Universitat Politècnica de Catalunya

Facultat d'Informàtica de Barcelona

Małgorzata Wesołowska

Identity card: AMS538415

A study on feature selection based on AICc and its application to microarray data

Final project
in the field of INFORMATICS

The supervisor of the project
Lluis Belanche

June 2009

supervisor's signature

author's signature

Abstract

The following study is a final project written at the Faculty of Informatics (Facultat d'Informàtica de Barcelona) on Tecnical University of Catalonia (Universitat Politècnica de Catalunya). It consists of a document and procedures all collected in a library. All procedures were implemented in R language.

Feature selection is a key issue in machine learning and its importance is beyond doubt. This is especially so in learning tasks that are characterized by a very high dimensionality and a low number of learning examples. The purpose of this project is to investigate the potential of Akaike's information criterion $\mathcal{AIC}$ for feature selection in such learning tasks, exemplified by microarray data, as well as its cooperation with resampling techniques like the bootstrap.

Reducing the dimension of data brings profits from the various points of view. It's obvious, that the more parameters used, the better can be the fit of the model to the given data. However some methods (like Least Squares) needs low dimensionality to ensure executability, that's why we shouldn't be too enthusiastic about selecting many features into the model. It's worth to mention, that when there are many explanatory variables used to predict a dependent variable, it's possible to develop a model with many variables being significant (according to the high values of t-statistics), even if they are independent of the target. However the problem known as a "curse of dimensionality" forces people to keep the number of candidate features small while trying to achieve a large sample size.

$\mathcal{AIC}$ criterion has been introduced by Akaike in 1973 either as a measure of **goodness of fit** and a measure of **predictive accuracy**. By taking into account accuracy and complexity, models with the lowest $\mathcal{AIC}$ indicator are supposed to have very good predictive properties. This is very hard to obtain especially when the data contains a low number of learning examples, which, in addition, we have to divide into **training** and **validating** samples. Following $\mathcal{AIC}$ criterion may be profitable in such cases.

$\mathcal{AIC}$ approach to data analysis and feature selection is based on the measure of **Kullback-Leibler** distance, which describes the information lost when some candidate model or function is used to approximate the real model, which we may not know at all. $\mathcal{AIC}$, as well as Kullback-Leibler distance, has a strong theoretical background in the information theory within a philosophy that no model based on the data is true, moreover, the dimension of the true model is perhaps infinite. This brings us to estimate directed distance between the fitted candidate model and the unknown real mechanism, which generates the data we observed.

The potential of $\mathcal{AIC}$ and its derivatives lies behind the capacity of investigate the best model, which seems to be the closest to the unknown truth. By penalizing the complexity of a model $\mathcal{AIC}$ introduces a trade between the **bias** and the **variance**, what prevents overfitting the data. The penalty for the complexity is proportional to the number of features included in the model.

Whole thesis consists of eleven chapters, which are divided into two basic parts: theoretical and practical. The theoretical part (chapters 1-4) presents the whole formalism of $\mathcal{AIC}$, feature selection, as well as characteristics of terms used in the further part of the work, it contains explanations and examples, which were designed by author. The practical part, i.e. chapters 5-10, is the author's input in whole, it contains the proposed approach to feature selection on microarray data, description and computational complexity of algorithms used. Analysis of problems connected with different statistical methods and final conclusions are

also included in this part. The last chapter contains the timetable and the costs of the project.

Key words

AIC criterion, AICc, feature selection, correlation, relevance

Field of the study (according to Socrates-Erasmus)

11.3 Informatics

ACM classification

62-07. Data analysis

Contents

List of Figures

List of Tables

Introduction

Undertaking to analyze social, biological or other phenomenon a need to pay attention on many characteristics, related to the different attributes of analyzed objects, arises. For example, let's introduce a typical analysis scenario connected with risk management in some financial company, e.g in a bank. Bank analysts are specially interested in a percentage of bank clients which already have, or will probably have, problems with paying off the debt. This factor is formally called the **default rate**. Analyzing the default rate for a certain business sector we observe different characteristics of each client, connected with his personal status, type of performed work and finally - relationship with the bank. On the basis of these characteristics we try to predict the "client's willingness" to pay off the loan - let simplify this to a binary variable: 0 - the client is going to pay the debt, 1 - the client probably will not pay the debt. This leads us to the classification task - how to predict whether a certain client is going or is not going to pay off his duty. If the client's willingness to pay of the debt was judged only by one characteristic, e.g. age, or even many characteristics but treated independently, we might not notice the relations between them. If we don't take into account interactions between certain characteristics, we may reach wrong conclusions. That's why it's important to consider many characteristics/features simultaneously while making decisions. This finally leads us to the issue of multivariate data analysis.

However as the data dimension becomes larger there arise some difficulties with the description and analysis of the data. We encounter problems, which didn't appear in the data set where the number of features is small. For example when the number of features exceeds the number of observations we just can't use some methods. Even the appropriate methods/algorithms have problems with converging in a sensible period of time. That's why many people still search for new methods and tools with an aim to make analyses more efficient.

Objectives

In the project we deal with the classification problem. We investigate the influence of many characteristics describing the investigated objects, called **features** and having continuous values, on the dependent variable - our **target**, which is binary - let us say it can obtain 0 or 1. The "mechanism" of classification - a rule how to judge whether a particular observation should be classified as 0 or as 1 - is called a **classifier** and usually consists of a certain mathematical formula involving a set of features - characteristics.

Classification tasks are present almost in every area of people's activity. In the medical researches the features would be the patient's examination results (e.g. the results of blood count) and the target would be the indicator of some disease. Every e-mail antispam filter implements a mechanism which decides whether to classify a new e-mail as a valuable message or an unwanted message.

The key issue of the project would be to select **few most important features** which influence on the target. We want to do it by a certain method - using the $\mathcal{AIC}$ criterion. The importance of feature selection in machine learning is beyond doubt - feature set reduction is especially desirable in the learning tasks characterized by a very high dimensionality and a low number of learning examples, like **microarray data**. Using the microarray data we have to face the problem of data partition in order to provide reasonable calculations. In the typical machine learning issue data division is performed in order to train models, select the superior one and finally perform the model's assessment. Dealing with microarray data we search for other methods allowing to "save the observations" - a traditional machine learning approach to develop a classifier commands data partition, in a sense of dividing the observation set, into three sets:

- training,
- validation,
- testing,

which means that we have to make three data sets out of one. The purpose for such division will be explained in detail in further chapters. Since we have a low number of observations in a whole data set, each separated subset would have still fewer observation. To little data sample may not provide enough information to build a sensible classifier - this will be explained in the further chapters. That is why we decided to follow other approach, having its background in information theory and called the **likelihood approach**. This approach will be described in the further part of the document, but we should mention that the main profit from follwing the likelihood methodology is that we need a division only into two subsets. That is why we will try to investigate the potential of $\mathcal{AIC}$, an accuracy measure developed on the basis of the likelihood theory, for feature selection in our microarray data as well as its cooperation with some resampling techniques like the bootstrap.

Generally the project can be described as a merger of four issues:

1. Feature selection,
2. Akaike's information criterion,
3. Microarray data,
4. Bootstrap resampling techniques.

In order to join the above we constructed **the feature selection path** which consists of several steps (fig 1):

1. **Filter selection** - at this early stage we pay attention on the relevance of each feature to the target. We compute some relevance measure individually for each existing feature and filter out the worse features in an accordance to the measure. This step assumes reduction of the dimension of the data from thousands to hundreds. The methodology used: ANOVA and Levene's test.
2. **Experimental selection** - since the dimension of the data is still high, we are not convinced to use the wrapper modeling. We will pay attention on correlation between features and so called *predictive accuracy improvement* which results from adding a second feature to a single feature model. The experimental methods are based on a conjecture, that if each two features from some feature set have good predictive

accuracy, the whole feature set might have good predictive accuracy too. This step should reduce the dimension of the data from hundreds to tens. It consists of three algorithms:

(a) Removing high correlations with respect to $\mathcal{AIC}$,

(b) Filtering the most helpful features according to $\mathcal{AIC}$,

(c) Selecting proposed subsets of features.

Summarizing, after processing these algorithms we receive such subsets of features, in which

- each feature is helpful to one another in a sense of $\mathcal{AIC}$ *improvement*,
- none of the features share some predetermined α or higher percentage of information with one another. The α level has to be taken somewhat arbitrarily.

3. **Wrapper selection** in cooperation with bootstrap resampling technique - we apply a wrapper and develop candidate models each basing on a different subset of features achieved in the previous step. The wrapper we are working with is the **glm model** with a target being binary. Since the proportion of the number of observations to the number of features is low we apply $\mathcal{AIC}$c criterion to each learning task. For choosing the best model we also use a **misclassification error** as an average of **0.632-boot** estimators performed on the bootstrap samples. We explore two experimental paths:

   (a) external bootstrap - when a model is developed on a whole data set and bootstrap is applied to calculate the $\mathcal{AIC}$c index of the model

   (b) internal bootstrap - when a model is developed inside each bootstrap sample and the best model shows up as a consequence of bagging.

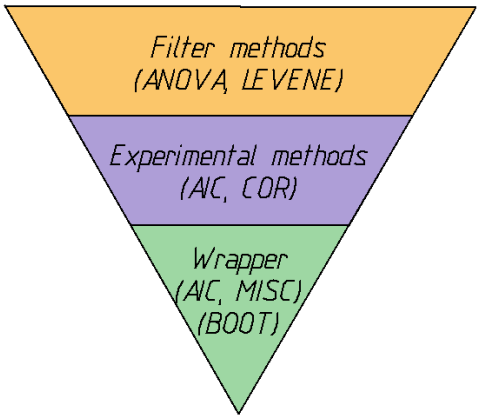


Figure 1: Feature selection process path

Constructing the selection algorithms we made some assumptions:

- the simple methodology used - basic statistics, $\mathcal{AIC}$, *glm* models, paying attention to correlation and relevance in the data set,
- using R libraries with many defined procedures,
- simple and quite intuitive algorithms,

- clear division into functional steps,
- providing methods which give good results at all,
- providing methods able to execute in a sensible period of time.

Author believes that the selection process developed in the project can be useful for feature selection in practice. Sociologists, doctors and representants of medical sciences can analyze data by themselves, but it is not a simple thing. Due to the simple methodology used in the project the feature selection path can be useful for broad practitioners group, representing variety of scientific societies.

Chapter 1

Data and a model

1.1 In search of dependencies

In a typical data analysis scenario even if our goal is to investigate certain characteristic, which is a random variable Y from the probability theory point of view, we usually don't limit ourselves just to collect Y items, but we collect some other additional informations. These informations - other random variables - may be significant in the further data analysis. Collecting additional information is motivated by the fact, that the investigated characteristic, even though it's random, significantly depends on other existing variables. Deep understanding of the shape of this dependence may bring profits in many tasks, e.g. to predict further values of interesting variable.

Let the variable we are interested in be Y and additional variables be $X_1, X_2, \ldots, X_p$ - let them form a vector $\underline{X} = [X_1, X_2, \ldots, X_p]$. Since Y is a random variable it comes from some unknown distribution $\mathcal{R}$ with a vector of parameters $\underline{\theta}$

$$Y \sim \mathcal{R}(\underline{\theta}).$$

If we knew the distribution function $\mathcal{R}(\underline{\theta})$ we could generate random values for Y, since the distribution function of a random variable may be considered as a **generation rule**. However we are interested in a rule f, that links up Y with $\underline{X}$

$$Y = f(\underline{X}). \tag{1.1}$$

Let us move to the real data analysis experiment. Collected values of interesting characteristic (the random variable Y) and values of other variables make up a whole experimental space with a size equal to the number of experiments (observations, tasks) and a dimension equal to the number of variables. Let us consider an example with the size n and the dimension $p+1$ (1.2).

$$\begin{pmatrix} y_1 & x_{1,1} & x_{1,2} & \ldots & x_{1,p} \\ y_2 & x_{2,1} & x_{2,2} & \ldots & x_{2,p} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ y_n & x_{n,1} & x_{n,2} & \ldots & x_{n,p} \end{pmatrix}. \tag{1.2}$$

Each column $[x_{1,j}, x_{2,j}, \ldots, x_{n,j}]^T$, $j = 1, 2, \ldots, p$, make up a **realization** of a random sample $[X_{1,j}, X_{2,j}, \ldots, X_{n,j}]^T$, where each $X_{i,j}$, $i = 1, 2, \ldots, n$ has the same distribution. The column $[y_1, y_2, \ldots, y_n]^T$ constitutes a realization of Y. Since the distribution functions of random

variables $Y, X_1, X_2, \ldots, X_n$ are unknown, instead of (1.1) we search for the function g which fulfill (1.3),

$$Y_i = g(x_{i,1}, x_{i,2}, \ldots, x_{i,p}, \varepsilon_i), \tag{1.3}$$

among all i observations, $i = 1, 2, \ldots, n$, where ε_i is a random factor. We can write a mathematical model envolving random variable Y, function g and the data $\mathbf{x}$

$$Y = g(\mathbf{x}) + \varepsilon, \tag{1.4}$$

where $\mathbf{x}$ denotes a realization of a vector of random variables $\underline{X}$ and ε is connected with randomness. Therefore we search the generation rule for a random variable Y based on the real data and including some randomness. The shape of the link function g relates to the type of dependencies in data, hence g has its set of parameters $\underline{\theta}$.

Notice 1.1.1

> *We have to notice that we never reach the true g and always deal with estimators of g having fixed form. For example we may set the function g to be linear or nonlinear with a specific shape. We can estimate parameters of function g following different methods, thus obtaining little bit different results.*

Now we will briefly introduce the idea of maximum likelihood estimators, linear and logistic regression - as we will use this approach in further considerations.

1.2 Data modeling - the maximum likelihood approach

1.2.1 Multiple linear regression

Let us consider a multiple regression model with a dependent random variable Y as a function of p independent variables x_1, x_2, ..., x_p. Let us assume that the impact of each explanatory variable is linear and independent of other variables. For each observation i, $i = 1, 2, \ldots, n$, we treat y_i as a value of a random variable Y_i and obtain n equations

$$Y_i = \beta_0 + \beta_1 x_{i,1} + \beta_2 x_{i,2} + \ldots + \beta_p x_{i,p} + \varepsilon_i, \quad i = 1, 2, \ldots, n, \tag{1.5}$$

where ε_i, $i = 1, 2, \ldots, n$ are random errors (called residuals) all having the same distribution, whereas $\underline{\beta} = [\beta_0, \beta_1, \ldots, \beta_p]^T$ are unknown parameters generating the approximation function g. Let us consider a matrix $\mathbf{X}_{n\times(p+1)}$, where a row i contains a vector $[x_{i,0}, x_{i,1}, x_{i,2}, \ldots, x_{i,p}]^T$ of values for explanatory variables of Y_i enlarged by an additional coordinate $x_{i,0} = 1$

$$\mathbf{X} = (x_{i,j})_{i=1,\ldots,n,\ j=0,\ldots,p} = \begin{pmatrix} 1 & x_{1,1} & \ldots & x_{1,p} \\ 1 & x_{2,1} & \ldots & x_{2,p} \\ \vdots & \vdots & \ddots & \vdots \\ 1 & x_{n,1} & \ldots & x_{n,p} \end{pmatrix} \tag{1.6}$$

If we consider $\mathbf{Y} = [Y_1, Y_2, \ldots, Y_n]^T$ as a vector of explained random variables and $\underline{\varepsilon} = [\varepsilon_1, \varepsilon_1, \ldots, \varepsilon_n]^T$ as a vector of random errors we may present n equations from (1.5) in a matrix form

$$\mathbf{Y} = \mathbf{X}\underline{\beta} + \underline{\varepsilon}. \tag{1.7}$$

Using (1.7) we may present residuals $\underline{\varepsilon}$ as

$$\underline{\varepsilon} = \mathbf{Y} - E(\mathbf{Y}) = \mathbf{Y} - \mathbf{X}\underline{\beta}. \tag{1.8}$$

The residuals ε_i, $i = 1, 2, \ldots, n$, are assumed to be independent and normally distributed with a mean equal to 0 and an unknown constant variance σ^2, hence each residuum has the distribution $f(\varepsilon_i; \underline{\theta})$, where $\underline{\theta} = \left[\underline{\beta}, \sigma\right]^T$. According to the maximum likelihood methodology we treat each ε_i as a realization of a random variable $\tilde{\varepsilon}_i$ which is normally distributed with a mean equal to 0 and a variance equal to σ^2. Whole family $\tilde{\varepsilon}_1, \tilde{\varepsilon}_2, \ldots, \tilde{\varepsilon}_n$ consists of independent random variables and therefore they have a joint distribution function

$$f(\varepsilon_1, \varepsilon_2, \ldots, \varepsilon_n; \underline{\theta}) = \prod_{i=1}^{n} f(\varepsilon_i; \underline{\theta}), \tag{1.9}$$

$$= \frac{1}{\sqrt{2\pi} \cdot \sigma^n} exp\left(-\frac{1}{2}\sum_{i=1}^{n}\left(\frac{\varepsilon_i}{\sigma}\right)^2\right). \tag{1.10}$$

Having $\varepsilon_1, \varepsilon_1, \ldots, \varepsilon_n$ we may treat the distribution function $f(\varepsilon_1, \varepsilon_2, \ldots, \varepsilon_n; \underline{\theta})$ as a function of unknown $\underline{\theta}$ - we call it the **likelihood** $\mathcal{L}(\underline{\theta})$. Since we obtained the real values of $\varepsilon_1, \varepsilon_1, \ldots, \varepsilon_n$ (as a result of the experiment), each $\varepsilon_i = y_i - (\beta_0 + \beta_1 x_{i,1} + \beta_2 x_{i,2} + \ldots + \beta_p x_{i,p})$, the true value of $\underline{\theta}$ ought to be propitious and therefore the likelihood should have the highest possible value. Following this methodology we try to maximize $\mathcal{L}(\underline{\theta})$ over the unknown $\underline{\theta}$. A vector $\hat{\underline{\theta}}$ which maximizes $\mathcal{L}(\underline{\theta})$ is called the **maximum likelihood estimator** of the unknown $\underline{\theta}$. The approximation function g is set by $\underline{\theta}$, thus the estimator $\hat{g}$ is a function of $\hat{\underline{\theta}}$.

1.2.2 Multiple logistic regression

Before we discuss the logistic regression details, let us follow an example.

Example: We will analyze the data "Earthquake" (source: Shumway (1988)).

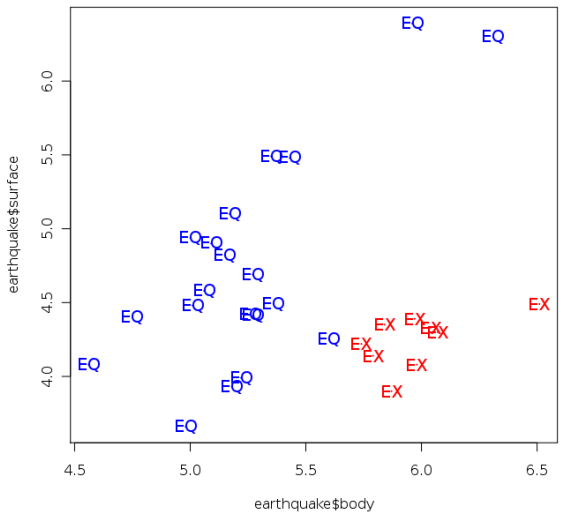


Figure 1.1: The dispersion plot of the earthquake data

The variables in the data set are

- *popn* - nuclear explosion vs. earthquake,

and two seismological features namely

- *body* - body wave magnitude,
- *surface* - surface wave magnitude.

Variable *popn*, as it is binary, plays the role of a target. We have 29 observations: 20 earthquakes and 9 explosions. We observe the values of the seismological features and their relations with the target. Let us have a look on the dispersion of the data (fig. 1.1). We assume value 0 and the blue colour for an earthquake and value 1 and the red colour for an explosion. We want to find an equation describing the dependence between *popn* and the seismological features - *body* and *surface*

$$popn \sim body + surface.$$

As the dependent variable is binary, with the $\{0, 1\}$ value set, the linear regression model may provide values lying out of the target's range - higher than 1 or lower than 0, which does not make sense at all. Let us try do develop such a linear model according to the formula $popn \sim body$, to obtain simple 2D graphical presentation (fig. 1.2).

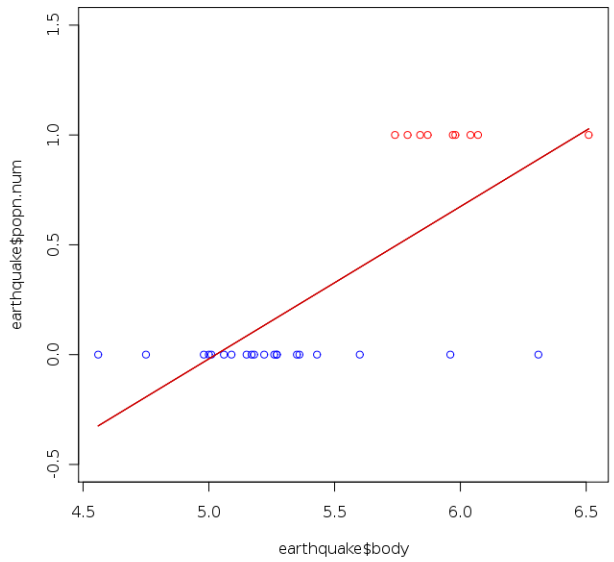


Figure 1.2: The type of the explosion and the body wave magnitude with a linear trend

---

```
Call:
lm(formula = "popn.num~body", data = earthquake)

Residuals:
     Min       1Q   Median       3Q      Max
-0.88899 -0.16783 -0.02768  0.29823  0.50626
```

```
Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)  -3.4865     0.7245  -4.812 5.05e-05 ***
body          0.6934     0.1318   5.260 1.51e-05 ***

Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.337 on 27 degrees of freedom
Multiple R-squared: 0.5061,     Adjusted R-squared: 0.4878
F-statistic: 27.67 on 1 and 27 DF,  p-value: 1.515e-05
```

The linear regression line has been fitted with the variable being significant, but a quite low R^2 indicating the goodness of fit. We see that the fitted linear regression provides *popn* values lower than 0 and greater that 1 when we ask for a predicted *popn* value for some *body* greater that 6.5. To get rid of this inconvenience and obtain more sensible results we use the **logistic regression** approach. We try to fit a curve, with a shape like on a figure 1.3, which correspond to a function obtaining values from the range $[0, 1]$. The value of that function is the **probability** that the target obtain a value equal to 1.

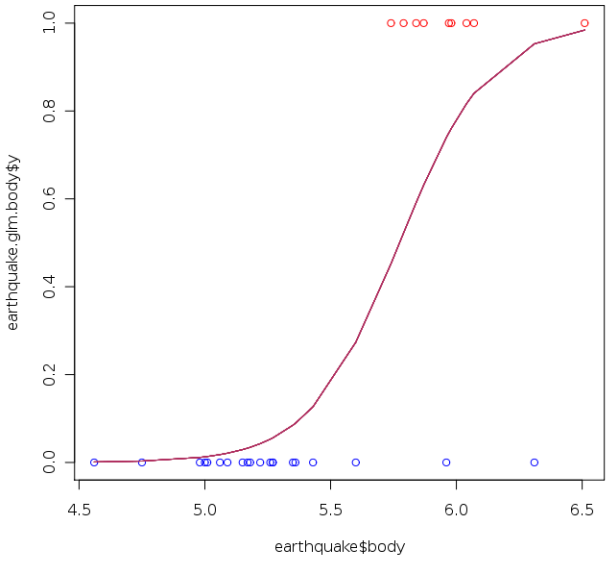


Figure 1.3: The type of the explosion and the body wave magnitude with a logistic trend

□

Let us assume the data in the form (1.2) where $\mathbf{Y}$ is a vector $[Y_1, Y_2, \ldots, Y_n]$ and each random variable Y_i can obtain either 0 or 1. We have

$$E(Y_i) = P(Y_i = 1) \stackrel{df}{=} \pi_i.$$

Instead of the linear dependency we demand

$$logit(E(\mathbf{Y})) = \mathbf{X}\underline{\beta}, \tag{1.11}$$

what means

$$logit(E(Y_i)) = \beta_0 + \beta_1 x_{i,1} + \beta_2 x_{i,2} + \ldots + \beta_p x_{i,p}, \quad i = 1, 2, \ldots, n, \tag{1.12}$$

or

$$logit(E(Y_i)) = \underline{\beta}^T \mathbf{x}_i \tag{1.13}$$

where $\mathbf{x}_i = [x_{i,1}, x_{i,2}, \ldots, x_{i,p}]$ and

$$logit(E(Y_i)) = log\left(\frac{\pi_i}{1-\pi_i}\right).$$

Unraveling the equation (1.12) we obtain

$$\pi_i = \frac{exp(\underline{\beta}^T \mathbf{x}_i)}{1 + exp(\underline{\beta}^T \mathbf{x}_i)}, \tag{1.14}$$

$$1 - \pi_i = \frac{1}{1 + exp(\underline{\beta}^T \mathbf{x}_i)}. \tag{1.15}$$

Like in the case of the linear regression, $y_1, y_2, \ldots, y_n$ are assumed to be independent and binomially distributed each with a distribution

$$\pi_i^{y_i}(1-\pi_i)^{1-y_i} \tag{1.16}$$

Following the maximum likelihood methodology we treat each y_i as a realization of a random variable Y_i, thus whole family $Y_1, Y_2, \ldots, Y_n$ consists of independent random variables and therefore they have a joint distribution function which becomes the likelihood

$$\mathcal{L}(\underline{\beta}) = \prod_{i=1}^{n} \pi_i^{y_i}(1-\pi_i)^{1-y_i} \tag{1.17}$$

Of course we always work on estimators instead of the real values. We received $\hat{\underline{\beta}}$ as the **maximum likelihood estimator** of $\underline{\beta}$ what brings $\hat{\pi}_i$ as an estimator of π_i, $i = 1, 2, \ldots, n$.

As a measure of goodness of fit in the case of likelihood approach we assume **model deviance** *dev*. For a sake of conviniency let us say that

- ω stands for a fixed logistic regression model,
- ω^* stands for the so called **sated model** where we have no limitation for the shape of the dependence.

It means that for n observations we have

$$\omega : logit(\pi_i) = \beta^T \mathbf{x}_i, i = 1, 2, \ldots, n, \tag{1.18}$$

$$\omega^* : \Pi_1, \Pi_2, \ldots, \Pi_n. \tag{1.19}$$

Since $\hat{\underline{\beta}}$ is the **maximum likelihood estimator** of $\underline{\beta}$ we received $\hat{\Pi}_1, \hat{\Pi}_2, \ldots, \hat{\Pi}_n$ as an estimator of the joint probability of $y_i = 1$, $i = 1, 2, \ldots, n$. For ω each $\hat{\Pi}_i$ fulfills the logit equation (1.18) while for ω^*

$$\hat{\Pi}_i = y_i,$$

since it maximizes the likelihood function (1.17).

Definition 1.2.1 (*Model deviance*)

For a logistic regression model ω model deviance dev is assumed as

$$dev_\omega = 2ln\frac{\mathcal{L}_{\omega^*}}{\mathcal{L}_\omega}. \tag{1.20}$$

Since $\mathcal{L}_{\omega^*} = 1$ we can evaluate (1.20) as

$$dev_\omega = 2(ln\mathcal{L}_{\omega^*} - ln\mathcal{L}_\omega) = -2ln\mathcal{L}_\omega \tag{1.21}$$

$$= -2ln\left(\prod_{i=1}^{n}\hat{\Pi}_i^{y_i}(1-\hat{\Pi}_i)^{1-y_i}\right) \tag{1.22}$$

$$= -2\sum_{i=1}^{n}\left(y_i ln\hat{\Pi}_i + (1-y_i)ln(1-\hat{\Pi}_i)\right). \tag{1.23}$$

Notice 1.2.1

The better the model ω approximates the logit dependence in data, what means $\omega \approx \omega^*$, the lower is the model deviance dev_ω.

Example: Now we can estimate the logistic regression model for the earthquake data.

```
Call:
glm(formula = popn ~ body, family = binomial, data = earthquake)

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-2.4719  -0.3389  -0.1899   0.5900   1.2597

Coefficients:
            Estimate Std. Error z value Pr(>|z|)
(Intercept)  -32.397     11.602  -2.792  0.00523 **
body           5.611      2.019   2.779  0.00546 **
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 35.924  on 28  degrees of freedom
Residual deviance: 17.624  on 27  degrees of freedom
AIC: 21.624
```

As a measure of the goodness of fit we will use the percentage of the deviation explained by the developed model (R^2_{dev})

$$R^2_{dev} = 1 - \frac{dev_\omega}{dev_{\omega_0}}, \tag{1.24}$$

where ω stands for the developed model and ω_0 stands for the model having only the intercept. In the example we obtain $R^2 = 0,51$. □

In the project we explore $\mathcal{AIC}$ criterion in logistic regression models what has a strong connection with the model deviance. The symbols and indicators introduced in this chapter will be useful in the next part of the document.

1.3 Model diagnostic

1.3.1 Strength and significance

Formally, every dependence in data we may characterise using two properties

- the strength (the power) of the dependency,
- the significance (the truthfulness) of the dependency.

The strength of dependence is more intuitive to understand. It describes the capacity of predicting one variable from another. If, for example, in our imaginary data sample every Gothic cathedral is higher than every Roman cathedral, we may say, that the relation between the height and the cathedral type is strong. Of course the conclusion is valid at least in our data sample.

The significance concern the problem of how much the result we obtained on the given sample is representative in an accordance to the whole population. It measures the probability that exactly the same relation would be found if we repeated the experiment on the other data (but from the same population). The significance is expressed by *p-value*.

1.3.2 Validating the model

According to the machine learning approach the model is being trained on the part of the data called **training set**. Training the model may result in overfitting the training data, which provide the model excellently fitted to the training set, but presenting poor predictability. One way to avoid this unwanted situation is to execute forecast on the other data set called **validation set**, independent from the training set, and evaluate the measures (i.e. predictive accuracy or misclassification error in the case of a classifier). Validation set must have the same structure as the training set - we must have some known points (target's values) y'_i,$i = 1, 2, \ldots, n'$. So as to obtain the reliable measures the validation set must be independent from the training set. If as a validation sample we use the training set, the results we receive will be obviously biased (underestimated). The model construction process consists in fitting the training data, therefore to judge the model accuracy we must take new observations.

Having the training set we can develop k different models $\hat{g}_j$, $j = 1, 2, \ldots, k$. Then if we are to choose the best one we validate them and select a model presenting the best predictive accuracy **on the validation sample**. More delicate question is how to definitely judge the chosen model? Since the accuracy measure obtained on the validation sample has lead to the election of this model, we need to have another independent data sample. This final assessment should be done on the so called **testing set**. This leads to the division of the original data set into three parts (fig. 1.4). The proportions between the samples should be taken arbitrarily, however a good practise is to keep the size of the training set large. Of course there's no point to separate the validation sample if we have developed just one model on the training data and we don't intend to compare it with any other one - the division into training and testing data set in this particulary case is sufficient.

Unfortunately presented behaviour demands large data sets. To save observations people sometimes resign from the testing sample or even the validation sample. In such cases we

Figure 1.4: Data set division

must appeal to the resampling techniques introducing the smallest possible bias. That is why model assessment based on the division into training, validation and testing data might be unsatisfactory.

But why can't we just assume the measure obtained for the best model on the validation set as the model's final measure? What is (or can be) "wrong" with that measure? The answer is that the final assessment should be done on the **testing set**, so as to prevent "overfitting the validation set". Since we use the specific validation set in order to select a model out of many proposed models, we may "overfit" the model selection. We can't exclude the possibility that if we had selected a different validation set we might have chosen other model as the best model.

Notice 1.3.1

From the general point of wiev,

- *the reason for using the validation sample is to prevent overfitting the training sample when* ***training a single model****,*
- *the reason for using the testing sample is to prevent overfitting the validation sample when* ***choosing the best model*** *from the models previously trained.*

1.3.3 Quality criterion extension

Another way to avoid overfitting the data is to include in a **quality criterion**, described for example by the model's deviation, an error resulted in fitting the training set directly. Imagine we have a training sample $(\mathbf{x}_i, y_i), i = 1, 2, \ldots, n$ and a model $\hat{g}$ that estimates the real dependency f in the data. Let us introduce some **loss function** L which describes the loss resulting in approximating y_i by $\hat{g}(\mathbf{x}_i)$, $i = 1, 2, \ldots, n$. The quality criterion of the model $\hat{g}$ can be

$$\frac{1}{n}\sum_{i=1}^{n} L(y_i, \hat{g}(\mathbf{x}_i)). \tag{1.25}$$

The value (1.25) seems to be a quite optimistic measure, since it takes into account only the data from the set on which the model has been trained. That's why we need to include a correction resulting from the fit to the data. Let us define a measure of the **approximation optimism** (O) as

$$O = \frac{1}{n}\sum_{i=1}^{n} E_{y_i'} L(y_i', \hat{g}(\mathbf{x}_i)) - \frac{1}{n}\sum_{i=1}^{n} E_{y_i} L(y_i, \hat{g}(\mathbf{x}_i)), \tag{1.26}$$

where y_i', $i = 1, 2, \ldots, n$ denote new values for the dependent variable, independent from y_i, corresponding to the same elements of the training sample $\mathbf{x}_i$. E_{y_i} and $E_{y_i'}$ are expectations towards the random variables corresponding to y_i and y_i'. Model $\hat{g}$ is fitted to the training

sample, hence the optimism (1.26) usually obtains positive values. It has to be stressed that, since $\mathbf{x}_i$, $i = 1, 2, \ldots, n$ have been fixed, the left side of (1.26) is not a measure of risk but an expected loss on the condition of $\mathbf{x}$. It has been proved that when the loss function is quadratic or binomial (for the dependent variable being binary) the optimism holds

$$O = \frac{2}{n} \sum_{i=1}^{n} Cov(\hat{y}_i, y_i), \tag{1.27}$$

where $\hat{y}_i = \hat{g}(\mathbf{x}_i)$. Therefore the better model fits the training data, the higher is its approximation optimism. If the real model has a form $Y = g(\mathbf{x}) + \varepsilon$, where g is described by p basis functions (p variables) and ε is a random error with a mean equal to 0 and a variance σ^2, then

$$\sum_{i=1}^{n} Cov(\hat{y}_i, y_i) = p\sigma^2. \tag{1.28}$$

Finally instead of minimizing only (1.25) it's more reasonable to take into account the correction (1.27)

$$\frac{1}{n} \sum_{i=1}^{n} L(y_i, \hat{g}(\mathbf{x}_i)) + O = \frac{1}{n} \sum_{i=1}^{n} L(y_i, \hat{g}(\mathbf{x}_i)) + \frac{2}{n} p\sigma^2. \tag{1.29}$$

The criterion (1.29) is known as the **Mallow's indicator** C_p. The penalty for the complexity of the model is hidden in the optimism Op and it's proportional to the number of variables included in the model $\hat{g}$.

Notice 1.3.2

> *In a very similar way we can derive Akaike's information criterion $\mathcal{AIC}$, since we use the likelihood methodology to estimate $\hat{g}$ and the goodness of fit is expressed by the model deviance. We will do it in the further chapter.*

Chapter 2

Feature Selection

2.1 Motivation for selecting features

As we discussed in the previous chapter, the issue of approximating the information held in the data reduces to searching the best approximating model g which holds (1.3). The function f (1.1) is considered to be a true "information generator" and function g - an equivalent of f based on the data. We assume that function g depends on some $\underline{\theta}$ parameters. We can develop many approximation function, each parameterized by some $\underline{\theta}$ from the possible parameter space $\underline{\Theta}$. Since we have got features to describe their influence on the dependent random variable Y (the target), we can think of selecting the best approximating model g in terms of selecting **the best feature set** to describe the target. According to this way of thinking each parameter $\underline{\theta}$ denotes some candidate feature set. However feature selection is not only an optimal choice of one feature set in order to describe the target in the best possible way. We should know, that modeling task often faces the problems of complexity, executability and significance.

Considering the above, feature selection shall be processed in the context of the following aspects:

The complexity
The general objective of feature selection is to reduce the complexity of the task. On the one hand it seems intuitive that the more parameters used, the better can be the fit of the model to the given data. But on the other hand having many variables causes another problem, known as as a "curse of dimensionality", which stands that addition of extra variables (dimensions) causes an exponential increase of data and operations necessary to execute the approximation.

Fitting the data
The most obvious objective is to select features with good descriptive properties, in order to describe the target in the best possible way. That is why we should put attention on the models' goodness of fit. This aspect doesn't demand any explanation.

Redundancy reduction
Often a smaller feature set contains almost the same information as the larger, initial data set. It's obvious that when two variables are highly correlated they introduce redundancy when taken together to a model. The best solution is to select features which share the smallest possible information.

The significance of the results
The fitted model based on a certain feature set besides sufficient goodness of fit and accuracy should provide the parameters being significant. It's worth to mention, that when there are many explanatory variables used to predict a target, it's possible to develop a model with many variables being significant (according to the high values of t-statistics), even if they are independent of the target. This problem is known as the Freedman's paradox [6]. That's why we need a reasonable trade in the context of the optimal size of the feature set.

The sensibility of the results
What to do, if a particular model contains features which together doesn't make sense from the scientific point of view? The decision whether to take or not to take such model into consideration is arbitrary. Many people recommend the inclusion of all models that seem to have **reasonable justification** rather than paying attention only on the data analysis.

The executability according to the estimation methodology used
Some methods (like the Least Squares or the Maximum Likelihood) needs low dimensionality to ensure executability. We can't have less equations than parameters to estimate! Very often, even if we select the number of features less than the number of observation, the algorithms have problems with converging. This situation occurs mainly when we deal with the low number of learning examples.

Time consumption
We have to keep in mind whole data analysis process and focus on the time the whole process may take. It's clear that the more variables used, the much time is needed to execute the modeling process.

Concerning the foregoing we should use different approaches to perform different feature selection tasks. This leads us to the next section.

2.2 General approaches

2.2.1 Wrapper modeling

In the wrapper methodology of feature selection we use the **classifier's accuracy** as the performance measure. We build examples of a certain type of model (a regression model, a neural network, a tree, etc.) and select the best model according to its predictive accuracy. We treat features used by this classifier as the **optimal features**.

Wrapper modeling consists of two steps:

1. Feature subset selection - selecting the best features according to the classifier's accuracy. For each proposed subset of features we develop a model and calculate its accuracy. After processing all proposed feature sets we keep the feature set which provides the model with the best accuracy measure.

2. Learning the model - the model is being learned and tested on the data with the best features. Since we consider only one feature set we train the model and test the model on the independent testing data set. The predictive accuracy obtained on the testing data is perceived as the model's final predictive accuracy.

There always exists a problem about how to truly estimate the accuracy of the developed model. We deliberated on this problem in the first chapter. Putting aside the accuracy estimation the wrapper modeling is sensitive for computational complexity, since we develop a model on each proposed feature set, thus we should avoid computational intensive classifiers.

2.2.2 Filter modeling

When the reason of feature selection is more than just considering accuracy, we need other selection methodology. Each wrapper introduces its own bias when estimating the accuracy, that's why wrapper taken to select features determines the type of the model to be finally trained. Moreover, in a large data set processing wrappers may be quite difficult or even impossible due to the wrapper's handicap of handling high dimensional data. To decrease the size of the feature set from thousands to hundreds or tens, in a sensible period of time, we need simple methods selecting a subset of features with an aim to achieve the highest relevance to the target on the basis of the given data. These methods are known as filter methods.

A filter model consists of two steps:

1. Feature subset selection - selecting the best features according to the measures which doesn't involve the classifier (i.e. distance, dependence, deviation or consistency).

2. Learning the model - the same process as in the wrapper modeling path.

Since a filter method gains information in a cheaper way than measuring the accuracy of a classifier, the filter method can select optimal subsets faster. As the filter model handles very large data sets it can be used for to reduce high dimensionality.

Chapter 3

Microarray Data

3.1 Microarray data structure

Since we discussed the feature selection methods let's focus on the data set. If we have almost unlimited possibility to collect the data by repeating the experiments, as for example in a laboratory, we may have strong conviction to obtain statistically satisfying results. To solove many computational problems we just might need more data! But when the collection of observations connects with an increase of the costs, accepting worse results could be less painful than the extension of the data sample. This situation usually appears in medicine researches when to obtain a single observation we must carry out some expensive experiments. In the project we are going to deal with such data called **microarray data** and represented by a matrix $n \times p$, where $n \ll p$ (table 3.1). Microarray data which we are going to analyze

$x_{1,1}$	$x_{1,2}$	$\cdots$	$\cdots$	$\cdots$	$\cdots$	$\cdots$	$\cdots$	$\cdots$	$\cdots$	$\cdots$	$\cdots$	$\cdots$	$x_{1,p}$
$x_{2,1}$	$x_{2,2}$	$\cdots$	$\cdots$	$\cdots$	$\cdots$	$\cdots$	$\cdots$	$\cdots$	$\cdots$	$\cdots$	$\cdots$	$\cdots$	$x_{2,p}$
$\cdots$	$\cdots$	$\cdots$	$\cdots$	$\cdots$	$\cdots$	$\cdots$	$\cdots$	$\cdots$	$\cdots$	$\cdots$	$\cdots$	$\cdots$	$\cdots$
$x_{n,1}$	$x_{n,2}$	$\cdots$	$\cdots$	$\cdots$	$\cdots$	$\cdots$	$\cdots$	$\cdots$	$\cdots$	$\cdots$	$\cdots$	$\cdots$	$x_{n,p}$

Table 3.1: Microarray Data

is a **DNA microarray**. It consists of an arrayed series of thousands of microscopic spots of DNA oligonucleotides, called **features**, each containing picomoles of a specific DNA sequence. This can be a short section of a gene or other DNA element that are used to describe a classification variable, called **target**. The target may indicate for example a serious illness. Microarrays present new statistical problems because the data is very high dimensional with very little replication. Using microarrays we investigate thousands of genes simultaneously, in the project we want to discover the most important genes to describe the target.

3.2 Analysis problems

In the first chapter we said that the dependence in data may be characterized by two factors: **strength** and **significance**. Despite these are two different factors they are not completely independent from themselves. Generally speaking, the dependence in data the strongest, the more significant. Why we can say that? If we assume that in the population between certain characteristics doesn't occur any relation, the same lack of relation is probably going to occur in the statistical sample. We can draw a conclusion that the stronger relation has been discovered in the sample, the less possible is the lack of this relation in the whole population.

Of course we consider some fixed sample size, since one relation may turn out to be significant or nonsignificant according to the sample size. If we deal with a small number of observations, we obtain a small number of possible combinations of different values for particular variables. That's why the propability that a certain combination indicating a strong dependence appear by chance is relatively high.

Example: Imagine we have a sample consisting of two variables each having two possible values and five observations. Let it be five cathedrals, each having its type (Roman or Gothic) and height (high or low). Let's have two Roman and three Gothic cathedral. The probability that we discover a 100% relation just by chance is equal to $\frac{1}{16}$. The probability that both Roman cathedrals are low and every Gothic cathedral is high (or vice versa) is equal to the below

$$P((height = (l,l,h,h,h) \cap type = (r,r,g,g,g)) \cup (height = (h,h,l,l,l) \cap type = (r,r,g,g,g)))$$

$$= \frac{\frac{1}{2} \cdot 1 \cdot 1 \cdot 1 \cdot 1}{\left(\frac{1}{2}\right)^5} = \left(\frac{1}{2}\right)^4 = \frac{1}{16}.$$

The same chance in a 100 sized sample is around zero. □

Notice 3.2.1

> *The less observation in the sample, the more often we obtain a dependence that doesn't exist in the whole population.*

The separate problem is the strength of the dependencies possible to discover in a low-size sample. If the dependence between certain features in not so strong in the general population, there's no other way to detect it but using a numerous sample. Analogously, in the case of strong dependence in the general population, to prove the dependence we can even use a low-size sample. Exemplifying, if we flip a little bit asymmetrical coin (40% heads, 60% tails), ten experiments won't be enough to ensure anyone that the coin is in fact asymmetrical (even if we receive 4 heads and 6 tails). If the coin was more asymmetrical (95% heads, 5% tails), ten experiments could be enough to prove the thesis about the strong asymmetry. Then if the relation is strong it will be significant even in a small data sample.

Taking above into consideration, when the number of observations is low, we have to deal with the problem of relations detectability. Linking the above with necessity of division the data into three samples (training, validation and testing) we receive a tough research problem. The proposed solution would be to save observations by, for example, not separating the validation set and using data resampling techniques - what we are going to discuss in the following chapter

3.3 Perfect separation

In a microarray data it is highly possible that we obtain a situation called the **perfect separation**. Having a single explanatory variable X and a binary target Y, perfect separation in logistic regression model occurs when

$$\begin{aligned} y = 0 \quad & \forall x \leq x_0, \\ y = 1 \quad & \forall x > x_0, \end{aligned} \tag{3.1}$$

or

$$\begin{aligned} y = 1 \quad \forall x \leq x_0, \\ y = 0 \quad \forall x > x_0, \end{aligned} \tag{3.2}$$

where x_0 is some value of x.We have a glm model

$$logit(E(Y)) \quad = \quad \beta_0 + x\beta_1. \tag{3.3}$$

For a sake of simplicity, let us assume that the model has a form below, with some mean μ and dispersion σ

$$logit(E(Y)) \quad = \quad \frac{x - \mu}{\sigma}, \tag{3.4}$$

then as $\sigma \longrightarrow 0$ (the separability becomes ideal) assuming 3.1 we have

$$\begin{aligned} E(Y) \longrightarrow 0 \quad for\ x < \mu, \\ E(Y) \longrightarrow 1 \quad for\ x > \mu, \end{aligned}$$

In terms of model fitting this situation looks very good. However such situations are highly unusual in real life and some people view such a result with a mistrust, as caused by chance rather than by a real variability occurring in data. Let us remind that the probability of obtaining the ideal 100% dependence in data rises by the decrease of observations. We will discuss this issue once again in the Problems and Solutions chapter.

3.4 Feature selection in microarray data

Since we work with high dimensional task and want to select the best features describing a target (Y), we have to very carefully choose the methodology. We would like to obtain a set of features $\hat{f}$, that provides the best possible model $\hat{g}$ for future target forecast.

$$Y = \hat{g}(\hat{f}). \tag{3.5}$$

Function $\hat{g}$ is denoted as an estimator of the real dependence g.

As the dimensionality is high, first step of feature selection can't involve the model. Common methodology is firstly to use filters, secondly wrappers. We should notice that we deal with a small number of observations, so we have to save them as much as possible. This conclusion leads us to the information theory approach introduced in the next chapter.

Chapter 4

AIC theoretical approach

4.1 A brief introduction to $\mathcal{AIC}$

In the first chapter we discussed two ways of evaluating the selected model's accuracy

- data partition into training, validation and testing set,
- quality criterion extension.

First approach, as it bases on the data partition into three parts, demands larger data set. We use training set to train many models, validating set to select one model out of many, finally testing set to evaluate the model's accuracy. In the microarray data, when every observation is "precious", the data partition approach may lower the quality of modeling. As we discussed in the third chapter, the fewer observations in the sample, the strongest must be the dependence to be traced. Concerning the above we come up with the second approach, which doesn't demand a validation sample. The quality criterion approach includes a penalty for overfitting the data, what, we can say, plays the role of the validation set.

4.2 Derivation of $\mathcal{AIC}$

4.2.1 The Kullback-Leibler Distance

In this section we will briefly introduce the idea of $\mathcal{AIC}$. The complete details about the conceptual and mathematical background of the derivation of $\mathcal{AIC}$ are provided in [1].

Let's consider two multivariable models f and g on a p dimensional space, $f, g : \mathbb{R}^p \to \mathbb{R}$, $\mathbf{x} = (x_1, x_2, \ldots, x_p)$.

Definition 4.2.1 (*Kullback-Leibler Distance (Information, Entropy)*)

The Kullback-Leibler (K-L) distance between two models f and g, $f, g : \mathbb{R}^p \to \mathbb{R}$, is defined as an integral

$$I(f, g) = \int f(\mathbf{x}) log\left(\frac{f(\mathbf{x})}{g(\mathbf{x})}\right) \mathrm{d}\mathbf{x}, \tag{4.1}$$

where $\mathbf{x} = (x_1, x_2, \ldots, x_n)$, log denotes a natural logarithm.

For discrete functions f and g, $f = (p_i)_{i=1,2,\ldots,k}$, $g = (\pi_i)_{i=1,2,\ldots,k}$ the K-L distance is expressed

as

$$I(f,g) = \sum_{i=1}^{k} p_i log\left(\frac{p_i}{\pi_i}\right). \tag{4.2}$$

We can think of functions f and g as of distribution functions (continuous or discrete). Then f is an ideal distribution and g is an empirical distribution.

Let's bring back the situation from chapter 1 where f was a true generation function for the unknown random variable Y and g was the generation function for Y based on the given data (1.3). The K-L distance becomes

$$I(f,g) = \int f(\mathbf{x}) log\left(\frac{f(\mathbf{x})}{g(\mathbf{x};\underline{\theta})}\right) \mathrm{d}\mathbf{x}, \tag{4.3}$$

where $\underline{\theta}$ is a vector of parameters for g.

Since the function f is considered to be a true model and function g - an equivalent of f based on the data, which depends on $\underline{\theta}$ parameters, the K-L distance measures the information lost when g is used to approximate f. Since the difference between f and g is expressed by $log\left(\frac{f(\mathbf{x})}{g(\mathbf{x};\underline{\theta})}\right)$ and multiplied by the value of f over the whole domain of f (and g), we can interpret $I(f,g)$ as a measure of the inefficiency of assuming the g distribution when the true distribution is f ([4]).

From the probability theory point of view the K-L distance is an expected value for the logarithm of the ratio of two distributions with respect to the distribution of f. In the discrete case we can treat K-L distance as a kind of average,

$$I(f,g) = E_X\left(log\left(\frac{f(\mathbf{x})}{g(\mathbf{x};\underline{\theta})}\right)\right). \tag{4.4}$$

Notice 4.2.1

We should notice that

$$I(f,g) \neq I(g,f),$$

*as the difference $f - g$ plays different role in $I(f,g)$ and $I(g,f)$. We see that $I(f,g)$ is an **oriented** measure.*

Notice 4.2.2

We can express K-L distance of two models f and g in terms of expectations

$$I(f,g) = E_X\left(log(f(\mathbf{x}))\right) - E_X\left(log(g(\mathbf{x}|\underline{\theta}))\right). \tag{4.5}$$

4.2.2 $\mathcal{AIC}$ criterion

Now let's consider a parameter space $\underline{\Theta}$ that is the set of all possible $\underline{\theta}$. Without doubts there exists such $\tilde{\underline{\theta}}$ for which $I(f,g)$ obtain the smallest possible value. We can say, that f express "the truth" and $\tilde{g} = g(\cdot;\tilde{\underline{\theta}})$ minimizing $I(f,g)$ over all $\underline{\theta} \in \underline{\Theta}$ - the closest point to this "truth" that we can reach including given $\underline{\Theta}$ conditions. That's why our concept to select the best model is based on the minimization of K-L distance.

In data analysis we use estimators instead of true parameters of the model what usually brings some uncertainty. If we base on $\hat{\underline{\theta}}$ rather than $\underline{\theta}$, it causes the change in the model

selection criterion to that of minimizing the estimated K-L distance

$$\hat{I}(f,g) = E_X\left(log\left(\frac{f(\mathbf{x})}{g(\mathbf{x};\underline{\hat{\theta}})}\right)\right). \tag{4.6}$$

We shall link $\underline{\theta}$ with $\underline{\theta}(\mathbf{y})$, where $\mathbf{y}$ is a vector connected with **the given data**, whereas $\mathbf{x}$ is rather theoretical. Both $\mathbf{x}$ and $\mathbf{y}$ have the same probability distribution and come from the same sample space. Over the data we expect the estimator of K-L to have an average value equal to (4.7)

$$\begin{aligned} E_{\underline{\hat{\theta}}}(\hat{I}(f,g)) &= E_{\underline{\hat{\theta}}}\left(E_X\left(log\left(\frac{f(\mathbf{x})}{g(\mathbf{x};\underline{\hat{\theta}})}\right)\right)\right) &(4.7)\\ &= \int f(\mathbf{y})\left[\int f(\mathbf{x}) log\left(\frac{f(\mathbf{x})}{g(\mathbf{x};\underline{\hat{\theta}})}\right)\mathrm{d}\mathbf{x}\right]\mathrm{d}\mathbf{y}. &(4.8)\end{aligned}$$

In terms of K-L distance the problem of finding the best model $g(\cdot;\underline{\hat{\theta}}(\mathbf{y}))$ to approximate the unknown truth f over $\mathbf{y}$ reduces to the minimization of

$$E_{\underline{\hat{\theta}}}(\hat{I}(f,g)),$$

which turns out to be equal to the maximization of

$$E_Y E_X(log(g(\mathbf{x};\underline{\hat{\theta}}(\mathbf{y})))). \tag{4.9}$$

Since we have the data and don't have the imaginary ideal we use $\mathbf{y}$ instead of $\mathbf{x}$ and obtain

$$log(g(\mathbf{y};\underline{\hat{\theta}}(\mathbf{y}))) \equiv log(\mathcal{L}(\underline{\hat{\theta}}(\mathbf{y});\mathbf{y})). \tag{4.10}$$

However (4.10) turns out to be biased and under certain conditions the bias is proportional to the number of estimated parameters k in the approximating model g, $k = dim(\underline{\hat{\theta}})$. An approximately unbiased estimator of (4.9) for large samples is

$$log(\mathcal{L}(\underline{\hat{\theta}});y) - k. \tag{4.11}$$

Then we obtain an estimator of relative expected K-L distance

$$-log(\mathcal{L}(\underline{\hat{\theta}});y) + k. \tag{4.12}$$

This leads us to obtain $\mathcal{AIC}$ information criterion

$$\mathcal{AIC} = -2log(\mathcal{L}(\underline{\hat{\theta}});y) + 2k. \tag{4.13}$$

Notice 4.2.3

> *The formula (4.13) resembles the Mallow's C_p indicator (1.29), since they have similar origins.*

4.3 $\mathcal{AIC}$ overview

4.3.1 $\mathcal{AIC}$ in regression

Let's make an overview of $\mathcal{AIC}$ criterion for a linear and logistic regression. Let's assume a data set consisting of n observations and p variables. For a linear regression and the maximum likelihood methodology we obtain

$$\mathcal{AIC} = nlog\left(\frac{SSE}{n}\right) + 2k, \tag{4.14}$$

since $\frac{SSE}{n}$ is a maximum likelihood estimator of σ^2 and $k = p + 1$ is a number of estimated parameters in the linear regression model.

Proof: Since the second factor in (4.13) is constant we will calculate the first factor.

$$\begin{aligned}
log(\mathcal{L}(\underline{\hat{\theta}}); y) &= log\left[\left(\frac{1}{\sqrt{2\pi\sigma^2}}\right)^n exp\left(-\frac{1}{2\sigma^2}\sum_{i=1}^{n}\varepsilon_i^2\right)\right], \\
&\overset{\left[\sum_{i=1}^{n}\varepsilon_i^2 = SSE\right]}{=} log\left[(2\pi\sigma^2)^{-\frac{1}{2}n} \cdot exp\left(-\frac{1}{2\sigma^2}SSE\right)\right], \\
&= -\frac{1}{2}nlog(2\pi\sigma^2) - \frac{1}{2\sigma^2}SSE, \\
&\overset{\left[\hat{\sigma^2}_{MLE} = \frac{SSE}{n}\right]}{=} -\frac{1}{2}n\left[log(2\pi\sigma^2) + 1\right].
\end{aligned} \tag{4.15}$$

Replacing the first term in (4.13) by (4.15) we obtain

$$\begin{aligned}
\mathcal{AIC} &= n\left[log(2\pi\sigma^2) + 1\right] + 2K, \\
&= nlog\sigma^2 + nlog(2\pi) + n + 2K, \\
&= nlog\sigma^2 + 2K,
\end{aligned} \tag{4.16}$$

as $nlog(2\pi) + n$ is a constant. ■

For a logistic regression, since we use the likelihood methodology and the goodness of fit is expressed by the model deviance, we obtain

$$\mathcal{AIC} = dev + 2k, \tag{4.17}$$

as $dev = -2log(\mathcal{L}(\underline{\hat{\theta}}); y)$, what we discussed in the first chapter.

4.3.2 $\mathcal{AIC}$ variants

It has been proved that $\mathcal{AIC}$ may perform poorly if there are relatively many parameters comparing to the number of observations. We multiply the penalty for complexity term by the correction factor $\left(\frac{n}{n-k-1}\right)$

$$\mathcal{AIC}_c = -2log(\mathcal{L}(\underline{\hat{\theta}}); y) + 2k\left(\frac{n}{n-k-1}\right), \tag{4.18}$$

thus we obtain

$$\mathcal{AIC}_c = \mathcal{AIC} + \frac{2k(k+1)}{n-k-1}. \tag{4.19}$$

We see that when n is large with respect to k the correction factor is approximately 1. $\mathcal{AIC}_c$ came up as a result of many statistical researches and investigations. Many people suggest to use $\mathcal{AIC}_c$ indicator instead of $\mathcal{AIC}$ when a proportion $\frac{n}{k}$ is small, arbitrarily set to less that 40.

There exist some other variants of $\mathcal{AIC}$, like $\mathcal{QAIC}$, but as they were fully described in [1] and we don't use them in practice, they won't be described here.

4.4 Conclusions

In the linear regression a goodness of fit was expressed by SSE, in the logistic regression it's the model deviance. We can say in general that

- the higher the number of features, the better the fit of the model to the data,
- the lower the number of features, the worse the fit of the model to the data,

where the fit can be expressed by SSE or the model deviance.

$\mathcal{AIC}$ formula penalizes the high complexity and at the same time awards the good fit. It searches for the golden mean between complexity and fit. It prefers less complicated models, however if for a more complicated model the the award from a good fit is higher than the penalty for the complexity, the complicated model will be favoured.

Notice 4.4.1

Summarizing, $\mathcal{AIC}$ can be considered as

- *a measure of the* ***goodness of fit***
- *a measure of the* ***information lost***
- *a trade between* ***bias*** *and* ***variance***

Chapter 5

Filter methods

5.1 Filtering the subset of features

In a large data set feature selection process which put attention on the classifier's accuracy may be quite difficult or even impossible to execute. All regression models demand a data set to have at least as many observations as features. If this demand is satisfied we can then select features according to a common methodology of backward or forward selection. However how to decrease the size of the feature set from thousands to hundreds or tens? Besides executability, how to do the selection in a sensible period of time? That's why we should come up with simple methods selecting a subset of features with an aim to achieve the highest relevance to the target on the basis of the given data.

In this early stage of feature selection we should

- reject features that have little chance to be usefull in the further analysis rather than select very good features,
- use methods which are independent of the specific predictors, since the data size is huge and a classifier cannot be directly applied to the data set,
- pay attention on the executability and the computational effort of the selection process.

Taking above into consideration we will base on the measures calculated directly from the data, without any predictor's assistance. This leads us to *filter methods* of feature selection.

Filter methods used in the project would pay attention on the relevance of each feature to the dependent variable. We will compute some relevance measure individually for each existing feature and filter out the worse features in an accordance to the measure. At this stage we leave the problem of correlation and reducing redundancy - this will show up in the next chapter.

5.2 Filtering by testing

5.2.1 Filtering by testing equality of means

Considering a feature selection process, we want to select the subset of features, which describes the dependent variable in the "best possible way" (better is to say "in a very good way"). From the common point of view, we should search variables, which obtain quite different values among different levels of presented factor. Let's assume

$$F(X|Y = k) \stackrel{df}{=} F_X^k. \tag{5.1}$$

Variable presenting statistically the same distributions among different levels of the factor (F_X^k are equal for each k or even some of k) doesn't have good discriminative properties. The best descriptors would be features which have the most different F's F_X^k for all k. Of course we can't say that the best variables, in a sense of different F_X^k, together describe the factor in the best way. We won't know until we make a model using this set of variables and the factor, but at this stage we ignore modeling because of too large number of features being selected in this early stage. Following this way of thinking, we could test an equality of probability distributions of one variable in each value of the factor by comparing two samples (Kolmogorov-Smirnov test). In a single K-S test a distance between the empirical distribution functions of two samples is quantified. The null hypothesis is that samples are drawn from the same distribution, the alternative is that they're not. Such a test doesn't have adaptative properties, therefore we can't expect it to have such a good quality as, for example, tests for normality. It's rather intuitive, that K-S test might have quite good statistical power only when two distribution functions are significantly different in a neighborhood of the means.

We are interested in measuring dependence between some continuous variable and a factor. How we decide to present this interaction it's a matter of approach. We can say we are measuring an influence of a continuous variable over a factor or, quite the reverse, an influence of a factor over a continuous variable. Let's take the second approach when a continuous variable (i.e gene) is taken as a dependent variable and a factor plays the role of an independent variable. Instead of K-S test, testing equality of probability distributions of continuous variable in each value of the factor, we can use analysis of variance (ANOVA). We are not interested in how the dependence between a factor and a continuous variable looks like from the mathematical point of view, moreover, we don't want to express this dependence in any mathematical form. We would just like to know, whether a change of a factor level would influence on the mean of a dependent variable. In the simplest form ANOVA provides a statistical test of whether the means of a dependent variable within several groups of a factor are all equal. In fact ANOVA generalizes Student's two-sample t-test to more than two groups.

Let's assume a microarray data set with n observations consisting of one binary variable, let's call it Y, and p continuous variables representing different genes, let it be a matrix called X.

$$Y = \{0, 1\}\,,\ X = |g_1, g_2, \ldots, g_p| \tag{5.2}$$

We want to check whether each gene g_i has different mean in a different level of a factor. We execute p such tests:

$$H_0 : \mu_0 = \mu_1, \tag{5.3}$$
$$H_1 : \mu_0 \neq \mu_1, \tag{5.4}$$

where μ_0 is a mean of g_i based on those observations which obtain 0 as Y, μ_1 is a mean of g_i, analogously, when $Y = 1$. For the categorical variable Y, having k levels, each test would consist of such hypothesis:

$$H_0 : \mu_1 = \mu_2 = \ldots = \mu_k, \tag{5.5}$$
$$H_1 : \exists_{i,j,i\neq j} \mu_i \neq \mu_j, \tag{5.6}$$

where μ_i is a mean of g_j (continuous variable) when $Y = i$ across all $j = 1, \ldots, p$.

We will need p-value of each test to select the best subset of features.

5.2.2 Filtering by testing equality of variances

ANOVA gives good results in order to indicate whether a feature (variable) has significantly different mean in a different levels of a factor. We suspect, that the more different means, the better discriminative properties tested feature presents. For example, if healthy patients present, in most cases, higher value of some gene than ill patients, it may mean that this gene is somehow responsible for that illness or can be used to distinguish people from a special risk group. It has to be stressed that in a single ANOVA test an influence of a different levels of a factor on a **single** variable is tested. However paying attention only on the means we may loose some important features which can describe a dependent variable together with other variables. That's why it could be profitable to keep features which have significantly different variance in a different levels of a factor.

For testing homogeneity of variances in k groups we can use Levene's test. The test is based on the ANOVA statistics applied to absolute deviations of observations representing corresponding group of mean.

Example: Let's consider the cathedrals data (from R library MASS). The data contains some sample cathedrals each described by three characteristics:

- *style* - the style which the cathedral represents (r - Roman style, g - Gothic style),
- x - the width of the cathedral,
- y - the height of the cathedral.

name	style	x	y	style.num
Durham	r	75	502	0
Canterbury	r	80	522	0
Gloucester	r	68	425	0
Hereford	r	64	344	0
Norwich	r	83	407	0
Peterborough	r	80	451	0
St.Albans	r	70	551	0
Winchester	r	76	530	0
Ely	r	74	547	0
York	g	100	519	1
Bath	g	75	225	1
Bristol	g	52	300	1
Chichester	g	62	418	1
Exeter	g	68	409	1
GloucesterG	g	86	425	1
Lichfield	g	57	370	1
Lincoln	g	82	506	1
NorwichG	g	72	407	1
Ripon	g	88	295	1
Southwark	g	55	273	1
Wells	g	67	415	1
St.Asaph	g	45	182	1
WinchesterG	g	103	530	1
Old.St.Paul	g	103	611	1
Salisbury	g	84	473	1

Table 5.1: Data cathedrals

We are interested in the relation between a cathedral's style and cathedral's parameters. We will consider *style* as a dependent variable, x and y as independent variables. Now let's have a look on each independent variable by drawing its boxplot (fig. 5.1).

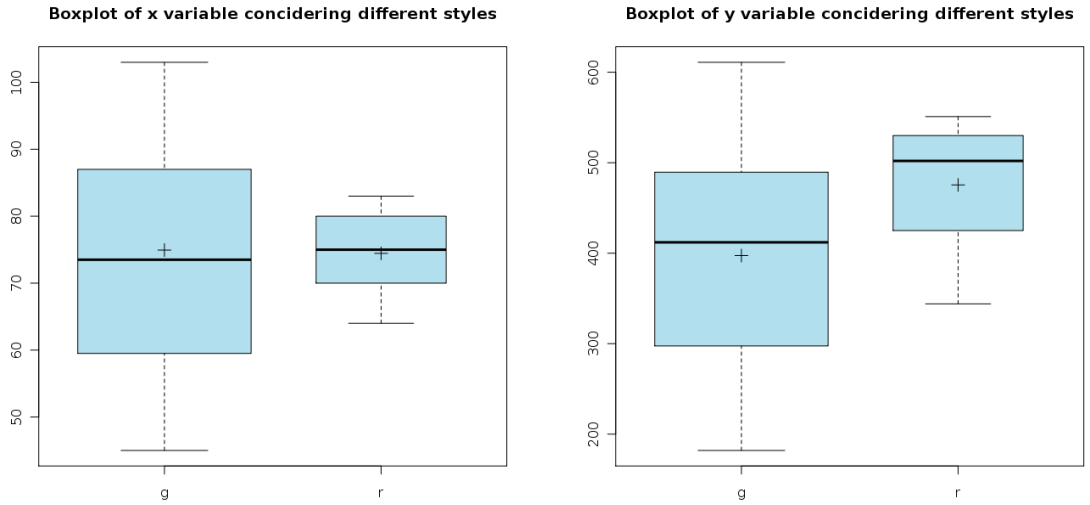


Figure 5.1: Boxplots of x and y in the different levels of *style*.

Let's make an ANOVA test for the equalities of means in each parameter of cathedral (`aov()` procedure in R).

ANOVA for x:

```
Call:
   aov(formula = x ~ style, data = cathedrals)

Terms:
                  style Residuals
Sum of Squares     1.40   5365.16
Deg. of Freedom       1        23

Residual standard error: 15.27311
Estimated effects may be unbalanced

Summary:
            Df Sum Sq Mean Sq F value Pr(>F)
style        1    1.4     1.4   0.006  0.939
Residuals   23 5365.2   233.3
```

ANOVA for y:

```
Call:
   aov(formula = y ~ style, data = cathedrals, projections = T)

Terms:
                     style Residuals
```

```
Sum of Squares   35106.27 251325.97
Deg. of Freedom         1        23

Residual standard error: 104.5333
Estimated effects may be unbalanced

Summary:
           Df Sum Sq Mean Sq F value  Pr(>F)
style       1  35106   35106  3.2127 0.08623
Residuals  23 251326   10927
```

According to ANOVA results the x variable has almost the same mean for both Gothic and Roman style cathedrals while the y variable presents different means with almost 92% confidence. Let's try to adjust logistic regression models in three situations:

1. $style \sim x$
2. $style \sim y$
3. $style \sim x+y$

Ad 1. $style \sim x$:

```
Call:
glm(formula = style ~ x, family = binomial, data = cathedrals)

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-0.9710  -0.9506  -0.9331   1.4246   1.4380

Coefficients:
             Estimate Std. Error z value Pr(>|z|)
(Intercept) -0.403655   2.164775  -0.186    0.852
x           -0.002299   0.028468  -0.081    0.936

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 32.671  on 24  degrees of freedom
Residual deviance: 32.664  on 23  degrees of freedom
AIC: 36.664

Number of Fisher Scoring iterations: 4
```

Ad 2. $style \sim y$:

```
Call:
glm(formula = style ~ y, family = binomial, data = cathedrals)

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-1.5459  -0.8788  -0.5626   1.1067   1.7826

Coefficients:
```

```
             Estimate Std. Error z value Pr(>|z|)
(Intercept) -4.188405   2.306863  -1.816   0.0694 .
y            0.008221   0.005000   1.644   0.1001
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 32.671  on 24  degrees of freedom
Residual deviance: 29.284  on 23  degrees of freedom
AIC: 33.284

Number of Fisher Scoring iterations: 4
```

Ad 3. $style \sim x+y$:

```
R summary output:
Call:
glm(formula = style ~ ., family = binomial, data = cathedrals)

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-1.2564  -0.8473  -0.5200   0.7966   2.0587

Coefficients:
            Estimate Std. Error z value Pr(>|z|)
(Intercept) -1.89309    2.71443  -0.697   0.4855
x           -0.09369    0.05646  -1.659   0.0970 .
y            0.01886    0.00912   2.069   0.0386 *
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 32.671  on 24  degrees of freedom
Residual deviance: 25.679  on 22  degrees of freedom
AIC: 31.679

Number of Fisher Scoring iterations: 5
```

We can see that the AIC parameter for the third model has the lowest value. The estimate parameters are significant considering 0.1 as a significance level. If we fixed the significance level on 0.05, the only significant variable would be y. The first model, using only the x variable to describe *style*, is inappropriate because of the low confidence of estimated parameter for x, whereas the second model can be accepted with a significance level set to about 0.1. Let's test the variances of x and y.

```
        Classical Levene's test based on the absolute deviations from the mean

data:  cathedrals$x
Test Statistic = 9.6912, p-value = 0.004894
```

```
        Classical Levene's test based on the absolute deviations from the mean

data:  cathedrals$y
Test Statistic = 1.604, p-value = 0.218
```

The width of the cathedral vary more for Gothic cathedrals than for Roman cathedrals and this difference is significant (whereas both kinds of cathedrals have on the average the same width). Even if we can't determine the style of a cathedral using only its width, the information about the width can help together with an information about a cathedral's height. □

5.2.3 Choosing the best features

Let's assume a microarray data set with n observations, where a factor will be called Y and all descriptive variables $(g_1, g_2, \ldots, g_p)$ will be included in X matrix $(n \times p)$. When performing ANOVA or Levene's test we have p tests:

$$\begin{aligned} ANOVA(g_1 \sim y) \\ ANOVA(g_2 \sim y) \\ ANOVA(g_3 \sim y) \\ \ldots \\ ANOVA(g_p \sim y) \end{aligned} \tag{5.7}$$

Each test uses different data, so we may call them "independent". We want to test each variable (gene), have a look on the means in each group of Y and decide whether such difference is significantly high. When we assume some α level (let it be 0.05) we agree to have an error approximately 5% when rejecting the H_0 hypothesis since it says true. This can be treated as our mistake in judging. When we develop p tests, each with a mistake equal to α, we may get a *family error* much greater than α. According to this assumptions:

- one test doesn't deceive on the $(1 - \alpha_{test})$ percent of cases,
- p tests don't deceive on the $(1 - \alpha_{test})^p$ percent of cases,
- whole family of p tests is surely unerring on the $1 - (1 - \alpha_{test})^p$ percent of cases.

If we assume

$$\alpha_{family} = 1 - (1 - \alpha_{test})^p, \tag{5.8}$$

after some transformations we get

$$\alpha_{test} = 1 - \sqrt[p]{1 - \alpha_{family}} \tag{5.9}$$

as a single test error.

So to have sure that this mistake on whole family is not greater than α we should follow the formula (5.9). Approximately we can divide the significance level of each test by the number of tests - this is called the *Bonferroni adjustment*.

However in practice very often a family of p ANOVA or Levene's tests, when assuming some low α, are too much restrictive and leave too less features as significantly discriminative. It happens especially when the number of features is greater than 1000 - the significance level of testing one feature becomes extremally low!

The alternative approach is not paying attention on the significance of the family of p independent tests, but filter the best features as a percent of all features having been tested (better is to say - to reject the worse features and leave some "good enough"). Our goal in this early stage is just to select some amount of features, each being a "potentially discriminative" feature, having some good "properties" in differentiation.

In the authors opinion, this step should leave about 10% up to 15% of features - the decision should be made arbitrarily. The step doesn't have to be very restrictive, even shouldn't be, to avoid loosing some important features - important when together with other "strong-in-a-sense-of-mean" feature. Let's stress that in this stage we are testing descriptive/discriminative properties of each variable **alone** - we don't check the properties of any pair of variables, any triple, etc.

5.2.4 Algorithms

Let's assume a microarray data set with n observations consisting of one binary variable, let it be Y, and p continuous variables (features) representing different genes, let it be a matrix called X. First algorithm (alg. 5.2.1) executes p ANOVA tests and returns a vector of p p-values. Second algorithm (alg. 5.2.2) executes p Levene's tests and similarly returns a vector of p p-values. Algorithm 5.2.3 takes some number of the best features by selecting the features with the lowest p-value from ANOVA tests ("ANOVA subset") or the features with the lowest p-value from Levene's tests ("Levene's subset"). Author proposes to take more variables according to the ANOVA tests than to Levene's tests, for example 15% of the best features according to ANOVA and 5% of the best features according to Levene's test. Of course we may get some variables which belong to an intersection of those two subsets - that's why in this particular case we finally filter between 15% and 20% of features from the original microarray data set.

Algorithm 5.2.1

Input: M - microarray data set with p features.
Output: a vector of p-values.

- *Execute ANOVA test for each feature, return a p-value of the test.*

Algorithm 5.2.2

Input: M - microarray data set with p features.
Output: a vector of p-values.

- *Execute Levene's test for each feature, return a p-value of the test.*

Algorithm 5.2.3

Input: M - microarray data set, percent.mean - percent of the variables filtered from ANOVA tests, percent.deviation - percent of the variables filtered from Levene's tests.
Output: filtered microarray data set.

1. *From the M data set take the percent.mean of the features with the lowest p-values.*
2. *From the M data set take the percent.deviation of the features with the lowest p-values.*
3. *Take the union of the data sets from 1) and 2).*
4. *Return a filtered data set consisted of filtered features from 3).*

Chapter 6

Experimental methods

6.1 Main idea

In the previous chapter we put attention on predictive/descriptive properties of a single feature. Let's assume a microarray data set with n observations which have one binary variable - a target, let it be Y, and q continuous variables - features, let it be a matrix called X_q. Since we have selected a subset of features using simple filtering methods, which provided data set size reduction, it's high time we looked at **interactions** between features and paid attention on descriptive properties of more than one feature. According to common methodology of feature selection we can use the classifier's accuracy as the performance measure. Our aim in building classifiers would be to achieve the highest predictive accuracy possible. Since we have build some number of classifiers and have selected the best one according to its predictive accuracy, we take features used by this classifier as the **optimal features**. However, since we have thousands of features and tens of observations, in the even previously filtered microarray data such *wrapper modeling* may be impossible to execute. In the project we would like to use the $\mathcal{AIC}$ measure as a measure of a classifier's predictive accuracy. Since the target is binary, we would use a logistic regression as the model, therefore a number of features q taken into the model should be less than the number of observations n. We can deal with the problem by selecting all k-dimensional subsets ($k \leq n$) out of q possible features ($k \ll q$), but this algorithm seems to be impossible to execute in a sensible period of time due to its very high computational complexity. Moreover it would be profitable to reduce or even remove redundancies hidden in the data.

The idea of this step is to select some best features according to:

- correlation between features,
- *predictive accuracy improvement* which results from adding a second feature to a single feature model.

Before we discuss algorithms let's introduce definitions of $\mathcal{AIC}$ *matrices* and *improvement* $\mathcal{AIC}$ *matrices*.

6.2 $\mathcal{AIC}$ matrices

For a sake of conveniency let's say that

- m_{ij} stands for a logistic regression model based on the $y \sim v_i + v_j$ formula,

- m_i stands for a logistic regression model based on the $y \sim v_i$ formula,
- m_0 stands for a logistic regression model where a classifier is described only by an intercept (the $y \sim 1$ formula).

These models can play two different roles:

- a *basic model* role,
- a *full model* role.

Let's consider a matrix of logistic regression basic models over all given features (tab. 6.1). We will call it *basic model matrix*.

	v_1	v_2	...	v_{n-1}	v_n
v_1	m_0	m_1	...	m_1	m_1
v_2	m_2	m_0	...	m_2	m_2
...	...	...	...	...	...
v_{n-1}	m_{n-1}	m_{n-1}	...	m_0	m_{n-1}
v_n	m_n	m_n	...	m_n	m_0

Table 6.1: Basic model matrix

In case of a *basic model matrix* a variable connected with a **row** becomes the basic variable and a model with this variable and an intercept is build in each cell corresponding with this row. Exceptionally, in cells on the diagonal we build a simple $target \sim intercept$ model. Let's imagine we add a variable corresponding with **column** to each model existing in the *basic model matrix*. We receive then a so called *full model matrix* (tab. 6.2).

	v_1	v_2	...	v_{n-1}	v_n
v_1	m_1	$m_{1,2}$	...	$m_{1,n-1}$	$m_{1,n}$
v_2	$m_{2,1}$	m_2	...	$m_{2,n-1}$	$m_{2,n}$
...	...	...	...	...	...
v_{n-1}	$m_{n-1,1}$	$m_{n-1,2}$	...	m_{n-1}	$m_{n-1,n}$
v_n	$m_{n,1}$	$m_{n,2}$	...	$m_{n,n-1}$	m_n

Table 6.2: Full model matrix

For each i, j, $i \neq j$, model $m_{i,j}$ is identical with $m_{j,i}$, hence the *full model matrix* is symmetrical. Let's calculate the $\mathcal{AIC}$ over the proposed models. Let's say that

- $\mathcal{AIC}_{ij}$ stands for an $\mathcal{AIC}$ of a logistic regression model based on the $y \sim v_i + v_j$ formula,
- $\mathcal{AIC}_i$ stands for an $\mathcal{AIC}$ of a logistic regression model based on the $y \sim v_i$ formula,
- $\mathcal{AIC}_0$ stands for an $\mathcal{AIC}$ of a logistic regression model based on the $y \sim 1$ formula.

$\mathcal{AIC}$ matrix corresponding with *full model matrix* will be called *full* $\mathcal{AIC}$ *matrix*, whereas $\mathcal{AIC}$ matrix corresponding with *basic model matrix* will be called *basic* $\mathcal{AIC}$ *matrix*. Both matrices are presented in table 6.3 and 6.4.

The $\mathcal{AIC}$ index tells us how "good" is the model, $\mathcal{AIC}_0$ gives information about a "predictive goodness" of a feature v_i, $\mathcal{AIC}_0$ tells about a predictive accuracy of a model with an

	v_1	v_2	...	v_{n-1}	v_n
v_1	$\mathcal{AIC}_1$	$\mathcal{AIC}_{1,2}$	...	$\mathcal{AIC}_{1,n-1}$	$\mathcal{AIC}_{1,n}$
v_2	$\mathcal{AIC}_{2,1}$	$\mathcal{AIC}_2$	...	$\mathcal{AIC}_{2,n-1}$	$\mathcal{AIC}_{2,n}$
...	...	...	...	...	...
v_{n-1}	$\mathcal{AIC}_{n-1,1}$	$\mathcal{AIC}_{n-1,2}$	...	$\mathcal{AIC}_{n-1}$	$\mathcal{AIC}_{n-1,n}$
v_n	$\mathcal{AIC}_{n,1}$	$\mathcal{AIC}_{n,2}$	...	$\mathcal{AIC}_{n,n-1}$	$\mathcal{AIC}_n$

Table 6.3: $\mathcal{AIC}$ for full model matrix

	v_1	v_2	...	v_{n-1}	v_n
v_1	$\mathcal{AIC}_0$	$\mathcal{AIC}_1$	...	$\mathcal{AIC}_1$	$\mathcal{AIC}_1$
v_2	$\mathcal{AIC}_2$	$\mathcal{AIC}_0$	...	$\mathcal{AIC}_2$	$\mathcal{AIC}_2$
...	...	...	...	...	...
v_{n-1}	$\mathcal{AIC}_{n-1}$	$\mathcal{AIC}_{n-1}$	...	$\mathcal{AIC}_0$	$\mathcal{AIC}_{n-1}$
v_n	$\mathcal{AIC}_n$	$\mathcal{AIC}_n$	...	$\mathcal{AIC}_n$	$\mathcal{AIC}_0$

Table 6.4: $\mathcal{AIC}$ for basic model matrix

intercept, which is a mean of a target, whereas $\mathcal{AIC}_{i,j}$ tells how good are predictive properties of features v_i and v_j in describing the dependent variable. But what does the relation between $\mathcal{AIC}$ indexes mean? What we can say about features v_i and v_j, if for example $\mathcal{AIC}_i$ is greater than $\mathcal{AIC}_j$? Now for some i, j let's consider several situations:

1. when $\mathcal{AIC}_i < \mathcal{AIC}_0$, a model with v_i better describes a target than a mean of a target,
2. when $\mathcal{AIC}_i < \mathcal{AIC}_j$, a model with v_i better describes a target than a model with v_j,
3. when $\mathcal{AIC}_{i,j} < \mathcal{AIC}_i$, a model with two variables, v_i and v_j, better describes a target than a model with only v_i. An addition of v_j to the model with v_i has improved the model's predictive accuracy.
4. when $\mathcal{AIC}_{i,j} > \mathcal{AIC}_i$, a model with two variables, v_i and v_j, is worse than a model with only v_i. An addition of v_j to the model with v_i has only worsened the model's predictive accuracy.

If, after adding some variable to the existing model, $\mathcal{AIC}$ of the new model got lower, it means the new variable has a positive impact on describing a target together with the existing variable. We can say the new variable "helped" the previous variable. The lower the new $\mathcal{AIC}$ is (comparing to the $\mathcal{AIC}$ of the previous model), the better *predictive progress* has been made. Taking the foregoing into account, we can consider two kinds of $\mathcal{AIC}$ **predictive accuracy improvement**:

- absolute improvement - $\mathcal{AI}^{\mathcal{AIC}}$,

$$\mathcal{AI}^{\mathcal{AIC}} = \mathcal{AIC}(basic\ model) - \mathcal{AIC}(full\ model), \tag{6.1}$$

- relative improvement - $\mathcal{RI}^{\mathcal{AIC}}$,

$$\mathcal{RI}^{\mathcal{AIC}} = \frac{\mathcal{AIC}(basic\ model) - \mathcal{AIC}(full\ model)}{\mathcal{AIC}(basic\ model)}. \tag{6.2}$$

We can imagine a matrix of $\mathcal{AI}^{\mathcal{AIC}}$, where

- $\forall_{i,j,i\neq j}$ a cell (i,j) contains $\mathcal{AI}^{\mathcal{AIC}}_{i,j}$,

$$\mathcal{AI}^{\mathcal{AIC}}_{i,j} = \mathcal{AIC}_i - \mathcal{AIC}_{i,j}, \tag{6.3}$$

- $\forall_{i,j,i=j}$ a cell (i,i) contains $\mathcal{AI}^{\mathcal{AIC}}_{i,i}$,

$$\mathcal{AI}^{\mathcal{AIC}}_{i,i} = \mathcal{AIC}_i - \mathcal{AIC}_0, \tag{6.4}$$

Analogously, we can imagine a matrix of $\mathcal{RI}^{\mathcal{AIC}}$. If a cell from $\mathcal{AI}^{\mathcal{AIC}}$ or $\mathcal{RI}^{\mathcal{AIC}}$ matrix, lying outside the diagonal, contains positive value, variables corresponding to the cell's coordinates better describe the target than the single variable corresponding to the row coordinate. According to the sign of the cells lying on the diagonal we will differentiate two kinds of variables

- if $\mathcal{AI}^{\mathcal{AIC}}_{i,i}$ (or $\mathcal{RI}^{\mathcal{AIC}}_{i,i}$) is **positive** we will call v_i a **strong variable**,
- if $\mathcal{AI}^{\mathcal{AIC}}_{i,i}$ (or $\mathcal{RI}^{\mathcal{AIC}}_{i,i}$) is **negative** we will call v_i a **weak variable**.

A strong variable can describe the target better than only an intercept, whereas a weak variable cannot. However it's possible that a weak variable describes the target fairly well together with other variable. Let's note that for some i,j we can obtain:

1. $\mathcal{AI}^{\mathcal{AIC}}_{i,j} > 0$ and $\mathcal{AI}^{\mathcal{AIC}}_{j,i} > 0$, what means that a model with features v_i and v_j is **better** than a model with only v_i and a model with only v_j. This situation indicates a "cooperation" between two features v_i and v_j (tab 6.5).
2. $\mathcal{AI}^{\mathcal{AIC}}_{i,j} > 0$ and $\mathcal{AI}^{\mathcal{AIC}}_{j,i} < 0$, what means that a model with features v_i and v_j is **better** than a model with only v_i, but it's still **worse** than a model with only v_j. We can link this situation with a "domination" of v_j over v_i (tab 6.6).
3. $\mathcal{AI}^{\mathcal{AIC}}_{i,j} < 0$ and $\mathcal{AI}^{\mathcal{AIC}}_{j,i} < 0$, what means that a model with features v_i and v_j is **worse** than a model with only v_i and a model with only v_j. This provide a lack of cooperation between the features, which we will call an "interference" (tab 6.7).

$\mathcal{AI}^{\mathcal{AIC}}$	v_1	...	v_i	...	v_j	...	v_n
v_1	...	...	...	...	...	...	...
...	...	...	...	...	...	...	...
v_i	...	...	+	...	+	...	...
...	...	...	...	...	...	...	...
v_j	...	...	+	...	+	...	...
...	...	...	...	...	...	...	...
v_n	...	...	...	...	...	...	...

Table 6.5: $\mathcal{AIC}$ absolute improvement - a cooperation between two strong features v_i and v_j

Experimental methods are based on a conjecture, that if each two features from some feature set F have good predictive accuracy, F might have good predictive accuracy too. For example, having $F = \{v_i, v_j, v_k\}$ with $\mathcal{RI}^{AIC}$ of each pair being positive, we may expect that a model based on the features from F would have good predictive accuracy.

$\mathcal{AI}^{\mathcal{AIC}}$	v_1	...	v_i	...	v_j	...	v_n
v_1	...	...	...	...	...	...	...
...	...	...	...	...	...	...	...
v_i	...	...	+	...	+	...	...
...	...	...	...	...	...	...	...
v_j	...	...	-	...	+	...	...
...	...	...	...	...	...	...	...
v_n	...	...	...	...	...	...	...

Table 6.6: $\mathcal{AIC}$ absolute improvement - a domination of one strong feature v_j over another strong feature v_i

$\mathcal{AI}^{\mathcal{AIC}}$	v_1	...	v_i	...	v_j	...	v_n
v_1	...	...	...	...	...	...	...
...	...	...	...	...	...	...	...
v_i	...	...	+	...	-	...	...
...	...	...	...	...	...	...	...
v_j	...	...	-	...	+	...	...
...	...	...	...	...	...	...	...
v_n	...	...	...	...	...	...	...

Table 6.7: $\mathcal{AIC}$ absolute improvement - an interference of two strong features v_i and v_j

6.3 Removing high correlations with respect to $\mathcal{AIC}$

6.3.1 Goal

Since we have a data set with one factor, let it be Y, and r continuous variables (features) presented in a form of X matrix, it would be profitable if we look at the correlations between features. It's obvious that when two variables are highly correlated they introduce redundancy when taken together to a model. Not uncommonly it turns out, that the smaller m-subset of n-set of features ($m < n$) includes almost the same information than the original n-set. Our goal would be to remove redundancy by removing some features and leave others without any transformations of features left. That's why we don't apply for example Principal Component Analysis.

First of all we have to assume some level of correlation (let it be γ), above which no correlation is acceptable. This arbitrarily taken level should be quite high. The idea of the step is to observe correlations between all features (let's call this set of features as M) and remove a subset of variables (let it be R) fulfilling three conditions:

1. in the $M \setminus R$ set we don't have any correlation higher than γ,
2. the $M \setminus R$ $\mathcal{AIC}$ matrix has the best possible $\mathcal{AIC}$ measure,
3. there doesn't exist such $\tilde{R}$ that fulfills 1) and 2) and $\tilde{R} \subset R$.

If these conditions are complete we are sure that we remove all high correlations and leave "the best possible variables" according to $\mathcal{AIC}$ measure.

6.3.2 Algorithm

For better understanding let's imagine a graph of n vertices (nodes) and $(n-1)!$ edges - each node is linked with all other nodes ($G(V, E)$). Let the nodes be the features and the edges be the correlations and let they relate to our feature set V. Let's then imagine that the worse correlations are painted in red. We have got a graph of n vertices among which, let's say, k vertices are red. We want to obtain a subgraph, $H(W, F)$, where $W \subset V$ and $F \subset E$, with such properties:

1. $H(W, F)$ doesn't have any red edges,
2. the $\mathcal{AIC}$ matrix related to the features from W has the best possible $\mathcal{AIC}$ measure,
3. there doesn't exist such $H'(W', F')$ that fulfills 1) and 2) and $H'(W', F')$ is a subgraph of $H(W, F)$.

Definition 6.3.1 (*Vertex covering*)

Formally, a vertex covering of a graph G is a set of vertices V_c such that each edge of G is incident to at least one vertex in V_c. The set V_c is said to cover the edges of a graph G. A minimum vertex covering is a vertex covering of the smallest possible size.

Algorithm 6.3.1

1. *Find all **minimum vertex coverings** of all red edges - let's call this family $\mathbf{V}_c$.*
2. *For each $V_c \in \mathbf{V}_c$ calculate $\mathcal{AIC}$ measure on the $V - V_c$ set of features.*
3. *Choose such $\tilde{V_c} \in \mathbf{V}_c$ which maximize $\mathcal{AIC}$ measure over all $V - V_c$ set of features, $V_c \in \mathbf{V}_c$.*

Let's find more details of the algorithm using an example.

Example: Let our data set consists of 7 variables with a correlation matrix shown in table 6.8. A graph representation of the data set has been printed in fig 6.1. We can identify each

	v_1	v_2	v_3	v_4	v_5	v_6	v_7
v_1	1	0.1	0.1	0.75	0.2	0.8	0.3
v_2		1	0.3	0.1	0.35	0.25	0.8
v_3			1	0.25	0.4	0.75	0.45
v_4				1	0.9	0.1	0.9
v_5					1	0.5	0.45
v_6						1	0.2
v_7							1

Table 6.8: Correlation table

high correlation with a red edge. Correlations can be listed randomly - no order is required (tab 6.9). We have 6 high, unwanted correlations - *red edges* (re_i). Let's start with drawing a tree. We will be adding new vertices in several combinations to receive all possible coverings. Each path from a root to a leave would be a single covering.

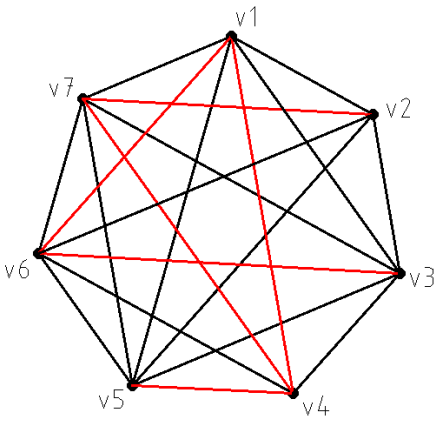


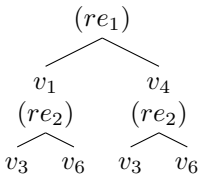
Figure 6.1: A graph representation of tab 6.8

number of correlation	incident vertices	
re_1	v_1	v_4
re_2	v_3	v_6
re_3	v_2	v_7
re_4	v_4	v_7
re_5	v_4	v_5
re_6	v_1	v_6

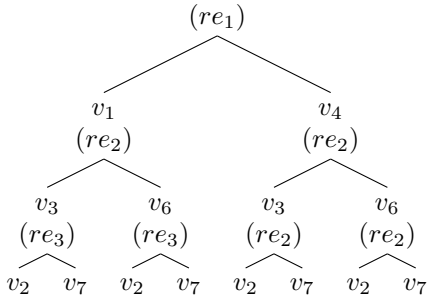
Table 6.9: Listed correlations

1. First red edge can be covered by a vertex v_1 or v_4.

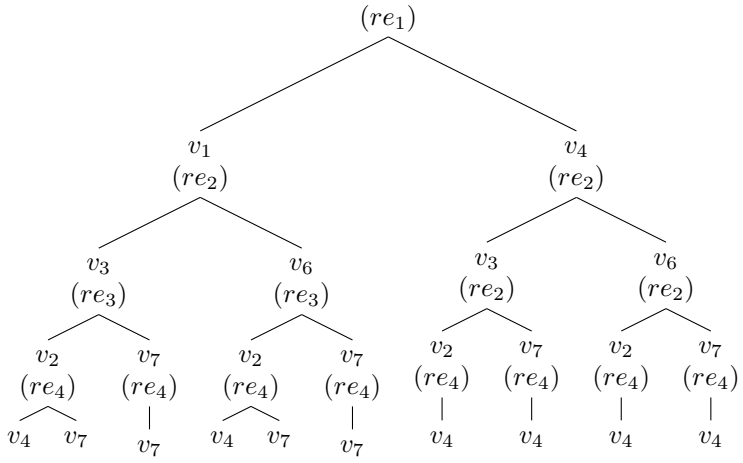
2. Second red edge can be covered by a vertex v_3 or v_6. These vertices are new to each covering.



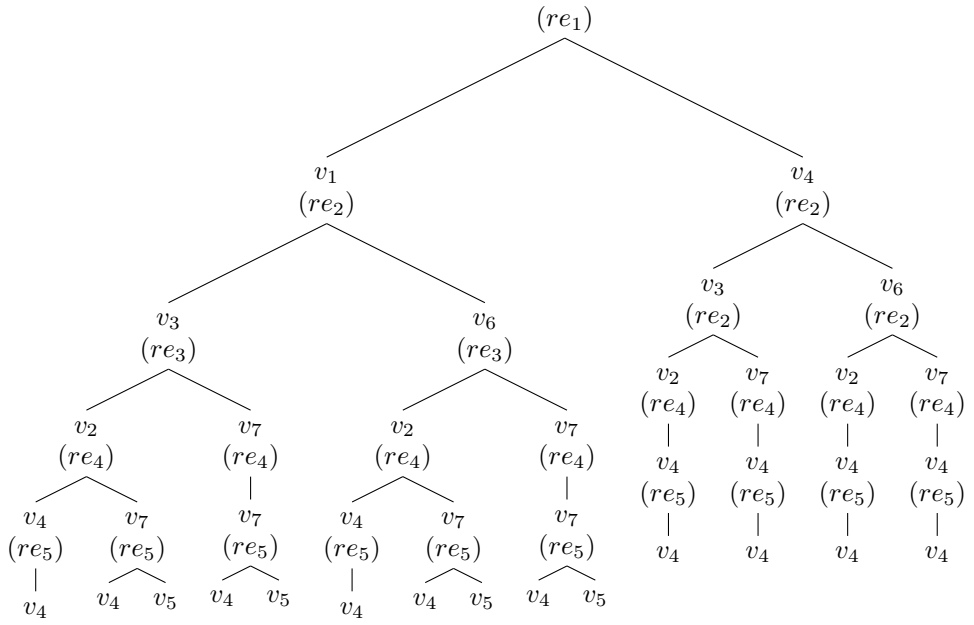
3. Third red edge we can cover by a vertex v_2 or v_7. These vertices are also new to each existing covering.



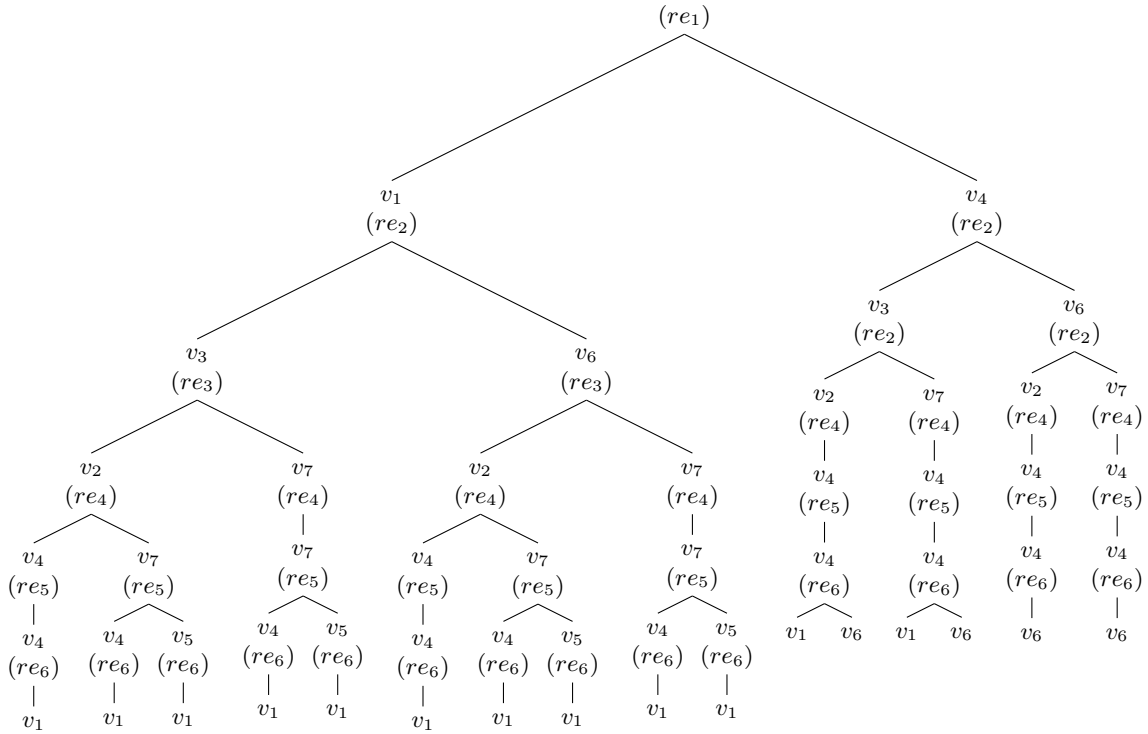
4. Is possible to cover fourth red edge by a vertex v_4 or v_7, however if one vertex has previously appeared in the covering, we use it again - if possible we try to avoid taking new vertices to the covering. Of course are behaving like that among all steps.



5. Fifth red edge can be covered by a vertex v_4 or v_5. If possible we cover an edge by the vertices which already exist in the covering.



6. Sixth red edge we cover by a vertex v_1 or v_6. If both appeared in certain covering, then it doesn't mather which of them we will choose.



We received 16 proposed coverings, each as a result of processing one path in the tree (tab. 6.10).

covering number	features
V_c^1	$\{1,3,2,4\}$
V_c^2	$\{1,3,2,7,4\}$
V_c^3	$\{1,3,2,7,5\}$
V_c^4	$\{1,3,7,4\}$
V_c^5	$\{1,3,7,5\}$
V_c^6	$\{1,6,2,4\}$
V_c^7	$\{1,6,2,4,7\}$
V_c^8	$\{1,6,2,7,5\}$
V_c^9	$\{1,6,7,4\}$
V_c^{10}	$\{1,6,7,5\}$
V_c^{11}	$\{4,3,2,1\}$
V_c^{12}	$\{4,3,2,6\}$
V_c^{13}	$\{4,3,7,1\}$
V_c^{14}	$\{4,3,7,6\}$
V_c^{15}	$\{4,6,2\}$
V_c^{16}	$\{4,6,7\}$

Table 6.10: Obtained coverings

Finally we choose only different minimal vertex coverings, we remove duplicates and these coverings which are supersets of other coverings. In our example we received 6 different minimal vertex coverings as presented in table 6.11. Since we have found the minimal vertex coverings we calculate $\mathcal{AIC}$ measure for each $V - V_c^i$ set of features, let V be an initial set of features, we finally take such covering $\tilde{V}_c$ which gives **the best $\mathcal{AIC}$ measure** on the $V - \tilde{V}_c$ set.

□

covering number	features
V_c^1	$\{2,4,6\}$
V_c^2	$\{4,6,7\}$
V_c^3	$\{1,5,6,7\}$
V_c^4	$\{1,2,3,4\}$
V_c^5	$\{1,3,5,7\}$
V_c^6	$\{1,3,4,7\}$

Table 6.11: Minimal vertex coverings

As the algorithm 6.3.1 has an exponential computational complexity it is sometimes worth to use some simplifier procedure (alg. 6.3.2), which may not give the best possible result, however tries to maximize the $\mathcal{AIC}$ measure in each step.

Algorithm 6.3.2

1. *Sort all red edges in a descending order, create a list* L_{re}
2. *Perform until* L_{re} *is empty*
   - *Take the first red edge* re_1 *from the* L_{re}*.*
   - *Calculate the average* $\mathcal{AIC}$ *measure in column related to the first incident vertex* $v^1_{re_1}$ *of* re*.*
   - *Calculate the average* $\mathcal{AIC}$ *measure in column related to the second incident vertex* $v^2_{re_1}$ *of* re*.*
   - *Remove the vertex* $\tilde{v}_{re_1} \in \left\{v^1_{re_1}, v^2_{re_1}\right\}$ *which has lower average* $\mathcal{AIC}$ *column measure.*
   - *Remove all red edges* $re_i \in L_{re}$*,* $i = 1, \ldots, r$*, which are incident to the vertex* $\tilde{v}_{re_1}$

6.4 Filtering the most helpful features according to $\mathcal{AIC}$

6.4.1 Goal

Since the data size is huge (thousands or hundreds of features) a classifier cannot be directly applied to the whole data set yet. However we already want pay attention on features predictive properties. Since we have chosen $\mathcal{AIC}$ index as a predictive accuracy measure we pay attention on the $\mathcal{AIC}$ improvement as a general result of developing model with two variables in comparison to the model with one variable. We came up with that methodology not only for the reason of a very good and intuitive matrix representation of two variable models, but the conjecture that it's possible to "approximate" predictive accuracy for multivariable model by predictive accuracy of many bivariable models. Being more precise, if we consider many bivariable models on the basis of some feature set (every combination of two features appear) which all have positive predictive accuracy improvement (adding second feature to the first is helpful in a sense of $\mathcal{AIC}$ improvement), the multivariable model based on the whole feature set may present good predictive accuracy too.

Let's have a look on the sample $\mathcal{AI}^{\mathcal{AIC}}$ matrix (tab 6.12). We see cooperation of four features, two of them are strong (v_1, v_i) and two are weak (v_j, v_n). The algorithm chooses the features so as to obtain the **cooperation** between them, no matter if the variable is weak or strong.

$\mathcal{AI}^{\mathcal{AIC}}$	v_1	...	v_i	...	v_j	...	v_p
v_1	+	...	+	...	+	...	+
...	...	...	...	...	...	...	...
v_i	+	...	+	...	+	...	+
...	...	...	...	...	...	...	...
v_j	+	...	+	...	-	...	+
...	...	...	...	...	...	...	...
v_p	+	...	+	...	+	...	-

Table 6.12: $\mathcal{AIC}$ absolute improvement matrix - a cooperation between every two features

Since the algorithm goes forward, starts from one feature and adds another features, the question is which feature to choose as the first feature. The selection of the first feature determines the selection of other variables to a certain degree. First feature should be helpful to both, the target and other features.
We specify three selection methods:

1. the best column sum in $\mathcal{AI}^{\mathcal{AIC}}$ or $\mathcal{RI}^{\mathcal{AIC}}$ matrix,

2. the best column sum in $\mathcal{AI}^{\mathcal{AIC}}$ or $\mathcal{RI}^{\mathcal{AIC}}$ matrix considering only those columns which have the maximum number of positive cells,

3. the best element on the diagonal.

First approach try to find such a feature, which introduces the best "helpfulness" considering all feature set. Second way is first to select features, which bring "help" to the maximum number of other features (positive cells), second to choose the most "helpful" feature (the best column sum). Third approach is to just select the strongest feature among all. It's possible to mix these criteria.

6.4.2 Algorithm

For a sake of conveniency let's say that

- FF stands for a *final feature set*, the set containing finally selected features (by this algorithm),

- FL stands for a *features left set*, the set of available features we have at the beginning of each step of the algorithm,

- FC stands for a *features candidate set*, the set containing candidates to FF.

Algorithm 6.4.1

1. *Select first feature, according to the chosen algorithm of searching the best feature. Add this feature to FF. Take the initial feature set without the first feature as FL.*
2. *Perform in a loop until FL is empty:*
   - *Choose feature candidates into feature candidates set FC as those features f_c from FL, which fulfill:*
   $$\forall_{f \in FF} \left(\mathcal{RI}^{\mathcal{AIC}}_{f,f_c} > 0 \wedge \mathcal{RI}^{\mathcal{AIC}}_{f_c,f} > 0 \right). \tag{6.5}$$
   - *Check the weight of each feature candidate $f_c \in FC$ as*
   $$W^{\mathcal{RI}}_{f_c,FC} = \sum_{f \in FC} \left(\mathcal{RI}^{\mathcal{AIC}}_{f,f_c} + \mathcal{RI}^{\mathcal{AIC}}_{f_c,f} \right). \tag{6.6}$$
   - *Take such $\tilde{f}_c \in FC$ which has the highest candidate weight $W^{\mathcal{RI}}_{\tilde{f}_c,FC}$ and add it to FF.*
   - *Update FL set by excluding the selected feature $\tilde{f}_c$ from FC.*
   $$FL = FC \setminus \left\{ \tilde{f}_c \right\} \tag{6.7}$$
3. *Return FF as the final feature set.*

For better understanding let's follow the example below.

Example: Let's take a few features from leukemia data set:

$$leukemia.exp = \{g48,\ g49,\ g50,\ g65,\ g88,\ g92,\ g98,\ g112,\ g133,\ g134,\ g136,\ g139\}. \tag{6.8}$$

This features come from previously filtered by ANOVA and Levene's tests data set. Let's do a mapping:

$$g48 \to v_1,\ g49 \to v_2, \ldots,\ g136 \to v_{11},\ g139 \to v_{12}, \tag{6.9}$$

and build an $\mathcal{RI}^{\mathcal{AIC}}$ matrix (tab. 6.13). Let's calculate some measures:

- Sum of $\mathcal{RI}^{\mathcal{AIC}}$ in columns (AIC.wgt.sum.col):

$\mathcal{RI}^{\mathcal{AIC}}$	v_1	v_2	v_3	v_4	v_5	v_6	v_7	v_8	v_9	v_{10}	v_{11}	v_{12}
	0.781	0.848	1.011	2.233	2.742	1.024	1.049	0.767	0.577	1.604	0.521	1.867

- Sum of $\mathcal{RI}^{\mathcal{AIC}}$ in rows (AIC.wgt.sum.row):

$\mathcal{RI}^{\mathcal{AIC}}$	v_1	v_2	v_3	v_4	v_5	v_6	v_7	v_8	v_9	v_{10}	v_{11}	v_{12}
	1.163	1.274	1.216	1.213	1.203	1.318	1.182	1.214	1.184	1.286	1.066	1.704

$\mathcal{RI}^{\mathcal{AIC}}$	v_1	v_2	v_3	v_4	v_5	v_6	v_7	v_8	v_9	v_{10}	v_{11}	v_{12}
v_1	0,08	-0,012	0,003	0,248	0,201	0,088	0,052	0,067	0,037	0,180	0,040	0,175
v_2	-0,007	0,080	-0,002	0,237	0,250	0,070	0,089	0,061	0,054	0,157	0,053	0,233
v_3	-0,012	-0,023	0,098	0,233	0,229	0,058	0,077	0,055	0,036	0,168	0,057	0,238
v_4	0,139	0,123	0,136	0,200	0,257	-0,026	0,124	0,036	0,000	0,022	0,029	0,173
v_5	0,035	0,090	0,083	0,216	0,242	0,209	0,029	0,059	0,046	0,148	0,005	0,042
v_6	0,081	0,058	0,065	0,096	0,340	0,092	0,169	0,072	0,029	0,078	0,071	0,167
v_7	0,029	0,063	0,070	0,216	0,177	0,156	0,106	0,070	0,037	0,152	0,006	0,101
v_8	0,072	0,060	0,074	0,161	0,224	0,084	0,096	0,080	0,079	0,122	0,040	0,122
v_9	0,055	0,067	0,068	0,142	0,225	0,054	0,077	0,092	0,067	0,096	0,052	0,189
v_{10}	0,124	0,095	0,124	0,086	0,246	0,022	0,115	0,058	0,015	0,143	0,058	0,199
v_{11}	0,051	0,060	0,082	0,162	0,187	0,089	0,041	0,048	0,046	0,129	0,073	0,098
v_{12}	0,130	0,187	0,209	0,237	0,164	0,128	0,074	0,070	0,129	0,209	0,037	0,132

Table 6.13: leukemia.exp.AIC.perc.matrix

We can interpret AIC.wgt.sum.col[i] as "how much is variable v_i helpfull to other variables" and AIC.wgt.sum.row[i] as "how much other variables help variable v_i". We will select first feature according to the best "AIC.wgt.sum.col" and add the feature to the *final feature set* FF.

$$FF = \{v_5\}\,.$$

We obtain *feature left set* FL as the initial feature set without the chosen variable. Since we have chosen first feature, we start to process the main algorithm.

1^{st} selection

- We choose feature candidates into *feature candidates set* FC as those features f_c from FL, which fulfill:

$$\forall_{f \in FF} \left(\mathcal{RI}^{\mathcal{AIC}}_{f,f_c} > 0 \wedge \mathcal{RI}^{\mathcal{AIC}}_{f_c,f} > 0 \right). \tag{6.10}$$

$$FC = \{v_1, v_2, v_3, v_4, v_6, v_7, v_8, v_9, v_{10}, v_{11}, v_{12}\}$$

- We check the weight of each feature candidate $f_c \in FC$ as

$$W^{\mathcal{RI}}_{f_c,FC} = \sum_{f \in FC} \left(\mathcal{RI}^{\mathcal{AIC}}_{f,f_c} + \mathcal{RI}^{\mathcal{AIC}}_{f_c,f} \right). \tag{6.11}$$

$W^{\mathcal{RI}}_{f_c,\{v_5\}}$	v_1	v_2	v_3	v_4	v_6	v_7	v_8	v_9	v_{10}	v_{11}	v_{12}
	0.035	0.090	0.083	0.216	0.209	0.029	0.059	0.046	0.148	0.005	0.042

- We take $\tilde{f}_c \in FC$ to FF as a feature candidate with the highest candidate weight $W^{\mathcal{RI}}_{\tilde{f}_c,FC}$ - in this step we add v_4.

$$FF = \{v_5, v_4\}\,.$$

- Update FL set by excluding the selected feature $\tilde{f}_c$ from FC.

$$FL = \{v_1, v_2, v_3, v_6, v_7, v_8, v_9, v_{10}, v_{11}, v_{12}\}\,.$$

2^{nd} selection

- We choose feature candidates into *feature candidates set* FC:

$$FC = \{v_1, v_2, v_3, v_7, v_8, v_{10}, v_{11}, v_{12}\}$$

  We removed v_6 and v_9, because $\mathcal{RI}^{\mathcal{AIC}}_{4,6} < 0$ and $\mathcal{RI}^{\mathcal{AIC}}_{4,9} = 0$.
- We check the weight of each feature candidate $f_c \in FC$:

$W^{\mathcal{RI}}_{f_c,\{v_5,v_4\}}$	v_1	v_2	v_3	v_7	v_8	v_{10}	v_{11}	v_{12}
	0.175	0.213	0.219	0.153	0.094	0.170	0.035	0.214

- We take $\tilde{f}_c \in FC$ to FF as a feature candidate with the highest candidate weight $W^{\mathcal{RI}}_{\tilde{f}_c,FC}$ - in this step we add v_3.

$$FF = \{v_5, v_4, v_3\}.$$

- Update FL set:

$$FL = \{v_1, v_2, v_7, v_8, v_{10}, v_{11}, v_{12}\}.$$

3^{rd} selection

- We choose feature candidates into *feature candidates set* FC:

$$FC = \{v_7, v_8, v_{10}, v_{11}, v_{12}\}$$

- We check the weight of each feature candidate $f_c \in FC$:

$W^{\mathcal{RI}}_{f_c,\{v_5,v_4,v_3\}}$	v_7	v_8	v_{10}	v_{11}	v_{12}
	0.230	0.150	0.338	0.091	0.452

- We take $\tilde{f}_c \in FC$ to FF as a feature candidate with the highest candidate weight $W^{\mathcal{RI}}_{\tilde{f}_c,FC}$ - in this step we add v_{12}.

$$FF = \{v_5, v_4, v_3, v_{12}\}.$$

- Update FL set:

$$FL = \{v_7, v_8, v_{10}, v_{11}\}.$$

4^{th} selection

- We choose feature candidates into *feature candidates set* FC:

$$FC = \{v_7, v_8, v_{10}, v_{11}\}$$

- We check the weight of each feature candidate $f_c \in FC$:

$W^{\mathcal{RI}}_{f_c,\{v_5,v_4,v_3,v_{12}\}}$	v_7	v_8	v_{10}	v_{11}
	0.304	0.219	0.547	0.128

- We take $\tilde{f}_c \in FC$ to FF as a feature candidate with the highest candidate weight $W^{\mathcal{RI}}_{\tilde{f}_c,FC}$ - in this step we add v_{10}.

$$FF = \{v_5, v_4, v_3, v_{12}, v_{10}\}.$$

- Update FL set:

$$FL = \{v_7, v_8, v_{11}\}.$$

Following the algorithm we finally receive

$$FF = \{v_5, v_4, v_3, v_{12}, v_{10}, v_7, v_8, v_{11}\}. \tag{6.12}$$

Now let's develop several models:

1. Full model without filtering features according to $\mathcal{AIC}$ matrix(leukemia.exp.full):

```
Call:
glm(formula = Y ~ g48 + g49 + g50 + g65 + g92 + g98 + g112 +
    g133 + g134 + g136 + g139, family = binomial, data = leukemia.filter1)

Deviance Residuals:
      Min         1Q     Median         3Q        Max
-1.061179  -0.239213  -0.016011   0.003217   3.390871

Coefficients:
             Estimate Std. Error z value Pr(>|z|)
(Intercept) -5.519726   3.435509  -1.607  0.10813
g48         -0.004716   0.006751  -0.699  0.48482
g49          0.010480   0.010659   0.983  0.32552
g50          0.009887   0.006450   1.533  0.12529
g65         -0.025659   0.018262  -1.405  0.16002
g92          0.007184   0.013680   0.525  0.59948
g98          0.006830   0.005537   1.233  0.21743
g112         0.005406   0.006567   0.823  0.41037
g133        -0.007094   0.012169  -0.583  0.55991
g134        -0.048941   0.021025  -2.328  0.01993 *
g136        -0.002581   0.002592  -0.996  0.31942
g139         0.006467   0.002388   2.708  0.00678 **
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 92.982  on 71  degrees of freedom
Residual deviance: 22.015  on 60  degrees of freedom
AIC: 46.015
```

2. Model full without filtering features according to $\mathcal{AIC}$ matrix after forward $\mathcal{AIC}$ selection (leukemia.exp.full.back):

```
 Call:
glm(formula = Y ~ g50 + g65 + g134 + g139, family = binomial,
    data = leukemia.filter1)
```

```
Deviance Residuals:
      Min         1Q     Median         3Q        Max
-1.299924  -0.225512  -0.041118   0.005087   3.363944

Coefficients:
             Estimate Std. Error z value Pr(>|z|)
(Intercept) -7.660082   3.108758  -2.464  0.01374 *
g50          0.011908   0.004357   2.733  0.00628 **
g65         -0.012724   0.006725  -1.892  0.05851 .
g134        -0.051518   0.020475  -2.516  0.01187 *
g139         0.007117   0.002615   2.721  0.00650 **
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 92.982  on 71  degrees of freedom
Residual deviance: 26.250  on 67  degrees of freedom
AIC: 36.25
```

---

3. Model full based on the features selected by the $\mathcal{AIC}$ algorithm (leukemia.exp.AIC.full):

```
 Call:
glm(formula = Y ~ g88 + g65 + g50 + g139 + g134 + g98 + g112 +
    g136, family = binomial, data = leukemia.filter1)

Deviance Residuals:
       Min          1Q      Median          3Q         Max
-1.404e+00  -7.048e-03  -8.754e-06   2.107e-08   2.140e+00

Coefficients:
             Estimate Std. Error z value Pr(>|z|)
(Intercept) -24.152246  15.180327  -1.591    0.112
g88           0.012124   0.008234   1.472    0.141
g65          -0.055789   0.038032  -1.467    0.142
g50           0.038308   0.024916   1.537    0.124
g139          0.019711   0.012624   1.561    0.118
g134         -0.147884   0.093992  -1.573    0.116
g98          -0.002405   0.008412  -0.286    0.775
g112          0.007118   0.009363   0.760    0.447
g136         -0.011467   0.010039  -1.142    0.253

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 92.982  on 71  degrees of freedom
Residual deviance: 13.888  on 63  degrees of freedom
AIC: 31.888
```

---

4. Step procedure applied to leukemia.exp.AIC.full:

```
 Start:  AIC=31.89
Y ~ g88 + g65 + g50 + g139 + g134 + g98 + g112 + g136

       Df Deviance    AIC
- g98   1   13.974 29.974
- g112  1   14.553 30.553
<none>      13.888 31.888
- g136  1   16.030 32.030
- g65   1   22.389 38.389
- g88   1   23.823 39.823
- g134  1   28.182 44.182
- g139  1   30.417 46.417
- g50   1   38.321 54.321

Step:  AIC=29.97
Y ~ g88 + g65 + g50 + g139 + g134 + g112 + g136

       Df Deviance    AIC
- g112  1   14.812 28.812
<none>      13.974 29.974
- g136  1   16.487 30.487
- g65   1   22.444 36.444
- g88   1   25.923 39.923
- g134  1   28.803 42.803
- g139  1   30.590 44.590
- g50   1   41.034 55.034

Step:  AIC=28.81
Y ~ g88 + g65 + g50 + g139 + g134 + g136

       Df Deviance    AIC
- g136  1   16.682 28.682
<none>      14.812 28.812
- g65   1   23.884 35.884
- g88   1   26.225 38.225
- g134  1   28.803 40.803
- g139  1   30.661 42.661
- g50   1   41.915 53.915

Step:  AIC=28.68
Y ~ g88 + g65 + g50 + g139 + g134

       Df Deviance    AIC
<none>      16.682 28.682
- g65   1   23.884 33.884
- g88   1   26.250 36.250
- g134  1   28.886 38.886
- g139  1   30.784 40.784
- g50   1   42.022 52.022
```

The final model:

```
Call:
glm(formula = Y ~ g88 + g65 + g50 + g139 + g134, family = binomial,
    data = leukemia.filter1)

Deviance Residuals:
       Min          1Q      Median          3Q         Max
-1.830e+00  -7.486e-02  -2.003e-03   3.155e-06   1.821e+00

Coefficients:
             Estimate Std. Error z value Pr(>|z|)
(Intercept) -19.415704   9.228918  -2.104   0.0354 *
g88           0.006925   0.003365   2.058   0.0396 *
g65          -0.022928   0.012047  -1.903   0.0570 .
g50           0.020125   0.008665   2.323   0.0202 *
g139          0.011832   0.006281   1.884   0.0596 .
g134         -0.085020   0.040482  -2.100   0.0357 *
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 92.982  on 71  degrees of freedom
Residual deviance: 16.682  on 66  degrees of freedom
AIC: 28.682
```

We can see that the $\mathcal{AIC}$ feature selection improved the model's $\mathcal{AIC}$. The full model after $\mathcal{AIC}$ selection (item 3.) has even better $\mathcal{AIC}$ than the first model after backward selection (item 2.). The backward selection applied to the third model has not only still improved the $\mathcal{AIC}$ index and reduced the dimensionality, but has made the coefficients significant. □

6.5 Selecting proposed subsets of features

6.5.1 Goal

Since we have a set of "good" features, each helpful to one another in a sense of AIC matrix, with no high correlations, we can start with wrapper modeling. However if we have tens of features and tens of observations, still the problem of executability appears. We have in mind, that to execute logistic regression the number of features (p) should be no greater than the number of observations (n). Moreover to receive sensible results p should be much less than n.

The idea is to create several subsets of features out of given set of p features. It could be simply done by sampling k features out of p and develop a model on each k-subset. However more profitable is to select subsets of features with some good properties rather than randomly. That's why we will select subsets of features paying attention on correlation so as to still better reduce redundancy. As an example, let's imagine we have a variable v_i correlated with variables v_j and v_k above some level (let's say α). We may say that v_i and v_j share the same information in the α percentage. The same thing we may say about variables

v_i and v_k, but not about variables v_j and v_k, since we haven't considered the correlation between them. If we accept this α level of correlation as "bringing redundancy", we can decide to take only one variable representing (to some α level) one "piece of information". It means we can take:

- only v_i,
- only v_j, since it shares information with v_i,
- only v_k, since it shares information with v_i,
- both v_j and v_k, since they share information with v_i, but don't share information with each other - in other words - are not correlated (correlated less than α).

Summarizing, we would like to create such subsets of features, in which none of the features share some α or higher percentage of information with one another. The α level have to be taken arbitrarily. The lower it is, the less features we take to each subset and the more subsets we obtain.

6.5.2 Algorithm

For a sake of conveniency let's say that

- PS stands for a *proposed subset* - a final subset of not correlated features (in which all correlations are less than a fixed α level).
- FS stands for a *feature set* containing features available to choose in actual step.
- AF stands for an *actual feature* - a feature selected in actual step into *proposed subset*.
- PF stands for *proposed features* - a subset of *feature set*, one feature from this set is analyzed in actual step as an *actual feature*.
- CF stands for *correlated features*, features correlated with *actual feature* which are also in actual *feature set*.
- Cor_{v_i} stands for *correlants of feature* v_i, features correlated with feature v_i (which correlation with feature i is higher than a fixed α level).
- H stands for a *head*. In our algorithm always a first feature from FS becomes a *head*, but it doesn't matter which feature out of FS we accept. This feature and its correlants we assume as *proposed features* - proposed to be a members of different *proposed subsets* PS.

Algorithm 6.5.1

Input: FS - feature set, $\forall_{v_i \in FS} Cor_{v_i}$.
Output: a list of PS - proposed subsets.

1. *Take the first feature from FS and name it a head (H).*
2. *Evaluate proposed features set (PF) as*

$$PF = (Cor_H \cap FS) \cup \{H\}. \tag{6.13}$$

   It means taking a head and all correlants of a head which belong to the actual feature set FS.

3. *For each feature f from PF do*
   - *Set feature f as an actual feature and take it into actual proposed subset PS.*
   - *Evaluate correlated features set CF as*

$$CF = Cor_{AF} \cap FS. \tag{6.14}$$

     CF includes all correlants of an actual feature which belong to the actual feature set FS

   - *Exclude AF and all correlants existing in CF from FS:*

$$FS := FS \setminus (AF \cup CF) \tag{6.15}$$

   - *Execute whole path from 1. - 3. recursively for a new feature set FS until an updated feature set is empty. It means that we have found one proposed subset PS. Add this subset to a list of such subsets.*

4. *Remove all duplicate PS obtaining final PS list.*

For better understanding of the algorithm above let's follow an example.

Example: Imagine we have a data set with 8 features with a correlation matrix shown on the table 6.14. We want to select proposed subsets of features and use each subset and a target to develop a single model. However we want to propose such subsets of features in with no correlation higher or equal than 0.5 exists. Correlations which we want to avoid are painted in red. Let's list all correlants of each variable (tab. 6.15).

	v_1	v_2	v_3	v_4	v_5	v_6	v_7	v_8
v_1	1	0.5	0.1	0.62	0.3	0.3	0.5	0.15
v_2		1	0.55	0.1	0.35	0.65	0.1	0.8
v_3			1	0.25	0.4	0.75	0.6	0.2
v_4				1	0.9	0.1	0.2	0.45
v_5					1	0.5	0.45	0.1
v_6						1	0.2	0.35
v_7							1	0.2
v_8								1

Table 6.14: Correlation table

correlation sets	correlants
cor_{v_1}	v_2 v_4 v_7
cor_{v_2}	v_1 v_3 v_6 v_8
cor_{v_3}	v_2 v_6 v_7
cor_{v_4}	v_1 v_5
cor_{v_5}	v_4 v_6
cor_{v_6}	v_2 v_3 v_5
cor_{v_7}	v_1 v_3
cor_{v_8}	v_2

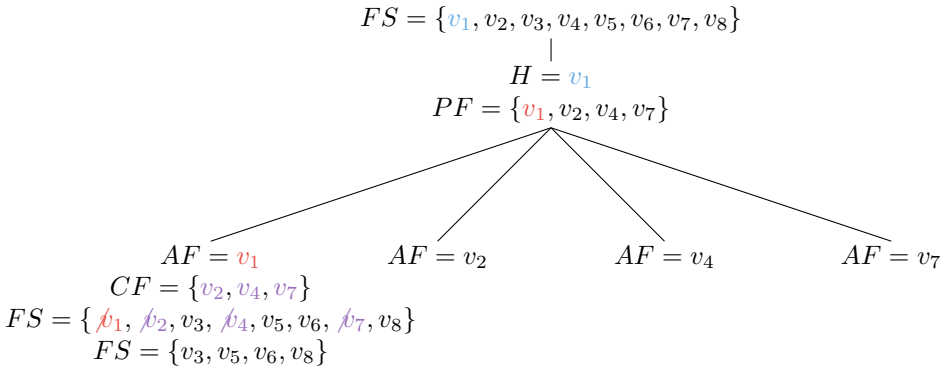
Table 6.15: Correlation sets

Let's follow the algorithm using a tree.

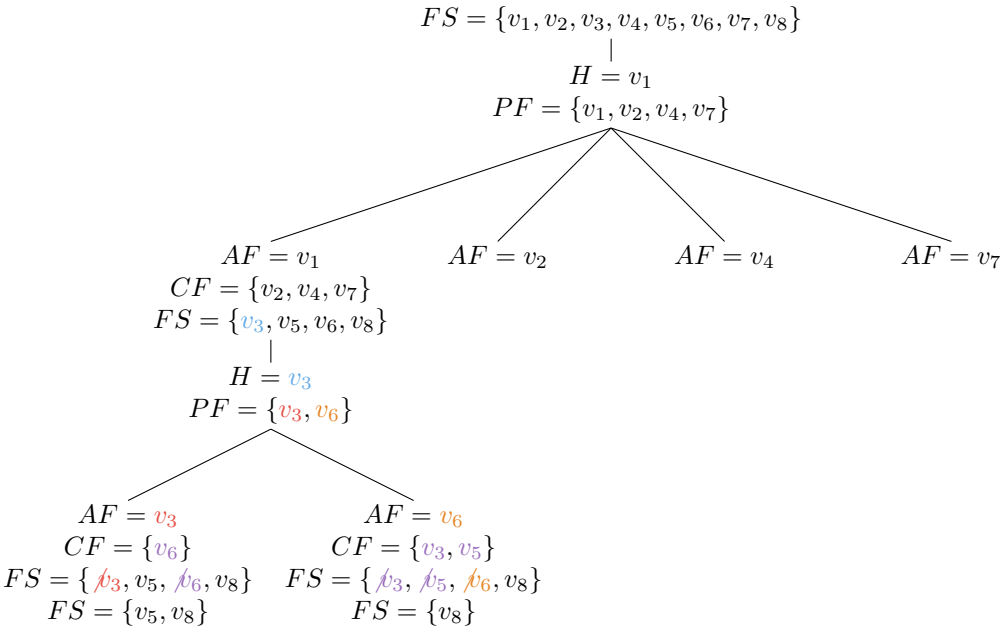
1. We take the initial *feature set* FS, take first feature as a *head* H, evaluate *proposed features* $(PF = (Cor_H \cap FS) \cup \{H\})$

$$FS = \{v_1, v_2, v_3, v_4, v_5, v_6, v_7, v_8\}$$
$$|$$
$$H = v_1$$
$$PF = \{v_1, v_2, v_4, v_7\}$$

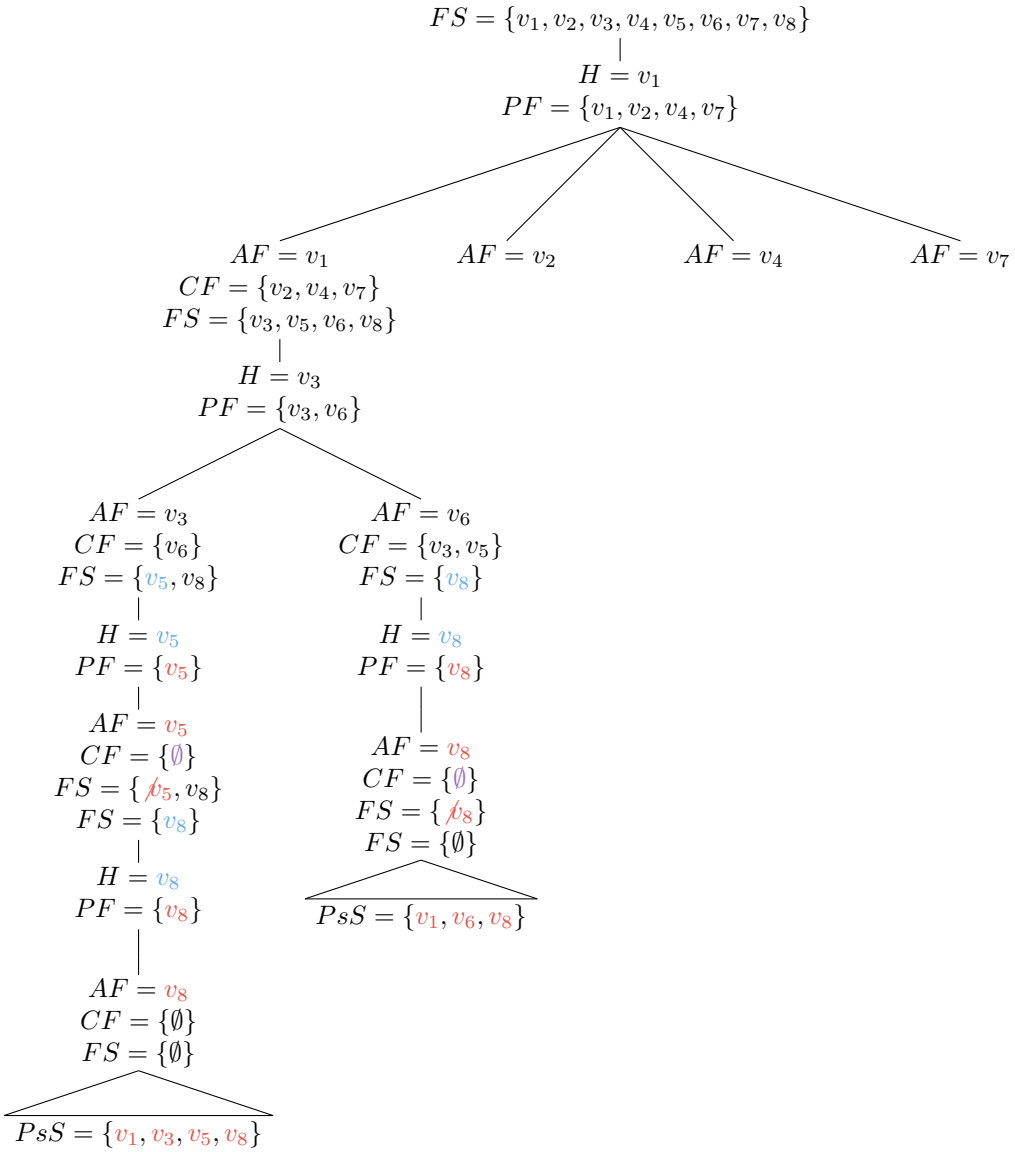
2. We set each variable from PF as an *actual feature* AF. This step creates several options. For a sake of clarity let's pay attention on each option separately and now let's analyze option with $AF = v_1$. We take v_1 into actual *proposed subset* PS, evaluate *correlated features* set CF as $CF = Cor_{AF} \cap FS$ and exclude AF and all its correlants existing in CF from FS ($FS := FS \setminus (AF \cup CF)$).



3. We take the first feature from new FS and set it as *head* H. We execute the same activities as in step 1. and 2. Always *actual feature* AF is added to the existing *proposed subset* PS.



4. We take the first feature from new FS and set it as *head* H. We execute the same activities as in step 1. and 2. When *feature set* FS is empty we receive one *proposed subset* PS from one path.



All paths in this tree give a list of *proposed subsets* PS. After removing all duplicate *proposed subsets* we receive final PS list:

Proposed subset	Variables
PsS_1	$\{v_1, v_3, v_5, v_8\}$
PsS_2	$\{v_1, v_6, v_8\}$
PsS_3	$\{v_2, v_4, v_7\}$
PsS_4	$\{v_2, v_5, v_7\}$
PsS_5	$\{v_3, v_4, v_8\}$
PsS_6	$\{v_4, v_6, v_7, v_8\}$
PsS_7	$\{v_5, v_7, v_8\}$

Table 6.16: Proposed subset list

□

Chapter 7

Model methods

7.1 Bootstrap resampling

Developing a statistical model is always related with certain training data. Good predictive quality obtained on the training data doesn't mean that the model will provide excellent results on any other data set. If the model has been trained on the certain data set, some accuracy measures may be obtained as a result of overfitting the data. The better idea could be to have more data sets, develop models and calculate measures on these sets and compare the results. However we don't always have such a large data set to divide it into many samples - such samples should be independent one from another. For this significant reason some people use resampling techniques so as to create many data samples coming from a one data set. On each data sample we develop a single model and calculate some measures (i.e. predictive accuracy). Then as a real value of a certain indicator we may obtain some statistics (i.e. a mean) involving indicators each calculated on a single sample or we may follow the "voting" of all models.

Let's briefly describe the mechanism of bootstrap resampling. From the original data set S containing n elements we create m samples each of a size n. Each sample s_j, $j = 1, 2, \ldots, m$ is formed in such a way: n times we draw one element from S, "copy" its value to the sample s_j and return the value to the original data set S. Single element is being taken into a sample s_k with the probability amounting to $\frac{1}{n}$ - the same for every observation. The probability for the element not to be taken into one certain sample is equal to

$$(1 - \frac{1}{n})^n = exp(-1) \approx 0.368.$$

It means that on the average a single sample s_j, $j = 1, 2, \ldots, m$, doesn't contain about 0.368 of original sample S. As a result of the resampling process we receive m samples $s_1, s_2, \ldots, s_m$, each including

- repeated observations
- observations which occur in other samples

As it has been proved, bootstrap resampling provides very good results. Since the samples are not quite the same, the models (thus the indicators) are not strongly dependent to one another.

7.2 Bootstrap experimental approaches

7.2.1 Model assumptions

In the project we use bootstrap resampling within two approaches

1. external,
2. internal.

Before we start to discuss the bootstrap approaches let's make few assumptions.

- Let S be the original data set including n tasks (observations).
- Let $F^1, F^2, \ldots, F^M$ be a list of proposed features. Each element F^i, $i = 1, 2, \ldots, M$ contains a set of features which will be used to build a single model.
- Let $B_1, B_2, \ldots, B_R$ be a set of bootstrap sample from the original set S, each having size equal to n.

So as to be clear and avoid confusion we need to explain a few details:

- To **build** a model on a sample s with a feature set f means to build a logistic regression model using observations from s and all features from f. The model will be called **full model**.
- To **develop** a model on a sample s with a feature set f means to build a logistic regression model using observations from s and some subset of features from f. The model is created from a full model by $\mathcal{AIC}$ **step procedure**, which reduces the number of features to the optimal subset. We will call this model **optimal model**. The subset of the feature set f defining the optimal model will be called the **optimal feature set** and denoted by an upper star (*). Since a feature set defines a model we will use notation connected with feature set (f, f^*) when talking about the model.

Notice 7.2.1

> *We don't take into consideration the parameters of a certain model $\underline{\theta}$. We develop or build a model **only** in order to select features and calculate $\mathcal{AIC}$.*

We can develope a certain model in a four different ways:

1. **backward selection** - we start from a full model and remove the less important features from the model one after the other (with an aim to lower $\mathcal{AIC}$),
2. **forward selection** - we start from a model with an intercept and add the most important features to the model one after the other (with an aim to lower $\mathcal{AIC}$),
3. **stepwise backward selection** - we start from a full model and remove and add the less important features from the model one after the other (additions and removals are performed altervatively with an aim to receive the lowest $\mathcal{AIC}$),
4. **stepwise forward selection** - the same mechanism like in stepwise backward selection but starting from a model with an intercept.

7.2.2 External bootstrap

The idea of external bootstrap is to use bootstrap resampling to calculate the $\mathcal{AIC}$ of the previously developed model.

Algorithm 7.2.1

1. *Perform in a M loop iterating by i:*
   (a) *Develop a model using F^i and the initial data set S. Obtain a F^{i*} subset of features.*
   (b) *Perform in a R loop iterating by j:*
      - *Build a model using F^{i*} and bootstrap sample B_j.*
      - *Calculate $\mathcal{AIC}$ of the obtained model ($(\mathcal{AIC}^{F^{i*}})_{B_j}$).*
   (c) *Calculate a mean for all $(\mathcal{AIC}^{F^{i*}})_{B_j}$ indicators for a fixed i ($mean_j((\mathcal{AIC}^{F^{i*}})_{B_j})$). Take this mean as $\mathcal{AIC}$ for the model F^{i*} ($\mathcal{AIC}(F^{i*})$).*
2. *Select the best model (or few best models) $best_i(F^{i*})$ according to the lowest $\mathcal{AIC}^{F^{i*}}$. Take features from $best_i(F^{i*})$ as the **optimal feature set**.*

The algorithm is shown on a table 7.1.

whole sample	bootstrap samples	final measure	final selection
$F^1 \xrightarrow{Develop} F^{1*}$	$B_1: (F^{1*})_{B_1} \xrightarrow{calc.} (\mathcal{AIC}^{F^{1*}})_{B_1}$ $B_2: (F^{1*})_{B_2} \xrightarrow{calc.} (\mathcal{AIC}^{F^{1*}})_{B_2}$ ... $B_R: (F^{1*})_{B_R} \xrightarrow{calc.} (\mathcal{AIC}^{F^{1*}})_{B_R}$	$avg_j((\mathcal{AIC}^{F^{1*}})_{B_j})$	$best_i(avg_j(\mathcal{AIC}^{F^{i*}}_{B_j})$
$F^2 \xrightarrow{Develop} F^{2*}$	$B_1: (F^{2*})_{B_1} \xrightarrow{calc.} (\mathcal{AIC}^{F^{2*}})_{B_1}$ $B_2: (F^{2*})_{B_2} \xrightarrow{calc.} (\mathcal{AIC}^{F^{2*}})_{B_2}$ ... $B_R: (F^{2*})_{B_R} \xrightarrow{calc.} (\mathcal{AIC}^{F^{2*}})_{B_R}$	$avg_j((\mathcal{AIC}^{F^{2*}})_{B_j})$	
...	...	...	
$F^M \xrightarrow{Develop} F^{M*}$	$B_1: (F^{M*})_{B_1} \xrightarrow{calc.} (\mathcal{AIC}^{F^{M*}})_{B_1}$ $B_2: (F^{M*})_{B_2} \xrightarrow{calc.} (\mathcal{AIC}^{F^{M*}})_{B_2}$ ... $B_R: (F^{M*})_{B_R} \xrightarrow{calc.} (\mathcal{AIC}^{F^{M*}})_{B_R}$	$avg_j((\mathcal{AIC}^{F^{M*}})_{B_j})$	

Table 7.1: External bootstrap processing

7.2.3 Internal bootstrap

Internal bootstrap approach is more complicated than the external. First we use bootstrap samples to develop models and calculate $\mathcal{AIC}$ (a certain model is developed inside each bootstrap sample), second the best model shows up as a consequence of models voting.

Algorithm 7.2.2

1. *Perform in a R loop iterating by j:*
   (a) *Perform in a M loop iterating by i:*
      - *Develop a model using F^i and bootstrap sample B_j. Obtain a $F^{i*}_{B_j}$.*
      - *Calculate $\mathcal{AIC}$ of the obtained model ($\mathcal{AIC}^{F^{i*}}_{B_j}$).*

   (b) *Select the best model (or few best models) for all $\mathcal{AIC}^{F^{i*}}_{B_j}$ indicators for a fixed j ($best_i(\mathcal{AIC}^{F^{i*}}_{B_j})$) according to the lowest $\mathcal{AIC}$.*
2. *Select the model (or few models), which appear most often. Take features of this model as the **optimal feature set**.*

The algorithm is shown on a table 7.1.

bootstrap samples		final measure	final selection
B_1 :	$F^1_{B_1} \overset{Develop}{\longrightarrow} F^{1*}_{B_1}$ $F^{1*}_{B_1} \overset{calc.}{\longrightarrow} \mathcal{AIC}^{F^{1*}}_{B_1}$	$best_i(\mathcal{AIC}^{F^{i*}}_{B_1})$	$freq_j(best_i(\mathcal{AIC}^{F^{i*}}_{B_j})$
	$F^2_{B_1} \overset{Develop}{\longrightarrow} F^{2*}_{B_1}$ $F^{2*}_{B_1} \overset{calc.}{\longrightarrow} \mathcal{AIC}^{F^{2*}}_{B_1}$		
	... ...		
	$F^M_{B_1} \overset{Develop}{\longrightarrow} F^{M*}_{B_1}$ $F^{M*}_{B_1} \overset{calc.}{\longrightarrow} \mathcal{AIC}^{F^{M*}}_{B_1}$		
B_2 :	$F^1_{B_2} \overset{Develop}{\longrightarrow} F^{1*}_{B_2}$ $F^{1*}_{B_2} \overset{calc.}{\longrightarrow} \mathcal{AIC}^{F^{1*}}_{B_2}$	$best_i(\mathcal{AIC}^{F^{i*}}_{B_2})$	
	$F^2_{B_2} \overset{Develop}{\longrightarrow} F^{2*}_{B_2}$ $F^{2*}_{B_2} \overset{calc.}{\longrightarrow} \mathcal{AIC}^{F^{2*}}_{B_2}$		
	... ...		
	$F^M_{B_2} \overset{Develop}{\longrightarrow} F^{M*}_{B_2}$ $F^{M*}_{B_2} \overset{calc.}{\longrightarrow} \mathcal{AIC}^{F^{M*}}_{B_2}$		
...	... ...	...	
B_R :	$F^1_{B_R} \overset{Develop}{\longrightarrow} F^{1*}_{B_R}$ $F^{1*}_{B_R} \overset{calc.}{\longrightarrow} \mathcal{AIC}^{F^{1*}}_{B_R}$	$best_i(\mathcal{AIC}^{F^{i*}}_{B_R})$	
	$F^2_{B_R} \overset{Develop}{\longrightarrow} F^{2*}_{B_R}$ $F^{2*}_{B_R} \overset{calc.}{\longrightarrow} \mathcal{AIC}^{F^{2*}}_{B_R}$		
	... ...		
	$F^M_{B_R} \overset{Develop}{\longrightarrow} F^{M*}_{B_R}$ $F^{M*}_{B_R} \overset{calc.}{\longrightarrow} \mathcal{AIC}^{F^{M*}}_{B_R}$		

Table 7.2: Internal bootstrap processing

7.3 Selecting the best model

7.3.1 Model diagnostic

It is obvious that model diagnostic is of crucial importance. It let us tell how well the model fits the given data and, furthermore, how to finally judge the model's adequacy. Even a perfect fit to the training data is not equivalent with the model's adequacy, moreover, it means that the model is inadequate because it doesn't include randomness in the data.

The basic indicator of the model's goodness of fit is the model's deviance. We described the deviance in the first chapter. Model deviance can be treated as a measure of distance between a certain model and the sated model (the best model according to the given model type).

Now let's pay attention on the importance and significance of model's variables. As a rule such analysis is based on the comparison of two model deviances:

- the deviance of the model with a certain variable,
- the deviance of the model without a certain variable.

Let's assume two logistic regression models:

- ω - a model with a certain set of features $F_\omega = \{f_1, f_2, \ldots, f_q\}$,
- Ω - a larger model which contains F_ω and some additional features $F_{\Omega\setminus\omega} = \{f_{q+1}, \ldots, f_p\}$.

We can say that $\omega \subset \Omega$. If we want to test whether one of the variables from $F_{\Omega\setminus\omega}$ brings a valuable information, we can test the hypothesis about the adequacy of the model ω versus the hypothesis about the adequacy of the model Ω:

$$H_0 : \omega, \tag{7.1}$$
$$H_1 : \Omega, \tag{7.2}$$

what in details means:

$$H_0 : \ \beta_{q+1} = \beta_{q+2} = \ldots = \beta_p = 0, \tag{7.3}$$
$$H_1 : \ \exists \beta_i, i = q+1, q+2, \ldots, \beta_i \neq 0. \tag{7.4}$$

As the testing statistic we use such measure

$$D = dev_\omega - dev_\Omega, \tag{7.5}$$

which (when H_0 is true) has asymptoticaly a χ^2 distribution with $p - q$ degrees of freedom. We can perform other analysis/test of variables significance, which is actually performed in glm procedure implemented in R. This test is called **Wald's test** and it uses the same hypothesis as in (7.3). In the simplest case, when we test an influence of a single variable f_i, which relate to β_i, the testing statistic assumes such form

$$\frac{\hat{\beta}_i}{SE(\hat{\beta}_i)}, \tag{7.6}$$

where $SE(\hat{\beta}_i)$ is a standard estimation error for $\hat{\beta}_i$. The statistic is normally distributed.

It is worth to mention, that not rejecting H_0, saying that the coefficient β_i is equal to zero, **does not mean** that a particular variable f_i has no influence on the target! It means, that this influence has been explained by other variables existing in the model. This situation occurs when f_i is correlated with other variables in the model. This phenomenon is called **variable masking**.

Notice 7.3.1

> *Considering the foregoing we have to pay attention not only on the model's accuracy (measured for example by $\mathcal{AIC}$), but on the whole model adequacy including the significance of the coefficients used.*

7.3.2 0.632-bootstrap

When we develop an external bootstrap scenario we use bootstrap samples to count an average $\mathcal{AIC}$c of each previously developed model. Then we can select the best model as the model with the best $\mathcal{AIC}$c. However in practice the model with the best $\mathcal{AIC}$c has also insignificant coefficients (p-values of t-statistics for testing the significance of coefficients usually greater than 0.1). This problem has been described in the forther chapter "Problems and solutions". Then instead model selection by the lowest $\mathcal{AIC}$ we may use the observations that haven't been used in evaluating the $\mathcal{AIC}$c and develop a misclassification error called **0.632-bootstrap**. Let us remind how the bootstrap samples were constructed: from the whole data set of size n we took n randomly selected variables with replacing - in this way we constructed R bootstrap samples. However in every resampling process we didn't take about 0.368 of observations! Let us pay attention to a single resampling process. Let us call the observations taken to the bootstrap sample *bootstrap.train* and the set of observations left in the resampling process -*bootstrap.test*. We can count misclassification errors on both *bootstrap.train* and *bootstrap.test* sample - let us use symbols:

- for a misclassification error on *bootstrap.train*

$$err_{boot.train}$$

- for a misclassification error on *bootstrap.test*

$$err_{boot.test}$$

A misclassification error on *bootstrap.test* is biased - we constructed a classifier basing on about $0.623n$ not n observations. In the other hand $err_{boot.train}$ is an "optimistic" estimator of a real misclassification error performed by out classifier.

Taking above into consideration some people investigated such an estimator of a misclassification error:

$$0.632 err_{boot.test} + 0.368 err_{boot.train}.$$

It turned out that this estimator has very good properties and is worth using [10]. We will use this classifier to select the best model in an external bootstrap process.

Chapter 8

Experimental Part

8.1 The process

The whole experimental process path consists of five steps (each having its substeps):

1. Data preparation
   (a) Constant gene removal
   (b) Overwriting the target values
   (c) Data division into training and testing samples
2. Filter selection
   (a) Evaluating correlation matrix
   (b) Executing ANOVA tests
   (c) Executing Levene's tests
   (d) Selecting some top percentage of features according to the results of ANOVA and Levene's tests
3. Experimental selection
   (a) Evaluating $\mathcal{AIC}$ matrices
   (b) Removing the highest correlations with respect to $\mathcal{AIC}$
   (c) Filtering the best features according to $\mathcal{AIC}$ helpfulness
   (d) Evaluating proposed feature subsets
4. Wrapper selection
   (a) External bootstrap
   (b) Internal bootstrap
5. Testing the accuracy

Data preparation

After the first step we receive data divided into training and validating sample (*data.train* and *data.test*), ready for analyses. We take *data.train* for further analyses and keep *data.test* for the final accuracy assessment.

Filter selection

After this step we receive the data filtered by testing methods (*data.train.filter1*). The dimensionality reduction should be at least 80%, however we should not be too strict. In the project we always kept less than 20% of features, but not more than 2000, due to the further computational complexity and time consumption of the experimental methods.

Experimental selection

Third step consists of four very important algorithms. First evaluates the matrices of $\mathcal{AIC}$ helpfulness, which we will use to execute second and third algorithm. Second algorithm removes very high correlations. The data set (*data.train.filter2*), created after executing this algorithm, doesn't have any single high correlation. Of course we arbitrarily determine the level of absolute correlation value to be assumed as high - in our case it is mostly 0.7. Third algorithm selects the cooperation feature set. The procedure in each step adds a single feature to the existing cooperation feature set, which (this feature) has the best cooperation properties to the mentioned cooperation feature set. The data set after performing third algorithm (*data.train.filter3*) sometimes needs trimming, as the length of proposed non-correlated best features is to large. In the experiments we always take not more than 50 features.

Wrapper selection

Fourth stage consists of two independent processes on the basis of *data.train.filter3*. After performing each kind selection we obtain a list of best proposed feature sets in a descending order. Feature sets which come first have the best $\mathcal{AIC}$ or $\mathcal{AIC}$c indicator. We have to be watchful, because some of the models may have nonsignificant coefficients! This situation/problem has been explained in the next chapter. We take feature set with the best possible $\mathcal{AIC}$ and coefficients being significant.

Testing the accuracy

We test the best feature sets from external and internal bootstrap processes by performing prediction (predict.glm) according to the model developed on the best feature sets. We count misclassification error as the measure of classifiers' accuracy.

8.2 The results - summary

We performed the process on five popular microarray data sets:

1. *colon.tumor*,
2. *leukemia*,
3. *lung.cancer*
4. *prostate.cancer*
5. *breast.cancer*

The results of the experiments are shown in table 8.1 whereas the details in table 8.2. The numbers in braces for $\mathcal{AIC}$ selection method means the methodology used for selecting the best feature set and the number of features, while for other methods they describe the parameters and the name of model used. (**F**) indicates that referenced work proposes a Filter-Based Algorithm, (**W**) for wrapper and (**F-W**) for a combination of both schemes, also enclosed by parenthesis are pointed the no. of genes and inducer employed. We have investigated the external bootstrap path. The internal bootstrap path has been already developed by the author however still needs little improvement, so this can be left for future work.

Work	Validation	Colon Tumor	Leukemia	Lung Cancer	Breast Cancer	Prostate Cancer
$\mathcal{AIC}$c(F-W)	$\mathcal{AIC}$c, B.632	88,2	90,0	91,4	73,7	79,4
		(3, B.632)	(3, B.632)	(2, B.632)	(8, $\mathcal{AIC}$c)	(5, $\mathcal{AIC}$c)
[5](F)	5-2CV	89.68	96.39	98.45	84.45	93.68
		(8,SVM^2)	(2,SVM^R)	(3,LR)	(6,KNN)	(2,SVM^R)
[17](F)	200-B.632	88.75	98.2	-	-	-
		(14,SVM^L)	(23,SVM^L)	-	-	-
[16](W)	10-10CV	85.48	93.40	-	-	-
		(3,NB)	(2,NB)	-	-	-
[14](W)	100-RS	87.31	-	72.20	-	-
		(94,SVM)	-	(23,SVM)	-	-
[3](W)	50-HO	77.00	96.00	99.00	79.00	93.00
		(33,SVM^R)	(30,SVM^R)	(38,SVM^R)	(46,SVM^R)	(47,SVM^R)
[11](FW)	10-10CV	-	-	99.40	-	96.30
		-	-	(135,5NN)	-	(79,5NN)
[18](F)	10CV	-	98.6	99.45	68.04	91.18
		-	(2,SVM)	(5,SVM)	(8,SVM)	(6,SVM)

Table 8.1: Best results of classifiers obtained with $\mathcal{AIC}$ step algorithm plus comparison references using one or more datasets.

Work	Colon Tumor	Leukemia	Lung Cancer	Breast Cancer	Prostate Cancer
features	2000	7129	12533	24481	12600
train/test	45/17	52/20	100/81	78/19	102/34
train.filter1	311	1158	1528	1444	1648
	(aov0.15/levene0.05)	(aov0.15/levene0.05)	(aov0.1/levene0.05)	(aov0.05/levene0.0025)	(aov0.1/levene0.05)
train.filter2	108	846	1326	1144	679
	($cor < 0.8$)	($cor < 0.7$)	($cor < 0.7$)	($cor < 0.7$)	($cor < 0.7$)
train.filter3	21	210 $\rightarrow$ 40	38	152 $\rightarrow$ 50	72 $\rightarrow$ 50
					-
n.subsets	141(0.5)	1002(cor0.55)	700(cor0.5)	1536 (cor0.46)	1354
(max length)	(21)	(19)	(15)	(32) -	(26)
external.bootstrap	86	108	116	91	227
$dev_{best.model}$	19,18	9,73	5.58	9.96	6.06
		-		-	

Table 8.2: The details of the process. The numbers in brackets indicate the step parameters.

The project results are quite comparable with the results according to the references. Only on the prostate.cancer data the results obtained by the author are lower than the other scientists' results. However the potential of feature selection path can be investigated in further analyses.

Chapter 9

Problems and solutions

Within each step we may encounter different problems having four general "backgrounds":

- data,
- statistical,
- computational,
- algorithmic.

Problems having their source in data occurs as the problems of fitting the data. Since the model involves the data a proper task selection is essential. Second kind of problems occurs due to the obtained methodology. They are paradoxes rather than real errors. It's possible to obtain very low $\mathcal{AIC}$, almost the ideal model deviance, but nonsignificant coefficients in the model. We will try to explain such kind of problems in further sections. The computational problems have their cause in the high dimensionality of data taken to analyses. Some algorithms have exponential computational complexity, so it's important to have this in mind while executing algorithms on a large data sets. The algorithmical problems occur rarely and they are connected with R procedures converging.

Division into training and testing sample

When the number of learning tasks is low the probability that we discover a 100% relation just by chance is not so low. That's why we should pay attention on the data taken to the training process. If we encounter the variable, that perfectly fits the target, we should be suspicious rather than content. The perfect separability (as it was discussed in the Microarray Data chapter) towards a single feature is a thing we should avoid. When we can classify correctly 100% observations by involving only one variable others seem to be useless! We should be suspicious and treat this kind of ideal fit as obtained "by chance" rather than resulting from the real existing 100% dependence. We should try to avoid the problem by splitting the data into training and testing set with an aim to obtain **the less possible "regularity"** within the training set. What to do if we always obtain the perfect separability, no matter how we split the data? The simplest answer, however for many people not satisfactory, could be that "we have to take more data". Maybe not a bad idea is just to consult the data with an expert? Whatever we will do it's beyond doubt that this feature needs special attention.

Example: Let's investigate the *leukemia* microarray. We selected a feature *g4847* and draw its boxplot (fig. 9.1) towards a split into training and validating sample. If we switched

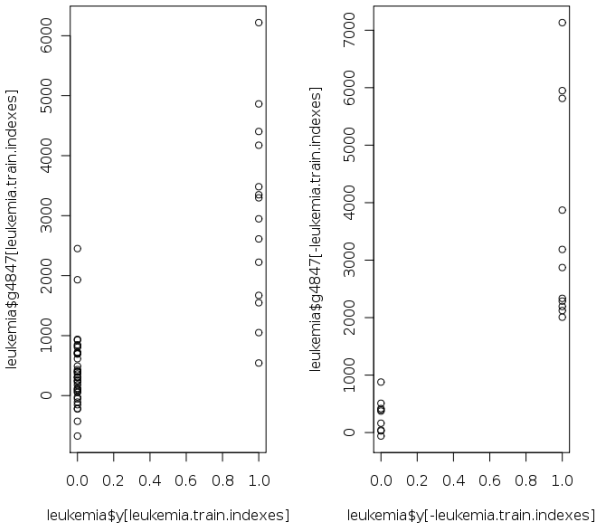


Figure 9.1: Boxplot of gene *g4847* from *leukemia*

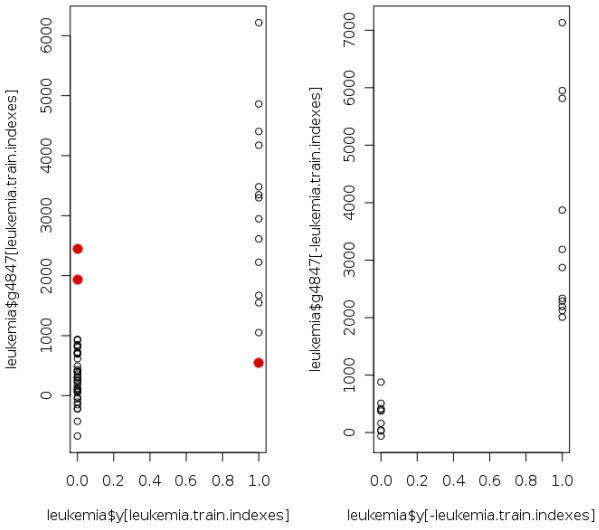


Figure 9.2: Boxplot of gene *g4847* from *leukemia* - three important observations

the destination of the samples (testing sample played a role of a training sample) or remove three features, coloured in red on fig. 9.2, from the training sample, we would obtain an ideal separation according to gene *g4847* in a new training set. □

Minimizing $\mathcal{AIC}$ versus improving significance

The most serious problem we had to face in the project was a conflict between $\mathcal{AIC}$ minimizing and improving significance of models' coefficients. Most of models developed by $\mathcal{AIC}$ improvement turned out to be totally insignificant! The majority of the models based on

the proposed feature sets obtained after the third step of the process (see chapter eight) had very low AIC, as it was expected and wanted, but was insignificant in a term of F-test. After executing external or internal bootstrap the best models appeared as the models with $\mathcal{AIC}$ equal to the doubled number of parameters (number of features used plus one) and insignificant coefficients (t-statistics equal to 0.99). They perform ideal prediction on the training set (100%) but different prediction on the testing samples, from very good (90%) to very bad (60%). Let's follow an example.

Example: We have a sample model for the data *colon.tumor*. The model deviance suggests reaching "the ideal" fit, while the coefficients have over 100% standard errors. This model perfectly fits the training data (100%) and has good accuracy on the testing set (82 %). $\mathcal{AIC}$ of the model is equal to the doubled number of coefficients.

```
Call:
glm(formula = y ~ g493 + g661 + g992 + g1058 + g1346 + g1671,
   family = binomial, data = colon.tumor.train.filter3)

Deviance Residuals:
      Min          1Q      Median          3Q         Max
-5.362e-05  -2.107e-08  -2.107e-08   2.107e-08   5.233e-05

Coefficients:
             Estimate Std. Error   z value Pr(>|z|)
(Intercept) -1.216e+01  4.345e+04 -0.000280    1.000
g493         2.524e-01  2.736e+02     0.001    0.999
g661        -5.234e-02  3.208e+01    -0.002    0.999
g992         2.564e-02  1.130e+02  0.000227    1.000
g1058        1.058e-01  6.522e+02  0.000162    1.000
g1346       -1.111e+00  9.745e+02    -0.001    0.999
g1671        3.851e-02  1.455e+02  0.000265    1.000

(Dispersion parameter for binomial family taken to be 1)

   Null deviance: 5.8574e+01  on 44  degrees of freedom
Residual deviance: 7.9700e-09  on 38  degrees of freedom
AIC: 14

Number of Fisher Scoring iterations: 25
```

Classification tables according to prediction towards the glm model

- on the training sample

y \ predict	0	1
0	29	0
1	0	16

- on the testing sample

y \ predict	0	1
0	9	2
1	1	5

When we apply backward $\mathcal{AIC}$ elimination to this model we receive better model in a sense of $\mathcal{AIC}$ ($\mathcal{AIC} = 2 \cdot 4 = 8$), but also no coefficient is significant!

```
Call:
glm(formula = y ~ g493 + g661 + g1346, family = binomial, data = colon.tumor.train.filter3)

Deviance Residuals:
       Min          1Q      Median          3Q         Max
-8.035e-05  -2.107e-08  -2.107e-08   2.107e-08   7.433e-05

Coefficients:
             Estimate Std. Error z value Pr(>|z|)
(Intercept)  2.326e+01  2.156e+04   0.001    0.999
g493         4.307e-01  1.411e+02   0.003    0.998
g661        -9.401e-02  3.156e+01  -0.003    0.998
g1346       -1.605e+00  5.209e+02  -0.003    0.998

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 5.8574e+01  on 44  degrees of freedom
Residual deviance: 1.8634e-08  on 41  degrees of freedom
AIC: 8

Number of Fisher Scoring iterations: 25
```

Classification tables according to prediction towards the glm model

- on the training sample

y \ predict	0	1
0	29	0
1	0	16

- on the testing sample

y \ predict	0	1
0	9	2
1	1	5

Classification tables according to prediction towards lda (linear discriminant analysis)

- on the training sample

y \ predict	0	1
0	29	0
1	1	15

- on the testing sample

y \ predict	0	1
0	10	1
1	1	5

□

We obtained the model with the best predictive accuracy but useless for prediction (in a sense of significance)! Then, how can we explain this phenomenon? An explanation has been introduced in [2]. The conclusion is that information-theory approach and frequentist approach may reach different conclusions. Both criteria are not independent, as it's clearly described in [2] page 27-28. In some situations the criteria lead to opposite conclusions:

- the value of F-statistics testing assessment of two models (model full versus model reduced) can be low, what means that model reduced is superior to model full,
- while the $\mathcal{AIC}_{full} - \mathcal{AIC}_{reduced}$ is below zero, what - in the other hand - indicates that model full is a superior model.

But what to do then? In whole process of $\mathcal{AIC}$ selection we put attention only on $\mathcal{AIC}$ and to correlations but with respect to $\mathcal{AIC}$ - we never put attention on the significance! But even if we wanted to include significance into computations - how to do that in such large datasets?

The conclusion can be that following some criterion may lead to the results appropriate in terms of this criterion, but unacceptable according to the other conditions. Of course it's possible to reach good results (AIC low and coefficients significant), however it's more likely when the data dimension is small.

Notice 9.0.1

> *"The best" in a sense of $\mathcal{AIC}$ doesn't mean "the best" in a sense of F-test (accepting the model as a statistically good model for prediction).*

Probably the experimental selection favours such situation. In this step we choose the best features according to $\mathcal{AIC}$ helpfullness. As a result we obtain very good features in a sense of $\mathcal{AIC}$, but very poor in a sense of constructing significant model. The features "help" each other in "such a good way", that they finally "work out" very low $\mathcal{AIC}$ (maybe the closest to the lowest possible). However the process of selecting the most helpful features is consistent with our assumptions: "selecting features according to the lowest $\mathcal{AIC}$ of the model" or "the lowest AIC, the better".

The idea how to correct the results can be removing the features which lower the model deviance to zero. Step procedures implemented in R reach end when obtain the less possible $\mathcal{AIC}$, but they don't pay attention on the model deviance! If the deviance is too small (around zero) it's almost sure that the final model has insignificant coefficients while having very low $\mathcal{AIC}$. The correction to step procedure for a single model is described as an algorithm 9.0.1.

Algorithm 9.0.1

Input: a proposed feature set $F = \{f_1, f_2, \cdots f_p\}$*, data set* S*.*
Output: a glm model developed on F*.*

Repeat until $\mathcal{AIC}$ *of the developed model* M *is greater than doubled number of model coefficients:*

1. *Develop a glm model* M *according to the R step procedure using initially* F *feature set and obtaining* F_M *feature set. The model* M *would use some* p' *features* ($p' \leq p$) *from* F_M *feature set* ($F_M = \{f_1, f_2, \ldots, f_{p'}\}$).
2. *Check the stop condition: if* ($\mathcal{AIC}_M > 2 \cdot p'$) *return* M *as the final model.*
3. *Perform in a* p'*-loop iteratively by* i*:*
   - *Develop a glm model* M_i *using* $F_{p'} \setminus \{f_i\}$.
   - *Evaluate the deviance of* M_i.
4. *Select a model* $\tilde{M}$ *from the model family* $(M_i)_{i=1,2,\ldots,p}$ *with the smallest deviance. The feature removed from the model* M *so as to obtain* $\tilde{M}$ *would be* $\tilde{f}$.
5. *Actualize* F *by leaving out* $\tilde{f}$ ($F = F \setminus \tilde{f}$). *The size of* F *has been reduced from* p *to* $p - 1$.

Due to the step procedure correction we manage to avoid the mentioned problem ($\mathcal{AIC}$ small but coefficients errors exaggerating the coefficients by over 100%). We obtained models that present better significance, however still some of them have p-values of coefficient tests greater that for example 0.1 or 0.2. Of course assuming a "border p-value" for significance tests is always a matter of some arbitrage, that is what the statistical analyst should do: take into account many criteria and indicators and try to select the optimal solution.

The other solution to the problem of insignificant results could be to involve significance in more early stage than in external/internal bootstrapping, however this idea has been postponed.

Chapter 10

Conclusions

In the author opinion the project ended with success. Almost whole selection path has been developed in accordance with the initial assumptions. Previously there was no intention to develop the experimental selection part, instead of which some famous filter methods had been considered. After several tests and experiments author decided to introduce the experimental approach, based on the $\mathcal{AIC}$ *improvement*, to feature selection before the real wrapper modeling. This part, mixing together filter and wrapper approach, was a key stage in the project.

The project lasted only four months, what was enough for research, implementation and testing the algorithms, performing few model selection paths and make some final conclusions. However it would be profitable if we had have more time for computations. Unfortunately large data sets demand much time to be thoroughly investigated and, of course, patient and tenacious analyst.

As a conclusion we can say that following $\mathcal{AIC}$ in cooperation with the bootstrap in feature selection gives good, even very good, results. We are able to reach best models with less than ten variables, having previously thousands of features. The obtained results are comparable with the results of some scientists (the experimental part chapter). The models perform good predictive accuracy on both training and testing set.

Of course there always must occur some problems - and we didn't avoid them. The main of them are:

1. data division into training and testing sets - ideal separability towards some gene in the training sample,

2. $\mathcal{AIC}$ approach versus frequentist approach - models with very low $\mathcal{AIC}$ have insignificant coefficients, what can be interpreted as a lack of influence of a certain feature, however it's probably a result of **variables masking**,

3. computational complexity - since some algorithms have exponential complexity,

All problems and proposed solutions have been described in detail in the previous chapter.

Despite we pointed some best models the final model selection always depends on the analyst. He should pay attention to several aspects of data analysis, like the significance, the accuracy, misclassification errors, finally the meaning of the variables. That gives the analyst a great deal of freedom.

In the project we paid attention to the practical application of the investigated methodology. We tried to point the cases where $\mathcal{AIC}$ approach and whole developed path works well and why. Even if we encountered some problems, we tried to investigate the reason

and propose the solution. After the project finished we know how important is to carefully prepare the data and what could be the consequence of giving it up. The general advice to select a proper methodology is always to weigh all the pros and cons. The real success in data mining issue is a matter of thoroughly performed analysis and proper methodology selection.

Chapter 11

Timetable and the costs

11.1 The costs of the project

The total expense of the project reduces to the costs of implementation. One hour of work of a programmer or a statistics analyst has been set to 30 euro. It turned out that we needed some additional hours, which corresponded to two weekends (5 hours per one day). One extra hour cost 50 euro. The implementation work has not been divided into the programmer part and the statistics analyst part due to the experimental nature of the project. It was preferable that both these skills were held by one person. The costs of the project are presented in the table 11.1.

used tool			amount	the costs (euro)
1	Operating system	Arch Linux	1	0
2	Programming environment	R	1	0
3	Programming editor	vi	1	0
4	Professional document preparation system	latex	1	0
5	Professional document preparation editor	kile	1	0
6	Working costs (hours)	programmer, statistics analyst (16 weeks, 5 days per week, 7 hours per day)	560	16 800 (30 per hour)
7	Extra hours	programmer, statistics analyst	20	1000 (50 per hour)
Total cost				17 800

Table 11.1: The costs of the project

11.2 Timetable

The project started in the second part of February, however the first steps of implementation were made in the half of March.

According to the project timetable we can distinguish ten huge steps:

1. Formulating the objectives
2. Literature research
3. Constructing the project schema
4. Data preparation
   (a) Reading microarray data
   (b) Exploring the data
5. Filter methods
   (a) Filter methods research
   (b) Filter methods experiments
   (c) Filter methods testing
6. Wrapper Methods
   (a) Wrapper methods construction
   (b) Wrapper methods first implementation
   (c) Wrapper methods testing and verification
   (d) Wrapper methods final implementation
7. Experimental methods
   (a) Experimental methods research
   (b) Experimental methods experiments
   (c) Experimental methods implementation
   (d) Experimental methods testing
8. Optimization
   (a) Optimizing the algorithms
   (b) Optimizing the code
9. Processing the whole path on a real data
10. Documentation
    (a) Writing the document
    (b) Writing "help"

The gantt chart of the project is shown on fig 11.1.

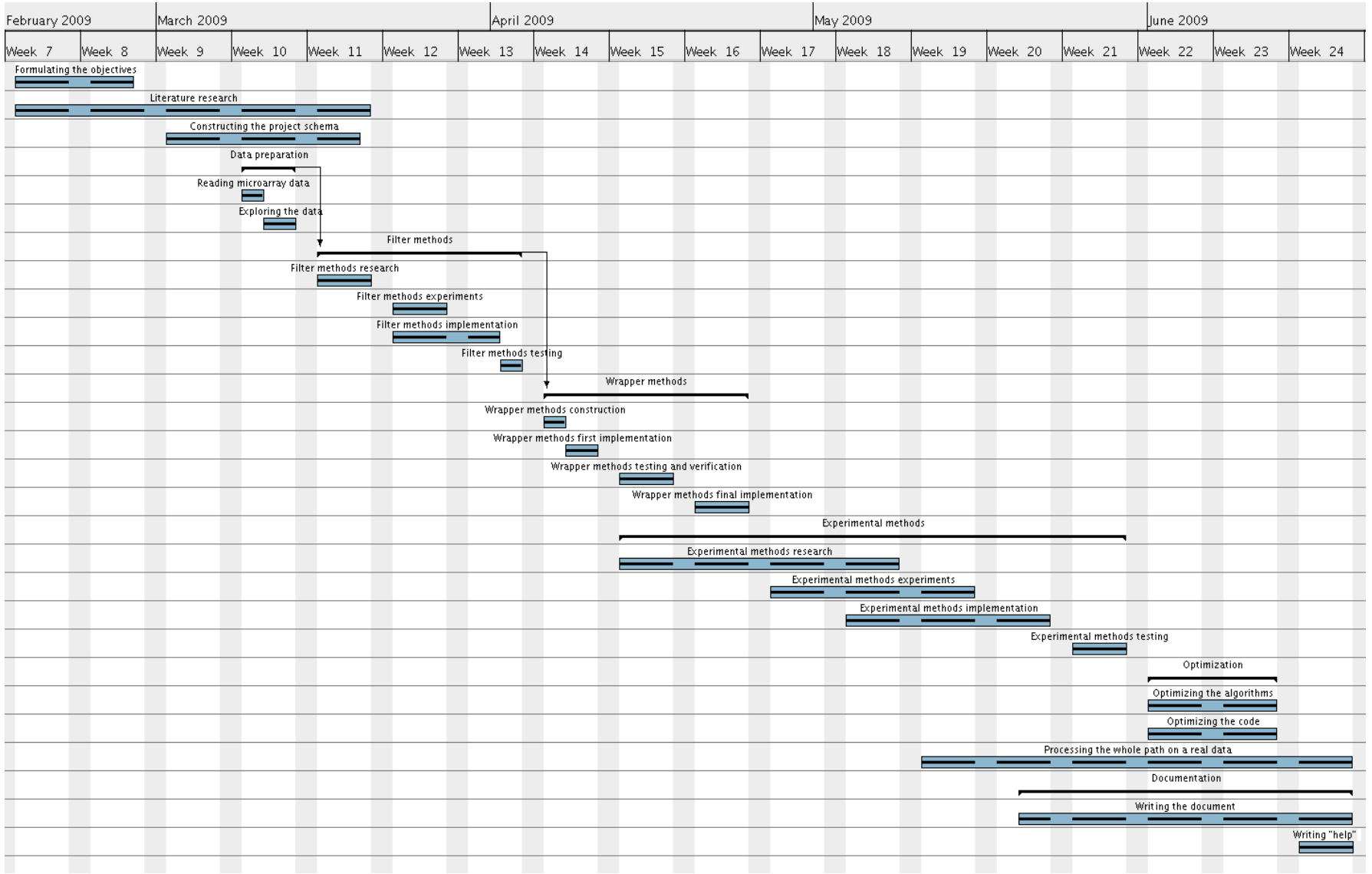


Figure 11.1: The timetable

Bibliography

[1] Burnham Kenneth P., Anderson David R. (1998), *Model Selection and Inference. A practical Information-Theoretic Approach*, Springer-Verlag, New York

[2] Bonate P. L. (2005), *Pharmacokinetic-pharmacodynamic modeling and simulation*, Springer-Verlag

[3] Bu, H.-L., Li, G.-Z., and Zeng, X.-Q. (2007), *Reducing error of tumor classification by using dimension reduction with feature selection*, in The First International Symposium on Optimization and Systems Biology, pp. 232–241.

[4] Cover T. M, Thomas J.A (2001) *Elements of Information Theory*, Hardcover

[5] Felix F. Gonzalez N., Belanche L. A. (2009) *Gene Subset Selection in Microarray Data using Entropic Filtering for Cancer Classification*, Expert Systems, Special Issue on Advances in Medical Decision Support Systems. Volume 26, Issue 1, Pages 113 - 124. 2009.

[6] Lukacs P. M., Burnham K. P., Anderson D. R (2009), *MOdel Selection Bias and Freedman's Paradox*, Springer.

[7] Guyon I., Gunn S., Nikravesh M., Zadeh L., Editors. (2006),*Feature Extraction, Foundations and Applications*, Series Studies in Fuzziness and Soft Computing, Physica-Verlag, Springer, pp. 89-118.

[8] Huan Liu, Motoda, Hiroshi (1998), *Feature Selection for Knowledge Discovery and Data Mining*, The Springer International Series in Engineering and Computer Science, Vol. 454, Springer.

[10] Hastie T., Tibshirani R., Friedman J,. (2002) *The Elements of Statistical LEarning*, Springer-Verlag

[11] Hong, J.-H. Cho, S.-B. (2008), *Cancer classification with incremental gene selection based on dna microarray data*, in IEEE Symposium on Computational Intelligence in Bioinformatics and Computational Biology, pp. 70–74.

[12] Koronacki J., Mielniczuk J. (2001), *Statystyka dla studentów kierunków technicznych i przyrodniczych*, Wydawnictwo Naukowo-Techniczne, Warszawa.

[13] Koronacki J., Ćwik J. (2005), *Statystyczne systemy uczace sie*, Wydawnictwo Naukowo-Techniczne, Warszawa.

[14] Wang, L., Zhu, J., Zou, H. (2008), *Hybrid huberized support vector machines for microarray classification and gene selection*, Bioinformatics, 24(3), 412–419.

[15] Liu, H. Motoda, H. (1998), *Feature extraction, construction and selection. A data mining perspective*, Kluwer Academic Publishers.

[16] , Ruiz, R., Riquelme, J., Aguilar, J. (2006), *Incremental wrapper-based gene selection from microarray data for cancer classification*, Pattern Recognition, 39, 2383–2392.

[17] Cai, R., Hao, Z., Yang, X., Wen, W. (2009), *An efficient gene selection algorithm based on mutial information*, Neurocomputing, 72, 991–999.

[18] Hewett, R. Kijsanayothin, F. (2008), *Tumor classification ranking from microarray data*, BMC Genomics, 9(2).