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

Name: Vilém Zouhar

School year: 2nd

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

1.1 Multiple linear regression

```
 (Intercept)     cylinders  displacement    horsepower        weight
   -17.95460      -0.48971       0.02398      -0.01818      -0.00671

acceleration          year   is_european   is_japanese
     0.07910       0.77703       2.63000       2.85323
```

Model interpretation (from highest to lowest numbers)

is_european, is_japanese

We created dummy variables (*is_european, is_japanese*) from input variable *origin* (from the ISLR package: 1. American, 2. European, 3. Japanese). We used *is_american* as base class, because if included, it would be detected as a linearly dependent variable. We can interpret it as: the more American the car, the less mpg it has, with Japanese cars having slightly more mpg.

year

The newer the car is, the higher mpg it has.

cylinders

The more cylinders the car has, the less mpg it achieves. This is presumably because higher number of cylinders indicate a sports car, not an economical car.

engine displacement

Higher engine volume results in more mpg.

horsepower/weight

Higher horsepower/weight implies somewhat less mpg. Probably the same reason as with cylinders.

acceleration

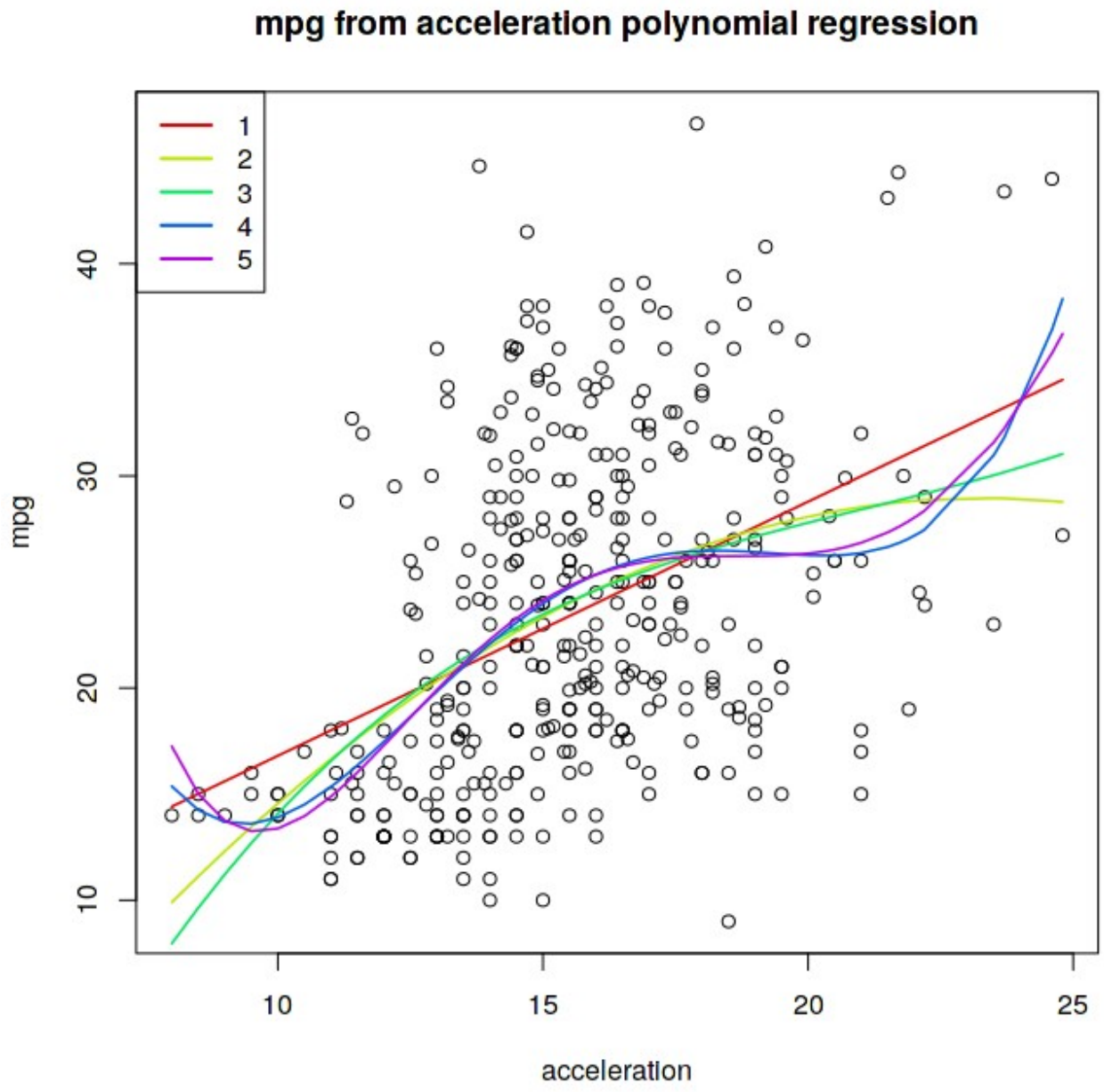
Acceleration in this case is the time it takes the car to accelerate from 0 to 60 mpg (in sec.). Hence less agile cars have higher mpg. Same reason as above.

intercept

Base mpg, if all features were zero. This will naturally not be the case in reality (zero acceleration, weight, etc..).

The features were unscaled and hence we don’t know which feature is the most important one. (eg. cylinders seems quite significant, but this number spans from 2 to 8)

1.2 Polynomial regression




```
Degree 1 R^2: 0.179207050156255
Degree 2 R^2: 0.193964011032176
Degree 3 R^2: 0.195508000328506
Degree 4 R^2: 0.213597901585262
Degree 5 R^2: 0.214801832251011
```

2.1 Binary attribute `mpg01` and its entropy

Since there is no example of a car with **mpg** equal to the global median (even number of examples with two different middle values), it does not matter where we categorize these mpg median cars.

H[mpg01] = 1

This should be obvious, because half of the cars will have **mpg01** equal to **1**, the other equal to **0** (even number of examples). Then **P[mpg01 = 1] = P[mpg01 = 0] = 0.5**. From this follows:

- H[mpg01] = P[mpg01 = 1]*log_2(P[mpg01 = 1]) + P[mpg01 = 0]*log_2(P[mpg01 = 0])

- H[mpg01] = 0.5 * log_2(0.5) + 0.5 * log_2(0.5) = log_2(0.5) = -1

Computation in R gives the same result.

2.3 Trivial classifier accuracy

```
MFC accuracy: 46.84%
```

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

---

```
Logistic regression test error rate:  11.39%
Logistic regression train error rate: 7.67%
```

```
                  true test values
predicted test values  0  1
                    0 37  4
                    1  5 33
```

Model:

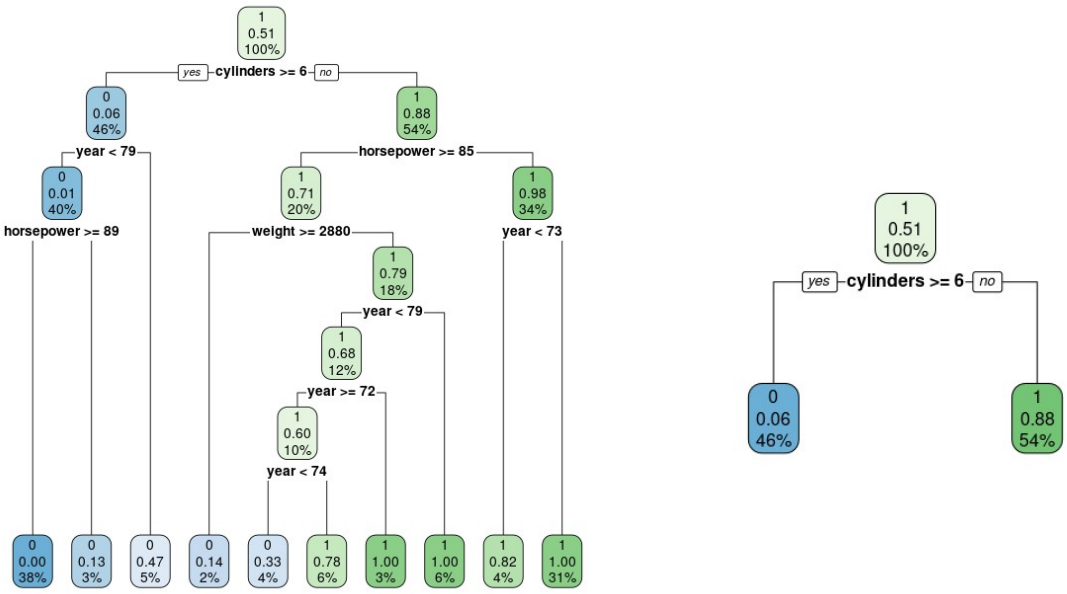
```
 (Intercept)     cylinders  displacement    horsepower        weight
  -20.844548     -0.496219      0.023211     -0.045976     -0.005976

acceleration          year   is_european   is_japanese
    0.060631      0.511699      2.313537      1.514192
```

The model values are somewhat similar to the simple linear regression model (predicting raw mpg) and thus interpretation the same. There are some minor differences between the two: *is_japanese* and *year*. They can both be explained due to different fitting function (logistic regression transforms the output using the sigmoid function) and splitting the mpg into two classes. Thus we can conclude, that for above median mpg it doesn't make much difference whether the car is European or Japanese.

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

---



max decision tree vs optimally pruned decision tree

```
Decision tree (min cp) train error rate: 6.39%
Decision tree (min cp) test error rate:  13.92%
Decision tree (optimal cp) train error rate: 9.27%
Decision tree (optimal cp) test error rate:  11.4%
```

```
           CP    xerror
1 0.811688312 1.1428571
2 0.016233766 0.2077922
3 0.008658009 0.2077922
4 0.000000000 0.1883117
```

```
Optimal cp: 0.016233766
```

Above is the **cptable** from the max decision tree. The minimum is **0.1883117** and the **sd** is **0.4708688**. The first **cp** associated with an **xerror** lower or equal to computed **min + sd (0.6591805)** is **0.016233766**. Hence this is the optimal **cp** parameter. (It is optimal, because it is in **sd** range of **min** and produces the simplest decision tree with this property.) It would be more apparent from a cp graph, but there's no space left on this page.

The above computation can be done with this R code:

```
trm.cpt = trm.default$cptable[, c(1,4)]
trm.o_cp = trm.cpt[trm.cpt[,2] <= min(trm.cpt[,2]) + sd(trm.cpt[,2])][[1]]
```

Different samplings would produce vastly different trees, so this method should be wrapped in cross validation.

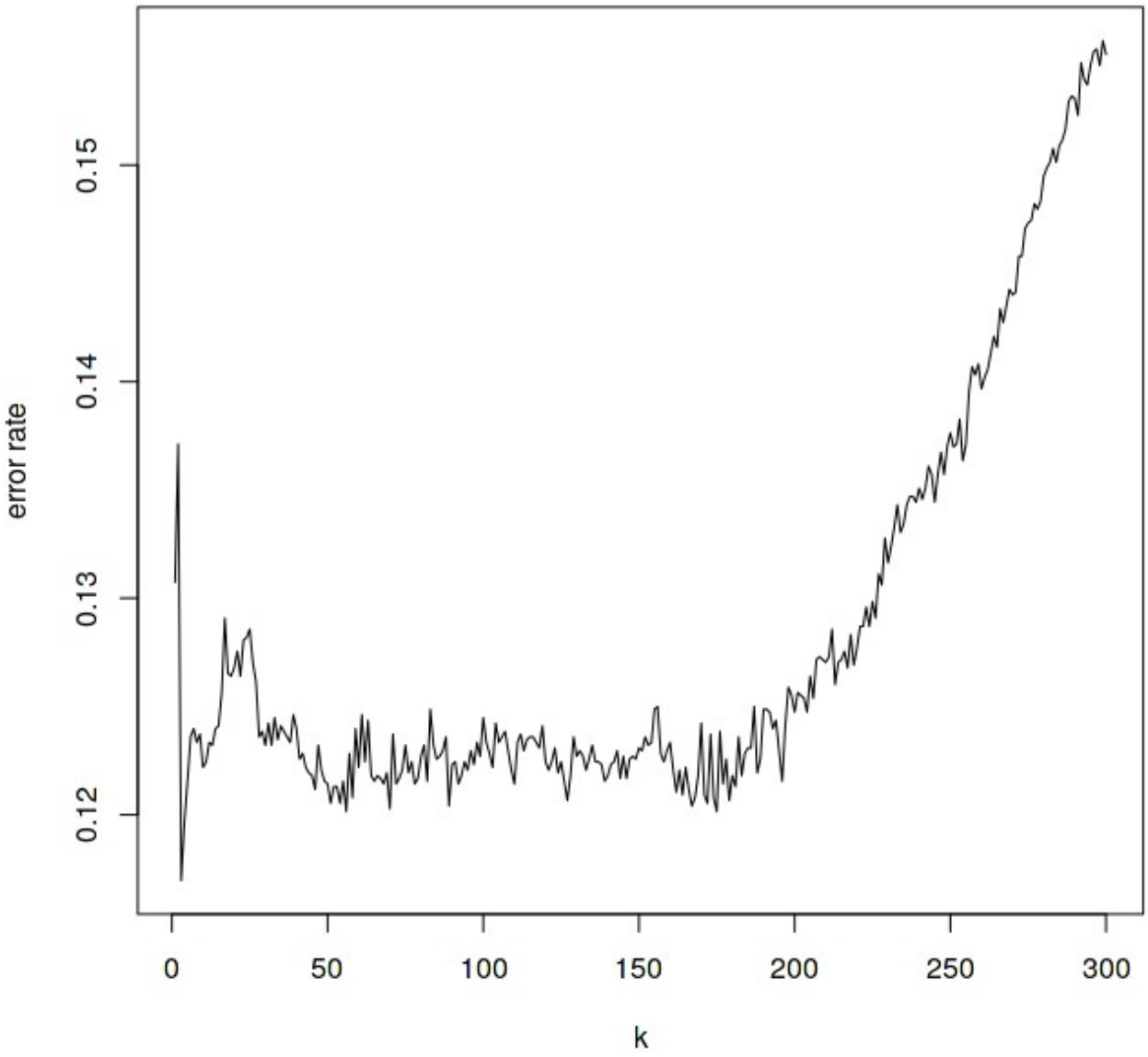
2.6 Naive Bayes algorithm – precision and recall

```
Naive Bayes test error rate:  10.13%
Naive Bayes train error rate: 9.27%

                  true test values
predicted test values  0  1
                     0 37  3
                     1  5 34

Naive Bayes precision: 88.1%
Naive Bayes recall:    92.5%
```

2.7 k-NN -- cross validation error rate



The graph shows, that the best value for k is around **50 (error rate 12%)**. K closing to 343 is equivalent to MFC (all training examples are taken into account without distance weights).