VIT BHOPAL

Report for Advanced data analytics project

Breast Cancer Prediction and Analysis

Group Number – 1

Group Members:

- Abhijay Wadhwani(18BCE10005)
- Kartikey Naraiyan Garg(18BCE10140)
- Abhishek Mishra(18BCE10008)
- Ritik Jain(18BCE10220)
- Kunal Verma(18BCE10150)
- Chitwan Singh(18BCE10085)
- Bhuvnesh Birla(18BCE10079)
- Rakshita Sharma(18BCE10210)
- Gourangi Tiwari(18BCE10101)
- Shreyas Chiney(18BCE10260)

NAS 2001- NASSCOM-
Advance Data Analytics
B11
Dr. Nilamadhab Mishra

1.Overview

Breast cancer is one of the most severe cancers. It has taken hundreds of thousands of lives every year. Early prediction of breast cancer plays an important role in successful treatment and saving the lives of thousands of patients every year. However, the conventional approaches are limited in providing such capability. The recent breakthrough of data analytics and data mining techniques have opened a new door for healthcare diagnostic and prediction. Machine learning methods for diagnosis can significantly increase processing speed and on a big scale can make the diagnosis significantly cheaper. So, we are going to analyze it and to try several machine learning classification models to compare their results. We can create a classifier that can help diagnose patients and predict the likelihood of breast cancer. A few machine learning techniques will be explored. This research is carried out to predict the accuracy.

2.Problem Statement

Breast cancer is the most common cancer among women. Every thirteen minutes a woman dies with the diagnosis of breast cancer. These facts have led researchers to continue studying how to treat and detect breast cancer in women, especially older women, who are of higher risk. Breast cancer awareness is an effort to raise awareness and reduce the stigma of breast cancer through education on symptoms and treatment.

Awareness surrounding breast cancer is incredibly important as early detection, often through screening, can catch the disease when it is most treatable because: -

- Breast cancer is a significant public health problem. About 252,710 new cases of invasive breast cancer will be diagnosed in women. And 2,470 men will be diagnosed with breast cancer.

- Breast cancer is the second leading cause of cancer death in women with 40,610 deaths.

- Since 2007, breast cancer death rates have remained steady in women 50 years or age or younger.

3.Objectives

General objective:

Breast cancer is one of the top cancers that occur in women. It is the second main cause of death of women in the USA and many Asian countries. So, if we could predict and identify this disease in early stages, there is a better chance of curing it. Therefore, during this project we have tried to make use of some ML algorithms to make a model for this breast cancer prediction and analysis.

Specific objective:

1. To extract and clean the dataset before using it in prediction
2. Reducing the dimensions of the dataset using LDA and PCA for better and faster diagnosis
3. The goal is to classify whether the breast cancer is benign or malignant. To achieve this we will use machine learning classification methods to fit a function that can predict the discrete class of new input
4. Modeling the data using different classification algorithms like logistic regression, Random forest, KNN, SVM (Support Vector Machines) and a Neural Network.
5. Evaluating and comparing the results from each of the model and selecting the one with the highest sensitivity.

4. Approach/Methodology

→ As the first step, we aim to collect the data from (URL: https://archive.ics.uci.edu/ml/datasets/Breast+Cancer+Wisconsin+%28Diagnostic%29) consisting of 569 instances and 32 attributes, which will be read through a csv file and converted to a data frame for further use.

→ Following next will involve tidying up the data by enumerating the outcome variable and renaming badly encoded variables. Packages such as tidy verse, dplyr would be used in this step.

→ Once our data frame is made explicable, we build a deeper understanding of our data through visualization by plotting density curves and one-on-one scatter plots and box plots for all the different attributes using the ggplot2 library.

→ Next, we check for any missing values or null values and if present we apply appropriate imputations to our data based on the pattern of missing entries and also get hints on the mechanism. The VIM package of R would come handy for this purpose.

→ Since the ratio of attributes to instances is quite high in our case, we next aim at reducing a few of the attributes to avoid overfitting of the data. We plot the correlation plot and deploy the caret package to remove highly correlated variables, providing redundant data, based on the cutoff value of 0.9.

→ To further enhance visualization and preprocess our data we apply PCA as part of EDA. This converts our original variables into a smaller number of "Principal Components". This is done by finding the straight line that best spreads the data out when it is projected along it i.e. transforming a set of x correlated variables over y samples to a set of p uncorrelated principal components over the same samples.

→ For dimensionality reduction, we also apply LDA. The LDA algorithm tries to find linear combinations of the predictor variables that can maximize the separation among the outcome classes which would then be used for predicting the class of each and every instance.

→ Now that our preprocessing is done, we would partition our final data frame into training and testing sets. We use 80% of the data for training while remaining 20% for testing. We also apply cross-validation technique to resample the data at least 15 times.

→ We will be Applying different machine learning models and determining all the performance measures, Confusion Matrix and Statistics comprising of Accuracy, sensitivity, specificity etc.

→ All the models use ROC as a metric. The ROC metric measures the auc of the roc curve of each model. This metric is independent of any threshold

→ Our first model will be doing logistic regression on the data frame where we took away the highly correlated variables which is the training dataset.

→ Our second model uses random forest and Induction. Similarly, we are using the data frame, the one where we took away the highly correlated variables and also, we will be making some diagnostic plots here.

→ Our third model uses KNN (k-nearest neighbors' algorithm) on the training dataset.

→ Our fourth model will be using the SVM (Support Vector Machines) on a non-PCA training dataset. For better results with SVM when doing it on the PCA data set.

→ Our last and best model Neural networks with LDA, to use the LDA pre-processing step, we will also create the same training and testing set.

→ After Training all the models we will do model evaluation this is done distributions are summarized in terms of the percentiles. The distributions are summarized as box plots and finally the distributions are summarized as dot plots.

→ The model which will have the best results for sensitivity (detection of breast cases) will be used in the application.

5. Literature review

Section-1

∅ Introduction

Breast cancer is the most commonly occurring cancer in women and the second most common cancer overall. There were over 2 million new cases in 2018. Breast cancer ratio are statistically higher in women in more developed countries as compared to other diseases. But it is also globally increasing day by day.

To discourage the growth of breast cancer, it is important to focus on early detection. Early diagnosis and screening are two main methods of advance detection of breast cancer. The poor regions can be made aware by familiarizing with early diagnosis programs, as stated by the World Health Organization. So, mortality rate of women due to breast cancer can be reduced if it can be detected at a relatively early stage that includes early diagnosis, screening, mammography and Clinical Breast Exam (CBE).

Certainly, analysis of a patient's clinical data and physician's judgment are the most considerable features in diagnosis. Most of the possible medical flaws can be avoided by the using classification systems, and also offer healthcare data to be analyzed in lesser time and in more exhaustive manner. Accurate and timely prediction of breast cancer allows physicians and healthcare providers to make most favorable decisions about the patient treatment.

∅ Preview of the previous findings

- Ultrasound characterization of breast masses by S. Gokhale written by proposed a system where they found that doctors have known and experienced that breast cancer occurs when some breast cells begin to grow abnormally.
    - These cells divide more briskly and disperse faster than healthy cells do and continue to accumulate, forming a lump or mass that they may start causing pain. Cells may spread rapidly through your breast to your lymph nodes or to other parts of your body.
    - Some women can be at a higher risk for breast cancer because of their family history, lifestyle, obesity, radiation, and reproductive factors.
    - In the case of cancer, if the diagnosis occurs quickly, the patient can be saved as there have been advances in cancer treatment.
- The purpose of the paper "Breast Cancer Prediction and Detection Using Data Mining Classification Algorithms: A Comparative Study" by Mumine Kaya Keles

was to predict and detect breast cancer early even if the tumor size is petite with non-invasive and painless methods that use data mining classification algorithms.

Section-2

∅ Our objective

In vision of the problem statement described in the introduction section, a classification model is proposed with boosted accuracy to predict the breast cancer patient. The framework is composed of the following important phases:

- Dataset Selection
- Data Pre-processing
- Learning by Classifier (Training) i.e. SVM, Logistic Regression and KNN.
- Achieving trained model with highest accuracy
- Using trained model for prediction

∅ Our Recommendation:

- Train the datasets using Random Forest, k-nearest neighbors' algorithm (k-NN), Support Vector Machine (SVM) and Naïve Bayes.
- Start with something simple dataset, experiment or analyze it and proceed to build on.

∅ Conclusion

A decision support system for predicting breast cancer helps the real-world patients and assists physicians in making optimum, accurate and timely decisions, and reduces the overall cost of treatment. Breast cancer if found at an early stage will help save the lives of thousands of women or even men. By using machine learning algorithms, we will be able to classify and predict the cancer into being or malignant. Machine learning algorithms can be used for medical oriented research, it advances the system, reduces human errors and lowers manual mistakes.

Implementaion ->

Dataset

Features are computed from a digitized image of a fine needle aspirate (FNA) of a breast mass. They describe characteristics of the cell nuclei present in the image. The dataset is taken from the UCI Machine learning website

Attribute Information:

1) ID number
2) Diagnosis (M = malignant, B = benign)
3-32)

Ten real-valued features are computed for each cell nucleus:

a) radius (mean of distances from center to points on the perimeter)
b) texture (standard deviation of gray-scale values)
c) perimeter
d) area
e) smoothness (local variation in radius lengths)
f) compactness (perimeter^2 / area - 1.0)
g) concavity (severity of concave portions of the contour)
h) concave points (number of concave portions of the contour)
i) symmetry
j) fractal dimension ("coastline approximation" - 1)

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 features. For instance, field 3 is Mean Radius, field 13 is Radius SE, field 23 is Worst Radius.

All feature values are recoded with four significant digits.

Missing attribute values: none

Class distribution: 357 benign, 212 malignant

Breast cancer is a malignant tumor that grows in or around the breast tissue, mainly in the milk ducts and glands. A tumor usually starts as a lump or calcium deposit that develops as a result of abnormal cell growth. Most breast lumps are benign but can be premalignant (may become cancer).

```r
df <- read_csv("dataset/BreastCancer.csv")

# This is defintely an most important step:
# Check for appropriate class on each of the variable.
glimpse(df)
```

```
## Observations: 569
## Variables: 32
## $ id                      <dbl> 842302, 842517, 84300903, 84348301, 8435…
## $ diagnosis               <chr> "M", "M", "M", "M", "M", "M", "M", "M", …
## $ radius_mean             <dbl> 17.990, 20.570, 19.690, 11.420, 20.290, …
## $ texture_mean            <dbl> 10.38, 17.77, 21.25, 20.38, 14.34, 15.70…
## $ perimeter_mean          <dbl> 122.80, 132.90, 130.00, 77.58, 135.10, 8…
## $ area_mean               <dbl> 1001.0, 1326.0, 1203.0, 386.1, 1297.0, 4…
## $ smoothness_mean         <dbl> 0.11840, 0.08474, 0.10960, 0.14250, 0.10…
## $ compactness_mean        <dbl> 0.27760, 0.07864, 0.15990, 0.28390, 0.13…
## $ concavity_mean          <dbl> 0.30010, 0.08690, 0.19740, 0.24140, 0.19…
## $ concave_points_mean     <dbl> 0.14710, 0.07017, 0.12790, 0.10520, 0.10…
## $ symmetry_mean           <dbl> 0.2419, 0.1812, 0.2069, 0.2597, 0.1809, …
## $ fractal_dimension_mean  <dbl> 0.07871, 0.05667, 0.05999, 0.09744, 0.05…
## $ radius_se               <dbl> 1.0950, 0.5435, 0.7456, 0.4956, 0.7572, …
## $ texture_se              <dbl> 0.9053, 0.7339, 0.7869, 1.1560, 0.7813, …
## $ perimeter_se            <dbl> 8.589, 3.398, 4.585, 3.445, 5.438, 2.217…
## $ area_se                 <dbl> 153.40, 74.08, 94.03, 27.23, 94.44, 27.1…
## $ smoothness_se           <dbl> 0.006399, 0.005225, 0.006150, 0.009110, …
## $ compactness_se          <dbl> 0.049040, 0.013080, 0.040060, 0.074580, …
## $ concavity_se            <dbl> 0.05373, 0.01860, 0.03832, 0.05661, 0.05…
## $ concave_points_se       <dbl> 0.015870, 0.013400, 0.020580, 0.018670, …
## $ symmetry_se             <dbl> 0.03003, 0.01389, 0.02250, 0.05963, 0.01…
## $ fractal_dimension_se    <dbl> 0.006193, 0.003532, 0.004571, 0.009208, …
## $ radius_worst            <dbl> 25.38, 24.99, 23.57, 14.91, 22.54, 15.47…
## $ texture_worst           <dbl> 17.33, 23.41, 25.53, 26.50, 16.67, 23.75…
## $ perimeter_worst         <dbl> 184.60, 158.80, 152.50, 98.87, 152.20, 1…
## $ area_worst              <dbl> 2019.0, 1956.0, 1709.0, 567.7, 1575.0, 7…
## $ smoothness_worst        <dbl> 0.1622, 0.1238, 0.1444, 0.2098, 0.1374, …
## $ compactness_worst       <dbl> 0.6656, 0.1866, 0.4245, 0.8663, 0.2050, …
## $ concavity_worst         <dbl> 0.71190, 0.24160, 0.45040, 0.68690, 0.40…
## $ concave_points_worst    <dbl> 0.26540, 0.18600, 0.24300, 0.25750, 0.16…
## $ symmetry_worst          <dbl> 0.4601, 0.2750, 0.3613, 0.6638, 0.2364, …
## $ fractal_dimension_worst <dbl> 0.11890, 0.08902, 0.08758, 0.17300, 0.07…
```

So we have 569 observations with 32 variables. Ideally for so many variables, it would be appropriate to get a few more observations.

Tidy the data

Tidy data is a standard way of mapping the meaning of a dataset to its structure. A dataset is messy or tidy depending on how rows, columns and tables are matched up with observations, variables and types. In tidy data:

1.Each variable forms a column.
2.Each observation forms a row.
3.Each type of observational unit forms a table.

Following is the code of our implementation of tidy the data.

```
df$diagnosis <- as.factor(df$diagnosis)


#df <- df %>% rename(concave_points_mean = `concave points_mean`,
#                    concave_points_se = `concave points_se`,
#                    concave_points_worst = `concave points_worst`)
```

Understanding the data

This is the circular phase of our dealing with data. This is where each of the transforming, visualizing and modeling stage reinforce each other to create a better understanding.

Checking for missing values

```
map_int(df, function(.x) sum(is.na(.x)))
```

```
##                     id              diagnosis            radius_mean
##                      0                      0                      0
##            texture_mean         perimeter_mean              area_mean
##                      0                      0                      0
##         smoothness_mean        compactness_mean          concavity_mean
##                      0                      0                      0
##      concave_points_mean           symmetry_mean  fractal_dimension_mean
##                      0                      0                      0
##               radius_se              texture_se            perimeter_se
##                      0                      0                      0
##                 area_se           smoothness_se          compactness_se
##                      0                      0                      0
##            concavity_se       concave_points_se             symmetry_se
##                      0                      0                      0
##     fractal_dimension_se            radius_worst           texture_worst
##                      0                      0                      0
##         perimeter_worst              area_worst        smoothness_worst
##                      0                      0                      0
##       compactness_worst         concavity_worst    concave_points_worst
##                      0                      0                      0
##          symmetry_worst fractal_dimension_worst
##                      0                      0
```

As we see there are no missing values.

In the case that there would be many missing values, we would go on the transforming data for some appropriate imputation.

Let's check how balanced is our response variable

```
round(prop.table(table(df$diagnosis)), 2)
```

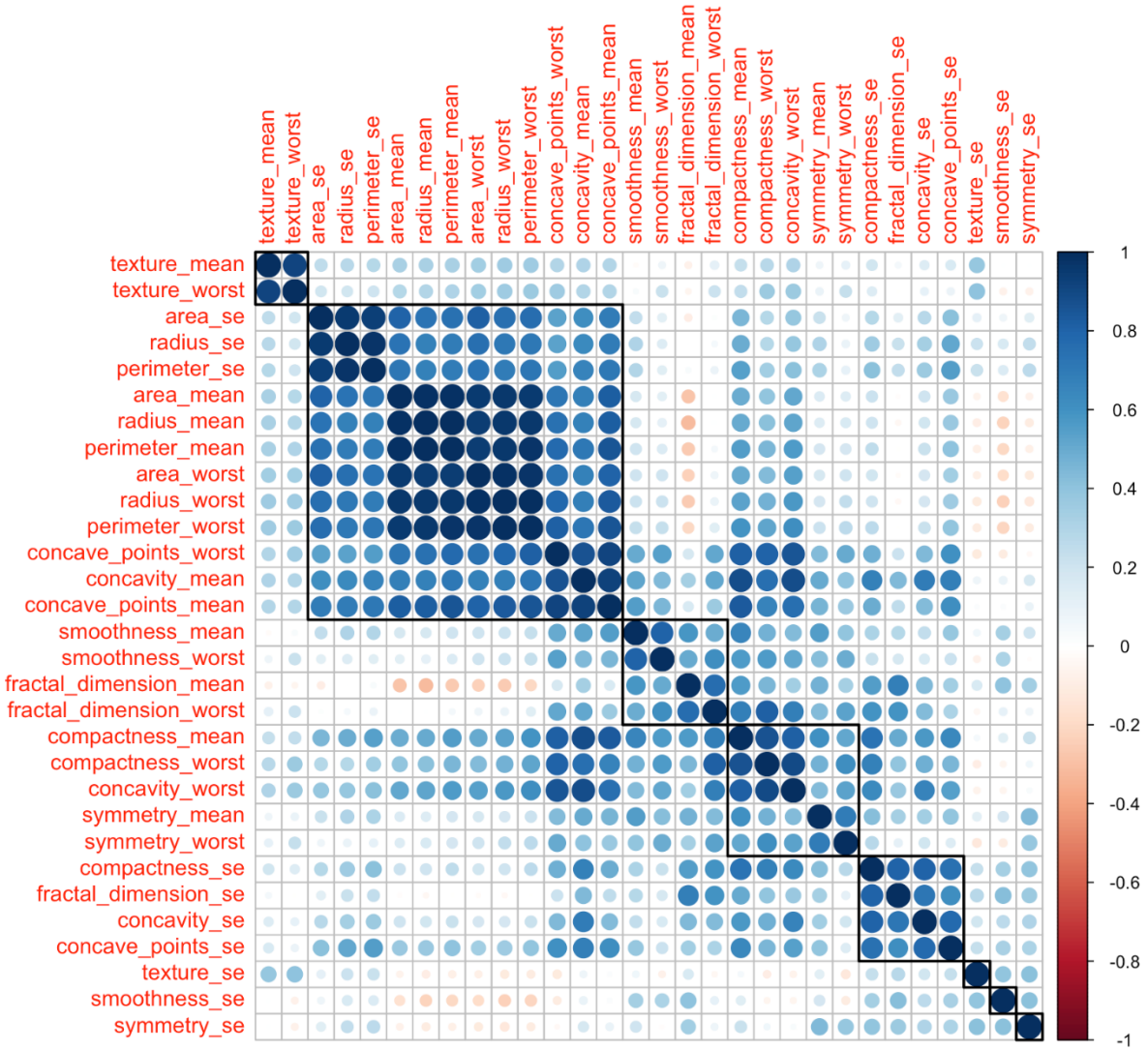
```
##
##    B    M
## 0.63 0.37
```

The response variable is slightly unbalanced.

Let's check for correlations.

For an analysis to be robust it is good to remove multicollinearity (i. e remove highly correlated predictors). Because multicollinearity reduces the precision of the estimate coefficients, which weakens the statistical power of the regression model. And it might become difficult to trust the p-values to identify independent variables that are statistically significant.

```
df_corr <- cor(df %>% select(-id, -diagnosis))
corrplot::corrplot(df_corr, order = "hclust", tl.cex = 1, addrect = 8)
```



There are few variables that are highly correlated such as perimeter_worst with area_mean,area_worst with area_mean,radius_mean with perimeter_mean and many more variables . On the next step, we will remove the highly correlated ones using the caret package. Because when we have two independent **variables** that are very highly **correlated**, we **should remove** one of them because it leads to multicollinearity.

Transformation and preprocessing

```
library(caret)
# The findcorrelation() function from caret package remove highly correlated predictors
# based on whose correlation is above 0.9. This function uses a heuristic algorithm
# to determine which variable should be removed instead of selecting blindly
df2 <- df %>% select(-findCorrelation(df_corr, cutoff = 0.9))

#Number of columns for our new data frame
ncol(df2)
```

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

So our new data frame df2 is 10 variables shorter.

Using PCA Algorithm

PCA is a type of linear transformation on a given data set that has values for a certain number of variables (coordinates) for a certain amount of spaces. This linear transformation fits this dataset to a new coordinate system in such a way that the most significant variance is found on the first coordinate, and each subsequent coordinate is orthogonal to the last and has a lesser variance. In this way, you transform a set of x correlated variables over y samples to a set of p uncorrelated principal components over the same samples.

Let's first go on an unsupervised analysis with a PCA analysis. To do so, we will remove the id and diagnosis variable, then we will also scale and center the variables.

First applying on the original dataset with all the correlated variables

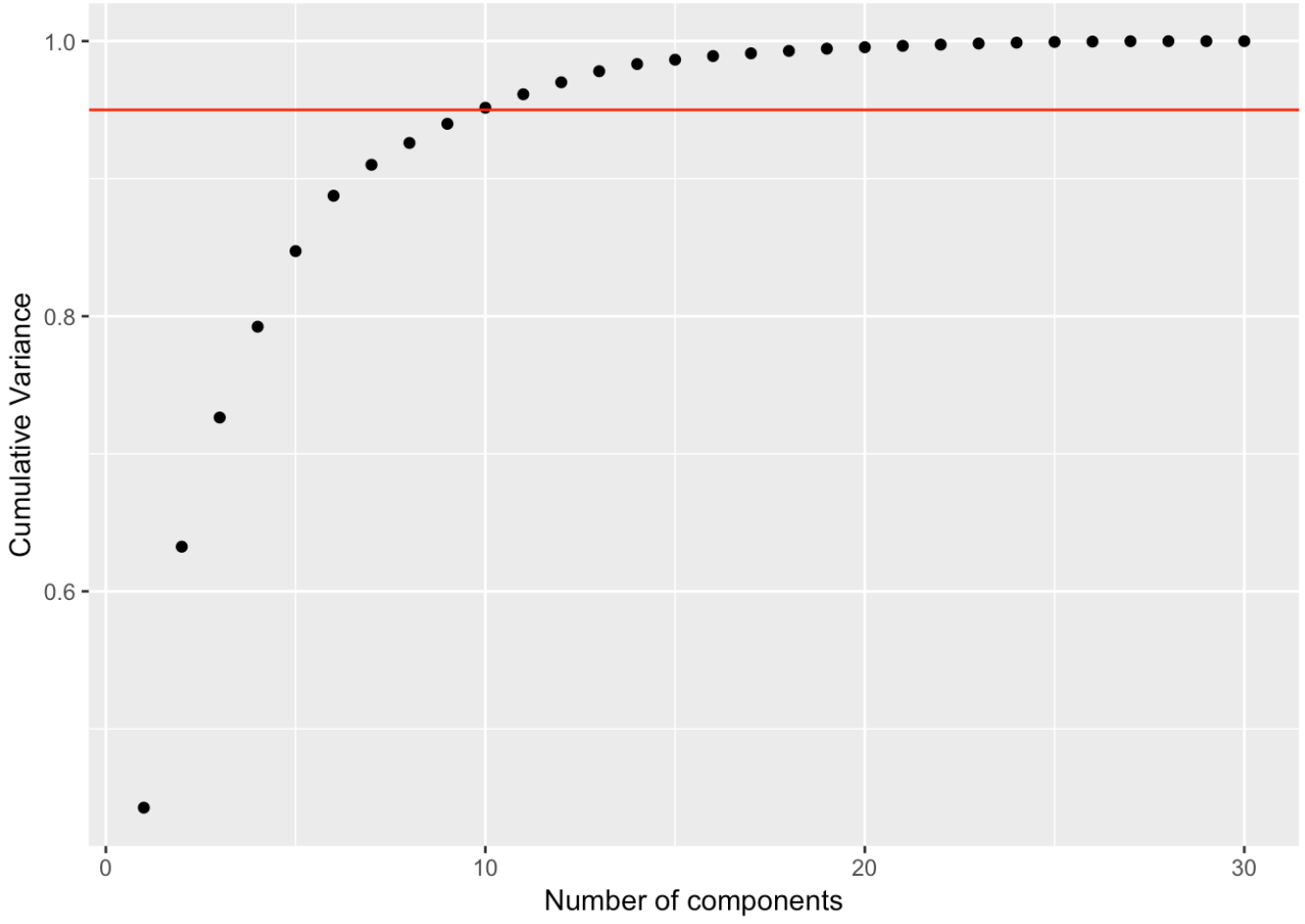
```
preproc_pca_df <- prcomp(df %>% select(-id, -diagnosis), scale = TRUE, center = TRUE)
summary(preproc_pca_df)
```

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

A principal component is a normalized linear combination of the original predictors in a data set. Let's say we have a set of predictors as $X^1, X^2...,X^p$
The principal component can be written as:

$$Z^1 = \Phi^{11}X^1 + \Phi^{21}X^2 + \Phi^{31}X^3 + ... + \Phi^{p1}X^p$$

```r
# Calculate the proportion of variance explained
pca_df_var <- preproc_pca_df$sdev^2
pve_df <- pca_df_var / sum(pca_df_var)
cum_pve <- cumsum(pve_df)
pve_table <- tibble(comp = seq(1:ncol(df %>% select(-id, -diagnosis))), pve_df, cum_pve)

ggplot(pve_table, aes(x = comp, y = cum_pve)) +
  geom_point() +
  geom_abline(intercept = 0.95, color = "red", slope = 0) +
  labs(x = "Number of components", y = "Cumulative Variance")
```



With the original dataset, 95% of the variance is explained with 10 PC's.

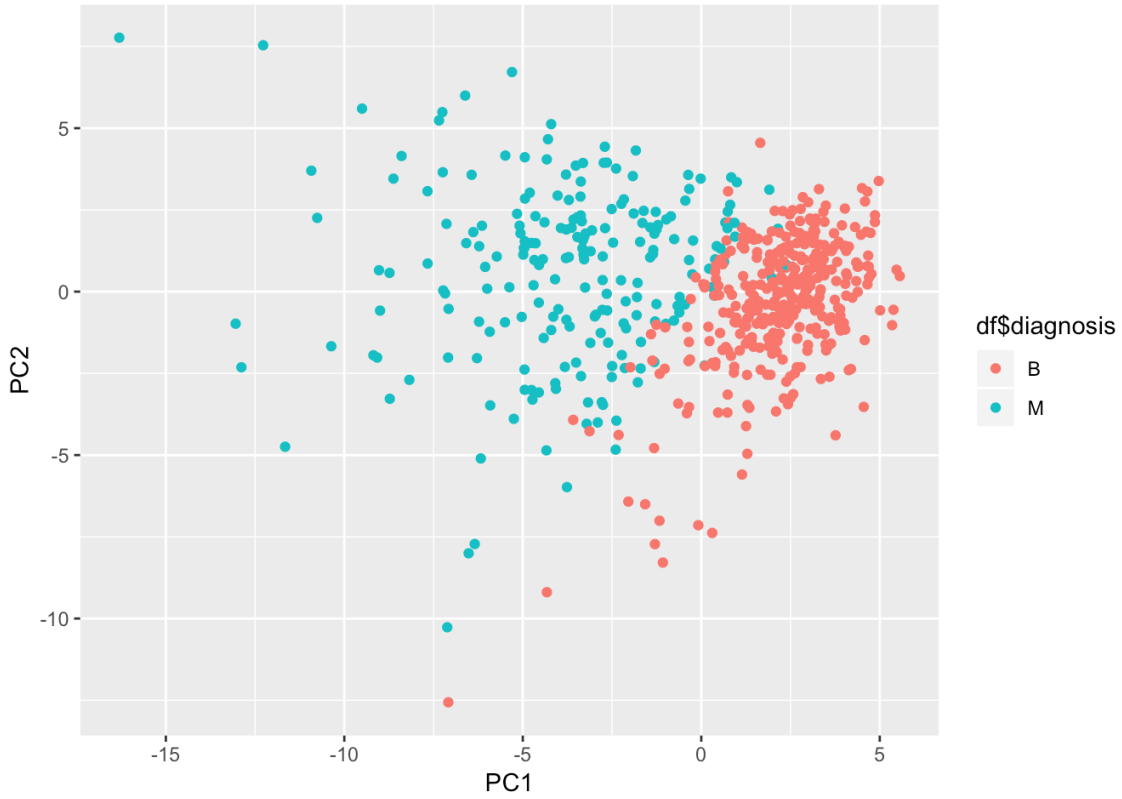
First principal component is a linear combination of original predictor variables which captures the maximum variance in the data set. It determines the direction of highest variability in the data. Larger the variability captured in first component, larger the information captured by component. No other component can have variability higher than first principal component.

The first principal component results in a line which is closest to the data i.e. it minimizes the sum of squared distance between a data point and the line.

Second principal component (Z^2) is also a linear combination of original predictors which captures the remaining variance in the data set and is uncorrelated with Z^1. In other words, the correlation between first and second component should is zero.

```
pca_df <- as_tibble(preproc_pca_df$x)


ggplot(pca_df, aes(x = PC1, y = PC2, col = df$diagnosis)) + geom_point()
```

checking on the most influential variables for the first 2 components

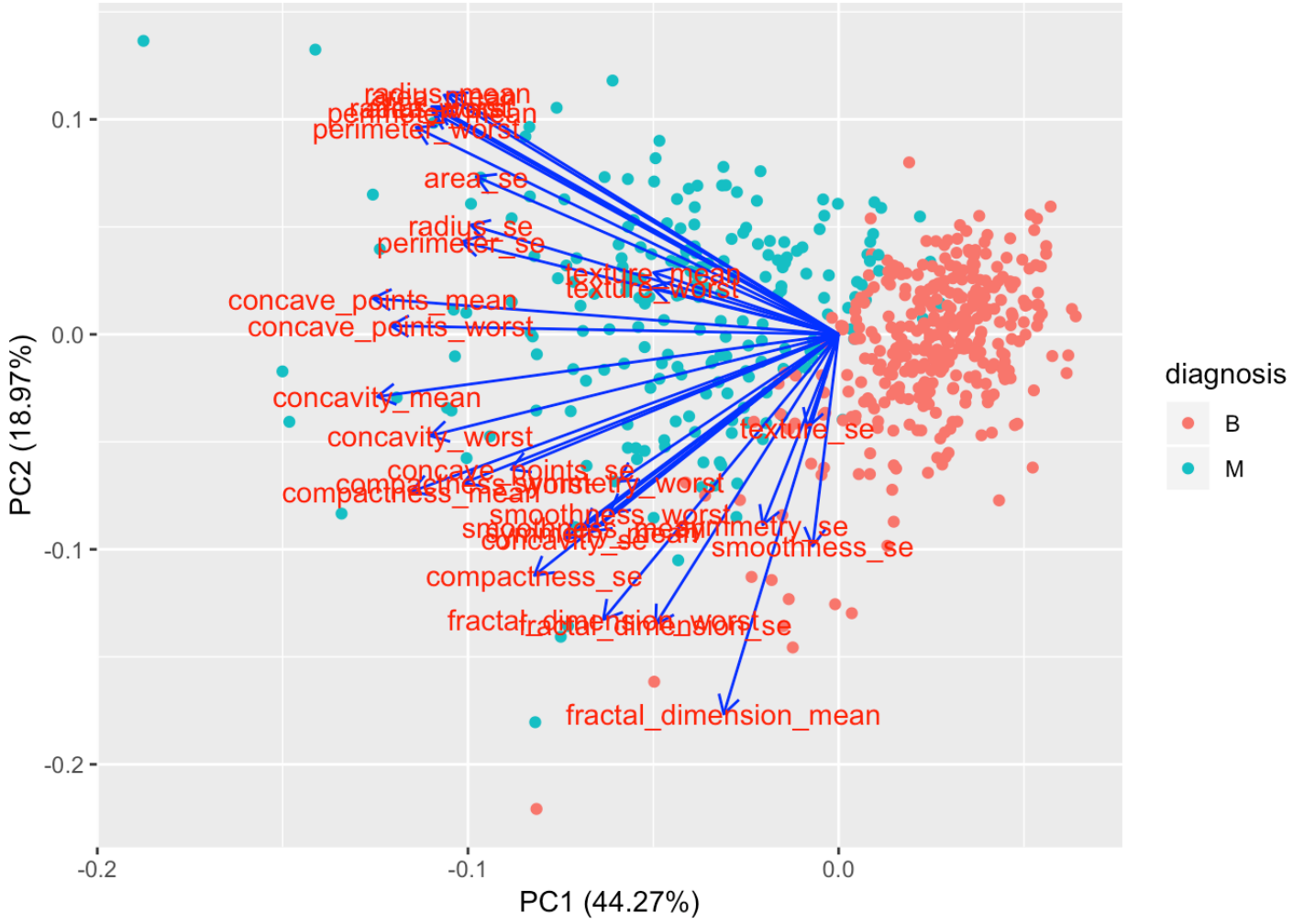


As expected, the first 2 components managed to separate the diagnosis quite well. Lots of potential here.

For a more detailed analysis of what variables are the most influential in the first 2 components, we can use the ggfortify library.

```
library(ggfortify)
autoplot(preproc_pca_df, data = df,  colour = 'diagnosis',
                     loadings = FALSE, loadings.label = TRUE, loadings.colour = "blue")
```

Plot:



We will do the same exercise with our second df, the one where we removed the highly correlated predictors.

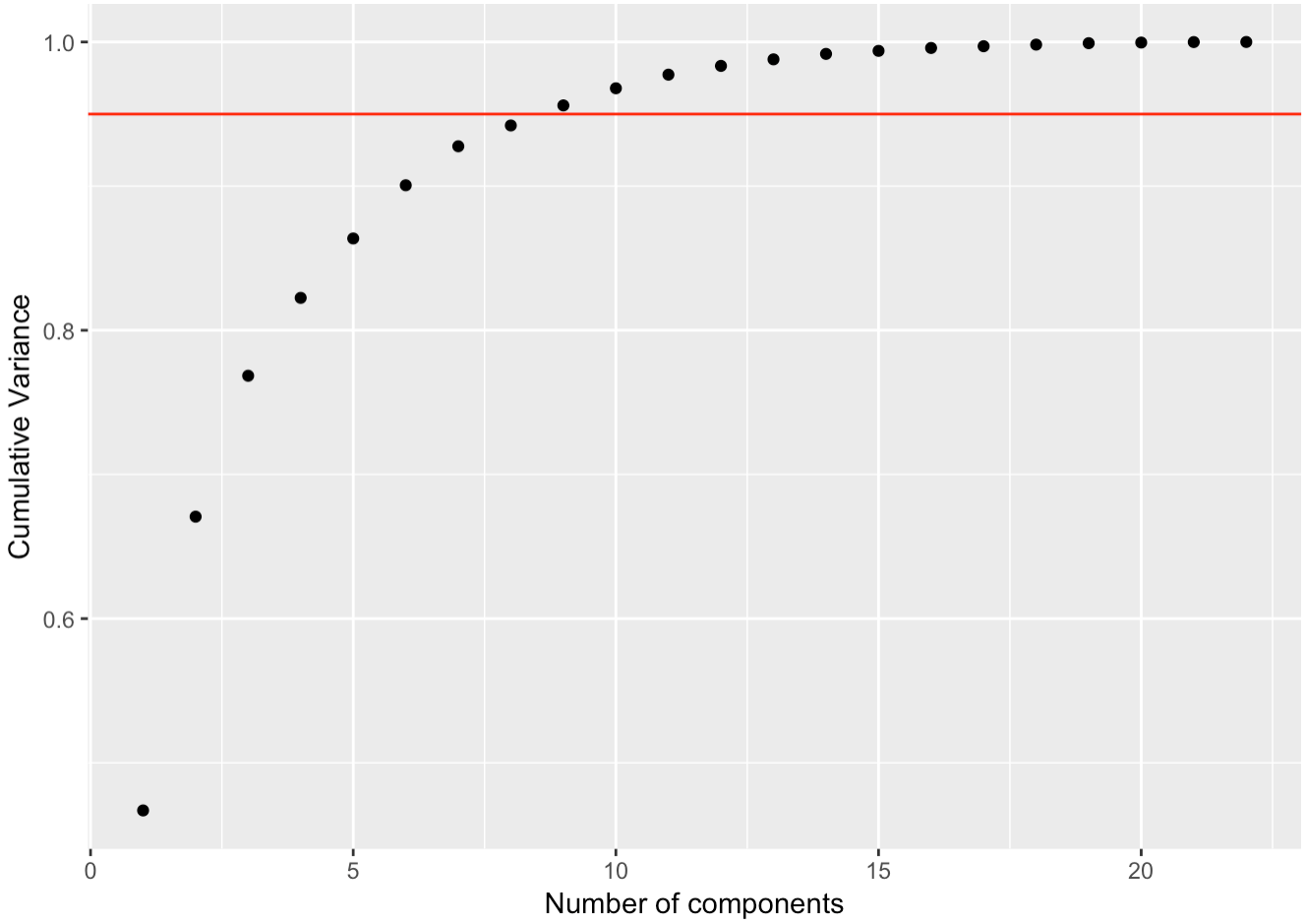
```
preproc_pca_df2 <- prcomp(df2, scale = TRUE, center = TRUE)
summary(preproc_pca_df2)
```

```
## Importance of components:
##                            PC1    PC2     PC3     PC4     PC5     PC6
## Standard deviation     3.2051 2.1175 1.46634 1.09037 0.95215 0.90087
## Proportion of Variance 0.4669 0.2038 0.09773 0.05404 0.04121 0.03689
## Cumulative Proportion  0.4669 0.6707 0.76847 0.82251 0.86372 0.90061
##                            PC7     PC8    PC9    PC10    PC11    PC12
## Standard deviation     0.77121 0.56374 0.5530 0.51130 0.45605 0.36602
## Proportion of Variance 0.02703 0.01445 0.0139 0.01188 0.00945 0.00609
## Cumulative Proportion  0.92764 0.94209 0.9560 0.96787 0.97732 0.98341
##                           PC13    PC14   PC15   PC16    PC17   PC18   PC19
## Standard deviation     0.31602 0.28856 0.2152 0.2098 0.16346 0.1558 0.1486
## Proportion of Variance 0.00454 0.00378 0.0021 0.0020 0.00121 0.0011 0.0010
## Cumulative Proportion  0.98795 0.99174 0.9938 0.9958 0.99706 0.9982 0.9992
##                           PC20    PC21    PC22
## Standard deviation     0.09768 0.08667 0.03692
## Proportion of Variance 0.00043 0.00034 0.00006
## Cumulative Proportion  0.99960 0.99994 1.00000
```

We can see that the 1st Principle component explains 46.69% of the variance and the 2nd Principle component explains 20.38%.

```r
pca_df2_var <- preproc_pca_df2$sdev^2

# proportion of variance explained
pve_df2 <- pca_df2_var / sum(pca_df2_var)
cum_pve_df2 <- cumsum(pve_df2)
pve_table_df2 <- tibble(comp = seq(1:ncol(df2)), pve_df2, cum_pve_df2)

ggplot(pve_table_df2, aes(x = comp, y = cum_pve_df2)) +
  geom_point() +
  geom_abline(intercept = 0.95, color = "red", slope = 0) +
  labs(x = "Number of components", y = "Cumulative Variance")
```

Plotting the number of components and cumulative variance graph.



We can see that In this case, around 8 PC's explained 95% of the variance.

Using LDA Algorithm

Linear Discriminant Analysis is a well-established machine learning technique and classification method for predicting categories. Its main advantages, compared to other classification algorithms such as neural networks and random forests, are that the model is interpretable, and that prediction is easy. Linear Discriminant Analysis is frequently used as a dimensionality reduction technique for pattern recognition or classification of data and machine learning.

LDA determines group means and computes, for each individual, the probability of belonging to the different groups. The individual is then affected to group with the highest probability score.

The lda() outputs contain the following elements:

Prior probabilities of groups: the proportion of training observations in each group.
Group means: group center of gravity. Shows the mean of each variable in each group.
Coefficients of linear discriminants: Shows the linear combination of predictor variables that are used to form the LDA decision rule.

We have used LDA to reduce the dimensions of our project.
Breast cancer dataset have many different classes and categories

1. We are taking the different classes of our dataset. Here package MASS ("Modern Applied Statistics with S") contains LDA and QDA.

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

-> Now after reducing the dimensions of our dataset we make predictions using the predict functionality
-> the predict functions give us the predicted classes of the observations

2. Here the predict function is used to predict the classes for preproc_lda_df .
3. cbind is used to combine two columns and as data frame is used to convert to data frame

```
# Making a df out of the LDA for visualization purpose.
predict_lda_df <- predict(preproc_lda_df, df)$x %>%
  as_data_frame() %>%
  cbind(diagnosis = df$diagnosis)
```

```
## Warning: `as_data_frame()` is deprecated, use `as_tibble()` (but mind the new semantics).
## This warning is displayed once per session.
```

```
glimpse(predict_lda_df)
```

```
## Observations: 569
## Variables: 2
## $ LD1       <dbl> 3.3257395, 2.3298023, 3.7416859, 4.0209903, 2.2754286,…
## $ diagnosis <fct> M, M, M, M, M, M, M, M, M, M, M, M, M, M, M, M, M, …
```

4.Using the glimpse function from dplyr package we print the output of the prediction we made in our last step

```
glimpse(predict_lda_df)
```

```
## Observations: 569
## Variables: 2
## $ LD1       <dbl> 3.3257395, 2.3298023, 3.7416859, 4.0209903, 2.2754286,…
## $ diagnosis <fct> M, M, M, M, M, M, M, M, M, M, M, M, M, M, M, M, M, …
```

-> LD1: it is a linear discriminant function. It achieves the maximal separation of our class. Lda creates one or more linear combinations of predictors , creating a new variable for each function.

Modelling the Data

Now we are going to develop our model. We will start by first splitting our dataset into two parts using caret; one as training set for the model, and the other as a test set to validate the predictions that the model will make. If we omit this step, the model will be trained and tested on the same dataset, and it will underestimate the true error rate, a phenomenon known as overfitting.

```
set.seed(1815)
df3 <- cbind(diagnosis = df$diagnosis, df2)
df_sampling_index <- createDataPartition(df3$diagnosis, times = 1, p = 0.8, list = FALSE)
df_training <- df3[df_sampling_index, ]
df_testing <-  df3[-df_sampling_index, ]
df_control <- trainControl(method="cv",
                           number = 15,
                           classProbs = TRUE,
                           summaryFunction = twoClassSummary)
```

we would partition our final data frame into training and testing sets. We use 80% of the data for training while remaining 20% for testing. We also apply cross-validation technique to resample the data at least 15 times.

1)Logistic Regression

Logistic regression is a type of classification method in statistical learning. It models the probability that Y (Y=p(X)) belongs to a particular class using the logistic function,

$$p(X) = e^{\beta_0+\beta_1 x_1}/(1+e^{\beta_0+\beta_1 x_1})$$

where the coefficients are estimated using the maximum likelihood function. The response is a probability estimate that indicates the odds in which an observation will belong to a certain class. For binary classes, typically, the decision boundary is 0.5. In this case, if the probability is > 0.5, then the prediction is M, the diagnosis is malignant.

Implementation of Logistic Regression

Our first model is doing logistic regression on df2, the data frame where we took away the highly correlated variables.

```
model_logreg_df <- train(diagnosis ~., data = df_training, method = "glm",
                         metric = "ROC", preProcess = c("scale", "center"),
                         trControl = df_control)


prediction_logreg_df <- predict(model_logreg_df, df_testing)
cm_logreg_df <- confusionMatrix(prediction_logreg_df, df_testing$diagnosis, positive = "M")
cm_logreg_df
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  B  M
##          B 71  2
##          M  0 40
##
##                Accuracy : 0.9823
##                  95% CI : (0.9375, 0.9978)
##     No Information Rate : 0.6283
##     P-Value [Acc > NIR] : <2e-16
##
##                   Kappa : 0.9617
##  Mcnemar's Test P-Value : 0.4795
##
##             Sensitivity : 0.9524
##             Specificity : 1.0000
##          Pos Pred Value : 1.0000
##          Neg Pred Value : 0.9726
##              Prevalence : 0.3717
##          Detection Rate : 0.3540
##    Detection Prevalence : 0.3540
##       Balanced Accuracy : 0.9762
##
##        'Positive' Class : M
##
```

By applying Logistic Regression, it predict positive class that means diagnosis as M (class=1) and we attain quit high accuracy of 98.23% with the Sensitivity and Specificity of 90.24% and 100% respectively.

2)Random Forest Algorithm

Random forest is a tree-based algorithm which involves building several trees (decision trees), then combining their output to improve generalization ability of the model.

While creating random trees it split into different nodes or subsets. Then it searches for the best outcome from the random subsets. This results in the better model of the algorithm.

Implementation

Firstly, we you create train and test folds

Then we fit a model using the train function. We use the metric ROC (It is a plot of the **True Positive Rate (on the y-axis)** versus the **False Positive Rate (on the x-axis)** for every possible classification threshold.)

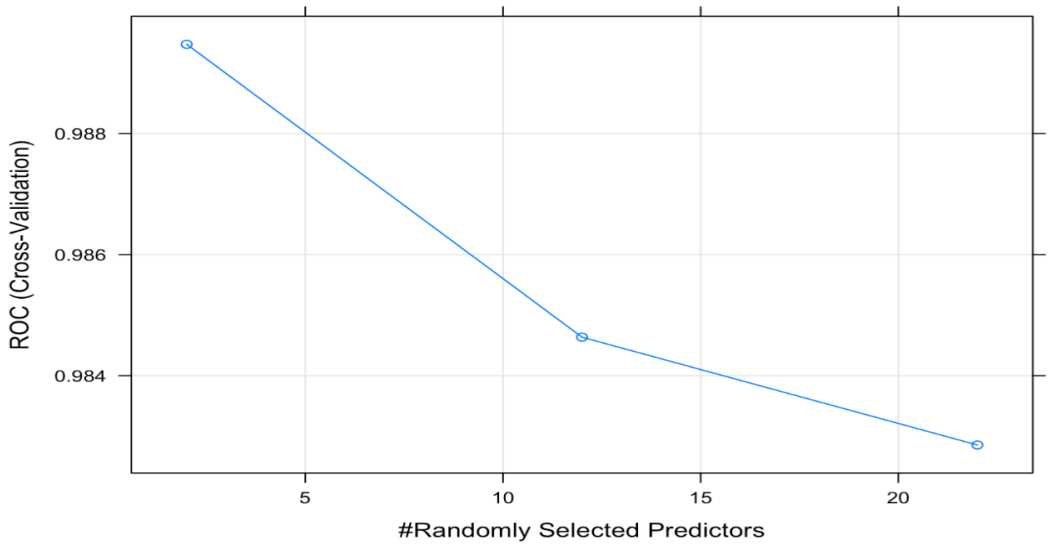
```
model_rf_df <- train(diagnosis ~., data = df_training,
                     method = "rf",
                     metric = 'ROC',
                     trControl = df_control)


prediction_rf_df <- predict(model_rf_df, df_testing)
cm_rf_df <- confusionMatrix(prediction_rf_df, df_testing$diagnosis, positive = "M")
cm_rf_df
```

```
## 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 confusion matrix is a good way of looking at how good our classifier is performing when presented with new data.

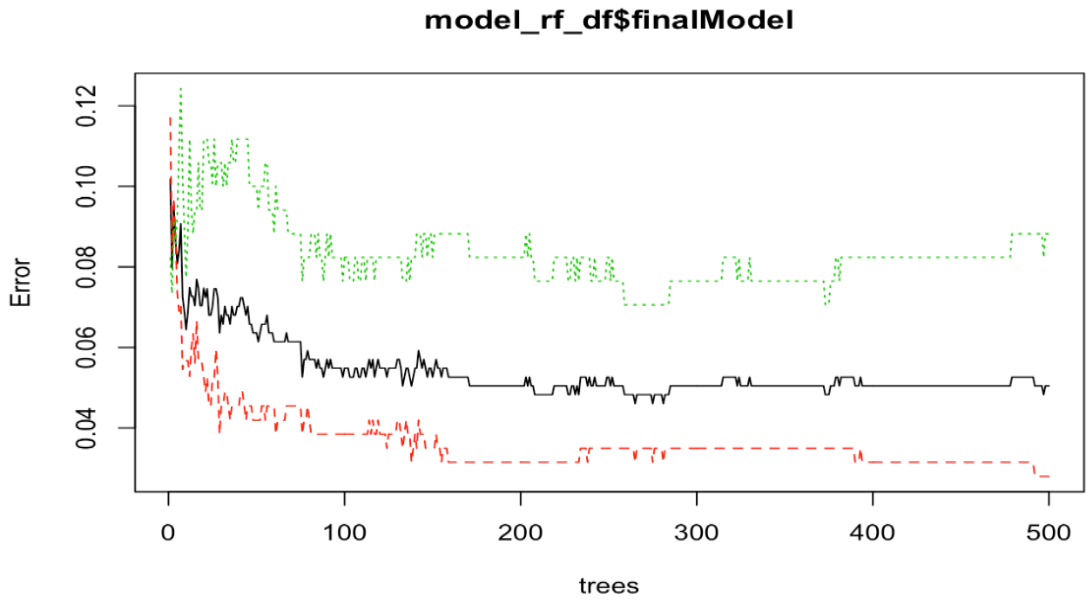
Accuracy of the Random forest model is 97.35 percent

```
plot(model_rf_df)
```



X-axis consists of the hyperparameter which is the number of randomly selected variables used at each split .
Y -axis denotes the ROC Cross validation values (Cross Validation is a technique which involves reserving a particular sample of a data set on which we do not train the model.)
From the plot we can infer that values are higher of ROC if the number of randomly selected predictors is less. We see a negative slope as the number of randomly selected predictors increase.

```
plot(model_rf_df$finalModel)
```



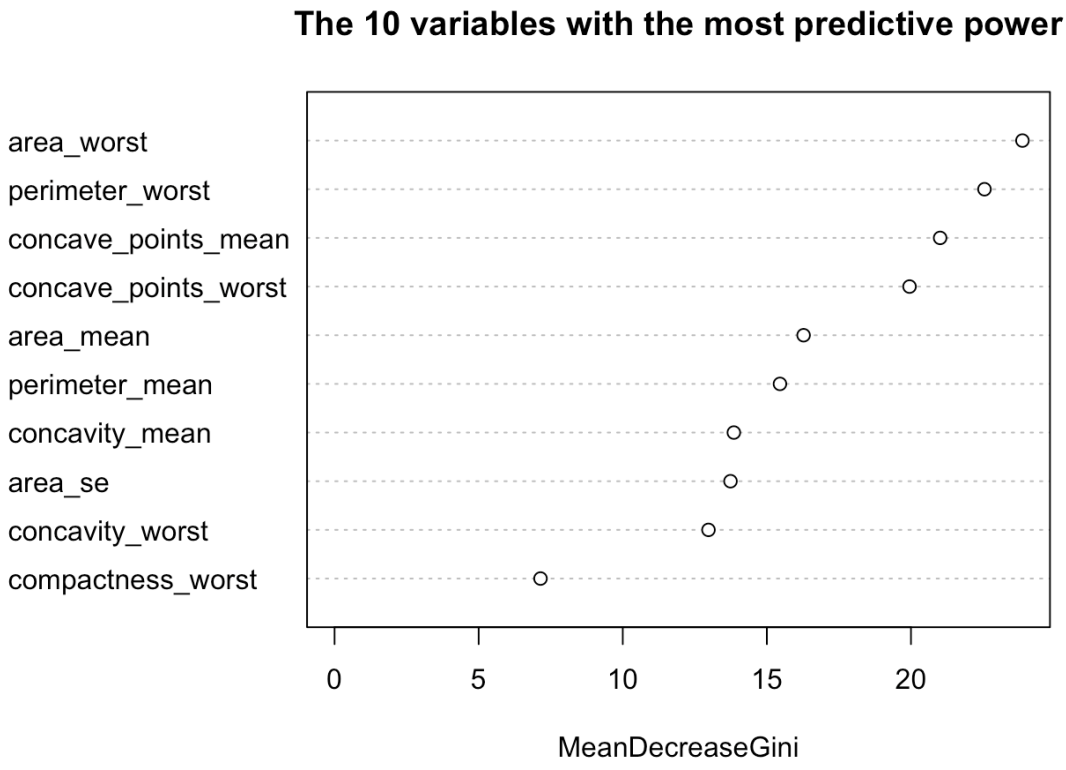
This plot depicts The test error which is displayed as a function of the number of trees.

Each coloured line corresponds to a different value, the number of predictors available for splitting at each interior tree node.

The test error is lowest around 400 trees and the error values are largely stable after 250 trees.

Now we look at a plot of Mean Decrease Gini vs variables having the most predictive power.

```
randomForest::varImpPlot(model_rf_df$finalModel, sort = TRUE,
           n.var = 10, main = "The 10 variables with the most predictive power")
```



MeanDecreaseGini : GINI is a measure of node impurity. Think of it like this, if you use this feature to split the data, how pure will the nodes be. Highest purity means that each node contains only elements of a single class. Assessing the decrease in GINI when that feature is omitted leads to an understanding of how important that feature is to split the data correctly.
From this plot we can infer that higher the value of mean decrease Gini higher the predictive power. From this plot we can infer that higher the value of mean decrease Gini higher the predictive power.

3)K-Nearest Neighbor (K-NN)

A k-nearest-neighbor algorithm, often abbreviated k-nn, is an approach to data classification that estimates how likely a data point is to be a member of one group or the other depending on what group the data points nearest to it are in.

The k-nearest-neighbor is an example of a "lazy learner" algorithm, meaning that it does not build a model using the training set until a query of the data set is performed.

A k-nearest-neighbor is a data classification algorithm that attempts to determine what group a data point is in by looking at the data points around it.

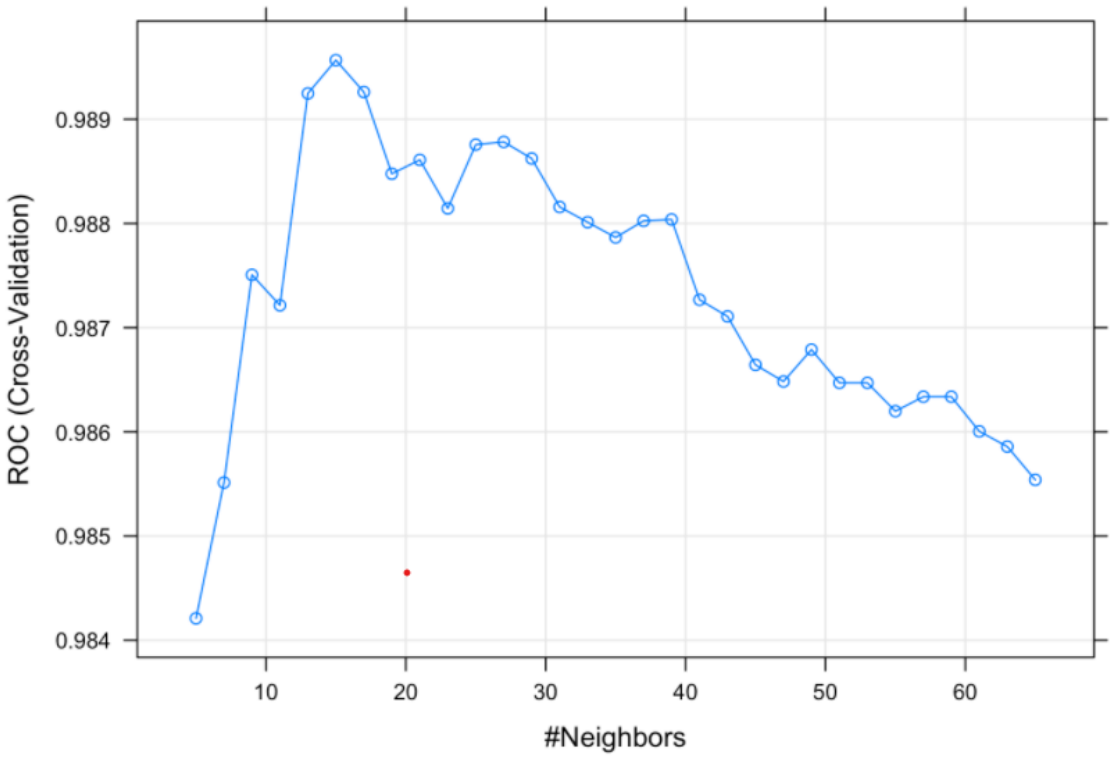
Importance of K

You can't pick any random value for k. The whole algorithm is based on the k value. Even small changes to k may result in big changes. Like most machine learning algorithms, the K in KNN is a hyperparameter. You can think of K as a controlling variable for the prediction.

Implementation

We have created a train and test folds, then we fitted a model using the train function. Here we have used the metric ROC i.e., a plot of the True Positive Rate on the Y-axis versus the False Positive Rate on the X-axis for every possible classification threshold.

```
model_knn_df <- train(diagnosis ~., data = df_training,
                      method = "knn",
                      metric = "ROC",
                      preProcess = c("scale", "center"),
                      trControl = df_control,
                      tuneLength =31)


plot(model_knn_df)
```



→ X-axis consists of the neighbours which means the different values of k.
→ Y -axis denotes the ROC Cross validation values and we have used this here because it is a technique which involves reserving a particular sample of a data set on which we do not train the model.
→ Here, we have **used ROC** to select the optimal model using the largest value. That is why **31 times** resampling the results across the **tuning parameters** has been done.

So, the **AUC of ROC** is attaining the peak value when the neighbours is 15 that means we get an optimal **K-NN model at k=15.**

```
prediction_knn_df <- predict(model_knn_df, df_testing)

cm_knn_df <- confusionMatrix(prediction_knn_df, df_testing$diagnosis, positive = "M")

cm_knn_df
```

When we get the data, after data cleaning, pre-processing and wrangling, the first step we do is to feed it to an outstanding model and of course, get output in probabilities.

For better the effectiveness, better the performance confusion matrix comes into the limelight. Confusion Matrix is a performance measurement for machine learning classification.

It is extremely useful for measuring Recall, Precision, Specificity, Accuracy and most importantly AUC-ROC Curve.
The Accuracy of **K-Nearest Neighbor(K-NN)model** is **94.69% .**

```
## 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
##
```

4)SVM (Support Vector Machines) with PCA

SVM or Support Vector Machine is a supervised learning algorithm which has the ability to produce a decision boundary (hyperplane) that can segregate an n-dimensional space into classes so that we can easily put the new data point in the correct category. SVM chooses the extreme points/vectors that help in creating the hyperplane which are known as support vectors. The hyperplane thus produced can be used for classification, regression, and other tasks like outliers detection.

Code

Before training and testing the dataset, we apply PCA to our model. The trainControl function specifies number of parameters in the model. The object outputted from trainControl is given as argument in train.

```r
set.seed(1815)
df_control_pca <- trainControl(method="cv",
                               number = 15,
                               preProcOptions = list(thresh = 0.9), # threshold for pca preprocess
                               classProbs = TRUE,
                               summaryFunction = twoClassSummary)

model_svm_pca_df <- train(diagnosis~.,
                          df_training, method = "svmLinear", metric = "ROC",
                          preProcess = c('center', 'scale', "pca"),
                          trControl = df_control_pca)

prediction_svm_pca_df <- predict(model_svm_pca_df, df_testing)
cm_svm_pca_df <- confusionMatrix(prediction_svm_pca_df, df_testing$diagnosis, positive = "M")
cm_svm_pca_df
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  B  M
##          B 70  2
##          M  1 40
##
##                Accuracy : 0.9735
##                  95% CI : (0.9244, 0.9945)
##     No Information Rate : 0.6283
##     P-Value [Acc > NIR] : <2e-16
##
##                   Kappa : 0.9429
##  Mcnemar's Test P-Value : 1
##
##             Sensitivity : 0.9524
##             Specificity : 0.9859
##          Pos Pred Value : 0.9756
##          Neg Pred Value : 0.9722
##              Prevalence : 0.3717
##          Detection Rate : 0.3540
##    Detection Prevalence : 0.3628
##       Balanced Accuracy : 0.9691
##
##        'Positive' Class : M
```

Having applied SVM with PCA, we attain the accuracy of 97.35%. Considering this as a prediction algorithm, the accuracy attained here is quite high. Other performance measures such as Sensitivity and Specificity are high as well, each of them being 95.24% and 98.59% respectively.

5)Neural Nets with LDA

A neural network is a machine learning algorithm based on the model of a human neuron. In general, Neural Network has three neurons, namely input neuron, hidden neuron, and output neuron.

Next, there are two main processes in the Neural Network that are commonly used, namely feedforward and backpropagation. Feedforward is an algorithm for calculating output values based on input values, while backpropagation is an algorithm for training neural networks (changing weights) based on errors obtained from output values.

The process for calculating the output is as follows.

i. Multiply the weights by the input of the neurons then add them up.
ii. Enter the results of the multiplication of weights by the input into the activation function which can be a sigmoid function given by –

$$\sigma = \frac{1}{1 + e^{-x}}$$

After we get the value for the output layer, the next process is to get the error value from the output layer. This error value will later become a parameter to change the weight value. To calculate errors there are also many methods such as MSE (Mean Squared Error), SSE (Sum of Squared Error), etc.

These 2 processes will continue to repeat until the minimum error value is reached.

After the training process is deemed sufficient, the model can be used for the classification process.

Training & Testing the Model

Prior to training our model we require a training and testing set. Now, as defined earlier we have used LDA for dimensionality reduction, so we will be making use of the data frame (*predict_lda_df*) obtained from there to create a training and testing set and apply Neural Network with LDA.

```
lda_training <- predict_lda_df[df_sampling_index, ]
lda_testing <- predict_lda_df[-df_sampling_index, ]
```

While training our *nnet* model we have centered and scaled our feature values for unit independency while using the ROC curve as the metric for obtaining an optimal model. We tune our model over 10 values and use the best combination to train the final model with.

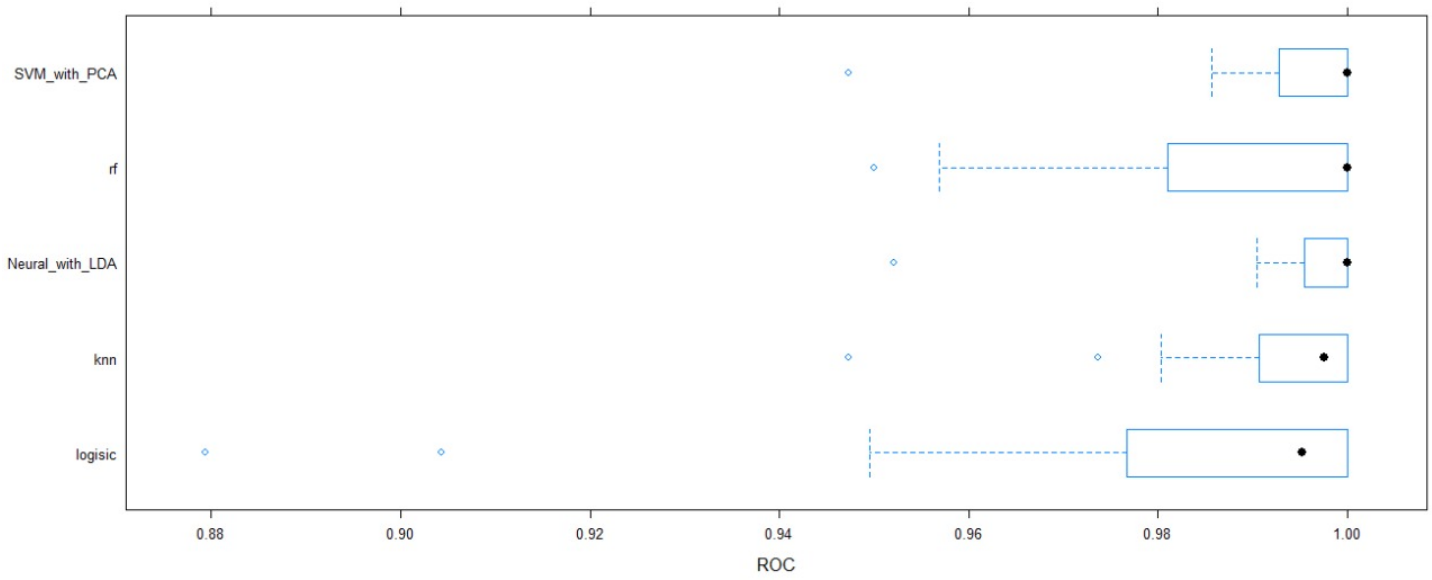
```r
model_nnetlda_df <- train(diagnosis ~., lda_training,
                          method = "nnet",
                          metric = "ROC",
                          preProcess = c("center", "scale"),
                          tuneLength = 10,
                          trace = FALSE,
                          trControl = df_control)

prediction_nnetlda_df <- predict(model_nnetlda_df, lda_testing)
cm_nnetlda_df <- confusionMatrix(prediction_nnetlda_df, lda_testing$diagnosis, positive = "M")
cm_nnetlda_df
```

Finally, after predicting on the test set, we plot the confusion matrix and statistics for performance analysis.

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  B  M
##          B 71  1
##          M  0 41
##
##                Accuracy : 0.9912
##                  95% CI : (0.9517, 0.9998)
##     No Information Rate : 0.6283
##     P-Value [Acc > NIR] : <2e-16
##
##                   Kappa : 0.981
##  Mcnemar's Test P-Value : 1
##
##             Sensitivity : 0.9762
##             Specificity : 1.0000
##          Pos Pred Value : 1.0000
##          Neg Pred Value : 0.9861
##              Prevalence : 0.3717
##          Detection Rate : 0.3628
##    Detection Prevalence : 0.3628
##       Balanced Accuracy : 0.9881
```

As can be seen, we have achieved 99.12% of accuracy with 97.62% sensitivity and 100 % specificity which is the highest among the above applied algorithms. This is because neural network can perform exceptionally well even in case of loss in data during the dimensionality reduction or pre-processing steps.
Also, even if a neuron is not responding or a piece of information is missing, the network can detect the fault and still produce the output. The use of LDA further strengthens our nnet model.

Model Evaluation




```
#Model Evaluation
model_list <- list(logisic = model_logreg_df, rf = model_rf_df, knn=model_knn_df,
                   SVM_with_PCA = model_svm_pca_df,Neural_with_LDA = model_nnetlda_df)
results <- resamples(model_list)

summary(results)

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

The logistic has to much variability for it to be reliable. The Random Forest and Neural Network with LDA pre-processing are giving the best results. The ROC metric measure **the auc of the roc curve** of each model. This metric is independent of any threshold. Let's remember how these models result with the testing dataset. Prediction classes are obtained by default with a threshold of 0.5 which could not be the best with an unbalanced dataset like this.

```
> summary(results)

Call:
summary.resamples(object = results)

Models: logisic, rf, knn, SVM_with_PCA, Neural_with_LDA
Number of resamples: 15

ROC
                    Min.   1st Qu.    Median      Mean 3rd Qu. Max. NA's
logisic         0.8793860 0.9766746 0.9952153 0.9769431       1    1    0
rf              0.9500000 0.9810606 1.0000000 0.9885859       1    1    0
knn             0.9473684 0.9906699 0.9976077 0.9912201       1    1    0
SVM_with_PCA    0.9473684 0.9928230 1.0000000 0.9933812       1    1    0
Neural_with_LDA 0.9521531 0.9954147 1.0000000 0.9949229       1    1    0

Sens
                    Min.   1st Qu.    Median      Mean 3rd Qu. Max. NA's
logisic         0.8421053 0.9473684 0.9473684 0.9580702       1    1    0
rf              0.8947368 0.9473684 0.9500000 0.9685965       1    1    0
knn             0.8947368 0.9750000 1.0000000 0.9826316       1    1    0
SVM_with_PCA    0.9473684 0.9736842 1.0000000 0.9859649       1    1    0
Neural_with_LDA 0.8947368 1.0000000 1.0000000 0.9894737       1    1    0

Spec
                    Min.   1st Qu.    Median      Mean   3rd Qu. Max. NA's
logisic         0.8181818 0.9090909 0.9166667 0.9414141 1.0000000    1    0
rf              0.5833333 0.9090909 0.9166667 0.9191919 1.0000000    1    0
knn             0.8181818 0.8257576 0.9090909 0.9050505 0.9583333    1    0
SVM_with_PCA    0.8181818 0.9128788 1.0000000 0.9409091 1.0000000    1    0
Neural_with_LDA 0.9090909 0.9090909 1.0000000 0.9580808 1.0000000    1    0
```

Comparison of Performance Measures

Sensitivity

Neural Network with LDA	**0.9762**
Logistic regression	0.9524
SVM with PCA dataset	0.9524
Random Forest	0.9286
KNN	0.8810

Accuracy

Neural Network with LDA	**0.9912**
Logistic regression	0.9823
Random Forest	0.9735
SVM with PCA dataset	0.9735
KNN	0.9469

F1- score

Neural Network with LDA	**0.988**
Logistic regression	0.976
Random Forest	0.963

Specificity

Logistic regression	**1.0000**
Random Forest	1.0000
Neural Network with LDA	1.0000
SVM with PCA dataset	0.9859
KNN	0.9859

Conclusion

Breast cancer is one of the most severe cancers. It has taken hundreds of thousands of lives every year. Early prediction of breast cancer plays an important role in successful treatment and saving the lives of thousands of patients every year. However, the conventional approaches are limited in providing such capability.

The recent breakthrough of data analytics and data mining techniques have opened a new door for healthcare diagnostic and prediction. Machine learning methods for diagnosis can significantly increase processing speed and on a big scale can make the diagnosis significantly cheaper.

This research was carried out to predict the accuracy of determining cancer at early state, after comparing five different models, **The best results for sensitivity (detection of breast cases) is LDA_NNET which also has a great F1 score.**

Code

```r
#Loading Packages
library(caret)
library(ggfortify)
library(dplyr)
library(tidyverse)
library(magrittr)

#Loading dataset
df <- read.csv("data.csv")

# This is defintely an most important step:
# Check for appropriate class on each of the variable.
glimpse(df)

# the 33 column is not right
df[,33] <- NULL

summary(df)

df$diagnosis <- as.factor(df$diagnosis)

#Missing values
map_int(df, function(.x) sum(is.na(.x)))

round(prop.table(table(df$diagnosis)), 2)

#Checking correlations
df_corr <- cor(df %>% select(-id, -diagnosis))
corrplot::corrplot(df_corr, order = "hclust", tl.cex = 1, addrect = 8)

# The findcorrelation() function from caret package remove highly correlated predictors
# based on whose correlation is above 0.9. This function uses a heuristic algorithm
# to determine which variable should be removed instead of selecting blindly
df2 <- df %>% select(-findCorrelation(df_corr, cutoff = 0.9))

ncol(df2)

#PCA analysis
preproc_pca_df <- prcomp(df %>% select(-id, -diagnosis), scale = TRUE, center = TRUE)
summary(preproc_pca_df)

# Calculate the proportion of variance explained
pca_df_var <- preproc_pca_df$sdev^2
pve_df <- pca_df_var / sum(pca_df_var)
```

```r
cum_pve <- cumsum(pve_df)
pve_table <- tibble(comp = seq(1:ncol(df %>% select(-id, -diagnosis))), pve_df, cum_pve)
pve_table

#Ploting
ggplot(pve_table, aes(x = comp, y = cum_pve)) +
  geom_point() +
  geom_abline(intercept = 0.95, color = "red", slope = 0) +
  labs(x = "Number of components", y = "Cumulative Variance")

#Most influential variables for the first 2 components
pca_df <- as_tibble(preproc_pca_df$x)
ggplot(pca_df, aes(x = PC1, y = PC2, col = df$diagnosis)) + geom_point()

autoplot(preproc_pca_df, data = df,  colour = 'diagnosis',
         loadings = FALSE, loadings.label = TRUE, loadings.colour = "blue")

#For df2
preproc_pca_df2 <- prcomp(df2, scale = TRUE, center = TRUE)
summary(preproc_pca_df2)

pca_df2_var <- preproc_pca_df2$sdev^2

# proportion of variance explained
pve_df2 <- pca_df2_var / sum(pca_df2_var)
cum_pve_df2 <- cumsum(pve_df2)
pve_table_df2 <- tibble(comp = seq(1:ncol(df2)), pve_df2, cum_pve_df2)

ggplot(pve_table_df2, aes(x = comp, y = cum_pve_df2)) +
  geom_point() +
  geom_abline(intercept = 0.95, color = "red", slope = 0) +
  labs(x = "Number of components", y = "Cumulative Variance")

#LDA
preproc_lda_df <- MASS::lda(diagnosis ~., data = df, center = TRUE, scale = TRUE)
preproc_lda_df

# Making a df out of the LDA for visualization purpose.
predict_lda_df <- predict(preproc_lda_df, df)$x %>%
  as_tibble() %>%
  cbind(diagnosis = df$diagnosis)

glimpse(predict_lda_df)

#Model the data
set.seed(1815)
df3 <- cbind(diagnosis = df$diagnosis, df2)
```

```
df_sampling_index <- createDataPartition(df3$diagnosis, times = 1, p = 0.8, list = FALSE)
df_training <- df3[df_sampling_index, ]
df_testing <-  df3[-df_sampling_index, ]
df_control <- trainControl(method="cv",
                           number = 15,
                           classProbs = TRUE,
                           summaryFunction = twoClassSummary)

#Logistic Regression
model_logreg_df <- train(diagnosis ~., data = df_training, method = "glm",
                         metric = "ROC", preProcess = c("scale", "center"),
                         trControl = df_control)

prediction_logreg_df <- predict(model_logreg_df, df_testing)
cm_logreg_df <- confusionMatrix(prediction_logreg_df, df_testing$diagnosis, positive = "M")
cm_logreg_df

#Random Forest
model_rf_df <- train(diagnosis ~., data = df_training,
                     method = "rf",
                     metric = 'ROC',
                     trControl = df_control)

prediction_rf_df <- predict(model_rf_df, df_testing)
cm_rf_df <- confusionMatrix(prediction_rf_df, df_testing$diagnosis, positive = "M")
cm_rf_df

plot(model_rf_df)

plot(model_rf_df$finalModel)

randomForest::varImpPlot(model_rf_df$finalModel, sort = TRUE,
                         n.var = 10, main = "The 10 variables with the most predictive power"
)

#KNN
model_knn_df <- train(diagnosis ~., data = df_training,
                      method = "knn",
                      metric = "ROC",
                      preProcess = c("scale", "center"),
                      trControl = df_control,
                      tuneLength =31)

plot(model_knn_df)

prediction_knn_df <- predict(model_knn_df, df_testing)
cm_knn_df <- confusionMatrix(prediction_knn_df, df_testing$diagnosis, positive = "M")
```

```
cm_knn_df

#SVM with PCA
set.seed(1815)
df_control_pca <- trainControl(method="cv",
                               number = 15,
                               preProcOptions = list(thresh = 0.9), # threshold for pca prepr
ocess
                               classProbs = TRUE,
                               summaryFunction = twoClassSummary)

model_svm_pca_df <- train(diagnosis~.,
                          df_training, method = "svmLinear", metric = "ROC",
                          preProcess = c('center', 'scale', "pca"),
                          trControl = df_control_pca)

prediction_svm_pca_df <- predict(model_svm_pca_df, df_testing)
cm_svm_pca_df <- confusionMatrix(prediction_svm_pca_df, df_testing$diagnosis, positive = "M")
cm_svm_pca_df

#Neural Network
lda_training <- predict_lda_df[df_sampling_index, ]
lda_testing <- predict_lda_df[-df_sampling_index, ]
model_nnetlda_df <- train(diagnosis ~., lda_training,
                          method = "nnet",
                          metric = "ROC",
                          preProcess = c("center", "scale"),
                          tuneLength = 10,
                          trace = FALSE,
                          trControl = df_control)

prediction_nnetlda_df <- predict(model_nnetlda_df, lda_testing)
cm_nnetlda_df <- confusionMatrix(prediction_nnetlda_df, lda_testing$diagnosis, positive = "M"
)
cm_nnetlda_df

#Model Evaluation
model_list <- list(logisic = model_logreg_df, rf = model_rf_df, knn=model_knn_df,
                   SVM_with_PCA = model_svm_pca_df,Neural_with_LDA = model_nnetlda_df)
results <- resamples(model_list)

summary(results)

bwplot(results, metric = "ROC")
#END
```

6.Scope and Limitations

Scope

The scope of the project is confined only to the prediction of breast cancer to be malignant or benign. By application of several data mining and machine learning techniques, classification will be done whether the tumor mass is benign or malignant in women. This will help in understanding the important underlying importance of attributes thereby helping in predicting the stage of breast cancer depending on the values of these attributes. Performance measures of all the algorithms will be taken into account in order to do a comparative analysis.

Limitations

- LDA limitations:
  1. Fixed K (the number of topics is fixed and must be known ahead of time).
  2. Uncorrelated topics (Dirichlet topic distribution cannot capture correlations).
  3. Non-hierarchical (in data-limited regimes hierarchical models allow sharing of data).
- PCA Limitations-Mean and covariance doesn't describe some distributions. There are many statistics distributions in which mean, and covariance doesn't give relevant information about them. In fact, mean and covariance are used (or could be considered important) for Gaussians.
- Random Forest Limitations t-It surely does a good job at classification but not as for regression problem as it does not give precise continuous nature prediction. In case of regression, it doesn't predict beyond the range in the training data, and that they may over fit data sets that are particularly noisy.
- KNN Limitations -It is Sensitive to the scale of the data and irrelevant features. Also, it requires high memory.
- Logistic regression -The major limitation of Logistic Regression is the assumption of linearity between the dependent variable and the independent variables.

7.Significance of Research

Data visualization and machine learning techniques can provide significant benefits and impact cancer detection in the decision-making process.Using comparative analysis of various algorithms we can find a model with high accuracy which means it can predict a

greater number of correct values than negatives. This research has a translational potential for women, who have abnormal mammogram findings or who have been diagnosed with breast cancer.

Finding new ways to determine the stage of metastatic breast cancer would have major clinical impact. Heat Maps ,scatter plot and box plot visualization helped to understand the correlation between each feature and brought out unnecessary features that were not essential to use while making predictions.

8.References

- Madhu Kumaria , Vijendra Singh, Breast Cancer Prediction system, International Conference on Computational Intelligence and Data Science (ICCIDS 2018), Published in Procedia Computer Science, Volume 132, 2018.
- Vivek Kumar, Brojo Kishore Mishra , Manuel Mazzara, Dang N. H. Thanh, Abhishek Verma, Prediction of Malignant & Benign Breast Cancer: A Data Mining Approach in Healthcare Applications.
- Nikita Rane, Jean Sunny, Rucha Kanade, Prof. Sulochana Devi, Breast Cancer Classification and Prediction using Machine Learning, International Journal of Engineering Research & Technology (IJERT), Published in http://www.ijert.org/, Vol. 9 Issue 02, February-2020.

- https://cran.r-project.org/web/packages/caret/index.html
- https://machinelearningmastery.com/compare-models-and-select-the-best-using-the-caret-r-package/
- https://rpubs.com/Aakansha_garg/aakansha_cancer
- https://canceratlas.cancer.org/the-burden/
- https://rstudio-pubs-static.s3.amazonaws.com/470274_250703bdc9f14d4295e86fb1e3000f5b.html
- https://www.kaggle.com/lbronchal/breast-cancer-dataset-analysis
- https://archive.ics.uci.edu/ml/datasets/Breast+Cancer+Wisconsin+(Diagnostic)