MAT 325 Essentials of Data Science

Matt McCullough

Homework 5 (Due: December 1, 2021)

Note: *Insert blocks of R code and answers to the questions below. Answers in text should be displayed in italics.*

(1) (5 pts) Write a block of R code that loads the libraries ggplot2, dplyr, class, caret and ModelMetrics and implements any R functions needed to answer the questions below.

```
library(ggplot2)
library(dplyr)
library(class)
library(caret)
library(ModelMetrics)

# Normalization function
# Input: vector x
# Output: normalized vector
nor <- function(x) {
  return( (x -min(x))/(max(x)-min(x)) )
}

# This function calculates the accuracy of a
# learning algorithm
accuracy <- function(T){
    return (sum(diag(T)/(sum(rowSums(T)))) * 100)
}
```

(2). (30 pts total) *Environmental Science and Technology* (January 2005) reported on a study of the reliability of a commercial kit to test for arsenic in groundwater. The field kit was used to test a sample of 328 groundwater wells in Bangladesh. In addition to the arsenic level (micrograms per liter), the latitude (degrees), longitude (degrees), and depth (feet) of each well was measured. The data are saved in the ASWELLS.Rdata file. We wish to fit a first-order model for arsenic level as a function of latitude, longitude, and depth.

(a). (3 pts) Write a first-order model for arsenic level (y) as a function of latitude (x_1), longitude (x_2), and depth (x_3). (This is just the functional form relating y to the predictor variables that involves unspecified β coefficients.)

First Order Model: $y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \beta_3 x_3$

(b). (5 pts) Load the ASWELLS data set and perform linear regression using the first order model in (a). Show the R code and the output of the *summary* function.

```
arsenic = load("ASWELLS.Rdata")
y=ASWELLS$ARSENIC
x1=ASWELLS$LATITUDE
x2=ASWELLS$LONGITUDE
x3=ASWELLS$DEPTHFT
m=lm(y~x1+x2+x3)
summary(m)
```

```
##
## Call:
## lm(formula = y ~ x1 + x2 + x3)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -134.41  -65.51  -26.85   27.05  469.32
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept) -8.687e+04  3.122e+04  -2.782  0.00572 **
## x1          -2.219e+03  5.268e+02  -4.212 3.29e-05 ***
## x2           1.542e+03  3.731e+02   4.134 4.55e-05 ***
## x3          -3.496e-01  1.566e-01  -2.232  0.02628 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 103.3 on 323 degrees of freedom
##   (1 observation deleted due to missingness)
## Multiple R-squared:  0.128,  Adjusted R-squared:  0.1199
## F-statistic:  15.8 on 3 and 323 DF,  p-value: 1.308e-09
```

(c). (3 pts) Write the least squares prediction equation for arsenic level obtained from the regression output.

```
m
```

```
##
## Call:
## lm(formula = y ~ x1 + x2 + x3)
##
## Coefficients:
## (Intercept)           x1           x2           x3
##  -8.687e+04   -2.219e+03    1.542e+03   -3.496e-01
```

The line would be y = -86870 - 2219x_1 + 1542x_2 - 0.3496x_3

(d). (5 pts) Would it be appropriate to interpret the estimate of the coefficient of the depth variable in the model? Why or why not? If so, give a practical interpretation.

Yes, it would be appropriate to interpret the estimate of the coefficient of the depth variable in the model as it is a linear model.
A practical interpretation of this is that for every decrease of one foot of the depth of each well, the arsenic level decreases by 0.3496 micrograms per liter

(e). (3 pts) Find the model standard deviation, s, and interpret its value.

s is 103.3 and it's value says that 95% of the data lies within 2s, or 206.6

(f). (4 pts) What are the R^2 and the R_a^2 values? Interpret both values.

R^2 is 0.128 and R_a^2 is 0.1199. R^2 represents the fact that 12.8% of the variability of the arsenic level is explained by the regression line and the R_a^2 represents an adjusted R^2 and that 11.99% of the variability is explained by the regression line

(g). (5 pts) Conduct a test of overall model utility (the global F-test) at the $\alpha = 0.05$ level. State the hypotheses, write the value of the test statistic, write the P-value, and state the conclusion.

Hypothesis: H_0 : $\beta_1 + \beta_2 + \beta_3$ vs. H_a: at least one $B_k \neq 0$
F Statistic: 15.8
P value: 1.308 x 10^-9
Conclusion: Given this information there is evidence to reject the null hypothesis as $\alpha < 0.05$

(h). (2 pts) Based on the results, parts (d)–(f), would you recommend using the model to predict arsenic level (y)? Explain.

Based on these results, I would not use this to predict arsenic level since the R^2 value is low and that one of the coefficients is not 0 given the hypothesis test

(3). (5 pts) For this exercise, we will use the diamonds data set from the ggplot2 library. First, set the random seed to some integer between 1 and 1000 to ensure replicability of the results. Next, create a smaller data set by randomly selecting 10% of the rows of the diamonds data set and then choosing only the columns corresponding to the following variables (in this order): cut, price, carat, depth, table, x, y, and z. The descriptions of these variables can be found by typing "?diamonds" on the console.

Important Note: The diamonds data set is an example of a tibble, which differs slightly from a data frame. Use the *as.data.frame* function to ensure that the resulting data set is a data frame before applying the knn function.

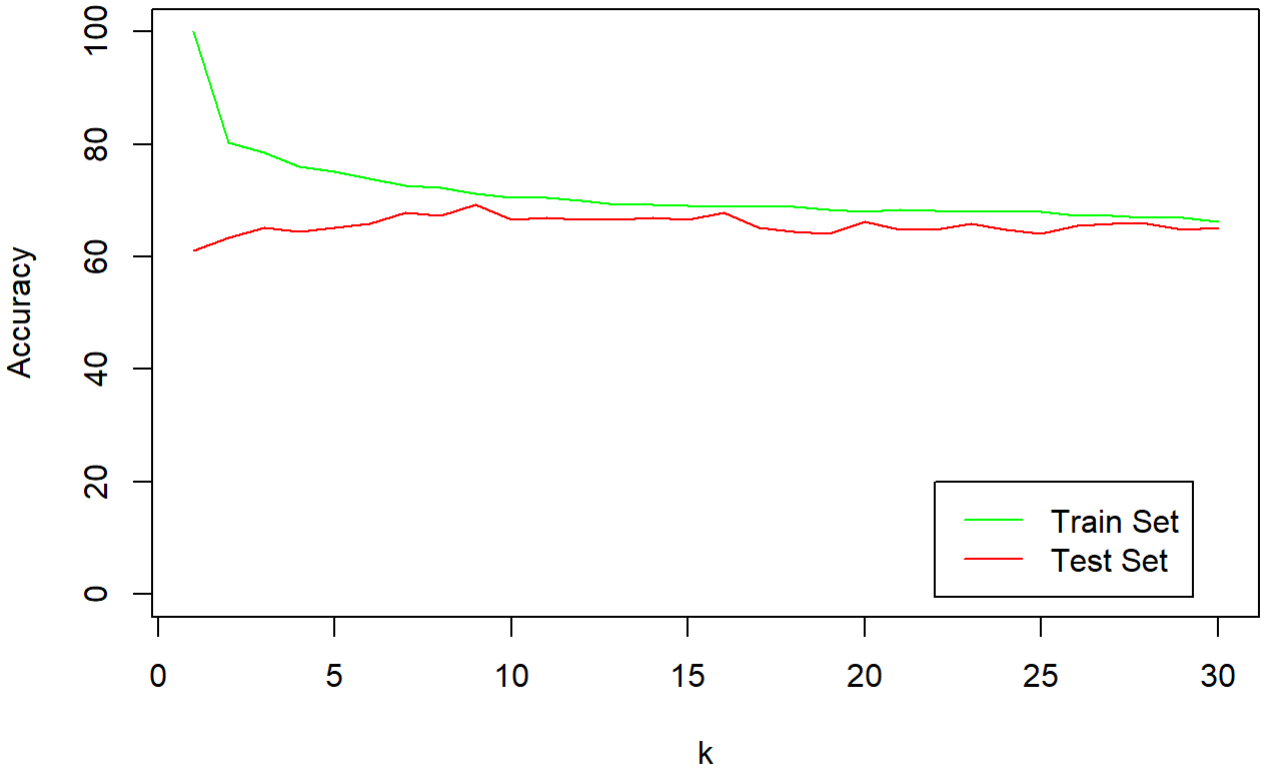
```
#I used 5% of the data because my computer is slow

set.seed(1)
diamondData = diamonds[,c("cut", "price", "carat", "depth", "table", "x", "y", "z")]
set = sample(1:nrow(diamondData), floor(.05 * nrow(diamondData)))

subsetDiamond = as.data.frame(diamondData[set,])
```

(4). (25 pts total) For this exercise, use the subset of the diamonds data set from (3). The goal is to develop models that can predict the *quality of the cut* of a diamond (Fair, Good, Very Good, Premium, Ideal) based on the quantitative variables in the data set, namely, the *price, carat, depth, table, x, y*, and *z*. Use k-NN to predict the quality of the cut of a diamond based on the above quantitative variables.

(a). (10 pts) Split the data set into 90% training set and 10% test set. Determine the accuracy of k-NN on the training set for $k = 1$ to 30. Determine also the accuracy of kNN on the test set for $k = 1$ to 30. Show the plot of the accuracy on the training set vs. k and the plot of the accuracy on the test set vs k on the same graph. Provide the R code below.

```
# normalize first columns that are not cut (these correspond to
# the predictors)
diamond_norm <- as.data.frame(lapply(subsetDiamond[,2:8], nor))

# set random seed
set.seed(1)

# Randomly select 90% of the data
ran <- sample(1:nrow(subsetDiamond), floor(0.9 * nrow(subsetDiamond)))

# extract training set
diamond_train <- diamond_norm[ran,]

# extract testing set
diamond_test <- diamond_norm[-ran,]

#get cut for category
diamond_train_category <- subsetDiamond[ran,1]
diamond_test_category <- subsetDiamond[-ran,1]

numk <- 30
AccuracyTestVec <- numeric(numk)
AccuracyTrainVec <- numeric(numk)

for (k in 1:numk)
{
    #test set

    # run knn function
    pr <- knn(diamond_train,diamond_test,
              cl=diamond_train_category,k)

    # create confusion matrix
    tab <- table(pr,diamond_test_category)

    # calculate the accuracy
    AccuracyTestVec[k] = accuracy(tab)

    #training set

    # run knn function
    pr <- knn(diamond_train,diamond_train,
              cl=diamond_train_category,k)

    # create confusion matrix
    tab <- table(pr,diamond_train_category)

    # calculate the accuracy
    AccuracyTrainVec[k] = accuracy(tab)
}
```

```
# plot accuracy of Training Set vs k
plot(1:numk,AccuracyTrainVec,type="l",col="green",
     lty=1,pch=19,ylim = c(0,100),xlab="k",ylab="Accuracy")
# plot accuracy of Test Set vs k (superimposed on previous plot)
lines(1:numk,AccuracyTestVec,type="l",col="red",lty=1,pch=19)
legend(22,20,legend=c('Train Set','Test Set'),col=c('green','red'),lty=1)
```



(b). (5 pts) What can you say about the plot of k-NN accuracy on the training set vs k? What about the plot of k-NN accuracy on the test set vs k? Are the results consistent with what are expected? Based on these plots, what would be your recommended value of k for this data set?

I can say that the accuracy of the training set goes from 100% to quickly drop to the high 60%'s and the accuracy of the test set does the opposite where it increases from around 60% to 70%. This is what I expected to happen and are consistent with what is expected. Given this data, my recommended value for k would be 9 as it's the first substantial increase in the test set accuracy, which would yield a smaller k value with relatively good accuracy

(c). (10 pts) Write an R script containing R functions that calculate the LOOCV performance (accuracy) of k-NN for $k = 1$ to 30 on your data set from (3). What value of k would be suitable for this data set? Show the R script you used below and the output obtained.

```
# This function calculates the accuracy in LOOCV for a fixed k
kNN_LOOCV_Accuracy <- function(k) {

  # normalize the predictors
  diamond_norm <- as.data.frame(lapply(subsetDiamond[,2:8], nor))

  # number of data points
  numdata <- nrow(diamond_norm)

  # number of correct predictions
  numcorrect <- 0

  for (i in 1:numdata) {

    # extract training set
    diamond_train <- diamond_norm[-i,]

    # extract testing set
    diamond_test <- diamond_norm[i,]

    # extract column of training set (to be used
    # as 'cl' argument in the knn function)
    diamond_train_category <- subsetDiamond[-i,1]

    # extract column of test set (to be used for
    # measuring the accuracy)
    diamond_test_category <- subsetDiamond[i,1]

    # run knn function
    pr <- knn(diamond_train,diamond_test,
              cl=diamond_train_category,k)

    testY <- diamond_test_category[1]
    testY<-factor(testY, ordered=FALSE)

    if (pr[1]==testY) {
      numcorrect <- numcorrect+1
    }

  }
  print(paste('Number of correct predictions: ',
              numcorrect))
  accuracy <- numcorrect/numdata
  return(accuracy)

}

# This function returns the LOOCV accuracy of k-NN
# for many values of k and chooses the smallest k
# that gives maximum accuracy
Choosek <- function() {
  numdata <- nrow(subsetDiamond)
```

```
  numk <- 30  # max value of k to consider
  AccuracyVec <- integer(numk)
  for (k in 1:numk) {
    AccuracyVec[k] <- kNN_LOOCV_Accuracy(k)
  }
  k_best <- which.max(AccuracyVec)
  print(paste('The best accuracy is ',max(AccuracyVec),' at k =',k_best))
  print(paste('Accuracy of k-NN for k = 1 to ',numk,':'))
  return(AccuracyVec)
}

Choosek()
```

```
## [1] "Number of correct predictions:  1611"
## [1] "Number of correct predictions:  1578"
## [1] "Number of correct predictions:  1697"
## [1] "Number of correct predictions:  1716"
## [1] "Number of correct predictions:  1743"
## [1] "Number of correct predictions:  1733"
## [1] "Number of correct predictions:  1762"
## [1] "Number of correct predictions:  1767"
## [1] "Number of correct predictions:  1764"
## [1] "Number of correct predictions:  1779"
## [1] "Number of correct predictions:  1764"
## [1] "Number of correct predictions:  1755"
## [1] "Number of correct predictions:  1758"
## [1] "Number of correct predictions:  1768"
## [1] "Number of correct predictions:  1774"
## [1] "Number of correct predictions:  1757"
## [1] "Number of correct predictions:  1751"
## [1] "Number of correct predictions:  1749"
## [1] "Number of correct predictions:  1757"
## [1] "Number of correct predictions:  1765"
## [1] "Number of correct predictions:  1750"
## [1] "Number of correct predictions:  1746"
## [1] "Number of correct predictions:  1758"
## [1] "Number of correct predictions:  1760"
## [1] "Number of correct predictions:  1749"
## [1] "Number of correct predictions:  1749"
## [1] "Number of correct predictions:  1740"
## [1] "Number of correct predictions:  1747"
## [1] "Number of correct predictions:  1746"
## [1] "Number of correct predictions:  1738"
## [1] "The best accuracy is  0.659621802002225  at k = 10"
## [1] "Accuracy of k-NN for k = 1 to  30 :"
```

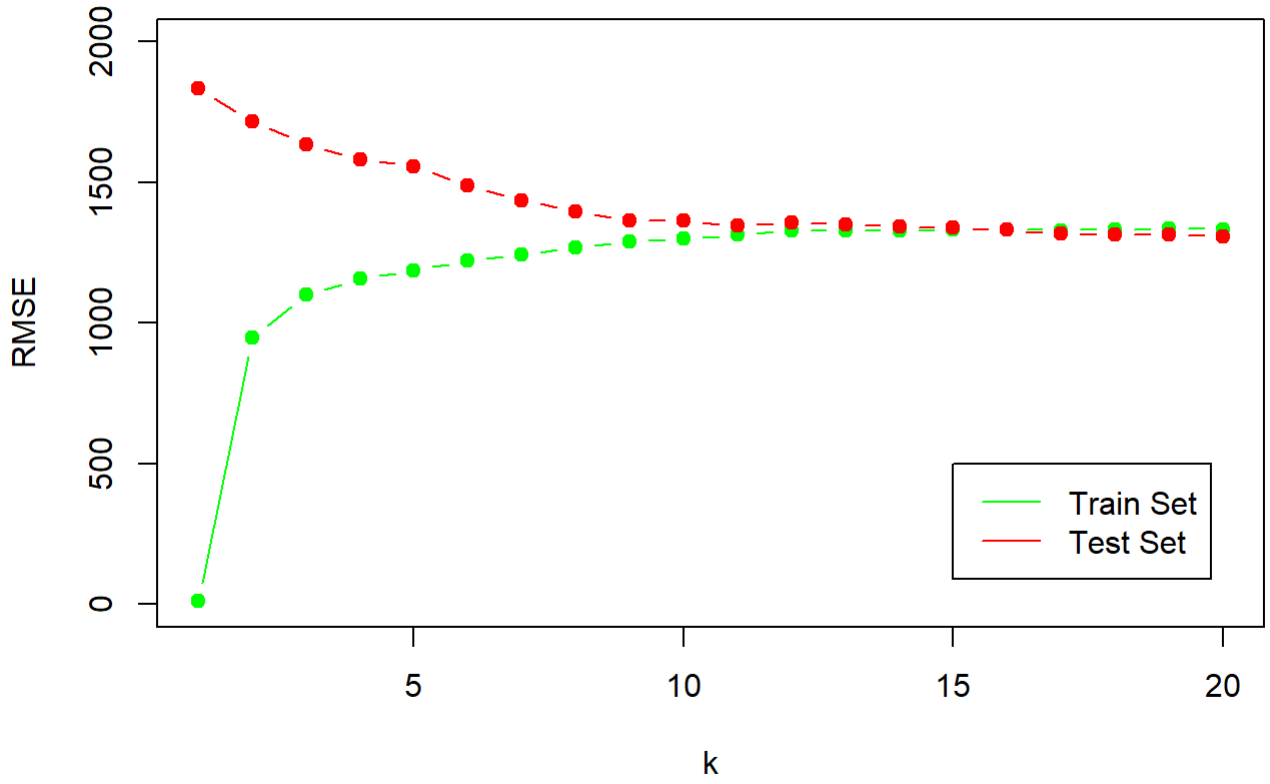
```
##  [1] 0.5973304 0.5850945 0.6292176 0.6362625 0.6462736 0.6425658 0.6533185
##  [8] 0.6551724 0.6540601 0.6596218 0.6540601 0.6507230 0.6518354 0.6555432
## [15] 0.6577679 0.6514646 0.6492399 0.6484983 0.6514646 0.6544308 0.6488691
## [22] 0.6473860 0.6518354 0.6525769 0.6484983 0.6484983 0.6451613 0.6477568
## [29] 0.6473860 0.6444197
```

(5). (15 pts total) We use the same data set as in (4) but this time we will use k-NN to predict the price of a diamond based on the variables carat, depth, table, x, y, and z. (Note: There is no need to modify the data frame from (4). Simply extract the columns that are needed by the knnreg function.)

(a). (10 pts) Split the data set into 90% training set and 10% test set. Determine the RMSE (root mean square error) of k-NN regression on the training set for $k = 1$ to 20. Determine also the RMSE of k-NN regression on the test set for $k = 1$ to 20. Show the plot of the RMSE on the training set vs k and the plot of the RMSE on the test set vs k on the same graph. Provide the R code below.

```
# normalize first columns that are not cut and price (these correspond to
# the predictors)
diamond_norm <- as.data.frame(lapply(subsetDiamond[,3:8], nor))

# set random seed
set.seed(1)

# Randomly select 90% of the data
ran <- sample(1:nrow(subsetDiamond), floor(0.9 * nrow(subsetDiamond)))

# extract training set
diamond_trainX <- diamond_norm[ran,]
diamond_trainY <- subsetDiamond[ran,2]


# extract testing set
diamond_testX <- diamond_norm[-ran,]
diamond_testY <- subsetDiamond[-ran,2]

# plot kNN RMSE vs k
numk <- 20 # number of values of k to consider
kvalues <- 1:numk

# calculate RMSE on the training set for
# different values of k
RMSETrainSet <- numeric(numk)
for (k in 1:numk) {

  # run knnreg function
  fit <- knnreg(diamond_trainX, diamond_trainY, k)

  # predict on training set
  diamond_pred <- predict(fit, diamond_trainX)

  # calculate rmse
  RMSETrainSet[k] <- rmse(diamond_trainY,diamond_pred)

}


# calculate RMSE on the test set for
# different values of k
RMSETestSet <- numeric(numk)
for (k in 1:numk) {

  # run knnreg function
  fit <- knnreg(diamond_trainX, diamond_trainY, k)

  # predict on test set
  diamond_pred <- predict(fit, diamond_testX)

  # calculate rmse
```

```
  RMSETestSet[k] <- rmse(diamond_testY,diamond_pred)

}

# plot RMSE on Training Set vs k
plot(1:numk,RMSETrainSet,type="b",col="green",
     lty=1,pch=19,ylim = c(0,2000),xlab="k",ylab="RMSE")
# plot RMSE on Test Set vs k (superimposed on previous plot)
lines(1:numk,RMSETestSet,type="b",col="red",lty=1,pch=19)
legend(15,500,legend=c('Train Set','Test Set'),col=c('green','red'),lty=1)
```



(b). (5 pts) What can you say about the plot of RMSE on the training set vs k? What about the plot of RMSE on the test set vs k? Are the results consistent with what are expected? Based on these plots, What would be your recommended value of k for this data set?

I would say that is what was expected as the training set started near 0 and spiked while the test set stayed consistent. I would use the value 9 for k as it's the smallest value of k when the test set RMSE dips

(c). (Extra Credit) (10 pts) Write an R script containing R functions that calculate the LOOCV performance (RMSE) of k-NN for $k = 1$ to 30 on the data set. What value of k would be suitable for this data set? Show the R code and the output you obtained below.

```
# normalize first columns that are not cut and price (these correspond to
# the predictors)
diamond_norm <- as.data.frame(lapply(subsetDiamond[,3:8], nor))

# set random seed
set.seed(1)

# Randomly select 90% of the data
ran <- sample(1:nrow(subsetDiamond), floor(0.9 * nrow(subsetDiamond)))

# extract training set
diamond_trainX <- diamond_norm[ran,]
diamond_trainY <- subsetDiamond[ran,2]


# extract testing set
diamond_testX <- diamond_norm[-ran,]
diamond_testY <- subsetDiamond[-ran,2]

# plot kNN RMSE vs k
numk <- 30 # number of values of k to consider
kvalues <- 1:numk

# calculate RMSE on the test set for
# different values of k
RMSETestSet <- numeric(numk)
for (k in 1:numk) {

  # run knnreg function
  fit <- knnreg(diamond_trainX, diamond_trainY, k)

  # predict on test set
  diamond_pred <- predict(fit, diamond_testX)

  # calculate rmse
  RMSETestSet[k] <- rmse(diamond_testY,diamond_pred)

  print(paste('RMSE: ',RMSETestSet[k]))

}
```

```
## [1] "RMSE:  1835.27092182094"
## [1] "RMSE:  1717.97898335306"
## [1] "RMSE:  1634.36154912278"
## [1] "RMSE:  1581.08840708791"
## [1] "RMSE:  1557.29904759158"
## [1] "RMSE:  1487.55822816334"
## [1] "RMSE:  1437.16161991835"
## [1] "RMSE:  1397.39531369312"
## [1] "RMSE:  1362.73015759565"
## [1] "RMSE:  1364.43141251284"
## [1] "RMSE:  1344.76404731261"
## [1] "RMSE:  1355.93309323947"
## [1] "RMSE:  1351.225313474"
## [1] "RMSE:  1342.26918921302"
## [1] "RMSE:  1338.63324028194"
## [1] "RMSE:  1330.78053394935"
## [1] "RMSE:  1317.74495695005"
## [1] "RMSE:  1313.8865137306"
## [1] "RMSE:  1313.56261839552"
## [1] "RMSE:  1307.38682820775"
## [1] "RMSE:  1300.59475969846"
## [1] "RMSE:  1298.83719186096"
## [1] "RMSE:  1285.19480045527"
## [1] "RMSE:  1296.02739869302"
## [1] "RMSE:  1287.66198352893"
## [1] "RMSE:  1298.52122443028"
## [1] "RMSE:  1302.71529246459"
## [1] "RMSE:  1299.4701166059"
## [1] "RMSE:  1303.29264946342"
## [1] "RMSE:  1301.97140847549"
```

```
k_best <- which.min(RMSETestSet)
print(paste('The best RMSE is ',min(RMSETestSet),' at k =',k_best))
```

```
## [1] "The best RMSE is  1285.19480045527  at k = 23"
```