DM Final

Chen Zhang

December 5, 2018

### Data Reading

#### 1.Read the Biodeg data into R.

bio.data = read.csv("Biodeg.csv",header = T,sep = ",")

###2.V29, V24, and V25 are factor variables. Convert them to factor variables.

bio.data$V29 = as.factor(bio.data$V29)  
bio.data$V24 = as.factor(bio.data$V24)  
bio.data$V25 = as.factor(bio.data$V25)  
class(bio.data$V29);class(bio.data$V24);class(bio.data$V25)

## [1] "factor"

## [1] "factor"

## [1] "factor"

### 3. Partition the data to 50% train data and 50% test data.

**Use set.seed(123) for the sampling.**

set.seed(123)  
train.sample = sample(1:dim(bio.data)[1],dim(bio.data)\*0.5, replace = FALSE)  
test.sample = -train.sample  
train = bio.data[train.sample,]  
test = bio.data[test.sample,]

**For any cross-validation or randomForest running use set.seed(1) before running any model or cross-validation.**

### Tree Model

### 1.Build tree model on train data and prune it using best tree size to classify V42 (RB or NRB) type using V1 through V41. What is best tree size?

library(tree)  
set.seed(1)  
tree.bio = tree(V42~., data=train) #build a tree model  
  
#prune the tree  
cv.bio = cv.tree(tree.bio, FUN=prune.misclass)  
cv.bio

## $size  
## [1] 16 10 9 7 5 3 2 1  
##   
## $dev  
## [1] 99 102 95 90 100 105 181 186  
##   
## $k  
## [1] -Inf 0.0 1.0 2.0 6.5 7.5 39.0 48.0  
##   
## $method  
## [1] "misclass"  
##   
## attr(,"class")  
## [1] "prune" "tree.sequence"

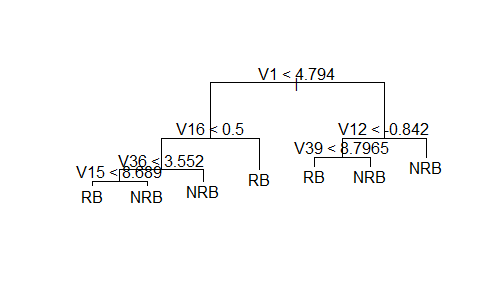
prune.bio = prune.misclass(tree.bio, best=7)  
prune.bio

## node), split, n, deviance, yval, (yprob)  
## \* denotes terminal node  
##   
## 1) root 526 678.50 NRB ( 0.65399 0.34601 )   
## 2) V1 < 4.794 232 311.60 RB ( 0.39655 0.60345 )   
## 4) V16 < 0.5 109 136.80 NRB ( 0.67890 0.32110 )   
## 8) V36 < 3.552 53 73.00 RB ( 0.45283 0.54717 )   
## 16) V15 < 8.689 19 16.57 RB ( 0.15789 0.84211 ) \*  
## 17) V15 > 8.689 34 45.23 NRB ( 0.61765 0.38235 ) \*  
## 9) V36 > 3.552 56 38.14 NRB ( 0.89286 0.10714 ) \*  
## 5) V16 > 0.5 123 102.40 RB ( 0.14634 0.85366 ) \*  
## 3) V1 > 4.794 294 241.10 NRB ( 0.85714 0.14286 )   
## 6) V12 < -0.842 50 69.23 NRB ( 0.52000 0.48000 )   
## 12) V39 < 8.7965 25 25.02 RB ( 0.20000 0.80000 ) \*  
## 13) V39 > 8.7965 25 21.98 NRB ( 0.84000 0.16000 ) \*  
## 7) V12 > -0.842 244 128.50 NRB ( 0.92623 0.07377 ) \*

Based on the result, we can find the lowest dev. In this case, the best tree size is 7.

### 2.Plot the tree. What variables are used in building the optimal trees?

plot(prune.bio)  
text(prune.bio,pretty=0)



Based on the plot, we can see that V1, V16,V12,V36,V39,V15 are used in building the optimal trees.

### 3.Apply the model to the test data and get the confusion matrix and error rate for each class and overall accuracy.

tree.y = test$V42  
tree.pred = predict(prune.bio, newdata = test,type = "class")  
table(tree.y , tree.pred)

## tree.pred  
## tree.y NRB RB  
## NRB 298 56  
## RB 58 114

error.tree.NRB = 58/(298+58)  
error.tree.RB = 56/(56+114)  
error.tree = mean (tree.y != tree.pred)  
error.tree

## [1] 0.21673

Based on the confusion matrix, we can see that error rate of class ‘NBR’ is 0.1629213 , the error rate of class ‘RB’ is 0.6705882. The overall accuracy is 0.21673.

### Random Forest and Bagging

#### 4. Using the train data, build a bagging model and get OOB error rate and apply it to the test data and get error rate. Use ntree = 500.

set.seed(1)  
library(randomForest)

## randomForest 4.6-14

## Type rfNews() to see new features/changes/bug fixes.

nvar = dim(bio.data)[2]-1  
bag.bio = randomForest(V42~.,data=train,mtry=nvar,ntree=500,importance=T)  
bag.bio

##   
## Call:  
## randomForest(formula = V42 ~ ., data = train, mtry = nvar, ntree = 500, importance = T)   
## Type of random forest: classification  
## Number of trees: 500  
## No. of variables tried at each split: 41  
##   
## OOB estimate of error rate: 13.69%  
## Confusion matrix:  
## NRB RB class.error  
## NRB 315 29 0.08430233  
## RB 43 139 0.23626374

bag.bio$err.rate[500,1] #out of bag error rate

## OOB   
## 0.1368821

#apply to test  
bag.pred = predict(bag.bio, newdata = test, type = "class")  
table(bag.pred , tree.y)

## tree.y  
## bag.pred NRB RB  
## NRB 310 40  
## RB 44 132

bag.error = mean(bag.pred != tree.y)  
bag.error

## [1] 0.1596958

Based on the result, the OOB error is 0.1368821. When applying to test data, the error rate is 0.1596958.

#### 5. Using the train data, build a random Forest Model using the best mtry and get OOB error rate. Use ntree =500. Try for mtry=sqrt(41)-1, sqrt(41), sqrt(41)+1.

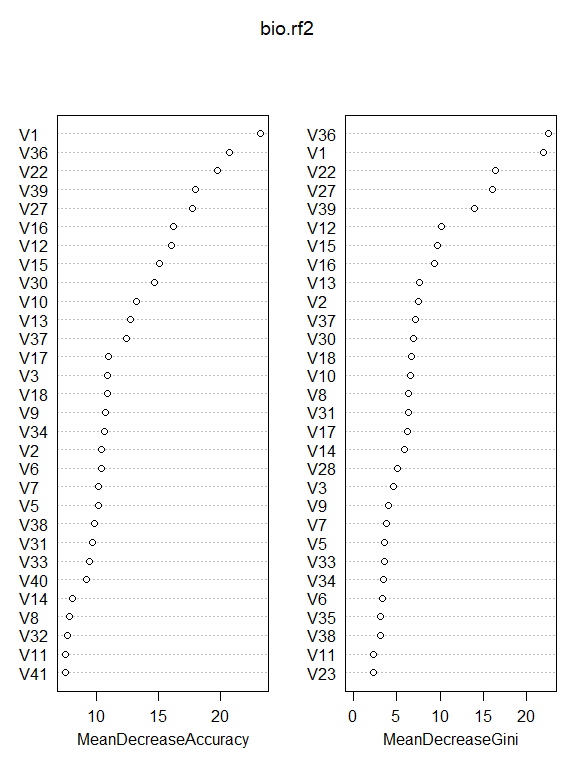
set.seed(1)  
bio.rf1 = randomForest(V42~., data=train, ntree=500,mtry=sqrt(nvar)-1,importance = TRUE)  
set.seed(1)  
bio.rf2 = randomForest(V42~., data=train, ntree=500,mtry=sqrt(nvar),importance = TRUE)  
set.seed(1)  
bio.rf3 = randomForest(V42~., data=train, ntree=500,mtry=sqrt(nvar)+1,importance = TRUE)  
bio.rf.compare = data.frame('mtry'=c('sqrt(41)-1','sqrt(41)', 'sqrt(41)+1'),'OOB error' = c(bio.rf1$err.rate[500,1],bio.rf2$err.rate[500,1],bio.rf3$err.rate[500,1]))  
bio.rf.compare

## mtry OOB.error  
## 1 sqrt(41)-1 0.1406844  
## 2 sqrt(41) 0.1368821  
## 3 sqrt(41)+1 0.1387833

Based on the result, we need to find lowest OOB error which is 0.1368821, in this case, the best model is when mtry = sqrt(41).

#### 6. Find 20 most important variables used in this random Forest.

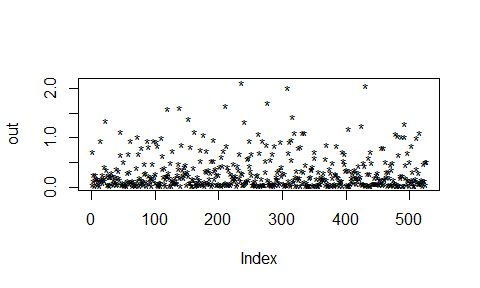
import\_var = round(importance(bio.rf2),2)  
varImpPlot(bio.rf2)



The 20 important variables are V36,V1,V22,V27,V39,V12,V15,V16,V13,V2,V37,V30,V18,V10,V8,V31,V17,V14,V28,V3.

#### 7. Find the outlying-ness score from proximity matrix and plot them. Are there any outliers?

set.seed(1)  
bio.prox = randomForest(V42~.,data=train, proximity=TRUE,oob.prox=FALSE)  
out = apply(bio.prox$proximity,1 , function(x) 1/(sum(x^2)-1))  
plot(out,pch="\*")



outlier = sort(out,decreasing = T)  
outlier[1:10]

## 214 593 7 755 146 896 547 592   
## 2.111504 2.065808 2.009679 1.713573 1.650100 1.610088 1.600297 1.425379   
## 449 715   
## 1.382055 1.341770

Based on the result ,outliers score should be bigger than 5 or 10, in this case, there is no outliers.

#### 8. Apply it to the test data and get confusion matrix and error rates for each class and overall error rate.

rf.pred = predict(bio.rf2, newdata=test, type = "class")  
table(tree.y , rf.pred)

## rf.pred  
## tree.y NRB RB  
## NRB 317 37  
## RB 39 133

error.rf.NBR = 39/(317+39)  
error.rf.RB = 37/(37+133)  
error.rf = mean(tree.y != rf.pred)  
error.rf

## [1] 0.1444867

Based on the confusion matrix, we can see that error rate of class NBR is 0.1095506, the error rate of class RB is 0.2176471. And the overall error rate is 0.1444867.

### GAM

#### 9. Based on 20 most important variables selected in Random Forest model above, build a gam model for V42 on train data.

library(gam)

## Loading required package: splines

## Loading required package: foreach

## Loaded gam 1.16

bio.gam = gam(V42 ~s(V36)+s(V1)+s(V22)+s(V27)+s(V39)+s(V12)+s(V15)+s(V16)+s(V13)+s(V2)+s(V37)+s(V30)+s(V18)+s(V10)+s(V8)+s(V31)+s(V17)+s(V14)+s(V28)+s(V3),data=train, family = "binomial")  
summary(bio.gam)

##   
## Call: gam(formula = V42 ~ s(V36) + s(V1) + s(V22) + s(V27) + s(V39) +   
## s(V12) + s(V15) + s(V16) + s(V13) + s(V2) + s(V37) + s(V30) +   
## s(V18) + s(V10) + s(V8) + s(V31) + s(V17) + s(V14) + s(V28) +   
## s(V3), family = "binomial", data = train)  
## Deviance Residuals:  
## Min 1Q Median 3Q Max   
## -2.72767 -0.32224 -0.03246 0.28695 2.70539   
##   
## (Dispersion Parameter for binomial family taken to be 1)  
##   
## Null Deviance: 678.477 on 525 degrees of freedom  
## Residual Deviance: 210.8703 on 444.9993 degrees of freedom  
## AIC: 372.8717   
##   
## Number of Local Scoring Iterations: 19   
##   
## Anova for Parametric Effects  
## Df Sum Sq Mean Sq F value Pr(>F)   
## s(V36) 1 7.654 7.654 12.6180 0.0004228 \*\*\*  
## s(V1) 1 0.898 0.898 1.4807 0.2243091   
## s(V22) 1 3.245 3.245 5.3491 0.0211879 \*   
## s(V27) 1 1.942 1.942 3.2007 0.0742887 .   
## s(V39) 1 1.832 1.832 3.0198 0.0829458 .   
## s(V12) 1 15.619 15.619 25.7476 5.726e-07 \*\*\*  
## s(V15) 1 1.346 1.346 2.2193 0.1370000   
## s(V16) 1 46.842 46.842 77.2209 < 2.2e-16 \*\*\*  
## s(V13) 1 0.240 0.240 0.3949 0.5300419   
## s(V2) 1 0.586 0.586 0.9652 0.3264107   
## s(V37) 1 10.098 10.098 16.6463 5.338e-05 \*\*\*  
## s(V30) 1 4.662 4.662 7.6858 0.0057992 \*\*   
## s(V18) 1 18.004 18.004 29.6799 8.451e-08 \*\*\*  
## s(V10) 1 8.688 8.688 14.3221 0.0001751 \*\*\*  
## s(V8) 1 6.557 6.557 10.8095 0.0010899 \*\*   
## s(V31) 1 10.749 10.749 17.7194 3.099e-05 \*\*\*  
## s(V17) 1 7.147 7.147 11.7828 0.0006537 \*\*\*  
## s(V14) 1 0.317 0.317 0.5233 0.4698378   
## s(V28) 1 6.109 6.109 10.0710 0.0016106 \*\*   
## s(V3) 1 2.428 2.428 4.0030 0.0460251 \*   
## Residuals 445 269.938 0.607   
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## Anova for Nonparametric Effects  
## Npar Df Npar Chisq P(Chi)   
## (Intercept)   
## s(V36) 3 6.5485 0.087776 .   
## s(V1) 3 1.7299 0.630313   
## s(V22) 3 10.3146 0.016076 \*   
## s(V27) 3 5.0121 0.170917   
## s(V39) 3 2.5080 0.473831   
## s(V12) 3 13.5307 0.003619 \*\*   
## s(V15) 3 2.5894 0.459353   
## s(V16) 3 11.0612 0.011400 \*   
## s(V13) 3 4.8702 0.181553   
## s(V2) 3 1.5494 0.670931   
## s(V37) 3 24.2970 2.166e-05 \*\*\*  
## s(V30) 3 10.9420 0.012042 \*   
## s(V18) 3 2.8164 0.420804   
## s(V10) 3 4.5386 0.208864   
## s(V8) 3 0.6360 0.888133   
## s(V31) 3 1.1332 0.769074   
## s(V17) 3 1.9987 0.572676   
## s(V14) 3 5.2033 0.157519   
## s(V28) 3 6.5927 0.086072 .   
## s(V3) 3 0.2983 0.960339   
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1

#### 10. What variables have non-linear effect?

Based on the Anova for nonparametric effects, if the p value is smaller than 0.05, we can say that it has non-linear effect. In this case, V22, V12, V16, V37,V30 have non-linear effect.

#### 11. Apply the model to the test data and get the confusion matrix. Find overall error rates and error rates for individual class (spam and email class).

prob.bio= predict(bio.gam,newdata = test, type="response")  
le = levels(tree.y)  
pred.bio = ifelse(prob.bio >= 0.5, le[2],le[1])  
table(tree.y,pred.bio)

## pred.bio  
## tree.y NRB RB  
## NRB 304 50  
## RB 36 136

gam.error.NBR = 36/(36+304)  
gam.error.RB = 50/(50+136)  
gam.error = mean(tree.y != pred.bio)  
gam.error

## [1] 0.1634981

Based on the confusion matrix, we can see that error rate of class NBR is 0.1058824, the error rate of class RB is 0.2688172. And the overall error rate is 0.1634981.

### SVM

#### 12. Build a svm model to classify V42 using V1 through V41 on train data. Use “radial” kernel function and find the best gamma and cost first. What are the best gamma and cost? What’s error rate for the optimal gamma and cost? Try among the cost=c(0.001, 0.01, 0.1, 1, 10,100) and gamma=c(.01,.1,1,5,10)

library(e1071)  
set.seed(1)  
svm.bio = tune(svm,V42~.,data=train,kernel="radial",  
 ranges=list(cost=c(0.001, 0.01, 0.1, 1, 10,100),  
 gamma=c(.01,.1,1,5,10)))  
svm.best = svm.bio$best.model  
svm.bio$best.parameters

## cost gamma  
## 5 10 0.01

svm.bio$best.performance

## [1] 0.1234761

Based on the result, the best cost is 1 and the best gamma is 0.1 , the error rate of optimal cost and gamma is 0.1234761.

#### 13. Apply the model to the test data and get the confusion matrix and error rates.

svm.pred = predict(svm.best,newdata = test)  
table(tree.y, svm.pred)

## svm.pred  
## tree.y NRB RB  
## NRB 318 36  
## RB 33 139

svm.error = mean(tree.y != svm.pred)  
svm.error

## [1] 0.1311787

#### 14.Compare error rates of the best tree, bagging, random Forest, gam, and svm for the predicted values on test data above. Which model do you prefer?

compare.model = data.frame('model' = c('best tree','bagging','random forest','gam','svm'), 'test error' =c(error.tree, bag.error,error.rf,gam.error,svm.error))  
compare.model

## model test.error  
## 1 best tree 0.2167300  
## 2 bagging 0.1596958  
## 3 random forest 0.1444867  
## 4 gam 0.1634981  
## 5 svm 0.1311787

Based on the result, we can see that SVM has the lowest test error which is 0.1311787.

### Clustering

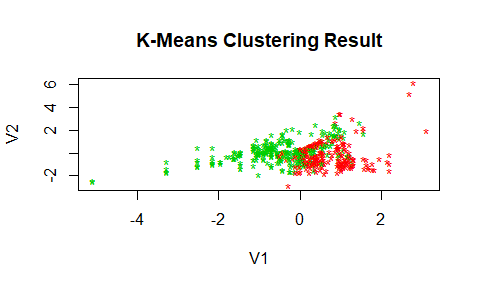
#### 15.For the training data eliminate factor variables V24,V25, V29 and V42 and scale the data. Find the optimal number of clusters using average silhouette length test. (Use manhattan distance and method = “complete” for scaled data.) It may take around 7-10 minutes so run it using separate Rmd file or run it using R script file and add the result in word file.

cluster.data = train[,-c(24,25,29,42)]  
cluster.data = scale(cluster.data)  
  
#Based on the result, the best cluster is 2.

#### 

#### 16.Using the above result, find kmeans cluster with the optimal number of clusters above.

set.seed(1)  
km.bio = kmeans(cluster.data,2,nstart = 40 )  
plot(cluster.data, col=(km.bio$cluster +1),main="K-Means Clustering Result",pch="\*")



#### 17.Using table function, explain how the clusters corresponds to the V42. Do you think clustering can be used in classifying V42? Compare it to the above classification methods.

cluster.y = train$V42  
table(cluster.y,km.bio$cluster)

##   
## cluster.y 1 2  
## NRB 194 150  
## RB 30 152

(30+150)/(194+150+30+152)

## [1] 0.3422053

In the cluster 1, there has 194 belong to NRB class, and 30 belongs to RB class, Based on majority vote, we can see that class 1 represents NRB. In cluster 2, we have 150 are NBR class, and 152 are RB class. If we say cluster 1 are NBR but we can’t say that cluster 2 represents RB class, becasue the data is too close. if we want to calculate error rate, we can assume that culster1 represents NRB class, and cluster represents RB class, then the error rate will be (30+150)/(194+150+30+152) = 0.3422053, which is pretty high. Much higher than above classfication methods. I think we can’t use clustering in classifying v42, becasue clustering is unsurpervised learning, which means that response varibales are unknown. What’s more, NbClust function couldn’t define the exact number of clusters.