**Question 2.2**

Source: credit\_card\_data-headers.txt (with headers)

Demographics: dataset with 654 data points,6 continuous and 4 binary predictor variables.

# **Ask:**

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.)

# Process:

## Initial analysis:



## Ingest data:

Data as csv

Started with simple csv read to ensure the data is read correctly

Fix(Data) to view the data in table format

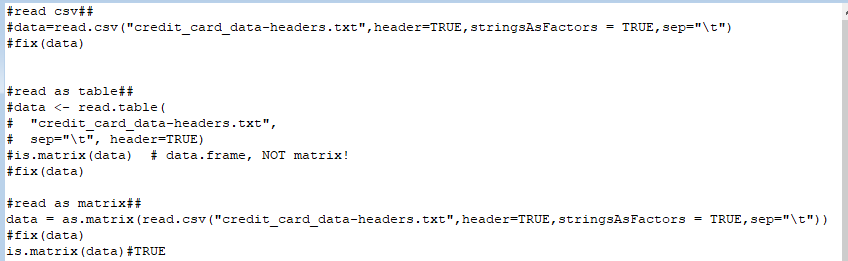
Data as table

Workes fine and remove \t delimiter

*Challenges* : not successful in using this to create a model

Data as matrix

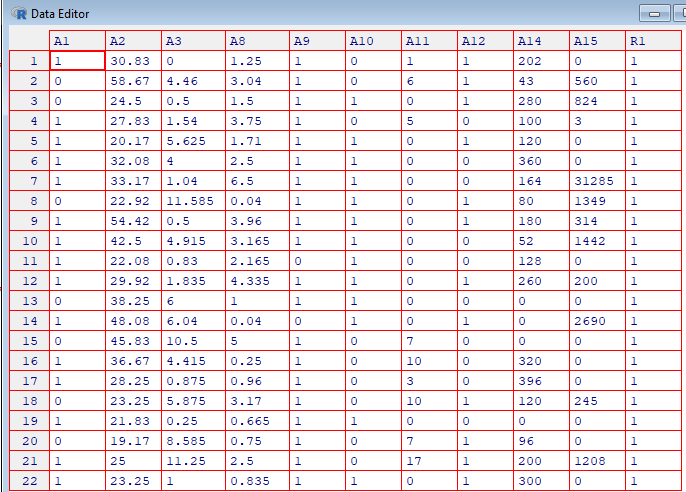
Using as.matrix, able to view data



### Results:

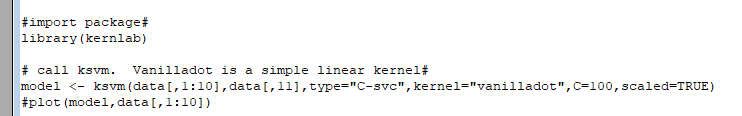
Validation: confirmed the data is read as matrix



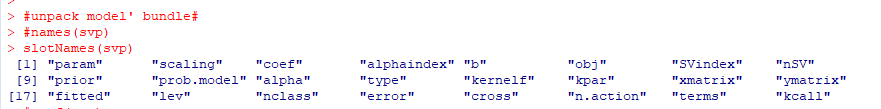


## Classifier creation

Created first classifier with cost of 100



### Unpacked the bundle of classifier to get co-efficients



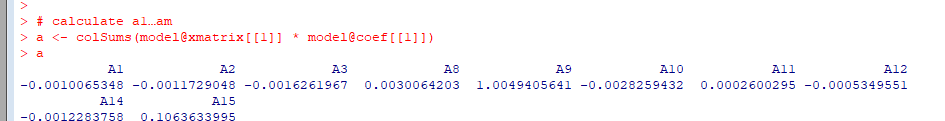
### Model error



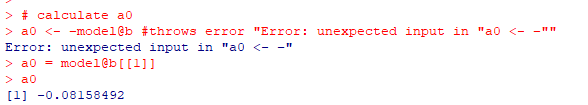
### **Classifier Co-efficients and Intercepts**

**w = ∑ αi yi xi +a0**

**Classifier : ∑ αi yi < xi · x > + b = 0**

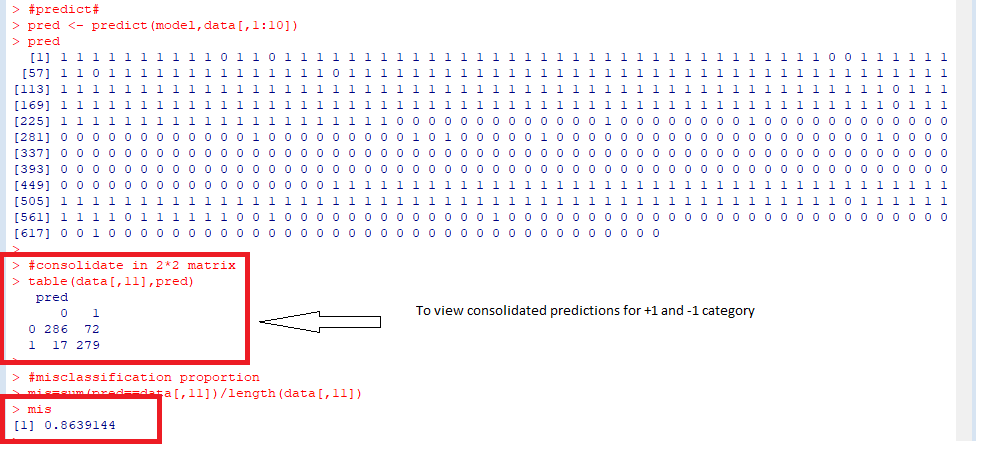


The process to obtain optimal intercepts was giving me an error



Hence, used alternate option to obtain intercepts

## Predict and understand misclassification proportion



# Classifier Validation & Optimization

By altering C factor, studied the model error and misclassification proportion of the classifier



More results…

Below classifier shows lower classifier mis classification and cost classification

To summarize, in my opinion the major **drawback** of the model is the performance for a relatively smaller subset of data (<1000), especially for higher values of Cost factor



# R Code below:

#read csv##

#data=read.csv("credit\_card\_data-headers.txt",header=TRUE,stringsAsFactors = TRUE,sep="\t")

#fix(data)

#read as table##

#data <- read.table(

# "credit\_card\_data-headers.txt",

# sep="\t", header=TRUE)

#is.matrix(data) # data.frame, NOT matrix!

#fix(data)

#read as matrix##

data = as.matrix(read.csv("credit\_card\_data-headers.txt",header=TRUE,stringsAsFactors = TRUE,sep="\t"))

#fix(data)

is.matrix(data)#TRUE

#import package#

library(kernlab)

# call ksvm. Vanilladot is a simple linear kernel#

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

#plot(model,data[,1:10])

#unpack model' bundle#

#names(svp)

slotNames(svp)

#coef(svp)

error(model)

#plot(data[,1:10],col=ifelse(data[,11]>0,'blue','red'))

# calculate a1…am

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

a

# calculate a0

a0 <- –model@b #throws error "Error: unexpected input in "a0 <- –""

a0 = model@b[[1]]

a0

#predict#

pred <- predict(model,data[,1:10])

pred

#consolidate in 2\*2 matrix

table(data[,11],pred)

#misclassification proportion

mis=sum(pred==data[,11])/length(data[,11])

mis

# see what fraction of the model’s predictions match the actual classification(same as above)

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

#plot(model]) #error in local(object): test vector doesnt match model

#Asses performance of model

#modelC=1#

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

error(model1)

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

table(data[,11],pred1)

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

#modelC=10#

model10 <- ksvm(data[,1:10],data[,11],type="C-svc",kernel="vanilladot",C=10,scaled=TRUE)

error(model10)

pred2 <- predict(model10,data[,1:10])

table(data[,11],pred2)

sum(pred2==data[,11])/length(data[,11])

#modelC=100#

model100 <- ksvm(data[,1:10],data[,11],type="C-svc",kernel="vanilladot",C=100,scaled=TRUE)

error(model100)

pred100 <- predict(model100,data[,1:10])

table(data[,11],pred100)

sum(pred100==data[,11])/length(data[,11])

#modelC=M#

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

error(modelM)

predM <- predict(modelM,data[,1:10])

table(data[,11],predM)

sum(predM==data[,11])/length(data[,11])

#scaling

modelF <- ksvm(data[,1:10],data[,11],type="C-svc",kernel="vanilladot",C=1000000,c(F,F))

error(modelF)

predF <- predict(modelF,data[,1:10])

table(data[,11],predF)

sum(predF==data[,11])/length(data[,11])