Exercise 1: The idea of credit scorning is that you try to classify people into a group that will pay the money back and a group that won't. We'll use a version of a well-known dataset called Kredit, which contained data from 1000 germans. This version we'll use is from kaggle.

a. Load the csv german_credit_data.

Solution:
Reading in the data we find that the checking and saving account variables contain all the missing values. Simply removing all the rows with missing values reduces our number of observations to 522, which is about half. I am not necessary sure we should be throwing out so much of our data, for two variables which are not entirely indicative of an individuals credit responsibility. There are a few actions we can take to try and save this data. We could impute the medium value for these missing values, use a discriminant method that is robust to missing values like decision trees, or we could simply drop these categorical predictors. I ended up deciding to remove the variables.
Code:

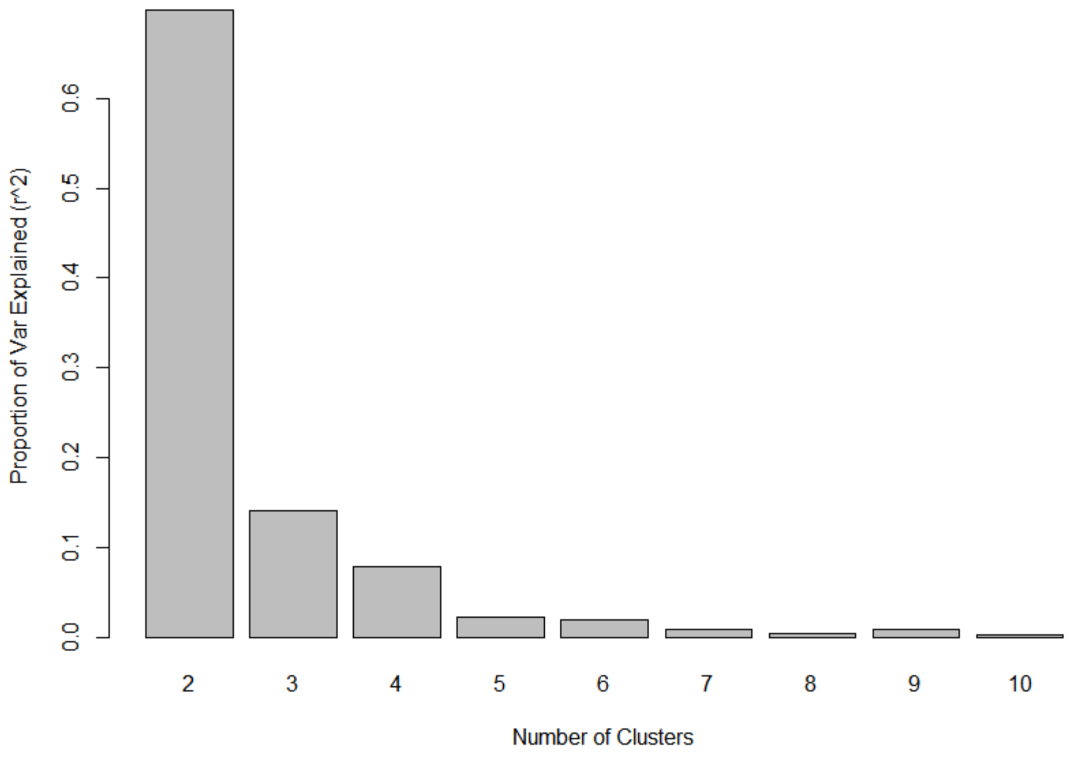
```
## Reading in the data.
data <- read.csv('german_credit_data.csv', header = TRUE)

## Note that checking and saving account variable contains a lot of NAs.
length(na.omit(data$Saving.accounts))
# [1] 817
length(na.omit(data$Checking.account))
# [1] 606
length(na.omit(data)[,1])

## Dropping these two columns removes all NAs. Perhaps we
## can explore a method more robust to NA, like decision trees
## and bring that data in.
data <- data[, -c(1,7,8)]

## Turning categorical variables into factors.
data$Sex <- as.numeric(as.factor(data$Sex))
data$Housing <- as.numeric(as.factor(data$Housing))
data$Purpose <- as.numeric(as.factor(data$Purpose))
```

b. Try a k-nearest neighbors classification. Select a value of k that you think is reasonable, then split the data into a training set and a testing set, then look at how well it classified the test data.

Solution:
From the data set we know that there are likely only two groups represented in the

data. We can verify this by simulating kmeans clusters and computing the rsquared between them. Doing so we found that $k = 2$ is explaining a majority of the variance. Using a 80/20 split for training testing data we found that the kmeans clustering performed rather poorly, misclassifying 86 of the 200 testing data achieving an accuracy of 57%.

Figure 1: Proportion of rSquared for each K.



Code:

```
## Simulating Clusters and Computing rSquared.
rsquared <- rep(NA, 10)
for (i in 1:10){
  tmp <- kmeans(data[,-1], centers = i)
  rsquared[i] <- tmp$betweenss/tmp$totss
}
plot(rsquared, type="l", lwd=2, xlab = '# of Clusters')
barplot((rsquared[2:10] - rsquared[1:9]), names.arg = seq(2,10),
        ylab = 'Proportion of Var Explained (r^2)',
        xlab = 'Number of Clusters' )
```

```
## Split the data and cluster using kmeans
split <- sample(c(rep(0, 0.8 * nrow(data)), rep(1, 0.2 * nrow(data))))
test_data <- data[split == 1,]
train_data <- data[split == 0,]

kmeansClustering <- knn(train = train_data[,-1],test = test_data[,-1], cl = train_c
table(test_data[,1], kmeansClustering)

## kmeansClustering
##     0  1
## 0  24  41
## 1  45  90
```