CancerDetection

Visha Arumugam(vcu526), Michael Grogan(ldl776),Sanyogita Apte(jlh562)

November 21, 2021

I - Executive Summary

To determine the model that is best suited to accurately diagnose the occurrence of breast cancer from the Fine Needle Aspiration(FNA) test results, we tested a variety of different classification algorithms with different feature selection techniques. Ultimately we came to the conclusion that neural network model with LDA dimension reduction technique is the best model to accurately diagnose the occurrence of breast cancer.

II - The Problem

The main task of this case study is to accurately diagnose the occurrence of breast cancer from the Fine Needle Aspiration(FNA) test results. Also need to find out the features which plays the significant role in diagnose the breast cancer.

The data set consist FNA test observation result of a few breast cancer patients. Fine Needle Aspiration test, is a simple,inexpensive, noninvasive and accurate technique for detecting breast cancer. Expending the suitable features of the Fine Needle Aspiration results is the most important diagnostic problem in early stages of breast cancer.

The following sections will describe in greater detail the nature of the data set and related literature as well as the methodology used to produce the predictive model to accurately diagnose the occurrence of breast cancer.

III - Review of Related Literature

There were very many prediction analysis happened with this Breast cancer data set and based upon the usage of various tools and technologies,various exploration and various prediction techniques, different analysis came with different conclusion and Recommendation.

As per the journal **“Analysis of the Wisconsin Breast Cancer Dataset and Machine Learning for Breast Cancer Detection”** by **“Borges, Lucas Rodrigues”**, two well known machine learning techniques such asBayesian Networks and J48(tree based algorithm) are tested and concluded Bayesian Networks demonstrated a good performance when dealing with imbalanced data (97.80% of accuracy).

As per the paper **"Evaluating the Performance of Machine Learning Techniques in the Classification of Wisconsin Breast Cancer** by **"Omar Ibrahim Obaid 1, Mazin Abed Mohammed2, Mohd Khanapi Abd Ghani 3, Salama A. Mostafa 4, Fahad Taha AL-Dhief 5**, three machine-learning algorithms such as Support Vector Machine, K-nearest neighbors, and Decision tree have been tested and concluded that the quadratic support vector machine grants the largest accuracy of (98.1%) with lowest false discovery rates.

In this analysis we are also going to try different classification techniques such as LDA,Logistic regression,Decision Trees, support vector Machines, Random Forest, Neural network,KNN classification with different hyper parameters and different predictors and compare the results based on the accuracy in order to identify the appropriate model to accurately diagnose the occurrence of breast cancer.

IV - Methodology

We are going to use the few prediction algorithm techniques to accurate diagnose the presence of breast cancer through the FNA test.The algorithms are as follows: Logistic Regression, Random Forest, Decision Tree, Linear Support Vector Machine,Linear Discriminant Analysis, Neural Network and Naive Bayes.

In order to get a better accuracy from the models,data then needs to be balanced for the target class, because with the unbalanced data set a classifier could achieve high accuracy by depending on the population bias in the sample. Based on the number of records and also on the distribution of observation between classes, the data set is unbalanced. A balanced training set is created by **Synthetic Data Generation**,which overcome imbalances by generates artificial data based on feature space (rather than data space) similarities from minority samples. However, after the classifier is trained, it will be tested on the unbalanced test set in order to determine how the classifier would perform under real conditions.

To simplify and improve our models, we would eliminate predictors from the model that are not significant. However when performing exploratory modeling of the training data, most of the variables were shown to be significant. Also most of the variables are correlated with each other. By Removing the highly correlated variable and also implementing Dimension reduction techniques such as PCA and LDA, implemented the classification algorithm techniques to diagnose the presence of Breast cancer.

Following is a brief summary of the classifiers we used:

Logistic Regression: Logistic Regression is a parametric classification method in which is used to model the probability of a certain class or event existing based upon the independent variables.In Logistic Regression, we don’t directly fit a straight line to our data like in linear regression. Instead, we fit a S shaped curve, called Sigmoid, to our observations.

Random Forests: Random forest is a supervised learning algorithm used for classification and regression tasks. It is distinguished from decision trees by the randomized process of finding root nodes to split features. Random forest is efficient in handling missing values. Unless a sufficient number of trees is generated to enhance prediction accuracy, the over fitting problem is a possible drawback of this algorithm.

Linear Discriminant Analysis: Linear Discriminant Analysis(LDA) is a simple and effective method for classification.It is a discriminant approach that attempts to model differences among samples assigned to certain groups. The aim of the method is to maximize the ratio of the between-group variance and the within-group variance. When the value of this ratio is at its maximum, then the samples within each group have the smallest possible scatter and the groups are separated from one another the most.LDA often produces robust, decent, and interpretable classification results.

Decision Tree: Decision Trees are a non-parametric supervised learning method used for both classification and regression tasks. The goal is to create a model that predicts the value of a target variable by learning simple decision rules inferred from the data features.The decision rules are generally in form of if-then-else statements.The deeper the tree, the more complex the rules and fitter the model

Support Vector Machines: SVM is a learning algorithm used in regression tasks. However, SVM is preferable in classification tasks. This algorithm is based on the following idea: if a classifier is effective in separating convergent non-linearly separable data points, then it should perform well on dispersed ones. SVM finds the best separating line that maximizes the distance between the hyperplanes of decision boundaries.

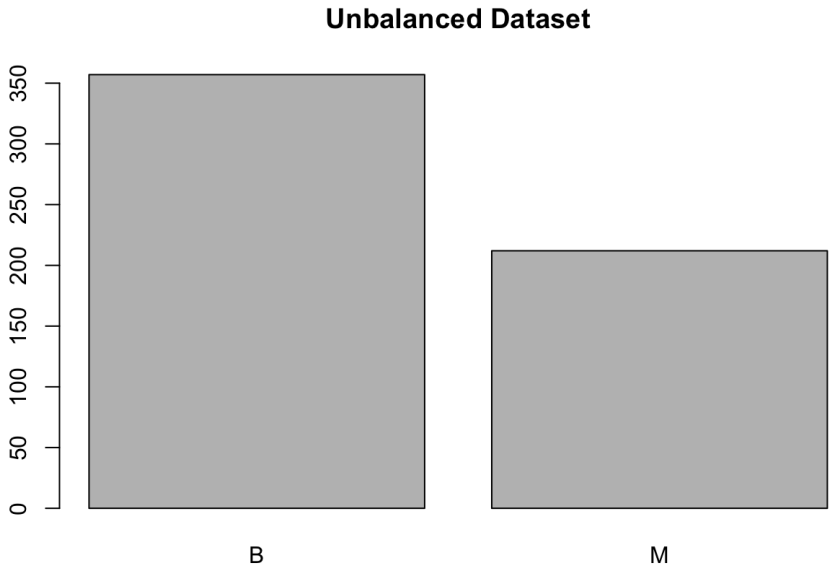
Neural Network Neural Network is a computational learning system that uses a network of functions to understand and translate a data input of one form into a desired output, usually in another form.Machine learning algorithms that use neural networks generally do not need to be programmed with specific rules that define what to expect from the input. The neural net learning algorithm instead learns from processing many labeled examples (i.e. data with with “answers”) that are supplied during training and using this answer key to learn what characteristics of the input are needed to construct the correct output. Once a sufficient number of examples have been processed, the neural network can begin to process new, unseen inputs and successfully return accurate results. The more examples and variety of inputs the program sees, the more accurate the results typically become because the program learns with experience.

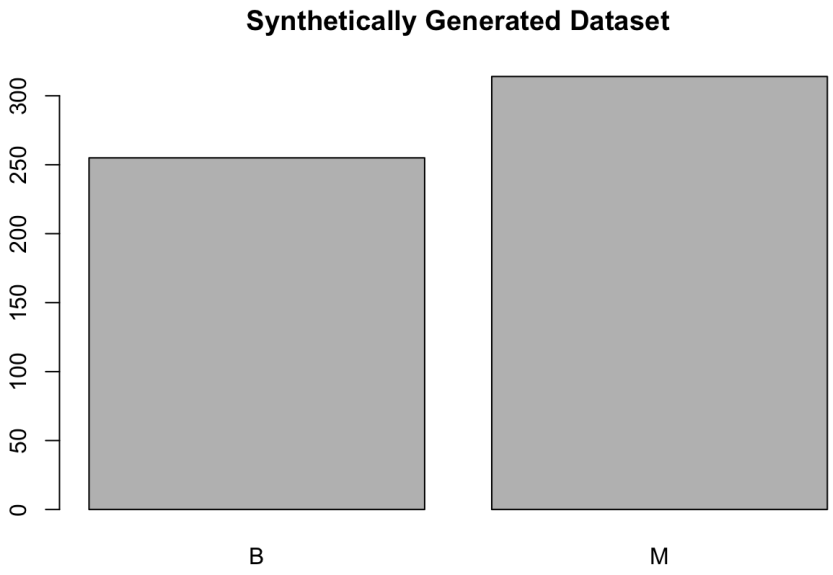
KNN Classification K-Nearest Neighbors (KNN) is one of the simplest algorithms used in Machine Learning for classification problem. KNN algorithms use data and classify new data points based on similarity measures.Classification is done by a majority vote to its neighbors. The data is assigned to the class which has the nearest neighbors. As you increase the number of nearest neighbors, the value of k, accuracy might increase.

V - Data

The data set to be used is a representative sample of the FNA results of a breast cancer patients set called “CancerData.csv”.Features in the data set are computed from a digitized image of a fine needle aspirate (FNA) of a breast mass for breast cancer patients. They describe characteristics of the cell nuclei present in the image.

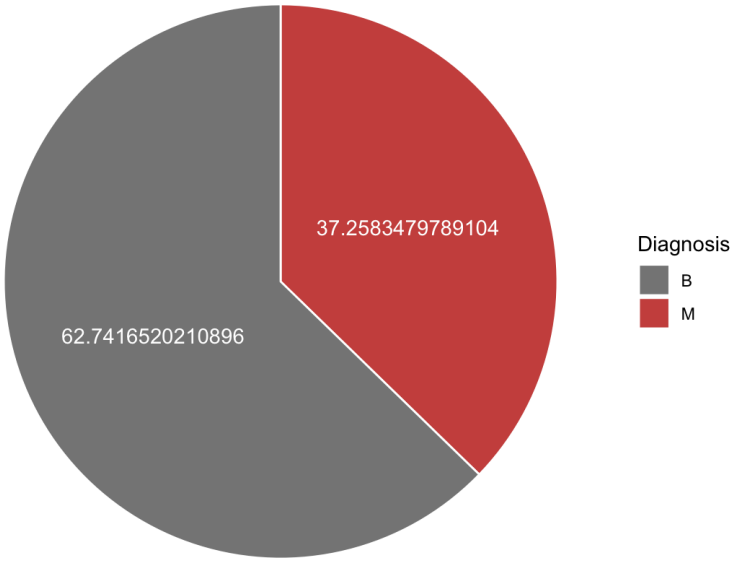
The data set doesn’t have any missing values and along the 569 records in the dataset, 357 members who are diagnosed with the absence of cancer cells and 212 members who are diagnosed with the presence of cancer cells, which ends up in a imbalanced data set. As part of this Case study we have tried Synthetically Data Generation sampling in order to increase the prediction of members who are diagnosed with the presence of cancer cells.



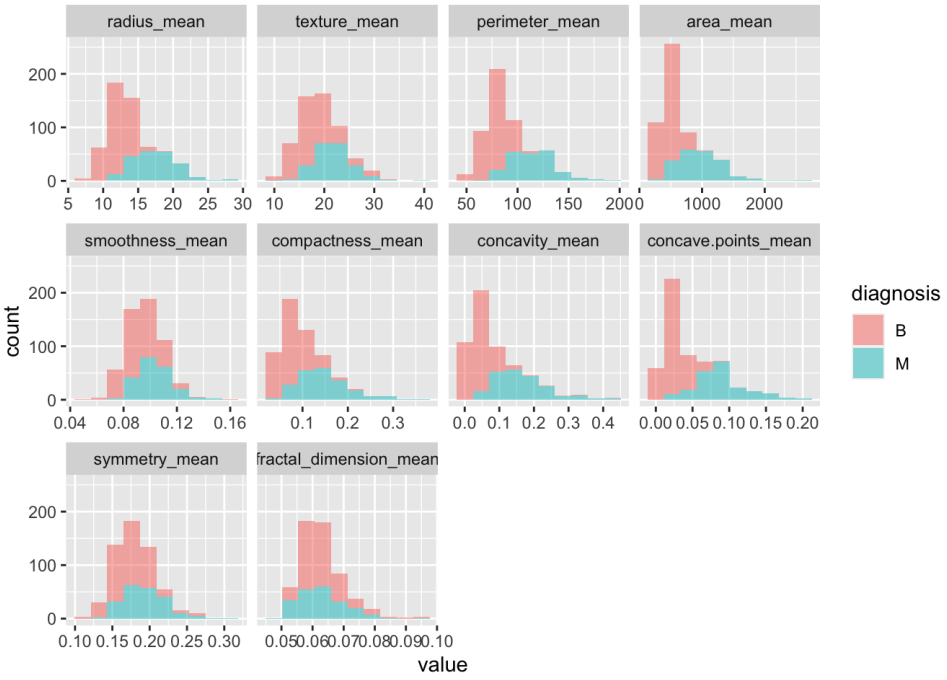


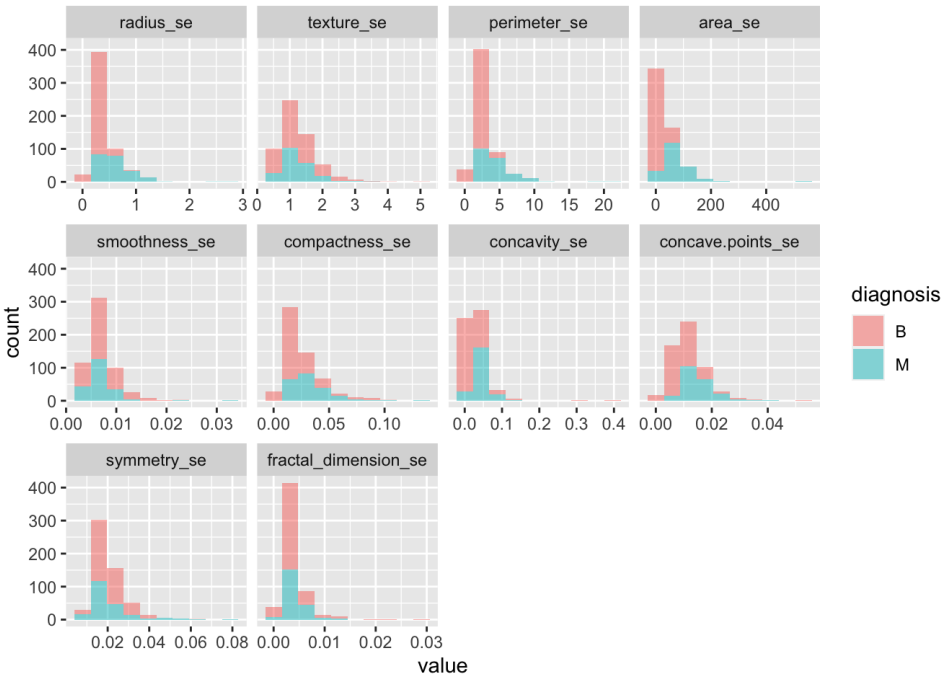
Before Going to the actual modelling using the features to predict the presence and absence of breast cancer cells, we did some exploratory analysis to visualize which features are are most helpful in predicting malignant or benign cancer. The other is to see general trends that may aid us in model selection and hyper parameter selection.

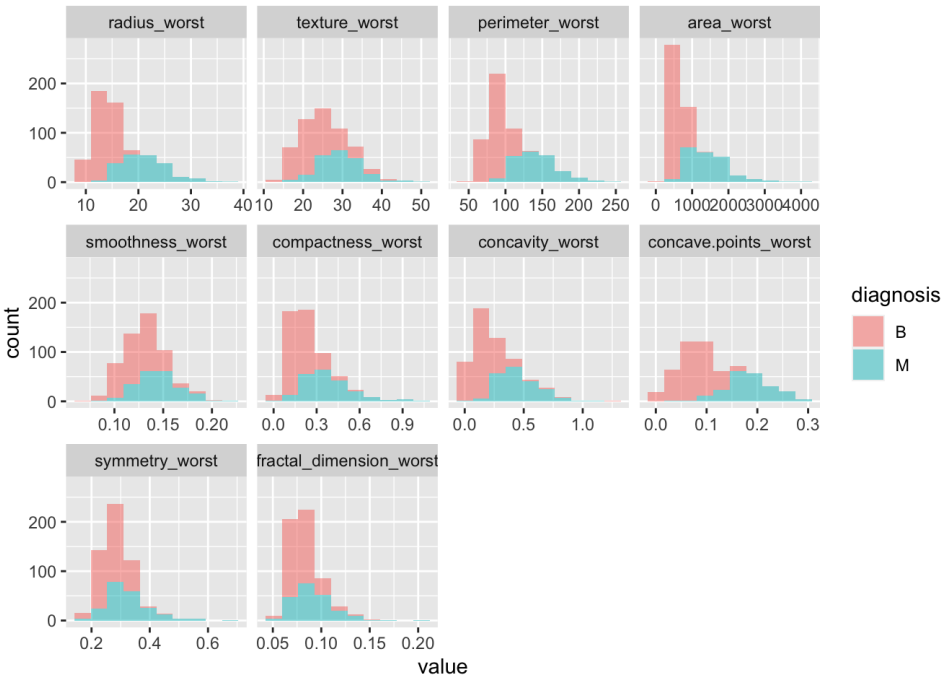
Along the total 569 observation, 357 observations which account for 62.7% of all observations indicating the absence of cancer cells, 212 which account for 37.3% of all observations shows the presence of cancerous cell.



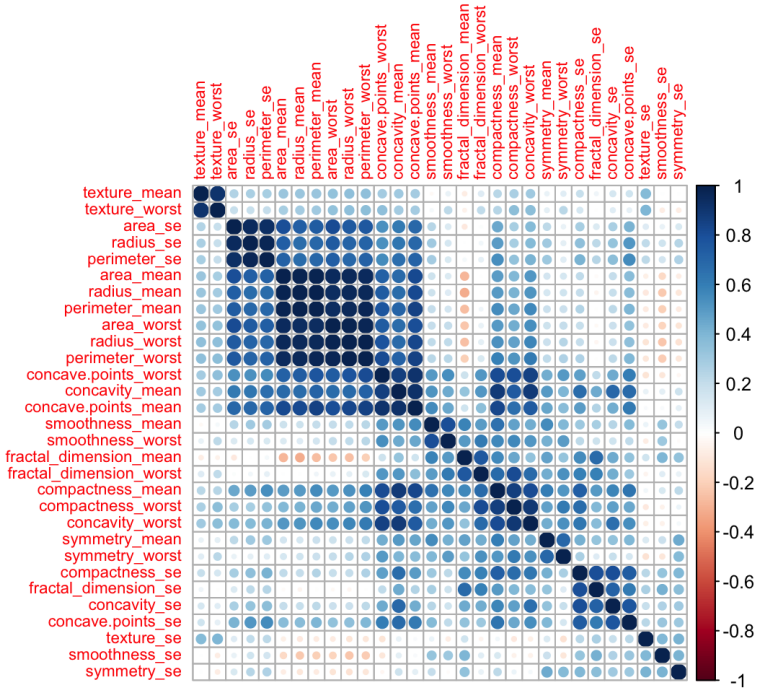
Next we will visualize the distribution of all the features corresponding to the response variable.Based on the below visualization we can infer that most of the variables are normally distributed with respect to the response variable.


radius_mean
texture_mean
perimeter_mean
area_mean
smoothness_mean
compactness_mean
concavity_mean
concave.points_mean
symmetry_mean
fractal_dimension_mean
count
value
diagnosis
B
M


radius_se
texture_se
perimeter_se
area_se
smoothness_se
compactness_se
concavity_se
concave.points_se
symmetry_se
fractal_dimension_se
count
value
diagnosis
B
M


radius_worst
texture_worst
perimeter_worst
area_worst
smoothness_worst
compactness_worst
concavity_worst
concave.points_worst
symmetry_worst
fractal_dimension_worst
count
value
diagnosis
B
M

Next we will whether there exists any correlation between the features.As per the correlation plot, we can say that most of the variables are correlated with each each. Very few or not.



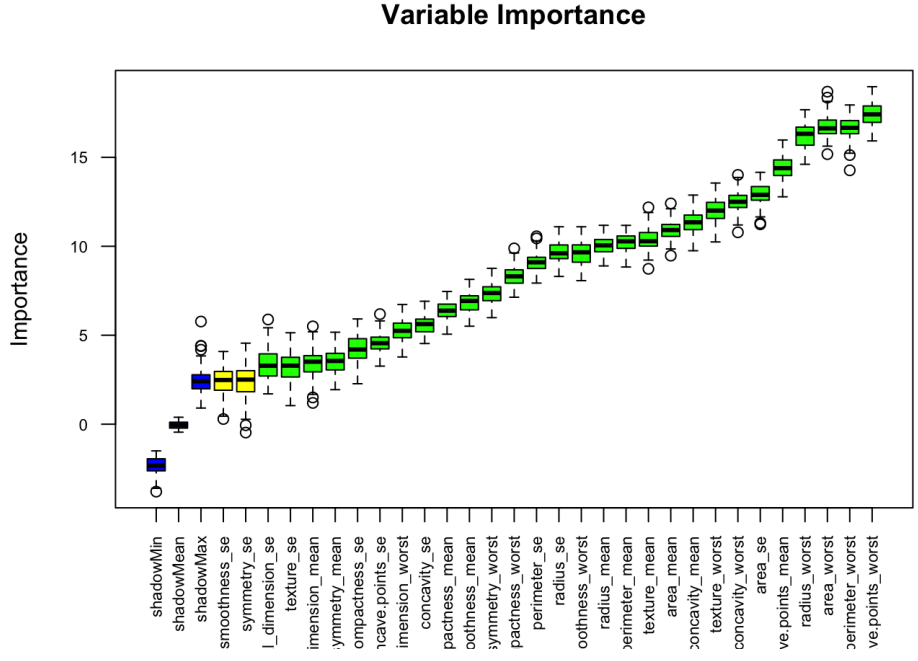
Since most of the variables are highly correlated, we have removed the variable with correlation score greater than 0.9 as below.

```
##  [1] "smoothness_mean"  "compactness_mean" "radius_worst"     "symmetry_se"
##  [5] "radius_mean"      "texture_worst"    "id"               "radius_se"
##  [9] "texture_se"       "diagnosis"
```

Also we have tried feature selection and dimension reduction technique to find out the significant variables which plays a major role in classifying the members with presence or absence of cancer cells.

Variable selection Using full Logistic Regression

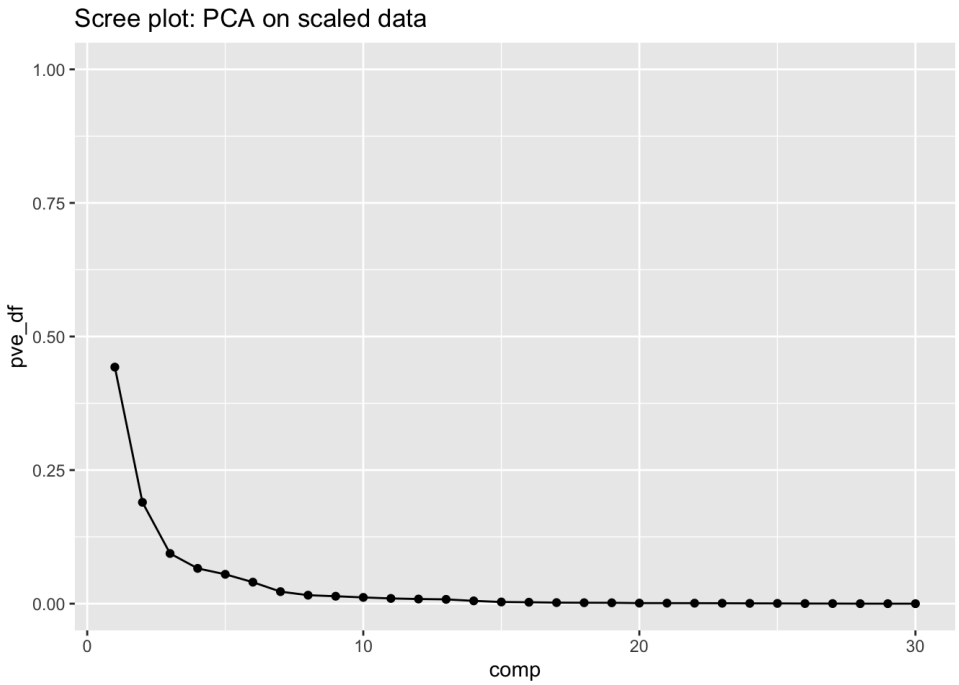
```
##    Importance  Variable_Name
## 29   72.92305 symmetry_worst
## 19   70.05453    symmetry_se
## 15   61.21310  smoothness_se
## 3    59.79138 perimeter_mean
## 9    52.09289  symmetry_mean
## 13   36.03208   perimeter_se
## 14   34.83993        area_se
## 4    33.30130      area_mean
## 24   32.65753     area_worst
## 12   31.76303     texture_se
```



As per the above feature selection methods almost all of the variables plays a significant role in predicting the presence or absence of Brest cancer.

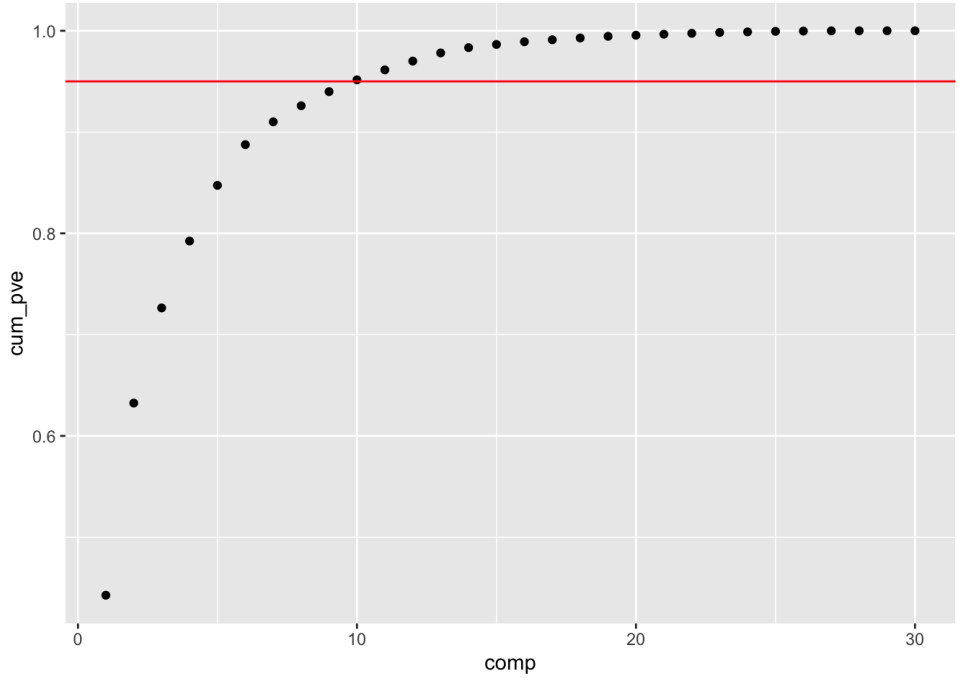
Since the correlation is very high in most parameters we have tried PCA and LDA to find out the number of variables associated in predicting the response.

PCA including all Variables



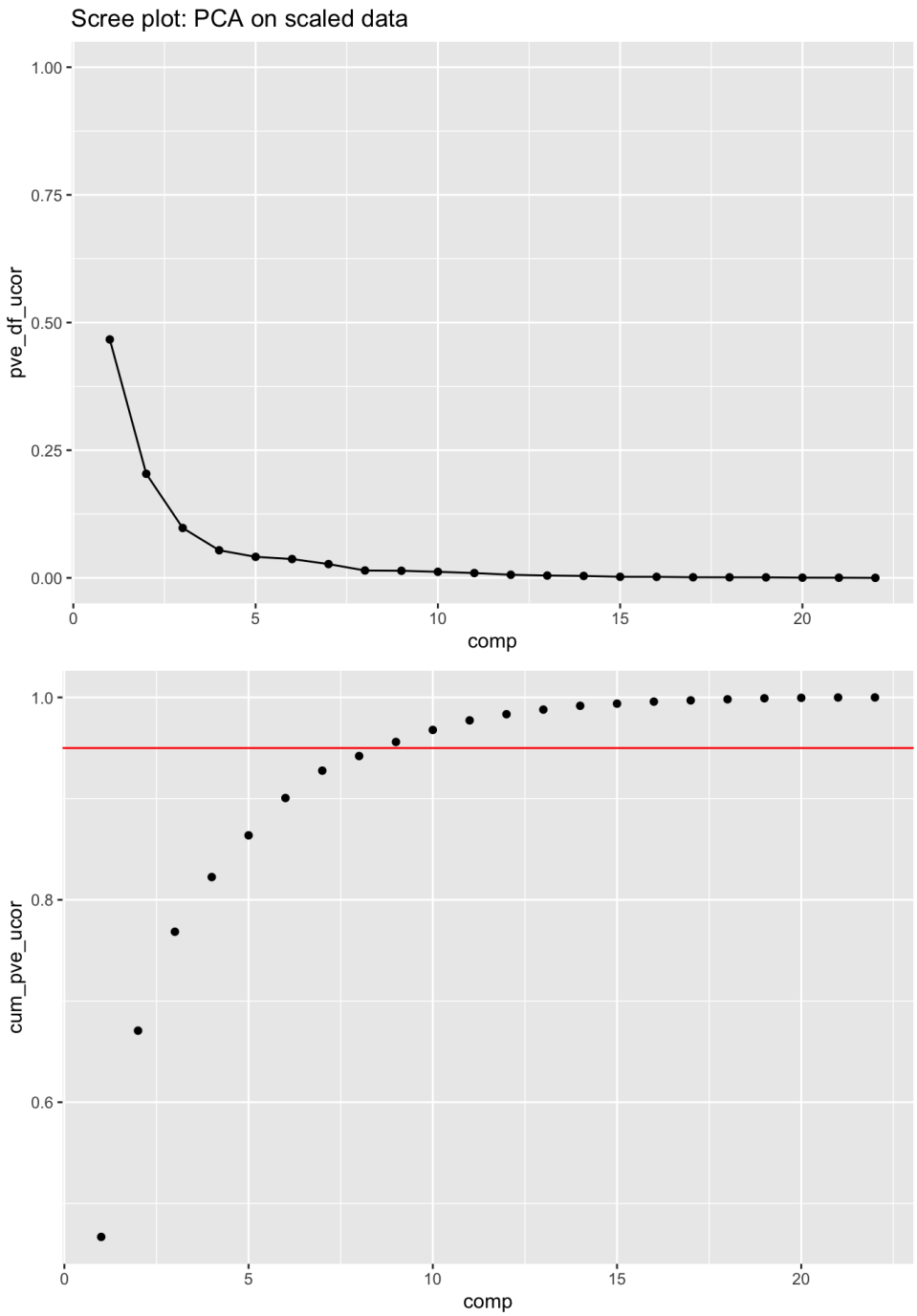
Based on the above plot , the first 7 component explained all the variance. After the Component 8 the varaince explained by the variable is minimum and looks the same.

```
ggplot(pve_table, aes(x = comp, y = cum_pve)) +
  geom_point() +
  geom_abline(intercept = 0.95, color = "red", slope = 0)
```

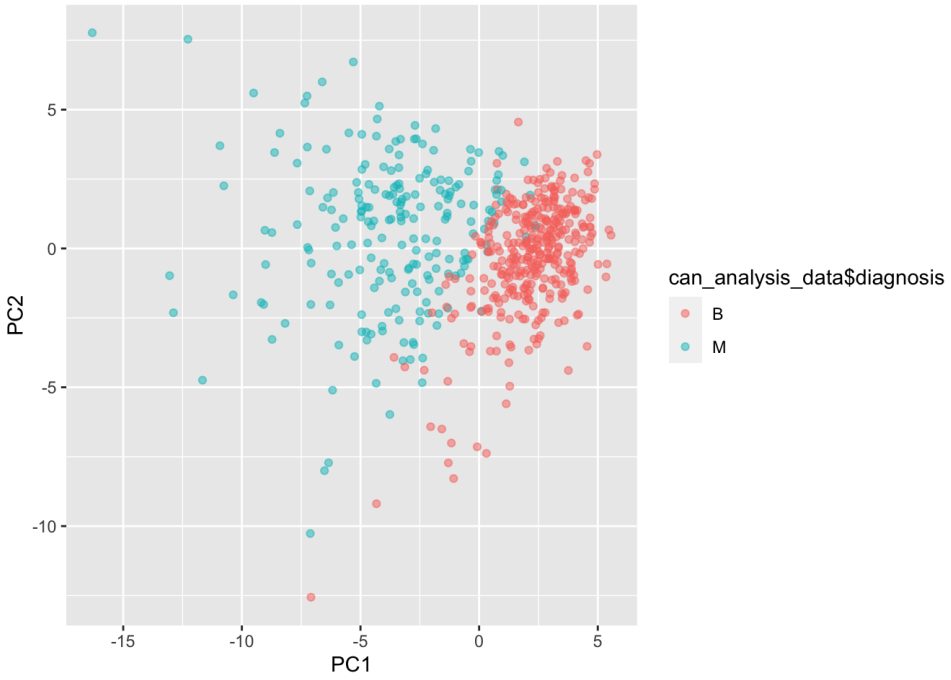


Based on the above plot , the first 10 component explained 95% variance. Top 15-17 component explained 99% of variance.

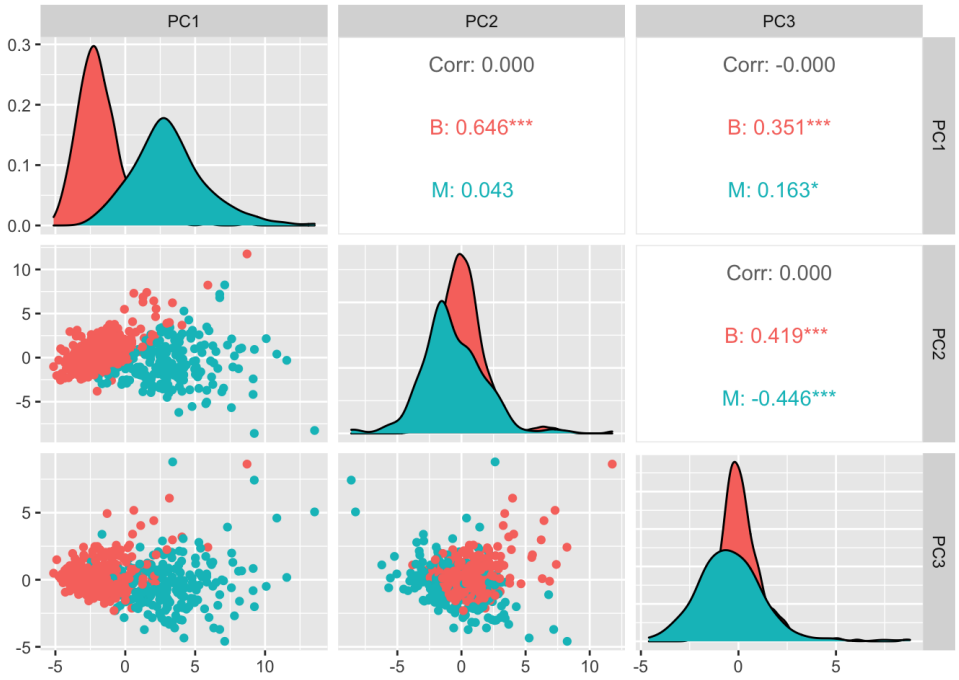
PCA including only uncorrelated variables



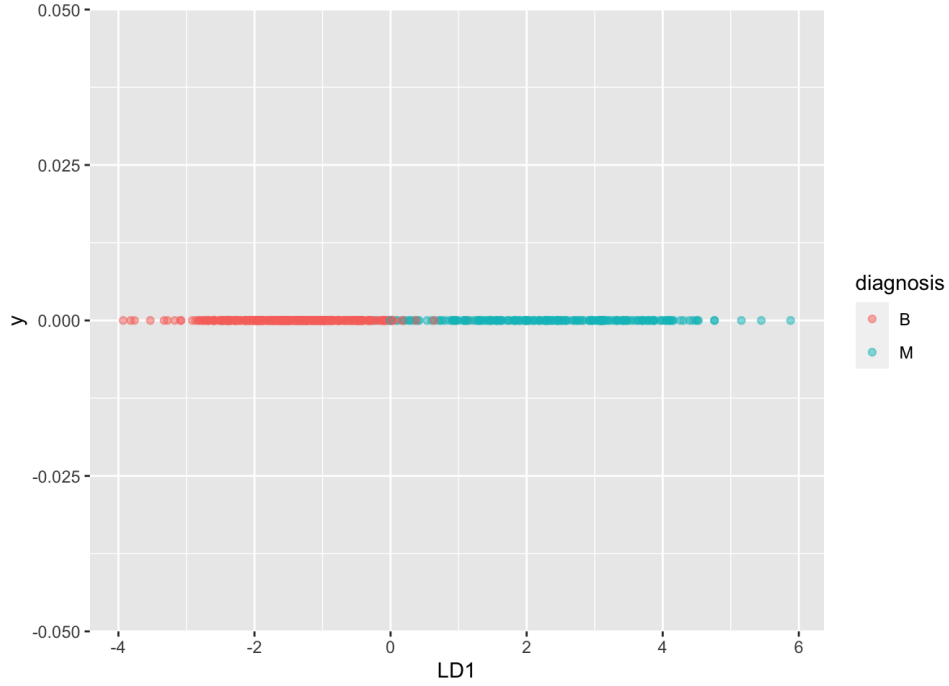
As it can be seen from the below plots the first 3 principal components separate the two classes to some extent only, this is expected since the variance explained by these components is not large. so while modelling we will use .99 as threshold for the PCA.

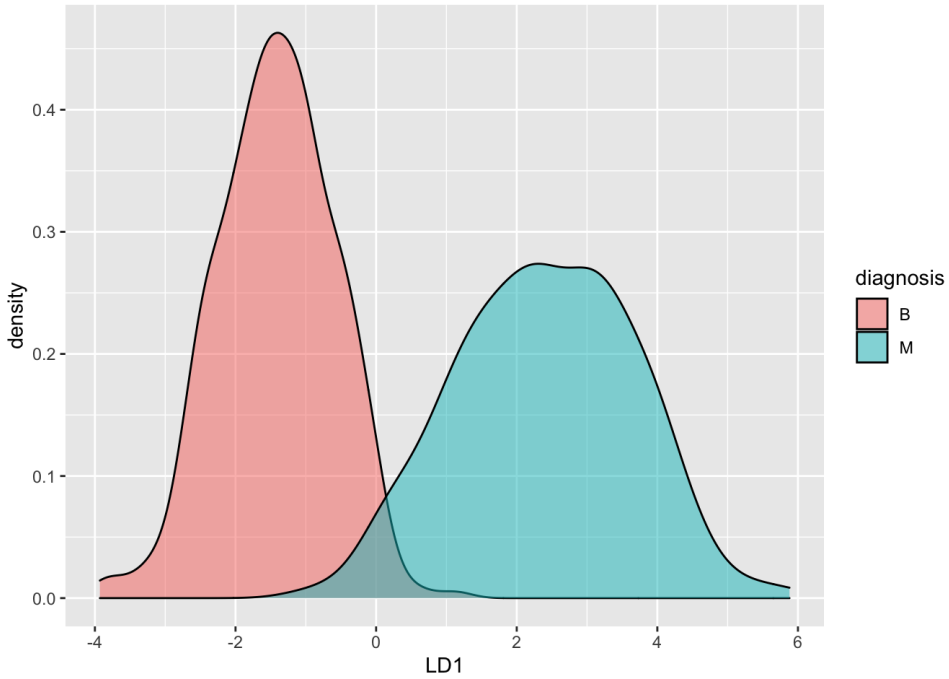



```
## Registered S3 method overwritten by 'GGally':
##   method from
##   +.gg   ggplot2
```



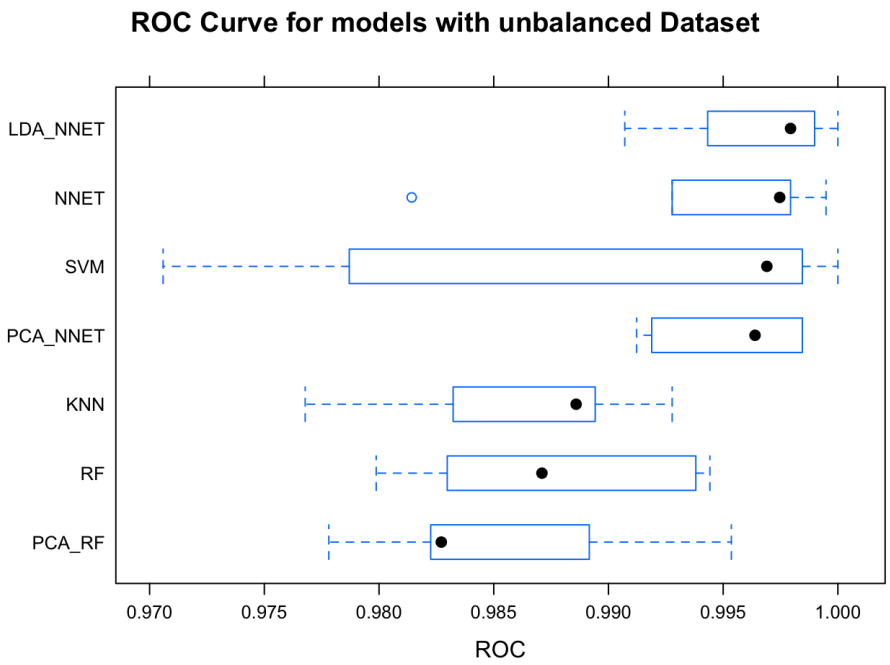
Whereas LDA separates the class more effectively than PCA as per the below Plots.





VI - Findings

Model Comparision for UnBalanced Dataset

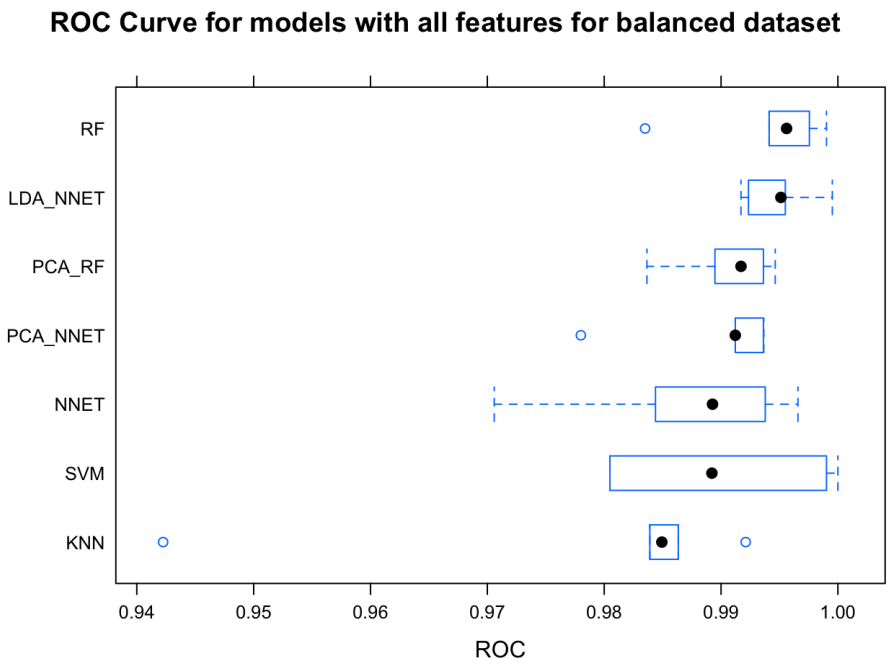


As per the ROC , SVM and Rf model shows high variability whereas LDA_NNET(Neural network with LDA) has the highest AUC.

Model Comparision using Accuracy

```
##                           RF    PCA_RF      NNET  PCA_NNET  LDA_NNET
## Sensitivity          0.9761905 0.9761905 0.9761905 0.9761905 1.0000000
## Specificity          0.9859155 0.9859155 0.9718310 0.9718310 0.9859155
## Pos Pred Value       0.9761905 0.9761905 0.9534884 0.9534884 0.9767442
## Neg Pred Value       0.9859155 0.9859155 0.9857143 0.9857143 1.0000000
## Precision            0.9761905 0.9761905 0.9534884 0.9534884 0.9767442
## Recall               0.9761905 0.9761905 0.9761905 0.9761905 1.0000000
## F1                   0.9761905 0.9761905 0.9647059 0.9647059 0.9882353
## Prevalence           0.3716814 0.3716814 0.3716814 0.3716814 0.3716814
## Detection Rate       0.3628319 0.3628319 0.3628319 0.3628319 0.3716814
## Detection Prevalence 0.3716814 0.3716814 0.3805310 0.3805310 0.3805310
## Balanced Accuracy    0.9810530 0.9810530 0.9740107 0.9740107 0.9929577
##                           KNN       SVM        DT
## Sensitivity          0.9047619 1.0000000 0.9761905
## Specificity          1.0000000 0.9718310 0.9859155
## Pos Pred Value       1.0000000 0.9545455 0.9761905
## Neg Pred Value       0.9466667 1.0000000 0.9859155
## Precision            1.0000000 0.9545455 0.9761905
## Recall               0.9047619 1.0000000 0.9761905
## F1                   0.9500000 0.9767442 0.9761905
## Prevalence           0.3716814 0.3716814 0.3716814
## Detection Rate       0.3362832 0.3716814 0.3628319
## Detection Prevalence 0.3362832 0.3893805 0.3716814
## Balanced Accuracy    0.9523810 0.9859155 0.9810530
```

Model Comparision for Balanced Dataset



As per the ROC for the balanced data set, SVM model shows high variability whereas RF(Random Forest) has the highest AUC.

```
##                            RF    PCA_RF      NNET  PCA_NNET  LDA_NNET
## Sensitivity          0.9047619 0.9047619 0.9285714 0.9285714 1.0000000
## Specificity          0.9859155 0.9859155 1.0000000 1.0000000 0.9859155
## Pos Pred Value       0.9743590 0.9743590 1.0000000 1.0000000 0.9767442
## Neg Pred Value       0.9459459 0.9459459 0.9594595 0.9594595 1.0000000
## Precision            0.9743590 0.9743590 1.0000000 1.0000000 0.9767442
## Recall               0.9047619 0.9047619 0.9285714 0.9285714 1.0000000
## F1                   0.9382716 0.9382716 0.9629630 0.9629630 0.9882353
## Prevalence           0.3716814 0.3716814 0.3716814 0.3716814 0.3716814
## Detection Rate       0.3362832 0.3362832 0.3451327 0.3451327 0.3716814
## Detection Prevalence 0.3451327 0.3451327 0.3451327 0.3451327 0.3805310
## Balanced Accuracy    0.9453387 0.9453387 0.9642857 0.9642857 0.9929577
##                           KNN       SVM        DT
## Sensitivity          0.7142857 0.9047619 0.9047619
## Specificity          1.0000000 0.9859155 0.9859155
## Pos Pred Value       1.0000000 0.9743590 0.9743590
## Neg Pred Value       0.8554217 0.9459459 0.9459459
## Precision            1.0000000 0.9743590 0.9743590
## Recall               0.7142857 0.9047619 0.9047619
## F1                   0.8333333 0.9382716 0.9382716
## Prevalence           0.3716814 0.3716814 0.3716814
## Detection Rate       0.2654867 0.3362832 0.3362832
## Detection Prevalence 0.2654867 0.3451327 0.3451327
## Balanced Accuracy    0.8571429 0.9453387 0.9453387
```

VII - Conclusions

As per the Pre-processing, the features aligned with PCA and After removing the higher correlations represents the same. Also when compared to PCA, LDA separates the classification very well. So for comparing the model we have used both PCA with the threshold of 99 and LDA.

Based on the ROC curve the RF model for the Balanced dataset shows the highest AUC which makes the Random forest model as a Best model

Whereas based on the comparison of other metrics parameter for the unbalanced and balanced data set model as below, we have concluded that the Neural network with LDA parameters have the best **accuracy (99%)** with the best **f1 score as 98%**in predicting the occurrence of cancer cells or not.

Unbalanced Dataset Model report

```
##                  metric best_model     value
## 1           Sensitivity   LDA_NNET 1.0000000
## 2           Specificity        KNN 1.0000000
## 3        Pos Pred Value        KNN 1.0000000
## 4        Neg Pred Value   LDA_NNET 1.0000000
## 5             Precision        KNN 1.0000000
## 6                Recall   LDA_NNET 1.0000000
## 7                    F1   LDA_NNET 0.9882353
## 8            Prevalence   LDA_NNET 0.3716814
## 9        Detection Rate   LDA_NNET 0.3716814
## 10 Detection Prevalence        SVM 0.3893805
## 11    Balanced Accuracy   LDA_NNET 0.9929577
```

Balanced Dataset Model report

```
##                  metric best_model     value
## 1           Sensitivity   LDA_NNET 1.0000000
## 2           Specificity       NNET 1.0000000
## 3        Pos Pred Value   PCA_NNET 1.0000000
## 4        Neg Pred Value   LDA_NNET 1.0000000
## 5             Precision   PCA_NNET 1.0000000
## 6                Recall   LDA_NNET 1.0000000
## 7                    F1   LDA_NNET 0.9882353
## 8            Prevalence       NNET 0.3716814
## 9        Detection Rate   LDA_NNET 0.3716814
## 10 Detection Prevalence   LDA_NNET 0.3805310
## 11    Balanced Accuracy   LDA_NNET 0.9929577
```

Appendix

Preprocessing the Data

```
set.seed(12345)
cancer_data<-read.csv("CancerData.csv")

# Display the structure of the dataset
str(cancer_data)

# Convert the response variable to a factor
cancer_data$diagnosis=as.factor(cancer_data$diagnosis)

# Remove the ID variable from the dataset which doesn't have useful information for classification.
can_analysis_data=cancer_data[,-c(0:1)]

# Display the summary of the dataset
summary(can_analysis_data)

#Check for Missing values
sapply(can_analysis_data, function(x) sum(is.na(x)))

#Split the Data Set into Train and Test Dataset
inTrain <- createDataPartition(can_analysis_data$diagnosis, p=0.8, list=FALSE)
train_can=can_analysis_data[inTrain,]
test_can=can_analysis_data[-inTrain,]

# create the Balanced dataset
can_analysis_data_rose=ROSE(diagnosis~.,data=can_analysis_data,seed=12345)$data
train_can_rose=ROSE(diagnosis~.,data=train_can,seed=12345)$data
```

```
# frequency of the cancer distribution
diag.table = table(cancer_data$diagnosis)

# Create a pie chart
diag.prop.table <- prop.table(diag.table)*100
diag.prop.df <- as.data.frame(diag.prop.table)
colnames(diag.prop.df)=c("Diagnosis","Proportion")

diag.prop.df <- diag.prop.df %>%
  arrange(desc(Diagnosis)) %>%
  mutate(lab.ypos = cumsum(Proportion) - 0.5*Proportion)

data_mean <- cancer_data[ ,c("diagnosis", "radius_mean", "texture_mean","perimeter_mean", "area_mean", "smoothnes
s_mean", "compactness_mean", "concavity_mean", "concave.points_mean", "symmetry_mean", "fractal_dimension_mean"
 )]

data_se <- cancer_data[ ,c("diagnosis", "radius_se", "texture_se","perimeter_se", "area_se", "smoothness_se", "co
mpactness_se", "concavity_se", "concave.points_se", "symmetry_se", "fractal_dimension_se" )]

data_worst <- cancer_data[ ,c("diagnosis", "radius_worst", "texture_worst","perimeter_worst", "area_worst", "smoo
thness_worst", "compactness_worst", "concavity_worst", "concave.points_worst", "symmetry_worst", "fractal_dimensi
on_worst" )]

corMatMy <- cor(can_analysis_data[,2:31])
highlyCor <- colnames(cancer_data)[findCorrelation(corMatMy, cutoff = 0.9, verbose = TRUE)]
cancer_data_uncor=can_analysis_data[,which(!colnames(can_analysis_data) %in% highlyCor )]
```

Feature Selection

```
log_model=glm(diagnosis~.,data=can_analysis_data,family = "binomial")
#summary(log_model)
stepped<-stepAIC(log_model, direction = "both",trace = FALSE)

idx <- order(coef(summary(stepped))[,4])
out <- coef(summary(stepped))[idx,]

imp <- as.data.frame(varImp(log_model))
imp <- data.frame(Importance = imp$Overall,
           Variable_Name   = rownames(imp))

boruta_var_imp_output=Boruta(diagnosis~.,data=can_analysis_data,doTrace=1)
```

Dimensionality Reduction Techniques

```
## PCA with correlated variables
cancer_pca <- prcomp(can_analysis_data[, 2:31], center=TRUE, scale=TRUE)
#summary(cancer_pca)
pca_var <- cancer_pca$sdev^2
pve_df <- pca_var / sum(pca_var)
cum_pve <- cumsum(pve_df)
pve_table <- tibble(comp = seq(1:30), pve_df, cum_pve)
## PCA with uncorrelated variables.
cancer_pca_ucor <- prcomp(cancer_data_uncor, center=TRUE, scale=TRUE)
#summary(cancer_pca_ucor)
pca_var_ucor <- cancer_pca_ucor$sdev^2
pve_df_ucor <- pca_var_ucor / sum(pca_var_ucor)
cum_pve_ucor <- cumsum(pve_df_ucor)
pve_table_ucor <- tibble(comp = seq(1:ncol(cancer_data_uncor)), pve_df_ucor, cum_pve_ucor)
pca_cor_df=as.data.frame(cancer_pca$x)
df_pcs <- cbind(as_tibble(can_analysis_data$diagnosis), as_tibble(cancer_pca_ucor$x))
#head(pca_cor_df)

lda_res <- lda(diagnosis~.,can_analysis_data, center = TRUE, scale = TRUE)
lda_df <- predict(lda_res, can_analysis_data)$x %>% as.data.frame() %>% cbind(diagnosis=can_analysis_data$diagnos
is)
train_can_lda <- lda_df[inTrain,]
test_can_lda <- lda_df[-inTrain,]
train_can_lda_rose=ROSE(diagnosis~.,data=train_can_lda,seed=12345)$data
```

Various Modelling Techniques

```
fitControl <- trainControl(method="cv",
                           number = 5,
                           preProcOptions = list(thresh = 0.99), # threshold for pca preprocess
                           classProbs = TRUE,
                           summaryFunction = twoClassSummary)

# Random forest
model_rf <- train(diagnosis~.,
                  data = train_can,
                  method="rf",
                  metric="ROC",
                  preProcess = c('center', 'scale'),
                  trControl=fitControl)
pred_rf <- predict(model_rf, test_can)
cm_rf <- confusionMatrix(pred_rf, test_can$diagnosis, positive = "M")
# Random forest with pca
model_pca_rf <- train(diagnosis~.,
                  data = train_can,
                  method="rf",
                  metric="ROC",
                  preProcess = c('center', 'scale','pca'),
                  trControl=fitControl)
pred_pca_rf <- predict(model_rf, test_can)
cm_pca_rf <- confusionMatrix(pred_pca_rf, test_can$diagnosis, positive = "M")
# Knn
model_knn <- train(diagnosis~.,
                   data = train_can,
                   method="knn",
                   metric="ROC",
                   preProcess = c('center', 'scale'),
                   tuneLength=10,
                   trControl=fitControl)

pred_knn <- predict(model_knn, test_can)
cm_knn <- confusionMatrix(pred_knn, test_can$diagnosis, positive = "M")
## Neuarl Network
model_nnet <- train(diagnosis~.,
                    data = train_can,
                    method="nnet",
                    metric="ROC",
                    preProcess=c('center', 'scale'),
                    trace=FALSE,
                    tuneLength=10,
                    trControl=fitControl)
pred_nnet <- predict(model_nnet, test_can)
cm_nnet <- confusionMatrix(pred_nnet, test_can$diagnosis, positive = "M")
## Neural network with pca
model_pca_nnet <- train(diagnosis~.,
                    data = train_can,
                    method="nnet",
                    metric="ROC",
                    preProcess=c('center', 'scale','pca'),
                    trace=FALSE,
                    tuneLength=10,
                    trControl=fitControl)
pred_pca_nnet <- predict(model_nnet, test_can)
cm_pca_nnet <- confusionMatrix(pred_pca_nnet, test_can$diagnosis, positive = "M")

## Neural network with lda
model_lda_nnet <- train(diagnosis~.,
                    data = train_can_lda,
                    method="nnet",
                    metric="ROC",
                    preProcess=c('center', 'scale'),
                    trace=FALSE,
                    tuneLength=10,
                    trControl=fitControl)
pred_lda_nnet <- predict(model_lda_nnet, test_can_lda)
cm_lda_nnet <- confusionMatrix(pred_lda_nnet, test_can_lda$diagnosis, positive = "M")
# Decision Tree
model_dt <- train(diagnosis~.,
                  data = train_can,
                  method="rpart",
                  metric="ROC",
                  preProcess = c('center', 'scale'),
                  trControl=fitControl)
pred_dt <- predict(model_rf, test_can)
cm_dt <- confusionMatrix(pred_dt, test_can$diagnosis, positive = "M")
#SVM kernel
model_svm <- train(diagnosis~.,
```

```
                    data = train_can,
                    method="svmRadial",
                    metric="ROC",
                    preProcess=c('center', 'scale'),
                    trace=FALSE,
                    trControl=fitControl)
pred_svm <- predict(model_svm, test_can)
cm_svm <- confusionMatrix(pred_svm, test_can$diagnosis, positive = "M")
```

```
# Random forest
model_rf_bal <- train(diagnosis~.,
                  data = train_can_rose,
                  method="rf",
                  metric="ROC",
                  preProcess = c('center', 'scale'),
                  trControl=fitControl)
pred_rf_bal <- predict(model_rf_bal, test_can)
cm_rf_bal <- confusionMatrix(pred_rf_bal, test_can$diagnosis, positive = "M")
# Random forest with pca
model_pca_rf_bal <- train(diagnosis~.,
                  data = train_can_rose,
                  method="rf",
                  metric="ROC",
                  preProcess = c('center', 'scale','pca'),
                  trControl=fitControl)
pred_pca_rf_bal <- predict(model_rf_bal, test_can)
cm_pca_rf_bal <- confusionMatrix(pred_pca_rf_bal, test_can$diagnosis, positive = "M")
# Knn
model_knn_bal <- train(diagnosis~.,
                   data = train_can_rose,
                   method="knn",
                   metric="ROC",
                   preProcess = c('center', 'scale'),
                   tuneLength=10,
                   trControl=fitControl)

pred_knn_bal <- predict(model_knn_bal, test_can)
cm_knn_bal <- confusionMatrix(pred_knn_bal, test_can$diagnosis, positive = "M")
## Neuarl Network
model_nnet_bal <- train(diagnosis~.,
                    data = train_can_rose,
                    method="nnet",
                    metric="ROC",
                    preProcess=c('center', 'scale'),
                    trace=FALSE,
                    tuneLength=10,
                    trControl=fitControl)
pred_nnet_bal <- predict(model_nnet_bal, test_can)
cm_nnet_bal <- confusionMatrix(pred_nnet_bal, test_can$diagnosis, positive = "M")
## Neural network with pca
model_pca_nnet_bal <- train(diagnosis~.,
                    data = train_can_rose,
                    method="nnet",
                    metric="ROC",
                    preProcess=c('center', 'scale','pca'),
                    trace=FALSE,
                    tuneLength=10,
                    trControl=fitControl)
pred_pca_nnet_bal <- predict(model_nnet_bal, test_can)
cm_pca_nnet_bal <- confusionMatrix(pred_pca_nnet_bal, test_can$diagnosis, positive = "M")

## Neural network with lda
model_lda_nnet_bal <- train(diagnosis~.,
                    data = train_can_lda_rose,
                    method="nnet",
                    metric="ROC",
                    preProcess=c('center', 'scale'),
                    trace=FALSE,
                    tuneLength=10,
                    trControl=fitControl)
pred_lda_nnet_bal <- predict(model_lda_nnet_bal, test_can_lda)
cm_lda_nnet_bal <- confusionMatrix(pred_lda_nnet_bal, test_can_lda$diagnosis, positive = "M")
# Decision Tree
model_dt_bal <- train(diagnosis~.,
                  data = train_can_rose,
                  method="rpart",
                  metric="ROC",
                  preProcess = c('center', 'scale'),
                  trControl=fitControl)
pred_dt_bal <- predict(model_rf_bal, test_can)
cm_dt_bal <- confusionMatrix(pred_dt_bal, test_can$diagnosis, positive = "M")
#SVM kernel
model_svm_bal <- train(diagnosis~.,
                    data = train_can_rose,
                    method="svmRadial",
                    metric="ROC",
                    preProcess=c('center', 'scale'),
                    trace=FALSE,
                    trControl=fitControl)
```

```
pred_svm_bal <- predict(model_svm_bal, test_can)
cm_svm_bal <- confusionMatrix(pred_svm_bal, test_can$diagnosis, positive = "M")
```