EPI\_ML Homework 6

Arielle Coq

2/25/2020

#### Question 1- 1. Restrict the NHANES data to the list of 12 variables below. Partition the data into training and testing using a 70/30 split.

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

First imported the data into R and did some initial data cleaning kept the necessary variables and dropped the NAs in the dataset.

data(NHANES)  
  
nhanes <- data(NHANES)   
view(NHANES)  
  
  
nhanes =  
 NHANES %>%   
 select(Age, Gender, Race1, Education, HHIncome, Weight, Height, Pulse, Diabetes, BMI, PhysActive, Smoke100) %>%   
 mutate(Diabetes = recode(Diabetes, "Yes"= 1, "No" = 0)) %>%   
 drop\_na() %>%   
 janitor::clean\_names()  
  
nhanes$diabetes<-as.factor(nhanes$diabetes)

Next I will partition the data so that there is a 70/30 split and set the seed to 100.

set.seed(100)  
training.data<-nhanes$diabetes %>% createDataPartition(p=0.7, list=F)  
train.data<-nhanes[training.data, ]  
test.data<-nhanes[-training.data, ]  
  
  
#Store outcome   
diabetes.train<-train.data$diabetes  
diabetes.test<-test.data$diabetes  
  
# Store the outcome in one train and test and the predictors in another   
#model matrix- will create indicator variables for categorical varaibles, it does not do anything to the continuous variables  
x.train<-model.matrix(diabetes~., train.data)[,-1]  
x.test<-model.matrix(diabetes~., test.data)[,-1]

#### Question 2. 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:

1. Classification Tree
2. Support Vector Classifier (i.e. Support Vector Machine with a linear classifier)
3. Logistic regression.

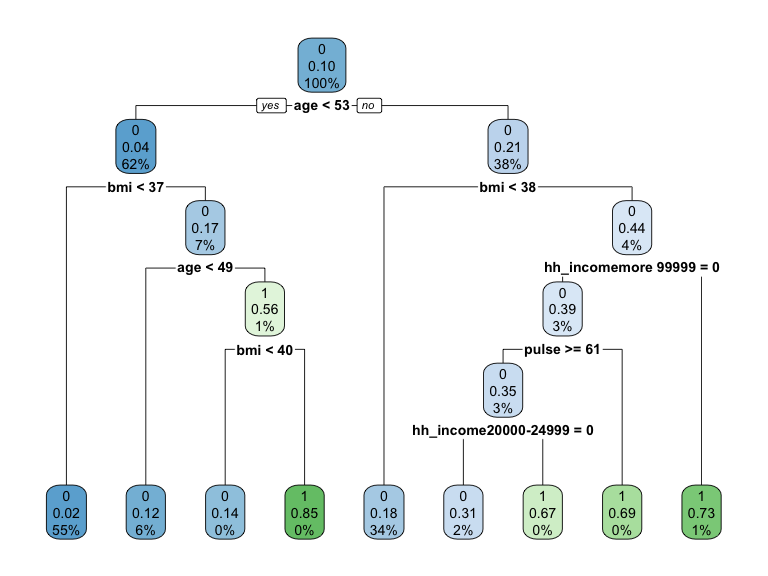
Going to set up a prediction model using a classification tree to predict diabetes using the 11 other features in the NHANES dataset.

#### Classification Tree prediction Question 2-4

train.control<-trainControl(method="cv", number=10)  
tree.diabetes<-train(diabetes~., data=train.data, method="rpart",trControl=train.control)  
tree.diabetes$bestTune

## cp  
## 2 0.008658009

rpart.plot(tree.diabetes$finalModel)



varImp(tree.diabetes)

## rpart variable importance  
##   
## only 20 most important variables shown (out of 35)  
##   
## Overall  
## age 100.000  
## bmi 85.280  
## weight 58.870  
## race1White 22.478  
## phys\_activeYes 22.263  
## pulse 11.946  
## hh\_incomemore 99999 11.235  
## height 8.811  
## race1Mexican 8.144  
## hh\_income20000-24999 6.610  
## gendermale 5.643  
## hh\_income 5000-9999 5.562  
## hh\_income10000-14999 2.706  
## educationHigh School 2.378  
## smoke100Yes 1.797  
## hh\_income65000-74999 1.700  
## hh\_income45000-54999 1.698  
## educationCollege Grad 1.152  
## race1Other 0.000  
## `hh\_income25000-34999` 0.000

pred.diabetes<-predict(tree.diabetes, test.data)  
pred.diabetes.prob<-predict(tree.diabetes, test.data, type="prob")  
  
eval.results<-confusionMatrix(pred.diabetes, test.data$diabetes, positive = "1")  
print(eval.results)

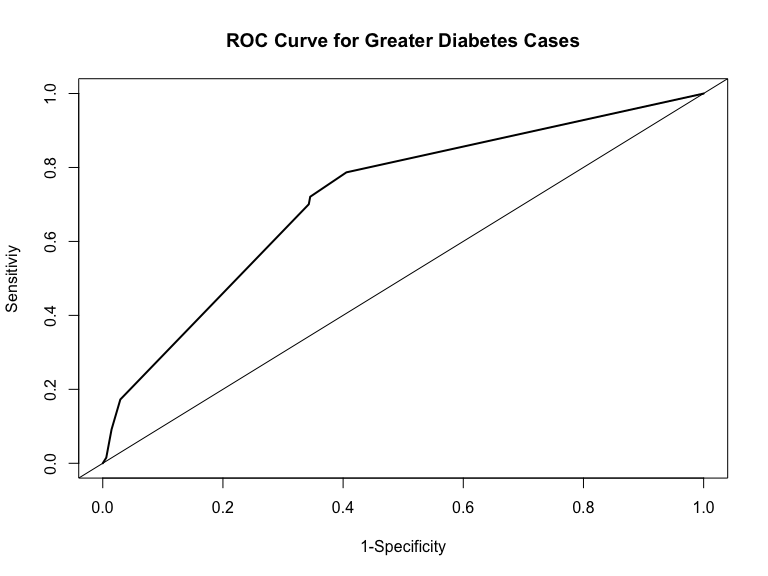
## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction 0 1  
## 0 1684 179  
## 1 25 18  
##   
## Accuracy : 0.893   
## 95% CI : (0.8782, 0.9065)  
## No Information Rate : 0.8966   
## P-Value [Acc > NIR] : 0.716   
##   
## Kappa : 0.1173   
##   
## Mcnemar's Test P-Value : <2e-16   
##   
## Sensitivity : 0.091371   
## Specificity : 0.985372   
## Pos Pred Value : 0.418605   
## Neg Pred Value : 0.903918   
## Prevalence : 0.103358   
## Detection Rate : 0.009444   
## Detection Prevalence : 0.022560   
## Balanced Accuracy : 0.538371   
##   
## 'Positive' Class : 1   
##

analysis <- roc(response=test.data$diabetes, predictor=pred.diabetes.prob[,2])

## Setting levels: control = 0, case = 1

## Setting direction: controls < cases

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



Going to set up a prediction model using a support vector classifer with a linear classifier to predict diabetes using the 11 other features in the NHANES dataset.

##### Support Vector Classifer with a linear classifier model Questions 2-4

svm.diabetes<-svm(diabetes ~ ., data=train.data, kernel="linear", cost=1, scale=TRUE)  
print(svm.diabetes)

##   
## Call:  
## svm(formula = diabetes ~ ., data = train.data, kernel = "linear",   
## cost = 1, scale = TRUE)  
##   
##   
## Parameters:  
## SVM-Type: C-classification   
## SVM-Kernel: linear   
## cost: 1   
##   
## Number of Support Vectors: 1162

### Cost- hyper parameter   
svm.pred<-predict(svm.diabetes, newdata=train.data)  
table(svm.pred, train.data$diabetes)

##   
## svm.pred 0 1  
## 0 3988 462  
## 1 0 0

misClasificError <- mean(svm.pred != train.data$diabetes, na.rm=T)  
print(paste('Accuracy Model 2',1-misClasificError))

## [1] "Accuracy Model 2 0.896179775280899"

features<-x.train  
outcome<-train.data$diabetes  
  
svm\_tune <- tune(svm, train.x=features, train.y=outcome, kernel="linear", range = list(cost=10^(-1:1)))  
  
summary(svm\_tune)

##   
## Parameter tuning of 'svm':  
##   
## - sampling method: 10-fold cross validation   
##   
## - best parameters:  
## cost  
## 0.1  
##   
## - best performance: 0.1038202   
##   
## - Detailed performance results:  
## cost error dispersion  
## 1 0.1 0.1038202 0.01598861  
## 2 1.0 0.1038202 0.01598861  
## 3 10.0 0.1038202 0.01598861

svm.diabetes.new<-svm(diabetes ~ ., data=train.data, kernel="linear", cost=0.1, scale=TRUE)  
  
print(svm.diabetes.new)

##   
## Call:  
## svm(formula = diabetes ~ ., data = train.data, kernel = "linear",   
## cost = 0.1, scale = TRUE)  
##   
##   
## Parameters:  
## SVM-Type: C-classification   
## SVM-Kernel: linear   
## cost: 0.1   
##   
## Number of Support Vectors: 1021

svm.pred.new<-predict(svm.diabetes.new, newdata=test.data)  
table(svm.pred.new, test.data$diabetes)

##   
## svm.pred.new 0 1  
## 0 1709 197  
## 1 0 0

misClasificError.new <- mean(svm.pred.new != train.data$diabetes, na.rm=T)  
print(paste('Accuracy Model 2',1-misClasificError.new))

## [1] "Accuracy Model 2 0.896179775280899"

#### Logistic Regression Model Questions 2-4

model.3<-glm(diabetes~., family = binomial(link = "logit"), data = train.data)  
summary (model.3)

##   
## Call:  
## glm(formula = diabetes ~ ., family = binomial(link = "logit"),   
## data = train.data)  
##   
## Deviance Residuals:   
## Min 1Q Median 3Q Max   
## -1.9433 -0.4564 -0.2761 -0.1540 3.2873   
##   
## Coefficients:  
## Estimate Std. Error z value Pr(>|z|)   
## (Intercept) -10.000536 4.040569 -2.475 0.013322 \*   
## age 0.068215 0.004166 16.375 < 2e-16 \*\*\*  
## gendermale 0.610833 0.161543 3.781 0.000156 \*\*\*  
## race1Hispanic -0.496222 0.301530 -1.646 0.099830 .   
## race1Mexican 0.103645 0.250027 0.415 0.678481   
## race1White -0.709854 0.165392 -4.292 1.77e-05 \*\*\*  
## race1Other 0.192717 0.253325 0.761 0.446806   
## education9 - 11th Grade -0.535633 0.240615 -2.226 0.026008 \*   
## educationHigh School -0.198757 0.220708 -0.901 0.367831   
## educationSome College -0.123687 0.220873 -0.560 0.575487   
## educationCollege Grad -0.165696 0.238717 -0.694 0.487612   
## hh\_income 5000-9999 0.164339 0.482130 0.341 0.733208   
## hh\_income10000-14999 0.085185 0.438736 0.194 0.846051   
## hh\_income15000-19999 -0.086684 0.442862 -0.196 0.844816   
## hh\_income20000-24999 0.169848 0.437304 0.388 0.697721   
## hh\_income25000-34999 -0.384078 0.428977 -0.895 0.370609   
## hh\_income35000-44999 -0.166539 0.426551 -0.390 0.696216   
## hh\_income45000-54999 -0.408940 0.441573 -0.926 0.354394   
## hh\_income55000-64999 0.060934 0.442760 0.138 0.890539   
## hh\_income65000-74999 -0.488747 0.465890 -1.049 0.294150   
## hh\_income75000-99999 -0.280689 0.433399 -0.648 0.517215   
## hh\_incomemore 99999 -0.394045 0.424210 -0.929 0.352945   
## weight -0.011271 0.020615 -0.547 0.584572   
## height 0.004848 0.023615 0.205 0.837338   
## pulse 0.013301 0.004618 2.880 0.003977 \*\*   
## bmi 0.128778 0.058800 2.190 0.028517 \*   
## phys\_activeYes -0.183019 0.117649 -1.556 0.119795   
## smoke100Yes 0.198313 0.112726 1.759 0.078536 .   
## ---  
## 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: 2339.5 on 4422 degrees of freedom  
## AIC: 2395.5  
##   
## Number of Fisher Scoring iterations: 6

model.3a <-glmnet(x.train, diabetes.train, method = "glm", standardize = TRUE, family = "binomial")

Logistic Regression Prediction

model.3\_fitted<-predict(model.3a, x.test, type= "response")  
  
fitted.results.p <- ifelse( model.3\_fitted > 0.5,1,0)  
  
testing.model.3<-(as.numeric(test.data$diabetes)-1)  
  
model.3\_Error <- mean(fitted.results.p != testing.model.3, na.rm=T)  
  
print(paste('Accuracy Model 3',1-model.3\_Error))

## [1] "Accuracy Model 3 0.896020663491807"

The model that shoud be used based on the three analysis, would be the support vector classifier because it had the best accuracy of 89.62%. The logistis regression is the second best with an accuracy of 89.69%. The third best for this anaylysis is the classification tree wit 89.4% accuracy.

#### Question 5. List and describe at least two limitations of the model generated by this analysis. Limitations can be analytical or they can be regarding how the model would be used in practice.

1. One limitation for the svc is that it doesnt perform that great with a larger data set. If the dataset is to larger than it it will take a long time for the svm code to run depending on the range that is set. The function is looking for the best cost to implement in the prediction model which may take a while to find.
2. The second limitaion for svc is that it is harder to interpret compared to a logistic regression and classification tree. Since SVM uses hyper planes its hard to describe most of the time what is exactly is happening during that time and therefore hard to intrepret the final results. Logistic and classification tree are more straight to the point methods and allow for easy intrepretability.