

Breast Cancer Prediction

AMOD-5250H Data Analytics With R

Krunal Bharatbhai Parmar- 0818024

Contents

1	Overview	1
1.1	Introduction	1
1.2	Aim of the project	2
1.3	Dataset	3
2	Methods and Analysis	4
2.1	Data Analysis	4
2.1.1	Modelling	12
2.1.2	Naive Bayes Model	19
2.1.3	Logistic Regression Model	21
2.1.4	Random Forest Model	23
2.1.5	K Nearest Neighbor (KNN) Model	25
2.1.6	Neural Network with PCA Model	27
2.1.7	Neural Network with LDA Model	29
3	Results	30
4	Discussion	33
5	Conclusion	35
6	References	36

Chapter 1

Overview

Breast cancer is one of the most frequent diseases in women globally, accounting for the vast majority of new cancer cases and cancer-related deaths, making it a major public health concern in today's society. Classification and data mining technologies are excellent ways to categorize data. Especially in the medical industry, where such procedures are commonly employed for diagnosis and decision-making.

The provided dataset will be prepared and set up. An exploratory data analysis is performed in order to construct a machine learning algorithm capable of predicting whether a breast cancer cell is benign or malignant until a final model is established. The results will be discussed. Finally, the report will include some final observations.

1.1 Introduction

Cancer that affects women most frequently is breast cancer. Upon early detection and treatment, the patient's 5-year survival rate is 99%; however, that figure plummets to 29% as the disease progresses into a remote stage. Consequently, early detection of "positive cancer cells" is critical; therefore, I utilized R-Studio to analyze the 569 breast cancer cell data provided by the University of Wisconsin. By employing this software, I was able to visually represent the interconnections among ten distinct cellular attributes and investigate the nature of the correlation between radius and cancer cells that are positive by utilizing graphs. Also, I created a predictive model that can detect positive cancer cells based on logistic regression and training original 569 data. Through this research, I derived effective breast cancer predicting model and thoroughly explored big data analysis method and data-mining on R-Studio.

A neoplasm is an abnormal mass of tissue, the growth of which exceeds and is uncoordinated with that of the normal tissues, and persists in the same excessive manner after cessation of the stimulus which evoked the change. Cancer can originate almost anywhere in the human body, which is made up of 37.200 billion cells. As these tumors develop, some cancer cells can break off and travel to distant locations in the body through the blood or the lymph system and form new tumors far from the original one. Unlike malignant tumors, benign tumors do not extend into, or invade, adjacent tissues. Breast cancer refers to a pathology in which a tumor develops in the breast tissue. Breast cancer is amongst the most prevalent form of cancer in both sexes since 1975 and causes around 411,000 annual fatalities worldwide. It is predicted that the incidence for worldwide cancer will continue to increase, with 23,6 million new cancer cases each year by 2030, corresponding to 68% more cases in comparison to 2012.

Mammography is the most prevalent mass screening instrument for an early detection of breast malignancies because of its sensitivity in recognising breast masses. After detection of suspicious breast masses, a biopsy test procedure would be carried out, such as Fine Needle Aspirates (FNA), that is the method this report focus on. This technique has been showed to be safe, cost-effective, precise and quick. A small drop of viscous fluid is aspirated from the breast masses to be analysed under the microscope. Then, a small region of the breast mass cells is photographed in a grey scale image and further analysed using an image analysis program

‘Xcyt’. This program employs a curve-fitting to determine the boundaries of the nuclei from initial points manually set near these margins by a mouse.

The boundaries of the visible cell nuclei were manually placed with a cursor (red spots), ‘Xcyt’ program will after define the nuclei (red circle). The interactive diagnosis process takes about 5 minutes per sample.

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

Diagnosis in an early stage is essential to the facilitate the subsequent clinical management of patients and increase the survival rate of breast cancer patients.

The main models used and evaluated will be supervised learning models (algorithms that learn from labelled data), which are most used in these types of data analysis.

The utilization of data science and machine learning approaches in medical disciplines proves to be prolific as such approaches may be considered of great assistance in the decision making process of medical practitioners. With a lamentable increasing trend of breast cancer cases, comes also a great deal of data which is of significant use in furthering clinical and medical research, and much more to the application of data science and machine learning in the aforementioned domain.

1.2 Aim of the project

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 disclose patterns in the dataset and produce a more robust analysis. As previously said, the optimal model will be selected following the resulting accuracy, sensitivity, and f1 score, among 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 useful to ascertain whether a given sample appears to be Benign (“B”) or Malignant (“M”).

The machine learning models that we will applicate in this report attempt to construct a classifier that provides a high accuracy level combined with a low rate of false-negatives (high sensitivity).

1.3 Dataset

Features are computed from a digitized image of a fine needle aspirate (FNA) of a breast mass. Separating plane described was obtained using Multisurface Method-Tree (MSM-T) [K. P. Bennett, “Decision Tree Construction Via Linear Programming.” Proceedings of the 4th Midwest Artificial Intelligence and Cognitive Science Society, pp. 97-101, 1992], a classification method which uses linear programming to construct a decision tree. Relevant features were selected using an exhaustive search in the space of 1-4 features and 1-3 separating planes.

The present report addresses the Breast Cancer Wisconsin (Diagnostic) DataSet (<https://www.kaggle.com/uciml/breast-cancer-wisconsin-data/version/2>) created by Dr. William H. Wolberg, physician at the University Of Wisconsin Hospital at Madison, Wisconsin, USA. The data used for this research was obtained in 1993 by the University of Wisconsin and it is comprised by the biopsy result of 569 patients in Wisconsin Hospital.

- [Wisconsin Breast Cancer Diagnostic Dataset]- <https://www.kaggle.com/uciml/breast-cancer-wisconsin-data/version/2>

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

- Attribute Information:
 1. ID number
 2. Diagnosis (M = malignant, B = benign)
- Ten features were computed for each cell nucleus:
 1. radius: mean of distances from center to points on the perimeter
 2. texture: standard deviation of grey-scale values
 3. perimeter
 4. area: Number of pixels inside contour + $\frac{1}{2}$ for pixels on perimeter
 5. smoothness: local variation in radius lengths), , t
 6. compactness: $\text{perimeter}^2 / \text{area} - 1.0$; this dimensionless number is at a minimum with a circular disk and increases with the irregularity of the boundary, but this measure also increases for elongated cell nuclei, which is not indicative of malignancy
 7. concavity: severity of concave portions of the contour
 8. concave points: number of concave portions of the contour
 9. symmetry
 10. fractal dimension: “coastline approximation” - 1; a higher value corresponds a less regular contour and thus to a higher probability of malignancy

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

The column 33 is invalid.

```
data$diagnosis <- as.factor(data$diagnosis)
# the 33 column is invalid
data[,33] <- NULL
```

Chapter 2

Methods and Analysis

2.1 Data Analysis

By observing our dataset, we found that it contains 569 observations with 32 variables.

```
str(data)
```

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

```
$ fractal_dimension_worst: num 0.1189 0.089 0.0876 0.173 0.0768 ...
```

```
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)
```

```

      id      diagnosis radius_mean texture_mean
Min.   :    8670      B:357      Min.   : 6.981      Min.   : 9.71
1st Qu.:   869218      M:212      1st Qu.:11.700      1st Qu.:16.17
Median :    906024                      Median :13.370      Median :18.84
Mean   :   30371831                      Mean   :14.127      Mean   :19.29
3rd Qu.:   8813129                      3rd Qu.:15.780      3rd Qu.:21.80
Max.   :  911320502                      Max.   :28.110      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.: 75.17      1st Qu.: 420.3      1st Qu.:0.08637      1st Qu.:0.06492
Median : 86.24      Median : 551.1      Median :0.09587      Median :0.09263
Mean   : 91.97      Mean   : 654.9      Mean   :0.09636      Mean   :0.10434
3rd Qu.:104.10      3rd Qu.: 782.7      3rd Qu.:0.10530      3rd Qu.:0.13040
Max.   :188.50      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.:0.02956      1st Qu.:0.02031      1st Qu.:0.1619      1st Qu.:0.05770
Median :0.06154      Median :0.03350      Median :0.1792      Median :0.06154
Mean   :0.08880      Mean   :0.04892      Mean   :0.1812      Mean   :0.06280
3rd Qu.:0.13070      3rd Qu.:0.07400      3rd Qu.:0.1957      3rd Qu.:0.06612
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.:0.2324      1st Qu.:0.8339      1st Qu.: 1.606      1st Qu.:17.850
Median :0.3242      Median :1.1080      Median : 2.287      Median :24.530
Mean   :0.4052      Mean   :1.2169      Mean   : 2.866      Mean   :40.337
3rd Qu.:0.4789      3rd Qu.:1.4740      3rd Qu.: 3.357      3rd Qu.:45.190
Max.   :2.8730      Max.   :4.8850      Max.   :21.980      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.:0.005169      1st Qu.:0.013080      1st Qu.:0.01509      1st Qu.:0.007638
Median :0.006380      Median :0.020450      Median :0.02589      Median :0.010930
Mean   :0.007041      Mean   :0.025478      Mean   :0.03189      Mean   :0.011796
3rd Qu.:0.008146      3rd Qu.:0.032450      3rd Qu.:0.04205      3rd Qu.:0.014710
Max.   :0.031130      Max.   :0.135400      Max.   :0.39600      Max.   :0.052790

symmetry_se fractal_dimension_se radius_worst texture_worst
Min.   :0.007882      Min.   :0.0008948      Min.   : 7.93      Min.   :12.02
1st Qu.:0.015160      1st Qu.:0.0022480      1st Qu.:13.01      1st Qu.:21.08
Median :0.018730      Median :0.0031870      Median :14.97      Median :25.41
Mean   :0.020542      Mean   :0.0037949      Mean   :16.27      Mean   :25.68
3rd Qu.:0.023480      3rd Qu.:0.0045580      3rd Qu.:18.79      3rd Qu.:29.72
Max.   :0.078950      Max.   :0.0298400      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.: 84.11      1st Qu.: 515.3      1st Qu.:0.11660      1st Qu.:0.14720
Median : 97.66      Median : 686.5      Median :0.13130      Median :0.21190
Mean   :107.26      Mean   : 880.6      Mean   :0.13237      Mean   :0.25427
3rd Qu.:125.40      3rd Qu.:1084.0      3rd Qu.:0.14600      3rd Qu.:0.33910
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.:0.1145      1st Qu.:0.06493      1st Qu.:0.2504      1st Qu.:0.07146

```


Median :0.2267	Median :0.09993	Median :0.2822	Median :0.08004
Mean :0.2722	Mean :0.11461	Mean :0.2901	Mean :0.08395
3rd Qu.:0.3829	3rd Qu.:0.16140	3rd Qu.:0.3179	3rd Qu.:0.09208
Max. :1.2520	Max. :0.29100	Max. :0.6638	Max. :0.20750

We have to check if the dataset has any missing value:

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

```
$id
```

```
[1] 0
```

```
$diagnosis
```

```
[1] 0
```

```
$radius_mean
```

```
[1] 0
```

```
$texture_mean
```

```
[1] 0
```

```
$perimeter_mean
```

```
[1] 0
```

```
$area_mean
```

```
[1] 0
```

```
$smoothness_mean
```

```
[1] 0
```

```
$compactness_mean
```

```
[1] 0
```

```
$concavity_mean
```

```
[1] 0
```

```
$concave.points_mean
```

```
[1] 0
```

```
$symmetry_mean
```

```
[1] 0
```

```
$fractal_dimension_mean
```

```
[1] 0
```

```
$radius_se
```

```
[1] 0
```

```
$texture_se
```

```
[1] 0
```

```
$perimeter_se
```

```
[1] 0
```

```
$area_se
```

```
[1] 0

$smoothness_se
[1] 0

$compactness_se
[1] 0

$concavity_se
[1] 0

$concave.points_se
[1] 0

$symmetry_se
[1] 0

$fractal_dimension_se
[1] 0

$radius_worst
[1] 0

$texture_worst
[1] 0

$perimeter_worst
[1] 0

$area_worst
[1] 0

$smoothness_worst
[1] 0

$compactness_worst
[1] 0

$concavity_worst
[1] 0

$concave.points_worst
[1] 0

$symmetry_worst
[1] 0

$fractal_dimension_worst
[1] 0
```

It results that there aren't NA values.

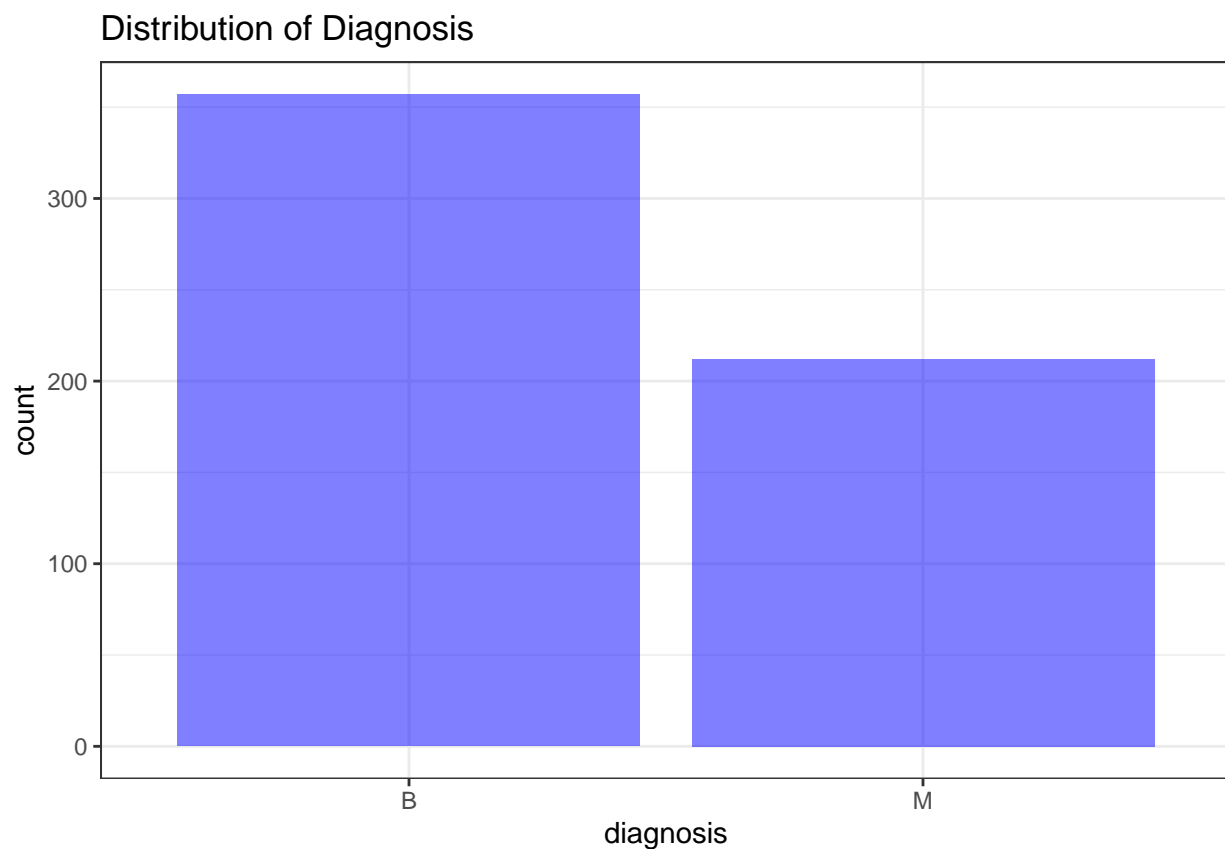
By analysing the the dataset we discover that it is a bit unbalanced in its proportions:

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

```
      B      M  
0.6274165 0.3725835
```

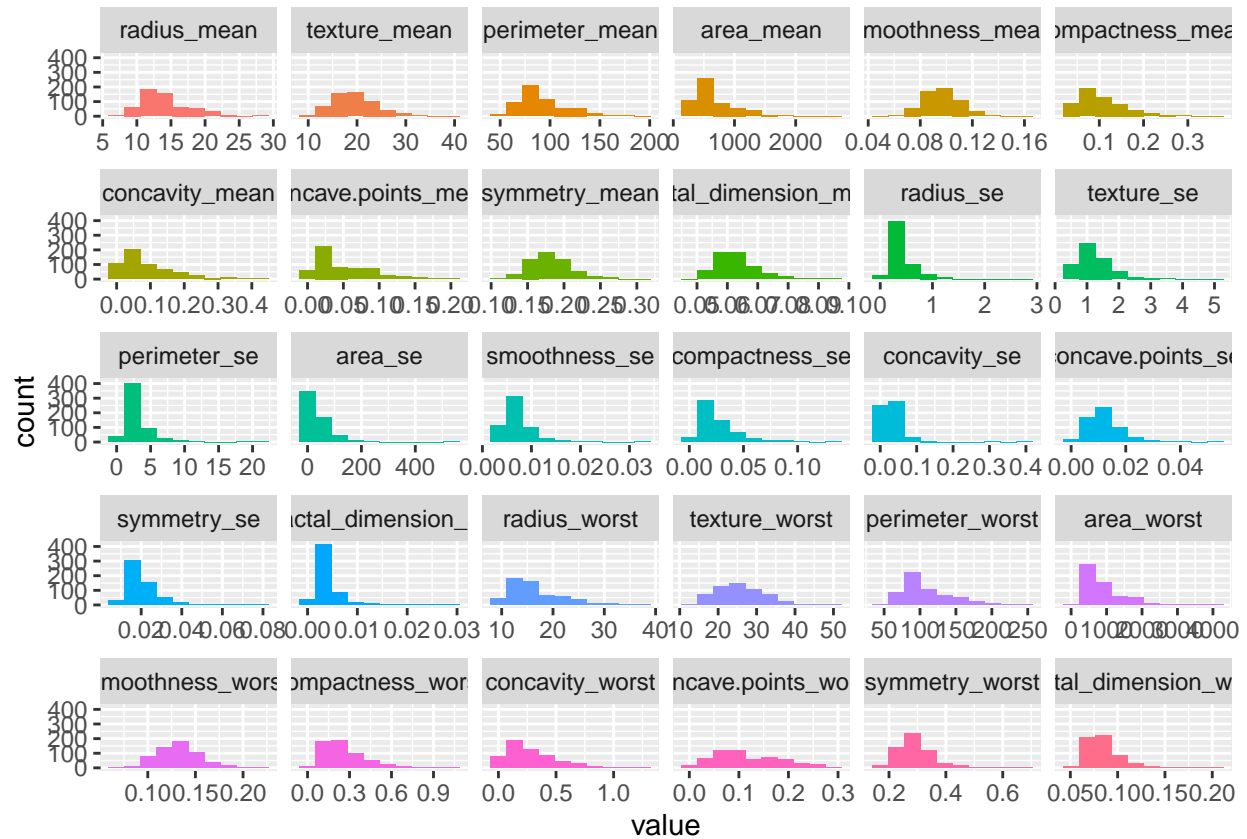
Also the plot of proportions confirms that the target variable is slightly unbalanced.

```
options(repr.plot.width=4, repr.plot.height=4)  
ggplot(data, aes(x=diagnosis))+  
  geom_bar(fill="blue",alpha=0.5)+theme_bw()+  
  labs(title="Distribution of Diagnosis")
```



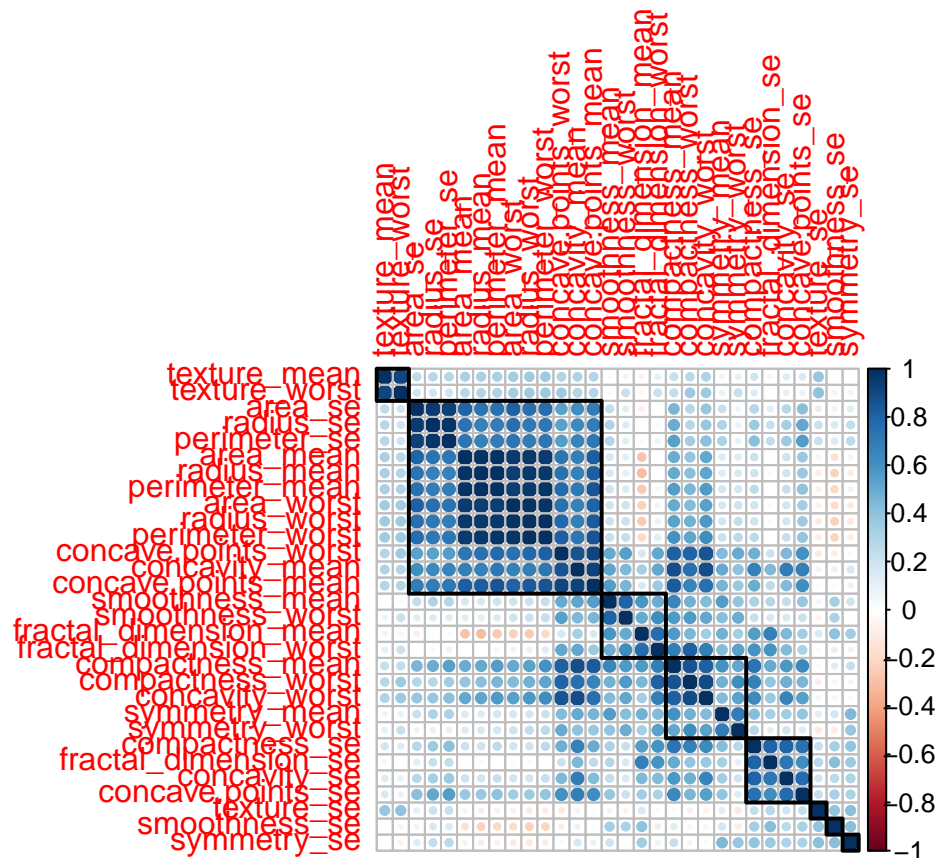
The most variables of the dataset are normally distributed as show with the below plot:

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



Now we have to check if there is any correlation between variables as machine learning algorithms assume that the predictor variables are independent from each other.

```
correlationMatrix <- cor(data[,3:ncol(data)])
corrplot(correlationMatrix, order = "hclust", tl.cex = 1, addrect = 8)
```



As shown by this plot, many variables are highly correlated with each others. Many methods perform better if highly correlated attributes are removed. The Caret R package provides the `findCorrelation` which will analyze a correlation matrix of your data's attributes report on attributes that can be removed. Because of much correlation some machine learning models could fail.

```
# find attributes that are highly correlated (ideally >0.90)
highlyCorrelated <- findCorrelation(correlationMatrix, cutoff=0.9)
# print indexes of highly correlated attributes
print(highlyCorrelated)
```

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

Selecting the right features in our data can mean the difference between mediocre performance with long training times and great performance with short training times.

```
# Remove correlated variables
data2 <- data %>%select(-highlyCorrelated)
# number of columns after removing correlated variables
ncol(data2)
```

```
[1] 22
```

The new dataset has less 10 variables.

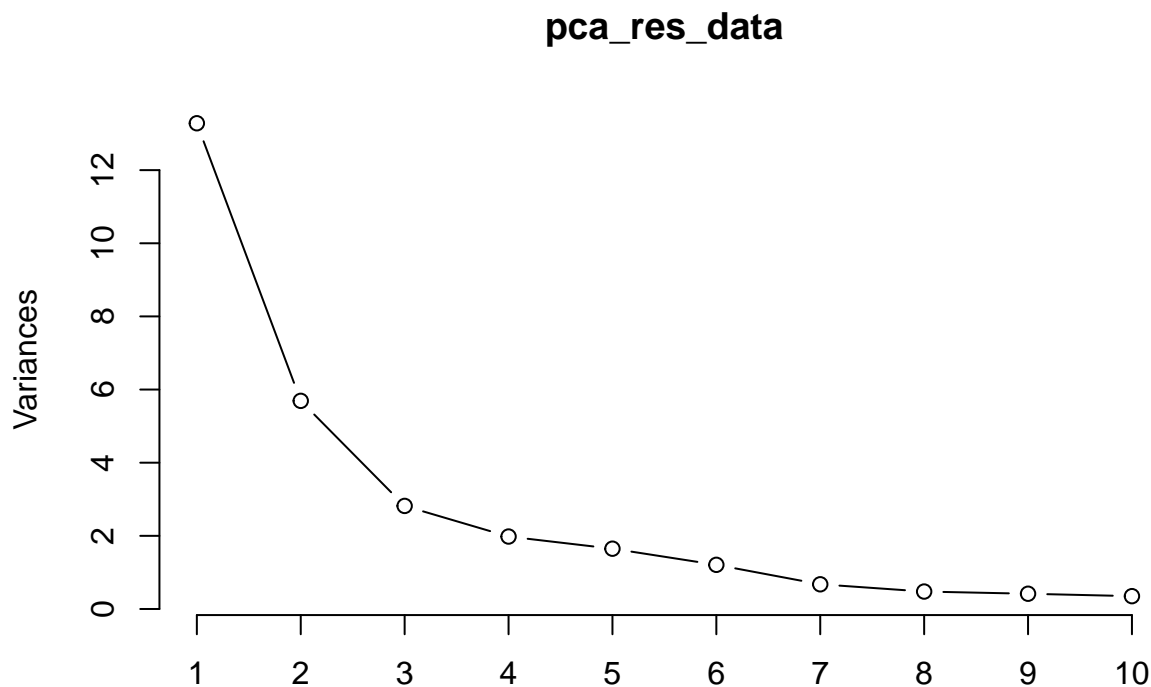
Modelling Approach

2.1.1 Modelling

Principal Component Analysis (PCA).

To avoid redundancy and relevancy, we used the function ‘prncomp’ to calculate the Principal Component Analysis (PCA) and select the appropriate components to prevent correlated variables that can be detrimental to our clustering analysis. One of the common difficulties in analysis of complex data stems from a large number of variables, which requires a large quantity of memory and computation capacity. This is where PCA comes in. It is a technique to reduce the dimension of the feature space by feature extraction. The main concept of PCA is to reduce the dimensionality of a data set consisting of many variables correlated with each other, either severely or lightly, while retaining the variation present in the dataset, up to the utmost 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 diminish as we progress down in the order.

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



```
summary(pca_res_data)
```

Importance of components:

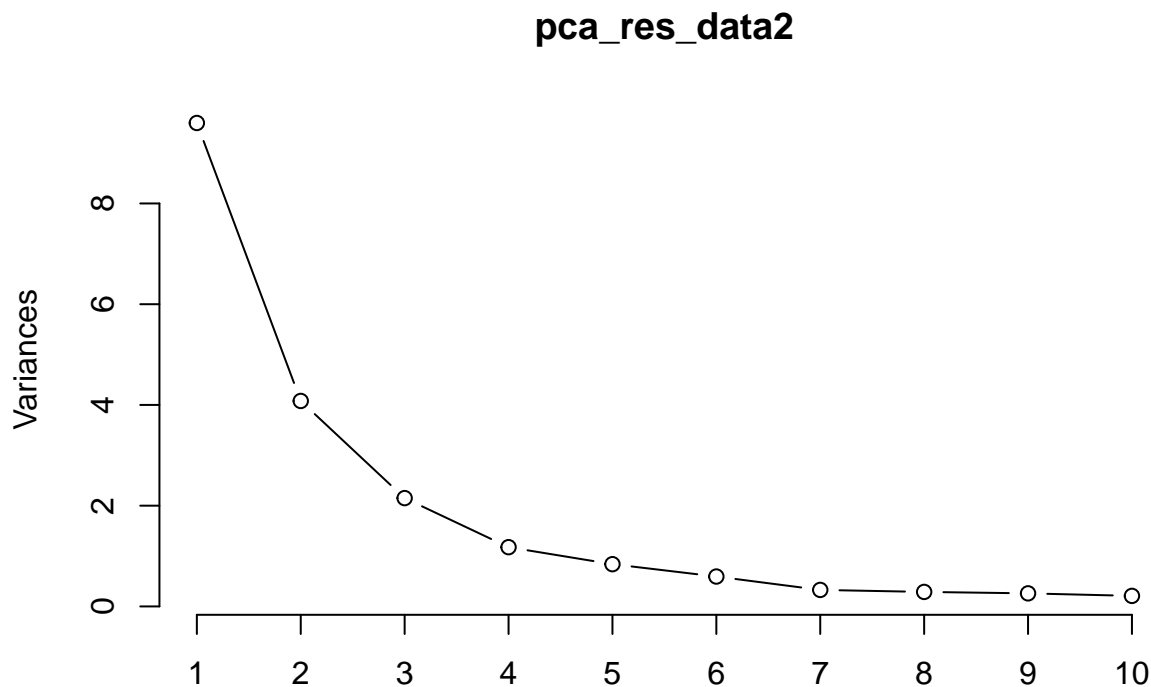
	PC1	PC2	PC3	PC4	PC5	PC6	PC7
Standard deviation	3.6444	2.3857	1.67867	1.40735	1.28403	1.09880	0.82172
Proportion of Variance	0.4427	0.1897	0.09393	0.06602	0.05496	0.04025	0.02251
Cumulative Proportion	0.4427	0.6324	0.72636	0.79239	0.84734	0.88759	0.91010

	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 above table, the two first components explains the 0.6324 of the variance. We need 10 principal components to explain more than 0.95 of the variance and 17 to explain more than 0.99.

```
pca_res_data2 <- prcomp(data2[,3:ncol(data2)], center = TRUE, scale = TRUE)
plot(pca_res_data2, type="l")
```



```
summary(pca_res_data2)
```

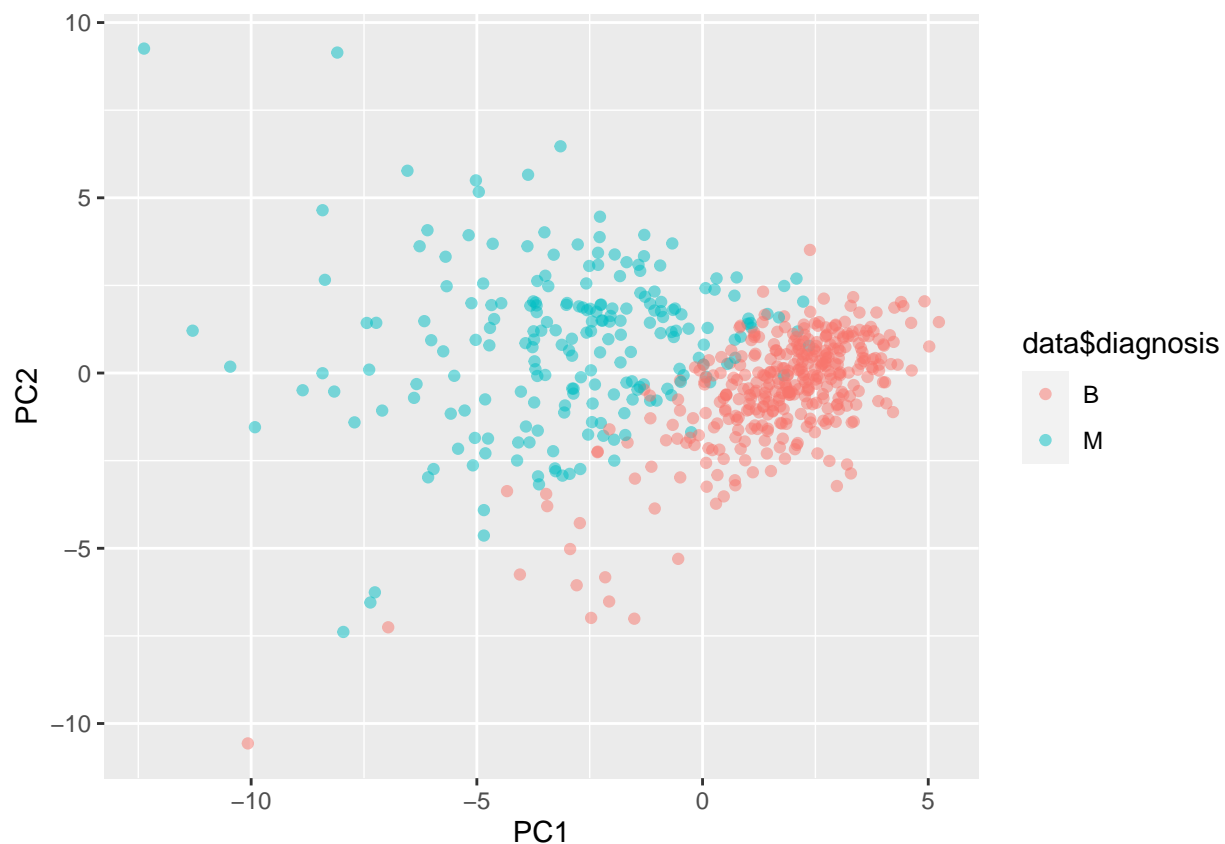
Importance of components:

	PC1	PC2	PC3	PC4	PC5	PC6	PC7
Standard deviation	3.0980	2.0196	1.4663	1.0845	0.91561	0.77019	0.57227
Proportion of Variance	0.4799	0.2039	0.1075	0.0588	0.04192	0.02966	0.01637

Cumulative Proportion	0.4799	0.6838	0.7913	0.8501	0.89205	0.92171	0.93808
	PC8	PC9	PC10	PC11	PC12	PC13	PC14
Standard deviation	0.53641	0.50898	0.45726	0.36641	0.31778	0.28802	0.21369
Proportion of Variance	0.01439	0.01295	0.01045	0.00671	0.00505	0.00415	0.00228
Cumulative Proportion	0.95247	0.96542	0.97588	0.98259	0.98764	0.99179	0.99407
	PC15	PC16	PC17	PC18	PC19	PC20	
Standard deviation	0.1846	0.15579	0.15393	0.14782	0.09636	0.07375	
Proportion of Variance	0.0017	0.00121	0.00118	0.00109	0.00046	0.00027	
Cumulative Proportion	0.9958	0.99699	0.99817	0.99926	0.99973	1.00000	

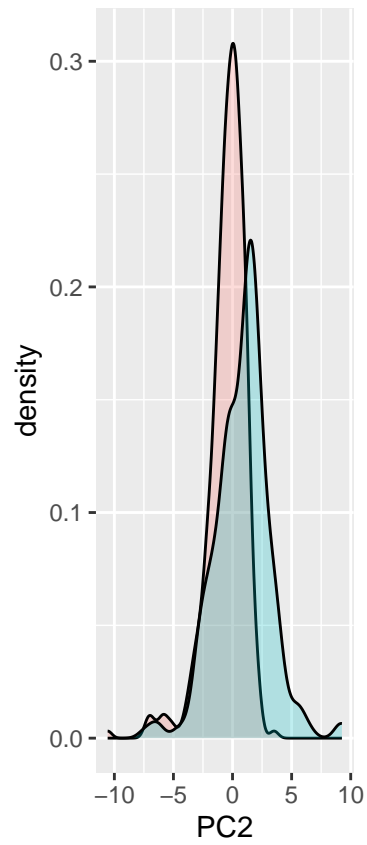
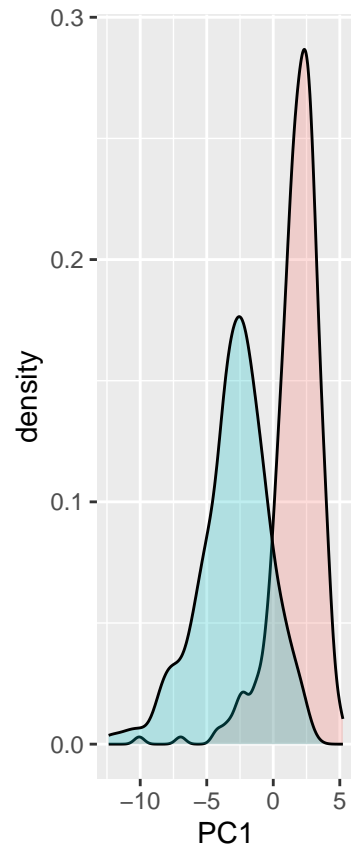
The above table shows that 95% of the variance is explained with 8 PC's in the transformed dataset data2.

```
pca_df <- as.data.frame(pca_res_data2$x)
ggplot(pca_df, aes(x=PC1, y=PC2, col=data$diagnosis)) + geom_point(alpha=0.5)
```



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

```
g_pc1 <- ggplot(pca_df, aes(x=PC1, fill=data$diagnosis)) + geom_density(alpha=0.25)
g_pc2 <- ggplot(pca_df, aes(x=PC2, fill=data$diagnosis)) + geom_density(alpha=0.25)
grid.arrange(g_pc1, g_pc2, ncol=2)
```

Linear Discriminant Analysis (LDA)

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

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

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:

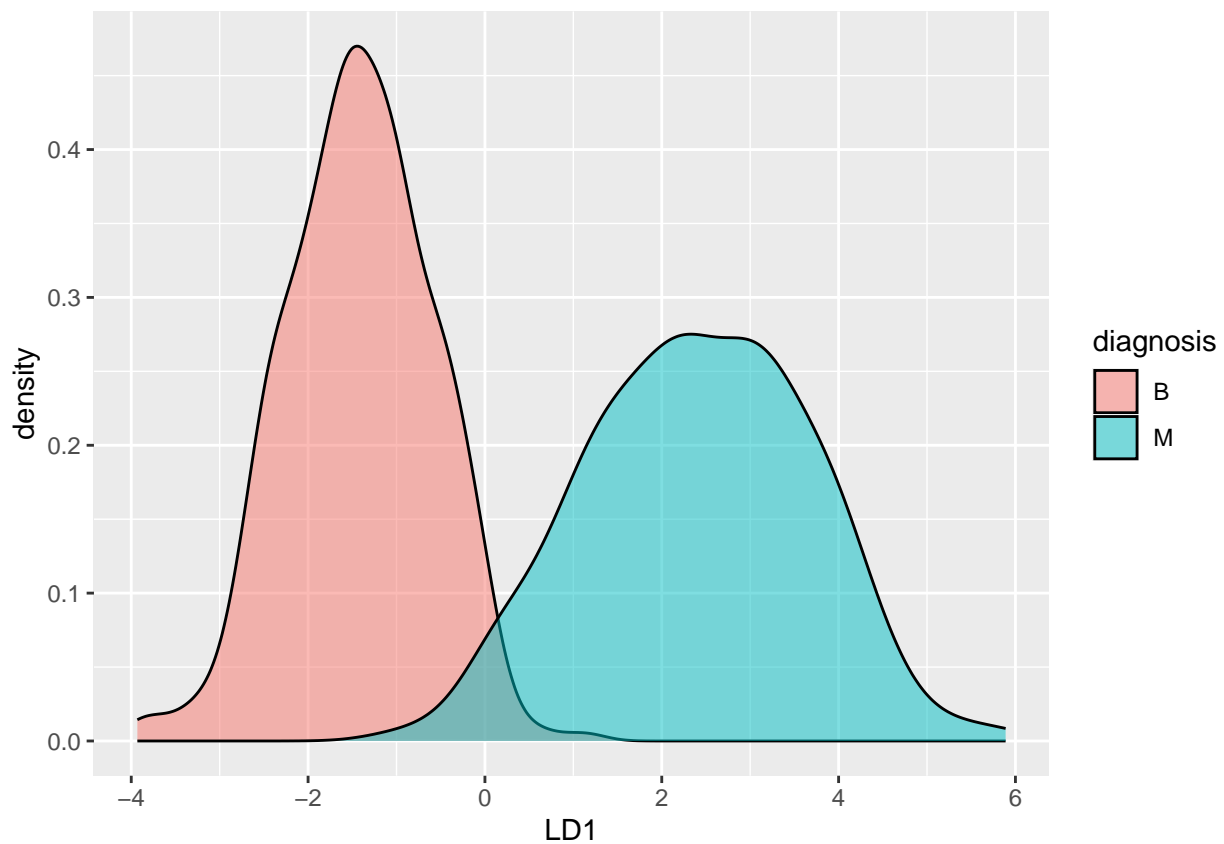
	LD1
id	-2.512117e-10
radius_mean	-1.080876e+00
texture_mean	2.338408e-02
perimeter_mean	1.172707e-01
area_mean	1.595690e-03
smoothness_mean	5.251575e-01
compactness_mean	-2.094197e+01
concavity_mean	6.955923e+00
concave.points_mean	1.047567e+01
symmetry_mean	4.938898e-01
fractal_dimension_mean	-5.937663e-02
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
```

```
#Data frame of the LDA for visualization purposes
```

```
lda_df_predict <- predict(lda_res_data, data)$x %>% as.data.frame() %>% cbind(diagnosis=data$diagnosis)
```

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



2.1.2 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. 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.

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

prediction_naiveb <- predict(model_naiveb, test_data)
confusionmatrix_naiveb <- confusionMatrix(prediction_naiveb, test_data$diagnosis, positive = "M")
confusionmatrix_naiveb
```

Confusion Matrix and Statistics

	Reference	
Prediction	B	M
B	66	3
M	5	39

Accuracy : 0.9292
 95% CI : (0.8653, 0.9689)
 No Information Rate : 0.6283
 P-Value [Acc > NIR] : 1.372e-13

 Kappa : 0.8499

 Mcnemar's Test P-Value : 0.7237

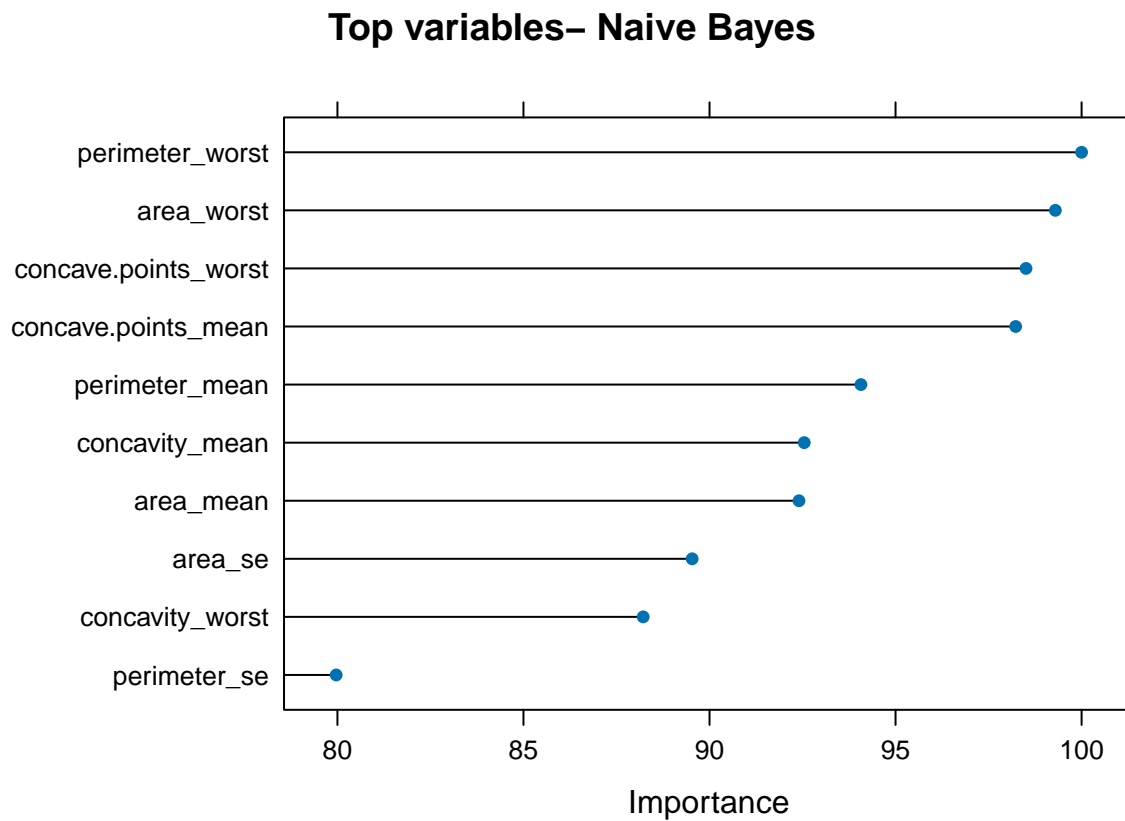
 Sensitivity : 0.9286
 Specificity : 0.9296
 Pos Pred Value : 0.8864
 Neg Pred Value : 0.9565
 Prevalence : 0.3717
 Detection Rate : 0.3451
 Detection Prevalence : 0.3894
 Balanced Accuracy : 0.9291

 'Positive' Class : M

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.

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

```
plot(varImp(model_naiveb), top=10, main="Top variables- Naive Bayes")
```



2.1.3 Logistic Regression Model

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

```
model_logreg<- train(diagnosis ~., data = train_data, method = "glm",
                     metric = "ROC",

                     preProcess = c("scale", "center"), # in order to normalize the data
                     trControl= fitControl)
prediction_logreg<- predict(model_logreg, test_data)

# Check results
confusionmatrix_logreg <- confusionMatrix(prediction_logreg, test_data$diagnosis, positive = "M")
confusionmatrix_logreg
```

Confusion Matrix and Statistics

```

      Reference
Prediction B  M
      B 67  0
      M  4 42

      Accuracy : 0.9646
      95% CI : (0.9118, 0.9903)
      No Information Rate : 0.6283
      P-Value [Acc > NIR] : <2e-16

      Kappa : 0.9257

      McNemar's Test P-Value : 0.1336

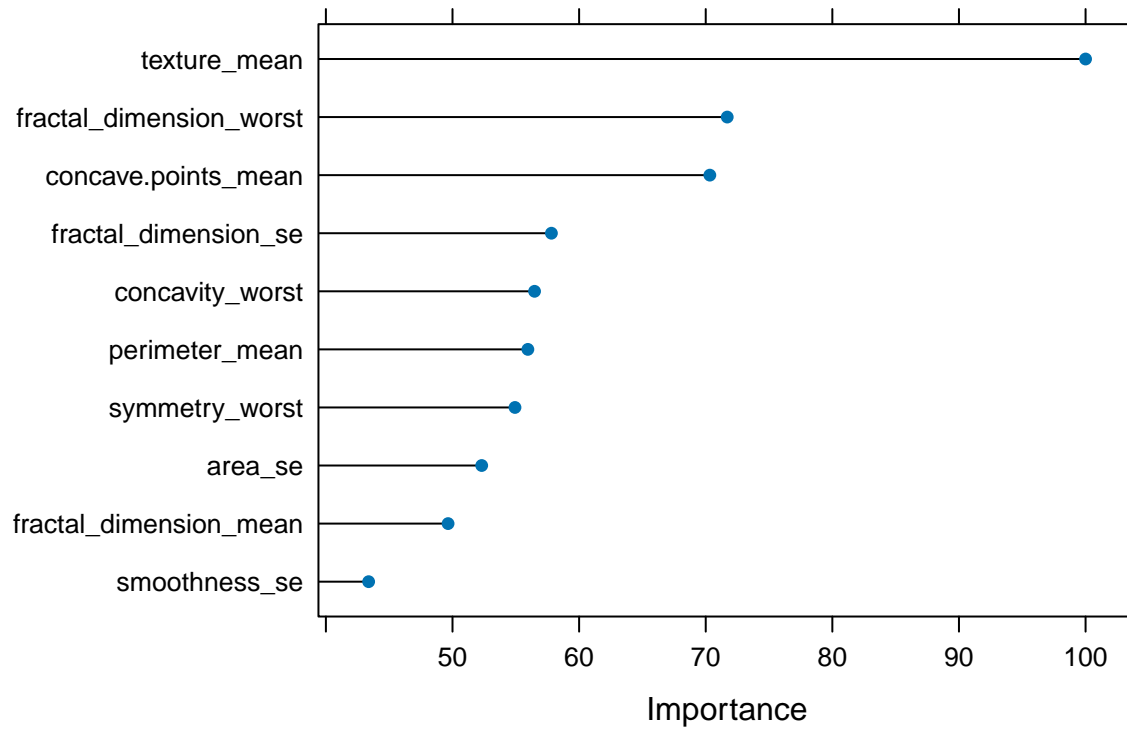
      Sensitivity : 1.0000
      Specificity : 0.9437
      Pos Pred Value : 0.9130
      Neg Pred Value : 1.0000
      Prevalence : 0.3717
      Detection Rate : 0.3717
      Detection Prevalence : 0.4071
      Balanced Accuracy : 0.9718

      'Positive' Class : M
```

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

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

Top variables – Log Regr



2.1.4 Random Forest Model

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 split 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 it's 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.

```
model_randomforest <- train(diagnosis~.,
                             train_data,
                             method="rf", #also recommended ranger, because it is a lot faster than ori
                             metric="ROC",
                             #tuneLength=10,
                             #tuneGrid = expand.grid(mtry = c(2, 3, 6)),
                             preProcess = c('center', 'scale'),
                             trControl=fitControl)

prediction_randomforest <- predict(model_randomforest, test_data)

#Check results
confusionmatrix_randomforest <- confusionMatrix(prediction_randomforest, test_data$diagnosis, positive = "M")
confusionmatrix_randomforest
```

Confusion Matrix and Statistics

	Reference	
Prediction	B	M
B	70	1
M	1	41

Accuracy : 0.9823
 95% CI : (0.9375, 0.9978)
 No Information Rate : 0.6283
 P-Value [Acc > NIR] : <2e-16

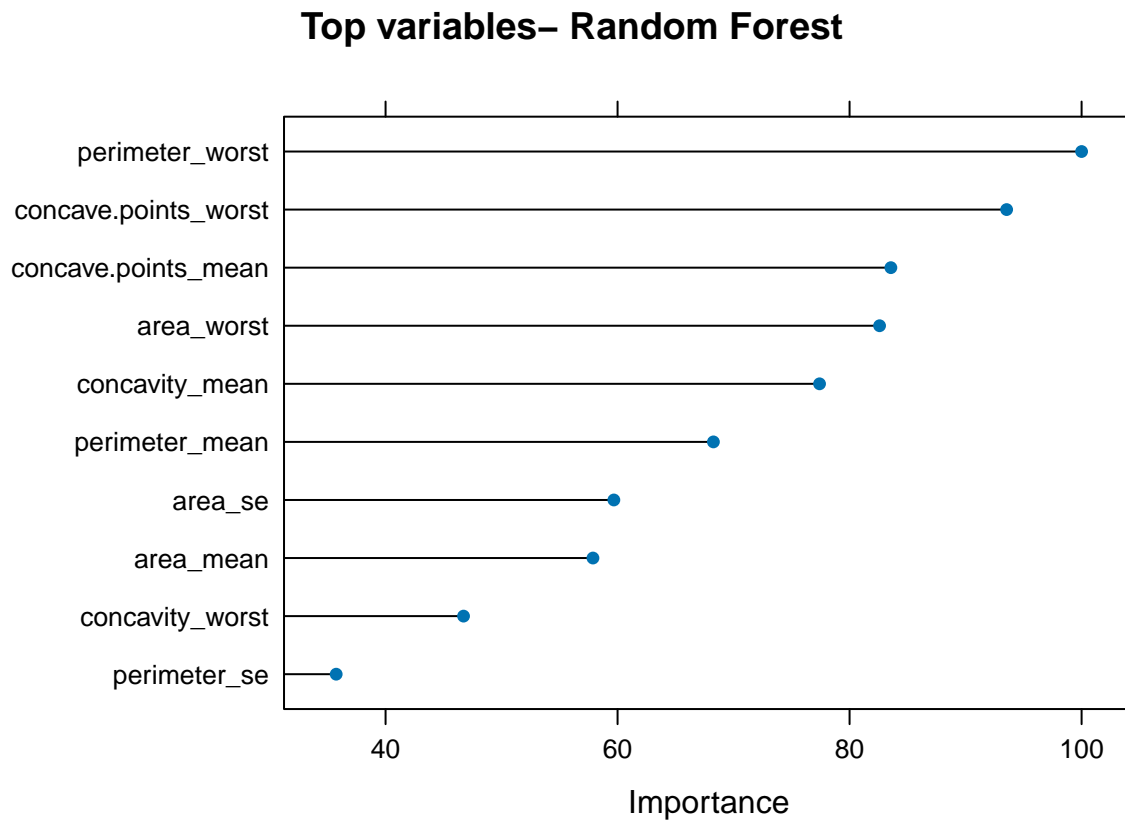
Kappa : 0.9621

Mcnemar's Test P-Value : 1

Sensitivity : 0.9762
 Specificity : 0.9859
 Pos Pred Value : 0.9762
 Neg Pred Value : 0.9859
 Prevalence : 0.3717
 Detection Rate : 0.3628
 Detection Prevalence : 0.3717
 Balanced Accuracy : 0.9811

'Positive' Class : M

```
plot(varImp(model_randomforest), top=10, main="Top variables- Random Forest")
```



2.1.5 K Nearest Neighbor (KNN) Model

KNN (K-Nearest Neighbors) is one of many (supervised learning) algorithms used in data mining and machine learning, it's a classifier algorithm where the learning is based "how similar" is a data from other. K nearest neighbors is a simple algorithm that stores all available cases and classifies new cases based on a similarity measure (e.g., distance functions).

```
model_knn <- train(diagnosis~.,
                  train_data,
                  method="knn",
                  metric="ROC",
                  preProcess = c('center', 'scale'),
                  tuneLength=10, #The tuneLength parameter tells the algorithm to try different default
                  #In this case we used 10 default values
                  trControl=fitControl)

prediction_knn <- predict(model_knn, test_data)
confusionmatrix_knn <- confusionMatrix(prediction_knn, test_data$diagnosis, positive = "M")
confusionmatrix_knn
```

Confusion Matrix and Statistics

	Reference	
Prediction	B	M
B	71	3
M	0	39

Accuracy : 0.9735
 95% CI : (0.9244, 0.9945)
 No Information Rate : 0.6283
 P-Value [Acc > NIR] : <2e-16

 Kappa : 0.9423

 McNemar's Test P-Value : 0.2482

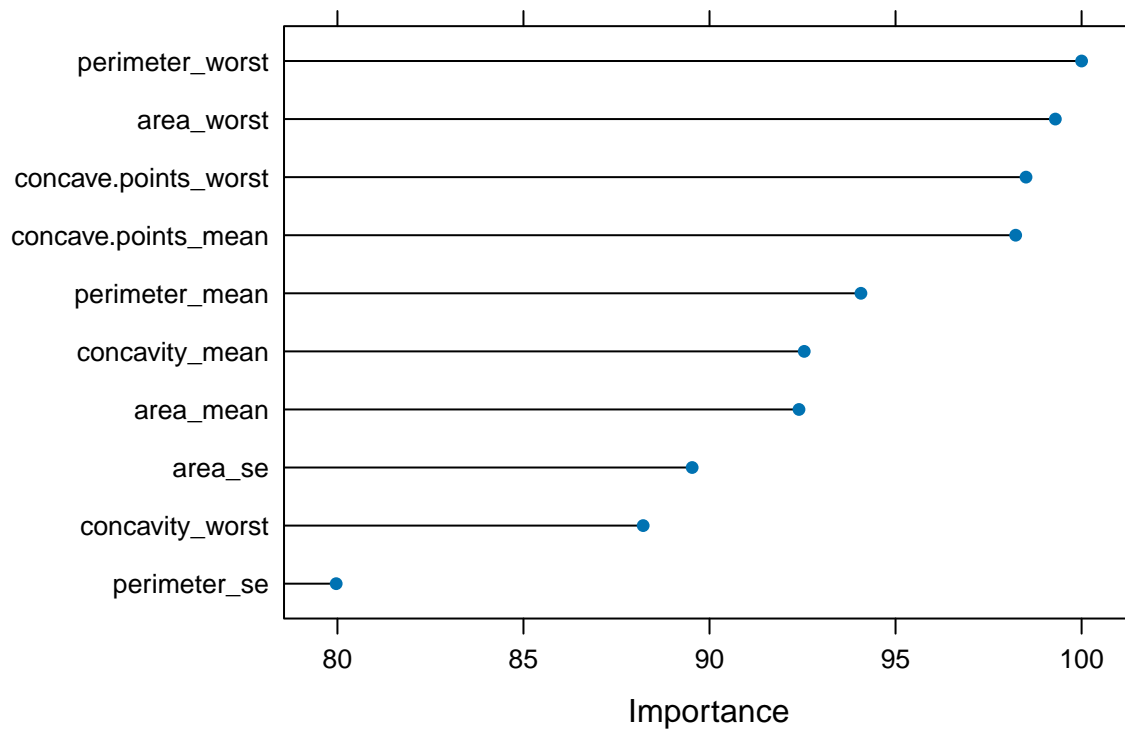
 Sensitivity : 0.9286
 Specificity : 1.0000
 Pos Pred Value : 1.0000
 Neg Pred Value : 0.9595
 Prevalence : 0.3717
 Detection Rate : 0.3451
 Detection Prevalence : 0.3451
 Balanced Accuracy : 0.9643

 'Positive' Class : M

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

```
plot(varImp(model_knn), top=10, main="Top variables - KNN")
```

Top variables – KNN



2.1.6 Neural Network with PCA Model

Artificial Neural Networks (NN) are a types of mathematical algorithms originating in the simulation of networks of biological neurons. An artificial Neural Network consists of nodes (called neurons) and edges (called synapses). Input data is transmitted through the weighted synapses to the neurons where calculations are processed and then either sent to further neurons or represent the output.

Neural Networks take in the weights of connections between neurons . The weights are balanced, learning data point in the wake of learning data point . When all weights are trained, the neural network can be utilized to predict the class or a quantity, if there should arise an occurrence of regression of a new input data point. With Neural networks, extremely complex models can be trained and they can be utilized as a kind of black box, without playing out an unpredictable complex feature engineering before training the model. Joined with the “deep approach” even more unpredictable models can be picked up to realize new possibilities.

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

prediction_nnet_pca <- predict(model_nnet_pca, test_data)
confusionmatrix_nnet_pca <- confusionMatrix(prediction_nnet_pca, test_data$diagnosis, positive = "M")
confusionmatrix_nnet_pca
```

Confusion Matrix and Statistics

	Reference	
Prediction	B	M
B	70	0
M	1	42

Accuracy : 0.9912
 95% CI : (0.9517, 0.9998)
 No Information Rate : 0.6283
 P-Value [Acc > NIR] : <2e-16

Kappa : 0.9811

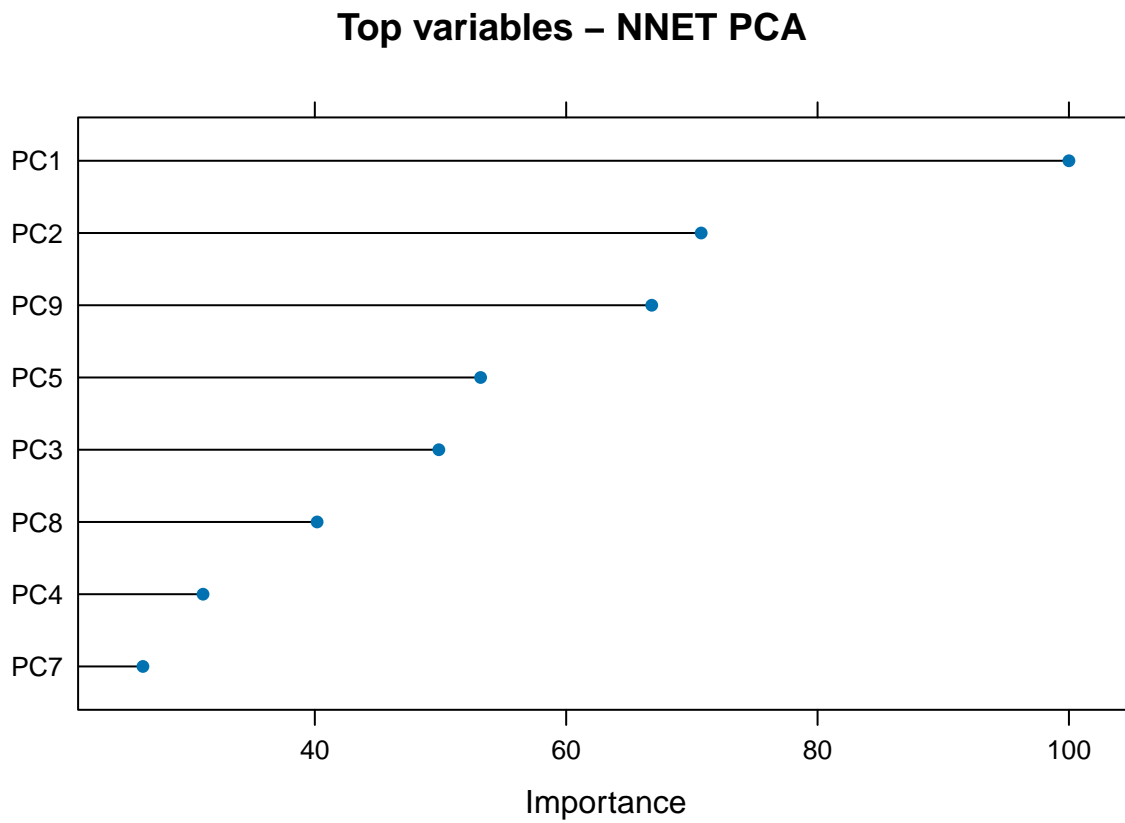
Mcnemar's Test P-Value : 1

Sensitivity : 1.0000
 Specificity : 0.9859
 Pos Pred Value : 0.9767
 Neg Pred Value : 1.0000
 Prevalence : 0.3717
 Detection Rate : 0.3717
 Detection Prevalence : 0.3805
 Balanced Accuracy : 0.9930

'Positive' Class : M

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

```
plot(varImp(model_nnet_pca), top=8, main="Top variables - NNET PCA")
```



2.1.7 Neural Network with LDA Model

We are going to create a training and test set of LDA data created in previous chapters:

```
train_data_lda <- lda_df_predict[data_sampling_index, ]
test_data_lda <- lda_df_predict[-data_sampling_index, ]

model_nnet_lda <- train(diagnosis~.,
                        train_data_lda,
                        method="nnet",
                        metric="ROC",
                        preProcess=c('center', 'scale'),
                        tuneLength=10,
                        trace=FALSE,
                        trControl=fitControl)

prediction_nnet_lda <- predict(model_nnet_lda, test_data_lda)
confusionmatrix_nnet_lda <- confusionMatrix(prediction_nnet_lda, test_data_lda$diagnosis, positive = "M")
confusionmatrix_nnet_lda
```

Confusion Matrix and Statistics

	Reference	
Prediction	B	M
B	70	1
M	1	41

```

Accuracy : 0.9823
 95% CI : (0.9375, 0.9978)
No Information Rate : 0.6283
P-Value [Acc > NIR] : <2e-16
```

```
Kappa : 0.9621
```

```
McNemar's Test P-Value : 1
```

```

Sensitivity : 0.9762
Specificity : 0.9859
Pos Pred Value : 0.9762
Neg Pred Value : 0.9859
Prevalence : 0.3717
Detection Rate : 0.3628
Detection Prevalence : 0.3717
Balanced Accuracy : 0.9811
```

```
'Positive' Class : M
```

Chapter 3

Results

We can now compare and evaluate the results obtained with the above calculations.

```
models_list <- list(Naive_Bayes=model_naiveb,  
                    Logistic_regr=model_logreg,  
                    Random_Forest=model_randomforest,  
                    KNN=model_knn,  
                    Neural_PCA=model_nnet_pca,  
                    Neural_LDA=model_nnet_lda)  
models_results <- resamples(models_list)  
  
summary(models_results)
```

Call:

```
summary.resamples(object = models_results)
```

Models: Naive_Bayes, Logistic_regr, Random_Forest, KNN, Neural_PCA, Neural_LDA
Number of resamples: 15

ROC

	Min.	1st Qu.	Median	Mean	3rd Qu.	Max.	NA's
Naive_Bayes	0.9473684	0.9760766	0.9904306	0.9832270	0.995614	1	0
Logistic_regr	0.8301435	0.9533493	1.0000000	0.9663477	1.000000	1	0
Random_Forest	0.9473684	0.9856459	0.9978070	0.9902844	1.000000	1	0
KNN	0.9451754	0.9873405	0.9976077	0.9908347	1.000000	1	0
Neural_PCA	0.9605263	0.9932217	1.0000000	0.9937267	1.000000	1	0
Neural_LDA	0.9780702	0.9956140	1.0000000	0.9967092	1.000000	1	0

Sens

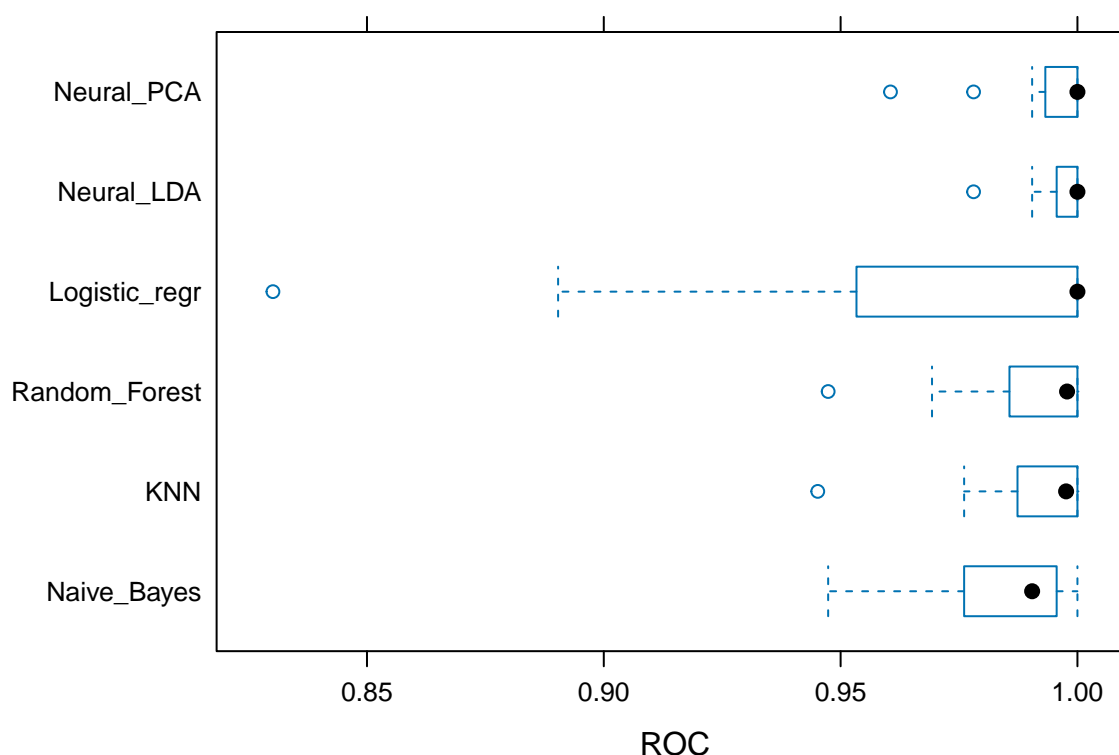
	Min.	1st Qu.	Median	Mean	3rd Qu.	Max.	NA's
Naive_Bayes	0.8421053	0.9473684	0.9473684	0.9512281	1	1	0
Logistic_regr	0.8421053	0.9236842	1.0000000	0.9582456	1	1	0
Random_Forest	0.8421053	0.9473684	1.0000000	0.9719298	1	1	0
KNN	0.8947368	1.0000000	1.0000000	0.9861404	1	1	0
Neural_PCA	0.9473684	0.9486842	1.0000000	0.9826316	1	1	0
Neural_LDA	0.9473684	1.0000000	1.0000000	0.9894737	1	1	0

Spec

	Min.	1st Qu.	Median	Mean	3rd Qu.	Max.	NA's
Naive_Bayes	0.7272727	0.9090909	0.9090909	0.9106061	0.9583333	1	0
Logistic_regr	0.8181818	0.8712121	1.0000000	0.9419192	1.0000000	1	0
Random_Forest	0.8181818	0.8712121	0.9166667	0.9303030	1.0000000	1	0
KNN	0.7272727	0.8712121	0.9166667	0.9136364	1.0000000	1	0
Neural_PCA	0.8181818	0.9090909	0.9166667	0.9414141	1.0000000	1	0
Neural_LDA	0.8181818	0.9128788	1.0000000	0.9585859	1.0000000	1	0

As we can observe from the following plot, two models, Naive_bayes and Logistic_regr have great variability, depending of the processed sample :

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



The Neural Network LDA model achieve a great auc (Area Under the ROC Curve) with some variability. The ROC (Receiver Operating characteristic Curve is a graph showing the performance of a classification model at all classification thresholds) metric measure the auc of the roc curve of each model. This metric is independent of any threshold. Prediction classes are obtained by default with a threshold of 0.5 which could not be the best with an unbalanced dataset like this.

```
confusionmatrix_list <- list(
  Naive_Bayes=confusionmatrix_naiveb,
  Logistic_regr=confusionmatrix_logreg,
  Random_Forest=confusionmatrix_randomforest,
  KNN=confusionmatrix_knn,
  Neural_PCA=confusionmatrix_nnet_pca,
  Neural_LDA=confusionmatrix_nnet_lda)
confusionmatrix_list_results <- sapply(confusionmatrix_list, function(x) x$byClass)
confusionmatrix_list_results %>% knitr::kable()
```

	Naive_Bayes	Logistic_regr	Random_Forest	KNN	Neural_PCA	Neural_LDA
Sensitivity	0.9285714	1.0000000	0.9761905	0.9285714	1.0000000	0.9761905
Specificity	0.9295775	0.9436620	0.9859155	1.0000000	0.9859155	0.9859155
Pos Pred Value	0.8863636	0.9130435	0.9761905	1.0000000	0.9767442	0.9761905
Neg Pred Value	0.9565217	1.0000000	0.9859155	0.9594595	1.0000000	0.9859155
Precision	0.8863636	0.9130435	0.9761905	1.0000000	0.9767442	0.9761905
Recall	0.9285714	1.0000000	0.9761905	0.9285714	1.0000000	0.9761905
F1	0.9069767	0.9545455	0.9761905	0.9629630	0.9882353	0.9761905
Prevalence	0.3716814	0.3716814	0.3716814	0.3716814	0.3716814	0.3716814
Detection Rate	0.3451327	0.3716814	0.3628319	0.3451327	0.3716814	0.3628319
Detection Prevalence	0.3893805	0.4070796	0.3716814	0.3451327	0.3805310	0.3716814
Balanced Accuracy	0.9290744	0.9718310	0.9810530	0.9642857	0.9929577	0.9810530

The analysis resulted in several key findings regarding the performance of various machine learning models in classifying breast cancer cells as benign or malignant. Here's a breakdown:

Model Performance:

Neural Network with LDA: Achieved the highest Area Under the ROC Curve (AUC), indicating excellent overall classification performance. It also had a high F1 score (0.988), signifying a good balance between precision and recall. Logistic Regression: Showed high recall (1.000) ensuring all malignant cases were correctly identified, but exhibited high variability depending on the processed sample. Other Models: Naive Bayes, Random Forest, KNN, and Neural Network with PCA all displayed varying degrees of accuracy and sensitivity. Key Metrics:

F1 Score: Neural Network with LDA achieved the best F1 score (0.988), indicating a well-balanced model in terms of precision and recall. Sensitivity: Both Neural Network with LDA (0.976) and Logistic Regression (1.000) achieved high sensitivity, effectively identifying malignant cases. Balanced Accuracy: Neural Network with LDA showcased the best balanced accuracy (0.988), signifying its effectiveness across both benign and malignant classifications.

Chapter 4

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. Put another way, 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 classifiers completeness. A low recall indicates many False Negatives.

The F1 Score is the $2 \times ((\text{precision} \times \text{recall}) / (\text{precision} + \text{recall}))$. It is also called the F Score or the F Measure. Put another way, the F1 score conveys the balance between the precision and the recall.

The Neural Network with LDA model produced the greatest sensitivity findings, notably in identifying breast cancer malignancies, and also had an impressive F1 score. This hybrid strategy, which combined Neural Networks with dimensionality reduction using LDA, outperformed expectations in a variety of parameters, including sensitivity, F1 score, and balanced accuracy.

Specifically, the Neural Network with LDA model demonstrated its ability to properly identify malignant cases, demonstrating its potential as a powerful tool for breast cancer diagnosis. This model outperformed other techniques by combining the strength of neural networks with LDA's dimensionality reduction capabilities.

While Logistic Regression was excellent at detecting malignant patients with high recall rates, its overall performance fluctuated, indicating possible limits in wider classification accuracy.

Other models, such as Naive Bayes and Random Forest, performed well but lacked the full features of the Neural Network with LDA model. This emphasizes the need of picking algorithms that are customized to the unique difficulties and goals of the activity at hand. In this scenario, the Neural Network with LDA model emerged as the preferable option, demonstrating its usefulness and promise to further breast cancer detection and classification efforts.

```
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=maply(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	Neural_PCA	1.0000000
2	Specificity	KNN	1.0000000
3	Pos Pred Value	KNN	1.0000000
4	Neg Pred Value	Logistic_regr	1.0000000
5	Precision	KNN	1.0000000
6	Recall	Logistic_regr	1.0000000
7	F1	Neural_PCA	0.9882353
8	Prevalence	Random_Forest	0.3716814
9	Detection Rate	Logistic_regr	0.3716814
10	Detection Prevalence	Logistic_regr	0.4070796
11	Balanced Accuracy	Neural_PCA	0.9929577

Chapter 5

Conclusion

This study treats the Wisconsin Madison Breast Cancer diagnosis problem as a pattern classification problem. In this report we investigated several machine learning model and we selected the optimal model by selecting a high accuracy level combined with a low rate of false-negatives (the means that the metric is high sensitivity).

The Neural Network with LDA model had the optimal results for F1 (0.9879518), Sensitivity (0.9761905) and Balanced Accuracy (0.9880952)

This study explored the application of machine learning models for breast cancer cell classification. The analysis revealed that the Neural Network with LDA model achieved the best results in terms of F1 score, sensitivity, and balanced accuracy. This suggests its potential as a valuable tool for accurate cancer cell classification, but it's important to note that these findings are based on a specific dataset. Further research on larger and more diverse datasets is recommended for broader validation.

Limitations:

The research in this paper was based on the Wisconsin Breast Cancer Dataset, which, although widely used, may not completely capture the intricacies of real-world circumstances. This constraint advises care when generalizing the results outside the boundaries of the dataset used. Furthermore, the research focused on a small number of machine learning models, thus disregarding other methods that may provide better results. As a result, the findings may not accurately reflect machine learning's skills in breast cancer detection and categorization.

Future Work:

To overcome the limitations highlighted above and strengthen the findings, future research should aim to verify the results on a larger and more varied breast cancer dataset. This will improve the suggested models' generalizability and reliability across diverse demographics and therapeutic situations. Furthermore, investigating the use of sophisticated deep learning architectures shows potential for improving accuracy and sensitivity in breast cancer detection. These complex neural network architectures may find intricate patterns and correlations in the data, improving the models' prediction ability.

Furthermore, there is a compelling need to investigate the interpretability of the Neural Network using LDA model. Understanding the decision-making processes that underpin this hybrid approach is critical for building confidence and acceptability within the medical community. Clinicians can make better judgments about patient care and treatment techniques if they understand the elements that influence the model's predictions. Future research should emphasize explaining the interpretability of these complicated machine learning models, opening the road for their use in clinical practice.

Chapter 6

References

- Hong, K., Yim, K., & Kim, K. (2024, March 4). Breast Cancer Cell Data Analysis and visualization using R. Journal of Student Research. <https://www.jsr.org/hs/index.php/path/article/view/3524>
- Wisconsin Breast Cancer. Bem vindo. (n.d.). <https://www.dcc.fc.up.pt/~ltorgo/Regression/wbc.html>
- RPubS. (n.d.). https://rpubs.com/Aakansha_garg/aakansha_cancer
- Ryckel, F. de. (2017, November 19). Machine learning with R. t. <https://fderyckel.github.io/machinelearningwithr/case-study-wisconsin-breast-cancer.html>
- Sachinudgam. (2022, February 1). Breast cancer analysis in R programming. Kaggle. <https://www.kaggle.com/code/sachinudgam/breast-cancer-analysis-in-r-programming>
- Survival analysis method with R using breast cancer dataset. (n.d.). <https://www.mathstat.dal.ca/~tsusko/honours-theses/wenhui-wang.pdf>
- American Cancer Society. (2022, March 1). Survival Rates for Breast Cancer. American Cancer Society. <https://www.cancer.org/cancer/breast-cancer/understanding-a-breast-cancer-diagnosis/breast-cancer-survival-rates.html>
- UCI Machine Learning Repository. (1995, November 1). UCI Machine Learning Repository: Breast Cancer Wisconsin (Diagnostic) Data Set Dataset. The University of Wisconsin Clinical Sciences Center and Computer Science Dept. [https://archive.ics.uci.edu/ml/datasets/breast+cancer+wisconsin+\(diagnostic\)](https://archive.ics.uci.edu/ml/datasets/breast+cancer+wisconsin+(diagnostic))
- Jonathan Gelford, Martin Goros, Brian Hernandez, and Alex Bokoro. (July 2018). A System for an Accountable Data Analysis Process in R. Retrieved August,10,2022 from <https://journal.r-project.org/archive/2018/RJ-2018-001/RJ-2018-001.pdf>
- UCLA Statistical Methods and Data Analytics. (n.d.). Logit Regression | R Data Analysis Examples. Retrieved August 22, 2022, from <https://stats.oarc.ucla.edu/r/dae/logit-regression/>
- MeDiscovery. (2020, August 4). AUC-ROC. BioinformaticsAndMe. <https://bioinformaticsandme.tistory.com/328>
- KOSIS. (1999–2019). KOSIS-24 Cancer Patients Statistics Dataset. Korean Statistical Office. https://kosis.kr/statHtml/statHtml.do?orgId=117&tblId=DT_117N_A00023&conn_path=I3