41204-01 Machine Learning | Professor Mladen Kolar

Homework 1

Patrick Miller, Vandana Ramakrishnan, Nathaniel Matare, Ernie Mori, Jordan Bell-Masterson

January 21, 2017

Problem 1

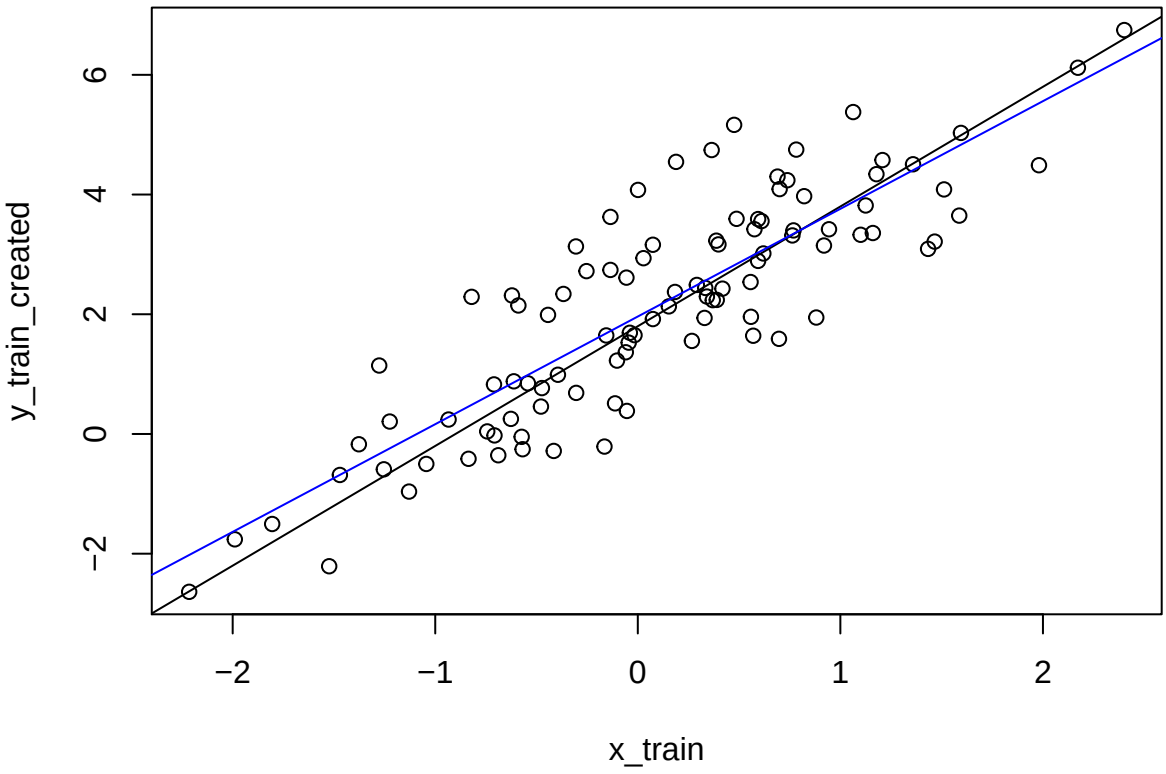
1.1

We simulate the data in accordance with section 1.

```
set.seed(1)
x_train = rnorm(100)
epsilon = rnorm(100)
y_train_created = 1.8*x_train+2+epsilon
x_test = rnorm(10000)
x_test=x_test[order(x_test)]
epsilon_test = rnorm(10000)
y_test_created = 1.8*x_test+2+epsilon_test
```

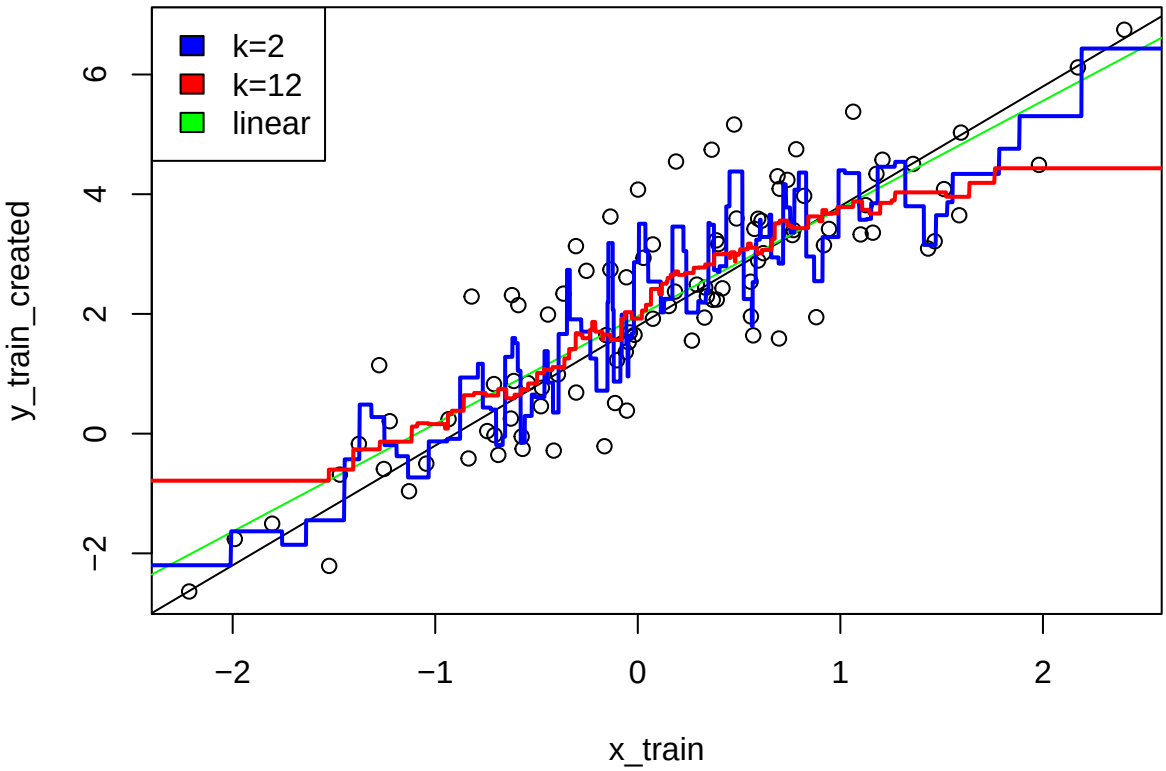
1.2 / 1.3

We plot the true relationship and the best fit line.



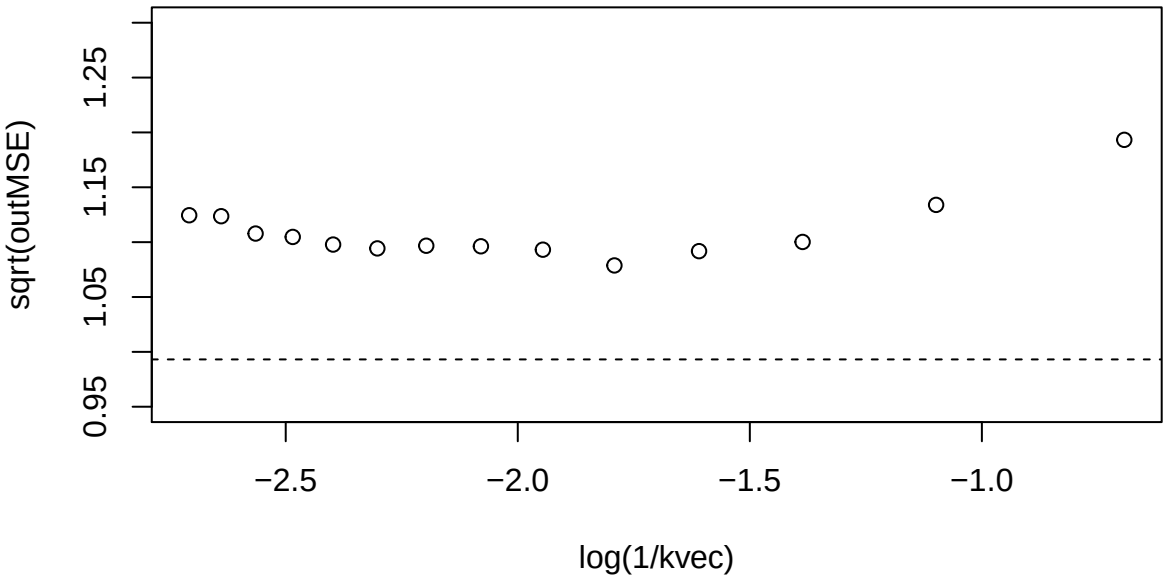
1.4

We show the linear model, KNN at K = 2, and KNN at K = 12



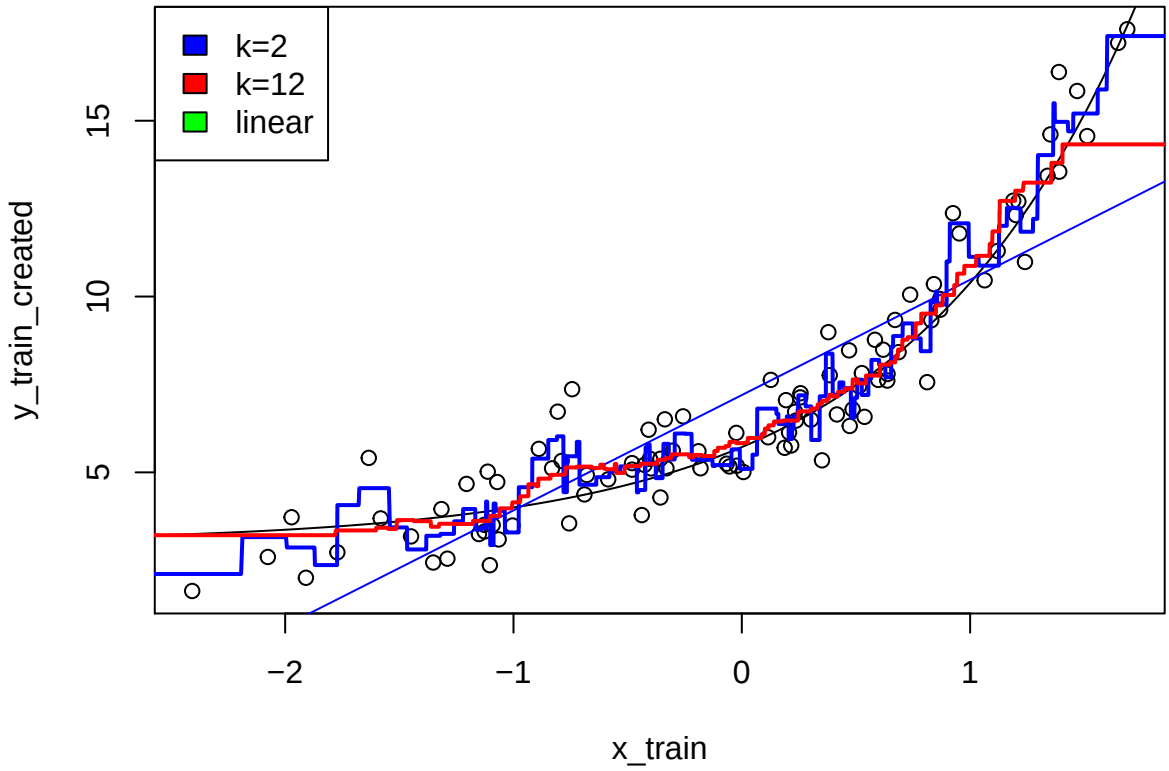
1.5

We show the RMSE vs KNN at K = 2:15

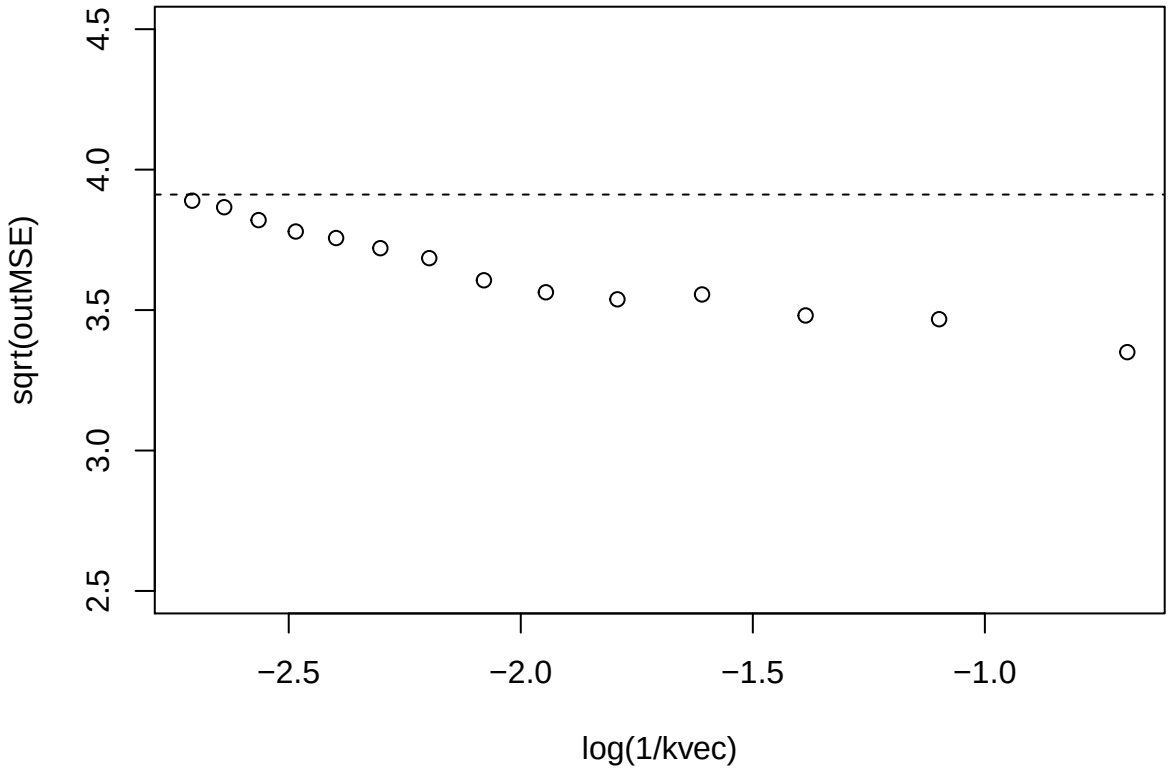


1.6

We show the linear model, KNN at K = 2, and KNN at K = 12

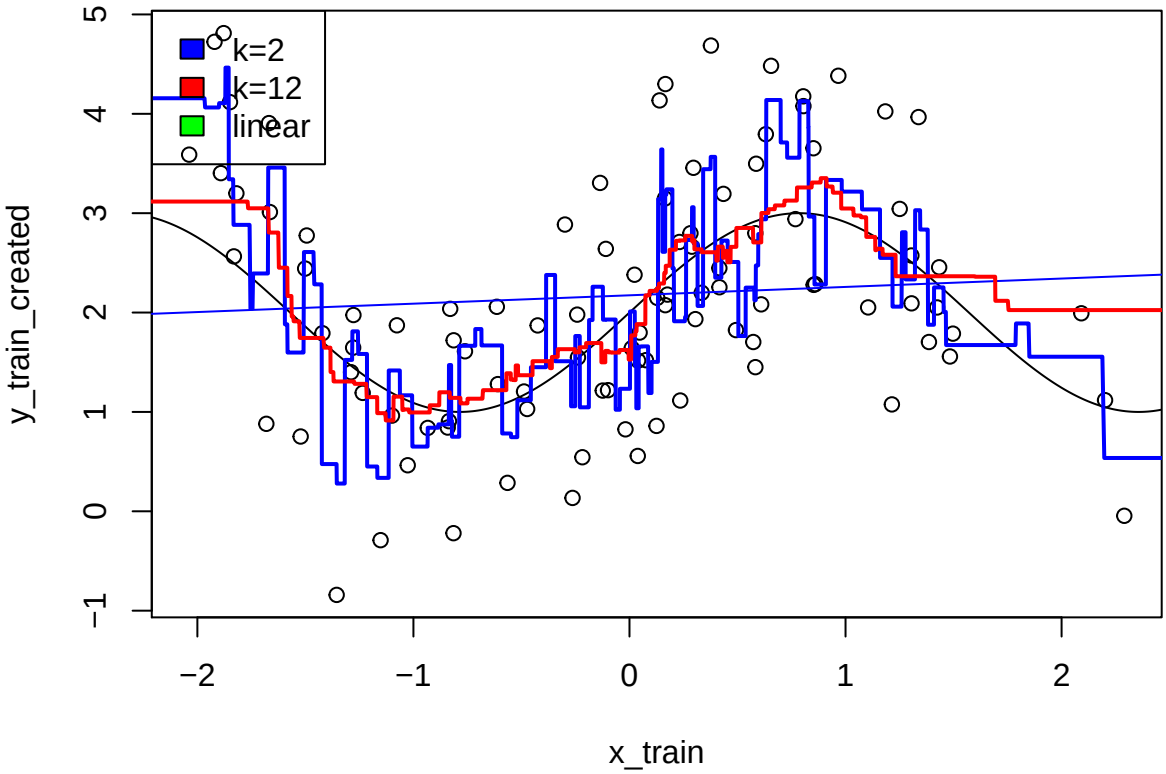


We show the RMSE vs KNN at K = 2:15

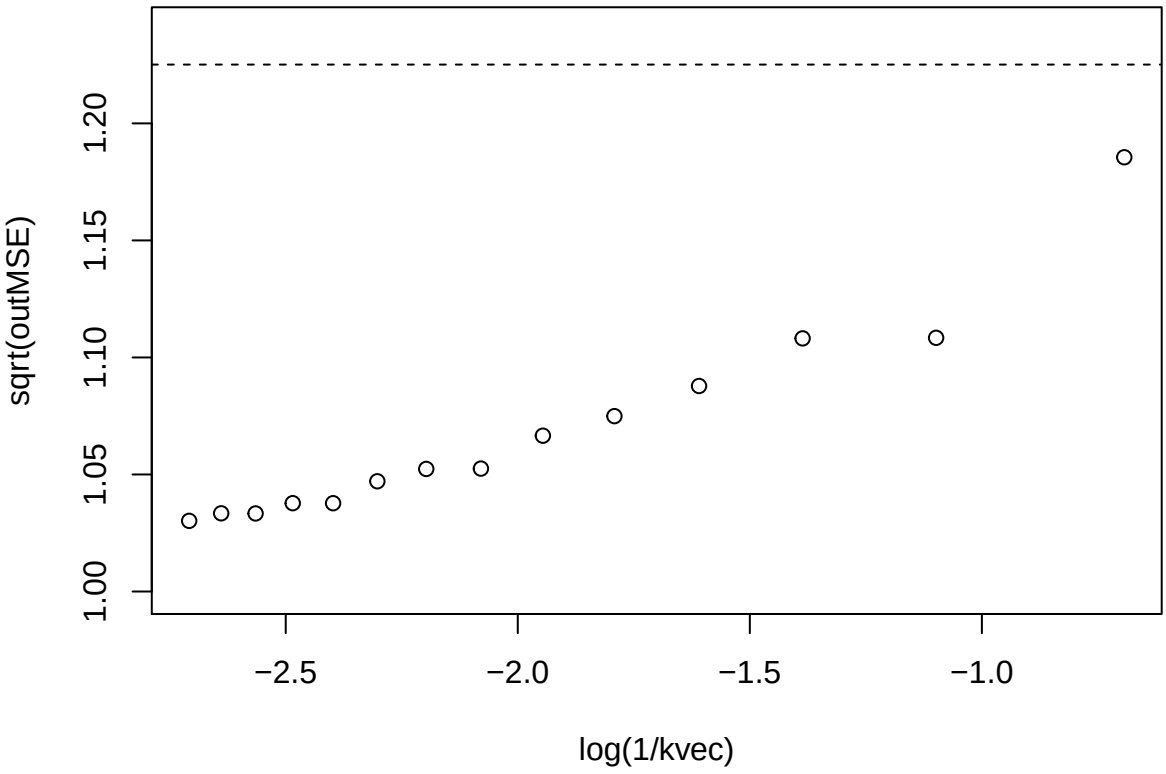


1.7

We show the linear model, KNN at K = 2, and KNN at K = 12

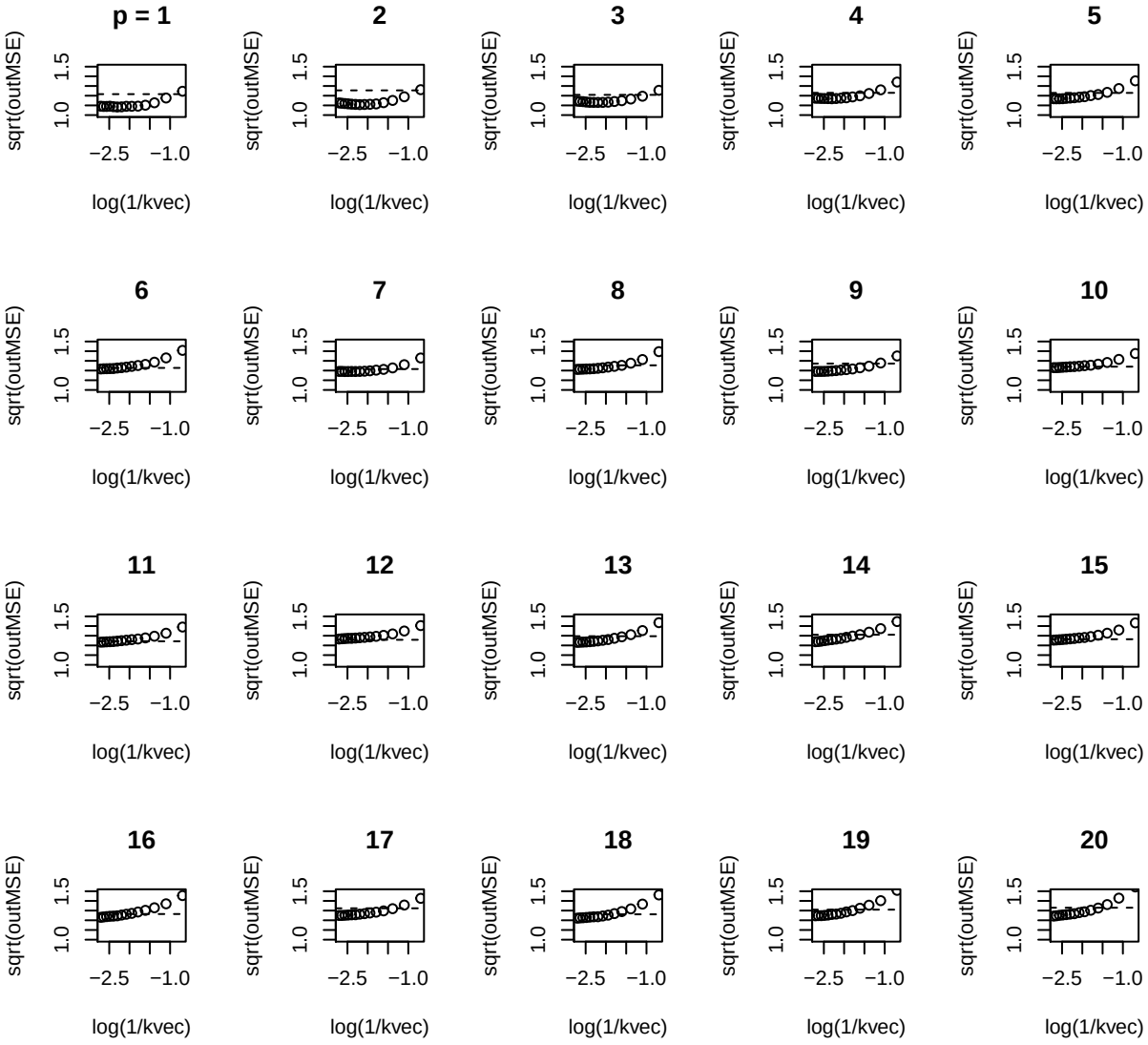


We show the RMSE vs KNN at K = 2:15



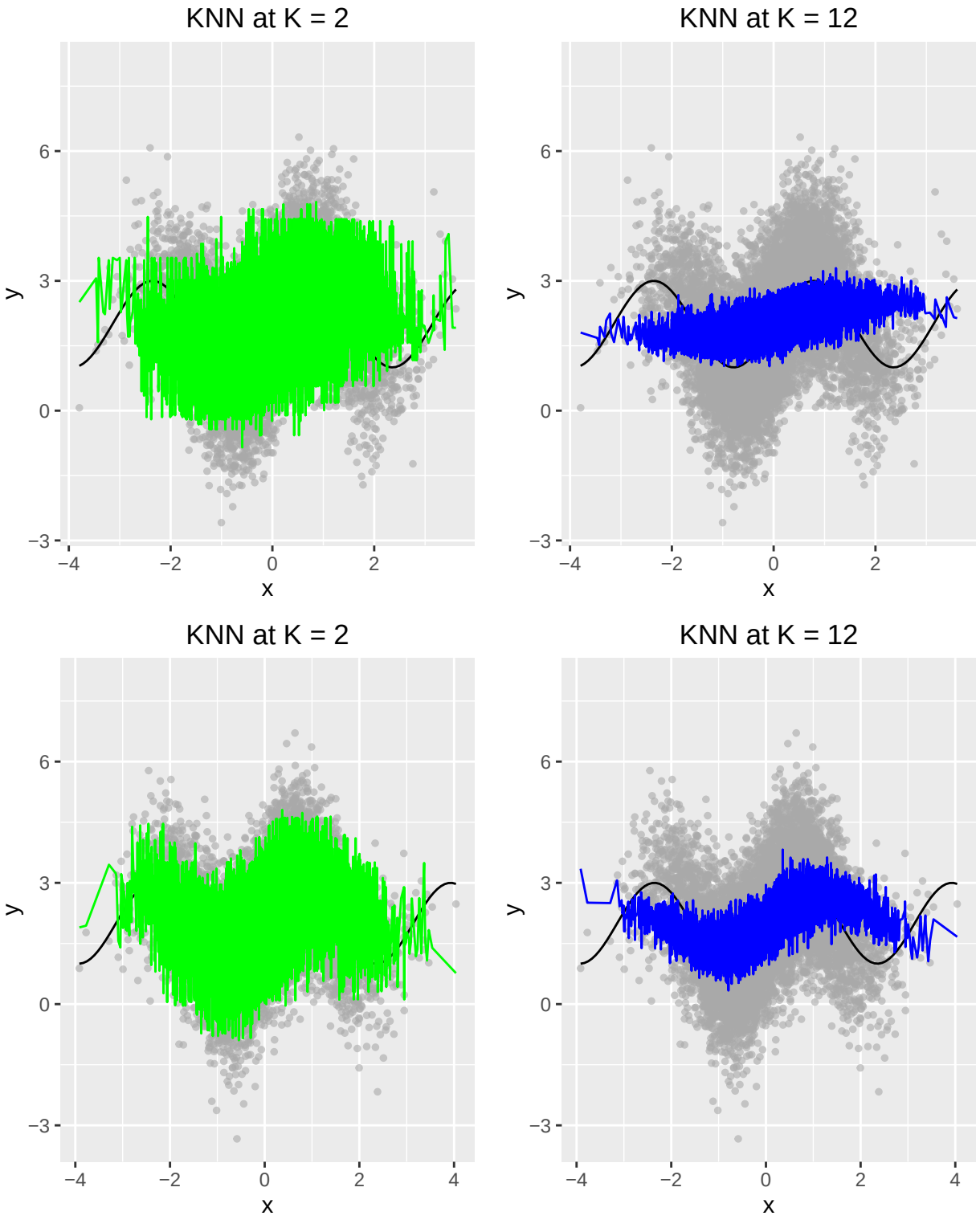
1.8

The superfluous features have no predictive power. Thus, when the amount of noise increases in the dataset, the Knn algorithm uses spurious features to predict the value of y. If K is small and the number of superfluous features is large, then there is a high likelihood that the algorithm uses many erroneous covariates to attempt to predict y. As K increases, the algorithm uses more features to predict y. That is, as the likelihood of the number of spurious features decreases, giving a lower MSE. Please see the below graphs depicting the decrease in MSE as the amount of noise in the dataset increases. KNN, at all values of K, decreases in accuracy as the amount of noise increases.



Bonus

Holding the amount of noise fixed, as the training dataset increases in size, the likelihood of superfluous features chosen to predict y decreases; there is a greater likelihood that KNN will select features with true predictive power as opposed to simple noise. As before, holding the amount of noise fixed, as K increases, the algorithm uses more features to predict y-hat. Thus, the likelihood that the chosen features are spurious decreases, giving a lower MSE. See the below graphs: the first set of graphs shows a training dataset of 100 while the second set of graphs show a training dataset of 1000. Noise is held constant in both graphs at five (Five columns of random features)



Problem 2

2.1

After inspecting the data, we find that we could use several features to predict used car price. If we were a car dealer, doing so would allow us to better price our vehicles. We could also determine the range of prices customers might be wiling to pay for comparable used cars.

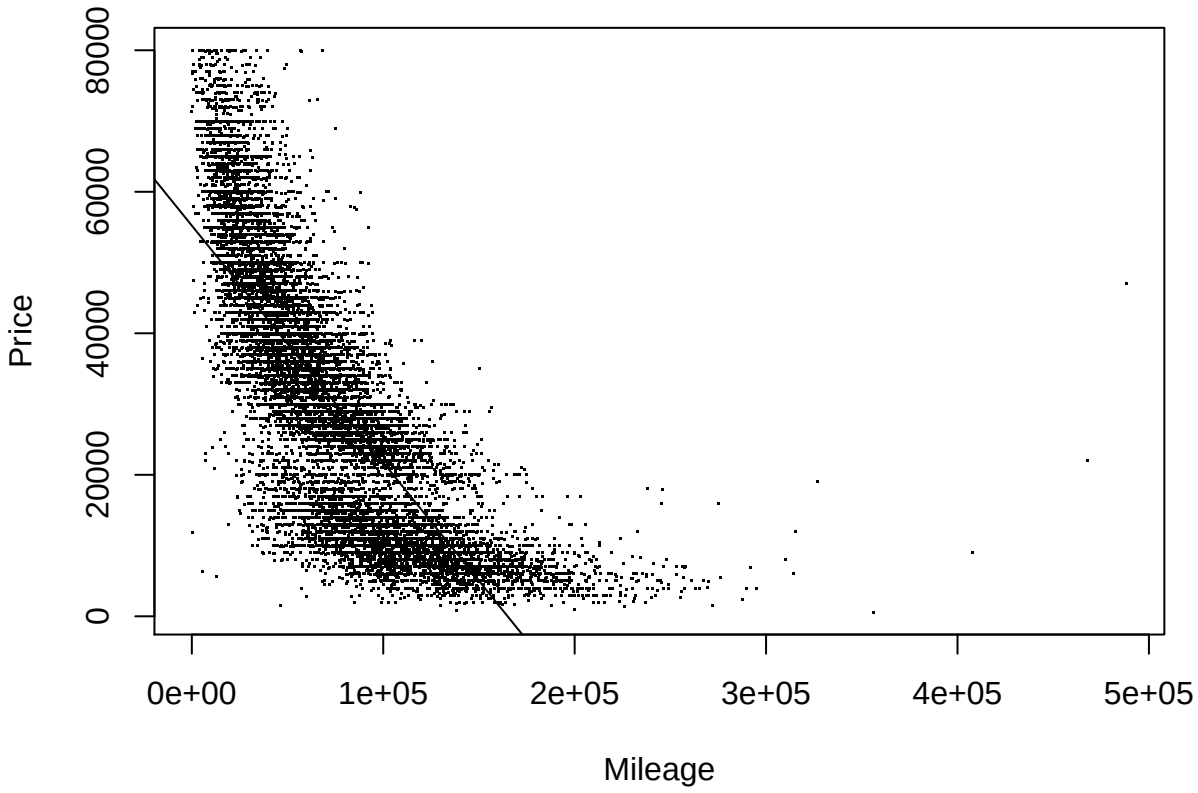
2.2

We split the data into two parts: a training and testing set.

```
set.seed(1)
sample.index=sample(nrow(used.cars),nrow(used.cars)/4)
used.cars.test=used.cars[sample.index,]
used.cars.train=used.cars[-sample.index,]
```

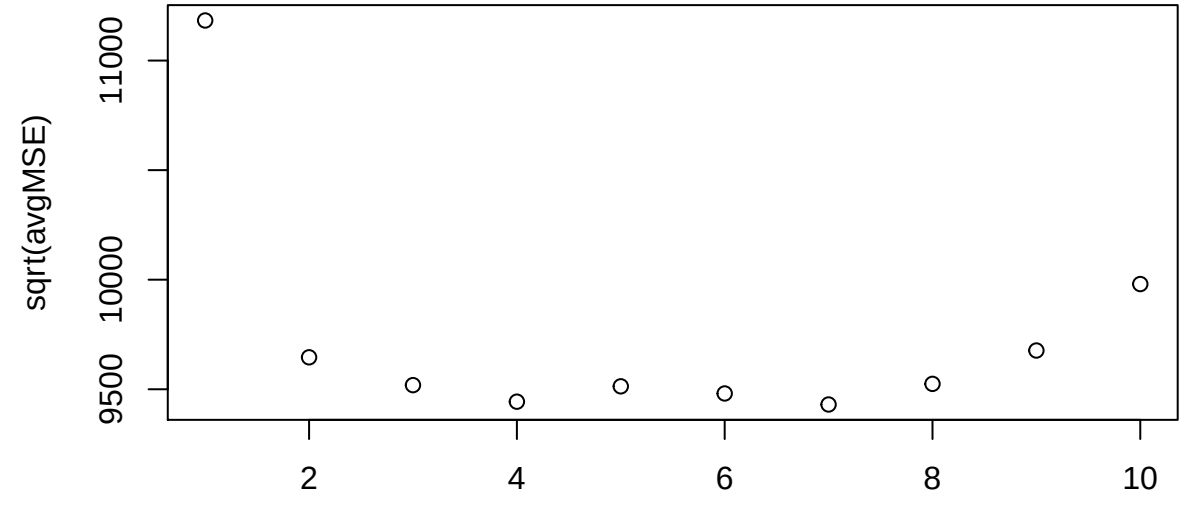
2.3

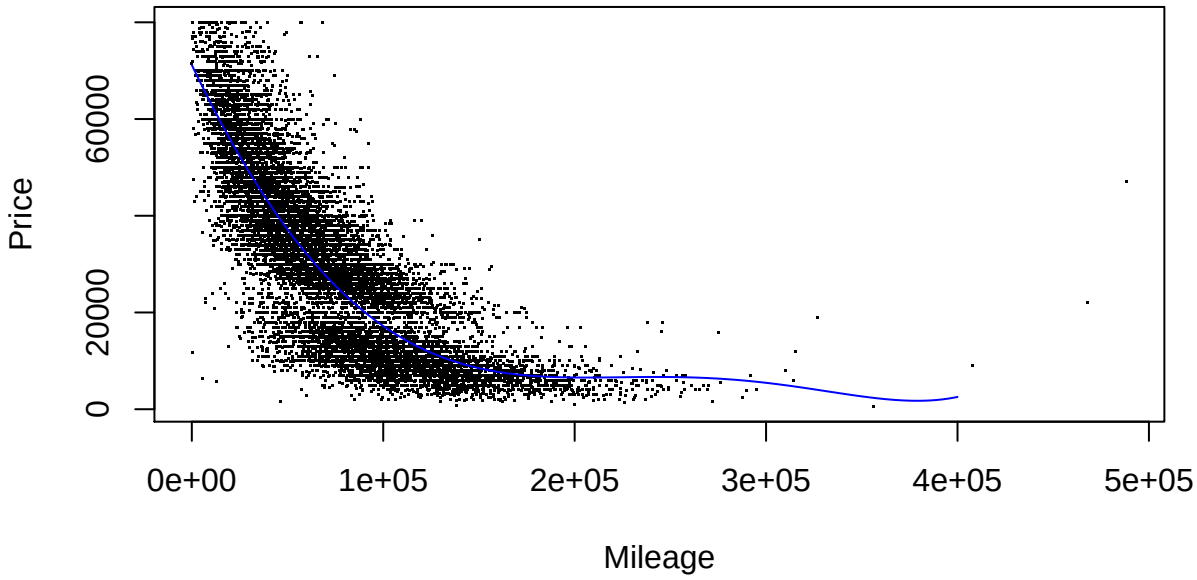
We plot the best fit linear regression onto the data



2.4

We use cross validation to select the optimal polynomial degree. We find that the optimal polynomial degree is five. We next plot the CV MSE as a function of the degree of the polynomial, and a linear polynomial degree five model.

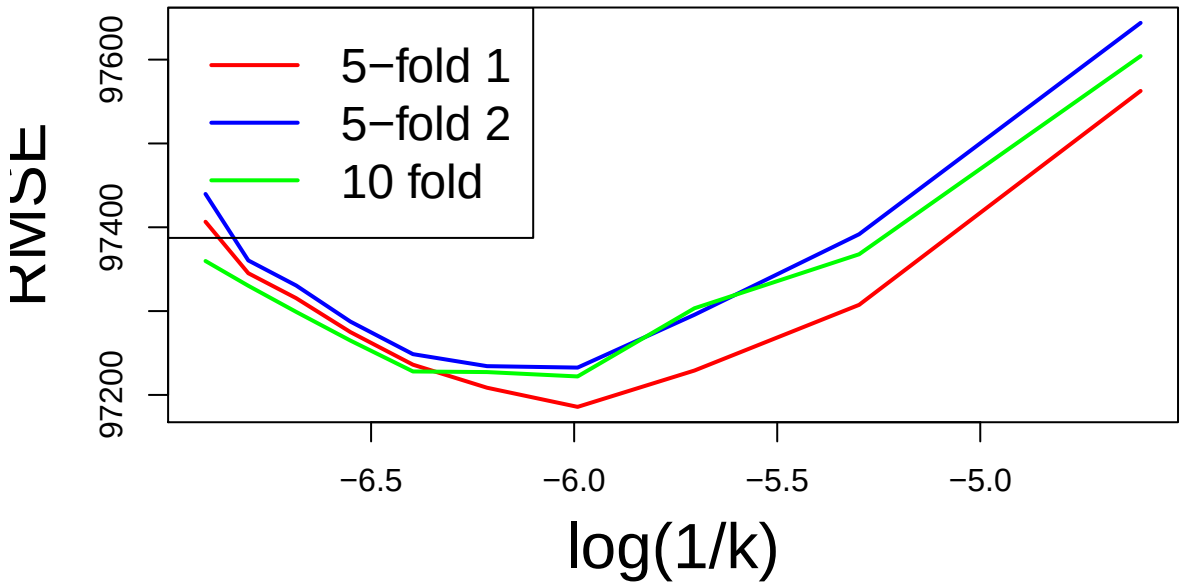


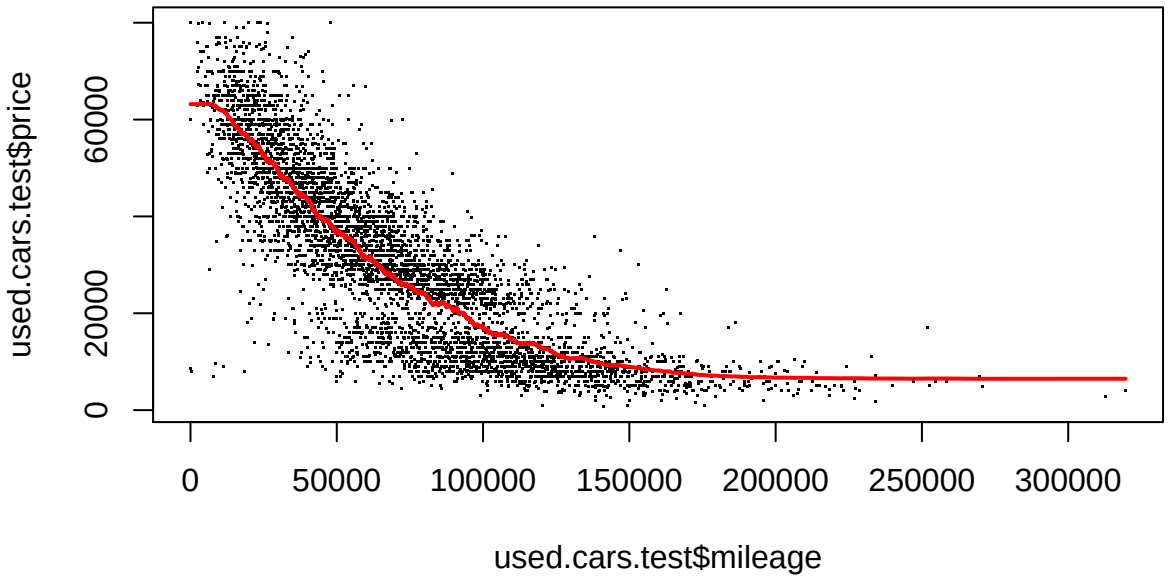


2.5

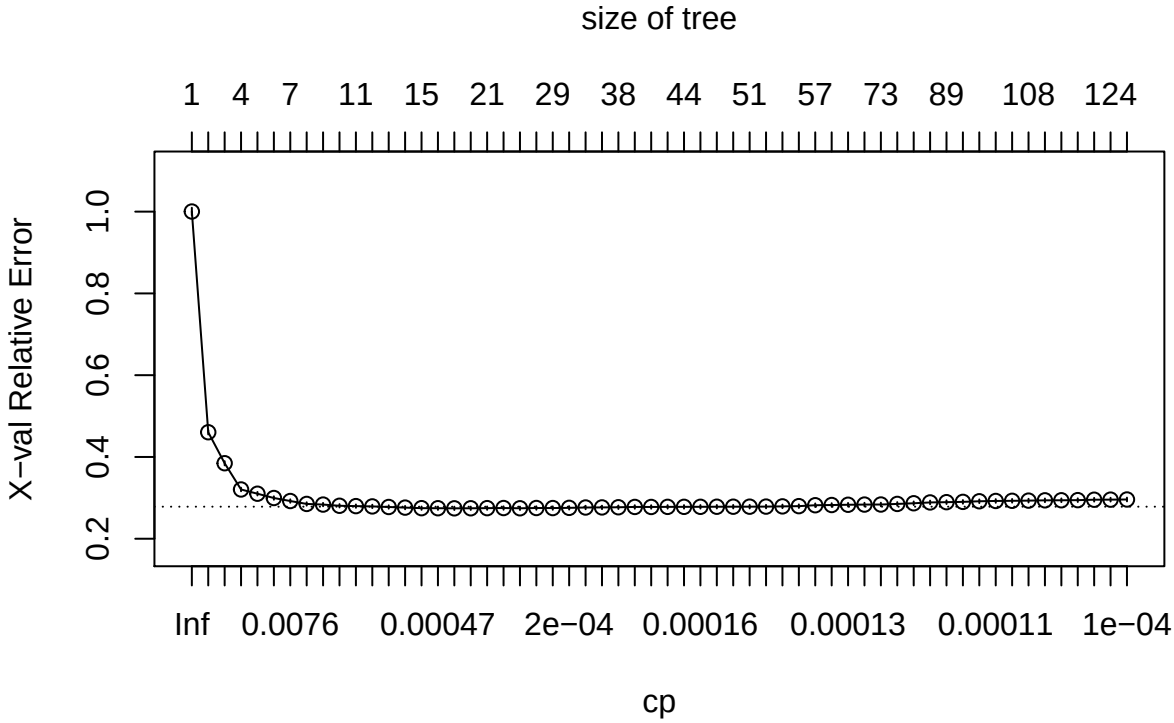
We use cross validation to select the optimal K, and find that MSE is minimized somewhere in the range of K = [400, 600]. We select K = 400 for simplicity. We next plot the CV MSE as a function of the degree of k, and a KNN K = 400 model

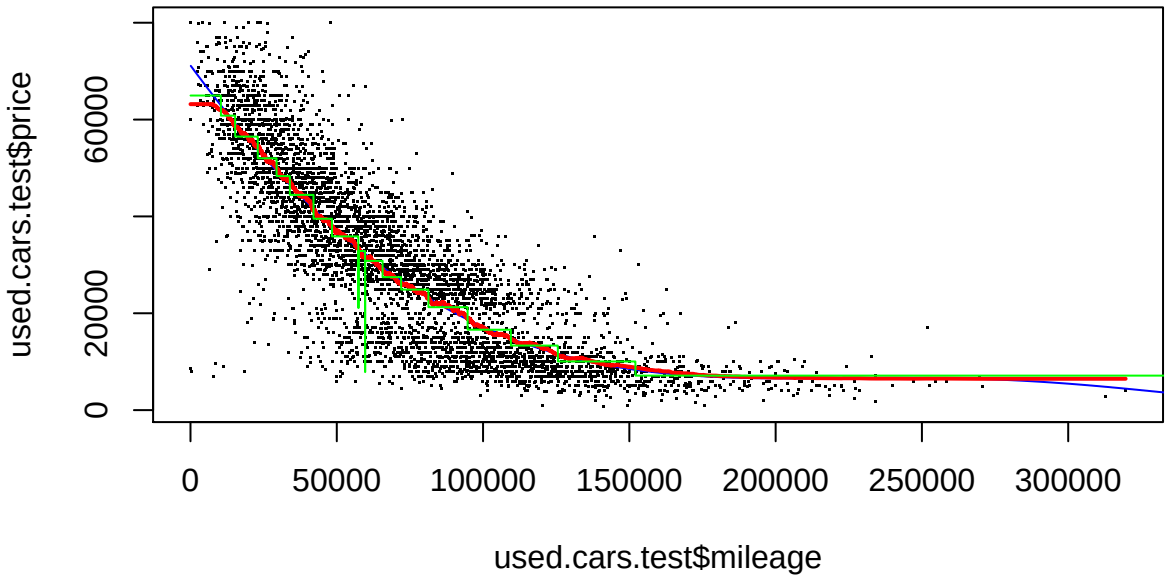
```
## in docv: nset,n,nfold:  10 15048 5
## on fold:  1 , range:  1 : 3010
## on fold:  2 , range:  3011 : 6020
## on fold:  3 , range:  6021 : 9030
## on fold:  4 , range:  9031 : 12040
## on fold:  5 , range:  12041 : 15048

## in docv: nset,n,nfold:  10 15048 5
## on fold:  1 , range:  1 : 3010
## on fold:  2 , range:  3011 : 6020
## on fold:  3 , range:  6021 : 9030
## on fold:  4 , range:  9031 : 12040
## on fold:  5 , range:  12041 : 15048

## in docv: nset,n,nfold:  10 15048 10
## on fold:  1 , range:  1 : 1505
## on fold:  2 , range:  1506 : 3010
## on fold:  3 , range:  3011 : 4515
## on fold:  4 , range:  4516 : 6020
## on fold:  5 , range:  6021 : 7525
## on fold:  6 , range:  7526 : 9030
## on fold:  7 , range:  9031 : 10535
## on fold:  8 , range:  10536 : 12040
## on fold:  9 , range:  12041 : 13545
## on fold:  10 , range:  13546 : 15048
```





We use cross validation to select the complexity parameter for CART, and find that MSE is minimized at alpha = 0.00030, or where the size of our tree is 131. We next plot the relative error as a function of alpha. Our OOS RMSE is minimized at 133.57 when using the KNN model; we select the KNN model.

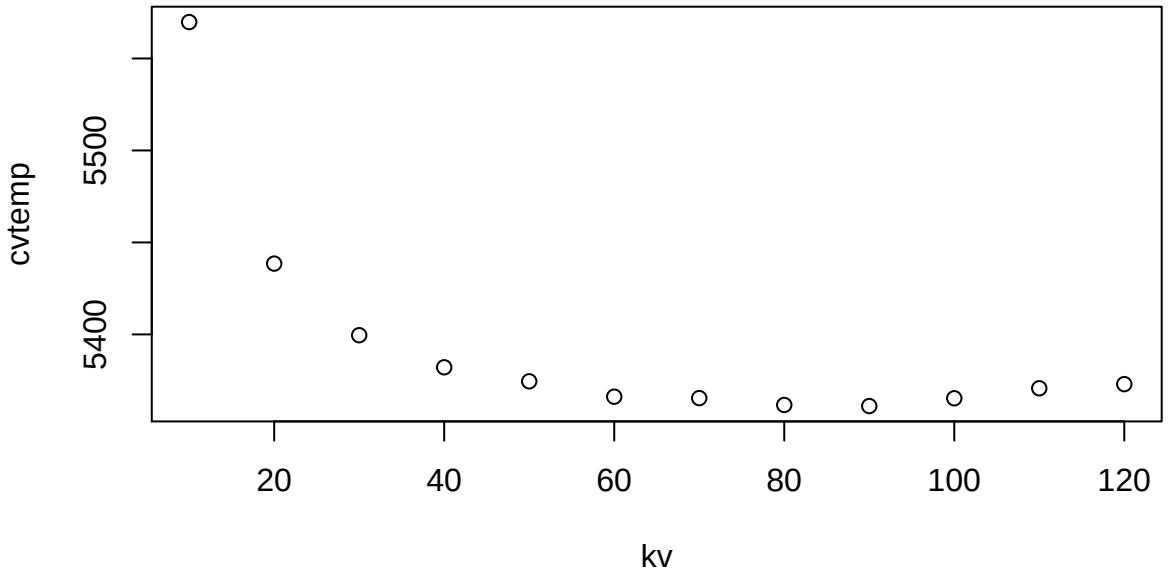


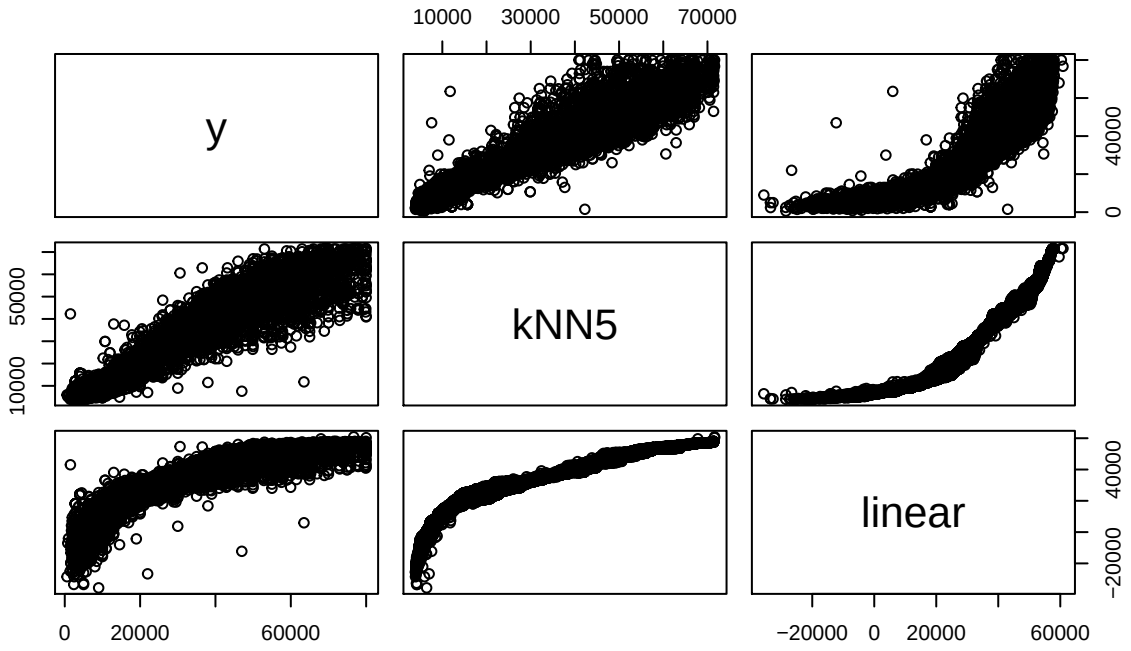

used.cars.test$price
0
20000
60000
0
50000
100000
150000
200000
250000
300000
used.cars.test$mileage

2.6

We now include both year and mileage as features. For KNN we find that our optimal K is now 80 and our optimal complexity parameter for CART is now 0.00019. We next plot the relative error as a function of alpha. Naturally our model performs better when we add more explanatory features; our MSE is now 76.51. For comparison, we should the correlation between the true y and the yhats from the linear, KNN, and tree based model. The size of the tree is 131.

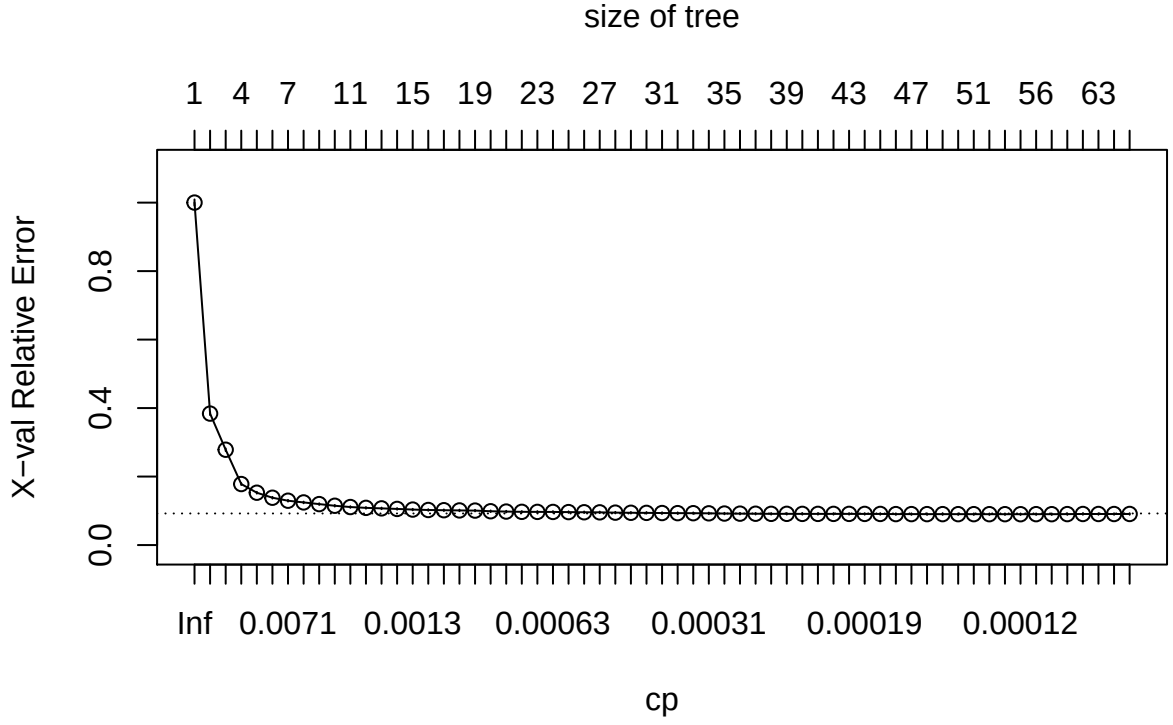
```
## in docv: nset,n,nfold:  12 15048 10
## on fold:  1 , range:  1 : 1505
## on fold:  2 , range:  1506 : 3010
## on fold:  3 , range:  3011 : 4515
## on fold:  4 , range:  4516 : 6020
## on fold:  5 , range:  6021 : 7525
## on fold:  6 , range:  7526 : 9030
## on fold:  7 , range:  9031 : 10535
## on fold:  8 , range:  10536 : 12040
## on fold:  9 , range:  12041 : 13545
## on fold:  10 , range:  13546 : 15048
```





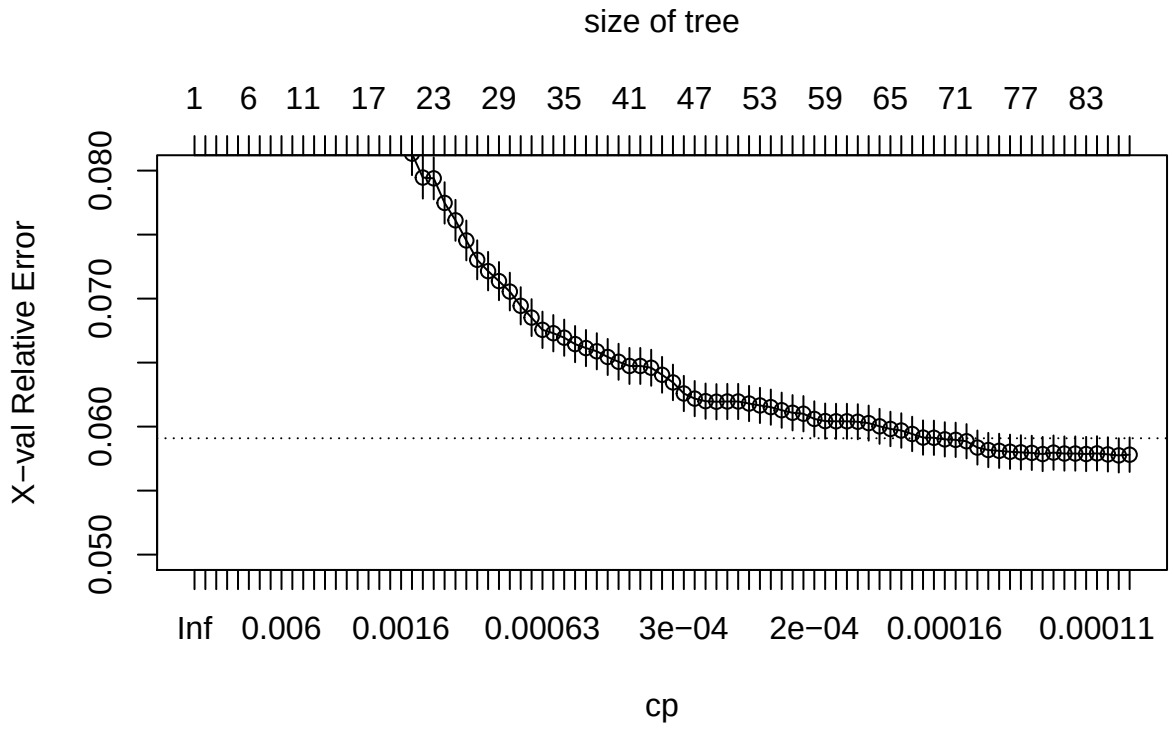

```
##                y      kNN5    linear
## y      1.0000000 0.9569714 0.9109632
## kNN5   0.9569714 1.0000000 0.9522378
## linear 0.9109632 0.9522378 1.0000000
```

```
## [1] 99.47863
```



2.7

We now use all availalbe features in a CART to predict price. We report a MSE of 63.113



Bonus

In order to find the most relevant variables we look towards the complexity parameter output found in the tree output. We note the tabulated results indicate how much each split contributes to improving the 'fit' of the tree model. These are the most important variables. We could now isolate these splits and their respective variables. Then, we could use these variables as the inputs to a more simple, interactive linear model. Because these variables are the most important explanatory features in the dataset, our interacted linear model *should now predict better than a naive linear or polynomial model.