INTRODUCTION TO MACHINE LEARNING (NPFL054)
A template for Homework #2

Name: Jakub Hejhal

School year: 2nd year

- **Provide answers for the exercises.**
- **For each exercise, your answer cannot exceed one sheet of paper.**

1.1 Multiple linear regression

```
Coefficients:
 (Intercept)     cylinders  displacement    horsepower  
  -17.218435     -0.493376      0.019896     -0.016951  
      weight  acceleration          year        origin  
   -0.006474      0.080576      0.750773      1.426140  
```

Intercept **-17.218435**

Car with no cylinders, zero displacement, weighing nothing etc. is not really a car, so talking about its mpg does not really make much sense.

Cylinders **-0.493376**

More cylinders generaly means more fuel is burned per cycle, so the mpg goes down.

Displacement **0.019896**

When everything is kept that same, and the only thing changing is engine displacement, the mpg goes slightly up, which is probably because engines with bigger displacements and the same power output are probably running leaner fuel-air mixtures, which can be more efficient. But if we take a look only at mpg ~ displacement relationship, it is completely opposite (coefficient of -0.06005), which agrees with our intuition that bigger engines are used in heavier cars, which naturaly have lower mpg. And indeed, displacement and weight is highly positively correlated (cor(displacement, weight) = 0.9329944)

Horsepower **-0.016951**

Powerful cars are either sports cars, which have notoriously bad fuel economy, or they are heavy. Either way with higher hp, mpg goes down.

Weight **-0.006474**

Heavier cars have worse fuel economy for the reasons above. The coefficient is quite small though, which corresponds to the fact, that weight feature has much bigger standard deviation than other features.

Acceleration **0.080576**

If acceleration of a car increases, while all other features stay constant, mpg goes up, which may be due to better aerodynamics, though the relationship is weak.

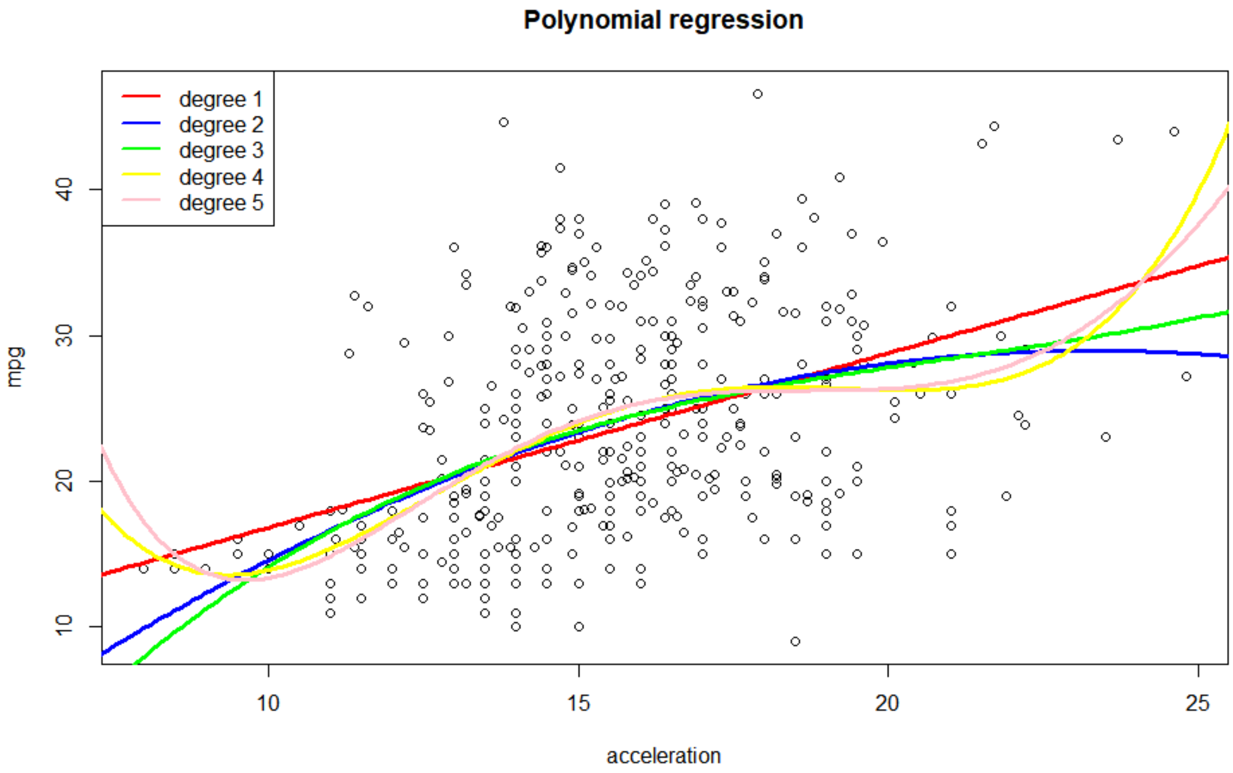
Year **0.750773**

Newer car has better mpg, probably due to better technology, turbocharged engines etc.

Origin **1.426140**

American cars in the 70s and 80s had terrible mpg, but because gas was cheap, it wan’t a problem.

1.2 Polynomial regression



```
polynomial degree: 1, R^2: 0.179207050156255
polynomial degree: 2, R^2: 0.193964011032176
polynomial degree: 3, R^2: 0.195508000328506
polynomial degree: 4, R^2: 0.213597901585262
polynomial degree: 5, R^2: 0.214801832251011
```

2.1 Binary attribute `mpg01` and its entropy

Entropy of d$mpg01 is 1, because of how it is constructed. Half of cars have mpg higher than median and half have mpg lower than median, so there will be the same number of ones and zeros in the mpg01 vector. The entropy is therefore 2* 0.5 * log2(2) = log2(2) = 1.

2.3 Trivial classifier accuracy

Accuracy of MFC trained on train data, evaluated on test data is 43.59%.

Note, that because train and test data have the same number of ones and zeros in mpg01 vector, MFC, that selects the value, that is most common in train data, will ALWAYS have accuracy less or equal to 50% when evaluated on test data. That is because if zeros are more common in train dataset, they will be less common in test dataset and and vice versa.

2.4 Logistic regression – training and test error rate, confusion matrix, interpretation

```
GLM error rate on training data: 0.0987261146496815
GLM error rate on test data: 0.0769230769230769
```

Test error rate is usually greater than train error rate, which is not the case here. That is most likely caused by the small size of the dataset (only 78 cars in test set), so the confidence interval is quite spread out.

```
Confusion matrix test.true vs test.predicted:
```

```
    predicted
true  0  1
   0 29  5
   1  1 43
```

Model parameters

```
Call:  glm(formula = mpg01 ~ ., family = binomial(link = "logit"), data = 
d.train)

Coefficients:
 (Intercept)     cylinders  displacement    horsepower  
  -17.379228      0.094411     -0.001464     -0.049519  

  weight  acceleration          year        origin  
 -0.003941      0.019771      0.418720      0.491627  
```

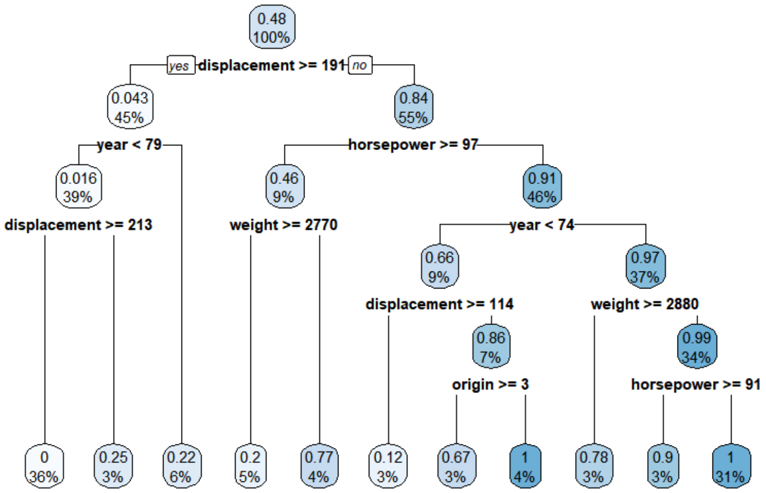
Model parameters are quite similar to the linear regression hypotesis parameters. The minor differences are due to the fact, that output of linear model is fed to the sigmoid squishification function and because logistic regression uses Log loss instead of MSE. Though the relationship between features and probabilities predicted by the model is non-linear, it preserves monotonicity. So if for example horsepower increases, because horsepower coefficient is negative, the probability of high gas milage goes down. Similarly, if year increases, the probability of high gas milage goes up quite a bit. So the explanation of all the hypothesis parameters would be exactly the same as in excercise 1.1.

2.5 Decision tree algorithm – training and test error rate, `cp` parameter

a)

```
train error rate:
     0.0605095541401274
test error rate:
     0.0769230769230769
```

When choosing the best cp parameter, the rule of thumb is to choose the lowest level (minimum number of splits), where xerror falls into the ±1 xstd range of min(xerror), i.e., xerror < min(xerror) + xstd. This method takes into account the variability of xerror resulting from cross-validation.



tree with cp=0.0001

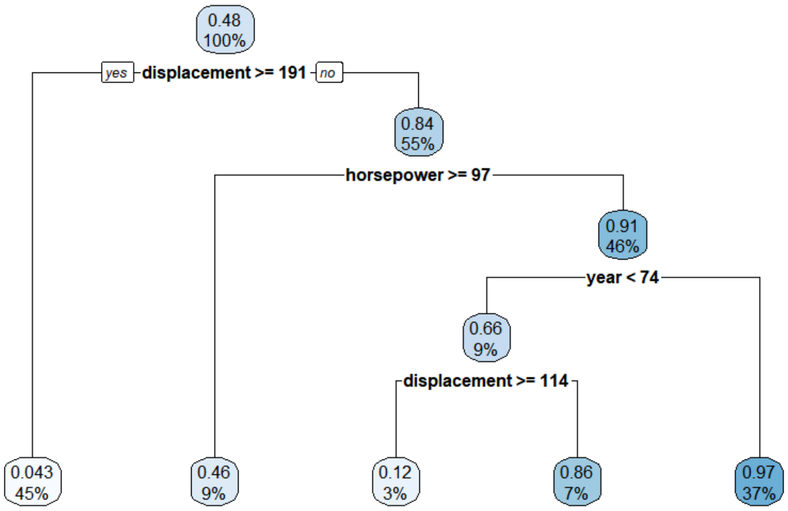
cptable

```
       CP          xerror     xstd
1      0.6271746   1.00459    0.0040271
2      0.0597740   0.43435    0.0565681
3      0.0348950   0.36268    0.0534779
4      0.0287756   0.33757    0.0529388
5      0.0084740   0.31825    0.0526457
6      0.0072867   0.31541    0.0509605
7      0.0059578   0.32344    0.0517137
8      0.0048046   0.32520    0.0519322
9      0.0011571   0.32399    0.0512734
10     0.0001000   0.32156    0.0505955
```

min(xerror) + xstd = 0.31541 + 0.0509 = 0.366, So the optimal cp parameter is 0.0348950.

```
optimal cp train error rate :
     0.0828025477707006
optimal cp test error rate :
0.115384615384615
```

Test error rate is bigger in this case, but this is probably due to the small test set size and thus big varience of test results.



Optimal cp tree

2.6 Naive Bayes algorithm – precision and recall

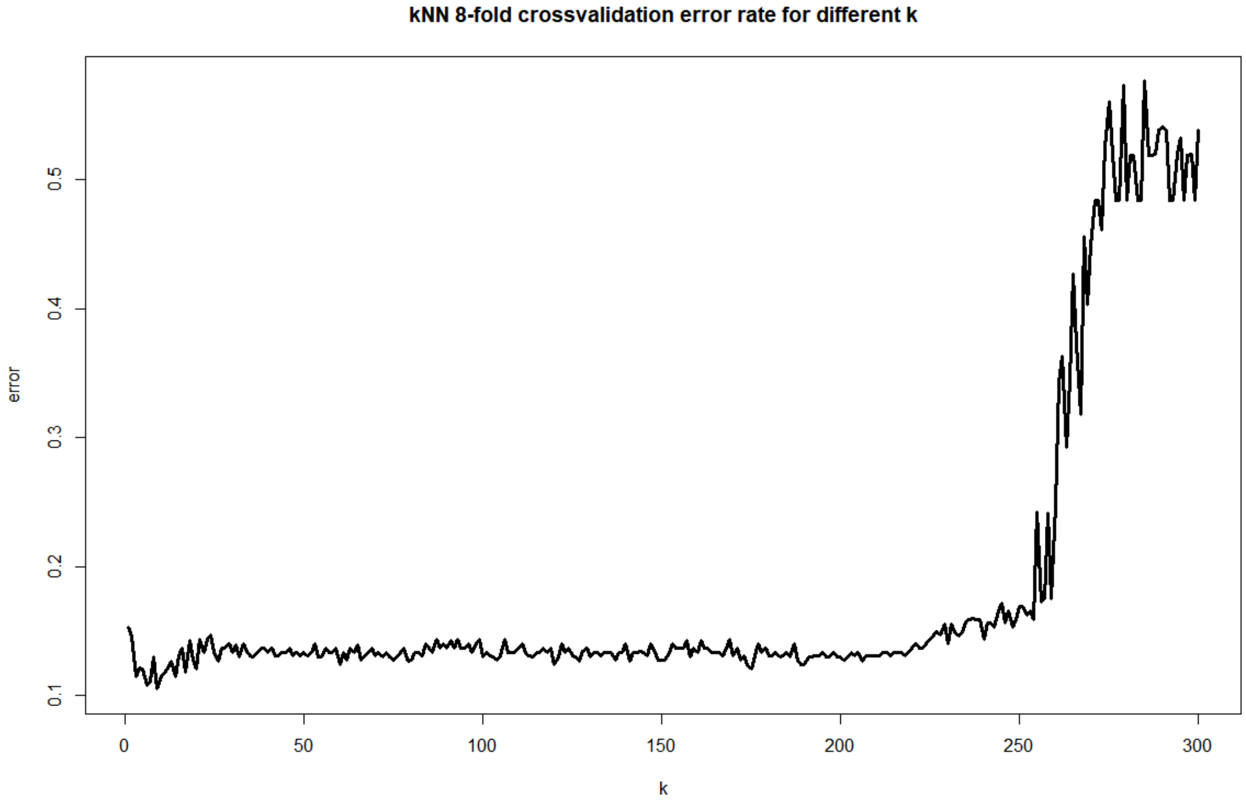
```
train accuracy:   0.898089171974522
test accuracy:    0.923076923076923
```

```
confusion matrix:
```

```
         true
test.pred  0  1
        0 30  2
        1  4 42
```

```
test precision:   0.954545454545455
test recall:      0.91304347826087
```

2.7 k-NN -- cross validation error rate



This graph show that wide range of k yields quite similar results, but as the k is near the train set size, kNN turns into MFC classifier and performs noticebly worse.