HW1

# Question 2.1 Describe a situation or problem from your job, everyday life, current events, etc., for which a classification model would be appropriate. List some (up to 5) predictors that you might use.

## HIV Infection or not?

## Predictors: use intravenous drugs, have unprotected sex, have multiple sexual partners, have sexually transmitted infections (STIs) etc.

# load the packages

library(kernlab)

## Warning: package 'kernlab' was built under R version 3.5.2

library(kknn)

## Warning: package 'kknn' was built under R version 3.5.3

library(caret)

## Warning: package 'caret' was built under R version 3.5.3

## Loading required package: lattice

## Warning: package 'lattice' was built under R version 3.5.3

## Loading required package: ggplot2

## Warning: package 'ggplot2' was built under R version 3.5.2

##   
## Attaching package: 'ggplot2'

## The following object is masked from 'package:kernlab':  
##   
## alpha

##   
## Attaching package: 'caret'

## The following object is masked from 'package:kknn':  
##   
## contr.dummy

# read the data

setwd("//cdc.gov/private/L137/yks5/OMSA/ISYE6501/Homework1/week\_1\_data-summer/data 2.2")  
card<-read.table("credit\_card\_data.txt")

# get familiar with the data

head(card)

## V1 V2 V3 V4 V5 V6 V7 V8 V9 V10 V11  
## 1 1 30.83 0.000 1.25 1 0 1 1 202 0 1  
## 2 0 58.67 4.460 3.04 1 0 6 1 43 560 1  
## 3 0 24.50 0.500 1.50 1 1 0 1 280 824 1  
## 4 1 27.83 1.540 3.75 1 0 5 0 100 3 1  
## 5 1 20.17 5.625 1.71 1 1 0 1 120 0 1  
## 6 1 32.08 4.000 2.50 1 1 0 0 360 0 1

str(card)

## 'data.frame': 654 obs. of 11 variables:  
## $ V1 : int 1 0 0 1 1 1 1 0 1 1 ...  
## $ V2 : num 30.8 58.7 24.5 27.8 20.2 ...  
## $ V3 : num 0 4.46 0.5 1.54 5.62 ...  
## $ V4 : num 1.25 3.04 1.5 3.75 1.71 ...  
## $ V5 : int 1 1 1 1 1 1 1 1 1 1 ...  
## $ V6 : int 0 0 1 0 1 1 1 1 1 1 ...  
## $ V7 : int 1 6 0 5 0 0 0 0 0 0 ...  
## $ V8 : int 1 1 1 0 1 0 0 1 1 0 ...  
## $ V9 : int 202 43 280 100 120 360 164 80 180 52 ...  
## $ V10: int 0 560 824 3 0 0 31285 1349 314 1442 ...  
## $ V11: int 1 1 1 1 1 1 1 1 1 1 ...

summary(card)

## V1 V2 V3 V4   
## Min. :0.0000 Min. :13.75 Min. : 0.000 Min. : 0.000   
## 1st Qu.:0.0000 1st Qu.:22.58 1st Qu.: 1.040 1st Qu.: 0.165   
## Median :1.0000 Median :28.46 Median : 2.855 Median : 1.000   
## Mean :0.6896 Mean :31.58 Mean : 4.831 Mean : 2.242   
## 3rd Qu.:1.0000 3rd Qu.:38.25 3rd Qu.: 7.438 3rd Qu.: 2.615   
## Max. :1.0000 Max. :80.25 Max. :28.000 Max. :28.500   
## V5 V6 V7 V8   
## Min. :0.0000 Min. :0.0000 Min. : 0.000 Min. :0.0000   
## 1st Qu.:0.0000 1st Qu.:0.0000 1st Qu.: 0.000 1st Qu.:0.0000   
## Median :1.0000 Median :1.0000 Median : 0.000 Median :1.0000   
## Mean :0.5352 Mean :0.5612 Mean : 2.498 Mean :0.5382   
## 3rd Qu.:1.0000 3rd Qu.:1.0000 3rd Qu.: 3.000 3rd Qu.:1.0000   
## Max. :1.0000 Max. :1.0000 Max. :67.000 Max. :1.0000   
## V9 V10 V11   
## Min. : 0.00 Min. : 0 Min. :0.0000   
## 1st Qu.: 70.75 1st Qu.: 0 1st Qu.:0.0000   
## Median : 160.00 Median : 5 Median :0.0000   
## Mean : 180.08 Mean : 1013 Mean :0.4526   
## 3rd Qu.: 271.00 3rd Qu.: 399 3rd Qu.:1.0000   
## Max. :2000.00 Max. :100000 Max. :1.0000

# 2.2.1. Using the support vector machine function ksvm contained in the R package kernlab, find a good classifier for this data. Show the equation of your classifier, and how well it classifies the data points in the full data set. (Don’t worry about test/validation data yet; we’ll cover that topic soon.)

# convert the dataset to matrix

data<-as.matrix(card)  
class(data)

## [1] "matrix"

class(data[,11])

## [1] "numeric"

# Model1, set C=100

model1<-ksvm(data[,1:10],as.factor(data[,11]),type="C-svc",  
 kernel="vanilladot",C=100,scaled=TRUE)

## Setting default kernel parameters

## calculate a1.am

a <- colSums(model1@xmatrix[[1]] \* model1@coef[[1]])  
a

## V1 V2 V3 V4 V5   
## -0.0010065348 -0.0011729048 -0.0016261967 0.0030064203 1.0049405641   
## V6 V7 V8 V9 V10   
## -0.0028259432 0.0002600295 -0.0005349551 -0.0012283758 0.1063633995

## calculate a0

a0 <- -model1@b  
a0

## [1] 0.08158492

## see what the model predicts

pred1 <- predict(model1,data[,1:10])

## see what fraction of the model’s predictions match the actual classification

sum(pred1 == data[,11]) / nrow(data)

## [1] 0.8639144

# Model2, set C=1 and create a new model

model2<-ksvm(data[,1:10],as.factor(data[,11]),type="C-svc",  
 kernel="vanilladot",C=1,scaled=TRUE)

## Setting default kernel parameters

a2 <- colSums(model2@xmatrix[[1]] \* model2@coef[[1]])  
a2

## V1 V2 V3 V4 V5   
## -0.0011026642 -0.0008980539 -0.0016074557 0.0029041700 1.0047363456   
## V6 V7 V8 V9 V10   
## -0.0029852110 -0.0002035179 -0.0005504803 -0.0012519187 0.1064404601

a02 <- -model2@b  
a02

## [1] 0.08148382

pred2 <- predict(model2,data[,1:10])  
  
p2<-sum(pred2 == data[,11]) / nrow(data)  
p2

## [1] 0.8639144

## the result are the same to the model1

# Model3, set C=.001 and create a new model

model3<-ksvm(data[,1:10],as.factor(data[,11]),type="C-svc",  
 kernel="vanilladot",C=.001,scaled=TRUE)

## Setting default kernel parameters

a3 <- colSums(model3@xmatrix[[1]] \* model3@coef[[1]])  
a3

## V1 V2 V3 V4 V5   
## -0.002159778 0.032338170 0.046612449 0.111223162 0.375305335   
## V6 V7 V8 V9 V10   
## -0.202026081 0.169560847 -0.004923501 -0.025210266 0.081189766

a03 <- -model3@b  
a03

## [1] -0.2226155

pred3 <- predict(model3,data[,1:10])  
  
p3<-sum(pred3 == data[,11]) / nrow(data)  
p3

## [1] 0.8379205

p3-p2

## [1] -0.02599388

## the result are slightly worse than the model1 and model2. I will set C to a larger vlaue to see if it would improve the results

# Model4, set C=100000 and create a new model

model4<-ksvm(data[,1:10],as.factor(data[,11]),type="C-svc",  
 kernel="vanilladot",C=100000,scaled=TRUE)

## Setting default kernel parameters

a4 <- colSums(model4@xmatrix[[1]] \* model4@coef[[1]])  
a4

## V1 V2 V3 V4 V5   
## -0.004117738 -0.086896089 0.129715260 -0.083744032 0.988381368   
## V6 V7 V8 V9 V10   
## 0.031253888 -0.055666972 -0.037281856 0.021940744 0.018521785

a04<- -model4@b  
a04

## [1] 0.08054451

pred4 <- predict(model4,data[,1:10])  
  
p4<-sum(pred4 == data[,11]) / nrow(data)  
p4

## [1] 0.8639144

p4-p3

## [1] 0.02599388

p4-p2

## [1] 0

## This model performs similiar to the model 1 and model 2

# Model5, set C=1000000 and create a new model

model5<-ksvm(data[,1:10],as.factor(data[,11]),type="C-svc",  
 kernel="vanilladot",C=1000000,scaled=TRUE)

## Setting default kernel parameters

a5<- colSums(model5@xmatrix[[1]] \* model5@coef[[1]])  
a5

## V1 V2 V3 V4 V5 V6   
## -0.8283471 -0.2217216 -0.3301782 0.2825488 0.5750731 0.6143978   
## V7 V8 V9 V10   
## 0.2607774 -0.5943042 -1.1175369 0.9336833

a05<- -model5@b  
a05

## [1] -0.1281168

pred5 <- predict(model5,data[,1:10])  
  
p5<-sum(pred5 == data[,11]) / nrow(data)  
p5

## [1] 0.6253823

p5-p2

## [1] -0.2385321

p5-p3

## [1] -0.2125382

## this model is even worse than the model3. So increasing the C not nessarily improve the result.

## The “best” C may be somewhere between (100000,1000000) and (.001,100). So far, c=100 is the best solution. The equation of the classifier would be:0=0.08158492-0.0010065348\*v1-0.0011729048v2-0.0016261967v3+0.0030064203v4+1.0049405641v5-0.0028259432v6+0.0002600295v7-0.0005349551v8-0.0012283758v9+0.1063633995v10

# 2.2.2. You are welcome, but not required, to try other (nonlinear) kernels as well; we’re not covering them in this course, but they can sometimes be useful and might provide better predictions than vanilladot.

model6<-ksvm(data[,1:10],as.factor(data[,11]),type="C-svc",  
 kernel="polydot",C=100,scaled=TRUE)

## Setting default kernel parameters

a6<- colSums(model6@xmatrix[[1]] \* model6@coef[[1]])  
a6

## V1 V2 V3 V4 V5   
## -0.0010929705 -0.0012425741 -0.0015628157 0.0027739329 1.0051781402   
## V6 V7 V8 V9 V10   
## -0.0026901076 -0.0001935512 -0.0005270357 -0.0014583698 0.1063997443

a06<- -model6@b  
a06

## [1] 0.08157716

pred6 <- predict(model6,data[,1:10])  
  
p6<-sum(pred6 == data[,11]) / nrow(data)  
p6

## [1] 0.8639144

p6-p2

## [1] 0

p6-p3

## [1] 0.02599388

## This method provids similiar predictions as vanilladot, when c=100

# 2.2.3. Using the k-nearest-neighbors classification function kknn contained in the R kknn package, suggest a good value of k, and show how well it classifies that data points in the full data set. Don’t forget to scale the data (scale=TRUE in kknn).

acc<-function(k){  
  
fit<-rep(0,(nrow(card))) #start with 0s.  
   
for (i in 1:nrow(card)){  
 train<-card[-i,] #exclude itself  
 test<- card[i,]  
modelkknn<-kknn(V11~.,train,test,k = k, kernel = "optimal", ykernel = NULL, scale=TRUE)  
  
fit[i]<-as.integer(fitted(modelkknn)+0.5) #for rounding  
}  
 p223<-sum(fit == card[,11]) / nrow(card)   
 p223  
}

## Now call the function for values of k from 3 to 20

kseq <- rep(0,18)   
for (k in 3:20){  
 kseq[k] <-acc(k)   
}  
  
accuracy <- as.data.frame(kseq \* 100) #set accuracy as percentage  
accuracy

## kseq \* 100  
## 1 0.00000  
## 2 0.00000  
## 3 81.49847  
## 4 81.49847  
## 5 85.16820  
## 6 84.55657  
## 7 84.70948  
## 8 84.86239  
## 9 84.70948  
## 10 85.01529  
## 11 85.16820  
## 12 85.32110  
## 13 85.16820  
## 14 85.16820  
## 15 85.32110  
## 16 85.16820  
## 17 85.16820  
## 18 85.16820  
## 19 85.01529  
## 20 85.01529

max<-max(accuracy)  
final<-subset(accuracy,kseq \* 100==max)

## when k=12 or k=15, we got max accuracy 85.32%

# Question 3.1

# Using the same data set (credit\_card\_data.txt or credit\_card\_data-headers.txt) as in Question 2.2, use the ksvm or kknn function to find a good classifier:

# (a) using cross-validation (do this for the k-nearest-neighbors model; SVM is optional); and

set.seed(1234)

## set a max number of k

kmax<-20

## start with 0s.

rate<-rep(0,kmax)   
  
for (k in 1:kmax){  
 model3a<-cv.kknn(V11~.,card,kcv=10,#10-fold cross-validation  
 k = k,#number of neighbor, max=kmax=20  
 kernel = "optimal", ykernel = NULL, scale=TRUE)  
   
 fit<-as.integer(model3a[[1]][,2]+0.5) #round to 0 or 1  
 rate[k]<-sum(fit==card$V11)/nrow(card)  
}  
acc\_rate<-as.data.frame(rate \* 100) #set accuracy as percentage  
max3a<-max(acc\_rate)  
final3a<-subset(acc\_rate,rate \* 100==max3a)  
final3a

## rate \* 100  
## 11 86.54434

## when k=13, we got max accuracy 85.62691%

# (b) splitting the data into training, validation, and test data sets (pick either KNN or SVM; the other is optional).

## about 60% were selected for the train data

selecttrain<-sample(1:nrow(card),392)   
remain<-card[-selecttrain,]  
b\_train<-card[selecttrain,]

## the reamining 30% were equally divided into test and valid datasets

b\_test<-remain[1:131,]   
b\_valid<-remain[132:262,]  
  
cvalue<-c(0.00001,0.0001,0.001,0.01,0.1,1,100,10000,100000,1000000)  
p3b<-rep(0,10)  
  
for (i in 1:10){  
model3b<-ksvm(as.matrix(b\_train[,1:10]),  
 as.factor(b\_train[,11]),  
 C=cvalue[i],  
 type="C-svc", kernel="vanilladot",scaled=TRUE)  
  
pred3b <- predict(model3b,b\_valid[,1:10])  
p3b[i]<-sum(pred3b == b\_valid$V11) / nrow(b\_valid)  
  
}

## Setting default kernel parameters   
## Setting default kernel parameters   
## Setting default kernel parameters   
## Setting default kernel parameters   
## Setting default kernel parameters   
## Setting default kernel parameters   
## Setting default kernel parameters   
## Setting default kernel parameters   
## Setting default kernel parameters   
## Setting default kernel parameters

p3b[1:10]

## [1] 0.7328244 0.7328244 0.8015267 0.9236641 0.9236641 0.9236641 0.9236641  
## [8] 0.9236641 0.9236641 0.5114504

acc\_rate3b<-as.data.frame(p3b \* 100) #set accuracy as percentage  
max3b<-max(acc\_rate3b)  
final3b<-subset(acc\_rate3b,p3b \* 100==max3b)  
final3b

## p3b \* 100  
## 4 92.36641  
## 5 92.36641  
## 6 92.36641  
## 7 92.36641  
## 8 92.36641  
## 9 92.36641

## when c in c(0.01,0.1,1,100,10000,100000), we got the highest accuracy rate:91.60305%

## ues c=0.01 to re-train the model on the test dataset

model3r<-ksvm(as.matrix(b\_train[,1:10]),  
 as.factor(b\_train[,11]),  
 C=0.01,  
 type="C-svc", kernel="vanilladot",scaled=TRUE)

## Setting default kernel parameters

pred3r <- predict(model3r,b\_test[,1:10])  
p3r<-sum(pred3r == b\_test$V11) / nrow(b\_test)  
p3r\*100

## [1] 82.44275

## Performance on test data = 82.44275%