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

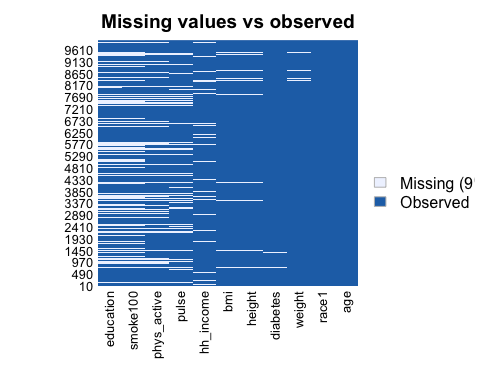
data(NHANES)  
  
NH = NHANES %>%  
 janitor::clean\_names() %>%  
 select(age, race1, education, hh\_income, weight, height, pulse, diabetes, bmi, phys\_active, smoke100)  
  
NH$diabetes <- factor(NH$diabetes, levels = c("No", "Yes"))  
  
#Check distributions, missing data etc, omitting the NAs  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

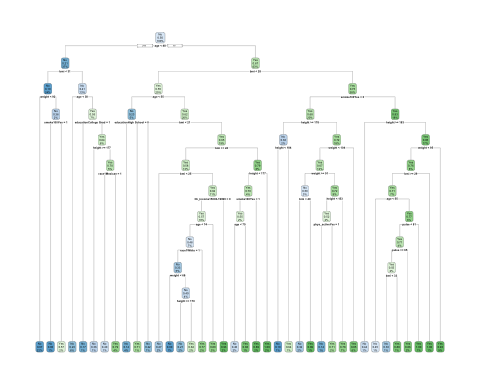
#The diabetes data is unbalanced so we will downsize.  
  
set.seed(100)  
#Creating 10-fold cross-validation and using down-sampling because of imbalance in data  
train\_control\_ct <- trainControl(method = "cv", number = 10, sampling = "down")  
  
#Create sequence of cp parameters to try   
grid\_ct <- 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 0.7242812 0.2350892  
## 0.011 0.7004443 0.2174678  
## 0.021 0.6667394 0.1998333  
## 0.031 0.6406796 0.1852829  
## 0.041 0.6143696 0.1633567  
## 0.051 0.6036073 0.1604889  
## 0.061 0.6036073 0.1604889  
## 0.071 0.6036073 0.1604889  
## 0.081 0.6036073 0.1604889  
## 0.091 0.6036073 0.1604889  
## 0.101 0.6036073 0.1604889  
## 0.111 0.6036073 0.1604889  
## 0.121 0.6036073 0.1604889  
## 0.131 0.6036073 0.1604889  
## 0.141 0.6036073 0.1604889  
## 0.151 0.6036073 0.1604889  
## 0.161 0.6036073 0.1604889  
## 0.171 0.6036073 0.1604889  
## 0.181 0.6036073 0.1604889  
## 0.191 0.6036073 0.1604889  
## 0.201 0.6036073 0.1604889  
## 0.211 0.6036073 0.1604889  
## 0.221 0.6036073 0.1604889  
## 0.231 0.6036073 0.1604889  
## 0.241 0.6036073 0.1604889  
## 0.251 0.6036073 0.1604889  
## 0.261 0.6036073 0.1604889  
## 0.271 0.6036073 0.1604889  
## 0.281 0.6036073 0.1604889  
## 0.291 0.6036073 0.1604889  
##   
## 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 36)  
##   
## Overall  
## age 100.0000  
## bmi 76.0288  
## weight 57.2588  
## height 28.6762  
## race1White 24.5571  
## hh\_incomemore 99999 21.3208  
## pulse 20.5735  
## phys\_activeYes 20.2194  
## smoke100Yes 8.3912  
## race1Other 6.2043  
## hh\_income10000-14999 5.1507  
## educationCollege Grad 4.1331  
## educationHigh School 3.8679  
## hh\_income45000-54999 3.8456  
## hh\_income15000-19999 3.1392  
## race1Mexican 2.3073  
## hh\_income75000-99999 1.4468  
## race1Hispanic 1.1820  
## educationSome College 1.0526  
## hh\_income35000-44999 0.5156

# 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 64.9 2.8  
## Yes 24.8 7.6  
##   
## Accuracy (average) : 0.7243

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

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, classProbs = T)  
  
svm <- train(diabetes ~ ., data = train\_data, method = "svmLinear", trControl = train\_controlSVC, preProcess = c("center", "scale"))

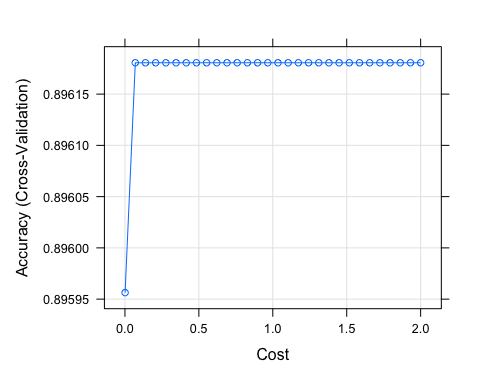
## maximum number of iterations reached 0.001091331 0.001081827maximum number of iterations reached 0.005337625 0.005046684maximum number of iterations reached 0.00965003 0.008812599maximum number of iterations reached 0.002363541 0.002321798maximum number of iterations reached 0.006878582 0.006529195maximum number of iterations reached 0.007040171 0.006467533maximum number of iterations reached 0.0028624 0.002795447maximum number of iterations reached 0.006751751 0.006353802maximum number of iterations reached 0.00917792 0.008484597maximum number of iterations reached 0.008485594 0.007816181maximum number of iterations reached 0.002210029 0.002165843

svm #accuracy: 0.8961

## 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, ...   
## Resampling results:  
##   
## Accuracy Kappa  
## 0.8961806 0   
##   
## 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) : 0.8962

## Cross-Validated (10 fold) Confusion Matrix   
##   
## (entries are percentual average cell counts across resamples)  
##   
## Reference  
## Prediction No Yes  
## No 89.6 10.4  
## Yes 0.0 0.0  
##   
## Accuracy (average) : 0.8962

logistic\_control <- trainControl(method = "cv", number = 10, classProbs = T)  
  
set.seed(100)  
logistic <- train(diabetes ~ ., data = train\_data, method = "glm", family = "binomial", trControl = logistic\_control)  
  
summary(logistic)

##   
## Call:  
## NULL  
##   
## Deviance Residuals:   
## Min 1Q Median 3Q Max   
## -1.9586 -0.4649 -0.2789 -0.1602 3.2896   
##   
## Coefficients:  
## Estimate Std. Error z value Pr(>|z|)   
## (Intercept) -15.734096 3.939161 -3.994 6.49e-05 \*\*\*  
## age 0.067202 0.004141 16.228 < 2e-16 \*\*\*  
## race1Hispanic -0.273837 0.280297 -0.977 0.32859   
## race1Mexican 0.020865 0.254630 0.082 0.93469   
## race1White -0.747368 0.163511 -4.571 4.86e-06 \*\*\*  
## race1Other 0.346826 0.240293 1.443 0.14892   
## `education9 - 11th Grade` -0.569756 0.231245 -2.464 0.01374 \*   
## `educationHigh School` -0.336450 0.212447 -1.584 0.11326   
## `educationSome College` -0.357111 0.211702 -1.687 0.09163 .   
## `educationCollege Grad` -0.452803 0.230725 -1.963 0.04970 \*   
## `hh\_income 5000-9999` 0.209207 0.497743 0.420 0.67426   
## `hh\_income10000-14999` 0.134197 0.454716 0.295 0.76790   
## `hh\_income15000-19999` 0.115102 0.456867 0.252 0.80109   
## `hh\_income20000-24999` 0.144930 0.450588 0.322 0.74772   
## `hh\_income25000-34999` -0.067981 0.441051 -0.154 0.87750   
## `hh\_income35000-44999` 0.112152 0.440979 0.254 0.79925   
## `hh\_income45000-54999` -0.404903 0.464955 -0.871 0.38384   
## `hh\_income55000-64999` 0.098261 0.455363 0.216 0.82915   
## `hh\_income65000-74999` -0.043944 0.467142 -0.094 0.92505   
## `hh\_income75000-99999` 0.020227 0.449215 0.045 0.96409   
## `hh\_incomemore 99999` -0.377429 0.442702 -0.853 0.39391   
## weight -0.024039 0.020497 -1.173 0.24088   
## height 0.042556 0.022768 1.869 0.06161 .   
## pulse 0.013086 0.004538 2.884 0.00393 \*\*   
## bmi 0.157884 0.058396 2.704 0.00686 \*\*   
## phys\_activeYes -0.189638 0.117756 -1.610 0.10730   
## smoke100Yes 0.222943 0.112868 1.975 0.04824 \*   
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## (Dispersion parameter for binomial family taken to be 1)  
##   
## Null deviance: 2967.2 on 4449 degrees of freedom  
## Residual deviance: 2368.4 on 4423 degrees of freedom  
## AIC: 2422.4  
##   
## Number of Fisher Scoring iterations: 6

confusionMatrix(logistic) #Accuracy (average) : 0.8926

## Cross-Validated (10 fold) Confusion Matrix   
##   
## (entries are percentual average cell counts across resamples)  
##   
## Reference  
## Prediction No Yes  
## No 88.3 9.5  
## Yes 1.3 0.9  
##   
## Accuracy (average) : 0.8926

## 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 = 0.0699310344827586   
##   
## Linear (vanilla) kernel function.   
##   
## Number of Support Vectors : 1040   
##   
## Objective Function Value : -64.6163   
## Training error : 0.10382   
## 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 = "No") #Accuracy 0.8966 #Sensitivity:1.00 #Specificity:0.00

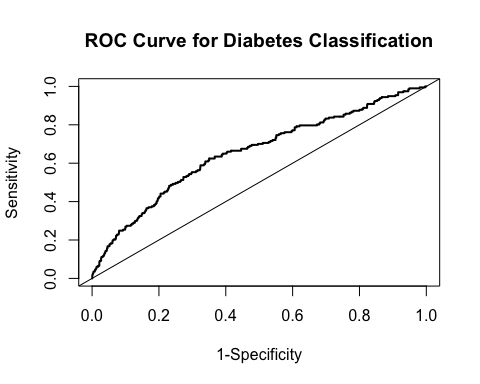
## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction No Yes  
## No 1709 197  
## Yes 0 0  
##   
## Accuracy : 0.8966   
## 95% CI : (0.8821, 0.91)  
## No Information Rate : 0.8966   
## P-Value [Acc > NIR] : 0.519   
##   
## Kappa : 0   
##   
## Mcnemar's Test P-Value : <2e-16   
##   
## Sensitivity : 1.0000   
## Specificity : 0.0000   
## Pos Pred Value : 0.8966   
## Neg Pred Value : NaN   
## Prevalence : 0.8966   
## Detection Rate : 0.8966   
## Detection Prevalence : 1.0000   
## Balanced Accuracy : 0.5000   
##   
## 'Positive' Class : No   
##

#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)



## 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.