Question 1

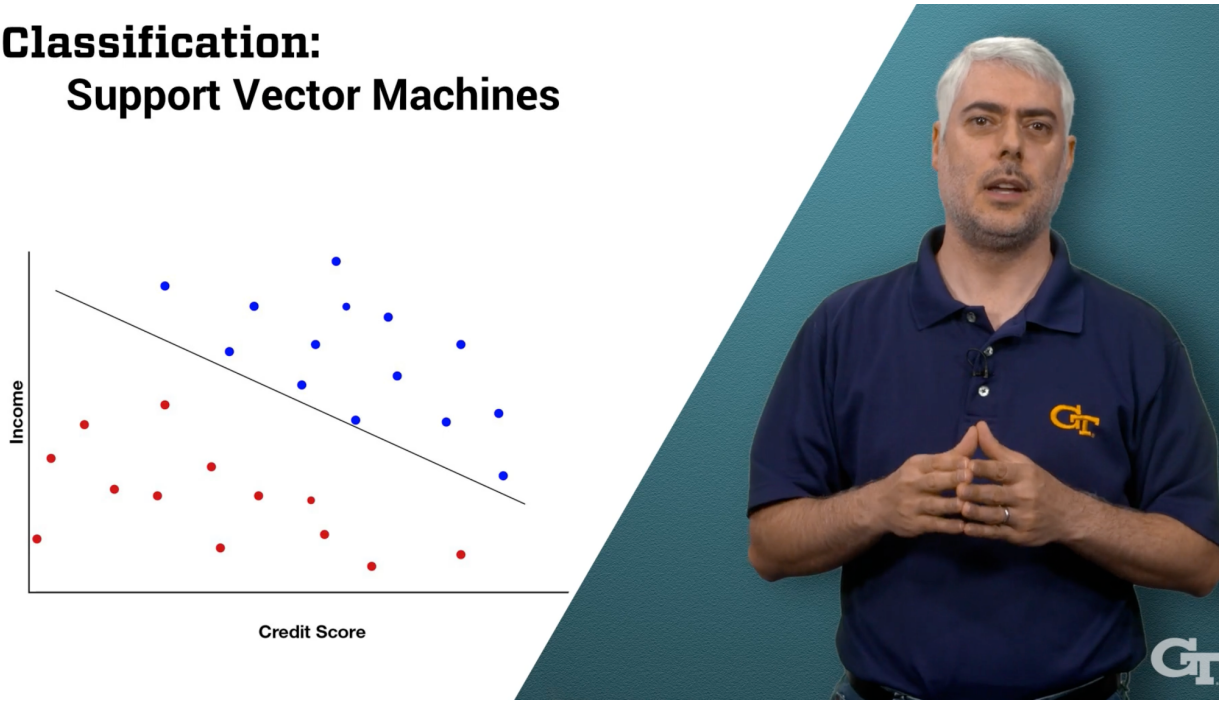
QUESTION: *Describe a situation or problem from your job, everyday life, current events, etc., for which a classification model would be appropriate. List some (up to 5) predictors that you might use*.

ANSWER:

One situation where I think classification could be useful is politics. It would be very interesting to try and predict an individuals political orientation based on a few different predictors (level of education, income, urbanicity, race, gender). If you can accurately make this prediction, this could immensely help political parties with how the advertise and message at an individual level. For example, if they can reasonably predict that you are a moderate who often swings your vote, they might message in a particular way to you and be willing to spend more to try and persuade you.

This classification might work by having a classification line that separates republicans from democrats. Theoretically, the closer you are to the line, the more likely you are to be an independent. You would need a good response data set, which may difficult to acquire considering individuals are not required to reveal who they voted for (and their political orientation can change). However, there are reasonable ways you can get a response data set. For example, political parties often have voter registration data showing whether or not you are a registered democrat, independent, or conservative. You can join this data set with census data that readily provides information on the predictors (level of education, income, urbanicity, race, gender, etc.)

The interesting thing here is that some of these predictors are categorical (level of education, race, gender) and others are continuous (income, potentially urbanicity). Because of this combination of categorical and continuous data, you need to be very careful in scaling your data. Additionally, one interesting idea is that this can be an extremely soft classification since the risks of getting an error in your classification are relatively harmless. While you may make the mistake of targeting unpersuadable voters with your media dollars, that is a small risk (compared to viruses, death, etc.) and could potentially have the benefit that you do happen to nudge someone in a way you wanted to. Because this could be a ‘soft’ classification, you can not worry as much about the error (i.e., if you were working for the Democratic party, your classification line could be extremely close or relatively deep into conservative terrain so you are sure to capture as much of the potential persuadable vote as possible). This would be more similar to this type of classifier from Professor Sokkel’s lecture:



Note that the classifier line is deep in blue terrain and not necessarily 'centered'.

Note: regarding using both categorical and continuous data in an SVM, II found this answer on substack, so I imagine a lot of the key work happens with the scaling: "SVMs will handle both binary and continuous variables as long as you make some preprocessing: all features should be scaled or normalised. After that step, from the algorithms' perspective it doesn't matter if features are continuous or binary: for binaries, it sees samples that are either "far" away, or very similar; for continuous there are also the in between values. Kernel doesn't matter in respect to the type of variables."

Question 2.1

QUESTION: *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.)*

Classifier line formula: a1x1 + a2x2… anxn + a0 = 0

Our line: (-.001)x1 - (.001)x2 - (.002)x3 +(.003)x4 + (1.00)x5 - (.003)x6 + (.000)x7 - (.001)x8 - (.001)x9 + (.106)x10 - .082 = 0

Note: I rounded to three digits for ease of writing the solution. For mathematical precision, I would've rounded further.

Predictor = The model accurately determines the response 86.3% of the time using 100 as C.

Note: for the purpose of assessing credit card applications, I think 86% accuracy is pretty low as the cost of a default could be pretty high for a financial institution. To further test this it would be helpful to know the avg profitability of a cc applicant who doesn't default and the avg loss to a bank for each default. This can help you triangulate what the right C should be.

```
install.packages("devtools")
library(devtools)
install.packages("kernlab", dependencies = TRUE, repos = "http://cran.r-project.org")
library(kernlab)

rm(list = ls())
set.seed(42)

ccdataheaders <- read.table("/Users/imranahmed/Desktop/GATech/Intro_To_Analytics_Modeling/hw1-SP22/data 2.2/credit_card_data-headers.txt")
ccdataheadersmatrix <- as.matrix(ccdataheaders)
ccdata <- read.table("/Users/imranahmed/Desktop/GATech/Intro_To_Analytics_Modeling/hw1-SP22/data 2.2/credit_card_data.txt")
ccdatamatrix <- as.matrix(ccdata)

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

a <- colSums(model@xmatrix[[1]]*model@coef[[1]])
a0 <- model@b
a0
a

1-model@error

pred <- predict(model,ccdatamatrix[,1:10])
pred
sum(pred == ccdatamatrix[,11]) / nrow(ccdatamatrix)
```

Question 2.3

Using k = 10, I am getting that the model classifies the dataset with an 85% accuracy. Below is the code. It appears though, that k=15 has a slightly better accuracy (85.3% vs. 85.0%). See code below:

```r
install.packages("kknn", dependencies = TRUE, repos = "http://cran.r-project.org")
library(kknn)

rm(list = ls())
set.seed(42)

ccdataheaders <- read.table("/Users/imranahmed/Desktop/GATech/Intro_To_Analytics_Modeling/hw1-SP22/data 2.2/credit_card_data-headers.txt")
ccdataheadersmatrix <- as.matrix(ccdataheaders)
ccdata <- read.table("/Users/imranahmed/Desktop/GATech/Intro_To_Analytics_Modeling/hw1-SP22/data 2.2/credit_card_data.txt")
ccdatamatrix <- as.matrix(ccdata)

k_values <- c(5, 10, 15, 20, 25, 30, 35, 40)
x1 <- c()
y1 <- c()
for (k in k_values) {
  z1 <- c()
  for (i in 1:654){
    kknnmodel = kknn(V11~V1+V2+V3+V4+V5+V6+V7+V8+V9+V10,
                     ccdata[-i,],
                     ccdata[i,],
                     k=k,
                     distance = 2,
                     kernel = "optimal",
                     scale = TRUE)
    x = round(fitted.values(kknnmodel))
    y = ccdata[i,11]
    z = x==y
    x1<-c(x1,x)
    y1<-c(y1,y)
    z1<-c(z1,z)
  }
  a <- sum(z1)/654
  print(paste0("Accuracy for k = ", k, ": ", a))
}
```

```
[1] "Accuracy for k = 5: 0.851681957186544"
[1] "Accuracy for k = 10: 0.850152905198777"
[1] "Accuracy for k = 15: 0.853211009174312"
[1] "Accuracy for k = 20: 0.850152905198777"
[1] "Accuracy for k = 25: 0.845565749235474"
[1] "Accuracy for k = 30: 0.840978593272171"
[1] "Accuracy for k = 35: 0.831804281345566"
[1] "Accuracy for k = 40: 0.831804281345566"
```