Homework10

Richard Albright

3/17/2019

Question 14.1

The breast cancer data set breast-cancer-wisconsin.data.txt from http://archive.ics.uci.edu/ml/machine-learning-databases/breast-cancer-wisconsin/ (description at http://archive.ics.uci.edu/ml/datasets/Breast+Cancer+Wisconsin+%28Original%29) has missing values.

Read in the CSV.

```
rawdata <-
  read.table(
    "/Users/ralbright/Dropbox/ISYE6501/week10/homework/breast-cancer-wisconsin.data.txt",
    header=FALSE,
    sep=","
  )
summary(rawdata)
```

```
##        V1                 V2               V3               V4
##  Min.   :   61634   Min.   : 1.000   Min.   : 1.000   Min.   : 1.000
##  1st Qu.:  870688   1st Qu.: 2.000   1st Qu.: 1.000   1st Qu.: 1.000
##  Median : 1171710   Median : 4.000   Median : 1.000   Median : 1.000
##  Mean   : 1071704   Mean   : 4.418   Mean   : 3.134   Mean   : 3.207
##  3rd Qu.: 1238298   3rd Qu.: 6.000   3rd Qu.: 5.000   3rd Qu.: 5.000
##  Max.   :13454352   Max.   :10.000   Max.   :10.000   Max.   :10.000
##
##        V5               V6               V7            V8
##  Min.   : 1.000   Min.   : 1.000   1      :402   Min.   : 1.000
##  1st Qu.: 1.000   1st Qu.: 2.000   10     :132   1st Qu.: 2.000
##  Median : 1.000   Median : 2.000   2      : 30   Median : 3.000
##  Mean   : 2.807   Mean   : 3.216   5      : 30   Mean   : 3.438
##  3rd Qu.: 4.000   3rd Qu.: 4.000   3      : 28   3rd Qu.: 5.000
##  Max.   :10.000   Max.   :10.000   8      : 21   Max.   :10.000
##                                    (Other): 56
##        V9              V10              V11
##  Min.   : 1.000   Min.   : 1.000   Min.   :2.00
##  1st Qu.: 1.000   1st Qu.: 1.000   1st Qu.:2.00
##  Median : 1.000   Median : 1.000   Median :2.00
##  Mean   : 2.867   Mean   : 1.589   Mean   :2.69
##  3rd Qu.: 4.000   3rd Qu.: 1.000   3rd Qu.:4.00
##  Max.   :10.000   Max.   :10.000   Max.   :4.00
##
```

1. Use the mean/mode imputation method to impute values for the missing data.

From the summary we can tell that V7 has missing data because it is missing summary statistics. To get the missing indices use the which function to look for rows with "?". Assign a binary column to the rawdata

dataframe to indicate missing data. Then get the indices.

```
rawdata[,12] <- as.integer(rawdata[,7] == "?")
summary(rawdata)
```

```
##        V1                  V2               V3               V4
##  Min.   :   61634   Min.   : 1.000   Min.   : 1.000   Min.   : 1.000
##  1st Qu.:  870688   1st Qu.: 2.000   1st Qu.: 1.000   1st Qu.: 1.000
##  Median : 1171710   Median : 4.000   Median : 1.000   Median : 1.000
##  Mean   : 1071704   Mean   : 4.418   Mean   : 3.134   Mean   : 3.207
##  3rd Qu.: 1238298   3rd Qu.: 6.000   3rd Qu.: 5.000   3rd Qu.: 5.000
##  Max.   :13454352   Max.   :10.000   Max.   :10.000   Max.   :10.000
##
##        V5               V6               V7            V8
##  Min.   : 1.000   Min.   : 1.000   1      :402   Min.   : 1.000
##  1st Qu.: 1.000   1st Qu.: 2.000   10     :132   1st Qu.: 2.000
##  Median : 1.000   Median : 2.000   2      : 30   Median : 3.000
##  Mean   : 2.807   Mean   : 3.216   5      : 30   Mean   : 3.438
##  3rd Qu.: 4.000   3rd Qu.: 4.000   3      : 28   3rd Qu.: 5.000
##  Max.   :10.000   Max.   :10.000   8      : 21   Max.   :10.000
##                                    (Other): 56
##        V9               V10              V11            V12
##  Min.   : 1.000   Min.   : 1.000   Min.   :2.00   Min.   :0.00000
##  1st Qu.: 1.000   1st Qu.: 1.000   1st Qu.:2.00   1st Qu.:0.00000
##  Median : 1.000   Median : 1.000   Median :2.00   Median :0.00000
##  Mean   : 2.867   Mean   : 1.589   Mean   :2.69   Mean   :0.02289
##  3rd Qu.: 4.000   3rd Qu.: 1.000   3rd Qu.:4.00   3rd Qu.:0.00000
##  Max.   :10.000   Max.   :10.000   Max.   :4.00   Max.   :1.00000
##
```

```
miss_idx <- which(rawdata[,7] == "?")
miss_idx
```

```
##  [1]  24  41 140 146 159 165 236 250 276 293 295 298 316 322 412 618
```

The good values are needed too in order perform summary statistics to get the mean and mode. Because "?"" was in the column the data was read in as character data and needs converted to integer.

```
good_idx <- which(rawdata[,7] != "?")
data_v7 <- as.matrix(as.integer(rawdata[good_idx, 7]))
```

Round the resulting mean to integer since column V7 represents integer values from 1 to 10.

```
data_v7.mean <- as.integer(round(mean(data_v7)))
data_v7.mean
```

```
## [1] 3
```

Create a mode function since R does not come with one and get the mode from the good V7 column values.

```
getmode <- function(v) {
   uniqv <- unique(v)
   uniqv[which.max(tabulate(match(v, uniqv)))]
}
data_v7.mode <- getmode(data_v7)
data_v7.mode
```

```
## [1] 2
```

Create a copy of our raw data so we can impute the missing data with the mean.

```
data_mean <- copy(rawdata)
data_mean[miss_idx, 7] <- data_v7.mean
data_mean[, 7] <- sapply(data_mean[, 7], as.integer)
summary(data_mean)
```

```
##        V1                 V2               V3               V4
##  Min.   :   61634   Min.   : 1.000   Min.   : 1.000   Min.   : 1.000
##  1st Qu.:  870688   1st Qu.: 2.000   1st Qu.: 1.000   1st Qu.: 1.000
##  Median : 1171710   Median : 4.000   Median : 1.000   Median : 1.000
##  Mean   : 1071704   Mean   : 4.418   Mean   : 3.134   Mean   : 3.207
##  3rd Qu.: 1238298   3rd Qu.: 6.000   3rd Qu.: 5.000   3rd Qu.: 5.000
##  Max.   :13454352   Max.   :10.000   Max.   :10.000   Max.   :10.000
##        V5               V6               V7               V8
##  Min.   : 1.000   Min.   : 1.000   Min.   : 2.000   Min.   : 1.000
##  1st Qu.: 1.000   1st Qu.: 2.000   1st Qu.: 2.000   1st Qu.: 2.000
##  Median : 1.000   Median : 2.000   Median : 2.000   Median : 3.000
##  Mean   : 2.807   Mean   : 3.216   Mean   : 3.258   Mean   : 3.438
##  3rd Qu.: 4.000   3rd Qu.: 4.000   3rd Qu.: 3.000   3rd Qu.: 5.000
##  Max.   :10.000   Max.   :10.000   Max.   :11.000   Max.   :10.000
##        V9              V10              V11             V12
##  Min.   : 1.000   Min.   : 1.000   Min.   :2.00   Min.   :0.00000
##  1st Qu.: 1.000   1st Qu.: 1.000   1st Qu.:2.00   1st Qu.:0.00000
##  Median : 1.000   Median : 1.000   Median :2.00   Median :0.00000
##  Mean   : 2.867   Mean   : 1.589   Mean   :2.69   Mean   :0.02289
##  3rd Qu.: 4.000   3rd Qu.: 1.000   3rd Qu.:4.00   3rd Qu.:0.00000
##  Max.   :10.000   Max.   :10.000   Max.   :4.00   Max.   :1.00000
```

Do the same using the mode

```
data_mode <- copy(rawdata)
data_mode[miss_idx, 7] <- data_v7.mode
data_mode[, 7] <- sapply(data_mode[, 7], as.integer)
summary(data_mode)
```

```
##        V1                 V2               V3               V4
##  Min.   :   61634   Min.   : 1.000   Min.   : 1.000   Min.   : 1.000
##  1st Qu.:  870688   1st Qu.: 2.000   1st Qu.: 1.000   1st Qu.: 1.000
##  Median : 1171710   Median : 4.000   Median : 1.000   Median : 1.000
##  Mean   : 1071704   Mean   : 4.418   Mean   : 3.134   Mean   : 3.207
##  3rd Qu.: 1238298   3rd Qu.: 6.000   3rd Qu.: 5.000   3rd Qu.: 5.000
##  Max.   :13454352   Max.   :10.000   Max.   :10.000   Max.   :10.000
##        V5               V6               V7               V8
##  Min.   : 1.000   Min.   : 1.000   Min.   : 2.000   Min.   : 1.000
##  1st Qu.: 1.000   1st Qu.: 2.000   1st Qu.: 2.000   1st Qu.: 2.000
##  Median : 1.000   Median : 2.000   Median : 2.000   Median : 3.000
##  Mean   : 2.807   Mean   : 3.216   Mean   : 3.235   Mean   : 3.438
##  3rd Qu.: 4.000   3rd Qu.: 4.000   3rd Qu.: 3.000   3rd Qu.: 5.000
##  Max.   :10.000   Max.   :10.000   Max.   :11.000   Max.   :10.000
##        V9              V10              V11             V12
##  Min.   : 1.000   Min.   : 1.000   Min.   :2.00   Min.   :0.00000
##  1st Qu.: 1.000   1st Qu.: 1.000   1st Qu.:2.00   1st Qu.:0.00000
##  Median : 1.000   Median : 1.000   Median :2.00   Median :0.00000
##  Mean   : 2.867   Mean   : 1.589   Mean   :2.69   Mean   :0.02289
##  3rd Qu.: 4.000   3rd Qu.: 1.000   3rd Qu.:4.00   3rd Qu.:0.00000
```

```
##  Max.   :10.000   Max.   :10.000   Max.   :4.00   Max.   :1.00000
```

2. Use regression to impute values for the missing data.

```
data_regress1 = copy(rawdata[good_idx,])
data_regress2 = copy(rawdata[miss_idx,])
data_regress1[, 7] <- sapply(data_regress1[, 7], as.integer)
impute_model <- lm(V7~V2+V3+V4+V5+V6+V8+V9+V10, data=data_regress1)
predictions <- predict(impute_model)
regress_mean = round(mean(predictions))
data_regress2[,7] = regress_mean
data_regress = rbind(data_regress1, data_regress2)
summary(data_regress)
```

```
##        V1                 V2               V3               V4
##  Min.   :   61634   Min.   : 1.000   Min.   : 1.000   Min.   : 1.000
##  1st Qu.:  870688   1st Qu.: 2.000   1st Qu.: 1.000   1st Qu.: 1.000
##  Median : 1171710   Median : 4.000   Median : 1.000   Median : 1.000
##  Mean   : 1071704   Mean   : 4.418   Mean   : 3.134   Mean   : 3.207
##  3rd Qu.: 1238298   3rd Qu.: 6.000   3rd Qu.: 5.000   3rd Qu.: 5.000
##  Max.   :13454352   Max.   :10.000   Max.   :10.000   Max.   :10.000
##        V5               V6               V7               V8
##  Min.   : 1.000   Min.   : 1.000   Min.   : 2.000   Min.   : 1.000
##  1st Qu.: 1.000   1st Qu.: 2.000   1st Qu.: 2.000   1st Qu.: 2.000
##  Median : 1.000   Median : 2.000   Median : 2.000   Median : 3.000
##  Mean   : 2.807   Mean   : 3.216   Mean   : 3.212   Mean   : 3.438
##  3rd Qu.: 4.000   3rd Qu.: 4.000   3rd Qu.: 3.000   3rd Qu.: 5.000
##  Max.   :10.000   Max.   :10.000   Max.   :11.000   Max.   :10.000
##        V9              V10              V11             V12
##  Min.   : 1.000   Min.   : 1.000   Min.   :2.00   Min.   :0.00000
##  1st Qu.: 1.000   1st Qu.: 1.000   1st Qu.:2.00   1st Qu.:0.00000
##  Median : 1.000   Median : 1.000   Median :2.00   Median :0.00000
##  Mean   : 2.867   Mean   : 1.589   Mean   :2.69   Mean   :0.02289
##  3rd Qu.: 4.000   3rd Qu.: 1.000   3rd Qu.:4.00   3rd Qu.:0.00000
##  Max.   :10.000   Max.   :10.000   Max.   :4.00   Max.   :1.00000
```

3. Use regression with perturbation to impute values for the missing data.

```
data_regress1 = copy(rawdata[good_idx,])
data_regress2 = copy(rawdata[miss_idx,])
data_regress1[, 7] <- sapply(data_regress1[, 7], as.integer)
impute_model <- lm(V7~V2+V3+V4+V5+V6+V8+V9+V10, data=data_regress1)
predictions <- predict(impute_model)
regress_mean = round(mean(predictions))
regress_std = round(sd(predictions))
data_regress2[,7] = round(rnorm(length(miss_idx), regress_mean, regress_std))
data_regress_pert = rbind(data_regress1, data_regress2)
summary(data_regress_pert)
```

```
##        V1                 V2               V3               V4
##  Min.   :   61634   Min.   : 1.000   Min.   : 1.000   Min.   : 1.000
##  1st Qu.:  870688   1st Qu.: 2.000   1st Qu.: 1.000   1st Qu.: 1.000
```

```
##  Median : 1171710   Median : 4.000   Median : 1.000   Median : 1.000
##  Mean   : 1071704   Mean   : 4.418   Mean   : 3.134   Mean   : 3.207
##  3rd Qu.: 1238298   3rd Qu.: 6.000   3rd Qu.: 5.000   3rd Qu.: 5.000
##  Max.   :13454352   Max.   :10.000   Max.   :10.000   Max.   :10.000
##        V5               V6               V7               V8
##  Min.   : 1.000   Min.   : 1.000   Min.   : 1.000   Min.   : 1.000
##  1st Qu.: 1.000   1st Qu.: 2.000   1st Qu.: 2.000   1st Qu.: 2.000
##  Median : 1.000   Median : 2.000   Median : 2.000   Median : 3.000
##  Mean   : 2.807   Mean   : 3.216   Mean   : 3.207   Mean   : 3.438
##  3rd Qu.: 4.000   3rd Qu.: 4.000   3rd Qu.: 3.000   3rd Qu.: 5.000
##  Max.   :10.000   Max.   :10.000   Max.   :11.000   Max.   :10.000
##        V9              V10              V11             V12
##  Min.   : 1.000   Min.   : 1.000   Min.   :2.00   Min.   :0.00000
##  1st Qu.: 1.000   1st Qu.: 1.000   1st Qu.:2.00   1st Qu.:0.00000
##  Median : 1.000   Median : 1.000   Median :2.00   Median :0.00000
##  Mean   : 2.867   Mean   : 1.589   Mean   :2.69   Mean   :0.02289
##  3rd Qu.: 4.000   3rd Qu.: 1.000   3rd Qu.:4.00   3rd Qu.:0.00000
##  Max.   :10.000   Max.   :10.000   Max.   :4.00   Max.   :1.00000
```

4. (Optional) Compare the results and quality of classification models (e.g., SVM, KNN) build using

(1) the data sets from questions 1,2,3;

The 1st model does not include the binary missing column variable. The 2nd model includes the binary missing column variable.

```
model1 <- V11~V2+V3+V4+V5+V6+V7+V8+V9+V10
model2 <- V11~V2+V3+V4+V5+V6+V7+V8+V9+V10+V12
```

Define the function for performing KKNN K-fold validation

```
train_kknn_fold <- function(modelcols, data, k_val) {
  model.cv <- cv.kknn(modelcols, data, kcv=10, k=k_val, scale=TRUE)
  fitted_model <- as.matrix(model.cv[[1]])
  predictions  <- as.matrix(lapply(fitted_model[, 2], round))
  accuracy <- sum(predictions == data[, c('V11')]) / nrow(data)
  return(accuracy)
}
```

Create a function to get our training and testing sets for each imputed data set. Put our data frames modified by using the mean, mode, and linear regression (including peturbations) into a list. Split the data so 1 of every 5 missing values will go to the test set. So 80% train/validate and 20% test is adhered to for missing data.

```
get_train_test_idxs <- function(data) {
  miss <- which(data[,12] == 1)
  good <- which(data[,12] != 1)
  train_miss <- sample(1:length(miss), 0.8 * length(miss))
  test_miss <- miss[-train_miss]
  train_good <- sample(1:length(good), 0.8 * length(good))
  test_good <- good[-train_good]
  train_idx <- c(train_miss, train_good)
  test_idx <- c(test_miss, test_good)
  return(list(train_idx, test_idx))
```

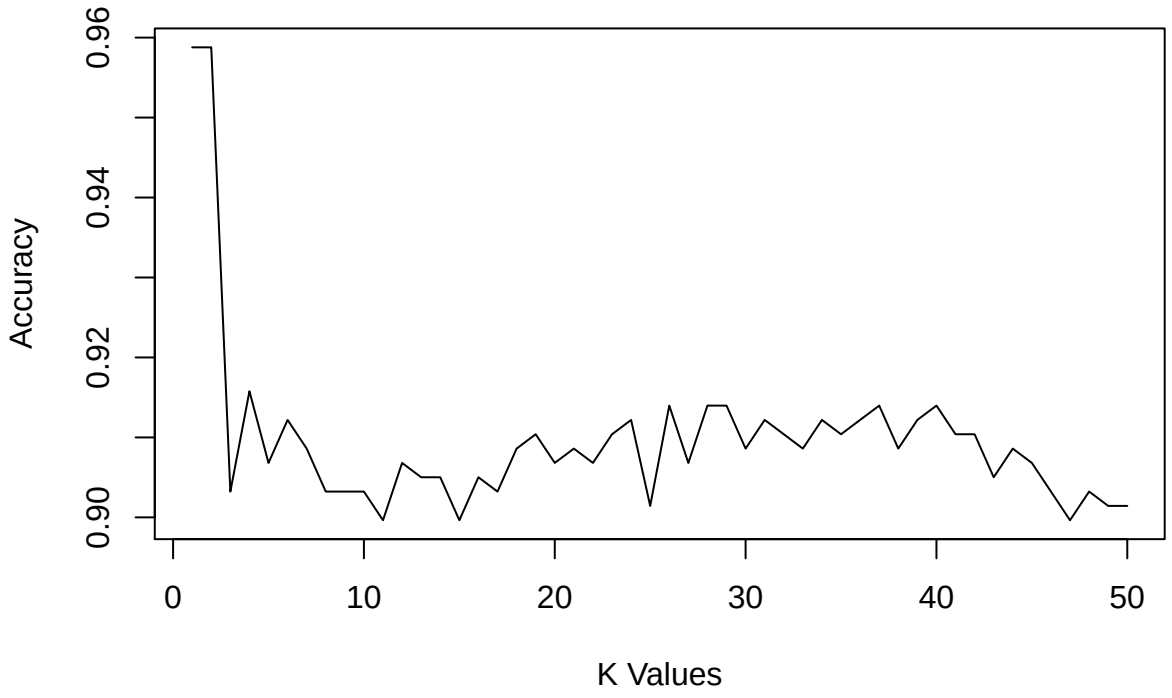
```
}
```

Create training and test sets for each of the imputed data sets using the function above.

```
datasets = get_train_test_idxs(data_mean)
train_idx = datasets[1][[1]]
test_idx = datasets[2][[1]]
data_mean_train = data_mean[train_idx,]
data_mean_test = data_mean[test_idx,]

datasets = get_train_test_idxs(data_mode)
train_idx = datasets[1][[1]]
test_idx = datasets[2][[1]]
data_mode_train = data_mode[train_idx,]
data_mode_test = data_mode[test_idx,]

datasets = get_train_test_idxs(data_regress)
train_idx = datasets[1][[1]]
test_idx = datasets[2][[1]]
data_regress_train = data_mean[train_idx,]
data_regress_test = data_mean[test_idx,]

datasets = get_train_test_idxs(data_regress_pert)
train_idx = datasets[1][[1]]
test_idx = datasets[2][[1]]
data_regress_pert_train = data_mean[train_idx,]
data_regress_pert_test = data_mean[test_idx,]

datasets = get_train_test_idxs(data_regress1)
train_idx = datasets[1][[1]]
test_idx = datasets[2][[1]]
data_exclude_missing_train = data_regress1[train_idx,]
data_exclude_missing_test = data_regress1[test_idx,]
```

Loop through K to find the best one in each of our modified data sets.

```
ksteps = seq(1,50)
kknn_accuracies = matrix(1:length(ksteps), nrow = length(ksteps), ncol = 1)
for(k in ksteps) {
  kknn_accuracies[k] = train_kknn_fold(model1, data_mean_train, k)
}
plot(kknn_accuracies, type='l', xlab='K Values', ylab='Accuracy')
```




```
knn_accuracy <- max(knn_accuracies)
train_ks = which.max(knn_accuracies)
knn_accuracy
```

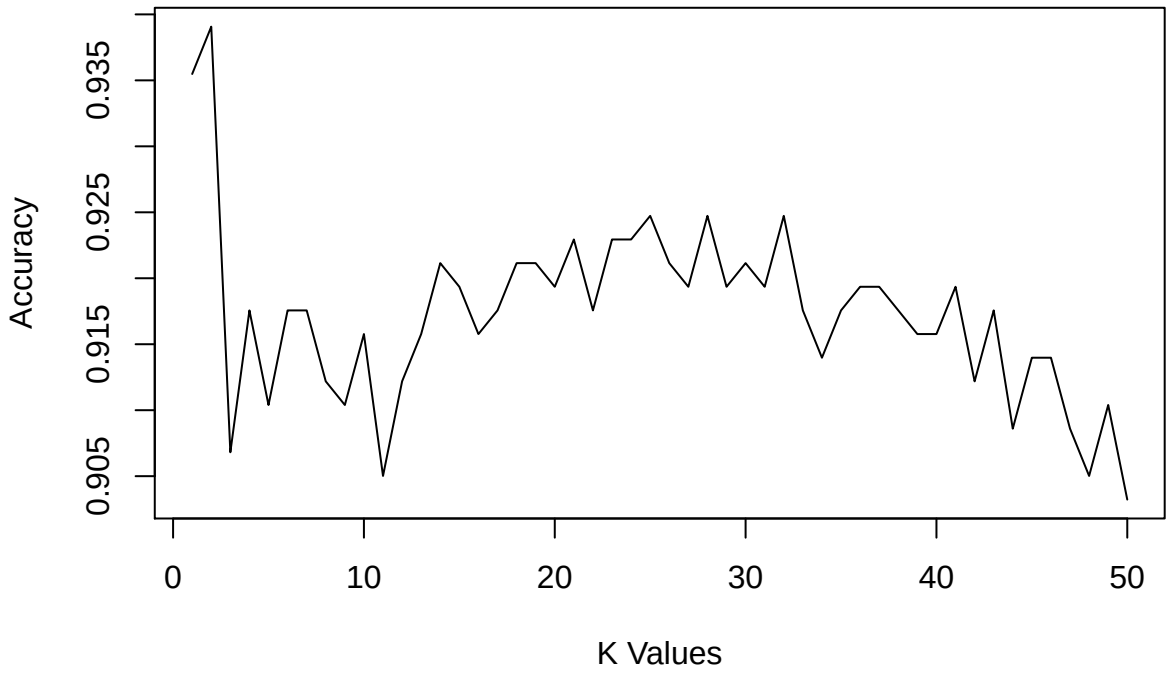
```
## [1] 0.9587814
```

```
train_ks
```

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

The best K for imputing missing data with the mean is 1, with an accuracy of 0.9587814.

```
ksteps = seq(1,50)
knn_accuracies = matrix(1:length(ksteps), nrow = length(ksteps), ncol = 1)
for(k in ksteps) {
  knn_accuracies[k] = train_knn_fold(model1, data_mode_train, k)
}
plot(knn_accuracies, type='l', xlab='K Values', ylab='Accuracy')
```




```
kknn_accuracy <- max(kknn_accuracies)
train_ks = which.max(kknn_accuracies)
kknn_accuracy
```

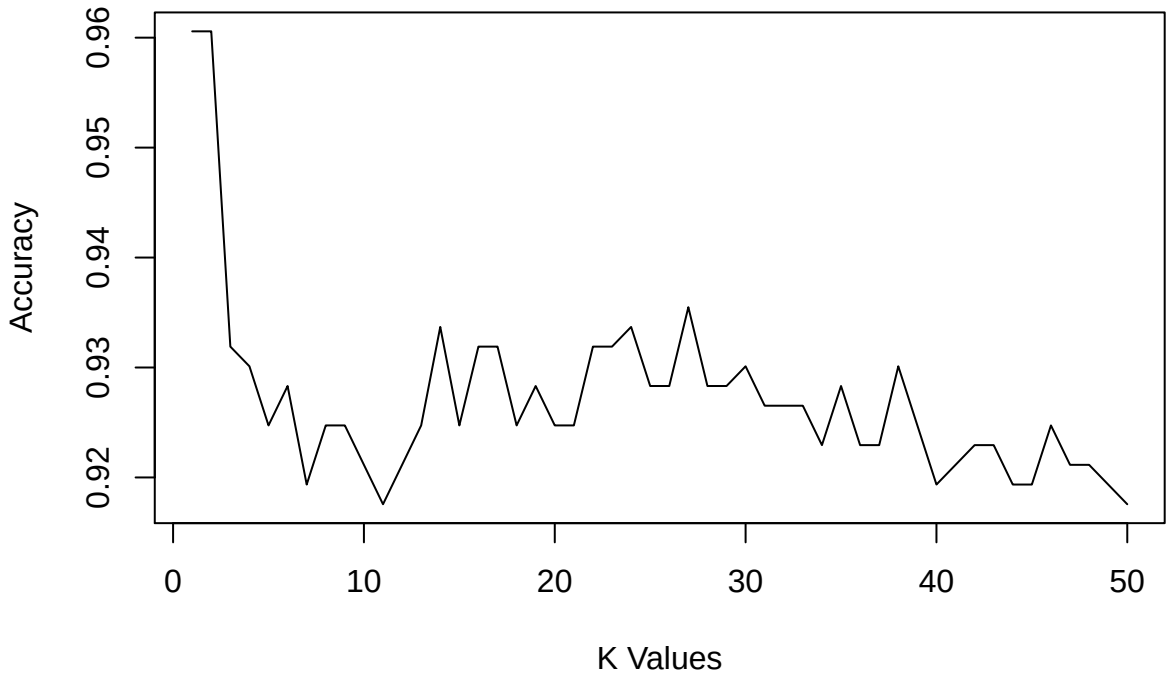
```
## [1] 0.9390681
```

```
train_ks
```

```
## [1] 2
```

The best K for imputing missing data with the mode is 2, with an accuracy of 0.9390681.

```
ksteps = seq(1,50)
kknn_accuracies = matrix(1:length(ksteps), nrow = length(ksteps), ncol = 1)
for(k in ksteps) {
  kknn_accuracies[k] = train_kknn_fold(model1, data_regress_train, k)
}
plot(kknn_accuracies, type='l', xlab='K Values', ylab='Accuracy')
```




```
kknn_accuracy <- max(kknn_accuracies)
train_ks = which.max(kknn_accuracies)
kknn_accuracy
```

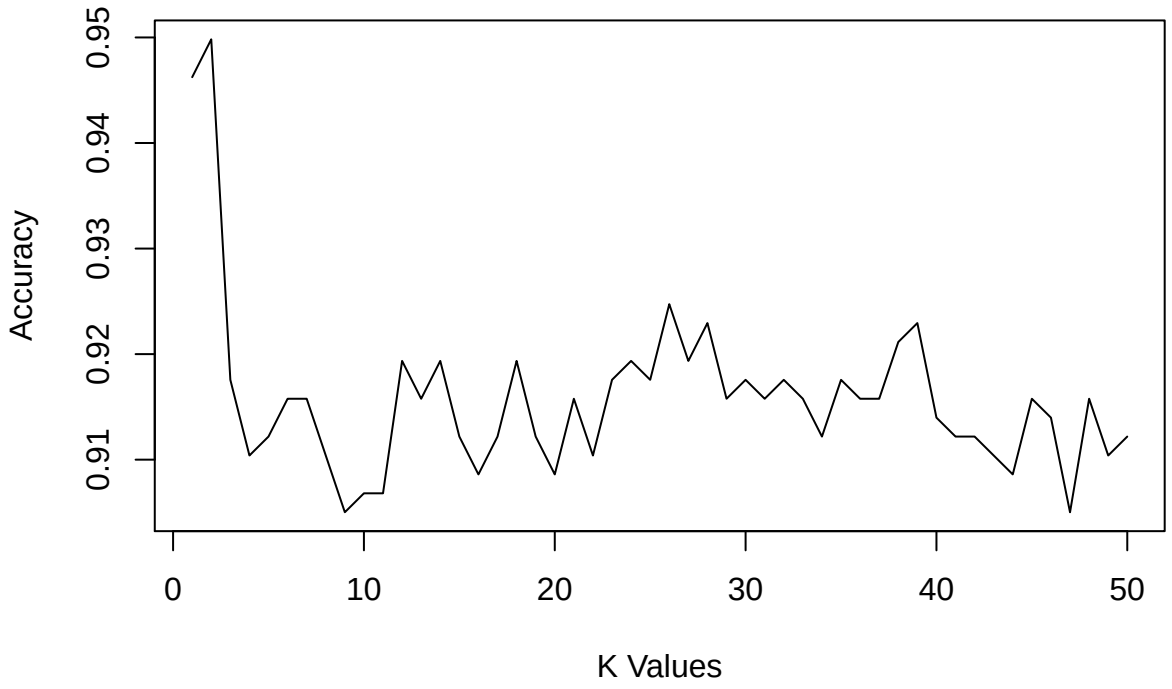
```
## [1] 0.9605735
```

```
train_ks
```

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

The best K for imputing missing data with linear regression is 1, with an accuracy of 0.9605735.

```
ksteps = seq(1,50)
kknn_accuracies = matrix(1:length(ksteps), nrow = length(ksteps), ncol = 1)
for(k in ksteps) {
  kknn_accuracies[k] = train_kknn_fold(model1, data_regress_pert_train, k)
}
plot(kknn_accuracies, type='l', xlab='K Values', ylab='Accuracy')
```




```
kknn_accuracy <- max(kknn_accuracies)
train_ks = which.max(kknn_accuracies)
kknn_accuracy
```

```
## [1] 0.9498208
```

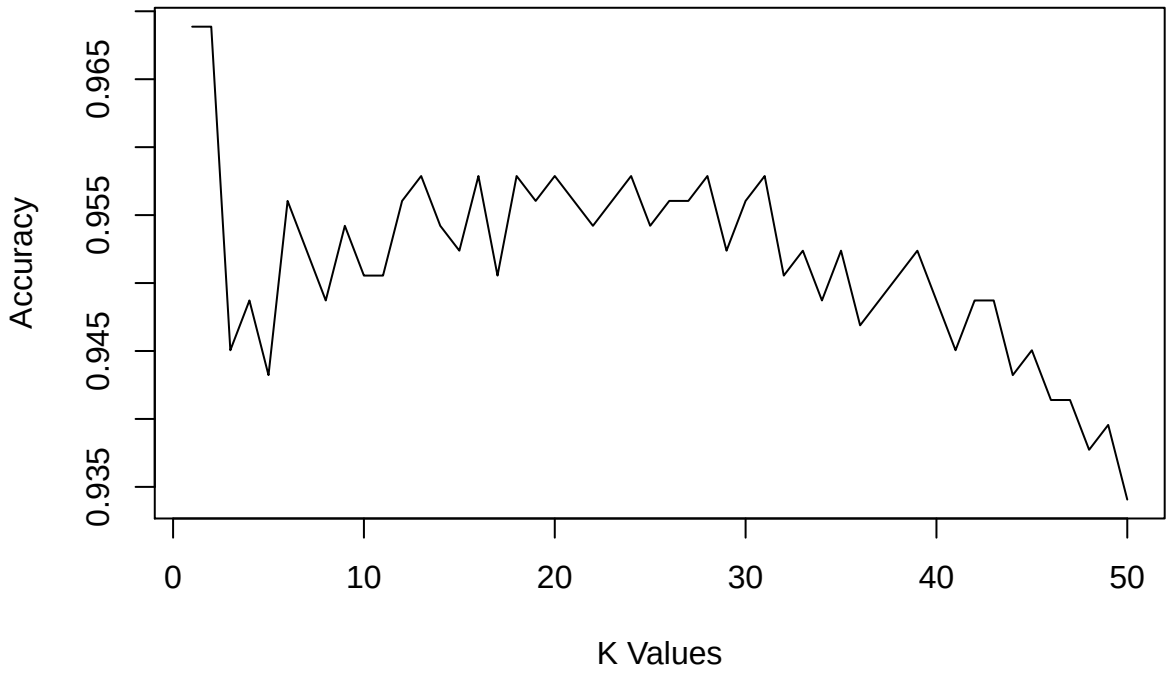
```
train_ks
```

```
## [1] 2
```

The best K for imputing missing data with linear regression with pertubation is 2, with an accuracy of 0.9498208.

(2) the data that remains after data points with missing values are removed;

```
ksteps = seq(1,50)
kknn_accuracies = matrix(1:length(ksteps), nrow = length(ksteps), ncol = 1)
for(k in ksteps) {
  kknn_accuracies[k] = train_kknn_fold(model1, data_exclude_missing_train, k)
}
plot(kknn_accuracies, type='l', xlab='K Values', ylab='Accuracy')
```




```
kknn_accuracy <- max(kknn_accuracies)
train_ks = which.max(kknn_accuracies)
kknn_accuracy
```

```
## [1] 0.9688645
```

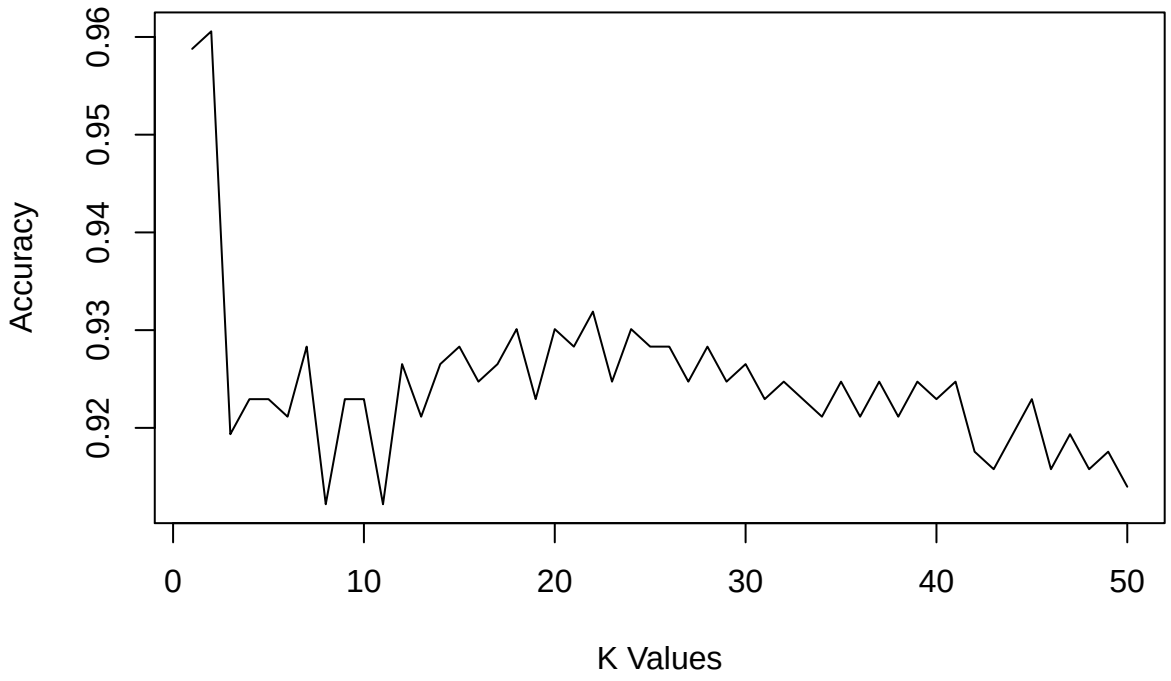
```
train_ks
```

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

The best K for leaving out missing data in its entirety is 1, with an accuracy of 0.9688645.

(3) the data set when a binary variable is introduced to indicate missing values.

```
ksteps = seq(1,50)
kknn_accuracies = matrix(1:length(ksteps), nrow = length(ksteps), ncol = 1)
for(k in ksteps) {
  kknn_accuracies[k] = train_kknn_fold(model2, data_regress_train, k)
}
plot(kknn_accuracies, type='l', xlab='K Values', ylab='Accuracy')
```




```
kknn_accuracy <- max(kknn_accuracies)
train_ks = which.max(kknn_accuracies)
kknn_accuracy
```

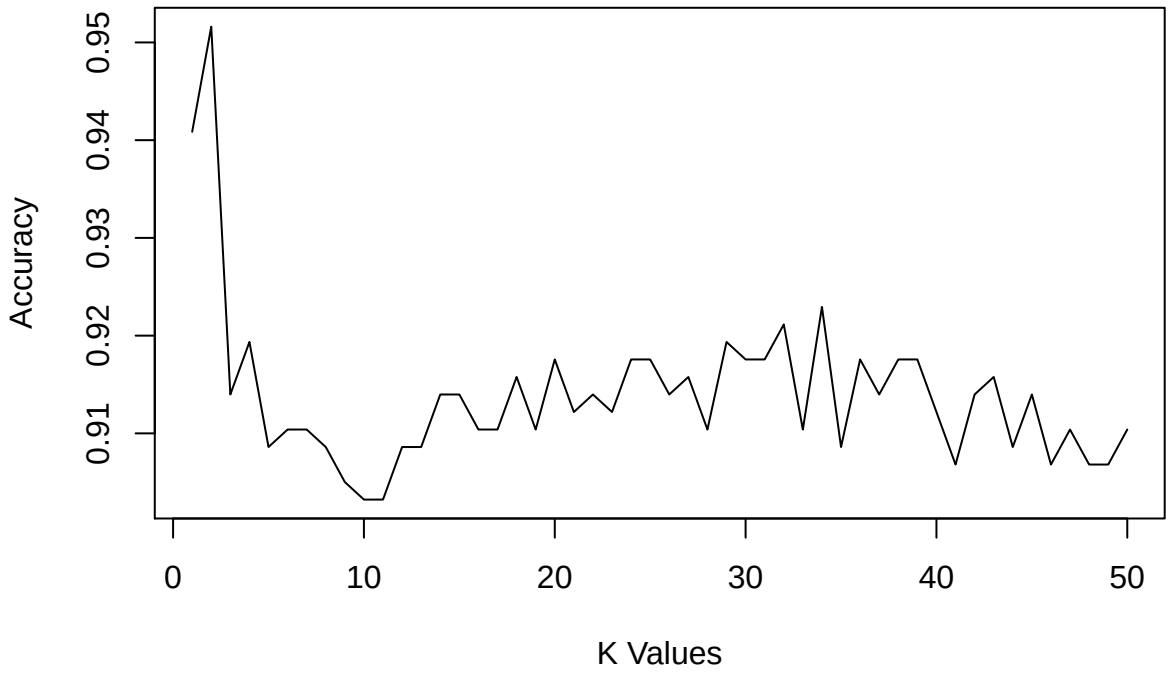
```
## [1] 0.9605735
```

```
train_ks
```

```
## [1] 2
```

The best K for imputing missing data with linear regression and adding a missing binary column is 2, with an accuracy of 0.9605735.

```
ksteps = seq(1,50)
kknn_accuracies = matrix(1:length(ksteps), nrow = length(ksteps), ncol = 1)
for(k in ksteps) {
  kknn_accuracies[k] = train_kknn_fold(model2, data_regress_pert_train, k)
}
plot(kknn_accuracies, type='l', xlab='K Values', ylab='Accuracy')
```




```
kknn_accuracy <- max(kknn_accuracies)
train_ks = which.max(kknn_accuracies)
kknn_accuracy
```

```
## [1] 0.9516129
```

```
train_ks
```

```
## [1] 2
```

The best K for imputing missing data with linear regression with pertubations and adding a missing binary column is 2, with an accuracy of 0.9516129.

Imputing data introduces error into your data sets. This can be seen as the accuracy of leaving out data with missing values is more accurate than all models that impute missing data. Generally if less than 5% of your data is missing you can impute it. Using this data set. The best model using imputed data uses regression and adding a missing column binary variable. Assuming you wanted to use the model with imputed data I would use that one. Assuming that running kknn using the K value on the test set, produces the following accuracy.

```
model = kknn(model2, data_regress_train, data_regress_test, k=2)
predictions <- as.integer(fitted(model)+0.5)
test_accuracy = sum(predictions == data_regress_test$V11) / nrow(data_regress_test)
test_accuracy
```

```
## [1] 0.9432624
```

Question 15.1

I work for a FinTech Company that analyzes Insider Trading Data and Quarterly and Annual Reports. We use Amazon aws to host our website. We need more capacity during peak filing periods when demand is higher. Since aws can be quite costly, we don't want to be running extra ec2 instances when demand is not high. Optimization would be good for determining when we should create extra ec2 instances, and how many to create, during peak filing periods.