MLHW6\_JF

February 23, 2022

## Q1: Restrict the NHANES data to the list of 11 variables below. Partition the data into training and testing using a 70/30 split. REMINDER: Look at the frequency of your outcome variable to check for balance

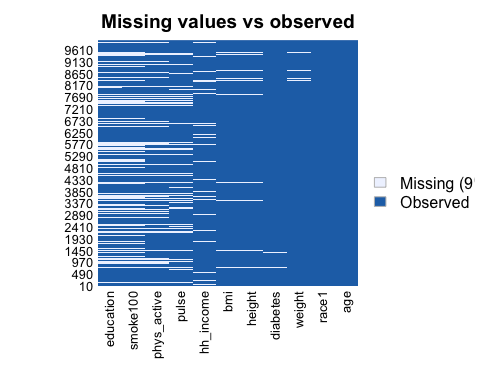
data(NHANES)  
  
NH = NHANES %>%  
 janitor::clean\_names() %>%  
 select(age, race1, education, hh\_income, weight, height, pulse, diabetes, bmi, phys\_active, smoke100)  
  
NH$diabetes <- factor(NH$diabetes, levels = c("No", "Yes"))  
  
#Check distributions, missing data etc, omitting the NAs  
summary(NH)

## age race1 education hh\_income   
## Min. : 0.00 Black :1197 8th Grade : 451 more 99999 :2220   
## 1st Qu.:17.00 Hispanic: 610 9 - 11th Grade: 888 75000-99999:1084   
## Median :36.00 Mexican :1015 High School :1517 25000-34999: 958   
## Mean :36.74 White :6372 Some College :2267 35000-44999: 863   
## 3rd Qu.:54.00 Other : 806 College Grad :2098 45000-54999: 784   
## Max. :80.00 NA's :2779 (Other) :3280   
## NA's : 811   
## weight height pulse diabetes bmi   
## Min. : 2.80 Min. : 83.6 Min. : 40.00 No :9098 Min. :12.88   
## 1st Qu.: 56.10 1st Qu.:156.8 1st Qu.: 64.00 Yes : 760 1st Qu.:21.58   
## Median : 72.70 Median :166.0 Median : 72.00 NA's: 142 Median :25.98   
## Mean : 70.98 Mean :161.9 Mean : 73.56 Mean :26.66   
## 3rd Qu.: 88.90 3rd Qu.:174.5 3rd Qu.: 82.00 3rd Qu.:30.89   
## Max. :230.70 Max. :200.4 Max. :136.00 Max. :81.25   
## NA's :78 NA's :353 NA's :1437 NA's :366   
## phys\_active smoke100   
## No :3677 No :4024   
## Yes :4649 Yes :3211   
## NA's:1674 NA's:2765   
##   
##   
##   
##

missmap(NH, main = "Missing values vs observed")

## Warning: Unknown or uninitialised column: `arguments`.  
  
## Warning: Unknown or uninitialised column: `arguments`.

## Warning: Unknown or uninitialised column: `imputations`.



NH <- na.omit(NH)  
summary(NH$diabetes) #Notice that the data is unbalanced so we will downsize

## No Yes   
## 5697 659

#tidyverse way to create data partition  
train\_indices <- createDataPartition(y = NH$diabetes,p = 0.7,list = FALSE)  
train\_data <- NH[train\_indices, ]  
test\_data <- NH[-train\_indices, ]

## Q2: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: Q3: You will optimize each model using cross-validation to choose hyperparameters in the training data and then compare performance across models.

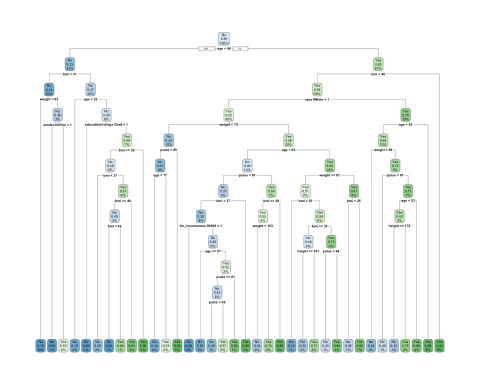
#The diabetes data is unbalanced so we will downsize.  
  
set.seed(100)  
#Creating 10-fold cross-validation and using down-sampling because of imbalance in data  
train\_control\_ct <- trainControl(method = "cv", number = 10, sampling = "down")  
  
#Create sequence of cp parameters to try   
grid\_ct <- expand.grid(cp = seq(0.001, 0.3, by = 0.01))  
  
#Train model  
ct\_diabetes <- train(diabetes~., data = train\_data, method = "rpart",trControl = train\_control\_ct, tuneGrid = grid\_ct)  
  
ct\_diabetes$bestTune #cp:0.001

## cp  
## 1 0.001

ct\_diabetes

## CART   
##   
## 4450 samples  
## 10 predictor  
## 2 classes: 'No', 'Yes'   
##   
## No pre-processing  
## Resampling: Cross-Validated (10 fold)   
## Summary of sample sizes: 4004, 4005, 4005, 4005, 4006, 4006, ...   
## Addtional sampling using down-sampling  
##   
## Resampling results across tuning parameters:  
##   
## cp Accuracy Kappa   
## 0.001 0.7074116 0.2025368  
## 0.011 0.7049281 0.2187718  
## 0.021 0.6822521 0.2059659  
## 0.031 0.6415784 0.1782574  
## 0.041 0.6148170 0.1631286  
## 0.051 0.5845036 0.1459509  
## 0.061 0.5739180 0.1405461  
## 0.071 0.5521203 0.1341666  
## 0.081 0.5521203 0.1341666  
## 0.091 0.5521203 0.1341666  
## 0.101 0.5521203 0.1341666  
## 0.111 0.5521203 0.1341666  
## 0.121 0.5521203 0.1341666  
## 0.131 0.5521203 0.1341666  
## 0.141 0.5521203 0.1341666  
## 0.151 0.5521203 0.1341666  
## 0.161 0.5521203 0.1341666  
## 0.171 0.5521203 0.1341666  
## 0.181 0.5521203 0.1341666  
## 0.191 0.5521203 0.1341666  
## 0.201 0.5521203 0.1341666  
## 0.211 0.5521203 0.1341666  
## 0.221 0.5521203 0.1341666  
## 0.231 0.5521203 0.1341666  
## 0.241 0.5521203 0.1341666  
## 0.251 0.5521203 0.1341666  
## 0.261 0.5521203 0.1341666  
## 0.271 0.5521203 0.1341666  
## 0.281 0.5521203 0.1341666  
## 0.291 0.5521203 0.1341666  
##   
## 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(ct\_diabetes$finalModel)



#variable importance on the final model within training data  
varImp(ct\_diabetes)

## rpart variable importance  
##   
## only 20 most important variables shown (out of 37)  
##   
## Overall  
## bmi 100.0000  
## age 99.3008  
## weight 86.4769  
## pulse 43.1240  
## height 36.6019  
## race1White 19.5681  
## phys\_activeYes 15.8161  
## hh\_incomemore 99999 14.1336  
## hh\_income75000-99999 13.5237  
## educationCollege Grad 9.4181  
## hh\_income35000-44999 6.4980  
## smoke100Yes 5.1064  
## hh\_income10000-14999 3.9180  
## hh\_income55000-64999 3.1284  
## race1Hispanic 2.3011  
## hh\_income65000-74999 1.9059  
## educationSome College 1.4185  
## educationHigh School 1.0596  
## hh\_income45000-54999 0.8546  
## education9 - 11th Grade 0.7243

# top three most important variables were age(100), bmi(81.99), weight(65.090) and the least important variable was income.   
  
#accuracy metric and confusion matrix from training.  
confusionMatrix(ct\_diabetes) #Accuracy(average): 0.7171

## Cross-Validated (10 fold) Confusion Matrix   
##   
## (entries are percentual average cell counts across resamples)  
##   
## Reference  
## Prediction No Yes  
## No 63.6 3.3  
## Yes 26.0 7.1  
##   
## Accuracy (average) : 0.7074

#if this the measure you choose then I can predictions on the test data as probabilities and/or produce a ROC curve.

The accuracy of the classification tree model is 71%.

modelLookup("svmLinear")

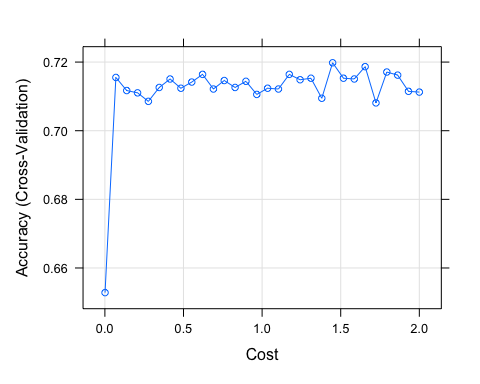
## model parameter label forReg forClass probModel  
## 1 svmLinear C Cost TRUE TRUE TRUE

set.seed(100)  
  
#Set 10-fold cross-validation. Note if you want predicted probabilities, you need to set class Probs=True  
train\_controlSVC <- trainControl(method = "cv", number = 10, sampling = "down", classProbs = T)  
  
svm <- train(diabetes ~ ., data = train\_data, method = "svmLinear", trControl = train\_controlSVC, preProcess = c("center", "scale"))  
  
svm #accuracy: 0.7236

## Support Vector Machines with Linear Kernel   
##   
## 4450 samples  
## 10 predictor  
## 2 classes: 'No', 'Yes'   
##   
## Pre-processing: centered (26), scaled (26)   
## Resampling: Cross-Validated (10 fold)   
## Summary of sample sizes: 4004, 4005, 4005, 4005, 4006, 4006, ...   
## Addtional sampling using down-sampling prior to pre-processing  
##   
## Resampling results:  
##   
## Accuracy Kappa   
## 0.7148294 0.2364933  
##   
## Tuning parameter 'C' was held constant at a value of 1

#Incorporate different values for cost parameter(cp) bc this method won't tune the hyperparameters for us. The CP is how much misclassification the support vector will allow  
svm\_caret <- train(diabetes ~ ., data = train\_data, method = "svmLinear", trControl = train\_controlSVC, preProcess = c("center", "scale"), tuneGrid = expand.grid(C = seq(0.001,2, length = 30)))

#Visualize accuracy versus values of C. This shows how the accuracy changes based on the level of the cost I chose.   
plot(svm\_caret)



#Obtain metrics of accuracy from training  
confusionMatrix(svm\_caret) #Accuracy (average) : 0.7364

## Cross-Validated (10 fold) Confusion Matrix   
##   
## (entries are percentual average cell counts across resamples)  
##   
## Reference  
## Prediction No Yes  
## No 64.1 2.5  
## Yes 25.6 7.9  
##   
## Accuracy (average) : 0.7198

The accuracy of the SVC model is 73.64%.

set.seed(100)  
logit <- train(  
 diabetes ~ ., data = train\_data, method = "glm", family = "binomial", trControl = trainControl(method = "cv", number = 10, sampling = "down"), preProc = c("center", "scale"))  
  
logit

## Generalized Linear Model   
##   
## 4450 samples  
## 10 predictor  
## 2 classes: 'No', 'Yes'   
##   
## Pre-processing: centered (26), scaled (26)   
## Resampling: Cross-Validated (10 fold)   
## Summary of sample sizes: 4004, 4005, 4005, 4005, 4006, 4006, ...   
## Addtional sampling using down-sampling prior to pre-processing  
##   
## Resampling results:  
##   
## Accuracy Kappa   
## 0.7188789 0.2410717

logit$results

## parameter Accuracy Kappa AccuracySD KappaSD  
## 1 none 0.7188789 0.2410717 0.02332836 0.04081973

confusionMatrix(logit) #Accuracy (average) : 0.7348

## Cross-Validated (10 fold) Confusion Matrix   
##   
## (entries are percentual average cell counts across resamples)  
##   
## Reference  
## Prediction No Yes  
## No 64.0 2.4  
## Yes 25.7 7.9  
##   
## Accuracy (average) : 0.7189

The accuracy of the logistic regression model is 73.48%.

## Q4: Select a “optimal” model and calculate final evaluation metrics in the test set.

The optimal model I selected was the support vector classifier model because the accuracy was the highest out of the three models I’ve constructed.

#Checking out info about final model  
svm\_caret$finalModel

## Support Vector Machine object of class "ksvm"   
##   
## SV type: C-svc (classification)   
## parameter : cost C = 1.44855172413793   
##   
## Linear (vanilla) kernel function.   
##   
## Number of Support Vectors : 549   
##   
## Objective Function Value : -772.1589   
## Training error : 0.237013   
## Probability model included.

#Make predictions in testset  
svm\_pred\_test <- predict(svm\_caret, test\_data)  
  
#Get evaluation metrics from test set  
confusionMatrix(svm\_pred\_test, test\_data$diabetes, positive = "Yes") #Accuracy 0.7219 #Sensitivity:0.731 #Specificity:0.721

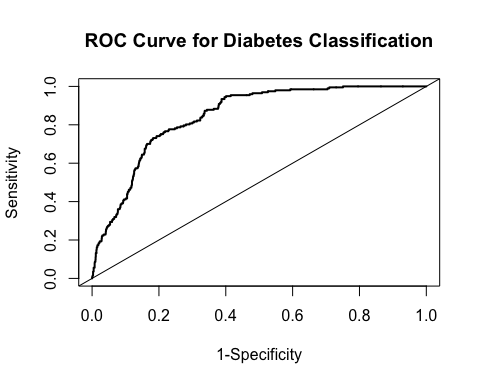
## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction No Yes  
## No 1167 35  
## Yes 542 162  
##   
## Accuracy : 0.6973   
## 95% CI : (0.6761, 0.7178)  
## No Information Rate : 0.8966   
## P-Value [Acc > NIR] : 1   
##   
## Kappa : 0.2362   
##   
## Mcnemar's Test P-Value : <2e-16   
##   
## Sensitivity : 0.82234   
## Specificity : 0.68286   
## Pos Pred Value : 0.23011   
## Neg Pred Value : 0.97088   
## Prevalence : 0.10336   
## Detection Rate : 0.08499   
## Detection Prevalence : 0.36936   
## Balanced Accuracy : 0.75260   
##   
## 'Positive' Class : Yes   
##

#Create ROC Curve for Analysis  
pred.prob <- predict(svm\_caret, test\_data, type = "prob")  
  
#Another potential evaluation: Area under the Receiver Operating Curve (AUROC)  
#The ROC curve shows the trade-off between sensitivity (or TPR) and specificity (1 – FPR). Classifiers that give curves closer to the top-left corner indicate a better performance. The closer the curve comes to the 45-degree diagonal of the ROC space, the less accurate the test.  
analysis <- roc(response = test\_data$diabetes, predictor = pred.prob[,2])

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

## Setting direction: controls < cases

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

 The accuracy of the SVC model on the test data was 72.19% with a sensitivity of 0.731 and a specificity of 0.721.

## Q5: 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.

Some advantages of support vector classification is that it is very effective with high dimensional data and it can be used for both regression and classification problem. Some disadvantages of this algorithm type is that took more time to train the data, which is most likely due to this being a data set. SVC is not a probabilistic model so we can not explain the diabetes classification in terms of probability, which could be an issue if we wanted to use this model to explain the predictions to those who do not have a machine learning background. Essentially we would have issues with interpretability and generalizability as well. Also, this may not be applicable to small populations as they may not behave the same way.