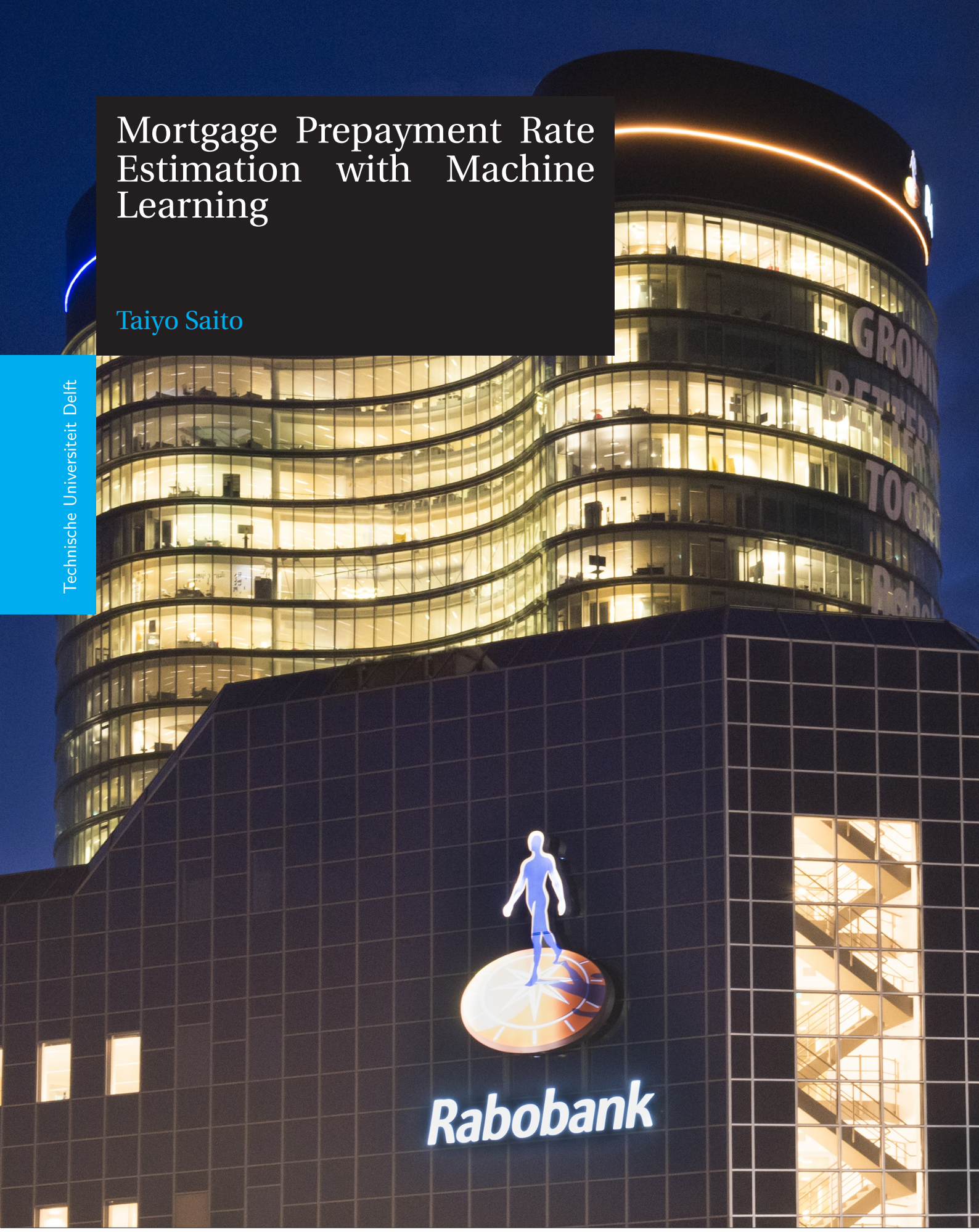
Mortgage Prepayment Rate Estimation with Machine Learning

Taiyo Saito

Technische Universiteit Delft



TU Delft Delft University of Technology

Challenge the future

Mortgage Prepayment Rate Estimation with Machine Learning

Author:
TAIYO SAITO (4626869)

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE OF

MASTER OF SCIENCE
IN APPLIED MATHEMATICS

AT THE DELFT UNIVERSITY OF TECHNOLOGY

Delft, July 13, 2018

Supervisor:	Prof. Dr. Ir. C.W. Oosterlee,	TU Delft
Thesis committee	Dr. Ir. L.A. Grzelak,	TU Delft
	Dr. P. Cirillo,	TU Delft
	Ir. F. Mulder,	Rabobank

An electronic version of this thesis is available at http://repository.tudelft.nl/.

Rabobank

TU Delft

ABSTRACT

The aim of this thesis is to forecast the evolution of the prepayment rate in a mortgage portfolio. In the Netherlands, people with a loan have the possibility to repay (part of) their outstanding loan before the due date. These prepayments make the length of the portfolio of loans stochastic, which creates problems in the refinancing policy of the bank, and affects the Asset & Liability Management. Moreover, interest rate risk arises from prepayments, meaning that being able to forecast the prepayment rate can increase the performance of the hedging strategy of a bank.
Given the magnitude of the mortgage portfolio in the balance sheet of a bank, estimating the prepayment rate is therefore crucial. There are two kinds of models in the literature, the optimal prepayment model, which sees prepayment as a consequence of rational behavior (e.g. prepayments are always exercised at an optimal time), and the exogenous model which also takes into account other macroeconomic variables, client specifics and loan characteristics. Our focus will be on the second kind of techniques, precisely we will approach the problem as a classification task that will be carried out with two different machine learning techniques: Random Forests and Artificial Neural Networks. Since prepayments are rare events, this leads to an imbalanced data set framework. The imbalance between classes creates complications in the development of the algorithm, hence ad hoc corrections are applied to solve them.

Keywords: Neural networks, Random Forests, Conditional Prepayment Rate, Imbalanced Data Set, Mortgages.

ACKNOWLEDGEMENT

This thesis has been submitted in partial fulfillment of the requirements for the degree of Master of Science in Applied Mathematics at Delft University of Technology. The academic supervisor of the project has been Prof. dr. ir. C.W. Oosterlee of the Numerical Analysis group at the Delft Institute of Applied Mathematics. The research has been conducted under the supervision of L.A. Grzelak and F. Mulder at the Rabobank office in Utrecht. This work has been promoted by the Interest Rate Risk management team with the intent of achieving a better evaluation of the interest rate position of Rabobank.

First of all, I would like to thank my supervisor Kees Oosterlee for his invaluable help, his patience and his feedbacks throughout the whole development of my thesis. Then, I would like to express my gratitude for my professor and daily supervisor Lech Grzelak for letting me know about this project and for his important advices whenever I needed one. Moreover, I would like to thank Frank Mulder for allowing me to join his team, and for the important suggestions he supported me with during my stay in Rabobank. In addition, many thanks to Sander Bohte from CWI for his fundamental help on the development of the models. Last, a thanks to my friends, who helped me keep going on with a smile during some hard moments, and a special thank you to my family, who always believed in me and supported me giving me the chance to accomplish what I have done so far.

Taiyo Saito
Delft, July 2018

CONTENTS

1

INTRODUCTION

A mortgage is the most common way to obtain a consistent amount of money in a short time period to cover a big expense, e.g. the purchase of a house. A mortgage is a contract between two parties, the mortgagee which usually is a bank and the mortgagor which is the person or company who asks for the loan. The mortgagee is the lender, the one that lends the money, and in exchange of this service receives an interest part calculated on the amount (notional) borrowed by the mortgagor.
Mortgagors have the possibility to repay part of the notional before the agreed time, these anticipated payments of the notional part are called prepayments. This opportunity to prepay makes the duration of a portfolio of mortgages stochastic. It could be that the expected cash flows for a mortgage do not match the actual registered cash flows. Therefore, the prepayment option is a risk for the lender. In particular, two kinds of risks arise from it: liquidity risk and interest rate risk.

In order to cover itself against the prepayment risk, a bank must be able to forecast the prepayment rate for the coming years in order to implement some effective hedging strategies.
To predict when people are going to prepay, two main approaches are available: an optional theoretical model and an exogenous model. The first approach only takes into account financial aspects, therefore according to this model people will prepay only when it is economically convenient, e.g. the interest rate in the market is lower than the one paid on the mortgage. However, people do not always follow a rational reasoning when it comes to prepay their mortgage, so a model of this kind is not able to thoroughly explain the general prepayment behavior. The latter approach is instead able to take into account both financial and non-financial related information about the clients. These are methods based on data analysis. A data set with information related to the mortgages, the clients and macroeconomic factors is needed in order to generate such a model.

In this thesis, our focus is on exogenous configurations. The purpose of our work is to develop an innovative model which is able to predict the conditional prepayment rate for a portfolio of mortgages given a set of input features.
To fulfill our goal, we will use two common machine learning techniques: artificial neural networks and random forests. Although we are in the context of time series forecasting, we decide to tackle this problem as a classification task. To solve it, models based on logistic regression are already implemented. We will to take these models a step further by using more complex statistical tools which should be able to achieve better performance due to their capability of learning complicated nonlinear relationships between the input features and the output target.
Following this approach, the data set that we have can be divided in two classes: the class of the

observations that have prepaid and the class of observation that have not prepaid. Prepayment observations are definitely less frequent than the observations for non prepayments. More precisely, prepayment samples account for less than the 2% of the total number of available observations. This means that the dimensions of the two classes are really different and this class imbalance leads to several problems in the development of the models. In fact, classification algorithms are in general biased towards the majority class. This problem can be tackled with some specific corrections applied both to the algorithm and to the data set.

The novelties stay in the fact that with these models we are able to tackle the prepayment problem both at a single loan level and at a portfolio level, and we achieve that by applying some statistical techniques to let the algorithms learn in an imbalanced framework as well. This set of techniques is largely used in practice, for instance in medicine for disease detection or also in banks for fraud detection, where having an imbalanced dataset is standard. We apply those procedures to the prepayment problem in mortgages in an innovative way. In particular, how these corrections affect the algorithms is investigated in order to maintain the marginal probabilities in the output of the models, which is a fundamental property we want to preserve in our framework. This aspect is not treated in the other settings since the interest is always on the single observation and never on the whole average level.
The general background on which the model is thought and developed is the Dutch mortgage market. This model is tailored to some specific characteristics that derive from the structure of Dutch mortgages. For instance, the presence or absence of penalties for prepayments. In the Netherlands people can prepay up to 20% of the notional each year without a penalty, and prepayments due to movement are always free of charge.
Moreover, the model has been adapted to the particular kind of portfolio that we are dealing with. Therefore, tailor made variables are added to the models in order to enhance the performance when applying the models to our framework.

1.1. OUTLINE

In chapter 2, we start with an introduction of the Dutch mortgage market followed by general notions regarding mortgages and prepayments needed to understand the rest of the work. Then, we introduce the prepayment risk and explain how it affects the balance sheet of a bank. Last, we give a description of the existing literature about prepayments. In particular, the most common causes for prepayment in the literature are introduced and an idea of the existing prepayment models, based both on survival analysis and logistic regression, is given to the reader.

Chapter 3 can be divided in two parts. The first part starts with a description of the data set. A brief description of all the variables is then given. Then, some plots that show the distribution of the prepayment observations against some of the important variables are displayed.
In the second part of the chapter, the problem of having an imbalanced data set is explained and the various techniques to tackle this complication are introduced. Last, the impact of these modifications is taken into account.

In chapter 4, we give the reader the required theoretical background in order to understand the models that we develop. It starts with a general description of data analysis techniques and of different kinds of learning. Then, the focus goes to artificial neural networks, in particular to fully connected feedforward networks. A section with details about the adaptation of the algorithm to a rare class problem is present as well.
At this point, we also introduce decision trees and the theory behind them. This is done because trees are at the base of random forests. In fact, random forests are the second type of machine learning techniques that we use, and they are found to be a collection of decision trees. Also for random forests, a section with an explanation of the correction for rare class problems is present.

Chapter 5 starts with a motivation for the choice of the approaches, followed by a detailed descrip-

tion of the model for the forecasting of the prepayment rate. Then, there is a section related to the practical implementation of the models.
The chapter ends with a presentation of the results and of the performance obtained by the models in predicting the prepayment rate. Moreover, also a sensitivity analysis is performed in order to assess the confidence in the final output.

We conclude in chapter 6. First, we sum up the findings and the results of our work and give our final considerations. Then, there is a section with indications for possible directions to follow for future research on this topic.

2

MORTGAGES AND PREPAYMENTS

In this chapter we will give an overview of the Dutch mortgage market and explain what is prepayment and why it is a risk for the bank.

2.1. DUTCH MORTGAGE MARKET

In the Dutch market, the mortgage portfolio of lenders is formed by 7.7 million households. 83% of the Dutch homeowners has a mortgage debt with their house as a collateral. The total mortgage debt was EUR 637 billion in 2013 [1]. The mortgages have fixed a interest rate for a period of time, chosen by the borrower, which can vary between one and thirty years. In general, fixed interest rates for shorter times are lower than the ones for longer periods. Usually, the fixed time is five or ten years. A new interest rate is set on the reset date [2].
The mortgage market in the Netherlands is strongly influenced by tax facilities and a strictly regulated rental segment. As reported by the Nederlandse Vereniging van Banken (NVB) in [1], just 56% of the houses are in the owner-occupied segment and that is a low percentage compared to other countries in the euro-zone. Moreover, interest paid in loans for the main house is deductible from pre-tax income for thirty years. It means that Dutch mortgagors use up to 51% of their taxation to pay the interests on the loan. This factor has substantially increased the homeownership from 48% in 1993 to 56% in 2013. Clearly, these are not the only factors that have driven this rising demand. Increasing incomes, very low interest rates and the possibility of basing the mortgage on two incomes have also fueled this growth.
The tax benefit ensures that most mortgages are constructed to obtain the maximum gain from this taxation system. This is translated in the stipulation of interest only (I/O) mortgages, in which no payment of the notional is due during the loan period. These kinds of mortgages become popular in the eighties and are characterized by a life insurance. Since the crisis in 2008, savings accounts in which the sum to repay the loan is accumulated are accompanied to I/O mortgages. In such mortgages, no repayment happens during the lifetime of the loan, therefore the benefit from the tax deduction is the greatest possible since the notional is not reduced, but at the same time this account guarantees the repayment of the entire notional at the end date. Moreover, from a mortgagor point of view, the rate paid every month (interest+notional in the savings account-interest received from the savings account) is the same as for an annuity. Since 2013 regulations about tax deductibility have changed, deductibility is possible only in annuity mortgages and this change has made I/O mortgages unattractive.

A highly regulated rental segment and the tax facilities have led to high portfolio Loan to Value (LTV) ratios and created a large difference between Loan to Income (LTI) ratios calculated on net and gross income. Threshold values for the LTI indicator at origination are calculated each year based

on interest rate and purchasing power. In 2014, a cap of 104% has been set for LTV, this maximum will be decreased to 100% in 2018. These laws are, according to the OECD (2011) reports, ones of the most strict versions in Europe. LTV ratios have increased by 20% from 2006 to 2013 mainly due to a constant decline of the house value.
Using these two variables it is possible to identify four buckets of mortgages: one with low LTV and low LTI which has low default risk, one with low LTV and high LTI which is more risky but losses in case of default are usually low, one with high LTV and low LTI which is as risky as the second group but will incur in a greater loss, and finally the riskiest group which is the one formed by high LTV and high LTI [3]. Of course high values of LTI and LTV in themselves are not risky, they are just indicators of possible consequences in case of a negative economic situation like a crisis. For instance, a high LTV ratio means that there is a higher risk that the mortgagor will not be able to repay the debt if he/she decides to sell the house, whether it is for a personal choice or forced by the bank. If the notional of the loan exceeds the value of the underlying house, it is said that "the households suffer from negative home equity, their mortgages are 'underwater'"[1]. This situation is not necessarily bad for the lender, but it slows down the housing market. Another factor which contributes to that is portability. In fact, most of the mortgages in the Netherlands are portable, which is translated by the fact that a borrower who decides to move to a new house can transfer the mortgage from one house to the other.

2.1.1. DIFFERENT KINDS OF MORTGAGES

In general, mortgages in the Netherlands have a fixed rate for a determined period. This fixed period could have different lengths, from one to thirty years, but the more common choice is a value between five and ten years. The end of this period takes the name of reset date, since there is a reset of the interest rate.
The three most common kinds of mortgages in the Netherlands, regarding the composition of the monthly installments, are:

- Annuity mortgage: it has fixed monthly payments for the length of the fixed interest rate period. The amount is fixed, but its composition changes. Close to the start of the mortgage, most of the amount is made up by interest and just a small amount is repayment. With these payments the remaining notional decreases and so does the interest to be paid. Therefore, as time passes, the interest part gets smaller and more and more of that monthly amount goes for repaying the notional.

- Linear mortgage: it has decreasing monthly payments. In fact, every month the borrower pays a fixed amount of notional plus the interest part which decreases every time. This means that it has higher initial payments and reduces the actual debt faster than an annuity.

- Bullet mortgage: it has fixed monthly payments which consist only in the interest part. The notional is entirely repaid at the end of the mortgage. This kind of mortgage was often used in the past due to the high tax incentives given by the state on loan interests. Usually, this kind of loan is associated with a money savings account in which people put the money to repay the notional at maturity.

It is also possible to choose whether to have a fixed or variable interest rate. If it is chosen to be variable, it will follow the current rate of the lender, if it is chosen fixed, it will stay the same for a determined period of time. Usually, most of the mortgagors opt for a fixed rate, which is slightly higher, for several years in order to mitigate interest rate risk.

2.2. PREPAYMENT RISK

Mortgagors have the possibility to repay (part of) the notional before the due date, these kinds of anticipated and unexpected payments are called prepayments. This makes the duration of the mort-

gage portfolio stochastic, which creates complications in the refinancing policy of the lender (the mortgagee). The bank is usually interested in the prepayment rate on a portfolio level, not of a single mortgage.
Prepayment is a risk from the point of view of the issuer of the loan. A prepayment option is an option which "reflects the difference between the value of the outstanding loan repayments at the interest rate at the time of prepayment for the remaining term of the loan minus the amount of the loan then outstanding (which is the value of the outstanding loan repayments at the original loan interest rate)" [4]. If the interest rate goes down then the option is in the money (positive payoff), if it goes up then the option is out of the money. The exercise price of this call option is equal to the loan outstanding. Rationally, the call should be exercised only if the difference mentioned above is positive since, otherwise, the option is worth more if you don't exercise it. Meaning that it is more convenient to put money in a savings account rather than prepay the mortgage. People do not always act economically rationally though, from now on we will use the term non-rational to identify something that differs from the economically rational behavior explained earlier.

An American call option on an amortizing reducing term swap could be a good instrument to hedge the prepayment option, but the ideal hedge instrument may not exist. For instance, even assuming that the durations of the financing and of the loan are perfectly matched, suppose a 20 years mortgage at 4% is financed with a 20 years bond paying 3.5%. If the interest goes down to 3%, the borrower will refinance at the new rate prepaying the old mortgage and opening a new one at the lower rate. However, the bank will still have the obligation of paying 3.5% interest on the bond used to finance the first loan. Moreover, since the price of the perfect hedge instruments is equal to the cost of the prepayment option, if this instrument existed, it could be used to price the option [4].
The possibility to prepay exposes the bank to prepayment risk, which can be divided in two classes [5]:

- Interest rate risk: it arises because fixed rate mortgages are usually hedged against interest rates changes using Interest Rate Swaps (IRS). The bank receives the floating rate and pays the fixed one, this means that a prepayment may expose the bank to the possibility of paying a fixed rate on the derivative which is higher than the one obtained on the new loans.

- Liquidity risk: it arises from the fact that the liquidity profile of a bank is strongly influenced by the maturity profile of mortgages conditional to prepayments. An incorrect estimation of this quantity leads to the risk of under or over funding and of higher long-term liquidity costs.

Since prepayments are such a big risk for the bank, it is fundamental to be able to predict the conditional prepayment rate (CPR), which is the proportion of the principal of a pool of loans assumed to prepay in each period, in order to better forecast future cash flows.
When modeling mortgage prepayments, as touched upon earlier, we cannot just use rational principles to evaluate contracts (e.g. prepay only when the interest rate goes down), but we have to take into account also behavioral aspects, and hence have a behavioral approach. In such a model we take into account factors usually estimated through statistical analysis, in order to explain effects that otherwise could not have been explained. These kinds of techniques are called exogenous models, in opposition to optimal prepayment models that see prepayments just as a consequence of rational behavior, e.g. prepayment is always exercised at an optimal time by the borrower.
For these reasons, rather than using a rational prepayment model for which the prepayment rate depends just on the rate evolution, an empirical model which is capable of taking into account other factors is chosen. This CPR is a function of some variables like the refinance incentive, the seasoning, the age of the loan, etc, and can be estimated by means of some statistical tools which we will explain later. The output of these models is then used to build a hedging portfolio with instruments that have a notional that adjusts according to this CPR.

2.2.1. DIFFERENT KINDS OF PREPAYMENTS

Besides prepayments, we can also have a loss of part of the notional. This is what is called a default event, and the amount lost takes the name of loss given default (LGD). From an interest rate perspective, this is different from a prepayment event though, that is because there is no payment.
Prepayments are gathered in two main categories inside a bank's Assets & Liabilities Management (ALM) department [6]:

- Redemption: it is when a customer pays back part of the notional. Redemptions may be scheduled as part of the amortization scheme, or not. If they happen on the mortgage rate reset date, then there is no interest rate risk since the entire notional would have been repriced anyway. However, if they do not occur on a reset date, then it is early and we have interest rate risk. Early full redemptions can occur due to the death of the client, the insurance pay-out following the collateral destruction, movement to another house, or voluntarily. A partial early redemption is always voluntary.

- Repricing: it is when the interest rate on a mortgage changes. This could be due to the amortization scheme, the remaining interest rate fixed period, or the client rating changes. Repricing could be both scheduled and unscheduled. Unscheduled ones could happen due to movement, salvaging after default or death of one of the debtors, when the client has the possibility to reprice or to stick to the current client rate, or even voluntarily by request of the client.

2.2.2. PREPAYMENTS AND PENALTIES

A prepayment can occur with or without a penalty. Usually in the Netherlands it is possible to prepay up to 20% (10% in certain kinds of contracts) a year without penalty and prepayments due to movement are always free of penalty. Therefore, penalties may occur for voluntary early partial or full redemptions, and for voluntary repricing if they exceed the threshold.
If there is a penalty, prepayments do not bear interest rate risk, because theoretically the present value stays the same and therefore there is no delta associated to the prepayment. This is due to the fact that penalties should be enough to cover the loss derived from the prepayment. However, the expected cash flows change relevantly, therefore penalized prepayments are still relevant and need to be taken into consideration in a different way from free prepayments.
Moreover, prepayments that happen at a reset date are penalty free.

2.2.3. CONDITIONAL PREPAYMENT RATE

The conditional prepayment rate is the mortgage prepayment rate and is equal to the proportion of the pool of loans notional that is assumed to be prepaid in each period. The CPR is an annual quantity and the higher it is, the faster mortgagors are expected to repay their loans.
Considering that mortgages could prepay monthly, we need a prepayment rate which is updated monthly as well. This quantity indicates the monthly prepayment rate of the pool of mortgages and takes the name of *single monthly mortality rate* (SMM). It is defined as the amount prepaid during that month divided by the notional then outstanding. It is related to the CPR by the following relation:

$$\text{CPR} = 1 - (1 - SMM)^{12}. \tag{2.1}$$

2.2.4. CPR AND HEDGING

In this section, we explain more technically why it is important to know the future values of the CPR and how it comes into play in a bank strategy.
In order to reduce the interest rate risk introduced by prepayments, banks hedge their position by pairing each mortgage with interest rate swaps. However, if a prepayment occurs, the expected cash flows of the hedge do not match the ones of the loan anymore. In order to fix this problem in our

hedging strategy, we can use an Index Amortizing Swap (IAS) that receives the prepayment rate as input. This way the notional adjusts accordingly to the one of the portfolio of mortgages, and therefore the bank is able to exchange the correct amount of the fixed rate for the floating one in the hedging strategy.
We now explain in the detail how an IAS works in order to clarify where the CPR is plugged in in the strategy. Index Amortizing Swaps are over the counter interest rate swaps. Their amortization scheme is not deterministic but determined by a function of one or more variables such as the interest rate, the seasonality, or other endogenous characteristics. In particular, we want the notional of the IAS to to replicate the notional of the mortgage, so as in Equation 2.2:

$$N_{t_i} = N_{t_{i-1}}(1 - \mathrm{CP} - \mathrm{CPR}), \tag{2.2}$$

where N_{t_i} is the notional at time i, and CP is the quantity that takes into account the contractual payments. We see immediately that in order to calculate that amount, the only quantity that is not known is the CPR.
This is where the prepayment models come into play. Using them it is possible to derive the CPR which is then used to model the notional of the IAS that replicates the loan in the hedging strategy.

2.3. GENERAL DRIVING FACTORS

There are many different determinants for prepayments, in fact many variables have influence over the borrowers' decision to prepay or not. These factors could be related to:

- Borrower specific characteristics: age, kind of employment, marital status, income.
- Loan specific characteristics: notional, mortgage age, mortgage rate, collateral, penalty.
- Macroeconomic information: unemployment rate, housing market, divorce rate, month of the year.

Some of the incisive variables are the following ones.
As shown in [2], the *refinancing incentive* is a key determinant of prepayment. A precipitous decrease of market rates may spark the prepayment option. Not the whole pool of mortgages will act as one, some mortgagors will refinance fast, others will not since the refinancing incentive is sometimes non-rationally managed. This phenomenon is known in literature as the burnout effect, and it is due to the difference in borrowers' behavior. In front of a refinancing incentive some will prepay, these are the fast and aware mortgagors. The rest of the mortgagors in the pool may be the slow and not aware ones. In the future, therefore, the pool will be less sensitive to other refinancing occasions, the pool of loans is burned out [7].
Another possible driver of prepayment is related to the loan age. The *Seasoning effect* represents the probability of selling the house as the loan age increases [2]. Usually, a S-shaped relation exists between the loan and its age. In fact, prepayments rarely occur immediately after origination, then they will start to grow until they reach a peak, and eventually they will reach a steady level till maturity. This seasoning is also related to the housing market, during a consistent economic growth more money will circulate and more houses will be sold leading to more prepayments due to relocation. In averse market conditions, these kinds of prepayments will be fewer, and prepayment caused by default will increase.
The presence of some kind of penalty influences the prepayment behavior as well. In fact, if the spread between the current interest rate paid on the mortgage and the market value is not big enough, it could be that the gain from refinancing the loan is not large enough to cover the penalty applied in case of prepayment. In the Netherlands no penalty is applied in case of prepayment when moving to a new house, and it is possible to prepay up to 20% of the original loan every year penalty-free.

Prepayments are also correlated with the month of the year, this effect is called seasonality. This could be linked to the fact of a cyclical trend in house sales. Moreover, from our data, the months December and January have a particular effect on voluntary prepayments. The explanation of this could be that in December there are more curtailments and relocations due to a tax related cause. In fact, in this way closing costs, which are tax deductible, paid before the end of the year could be deducted from the income of that year. In addition, workers receive Christmas bonuses in December and it is not unlikely that a prepayment which happened in late December is registered in January.
Housing turnover is another cause of prepayments. In fact, selling a home will usually trigger a prepayment, unless the loan is portable or is assumable, which means that the new owner of the house can assume the obligations of the loan stipulated by the old owner of the house. Even if the mortgage has those two properties, it is still possible a prepayment will occur due to more favorable interest rate at the moment of the selling.
Other variables like the LTV (Loan To Value) ratio and the geographic location seem to be important from [8]. It is observed that loans with low LTV are more likely to prepay opposed to loans with higher LVT. Moreover, mortgagors behavior is influenced by macroeconomic factors, such as the unemployment rate or the house prices, which are geographically sensitive.

Another factor that could influence the prepayments is the coupon or lock-in effect [9]. It is the effect of a low coupon on the ongoing loan which results in a slowing down of turnover related speeds. In fact, if the coupon is low compared to the actual market rate, people who move will keep their current mortgage if it is portable or transfer it to the new owner if it is assumable. Nevertheless, even if the loan is not portable nor assumable, the big difference between the present coupon and the one on the mortgage could dissuade the borrower to move.
It has been observed that mortgagors behavior is influenced by their point of view on current rates compared to the past. When the rates go down to multiyear lows, prepayments due to refinancing grow more compared to an increase in the refinancing incentive only. This effect has been named the *media effect* [9]. It looks like information about these lows in loan rates makes the people realize more about their chances to refinance, and at the same time pushes mortgagees to solicit possible refinancers.
In [10] is shown that a small outstanding principal balance is an indication of more prepayments. In addition, the purpose and the kind of the loan are also influential. They found out that mortgages originated to purchase a home have higher possibility of prepayment. Also, the down payment made at the origination of the loan influences its contract rate, this means that a big down payment will imply a lower rate and, therefore, a resistance to prepayment.
Another factor that could be important in order to forecast prepayment behavior is whether or not the loan has been originated through intermediaries. In particular, in [11] it is said that loans obtained through intermediaries are more likely to prepay since these intermediaries have the information about the mortgage and are more alert to a good opportunity to refinance since they will receive a commission on that.

2.4. EXISTING PREPAYMENT MODELS

As touched upon earlier, there are two main approaches to prepayment rate estimation. One that uses optional theoretical models and one that uses exogenous models. The first kind of approach just contemplates financial considerations, which means that a mortgagor would prepay only if the actual value of the asset is greater than the remaining mortgage notional plus the transaction costs. Basically, the prepayment option is seen as a call option, which is assumed to be exercised only under optimal conditions. This kind of approach removes all the irrational behavior though. It has been shown in [5] that including behavioral aspects in the model helps obtaining better overall results. In fact, the data reveals that often mortgages are prepaid when the rate on the loan is lower than the prevailing mortgage rate in the market, and it is not uncommon that mortgagors do not prepay when

it may be optimal to do so. This is particularly true for the retail mortgage market in which the average mortgagor does not have knowledge of market theory.
This leads us to the second kind of models, the exogenous ones which in turn are divided in two main categories based on the two most used statistical frameworks that are applied. These two models are survival analysis and logistic regression. In the last years, approaches using machine learning algorithm are also considered [8].

2.4.1. SURVIVAL ANALYSIS

In this framework, we are interested in finding the time to event, in this case the prepayment event, and its distribution which is dependent on the hazard rate [12]. With the survival analysis we are able to obtain the probability distribution for the duration of the mortgage, in fact we can model the time until the occurrence of a certain event. The link between the survival function and the covariates is expressed on the basis of the proportional hazard model. The hazard rate is given by the probability of a mortgage termination at time t given the non occurrence of that event until time t:

$$h(t,x) = \lim_{\Delta t \to 0} \frac{P(t \leq T < t + \Delta t | T \geq t)}{\Delta t} = h_0(t) e^{-X'_{it}\beta}. \tag{2.3}$$

Writing the hazard rate as in equation (2.3) allows us to separate the survival function and the covariates to analyze the relationship between them. $h_0(t)$ captures the distribution function of the failure time when the explanatory variables are equal to 0 (the base rate). It is a reflection of the prepayment rate and varies with the mortgage age (seasoning effect). The second part includes the effects of the explanatory variables on the hazard rate, and it incorporates loan and time specific effects.
The survival function is defined as the probability to survive up to time t:

$$S(t,x) = e^{-H(t,x)},$$

where $H(t,x) = \sum_t h(t,x)$ is the cumulative hazard function.
Combining the results, we obtain the cause-specific density for a mortgage which is terminated at time t:

$$f(t,x) = h(t,x)S(t,x).$$

Therefore, the log likelihood of the competing risk model is similar to the one of the multinomial logit (MNL) model and is given by:

$$\ln L(\beta) = \sum_i \sum_t d_{it} \ln f(t,x).$$

In this competing risk model, the analysis can be divided in two parts: the MNL part determines the cause of death and the hazard model determines the overall risk. When h_0 is small relative to the effect of the covariates then the competing risk model and the MNL are comparable.
The competing risk models have difficulties in including time varying covariates in the analysis [6]. The major drawback of MNL, though, is the implicit assumption of independence between consecutive observations. In fact, they are not independent since are observations related to the same mortgage.

2.4.2. LOGISTIC REGRESSION

Logistic regression in one of the most common techniques to solve classification problems. In our specific case, we are going to see whether a mortgage will prepay ($Y_i = 1$) or not ($Y_i = 0$). Since Y_i can only be either 0 or 1, it is distributed as a Bernoulli random variable:

$$Y_i | X_i = x_i \sim \text{Bernoulli}(p_i),$$

where p_i is modeled as:

$$\text{logit}(p_i) = \beta_0 + \sum_{j=1}^{k} \beta_j x_{ij} \Leftrightarrow p_i \equiv P(Y_i = 1|X = x) = \frac{e^{\beta_0 + \sum_{j=1}^{k} \beta_j x_{ij}}}{1 + e^{\beta_0 + \sum_{j=1}^{k} \beta_j x_{ij}}},$$

and where $\text{logit}(p) = \log(\frac{p}{1-p})$ is called the logistic function. The name logistic regression comes from the fact that this model uses this function.
The likelihood function $L(\beta)$ for this random variable is the one for a Bernoulli random variable, and is given by:

$$L(\beta) = \prod_{i=1}^{m} p_i(\beta)^{Y_i} (1 - p_i(\beta))^{1-Y_i}.$$

From this, keeping in mind that $p_i = \frac{1}{1+e^{-\beta_0 - \sum_{j=1}^{k} \beta_j x_{ij}}} = g(\beta_0 + \sum_{j=1}^{k} \beta_j x_{ij})$, we can define the log-likelihood as:

$$l(\beta) = \sum_{i=1}^{m} \left(Y_i \ln(g(\boldsymbol{\beta} \cdot \mathbf{x}_i)) + (1 - Y_i) \ln(1 - g(\boldsymbol{\beta} \cdot \mathbf{x}_i)) \right).$$

The values for $\hat{\beta}$, the weights, are obtained by minimizing the cost function, which is derived by the log-likelihood and is given by:

$$J(\beta) = -\frac{1}{m} \sum_{i=1}^{m} \left(Y_i \ln(g(\boldsymbol{\beta} \cdot \mathbf{x}_i)) + (1 - Y_i) \ln(1 - g(\boldsymbol{\beta} \cdot \mathbf{x}_i)) \right). \tag{2.4}$$

Contrary to survival analysis, logistic models are able to include time varying covariates. The assumptions for logistic regression are:

- the $Y_i s$ are independently (conditioned) distributed as a Bernoulli,
- the Y_i are binary,
- a linear relation between the logit of the response variable and the explanatory variables,
- homoscedasticity is not needed,
- the error must be independent but need not be normally distributed.

2.4.3. THE DUTCH SITUATION

An early empirical approach to estimate the CPR in the Dutch mortgage market is made by Van Bussel (1998). It was a pool level method based on a very small data set that reached unsatisfying results. In more recent times, models were developed following the same approach used all over the world. Alink (2002) managed to obtain a meaningful data set from the Dutch bank SNS and analysed it using a logit model. Charlier and Van Bussel (2003) developed a model based on survival analysis, more precisely on the proportional hazard model. A different kind of approach is tried by Hayre (2003), precisely it is a descriptive study which uses partial correlations between home sales, defaults, curtailments and refinancing.

3

DATA SET AND FRAMEWORK

In the previous chapter, we introduced all the general and more qualitative concepts, like what it means to prepay a mortgage, what is the prepayment rate, and explained famous driving factors for prepayments in the literature.
We develop a model based on machine learning techniques which requires a data set to be trained on. In this chapter, we introduce the data set we are working on, giving some insights on the actual data to see if we find a match between what expected and the real observations. Moreover, we tackle the problem of class imbalance for our specific situation. A data set in a classification context is imbalanced if the amount of observations belonging to one class is way larger than the amount belonging to the other class.

3.1. DATA SET

The data set contains monthly observations for the time period that goes from February 2011 to February 2016 for each mortgage loan part. Observations regarding June 2011, August 2012, October 2013, December 2014, and February 2016 are not available in the set because they were removed during the data set construction to have some out of sample observations. The data set consists of more than 90 million observations and more than 20 independent explanatory variables regarding a multitude of loan contracts and rates, and client characteristics. The data set has been cleaned from observations with missing or impossible values, e.g. a negative value for the variable related to the age of the mortgage, and from not significant outliers.
Due to computational limits, all the results and images shown in the following sections of this chapter are obtained from a subset of the entire data set. More precisely, it is a subset of ten million observations which we assume to be a feasible representation of the whole portfolio of mortgages.

3.1.1. DEPENDENT VARIABLES

In this section, we will introduce the dependent variables, which correspond to the prepaid amount in Euro by each mortgage in each month. In the data set, prepayments are grouped in four categories based on the type and on the cause of prepayment.
The four categories are the following:

- Early Redemption Movement: a redemption before the reset date due to movement of the mortgagor to another house.
- Early Redemption Voluntary: a voluntary redemption before the reset date.
- Early Repricing Movement: a repricing before the reset date due to movement of the mortgagor to another house.

- Early Repricing Voluntary: a voluntary repricing before the reset.

Voluntary prepayments categories gather all the prepayments for every reason but movement. In particular, a prepayment associated with an address change in the same month, or two months prior or after that month has been categorized in the movement class.
From these quantities, it is first possible to obtain the single month mortality rate as:

$$\text{SMM} = \frac{\sum_{i=1}^{N_{clients}} \sum_{t=1}^{T_i} \sum_{j=1}^{4} (\text{Prepaid Amount in Euros}_{i,t,j})}{\sum_{i=1}^{N_{clients}} \sum_{t=1}^{T_i} (\text{Starting Balance}_{i,t})}, \tag{3.1}$$

where the index i refers to the particular client, t indicates a specific month, and j represents one of the four causes. Therefore, the Prepaid Amount in Euros$_{i,t,j}$ is the amount prepaid by client i at time t for cause j. Starting Balance$_{i,t}$ is the notional outstanding for client i at time t. Basically it is like considering the portfolio as a huge mortgage with a different starting balance every month. This corresponds to determining an average of the prepayment rate weighted by the associated notional. Then, from this quantity we can derive the CPR, which is an annualized quantity, using equation (2.1). It is possible to see the composition of the total CPR due to all the causes. It is worth to notice that voluntary prepayments are definitely more frequent than the ones for movement and that almost all the prepayments are redemptions. This is the reason why instead of building four different models, one for each group, we build only two different models, one for voluntary and one for movement prepayments, where there is no distinction between repricing and redemption.

Moreover, by analysing the dependent variables it was possible to see that around 9 out of 10 prepayments due to movement are full prepayments. While, for voluntary prepayments the opposite is true. Precisely, there are many small prepayments but more frequently.

3.1.2. INDEPENDENT VARIABLES

There are more than twenty independent variables, which are related to clients' specifics, mortgage characteristics, or macroeconomics factors. In this section, we will proceed presenting some of the variables. In particular, the variables that are found to have the most predictive power in literature as explained in part in chapter 2.

We would like to start taking into account the relationship that exists between prepayments and market interest rate, since it should be the main driving factor theoretically speaking. In particular, we are going to analyse the interest rate incentive which is defined as the difference between the rate on the mortgage and the current market rate. Therefore, if this quantity is positive, it means that the interest rate went down and it is reasonable to prepay the mortgage. On the contrary, if this quantity is negative, it means that the rate went up and therefore it is probably better to invest money rather than using them to prepay.

As we would expect from a rational point of view, in Figure 3.1 the number of prepayments, both for voluntary and movement, increases as this interest rate difference becomes more positive. The effect is more visible for voluntary prepayments, this makes sense considering that people do not usually move just because of a more favorable interest rate condition.
Moreover, as touched upon earlier, people do not always prepay following a logical reasoning. This was the main motivation to abandon option theoretical models in favor of statistical ones. The data supports this assumption. In fact, in Figure 3.1 it is possible to see a considerable amount of prepayments for both causes associated to negative values of the interest rate incentive.

The next variable that we are taking into account is time, more precisely how prepayments are distributed over time. In Figure 3.2, a clear seasonal pattern emerges. In particular, as touched upon earlier in chapter 2, the months of December and January have a considerable higher number of prepayments. Going a bit deeper in the analysis, we discover that the main contribution to this seasonal pattern is given by voluntary prepayments. There is no clear seasonality in movement. In order to

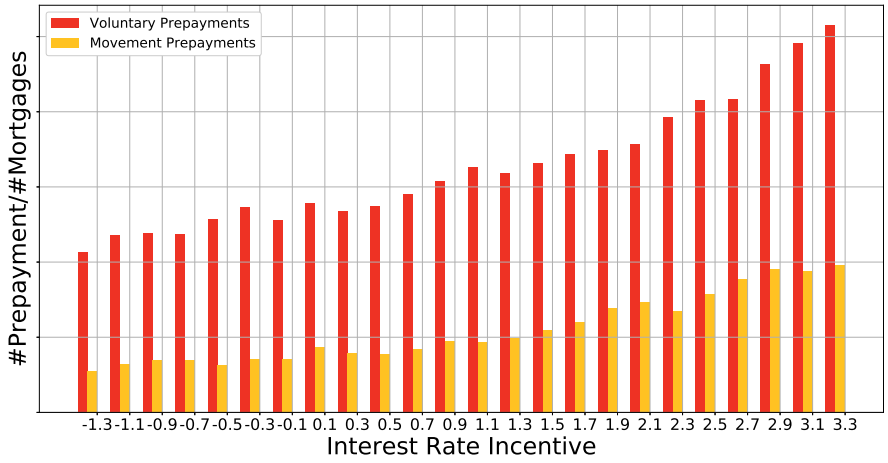


Figure 3.1: Prepayments behavior over time.

take into account this seasonality, indicator variables for the months of December and January are used in the models. Precisely, these two variables assume the value one if the observation took place in one of those two months, zero otherwise. The yellow bar which should represent December 2014 is not present just because we do not have information about that particular month in the data set, not because there is not a higher number of prepayments.

Moreover, a not steep but constant upward trend is visible. As time passes, the average number of

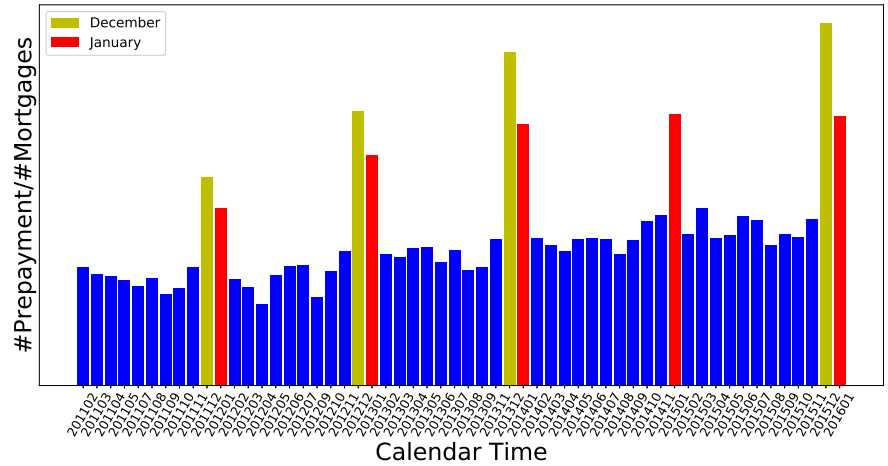


Figure 3.2: Prepayments behavior over time.

prepayments increases. Doing a more thorough examination, we can find two main contributions to the trend. One is related to voluntary prepayments and could be explained by the *media effect* described in section 2.3. In fact, since after the crisis in 2008 we are experiencing a situation of constant decrease of the interest rates, a situation of multiyear lows, and this triggers more and more prepayments. The second one is related to movement prepayments. From data emerged a decisive upward trend in prepayment due to movement, way steeper than the one for voluntary. However, even if a steep increase in movement prepayment could lead to the idea that the low frequency upward trend in the total CPR is just because of movements, it is not. In fact, since this class of prepayments ac-

counts for just a small percentage of the total prepayments (≈ 30%), the impact on the total CPR is comparable for the two class.

The third variable we are going to analyse is the client age. For this plot, we are dividing voluntary and movement prepayments. The reason we do that is because they behave quite differently. For voluntary, we have a slowly increasing trend until a peak at around age 55, then it starts going down to reach a plateau. This is reasonable because it makes sense that people in their fifties have enough money to prepay bigger parts of the notional. Regarding movement prepayments, the rela-

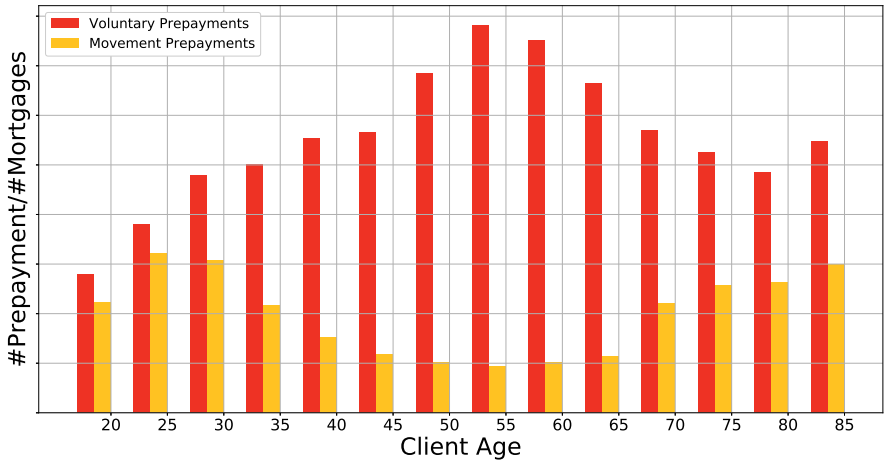


Figure 3.3: Prepayments behavior over time.

tion between input and output is different. We have a peak around the age of 30, then it goes down and starts to grow again around the age of 70. This behavior for movement prepayments also makes sense. In fact, people in their thirties start to build a family and they probably need a bigger house. Moreover, it is normal that with age people die or are not able to take care of themselves anymore so they will move to some helping facilities.

To help the model improve its performance, we defined a new variable starting from the age of the client. The main idea is to cluster in the same group observations which are similar. More precisely, the goal is to treat observations with different client age but similar amount of prepayments as the same, as it appears to be for some of the values from Figure 3.3. The new variable that we call *AgePattern* is defined as follow:

$$\text{AgePattern} = |30 - |30 - \text{Client Age}||.$$

20	25	30	25	20	15	10	5	0	5	10	15	20	25

Figure 3.4: AgePattern variable

In Figure 3.4, the values assumed by 'AgePattern' for values of the client age that go from 20 to 85 with pace 5 are shown. It is clear that the values that it assumes are in agreement with the pattern shown in Figure 3.3. This variable will show to be meaningful in improving forecasting results of our models.

The next one we are going to consider is a macroeconomic variable related to the housing market. In particular, it is the variable that takes into account the house sales in the Netherlands. The prepayment behavior with respect to this variable can be seen in Figure 3.5. Also in this case, the behavior we expected is manifested in the data. On one hand, there is no relation between the number of houses sold and voluntary prepayments. On the other hand, the average number of prepayments for movement is positively correlated with home sales, as one increases so does the other.

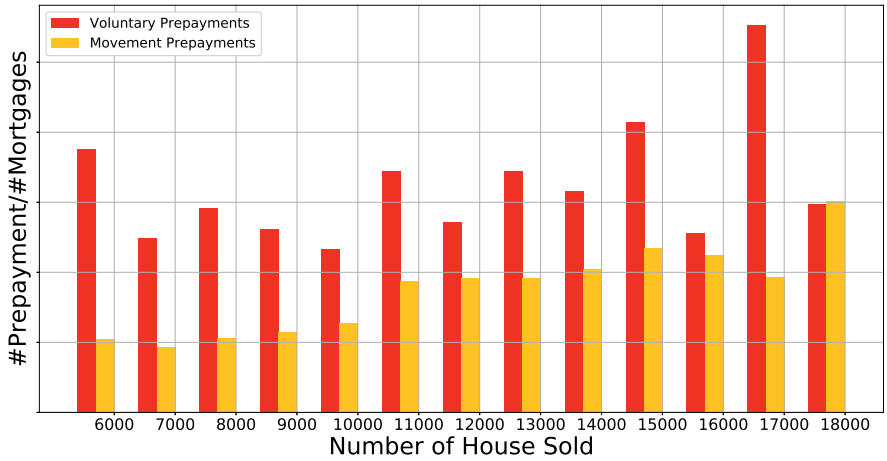


Figure 3.5: Prepayments behavior over time.

Last, we analyze the distribution of prepayments related to the original notional of the mortgage. From Figure 3.6, it emerges that owners of mortgages with bigger original balance voluntarily prepay more than the ones with a small loan. Regarding prepayments due to movement, they are almost

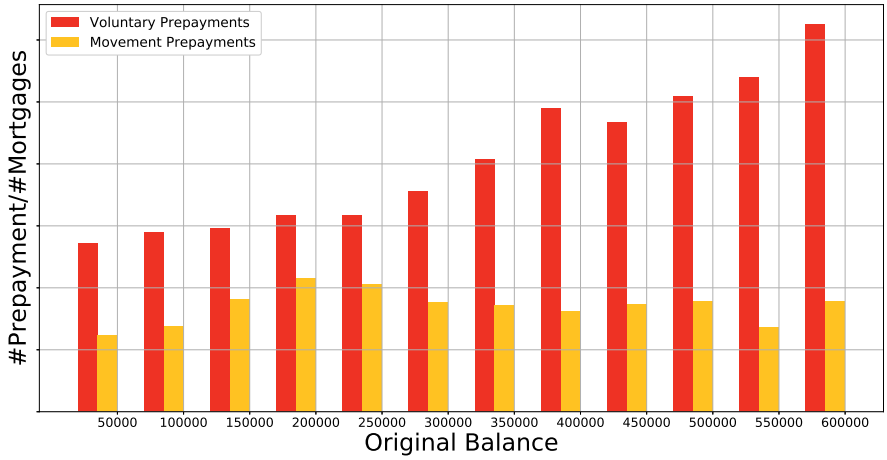


Figure 3.6: Prepayments behavior over time.

constant over the different values of the original notional. There is a small peak around a notional of Euro 200.000.

ADDITION OF A MACROECONOMIC VARIABLE

In order to follow the low frequency upward trend which is present in the CPR evolution, we include a variable which is positively correlated with it. From the *Centraal Bureau voor de Statistiek* (CBS) website, we were able to look at the development of some macroeconomic indexes that have been found meaningful in prepayment prediction literature [8].
The trend we described is present particularly in movement prepayment. Therefore, the first variable we take into account is related to the number of movements that occurs within the Netherlands. Indeed, a positive correlation between these two quantities exists. When the CPR starts to increase,

so does the amount of movements. However, our goal is to forecast future values of the CPR, and we are not able to forecast this variable with certainty in order to feed it to the model to make predictions. Hence, we look for a variable with a lagged relationship with the CPR. The unemployment rate in the Netherlands reveals to be the variable we are looking for. In fact, from Figure 3.7 it is clear that a positive lagged correlation exists. In detail, there appears to be a three years lag and the choice is not

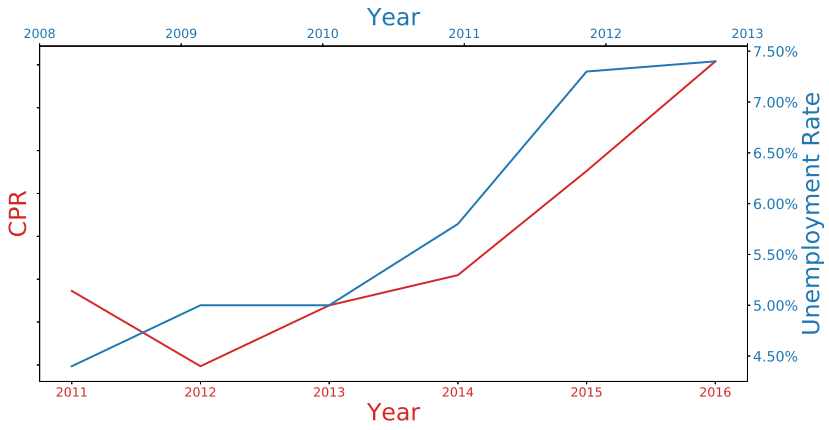


Figure 3.7: Unemployment rate and CPR evolution comparison.

randomly taken. In fact, the forecasting horizon we are interested in is indeed three years. In this way we are able to exactly know the value of the unemployment rate in the time horizon we want to forecast.

PREPAYMENT INDICATOR VARIABLE

People who prepaid once are more likely to prepay again. For this reason, we would like to introduce a categorical variable which tells us whether a determinate mortgage already prepaid in the past or not. However, since we are interested in forecasting the following three years, we need to introduce a third class and a three years lag in the indicator. With these adjustments, we know the values for this variable up to three years in the future for each mortgage.
In particular, the three categories are built as follows:

1. Class = 1: the mortgage age is greater than 3 years and no prepayment occurred in the time frame between origination and three years before the observation is taken.

2. Class = 2: the mortgage age is smaller or equal to 3 years

3. Class = 3: the mortgage age is greater than 3 and a prepayment already occurred in the time frame between origination and three years before the observation is taken.

Even though the amount of information gained from this variable is not as big as it would be in the case of just two classes, a mortgage already prepaid or it did not, its contribution to the final value of the CPR is meaningful and therefore it is worth to introduce it.

3.1.3. SETS CONSTRUCTION

When training a statistical model, it is common practice to divide the whole data set in subsets to see the generalization power of the algorithm, in particular to assess overfitting. We divide the data set in three subsets:

- Training set: is the set used to train the models.
- Validation set: is the set used to select the best among all the trained models.
- Test set: is the set used to assess the generalization power of the model chosen in the validation part.

These three sets are obviously disjoint in the sense that no observation can be in two different sets at the same time. In particular, the training set accounts for 60% of the observations, and validation and test sets account for 20% each. The latter two are created by randomly taking observations from the data set, and what is left becomes the training set.

NEW OBSERVATIONS

New observations for the period that goes from February 2016 until April 2017 become available. For this reason, we create a new set that we call forecasting set which contains just the observations for that time period. The scope of having such a data set is to assess the forecasting power of the model testing it on observations that are disjoint in the time dimension.

3.2. IMBALANCED DATA SET

The prepayment observations in the data set account for just approximately 1.5% of the total number of observations. Therefore, we can say that prepayments are rare events. If we approach the CPR estimation as a classification problem, where one class is the Prepayment class and the other is the noPrepayment class, then the size of the two classes is significantly different. This imbalance usually leads to an underestimation of the probability of belonging to the rare class and an overestimation of the probability of belonging to the majority class.
In classification problems literature, e.g. [15], [16], we talk of an *imbalanced data set* when the amount of observations of one class is significantly smaller than the amount of the other class, e.g. under 5% of the total number of observations belongs to one class. High imbalance is really common in real world problems where the model goal is to find rare but important cases. This kind of framework is identified as anomaly detection problem, and the prediction of prepayment events in a pool of mortgages is one of these cases, where most of the mortgages will not prepay and just a small part will do.
If we have this kind of data set, a usual machine learning algorithm could be inaccurate and get the wrong results. This is because of what the algorithm usually does, and that is to improve the overall classification accuracy reducing the error. However, assessing the performance of the model this way does not work when we are handling imbalanced data sets. In fact, standard classifiers are biased in the direction of the majority class, and it can happen that the minority class is considered as noise. In support to this, in Figure 3.8 a possible example of class conditional distributions for an imbalanced data set is presented. It shows that observations from the rare class are gathered in local regions among all the other samples and are therefore hard to get [17]. For instance, if just 1% of the data set belongs to one class and the model classifies all the observations as belonging to the majority class eliminating the minority class as noise, an accuracy of 99% is achieved. However, the model is completely useless because it can not see any rare event, which is what the user is actually interested in. For these reasons, instead of focusing on maximizing the accuracy, we should focus on optimizing the area under the curve (AUC) of the Receiver Operator Characteristic (ROC) curve. From now on, we will refer to this measure simply as AUC. AUC is often used as evaluation metric in imbalanced class problems, therefore it makes sense to directly optimize it in the training process. Moreover, the explosion of the amount of data in recent years resulted in many large scale *rare class* (RC) problems. There are two different approaches in handling this problem. One is to act at the data level, which means to apply re-sampling techniques to the data set before feeding it to the model. The second

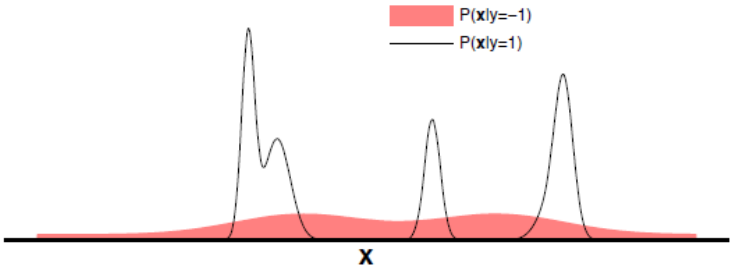


Figure 3.8: Example of class conditional distributions.

one is to act at an algorithmic level, which means to improve the classification algorithm. It is very common to combine this two approaches in order to get the best results [18].

A HEURISTIC EXPLANATION

In this section, we give an explanation to why imbalanced data sets lead to under and overestimation of the class probabilities. Let's assume a setting with two imbalanced classes and just one explanatory variable. In Figure 3.9 it is possible to see the classes distributions. On the left side there is the

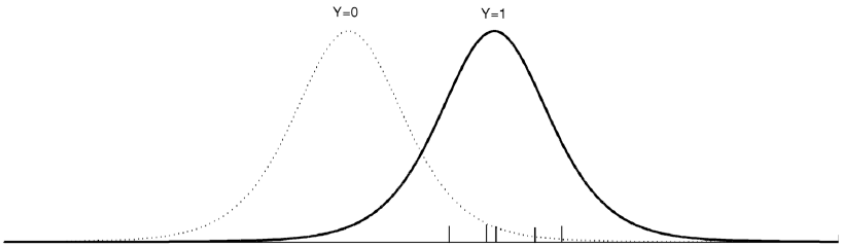


Figure 3.9: Class conditional distributions.

conditional distribution for the majority class $X|Y = 0$ with the distribution as a dotted line. On the right side there are the few positive cases indicated as vertical lines and the corresponding distribution $X|Y = 1$. Because of the high amount of observations belonging to the majority class, we are able to precisely identify the dotted distribution, while our approximation for the density of the minority class is not precise and is usually biased towards too small tails [11]. In fact, what happens is that, given the higher number of zeros with respect to ones, the right tail of the distribution $X|Y = 0$ is well captured while the left tail of the $X|Y = 1$ distribution is poorly approximated. This means that if we want to find a threshold on X to optimize the division between zeros and ones, this quantity is chosen in relation to the maximum likelihood estimator and would probably be positioned on the left of the last or second last positive observation starting counting from the right.

3.2.1. DATA LEVEL METHODS

In this section, we will present some methods that focus on re-sampling the initial data set. The goal of doing this is to either increase the frequency of the rare event class or decrease the frequency of the majority class.

UNDER-SAMPLING

Random under-sampling is a method which achieves a balanced data set by randomly removing some samples that belong to the majority class. This way, we solve the problem of having an imbalanced data set and reduce the run time and the hardware demand, but there are two major flaws in this

technique. The first one is that we can potentially discard useful information which could be important for the classification process. The second is that the selected random sample could be biased, therefore not representative of the entire population.

OVER-SAMPLING

Random over-sampling is a method which achieves a balanced data set by randomly adding one or more duplicate observations from the minority class. On one hand, this technique does not lead to any information loss, and usually outperforms under-sampling. On the other hand, it could lead to an increase of computational and storage costs, and it increases the likelihood of overfitting. In fact, although the rules used by the classifier could seem accurate, they actually just deal with the same repeated samples.
There is a smarter way to do over-sampling, that is to generate minority samples using some interpolation techniques between minority samples that are close together. Using this more advanced method of over-sampling, we are able to avoid the overfitting problem, and to extend the boundaries of this class making the data set even more balanced.

3.2.2. ALGORITHMIC LEVEL METHODS

In this subsection, we fix this same problem but by modifying the classification algorithm in order to make it able to handle imbalanced data sets. The main goal with these methods is to enhance the performance of single classifiers.

THRESHOLD METHOD

In binary classification problems, the output of the model is usually a number between 0 and 1 and it could be interpreted as a probability. The probability of belonging to one class, $y = 0$, or the other, $y = 1$. Therefore, the model could be more or less sure about how to classify different samples. Then, observations are associated to one class or the other using a threshold. Usually this threshold is chosen to be 0.5, any observation with output smaller than the threshold belongs to the $y = 0$ class and any greater one to the other class. Moving this threshold affects the classification process. For instance, selecting a bigger threshold means that we want a greater degree of certainty in order to classify an observation in the $y = 1$ class.

ONE-CLASS LEARNING

In a normal classification problem, the goal of the algorithm is to identify and classify observations in different classes learning from a data set containing all the elements from the different classes. The goal of a one-class classification algorithm is to identify a class of observations among all the others, but learning from a data set containing just the observations belonging to that class.
RIPPER (Repeated Incremental Pruning to Produce Error Reduction) is an algorithm which builds its rules by adding more and more conditions until no negative observations are covered. It will come up with rules for every class, starting from the rarest class, which is why it is easy and possible to learn rules just for the minority class. This method is particularly effective when the data set is highly imbalanced and there is a high dimensional noisy feature space. This method can be related to an aggressive features selection model, and it is often preferred since it is less expensive to apply.

COST SENSITIVE LEARNING

Another approach to have different costs included in the decision-making process is to introduce "fixed and unequal misclassification costs between classes" [18]. We consider a cost matrix in which the element λ_{ij} is the cost of classifying an element from class j as belonging to class i. The diagonal of this matrix is usually formed by zeros since there is no cost for correct classification. In the light of this, we define a conditional risk given by:

$$R(\alpha_i|x) = \sum_i \lambda_{ij} P(v_j|x).$$

The meaning of the formula is that the misclassification cost depends on a fixed part, λ_{ij}, and on an uncertain part, the posterior probability related to our uncertain knowledge of the true class of x. Hence, in this method to minimize our cost we choose the class v_j which minimizes the conditional risk.

BOOSTING ALGORITHMS

A boosting algorithm is an iterative method which gives different weights to the training distribution every iteration. In each iteration, the weights related to incorrectly classified observations increase, and the ones associated with the correct classified ones decrease. This way, the learning algorithm is forced to focus more on misclassified examples in the following iteration. This boosting should improve the classification performance of the rare class since it is more prone to error than the common class, and therefore the weights associated to the minority class increase. Another possibility could be to directly penalize more heavily misclassification errors for the minority observations in order to reduce the bias towards the common class [19]. AdaBoost and Rare-Boost are two of these techniques [18].

3.2.3. ASSESSING THE POWER OF THE ALGORITHM

The prediction obtained with the algorithm could either be right or wrong. If it is correct, then it means that whether we predict success, $y = 1$, when the observation is a success and this is called a true positive, or we predict a failure, $y = 0$, when it is a failure and this is called a true negative. Also if we are wrong we can have two different kinds of errors, a false positive is when we predict $y = 1$ but it is $y = 0$, and a false negative is the other way around, as can be easily observed from Table 3.1.

		Predicted Class	
		0	1
Actual Class	0	True Negatives	False Positives
	1	False Negatives	True Positives

Table 3.1: Confusion matrix

Once defined these 4 concepts, we can define 3 tools to assess the output of the algorithm:

- Accuracy= $\frac{\text{True Positives+True Negatives}}{\text{Total Observations}}$
- Precision= $\frac{\text{True Positives}}{\text{Predicted Positives}} = \frac{\text{True Positives}}{\text{True Positives+False Positives}}$
- Recall= $\frac{\text{True Positives}}{\text{Actual Positives}} = \frac{\text{True Positives}}{\text{True Positives+False Negatives}}$

On one hand, Recall is computed as the number of true positives divided by the number of actual positive observations, which represents the percentage of observations that belongs to the positive class that is actually classified as such. On the other hand, Precision is obtained by dividing the number of true positives by the total number of positive outcomes, which represents the percentage of observations which is classified in the positive class that actually belong to that class. It is important to have a good balance between these two values. The reason to introduce these two quantities is immediately obvious. For instance, if we consider again a case where one class accounts for just 1% of the total number of observations, the naive algorithm which classifies every observation as belonging to the majority class (the negative one) achieves an accuracy of 99%, but the values of Precision and Recall are zero.
An accurate way to find a compromise between Precision and Recall is to define the score $F_1 = 2\frac{PR}{P+R}$ (P=Precision and R=Recall), which is the harmonic mean of these two quantities. Then, the algorithm with higher F_1 score is chosen.

As mentioned at the beginning of section 3.2, AUC is another measure that is suitable in our context. We will now present how this quantity is empirically derived and explain its meaning.
The ROC curve shows the classification ability of a binary classifier as the threshold that divides the two classes is varied. It is obtained by plotting the true positive rate (Recall) against the false positive rate, which is equal to (1 – Specificity) and Specificity represents the proportion of actual negative samples that are classified as such.
The AUC can be empirically computed with the following formula:

$$AUC = \frac{1}{m_+ m_-} \sum_{i:y_i=1} \sum_{j:y_j=0} \mathbb{1}(\hat{y}(x_i) > \hat{y}(x_j)),$$

where $\hat{y}(\cdot)$ represents the output of the model given the input $\cdot$, m_+ and m_- represent the number of observations belonging to the majority and minority class respectively, and $\mathbb{1}$ is the indicator function.
Hence, the AUC of a classifier can be interpreted as the probability that the classifier will rank a randomly chosen positive example higher than a randomly chosen negative example:

$$AUC = P(\hat{y}(x^+) > \hat{y}(x^-)).$$

Therefore, the higher this value, the better. Ideally, we would like to have its value to be equal to one. If that is the case, it would mean that we are able to find a certain threshold with which we are able to correctly classify all the observations in the set.

3.2.4. EFFICIENCY OF THE ADJUSTMENTS

In this section, we would like to briefly show how models are adjusted in order to obtain better performance and how in general machine learning algorithms achieve a superior level of Accuracy, Precision and Recall. To do that, we applied a logistic regression (LR), an artificial neural network (ANN) and a random forest (RF) to the fraud detection data set downloaded from *Kaggle.com*. Only the results are shown in this section to underline the efficiency of the corrections. A more thorough and detailed explanation of the applied corrections will be given in Chapter 4.
The data set has 284.807 observations, which represent the credit card transactions of European card holders for two days in the month of September 2013. We choose this data set because it is highly imbalanced, in fact the positive class, which represents the fraud transactions, accounts for just 0.172% of the total data set (492 frauds).

To train the model, we took a subset of 49000 observations from the initial data set in order to have a ratio between the two classes of 1:100. Moreover, weights are added to the loss function to achieve even more balance. Then, we split the subset in a training, validation and test set. Each of the last two sets accounts for 20% of the observations, and the first one of the remaining 60%. After training the models and choosing the best one on the validation set, we test this model on the test set but also on the whole data set. The results for the model without corrections were poor, the accuracy stayed quite high but the values for Precision and Recall were close to zero. Meaning that the imbalanced data set indeed biased the model outcomes toward the majority class. In Table 3.2, the performance of the best model for each technique on the test set and on the whole data set is shown.

		Precision	Recall	F_1 score	AUC	Accuracy
LR	Test	0.9213	0.7663	0.8367	0.8828	0.9967
	Data set	0.5167	0.7541	0.6231	0.8764	0.9983
ANN	Test	0.9100	0.8505	0.8792	0.9248	0.9974
	Data set	0.7023	0.9349	0.8021	0.9671	0.9989
RF	Test	0.9574	0.8411	0.8955	0.9203	0.9980
	Data set	0.8697	0.9634	0.9142	0.9816	0.9992

Table 3.2: Results

We have to keep in mind that we are dealing with an imbalanced data set and, therefore, looking just at the accuracy could be misleading. In fact, even if the values for it are almost equal across the three models, the other indicators tell us that machine learning techniques are better performing on this imbalanced data set than logistic regression. For instance, the higher value for Precision means that the algorithm better learns the fraud transaction events. In fact, in logistic regression for the whole data set we obtain a Precision of $\approx 51\%$, which means that almost every other observation classified as fraud, it is not actually a fraud. This ratio reduces to 1 every 3 and 1 every 5 observations for the machine learning approaches. Moreover, another confirmation comes from higher values of Recall which are translated by the fact that the second kind of approach is better able to identify frauds. In logistic regression, one fourth of the frauds is missed by the algorithm against a 5% for the other two methods.

3.3. HOW TO TAKE INTO ACCOUNT THE CORRECTIONS

In the previous section, we introduced the issue of having an imbalanced data set and gave some insights about possible corrections to apply to the model in order to cope with this problem. In particular, we introduced two main classes of adjustments: one that acts on a data level, and the other one that acts on an algorithmic level. Both kinds of modifications affect the characteristics and the output of the model. Therefore, we investigate how the algorithm is affected in order to be able to take into account these changes.
Every alteration has its specific effects. In this section, we are going to take into account how corrections on a data level influence the models. We will take into account algorithm modifications in Chapter 4 since to understand these second kind of changes a deeper understanding on how the models work is needed.

Among data level methods, the one that turns out to be the most effective is, without any doubt, undersampling. As touched upon earlier, this technique consists in removing some observations from the majority class in order to achieve a balanced data set.
We explained in section 3.1.3 how the data set has been divided in four subsets and that the main assumption in this procedure is that the observations in all the sets are drawn from the same population and have indeed the same distribution. When we undersample the majority class in the training set, a procedure that is widely used in the context of imbalanced data sets [20], we violate this very assumption. The training set is now balanced, while the validation and test sets are still imbalanced and have therefore different distributions. Having different distributions in the various sets is a known problem in statistical analysis and has been tackled many times in related literature, [21], [22].
In fact, undersampling implies a change in the prior distribution of the training set and this biases the posterior probability of the classifier. This kind of problem is called *prior probability shift* and happens when the distribution over $\mathbf{y}$ changes while everything else remains the same [23].
Balancing the data set is a sample selection bias problem with a known selection bias, since we introduced it, and we want to investigate the entity of this bias that occurs when undersampling the majority class.

A relationship between the output of the classification, which represents the posterior probabilities, of a model trained on the balanced set and of one trained on the unmodified set exists [24]. We will now proceed to show this relation. Let's define a random binary selection variable s that takes the value 1 if the sample is in the undersampled set and 0 otherwise. Then, assume that undersampling the majority class does not change the within class distributions. From the Bayes' rule we have that the output probabilities of the model are given by the following formula [24]:

$$p(y=1|x,s=1) = \frac{p(s=1|y=1)p(y=1|x)}{p(s=1|y=1)p(y=1|x)+p(s=1|y=0)p(y=0|x)},$$

and since $p(s=1|y=1)=1$ because all the samples belonging to the positive class are in the undersampled set, we have:

$$p(y=1|x,s=1) = \frac{p(y=1|x)}{p(y=1|x)+p(s=1|y=0)p(y=0|x)}.$$

Denoting the probability of choosing a negative sample while undersampling as $\beta = p(s=1|y=0)$, the posterior probability of being in the positive class for the original set as $p = p(y=1|x)$, and the posterior probability of being in the positive class in the undersampled set as $p_s = p(y=1|x,s=1)$, we can rewrite the previous equation as:

$$p_s = \frac{p}{p+\beta(1-p)}. \tag{3.2}$$

In equation (3.2), the relationship between p_s, which represents the output of the model trained on the undersampled set, and p, which represents the output we should expect for a model applied to the non-undersampled set, is shown. It is trivial to obtain the expression for p as a function of p_s:

$$p = \frac{\beta p_s}{\beta p_s - p_s + 1}. \tag{3.3}$$

Now, we are able to preserve marginal probabilities for the classes in the validation and test sets even when applying a model which is trained on a subsampled set. β is the parameter that represents the amount of samples removed while undersampling.
The β parameter in equation (3.3) is quite informative. For instance, $\beta = 1$ is the upper bound and it means that no undersampling is done, in fact, it results in $p = p_s$. The lower bound is $\beta = \frac{N^+}{N^-}$, where N^+ and N^- are the number of positive and negative observations respectively in the original data set. Such a value corresponds to a balanced set, it is like the data set is undersampled until $N^+ = N^-$. Hence, β is an indicator of the degree of balancedness achieved. As β decreases from 1 towards $\frac{N^+}{N^-}$, the training set that we will obtain becomes more and more balanced.
If we call p' the bias corrected posterior probability that we obtain from the empirical quantity p_s using equation (3.3):

$$p' = \frac{\beta p_s}{\beta p_s - p_s + 1},$$

and, as before, we call p the actual posterior probability for the original set,

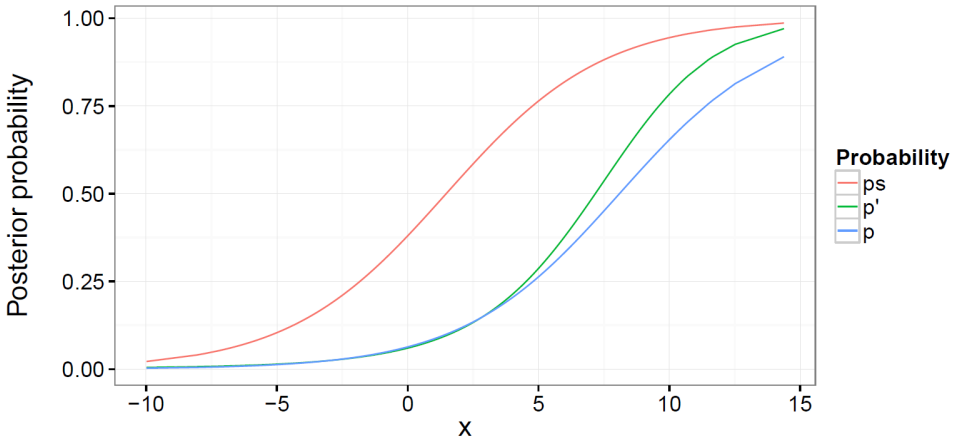
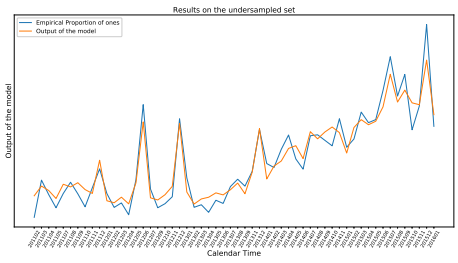


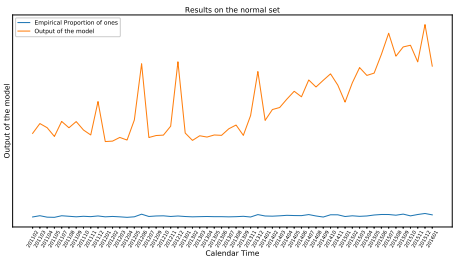
Figure 3.10: Posterior probabilities.

we observe from Figure 3.10 that p' is indeed able to approximate the initial posterior p.

In Figure 3.11 it is possible to see in practice what happens when we train on an undersampled set and then we try to apply the model to a set in which the original ratio between classes is maintained. In the left picture, it is possible to see the model applied to a set in which the ratio between classes is the same as the one in the set we used to train the model. The prediction is quite accurate and on scale.



(a) Result on the undersampled set



(b) Result on the set without undersampling

Figure 3.11: Results of the model for both the not and the undersampled set.

In the right picture, we see the results of the model applied to a set $\mathbb{S}$ which has the initial ratio between classes, the one before undersampling. It means that in this set the amount of observations belonging to the majority class is way bigger than before. Now, rare class observations account just for 1.5% of the total. This is what practically means a change in the prior. Now, since the proportion of ones is way smaller, it is normal that the model overestimates the output.
If we apply the correction introduced earlier in the chapter, the results that we obtain for the same set $\mathbb{S}$ are shown in Figure 3.12.

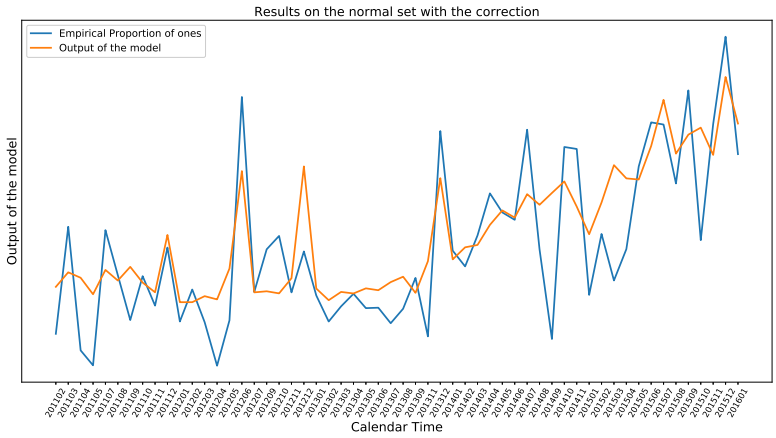


Figure 3.12: Results on the non undersampled set with the correction.

It appears from the plot that the correction is indeed effective. The same model goes from the poor performance shown on the left in Figure 3.11 to the results illustrated in Figure 3.12. It is normal that the overall performance on the non undersampled set is slightly worse than the one on the training set. In fact, in the set $\mathbb{S}$ there are many observations that were never seen before by the algorithm. Therefore, it is understandable that it can approximate the final plot a bit less precisely.

4

STATISTICAL AND COMPUTER SCIENCE MODELS

In the previous chapters, we introduced the prepayment rate, explained that it is important for a bank to estimate this quantity for the hedging strategies, gave some insights on what causes the prepayments, and introduced the data set.
As stated earlier, the purpose of this thesis is to build a model which is able to make a forecast of the CPR for the coming 3 years. The model that will be developed is based on machine learning techniques, and in this chapter we present the theoretical background needed for our goal. In the first part of the chapter, some general information about machine learning are displayed. Then, we give a theoretical explanation of how neural networks and random forests work with a specific section in which the modifications made to the model for this specific problem are explained.

Machine Learning has become more and more popular in the last years, but, what is it? It is, to quote Arthur Samuel's definition, the "Field of study that gives computers the ability to learn without being explicitly programmed". Samuel was a pioneer in artificial intelligence and was the one who coined the term *Machine Learning* in 1959 [25]. The origin of this field of study can, therefore, be traced back to the 1950s, it is not something that has popped up in the recent years. The main reason behind the recent popularity is linked to the increasing computational capacity of nowadays PCs and the increasing availability of digitalized data. In fact, the main goal of a machine learning algorithm is to learn from and make predictions on data with the peculiarity of taking data-driven decisions rather than following program directives. Machine learning can perform the tasks of linear and logistic regression, but can take them a step further. For instance, it is able to take into account a considerably larger amount of data and predictors, and is capable of understanding more complex nonlinear relations between input and output.

4.1. DIFFERENT KINDS OF MACHINE LEARNING

In order to give a better insight in Machine Learning, we would like to give a more formal and precise definition that was given in the late 1990s by Tom Mitchell [25]. He defined a well posed machine learning problem as: "A computer program is said to learn from experience E with respect to some task T and some performance measure P, if its performance on T, as measured by P, improves with experience E". We give an example to help the reader understand the definition.
Let's suppose that we want to develop a program which is capable to classify emails as spam or not spam on the base of what the user did or did not mark as spam. In this setting, the task T obviously is to divide emails in spam and not spam, the experience E is watching the user selecting which emails are or are not spam, and the measure P is the fraction of emails correctly labeled as spam/not spam by

the program. It is worth to notice that a correct choice of the measure P is fundamental. For instance, if the program identifies all the emails as spam, of course it identifies all the spam emails, but it would be useless. We want a good compromise between marking all spam emails as spam, and marking not spam email as not spam. Our measure needs to be more sensible than the simplistic one we stated earlier.

A big data set is given to every machine learning algorithm in order for it to train itself. The way it does this task is not always the same though. Depending on the way it works, it is possible to identify two broad categories:

- **Supervised learning**: In the training data set both the inputs and the correct outputs are present, and the objective is to learn how to properly map future inputs into the correct outputs. An example could be the aforementioned one about the spam emails, where both the input, information about the email, and the associated label, spam/not spam, are given to the algorithm in order to be trained.
- **Unsupervised learning**: In the training data set the correct outputs are not present anymore, the goal now is to find similarities, patterns and structures and then group the data into clusters. This clustering can be the goal itself, or the means to an end. An example could be market segmentation, data about clients are fed to the algorithm and it tries to identify groups of customers to make, for instance, targeted ads.

Within these two big classes, another classification, based on the machine learning tasks and the kinds of problems the algorithm can solve, exists. The first two classes belong to the category of supervised learning problems, while the last two to the unsupervised ones:

- *Classification*: is a problem where the output can assume one of finitely many possible values. The spam email example is one of this kind.
- *Regression*: is a problem where the output is a continuous value. An example could be an algorithm which receives data about houses and their selling prices, and learns how to correctly price houses.
- *Segmentation*: is a problem where the goal is to learn the structure of the data and build clusters of comparable elements. The market segmentation is an example of this kind.
- *Network Analysis*: is a problem where the goal is to identify nodes in a network and define their importance and their role. An example of this could be found in social network analysis.

4.1.1. BASIC TERMINOLOGY

In this part, we will focus more on supervised learning, both regression and classification types.
When we feed data to the algorithm, it will train itself to replicate the desired given outputs, but the goal is actually to train it in order to be able to make predictions on different data from the one used for training. This is a complicated task, and we assume that the unseen data is somehow similar to the one in the training set. In particular, we assume that the distribution of the population for the training set is representative of the full population and indeed of the test set. In this way, if a model performs well on the data we can see, then it should also work for the new data.
We said "it should work" because of a major problem known in literature as overfitting [26]. In every data set there are both observations and noise, our goal is to separate the two in order to fit the data in the best way possible. However, trying to perfectly explain as many inputs as possible by doing overly complex assumptions often leads to consider also the noise in the building of the model, which will result in a poor prediction for data which is not present in the training set. This is the phenomenon of overfitting, which is characterized by a large gap between the error obtained in the training set and

the one in the test set. This happens because a model that overfits the data will perfectly explain the training set (small error) but it will struggle to make predictions on a new data set (test set).
This kind of problem is associated with high variance, which is intended as the error due to small variations in the training set. Basically, it means that random noise is taken into account by the algorithm in the modeling process.
We have the opposite problem when we are dealing with underfitting, that is when the model cannot follow the underlying trend of the data. These kinds of errors are the result of a too simple model, for example if we try to fit a linear model to data which is not linear. This kind of problem is associated with high bias instead, which is the error which derives from erroneous assumptions in the learning algorithm that leads to miss some relevant relations between data points. It is fundamental to find the right trade off between bias and variance.

4.2. ARTIFICIAL NEURAL NETWORKS

Artificial neural networks (ANNs), or in this context simply neural networks, are computational systems which try to roughly mimic the networks of neurons in a biological brain [27]. A big advantage of this method is the fact that it uses the basic not computationally expensive operations (e.g. basic logic operations, multiplications, additions) to solve complex problems that are not linear or mathematically not well posed.

4.2.1. THE NEURONS

A neuron is the essential unit which processes information in a neural network architecture. It is possible to see a scheme of a neuron in Figure 4.1, in which it is possible to identify three main components:

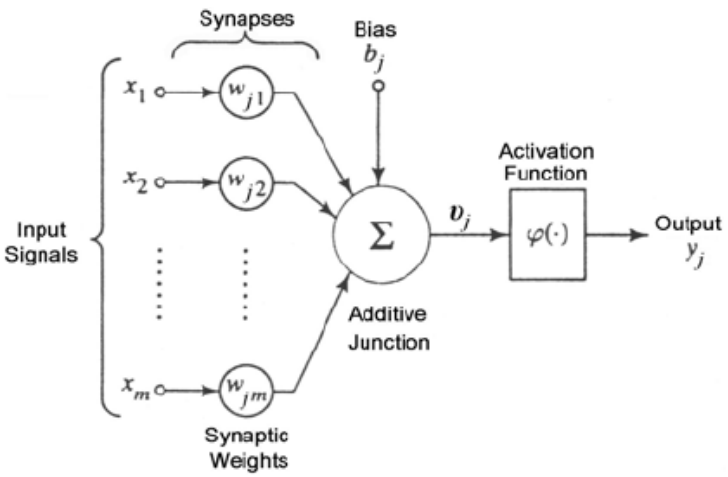


Figure 4.1: Model of a neuron

- Synapses: a set of connecting channels each characterized by its own weights. An input signal, before reaching the neuron, passes through these synapses and it is multiplied by the weights. In particular, the input signal x_i that has to go through neuron j is weighted by the weight w_{ji}.

$$x_1, x_2, \ldots, x_m \Rightarrow w_{j1}x_1, w_{j2}x_2, \ldots, w_{jm}x_m.$$

- Adder: an adder which is able to sum all the input variables after they have been weighted in

the synapses. Usually a linear combiner is used, i.e.,

$$u_j = \sum_{i=1}^{m} w_{ji} x_i.$$

- Activation function: inside the neuron, the quantity $v_j = u_j + b_j$, which is the weighted sum of the inputs plus the bias term b_j, is passed through the activation function which is used in order to limit the range of the output of the neuron.

$$y_j = \varphi(v_j),$$

Alternatively, we can reformulate the previous formulas as follows:

$$v_j = \sum_{i=0}^{m} w_{ji} x_i \Rightarrow y_j = \varphi(v_j),$$

which is equivalent to the previous formulation. More precisely, a synapses has been added and its input is $x_0 = 1$ and the corresponding weight is $w_{j0} = b_j$, which correspond to the bias in the precedent formula.
Synapses weights are found during the training of the algorithm, activation functions are chosen by the user. There are different kinds of activation functions which we will now introduce.

ACTIVATION FUNCTIONS

It is possible to identify three basic categories of activation functions:

1. Threshold functions:

$$\varphi(v_j) = \begin{cases} 1 & \text{if } v_j \geq 0 \\ 0 & \text{if } v_j < 0 \end{cases}.$$

   This kind of neuron has the all or none property since it is either completely activated, when the weighted sum is positive, or not activated at all when that sum is negative.
   One variation of this kind of activation function is given by the largely used Rectifier Linear Unit (ReLU) function, which has the following form:

$$\varphi(v_j) = \begin{cases} v_j & \text{if } v_j \geq 0 \\ 0 & \text{if } v_j < 0 \end{cases} = \max(o, v_j).$$

2. Piecewise linear functions:

$$\varphi(v_j) = \begin{cases} 1 & \text{if } v_j \geq \frac{1}{2} \\ v_j & \text{if } -\frac{1}{2} < v_j < \frac{1}{2} \\ 0 & \text{if } v_j < -\frac{1}{2} \end{cases},$$

   where in the linear region the amplification factor is chosen to be one. This is a sort of approximation of nonlinear amplifier. Threshold functions are special forms of these class activation functions.

3. Sigmoid functions: the graphs of these kind of functions is s-shaped and strictly increasing but with a good balance between linear and non linear part.

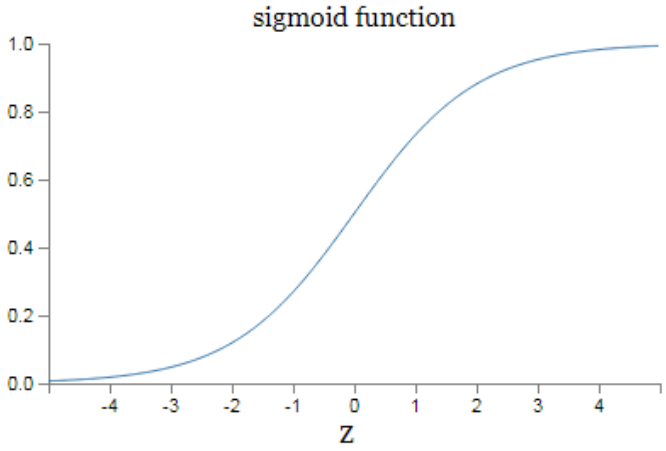


Figure 4.2: Sigmoid function.

One of the most used functions for this class is the logistic function:

$$\varphi(v_j) = \frac{1}{1+e^{-v_j}}.$$

It is common to choose such functions for their smoothness, in fact this implies that small changes in the weights will not affect significantly the output. We want this property since making small changes in the weights and biases is how the algorithm trains to improve the output, this is how the network is learning. Moreover, another desired property is that they are bounded, the logistic function has value in $[0,1]$ for instance. Anyway, the sigmoid function is not the only choice [28].

4.2.2. STRUCTURE OF AN ANN

There are different ways to combine these basic units, the neurons, to form a network. In particular, the networks we implement are multilayer feedforward networks.
In this kind of architecture, the neurons are organized in layers. We can distinguish three kinds of layers, as it is clear from Figure 4.3.

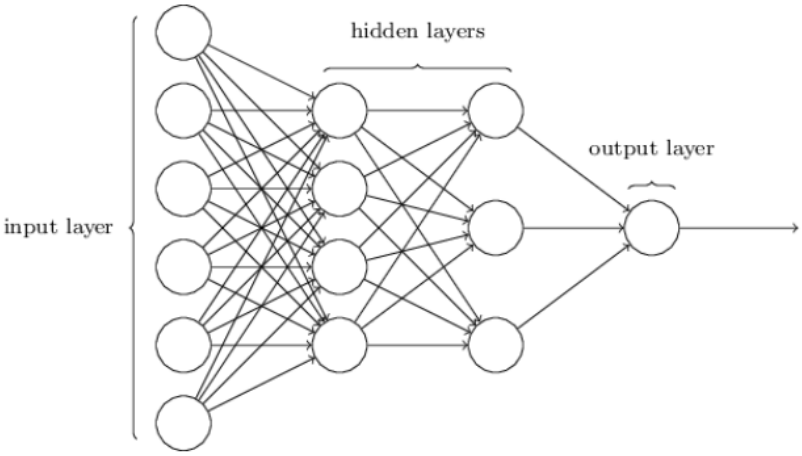


Figure 4.3: Example of a simple neural network.

The first layer, the one that receives the inputs, is called *input layer*, which consist of $N_1 = 6$ input neurons in this picture; the last one is called *output layer*, which in our case consists of $N_4 = 1$ output

neuron. In the middle we have *n hidden layers* with N_l hidden nodes, which are called like that since they are neither an input nor an output (for instance, for the example in the previous page we have $n = 2$, $N_2 = 4$, $N_3 = 3$). Hidden neurons are applied in series to the observations between input and output and this enables the network to extract higher order statistics [29].
In this example of ANN, and in general for feed forward networks, the input of one layer is given by the output from the previous one, except for the input layer for which the inputs are the observations from the data set. In addition, this means that the output of each layer is used as input by the next layer besides for the output layer. The output from the last layer represents the overall result of the network. These kinds of ANNs are called feed forward neural networks since no loops are involved in the network. Moreover, they are often referred to as multilayer perceptrons as well.
The network shown in Figure 4.3 is fully connected. This means that each node in each layer is connected to every other node in any adjacent layer.

4.2.3. LEARNING PROCESS

The learning process includes three main steps. First, the network is fed with the data in order to be triggered. Second, the network changes its parameters according to some decision rules or loss functions as a consequence of this titillation. Third, the network is now different from how it was before having processed the data since the parameters in the architecture have changed.
There are various learning rules, in feedforward neural networks an error correction learning is implemented. This rule is known as *error back propagation algorithm*, and it enables the multilayer perceptron to solve complex problems. This kind of learning rule is only possible in supervised learning. We will now explain how it works.
There are two different and distinct parts in the computation of artificial neural networks with backpropagation. The first one is known as forward pass, the second one as backward pass. In the forward pass, the information travels from left to right and the weights are not modified. In the backward pass, the information travels from right to left and it is at this stage that the weights are modified. Once the model is trained, data passes through the model as in the forward pass.

FORWARD PASS

Assume we have $x_1, \dots, x_m$ samples as inputs and let's call $a_i^1 = x_i, \forall i$, the inputs for the first hidden layer. Then, for each hidden node in each hidden layer we first have:

$$v_j^l = \sum_{i=0}^{N_{l-1}} w_{ji}^l a_i^{l-1}, \ \forall\ j = 1, \dots, N_l, \forall\ l = 2, \dots, L-1.$$

Then, after that the input is weighted and summed, the activation function is applied:

$$a_j^l = \varphi(v_j^l), \ \forall\ j = 1, \dots, N_l, \forall\ l = 2, \dots, L-1.$$

This function is, in general, nonlinear as shown above, hence it allows the algorithm to learn nonlinear relations between the data. Then, this a_j^l output is used as input for the $l+1$ layer and we repeat this procedure for every layer. For the output layer, we have $l = L$ and a_j^L coincide with y_j which is the final output of the model:

$$v_j^L = \sum_{i=0}^{N_{L-1}} w_{ji}^L a_j^{L-1}, \ \forall\ j = 1, \dots, N_L,$$

$$y_j = a_j^L = \varphi(v_j^L) = \varphi\left(\sum_{i=0}^{N_{L-1}} w_{ji}^L a_i^{L-1} \right), \ \forall\ i = 1, \dots, N_L.$$

Now, if we call $y_j(n)$ the output of the output layer of the network for the vector of observations $\mathbf{x}(n)$, where n represents a time step in the iterative training process, a comparison between this output

and the target output $d_j(n)$ can be made, and an error is calculated as:

$$e_j(n) = d_j(n) - y_j(n). \tag{4.1}$$

Then, through this quantity a whole mechanism comes into play which will make adjustments on the weights of each neuron with the aim of achieving an output signal which is the closest possible to the desired target at each iteration. This goal is practically achieved by minimizing a cost function which depends on the error that we defined earlier. After the forward pass, it is now time for the back-propagation part.

4.2.4. BACK-PROPAGATION

The back-propagation starts at the last layer and moves backwards to the first one passing through each neuron and calculating the gradient in each of it. During this procedure the weights are modified according to some rules.
From the quantity defined in equation (4.1), it is possible to introduce an error measure for the output layer which depends on $e_j(n)$ defined by:

$$\mathscr{E}(n) = \frac{1}{2}\sum_{j\in C} e_j^2(n),$$

where C represents the set of all the neurons belonging to the same layer. Then, if we iterate the process N times for training, each one of these iterations is called epoch, we can sum the errors at each iteration and divide that quantity by N and obtain the average error $\mathscr{E}_{av}$. This quantity is what we call cost function and it is a function of the bias level and of the neuron weights. Basically, the algorithm will try to adjust the parameters in order to minimize this quantity $\mathscr{E}_{av}$ at each epoch.
In a back-propagation algorithm, the correction term $\Delta w_{ji}(n)$ of the synaptic weights is proportional to the partial derivative of the error with respect to the weights. This proportionality factor represents the sensitivity factor which indicates the direction of search in the space:

$$\Delta w_{ji} = -\eta \frac{\partial \mathscr{E}(n)}{\partial w_{ji}(n)}, \tag{4.2}$$

where η is the learning rate parameter that regulates the speed of the back-propagation. After some calculations, we can derive the following form for the partial derivative [29]:

$$\frac{\partial \mathscr{E}(n)}{\partial w_{ji}(n)} = -e_j(n)\varphi_j'(v_j(n))y_i(n). \tag{4.3}$$

The minus sign is for the gradient descent in the weight space.
It follows from equations (4.2) and (4.3) that:

$$\Delta w_{ji} = \eta \delta_j(n) y_i(n), \tag{4.4}$$

where $\delta_j(n) = e_j(n)\varphi_j'(v_j(n))$ is the local gradient. The computation of δ_j requires the knowledge of the derivative of the activation function, for this reason we require this function to be differentiable.
Now, we can define two distinct cases, one if neuron j belongs to one of the hidden layers and one if the neuron belongs to the output layer. The latter case is easy since it is possible to directly calculate the error as a difference between output and target. It is more complex for the hidden neurons because it is hard to understand which neuron has to be rewarded or penalized. To solve this, the error is back-propagated through the network.

OUTPUT LAYER

If neuron j belongs to the output layer, then the error $e_j(n)$ is immediately computed. Once the error is available, it is easy to calculate the local gradient $\delta_j(n)$ as:

$$\delta_j(n) = e_j(n)\varphi_j'(v_j(n)).$$

HIDDEN LAYER

If neuron j belongs to a hidden layer, then its desired output is unknown. The error must be calculated taking into account all the errors in the connected neurons, and here is where the back-propagation gets harder. The local gradient has a different form now, because $\frac{\partial \mathscr{E}(n)}{\partial y_j(n)} \neq e_j(n)$. It is possible to calculate that partial derivative and arrive at the form [29]:

$$\frac{\partial \mathscr{E}(n)}{\partial y_j(n)} = -\sum_k \delta_k(n) w_{kj}(n),$$

where the subscript k refers to the layer in the immediate right of hidden layer j.
Therefore, we have that the final formulation of the back-propagation for neuron j in a hidden layer is:

$$\delta_j(n) = \varphi'(v_j(n)) \sum_k \delta_k(n) w_{kj}(n).$$

BACK-PROPAGATION WITH BINARY CROSS ENTROPY LOSS

In classification problems, a different cost function is used. More precisely, what we use is the binary cross entropy loss and using this form implies changes in the back-propagation algorithm. The binary cross entropy loss has the following form:

$$\mathscr{E}(n) = \sum_{j \in C} \left[-d_j(n) \ln(y_j(n)) - (1 - d_j(n)) \ln(1 - y_j(n)) \right],$$

where, again, C represents the set of all the neurons belonging to the same layer. We can decompose the derivative of the error with respect to the weights as:

$$\frac{\partial \mathscr{E}(n)}{\partial w_{ji}(n)} = \frac{\partial \mathscr{E}(n)}{\partial y_j(n)} \frac{\partial y_j(n)}{\partial v_j(n)} \frac{\partial v_j(n)}{\partial w_{ji}(n)}.$$

These quantities are easy to calculate and result in the following for a node in the output layer [30]:

$$\frac{\partial \mathscr{E}(n)}{\partial w_{ji}(n)} = (y_j(n) - d_j(n)) \varphi(v_i(n)) = (y_j(n) - d_j(n)) y_i(n),$$

and in the following form for a node in a hidden layer:

$$\begin{aligned} \frac{\partial \mathscr{E}(n)}{\partial w_{ji}(n)} &= \sum_k (y_k(n) - d_k(n)) w_{kj}(n) \underbrace{y_j(n)(1 - y_j(n))}_{\varphi'(v_j)} \varphi(v_i(n)) = \\ &= \varphi'(v_j(n)) \sum_k \delta_k(n) w_{kj}(n) y_i. \end{aligned}$$

4.3. MACHINE LEARNING VS REGRESSION

We would like to say, from the beginning, that a model which is overall better than another model in every circumstance does not exist. Linear and logistic regression have fewer parameters to train than an ANN for the same inputs, but they are only able to get linear relations between the independent and the dependent variables. It is true that it is possible to complicate the regression model by adding some terms which are transformations of the predictors, nevertheless we do not know the functional form that explains these relations in advance and, therefore, rather than making subjective guesses it is better to learn it from the data, as a machine learning algorithm would do. In theory an ANN should be able to automatically model nonlinearities, the *universal approximation theorem* states that a feed forward network with just one hidden layer and a finite number of nodes can approximate continuous functions under some assumptions. More precisely, let N_0 be the number of input nodes, $N = N_L$ be the number of output nodes, and in general N_l the number of nodes in layer l, the theorem states [29]:

Theorem 1. *Let $\varphi(\cdot)$ be a nonconstant, bounded, and monotonically-increasing continuous function. Let I_{N_0} denote the N_0-dimensional unit hypercube $[0,1]^{N_0}$. The space of continuous functions on I_{N_0} is denoted by $C(I_{N_0})$. Then given any function $f \in C(I_{N_0})$ and $\epsilon > 0$, there exists an integer N and sets of real constants α_i, b_i, and w_{ij}, where $i = 1\ldots, N_1$ and $j = 1,\ldots, N_0$ such that we may define*

$$F(x_1,\ldots,x_{N_0}) = \sum_{i=1}^{N_1} \alpha_i \varphi(\sum_{j=1}^{N_0} w_{ij}x_j + b_i),$$

as an approximate realization of the function $f(\cdot)$; that is,

$$|F(x_1,\ldots,x_{N_0}) - f(x_1,\ldots,x_{N_0})| < \epsilon,$$

for all $x_1, x_2,\ldots,x_{N_0}$ that lie in the input space.

The first equation in the theorem represents the output of a single hidden layer perceptron:

- the network is formed by an input layer of m_0 nodes and a hidden layer of m_1 neurons. $x_1,\ldots,x_{N_0}$ are the inputs,
- the weights of the i^{th} hidden node are $w_{i1},\ldots,w_{iN_0}$, and the bias is b_i,
- $\alpha_1,\ldots,\alpha_{N_1}$ are the weights of the output layer, and the output of the network is a linear combination of the outputs of the hidden layer.

This means that "a single hidden layer is sufficient for a multilayer perceptron to compute a uniform ϵ approximation to a given training set" [29]. This is just an existence theorem, it does not say anything about this model being optimal in the sense of computational time for training, or generalization.
The main advantage is, therefore, the capability of neural networks to see complicated nonlinear relations, even if they need lots of data to be trained on in order to do so. Moreover, if the number of predictors is large, a machine learning algorithm can be trained fast with many input variables while it is harder for simple regression. On the other hand, training a simple regression requires less data and it is, in general, faster.

4.3.1. LOGISTIC REGRESSION AS NEURAL NETWORK

In this section, we would like to show how logistic regression can be replicated through the means of a simple neural network. We would like to do that to underline the fact that classic logistic regression is a simple model in comparison to what one can achieve with artificial neural networks.
We proceed showing that a simple neural network with identity activation in the input layer, no hidden layers and the logistic function as activation function in the one and only output node is mathematically identical to a logistic regression.
For the input layer we have, as stated before, that $a_i^1 = x_i$. Now, there are no hidden layers, so these outputs a_i become the input for the output layer ($L = 2$) and we obtain:

$$v_j^L = \sum_{i=0}^{N_{L-1}} w_{ji}^L a_i^{L-1} = \sum_{i=0}^{N_1} w_{ji}^2 x_i, \ \forall\ j = 1,\ldots,N_1.$$

Each of these v_j^2 is then transformed using the activation function, which is chosen to be the logistic function $\varphi(x) = \frac{1}{1+e^{-x}}$, obtaining the following output:

$$y_j = a_j^2 = \varphi(v_j^2) = \varphi\left(\sum_{i=0}^{N_1} w_{ji}^2 x_i\right) = \frac{1}{1+e^{-\left(\sum_{j=0}^{N_1} w_{ji}^2 x_i\right)}}, \ \forall\ i = 1,\ldots,N_1,$$

which is exactly the same one of the logistic regression.
Moreover, if we select the binary cross entropy loss for the network and we pick as activation function the logistic function, we notice that the cost function that is minimized in this problem is the same as the one for logistic regression. In fact, given that $N_L = N_2 = 1$, and $\varphi(\cdot)$ is the logistic function we have:

$$\mathscr{E} = -\frac{1}{N_1}\sum_{i=1}^{N_1}\left[y_i \ln(\varphi(\mathbf{w}^2 \cdot \mathbf{x}_i)) + (1-y_i)\ln(1-\varphi(\mathbf{w}^2 \cdot \mathbf{x}_i))\right].$$

This formulation is indeed equal to the one for logistic regression expressed in equation (2.4).

4.3.2. ADVANTAGES OF NEURAL NETWORKS

The strength of neural networks lies in their parallel distributed structure, and their ability to learn from the data. This capability to learn is what makes a neural network so attractive to users, who are generally interested in applying the model on never seen before observations and are therefore looking for generalization properties.
Some important features of neural networks are:

- Nonlinearity: it is possible to have linear or nonlinear neurons, therefore it is not uncommon to have nonlinear networks. This is a particular kind of nonlinearity though. In fact, it is not distributed throughout the network. In general, nonlinear relations are learned throughout the layers in the network. It is the subsequent application of linear functions that is translated to the ability of catching nonlinearity.

- Input-Output relation: in supervised learning, the one we are going to use, the network is trained modifying the weights in each neuron using a set of labeled training observations. Each observation has a set of unique characteristics which correspond to the input, and a label of the desired outcome which represents the target vector. Then, the network is trained feeding it with observations randomly chosen from the set. While training, the neuron weights are modified according to an appropriate statistical rule in order to achieve the smallest difference possible between the actual target and the output of the model. This output is obtained by applying the network to the corresponding input. This procedure is repeated until we have no significant change in the weights anymore. The network creates an input-output mapping for the specific problem and without taking into account any prior assumption on the data.

- Adaptivity: artificial neural networks have the ability to adapt the neuron weights with changes in the training environment. It is possible to retrain an already trained network in order to take into account the small changes in the environment. Moreover, if the environment is non stationary, it is possible to modify the weights in real time. In general, the more adaptive the model, given that we have stability, the better the performance on a nonstationary setting. This trade off between adaptivity and stability of the model is known in literature as stability-plasticity dilemma [29].

- Fault tolerance: artificial neural networks have the possibility of being fault tolerant, or able to perform robust computations, meaning that their performance deteriorates slowly under adverse conditions. In fact, if a neuron is damaged, the quality of the model is affected. However, the damage has to be really extensive in order to seriously ruin the final results. This particular structure of neural networks helps with the handling of contextual information as well [31].

4.4. DECISION TREES

To use Merriam-Webster definition, a decision tree is "a tree diagram which is used for making decisions in business or computer programming and in which the branches represent choices with associated risks, costs, results, or probability". A tree is a flowchart structure where the nodes are an

assessment on the attribute, the branches are the outcome of the assessment, and each leaf (final node) is a class label. The whole path from root to leaf is the classification rule, and it allows the user to go from the information about an observation to the classification of its target value.
More formally, let's say that we want to predict a class Y from the input $X_1, X_2, \ldots, X_n$. To do that, we build a decision tree, and at each node we apply a test to one of the input X_i. Conditional on the outcome of the test, we will move either towards the left or the right sub-branch. In the end, a leaf node will be reached where the prediction is done.

Definition 1. *A binary rooted tree is a graph in which any two nodes are connected by exactly one path, one of the nodes is been selected to be the root and all the branches move away from the root. Moreover, each node has exactly two children as the tree is binary.*

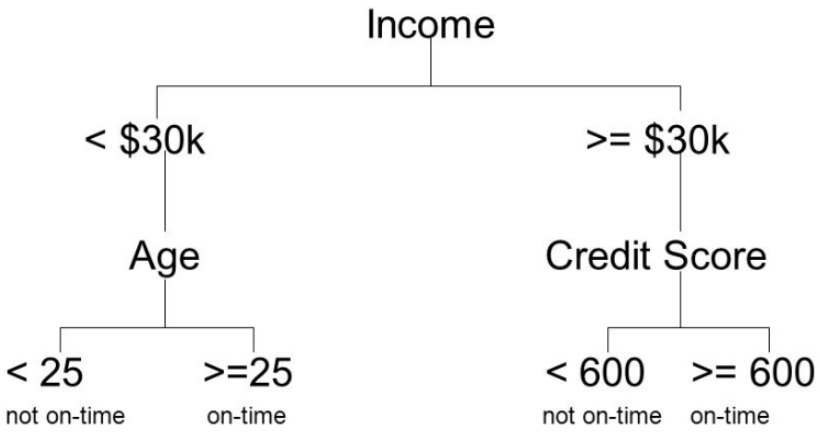


Figure 4.4: Example of a simple decision tree.

A visual example of a simple decision tree can be observed in Figure 4.4. For this case, the first slit in the root node is made looking whether the income is bigger or smaller than $ 30K. Then each child node is split using some other variables.

4.4.1. ADVANTAGES OF DECISION TREES

There are many factors that contribute to the attractiveness of decision trees in real world applications. The most important are the following:

- No parametrization: decision trees are non parametric models, which means that are able to model the complex relationship between input and output without the need of a priori assumptions.
- Heterogeneous data: decision trees are able to handle different kinds of variables, e.g. categorical and continuous variables, at the same time.
- Feature selection: feature selection in trees is intrinsically made in the building of the trees. This means that trees are robust with respect to non relevant variables and noise.
- Easiness of interpretation: trees are easy to understand even for non statisticians. Moreover, they are robust to outliers and errors in the target vector.

4.4.2. TREE CREATION

When handling data with complicated nonlinear relations, global models, such as regression, are complicated to build and when one manage to do it they are often confusing. An alternative approach is to partition the space in smaller subspaces where the relations between the variables are

simpler. Then, the subspace is partitioned again until a space in which it is possible to fit really simple models is reached. This procedure is called recursive partitioning. Decision trees are an example of recursive partitioning. Each leaf of the partition represents a subspace and has a model which applies only to itself. This means that a point X_i belongs to a leaf if X_i falls in the subspace related to that leaf. To see which subspace the variable belongs to, one has to start from the starting node (root) and follow the tree [32]. Cause of the way they are built, if there are two different observations which are identical in the input space but their target is different, then trees are not able to correctly predict both the observations.
To build a good tree, we have to find a good partitioning of the data set. By doing so, we try to maximize the information the variable X_i gives about the dependent variable Y. First, we start finding the assessment on variable X_i which gives the maximum information on Y, this will become the root node. In the root node the assessment takes place and the observations split in two sub-nodes. Then, the same procedure is repeated recursively for each child node. This means that a tree uses one feature at the time, if more features are equally predictive, which one is taken into account first is an arbitrary decision or a matter of chance.

SPLITTING CRITERIA

There are different criteria based on which to split the samples at each node. We will now present a couple of them, the first one is quite intuitive and it will give the reader an idea of what is happening, while the second one is the most commonly used in practice [33].
The first criterion consists in minimizing an error defined as follows. Let's introduce the variance for leaf c as:

$$V_c = \frac{1}{n_c} \sum_{i \in c} \left(y_i - \frac{1}{n_c} \sum_{i \in c} y_i \right)^2 .$$

Then we can define the sum of squared errors for the tree T as:

$$S = \sum_{c \in \text{leaves}(T)} n_c V_c. \tag{4.5}$$

The splits are then performed minimizing the sum of squared errors S.
The other possibility for the splitting is using the Gini impurity. This indicator tells us how common it is to incorrectly classify a randomly chosen observation if it was randomly classified according to the distribution of targets in the set. It is practically computed making the sum of the probabilities for an observation with label i to be correctly classified, p_i, times the probability of a wrong classification of that observation, $1 - p_i$. If we are in a binary classification setting, then the Gini impurity is computed as:

$$\text{Gini} = \sum_{i=1}^{2} p_i (1 - p_i) = 1 - \sum_{i=1}^{2} p_i^2.$$

STOPPING CRITERIA

A stopping criterion must be chosen carefully in order to achieve a good compromise between performance and complexity. Too big and complex structures could capture noise and lead to overfitting for example.
The main stopping criteria used are the following:

- Setting the minimum amount p of observations in every leaf.
- Setting the minimum amount q of observations in each node in order to have a split.
- Setting a threshold δ for the decrease in the impurity after a split. Basically the algorithm stops splitting if the amount of information gained is less than a certain threshold.

- Setting a limit D to the depth of the tree.

All these are free parameters for the user to tune in order to find the right trade-off. Too large trees will have a larger generalization error due to overfitting, while too small trees will lose some information from the data resulting in a larger generalization error as well.
We can now write the recursive algorithm as follows:

Algorithm 1 Decision Tree Construction

```
procedure
    c=createNode()
    while TreeDepth<D do
        Compute S as in equation (4.5)
        if ∀x_c : x_c ∈ Class(y = 0) or ∀x_c : x_c ∈ Class(y = 1) then:
            The node is a leaf node
            return
        Find the binary split that reduces S the most
        Compute the impurity reduction Δi
        Count the number of observations in each child node c_L and c_R
        if Δi < δ or c_L < q or c_R < q then:
            The node is a leaf node
            return
        Set the child node as the parent node
```

The problem with this simple algorithm is that it may stop too early. A way to overcome these issues is to use cross-validation. The data set is split in a training and a validation set, and the algorithm is applied to the training set putting the thresholds to $q = 1$, and $\delta = 0$, meaning that we generate the biggest possible tree. This tree is generally too big and will overfit the data, that is why we will then use the validation set to perform cross-validation to *prune* the tree. Pruning means to cut some leafs or branches from the tree, which means reducing the dimension of the tree. To do that, we look at the error calculated on the validation set for each pair of leaf nodes that has the parent node in common, and see if by eliminating these two nodes the total error would reduce. If that happens, we prune the tree which means that the two leaf nodes are eliminated and the parent node becomes a leaf, otherwise no pruning is done.

4.5. RANDOM FORESTS

Random forests are machine learning techniques used for both classification and regression. They are a collection of *Decision Trees* generated using random samples of the data. Random forests are able to handle the variance-bias trade off, and to reduce both bias and variance. They reduce bias using lots of predictors, in fact a random forest is able to handle more predictors than observations, and reduce variance by averaging the results obtained in all the different trees that form the forest.
The main assumption behind this methodology is that a group of models with weak predictive power can be put together to obtain one with strong predictive power. Each decision tree, which forms a random forest, is created different from the other. These trees are generated using a starting data set, in which both independent (features) and dependent (outcome) variables are present (supervised learning) [34]. During the construction of each tree, before the splitting of each node we take a random sample of predictors as possible choice to define the split. Using this random sample of predictors at each split makes the fitted values across trees more independent. Moreover, randomly building these trees implies that each predictor has multiple possibilities of being chosen to define a split. And in these occasions, often the dominant predictors are not included which means that

variables with less predictive power could still be included in some fit. Therefore, all these trees are randomly generated, but they have a common feature. For every tree k, a random vector Θ_k is created independent from the other Θ_i but with the same distribution. The k^{th} tree is then built using the training set and Θ_k, which results in a classifier $h(\mathbf{x},\Theta_k)$, where $\mathbf{x}$ is the input vector [33]. When a many trees are created, they vote for the most popular class. This procedure is called Random Forest.

Definition 2. *A random forest is a classifier consisting of a collection of tree-structured classifiers* $\{h(\boldsymbol{x},\Theta_k), k = 1\ldots\}$ *where the* $\{\Theta_k\}$ *are independent identically distributed random vectors and each tree casts a unit vote for the most popular class at input* $\boldsymbol{x}$. *[33]*

These Θ_k, that we have just introduced, represent both the way to randomly select with replacement observations from the data set, and the way to randomly select without replacement the predictors for each split. Θ_k is a collection of integers.

4.5.1. BIAS-VARIANCE DECOMPOSITION

It is possible to decompose the expected generalization error in three components:

1. Noise: it is the residual error. It is not dependent on the model or on the training set, and could be considered as a lower bound for the generalization error for any model.
2. Bias: it measures the distance between the average prediction of the model and the true value of the parameter.
3. Variance: it represents the volatility in the predictions over the training sets.

This division is made to better assess underfitting and overfitting. Usually, high bias and low variance are associated with underfitting, while high variance and low bias are associated with overfitting. In order to achieve good results, we need to find the right trade-off between these two quantities.
In order to reduce the generalization error, a reduction in the prediction variance is needed whilst keeping the related bias at the same level or keeping the increase moderate. A way to easily do that is to build ensemble methods, random forests belong to this class.

ENSEMBLE METHODS

Ensemble methods are a combination of multiple models. They combine the results of each model to form a final prediction which has more predictive power than any of the single methods on their own. There are various reasons why ensembles usually work better than single models. A statistical reason is that when the data set is not informative enough, it is possible to find different models with the same performance, and averaging over all the models help reducing the risk of picking the wrong hypothesis. There is also a computational reason, in fact single models can have greedy assumptions or can find local optima. On the contrary, ensembles use models that are trained from different starting points and therefore can have a better approximation of the complete unknown underlying function. Moreover, in most of the cases the function we are trying to find is so complex that a single model cannot represent it, while a combination of methods could be able to predict this function.
Random forests belong to this class of methods and, in particular, are an ensemble of decision trees grown randomly and independently from one another. The reason why trees are aggregated in forests is because usually a tree classifier suffers from low bias and high variance (overfitting), and gathering them in an ensemble method reduces this variance. In fact, it can be proven [35] that the component related to the random effects in the decomposition of the variance is inversely proportional to the number of trees. This means that for the number of trees tending to infinity, this variance term would go to zero. Therefore, increasing the number of trees actually reduces the overall variance. It has been shown that after a certain amount of trees, a plateau for the performance of the model is reached meaning that the results obtained are not enough to justify the higher computational costs [33].

Once all the trees are created, each one independently from the other, every one of them casts a vote on the observations. The final result for the forest is obtained by averaging each single result over all the trees.

4.5.2. RANDOM FOREST CONSTRUCTION

Let X be the random variable that represents the predictors, and Y the random variable that represents the actual class for the data. Now, suppose we have a random forest classifier, $f(X,\Theta)$. Then, we can define the margin function as in [36]:

$$mr(X,Y) = P_\Theta(f(X,\Theta) = Y) - \max_{j \neq Y} P_\Theta(f(X,\Theta) = j),$$

where $f(X,\Theta)$ is a set of classifications which varies because of the random mechanism imposed by Θ. This margin function is, therefore, the probability of a correct classification over trees minus the maximum probability over trees of an incorrect one. The larger the margin, the more confidence in the prediction.
Using the margin function it is possible to define the generalization error:

$$g = P_{X,Y}(mr(X,Y) < 0).$$

Here, P is the probability over the space represented by X and Y. Basically, the generalization error focuses on what happens to the margin function for different realizations of the observation, and represents the probability of going from one class to another. A high probability is an indication that the assigned class is not stable over random samples from the same population.
It is proven by Breiman in [33] that the estimated generalization error converges to the population generalization error as the number of trees increases. The population generalization error is:

$$P_{X,Y}(P_\Theta(f(X,\Theta) = Y) - \max_{j \neq Y} P_\Theta(f(X,\Theta) = j) < 0),$$

which represents the probability that a vote will be overturned. This implies that as the number of trees grows, the random forest will less likely overfit the data.
If different realizations of the observation are taken into account, then the value of the margin function for a random forest changes for each realization. However, if we take the expected value of the margin function for different realizations, then the strength of a random forest could be defined as:

$$s = \mathbb{E}_{X,Y}[mr(X,Y)].$$

The bigger this expected value is, the better.
Breiman shows that an upper bound exists for the generalization error, and it is equal to:

$$g^* = \frac{\bar{\rho}(1-s^2)}{s^2},$$

where $\bar{\rho}$ is the average correlation over realizations of the data and trees. Clearly, we would like this quantity to be small, and s to be close to 1. This formula also shows that the upper bound for the generalization error depends on the strength of the single trees and on the correlation between them in term of the margin function.
Moreover, the generalization error does not increase with the number of trees. This information and the fact that there is no overfitting for an increased number of trees lead to generally prefer more trees rather than fewer trees.

RANDOM FEATURES

In order to improve accuracy and generalization, the features used to split in every node are chosen randomly among a set of features. This results in a reduction of the correlation $\bar{\rho}$ while maintaining the predictive power. Some of the properties related to this procedure are the following:

- It is faster than doing bagging or boosting because it is easily parallelized.
- It makes the tree more robust to noise and outliers.
- It allows the user to give an estimate of the correlation and the variables importance.

4.5.3. FEATURE IMPORTANCES

In random forests it is possible to directly rank the importance of an input variable [37]. Random forests are an ensembles of decision trees, and each node of a tree can be seen as a condition on a variable that splits the data set in two subsets with a common characteristic. The split is performed in order to reduce the impurity in the most effective way. While training a tree it is therefore possible to see which variable has the biggest influence on the impurity decrease.
More technically, if we define the decrease in impurity in a generic node as $\Delta I(s_i, t)$, where t represents the node we are considering and s_i represent the splitting decision associated to variable x_i, then it is possible to introduce the variable that measures the importance of feature i as follow:

$$M(x_i) = \sum_{t \in T} \Delta I(s_i, t), \tag{4.6}$$

where T represent one tree of the forest. Then, to obtain the importance for the forest we must find this quantity for each tree and average it over all the trees. The larger this quantity, the better. In fact, a big value of $M(x_i)$ means that we have a big decrease in the impurity, which is our final goal.

4.6. HOW TO TAKE INTO ACCOUNT ALGORITHMIC CORRECTIONS

Now that we have presented all the theoretical notions in the previous sections, it is possible to go into the details of how these machine learning models have been adapted to our particular needs.
In particular, in this section we are going to describe how the outcome of the classification model is affected by changes in the algorithm and therefore in the building process. The most effective correction turns out to be cost sensitive learning for this specific case. The main idea behind this technique is to make the model focus more on rare class observations reducing the bias towards the majority class. To do that, some modifications are introduced in the loss function or in the optimization of the model. Specifically, different weights for the different classes are added. More practically, the introduction of these weights enables the algorithm to assign different costs to different errors. While the algorithm is training, a misclassification of a prepayment observation, i.e. classifying as noPrepayment a prepayment observation, is this way interpreted more expensive than a misclassification of the noPrepayment class.

4.6.1. ASSESS CORRECTIONS IN NEURAL NETWORKS

In this section, we analyse how the basic algorithm for a neural network is modified in the case we are dealing with a classification problem on a highly imbalanced data set.
In this framework, the correction is applied to the loss function. Two weights are added in order to obtain a cost sensitive loss which penalizes the error for the rare class more than the ones for the majority class.

We start by introducing the most commonly used loss function for classification problems [38].

That is the binary cross entropy loss and is defined as follows:

$$\begin{aligned} l &= -\frac{1}{n}\sum_i y_i \ln\left(\frac{p_i}{1-p_i}\right) - \frac{1}{n}\sum_i \ln(1-p_i) = \\ &= -\frac{1}{n}\sum_i y_i \ln(p_i) - \frac{1}{n}\sum_i (1-y_i)\ln(1-p_i), \end{aligned} \tag{4.7}$$

where y_i is the realization of the observation and $p_i = P(y_i = 1|x)$, which represents the probability of belonging to the class $y = 1$ and it is given by the output of the model. It is easy to understand why this is a common loss function for a binary classification problem. The first term is related to the error for the class $y = 1$, in fact it is zero whenever the observation we are considering has target $y_i = 0$. On the contrary, the second term only exists when the target label is $y_i = 0$ and therefore it assesses the performance on the $y = 0$ class. The output probability is then magnified in the loss function using the logarithm, which allows the user to pass from values of p_i in $[0,1]$ to values that span in $[0,+\infty)$. Moreover, a property of the binary cross entropy loss is that it preserves marginal probability, which means that the average of the predicted probabilities equals the average proportion of ones in the target vector:

$$\frac{1}{n}\sum_i y_i = \frac{1}{n}\sum_i \hat{p}_i. \tag{4.8}$$

This is a property particularly needed in our framework as it will be better explained in the results chapter.
Now, we show the validity of this property in the simple case of no weights added. In general, the goal is to minimize this loss w.r.t. the weights β. In equation (4.7) no β appears, but it is already hidden in that formulation.
In fact, keeping in mind that the following relationship holds:

$$\ln\left(\frac{p_i}{1-p_i}\right) = \beta^T \mathbf{x}_i,$$

it is straightforward to derive the formulas for p_i and $1-p_i$ as functions of β:

- $p_i = \dfrac{1}{1+e^{-\beta^T \mathbf{x}_i}}$,
- $1-p_i = \dfrac{e^{-\beta^T \mathbf{x}_i}}{1+e^{-\beta^T \mathbf{x}_i}}$.

At this point, we have to solve the set of equations obtained by taking the partial derivatives of the loss function displayed in equation (4.7) w.r.t. the weights:

$$\frac{\partial l}{\partial \beta_k} = 0,\ k = 0,\dots K.$$

Observing that the values of the derivatives are:

- $\dfrac{\partial ln(p_i)}{\partial \beta_k} = x_{ik}\dfrac{e^{-\beta^T \mathbf{x}_i}}{1+e^{-\beta^T \mathbf{x}_i}}$,
- $\dfrac{\partial ln(1-p_i)}{\partial \beta_k} = \dfrac{-x_{ik}}{1+e^{-\beta^T \mathbf{x}_i}}$.

we obtain the following formulation for a general partial derivative of the loss function:

$$\begin{aligned} \frac{\partial l}{\partial \beta_k} &= -\frac{1}{n}\sum_i y_i x_{ik}\frac{e^{-\beta^T \mathbf{x}_i}}{1+e^{-\beta^T \mathbf{x}_i}} + \frac{1}{n}\sum_i (1-y_i)\frac{x_{ik}}{1+e^{-\beta^T \mathbf{x}_i}} = \\ &= -\frac{1}{n}\sum_i y_i x_{ik}\frac{e^{-\beta^T \mathbf{x}_i}}{1+e^{-\beta^T \mathbf{x}_i}} + \frac{1}{n}\sum_i \frac{x_{ik}}{1+e^{-\beta^T \mathbf{x}_i}} - \frac{1}{n}\sum_i y_i \frac{x_{ik}}{1+e^{-\beta^T \mathbf{x}_i}} = \\ &= -\frac{1}{n}\sum_i y_i x_{ik} + \frac{1}{n}\sum_i \frac{x_{ik}}{1+e^{-\beta^T \mathbf{x}_i}} = -\frac{1}{n}\sum_i y_i x_{ik} + \frac{1}{n}\sum_i x_{ik} p_i. \end{aligned}$$

Therefore, the weights $\hat{\beta}$ (the hat quantities are the actual values obtained for the parameters after the training) must satisfy the following equation:

$$\frac{1}{n}\sum_i y_i x_{ik} = \frac{1}{n}\sum_i x_{ik}\hat{p}_i,$$

where $\hat{p}_i$ is p_i calculated using $\hat{\beta}$ instead of β. This is true thanks to the property of invariance of the MLE.
Hence, if x_{ik} is a constant term for some k ($k = 0$ the intercept), then we can simplify it to obtain the following relation:

$$\frac{1}{n}\sum_i y_i = \frac{1}{n}\sum_i \hat{p}_i,$$

which as the same form as equation (4.8). This is translated by the fact that the average response rate in the data is equal to the average predicted probability of the model and therefore these marginal probabilities are indeed maintained [39].

Now, we will consider the case in which the weights are added and examine how this property is affected. In the light of what has been said, if our goal is to give different losses to observations belonging to different classes, then it is straightforward where to apply the weights in the loss function. In fact, as stated earlier, this loss function has two terms and each one is related to the error of a particular class.
The addition of two different weights to achieve different misclassification errors, changes the loss as follows:

$$l_w = -\frac{1}{n}w_1\sum_i y_i \ln(p_i) - \frac{1}{n}w_0\sum_i (1-y_i)\ln(1-p_i), \tag{4.9}$$

where w_0 and w_1 are the weights associated respectively to the negative ($y = 0$) and positive ($y = 1$) class. This change implies that the set of equations to obtain the value of the weights changes and therefore also the relation between the target vector and the output of the model. The new form for the derivative of the weighted log-likelihood is the following:

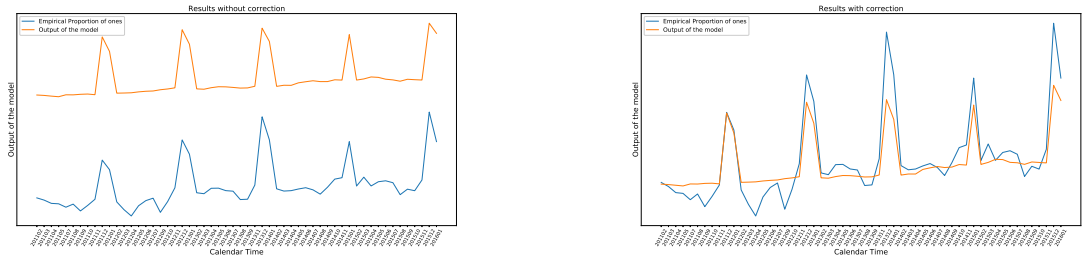
$$\begin{aligned}
\frac{\partial l_w}{\partial \beta_k} &= -\frac{1}{n}w_1\sum_i y_i x_{ik}\frac{e^{-\beta^T x_i}}{1+e^{-\beta^T x_i}} - \frac{1}{n}w_0\sum_i (1-y_i)\frac{-x_{ik}}{1+e^{-\beta^T x_i}} = \\
&= -\frac{1}{n}\sum_i y_i x_{ik}\frac{w_0 + w_1 e^{-\beta^T x_i}}{1+e^{-\beta^T x_i}} + \frac{1}{n}\sum_i \frac{w_0 x_{ik}}{1+e^{-\beta^T x_i}} = \\
&= -\frac{1}{n}\sum_i y_i x_{ik} p_i (w_0 + w_1 e^{-\beta^T x_i}) + \frac{1}{n}\sum_i w_0 x_{ik} p_i = \\
&= -\frac{1}{n}\sum_i \left(y_i x_{ik} p_i (w_0 - w_1) + w_1 y_i x_{ik} p_i \underbrace{(1+e^{-\beta^T x_i})}_{=1/p_i} \right) + \frac{1}{n}\sum_i w_0 x_{ik} p_i = \\
&= -\frac{1}{n}\sum_i y_i (x_{ik} p_i (w_0 - w_1) + w_1 x_{ik}) + \frac{1}{n}\sum_i w_0 x_{ik} p_i.
\end{aligned}$$

If, again, x_{ik} includes a constant term, the intercept, then also the $\hat{p}_i$ are constant for those observations. Therefore, the weighted form of the loss function implies the following new relationship between the output of the neural network and the target vector:

$$\frac{1}{n}\sum_i y_i = \frac{1}{n}\sum_i \frac{w_0 \hat{p}_i}{w_1 + (w_0 - w_1)\hat{p}_i}.$$

This means that the average response rate in the data is equal to the average of the above function of the predicted probabilities of the model.

In Figure 4.5 we see in practice what has been explained so far. In the left picture, we see the effects of adding weights in the loss function. From the plot it is clear that the model is overestimating the proportion of ones and this is in line with what expected. In fact, adding weights has the goal of putting more emphasis on the minority class $y = 1$, which is translated by the fact that the output probabilities are shifted towards one and therefore tend to be higher on average.



(a) Result without any correction

(b) Result with the correction

Figure 4.5: Artificial neural network results with and without corrections.

However, it appears that the pattern of the data is recovered from the algorithm, even if the scale is influenced by the weights.
In the right picture, the results for the same model with the correction applied are shown. Now, the desired properties of preserving marginal probabilities is respected. In fact, the model is able to follow both the trend and the scale of the observed quantity.

4.6.2. ASSESS CORRECTIONS IN RANDOM FORESTS

In this section, the effects of the corrections on random forests are taken into account and are analysed. The weights in this kind of model play a different role and they are not inserted in the loss function, since we do not have one when working with forests. Weights come into play in two different steps of the algorithm. First, they are taken into account at each node split in the Gini criterion for splitting. Second, the weights occur when calculating the probability of belonging to one class in the leafs of each tree.

We start taking into account the effects on the splitting rule. The normal Gini criterion measures how often a randomly picked observation would be not correctly classified if it was randomly assigned to a class according to the distribution of labels in the set. It is defined as follows:

$$\text{Gini} = \sum_i f_i(1 - f_i) = 1 - \sum_i f_i^2, \tag{4.10}$$

where f_i is the fraction of elements in the whole group belonging to class i. In order to give a more operative way to calculate the Gini measure we need to introduce some quantities. Let's define the sum of all the observations in a potential child node c as $t_c = \sum_i n_i$, where n_i is the number of observations in class i. The impurity of the child node is then given by:

$$i_c = 1 - \sum_i \left(\frac{n_i}{t_c}\right)^2. \tag{4.11}$$

Then, we define the impurity of the entire potential split as:

$$\text{Gini} = \sum_c \left(\frac{t_c}{t_p}\right) i_c,$$

where t_p is the total of all observations in the parent node, and t_c as in Equation 4.11.

Now, we introduce the weights and show how the Gini measure is affected. Let's define the weighted sum of all the observations in a potential child node c as $t_c^w = \sum_i w_i n_i$, where n_i is the number of observations in class i and w_i is the weight assigned to class i. The impurity of the child node is then given by:

$$i_c^W = 1 - \sum_i \left(\frac{w_i n_i}{t_c^w} \right)^2.$$

Again, we can define the impurity of the entire potential split as:

$$\text{Gini}_{\text{weighted}} = \sum_c \left(\frac{t_c^w}{t_p^w} \right) i_c^w,$$

where t_p^w is the weighted total of all observations in the parent node. Basically, it is the fraction of the total weighted observations in the parental node which is in each child node c multiplied by its impurity. Then, the potential split which has the lowest impurity is chosen. These changes help enhance the performance of the algorithm without shifting the average outcome of the model.

We now proceed to analyse the effects of the second correction. It is at this stage, with this modification, that the weights scale the output probabilities of the random forest. In fact, the final result of a random forest is obtained by averaging the results for each observation for each tree, but each single result of each tree is influenced by the weights.
At this point, we present how the output probabilities for each tree are calculated. Suppose that in a leaf node we have a observations belonging to class $y = 0$ and b observations belonging to class $y = 1$. Then, for that node, all the observations have a probability of belonging to class $y = 1$ equal to:

$$P(y = 1) = \frac{b}{a + b}.$$

If weights are added, let's say w_0 and w_1 for the class $y = 0$ and $y = 1$ respectively, then the probability of belonging to class $y = 1$ changes and assumes the following expression:

$$P_w(y = 1) = \frac{w_1 b}{w_0 a + w_1 b}. \tag{4.12}$$

From the two equations above, we can derive the relationship between $P(y = 1)$ and $P_w(y = 1)$ taking advantage of the relation that exists between a and b. From equation (4.12), it is possible to derive the following formula:

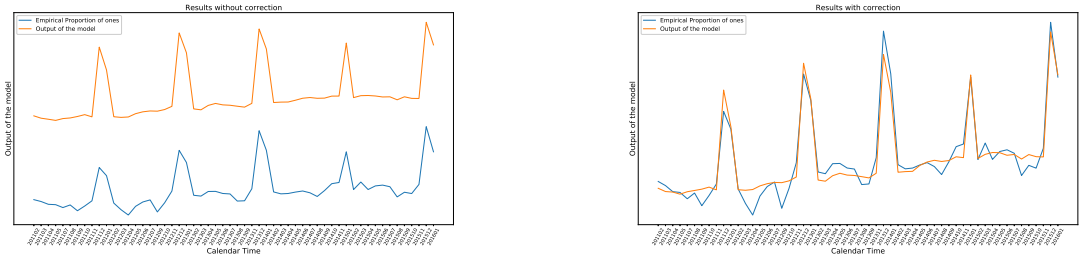
$$a = \frac{w_1}{w_0} \frac{1 - P_w(y = 1)}{P_w(y = 1)} b.$$

Knowing this, we can relate $P(y = 1)$ and $P_w(y = 1)$, in fact:

$$P(y = 1) = \frac{b}{a + b} = \frac{b}{\frac{w_1}{w_0} \frac{1 - P_w(y=1)}{P_w(y=1)} b + b} = \frac{w_0 P_w(y = 1)}{w_1 + (w_0 - w_1) P_w(y = 1)}. \tag{4.13}$$

Therefore, the relations between the outcome for the weighted model and the outcome for the general model is given by equation (4.13). It means that if we use the above function of the weighted outcome of each tree and then average over the whole forest, we obtain results in which the weighting has already been taken into account.

In Figure 4.6 it is possible to see in practice the effects of the weighting and of the corrections for the random forests. In the left picture, we see the effects of adding weights in the training process of the random forest. From the plot it is clear that the model is overestimating the proportion of ones and this is in line with what is expected. In fact, adding weights has the goal of putting more emphasis on the minority class $y = 1$. This is translated in the fact that the output probabilities are shifted towards one and therefore tend to be higher on average.



(a) Result without any correction

(b) Result with the correction

Figure 4.6: Random forest results with and without corrections

However, also for random forests, it appears that the pattern of the data is recovered from the algorithm, even if the scale is influenced by the weights.
On the right picture, the results for the same model with the correction applied are shown. Now, the desired properties of preserving marginal probabilities is respected. In fact, the model is able to follow both the trend and the scale of the observed quantity.

5

THE MODEL AND THE RESULTS

In the previous chapters, we introduced the theoretical knowledge and financial notions needed to develop a model to forecast the prepayment rate. In this chapter, we are going to present the details of the models that have been developed to forecast the CPR. In particular, we explain why this specific approach has been chosen and how we use a classification learning method in order to forecast a time series. Then, we will display the results that we obtained.

5.1. WHY A CLASSIFICATION APPROACH

The prepayment rate can be considered to be a time series. Usually, time series forecasting is solved with the help of univariate methods in which it is assumed that the future values of the time series depend only on the present and past values of the series [40]. Moreover, it is also common to use a multivariate model in which also additional time series variables are added to the model in order to enhance the performance.
Nevertheless, a data set with more than 90 million observations is available. It contains a lot of information about the clients, the mortgages and macroeconomic factors. We are interested in an exogenous approach, which takes into account individual characteristics, to give an insight in the prepayment behavior of mortgagors. In order to use all the information available, a time series forecasting has been immediately abandoned. In fact, in such an approach most of the information available would go lost.
The information that we have spans from February 2011 to January 2016 for a total of 56 months. The months are just 56 since some of them are missing as explained in section 3.1. This means that in a time series forecasting framework, the 90 million lines that are available would be reduced to just 56 observations, one for each month available. Moreover, all the information related to the clients or to the mortgages for a single month would go lost. In fact, either they are not take into account at all, or an average of the observations for each characteristic is considered, losing all the information on a single level. In addition, the average values for most variables are really similar between different months in which the prepayment rate is different, meaning that no useful information can be extracted this way.

Another possible approach is to use survival analysis to predict when the prepayment event is going to happen. The main reason why this approach is discarded is that it is not able to generalize its predictions for the forecasting set. In fact, when the model is applied to a set of observations that is disjoint in time the model fails as shown in Figure 5.1. Moreover, there is another drawback, but to explain it we need to first make a distinction.
We aim to forecast the prepayment rate of a portfolio of mortgages, but the information that we have is related to each single mortgage. This means that we can approach this problem looking at a loan

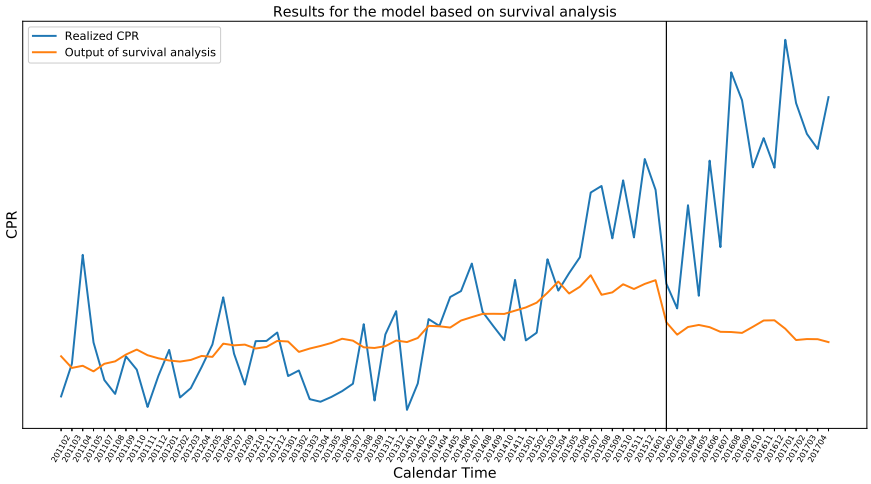


Figure 5.1: Results for a model based on survival analysis.

level, so basically aiming to understand whether each single mortgage is going to prepay or not and then aggregate the results, or looking at a portfolio level, where we are interested in the average value of the portfolio rather than on each individual. The main goal is to forecast the CPR, therefore more weight is given to the portfolio level. However, being able to correctly forecast each loan on a single level would be more insightful and efficient. In fact, if we know everything on a loan level, it is then easy to pass to the portfolio level while the other way around is not possible.
Going back to survival analysis, the drawbacks that we mentioned are related to the fact that the model is indeed able to follow the path of the CPR on the test set, but it is unable to learn which observations are more likely to prepay. It is just able to capture the average level of the total prepayment rate without giving any added information regarding the single mortgages. This behavior is also visible from Figure 5.1 where it appears clear that the model is just predicting the average value for the CPR and is completely unable to forecast any peaks in the prepayments pattern.

In order to overcome the problems stated earlier, we decide to use a different kind of approach from the ones already introduced. The model that will be developed is a three factor model based on classification. The choice of this approach is made following two main reasons. The first one is that this kind of approach allows us to incorporate behavioral and macroeconomic aspects to the model. The second one is that since we are focusing on classifying each single observation as belonging to one class or the other, we will have more insight also in a single loan level. Moreover, developing machine learning algorithms to complete this task, no need for assumptions in the distributions of the data set is needed.

5.2. THE MODEL

In this section we introduce in more detail the model and its characteristics. As touched upon earlier, it is a three factor model and each factor assesses a different aspect of the prepayment event. A simplified visual formula for the CPR can be seen in Figure 5.2. We will now proceed to introduce each term and explain its purpose.

Figure 5.2: CPR formula.

5.2.1. FIRST FACTOR:"CLASSIFICATION OUTPUT"

The first factor is the one that assesses whether a certain observation, a mortgage, is going to prepay or not. This first factor is the one that influences the final value of the predicted CPR the most. In fact, all the characteristics in the CPR pattern, like the seasonality, the upward trend or all the small peaks, are caught by the model at this stage through this classification problem. It is possible to achieve these results taking advantage of the property of preserving marginal probabilities in classification. In fact, assuming that all prepayment observations are full prepayments, this first factor would be enough to forecast the CPR. This is because the CPR would coincide with the proportion of ones that is given to the model as a target vector. However, most of the observations in the data set are partial prepayments, therefore a corrective term to take that into account is needed.

The values for this factor are obtained by means of a binary classification in which we assign each sample to the Prepayment or noPrepayment class. The inputs to the model are the information about the mortgage and the target value is 1 if the mortgage prepays, either it is a partial or a full prepayment, and 0 otherwise.
Precisely, we do not focus on the final assigned class, but directly on the output of the model which can be interpreted as the probability of belonging to the class. The reasons are mainly twofolds: the first one is that probabilities are more informative than the single class, output probabilities of 0.6 and of 0.9 would assign the sample to the same class but if the output is 0.9 the model is more certain about its collocation. The second one is that the property derived in chapter 4 states that the average proportion of ones is equal to the average of the output probabilities, and not to the average predicted proportion of ones.

Two different models are developed at this stage, one to forecast the CPR component due to movement prepayments and one for the voluntary prepayments component. These two models are obviously different, however, they are based on the same statistical methods. This means that two different random forests and two different neural networks are trained for movement and voluntary prepayments respectively. The only difference between them is the choice of the parameters.

5.2.2. SECOND FACTOR: "HEIGHT OF PREPAYMENT"

The second factor assesses the height of the prepayments, it checks whether the prepayment was partial or full. The addition of this term is needed since the first factor is not able to learn whether a prepayment happens or not and its magnitude at the same time. In fact, as mentioned in the previous section the first term is built in a way that considers all prepayments as full.
Since the nature of prepayments due to movement and voluntary is quite different, two completely distinct factors are introduced for the two classes of prepayments.

MOVEMENT

In the class of movement prepayments, approximately 80% of the observations belonging to this class represents a full prepayment. It means that most of the prepayments for this reason are full. For this reason, the correction term that needs to be applied is small, since approximating every prepayment as full would be an accurate approximation of reality in this case.
Following this reasoning, we decide to use a constant term to adjust the model given by:

$$h_M = \mathbb{E}[\mathrm{CPR}_M].$$

Basically, h_M is the average value of the CPR for the observations that have prepaid due to movement.

VOLUNTARY

In the voluntary prepayment class, the ratio between partial and full prepayment is quite different from the one for movement. This time, most of the prepayments are partial. Full prepayments account for just about 15% of the total observations. Therefore, this time we need to apply a bigger

correction to the first term of the formula.
To assess the size of a prepayment we develop a new classification model. In particular, it is again a classification problem but this time the two classes are not Prepayment and noPrepayment. The classes now are prepayments less or equal than 20% of the starting notional, and prepayments greater than 20%. This threshold of 20% has been selected because it is the threshold for applying a penalty or not in the Netherlands. Therefore, it is expected that small prepayments without penalty could have different characteristics than bigger prepayments where a penalty comes into play and therefore simply a positive refinancing incentive would not be enough to convince a mortgagor to prepay. In fact, if a penalty exists, a small incentive can not cover the costs that derive from it.

Before presenting the second factor for voluntary prepayments h_V, we need to introduce some other quantities. First, let's define the average of the prepayment rates for observations that have prepaid less than 20% of the starting balance as $\mathbb{E}_{<20\%}$. Analogously, we define $\mathbb{E}_{>20\%}$ as the average of the prepayment rates for samples that have prepaid more than 20%. Now, if we call the output of the model $\hat{y}$, then $\hat{y}$ represents the probability that the corresponding observation has prepaid more than 20% of the starting notional. The probability that a sample prepays less than 20% is then given by $1-\hat{y}$. Having defined these quantities, it is possible to obtain a formula for the second term:

$$h_V = \hat{y}\mathbb{E}_{>20\%} + (1-\hat{y})\mathbb{E}_{<20\%}. \tag{5.1}$$

The quantity in equation (5.1) is different for each observation and it depends on the likelihood of the observation to belong to one class or the other.

5.2.3. THIRD FACTOR:"STARTING BALANCE"

In Figure 5.2 the third factor is called starting balance, and it represents the notional on which the prepayment rate is based. It has been added to weigh the CPR for the notional of the mortgage. This way loans with a bigger notional have a bigger impact on the total CPR as it makes sense. In order to present the value of the CPR as a percentage, the total predicted amount prepaid for each month is divided by the total amount outstanding for that month. We are able to calculate an average prepayment rate for the whole portfolio following this procedure.

5.3. VARIABLE SELECTION

In our data set, we have more than 25 variables available as stated in chapter 3. In order to reduce the storage and computational costs, we decide to perform some variable selection. The first model to determine the first factor introduced in the previous section is a simple logistic regression. In fact, doing that gives a first impression of which variables are important and to have a lower bound for the performance of the algorithm, since we expect machine learning to work better than logistic regression. The variable selection has been done using the LASSO (Least Absolute Shrinkage and Selection Operator) method [41]. The variables that reveal to be meaningful in predicting the CPR for logistic regression are the ones that we presented in chapter 3 as meaningful in the literature.

NEURAL NETWORKS

In neural networks it is more complicated to look into the models because of their black box construction. Therefore, it is more complicated than in logistic regression to assess if and how each input affects the final output of the problem. A property of neural networks is that they should be able to perform variable selection automatically within themselves. In fact, what happens inside the network is that the neurons in the first layer tend not to be activated in the long run if the variable that enters is not meaningful for the prediction process. "not activate" means that the neuron outputs a 0.
Nevertheless, in order to not overfit the model and keep it computationally less expensive, we want to perform a variable selection. We start by training a model with all the available variables, then the ones that result not meaningful in the literature and in the logistic regression are shocked to see

whether the results are affected or not. If the results do not change, then the variable is also removed and an ulterior control of the output is made. If no change occurs, then we consider that variable as meaningless in the prediction process of the model. It is interesting to notice that the removal of some variables actually increased the performance of the model especially in the test and forecasting set. This may be due to the fact that a greater number of variables tends to overfit the training set.
Following this procedure, the resulting variables are shown in Table 5.1 and Table 5.2.

1	Original Notional	6	Age of the client
2	Interest rate incentive 1	7	Months to reset
3	Age of the mortgage	8	Indicator for December
4	Already prepaid once	9	Indicator for January
5	Interest rate incentive 2	10	Age of the client

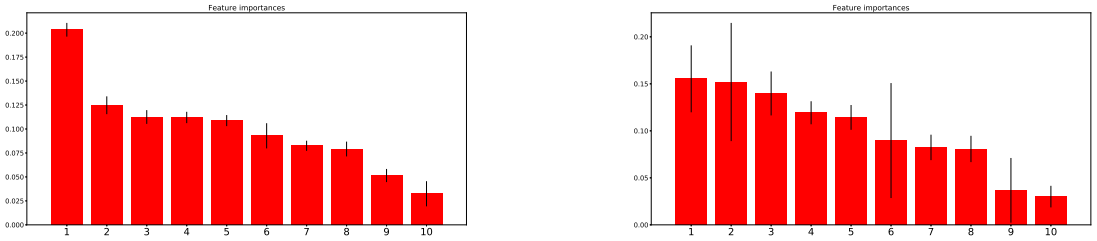
Table 5.1: Independent Variables for Voluntary prepayments

1	Age of the mortgage	6	Mortgage type
2	AgePattern	7	Months to reset
3	House sales	8	Age of the client
4	Original Notional	9	Rate age linear
5	Interest rate incentive 1	10	Unemployment rate

Table 5.2: Independent Variables for movement prepayments

RANDOM FORESTS

Regarding random forests, it is possible to get insight in the importance that each input variable has in the model. As touched upon in section 4.5.3, a quantity $M(x_i)$ defined in equation (4.6) exists that represents the decrease in impurity that we obtain by splitting using that variable.
In the light of this, the procedure we follow for variable selection is the following. We train a forest using all the predictors and compute the importance for each feature. Then, the variables with an importance significantly lower than the others are removed. The first tree built using more than 20 predictors is computationally really expensive and takes a long time to be trained. By removing one or two variables at the time, we reduce the set of significant variables for the two models until all the remaining features are too meaningful to be removed.
The variables that have been selected result to be the same as for the neural networks, therefore the reader can have a look at Table 5.1 and Table 5.2. For random forests, in addition we can look at the importance of these features.
A plot of the feature importances for both models is shown in Figure 5.3.



(a) Voluntary Prepayments (b) Movement Prepayments

Figure 5.3: Plot of the feature importances for the random forests

The red bars show the importance of the features in the forest, while the black lines are a representation of their inter-tree variability. The numbers under the columns identify the variables as introduced in the tables, e.g. 1 for voluntary corresponds to the original notional.
It is worth noticing that for the movement model the variable for the unemployment rate has the smallest importance, but it is fundamental in order to catch the upward trend present in movement prepayments.

5.4. IMPLEMENTATION AND RESULTS

In this section, we explain how the models are implemented and describe the choice of the parameters. Everything has been implemented in Python.

We start by developing an algorithm to build neural networks, and experiment it on some simple test data set like the *Iris* data set. The performance achieved on this set is comparable to the one obtained using a network built using the Python library *Keras*. All the network presented in the following sections have been implemented using the *Keras* library though due to computational limitation of our code.
For random forests, we use the well known *Scikit-learn* library which contains the ensemble module with which is possible to build ensemble based methods for both classification and regression.

5.4.1. ERROR MEASURE

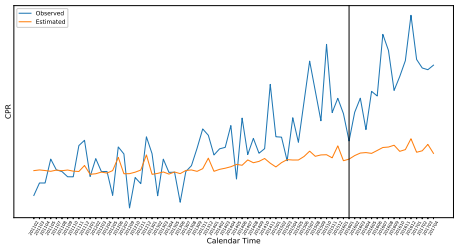
In the development of both models, the hardest part is to find a compromise between performances on a loan level and on a portfolio level. Which corresponds to finding a balance between the classification performance, correctly classify the observations, and the goodness of the approximation of the CPR.
The value for the annualized monthly CPR is obtained from the output of our models by averaging the prepayment rate of each single observation. However, during training the model it is not possible to take into account a measure of the error that we are making in estimating the CPR. In fact, this would mean that the algorithm should be able to group observations from the same month, average them and confront them with the value of the CPR for that month, which is not even an input to the model.

Before proceeding in showing an example of this phenomenon, we would like to introduce a measure of the goodness of the prediction of the CPR. In fact, we already introduced ways to assess classification performance in section 3.2.3, but we are still missing one to assess the distance between the CPR we estimated and the one we observed.
Let's define the error for method i at time t as $\text{Err}_{it} = |y_t - \hat{y}_{it}|$, where y_t is the observed value and $\hat{y}_{it}$ is the value predicted by model i. Then, we can introduce the average error over time for model i as:

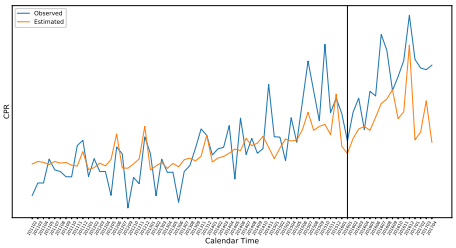
$$\text{Err}_i = \mathbb{E}_t[\text{Err}_{it}]. \tag{5.2}$$

This quantity represents the average distance between the predictions and the observed values in absolute terms. Clearly, we would like this quantity to be as small as possible.

In some cases, we obtain a model which is forecasting the CPR poorly while achieving classification results that are better than other models. An example of this can be seen in Figure 5.4, in the left picture the plot for the model with better classification performance is shown, while in the right graph the plot for the one with a better approximation of the CPR is displayed.



(a) Model with better classification performance



(b) Model with better approximation of the CPR

Figure 5.4: Examples of the difference between looking at it on a portfolio or loan level.

When we talk about classification performance, we refer to the AUC. The Auc for the left graph is of 69% and for the right one of 67%, meaning that the left one performs better for this measure. While when we talk about CPR estimation error we refer to the formula in equation (5.2). This error is clearly smaller for the right graph, meaning it is better at forecasting the CPR.
Both plots are subdivided in two parts by a vertical line, this line separates the test set from the forecasting set. Therefore, on the left side of each graph the results on the test set, never seen before observations but on the same time frame, are shown. On the right part, the results on the forecasting set, never seen before and disjoint in time observations, are shown.

5.4.2. IMPLEMENTATION

In this section, we give some details about the specifics for the implementation of the models. For both movement and voluntary prepayments we worked on subset of 10 million observations taken from the full dataset. This is done due to computational limitations of our machines.

VOLUNTARY

We start by giving the details for the model for voluntary prepayments. In order to achieve a more balanced dataset, we start applying some data level techniques. The one that results to be the more effective is undersampling. Two different approaches for the undersampling are tested, the first one is to remove observations which are far from the decision boundaries of the algorithm since they should be considered less relevant in the learning process of the algorithm [18]. The other one is random undersampling, which means randomly removing observations from the majority class. The results achieved applying both methods are comparable, therefore we decide to go for random undersampling to reduce the total computational time.

In the data set, the voluntary prepayment observation are $\approx 1.5\%$ of the total number, so that we are indeed in an imbalanced data set framework. The ratio between observations of the two classes in this subset is the same as the one for the full dataset.
The undersampling has been performed until reaching a ratio of Prepayment-noPrepayment observations of 1 to 4. This value has been selected as a compromise between the training times and the effectiveness of the correction. In this way, we had enough observations to get a significant training while maintaining low the computational costs.
It follows that we have a value for the β parameter introduced in equation (3.2) that is equal to 0.061. The reader can notice that multiplying the total number of observations by β, the number obtained is equal to more or less 4 times the number of observations in the prepayment class.

A perfect balance between classes is then achieved with the corrections on an algorithmic level. The correction we apply at this stage is based on cost sensitive learning. We modify the loss function as shown in equation (4.9) in order to give different magnitudes to different kinds of errors.
Let us call w_0 the weight related to the class $y = 0$, the noPrepayment class, and w_1 the weight associated to the class $y = 1$, the Prepayment class. Then, a common choice for the weights is the following

[42]. If we call n_0 the number of observations in the noPrepayment class, n_1 the number of samples in the other one, and N the total number of observations in the set, the weights are defined as follows:

$$w_i = \frac{N}{n_i}, \quad i = 0, 1.$$

For our framework, this means $w_0 = 1.25$ and $w_1 = 5$. We start from these two values, however we experiment also with different combinations. The values that turn out to have the best performance for the neural network are $w_0 = 1$ and $w_1 = 4$, while for the random forest are $w_0 = 1$ and $w_1 = 5$.

As for the set of parameters related to the models themselves, we test different combinations for both random forests and neural networks. We then take the set of parameters with the best compromise between classification performance and goodness in the estimation of the CPR.
Some of the settings for the neural networks together with the respective performance are visible in Table 5.3. In the table some abbreviations are used such as: #HL stands for number of hidden layers, #HN stands for number of hidden nodes, and TT stands for training time and it is expressed in seconds. The other parameters refer to the weights ($w_0 : w_1$), the batch dimension which represents how many observations are taken into account before updating the gradient in back propagation, and the number of epochs which represents the number of iterations in the training process.

Weights	Batch Size	#HL	#HN	#Epochs	TT (s)	Precision	Recall	F-Measure	Accuracy	AUC
1:4	128	1	20	200	462	0.0244	0.5114	0.0465	0.6577	0.6442
1:4	128	2	20	200	588	0.0266	0.4783	0.0504	0.6329	0.6491
1:4	128	1	30	200	476	0.0269	0.4616	0.0508	0.6484	0.6498
1:4	128	2	30	200	608	0.0269	0.4561	0.0507	0.6276	0.6477
1:4	64	1	30	200	592	0.0274	0.4497	0.0516	0.6377	0.6474
1:4	64	2	30	200	784	0.0250	0.5074	0.0477	0.6519	0.6473
1:4	128	5	20	200	871	0.0397	0.2584	0.0688	0.5464	0.6471
1:4	128	10	10	200	1508	0.0976	0.0853	0.0911	0.5475	0.6401
1:3	128	1	30	200	448	0.0409	0.2340	0.0696	0.5301	0.6456
1:5	128	1	30	200	452	0.0197	0.7067	0.0383	0.7307	0.6460

Table 5.3: Results on the validation set for different settings of the artificial neural network for voluntary prepayments

The combination that reveals to be the best is the highlighted one. The reader must keep in mind that the selected model is not the one with the best performance in every measure. A grater importance is given to the AUC for example, and also the results on the test and forecasting set are taken into account for the final selection.
For every model, the values for precision are low, which means that we classify some noPrepayment observations as prepayments. This looks like it is unavoidable due to the class imbalance present in the data set. Such values for recall mean that we are able to correctly predict more than half of the observation for prepayments. The AUC is the parameter we look at the most, since due to the imbalanced data set is the one that better assesses the performance.
Adding complexity to the model does not help the performance, it just increases the computational time. Performance are not increased from a reduction of the batch size either.

While for the random forest the reader should look at Table 5.4. Also in this table we use some abbreviations. #SF stands for number of splitting features which represents the dimension of the set among which the algorithm randomly choose the feature that uses to split the node. Min Split stands for the minimum amount of observations that we must have in a node to split it. Min leaf stands for the minimum amount of samples that we require in a leaf node. The quantities TT and weights are the same as in Table 5.3. The only remaining parameter is the one that represents the number of trees in the forest.

Weights	#Trees	#SF	Min Split	Min Leaf	TT (s)	Precision	Recall	F-Measure	Accuracy	AUC
1:5	200	4	30	30	100	0.0307	0.6368	0.0585	0.6959	0.7175
1:5	200	4	50	50	95	0.0273	0.6630	0.0525	0.6640	0.7267
1:5	200	4	70	70	91	0.0260	0.6847	0.0501	0.6141	0.7001
1:5	200	6	30	30	140	0.0318	0.6297	0.0605	0.7096	0.7136
1:5	200	6	50	50	133	0.0285	0.6639	0.0547	0.6592	0.7131
1:5	200	6	70	70	127	0.0267	0.6792	0.0514	0.6275	0.7049
1:5	200	8	50	50	169	0.0292	0.6627	0.0559	0.6673	0.7174
1:5	200	8	70	70	164	0.0272	0.6766	0.0523	0.6352	0.7087
1:4	200	4	50	50	95	0.0331	0.5355	0.0624	0.7576	0.7071
1:6	200	4	50	50	94	0.0235	0.7561	0.0455	0.5290	0.7068

Table 5.4: Results on the validation set for different settings of the random forest for voluntary prepayments

Again, the combination of parameters we use is the one highlighted in the table. The results are similar to the ones of Table 5.3. The training times for the random forests are generally smaller than for neural networks.

MOVEMENT

In this section the details for the model for prepayments due to movement are presented. Again, we start from data level techniques to achieve a more balanced set. As for the voluntary prepayment framework, we decide to apply random undersampling.

In the data set, we expect less movement than voluntary prepayments from the CPR composition showed in chapter 3. Therefore, for this setting we have classes that are even more imbalanced.
The undersampling has been performed until reaching a ratio of Prepayment noPrepayment observations of 1 to 9 this time. This value has been selected as a compromise between the training times and the effectiveness of the correction. The ratio is smaller in this occasion, the decision for this value is made in order to have enough observations to effectively train the model.
This time, the value for the β parameter is equal to 0.014. As we would expect, it is smaller than the one for movement. In fact, the number of observations to remove is bigger even though the ratio is smaller. In this occasion, multiplying the total number of observations for β, the number obtained is equal more or less to 9 times the number of observations in the prepayment class.

Like in the setting for voluntary prepayments, a perfect balance between classes is then achieved with the corrections on an algorithmic level. We use cost sensitive learning in this setting as well.
We make the same considerations on the weights w_0 and w_1 as in the previous section. The values that turn out to have the best performance are the same for both models, and they are equal to $w_0 = 1$ and $w_1 = 10$.

As for the set of parameters related to the models themselves, we test different combinations as explained in the previous section.
Some of the settings for the neural networks are visible in Table 5.5. For a more detailed explanation on the meanings of the headers of the columns please refer to the explanation made for Table 5.3.

Weights	Batch Size	#HL	#HN	#Epochs	TT (s)	Precision	Recall	F-Measure	Accuracy	AUC
1:10	128	1	4	200	65	0.0029	0.6463	0.0058	0.7192	0.6899
1:10	128	2	4	200	75	0.0025	0.7384	0.0051	0.7283	0.6829
1:10	128	1	10	200	69	0.0028	0.6472	0.0057	0.6911	0.6879
1:10	128	2	10	200	82	0.0029	0.6434	0.0058	0.6754	0.6866
1:10	64	1	4	200	81	0.0030	0.6163	0.0060	0.6817	0.6849
1:10	64	2	4	200	92	0.0035	0.5271	0.0070	0.6813	0.6742
1:10	128	5	4	200	130	0.0036	0.5058	0.0072	0.7019	0.6554
1:10	128	10	4	200	192	0.0017	0.5935	0.0034	0.6826	0.6631
1:9	128	1	4	200	66	0.0038	0.5174	0.0075	0.6211	0.6799
1:11	128	1	4	200	70	0.0031	0.6318	0.0061	0.7302	0.6888

Table 5.5: Results on the validation set for different settings of the artificial neural network for movement prepayments

The combination that reveals to be the best is the highlighted one. The considerations for these

results are similar to the one for Table 5.3. It is worth noticing how a greater imbalance in the classes, for movement the ratio Prepayment-noPrepayment is of 1/9 while for voluntary is 1/4, lead to lower values of precision. The training times are smaller just because the training set is smaller due to data limitations.

While for the various settings in the random forest framework, the reader should look at Table 5.6. For an explanation of the headers please refer to Table 5.4.

Weights	#Trees	#SF	Min Split	Min Leaf	TT (s)	Precision	Recall	F-Measure	Accuracy	AUC
1:10	200	3	20	20	24	0.0045	0.3895	0.0090	0.8520	0.6987
1:10	200	3	40	40	23	0.0039	0.4884	0.0077	0.7835	0.7015
1:10	200	3	60	60	21	0.0036	0.5310	0.0071	0.7462	0.6970
1:10	200	6	20	20	39	0.0049	0.3740	0.0096	0.8669	0.6980
1:10	200	6	40	40	38	0.0039	0.4574	0.0078	0.8005	0.6976
1:10	200	6	60	60	35	0.0037	0.5155	0.0074	0.7631	0.6985
1:10	200	8	40	40	52	0.0041	0.4671	0.0082	0.8062	0.6949
1:10	200	8	60	60	49	0.0038	0.5078	0.0075	0.7687	0.6954
1:9	200	3	40	40	23	0.0043	0.4612	0.0086	0.8064	0.6872
1:11	200	3	40	40	55	0.0039	0.4884	0.0077	0.7735	0.6981

Table 5.6: Results on the validation set for different settings of the random forest for movement prepayments

Again, the combination of parameters we used is the one highlighted in the table.

5.4.3. RESULTS

In this section, we proceed to show the results obtained in the prediction of the CPR. To obtain the final value for each month, the predicted values of the CPR from the same month are clustered together and then averaged. Our models are actually forecasting the single month mortality rate introduced in equation (3.1). Nevertheless, the results show the values for the annualized CPR obtained applying equation (2.1) to our estimation.

We start presenting the results obtained for voluntary prepayments. A plot for the estimated prepayment rate due to voluntary causes can be observed in Figure 5.5. In the picture, the results related to a logistic regression are visible as well. All the corrections and new variables are used in the logistic regression, and the reason why we are showing these results is to see whether machine learning algorithms actually perform better than standard logistic regression both in terms of AUC and the error measure defined in (5.2).

The graph is divided in two parts by a vertical blue line. In the left part of the graph, it is possible to see the performance of the models on the test set. While on the right part, the results for the forecasting set are shown.

In Table 5.7 the results for the AUC and the error measure defined by equation (5.2) are displayed. The abbreviations are as follows: LR stands for logistic regression, ANN for artificial neural networks, and RF for random forest.

Method	Set	AUC	Error
LR	Test	68.4%	0.9491%
	Forecasting	63.1%	
ANN	Test	69.5%	0.6955%
	Forecasting	63.9%	
RF	Test	69.8%	0.7068%
	Forecasting	64.4%	

Table 5.7: Values of Error and AUC for voluntary models.

Both machine learning models perform slightly better than the logistic regression. In particular, it seems that the neural network performs a bit better on a portfolio level, i.e. smaller error, while the random forest is more accurate on a loan level, i.e. higher AUC.

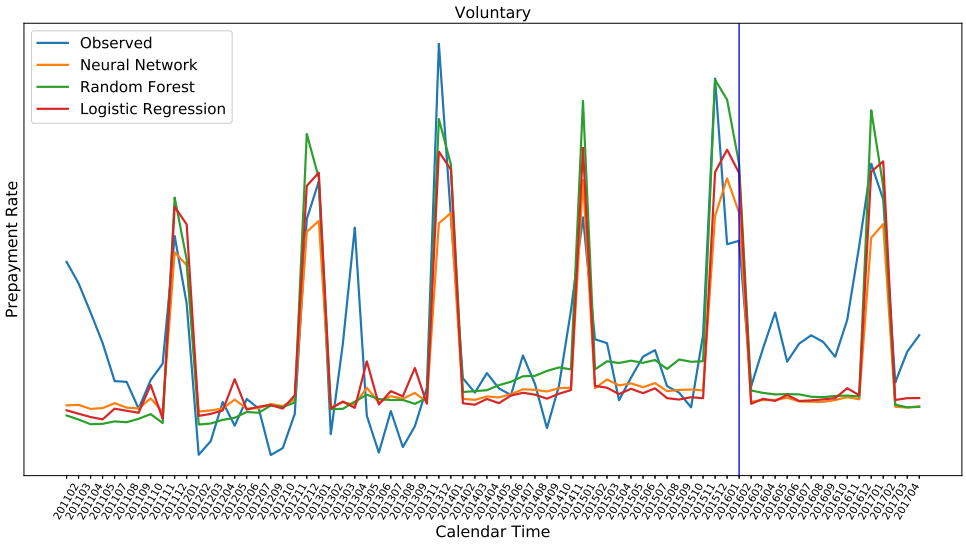


Figure 5.5: Results for voluntary CPR.

We proceed to show the results for the movement prepayment models as well. In Figure 5.6, the same division in two parts of the graph as in Figure 5.5 is performed. Also in this case, we are presenting the results obtained with a simple logistic regression. The values for the error measure defined in

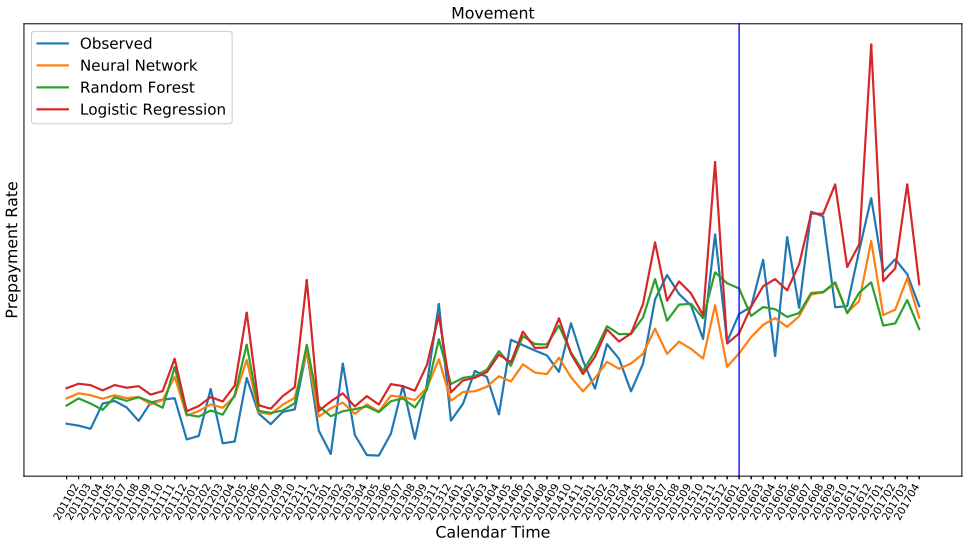


Figure 5.6: Results for movement CPR.

(5.2) and for the AUC related to these models can be found in Table 5.8.

Method	Set	AUC	Error
LR	Test	66.3%	0.5520%
	Forecasting	64.0%	
ANN	Test	68.6%	0.4915%
	Forecasting	66.6%	
RF	Test	68.8%	0.4872%
	Forecasting	67.7%	

Table 5.8: Values of Error and AUC for movement models.

We notice that all the models perform better for forecasting the movement prepayments rather than the voluntary ones, in fact we have higher values for the AUC and smaller values for the error measure. Moreover, also in this occasion the machine learning techniques are performing better than logistic regression.
In this example the random forest is the method that works best both on a loan level and on a portfolio level.

Last, we present the results for the total CPR, so for the prepayment rate for both causes. The plot is shown in Figure 5.7 and is obtained by summing the results obtained by the two models. The same division is made here as well.

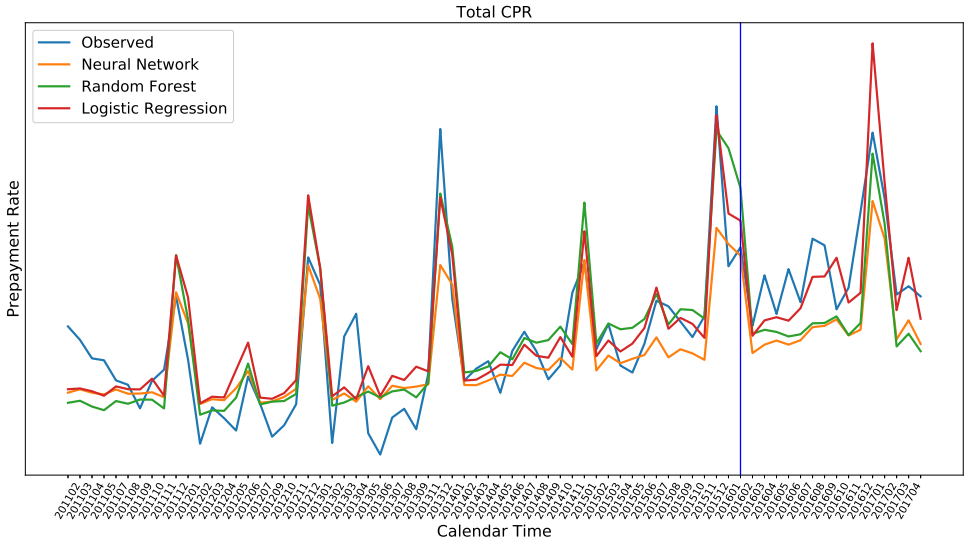


Figure 5.7: Results for the total CPR.

In this graph we can appreciate both the seasonality, typical for voluntary prepayments, and the upward trend that starts late 2014 shown in the movement prepayment rate.
This time, in Table 5.9 just the values of the error measure are shown.

Method	Set	Error
LR	Whole time frame	1.2636%
ANN	Whole time frame	0.9761%
RF	Whole time frame	0.9697%

Table 5.9: Values of Error for the total CPR.

Again, better results are achieved by machine learning techniques. The method that works best is the random forest. It improves the results of the prediction of the CPR of $\approx 0.3\%$ on average for each month. Such a small improvement could seem like meaningless, but we have to keep in mind

that these are absolute errors and we are dealing with percentages, therefore it is significant. In fact, considering a portfolio of 150 billion, which is a plausible quantity, a 0.3% difference in the CPR is translated by a difference in the expected amount prepaid of 450 million. Such a quantity has an impact on the hedging strategy of a bank when calculating the amortizing notional of the IAS introduced in equation (2.2).

5.5. SENSITIVITY ANALYSIS

In this section we perform a sensitivity analysis to assess the degree of confidence in our forecasts. In particular, this analysis will be performed on the random forests because it is the method with the best performance.

5.5.1. CONFIDENCE

The plot of the estimation of the CPR together with its confidence intervals for voluntary prepayments can be seen in Figure 5.8. In the picture, the 95% confidence intervals are shown. The observed value for the CPR is not always within these intervals.

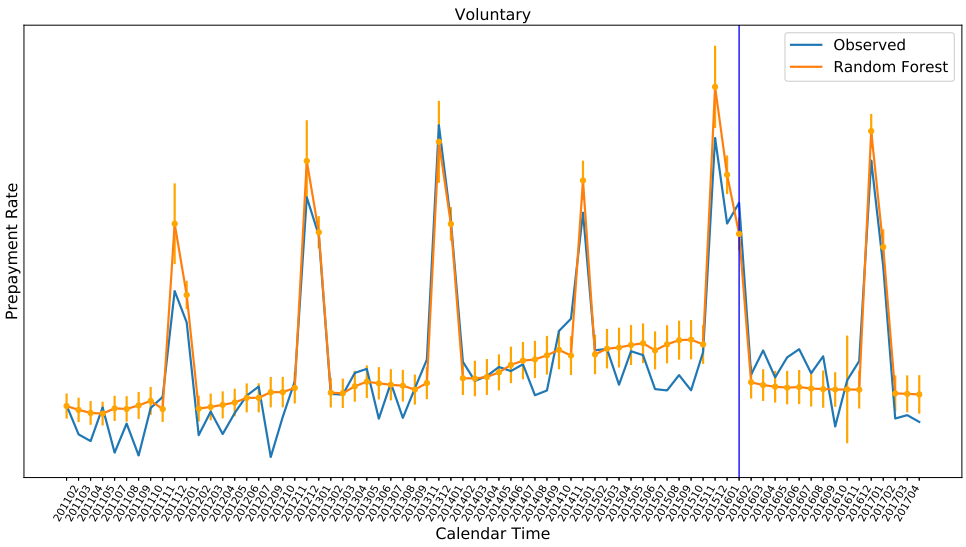


Figure 5.8: Plot of the results with a 95% confidence interval.

In order to make sure that all the values of the observed CPR are in the confidence intervals of the prediction, we need to lower the confidence to 90%. As expected, for the same degree of confidence we have slightly larger intervals for the forecasting set. At the peaks the intervals are always the widest due to a greater variability of the values of the CPR.

For the movement prepayments model based on random forests the results with the respective 95% confidence intervals are displayed in Figure 5.9.

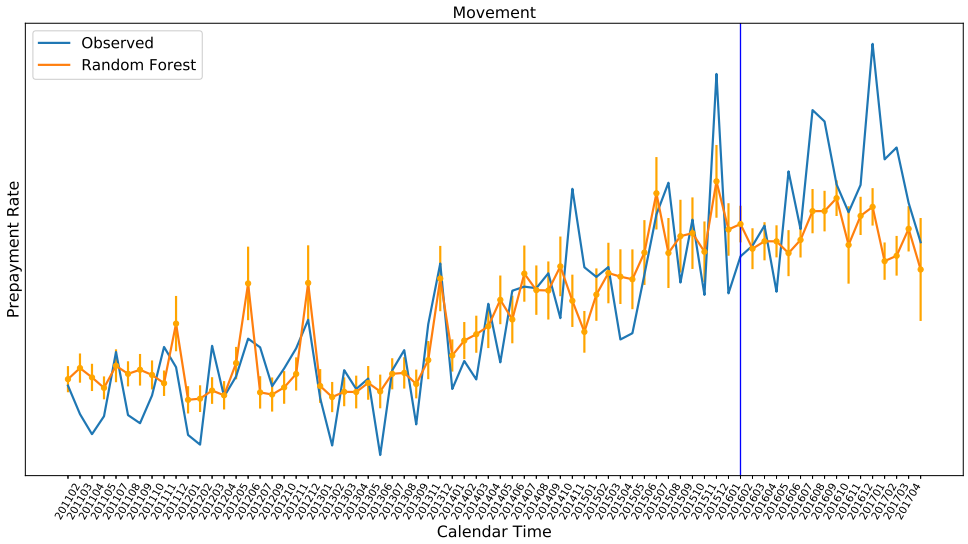


Figure 5.9: Plot of the results with a 95% confidence interval.

Also in this case, the observed values of the CPR are not always contained in the confidence intervals. The considerations made earlier for the plot for voluntary prepayments apply to this model as well. In the forecasting set part, the prediction seems a bit off and that could be due to the significant change in the prior distribution respect to the training set. For a more detailed explanation of this phenomenon please refer to appendix B.

5.5.2. PARTIAL DEPENDENCE PLOT

In this section, we give an insight in how the random forests are working. In particular, we show how the input variables affect the output of the models. A way to do that is through the partial dependence plot.
A partial dependence plot shows how each variable in the input affects the prediction of the model. It is calculated by changing the value of the variable under consideration while keeping all the others the same, and then looking at how the predictions change. These plots show the variation in the output, therefore negative values are also shown meaning that the output would have been less than the average for that value of the input variable we are considering.

We show the partial dependence plots for a couple of variables which appears to have a great importance in the prediction process of the forest.
We start by showing in Figure 5.10 and in Figure 5.11 the plots for the effects of the interest rate incentive and the original notional of the mortgage in the random forest developed for voluntary prepayments.
In the graphs, each cross represents an observation, if it is green it means that for bigger values of the variable the corresponding output increases, if it is red it means the opposite. The thicker line with yellow details is the line that represents the overall relation between output and input, and it is obtained by averaging the other thinner lines which represent the behavior of clusters of observations, plotting a line for each observation results in an unreadable graph and is therefore avoided.

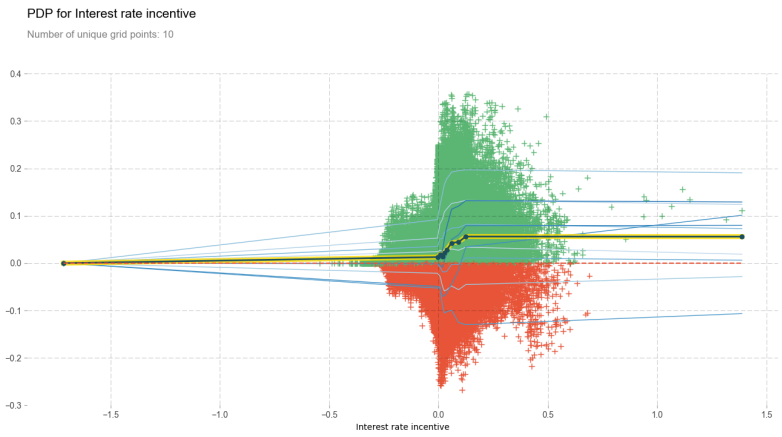


Figure 5.10: Partial dependence plot for the interest rate incentive in movement prepayments.

It appears that negative values of the incentive do not influence the prepayments on average since positive and negative components cancel each other out. Then, at 0 we have that for bigger values of the incentive the prepayments increase until reaching a plateau.

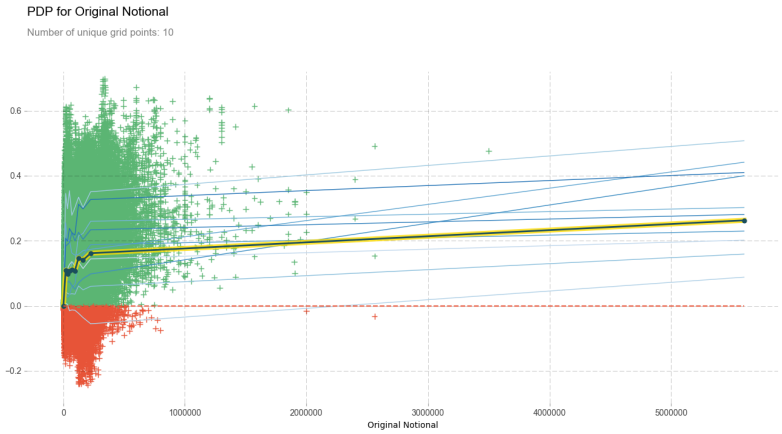


Figure 5.11: Partial dependence plot for the original notional in movement prepayments.

Regarding the plot related to the original notional we can see that in general a bigger original notional results in more chances to prepay, but that after a certain amount, i.e. $\approx 30K$, the impacts stays stable.

Now, we proceed to show the partial dependence plots for the movement prepayments model. The results for the age of the mortgage and the house sales variables are displayed in Figure 5.12 and in Figure 5.13.

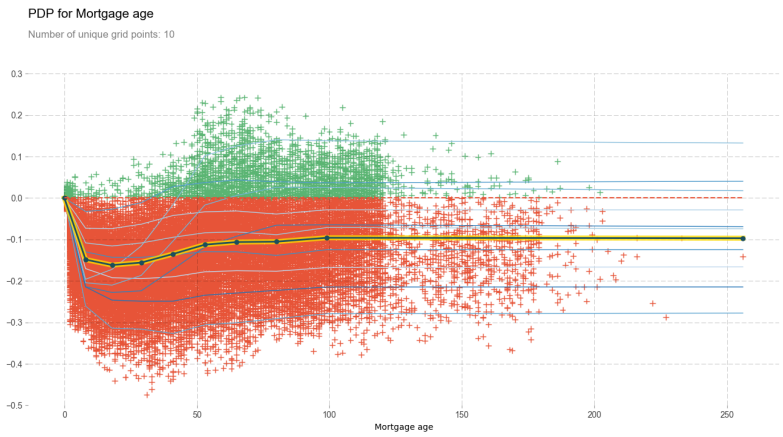


Figure 5.12: Partial dependence plot for the mortgage age in voluntary prepayments.

For the age of the mortgage, we see that small values of the age are strictly associated to non prepayment events. Moreover, in general the age of the mortgage has negative correlation with the output of the model.

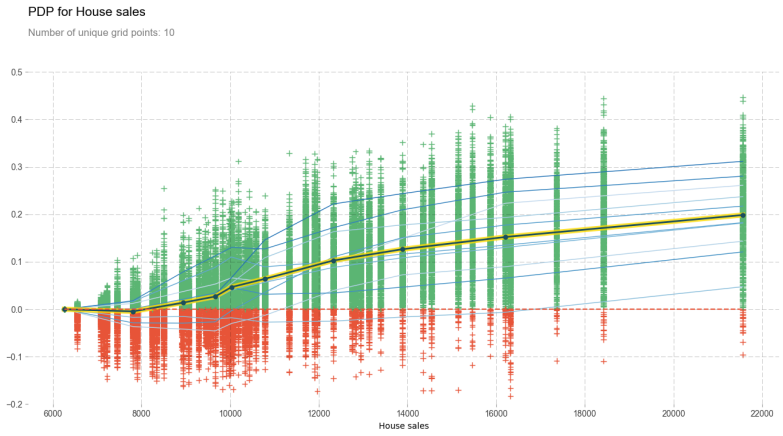


Figure 5.13: Partial dependence plot for the house sales in voluntary prepayments.

In the graph related to the house sales, the observations are distributed in a specific way due to the nature of the variable, being a macroeconomic variable it just assumes a discrete set of values for each observation. Moreover, the reader can see that, for a higher number of houses sold, the output increases which is translated by more prepayments, as one would expect.

6

Conclusion and Outlook

In the previous chapters, we explained how we developed a model based on machine learning to predict the prepayment rate in a mortgage portfolio. In this section we proceed to give a summary of our work and then to draw a conclusion. A section with recommendations for future research is also present.

6.1. Summary and conclusion

Throughout this thesis we have investigated the exogenous prepayment models to forecast the prepayment rate. The CPR, conditional prepayment rate, has been introduced and explained in chapter 2 together with the basic notions of a mortgage contract and of a prepayment option. Then, we explained that interest rate risk arises from prepayments due to their influence in the hedging strategy of a financial institution, and what are some famous drivers for prepayments in the literature.
An insight on the existing methods to forecast the CPR is carried out. In particular we did not focus on option theoretical models but on exogenous models. Most of the existing methods were based on logistic regression and survival analysis. We decided to have a classification approach rather than using survival analysis because of its limitations in generalize the results on data never seen before. A time series analysis approach was also investigated but discarded due to the loss of behavioral information in the implementation of such a method.

We continue introducing the data set that we use to train our models. In particular, we show how most of the phenomena we would expect from literature are actually present in our data set as well. For instance, we show how for higher values of the interest rate incentive we actually have more prepayments, but also that a significant amount of prepayments is present for negative values of it meaning that people do not always act following only economically rational principles.
Moreover, looking at the distribution of prepayments against some of the variables we were able to create a variable, related to the age of the mortgagors, which ended up being of greater impact in the performance of the models. Also a macroeconomic variable is introduced to help the model in following some low frequency trend. This variable is introduced with a lag, meaning that an observation for today has the value for that macroeconomic factor of three years ago, in order to be able to use the models for forecasting.
Then, we proceed to explain the problem of having a data set which has imbalanced classes, the prepayment observations account for just $\approx 1.5\%$ of the number of total observations. Some techniques to tackle this problem are introduced, and their effects on the final results are shown. In particular it is shown how undersampling affects the prior distribution in the data and which corrections must be made in order to take that into account.

In chapter 4, we gave the theoretical background needed in order to develop such a model. The

details on how an artificial neural network and a random forest work are shown. Then, how these algorithms have been modified to adapt to our particular problem is explained. What was done was basically adding weights in the loss function or in the impurity function in order to give different weights to different kinds of error. This means that classifying a prepayment observation in the no-Prepayment class has a bigger impact in the total loss, or in the total decrease of the impurity, than classifying a non prepayment observation as Prepayment. This makes sense since algorithms tend to be biased towards the majority class and in this way a greater focus goes the the rare class.

The model that we used to forecast the CPR is now introduced. It is a three factor model in which each factor assesses a different aspect of prepayments: the first one determines whether a mortgage will prepay or not, the second one assesses the magnitude of this prepayment, whether it will be full or not, and the third one is the one that weights the previous quantity for the notional amount, a bigger loan has a greater influence on the amount prepaid.
The model we developed was able to follow the general behavior of the observed CPR. The seasonality was well learned thanks to the use of indicator variables for the months interested in the seasonality. In addition, the slow upward trend is learned by the model as well. All the smaller peaks, the high frequency components, were not properly caught probably because too random to be predicted without overfitting.
Among the models, the ones based on machine learning techniques performed better than the one based on logistic regression both in terms of absolute error and in terms of AUC. In particular, the one based on random forest resulted to be the model with the best performance reaching a result that is on average 7.5% more accurate than logistic regression. This is translated by an average improvement of 450 million in the estimation of the amount prepaid for each month. This enhancement in the forecast impacts the hedging strategy of a financial institution.

6.2. FUTURE RESEARCH

One of the main limitation of the models developed within this thesis is the framework on which they are trained. What we mean is that our data set spans from 2011 to 2016, and in these years we just experienced lower and lower values for the interest rate. Therefore, it would be interesting to see how the models behave in a scenario where the interest rate is increasing.

Another limitation of the models is their incapability of adapting to shifts in the prior distribution of the target vector. Such a phenomenon is quite common in real life dataset. For instance, in our case we experience a constant growth in the prepayment rate and therefore the prior distribution of the forecasting set is significantly different from the one of the training and test set. Looking into techniques that are able to predict the new prior given the performance of the model on the new set would, in our opinion, greatly enhance the prediction power of the algorithm.

Moreover, we believe that the addition of new variables could be really beneficial in order to enhance the performance of the model. We say that after seeing the benefit brought by the addition of the unemployment rate variable.
Some of the variables that could be interesting to add are the loan to value ratio, the purpose of the loan, and another macroeconomic variable, like one that is related to the region of origination of the loan. It is expected that different areas have different kinds of mortgagors behaviors.

Last, the revolutionary change in this kind of approach would be to incorporate in the training algorithm a quantity that assesses the goodness of the approximation of the CPR. Being able to take into account in the loss function or in the impurity calculation how far our prediction is from the observed CPR would allow the models to modify themselves accordingly. This procedure would result in better results in the estimation of the CPR.

BIBLIOGRAPHY

[1] Dutch Banking Association (NVB), *The Dutch Mortgage Market*, Tech. Rep. (2014).

[2] J. Jacobs, R. Koning, and E. Sterken, *Modelling prepayment risk,* Dept. Economics, University of Groningen (2005).

[3] M. Mastrogiacomo and R. van der Molen, *Dutch mortgages in the DNB loan level data*, Tech. Rep. (2015).

[4] M. Sherris, *Pricing and hedging Loan prepayment risk,* Transactions of society of actuaries (1994).

[5] M. Consalvi and G. S. di Freca, *Measuring prepayment risk: an application to UniCredit Family Financing*, Tech. Rep. (Working Paper Series, UniCredit&Universities, 2010).

[6] A. Ploegh and H. Naaktgeboren, *Model documentation dynamic prepayment model.*, Tech. Rep. (Rabobank, 2016).

[7] Y. Goncharov, *An intensity-based approach to the valuation of mortgage contracts and computation of the endogenous mortgage rate,* International Journal of Theoretical and Applied Finance **9**, 889 (2006).

[8] J. Sirignano, A. Sadhwani, and K. Giesecke, *Deep learning for mortgage risk,* homepage: https://ssrn.com/abstract=2799443. pdf (online) (2016).

[9] L. S. Hayre, *Prepayment modeling and valuation of Dutch mortgages,* The Journal of Fixed Income **12**, 25 (2003).

[10] A. Goodarzi, R. Kohavi, R. Harmon, and A. Senkut, *Loan prepayment modeling,* in *KDD Workshop on Data Mining in Finance, edited by T. H Hann and G. Nakhaeizadeh* (1998) pp. 62–69.

[11] B. J. Alink, *Mortgage prepayment in the Netherlands*, Ph.D. thesis, University of Twente (2002).

[12] D. G. Kleinbaum and M. Klein, *Survival analysis*, Vol. 3 (Springer, 2010).

[13] A. P. J. M. van Bussel, *Valuation and interest rate risk of mortgages in The Netherlands*, Ph.D. thesis, University of Maastricht (1998).

[14] E. Charlier and A. Van Bussel, *Prepayment behavior of dutch mortgagors: an empirical analysis,* Real estate economics **31**, 165 (2003).

[15] H. He and E. A. Garcia, *Learning from imbalanced data,* IEEE Transactions on Knowledge and Data Engineering **21**, 1263 (2009).

[16] G. M. Weiss, *Foundations of imbalanced learning,* in *Imbalanced Learning* (Wiley-Blackwell, 2013) Chap. 2, pp. 13–41, https://onlinelibrary.wiley.com/doi/pdf/10.1002/9781118646106.ch2 .

[17] A. Tayal, T. F. Coleman, and Y. Li, *Bounding the difference between rankrc and ranksvm and application to multi-level rare class kernel ranking,* Data Mining and Knowledge Discovery **32**, 417 (2018).

[18] S. Kotsiantis, D. Kanellopoulos, P. Pintelas, *et al.*, *Handling imbalanced datasets: A review,* GESTS International Transactions on Computer Science and Engineering **30**, 25 (2006).

[19] A. Tayal, T. F. Coleman, and Y. Li, *Rankrc: Large-scale nonlinear rare class ranking,* IEEE transactions on knowledge and data engineering **27**, 3347 (2015).

[20] G. King and L. Zeng, *Logistic regression in rare events data,* Political analysis **9**, 137 (2001).

[21] K. Zhang, B. Schölkopf, K. Muandet, and Z. Wang, *Domain adaptation under target and conditional shift,* in *International Conference on Machine Learning* (2013) pp. 819–827.

[22] M. Saerens, P. Latinne, and C. Decaestecker, *Adjusting the outputs of a classifier to new a priori probabilities: a simple procedure,* Neural computation **14**, 21 (2002).

[23] A. Storkey, *When training and test sets are different: characterizing learning transfer,* MIT Press Scholarship Online (2013).

[24] A. D. Pozzolo, O. Caelen, R. A. Johnson, and G. Bontempi, *Calibrating probability with undersampling for unbalanced classification, 2015 IEEE Symposium Series on Computational Intelligence,* , 159 (2015).

[25] J. F. Puget, *What is machine learning?* https://www.ibm.com/developerworks/community/blogs/jfp/entry/What_Is_Machine_Learning?lang=en (2016), [Online; accessed 20-November-2017].

[26] T. Hill, P. Lewicki, and P. Lewicki, *Statistics: methods and applications: a comprehensive reference for science, industry, and data mining* (StatSoft, Inc., 2006).

[27] D. Graupe, *Principles of artificial neural networks*, Vol. 7 (World Scientific, 2013).

[28] I. Goodfellow, Y. Bengio, A. Courville, and Y. Bengio, *Deep learning*, Vol. 1 (MIT press Cambridge, 2016).

[29] S. Haykin and N. Network, *A comprehensive foundation,* Neural networks **2**, 41 (2004).

[30] P. Sadowski, *Notes on backpropagation,* homepage: https://www. ics. uci. edu/˜ pjsadows/notes. pdf (online) (2016).

[31] Y. A. LeCun, L. Bottou, G. B. Orr, and K.-R. Müller, *Efficient backprop,* in *Neural networks: Tricks of the trade* (Springer, 2012) pp. 9–48.

[32] C. Shalizi, *Classification and regression trees,* unpublished report on data mining procedures available at http://www. stat. cmu. edu/cshalizi/350/lectures/22/lecture-22. pdf (2009).

[33] L. Breiman, *Random forests,* Machine learning **45**, 5 (2001).

[34] S. Hartshorn, *Machine learning with random forests and decision trees, kindle,* Google Scholar (2016).

[35] G. Louppe, *Understanding Random Forests*, Ph.D. thesis, University of Liège (2014).

[36] R. A. Berk, *Statistical learning from a regression perspective* (Springer, 2016).

[37] F. Breiman and J. Friedman, *Olshen, and stone.(1984) classification and regression trees,* .

[38] M. Sugiyama and M. Kawanabe, *Machine learning in non-stationary environments: Introduction to covariate shift adaptation* (MIT press, 2012).

[39] G. S. Maddala, *Limited-dependent and qualitative variables in econometrics*, 3 (Cambridge university press, 1983).

[40] C. Chatfield, *Time-series forecasting* (CRC Press, 2000).

[41] R. Tibshirani, *Regression shrinkage and selection via the lasso,* Journal of the Royal Statistical Society. Series B (Methodological) , 267 (1996).

[42] S. Wang, W. Liu, J. Wu, L. Cao, Q. Meng, and P. J. Kennedy, *Training deep neural networks on imbalanced data sets,* in *Neural Networks (IJCNN), 2016 International Joint Conference on* (IEEE, 2016) pp. 4368–4374.

A

TIME SERIES APPROACH

We make an attempt to predict the CPR with the means of a time series forecasting model. The time dedicated to this model was limited.
In this framework, the future values of the time series depend only on present and past values of the series itself. Therefore, what we do is to extract the values of the CPR from the dataset, obtaining therefore 56 observations which represent the subsequent values of the CPR over a time period of more or less 5 years. These values are both the input and the output. The output is always the observation at time $t+1$, while for the input we have different possibilities. We can decide how far to look in the past values, meaning that we decide how many past observations the model receives for training, e.g. if we decide two time steps in the past, it means that to forecast the $t+1$ observation we give as input the observations at time t and $t-1$. This quantity is called look back window.
This kind of approach has been practically implemented with a neural network using the *Keras* library in Python. The network is a single layer network built with LSTM (long short-term memory) neurons since they are well suited for predicting time series. The reasons are multiple:

- Due to the ability of storing information, it is really good at recognizing patterns
- It allows multiple inputs. This means we can use correlated time series in the training process as well.
- It allows an easy selection of the look back window.

The results we obtain are displayed in Figure A.1. It appears that the results obtained by the classifi-

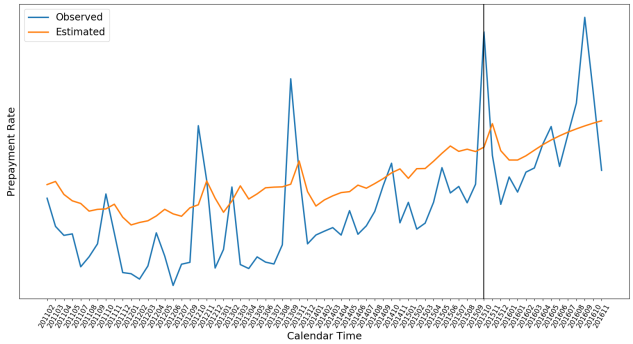


Figure A.1: Results for the time series based model.

cation based methods are better than the ones obtained using this model.

B

CHANGE IN THE PRIOR DISTRIBUTION.

It sometimes happens that observations in the training set do not reflect the real a priori probability of the target classes of the real data observations we would like to analyse. This could lead to bad performance of the models when applied to real world data sets.
Suppose we have two classes $\{\omega_0, \omega_1\}$. Let's call $\hat{p}_t(\omega_1) = \frac{N_1}{N_{tot}}$ the a priori probability for the training set, given by the proportion of ones in the data set. If we train a classifier, we can define its output posterior probability as $\hat{p}_t(\omega_i|x)$. Let us also define $p(x|\omega_i)$ as the within class densities and suppose they do not change between training and test set. Subscript t stands for training.
Then, the Bayes' theorem says that [22]:

$$\hat{p}_t(x|\omega_i) = \frac{\hat{p}_t(\omega_i|x)\hat{p}_t(x)}{\hat{p}_t(\omega_i)},$$

where $\hat{p}_t(x)$ the probability density function for the training.
If the prior changed, the corrected formula for the a posteriori probability then becomes:

$$\hat{p}(x|\omega_i) = \frac{\hat{p}(\omega_i|x)\hat{p}(x)}{\hat{p}(\omega_i)},$$

where $\hat{p}(\omega_i)$ are the new a priori probabilities and $\hat{p}(x)$ the new probability density function.
Keeping these formulations in mind, assuming that we know the new a priori probability, we can derive the following formula for the corrected a posteriori probability $\hat{p}(\omega_i|x)$:

$$\hat{p}(\omega_i|x) = \frac{\frac{\hat{p}(\omega_i)}{\hat{p}_t(\omega_i)}\hat{p}_t(\omega_i|x)}{\sum_{j=1}^{n}\frac{\hat{p}(\omega_j)}{\hat{p}_t(\omega_j)}\hat{p}_t(\omega_j|x)}$$

We observe the effects of this correction in the following plots. In Figure B.1 the results of an early model without the correction is shown. We can see that the average value of the target in the training part, left part, represented by the black horizontal line is way smaller than the average value in the right part. Due to its nature, the model tends to stay around this average also in the forecasting set resulting in bad predictions.

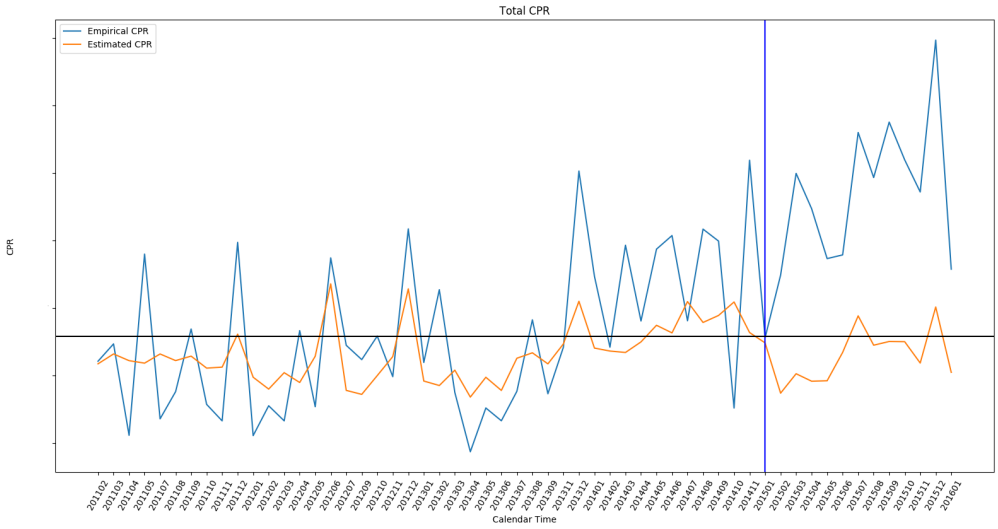


Figure B.1: Results without the correction.

If we calculate the new prior for the forecasting set from the data and we plug it in in the previous formulation, we obtain the results shown in Figure B.2.

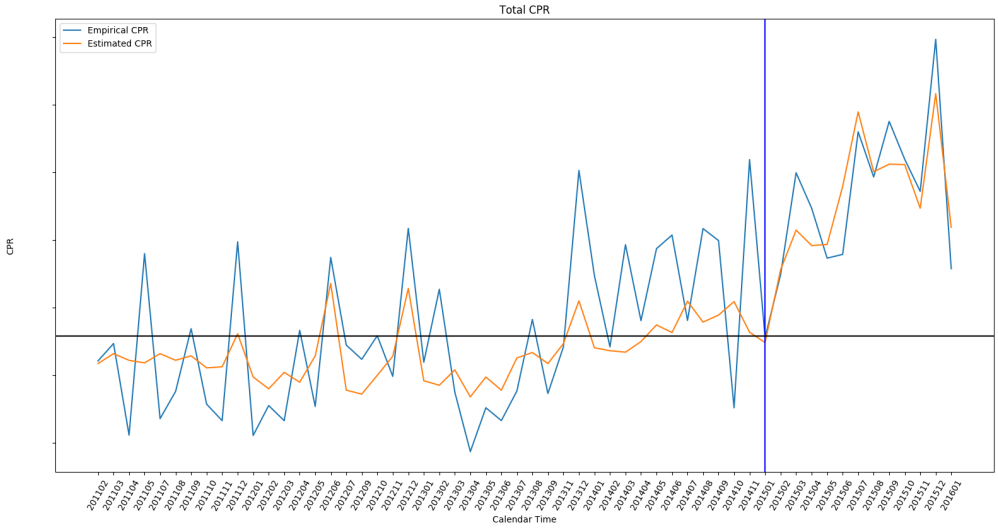


Figure B.2: Results with the correction.

Now, the prediction is really accurate. In fact, the behavior of the prepayments is learned by the model, the problem is in the prior shift.
In the real setting, we are interested in forecasting a future quantity and therefore we do not know the new prior. It is therefore not possible to apply this correction directly.
We develop the algorithm shown in [22] to estimate the new prior from the results of the model on the test set. The performance of our model are not good enough to obtain an enough accurate esteem of the new prior. The possibility of following this path to enhance the results of the models is therefore abandoned at this stage.