3.1

In [1]:
```
setwd('/Users/Mitchell.Ramey/Desktop')
```

In [3]:
```
d0 = read.table('credit_card_data-headers.txt', header=TRUE)
head(d0)
```

A1	A2	A3	A8	A9	A10	A11	A12	A14	A15	R1
1	30.83	0.000	1.25	1	0	1	1	202	0	1
0	58.67	4.460	3.04	1	0	6	1	43	560	1
0	24.50	0.500	1.50	1	1	0	1	280	824	1
1	27.83	1.540	3.75	1	0	5	0	100	3	1
1	20.17	5.625	1.71	1	1	0	1	120	0	1
1	32.08	4.000	2.50	1	1	0	0	360	0	1

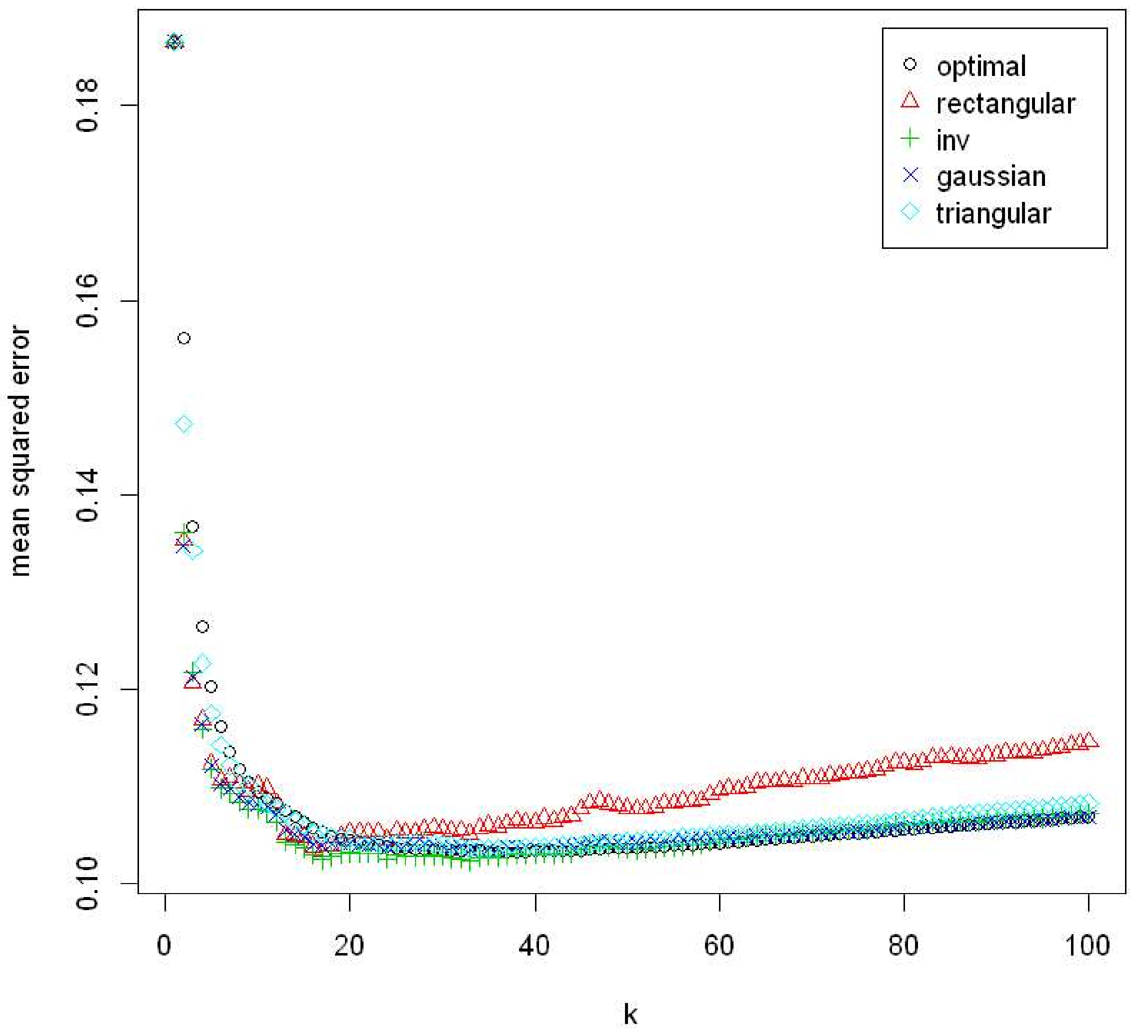
In [7]:
```
library(kknn)
model1 <- train.kknn(d0$R1~., d0, kmax = 100, distance = 0.25, kernel = c("opt
imal", "rectangular", "inv", "gaussian", "triangular"), sclae = TRUE)
model1
```

```
Call:
train.kknn(formula = d0$R1 ~ ., data = d0, kmax = 100, distance = 0.25,     k
ernel = c("optimal", "rectangular", "inv", "gaussian", "triangular"),     scl
ae = TRUE)

Type of response variable: continuous
minimal mean absolute error: 0.184007
Minimal mean squared error: 0.1023529
Best kernel: inv
Best k: 33
```

I tested a few different models using multiple distance values. A distance value of 0.25 gave the lowest mean squared error. The best kernel is inv and best k is 33.

In [8]:
```
plot(model1)
```



In [9]:
```
# Splitting the data into training, test, and validation sets
# assigns a train, validate, or test label to each row of a data frame and the
n splits based on the label of each row
spec = c(train = .6, test = .2, validate = .2)
x = sample(cut(seq(nrow(d0)), nrow(d0)*cumsum(c(0,spec)), labels = names(spec
)))
d = split(d0, x)
```

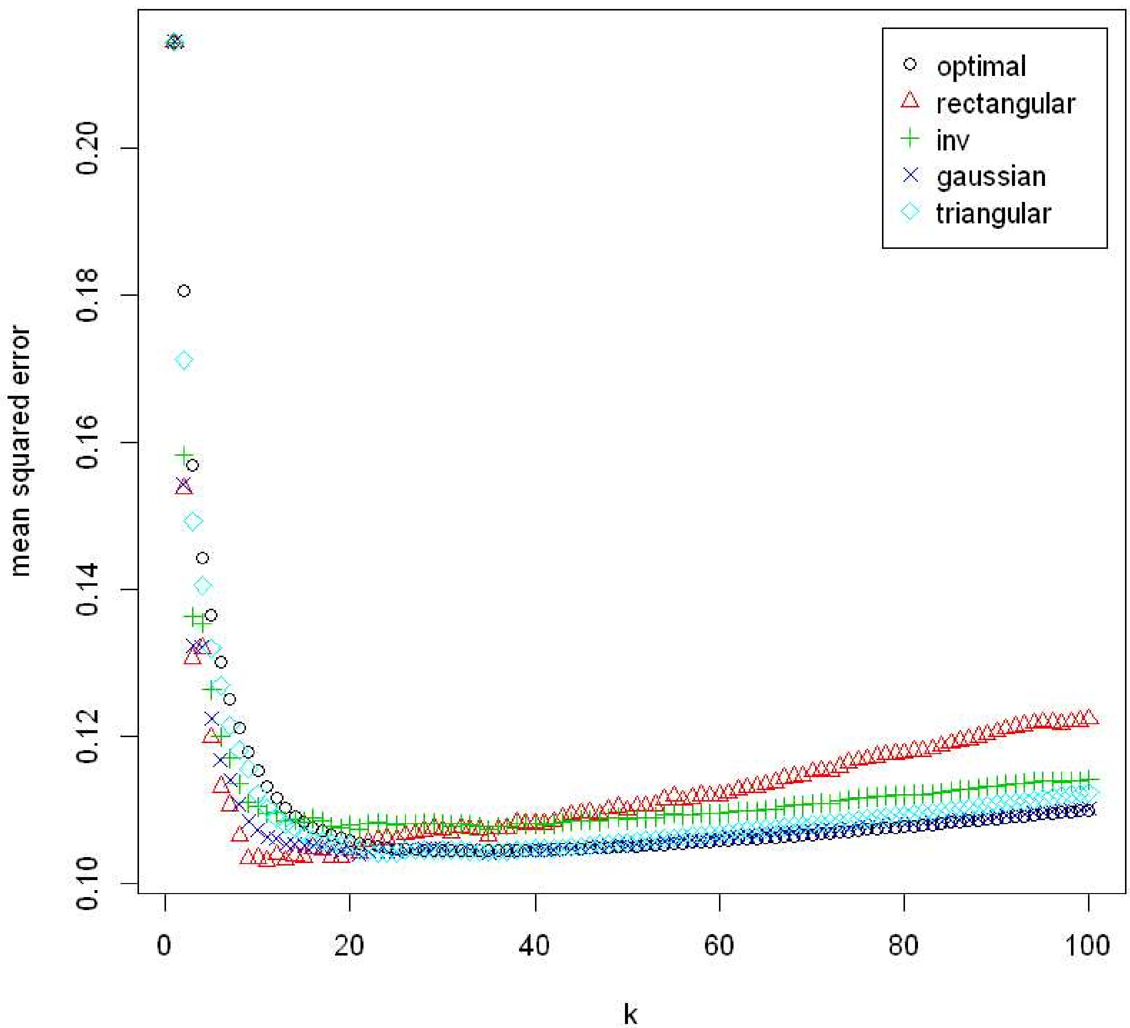
In [13]:
```
# Leave-one-out cross validation model using the test data to determine optima
l K, distance and kernel
train_model = train.kknn(R1 ~ ., data = d$train, kmax = 100, distance = 0.25,
kernel = c("optimal","rectangular", "inv", "gaussian", "triangular"), scale =
TRUE)
train_model
```

```
Call:
train.kknn(formula = R1 ~ ., data = d$train, kmax = 100, distance = 0.25,
kernel = c("optimal", "rectangular", "inv", "gaussian", "triangular"),     sc
ale = TRUE)

Type of response variable: continuous
minimal mean absolute error: 0.2015306
Minimal mean squared error: 0.1031371
Best kernel: rectangular
Best k: 11
```

After performing leave-one-out cross validation with a distance value of 0.25, the best kernel is rectangular and the best k = 11

In [14]:
```
plot(train_model)
```



In [15]:
```
# Testing the model using the test data
pred <- predict(train_model, d$test)
pred_bin <- round(pred)
pred_accuracy <- table(pred_bin, d$test$R1)
pred_accuracy
```

```
pred_bin  0  1
       0 60  7
       1 11 53
```

In [16]:
```
sum(pred_bin==d$test$R1)/length(d$test$R1)
```

0.862595419847328

Performing a prediction using the test data on the trained model, the accuracy is 86%

In [17]:
```
# Validating the model using the validation data set
val <- predict(train_model, d$validate)
pred_bin <- round(val)
pred_accuracy <- table(pred_bin, d$validate$R1)
sum(pred_bin==d$validate$R1)/length(d$validate$R1)
```

0.885496183206107

Performing a prediction using the validation data on the trained model, the accuracy is 89%

4.1

I am currently a data analyst for a tele-communications client. I could leverage clustering to find out the best location to build new data centers by clustering the locations of the customers.

4.2

In [3]:
```
library(datasets)
data(iris)
head(iris)
```

Sepal.Length	Sepal.Width	Petal.Length	Petal.Width	Species
5.1	3.5	1.4	0.2	setosa
4.9	3.0	1.4	0.2	setosa
4.7	3.2	1.3	0.2	setosa
4.6	3.1	1.5	0.2	setosa
5.0	3.6	1.4	0.2	setosa
5.4	3.9	1.7	0.4	setosa

In [4]:
```
#Split the data
iris.pred<- iris[,c(1,2,3,4)]
iris.class<- iris[,"Species"]
head(iris.pred)
```

Sepal.Length	Sepal.Width	Petal.Length	Petal.Width
5.1	3.5	1.4	0.2
4.9	3.0	1.4	0.2
4.7	3.2	1.3	0.2
4.6	3.1	1.5	0.2
5.0	3.6	1.4	0.2
5.4	3.9	1.7	0.4

In [5]:
```
#Normalize the values of the predictors to values between 0 and 1

normalize <- function(x){
  return ((x-min(x))/(max(x)-min(x)))
}

iris.pred$Sepal.Length<- normalize(iris.pred$Sepal.Length)
iris.pred$Sepal.Width<- normalize(iris.pred$Sepal.Width)
iris.pred$Petal.Length<- normalize(iris.pred$Petal.Length)
iris.pred$Petal.Width<- normalize(iris.pred$Petal.Width)
head(iris.pred)
```

Sepal.Length	Sepal.Width	Petal.Length	Petal.Width
0.22222222	0.6250000	0.06779661	0.04166667
0.16666667	0.4166667	0.06779661	0.04166667
0.11111111	0.5000000	0.05084746	0.04166667
0.08333333	0.4583333	0.08474576	0.04166667
0.19444444	0.6666667	0.06779661	0.04166667
0.30555556	0.7916667	0.11864407	0.12500000

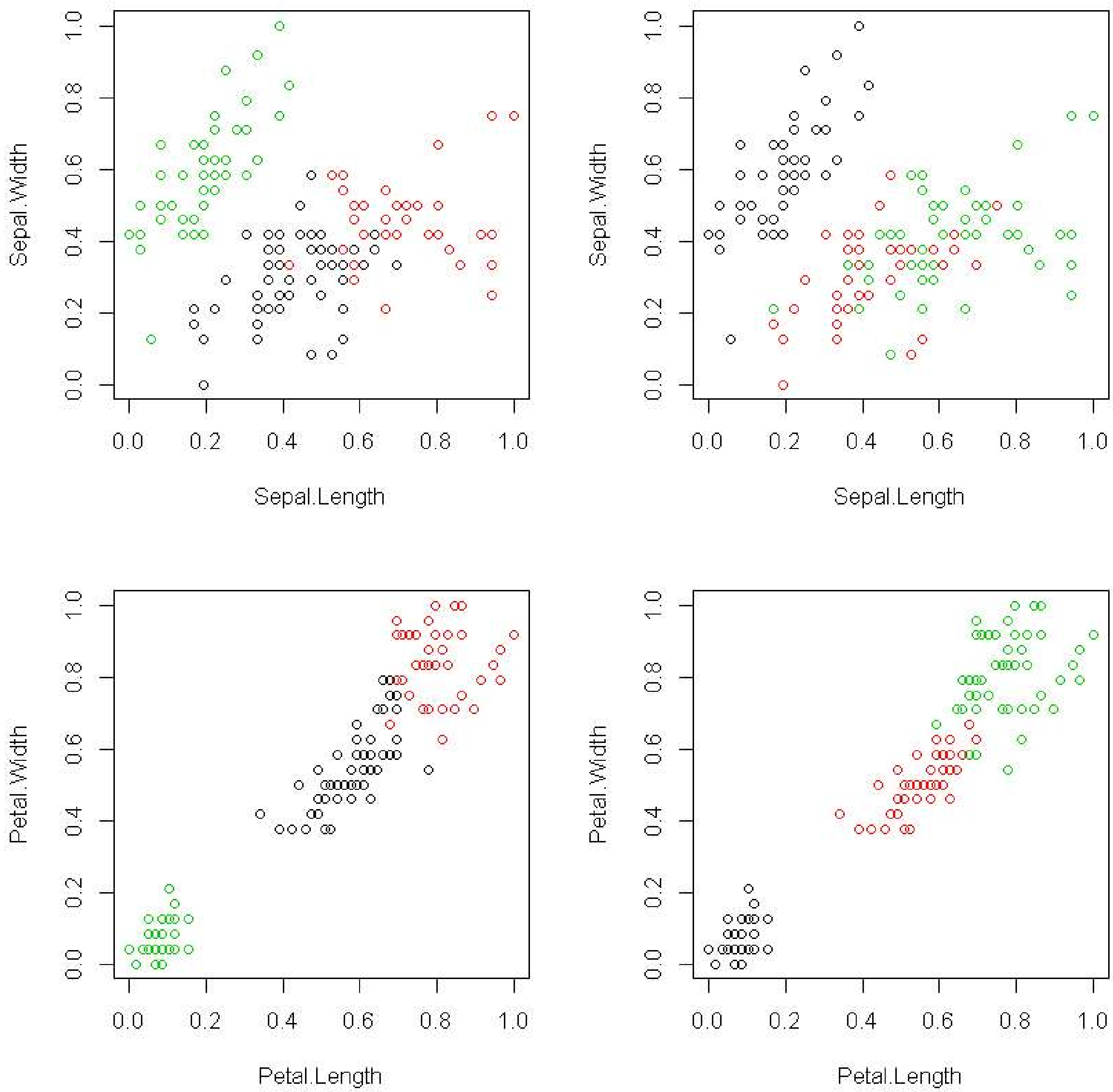
In [10]:
```
#Apply K-means clustering with 3 centroids & view number of records in each cl
uster
clstr <- kmeans(iris.pred,3)
clstr$size
```

61 39 50

In [14]:
```
# View the values of the center data points of each cluster
clstr$centers
```

Sepal.Length	Sepal.Width	Petal.Length	Petal.Width
0.4412568	0.3073770	0.57571548	0.54918033
0.7072650	0.4508547	0.79704476	0.82478632
0.1961111	0.5950000	0.07830508	0.06083333

In [19]:
```
par(mfrow=c(2,2), mar=c(5,4,2,2))
# Plot to see how Sepal.Length and Sepal.Width data points have been distribut
ed in clusters
plot(iris.pred[c(1,2)], col=clstr$cluster)
# Plot to see how Sepal.Length and Sepal.Width data points have been distribut
ed originally as per "class" attribute in dataset
plot(iris.pred[c(1,2)], col=iris.class)
# Plot to see how Petal.Length and Petal.Width data points have been distribut
ed in clusters
plot(iris.pred[c(3,4)], col=clstr$cluster)
plot(iris.pred[c(3,4)], col=iris.class)
```



In [20]:
```
table(clstr$cluster,iris.class)
```

```
   iris.class
    setosa versicolor virginica
  1      0         47        14
  2      0          3        36
  3     50          0         0
```

Table results show that Cluster 1 belongs to Versicolor, Cluster 2 belongs to Virginica, and Cluster 3 belongs to Setoso