ISYE 6501- HW 1 Classification

Ryan Cherry

2024-01-09

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

One of the things I enjoy doing in my free time is watching TV shows on my list (of about 90). Whether I want to watch the TV show now or at a later date could be determined by a combination of some of the following predictors:

Mood of the plot: is the plot thought-provoking and challenging, or light and fun Number of episodes: is there a large number of episodes, over 80, which will require a longer time commitment Streaming service on: am I currently subscribed to the streaming service the TV show is on or not Am I busy: how many hours of the day am I focusing on my master's program or life in general

Usually, if I am busy with school like I am now, I pick a show that does not have a thought-provoking or complicated plot or that has a high episode count that requires a longer time commitment. The combination of these predictors helps me determine the next TV show I watch in my spare time.

Question 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.

The value of c was 0.8639, meaning that the support vector machine model with a vanilla dot kernel correctly predicted about 86% of the model's predictions. With each successive value of c, the classification rate remained unchanged, so a value of 10 for C should be used.

The resulting equation is as follows:

V11 = 0.0816 - 0.000903 *V1 - 0.000789* V2 - 0.00170 *V3 + 0.00261* V4 + 1.005 *V5 - 0.00284* V6 - 0.000160 *V7 - 0.000393* V8 - 0.00128 *V9 + 0.106* V10

Note: I experimented with values of c up to 10000 but, for the sake of reducing the code output, have only included c values up to 1000.

```
#the necessary packages were installed
install.packages("tidyverse", repos = "http://cran.us.r-project.org")
```

```
## Installing package into 'C:/Users/ryanc/AppData/Local/R/win-library/4.2'
## (as 'lib' is unspecified)
```

```
## package 'tidyverse' successfully unpacked and MD5 sums checked
##
## The downloaded binary packages are in
##  C:\Users\ryanc\AppData\Local\Temp\RtmpaE2cb9\downloaded_packages
```

```r
install.packages("kernlab", repos = "http://cran.us.r-project.org")
```

```
## Installing package into 'C:/Users/ryanc/AppData/Local/R/win-library/4.2'
## (as 'lib' is unspecified)
```

```
## package 'kernlab' successfully unpacked and MD5 sums checked
```

```
## Warning: cannot remove prior installation of package 'kernlab'
```

```
## Warning in file.copy(savedcopy, lib, recursive = TRUE): problem copying
## C:\Users\ryanc\AppData\Local\R\win-library\4.2\00LOCK\kernlab\libs\x64\kernlab.dll
## to C:\Users\ryanc\AppData\Local\R\win-library\4.2\kernlab\libs\x64\kernlab.dll:
## Permission denied
```

```
## Warning: restored 'kernlab'
```

```
##
## The downloaded binary packages are in
##  C:\Users\ryanc\AppData\Local\Temp\RtmpaE2cb9\downloaded_packages
```

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

```
## Installing package into 'C:/Users/ryanc/AppData/Local/R/win-library/4.2'
## (as 'lib' is unspecified)
```

```
## package 'kknn' successfully unpacked and MD5 sums checked
```

```
## Warning: cannot remove prior installation of package 'kknn'
```

```
## Warning in file.copy(savedcopy, lib, recursive = TRUE): problem copying
## C:\Users\ryanc\AppData\Local\R\win-library\4.2\00LOCK\kknn\libs\x64\kknn.dll
## to C:\Users\ryanc\AppData\Local\R\win-library\4.2\kknn\libs\x64\kknn.dll:
## Permission denied
```

```
## Warning: restored 'kknn'
```

```
##
## The downloaded binary packages are in
##  C:\Users\ryanc\AppData\Local\Temp\RtmpaE2cb9\downloaded_packages
```

```r
library(kernlab)
```

```
## Warning: package 'kernlab' was built under R version 4.2.2
```

```r
library(tidyverse)
```

```
## Warning: package 'tidyverse' was built under R version 4.2.3
```

```
## Warning: package 'ggplot2' was built under R version 4.2.3

## Warning: package 'tibble' was built under R version 4.2.3

## Warning: package 'tidyr' was built under R version 4.2.3

## Warning: package 'readr' was built under R version 4.2.3

## Warning: package 'purrr' was built under R version 4.2.3

## Warning: package 'dplyr' was built under R version 4.2.3

## Warning: package 'stringr' was built under R version 4.2.3

## Warning: package 'forcats' was built under R version 4.2.3

## Warning: package 'lubridate' was built under R version 4.2.3

## -- Attaching core tidyverse packages ------------------------ tidyverse 2.0.0 --
## v dplyr     1.1.4     v readr     2.1.4
## v forcats   1.0.0     v stringr   1.5.1
## v ggplot2   3.4.4     v tibble    3.2.1
## v lubridate 1.9.3     v tidyr     1.3.0
## v purrr     1.0.2

## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x ggplot2::alpha() masks kernlab::alpha()
## x purrr::cross()   masks kernlab::cross()
## x dplyr::filter()  masks stats::filter()
## x dplyr::lag()     masks stats::lag()
## i Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to become errors
```

```
library(kknn)
```

```
## Warning: package 'kknn' was built under R version 4.2.3
```

```
#the credit card data file was read in
credit_card_data <- read.delim("C:/Users/ryanc/Downloads/credit_card_data.txt", header=FALSE)

#to get a feel for the data, the first 5 observations of the data
#set were viewed
head(credit_card_data)
```

```
##   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
```

```r
#the variable type of each variable in the data set was checked

class(credit_card_data$V1)
```

```
## [1] "integer"
```

```r
class(credit_card_data$V2)
```

```
## [1] "numeric"
```

```r
class(credit_card_data$V3)
```

```
## [1] "numeric"
```

```r
class(credit_card_data$V4)
```

```
## [1] "numeric"
```

```r
class(credit_card_data$V5)
```

```
## [1] "integer"
```

```r
class(credit_card_data$V6)
```

```
## [1] "integer"
```

```r
class(credit_card_data$V7)
```

```
## [1] "integer"
```

```r
class(credit_card_data$V8)
```

```
## [1] "integer"
```

```r
class(credit_card_data$V9)
```

```
## [1] "integer"
```

```r
class(credit_card_data$V10)
```

```
## [1] "integer"
```

```r
class(credit_card_data$V11)
```

```
## [1] "integer"
```

```r
#The variables were converted to the appropriate type, 6 numeric
#and 5 categorical.
as.factor(credit_card_data$V1)
```

```
##   [1] 1 0 0 1 1 1 1 0 1 1 1 1 0 1 0 1 1 0 1 0 1 1 0 0 0 0 0 1 1 1 1 1 1 0 0 1 1
##  [38] 0 1 1 1 1 1 1 1 1 1 0 1 1 1 0 1 1 1 1 1 1 1 1 1 1 1 0 0 1 1 0 1 1 1 1 0 1 1
##  [75] 1 1 0 1 1 1 0 1 1 1 1 1 1 0 1 1 1 1 0 1 0 0 1 1 1 1 1 1 1 0 1 0 1 0 1 0 1 0
## [112] 1 1 1 0 1 1 0 0 0 1 1 1 1 1 0 1 1 0 1 1 1 1 1 1 0 0 0 0 1 1 1 1 1 0 0 0 0 0 1
## [149] 1 1 0 0 0 1 1 1 1 1 1 1 0 0 1 0 1 1 0 1 1 0 1 1 1 0 0 1 1 0 0 0 0 1 1 1 0 0
## [186] 1 1 1 1 1 1 1 1 1 1 1 0 0 0 1 0 1 1 1 1 1 1 1 1 1 1 1 1 0 0 1 0 0 0 1 1 0 1
## [223] 1 1 0 1 1 1 0 0 1 1 1 1 1 1 1 1 1 1 1 0 1 1 1 0 1 1 1 1 1 0 0 1 0 0 0 1 1 1 0 1
## [260] 1 1 1 1 1 1 1 0 0 1 1 1 1 1 0 1 1 1 1 1 0 1 0 1 1 0 0 1 1 1 1 0 1 1 0 0 1 1 1 1
## [297] 0 0 1 0 0 1 1 1 1 1 1 1 1 0 0 1 1 0 1 1 0 1 0 1 1 1 1 0 0 1 0 1 1 1 1 1 1 1 1
## [334] 0 1 1 1 1 1 0 1 0 1 0 0 1 1 1 1 1 1 0 1 0 1 1 1 0 1 0 1 1 1 1 1 1 0 0 0 1 1 1 1
## [371] 1 1 1 1 1 1 1 1 0 1 0 1 1 1 1 1 0 0 1 0 0 1 1 1 1 1 0 1 1 0 1 1 1 1 0 1 1 1 1 1 1
## [408] 0 1 1 1 1 1 1 1 1 1 1 1 1 0 0 1 1 1 1 1 1 0 1 0 1 1 1 1 1 1 1 1 1 1 1 0 1 1 0 0 0 1
## [445] 0 1 1 0 1 1 1 1 1 1 1 1 1 0 1 0 1 1 1 1 1 1 1 0 1 1 0 0 1 0 1 1 1 1 1 0 1 0 1 0
## [482] 0 0 1 1 0 0 1 1 0 1 0 1 0 1 0 0 1 1 0 0 1 0 0 1 1 1 0 0 1 1 1 1 1 1 1 1 1 1
## [519] 1 1 1 1 1 1 1 1 1 0 1 1 0 1 0 0 1 1 1 1 1 0 1 0 1 0 1 1 1 0 0 1 1 1 1 1 1 1 1
## [556] 1 0 1 1 1 1 1 1 1 1 1 1 1 1 0 0 1 0 1 1 1 1 0 1 1 1 1 1 1 1 1 0 1 1 1 1 1 1 1 1
## [593] 1 1 1 0 0 0 0 1 0 1 1 1 1 1 1 1 1 0 1 1 0 1 1 0 1 0 0 1 0 0 1 0 0 0 1 1 0 1
## [630] 1 1 0 0 1 1 1 1 1 0 0 0 0 1 0 0 1 1 1 1 1 1 1 0 0 1 1
## Levels: 0 1
```

```r
as.factor(credit_card_data$V5)
```

```
##   [1] 1 1 1 1 1 1 1 1 1 1 1 0 1 1 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
##  [38] 1 1 1 1 1 1 1 1 1 1 1 1 0 0 1 1 1 1 1 1 1 1 1 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0
##  [75] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
## [112] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
## [149] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
## [186] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 1
## [223] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1
## [260] 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0
## [297] 0 0 0 0 0 0 1 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0
## [334] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
## [371] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
## [408] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
## [445] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
## [482] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
## [519] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 1
## [556] 1 1 1 1 1 1 1 1 1 0 1 1 1 1 1 1 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0
## [593] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0
## [630] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
## Levels: 0 1
```

```r
as.factor(credit_card_data$V6)
```

```
##   [1] 0 0 1 0 1 1 1 1 1 1 1 1 1 1 0 0 0 0 1 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0
##  [38] 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1
##  [75] 0 1 1 0 1 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 1 1 0 0 0 0 1 1 1 0 0 0 1 0 0 0
## [112] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
```

```
## [149] 0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0
## [186] 1 1 1 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 1 0 0 0 1 0 0 0 0 0 0 1 1 1 1 1 0
## [223] 1 0 0 0 1 0 0 1 0 1 0 1 0 1 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1
## [260] 1 1 1 1 1 1 1 1 1 1 1 1 0 0 0 0 0 1 1 0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1
## [297] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 0
## [334] 1 1 1 1 1 1 0 1 1 1 1 1 1 1 1 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
## [371] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 0 0 0 0 1 1 1 1 1 1 0 1 1 1 1 1 1
## [408] 1 1 1 1 1 1 1 1 1 1 0 0 1 0 0 0 0 1 1 1 1 1 0 1 1 1 1 1 1 1 1 1 0 1 0 0 0 0
## [445] 0 1 1 1 1 1 1 1 1 1 1 1 1 0 0 0 0 1 0 1 1 1 0 0 0 1 1 1 1 0 0 0 0 0 0 0 0 0
## [482] 0 0 0 0 1 1 1 1 0 0 0 0 0 0 0 1 1 1 1 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1 0 1
## [519] 0 0 0 1 0 0 0 0 0 0 0 0 0 1 0 0 1 1 1 1 1 0 1 0 0 0 0 0 0 0 0 0 0 0 0 1
## [556] 1 1 1 0 0 0 0 0 0 0 1 1 1 0 0 0 0 1 1 1 1 1 1 1 1 0 0 0 1 0 0 0 0 1 1 1 1 1
## [593] 1 1 1 1 1 1 0 1 1 0 0 1 1 1 1 1 1 1 1 1 1 1 1 0 1 0 1 1 0 1 0 1 1 0 1 1 1 1
## [630] 1 1 1 0 1 1 1 1 1 1 1 0 0 1 1 1 1 1 1 1 1 1 0 0 1 1
## Levels: 0 1
```

```
as.factor(credit_card_data$V8)
```

```
##   [1] 1 1 1 0 1 0 0 1 1 0 0 1 0 1 0 0 0 1 0 1 1 1 0 1 0 1 0 0 1 0 0 1 0 0 0 0 0
##  [38] 0 1 0 0 1 1 0 1 0 0 1 1 1 0 0 0 0 0 0 0 1 0 0 1 0 1 0 1 0 0 1 1 1 0 0 0 1 1
##  [75] 1 0 0 1 1 1 1 1 0 0 1 0 0 1 1 0 0 0 1 0 0 0 1 1 0 0 1 1 1 0 0 0 1 0 0 0 1
## [112] 0 1 0 0 1 0 0 0 0 0 0 0 1 1 0 1 1 1 0 0 1 1 0 1 0 1 1 1 1 1 0 1 0 1 0 1 0
## [149] 0 0 0 1 1 0 0 1 0 0 0 0 1 0 1 0 0 0 1 0 0 1 0 0 1 1 1 1 1 1 1 1 1 1 1 0 1
## [186] 0 1 1 1 0 0 0 1 0 1 1 0 0 1 1 1 1 1 0 0 0 1 1 1 1 1 1 0 0 0 0 1 0 0 1 1 0
## [223] 1 1 0 0 0 1 1 0 1 1 1 1 1 0 1 0 0 1 1 1 0 1 1 1 0 1 1 1 0 1 1 0 1 1 1 1 0
## [260] 1 0 1 1 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 1 1 1 1 0 1 0 1 0 1 0 0 1 1 0 1 1 0
## [297] 1 0 0 1 1 1 0 1 0 0 0 1 0 1 1 1 1 1 1 1 0 1 0 1 0 0 0 1 0 1 1 0 1 0 1 1 0 0
## [334] 0 1 0 0 0 0 1 0 0 1 0 0 1 1 1 0 1 1 0 0 0 0 0 1 1 1 0 0 0 0 1 1 0 0 0 0 0
## [371] 0 1 1 1 1 0 0 1 0 1 1 0 1 1 1 0 1 0 1 1 1 1 1 1 1 1 0 1 1 0 1 0 0 1 1 1 1 1
## [408] 0 1 1 1 1 1 1 0 1 0 1 1 0 1 0 1 1 0 0 1 1 1 1 1 1 1 0 1 1 1 0 1 1 1 1 1 0 0
## [445] 1 1 0 1 1 1 1 0 1 0 0 0 1 0 0 0 1 1 1 1 0 0 1 0 1 1 1 1 1 0 0 0 1 0 1 1 0 1 1
## [482] 1 1 0 0 1 1 1 1 1 1 0 1 0 0 1 1 0 0 1 0 0 1 0 1 1 1 1 0 1 1 0 0 1 0 1 0 0 0 1
## [519] 0 1 0 1 0 1 0 1 1 1 0 1 1 0 1 1 1 0 0 1 0 1 0 1 0 0 0 0 1 0 1 1 1 0 0 1 0
## [556] 0 1 0 0 1 0 0 1 0 0 0 1 0 1 0 1 1 1 1 0 1 0 0 1 0 1 0 1 0 0 1 0 1 1 0 1 1
## [593] 1 1 1 1 1 0 0 1 1 0 0 0 1 0 0 0 0 1 0 0 1 0 0 1 1 0 0 0 0 1 0 1 1 0 1 1 0 0
## [630] 1 1 0 0 0 1 1 1 0 1 1 0 1 1 1 1 1 1 0 1 0 1 0 0 1 0
## Levels: 0 1
```

```
as.factor(credit_card_data$V11)
```

```
##   [1] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
##  [38] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 0 0 0 0
##  [75] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
## [112] 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
## [149] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
## [186] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
## [223] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 1
## [260] 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
## [297] 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
## [334] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
## [371] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
## [408] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
## [445] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
```

```
## [482] 1 1 1 1 1 1 1 1 1 1 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
## [519] 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
## [556] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0
## [593] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
## [630] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
## Levels: 0 1
```

```
as.numeric(credit_card_data$V2)
```

```
##   [1] 30.83 58.67 24.50 27.83 20.17 32.08 33.17 22.92 54.42 42.50 22.08 29.92
##  [13] 38.25 48.08 45.83 36.67 28.25 23.25 21.83 19.17 25.00 23.25 47.75 27.42
##  [25] 41.17 15.83 47.00 56.58 57.42 42.08 29.25 42.00 49.50 36.75 22.58 27.83
##  [37] 27.25 23.00 27.75 54.58 34.17 28.92 29.67 39.58 56.42 54.33 41.00 31.92
##  [49] 41.50 23.92 25.75 26.00 37.42 34.92 34.25 23.33 23.17 44.33 35.17 43.25
##  [61] 56.75 31.67 23.42 20.42 26.67 34.17 36.00 25.50 19.42 35.17 32.33 38.58
##  [73] 44.25 44.83 20.67 34.08 19.17 21.67 21.50 49.58 27.67 39.83 27.25 37.17
##  [85] 25.67 34.00 49.00 62.50 31.42 52.33 28.75 28.58 23.00 22.50 28.50 37.50
##  [97] 35.25 18.67 25.00 27.83 54.83 28.75 25.00 40.92 19.75 29.17 24.50 24.58
## [109] 33.75 20.67 25.42 37.75 52.50 57.83 20.75 39.92 25.67 24.75 44.17 23.50
## [121] 34.92 47.67 22.75 34.42 28.42 67.75 20.42 47.42 36.25 32.67 48.58 39.92
## [133] 33.58 18.83 26.92 31.25 56.50 43.00 22.33 27.25 32.83 23.25 40.33 30.50
## [145] 52.83 46.67 58.33 37.33 23.08 32.75 21.67 28.50 68.67 28.00 34.08 27.67
## [157] 44.00 25.08 32.00 60.58 40.83 19.33 32.33 36.67 37.50 25.08 41.33 56.00
## [169] 49.83 22.67 27.00 25.00 26.08 18.42 20.17 47.67 21.25 20.67 57.08 22.42
## [181] 48.75 40.00 40.58 28.67 33.08 21.33 42.00 41.75 22.67 34.50 28.25 33.17
## [193] 48.17 27.58 22.58 24.08 41.33 20.75 36.33 35.42 28.67 35.17 39.50 39.33
## [205] 24.33 60.08 23.08 26.67 48.17 41.17 55.92 53.92 18.92 50.08 65.42 17.58
## [217] 18.83 37.75 23.25 18.08 22.50 19.67 22.08 25.17 47.42 33.50 27.67 58.42
## [229] 20.67 26.17 21.33 42.83 38.17 20.50 48.25 28.33 18.50 33.17 45.00 19.67
## [241] 21.83 40.25 41.42 17.83 23.17 18.17 20.00 20.00 20.75 24.50 32.75 52.17
## [253] 48.17 20.42 50.75 17.08 18.33 32.00 59.67 18.00 32.33 18.08 38.25 30.67
## [265] 18.58 19.17 18.17 16.25 21.17 23.92 17.67 16.50 23.25 17.58 29.50 18.83
## [277] 21.75 23.00 18.25 25.42 35.75 16.08 31.92 69.17 32.92 16.33 22.17 57.58
## [289] 18.25 23.42 15.92 24.75 48.75 23.50 18.58 27.75 31.75 24.83 19.00 16.33
## [301] 18.58 16.25 23.00 21.17 17.50 19.17 36.75 21.25 18.08 33.67 48.58 33.67
## [313] 29.50 30.17 34.83 33.25 34.08 25.25 34.75 27.67 47.33 34.83 33.25 28.00
## [325] 39.08 42.75 26.92 33.75 38.92 62.75 26.75 63.33 27.83 26.17 22.17 22.50
## [337] 30.75 36.67 16.00 41.17 19.50 32.42 36.75 30.25 23.08 26.83 16.92 24.42
## [349] 42.83 22.75 39.42 23.58 21.42 33.00 26.33 45.00 26.25 20.83 28.67 20.67
## [361] 34.42 33.58 43.17 22.67 24.33 56.83 22.08 34.00 22.58 21.17 26.67 22.92
## [373] 15.17 39.92 27.42 24.75 41.17 33.08 29.83 23.58 26.17 31.00 20.75 28.92
## [385] 51.92 22.67 34.00 69.50 19.58 16.00 17.08 31.25 25.17 22.67 40.58 22.25
## [397] 22.25 22.50 23.58 38.42 26.58 35.00 20.42 29.42 26.17 33.67 24.58 27.67
## [409] 37.50 49.17 33.58 51.83 22.92 21.83 25.25 58.58 19.00 19.58 53.33 27.17
## [421] 25.92 23.08 39.58 30.58 17.25 17.67 16.50 27.33 31.25 20.00 39.50 36.50
## [433] 52.42 36.17 29.67 36.17 25.67 24.50 24.08 21.92 36.58 23.00 27.58 31.08
## [445] 30.42 22.08 16.33 21.92 21.08 17.42 19.17 20.67 26.75 23.58 39.17 22.75
## [457] 16.92 23.50 17.33 23.75 34.67 74.83 28.17 24.50 18.83 47.25 24.17 39.25
## [469] 20.50 18.83 19.17 25.00 20.17 25.75 20.42 39.00 64.08 28.25 28.75 31.33
## [481] 18.92 24.75 30.67 21.00 13.75 46.00 44.33 20.25 22.67 60.92 16.08 28.17
## [493] 39.17 30.00 22.83 22.50 28.58 45.17 41.58 57.08 55.75 43.25 25.33 24.58
## [505] 43.17 40.92 31.83 33.92 24.92 35.25 34.25 80.25 19.42 42.75 19.67 36.33
## [517] 30.08 44.25 23.58 23.92 33.17 48.33 76.75 51.33 34.75 38.58 22.42 41.92
## [529] 29.58 32.17 51.42 22.83 25.00 26.75 23.33 24.42 42.17 20.83 23.08 25.17
```

```
## [541] 43.08 35.75 59.50 21.00 21.92 65.17 20.33 32.25 30.17 25.17 39.17 39.08
## [553] 31.67 41.00 48.50 32.67 28.08 73.42 64.08 51.58 26.67 25.33 30.17 27.00
## [565] 34.17 38.67 25.75 46.08 21.50 20.50 29.50 29.83 20.08 23.42 29.58 16.17
## [577] 32.33 47.83 20.00 27.58 22.00 19.33 38.33 29.42 22.67 32.25 29.58 18.42
## [589] 22.17 22.67 18.83 21.58 23.75 36.08 29.25 19.58 22.92 27.25 38.75 32.42
## [601] 23.75 18.17 40.92 19.50 28.58 35.58 34.17 31.58 52.50 36.17 37.33 20.83
## [613] 24.08 25.58 35.17 48.08 15.83 22.50 21.50 23.58 21.08 25.67 38.92 15.75
## [625] 28.58 22.25 29.83 23.50 32.08 31.08 31.83 21.75 17.92 30.33 51.83 47.17
## [637] 25.83 50.25 37.33 41.58 30.58 19.42 17.92 20.08 19.50 27.83 17.08 36.42
## [649] 40.58 21.08 22.67 25.25 17.92 35.00
```

```
as.numeric(credit_card_data$V3)
```

```
##   [1]  0.000  4.460  0.500  1.540  5.625  4.000  1.040 11.585  0.500  4.915
##  [11]  0.830  1.835  6.000  6.040 10.500  4.415  0.875  5.875  0.250  8.585
##  [21] 11.250  1.000  8.000 14.500  6.500  0.585 13.000 18.500  8.500  1.040
##  [31] 14.790  9.790  7.585  5.125 10.750  1.500  1.585 11.750  0.585  9.415
##  [41]  9.170 15.000  1.415 13.915 28.000  6.750  2.040  4.460  1.540  0.665
##  [51]  0.500  1.000  2.040  2.500  3.000 11.625  0.000  0.500  4.500  3.000
##  [61] 12.250 16.165  0.790  0.835  4.250  1.540  1.000  0.375  6.500 25.125
##  [71]  7.500  5.000  0.500  7.000  5.290  6.500  0.585  1.165  9.750 19.000
##  [81]  1.500  0.500  0.625  4.000  2.210  4.500  1.500 12.750 15.500  1.375
##  [91]  1.500  3.540  0.625 11.000  1.000  1.750 16.500  5.000 12.000  4.000
## [101] 15.500  1.165 11.000  2.250  0.750  3.500  1.040 12.500  0.750  1.250
## [111]  1.125  7.000  6.500  7.040 10.335  6.210 12.500 12.500  6.665  9.000
## [121]  5.000  2.500 11.000  4.250  3.500  5.500  1.835  8.000  5.000  5.500
## [131]  6.500  0.540  2.750  9.500 13.500  3.750 16.000  0.290 11.000  1.665
## [141]  2.500  1.500  7.540  6.500 15.000  0.460 10.000  6.500  2.500  1.500
## [151] 11.500  3.040 15.000  2.000  0.080  2.000  2.000  1.710  1.750 16.500
## [161] 10.000  9.500  0.540  3.250  1.125  2.540  0.000 12.500 13.585 10.500
## [171]  1.500 12.500  8.665  9.250  8.170  0.290  2.335  3.000 19.500  5.665
## [181]  8.500  6.500  5.000  1.040  4.625 10.500  0.205  0.960  1.585  4.040
## [191]  5.040  3.165  7.625  2.040 10.040  0.500  1.000 10.250  2.125 12.000
## [201]  9.335  2.500  4.250  5.875  6.625 14.500 11.500  2.710  3.500  4.040
## [211] 11.500  9.625  9.250 12.540 11.000  9.000  9.540  5.500  4.000  5.500
## [221]  8.460  0.375 11.000  3.500  3.000  1.750 13.750 21.000  1.835  0.250
## [231]  7.500  4.625 10.125 10.000 25.085  5.000  2.000  3.040  8.500  0.210
## [241] 11.000 21.500  5.000 11.000 11.125 10.250 11.045  0.000  9.540  1.750
## [251]  2.335  0.000  1.335 10.500  0.585  0.085  1.210  6.000  1.540  0.165
## [261]  2.500  6.750 10.125  2.500  5.710  5.415 10.000  0.835  0.875  0.585
## [271]  4.460  1.250 12.625 10.000  0.580  0.415  1.750  0.750 10.000  0.540
## [281]  2.415  0.335  3.125  9.000  2.500  2.750 12.125  2.000  0.165  1.000
## [291]  2.875 13.665 26.335  2.750 10.290  1.290  3.000  4.500  1.750  0.210
## [301] 10.000  0.000  0.750  0.250 22.000  0.000  0.125  1.500  0.375  0.375
## [311]  0.205  1.250  1.085  1.085  2.500  2.500  2.500 12.500  2.500  0.750
## [321]  6.500  1.250  3.000  3.000  4.000  4.085  2.250  2.750  1.750  7.000
## [331]  4.500  0.540  1.500  2.000  0.585 11.500  1.585  2.000  0.165  1.335
## [341]  0.165  3.000  4.710  5.500  2.500  0.540  0.335  2.000  1.250  6.165
## [351]  1.710 11.500  0.750  2.500 13.000  4.585  1.540  0.500 14.500  0.835
## [361]  1.335  0.250  5.000  7.000  2.500  4.250 11.460  5.500  1.500  0.000
## [371] 14.585  0.170  7.000  5.000 12.500  0.540  1.250  1.625  2.040  0.585
## [381] 12.500  2.085  5.085  0.375  6.500  0.335  5.085  6.000  0.665  3.125
## [391]  0.250  2.835  3.000  0.790  1.500  0.460  1.250  0.125  1.790  0.705
## [401]  2.540  2.500  1.085  1.250  0.835  2.165  1.250  2.040  0.835  2.290
```

```
## [411]  0.335  3.000  3.165  1.540  1.000  2.710  0.000  0.585  0.165  1.250
## [421]  0.875  0.000  5.000  2.710  3.000  0.000  0.125  1.665  1.125  7.000
## [431]  1.625  4.250  1.500 18.125  0.750  5.500  0.290  2.415  0.875  0.500
## [441]  0.290  1.835  3.000  3.085  1.375  2.335  4.085 11.665  4.125  6.500
## [451]  4.000  0.415  2.000  0.835  2.500 11.500  0.500  3.165  9.500  0.415
## [461]  1.080 19.000  0.125 13.335  3.540  0.750  0.875  9.500 11.835  4.415
## [471]  9.500  0.875  9.250  0.500  7.000  5.000  0.165  5.125  3.750 19.500
## [481]  9.000  3.000 12.000  4.790  4.000  4.000  0.000  9.960  2.540  5.000
## [491]  0.750  0.375  1.710  5.290  3.000  8.500  1.665  1.500  1.750  0.335
## [501]  7.080 25.210  2.085  0.670  2.250  0.835  2.500  1.585  1.250  3.165
## [511]  1.750  5.500  1.500  3.000 10.000  3.790  1.040 11.000  0.460  1.500
## [521]  1.000 12.000 22.290 10.000 15.000  3.335 11.250  0.420  4.500  1.460
## [531]  0.040  2.290 12.330  1.125  1.500 12.335  5.040  3.000 11.500  2.875
## [541]  0.375  0.915  2.750  3.000  0.540 14.000 10.000  0.165  0.500  6.000
## [551]  1.625  6.000  0.830  0.040  4.250  9.000 15.000 17.750 20.000 15.000
## [561]  1.750  0.580  6.500  0.750  5.250  0.210  0.750  3.000  6.000  2.415
## [571]  0.460  1.250  0.250  0.585  1.750  0.040  3.500  4.165  1.250  3.250
## [581]  0.790 10.915  4.415  1.250  0.750 14.000  4.750 10.415  2.250  0.165
## [591]  0.000  0.790 12.000  2.540 13.000  0.665  1.250  0.290  1.500  2.165
## [601]  0.710  2.460  0.500  9.585  3.625  0.750  2.750  0.750  7.000  0.420
## [611]  2.665  8.500  9.000  0.335  3.750  3.750  7.625  0.415 11.500  0.830
## [621]  5.000  3.250  1.665  0.375  3.750  9.000  3.500  1.500  4.000  1.500
## [631]  0.040 11.750  0.540  0.500  2.040  5.835 12.835  0.835  2.500  1.040
## [641] 10.665  7.250 10.210  1.250  0.290  1.000  3.290  0.750  3.290 10.085
## [651]  0.750 13.500  0.205  3.375
```

```
as.numeric(credit_card_data$V4)
```

```
##   [1]  1.250  3.040  1.500  3.750  1.710  2.500  6.500  0.040  3.960  3.165
##  [11]  2.165  4.335  1.000  0.040  5.000  0.250  0.960  3.170  0.665  0.750
##  [21]  2.500  0.835  7.875  3.085  0.500  1.500  5.165 15.000  7.000  5.000
##  [31]  5.040  7.960  7.585  5.000  0.415  2.000  1.835  0.500  0.250 14.415
##  [41]  4.500  5.335  0.750  8.625 28.500  2.625  0.125  6.040  3.500  0.165
##  [51]  0.875  1.750  0.040  0.000  7.415  0.835  0.085  5.000  5.750  6.000
##  [61]  1.250  3.000  1.500  1.585  4.290  1.540  2.000  0.250  1.460  1.625
##  [71]  1.585 13.500 10.750  1.625  0.375  0.125  0.585  2.500  0.250  0.000
##  [81]  2.000  0.250  0.455  5.000  4.000  1.000  0.000  5.000  0.500  9.460
##  [91]  1.500  0.500  0.125  3.000  1.000  0.250  4.000  0.375  2.250  5.750
## [101]  0.000  0.500  4.500 10.000  0.795  3.500  0.500  0.875  1.000  1.375
## [111]  1.290 11.500  6.290 14.000  0.335  0.040  1.210  1.500  7.375  8.500
## [121]  7.500  2.500  2.500  3.250  0.835 13.000  2.250  6.500  2.500  5.500
## [131]  6.000  0.500  4.250  1.625  5.000  0.625  0.000  1.750  2.000  5.085
## [141]  2.750  2.375  8.000  4.000  5.500  0.415  4.000  4.250  1.085  5.500
## [151]  0.000  2.540  0.000  4.165  0.040  1.000  1.750  1.665  0.040 11.000
## [161]  1.750  1.000  0.040  9.000  1.500  0.250 15.000  8.000  8.500  1.335
## [171]  0.375  3.000  1.415  1.210  1.960 15.000  0.500  0.165  5.500  2.585
## [181] 12.500  3.500  5.000  2.500  1.625  3.000  5.125  2.500  3.085  8.500
## [191]  1.500  3.165 15.500  2.000  0.040  1.250  2.250  0.710  0.085 14.000
## [201]  5.665  4.500  6.500 10.000  5.500 18.000  3.500  5.250  3.500  7.000
## [211]  5.000  8.665  1.000  2.290 20.000  1.375  0.085  0.125  0.250  0.500
## [221]  2.460  2.000  0.665  0.625 13.875  4.500  5.750 10.000  2.085  0.000
## [231]  1.415  4.580  2.500  2.500  1.750 11.000  1.500  2.040 14.000  0.290
## [241]  0.290 20.000  5.000  1.000  0.460  1.085  2.000  0.500  0.040  0.165
## [251]  5.750  0.000  0.335  0.000  0.000  0.040  0.000  1.250  0.125  0.210
```

```
## [261]  1.250  0.040  0.125  2.250  0.540  0.290  0.165  0.085  0.250  0.125
## [271]  0.250  0.250  0.125  0.165  0.290  0.165  0.000  0.500  1.000  0.165
## [281]  0.125  0.000  3.040  4.000  1.750  0.665  3.335  6.500  0.250  0.500
## [291]  0.085  1.500  0.000  4.500  0.415  0.250  0.000  1.000  2.335  0.125
## [301]  0.415  0.250  0.500  0.250  0.000  0.000  1.500  1.500 10.000  0.375
## [311]  0.250  1.165  1.000  0.040  3.000  2.500  1.000  1.000  0.500  0.165
## [321]  1.000  0.500  2.000  0.750  3.000  0.040  0.500  0.000  0.500  0.000
## [331]  2.500  0.585  2.250  0.000  0.000  1.500  0.585  0.250  1.000  0.165
## [341]  0.040  0.165  0.000  5.500  0.085  0.000  0.290  0.165 13.875  0.165
## [351]  0.165  3.000  0.750  7.000  0.000  1.000  0.125  1.000  0.125  2.000
## [361]  0.125  4.000  2.250  0.165  4.500  5.000  1.585  1.500  0.540  0.500
## [371]  0.000  0.085  1.000  0.210  0.250  1.000  0.250  0.540  0.040  0.125
## [381]  1.250  0.085  0.290  0.290  3.085  0.750  1.085  0.000  1.000  0.085
## [391]  0.335  0.000  1.250  0.085  0.000  0.125  3.250  0.125  0.540  0.375
## [401]  0.000  1.000  1.500  1.750  1.165  1.500  0.250  0.250  0.040  0.290
## [411]  0.085  1.500  0.165  0.085  0.500  2.415  0.000  0.000  0.000  0.000
## [421]  0.375  1.000  0.000  0.125  0.040  0.000  0.165  0.000  0.000  0.500
## [431]  1.500  3.500  3.750  0.085  0.040  5.000  1.500  0.000  0.085  0.125
## [441]  0.000  0.000  2.790  2.500  0.040  0.750  0.415  0.085  0.040  0.125
## [451]  1.000  0.125  0.750  0.085 10.000  0.415  0.165  0.415  1.750  0.040
## [461]  1.165  0.040  0.085  0.040  0.000  2.750  4.625  6.500  6.000  3.000
## [471]  1.500  1.040  1.665  1.460  1.625  3.500  0.000  4.750  1.085  7.000
## [481]  0.750  1.835  2.000  2.250  1.750  0.000  2.500  0.000  2.585  4.000
## [491]  1.750  0.585  0.125  2.250  1.290  1.750  2.415  2.500  0.210  1.000
## [501]  6.750  0.210  2.750  1.750  0.750  0.000  7.500  0.000  0.000  3.750
## [511]  0.250  0.540  2.000  1.000  0.835  1.165  0.500  1.500  2.625  1.875
## [521]  0.750 16.000 12.750  0.000  5.375  4.000  0.750  0.210  7.500  1.085
## [531]  0.040  2.290  3.500  1.250  1.415  1.585 12.750  0.040  2.125  0.875
## [541]  0.375  0.750  1.750  1.085  0.040  0.000  1.000  3.250  1.750  1.000
## [551]  1.500  1.290  1.335  0.040  0.125  5.250  0.000  0.000 17.500  8.500
## [561]  1.000  0.290  3.125  4.250  0.085  0.085  0.250  2.375  2.500  2.000
## [571]  0.540  0.250  0.125  0.085  1.250  0.040  0.500  0.085  0.125  5.085
## [581]  0.290  0.585  0.125  0.250  1.585  0.000  2.000  0.125  0.125  2.250
## [591]  0.665  0.665  2.085  0.000  0.500  1.665  0.250  0.125  0.000  0.000
## [601]  0.250  0.960  0.500  0.790  0.250  1.500  2.500  3.500  3.000  0.290
## [611]  0.165  0.165  0.250  3.500  0.000  1.000  0.125  0.335  0.500  0.415
## [621]  0.000  2.290  0.250  1.000  0.250  0.085  0.165  0.875  1.500  0.040
## [631]  0.040  0.250  1.750  0.085  1.500  5.500  0.500  0.500  0.210  0.665
## [641]  0.085  0.040  0.000  0.000  0.290  3.000  0.335  0.585  3.500  1.250
## [651]  2.000  2.000  0.040  8.290
```

```
as.numeric(credit_card_data$V7)
```

```
##   [1]  1  6  0  5  0  0  0  0  0  0  0  0  0  0  7 10  3 10  0  7 17  0  6  1  3
##  [26]  2  9 17  3  6  5  8 15  0  5 11 12  2  2 11 12 11  1  6 40 11 23  3  0  0
##  [51]  0  0  0  0  0  0  0  0  0 11  4  9  2  1  1  1 11  3  7  1  0  0  0  0  1
##  [76]  0  0  1  0  1  0  0  0  0  0  0  0  0  0  0  0  0  0  0  2  0  0  2  2  2
## [101] 20  0  0  0  5  3  3  0  3  3  2  7 15  6  1  1 67 12  3  5  6 12  7  2  0
## [126]  1  1  6  6 12  0  3  6  6  2  9 15  8  1  9  6  3 14  7 14 11 14 12 11  3
## [151] 11  1 14  2  1  4  2  1  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  4 14
## [176] 20  4  3  7  7  9  1  7  5  2  0  0  0  6  7  8  3 12  3  9  1  0  2  1  8
## [201]  6  7 16 14  0 15  9  1  0  8  5  5  4  3  7  0  0  0  0  0  0  2  0  7  2
## [226]  4  0 13  5  0  1  0  6  0  3  0  2  1  1 11  6 11  6 11  1  0  0  0  0  0
## [251]  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  1  2  1  1
```

```
## [276]  1  0  0  1  1  2  1  2  1  2  1  2  1  0  0  0  0  0  0  0  0  0  0  0  0
## [301]  0  0  0  0  0  0  0  0  0  0 11  0  0  0  0  0  0  0  0  0  0  0  0  0  0
## [326]  0  0  0  0  0  0  3  1  0  0  0  0  0  2  0  0  0  0  0  0  0  0  2  1  0
## [351]  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
## [376]  0  0  0  0  0  0  0  0  0  0  0  0  0  1  1  4  5  1  0  0  0  0  0  0  2
## [401]  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  4  3  0  1  2 11  2  0  0
## [426]  0  0  0  1  0  0  0  0  0  0  0  0  0  4  0 10  1  1  2  3  0  0  0  0  0
## [451]  0  0  0  0  0  0  6  1 10  2  0  2  0  0  0  1  2 14  0  0  0  0  3  5  3
## [476] 10  1  2  1 16  2 19  1  1  2  0  0  0  0  4  5  4  5  5  1 10  0  0  0  0
## [501]  3  1  0  0  0  0  0  0  0  0  0  0  0  0  0  0 10  0  6  6  7  0  1 11  9
## [526] 14  4  6  2 16  0  7  6  0  0  0  0  0 11  0  8  4  5  8  1 11  4  1 11  3
## [551] 10  5  8  1  0  0  0  0  9  9  5  7  8  3  0  0  0  8  3 11  4  0  0  0  0
## [576]  0  0  0  0  2  1  2  0  2  1  2  1  0  0  0  0  0  0  0  0  0  0  1  0  0
## [601]  1  2  0  0  0  0  0  0  0  0  0  0  0  0  6  0  1  0  0  1  0  1  0  0  1
## [626]  0  0  0  0  0  0  0  1  0  0  0  0  0  0  0 12  1  0  0  0  0  0  0  0  0
## [651]  2  1  0  0
```

```
as.numeric(credit_card_data$V9)
```

```
##   [1] 202  43 280 100 120 360 164  80 180  52 128 260   0   0   0
##  [16] 320 396 120   0  96 200 300   0 120 145 100   0   0   0 500
##  [31] 168   0   0   0   0 434 583 300 260  30   0   0 240  70   0
##  [46]   0 455 311 216 100 491 280 400 239   0 160   0 320 711  80
##  [61] 200 250  80   0 120 520   0 260  80 515 420 980 400 160 160
##  [76] 443 160 180 140  94 368 288 200 280 188 240 100 112 120 200
##  [91]   0 171 180 268 167 164  80   0 120  75 152 280 120 176 140
## [106] 329 180 260 212 140 200 300   0 360  80 200 140 120   0 120
## [121]   0 410 100 274 280   0 100 375   0 408 350 200 204  40   0
## [136] 181   0 100  80 399 160   0   0   0   0 440   0  93  60   0
## [151]   0  70   0 181 280 140   0 395 393  21  29  60 440 102 431
## [166] 370   0  24   0 100 260  20 160  60  60   0  80 100   0 129
## [181] 181   0   0 300   0   0 400 510  80 195 144 380   0 370  60
## [196]   0   0  49  50   0 381 150 117 399 100   0  56 211 230 320
## [211]   0   0  80 156  22   0 100 228 160  80 164  80 100   0 519
## [226] 253 487   0 220   0  80   0 520  40 120  70 120 180  88  80
## [241] 121   0 470   0 100 320 136 144 200 132 292   0   0 154 145
## [256] 140 100 272 260 200 280 140 160 340 120  80 340 200 280 240
## [271]  80 108   0 120 340 200 160 320 120 272 220 160 200  70 720
## [286]  80 180   0 280 280 120 280   0 160  80 140 160 360 112 200
## [301]  80  60 320 280 450 500 232 150 300 300 380 120 280 170 200
## [316]   0 460 180 348 220   0 160 180 300 480 108 640 180 300   0
## [331] 200 180 100 276 100   0   0 221 320 168 380 120 160 100 100
## [346] 100 200 320 352 220 400  20 132 280 140 240 100 260   0 240
## [361] 440 420 141 160 200   0 100  60 120   0 178   0 600 550 720
## [376] 120   0   0 128 120   0 300 140 220  73 160 480   0 2000   0
## [391] 160 176   0 144 300 280 280 200 136 225 180 210 108 200 100
## [406] 120 110 180 120 200 180 180 160 356 200 320  45 350  62  92
## [421] 174   0  17  80 160  86 132 340  96   0   0 454   0 320 240
## [436] 210 160 120 254 360 200 200 280 160   0 180 120 320 140  60
## [451] 360   0  80 220 200   0 240 280   0 128  28   0 216 120 180
## [466] 333 520 240 340 240 120 160  40 312 200   0 232 420 371   0
## [481]  88   0 220  80 120 100   0   0   0   0 352  80 480  99 260
## [496]  80 440 140 160 252 100 760 360 400 560 130 523 320  80 680
## [511] 163   0 100   0 140 200 132   0 208 200 340 110   0   0   0
```

```
##  [526]  383    0  220  330  120    0  140  400    0  422  120   92  100  290  360
##  [541]  300    0   60  160  840    0   50  432   32    0  186  108  303  560  225
##  [556]  154    0    0    0    0  160   96  330  312  290  280  349  396   80  200
##  [571]  380  224  200  180  280    0  232  520  140  369  420  200  160  400  400
##  [586]  160  460  120  160    0  160  160   80    0  228  220  120  272   76  120
##  [601]  240  160  130   80  100  231  232  320    0  309    0    0    0  340    0
##  [616]  100    0  144  100  200    0  416    0  120   40    0  216  160  120  160
##  [631]    0  180   80  252  120  465    0  240  260  240  129  100    0    0  280
##  [646]  176  140  240  400  260  200  200  280    0
```

```
as.numeric(credit_card_data$V10)
```

```
##   [1]     0   560   824     3     0     0 31285  1349   314  1442
##  [11]     0   200     0  2690     0     0     0   245     0     0
##  [21]  1208     0  1260    11     0     0     0     0     0 10000
##  [31]     0     0  5000  4000   560    35   713   551   500   300
##  [41]   221  2283   100     0    15   284  1236   300     0     0
##  [51]     0     0  5800   200     0   300     0     0     0     0
##  [61]     0   730   400     0     0 50000   456 15108  2954   500
##  [71]     0     0     0     2     0     0     0    20     0     0
##  [81]     0     0     0     0     0     0    27     0     0   100
##  [91]   225     0     1     0   500   400     0    38     5     0
## [101]   130     0     0     0     5     0   147     0     0   210
## [111]     0     5 11202  1332    50   300   258   567     0     0
## [121]  1000  2510   809   610     0     0   150 51100   367  1000
## [131]     0  1000     0   600  5000     0   247   375   278   827
## [141]  2072   582  2300  3065  2200     6  1602     0  2184     0
## [151]     0     0  3376     0  2000  7544    15    20     0 10561
## [161]   837   400 11177   639     0     0     0  2028     0     0
## [171]  1065     0   150   540   158 15000     0     6  3000  3257
## [181]  1655   500  3065  1430     0     0     0   600     0     0
## [191]     7     0   790   560   396   678   300     0  1187  6590
## [201]   168  1270  1210     0     0  1000   742     0     0     0
## [211]  8851     0   500     0     0     0     0     0     0     0
## [221]     0     0     0  7059  1704   857   500  6700  2503     0
## [231]  9800     0   196     0    14     0   300 18027  2000    99
## [241]     0  1200     0  3000     0    13     0     0  1000     0
## [251]     0     0   120    32     0   722     0     0     0    40
## [261]     0     0     0     0     0   484     0     0   204     1
## [271]     0    98  5552     1  2803     1     0     0     1   444
## [281]     1   126     4     6     0    21   173    10     0     0
## [291]     0     1     0    25     0     0    20     6     6     1
## [301]    42     0     0   204 100000     1   113     8     0    44
## [311]  2732     0    13   179     0     2    16  1062     0   251
## [321]   228     0     0    67     0   100  4000     0     2    12
## [331]  1210     0     3     1     0  4000     0     0     1     0
## [341]     0     0     0     0  4208     0     0  1300   112  1000
## [351]     0    16     2     0  1110     0     0     0   286     0
## [361]  4500     0     0     0   456     4  1212     0    67     0
## [371]     0     0     0     0     0     1   195     0     1    87
## [381]    17     0   184   140     0     0     0     0     2     6
## [391]     8   146    22     0     0    55     0    70     1   500
## [401]    60     0     7     0     0     0     0    50     5     3
## [411]     0     4  1058     0     0     0     1   769    27   300
```

```
## [421]    3    0    1    0   40    0    0    1   19    0
## [431]  316   50  350 3552    0  687    0    0 1950    0
## [441]   18   53   10   41   33    0    0    5  100  100
## [451] 1000   44    0    5    0    0   35   80   10    6
## [461]    0  351 2100  475    1  892 2000 4607    0    0
## [471] 2206 5860   28    0 1391    0  100    7    0 5000
## [481]  591  500   19  300 1000  960    0    0    0   99
## [491]  690    0    0  500  800  990    0    0    0 2197
## [501]   50   90    1    0    0    1    0    0    0    0
## [511]    0  340   20  200    0    0   28    0  347  327
## [521] 4071    0  109 1249  134 1344  321  948    0 2079
## [531] 3000 2384  458 5298  200    0    0    0  284    0
## [541]  162 1583   58    1   59 1400 1465 8000  540    0
## [551] 4700 1097 3290    0    0    0 13212    0 1000    0
## [561] 5777 5124 1200  150    6    0   23 4159  918 3000
## [571]  500    0    0    0    0    0    0    0    4    1
## [581]  283    7    0  108    9    1   68  375   10    0
## [591]    1    0    0 1000    0    5  809  108    0    0
## [601]    4  587    0  350    0    0  200    0    0    2
## [611]  501  351    0    0  200    2  160    0   68   11
## [621]    0   21  390   18  154    0    0    0    0    0
## [631]    0    0    5    0    1  150    2  117  246  237
## [641]    3    1   50    0  364  537    2    3    0    0
## [651]  394    1  750    0
```

This is the beginning of the modeling code, specifically where the vanilla dot ksvm model was created.

```
#the appropriate seed was set to provide reproducibility
set.seed(234)

#a support vector machine model was created with a "vanilla dot"
#kernel using the ksvm function from the kernlab package
credit_card_model <- ksvm(as.matrix(credit_card_data[,1:10]),
as.factor(credit_card_data[,11]),type= 'C-svc',
kernel= "vanilladot", C=10, scaled=TRUE)
```

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

```
#The value of c that produces the best classification rate was
#determined from a range of values between 10 and 1000
for (num in seq(10, 1000, 10)) {

  # the coefficients for each of the predictors were determined
a <- colSums(credit_card_model@xmatrix[[1]] * credit_card_model@coef[[1]])

#the intercept, a0, was calculated
a0 <- -credit_card_model@b

# see what the model predicts
predictions <- predict(credit_card_model,credit_card_data[,1:10])

# see the fraction of classifications that match the model predictions
classification_rate = sum(predictions == credit_card_data[,11]) / nrow(credit_card_data)
```

```
print(paste(num, "=", classification_rate))
}
```

```
## [1] "10 = 0.863914373088685"
## [1] "20 = 0.863914373088685"
## [1] "30 = 0.863914373088685"
## [1] "40 = 0.863914373088685"
## [1] "50 = 0.863914373088685"
## [1] "60 = 0.863914373088685"
## [1] "70 = 0.863914373088685"
## [1] "80 = 0.863914373088685"
## [1] "90 = 0.863914373088685"
## [1] "100 = 0.863914373088685"
## [1] "110 = 0.863914373088685"
## [1] "120 = 0.863914373088685"
## [1] "130 = 0.863914373088685"
## [1] "140 = 0.863914373088685"
## [1] "150 = 0.863914373088685"
## [1] "160 = 0.863914373088685"
## [1] "170 = 0.863914373088685"
## [1] "180 = 0.863914373088685"
## [1] "190 = 0.863914373088685"
## [1] "200 = 0.863914373088685"
## [1] "210 = 0.863914373088685"
## [1] "220 = 0.863914373088685"
## [1] "230 = 0.863914373088685"
## [1] "240 = 0.863914373088685"
## [1] "250 = 0.863914373088685"
## [1] "260 = 0.863914373088685"
## [1] "270 = 0.863914373088685"
## [1] "280 = 0.863914373088685"
## [1] "290 = 0.863914373088685"
## [1] "300 = 0.863914373088685"
## [1] "310 = 0.863914373088685"
## [1] "320 = 0.863914373088685"
## [1] "330 = 0.863914373088685"
## [1] "340 = 0.863914373088685"
## [1] "350 = 0.863914373088685"
## [1] "360 = 0.863914373088685"
## [1] "370 = 0.863914373088685"
## [1] "380 = 0.863914373088685"
## [1] "390 = 0.863914373088685"
## [1] "400 = 0.863914373088685"
## [1] "410 = 0.863914373088685"
## [1] "420 = 0.863914373088685"
## [1] "430 = 0.863914373088685"
## [1] "440 = 0.863914373088685"
## [1] "450 = 0.863914373088685"
## [1] "460 = 0.863914373088685"
## [1] "470 = 0.863914373088685"
## [1] "480 = 0.863914373088685"
## [1] "490 = 0.863914373088685"
## [1] "500 = 0.863914373088685"
```

```
## [1] "510 = 0.863914373088685"
## [1] "520 = 0.863914373088685"
## [1] "530 = 0.863914373088685"
## [1] "540 = 0.863914373088685"
## [1] "550 = 0.863914373088685"
## [1] "560 = 0.863914373088685"
## [1] "570 = 0.863914373088685"
## [1] "580 = 0.863914373088685"
## [1] "590 = 0.863914373088685"
## [1] "600 = 0.863914373088685"
## [1] "610 = 0.863914373088685"
## [1] "620 = 0.863914373088685"
## [1] "630 = 0.863914373088685"
## [1] "640 = 0.863914373088685"
## [1] "650 = 0.863914373088685"
## [1] "660 = 0.863914373088685"
## [1] "670 = 0.863914373088685"
## [1] "680 = 0.863914373088685"
## [1] "690 = 0.863914373088685"
## [1] "700 = 0.863914373088685"
## [1] "710 = 0.863914373088685"
## [1] "720 = 0.863914373088685"
## [1] "730 = 0.863914373088685"
## [1] "740 = 0.863914373088685"
## [1] "750 = 0.863914373088685"
## [1] "760 = 0.863914373088685"
## [1] "770 = 0.863914373088685"
## [1] "780 = 0.863914373088685"
## [1] "790 = 0.863914373088685"
## [1] "800 = 0.863914373088685"
## [1] "810 = 0.863914373088685"
## [1] "820 = 0.863914373088685"
## [1] "830 = 0.863914373088685"
## [1] "840 = 0.863914373088685"
## [1] "850 = 0.863914373088685"
## [1] "860 = 0.863914373088685"
## [1] "870 = 0.863914373088685"
## [1] "880 = 0.863914373088685"
## [1] "890 = 0.863914373088685"
## [1] "900 = 0.863914373088685"
## [1] "910 = 0.863914373088685"
## [1] "920 = 0.863914373088685"
## [1] "930 = 0.863914373088685"
## [1] "940 = 0.863914373088685"
## [1] "950 = 0.863914373088685"
## [1] "960 = 0.863914373088685"
## [1] "970 = 0.863914373088685"
## [1] "980 = 0.863914373088685"
## [1] "990 = 0.863914373088685"
## [1] "1000 = 0.863914373088685"
```

```
#print out each of the parameter values for the model
a
```

```
##            V1            V2            V3            V4            V5
## -0.0009033671 -0.0007891036 -0.0016972133  0.0026113629  1.0050221405
##            V6            V7            V8            V9           V10
## -0.0028363016 -0.0001569285 -0.0003925964 -0.0012784443  0.1064387167
```

```
a0
```

```
## [1] 0.08157559
```

Question 2.2.2 #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.'

I tried two other models with different kernels than "vanilla dot." A model with a radial basis kernel (rbfdot) and one using a polynomial kernel (polydot) were tested to see if either could produce a classification rate better than 86.39%. The radial basis kernel was able to get a higher classification rate of 90.97% at a value for C of 10, a nearly 5% improvement. With the polynomial kernel, the classification rate and model parameters were both unchanged.

The resulting equation from the radial basis kernel SVM model is as follows:

V11 = 0.445 - 3.008 *V1* - *18.080*V2 + 3.814 *V3* + *25.613*V4 + 30.550 *V5* - *12.609*V6 + 14.401 *V7* - *10.173*V8 - 32.763 *V9* + *35.589*V10

Due to the higher classification rate and the fact that none of the resulting variable coefficients are close to zero, I would recommend choosing the radial basis kernel SVM model in further analyses.

Note: I experimented with values of c up to 10000 but, for the sake of reducing the code output, have only included c values up to 1000.

```
#a support vector machine model was created with a "rbfdot" kernel
#using the ksvm function from the kernlab package
credit_card_model <- ksvm(as.matrix(credit_card_data[,1:10]),
as.factor(credit_card_data[,11]),type= 'C-svc',
kernel= "rbfdot", C=10 , scaled=TRUE)

#The value of c that produces the best classification rate was
#determined from a range of values between 10 and 1000
for (num in seq(10, 1000, 10)) {
# the coefficients for each of the predictors were determined
a <- colSums(credit_card_model@xmatrix[[1]] * credit_card_model@coef[[1]])

#the intercept, a0, was calculated
a0 <- -credit_card_model@b

# see what the model predicts
predictions <- predict(credit_card_model,credit_card_data[,1:10])

# see the fraction of classifications that match the model predictions
classification_rate = sum(predictions == credit_card_data[,11]) / nrow(credit_card_data)

print(paste(num, "=", classification_rate))
}
```

```
## [1] "10 = 0.914373088685015"
```

```
## [1] "20 = 0.914373088685015"
## [1] "30 = 0.914373088685015"
## [1] "40 = 0.914373088685015"
## [1] "50 = 0.914373088685015"
## [1] "60 = 0.914373088685015"
## [1] "70 = 0.914373088685015"
## [1] "80 = 0.914373088685015"
## [1] "90 = 0.914373088685015"
## [1] "100 = 0.914373088685015"
## [1] "110 = 0.914373088685015"
## [1] "120 = 0.914373088685015"
## [1] "130 = 0.914373088685015"
## [1] "140 = 0.914373088685015"
## [1] "150 = 0.914373088685015"
## [1] "160 = 0.914373088685015"
## [1] "170 = 0.914373088685015"
## [1] "180 = 0.914373088685015"
## [1] "190 = 0.914373088685015"
## [1] "200 = 0.914373088685015"
## [1] "210 = 0.914373088685015"
## [1] "220 = 0.914373088685015"
## [1] "230 = 0.914373088685015"
## [1] "240 = 0.914373088685015"
## [1] "250 = 0.914373088685015"
## [1] "260 = 0.914373088685015"
## [1] "270 = 0.914373088685015"
## [1] "280 = 0.914373088685015"
## [1] "290 = 0.914373088685015"
## [1] "300 = 0.914373088685015"
## [1] "310 = 0.914373088685015"
## [1] "320 = 0.914373088685015"
## [1] "330 = 0.914373088685015"
## [1] "340 = 0.914373088685015"
## [1] "350 = 0.914373088685015"
## [1] "360 = 0.914373088685015"
## [1] "370 = 0.914373088685015"
## [1] "380 = 0.914373088685015"
## [1] "390 = 0.914373088685015"
## [1] "400 = 0.914373088685015"
## [1] "410 = 0.914373088685015"
## [1] "420 = 0.914373088685015"
## [1] "430 = 0.914373088685015"
## [1] "440 = 0.914373088685015"
## [1] "450 = 0.914373088685015"
## [1] "460 = 0.914373088685015"
## [1] "470 = 0.914373088685015"
## [1] "480 = 0.914373088685015"
## [1] "490 = 0.914373088685015"
## [1] "500 = 0.914373088685015"
## [1] "510 = 0.914373088685015"
## [1] "520 = 0.914373088685015"
## [1] "530 = 0.914373088685015"
## [1] "540 = 0.914373088685015"
## [1] "550 = 0.914373088685015"
```

```
## [1] "560 = 0.914373088685015"
## [1] "570 = 0.914373088685015"
## [1] "580 = 0.914373088685015"
## [1] "590 = 0.914373088685015"
## [1] "600 = 0.914373088685015"
## [1] "610 = 0.914373088685015"
## [1] "620 = 0.914373088685015"
## [1] "630 = 0.914373088685015"
## [1] "640 = 0.914373088685015"
## [1] "650 = 0.914373088685015"
## [1] "660 = 0.914373088685015"
## [1] "670 = 0.914373088685015"
## [1] "680 = 0.914373088685015"
## [1] "690 = 0.914373088685015"
## [1] "700 = 0.914373088685015"
## [1] "710 = 0.914373088685015"
## [1] "720 = 0.914373088685015"
## [1] "730 = 0.914373088685015"
## [1] "740 = 0.914373088685015"
## [1] "750 = 0.914373088685015"
## [1] "760 = 0.914373088685015"
## [1] "770 = 0.914373088685015"
## [1] "780 = 0.914373088685015"
## [1] "790 = 0.914373088685015"
## [1] "800 = 0.914373088685015"
## [1] "810 = 0.914373088685015"
## [1] "820 = 0.914373088685015"
## [1] "830 = 0.914373088685015"
## [1] "840 = 0.914373088685015"
## [1] "850 = 0.914373088685015"
## [1] "860 = 0.914373088685015"
## [1] "870 = 0.914373088685015"
## [1] "880 = 0.914373088685015"
## [1] "890 = 0.914373088685015"
## [1] "900 = 0.914373088685015"
## [1] "910 = 0.914373088685015"
## [1] "920 = 0.914373088685015"
## [1] "930 = 0.914373088685015"
## [1] "940 = 0.914373088685015"
## [1] "950 = 0.914373088685015"
## [1] "960 = 0.914373088685015"
## [1] "970 = 0.914373088685015"
## [1] "980 = 0.914373088685015"
## [1] "990 = 0.914373088685015"
## [1] "1000 = 0.914373088685015"
```

```
#print out each of the parameter values for the model
a
```

```
##         V1         V2         V3         V4         V5         V6         V7
##  -3.002540 -17.981002   3.773051  25.802859  30.828269 -12.607206  14.752166
##         V8         V9        V10
## -10.054629 -32.820604  35.895918
```

```
a0
```

```
## [1] 0.4442776
```

This is the code and output for the "polydot" kernel model.

Note: I experimented with values of c up to 10000 but, for the sake of reducing the code output, have only included c values up to 1000.

```
#a support vector machine model was created with a "polydot" kernel
#using the ksvm function from the kernlab package
credit_card_model <- ksvm(as.matrix(credit_card_data[,1:10]),
as.factor(credit_card_data[,11]),type= 'C-svc',
kernel= "polydot", C=10 , scaled=TRUE)
```

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

```
#The value of c that produces the best classification rate
#was determined from a range of values between 10 and 1000
for (num in seq(10, 1000, 10)) {
# the coefficients for each of the predictors were determined
a <- colSums(credit_card_model@xmatrix[[1]] * credit_card_model@coef[[1]])

#the intercept, a0, was calculated
a0 <- -credit_card_model@b

# see what the model predicts
predictions <- predict(credit_card_model,credit_card_data[,1:10])

# see the fraction of classifications that match the model predictions
classification_rate = sum(predictions == credit_card_data[,11]) / nrow(credit_card_data)

print(paste(num, "=", classification_rate))
}
```

```
## [1] "10 = 0.863914373088685"
## [1] "20 = 0.863914373088685"
## [1] "30 = 0.863914373088685"
## [1] "40 = 0.863914373088685"
## [1] "50 = 0.863914373088685"
## [1] "60 = 0.863914373088685"
## [1] "70 = 0.863914373088685"
## [1] "80 = 0.863914373088685"
## [1] "90 = 0.863914373088685"
## [1] "100 = 0.863914373088685"
## [1] "110 = 0.863914373088685"
## [1] "120 = 0.863914373088685"
## [1] "130 = 0.863914373088685"
## [1] "140 = 0.863914373088685"
## [1] "150 = 0.863914373088685"
## [1] "160 = 0.863914373088685"
## [1] "170 = 0.863914373088685"
```

```
## [1] "180 = 0.863914373088685"
## [1] "190 = 0.863914373088685"
## [1] "200 = 0.863914373088685"
## [1] "210 = 0.863914373088685"
## [1] "220 = 0.863914373088685"
## [1] "230 = 0.863914373088685"
## [1] "240 = 0.863914373088685"
## [1] "250 = 0.863914373088685"
## [1] "260 = 0.863914373088685"
## [1] "270 = 0.863914373088685"
## [1] "280 = 0.863914373088685"
## [1] "290 = 0.863914373088685"
## [1] "300 = 0.863914373088685"
## [1] "310 = 0.863914373088685"
## [1] "320 = 0.863914373088685"
## [1] "330 = 0.863914373088685"
## [1] "340 = 0.863914373088685"
## [1] "350 = 0.863914373088685"
## [1] "360 = 0.863914373088685"
## [1] "370 = 0.863914373088685"
## [1] "380 = 0.863914373088685"
## [1] "390 = 0.863914373088685"
## [1] "400 = 0.863914373088685"
## [1] "410 = 0.863914373088685"
## [1] "420 = 0.863914373088685"
## [1] "430 = 0.863914373088685"
## [1] "440 = 0.863914373088685"
## [1] "450 = 0.863914373088685"
## [1] "460 = 0.863914373088685"
## [1] "470 = 0.863914373088685"
## [1] "480 = 0.863914373088685"
## [1] "490 = 0.863914373088685"
## [1] "500 = 0.863914373088685"
## [1] "510 = 0.863914373088685"
## [1] "520 = 0.863914373088685"
## [1] "530 = 0.863914373088685"
## [1] "540 = 0.863914373088685"
## [1] "550 = 0.863914373088685"
## [1] "560 = 0.863914373088685"
## [1] "570 = 0.863914373088685"
## [1] "580 = 0.863914373088685"
## [1] "590 = 0.863914373088685"
## [1] "600 = 0.863914373088685"
## [1] "610 = 0.863914373088685"
## [1] "620 = 0.863914373088685"
## [1] "630 = 0.863914373088685"
## [1] "640 = 0.863914373088685"
## [1] "650 = 0.863914373088685"
## [1] "660 = 0.863914373088685"
## [1] "670 = 0.863914373088685"
## [1] "680 = 0.863914373088685"
## [1] "690 = 0.863914373088685"
## [1] "700 = 0.863914373088685"
## [1] "710 = 0.863914373088685"
```

```
## [1] "720 = 0.863914373088685"
## [1] "730 = 0.863914373088685"
## [1] "740 = 0.863914373088685"
## [1] "750 = 0.863914373088685"
## [1] "760 = 0.863914373088685"
## [1] "770 = 0.863914373088685"
## [1] "780 = 0.863914373088685"
## [1] "790 = 0.863914373088685"
## [1] "800 = 0.863914373088685"
## [1] "810 = 0.863914373088685"
## [1] "820 = 0.863914373088685"
## [1] "830 = 0.863914373088685"
## [1] "840 = 0.863914373088685"
## [1] "850 = 0.863914373088685"
## [1] "860 = 0.863914373088685"
## [1] "870 = 0.863914373088685"
## [1] "880 = 0.863914373088685"
## [1] "890 = 0.863914373088685"
## [1] "900 = 0.863914373088685"
## [1] "910 = 0.863914373088685"
## [1] "920 = 0.863914373088685"
## [1] "930 = 0.863914373088685"
## [1] "940 = 0.863914373088685"
## [1] "950 = 0.863914373088685"
## [1] "960 = 0.863914373088685"
## [1] "970 = 0.863914373088685"
## [1] "980 = 0.863914373088685"
## [1] "990 = 0.863914373088685"
## [1] "1000 = 0.863914373088685"
```

```
#print out each of the parameter values for the model
a
```

```
##            V1            V2            V3            V4            V5
## -0.0009647181 -0.0010953376 -0.0015706841  0.0026559397  1.0050171641
##            V6            V7            V8            V9           V10
## -0.0028674084 -0.0002663424 -0.0005284851 -0.0013955524  0.1064078260
```

```
a0
```

```
## [1] 0.0815459
```

Question 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).

The k-nearest neighbors model below produced its highest classification rate at k=12 and k=15. Both values produced a classification rate of 85.32%. Notably, at k=5, a slightly smaller classification rate of 85.16% was produced. However, with higher values of k, the model can better classify outliers and have lower variability. But, we do not want to value for k to be so high that the model cannot be generalized to new data.

As a result, I would choose k=12 as the optimal value in this case. It is not the highest value of k with a classification rate over 85%, but it is also large enough that variability is lower.

```
#create an empty prediction array of zeroes for the kknn
#model to populate with its predictions for each data point
prediction <- c(rep(0, nrow(credit_card_data)))

#the k value that produces the best classification rate
#was determined from a range of values between 1 and 25
for (k in seq(1,25, 1)) {

#each of the data points were iterated through to prevent R
#from using i when determining the closest points to i itself
  for (i in 1:nrow(credit_card_data)) {

#a weighted k-nearest neighbors model was created using
#the kknn function from the kknn library with the data scaled
    kknn_model = kknn(V11 ~ V1 + V2 + V3 + V4 + V5 + V6 + V7 + V8 + V9 + V10,
    train = credit_card_data[-i,],test = credit_card_data[i, ],
k = k, scale = TRUE)

#an is-else loop was used to convert the outputted probabilities
#to classifications of one's and zero's
   if (fitted(kknn_model) >= 0.5) {
     prediction[i] = 1
   } else if (fitted(kknn_model) < 0.5) {
     prediction[i] = 0
   }

#see the fraction of classifications that match the model
#predictions for each value of k
   classification <- sum(prediction == credit_card_data[,11]) / nrow(credit_card_data)
  }

  print(paste(k, ":", classification))
}
```

```
## [1] "1 : 0.814984709480122"
## [1] "2 : 0.814984709480122"
## [1] "3 : 0.814984709480122"
## [1] "4 : 0.814984709480122"
## [1] "5 : 0.851681957186544"
## [1] "6 : 0.845565749235474"
## [1] "7 : 0.847094801223242"
## [1] "8 : 0.848623853211009"
## [1] "9 : 0.847094801223242"
## [1] "10 : 0.850152905198777"
## [1] "11 : 0.851681957186544"
## [1] "12 : 0.853211009174312"
## [1] "13 : 0.851681957186544"
## [1] "14 : 0.851681957186544"
## [1] "15 : 0.853211009174312"
## [1] "16 : 0.851681957186544"
## [1] "17 : 0.851681957186544"
## [1] "18 : 0.851681957186544"
## [1] "19 : 0.850152905198777"
```

```
## [1] "20 : 0.850152905198777"
## [1] "21 : 0.848623853211009"
## [1] "22 : 0.847094801223242"
## [1] "23 : 0.844036697247706"
## [1] "24 : 0.845565749235474"
## [1] "25 : 0.845565749235474"
```