biopsy is to differentiate breast cancer from benign masses. Most breast tumors are not cancerous, but it is critical to correctly identify malignant tumors. Because advanced breast cancer is life-threatening, a false negative test result that misses a diagnosis could be a fatal error. It is also important to reduce false positives and correctly identify patients who have benign masses, both to reduce unwarranted psychological and financial distress associated with a cancer diagnosis and to avoid morbidity from unnecessary cancer treatment.

Many breast tumors are assessed by fine needle aspiration (FNA) cytology, the most cost-effective and least invasive biopsy method (Mitra and Dey, 2016). FNA consists of using a thin needle to extract a piece of the tumor, which a pathologist analyzes for physical features that distinguish malignant cells. However, FNA biopsy accuracy changes dramatically with experience of the cytopathologist (Feoli et al., 2008). The false negative rate is variable across studies, with many recent studies showing a 5-10% false negative rate and some centers reporting rates of 15% or greater (Mitra and Dey, 2016). There has been a movement away from use of FNA towards the more invasive and expensive core needle biopsy method, which is widely reputed to be better, although studies show that FNA and core needle biopsy have similar false negative rates of 1.7% when performed by an experienced cytopathologist (Brancato et al., 2012).

Digital analysis of breast biopsies could potentially reduce the disparity in FNA false negative rates. Cytology relies on visual pattern matching of cellular features by the pathologist, and these pattern matching strategies can be mimicked through machine learning. A classifier could supplement the opinion of a cytopathologist, potentially compensating for lack of pathologist experience and providing an accessible and objective second opinion about ambiguous samples. The model could also serve as a teaching tool and highlight which features are most useful for visually distinguishing cancer and normal tissue.

2. Overview

This project is related to the Choose-your-own project of the HarvardX: PH125.9x Data Science: Capstone Project.

In the project, the given dataset will be prepared and setup. An exploratory data analysis is carried out in order to develop a machine learning algorithm Results will be explained. Finally, the report will end with some concluding remarks.

The objective of this report is to train machine learning models to predict whether a breast cancer cell is Benign or Malignant. Data will be transformed and its dimension reduced to reveal patterns in the dataset and create a more robust analysis. As previously said, the optimal model will be selected following the resulting accuracy, sensitivity, and f1 score, amongst other factors. We will later define these metrics. We can use machine learning method to extract the features of cancer cell nuclei image and classify them. It would be helpful to determine whether a given sample appears to be Benign (“B”) or Malignant (“M”). The machine learning models that we will applicate in this report try to create a classifier that provides a high accuracy level combined with a low rate of false-negatives (high sensitivity).

3. Dataset

The .csv format file containing the data is loaded from my personal github account. ("https://raw.githubusercontent.com/gustalab/Data-Science-Capstone-CYO-Project/main/data.csv")

The Breast Cancer Wisconsin Diagnostic Dataset consists of 30 features computationally extracted from digital images of FNA biopsy slides of a consecutive series of 569 breast tumors (Street et al., 1993). Features describe properties of the cell nuclei, including attributes related to nucleus size, shape and regularity. For each feature, the average value of the feature across all cells in the image, standard error of the feature across all nuclei (se), and most extreme (worst) value of the feature in the image are reported.

The dataset's features describe characteristics of the cell nuclei on the image. The features information are specified below:

- **radius:** Nucleus radius (mean of distances from center to points on the perimeter).
- **texture:** Nucleus texture (standard deviation of gray-scale values).
- **perimeter:** Nucleus perimeter.
- **area:** Nucleus area.
- **smoothness:** Nucleus smoothness (local variation in radius lengths).
- **compactness:** Nucleus compactness (perimeterˆ2/area - 1).
- **concavity:** Nucleus concavity (severity of concave portions of the contour).
- **concave_pts:** Number of concave portions of the nucleus contour.
- **symmetry:** Nucleus symmetry.
- **fractal_dim:** Nucleus fractal dimension ("coastline approximation" - 1).

Attribute Information:

1. ID number
2. Diagnosis (M = malignant, B = benign)

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. From this diagnosis, 357 of the cases were classified as benign tumors and 212 were considered malignant tumors. All cancers and some of the benign masses were histologically confirmed.

4. Methods, Analysis and Visualizations

In order to figure out the dataset, you can find the first rows of dataset below. There are 569 observations and 32 variables, including “id” and “diagnosis”.

head(data)

```
##         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(data)

```
##                  radius_mean texture_mean perimeter_mean area_mean
## radius_mean             1.00         0.32           1.00      0.99
## texture_mean            0.32         1.00           0.33      0.32
## perimeter_mean          1.00         0.33           1.00      0.99
## area_mean               0.99         0.32           0.99      1.00
## smoothness_mean         0.17        -0.02           0.21      0.18
## compactness_mean        0.51         0.24           0.56      0.50
##                  smoothness_mean compactness_mean concavity_mean
## radius_mean                 0.17            0.51           0.68
## texture_mean               -0.02            0.24           0.30
## perimeter_mean              0.21            0.56           0.72
## area_mean                   0.18            0.50           0.69
## smoothness_mean             1.00            0.66           0.52
## compactness_mean            0.66            1.00           0.88
##                  concave.points_mean symmetry_mean fractal_dimension_mean
## radius_mean                     0.82          0.15                  -0.31
## texture_mean                    0.29          0.07                  -0.08
## perimeter_mean                  0.85          0.18                  -0.26
## area_mean                       0.82          0.15                  -0.28
## smoothness_mean                 0.55          0.56                   0.58
## compactness_mean                0.83          0.60                   0.57
##                  radius_se texture_se perimeter_se area_se smoothness_se
## radius_mean           0.68      -0.10         0.67    0.74         -0.22
## texture_mean          0.28       0.39         0.28    0.26          0.01
## perimeter_mean        0.69      -0.09         0.69    0.74         -0.20
## area_mean             0.73      -0.07         0.73    0.80         -0.17
## smoothness_mean       0.30       0.07         0.30    0.25          0.33
## compactness_mean      0.50       0.05         0.55    0.46          0.14
##                  compactness_se concavity_se concave.points_se symmetry_se
## radius_mean              0.21         0.19              0.38       -0.10
## texture_mean             0.19         0.14              0.16        0.01
## perimeter_mean           0.25         0.23              0.41       -0.08
## area_mean                0.21         0.21              0.37       -0.07
## smoothness_mean          0.32         0.25              0.38        0.20
## compactness_mean         0.74         0.57              0.64        0.23
##                  fractal_dimension_se radius_worst texture_worst
## radius_mean                     -0.04         0.97          0.30
## texture_mean                     0.05         0.35          0.91
## perimeter_mean                  -0.01         0.97          0.30
## area_mean                       -0.02         0.96          0.29
## smoothness_mean                  0.28         0.21          0.04
## compactness_mean                 0.51         0.54          0.25
##                  perimeter_worst area_worst smoothness_worst compactness_worst
## radius_mean                 0.97       0.94             0.12              0.41
## texture_mean                0.36       0.34             0.08              0.28
## perimeter_mean              0.97       0.94             0.15              0.46
## area_mean                   0.96       0.96             0.12              0.39
## smoothness_mean             0.24       0.21             0.81              0.47
## compactness_mean            0.59       0.51             0.57              0.87
```

```
##                  concavity_worst concave.points_worst symmetry_worst
## radius_mean                 0.53                 0.74           0.16
## texture_mean                0.30                 0.30           0.11
## perimeter_mean              0.56                 0.77           0.19
## area_mean                   0.51                 0.72           0.14
## smoothness_mean             0.43                 0.50           0.39
## compactness_mean            0.82                 0.82           0.51
##                  fractal_dimension_worst
## radius_mean                         0.01
## texture_mean                        0.12
## perimeter_mean                      0.05
## area_mean                           0.00
## smoothness_mean                     0.50
## compactness_mean                    0.69
```

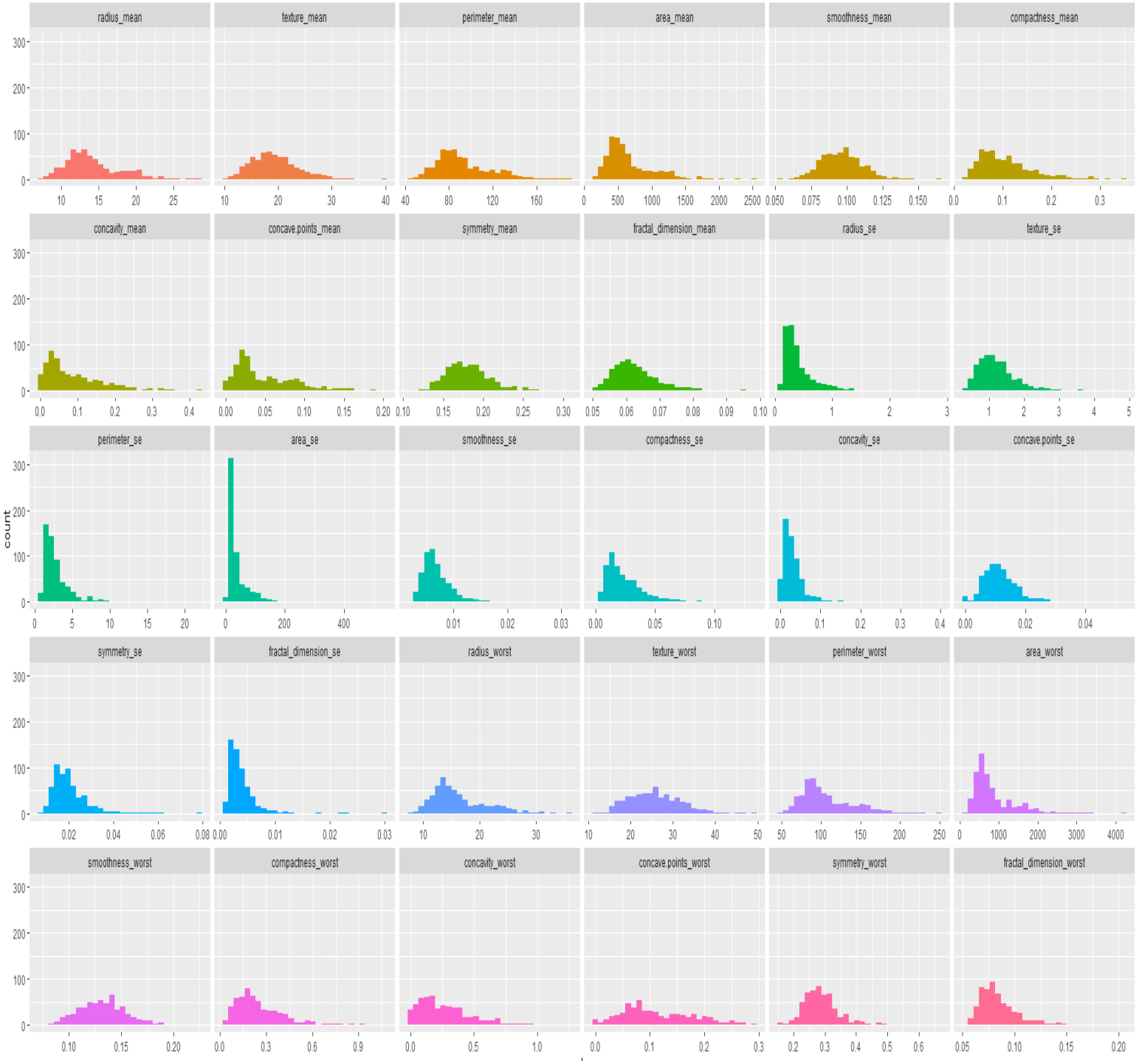
The proportions of the diagnosis are as follows; (it looks they are not balanced)

mean(data$diagnosis == "B")
mean(data$diagnosis == "M")
prop.table(table(data$diagnosis))

```
##         B         M
## 0.6274165 0.3725835
```

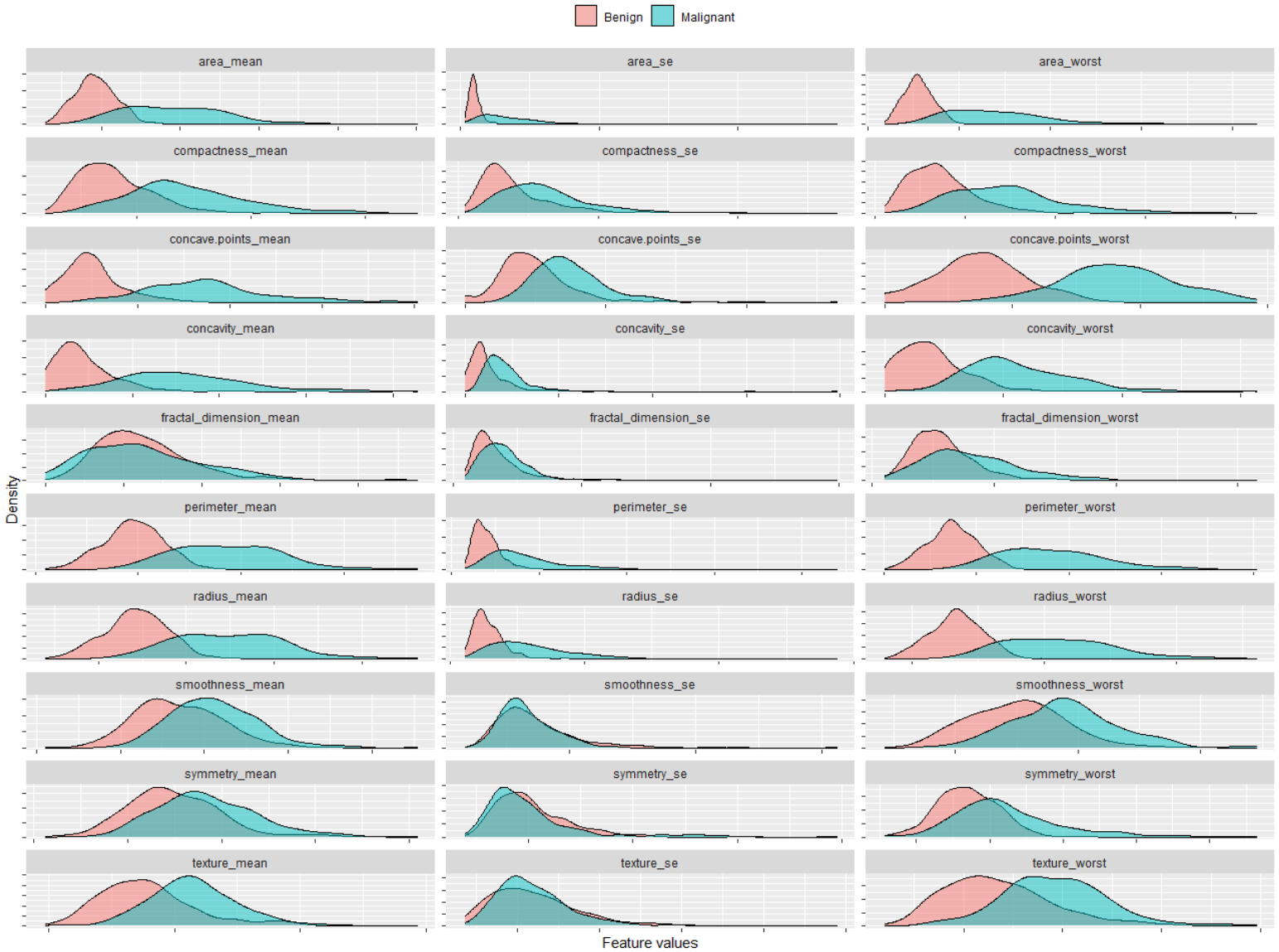
The distributions of the variables in the dataset are shown below, (It looks they are normally distributed).

```
# Plotting Numerical Data
plot_num(data %>% select(-id), bins=30)
```



Most of the 30 nuclear features show a significant difference in distribution between benign and malignant samples

```
# Plot and facet wrap density plots for each feature by diagnosis
data %>% select(-id) %>%
  gather("feature", "value", -diagnosis) %>%
  ggplot(aes(value, fill = diagnosis)) +
  geom_density(alpha = 0.5) +
  xlab("Feature values") +
  ylab("Density") +
  theme(legend.position = "top",
        axis.text.x = element_blank(), axis.text.y = element_blank(),
        legend.title=element_blank()) +
  scale_fill_discrete(labels = c("Benign", "Malignant")) +
  facet_wrap(~ feature, scales = "free", ncol = 3)
```



The correlation matrix of the variables (the features) demonstrates that many feature combinations encode distinct information, as shown by the fact that there are several independent clusters of features along the primary diagonal that are relatively uncorrelated with each other. There are also several features that are highly collinear, shown by dark blue squares along the main diagonal.

```
# Correlation plot
correlationMatrix <- cor(data[,3:ncol(data)])
head(round(correlationMatrix,2))
```

```
##                  radius_mean texture_mean perimeter_mean area_mean
## radius_mean             1.00         0.32           1.00      0.99
## texture_mean            0.32         1.00           0.33      0.32
## perimeter_mean          1.00         0.33           1.00      0.99
## area_mean               0.99         0.32           0.99      1.00
## smoothness_mean         0.17        -0.02           0.21      0.18
## compactness_mean        0.51         0.24           0.56      0.50
##                  smoothness_mean compactness_mean concavity_mean
## radius_mean                 0.17             0.51           0.68
## texture_mean               -0.02             0.24           0.30
## perimeter_mean              0.21             0.56           0.72
## area_mean                   0.18             0.50           0.69
## smoothness_mean             1.00             0.66           0.52
## compactness_mean            0.66             1.00           0.88
##                  concave.points_mean symmetry_mean fractal_dimension_mean
## radius_mean                     0.82          0.15                  -0.31
## texture_mean                    0.29          0.07                  -0.08
## perimeter_mean                  0.85          0.18                  -0.26
## area_mean                       0.82          0.15                  -0.28
## smoothness_mean                 0.55          0.56                   0.58
## compactness_mean                0.83          0.60                   0.57
##                  radius_se texture_se perimeter_se area_se smoothness_se
## radius_mean           0.68      -0.10         0.67    0.74         -0.22
## texture_mean          0.28       0.39         0.28    0.26          0.01
## perimeter_mean        0.69      -0.09         0.69    0.74         -0.20
## area_mean             0.73      -0.07         0.73    0.80         -0.17
## smoothness_mean       0.30       0.07         0.30    0.25          0.33
## compactness_mean      0.50       0.05         0.55    0.46          0.14
##                  compactness_se concavity_se concave.points_se symmetry_se
## radius_mean                0.21         0.19              0.38       -0.10
## texture_mean               0.19         0.14              0.16        0.01
## perimeter_mean             0.25         0.23              0.41       -0.08
## area_mean                  0.21         0.21              0.37       -0.07
## smoothness_mean            0.32         0.25              0.38        0.20
## compactness_mean           0.74         0.57              0.64        0.23
##                  fractal_dimension_se radius_worst texture_worst
## radius_mean                     -0.04         0.97          0.30
## texture_mean                     0.05         0.35          0.91
## perimeter_mean                  -0.01         0.97          0.30
## area_mean                       -0.02         0.96          0.29
## smoothness_mean                  0.28         0.21          0.04
## compactness_mean                 0.51         0.54          0.25
```

```
##                   perimeter_worst area_worst smoothness_worst compactness_worst
## radius_mean                  0.97       0.94             0.12              0.41
## texture_mean                 0.36       0.34             0.08              0.28
## perimeter_mean               0.97       0.94             0.15              0.46
## area_mean                    0.96       0.96             0.12              0.39
## smoothness_mean              0.24       0.21             0.81              0.47
## compactness_mean             0.59       0.51             0.57              0.87
##                   concavity_worst concave.points_worst symmetry_worst
## radius_mean                  0.53                 0.74           0.16
## texture_mean                 0.30                 0.30           0.11
## perimeter_mean               0.56                 0.77           0.19
## area_mean                    0.51                 0.72           0.14
## smoothness_mean              0.43                 0.50           0.39
## compactness_mean             0.82                 0.82           0.51
##                   fractal_dimension_worst
## radius_mean                          0.01
## texture_mean                         0.12
## perimeter_mean                       0.05
## area_mean                            0.00
## smoothness_mean                      0.50
## compactness_mean                     0.69
```

```
corrplot(correlationMatrix, tl.cex = 1, addrect = 8)
```

As shown by this plot, many variables are highly correlated with each other. Many methods perform better if highly correlated attributes are removed. Correlated attributes are usually removed because they are similar in behavior and will have similar impact in prediction calculations, so keeping attributes with similar impacts is redundant. Removing correlated attributes saves space and time of calculation of complex algorithms. Moreover, it also makes processes easier to design, analyze, understand and comprehend.

```
# Find variables that have high correlation between (>0.90)
cutoff <- 0.9
highcorrelation <- findCorrelation(correlationMatrix, cutoff=cutoff, names=TRUE)

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

```
##  [1] "concavity_mean"      "concave.points_mean" "perimeter_worst"
##  [4] "radius_worst"        "perimeter_mean"      "area_worst"
##  [7] "radius_mean"         "perimeter_se"        "area_se"
## [10] "texture_mean"
```

When we omit the correlated variables, we have 22 left.

```
# Remove correlated variables
corred_data <- data %>%select(-highcorrelation)
```

```
# Number of columns after removing correlated variables
ncol(corred_data)
```

```
## [1] 22
```

5. Modelling Approach

a. Modelling

Principal Component Analysis (PCA)

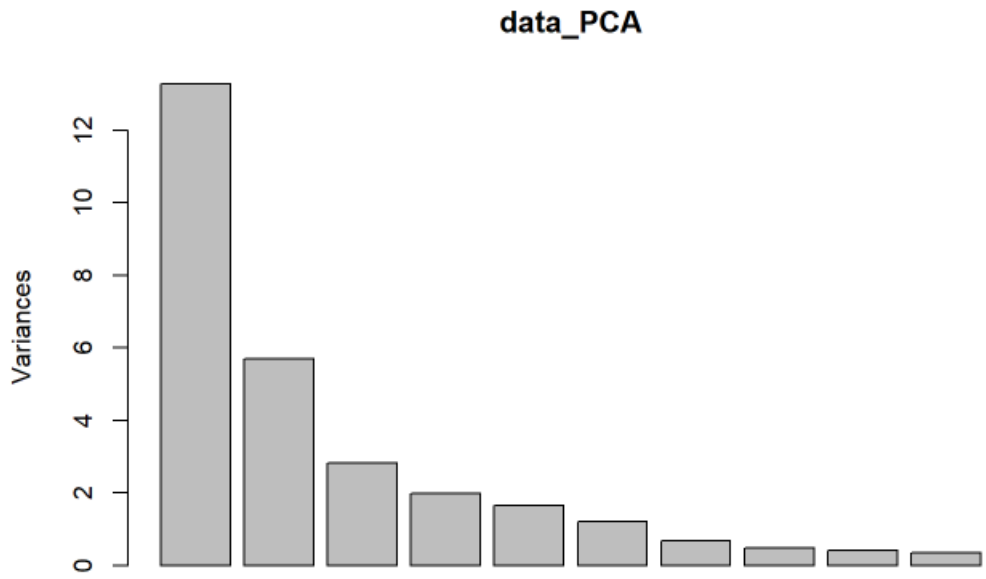
PCA is used in exploratory data analysis and for making predictive models. It is commonly used for dimensionality reduction by projecting each data point onto only the first few principal components to obtain lower-dimensional data while preserving as much of the data's variation as possible. The main idea of PCA is to reduce the dimensionality of a data set consisting of many variables correlated with each other, either heavily or lightly, while retaining the variation present in the dataset, up to the maximum extent. The same is done by transforming the variables to a new set of variables, which are known as the principal components (or simply, the PCs) and are orthogonal, ordered such that the retention of variation present in the original variables decrease as we move down in the order.

Principal component analysis is a highly effective method for reducing the number of dimensions without a proportional loss in the variance within the data and can help improve the computational efficiency of models with very large data sets.

To avoid redundancy and relevancy, we used the function 'prncomp' to calculate the Principal Component Analysis (PCA) and select the rights components to avoid correlated variables that can be detrimental to our clustering analysis.

Principal component analysis shows that the first principal component (PC) accounts for 44.27% of the variance, the second PC accounts for 18.97% of the variance, and every subsequent PC accounts for less than 10% of the variance.

```
# Principal Component Analysis (PCA)

data_PCA <- prcomp(data[,3:ncol(data)], center = TRUE, scale = TRUE)
plot(data_PCA, type="b")
```



```r
# Summary of data after PCA
summary(data_PCA)
```

```
## Importance of components:
##                           PC1    PC2     PC3     PC4     PC5     PC6     PC7
## Standard deviation     3.6444 2.3857 1.67867 1.40735 1.28403 1.09880 0.82172
## Proportion of Variance 0.4427 0.1897 0.09393 0.06602 0.05496 0.04025 0.02251
## Cumulative Proportion  0.4427 0.6324 0.72636 0.79239 0.84734 0.88759 0.91010
##                           PC8    PC9    PC10   PC11    PC12    PC13    PC14
## Standard deviation     0.69037 0.6457 0.59219 0.5421 0.51104 0.49128 0.39624
## Proportion of Variance 0.01589 0.0139 0.01169 0.0098 0.00871 0.00805 0.00523
## Cumulative Proportion  0.92598 0.9399 0.95157 0.9614 0.97007 0.97812 0.98335
##                           PC15    PC16    PC17    PC18    PC19    PC20   PC21
## Standard deviation     0.30681 0.28260 0.24372 0.22939 0.22244 0.17652 0.1731
## Proportion of Variance 0.00314 0.00266 0.00198 0.00175 0.00165 0.00104 0.0010
## Cumulative Proportion  0.98649 0.98915 0.99113 0.99288 0.99453 0.99557 0.9966
##                           PC22    PC23   PC24    PC25    PC26    PC27    PC28
## Standard deviation     0.16565 0.15602 0.1344 0.12442 0.09043 0.08307 0.03987
## Proportion of Variance 0.00091 0.00081 0.0006 0.00052 0.00027 0.00023 0.00005
## Cumulative Proportion  0.99749 0.99830 0.9989 0.99942 0.99969 0.99992 0.99997
##                           PC29    PC30
## Standard deviation     0.02736 0.01153
## Proportion of Variance 0.00002 0.00000
## Cumulative Proportion  1.00000 1.00000
```

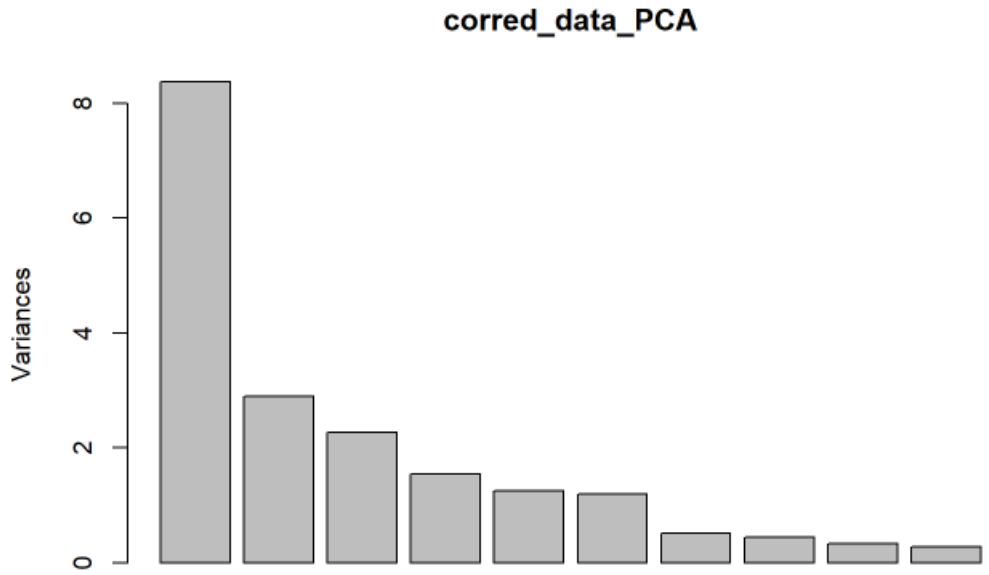
As we can observe from the table above, the two first components explain the 0.6324 of the variances. We need 10 principal components to explain more than 0.95 of the variances and 17 to explain more than 0.99.

```
# Reduce the number of variables
corred_data_PCA <- prcomp(corred_data[,3:ncol(corred_data)], center = TRUE, scale = TRUE)
corred_data_PCA
```

```
## Standard deviations (1, .., p=20):
##  [1] 2.8943927 1.7018557 1.5050877 1.2394807 1.1211555 1.0923213 0.7112027
##  [8] 0.6626042 0.5734860 0.5223779 0.4634184 0.4104866 0.4036281 0.3623720
## [15] 0.2741308 0.2374218 0.2116321 0.1832333 0.1504758 0.1333983
##
## Rotation (n x k) = (20 x 20):
##                                  PC1          PC2          PC3           PC4
## area_mean               -0.13805060  0.36290313 -0.368118361  0.1267158714
## smoothness_mean         -0.23146657 -0.02136021  0.210637905  0.3134909175
## compactness_mean        -0.32314531  0.08602783 -0.008823706  0.0007336756
## symmetry_mean           -0.22998053 -0.05343448  0.126407818  0.3133949361
## fractal_dimension_mean  -0.22238927 -0.29687987  0.288398171 -0.0791747176
## radius_se               -0.16791064  0.10401308 -0.408325959  0.2654218242
## texture_se              -0.04678534 -0.29058574 -0.263087450  0.3057647733
## smoothness_se           -0.09029121 -0.39664158 -0.053784958  0.2365692891
## compactness_se          -0.28058331 -0.17376100 -0.154741693 -0.2463507147
## concavity_se            -0.24607667 -0.15796919 -0.215328864 -0.2937198399
## concave.points_se       -0.24920036 -0.10510338 -0.302536686 -0.0976310171
## symmetry_se             -0.11450518 -0.32885390 -0.090852095  0.3183175098
## fractal_dimension_se    -0.22337109 -0.32637935 -0.083049142 -0.2784644460
## texture_worst           -0.09862460  0.18301194 -0.114444349  0.1689012379
## smoothness_worst        -0.21664015  0.08348862  0.326898567  0.2355268183
## compactness_worst       -0.29677594  0.18708145  0.108277286 -0.1469451758
## concavity_worst         -0.29455530  0.21064835 -0.018739459 -0.1653338944
## concave.points_worst    -0.27275126  0.31542628 -0.086026149  0.0119681950
## symmetry_worst          -0.20230292  0.12825618  0.274655453  0.2188717119
## fractal_dimension_worst -0.26618242 -0.01504947  0.300315954 -0.2119859140
##                                  PC5          PC6          PC7          PC8
## area_mean               -0.160719270  0.02923182 -0.153628710  0.17486716
## smoothness_mean         -0.251361612  0.28386935 -0.081471443 -0.30434475
## compactness_mean        -0.064721377  0.01678622 -0.194790583 -0.02063720
## symmetry_mean           -0.071086680 -0.34576791 -0.023073190 -0.40594703
## fractal_dimension_mean   0.002425126  0.08668080 -0.340537826 -0.07054488
## radius_se               -0.203732323  0.04855302 -0.397217160  0.19144458
## texture_se               0.482921007  0.13218180 -0.161593360 -0.28453422
## smoothness_se           -0.126739596  0.35010646  0.369458063  0.45523661
## compactness_se           0.046419204 -0.07549997  0.009746728  0.14453710
## concavity_se            -0.005985594 -0.06821041  0.351249172 -0.18679786
## concave.points_se       -0.166111315  0.03053200  0.302925328 -0.34753521
## symmetry_se             -0.011061722 -0.47297957  0.040607397  0.32248872
## fractal_dimension_se     0.015966751  0.02637426 -0.235015003  0.05588573
## texture_worst            0.712842576  0.17107919  0.106193121  0.03236116
## smoothness_worst        -0.066607812  0.38154991  0.244256617  0.08609053
## compactness_worst        0.132723306 -0.03768592 -0.049368228  0.19146725
## concavity_worst          0.084898970 -0.01626102  0.186111316  0.03233955
## concave.points_worst    -0.067299904  0.04964505  0.114437084 -0.06166150
## symmetry_worst           0.109288594 -0.47930911  0.197646331  0.13153765
## fractal_dimension_worst  0.164065711  0.06620909 -0.250274129  0.17547981
```

```
##                                PC9         PC10         PC11         PC12
## area_mean               -0.046921115  0.001334594 -0.053416738  0.461453950
## smoothness_mean          0.297479277 -0.117137074 -0.003677549  0.313713954
## compactness_mean         0.174575407  0.325570621 -0.073822671  0.093897800
## symmetry_mean           -0.570608106  0.250869323 -0.198736344  0.028080336
## fractal_dimension_mean   0.041259375 -0.115960448  0.057419921 -0.170697282
## radius_se               -0.222188766 -0.318382011 -0.015185145 -0.466279824
## texture_se               0.231006390  0.064854301 -0.271668713 -0.078185984
## smoothness_se           -0.272389923  0.252635449 -0.133999514 -0.046938360
## compactness_se           0.043067717  0.353963315 -0.049116595  0.133758160
## concavity_se            -0.047673254 -0.476510749 -0.327150263  0.069918764
## concave.points_se        0.086117926  0.140919112  0.522403430 -0.320670530
## symmetry_se              0.446227709 -0.134086526  0.117567699  0.066922569
## fractal_dimension_se    -0.275593890 -0.178658762  0.287987596  0.445888630
## texture_worst           -0.180475314 -0.071332061  0.298047593  0.127197002
## smoothness_worst         0.003461906 -0.208055225  0.048393525  0.008387421
## compactness_worst        0.142057315  0.307042156 -0.151600286 -0.161673138
## concavity_worst          0.097189858 -0.170004531 -0.429654738 -0.090040940
## concave.points_worst     0.106429265  0.084518824  0.220029991 -0.066401954
## symmetry_worst          -0.085966586 -0.159420476  0.162094046  0.003289351
## fractal_dimension_worst -0.084035459 -0.091005184  0.065938577 -0.200868122
##                               PC13        PC14        PC15         PC16
## area_mean               -0.22941906  0.28819366 -0.12653180  0.139605447
## smoothness_mean          0.20860595 -0.22443409  0.24969187  0.422181942
## compactness_mean         0.24039165  0.22790753 -0.22351274 -0.120976879
## symmetry_mean            0.03084608  0.04564346  0.21894729 -0.192868880
## fractal_dimension_mean   0.16967334  0.50849358 -0.33920274 -0.017830453
## radius_se                0.18818461 -0.25522842 -0.03507540  0.005665164
## texture_se              -0.46031399 -0.10391327 -0.17710531  0.020212229
## smoothness_se           -0.05997056  0.21030113  0.04121875  0.243971794
## compactness_se           0.23285729 -0.43655201 -0.22049136 -0.044334777
## concavity_se             0.16665393  0.06954031 -0.16047003  0.086017877
## concave.points_se       -0.07285482  0.03566715  0.06427652  0.124599391
## symmetry_se              0.08595085  0.12366646  0.30933377 -0.256017604
## fractal_dimension_se    -0.24841859 -0.17072884  0.11313079 -0.155832050
## texture_worst            0.43738186  0.11995009  0.12501323  0.021417845
## smoothness_worst        -0.10126687 -0.24532846 -0.21791857 -0.553645303
## compactness_worst        0.02088688 -0.21910654  0.09240204  0.082740311
## concavity_worst         -0.06819439  0.12587042  0.31554831 -0.080858129
## concave.points_worst    -0.29716697  0.18651424  0.01727178 -0.203660331
## symmetry_worst          -0.18265019 -0.13465122 -0.44404181  0.396241483
## fractal_dimension_worst -0.28408762  0.03668478  0.34454625  0.237884733
##                               PC17         PC18          PC19         PC20
## area_mean               -0.442028257  0.001957541  0.1315536269  0.122170053
## smoothness_mean          0.080490801  0.125944958 -0.0507537357  0.016178329
## compactness_mean         0.299464932 -0.534420816  0.0156963493 -0.380347312
## symmetry_mean           -0.134754660  0.027303252 -0.0488042230  0.043050279
## fractal_dimension_mean  -0.088455846  0.323803406  0.1175997989  0.260160391
## radius_se                0.119519109  0.015307554 -0.0149444084 -0.023863132
## texture_se               0.010667077 -0.025762183 -0.0005881574 -0.013436451
## smoothness_se            0.153439237  0.012647912 -0.0483452823  0.004786199
## compactness_se          -0.245455343  0.455051077  0.0114222057 -0.259469632
## concavity_se            -0.114766410 -0.198872973 -0.3965573165  0.043663288
## concave.points_se       -0.184491745 -0.154553544  0.3082658257  0.050247882
## symmetry_se             -0.119909402 -0.029652401 -0.0671303527  0.024638363
## fractal_dimension_se     0.365041850 -0.097541303  0.1166206037  0.160932991
## texture_worst           -0.007446169  0.028057196 -0.0039301711  0.010523081
## smoothness_worst        -0.257094120 -0.142427981  0.1004748406  0.005396813
## compactness_worst       -0.004321504 -0.221168946 -0.1186756337  0.703599850
## concavity_worst          0.238119202  0.236627166  0.5630584157 -0.103120908
## concave.points_worst     0.310284523  0.405095519 -0.5443036848 -0.043213002
## symmetry_worst           0.187502804 -0.002794369  0.1312018124 -0.063984627
## fractal_dimension_worst -0.367601446 -0.144508750 -0.1893838184 -0.399735895
```

```
plot(corred_data_PCA, type="b")
```



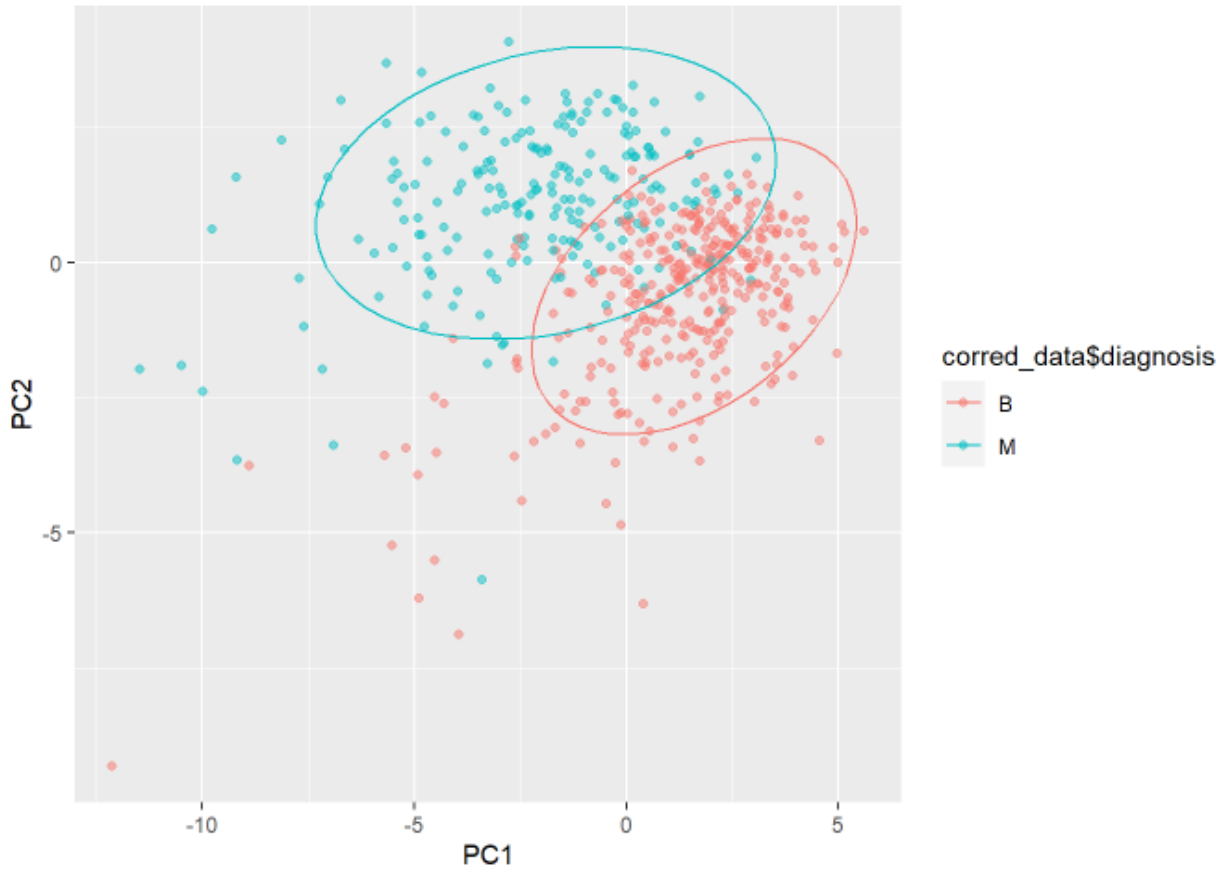

```
summary(corred_data_PCA)
```

```
## Importance of components:
##                           PC1    PC2    PC3     PC4     PC5     PC6     PC7
## Standard deviation     2.8944 1.7019 1.5051 1.23948 1.12116 1.09232 0.71120
## Proportion of Variance 0.4189 0.1448 0.1133 0.07682 0.06285 0.05966 0.02529
## Cumulative Proportion  0.4189 0.5637 0.6770 0.75377 0.81662 0.87628 0.90157
##                           PC8     PC9    PC10    PC11    PC12    PC13    PC14
## Standard deviation     0.66260 0.57349 0.52238 0.46342 0.41049 0.40363 0.36237
## Proportion of Variance 0.02195 0.01644 0.01364 0.01074 0.00842 0.00815 0.00657
## Cumulative Proportion  0.92352 0.93997 0.95361 0.96435 0.97277 0.98092 0.98748
##                           PC15    PC16    PC17    PC18    PC19    PC20
## Standard deviation     0.27413 0.23742 0.21163 0.18323 0.15048 0.13340
## Proportion of Variance 0.00376 0.00282 0.00224 0.00168 0.00113 0.00089
## Cumulative Proportion  0.99124 0.99406 0.99630 0.99798 0.99911 1.00000
```

The table shows that 95% of the variance is explained with 8 PC's in the transformed dataset.

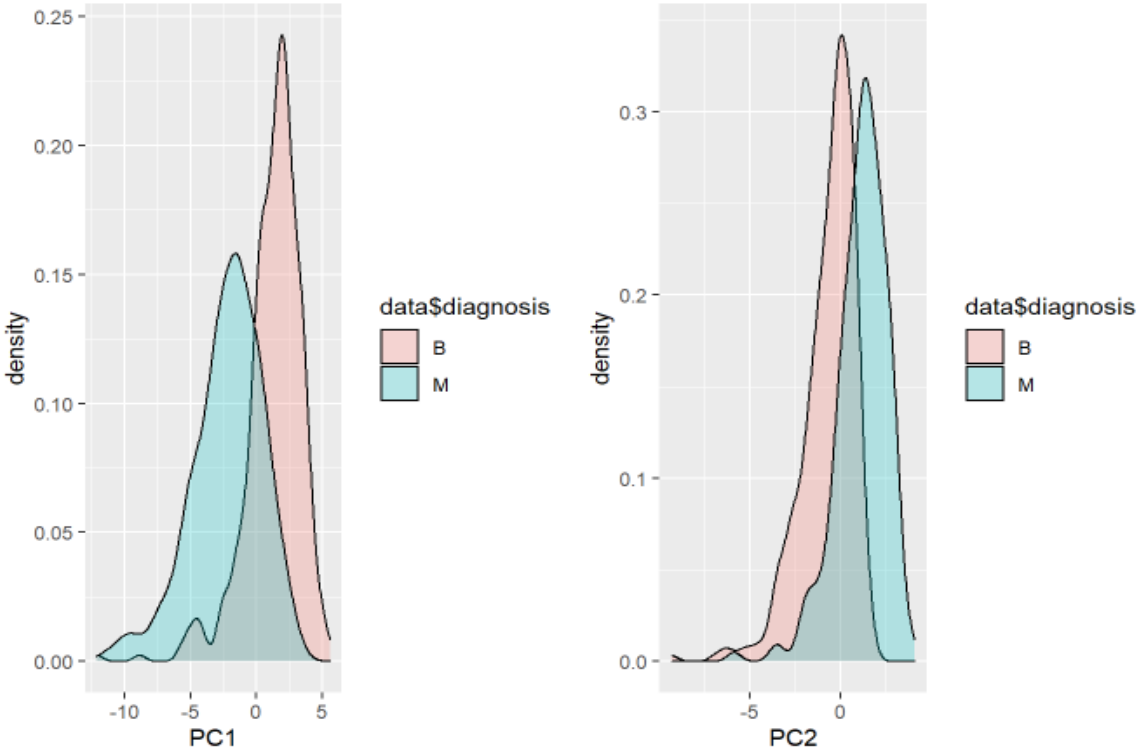
The figure below shows the first principal component separating samples into benign and malignant clusters.

```
ggplot(df_PCA, aes(x=PC1, y=PC2, col=corred_data$diagnosis)) +
  geom_point(alpha=0.5) +
  stat_ellipse()
```



The data of the first 2 components can be easly separated into two classes. This is caused by the fact that the variance explained by these components is not large. The data can be easly separated.

```
# Plot of PC1 and PC2
PCA1 <- ggplot(df_PCA, aes(x=PC1, fill=data$diagnosis)) + geom_density(alpha=0.25)
PCA2 <- ggplot(df_PCA, aes(x=PC2, fill=data$diagnosis)) + geom_density(alpha=0.25)
grid.arrange(PCA1, PCA2, ncol=2)
```



Linear Discriminant Analysis (LDA)

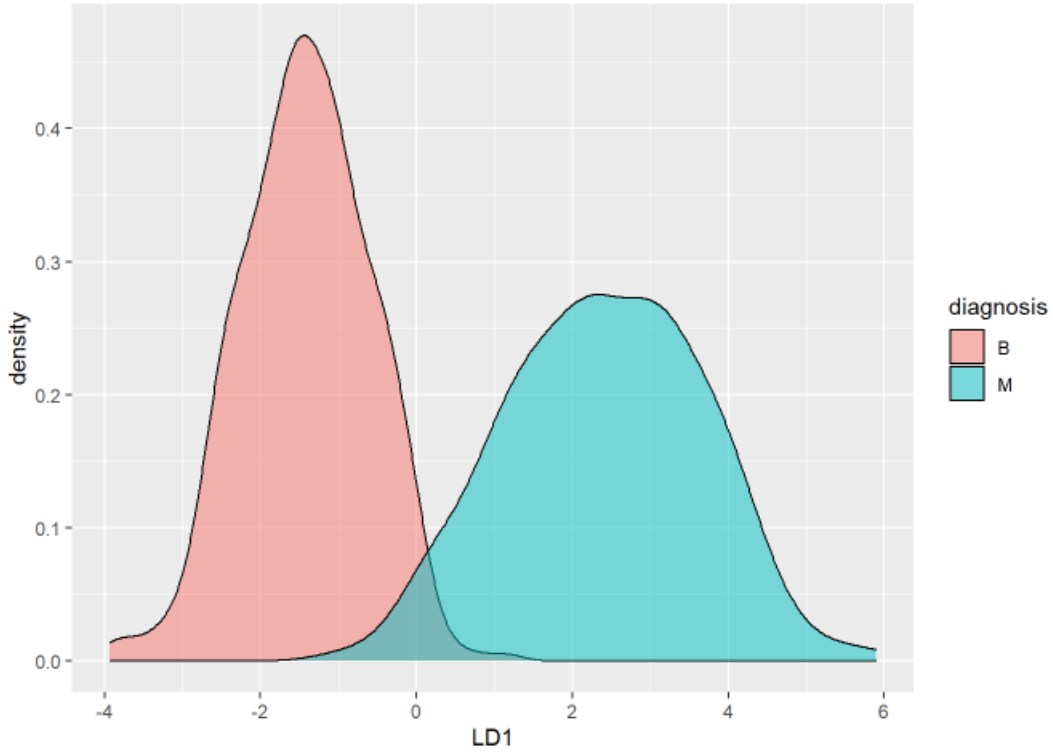
Another approach is to use the Linear Discriminant Analysis (LDA) instead of PCA. LDA takes in consideration the different classes and could get better results. The particularity of LDA is that it models the distribution of predictors separately in each of the response classes, and then it uses Bayes' theorem to estimate the probability. It is important to know that LDA assumes a normal distribution for each class, a class-specific mean, and a common variance.

```
# Linear Discriminant Analysis (LDA)

# Data with LDA
data_LDA <- MASS::lda(diagnosis~., data = data, center = TRUE, scale = TRUE)
data_LDA
```

```
## Call:
## lda(diagnosis ~ ., data = data, center = TRUE, scale = TRUE)
##
## Prior probabilities of groups:
##         B         M
## 0.6274165 0.3725835
##
## Group means:
##         id radius_mean texture_mean perimeter_mean area_mean smoothness_mean
## B 26543825    12.14652     17.91476       78.07541  462.7902      0.09247765
## M 36818050    17.46283     21.60491      115.36538  978.3764      0.10289849
##   compactness_mean concavity_mean concave.points_mean symmetry_mean
## B       0.08008462     0.04605762          0.02571741      0.174186
## M       0.14518778     0.16077472          0.08799000      0.192909
##   fractal_dimension_mean radius_se texture_se perimeter_se  area_se
## B             0.06286739 0.2840824   1.220380     2.000321 21.13515
## M             0.06268009 0.6090825   1.210915     4.323929 72.67241
##   smoothness_se compactness_se concavity_se concave.points_se symmetry_se
## B   0.007195902     0.02143825   0.02599674       0.009857653  0.02058381
## M   0.006780094     0.03228117   0.04182401       0.015060472  0.02047240
##   fractal_dimension_se radius_worst texture_worst perimeter_worst area_worst
## B          0.003636051     13.37980      23.51507        87.00594   558.8994
## M          0.004062406     21.13481      29.31821       141.37033  1422.2863
##   smoothness_worst compactness_worst concavity_worst concave.points_worst
## B        0.1249595         0.1826725       0.1662377           0.07444434
## M        0.1448452         0.3748241       0.4506056           0.18223731
##   symmetry_worst fractal_dimension_worst
## B      0.2702459              0.07944207
## M      0.3234679              0.09152995
```

```
## Coefficients of linear discriminants:
##                                LD1
## id                     -2.512117e-10
## radius_mean            -1.080876e+00
## texture_mean            2.338408e-02
## perimeter_mean          1.172707e-01
## area_mean               1.595690e-03
## smoothness_mean         5.251575e-01
## compactness_mean       -2.094197e+01
## concavity_mean          6.955923e+00
## concave.points_mean     1.047567e+01
## symmetry_mean           4.938898e-01
## fractal_dimension_mean -5.937663e-02
## radius_se               2.101503e+00
## texture_se             -3.979869e-02
## perimeter_se           -1.121814e-01
## area_se                -4.083504e-03
## smoothness_se           7.987663e+01
## compactness_se          1.387026e-01
## concavity_se           -1.768261e+01
## concave.points_se       5.350520e+01
## symmetry_se             8.143611e+00
## fractal_dimension_se   -3.431356e+01
## radius_worst            9.677207e-01
## texture_worst           3.540591e-02
## perimeter_worst        -1.204507e-02
## area_worst             -5.012127e-03
## smoothness_worst        2.612258e+00
## compactness_worst       3.636892e-01
## concavity_worst         1.880699e+00
## concave.points_worst    2.218189e+00
## symmetry_worst          2.783102e+00
## fractal_dimension_worst 2.117830e+01
```

```
# Visualization of LDA data frame
data_LDA_predict <- predict(data_LDA, data)$x %>% as.data.frame() %>% cbind(diagnosis=data$diagnosis)

ggplot(data_LDA_predict, aes(x=LD1, fill=diagnosis)) +
  geom_density(alpha=0.5)
```



Testing and training sets

The data is splitted into Train (80%) and Test (20%) sets, in order to predict is whether a cancer cell is Benign or Malignant, by building machine learning classification models.
The training and test sets had equal proportions of diagnosis types, as seen in the data frame below.

```
# Creation of the partition 80% and 20%
set.seed(1)
data2 <- cbind (diagnosis=data$diagnosis, corred_data)
data_sampling_index <- createDataPartition(data$diagnosis, times=1, p=0.8, list = FALSE)
train_data <- data2[data_sampling_index, ]
test_data <- data2[-data_sampling_index, ]


data.frame(Dataset = c("Train", "Test"),
           Benign = c(mean(train_data$diagnosis == "B"), mean(train_data$diagnosis == "B")),
           Malignant = c(mean(test_data$diagnosis == "M"), mean(test_data$diagnosis == "M")))
```

```
##   Dataset   Benign Malignant
## 1   Train 0.627193 0.3716814
## 2    Test 0.627193 0.3716814
```

i. Logistic Regression Model

Logistic regression is the most commonly used form of generalized linear model (GLM). Linear regression assumes that the predictor, X, and the outcome Y, follow a bivariate normal distribution such that the conditional expectation, i.e. the expected outcome Y for a given predictor X, fits the regression line.

$$p(x) = \Pr(Y = 1|X = x) = \beta_0 + \beta_1 x$$

Logistic regression is an extension of linear regression, where g is a function that transforms the probability, p, to log odds (g(p) = log p/1−p) such the conditional probability can be modelled as below. A logistic regression model was developed using the caret package to train the normalized train set before predicting outcomes in the normalised test set. In addition, the model was also run incorporating the outputs from PCA via caret's pre-processing functionality. Dimension reduction is most useful in highly dimensional data but there is some evidence in the literature that dimension reduction via PCA can also improve the predictive accuracy of models such as logistic regression (Hsu, Huang, and Chen 2014; Sabharwal and Anjum 2016).

$$g\{\Pr(Y = 1|X = x)\} = \beta_0 + \beta_1 x$$

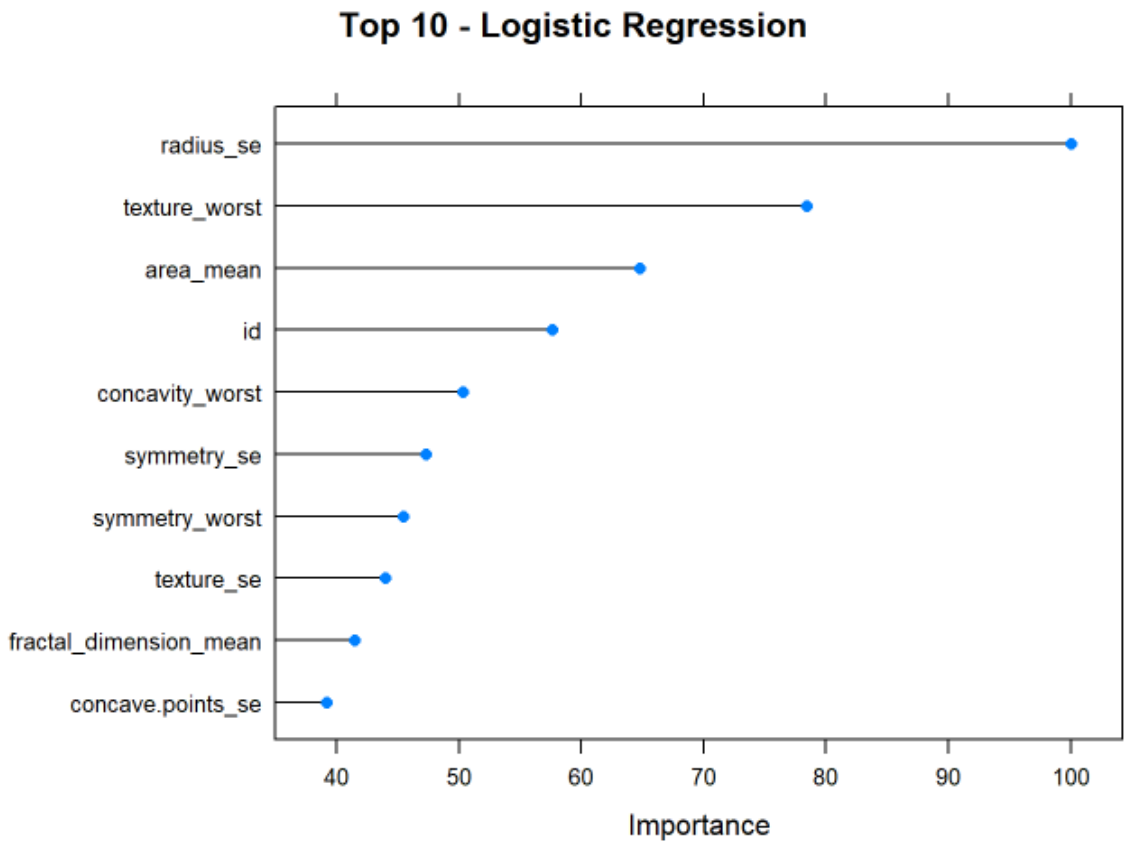
The most common logistic regression models a binary outcome; something that can take two values such as true/false, yes/no, and so on. The binary logistic model is used to estimate the probability of a binary response based on one or more predictor (or independent) variables (features).

```
# Creation of Logistic Regression Model
Lreg_model <- train(diagnosis ~.,
                    train_data,
                    method = "glm",
                    metric = "ROC",
                    preProcess = c("scale", "center"),  # in order to normalize the data
                    trControl= fitControl)
```

```
# Prediction
Lreg_model_prediction <- predict(Lreg_model, test_data)

# Confusion matrix
confusionmatrix_Lreg <- confusionMatrix(Lreg_model_prediction, test_data$diagnosis, positive = "M")
confusionmatrix_Lreg
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  B  M
##          B 70  5
##          M  1 37
##
##                Accuracy : 0.9469
##                  95% CI : (0.888, 0.9803)
##     No Information Rate : 0.6283
##     P-Value [Acc > NIR] : 1.866e-15
##
##                   Kappa : 0.8841
##
##  Mcnemar's Test P-Value : 0.2207
##
##             Sensitivity : 0.8810
##             Specificity : 0.9859
##          Pos Pred Value : 0.9737
##          Neg Pred Value : 0.9333
##              Prevalence : 0.3717
##          Detection Rate : 0.3274
##    Detection Prevalence : 0.3363
##       Balanced Accuracy : 0.9334
##
##        'Positive' Class : M
##
```

```r
# Plot of top important variables
plot(varImp(Lreg_model), top=10, main="Top 10 - Logistic Regression")
```



ii. K Nearest Neighbour (KNN) Model

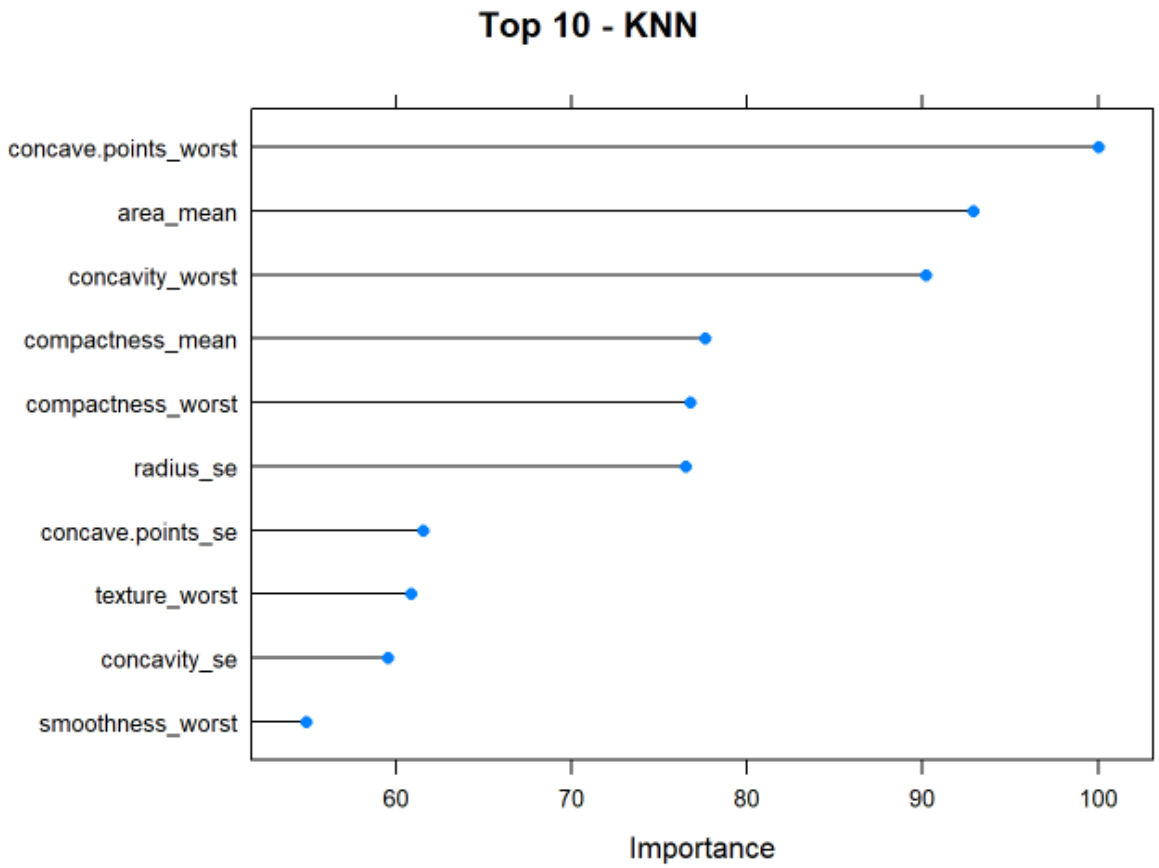
The k-Nearest neighbour model (kNN) is a simple approach to supervised machine learning that assumes proximity equates to similarity, once again measuring the Euclidean distance (2) between two points in multidimensional data. Unlike hierarchical and k-means clustering, the KNN model is a form of supervised learning, i.e. it relies on and makes use of the diagnosis labels in the training set in order to predict diagnosis in an unlabeled test set.

Whereas in k-means clustering, the k represents the number of clusters, or centres, within the data, in the kNN model, k represents the number of neighbours for any given data-point. As with the use of bins in smoothing, larger values of k result in smoother estimates. k is a tuning parameter within the train function for the kNN model (Kuhn 2019), and cross-validation within the train set was used to tune a value for k between 1 and 30 in increments of 2 to optimise the model before using it to predict outcome in the test set.

```
# Creation of K Nearest Neighbor (KNN) Model
Knn_model <- train(diagnosis~.,
                   train_data,
                   method="knn",
                   metric="ROC",
                   preProcess = c('center', 'scale'),
                   tuneLength=8, #The tuneLength parameter tells the algorithm to try different default values fo
r the main parameter
                   #In this case we used 8 default values
                   trControl=fitControl)
# Prediction
prediction_Knn <- predict(Knn_model, test_data)

# Confusion matrix
confusionmatrix_Knn <- confusionMatrix(prediction_Knn, test_data$diagnosis, positive = "M")
confusionmatrix_Knn
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  B  M
##          B 70  7
##          M  1 35
##
##                Accuracy : 0.9292
##                  95% CI : (0.8653, 0.9689)
##     No Information Rate : 0.6283
##     P-Value [Acc > NIR] : 1.372e-13
##
##                   Kappa : 0.8439
##
##  Mcnemar's Test P-Value : 0.0771
##
##             Sensitivity : 0.8333
##             Specificity : 0.9859
##          Pos Pred Value : 0.9722
##          Neg Pred Value : 0.9091
##              Prevalence : 0.3717
##          Detection Rate : 0.3097
##    Detection Prevalence : 0.3186
##       Balanced Accuracy : 0.9096
##
##        'Positive' Class : M
##
```

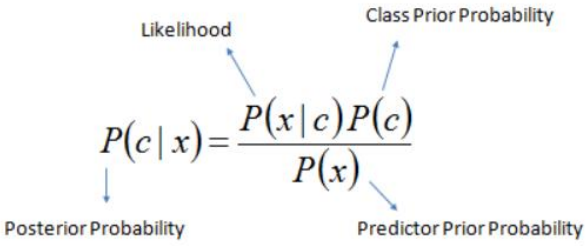
```
# Plot of top important variables
plot(varImp(Knn_model), top=10, main="Top 10 - KNN")
```



iii. Naive Bayes Model

The Naive Bayesian classifier is based on Bayes' theorem with the independence assumptions between predictors. A Naive Bayesian model is easy to build, with no complicated iterative parameter estimation which makes it particularly useful for very large datasets. Despite its simplicity, the Naive Bayesian classifier often does surprisingly well and is widely used because it often outperforms more sophisticated classification methods.

Bayes theorem provides a way of calculating the posterior probability, P(c|x), from P(c), P(x), and P(x|c). Naive Bayes classifier assume that the effect of the value of a predictor (x) on a given class (c) is independent of the values of other predictors. This assumption is called class conditional independence.



$$P(c \mid \mathrm{X}) = P(x_1 \mid c) \times P(x_2 \mid c) \times \cdots \times P(x_n \mid c) \times P(c)$$

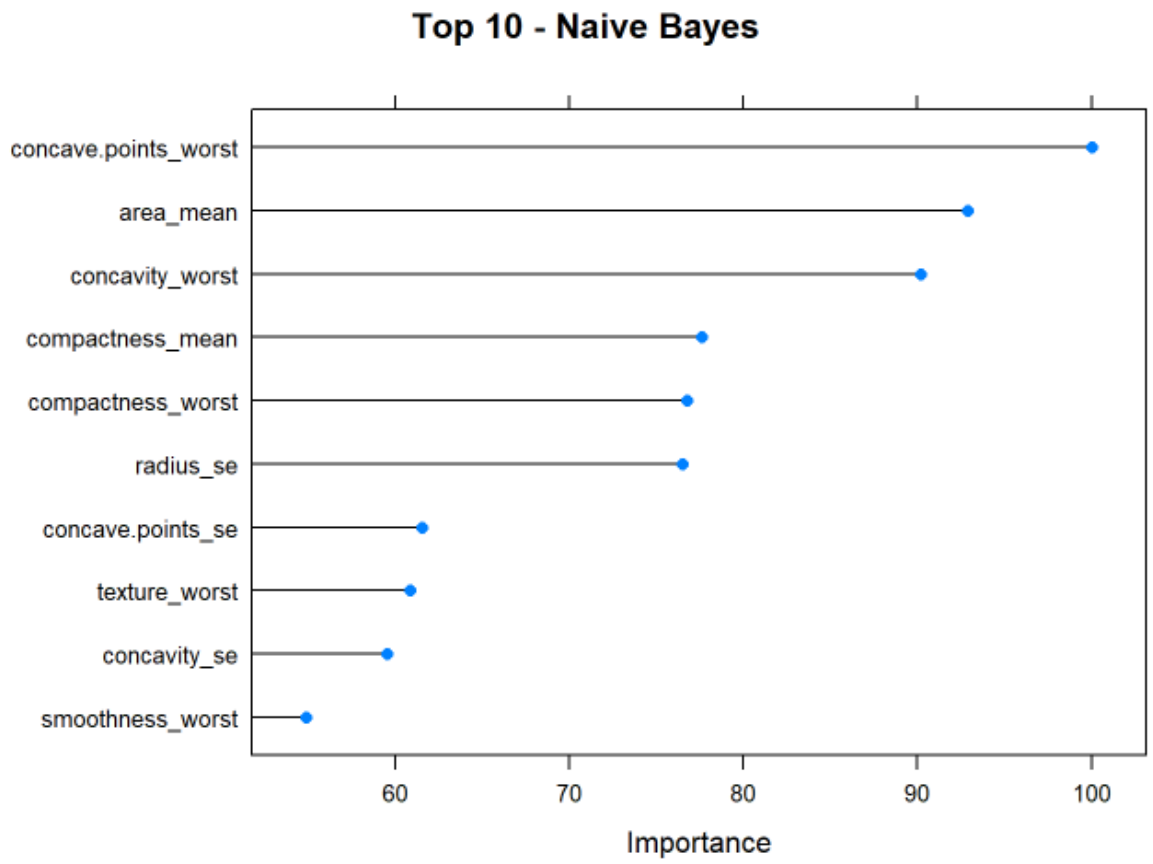
```r
# Creation of Naive Bayes Model
nbayes_model <- train(diagnosis~.,
                      train_data,
                      method="nb",
                      metric="ROC",
                      preProcess=c('center', 'scale'), #in order to normalize de data
                      trace=FALSE,
                      trControl=fitControl)
```

```r
# Prediction
nbayes_prediction <- predict(nbayes_model, test_data)
```

```r
# Confusion matrix
confusionmatrix_nbayes <- confusionMatrix(nbayes_prediction, test_data$diagnosis, positive = "M")
confusionmatrix_nbayes
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  B  M
##          B 67  8
##          M  4 34
##
##                Accuracy : 0.8938
##                  95% CI : (0.8218, 0.9439)
##     No Information Rate : 0.6283
##     P-Value [Acc > NIR] : 1.762e-10
##
##                   Kappa : 0.7681
##
##  Mcnemar's Test P-Value : 0.3865
##
##             Sensitivity : 0.8095
##             Specificity : 0.9437
##          Pos Pred Value : 0.8947
##          Neg Pred Value : 0.8933
##              Prevalence : 0.3717
##          Detection Rate : 0.3009
##    Detection Prevalence : 0.3363
##       Balanced Accuracy : 0.8766
##
##        'Positive' Class : M
##
```

```
# Plot of top important variables
plot(varImp(nbayes_model), top=10, main="Top 10 - Naive Bayes")
```



We can note the accuracy with such model. We will later describe better these metrics, where: Sensitivity (recall) represent the true positive rate: the proportions of actual positives correctly identified. Specificity is the true negative rate: the proportion of actual negatives correctly identified. Accuracy is the general score of the classifier model performance as it is the ratio of how many samples are correctly classified to all samples. F1 score: the harmonic mean of precision and sensitivity. Accuracy and F1 score would be used to compare the result with the benchmark model. Precision: the number of correct positive results divided by the number of all positive results returned by the classifier.

iv. Random Forest Model

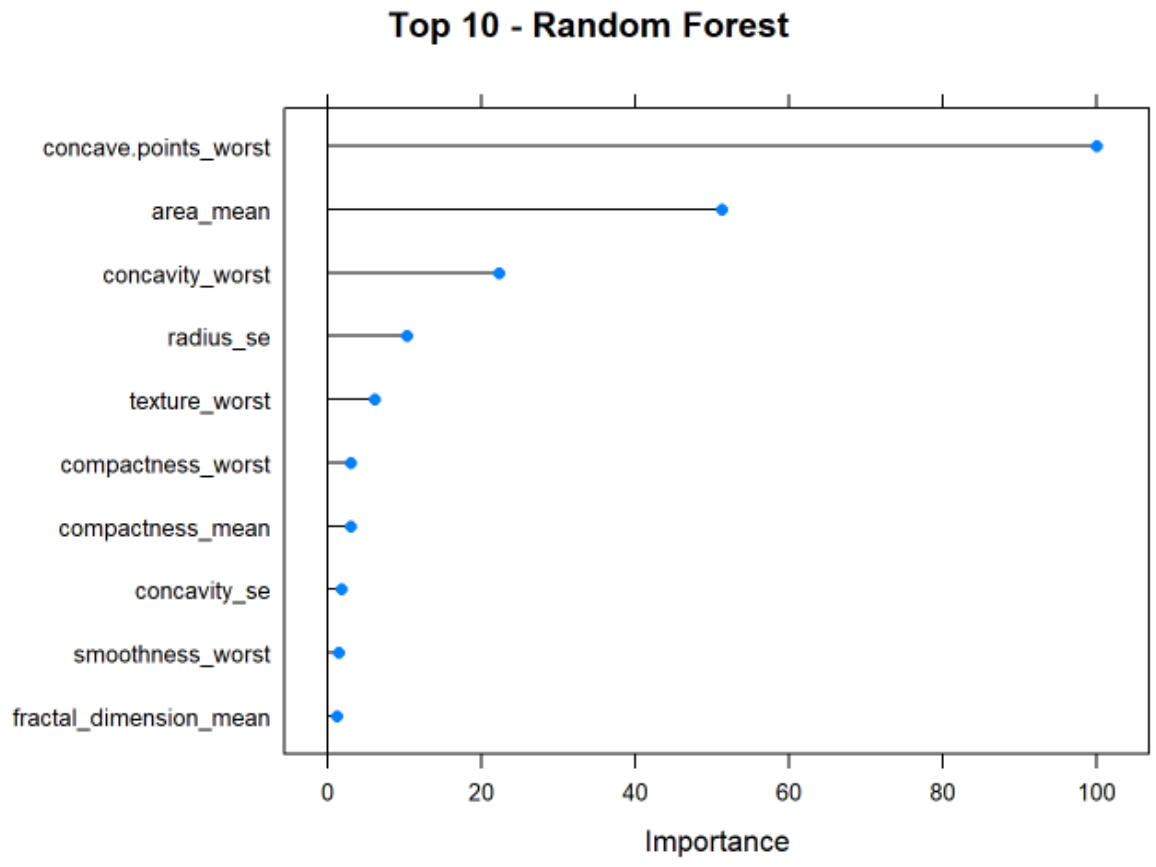
Random forests or random decision forests are an ensemble learning method for classification, regression and other tasks that operate by constructing a multitude of decision trees at training time and outputting the class that is the mode of the classes (classification) or mean/average prediction (regression) of the individual trees.

Random forests are a very popular machine learning approach that addresses the shortcomings of decision trees using a clever idea. The goal is to improve prediction performance and reduce instability by averaging multiple decision trees (a forest of trees constructed with randomness). Random forest is another ensemble method based on decision trees. It splits data into sub-samples, trains decision tree classifiers on each sub-sample and averages prediction of each classifier. Splitting dataset causes higher bias but it is compensated by large decrease in variance. Random Forest is a supervised learning algorithm and it is flexible, easy to use machine learning algorithm that produces, even without hyper-parameter tuning, a great result most of the time. It is also one of the most used algorithms, because of its simplicity and the fact that it can be used for both classification and regression tasks. Random forest builds multiple decision trees and merges them together to get a more accurate and stable prediction.

```
# Creation of Random Forest Model
Rforest_model <- train(diagnosis~.,
                       train_data,
                       method="rf",
                       metric="ROC",
                       preProcess = c('center', 'scale'),
                       trControl=fitControl)
# Prediction
prediction_Rforest <- predict(Rforest_model , test_data)

# Confusion matrix
confusionmatrix_Rforest <- confusionMatrix(prediction_Rforest, test_data$diagnosis, positive = "M")
confusionmatrix_Rforest
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  B  M
##          B 69  5
##          M  2 37
##
##                Accuracy : 0.9381
##                  95% CI : (0.8765, 0.9747)
##     No Information Rate : 0.6283
##     P-Value [Acc > NIR] : 1.718e-14
##
##                   Kappa : 0.8654
##
##  Mcnemar's Test P-Value : 0.4497
##
##             Sensitivity : 0.8810
##             Specificity : 0.9718
##          Pos Pred Value : 0.9487
##          Neg Pred Value : 0.9324
##              Prevalence : 0.3717
##          Detection Rate : 0.3274
##    Detection Prevalence : 0.3451
##       Balanced Accuracy : 0.9264
##
##        'Positive' Class : M
##
```

```r
# Plot of top important variables
plot(varImp(Rforest_model), top=10, main="Top 10 - Random Forest")
```



v. Neural Network with PCA Model

A neural network is a network or circuit of neurons, or in a modern sense, an artificial neural network, composed of artificial neurons or nodes.[1] Thus a neural network is either a biological neural network, made up of real biological neurons, or an artificial neural network, for solving artificial intelligence (AI) problems. The connections of the biological neuron are modeled as weights. A positive weight reflects an excitatory connection, while negative values mean inhibitory connections. All inputs are modified by a weight and summed. This activity is referred to as a linear combination. Finally, an activation function controls the amplitude of the output. For example, an acceptable range of output is usually between 0 and 1, or it could be −1 and 1.

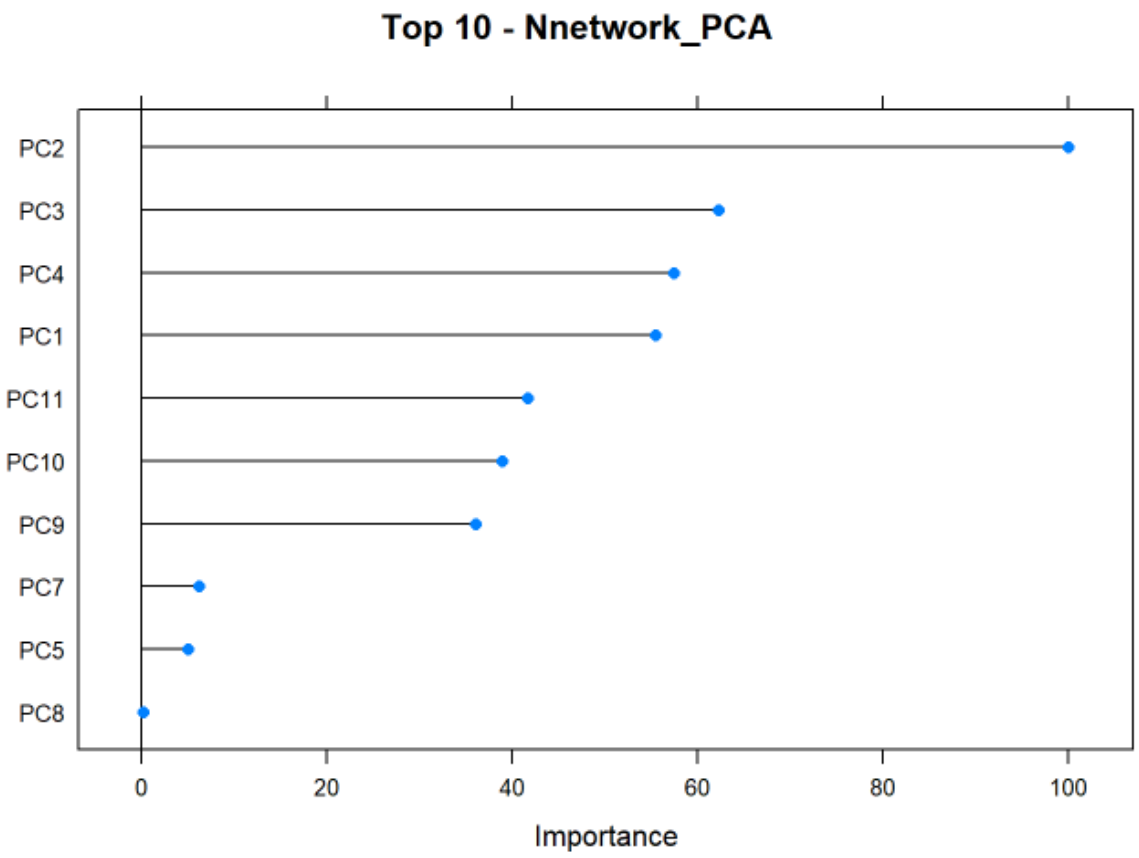
These artificial networks may be used for predictive modeling, adaptive control and applications where they can be trained via a dataset. Self-learning resulting from experience can occur within networks, which can derive conclusions from a complex and seemingly unrelated set of information.

```
# Creation of Random Forest Model (**Neural Network with PCA model may take some time)
Nnetwork_PCA_model <- train(diagnosis~.,
                        train_data,
                        method="nnet",
                        metric="ROC",
                        preProcess=c('center', 'scale', 'pca'),
                        tuneLength=10,
                        trace=FALSE,
                        trControl=fitControl)
# Prediction
prediction_Nnetwork_PCA <- predict(Nnetwork_PCA_model, test_data)
# Confusion matrix
confusionmatrix_Nnetwork_PCA <- confusionMatrix(prediction_Nnetwork_PCA, test_data$diagnosis, positive = "M")
confusionmatrix_Nnetwork_PCA
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  B  M
##          B 70  4
##          M  1 38
##
##                Accuracy : 0.9558
##                  95% CI : (0.8998, 0.9855)
##     No Information Rate : 0.6283
##     P-Value [Acc > NIR] : <2e-16
##
##                   Kappa : 0.9039
##
##  Mcnemar's Test P-Value : 0.3711
##
##             Sensitivity : 0.9048
##             Specificity : 0.9859
##          Pos Pred Value : 0.9744
##          Neg Pred Value : 0.9459
##              Prevalence : 0.3717
##          Detection Rate : 0.3363
##    Detection Prevalence : 0.3451
##       Balanced Accuracy : 0.9453
##
##        'Positive' Class : M
##
```

The most important variables for best prediction are as follows:

```
# Plot of top important variables
plot(varImp(Nnetwork_PCA_model), top=10, main="Top 10 - Nnetwork_PCA")
```



vi. Neural Network with LDA Model

We are going to create a training and test set of LDA data created previously;

```
# Creation of training set and test set with LDA modified data
train_data_lda <- data_LDA_predict[data_sampling_index, ]
test_data_lda <- data_LDA_predict[-data_sampling_index, ]

# Creation of Neural Network with LDA Mode (**Neural Network with LDA model may take some time)
Nnetwork_LDA_model <- train(diagnosis~.,
                        train_data_lda,
                        method="nnet",
                        metric="ROC",
                        preProcess=c('center', 'scale'),
                        tuneLength=10,
                        trace=FALSE,
                        trControl=fitControl)
# Prediction
prediction_Nnetwork_LDA <- predict(Nnetwork_LDA_model, test_data_lda)
# Confusion matrix
confusionmatrix_Nnetwork_LDA <- confusionMatrix(prediction_Nnetwork_LDA, test_data_lda$diagnosis, positive = "M")
confusionmatrix_Nnetwork_LDA
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  B  M
##          B 70  5
##          M  1 37
##
##                Accuracy : 0.9469
##                  95% CI : (0.888, 0.9803)
##     No Information Rate : 0.6283
##     P-Value [Acc > NIR] : 1.866e-15
##
##                   Kappa : 0.8841
##
##  Mcnemar's Test P-Value : 0.2207
##
##             Sensitivity : 0.8810
##             Specificity : 0.9859
##          Pos Pred Value : 0.9737
##          Neg Pred Value : 0.9333
##              Prevalence : 0.3717
##          Detection Rate : 0.3274
##    Detection Prevalence : 0.3363
##       Balanced Accuracy : 0.9334
##
##        'Positive' Class : M
##
```

b. Results

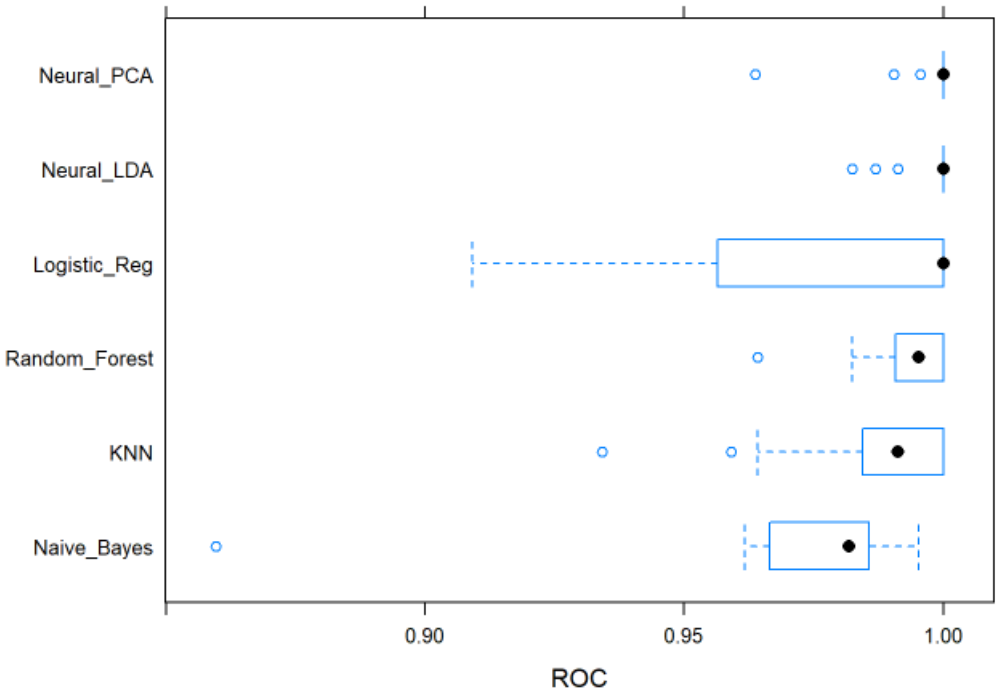
We will make comparisons between different models and evaluate the results.

```
# Creation of the list of all models
models_list <- list(Naive_Bayes=nbayes_model,
                    Logistic_Reg=Lreg_model,
                    Random_Forest=Rforest_model,
                    KNN=Knn_model,
                    Neural_PCA=Nnetwork_PCA_model,
                    Neural_LDA=Nnetwork_LDA_model)
models_results <- resamples(models_list)

# Print the summary of models
summary(models_results)
```

```
## Call:
## summary.resamples(object = models_results)
##
## Models: Naive_Bayes, Logistic_Reg, Random_Forest, KNN, Neural_PCA, Neural_LDA
## Number of resamples: 15
##
## ROC
##                    Min.   1st Qu.    Median      Mean   3rd Qu.      Max. NA's
## Naive_Bayes   0.8596491 0.9665072 0.9818182 0.9714620 0.9856459 0.9952153    0
## Logistic_Reg  0.9090909 0.9564394 1.0000000 0.9770813 1.0000000 1.0000000    0
## Random_Forest 0.9641148 0.9908293 0.9952153 0.9928549 1.0000000 1.0000000    0
## KNN           0.9342105 0.9844498 0.9912281 0.9862945 1.0000000 1.0000000    0
## Neural_PCA    0.9636364 1.0000000 1.0000000 0.9966454 1.0000000 1.0000000    0
## Neural_LDA    0.9824561 1.0000000 1.0000000 0.9973684 1.0000000 1.0000000    0
##
## Sens
##                    Min.   1st Qu.    Median      Mean   3rd Qu. Max. NA's
## Naive_Bayes   0.8421053 0.8947368 0.9473684 0.9336842 0.9736842    1    0
## Logistic_Reg  0.8947368 0.9473684 0.9473684 0.9615789 1.0000000    1    0
## Random_Forest 0.9000000 0.9473684 1.0000000 0.9722807 1.0000000    1    0
## KNN           0.8947368 1.0000000 1.0000000 0.9824561 1.0000000    1    0
## Neural_PCA    0.8947368 0.9736842 1.0000000 0.9824561 1.0000000    1    0
## Neural_LDA    0.9473684 1.0000000 1.0000000 0.9929825 1.0000000    1    0
##
## Spec
##                    Min.   1st Qu.    Median      Mean   3rd Qu. Max. NA's
## Naive_Bayes   0.8181818 0.8333333 0.9090909 0.9000000 0.9166667    1    0
## Logistic_Reg  0.8181818 0.9128788 1.0000000 0.9590909 1.0000000    1    0
## Random_Forest 0.8181818 0.8333333 0.9090909 0.9186869 1.0000000    1    0
## KNN           0.7500000 0.8181818 0.9090909 0.8772727 0.9128788    1    0
## Neural_PCA    0.8333333 1.0000000 1.0000000 0.9717172 1.0000000    1    0
## Neural_LDA    0.8181818 0.9583333 1.0000000 0.9651515 1.0000000    1    0
```

As we can see from the following plot, two models, Naive_Bayes and Logistic_Reg models have great variability.

```
# Plot of the models results
bwplot(models_results, metric="ROC")
```




```
# Confusion matrix of the models
confusionmatrix_list <- list(
  Naive_Bayes=confusionmatrix_nbayes,
  Logistic_regr=confusionmatrix_Rforest,
  KNN=confusionmatrix_Knn,
  NNetwork_PCA=confusionmatrix_Nnetwork_PCA,
  NNetwork_LDA=confusionmatrix_Nnetwork_LDA)

confusionmatrix_list
```

```
## $Naive_Bayes
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  B  M
##          B 67  8
##          M  4 34
##
##                Accuracy : 0.8938
##                  95% CI : (0.8218, 0.9439)
##     No Information Rate : 0.6283
##     P-Value [Acc > NIR] : 1.762e-10
##
##                   Kappa : 0.7681
##
##  Mcnemar's Test P-Value : 0.3865
##
##             Sensitivity : 0.8095
##             Specificity : 0.9437
##          Pos Pred Value : 0.8947
##          Neg Pred Value : 0.8933
##              Prevalence : 0.3717
##          Detection Rate : 0.3009
##    Detection Prevalence : 0.3363
##       Balanced Accuracy : 0.8766
##
##        'Positive' Class : M
##
```

```
## $Logistic_regr
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  B  M
##          B 69  5
##          M  2 37
##
##                Accuracy : 0.9381
##                  95% CI : (0.8765, 0.9747)
##     No Information Rate : 0.6283
##     P-Value [Acc > NIR] : 1.718e-14
##
##                   Kappa : 0.8654
##
##  Mcnemar's Test P-Value : 0.4497
##
##             Sensitivity : 0.8810
##             Specificity : 0.9718
##          Pos Pred Value : 0.9487
##          Neg Pred Value : 0.9324
##              Prevalence : 0.3717
##          Detection Rate : 0.3274
##    Detection Prevalence : 0.3451
##       Balanced Accuracy : 0.9264
##
##        'Positive' Class : M
##
##       Balanced Accuracy : 0.9096
##
##        'Positive' Class : M
##
```

```
## $NNetwork_PCA
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  B  M
##          B 70  4
##          M  1 38
##
##                Accuracy : 0.9558
##                  95% CI : (0.8998, 0.9855)
##     No Information Rate : 0.6283
##     P-Value [Acc > NIR] : <2e-16
##
##                   Kappa : 0.9039
##
##  Mcnemar's Test P-Value : 0.3711
##
##             Sensitivity : 0.9048
##             Specificity : 0.9859
##          Pos Pred Value : 0.9744
##          Neg Pred Value : 0.9459
##              Prevalence : 0.3717
##          Detection Rate : 0.3363
##    Detection Prevalence : 0.3451
##       Balanced Accuracy : 0.9453
##
##        'Positive' Class : M
```

```
## $NNetwork_LDA
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  B  M
##          B 70  5
##          M  1 37
##
##                Accuracy : 0.9469
##                  95% CI : (0.888, 0.9803)
##     No Information Rate : 0.6283
##     P-Value [Acc > NIR] : 1.866e-15
##
##                   Kappa : 0.8841
##
##  Mcnemar's Test P-Value : 0.2207
##
##             Sensitivity : 0.8810
##             Specificity : 0.9859
##          Pos Pred Value : 0.9737
##          Neg Pred Value : 0.9333
##              Prevalence : 0.3717
##          Detection Rate : 0.3274
##    Detection Prevalence : 0.3363
##       Balanced Accuracy : 0.9334
##
##        'Positive' Class : M
```

```
confusionmatrix_list_results <- sapply(confusionmatrix_list, function(x) x$byClass)
confusionmatrix_list_results
```

```
##                      Naive_Bayes Logistic_regr       KNN NNetwork_PCA
## Sensitivity            0.8095238     0.8809524 0.8333333    0.9047619
## Specificity            0.9436620     0.9718310 0.9859155    0.9859155
## Pos Pred Value         0.8947368     0.9487179 0.9722222    0.9743590
## Neg Pred Value         0.8933333     0.9324324 0.9090909    0.9459459
## Precision              0.8947368     0.9487179 0.9722222    0.9743590
## Recall                 0.8095238     0.8809524 0.8333333    0.9047619
## F1                     0.8500000     0.9135802 0.8974359    0.9382716
## Prevalence             0.3716814     0.3716814 0.3716814    0.3716814
## Detection Rate         0.3008850     0.3274336 0.3097345    0.3362832
## Detection Prevalence   0.3362832     0.3451327 0.3185841    0.3451327
## Balanced Accuracy      0.8765929     0.9263917 0.9096244    0.9453387
##                      NNetwork_LDA
## Sensitivity             0.8809524
## Specificity             0.9859155
## Pos Pred Value          0.9736842
## Neg Pred Value          0.9333333
## Precision               0.9736842
## Recall                  0.8809524
## F1                      0.9250000
## Prevalence              0.3716814
## Detection Rate          0.3274336
## Detection Prevalence    0.3362832
## Balanced Accuracy       0.9334339
```

6. Discussion

We will now describe the metrics that we will compare in this section.

Accuracy is our starting point. It 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 is the number of True Positives divided by the number of True Positives and False Positives. In other words, it is the number of positive predictions divided by the total number of positive class values predicted. It is also called the Positive Predictive Value (PPV). A low precision can also indicate a large number of False Positives.
Recall (Sensitivity) is the number of True Positives divided by the number of True Positives and the number of False Negatives. Put another way it is the number of positive predictions divided by the number of positive class values in the test data. It is also called Sensitivity or the True Positive Rate. Recall can be thought of as a measure of a classifier's completeness. A low recall indicates many False Negatives.

The F1 Score is the 2 x ((precision x recall) / (precision + recall)). It is also called the F Score or the F Measure. In other words, the F1 score conveys the balance between the precision and the recall.

The best results for sensitivity (detection of breast cancer malign cases) is Neural Network with PCA model which also has a great F1 score.

```r
# Find the best result for each metric
confusionmatrix_results_max <- apply(confusionmatrix_list_results, 1, which.is.max)
output_report <- data.frame(metric=names(confusionmatrix_results_max),
                            best_model=colnames(confusionmatrix_list_results)
                            [confusionmatrix_results_max],
                            value=mapply(function(x,y)
                            {confusionmatrix_list_results[x,y]},
                            names(confusionmatrix_results_max),
                            confusionmatrix_results_max))
rownames(output_report) <- NULL
output_report
```

```
##                  metric   best_model     value
## 1           Sensitivity NNetwork_PCA 0.9047619
## 2           Specificity          KNN 0.9859155
## 3        Pos Pred Value NNetwork_PCA 0.9743590
## 4        Neg Pred Value NNetwork_PCA 0.9459459
## 5             Precision NNetwork_PCA 0.9743590
## 6                Recall NNetwork_PCA 0.9047619
## 7                    F1 NNetwork_PCA 0.9382716
## 8            Prevalence          KNN 0.3716814
## 9        Detection Rate NNetwork_PCA 0.3362832
## 10 Detection Prevalence NNetwork_PCA 0.3451327
## 11    Balanced Accuracy NNetwork_PCA 0.9453387
```

7. Conclusion

We investigated several machine learning models have been used in this project and we selected the optimal model by selecting a high accuracy level combinated with a low rate of false-negatives (the means that the metric is high sensitivity).

The Neural Network with PCA Model has the optimal results for F1 (0.9382716), Sensitivity (0.9047619) and Balanced Accuracy (0.9453387)

The objective of this project was to use the Wisconsin breast cancer data set to train different algorithms to accurately diagnosis breast cancer. An exploratory analysis of the data revealed that measures of both distance and correlation of nuclear features could be useful in both clustering and classifying individual samples. All of the models developed performed better than random sampling.

8. Appendix - Environment

```
# Print system information
print("Operating System:")
```

```
##                _
## platform       x86_64-w64-mingw32
## arch           x86_64
## os             mingw32
## system         x86_64, mingw32
## status
## major          4
## minor          0.3
## year           2020
## month          10
## day            10
## svn rev        79318
## language       R
## version.string R version 4.0.3 (2020-10-10)
## nickname       Bunny-Wunnies Freak Out
```