Homework 6 - P8451

Will Simmons

3 March 2020

library(tidyverse)  
library(NHANES)  
library(caret)  
library(pROC)  
library(e1071)

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

set.seed(1)  
  
data =  
 NHANES %>%   
 select("Age", "Gender", "Race1", "Education", "HHIncome", "Weight", "Height", "Pulse", "Diabetes", "BMI", "PhysActive", "Smoke100")  
  
# Amelia::missmap(data)  
## need to drop missing  
  
data =   
 data %>%   
 drop\_na() %>%   
 mutate(  
 Diabetes = relevel(Diabetes, ref = "No")  
 )  
  
# data\_tf =  
# data %>%   
# drop\_na() %>%   
# mutate(  
# Diabetes = relevel(Diabetes, ref = "No"),  
# Diabetes\_tf = as.numeric(Diabetes) - 1,  
# Diabetes\_tf = as.logical(Diabetes\_tf))  
  
## all data formats look correct by default  
  
train\_idx =  
 createDataPartition(data$Diabetes,  
 p = 0.7,   
 list = FALSE)  
  
train = data[train\_idx, ]  
test = data[-train\_idx, ]  
  
rm(train\_idx)

# 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:

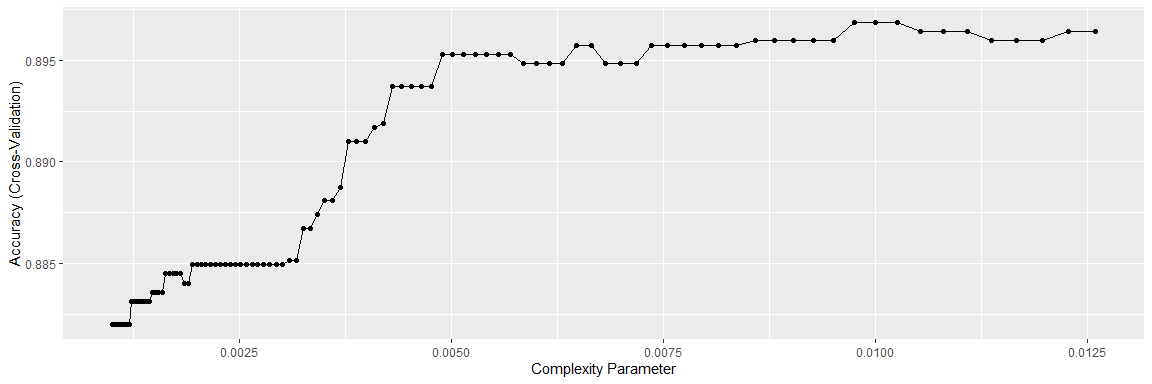
## a) Classification Tree

First, I’ll fit a classification tree using caret, tuning Cp using 10-fold cross-validation.

set.seed(1)  
  
tc = trainControl(method = "cv", number = 10)  
cp = 10^seq(-3, -1.9, length = 100) ## found range iteratively  
  
tree\_model =   
 train(Diabetes~.,  
 data = train,  
 method = "rpart",  
 trControl = tc,  
 tuneGrid = expand.grid(cp = cp)  
 )

As we can see in the graph of cross-validated tuning parameter values from this model, the max occurs at Cp =**0.0102591**.

ggplot(tree\_model)



tree\_model$bestTune

## cp  
## 92 0.01025914

## 0.01025914

Finally, we’ll look at the characteristics of our final classification tree, including important variables, a confusion matrix of how the model has classified the training data, and the final tree visualization.

### Important variables - tree-based model

varImp(tree\_model$finalModel) %>%   
 rownames\_to\_column(var = "Variable") %>%   
 arrange(desc(Overall)) %>%   
 rename(Importance = Overall) %>%   
 knitr::kable()

|  |  |
| --- | --- |
| Variable | Importance |
| Age | 66.530693 |
| BMI | 46.530415 |
| Weight | 39.002986 |
| Race1White | 22.083815 |
| HHIncomemore 99999 | 18.294932 |
| Race1Mexican | 10.698946 |
| PhysActiveYes | 10.477083 |
| Pulse | 9.585926 |
| Height | 4.546778 |
| HHIncome20000-24999 | 3.817335 |
| Smoke100Yes | 3.133389 |
| EducationSome College | 2.143529 |
| Gendermale | 0.000000 |
| Race1Hispanic | 0.000000 |
| Race1Other | 0.000000 |
| Education9 - 11th Grade | 0.000000 |
| EducationHigh School | 0.000000 |
| EducationSome College | 0.000000 |
| EducationCollege Grad | 0.000000 |
| HHIncome 5000-9999 | 0.000000 |
| HHIncome10000-14999 | 0.000000 |
| HHIncome15000-19999 | 0.000000 |
| HHIncome20000-24999 | 0.000000 |
| HHIncome25000-34999 | 0.000000 |
| HHIncome35000-44999 | 0.000000 |
| HHIncome45000-54999 | 0.000000 |
| HHIncome55000-64999 | 0.000000 |
| HHIncome65000-74999 | 0.000000 |
| HHIncome75000-99999 | 0.000000 |
| HHIncomemore 99999 | 0.000000 |

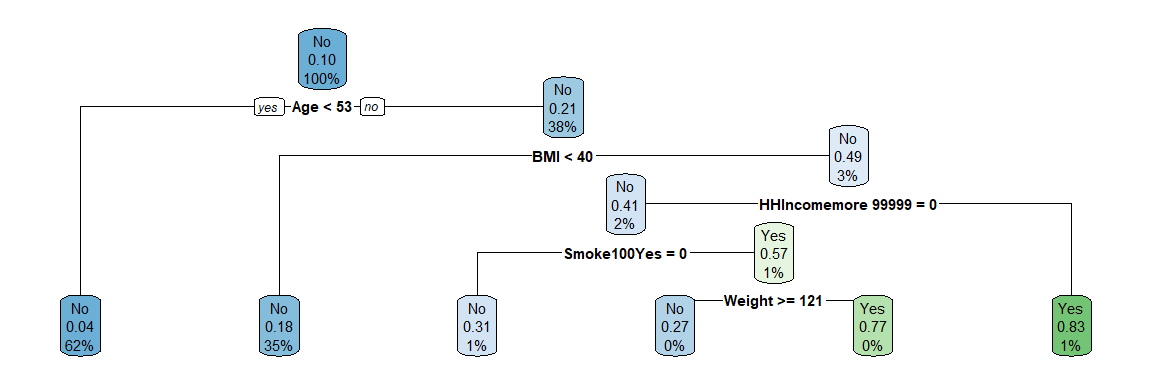
### Confusion matrix - tree-based model

tree\_pred = predict(tree\_model, newdata = train[, -9])  
table(tree\_pred, train$Diabetes)

##   
## tree\_pred No Yes  
## No 3979 425  
## Yes 9 37

### Classification Tree Plot - Diabetes Yes/No

rpart.plot::rpart.plot(tree\_model$finalModel)



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

First, we’ll tune a support vector classifier on the training data.

set.seed(1)  
  
# cost\_list = list(cost = 10^seq(-100, 0, length = 100))  
cost = 10^seq(-1, 1, length = 3)  
  
# svm\_model =  
# tune(svm,  
# train.x = Diabetes~.,  
# data = sample\_frac(train, 0.01),  
# kernel = "linear",  
# range = cost\_list,  
# scale = TRUE)  
  
  
# svm\_model$best.parameters  
# svm\_model$performances  
# plot(svm\_model)  
  
svm\_model2 =   
 train(Diabetes~.,  
 data = train,  
 method = "svmLinear",  
 trControl = tc,  
 tuneGrid = expand.grid(C = cost)  
 )

After iteratively adjusting the range of cross-validated Cost values, we can see that the best Cost parameter is actually the smallest in all cases. This is because the SVC cannot seem to classify this unbalanced data any better than classifiying all observations as ‘No’ for diabetes.

### Confusion matrix - SVM

To illustrate this, we can create a confusion matrix for the SVC-classified data and see that the CV-tuned model simply classifies all observations as ‘No’.

svm\_pred = predict(svm\_model2, newdata = train[, -9])  
table(svm\_pred, train$Diabetes)

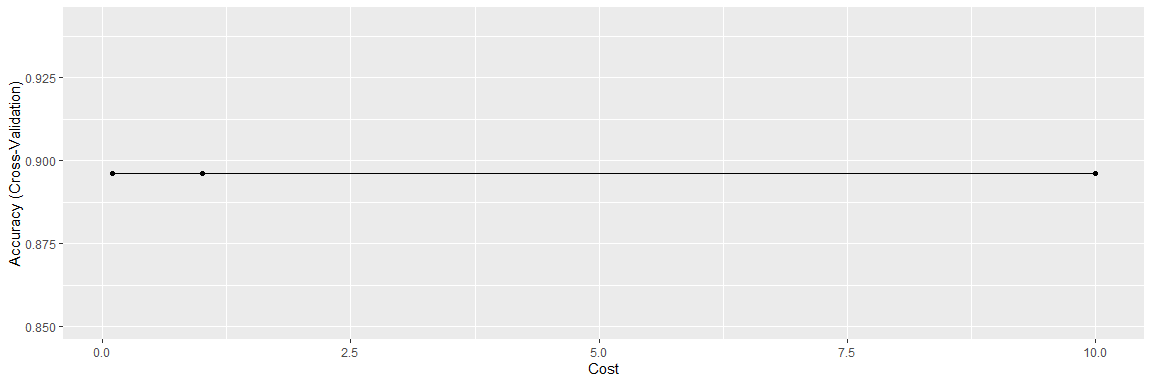
##   
## svm\_pred No Yes  
## No 3988 462  
## Yes 0 0

# misClasificError = mean(svm\_pred != train$Diabetes, na.rm=T)  
# svm\_acc = (1-misClasificError)

### Plot of accuracy rates across Cost parameter values

Finally, we can plot the cross-validated accuracy rates and see that the predictive accuracy does not change appreciably over small enough values of C.

ggplot(svm\_model2)



## c) Logistic regression

First, we’ll perform cross-validation on a simple logistic regression model so that we can estimate the cross-validated accuracy rate alongside other models.

set.seed(1)  
  
## use train to get CV error est.  
logistic\_model =  
 train(Diabetes~.,  
 data = train,  
 method = "glm",  
 trControl = tc)

### Parameter estimates - logistic regression

To illustrate what our model looks like, we can look at the parameter estimates:

logistic\_model$finalModel %>%   
 broom::tidy() %>%   
 knitr::kable()

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| term | estimate | std.error | statistic | p.value |
| (Intercept) | -11.5493193 | 4.1365835 | -2.7919948 | 0.0052384 |
| Age | 0.0659564 | 0.0041106 | 16.0455847 | 0.0000000 |
| Gendermale | 0.4127545 | 0.1626514 | 2.5376636 | 0.0111595 |
| Race1Hispanic | 0.0215245 | 0.2757075 | 0.0780701 | 0.9377723 |
| Race1Mexican | 0.1592733 | 0.2539024 | 0.6273012 | 0.5304618 |
| Race1White | -0.6430289 | 0.1658860 | -3.8763304 | 0.0001060 |
| Race1Other | 0.4463858 | 0.2465834 | 1.8102832 | 0.0702519 |
| Education9 - 11th Grade | -0.2946221 | 0.2339033 | -1.2595894 | 0.2078175 |
| EducationHigh School | -0.0582229 | 0.2193900 | -0.2653854 | 0.7907126 |
| EducationSome College | 0.0055202 | 0.2156155 | 0.0256019 | 0.9795748 |
| EducationCollege Grad | -0.1220218 | 0.2333182 | -0.5229845 | 0.6009851 |
| HHIncome 5000-9999 | 0.0299465 | 0.4572855 | 0.0654875 | 0.9477859 |
| HHIncome10000-14999 | -0.0558669 | 0.4147227 | -0.1347091 | 0.8928419 |
| HHIncome15000-19999 | -0.1578362 | 0.4177562 | -0.3778189 | 0.7055652 |
| HHIncome20000-24999 | -0.2450954 | 0.4142987 | -0.5915909 | 0.5541245 |
| HHIncome25000-34999 | -0.4902056 | 0.4024855 | -1.2179459 | 0.2232445 |
| HHIncome35000-44999 | -0.4420715 | 0.4046004 | -1.0926127 | 0.2745639 |
| HHIncome45000-54999 | -0.8785895 | 0.4270446 | -2.0573719 | 0.0396505 |
| HHIncome55000-64999 | -0.5528513 | 0.4244324 | -1.3025662 | 0.1927229 |
| HHIncome65000-74999 | -0.3648682 | 0.4235126 | -0.8615285 | 0.3889470 |
| HHIncome75000-99999 | -0.5375376 | 0.4097122 | -1.3119883 | 0.1895241 |
| HHIncomemore 99999 | -0.7487196 | 0.4007651 | -1.8682254 | 0.0617307 |
| Weight | -0.0111193 | 0.0212902 | -0.5222741 | 0.6014795 |
| Height | 0.0156822 | 0.0243054 | 0.6452158 | 0.5187874 |
| Pulse | 0.0175443 | 0.0045564 | 3.8504657 | 0.0001179 |
| BMI | 0.1181519 | 0.0599694 | 1.9702037 | 0.0488150 |
| PhysActiveYes | -0.1370340 | 0.1170590 | -1.1706408 | 0.2417432 |
| Smoke100Yes | 0.1727518 | 0.1129624 | 1.5292857 | 0.1261936 |

### Confusion matrix - logistic regression

As well as a confusion matrix indicating how the model has classified the training data.

logistic\_pred = predict(logistic\_model, newdata = train[, -9])  
table(logistic\_pred, train$Diabetes)

##   
## logistic\_pred No Yes  
## No 3934 413  
## Yes 54 49

# misClasificError = mean(logistic\_pred != train$Diabetes, na.rm=T)  
# logistic\_acc = (1 - misClasificError)

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

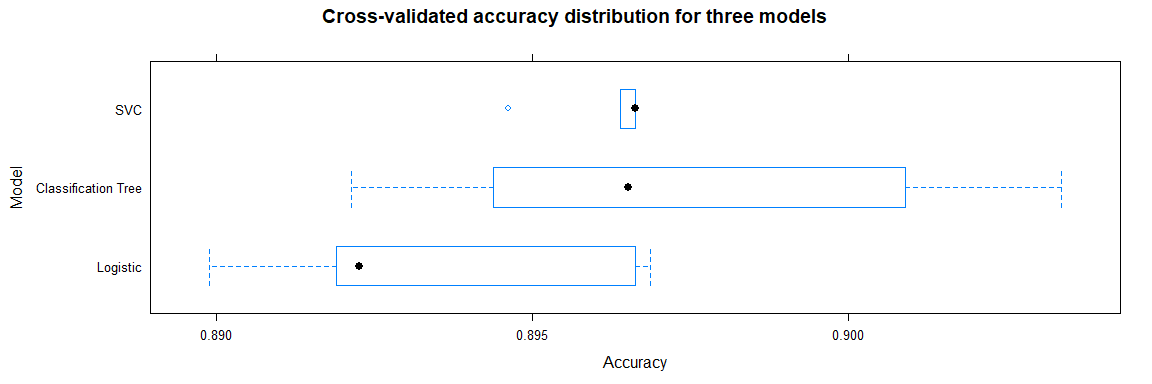
## Comparing performance across models

First, we’ll use the cross-validated accuracy rates of each of our resampled models to select the best-performing model.

resamp <- resamples(list("Classification Tree" = tree\_model,   
 "SVC" = svm\_model2,  
 "Logistic" = logistic\_model))  
  
a = summary(resamp)  
cv\_data =  
 a[[3]][[1]] %>%   
 as\_tibble() %>%   
 knitr::kable(caption = "Cross-validated accuracy estimates")

In the table above, we can see that both mean and median accuracy rates (cross-validated) for SVC and Tree methods are nearly identical. Let’s plot the distributions of their CV accuracy rates to look further.

bwplot(resamp,   
 metric = "Accuracy",   
 main = "Cross-validated accuracy distribution for three models",  
 ylab = "Model")



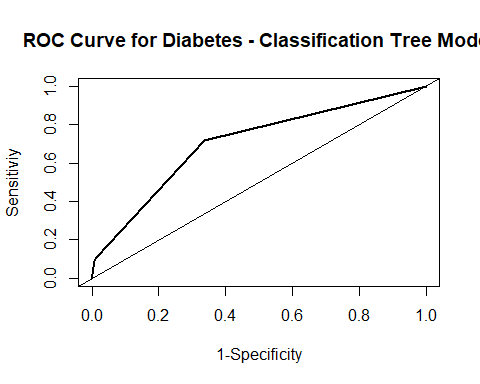
We can now see that the distributions of these models’ accuracy in resamples actually differs dramatically. SVC’s narrow distribution is likely due to its poor response to unbalanced data.

Therefore, since I know the SVC model will likely classify any new data as ‘No’ for diabetes, I will choose the Classification Tree method with comparable cross-validated accuracy.

# 4. Calculate final accuracy in a test set for the model you determine to be the most appropriate model.

Finally, I’ll calculate the test accuracy for the Classification Tree model and perform some other diagnostics such as the ROC curve.

set.seed(1)  
  
pred\_diab = predict(tree\_model, test)  
pred\_diab\_prob = predict(tree\_model, test, type = "prob")  
  
tree\_results = confusionMatrix(pred\_diab, test$Diabetes, positive = "Yes")  
#print(tree\_results)  
  
analysis = roc(response = test$Diabetes, predictor = pred\_diab\_prob[,2])  
plot(1 - analysis$specificities, analysis$sensitivities, type = "l",  
 ylab = "Sensitiviy",   
 xlab = "1-Specificity",  
 col = "black",  
 lwd = 2,  
 main = "ROC Curve for Diabetes - Classification Tree Model")  
   
abline(a = 0, b = 1)



tree\_results$overall

## Accuracy Kappa AccuracyLower AccuracyUpper AccuracyNull   
## 8.992655e-01 8.782460e-02 8.848708e-01 9.124177e-01 8.966422e-01   
## AccuracyPValue McnemarPValue   
## 3.708001e-01 3.553009e-38

We can see that the test accuracy for the classification tree model is equal to **0.8992655**.

# 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.

**Limitation 1.**

Although the tree-based classification method demonstrated the ‘best’ predictive accuracy, we can still see that it classified the overwhelming majority of its new data as ‘No’ due to our unbalanced data.

tree\_results$table

## Reference  
## Prediction No Yes  
## No 1703 186  
## Yes 6 11

Additional methods are needed to address the issue of unbalanced data, in which only 0.1036816 of observations actually have the outcome of interest.

**Limitation 2.**

A logical outcome of the previous limitation is that I would not use this model to predict diabetes outcomes in any population, given its extremely low sensitivity (**0.0558376**), even despite its high specificity (**0.9964892**). Clinicians or practitioners could essentially do just as well as this ‘predictive’ algorithm by classifying everyone as diabetes-negative.