ML\_Assignment 6

2023-02-22

### Load Packages Needed for the Assignment

library(tidyverse)

## ── Attaching packages ─────────────────────────────────────── tidyverse 1.3.2 ──  
## ✔ ggplot2 3.4.1 ✔ purrr 1.0.1  
## ✔ tibble 3.1.8 ✔ dplyr 1.1.0  
## ✔ tidyr 1.3.0 ✔ stringr 1.5.0  
## ✔ readr 2.1.4 ✔ forcats 1.0.0  
## ── Conflicts ────────────────────────────────────────── tidyverse\_conflicts() ──  
## ✖ dplyr::filter() masks stats::filter()  
## ✖ dplyr::lag() masks stats::lag()

library(janitor)

##   
## Attaching package: 'janitor'  
##   
## The following objects are masked from 'package:stats':  
##   
## chisq.test, fisher.test

library(yardstick)

## For binary classification, the first factor level is assumed to be the event.  
## Use the argument `event\_level = "second"` to alter this as needed.  
##   
## Attaching package: 'yardstick'  
##   
## The following object is masked from 'package:readr':  
##   
## spec

library(stats)  
library(factoextra)

## Welcome! Want to learn more? See two factoextra-related books at https://goo.gl/ve3WBa

library(cluster)  
library(caret)

## Loading required package: lattice  
##   
## Attaching package: 'caret'  
##   
## The following objects are masked from 'package:yardstick':  
##   
## precision, recall, sensitivity, specificity  
##   
## The following object is masked from 'package:purrr':  
##   
## lift

library(NHANES)  
library(e1071)  
library(rpart)  
library(rpart.plot)  
library(pROC)

## Type 'citation("pROC")' for a citation.  
##   
## Attaching package: 'pROC'  
##   
## The following objects are masked from 'package:stats':  
##   
## cov, smooth, var

library(kernlab)

##   
## Attaching package: 'kernlab'  
##   
## The following object is masked from 'package:purrr':  
##   
## cross  
##   
## The following object is masked from 'package:ggplot2':  
##   
## alpha

## Assignment Instructions:

For this assignment, you will:

1. Restrict the NHANES data to the list of 11 variables below. Partition the data into training and testing using a 70/30 split.
2. Construct three prediction models to predict diabetes using the 11 features from NHANES. You will use the following three algorithms to create your prediction models:
3. Classification Tree
4. Support Vector Classifier (i.e. Support Vector Machine with a linear classifier)
5. Logistic regression.
6. You will optimize each model using cross-validation to choose hyperparameters in the training data and then compare performance across models.
7. Select a “optimal” model and calculate final evaluation metrics in the test set.

### Step 1: Load and Prepare Dataset

data(NHANES)  
  
a6.data <- NHANES %>%  
 select("Age", "Race1", "Education", "HHIncome", "Weight", "Height", "Pulse", "Diabetes", "BMI", "PhysActive", "Smoke100")  
  
#Remove Missing Variables  
a6.data <- na.omit(a6.data)  
  
#Renove Duplicate Variables  
a6.data <- distinct(a6.data)  
  
#Centering and Scaling: Although not needed for classification trees, this would be important for SVM and logistic regression.  
#set.up.preprocess<-preProcess(a6.data, method=c("center", "scale"))  
  
#Output pre-processed values  
#transformed.vals<-predict(set.up.preprocess, a6.data)

### Step 2: Data Partitioning

set.seed(123)  
  
train.index<-createDataPartition(a6.data$Diabetes, p=0.7, list=FALSE)  
  
a6.data.train<-a6.data[train.index,]  
a6.data.test<-a6.data[-train.index,]

### Step 3: Training the Model using different methods:

#### 1. Classification Tree

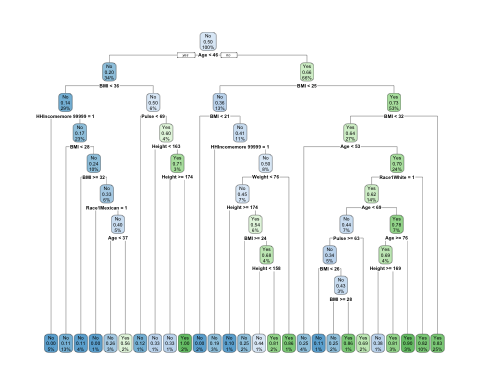
set.seed(123)  
  
#Creating 10-fold cross-validation and using down-sampling because of imbalance in data  
train.control.class<-trainControl(method="cv", number=10, sampling="down", classProbs = T)  
  
#Create sequence of cp parameters to try   
grid.2<-expand.grid(cp=seq(0.001, 0.3, by=0.01))  
  
#Train model  
tree.diabetes <-train(Diabetes ~., data=a6.data.train, method="rpart",trControl=train.control.class, tuneGrid=grid.2)  
tree.diabetes$bestTune

## cp  
## 1 0.001

tree.diabetes

## CART   
##   
## 2829 samples  
## 10 predictor  
## 2 classes: 'No', 'Yes'   
##   
## No pre-processing  
## Resampling: Cross-Validated (10 fold)   
## Summary of sample sizes: 2546, 2546, 2546, 2546, 2547, 2546, ...   
## Addtional sampling using down-sampling  
##   
## Resampling results across tuning parameters:  
##   
## cp Accuracy Kappa   
## 0.001 0.6726446 0.1848086  
## 0.011 0.6709316 0.2144341  
## 0.021 0.6702248 0.2114788  
## 0.031 0.6518352 0.1989322  
## 0.041 0.6518352 0.1989322  
## 0.051 0.6472178 0.1911850  
## 0.061 0.6363023 0.1843559  
## 0.071 0.6224725 0.1803389  
## 0.081 0.6224725 0.1803389  
## 0.091 0.6224725 0.1803389  
## 0.101 0.6224725 0.1803389  
## 0.111 0.6224725 0.1803389  
## 0.121 0.6224725 0.1803389  
## 0.131 0.6224725 0.1803389  
## 0.141 0.6224725 0.1803389  
## 0.151 0.6224725 0.1803389  
## 0.161 0.6224725 0.1803389  
## 0.171 0.6224725 0.1803389  
## 0.181 0.6224725 0.1803389  
## 0.191 0.6224725 0.1803389  
## 0.201 0.6224725 0.1803389  
## 0.211 0.6224725 0.1803389  
## 0.221 0.6224725 0.1803389  
## 0.231 0.6224725 0.1803389  
## 0.241 0.6224725 0.1803389  
## 0.251 0.6224725 0.1803389  
## 0.261 0.6224725 0.1803389  
## 0.271 0.6224725 0.1803389  
## 0.281 0.6224725 0.1803389  
## 0.291 0.6224725 0.1803389  
##   
## Accuracy was used to select the optimal model using the largest value.  
## The final value used for the model was cp = 0.001.

rpart.plot(tree.diabetes$finalModel)



#Note you can obtain variable importance on the final model within training data  
varImp(tree.diabetes)

## rpart variable importance  
##   
## only 20 most important variables shown (out of 35)  
##   
## Overall  
## Age 100.000  
## BMI 89.936  
## Weight 53.262  
## PhysActiveYes 29.921  
## Height 26.160  
## Pulse 24.373  
## EducationCollege Grad 23.467  
## HHIncomemore 99999 17.664  
## Race1White 13.174  
## HHIncome75000-99999 6.656  
## Education9 - 11th Grade 5.380  
## EducationHigh School 4.971  
## HHIncome45000-54999 3.210  
## HHIncome 5000-9999 2.975  
## Smoke100Yes 2.426  
## Race1Mexican 2.062  
## EducationSome College 1.905  
## HHIncome25000-34999 1.145  
## `EducationSome College` 0.000  
## Race1Hispanic 0.000

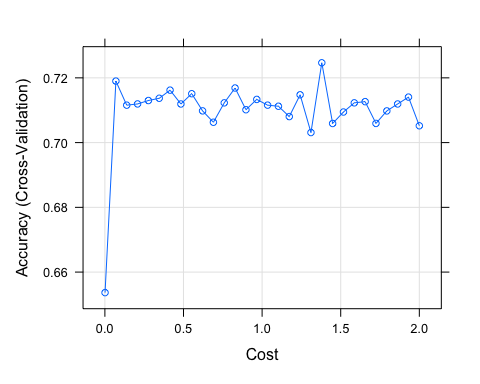
#Note you can get accuracy metric and confusion matrix from training.  
cm.tree <- confusionMatrix(tree.diabetes)  
cm.tree

## Cross-Validated (10 fold) Confusion Matrix   
##   
## (entries are percentual average cell counts across resamples)  
##   
## Reference  
## Prediction No Yes  
## No 59.2 3.8  
## Yes 29.0 8.1  
##   
## Accuracy (average) : 0.6727

After cross-validating hyperparameters and evaluating on the training set, the average accuracy of the classification tree model is **0.6727**.

#### 2. Support Vector Classifier (i.e. Support Vector Machine with a linear classifier)

set.seed(123)  
  
#Creating 10-fold cross-validation and using down-sampling because of imbalance in data  
train.control.class<-trainControl(method="cv", number=10, sampling="down", classProbs = T)  
  
#Incorporate different values for cost (C)  
svm.diabetes <- train(Diabetes ~ ., data = a6.data.train, method="svmLinear",   
 trControl=train.control.class,  
 preProcess=c("center", "scale"),  
 tuneGrid=expand.grid(C=seq(0.001,2, length=30)))  
  
#Visualize accuracy versus values of C  
plot(svm.diabetes)



#See information about final model  
svm.diabetes$finalModel

## Support Vector Machine object of class "ksvm"   
##   
## SV type: C-svc (classification)   
## parameter : cost C = 1.37962068965517   
##   
## Linear (vanilla) kernel function.   
##   
## Number of Support Vectors : 398   
##   
## Objective Function Value : -529.7969   
## Training error : 0.240299   
## Probability model included.

#Obtain metrics of accuracy from training  
confusionMatrix(svm.diabetes)

## Cross-Validated (10 fold) Confusion Matrix   
##   
## (entries are percentual average cell counts across resamples)  
##   
## Reference  
## Prediction No Yes  
## No 63.3 2.7  
## Yes 24.8 9.2  
##   
## Accuracy (average) : 0.7246

After cross-validating hyperparameters and evaluating on the training set, the average accuracy of the SVM model is **0.7246**.

#### 3. Logistic Regression Model

set.seed(123)  
  
#Setting a cross validation control  
cv\_results <- trainControl(method = "cv", number = 10, sampling = "down", classProbs = T)  
  
#Training a logistic regression model with cross validation  
log.diabetes <- train(Diabetes ~ ., data = a6.data.train,  
 method = "glm",  
 family = "binomial",  
 trControl = cv\_results,  
 preProc=c("center", "scale"))  
  
  
log.diabetes$results

## parameter Accuracy Kappa AccuracySD KappaSD  
## 1 none 0.7186289 0.2586378 0.03169238 0.04918597

confusionMatrix(log.diabetes)

## Cross-Validated (10 fold) Confusion Matrix   
##   
## (entries are percentual average cell counts across resamples)  
##   
## Reference  
## Prediction No Yes  
## No 63.0 2.9  
## Yes 25.2 8.9  
##   
## Accuracy (average) : 0.7186

After cross-validating hyperparameters and evaluating on the training set, the average accuracy of the logistic regression model is **0.7186**.

### Step 4: Final Model Evaluation

After comparing the accuracies of all 3 models, the SVM model has the highest accuracy. I would go ahead with this model for the final evaluation.

set.seed(123)  
  
svm.prediction.final <- predict(svm.diabetes,newdata = a6.data.test)  
svm.confusion.matrix <- confusionMatrix(svm.prediction.final, a6.data.test$Diabetes, positive = "Yes")  
  
postResample(svm.prediction.final,a6.data.test$Diabetes)

## Accuracy Kappa   
## 0.7010735 0.2479912

svm.confusion.matrix

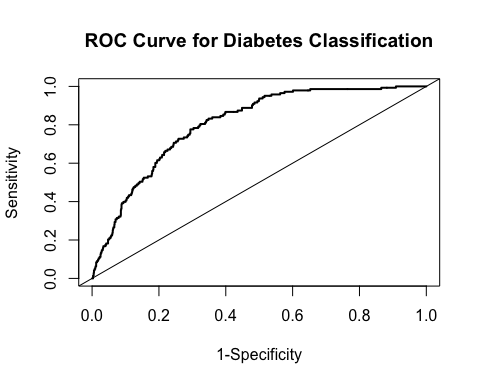
## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction No Yes  
## No 737 31  
## Yes 331 112  
##   
## Accuracy : 0.7011   
## 95% CI : (0.6744, 0.7268)  
## No Information Rate : 0.8819   
## P-Value [Acc > NIR] : 1   
##   
## Kappa : 0.248   
##   
## Mcnemar's Test P-Value : <2e-16   
##   
## Sensitivity : 0.78322   
## Specificity : 0.69007   
## Pos Pred Value : 0.25282   
## Neg Pred Value : 0.95964   
## Prevalence : 0.11808   
## Detection Rate : 0.09249   
## Detection Prevalence : 0.36581   
## Balanced Accuracy : 0.73665   
##   
## 'Positive' Class : Yes   
##

#Create ROC Curve for Analysis  
pred.prob<-predict(svm.diabetes, a6.data.test, type="prob")  
  
#Another potential evaluation: Area under the Reciver Operating Curve (AUROC)  
svm.analysis <- roc(response=a6.data.test$Diabetes, predictor=pred.prob[,2])

## Setting levels: control = No, case = Yes

## Setting direction: controls < cases

plot(1-svm.analysis$specificities,svm.analysis$sensitivities,type="l",  
ylab="Sensitivity",xlab="1-Specificity",col="black",lwd=2,  
main = "ROC Curve for Diabetes Classification")  
abline(a=0,b=1)



#Area Under the Curve  
svm.analysis$auc

## Area under the curve: 0.8038

**Final Evaluation Metrics for the SVM Model**:

* **Accuracy** : 0.7011 (95% CI: 0.6744 - 0.7268)
* **Sensitivity** : 0.78322
* **Specificity** : 0.69007
* **ROC Area Under the Curve**: 0.8038

**5. List and describe at least two limitations/considerations of the model generated by this analysis. Limitations can be analytical or they can be considerations that need to be made regarding how the model would be applied in practice.**

* 1. **Causal Model vs Predictive Model**: The SVM model created through my analysis is effective in predicting the Diabetes among patients based on different features from a publically available dataset, it may be inappropriate to make causal inference about the risk factors that cause diabetes in different patients. To make causal inference, we need methodology based on theory with a preexisting causal hypothesis and selected features to be used in the model, with other epidemiology study considerations.
  2. **Generalizability of the SVM Model**: In this analysis, the SVM model was trained on NHANES data. The predictions and performance metrics of this model may not apply to populations from different countries or populations having different disease prevalence, or differing proportions of demographic, environment, or social factors. Moreover, NHANES data is collected through self-reported surveys, which may also introduce bias within the data set that may not reflect the true data.