Homework 4.5- Caravan dataset

Hamed

3/2/2020

## This question uses the Caravan data set.

##(a) Create a training set consisting of the first 1,000 observations, and a test set consisting of the remaining observations.

library(gbm)

## Loaded gbm 2.1.5

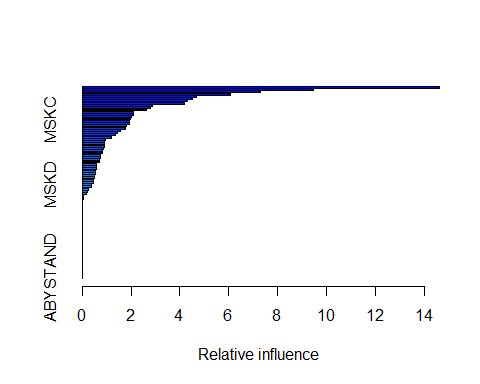
library(ISLR)  
data("Caravan")  
  
set.seed(1)  
Caravan$Purchase = ifelse(Caravan$Purchase=="Yes",1,0)  
  
caravan.train = Caravan[1:1000,]  
caravan.test = Caravan[1001:5822,]

##(b) Fit a boosting model to the training set with Purchase as the response and the other variables as predictors. Use 1,000 trees, and a shrinkage value of 0.01. Which predictors appear to be the most important?

#Variable importance  
caravan.boost = gbm(formula=Purchase ~ .,data=caravan.train, n.trees=1000,  
 shrinkage=.01)

## Distribution not specified, assuming bernoulli ...

summary(caravan.boost)



## var rel.inf  
## PPERSAUT PPERSAUT 14.63504779  
## MKOOPKLA MKOOPKLA 9.47091649  
## MOPLHOOG MOPLHOOG 7.31457416  
## MBERMIDD MBERMIDD 6.08651965  
## PBRAND PBRAND 4.66766122  
## MGODGE MGODGE 4.49463264  
## ABRAND ABRAND 4.32427755  
## MINK3045 MINK3045 4.17590619  
## MOSTYPE MOSTYPE 2.86402583  
## PWAPART PWAPART 2.78191075  
## MAUT1 MAUT1 2.61929152  
## MBERARBG MBERARBG 2.10480508  
## MSKA MSKA 2.10185152  
## MAUT2 MAUT2 2.02172510  
## MSKC MSKC 1.98684345  
## MINKGEM MINKGEM 1.92122708  
## MGODPR MGODPR 1.91777542  
## MBERHOOG MBERHOOG 1.80710618  
## MGODOV MGODOV 1.78693913  
## PBYSTAND PBYSTAND 1.57279593  
## MSKB1 MSKB1 1.43551401  
## MFWEKIND MFWEKIND 1.37264255  
## MRELGE MRELGE 1.20805179  
## MOPLMIDD MOPLMIDD 0.93791970  
## MINK7512 MINK7512 0.92590720  
## MINK4575 MINK4575 0.91745993  
## MGODRK MGODRK 0.90765539  
## MFGEKIND MFGEKIND 0.85745374  
## MZPART MZPART 0.82531066  
## MRELOV MRELOV 0.80731252  
## MINKM30 MINKM30 0.74126812  
## MHKOOP MHKOOP 0.73690793  
## MZFONDS MZFONDS 0.71638323  
## MAUT0 MAUT0 0.71388052  
## MHHUUR MHHUUR 0.59287247  
## APERSAUT APERSAUT 0.58056986  
## MOSHOOFD MOSHOOFD 0.58029563  
## MSKB2 MSKB2 0.53885275  
## PLEVEN PLEVEN 0.53052444  
## MINK123M MINK123M 0.50660603  
## MBERARBO MBERARBO 0.48596479  
## MGEMOMV MGEMOMV 0.47614792  
## PMOTSCO PMOTSCO 0.46163590  
## MSKD MSKD 0.39735297  
## MBERBOER MBERBOER 0.36417546  
## MGEMLEEF MGEMLEEF 0.26166240  
## MFALLEEN MFALLEEN 0.21448118  
## MBERZELF MBERZELF 0.15906143  
## MOPLLAAG MOPLLAAG 0.05263665  
## MAANTHUI MAANTHUI 0.03766014  
## MRELSA MRELSA 0.00000000  
## PWABEDR PWABEDR 0.00000000  
## PWALAND PWALAND 0.00000000  
## PBESAUT PBESAUT 0.00000000  
## PVRAAUT PVRAAUT 0.00000000  
## PAANHANG PAANHANG 0.00000000  
## PTRACTOR PTRACTOR 0.00000000  
## PWERKT PWERKT 0.00000000  
## PBROM PBROM 0.00000000  
## PPERSONG PPERSONG 0.00000000  
## PGEZONG PGEZONG 0.00000000  
## PWAOREG PWAOREG 0.00000000  
## PZEILPL PZEILPL 0.00000000  
## PPLEZIER PPLEZIER 0.00000000  
## PFIETS PFIETS 0.00000000  
## PINBOED PINBOED 0.00000000  
## AWAPART AWAPART 0.00000000  
## AWABEDR AWABEDR 0.00000000  
## AWALAND AWALAND 0.00000000  
## ABESAUT ABESAUT 0.00000000  
## AMOTSCO AMOTSCO 0.00000000  
## AVRAAUT AVRAAUT 0.00000000  
## AAANHANG AAANHANG 0.00000000  
## ATRACTOR ATRACTOR 0.00000000  
## AWERKT AWERKT 0.00000000  
## ABROM ABROM 0.00000000  
## ALEVEN ALEVEN 0.00000000  
## APERSONG APERSONG 0.00000000  
## AGEZONG AGEZONG 0.00000000  
## AWAOREG AWAOREG 0.00000000  
## AZEILPL AZEILPL 0.00000000  
## APLEZIER APLEZIER 0.00000000  
## AFIETS AFIETS 0.00000000  
## AINBOED AINBOED 0.00000000  
## ABYSTAND ABYSTAND 0.00000000

## (c) Use the boosting model to predict the response on the test data.

## Predict that a person will make a purchase if the estimated probability of purchase is greater than 20 %.

#Prediction   
caravan.probs = predict(caravan.boost, newdata=caravan.test,n.trees=1000,type="response")  
  
caravan.pred = ifelse(caravan.probs>.2,"Yes","No")  
caravan.pred[1:5] #first five predicted values

## [1] "No" "No" "No" "No" "No"

test.y = ifelse(caravan.test$Purchase==1,"Yes","No")  
test.y[1:5] #fist five

## [1] "No" "Yes" "No" "No" "No"

# Confusion matrix  
confusionMat=table(caravan.pred,test.y)  
  
#What fraction of the people predicted to make a purchase do in fact make one?   
confusionMat[4]/(confusionMat[3]+confusionMat[4])

## [1] 0.1141869

# How does this compare with the results obtained from applying KNN or logistic regression to this data set?

# Apply Logistic regression  
lm.caravan <- glm(Purchase ~ . , data=caravan.train, family=binomial)  
lm.prob <- predict(lm.caravan, caravan.test, type="response")  
lm.pred <- ifelse(lm.prob > 0.2, "Yes", "No")  
  
# confusion matrix  
CF=table(test.y, lm.pred)  
CF

## lm.pred  
## test.y No Yes  
## No 4183 350  
## Yes 231 58

#What fraction of the people predicted to make a purchase do in fact make one?  
CF[4]/(CF[3]+CF[4])

## [1] 0.1421569

##(d) Try a stacking ensemble with at least two different models.set the response variable to factor

library(mlbench)  
library(caret)

## Loading required package: lattice

## Loading required package: ggplot2

library(caretEnsemble)

##   
## Attaching package: 'caretEnsemble'

## The following object is masked from 'package:ggplot2':  
##   
## autoplot

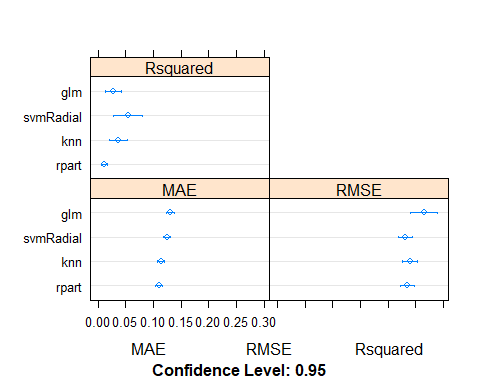
# Reload the data and convert the response variable to factor  
  
set.seed(1)  
Caravan2=as.data.frame(Caravan)  
  
Caravan$Purchase<-as.factor(Caravan2$Purchase)  
caravan2.train = Caravan2[1:1000,]  
caravan2.test = Caravan2[1001:5822,]  
table(caravan2.train$Purchase)

##   
## 0 1   
## 941 59

# Example of Stacking algorithms  
# create submodels  
control<-trainControl(method="repeatedcv", number=10, repeats=3,  
 savePredictions=TRUE, classProbs=TRUE)  
  
algorithmList <- c('rpart','glm', 'knn', 'svmRadial')  
  
models <- caretList(Purchase~.,data=caravan2.train,trControl=control  
 ,methodList=algorithmList)  
  
results <- resamples(models)  
summary(results)

##   
## Call:  
## summary.resamples(object = results)  
##   
## Models: rpart, glm, knn, svmRadial   
## Number of resamples: 30   
##   
## MAE   
## Min. 1st Qu. Median Mean 3rd Qu. Max. NA's  
## rpart 0.08656441 0.1007138 0.1092998 0.1101335 0.1141927 0.1566368 0  
## glm 0.09981459 0.1136283 0.1300362 0.1306945 0.1391407 0.1790675 0  
## knn 0.08655556 0.1049003 0.1124369 0.1143460 0.1225215 0.1597778 0  
## svmRadial 0.09983459 0.1126753 0.1247236 0.1247884 0.1340338 0.1707543 0  
##   
## RMSE   
## Min. 1st Qu. Median Mean 3rd Qu. Max. NA's  
## rpart 0.1736038 0.2159735 0.2359127 0.2345181 0.2499894 0.3327977 0  
## glm 0.1796049 0.2244937 0.2474569 0.2648087 0.2782417 0.4467130 0  
## knn 0.1730090 0.2164338 0.2379691 0.2397362 0.2556221 0.3366612 0  
## svmRadial 0.1593518 0.2084532 0.2309396 0.2315712 0.2492444 0.3251517 0  
##   
## Rsquared   
## Min. 1st Qu. Median Mean 3rd Qu. Max.  
## rpart 5.316321e-04 0.0007333092 0.00727405 0.01140418 0.02135469 0.03309327  
## glm 3.025651e-05 0.0018185298 0.01225743 0.02812641 0.03623453 0.16072619  
## knn 2.454561e-05 0.0034217221 0.01878565 0.03663336 0.05615734 0.14282734  
## svmRadial 1.194540e-05 0.0045773998 0.02445512 0.05435595 0.07956876 0.26626208  
## NA's  
## rpart 12  
## glm 0  
## knn 0  
## svmRadial 0

dotplot(results)



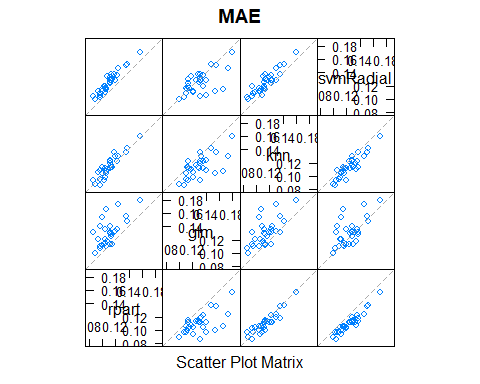
When we combine the predictions of different models using stacking, it is desirable that the predictions made by the sub-models have low correlation. This would suggest that the models are skillful but in different ways, allowing a new classifier to figure out how to get the best from each model for an improved score.

If the predictions for the sub-models were highly correlated (>0.75) then they would be making the same or very similar predictions most of the time reducing the benefit of combining the predictions.

# correlation between results  
modelCor(results)

## rpart glm knn svmRadial  
## rpart 1.0000000 0.6336961 0.9176363 0.9556571  
## glm 0.6336961 1.0000000 0.6549507 0.6104222  
## knn 0.9176363 0.6549507 1.0000000 0.9050058  
## svmRadial 0.9556571 0.6104222 0.9050058 1.0000000

#We can see that all pairs of predictions have generally low correlation.  
#The two methods with the highest correlation between their predictions are Logistic Regression (GLM) and kNN at 0.517 correlation which is not considered high (>0.75).  
  
splom(results)



Let’s combine the predictions of the classifiers using a simple linear model. stack using glm

stackControl <- trainControl(method="repeatedcv", number=10, repeats=3, savePredictions=TRUE, classProbs=TRUE)  
set.seed(1)  
stack.glm <-caretStack(models, method="glm", trControl=stackControl)  
print(stack.glm)

## A glm ensemble of 4 base models: rpart, glm, knn, svmRadial  
##   
## Ensemble results:  
## Generalized Linear Model   
##   
## 3000 samples  
## 4 predictor  
##   
## No pre-processing  
## Resampling: Cross-Validated (10 fold, repeated 3 times)   
## Summary of sample sizes: 2700, 2700, 2700, 2700, 2700, 2700, ...   
## Resampling results:  
##   
## RMSE Rsquared MAE   
## 0.2319215 0.03232413 0.1087416

We can also use more sophisticated algorithms to combine predictions in an effort to tease out when best to use the different methods. In this case, we can use the random forest algorithm to combine the predictions.

# stack using random forest  
# set.seed(123)  
# library(mlbench)  
# library(caret)  
# library(caretEnsemble)  
#   
# stack.rf <- caretStack(models, method="rf", metric="Accuracy", trControl=stackControl)  
# print(stack.rf)