41204-01 Machine Learning | Homework 1

Professor Mladen Kolar

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

January 21, 2017

Problem 1

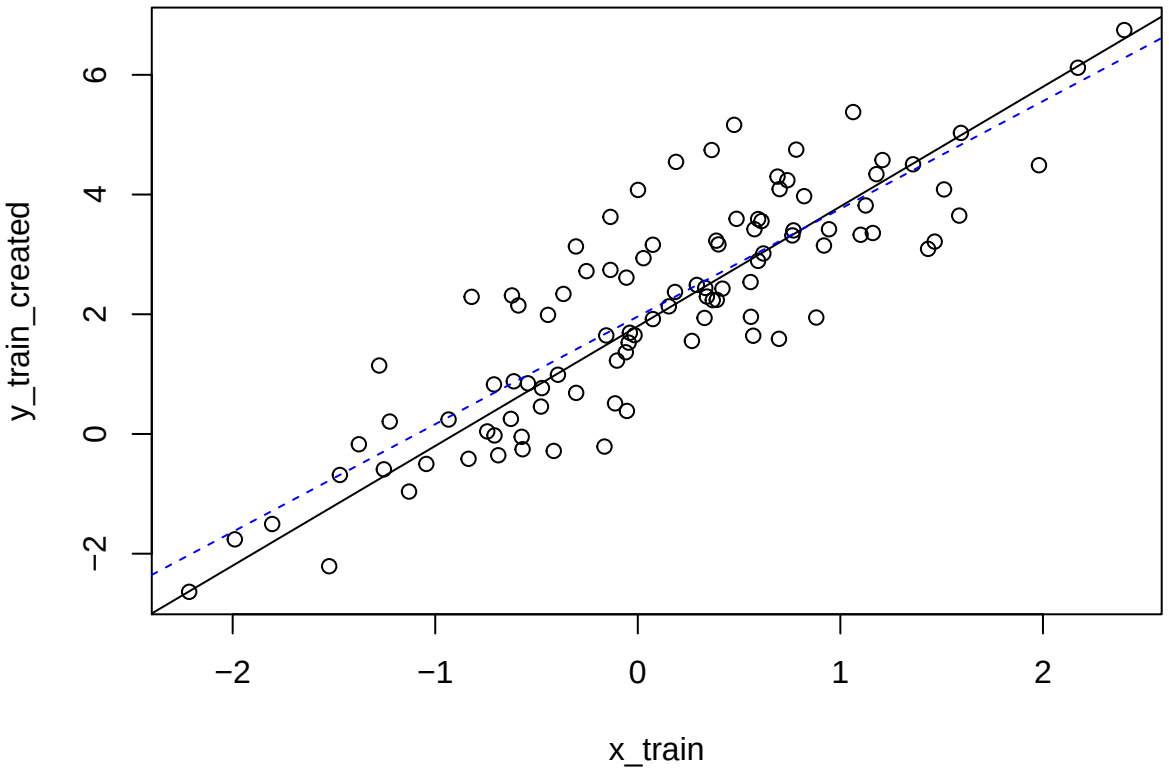
1.1

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
```

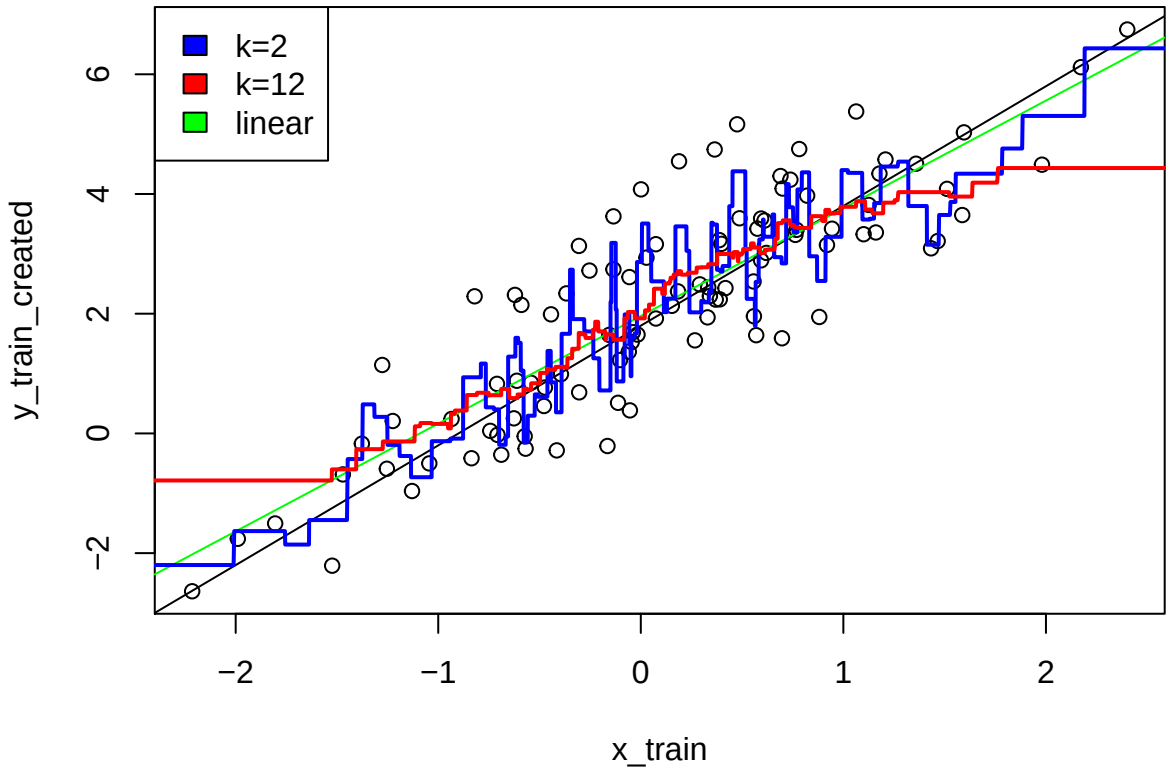
1.2 / 1.3

We plot the true relationship and the best fit line.



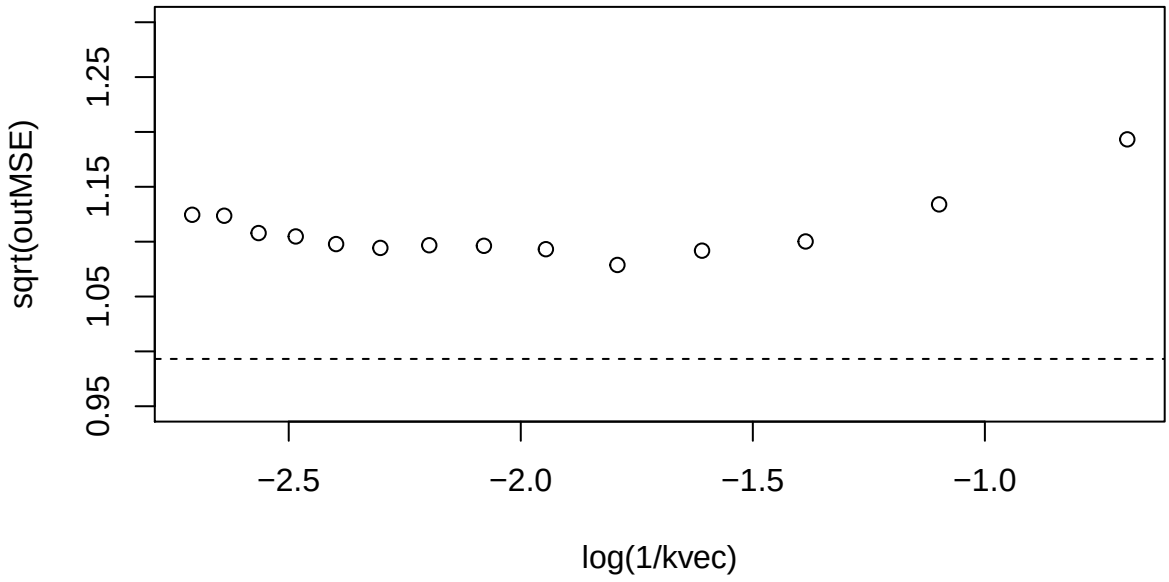
1.4

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



1.5

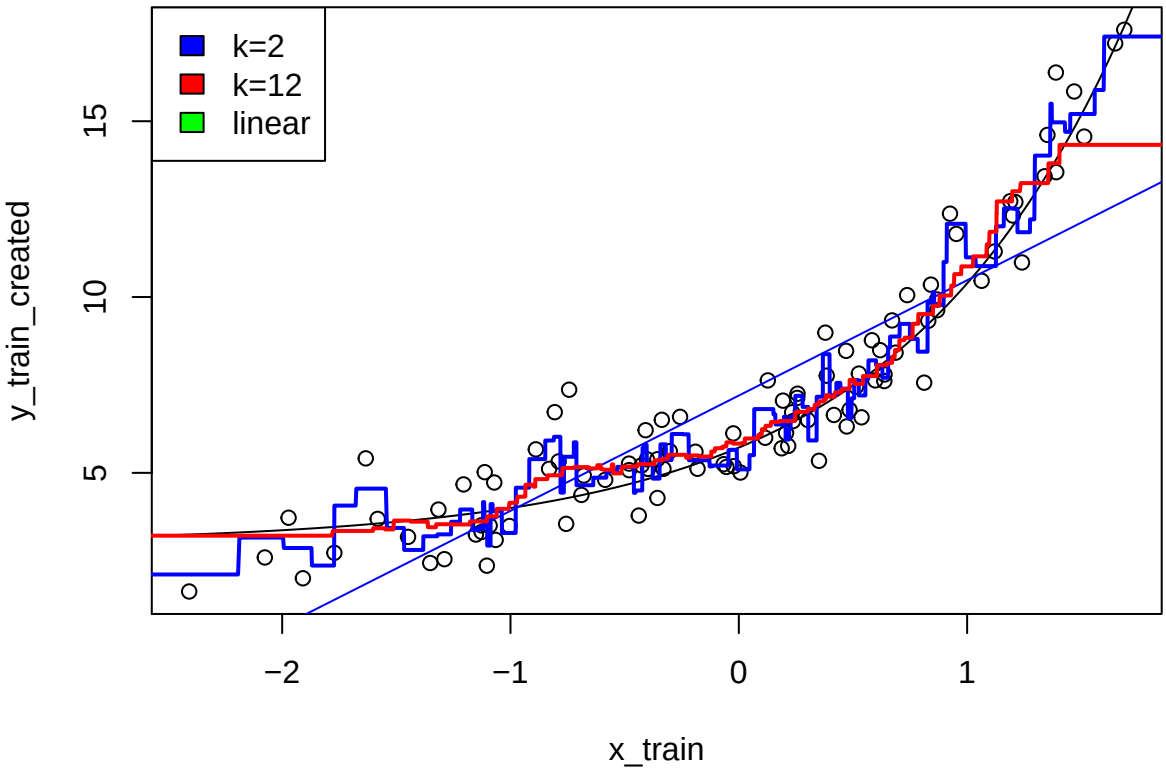
We show the RMSE vs KNN at K = 2:15



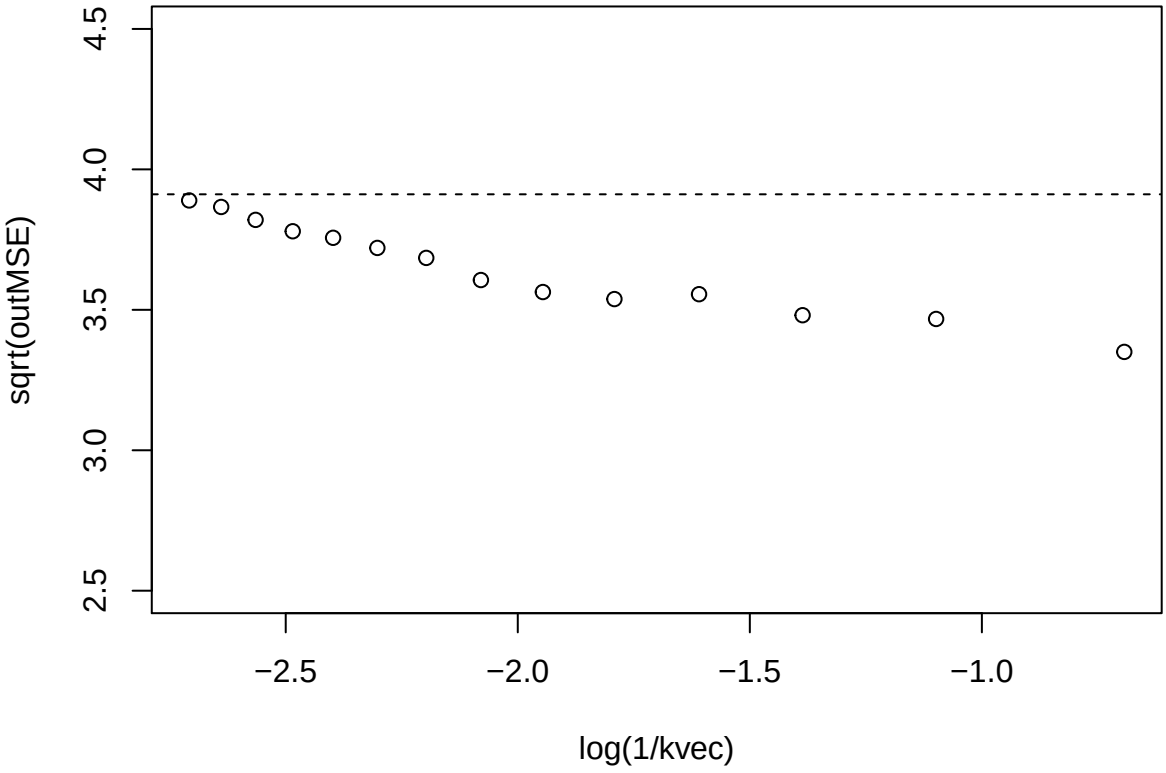
Clearly, at least based on MSE, the linear model performs much better. This makes some intuitive sense as the actual relationship is linear. This is true regardless of the k value used (as seen in the graph). Beyond the actual MSE, just looking at the graphs shows how much closed the linear model appears to the "true" model (the black line).

1.6

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



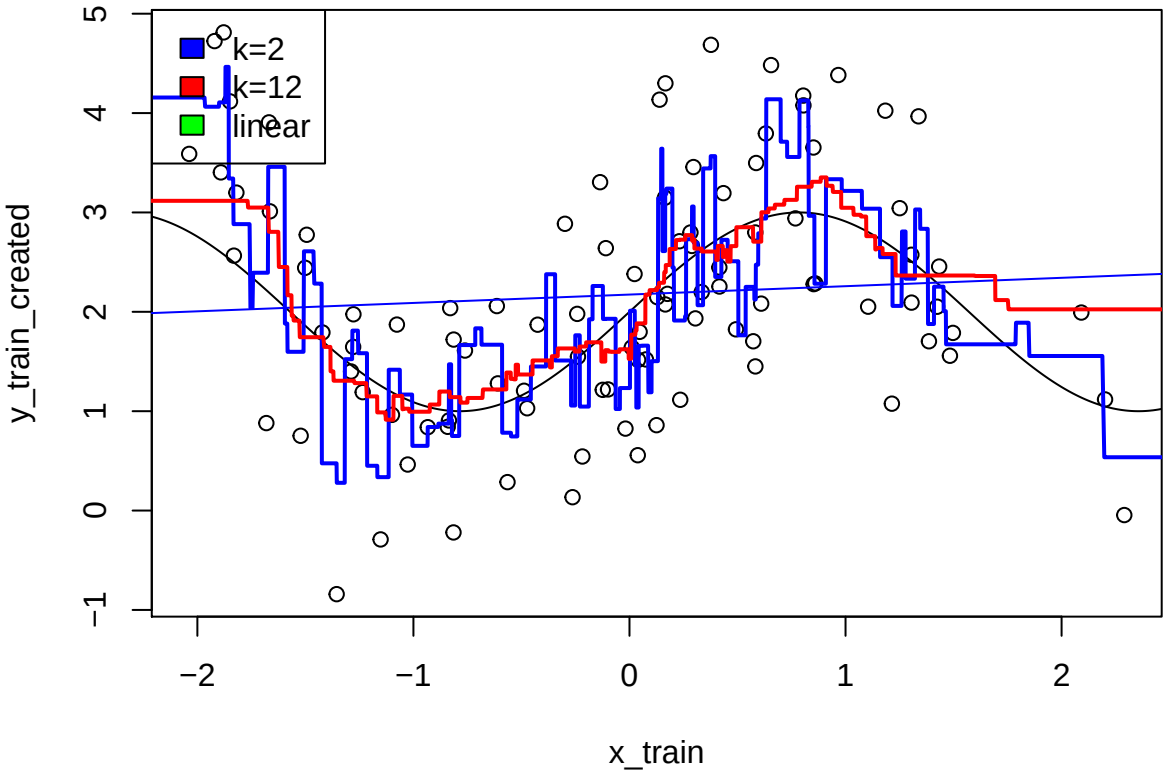
We show the RMSE vs KNN at K = 2:15



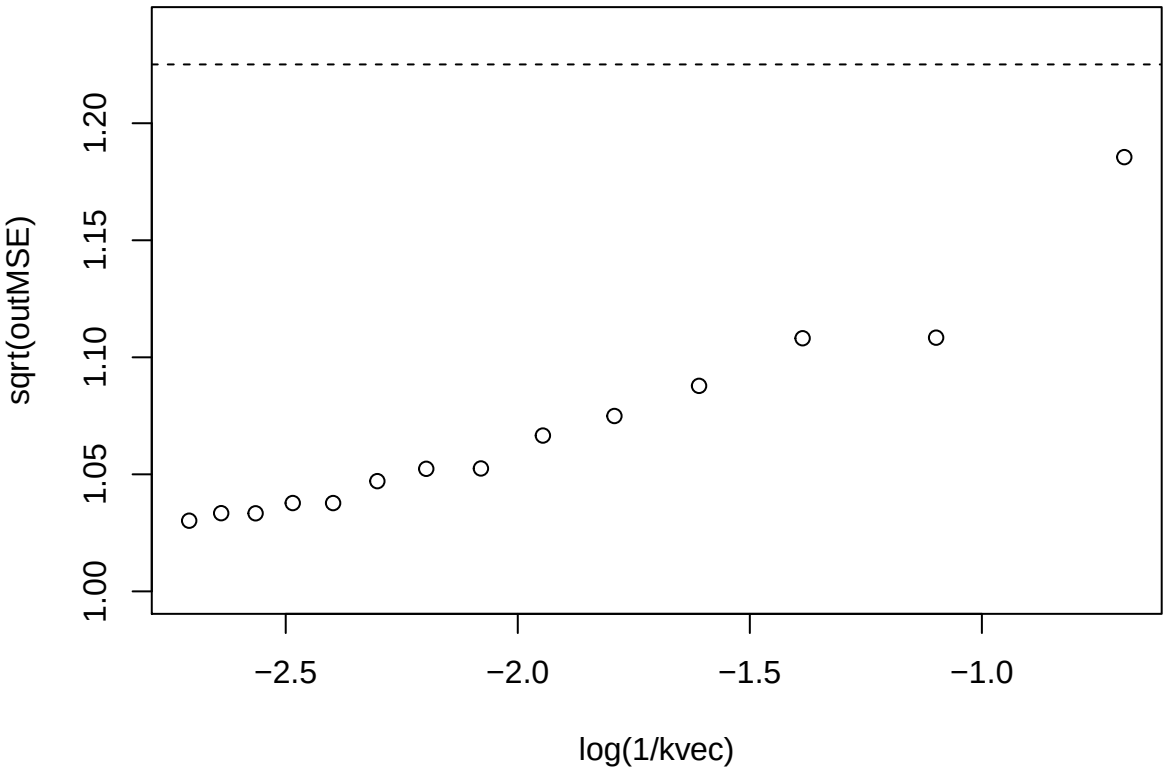
Here, we have the opposite conclusion - the linear model performs worse than even the worst k value knn model. In this case, knn does a significantly better job. Again, this is a somewhat intuitive result as a linear model will never be able to fit a curved relationship like this - something knn would be much more able to do.

1.7

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



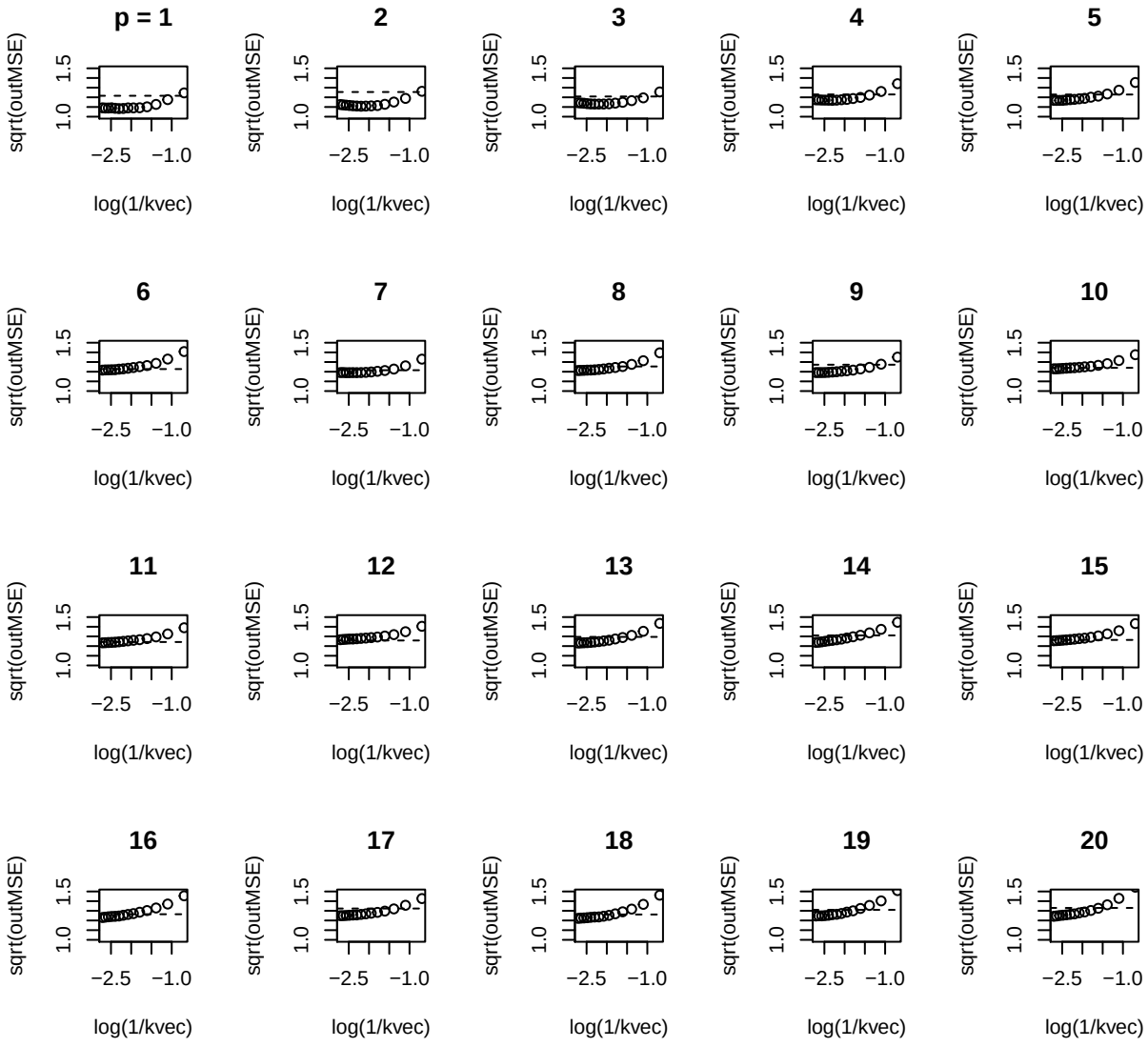
We show the RMSE vs KNN at K = 2:15



In this case, the knn model again significantly outperforms the linear model, even more definitively in the previous case. A linear model is hopeless at even capturing some of the trend in the data, whereas a knn model can create a fairly accurate sine curve.

1.8

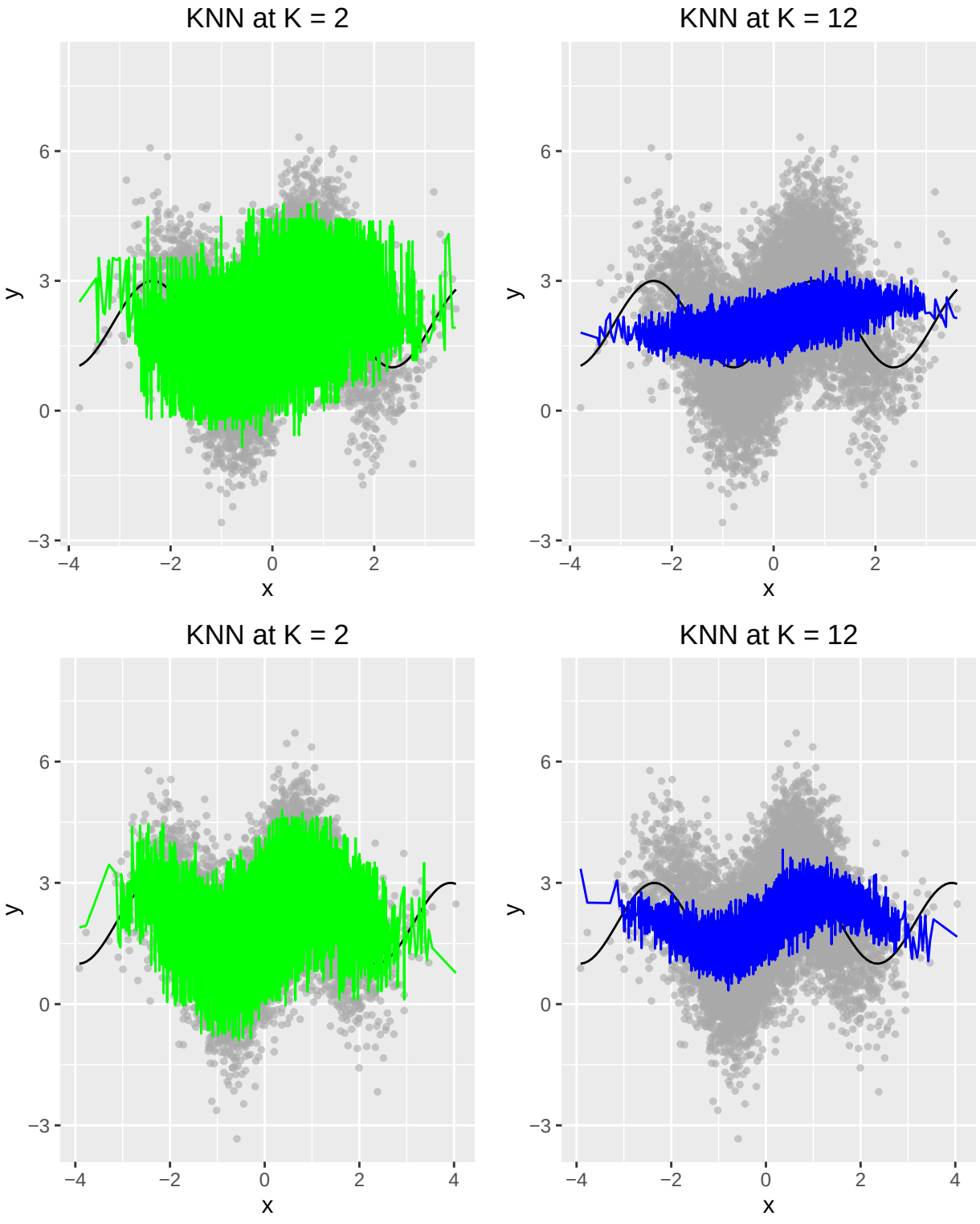
The superfluous features have no predictive power. Thus, when the amount of noise increases in the dataset, the KNN algorithm uses spurious features to predict the value of y. If K is small and the number of superfluous features is large, then there is a high likelihood that the algorithm uses many erroneous covariates to attempt to predict y. As K increases, the algorithm uses more correct features to predict y. That is, as the likelihood of the number of spurious features decreases the MSE increases. 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. Note though that the linear model has similar issues - it too can get tricked by the noise.



As you can see, depending on the specific run, knn frequently outperforms the linear model, but only for certain k values (generally, higher k values). As you can see, as the number of variables increase, the predictions generally get worse (higher MSE). This is due to the additional noise that complicates the model.

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. At the same time, a customer might want to value how much they could sell their car for, and this data would help make the prediction.

Another important aspect of this data would be to value individual features/aspects of a car. If you wanted to estimate the mileage cost on a car, you could look at the relationship between mileage and cost. Similarly, when selling a car, this data could be used to value whether or not adding a sound system might be worth the cost. Heck, you could even use this data to estimate the value of car color.

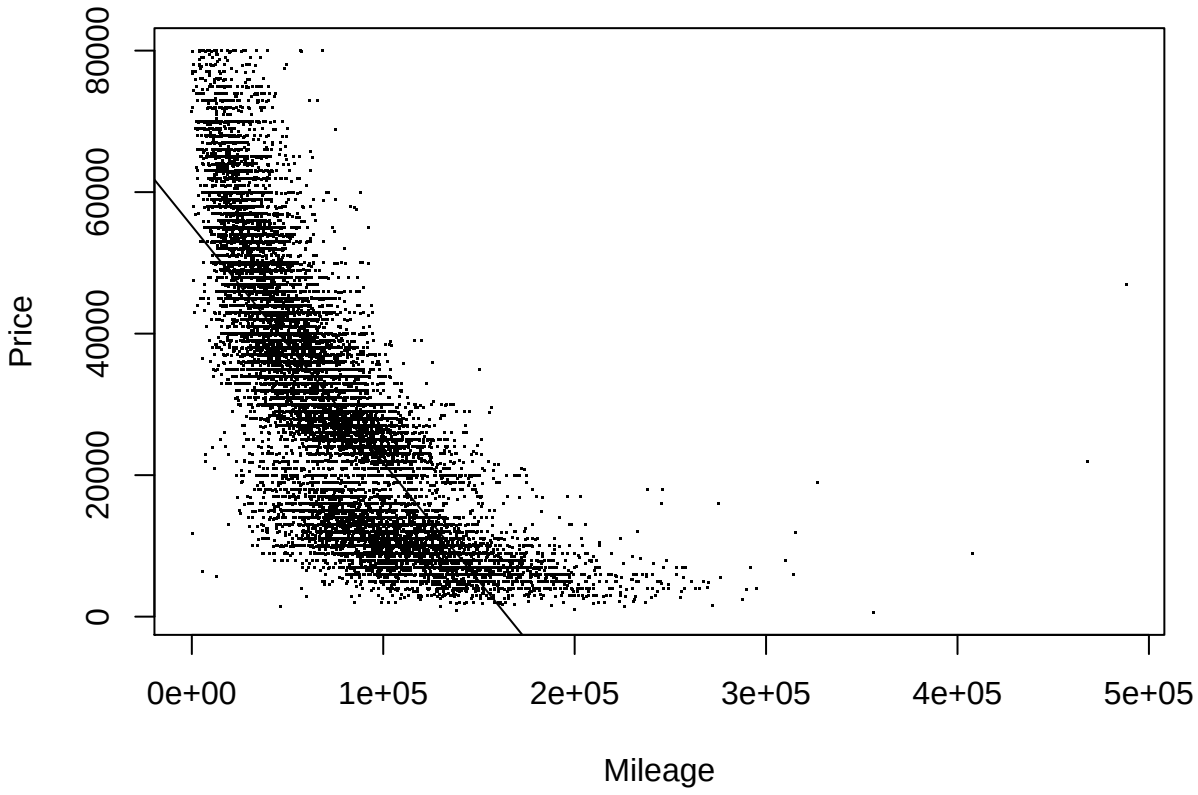
2.2

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,]
```

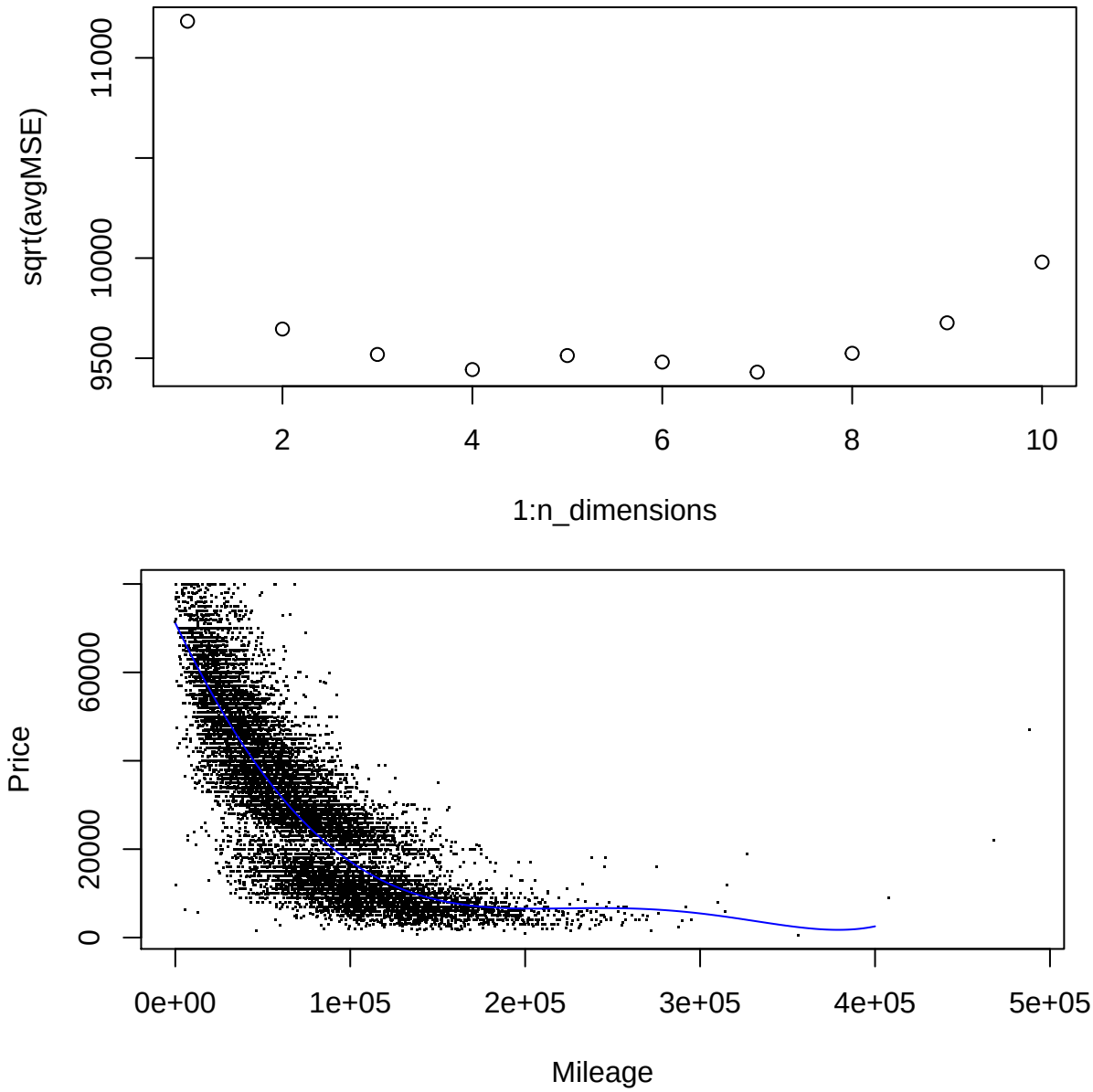
2.3

We plot the best fit linear regression onto the data



2.4

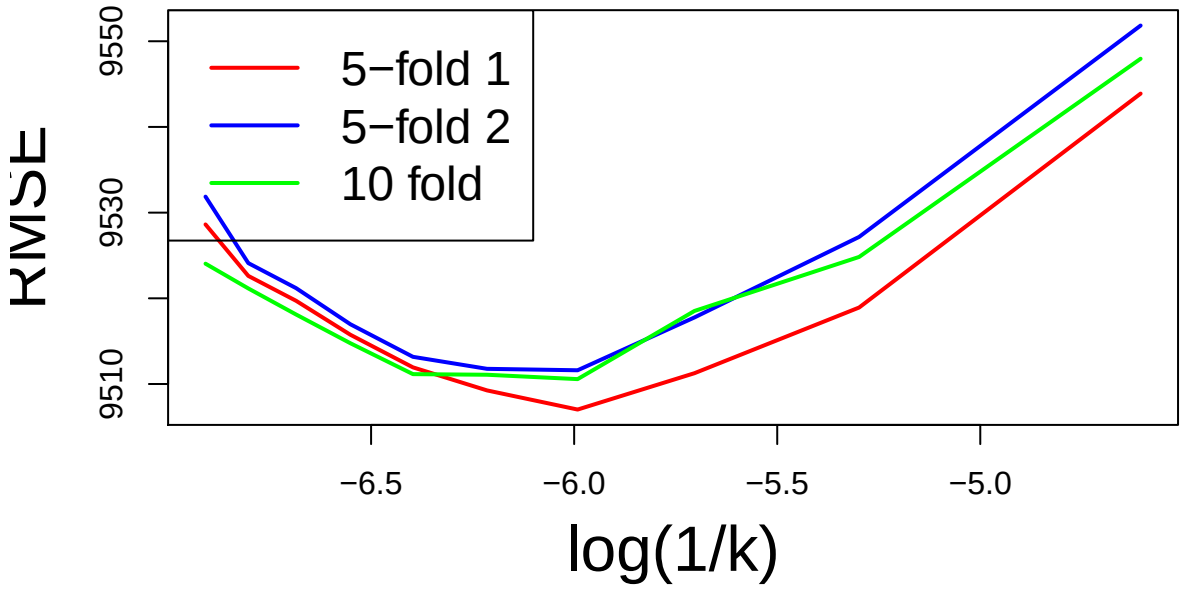
We use cross validation to select the optimal polynomial degree. We find that the optimal polynomial degree is five. Note though how close the MSE is for polynomials 3 to 9; there is not much of a difference between these different degree polynomials and could likely choose any depending on what seed we chose or the number of folds used. We next plot the CV MSE as a function of the degree of the polynomial, and a linear polynomial degree five model.

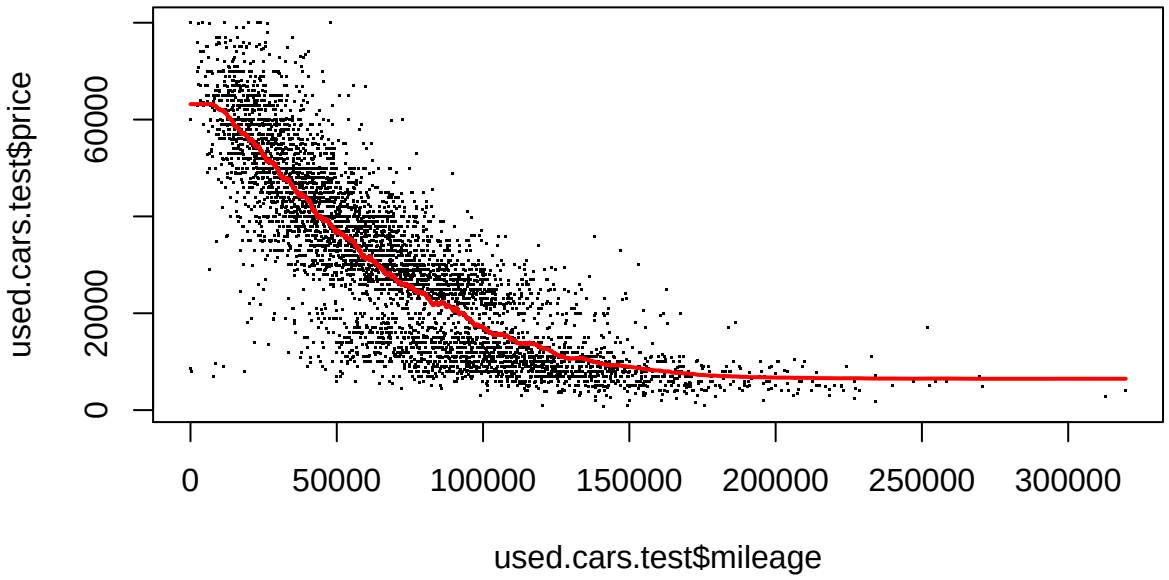


2.5

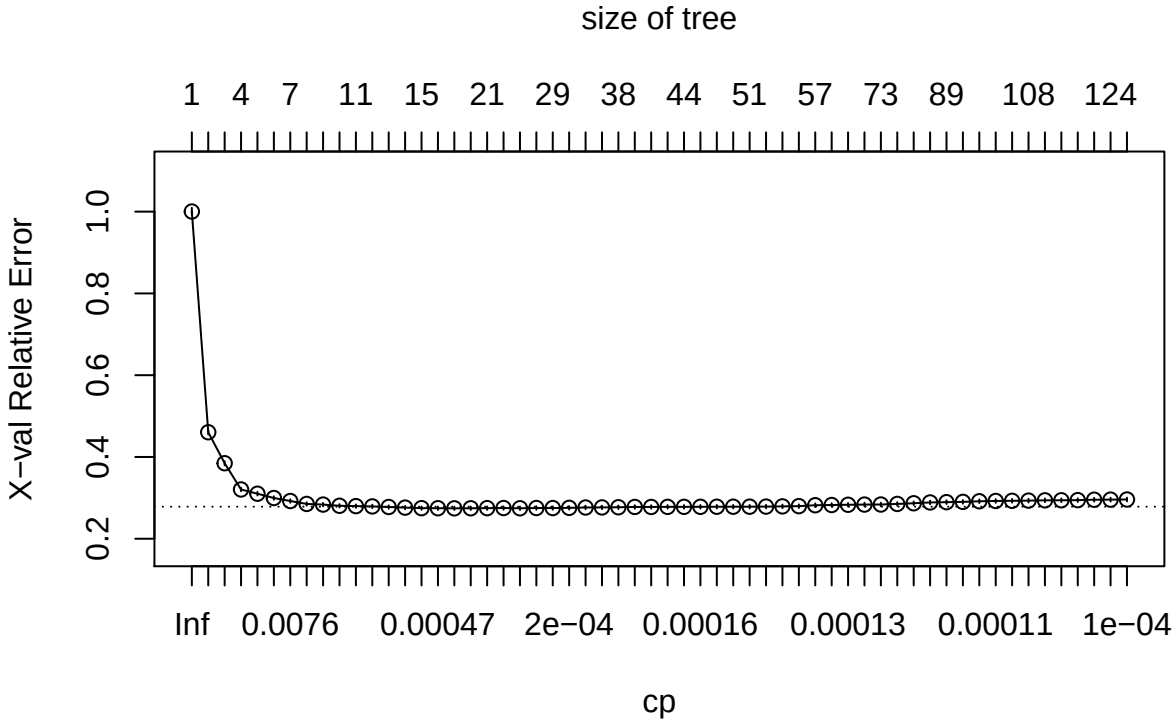
We use cross validation to select the optimal K, and find that MSE is minimized somewhere in the range of K = [400, 600]. We select K = 400 for simplicity. We next plot the CV MSE as a function of the degree of k, and a KNN K = 400 model

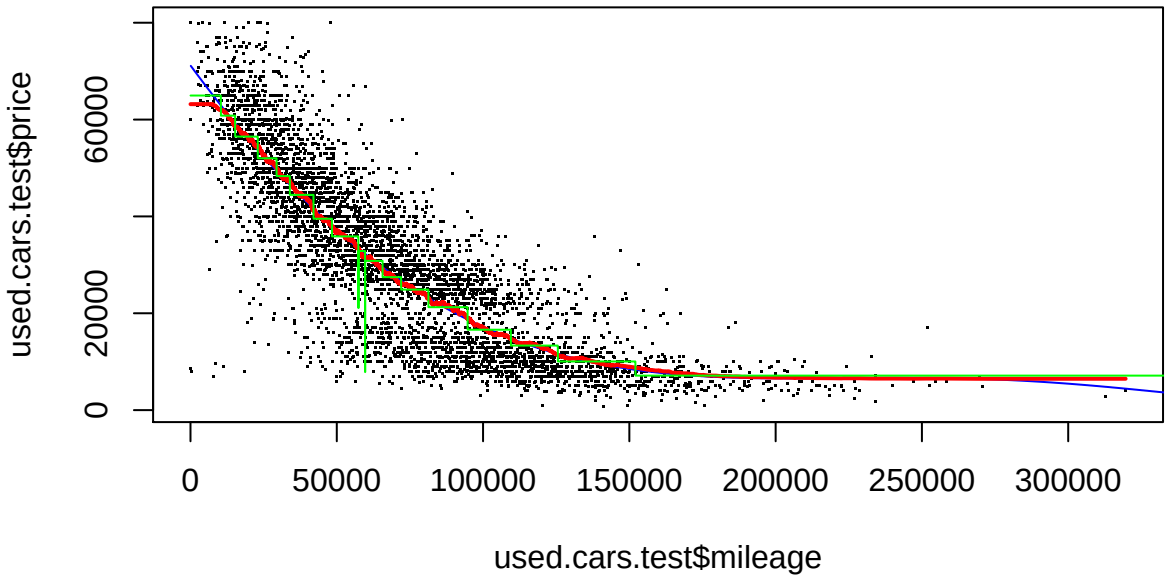
```
## in docv: nset,n,nfold:  10 15048 5
## on fold:  1 , range:  1 : 3010
## on fold:  2 , range:  3011 : 6020
## on fold:  3 , range:  6021 : 9030
## on fold:  4 , range:  9031 : 12040
## on fold:  5 , range:  12041 : 15048

## 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.





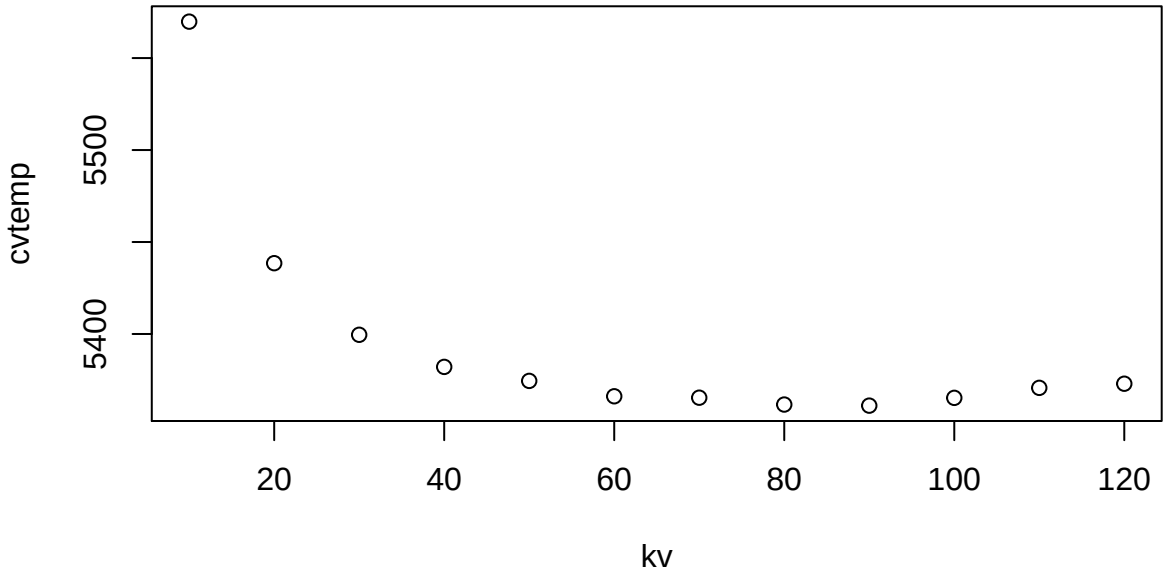
We would use the knn model. They all have nearly identical RMSE in the test sample (by our most recent calculations - KNN was 9448, polynomial was 9459, and the tree was 9560), but we would think that knn would be best able to capture the general trends in the data - this relationship is one that would not be linear and might have some oddities a polynomial function might have trouble accounting for (e.g. consumers might irrationally value a car with <100000 miles more than a car with 100001 on it). That said, it is pretty much a tossup between knn and the tree model. The RMSE of the knn model was 9447.623.

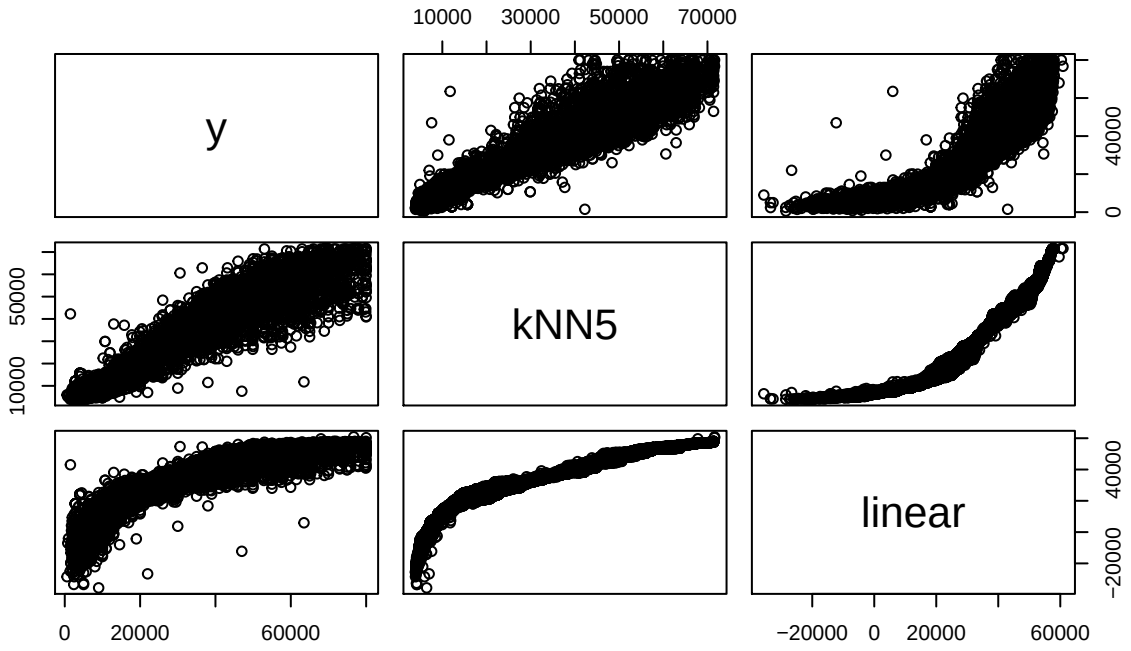
2.6

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. In both cases, our model opts for a "smaller" fit - in the case of knn our k value is much smaller, likely due in large part to the increased amount of data for a given point and better ability to find a similar point. At the same time, the best fit tree is much smaller for similar reasons.

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 7045. For comparison, we show the correlation between the true y and the yhats from the linear, KNN, and tree based model. The size of the tree is 131. Ultimately, both of these models are significantly more accurate with the addition of mileage.

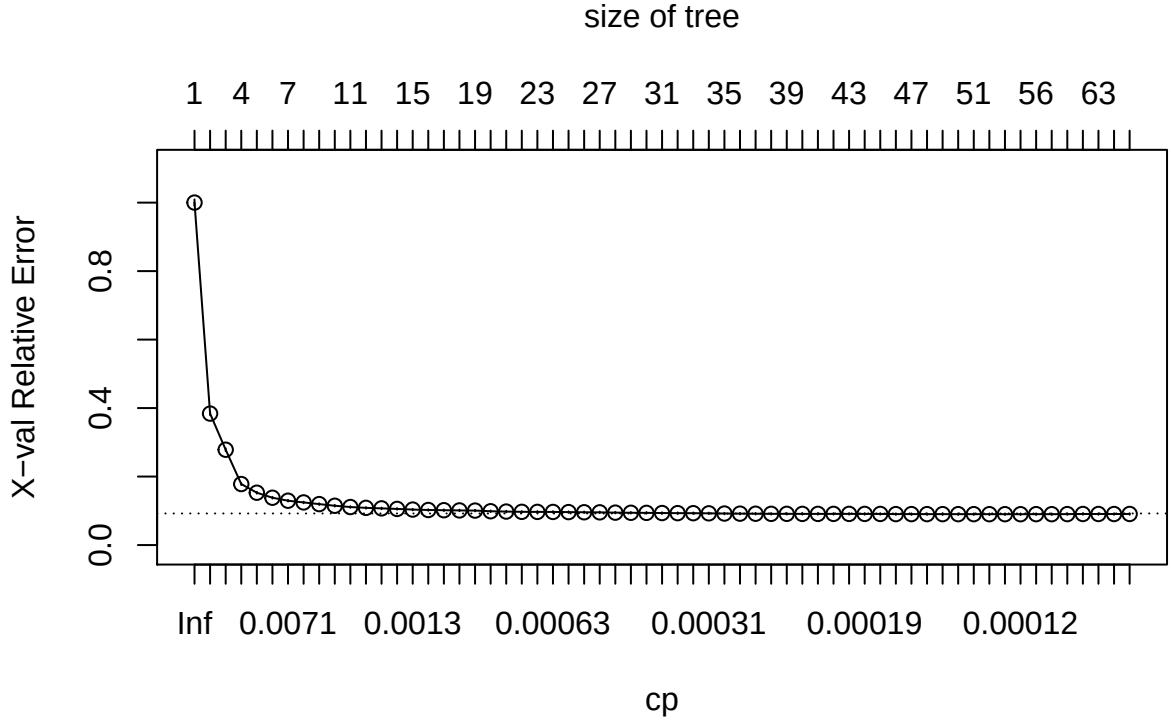
```
## in docv: nset,n,nfold:  12 15048 10
## on fold:  1 , range:  1 : 1505
## on fold:  2 , range:  1506 : 3010
## on fold:  3 , range:  3011 : 4515
## on fold:  4 , range:  4516 : 6020
## on fold:  5 , range:  6021 : 7525
## on fold:  6 , range:  7526 : 9030
## on fold:  7 , range:  9031 : 10535
##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





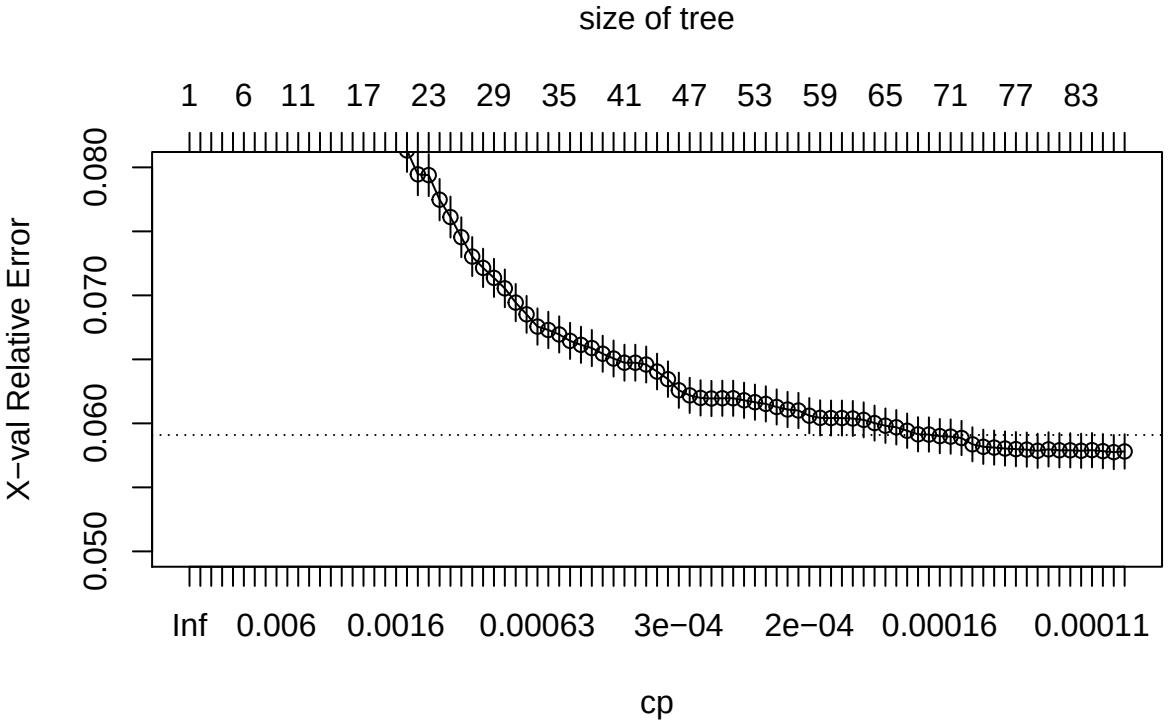

```
##                y      kNN5    linear
## y      1.0000000 0.9569714 0.9109632
## kNN5   0.9569714 1.0000000 0.9522378
## linear 0.9109632 0.9522378 1.0000000
```

```
## [1] 7044.745
```



2.7

We now use all availalbe features in a CART to predict price. We report a MSE of 4466, much better than any of our previous models. Here the added amount of data and variables allowed us to have a significantly more accurate prediction of price.



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.