Exercise #5

Multiple Regression and k-NN with R

Brian Schweigler; 16-102-071

15/04/2022

Preliminaries

Load required libraries

```
library(dplyr)
```

```
## Warning: package 'dplyr' was built under R version 4.0.5
```

```
##
## Attaching package: 'dplyr'
```

```
## The following objects are masked from 'package:stats':
##
##     filter, lag
```

```
## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```
library(tidyr)
```

```
## Warning: package 'tidyr' was built under R version 4.0.5
```

```
library(ggplot2)
```

```
## Warning: package 'ggplot2' was built under R version 4.0.5
```

```
library(FNN)
```

```
## Warning: package 'FNN' was built under R version 4.0.5
```

Loading the education dataset:

```
setwd(dirname(rstudioapi::getActiveDocumentContext()$path))

education_df = read.csv("EducationBis.txt", header = TRUE, sep = "\t", comment.char = "#")
```

The data has already been modified and removed of outliers.

Loading the computers dataset:

```
setwd(dirname(rstudioapi::getActiveDocumentContext()$path))

computers_df = read.csv("Computers.txt", header = TRUE, sep = "\t"  , comment.char = "#")
```

As we know from exercise 4, Computers.txt has no outliers discernible (without additional information).

Loading the cars dataset and cleaning NAs:

Loading the cars dataset:

```r
setwd(dirname(rstudioapi::getActiveDocumentContext()$path))

cars_df = read.csv("Cars.txt", header = TRUE, sep = "\t", comment.char = "#")
```

There are 6 NAs for horsepower that need to be removed -> delete the whole corresponding rows. These are also mentioned in Cars.pdf.

```r
cars_df_cleaned <- cars_df[!is.na(cars_df$horsepower),]
```

Besides the NAs found for horspower, the data seems to be good (without additional information.)

1. Apply the lm() function to all the data (education) and interpret the output you obtain with R

```
lm_education = lm(Wage ~ Education + Gender ,data = education_df)
summary(lm_education)
```

```
##
## Call:
## lm(formula = Wage ~ Education + Gender, data = education_df)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -243.473  -76.073    0.354   73.126  280.275
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) -569.784     23.100  -24.67   <2e-16 ***
## Education    397.975      1.543  258.00   <2e-16 ***
## Gendermale   597.904      9.173   65.18   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 100.1 on 494 degrees of freedom
## Multiple R-squared:  0.9931, Adjusted R-squared:  0.9931
## F-statistic: 3.544e+04 on 2 and 494 DF,  p-value: < 2.2e-16
```

There is likely to be a linear relationship between wage and education, as can be guessed by the low p-values.

2. Remove the variables that are not good predictors and use all the remaining ones to build a multiple regression model. Which variables do you use? Does your model explain something? Explain the model building strategy you have applied. Interpret the most important values of the final model you obtain with R.

Variables such as the model or vendor are not applicable to predict performance. ERP should not be used, as it is the result of a linear prediction of PRP (based on the available predictor values).

This leaves us with:

```
lm_computers = lm(PRP ~ MYCT + MMIN + MMAX + CACH + CGMIN + CHMAX, data = computers_df)
summary(lm_computers)
```

```
##
## Call:
## lm(formula = PRP ~ MYCT + MMIN + MMAX + CACH + CGMIN + CHMAX,
##     data = computers_df)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -195.82  -25.17    5.40   26.52  385.75
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) -5.589e+01  8.045e+00  -6.948 5.00e-11 ***
## MYCT         4.885e-02  1.752e-02   2.789   0.0058 **
## MMIN         1.529e-02  1.827e-03   8.371 9.42e-15 ***
## MMAX         5.571e-03  6.418e-04   8.681 1.32e-15 ***
## CACH         6.414e-01  1.396e-01   4.596 7.59e-06 ***
## CGMIN       -2.704e-01  8.557e-01  -0.316   0.7524
## CHMAX        1.482e+00  2.200e-01   6.737 1.65e-10 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 59.99 on 202 degrees of freedom
## Multiple R-squared:  0.8649, Adjusted R-squared:  0.8609
## F-statistic: 215.5 on 6 and 202 DF,  p-value: < 2.2e-16
```

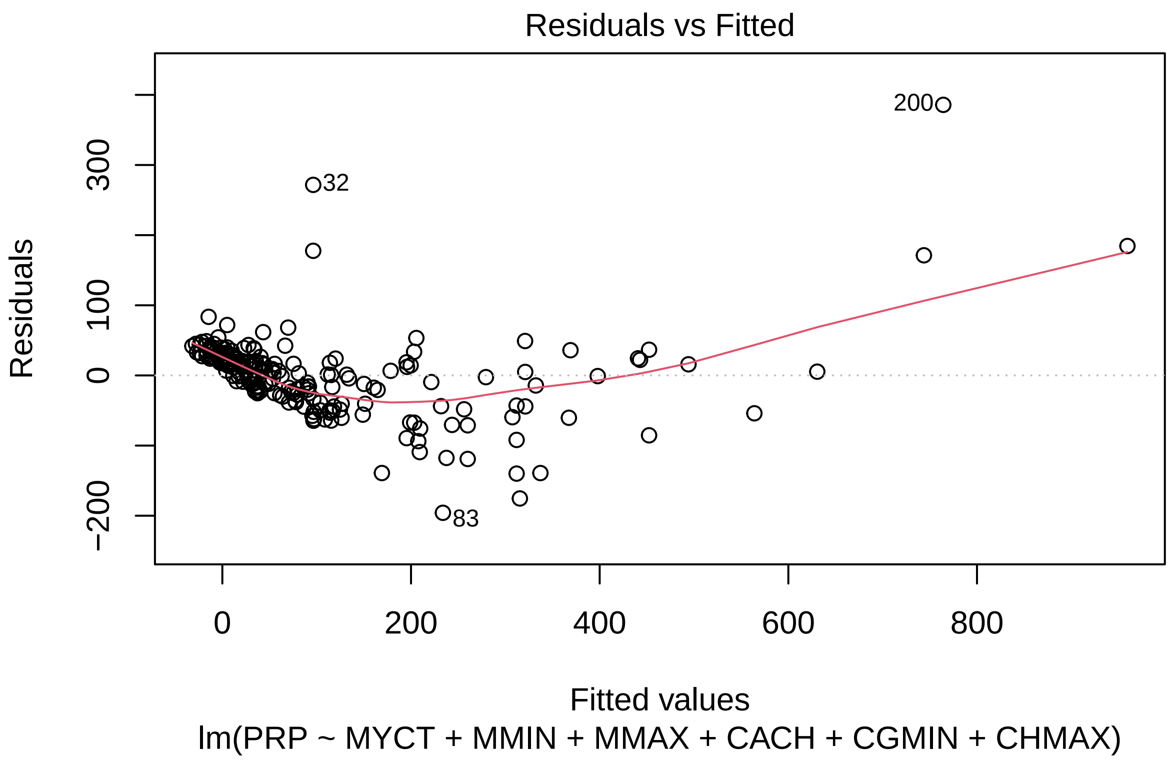
From the output, we can tell that MMIN, MMAX, CACH and CHMAX, are very likely to be in a liner relation with PRP, as their p value are very small. MYCT could also be used to a less extent, as its p value of 0.0058 is low enough to reject H_0.

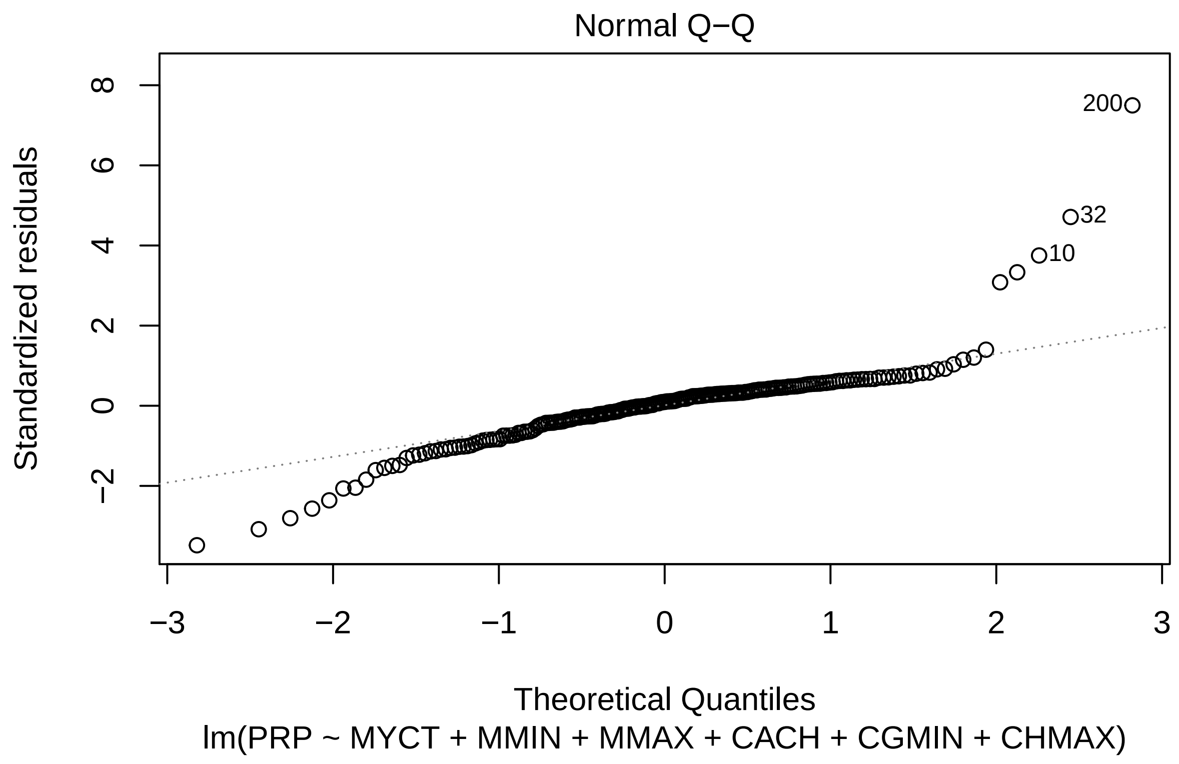
CGMIN has a p value of 0.75, which too high (unable to reject H0), and thus cannot we do not know if there is a linear relationship to PRP.

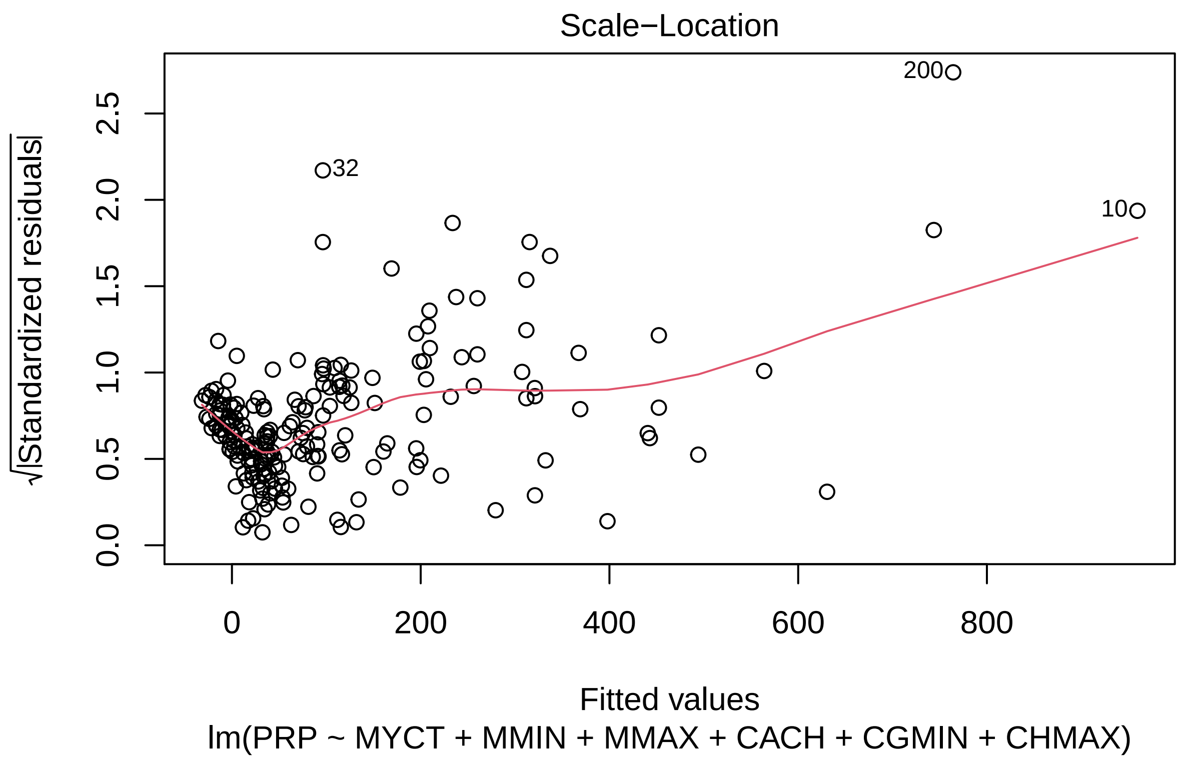
3. Visualize graphically the relationship found by you model (by considering the most important variable). Do you notice outlier(s) or hard observations to predict with your model?

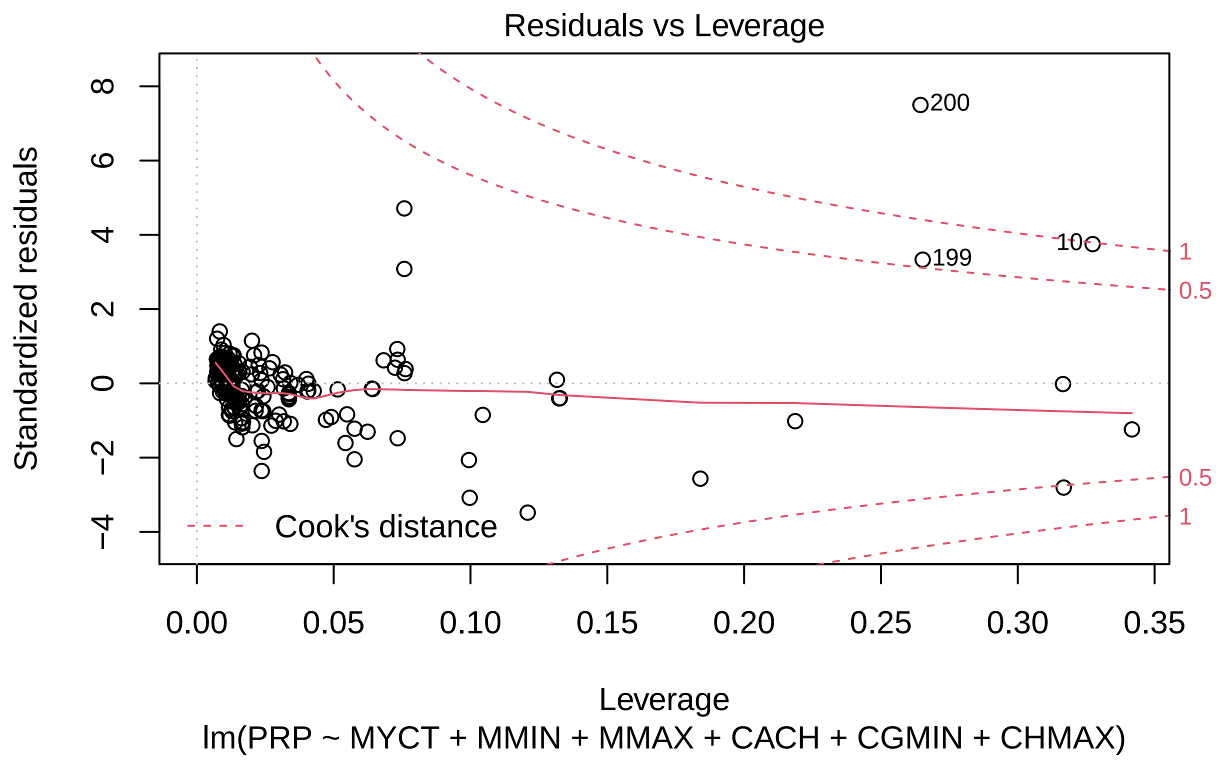
We can also take a look at some of the models' plots

```
plot(lm_computers)
```




Normal Q−Q
Standardized residuals
Theoretical Quantiles
lm(PRP ~ MYCT + MMIN + MMAX + CACH + CGMIN + CHMAX)
200
32
10
−3
−2
−1
0
1
2
3
8
6
4
2
0
−2

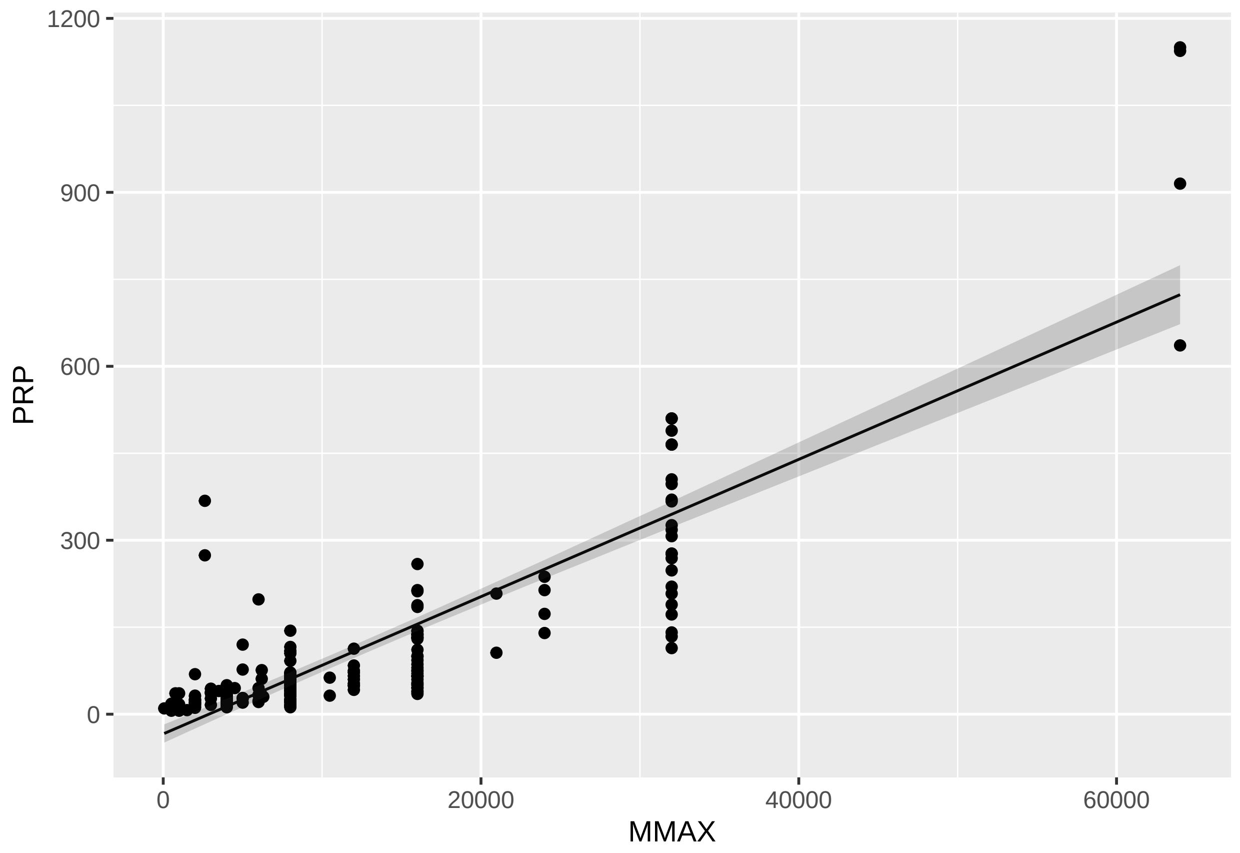

Scale−Location
√|Standardized residuals|
0.0
0.5
1.0
1.5
2.0
2.5
0
200
400
600
800
Fitted values
lm(PRP ~ MYCT + MMIN + MMAX + CACH + CGMIN + CHMAX)
200
32
10



The distribution of residuals implies that there is either a non-linear relationship or an additional set of linear relations that we did not regard. In the QQ plot, the residuals follow a roughly normal distribution, except for the head and tail ends of the distribution, where there are a few outliers.

Lastly we can reduce our model to one variable (MMAX), and plot this prediction.

```
lm_computers = lm(PRP ~ MMAX, data = computers_df)
computers_df.predict = cbind(computers_df, predict(lm_computers, interval = 'confidence'))
ggplot(computers_df.predict, aes(x = MMAX, y = PRP)) +
  geom_point() +
  geom_line(aes(x = MMAX, y = fit)) +
  geom_ribbon(aes(ymin = lwr, ymax = upr), alpha = 0.2)
```



There quite a few outliers at all stages, making this definitely not a perfect model.

2b. and 3b: The same as 2 & 3, but with the cars dataset:

Obviously name can't be used to explain the mpg, the origin is unlikely to be of impact too. Furthermore, the year might corrolate with mpg if the efficiency increased over time, but is not a direct predictor for mpg.

This leaves us with:

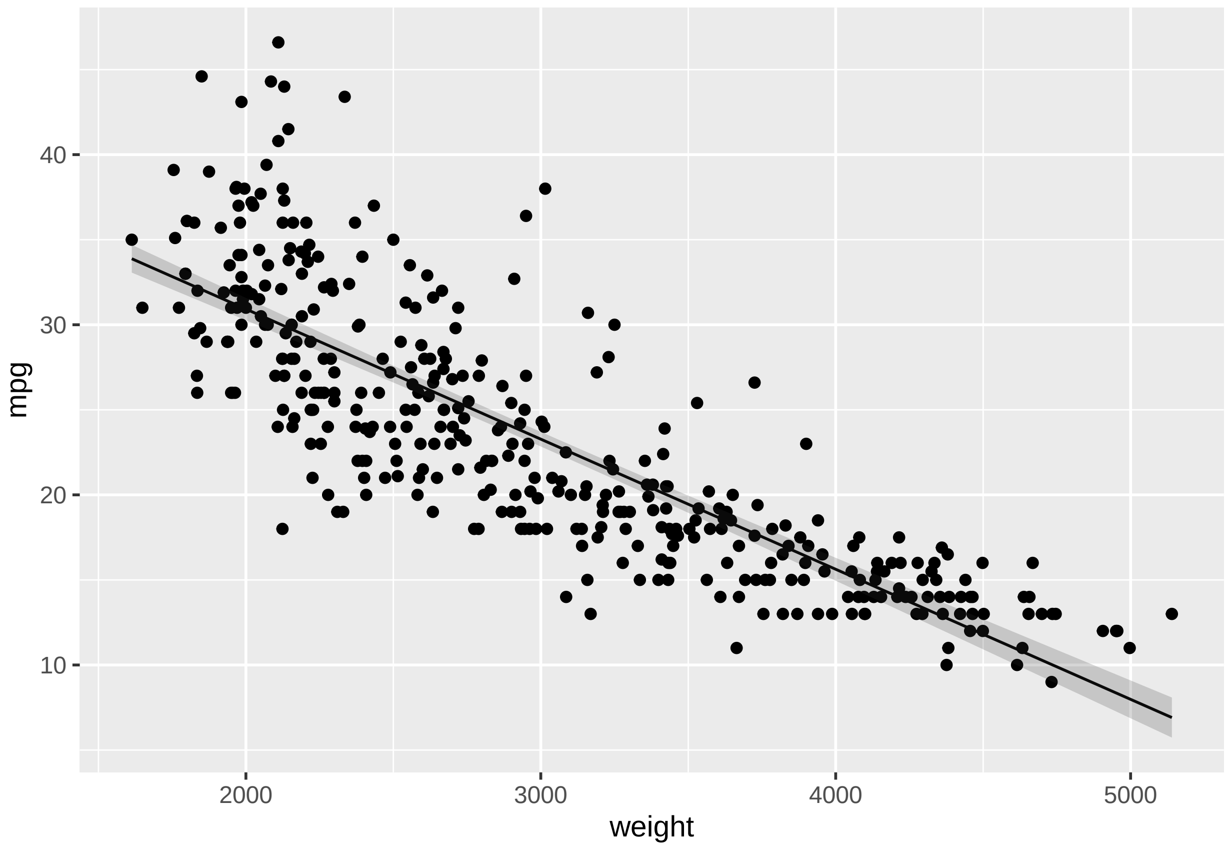
```
lm_cars = lm(mpg ~ cylinders + displacement + horsepower + weight + acceleration + year, data = cars_df
summary(lm_cars)
```

```
##
## Call:
## lm(formula = mpg ~ cylinders + displacement + horsepower + weight +
##     acceleration + year, data = cars_df_cleaned)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -8.6927 -2.3864 -0.0801  2.0291 14.3607
##
## Coefficients:
##                Estimate Std. Error t value Pr(>|t|)
## (Intercept)  -1.454e+01  4.764e+00  -3.051  0.00244 **
## cylinders    -3.299e-01  3.321e-01  -0.993  0.32122
## displacement  7.678e-03  7.358e-03   1.044  0.29733
## horsepower   -3.914e-04  1.384e-02  -0.028  0.97745
## weight       -6.795e-03  6.700e-04 -10.141  < 2e-16 ***
## acceleration  8.527e-02  1.020e-01   0.836  0.40383
## year          7.534e-01  5.262e-02  14.318  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 3.435 on 385 degrees of freedom
## Multiple R-squared:  0.8093, Adjusted R-squared:  0.8063
## F-statistic: 272.2 on 6 and 385 DF,  p-value: < 2.2e-16
```

The car's weight has an inverse correlation with the mpg The Year has a slight correlation, implying that efficiency of cars has kept increasing over time.

Limiting ourselves to one attribute, we will be using the weight. The manufacturing year could probably be used as well.

```
lm_cars = lm(mpg ~ weight, data = cars_df_cleaned)
cars_df_cleaned.predict = cbind(cars_df_cleaned, predict(lm_cars, interval = 'confidence'))
ggplot(cars_df_cleaned.predict, aes(x = weight, y = mpg)) +
  geom_point() +
  geom_line(aes(x = weight, y = fit)) +
  geom_ribbon(aes(ymin = lwr, ymax = upr), alpha = 0.2)
```



The higher the weigt, the lower the miles per gallon. Do note that the variance is quite high, but on both sides. It seems a "good enough" approach to use the weight to get the mpg of the vehicles.

5. Write a R function to compute the distance between two observations. As a parameter, we can ask to compute the Euclidean distance (by default) or the L1-norm.

So, a function either L2 (euclidean) or L1 norm:

```r
dist_L1_L2 = function(a, b, L2 = TRUE) {

  if (!is.vector(a) || !is.vector(b)) {
    return("a and b must be vectors!")
  }

  if (length(a) != length(b)) {
    return("a and b must be of equal length!")
  }

  if (L2) {
    diff = abs(a - b)**2
    return(sqrt(sum(diff)))
  } else {
      diff = abs(a - b)
        return(sum(diff))
  }
}
```

6. Consider both the Computers and Cars datasets. In Exercise #4 and in the previous questions of Exercise #5, you have proposed a regression model based on a single linear predictor or by taking in account many predictors. As a new regression model, apply the knn algorithm. You're free to select the value of k, but you have to justify your choice over other possible values.

We will be using the FNN package loaded in the preliminaries section. First, we will exclude vendor, model and ERP in the dataframe:

```
computers_df_short = computers_df[, 3:9]
```

This needs to be normalized:

```
normalize = function(x) {
  (x - min(x)) /
        (max(x) - min(x))
}
```

```
computers_df_norm = as.data.frame(lapply(computers_df_short, normalize))
```

Variables required for FNN:

```
n = nrow(computers_df_norm)
train = sample(1:n, size = n/2)
test = (1:n)[-train]
```

Remove PRP from training data:

```
computers_train = computers_df_norm[train, -7]
computers_train_labels = computers_df_norm[test, 7]
computers_test = computers_df_norm[test, -7]
computers_test_labels = computers_df_norm[test, 7]
```

Now to find k with lowest Mean Squared Error

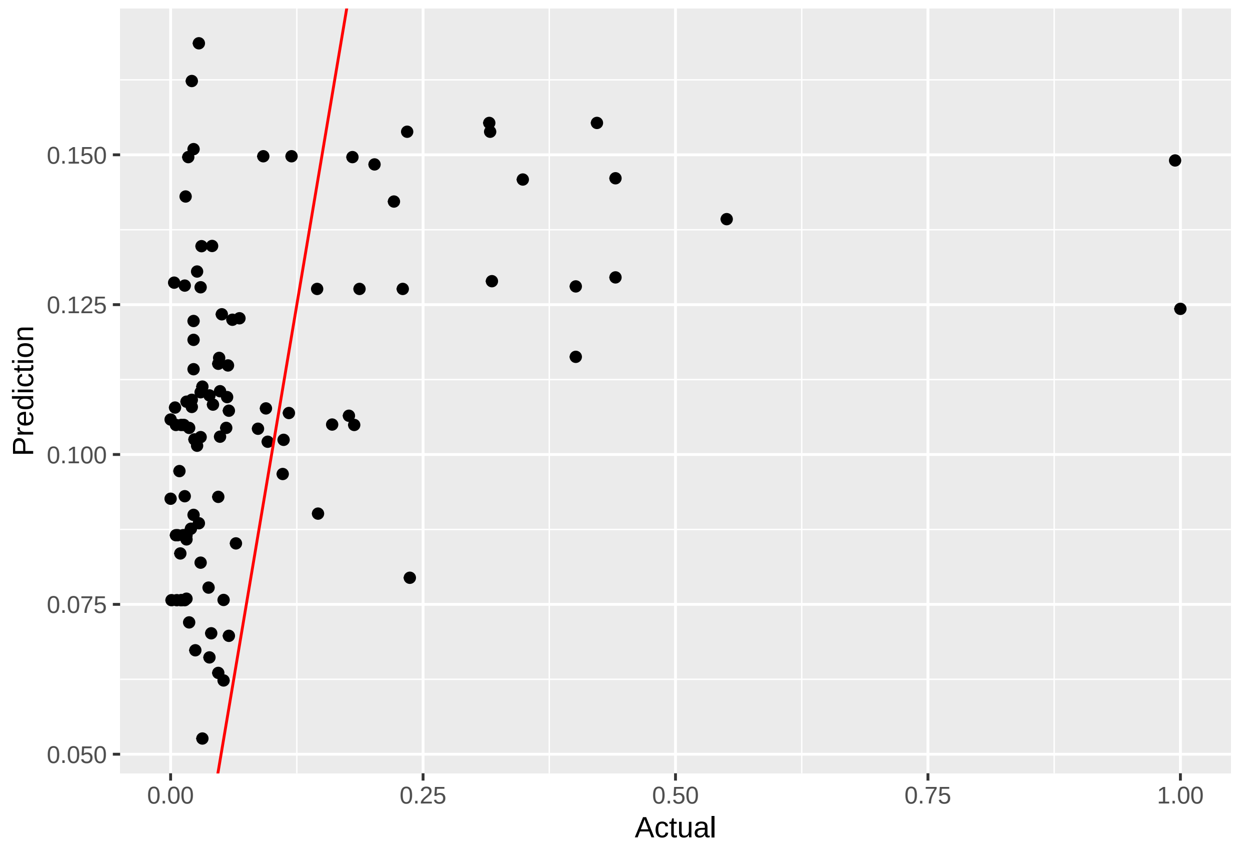
```
k_best = 0
mse_best = 10000
for (k in 1:100) {
  computers_knn =  knn.reg(
    train=computers_train,
    test=computers_test,
    y = computers_train_labels,
    k = k
  )

  mse = mean((computers_knn$pred - computers_test_labels)^2)

  if (mse < mse_best) {
    mse_best = mse
    k_best = k
  }
}
computers_knn =  knn.reg(
  train=computers_train,
  test=computers_test,
  y = computers_train_labels,
  k = k_best
)
```

```r
mse = mean((computers_knn$pred - computers_test_labels)^2)
print(sprintf("Best k = %d with MSE = %f", k_best, mse))
```

```
## [1] "Best k = 25 with MSE = 0.026001"
```

Plot this information:

```r
computers_predicted = data.frame(Prediction = computers_knn$pred, Actual = computers_train_labels)
ggplot(computers_predicted, aes(x = Actual, y = Prediction)) +
  geom_point() +
  geom_abline(slope = 1, intercept = 0, color = "red")
```



With such small MSE, the random train/test split will lead to multiple runs having different k values. k = 69 was the best in one run with MSE 0.018, but k = 8 has MSE 0.022, which a performance improvement with minimal MSE loss. So we could use 8 here.

Same for the cars dataset

We will be using the FNN package loaded in the preliminaries section. First, we will exclude name:

```
cars_df_short = cars_df_cleaned[, 1:8]
```

This needs to be normalized:

```
normalize = function(x) {
  (x - min(x)) /
        (max(x) - min(x))
}
```

```
cars_df_norm = as.data.frame(lapply(cars_df_short, normalize))
```

Variables required for FNN:

```
n_car = nrow(cars_df_norm)
train_car = sample(1:n_car, size = n_car/2)
test_car = (1:n_car)[-train_car]
```

Remove MPG from training data:

```
cars_train = cars_df_norm[train_car, -1]
cars_train_labels = cars_df_norm[test_car, 1]
cars_test = cars_df_norm[test_car, -1]
cars_test_labels = cars_df_norm[test_car, 1]
```

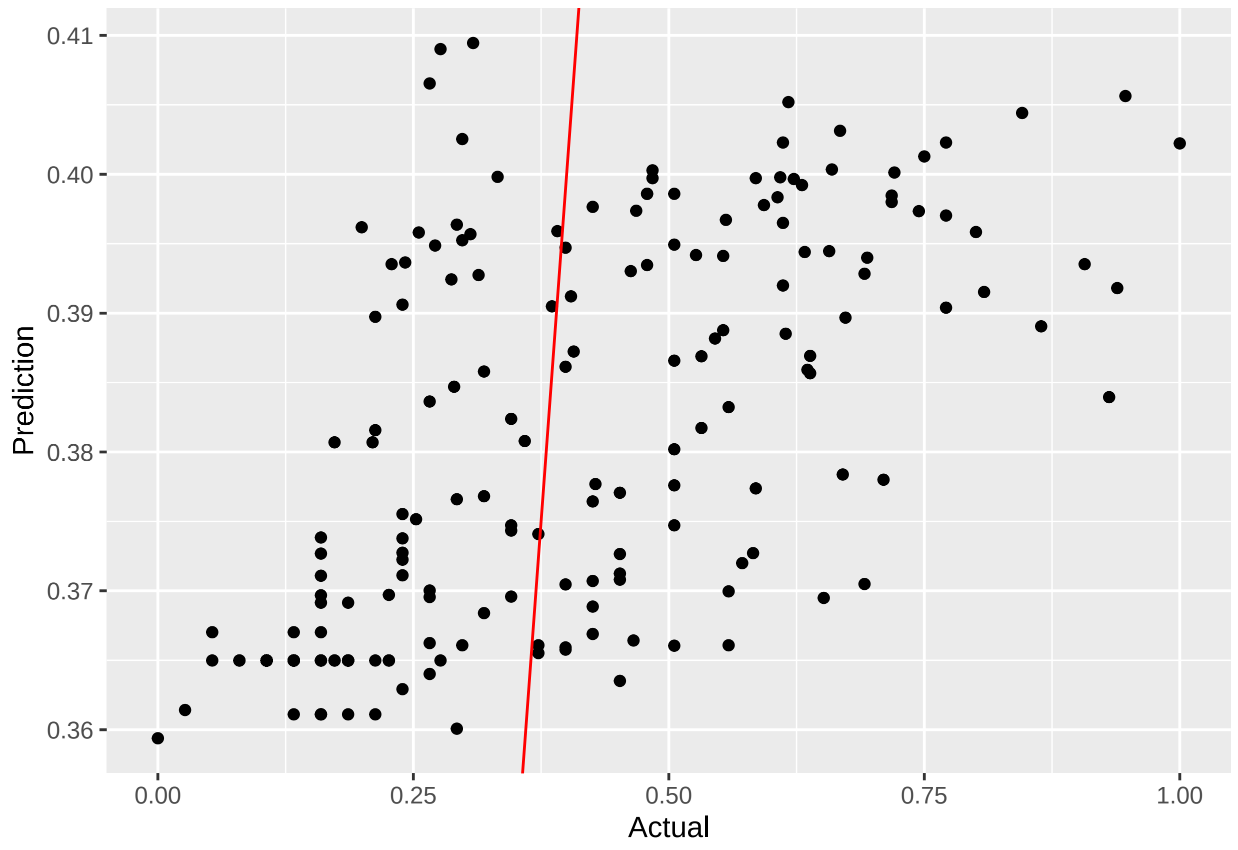
Now to find k with lowest Mean Squared Error

```
k_best = 0
mse_best = 10000
for (k in 1:100) {
  cars_knn =  knn.reg(
    train = cars_train,
    test = cars_test,
    y = cars_train_labels,
    k = k
  )

  mse = mean((cars_knn$pred - cars_test_labels)^2)

  if (mse < mse_best) {
    mse_best = mse
    k_best = k
  }
}
cars_knn =  knn.reg(
  train = cars_train,
  test = cars_test,
  y = cars_train_labels,
  k = k_best
)
mse = mean((cars_knn$pred - cars_test_labels)^2)
print(sprintf("Best k = %d with MSE = %f", k_best, mse))
```

```
## [1] "Best k = 85 with MSE = 0.046534"
```

Plot this information:

```
cars_predicted = data.frame(Prediction = cars_knn$pred, Actual = cars_train_labels)
ggplot(cars_predicted, aes(x = Actual, y = Prediction)) +
  geom_point() +
  geom_abline(slope = 1, intercept = 0, color = "red")
```



The best k-value is not constan (changes with the sampling) for this dataset too. Nonetheless, a small value (larger than 2 and smaller than 10) is likely to provide a good trade-off between performance and computation time.