# ESA Review

目录

[ESA Review 1](#_Toc90622986)

[Week 8-9 1](#_Toc90622987)

[Lex analytics: 1](#_Toc90622988)

[Bagging and random forest 4](#_Toc90622989)

[CvsC(ggplot) 7](#_Toc90622990)

[Week 10 Text Analysis 7](#_Toc90622991)

[Week 11 Recommendation System 11](#_Toc90622992)

[Clustering\_Review 15](#_Toc90622993)

[Water 16](#_Toc90622994)

[Practise Question: 17](#_Toc90622995)

[Exercise 5 17](#_Toc90622996)

[Exercise 6 22](#_Toc90622997)

[Exercise 7 25](#_Toc90622998)

## Week 8-9

### Lex analytics:

#### # read csv

stevens$rev<-as.integer((stevens$lctdir=="conser"&stevens$stevdir==0)|(stevens$lctdir=="liberal"&stevens$stevdir==1))

table(stevens$rev)

#### #train&test split

if(!require(caTools)){ # install or load the package

install.packages("caTools")

library(caTools)

}

set.seed(123) # set seed for random sampling

spl <- sample.split(stevens$rev,SplitRatio=0.7) # We use 70% of the data for training

train <- subset(stevens,spl==TRUE); # training dataset

test <- subset(stevens,spl==FALSE); # testing dataset

#### #logistic regression and predict

m1 <- glm(rev~petit+respon+circuit+unconst+issue+lctdir,data=train,family="binomial")

summary(m1)

test <- subset(test,test$issue!="IR")

p1 <- predict(m1,newdata=test,type="response")

table(p1>=0.5,test$rev)

#### #ROC curve

if(!require(ROCR)){

install.packages("ROCR")

library(ROCR)

}

pred <- prediction(p1,test$rev)

perf <- performance(pred, x.measure="fpr", measure="tpr")

# performance(pred,measure="auc")

plot(perf)

#### #cart

cart1 <- rpart(rev~petit+respon+circuit+lctdir+issue+unconst,data=train,method="class")

print(cart1)

summary(cart1)

#visuals

if(!require(rpart.plot)){

install.packages("rpart.plot")

library(rpart.plot)

}

# tree structure

prp(cart1)

prp(cart1,type=1) # labels all nodes (not just the leaves)

prp(cart1,type=4) # draws separate labels for left and right directions for all nodes and label nodes

prp(cart1,type=4,extra=4) # in addition, this also plots the probability per class of observation

# prp(cart1,type=4,extra=9) # probablities times fraction of observations at the node (the sum across all leaves is 1)

#fancyplot

if(!require(rattle)){

install.packages("rattle")

library(rattle)

}

if(!require(RColorBrewer)){

install.packages("RColorBrewer")

library(RColorBrewer)

}

fancyRpartPlot(cart1) # Run fancyRpartPlot

predictcart1 <- predict(cart1,newdata=test,type="class")

table(test$rev,predictcart1)

#roc curve

predictcart1\_prob <- predict(cart1,newdata=test) # predicted probabilities of rev=0 and rev=1 for each test observation

pred\_cart1 <- prediction(predictcart1\_prob[,2],test$rev)

perf\_cart1 <- performance(pred\_cart1, x.measure="fpr",measure="tpr")

plot(perf\_cart1)

# performance(pred\_cart1,measure="auc")

# prune

printcp(cart1)

opt <- which.min(cart1$cptable[,"xerror"]) # get index of CP with lowest xerror

cp <- cart1$cptable[opt, "CP"] # get the corresponding value

cart2 <- prune(cart1,cp)

prp(cart2,type=4,extra=4)

### Bagging and random forest

#import data

supreme <- read.csv("supreme.csv") # We have 623 observations and 20 variables

# str(supreme) # Internal structure of the dataframe

# head(supreme) # First part of the dataframe

# Prepare the output variable (justice Stevens' decision)

stevens <- subset(supreme[,c("docket","term","stevdir","petit","respon","circuit","unconst","lctdir","issue","result")],supreme$stevdir!=9)

stevens$rev <- as.integer((stevens$lctdir=="conser" & stevens$stevdir==0) | (stevens$lctdir=="liberal" & stevens$stevdir==1))

## Splitting the dataset

To split the data into two groups (for training and testing), we use the package `caTools`.

```{r, message=FALSE}

if(!require(caTools)){ # install or load the package

install.packages("caTools")

library(caTools)

}

set.seed(123) # set seed for random sampling

spl <- sample.split(stevens$rev,SplitRatio=0.7) # We use 70% of the data for training

train <- subset(stevens,spl==TRUE); # training dataset

test <- subset(stevens,spl==FALSE); # testing dataset

test <- subset(test,test$issue!="IR") # remove the only realization of the IR value (please refer to Week 8's material)

#### # Bagging

## R package

if(!require(ipred)){

install.packages("ipred")

library(ipred)

}

## Building bagged trees

```{r}

mt <- bagging (as.factor(rev)~petit+respon+circuit+lctdir+issue+unconst,data=train, coob=TRUE)

```

```{r}

# summary(mt) # long summary

print(mt) # short summary

## Testing bagged trees

How well does this bagged model perform on the test set?

```{r}

predict\_bag\_model <- predict(mt,newdata=test,type="class")

table(test$rev,predict\_bag\_model)

#### # Random Forests

## Training and testing

library(randomForest)

forest <- randomForest(as.factor(rev)~petit+respon+circuit+unconst+lctdir+issue,data=train)

forest

predictforest <- predict(forest,newdata=test,type="class")

table(test$rev,predictforest)

# important factors

importance(forest) # tabulated results

varImpPlot(forest) # plot

varUsed(forest, by.tree=FALSE, count=TRUE)

#### Find out which predictor variables are actually used in the random forest.

```{r}

# ?varUsed

# varUsed(model2)

order2 <- sort(varUsed(model2), decreasing=TRUE, index.return=TRUE) # Return sorted frequency and indices

# head(order2$ix)

names(test[,head(order2$ix)]) # we can see which words are important!

MeanDecreaseGini mean This is a measure of how important a variable is for estimating the value of the target variable across all of the trees that make up the forest. A higher Mean Decrease in Gini indicates higher variable importance.

#find best ntreee

# Values of ntree to be tested

B <- seq(5,500,by=5)

OOB <- vector(); OOB <- c(OOB, 1:length(B))for (i in 1:length(B)){

forest\_temp <- randomForest(as.factor(rev)~petit+respon+circuit+unconst+lctdir+issue,data=train,ntree=B[i])

OOB[i] <- forest\_temp$err.rate[B[i],1] rm(forest\_temp)

}

library(ggplot2)

ggplot(data = data.frame(ntree=B, error=OOB),

mapping = aes(x = ntree, y = error)) +

geom\_point(size=3) +

coord\_cartesian(xlim=c(0,500), ylim=c(0.325,0.450)) +

labs(x="Number of Trees", y="OOB error")

Now, we repeat the experiment, but changing both `mtry` and `ntree` at the same time.

# Values of ntree to be tested

B <- seq(5,500,by=5)

# Values of mtry to be tested

m <- seq(1,6,by=1)

# Initialize a matrix for the OOB value

OOB\_matrix <- matrix(0,nrow=length(B),ncol=length(m))

# For loop

for (i in 1:length(B)){

for (j in 1:length(m)){

# train a forest with B[i] trees

forest\_temp <- randomForest(as.factor(rev)~petit+respon+circuit+unconst+lctdir+issue,data=train,ntree=B[i],mtry=m[j])

# model performance

OOB\_matrix[i,j] <- forest\_temp$err.rate[B[i],1]

# remove the temporary variable

rm(forest\_temp)

}

}

#visuals

library(reshape2)

rownames(OOB\_matrix) <- B

colnames(OOB\_matrix) <- m

# longData <- as.data.frame(OOB\_matrix)

# row.names(longData) <- B; colnames(longData) <- m

longData <- melt(OOB\_matrix)

# ...

ggplot(longData, aes(x = Var1, y = Var2)) +

geom\_raster(aes(fill=value)) +

scale\_fill\_gradient(low="grey90", high="red") +

labs(x="Number of Trees", y="Number of predictors")

which.min(OOB\_matrix[,2])

best\_forest <- randomForest(as.factor(rev)~petit+respon+circuit+unconst+lctdir+issue,data=train,mtry=2,ntree=95)

# best\_forest

predict\_bestforest <- predict(best\_forest,newdata=test,type="class")

table(test$rev,predict\_bestforest)

### CvsC(ggplot)

## Week 10 Text Analysis

#### #read data

twitter <- read.csv("twitter.csv",stringsAsFactors=FALSE)

twitter$Neg <- as.factor(twitter$sentiment<=2)

table(twitter$Neg)

library(tm)

#### #preprocessing

corpus <- tm\_map(corpus, function(x) iconv(enc2utf8(x), sub = "byte"))

corpus <- tm\_map(corpus, content\_transformer(function(x) iconv(enc2utf8(x), sub = "bytes")))

corpus <- tm\_map(corpus, content\_transformer(tolower))

corpus <- Corpus(VectorSource(twitter$tweet))

corpus <- tm\_map(corpus,content\_transformer(tolower))

corpus <- tm\_map(corpus,removeWords,stopwords("english"))

corpus <- tm\_map(corpus,removePunctuation)

corpus <- tm\_map(corpus,removeNumbers)

corpus <- tm\_map(corpus,stemDocument)

#remove hot words

corpus <- tm\_map(corpus,removeWords,c("drive","driving","driver","self-driving","car","cars"))

#### #Snowballc

library(SnowballC)

corpus <- tm\_map(corpus,stemDocument)

dtm <- DocumentTermMatrix(corpus)

dtm

dtm[1,]

inspect(dtm[1,])

findFreqTerms(dtm,lowfreq=50)

dtm <- removeSparseTerms(dtm,0.995)

twittersparse <- as.data.frame(as.matrix(dtm))

colnames(twittersparse) <- make.names(colnames(twittersparse))

#### #wordcloud

install.packages("wordcloud")

# Get word counts in decreasing order

word\_freqs = sort(colSums(twittersparse), decreasing=TRUE)

# Create data frame with words and their frequencies

dm = data.frame(word=names(word\_freqs), freq=unname(word\_freqs))

# Plot wordcloud

wordcloud(dm$word, dm$freq, random.order=FALSE, colors=brewer.pal(8, "Dark2"),max.word=100)

twittersparse$Neg <- twitter$Neg

#### # Train and test a classifier

Let's prepare the data for our modelling exercise:

```{r}

# Load the caTools package and set the seed

if(!require(caTools)){

install.packages("caTools")

library(caTools)

}

set.seed(123)

# Create train and test sets (with balanced response)

spl <- sample.split(twittersparse$Neg,SplitRatio=0.7)

train <- subset(twittersparse,spl==TRUE)

test <- subset(twittersparse,spl==FALSE)

# Random Forests

Train:

```{r}

library(randomForest)

# We use the default parameters to fit the model (500 trees)

model3 <- randomForest(Neg~.,data=train)

# summary(model3)

model3

```

Prediction

```{r}

predict3 <- predict(model3,newdata=test,type="class")

table(predict3,test$Neg)

#### # strwrap

strwrap(energy$email[1])

energy$responsive[1] # This takes value 0 since the email is not responsive to energy bid and schedule.

#### # Naive Bayes Classifier

The function `naiveBayes` (package `e1071`) computes the conditional a posteriori probabilities of a categorical class variable given independent predictor variables using the Bayes rule.

if(!require(e1071)){

install.packages("e1071")

library(e1071)

}

model3 <- naiveBayes(as.factor(responsive)~.,data=train)

# summary(model3)

#Conditional probabilities:

model3$apriori

#Conditional probabilities for each predictor. For each predictor, we get target class, mean #and standard deviation. Let's look, for example, at the predictor `california` or `market`.

model3$tables$california

model3$tables$market

```

```{r}

predict3 <- predict(model3,newdata=test,type="class")

table(predict3,test$responsive)

```

## Week 11 Recommendation System

#### #read data with(sep) function

movies <- read.csv("movies.csv",stringsAsFactors=FALSE)

str(movies)

# Count the number of fields separated by "|" in each row of genres.csv

countfields <- count.fields("genres.csv", sep="|")

# The first movie, for example has 5 genres listed

countfields[1]

# With this command, we can see that movies have from 1 to 7 genres listed

min(countfields)

max(countfields)

# Now, we load genres.csv. The command creates a dataframe with 8569 observations and 7 variables with column names X1 ... X7.

genres <- read.csv("genres.csv", header=FALSE, sep="|", col.names=c("X1","X2","X3","X4","X5","X6","X7"), stringsAsFactors=TRUE)

# Note that each variable has different number of levels

str(genres)

#### #preprocessing

source('process\_genre\_data.R')

M <- process\_genre\_data(genres)

Data <- as.data.frame(M)

Data$title <- movies$title

Data <- Data[,-19] # Drops the 19th column, which corresponds to the "" category

#### # Hierarchical clustering

distances <- dist(Data[,1:19], method="euclidean")

dim(Data)

#ward distance method

clusterMovies1 <- hclust(distances, method="ward.D2")

clusterGroups1 <- cutree(clusterMovies1, k=10)

tapply(Data[,1], clusterGroups1, mean)

Cat1 <- matrix(0,nrow=19,ncol=10)

for(i in 1:19)

{Cat1[i,] <- tapply(Data[,i], clusterGroups1, mean)}

rownames(Cat1) <- colnames(Data)[1:19]

subset(Data, movies$title=="X-Men: First Class (2011)")

clusterGroups1[7849]

#### #K-mean clustering

set.seed(1)

clusterMovies2 <- kmeans(Data[,1:19],centers=10,nstart=20)

clusterMovies2$tot.withinss

#We wanna withinss to be small!

set.seed(1)

clusterMovies3 <- kmeans(Data[,1:19],centers=10,nstart=1)

clusterMovies3$tot.withinss

set.seed(1)

fit <- 0

for(k in 1:15)

{clusterMovies4 <- kmeans(Data[,1:19], centers=k, nstart=20)

fit[k] <- clusterMovies4$tot.withinss}

plot(1:15,fit)

Cat2 <- matrix(0,nrow=19,ncol=10)

for(i in 1:19)

{Cat2[i,] <- tapply(Data[,i], clusterMovies2$cluster, mean)}

rownames(Cat2) <- colnames(Data)[1:19]

#### Part2

#### # preanalysis(unique sort)

length(unique(ratings$userId))

length(unique(ratings$movieId))

sort(unique(ratings$rating))

max(table(ratings$userId))

which(table(ratings$userId)==2268)

max(table(ratings$movieId))

#### # Data preparation

Data <- matrix(nrow=length(unique(ratings$userId)), ncol=length(unique(ratings$movieId)))

rownames(Data) <- unique(ratings$userId)

colnames(Data) <- unique(ratings$movieId)

for(i in 1:nrow(ratings))

{Data[as.character(ratings$userId[i]),as.character(ratings$movieId[i])] <- ratings$rating[i]}

# dim(Data)

#### #nomalize the data

hist(as.vector(Data))

Datanorm <- Data - rowMeans(Data,na.rm=TRUE)

hist(as.vector(Datanorm))

#### Train test split

set.seed(1)

spl1 <- sample(1:nrow(Data), 0.98\*nrow(Data)) # spl1 has 98% of the rows

spl1c <- setdiff(1:nrow(Data),spl1) # spl1c has the remaining ones

set.seed(2)

spl2 <- sample(1:ncol(Data), 0.8\*ncol(Data)) # spl2 has 80% of the columns

spl2c <- setdiff(1:ncol(Data),spl2)

length(spl1)

length(spl1c)

length(spl2)

length(spl2c)

#### #model

Base1 <- matrix(nrow=length(spl1c), ncol=length(spl2c))

Base2 <- matrix(nrow=length(spl1c), ncol=length(spl2c))

UserPred <- matrix(nrow=length(spl1c), ncol=length(spl2c))

#base line model1

for(i in 1:length(spl1c))

{Base1[i,] <- colMeans(Data[spl1,spl2c], na.rm=TRUE)}

Base1[,1:10]

#base line model2

for(j in 1:length(spl2c))

{Base2[,j] <- rowMeans(Data[spl1c,spl2], na.rm=TRUE)}

Base2[,1:10]

#### # User-based model

Cor <- matrix(nrow=length(spl1),ncol=1) # keeps track of the correlation between users

Order <- matrix(nrow=length(spl1c), ncol=length(spl1)) # sort users in term of decreasing correlations

for(i in 1:length(spl1c)){

for(j in 1:length(spl1)){

Cor[j] <- cor(Data[spl1c[i],spl2],Data[spl1[j],spl2],use = "pairwise.complete.obs")

}

V <- order(Cor, decreasing=TRUE, na.last=NA)

Order[i,] <- c(V, rep(NA, times=length(spl1)-length(V)))

}

length(Cor)

dim(Order)

#### #prediction

for(i in 1:length(spl1c))

{UserPred[i,] <- colMeans(Data[spl1[Order[i,1:250]],spl2c], na.rm=TRUE)}

UserPred[,1:10]

RMSEBase1 <- sqrt(mean((Data[spl1c,spl2c] - Base1)^2, na.rm=TRUE))

RMSEBase2 <- sqrt(mean((Data[spl1c,spl2c] - Base2)^2, na.rm= TRUE))

RMSEUserPred <- sqrt(mean((Data[spl1c,spl2c] - UserPred)^2,na.rm=TRUE))

RMSE <- rep(NA, times=490)

for(k in 10:499)

{for(i in 1:length(spl1c))

{UserPred[i,] <- colMeans(Data[spl1[Order[i,1:k]],spl2c], na.rm=TRUE)}

RMSE[k-10] <- sqrt(mean((Data[spl1c,spl2c] - UserPred)^2,na.rm=TRUE))}

plot(10:499,RMSE)

## Clustering\_Review

#### #scale the data(the data is imbalanced)

pollution\_scaled <- data.frame(scale(pollution))

# head(pollution)

# head(pollution\_scaled)

colnames(pollution\_scaled) <- names(pollution)

distances\_scaled <- dist(pollution\_scaled, method="euclidean")

clusterpollution2 <- hclust(distances\_scaled, method="ward.D2")

plot(clusterpollution2)

# cut dendogram into k=4 clusters

clusterGroups2 <- cutree(clusterpollution2, k=4)

plot(pollution\_scaled$DENSReal,pollution\_scaled$MORTReal, col=clusterGroups2, pch=19)

#### # K-means clustering

set.seed(1)

clusterpollution3 <- kmeans(pollution\_scaled,centers=4,nstart=20)

clusterpollution3$tot.withinss # smaller the better

#### #adjust K

set.seed(1)

fit <- 0 # to store the total euclidean dist sum per cluster

for(k in 1:15) {

clusterpollution4 <- kmeans(pollution\_scaled, centers=k, nstart=20)

fit[k] <- clusterpollution4$tot.withinss

}

plot(1:15,fit)

Centroid2 <- matrix(0,nrow=16,ncol=4) # to store the centroid data

for(i in 1:16) {

Centroid2[i,] <- tapply(pollution\_scaled[,i], clusterpollution3$cluster, mean)

}

rownames(Centroid2) <- colnames(pollution\_scaled)[1:16]

Centroid2

#### #Comparison(check original code)

#### #PCA

pr.out<-prcomp(pollution\_scaled,scale=F) # note: we are already using scaled data

#summary(pr.out)

pve<-pr.out$sdev^2/sum(pr.out$sdev^2)

cpve<-cumsum(pve)

plot(cpve,xlab="Principal components",type="l",ylim=c(0,1))

# first 5 principal components explain almost 80% of variance in data

library(factoextra)

fviz\_pca\_biplot(pr.out, label = "var", habillage=clusterpollution3$cluster,addEllipses=TRUE, ellipse.level=0.95)

## Water

water2<-subset(water[,c("LEVEL\_T7","FLOW\_PU6","LEVEL\_T6","PRESS\_J289","PRESS\_J300","FLOW\_PU7", "FLOW\_PU4","LEVEL\_T1","PRESS\_J422","FLOW\_V2","LEVEL\_T2","PRESS\_J280","PRESS\_J256","PRESS\_J415","LEVEL\_T4","PRESS\_J317","Anomaly")])

# Train+Test set generation

set.seed(1234) # set seed for random sampling

spl2 <- sample.split(water2$Anomaly,SplitRatio=0.7) # We use 70% of the data for training

train2 <- subset(water2,spl2==TRUE); # training dataset

test2 <- subset(water2,spl2==FALSE); # testing dataset

# GLM2 + Prediction accuracy on test set with threshold 0.5

glm\_2 <- glm(Anomaly ~ ., family = "binomial", data = train2)

pred\_1\_b <- predict(glm\_2, newdata = test2, type = "response")

table\_1\_b <- table(test2$Anomaly, pred\_1\_b >= 0.5)

sum(diag(table\_1\_b))/sum(table\_1\_b)

# Baseline Accuracy

base\_1 <- names(table(train2$Anomaly)[which.max(table(train2$Anomaly))])

unname(table(test2$Anomaly)[base\_1]/sum(nrow(test2)))

## Practise Question:

### Exercise 5

#### Significant level:

The follow code will print out all predictors significant at the 10% significance level.

names(which(coef(summary(glm\_1))[,4] < 0.1))

#### Baseline accuracy

base\_1 <- names(table(train\_1$over50k)[which.max(table(train\_1$over50k))]) unname(table(test\_1$over50k)[base\_1]/sum(nrow(test\_1)))

#### AUC( the area of ROC curve)

library(ROCR) rocr\_1\_d <- prediction(pred\_1\_b, test\_1$over50k) performance(rocr\_1\_d, "auc")@y.values

#### Classification tree and number of trees node (cp table)

unname(tail(tree\_1\_e$cptable[,2], 1))

#### Tree variable at different level

tree\_1\_e$frame["1", 1]

tree\_1\_e$frame["2", 1]

tree\_1\_e$frame["3", 1]

#### prediction for CART model

pred\_1\_j <- predict(tree\_1\_e, newdata = test\_1, type = "prob")

rocr\_1\_j <- prediction(pred\_1\_j[,2], test\_1$over50k)

perf\_1\_j\_2 <- performance(rocr\_1\_j, measure = "tpr", x.measure = "fpr")

plot(perf\_1\_j\_2)

#auc value

performance(rocr\_1\_j, measure = "auc")@y.values

#### RF

et.seed(1)

trainSmall <- train\_1[sample(nrow(train\_1), 2000),]

library(randomForest)

set.seed(1)

rf\_1 <- randomForest(over50k ~ ., data = trainSmall)

pred\_1\_l <- predict(rf\_1, newdata = test\_1)

table\_1\_l <- table(test\_1$over50k, pred\_1\_l)

sum(diag(table\_1\_l))/sum(table\_1\_l)

#### Important variable(\*)

vu <- varUsed(rf\_1, count = TRUE)

vusorted <- sort(vu, decreasing = FALSE, index.return = TRUE)

dotchart(vusorted$x, names(rf\_1$forest$xlevel[vusorted$ix]))

tail(names(rf\_1$forest$xlevel[vusorted$ix]), 1)

varImpPlot(rf\_1)

rownames(rf\_1$importance)[which.max(rf\_1$importance)]

#### Prune CART with cp value

tree\_1\_o\_1 <- rpart(over50k ~., data = train\_1, cp = 0.0001)

printcp(tree\_1\_o\_1) # uncomment but output is fairly big and uninformative

plotcp(tree\_1\_o\_1)

We choose the one with the smallest xerror

tree\_1\_o\_2 <- prune(tree\_1\_o\_1, cp = 0.00422352)

prp(tree\_1\_o\_2)

#### MSE(mean square error)

pred\_2 <- predict(tree\_2\_a, newdata = test\_2)

mse\_2\_b <- mean((pred\_2 - test\_2$medv)^2)

mse\_2\_b

#### Why not prune the tree?

We could prune the tree as the number of splits which result in lowest xerror is given by a larger cp parameter. However, the differences are minimal, although a smaller tree is preferable to a larger tree. We decide to not prune the tree.

#### prune the tree to 5 nodes.

5 nodes implies 4 splits. We hence need to get the corresponding `cp` value:

cp\_val\_2 <- tree\_2\_a$cptable[tree\_2\_a$cptable[,2] == 4, 1]

tree\_2\_d <- prune(tree\_2\_a, cp = cp\_val\_2)

pred\_2\_d <- predict(tree\_2\_d, newdata = test\_2)

mse\_2\_d <- mean((pred\_2\_d - test\_2$medv)^2)

mse\_2\_d

#### important variable

import\_2 <- as.vector(importance(rf\_2))

names(import\_2) <- rownames(importance((rf\_2)))

names(sort(import\_2, decreasing = TRUE)[1:2])

varImpPlot(rf\_2)

The two most important variables are ``r names(sort(import\_2, decreasing = TRUE)[1])`` and ``r names(sort(import\_2, decreasing = TRUE)[2])``.

#### Define a new variable

图形用户界面, 文本, 应用程序, 电子邮件

描述已自动生成

#### classification tree

#### Difference between Rpart(as.factor(A)~ and Rpart(A~)

#### Tree variables(list all)

spl\_vars\_3 <- as.character(unique(tree\_3\_d$frame[,1]))

spl\_vars\_3 <- spl\_vars\_3[!spl\_vars\_3 %in% "<leaf>"]

#### Leaf node with fewest number

Search “ree\_3\_g$frame[which.min(tree\_3\_g$frame[,2]),] “

#### Compare 2 tables

pred\_3\_i1 <- predict(tree\_3\_d, newdata = supreme, type = "class")

pred\_3\_i2 <- predict(tree\_3\_g, newdata = supreme, type = "class")

table(pred\_3\_i1, pred\_3\_i2) ["1", "1"]

table(pred\_3\_i1, pred\_3\_i2) [“0”,”0”]

#### multiple classification trees:

judges\_3 <- colnames(supreme)[5:13]

df\_3\_k <- subset(supreme, pred\_3\_i1 == pred\_3\_i2)

# conservative decisiosn

total\_cons\_3 <- integer(nrow(df\_3\_k)) # initialise as vector of 0s

for (judge in judges\_3) {

formula\_i <- as.formula(paste0("as.factor(", judge, ")",

"~ petit + respon + ",

"circuit + unconst + ",

"lctdir+issue"))

tree\_i <- rpart(formula\_i, data = df\_3\_k)

pred\_i <- predict(tree\_i, newdata = df\_3\_k,

type = "class")

total\_cons\_3 <- total\_cons\_3 + as.numeric(levels(pred\_i))[as.integer(pred\_i)]

}

table\_3\_k <- table(total\_cons\_3 >= 5, df\_3\_k$result)

sum(diag(table\_3\_k))/nrow(df\_3\_k)

#### Overall accuracy for 2 models:

df\_3\_l <- subset(supreme, pred\_3\_i1 != pred\_3\_i2)

pred\_3\_l <- predict(tree\_3\_d, newdata = df\_3\_l, type = "class")

table\_3\_l <- table(pred\_3\_l, df\_3\_l$result)

sum(diag(table\_3\_l))

#### Predict on an actual example

data\_3\_m <- data.frame(petit = "BUSINESS", respon = "BUSINESS",

circuit = "6th", unconst = 0,

lctdir = "conser", issue = "ECN")

predict(tree\_3\_d, newdata = data\_3\_m, type = "class")

#### CART is more interpretable than the random forest, but the latter has a greater accuracy.

### Exercise 6

#### Find the max length

max(nchar(emails$text))

emails$text[which.min(nchar(emails$text))]

#### Normal preprocessing:

library(tm)

corpus <- Corpus(VectorSource(emails$text))

corpus <- tm\_map(corpus, content\_transformer(tolower))

corpus <- tm\_map(corpus, removePunctuation)

corpus <- tm\_map(corpus, removeWords, stopwords("english"))

corpus <- tm\_map(corpus, stemDocument)

dtm <- DocumentTermMatrix(corpus)

spdtm <- removeSparseTerms(dtm, 0.95)

ncol(spdtm)

emailsSparse <- as.data.frame(as.matrix(spdtm))

colnames(emailsSparse) <- make.names(colnames(emailsSparse))

names(which.max(colSums(emailsSparse)))

#### Word stem frequency >5000

emailsSparse$spam <- emails$spam

names(which(colSums(subset(emailsSparse, emailsSparse$spam == 0)) >= 5000))

#### Find siginificant variable with p-value

names(which(coef(summary(spamLog))[,4] < 0.05))

#### Variable relation and occurance in CART tree(new)

intersect(spamCART$frame[,1], c("enron", "hou", "vinc", "kaminski"))

#### random

#### \*\*\*Function for accuracy and auc useful when we compare 3 different models

accuracy <- function(predict\_object, data, threshold=0.5) {

return(sum(diag(table(predict\_object >= threshold, data))) /

sum(table(predict\_object >= threshold, data)))

}

auc <- function(predict\_object, data) {

prediction\_obj <- prediction(predict\_object, data)

perf\_obj <- performance(prediction\_obj, measure = "auc")

return(perf\_obj@y.values[[1]])

}

accuracy(predictRF, train$spam)

#test accuracy and auc values

predLogtest <- predict(spamLog, newdata = test, type = "response")

predCARTtest <- predict(spamCART, newdata = test)

predRFtest <- predict(spamRF, newdata = test, type = "prob")

predCARTtest <- predCARTtest[,2]

predRFtest <- predRFtest[,2]

method\_str <- sprintf("%-20s" ,

c("Logistic Regression",

"CART",

"Random Forest")

)

big\_test\_obj <- cbind(predLogtest, predCARTtest, predRFtest)

for (pred\_idx in 1:3) {

if (pred\_idx %in% c(2,3)) { # not logistic regression

print(paste0(method\_str[pred\_idx],

sprintf("%-12s", " accuracy"),

accuracy(big\_test\_obj[,pred\_idx], test$spam)))

}

print(paste0(method\_str[pred\_idx],

sprintf("%-12s", " AUC"),

auc(big\_test\_obj[,pred\_idx], test$spam)))

}

文本

描述已自动生成

#### AUC and accuracy is better if closer to 1!

### Exercise 7

#### Propotion of positive number

(table(stocks$PositiveDec)/nrow(stocks))["1"]

#### Find max correlation

max(cor(stocks[,1:11]) - diag(rep(1, 11)))

#### Find smallest

names(sort(colMeans(stocks[,1:11]), decreasing = FALSE)[1]) # smallest

#### Baseline

most\_common <- names(sort(table(stocksTrain$PositiveDec),

decreasing = TRUE)[1])

# baseline model predicts 1 always

table(stocksTest$PositiveDec)[most\_common]/nrow(stocksTest)

#### Kmeans

set.seed(144)

km <- kmeans(normTrain, centers = 3)

names(sort(table(km$cluster), decreasing = TRUE)[1])

#### flexcluster

set.seed(144)

km <- kmeans(normTrain, centers = 3)

names(sort(table(km$cluster), decreasing = TRUE)[1])

#### Different sign of variable

StocksModel1 <- glm(PositiveDec ~ ., data = stocksTrain1, family = "binomial")

StocksModel2 <- glm(PositiveDec ~ ., data = stocksTrain2, family = "binomial")

StocksModel3 <- glm(PositiveDec ~ ., data = stocksTrain3, family = "binomial")

stock\_returns <- colnames(stocksTrain)[1:11]

*# put all coefficients of model into a single matrix*

coef\_mat <- cbind(coef(StocksModel1)[stock\_returns],

coef(StocksModel2)[stock\_returns],

coef(StocksModel3)[stock\_returns])

*# find variables where coefficients in all models have different "sign"*

*# ie not all positive or all negative*

names(which(!(apply(coef\_mat > 0, 1, all) | apply(coef\_mat < 0, 1, all))))

#### complex sorting (average+max)

names(sort(tapply(citi$tripduration, citi$day, mean), decreasing = TRUE)[1])

names(which.min(table(citi$starttime)))

unname(table(citi$gender)["2"]/nrow(citi))

which variable will dominate? Check max!

apply(citi[c("tripduration", "gender", "age", "starttime", "Mon")], 2, max)

#### Normalise the data

citi\_norm <- citi

*# only normalise numerical variables (use scale())*

vars\_to\_norm <- setdiff(names(citi), c("startstation", "endstation", "day"))

citi\_norm[vars\_to\_norm] <- apply(citi\_norm[vars\_to\_norm], 2, scale)

*# find the maximum tripduration after scaling*

max(citi\_norm$tripduration)

set.seed(100)

# k means clustering with normalised variables, k=10 (clusters)

km\_i <- kmeans(citi\_norm[,vars\_to\_norm], centers = 10)

# $size: number of observations (trips) in each cluster

min(km\_i$size)

max(km\_i$size)

#### # function which gives the closest clusters for each quantile

print\_closest\_by\_quantiles <- function(km\_obj, input\_variables) {

for (p in seq(0.8, 1, 0.05)) {

print(paste(sprintf("%-6s", p),

which.min(apply(

(apply(citi\_norm[input\_variables],

2, quantile, probs = p) -

t(km\_obj$centers[,input\_variables])) ^ 2,

2, sum))

)

)

}

}

print\_closest\_by\_quantiles(km\_j, var\_j)

#### Cluster distance application

Which cluster best fits the description “longer trips taken primarily by female users either on Tuesdays or Wednesdays”? You can use the centers of the clusters to answer this question.

# longer trips - tripduration (+), female - gender (+), Tues or Weds - Tue/Wed (+)

var\_k <- c("gender", "Tue", "Wed")

km\_j$centers[,var\_k]

print\_closest\_by\_quantiles(km\_j, var\_k)

#### add binary variable

**for** (day **in** days) {

citi[day] <- as.integer(citi$day == day)

}

#### Define a new class and clustering

*# create new variable 'weekday' - weekends correspond to 0, weekdays 1*

weekend\_days <- c("Sat", "Sun")

citi\_norm$weekday <- as.integer(!(citi$day %in% weekend\_days))

*# normalise variable 'weekday'*

citi\_norm$weekday <- scale(citi\_norm$weekday)

set.seed(100)

*# k means clusering for selected variables*

cluster\_var <- c("tripduration", "gender", "age", "starttime", "weekday")

km\_l <- kmeans(citi\_norm[,cluster\_var], centers = 10)

*# longer trips - tripduration (+), older - age (+), females - gender (+), weekdays - weekday (+)*

var\_n <- c("tripduration", "gender", "age", "weekday")

km\_l$centers[,var\_n]

print\_closest\_by\_quantiles(km\_l, var\_n)

#### Transform negative to positive

neg\_to\_pos <- c("tripduration", "age", "gender", "starttime")

km\_l$centers[,neg\_to\_pos] <- -km\_l$centers[,neg\_to\_pos]

*# after transforming the data, we want all the selected variables to be as positive as possible*

print\_closest\_by\_quantiles(km\_l, var\_o)

#### Question 3 a weighted average where the weights are the similarity metric defined by the Pearson correlation