Assignment 6

2023-02-27

library(tidyverse)  
library(rpart)  
library(caret)  
library(rpart.plot)  
library(pROC)  
library(NHANES)  
  
  
knitr::opts\_chunk$set(  
 echo = TRUE,  
 warning = FALSE,  
 fig.width = 8,   
 fig.height = 6,  
 out.width = "90%"  
)

## Restricting the NHANES data to the list of 11 variables below and Partitioning the data into training and testing using a 70/30 split.

## “Age”, “Race1”, “Education”, “HHIncome”, “Weight”, “Height”, “Pulse”, “Diabetes”, “BMI”, “PhysActive”, “Smoke100”

set.seed(123)  
hw6.df=NHANES %>%   
 janitor::clean\_names() %>%   
 select( age, race1, education, hh\_income, weight, height, pulse, diabetes, bmi, phys\_active, smoke100 )   
  
hw6 <-hw6.df[complete.cases(hw6.df),]  
  
summary(hw6$diabetes)

## No Yes   
## 5697 659

train.indices<-createDataPartition(y=hw6$diabetes,p=0.7,list=FALSE)  
train.data<-hw6[train.indices, ]  
test.data<-hw6[-train.indices, ]

## Constructing a classification tree model within the training data set

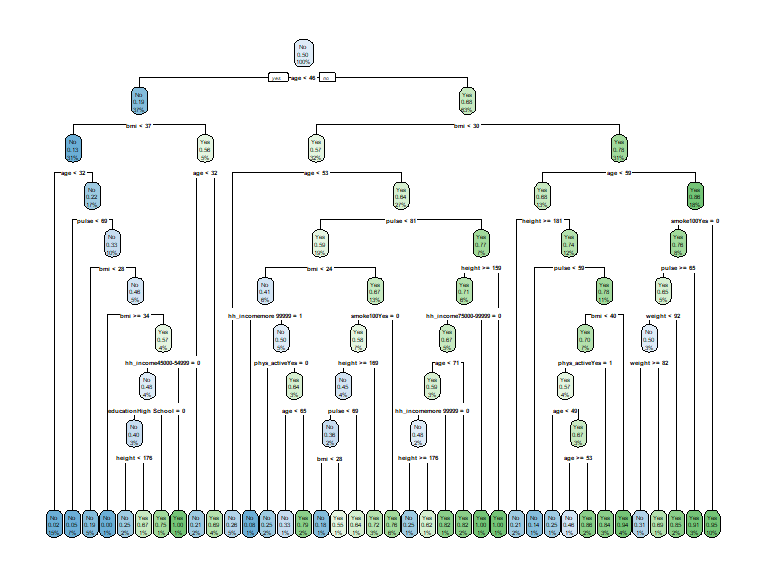
set.seed(123)  
  
### Creating 10-fold cross-validation and using down-sampling because of imbalance in data  
train.control<-trainControl(method="cv", number=10, sampling="down")  
  
### Create sequence of cp parameters to try   
grid.2<-expand.grid(cp=seq(0.001, 0.3, by=0.01))  
  
#### Train model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

## cp Accuracy Kappa AccuracySD KappaSD  
## 1 0.001 0.7044851 0.2225926 0.02841200 0.03349895  
## 2 0.011 0.6692093 0.2027177 0.04085624 0.03930518  
## 3 0.021 0.6426893 0.1774506 0.04719608 0.04582471  
## 4 0.031 0.6269443 0.1755479 0.05187012 0.04626204  
## 5 0.041 0.6090081 0.1640054 0.04768762 0.04897937  
## 6 0.051 0.5793452 0.1431098 0.03648829 0.02745343  
## 7 0.061 0.5793452 0.1431098 0.03648829 0.02745343  
## 8 0.071 0.5793452 0.1431098 0.03648829 0.02745343  
## 9 0.081 0.5793452 0.1431098 0.03648829 0.02745343  
## 10 0.091 0.5793452 0.1431098 0.03648829 0.02745343  
## 11 0.101 0.5793452 0.1431098 0.03648829 0.02745343  
## 12 0.111 0.5793452 0.1431098 0.03648829 0.02745343  
## 13 0.121 0.5793452 0.1431098 0.03648829 0.02745343  
## 14 0.131 0.5793452 0.1431098 0.03648829 0.02745343  
## 15 0.141 0.5793452 0.1431098 0.03648829 0.02745343  
## 16 0.151 0.5793452 0.1431098 0.03648829 0.02745343  
## 17 0.161 0.5793452 0.1431098 0.03648829 0.02745343  
## 18 0.171 0.5793452 0.1431098 0.03648829 0.02745343  
## 19 0.181 0.5793452 0.1431098 0.03648829 0.02745343  
## 20 0.191 0.5793452 0.1431098 0.03648829 0.02745343  
## 21 0.201 0.5793452 0.1431098 0.03648829 0.02745343  
## 22 0.211 0.5793452 0.1431098 0.03648829 0.02745343  
## 23 0.221 0.5793452 0.1431098 0.03648829 0.02745343  
## 24 0.231 0.5793452 0.1431098 0.03648829 0.02745343  
## 25 0.241 0.5793452 0.1431098 0.03648829 0.02745343  
## 26 0.251 0.5793452 0.1431098 0.03648829 0.02745343  
## 27 0.261 0.5793452 0.1431098 0.03648829 0.02745343  
## 28 0.271 0.5793452 0.1431098 0.03648829 0.02745343  
## 29 0.281 0.5793452 0.1431098 0.03648829 0.02745343  
## 30 0.291 0.5793452 0.1431098 0.03648829 0.02745343

tree.diabetes$bestTune

## cp  
## 1 0.001

### PLotting the classification tree  
rpart.plot(tree.diabetes$finalModel)



### Obtaining 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 100.0000  
## bmi 69.7194  
## weight 49.7904  
## height 29.0285  
## pulse 28.3578  
## educationCollege Grad 22.7297  
## phys\_activeYes 11.6649  
## hh\_incomemore 99999 11.2164  
## race1White 4.5234  
## smoke100Yes 4.4698  
## hh\_income45000-54999 4.0866  
## race1Mexican 3.7162  
## race1Hispanic 1.9055  
## educationSome College 1.8275  
## hh\_income55000-64999 1.1010  
## hh\_income65000-74999 1.0431  
## educationHigh School 0.8472  
## hh\_income75000-99999 0.7452  
## hh\_income20000-24999 0.5552  
## `hh\_income55000-64999` 0.0000

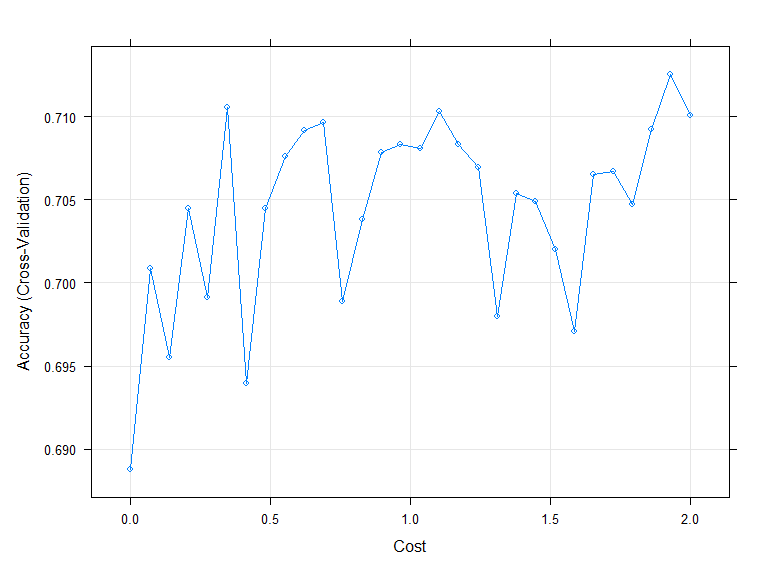
### Getting accuracy metric and confusion matrix from training.  
confusionMatrix(tree.diabetes)

## Cross-Validated (10 fold) Confusion Matrix   
##   
## (entries are percentual average cell counts across resamples)  
##   
## Reference  
## Prediction No Yes  
## No 62.6 2.6  
## Yes 27.0 7.8  
##   
## Accuracy (average) : 0.7045

* The Best Tuned (Final) model had a CP of 0.001 and an accuracy of 0.7045. Age was found to be the most important predictor of diabetes followed by bmi and weight.

## Constructing a SVC model using the training dataset

set.seed(123)  
  
### Setting a 10 fold CV,  
train\_control<-trainControl(method="cv", number=10)  
  
  
### Scaling the data and Incorporating different values for cost (C) to train the model  
  
svm.diabetes<-train(diabetes ~ ., data=train.data, method="svmLinear", trControl=train.control, preProcess=c("center", "scale"), tuneGrid=expand.grid(C=seq(0.001,2, length=30)))  
  
### Visualizing accuracy versus values of C  
plot(svm.diabetes)



### information about final model  
svm.diabetes$results

## C Accuracy Kappa AccuracySD KappaSD  
## 1 0.00100000 0.6887583 0.2070623 0.02558989 0.02698443  
## 2 0.06993103 0.7008957 0.2355595 0.03337312 0.04468051  
## 3 0.13886207 0.6954994 0.2266883 0.03123086 0.04431058  
## 4 0.20779310 0.7044837 0.2376260 0.02652449 0.03992456  
## 5 0.27672414 0.6990970 0.2283955 0.02458018 0.03718135  
## 6 0.34565517 0.7105506 0.2418594 0.03477403 0.04902698  
## 7 0.41458621 0.6939259 0.2225483 0.02544033 0.03236684  
## 8 0.48351724 0.7044968 0.2377812 0.03233465 0.05017294  
## 9 0.55244828 0.7076368 0.2373929 0.02664660 0.03807518  
## 10 0.62137931 0.7092104 0.2408223 0.02536140 0.03986034  
## 11 0.69031034 0.7096558 0.2457647 0.02616778 0.04077049  
## 12 0.75924138 0.6988667 0.2253068 0.02967305 0.04591711  
## 13 0.82817241 0.7038090 0.2357510 0.03312764 0.04433242  
## 14 0.89710345 0.7078635 0.2368134 0.02669722 0.04559831  
## 15 0.96603448 0.7083120 0.2371555 0.03047789 0.04163927  
## 16 1.03496552 0.7080807 0.2437129 0.02309712 0.03750777  
## 17 1.10389655 0.7103238 0.2382504 0.02969149 0.05097114  
## 18 1.17282759 0.7083105 0.2398567 0.02187385 0.03569114  
## 19 1.24175862 0.7069551 0.2338967 0.02616952 0.04393282  
## 20 1.31068966 0.6979703 0.2305565 0.02847907 0.04096510  
## 21 1.37962069 0.7053851 0.2371905 0.03253801 0.04564507  
## 22 1.44855172 0.7049311 0.2386977 0.03117527 0.04663823  
## 23 1.51748276 0.7020138 0.2337670 0.03239458 0.04487877  
## 24 1.58641379 0.6970674 0.2341420 0.02459489 0.03814218  
## 25 1.65534483 0.7065112 0.2318920 0.03037704 0.04460001  
## 26 1.72427586 0.7067349 0.2415578 0.03300357 0.04457183  
## 27 1.79320690 0.7047144 0.2331321 0.02882514 0.03916682  
## 28 1.86213793 0.7092194 0.2407636 0.02858939 0.04047193  
## 29 1.93106897 0.7125650 0.2467470 0.03252101 0.05436676  
## 30 2.00000000 0.7101122 0.2434769 0.03240846 0.04978576

svm.diabetes$finalModel

## Support Vector Machine object of class "ksvm"   
##   
## SV type: C-svc (classification)   
## parameter : cost C = 1.93106896551724   
##   
## Linear (vanilla) kernel function.   
##   
## Number of Support Vectors : 544   
##   
## Objective Function Value : -1011.328   
## Training error : 0.252165

### Obtaining 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 63.0 2.1  
## Yes 26.7 8.3  
##   
## Accuracy (average) : 0.7126

The final model has cost C = 1.93 and an accuracy of 0.7126 (#29). However it can also be observed that in #6 the accuracy is almost similar (0.7106) for a much lower cost C of 0.34.

## Constructing a logistic regression model

set.seed(123)  
  
  
### training model with logistic regression (regularized)  
  
logreg.diabetes <- train(  
 diabetes ~ .,   
 data = train.data,   
 method = "glm",   
 trControl = train.control,  
 family = "binomial",  
 preProcess=c("center", "scale")  
   
)  
  
  
### Printing results  
logreg.diabetes$results

## parameter Accuracy Kappa AccuracySD KappaSD  
## 1 none 0.7148238 0.2362721 0.03751177 0.05495224

confusionMatrix(logreg.diabetes)

## Cross-Validated (10 fold) Confusion Matrix   
##   
## (entries are percentual average cell counts across resamples)  
##   
## Reference  
## Prediction No Yes  
## No 63.6 2.5  
## Yes 26.0 7.9  
##   
## Accuracy (average) : 0.7148

The logistic regression model had anaccuracy of 0.7148 and a kappa of 0.23. Upon comparing the accuracies of all the 3 models, the logistic regression model had the highest accuracy and hence I will choose **logreg.diabetes**as my final model.

## Generating predictions in the test data

#Applying model in test set to generate predictions and construct evaluation metrics  
  
#First create predictions  
pred.log<-predict(logreg.diabetes, newdata=test.data)  
#Then use postResample to obtain evaluation metrics  
confusionMatrix(pred.log, test.data$diabetes, positive="Yes")

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction No Yes  
## No 1194 39  
## Yes 515 158  
##   
## Accuracy : 0.7093   
## 95% CI : (0.6884, 0.7297)  
## No Information Rate : 0.8966   
## P-Value [Acc > NIR] : 1   
##   
## Kappa : 0.242   
##   
## Mcnemar's Test P-Value : <2e-16   
##   
## Sensitivity : 0.8020   
## Specificity : 0.6987   
## Pos Pred Value : 0.2348   
## Neg Pred Value : 0.9684   
## Prevalence : 0.1034   
## Detection Rate : 0.0829   
## Detection Prevalence : 0.3531   
## Balanced Accuracy : 0.7503   
##   
## 'Positive' Class : Yes   
##

## Reporting final evaluation Metrics:

After applying the logistic regression model in the test dataset, the accuracy was 0.7093 and Sensitivity was 0.80

## Limitations:

1. Interpretability: Support Vector Classifiers and classification trees have higher interpretability as compared to logistic regression models and thus it has limited interpretability. Additionally we won’t be able to identuf the predictors that are more important to predict diabetes.
2. Since we used down-sampling for the training data and we did not balance the test data it may not accurately reflect the performance of the model in the real world.