Exercise 2

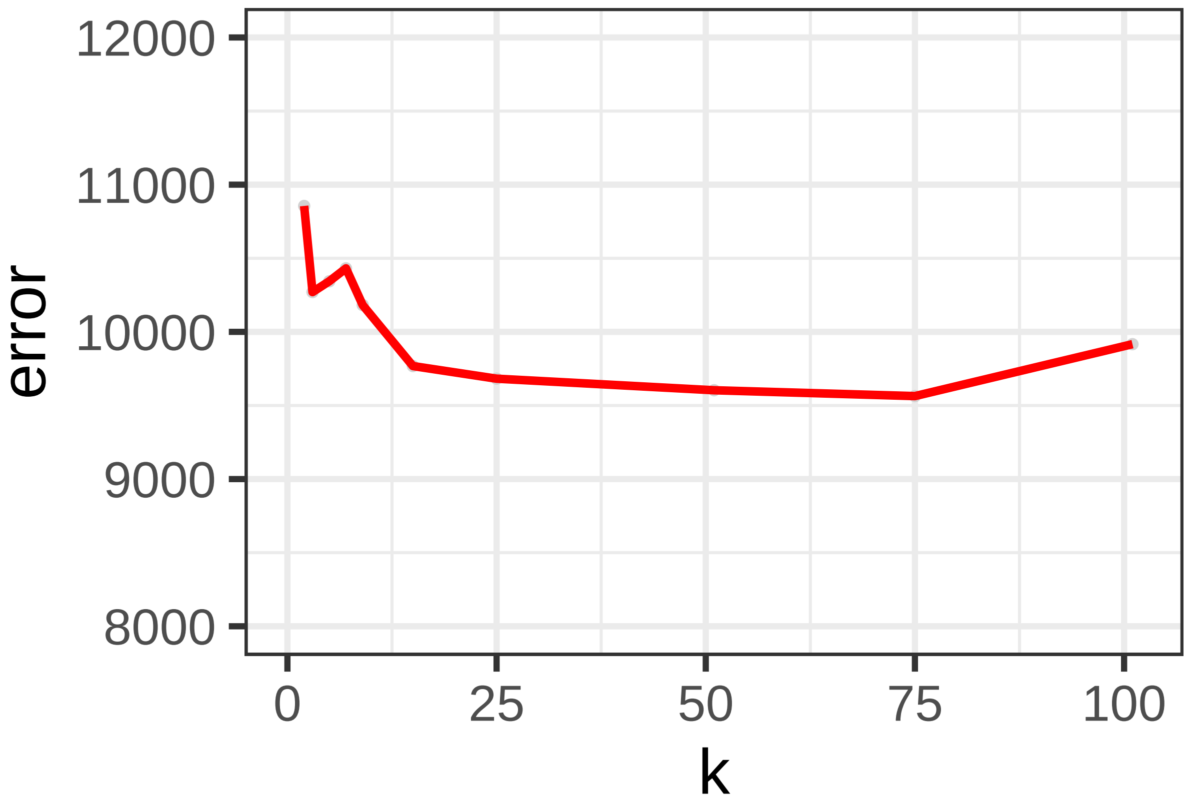
KNN practice

Setting up the data:

```
carData <- read.csv("~/Documents/SDS323Assignments/sclass.csv")
carData350 <- filter(carData, trim == 350)
n = length(carData350$trim)
n_train = round(0.8*n)
n_test = n - n_train
train_ind = sample.int(n, n_train)
y = carData350['price']$price
X = carData350['mileage']
X_train = X[train_ind,]
X_test = X[-train_ind,]
y_train = y[train_ind]
y_test = y[-train_ind]
knn_trainset = data.frame(X_train, y_train = y_train)
knn_testset = data.frame(X_test, y_test = y_test)
X_train = knn_trainset['X_train']
X_test = knn_testset['X_test']
```

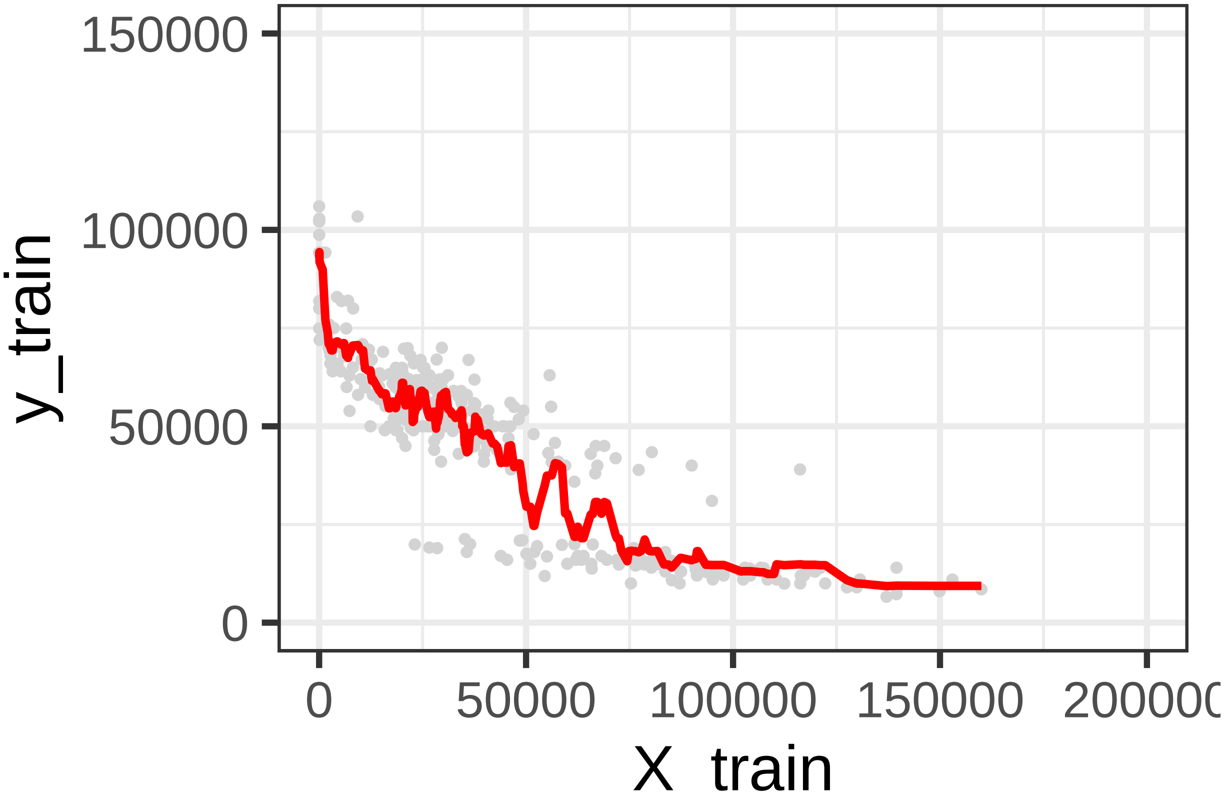
Analysis

```
df = data.frame(k=integer(0),error=numeric(0))
kValues = c(2, 3, 5, 7, 9, 15, 25, 51, 75, 101)
for(i in kValues)
{
  knn = knn.reg(train = X_train, test = X_test, y = y_train, k = i)
  err = (sum((y_test - knn['pred']$pred) ^ 2)/n_test)^0.5
  df = add_row(df, k = i, error = err)
}
p_train = ggplot(data = df) +
  geom_point(mapping = aes(x = k, y = error), color='lightgrey') +
  theme_bw(base_size=24) +
  ylim(8000, 12000) + xlim(0,102)
p_train + geom_path(mapping = aes(x=k, y=error), color='red', size=1.5)
```



We observe that k=9 seems to be optimal.

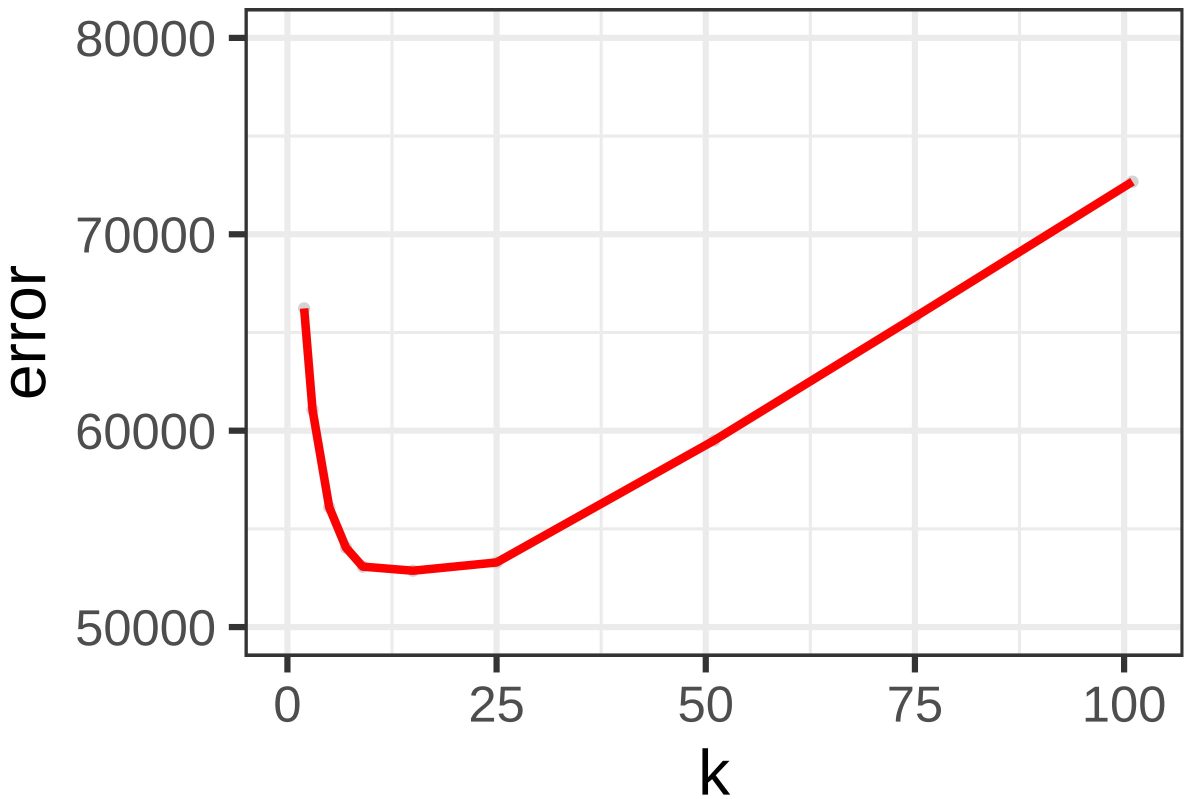
```
#We fit the model on the train set.
knn = knn.reg(train = X_train, test = X_train, y = y_train, k = 9)
X_train2 = X_train
X_train2$pred = knn$pred
X_train2$y_train = y_train
X_train2 = X_train2[order(X_train),] #We sort to make graph look smoother
p_train = ggplot(data = X_train2) +
  geom_point(mapping = aes(x = X_train, y = y_train), color='lightgrey') +
  theme_bw(base_size=24) +
  ylim(0, 150000) + xlim(0, 200000)
p_train + geom_path(mapping = aes(x = X_train, y = pred), color='red', size=1.5)
```



Repeat for trim = 63:

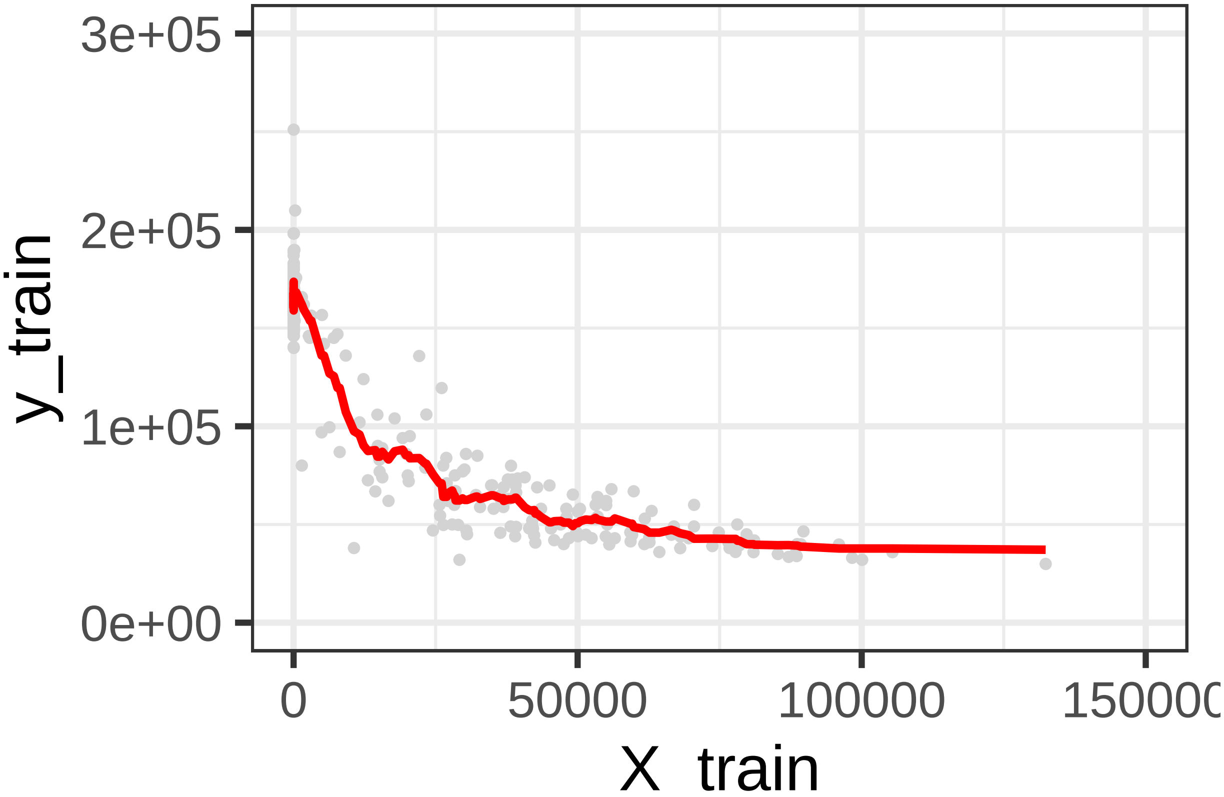
```
carData <- read.csv("~/Documents/SDS323Assignments/sclass.csv")
carData63 <- filter(carData, trim == '63 AMG')
n = length(carData350$trim)
n_train = round(0.8*n)
n_test = n - n_train
train_ind = sample.int(n, n_train)
y = carData63['price']$price
X = carData63['mileage']
X_train = X[train_ind,]
X_test = X[-train_ind,]
y_train = y[train_ind]
y_test = y[-train_ind]
knn_trainset = data.frame(X_train, y_train = y_train)
knn_testset = data.frame(X_test, y_test = y_test)
X_train = knn_trainset['X_train']
X_test = knn_testset['X_test']
kValues = c(2, 3, 5, 7, 9, 15, 25, 51, 75, 101)
df = data.frame(k=integer(0),error=numeric(0))
for(i in kValues)
{
  knn = knn.reg(train = X_train, test = X_test, y = y_train, k = i)
  err = (sum((y_test - knn['pred']$pred) ^ 2)/n_test)^0.5
  df = add_row(df, k = i, error = err)
}
p_train = ggplot(data = df) +
  geom_point(mapping = aes(x = k, y = error), color='lightgrey') +
  theme_bw(base_size=24) +
```

```
  ylim(50000, 80000) + xlim(0,102)
p_train + geom_path(mapping = aes(x=k, y=error), color='red', size=1.5)
```



We observe that k=15 seems to be optimal.

```
#We fit the model on the train set.
knn = knn.reg(train = X_train, test = X_train, y = y_train, k = 15)
X_train2 = X_train
X_train2$pred = knn$pred
X_train2$y_train = y_train
X_train2 = X_train2[order(X_train),] #We sort to make graph look smoother
p_train = ggplot(data = X_train2) +
  geom_point(mapping = aes(x = X_train, y = y_train), color='lightgrey') +
  theme_bw(base_size=24) +
  ylim(0, 300000) + xlim(0, 150000)
p_train + geom_path(mapping = aes(x = X_train, y = pred), color='red', size=1.5)
```



Saratoga house prices

```r
data(SaratogaHouses)
summary(SaratogaHouses)
```

```
##      price           lotSize             age           landValue
##  Min.   :  5000   Min.   : 0.0000   Min.   :  0.00   Min.   :   200
##  1st Qu.:145000   1st Qu.: 0.1700   1st Qu.: 13.00   1st Qu.: 15100
##  Median :189900   Median : 0.3700   Median : 19.00   Median : 25000
##  Mean   :211967   Mean   : 0.5002   Mean   : 27.92   Mean   : 34557
##  3rd Qu.:259000   3rd Qu.: 0.5400   3rd Qu.: 34.00   3rd Qu.: 40200
##  Max.   :775000   Max.   :12.2000   Max.   :225.00   Max.   :412600
##    livingArea     pctCollege       bedrooms       fireplaces       bathrooms
##  Min.   : 616   Min.   :20.00   Min.   :1.000   Min.   :0.0000   Min.   :0.0
##  1st Qu.:1300   1st Qu.:52.00   1st Qu.:3.000   1st Qu.:0.0000   1st Qu.:1.5
##  Median :1634   Median :57.00   Median :3.000   Median :1.0000   Median :2.0
##  Mean   :1755   Mean   :55.57   Mean   :3.155   Mean   :0.6019   Mean   :1.9
##  3rd Qu.:2138   3rd Qu.:64.00   3rd Qu.:4.000   3rd Qu.:1.0000   3rd Qu.:2.5
##  Max.   :5228   Max.   :82.00   Max.   :7.000   Max.   :4.0000   Max.   :4.5
##      rooms                     heating            fuel
##  Min.   : 2.000   hot air         :1121   gas     :1197
##  1st Qu.: 5.000   hot water/steam: 302   electric: 315
##  Median : 7.000   electric        : 305   oil     : 216
##  Mean   : 7.042
##  3rd Qu.: 8.250
##  Max.   :12.000
##                 sewer      waterfront newConstruction centralAir
```

```
##  septic            : 503   Yes:  15   Yes:  81        Yes: 635
##  public/commercial:1213   No :1713   No :1647        No :1093
##  none              :  12
##
##
##
```

```
n = nrow(SaratogaHouses)
n_train = round(0.8*n)  # round to nearest integer
n_test = n - n_train
rmse = function(y, yhat) {
  sqrt(mean((y - yhat)^2))
}
err = 0
for(i in 1:1000){
train_cases = sample.int(n, n_train, replace=FALSE)
test_cases = setdiff(1:n, train_cases)
saratoga_train = SaratogaHouses[train_cases,]
saratoga_test = SaratogaHouses[test_cases,]
lm2 = lm(price ~ . - sewer - waterfront - landValue - newConstruction, data=saratoga_train)
yhat_test2 = predict(lm2, saratoga_test)
err = err + rmse(saratoga_test$price, yhat_test2)}
err / 1000
```

```
## [1] 66528.04
```

```
lm2$coefficients
```

```
##           (Intercept)              lotSize                  age
##           26745.40839          10834.72184             62.23539
##            livingArea           pctCollege             bedrooms
##              92.73855            333.52890         -16582.34341
##            fireplaces            bathrooms                rooms
##             383.80654          23179.41650           3124.99636
## heatinghot water/steam      heatingelectric          fuelelectric
##           -7910.50047          -4213.19521          -8971.10987
##               fueloil         centralAirNo
##          -14364.06609         -18713.56836
```

Our baseline model error is around 66,000 to 67,000. I observed by looking at the data that the houses with the highest prices tended to have pctCollege either 57 or 62, suggesting that those numbers may correspond to neighborhoods that are very affluent. I also took the logarithm of the price and the living area.

```
err = 0
for(i in 1:1000){
train_cases = sample.int(n, n_train, replace=FALSE)
test_cases = setdiff(1:n, train_cases)
saratoga_train = SaratogaHouses[train_cases,]
saratoga_test = SaratogaHouses[test_cases,]
lm2 = lm(log(price) ~ . + (pctCollege == 62) + (pctCollege == 57) + log(livingArea) - livingArea - sewe
yhat_test2 = predict(lm2, saratoga_test)
err = err + rmse(saratoga_test$price, exp(yhat_test2))
}
err / 1000
```

```
## [1] 62410.69
```

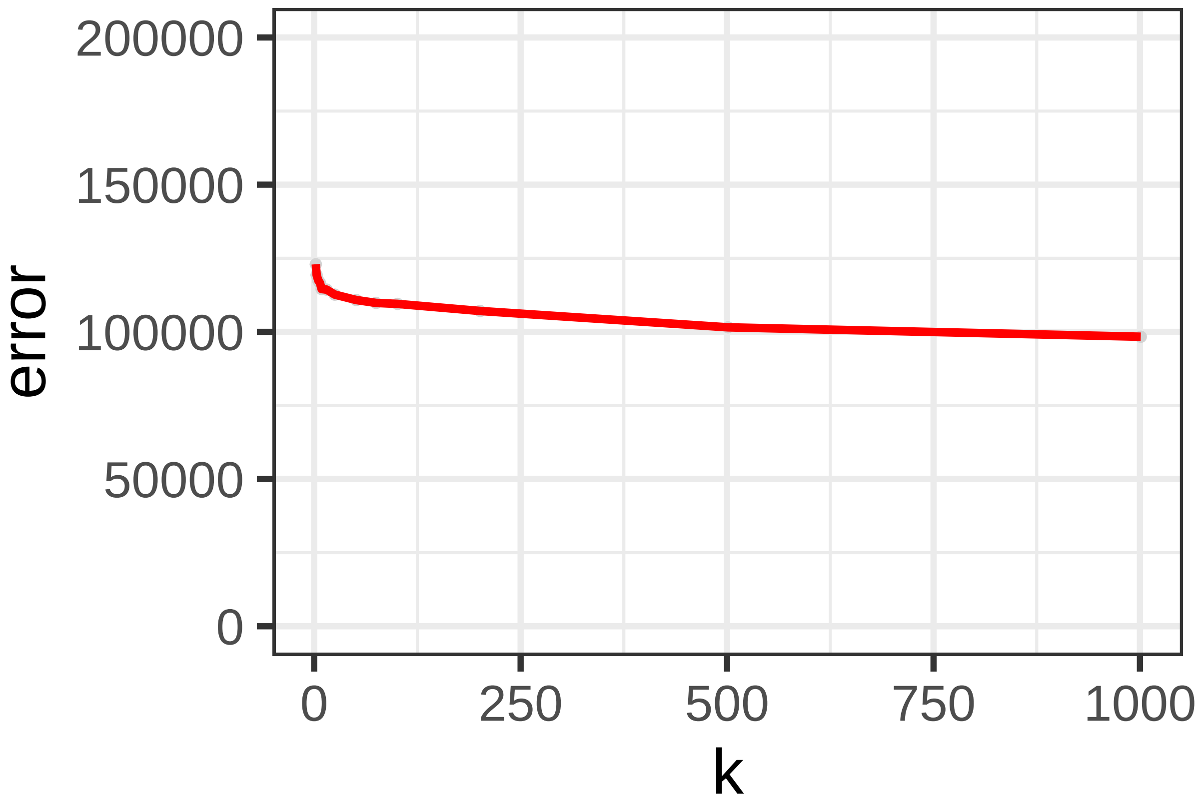
```r
lm2$coefficients
```

```
##           (Intercept)               lotSize                   age
##          7.0939046795          0.0486831368         -0.0009170303
##            pctCollege              bedrooms            fireplaces
##          0.0007184264         -0.0229629857          0.0253220223
##             bathrooms                 rooms heatinghot water/steam
##          0.0927379041          0.0153758788         -0.0251828490
##       heatingelectric          fuelelectric               fueloil
##          0.0088849690         -0.0683223771         -0.0100012658
##          centralAirNo   pctCollege == 62TRUE   pctCollege == 57TRUE
##         -0.0365516071          0.3353078043          0.1455671217
##       log(livingArea)
##          0.6471779216
```

By adding the indications of those neighborhoods, the error dropped to around 62000 to 63000, more than 6% lower than the base model. While obviously, the pctCollege being 57 or 62 may not be significant more, the improvement in the model error does indicate that location may be a major factor, and that adding another variable measuring the wealth of the location of the house may be very helpful. We now turn to KNN analysis, using the same variables above:

```r
kValues = c(2, 3, 5, 7, 9, 15, 25, 51, 75, 101, 201, 501, 1001)
df = data.frame(k=integer(0),error=numeric(0))
for (j in kValues)
{
err = 0

for(i in 1:100)
{
train_cases = sample.int(n, n_train, replace=FALSE)
test_cases = setdiff(1:n, train_cases)
xData = model.matrix(~ .   - sewer - waterfront - landValue - newConstruction - 1 - price, data = Sarat
yData = model.matrix(~log(price) - 1, data = SaratogaHouses)
X_train = xData[train_cases,]
X_test = xData[test_cases,]
y_train = yData[train_cases,]
y_test = yData[test_cases,]
scaling = apply(X_train, 2, sd)
X_train_scaled = scale(X_train, scale = scaling)
X_test_scaled = scale(X_test, scale = scaling)
knn = knn.reg(train = X_train_scaled, test = X_test_scaled, y = y_train, k = j)
err = err + rmse(saratoga_test$price, exp(knn['pred']$pred))
}
df = add_row(df, k = j, error = err / 100)
}
p_train = ggplot(data = df) +
  geom_point(mapping = aes(x = k, y = error), color='lightgrey') +
  theme_bw(base_size=24) +
  ylim(0, 200000) + xlim(0,1002)
p_train + geom_path(mapping = aes(x=k, y=error), color='red', size=1.5)
```



The performance of KNN is much worse on this set of data; with error never falling below 100,000. ## Online News

```
data <- read.csv("~/Documents/SDS323Assignments/online_news.csv")[,-1]
n = nrow(data)
n_train = round(0.8*n)  # round to nearest integer
n_test = n - n_train
TP = 0
FP = 0
TN = 0
FN = 0
for(i in 1:100){
train_cases = sample.int(n, n_train, replace=FALSE)
test_cases = setdiff(1:n, train_cases)
trainData = data[train_cases,]
testData = data[test_cases,]
lm2 = lm(log(shares) ~ . - weekday_is_sunday - is_weekend, data=trainData)
#remove redundant variables
yhat_test2 = predict(lm2, testData)
pred = (exp(yhat_test2) > 1400)
actual = (testData$shares > 1400)
TP = TP + (sum(actual & pred)) / n_test
TN = TN + (sum(!actual & !pred)) / n_test
FP = FP + (sum(!actual & pred)) / n_test
FN = FN + (sum(actual & !pred)) / n_test
}
paste("True positive: ", round(TP / 100, digits = 3))
```

```
## [1] "True positive:  0.421"
```

```r
paste("True negative: ", round(TN / 100, digits = 3))
```

```
## [1] "True negative:  0.165"
```

```r
paste("False positive: ", round(FP / 100, digits = 3))
```

```
## [1] "False positive:  0.342"
```

```r
paste("False negative: ", round(FN / 100, digits = 3))
```

```
## [1] "False negative:  0.072"
```

```r
paste("Accuracy: ", round((TP + TN) / 100, digits = 3))
```

```
## [1] "Accuracy:  0.586"
```

```r
nullModel = sum(data$shares > 1400) / n
paste("Null model accuracy: ", round(max(nullModel, 1 - nullModel), 3))
```

```
## [1] "Null model accuracy:  0.507"
```

We observe that our correct prediction rate is somewhat better than the null, but there are a lot of false positives.

```r
data <- read.csv("~/Documents/SDS323Assignments/online_news.csv")[,-1]
n = nrow(data)
n_train = round(0.8*n)  # round to nearest integer
n_test = n - n_train
TP = 0
FP = 0
TN = 0
FN = 0
for(i in 1:100){
train_cases = sample.int(n, n_train, replace=FALSE)
test_cases = setdiff(1:n, train_cases)
trainData = data[train_cases,]
trainData$shares = ifelse(trainData$shares > 1400, 1, 0)
testData = data[test_cases,]
testData$shares = ifelse(testData$shares > 1400, 1, 0)
logit = glm(shares ~ . - weekday_is_sunday - is_weekend, data=trainData, family='binomial')
#remove redundant variables
yhat_test2 = predict(logit, testData)
pred = (yhat_test2 > 0)
TP = TP + (sum(actual & pred)) / n_test
TN = TN + (sum(!actual & !pred)) / n_test
FP = FP + (sum(!actual & pred)) / n_test
FN = FN + (sum(actual & !pred)) / n_test
}
paste("True positive: ", round(TP / 100, digits = 3))
```

```
## [1] "True positive:  0.247"
```

```r
paste("True negative: ", round(TN / 100, digits = 3))
```

```
## [1] "True negative:  0.255"
```

```r
paste("False positive: ", round(FP / 100, digits = 3))
```

```
## [1] "False positive:  0.25"
```

```r
paste("False negative: ", round(FN / 100, digits = 3))
```

```
## [1] "False negative:  0.247"
```

```r
paste("Accuracy: ", round((TP + TN) / 100, digits = 3))
```

```
## [1] "Accuracy:  0.502"
```

```r
nullModel = sum(data$shares > 1400) / n
paste("Null model accuracy: ", round(max(nullModel, 1 - nullModel), 3))
```

```
## [1] "Null model accuracy:  0.507"
```

Our prediction is now no better than the null. I think the regress-first method is more accurate because the boundary is arbitrarily defined; it is not clear why 0 and 1399 should be treated the same but 1399 and 1401 treated differently when training the model, which is what the threshold-first does.