Breast Cancer Predictive Modeling

Authors: Sam Kalkman and Puja Ammineni

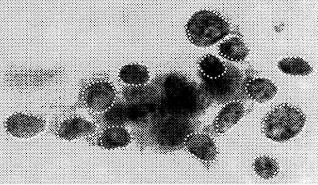
Group 1

MA4790/MA5790

December 2024

# Abstract

Breast cancer is the most widespread cancer among women and is seldom easy to diagnose. We search for a less subjective and invasive diagnostic approach than traditional biopsies. Using a dataset of characteristics of the cell nuclei derived from medical images, we develop predictive models that classify diagnoses into malignant and benign. All steps that form pre-processing included the removal of the most correlated predictors, Box-Cox for symmetry, and spatial sign transformations for outliers. Stratification resulted in an 80:20 split between training and testing via stratified random sampling of data. The dataset was explored multiple times through training with several linear and non-linear models, after which SVM and Neural Networks resulted in the best models so far. Whereupon testing, the SVM model slightly outperformed the others, with a corresponding ROC-AUC of 1.0. Finally, we investigated the importance of various predictors in the final model to enhance interpretability and diagnostic reliability.



# 

# Table of Contents

[**Abstract**](#_1bhuyhbbk6ap) **2**

[**Table of Contents**](#_okf1k5vtoyw5) **3**

[**Background**](#_2xf6myi4fs2u) **4**

[**Dataset Background**](#_x3vf11iqic8n) **4**

[**Preprocessing**](#_ow5098hc7nli) **5**

[Correlation](#_yfna5mcimfcb) 5

[Transformations](#_myow91gl60n9) 5

[**Data Splitting**](#_4dx67a7cnq05) **6**

[**Model Fitting**](#_poxl0xobrmny) **7**

[Linear Models](#_ucywivnf6cn3) 7

[Non-Linear Models](#_zi2w281ohkxs) 8

[Best Models and Testing](#_i5g9bi4p7hh3) 8

[**Summary**](#_t6hsrtzg7225) **11**

[**References**](#_py2sysy3jo9j) **11**

[**Appendix 1: Supplemental Materials for Linear Models**](#_ctaxga69dbrl) **11**

[**Appendix 2: Supplemental Materials for Non-Linear Models**](#_psymtenzc848) **12**

[**Appendix 3: R Code**](#_ud0vuof11lfe) **16**

[Uncorrelated Data Code](#_65o6nrgqcblh) 16

[Correlated Data Code](#_86o0s1t8ug1g) 22

# 

# Background

Breast cancer is one of the most common and dangerous cancers that we know of. It affects millions of lives each year and can be difficult to catch early. Unfortunately, this difficulty can be deadly; the later breast cancer is caught, the more dangerous it may be. So, many researchers around the world look for ways to catch and treat breast cancer as early as they can.

In 1992, a team of researchers was looking to develop a way to use medical imaging data to diagnose breast cancer using computation that would be less invasive than the biopsies that were the primary method of diagnosis at the time. They hoped that looking at the individual characteristics of the nuclei breast cells of the patient near or around the suspected tumor would provide an accurate and less invasive solution that could save lives. Fine needle aspirations (FNAs) are a much less invasive solution to get these cells, and the model building can provide a far less subjective analysis of the data this surgery provides.

# Dataset Background

We are using a dataset that includes medical imaging data from 569 patients with confirmed malignant (M) or benign (B) diagnoses. This diagnosis is the response variable which only has two levels M and B. For each of these samples, 30 continuous predictors are extracted from the medical imaging. There are 10 feature types relating to the nuclei found in the medical images (see reference paper):

1. Radius - the radius of the nuclei
2. Perimeter - the perimeter of the nuclei
3. Area - the area shown in the image of the nuclei
4. Compactness - this combines the perimeter and area of the nuclei (Perimeter/Area)
5. Smoothness - a metric that compares the length of radii of the nuclei to the mean length of its neighbors
6. Concavity - chords are made and used to measure how much concavity the nuclei have to one number
7. Concave Points - counts the number of contour concavities shown by the concavity feature
8. Symmetry - A major axis is made across each nucleus and then the length of perpendicular lines is calculated and compared
9. Fractal Dimension - fewer and fewer chords are created on the nuclei and the perimeter decreases. This process is plotted on log scale and the slope is calculated
10. Texture - variance in gray-scale intensities in the component images of the nuclei

For each of these 10 features, 3 statistics are calculated: mean, extreme (maximum), and standard deviation for all nuclei represented in one medical slide. In the dataset this is represented by the form [feature name]1, [feature name]2, and [feature name]3 respectively for mean, extreme value, and standard deviation.

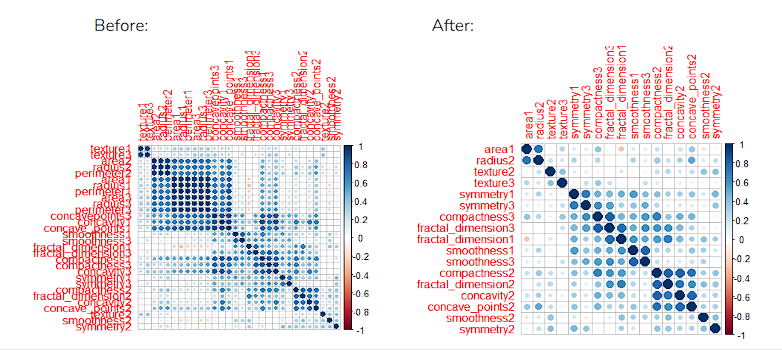
# Preprocessing

Before we begin our model creation, we need to pre-process our data to account for the needs of different models. Some models, such as the logistic regression model, require that predictors do not have high correlations, few outliers, and/or roughly symmetric data. We will remove highly correlated predictors as necessary for these models but include that information for models that can handle highly correlated predictors.

Normally we would also check for missing data perform imputation look for near-zero variance and create dummy variables for categorical predictors. Still, as our dataset is missing such features, these actions were not applied.

## Correlation

We created a correlation plot of our predictors to examine how our predictors were correlated with each other, as shown in Figure 1. At the 0.85 correlation level, we removed 13 predictors: concavity1, concave\_points1, compactness1, concavepoints3, concavity3, perimeter3, radius3, perimeter1, area3, radius1, perimeter2, area2, and texture1. This left us with 17 remaining predictors to use for the models that cannot handle highly correlated predictors.



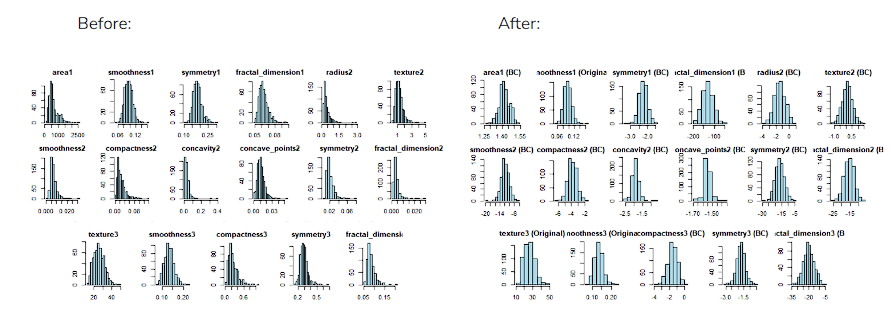
*Figure 1: Correlation plot before and after we removed the predictors correlated at the 0.85 level*

## 

## 

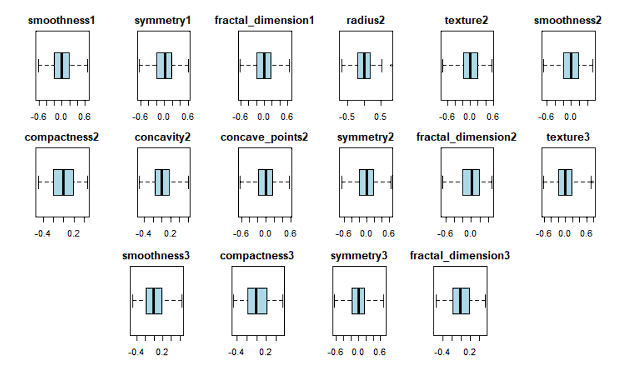
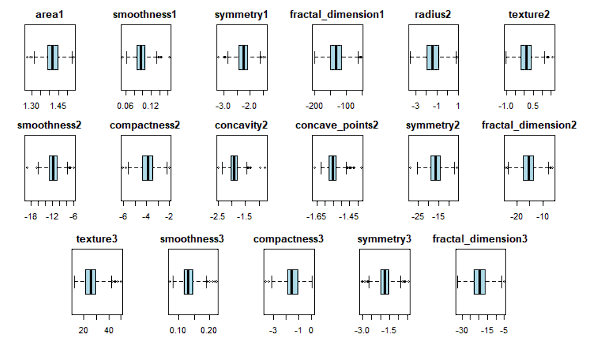
## Transformations

We performed box-cox transformations on the predictors which were not roughly symmetric. The visualization before and after is depicted in Figure 2. This was done for both the correlated and uncorrelated datasets. Transformations were done to all predictors because they have little harmful effect on model performance and do not remove any information.



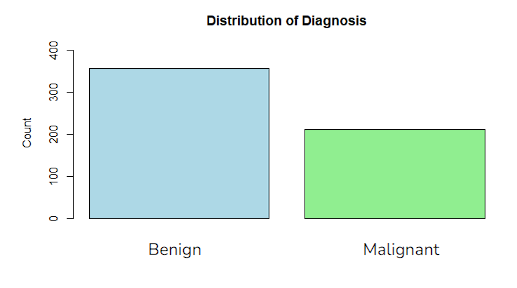
*Figure 2: before and after the box-cox transformation was applied to the 17 predictors in the uncorrelated dataset*

To remove outliers, we applied center and and scale transformations then applied spatial sign transformation. This transformation is visualized in Figure 3.

*Figure 3: Before and after spatial sign transformation was applied to the 17 predictors in the uncorrelated dataset*

# Data Splitting

To create our models we will create a training and testing dataset. Because our response variable, diagnosis, is unbalanced (see Figure 4), we will perform stratified random sampling to create our split. Taking into account the size of our dataset, we settled on splitting the dataset into 80% training data and 20% testing data.

*Figure 4: Distribution of the response variable, Diagnosis*

At the end of our preprocessing and data splitting, we are left with 30 continuous predictors in the correlated dataset, 17 predictors in the uncorrelated dataset, 456 data points in the training set, and 113 data points in the testing set.

# Model Fitting

We worked with two data sets: one with highly correlated predictors and the other one without them. All pre-processing steps were applied consistently to both data sets for comparability.

We develop several classification models that include the following: Logistic Regression,

Linear Discriminant Analysis, Partial Least Square Discriminant Analysis, Penalized Models, Quadratic Discriminant Analysis, Regularized Discriminant Analysis, Mixture Discriminant Analysis, Neural Networks, Flexible Discriminant Analysis, Support Vector Machine Discriminant Analysis, k-nearest Neighbors, Naive Bayes Classifier, among others. Considering that our outcome variable is binary and unbalanced, ROC-AUC has been selected as a measure of performance.

Logistic Regression, LDA, QDA, RDA, MDA, FDA, and KNN models took our uncorrelated data as an input.

The PLSDA, Penalized Model, Neural Network, SVMDA, and Naive Bayes models used the correlated data.

We will train the models using 5 repeats of 10-fold cross-validation.

## Linear Models

1. Logistic Regression
2. Linear Discriminant Analysis
3. Partial Least Square Discriminant Analysis
4. Penalized Models

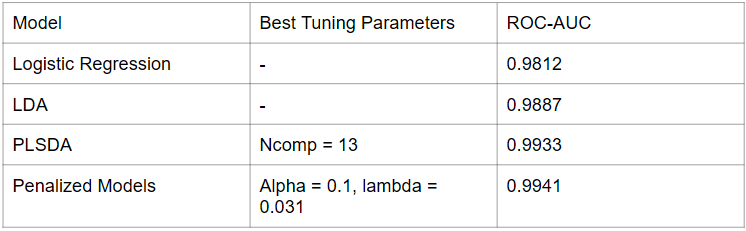
## Non-Linear Models

1. Quadratic Discriminant Analysis
2. Regularized Discriminant Analysis
3. Mixture Discriminant Analysis
4. Neural Network
5. Flexible Discriminant Analysis
6. Support Vector Machine Discriminant Analysis
7. k-Nearest Neighbors
8. Naive Bayes Classifier

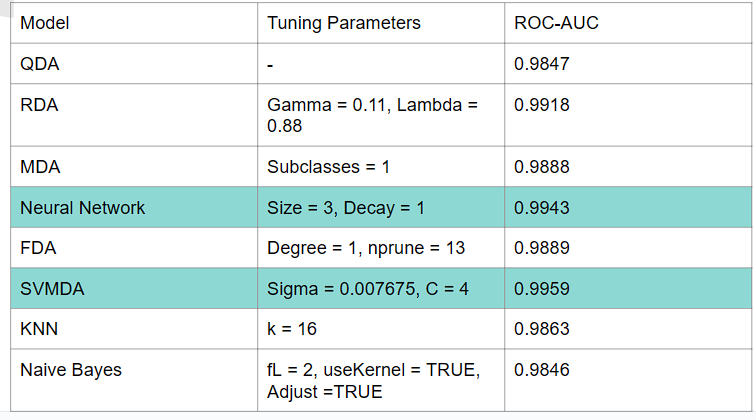
## Best Models and Testing

We trained each of the above models on our training set and analyzed their performance using the ROC-AUC statistic. We found that two non-linear models, the Neural Network and SVM models performed the best.

*Linear Models*

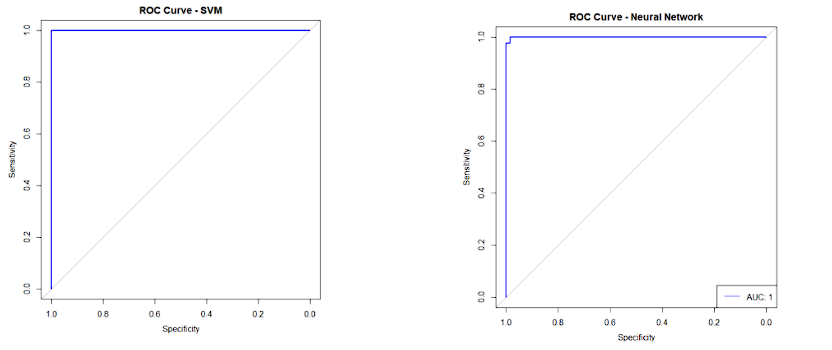


*Figure 5: Performance Table for Linear Models*

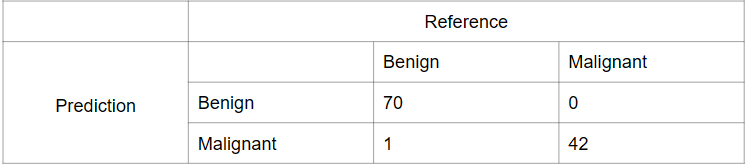
*Non-Linear Models*

*Figure 6: Performance Table for Non-Linear Models*

With this information, we tested these trained models against our testing data to find which performs better. The SVM model ended up with an ROC-AUC of 1 against our testing set while the Neural Network had 0.9997 meaning the SVM model performed slightly better in our testing set. Figure 7 shows this relationship, though it rounds off the ROC-AUC for the Neural Network.



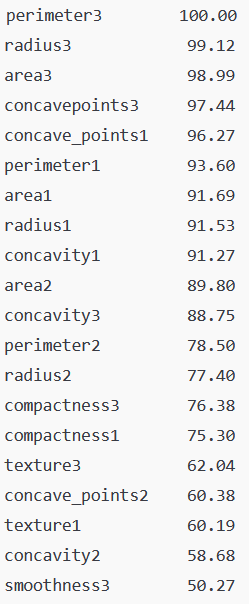
*Figure 7: ROC curves for the SVM and Neural Network models against the training set*



*Figure 8: Confusion matrix for the SVM model against the testing set*

Lastly, we looked at the top important predictors for our trained SVM model. These are depicted in Figure 9. As we can see, our top predictors mostly consist of the type 3, or standard deviation, predictors. This makes sense because the more variance there is between nuclei, the more likely it is that there may be something wrong with the cells they are from.

*Predictor Importance*

*Figure 9: Predictor Importance*

# Summary

Support Vector Machines Discriminant Analysis and Neural Network models emerged as the best performers among the models evaluated. SVMDA had the highest score in the ROC-AUC of 0.9959, which indicated its exceptional ability in the discrimination of benign and malignant cases. In this model, the parameter sigma was kept constant at 0.007, while the regularization parameter C was tuned to 4 for optimized performance. The SVMDA model gave excellent sensitivity and specificity, with only one misclassification, hence the best model. The Neural Network, which came second, had a very good score for ROC-AUC about 0.9943. The best parameters for this were a hidden layer size of 3 and a regularization decay of 1, capturing the nonlinear relationship that exists within the data. Both models showcased remarkable predictive accuracy, with SVMDA slightly outperforming the Neural Network in distinguishing breast cancer cases.

We were happy with the performance of the SVM model and think that the data can be used with this model to help with the prediction of breast cancer.

# References

* <https://archive.ics.uci.edu/dataset/17/breast+cancer+wisconsin+diagnostic>
* <https://minds.wisconsin.edu/bitstream/handle/1793/59692/TR1131.pdf;jsessionid=4D9E6799EA8DE598B71EE0F846523D01?sequence=1>

# Appendix 1: Supplemental Materials for Linear Models

1. Logistic Regression

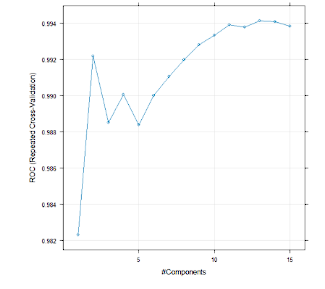
There are no tuning parameters for the logistic model.

1. Linear Discriminant Analysis

There are no tuning parameters for the LDA model

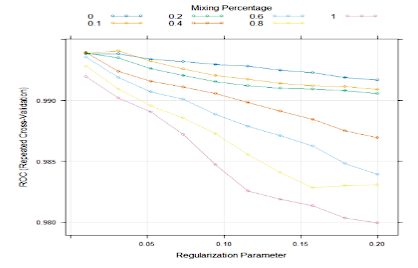
1. Partial Least Square Discriminant Analysis

The highest performing model for PLS was with a tuning parameter of ncomp=13



1. Penalized Models

The highest performing model for the Penalized Models was with tuning parameters of alpha=0.1 and lambda = 0.0311



# 

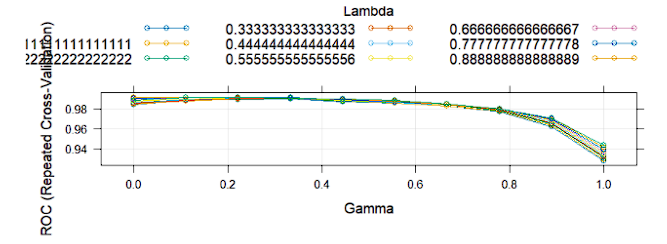
# Appendix 2: Supplemental Materials for Non-Linear Models

1. Quadratic Discriminant Analysis

QDA requires no tuning parameters

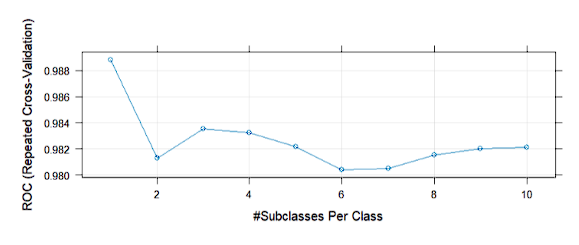
1. Regularized Discriminant Analysis

The highest performing model for the RDA was with tuning parameters of gamma=0.11 and lambda = 0.88



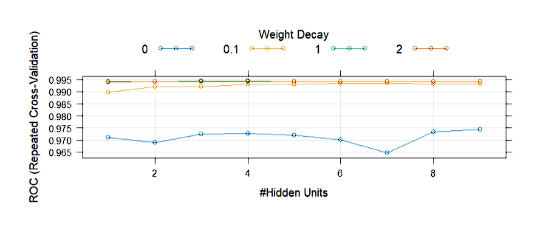
1. Mixture Discriminant Analysis

The highest performing model for the MDA was with a tuning parameter of subclasses=1



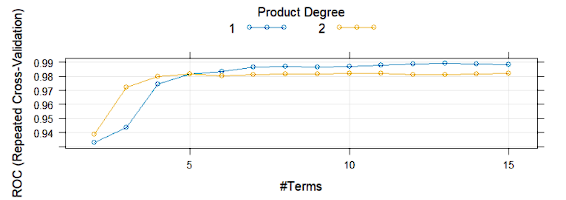
1. Neural Network

The highest performing model for the Neural Network was with tuning parameters of size=3 and decay=1



1. Flexible Discriminant Analysis

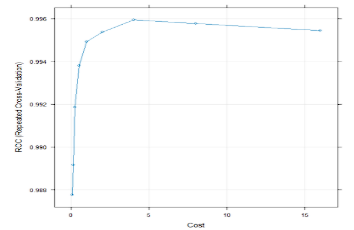
The highest performing model for the FDA was with tuning parameters of degree=1 and nprune=13



1. Support Vector Machine Discriminant Analysis

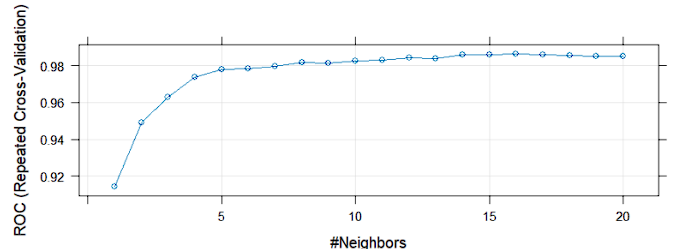
Sigma was held constant at 0.007675455

The highest-performing model for the SVM was with a tuning parameter of C = 4



1. k-Nearest Neighbors

The highest-performing model for the SVM was with a tuning parameter of C = 4



1. Naive Bayes Classifier

Naive Bayes requires no tuning parameters

# Appendix 3: R Code

## Uncorrelated Data Code

library(AppliedPredictiveModeling)

library(caret)

library(corrplot)

library(ggplot2)

library(gridExtra)

library(e1071)

library(pROC)

# Load dataset

wdbc <- read.csv("C:\\Users\\Puja\\Desktop\\Predictive\\Data\\wdbc.data", header = TRUE)

wdbc <- wdbc[, -1]

# Step 1: Correlated Predictor Removal

numeric\_wdbc <- wdbc[, sapply(wdbc, is.numeric)]

correlations <- cor(numeric\_wdbc)

corrplot(correlations, order = "hclust", main = "Correlation Heatmap (Before Removal)") # Initial correlation heatmap

# Identify and remove highly correlated predictors

highCorr <- findCorrelation(correlations, cutoff = 0.85)

filtered\_data <- numeric\_wdbc[, -highCorr]

cat("Number of predictors removed:", length(highCorr), "\n")

cat("Remaining predictors:", ncol(filtered\_data), "\n")

# Plot correlation heatmap after removal

new\_correlations <- cor(filtered\_data)

corrplot(new\_correlations, order = "hclust", main = "Correlation Heatmap (After Removal)")

# Step 2: Box-Cox Transformation

boxcox\_data <- filtered\_data # Copy filtered data for Box-Cox transformation

for (col in colnames(boxcox\_data)) {

if (min(boxcox\_data[[col]]) <= 0) {

boxcox\_data[[col]] <- boxcox\_data[[col]] + abs(min(boxcox\_data[[col]])) + 0.001

}

BCTrans <- BoxCoxTrans(boxcox\_data[[col]])

boxcox\_data[[col]] <- predict(BCTrans, newdata = boxcox\_data[[col]])

}

# Visualize skewness before and after Box-Cox transformation

skew\_before <- apply(filtered\_data, 2, skewness) # Skewness before Box-Cox

skew\_after <- apply(boxcox\_data, 2, skewness) # Skewness after Box-Cox

barplot(skew\_before, main = "Skewness Before Box-Cox",

xlab = "Predictors", ylab = "Skewness", las = 2, col = "lightblue")

barplot(skew\_after, main = "Skewness After Box-Cox",

xlab = "Predictors", ylab = "Skewness", las = 2, col = "lightgreen")

# Step 3: Center and Scale Transformations

preprocess\_trans <- preProcess(boxcox\_data, method = c("center", "scale")) # Centering and scaling

centered\_scaled\_data <- predict(preprocess\_trans, boxcox\_data)

# Step 4: Spatial Sign Transformation

spatial\_sign\_data <- caret::spatialSign(centered\_scaled\_data)

# Visualize correlation heatmap after spatial sign transformation

final\_correlations <- cor(spatial\_sign\_data)

corrplot(final\_correlations, order = "hclust", main = "Correlation Heatmap (After Spatial Sign)")

# Optional: Generate Histograms Before and After Box-Cox Transformation

histograms\_before <- lapply(names(filtered\_data), function(col) {

ggplot(data.frame(value = filtered\_data[[col]]), aes(x = value)) +

geom\_histogram(bins = 30, fill = "lightblue", color = "black") +

ggtitle(paste("Before Box-Cox:", col)) +

xlab(col) + ylab("Count") +

theme\_minimal()

})

histograms\_after <- lapply(names(boxcox\_data), function(col) {

ggplot(data.frame(value = boxcox\_data[[col]]), aes(x = value)) +

geom\_histogram(bins = 30, fill = "lightgreen", color = "black") +

ggtitle(paste("After Box-Cox:", col)) +

xlab(col) + ylab("Count") +

theme\_minimal()

})

# Optional: Generate Boxplots Before and After Spatial Sign Transformation

boxplots\_before <- lapply(names(centered\_scaled\_data), function(col) {

ggplot(data.frame(value = centered\_scaled\_data[[col]]), aes(y = value)) +

geom\_boxplot(fill = "lightblue", color = "black") +

ggtitle(paste("Before Spatial Sign:", col)) +

ylab(col) +

theme\_minimal()

})

# Generate Boxplots After Spatial Sign Transformation

boxplots\_after <- lapply(names(spatial\_sign\_data), function(col) {

ggplot(data.frame(value = spatial\_sign\_data[[col]]), aes(y = value)) +

geom\_boxplot(fill = "lightgreen", color = "black") +

ggtitle(paste("After Spatial Sign:", col)) +

ylab(col) +

theme\_minimal()

})

# Display Plots

cat("Displaying Histograms Before Box-Cox Transformation...\n")

grid.arrange(grobs = histograms\_before, ncol = 3)

cat("Displaying Histograms After Box-Cox Transformation...\n")

grid.arrange(grobs = histograms\_after, ncol = 3)

cat("Displaying Boxplots Before Spatial Sign Transformation...\n")

grid.arrange(grobs = boxplots\_before, ncol = 3)

cat("Displaying Boxplots After Spatial Sign Transformation...\n")

grid.arrange(grobs = boxplots\_after, ncol = 3)

spatial\_sign\_data <- as.data.frame(caret::spatialSign(as.matrix(centered\_scaled\_data)))

names(spatial\_sign\_data) <- names(centered\_scaled\_data)

spatial\_sign\_data <- caret::spatialSign(as.matrix(centered\_scaled\_data))

# Create Training and Testing Sets Using Spatial Sign Data

set.seed(123)

trainIndex <- createDataPartition(wdbc$Diagnosis, p = 0.8, list = FALSE)

spatial\_train <- spatial\_sign\_data[trainIndex, ]

spatial\_test <- spatial\_sign\_data[-trainIndex, ]

train\_labels <- wdbc$Diagnosis[trainIndex]

test\_labels <- wdbc$Diagnosis[-trainIndex]

# Train Control for Repeated Cross-Validation

train\_control <- trainControl(

method = "repeatedcv",

number = 10,

repeats = 5,

summaryFunction = twoClassSummary,

classProbs = TRUE,

savePredictions = TRUE

)

# Logistic Regression

logistic\_model <- train(

x = spatial\_train, y = train\_labels,

method = "glm",

metric = "ROC",

trControl = train\_control

)

print(logistic\_model)

summary(logistic\_model)

plot(logistic\_model)

predicted\_labels <- predict(logistic\_model, newdata = spatial\_test)

predicted\_probs <- predict(logistic\_model, newdata = spatial\_test, type = "prob")

conf\_matrix <- confusionMatrix(factor(predicted\_labels), factor(test\_labels), positive = "M")

print(conf\_matrix)

roc\_auc <- roc(test\_labels, predicted\_probs[, "M"], levels = c("B", "M"))

plot(roc\_auc, main = "ROC Curve - Logistic Regression", col = "blue")

legend("bottomright", legend = paste("AUC:", round(auc(roc\_auc), 3)), col = "blue", lty = 1)

# LDA

lda\_model <- train(

x = spatial\_train, y = train\_labels,

method = "lda",

metric = "ROC",

trControl = train\_control

)

print(lda\_model)

summary(lda\_model)

plot(lda\_model)

predicted\_labels <- predict(lda\_model, newdata = spatial\_test)

predicted\_probs <- predict(lda\_model, newdata = spatial\_test, type = "prob")

conf\_matrix <- confusionMatrix(factor(predicted\_labels), factor(test\_labels), positive = "M")

print(conf\_matrix)

roc\_auc <- roc(test\_labels, predicted\_probs[, "M"], levels = c("B", "M"))

plot(roc\_auc, main = "ROC Curve - LDA", col = "blue")

legend("bottomright", legend = paste("AUC:", round(auc(roc\_auc), 3)), col = "blue", lty = 1)

# QDA

qda\_model <- train(

x = spatial\_train, y = train\_labels,

method = "qda",

metric = "ROC",

trControl = train\_control

)

print(qda\_model)

summary(qda\_model)

plot(qda\_model)

predicted\_labels <- predict(qda\_model, newdata = spatial\_test)

predicted\_probs <- predict(qda\_model, newdata = spatial\_test, type = "prob")

conf\_matrix <- confusionMatrix(factor(predicted\_labels), factor(test\_labels), positive = "M")

print(conf\_matrix)

roc\_auc <- roc(test\_labels, predicted\_probs[, "M"], levels = c("B", "M"))

plot(roc\_auc, main = "ROC Curve - QDA", col = "blue")

legend("bottomright", legend = paste("AUC:", round(auc(roc\_auc), 3)), col = "blue", lty = 1)

# FDA

fda\_model <- train(

x = spatial\_train, y = train\_labels,

method = "fda",

tuneGrid = expand.grid(.degree = 1:2, .nprune = 2:15),

metric = "ROC",

preProc = c("center", "scale"),

trControl = train\_control

)

print(fda\_model)

plot(fda\_model)

predicted\_labels <- predict(fda\_model, newdata = spatial\_test)

predicted\_probs <- predict(fda\_model, newdata = spatial\_test, type = "prob")

conf\_matrix <- confusionMatrix(factor(predicted\_labels), factor(test\_labels), positive = "M")

print(conf\_matrix)

roc\_auc <- roc(test\_labels, predicted\_probs[, "M"], levels = c("B", "M"))

plot(roc\_auc, main = "ROC Curve - FDA", col = "blue")

legend("bottomright", legend = paste("AUC:", round(auc(roc\_auc), 3)), col = "blue", lty = 1)

# KNN

knn\_model <- train(

x = spatial\_train, y = train\_labels,

method = "knn",

tuneGrid = data.frame(.k = 1:25),

metric = "ROC",

trControl = train\_control

)

print(knn\_model)

plot(knn\_model)

predicted\_labels <- predict(knn\_model, newdata = spatial\_test)

predicted\_probs <- predict(knn\_model, newdata = spatial\_test, type = "prob")

conf\_matrix <- confusionMatrix(factor(predicted\_labels), factor(test\_labels), positive = "M")

print(conf\_matrix)

roc\_auc <- roc(test\_labels, predicted\_probs[, "M"], levels = c("B", "M"))

plot(roc\_auc, main = "ROC Curve - KNN", col = "blue")

legend("bottomright", legend = paste("AUC:", round(auc(roc\_auc), 3)), col = "blue", lty = 1)

# RDA

rda\_model <- train(

x = spatial\_train, y = train\_labels,

method = "rda",

tuneGrid = expand.grid(.gamma = seq(0, 1, length = 10), .lambda = seq(0, 1, length = 15)),

metric = "ROC",

trControl = train\_control

)

print(rda\_model)

plot(rda\_model)

predicted\_labels <- predict(rda\_model, newdata = spatial\_test)

predicted\_probs <- predict(rda\_model, newdata = spatial\_test, type = "prob")

conf\_matrix <- confusionMatrix(factor(predicted\_labels), factor(test\_labels), positive = "M")

print(conf\_matrix)

roc\_auc <- roc(test\_labels, predicted\_probs[, "M"], levels = c("B", "M"))

plot(roc\_auc, main = "ROC Curve - RDA", col = "blue")

legend("bottomright", legend = paste("AUC:", round(auc(roc\_auc), 3)), col = "blue", lty = 1)

# MDA

mda\_model <- train(

x = spatial\_train, y = train\_labels,

method = "mda",

tuneGrid = expand.grid(.subclasses = seq(1, 10, by = 1)),

metric = "ROC",

trControl = train\_control

)

print(mda\_model)

plot(mda\_model)

predicted\_labels <- predict(mda\_model, newdata = spatial\_test)

predicted\_probs <- predict(mda\_model, newdata = spatial\_test, type = "prob")

conf\_matrix <- confusionMatrix(factor(predicted\_labels), factor(test\_labels), positive = "M")

print(conf\_matrix)

roc\_auc <- roc(test\_labels, predicted\_probs[, "M"], levels = c("B", "M"))

plot(roc\_auc, main = "ROC Curve - MDA", col = "blue")

legend("bottomright", legend = paste("AUC:", round(auc(roc\_auc), 3)), col = "blue", lty = 1)

## Correlated Data Code

# Load required libraries

library(caret)

library(corrplot)

wdbc <- read.csv("C:\\Users\\Puja\\Desktop\\Predictive\\Data\\wdbc.data", header = TRUE)

wdbc

wdbc <- wdbc[, -1]

numeric\_wdbc <- wdbc[, sapply(wdbc, is.numeric)]

diagnosis <- wdbc$Diagnosis

correlations <- cor(numeric\_wdbc)

corrplot(correlations, order = "hclust", main = "Correlation Matrix")

length(numeric\_wdbc)

unfiltered\_preprocess <- preProcess(numeric\_wdbc, method = c("center", "scale"))

scaled\_unfiltered <- predict(unfiltered\_preprocess, numeric\_wdbc)

unfiltered\_dataset <- data.frame(scaled\_unfiltered, Diagnosis = diagnosis)

unfiltered\_dataset$Diagnosis <- factor(unfiltered\_dataset$Diagnosis, levels = c("B", "M"))

set.seed(123)

trainIndex <- createDataPartition(unfiltered\_dataset$Diagnosis, p = 0.8, list = FALSE)

unfiltered\_train <- unfiltered\_dataset[trainIndex, ]

unfiltered\_test <- unfiltered\_dataset[-trainIndex, ]

#plsda

pls\_model <- train(

x = unfiltered\_train[, -ncol(unfiltered\_train)], y = unfiltered\_train$Diagnosis,

method = "pls",

tuneGrid = expand.grid(.ncomp = 1:15),

metric = "ROC",

trControl = trainControl(method = "repeatedcv", number = 10, repeats = 5, classProbs = TRUE, summaryFunction = twoClassSummary)

)

pls\_model

plot(pls\_model)

predicted\_labels <- predict(pls\_model, newdata = unfiltered\_test[, -ncol(unfiltered\_test)])

predicted\_probs <- predict(pls\_model, newdata = unfiltered\_test[, -ncol(unfiltered\_test)], type = "prob")

conf\_matrix <- confusionMatrix(factor(predicted\_labels), factor(unfiltered\_test$Diagnosis), positive = "M")

print(conf\_matrix)

roc\_auc <- roc(unfiltered\_test$Diagnosis, predicted\_probs[, "M"], levels = c("B", "M"))

plot(roc\_auc, main = "ROC Curve - PLSDA", col = "blue")

legend("bottomright", legend = c(paste("AUC:", round(auc(roc\_auc), 3))), col = "blue", lty = 1)

#penalized models

glmnet\_model <- train(

x = unfiltered\_train[, -ncol(unfiltered\_train)], y = unfiltered\_train$Diagnosis,

method = "glmnet",

tuneGrid = expand.grid(.alpha = c(0, .1, .2, .4, .6, .8, 1), .lambda = seq(.01, .2, length = 10)),

metric = "ROC",

trControl = trainControl(method = "repeatedcv", number = 10, repeats = 5, classProbs = TRUE, summaryFunction = twoClassSummary)

)

glmnet\_model

plot(glmnet\_model)

predicted\_labels <- predict(glmnet\_model, newdata = unfiltered\_test[, -ncol(unfiltered\_test)])

predicted\_probs <- predict(glmnet\_model, newdata = unfiltered\_test[, -ncol(unfiltered\_test)], type = "prob")

conf\_matrix <- confusionMatrix(factor(predicted\_labels), factor(unfiltered\_test$Diagnosis), positive = "M")

print(conf\_matrix)

roc\_auc <- roc(unfiltered\_test$Diagnosis, predicted\_probs[, "M"], levels = c("B", "M"))

plot(roc\_auc, main = "ROC Curve - Penalized Logistic Regression", col = "blue")

legend("bottomright", legend = c(paste("AUC:", round(auc(roc\_auc), 3))), col = "blue", lty = 1)

#svm

svm\_model <- train(

x = unfiltered\_train[, -ncol(unfiltered\_train)], y = unfiltered\_train$Diagnosis,

method = "svmRadial",

tuneGrid = expand.grid(.sigma = sigest(as.matrix(unfiltered\_train[, -ncol(unfiltered\_train)]))[1], .C = 2^(seq(-4, 4))),

metric = "ROC",

preProc = c("center", "scale"),

trControl = trainControl(method = "repeatedcv", number = 10, repeats = 5, classProbs = TRUE, summaryFunction = twoClassSummary)

)

svm\_model

plot(svm\_model)

predicted\_labels <- predict(svm\_model, newdata = unfiltered\_test[, -ncol(unfiltered\_test)])

predicted\_probs <- predict(svm\_model, newdata = unfiltered\_test[, -ncol(unfiltered\_test)], type = "prob")

conf\_matrix <- confusionMatrix(factor(predicted\_labels), factor(unfiltered\_test$Diagnosis), positive = "M")

print(conf\_matrix)

roc\_auc$auc

roc\_auc <- roc(unfiltered\_test$Diagnosis, predicted\_probs[, "M"], levels = c("B", "M"))

plot(roc\_auc, main = "ROC Curve - SVM", col = "blue")

legend("bottomright", legend = c(paste("AUC:", round(auc(roc\_auc), 3))), col = "blue", lty = 1)

varImp(svm\_model)

# Naive Bayes

nb\_model <- train(

x = unfiltered\_train[, -ncol(unfiltered\_train)],

y = unfiltered\_train$Diagnosis,

method = "nb",

tuneGrid = data.frame(.fL = 2,.usekernel = TRUE,.adjust = TRUE),

metric = "ROC",

trControl = trainControl(method = "repeatedcv",

number = 10,

repeats = 5,

classProbs = TRUE,

summaryFunction = twoClassSummary)

)

print(nb\_model)

plot(nb\_model)

predicted\_labels <- predict(nb\_model, newdata = unfiltered\_test[, -ncol(unfiltered\_test)])

predicted\_probs <- predict(nb\_model, newdata = unfiltered\_test[, -ncol(unfiltered\_test)], type = "prob")

conf\_matrix <- confusionMatrix(factor(predicted\_labels), factor(unfiltered\_test$Diagnosis), positive = "M")

print(conf\_matrix)

roc\_auc <- roc(unfiltered\_test$Diagnosis, predicted\_probs[, "M"], levels = c("B", "M"))

plot(roc\_auc, main = "ROC Curve - Naive Bayes", col = "blue")

legend("bottomright", legend = c(paste("AUC:", round(auc(roc\_auc), 3))), col = "blue", lty = 1)

# Neural Network

nn\_model <- train(

x = unfiltered\_train[, -ncol(unfiltered\_train)],

y = unfiltered\_train$Diagnosis,

method = "nnet",

tuneGrid = expand.grid(.size = 1:9, .decay = c(0, .1, 1, 2)),

metric = "ROC",

preProc = c("center", "scale"),

trControl = trainControl(method = "repeatedcv",

number = 10,

repeats = 5,

classProbs = TRUE,

summaryFunction = twoClassSummary),

trace = FALSE

)

print(nn\_model)

plot(nn\_model)

predicted\_labels <- predict(nn\_model, newdata = unfiltered\_test[, -ncol(unfiltered\_test)])

predicted\_probs <- predict(nn\_model, newdata = unfiltered\_test[, -ncol(unfiltered\_test)], type = "prob")

conf\_matrix <- confusionMatrix(factor(predicted\_labels), factor(unfiltered\_test$Diagnosis), positive = "M")

print(conf\_matrix)

roc\_auc$auc

roc\_auc <- roc(unfiltered\_test$Diagnosis, predicted\_probs[, "M"], levels = c("B", "M"))

plot(roc\_auc, main = "ROC Curve - Neural Network", col = "blue")

legend("bottomright", legend = c(paste("AUC:", round(auc(roc\_auc), 3))), col = "blue", lty = 1)