Building Statistical Learning Models to Predict Cancer Related Deaths Rates per Capita

Module 3: Classification

Team Leader: Brandon Coates

Assistant Leader: Samantha Hair

Team Members: Lorenzo Gordon, Shaelyn Doyle, Trinidad Cisneros

Classification Models

Data Pre-Processing and Splitting

In order to test if classification based predictive models could yield greater accuracies than the previously examined models, the quantitative response variable of cancer deaths per capita (per 100,000 people) needed to be transformed into a categorical response variable. To do this, our team decided to compare the reported cancer deaths per capita to the mean cancer deaths per capita reported by the CDC[1]. This created a binary response variable in which the levels dictated if the reported cancer death rate was above or below the average cancer death rate per capita in the United States. The responses that were below the national average were classified as "L" and those that were above were classified as "H". This response variable will allow for the examination of how demographical factors are correlated with counties in which the cancer death rates are above the national average. Once the response variable was transformed into the binary response, the data was cleaned by eliminating variables that are highly correlated with the response, variables with a large amount of missing data, and highly correlated variables. The predictor variables were also examined for skewness, where it was found that most variables were highly skewed. In order to remedy this issue, the predictor variables were centered, scaled, and transformed using the Box Cox transformation method after a constant of 0.5 was added to the variables. This constant was added in order to allow for implementation of the Box Cox transformation since a large amount of variables contained a value of 0, which the Box Cox method cannot work on. After the transformation was conducted, it was found that the skewness was reduced to allow for the predictor variables to exhibit approximately normal distributions. It should be noted that this is the same pre-processing approach that is used for the rest of the models built during this effort.

Once the data was transformed to remove the skewness present within the predictor variables, the data was split so that a training data set that consisted of 80% of the total data was formed. The remaining 20% of the data was assigned to the testing/validation data set. Stratified random sampling was used in order to randomly assign observations to these data sets that would be used to train and test the models constructed. The resulting training data set contained a total of 2,437 observations, while the remaining 610 observations were assigned to the testing/validation data set.

After data splitting had been conducted, classification models were trained using the training data set. Cross-validation was utilized in the form of Leave One Group Out Cross-Validation (LGOCV) and the tuning parameters, if applicable, were chosen using the area under the ROC curve as the selection criteria. The models that were trained during this step included Logistic Regression, Naïve Bayes Classification, Linear Discriminant Analysis, Quadratic Discriminant Analysis, and Support Vector Machine

[1] https://www.cdc.gov/cancer/dcpc/research/update-on-cancer-deaths/index.htm#:~:text=Healthy%20People%202030%20set%20an,124.5%20deaths%20per%20100%2C000%20population)

Classification. In the sections below, tables that include summaries of the model performance on the test set after the model was trained along with their respective ROC curves can be seen.

Model Training and Testing

Logistic Regression

A logistic regression model was fit to the training set data and then used to predict the results of the testing/validation data set. It was observed that when predicting the response of the validation data, the model correctly predicted 551 of the observation responses, while misclassifying 59 of the responses, giving the model an accuracy of 90.3% on the testing data. **Table 1** shows the summary of the model performance on the testing data. **Figure 1** also shows the ROC curve for the model. It was observed that the most important variables in this model were the percentage of population over 25 years old that had a high school graduation, the percentage of the population that was black, and the percentage of the households that were married. This can be seen in **Figure 2** which shows the variable importance plot of the logistic regression model.

Table 1. Logistic Regression Performance on Testing Data Summary

Accuracy	Misclassification Rate	Kappa Statistic	Sensitivity	Specificity	AUC
90.3%	9.7%	0.181	0.133	0.987	0.817

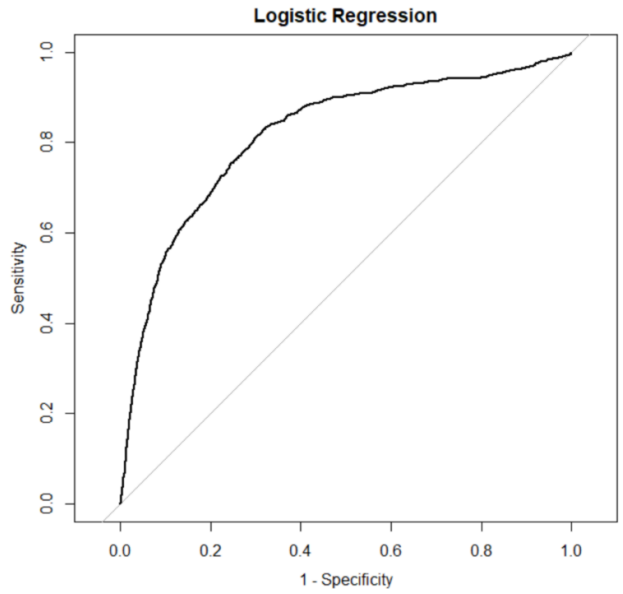

Figure 1. ROC Curve for Logistic Regression Model

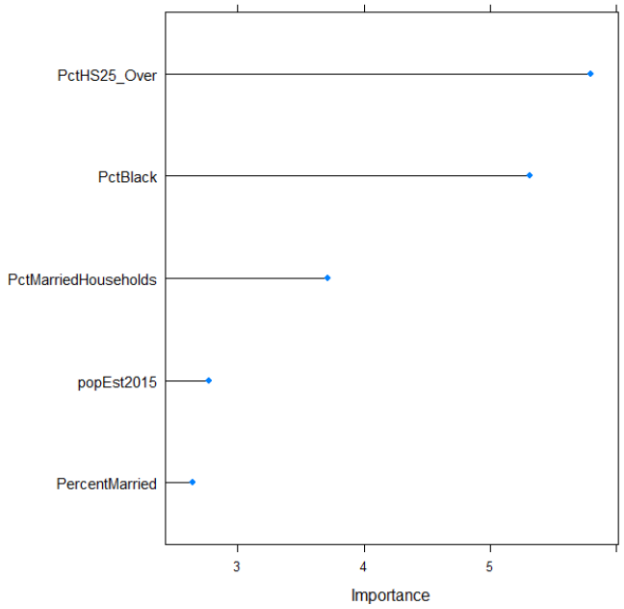

Figure 2. Variable Importance Plot for Logistic Regression Model

Naïve Bayes Model

A Naïve Bayes model was also trained on the training data and used to predict the response variable of the testing data observations. It was found that this model correctly predicted the response of the testing data observations 520 time and incorrectly predicted the response 90 times, giving the model an accuracy of 85.3%. **Table 2** shows the summary of the model performance on the testing data. **Figure 3** also shows the ROC curve for the model. It was observed that the most important variables in this model were the percentage of population over 16 years old that are unemployed, the percentage of the population that was black, and the percentage of the population over 25 years old that has a high school education. This can be seen in **Figure 4** which shows the variable importance plot of the logistic regression model.

Table 2. Naïve Bayes Performance on Testing Data

Accuracy	Misclassification Rate	Kappa Statistic	Sensitivity	Specificity	AUC
85.3%	14.7%	0.351	0.567	0.884	0.820

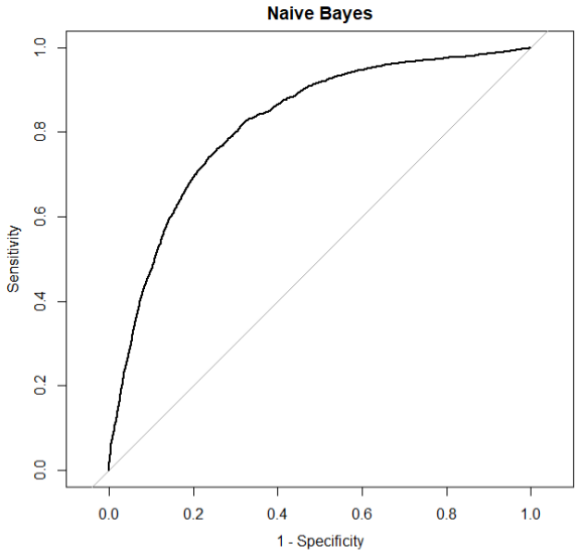

Figure 3. ROC Curve for Naïve Bayes Model

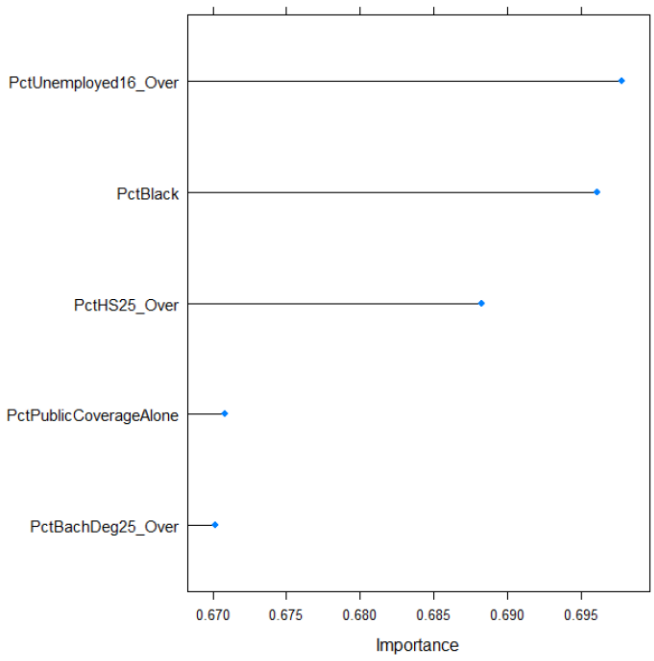

Figure 4. Variable Importance Plot for Naïve Bayes Model

Linear Discriminant Analysis

A linear discriminant analysis model was also trained on the training data and used to predict the response variable of the testing data observations. It was found that this model correctly predicted the response of the testing data observations 547 times and incorrectly predicted the response 63 times, giving the model an accuracy of 89.7%. **Table 3** shows the summary of the model performance on the testing data. **Figure**

5 also shows the ROC curve for the model. It was observed that the most important variables in this model were the percentage of population over 16 years old that are unemployed, the percentage of the population that was black, and the percentage of the population over 25 years old that has a high school education. This can be seen in **Figure 6** which shows the variable importance plot of the logistic regression model.

Table 3. Linear Discriminant Performance on Testing Data

Accuracy	Misclassification Rate	Kappa Statistic	Sensitivity	Specificity	AUC
89.7%	10.3%	0.213	0.183	0.975	0.829

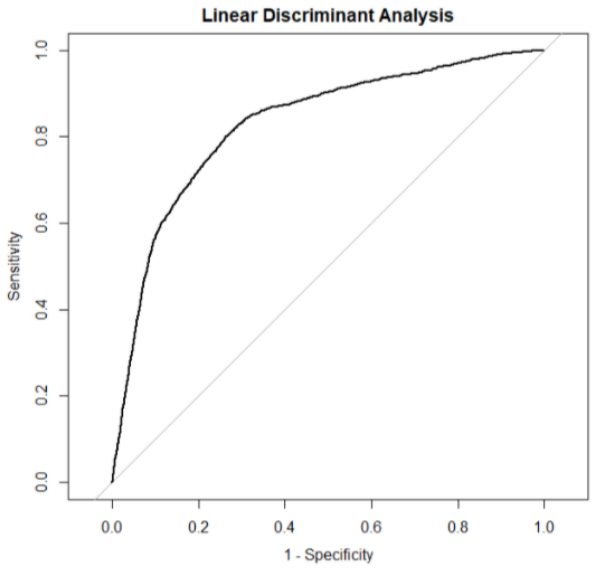

Figure 5. ROC Curve for Linear Discriminant Analysis Model

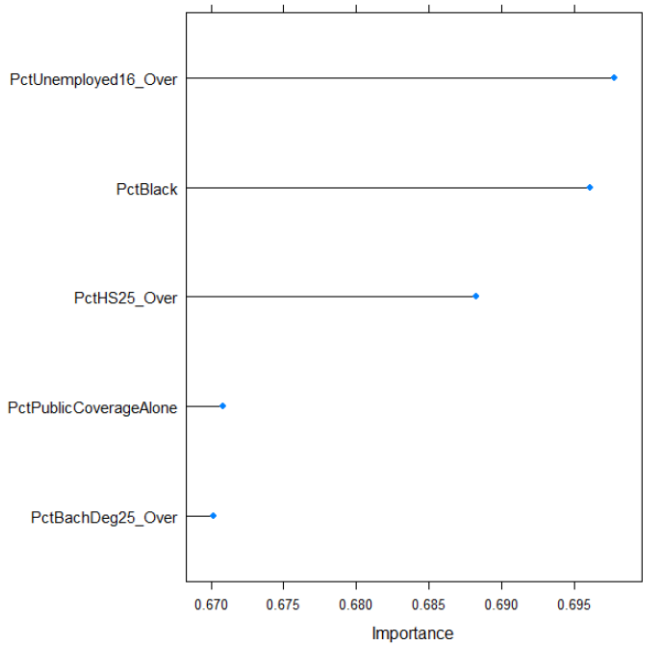

Figure 6. Variable Importance Plot for Linear Discriminant Analysis

Quadratic Discriminant Analysis

A quadratic discriminant analysis model was also trained on the training data and used to predict the response variable of the testing data observations. It was found that this model correctly predicted the response of the testing data observations 527 times and incorrectly predicted the response 83 times, giving the model an accuracy of 86.4%. **Table 4** shows the summary of the model performance on the testing data. **Figure 7** also shows the ROC curve for the model. It was observed that the most important variables in this model were the percentage of population over 16 years old that are unemployed, the percentage of the population that was black, and the percentage of the population over 25 years old that has a high school education. This can be seen in **Figure 8** which shows the variable importance plot of the logistic regression model.

Table 4. Quadratic Discriminant Performance on Testing Data

Accuracy	Misclassification Rate	Kappa Statistic	Sensitivity	Specificity	AUC
86.4%	13.6%	0.361	0.900	0.533	0.833

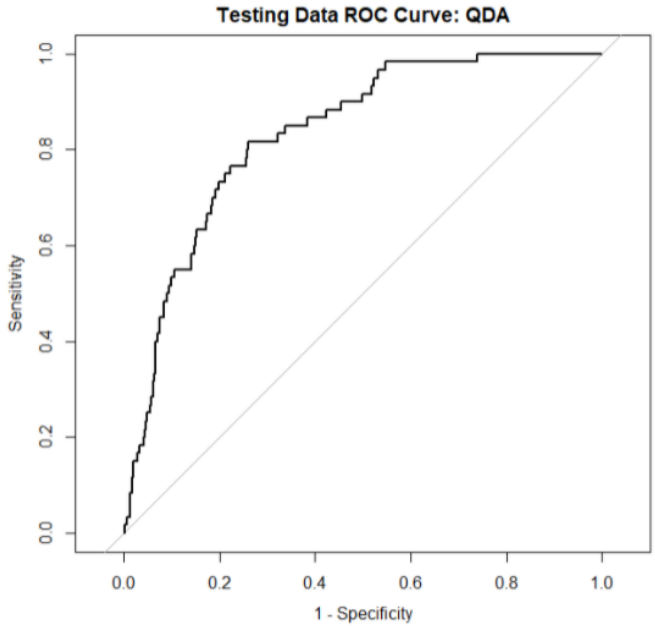

Figure 7. ROC Curve for Quadratic Discriminant Analysis Model

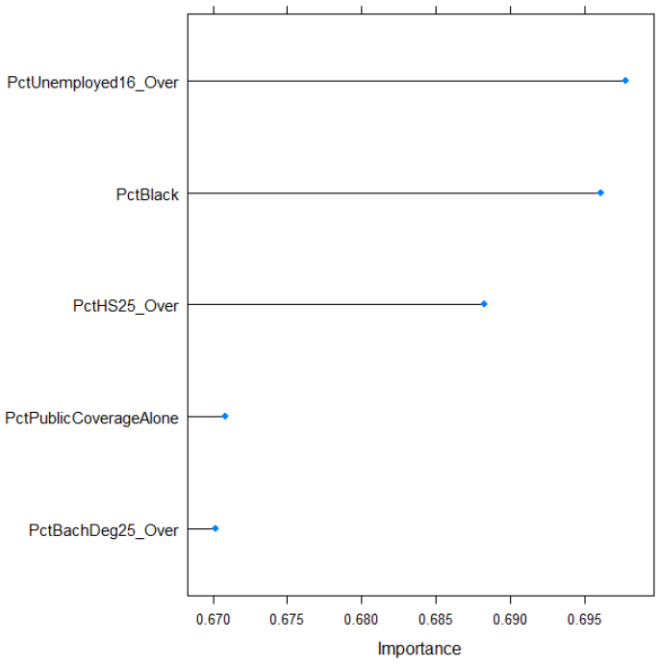

Figure 8. Variable Importance Plot for Quadratic Discriminant Model

Support Vector Machine

A support vector machine model was also trained on the training data and used to predict the response variable of the testing data observations. For this model, it was found through cross-validation that the optimum tuning parameters were $\sigma = 0.05475011$ and $Cost = 1$. The plot of the tuning parameters can be seen in **Figure 9**. It was found that this model correctly predicted the response of the testing data

observations 560 times and incorrectly predicted the response 50 times, giving the model an accuracy of 91.8%. **Table 5** shows the summary of the model performance on the testing data. **Figure 10** also shows the ROC curve for the model. It was observed that the most important variables in this model were the percentage of population over 16 years old that are unemployed, the percentage of the population that was black, and the percentage of the population over 25 years old that has a high school education. This can be seen in **Figure 11** which shows the variable importance plot of the logistic regression model.

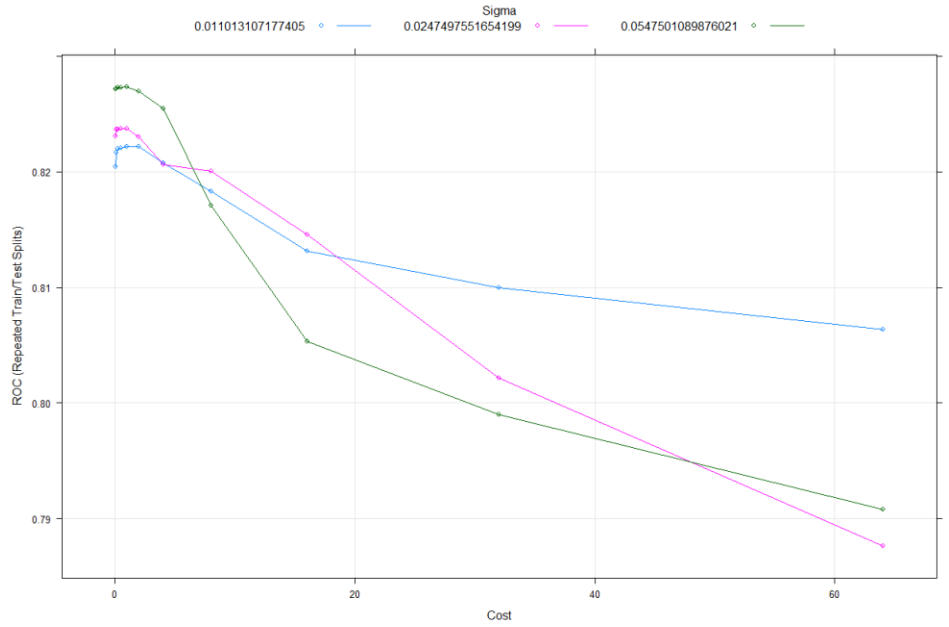

Figure 9. Tuning Parameter Plot for Support Vector Machine Model

Table 5. Support Vector Machine Performance on Testing Data

Accuracy	Misclassification Rate	Kappa Statistic	Sensitivity	Specificity	AUC
91.8%	8.2%	0.329	0.993	0.233	0.825

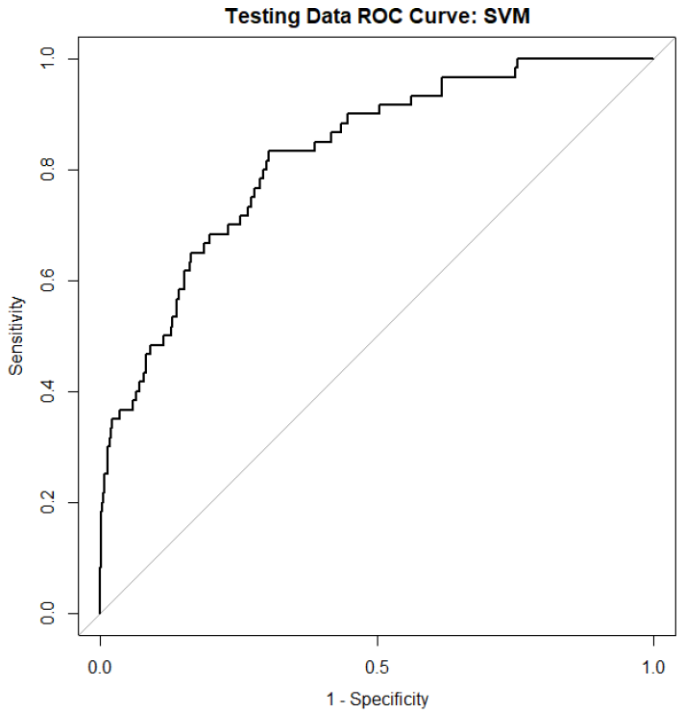

Figure 10. ROC Curve for Support Vector Machine Model

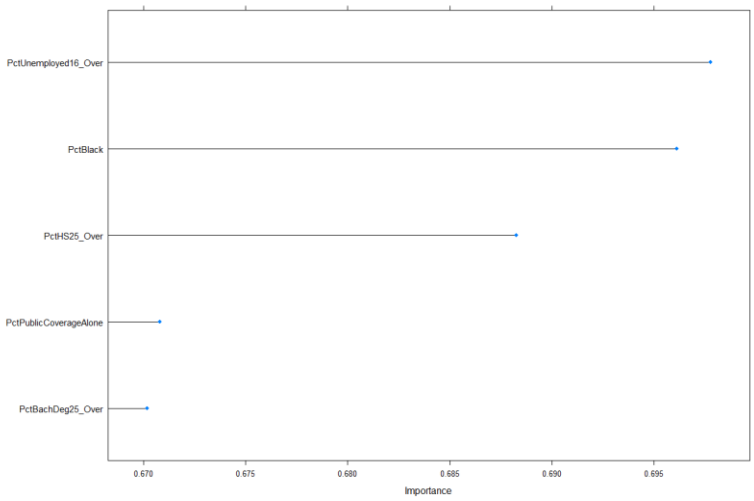

Figure 11. Variable Importance Plot for Support Vector Machine

Model Results and Conclusion

After the models were trained and tested on the test set data, the performance statistics of each model was examined in order to determine which model performed the best in predicting on whether the cancer death rate per capita was higher or lower than the average death rate per county in the United States. **Table 6** shows the performance statistics of each classification model that was trained during this section.

Table 6. Performance Statistics of Classification Models

Model	Accuracy	Misclassification Rate	Kappa Statistic	Sensitivity	Specificity	AUC
Logistic Regression	90.3%	9.7%	0.181	0.133	0.987	0.817
Naïve Bayes	85.3%	14.7%	0.351	0.567	0.884	0.820
LDA	89.7%	10.3%	0.213	0.183	0.975	0.829
QDA	86.4%	13.6%	0.361	0.900	0.533	0.833
SVM	91.8%	8.2%	0.329	0.993	0.233	0.825

After reviewing the misclassification rates of all of the models generated on the test data set, it was found that the support vector machine model was the best performer by only this metric. This model is the best performing model due to the fact that this method is able to generate nonlinear class boundaries that are able to separate each class of the response variable due to the kernel function that is implemented in the model. By allowing for the nonlinear class boundaries to be generated, the model is able to generate boundaries that are able to identify the response to a higher degree of accuracy than the other linear classification models that were examined. The downside to this is the fact that the SVM model has a high computation time, which was observed in this effort since it exhibited the longest running time of all models examined when trained. Another downside that was observed in this effort was that the specificity of the support vector machine model was low, meaning that the model would tend to be biased to predicting the cancer death rate as higher than the national average, so there would be a higher chance of false positives in the model. This may suggest that the quadratic discriminant analysis model would be a better fit since it also has favorable performance statistics, but a slightly higher misclassification rate, while not being as biased towards producing false positives. This would be up to the modeler and the customer, but for this effort, the misclassification rate was flagged as the main performance metric, so it is suggested that the support vector machine model be used.

R CODE

```
#Load Required Libraries

library(caret)

library(corrplot)

library(pls)

library(mlbench)

library(e1071)

library(pROC)

library(MASS)

library(kernlab)

library(earth)

library(klaR)

library(svmpath)


#Load Data Set

setwd("C:/Users/bjc_2/Documents/MA5751/Week 2")

cancer <- read.csv("cancer_reg.csv")


#Remove un-wanted predictors

GeographyIndex <- grep("Geography", colnames(cancer))

binnedIncIndex <- grep("binnedInc", colnames(cancer))

AvgDeathsIndex <- grep("avgDeathsPerYear",colnames(cancer))

AvgCountIndex <- grep("avgAnnCount", colnames(cancer))

incidenceIndex <- grep("incidenceRate", colnames(cancer))

cancer <- cancer[, -c(GeographyIndex, binnedIncIndex, AvgCountIndex, AvgDeathsIndex,
incidenceIndex)]


#Check for NA's in response variable

TARGET_deathRateNAs <- is.na(cancer$TARGET_deathRate)

NAsTrue <- grep("TRUE", TARGET_deathRateNAs)
```

```
NAsTrue


#Check for Na's in predictor variables
NAs <- is.na(cancer)
NAsTrue <- grep("TRUE", NAs)
length(NAsTrue)


#Remove Predictors with large NA counts
PctSomeCol18_24Index <- grep("PctSomeCol18_24", colnames(cancer))
PctEmployed16_OverIndex <- grep("PctEmployed16_Over", colnames(cancer))
PctPrivateCoverageAloneIndex <- grep("PctPrivateCoverageAlone", colnames(cancer))
cancer <- cancer[, -c(PctSomeCol18_24Index, PctEmployed16_OverIndex,
                      PctPrivateCoverageAloneIndex)]


# Convert Response Variable into binary
colnames(cancer)
death_mean <- 144.1
breaks  <- c(-Inf, death_mean, Inf)
labels  <- c("L", "H")  #0 - lower, 1-higher
binary_deathRate  <- cut(x=cancer$TARGET_deathRate, breaks=breaks, labels=labels, right=F)
cancer <- data.frame(cancer, binary_deathRate)
head(cancer$binary_deathRate) # confirm dataframe includes new variable


# drop original response variable
cancer = subset(cancer, select = -c(TARGET_deathRate) )
head(cancer$binary_deathRate)


#Subset data before transforming, dont want to transform a binary variable
cancer_pred = subset(cancer, select = -c(binary_deathRate) ) + 0.5
cancer_resp = subset(cancer, select = c(binary_deathRate) )
```

```
#Check skewness of predictor variables
skew <- apply(cancer_pred, 2, skewness)
skew

#Perform centering, scaling, and Box Cox transformation on predictor variables
trans <- preProcess(cancer_pred, method = c("center", "scale", "BoxCox"))
trans

#Apply the transformation to the data set
cancerTrans <- predict(trans, cancer_pred)

#Check skewness of predictor variables after transformation
skew.transform <- apply(cancerTrans, 2, skewness)
skew.transform

#Check for near zero variance predictor variables
nearZeroVar(cancerTrans) # None

#Construct a correlation plot of data
par(mfrow = c(1,1))
corrplot::corrplot(cor(cancerTrans), order = "hclust")

#Identify highly correlated predictor variables
cor90 <- findCorrelation(cor(cancerTrans), cutoff = 0.90)
cor90

# view which column is high correlation
colnames(cancerTrans)[cor90]
```

```
#Remove highly correlated predictors
cancerTrans <- cancerTrans[-c(cor90)]

# combine predictor and response dataframes again
transformed_df <- data.frame(cancer_resp, cancerTrans)
transformed_df$binary_deathRate <- as.factor(transformed_df$binary_deathRate)
head(transformed_df, 2)

#Conduct data splitting
set.seed(0)
trainIndex <- sample(1:nrow(transformed_df), nrow(transformed_df)*0.8, replace = FALSE)
length(trainIndex)

#Define training set and testing set
cancerTransTrain <- transformed_df[trainIndex,]
cancerTransTest <- transformed_df[-c(trainIndex),]
head(cancerTransTest)

#split output and predictors
TARGET_deathRateIdx <- grep("binary_deathRate", colnames(cancerTransTrain))
trainCanP <- cancerTransTrain[, -TARGET_deathRateIdx]
trainCanR <- cancerTransTrain[, TARGET_deathRateIdx]
testCanP <- cancerTransTest[, -TARGET_deathRateIdx]
testCanR <- cancerTransTest[, TARGET_deathRateIdx]

head(cancerTransTrain)

#Set cross-validation control
ctrl <- trainControl(method = "LGOCV",
                     summaryFunction = twoClassSummary,
```

```
                classProbs = TRUE,
                savePredictions = TRUE)

# logistic regression
# train the model on training set
lrfit <- train(binary_deathRate ~.,
               data = cancerTransTrain,
               trControl = ctrl,
               method = "glm",
               family=binomial(),
               metric = "ROC")

# print cv scores
summary(lrfit)

#confusion matrix - training data
confusionMatrix(data = lrfit$pred$pred,
                reference = lrfit$pred$obs)

#ROC curve - training data
trainROC.lr <- roc(response = lrfit$pred$obs,
                   predictor = lrfit$pred$H,
                   levels = rev(levels(lrfit$pred$obs)))
par(mfrow = c(1,1))
plot(trainROC.lr, legacy.axes = TRUE, main = "Training Data ROC Curve: Logistic Regression")
auc(trainROC.lr)


# predict on testing
lrPred <- predict(lrfit, testCanP)
```

```
summary(lrPred)
lrValues <- postResample(pred = lrPred, obs = testCanR)
lrValues


confusionMatrix(data = as.factor(lrPred),
                reference = testCanR)


# AUC
lrRoc <- roc(response = lrfit$pred$obs,
             predictor = lrfit$pred$H,
             levels = rev(levels(lrfit$pred$obs)))
plot(lrRoc, legacy.axes = TRUE, main = "Logistic Regression")
lrAUC <- auc(lrRoc)
lrAUC

glmnImp <- varImp(lrfit, scale = FALSE)
glmnImp
plot(glmnImp, top = 5, scales = list(y = list(cex = .95)), main = "Most Important Variables")



# Naive Bayes
set.seed(1000)
nbFit = train(binary_deathRate ~.,
              data = cancerTransTrain,
              trControl = ctrl,
              method = "nb",
              metric = "ROC")
summary(nbFit)
par(mfrow = c(1,1))
plot(nbFit)
```

```
nbImpSim <- varImp(nbFit, scale = FALSE)
nbImpSim
plot(nbImpSim, top = 5, scales = list(y = list(cex = .95)))

# predict on testing
nbPred <- predict(nbFit, testCanP)
summary(nbPred)
nbValues <- postResample(pred = nbPred, obs = testCanR)
nbValues

confusionMatrix(data = as.factor(nbPred),
   reference = testCanR)

# AUC
nbRoc <- roc(response = nbFit$pred$obs,
             predictor = nbFit$pred$H,
             levels = rev(levels(nbFit$pred$obs)))
plot(nbRoc, legacy.axes = TRUE, main = "Naive Bayes")
nbAUC <- auc(nbRoc)
nbAUC



# LDA
set.seed(1000)
ldaFit <- train(x = trainCanP,
                y = trainCanR,
                method = "lda",
                preProc = c("center", "scale"),
                metric = "ROC",
```

```
            trControl = ctrl)
ldaFit

# predict on test set
ldaPred = predict(ldaFit, testCanP)
ldaValues <- postResample(pred = ldaPred, obs = testCanR)
ldaValues

# test set
confusionMatrix(data = ldaPred,
            reference = testCanR)

ldaImpSim <- varImp(ldaFit, scale = FALSE)
ldaImpSim
plot(ldaImpSim, top = 5, scales = list(y = list(cex = .95)))

# AUC
ldaRoc <- roc(response = ldaFit$pred$obs,
            predictor = ldaFit$pred$H,
            levels = rev(levels(ldaFit$pred$obs)))
plot(ldaRoc, legacy.axes = TRUE, main = "Linear Discriminant Analysis")
ldaAUC <- auc(ldaRoc)
ldaAUC

#QDA
qdaFit <- train(x = trainCanP,
            y = trainCanR,
            method = "qda",
            preProc = c("center", "scale"),
            metric = "ROC",
```

```
            trControl = ctrl)
qdaFit

qdaImpSim <- varImp(qdaFit, scale = FALSE)
qdaImpSim
plot(qdaImpSim, top = 5, scales = list(y = list(cex = .95)))

#Predict on Test Data
testCanP$QDA.class <- predict(qdaFit, testCanP)
predict.QDA <- predict(qdaFit, testCanP, type="prob")
testCanP$QDAprob <- predict.QDA[,"H"]
testCanP$obs <- as.factor(testCanR)

#confusion matrix - testing data
confusionMatrix(data = testCanP$QDA.class,
            reference = testCanP$obs,
            positive = "H")

#ROC Curve - testing data
testROC.QDA <- roc(response = testCanP$obs,
              predictor = testCanP$QDAprob,
              levels = rev(levels(testCanP$obs)))
plot(testROC.QDA, legacy.axes = TRUE, main = "Testing Data ROC Curve: QDA")
auc(testROC.QDA)

# SVM
set.seed(1000)
#Define svmgrid
sigmaRangeReduced <- sigest(as.matrix(trainCanP))
svmGrid <- expand.grid(.sigma = sigmaRangeReduced,
```

```
                .C = 2^(seq(-4, 6)))

SVMtrain <- train(binary_deathRate ~.,
                  data = cancerTransTrain,
                  method = "svmRadial",
                  metric = "ROC",
                  tuneGrid = svmGrid,
                  trControl = ctrl)
SVMtrain
plot(SVMtrain)

svmImpSim <- varImp(SVMtrain, scale = FALSE)
svmImpSim
plot(svmImpSim, top = 5, scales = list(y = list(cex = .95)))

#Predict on Test Data
testCanP$SVM.class <- predict(SVMtrain, testCanP)
predict.SVM <- predict(SVMtrain, testCanP, type="prob")
testCanP$SVMprob <- predict.SVM[,"H"]
testCanP$obs <- as.factor(testCanR)

#confusion matrix - testing data
confusionMatrix(data = testCanP$SVM.class,
                reference = testCanP$obs,
                positive = "H")

#ROC Curve - testing data
testROC.SVM <- roc(response = testCanP$obs,
                   predictor = testCanP$SVMprob,
                   levels = rev(levels(testCanP$obs)))
```

```
plot(testROC.SVM, legacy.axes = TRUE, main = "Testing Data ROC Curve: SVM")
auc(testROC.SVM)
```