Homework 2 Complete

```r
library(class)
library(FNN)
```

```
##
## Attaching package: 'FNN'

## The following objects are masked from 'package:class':
##
##     knn, knn.cv
```

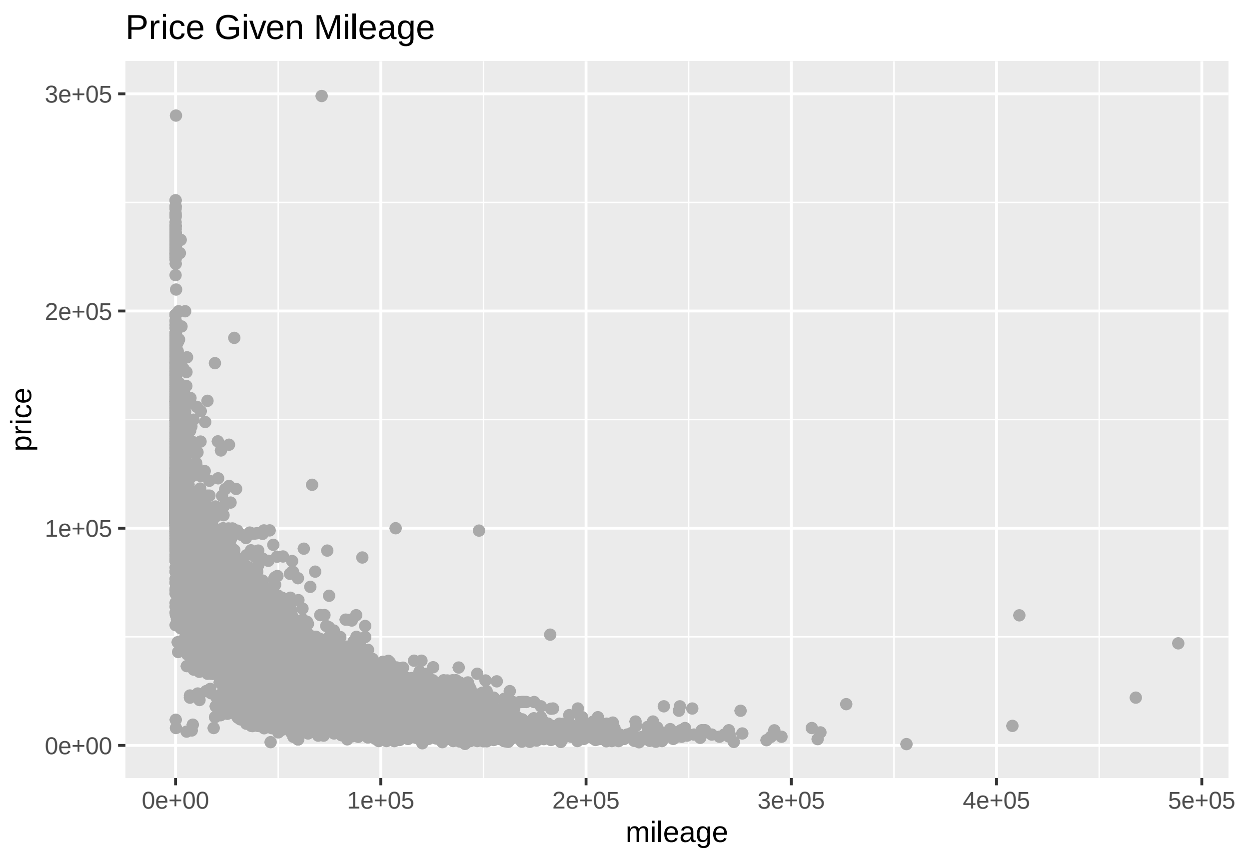
```r
library(tidyverse)
```

```
## -- Attaching packages ------------------------ tidyverse 1.3.0 --

## v ggplot2 3.2.1     v purrr   0.3.3
## v tibble  2.1.3     v dplyr   0.8.3
## v tidyr   1.0.0     v stringr 1.4.0
## v readr   1.3.1     v forcats 0.4.0

## -- Conflicts -------------------------- tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

```r
#Plot original data from sclass data to show relationship between the two variables before separating b
sclass <- read_csv("SDS323/data/sclass.csv")
```

```
## Parsed with column specification:
## cols(
##   id = col_double(),
##   trim = col_character(),
##   subTrim = col_character(),
##   condition = col_character(),
##   isOneOwner = col_logical(),
##   mileage = col_double(),
##   year = col_double(),
##   color = col_character(),
##   displacement = col_character(),
##   fuel = col_character(),
##   state = col_character(),
##   region = col_character(),
##   soundSystem = col_character(),
##   wheelType = col_character(),
##   wheelSize = col_character(),
##   featureCount = col_double(),
##   price = col_double()
## )
```

```r
ggplot(data = sclass) +
  geom_point(mapping = aes(x = mileage, y = price), color='darkgrey') +
  ylim(0, 300000)+
  ggtitle("Price Given Mileage")
```




```r
summary(sclass)
```

```
##        id             trim              subTrim            condition
##  Min.   :    2   Length:29466       Length:29466       Length:29466
##  1st Qu.:13231   Class :character   Class :character   Class :character
##  Median :26254   Mode  :character   Mode  :character   Mode  :character
##  Mean   :26269
##  3rd Qu.:39293
##  Max.   :52572
##  isOneOwner         mileage             year          color
##  Mode :logical   Min.   :     1   Min.   :1988   Length:29466
##  FALSE:25340     1st Qu.:    14   1st Qu.:2007   Class :character
##  TRUE :4126      Median : 26120   Median :2012   Mode  :character
##                  Mean   : 40387   Mean   :2010
##                  3rd Qu.: 68234   3rd Qu.:2015
##                  Max.   :488525   Max.   :2015
##  displacement           fuel              state              region
##  Length:29466       Length:29466       Length:29466       Length:29466
##  Class :character   Class :character   Class :character   Class :character
##  Mode  :character   Mode  :character   Mode  :character   Mode  :character
```

```
##
##
##
##  soundSystem         wheelType          wheelSize          featureCount
##  Length:29466       Length:29466       Length:29466       Min.   :  0.00
##  Class :character   Class :character   Class :character   1st Qu.: 18.00
##  Mode  :character   Mode  :character   Mode  :character   Median : 53.00
##                                                           Mean   : 46.48
##                                                           3rd Qu.: 70.00
##                                                           Max.   :132.00
##      price
##  Min.   :   599
##  1st Qu.: 28995
##  Median : 56991
##  Mean   : 67001
##  3rd Qu.:108815
##  Max.   :299000
```

```
#The three variables we will be focusing on are the trim, mileage, and price. Mileage ranges from 1 to
```

```
#In order to build a predictive model for price given mileage, we will separate the data based on trim.
#Below is the separation of 350 AMG:
sclass350 = subset(sclass, trim == '350')
dim(sclass350)
```

```
## [1] 416  17
```

```
summary(sclass350)
```

```
##       id            trim             subTrim           condition
##  Min.   :  282   Length:416         Length:416         Length:416
##  1st Qu.:14290   Class :character   Class :character   Class :character
##  Median :26658   Mode  :character   Mode  :character   Mode  :character
##  Mean   :26520
##  3rd Qu.:39599
##  Max.   :52220
##  isOneOwner         mileage            year         color
##  Mode :logical   Min.   :     6   Min.   :1994   Length:416
##  FALSE:310       1st Qu.: 19264   1st Qu.:2006   Class :character
##  TRUE :106       Median : 29998   Median :2012   Mode  :character
##                  Mean   : 42926   Mean   :2010
##                  3rd Qu.: 63479   3rd Qu.:2012
##                  Max.   :173000   Max.   :2013
##  displacement           fuel               state               region
##  Length:416         Length:416         Length:416         Length:416
##  Class :character   Class :character   Class :character   Class :character
##  Mode  :character   Mode  :character   Mode  :character   Mode  :character
##
##
##
##  soundSystem         wheelType          wheelSize          featureCount
##  Length:416         Length:416         Length:416         Min.   :  0.00
```

```
##  Class :character   Class :character   Class :character   1st Qu.: 31.75
##  Mode  :character   Mode  :character   Mode  :character   Median : 54.00
##                                                           Mean   : 49.22
##                                                           3rd Qu.: 70.00
##                                                           Max.   :112.00
##      price
##  Min.   :  6600
##  1st Qu.: 19401
##  Median : 52900
##  Mean   : 46854
##  3rd Qu.: 61991
##  Max.   :106010
```

```r
na.omit(sclass350)
```

```
## # A tibble: 416 x 17
##       id trim  subTrim condition isOneOwner mileage  year color displacement
##    <dbl> <chr> <chr>   <chr>     <lgl>        <dbl> <dbl> <chr> <chr>
##  1   282 350   unsp    CPO       FALSE        21929  2012 Black 3.0 L
##  2   284 350   unsp    CPO       FALSE        17770  2012 Silv~ 3.0 L
##  3   285 350   unsp    Used      FALSE        29108  2012 Black 3.0 L
##  4   288 350   unsp    CPO       FALSE        35004  2013 White 3.0 L
##  5   289 350   unsp    Used      TRUE         66689  2012 Black 3.0 L
##  6   290 350   unsp    CPO       FALSE        19567  2012 Black 3.0 L
##  7   949 350   unsp    CPO       TRUE         10616  2012 Black 3.0 L
##  8   950 350   unsp    CPO       FALSE         2578  2013 Black 3.0 L
##  9   951 350   unsp    CPO       FALSE        23677  2012 Black 3.0 L
## 10   952 350   unsp    CPO       TRUE         28384  2012 Black 3.0 L
## # ... with 406 more rows, and 8 more variables: fuel <chr>, state <chr>,
## #   region <chr>, soundSystem <chr>, wheelType <chr>, wheelSize <chr>,
## #   featureCount <dbl>, price <dbl>
```

```r
#Below is the separation of 65 AMG:
sclass65AMG = subset(sclass, trim == '65 AMG')
summary(sclass65AMG)
```

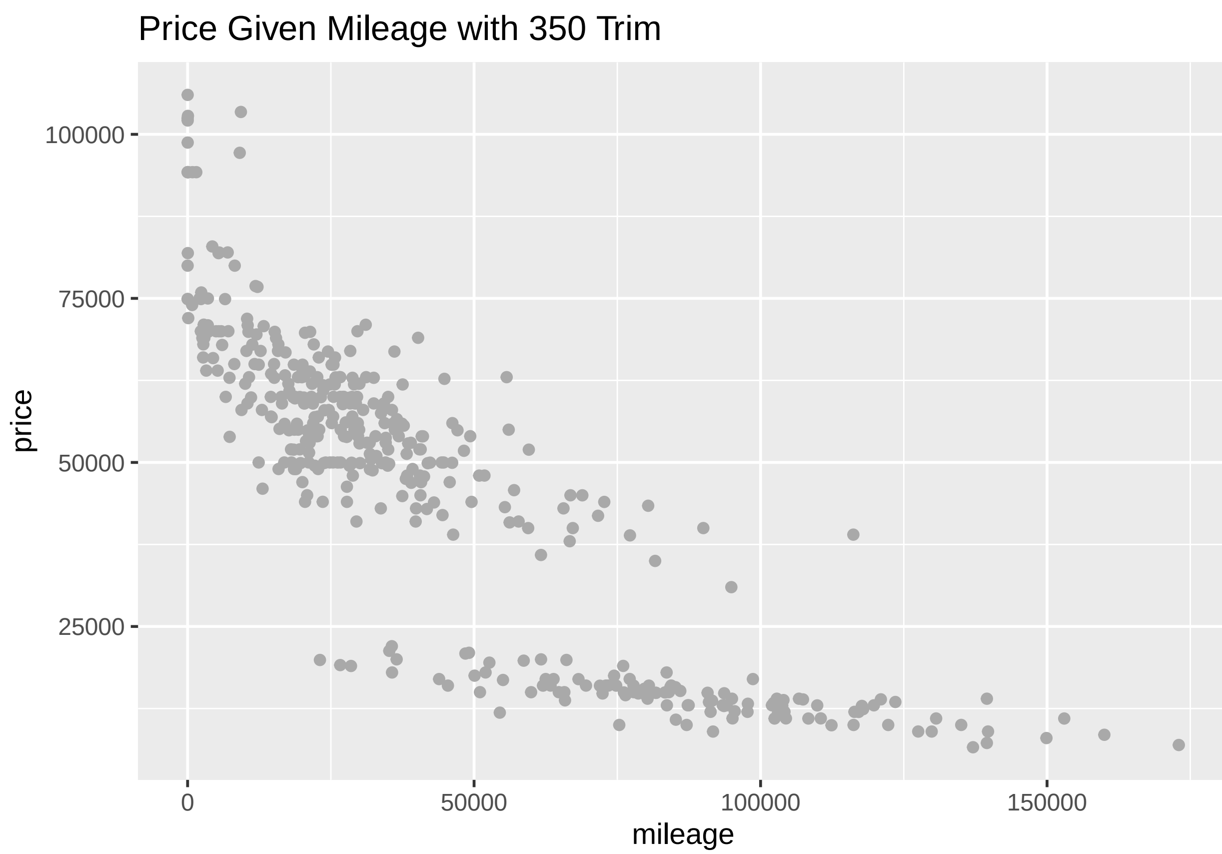
```
##        id            trim             subTrim           condition
##  Min.   : 1060   Length:292         Length:292         Length:292
##  1st Qu.:13977   Class :character   Class :character   Class :character
##  Median :26557   Mode  :character   Mode  :character   Mode  :character
##  Mean   :26444
##  3rd Qu.:38687
##  Max.   :52326
##  isOneOwner         mileage            year          color
##  Mode :logical   Min.   :     1   Min.   :2006   Length:292
##  FALSE:254       1st Qu.:    20   1st Qu.:2007   Class :character
##  TRUE :38        Median : 28803   Median :2010   Mode  :character
##                  Mean   : 33700   Mean   :2010
##                  3rd Qu.: 58496   3rd Qu.:2015
##                  Max.   :146975   Max.   :2015
##  displacement           fuel              state              region
##  Length:292         Length:292         Length:292         Length:292
##  Class :character   Class :character   Class :character   Class :character
```

```
##  Mode  :character   Mode  :character   Mode  :character   Mode  :character
##
##
##
##  soundSystem         wheelType          wheelSize          featureCount
##  Length:292         Length:292         Length:292         Min.   :  0.00
##  Class :character   Class :character   Class :character   1st Qu.: 17.00
##  Mode  :character   Mode  :character   Mode  :character   Median : 58.00
##                                                           Mean   : 48.09
##                                                           3rd Qu.: 72.00
##                                                           Max.   :112.00
##      price
##  Min.   : 18990
##  1st Qu.: 48711
##  Median : 79995
##  Mean   :117121
##  3rd Qu.:225975
##  Max.   :247075
```

```r
dim(sclass65AMG)
```

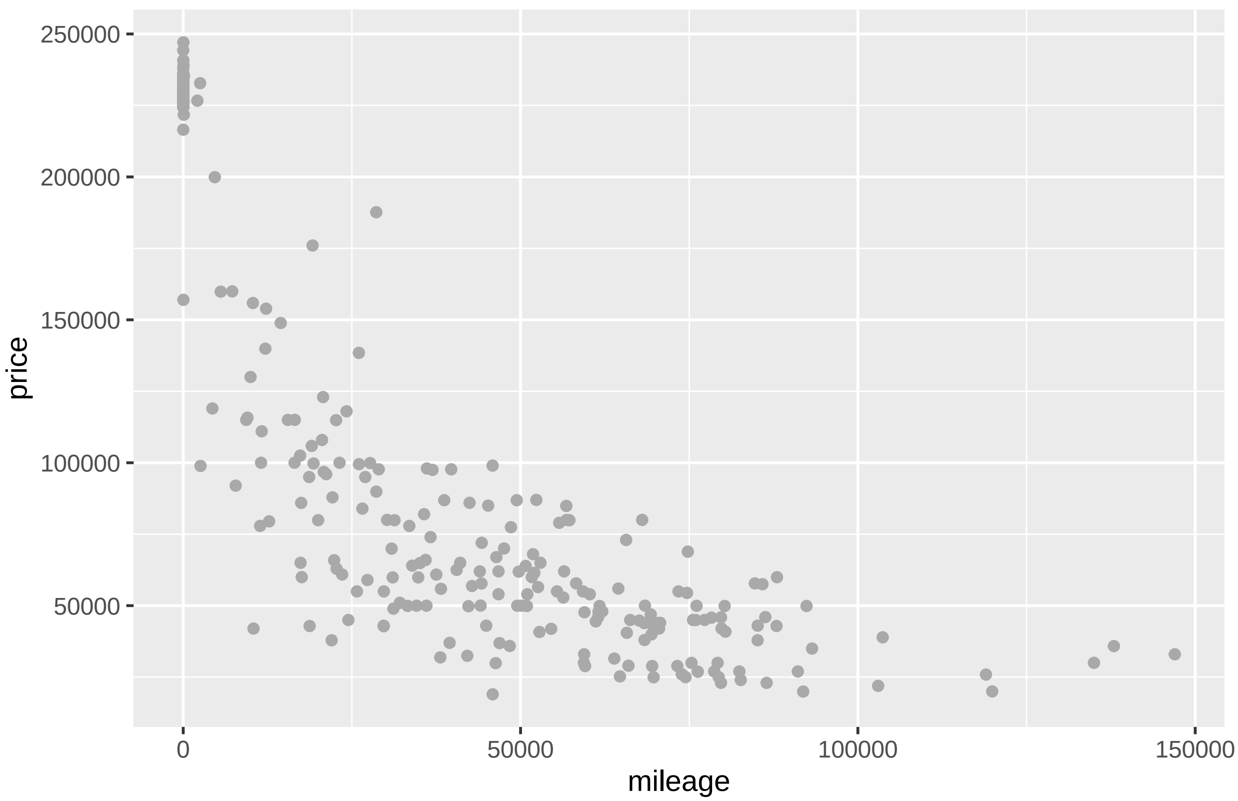
```
## [1] 292  17
```

```r
#Before doing the train-test split, here are two plots that show the data for each trim level. You can
ggplot(data = sclass350) +
  geom_point(mapping = aes(x = mileage, y = price), color='darkgrey')+
  ggtitle("Price Given Mileage with 350 Trim")
```




```r
ggplot(data = sclass65AMG) +
  geom_point(mapping = aes(x = mileage, y = price), color='darkgrey')+
  ggtitle("Price Given Mileage with 65 Trim")
```

Price Given Mileage with 65 Trim




```
#According to these plots, there seems to be more of an even scatter with 65 AMG and more of a break be
```

```
#Train-test split for 350 AMG trim level: We are looking for a model that gives the best explanation wi
N = nrow(sclass350)
N_train = floor(0.8*N)
N_test = N - N_train

train_ind = sort(sample.int(N, N_train, replace=FALSE))

D_train = sclass350[train_ind,]
D_train = arrange(D_train, mileage)
D_test = sclass350[-train_ind,]

y_train = D_train$price
X_train = data.frame(mileage=jitter(D_train$mileage))
X_test = data.frame(mileage=jitter(D_test$mileage))
y_test = D_test$price

#Running KNN test for various values of K to determine the optimal predictive model.This will start at
#KNN = 2
#The first step is to make predictions to the data frame.
lm1 = lm(price ~ mileage, data=D_train)
lm2 = lm(price ~ poly(mileage, 2), data=D_train)
knn2 = knn.reg(train = X_train, test = X_test, y = y_train, k=2)

#rmse calculation
```

```r
rmse = function(y, ypred) {
  sqrt(mean(data.matrix((y-ypred)^2)))
}

ypred_lm1 = predict(lm1, X_test)
ypred_lm2 = predict(lm2, X_test)
ypred_knn2 = knn2$pred

rmse(y_test, ypred_lm1)
```

```
## [1] 10838.68
```

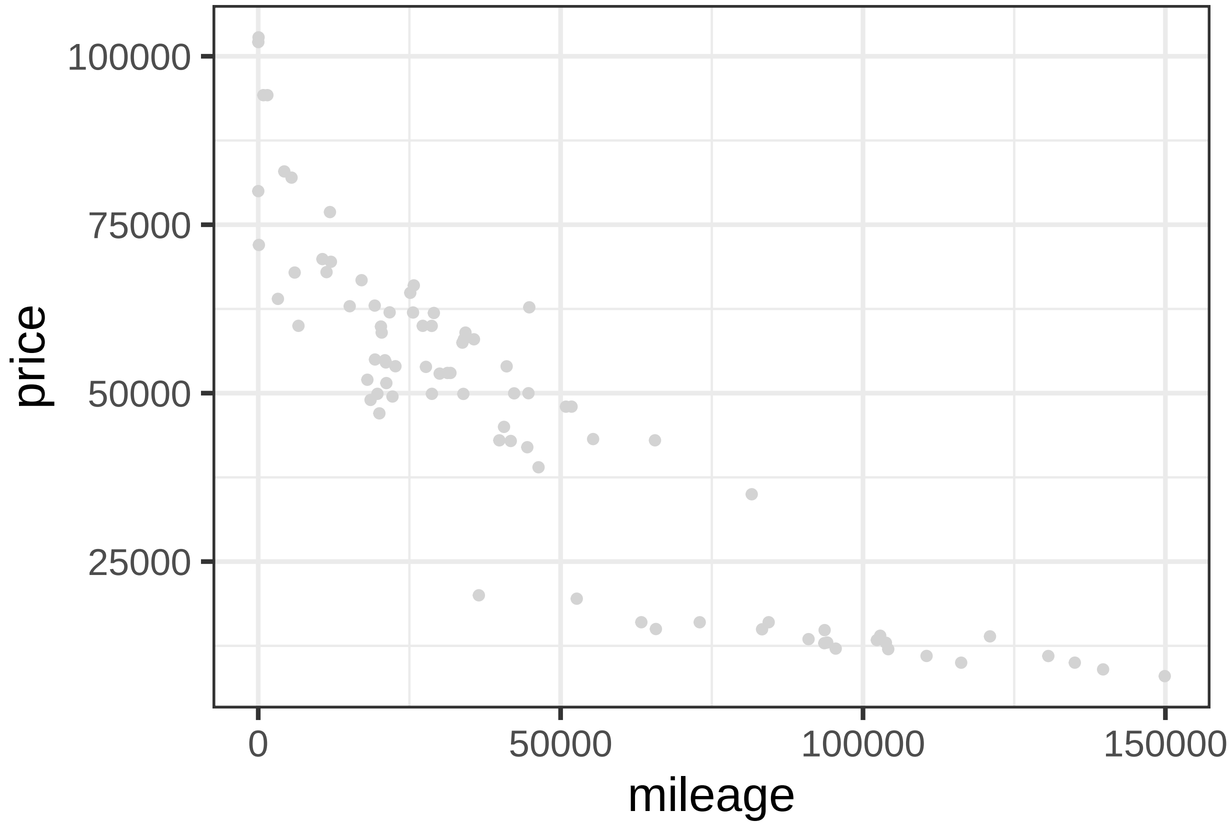
```r
rmse(y_test, ypred_lm2)
```

```
## [1] 9052.279
```

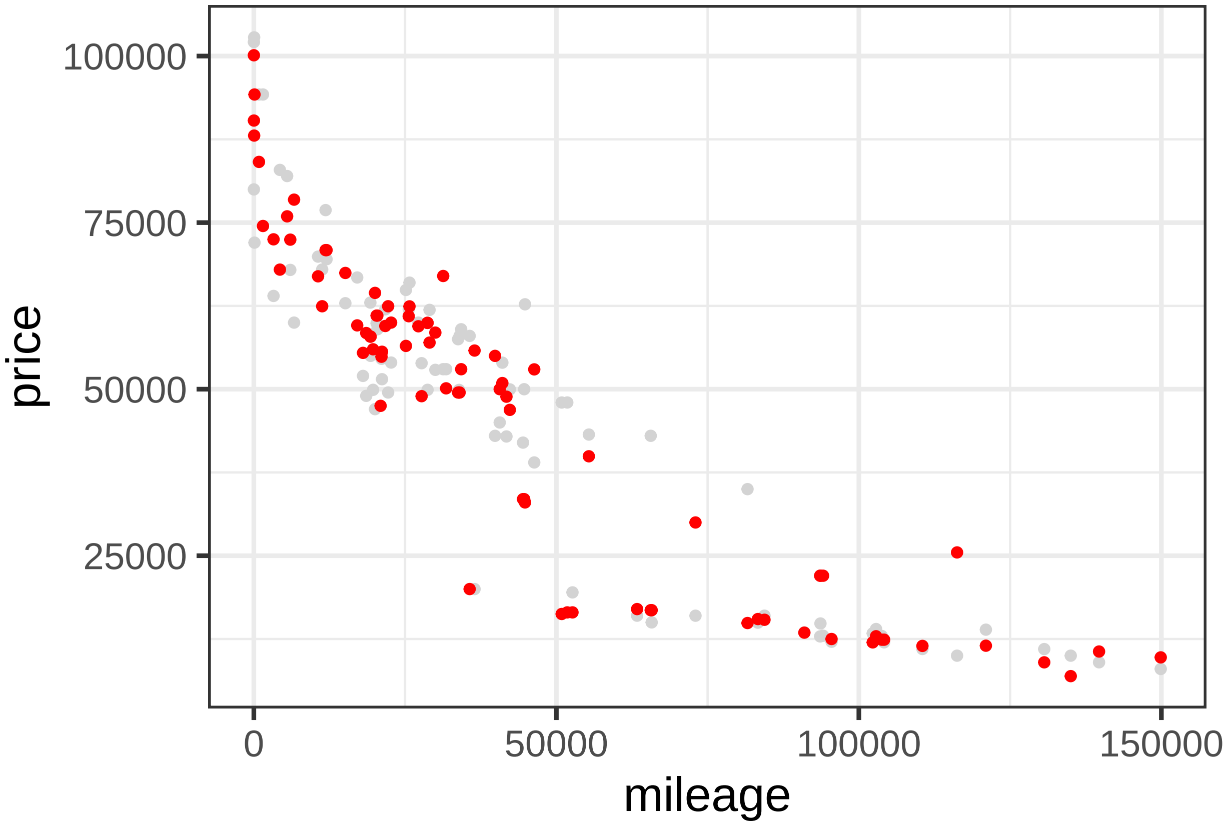
```r
rmse(y_test, ypred_knn2)
```

```
## [1] 12034.11
```

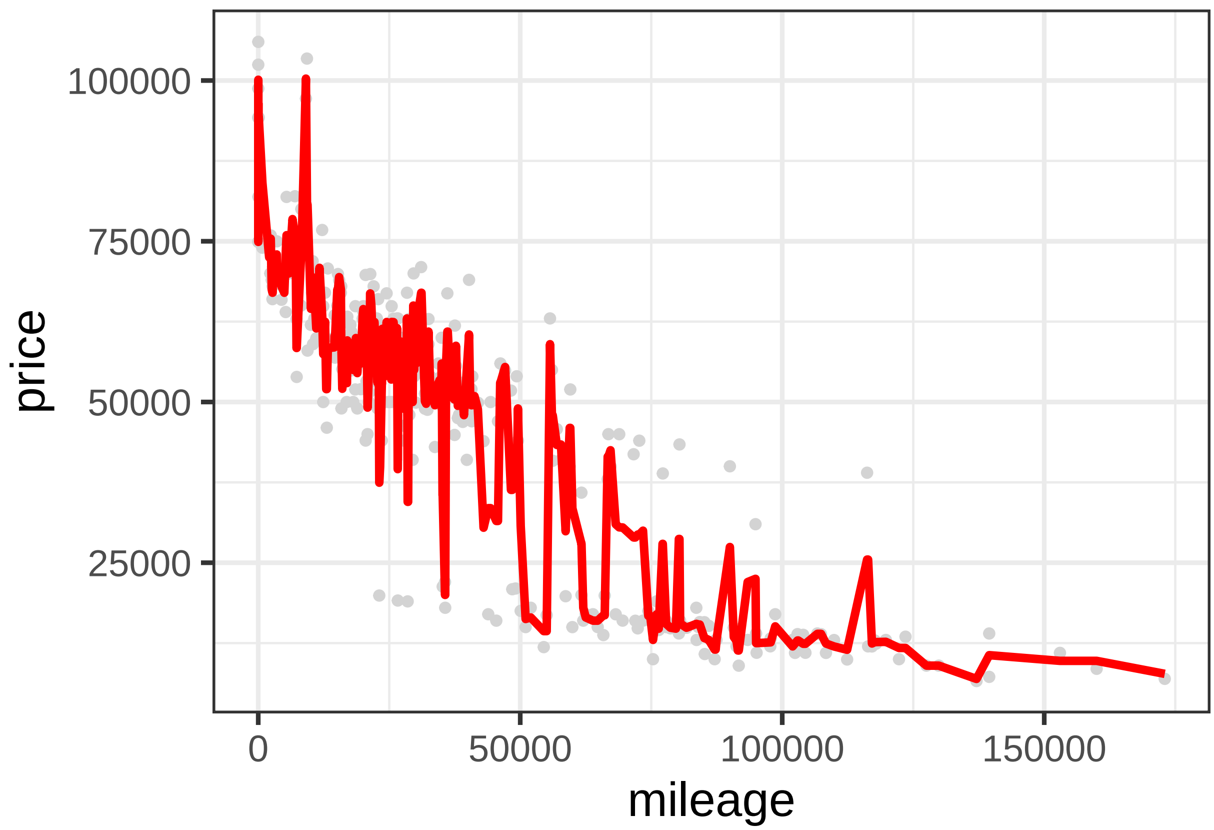
```r
#attach predictions to data frame
D_test$ypred_lm2 = ypred_lm2
D_test$ypred_knn2 = ypred_knn2

p_test = ggplot(data = D_test) +
  geom_point(mapping = aes(x = mileage, y = price), color='lightgrey') +
  theme_bw(base_size=18)
p_test
```




```
p_test + geom_point(aes(x = mileage, y = ypred_knn2), color='red')
```




```
#KNN variances
knn_model = knn.reg(X_train, X_train, y_train, k = 2)

D_train$ypred = knn_model$pred
p_train = ggplot(data = D_train) +
  geom_point(mapping = aes(x = mileage, y = price), color='lightgrey') +
  theme_bw(base_size=18)
p_train + geom_path(mapping = aes(x=mileage, y=ypred), color='red', size=1.5)
```




```r
#K = 3
lm1 = lm(price ~ mileage, data=D_train)
lm2 = lm(price ~ poly(mileage, 2), data=D_train)
knn3 = knn.reg(train = X_train, test = X_test, y = y_train, k=3)


#rmse
rmse = function(y, ypred) {
  sqrt(mean(data.matrix((y-ypred)^2)))
}

ypred_lm1 = predict(lm1, X_test)
ypred_lm2 = predict(lm2, X_test)
ypred_knn3 = knn3$pred

rmse(y_test, ypred_lm1)
```

```
## [1] 10838.68
```

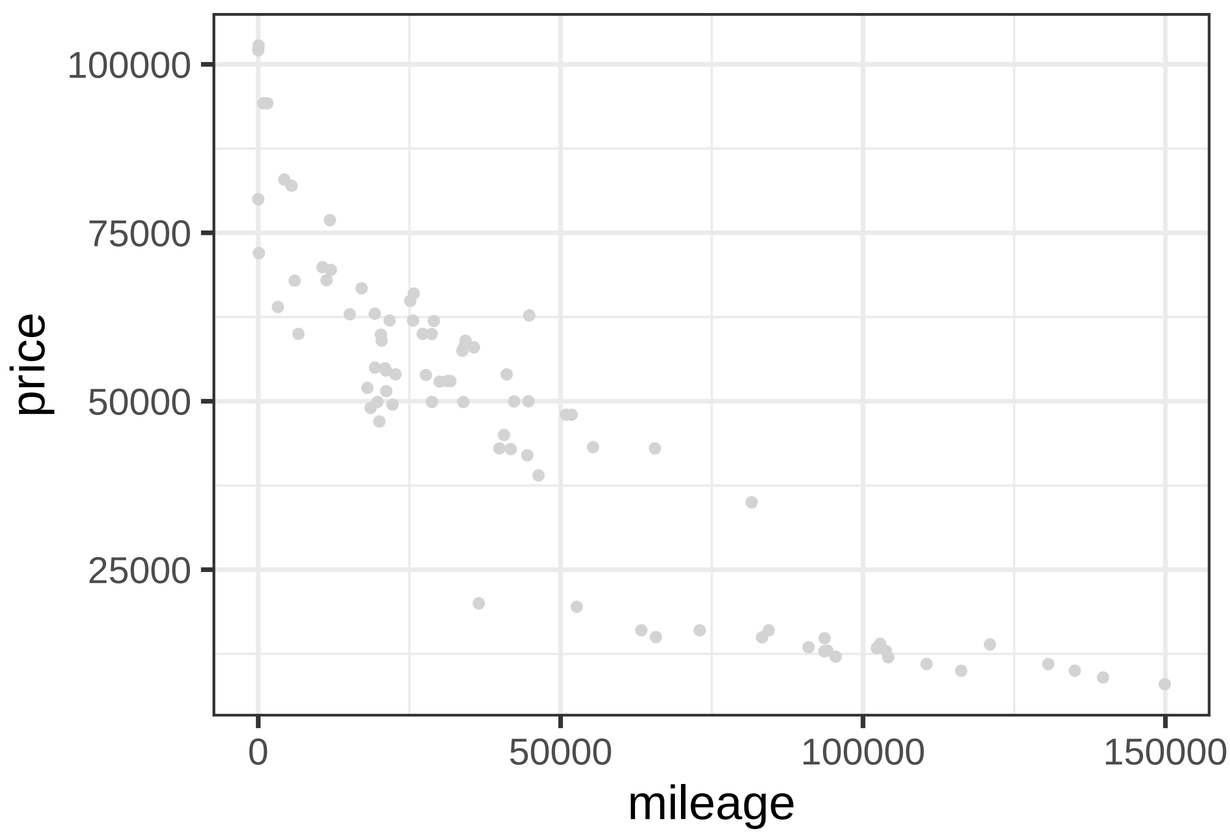
```r
rmse(y_test, ypred_lm2)
```

```
## [1] 9052.279
```

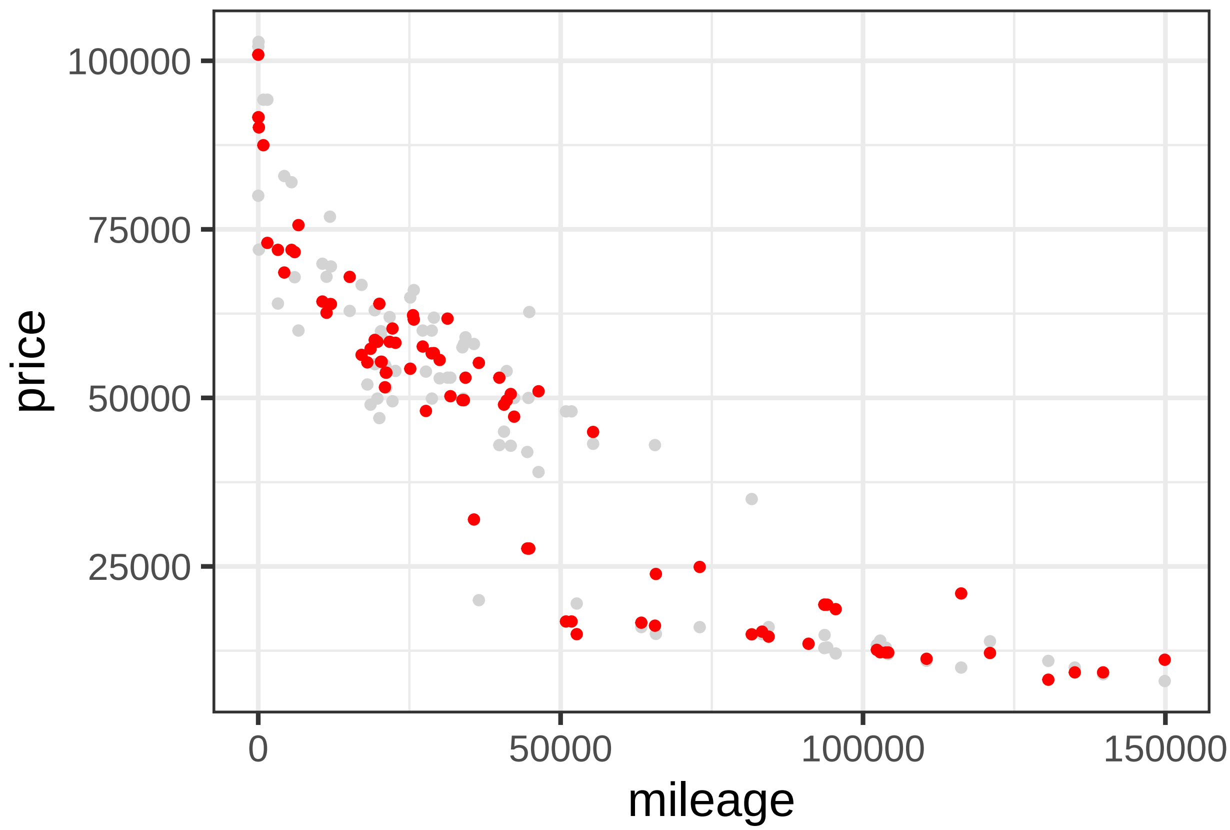
```r
rmse(y_test, ypred_knn3)
```

```
## [1] 11666.98
```

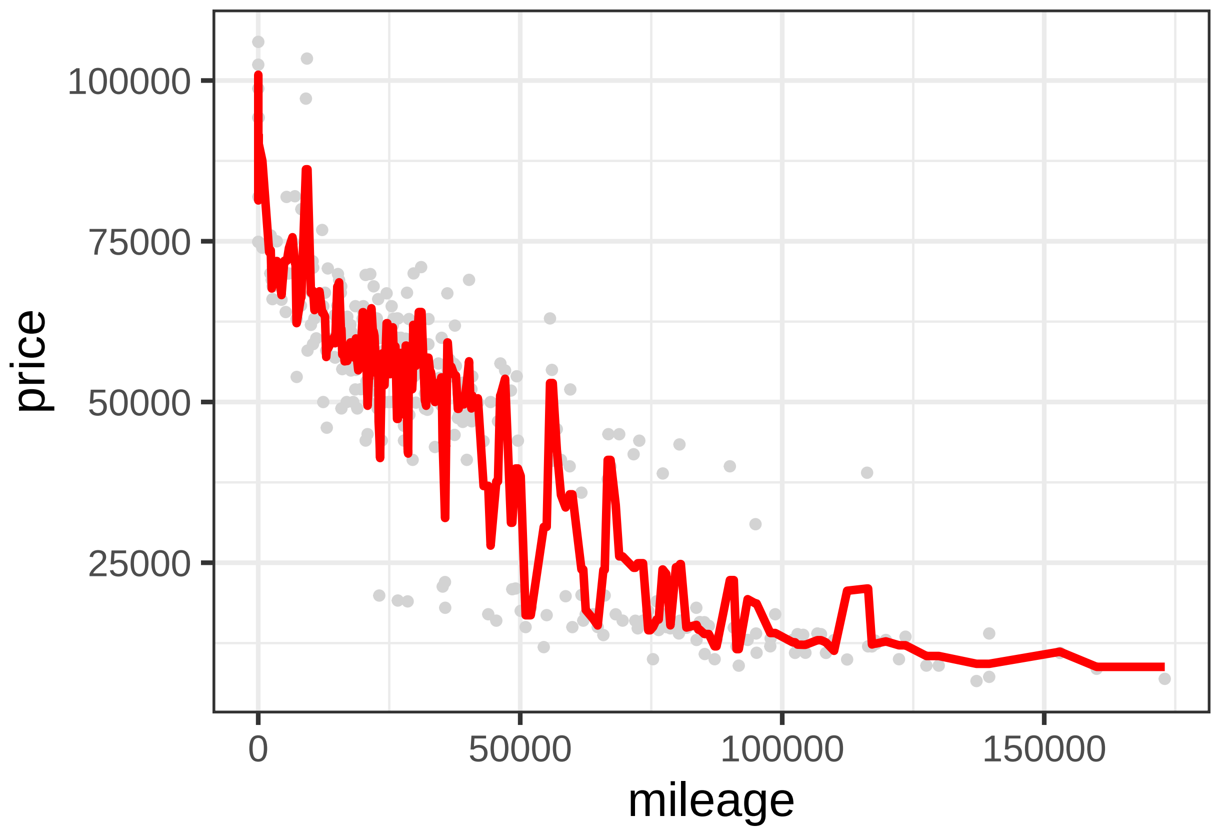
```r
#attach predictions to data frame
D_test$ypred_lm2 = ypred_lm2
D_test$ypred_knn3 = ypred_knn3

p_test = ggplot(data = D_test) +
  geom_point(mapping = aes(x = mileage, y = price), color='lightgrey') +
  theme_bw(base_size=18)
p_test
```




```r
p_test + geom_point(aes(x = mileage, y = ypred_knn3), color='red')
```




```
#KNN variances
knn_model = knn.reg(X_train, X_train, y_train, k = 3)

D_train$ypred = knn_model$pred
p_train = ggplot(data = D_train) +
  geom_point(mapping = aes(x = mileage, y = price), color='lightgrey') +
  theme_bw(base_size=18)
p_train + geom_path(mapping = aes(x=mileage, y=ypred), color='red', size=1.5)
```




```r
#K = 4
lm1 = lm(price ~ mileage, data=D_train)
lm2 = lm(price ~ poly(mileage, 2), data=D_train)
knn4 = knn.reg(train = X_train, test = X_test, y = y_train, k=4)


#rmse
rmse = function(y, ypred) {
  sqrt(mean(data.matrix((y-ypred)^2)))
}

ypred_lm1 = predict(lm1, X_test)
ypred_lm2 = predict(lm2, X_test)
ypred_knn4 = knn4$pred

rmse(y_test, ypred_lm1)
```

```
## [1] 10838.68
```

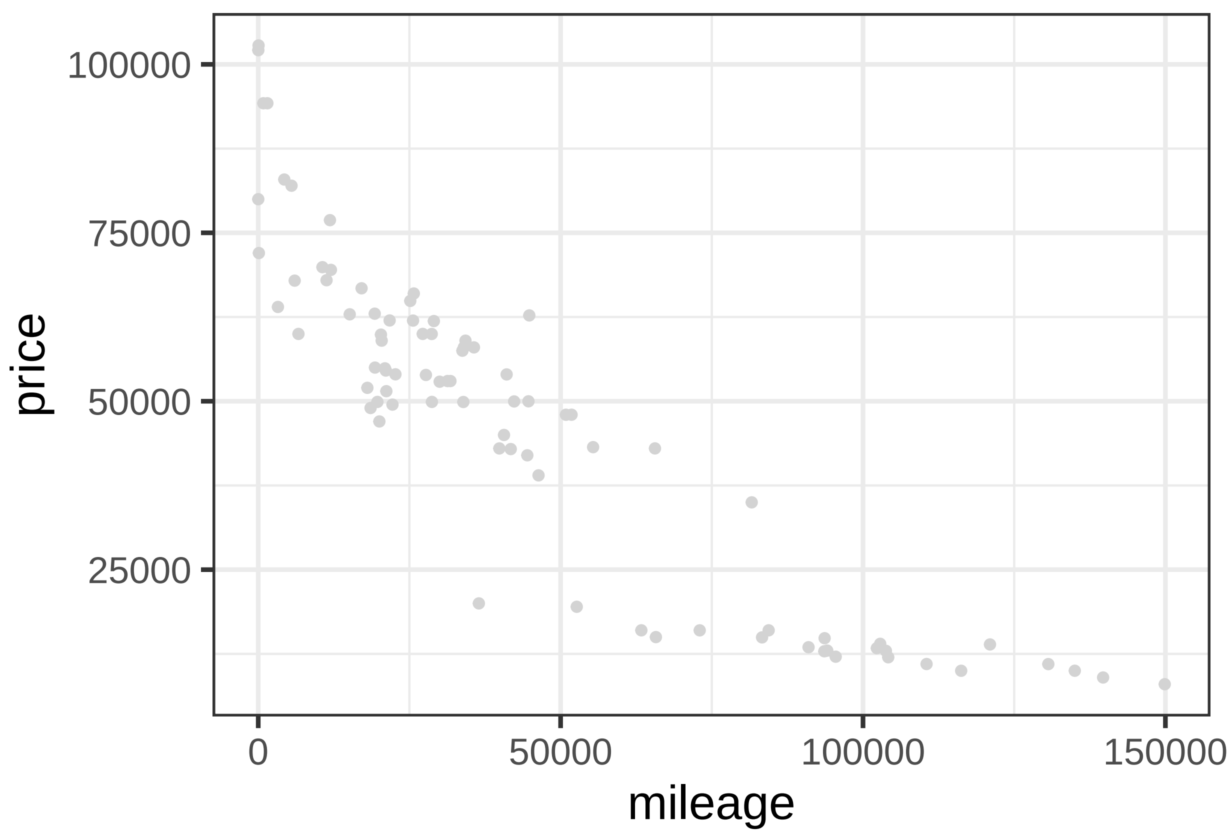
```r
rmse(y_test, ypred_lm2)
```

```
## [1] 9052.279
```

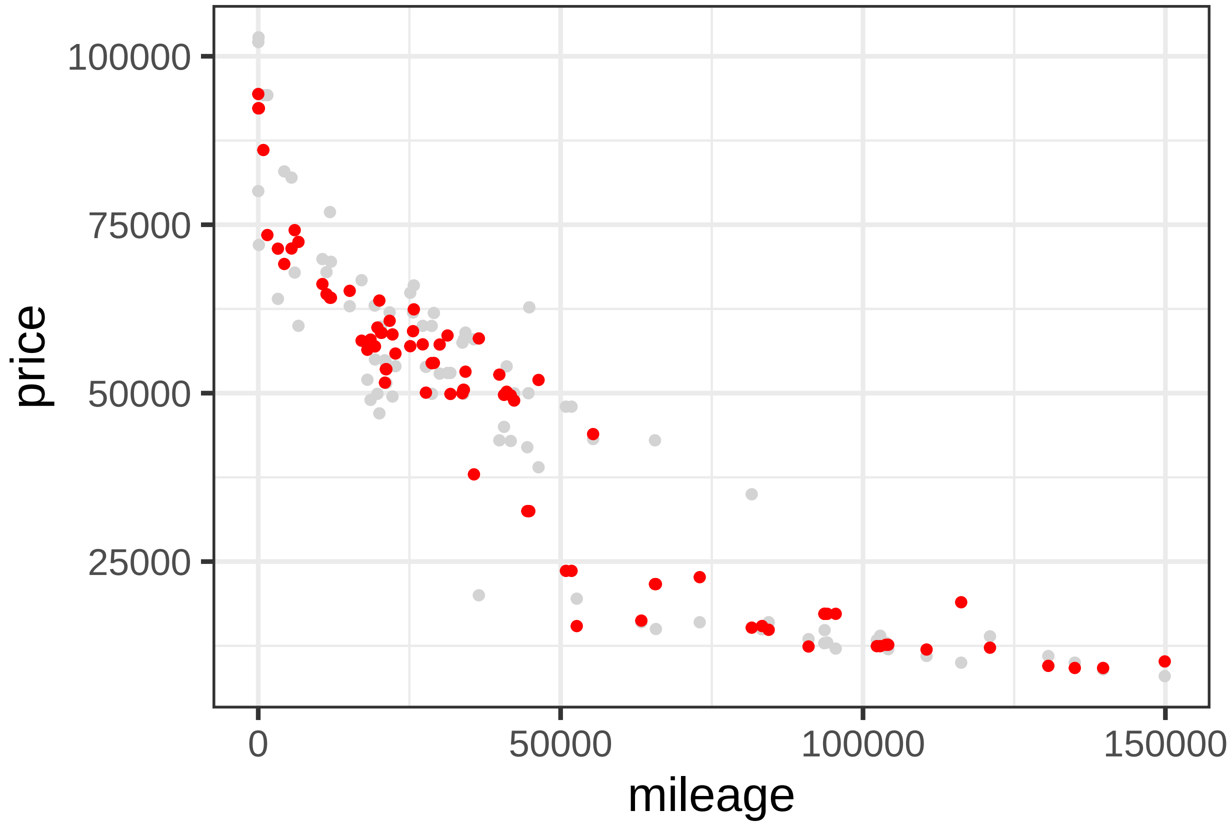
```r
rmse(y_test, ypred_knn4)
```

```
## [1] 10374.04
```

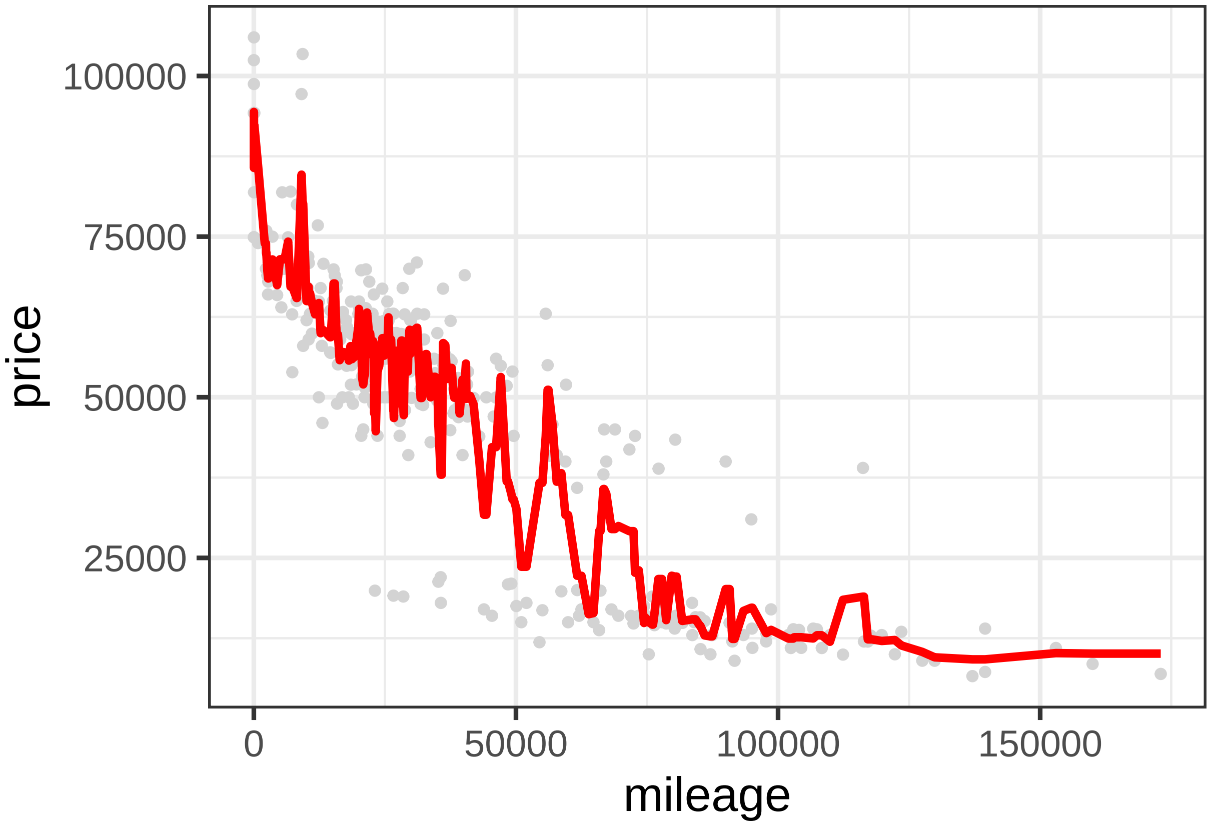
```r
#attach predictions to data frame
D_test$ypred_lm2 = ypred_lm2
D_test$ypred_knn4 = ypred_knn4

p_test = ggplot(data = D_test) +
  geom_point(mapping = aes(x = mileage, y = price), color='lightgrey') +
  theme_bw(base_size=18)
p_test
```




```r
p_test + geom_point(aes(x = mileage, y = ypred_knn4), color='red')
```




```r
#KNN variances
knn_model = knn.reg(X_train, X_train, y_train, k = 4)

D_train$ypred = knn_model$pred
p_train = ggplot(data = D_train) +
  geom_point(mapping = aes(x = mileage, y = price), color='lightgrey') +
  theme_bw(base_size=18)
p_train + geom_path(mapping = aes(x=mileage, y=ypred), color='red', size=1.5)
```




```
#K = 5
lm1 = lm(price ~ mileage, data=D_train)
lm2 = lm(price ~ poly(mileage, 2), data=D_train)
knn5 = knn.reg(train = X_train, test = X_test, y = y_train, k=5)


#rmse
rmse = function(y, ypred) {
  sqrt(mean(data.matrix((y-ypred)^2)))
}

ypred_lm1 = predict(lm1, X_test)
ypred_lm2 = predict(lm2, X_test)
ypred_knn5 = knn5$pred

rmse(y_test, ypred_lm1)
```

```
## [1] 10838.68
```

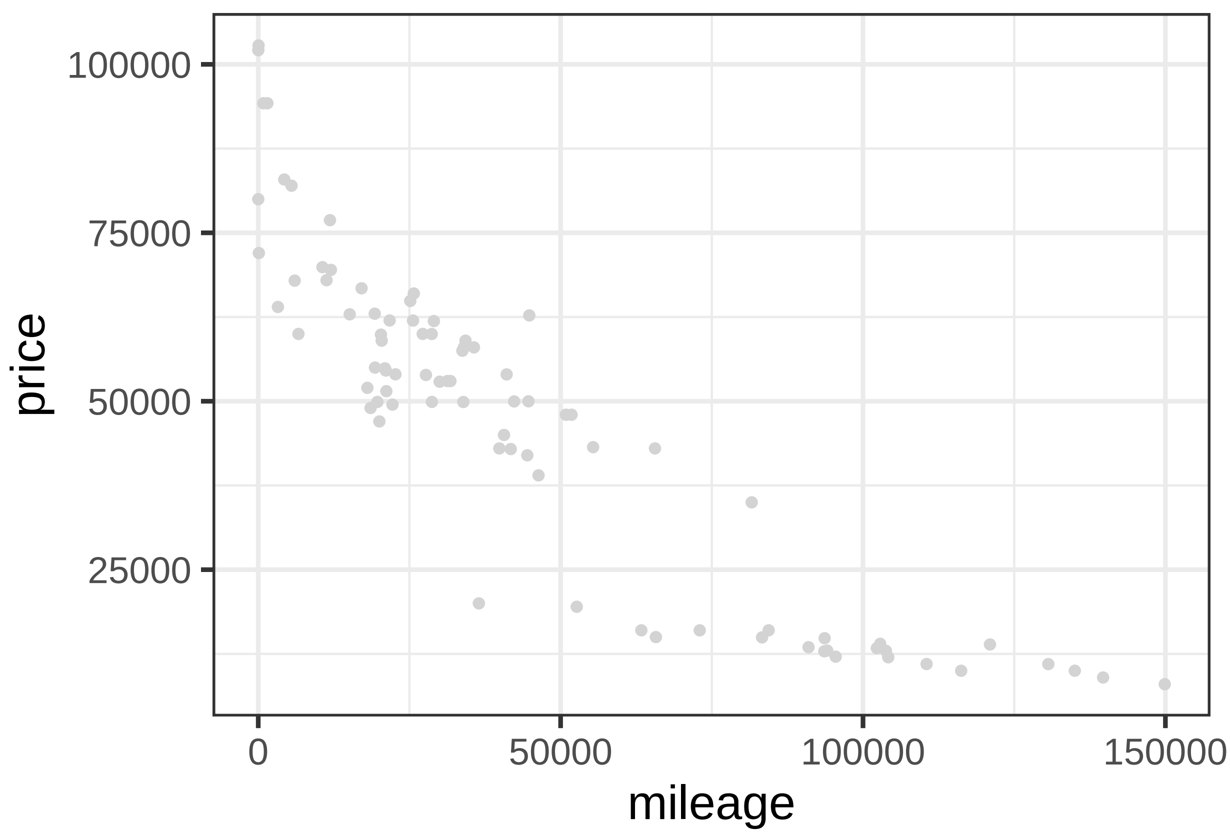
```
rmse(y_test, ypred_lm2)
```

```
## [1] 9052.279
```

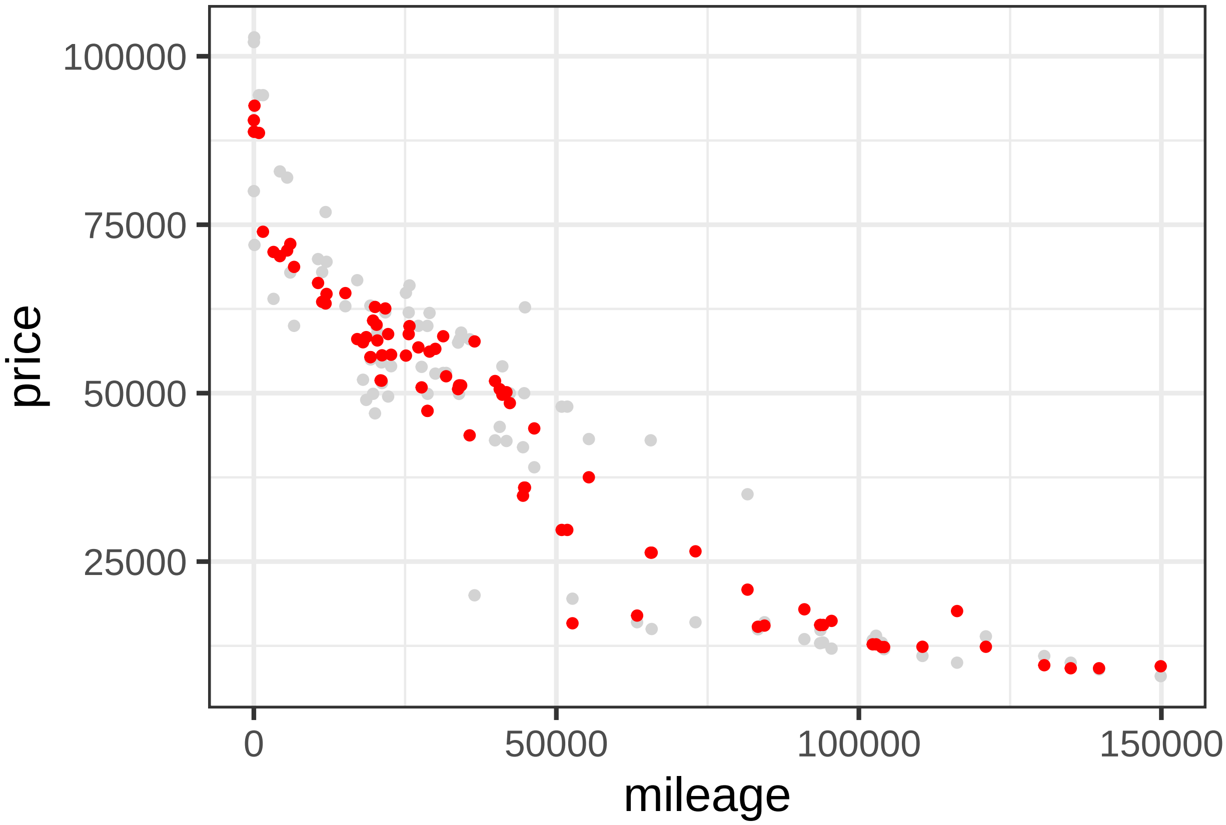
```r
rmse(y_test, ypred_knn5)
```

```
## [1] 9554.965
```

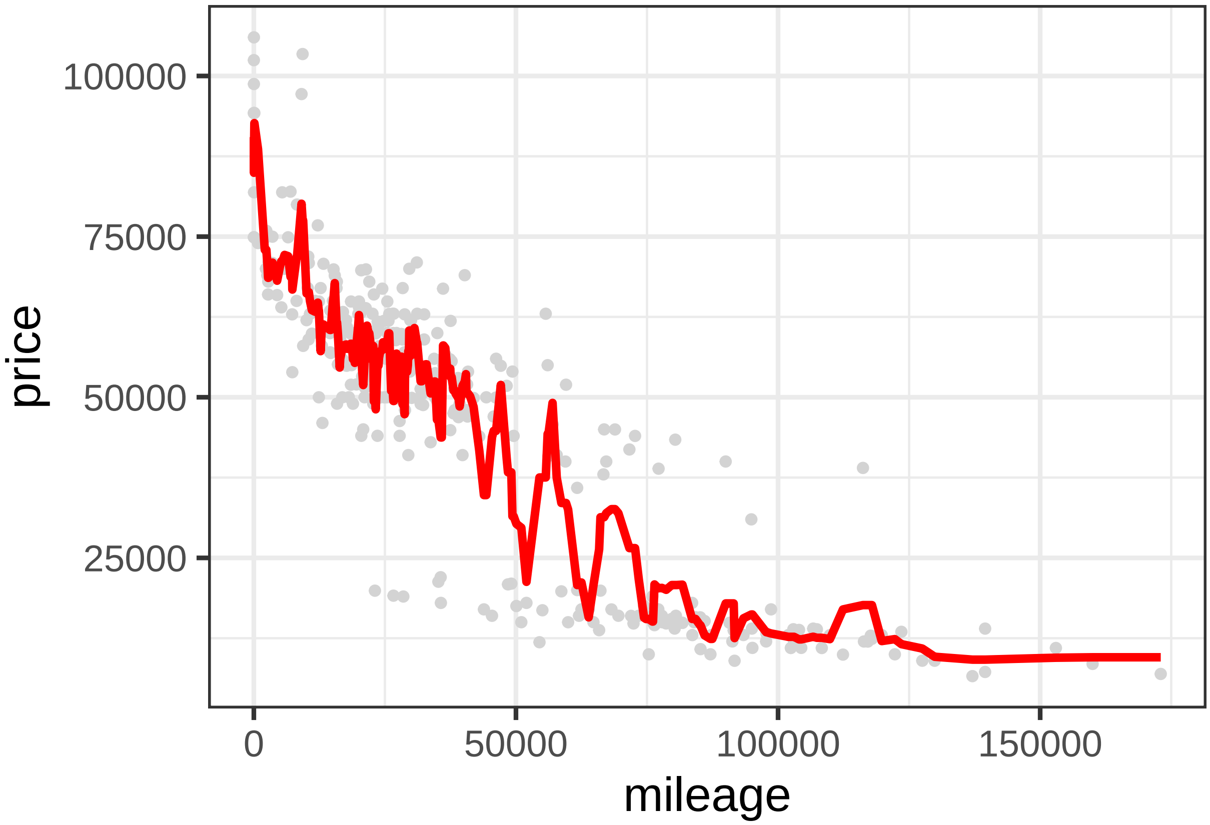
```r
#attach predictions to data frame
D_test$ypred_lm2 = ypred_lm2
D_test$ypred_knn5 = ypred_knn5

p_test = ggplot(data = D_test) +
  geom_point(mapping = aes(x = mileage, y = price), color='lightgrey') +
  theme_bw(base_size=18)
p_test
```




```r
p_test + geom_point(aes(x = mileage, y = ypred_knn5), color='red')
```




```r
#KNN variances
knn_model = knn.reg(X_train, X_train, y_train, k = 5)

D_train$ypred = knn_model$pred
p_train = ggplot(data = D_train) +
  geom_point(mapping = aes(x = mileage, y = price), color='lightgrey') +
  theme_bw(base_size=18)
p_train + geom_path(mapping = aes(x=mileage, y=ypred), color='red', size=1.5)
```




```
#K = 6
lm1 = lm(price ~ mileage, data=D_train)
lm2 = lm(price ~ poly(mileage, 2), data=D_train)
knn6 = knn.reg(train = X_train, test = X_test, y = y_train, k=6)


#rmse
rmse = function(y, ypred) {
  sqrt(mean(data.matrix((y-ypred)^2)))
}

ypred_lm1 = predict(lm1, X_test)
ypred_lm2 = predict(lm2, X_test)
ypred_knn6 = knn6$pred

rmse(y_test, ypred_lm1)
```

```
## [1] 10838.68
```

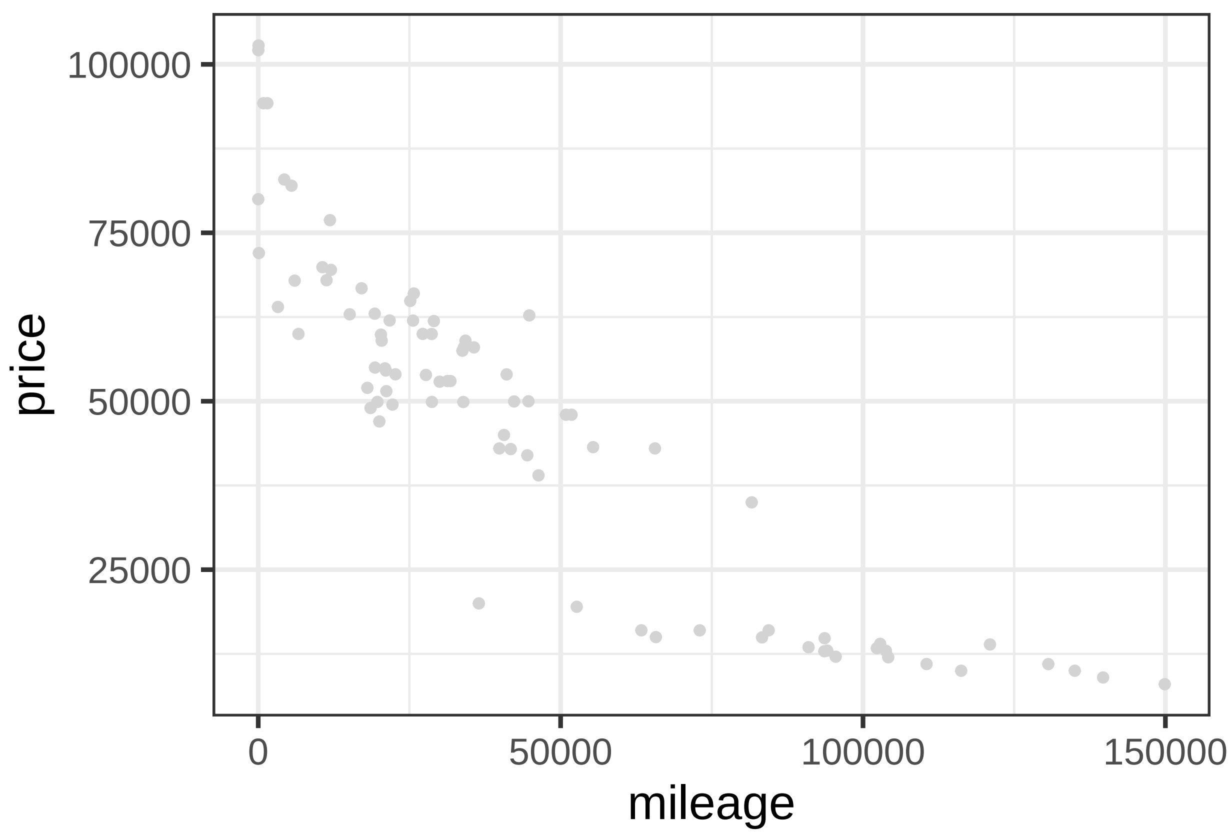
```
rmse(y_test, ypred_lm2)
```

```
## [1] 9052.279
```

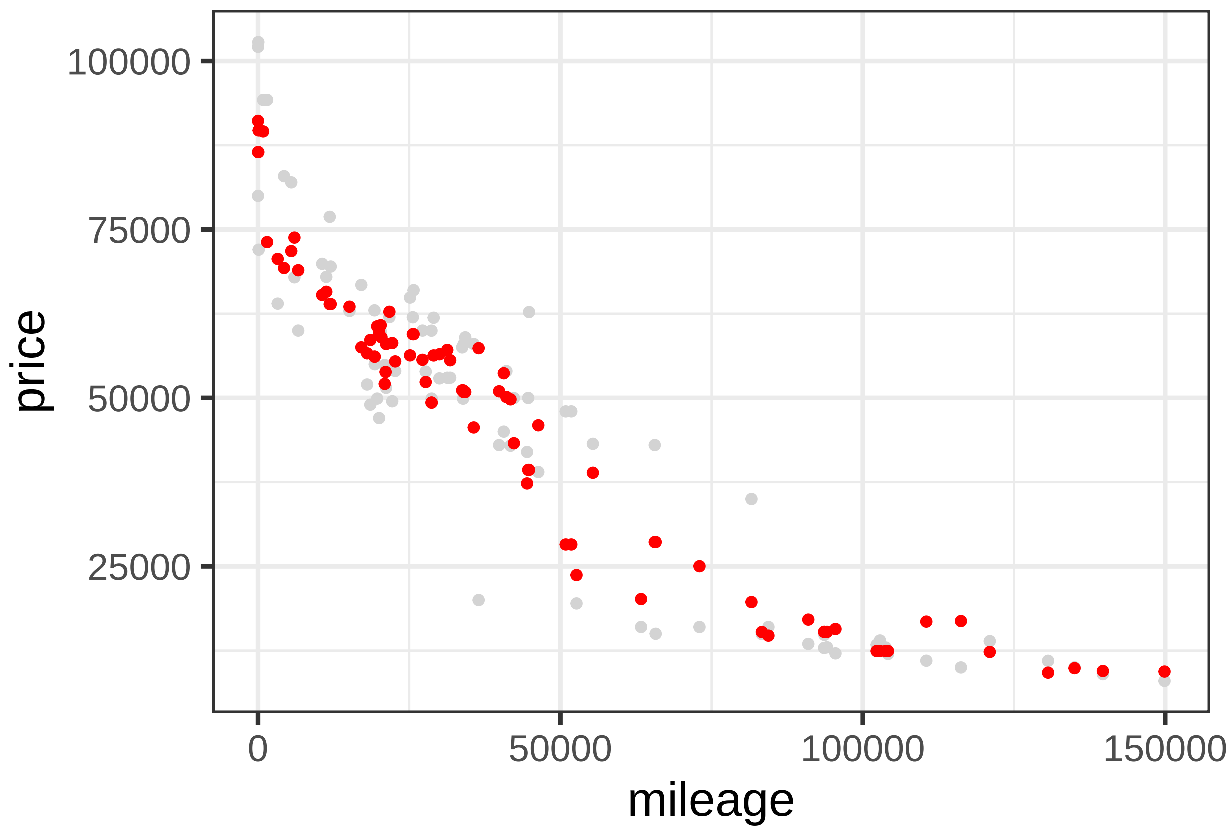
```r
rmse(y_test, ypred_knn6)
```

```
## [1] 9404.395
```

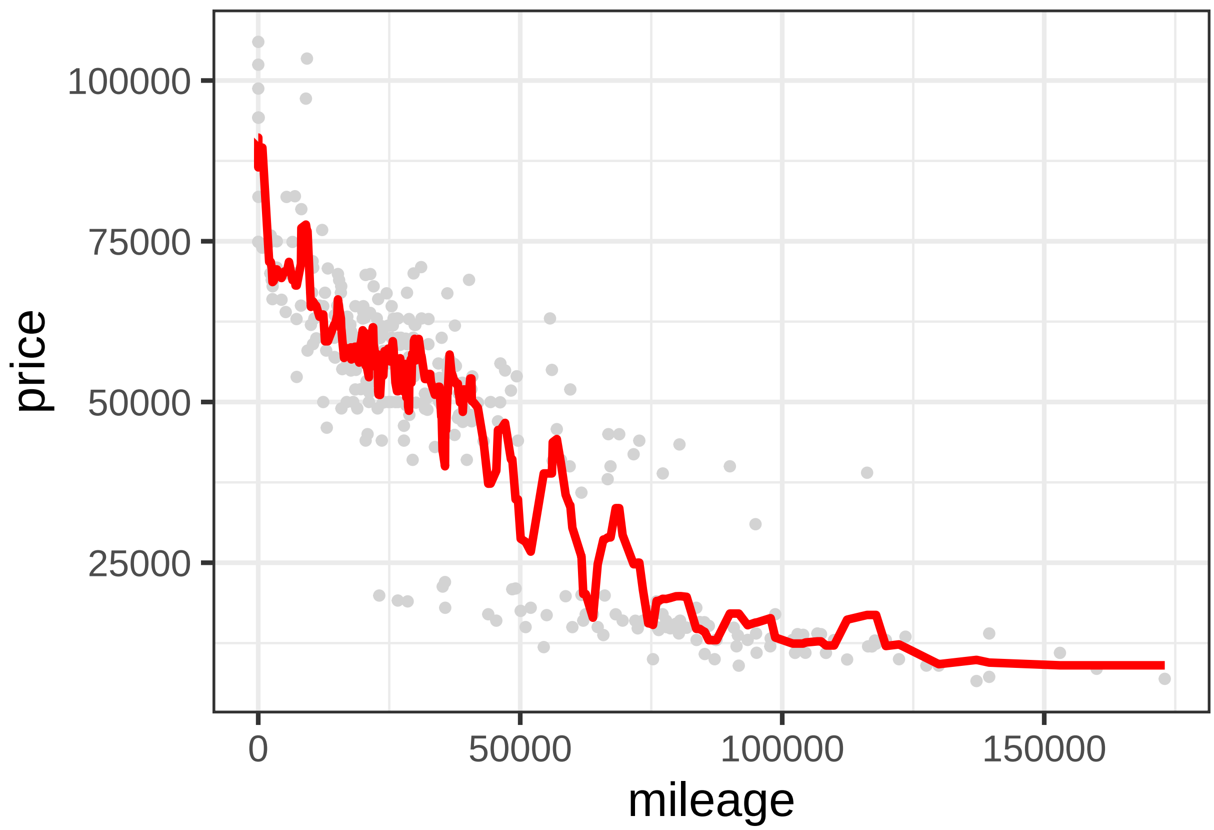
```r
#attach predictions to data frame
D_test$ypred_lm2 = ypred_lm2
D_test$ypred_knn6 = ypred_knn6

p_test = ggplot(data = D_test) +
  geom_point(mapping = aes(x = mileage, y = price), color='lightgrey') +
  theme_bw(base_size=18)
p_test
```




```r
p_test + geom_point(aes(x = mileage, y = ypred_knn6), color='red')
```




```
#KNN variances
knn_model = knn.reg(X_train, X_train, y_train, k = 6)

D_train$ypred = knn_model$pred
p_train = ggplot(data = D_train) +
  geom_point(mapping = aes(x = mileage, y = price), color='lightgrey') +
  theme_bw(base_size=18)
p_train + geom_path(mapping = aes(x=mileage, y=ypred), color='red', size=1.5)
```




```r
#K = 7
lm1 = lm(price ~ mileage, data=D_train)
lm2 = lm(price ~ poly(mileage, 2), data=D_train)
knn7 = knn.reg(train = X_train, test = X_test, y = y_train, k=7)


#rmse
rmse = function(y, ypred) {
  sqrt(mean(data.matrix((y-ypred)^2)))
}

ypred_lm1 = predict(lm1, X_test)
ypred_lm2 = predict(lm2, X_test)
ypred_knn7 = knn7$pred

rmse(y_test, ypred_lm1)
```

```
## [1] 10838.68
```

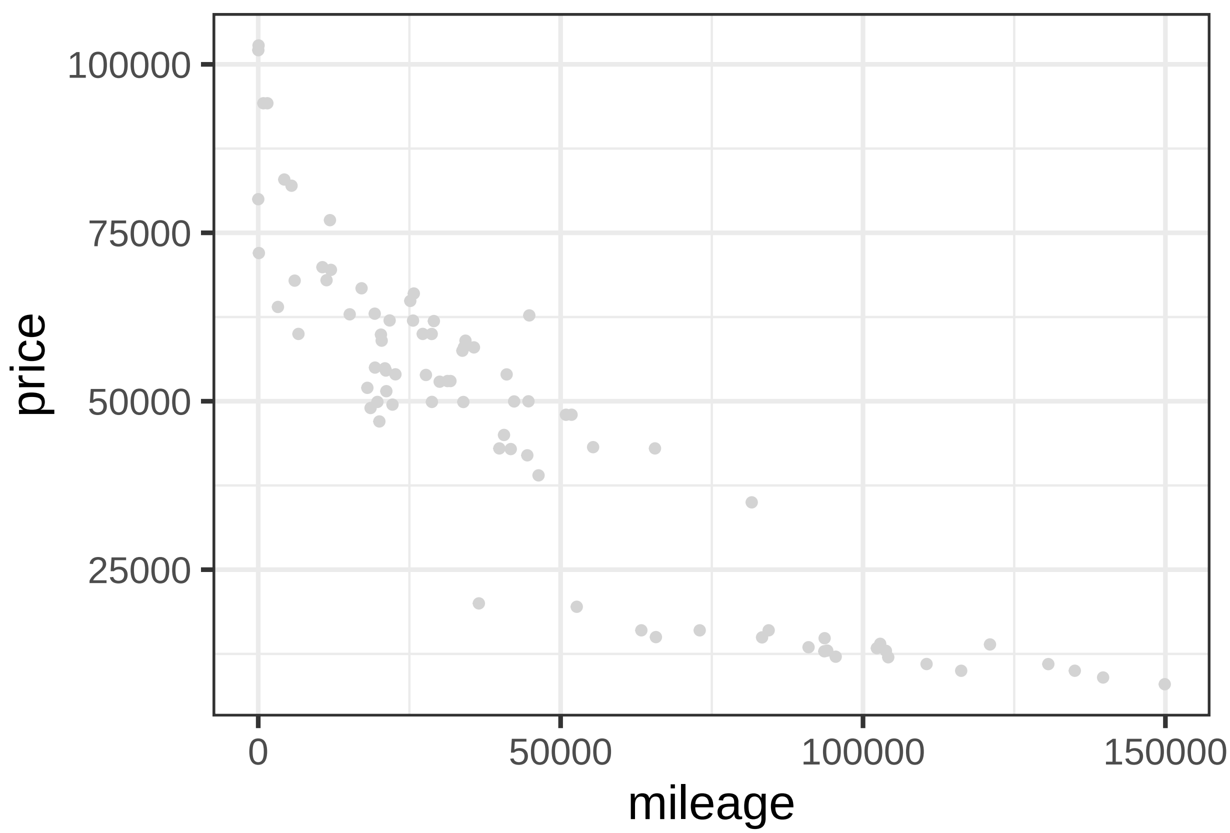
```r
rmse(y_test, ypred_lm2)
```

```
## [1] 9052.279
```

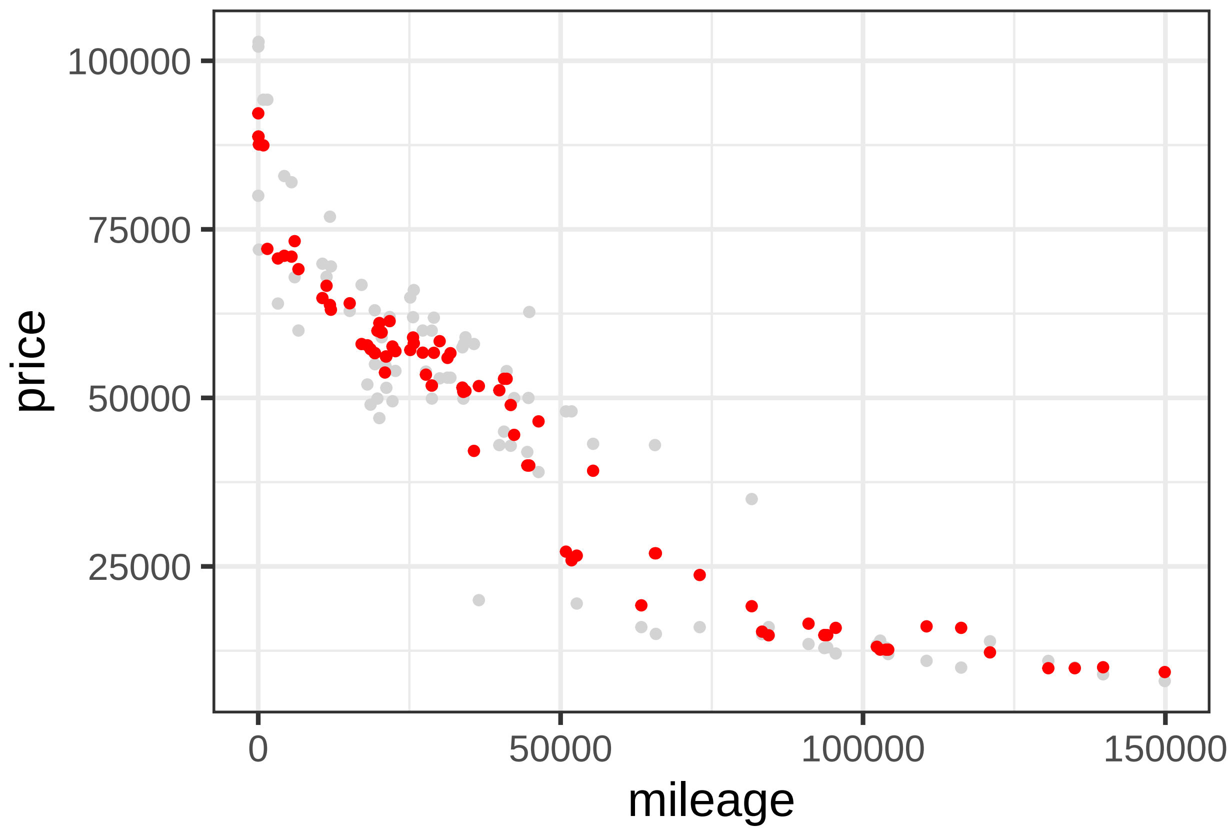
```r
rmse(y_test, ypred_knn7)
```

```
## [1] 9111.335
```

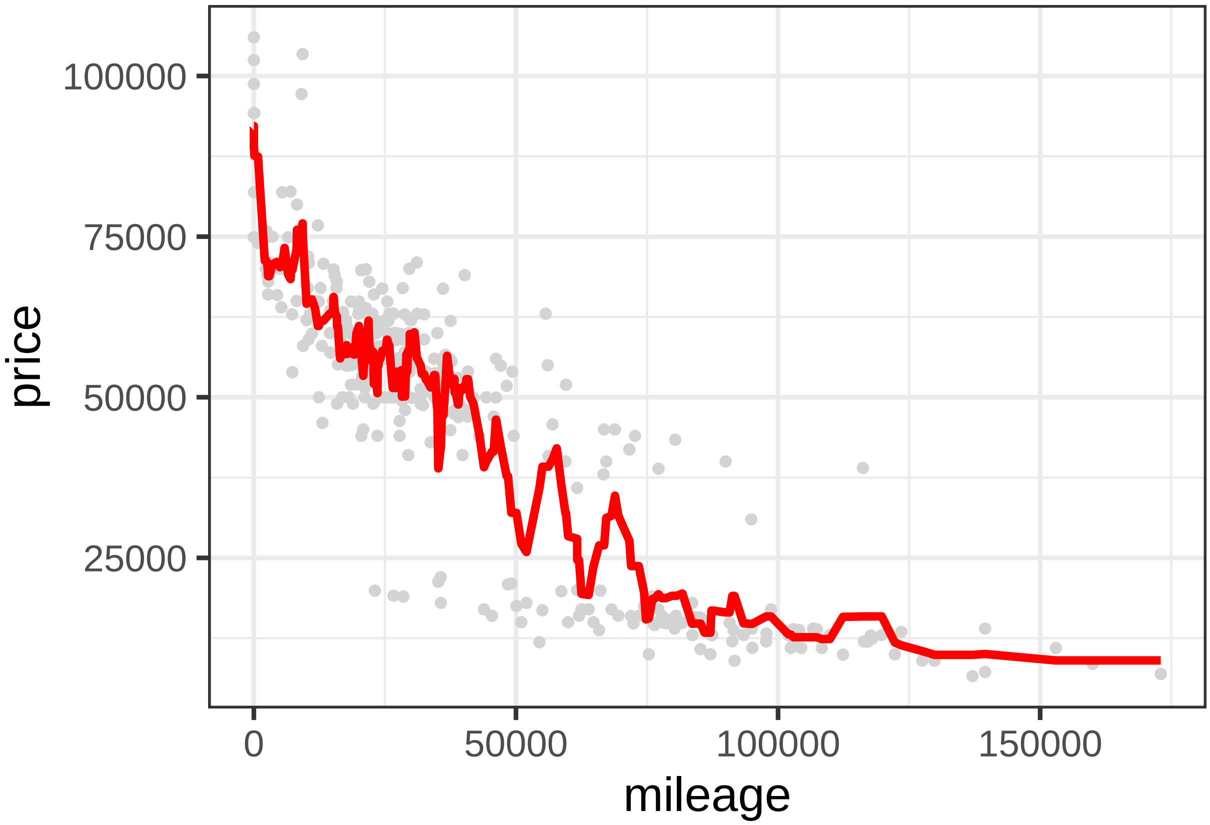
```r
#attach predictions to data frame
D_test$ypred_lm2 = ypred_lm2
D_test$ypred_knn7 = ypred_knn7

p_test = ggplot(data = D_test) +
  geom_point(mapping = aes(x = mileage, y = price), color='lightgrey') +
  theme_bw(base_size=18)
p_test
```




```r
p_test + geom_point(aes(x = mileage, y = ypred_knn7), color='red')
```




```r
#KNN variances
knn_model = knn.reg(X_train, X_train, y_train, k = 7)

D_train$ypred = knn_model$pred
p_train = ggplot(data = D_train) +
  geom_point(mapping = aes(x = mileage, y = price), color='lightgrey') +
  theme_bw(base_size=18)
p_train + geom_path(mapping = aes(x=mileage, y=ypred), color='red', size=1.5)
```




```r
#K = 8
lm1 = lm(price ~ mileage, data=D_train)
lm2 = lm(price ~ poly(mileage, 2), data=D_train)
knn8 = knn.reg(train = X_train, test = X_test, y = y_train, k=8)


#rmse
rmse = function(y, ypred) {
  sqrt(mean(data.matrix((y-ypred)^2)))
}

ypred_lm1 = predict(lm1, X_test)
ypred_lm2 = predict(lm2, X_test)
ypred_knn8 = knn8$pred

rmse(y_test, ypred_lm1)
```

```
## [1] 10838.68
```

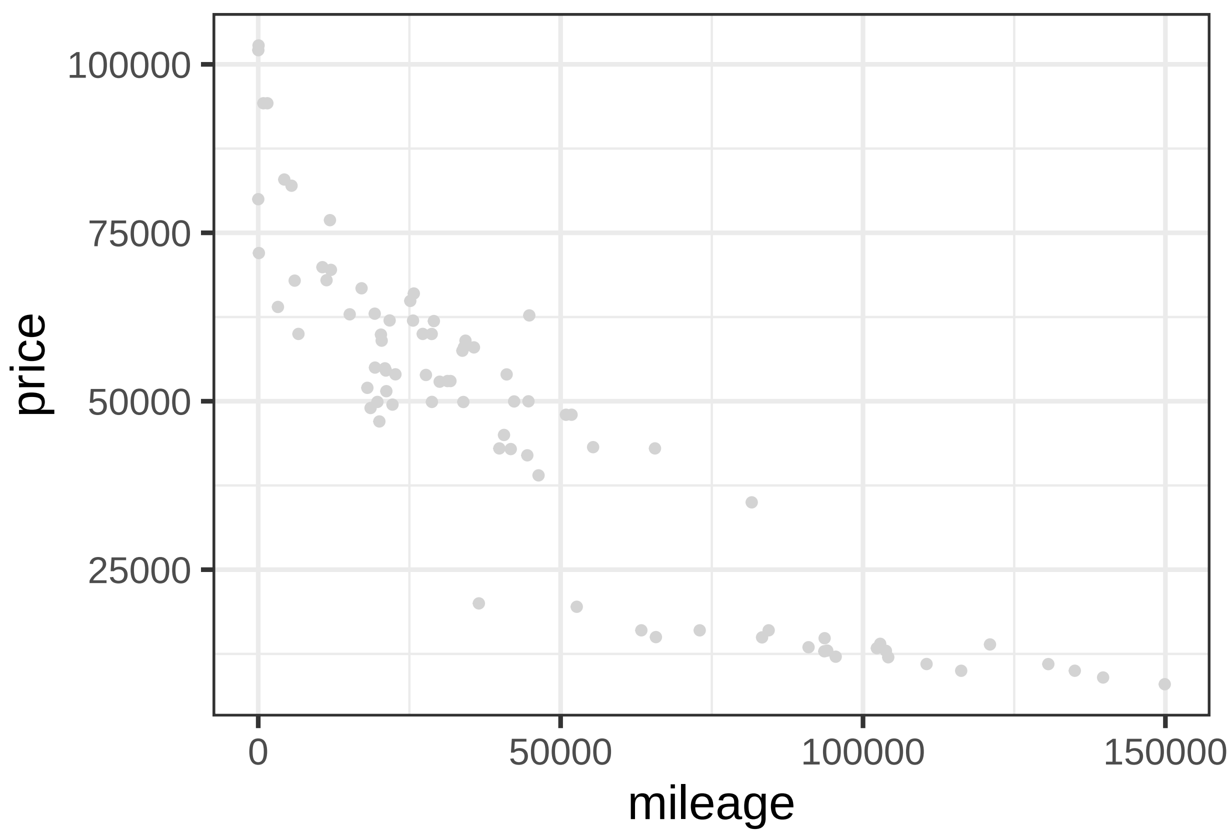
```r
rmse(y_test, ypred_lm2)
```

```
## [1] 9052.279
```

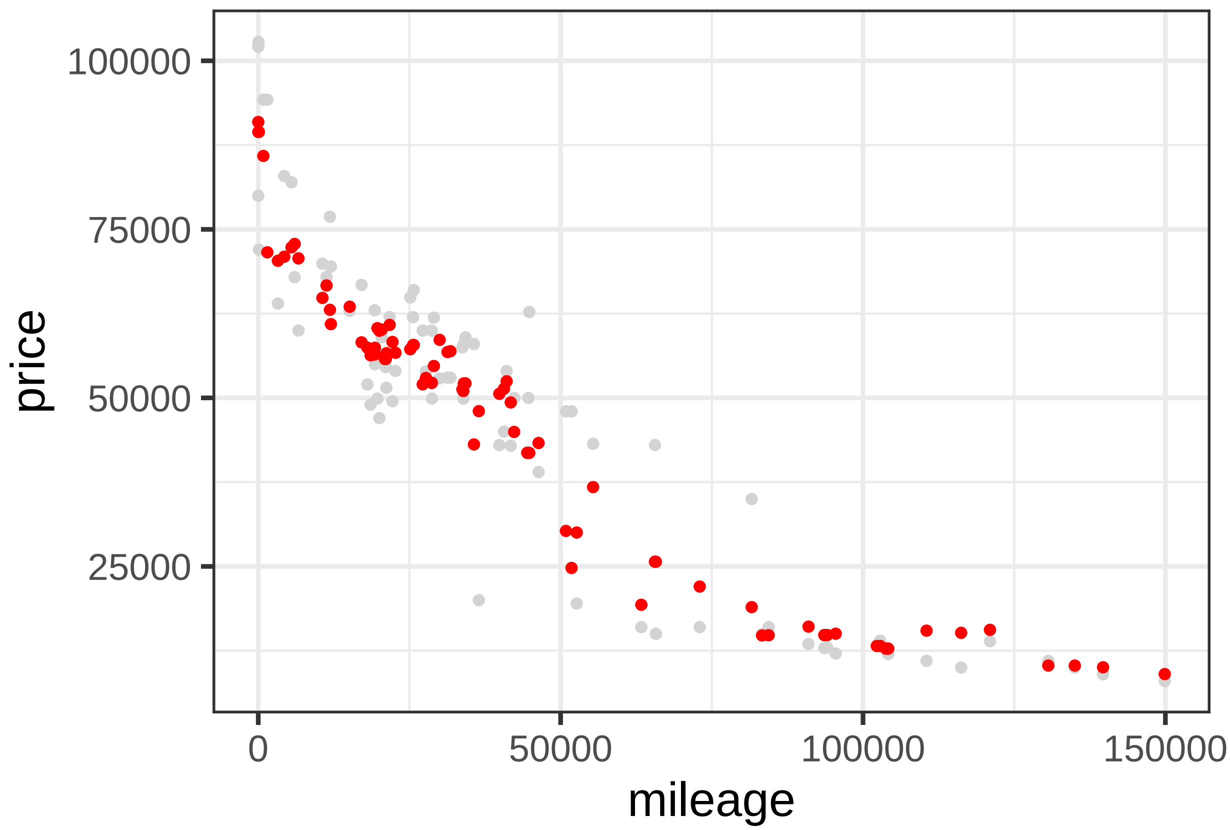
```r
rmse(y_test, ypred_knn8)
```

```
## [1] 8902.22
```

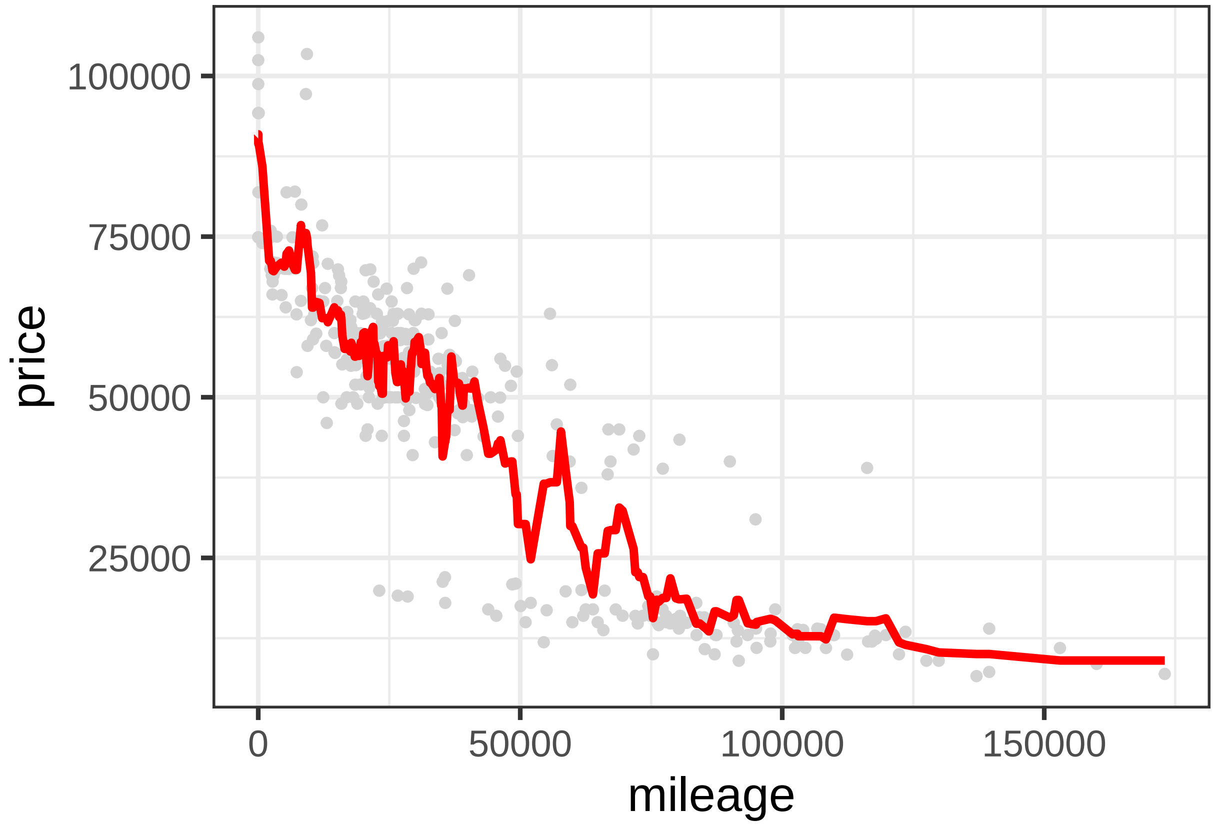
```r
#attach predictions to data frame
D_test$ypred_lm2 = ypred_lm2
D_test$ypred_knn8 = ypred_knn8

p_test = ggplot(data = D_test) +
  geom_point(mapping = aes(x = mileage, y = price), color='lightgrey') +
  theme_bw(base_size=18)
p_test
```




```r
p_test + geom_point(aes(x = mileage, y = ypred_knn8), color='red')
```




```r
#KNN variances
knn_model = knn.reg(X_train, X_train, y_train, k = 8)

D_train$ypred = knn_model$pred
p_train = ggplot(data = D_train) +
  geom_point(mapping = aes(x = mileage, y = price), color='lightgrey') +
  theme_bw(base_size=18)
p_train + geom_path(mapping = aes(x=mileage, y=ypred), color='red', size=1.5)
```




```
#K = 9
lm1 = lm(price ~ mileage, data=D_train)
lm2 = lm(price ~ poly(mileage, 2), data=D_train)
knn9 = knn.reg(train = X_train, test = X_test, y = y_train, k=9)


#rmse
rmse = function(y, ypred) {
  sqrt(mean(data.matrix((y-ypred)^2)))
}

ypred_lm1 = predict(lm1, X_test)
ypred_lm2 = predict(lm2, X_test)
ypred_knn9 = knn9$pred

rmse(y_test, ypred_lm1)
```

```
## [1] 10838.68
```

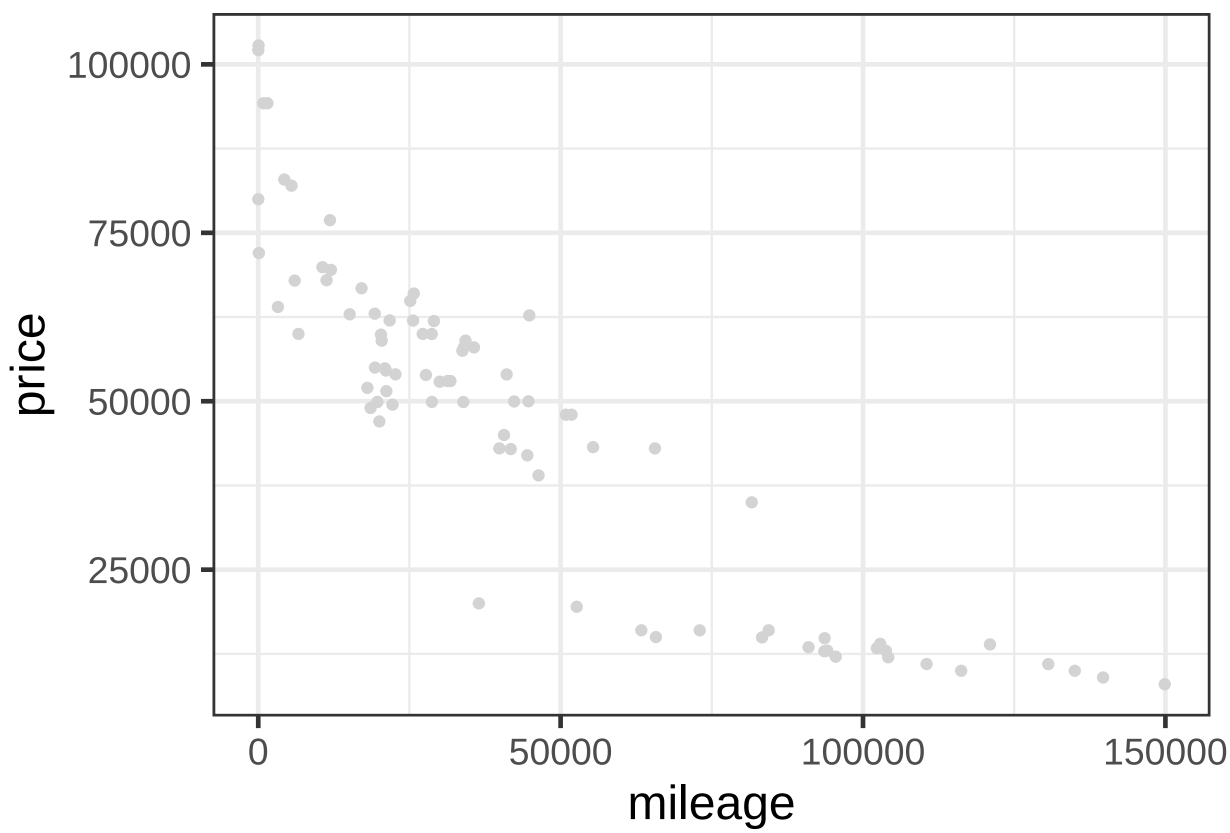
```
rmse(y_test, ypred_lm2)
```

```
## [1] 9052.279
```

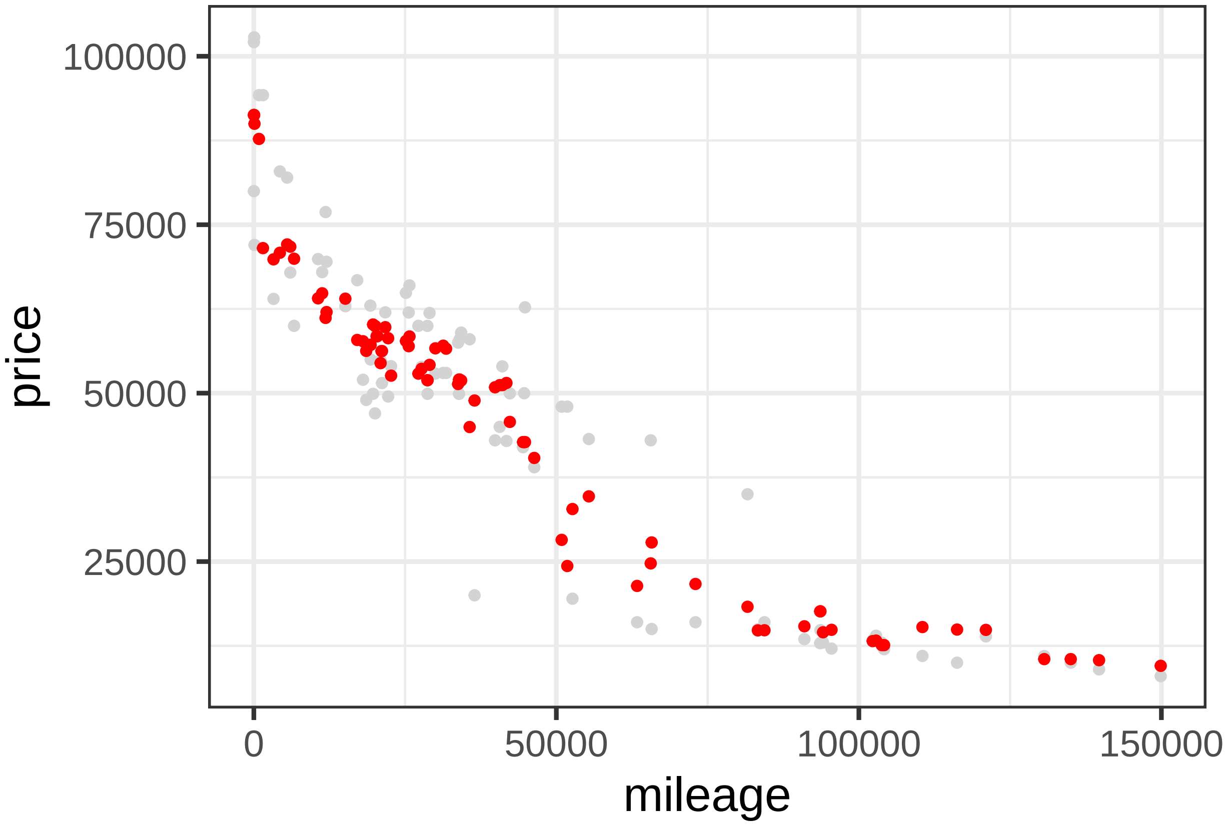
```r
rmse(y_test, ypred_knn9)
```

```
## [1] 9040.162
```

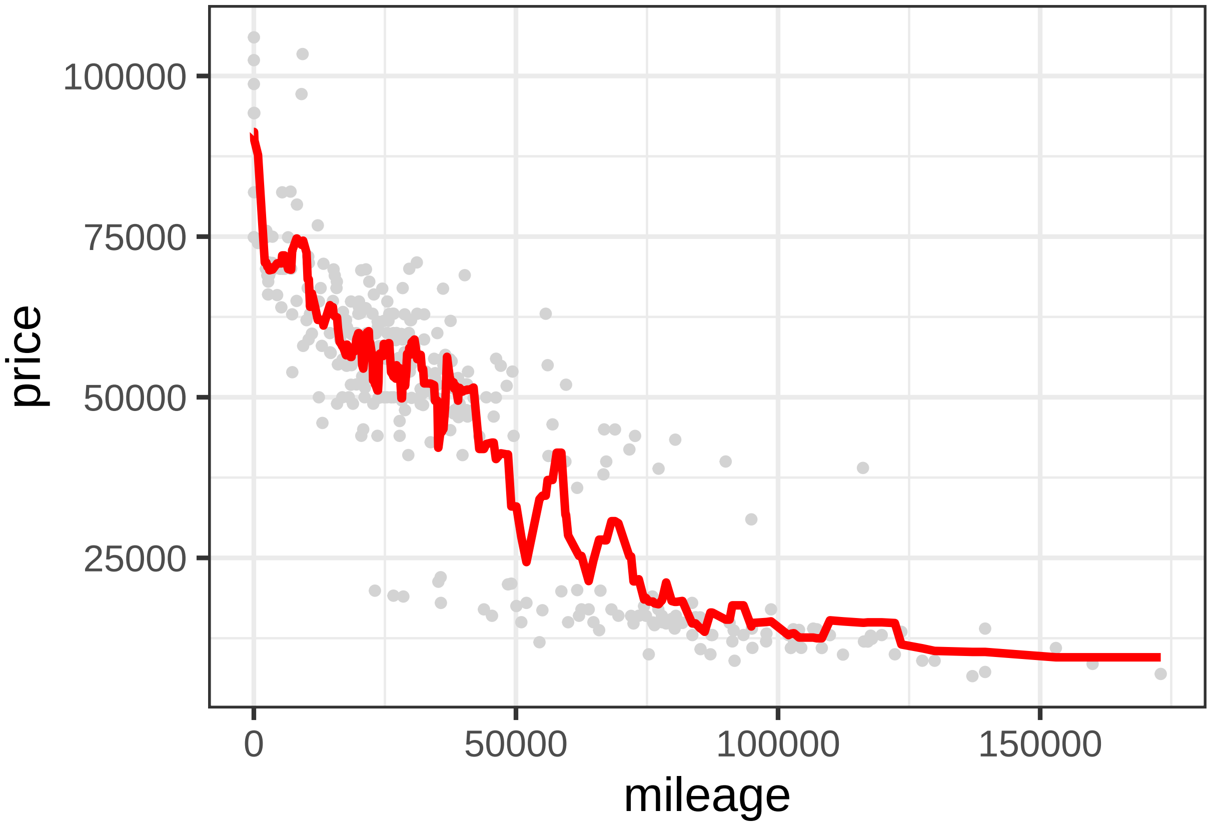
```r
#attach predictions to data frame
D_test$ypred_lm2 = ypred_lm2
D_test$ypred_knn9 = ypred_knn9

p_test = ggplot(data = D_test) +
  geom_point(mapping = aes(x = mileage, y = price), color='lightgrey') +
  theme_bw(base_size=18)
p_test
```




```r
p_test + geom_point(aes(x = mileage, y = ypred_knn9), color='red')
```




```
#KNN variances
knn_model = knn.reg(X_train, X_train, y_train, k = 9)

D_train$ypred = knn_model$pred
p_train = ggplot(data = D_train) +
  geom_point(mapping = aes(x = mileage, y = price), color='lightgrey') +
  theme_bw(base_size=18)
p_train + geom_path(mapping = aes(x=mileage, y=ypred), color='red', size=1.5)
```




```
#K = 6
lm1 = lm(price ~ mileage, data=D_train)
lm2 = lm(price ~ poly(mileage, 2), data=D_train)
knn6 = knn.reg(train = X_train, test = X_test, y = y_train, k=6)


#rmse
rmse = function(y, ypred) {
  sqrt(mean(data.matrix((y-ypred)^2)))
}

ypred_lm1 = predict(lm1, X_test)
ypred_lm2 = predict(lm2, X_test)
ypred_knn6 = knn6$pred

rmse(y_test, ypred_lm1)
```

```
## [1] 10838.68
```

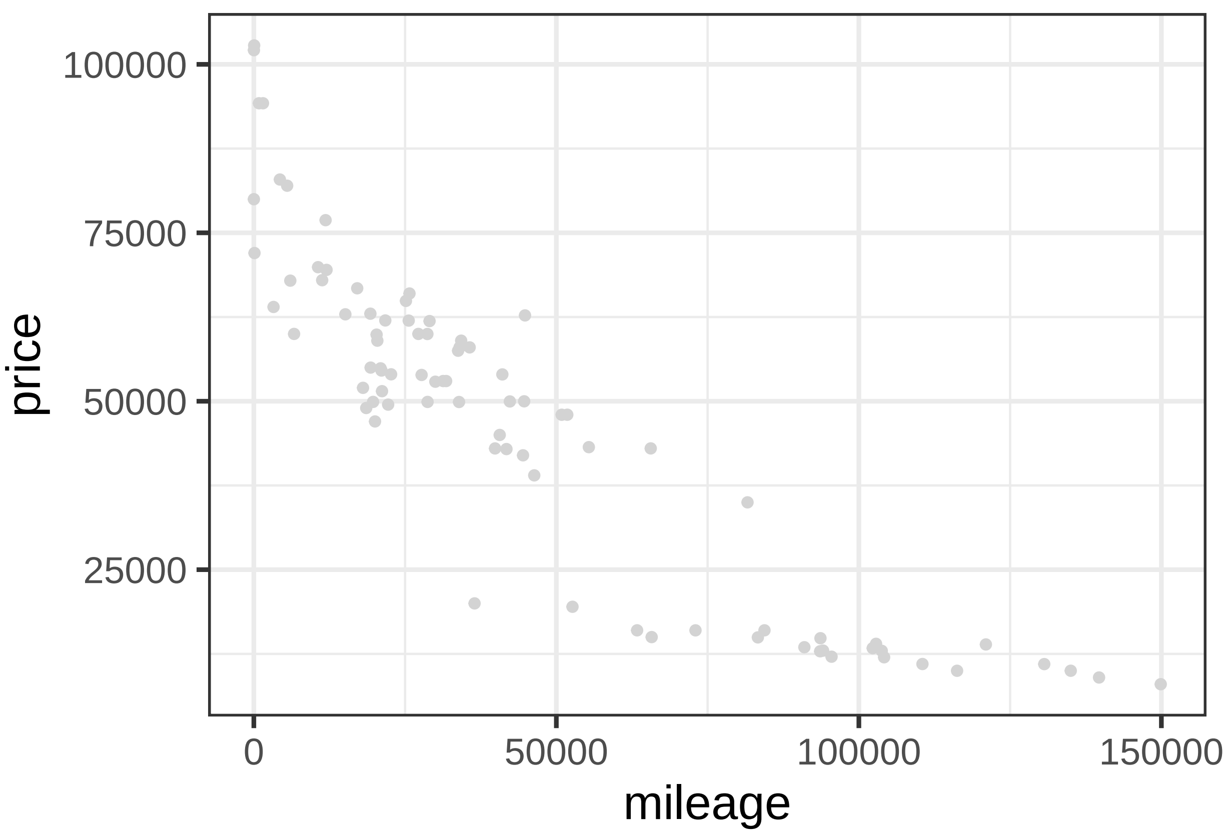
```
rmse(y_test, ypred_lm2)
```

```
## [1] 9052.279
```

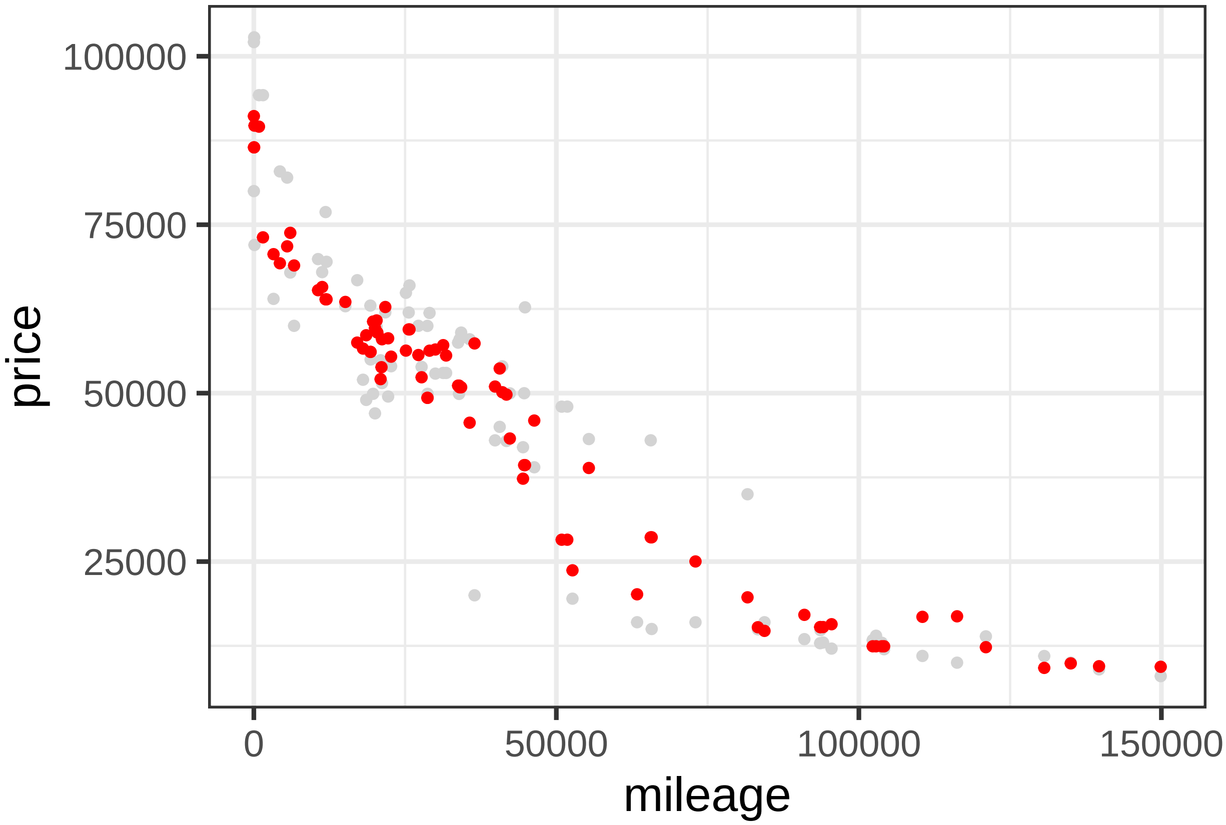
```r
rmse(y_test, ypred_knn6)
```

```
## [1] 9404.395
```

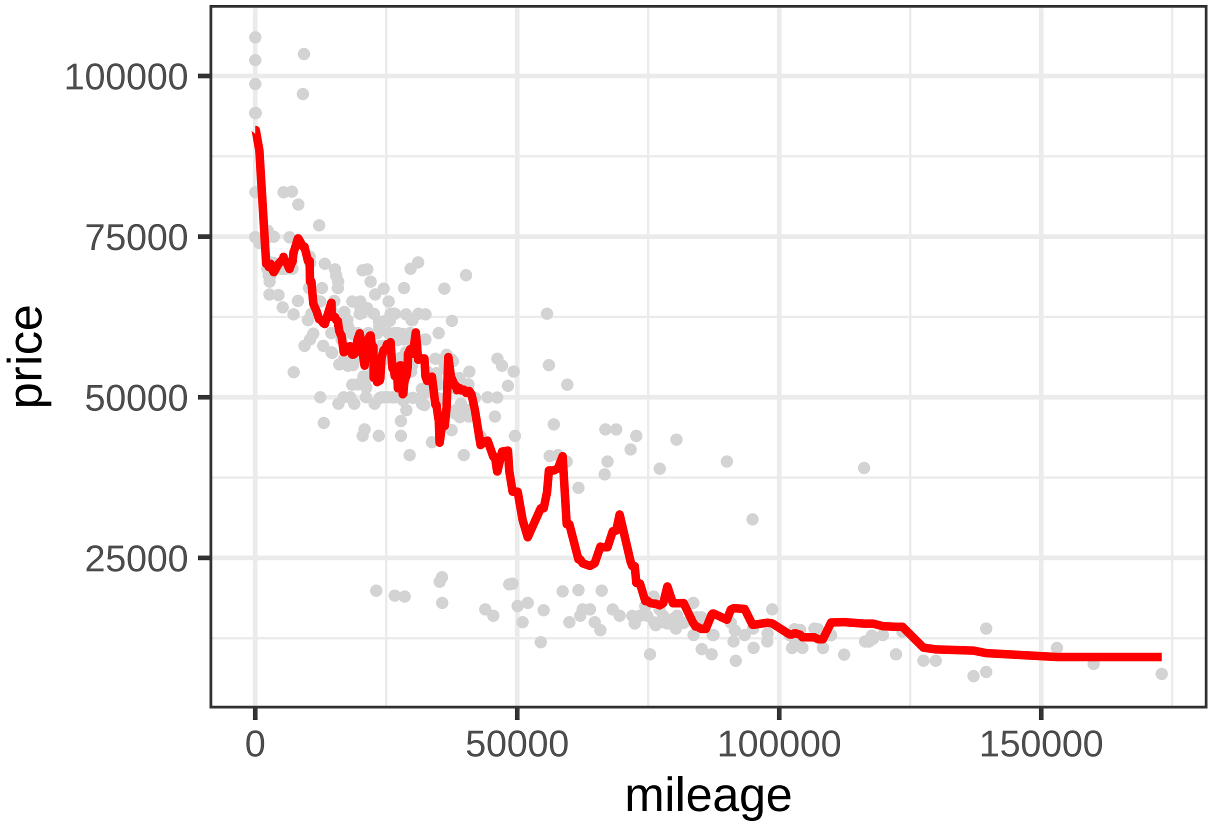
```r
#attach predictions to data frame
D_test$ypred_lm2 = ypred_lm2
D_test$ypred_knn6 = ypred_knn6

p_test = ggplot(data = D_test) +
  geom_point(mapping = aes(x = mileage, y = price), color='lightgrey') +
  theme_bw(base_size=18)
p_test
```




```r
p_test + geom_point(aes(x = mileage, y = ypred_knn6), color='red')
```




```r
#KNN variances
knn_model = knn.reg(X_train, X_train, y_train, k = 10)

D_train$ypred = knn_model$pred
p_train = ggplot(data = D_train) +
  geom_point(mapping = aes(x = mileage, y = price), color='lightgrey') +
  theme_bw(base_size=18)
p_train + geom_path(mapping = aes(x=mileage, y=ypred), color='red', size=1.5)
```




```
#K = 15
lm1 = lm(price ~ mileage, data=D_train)
lm2 = lm(price ~ poly(mileage, 2), data=D_train)
knn15 = knn.reg(train = X_train, test = X_test, y = y_train, k=15)


#rmse
rmse = function(y, ypred) {
  sqrt(mean(data.matrix((y-ypred)^2)))
}

ypred_lm1 = predict(lm1, X_test)
ypred_lm2 = predict(lm2, X_test)
ypred_knn15 = knn15$pred

rmse(y_test, ypred_lm1)
```

```
## [1] 10838.68
```

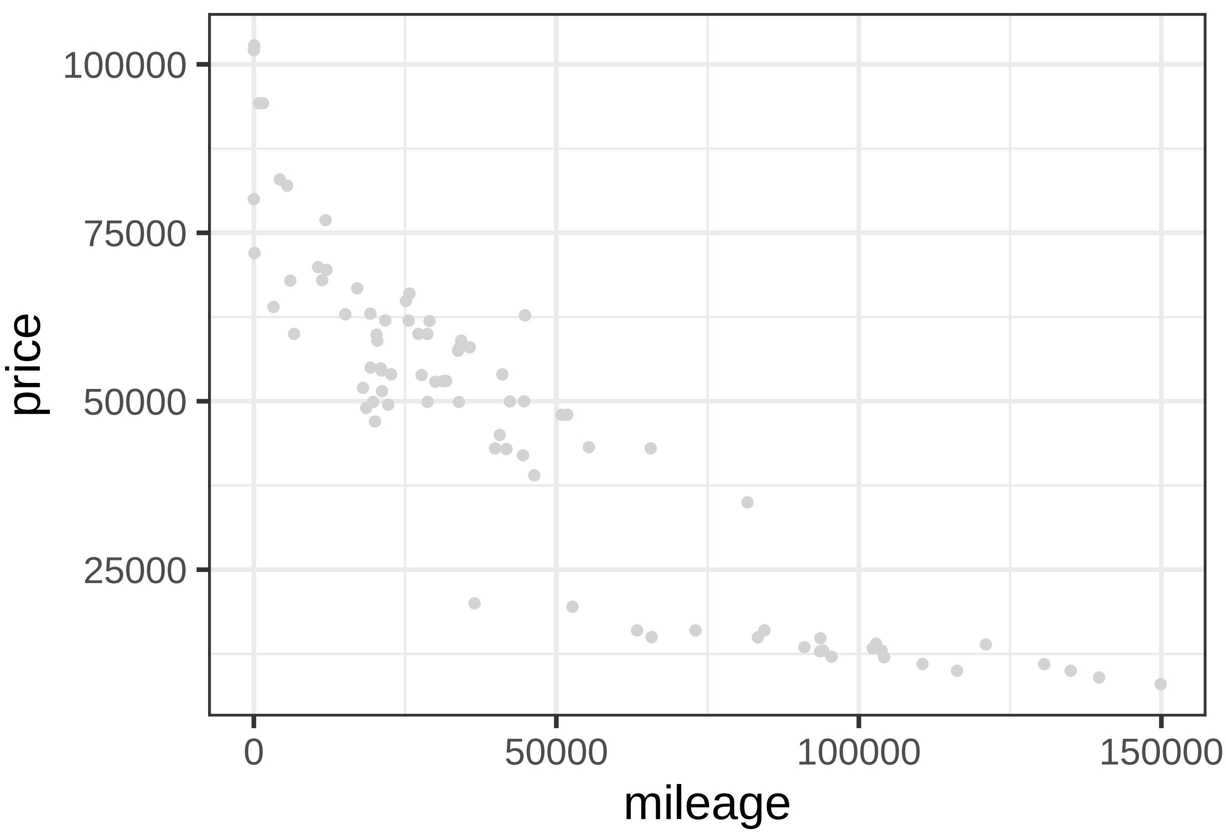
```
rmse(y_test, ypred_lm2)
```

```
## [1] 9052.279
```

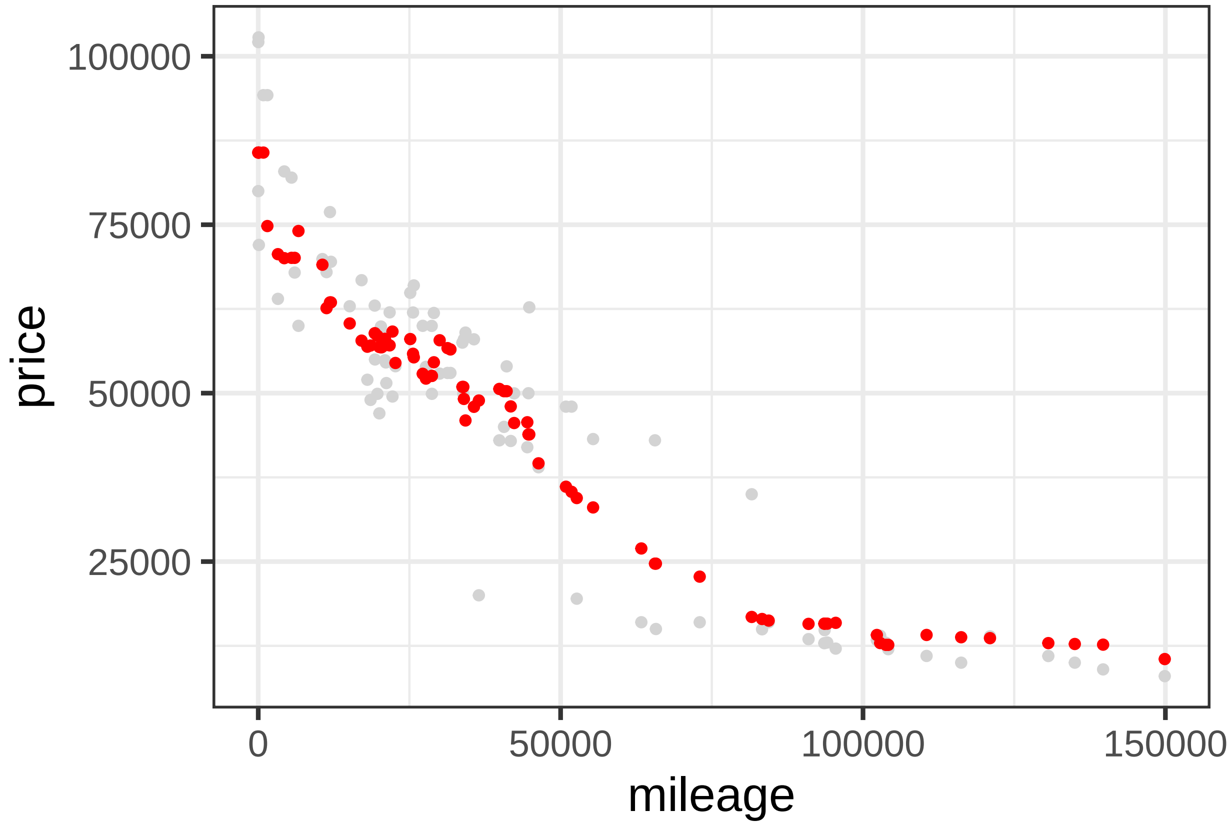
```r
rmse(y_test, ypred_knn15)
```

```
## [1] 8760.234
```

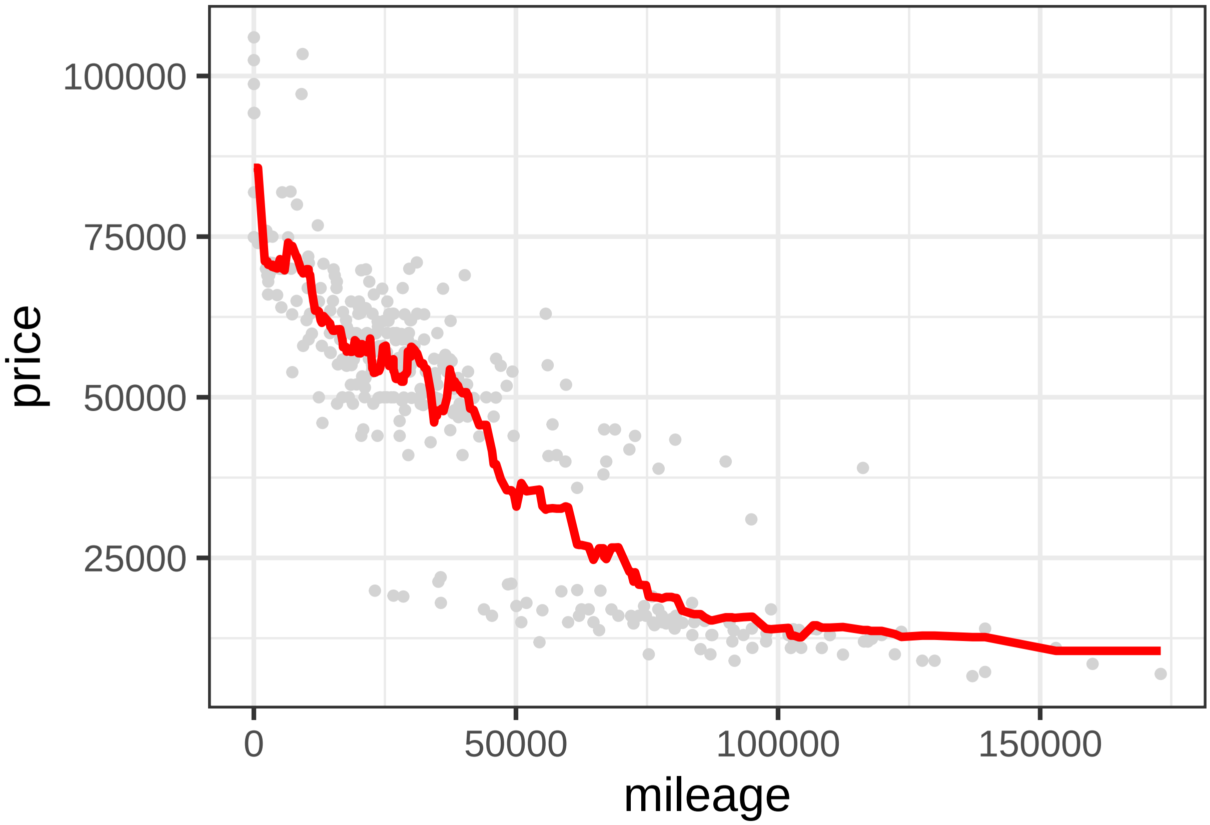
```r
#attach predictions to data frame
D_test$ypred_lm2 = ypred_lm2
D_test$ypred_knn15 = ypred_knn15

p_test = ggplot(data = D_test) +
  geom_point(mapping = aes(x = mileage, y = price), color='lightgrey') +
  theme_bw(base_size=18)
p_test
```




```r
p_test + geom_point(aes(x = mileage, y = ypred_knn15), color='red')
```




```
#KNN variances
knn_model = knn.reg(X_train, X_train, y_train, k = 15)

D_train$ypred = knn_model$pred
p_train = ggplot(data = D_train) +
  geom_point(mapping = aes(x = mileage, y = price), color='lightgrey') +
  theme_bw(base_size=18)
p_train + geom_path(mapping = aes(x=mileage, y=ypred), color='red', size=1.5)
```




```
#K = 30
lm1 = lm(price ~ mileage, data=D_train)
lm2 = lm(price ~ poly(mileage, 2), data=D_train)
knn30 = knn.reg(train = X_train, test = X_test, y = y_train, k=30)


#rmse
rmse = function(y, ypred) {
  sqrt(mean(data.matrix((y-ypred)^2)))
}

ypred_lm1 = predict(lm1, X_test)
ypred_lm2 = predict(lm2, X_test)
ypred_knn30 = knn30$pred

rmse(y_test, ypred_lm1)
```

```
## [1] 10838.68
```

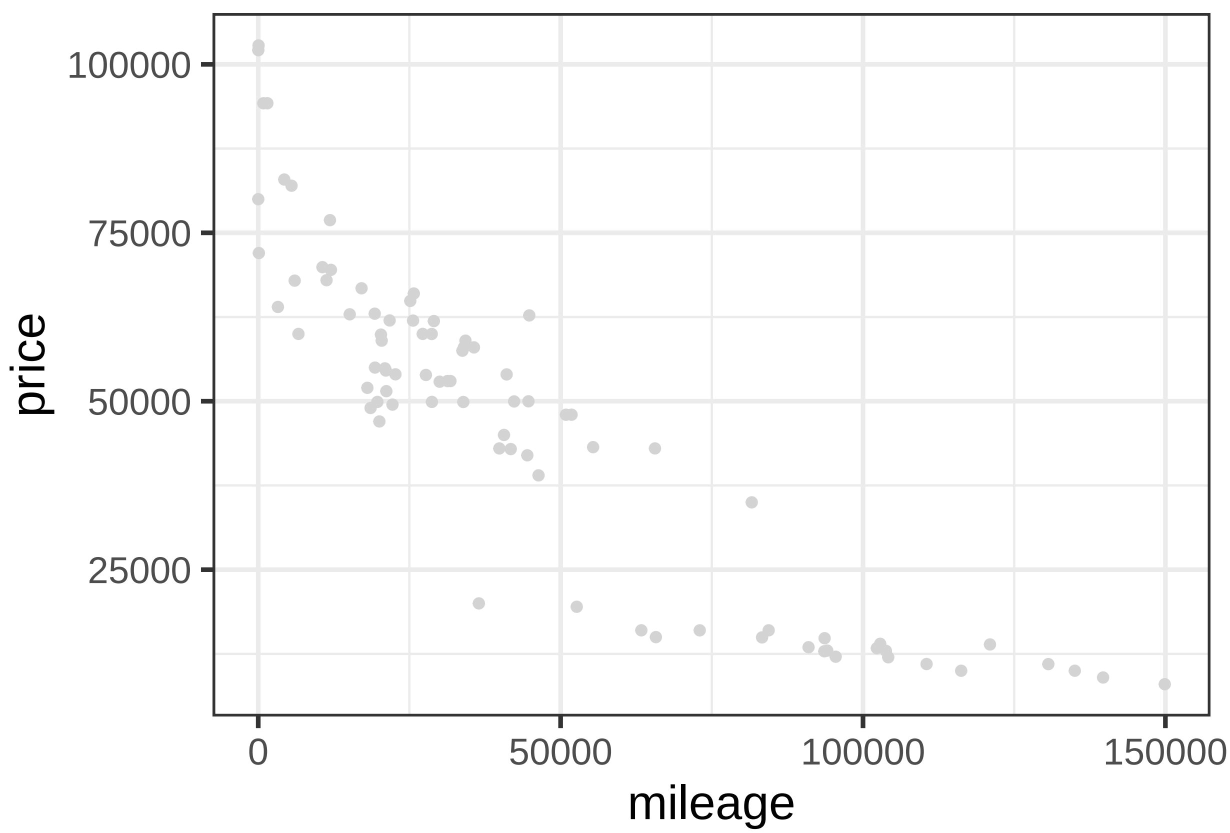
```
rmse(y_test, ypred_lm2)
```

```
## [1] 9052.279
```

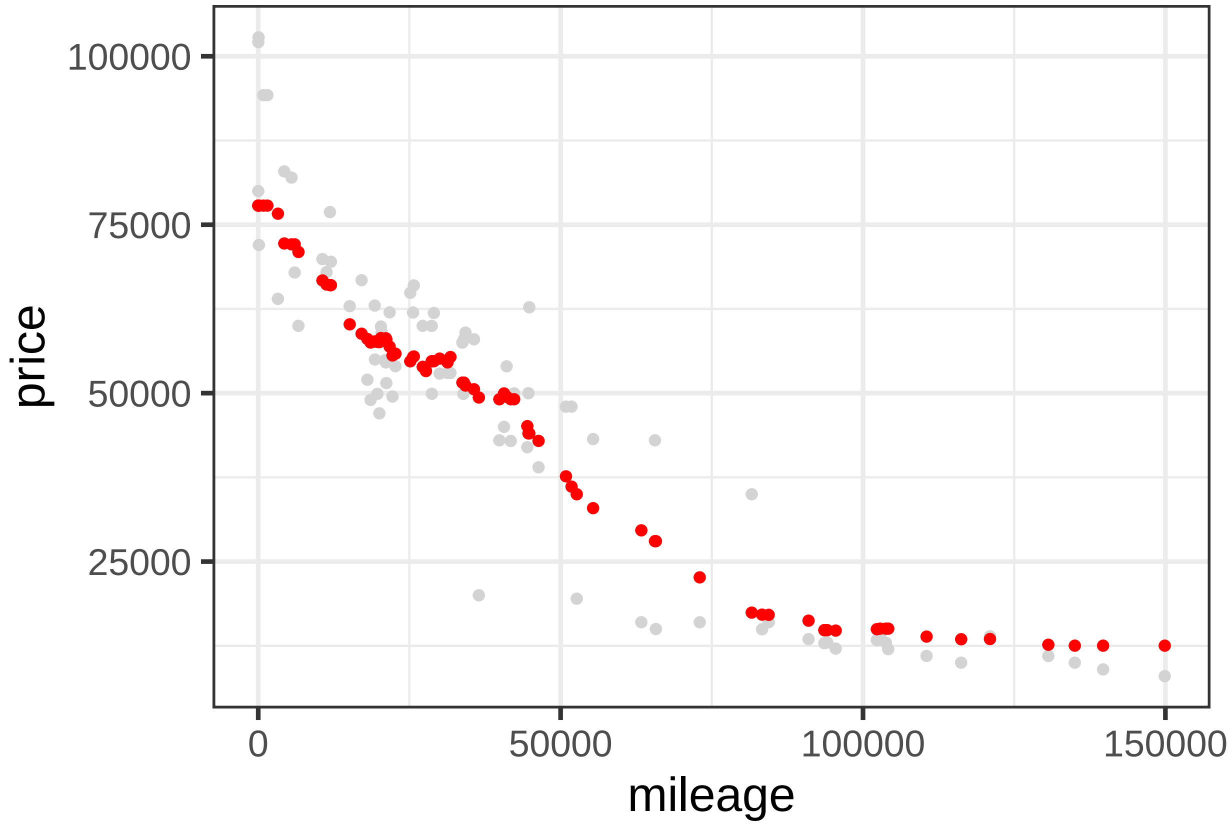
```r
rmse(y_test, ypred_knn30)
```

```
## [1] 8862.302
```

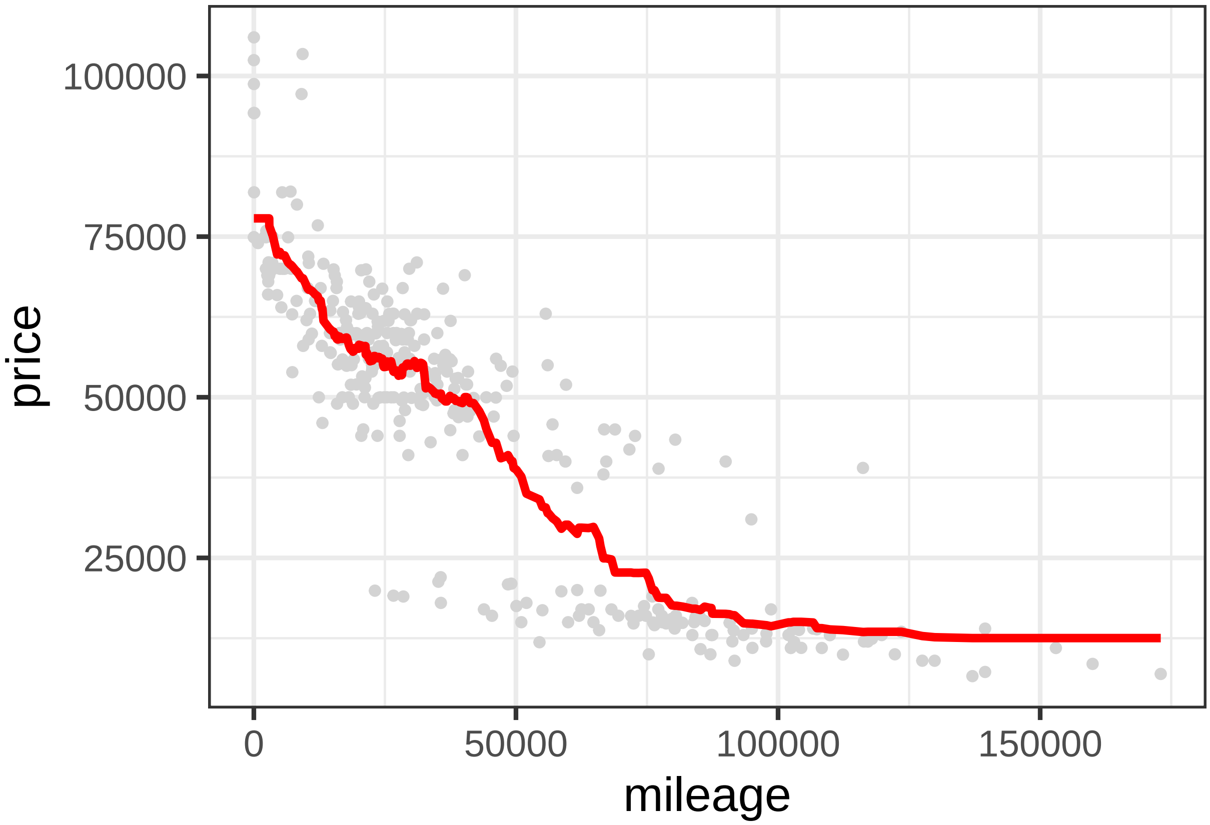
```r
#attach predictions to data frame
D_test$ypred_lm2 = ypred_lm2
D_test$ypred_knn30 = ypred_knn30

p_test = ggplot(data = D_test) +
  geom_point(mapping = aes(x = mileage, y = price), color='lightgrey') +
  theme_bw(base_size=18)
p_test
```




```r
p_test + geom_point(aes(x = mileage, y = ypred_knn30), color='red')
```




```
#KNN variances
knn_model = knn.reg(X_train, X_train, y_train, k = 30)

D_train$ypred = knn_model$pred
p_train = ggplot(data = D_train) +
  geom_point(mapping = aes(x = mileage, y = price), color='lightgrey') +
  theme_bw(base_size=18)
p_train + geom_path(mapping = aes(x=mileage, y=ypred), color='red', size=1.5)
```




```r
#K = 50
lm1 = lm(price ~ mileage, data=D_train)
lm2 = lm(price ~ poly(mileage, 2), data=D_train)
knn50 = knn.reg(train = X_train, test = X_test, y = y_train, k=50)


#rmse
rmse = function(y, ypred) {
  sqrt(mean(data.matrix((y-ypred)^2)))
}

ypred_lm1 = predict(lm1, X_test)
ypred_lm2 = predict(lm2, X_test)
ypred_knn50 = knn50$pred

rmse(y_test, ypred_lm1)
```

```
## [1] 10838.68
```

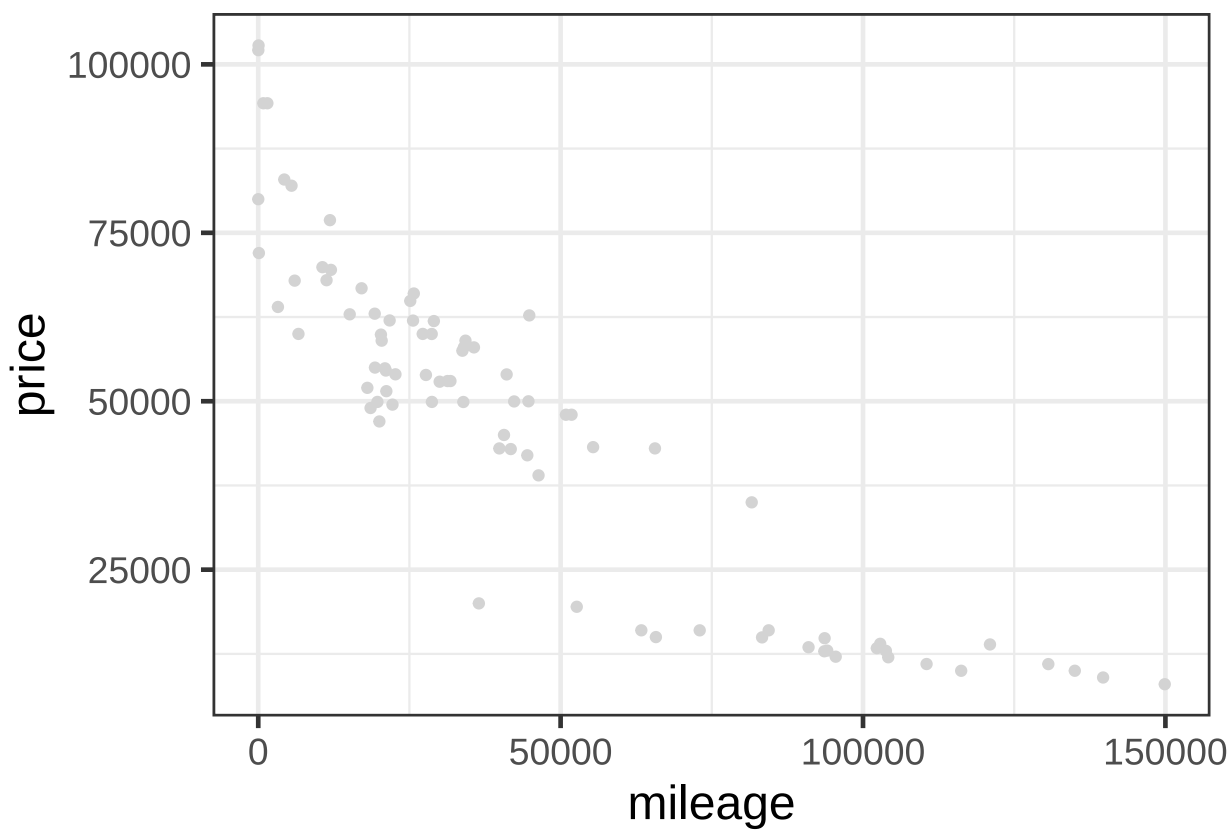
```r
rmse(y_test, ypred_lm2)
```

```
## [1] 9052.279
```

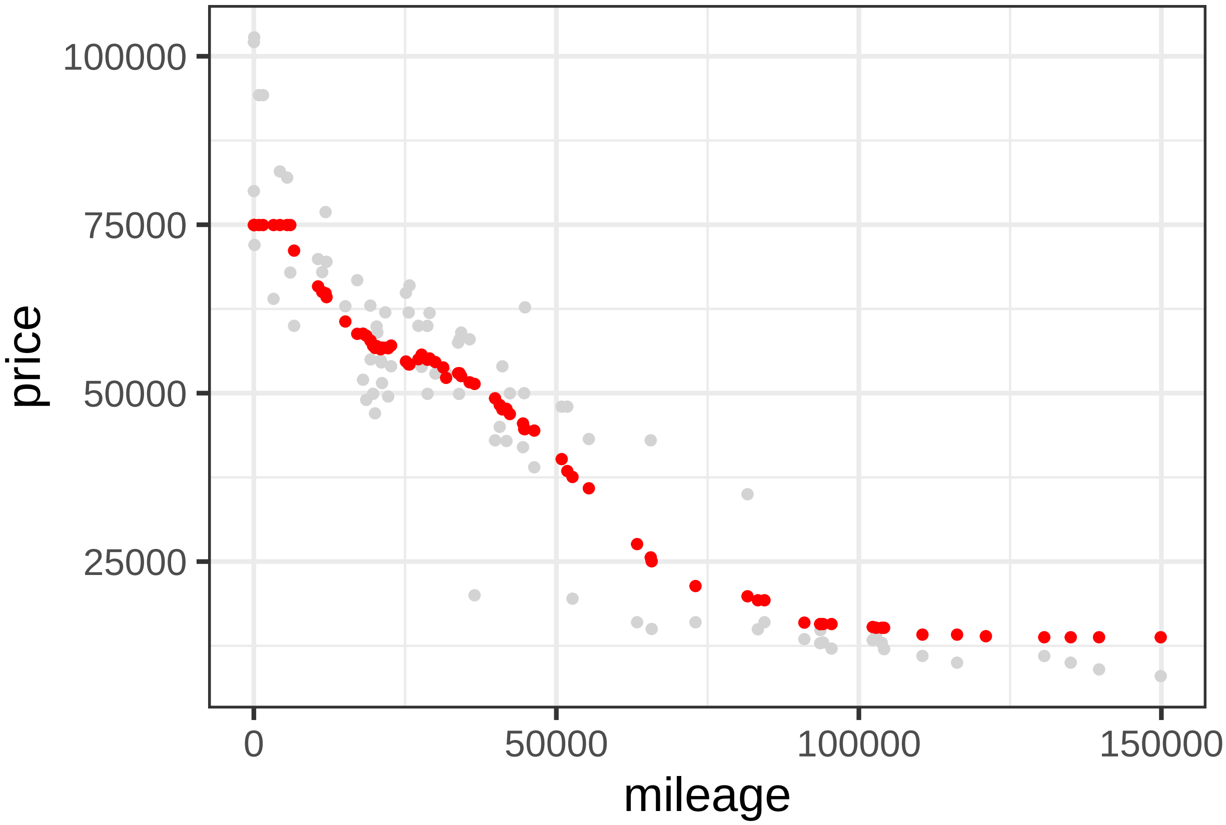
```r
rmse(y_test, ypred_knn50)
```

```
## [1] 9133.789
```

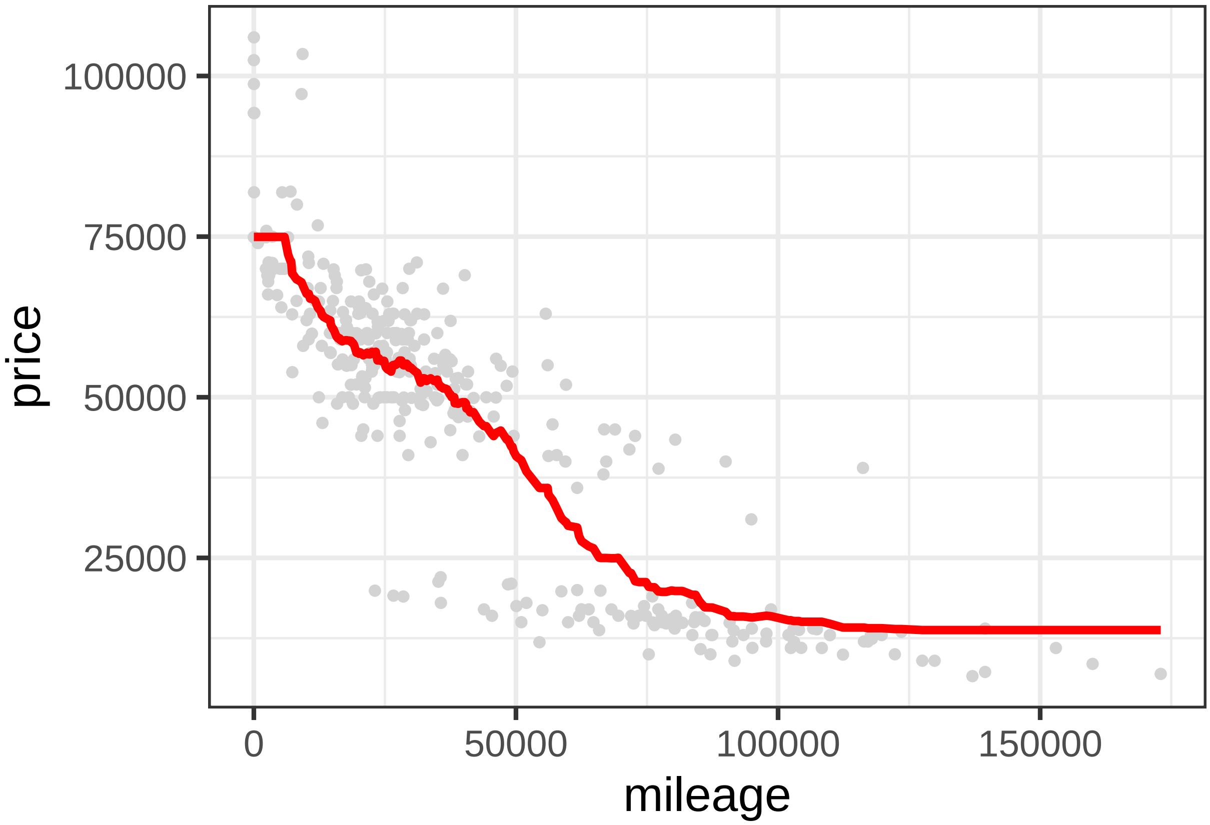
```r
#attach predictions to data frame
D_test$ypred_lm2 = ypred_lm2
D_test$ypred_knn50 = ypred_knn50

p_test = ggplot(data = D_test) +
  geom_point(mapping = aes(x = mileage, y = price), color='lightgrey') +
  theme_bw(base_size=18)
p_test
```




```r
p_test + geom_point(aes(x = mileage, y = ypred_knn50), color='red')
```




```
#KNN variances
knn_model = knn.reg(X_train, X_train, y_train, k = 50)

D_train$ypred = knn_model$pred
p_train = ggplot(data = D_train) +
  geom_point(mapping = aes(x = mileage, y = price), color='lightgrey') +
  theme_bw(base_size=18)
p_train + geom_path(mapping = aes(x=mileage, y=ypred), color='red', size=1.5)
```




```
#K = 70
lm1 = lm(price ~ mileage, data=D_train)
lm2 = lm(price ~ poly(mileage, 2), data=D_train)
knn70 = knn.reg(train = X_train, test = X_test, y = y_train, k=70)


#rmse
rmse = function(y, ypred) {
  sqrt(mean(data.matrix((y-ypred)^2)))
}

ypred_lm1 = predict(lm1, X_test)
ypred_lm2 = predict(lm2, X_test)
ypred_knn70 = knn70$pred

rmse(y_test, ypred_lm1)
```

```
## [1] 10838.68
```

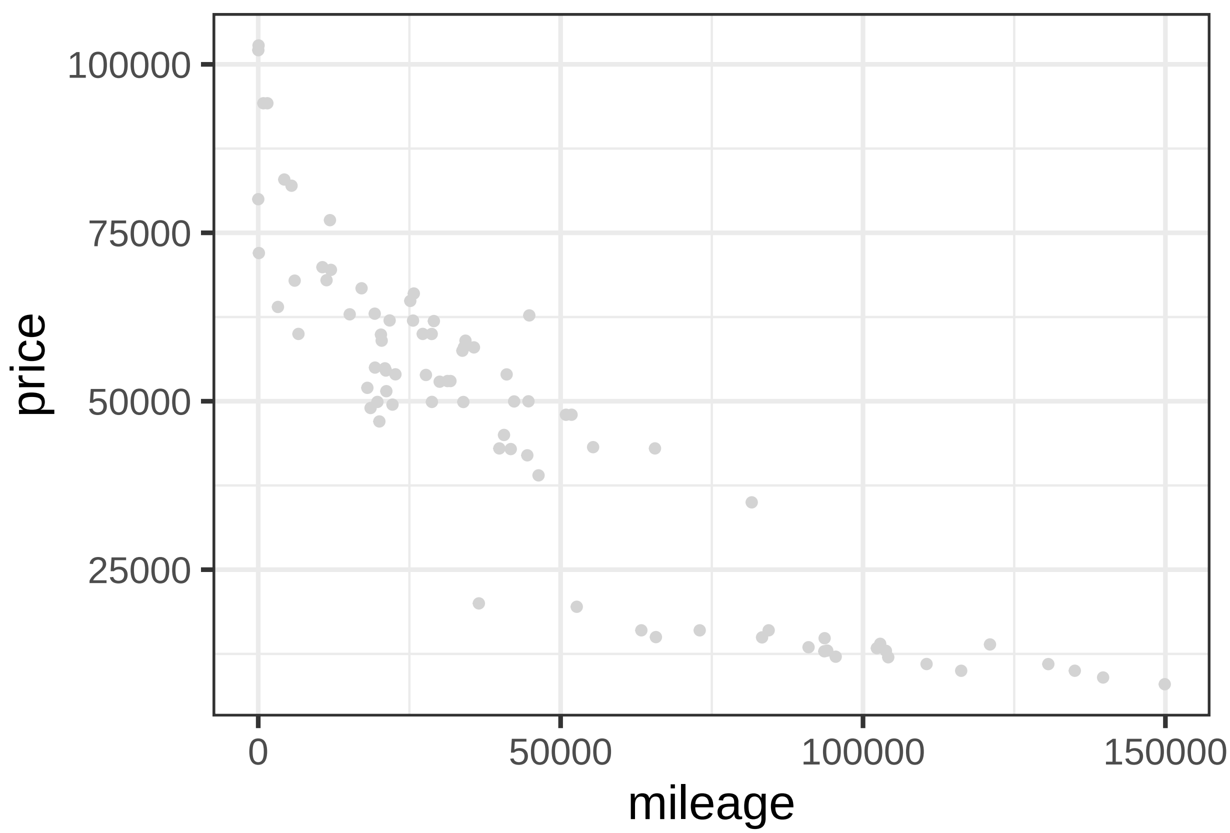
```
rmse(y_test, ypred_lm2)
```

```
## [1] 9052.279
```

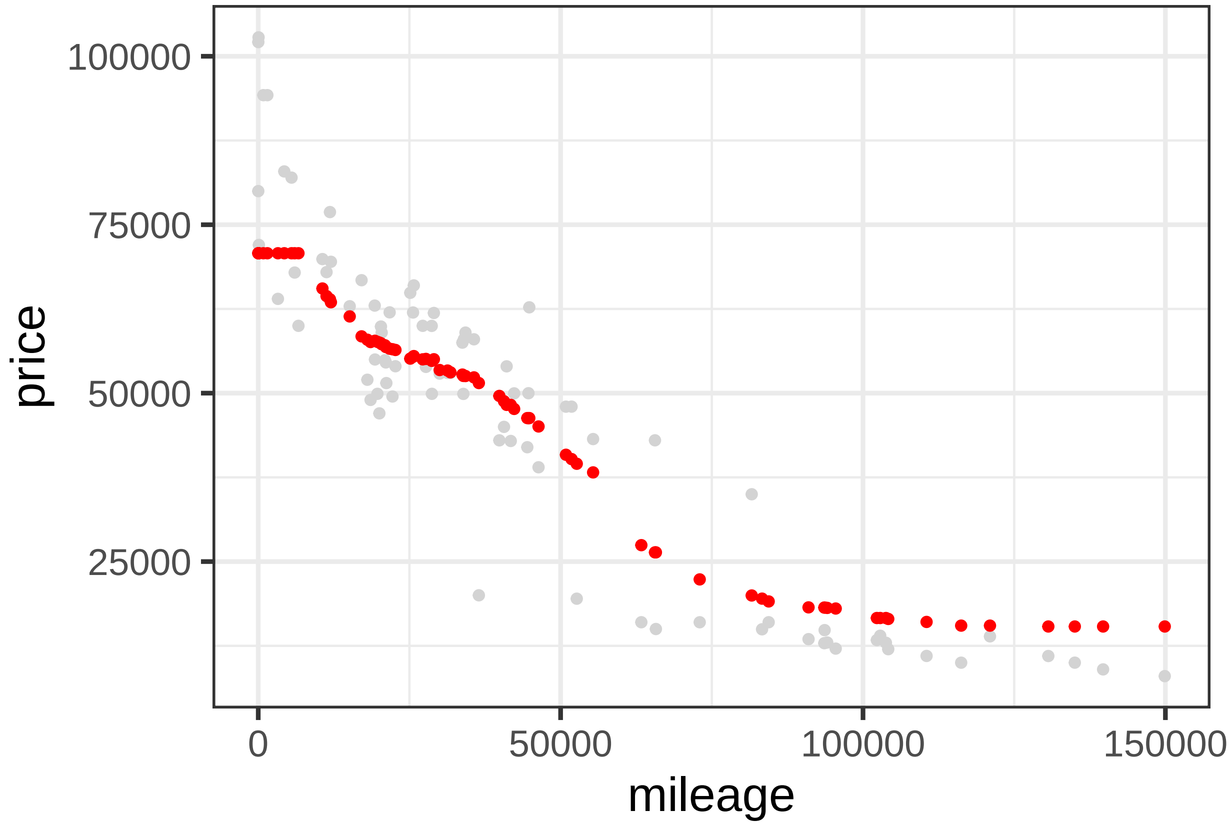
```r
rmse(y_test, ypred_knn70)
```

```
## [1] 9821.277
```

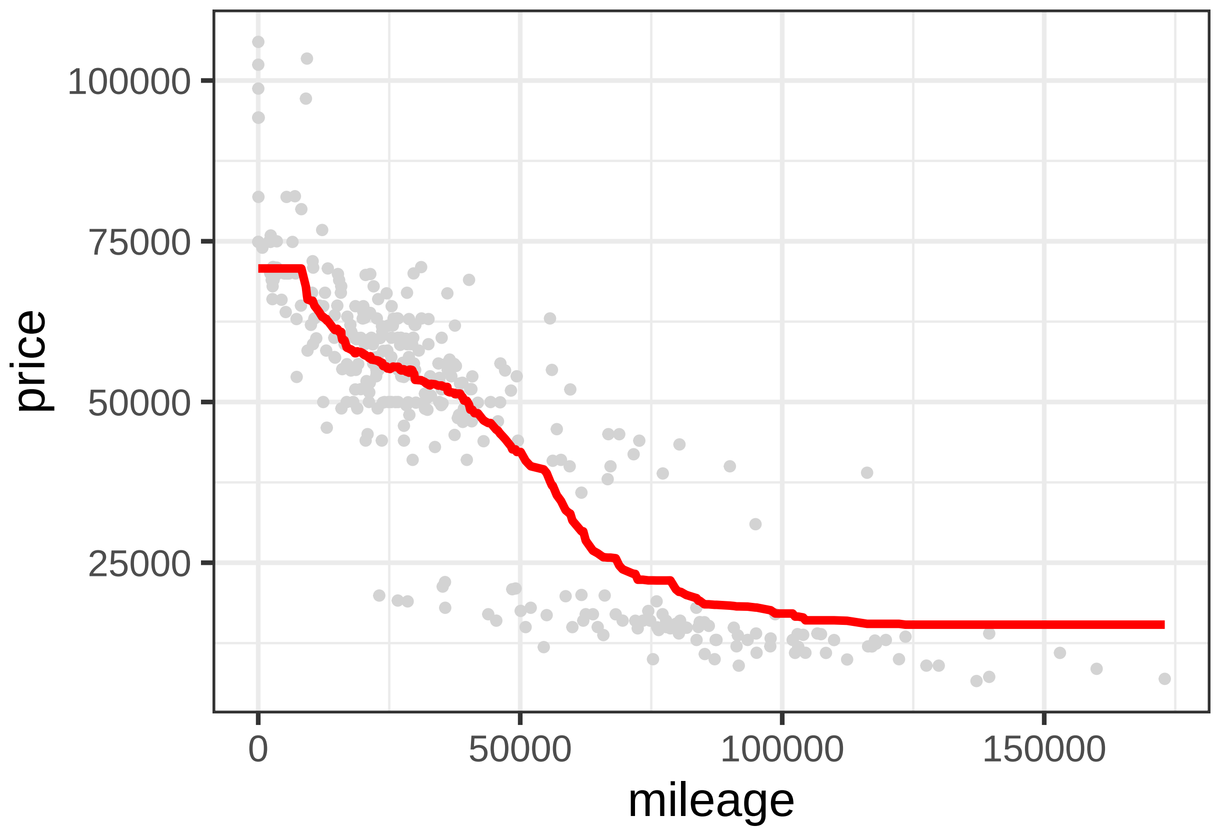
```r
#attach predictions to data frame
D_test$ypred_lm2 = ypred_lm2
D_test$ypred_knn70 = ypred_knn70

p_test = ggplot(data = D_test) +
  geom_point(mapping = aes(x = mileage, y = price), color='lightgrey') +
  theme_bw(base_size=18)
p_test
```




```r
p_test + geom_point(aes(x = mileage, y = ypred_knn70), color='red')
```




```
#KNN variances
knn_model = knn.reg(X_train, X_train, y_train, k = 70)

D_train$ypred = knn_model$pred
p_train = ggplot(data = D_train) +
  geom_point(mapping = aes(x = mileage, y = price), color='lightgrey') +
  theme_bw(base_size=18)
p_train + geom_path(mapping = aes(x=mileage, y=ypred), color='red', size=1.5)
```




```
#K = 80
lm1 = lm(price ~ mileage, data=D_train)
lm2 = lm(price ~ poly(mileage, 2), data=D_train)
knn80 = knn.reg(train = X_train, test = X_test, y = y_train, k=80)


#rmse
rmse = function(y, ypred) {
  sqrt(mean(data.matrix((y-ypred)^2)))
}

ypred_lm1 = predict(lm1, X_test)
ypred_lm2 = predict(lm2, X_test)
ypred_knn80 = knn80$pred

rmse(y_test, ypred_lm1)
```

```
## [1] 10838.68
```

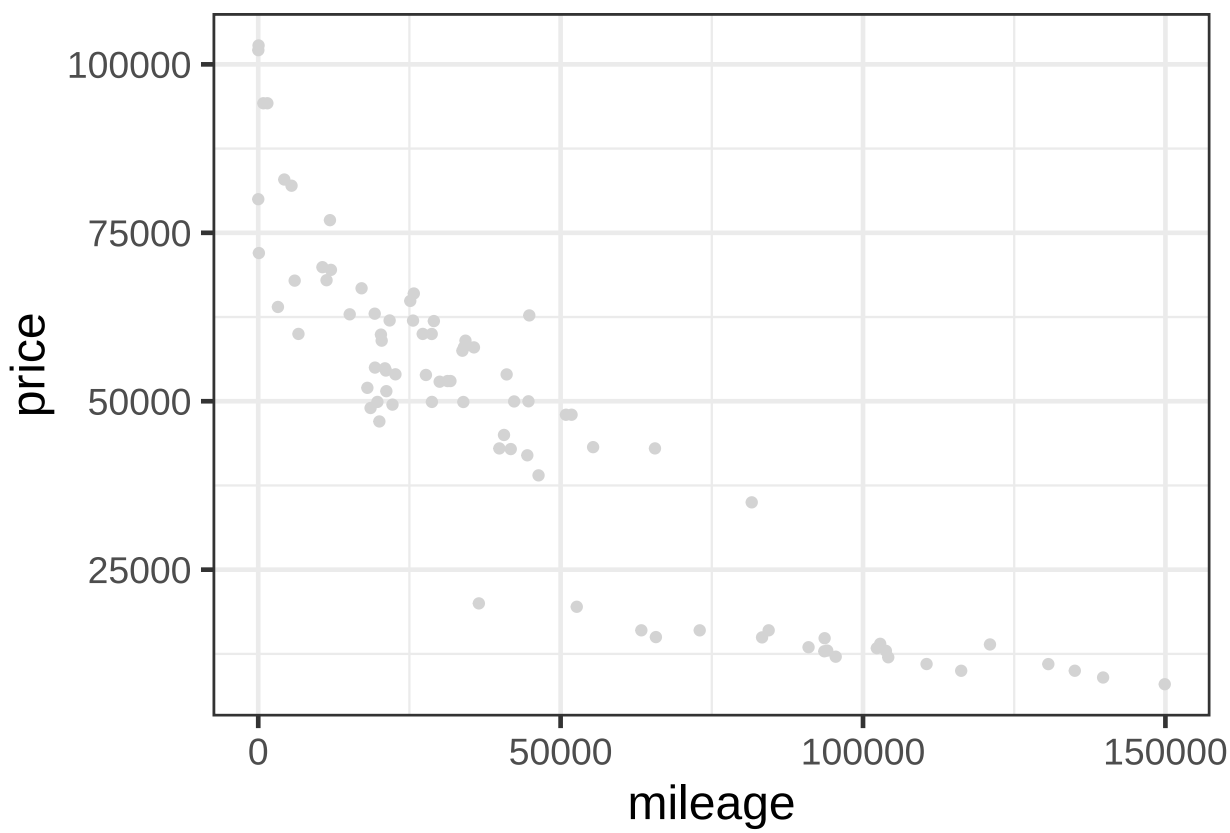
```
rmse(y_test, ypred_lm2)
```

```
## [1] 9052.279
```

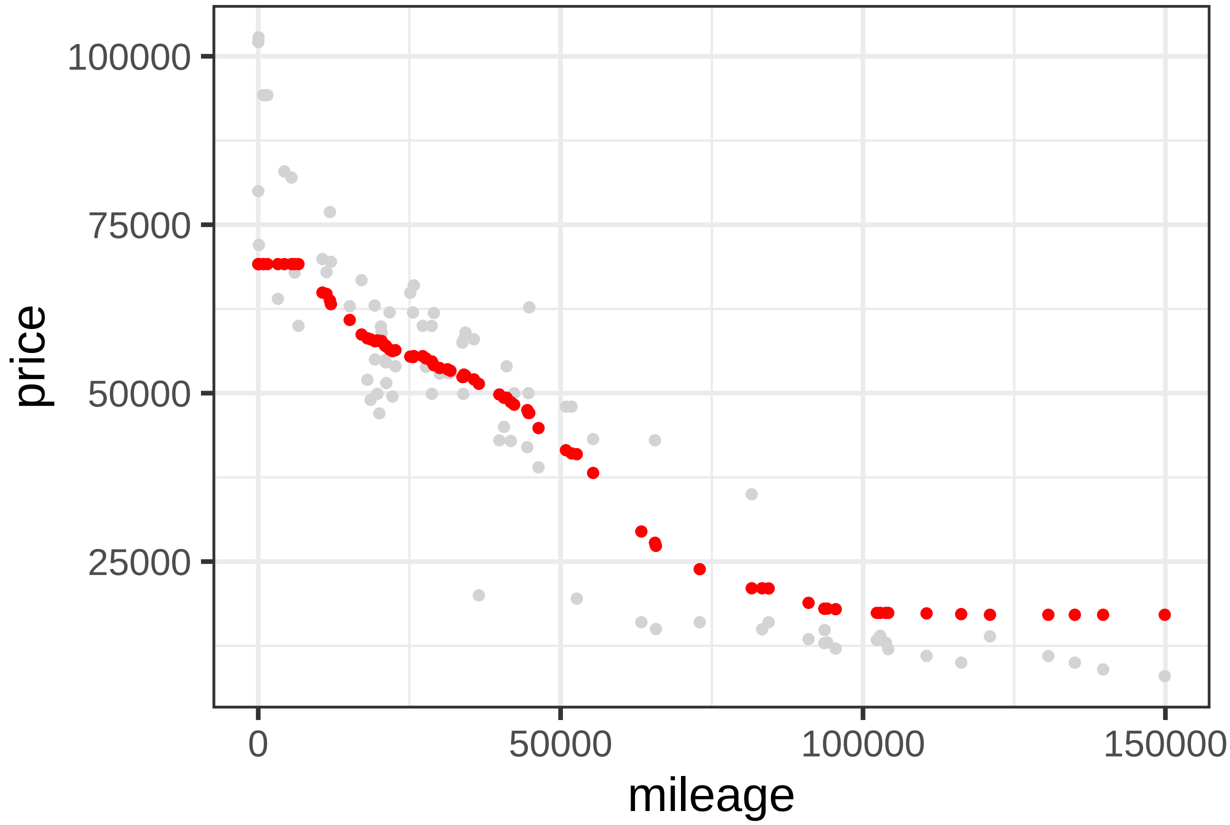
```r
rmse(y_test, ypred_knn80)
```

```
## [1] 10225.39
```

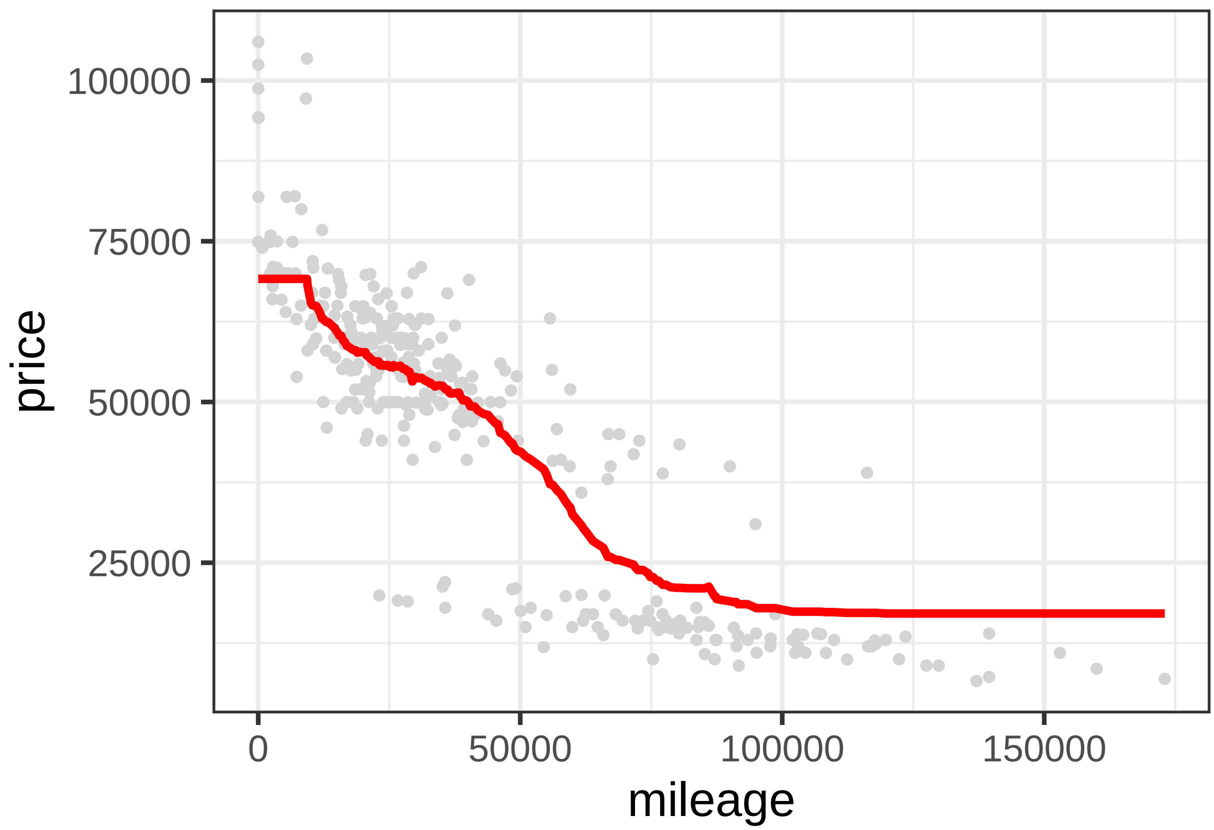
```r
#attach predictions to data frame
D_test$ypred_lm2 = ypred_lm2
D_test$ypred_knn80 = ypred_knn80

p_test = ggplot(data = D_test) +
  geom_point(mapping = aes(x = mileage, y = price), color='lightgrey') +
  theme_bw(base_size=18)
p_test
```




```r
p_test + geom_point(aes(x = mileage, y = ypred_knn80), color='red')
```




```
#KNN variances
knn_model = knn.reg(X_train, X_train, y_train, k = 80)

D_train$ypred = knn_model$pred
p_train = ggplot(data = D_train) +
  geom_point(mapping = aes(x = mileage, y = price), color='lightgrey') +
  theme_bw(base_size=18)
p_train + geom_path(mapping = aes(x=mileage, y=ypred), color='red', size=1.5)
```




```
#K = 90
lm1 = lm(price ~ mileage, data=D_train)
lm2 = lm(price ~ poly(mileage, 2), data=D_train)
knn90 = knn.reg(train = X_train, test = X_test, y = y_train, k=90)


#rmse
rmse = function(y, ypred) {
  sqrt(mean(data.matrix((y-ypred)^2)))
}

ypred_lm1 = predict(lm1, X_test)
ypred_lm2 = predict(lm2, X_test)
ypred_knn90 = knn90$pred

rmse(y_test, ypred_lm1)
```

```
## [1] 10838.68
```

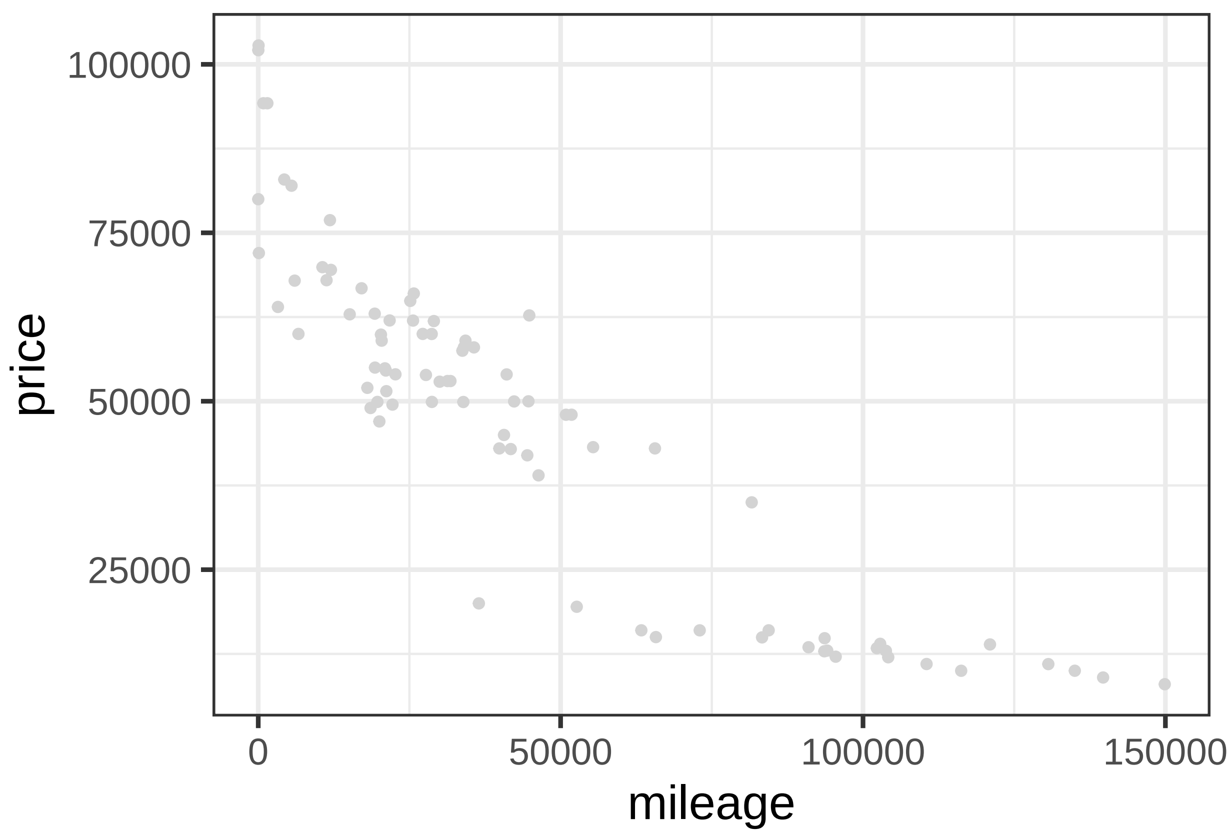
```
rmse(y_test, ypred_lm2)
```

```
## [1] 9052.279
```

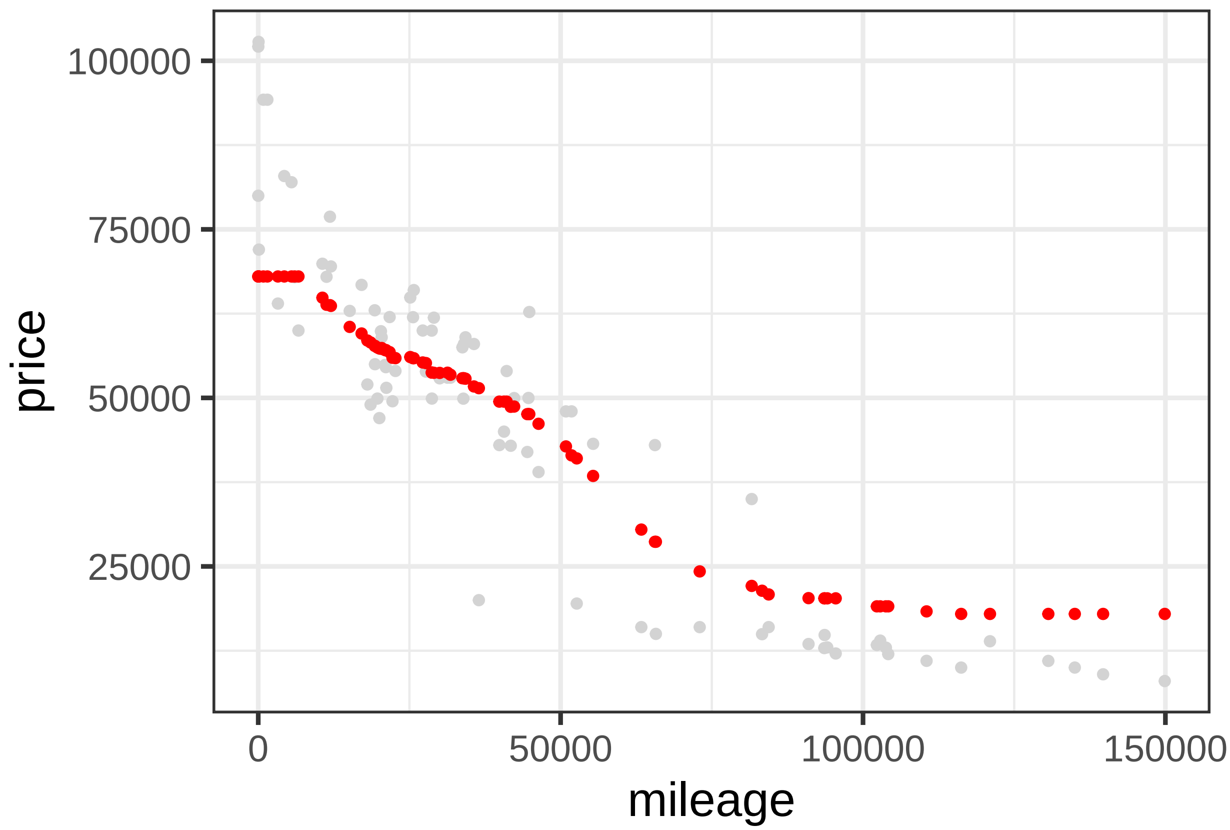
```r
rmse(y_test, ypred_knn90)
```

```
## [1] 10557.06
```

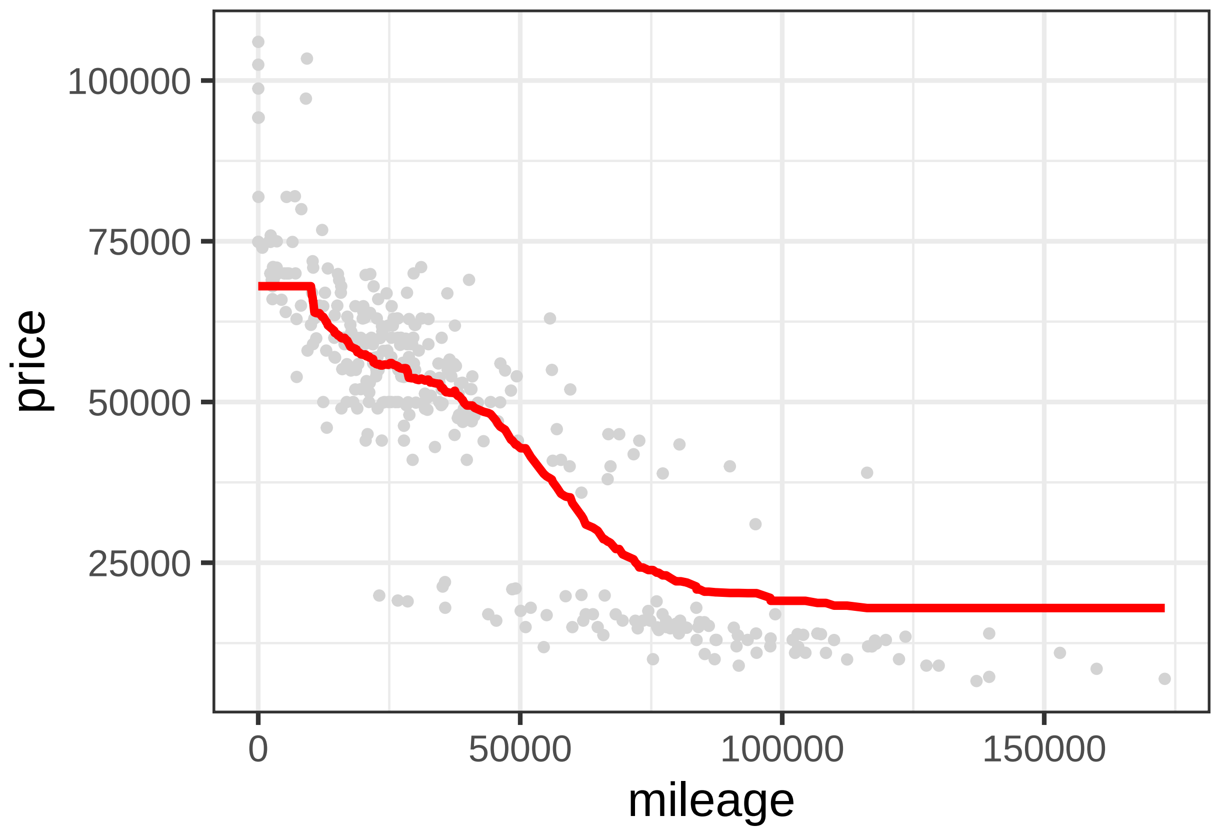
```r
#attach predictions to data frame
D_test$ypred_lm2 = ypred_lm2
D_test$ypred_knn90 = ypred_knn90

p_test = ggplot(data = D_test) +
  geom_point(mapping = aes(x = mileage, y = price), color='lightgrey') +
  theme_bw(base_size=18)
p_test
```




```r
p_test + geom_point(aes(x = mileage, y = ypred_knn90), color='red')
```




```r
#KNN variances
knn_model = knn.reg(X_train, X_train, y_train, k = 90)

D_train$ypred = knn_model$pred
p_train = ggplot(data = D_train) +
  geom_point(mapping = aes(x = mileage, y = price), color='lightgrey') +
  theme_bw(base_size=18)
p_train + geom_path(mapping = aes(x=mileage, y=ypred), color='red', size=1.5)
```




```
#K=100
lm1 = lm(price ~ mileage, data=D_train)
lm2 = lm(price ~ poly(mileage, 2), data=D_train)
knn100 = knn.reg(train = X_train, test = X_test, y = y_train, k=100)


#rmse
rmse = function(y, ypred) {
  sqrt(mean(data.matrix((y-ypred)^2)))
}

ypred_lm1 = predict(lm1, X_test)
ypred_lm2 = predict(lm2, X_test)
ypred_knn100 = knn100$pred

rmse(y_test, ypred_lm1)
```

```
## [1] 10838.68
```

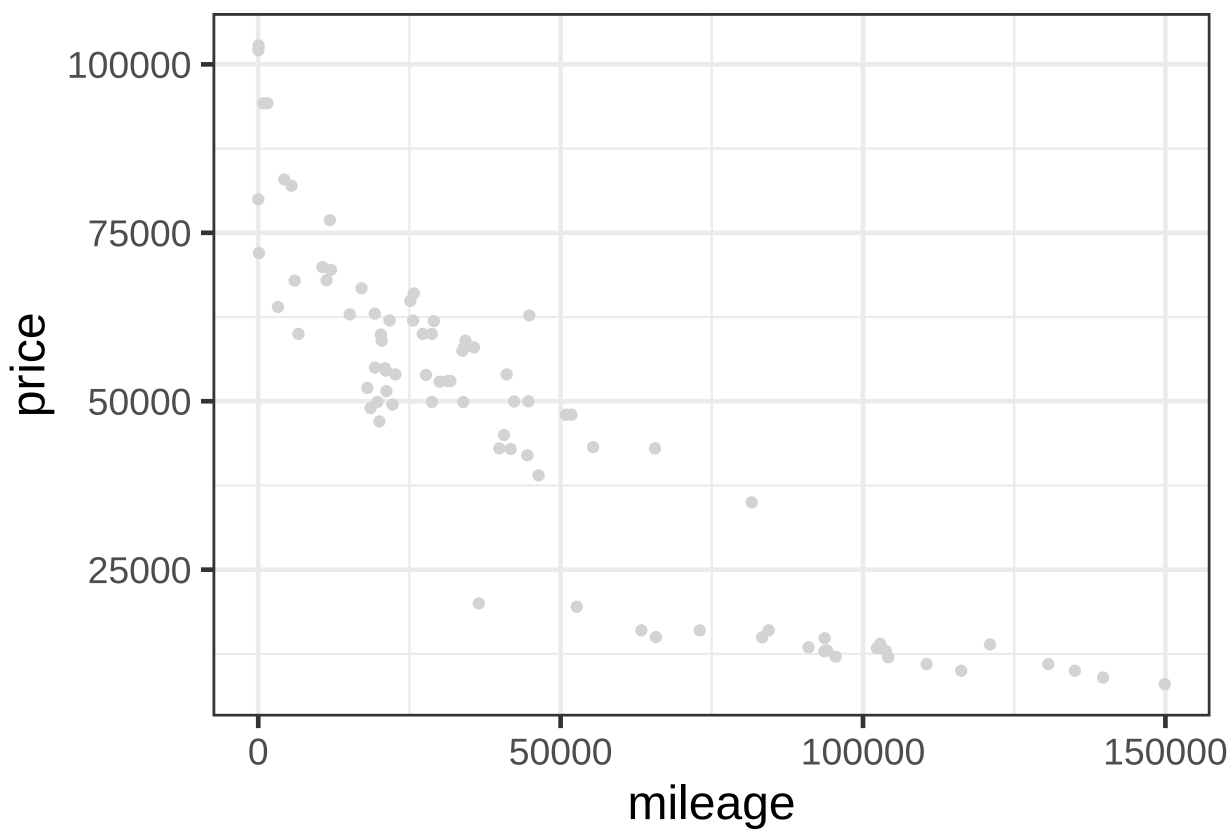
```
rmse(y_test, ypred_lm2)
```

```
## [1] 9052.279
```

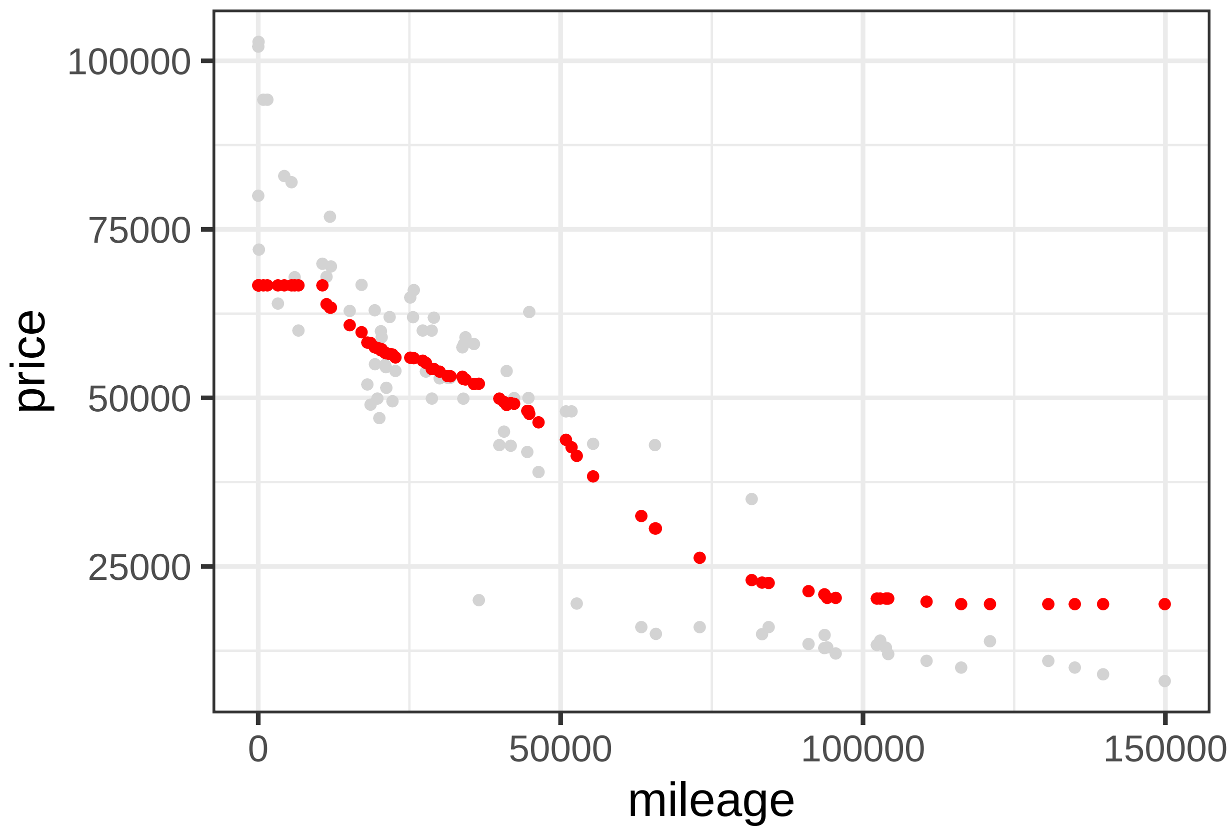
```r
rmse(y_test, ypred_knn100)
```

```
## [1] 11019.58
```

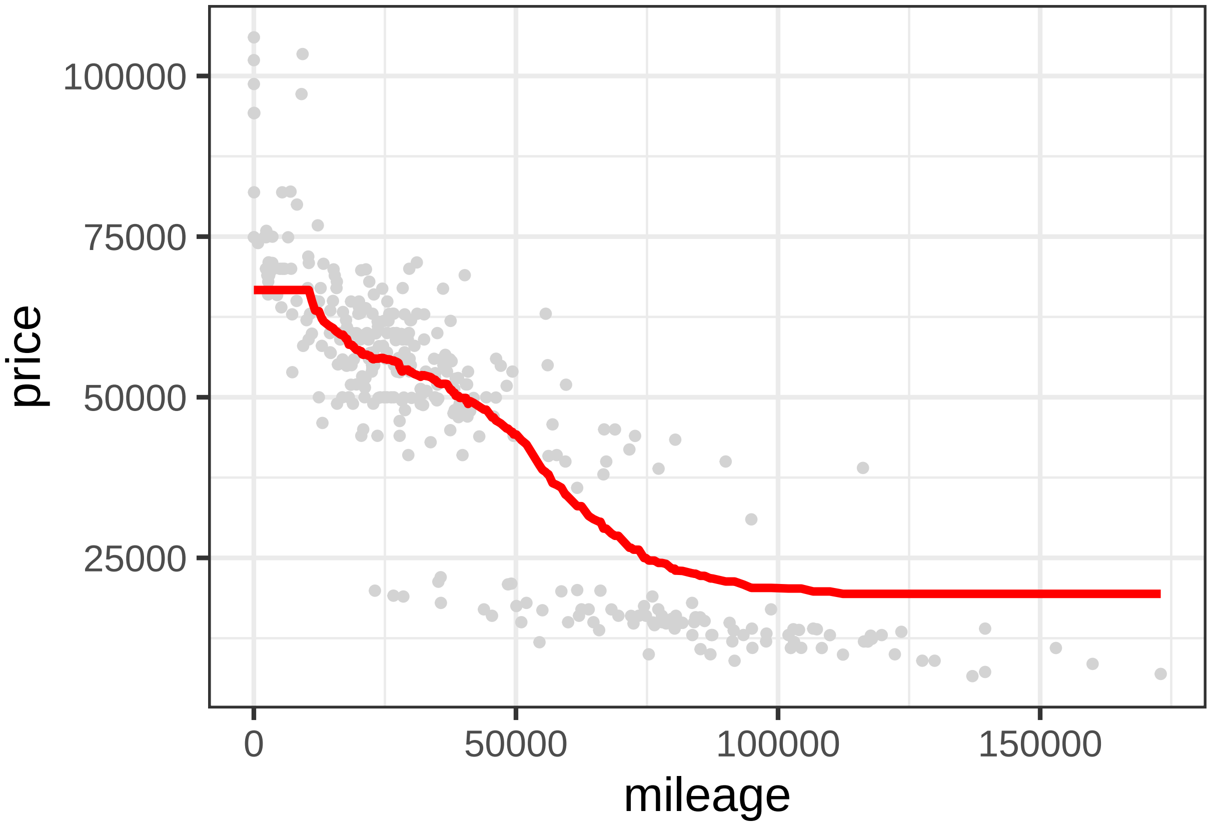
```r
#attach predictions to data frame
D_test$ypred_lm2 = ypred_lm2
D_test$ypred_knn100 = ypred_knn100

p_test = ggplot(data = D_test) +
  geom_point(mapping = aes(x = mileage, y = price), color='lightgrey') +
  theme_bw(base_size=18)
p_test
```




```r
p_test + geom_point(aes(x = mileage, y = ypred_knn100), color='red')
```




```
#KNN variances
knn_model = knn.reg(X_train, X_train, y_train, k = 100)

D_train$ypred = knn_model$pred
p_train = ggplot(data = D_train) +
  geom_point(mapping = aes(x = mileage, y = price), color='lightgrey') +
  theme_bw(base_size=18)
p_train + geom_path(mapping = aes(x=mileage, y=ypred), color='red', size=1.5)
```




```
#K = 120
lm1 = lm(price ~ mileage, data=D_train)
lm2 = lm(price ~ poly(mileage, 2), data=D_train)
knn120 = knn.reg(train = X_train, test = X_test, y = y_train, k=120)


#rmse
rmse = function(y, ypred) {
  sqrt(mean(data.matrix((y-ypred)^2)))
}

ypred_lm1 = predict(lm1, X_test)
ypred_lm2 = predict(lm2, X_test)
ypred_knn120 = knn120$pred

rmse(y_test, ypred_lm1)
```

```
## [1] 10838.68
```

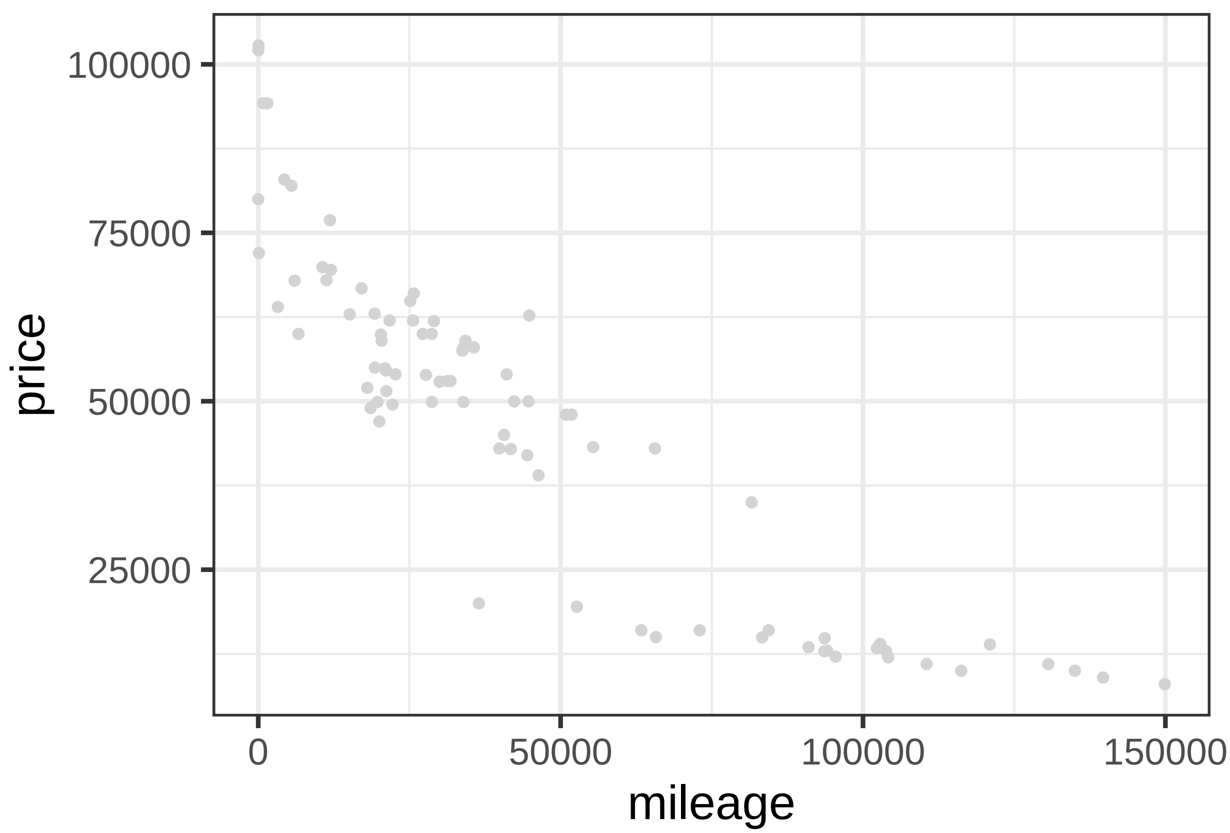
```
rmse(y_test, ypred_lm2)
```

```
## [1] 9052.279
```

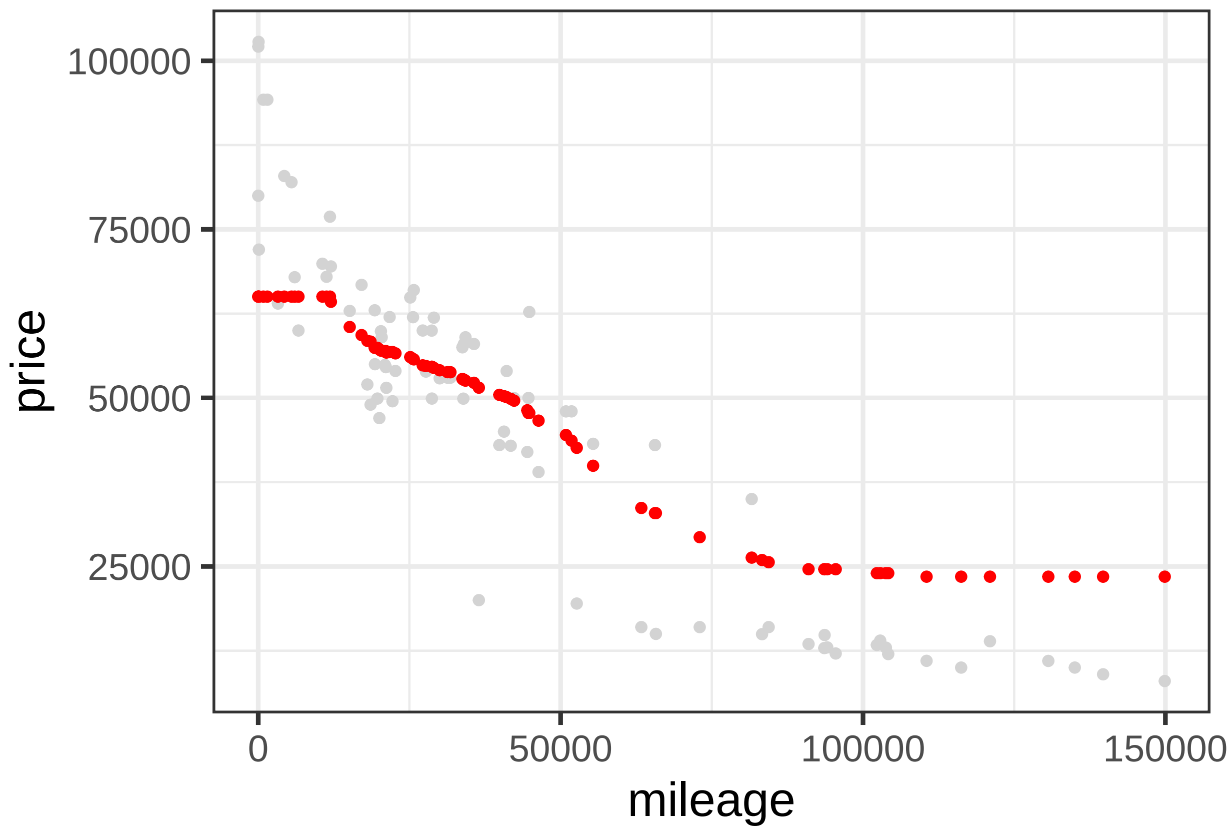
```r
rmse(y_test, ypred_knn120)
```

```
## [1] 12064.36
```

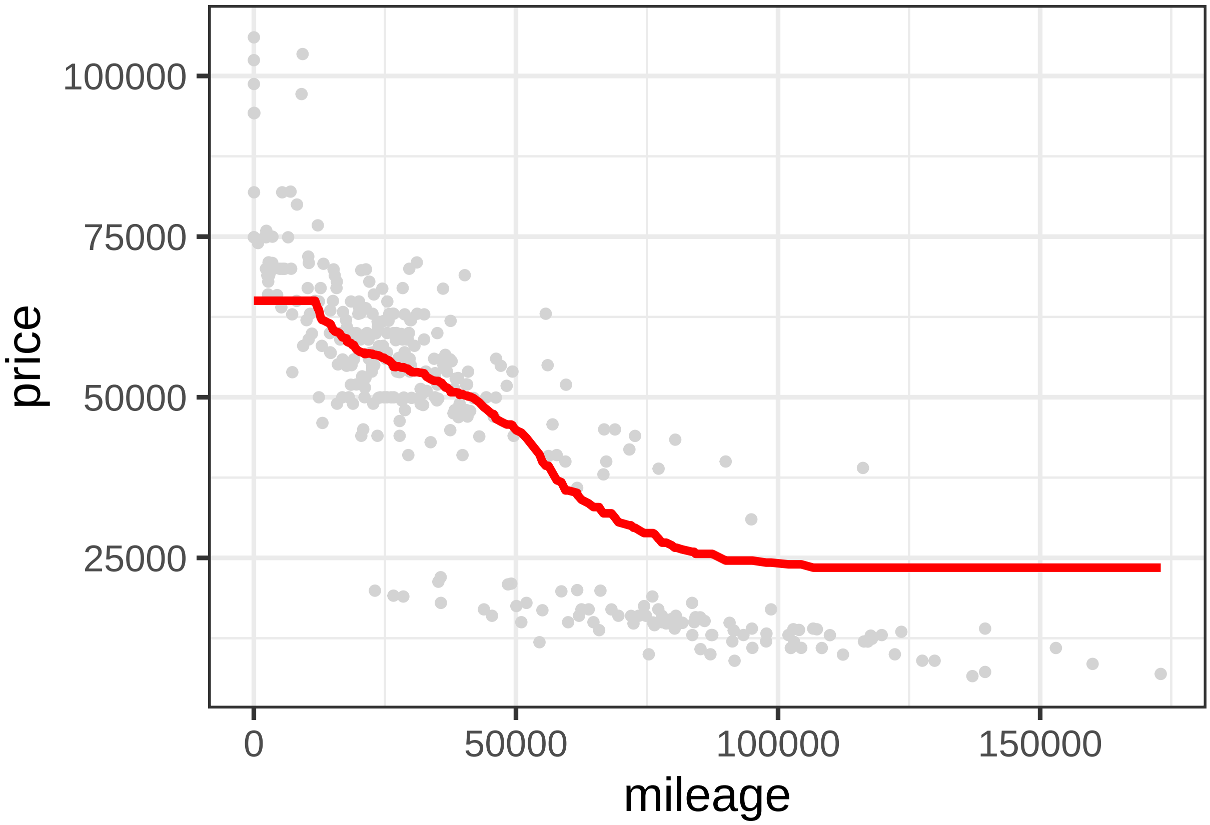
```r
#attach predictions to data frame
D_test$ypred_lm2 = ypred_lm2
D_test$ypred_knn120 = ypred_knn120

p_test = ggplot(data = D_test) +
  geom_point(mapping = aes(x = mileage, y = price), color='lightgrey') +
  theme_bw(base_size=18)
p_test
```




```r
p_test + geom_point(aes(x = mileage, y = ypred_knn120), color='red')
```




```r
#KNN variances
knn_model = knn.reg(X_train, X_train, y_train, k = 120)

D_train$ypred = knn_model$pred
p_train = ggplot(data = D_train) +
  geom_point(mapping = aes(x = mileage, y = price), color='lightgrey') +
  theme_bw(base_size=18)
p_train + geom_path(mapping = aes(x=mileage, y=ypred), color='red', size=1.5)
```




```
lm1 = lm(price ~ mileage, data=D_train)
lm2 = lm(price ~ poly(mileage, 2), data=D_train)
knn140 = knn.reg(train = X_train, test = X_test, y = y_train, k=140)


#rmse
rmse = function(y, ypred) {
  sqrt(mean(data.matrix((y-ypred)^2)))
}

ypred_lm1 = predict(lm1, X_test)
ypred_lm2 = predict(lm2, X_test)
ypred_knn140 = knn140$pred

rmse(y_test, ypred_lm1)
```

```
## [1] 10838.68
```

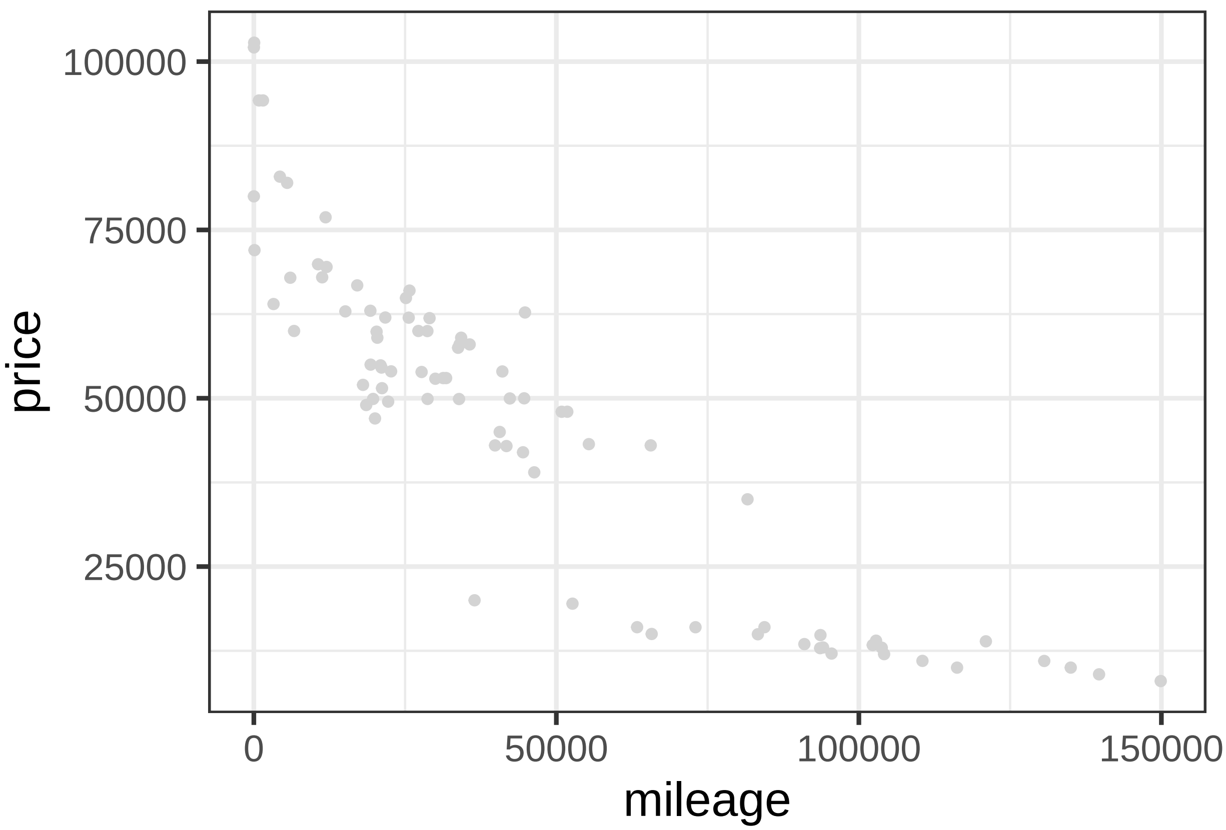
```
rmse(y_test, ypred_lm2)
```

```
## [1] 9052.279
```

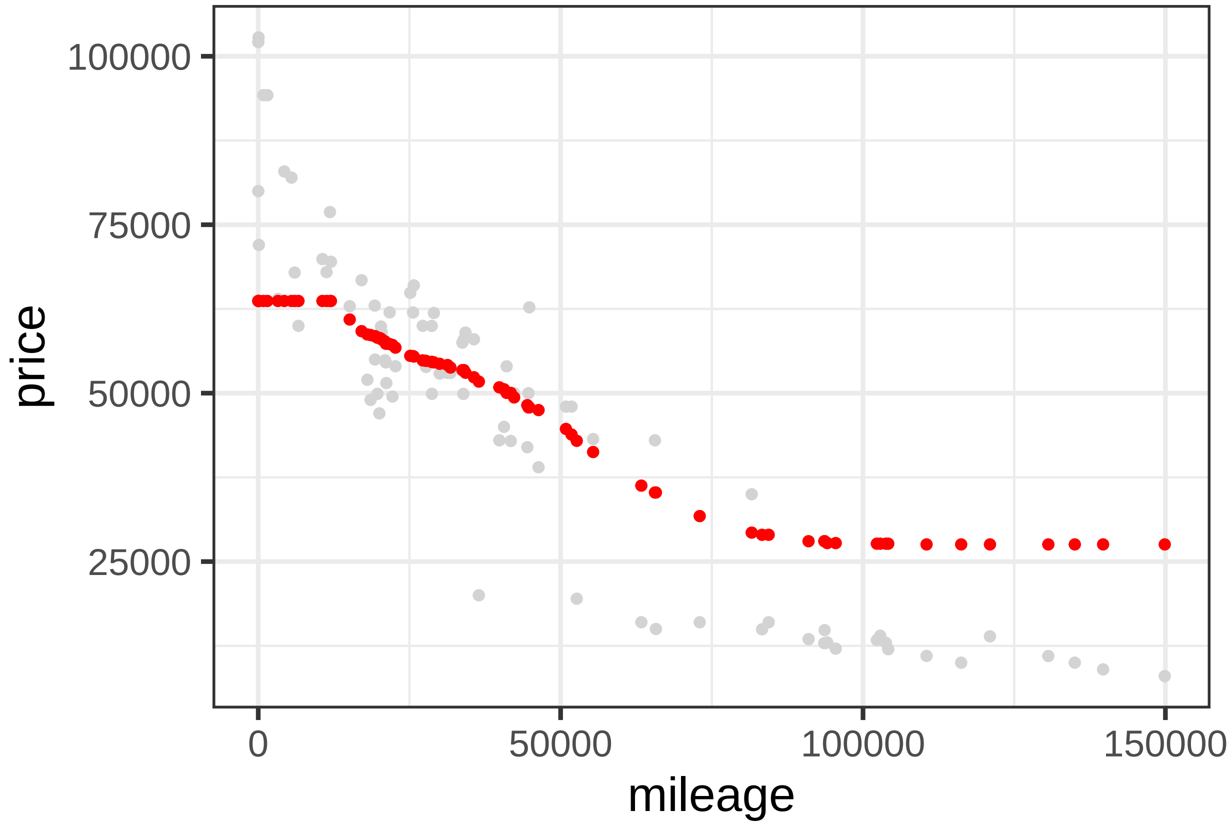
```
rmse(y_test, ypred_knn140)
```

```
## [1] 13314.88
```

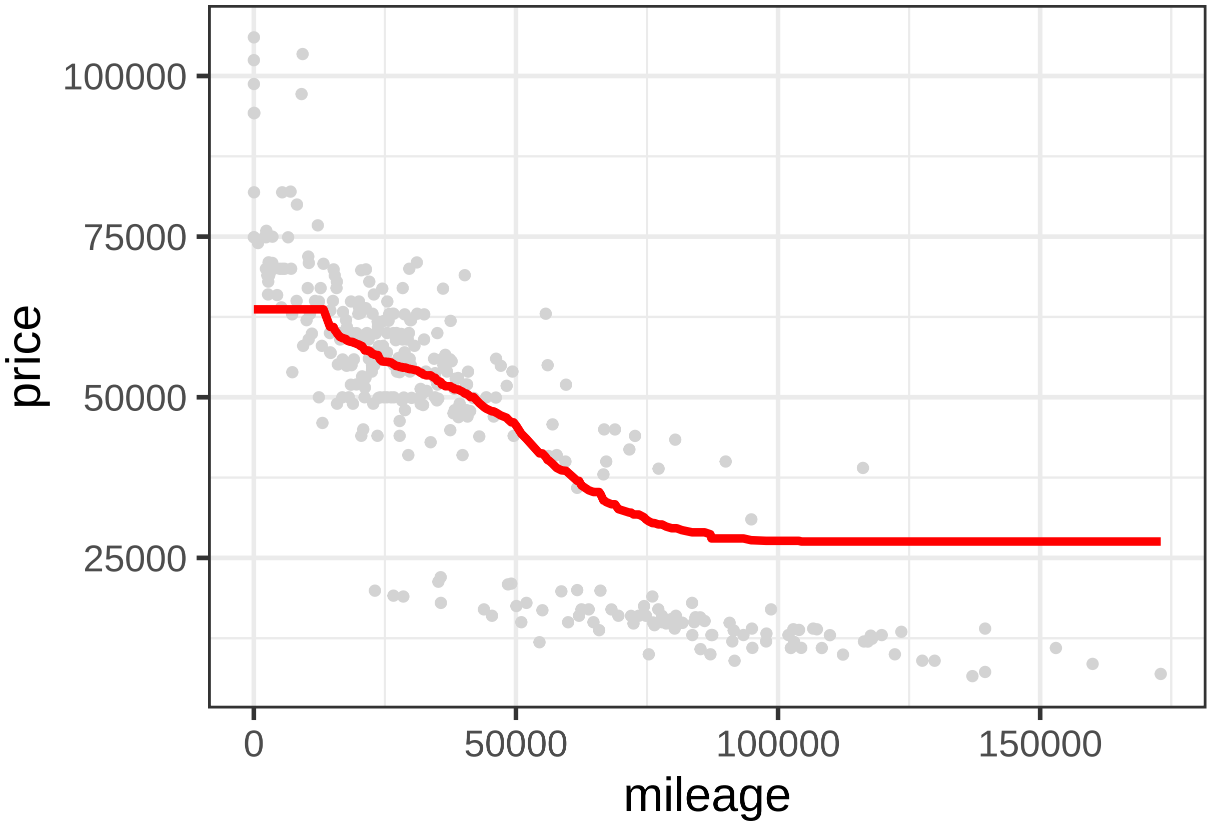
```r
#attach predictions to data frame
D_test$ypred_lm2 = ypred_lm2
D_test$ypred_knn140 = ypred_knn140

p_test = ggplot(data = D_test) +
  geom_point(mapping = aes(x = mileage, y = price), color='lightgrey') +
  theme_bw(base_size=18)
p_test
```




```r
p_test + geom_point(aes(x = mileage, y = ypred_knn140), color='red')
```




```
#KNN variances
knn_model = knn.reg(X_train, X_train, y_train, k = 140)

D_train$ypred = knn_model$pred
p_train = ggplot(data = D_train) +
  geom_point(mapping = aes(x = mileage, y = price), color='lightgrey') +
  theme_bw(base_size=18)
p_train + geom_path(mapping = aes(x=mileage, y=ypred), color='red', size=1.5)
```




```
lm1 = lm(price ~ mileage, data=D_train)
lm2 = lm(price ~ poly(mileage, 2), data=D_train)
knn150 = knn.reg(train = X_train, test = X_test, y = y_train, k=150)


#rmse
rmse = function(y, ypred) {
  sqrt(mean(data.matrix((y-ypred)^2)))
}

ypred_lm1 = predict(lm1, X_test)
ypred_lm2 = predict(lm2, X_test)
ypred_knn150 = knn150$pred

rmse(y_test, ypred_lm1)
```

```
## [1] 10838.68
```

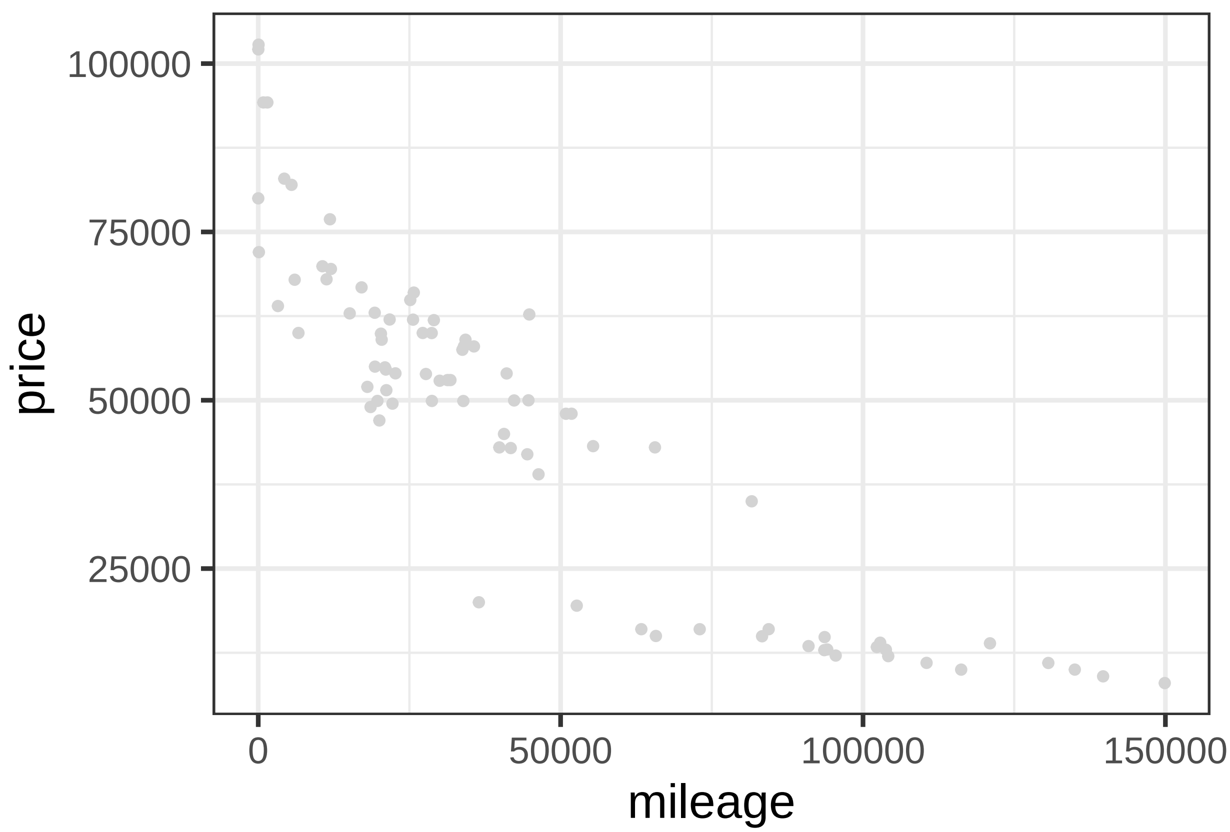
```
rmse(y_test, ypred_lm2)
```

```
## [1] 9052.279
```

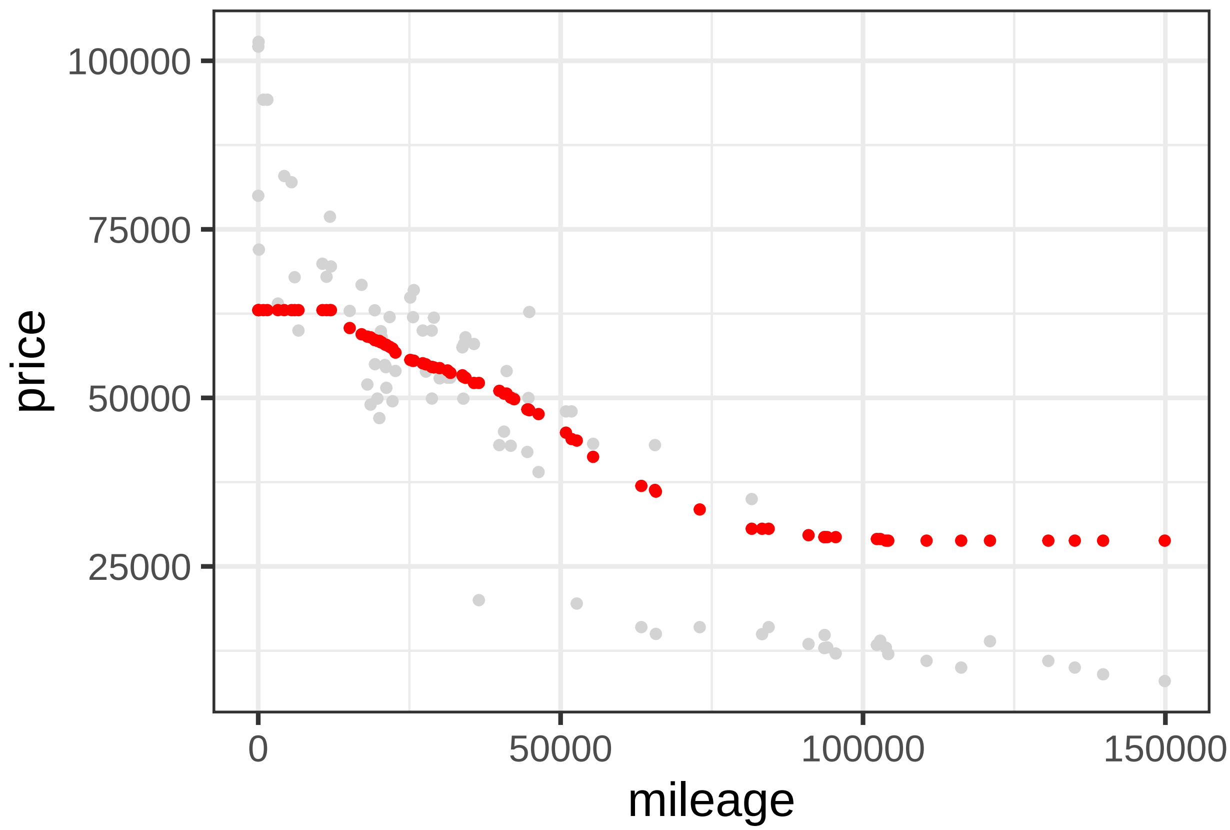
```
rmse(y_test, ypred_knn150)
```

```
## [1] 13878.61
```

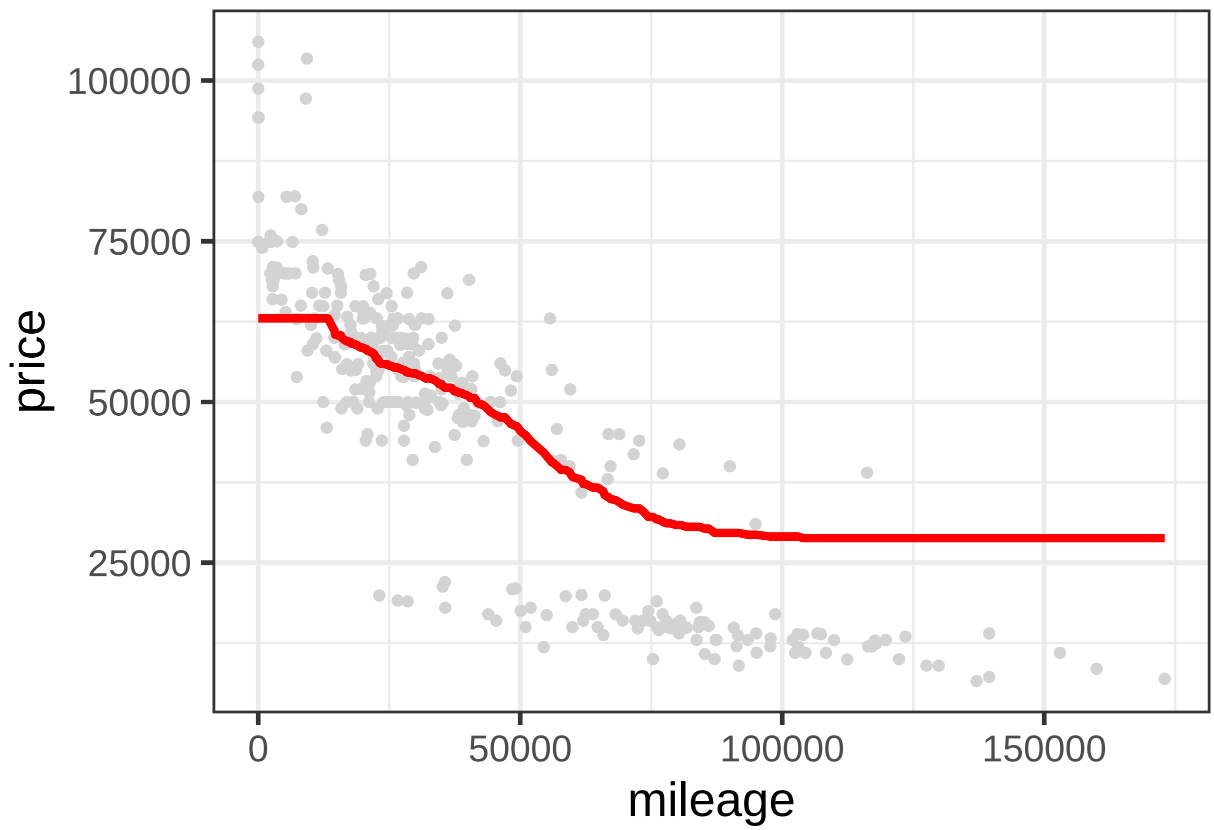
```r
#attach predictions to data frame
D_test$ypred_lm2 = ypred_lm2
D_test$ypred_knn150 = ypred_knn150

p_test = ggplot(data = D_test) +
  geom_point(mapping = aes(x = mileage, y = price), color='lightgrey') +
  theme_bw(base_size=18)
p_test
```




```r
p_test + geom_point(aes(x = mileage, y = ypred_knn150), color='red')
```




```
#KNN variances
knn_model = knn.reg(X_train, X_train, y_train, k = 150)

D_train$ypred = knn_model$pred
p_train = ggplot(data = D_train) +
  geom_point(mapping = aes(x = mileage, y = price), color='lightgrey') +
  theme_bw(base_size=18)
p_train + geom_path(mapping = aes(x=mileage, y=ypred), color='red', size=1.5)
```




```
#K=175
lm1 = lm(price ~ mileage, data=D_train)
lm2 = lm(price ~ poly(mileage, 2), data=D_train)
knn175 = knn.reg(train = X_train, test = X_test, y = y_train, k=175)


#rmse
rmse = function(y, ypred) {
  sqrt(mean(data.matrix((y-ypred)^2)))
}

ypred_lm1 = predict(lm1, X_test)
ypred_lm2 = predict(lm2, X_test)
ypred_knn175 = knn175$pred

rmse(y_test, ypred_lm1)
```

```
## [1] 10838.68
```

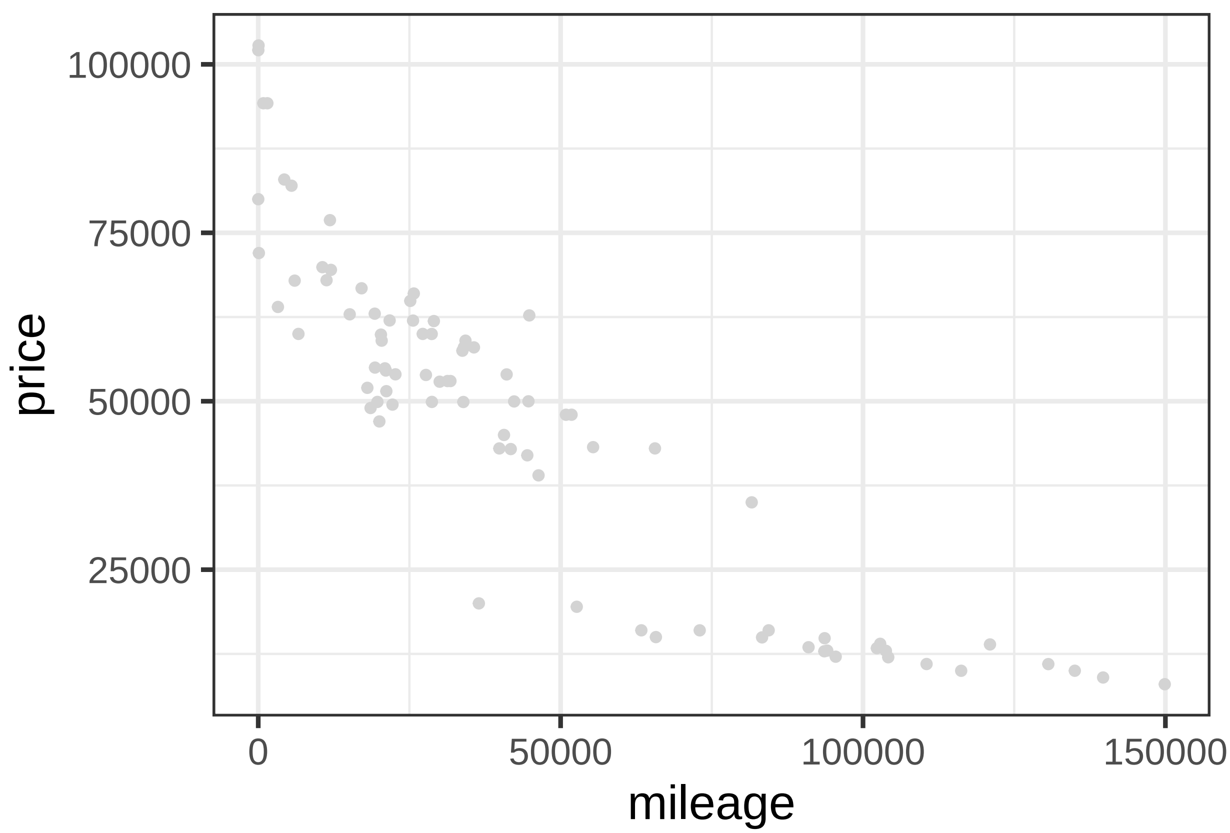
```
rmse(y_test, ypred_lm2)
```

```
## [1] 9052.279
```

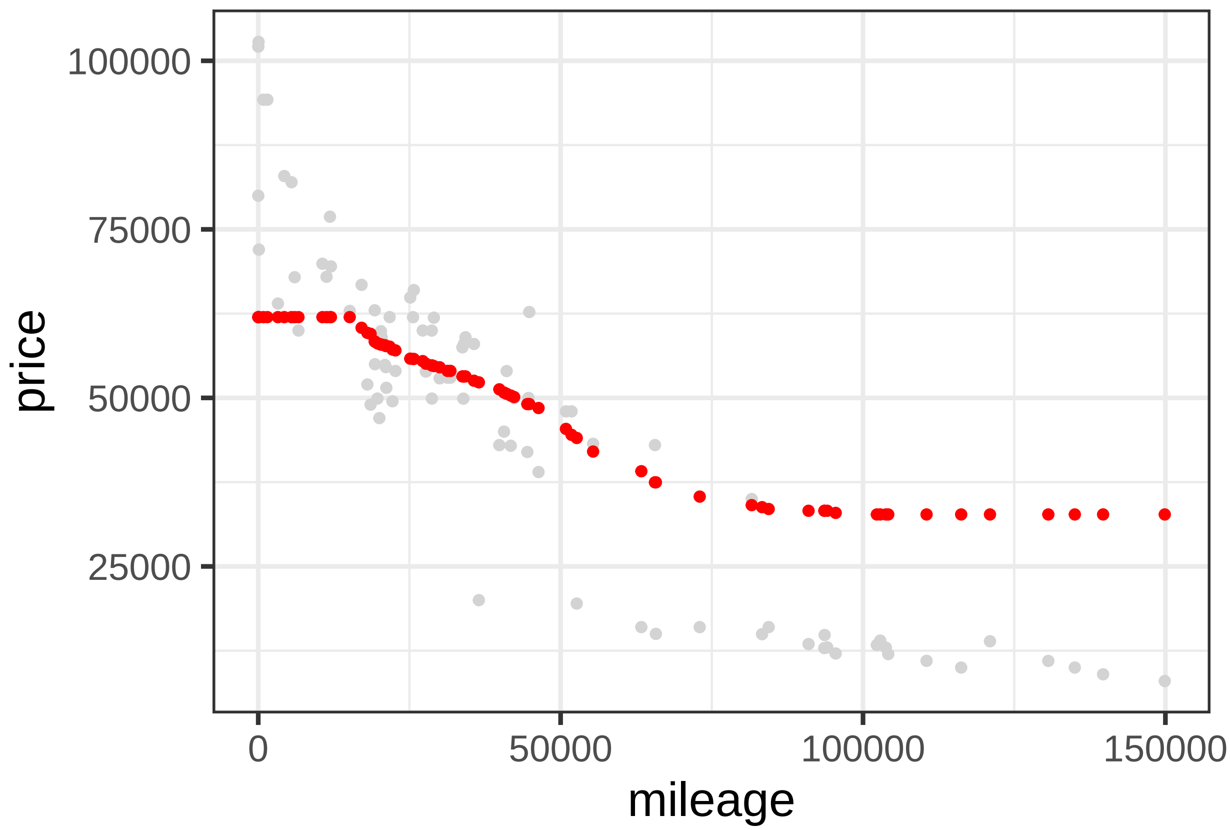
```
rmse(y_test, ypred_knn175)
```

```
## [1] 15197.6
```

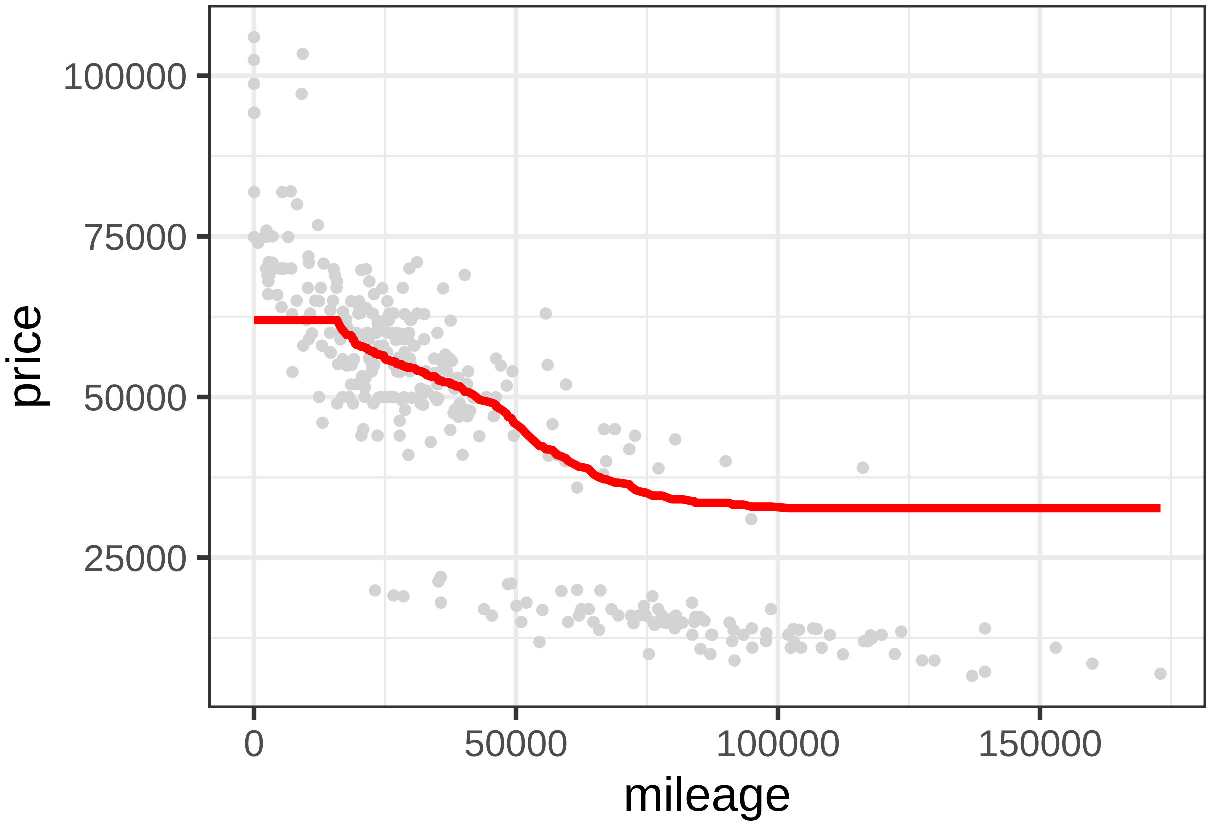
```
#attach predictions to data frame
D_test$ypred_lm2 = ypred_lm2
D_test$ypred_knn175 = ypred_knn175

p_test = ggplot(data = D_test) +
  geom_point(mapping = aes(x = mileage, y = price), color='lightgrey') +
  theme_bw(base_size=18)
p_test
```




```
p_test + geom_point(aes(x = mileage, y = ypred_knn175), color='red')
```




```
#KNN variances
knn_model = knn.reg(X_train, X_train, y_train, k = 175)

D_train$ypred = knn_model$pred
p_train = ggplot(data = D_train) +
  geom_point(mapping = aes(x = mileage, y = price), color='lightgrey') +
  theme_bw(base_size=18)
p_train + geom_path(mapping = aes(x=mileage, y=ypred), color='red', size=1.5)
```




```
#K=190
lm1 = lm(price ~ mileage, data=D_train)
lm2 = lm(price ~ poly(mileage, 2), data=D_train)
knn190 = knn.reg(train = X_train, test = X_test, y = y_train, k=190)


#rmse
rmse = function(y, ypred) {
  sqrt(mean(data.matrix((y-ypred)^2)))
}

ypred_lm1 = predict(lm1, X_test)
ypred_lm2 = predict(lm2, X_test)
ypred_knn190 = knn190$pred

rmse(y_test, ypred_lm1)
```

```
## [1] 10838.68
```

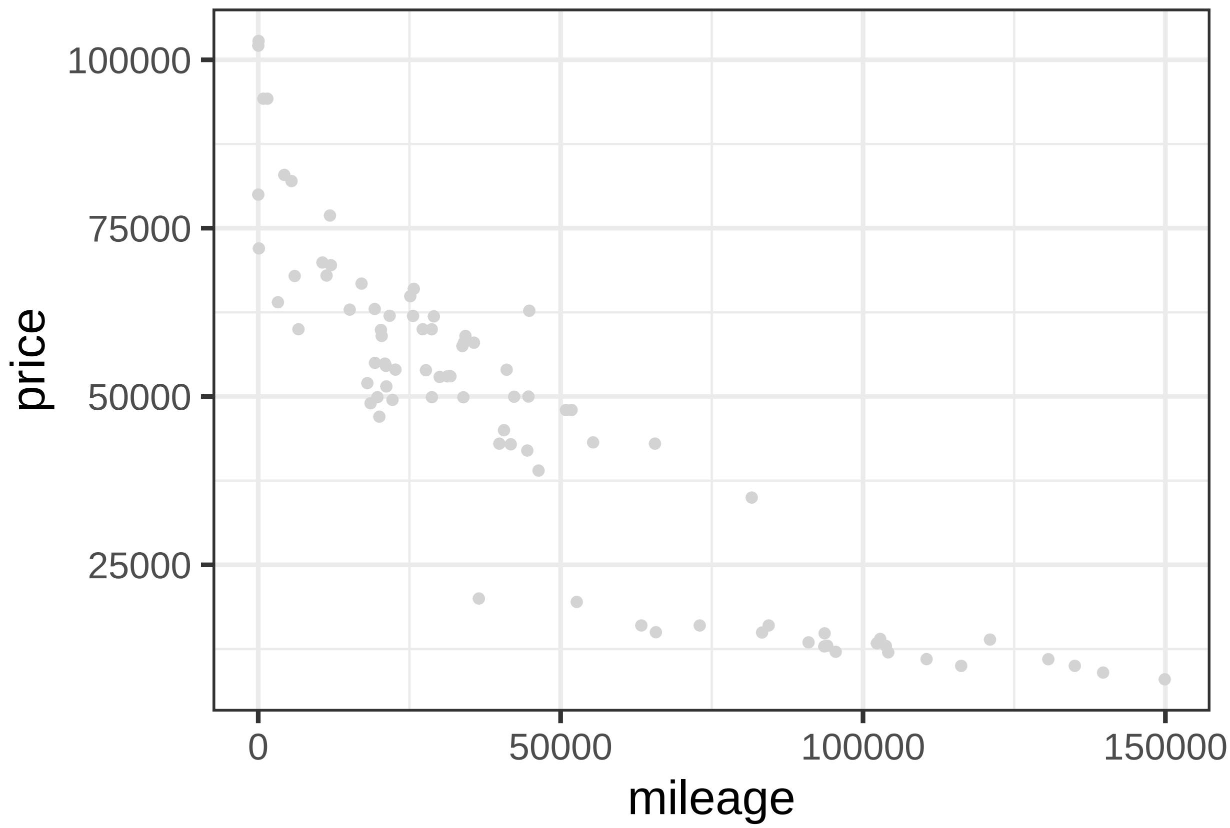
```
rmse(y_test, ypred_lm2)
```

```
## [1] 9052.279
```

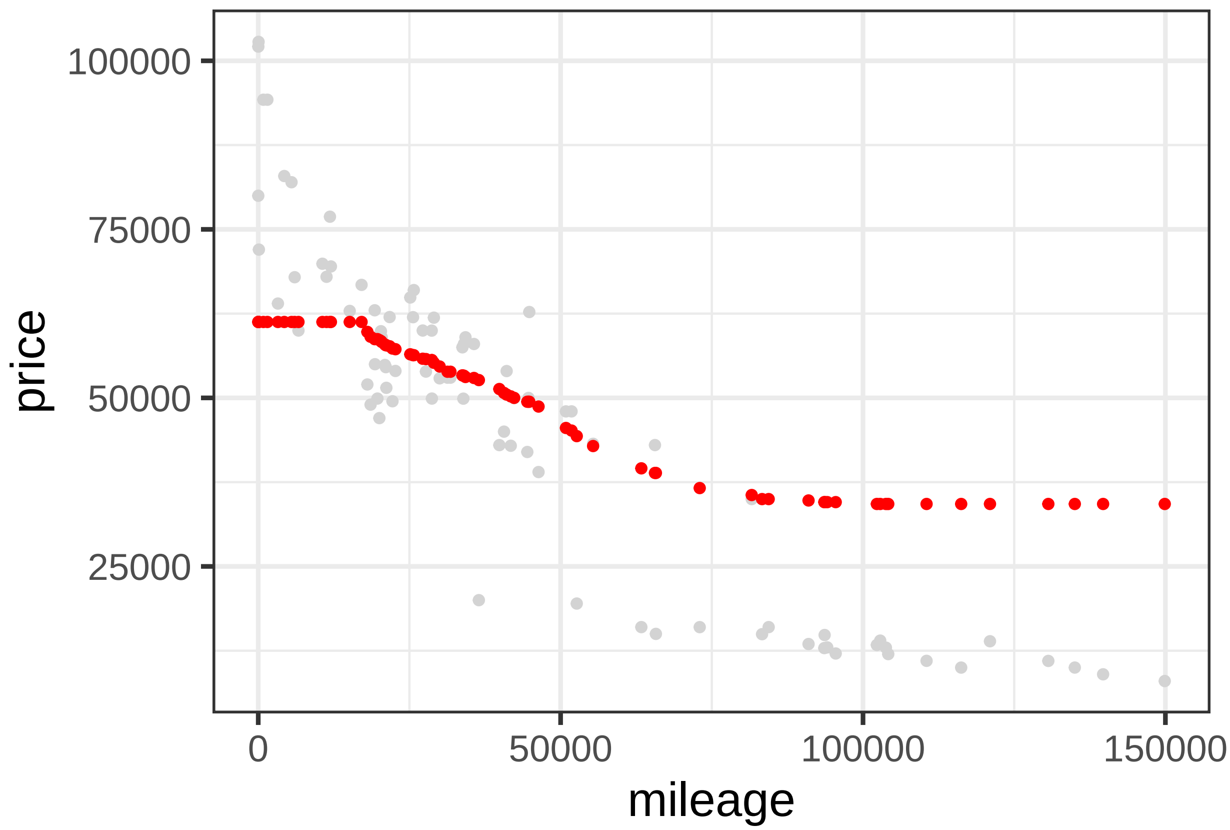
```r
rmse(y_test, ypred_knn190)
```

```
## [1] 15827.8
```

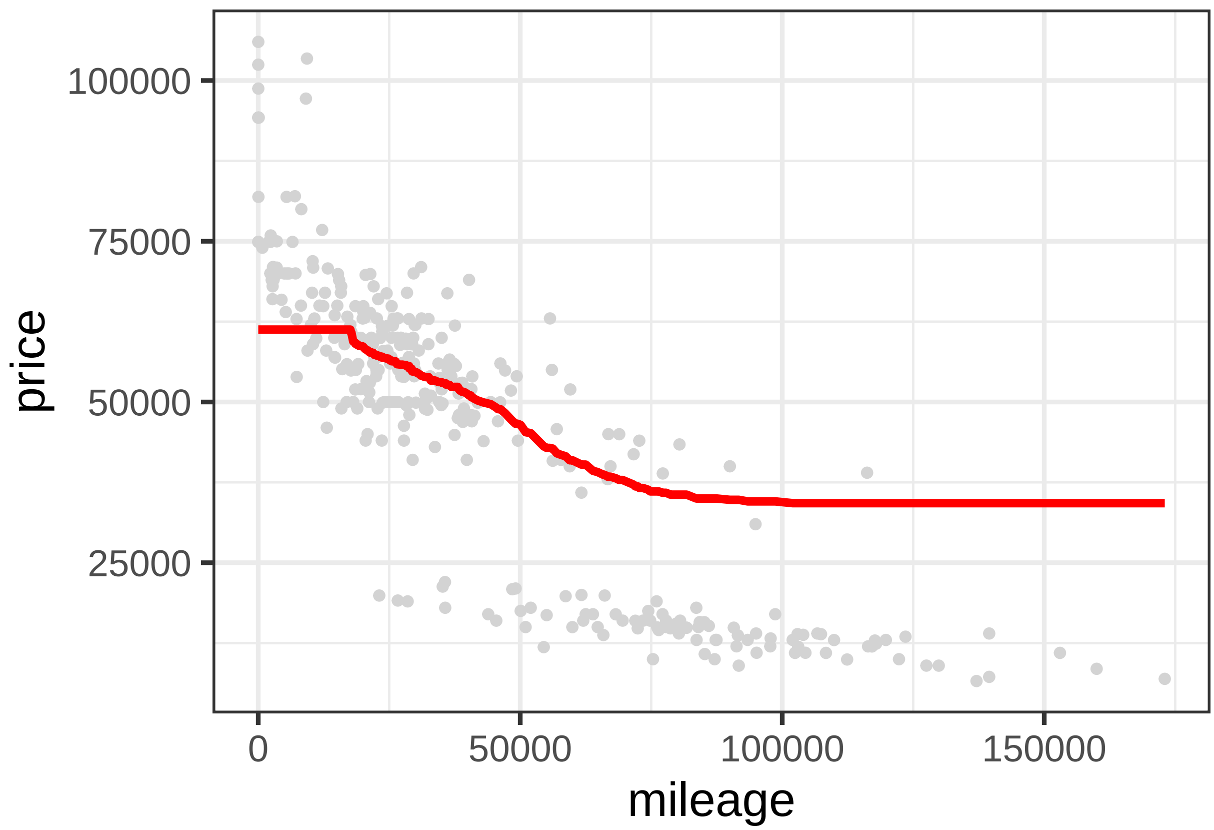
```r
#attach predictions to data frame
D_test$ypred_lm2 = ypred_lm2
D_test$ypred_knn190 = ypred_knn190

p_test = ggplot(data = D_test) +
  geom_point(mapping = aes(x = mileage, y = price), color='lightgrey') +
  theme_bw(base_size=18)
p_test
```




```r
p_test + geom_point(aes(x = mileage, y = ypred_knn190), color='red')
```




```
#KNN variances
knn_model = knn.reg(X_train, X_train, y_train, k = 190)

D_train$ypred = knn_model$pred
p_train = ggplot(data = D_train) +
  geom_point(mapping = aes(x = mileage, y = price), color='lightgrey') +
  theme_bw(base_size=18)
p_train + geom_path(mapping = aes(x=mileage, y=ypred), color='red', size=1.5)
```




```r
#K=210
lm1 = lm(price ~ mileage, data=D_train)
lm2 = lm(price ~ poly(mileage, 2), data=D_train)
knn210 = knn.reg(train = X_train, test = X_test, y = y_train, k=210)


#rmse
rmse = function(y, ypred) {
  sqrt(mean(data.matrix((y-ypred)^2)))
}

ypred_lm1 = predict(lm1, X_test)
ypred_lm2 = predict(lm2, X_test)
ypred_knn210 = knn210$pred

rmse(y_test, ypred_lm1)
```

```
## [1] 10838.68
```

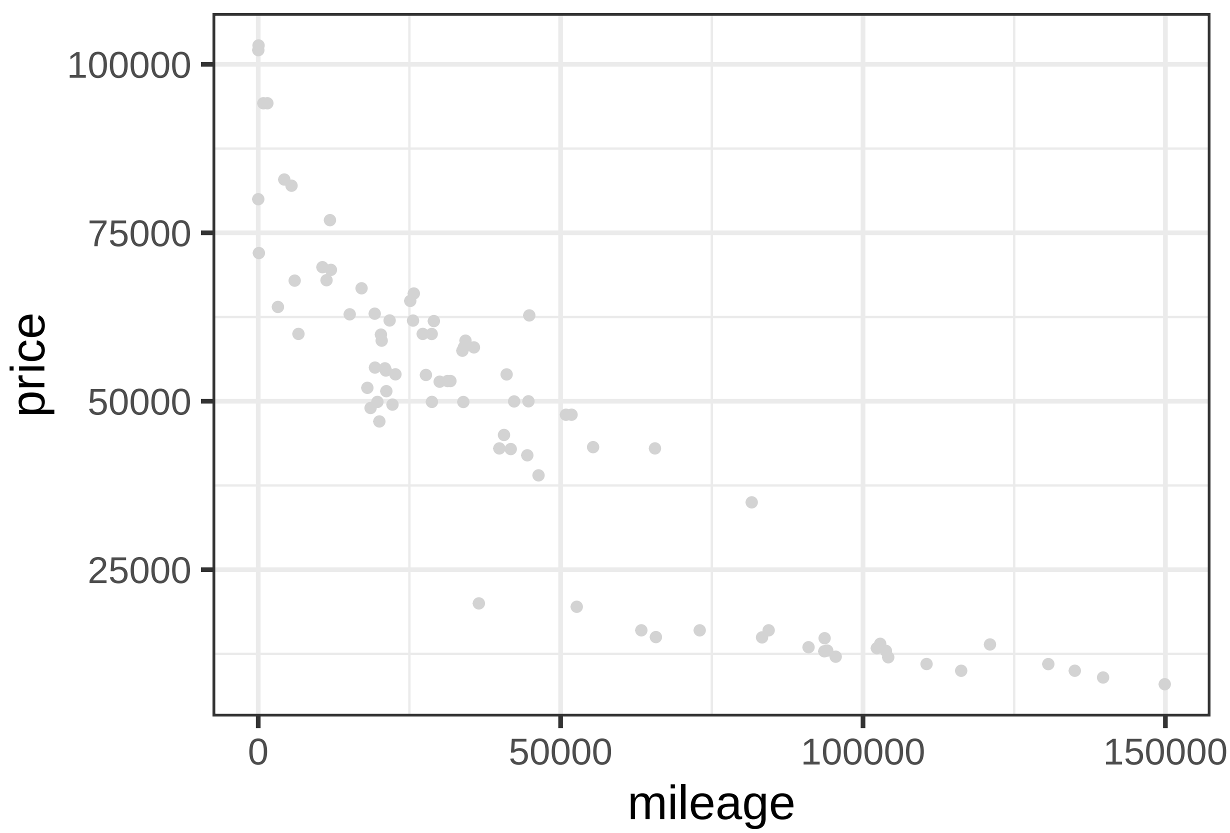
```r
rmse(y_test, ypred_lm2)
```

```
## [1] 9052.279
```

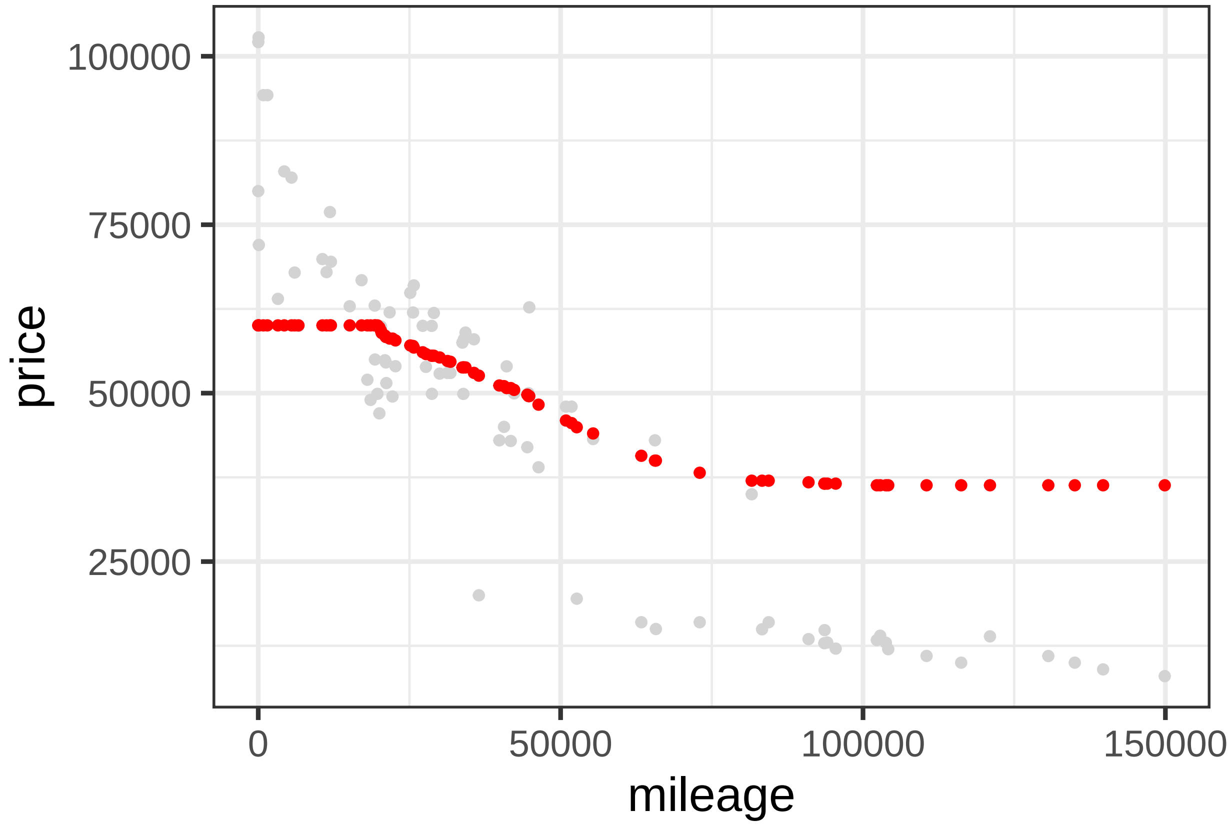
```r
rmse(y_test, ypred_knn210)
```

```
## [1] 16797.11
```

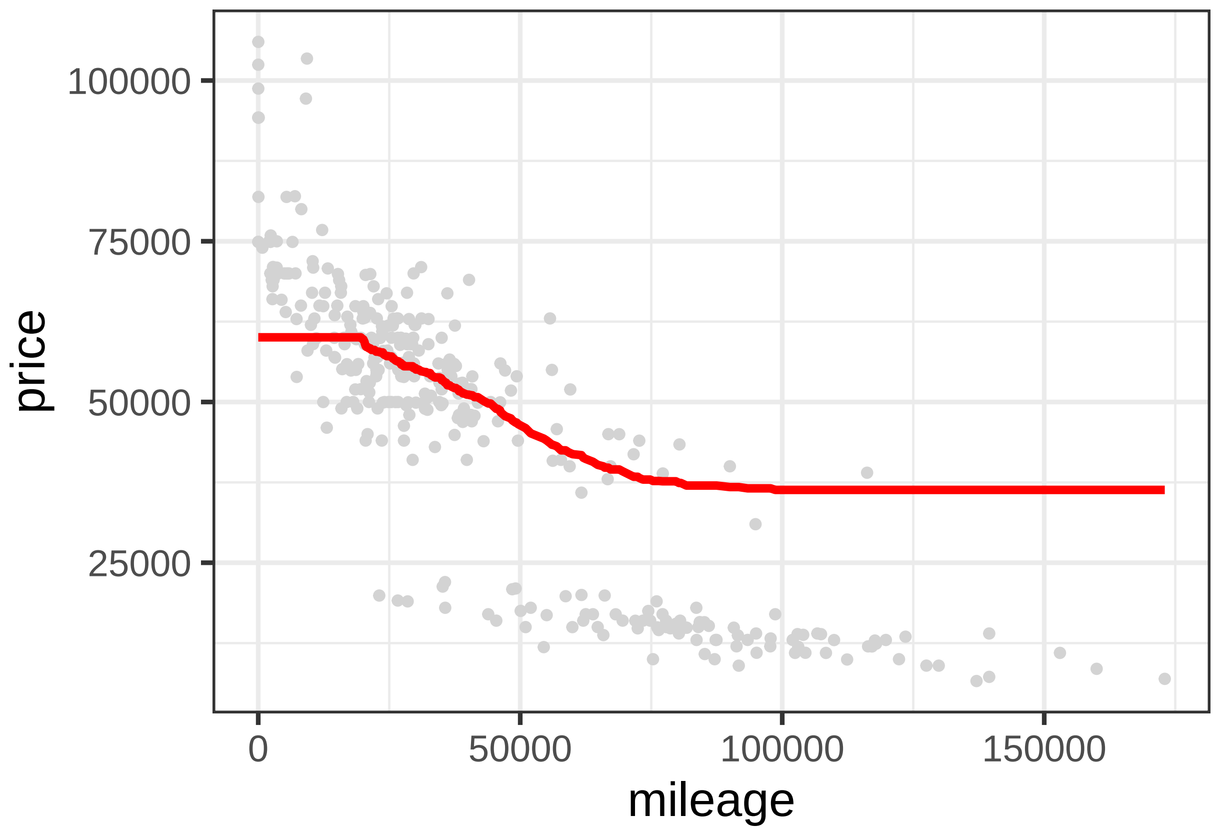
```r
#attach predictions to data frame
D_test$ypred_lm2 = ypred_lm2
D_test$ypred_knn210 = ypred_knn210

p_test = ggplot(data = D_test) +
  geom_point(mapping = aes(x = mileage, y = price), color='lightgrey') +
  theme_bw(base_size=18)
p_test
```




```r
p_test + geom_point(aes(x = mileage, y = ypred_knn210), color='red')
```




```
#KNN variances
knn_model = knn.reg(X_train, X_train, y_train, k = 210)

D_train$ypred = knn_model$pred
p_train = ggplot(data = D_train) +
  geom_point(mapping = aes(x = mileage, y = price), color='lightgrey') +
  theme_bw(base_size=18)
p_train + geom_path(mapping = aes(x=mileage, y=ypred), color='red', size=1.5)
```




```
#K=230
lm1 = lm(price ~ mileage, data=D_train)
lm2 = lm(price ~ poly(mileage, 2), data=D_train)
knn230 = knn.reg(train = X_train, test = X_test, y = y_train, k=230)


#rmse
rmse = function(y, ypred) {
  sqrt(mean(data.matrix((y-ypred)^2)))
}

ypred_lm1 = predict(lm1, X_test)
ypred_lm2 = predict(lm2, X_test)
ypred_knn230 = knn230$pred

rmse(y_test, ypred_lm1)
```

```
## [1] 10838.68
```

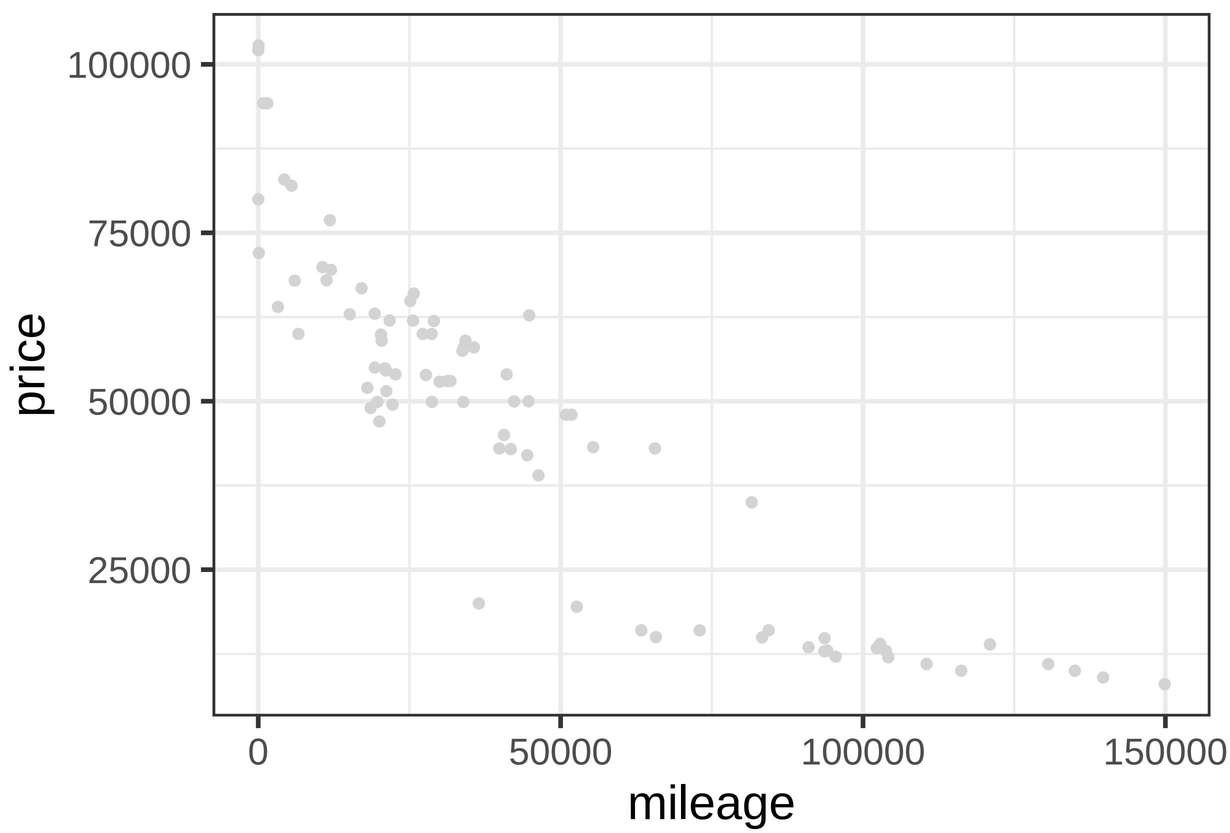
```
rmse(y_test, ypred_lm2)
```

```
## [1] 9052.279
```

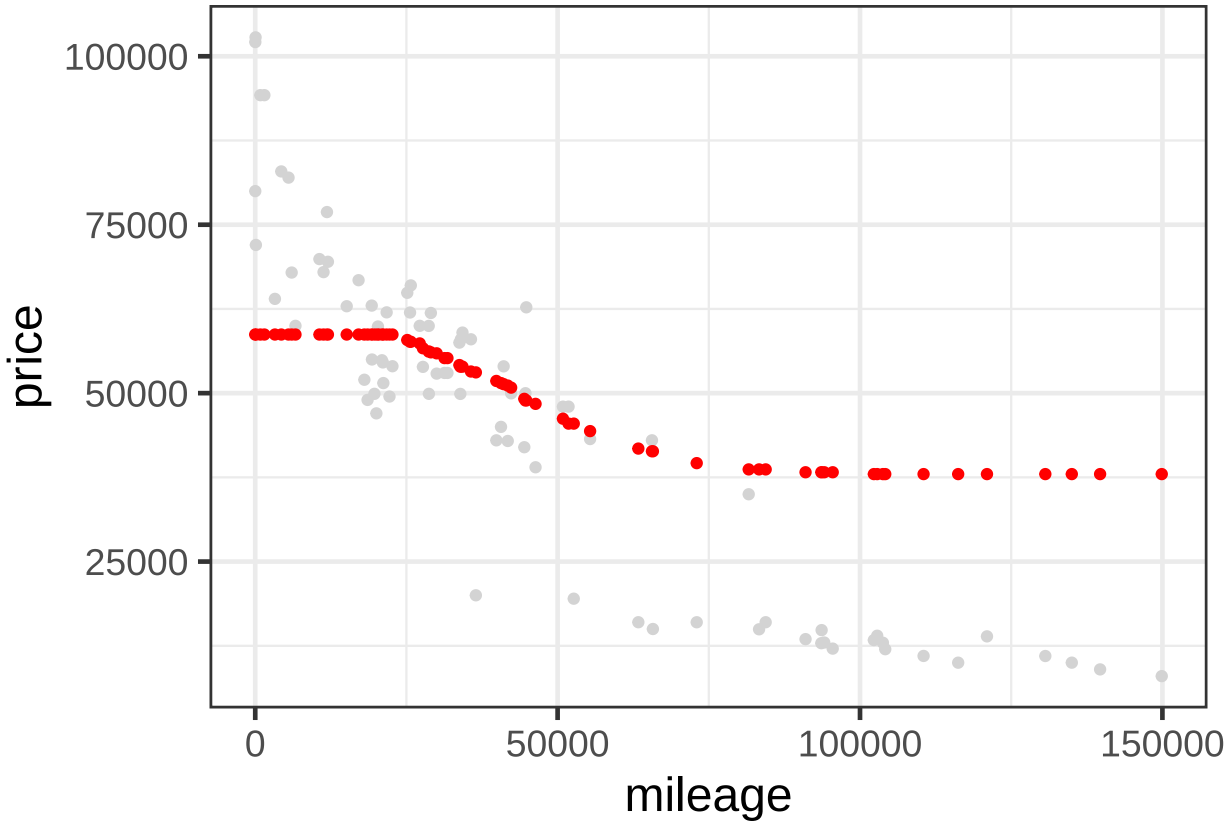
```r
rmse(y_test, ypred_knn230)
```

```
## [1] 17656.89
```

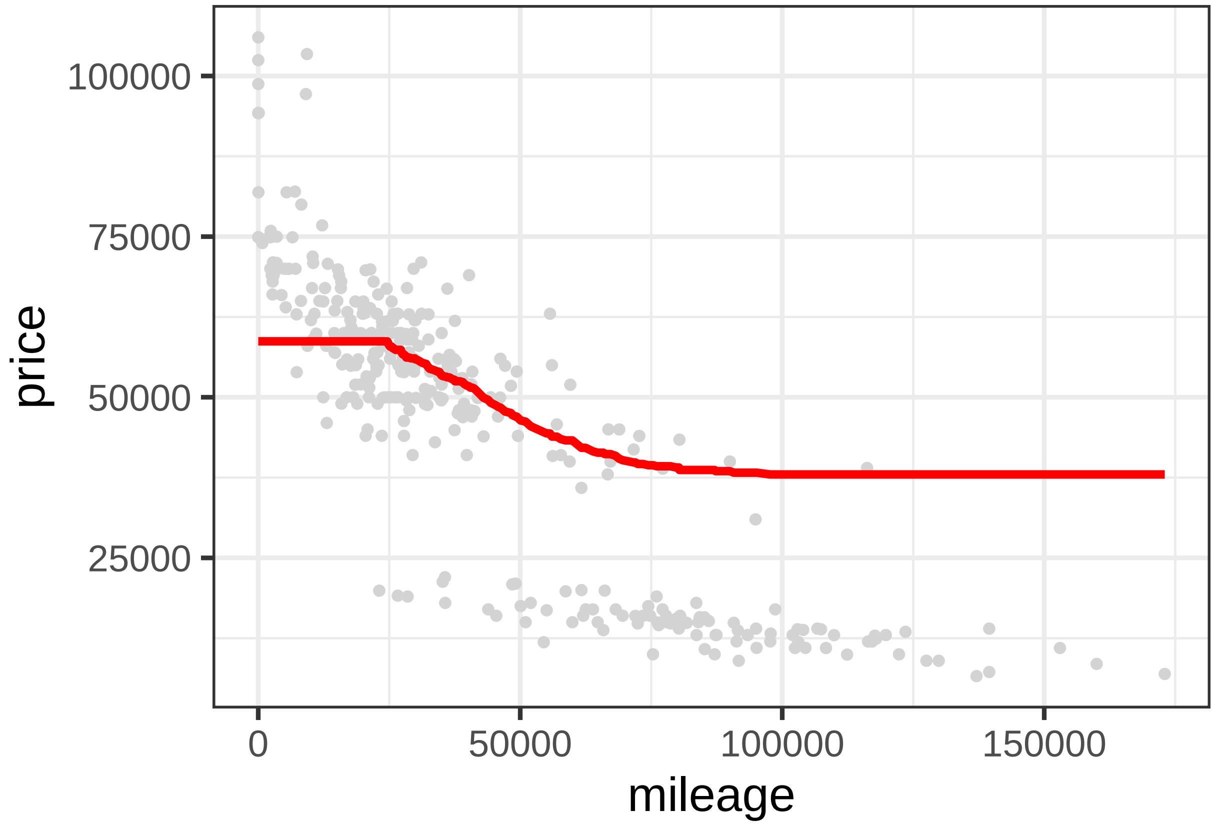
```r
#attach predictions to data frame
D_test$ypred_lm2 = ypred_lm2
D_test$ypred_knn230 = ypred_knn230

p_test = ggplot(data = D_test) +
  geom_point(mapping = aes(x = mileage, y = price), color='lightgrey') +
  theme_bw(base_size=18)
p_test
```




```r
p_test + geom_point(aes(x = mileage, y = ypred_knn230), color='red')
```




```
#KNN variances
knn_model = knn.reg(X_train, X_train, y_train, k = 230)

D_train$ypred = knn_model$pred
p_train = ggplot(data = D_train) +
  geom_point(mapping = aes(x = mileage, y = price), color='lightgrey') +
  theme_bw(base_size=18)
p_train + geom_path(mapping = aes(x=mileage, y=ypred), color='red', size=1.5)
```




```
#K=250
lm1 = lm(price ~ mileage, data=D_train)
lm2 = lm(price ~ poly(mileage, 2), data=D_train)
knn250 = knn.reg(train = X_train, test = X_test, y = y_train, k=250)


#rmse
rmse = function(y, ypred) {
  sqrt(mean(data.matrix((y-ypred)^2)))
}

ypred_lm1 = predict(lm1, X_test)
ypred_lm2 = predict(lm2, X_test)
ypred_knn250 = knn250$pred

rmse(y_test, ypred_lm1)
```

```
## [1] 10838.68
```

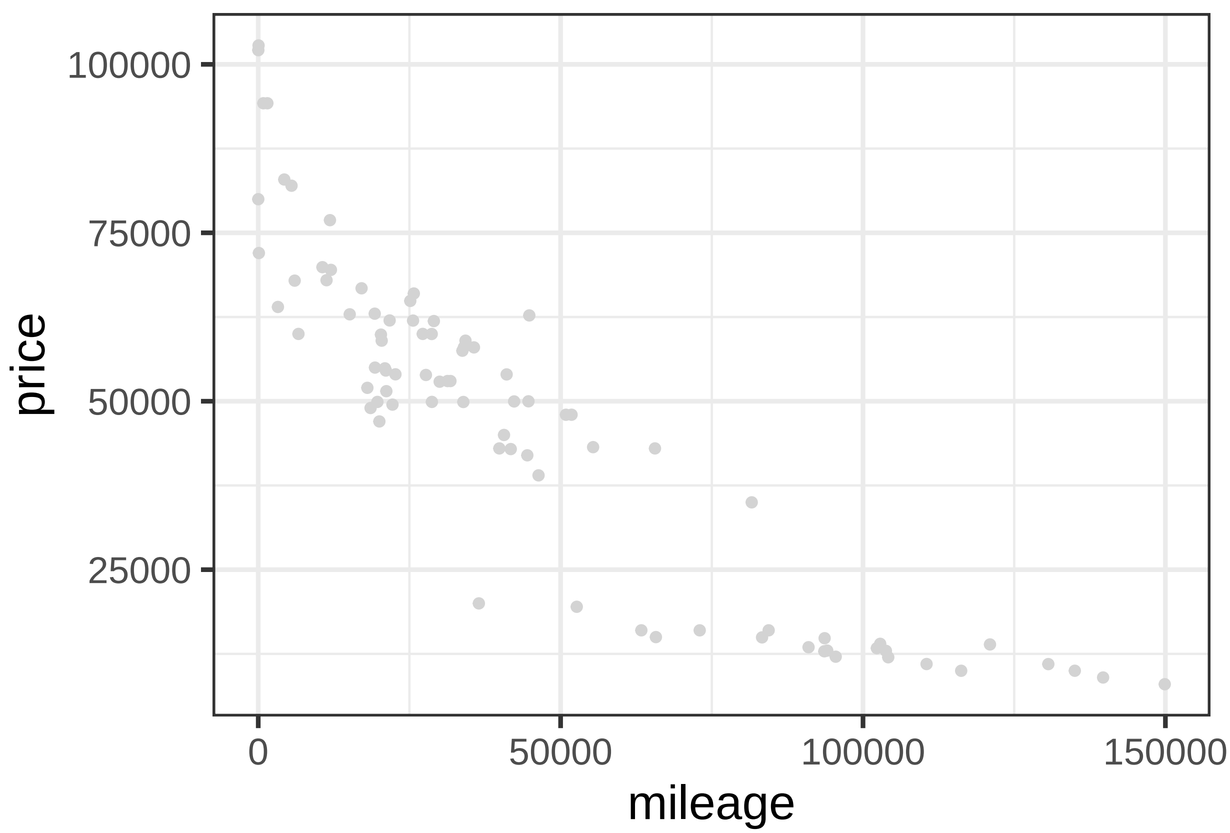
```
rmse(y_test, ypred_lm2)
```

```
## [1] 9052.279
```

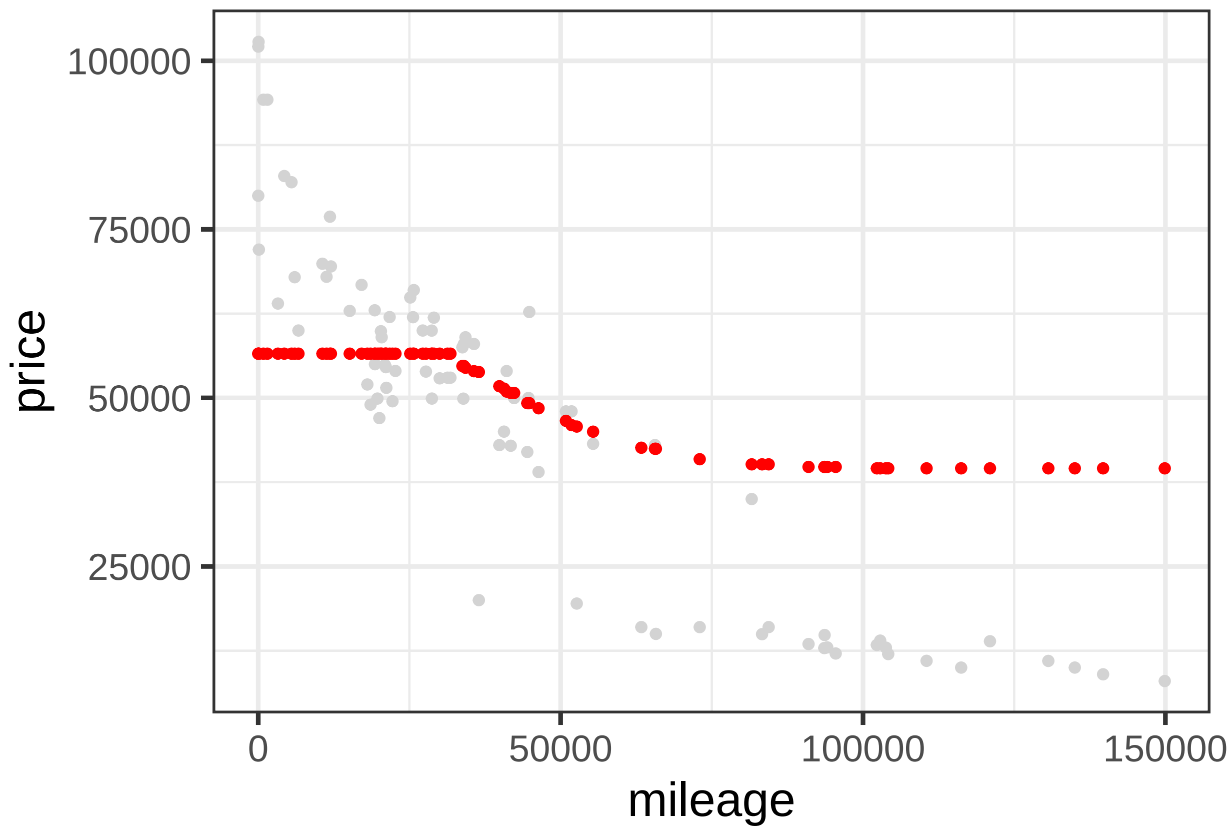
```r
rmse(y_test, ypred_knn250)
```

```
## [1] 18639.27
```

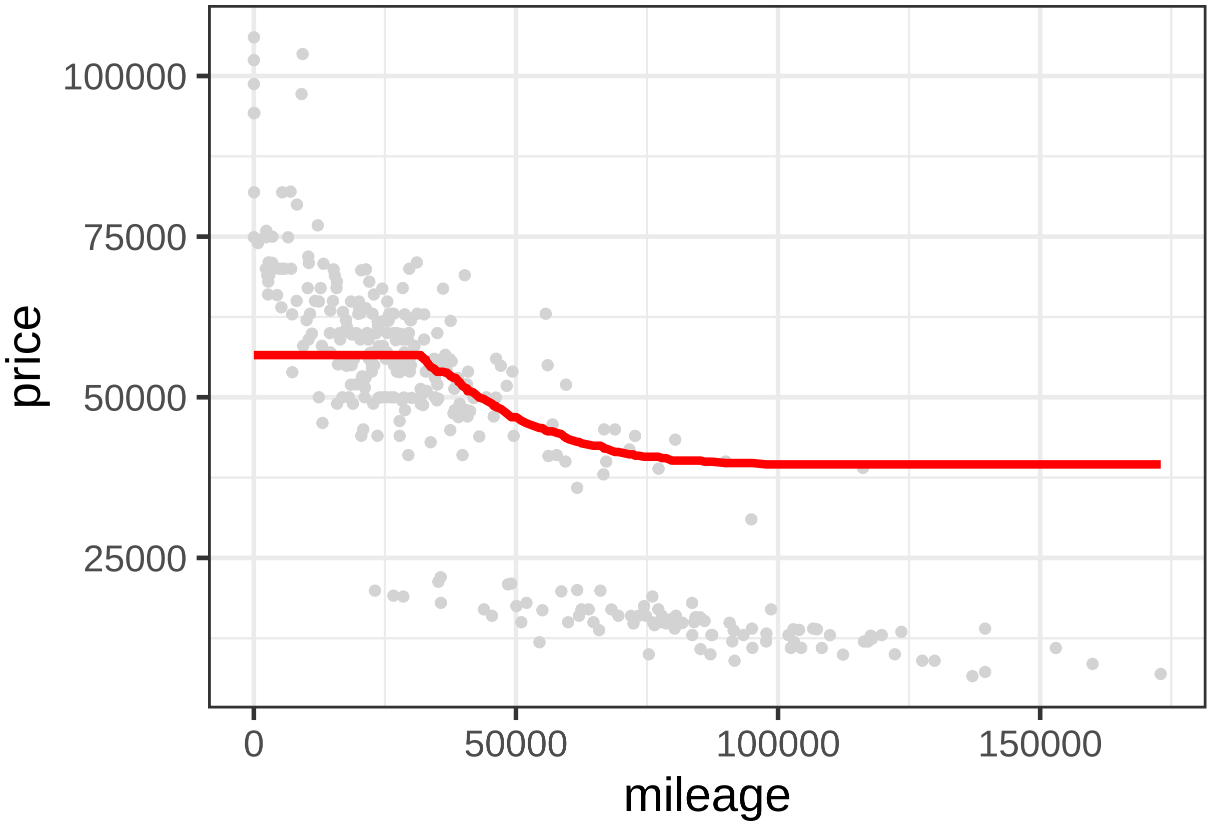
```r
#attach predictions to data frame
D_test$ypred_lm2 = ypred_lm2
D_test$ypred_knn250 = ypred_knn250

p_test = ggplot(data = D_test) +
  geom_point(mapping = aes(x = mileage, y = price), color='lightgrey') +
  theme_bw(base_size=18)
p_test
```




```r
p_test + geom_point(aes(x = mileage, y = ypred_knn250), color='red')
```




```
#KNN variances
knn_model = knn.reg(X_train, X_train, y_train, k = 250)

D_train$ypred = knn_model$pred
p_train = ggplot(data = D_train) +
  geom_point(mapping = aes(x = mileage, y = price), color='lightgrey') +
  theme_bw(base_size=18)
p_train + geom_path(mapping = aes(x=mileage, y=ypred), color='red', size=1.5)
```




```
#K=300
lm1 = lm(price ~ mileage, data=D_train)
lm2 = lm(price ~ poly(mileage, 2), data=D_train)
knn300 = knn.reg(train = X_train, test = X_test, y = y_train, k=300)


#rmse
rmse = function(y, ypred) {
  sqrt(mean(data.matrix((y-ypred)^2)))
}

ypred_lm1 = predict(lm1, X_test)
ypred_lm2 = predict(lm2, X_test)
ypred_knn300 = knn300$pred

rmse(y_test, ypred_lm1)
```

```
## [1] 10838.68
```

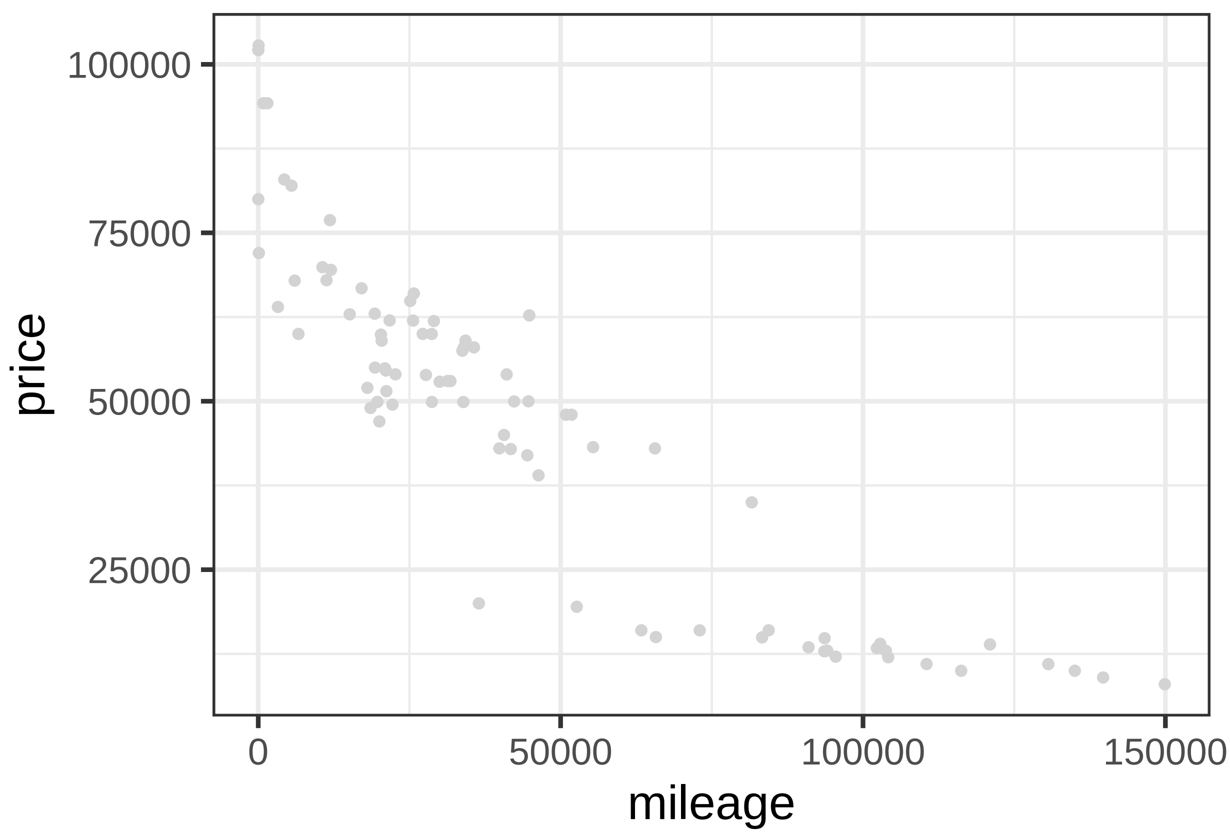
```
rmse(y_test, ypred_lm2)
```

```
## [1] 9052.279
```

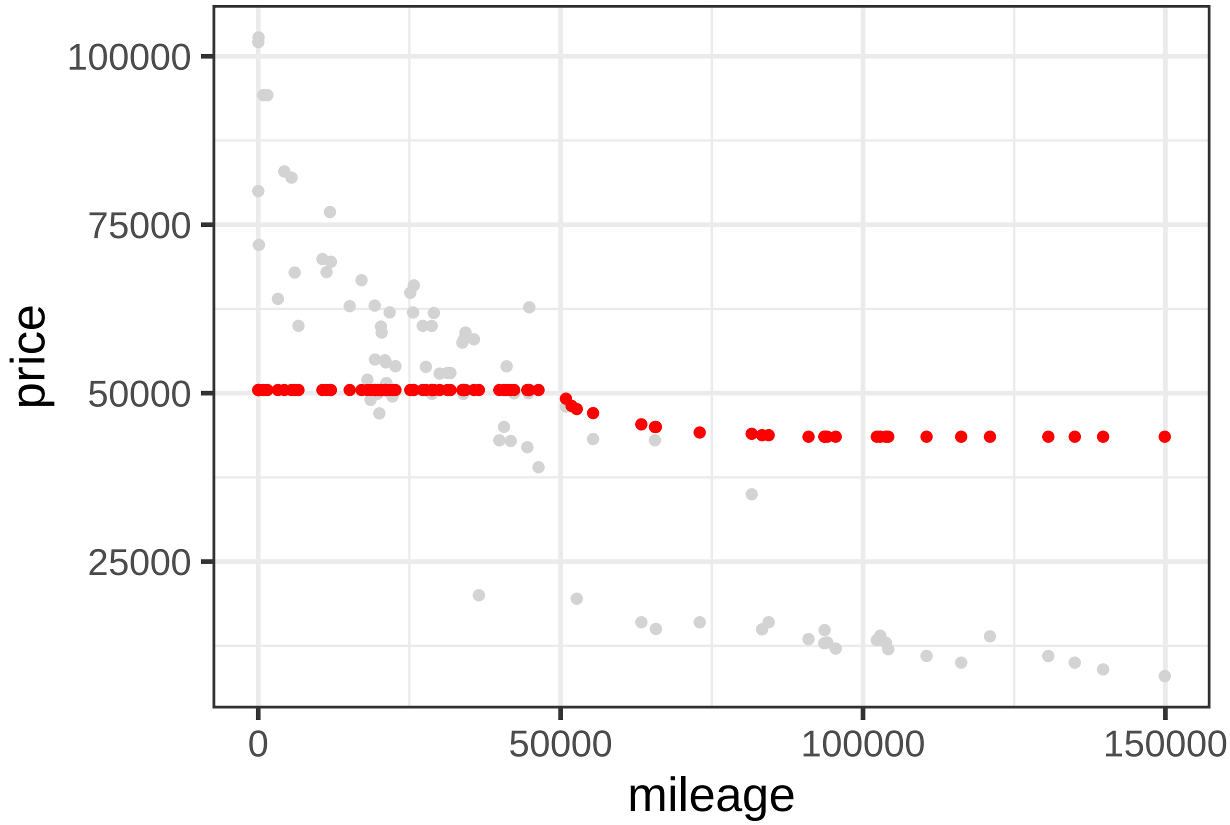
```r
rmse(y_test, ypred_knn300)
```

```
## [1] 21698.91
```

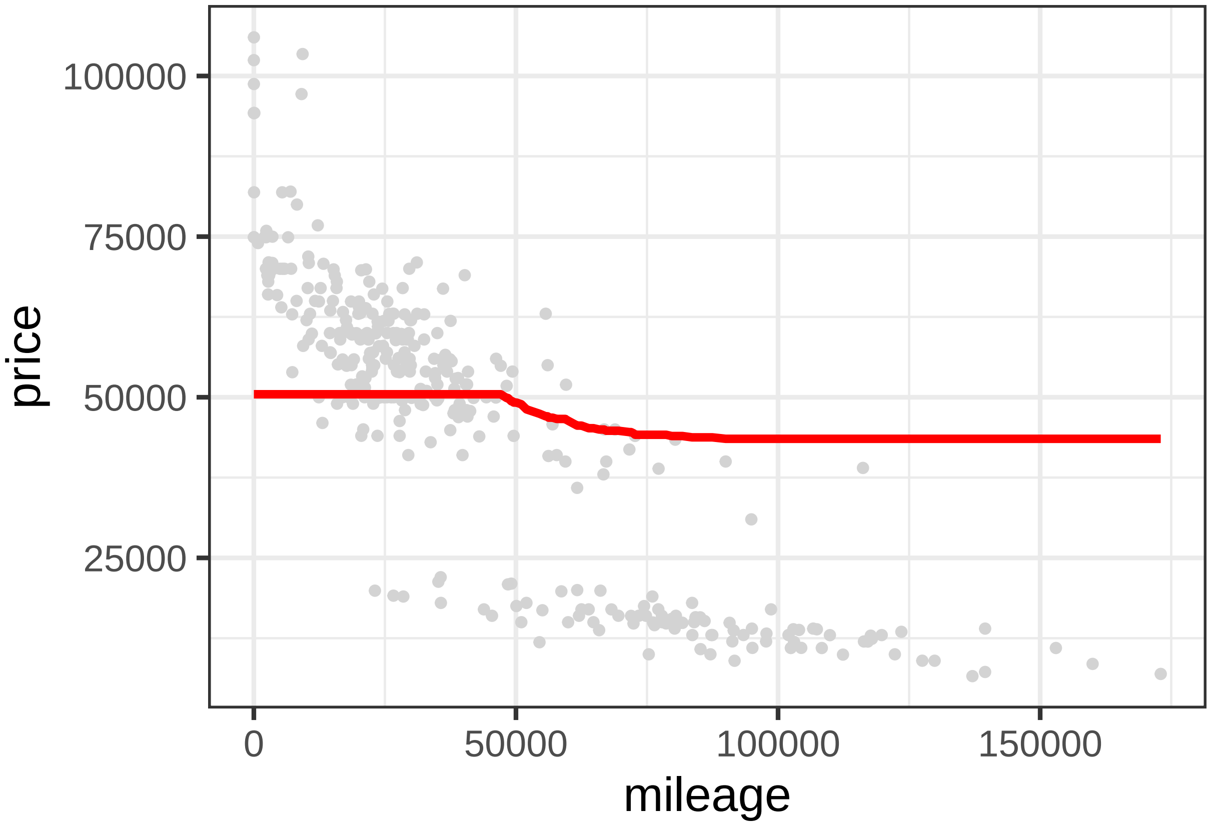
```r
#attach predictions to data frame
D_test$ypred_lm2 = ypred_lm2
D_test$ypred_knn300 = ypred_knn300

p_test = ggplot(data = D_test) +
  geom_point(mapping = aes(x = mileage, y = price), color='lightgrey') +
  theme_bw(base_size=18)
p_test
```




```r
p_test + geom_point(aes(x = mileage, y = ypred_knn300), color='red')
```




```
#KNN variances
knn_model = knn.reg(X_train, X_train, y_train, k = 300)

D_train$ypred = knn_model$pred
p_train = ggplot(data = D_train) +
  geom_point(mapping = aes(x = mileage, y = price), color='lightgrey') +
  theme_bw(base_size=18)
p_train + geom_path(mapping = aes(x=mileage, y=ypred), color='red', size=1.5)
```



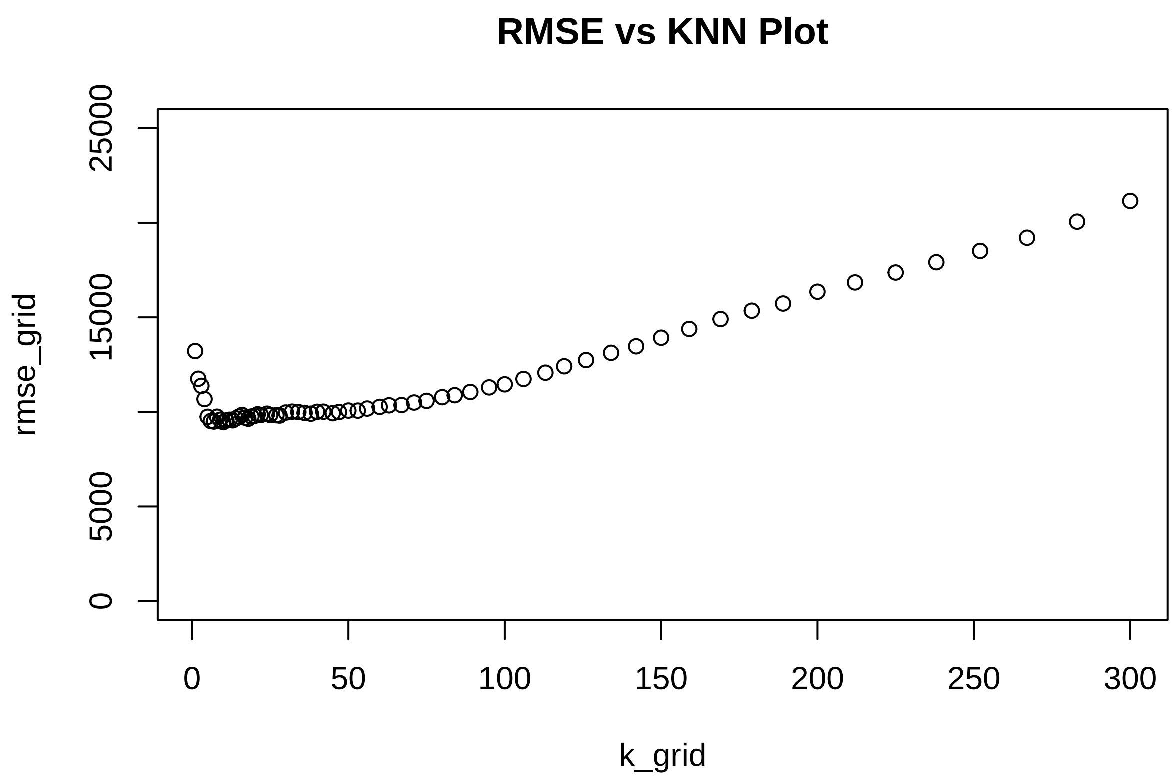

```
#RMSE plot vs K nearest neighbors 350 trim level.
N = nrow(sclass350)
N_train = floor(0.8*N)
N_test = N - N_train

train_ind = sort(sample.int(N, N_train, replace=FALSE))

D_train = sclass350[train_ind,]
D_train = arrange(D_train, mileage)
D_test = sclass350[-train_ind,]

y_train = D_train$price
X_train = data.frame(mileage=jitter(D_train$mileage))
X_test = data.frame(mileage=jitter(D_test$mileage))
y_test = D_test$price
library(foreach)
```

```
## Warning: package 'foreach' was built under R version 3.6.3
```

```
##
## Attaching package: 'foreach'
```

```
## The following objects are masked from 'package:purrr':
##
##     accumulate, when
```

```
k_grid = exp(seq(log(1), log(300), length=100)) %>% round %>% unique
rmse_grid = foreach(K = k_grid, .combine='c') %do% {
  knn_model = knn.reg(train = X_train, test = X_test, y = y_train, k=K)
  rmse(y_test, knn_model$pred)
}
rmse_plot = plot(k_grid, rmse_grid, ylim=c(0,25000), main="RMSE vs KNN Plot")
abline(rmse_plot)
```




```
#optimal K value for 350 AMG trim level

#K = 30
lm1 = lm(price ~ mileage, data=D_train)
lm2 = lm(price ~ poly(mileage, 2), data=D_train)
knn30 = knn.reg(train = X_train, test = X_test, y = y_train, k=30)


#rmse
rmse = function(y, ypred) {
  sqrt(mean(data.matrix((y-ypred)^2)))
}

ypred_lm1 = predict(lm1, X_test)
ypred_lm2 = predict(lm2, X_test)
ypred_knn30 = knn30$pred
```

```r
rmse(y_test, ypred_lm1)
```

```
## [1] 11384.8
```

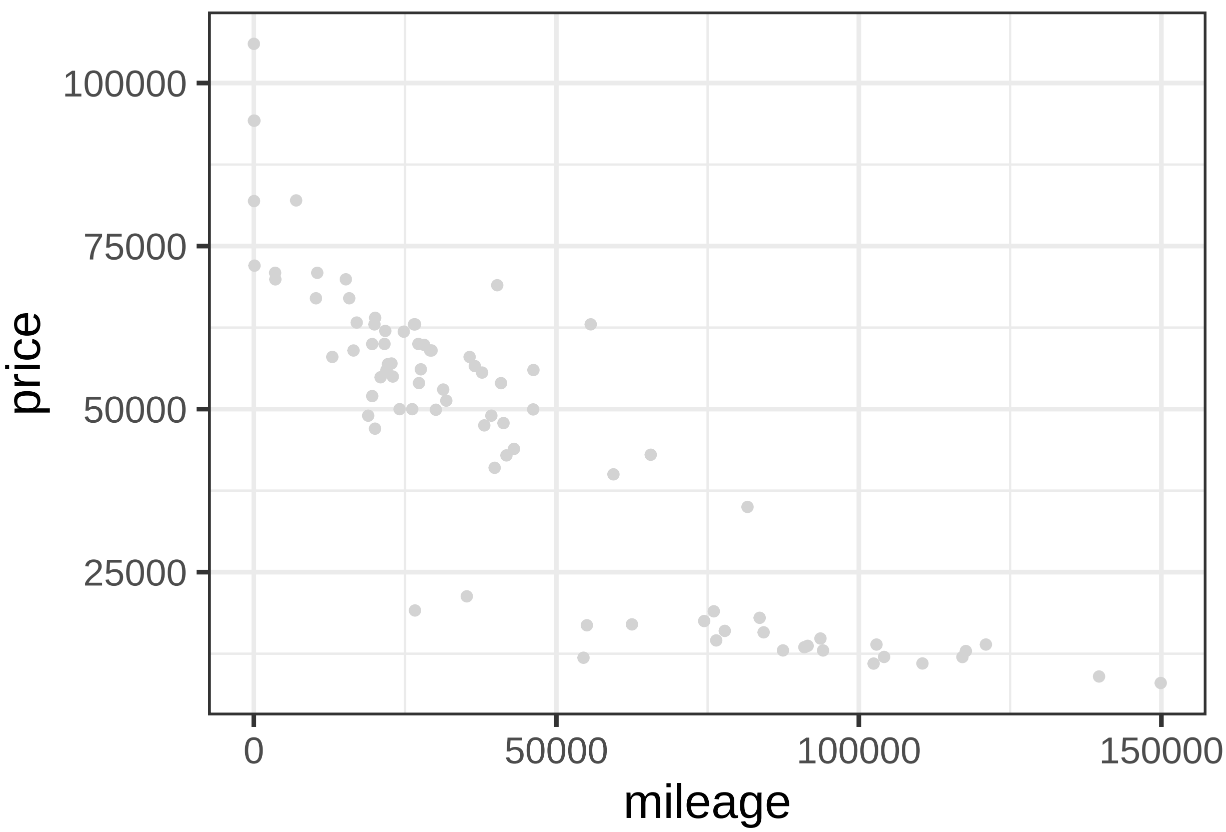
```r
rmse(y_test, ypred_lm2)
```

```
## [1] 10009.26
```

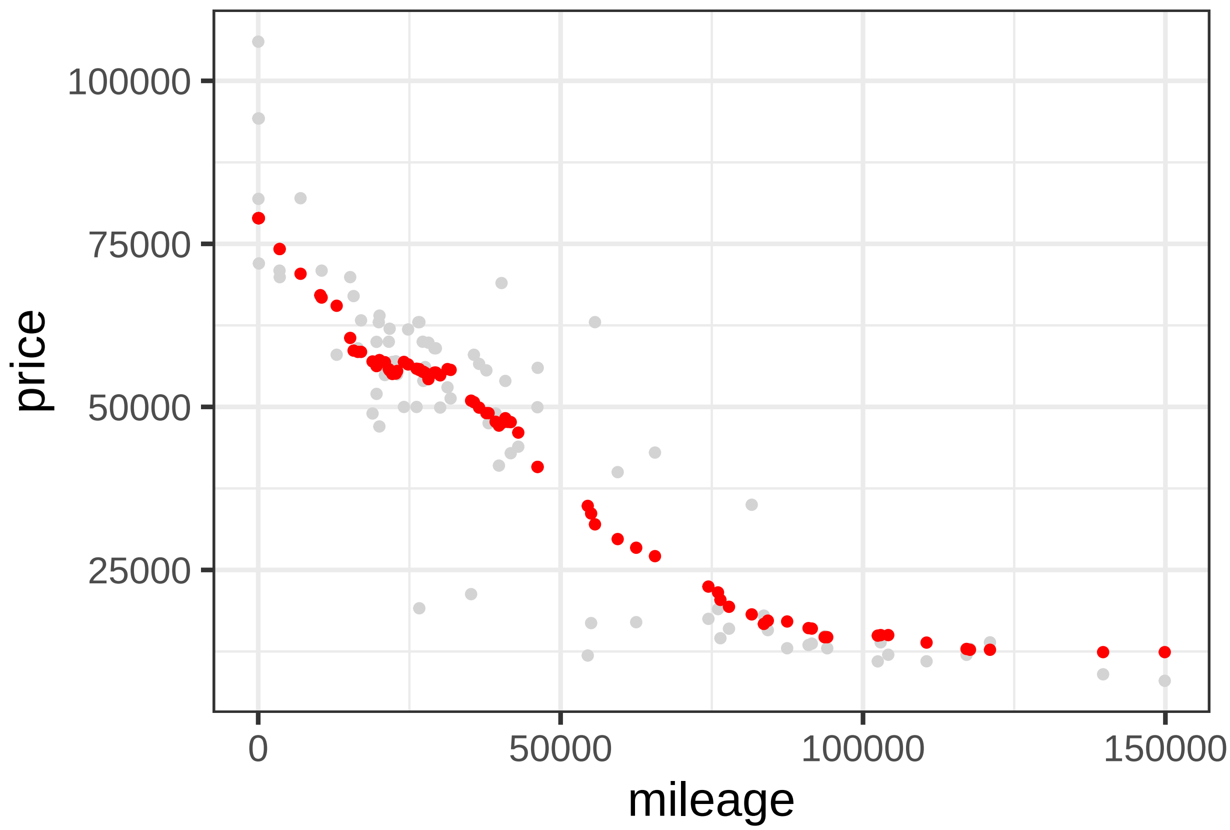
```r
rmse(y_test, ypred_knn30)
```

```
## [1] 9966.32
```

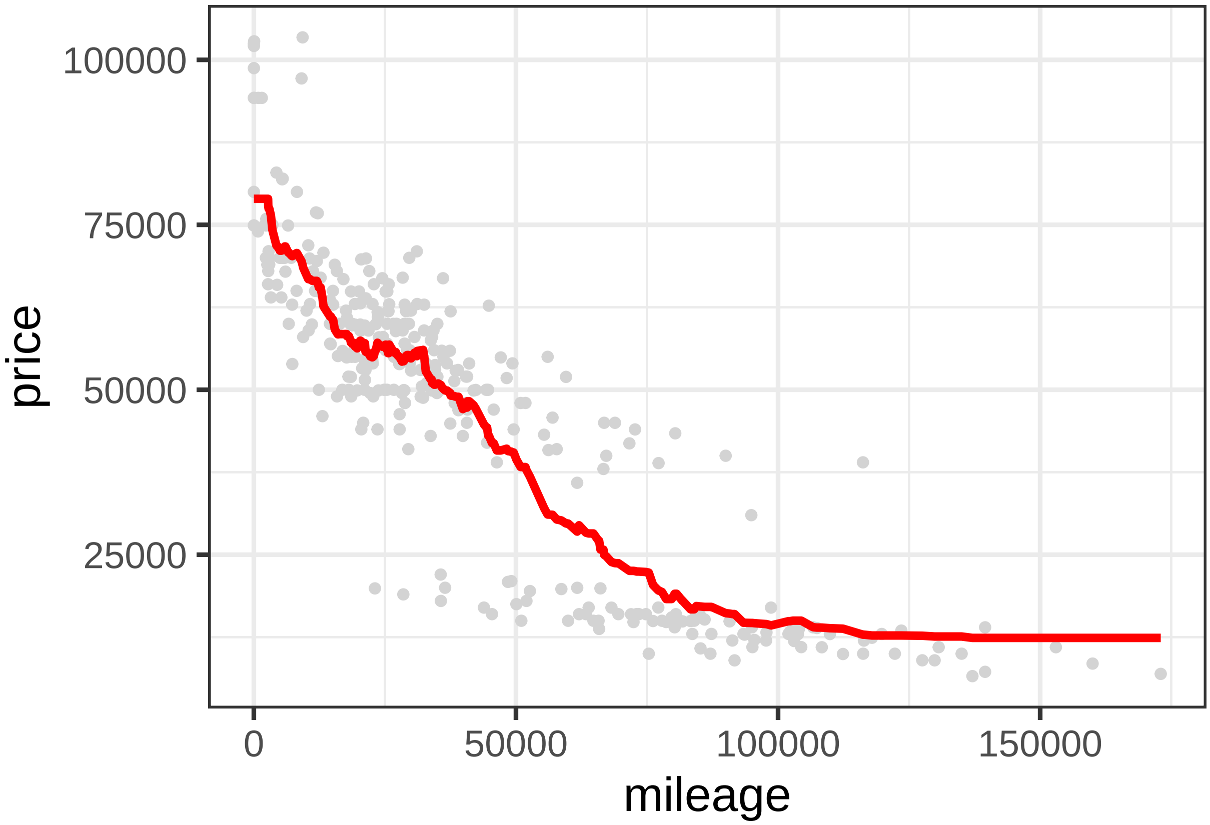
```r
#attach predictions to data frame
D_test$ypred_lm2 = ypred_lm2
D_test$ypred_knn30 = ypred_knn30

p_test = ggplot(data = D_test) +
  geom_point(mapping = aes(x = mileage, y = price), color='lightgrey') +
  theme_bw(base_size=18)
p_test
```



```
p_test + geom_point(aes(x = mileage, y = ypred_knn30), color='red')
```




```
#KNN variances
knn_model = knn.reg(X_train, X_train, y_train, k = 30)

D_train$ypred = knn_model$pred
p_train = ggplot(data = D_train) +
  geom_point(mapping = aes(x = mileage, y = price), color='lightgrey') +
  theme_bw(base_size=18)
p_train + geom_path(mapping = aes(x=mileage, y=ypred), color='red', size=1.5)
```




```
#The value of K=30 is optimal for the 350 AMG trim level with an out of sample RMSE of 25924.26. This v
```

```
#Train-test split for 65 AMG trim level: We are looking for a model that gives the best explanation wit
N = nrow(sclass65AMG)
N_train = floor(0.8*N)
N_test = N - N_train

train_ind = sort(sample.int(N, N_train, replace=FALSE))

D_train = sclass65AMG[train_ind,]
D_train = arrange(D_train, mileage)
D_test = sclass65AMG[-train_ind,]

y_train = D_train$price
X_train = data.frame(mileage=jitter(D_train$mileage))
X_test = data.frame(mileage=jitter(D_test$mileage))
y_test = D_test$price

#Running KNN test for various values of K to determine the optimal predictive model.This will start at
#KNN = 2
#The first step is to make predictions to the data frame.
lm1 = lm(price ~ mileage, data=D_train)
lm2 = lm(price ~ poly(mileage, 2), data=D_train)
knn2 = knn.reg(train = X_train, test = X_test, y = y_train, k=2)

#rmse calculation
```

```r
rmse = function(y, ypred) {
  sqrt(mean(data.matrix((y-ypred)^2)))
}

ypred_lm1 = predict(lm1, X_test)
ypred_lm2 = predict(lm2, X_test)
ypred_knn2 = knn2$pred

rmse(y_test, ypred_lm1)
```

```
## [1] 41419.01
```

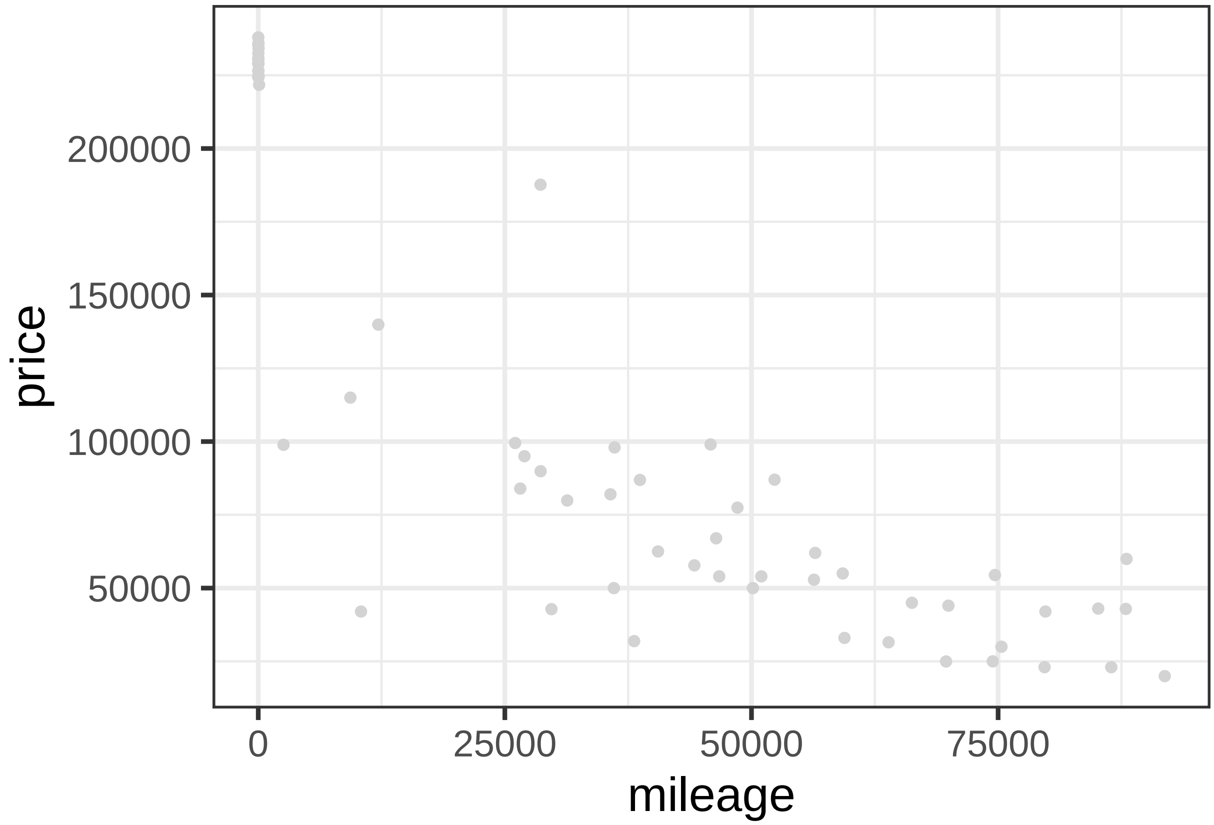
```r
rmse(y_test, ypred_lm2)
```

```
## [1] 32029.86
```

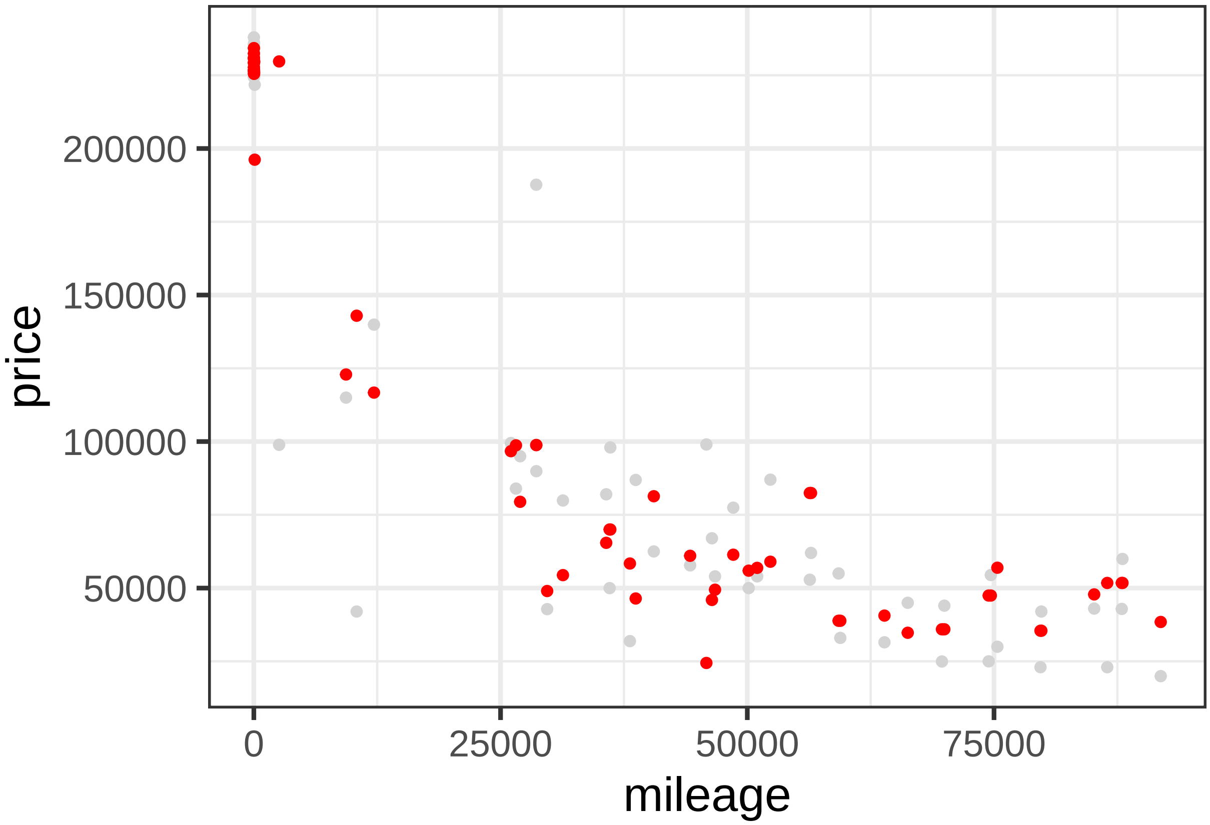
```r
rmse(y_test, ypred_knn2)
```

```
## [1] 30307.39
```

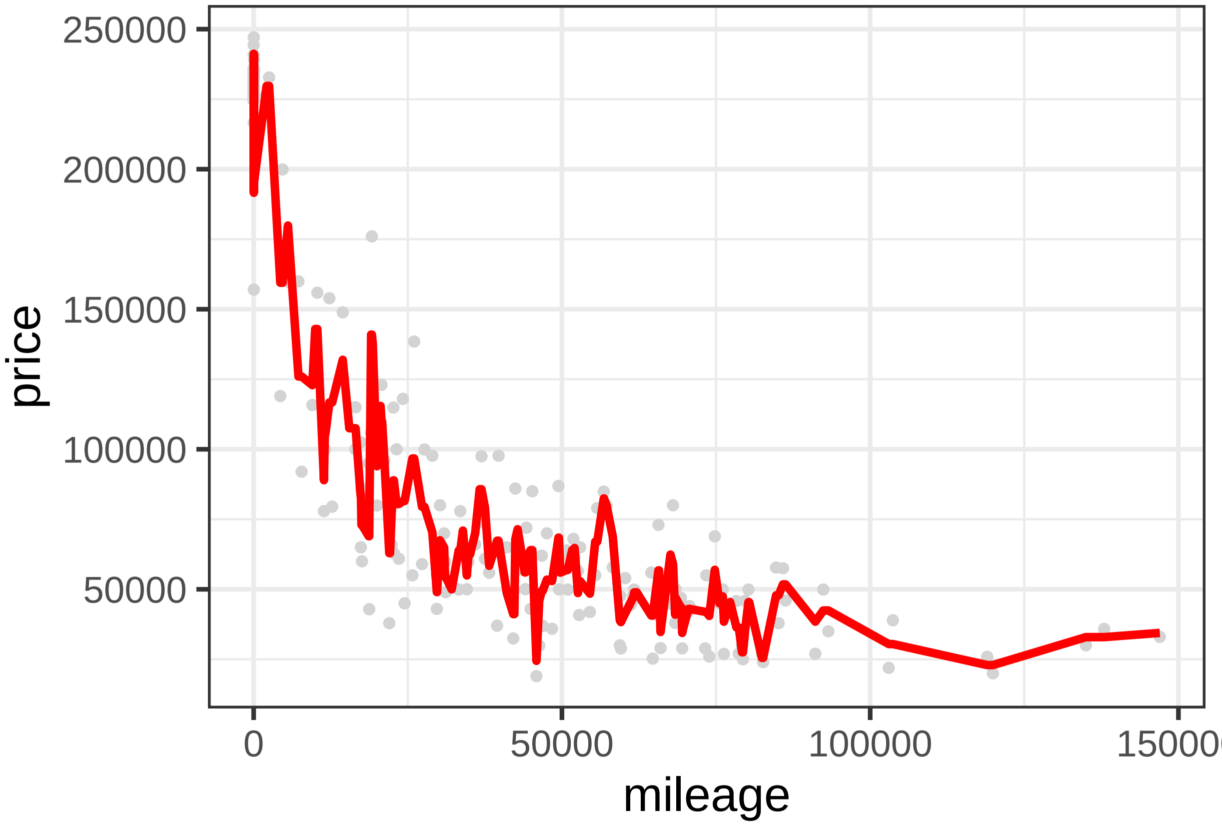
```r
#attach predictions to data frame
D_test$ypred_lm2 = ypred_lm2
D_test$ypred_knn2 = ypred_knn2

p_test = ggplot(data = D_test) +
  geom_point(mapping = aes(x = mileage, y = price), color='lightgrey') +
  theme_bw(base_size=18)
p_test
```




```
p_test + geom_point(aes(x = mileage, y = ypred_knn2), color='red')
```




```r
#KNN variances
knn_model = knn.reg(X_train, X_train, y_train, k = 2)

D_train$ypred = knn_model$pred
p_train = ggplot(data = D_train) +
  geom_point(mapping = aes(x = mileage, y = price), color='lightgrey') +
  theme_bw(base_size=18)
p_train + geom_path(mapping = aes(x=mileage, y=ypred), color='red', size=1.5)
```




```
#K = 3
lm1 = lm(price ~ mileage, data=D_train)
lm2 = lm(price ~ poly(mileage, 2), data=D_train)
knn3 = knn.reg(train = X_train, test = X_test, y = y_train, k=3)


#rmse
rmse = function(y, ypred) {
  sqrt(mean(data.matrix((y-ypred)^2)))
}

ypred_lm1 = predict(lm1, X_test)
ypred_lm2 = predict(lm2, X_test)
ypred_knn3 = knn3$pred

rmse(y_test, ypred_lm1)
```

```
## [1] 41419.01
```

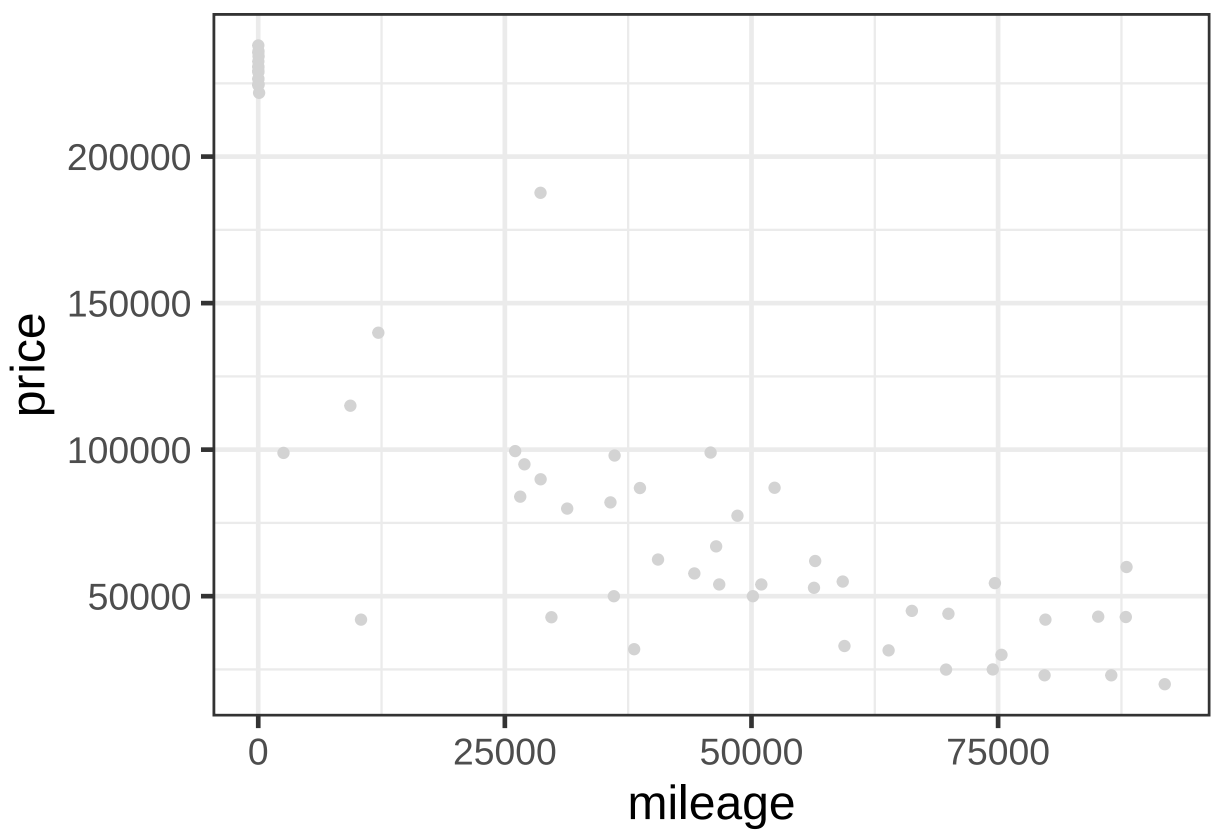
```
rmse(y_test, ypred_lm2)
```

```
## [1] 32029.86
```

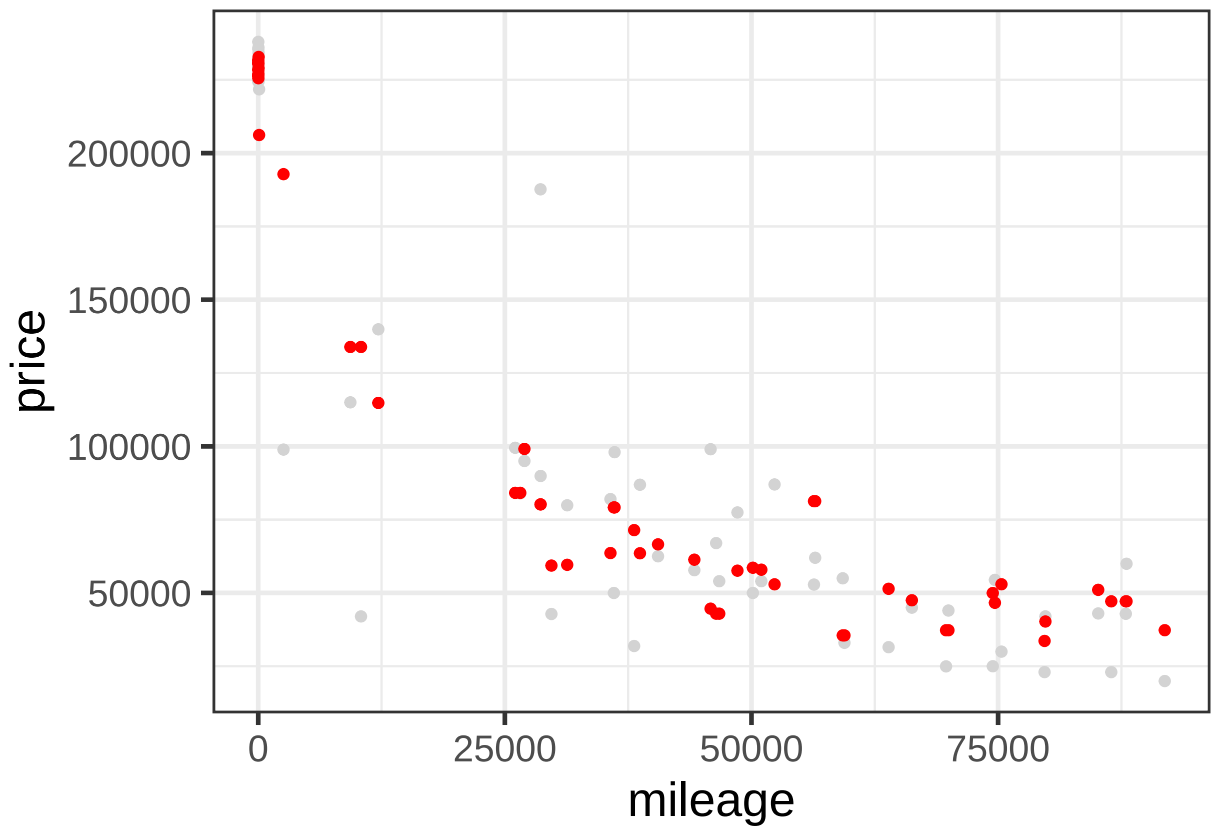
```r
rmse(y_test, ypred_knn3)
```

```
## [1] 27620.57
```

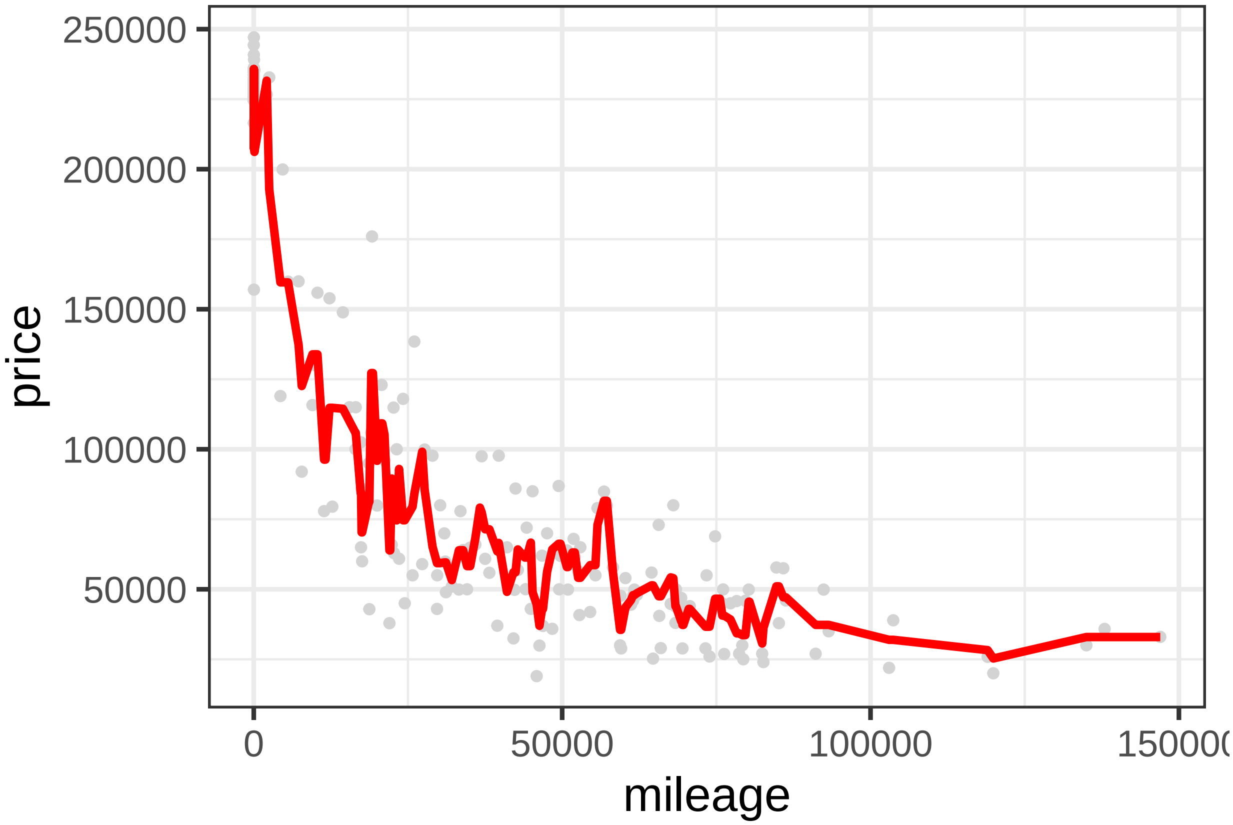
```r
#attach predictions to data frame
D_test$ypred_lm2 = ypred_lm2
D_test$ypred_knn3 = ypred_knn3

p_test = ggplot(data = D_test) +
  geom_point(mapping = aes(x = mileage, y = price), color='lightgrey') +
  theme_bw(base_size=18)
p_test
```




```r
p_test + geom_point(aes(x = mileage, y = ypred_knn3), color='red')
```




```
#KNN variances
knn_model = knn.reg(X_train, X_train, y_train, k = 3)

D_train$ypred = knn_model$pred
p_train = ggplot(data = D_train) +
  geom_point(mapping = aes(x = mileage, y = price), color='lightgrey') +
  theme_bw(base_size=18)
p_train + geom_path(mapping = aes(x=mileage, y=ypred), color='red', size=1.5)
```




```
#K = 4
lm1 = lm(price ~ mileage, data=D_train)
lm2 = lm(price ~ poly(mileage, 2), data=D_train)
knn4 = knn.reg(train = X_train, test = X_test, y = y_train, k=4)


#rmse
rmse = function(y, ypred) {
  sqrt(mean(data.matrix((y-ypred)^2)))
}

ypred_lm1 = predict(lm1, X_test)
ypred_lm2 = predict(lm2, X_test)
ypred_knn4 = knn4$pred

rmse(y_test, ypred_lm1)
```

```
## [1] 41419.01
```

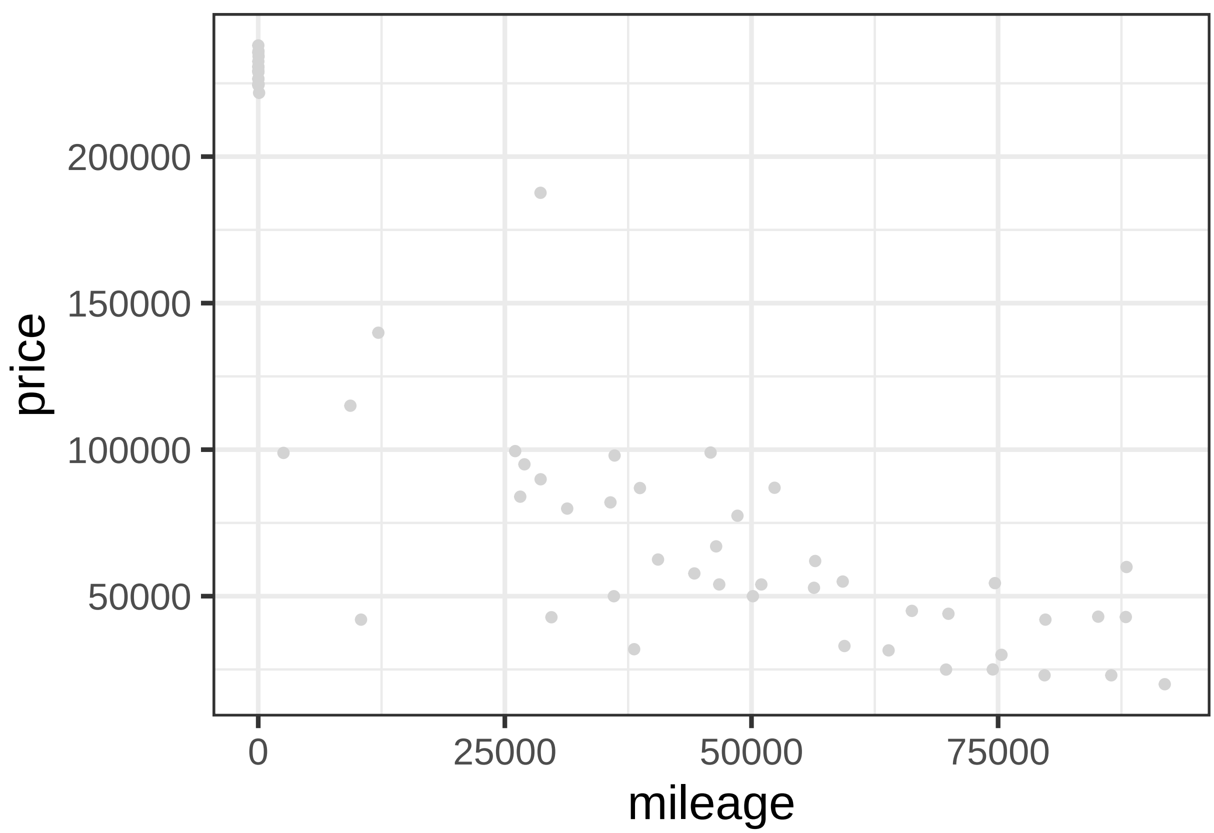
```
rmse(y_test, ypred_lm2)
```

```
## [1] 32029.86
```

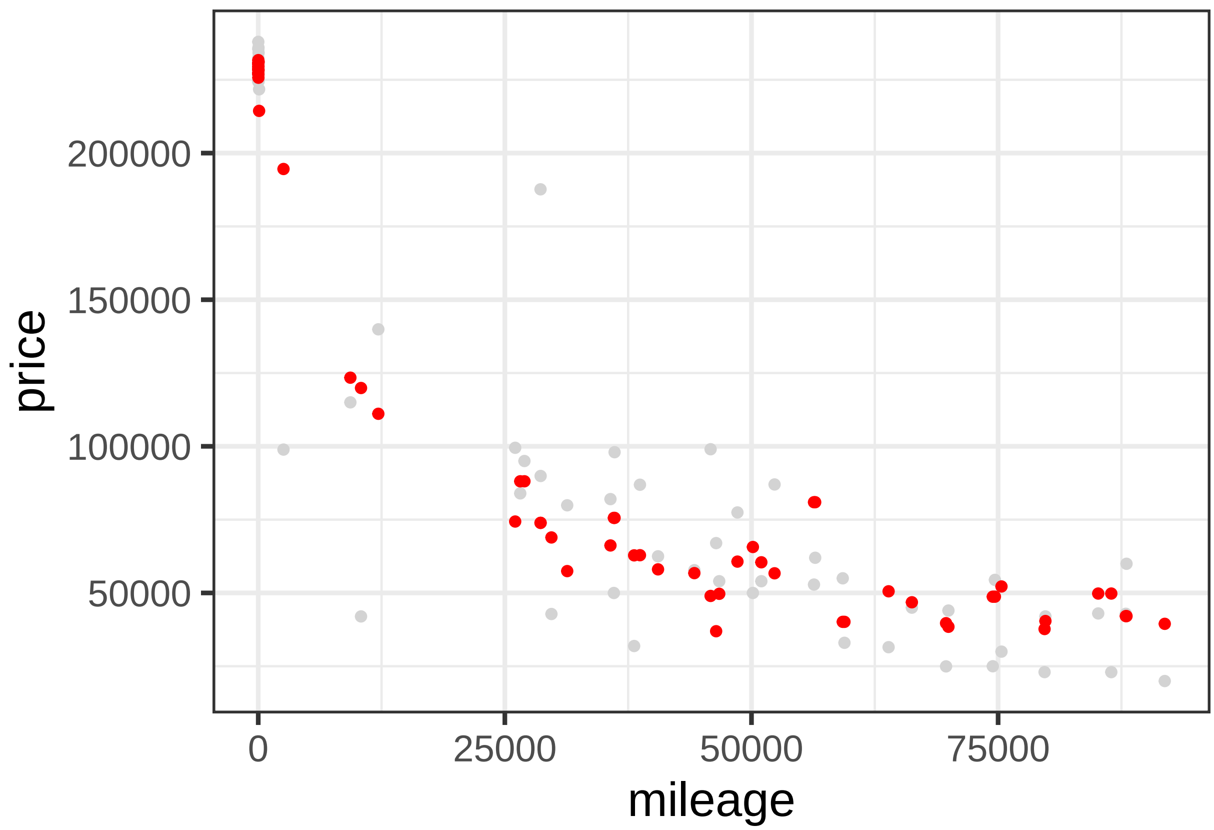
```r
rmse(y_test, ypred_knn4)
```

```
## [1] 27384.66
```

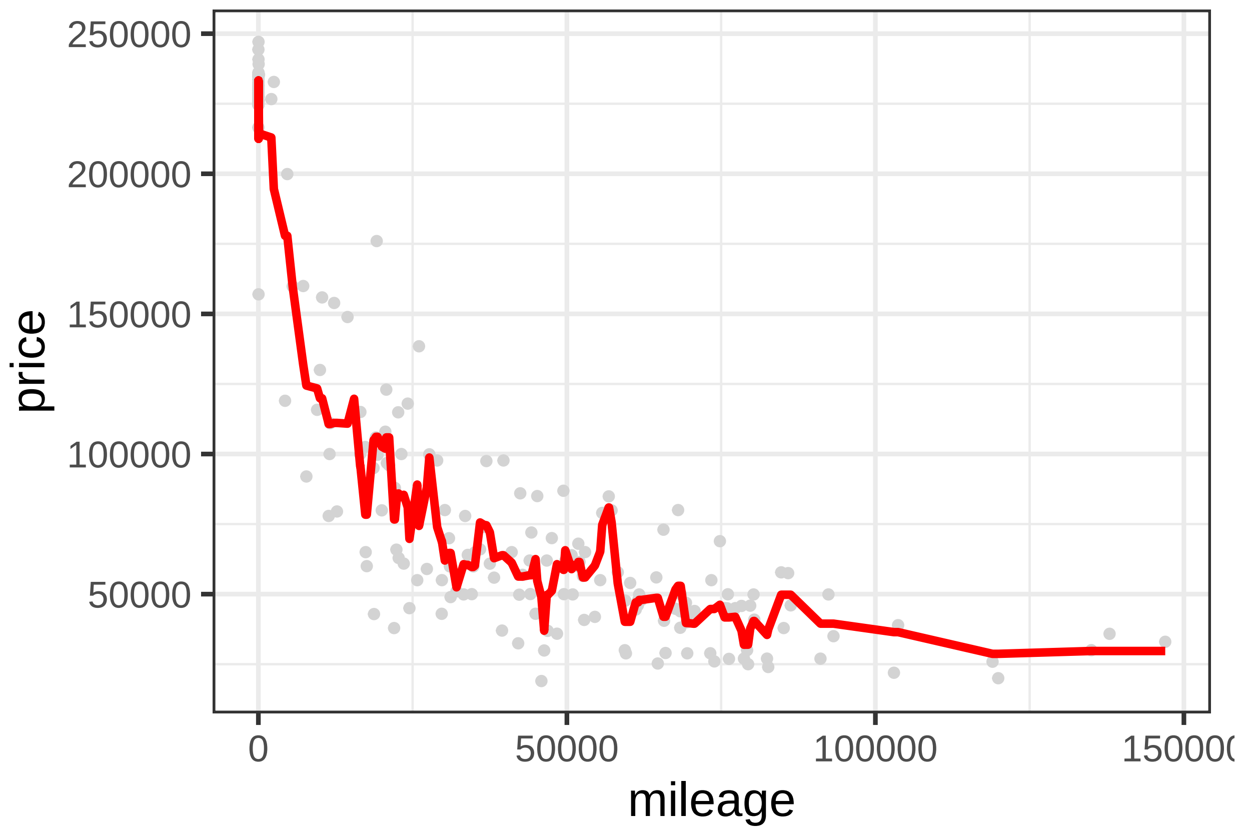
```r
#attach predictions to data frame
D_test$ypred_lm2 = ypred_lm2
D_test$ypred_knn4 = ypred_knn4

p_test = ggplot(data = D_test) +
  geom_point(mapping = aes(x = mileage, y = price), color='lightgrey') +
  theme_bw(base_size=18)
p_test
```




```r
p_test + geom_point(aes(x = mileage, y = ypred_knn4), color='red')
```




```
#KNN variances
knn_model = knn.reg(X_train, X_train, y_train, k = 4)

D_train$ypred = knn_model$pred
p_train = ggplot(data = D_train) +
  geom_point(mapping = aes(x = mileage, y = price), color='lightgrey') +
  theme_bw(base_size=18)
p_train + geom_path(mapping = aes(x=mileage, y=ypred), color='red', size=1.5)
```




```
#K = 5
lm1 = lm(price ~ mileage, data=D_train)
lm2 = lm(price ~ poly(mileage, 2), data=D_train)
knn5 = knn.reg(train = X_train, test = X_test, y = y_train, k=5)


#rmse
rmse = function(y, ypred) {
  sqrt(mean(data.matrix((y-ypred)^2)))
}

ypred_lm1 = predict(lm1, X_test)
ypred_lm2 = predict(lm2, X_test)
ypred_knn5 = knn5$pred

rmse(y_test, ypred_lm1)
```

```
## [1] 41419.01
```

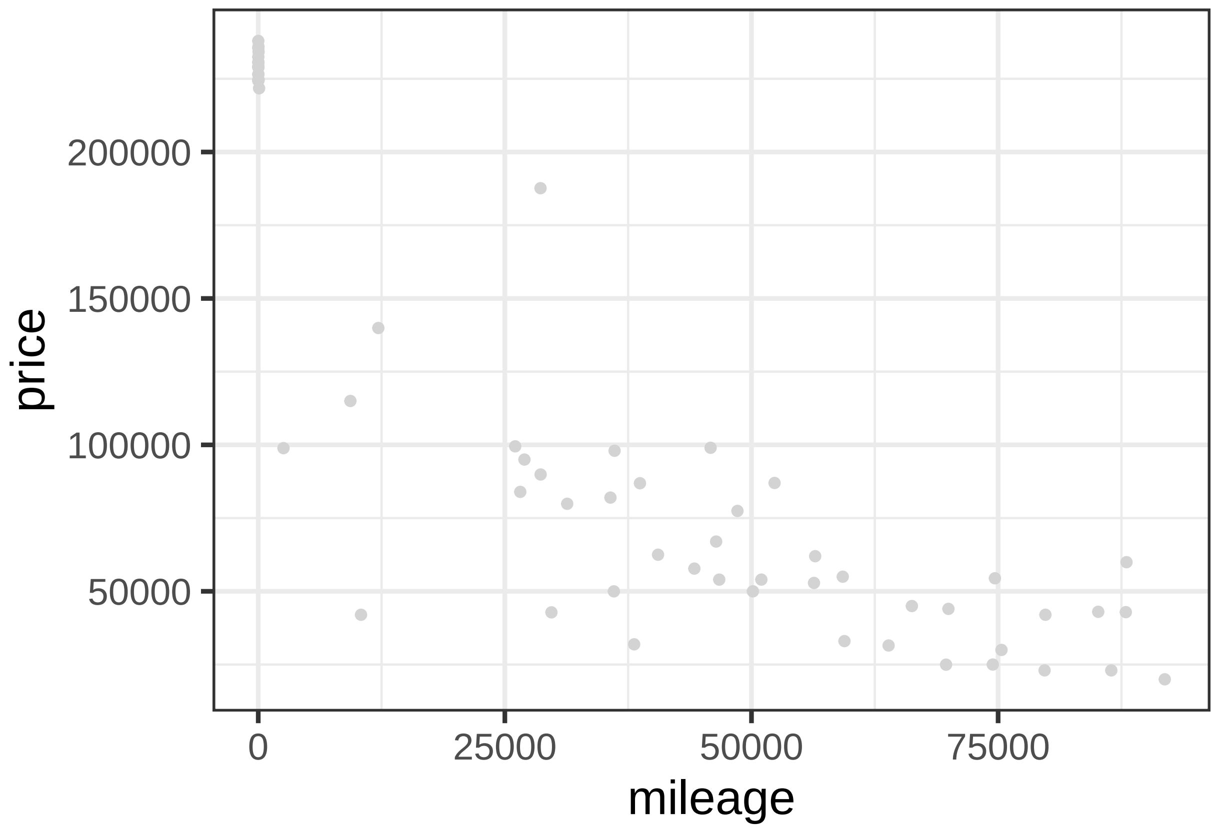
```
rmse(y_test, ypred_lm2)
```

```
## [1] 32029.86
```

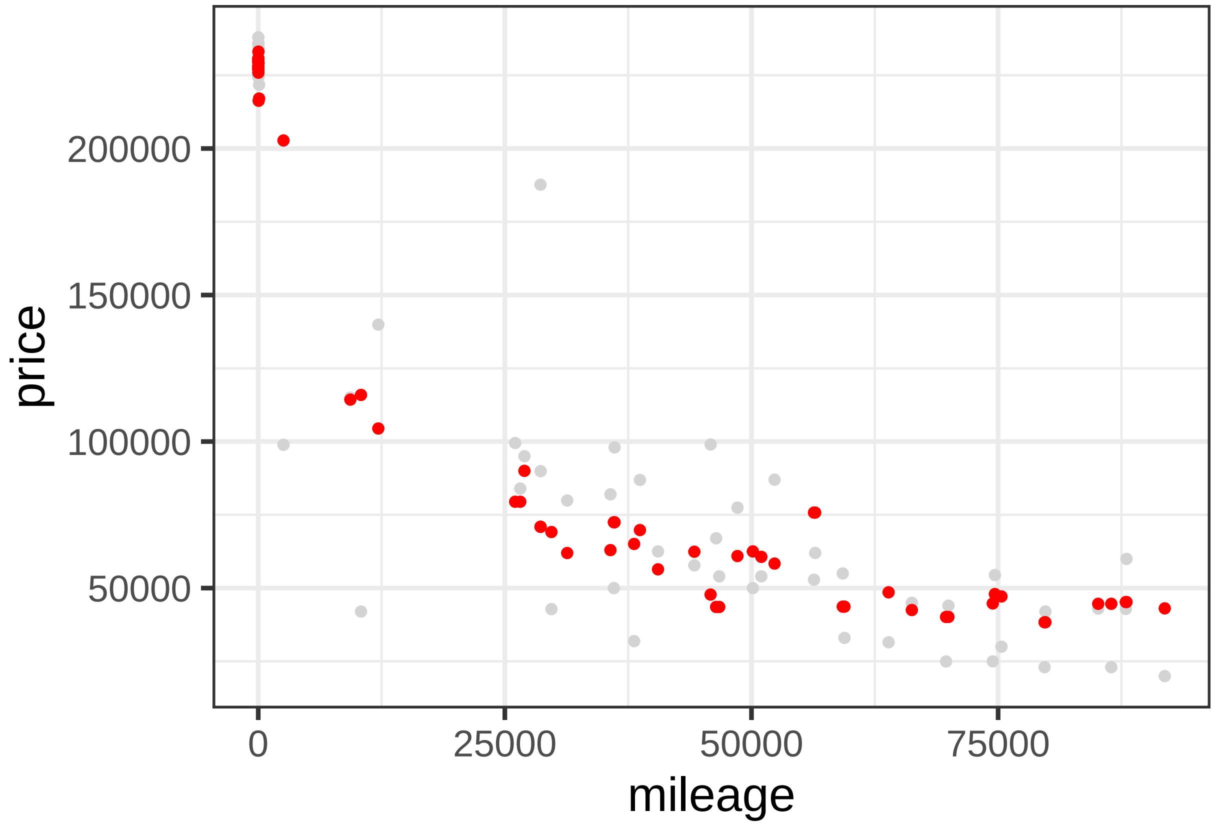
```r
rmse(y_test, ypred_knn5)
```

```
## [1] 27476
```

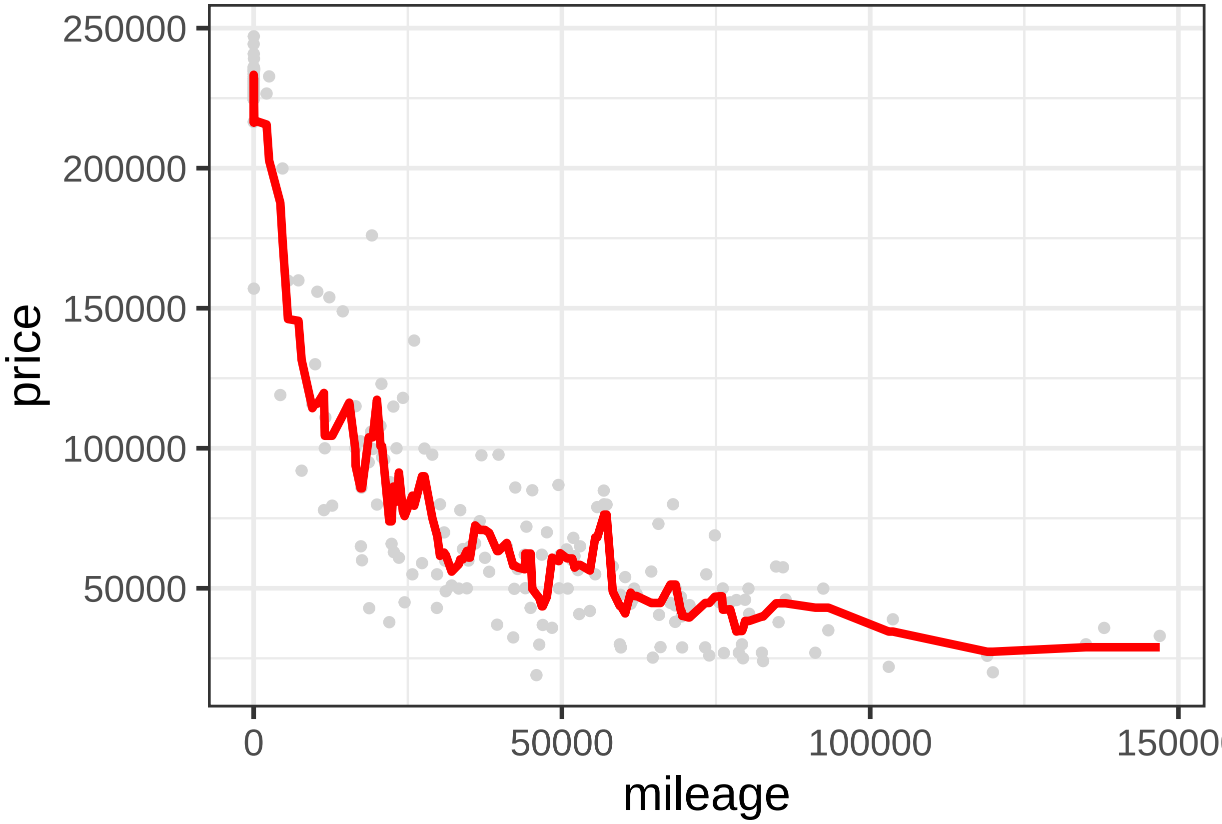
```r
#attach predictions to data frame
D_test$ypred_lm2 = ypred_lm2
D_test$ypred_knn5 = ypred_knn5

p_test = ggplot(data = D_test) +
  geom_point(mapping = aes(x = mileage, y = price), color='lightgrey') +
  theme_bw(base_size=18)
p_test
```




```r
p_test + geom_point(aes(x = mileage, y = ypred_knn5), color='red')
```




```
#KNN variances
knn_model = knn.reg(X_train, X_train, y_train, k = 5)

D_train$ypred = knn_model$pred
p_train = ggplot(data = D_train) +
  geom_point(mapping = aes(x = mileage, y = price), color='lightgrey') +
  theme_bw(base_size=18)
p_train + geom_path(mapping = aes(x=mileage, y=ypred), color='red', size=1.5)
```




```
#K = 6
lm1 = lm(price ~ mileage, data=D_train)
lm2 = lm(price ~ poly(mileage, 2), data=D_train)
knn6 = knn.reg(train = X_train, test = X_test, y = y_train, k=6)


#rmse
rmse = function(y, ypred) {
  sqrt(mean(data.matrix((y-ypred)^2)))
}

ypred_lm1 = predict(lm1, X_test)
ypred_lm2 = predict(lm2, X_test)
ypred_knn6 = knn6$pred

rmse(y_test, ypred_lm1)
```

```
## [1] 41419.01
```

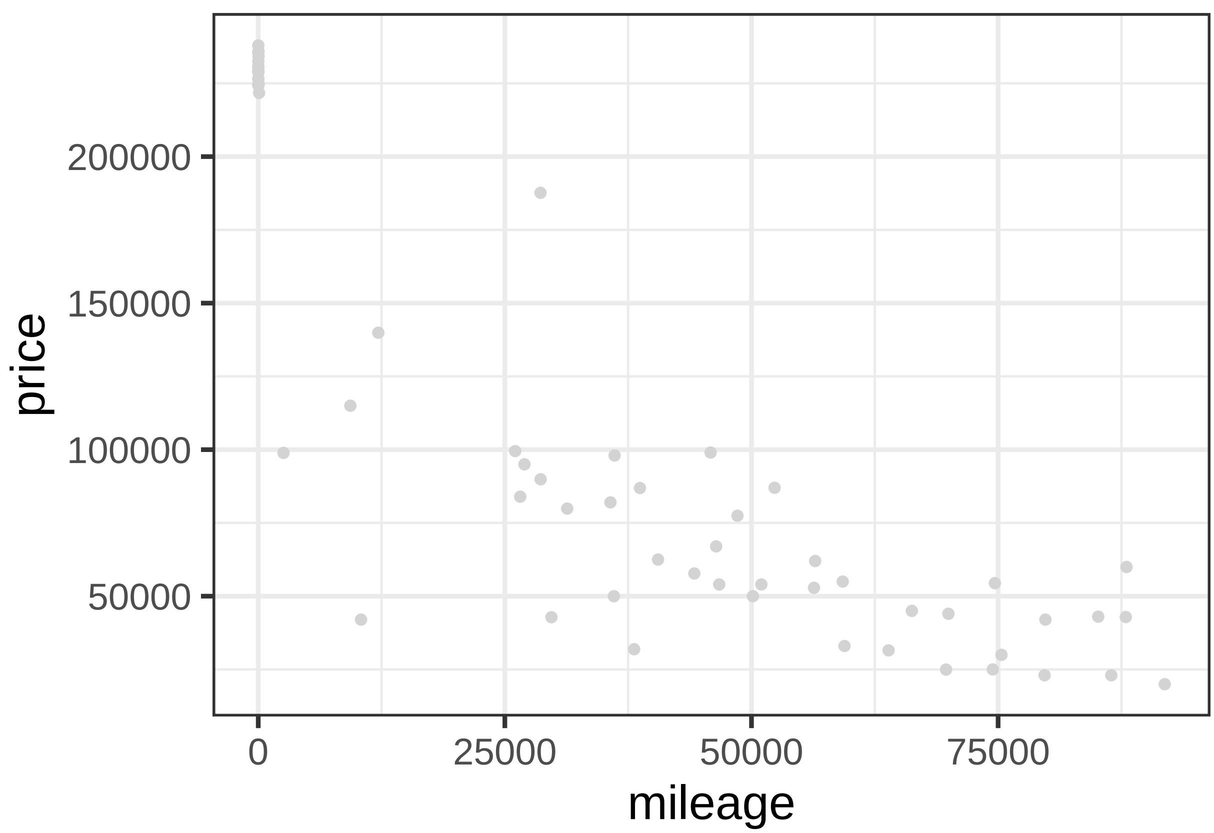
```
rmse(y_test, ypred_lm2)
```

```
## [1] 32029.86
```

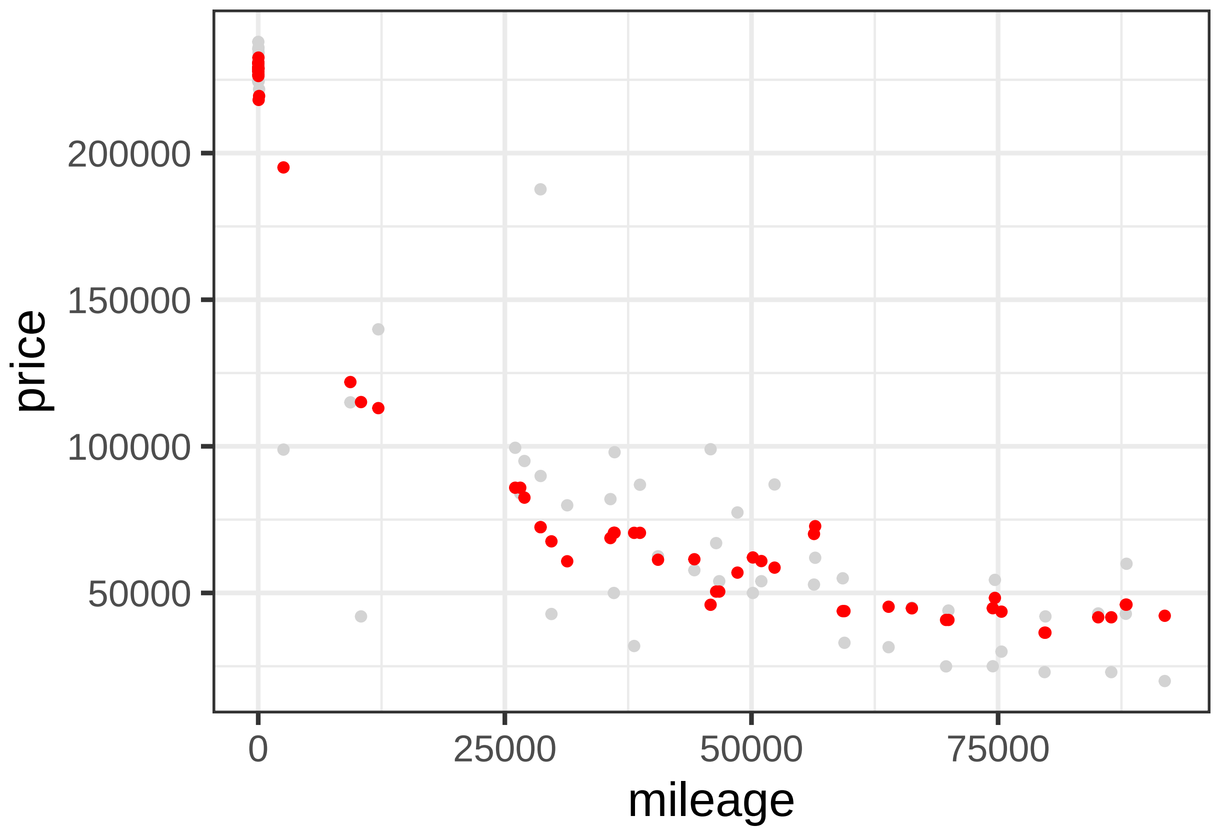
```r
rmse(y_test, ypred_knn6)
```

```
## [1] 26429.08
```

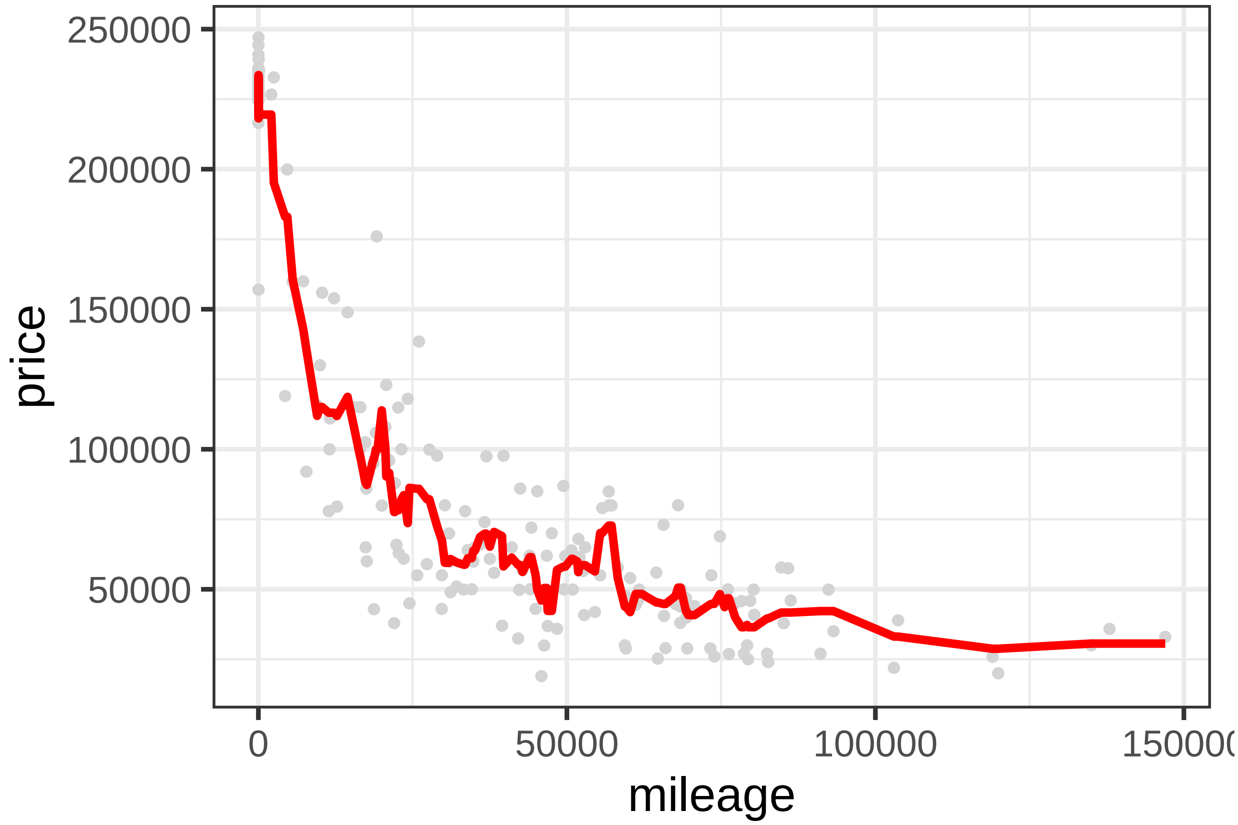
```r
#attach predictions to data frame
D_test$ypred_lm2 = ypred_lm2
D_test$ypred_knn6 = ypred_knn6

p_test = ggplot(data = D_test) +
  geom_point(mapping = aes(x = mileage, y = price), color='lightgrey') +
  theme_bw(base_size=18)
p_test
```




```r
p_test + geom_point(aes(x = mileage, y = ypred_knn6), color='red')
```




```
#KNN variances
knn_model = knn.reg(X_train, X_train, y_train, k = 6)

D_train$ypred = knn_model$pred
p_train = ggplot(data = D_train) +
  geom_point(mapping = aes(x = mileage, y = price), color='lightgrey') +
  theme_bw(base_size=18)
p_train + geom_path(mapping = aes(x=mileage, y=ypred), color='red', size=1.5)
```




```r
#K = 7
lm1 = lm(price ~ mileage, data=D_train)
lm2 = lm(price ~ poly(mileage, 2), data=D_train)
knn7 = knn.reg(train = X_train, test = X_test, y = y_train, k=7)


#rmse
rmse = function(y, ypred) {
  sqrt(mean(data.matrix((y-ypred)^2)))
}

ypred_lm1 = predict(lm1, X_test)
ypred_lm2 = predict(lm2, X_test)
ypred_knn7 = knn7$pred

rmse(y_test, ypred_lm1)
```

```
## [1] 41419.01
```

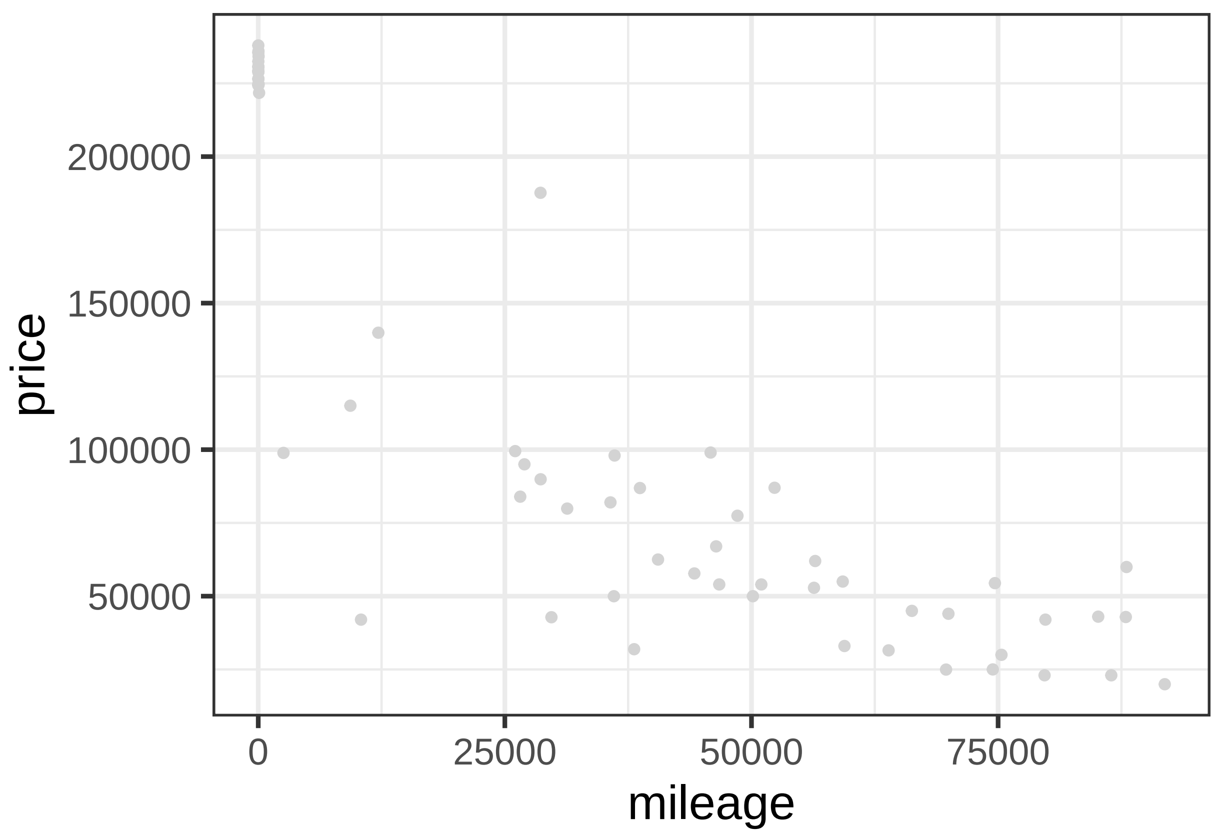
```r
rmse(y_test, ypred_lm2)
```

```
## [1] 32029.86
```

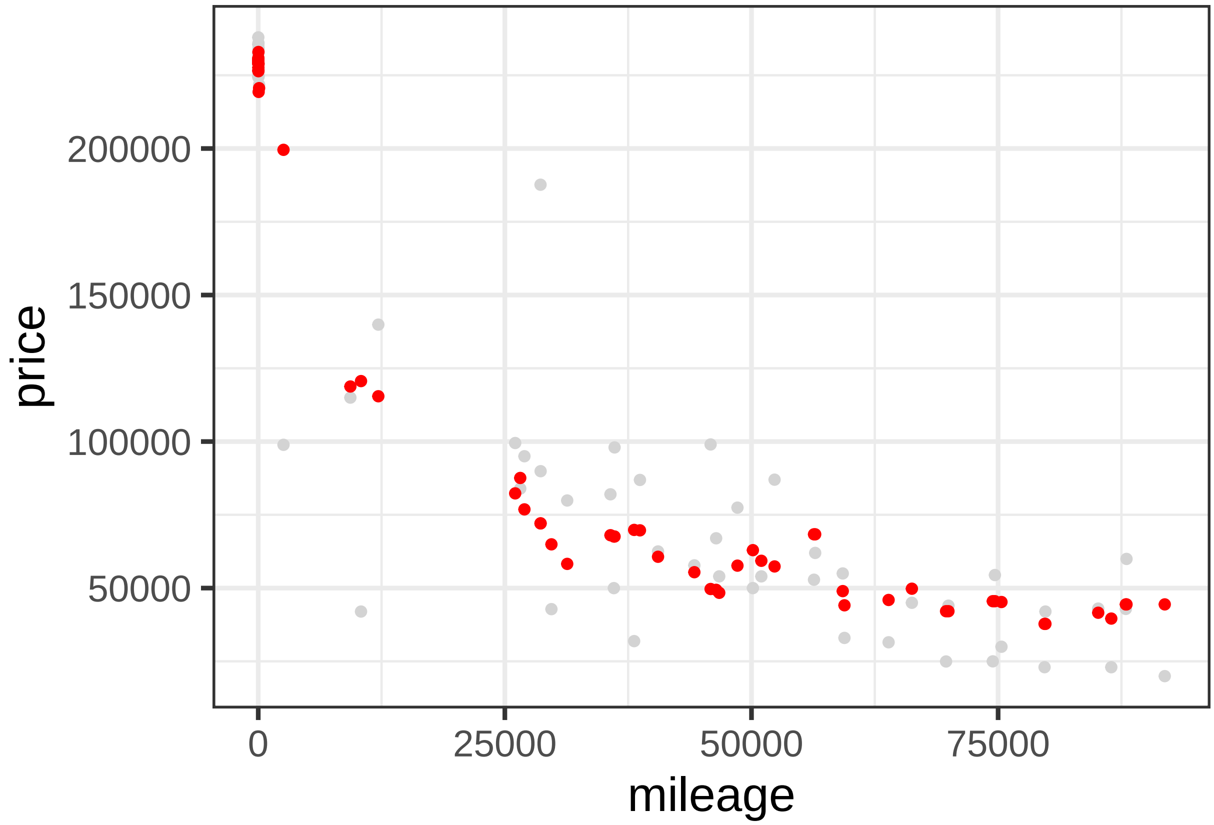
```r
rmse(y_test, ypred_knn7)
```

```
## [1] 26978.81
```

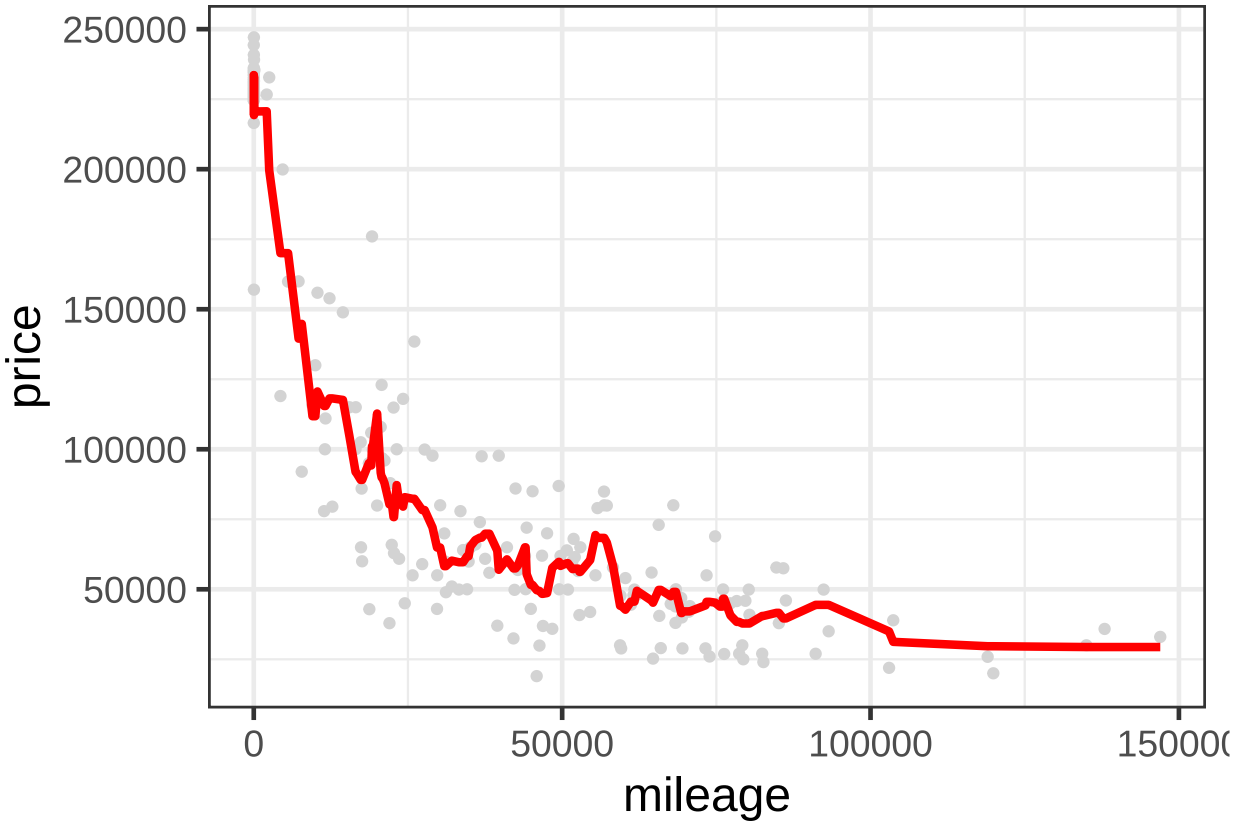
```r
#attach predictions to data frame
D_test$ypred_lm2 = ypred_lm2
D_test$ypred_knn7 = ypred_knn7

p_test = ggplot(data = D_test) +
  geom_point(mapping = aes(x = mileage, y = price), color='lightgrey') +
  theme_bw(base_size=18)
p_test
```




```r
p_test + geom_point(aes(x = mileage, y = ypred_knn7), color='red')
```




```
#KNN variances
knn_model = knn.reg(X_train, X_train, y_train, k = 7)

D_train$ypred = knn_model$pred
p_train = ggplot(data = D_train) +
  geom_point(mapping = aes(x = mileage, y = price), color='lightgrey') +
  theme_bw(base_size=18)
p_train + geom_path(mapping = aes(x=mileage, y=ypred), color='red', size=1.5)
```




```r
#K = 15
lm1 = lm(price ~ mileage, data=D_train)
lm2 = lm(price ~ poly(mileage, 2), data=D_train)
knn15 = knn.reg(train = X_train, test = X_test, y = y_train, k=15)


#rmse
rmse = function(y, ypred) {
  sqrt(mean(data.matrix((y-ypred)^2)))
}

ypred_lm1 = predict(lm1, X_test)
ypred_lm2 = predict(lm2, X_test)
ypred_knn15 = knn15$pred

rmse(y_test, ypred_lm1)
```

```
## [1] 41419.01
```

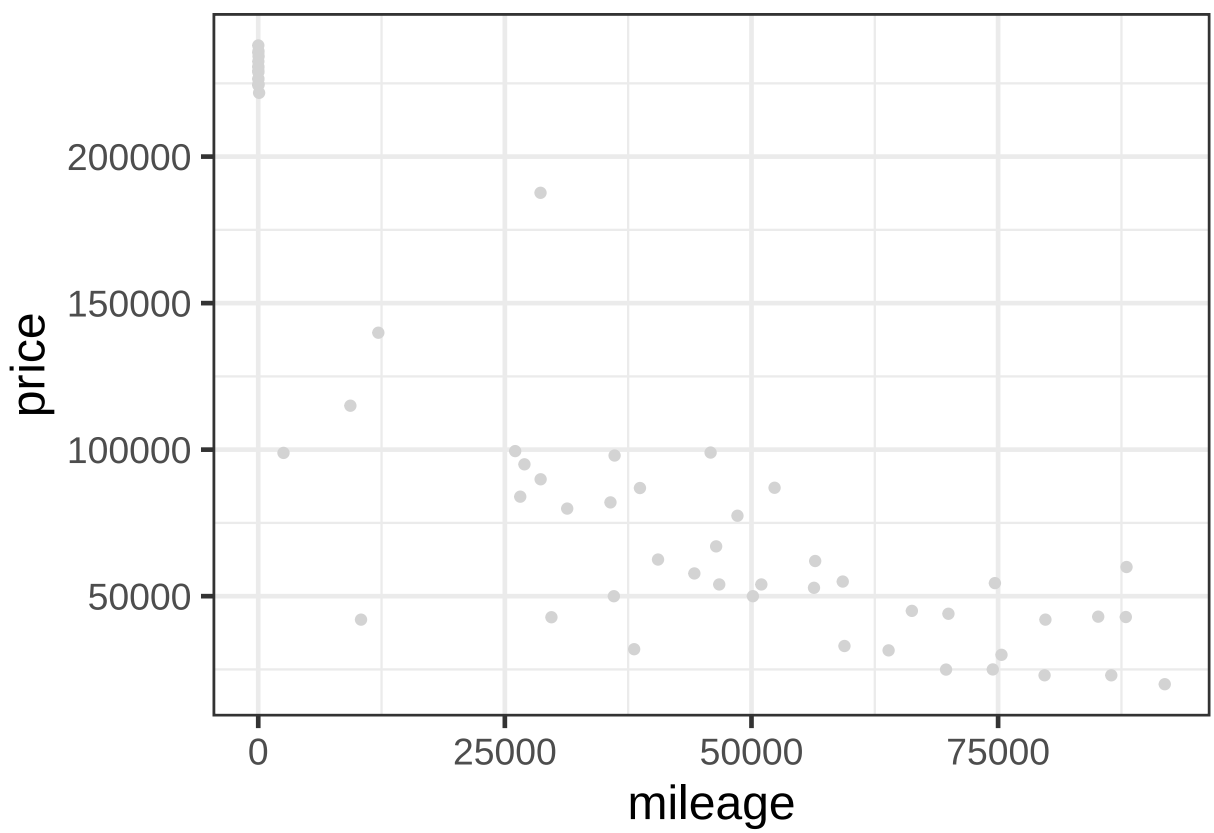
```r
rmse(y_test, ypred_lm2)
```

```
## [1] 32029.86
```

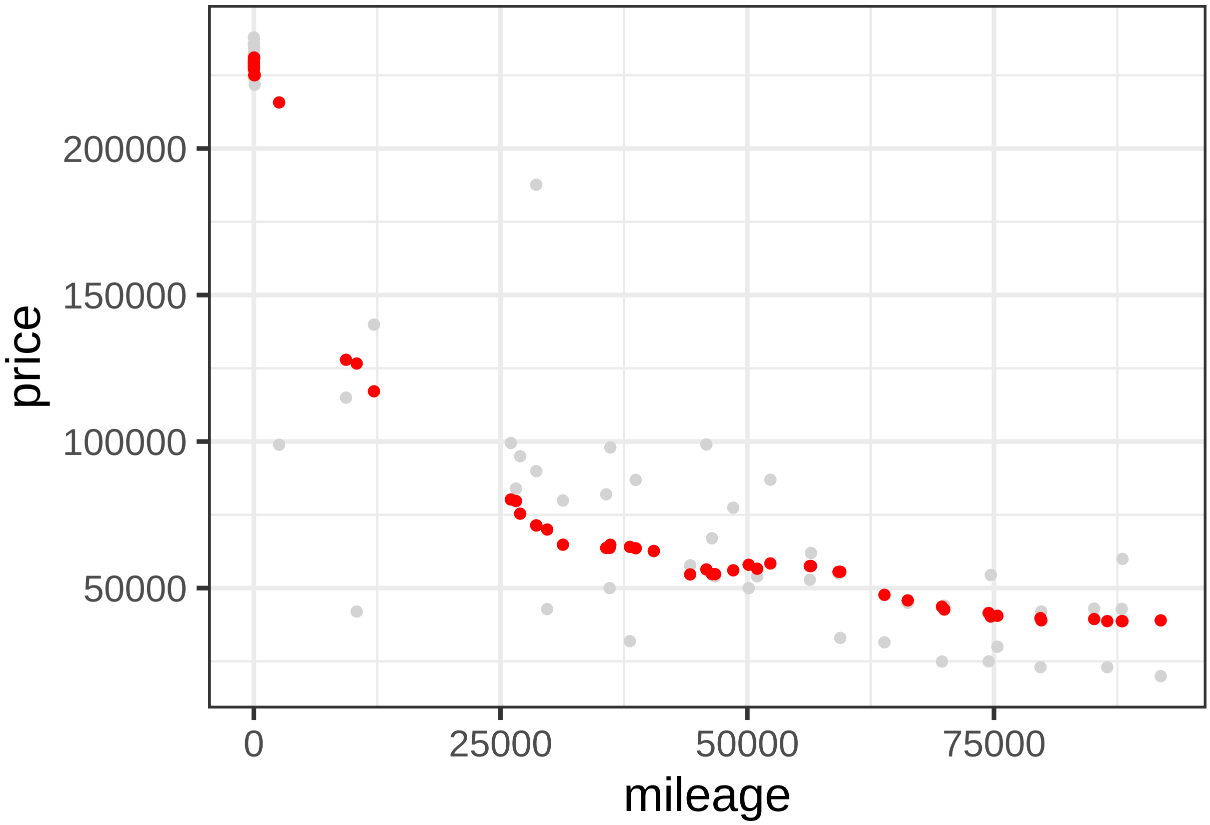
```r
rmse(y_test, ypred_knn15)
```

```
## [1] 28227.4
```

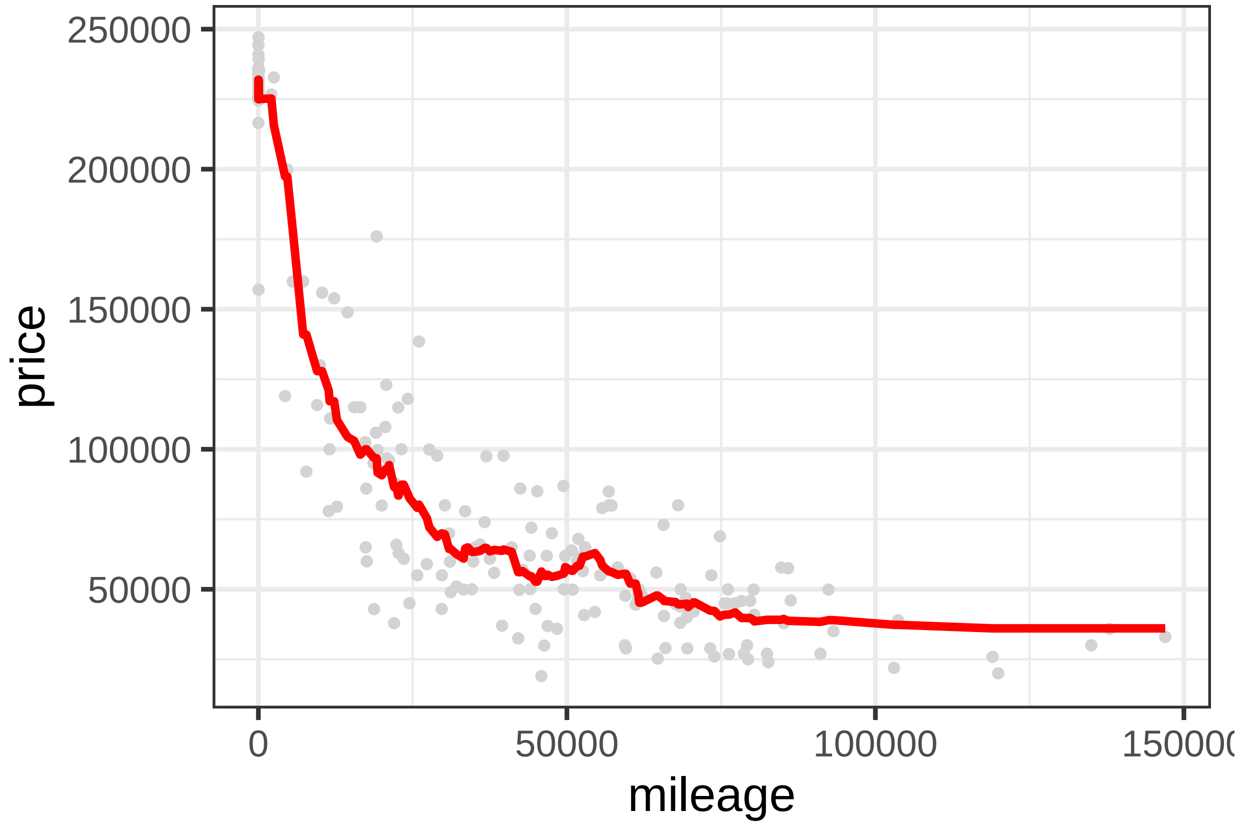
```r
#attach predictions to data frame
D_test$ypred_lm2 = ypred_lm2
D_test$ypred_knn15 = ypred_knn15

p_test = ggplot(data = D_test) +
  geom_point(mapping = aes(x = mileage, y = price), color='lightgrey') +
  theme_bw(base_size=18)
p_test
```




```r
p_test + geom_point(aes(x = mileage, y = ypred_knn15), color='red')
```




```
#KNN variances
knn_model = knn.reg(X_train, X_train, y_train, k = 15)

D_train$ypred = knn_model$pred
p_train = ggplot(data = D_train) +
  geom_point(mapping = aes(x = mileage, y = price), color='lightgrey') +
  theme_bw(base_size=18)
p_train + geom_path(mapping = aes(x=mileage, y=ypred), color='red', size=1.5)
```




```
#K = 30
lm1 = lm(price ~ mileage, data=D_train)
lm2 = lm(price ~ poly(mileage, 2), data=D_train)
knn30 = knn.reg(train = X_train, test = X_test, y = y_train, k=30)


#rmse
rmse = function(y, ypred) {
  sqrt(mean(data.matrix((y-ypred)^2)))
}

ypred_lm1 = predict(lm1, X_test)
ypred_lm2 = predict(lm2, X_test)
ypred_knn30 = knn30$pred

rmse(y_test, ypred_lm1)
```

```
## [1] 41419.01
```

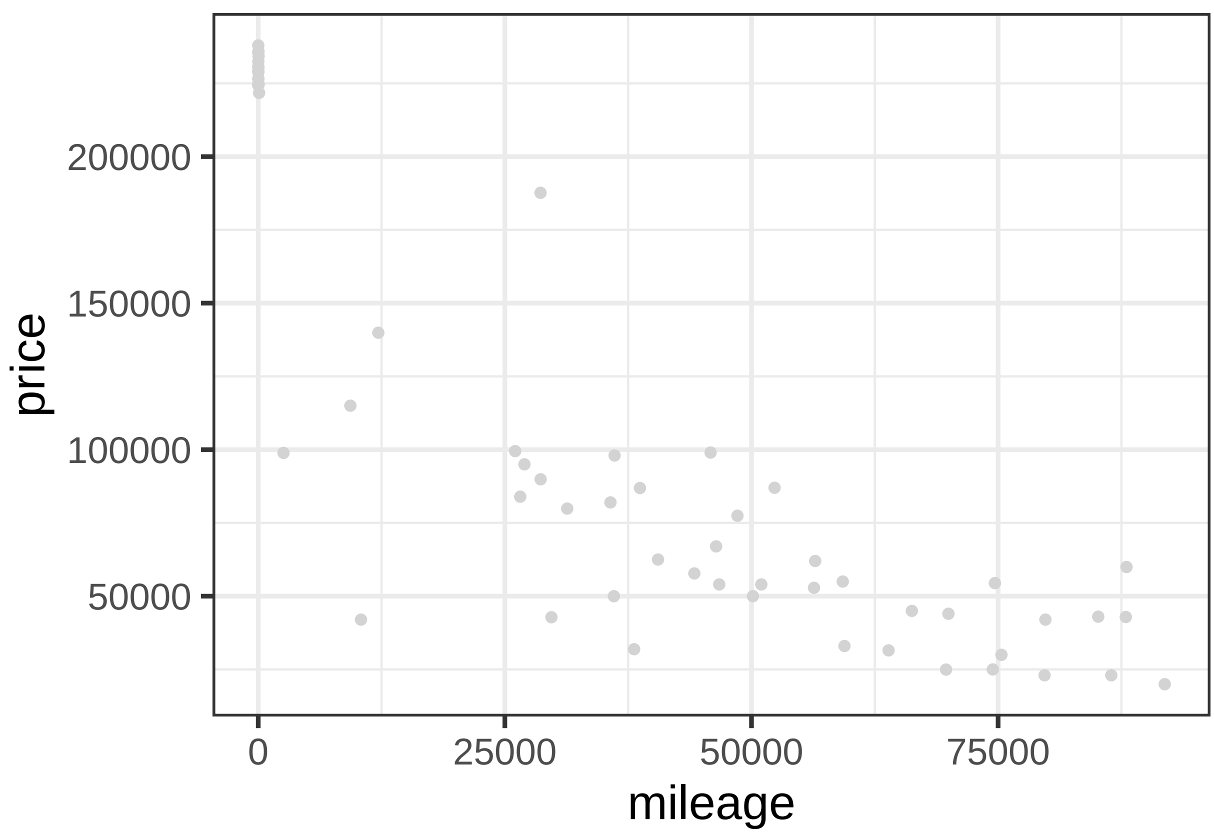
```
rmse(y_test, ypred_lm2)
```

```
## [1] 32029.86
```

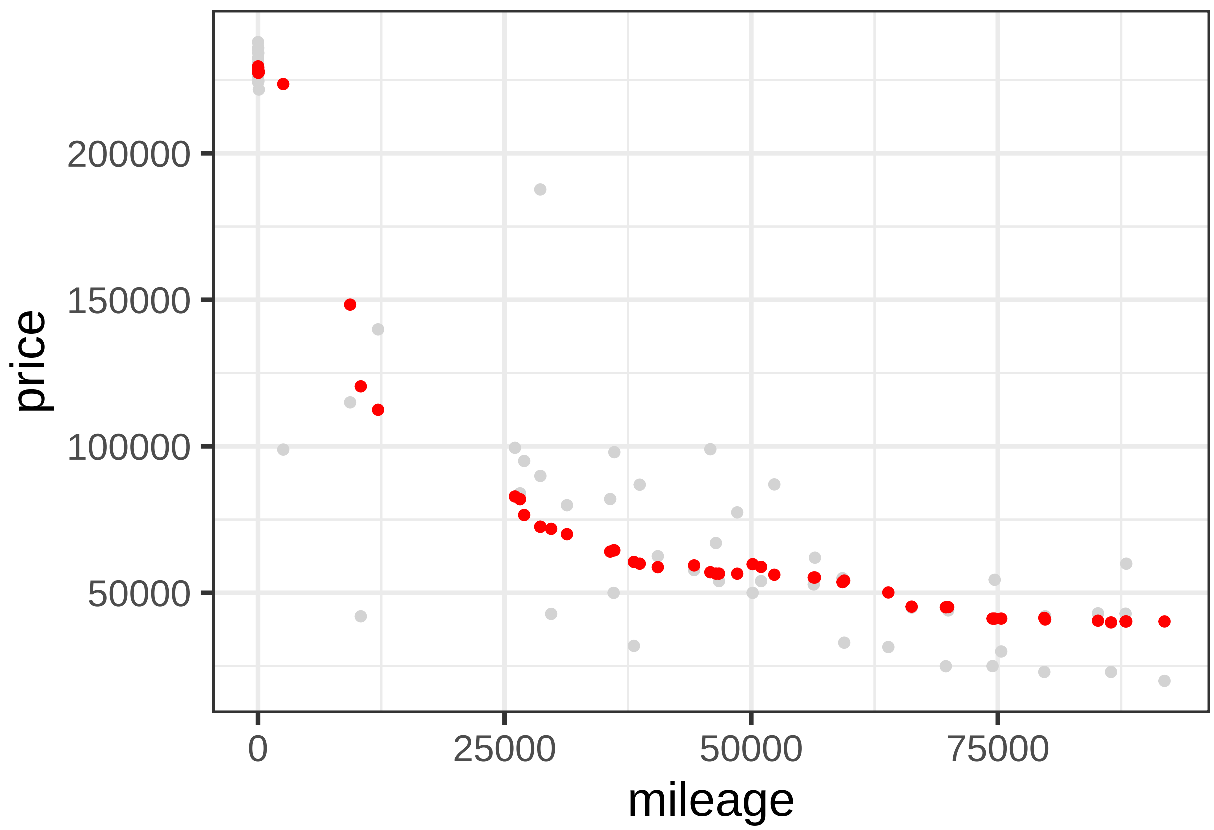
```r
rmse(y_test, ypred_knn30)
```

```
## [1] 28785.16
```

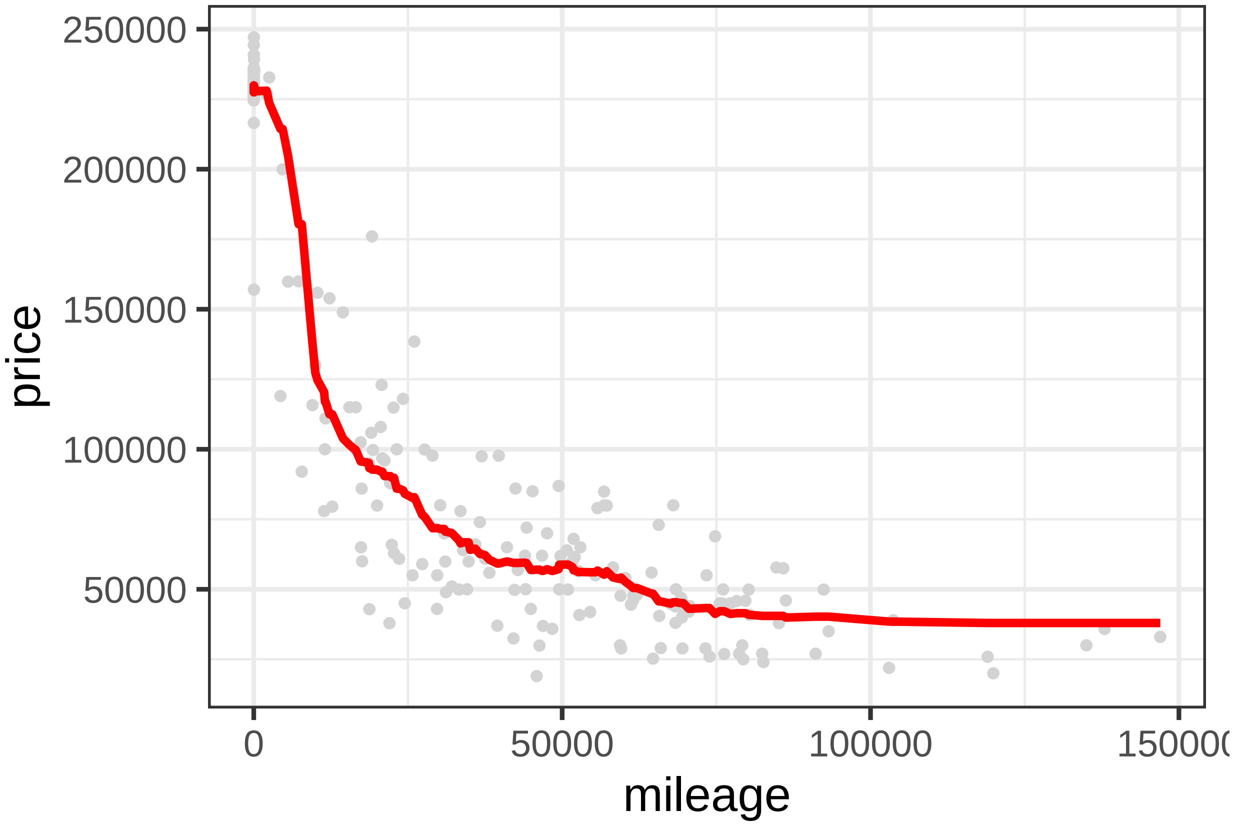
```r
#attach predictions to data frame
D_test$ypred_lm2 = ypred_lm2
D_test$ypred_knn30 = ypred_knn30

p_test = ggplot(data = D_test) +
  geom_point(mapping = aes(x = mileage, y = price), color='lightgrey') +
  theme_bw(base_size=18)
p_test
```




```r
p_test + geom_point(aes(x = mileage, y = ypred_knn30), color='red')
```




```r
#KNN variances
knn_model = knn.reg(X_train, X_train, y_train, k = 30)

D_train$ypred = knn_model$pred
p_train = ggplot(data = D_train) +
  geom_point(mapping = aes(x = mileage, y = price), color='lightgrey') +
  theme_bw(base_size=18)
p_train + geom_path(mapping = aes(x=mileage, y=ypred), color='red', size=1.5)
```




```
#K = 50
lm1 = lm(price ~ mileage, data=D_train)
lm2 = lm(price ~ poly(mileage, 2), data=D_train)
knn50 = knn.reg(train = X_train, test = X_test, y = y_train, k=50)


#rmse
rmse = function(y, ypred) {
  sqrt(mean(data.matrix((y-ypred)^2)))
}

ypred_lm1 = predict(lm1, X_test)
ypred_lm2 = predict(lm2, X_test)
ypred_knn50 = knn50$pred

rmse(y_test, ypred_lm1)
```

```
## [1] 41419.01
```

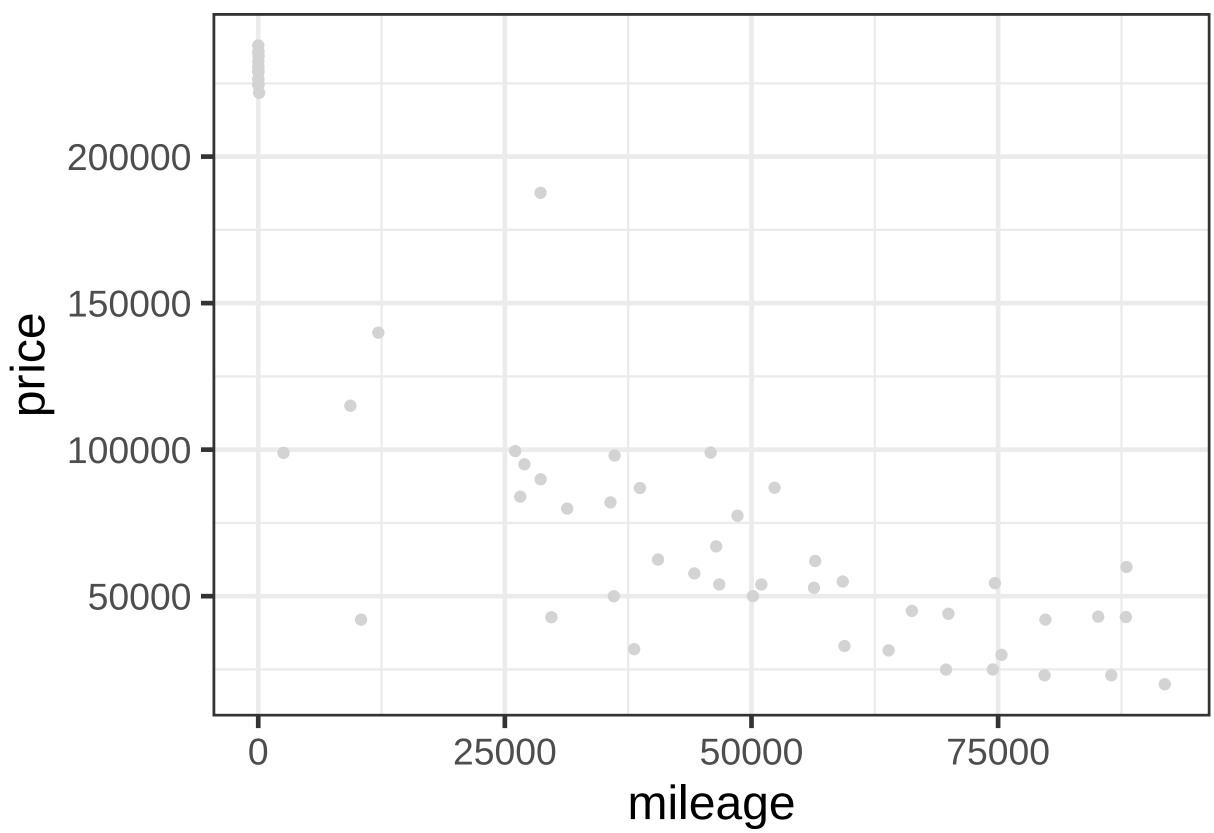
```
rmse(y_test, ypred_lm2)
```

```
## [1] 32029.86
```

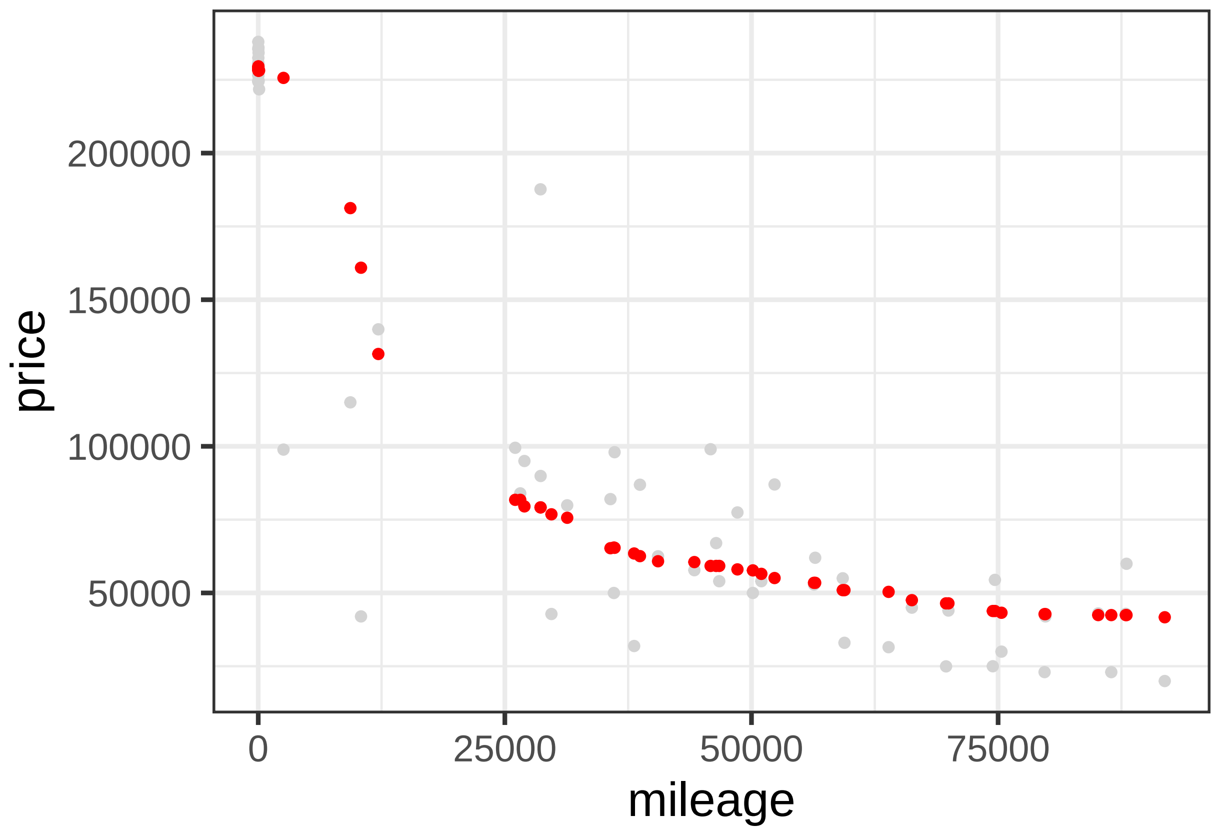
```r
rmse(y_test, ypred_knn50)
```

```
## [1] 31457.84
```

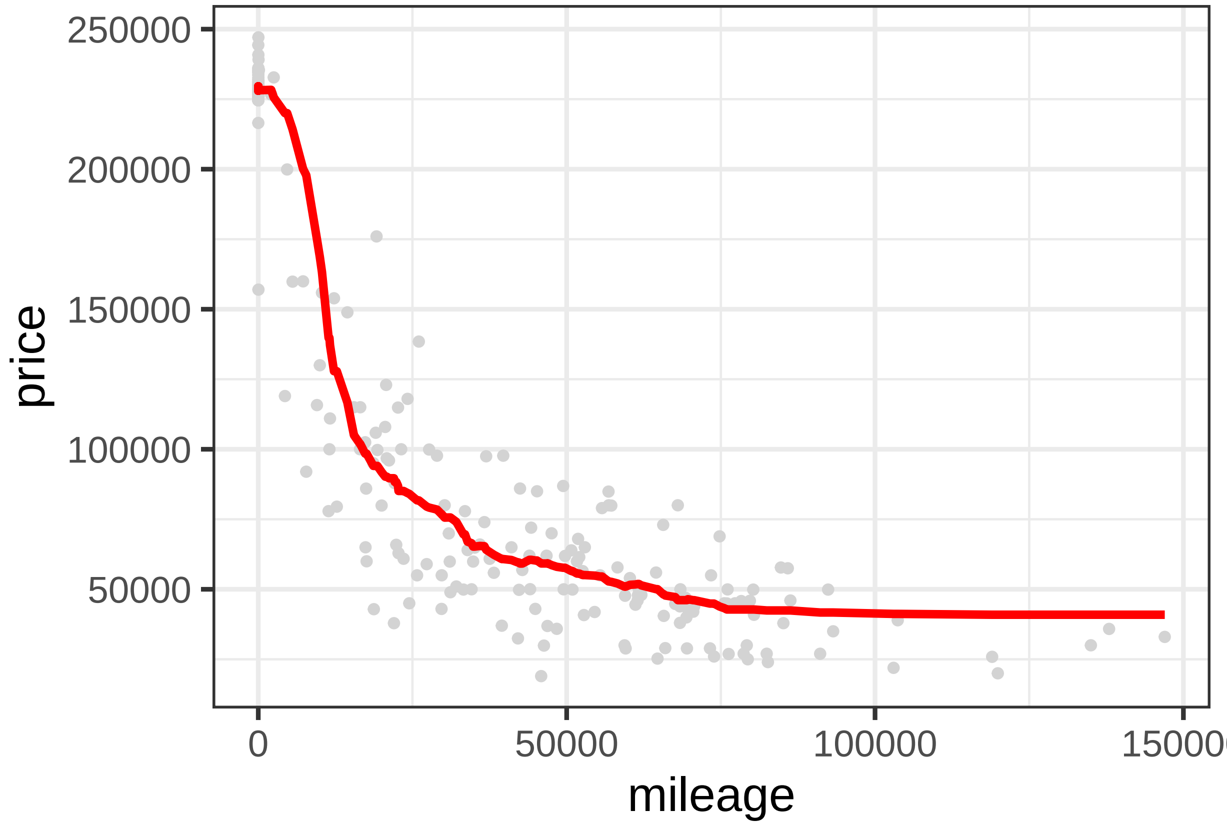
```r
#attach predictions to data frame
D_test$ypred_lm2 = ypred_lm2
D_test$ypred_knn50 = ypred_knn50

p_test = ggplot(data = D_test) +
  geom_point(mapping = aes(x = mileage, y = price), color='lightgrey') +
  theme_bw(base_size=18)
p_test
```




```r
p_test + geom_point(aes(x = mileage, y = ypred_knn50), color='red')
```




```
#KNN variances
knn_model = knn.reg(X_train, X_train, y_train, k = 50)

D_train$ypred = knn_model$pred
p_train = ggplot(data = D_train) +
  geom_point(mapping = aes(x = mileage, y = price), color='lightgrey') +
  theme_bw(base_size=18)
p_train + geom_path(mapping = aes(x=mileage, y=ypred), color='red', size=1.5)
```




```
#K = 60
lm1 = lm(price ~ mileage, data=D_train)
lm2 = lm(price ~ poly(mileage, 2), data=D_train)
knn60 = knn.reg(train = X_train, test = X_test, y = y_train, k=60)


#rmse
rmse = function(y, ypred) {
  sqrt(mean(data.matrix((y-ypred)^2)))
}

ypred_lm1 = predict(lm1, X_test)
ypred_lm2 = predict(lm2, X_test)
ypred_knn60 = knn60$pred

rmse(y_test, ypred_lm1)
```

```
## [1] 41419.01
```

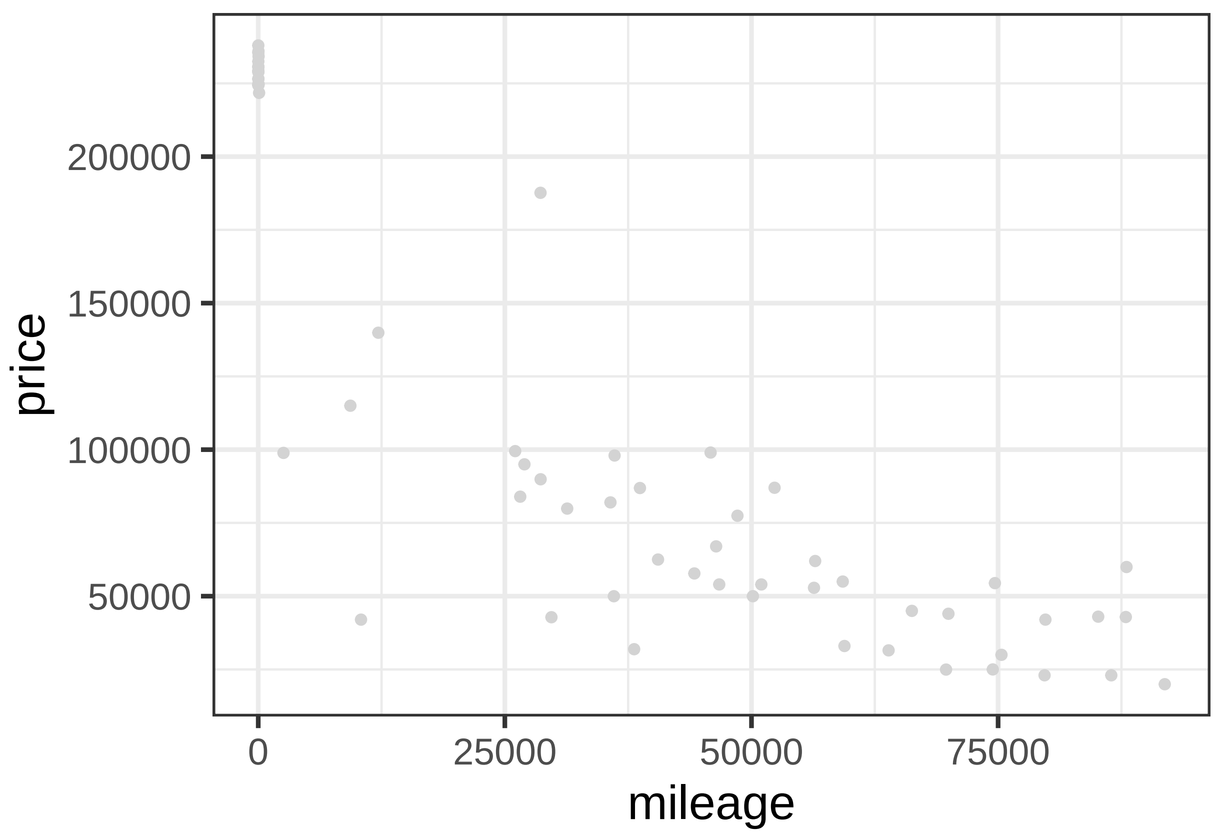
```
rmse(y_test, ypred_lm2)
```

```
## [1] 32029.86
```

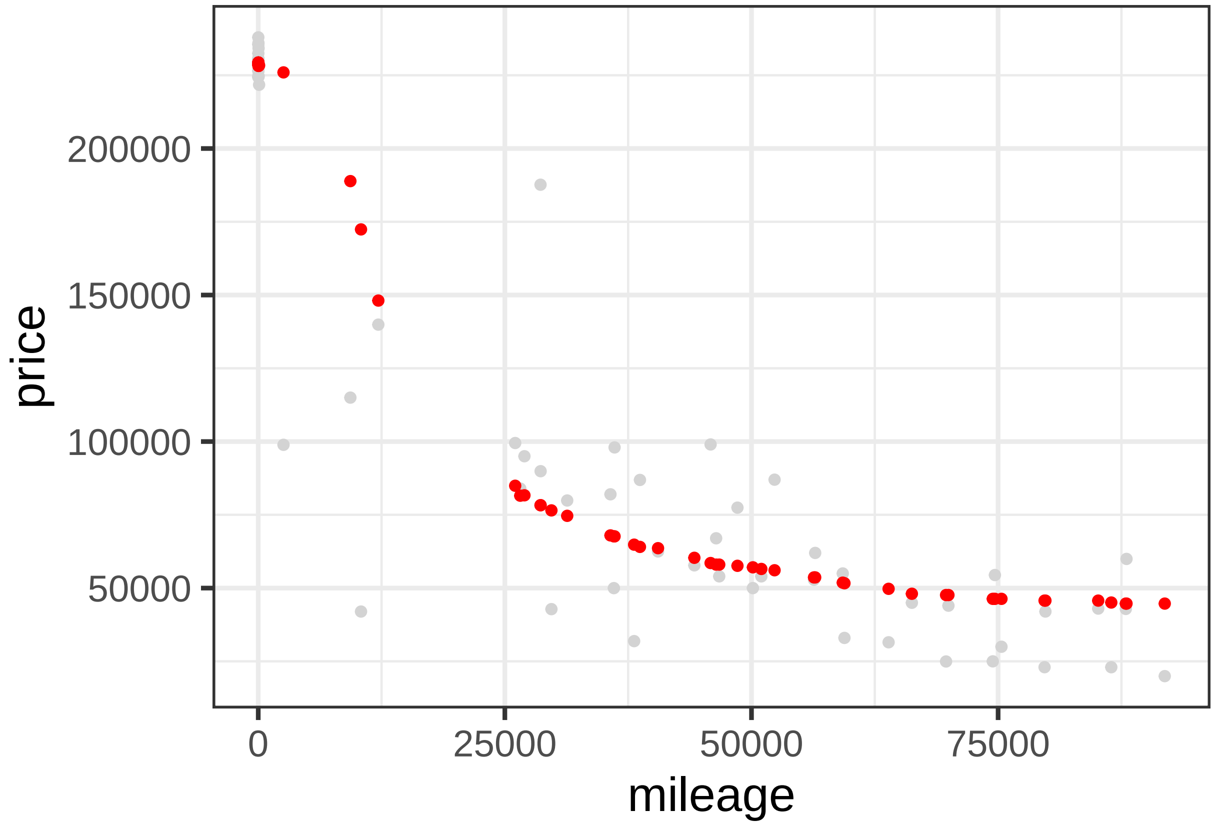
```r
rmse(y_test, ypred_knn60)
```

```
## [1] 32632.44
```

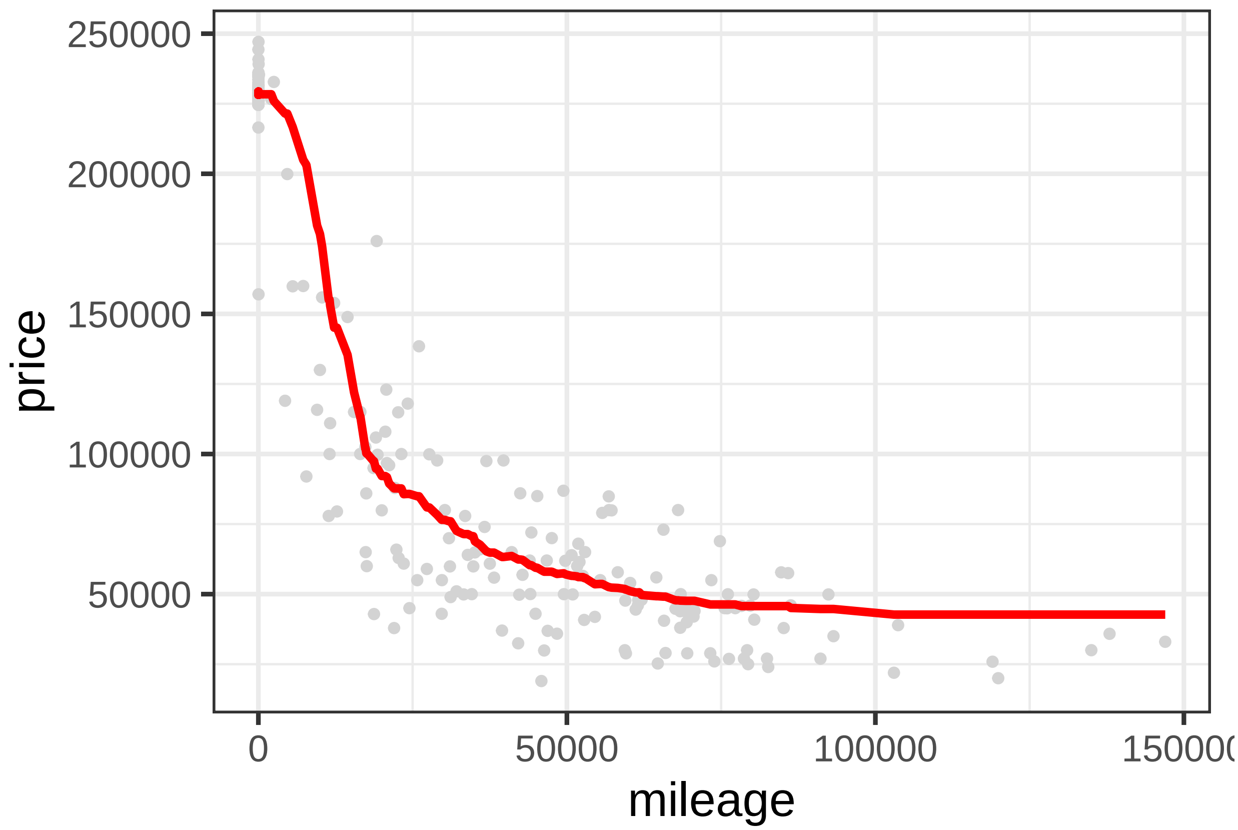
```r
#attach predictions to data frame
D_test$ypred_lm2 = ypred_lm2
D_test$ypred_knn60 = ypred_knn60

p_test = ggplot(data = D_test) +
  geom_point(mapping = aes(x = mileage, y = price), color='lightgrey') +
  theme_bw(base_size=18)
p_test
```




```r
p_test + geom_point(aes(x = mileage, y = ypred_knn60), color='red')
```




```
#KNN variances
knn_model = knn.reg(X_train, X_train, y_train, k = 60)

D_train$ypred = knn_model$pred
p_train = ggplot(data = D_train) +
  geom_point(mapping = aes(x = mileage, y = price), color='lightgrey') +
  theme_bw(base_size=18)
p_train + geom_path(mapping = aes(x=mileage, y=ypred), color='red', size=1.5)
```




```r
#K = 80
lm1 = lm(price ~ mileage, data=D_train)
lm2 = lm(price ~ poly(mileage, 2), data=D_train)
knn80 = knn.reg(train = X_train, test = X_test, y = y_train, k=80)


#rmse
rmse = function(y, ypred) {
  sqrt(mean(data.matrix((y-ypred)^2)))
}

ypred_lm1 = predict(lm1, X_test)
ypred_lm2 = predict(lm2, X_test)
ypred_knn80 = knn80$pred

rmse(y_test, ypred_lm1)
```

```
## [1] 41419.01
```

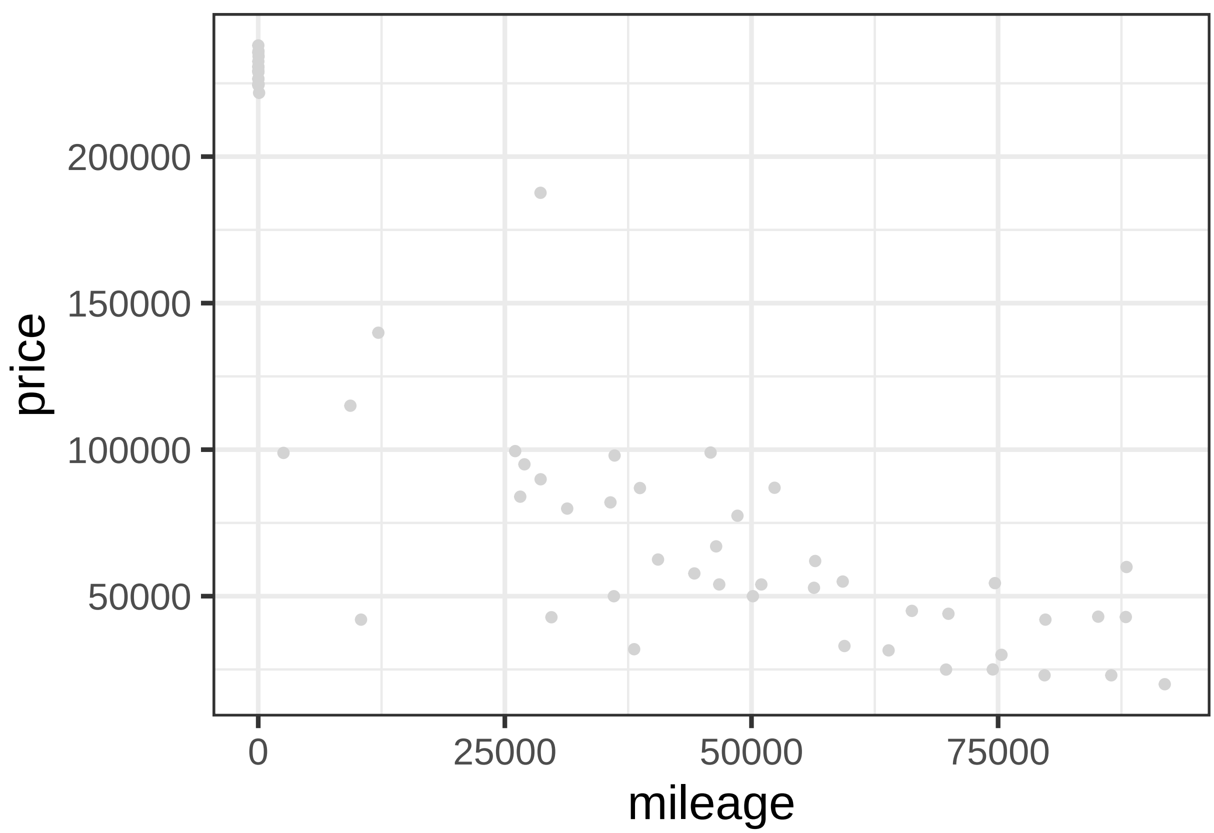
```r
rmse(y_test, ypred_lm2)
```

```
## [1] 32029.86
```

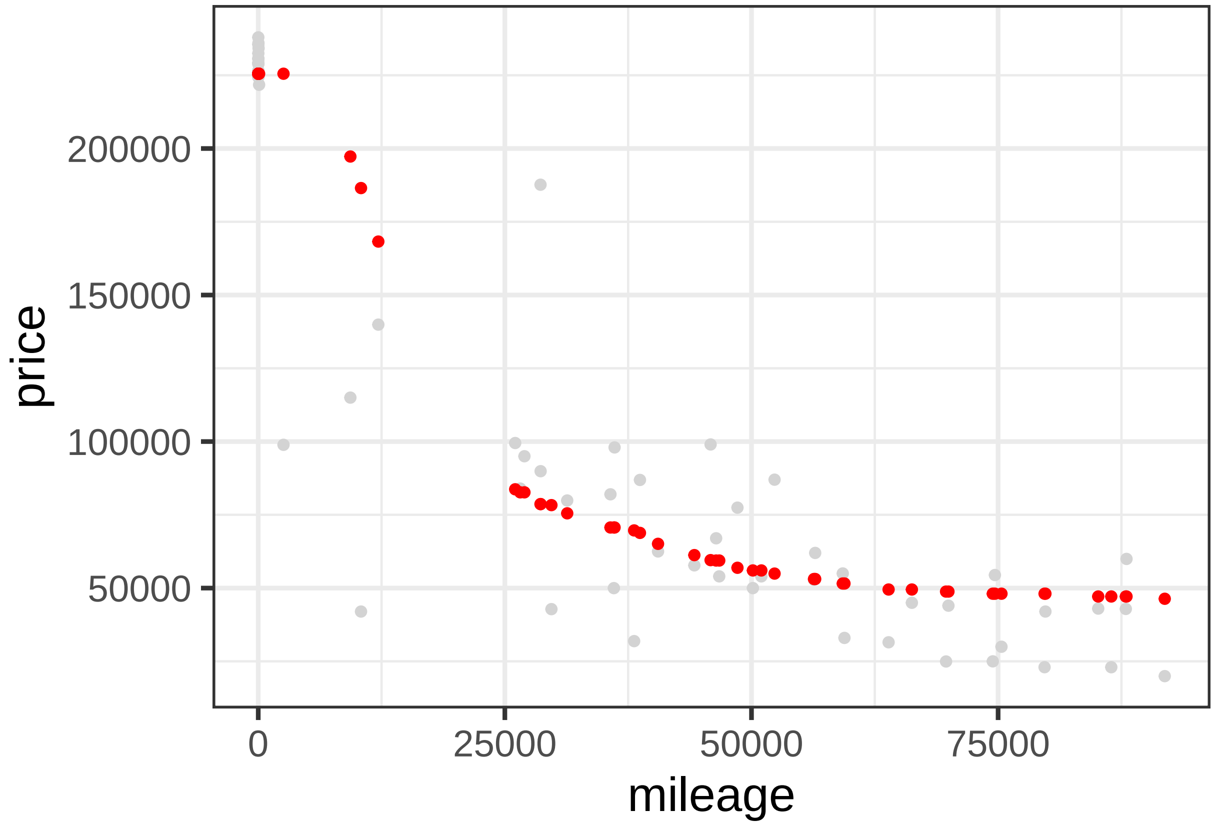
```
rmse(y_test, ypred_knn80)
```

```
## [1] 34297.55
```

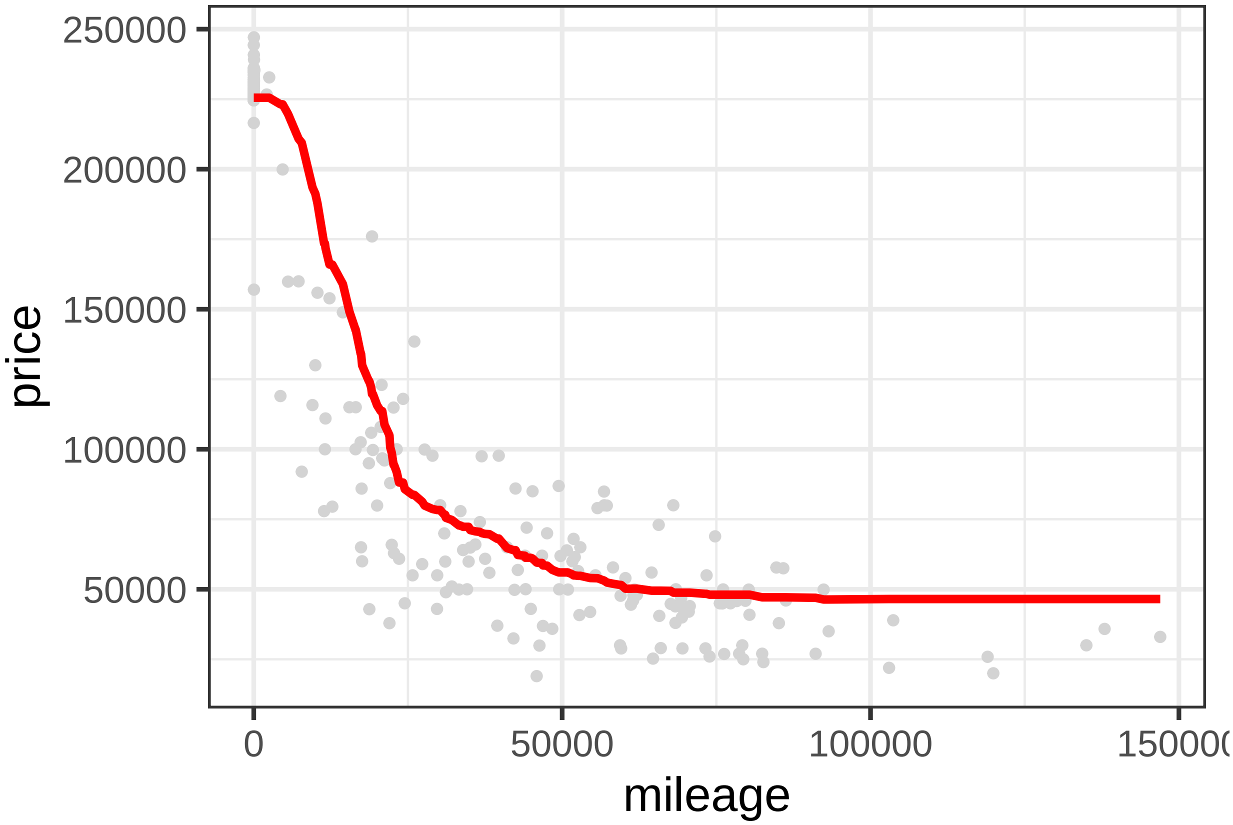
```
#attach predictions to data frame
D_test$ypred_lm2 = ypred_lm2
D_test$ypred_knn80 = ypred_knn80

p_test = ggplot(data = D_test) +
  geom_point(mapping = aes(x = mileage, y = price), color='lightgrey') +
  theme_bw(base_size=18)
p_test
```



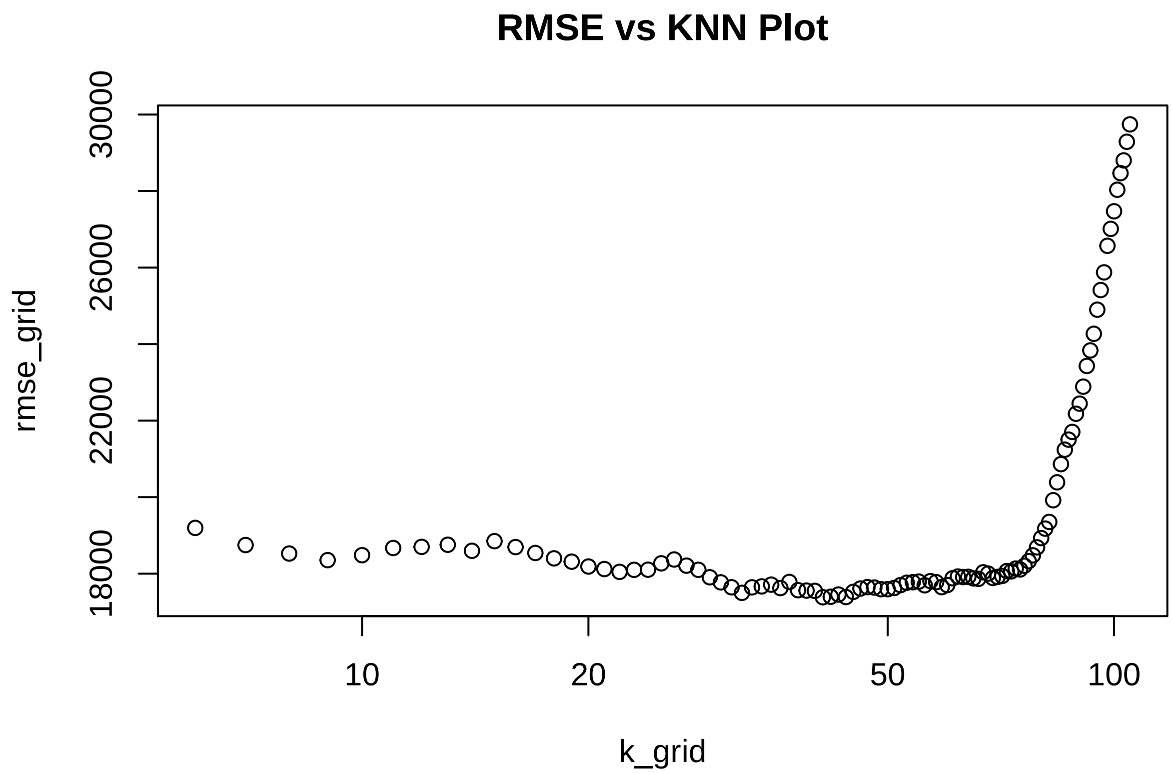

```
p_test + geom_point(aes(x = mileage, y = ypred_knn80), color='red')
```




```
#KNN variances
knn_model = knn.reg(X_train, X_train, y_train, k = 80)

D_train$ypred = knn_model$pred
p_train = ggplot(data = D_train) +
  geom_point(mapping = aes(x = mileage, y = price), color='lightgrey') +
  theme_bw(base_size=18)
p_train + geom_path(mapping = aes(x=mileage, y=ypred), color='red', size=1.5)
```




```r
#RMSE plot vs K nearest neighbors 65 AMG trim level.
N = nrow(sclass65AMG)
N_train = floor(0.8*N)
N_test = N - N_train

train_ind = sort(sample.int(N, N_train, replace=FALSE))

D_train = sclass65AMG[train_ind,]
D_train = arrange(D_train, mileage)
D_test = sclass65AMG[-train_ind,]

y_train = D_train$price
X_train = data.frame(mileage=jitter(D_train$mileage))
X_test = data.frame(mileage=jitter(D_test$mileage))
y_test = D_test$price
library(foreach)
library(FNN)

k_grid = (seq(log(300), length=100)) %>% round %>% unique
rmse_grid = foreach(K = k_grid, .combine='c') %do% {
  knn_model = knn.reg(train = X_train, test = X_test, y = y_train, k=K)
  rmse(y_test, knn_model$pred)
}
plot(k_grid, rmse_grid, log='x', main="RMSE vs KNN Plot")
abline(h=rmse(y_test, ypred_knn100), col='red')
```




```r
#Optimal value for K for 65 AMG trim level is
#K = 80
lm1 = lm(price ~ mileage, data=D_train)
lm2 = lm(price ~ poly(mileage, 2), data=D_train)
knn80 = knn.reg(train = X_train, test = X_test, y = y_train, k=80)


#rmse
rmse = function(y, ypred) {
  sqrt(mean(data.matrix((y-ypred)^2)))
}

ypred_lm1 = predict(lm1, X_test)
ypred_lm2 = predict(lm2, X_test)
ypred_knn80 = knn80$pred

rmse(y_test, ypred_lm1)
```

```
## [1] 39264.96
```

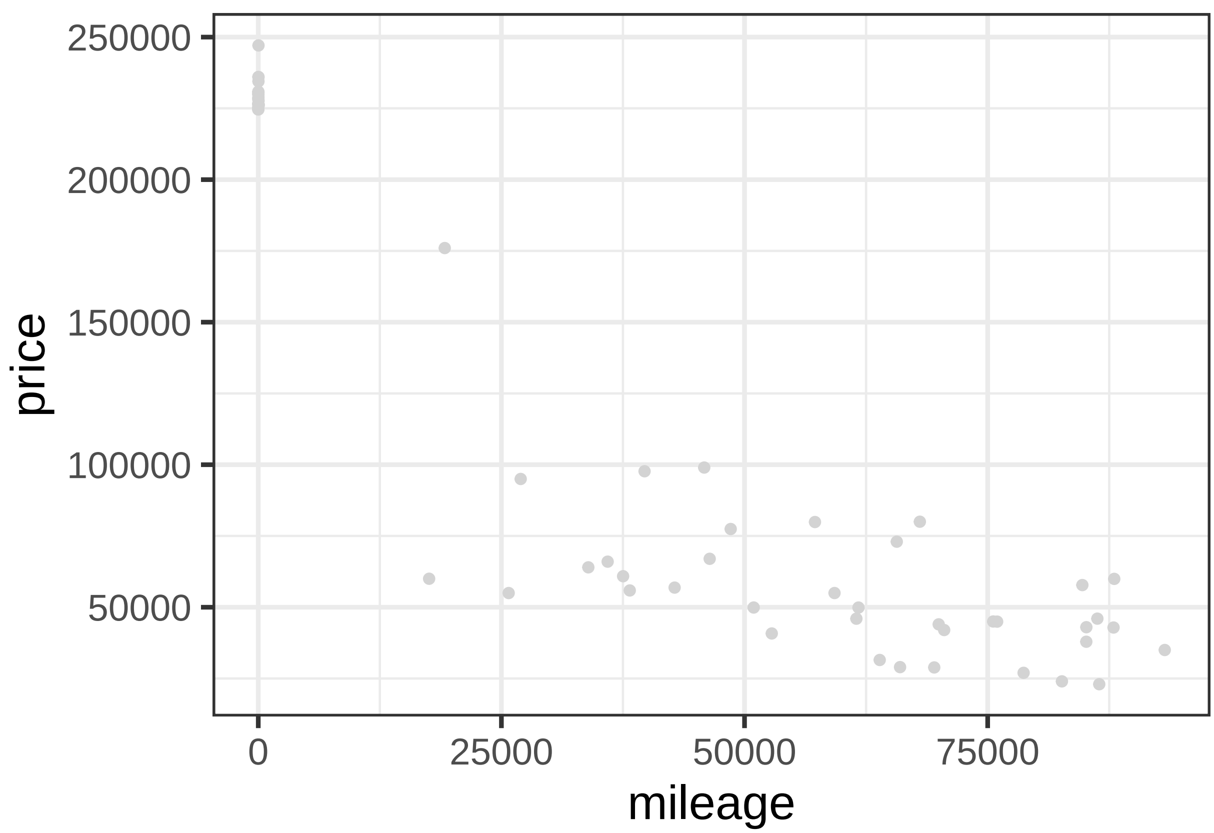
```r
rmse(y_test, ypred_lm2)
```

```
## [1] 27019.97
```

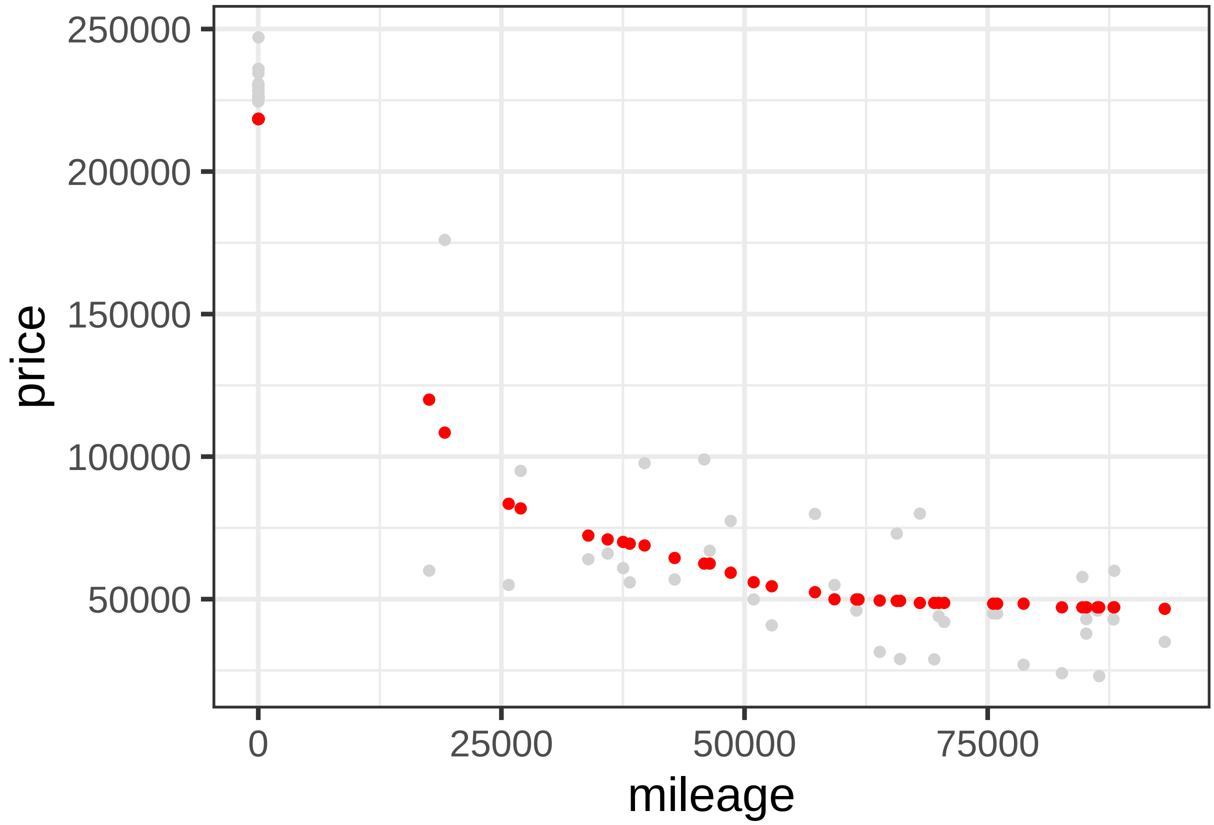
```r
rmse(y_test, ypred_knn80)
```

```
## [1] 18930.06
```

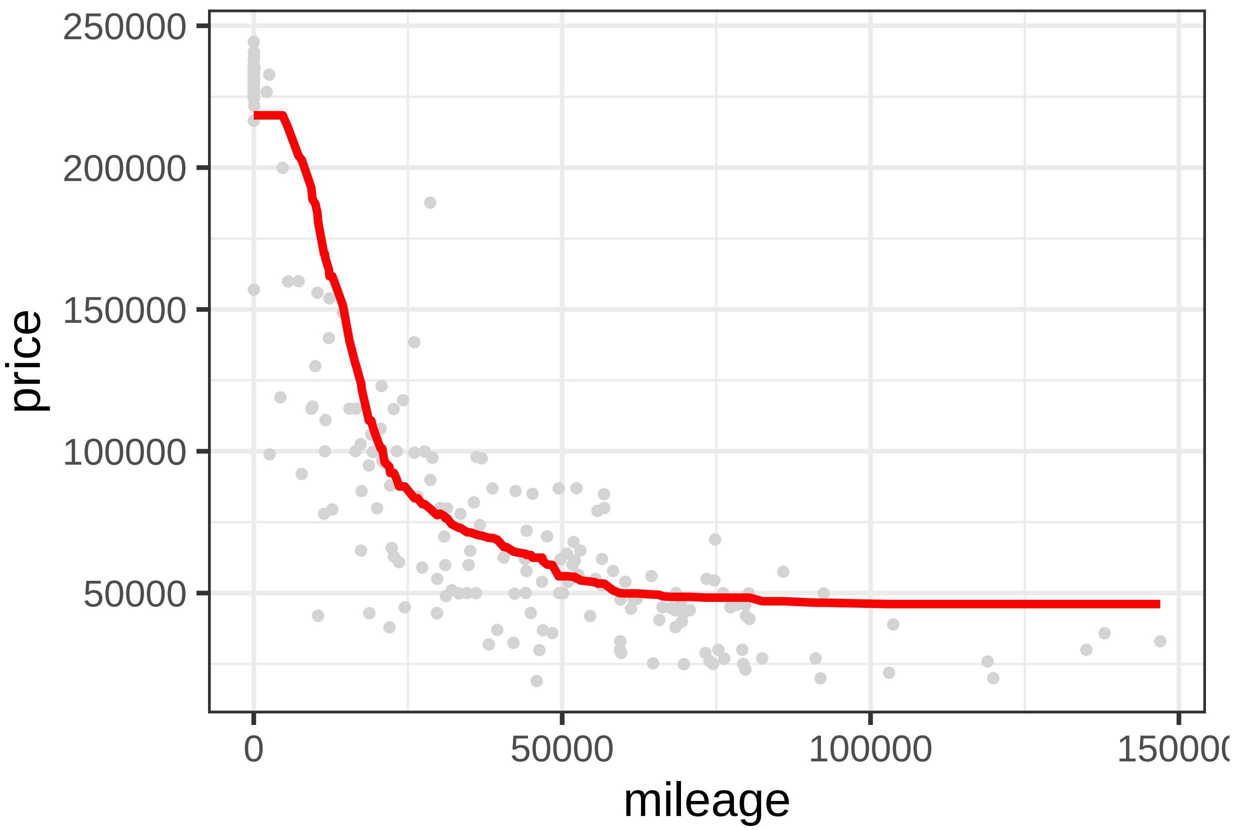
```r
#attach predictions to data frame
D_test$ypred_lm2 = ypred_lm2
D_test$ypred_knn80 = ypred_knn80

p_test = ggplot(data = D_test) +
  geom_point(mapping = aes(x = mileage, y = price), color='lightgrey') +
  theme_bw(base_size=18)
p_test
```




```r
p_test + geom_point(aes(x = mileage, y = ypred_knn80), color='red')
```




```
#KNN variances
knn_model = knn.reg(X_train, X_train, y_train, k = 80)

D_train$ypred = knn_model$pred
p_train = ggplot(data = D_train) +
  geom_point(mapping = aes(x = mileage, y = price), color='lightgrey') +
  theme_bw(base_size=18)
p_train + geom_path(mapping = aes(x=mileage, y=ypred), color='red', size=1.5)
```




```
#The optimal value of K for the 65 AMG trim level is K=80. With this value of K, the rmse is 32901.62, 
```

```
#Conclusion: The 65 AMG level has an optimal value of K that is higher than the 350 trim level optimal 
```

Problem 2 - Saratoga Houses

Question

This analysis tries to examine and identify key indicators in home prices in Saratoga, New York. Homes have a nearly infinite number of attributes that may contribute to the value of the home, and this report seeks to find the most relevant for predicting what the value of a home is.

Data

```
library(tidyverse)
library(mosaic)
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
##
##
##
```

There are 16 variables in a dataset of 1,728 observations. The objective is to predict the variable "Price" using some combination of the other 15 variables.

Method

```
library(mosaic)

n = nrow(SaratogaHouses)
n_train = round(0.8*n)
n_test = n - n_train

rmse = function(y, yhat) {
  sqrt( mean( (y - yhat)^2 ) )
}

rmse_vals = do(100)*{

  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,]

  lm1 = lm(price ~ lotSize + bedrooms + bathrooms, data=saratoga_train)
  lm2 = lm(price ~ . - sewer - waterfront - landValue - newConstruction, data=saratoga_train)
  lm3 = lm(price ~ (. - sewer - waterfront - landValue - newConstruction)^2, data=saratoga_train)
```

```
  lm4 = lm(price ~ . - sewer - landValue - newConstruction, data=saratoga_train)
  lm5 = lm(price ~ . - sewer - waterfront - landValue, data=saratoga_train)
  lm6 = lm(price ~ . - sewer - landValue, data=saratoga_train)
  lm7 = lm(price ~ lotSize + age + livingArea + pctCollege +
                    bedrooms + fireplaces + bathrooms + rooms + heating + fuel +
                    centralAir + lotSize:heating + livingArea:rooms + newConstruction + livingArea:newC
  lm8 = lm(price ~ (lotSize + age + livingArea + pctCollege +
            bedrooms + fireplaces + bathrooms + rooms + heating + fuel +
            centralAir + lotSize:heating + livingArea:rooms + newConstruction + livingArea:newConstruct
  lm9 = lm(price ~ . - sewer - landValue - rooms, data=saratoga_train)
  lm10 = lm(price ~ . - sewer - landValue - bathrooms - bedrooms, data=saratoga_train)
  lm11 = lm(price ~ (. - sewer - landValue)^2, data=saratoga_train)

  yhat_test1 = predict(lm1, saratoga_test)
  yhat_test2 = predict(lm2, saratoga_test)
  yhat_test3 = predict(lm3, saratoga_test)
  yhat_test4 = predict(lm4, saratoga_test)
  yhat_test5 = predict(lm5, saratoga_test)
  yhat_test6 = predict(lm6, saratoga_test)
  yhat_test7 = predict(lm7, saratoga_test)
  yhat_test8 = predict(lm8, saratoga_test)
  yhat_test9 = predict(lm9, saratoga_test)
  yhat_test10 = predict(lm10, saratoga_test)
  yhat_test11 = predict(lm11, saratoga_test)

  c(rmse(saratoga_test$price, yhat_test1),
    rmse(saratoga_test$price, yhat_test2),
    rmse(saratoga_test$price, yhat_test3),
    rmse(saratoga_test$price, yhat_test4),
    rmse(saratoga_test$price, yhat_test5),
    rmse(saratoga_test$price, yhat_test6),
    rmse(saratoga_test$price, yhat_test7),
    rmse(saratoga_test$price, yhat_test8),
    rmse(saratoga_test$price, yhat_test9),
    rmse(saratoga_test$price, yhat_test10),
    rmse(saratoga_test$price, yhat_test11))
}
```

To find the best model for predicting home price, we create a train-test split, and use the training data to to create models and test their accuracy by calculating the Root Mean Square Error (RMSE) of each model.

Results

```
rmse_vals
```

```
##            V1       V2       V3       V4       V5       V6       V7        V8
## 1    82172.80 73631.79 70692.00 70610.32 73535.59 70552.74 73429.12  96549.17
## 2    76769.52 65199.11 68789.37 65528.58 65386.89 65697.91 65012.26  71381.39
## 3    80435.94 71774.45 85120.30 70699.11 71634.05 70592.62 71349.45  94328.11
## 4    80267.98 69642.08 69752.54 65877.23 69481.53 65778.74 68781.83  69575.00
## 5    74932.32 68428.06 70389.78 67974.17 68302.09 67881.73 68636.98  89216.33
```

```
## 6   71351.06 58161.42  62992.30 57025.77 58085.74 56940.84 59906.22  70044.11
## 7   83656.71 69979.19  71723.62 66230.09 69816.44 66143.46 69466.18  79221.96
## 8   69359.73 59841.42  76805.36 58252.41 59925.49 58305.80 60507.77  95452.78
## 9   74445.91 59208.47  63854.65 58184.14 58970.53 57962.68 58724.02  86292.52
## 10  82096.88 68726.25  71821.68 69367.21 68414.27 69102.56 68003.63  69319.05
## 11  75446.56 67097.20  66856.04 67055.85 66780.29 66788.61 67019.42  95188.89
## 12  74962.27 65465.88 100052.60 64982.01 65373.49 64872.10 65834.54 135149.35
## 13  78147.01 65852.52  68299.10 63208.50 65751.69 63141.33 65802.35 134650.15
## 14  77204.88 64801.53  66960.84 60080.72 64587.43 59919.30 65121.46  79946.51
## 15  76628.59 66788.69  70665.29 64511.02 66515.73 64281.96 67478.84  94863.13
## 16  81013.20 68566.25  68665.60 64319.49 68445.41 64278.63 68978.20  78405.36
## 17  78107.94 68303.00 113157.59 68133.58 68222.08 68050.55 67709.34  98641.45
## 18  80158.19 73438.39  73175.16 73074.15 73305.46 72982.87 73452.90  73591.77
## 19  75972.00 65428.65  90266.63 62360.24 65295.52 62262.10 65487.21  77979.68
## 20  72160.63 64043.47  67286.40 62779.35 64540.91 63218.69 66056.70  73649.85
## 21  83860.99 68793.91  67498.35 68456.09 68722.37 68402.34 68926.94  85866.56
## 22  84098.57 75768.76  74064.86 74656.08 75626.57 74542.57 75747.08  78946.11
## 23  68383.40 59126.59  60626.71 58805.86 59028.21 58713.51 58824.26  78275.07
## 24  78021.36 69370.80  82108.11 65257.12 69350.10 65300.59 68832.50  86063.50
## 25  76951.96 64395.38  64724.33 63411.00 64339.17 63307.19 63539.31  66293.74
## 26  87033.76 74429.91  71205.20 70430.91 74425.92 70481.73 74069.11  83044.38
## 27  81280.88 66317.43  64314.82 65331.40 66207.92 65233.76 65771.66  72517.83
## 28  73269.44 62528.99  80245.53 61650.44 62523.86 61646.49 62352.21  89212.59
## 29  79908.72 71495.50  71954.85 68528.91 71635.04 68673.51 71149.59  72813.11
## 30  71024.65 57956.89  62858.60 57147.33 57866.05 57058.67 58172.98  62068.19
## 31  80464.49 68010.12  65511.50 64655.43 67666.61 64373.52 66772.29  66080.80
## 32  79549.49 65695.60  65032.89 65404.09 65720.96 65452.31 64906.32  81551.03
## 33  69837.64 60491.60  66171.86 58525.14 60461.22 58497.99 59624.42  69549.39
## 34  80998.22 69670.24  80243.60 67212.60 69374.00 66962.21 69956.22  68480.81
## 35  75877.43 62426.50  65899.32 61338.83 62360.36 61301.98 62528.79  65361.34
## 36  76057.57 64107.36  64139.74 62789.29 64180.47 62902.70 64361.57  71280.58
## 37  66035.21 57968.82  56396.24 57054.78 57841.75 56941.18 57947.70  57850.06
## 38  72131.92 57865.86  86889.02 58570.10 57595.01 58327.27 57417.25 181336.46
## 39  81519.59 66731.04  66726.99 63085.26 66922.07 63315.40 66537.96  72751.30
## 40  78617.82 69309.77  67813.04 68667.88 69165.43 68577.58 68550.99  67687.03
## 41  75882.78 64666.82  89957.48 64544.22 64593.58 64455.48 63897.39  95179.82
## 42  78252.80 66428.07  65097.52 65802.52 66303.15 65677.07 66353.78  64553.93
## 43  75897.20 65850.54  71502.80 64843.41 65624.27 64661.94 65577.08  77014.39
## 44  77642.38 69807.24  70553.03 67562.93 69597.46 67386.85 69494.03  80567.87
## 45  82339.10 73627.58  77932.86 71324.36 73920.34 71656.77 74308.23  85524.95
## 46  77036.59 68050.92  68600.02 67924.57 67858.19 67782.79 67799.04  77308.18
## 47  80006.47 68315.48  66668.93 66236.71 68230.90 66157.46 67672.38  68275.11
## 48  71514.93 58903.76  59756.51 57582.28 59170.98 57837.52 59393.79  82619.85
## 49  78748.10 65698.50  68739.72 65415.35 65532.53 65261.73 65322.64  75439.78
## 50  75512.64 61073.14  66063.32 60800.72 61464.61 61096.08 61360.88  83811.24
## 51  85455.77 71618.66  73836.39 69664.30 71442.11 69510.89 71879.07  81832.07
## 52  80642.44 72721.44  78485.85 69564.76 72622.71 69489.20 71964.34  90900.13
## 53  77793.38 68173.55  72321.63 64822.82 68196.32 64834.43 67576.31  86589.23
## 54  75731.54 65369.35  66413.72 62045.00 65362.32 62071.16 65275.91  81429.74
## 55  83012.04 69108.97  71200.67 65810.00 68806.47 65570.50 69749.72 101852.93
## 56  73796.35 62039.09  61695.40 58081.92 61852.80 57982.25 61997.16  61194.09
## 57  73028.01 62487.32  65403.88 63203.13 62597.16 63254.68 61887.37  79132.89
## 58  69536.04 62978.26  69044.60 63078.89 62976.81 63079.98 62377.29  64227.45
## 59  79146.54 64839.24  80087.38 64448.83 64670.07 64291.36 67293.75  75258.75
```

```
## 60  77217.51 70971.19  69871.11 71090.76 71107.26 71170.67 71568.00  78508.00
## 61  87226.70 71350.29  72956.68 67811.37 71173.17 67709.75 72273.83  73440.56
## 62  75830.48 65515.56  68641.10 63735.56 65375.37 63604.06 65069.21  74904.40
## 63  68094.48 61081.76  66630.12 59829.28 61064.68 59793.27 62078.67  87934.25
## 64  80835.63 69540.59  70493.18 65585.86 69386.78 65490.24 69542.41  70754.55
## 65  78603.19 70793.89  68590.91 70232.90 71046.43 70453.08 71140.79  69844.64
## 66  79249.61 70249.04  67367.34 68251.07 70362.86 68361.67 69749.93  74380.67
## 67  85224.90 78234.60  77609.21 75798.71 78079.96 75663.97 77868.24  77650.59
## 68  75191.72 65378.15  66152.17 63741.73 65510.90 63816.09 65671.63  67331.78
## 69  79212.51 63856.53  64493.72 63200.54 63721.32 63064.67 62796.53  64171.36
## 70  84165.16 73632.33  72227.05 72919.97 73761.60 73015.28 74155.36  73467.43
## 71  71585.66 59755.46  63063.15 59461.16 59686.36 59373.95 60999.67  74829.37
## 72  81365.41 67325.41  66707.39 65886.13 67098.97 65700.80 66615.30  69063.66
## 73  73946.53 63345.36  61012.35 63707.00 63163.17 63558.21 62411.40  63328.32
## 74  73255.92 62041.24  62028.62 60545.10 61779.85 60321.26 61154.48  71926.49
## 75  77642.26 62714.51  65782.66 62279.52 62622.60 62178.59 62373.99  68783.86
## 76  76708.31 68024.04  66539.48 68714.26 68058.83 68734.64 67564.05  79627.58
## 77  85008.33 76065.50  76185.77 73261.53 75791.51 73033.98 75510.01  78176.29
## 78  67846.37 59082.38  60931.54 57849.07 59099.59 57806.52 59568.80  67615.92
## 79  84549.27 77861.65  79762.08 76227.16 77755.84 76142.17 77059.06  98121.62
## 80  80383.07 69932.30  73631.49 66737.00 69859.84 66684.78 71757.65  79710.87
## 81  93045.20 80035.69  76401.03 77452.91 79834.04 77282.04 79148.93  80804.77
## 82  75917.68 65276.91  64485.99 60729.19 65174.66 60673.75 64844.36  64663.90
## 83  76724.07 69474.83  71359.18 65756.48 69514.85 65840.12 69160.32  72204.28
## 84  80854.50 68184.29  69939.59 65079.04 68046.40 65033.51 67420.59  82604.79
## 85  79096.66 69772.88  94767.17 68706.43 69613.13 68547.05 70700.97  79045.40
## 86  84825.80 73843.24  75174.25 71167.13 73645.65 71004.04 73709.51  74813.68
## 87  80284.46 71676.36  72616.89 68534.64 71775.06 68681.17 71903.71  73166.94
## 88  88524.90 77584.93  74007.86 78752.17 77438.21 78616.98 76960.87  74282.17
## 89  70621.97 63489.86  64890.65 64063.17 63325.67 63921.00 62662.71  66262.77
## 90  74595.88 64449.58  66597.18 60552.56 64322.77 60437.85 65198.66  66455.95
## 91  78313.31 67976.87  69893.42 66805.17 67937.10 66770.84 68772.25  77901.58
## 92  75378.01 66605.48  79697.68 63692.40 66484.84 63598.39 67971.91  67735.06
## 93  76812.49 66137.66  72479.25 65802.94 65897.83 65603.69 65474.57  74654.54
## 94  75471.03 64084.74 214659.33 62818.41 63822.59 62581.47 63108.94 185841.13
## 95  74934.29 66358.90  93975.46 63786.40 66340.81 63773.23 66661.54  77210.47
## 96  77360.53 66702.32  69792.03 65477.75 66697.32 65447.79 65954.40  71470.52
## 97  75003.54 64079.87  68156.07 62046.95 63747.20 61783.77 63360.22  74453.95
## 98  63439.66 53751.38  52435.44 53987.77 53595.20 53851.50 53415.60  53827.96
## 99  90171.33 82636.29  97429.03 81647.49 82444.50 81476.12 82461.01  95135.34
## 100 72477.93 63799.41  71055.83 62723.80 64235.18 63067.61 64563.57  89345.93
##           V9      V10        V11
## 1   70909.88 71057.42   87134.40
## 2   65884.63 65902.04  128321.72
## 3   70780.30 71895.75   85705.92
## 4   65678.91 67177.71  151025.19
## 5   68306.68 69606.91   74950.79
## 6   57171.84 57712.15   68089.00
## 7   66314.46 67303.28  149320.79
## 8   58708.68 59655.50  124878.49
## 9   58362.88 58496.64   74120.79
## 10  69406.50 70733.64   75567.86
## 11  66221.78 67608.32   67617.96
## 12  65345.50 65937.40  109682.00
```

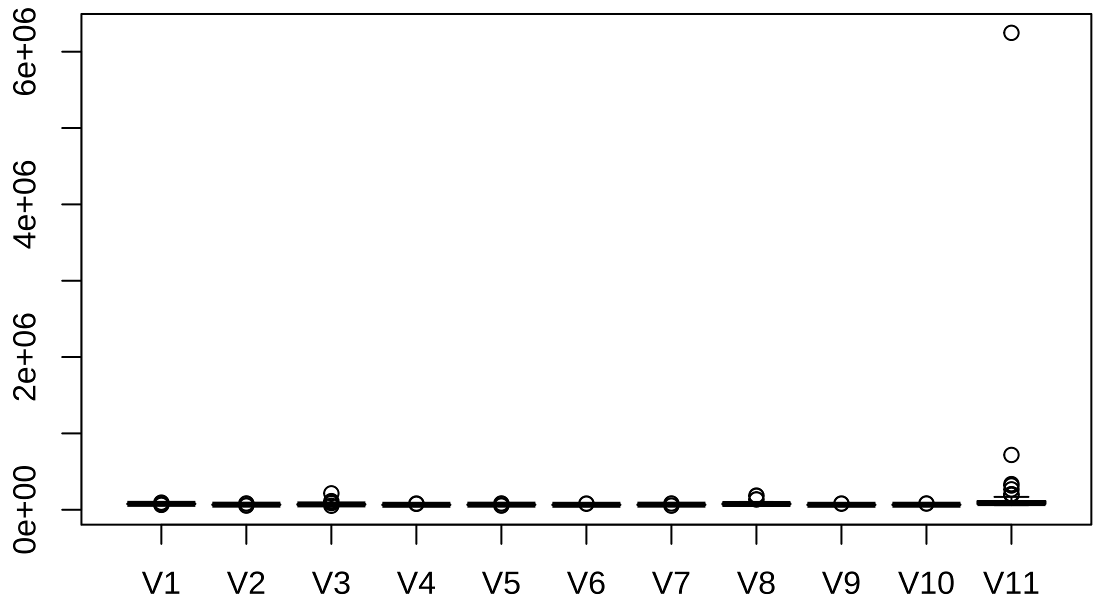
```
## 13  63277.15 63198.41   65018.66
## 14  60195.05 60736.14  717704.97
## 15  63966.61 65160.67  115320.02
## 16  64276.67 65318.54  336128.03
## 17  67774.02 68387.92  106003.05
## 18  73450.53 74589.66   75379.98
## 19  62429.17 63231.36  113413.43
## 20  62718.39 63231.33  101074.79
## 21  68593.77 68873.15   73884.12
## 22  74797.43 75959.70   82591.49
## 23  59073.50 59515.07   61935.07
## 24  65260.74 66643.60  168670.96
## 25  63476.08 64409.11   70715.26
## 26  70606.31 70394.78   93542.52
## 27  65318.29 65606.72   67425.78
## 28  61876.18 62757.93   82850.16
## 29  68911.66 69066.37   81493.02
## 30  57291.71 57724.19   98406.10
## 31  64454.34 64333.20  202629.22
## 32  65683.27 66169.09   67809.64
## 33  58400.69 58237.16   81409.46
## 34  66733.04 67000.14  101363.34
## 35  61496.23 61839.94   72158.62
## 36  62116.15 63035.41  162858.07
## 37  56752.32 58389.05   89784.04
## 38  58849.59 60082.49   95098.58
## 39  62996.48 63025.65   82833.25
## 40  68936.23 69881.89 6247008.80
## 41  63882.86 65606.12   82791.22
## 42  66231.75 66830.52   74962.58
## 43  65205.12 66082.22   73509.39
## 44  67816.09 68697.14   86161.25
## 45  72168.24 72814.16  204382.52
## 46  68045.40 69738.76   77434.75
## 47  66682.31 67478.35   96734.44
## 48  57827.33 58949.26   98053.82
## 49  65564.11 65799.28   81732.53
## 50  61384.37 62208.75   67577.45
## 51  70022.67 71015.87  145945.32
## 52  69621.63 71179.41   76909.40
## 53  65300.85 65342.33   82427.27
## 54  62043.75 63707.63   83018.58
## 55  65859.80 65982.19  267134.91
## 56  58126.74 59774.64   77530.46
## 57  63446.15 63680.92   88685.16
## 58  63203.32 63907.38   74191.92
## 59  64449.69 64767.08  140592.35
## 60  70935.49 72352.09   79052.24
## 61  67971.02 69267.19  100813.07
## 62  63386.60 63114.19   74880.09
## 63  60122.55 61926.66   72784.61
## 64  65766.31 67039.78   94737.49
## 65  70680.68 71274.23   81673.60
## 66  68797.21 69792.34  317088.89
```

```
## 67  75934.47 76464.00  168657.25
## 68  64227.49 65102.20  108306.10
## 69  62520.51 62995.65   75632.74
## 70  73458.10 74311.03  114040.92
## 71  59417.11 61011.83   80722.12
## 72  65875.15 66296.17   70594.82
## 73  63770.10 63909.20   75926.98
## 74  60916.45 61674.23   64381.56
## 75  62578.08 63617.95   99725.65
## 76  68666.80 70226.42   67794.06
## 77  73273.03 73990.00   86013.84
## 78  57891.82 59652.96  139638.01
## 79  76495.65 77257.61  103008.49
## 80  66976.72 67243.33  119217.07
## 81  77461.18 77928.50  140918.47
## 82  60791.77 61704.41  114137.94
## 83  65645.04 66521.77   99365.80
## 84  64725.93 65678.08   90976.28
## 85  68508.38 70175.59   95841.77
## 86  71219.07 71580.28  196530.12
## 87  68884.89 69591.78   79975.77
## 88  79077.90 80495.03  106345.07
## 89  64369.47 65826.04   67981.50
## 90  60918.51 61895.41   82184.44
## 91  66652.63 66986.71  107944.73
## 92  63674.68 64781.10   93182.74
## 93  65601.73 66118.18   77806.59
## 94  63086.43 63304.27  194299.22
## 95  63720.62 65573.77  118402.17
## 96  65658.45 66041.70   80743.77
## 97  61053.30 63329.36   76862.05
## 98  53527.32 55022.80   60317.07
## 99  81587.83 82707.29  103745.16
## 100 63038.20 63822.74   71234.94
```

```r
colMeans(rmse_vals)
```

```
##        V1        V2        V3        V4        V5        V6        V7        V8
##  77681.85  67012.70  72856.05  65404.38  66936.06  65344.95  66943.11  80186.42
##        V9       V10       V11
##  65485.51  66310.51 171441.86
```

```r
boxplot(rmse_vals)
```




```r
Xtrain = model.matrix(~ . - (price + sewer + landValue) - 1, data=saratoga_train)
Xtest = model.matrix(~ . - (price + sewer + landValue) - 1, data=saratoga_test)
ytrain = saratoga_train$price
ytest = saratoga_test$price

scale_train = apply(Xtrain, 2, sd)
Xtilde_train = scale(Xtrain, scale = scale_train)
Xtilde_test = scale(Xtest, scale = scale_train)

head(Xtrain, 2)
```

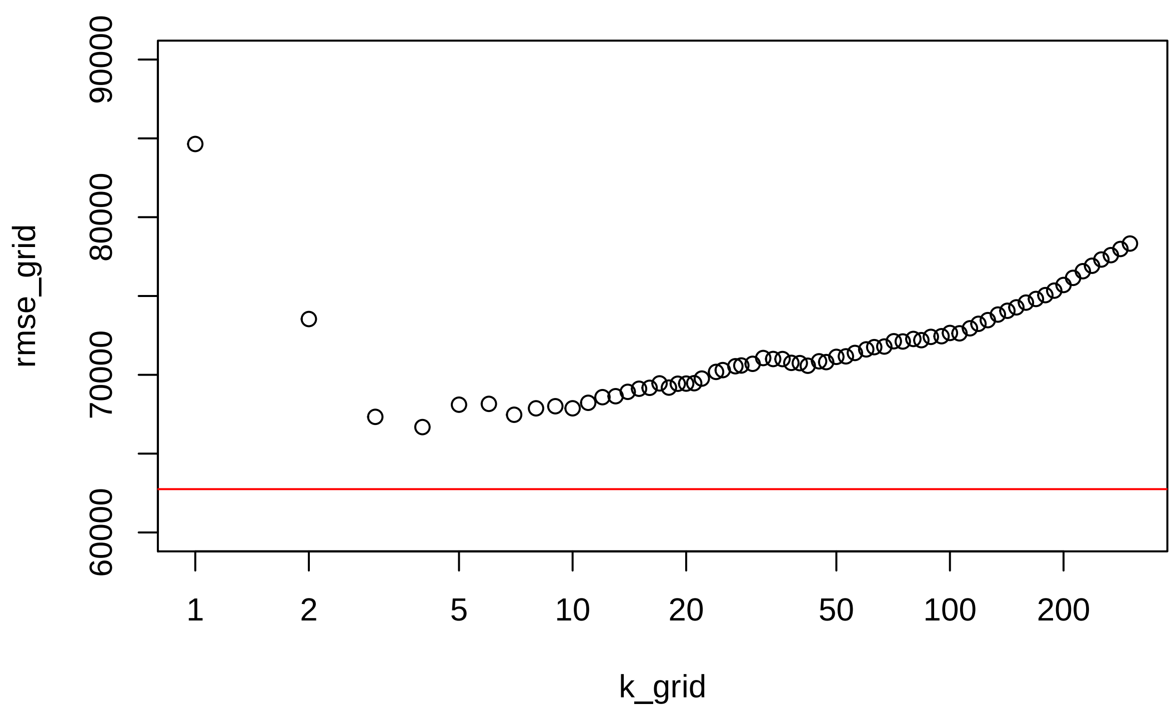
```
##      lotSize age livingArea pctCollege bedrooms fireplaces bathrooms rooms
## 1540    0.92  16       2290         75        4          1       2.0     7
## 1584    0.30  32       1918         64        4          1       1.5     6
##      heatinghot air heatinghot water/steam heatingelectric fuelelectric fueloil
## 1540              0                      1               0            0       1
## 1584              1                      0               0            0       0
##      waterfrontNo newConstructionNo centralAirNo
## 1540            1                 1            1
## 1584            1                 1            0
```

```r
library(FNN)
K = 10

knn_model = knn.reg(Xtilde_train, Xtilde_test, ytrain, k=K)
```

```r
rmse(ytest, knn_model$pred)
```

```
## [1] 67875.62
```

```r
library(foreach)
k_grid = exp(seq(log(1), log(300), length=100)) %>% round %>% unique
rmse_grid = foreach(K = k_grid, .combine='c') %do% {
  knn_model = knn.reg(Xtilde_train, Xtilde_test, ytrain, k=K)
  rmse(ytest, knn_model$pred)
}

plot(k_grid, rmse_grid, log='x', ylim=c(60000, 90000))
abline(h=rmse(ytest, yhat_test6), col="red")
```



After running 100 train-tests splits, we find the average RMSE for each model. We find that Linear Model #6 is the best model because it has the lowest average RMSE. This model is a modification of the original, medium sized model which factors in Lot Size, age, living area, percent of college students in the area, number of bedrooms, number of fireplaces, number of bathrooms, number of rooms, heating type, fuel, and central air system. The modification adds the variables Waterfront and New Constrution. Meanwhile, we exclude the land value and the sewer system from the model. We assume that some of the price already reflects the land value, so it doesn't explain any additional difference in home prices. We exclude the sewer system variable, because most home-buyers are unlikely to closely examine the type of sewer system in a potential home.

We observe that including interactions between variables makes the model more inaccurate (LM-11). We compare the model to a K-nearest neighbors model. We find that the best of the linear models (LM-6) is still better than all of the K-nearest neighbors models generated.

Conclusion

An improved model for predicting home prices is a linear model that includes Lot Size, age, living area, percent of college students in the area, number of bedrooms, number of fireplaces, number of bathrooms, number of rooms, heating type, fuel, central air system, Waterfront, and New Construction. This linear model excludes interactions and is slightly more accurate than the most accurate k-nearest neighbors models.

Problem 3 - Viral News

Regression

Testing different models out of a set of potentially meaningful predictors for number of shares (virality).

```
require(MASS)
```

```
## Loading required package: MASS

##
## Attaching package: 'MASS'

## The following object is masked from 'package:dplyr':
##
##     select
```

```
require(leaps)
```

```
## Loading required package: leaps

## Warning: package 'leaps' was built under R version 3.6.3
```

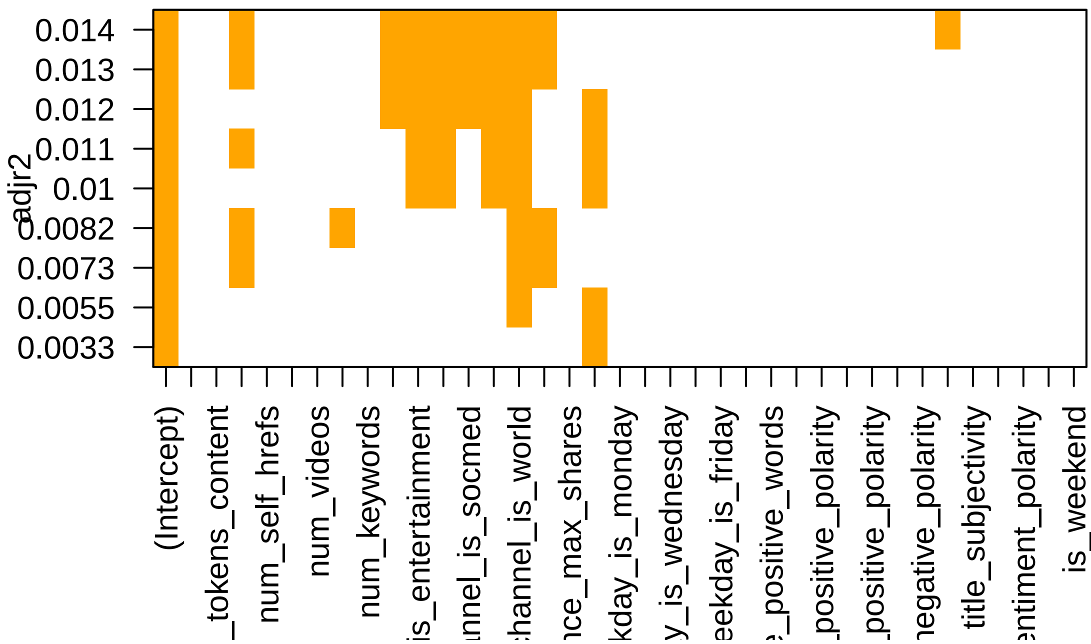
```
#create set of linear models
online_news <- read_csv("SDS323/data/online_news.csv")
```

```
## Parsed with column specification:
## cols(
##   .default = col_double(),
##   url = col_character()
## )

## See spec(...) for full column specifications.
```

```
mysubsets <-regsubsets(shares ~ . - url, data=online_news)
```

```
## Warning in leaps.setup(x, y, wt = wt, nbest = nbest, nvmax = nvmax, force.in =
## force.in, : 2 linear dependencies found

## Reordering variables and trying again:
```

```r
#rank by adjusted R-squared
plot(mysubsets, scale="adjr2", col="orange")
```



Resulting Model

```r
model1 <- lm(shares ~ n_tokens_content + num_hrefs + num_self_hrefs + num_imgs + num_videos + average_to
summary(model1)
```

```
##
## Call:
## lm(formula = shares ~ n_tokens_content + num_hrefs + num_self_hrefs +
##     num_imgs + num_videos + average_token_length + self_reference_avg_sharess +
##     avg_negative_polarity, data = online_news)
##
## Residuals:
##    Min     1Q Median     3Q    Max
## -25236  -2304  -1587   -398 838592
##
## Coefficients:
##                              Estimate Std. Error t value Pr(>|t|)
## (Intercept)                 4.685e+03  3.217e+02  14.563  < 2e-16 ***
## n_tokens_content           -5.724e-01  1.428e-01  -4.009 6.11e-05 ***
## num_hrefs                   5.494e+01  6.194e+00   8.869  < 2e-16 ***
## num_self_hrefs             -7.070e+01  1.674e+01  -4.223 2.42e-05 ***
## num_imgs                    4.700e+01  7.786e+00   6.036 1.59e-09 ***
```

```
## num_videos                  5.179e+01  1.453e+01   3.564 0.000365 ***
## average_token_length       -5.710e+02  7.473e+01  -7.641 2.21e-14 ***
## self_reference_avg_sharess  2.648e-02  2.409e-03  10.992  < 2e-16 ***
## avg_negative_polarity      -3.112e+03  4.877e+02  -6.382 1.77e-10 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 11570 on 39635 degrees of freedom
## Multiple R-squared:  0.009373,   Adjusted R-squared:  0.009173
## F-statistic: 46.88 on 8 and 39635 DF,  p-value: < 2.2e-16
```

```
adjr2_model <- lm(shares ~ num_hrefs + data_channel_is_lifestyle + data_channel_is_entertainment + data_
summary(adjr2_model)
```

```
##
## Call:
## lm(formula = shares ~ num_hrefs + data_channel_is_lifestyle +
##     data_channel_is_entertainment + data_channel_is_bus + data_channel_is_socmed +
##     data_channel_is_tech + data_channel_is_world + self_reference_min_shares +
##     max_negative_polarity, data = online_news)
##
## Residuals:
##    Min     1Q Median     3Q    Max
## -30123  -2102  -1454   -397 837044
##
## Coefficients:
##                                 Estimate Std. Error t value Pr(>|t|)
## (Intercept)                    5.098e+03  1.836e+02  27.760  < 2e-16 ***
## num_hrefs                      3.694e+01  5.184e+00   7.127 1.05e-12 ***
## data_channel_is_lifestyle     -2.174e+03  2.924e+02  -7.434 1.08e-13 ***
## data_channel_is_entertainment -2.738e+03  2.027e+02 -13.507  < 2e-16 ***
## data_channel_is_bus           -2.621e+03  2.092e+02 -12.529  < 2e-16 ***
## data_channel_is_socmed        -2.250e+03  2.815e+02  -7.992 1.37e-15 ***
## data_channel_is_tech          -2.631e+03  2.016e+02 -13.047  < 2e-16 ***
## data_channel_is_world         -3.385e+03  1.960e+02 -17.268  < 2e-16 ***
## self_reference_min_shares      2.965e-02  2.947e-03  10.060  < 2e-16 ***
## max_negative_polarity         -1.229e+03  6.135e+02  -2.003   0.0451 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 11550 on 39634 degrees of freedom
## Multiple R-squared:  0.01378,    Adjusted R-squared:  0.01356
## F-statistic: 61.55 on 9 and 39634 DF,  p-value: < 2.2e-16
```

Creating the testing and training sets

```
#random sample from data for train set
random_sample<-sample(seq_len(nrow(online_news)), size = 35000)
head(random_sample)
```

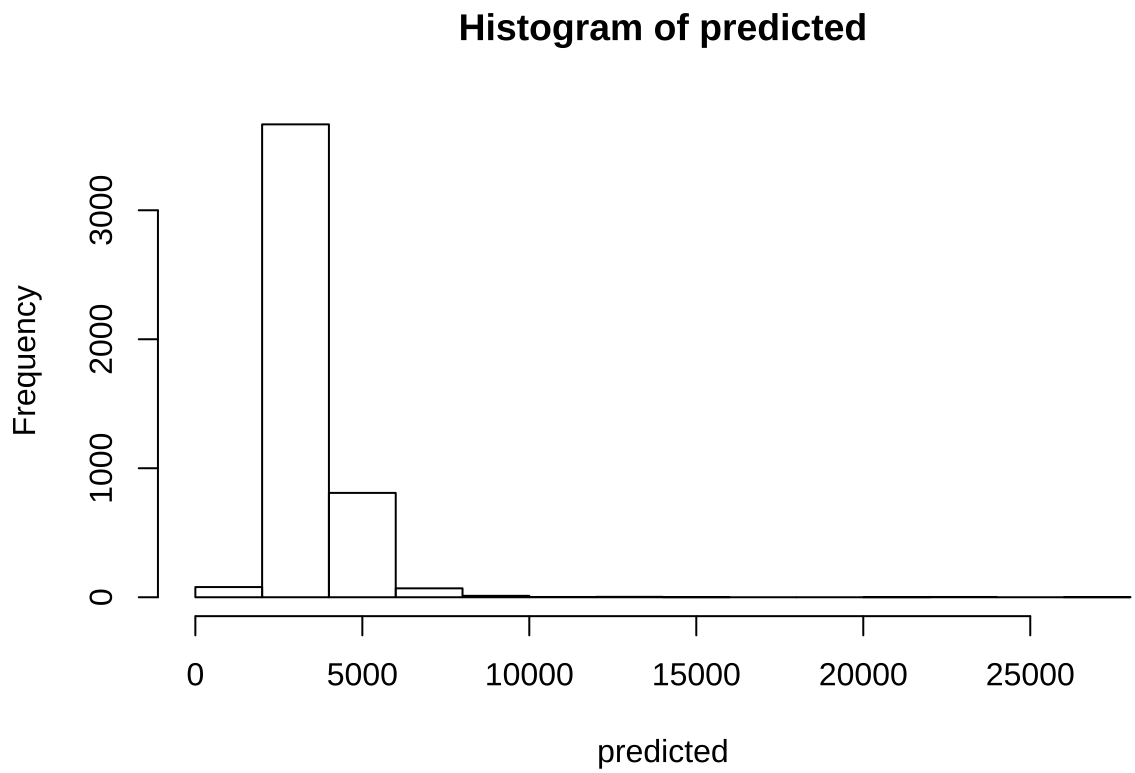
```
## [1]  2634 32574 13460 21511 31608  2048
```

```r
train<-online_news[random_sample,]
#use the rest for test
test<-online_news[-random_sample,]
#training model
training_adjr2_model <- lm(shares ~ n_tokens_content + num_hrefs + num_self_hrefs + num_imgs + num_vide
summary(training_adjr2_model)
```

```
##
## Call:
## lm(formula = shares ~ n_tokens_content + num_hrefs + num_self_hrefs +
##     num_imgs + num_videos + average_token_length + self_reference_avg_sharess +
##     avg_negative_polarity, data = train)
##
## Residuals:
##    Min     1Q Median     3Q    Max
## -24441  -2297  -1573   -382 686861
##
## Coefficients:
##                              Estimate Std. Error t value Pr(>|t|)
## (Intercept)                 4.796e+03  3.267e+02  14.679  < 2e-16 ***
## n_tokens_content           -5.119e-01  1.452e-01  -3.525 0.000424 ***
## num_hrefs                   5.081e+01  6.244e+00   8.138 4.14e-16 ***
## num_self_hrefs             -6.750e+01  1.700e+01  -3.970 7.21e-05 ***
## num_imgs                    4.716e+01  7.951e+00   5.932 3.03e-09 ***
## num_videos                  5.376e+01  1.480e+01   3.633 0.000281 ***
## average_token_length       -6.060e+02  7.592e+01  -7.983 1.47e-15 ***
## self_reference_avg_sharess  2.888e-02  2.492e-03  11.589  < 2e-16 ***
## avg_negative_polarity      -3.212e+03  4.940e+02  -6.502 8.01e-11 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 11070 on 34991 degrees of freedom
## Multiple R-squared:  0.01072,    Adjusted R-squared:  0.0105
## F-statistic: 47.41 on 8 and 34991 DF,  p-value: < 2.2e-16
```

Creating set of model predictions on the testing set

```r
#generate results
predicted<-predict.lm(training_adjr2_model,newdata=test)
hist(predicted)
```



Coverting to binary (viral or not) form:

```
bin_predicted = ifelse(predicted >= 1400,1,0)
bin_actual = ifelse(test$shares >= 1400,1,0)
#creating confusion matrix
conf_viral = table(bin_actual, bin_predicted)
conf_viral
```

```
##            bin_predicted
## bin_actual    0    1
##          0    0 2199
##          1    2 2443
```

Finding average values:

```
correct_neg <- mean(c(2,1,2,2,2))
false_neg <- mean(c(3,1,0,1,0))
correct_pos <- mean(c(2478,2473,2450,2453,2482))
false_pos <- mean(c(2161,2169,2192,2188,2160))
```

Converting average values into confusion matrix:

```
library(knitr)
library(kableExtra)
```

Table 1: Confusion Matrix

Actually_Viral	Predicted_Not	Predicted_Viral
No	1.8	2174.0
Yes	1.0	2467.2

```
## Warning: package 'kableExtra' was built under R version 3.6.3

##
## Attaching package: 'kableExtra'

## The following object is masked from 'package:dplyr':
##
##     group_rows
```

```
avg_conf_viral <- data.frame(
  Actually_Viral = c("No", "Yes"),
  Predicted_Not = c(correct_neg,false_neg),
  Predicted_Viral = c(false_pos,correct_pos)
)
kable(avg_conf_viral, caption = "Confusion Matrix") %>% row_spec(0,bold=FALSE) %>% kable_styling()
```

```
#total error rate
(false_neg+false_pos)/nrow(test) #46.83%
```

```
## [1] 0.4683463
```

```
#false positive rate
false_pos/(false_pos+correct_neg) #99.91%
```

```
## [1] 0.9991727
```

```
#false negative rate
false_neg/(false_neg+correct_pos) #0.04%
```

```
## [1] 0.0004051536
```

This linear model just barely outperforms the null model if the null model were to predict all articles to go viral. This means that the linear probability model we derived is not very useful, but that result is what we expected with an adjusted R-squared of only 0.0106. The model's total error rate is 46.51%, false positive rate is 99.91%, and false negative rate is 0.04%. This linear model predicts all but a few of the articles to viral. The reason that our error rate is below 50% (the expected error rate for random guesses) is there are simply more viral articles than not in the population.

Classification

Create a new binary variable (viral or not):

```
online_news$viral = ifelse(online_news$shares > 1400, 1, 0)
```

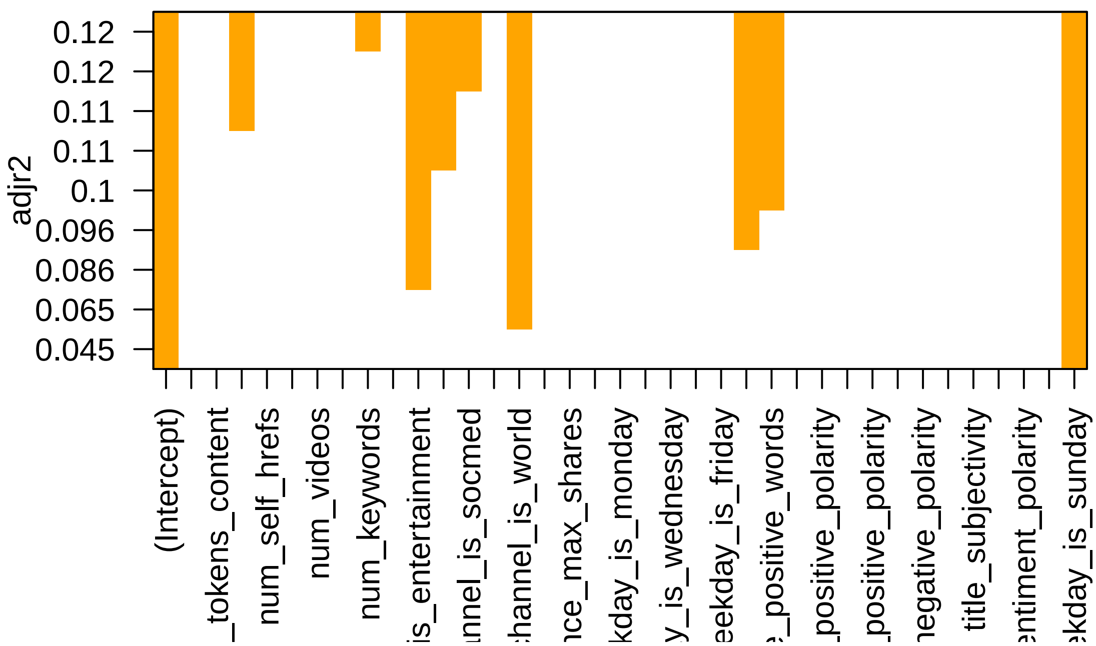
Test possible predictor combinations for predicting this new variable:

```
require(MASS)
require(leaps)
#create set of linear models - remove weekend b/c colinear
mybinsubsets <-regsubsets(viral ~ . - url - is_weekend, data=online_news)
```

```
## Warning in leaps.setup(x, y, wt = wt, nbest = nbest, nvmax = nvmax, force.in =
## force.in, : 1 linear dependencies found
```

```
## Reordering variables and trying again:
```

```
#rank by adjusted R-squared
plot(mybinsubsets, scale="adjr2", col="orange")
```



Resulting Model

```
binary_model <- lm(viral ~ num_hrefs + num_keywords + data_channel_is_entertainment + data_channel_is_bu
```

```
## Warning: In lm.fit(x, y, offset = offset, singular.ok = singular.ok, ...) :
##  extra argument 'family' will be disregarded
```

```r
summary(binary_model)
```

```
##
## Call:
## lm(formula = viral ~ num_hrefs + num_keywords + data_channel_is_entertainment +
##     data_channel_is_bus + data_channel_is_socmed + data_channel_is_world +
##     weekday_is_saturday + weekday_is_sunday + global_rate_positive_words,
##     data = online_news, family = "binomial")
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -1.0693 -0.4496 -0.2795  0.4535  0.7512
##
## Coefficients:
##                                Estimate Std. Error t value Pr(>|t|)
## (Intercept)                   0.4528646  0.0120051  37.723  < 2e-16 ***
## num_hrefs                     0.0026449  0.0002156  12.270  < 2e-16 ***
## num_keywords                  0.0104067  0.0013172   7.901 2.84e-15 ***
## data_channel_is_entertainment -0.2074708  0.0069628 -29.797  < 2e-16 ***
## data_channel_is_bus          -0.0837258  0.0073790 -11.346  < 2e-16 ***
## data_channel_is_socmed        0.1264028  0.0108247  11.677  < 2e-16 ***
## data_channel_is_world        -0.2352816  0.0066583 -35.336  < 2e-16 ***
## weekday_is_saturday           0.2105378  0.0100605  20.927  < 2e-16 ***
## weekday_is_sunday             0.1658719  0.0095576  17.355  < 2e-16 ***
## global_rate_positive_words    0.1234263  0.1437590   0.859    0.391
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.4797 on 39634 degrees of freedom
## Multiple R-squared:  0.07963,    Adjusted R-squared:  0.07942
## F-statistic:   381 on 9 and 39634 DF,  p-value: < 2.2e-16
```

Creating the testing and training sets

```r
#random sample from data for train set
random_bin_sample<-sample(seq_len(nrow(online_news)), size = 35000)
head(random_bin_sample)
```

```
## [1]  2038 31413 23066 19345 11565  7213
```

```r
bin_train<-online_news[random_bin_sample,]
#use the rest for test
bin_test<-online_news[-random_bin_sample,]
#training model
training_binary_model <- lm(viral ~ num_hrefs + num_keywords + data_channel_is_entertainment + data_char
```

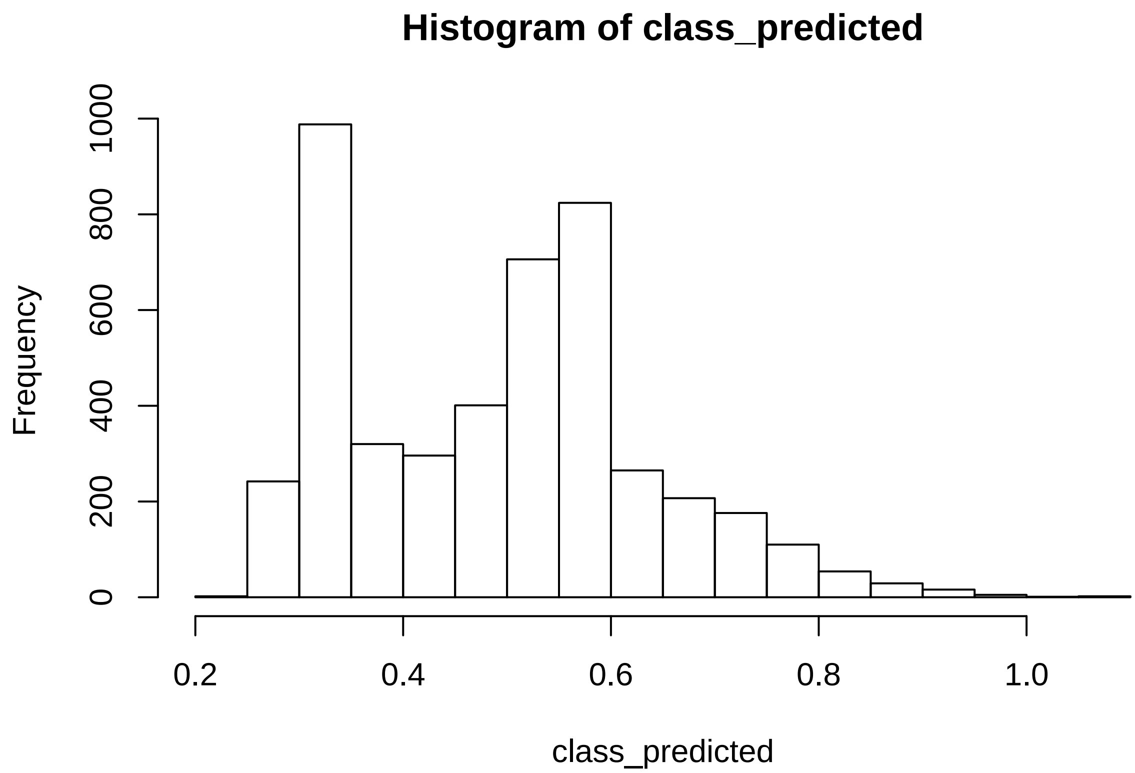
```
## Warning: In lm.fit(x, y, offset = offset, singular.ok = singular.ok, ...) :
##  extra argument 'family' will be disregarded
```

```r
summary(training_binary_model)
```

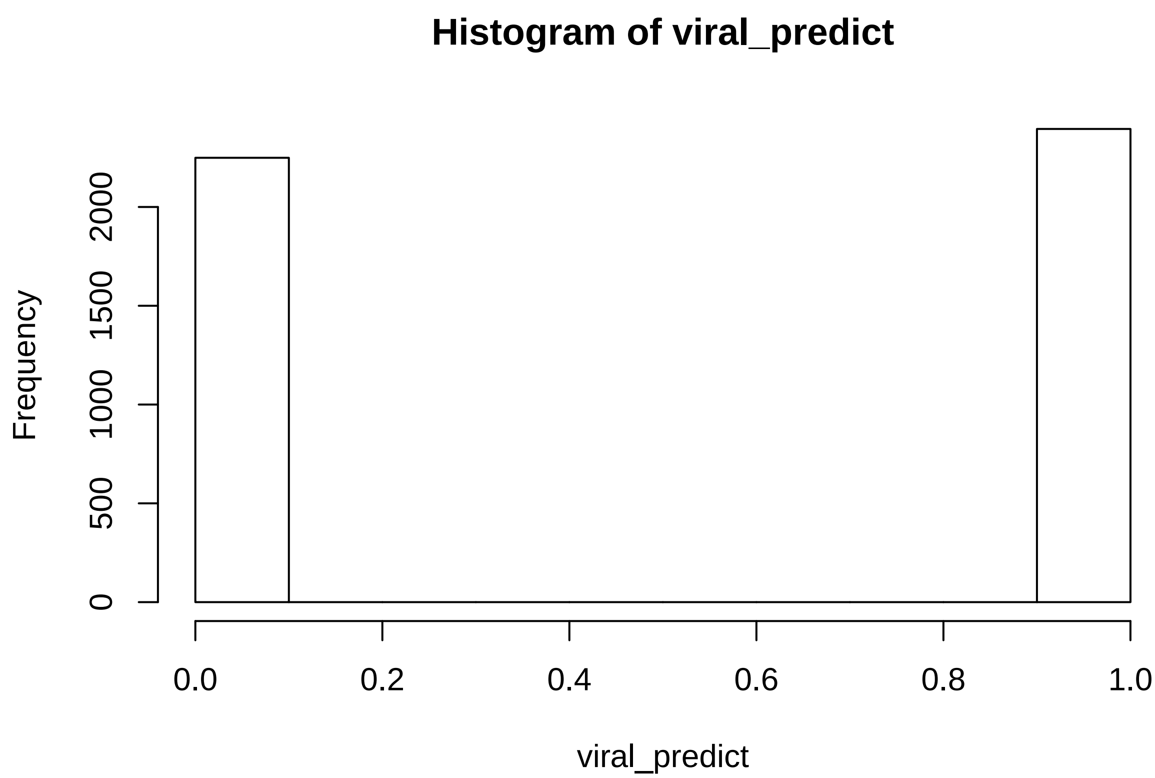
```
##
## Call:
## lm(formula = viral ~ num_hrefs + num_keywords + data_channel_is_entertainment +
##     data_channel_is_bus + data_channel_is_socmed + data_channel_is_world +
##     weekday_is_saturday + weekday_is_sunday + global_rate_positive_words,
##     data = bin_train, family = "binomial")
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -1.0595 -0.4489 -0.2778  0.4539  0.7471
##
## Coefficients:
##                                Estimate Std. Error t value Pr(>|t|)
## (Intercept)                   0.4512758  0.0127768  35.320  < 2e-16 ***
## num_hrefs                     0.0027534  0.0002275  12.102  < 2e-16 ***
## num_keywords                  0.0104096  0.0014009   7.431 1.11e-13 ***
## data_channel_is_entertainment -0.2118549  0.0074087 -28.596  < 2e-16 ***
## data_channel_is_bus           -0.0838751  0.0078284 -10.714  < 2e-16 ***
## data_channel_is_socmed         0.1254564  0.0115251  10.885  < 2e-16 ***
## data_channel_is_world         -0.2367060  0.0070932 -33.371  < 2e-16 ***
## weekday_is_saturday            0.2136292  0.0107143  19.939  < 2e-16 ***
## weekday_is_sunday              0.1655644  0.0101682  16.283  < 2e-16 ***
## global_rate_positive_words     0.1347303  0.1528647   0.881    0.378
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.4793 on 34990 degrees of freedom
## Multiple R-squared:  0.08126,    Adjusted R-squared:  0.08102
## F-statistic: 343.9 on 9 and 34990 DF,  p-value: < 2.2e-16
```

Creating set of probability predictions on the testing set

```r
#generate results
class_predicted<-predict.glm(training_binary_model,newdata=bin_test,type = "response")
hist(class_predicted)
```




```r
viral_predict = ifelse(class_predicted >= .5,1,0)
hist(viral_predict)
```



Comparing predictions to actual virality:

```
viral_actual = ifelse(bin_test$viral==1,1,0)
#creating confusion matrix
class_conf_viral = table(viral_actual, viral_predict)
class_conf_viral
```

```
##              viral_predict
## viral_actual    0    1
##            0 1396  932
##            1  853 1463
```

Finding average values:

```
class_correct_neg <- mean(c(1389,1365,1348,1419,1450))
class_false_neg <- mean(c(818,834,771,821,784))
class_correct_pos <- mean(c(1487,1486,1550,1479,1439))
class_false_pos <- mean(c(950,959,975,925,971))
```

Converting average values into confusion matrix:

```
library(knitr)
library(kableExtra)
bin_avg_conf_viral <- data.frame(
  Actually_Viral = c("No", "Yes"),
```

Table 2: Confusion Matrix

Actually_Viral	Predicted_Not	Predicted_Viral
No	1394.2	956.0
Yes	805.6	1488.2

```
  Predicted_Not = c(class_correct_neg,class_false_neg),
  Predicted_Viral = c(class_false_pos,class_correct_pos)
)
kable(bin_avg_conf_viral, caption = "Confusion Matrix") %>% row_spec(0,bold=FALSE) %>% kable_styling()
```

```
#total error rate
(class_false_neg+class_false_pos)/nrow(bin_test) #37.93%
```

```
## [1] 0.3793282
```

```
#false positive rate
class_false_pos/(class_false_pos+class_correct_neg) #40.68%
```

```
## [1] 0.4067739
```

```
#false negative rate
class_false_neg/(class_false_neg+class_correct_pos) #35.12%
```

```
## [1] 0.3512076
```

The classification model is more successful than the linear model and far outperformed the null model as both the false negative and false positive error rates are below 50%. This model may be useful, but it is not very accurate or predictive with an adjusted R-squared of 0.0810. The model's total error rate is 37.93%, false positive rate is 40.68%, and false negative rate is 35.12%. The classification model is more evenly split in its predictions that the regression model, predicting more articles to go viral than not which mirrors the data set.

#Summary ### The Classification approach (threshold first and regress second) performs better than the Regression approach (regress first and threshold second). This approach fits the problem better because Mashable is look for a yes or no response on if each article they publish goes viral, which ultimately comes down to a set of probabilities. By framing this question as a logistic regression, each predictor is given the chance to affect the probability that an article with certain features will reach 1400 shares (going viral) rather than just attempting to predict how many shares an article will have. The classification approache proved to deliver a more useful and accurate model, with a total error rate of 37.93% compared to 46.51% and adjusted R-squared of 0.0810 compared to 0.0106. The Regression model predicted that nearly all articles would go viral, making it nearly useless.