Model Selection

Arturo Perez Adroher

July 2018

Author: Arturo Perez Adroher

Supervisor (Rabobank): Pavel Mironchyck

Supervisor (Rabobank): Viktor Tchistiakov

Supervisor (UTwente): Berend Roorda

Supervisor (UTwente): Reinoud Joosten

Abstract

This thesis is on the comparison of model selection algorithms. It builds a framework for relevant algorithms or criteria that have been proposed so far in literature and that lead to a model easy on interpretation and with high predictive power. The properties and performance of these model selection algorithms are studied and compared.

We further propose redeveloped algorithms based on information criteria such as Akaike Information Criterion, Bayesian Information criterion, R^2_{adj} and shrinkage parameters functions like Least Absolute Shrinkage and Selection Operator and Elastic Net to evaluate models' predictive power and interpretation adapted to Rabobank requirements. These algorithms are programmed using open-source Python packages for numerical computation and statistical modelling, and tested with real data supplied by the company.

Keywords— Statistical modelling, model selection, variable selection, unsupervised machine learning, penalized likelihood, model selection information criteria

To my father's patience in teaching mathematics

Contents

List of Tables

List of Figures

1 Introduction

The financial service industry has to deal with several risks associated to their activities and operations in the market. Specifically in the banking sector these risks are divided into eight main categories: credit risk, market risk, operational risk, liquidity risk, solvency risk, foreign exchange risk and interest rate risk (Bessis, 2015). It is crucial for banks to control, measure, monitor and reduce all these risks. Therefore a large part of bank's workforce is specially dedicated to this purpose.

Credit risk, is particularly a critical issue for the financial service industry. It is essential from two points of view: the regulatory and business one (pricing, profitability and risk appetite control). A good estimation of credit defaults has a big impact on companies such as banks, hedge funds and others. This estimation is the result of internal models that attempt to predict the expected loss due to defaults. The models used to forecast and evaluate defaults are: Probability of Default Model (PD), Loss Given Default Model (LGD) and Exposure at Default Model (EAD). In particular the probability of default model gives the average percentage of obligors that default in a rating grade in the course of one year (Flores et al., 2010), the EAD model gives an estimation of the exposure amount that the bank has at an event of a client's default and the LGD model gives the total amount lost at the event of a default.

However, in the Netherlands there are only few banks that are able to build their own models for controlling these risks, while the rest have to comply with the rules and models established by the European Central Bank (ECB). Rabobank is one of the first ones, being allowed to build models themselves. These are developed following ECB regulation and also internal model development procedures. Inside Rabobank, the ALM & Analytics department is responsible for the redevelopment of A-IRB (Advanced Internal Rating-Based Approach) Credit Risk models. Modellers have designed and estimated parameters for models with use of such typical tools as MATLAB, Excel and MSSQL.

Probability of Default, Loss Given Default and Exposure at Default models are built based on data owned by the bank. Data which contain historical and personal information about the clients. The financial sector (due to regulatory issues) is probably one of the most data-rich industries and even more nowadays that bigger amount of data are available due to the evolution of the IT systems. However a great amount of this data is not relevant for the model or is just *noise*. Therefore there is a procedure to determine which variables from all data are relevant (to be included in the model). This process is called **feature selection**, however it can be also encountered in literature as variable selection or model selection and is defined as the technique to choose a subset of variables from an existing set of variables to build a model. In principle there is not a technique that gives the true model[1] for a given set of data. It is to this day an unsolved problem in statistical modelling. Also, what is considered the *true model* goes far beyond statistical modelling because is as complex as reality. Statistical models attempt to understand reality, but reality's complexity most of the times cannot be totally explained with a model. As it has already been said: *all models are wrong, but some are useful* (Box and Draper, 1987).

Model selection arises to tackle three aspects of the modelling world: interpretation, computing time and overfitting. The first one refers to the easiness to read through the model and have a good overview of how data are generated. A model that contains 150 variables is not as easy to read as a model that has 10. Model selection solves this problem by reducing the number of variables from the resulting model. Also, as a consequence of the dimensionality reduction of the model, the computational costs become lower than the ones associated with a model that considers all variables as candidates. The last reason for feature selection is to avoid decrease of predictive power due to high levels of variance in the model, also called overfitting.

One important aspect of model selection is the number of variables (and the problems associated within them, for example correlation) that the modeller considers for the final model. Searching for the best combination of predictors is labor intensive and not particularly an easy job. There are a lot of techniques and algorithms that perform model selection for a given set of data. These have been developed and improved over the years. However, even nowadays statisticians often differ on their criteria and the algorithms to be used. This is sometimes due to the nature of the data but also to personal preferences and intuitions from the scientists about the achievement of an optimum model.

There are different criteria, algorithms, measures, tests and considerations about model selection algorithms. The state of the art will be presented in the next section for the reader to get familiar with the different methods developed over the last

[1]The true model is the one that describes perfectly how data are generated

decades. Therefore there is an important question that must and will be addressed in this thesis: *what are the advantages of certain model selection methods against others.* This question will be reformulated after reviewing in the next section, model selection algorithms proposed so far in literature. For carrying out this comparison, the programming environment of Python is going to be used. This thesis will contribute to the CMLib (Credit Modelling Library) project carried out at Rabobank, that attempts to build a Python library similar to the one existing in MATLAB, to build Credit Risk Models. At this very moment, Python lacks some built-in model selection functions that allows the user to evaluate different models with certain inputs. In this research, these functions are programmed *(See Appendix A)* and tested in order to incorporate them to Rabobank's own library.

Also, additional to this problem there is the one of data, which have to be cleaned and structured to be able to be interpreted by the algorithms that the modeler is using. Data quality is a big issue nowadays in every company that uses data to evaluate business performance and train forecasting algorithms. Data comes from all types of sources and in all types of formats. Thus, there a substantial part of the workforce is dedicated to structure data (data engineers) and maintain the flows of data between different sources to data warehouses. This problem was addressed several years ago, when the information systems started to be able to obtain bigger amount of data from individuals and companies. The data are collected over years, meanwhile IT systems evolve and processes are changing. Also mistakes in administration cannot be completely eliminated. During the research this problem was faced but it will not be treated in this thesis. Nevertheless, it is relevant to mention, that data quality is crucial for credit risk modelling, since it is basically its point of departure.

2 Literature Review

Several statistical methods for model selection have been developed over the years. These methods have the objective of selecting an optimum regression model for a set of data, which is done by selecting a combination of variables (variable/feature selection) from a given set of variables. There has been a lot of research regarding different types of model parameters estimation and precision, and also during the last decades there has been several researchers publishing about model selection. This field of study is really important for statistics. However trying to narrow estimation tools to find the *true* model is not the same as finding an appropriate model (Burnham and Anderson, 2003)

Originally, the quality of a model was assessed by the error in the fit. The better the model fitted the data, the better the model's quality. This is done by the so called ordinary least squared method, developed by Gauss and presented around 1805. More than a century later the statistician Fisher presented to the Royal Society in 1922 (Stigler, 2007) the Maximum Log-Likelihood estimation. This method estimates parameters of a statistical model using the log-likelihood function. This was a crucial moment in the statistical world and the point of departure of several model selection algorithms and criteria. From this moment on, different mathematicians started to develop measures to asses model quality based on the log-likelihood estimation.

Model selection became a hot topic after the realization that using all the variables available to build your model is not always the best option due to model's interpretation and variance. As it was explained already in the introduction, the first one refers to the easiness to read through the model and have a good overview of how data is generated. Model selection solves this problem by reducing the number of variables in the model. The last reason for feature selection is to avoid decrease of predictive power due to high levels of variance in the model.

In 1960, Efroymson (1960) proposed an algorithm that recursively adds and deletes variables from a model. This algorithm, called stepwise regression, is a combination of two simpler algorithms: forward selection and backward elimination. Forward and backward are simple algorithms that perform variable selection by including or excluding (respectively) variables from the model. Both begin with an initial model (no model - full model) and apply a selection procedure under a certain criterion.

Stepwise regression is a combination of both algorithms considering in each step, variables for inclusion and exclusion simultaneously. Its methodology is the inclusion and exclusion of variables from the model till an optimum model is obtained under a certain criterion. This criterion can vary depending on the feeling or intuition of the statistician. This method of variable selection has been criticized by Flom and Cassell (2007) due to the limitations of the model in relation to the fitting method.

In 1973 the American Statistical Association published a paper from C.L. Mallows that introduced C_p, which is a measure to assess model fit (Mallows, 1973). C_p is based on the Least Squared Error method, but also considers the sample size and the number of variables used for fitting the data. This shows that the overfitting problem was already being taken into account, and the focus of statistical modelling was also considering the problem of prediction. Mallow's paper did an investigation in variable selection till that time and came up with a measure that served to compare models that fitted the same data but with different set of variables.

Only one year later in 1974, Hirotugu Akaike published a paper in which he presented a new model quality measure based on the maximum likelihood estimation. This new estimate was called the Akaike Information Criterion (AIC) (Akaike, 1974). It takes into account the number of variables in the model and the log-likelihood estimates. Akaike defined the model with the lowest AIC to be the best from the set of models considered for fitting the data. Interestingly, the Akaike Information Criterion is equivalent to the Mallow's C_p when the residuals of the model in linear regression are considered to follow a normal distribution with mean 0 (and standard deviation σ). This was demonstrated in 2014 in a paper published in the International Statistical Review by Boisbunon et al. (2014).

Four years later, in 1978, G. Schawrz presented in his paper Schwarz et al. (1978) a slightly different measure than the AIC for model's quality assessment. This measure was derived to serve as an asymptotic approximation to a transformation of the Bayesian posterior probability of a candidate model (Neath and Cavanaugh, 2012). This statistical tool is called the Bayesian Information Criterion (BIC) and its calculation is rather similar to the AIC (also based in the log-likelihood estimator). Still, there are some recent discussions as shown in the papers by Burnham and Anderson (2004) and Vrieze (2012), that discuss the advantages and disadvantages of using BIC instead of the AIC for model fit assessment. Essentially, both measures are based on the log-likelihood of the model but they apply a different penalty for larger models.

The penalty from the AIC is $2p$ and the penalty for the BIC is $ln(n)p$ (being n the number of samples and p the number of variables). A simulation study done by Vrieze (2012) concludes that even if the *true model* is in the candidate set the AIC performs better because it selects the model that is closer to the *true model*. Another study done by Yang (2005) shows that when the assumption that the *true model* is not in the possible models set, the AIC performs better in finding the best possible model. On the contrary, another study reflected in the book Burnham and Anderson (2003) says that theoretically BIC will select the true model with higher probability than the AIC, if the true model is in the model set.

All of these measures, C_p, AIC and BIC are attempts to asses a score to the model to be able to compare models that have been fitted maximizing the log-likelihood function to find the best variable estimates. In model estimation, the parameters of the resulting model are obtained by minimizing the error between the predicted values and the observed values. In literature sometimes the resulting function to be minimized in order to maximize log-likelihood is called cost function. This is the result of the maximization of the log-likelihood function (before mentioned), however it results in a different maximization depending on the regression nature (linear, logistic...). Therefore the model selection criteria by comparing C_p, AIC and BIC don't affect the fitting process of the parameters, but instead they apply a penalization to the already fitted model to asses model quality.

However, a different approach used for model selection directly applies to the maximization function of the log-likelihood function. It all began with the introduction of the concept of regularization to tackle the problem of the instability associated with least squared method due to correlation among variables (Hoerl and Kennard, 1970). Theoretically, if the correlation among variables is 0 (orthogonal relationship), the Gauss-Markov estimation procedure would be very suitable. However this is rarely the case, since correlation between variables is frequent in real life. This causes the estimation of the model by maximizing the log-likelihood function usually overfit the data. This naturally results in a relatively lowly biased model but with a poor prediction power. To solve this problem, in 1962 A. E. Hoerl presented paper in which he suggested the inclusion of an element in the regression problem in order to control the instability associated with least squared method due to overfitting. He introduced Ridge Regression. The basic idea is to shrink the parameters size in order to reduce the overfit and increase the predictive power by adding a regularization parameter.

This regression method would penalize the coefficients of the regression making them lower in absolute value than the equivalent OLS[2] regression. As a consequence the regression function does not stick that aggressively to the data, like OLS. Nevertheless, Ridge does not perform feature selection. It shrinks the variables but it always considers all variables for the final model. This was not useful for the analysis framework of model selection, because even though the model's variance was reduced, the model's interpretation was not.

Following this idea of modifying the objective function, during the 90's and the beginning of the new century, several developments were achieved in the field of model selection. In 1995, a study by Leo's Breiman called non-negative garrote was published (Breiman, 1995), which proposed the calculation of the minimization of the error function including the term $2\lambda \sum_{j=1}^{p} \mu_j$ (being μ the parameters vector and λ the penalty coefficient) subject to $\mu_j \geq 0, j = 1, ..., p$. This is similar to Ridge Regression but using a different criterion. This way the coefficients shrink but also get eliminated while with Ridge this was not possible. This paper was the main influence to Robert Tibshirani (Tibshirani, 2011), who in 1996 improved (solving for the p≥N situation, where there are more features than the samples) the theorem and proposed what he called the Lasso regression. In the paper where he presented his work, Tibshirani (1996) proposed a variation of Ridge in which he basically used the same minimization function but subject to a different constraint. Doing this Lasso performed variable selection (solving Ridge problem). However, Lasso had some practical and conceptual problems at that time. The first one was that the algorithms for the Lasso were difficult for modeler's interpretation. Also, Lasso was performing poorly with correlations and there was not an existing tool for fast statistical computation that allowed to run Lasso in a set of data (Tibshirani, 2011). It was because of this that Lasso regression did not become popular until the development of the LARS algorithm in 2002 which gave an efficient way to calculate Lasso (Tibshirani, 2011) with a similar procedure as forward regression. In the following years, a lot of variations of Lasso were published: Grouped Lasso (Yuan and Lin, 2006), Fused Lasso (Tibshirani et al., 2005), Adaptive Lasso (Zou, 2006), Graphical Lasso (Yuan and Lin, 2007), Dantzig Selector (Candes et al., 2007), Near Isotonic Regularization (Tibshirani et al., 2011), Matrix Completion (Candès and Tao, 2010), Compressive sensing (Donoho, 2006) and Multivariate Methods (Jolliffe et al., 2003).The most relevant among them was the Elastic Net by Zou and Hastie (2005), which is a combination of pure Lasso and Ridge

[2]Ordinary Least Squared method

regression.

In parallel way and with a different philosophy of model building, the same author that proposed the Non-Negative Garrote (Leo Breiman) worked on other algorithm based on Ho (1995) called *Random Decision Forests* that extended the model selection framework. In his paper (Breiman, 2001) he explained how to rank variable importance for a given set of variables. However, it was not till 2013 when this algorithm was developed in a programming language (Liaw, 2013). Also, another variant of model selection was developed by Anders and Korn (1999) using *Artificial Neural Networks* (invented by McCulloch and Pitts (1943)) in order to choose a model that fit the same dataset.

These two methods: *Random Decision Forests* and *Artificial Neural Networks* have become very famous in the field of modelling over the last years. Generally they achieve a higher accuracy than the traditional methods. However, the problem is that the resulting model lacks of easiness in interpretation. Several authors refer as the resulting models from both methods as a *Black Box*. Therefore, some modelers prefer Lasso, Elastic Net, AIC, BIC procedures in order to choose a model, because even if the model's accuracy is sometimes lower, the model is able to be interpreted and explained to the stakeholders involved in the final model.

One year after the release of Random Forests for R[3], other approaches for model selection in the field of medical treatment such as the treatment effect cross-validation (Rolling and Yang, 2014) were proposed. This method used a TECV statistic to asses model's quality (same philosophy as AIC, BIC, ...) based on a set of candidates model that fit the same dataset. The candidate models are constructed using different splits among the same data. The complete explanation can be found in the paper before mentioned. Only two years later a new statistic measure was introduced by Resende and Dorea (2016) called the Efficient Determination Criterion (EDC) that is a generalization of the Akaike Information Criterion and Bayesian Information Criterion.

Taking into account the state of art it is very interesting to carry out an investigation about the efficiency and convergence of different model selection methods for a given set of data. However, an analysis framework is necessary due to the different methodology of each model selection method. From the scope of study, algorithms that

[3]Software for statistical modelling

do not result in: a reduction of variables selected (Ridge Regression) or easiness on interpretation (Random Forests, Neural Networks...) are excluded from the analysis framework. In the next section this framework will be developed for getting deeper into the inspection of the model selection methods.

2.1 A framework for model selection algorithms

In this section we present a table in which all model selection algorithms and criterion mentioned in the last section are classified. The table is sorted by year of release of the algorithm and supports the research question:

Model Selection Method	**Year**	**Criteria**	**Mathematical Optimization**	**Reference**
Log - Likelihood	1922	Comparison	$argmax\left(\sum_{i=1}^{n}\log p(y^i\|x^i;\beta_0,\beta_1,...,\beta_p)\right)$	-
Forward Selection	1960	Comparison	*See Section 5*	(Efroymson, 1960)
Backward Selection	1960	Comparison	*See Section 5*	(Efroymson, 1960)
Stepwise Regression	1960	Comparison	*See Section 5*	(Efroymson, 1960)
Ridge Regression	1970	Optimization	$min_{\beta,\beta_0}\left\{\frac{1}{N}\left(y_i-\beta_0-x_i^T\beta\right)^2\right\}$ subject to $\sum_{j=1}^{p}\beta_j^2\leq t$	(Hoerl and Kennard, 1970)
C_p	1973	Comparison	$C_P=\frac{1}{\hat{\sigma}^2}RSS_P-n-2p$	(Mallows, 1973)
AIC	1974	Comparison	$AIC=2k-2ln(\hat{L})$	(Akaike, 1974)
BIC	1978	Comparison	$BIC=ln(n)k-2ln(\hat{L})$	(Schwarz et al., 1978)
Non-Negative Garrote	1995	Optimization	$min\left\{\sum_{i=1}^{N}\left(y_i-\sum_j c_jx_{ij}\beta'_j\right)^2\right\}$ subject to $c_j\geq 0$, $\sum_{j=1}^{p}c_j\leq t$	(Breiman, 1995)
Lasso	1996	Optimization	$min_{\beta,\beta_0}\left\{\frac{1}{N}\left(y_i-\beta_0-x_i^T\beta\right)^2\right\}$ subject to $\sum_{j=1}^{p}\|\beta_j\|\leq t$	(Tibshirani, 1996)

Table 1: Model Selection Methods.

Model Selection Method	**Year**	**Criteria**	**Mathematical Optimization**	**Reference**
LARS	2002	Optimization	Co-ordinate descent for Lasso	(Trevor et al., 2002)
Multivariate Methods	2003	Optimization	Sparse PCA, LDA and CCA	(Jolliffe et al., 2003)
Fused Lasso	2005	Optimization	$\lambda \sum \lvert\beta_{j+1} - \beta_j\rvert$	(Tibshirani et al., 2005)
Elastic Net	2005	Optimization	$\lambda_1 \sum \lvert\beta_j\rvert + \lambda_2 \sum \beta_j^2$	(Zou and Hastie, 2005)
Grouped Lasso	2006	Optimization	$\sum_g \lvert\lvert\beta_g\rvert\rvert_2$	(Yuan and Lin, 2006)
Adaptative Lasso	2006	Optimization	$\lambda_1 \sum w_j \lvert\beta_j\rvert$	(Zou, 2006)
Comprehensive Sensing	2006	Optimization	$\min(\lvert\beta\rvert_1)$ subject to $y = X\beta$	(Donoho, 2006)
Graphical Lasso	2007	Optimization	$loglik + \lambda\lvert \sum^{-1}\rvert_1$	(Yuan and Lin, 2007)
Dantzig Selector	2007	Optimization	$min \left\{X^T (y - X\beta) \lvert\lvert_\infty\right\} \lvert\lvert\beta\rvert\rvert_1 < t$	(Candes et al., 2007)
Matrix Completion	2010	Optimization	$\lvert\lvert X - \hat{X}\rvert^2 + \lambda\lvert\lvert\hat{X}\rvert\rvert_*$	(Candès and Tao, 2010)
Near Isotonic Regularization	2011	Optimization	$\sum (\beta_j - \beta_{j+1})_+$	(Tibshirani et al., 2011)
Random Forests in R	2013	RF Comparison	See Literature	(Liaw, 2013)
Treatment Effect Cross Validation	2014	Optimization	$TECV_\pi \left(\hat{\triangle}_{n_1,k}\right) = \sum_{j=1}^{\overline{n}_2} W_j \left\{\overline{\delta}_j - \hat{\triangle}_{n_1,k}(U_{j_t})\right\}^2$	(Rolling and Yang, 2014)
Efficient Determ. Criterion	2016	Comparison	$EDC(k) = -2 \log \hat{L}(k) + \gamma(k)c_n$	(Resende and Dorea, 2016)

Table 2: Model Selection Methods.

2.2 Research Question

Having a good overview of the different model selection algorithms and also on how do the perform the selection we formulate the research question for this thesis, based on model selection methods that result in ease of interpretation and less number of variables. Therefore, as mentioned before, algorithms like Ridge Regression (L_2 penalization) that does not perform variable selection or Random Decision Trees and Neural Networks that lead to a black box, will not be considered for study. Taking this into account our research question will be:

What are the differences between the final models achieved via different model selection algorithms (stepwise, BIC, AIC, Lasso and Elastic Net) and why?

3 Methodology

In this section the methodology followed in this thesis is explained. It is very important to consider that for an understanding of model selection methods, the reader must have a basic knowledge about regression methods (**model estimation**). In *Section 4* the most common model estimation techniques will be presented. This section is mainly for the reader to get familiarized with the regression procedures and concepts such as the difference between linear and logistic regression, the log-likelihood estimation and gradient (coordinate) descent algorithms. However, if the reader is familiarized with **all** of these methods can may consider to skip this section, and jump to *Section 5*. This section will present the proposed framework for model selection procedures with their respective explanation.

Afterwards, in *Section 6* three model validation methods are going to be presented. Model validation basically consists on the calibration of a model with out of sample data[4].

Section 7 will present the model selection procedures chosen for analysis.These methods will be slightly modified from their original settings in order to comply with certain requirements of Rabobank's internal framework models. The intuition behind these modifications will be explained as well as the original algorithms. This way the reader will understand clearly what the purpose of these modifications is and what is expected from the results. The methods will gather estimation, selection and calibration concepts previously explained in *Sections 4,5* and *6*.

These four algorithms are going to be trained for a specific set of confidential data from Rabobank in *Section 8*. The data have been provided to the author while carrying out the research inside the company. Therefore, the name of the different variables and the nature of their behaviour will not be mentioned nor explained. However, the results from these simulations are going to be presented and compared. From these results, conclusions and recommendations will be extracted in *Sections 9,10* and *11* respectively.

The main contribution of this research will be on the comparison between the performance of the model selection algorithms to be studied. They will be tested

[4]Out of sample data are data that have not been used for training the model, but for checking how the model performs on new data. Results can be used for calibration.

with different input types in order to prove which of them are more or less suitable for model selection with this data. Also we will study which algorithm outperforms the others and which is the one most recommended to use for feature selection in Credit Risk Modelling. These contributions will be highly relevant for modelers in the financial industry, but also for other sectors of the industry that want to improve the accuracy and quality of their predictive models (demand forecast, weather forecast, sales forecast,...etc).

4 Model Estimation

4.1 Introduction

A mathematical model is a language that aims to describe the behaviour of a certain system in order to be able to understand how data are generated. This language is composed by different variables that are estimated from an exiting set of data (training set). The estimation procedure depends highly on the nature of the data. In this chapter we will treat two types: linear regression and binary classification.

Linear regression estimates the output of the model as a linear function. Examples of the applications of linear regression can be found on real estate price estimation, demand forecasting and more. On the other hand, classification estimation identifies the category from a set of categories from a model that a new observation belongs to. Real life examples of classification problems are pattern recognition, illness prediction and more.

This chapter will present both of the estimation procedures and the way the are estimated. After reading this chapter the reader will get and understanding of the key concepts necessary for the understanding of model selection implementation.

4.2 Linear Regression

Linear regression is a mathematical modelling process in which a relationship is established between and output y^i (where i is the observation from 0 to n begin n the last observation) and a set of explanatory variables $x_0^i, x_1^i, ..., x_p^i$ (where p is the number of features). The variable y^i is called dependent or response variable (Seber and Lee, 2012) and $(x_0^i, x_1^i, ..., x_p^i)$ explanatory or independent variables (Seber and Lee, 2012). Each sample i: $x_0^i, x_1^i, ..., x_p^i$ can contain more than one feature. For example if we want to predict the price of a car we could use the brand, speed limit, media system, ...etc. Therefore we will encounter data that have the following form:

$$\begin{pmatrix} x_{0,0} & x_{0,1} & \cdots & x_{0,p} \\ x_{1,0} & x_{1,1} & \cdots & x_{1,p} \\ \vdots & \vdots & \ddots & \vdots \\ x_{n,0} & x_{n,1} & \cdots & x_{n,p} \end{pmatrix}$$

where n is the number of samples (also called observations) and p is the number of variables to be considered (brand, speed limit, media system). Each row is denoted by $(x^0, x^1, ..., x^n)$, and each column $(x_0, x_1, ..., x_p)$ Also, we will have the column of outputs (dependent variable) y:

$$\begin{pmatrix} y^0 \\ y^1 \\ \vdots \\ y^n \end{pmatrix}$$

that denotes the ouput.

The relationship established by linear regression can be represented as:

$$h\left(x_0, x_1, ..., x_p\right) = \beta_0 x_0 + \beta_1 x_1 + ... + \beta_p x_p + \epsilon \tag{1}$$

In this equation it can be seen that the output Y (dependent variable) is the result of the multiplication of a set independent variables $(x_0, ..., x_p)$ by a set of coefficients plus an error term. The objective of the regression is estimating the set of coefficients called *Beta*, for when new data is acquired an approximate output can be approximated with the function *h(..)*. In matrix form:

$$h_\beta\left(x\right) = \begin{bmatrix} \beta_0 & \beta_1 & ... & \beta_p \end{bmatrix} \begin{bmatrix} x_0^i \\ x_1^i \\ ... \\ x_p^i \end{bmatrix} = \beta^T X + \epsilon^i, \tag{2}$$

where ϵ^i is the estimation error on sample i

For getting a visual representation we will use the *diabetes* data provided by the Python built in library *sklearn*:

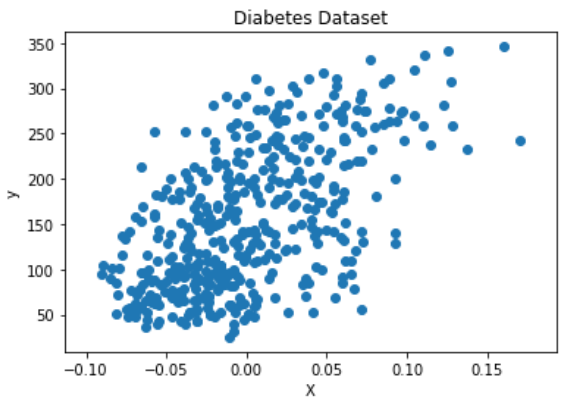


Figure 1: Diabetes Dataset.

This image shows a set of data. The objective of the linear regression is obtaining a linear function *h(x)* that best approximates the data. For example:

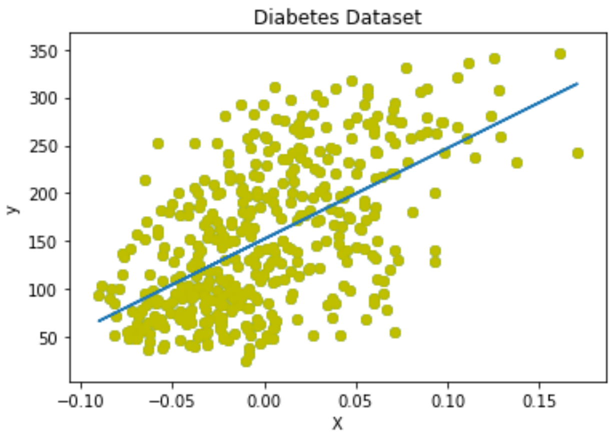


Figure 2: Diabetes Dataset linear regression.

In this graph the blue line represents the function *h(x)* that was mentioned before.

This case, as it can be seen, is rather simple. The regression only estimates two variables, a constant (β_0) and a coefficient (β_1). However, as it will be seen later, in real-life we will play with many more variables. The method by which these two parameters (constant and coefficient) are estimated is by the maximization of the log-likelihood function. This is explained in the following subsection.

Log-Likelihood

The maximum likelihood estimate is a statistical measure that is the result of the maximization of the likelihood function. This maximization does not necessarily assume that the parameters are under a prior distribution, but instead, they assume that their estimates do. Given a prediction for the sample i we have that:

$$h_\beta\left(x_0^i, x_1^i, ..., x_p^i\right) = \beta_0 x_0^i + \beta_1 x_1^i + ... + \beta_p x_p^i + \epsilon^i \tag{3}$$

In this equation X and Y do not necessarily have a specific distribution. Being ϵ^i the residual of each data point $(x_0^i, x_1^i, ..., x_p^i, y_i)$ independent from each other and following $N\ (0, \sigma^2)$, and assuming the linearity of the function $h_\beta(x)$. Then the likelihood function (having p, number of explanatory variables and n number of samples) given the dataset $x_0^i, x_1^i, ..., x_p^i$ where i is the observation from 0 to n begin n the last observation:

$$L(\beta|x) = p_\beta(x) = \prod_{i=1}^{n} p(y^i|x^i; \beta_0, \beta_1, ..., \beta_p, \sigma^2) \tag{4}$$

Assuming that the distribution of the residuals ϵ^i:

$$L(\beta|x) = \prod_{i=1}^{n} \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{\left(y^i - h_\beta(x^i)\right)^2}{2\sigma^2}} \tag{5}$$

For the convenience of calculations, the logarithm of this function is taken for its maximization (being its maximum at the same point as the likelihood function). This is mainly because it is easier to take the derivative of a sum than the derivative of a product:

$$\log L(\beta|x) = \log \prod_{i=1}^{n} p(y^i|x^i; \beta_0, \beta_1, ..., \beta_p) \tag{6}$$

Using basic logarithm properties this equation can be written as:

$$\log L(\beta|x) = \sum_{i=1}^{n} \log p(y^i|x^i; \beta_0, \beta_1, ..., \beta_p) \tag{7}$$

And then its maximization:

$$argmax \left(\sum_{i=1}^{n} \log p(y^i|x^i; \beta_0, \beta_1, ..., \beta_p) \right) \tag{8}$$

When the prior probability distribution of the estimates is taken under the assumptions mentioned before is assumed , then solving the previous equation:

$$argmax \left(\sum_{i=1}^{n} \log \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{\left(y_i - h_\beta(x_i)\right)^2}{2\sigma^2}} \right) = \tag{9}$$

$$argmax \left(\frac{n}{2} \log 2\pi - n \log \sigma - \frac{1}{2\sigma^2} \sum_{i=1}^{n} (y_i - h_\beta(x_i))^2 \right) \tag{10}$$

Which leads to the minimization (over β) of:

$$\sum_{i=1}^{n} \left(y^i - h_\beta(x^i)\right)^2 \tag{11}$$

The result of this equation is a set of parameters (or coefficients) $(\beta_0, \beta_1, ..., \beta_p)$ that define the regression function $h_\beta(x)$. *Equation 11* shows the **mean square error** of the fitted data is what is trying to be optimized. This function is also commonly called **cost function**. Optimizing this function lead to the values of β that lead to a lower mean square error
Therefore after all we can conclude that for linear regression with two or more variables, the estimation of the coefficients is obtained by the maximization of the log-likelihood function, that if certain assumptions are taken, lead to the minimization of the mean squared error (cost function).

Cost function and gradient/coordinate descent algorithms

The cost function is a function which value can be interpreted as a measure of fit of the model. It measures the mean square difference between the estimations and the real values of the data. This function is generally written as:

$$J\left(\beta_0, \beta_1, ..., \beta_p\right) = \frac{1}{2m} \sum_{i=1}^{m} \left(h_\beta(x^{(i)}) - y^{(i)}\right)^2 \tag{12}$$

It is directly intuitive that the lower the cost function the better the line fits the data. Therefore, the objective of the linear regression will be:

$$argmax_{\beta_0,\beta_1,...,\beta_p} \left(\frac{1}{2m} \sum_{i=1}^{m} \left(h_\beta(x^{(i)}) - y^{(i)}\right)^2\right) \tag{13}$$

As an example, *diabetes* data set will be used. In the case only two variables were considered, a constant (β_0) and a coefficient (β_1). The cost function in this case will be:

$$J\left(\beta_0, \beta_1\right) = \left(\frac{1}{2m} \sum_{i=1}^{m} \left(h_\beta(x_i) - y_i\right)^2\right) \tag{14}$$

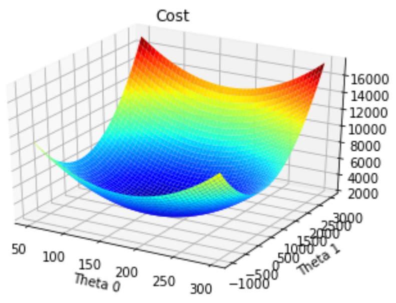


Figure 3: Cost function.

As it can be seen in the *Figure 3*, the cost function has minimum and maximum values. There are various algorithms that find the minimum value of the cost function. The most common is called **gradient descent**. This algorithm is very intuitive, it performs a sequence of iterations that updates β for values in which the cost function is lower. Roughly speaking it is as if we drop a ball in a point of the function and watch where the ball ends up. The iterative process for the *diabetes* dataset would be:

$$\beta_0 := \beta_0 - \lambda \frac{\partial}{\partial \beta_0} J\left(\beta_0, \beta_1\right) \tag{15}$$

$$\beta_1 := \beta_1 - \lambda \frac{\partial}{\partial \beta_1} J\left(\beta_0, \beta_1\right) \tag{16}$$

In this equation it can be seen that the parameters β_0 and β_1 are being **learned** to find the minimum value of the cost function. In the equation the value λ is the distance of each step taken in the update. The value of λ must be chosen carefully. Large values of λ could lead to miss the cost function minimum. Normally, built-in functions use a small value of λ. The learning process can be visualized in the 3D:

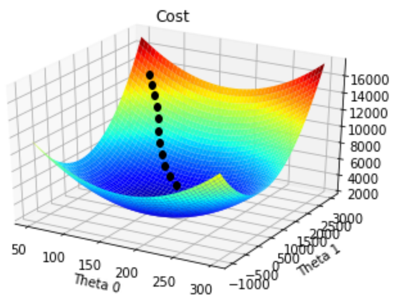


Figure 4: Gradient descent.

In a contour plot:

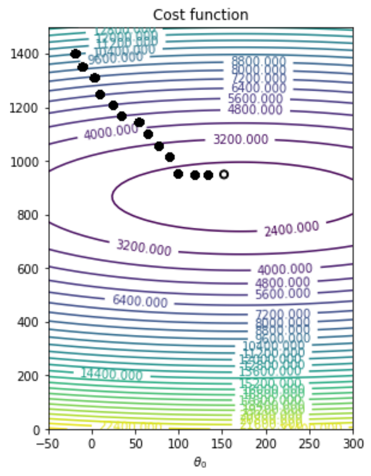
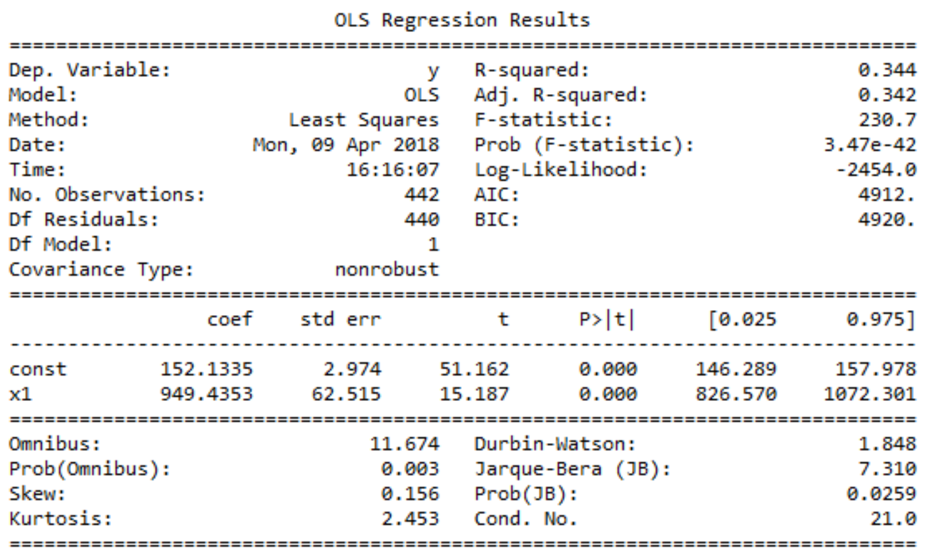


Figure 5: Contour plot.

Naturally, the gradient descent algorithm can be generalized to more than two variables. Doing so, gradient descent is applicable to a regression with more than two coefficients to estimate. The outcome of its implementation is the optimum value of the coefficients considered for the model. For the case of the diabetes dataset the resulting coefficients will be:



```
                            OLS Regression Results
==============================================================================
Dep. Variable:                      y   R-squared:                       0.344
Model:                            OLS   Adj. R-squared:                  0.342
Method:                 Least Squares   F-statistic:                     230.7
Date:                Mon, 09 Apr 2018   Prob (F-statistic):           3.47e-42
Time:                        16:16:07   Log-Likelihood:                -2454.0
No. Observations:                 442   AIC:                             4912.
Df Residuals:                     440   BIC:                             4920.
Df Model:                           1
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
const        152.1335      2.974     51.162      0.000     146.289     157.978
x1           949.4353     62.515     15.187      0.000     826.570    1072.301
==============================================================================
Omnibus:                       11.674   Durbin-Watson:                   1.848
Prob(Omnibus):                  0.003   Jarque-Bera (JB):                7.310
Skew:                           0.156   Prob(JB):                       0.0259
Kurtosis:                       2.453   Cond. No.                         21.0
==============================================================================
```

Figure 6: Results from Linear Regression on Diabetes Dataset.

The results just shown give the coefficients necessary to get the prediction function $h_\beta(x)$ (see *Figure 2*, the blue line):

$$h_\beta(x) = \beta_0 + \beta_1 x_1 = 152.1335 + 949.4353x \tag{17}$$

However, there are other alternatives than gradient descent. Another method commonly used is the so called **coordinate descent**. Coordinate descent performs the minimization of the cost function across its coordinates till it finds a minimum value of the mean square error. The built-in sklearn library uses coordinate descent for their cost function minimization.

4.3 Logistic Regression

Logistic regression is a mathematical modelling process in which a relationship is established between an output Y (which is a logical dependent variable, Boolean 0 or 1) and a set of explanatory variables. The variable y^i is called dependent or response variable Seber and Lee (2012) and $(x_0, x_1, ..., x_p)$ explanatory or independent variables Seber and Lee (2012). The main difference between linear and logistic regression is that the output variable is discrete. I takes values: 0, 1, 2 ,... etc. In this section we will consider only **binary classification problem**, in which the output is either *1* or *0*. The reason for this is that the models we are going to study in this thesis are PD Models which outputs are *0: Non-Default* or *1: Default*. Also it can serve for Cure Models with *0: Not cured* or *1: Cured*. In the following example we consider a dataset that contains data from different eyes, which final output is the eye colour (green or blue):

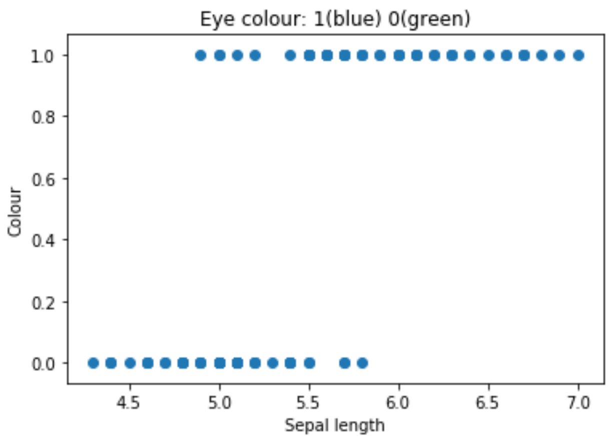


Figure 7: Iris dataset.

The previous image shows how data is classified in two sets (0 and 1). Intuitively attempting classification using linear regression is a bad idea because the output does not take values higher than one (in binary classification problems). Therefore the *linear regression function*:

$$h\left(x_0, x_1, ..., x_p\right) = \beta_0 x_0 + \beta_1 x_1 + ... + \beta_p x_p, \tag{18}$$

is plugged into the Sigmoid function ($g(h_\beta(X))$):

$$g(h_\beta(X)) = \frac{1}{1 + e^{-h_\beta(X)}}. \tag{19}$$

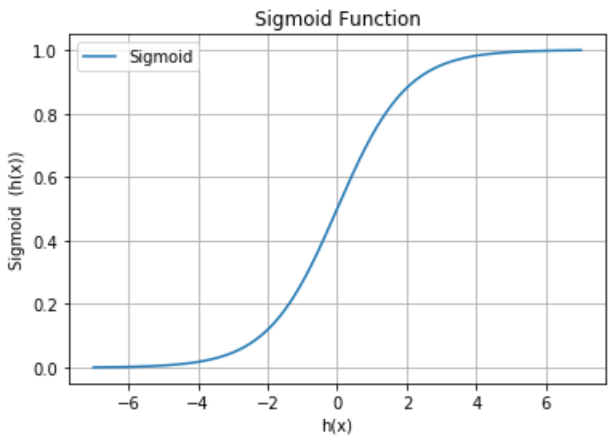


Figure 8: Sigmoid Function.

This function gives the probability that the output is 1. So basically, logistic regression is a linear classifier. Therefore if we consider the transition from linear to logistic this would be:

$$h_\beta(x)^{linear} \Rightarrow g(h_\beta(x)^{linear}) \tag{20}$$

The derivation of the log-likelihood changes for logistic regression. This is due to the nature of the distribution of the residuals and the regression nature type. As it can be seen in *Equation 18*, there not normal distribution residuals like in linear regression. This is due to the prior distribution we assume that the output has. Nevertheless, the parameters are estimated maximizing the likelihood function, as in linear regression.

Log-likelihood

First of all, we must notice that logistic regression predicts probabilities. The likelihood then is

$$L(\beta_0, \beta_1, ..., \beta p) = \prod_{i=1}^{n} p(y^i | x^i; \beta_0, \beta_1, ..., \beta_p), \tag{21}$$

and taking into account the distribution of this probabilities (Bernoulli) we have that

$$L(\beta_0, \beta_1, ..., \beta p) = \prod_{i=1}^{n} p(x^i)^{y^i} (1 - p(x^i))^{1-y^i}, \tag{22}$$

and taking logarithms to make the maximization easier we have that

$$\log(L(\beta_0, \beta_1, ..., \beta_p)) = \sum_{i=1}^{n} y^i \log p(x^i) + (1 - y^i) \log((1 - p(x^i))). \tag{23}$$

The maximization of the log-likelihood (which is the same as maximization of likelihood) is done in order to obtain the parameters that best fit the data. Having that:

$$p(x^{(i)}) = \frac{e^{\beta_0 + \beta_1 x_1^{(i)} + ... \beta_p x_p^{(i)}}}{1 + e^{\beta_0 + \beta_1 x_1^{(i)} + ... \beta_p x_p^{(i)}}} = \frac{1}{1 + e^{-(\beta_0 + \beta_1 x_1^{(i)} + ... \beta_p x_p^{(i)})}} = g(h_\beta(x)) \tag{24}$$

Therefore the cost function of logistic regression is slightly different than the one from linear regression, but the underlying idea is the same, a measurement of the error between the estimations and real values of data:

$$J(\beta_0, \beta_1, ..., \beta p) = -\frac{1}{n} \sum_{i=1}^{n} \Big[y^{(i)} \log(g(h_\beta(x^{(i)})) + (1 - y^{(i)}) \log(1 - g(h_\beta(x^{(i)})) \Big] \tag{25}$$

The minimization of this function is done the same way as in linear regression, using coordinate descent or gradient descent. The result of this minimization are the parameters of the vector $(\beta_0, \beta_1, ..., \beta_p)$ that define the regression logistic model.

4.4 Conclusion

In this section we have made an introduction to the two main types of regression methods: linear and logistic. Each of them serves to a different estimation purpose and therefore fitted in a different way. The implementation of these regression types has been summarized for the reader to get familiarized with the implementation methods such as gradient descent and coordinate descent. Also it is important to get the idea of the derivation of the log-likelihood function to obtain the cost function for each of the estimation types. This is so, because the model selection algorithms developed in *Section 7* will use these equations and concepts.

5 Model selection

5.1 Introduction

In the last section, two main types of regression were presented as well as some concepts used for their estimation. This has to be the first step when modelling: identifying the nature of your data. Secondly, a modeler must think carefully about the purpose of the model: explanatory (fit) or prediction. Paradoxically, a model that has a very good fit on the existing data may not be the best model for prediction. This is issue has been addressed several times in statistical modelling. One of the reasons for this to happen is overfitting.

Overfitting occurs when the regression models sticks too much to the data used for the model. It is illustrated in the following picture:

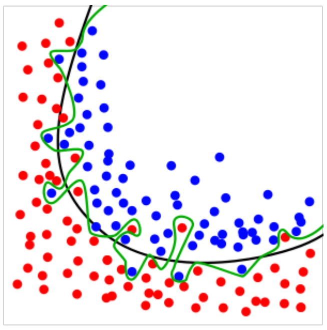


Figure 9: Overfitting (*source: Wikipedia*).

The image shows a binary classification regression fitted by two different models. Green model fits the data closely and this results in a high explanatory power. However it leads to a higher error on prediction with new data samples. This means that the green model has a low bias (close to reality) but a high variance (poor predictive power). On the contrary the black model is not that close to the existing data (meaning high bias) but has a better predicting power (low variance).

It must be noted from the model's shape that the green model has higher complexity (higher number of variables) than the black model. This means that the green model uses a higher number of explanatory variables for its estimation and also that there is a relationship between model complexity, variance and bias.

Intuitively we can think of variance and bias as error measures of our model. In the previous section we introduced the *Cost Function* expressed as the error associated with our model regarding the observations and predictions. This error was calculated in a different way for logistic and linear regression. In fact, this error can be decomposed into variance and bias. The mathematical demonstration can be found in Sammut and Webb (2017) and it will not be shown here. The result of the decomposition of the *total error* is:

$$E[Error] = Var[noise] + bias^2 + Var[y_i] = \sigma^2 + bias^2 + Var[y_i] \tag{26}$$

There is a direct relationship between model's complexity and total error (as a sum of variance and bias). This relationship mentioned previously in this section can be plotted as follows:

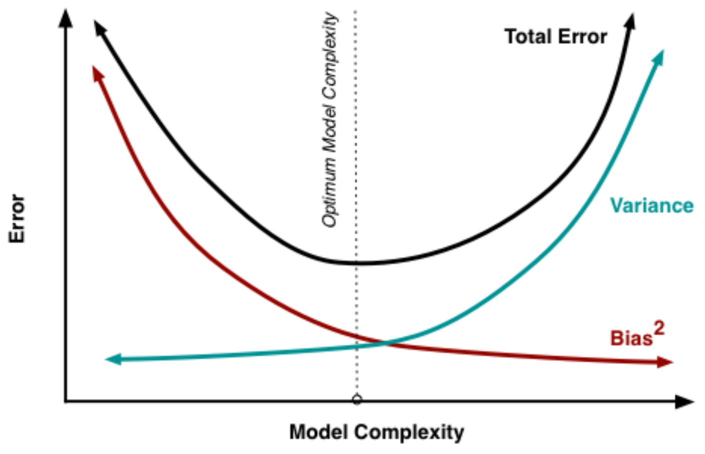


Figure 10: Variance Bias Trade-off (Fortmann-Roe, 2012).

Therefore, there is an optimum model complexity (number of variables considered for the model) that reaches the lower total error. This is achieved by selecting some variables from the set of available variables for inclusion into the final model.

The need for model selection

As it was previously explained, the purpose of this thesis is studying the efficiency of the algorithms that build predictive models. Taking into account what was mentioned previously in this section, these models must avoid overfitting achieving an optimum model complexity. This means that we must select some of the variables available from a dataset to include in the final model. This procedure can be automatized using mathematical algorithms. In this section we build a classification framework among these algorithms and explain those that we are going to use in *Section 7*

5.2 Model Selection by comparison

If we recall *Figure 6*, few parameters were estimated for the diabetes dataset with their correspondent value. Some of these parameters are actually a score of the model. Their values can be used for performing model selection based on comparison to models that fit the same data. In this thesis, the parameters that will be used for comparison will be:

- R squared
- AIC
- BIC
- P values

5.2.1 Akaike Information Criterion

When model's quality is assessed only by referring to fit, usually the model with more explanatory variables will always be chosen. However, this is not necessarily good. When a model uses too many parameters, overfitting [5] may occur. AIC raises to solve the problem of overfitting. It is an statistical measure developed by Akaike (1974) that asses model quality by calculating the AIC factor. It takes into account the log-likelihood estimation and the number of variables to be consider for regression. This factor can be expressed as:

$$AIC = 2k - 2ln(\hat{L}), \tag{27}$$

being $\hat{L}$ the value of the maximum of the likelihood function and k the number of parameters in the model.

The Akaike Information criterion is the derivation of Mallow's C_P if the residuals are normally distributed. This score is an estimate of the fit of the model (as it is derived from the log-likelihood), but taking into account the number of variables included. It is a relative measure so it is useful only when models are compared. The

[5]Overfitting is a phenomenon in which a model fits very close the existing data, see *Section 5.1*

model with the lowest AIC value is considered to be better. Therefore this measure is only useful for situations in which different models need to be compared.

However, AIC raises a problem when the number of observations is low, because then the correction of $2k$ is not enough for penalizing large models, this was shown in the paper (Hurvich and Tsai, 1989). Therefore a correction was developed by (Cavanaugh, 1997) for situations with small number of samples. The formula for the corrected version of the Akaike Information criterion is:

$$AIC_c = AIC + \frac{2k^2 + 2k}{n - k - 1}. \tag{28}$$

This formula must be used for when the sample is small. However when the sample tends to be big the *extra penalty* tends to 0, turning AIC_c into AIC. The following graph shows the comparison between both measures when applied to the diabetes dataset with few samples:

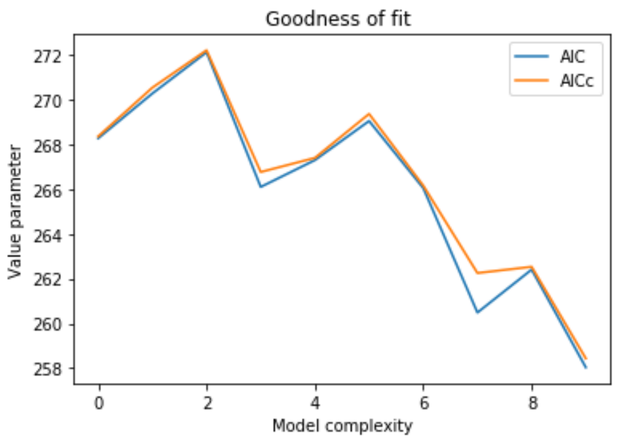


Figure 11: Comparison between AIC and AIC_c.

As we can see in the figure the penalization for larger models is more present in the corrected Akaike factor. This should be taken into account when testing data sets with small number of samples. This issue will be addressed in *Section 7*.

5.2.2 Bayesian Information Criterion

BIC also addresses the problem of overfitting by penalizing the log-likelihood estimate for models with higher complexity. The difference between BIC and AIC is that BIC takes into account the number of samples used in the model, even when the number of samples is big or small. the equation for calculating the BIC can be written as:

$$BIC = ln(n)k - 2ln(\hat{L}), \tag{29}$$

being $\hat{L}$ the value of the maximum of the likelihood function, n the number of samples and k the number of parameters in the model. As we can see in the equation AIC is similar to BIC. The only difference is that AIC uses a penalty of $2k$ while BIC uses $ln(n)k$. This can be shown in the following graph:

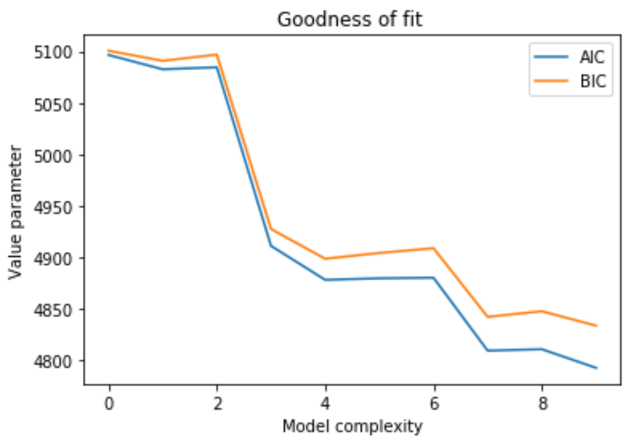


Figure 12: Comparison between AIC and BIC.

The main reason shown by (Burnham and Anderson, 2004) for using BIC instead of AIC is when the *true model* is in the set of candidate models. When this is the case BIC has a higher probability to choose the true model rather than AIC. This should be also taken into account when choosing the criteria for model selection.

5.2.3 R-squared

R-squared or the coefficient of determination can be intuitively defined as how good the variance of the model explains the variance of the data. It is also usually a measure of the goodness of fit of the model, representing how well does the model fit the data.

Linear Regression

$$R^2 = 1 - \frac{VAR_{model}}{VAR_{data}} \tag{30}$$

Its value is between 0 and 1, however, in occasions in which a very bad fitting model is chosen, this coefficient can even be out from this interval. R-squared increases every time a variable is included in the model, this is logical since the more explanatory variables in the model the best the data is fitted. However, this is is not necessarily good, since sometimes variables that are included in the model, even if they increase the simple R-square, they are rather not significant enough for the model.

Logistic Regression

In logistic regression, R-squared (called McFadden's pseudo R2) is defined as:

$$R^2 = 1 - \left(\frac{L(\hat{\beta})}{L(0)}\right)^{2/n}, \tag{31}$$

where $L(0)$ is the log-likelihood of the null model, $L(\hat{\beta})$ is the log-likelihood of the optimized model and n the number of samples. This definition of R-squared makes sense for the boundries of R^2. When the estimated model is as bad as the null model, R-squared is 0 and when the log-likelihood of the estimated model is 0 (this is the *true model*), R-squared is 1.

Corrected R^2 for Logistic and Linear Regression

As it was shown at the beginning of this section including too many variables can over fit a model. Overfitting can cause the predictive power of the model decrease.

To fix this problem extensions of R-squared are considered for model comparison with different number of explanatory variables, there is a way to adjust R-squared. This is the so called R-squared adjusted. It is defined as:

$$R^2_{adj} = 1 - (1 - R^2)\frac{n-1}{n-p-1} \tag{32}$$

In which n is the number of samples, R is the regular R-squared and p is the total number of explanatory variables. In the following graph the comparison between the non-adjusted and the adjusted can be seen:

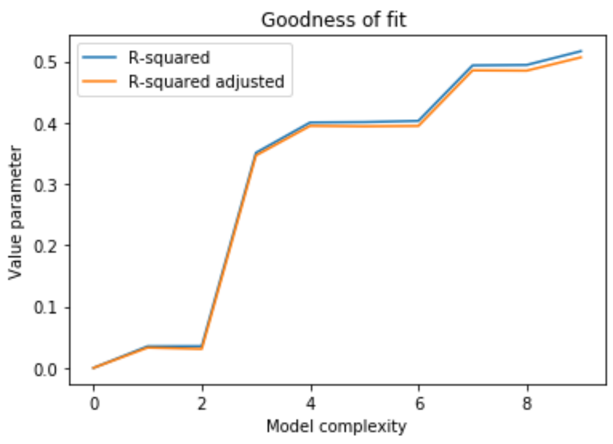


Figure 13: Comparison between R-squared and R-squared adjusted.

In the previous graph it can be seen how the adjusted R-squared decreases slightly. This adjusted R^2 can perform model selection by comparing two models that fit the same data but with different model complexity.

5.2.4 Stepwise Regression

Stepwise Regression is a model selection algorithm that fits the model given a certain selection criteria. In this thesis we will mainly focus on the p-value criteria that will be explained in this section.

Background

Stepwise Regression was developed by Efroymson (1960). It is a combination of Forward and Backward Regression. These algorithms work as follows:

- Forward Regression: the initial model contains no variables. Then all variables are considered for inclusion and the one that has the lowest p value is included in the model only if this p-value is lower than a given threshold *p-enter*. Afterwards, all the resting variables are considered again and the one that in combination with the first variable has a the lower p-value (and also lower than the threshold) is included in the model. The algorithm goes on and on till no variable makes the model significantly better.
- Backward Regression: the initial model contains all variables. Then variables are sequentially eliminated if their p-value is higher than a given threshold *p-remove*. When a variable is removed, the model is fitted again and all variables checked. The algorithm stops when no variable in the model has a higher p-value than *p-remove*.

Algorithm

Stepwise regression is a model selection method based on comparison. Each step of the algorithm considers each variable for inclusion to the model (forward) and all variables in the model for exclusion (backward). The addition of variables to the model is executed when a variable has pvalue lower than the p value threshold, to be specified by the user (by default 0.05). The exclusion of variables from the model is executed when a variable has a pvalue higher than the p value threshold, to be also specified by the user (by default 0.10).

Stepwise Regression is a combination of two other selection methods. Forward selection and backward selection. The forward selection algorithm includes variables in the model which in combination with the existing model makes the fit better and have an individual p value lower than a given threshold. This way the algorithm will start with no variables, and incorporate one by one, variables that make the model better, till no variable significantly improves the model. On the other hand, the backward selection starts with all variables in the model and eliminates one by one variables that are not significant for the model, given a threshold.

Stepwise regression is a combination of this two model selection methods. It sequentially performs forward and backward till no change significantly improves the model. Long story short, *forward* considers variables for inclusion and *backward* considers the selected variables from forward for exclusion.

5.3 Model Selection by optimization

The previous model selection algorithms where based on comparison of models fitted by maximizing the Log-likelihood (minimizing cost function), and once the parameters and the maximum Log-likelihood are obtained the scores were calculated. These scores were use then to determine which model is the best (in case of AIC, BIC and R^2_{adj}) or also to serve as an input for a more complex model selection algorithm (such as the p-values with stepwise regression).

However, some model selection methods perform variable selection from the moment the parameters are estimated. This is that they minimize a different *'cost function'* in order to perform model selection. As shown in the literature review (*Section 2*) there are several methods for model selection by optimization. Nevertheless, in this section we will only focus in *least absolute shrinkage and selection operator* (Lasso) and Elastic Net due to the reasons explained in *Section 2*.

5.3.1 Lasso

We can recall from *Section 4.2* that finding the minimum of

$$argmin_{\beta_0,\beta_1,\ldots,\beta_p}\left(\sum_{i=1}^{m}\left(h_\beta(x^{(i)})-y^{(i)}\right)^2\right) \tag{33}$$

leads to obtaining the coefficients for the parameters in linear regression that best fit the data. However, as it was said at the beginning of this section finding the best fit is not a good idea if the purpose of our model is prediction.

The Lasso, performs shrinkage and variable selection by solving the l_1-penalized regression problem (Tibshirani, 2011) on finding

$$argmin_{\beta_0,\beta_1,\ldots,\beta_p}\left(\sum_{i=1}^{m}\left(h_\beta(x^{(i)})-y^{(i)}\right)^2+\lambda\sum_{j=1}^{p}|\beta_j|\right) \tag{34}$$

which is basically minimizing the cost function plus an additional term subject to a constraint $\sum|\beta_j| \leq s$ where s determines the amount of regularization. This minimization leads to a shrinkage of the parameters that can end in variable selection (parameters can shrink to 0) depending on the value of the l_1-penalty, λ. The following graph shows how this penalty shrinks the parameters and performs variable selection when estimating a model for the diabetes dataset, that contains 10 variables:

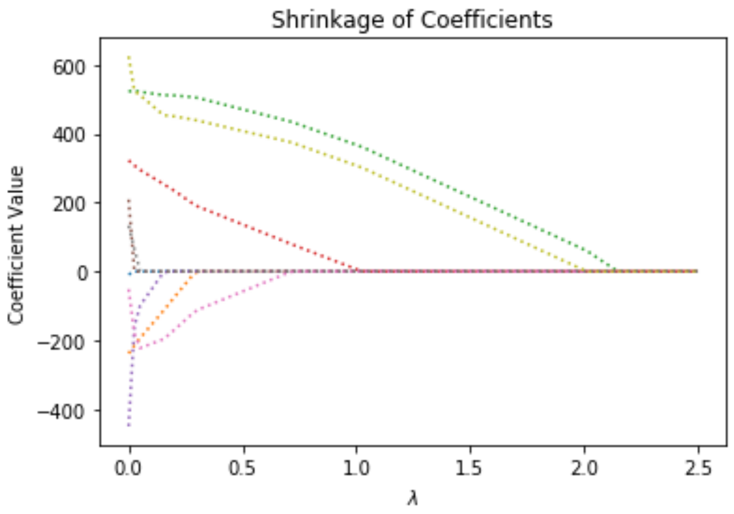


Figure 14: Performing variable selection for the diabetes dataset using Lasso

In the previous figure it can be seen how for higher values of λ model complexity becomes smaller. The only value necessary to adjust for implementing Lasso is λ. Traditionally, this penalization parameter must be chosen manually by the user, but this be done carefully. As it was shown in *Figure 14* the shrinkage of the parameters is not the same for all, some of them shrink faster and some slower, leading to different results (errors, variance...) in the final model. Later on, in *Section 7* we will propose an automatic method to chose the value of the penalization.

5.3.2 Elastic Net

Elastic Net is a generalization of *least absolute shrinkage and selection operator* (Lasso) that combines an l_1 penalty with a l_2 penalty (Ridge Regression). It was a method proposed by Zou and Hastie (2005) to solve three limitations that Lasso had (Zou and Hastie, 2005):

- $p \geq n$: when the number of variables is higher than the number of observations

Lasso selects at most n variables variables

- **Correlation**: if there is a group of variables highly correlated, Lasso tends to select one variable from the group and disregard the rest
- **Case** $n \geq p$. In cases where the number of observations is higher than the number of variables (which is usually the case) and there are high correlations among the variables, Ridge Regression has a better predictive power.

Elastic Net performs as well variable selection and parameter shrinkage by performing the minimization of the function:

$$argmin_{\beta_0,\beta_1,\dots,\beta_p}\left(\sum_{i=1}^{m}\left(h_\beta(x^{(i)})-y^{(i)}\right)^2+\lambda\alpha\sum_{j=1}^{p}|\beta_j|+\lambda(1-\alpha)\sum_{j=1}^{p}|\beta_j|^2\right) \tag{35}$$

which is basically minimizing the cost function plus two additional terms (l_1-penalty (Lasso) and l_2-penalty (Ridge)). The value of α is to be chosen by the user and represents the proportion from each penalty to be used in the optimization. If α is chosen to be 1 then the implementation would be Lasso and if α would be 0 then Ridge. Using for example $\alpha = 0.9$ the shrinkage process depending on the penalization will be:

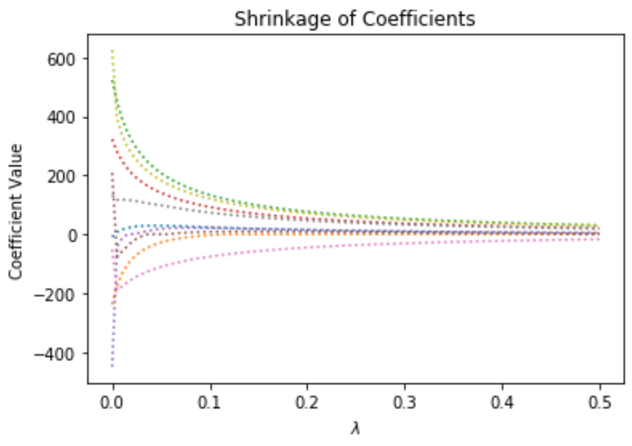


Figure 15: Performing variable selection for the diabetes dataset using elastic net with $\alpha = 0.9$.

5.4 Conclusion

We presented a model selection framework that divides model selection algorithms into two types: model selection by comparison (*Section 5.2*) and model selection by optimization (*Section 5.3*). Afterwards, several model selection algorithms have been explained with examples in order to see how do the different procedures perform variable selection.

The following table summarizes the model selection framework presented in this section:

Model Selection Criteria		Comments
Comparison	BIC	Select model with highest BIC
	AIC	Select model with lowest AIC
	R^2_{adj}	Select model with highest R^2_{adj}
	Stepwise	Selects model that does not significantly improve the fit
Optimization	Lasso	$min\left(\sum_{i=1}^{m}\left(h_\beta(x^{(i)})-y^{(i)}\right)^2+\lambda\sum_{j=1}^{p}\|\beta_j\|\right)$ subject to $\lambda\sum_{j=1}^{p}\|\beta_j\|\leq s$
	Elastic Net	$min\left(\sum_{i=1}^{m}\left(h_\beta(x^{(i)})-y^{(i)}\right)^2+\lambda(1-\alpha)\sum_{j=1}^{p}\|\beta_j\|+\lambda\alpha\sum_{j=1}^{p}\|\beta_j\|^2\right)$ subject to $\lambda(1-\alpha)\sum_{j=1}^{p}\|\beta_j\|+\lambda\alpha\sum_{j=1}^{p}\|\beta_j\|^2\leq s$

Table 3: Model Selection framework.

These procedures are going to be used as a point of departure in *Section 7* to build model selection tools with a wide range of features and possibilities. However, it is essential to deeply understand the previous section in order to extract conclusions from the simulations.

6 Model Validation

6.1 Introduction

In previous sections we discussed model estimation (logistic and linear regression) and some model selection procedures (AIC,BIC, Stepwise, Lasso...). Model estimation methods are necessary to understand in order to be able to identify data type and to understand estimation procedure. Also and equally important is to choose the model that aims to describe how data are generated (model selection). When the goal of these models is prediction, it is desired that the model's accuracy with new data is high. However sometimes the data used for training the model does not significantly represent all sets of data (it can be biased). Therefore, models that have been trained with some data must be validated. Cross validation is a well-known common technique for model validation which basically consists of partitioning a set of data into a training set and a validation set. The training set is used to fit the data and estimating the value of the parameters and the validation set is used for calibration (or evaluation) of the model. If the performance of the model in the validation set is good enough, there is no need for model calibration. However, if there is some error in the validation set, the modeler has to calibrate the parameters of the trained model taking into account the results from the validation set. For example, the validation set can be used for calibrate the regularization coefficient λ or the α ratio for Elastic Net.

There are several methods for partitioning data for cross validation models. The conventional way is partitioning data in two sets: 70% training and 30% validation. However if the partitioning is done randomly it can occur that the training set differs a lot from the validating set. For example, if we want to model Probability of Default, it would be very inefficient if all non-defaulted loans are in the training set, and all defaulted loans are in the validation set.

In this case, our model will be trained for not predicting any default and our validating model will try to calibrate our model for predicting all default. Taking this issue into account is important in order to *split* your data in a smart way. In this section we will talk about three methods of cross validation that will be used for calibrating our model selection methods. These are Leave One Out, K-fold and Stratified K-fold.

6.2 Leave One Out

Leave One Out is a method of cross validation that partitions the data sample by sample. Suppose we have a dataset with a number of samples n. This method divides the data in $n-1$ training set and 1 as a validation set. This is done n times for each sample, so each sample will eventually become a validation set. In the following example we will train the diabetes dataset using Lasso Model Selection and we will calibrate the regularization coefficient α using Leave One Out Cross Validation. For each value of α, n models are trained and validated using the resting sample. The mean error from the validation of each sample is taken:

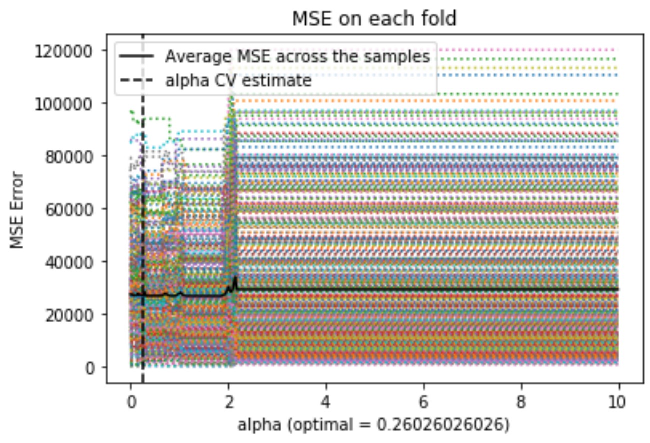


Figure 16: Leave One Out Cross Validation.

As it can be seen in this case we are calibrating alpha. The calibration is done in the following way:

- For each sample i, a model is trained excluding that sample i, and using $n-i$ as a training set.
- The error in the prediction of that sample i is computed for each value of α
- The average error across the samples is computed
- The chosen value of α is the one that gives the minimum average error

This method has a disadvantage when using very large datasets. This is because it is computationally very expensive to run n different models for datasets with very large number of samples. This issue will be addressed in *Section 9* when using Leave One Out Cross Validation to calibrate the value of α in Lasso Regression

6.3 K-Fold

K fold uses the same underlying idea of Leave One Out, but it does bigger partitions, and thus it becomes less computationally expensive. Naming k as the number of folds (data partitions), the samples are divided into k different folds. Then, the model is trained k times using $n/(k-1)$ samples as a training set and n/k samples as a validation set. Then the procedure for calibrating the model is the same as in Leave One Out but instead for each sample, for each fold.

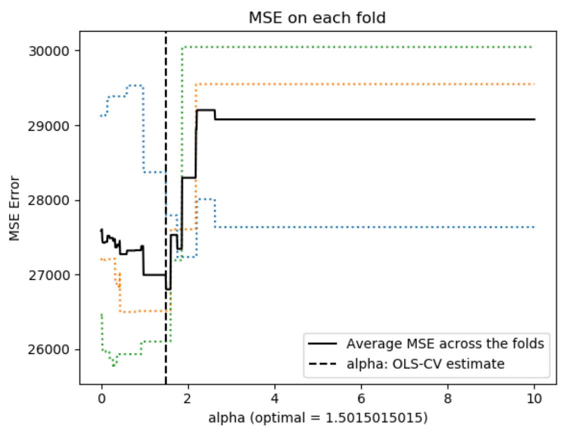


Figure 17: K Fold Cross Validation using 3 folds.

The number of folds is an important consideration to take into account. Notice that when $k \Rightarrow n$ K fold turns into Leave One Out. If we increase the number of folds the α selected will asymptotically converge to the α calculated in Leave One Out:

It can be concluded then that Leave One Out is a prudent way of performing cross validation, but due to its computational costs, K-Fold results less time consuming and thus is frequently more used when dealing with larger datasets. Even if K fold is computationally more efficient it does not address the issue presented in the

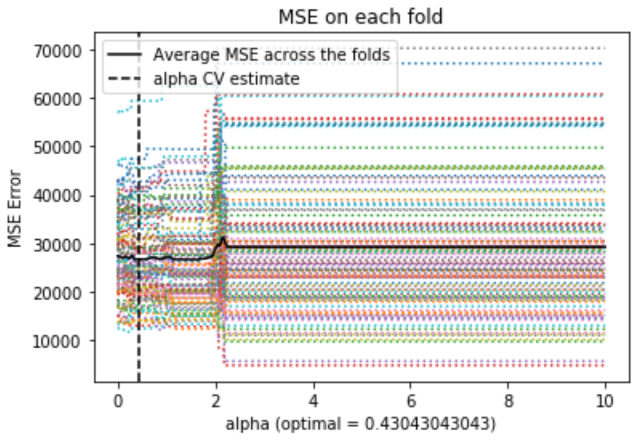


Figure 18: K Fold Cross Validation using 100 folds.

introduction of these section about the percentage of output type in the folds. The solution to this problem is presented in the next section.

6.4 Stratified K-Fold

Stratified K-fold is a variation of K-fold that solves the problem of percentage of output type in each fold. For example, in the case of **binary logistic regression** this cross validation method ensures that in each fold the percentage of 1s and 0s in each fold (mean response value) is approximately equal. This makes each fold a better representative of the whole dataset which leads to a better estimation of α.

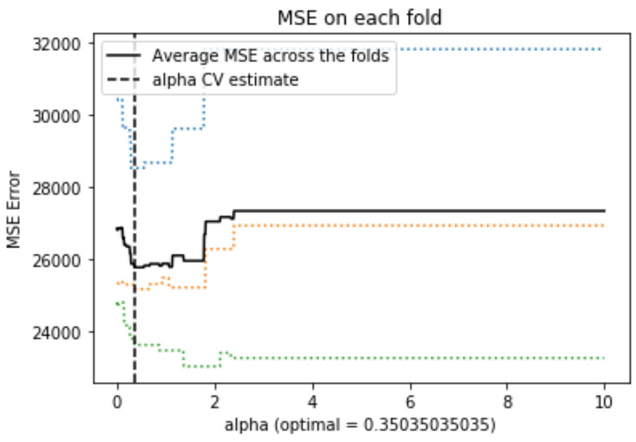


Figure 19: Stratified K Fold Cross Validation using 3 folds.

6.5 Conclusion

In this section we have presented three different methods to calibrate model parameters. These methods will be used in the next sections for algorithms that have penalization terms to be specified. However, cross validation methods can make this penalization selection automatic, calibrating the value of the parameter α. Therefore, we will combine model selection methods with cross validation.

7 Specifications for chosen model selection algorithms

7.1 Introduction

This section will present the algorithms that haven been programmed in order to test the model selection algorithms that want to be compared. If we recall these algorithms were Stepwise, Lasso, Elastic Net, BIC, AIC and R_{adj}.

The first algorithm will be built to test Stepwise Regression. It combines Forward selection and Backward elimination with different features to be adjusted by the user. The second algorithm called Bruteforce will test all the possible combinations of variables and will select the one that scores the best AIC, BIC, R_{adj} or Log-Likelihood (criterion must be specified by the user). The third algorithm combines Lasso shrinkage with automatic calibration of the penalization parameter via cross validation (CV method to be chosen by the user). The last algorithm performs Elastic Net with automatic calibration of the penalization and α.

7.2 Stepwise Regression with stopping criteria

Stepwise regression is a combination of this two model selection methods. It sequentially performs forward and backward till no change significantly improves the model. The algorithm *forward* considers variables for inclusion and *backward* considers the selected variables from forward for exclusion. This function has been programmed in Python and also it has been tailored for Rabobank purposes, and thus, includes special features. These are:

- White List: List of variables that are forced to be included in the model.
- Sign: Variables in the final model will only have a concrete sign.
- In model: List of variables to be initialized, but not necessarily end up in the final model.
- P-enter: Maximum p-value that specifies the max o p-value for a predictor to enter regression.

- P-remove: Minimum p-value that specifies the min o p-value for a predictor to be recommended for removal.
- Stopping criteria: Stopping criteria based on optimum BIC or AIC.
- Maximum number of variables: Maximum number of variables allowed to be in the model.

The algorithm can be represented as:

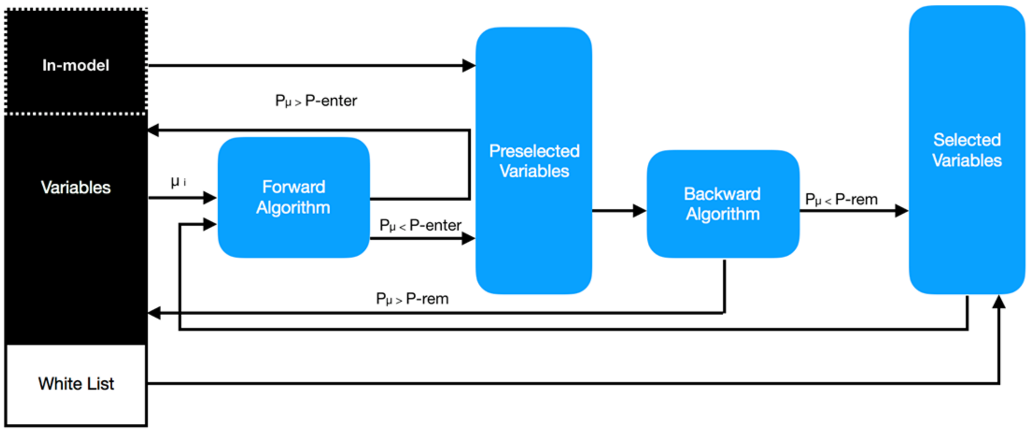


Figure 20: Stepwise algorithm.

7.3 Bruteforce

Bruteforce is an algorithm that given a dataset finds the best combination of features following a specific criterion. This criterion must be selected by the user, there are four options:

- AIC: Akaike information criterion (chooses the model with the lowest AIC score).
- BIC: Bayesian information criterion (chooses the model with the highest BIC score).
- R^2_{adj}: Coefficient of determination chooses the model with the highest R^2_{adj} score.

- *Log − Likelihood* (Chooses the model with the highest Log-Likelihood)

Given a set of data, Bruteforce rotates all possible combinations of attributes using the variables given without exceptions.

This function can be computationally very expensive, thus, is recommended not to apply Bruteforce to very large datasets. Instead, other selection methods can be used, such as Lasso Elastic Net or stepwise.

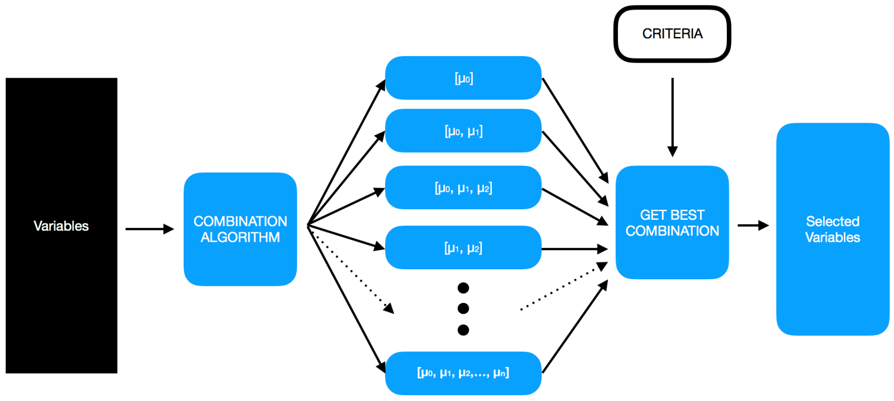


Figure 21: Bruteforce algorithm.

Bruteforce can be considered as a rather simple algorithm. The package *itertools* is specially imported for the purpose of creating all possible combinations of variables. (Note combinations, not permutations). After creating all possible combinations, each of them is used for building a model and fitting the data. Afterward, the criterion value is extracted from the model and compared to the ones from the other combinations. The best combination then is chosen.

7.4 Post Lasso CV

Post Lasso CV performs Lasso model selection for a given set of data for different values of the regularization parameter (alpha). The dataset is cross validated to calibrate the value of the regularization parameter in order to get the value of α that gives the lowest *error* for the resulting model.

7.4.1 Background

The error in regression methods is usually a decisive factor to choose for one method or another. Typically, the mean square error is an indicator of the model quality. If we recall *Equation 26* that explained the demonstration of (Sammut and Webb, 2017):

$$E[(y - f')] = \sigma^2 + var[f'] + bias[f']^2 \tag{36}$$

The variance in itself measures the prediction power of our model. In other words, how accurate is the prediction. On the other hand, *bias* measures how far is our data from reality. A straight forward reasoning would be concluding that the best model is the one with low variance and low bias. But this is not as easy as it seems. Models that are estimated maximizing the Log-likelihood function, usually fit good the data, achieving a low bias in the model. However, the price of model's fitted in this way is a poor predictive power. Lasso, Ridge and Elastic Net partially fix this problem by introducing a regularization parameter that reduces overfitting, by penalizing variables that are not very significant for the model. Lasso and Elastic Net also perform variable selection by penalizing some variables to 0 (which Ridge does not), reducing model's complexity. However, the price to pay for this if the regularization parameter is to high is a high bias in the model.

7.4.2 Purpose

The purpose of Post Lasso CV is to find the optimum regularization parameter such as the total error associated with the model is the lowest possible. If we recall *Figure 10* model's error has a concave shape. Post Lasso CV aims to find automatically the minimum of this function by computing the average error across the cross validation folds for different values of the regularization parameter.

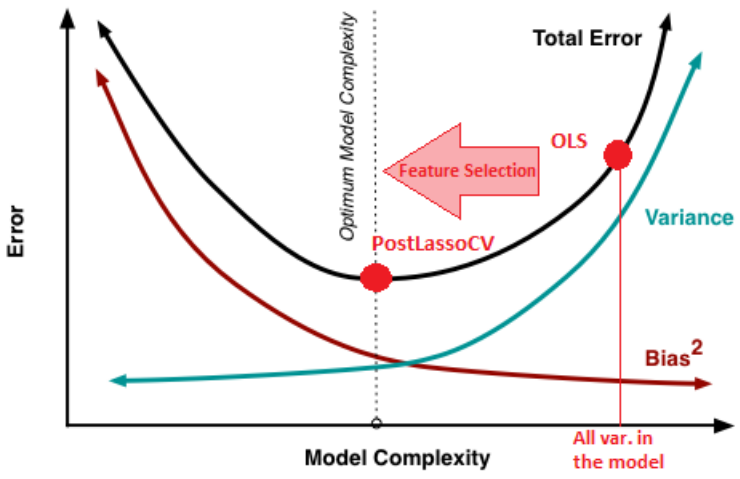


Figure 22: Variance - Bias Trade off.

As it can be seen we will start with a Lasso model with $\lambda = 0$ (which is the same as saying OLS for linear regression) and increase alpha sequentially till we have the whole picture, for finally select the optimum model.

7.4.3 Algorithm

As explained before, the function iterates over a range of values of the regularization parameter and outputs the selected variables for each λ. For each value of the regularization parameter ($l1$ penalization) the parameters selected by Lasso are re-fitted maximizing the Log-likelihood function without the regularization parameter. The final model is the one that has the least mean error across the folds for a given alpha. The user is able to choose the cross-validation method.

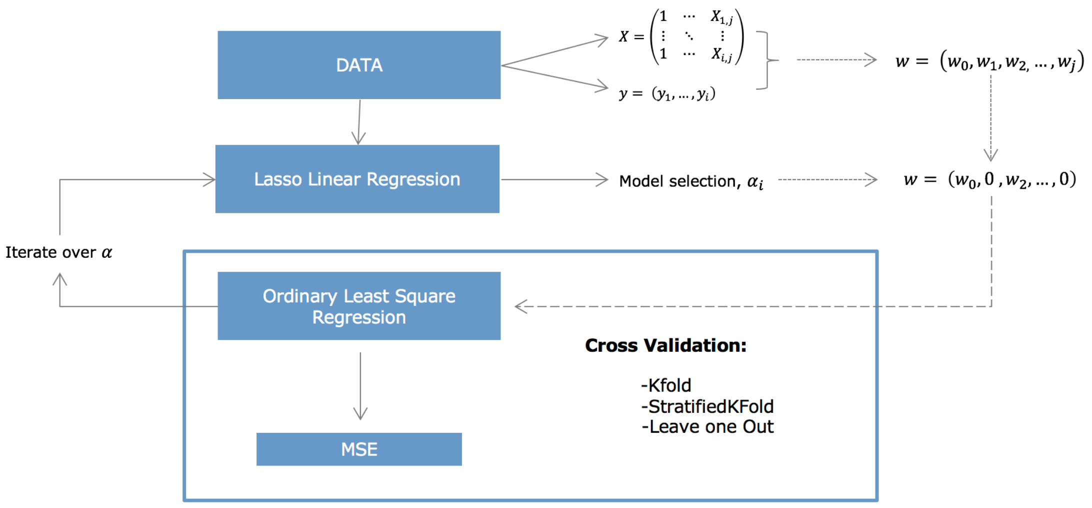

DATA
$X = \begin{pmatrix} 1 & \cdots & X_{1,j} \\ \vdots & \ddots & \vdots \\ 1 & \cdots & X_{i,j} \end{pmatrix}$
$y = (y_1, \ldots, y_i)$
$w = (w_0, w_1, w_2, \ldots, w_j)$
Lasso Linear Regression
Model selection, α_i
$w = (w_0, 0, w_2, \ldots, 0)$
Iterate over α
Ordinary Least Square Regression
MSE
Cross Validation:
-Kfold
-StratifiedKFold
-Leave one Out

Figure 23: Variance - Bias Trade-off.

7.4.4 Example

Using the diabetes dataset we perform PostLassoCV using Stratified K-fold cross validation method with 5 folds. The result from the function is:

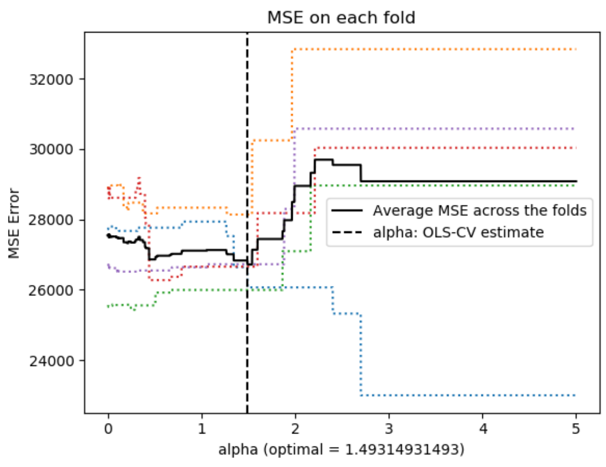


Figure 24: Final PostLasso.

If we look carefully at the graph we can even identify the variance bias trade-off in the average mean squared error across the folds:

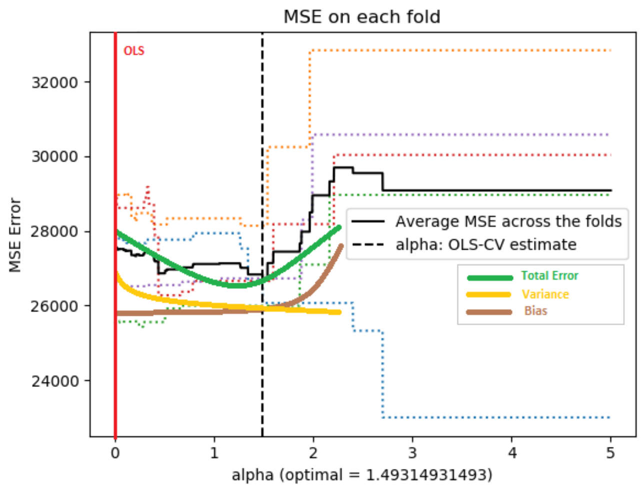


Figure 25: Final Post Lasso CV.

As we can see Post Lasso CV will select the value of α that gives the lowest Total Error leading to a higher prediction power of the model. As explained previously in last section this graph shows the error across the folds (in this case 5 folds), and the average of these errors (the thick black line). This cross validation method aims to simulate the variance-bias trade-off to be able to choose the lower error point. If alpha is set to 0, we will have a traditional model with all variables, if we increase the value of alpha we will be shrinking variables to 0 (*See Figure 14*), and thus, progressively as alpha becomes very big, there will be no model (or constant 0 model), which leads to high error.

7.5 Elastic Net CV

Elastic Net CV performs Elastic Net model selection for a given set of data for different values of the regularization parameter (alpha), and the penalization ratio. The dataset is cross validated aiming to calibrate the value of the regularization parameter in order to get the value of the regularization parameter in combination with the penalization ratio that gives the lower *Mean Error* across the folds.

This function is a generalization of the previous Post Lasso CV that adds another dimension to the optimization of the model selection algorithm. It will rotate the penalization parameter as well as the regularization and select the situation in which the error is lowest for the model.

7.6 Conclusion

In this section we have presented four algorithms that perform model selection (& calibration in Post Lasso CV and Elastic Net Cross Validation case). These algorithms differ slightly from the conventional ones. In the case of Stepwise Regression, the algorithm includes more input features that can be manipulated to obtain a better result. Bruteforce gives the chance to vary the selection criteria in order to evaluate the model selection.

Post Lasso CV and Elastic Net Cross Validation perform automatic calibration for conventional Lasso and Elastic Net procedures and output the final calibrated model and the regularization (and penalization for EN) parameter.

Section 8 will present data to be used for comparing the efficiency of these algorithms and the testing procedure to follow. The sections after this will discuss the results and present the comparison performance of these four model selection methods.

8 Testing

8.1 Introduction

In this section we are going to test data from Rabobank. Because of confidential reasons neither the **nature of this data** nor the **name of variables** will be mentioned. Instead, all variables will be named as $(x_0, x_1, ..., x_p)$ being p the variable index.

Firstly, the type of data is going to be slightly introduced, however not in much depth as it was mentioned before. Then the testing method framework is going to be described for the reader to understand the methodology followed for comparing each selection procedure. The chapter will end with a conclusion that ties up the testing procedure and its reasoning. Next section will present the results of this testing.

8.2 Data

The data that are going to be use for testing have the format of a matrix of n x p:

$$\begin{pmatrix} x_{0,0} & x_{0,1} & \cdots & x_{0,p} \\ x_{1,0} & x_{1,1} & \cdots & x_{1,p} \\ \vdots & \vdots & \ddots & \vdots \\ x_{n,0} & x_{n,1} & \cdots & x_{n,p} \end{pmatrix}$$

where in our case $n(= 43381)$ is the total number of samples and $p(= 26)$ is the number of variables to be considered. Also, we will have the column of outputs (dependent variable) y:

$$\begin{pmatrix} y_0 \\ y_1 \\ \vdots \\ y_n \end{pmatrix}$$

that denotes the default status (0 or 1).

Therefore we can conclude that the type of regression to be performed with this data is logistic regression (*see Section 4.2.*)

8.3 Testing procedure

The testing criteria have been developed taking into account the model selection methods to be considered. These are the ones explained in the previous section: *Post Lasso CV*, *Stepwise Regression* and *Elastic Net CV*.

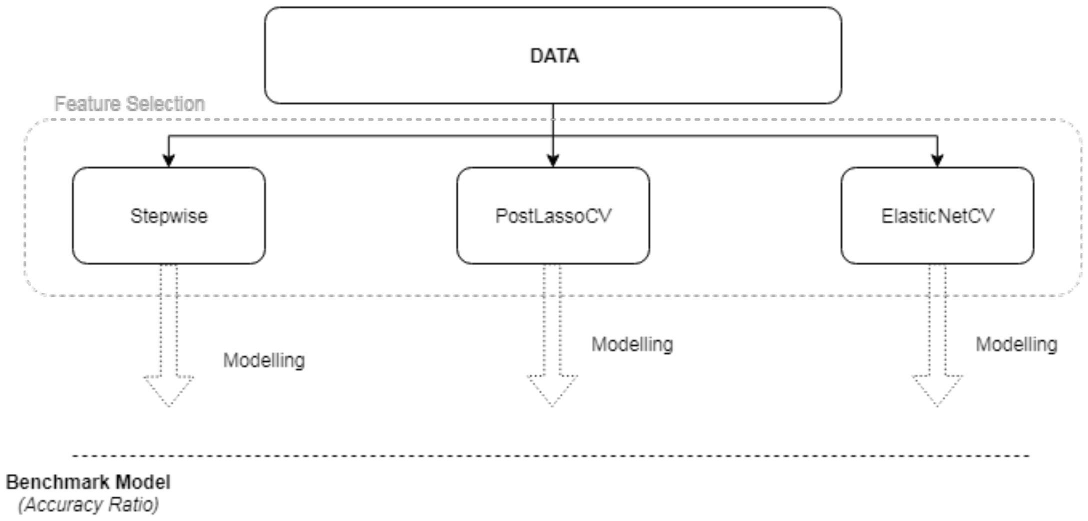


Figure 26: Testing procedure.

With each of this procedures a final model will be obtained. The performance of the resulting models are going to be compared on a training and also on a validation (testing) set, this will be mainly used for measuring the Accuracy Ratio on a set of data that has not been used for building the model. Therefore the splitting of the data will be:

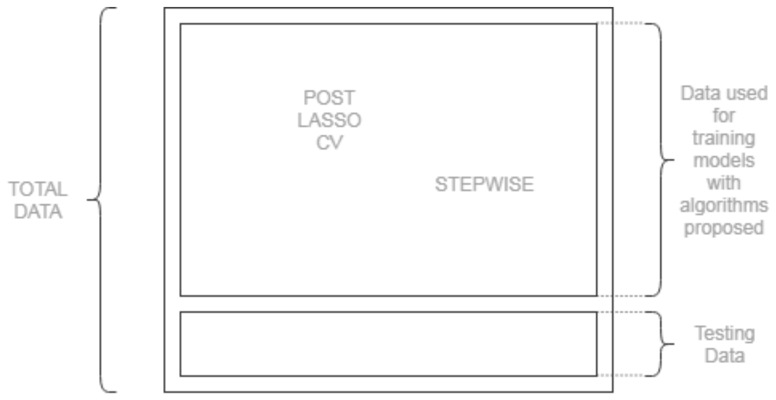


Figure 27: Data Usage.

Additionally, these functions have also some optional input parameters that may affect the implementation of the algorithm and thus, the outcome model. Therefore these input parameters will be also be evaluated when executing the model selection functions.

8.4 Conclusion

In this section the testing procedure and the data used have been presented. We establish a way to test different model selection algorithms that have been previously constructed in order to be able to compare their performance. The resulting models for each algorithm are going to be presented and compared in *Section 9.* After this we will finalize the thesis extracting some conclusions and recommendations.

9 Results

9.1 Introduction

In the following section the testing results are going to be presented. For each model selection procedure there are different input parameters that can be adjusted. Naturally the outcome depends on these parameters, and thus this will be tested as well.

For stepwise regression the p-value thresholds will be modified in order to find the combination of *p-enter* and *p-remove* that gives the best model. Also for Post Lasso CV the coefficient threshold for variable selection will be as well modified. The performance of both algorithms will be compared also in a test set as explained in the last section.

Finally we are going to make a fairer comparison between both using Stepwise and PLCV as model selector algorithms and calculating the accuracy ratio predicting in different folds and also the selection progress. This will give a robust ratification of the comparison.

9.2 Stepwise regression model

The results obtained for the stepwise regression algorithm are going to be shown. The inputs are varied in order to study the best possible combination (if any) from the p threshold values for inclusion and exclusion from the model. Usually the stepwise has by default a 5% threshold for entering the model and a 10% for exiting the model. However, in this thesis we will extend the number of combinations to find out if other threshold combination leads to better results than the one established from default. In the following tables we have measured different parameters: Akaike Information Criterion, Accuracy Ratio in Train Set, Accuracy Ratio in Test Set, Log-Likelihood and Degrees of Freedom. The resulting models are:

	p enter								
p remove		0.05	**0.1**	**0.15**	**0.20**	**0.25**	**0.3**	**0.35**	**0.45**
	0.010	49130.5	49130.5	49130.5	49130.5	49130.5	49130.5	49130.5	49130.5
	0.025	49130.5	49130.5	49130.5	49130.5	49130.5	49130.5	49130.5	49130.5
	0.050		49130.5	49130.5	49130.5	49130.5	49130.5	49130.5	49130.5
	0.075		49129	49129	49129	49129	49129	49129	49129
	0.100			49129	49129	49129	49129	49129	49129
	0.125			49129	49129	49129	49129	49129	49129
	0.150				49128.9	49128.9	49128.9	49128.9	49128.9
	0.175				49128.9	49128.9	49128.9	49128.9	49128.9
	0.200					49128.9	49128.9	49128.9	49128.9
	0.225					49128.9	49128.9	49128.9	49128.9
	0.250						49128.9	49128.9	49128.9
	0.300							49128.9	49128.9
	0.350								49128.9
	0.450								

Table 4: Akaike Information Criterion.

	p enter								
p remove		0.05	**0.1**	**0.15**	**0.20**	**0.25**	**0.3**	**0.35**	**0.45**
	0.010	16	16	16	16	16	16	16	16
	0.025	16	16	16	16	16	16	16	16
	0.050		16	16	16	16	16	16	16
	0.075		17	17	17	17	17	17	17
	0.100			17	17	17	17	17	17
	0.125			17	17	17	17	17	17
	0.150				18	18	18	18	18
	0.175				18	18	18	18	18
	0.200					18	18	18	18
	0.225					18	18	18	18
	0.250						18	18	18
	0.300							18	18
	0.350								18
	0.450								

Table 5: Degrees of Freedom.

	p enter								
p remove		**0.05**	**0.1**	**0.15**	**0.20**	**0.25**	**0.3**	**0.35**	**0.45**
	0.010	-24548.2	-24548.2	-24548.2	-24548.2	-24548.2	-24548.2	-24548.2	-24548.2
	0.025	-24548.2	-24548.2	-24548.2	-24548.2	-24548.2	-24548.2	-24548.2	-24548.2
	0.050		-24548.2	-24548.2	-24548.2	-24548.2	-24548.2	-24548.2	-24548.2
	0.075		-24546.5	-24546.5	-24546.5	-24546.5	-24546.5	-24546.5	-24546.5
	0.100			-24546.5	-24546.5	-24546.5	-24546.5	-24546.5	-24546.5
	0.125			-24546.5	-24546.5	-24546.5	-24546.5	-24546.5	-24546.5
	0.150				-24545.4	-24545.4	-24545.4	-24545.4	-24545.4
	0.175				-24545.4	-24545.4	-24545.4	-24545.4	-24545.4
	0.200					-24545.4	-24545.4	-24545.4	-24545.4
	0.225					-24545.4	-24545.4	-24545.4	-24545.4
	0.250						-24545.4	-24545.4	-24545.4
	0.300							-24545.4	-24545.4
	0.350								-24545.4
	0.450								

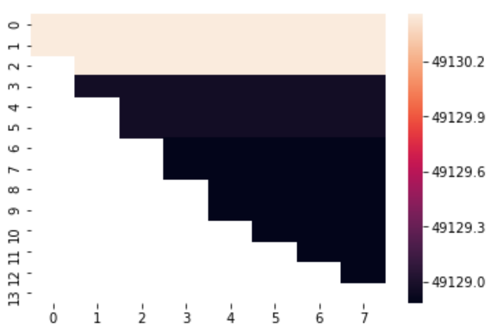
Table 6: Log-Likelihood.

	p enter								
p remove		0	**0.05**	**0.1**	**0.15**	**0.20**	**0.25**	**0.3**	**0.35**
	0.010	0.494885	0.494885	0.494885	0.494885	0.494885	0.494885	0.494885	0.494885
	0.025	0.494885	0.494885	0.494885	0.494885	0.494885	0.494885	0.494885	0.494885
	0.050		0.494885	0.494885	0.494885	0.494885	0.494885	0.494885	0.494885
	0.075		0.495013	0.495013	0.495013	0.495013	0.495013	0.495013	0.495013
	0.100			0.495013	0.495013	0.495013	0.495013	0.495013	0.495013
	0.125			0.495013	0.495013	0.495013	0.495013	0.495013	0.495013
	0.150				0.49514	0.49514	0.49514	0.49514	0.49514
	0.175				0.49514	0.49514	0.49514	0.49514	0.49514
	0.200					0.49514	0.49514	0.49514	0.49514
	0.225					0.49514	0.49514	0.49514	0.49514
	0.250						0.49514	0.49514	0.49514
	0.300							0.49514	0.49514
	0.350								0.49514
	0.450								

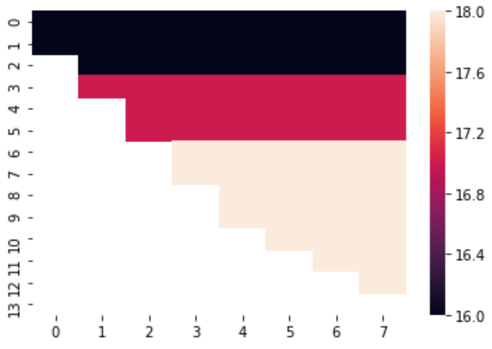
Table 7: Accuracy Ratio (*Training Set*).

	p enter								
p remove		0.05	**0.1**	**0.15**	**0.20**	**0.25**	**0.3**	**0.35**	**0.45**
	0.010	0.487272	0.487272	0.487272	0.487272	0.487272	0.487272	0.487272	0.487272
	0.025	0.487272	0.487272	0.487272	0.487272	0.487272	0.487272	0.487272	0.487272
	0.050		0.487272	0.487272	0.487272	0.487272	0.487272	0.487272	0.487272
	0.075		0.487499	0.487499	0.487499	0.487499	0.487499	0.487499	0.487499
	0.100			0.487499	0.487499	0.487499	0.487499	0.487499	0.487499
	0.125			0.487499	0.487499	0.487499	0.487499	0.487499	0.487499
	0.150				0.487343	0.487343	0.487343	0.487343	0.487343
	0.175				0.487343	0.487343	0.487343	0.487343	0.487343
	0.200					0.487343	0.487343	0.487343	0.487343
	0.225					0.487343	0.487343	0.487343	0.487343
	0.250						0.487343	0.487343	0.487343
	0.300							0.487343	0.487343
	0.350								0.487343
	0.450								

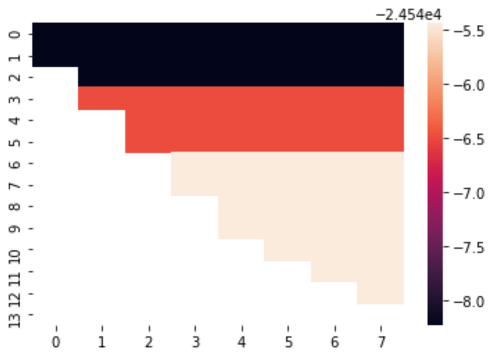
Table 8: Accuracy Ratio *(Test Set)*.



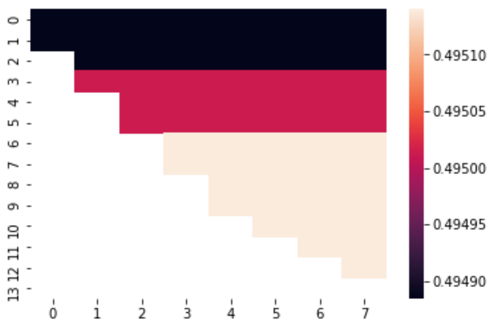
(a) AIC.



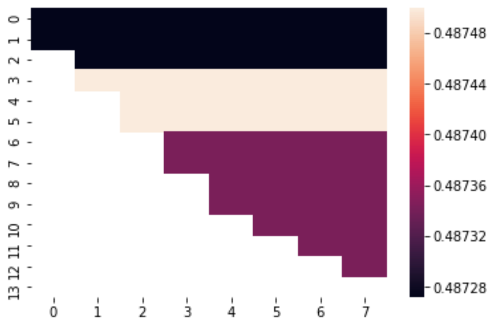
(b) Degrees of Freedom.



(c) Log-likelihood.



(d) Accuracy Ratio *Training Set.*



(e) Accuracy Ratio *Test Set.*

Figure 28: Results from stepwise.

9.2.1 Discussion

In the table shown before we have the results from the stepwise algorithm performing model selection with different algorithm input variables and the same dataset. We can observe a correlation between the number of variables chosen and the log-likelihood estimation. This can be intuitive if we consider the LL as a measure of fit, the more explanatory variables used for a model the better the fit will be. Also we can see that the models with higher AIC and are the models that have a stricter restriction for the p value to enter the model. If we recall the equation for AIC:

$$AIC = 2k - 2ln(\hat{L}), \tag{37}$$

we can explain the heat map from the Akaike Information Criterion based on the one of Log-Likelihood and Degrees of Freedom . If we would follow the criteria of selecting the model with better AIC we would have to select the model with the lowest AIC. Looking at the heat map this would mean that we would have to select the models that are constructed with more relaxed p-enter thresholds. This is the ones that have a P enter between (0.075 - 0.45) and p-remove (0.1 - 0.45).The same conclusion would be obtained by choosing the model based on the Accuracy Ratio Criterion. However, when we take a look at the accuracy ratio in the test set, we can observe that the accuracy ratio of the model that has a high AR in the training set does not have the highest AR in the test set. This demonstrates that the model with 18 variables is overfitting because it does not perform as well as the one with one variables less in the test set.

Another aspect to take into account is that when the p values of the most significant variables are calculated, we face that some of the variables are considered to have a p value of the order of 10^{-50}. In the previous tables we shown that we worked with p values that are rather intuitive to work with (0.0000001 to 0.1), and the results were models with about 15 degrees of freedom. However, if we would have wanted to reach a less complex model with 4 or 5 variables we would have had some problems specifying the p values threshold for the algorithm. We could say then that some of the p values are too low.

9.3 Post Lasso CV Regression Model

The results obtained for the Post Lasso CV regression algorithm are going to be shown. Post Lasso CV outputs the optimum penalized model. Because of the characteristics of the function we must pay attention to the range of α. In this section we are going to show what must be the analysis procedure to use this function in an efficient way. First of analyze the shrinkage of the parameters for different values of α:

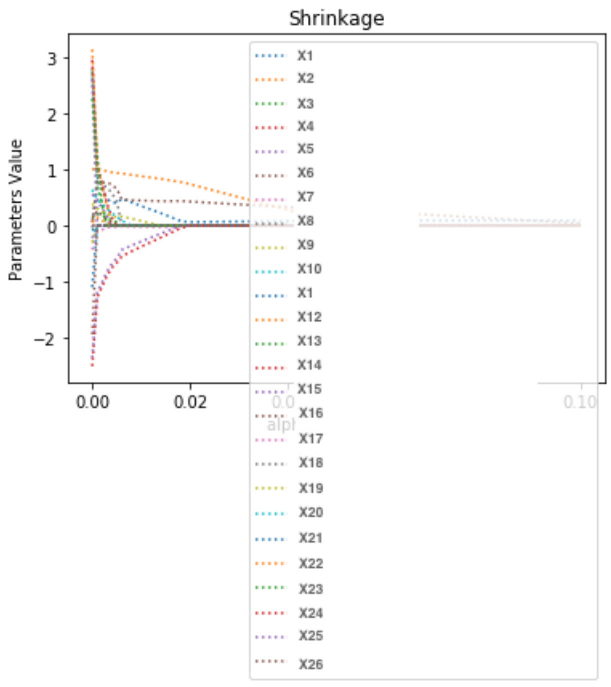


Figure 29: Shrinkage of parameters.

The following results show that for Lasso Shrinkage using a relatively high alpha (bigger than 0.02) there are less than 5 parameters with a non-zero value. This could result in an underfitting situation. Now we will evaluate Post Lasso between 0 and 0.02 to see where are we in the variance bias trade-off curve:

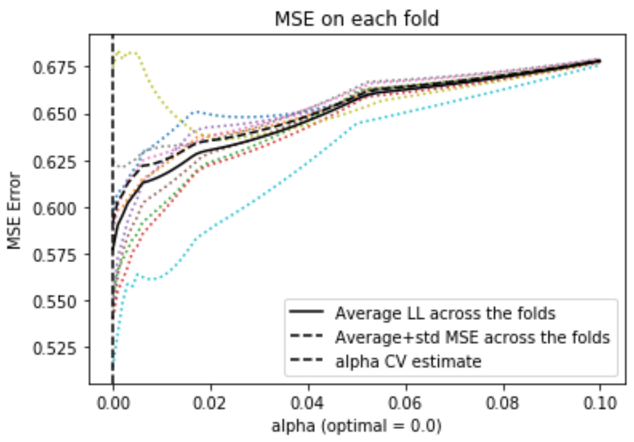


Figure 30: Post Lasso CV with a max alpha of 0.02.

We can see in *Figure 30* that PLCV computes the Log-Likelihood for each fold for different levels of the $l1$ penalization term. The α selected gives the minimum of the following function:

$$argmin\left(\frac{1}{k}\sum_{i=1}^{k}LL_i(\alpha)+\sqrt{\frac{1}{k}\sum_{I=1}^{k}\left(\frac{1}{k}\sum_{i=1}^{k}LL_i(\alpha)-LL_i(\alpha)\right)^2}\right), \qquad (38)$$

which can be more intuitively written as:

$$argmin\left(\frac{1}{k}\sum_{i=1}^{k}LL_i(\alpha)+\frac{std(LL)}{\sqrt{k}}\right) \qquad (39)$$

The reason for doing this is to avoid that biased folds influence too badly in the model outcome. That is why it is more recommendable to use stratified K-fold instead of K-fold cross validation. When using Stratified K-Fold each fold would become a better representative of data and more similar to other folds which will lead in a smaller standard deviation of the Log-likelihood across the folds. Going back to the PLCV results, we can see that the function selects an alpha equal to 0. This is intuetively wrong and thus it is important to evaluate as well the parameters shrinkage for that maximum alpha to see if we are missing any minimum value that we did not considered:

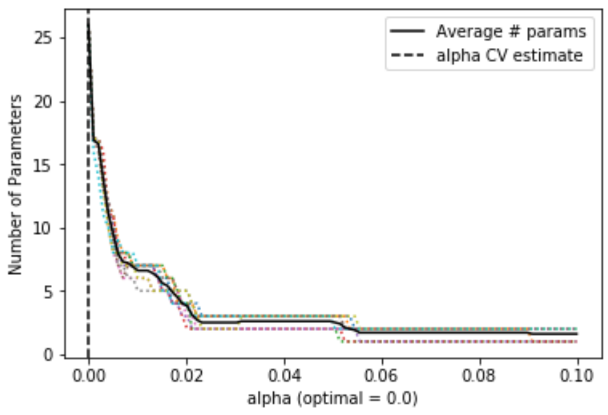


Figure 31: Post Lasso CV with a max alpha of 0.02.

In this graph we can see that all parameters are quickly shrunk between 0 and 0.02. Therefore, we must evaluate this function with different values of maximum α to be able to look for the variance bias trade-off curve. Therefore to be able to spot this phenomenon we must zoom into the range of α that capture it. Looking at the graph we can see that most of the parameters are shrunk from $\alpha = 0$ to $\alpha = 0.001$. As a consequence we will evaluate Lasso between these two values. The results from this small simualtion are shown in the following table:

alpha	N	Log-Likelihood	AR Training Set	AR Test Set
0	27 (all)	-24545	0.49515	0.48740
0.000005	25	-24545	0.49515	0.487414
0.00001	24	-24545	0.49514	0.487423
0.000015	23	-24816	0.486771	0.479379
0.000025	21	-24816	0.4867120	0.479439
0.00005	20	-24816	0.486677	0.479435
0.00025	19	-24825	0.486132	0.479245
0.0005	18	-24848	0.484629	0.477538
0.00055	17	-24853	0.4842485	0.477182
0.00065	16	-24863	0.483643	0.476611
0.00095	15	-24883.	0.482462	0.4756125
0.001	13	-24898	0.481706	0.475152

Table 9: Lasso simulation.

As we can see in the table we can spot a maximum in the accuracy ratio of the test set for an alpha equal to 0.00001 that leads to 24 parameters

9.3.1 Discussion

In the last section we simulated Post Lasso CV for different values of the $l1$ penalization parameter (α). It is very important to mention that the range of α to be evaluated must be chosen carefully. We have seen in the last graphs that Post Lasso CV selected *apparently* an α of 0, and this was only because we were not zooming correctly into the α range.

When we evaluated the range of α that contains the variance-bias curve (0 to 0.0001) we can appreciate (in *Table 9*), that the function selects an α of 0.00001 that selects 24 parameters and has a performance of 0.487423 in the accuracy ratio of the test set. In this way Post Lasso CV has only one weakness and is that the user must study first how the parameters are being shrunk to establish in a logical way the range of α worth to study.

Also, training times must be taken into account, it was faced during the research that this function takes approximately 4 days to output an optimum α. This is due to the solver that the GLM Binomial function[6] is using when fitting the data with regularization. Therefore, it is really important to choose a correct range of alphas (which is done by looking at the shrinkage graph) and also by specifying in the GLM.fit.regularized function a correct maximum number of iterations and convergence tolerance.

Regarding the cross validation method to be used, we can conclude that due to the nature of K-Fold and the methodology of our function, it is better to use Stratified K-Fold due to the distribution it establishes across the folds. Stratified K-Fold is expected to lead to smaller standard errors across the folds which will lead to a more accurate chosen alpha in *Equation 39*.

In the simulation, the cross validation method: Leave-One-Out has not been used due to computational issues. The LOO method performs cross validation considering each sample as a fold. However, the estimated time for computing LOO for PLCV (taking into account range of 50 α and 60.000 samples and 1 second per training) would be 5.14 years. This method would result in the most *safe* cross validation method but it takes a lot of time to compute and thus it has not been considered.

9.4 Selection progress

The last results shown that Lasso and Stepwise perform variable selection for a initial set of variables. It is interesting to see that the model that PLCV builds of 17 variables does not lead to the same results as the model that Stepwise regression builds with 17 variables. This is because the model selection algorithm is significantly different. Lasso and Elastic Net apply some coefficient shrinkage and stepwise selects parameters based on the F-statistic (p-vlaue) of the model. In this section we have built a table that shows the selection progress of stepwise and Lasso, and justifies the difference in the selection progress.

[6]Generalized Linear Models library of Python

	Variable										
# of Varables		27	25	23	21	19	18	17	16	15	13
	constant	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦
	X1	🟦	🟦	🟥	🟥	⬜	⬜	⬜	⬜	⬜	⬜
	X2	🟦	🟦	🟦	🟦	🟥	🟥	🟥	🟥	🟥	🟥
	X3	🟦	🟦	🟦	🟩	🟩	🟩	🟩	🟩	🟩	⬜
	X4	🟦	🟦	🟥	🟥	🟥	🟥	🟥	🟥	🟥	⬜
	X5	🟦	🟦	🟦	🟦	🟦	🟦	🟩	🟩	🟩	🟩
	X6	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦
	X7	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟩	🟩	🟩
	X8	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟥	⬜
	X9	🟦	🟦	🟦	⬜	⬜	⬜	⬜	⬜	⬜	⬜
	X10	🟦	🟥	⬜	⬜	⬜	⬜	⬜	⬜	⬜	⬜
	X11	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟥
	X12	🟦	🟥	🟥	🟥	🟥	⬜	⬜	⬜	⬜	⬜
	X13	🟦	🟩	🟩	🟩	🟩	🟩	🟩	⬜	⬜	⬜
	X14	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟩	🟩
	X15	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦
	X16	🟦	🟦	🟩	🟩	🟩	🟩	⬜	⬜	⬜	⬜
	X17	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦
	X18	🟦	🟩	🟩	🟩	⬜	⬜	⬜	⬜	⬜	⬜
	X19	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦
	X20	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦
	X21	🟦	🟦	🟦	🟦	🟩	⬜	⬜	⬜	⬜	⬜
	X22	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦
	X23	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦
	X24	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦
	X25	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦
	X26	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦

⬜	Variable not selected
🟦	Varaible seleted by both
🟥	Variable selected by Stepwise
🟩	Variable selected by Lasso

Table 10: Selection progress.

In this table we can appreciate how stepwise and Lasso do not follow the same selection path. As explained before this is because of the algorithm's selection procedure (Please revisit *Section 5* for selection criteria).

9.5 Comparison with 10 stratified folds on selected models

In the last sections we used stepwise regression and Lasso as model selectors, we compared their selection procedure and also we tested their resulting models in a test set. However, sometimes only one training set is not enough to capture the real performance of the algorithm in new sets of data. Therefore, in this section we will try to capture the performance of the models built in the last sections, using Stratified Cross validation (10 folds) in data in the following way:

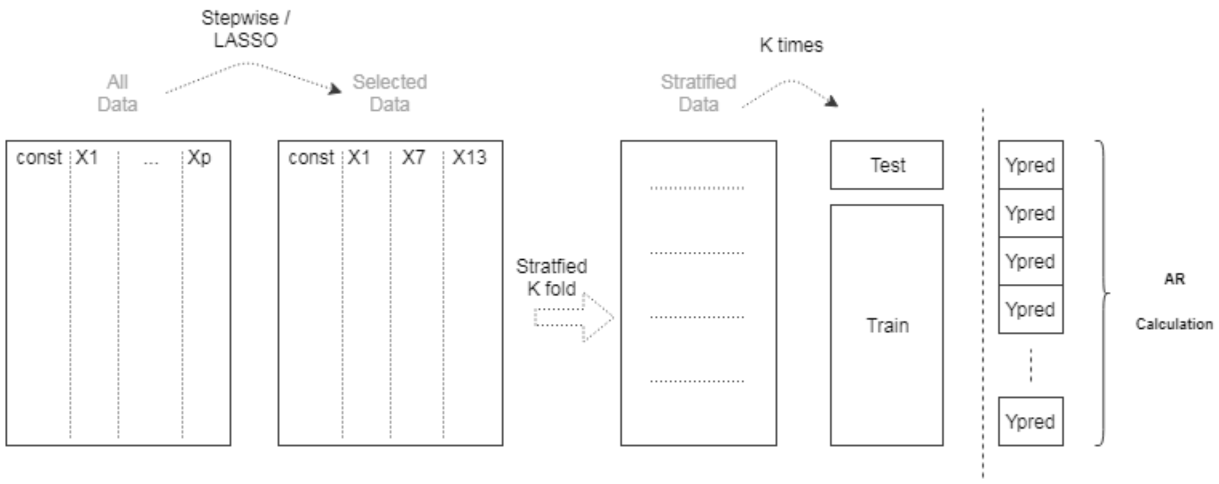


Figure 32: Comparison using 10 stratified folds.

The last picture shows how the accuracy ratio is calculated from different stratified test folds. In such way, we will divide the data into k stratified folds, and predict for each fold using the parameters trained with the other folds. This way each sample will be considered as test set, and thus it will output a more realistic and robust AR.

In the following tables we will show the AR and other parameters calculated using this methodology, for the models already obtained in the last sections:

α	N	AR	Log-Likelihood	AIC
0	27	0.4684230	-30706.64	61461.28
0.000005	25	0.4675377	-30706.64	61461.29
0.00001	25	0.4675377	-30706.66	61461.33
0.000015	25	0.4675377	-30706.69	61461.39
0.000025	23	0.4686447	-30706.93	61461.86
0.00005	21	0.4614357	-31043.52	62135.04
0.00025	19	0.4620104	-31055.13	62158.27
0.0005	16	0.4655152	-31075.96	62199.93
0.00055	16	0.4655152	-31081.54	62211.09
0.00065	15	0.4649976	-31104.12	62256.25
0.00095	15	0.4649976	-31144.23	62336.47
0.001	14	0.46278274	-31151.39	62350.79

Table 11: Results of robust comparison for PLCV.

p enter	p remove	N	AR
0.01	0.05	16	0.47154
0.075	0.1	17	0.47186
0.15	0.2	18	0.4718420

Table 12: Results of robust comparison for Stepwise.

As we can see the selected model for both algorithms remains almost the same, the one with 17 variables for stepwise, and the one with 23 variables for Lasso (before it was 24). However, we can see that the accuracy ratio of Lasso declines with respect to the one in *Section 9.2*, making stepwise the over performing algorithm instead of Lasso. This is due to the use of Lasso as variable selector with refit after the variable selection procedure. This brings instability to the conclusion that Post Lasso CV performs better as a variable selector than stepwise for a given set of data.

9.6 Conclusions

In the last section we have studied the performance of each model selection algorithms, their selection path and also we have made a robust comparison of the models built by both algorithms that brought some useful insights about the performance of both methods.

The last comparison reaches some conclusions that differ from the results obtained earlier in the chapter. The reason is that at the beginning of the chapter we tested the model only with a single test set and in the last chapter we performed Stratified Cross Validation with 10 folds in order to calculate the AR, based on the results of Lasso for each α penalization. In both cases we can appreciate the *curve* which is trying to be minimize, however, there is a difference in one point (0.01) in the calculation of the ratio. This point is significant because it makes the AR's calculated from stepwise higher than the ones from Post Lasso. However, if we would apply Lasso without refitting we would find out that the variance in prediction will decrease, not only because of model selection but because of regularization applied to the parameters that have not been totally shrinked. This will be proposed in the last chapter for future work.

10 Conclusions

In the previous section we performed an exhaustive analysis of the performance of two different model selection algorithms that fitted the same dataset. Each algorithm was as well tested for different input values to check the performance of each algorithm individually. In this section we will draw some conclusions in two levels: for the usage of each algorithm and also for which algorithm to use.

Instability of the accuracy ratio

For stepwise regression we tried different combinations of p values (enter and remove) threshold. Out of 27 variables considered, stepwise regression chooses over 15 to 18 variables in total (for the thresholds used, which are considered to be the most intuitive ones), which means that it significantly reduces the model complexity. However, the results from the accuracy ratio are not stable when analyzing them in a validation set. This can be seen from the values of AR and AR' that differ due to overfitting. The AR difference between the models built with stepwise is significantly high and thus we cannot assure that this accuracy ratio will hold in new raw data. The inconsistency between the training and the validation set results of the model built by stepwise has also been observed by other modelers, for example Judd et al. (2011) that shows on his book that stepwise methods *often fail when applied to new datasets.*

For PLCV happens there is also a difference but it is smaller than in stepwise. It it as well important to consider the last comparison using Stratified K Fold between both algorithms reached some different accuracy ratios for PLCV due to the testing method to compare the gap in Ar between the different models.

Risk of stepwise to converge to local optimum

Performing variable selection we observed that stepwise regression comes up with different subset for each combination of inputs (the names of these variables have not been shown due to confidentiality issues) than Lasso. This also shows that stepwise regression is subtler to converge always to a local optimum. This is due (partially) to the nature of the algorithm that sequentially adds variables to the model. Each step of the algorithm considers the variables yet not included for inclusion. When a variable is included, the optimization path gets conditioned by the combination of variables in the existing model, and thus the final model depends on the order of inclusion. This, as shown in the results, makes the probability of the algorithm to converge to local optimum higher. In the next section a recommendation will be given in order to avoid this problem.

Low P values

Also, stepwise regression for this data in particular leads to rather complex models (16-18 variables). This is inconvenient because it makes the interpretation of the model more difficult, as there are too many variables to consider (and additionally still we can face an overfitting problem). Additionally, if we would like to construct a model with low level of complexity (4-5 variables) we will have to specify thresholds that are not intuitive at all (10^{-50}). This becomes a significant difference between stepwise and Lasso. With the last one we can specify certain levels of Threshold that help to filter out some variables that have a rather small impact in the model. This threshold can be used in a more intuitive way than the p values of stepwise algorithm.

Grid of penalization values for PLCV must be accurate

Using Post Lasso CV we have obtained some interesting results. The accuracy ratio obtained from the simulations outperforms some of p(values)-combinations from stepwise simulations for one test set. This makes Post Lasso CV a more intuitive direct way to reach a better model than stepwise. However, Post Lasso has the problem of the range of α to be evaluated and the computing time it needs to find an optimum regularization parameter.

The parameters were selected vary rapidly with small variations of α. When using PLCV we must ensure that for each step of α there are on to three parameters less in the model. This way the variance - bias trade off is zoomed in and PLCV is able to choose the optimum model complexity.

Stratified K-Fold is the safer CV method

When using Post Lasso CV it is better to perform Stratified K-Fold Cross Validation because it ensures that there is the same percentage of output types in each fold and thus leads to a smaller standard error across the fold. As a consequence, the selection for the chosen value of α is more accurate using Stratified K-Fold, because each fold represents reality better.

Need for robust testing method to draw the comparison

In this section we have compared stepwise and PLCV and we have withdrawn some conclusions for both of them. However, in the first part of last section this comparison was done by comparing the results only on one test set. From this comparison we concluded that some combinations of stepwise outperform some PLCV simulations but that stepwise is not that intuitive to use. However, when we establish a more robust testing method by calculating the prediction on 10 stratified folds we found out that in this case Lasso with refit leads to one point lower accuracy ratio than stepwise regression. This is because of the model refit built by Lasso. This leads to the final conclusion that Lasso if not refited would perform better in prediction than stepwise, but Lasso as a model selector in most of the cases does not perform better.

In the next section we will make some recommendations for the usage of the functions but also for the understanding of the algorithms.

11 Recommendations

After having drawn some conclusions there are a few recommendations that can be given. As it has been shown in the simulations, stepwise can be a useful algorithm for model selection. However, it must be taken into account that its usage is not very easy due to the intuition behind the input parameters. The p values estimated for some of the variables are too low and thus it is difficult to choose the thresholds for forward selection and backward elimination in an intuitive way.

Also stepwise regression is subtler to converge to a local optimum than Lasso, it is because of this that if the user of the algorithm has an idea of which variables should be included in the model, he/she should make use of the optional input parameter: whitelist (to force variables) or inmodel (to initialize variables). This way the user can avoid that the algorithm converges to a local optimum that does not lead to the best model's performance.

When considering the use of Post Lasso Cross Validation it is recommended to use Leave One Out cross validation method if the number of samples is rather small. However, if the number of samples is big and LOO becomes computationally too expensive, we recommend the user to use Stratified K-Fold instead of K-Fold in order to decrease the standard error of the cost calculated across the folds.

Also the user must take into account that the higher the number of folds considered the results become more stable (for a number of folds equal to samples : LOO). However, at the same the user must consider computational time. In addition some studies like Hussein focused on approximating LOO for logistic regression, this should be considered for future study. It is expected that LOO leads to more accurate cross validation results.

As a final recommendation, Post Lasso CV is an easier (in terms of usage) algorithm to perform model selection than stepwise, and it leads in some cases to better results. However, it is recommended as well to do a pre-study of the shrinkage dynamic of Lasso regression in order to test Post Lasso with the correct range of alphas. If this is not done correctly, Post Lasso CV may recommend to use a penalization of 0. If this happens the user must consider to reduce the max-alpha input parameter of PLCV and choose one according to the shrinkage dynamic.

Therefore we recommend to use instead of stepwise regression as a model selector, if the user knows how to initialize stepwise and how to choose these *low* p values. PLCV model selector shows a slightly lower AR in some cases due to the refitted model.

12 Future Work

In this thesis several aspects of model selection have been addressed. However, the literature always keeps updating and thus other model selection methods can be considered. We propose for further study the following topics:

- Using the recently proposed Efficient Determination Criterion (Resende and Dorea, 2016) to score different models that fit the same data in order to choose the one one with a better score. Also it will be very interesting to compare the performance of EDC to AIC and BIC
- Study the impact of using Leave one Out cross validation method instead of Stratified K-Fold and K-Fold for Post Lasso CV (Hussein)
- Study the model selection via the Elastic Net Cross Validation and compare its performance with Post Lasso and Stepwise, which was not done in this thesis due to time constraints.
- Test Post Lasso CV with a dataset that has variables not relevant at all for the model to see how fast does Post Lasso CV cleans these variables. In the same way, it will be interesting to add to the original dataset totally random variables (noise) to see as well how does Lasso cleans the dataset.
- Develop Bruteforce in order to avoid consider combination of variables that don't make sense to consider. For example, totally correlated variables.
- Include automatic data cleaning procedures in the programmed algorithms in order to increase their robustness.
- Optimize the GLM. fit regularized Post Lasso CV solver for penalized Logistic regression. The actual one works very slow in is not optimum for the function's computing time.
- Use Post Lasso CV methodology without refitting the model and compare its performance to stepwise in terms of bias and variance using a bootstrapping method across the folds for the calculation of AR.

References

Akaike, H. (1974). A new look at the statistical model identification. *IEEE transactions on automatic control*, 19(6):716–723.

Anders, U. and Korn, O. (1999). Model selection in neural networks. *Neural networks*, 12(2):309–323.

Bessis, J. (2015). *Risk management in banking.* John Wiley & Sons.

Boisbunon, A., Canu, S., Fourdrinier, D., Strawderman, W., and Wells, M. T. (2014). Akaike's information criterion, cp and estimators of loss for elliptically symmetric distributions. *International Statistical Review*, 82(3):422–439.

Box, G. E. and Draper, N. R. (1987). *Empirical model-building and response surfaces.* John Wiley & Sons.

Breiman, L. (1995). Better subset regression using the nonnegative garrote. *Technometrics*, 37(4):373–384.

Breiman, L. (2001). Random forests. *Machine learning*, 45(1):5–32.

Burnham, K. P. and Anderson, D. R. (2003). *Model selection and multimodel inference: a practical information-theoretic approach.* Springer Science & Business Media.

Burnham, K. P. and Anderson, D. R. (2004). Multimodel inference: understanding aic and bic in model selection. *Sociological methods & research*, 33(2):261–304.

Candes, E., Tao, T., et al. (2007). The dantzig selector: Statistical estimation when p is much larger than n. *The Annals of Statistics*, 35(6):2313–2351.

Candès, E. J. and Tao, T. (2010). The power of convex relaxation: Near-optimal matrix completion. *IEEE Transactions on Information Theory*, 56(5):2053–2080.

Cavanaugh, J. E. (1997). Unifying the derivations for the akaike and corrected akaike information criteria. *Statistics & Probability Letters*, 33(2):201–208.

Donoho, D. L. (2006). Compressed sensing. *IEEE Transactions on information theory*, 52(4):1289–1306.

Efroymson, M. (1960). Multiple regression analysis. *Mathematical methods for digital computers*, pages 191–203.

Flom, P. L. and Cassell, D. L. (2007). Stopping stepwise: Why stepwise and similar selection methods are bad, and what you should use. In *NorthEast SAS Users Group Inc 20th Annual Conference: 11-14th November 2007; Baltimore, Maryland.*

Flores, J. A. E., Basualdo, T. L., and Sordo, A. R. Q. (2010). *Regulatory Use of System-wide Estimations of PD, LGD and EAD: FSI Award 2010 Winning Paper.* Financial Stability Inst., Bank for Internat. Settlements.

Fortmann-Roe, S. (2012). Understanding the bias-variance tradeoff.

Ho, T. K. (1995). Random decision forests. In *Document analysis and recognition, 1995., proceedings of the third international conference on*, volume 1, pages 278–282. IEEE.

Hoerl, A. E. and Kennard, R. W. (1970). Ridge regression: Biased estimation for nonorthogonal problems. *Technometrics*, 12(1):55–67.

Hurvich, C. M. and Tsai, C.-L. (1989). Regression and time series model selection in small samples. *Biometrika*, 76(2):297–307.

Hussein, M. H. Computing the approximate and exact leaving-one-out method in multiple group logistic discrimination with the sas system.

Jolliffe, I. T., Trendafilov, N. T., and Uddin, M. (2003). A modified principal component technique based on the lasso. *Journal of computational and Graphical Statistics*, 12(3):531–547.

Judd, C. M., McClelland, G. H., and Ryan, C. S. (2011). *Data analysis: A model comparison approach.* Routledge.

Liaw, A. (2013). Documentation for r package randomforest. *Retrieved.*

Mallows, C. L. (1973). Some comments on c p. *Technometrics*, 15(4):661–675.

McCulloch, W. S. and Pitts, W. (1943). A logical calculus of the ideas immanent in nervous activity. *The bulletin of mathematical biophysics*, 5(4):115–133.

Neath, A. A. and Cavanaugh, J. E. (2012). The bayesian information criterion: background, derivation, and applications. *Wiley Interdisciplinary Reviews: Computational Statistics*, 4(2):199–203.

Resende, P. A. A. and Dorea, C. C. Y. (2016). Model identification using the efficient determination criterion. *Journal of Multivariate Analysis*, 150:229–244.

Rolling, C. A. and Yang, Y. (2014). Model selection for estimating treatment effects. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 76(4):749–769.

Sammut, C. and Webb, G. I., editors (2017). *Bias Variance Decomposition*, pages 128–129. Springer US, Boston, MA.

Schwarz, G. et al. (1978). Estimating the dimension of a model. *The annals of statistics*, 6(2):461–464.

Seber, G. A. and Lee, A. J. (2012). *Linear regression analysis*, volume 329. John Wiley & Sons.

Stigler, S. M. (2007). The epic story of maximum likelihood. *Statistical Science*, pages 598–620.

Tibshirani, R. (1996). Regression shrinkage and selection via the lasso. *Journal of the*

Royal Statistical Society. Series B (Methodological), pages 267–288.

Tibshirani, R. (2011). Regression shrinkage and selection via the lasso: a retrospective. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 73(3):273–282.

Tibshirani, R., Saunders, M., Rosset, S., Zhu, J., and Knight, K. (2005). Sparsity and smoothness via the fused lasso. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 67(1):91–108.

Tibshirani, R. J., Hoefling, H., and Tibshirani, R. (2011). Nearly-isotonic regression. *Technometrics*, 53(1):54–61.

Trevor, B. E., Hastie, T., Johnstone, L., and Tibshirani, R. (2002). Least angle regression. In *Annals of Statistics*. Citeseer.

Vrieze, S. I. (2012). Model selection and psychological theory: a discussion of the differences between the akaike information criterion (aic) and the bayesian information criterion (bic). *Psychological methods*, 17(2):228.

Yang, Y. (2005). Can the strengths of aic and bic be shared? a conflict between model indentification and regression estimation. *Biometrika*, 92(4):937–950.

Yuan, M. and Lin, Y. (2006). Model selection and estimation in regression with grouped variables. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 68(1):49–67.

Yuan, M. and Lin, Y. (2007). Model selection and estimation in the gaussian graphical model. *Biometrika*, 94(1):19–35.

Zou, H. (2006). The adaptive lasso and its oracle properties. *Journal of the American statistical association*, 101(476):1418–1429.

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