Final Project Code

Matthew Grohotolski

11/29/2021

## Import Libraries

library(ISLR)  
library(gbm)

library(rpart)  
library(rpart.plot)  
library(glmnet)

library(tree)  
library(randomForest)

library(Rcpp)  
library(caret)

library(yardstick)

library(ggplot2)

# Import raw data

Covid.data\_raw = read.csv("~/Downloads/Covid data\_raw.csv", header=TRUE)  
#View(Covid.data\_raw)

# Data Cleaning

# Drop rows with na values  
Covid.data = na.omit(Covid.data\_raw)  
  
# Drop YEAR, NARSTAT, NHISHID, HISPETH  
Covid.data = subset(Covid.data, select=-c(YEAR, NHISHID, HISPETH))  
  
# Drop AGE 997, 999  
Covid.data = Covid.data[!(Covid.data$AGE==997 | Covid.data$AGE==999),]  
  
# Drop SEX 7, 8, 9  
Covid.data = Covid.data[!(Covid.data$SEX==7 | Covid.data$SEX==8 | Covid.data$SEX==9),]  
  
# Drop MARSTAT 0, 99  
Covid.data = Covid.data[!(Covid.data$MARSTAT==0 | Covid.data$MARSTAT==99),]  
  
# Drop RACEA 900, 970, 980, 990  
Covid.data = Covid.data[!(Covid.data$RACEA==900 | Covid.data$RACEA==970 | Covid.data$RACEA==980 | Covid.data$RACEA==990),]  
  
# Drop educ 0, 996, 997, 998, 999  
Covid.data = Covid.data[!(Covid.data$EDUC==0 | Covid.data$EDUC==996 | Covid.data$EDUC==997 | Covid.data$EDUC==998 | Covid.data$EDUC==999),]  
  
# Drop empstat 0, 900, 997, 998, 999  
Covid.data = Covid.data[!(Covid.data$EMPSTAT==0 | Covid.data$EMPSTAT==900 | Covid.data$EMPSTAT==997 | Covid.data$EMPSTAT==998 | Covid.data$EMPSTAT==999),]  
  
# Drop INCFAM07ON 96, 99  
Covid.data = Covid.data[!(Covid.data$INCFAM07ON==96 | Covid.data$INCFAM07ON==99),]  
  
# Drop HINOTCOVE 0, 7, 8, 9  
Covid.data = Covid.data[!(Covid.data$HINOTCOVE==0 | Covid.data$HINOTCOVE==7 | Covid.data$HINOTCOVE==8 | Covid.data$HINOTCOVE==9),]  
  
# Drop CVDDNGCARE 0, 7, 8, 9  
Covid.data = Covid.data[!(Covid.data$CVDDNGCARE==0 | Covid.data$CVDDNGCARE==7 | Covid.data$CVDDNGCARE==8 | Covid.data$CVDDNGCARE==9),]  
  
# Factor REGION, URBRRL, SEX, RACEA, HISPETH, EDUC, EMPSTAT, INCFAM07ON, HINOTCOV, CVDDNGCARE  
Covid.data$REGION = as.factor(Covid.data$REGION)  
Covid.data$URBRRL = as.factor(Covid.data$URBRRL)  
Covid.data$SEX = as.factor(Covid.data$SEX)  
Covid.data$MARSTAT = as.factor(Covid.data$MARSTAT)  
Covid.data$RACEA = as.factor(Covid.data$RACEA)  
Covid.data$EDUC = as.factor(Covid.data$EDUC)  
Covid.data$EMPSTAT = as.factor(Covid.data$EMPSTAT)  
Covid.data$INCFAM07ON = as.factor(Covid.data$INCFAM07ON)  
Covid.data$HINOTCOVE = as.factor(Covid.data$HINOTCOVE)  
Covid.data$CVDDNGCARE = as.factor(Covid.data$CVDDNGCARE)  
  
# Convert SEX, EMPSTAT, HINOTCOV, CVDDNGCARE from 1-2 to 0-1  
Covid.data$SEX = as.factor(unclass(Covid.data$SEX)-1)  
Covid.data$EMPSTAT = as.factor(unclass(Covid.data$EMPSTAT)-1)  
Covid.data$HINOTCOVE = as.factor(unclass(Covid.data$HINOTCOVE)-1)  
Covid.data$CVDDNGCARE = as.factor(unclass(Covid.data$CVDDNGCARE)-1)  
  
str(Covid.data)

# Optional to view dataframe  
# View(Covid.data)

# Modeling

# Perform 80% train split  
train = round(nrow(Covid.data) \* 0.80)  
  
# Train data set  
Covid.train.x = subset(Covid.data[1:train,], select=-c(CVDDNGCARE))  
Covid.train.x.matrix = model.matrix(~.,data=Covid.train.x)  
Covid.train.y = Covid.data[1:train, "CVDDNGCARE"]  
Covid.subset = sample(1:nrow(Covid.data), train)  
  
# Test data set  
Covid.test.x = subset(Covid.data[-train,], select=-c(CVDDNGCARE))  
Covid.test.x.matrix = model.matrix(~.,data=Covid.test.x)  
Covid.test.y = Covid.data[-train, "CVDDNGCARE"]

## 

## Decision Forest

# Create Decision Forest Classifier with training data  
df.covid = rpart(CVDDNGCARE~., data=Covid.data[1:train,], minsplit=1, method='class')  
  
summary(df.covid)

# Plot decision tree (1 root node)  
rpart.plot(df.covid, extra=106)

# Predict with test data  
yhat.df = predict(df.covid, Covid.test.x)  
yhat.df = yhat.df[,2]  
  
# Convert to binomial prediction  
df.pred = rep(0, nrow(Covid.test.x.matrix))  
df.pred[yhat.df > .5] = 1  
  
# Create Confusion Matrix  
table = table(df.pred, Covid.test.y)  
table

table = as.numeric(table)  
  
# Account for Predicting 1 Class  
if(length(table) == 2) {  
 print(table[1] / (table[1] + table[2]))  
} else {  
 print((table[1] + table[4]) / (table[1] + table[2] + table[3] + table[4]))  
}

## Boosting

# Create Boosting classifier with training data  
boost.covid = gbm(as.character(CVDDNGCARE)~., data=Covid.data[1:train,], distribution='bernoulli', n.trees=5000)  
boost.covid

# Predict using test data  
yhat.boost = predict(boost.covid, newdata=Covid.test.x)

# Convert to binomial prediction  
boost.pred = rep(0, nrow(Covid.test.x.matrix))  
boost.pred[yhat.boost > .5] = 1  
  
table = table(boost.pred, Covid.test.y)  
table

table = as.numeric(table)  
  
# Create Confusion Matrix  
# Account for Predicting 1 Class  
if(length(table) == 2) {  
 print(table[1] / (table[1] + table[2]))  
} else {  
 print((table[1] + table[4]) / (table[1] + table[2] + table[3] + table[4]))  
}

## 

## Random Forest

# Create Random Forest Classifier with training data  
# mtry is set automatically to floor(sqrt(ncol(x))) for factor y  
rf.covid = randomForest(CVDDNGCARE~., data=Covid.data, subset=Covid.subset, importance=TRUE, type='classification')  
yhat.df = predict (rf.covid, newdata=Covid.test.x)  
  
# Checkout importance of predictor variables  
varImpPlot(rf.covid)

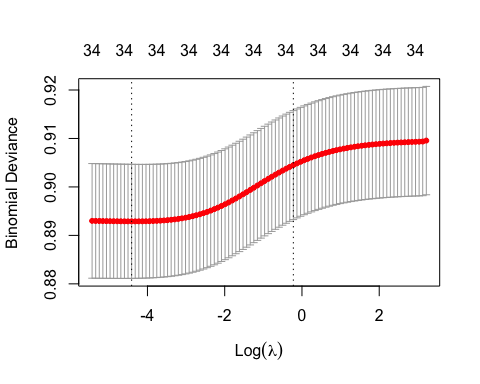
rf.pred = predict(rf.covid, Covid.test.x)  
  
table = table(rf.pred, Covid.test.y)  
table

table = as.numeric(table)  
  
# Create Confusion Matrix  
# Account for Predicting 1 Class  
if(length(table) == 2) {  
 print(table[1] / (table[1] + table[2]))  
} else {  
 print((table[1] + table[4]) / (table[1] + table[2] + table[3] + table[4]))  
}

## 

## Ridge Regression

grid = 10^seq (10, -2, length=100) #length implies the length, number of values in the sequence. Here we have 100 labmdas.  
  
# Train ridge regression model with all lambda values  
ridge.mod = glmnet(Covid.train.x.matrix, Covid.train.y, alpha=0, lambda=grid, family='binomial')  
  
  
# Apply cross validation  
cv.out = cv.glmnet(Covid.train.x.matrix, Covid.train.y, alpha=0, family='binomial')  
plot(cv.out)



# Get best lambda value  
bestlam.ridge = cv.out$lambda.min  
bestlam.ridge

# Train model on best lambda only  
ridge.probs = predict(ridge.mod, s=bestlam.ridge, newx=Covid.test.x.matrix)

# List coefficient estimates  
out = glmnet(Covid.train.x, Covid.train.y, alpha=0, family='binomial')  
predict(out, type="coefficients", s=bestlam.ridge)

# Create cutoff and accuracy lists  
Cutoff = seq(from=0.005, to=4, by=0.025)  
Accuracy = c()  
  
# Loop through cutoff values  
for(i in Cutoff) {  
 # Predict with cutoff  
 ridge.pred = rep(0, nrow(Covid.test.x.matrix))  
 ridge.pred[ridge.probs > i] = 1  
   
 # Create Confusion Matrix  
 table = as.numeric(table(ridge.pred, Covid.test.y))  
   
 # Account for Cutoff Predicting 1 Class  
 if(length(table) == 2) {  
 Accuracy = append(Accuracy, table[1] / (table[1] + table[2]))  
 print(table[1] / (table[1] + table[2]))  
 } else {  
 Accuracy = append(Accuracy, (table[1] + table[4]) / (table[1] + table[2] + table[3] + table[4]))  
 print((table[1] + table[4]) / (table[1] + table[2] + table[3] + table[4]))  
 }  
}

data <- data.frame(Cutoff, Accuracy)  
  
# Plot  
ggplot(data, aes(x=Cutoff, y=Accuracy)) +  
 geom\_line()

best.index = which.max(Accuracy)  
  
# Print Best Performing Cutoff  
Cutoff[best.index]

# Print Best Performing Accuracy  
Accuracy[best.index]

## Lasso Regression

grid = 10^seq (10, -2, length=100) #length implies the length, number of values in the sequence. Here we have 100 labmdas.  
  
# Train lasso regression model with all lambda values  
lasso.mod = glmnet(Covid.train.x.matrix, Covid.train.y, alpha=1, lambda=grid, family='binomial')  
  
  
# Apply cross validation  
cv.out = cv.glmnet(Covid.train.x.matrix, Covid.train.y, alpha=1, family='binomial')  
plot(cv.out)

# Get best lambda value  
bestlam.lasso = cv.out$lambda.min  
bestlam.lasso

# Train model on best lambda only  
lasso.probs = predict(lasso.mod, s=bestlam.lasso, newx=Covid.test.x.matrix)

# List coefficient estimates  
out = glmnet(Covid.train.x, Covid.train.y, alpha =0, family='binomial')  
predict(out, type="coefficients", s=bestlam.ridge)

# Create cutoff and accuracy lists  
Cutoff = seq(from=0.005, to=4, by=0.025)  
Accuracy = c()  
  
# Loop through cutoff values  
for(i in Cutoff) {  
 # Predict with cutoff  
 lasso.pred = rep(0, nrow(Covid.test.x.matrix))  
 lasso.pred[lasso.probs > i] = 1  
   
 # Create Confusion Matrix  
 table = as.numeric(table(lasso.pred, Covid.test.y))  
   
 # Account for Cutoff Predicting 1 Class  
 if(length(table) == 2) {  
 Accuracy = append(Accuracy, table[1] / (table[1] + table[2]))  
 print(table[1] / (table[1] + table[2]))  
 } else {  
 Accuracy = append(Accuracy, (table[1] + table[4]) / (table[1] + table[2] + table[3] + table[4]))  
 print((table[1] + table[4]) / (table[1] + table[2] + table[3] + table[4]))  
 }  
}

data <- data.frame(Cutoff, Accuracy)  
  
# Plot  
ggplot(data, aes(x=Cutoff, y=Accuracy)) +  
 geom\_line()

best.index = which.max(Accuracy)  
  
# Print Best Performing Cutoff  
Cutoff[best.index]

# Print Best Performing Accuracy  
Accuracy[best.index]