KU LEUVEN

FACULTEIT INGENIEURSWETENSCHAPPEN

Design, implementation and evaluation of data integration methods for biomedical cancer data

Michiel Ruelens

Thesis voorgedragen tot het behalen van de graad van Master of Science in de ingenieurswetenschappen: computerwetenschappen, hoofdspecialisatie Mens-machine communicatie

Promotoren:

Prof. dr. ir. Roel Wuyts
Prof. dr. ing. Olivier Gevaert

Assessoren:

Mario Cruz Torres
Prof. dr. ir. H. Blockeel

Begeleiders:

Ir. A. Assistent
D. Vriend

Academiejaar 2015 – 2016

Preface

I would like to thank everybody who kept me busy the last year, especially my promotor and my assistants. I would also like to thank the jury for reading the text. My sincere gratitude also goes to my wive and the rest of my family.

Michiel Ruelens

Contents

Abstract

Samenvatting

In dit `abstract` environment wordt een al dan niet uitgebreide Nederlandse samenvatting van het werk gegeven. Wanneer de tekst voor een Nederlandstalige master in het Engels wordt geschreven, wordt hier normaal een uitgebreide samenvatting verwacht, bijvoorbeeld een tiental bladzijden.

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Ut purus elit, vestibulum ut, placerat ac, adipiscing vitae, felis. Curabitur dictum gravida mauris. Nam arcu libero, nonummy eget, consectetuer id, vulputate a, magna. Donec vehicula augue eu neque. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Mauris ut leo. Cras viverra metus rhoncus sem. Nulla et lectus vestibulum urna fringilla ultrices. Phasellus eu tellus sit amet tortor gravida placerat. Integer sapien est, iaculis in, pretium quis, viverra ac, nunc. Praesent eget sem vel leo ultrices bibendum. Aenean faucibus. Morbi dolor nulla, malesuada eu, pulvinar at, mollis ac, nulla. Curabitur auctor semper nulla. Donec varius orci eget risus. Duis nibh mi, congue eu, accumsan eleifend, sagittis quis, diam. Duis eget orci sit amet orci dignissim rutrum.

List of Figures and Tables

List of Figures

List of Tables

Chapter 1

Introduction

1.1 The need for data integration methods

Current advancements in technology are leading to increasingly larger datasets[21]. This is a trend that is becoming very apparent in many different fields of research. One of these is the biomedical field. By larger datasets we mean that they are increasing both in the number of variables, aswell as in the number of samples that are available. The increase in variable count is mainly due to technology. We are reaching the point where we are able to sequence anyones DNA at a very low cost[37]. Keeping in mind that humans have around 20.000-25.000 proteine-coding genes, it is very common for current genome-datasets to have thousands of variables. Another example is that we have a range of highly advanced imaging instruments in almost every hospital[20][30]. These instruments provide high quality images from which we can extract even more feature variables to analyze. These are just two examples that show the growth of variable count. Next to the explosion of the number of variables, the number of samples that are available in databases grows aswell. This is because we are able to simply store much more data now than we could in the past, and on top of that, there are big efforts going on to make datasets open to the public to help researchers around the world work together[40][36][31][39]. All of these trends lead to huge amounts of data from different sources becoming available for analysis. This raises the question of how we have to combine (or integrate) data from these different sources to get the most information out of them.

1.2 Goals and modus operandi

The aim of this thesis is to develop several integration strategies and show that these strategies have an impact on the performance of predictive models. In order to do this we first must understand what a predictive model means and how we can compute one. The first chapter explains the first type of predictive models which are called generalized linear models. More specifically this thesis will focus on the logistic regression model. The next chapter explains the second type of predictive models which are called survival models. In this case we will take a closer

look at cox proportional hazards models. Once we have established the notion of predictive modelling, we can define the different integration strategies. This is done in chapter three. In order to show that these integration strategies have an effect on the performance of the predictive models we will apply them to a real world example in two case studies, this is the content of chapter four. The first case study will use logistic regression to build predictive models using the various integration strategies, and the second case study will do the same for survival models. Once we have constructed all these models we will compare them using appropriate metrics.

In order to facilitate all of the previous work we designed an interactive application that streamlines this whole process. The application is capable of comparing the performance of predictive models using the various integration strategies that will be presented in chapter three. Not only did this allow for the computation of the two case studies, but it also provides a platform for future research.

Chapter 2

Generalized Linear Models

2.1 Introduction

In this chapter we will explain the concepts of generalized linear models[18][2]. This term indicates a generalization of simple linear regression that allows for a wide range of output variables. First we will go over the basics of linear models, gradually building up to the definition of generalized linear models. Next, we will describe what actual data looks like and how a model is computed from it. After that we will tackle the more recent innovation of regularization that will greatly improve our previous models by exploiting the bias-variance trade-off to reduce overfitting. Lastly we will outline the validation method that will be used to test the performance of the models.

2.2 Classical linear models

When we think of classical linear models, we can imagine a set of numeric explanatory (or input) variables and a numerical dependent (or output) variable. By making a linear combination of the explanatory variables we attempt to estimate a value for the dependent variable. Depending on the type of dependent variable the linear method gets a different name. In the following sections we will outline several of them.

2.2.1 Linear Regression

The simplest version of a linear model is called linear regression[18][8]. In this case the input variables are combined using a linear combination, and the result of this calculation is immediately used as the final estimate. Imagine we have a dataset of size N where each sample has M explanatory variables. Then we can write linear regression as:

$$y_n = \sum_{i=1}^{M} w_i x_{in} = \boldsymbol{w}^{\boldsymbol{T}} \boldsymbol{x} \qquad for\ n = 1..N \tag{2.1}$$

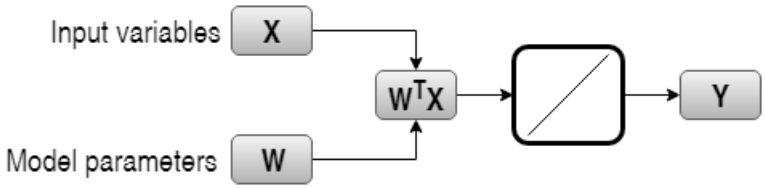


Figure 2.1: Schema for linear regression

where

- y_n is the output for sample n
- w_i is the model parameter (coefficient) for explanatory variable i
- x_{in} is the value for explanatory variable i for sample n
- $\boldsymbol{w^T x}$ is the vector notation for the inner product of $\boldsymbol{w}$ and $\boldsymbol{x}$

For the other linear methods we will define a function each time that is applied to the result of the linear combination. We could do the same for linear regression and say that the applied function is the identity function. A schema for this computation is shown on figure 2.1.

2.2.2 Linear Classification

The next method is called linear classification[18][7]. The difference with linear regression is that we have a different type of output variable. In a classification task we want to predict a class from a list of potential classes. For instance, we could try to predict whether tomorrow will be a sunny day or not. Notice that there are only 2 possible outcomes: 'sunny' or 'not sunny' and we could represent these outcomes as 0 and 1 in our model. This form would be called binary classification because we have 2 possible classes. It is very easy to extend this method to multi-class classification. The computation in this method starts out exactly the same, combining the input variables using a linear combination. Next, we have to define a threshold to indicate which examples belong to one class or another. In the case of binary classification we would define 1 threshold, and if the result of the linear combination is higher than the threshold we would predict one class. If it is lower, we would predict the other class. The function used here would be called a sign function, which maps real values onto one of 2 possible outcomes. We could represent this computation with the following formula:

$$y_n = \begin{cases} 0 & if\ \boldsymbol{w^T x} \leq t \\ 1 & if\ \boldsymbol{w^T x} > t \end{cases} \qquad for\ n = 1..N \tag{2.2}$$

where

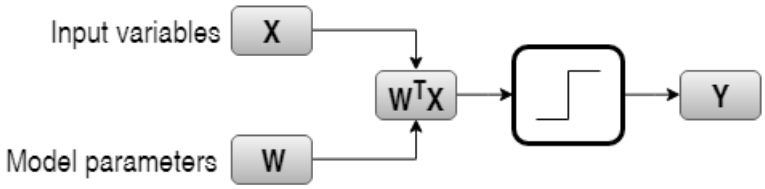


Figure 2.2: Schema for linear classification

- y_n is the output for sample n
- $\boldsymbol{w^T x}$ is the vector notation for the inner product of $\boldsymbol{w}$ and $\boldsymbol{x}$
- t is the classification threshold

A schema for this computation is shown on figure 2.2.

2.2.3 Logistic Regression

The third method we want to present is called logistic regression[18][9]. In this case, the output variable we want to predict comes from a binomial distribution. This means that they are the result of a probabilistic event. An example would be tossing a coin and checking whether the result is heads or tails. While the outcome is binary (heads or tails) we know that there is an underlying probability for the coin to be heads or tails, and we would like to know this probability.
The idea is still the same. We will make a linear combination of the input variables. However this time we will use a logistic function to produce our estimate. The logistic function is a function that maps real numbers onto the range $[0, 1]$. This result can then be interpreted as an estimate for the probability. We could represent logistic regression mathematically as:

$$y_n = \theta(\sum_{i=1}^{M} w_i x_{in}) = \theta(\boldsymbol{w^T x}) \qquad for\ n = 1..N \tag{2.3}$$

where

- y_n is the output for sample n
- w_i is the model parameter (coefficient) for explanatory variable i
- x_{in} is the value for explanatory variable i for sample n
- $\boldsymbol{w^T x}$ is the vector notation for the inner product of $\boldsymbol{w}$ and $\boldsymbol{x}$
- $\theta(x)$ is a logistic function, also called a sigmoid function. An example sigmoid function is $\frac{e^x}{1+e^x}$

A schema for this computation is shown on figure 2.3. The logistic regression method is the one that will be most widely used throughout this thesis.

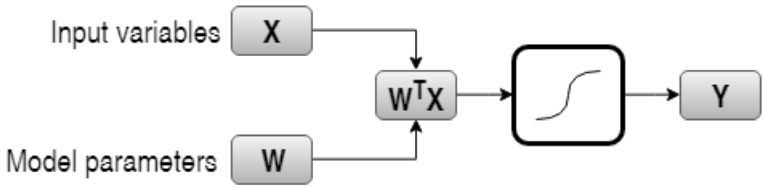


Figure 2.3: Schema for logistic regression

2.3 Training a model

In order to understand the integration strategies that will be explained later on, it is useful to know how exactly the models come to be. This section will explain what the input data for our linear models actually looks like, and how we get from this data to a model that we can use for future predictions.

2.3.1 The data

The data we use consists of two parts: the input data, which can be seen as a matrix where the columns are the explanatory variables and each row is an example. And secondly the output data, which can be seen as a vector where each value indicates the value of the dependent variable for a single example.
It is easy to see that the length of the output vector has to be equal to the amount of rows in the input matrix, indeed there should be one output value for each example. This amount is often called the size of the dataset and we would like it to be as big as possible. Especially when we are dealing with a large number of explanatory variables, it is essential to have a reasonable amount of examples aswell. This will be discussed in more detail in section 2.4 on overfitting.

2.3.2 Gradient descent

In this section we will explain how we get from the input data to the model[18][4][5]. The idea here is that we have some error measure. The error measure is a sort of rating for our current model as it indicates how big the mistakes are that our current model makes. Once we have a way of computing this error, we can try to minimize the error to obtain our 'best' possible model.

Error measure

In logistic regression the error measure we use is called the cross-entropy error[18]. The formula for this error is the following:

$$E_{in}(w) = \frac{1}{N}\sum_{n=1}^{N} ln(1 + e^{-y_n w^T x_n})$$

where

- x_n is the vector of values for the explanatory variables for example n.
- y_n is the value of the dependent variable for example n.
- w^T is the transpose of the weights vector. These are the parameters of our model that we can adjust.
- N is the size of our dataset.
- $E_{in}(w)$ is the in-sample error. This is the cross-entropy error that we make on the examples in our dataset. It is a function of the weights w.

We can intuitively see that this is a reasonable error measure. It is an averaged sum over all examples, where for each example we compute an individual error made on that example. Notice that $w^T x_n$ is the linear combination of the input variables that our current model suggests. This is the prediction that our current model would make for example n and is a real valued number. On the other hand y_n is the actual correct prediction for example n and has a value of 0 or 1. If the signs of $w^T x_n$ and y_n agree then our current model actually makes a correct prediction for this example. We can see that in this case the exponential becomes close to 0, making our error for example n very small, as we would expect. If however their signs are opposite, the exponential becomes larger as our incorrect prediction becomes larger. This in turn will increase the error, again as we would expect. Thus we can see that if we were to minimize this error, we are moving towards a model that tries to make correct predictions.

The gradient descent method

When trying to minimize a function, a general approach would be to try and compute the derivative of the function, and find the value where this derivative equals zero. In the case of linear regression it is actually possible to compute this minimum in one step. More details about this can be found in appendix //TODO ADD AND REFERENCE APPENDIX.
In the case of logistic regression however it is not possible to find an analytic solution to this problem. The best we can do is put ourselves somewhere on the error surface and try to move towards the minimum in small steps. This is called an iterative approach. Remember that our error function looks like this:

$$E_{in}(w) = \frac{1}{N} \sum_{n=1}^{N} ln(1 + e^{-y_n w^T x_n})$$

We can now compute its derivative with respect to w:

$$\nabla E_{in}(w) = -\frac{1}{N} \sum_{n=1}^{N} \frac{y_n x_n}{1 + e^{y_n w^T x_n}}$$

The problem is to find the set of weights w for which the derivative becomes 0 (or that minimizes the error). We can start out with an initial set of weights $w(0)$ and

then iteratively update these weights so we move towards the minimum. Let's call the direction in which we update our weights v. The update we make to w then becomes:

$$w(t+1) = w(t) + \eta v$$

where

- $w(t+1)$ are the updated weights for this iteration.
- $w(t)$ are the current weights before we make a move.
- v is a unit vector pointing in the direction we want to move.
- η is a number that indicates how big the move is that we make, also called the step size.

Remember that the gradient of a function at a certain point always points towards the steepest slope upwards[3][16]. In our case we would like to find the minimum, so it is a good idea to move our weights in the direction of steepest descent. The direction v that we are moving towards then becomes the normalized opposite direction of the gradient:

$$v = -\frac{\nabla E_{in}(w(t))}{\|\nabla E_{in}(w(t))\|}$$

We can now summarize the gradient descent method as follows:

Data: x, y
initialize weights w(0)
while *Stopcondition is not met* **do**
  Compute gradient $\nabla E_{in}(w(t))$
  Compute update direction v
  Update weights $w(t+1) = w(t) + \eta v$
end

Algorithm 1: Gradient Descent algorithm

There are two non-trivial issues in this computation: the initialization of the weights and the stopcondition.

Weight initialization is sometimes a very tricky thing to do, in the case of logistic regression however it is acceptable to set $w(0)$ equal to the zero-vector as this corresponds to no correlation between any of the input variables and the output variable, and the result of the sigmoid function would be 0.5 or 50% meaning the model has no preference for either outcome.

The stopcondition however is a bigger issue and usually the way to go here is to make a combination of several stop criteria. One criteria would be to simply limit the amount of iterations to a fixed number. This could avoid endlessly overfitting. Another criteria is to set up a target error we want to achieve (a small number), and stop when we have reached this target. This however raises the question of picking

the target error, and this is mostly an application dependent choice.

In the version of logistic regression explained here, it can however be shown that the error surface we are dealing with is a convex surface[10]. This makes it very easy to find its minimum and we don't need very complex initialization and stopping criteria to get good results. In other machine learning methods however these surfaces aren't always as nice, and the issue of local minima versus global minima becomes a big deal. There has been much research on this topic however and many sophisticated methods have been developped to deal with this issue.

2.4 Overfitting

Now that we have established a method of computing our models, it is time to deal with an issue known as overfitting[18][19]. Overfitting points to the fact that there are several mechanisms at work when we are building a model that prevent us from reaching the perfect model (a model that predicts correctly at all times). These mechanisms essentially originate from noise and uncertainty in many aspects of the learning process (the input data, choice of model, choice of algorithm, ...). We can however try to decompose this noise into several components and then attempt to influence them by making changes to our model computation. We will present two ways in which overfitting can be tackled: regularization and validation.

2.4.1 The problem of overfitting

Let's introduce some notation. From now on we will refer to the notion of 'in-sample error' or in symbolic notation E_{in} as the error that a model makes on the examples in our training set. The training set consists of the examples that were used to train (compute) the model in the first place.
Similarly we will define 'out-of-sample error' or E_{out} as the error we make on examples that were not used for training the model. Notice that E_{in} is something we could compute because we have access to the training data, but E_{out} is a quantity we cannot exactly compute but we could try to estimate it if we have some examples left that we did not use for training. Notice also that it is E_{in} that we minimize during our model computation, but it is E_{out} that we actually want to minimize! Indeed, E_{out} corresponds to the error that we get when we are going to deploy our model in practice and use it on examples we have never seen before. We can do this because we believe that E_{in} tracks E_{out} to a certain degree. And thus if we manage to minimize E_{in} we also minimize E_{out} to some extent.
We can only speak of overfitting when we are comparing two models. We say that one model, call it model A, is overfitting with respect to another model, model B, when model A managed to get a lower E_{in} than model B, but model B has a lower E_{out}.
Another way of looking at it is during the learning process. Let's have model A be the model that we computed when we started from model B and performed one more iteration of the training algorithm. Thus model A is 'more trained' than model

B. Now let's suppose model A is overfitting:

$$E_{in}^{modelA} < E_{in}^{modelB} \tag{2.4}$$

$$E_{out}^{modelA} > E_{out}^{modelB} \tag{2.5}$$

The additional iteration has decreased the in-sample error, and thus we are able to fit our training data better, but the out-of-sample error has increased, meaning that our model doesn't generalize as well to other examples outside the training set. This means that we are actually fitting our training data too well, while we are not really getting a better grasp of the underlying pattern that we wish to learn. We are overfitting the training data.

2.4.2 The bias and variance trade-off

There are several ways of looking at overfitting and pointing out its origins. We will introduce the notions of bias and variance and how they can describe the noise in our system.

Average hypothesis

First, let me explain the playing field. We are in a situation of learning, where we are given a set of examples that are produced by some target function f. It is this target function f that we wish to learn (or model). In order to do this we have to decide on the type of functions that we will use to model. In machine learning this is called the hypothesis set. It is the set of all functions that we consider possible candidates to fit our target f. Training a model then boils down to using the examples x in our dataset D to decide which hypothesis to pick.
Next, let's introduce the notion of average hypothesis $\bar{h}$. Imagine we have a very large number of datasets. For each of these datasets we apply the learning process and we will pick a certain hypothesis h from our hypothesis set. The average hypothesis is then equal to the average of all the hypothesis' just learned. Or in a formula:

$$\bar{h} = \mathbf{E}_D[h^{(D)}]$$

where

- $\bar{h}$ is the average hypothesis.
- $\mathbf{E}_D$ is the expected value over an infinite number of datasets
- $h^{(D)}$ is the hypothesis that was learned for a specific dataset D

We can also look at this average hypothesis as sort of the best we can do with the given hypothesis set. Indeed, when we imagine having an infinite number of datasets we would end up cancelling out much of the variation in the learned hypothesis' and end up with a very good one.

Bias

We can now define the bias[18] as the distance between the average hypothesis $\bar{h}$ and our target function f.

$$bias = (\bar{h} - f)^2$$

We can see the bias as an error we make due to our own choices. Namely our choice of hypothesis set. If we choose a very simple hypothesis set, we cannot expect to be able to find a fit for a very complex function. The target function simply isn't contained in our hypothesis set.
Therefore we introduced the notion of average hypothesis. We can view this as the best we can do given our current hypothesis set, and the distance to the target is what we call the bias.

Variance

In a real learning situation we generally never find this average hypothesis, because it requires a large amount (or even infinite amount) of datasets. We never have this luxury! In reality we always have only one dataset and that is all we can use to navigate through the hypothesis set. This is where the notion of variance comes in. We can define variance[18] as the error we get from not having the best hypothesis possible in our hypothesis set. Or in other words as the distance between the hypothesis that we actually found by learning from our dataset and the average hypothesis.

$$variance = (h^{(D)} - \bar{h})^2$$

Error due to variance mainly comes from two sources: the first is our finite dataset. In reality we are given a dataset of N examples and usually this dataset is not sufficient to find the best hypothesis and thus there will be a variance error made. The second is the complexity of the hypothesis set. As we increase the complexity, it becomes increasingly difficult to navigate through this set. There are simply many more hypothesis' to choose from. Again this makes it harder for our learning process to find the optimal hypothesis and as such will introduce a variance error.

The tradeoff

Having both bias and variance defined we can see that they are not disconnected, there is a tradeoff. If we look purely at bias we could think that simply choosing a super complex hypothesis set is always optimal. Indeed our bias will be zero since the target function will always be inside our hypothesis set.
However, an increasingly complex hypothesis set makes it harder to actually find the optimal hypothesis. We know that the optimal hypothesis is there, but we just cannot find it. The take-away message here is that we have to choose a hypothesis set complexity based on the resources that we have. In this case the resource is our dataset. The larger the dataset that we can learn from, the more complex hypothesis sets we can afford, and the better our results will be. But there is no gain in choosing overly complex hypothesis sets when you don't have the resources to afford them.

This will simply cause you to find hypothesis' that fit your training data very well (imagine fitting 3 datapoints with a 7th order polynomial, you would get an exact fit) but this model will not generalize to anything in the real world, it is a complete overfit.

2.5 Regularization

Now that we have the concepts of bias and variance, let's use this information to try and improve our models. The first method is called regularization[18][42][25]. In very simple terms this method will add a small amount of bias in order to greatly decrease the amount of variance, reducing the overall error we make.

2.5.1 Adding bias

Remember that bias is defined as the distance between the average (or best) hypothesis $\bar{h}$ and our target function f. Adding bias effectively means we are going to make another choice, which will impact the average hypothesis. The choice we are about to introduce is based on the following observation: when confronted with a set of similarly performing models, the simplest model is usually the best. Or in other words we should try to prefer simple models over very complex ones.

This observation does not have a mathematical proof, it is rather an observation from experience and reason. One good argument is the fact that noise is usually of high frequency. Meaning that distortions of our dataset (for instance measurement errors) will often be very scattered and random, while the underlying pattern that really makes up the data will be rather smooth. A similar argument can be made for the error due to bias, when we choose a hypothesis set that does not contain the target function, the error due to bias will be mostly random and of high frequency. Therefore if we want to reduce the impact of this noise in our final model, we should prefer models that are not able to fit these high frequencies exactly. Lastly we can remark that if we look at our current understanding of nature (let's say at a larger scale), systems almost always have smooth transitions. The most important laws of nature that we find are all written down in small, simple formulas. Nature doesn't work with instantaneous changes (high frequency), but rather it has smooth functions that govern the basic principle, and then it adds random noise and fluctuations on top of it. This principle is what we try to extrapolate here to machine learning.

Thus, the choice we will make is that we will prefer simple models over complex ones by adding a constraint to the weights.

2.5.2 Regularization types

There are many kinds of constraints that we could add to the weights and, depending on the constraint we choose, the regularization gets a different name and it will have a different effect. One of the most famous regularizers is called ridge or weight-decay[25].

The constraint for this regularizer is the following:

$$\sum_{i=1}^{N} w_i^2 \leq C$$

where

- w_i is the weight (model parameter) for the i'th explanatory variable.
- C is the constraint value

Using this regularizer will result in a preference for models with smaller weights. This keeps certain weights from getting out of control. This form of regularization is also often called the L_2 penalty.

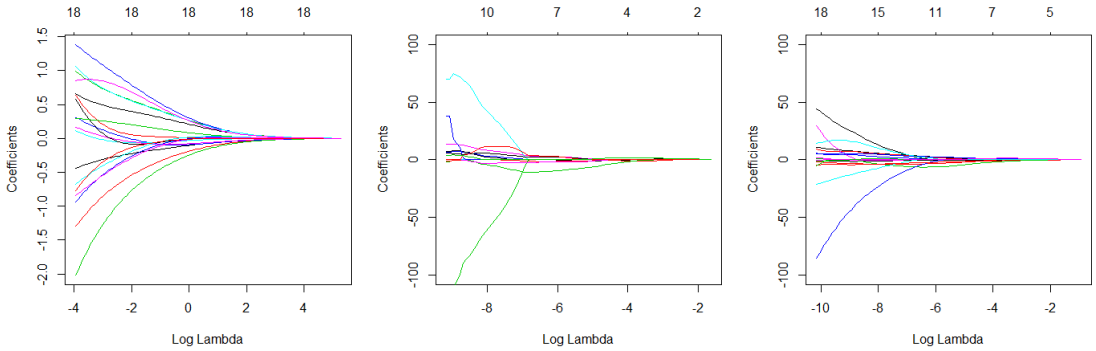
Another popular regularizer is called the lasso penalty (or L_1 penalty)[25]. The constraint in this case is:

$$\sum_{i=1}^{N} |w_i| \leq C$$

In addition to keeping the weights small, this form of regularization also performs parameter selection. This means that instead of just keeping the weights small it will also prefer to make weights actually zero for variables that don't contribute much to the model. This will cause the resulting model to have fewer parameters, but the parameters that do survive the penalty are sure to be very important. This regularizer is often used when there is a huge number of explanatory variables, and we wish to find only those that are really descriptive. Later in the thesis we will use datasets that contain gene expression information about cancer patients, these datasets often have thousands of explanatory variables and will provide a good example for using the lasso regularization (see section 5.5).
The last form of regularizer we wish to demonstrate is called the elastic net penalty. This regularizer is simply a linear combination of the ridge and lasso penalties and provides a way of balancing the two. It has the following constraint:

$$\alpha \sum_{i=1}^{N} w_i^2 + (1-\alpha) \sum_{i=1}^{N} |w_i| \leq C$$

To demonstrate the difference between the types of regularization, it is useful to look at a figure that plots the evolution of the weights as a function of the amount of regularization used. As explained in the next section (2.5.3), the amount of regularization used is called λ. Figure 2.4 shows the coefficient paths respectively for ridge, lasso and elastic net regularization. These are the values of the coefficients as a function of λ. The sample dataset in this case contained 18 explanatory variables. The ridge penalty selects all 18 variables in the model, while the lasso and elastic net penalties respectively select between 10-13 and 11-18 variables in their models, depending on the value of λ. This shows the variable selection of the lasso. Also notice how the y-axis dimension is different for the ridge regularization, showing the shrinkage of the weights.



(a) Ridge ($\alpha = 0$) (b) Lasso ($\alpha = 1$) (c) Elastic net ($\alpha = 0.5$)

Figure 2.4: Coëfficiënt paths as a function of λ for different regularization types.

2.5.3 Lambda

Using a regularizer introduces a constraint, this means that we now have to deal with a constrained optimization problem which is much harder to solve than an unconstrained problem. Fortunately, through some clever mathematics it is possible to convert the constrained minimization problem to an unconstrained one by incorporating the regularization constraint in the formula for the error itself[18]. The formula for the error with regularization then becomes:

$$E_{in}(w) = \frac{1}{N}\sum_{n=1}^{N} e(x_n, y_n, w) + \lambda R(w)$$

where

- $E_{in}(w)$ is the in-sample error.
- $e(x_n, y_n, w)$ is the individual error made on example n. In the case of logistic regression this would be a cross-entropy error term $ln(1 + e^{-y_n w^T x_n})$.
- λ is the regularization parameter that will be explained below.
- $R(w)$ is the regularization term dependent on the type of regularization used. For instance: $\sum_{i=1}^{N} |w_i|$ for lasso, $\sum_{i=1}^{N} w_i^2$ for ridge, ...
- x_n is the n'th sample in the dataset.
- y_n is the outcome for the n'th sample in the dataset.
- w is the vector of weights, our model parameters that we are trying to find.

Notice that we now again have an error measure that we want to minimize, and it is an unconstrained optimization problem. The important new part is λ. This is the amount of regularization we want to use. It is a new form for the constraint constant

C that was earlier introduced in the types of regularization. The higher λ the tighter the constraint (lower C) and vice versa. The value of λ will prove to be critical in getting good models. The way to calculate it is through validation.

2.6 Validation

In this section we will explain the method of validation which is used to estimate the out-of-sample error. We will first explain the issue of sample size and then present a method to work around this limitation.

2.6.1 The sample size dilemma

Remember that when we are training a model we use the in-sample error to navigate the hypothesis space. We do this because we believe that the in-sample error is a valid surrogate for the out-of-sample error (which is the error we actually want to minimize). A valid question to ask is: why don't we just estimate the out-of-sample error and minimize it directly? Consider the following situation:
We are given a dataset of N samples. We will use K samples of the dataset to train my model, this leaves me with $N - K$ samples that are not used for training. Since K samples were used for training, if we would compute the error the model makes on these samples we would be computing the in-sample error. If we want to estimate the out-of-sample error we have to use the $N - K$ samples that were not used for training. This sample set of $N - K$ samples if often called the validation set.
We can now make the following observations:

- The larger we choose K, the more samples are available for training and thus the better our model can be (due to lower variance!)
- The larger we choose K, the less accurate our out-of-sample error estimate is, because we have fewer datapoints for the estimation

So now it is clear that we have a tradeoff to make. We would like K to be as large as possible so that we have a large amount of samples to train a model from. On the other hand we would like K to be as small as possible so that we have enough samples to estimate the out-of-sample error. The solution to this apparent contradiction will be cross-validation.

2.6.2 Cross-validation

Cross-validation[18][32] is a technique that allows us to have plenty of samples left for training, while still getting a pretty good estimate for the out-of-sample error. The method is as follows: divide the dataset of N samples into F equal parts (often called folds). Each fold now has N/F samples. Train a model on $F - 1$ folds and use the remaining fold to estimate the out-of-sample error. Repeat this process F times, one time for each of the F folds, each time leaving out a different fold. In the end we have F estimates of the out-of-sample error and we can average them to get

a final result.

Notice that we are really using all samples for validation and training, but never both at the same time. It feels a bit like cheating, but in practice this method works wonderfully. Validation is often used to determine parameters of the learning process, for instance the λ parameter for regularization. We simply try several values for λ, compute the out-of-sample error using validation, and pick the λ that gives the lowest error. Once we have decided this λ we can then train a model on the full dataset of N points and use the λ we have just calculated to be optimal.

We have to remark however that when we use validation to calculate a value for λ as described above, we are really using the validation to help train the model. In this case we can no longer make the statement that the validation samples are not used for training. This is called data pollution. We are using the same data to train training parameters aswell as training the model itself and it is obvious that this will give rise to additional correlations in the data. However in practice it is generally accepted that if you use this technique to decide on just a few learning parameters (often just λ), and you have a big enough dataset, the data pollution is minimal and the results and estimates you get are still reliable[18].

Cross-validation is not only used to estimate learning parameters. It can also be used to test the performance of the model. In this case the fold that is not used for training is usually not called a validation set, but rather a test set. Because samples in this set will be used to test the models performance.

2.7 Conclusion

We have now covered the basis of Generalized Linear Models. We have described the gradient descent method that we can use to navigate the hypothesis space by minimizing an error function. We have seen that overfitting is a serious issue that is caused by noise at different levels of the learning process. We have broken down this noise into bias and variance and shown that we can have an impact on this process. We can use regularization to add a slight bias in order to greatly decrease the error due to variance and we have based this on the principle that we should prefer simple and smooth models. Lastly we have covered the method of validation. A method that we can use to estimate the out-of-sample error and which gives us the ability to choose learning parameters like λ and also test the performance of our model.

Chapter 3

Cox proportional hazards models

3.1 Introduction

This chapter explains the cox proportional hazards model[38][41][12]. This is a model that is used in survival analysis. The first section explains what survival analysis means and how it is different from the generalized linear models. Next we will define what the proportional hazards assumption is and lastly we will show kaplan-meier survival curves and how we can compute them from hazard functions.

3.2 Survival analysis

Survival analysis points to the fact that the outcome variables are time-to-event datapoints. A common example in biomedical research would be the time between the start of a patients treatment and the time of death. Note however that not all patients have to die in order to be useful in a survival analysis. A second outcome variable is used to indicate whether the patient survived until the end of the study, or not. This (often binary) variable is called the censoring variable.

A typical outcome in survival analysis is thus represented by two vectors: a time to event vector and a censoring vector. An example is shown in table 3.1. The idea of survival analysis is now to find a pattern between a set of explanatory variables and the time-to-event. A common term used in survival analysis is hazard, meaning 'risk of the event occurring'. A resulting model from survival analysis usually contains two components. The first component is a $\lambda_0(t)$ baseline hazard function. This function describes how the risk of the event occurring changes with time, assuming there is no influence from any of the explanatory variables. The second component describes the influence of each explanatory variable on the hazard. This brings us to the notion of proportional hazards.

Time (days)	Censor
249	1
345	1
152	1
452	0
120	1
...	...

Table 3.1: Example outcomes for survival data

3.3 Cox proportional hazards model

3.3.1 Proportional hazards condition

The proportional hazards condition means that we assume that hazard ratios are independent of time. A hazard ratio is the relative risk between two entities (patients). Furthermore the condition states that changes in the explanatory variables have an exponential effect on the hazard. The following paragraphs explain these statements in more detail.

If we assume the proportional hazards condition to be true then this gives us the opportunity to estimate the effect of explanatory variables without dealing with the underlying baseline hazard function. This was an observation made by Sir David Cox[22], hence the name of the model. We can represent the cox model as follows:

$$\lambda_i(t) = \lambda_0(t)e^{X_{i1}\beta_1+...+X_{iN}\beta_N} \tag{3.1}$$

where

- $\lambda_i(t)$ is the hazard for patient i at time t.
- $\lambda_0(t)$ is the baseline hazard at time t.
- $X_{i1}...X_{iN}$ are the values of the explanatory variables for patient i.
- $\beta_1...\beta_N$ are the values of the coefficients for each explanatory variable.

Now, let's look at relative risks between two patients:

$$\frac{\lambda_i(t)}{\lambda_j(t)} = \frac{\lambda_0(t)e^{X_{i1}\beta_1+...+X_{iN}\beta_N}}{\lambda_0(t)e^{X_{j1}\beta_1+...+X_{jN}\beta_N}} = e^{(X_{i1}-X_{j1})\beta_1+...+(X_{iN}-X_{jN})\beta_N} \tag{3.2}$$

This quantity is called the hazard ratio. Notice that this ratio does not depend on time. This means that if at the start of the study a patient has twice the risk of the event occurring compared to another patient, it will have twice the risk at any other time aswell. It's risk remains proportional, independent of time.

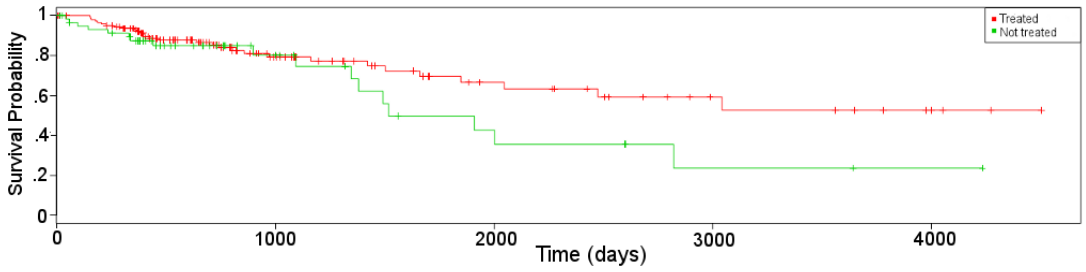


Figure 3.1: Example Kaplan-Meier survival curve. Red and green lines represent survival probabilities for treated and untreated patient-groups respectively.

We can also compute the effect on the hazard for a unit increase in an explanatory variable X_j. The subscripts i to indicate the patients have been omitted since this is independent of a specific patient. The hazard ratio then becomes:

$$\frac{\lambda(t|X_j+1)}{\lambda(t|X_j)} = e^{(X_j+1-X_j)\beta_j} = e^{\beta_j} \tag{3.3}$$

It is directly given by the exponential of the coefficient for the explanatory variable. This reflects the proportional hazards assumption that hazard ratios do not depend on time and that explanatory variables have an exponential effect on the hazard.

3.3.2 Using the hazard ratio

The proportional hazards assumption allows us to compute relative risks or hazard ratios without knowing the actual baseline hazard function. Knowing just relative ratios can be very useful however. Imagine the following scenario: we want to test if a certain drug increases the survivability of patients with a specific disease. We randomly give half of the patients the actual drug, the other half receives a placebo (or nothing at all). Performing a survival analysis using a cox model then allows us to compute the ratio of the hazards for the two groups. The coefficient for the explanatory variable that indicates which patients received the drug and which did not then tells us the effect of the drug on the hazard. We could then make a statement such as: "Taking the drug tends to half the risk of the event occurring (at any time)." This is obviously a useful result to support further research or development for the drug.

3.3.3 Kaplan-Meier survival curves

Kaplan-Meier survival curves[26] are often used in survival analysis. These curves show the probability of surviving up to any point t. It is thus a declining curve starting from 1 at $t = 0$. An example curve is shown on figure 3.1. In formula notation we could write this as:

$$S(t) = P(T > t) \tag{3.4}$$

where

- $S(t)$ is the survival function
- $P(x)$ means 'the probability of x'
- T is the observed event time (e.g. death)
- t is the time variable

Linking hazard to survival

Creating a kaplan-meier survival curve based on the outcome vectors (time and censoring) is very straightforward to do, it is simply a visual representation of the event occurring at certain times. However when we create a model that tries to explain these outcomes from explanatory variables we end up with a hazard function. In order to create survival curves we have to link this hazard function to the survival function[34]. Recall the hazard function $\lambda(t)$ from section 3.3. We could write this function in a similar probabilistic notation as we did for the survival function:

$$\lambda(t) = \lim_{\delta \to 0} \frac{P(t < T \leq t + \delta | T > t)}{\delta} \tag{3.5}$$

this is really just the definition of hazard: the risk of the event happening at time t is defined as the event happening during an infinitesimal timeframe after time t, given that the event has not happened yet up to time t. As we take the limit $\delta \to 0$ this becomes an instantaneous hazard. We can now apply the conditional probability rule $P(A|B) = \frac{P(A \cap B)}{P(B)}$ to get:

$$\begin{aligned}
\lambda(t) &= \lim_{\delta \to 0} \frac{P(t < T \leq t + \delta \cap t < T)}{\delta} \frac{1}{P(T > t)} \\
&= \lim_{\delta \to 0} \frac{P(t < T \leq t + \delta)}{\delta} \frac{1}{S(t)} \\
&= \lim_{\delta \to 0} \frac{P(t < T) - P(t + \delta \leq T)}{\delta} \frac{1}{S(t)} \\
&= -\lim_{\delta \to 0} \frac{P(t + \delta \leq T) - P(t < T)}{\delta} \frac{1}{S(t)} \\
&= -\frac{\partial S(t)}{\partial t} \frac{1}{S(t)} \\
&= -\frac{S'(t)}{S(t)}
\end{aligned} \tag{3.6}$$

Let's look at this result in an intuitive way. Assume the event we are talking about is dying. The hazard function then tells me what the risk is of dying at time t. Equation 3.6 tells us that this is equal to $-\frac{S'(t)}{S(t)}$. There are three parts to explain here:

- $\frac{1}{S(t)}$: The first thing we have to do is survive up to time t. Otherwise it makes no sense to ask what the risk of dying at time t is. The survival function tells

us how probable it is to survive up to time t so dividing by this quantity exactly provides what we need. If $S(t)$ is large, then this means many people survive up to time t and so the risk is low. If $S(t)$ is small then few people actually survive up to time t and thus the risk of having to survive up to this point gets bigger.

- $S'(t)$: Now that we have survived up to time t, we need to calculate the risk of dying at this point in time. Remember that the survival curve declines whenever people die at certain points in time. The slope of this function therefore tells us how many people die at that time t. If $S'(t)$ is very large, it means that many people die here. The risk of dying thus becomes large.
- The hazard is supposed to be a positive quantity. Since the survival function is a strictly declining curve (people usually don't resurrect from the dead), $S'(t)$ is negative. Therefore we need the minus sign to end up with a positive result.

Next, using the chain rule, we can observe that

$$\frac{\partial log(S(t))}{\partial t} = \frac{S'(t)}{S(t)} \tag{3.7}$$

and thus combining equations 3.6 and 3.7:

$$\lambda(t) = -\frac{\partial log(S(t))}{\partial t} \tag{3.8}$$

$$\int_0^t \lambda(t)dt = \int_0^t -\frac{\partial log(S(t))}{\partial t}dt \tag{3.9}$$

The left hand side of equation 3.9 is called the cumulative hazard fuction and often denoted $\Lambda(t)$. We can then write:

$$\Lambda(t) = -log(S(t)) \tag{3.10}$$

$$S(t) = e^{-\Lambda(t)} \tag{3.11}$$

Equation 3.11 allows us to relate the survival function to the hazard function. This allows us to create survival curves for any form of hazard function. The only thing that is left to do is to define the baseline hazard function, because otherwise we could not compute the integral in $\Lambda(t)$. Usually an estimator based on likelihood is used to estimate the baseline hazard function. There are many forms available for this estimator[35], but we will not go into any more detail here.

3.4 Conclusion

In this chapter we have explained the cox proportional hazards method. We have shown that it is used for survival analysis, where the targets are time and censoring variables. We have explained the proportional hazards assumption and shown its implications. Lastly we have shown how we can compute survival curves using the Kaplan-Meier method and we have shown a relationship between the hazard function and the survival function.

Chapter 4

Integration Strategies

4.1 Introduction

Now that we have established a firm understanding of the methods we will use to create predictive models, it is time to turn our attention to the integration step. In this chapter we will present three different integration strategies. We will start by providing a description of the datasets that we will use to build our models, followed by the integration strategies that show how we can combine these datasets to create predictive models. The first strategy is called early integration and is based on concatenation of datasets. The second strategy is called late integration which is based on ensemble learning. The last strategy is intermediate integration which makes extensive use of the variable selection present in the lasso regularization(2.5).

4.2 Description of the data

In order to integrate the datasets, they have to adhere to a number of requirements. Suppose we have a set of D datasets, each dataset can be represented by a matrix where the rows are the samples and the columns are the explanatory variables. Each dataset can have different explanatory variables, but they all need to have rows that can be linked across datasets, for instance using a unique patient identifier. Let me give a concrete example: imagine we have a set of 200 cancer patients. Each of these patients has had images taken of the cancer by different machines: an MRI scan, a PET scan, ... and they also had a bloodtest done measuring different proteïne levels in the blood (an ELISA test). Each of these methods will yield a dataset: one for each imaging scan and one for the blood test. The explanatory variables are obviously different for each dataset: the imaging datasets will have image features like blobs and pixel intensities, while the bloodtest will have values that represent the proteïne levels in the blood. Each dataset will however have a single row per patient, uniquely identified by that patients ID. This is not true in the case of missing data (e.g. a patient missed his MRI scan appointment). But there are ways of dealing with this: we can try to estimate the missing values using clustering techniques, or we can simply leave the patients with missing data out of the study.

Now that we have our input data, we need a target function (dependent variable). In the example case this could be whether the patient fully recovers from the treatment or not. This variable is supposed to have a binomial distribution and thus we could use logistic regression to try and estimate this variable. The task of integration is now the following: how do we combine the samples across the different datasets to come up with a predictive model that uses the information in all the datasets? In the following sections we will present three ways to do this: the early integration method which is concatenation, the late integration method that is based on an ensemble of models, and the intermediate integration method that takes advantage of variable selection present in the lasso regularization.

4.3 Early integration

The first integration method is called early integration. In this method we are simply going to concatenate the data for each sample. Figure 4.1 shows the schema for early integration of data sources. A new large dataset is constructed by concatenating the individual datasets by matching sample. The number of explanatory variables is now equal to the sum of explanatory variables of all individual datasets. This integrated dataset can then be used to train a predictive model using techniques seen in previous chapters (2,3).

4.4 Late integration

In late integration, as the name suggests, we will combine the different datasets at the end of the learning process. The schema for late integration is shown on figure 4.2. First, a model is learned for each dataset individually. This gives rise to D different models. To reuse the example from section 4.2, there will be a model for MRI data, PET data, etc. When we want to predict an outcome for a new sample (patient) we will present each model with the corresponding input from the patient. Each model will then compute an output and these are combined using a linear combination. If we have no preference for any model we could simply compute the average output. This average would then be our final output for the integrated model. In this case we cannot represent the final model by some model parameters (weights) but rather we have to view the full set of D models as well as the linear combination we chose as the full integrated model.

There is an analogy between late integration and the ensemble averaging technique in machine learning. Ensemble learning[24][23][1] means that instead of just building one model for a dataset, we build several models for the same dataset and then average their outcomes. Late integration takes this idea but applies it to a set of data sources instead of just one.

The thought behind this model is that we try to learn as much as possible from

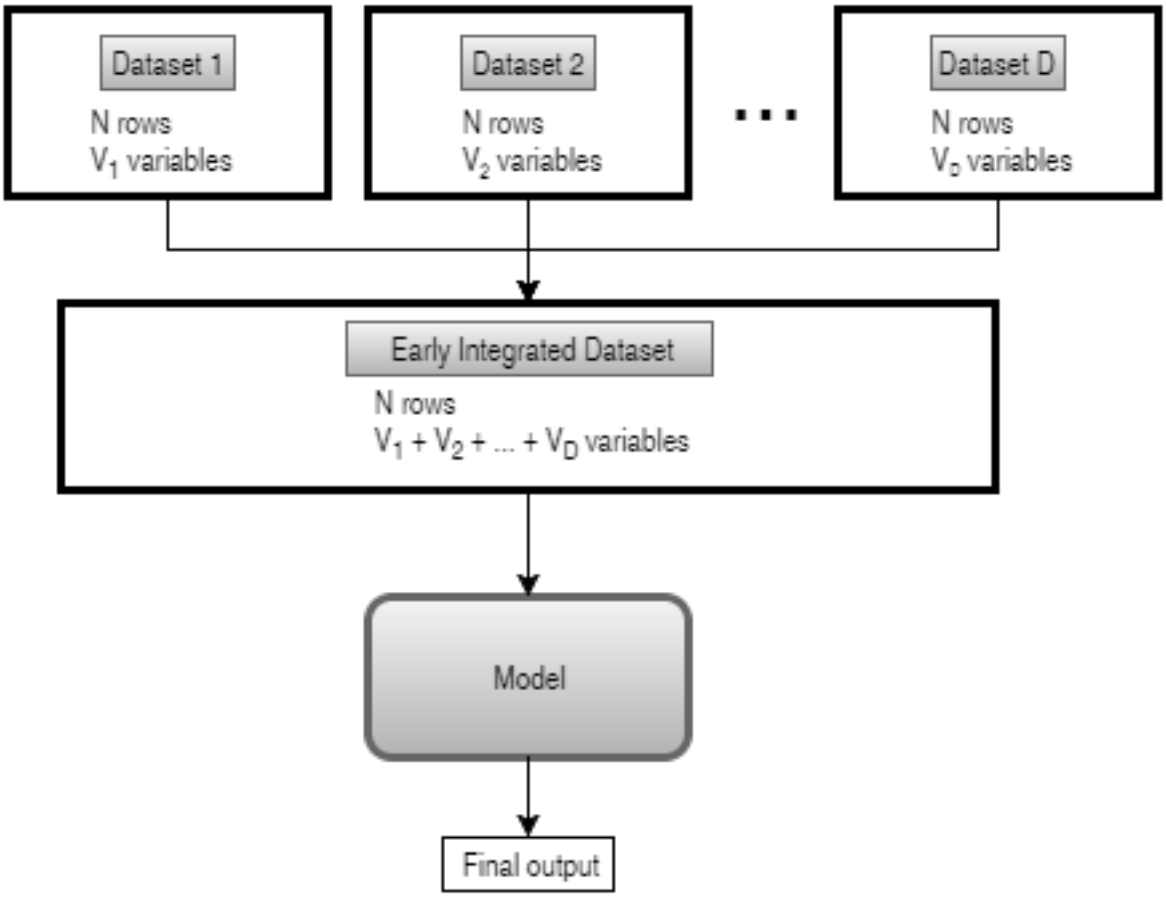


Figure 4.1: Scheme for early integration

each dataset individually, before we combine them. An example of this is that the variable selection by the lasso regularization (section 2.5.2) now has the opportunity to select the best variables for each dataset individually. While in the case of early integration, all the variables from the datasets are competing against each other at the same time to make it into the final model.

4.5 Intermediate integration

Intermediate integration is a novel integration technique that tries to take advantage of the variable selection when using a lasso penalty. The schema for intermediate integration is shown on figure 4.3. It starts out the same way as late integration, for each dataset we compute an individual model. However, instead of using the output of these models we will simply look at the variables that were chosen by each model to be informative. We will then go back to the original datasets and extract only those explanatory variables and concatenate them together into a new integrated dataset. Lastly, we use this integrated dataset to learn a final model.

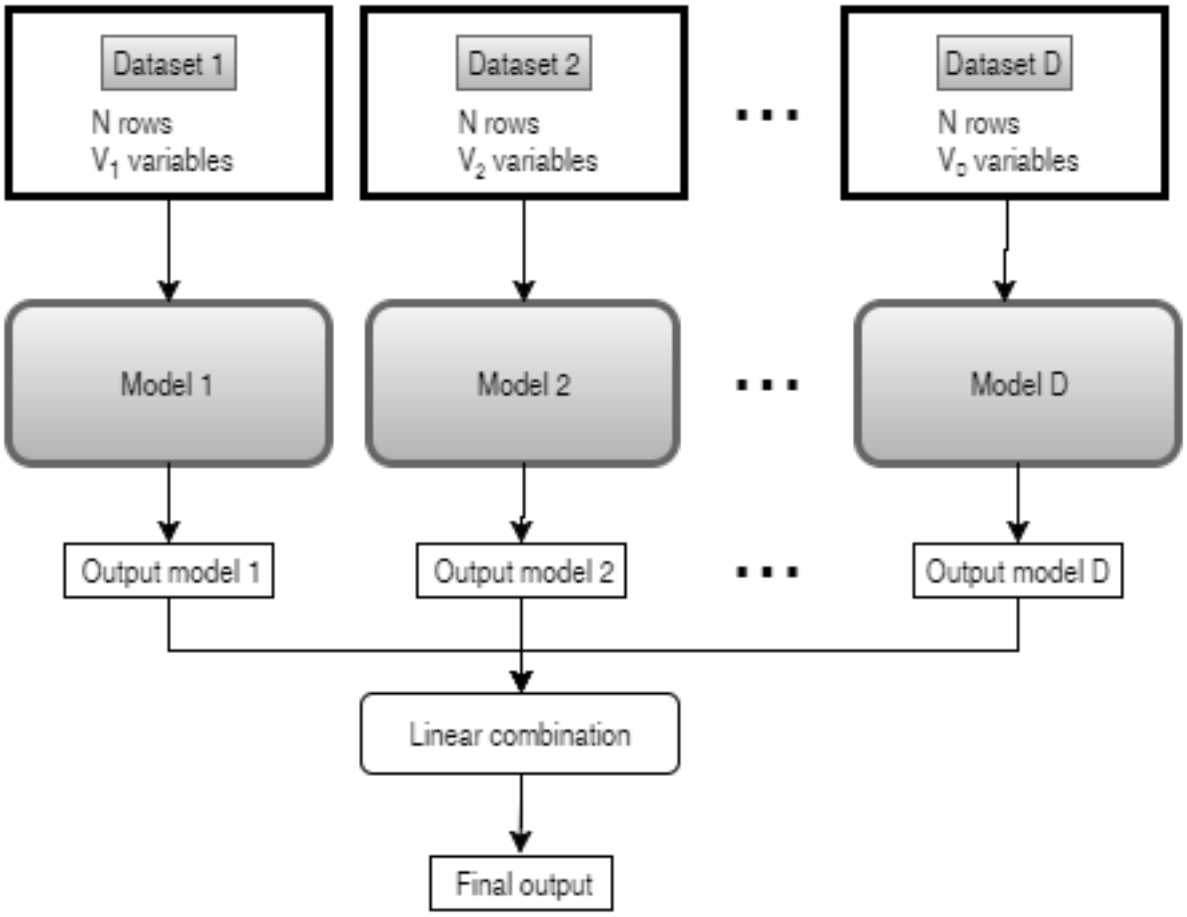


Figure 4.2: Scheme for late integration

The intermediate integration can be seen as a two-step learning process. First we preprocess the datasets by computing individual models and we extract only the informative variables. Then we will train our final model on the reduced dataset. This is very advantageous if the initial datasets have a huge amount of explanatory variables, as the preprocessing step will reduce this amount drastically while still keeping as much useful information as possible.

4.6 Conclusion

In this chapter we have shown the issue of large and high-dimensional datasets that we are currently facing. We have presented three different strategies to deal with this issue. The early integration method is the naïve method that simply concatenates the datasets together. The late integration method uses an analogous technique to ensemble learning in order to extract as much information as possible. Lastly, the

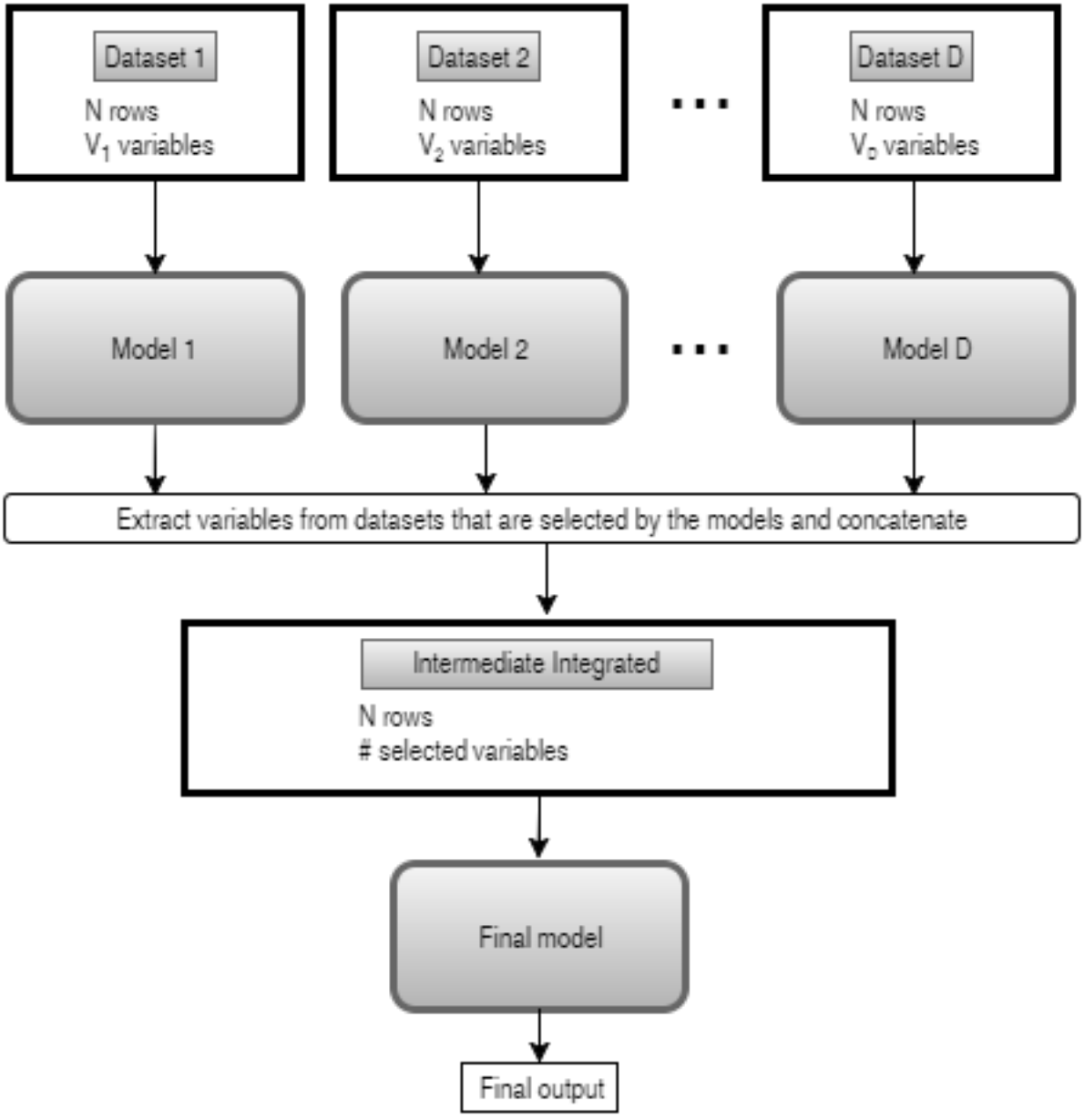


Figure 4.3: Scheme for intermediate integration

intermediate integration technique introduces a two-step learning method that takes advantage of variable selection to reduce the dimensionality of the datasets.

Chapter 5

Evaluation of integration strategies

5.1 Introduction

In this chapter we will show that different integration strategies can have better performance in different situations. We will do this by computing a model using each strategy and then comparing their performance using an appropriate metric.

5.2 Evaluating a logistic regression model

The first evaluation will be made using a logistic regression model. This is a model computed for a dependent variable that has a binomial distribution. To evaluate such model, a common technique used is called Receiver-Operating-Characteristic analysis[43][27][13] or ROC analysis for short. Remember that when we are using a logistic regression model to predict outcomes, we have to define a threshold to separate positive from negative cases. An ROC curve shows the performance of the model for any possible threshold, using metrics such as sensitivity and specificity. This gives a very good general impression of the performance of the model.

5.2.1 Predictions using validation

What we really want to do is estimate the out-of-sample error for our model. To do this we have seen that we can use the technique of cross-validation. In this case we are not going to compute any learning parameter but we are going to use the validation set as a test set. We will split our dataset into K folds. We will train a model on $K-1$ folds and we will use this model to predict the outcomes of the remaining samples in the test fold. By repeating this process for each folds we obtain a predicted outcome for each sample in the dataset. Notice however that we also have the real outcome of the samples in our dataset, and thus we can compare our predictions with the real values to estimate our out-of-sample error.

5.2.2 Metrics

When we compare our models predictions with the real outcomes, there are four possible scenario's:

- True Positive (TP): the model predicted positive and this is correct
- True Negative (TN): the model predicted negative and this is correct
- False Positive(FP): the model predicted positive and this is wrong (type 1 error)
- False Negative (FN): the model predicted negative and this is wrong (type 2 error)

Notice that this gives us more information than just registering whether we are right or wrong. It also tells us the type of mistake we made. This is a very important distinction because depending on the application there is usually a different cost attached to making type 1 or type 2 errors. And as we will see later on, there is a tradeoff between the two and we can tune our system to be more resistant towards making one type of error.

Now that we have established the notion of typed errors, there are several ways in which these numbers are combined to form evaluation metrics. Depending on your field of research these often get different names, we will use the ones applicable to the biomedical field.

Sensitivity

Sensitivity[14] measures the proportion of the real positive samples that our model correctly predicted as positive. It can be calculated with the following formula:

$$Sensitivity = \frac{\sum TP}{\sum P}$$

where

- $\sum TP$ is the number of true positives
- $\sum P$ is the number of real positive cases in the dataset

Sensitivity can be thought of, as its name suggests, as how sensitive the model is to detecting positive cases. If the model gets a real positive sample's input, what is the probability that it will detect it as such. A sensitivity of 1 means that our model is capable of correctly identifying all positive cases in the dataset. This means that the higher the sensitivity, the lower the type 2 error rate.

Specificity

Specificity[14] is the analogous metric to sensitivity, but for negative cases. It measures the proportion of the real negative samples that our model correctly predicts as negative.

$$Specificity = \frac{\sum TN}{\sum N}$$

where

- $\sum TN$ is the number of true negatives
- $\sum P$ is the number of real negative cases in the dataset

A specificity of 1 means that our model is capable of correctly identifying all negative cases in the dataset. This means that the higher the specificity, the lower the type 1 error rate.

The tradeoff

At this point it is obvious that we would like to maximize both sensitivity and specificity. Indeed, if both metrics are 1 then we make no errors and we have a perfect model. In practice however this is nearly impossible to achieve. It is easy to see that there is a tradeoff between the two metrics: when we try to increase the sensitivity, we will lose out on specificity and vice versa.

Consider the totally useless model that has a threshold lower than the smallest possible output. Meaning that this model will always predict a positive outcome regardless of the input. This model will have a sensitivity of 1, as we will get all positive cases correct. But it will also have a specificity of 0 because we get none of the negative cases correct. As we gradually increase the threshold we will cross critical values where some sample inputs will now produce an output below the threshold and thus be classified as negatives. If this happens to be a correct prediction our specificity will go up and sensitivity will remain unchanged. If it happens to be an error, specificity is unchanged and sensitivity will go down. The gradual increase of the threshold will thus cause sensitivity to drop and specificity to increase, up to the point where we reach the other extreme. The threshold is now bigger than the biggest possible output and the model always predicts negative regardless of the input. This threshold gives us a sensitivity of 0 and a specificity of 1.

5.2.3 Receiver Operating Characteristic curve

The ROC curve is a way of showing exactly this gradual increase of the threshold. The typical way of constructing a ROC curve is to put $(1 - specificity)$ on the X-axis and *sensitivity* on the Y axis. The values for these matrics are then plotted for values of the threshold ranging from the all-positive model to the all-negative model. An example ROC curve is shown on figure 5.1. The metric we will use to evaluate the performance of different models is the area under the ROC curve (AUC). This metric

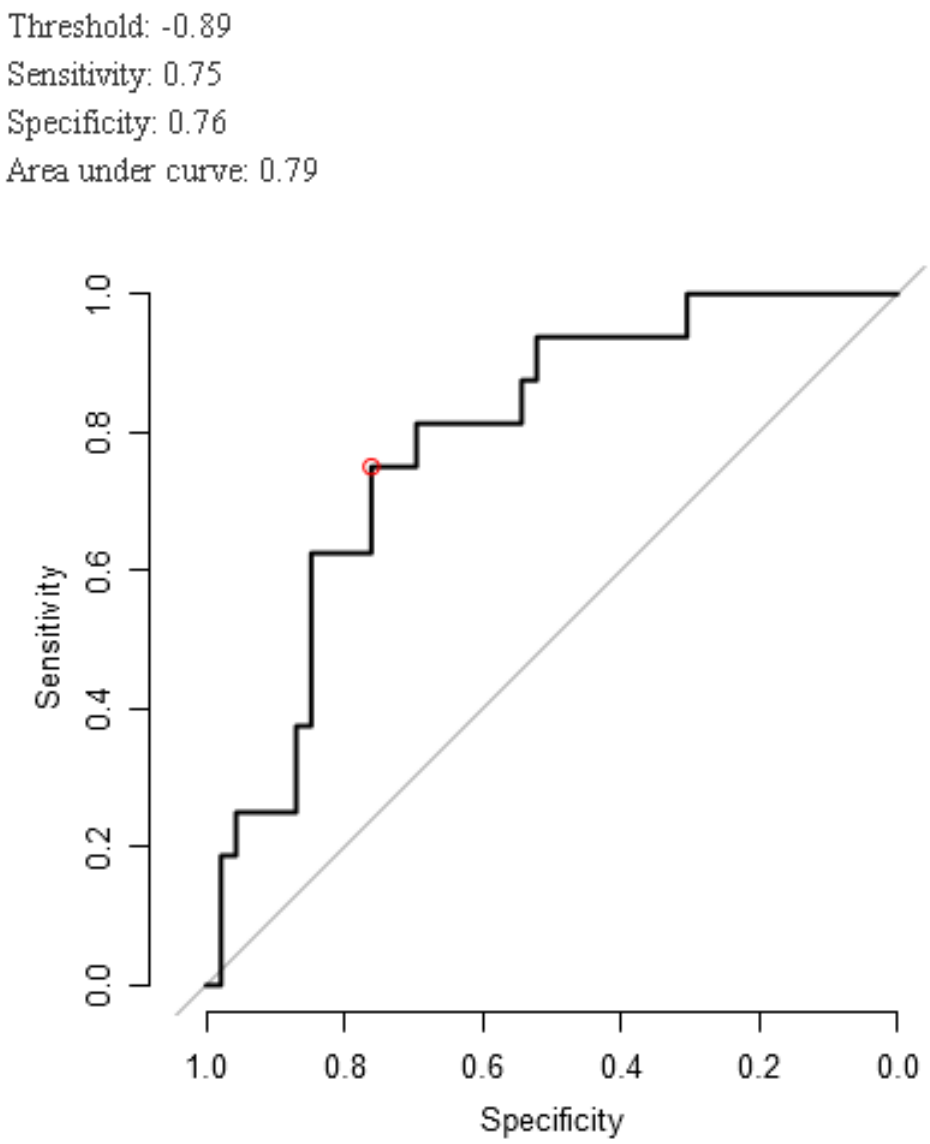


Figure 5.1: Example ROC curve and metrics

works because it indicates how far we can optimize both sensitivity and specificity for different thresholds. It shows how close we can get to the upper left corner of the plot, which indicates the perfect model. The AUC metric ranges from 0 to 1, but in fact its value is only informative in between 0.5 and 1. Imagine a model that makes completely random predictions, this model will get a correct prediction 50% of the time. If we would create a ROC curve for this model we would on average get a straight diagonal line from the bottom left to the top right. This line is therefore considered a lower bound for the performance of a model.

5.3 Case study 1: predicting stage outcome

5.3.1 Case details

The case we are showing here uses datasets created by the University Hospital of Leuven in a study on colorectal cancer patients. There are three source datasets that all contain imaging data: a dataset from MRI (Magnetic Resonance Imaging) scans, a dataset from DWI (Diffusion Weighted Magnetic Resonance Imaging) scans, and a dataset from PET (Positron Emission Tomography) scans. It is important to note that these scans were taken 3 times per patient at different periods in their treatment: one in the beginning, one roughly in the middle, and one near the end. This means that there is some temporal information present in the datasets, but this has been encoded in the explanatory variables. For instance: there is a variable in the MRI dataset that shows the size change of the tumour between the first and second scan. In this way we can treat this as a regular explanatory variable and we don't have to worry about the temporal aspect.

The dependent variable is a binary outcome based on whether the patient got a pathologically complete response. For a more in-depth explanation on what this means see appendix **??**. For now, we can view a positive outcome as a patient who has recovered from the cancer, and a negative outcome as a patient that has not. The aim is now to build a model that receives the explanatory variables from the scans as input, and predicts the outcome for that patient.

5.3.2 The results

The resulting AUC's for the individual and integrated datasets are shown respectively in tables 5.1 and 5.2. We can clearly see that the model for the individual MRI dataset contains the most predictive information out of the three datasets. This however doesn't mean that the other datasets are useless. Indeed, when we integrate the other dataset with the MRI dataset we get even better performances. From table 5.2 we can see that in this case the intermediate integration method outperforms the other strategies regardless of which datasets are used.

I have also included an overview of the model parameters that were selected (on average) by each strategy. The values in this table show the weights given to each corresponding variable in the model. A positive weight indicates that a higher value for this variable increases the chance of a positive outcome. A negative weight correlates the variable with a decreased probability of positive outcome. The overview is shown in table 5.3. The first column shows the name of the selected variable, these are imaging features. The next three columns show the model parameters for the individual models. The last two columns show the parameters that were selected by the integrated models. It is interesting to see how all models tend to select the same kind of variables and give them similar weights. This shows us that these variables indeed do contain some predictive information. Notice that the intermediate

Dataset	AUC
DWI	0.67
MRI	0.75
SUV	0.65

Table 5.1: AUC for models of individual datasets

	All	DWI+MRI	DWI+SUV	MRI+SUV
Early integration	0.76	0.82	0.69	0.74
Intermediate integration	0.79	0.83	0.78	0.75
Late integration	0.73	0.78	0.66	0.70

Table 5.2: AUC for models of various combinations of integrated datasets

	Individual			Integrated	
Variable	DWI	MRI	SUV	Early	Intermediate
ADCratioavg_TP1TP3	0.69			1.09	1.19
ADCratiohigh_TP1TP3	0.08			0.50	0.62
DeltaSphere_TP2TP3perc		1.67		1.35	1.44
DeltaSphere_TP1TP3perc		0.48		1.78	1.64
Volume_TP2		-0.99			
Volume_TP3		-0.24		-0.173	-0.94
RIdiameter_TP2TP3			0.16		
Diameter_TP3			-0.88	-0.01	
SUVmax_TP2			-0.88	-3.30	-4.41
deltadiameter_TP1TP2			-0.32		
RISUVpeak_TP1TP2				0.32	
RITLG_TP2TP3				-0.08	

Table 5.3: Overview of the model parameters (weights)

model selects fewer variables than the other models, while still outperforming them. It removes those variables that have very small weights in other models and thus acts as a very strict variable selector. This is no surprise if we remember how the intermediate integration method uses two steps of variable selection to perform the integration (chapter 4, section 4.5).

5.4 Evaluating a cox proportional hazards model

The evaluation of a cox proportional hazards model is different compared to logistic regression[33][28]. This is because, as we have seen in chapter 3, we normally don't directly compute survival times from a cox proportional hazards model. This means we cannot compare them to the real survival times. However we can still evaluate the models based on statistics that indicate significance.

5.4.1 Predictions for survival models

Recall the hazard ratio from section 3.3. If we have a model computed ($\beta's$ defined) and we have a test sample (with values X_i for the explanatory variables), we could compute the following prediction quantity:

$$prediction = exp(\sum_{i=0}^{N} X_i\beta_i) \tag{5.1}$$

where

- $exp(x)$ is the exponential of x
- X_i are the values for the explanatory variables
- β_i are the model coefficients

We can view this as the hazard ratio between the test sample and an imaginary patient that has the value 0 for all explanatory variables which we will call the baseline patient (subscript b):

$$\frac{\lambda_i(t)}{\lambda_b(t)} = \frac{\lambda_0(t)e^{X_{i1}\beta_1+...+X_{iN}\beta_N}}{\lambda_0(t)e^{0\beta_1+...+0\beta_N}} = e^{(X_{i1}-0)\beta_1+...+(X_{iN}-0)\beta_N} = exp(\sum_{j=0}^{N} X_{ij}\beta_j) \tag{5.2}$$

What this quantity tells us is whether the risk of our test patient is higher or lower than the risk of the baseline patient. This does not tell us anything in the absolute sense about survival times, but we can compare this prediction value in a relative way between patients.

We could, as we did in section 5.2, compute this prediction value for all patients in our dataset using cross-validation. We now want to test the predictive value of these predictions. To do this we construct a new survival model, but this time there is only one explanatory variable and that is our vector of predictions. If we can show that this variable is significant in predicting the survival, then our original model is also significant. The significance test we will use is called the Wald Test.

5.4.2 The Wald test

The Wald test[17] is a statistical test that can be used to test the significance of an estimated parameter. The Wald test can be used on a single parameter or on a vector of parameters at once. In the case of one parameter β the formula for the Wald test is:

$$\frac{(\hat{\beta} - \beta_0)^2}{var(\hat{\beta})} \sim \tilde{\chi}_1^2 \tag{5.3}$$

where

- $\hat{\beta}$ is the maximum likelihood estimate we calculated and want to test
- β_0 is the value of β if the null-hypothesis is true
- $var(\beta)$ is the variance of β
- $\tilde{\chi}_1^2$ is the chi-squared distribution with one degree of freedom

When we are testing the significance of a coefficient in our survival model we have to compare it against a so called null-hypothesis. In this case the null-hypothesis is the model where all coefficients are equal to zero. The values for $\hat{\beta}$ and $var(\beta)$ are defined by the model. Intuitively the Wald test computes the distance between the obtained value for the parameter and the parameter value under the null-hypothesis, relative to the variance of the parameter. If the variance is large, then we are unsure about the value for the parameter that we calculated and it is harder to reject the null-hypothesis. An example is demonstrated on figure 5.2.

The Wald test can tell us something about the significance of the predictions vector and therefore about the predictive ability of our initial model. We can go one step further and compute a p-value[6][11] for the wald-test statistic. This p-value contains the exact same information as the wald-test statistic, but it is a very well known quantity statistics so we will add it for completeness.

5.5 Case study 2: predicting survival outcomes

5.6 Conclusion

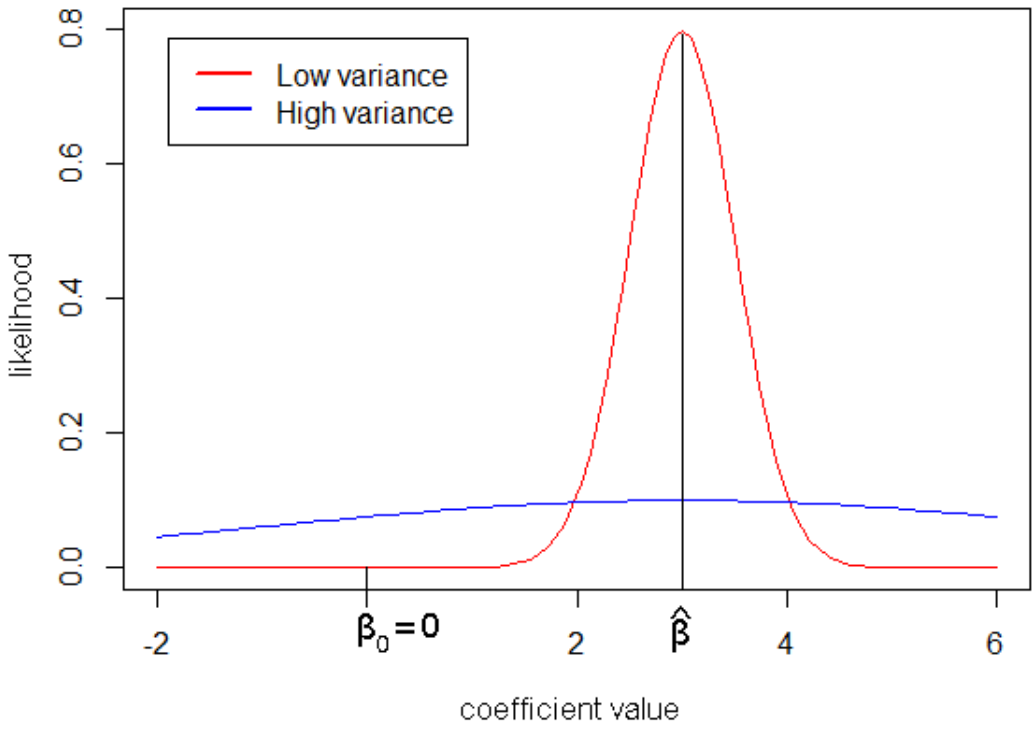


Figure 5.2: Example Wald test variance visualisation. Both curves give the same maximum likelihood estimate $\hat{\beta}$. The red curve however has low variance, providing good evidence to reject the null-hypothesis with value β_0. The blue curve has high variance, in this case $\hat{\beta}$ does not significantly differ from β_0 and we cannot reject the null-hypothesis.

Chapter 6

Tool for automated evaluation

6.1 Introduction

In the previous chapter we have shown that different integration strategies do make a difference for the performance of the resulting model. In order to facilitate future research on this topic, we have developed a tool that allows anyone to very easily compute all the integrated models and compare them.

6.2 Technologies

To develop this tool we needed a programming language that provided a lot of support for statistical algorithms, and we also wanted to create a very user-friendly interface to make an abstraction of the underlying machine learning methods.

6.2.1 The R language

The language we used to create the tool is R. R is a very popular language for statistical research as it provides many state-of-the-art algorithms in statistics. R has a huge repository with packages for all kinds of purposes. The package that we used to compute models is glmnet[29]. It is developed by researchers at the Stanford University and provides very fast implementations to compute the generalized linear models explained in chapter 2.

6.2.2 Web interface

I wanted to offer a very easy-to-use interface to the application. Therefore we chose to work with shiny[15]. Shiny is a framework for R that allows you to create a web-based front-end (using html, css, javascript, ...) that can also run R code in the back-end. This allows the creating of very interactive web applications that can run the powerful statistical analyses available in R.

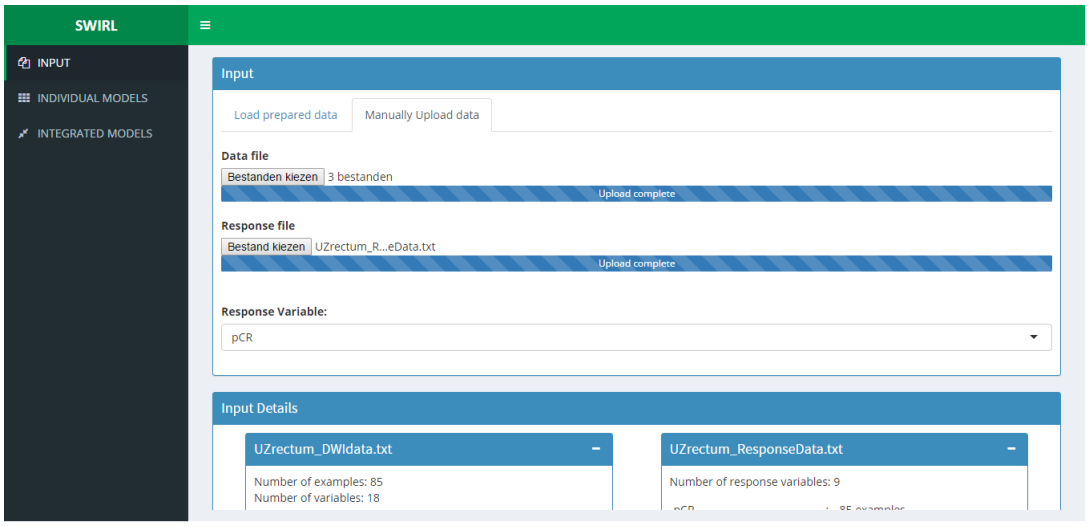


Figure 6.1: The input tab of the tool

6.3 Demonstration

The tool is divided into three sections: an input section where the user can upload their datasets, a section that computes and evaluates models for each dataset individually, and an integration section where all of the integrated models, explained in chapter 4, can be computed and evaluated.

The input tab

The input tab allows the user to either use a preloaded dataset or upload their own datasets to the application. It is possible to define multiple dependent variables in one file, the tool then gives you the option to select which variable has to be modeled.

When datasets are uploaded, the tool gives a small overview of the dimensions of each dataset so the user can verify everything works correctly. Figures 6.1 and 6.2 show the input page where a user has uploaded some datasets.

The individual models tab

Once a user has uploaded their datasets they can compute a model for each dataset individually. The tool will give a concise overview of the model parameters for each model, and show an evaluation of the model. In case of logistic regression, the evaluation would be a ROC curve (explained in section 5.2). Notice that the evaluation is interactive: the user can use the slider to see the performance metrics for different thresholds. Figure 6.3 shows an example model description. Figure 6.4 shows an example ROC curve.

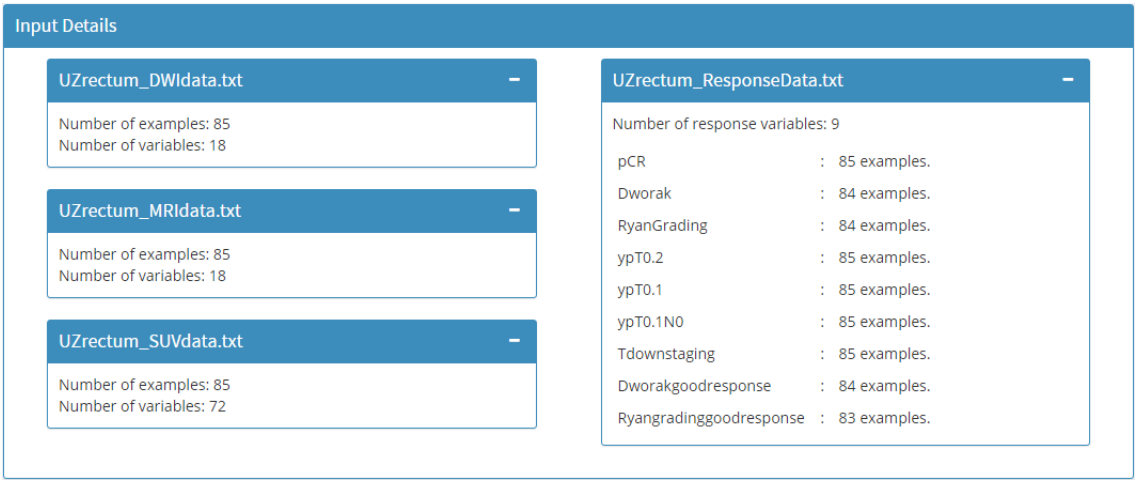


Figure 6.2: Example input overview. The boxes on the left show explanatory variable dataset dimensions, the box on the right shows the possible dependent variables.

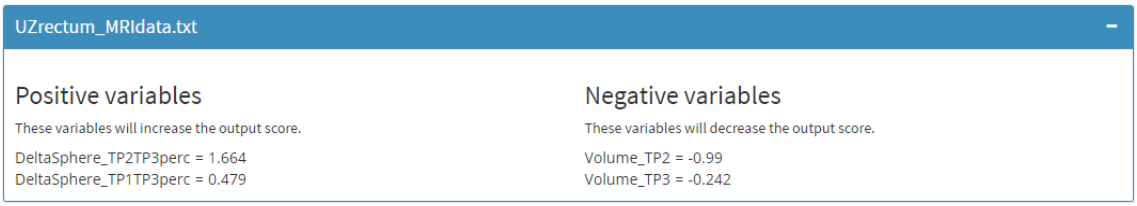


Figure 6.3: An example model description showing the coefficients for the variables in the model separated by their sign.

The integration tab

The integration tab is the most important part of the application, as it allows the user to compute and evaluate integrated models for their datasets. The integration tab is further split up into early-, late- and intermediate integration tabs. Each tab allows the user to compute and evaluate its respective integration method. The results will look exactly the same as shown for the individual models tab, as can be seen on figures 6.3 and 6.4.

6.4 Conclusion

To make further research easier we have made an interactive web-based tool. The tool allows users to compute and evaluate all integration strategies explained earlier. It currently supports logistic regression models and cox proportional hazard models. The back-end uses the statistical language R and the glmnet package to compute the models. The front-end is purely web-based and this is made possible by the Shiny framework.

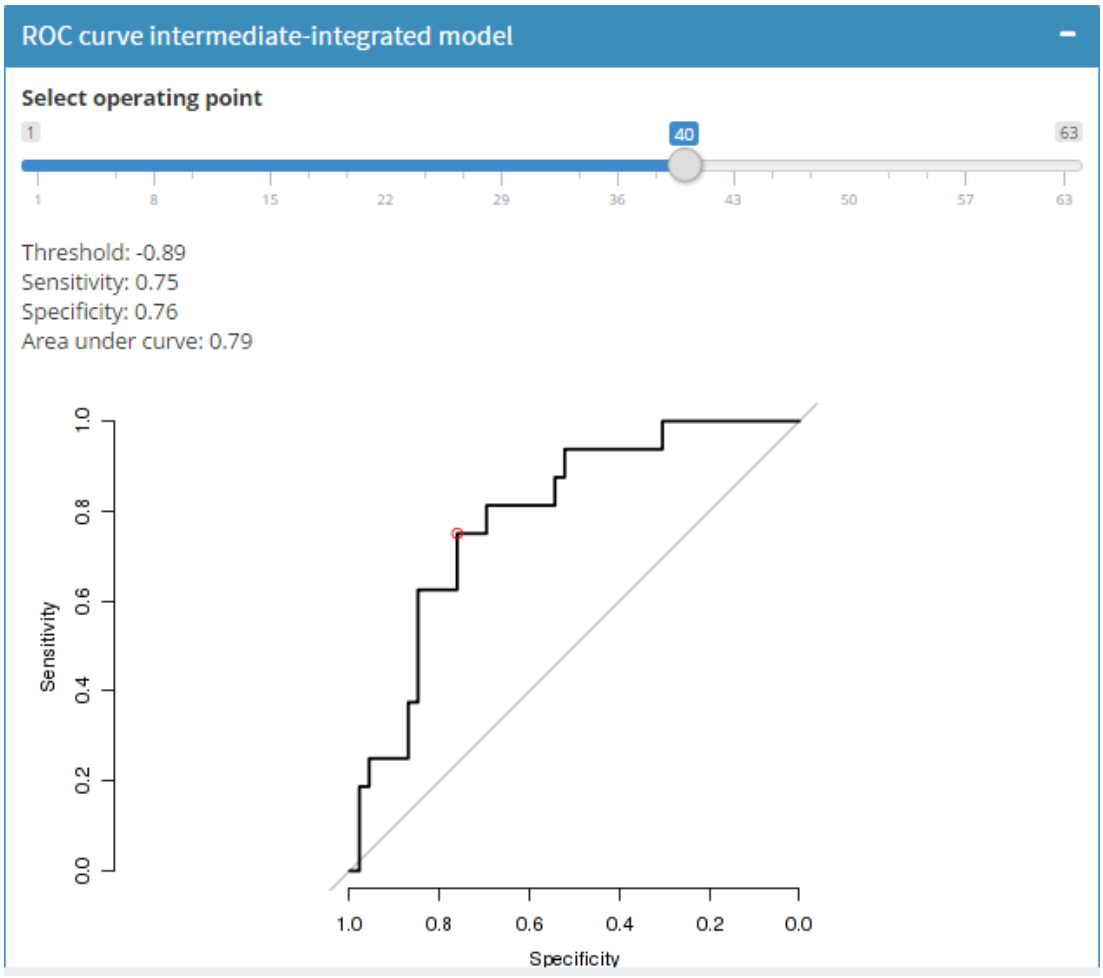


Figure 6.4: An example interactive ROC curve. The slider on top allows the user to vary the threshold. The plot interactively adjusts itself to the chosen threshold.

Chapter 7

Conclusion

The final chapter contains the overall conclusion. It also contains suggestions for future work and industrial applications.

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Ut purus elit, vestibulum ut, placerat ac, adipiscing vitae, felis. Curabitur dictum gravida mauris. Nam arcu libero, nonummy eget, consectetuer id, vulputate a, magna. Donec vehicula augue eu neque. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Mauris ut leo. Cras viverra metus rhoncus sem. Nulla et lectus vestibulum urna fringilla ultrices. Phasellus eu tellus sit amet tortor gravida placerat. Integer sapien est, iaculis in, pretium quis, viverra ac, nunc. Praesent eget sem vel leo ultrices bibendum. Aenean faucibus. Morbi dolor nulla, malesuada eu, pulvinar at, mollis ac, nulla. Curabitur auctor semper nulla. Donec varius orci eget risus. Duis nibh mi, congue eu, accumsan eleifend, sagittis quis, diam. Duis eget orci sit amet orci dignissim rutrum.

Nam dui ligula, fringilla a, euismod sodales, sollicitudin vel, wisi. Morbi auctor lorem non justo. Nam lacus libero, pretium at, lobortis vitae, ultricies et, tellus. Donec aliquet, tortor sed accumsan bibendum, erat ligula aliquet magna, vitae ornare odio metus a mi. Morbi ac orci et nisl hendrerit mollis. Suspendisse ut massa. Cras nec ante. Pellentesque a nulla. Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Aliquam tincidunt urna. Nulla ullamcorper vestibulum turpis. Pellentesque cursus luctus mauris.

Nulla malesuada porttitor diam. Donec felis erat, congue non, volutpat at, tincidunt tristique, libero. Vivamus viverra fermentum felis. Donec nonummy pellentesque ante. Phasellus adipiscing semper elit. Proin fermentum massa ac quam. Sed diam turpis, molestie vitae, placerat a, molestie nec, leo. Maecenas lacinia. Nam ipsum ligula, eleifend at, accumsan nec, suscipit a, ipsum. Morbi blandit ligula feugiat magna. Nunc eleifend consequat lorem. Sed lacinia nulla vitae enim. Pellentesque tincidunt purus vel magna. Integer non enim. Praesent euismod nunc eu purus. Donec bibendum quam in tellus. Nullam cursus pulvinar lectus. Donec et mi. Nam vulputate metus eu enim. Vestibulum pellentesque felis eu massa.

Quisque ullamcorper placerat ipsum. Cras nibh. Morbi vel justo vitae lacus tincidunt ultrices. Lorem ipsum dolor sit amet, consectetuer adipiscing elit. In

hac habitasse platea dictumst. Integer tempus convallis augue. Etiam facilisis. Nunc elementum fermentum wisi. Aenean placerat. Ut imperdiet, enim sed gravida sollicitudin, felis odio placerat quam, ac pulvinar elit purus eget enim. Nunc vitae tortor. Proin tempus nibh sit amet nisl. Vivamus quis tortor vitae risus porta vehicula.

Fusce mauris. Vestibulum luctus nibh at lectus. Sed bibendum, nulla a faucibus semper, leo velit ultricies tellus, ac venenatis arcu wisi vel nisl. Vestibulum diam. Aliquam pellentesque, augue quis sagittis posuere, turpis lacus congue quam, in hendrerit risus eros eget felis. Maecenas eget erat in sapien mattis porttitor. Vestibulum porttitor. Nulla facilisi. Sed a turpis eu lacus commodo facilisis. Morbi fringilla, wisi in dignissim interdum, justo lectus sagittis dui, et vehicula libero dui cursus dui. Mauris tempor ligula sed lacus. Duis cursus enim ut augue. Cras ac magna. Cras nulla. Nulla egestas. Curabitur a leo. Quisque egestas wisi eget nunc. Nam feugiat lacus vel est. Curabitur consectetuer.

Suspendisse vel felis. Ut lorem lorem, interdum eu, tincidunt sit amet, laoreet vitae, arcu. Aenean faucibus pede eu ante. Praesent enim elit, rutrum at, molestie non, nonummy vel, nisl. Ut lectus eros, malesuada sit amet, fermentum eu, sodales cursus, magna. Donec eu purus. Quisque vehicula, urna sed ultricies auctor, pede lorem egestas dui, et convallis elit erat sed nulla. Donec luctus. Curabitur et nunc. Aliquam dolor odio, commodo pretium, ultricies non, pharetra in, velit. Integer arcu est, nonummy in, fermentum faucibus, egestas vel, odio.

Sed commodo posuere pede. Mauris ut est. Ut quis purus. Sed ac odio. Sed vehicula hendrerit sem. Duis non odio. Morbi ut dui. Sed accumsan risus eget odio. In hac habitasse platea dictumst. Pellentesque non elit. Fusce sed justo eu urna porta tincidunt. Mauris felis odio, sollicitudin sed, volutpat a, ornare ac, erat. Morbi quis dolor. Donec pellentesque, erat ac sagittis semper, nunc dui lobortis purus, quis congue purus metus ultricies tellus. Proin et quam. Class aptent taciti sociosqu ad litora torquent per conubia nostra, per inceptos hymenaeos. Praesent sapien turpis, fermentum vel, eleifend faucibus, vehicula eu, lacus.

Appendices

Appendix A

Cross-entropy error derivation

In this appendix I will explain where the cross-entropy error measure for logistic regression comes from. First I will explain the setting and make sure all used symbols are clear. Next I will introduce the notion of likelihood. And lastly I will derive the cross-entropy error function based on the likelihood idea.

A.1 The setting of logistic regression

Remember that in logistic regression we are trying to estimate a probability. The output variable y is a binary variable that comes from an underlying binomial distribution, think of tossing a coin and observing heads or tails. Let's call the target function that we wish to fit $f(x)$. This function defines the probability of finding outcome y when given input x. We could write this in a formula:

$$P(y|x) = \begin{cases} f(x) & for\ y = 1 \\ 1 - f(x) & for\ y = -1 \end{cases} \tag{A.1}$$

Here $P(y|x)$ is the probability of finding y given x. It is equal to $f(x)$ if the event occurred and, since it is a binary event, it is equal to $1 - f(x)$ if the event didn't occur. The statement that we are making in logistic regression is that we can find a linear combination of the input variables and pass it through a sigmoid function to approximate $f(x)$. Finding in this case means defining the weights of the linear combination. We will call the final set of weights the final hypothesis $h(x)$.

$$h(x) = \theta(\boldsymbol{w}^{\boldsymbol{T}}\boldsymbol{x}) \approx f(x) \tag{A.2}$$

A.2 Likelihood

The notion of likelihood is basically an inversion of the usual thought process. What we usually think of when talking about supervised learning is: what is the probability of finding an outcome y given the data x. Here we are going to invert the statement

and define likelihood as: if $h(x) = f(x)$ how likely is it to see the data x given the output y. Or in a formula:

$$P(y|x) = \begin{cases} h(x) & for\ y = 1 \\ 1 - h(x) & for\ y = -1 \end{cases} \tag{A.3}$$

Notice that we now assume that $h(x)$ is generating the outcomes instead of $f(x)$. Remember that we defined $h(x) = \theta(\boldsymbol{w^T x})$, and thus:

$$P(y|x) = \begin{cases} \theta(\boldsymbol{w^T x}) & for\ y = 1 \\ 1 - \theta(\boldsymbol{w^T x}) & for\ y = -1 \end{cases} \tag{A.4}$$

A useful property of the sigmoid function is that $\theta(-x) = 1 - \theta(x)$. We can use this fact to get rid of the cases in the formula, because if we change the second case by $\theta(-x)$ we get:

$$P(y|x) = \begin{cases} \theta(\boldsymbol{w^T x}) & for\ y = 1 \\ \theta(\boldsymbol{-w^T x}) & for\ y = -1 \end{cases} \tag{A.5}$$

Now notice that the sign in each case corresponds to the value of y. We could multiply each case by the corresponding value of y to obtain:

$$P(y|x) = \theta(y\boldsymbol{w^T x}) \tag{A.6}$$

Now we define the likelihood of the dataset D by multiplying the likelihood of each sample:

$$Likelihood(D) = \prod_{n=1}^{N} P(y_n|x_n) = \prod_{n=1}^{N} \theta(y_n \boldsymbol{w^T x_n}) \tag{A.7}$$

We now want to maximize the above likelihood of the dataset, given that the hypothesis $h(x)$ is indeed the correct target.

A.3 Lorem 51

Maecenas dui. Aliquam volutpat auctor lorem. Cras placerat est vitae lectus. Curabitur massa lectus, rutrum euismod, dignissim ut, dapibus a, odio. Ut eros erat, vulputate ut, interdum non, porta eu, erat. Cras fermentum, felis in porta congue, velit leo facilisis odio, vitae consectetuer lorem quam vitae orci. Sed ultrices, pede eu placerat auctor, ante ligula rutrum tellus, vel posuere nibh lacus nec nibh. Maecenas laoreet dolor at enim. Donec molestie dolor nec metus. Vestibulum libero. Sed quis erat. Sed tristique. Duis pede leo, fermentum quis, consectetuer eget, vulputate sit amet, erat.

Appendix B

Cancer staging

Cancer staging is a system that provides a means to indicate to what extent a cancer has developed. This allows for easier communication and the aim is to have a standardized way of classifying cancers. The simplest form of staging assigns a number from 1 to 4 to the cancer. Where stage 1 indicates an isolated cancer of a mediocre size, and 4 indicates a large tumour that has spread throughout the body (called metastasis). Another very popular, more specific staging system is the TNM system.

B.1 TNM staging system

In the TNM staging system a separation is made based on clinical stage and pathological stage. Clinical stage is based on information that can be obtained while the tumour is still present in the patient (scans, bloodtests, biopsy, ...). Pathological stage is based on information gained by microscopically inspecting the tumour after it has been removed from the patient. This process is performed by a pathologist, hence the name.

The TNM staging system uses three combination of a letter and a number. The three letters each point to a specific type of development for the tumour, and the number indicates what the extent is of this development. The three letters are, unsurprisingly, TNM.

- T stands for the reach/extent of the primary tumour
- N stands for the spread of the tumour to regional lymph nodes
- M stands for distant spread to other parts of the body (metastasis)

Each letter is then joined by a number indicating the extent of the development. Unusual situations are indicated by multiple letters however. The possible combinations for each letter and their meaning is shown in table B.1. An example pathologic stage for a cancer could then be $pT1N0M0$ indicating a small primary tumour but no spread to lymph nodes or metastasis.

Code	Meaning
Tx	Tumour cannot be evaluated
Tis	Tumour cannot be evaluated
T0	No signs of a tumour
T1,T2,T3,T4	Tumour present, with increasing size
Nx	Lymph spread cannot be evaluated
N0	No spread to lymph nodes
N1	Spread to small number of nearby lymph nodes
N2	Spread to small number of distant lymph nodes
N3	Spread to multiple distant lymph nodes
M0	No metastasis
M1	Metastasis to distant organs

Table B.1: TNM staging system, meaning of codes

B.2 Pathologic stage in case study 1

In case study 1 (5.3) we are using logistic regression to predict the pathologic stage outcome. In this case the pathologic stage for the patients has been binarized. A value of 1 was given to patients with a TNM pathologic stage of $pT0N0M0$ or $pT0N1M0$. These are basically the two best stage outcomes a patient can get, all other patients were labeled as 0.

Bibliography

[1] Ensemble learning. https://en.wikipedia.org/wiki/Ensemble_learning.

[2] Generalized linear model. https://en.wikipedia.org/wiki/Generalized_linear_model.

[3] Gradient and directional derivative. http://math.oregonstate.edu/home/programs/undergrad/CalculusQuestStudyGuides/vcalc/grad/grad.html.

[4] Gradient descent. https://en.wikipedia.org/wiki/Gradient_descent.

[5] Gradient descent variants. http://sebastianruder.com/optimizing-gradient-descent/.

[6] Hypothesis testing (p-value approach). https://onlinecourses.science.psu.edu/statprogram/node/138; school=PennState Eberly College of Science.

[7] Linear classification. https://en.wikipedia.org/wiki/Linear_classifier.

[8] Linear regression. https://en.wikipedia.org/wiki/Linear_regression.

[9] Logistic regression. https://en.wikipedia.org/wiki/Logistic_regression.

[10] Logistic regression error surface proof of convexity. http://mathgotchas.blogspot.be/2011/10/why-is-error-function-minimized-in.html.

[11] P-value. https://en.wikipedia.org/wiki/P-value.

[12] Proportional hazards model. https://en.wikipedia.org/wiki/Proportional_hazards_model.

[13] Receiver operating characteristic. https://en.wikipedia.org/wiki/Receiver_operating_characteristic.

[14] Sensitivity and specificity. https://en.wikipedia.org/wiki/Sensitivity_and_specificity.

[15] Shiny framework for r. http://shiny.rstudio.com/.

[16] Understanding the gradient. https://betterexplained.com/articles/vector-calculus-understanding-the-gradient/.

[17] Wald test. https://en.wikipedia.org/wiki/Wald_test/.

[18] P. Y. Abu-Mostafa. Machine learning online course. https://work.caltech.edu/telecourse.html, 2012.

[19] M. A. Babyak. What you see may not be what you get: a brief, nontechnical introduction to overfitting in regression-type models. *Psychosomatic medicine*, 66(3):411–421, 2004.

[20] W. C. Black and H. G. Welch. Advances in diagnostic imaging and overestimations of disease prevalence and the benefits of therapy. *New England Journal of Medicine*, 328(17):1237–1243, 1993.

[21] F. S. Collins, E. D. Green, A. E. Guttmacher, and M. S. Guyer. A vision for the future of genomics research. *Nature*, 422(6934):835–847, 2003.

[22] P. R. Cox. *Life Tables*. Wiley Online Library, 1972.

[23] T. G. Dietterich. Ensemble methods in machine learning. In *Multiple classifier systems*, pages 1–15. Springer, 2000.

[24] T. G. Dietterich. Ensemble learning. *The handbook of brain theory and neural networks*, 2:110–125, 2002.

[25] J. Friedman, T. Hastie, and R. Tibshirani. Regularization paths for generalized linear models via coordinate descent. *Journal of statistical software*, 33(1):1, 2010.

[26] M. Goel, P. Khanna, and J. Kishore. Understanding survival analysis: Kaplan-meier estimate. *International journal of Ayurveda research*, 1(4):274, 2010.

[27] K. Hajian-Tilaki. Receiver operating characteristic (roc) curve analysis for medical diagnostic test evaluation. *Caspian journal of internal medicine*, 4(2):627, 2013.

[28] F. E. Harrell, K. L. Lee, and D. B. Mark. Tutorial in biostatistics multivariable prognostic models: issues in developing models, evaluating assumptions and adequacy, and measuring and reducing errors. *Statistics in medicine*, 15:361–387, 1996.

[29] P. T. Hastie and J. Qian. Glmnet vignette. http://web.stanford.edu/~hastie/glmnet/glmnet_alpha.html, 2014.

[30] S. Healthcare. Medical imaging types. http://www.medicalradiation.com/types-of-medical-imaging/.

[31] N. C. Institute. The cancer genome atlas. http://cancergenome.nih.gov/.

[32] R. Kohavi et al. A study of cross-validation and bootstrap for accuracy estimation and model selection. In *Ijcai*, volume 14, pages 1137–1145, 1995.

[33] R. B. Newson et al. Comparing the predictive powers of survival models using harrell's c or somers' d. *Stata Journal*, 10(3):339, 2010.

[34] G. Rodriguez. Survival models. http://data.princeton.edu/wws509/notes/c7.pdf.

[35] P. Royston and A. House. Estimating a smooth baseline hazard function for the cox model. *London: Department of Statistical Science, University College London*, 2011.

[36] A. Sandelin, W. Alkema, P. Engström, W. W. Wasserman, and B. Lenhard. Jaspar: an open-access database for eukaryotic transcription factor binding profiles. *Nucleic acids research*, 32(suppl 1):D91–D94, 2004.

[37] J. Shendure and H. Ji. Next-generation dna sequencing. *Nature biotechnology*, 26(10):1135–1145, 2008.

[38] N. Simon, J. Friedman, T. Hastie, R. Tibshirani, et al. Regularization paths for coxs proportional hazards model via coordinate descent. *Journal of statistical software*, 39(5):1–13, 2011.

[39] B. Smith, M. Ashburner, C. Rosse, J. Bard, W. Bug, W. Ceusters, L. J. Goldberg, K. Eilbeck, A. Ireland, C. J. Mungall, et al. The obo foundry: coordinated evolution of ontologies to support biomedical data integration. *Nature biotechnology*, 25(11):1251–1255, 2007.

[40] C. Stark, B.-J. Breitkreutz, T. Reguly, L. Boucher, A. Breitkreutz, and M. Tyers. Biogrid: a general repository for interaction datasets. *Nucleic acids research*, 34(suppl 1):D535–D539, 2006.

[41] R. Tibshirani et al. The lasso method for variable selection in the cox model. *Statistics in medicine*, 16(4):385–395, 1997.

[42] H. Zou and T. Hastie. Regularization and variable selection via the elastic net. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 67(2):301–320, 2005.

[43] M. H. Zweig and G. Campbell. Receiver-operating characteristic (roc) plots: a fundamental evaluation tool in clinical medicine. *Clinical chemistry*, 39(4):561–577, 1993.

Fiche masterproef

Student: Michiel Ruelens

Titel: Design, implementation and evaluation of data integration methods for biomedical cancer data

Nederlandse titel: Design, implementatie en evaluatie van data integratie methoden voor biomedische kankergegevens

UDC: 621.3

Korte inhoud:

Meer en meer worden we geconfronteerd met grootschalige problemen. Datasets met duizenden variabelen zijn eerder de norm dan de uitzondering en dit is niet anders in de biomedische wereld. We zijn tegenwoordig in staat om het volledige genoom van een patient te sequentiëren, expressie-niveau's van genen te bepalen en proteïnewaarden in het bloed te meten. En door hoogtechnologische instrumenten zoals MRI en PET scanners krijgen we zeer duidelijke beelden van de problemen waar we voor staan. De taak is nu om al deze bronnen van informatie te gebruiken om tot nieuwe inzichten en behandelingen te komen. In dit thesis gebruiken we de technieken van artificiéle intelligentie om predictieve modellen te bouwen. In essentie zoeken deze technieken naar patronen in de gegevens om voorspellingen te kunnen maken over interessante varaiabelen. Om de verschillende grote datasets te gebruiken als input voor deze technieken is het echter nodig een methodiek te definiëren waarop de datasets worden samengevoegd, we noemen dit data integratie en dit vormt de essentie van de thesis. We presenteren drie verschillende integratie methoden die elk de data op een ander niveau integreren. Vervolgens evalueren we de integratie technieken op twee concrete case-studies. Beide studies gebruiken gegevens omtrent darmkanker, bij de eerste studie gebruiken we een logistisch regressiemodel om pathologische respons te voorspellen, bij de andere stellen we een overlevingsmodel op met behulp van cox regressie. Beide studies tonen aan dat de performantie van de modellen afhankelijk is van de gebruikte integratie techniek. Dit geeft aan dat onderzoekers in de toekomst meer aandacht moeten besteden aan de manier waarop gegevens worden geïntegreerd. En dat er onderzoek mogelijk is naar nieuwe integratie technieken.

Thesis voorgedragen tot het behalen van de graad van Master of Science in de ingenieurswetenschappen: computerwetenschappen, hoofdspecialisatie Mens-machine communicatie

Promotoren: Prof. dr. ir. Roel Wuyts
Prof. dr. ing. Olivier Gevaert

Assessoren: Mario Cruz Torres
Prof. dr. ir. H. Blockeel

Begeleiders: Ir. A. Assistent
D. Vriend