**Comparing the performance of Logistic Regression model with Random Forests algorithm in the prediction of breast cancer**

Sahar Nouri

Statistical analysis for data science

November 2020

# Summary

Breast cancer is the second leading cause of cancer deaths among women in the United States. Although mortality rates have been decreasing over the past decade, it is important to continue to make advances in diagnostic procedures as early detection vastly improves chances for survival. This project aims to use classification machine learning models that accurately predict the presence of a malignant tumor using data from fine-needle aspiration (FNA) with visual interpretation.

The Wisconsin Breast Cancer dataset was used that contains 569 patients (63.7% benign and 37.3% malignant). Redundant variables were specified and dropped to reduce the dimensionality of data. Two alternative classification machine learning models include logistic regression and random forests were implemented to compare a conventional model for classification (i.e. logistic regression) with a more sophisticated one (i.e. random forests) and identify the best model with the lowest classification error.

The Random Forest model performed better than logistic regression with an accuracy score of 96%, a true positive rate of 97%, and a true negative rate of 93%.

# Introduction

Breast cancer is the most common cancer in women after skin cancer in the United States. Studies estimate that approximately 15% of newly-diagnosed cancer patients will die in 2015. Breast cancer occurs as a result of abnormal growth of cells in the breast tissue commonly named a tumor. A tumor does not mean cancer. It can be benign (not cancerous), pre-malignant (pre-cancerous), or malignant (cancerous). Currently, 3 methods are commonly used to diagnose cancer, including mammography, fine needle aspiration (FNA) with visual interpretation, and surgical biopsy. Fine needle aspiration (FNA) is a cost-effective method of diagnosis. It examines a small amount of tissue from the tumor and then carefully examines both the characteristics of individual cells and important contextual features. However, this process is highly subjective, depending on the skill and experience of the physician.

Researchers have been working diligently over the past decade to improve the sensitivity of this process. In addition to advances in technology and visual interpretation, researchers have been using various data mining methods to identify the key factors that can help doctors correctly diagnose malignant tumors. Nowadays, machine learning has become more accurate than human medical professionals in the diagnosis of cancerous and non-cancerous tumors. The goal of this project is to use alternative methods of predictive machine learning algorithms and evaluating their performances to provide an optimal diagnostic model. The model can thereby help clinicians make better decisions for the treatment of the breast tumor. In the next section, a description of the data set was provided. In the third section, explanatory analyses for a better understanding of the data at hand and methods for dimensionality reduction are proposed. The fourth section shows the results of the implementation of alternative machine learning methods and their comparisons.

# Data Description

This project utilized the Wisconsin Breast Cancer Data set. The data was created at the University Of Wisconsin Hospital at Madison, Wisconsin, USA, and is available at  <https://www.kaggle.com/uciml/breast-cancer-wisconsin-data>. The dataset contains 569 patients assessing the features of FNAs taken from patients’ breasts. There are 33 attributes per observation; including the ID and binary target variable. The target variable diagnoses whether the tumor is benign or malignant. To create the features they used fluid samples, taken from patients with solid breast masses, and an easy-to-use graphical computer program called Xcyt, which is capable of performing the analysis of cytological features based on a digital scan. The program uses a curve-fitting algorithm, to compute ten features from each one of the cells in the sample, then it calculates the mean value, extreme value, and standard error of each feature for the image, returning a 30 real-valuated vector. Table 1 shows the variable names, descriptions, and types.

**Table 1. List of variables, their descriptions, and types.**

|  |  |  |
| --- | --- | --- |
| Variable | Description | Type |
| ID number ​ | Patient code number | Numeric (Int64) / Input |
| Diagnosis ​ | The diagnosis of breast tissues (M=Malignant, B=Benign) ​ | Categorical (object)/output |
| radius ​ | Distances from the center to points on the perimeter ​ | Numeric (float64) ​/ Input |
| texture ​ | standard deviation of gray-scale values ​ | Numeric (float64) ​/ Input |
| perimeter ​ | ​ size of the core tumor | Numeric (float64) ​/ Input |
| area ​ | ​ | Numeric (float64) ​/ Input |
| smoothness  ​ | local variation in radius lengths ​ | Numeric (float64) ​/ Input |
| compactness ​ | perimeter^2 / area - 1.0 ​ | Numeric (float64) ​​/ Input |
| concavity ​ | the severity of concave portions of the contour ​ | Numeric (float64) ​​/ Input |
| Concave points ​ | number of concave portions of the contour ​ | Numeric (float64) ​​/ Input |
| symmetry ​ | ​ | Numeric (float64) ​​/ Input |
| Fractal dimension ​ | "coastline approximation" - 1 ​ | Numeric (float64) ​​/ Input |

There were no missing values in the data set. The id column was redundant and not useful. Also Unnamed: 33 feature includes NaN. They were dropped from the analysis.

# Explanatory Analyses

The target variable consists of 357 (62.7%) benign and 212 (37.3%) malignant patients (Figure 1).

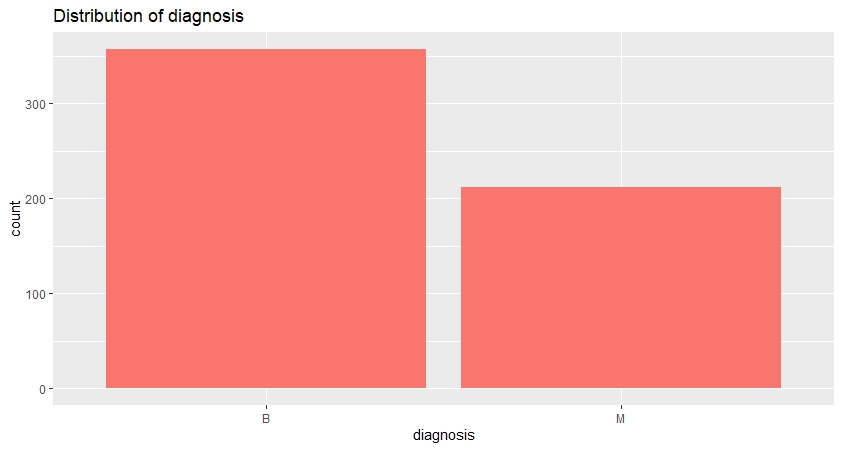
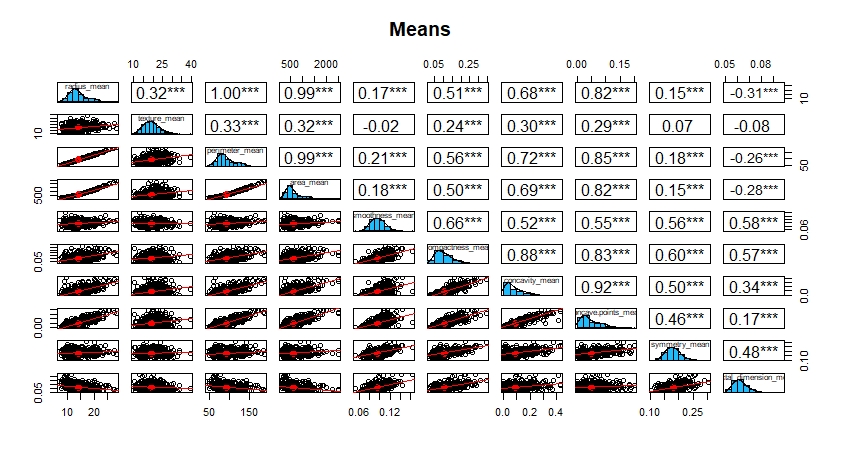
****

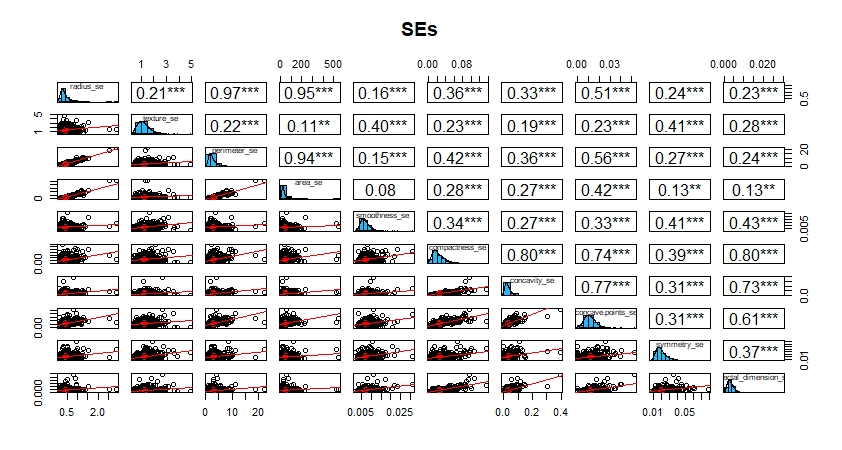
Figure1. Frequency of cancer diagnosis.

Figure 2 (a,b, and c) shows the correlations between mean, SE, and the worst values separately that show strong correlations for many pairs of variables. Also, a correlation heat map was calculated to show the linear relationships between all 30 features (Figure 3).

(a)



(b)



©

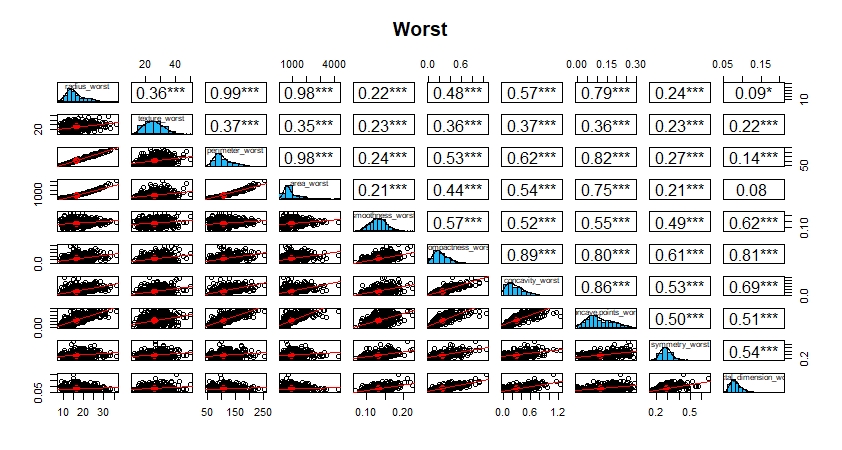


Figure 3. The correlation matrices of independent variables for mean, SE, and worst values separately.

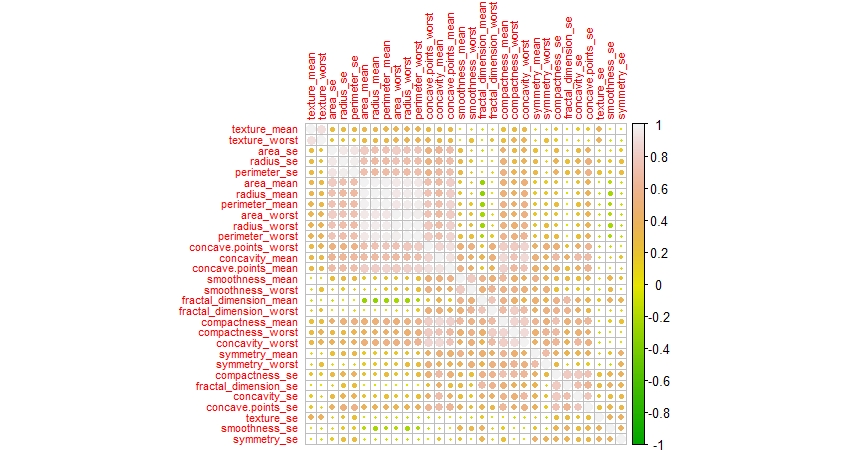


Figure 3. The correlation heat map of independent variables.

# Dimensionality Reduction

Since there were high correlations between variables, I used a principal component analysis for dimensionality reduction to use principal components (PCs) as predictors instead of original variables in predictive models.

Figure 4. shows a plot of cumulative variance that can be explained by PCs. The horizontal line shows a cumulative variance of 95%. Using the first 10 PCs 95% of the variance of data can be explained. So, I continue our analysis using these 10 PCs. (The data became normalized before using PCA.)

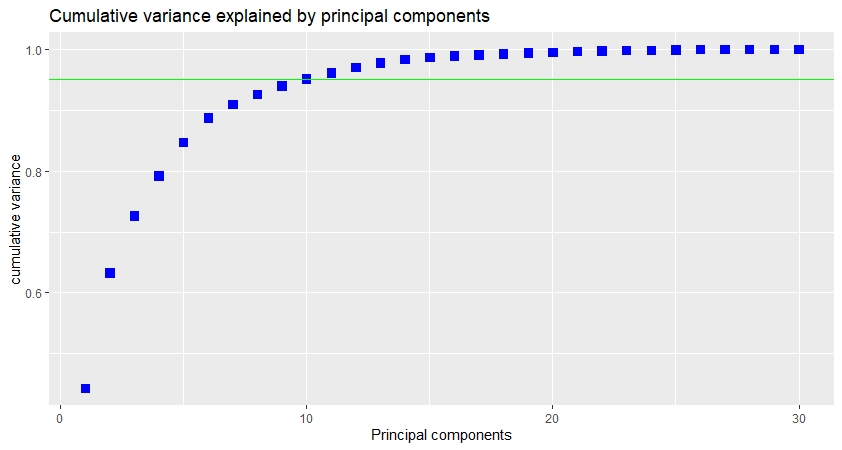


Figure 4. The cumulative variance is explained by PCs.

# Predictive Analyses

Classification machine learning algorithms are supervised learning approaches in which the computer program learns from the input data and then uses this learning to classify new observations. Different machine learning algorithms were implemented and evaluated to find the best model with the lowest classification errors. The data was split to train and test data with a ratio of 70/30. Training data was used for model training and test data was used for the evaluation of model performances and predictions. Then models were then tuned to optimize their performances. All analyses were performed using R and the codes are shown in appendix 1.

## Performance measurements

Several methods could be used to measure the performance of a classification model. Some of them are log-loss, AUC, confusion matrix, and precision-recall. ROC curve (Receiver Operating Characteristics Curve) is another metric to measure the performance of a classifier model, ROC curve depicts the locus of rate of true positives to the rate of false positives, this highlights the sensitivity of the classifier model.

Accuracy is the measure of correct prediction of the classifier compared to the overall data points. Simply put, it is the ratio of the units of correct predictions and the total number of predictions made by the classifiers. However, accuracy doesn’t give us the best picture of the cost of misclassification or unbalanced testing data set. Therefore we look at the confusion matrix which treats the failed examples of Class F and Class NF images differently.

Confusion Matrix as the name suggests gives us a matrix as output and describes the complete performance of the model. It is based on 4 terms:

* True Positives: The cases in which we predicted YES and the actual output was also YES.
* True Negatives: The cases in which we predicted NO and the actual output was NO.
* False Positives: The cases in which we predicted YES and the actual output was NO.
* False Negatives: The cases in which we predicted NO and the actual output was YES.

ROC curve (Receiver Operating Characteristics Curve) is another metric to measure the performance of a classifier model, ROC curve depicts the locus of rate of true positives to the rate of false positives, this highlights the sensitivity of the classifier model.

For this project, I have used the confusion matrix as the performance measurement.

# Logistic Regression

It is a statistical method for analyzing a data set in which there are one or more features that determine the target variable. The target is measured with a dichotomous variable (in which there are only two possible outcomes). The goal of logistic regression is to find the best fitting model to describe the relationship between the dichotomous characteristic of interest (dependent variable = response or outcome variable) and a set of independent (predictor or explanatory) variables. Figure 5. Shows the results of logistic regression for classifying the diagnosis variable after hyperparameter tuning. The accuracy was 94% with a 95% CI (.89 - .97). The true positive and true negative rates were 96.7% and 92.6% respectively.

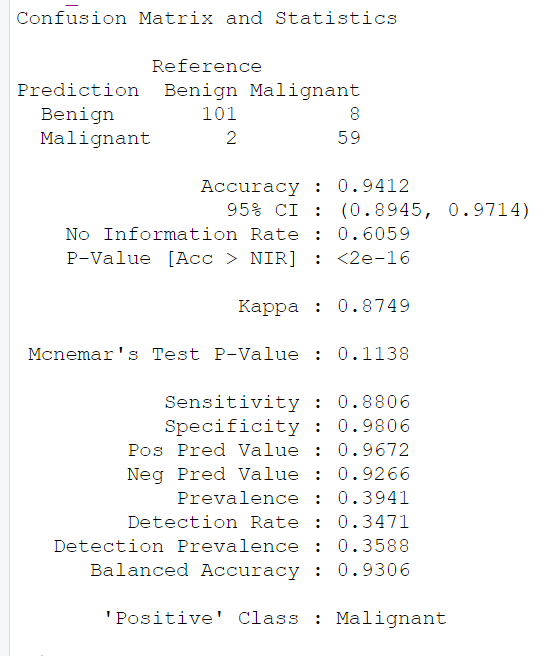


Figure 5. The confusion matrix of logistic regression after hyperparameter tuning.

## Random Forests

The random forest method is an ensemble learning method for classification and regression. It constructs a multitude of decision trees at training time and outputs the class or the mean prediction. The confusion matrix after hyperparameter tuning show an accuracy of 95.8%, a true positive rate of 96.8%, and a true negative rate of 95.2%.

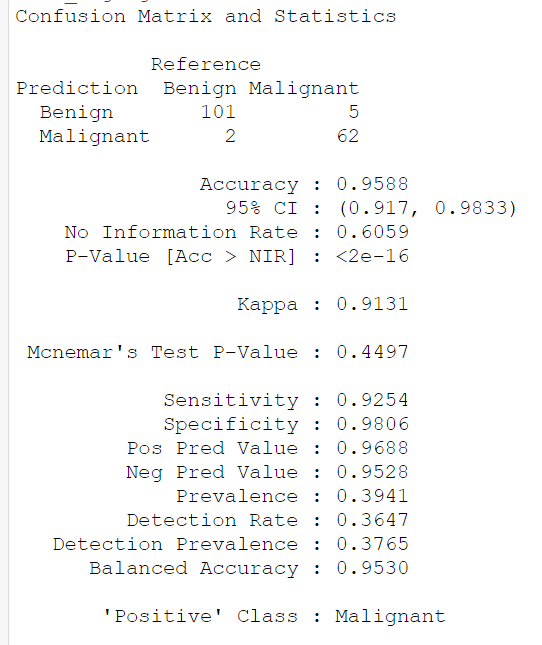


Figure 6. The confusion matrix of the random forest after hyperparameter tuning.

# Conclusion

We conducted two machine learning methods logistic regression and random forests as classification models for predicting breast cancer. When the number of variables is very large and they are highly correlated we can use feature selection or principal components analysis to reduce the number of independent variables and improve the performance of the model. For this data, we performed classification models using the first 10 principal components instead of using 30 original variables. Considering 212 malignant samples it was an acceptable number for independent variables (statistically we need 10 samples of the class of interest, that here is malignant, for each variable). Table 2 compares the results of the two alternative models after hyperparameter tuning. The Random Forest performed better than the logistic regression for predicting the type of tumor with an accuracy score of 96%. Ninety-seven percent of malignant tumors are detected correctly as malignant using random forests. Also, if a tumor is benign the random forests algorithm can predict it as benign with a probability of 93%.

Table 1. Final results of the two machine learning models on the Wisconsin Breast Cancer Data set.

|  |  |  |
| --- | --- | --- |
| Performance | Logistic regression | Random Forest |
| Accuracy | 94 % | 96% |
| Sensitivity​ | ​88% | 92% |
| Specificity | 98% | 98% |
| True Positive ​ | ​97% | 97% |
| True Negative | ​93% | 95% |

**Appendix 1: R codes for the project**

**library("ggplot2")**

**library("corrgram")**

**library("car")**

**library("lattice")**

**library("ROCR")**

**library("plotly")**

**library("tree")**

**library(devtools)**

**library(tidyverse)**

**library(corrplot)**

**library(PerformanceAnalytics)**

**library(psych)**

**library(GGally)**

**dt <- read.csv("F:\\canada\\Data Science\\DAT200-Sttatistical analysis for data science\\Project\\datasets\_180\_408\_data.csv", header=TRUE, sep=",")**

**View(dt)**

**colnames(dt)**

**dim(dt)**

**table(dt$diagnosis)**

**library(dplyr)**

**glimpse(dt)**

**#----------------------------------------------------------------------**

**#check for missing value**

**sapply(dt, function(x) sum(is.na(x)))**

**library(dplyr)**

**dt <- dt %>% select(-c("id","X"))**

**ncol(dt)**

**#----------------------------------------------------------------------**

**summary(dt)**

**summary(dt$diagnosis)**

**table(dt$diagnosis)**

**#-----------------------------------------------------------------------**

**dt$y[dt$diagnosis=="M"] = 1**

**dt$y[dt$diagnosis=="B"] = 0**

**#---------------------------------------------------------------------**

**#Correlation chart for means**

**library(PerformanceAnalytics)**

**chart.Correlation(dt[, c(3:12)], histogram=TRUE, col="grey10", pch=1, main="Cancer Means")**

**library(psych)**

**pairs.panels(dt[,c(3:12)], method="pearson",**

**hist.col = "#1fbbfa", density=TRUE, ellipses=TRUE, show.points = TRUE,**

**pch=1, lm=TRUE, cex.cor=1, smoother=F, stars = T, main="Means")**

**#correlation chart for SE**

**chart.Correlation(dt[, c(13:22)], histogram=TRUE, col="grey10", pch=1, main="Cancer SEs")**

**pairs.panels(dt[,c(13:22)], method="pearson",**

**hist.col = "#1fbbfa", density=TRUE, ellipses=TRUE, show.points = TRUE,**

**pch=1, lm=TRUE, cex.cor=1, smoother=F, stars = T, main="SEs")**

**#correlation chart for worst**

**chart.Correlation(dt[, c(23:32)], histogram=TRUE, col="grey10", pch=1, main="Cancer SEs")**

**pairs.panels(dt[,c(23:32)], method="pearson",**

**hist.col = "#1fbbfa", density=TRUE, ellipses=TRUE, show.points = TRUE,**

**pch=1, lm=TRUE, cex.cor=1, smoother=F, stars = T, main="Worst")**

**#-----------------------------------------------------------------------**

**library(corrplot)**

**corMat <- cor(dt[,2:31])**

**corrplot(corMat, order = "hclust", tl.cex = 0.7, col = terrain.colors(200))**

**#dropping covariates with more than .9 corr**

**library(caret)**

**highlyCor <- colnames(dt)[findCorrelation(corMat, cutoff = 0.9, verbose = TRUE)]**

**print(highlyCor)**

**#10 columns are flagged for removal.**

**dt\_cor <- dt[, which(!colnames(dt) %in% highlyCor)]**

**ncol(dt\_cor)**

**#-----------------------------------------------------------------------**

**#split data train and test subsets (70/30)**

**sample\_size = floor(0.7 \* nrow(dt))**

**# set the seed to make your partition reproductible**

**set.seed(1729)**

**train\_set = sample(seq\_len(nrow(dt)), size = sample\_size)**

**training = dt[train\_set, ]**

**testing = dt[-train\_set, ]**

**head(training)**

**nrow(training)**

**nrow(testing)**

**#---------------------------------------------------------------------**

**set.seed(1234)**

**df <- cbind(diagnosis=dt$y, dt\_cor)**

**train\_indx <- createDataPartition(df$diagnosis, p = 0.7, list = FALSE)**

**train\_set <- df[train\_indx,]**

**test\_set <- df[-train\_indx,]**

**nrow(train\_set)**

**nrow(test\_set)**

**#-----------------------------------------------------------------------**

**library("ggplot2")**

**ggplot(dt, aes(x = diagnosis)) +**

**geom\_bar(aes(fill = "blue")) +**

**ggtitle("Distribution of diagnosis") +**

**theme(legend.position="none")**

**#for training set**

**ggplot(training, aes(x = diagnosis)) +**

**geom\_bar(aes(fill = "blue")) +**

**ggtitle("Distribution of diagnosis for the training subset") +**

**theme(legend.position="none")**

**#for testing set**

**ggplot(testing, aes(x = diagnosis)) +**

**geom\_bar(aes(fill = "blue")) +**

**ggtitle("Distribution of diagnosis for the testing subset") +**

**theme(legend.position="none")**

**#----------------------------------------------------------------------**

**#fitting GLM on the training data**

**model = glm(diagnosis ~. ,family=binomial(link='logit'), control = list(maxit = 50),data=train\_set)**

**print(summary(model))**

**print(anova(model, test="Chisq"))**

**#selecting variables based on backward elimination method**

**step(model, direction = "backward", trace=FALSE )**

**#final model for training using selected variables**

**model\_final=glm(formula = diagnosis ~ smoothness\_mean + concave.points\_mean +**

**area\_se + concavity\_se + smoothness\_worst + compactness\_worst +**

**concavity\_worst + symmetry\_worst, family = binomial(link = "logit"),**

**data = train\_set, control = list(maxit = 50))**

**#---------------------------------------------------------------------**

**#prdictions using final model for training set**

**prediction\_training = predict(model\_final,train\_set, type = "response")**

**prediction\_training = ifelse(prediction\_training > 0.5, 1, 0)**

**error = mean(prediction\_training != train\_set$diagnosis)**

**print(paste('Model Accuracy',1-error))**

**#ROCcurve for training**

**p = predict(model\_final, train\_set, type="response")**

**library(ROCR)**

**pr = prediction(p, train\_set$diagnosis)**

**prf = performance(pr, measure = "tpr", x.measure = "fpr")**

**plot(prf)**

**#AUC for training**

**auc = performance(pr, measure = "auc")**

**auc = auc@y.values[[1]]**

**print(paste("Model Accuracy", auc))**

**#----------------------------------------------------------------------**

**#prdictions using final model for testing set**

**prediction\_testing = predict(model\_final,test\_set, type = "response")**

**prediction\_testing = ifelse(prediction\_testing > 0.5, 1, 0)**

**error = mean(prediction\_testing != test\_set$diagnosis)**

**print(paste('Model Accuracy',1-error))**

**#ROCcurve for testing**

**p = predict(model\_final, test\_set, type="response")**

**library(ROCR)**

**pr = prediction(p, test\_set$diagnosis)**

**prf = performance(pr, measure = "tpr", x.measure = "fpr")**

**plot(prf)**

**#AUC for training**

**auc = performance(pr, measure = "auc")**

**auc = auc@y.values[[1]]**

**print(paste("Model Accuracy", auc))**

**#--------------------------------------------------------------------**

**#Using PCA**

**dt.pca <- prcomp(dt[, 2:31], center=TRUE, scale=TRUE)**

**plot(dt.pca, type="l", main='')**

**grid(nx = 10, ny = 14)**

**title(main = "Principal components ", sub = NULL, xlab = "Components")**

**box()**

**pca\_var <- dt.pca$sdev^2**

**pve\_df <- pca\_var / sum(pca\_var)**

**cum\_pve <- cumsum(pve\_df)**

**pve\_table <- tibble(comp = seq(1:ncol(select(dt,-c("diagnosis","y")))), pve\_df, cum\_pve)**

**ggplot(pve\_table, aes(x = comp, y = cum\_pve)) +**

**geom\_point(shape=22, fill="blue", color="blue", size=3) +**

**geom\_abline(intercept = 0.95, color = "green", slope = 0)+xlab("Principal components") +**

**ylab("cumulative variance") +**

**ggtitle("Cumulative variance explained by principal components")**

**#selecting the first 10 PC for analysis**

**pca\_df <- as.data.frame(dt.pca$x)**

**pca\_dff <- pca\_df[,1:10]**

**View(pca\_dff)**

**#---------------------------------------------------------------------**

**set.seed(1234)**

**library(caret)**

**dfpca <- cbind(diagnosis=dt$y, pca\_dff)**

**trainpca\_indx <- createDataPartition(dfpca$diagnosis, p = 0.7, list = FALSE)**

**trainpca\_set <- dfpca[trainpca\_indx,]**

**testpca\_set <- dfpca[-trainpca\_indx,]**

**nrow(trainpca\_set)**

**nrow(testpca\_set)**

**#---------------------------------------------------------------------**

**#fitting GLM on the trainingpca data**

**modelpca = glm(diagnosis ~. ,family=binomial(link='logit'), control = list(maxit = 50),data=trainpca\_set)**

**print(summary(modelpca))**

**print(anova(modelpca, test="Chisq"))**

**#---------------------------------------------------------------------**

**#prdictions using final model for trainingpca set**

**prediction\_trainingpca = predict(modelpca,trainpca\_set, type = "response")**

**prediction\_trainingpca = ifelse(prediction\_trainingpca > 0.5, 1, 0)**

**error = mean(prediction\_trainingpca != trainpca\_set$diagnosis)**

**print(paste('Modelpca Accuracy',1-error))**

**#ROCcurve for trainingpca**

**p = predict(modelpca, trainpca\_set, type="response")**

**library(ROCR)**

**pr = prediction(p, trainpca\_set$diagnosis)**

**prf = performance(pr, measure = "tpr", x.measure = "fpr")**

**plot(prf)**

**#AUC for training**

**auc = performance(pr, measure = "auc")**

**auc = auc@y.values[[1]]**

**print(paste("Area under the Roc curve", auc))**

**#---------------------------------------------------------------------**

**#fitting GLM on the testingpca data**

**modelpca = glm(diagnosis ~. ,family=binomial(link='logit'), control = list(maxit = 50),data=testpca\_set)**

**print(summary(modelpca))**

**print(anova(modelpca, test="Chisq"))**

**#---------------------------------------------------------------------**

**#prdictions using final model for testingpca set**

**prediction\_testingpca = predict(modelpca,testpca\_set, type = "response")**

**prediction\_testingpca = ifelse(prediction\_testingpca > 0.5, 1, 0)**

**error = mean(prediction\_testingpca != testpca\_set$diagnosis)**

**print(paste('Modelpca Accuracy',1-error))**

**#ROCcurve for testingpca**

**p = predict(modelpca, testpca\_set, type="response")**

**library(ROCR)**

**pr = prediction(p, trainpca\_set$diagnosis)**

**prf = performance(pr, measure = "tpr", x.measure = "fpr")**

**plot(prf)**

**#AUC for training**

**auc = performance(pr, measure = "auc")**

**auc = auc@y.values[[1]]**

**print(paste("Area under the Roc curve", auc))**

**#--------------------------------------------------------------------**

**trainpca\_set$diagnosis<-factor(trainpca\_set$diagnosis,levels=c(0,1),labels=c("Benign","Malignant"))**

**df\_control <- trainControl(method="cv",**

**number = 15,**

**classProbs = TRUE,**

**summaryFunction = twoClassSummary)**

**model\_rf <- train(diagnosis ~., data = trainpca\_set,**

**method = "rf",**

**metric = 'ROC',**

**trControl = df\_control)**

**#---------------------------------------------------------------------**

**#prediction on trainingpca\_set**

**library(caret)**

**library(e1071)**

**prediction\_rf <- predict(model\_rf, trainpca\_set)**

**cm\_rf <- confusionMatrix(prediction\_rf, trainpca\_set$diagnosis, positive = "Malignant")**

**cm\_rf**

**#prediction on testingpca\_set**

**testpca\_set$diagnosis<-factor(testpca\_set$diagnosis,levels=c(0,1),labels=c("Benign","Malignant"))**

**prediction\_rf <- predict(model\_rf, testpca\_set)**

**cm\_rf <- confusionMatrix(prediction\_rf, testpca\_set$diagnosis, positive = "Malignant")**

**cm\_rf**

**#--------------------------------------------------------------------**

**#logistic model**

**model\_logreg <- train(diagnosis ~., data = trainpca\_set, method = "glm",**

**metric = "ROC",**

**trControl = df\_control)**

**prediction\_logreg <- predict(model\_logreg, testpca\_set)**

**cm\_logreg <- confusionMatrix(prediction\_logreg, testpca\_set$diagnosis, positive = "Malignant")**

**cm\_logreg**

**#ROCcurve for testingpca**

**library(ROCR)**

**pr = prediction(prediction\_rf, trainpca\_set$diagnosis)**

**prf = performance(pr, measure = "tpr", x.measure = "fpr")**

**plot(prf)**

**#AUC for training**

**auc = performance(pr, measure = "auc")**

**auc = auc@y.values[[1]]**

**print(paste("Area under the Roc curve", auc))**