Assignment 6: Comparison between Classification Trees, SVM and Logistic Regression

2024-02-26

library(tidyverse)  
library(rpart)  
library(caret)  
library(rpart.plot)  
library(pROC)  
library(kernlab)  
library(e1071)

The posted article by Yu et al utilized NHANES data from 1999-2004 to predict diabetes and pre-diabetes using Support Vector Machines. You will conduct a similar analysis using data within the NHANES package in R. For this exercise, you will try to predict Diabetes using similar (although not all) variables. The available data is also different, so you won’t get the same answers.

REMINDER: Look at the frequency of your outcome variable to check for balance

For this assignment, you will:

1. Load the NHANES data using the NHANES R package

set.seed(123)  
library(NHANES)  
  
nhanes = NHANES %>% janitor::clean\_names()

1. Restrict the NHANES data to the list of 11 variables below. Perform light data cleaning. Determine if you want to exclude any of the features before you start. Partition 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)  
nhanes = nhanes |> select(age, race1, education, hh\_income, weight, height, pulse, diabetes, bmi, phys\_active, smoke100) |> na.omit()  
  
partition <- createDataPartition(y = nhanes$diabetes, p = 0.7, list = FALSE)  
  
# Creating training and testing sets  
train\_data <- nhanes[partition, ]  
test\_data <- nhanes[-partition, ]

1. Construct three prediction models to predict diabetes using the features from NHANES. You will optimize each model using cross-validation to choose hyperparameters in the training data and then compare performance across models. You will use the following three algorithms to create your prediction models:
2. Classification Tree

AUC = 0.803

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

set.seed(123)  
  
# Setting up cross-validation and specifying AUC as the metric  
trainControl <- trainControl(method = "cv",  
 number = 10,  
 summaryFunction = twoClassSummary,  
 classProbs = TRUE,   
 savePredictions = "final")  
  
# Define a tuning grid for SVM hyperparameters, focusing on 'C'  
tuningGrid <- expand.grid(C = seq(0.001, 2, length = 20))  
  
# Train the SVM model with cross-validation  
svmModel <- train(diabetes ~ .,  
 data = train\_data,  
 method = "svmLinear",  
 trControl = trainControl,  
 tuneGrid = tuningGrid,  
 metric = "ROC",  
 preProcess = c("center", "scale")) # Pre-processing steps

## maximum number of iterations reached 0.01085536 0.00980073maximum number of iterations reached 0.0105926 0.009455061maximum number of iterations reached 0.006430752 0.006051302maximum number of iterations reached 0.007632995 0.007202278maximum number of iterations reached 0.008759899 0.008002154maximum number of iterations reached 0.009586973 0.008841425maximum number of iterations reached 0.007648205 0.007093961maximum number of iterations reached 0.006859105 0.006472282maximum number of iterations reached 0.006895285 0.006482649maximum number of iterations reached 0.003274548 0.003195983maximum number of iterations reached 0.002348811 0.002296483maximum number of iterations reached 0.003629752 0.003534933maximum number of iterations reached 0.007452818 0.00700977maximum number of iterations reached 0.006690161 0.006301762maximum number of iterations reached 0.002641242 0.002593289maximum number of iterations reached 0.007056303 0.006651266maximum number of iterations reached 0.0075106 0.007011183maximum number of iterations reached 0.008539698 0.00782128maximum number of iterations reached 0.001572821 0.001553448maximum number of iterations reached 0.003797638 0.003675989maximum number of iterations reached 0.01060242 0.009702986maximum number of iterations reached 0.01042833 0.009333045maximum number of iterations reached 0.009646118 0.00870616maximum number of iterations reached 0.009578017 0.008671998maximum number of iterations reached 0.01016488 0.009357056maximum number of iterations reached 0.007959817 0.007343079maximum number of iterations reached 0.008472749 0.007762795maximum number of iterations reached 0.009431544 0.008634122maximum number of iterations reached 0.007656596 0.007130574maximum number of iterations reached 0.004754928 0.004506524maximum number of iterations reached 0.01033998 0.009396155maximum number of iterations reached 0.009447884 0.008605109maximum number of iterations reached 0.008755594 0.007966667maximum number of iterations reached 0.01014694 0.009092049maximum number of iterations reached 0.009498351 0.008647564maximum number of iterations reached 0.009851991 0.008943701maximum number of iterations reached 0.008684208 0.007964971maximum number of iterations reached 0.01106746 0.009980503maximum number of iterations reached 0.008382971 0.00777695maximum number of iterations reached 0.006572887 0.006206025maximum number of iterations reached 0.01223453 0.01074358maximum number of iterations reached 0.008093871 0.007529459maximum number of iterations reached 0.005502343 0.005251647maximum number of iterations reached 0.01110284 0.009918239maximum number of iterations reached 0.00846611 0.007835372maximum number of iterations reached 0.006030454 0.005673067maximum number of iterations reached 0.006623621 0.006257477maximum number of iterations reached 0.009519082 0.00884415maximum number of iterations reached 0.00287108 0.002797764maximum number of iterations reached 0.005506588 0.005234291maximum number of iterations reached 0.003209828 0.003119843maximum number of iterations reached 0.006845263 0.006451546maximum number of iterations reached 0.008866409 0.008258484maximum number of iterations reached 0.007463622 0.007026245maximum number of iterations reached 0.00686748 0.006430351maximum number of iterations reached 0.008048106 0.00746089maximum number of iterations reached 0.003306424 0.003216098maximum number of iterations reached 0.007759722 0.007313184maximum number of iterations reached 0.00794312 0.007363876maximum number of iterations reached 0.004088451 0.003935129maximum number of iterations reached 0.01155396 0.01043179maximum number of iterations reached 0.008198767 0.007424497maximum number of iterations reached 0.01044579 0.009282173maximum number of iterations reached 0.005933092 0.005566327maximum number of iterations reached 0.005670222 0.005412959maximum number of iterations reached 0.007386427 0.006915731maximum number of iterations reached 0.007949888 0.007234836maximum number of iterations reached 0.007719597 0.007161224maximum number of iterations reached 0.00758063 0.007048963maximum number of iterations reached 0.009248278 0.008524408maximum number of iterations reached 0.008395919 0.007695226maximum number of iterations reached 0.007688617 0.007240563maximum number of iterations reached 0.0125506 0.01126069maximum number of iterations reached 0.009926728 0.008994845maximum number of iterations reached 0.006899033 0.006526866maximum number of iterations reached 0.00886823 0.008200474maximum number of iterations reached 0.009855733 0.008950335maximum number of iterations reached 0.005185814 0.004975901maximum number of iterations reached 0.0087981 0.008187907maximum number of iterations reached 0.005767575 0.005419254maximum number of iterations reached 0.006804734 0.006341295maximum number of iterations reached 0.002715361 0.002654047maximum number of iterations reached 0.004086087 0.003926573maximum number of iterations reached 0.005278274 0.005009466maximum number of iterations reached 0.006743362 0.006303314maximum number of iterations reached 0.007096468 0.006594738maximum number of iterations reached 0.007842709 0.007264613maximum number of iterations reached 0.007385702 0.006921812maximum number of iterations reached 0.007947019 0.007259249maximum number of iterations reached 0.001974467 0.00194343maximum number of iterations reached 0.004743656 0.004523841maximum number of iterations reached 0.003030654 0.002930455maximum number of iterations reached 0.006389767 0.006090893maximum number of iterations reached 0.00518114 0.00492724maximum number of iterations reached 0.004758579 0.004544472maximum number of iterations reached 0.007982299 0.007343182maximum number of iterations reached 0.006825749 0.006451215maximum number of iterations reached 0.005154153 0.004975442maximum number of iterations reached 0.00626369 0.005936234maximum number of iterations reached 0.008861044 0.008230845maximum number of iterations reached 0.009467417 0.008577508maximum number of iterations reached 0.008248263 0.007561548maximum number of iterations reached 0.01115784 0.009951679maximum number of iterations reached 0.007878095 0.007270897maximum number of iterations reached 0.008921287 0.008164513maximum number of iterations reached 0.004870082 0.004690076maximum number of iterations reached 0.01032689 0.009344155maximum number of iterations reached 0.008934256 0.008256321maximum number of iterations reached 0.005917513 0.005622288maximum number of iterations reached 0.0084884 0.007650453maximum number of iterations reached 0.006302658 0.005955411maximum number of iterations reached 0.006133949 0.005866582maximum number of iterations reached 0.008328134 0.007623662maximum number of iterations reached 0.006726749 0.006218494maximum number of iterations reached 0.009658162 0.008767478maximum number of iterations reached 0.005847811 0.005577936maximum number of iterations reached 0.006985116 0.006585462maximum number of iterations reached 0.0100694 0.009158904maximum number of iterations reached 0.005905616 0.005598334maximum number of iterations reached 0.005659003 0.005341061maximum number of iterations reached 0.0126926 0.01136104maximum number of iterations reached 0.008263454 0.007530344maximum number of iterations reached 0.01262747 0.01111145maximum number of iterations reached 0.01025998 0.009062715maximum number of iterations reached 0.003334462 0.003229274maximum number of iterations reached 0.006229387 0.0058589maximum number of iterations reached 0.009236182 0.008512616maximum number of iterations reached 0.00777726 0.007246239maximum number of iterations reached 0.006665268 0.006248736maximum number of iterations reached 0.00751335 0.006892072maximum number of iterations reached 0.003569568 0.003447011maximum number of iterations reached 0.007482702 0.006877251maximum number of iterations reached 0.008746374 0.008041872maximum number of iterations reached 0.003966533 0.003808976maximum number of iterations reached 0.006042926 0.005782704maximum number of iterations reached 0.009792682 0.008901119maximum number of iterations reached 0.002297919 0.002257438maximum number of iterations reached 0.007563345 0.006969852maximum number of iterations reached 0.009540281 0.008618758maximum number of iterations reached 0.006584011 0.006236922maximum number of iterations reached 0.01287895 0.01116772maximum number of iterations reached 0.009089759 0.008320187maximum number of iterations reached 0.01197081 0.01048455maximum number of iterations reached 0.005654533 0.005385424maximum number of iterations reached 0.005408894 0.005162888maximum number of iterations reached 0.003855094 0.003698424maximum number of iterations reached 0.00710971 0.006698959maximum number of iterations reached 0.008654061 0.008004025maximum number of iterations reached 0.007538914 0.007086476maximum number of iterations reached 0.006800459 0.006318979maximum number of iterations reached 0.007175998 0.006674645maximum number of iterations reached 0.005644954 0.005338886maximum number of iterations reached 0.007699636 0.007269111maximum number of iterations reached 0.009595553 0.00875419maximum number of iterations reached 0.003417332 0.003319191maximum number of iterations reached 0.004297633 0.004161713maximum number of iterations reached 0.005408163 0.005150764maximum number of iterations reached 0.009585237 0.008731246maximum number of iterations reached 0.006813322 0.006448285maximum number of iterations reached 0.007903999 0.007492719maximum number of iterations reached 0.009278815 0.008223741maximum number of iterations reached 0.00842558 0.007666027maximum number of iterations reached 0.007939963 0.00733653maximum number of iterations reached 0.005067282 0.004844863maximum number of iterations reached 0.005668442 0.00539391maximum number of iterations reached 0.007682646 0.007130121maximum number of iterations reached 0.006242993 0.005841008maximum number of iterations reached 0.003272703 0.003188913maximum number of iterations reached 0.006525548 0.006153936maximum number of iterations reached 0.005197164 0.004883786maximum number of iterations reached 0.01296987 0.0114526maximum number of iterations reached 0.0101001 0.009101941maximum number of iterations reached 0.007991555 0.007410133maximum number of iterations reached 0.006310006 0.006024063maximum number of iterations reached 0.006599213 0.006187233maximum number of iterations reached 0.007348458 0.006846504maximum number of iterations reached 0.008801611 0.008246722maximum number of iterations reached 0.005906047 0.005596697maximum number of iterations reached 0.002287762 0.002243496maximum number of iterations reached 0.004955927 0.004793578maximum number of iterations reached 0.01167813 0.01014945maximum number of iterations reached 0.0106535 0.009471494maximum number of iterations reached 0.01114062 0.009874986maximum number of iterations reached 0.007939136 0.007357904maximum number of iterations reached 0.01152945 0.01013226maximum number of iterations reached 0.01011191 0.009147048maximum number of iterations reached 0.01251445 0.01128809maximum number of iterations reached 0.009321572 0.008205296maximum number of iterations reached 0.009521744 0.008569203maximum number of iterations reached 0.007064436 0.006590269maximum number of iterations reached 0.006179598 0.005844302maximum number of iterations reached 0.01003708 0.00900283maximum number of iterations reached 0.009261981 0.008540321maximum number of iterations reached 0.009409718 0.008414993maximum number of iterations reached 0.01341965 0.01177212maximum number of iterations reached 0.01101157 0.009913959maximum number of iterations reached 0.009148721 0.008396371maximum number of iterations reached 0.009903562 0.008774564maximum number of iterations reached 0.009245791 0.008384683maximum number of iterations reached 0.0112127 0.01012064maximum number of iterations reached 0.01177063 0.01062817

# Print Model  
print(svmModel)

## 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: 4005, 4005, 4004, 4005, 4005, 4006, ...   
## Resampling results across tuning parameters:  
##   
## C ROC Sens Spec  
## 0.0010000 0.6306119 1 0   
## 0.1062105 0.5847105 1 0   
## 0.2114211 0.6380646 1 0   
## 0.3166316 0.5998203 1 0   
## 0.4218421 0.6055757 1 0   
## 0.5270526 0.6163412 1 0   
## 0.6322632 0.6125682 1 0   
## 0.7374737 0.6077398 1 0   
## 0.8426842 0.6227003 1 0   
## 0.9478947 0.6276718 1 0   
## 1.0531053 0.5807192 1 0   
## 1.1583158 0.6220556 1 0   
## 1.2635263 0.6065399 1 0   
## 1.3687368 0.5887346 1 0   
## 1.4739474 0.5973733 1 0   
## 1.5791579 0.6238115 1 0   
## 1.6843684 0.5954001 1 0   
## 1.7895789 0.6258926 1 0   
## 1.8947895 0.6168653 1 0   
## 2.0000000 0.6266020 1 0   
##   
## ROC was used to select the optimal model using the largest value.  
## The final value used for the model was C = 0.2114211.

AUC = 0.638

1. Logistic regression

set.seed(123)  
  
# Set up cross-validation with down-sampling for imbalance handling  
train.control <- trainControl(method = "cv", number = 10, summaryFunction = twoClassSummary, classProbs = TRUE, savePredictions = TRUE, sampling = "down")  
  
# Train the logistic regression model  
logistic.diabetes <- train(  
 diabetes ~ .,  
 data = train\_data,  
 method = "glm",  
 family = "binomial",  
 trControl = train.control,  
 metric = "ROC"  
)  
  
# Display the model  
print(logistic.diabetes)

## Generalized Linear Model   
##   
## 4450 samples  
## 10 predictor  
## 2 classes: 'No', 'Yes'   
##   
## No pre-processing  
## Resampling: Cross-Validated (10 fold)   
## Summary of sample sizes: 4005, 4005, 4004, 4005, 4005, 4006, ...   
## Addtional sampling using down-sampling  
##   
## Resampling results:  
##   
## ROC Sens Spec   
## 0.8116997 0.7098695 0.7574468

# model coefficients as an indication of importance  
coef(logistic.diabetes$finalModel)

## (Intercept) age race1Hispanic   
## -17.75238906 0.08363345 0.00587355   
## race1Mexican race1White race1Other   
## -0.33996094 -0.31476772 0.38709174   
## `education9 - 11th Grade` `educationHigh School` `educationSome College`   
## -0.59279910 -0.12930804 -0.18752616   
## `educationCollege Grad` `hh\_income 5000-9999` `hh\_income10000-14999`   
## -0.74332582 -0.43085274 -0.97167538   
## `hh\_income15000-19999` `hh\_income20000-24999` `hh\_income25000-34999`   
## -0.90323120 -0.95586731 -1.01499637   
## `hh\_income35000-44999` `hh\_income45000-54999` `hh\_income55000-64999`   
## -0.74797103 -0.90332763 -1.03173653   
## `hh\_income65000-74999` `hh\_income75000-99999` `hh\_incomemore 99999`   
## -1.19233954 -0.79696627 -1.21677931   
## weight height pulse   
## -0.04819816 0.05690495 0.01670732   
## bmi phys\_activeYes smoke100Yes   
## 0.25677116 0.35976558 0.53279940

AUC = 0.809

1. Select an “optimal” model and calculate final evaluation metrics in the test set. ONLY ONE MODEL SHOULD BE APPLIED IN THE TEST SET What do you conclude about your final model’s performance?

The optimal model that I have chosen is the *logistic regression* due to its higher AUC value, indicating it has the best overall ability to distinguish between the classes across all thresholds. It also has slightly higher sensitivity, meaning it’s marginally better at identifying true positives and slightly higher specificity, indicating a marginally better performance at identifying true negatives.

# Prediction and Evaluation on the Testing Set  
  
# Create predictions on the test set  
pred.diabetes <- predict(logistic.diabetes, newdata = test\_data)  
  
# Evaluation results on the test set  
eval.results <- confusionMatrix(pred.diabetes, test\_data$diabetes, positive = "Yes")  
print(eval.results)

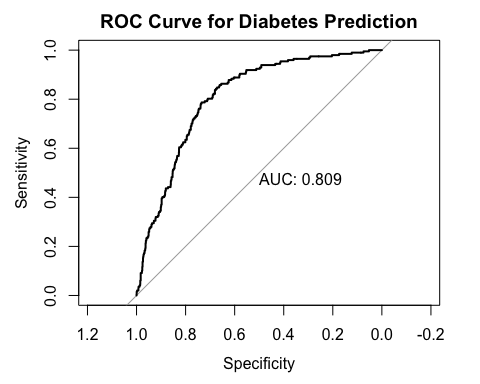
## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction No Yes  
## No 1194 39  
## Yes 515 158  
##   
## Accuracy : 0.7093   
## 95% CI : (0.6884, 0.7297)  
## No Information Rate : 0.8966   
## P-Value [Acc > NIR] : 1   
##   
## Kappa : 0.242   
##   
## Mcnemar's Test P-Value : <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   
##

# Predictions as probabilities on the test set  
pred.diabetes.prob <- predict(logistic.diabetes, newdata = test\_data, type = "prob")  
  
# ROC analysis  
roc.analysis.2 <- roc(response = as.numeric(test\_data$diabetes), predictor = pred.diabetes.prob[,2])

## Setting levels: control = 1, case = 2

## Setting direction: controls < cases

plot(roc.analysis.2, print.auc = TRUE, main = "ROC Curve for Diabetes Prediction")



**Model Performance Overview** \* The model demonstrates fair ability to identify positive cases but struggles with a high false positive rate, as indicated by the low PPV. \* The high NPV and sensitivity suggest the model is quite conservative, effectively identifying negative cases but at the cost of missing or incorrectly predicting a significant portion of positive cases.

1. In this analysis, we’ve used Race as one of the predictors in the model. Briefly discuss the ethical considerations of including race in a disease prediction model. (no more than 1-2 paragraphs)