Università degli Studi di Padova

DIPARTIMENTO DI INGEGNERIA DELL'INFORMAZIONE
Corso di Laurea Magistrale in Ingegneria Informatica

Machine learning techniques for customer churn prediction in banking environments

Relatori
Prof. Andrea Pietracaprina

Laureando
Valentino Avon
Matricola 1104319

Prof. Geppino Pucci

Correlatori
Ing. Massimo Ferrari
Dott. Riccardo Panizzolo
(everis Italia S.p.a.)

Anno Accademico 2015-2016

Contents

Chapter 1

Introduction

In an era of mature markets and intensive competitive pressure, it is fundamental for companies to manage relationships with their customers, in order to increase their revenues. In business economics, this concept is known as the "Customer Relationship Management" (CRM), which is a business strategy that aims at ensuring customers' satisfaction. The companies which successfully apply CRM to their business nearly always improve their retention power, that is, the probability that a customer will not leave. In fact, a high retention power avoids useless waste of money, since acquiring new customers can cost five times more than satisfying and retaining existing customers [14]. This led companies to put a lot of effort in understanding and analyzing their customers' behavior, in order to identify with adequate advance which clients will leave. In particular, these customers are subjected to several rectifying actions (e.g. promotions or gifts) for their retention.

The phenomenon related to the customers abandonment is commonly called *customer churn*, while the churners identification process is usually

called *customer churn prediction*. In this thesis we focus on the first phase only, that is, the correct identification of the customers who will leave the company (namely the "churners").

In this project, we studied the customer churn phenomenon in collaboration with Everis Italia, which is a consulting company located in Milano. More precisely, all the practical work has been done in the context of a curricular internship. In particular, we studied the behavior of about 730000 customers in banking environments, aiming to prove to the banks that predicting customer churn through the use of machine learning techniques is feasible, that is, identifying customers who will leave with quite good precision, avoiding unnecessary costs. Moreover, this thesis seeks to convince banks that a data driven product is valuable and competitive with the traditional ones, which do not use machine learning techniques.

Technically speaking, we chose to model the churn prediction problem as a standard binary classification task, labelling each customer as "churner" or "non-churner".

Since the bank database contains raw data, we put a lot of effort in creating a standard dataset, which can be given as input to common machine learning algorithms. In fact, the database contains many records per customer, in order to track the customers' operations and movements over time. A standard classification dataset, instead, describes each customer with a unique record, which summarizes his/her characteristics with a set of features. For this reason, we aggregated the data related to each customer through some summary functions, such as the mean or the sum, in order to compress all his/her records into a single one. In particular, we computed

these functions on different periods of time, in order to keep track of customers' movements over time, yet maintaining a single record per customer.

An important byproduct of our study is that we formally defined who is a "churner", accordingly to the bank needs. In fact, several definitions can be applied to label a customer as "churner", such as a customer who closes all the bank accounts or a customer who has a very low balance and does not make transactions for a long time. In particular, we labelled as churner at a given calendar month a customer who actually leaves during the subsequent month, in order to predict enough in advance which customers will leave. In this way banks have the chance to apply some rectifying actions useful for retaining their customers.

Furthermore, we presented a non-standard validation approach, which provides a closer estimate to the real business performances of a classifier. In particular, we subdivided the available data temporally, in order to evaluate the performances of a classifier on future data. This validation procedure is more penalizing than the standard one, which consists in randomly partitioning the data among observations. In fact, the behavior of the customers who leave changes over time, hence a model evaluated with the standard validation approach results in optimistic estimated performances, because it is evaluated on churners with behavioral characteristics similar to the ones contained in the training set. For this reason, the predictivity results obtained in this project appear to be much lower than those commonly reached in the literature. In fact, to the best of our knowledge, no work in the open literature has evaluated the classifiers with a temporal approach, thus resulting in optimistic performances which are severely misleading for the business.

In any case, domain knowledge ensures us that our results are considered standard in banking environments.

We predicted the customer churn event using different version of C5.0, an algorithm which implements the decision tree model. In particular, C5.0 was designed to work with large datasets and it was optimized to minimize the training time. More precisely, we evaluated the C5.0 on different preprocessed training sets, which were created by employing the undersampling technique (used to balance the number of records belonging to the positive and the negative class) and the outlier removal (to smooth out noise in the data). We tested an easy ensemble of C5.0 also, which is basically a classifier built through the use of a peculiar technique that, in some cases, allows to enhance the overall predictive accuracy. In particular, the easy ensemble of C5.0 resulted the best model of all tried in this thesis.

The obtained results confirm that machine learning techniques can be effectively applied to predict customer churn in banking environments, that is, succeeding in forecasting which customers will leave in the near future with high precision (i.e. avoiding false positive costs). Hence, more in-depth studies may be worth pursuing.

The remainder of the thesis is structured as follows. In the first chapter, we present a high-level review of classification, highlighting some general machine learning techniques, which have been successfully applied in the literature to solve churn prediction problems. In the second chapter, instead, we show a review of several ad hoc algorithms and techniques proposed in the literature, which were developed specifically for predicting customer churn. In the third chapter, we describe the practical problem that we faced, showing

the preprocessing actions used to generate a standard and clean dataset from the raw data. In particular, we show a non-trivial way to label the records, accordingly to the bank needs. Moreover, we present the aforementioned non-standard dataset partitioning (i.e. the generation of the training set and the test set), which takes into account the time dependencies inherent in customer churn, in order to avoid unrealistic model performance estimations. Finally, in the fourth chapter we present the tested machine learning techniques, the final implemented solution and the achieved results. The thesis ends with a chapter discussing some concluding remarks highlighting the final model which we chose to achieve our purposes, and providing directions for future work which may be useful to improve the predictive accuracy.

Chapter 2

Classification Review

Data mining is a very useful tool for many different tasks. In this chapter we define the classification task, which will be used to model the churn prediction problem, and some common techniques to evaluate it. Classification is a predictive task, which means that it is used to model prediction problems and to make forecasts. In particular, the classification assumes that the event we want to predict is represented by a finite and discrete domain of possible outcome values (i.e. classes). If this is not the case, then the problem becomes a regression task.

Definition 1 (Classification Task) *Given a dataset* $T = \{t_1, t_2, ..., t_N\}$ *of* N *records, where each* $t_i \in T$ *consists of* m *attributes* $A_1, ..., A_m$ *with domains* $D_1, ..., D_m$ *(i.e.* $t(A_j) \in D_j \ \forall j = 1...m$*), and a a class attribute* C*, such that* $t_i(C) \in \Gamma$ *(the finite and discrete domain of classes), the classification task is to build a function* $f : D_1 \times D_2 \times ... \times D_m \to \Gamma$*, which is called classification model or classifier.*

If Γ is a continue domain, the problem is called *Regression*. t_i is usually

called record, example or instance. A_j is called attribute, feature or predictor. C is called class, target attribute or category.

2.1 Attribute Types

Attributes are pieces of information which are related to the class to predict. They can assume various forms, but generally they are categorized in four types. The attribute type depends on different properties, as explained in Table 2.1.

	$= \neq$	$< >$	$+/-$	$\times/\div$	Examples
Nominal	yes				Sex, Marital Status
Ordinal	yes	yes			Price (low, medium, high)
Interval	yes	yes	yes		Date, Temperature in Celsius
Ratio	yes	yes	yes	yes	Lengths, Temperature in Kelvin

Table 2.1: Attribute Properties

The values of a nominal attribute are just different names, i.e., nominal attributes provide only enough information to distinguish one object from another ($= \neq$). The values of an ordinal attribute provide enough information to order objects ($< >$). For interval features, the differences between values are meaningful, i.e., a unit of a measurement exists ($+/-$). For ratio attributes, both differences and ratios are meaningful ($\times/\div$).

Nominal and Ordinal attributes are called Categoric or Qualitative, while Interval and Ratio attributes are called Numeric or Quantitative. This categorization is explanatory: the management of data heavily depends on the attribute type.

2.2 Training, Validation and Test phases

Given a set of historical data, it is possible to build a statistical model on it, using a specific algorithm. This phase is called the "training phase", which trains a classifier to fit the data (i.e. the training set). In this phase, models usually optimize a loss function, which empirically represents the model incapability to fit the training set. Although a good score on the training set is desirable, it does not give information about the model predictive capability. In fact, we do not know if the model has learned useful patterns or just noise. A model that gets low error on the training set and fails to make good predictions is said to be in *overfitting*. Overfitting is a serious problem: a model which seems very accurate but in fact fails when new data comes to evaluation. There are lots of reasons that cause overfitting, such as noisy data or small training set size. There are several approaches to recognize if a model overfits. The common procedure evaluates model predictions on new data, which were not used in the training phase. If the error made on new data is comparable to the training error, then the model has successfully learned some non random patterns. Otherwise, if the error made on new data drastically diverges from the training error, the model has just memorized records and fails to generalize. Typically, the training error is always smaller than the error made on new data: the model is trained on the training set, hence it was optimized on these data. All of this highlights the importance of such procedure, which allows to estimate real model performance. This phase is called the *validation phase*. Another fundamental purpose of this phase is the parameter tuning. The parameter tuning consists in setting the model

parameters properly in order to minimize the error. The error estimation should not be done using the training set, because this would give a biased feedback. Instead, the error estimation should be done using a validation set to get a reliable estimate of model performance.

Typical validation approaches are the *Holdout method* and the *k-Fold Cross Validation*. The first method randomly splits the initial data in a training and in a validation set. The training set is then used to build a model, which is then evaluated on the validation set. The second method, instead, partitions the initial data in k disjoint sets (i.e. the folds). k-1 folds are then used as the training set and the remainder one as the validation set. Iterating this procedure k times, each fold is used exactly once for validation and k-1 times for training the model. The model performance is then estimated with the mean of the k errors.

If the validation set is used to optimize the model parameters or to choose the best classifier for the problem, the estimated error will be optimistic. Therefore, another set of new data should be available to estimate the real model performance (i.e. the test set). This last phase is called the *test phase*. The test set should not be used to make choices or optimizations, otherwise the estimated error would still be optimistic.

It is important to notice that overfitting is not the only cause of bad predictions. There exists another undesirable phenomenon, which is called *underfitting*. Underfitting explains why a classifier gets a high error both on training and validation sets. This occurs when a model can not capture the underlying trend of the data, because it is too simple to fit the data properly. In this case, it is necessary to add new features or to use different classifiers,

in order to avoid underfitting.

2.3 Evaluation Metrics

In the previous section, we showed that a model affected by overfitting or underfitting makes "poor" predictions. To specify what is meant by poor predictions, it is necessary to introduce some formal metrics that measure the performance of a classifier. To optimize a classifier, it is important to specify at the beginning which metric will be used, in order to fix the optimization objective for the entire analysis. Sometimes the metric is given by design specifications, sometimes not. If the metric is not given, data analysts have to choose the proper metric according to the problem.

2.3.1 Accuracy

Accuracy is one of the most common metrics used in data science, because of its intuitiveness and simplicity. Accuracy simply measures how many records the classifier predicts correctly. The formula is very straightforward and it is also easy applicable to multi class problems.

$$Accuracy = \frac{number\ of\ correct\ predictions}{total\ number\ of\ records\ to\ be\ predicted}$$

This formula highlights that accuracy attributes the same relevance to each class. If some outcomes are more relevant than others, that is, we are more interested in predicting correctly some classes rather than others, then accuracy does not give a reliable metric of model's performance.

2.3.2 Area Under the Curve

The Area Under the Curve (AUC) is another popular metric in machine learning and data mining. It is only applicable for binary classification tasks (i.e. the outcome has only two possible values). The two classes are usually called positive and negative classes in this context. In this case, it is possible to define some quantities based on the predicted class versus the real one, as shown in Table 2.2. In particular, we define:

$$True\ Positive\ Rate = \frac{TP}{TP+FN}$$
$$False\ Positive\ Rate = \frac{FP}{FP+TN}$$

The true positive rate measures the fraction of positive records correctly classified by the model as positive ones. Instead, the false positive rate measures the fraction of negative records misclassified as positive ones.

	Positive (Actual)	**Negative (Actual)**
Positive (Predicted)	True Positive (TP)	False Positive (FP)
Negative (Predicted)	False Negative (FN)	True Negative (TN)

Table 2.2: Binary Confusion Matrix

The AUC measures the area under the Receiver Operating Characteristics (ROC) curve, which is a curve created by plotting the true positive rate versus the false positive rate at various threshold settings. Intuitively, the

AUC indicates how well a classifier is able to discriminate the two classes. The AUC plays an important role in imbalanced datasets, which means that the two classes are not equally present. For example, an imbalanced dataset contains the 95% of negative records and the 5% of positive ones. This is a well known problem in machine learning and it is explained in subsection 2.4.1. The AUC is useful in this case because allows to know how well the model understands the positive class, which is difficult to learn given its scarcity. Contrariwise, accuracy gives poor feedback of model performance: because of the scarcity of the positive class, a classifier can get a very high accuracy simply ignoring the positive class (i.e. classifying all records as negative).

2.3.3 F-measure

Another important evaluation metric is the *F-measure*. The definition of F-measure relies on other two metrics: *recall* and *precision*. Recall is the true positive rate. Precision, instead, is calculated according to Table 2.2 as:

$$Precision = \frac{TP}{TP+FP}$$

A high precision means that all the predicted positive records are very likely to be positive, that is, the model makes very few errors on the positive class. Usually a model with a high precision is able to catch only a little portion of positive records, because the error must be limited. A high recall, instead, means that the model is capable to classify correctly a lot of positive records, but it probably makes some errors in order to catch all the possible positive records. These considerations suggest that precision and recall are inversely

related: a high precision implies a low recall and a high recall implies a low precision. Since it is desirable to have good precision and recall, they can be merged in a single metric to make clearer and easier the process of optimization. In fact, maximizing a unique function is more intuitive rather than having two or more optimization objectives. This metric is the F-measure, which is the harmonic average of precision and recall. Specifically, let be $r = recall$ and $p = precision$. Then,

$$\text{F-measure} = \frac{2rp}{r+p}$$

Notice that F-measure is high if and only if both precision and recall are high. As well as the AUC, this metric is useful in evaluating models that deal with imbalanced datasets.

2.4 Common Problems in Classification

In this section we explain two typical problems that often occurs in data mining and some solutions to cope with them.

2.4.1 Class Imbalance

Class imbalance is a common problem in data mining. In the literature, it is usually defined in a binary classification context to make the issue clearer. Lets consider a binary classification task as explained in subsection 2.3.2. The class imbalance problem occurs if the dataset used for the analysis is unbalanced, which means that the number of negative records is much higher than the number of positive records. This disproportion leads classifiers to

ignore the rare class, that is, classifying all the records as negative, because this would imply a high accuracy. This problem is much more relevant if the rare class is more important than the negative one. In fact, the negative class has usually poor interest in being predicted, while the rare class often represents a very significant event. This means that the cost of misclassifying a positive record is higher than the cost of other errors. Moreover, the positive class is more prone overfitting given its scarcity. All of this makes learning from imbalanced datasets challenging. A lot of solutions have been proposed in the literature, but they can be summarized in three important groups:

- Undersampling
- Oversampling
- Synthetic Minority Over-sampling Technique (SMOTE)

Undersampling

The undersampling technique consists in randomly removing some negative records from the dataset, in order to balance the classes proportion. This allows to reduce the dataset size and consequently the computational time, but it may lead to an unrepresentative dataset, because a lot of information is thrown away. To decrease the loss of information, it is possible to create N different undersampled datasets, in order to cover all the initial data, on which to build N models. Then the predictions of the N models are merged with some chosen criteria.

Oversampling

The oversampling technique addresses the class imbalance problem by replicating the positive records, in order to give them more importance. Adding records leads to a bigger dataset, hence the computational time increases. Moreover, if the dataset has some outliers, then the oversampling will amplify their effect.

SMOTE

SMOTE is a more sophisticated technique than oversampling and undersampling. In particular, SMOTE synthesizes artificial positive records with a linear combination of some real positive ones. Moreover, SMOTE also applies undersampling to the negative class to balance the proportion without generating too many artificial records. In practice, SMOTE is very powerful and decreases the chance that a model overfits the rare class.

2.4.2 Curse of Dimensionality

The curse of dimensionality is another big problem in data mining. As we increase the number of features to describe the phenomenon of interest, the number of records needed to fill the feature space grows exponentially. This means that, to achieve good predictions with a high dimensional dataset, we need plenty of data. Moreover, the concept of euclidean distance vanishes in high dimensional space, because the instances tend to be very sparse. It is also more difficult to make hypothesis and exploratory data analysis when the number of features is too high. Although it is very common to have a lot

of features, sometimes only a subset of them is relevant, thus it is possible to reduce the feature space in order to improve the overall performance. This idea is the basic concept that motivates the development of feature selection algorithms, which try to avoid the curse of dimensionality while also removing noise from data. These algorithms are heuristics, because it can be easily shown that optimal feature selection requires exponential computational time.

Random Feature Selection

One of the simplest algorithm for selecting the most important features is the *Random Feature Selection* technique. In particular, this procedure consists in randomly selecting a subset of features, which is then utilized to train a classifier. The model just trained is evaluated on the validation set, in order to estimate its performance and, thus, the features predictive power. This procedure is iterated a number of times, with different random choices, in order this procedure to select the most promising subset of attributes.

Once a given subset is chosen, it is also possible to "lock" the selected variables and incrementally add an additional random feature, trying to improve the results just obtained.

2.5 Common Models

In previous sections we have described the classification task and its typical problems but have not shown how to build a classifier yet. This section defines some common classification techniques that will be used in this thesis.

2.5.1 Decision Trees

Decision trees are one of the most popular learning tools in data mining. They are very popular because they generate an easy-to-interpret model, which reveals the most relevant features. The model generated is a tree structure in which each internal node represents a "test" on a particular feature. The test outcomes split records in subsets, which create new nodes in the tree structure. Once a particular criteria is reached, a node stops splitting itself and becomes a leaf. A leaf is often labeled with the majority class, that is, the most frequent class in the leaf. Each path from the root to a leaf describes a rule. A record which satisfies all the conditions of one rule is labelled with the relative leaf label. This learning algorithm is powerful, because it can catch non-linear relationships between features, maintaining representation simplicity. A popular algorithm which implements the decision tree model is the C4.5 software. In particular, it exists also an enhanced version, called C5.0. The C5.0 classifier features a more efficient strategy for producing decision trees than the C4.5 algorithm, both with respect to memory and to computational costs [8]. Moreover, it is optimized to treat with big amount of data and with missing values. Unfortunately, a decision tree classifier is unstable: a small changes in the data can heavily modify the tree structure and the relationships generated, hence the computational time can vary a lot and sometimes this could be inefficient. The instability makes decision trees prone to overfitting, although there exist some techniques to avoid it. In particular, *random forest* leverages on this very instability to create a more stable and accurate model, as explained in subsection 2.5.4.

2.5.2 Rule Based Classifiers

A rule based model is a classifier which represents the relationships between features and classes through a set of If-Then rules. More precisely, the model is $R = (r_1 \vee r_2 \vee ... \vee r_k)$, where each r_i is a rule. A rule r_i is made by a set of requirements $condition_i$ and a class y_i, graphically represented by $r_i : (condition_i) \implies y_i$. The requirements are composed by some logical comparisons, combined by an AND logical operator. In particular, $condition_i = (A_1 op_2 V_1) \wedge (A_2 op_2 V_2) \wedge ... \wedge (A_l op_l V_l)$, where A_i is a feature value, V_i is a constant and op_i is an operator, such as "$<$" or "$=$".

If a record correctly fits all the requirements of a rule, then we say that the record triggers the rule. Instances could trigger more than one rule at a time, hence it is necessary to give a priority to the rules, placing them in hierarchical order. Typically, the first rules are more relevant than the latter ones. The rules order is significant: once a record triggers a rule, it is labelled with the related outcome. Therefore, the first rule triggered in the hierarchical order is the one that will label the record.

The procedure used to train a rule based classifier depends on the implementation. An effective learning algorithm is the Repeated Incremental Pruning to Produce Error Reduction (RIPPER).

RIPPER implements the above interface as follows:

- the hierarchical order is established by the class frequencies, i.e., rules which concern rare classes are the most relevant ones;
- RIPPER learns one rule using a greedy approach. Iteratively, it creates step by step the requirements set, adding each time the most relevant

condition, which maximizes an optimization function (i.e the foil information gain). Since long rules do not make sense and lead to overfitting, RIPPER is also designed to prune rules;

- the algorithm stops learning the rules for one specific class in two different cases:
  - each training record of that class triggers at least one rule, or
  - the rule set size, which represent that class, is too big.

2.5.3 Logistic Regression

The logistic regression is a classification model and it was developed in order to extend the linear regression model which is, instead, a regression model (i.e. the outcome is continuous). Although linear regression can be adapted for the classification task, it gives poor results. Therefore, researchers have developed the logistic regression model to study the relationships between a binary outcome (0 or 1) and features. The intuition behind logistic regression is pretty simple. Since we need a binary outcome, we could do the following:

- Map the linear regression predictions in $[0, 1]$
- Interpret the new result as the probability of having 1 as outcome
- Predict 1 if the probability is greater than a chosen threshold, otherwise predict 0

This makes logistic regression a powerful classifier, because it allows to specify the algorithm precision through the probability threshold. In particular, setting a high threshold leads to predict 1 only if we are very confident.

Conversely, a low threshold decreases the precision, but increases the recall. Typically, if the problem has equally relevant classes, the threshold is set to 0.5. Usually, the function used to map linear regression predictions in $[0, 1]$ is the logistic function (also called the sigmoid). This function was chosen mainly because its peculiar characteristics, which fit the problem requirements well. The sigmoid function is the following:

$$Sigmoid(z) = \frac{1}{1+e^{-z}}$$

The sigmoid transformation changes the model representation, therefore it is necessary to define a particular loss function in order to train the logistic regression. In particular, the loss function returns a small error if the sigmoid shows the probability of being 1 for an instance, coherently with the real record class. Otherwise it returns a high error. For example, if the real outcome is 0, the loss function returns a high error as the sigmoid tends to 1. The logistic regression is widely used in data mining, because it is easy to use and it can catch non-linear relationships in the data, thanks to the sigmoid transformation.

2.5.4 Random Forest

Random forest is an ensemble learning algorithm. An ensemble is a set of different classifiers merged together to make a more powerful model. In particular, random forest is based on the bagging technique, which is a statistical method to create some different training sets starting from a single one. Random forest builds on each training set a decision tree, using just a subset of random features for each classifier. Since decision trees are very

unstable, building them on different training sets with random features will lead to very diverse classifiers. This allows to lower the correlation among the models, hence increasing the overall performance. In fact, to classify a record, random forest merges all the decision trees with a voting system: each tree votes a class for the record and then random forest chooses the most voted one as the final outcome. The voting system averages decision trees predictions, which are affected by high variance, therefore the results reflect in an increased accuracy, even with large sets of data. This also means that random forest is less prone to overfitting than a single decision tree, because it averages the tree predictions, leading to more robust results. Moreover, it has only few parameters to set up, which is always desirable. The main disadvantage of this approach is the computational time, which increases proportionally with the number of trees.

2.5.5 Boosting

Boosting is a machine learning ensemble meta-algorithm (i.e. usable with all classifiers), which usually helps to improve the performances of weak models. In particular, boosting sequentially trains many classifiers, taking into account those errors that each model made at every step. This means that, at each iteration, boosting trains a new classifier, which tries to fix the errors made by the previous trained model. In this way, boosting usually improves the overall performances, fitting better and better the training set. In particular, this technique initially assigns equal weight to every record of the training set. Then, a classifier is trained and the weights are recomputed,

penalizing those records which are misclassified by the model. Thereafter, a new classifier is trained taking into account the modified weights. Notice that too many iterations of boosting can lead to overfitting, resulting in degraded performances. For this reason, the number of iterations have to be tuned during the validation phase.

Sometimes, simple classifiers cannot solve hard learning problems, because of their excessive simplicity. Boosting addresses this issue training iteratively many weak models to create a strong classifier, which usually enhances the overall predictive accuracy.

A popular boosting algorithm is *Adaboost*, which sequentially trains simple decision trees (called Decision Stumps) composed of just one split [31].

Chapter 3

Churn Prediction Review

Introduction

In this chapter we define the concept of customer churn and the main issues related to it (Sections 3.1 and 3.2). In particular, we show a data mining based approach to cope with this problem. All the different phases of this approach are shown, presenting some literature results. In Section 3.3 we highlight the most important aspects involved in collecting data for the churn prediction problem, while in Section 3.4 we show how to build a proper dataset, presenting three frameworks proposed by Lee et al. [38]. In Section 3.5 we show preprocessing actions typically taken to improve predictive accuracy, such as feature selection, missing value treatment and outlier detection. Finally, in Section 3.6 we show some ad hoc classifiers, which have been developed to manage customer churn. In particular, we present: Ant-Miner+, Active Learning Based Approach (ALBA), Fuzzy C-Means, Hybrid Neural Networks, Hierarchical Multiple Kernel Support Vector Machine and

Improved Balanced Random Forests.

3.1 Churn Prediction

The customer churn, also known as customer attrition, refers to the phenomenon whereby a customer leaves a service provider. Typically, the customer churn is calculated as a relative number in percentage (i.e. the *churn rate*). There are several ways to calculate the churn rate. It is usually expressed as follows:

- Fix a conventional period of time as a month or a year;
- Count the number of customers lost in this period;
- Divide this quantity by the number of customers that the firm had at the beginning of this period.

New costumers acquired during the selected period of time are not considered. Indeed, the churn rate estimation should not be altered by the acquisition of new customers in the same period of time. Conversely, losses of new customers in this period of time may be considered or not, depending on whether the churn rate should counts all the losses during the period. If the losses of new customers are included, then the churn rate measures the total number of customers who have left the company. Otherwise, it measures how many of the initial customers have left. This is a firm's decision.

Some studies confirmed that acquiring new customers can cost five times more than satisfying and retaining existing customers [14]. As a matter of

fact, there are a lot of benefits that encourage the tracking of the customer churn rate, for example:

- Marketing costs to acquire new customers are high. Therefore, it is important to retain customers so that the initial investment is not wasted;
- It allows to calculate customer lifetime value;
- It has a direct impact on the ability to expand the company;
- It allows to identify whether the current actions made by the firm are improving the customer churn or having a negative impact.

All these considerations are typically included in the concept of customer relationship management (CRM), which is a business strategy that modifies the processes management of a company. The proper use of CRM allows a company to improve its revenues, ensuring the customers' satisfaction (i.e. improving customer retention)[7]. It establishes a new approach to the market, placing the business focus on the customer rather than the product. The companies need the CRM to manage current and potential clients with actions and strategies aligned with the customer needs and expectations. It is based on communication, process-integration, people and strategies. It is subdivided in three areas [6], as shown in Figure 3.1:

- Collaborative, which establishes customized relationships with customers by the several existing distribution channels (e.g. e-mail, telephone, website);

- Operational, which refers to services that allow an organization to take care of their customers. It provides support for various business processes, which can include sales, marketing and service. Contact, call centres, data aggregation systems and web sites are a few examples of operational CRM;

- Analytical, which consists on data collection and data analysis allowing the management of the knowledge supporting future decisions.

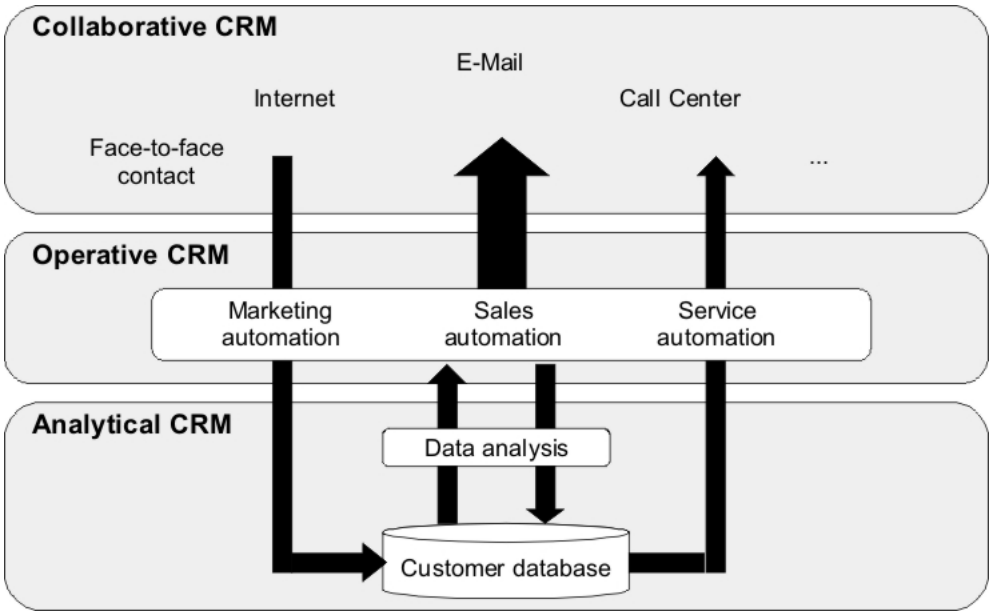

Figure 3.1: CRM sections

The objectives of CRM are manifold [25], but they can be summarized in the following five points:

- Know current and potential customers;

- Communicate with the customers and potential consumers;

- Attract new customers;
- Improve customers relationships;
- *Maintain the customers relationship in a long time term (Customer Churn Management).*

Therefore, the churn management is an important part of CRM. It manages the most relevant aspects that may change the customers' behavior, such as price, service quality, organization's reputation, effective advertising competition and distance in terms of reachability. To handle these factors, a company could use two different approaches: reactive or proactive [9]. The first one is easier and it makes use of responses to a service cancellation request made by the customer, offering promotions or something similar. The second approach, in turn, can lead to two different solutions: the firm can use an untargeted strategy offering promotions to all the customers (incurring more costs) or it can analyse customer data over time to make *churn predictions* (i.e. predict in advance if a customer will leave or not). Knowing in advance when a customer will leave gives much more power to the company to lower customer churn than the other cited methods. In fact, using predictions, the firm has enough time to build a specific campaign, in order to change the customers behavior. This can be done following three important steps in a cyclic manner [6]:

1. Churn prediction
2. Identification of the main causes of defection and related key service issues

3. Development of corrective actions to improve retention

The easiest way to make churn predictions is to observe customers' behavior and to create, with the help of experience, some rules that classify a customer as churner. For example, a bank could label as churner a user that has not made transactions for a long time and that has a low account balance. However, all these rules are created without a scientific method, using only experience and intuition, so the results may be below the expectations. A powerful method is required to make forecasts more reliable than those based just on experience. An effective alternative is Data Mining, which is an automatic discovery process of interesting information applied to large data stored in appropriate repositories as databases, data warehouses or files. One task is to analyse historical data in order to create a mathematical model (which conceptually represents the real world under investigation) and, later, to make predictions on recent data based on this model. Therefore, it perfectly fits our objective of understanding customers data to make churn prediction. The generation of insight is however not only due to Data Mining, but it is a result of a more general scheme, called *Knowledge Discovery in Databases* (KDD) process (Figure 3.2).

The firm data are usually structured in large data warehouse (a central repository of integrated data, periodically updated, used to store historical data in order to support decisional activities). It is not feasible to apply Data Mining techniques to such a data warehouse, because of noise, missing values and irrelevant information. Moreover, the typically large size of the data may result in high computation costs. To fix this, it is necessary to select a subset of the data to work with and to preprocess it (e.g. missing

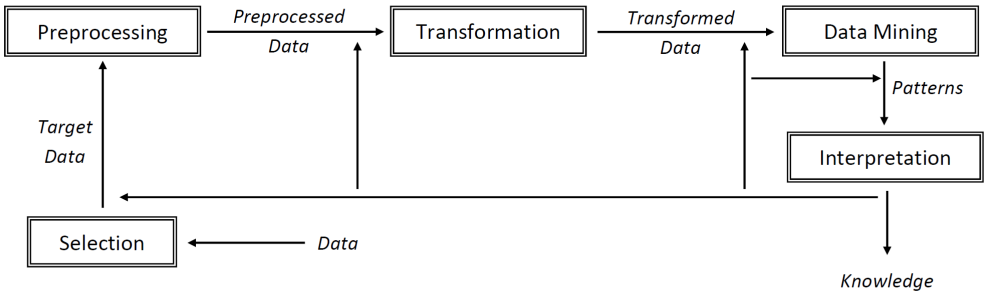

Figure 3.2: Knowledge Discovery in Databases scheme

value treatment, values normalization). These actions are represented by the first and the second phase of KDD: Selection and Preprocessing. It is also possible to transform the data (Transformation phase) in order to facilitate the task of mining data, for example changing the data format. Notice that the operation does not end after the Data Mining phase, but it continues with an Interpretation phase. This phase allows to discern the outcome obtained from the previous step. If some insight is discovered, then the process stops, otherwise it returns to a previous phase in order to improve the discovery process. This suggests that it is not feasible to mine insight if data are not prepared properly, making the entire KDD process a really significant scheme to follow [32].

Data Mining can be useful for companies, but it is much harder to implement than other empirical methods. It requires lots of knowledge and skills and data must be managed in a proper way. If Data Mining techniques are applied correctly, the firm may learn something over the prediction itself, such as the main features that characterize a customer who leaves the service.

In this thesis, we focus on *churn prediction in a banking context* using Data Mining techniques. The aim is to predict if a customer will cease his or her relationship with a bank. Typically, the decision to leave a bank is taken by a customer over a long period of time, during which several implications of the leave are carefully taken into consideration. This "slow" dynamics allows the analysis to use less stringent time constraints, unlike other fields, such as telecommunications, where customers typically switch from one operator to another in a very short period of time, thus making forecasting a particularly difficult task. Also, in the banking context, a slight lowering of the customer churn rate yields a great substantial lowering of the costs and a more sustainable CRM. For this reason, any minimal improvement is important.

3.2 Problem Definition

The previous section clarifies why customer churn predictions are essential to develop an efficient CRM. To apply data mining techniques, the problem of forecasting customer churn has to be formalized first, in order to use the proper tool. The high-level goal is pretty clear: we want to know if a bank customer will churn or not in a certain period of time. Hence, the outcome can assume only two values: churn or non-churn. This suggests that the problem can be prototyped as a classification task (Chapter 2), because the outcome domain is finite. In particular, since the outcome has only two possible values, the churn prediction problem is a binary classification task.

The decision of how long in advance we want to know if a customer will

churn or not, depends on the bank needs. In general, small time windows allow to make more accurate predictions, but they also reduce the bank reaction time for taking countermeasures. On the other hand, although large time windows increase the time allowed for the bank reaction, they could easily lead to wrong results given the high time distance. All of this suggests that it is necessary to find a good trade off between predictions accuracy and the allowed reaction time. Estimating the temporal distance between the instant when a customer start thinking about leaving the bank and when he or she effectively churns, could help finding this trade off. The procedure to set up the time window is explained in Subsection 3.4.3.

The concept of time is critical in churn prediction: it is necessary to model it, otherwise the results will be poor. This additional component diverts from the typical classification task, which is defined as static per se. Therefore, some tricks are often used in the literature to represent time in a static way.

As said in Chapter 2, to build a model and make predictions we need a dataset, which represents a sample of the population we want to analyse (i.e. the customers). The simplest choice is representing each customer with one record, whose columns (the features) are the most important customer's characteristics that affect the churn event. These features are usually selected with the aid of domain knowledge. Feature selection is another critical point in churn prediction, because customers' data are usually large and unaggregated. For example, given the customers' transactional data, we can wonder how to effectively summarize them in some features useful for the modelling phase, such as the total number of transactions, the average amount of money involved in transactions or the last transaction date. All the features which

seem useful to model the problem can be derived. However, we cannot select too many features, to avoid the curse of dimensionality (Subsection 2.4.2).

We can conclude that the main difficulties in making churn predictions are the following:

- find a trade off between accuracy and allowed reaction time;
- model the dynamics: it is important to analyse the customers' behavior over time, in order to model the problem successfully;
- choose a suitable feature set, so to achieve predictive accuracy and to avoid the curse of dimensionality.

3.3 Data Collection

The data collection phase naturally consists in defining how to collect the data according to the problem requirements. This procedure is highly related to the problem we want to solve and it depends on the source of data.

Sometimes, just a bunch of data are available. For example, let us consider the case in which we want to predict if a new medicine is effective or not. We have only a small group on which to test the drug, therefore it is necessary to spend a lot of time and money to increase the sample size. In this case, the main problem is how to collect more data to improve the analysis. On the other hand, sometimes we have plenty of data, therefore the previous issue does not occur. In this case, the main difficulty is to collect a representative sample of the population, avoiding as much as possible the

noise (i.e. outliers). Outliers detection is actually a fundamental phase in the preprocessing step, as explained in Subsection 3.5.1.

In churn prediction problems, available data are usually large, hence we have to select the most significant subset of them. However, the records related to customers who leave are much less than those related to costumers who do not. Hence, a large dataset does not guarantee a sufficient number of churn records for the analysis. As said in the previous section, time is a critical factor in forecasting customer churn. As a matter of fact, time also affects the data collection phase: to predict the customers' behavior well, it is necessary to train classifiers with recent data, in order to catch the most useful patterns. In this way, we can capture the changes of customers' behavior over time, which could be affected by environmental events [10, 40]. In fact, a period of flourishing economy or a crisis can influence the customers' sensitivity to leave the bank, or not. This leads to collect data during a recent and short period of time, in order to use good data and fit actual customers' behavior properly.

However, collecting data for a brief period of time can lead to an unrepresentative dataset. Since customers who churn are much less frequent than those who do not, the churn instances number could be insufficient to make useful predictions. This suggests that the time period has to be chosen according to the bank churn rate, in order to collect enough data.

Another effect due to the variance among class frequencies is the generation of an imbalanced dataset [11] (Subsection 2.4.1). This makes learning the churn class challenging. Moreover, since prediction errors on churners are much more expensive than the others, churn prediction becomes an even

more difficult problem.

Actually, it exists another protocol to collect data for the churn prediction problem: the case-based sampling [22]. This approach includes in the dataset all the churn records available and a random sample of non-churn instances, in order to balance the classes proportion. In this way, the class imbalance does not occur, making the entire analysis easier. The big disadvantage of this method is the implicit renounce of modelling recent customers' behavior, assuming that it does not change for the entire large period of time considered. This approach can be seen as an undersampling of all the available data, in order to tackle class imbalance (Subsection 2.4.1).

3.4 Dataset Structure

Once the data are collected, they have to be organized and transformed in order to define a suitable set of features. This allows to produce a dataset, which will be used to train classifiers. In this section we show how to divide and manage the different customers' attributes. In particular, we highlights three frameworks suggested in the literature, in order to create a proper dataset for predicting customer churn.

3.4.1 Static and Longitudinal Attributes

Generally, customers can be characterized with different types of attributes. Features which do not vary over time are called static. For instance, some customers' static features could be: gender, education level or social status. In particular, education level varies very slowly or it does not vary at all,

therefore it is considered as static. Static attributes are easy to manage. In fact, it is sufficient to collect them and they are almost ready to be used.

Features which depend on time are called longitudinal or dynamic attributes. They describe the customers' behavior changes over time. Therefore, they are the most powerful ingredient to predict customer churn accurately [10]. Dynamic attributes are more difficult to manage than the static ones, because they have to be discretized, in order to represent them in a dataset. This suggests that a sampling frequency has to be chosen. For example, if we want to model customer's visits to the bank, we can count his or her visits each month or every week. The sampling frequency is a fundamental decision: if it is too low the data are not enough to interpolate customers' behavior. On the other hand, if the sampling frequency is too high, a lot of data will be irrelevant, because longitudinal attributes might vary with a lower rate than the sampling frequency. Moreover, having too much data to analyse could increase complexity. Typically, demographics attributes are considered as static, while transactional ones as dynamic [10].

The literature suggests an enhanced way to collect dynamic attributes related to transactions. for the churn prediction problem [23], in order to build a more accurate classifier. In particular, instead of considering a fixed time period in which collecting data, it is suggested to use a dynamic timeline, as shown in Figures 3.3 and 3.4. To explain the problem, we give an example: suppose that a classifier is built using data of 1000 customers, of which 700 are active and 300 are known to have left. Suppose also that three months activities are analysed (e.g. from February 2006 to April 2006). In this case, the time period is fixed and only the activities done in this interval

are analysed. Imagine that a half of the 300 customers have left away in February. This implies that the model will not be fully trained with their data, because only one month's activity is available (it is not possible to have customer's data after he or she churns, obviously). This problem occurs because the time period is fixed. Therefore, the suggested solution is the following:

1. decide the length of the time period to analyse;
2. collect data from the last transaction date of each customer, in order to cover backwards the established period of time.

In this way, we have the guarantee that the model will be trained with proper data.

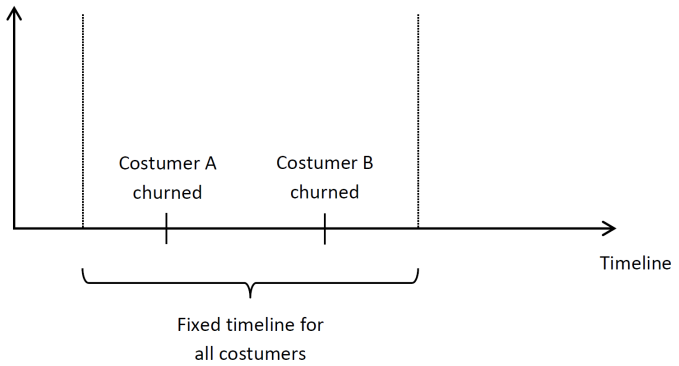

Figure 3.3: Fixed Timeline

3.4.2 Dataset Frameworks

In this section we introduce three dynamic churn prediction frameworks proposed by Özden Gür Ali & Umut Arıtürk [40], which utilize the longitudinal nature of the customer data.

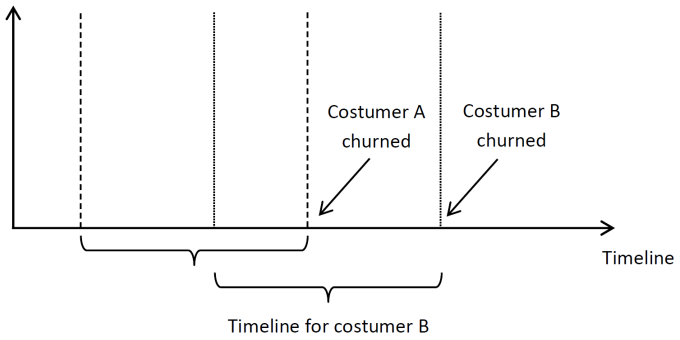

Figure 3.4: Dynamic Timeline

Single Period Training Data (SPTD)

The following is a framework first introduced by Lee et al. [38]. It is defined as the "standard framework for customer churn prediction", because of its simplicity.

The SPTD is a training set, which contains data collected in a single contiguous time period. It represents each customer with one record, whose dynamic features are summarized in a single value with a function, such as the mean or the maximum value. For example, suppose that we collect data related to withdrawal events every month. While analysing data, we see that a customer withdraws 500$ during the first month, 300$ in the second and 900$ over the third. Hence, if we choose the maximum as the summary function, the record created to represent this customer will have 900$ as withdraw feature value.

Typically, the SPTD is built analysing the most recent period available, in order to catch the current customers' behavior, as explained in Subsection 3.3.

Substantially, the SPTD is a normal training set, which models dynamic

attributes summarizing them in a single value. Therefore, one feature is created for each longitudinal feature. The outcome is defined as a binary variable, which assumes the value 1 if a customer leaves in the period considered and 0 otherwise.

Multiple Period Training Data (MPTD)

The MPTD is a simple extension of the SPTD. In fact, as the name suggests, a MPTD is a training set which contains several SPTDs relative to different time periods. It is defined as follows:

$$MPTD = \begin{bmatrix} SPTD_T & \\ SPTD_{T-1} & \\ \vdots & E \\ SPTD_{t1+1} & \\ SPTD_{t1} & \end{bmatrix}$$

T is the most recent period of time, while *t1* is the oldest one. The $SPTD_i$ denotes a SPTD built in the i-th period of time before T. This means that the MPTD represents N different periods of time for each customer, therefore there are N records for each customer. The MPTD labels are given in the same way as the SPTD labels. Modelling different periods of time allows to introduce environmental variables in the dataset (matrix E in figure). In particular, the number of rows in E is equal to the number of periods considered in the analysis, while the number of columns is equal to the number of environmental features previously chosen. These attributes model the environmental factors, which affect customers' behavior (as explained in Section

3.3). The environmental variables assume the same value for all the records in a specific time period. This means that only during various periods of time the environmental predictors may assume different values. Static attributes, instead, vary among customers, but not over time. Longitudinal features vary over time and among customers.

The MPTD contains $(T - t1 + 1)$ times the records of one SPTD. Therefore, this improves the predictive accuracy compared to SPTD, due to a richer set of data [40, 30]. Moreover, the MPTD contains $(T - t1 + 1)$ times more observations than the SPTD, which addresses the absolute rarity problem. However, the potential lack of independence, introduced by multiple observations of the same customer, may bias the parameter estimates and/or artificially increase the significance of the parameters.

SPTD+Lags

Since managing the MPTD could be difficult and the MPTD size can lead to a high training time, an alternative framework has been developed in order to exploit dynamic attributes. The SPTD+lags is an extended version of the SPTD, which is adapted to support more than just one sample for each longitudinal attribute. Instead of describing time vertically (i.e. assigning more than one record to each customer), it represents time horizontally. This means that the SPTD+lags models customers' behavior over different periods of time (as well as the MPTD), but across features. Substantially, the SPTD+lags considers k periods of time in which collect information. Then, the data collected in the different time periods are summarized with a function (as in the SPTD case), creating k features for each dynamic attribute.

The number of time periods to consider is a design variable. Although a large k provides more information about the customers' behavior, it significantly increases the number of features, incurring the curse of dimensionality problem.

3.4.3 Labelling the Dataset

In classification problems, the records labels are often well defined. However, this is not the case for churn prediction. In fact, sometimes customers do not close the relationship with a bank directly, but they simply become inactive. In this case, it is necessary to define a specific criteria in order to distinguish this kind of customers from the active ones. We can regard inactive customers as customers who have left if they do not make transactions or move enough money for too long [29]. In banking sector, a customer is usually labelled as churner if he or she is inactive for at least six months [3].

Another problem that arises in labelling the dataset is how to define precisely which temporal instant has to be predicted. In Section 3.2, we discussed that a bank has to define the reaction time required to respond to customers' behavior. To formally implement this concept into a classifier, it is necessary to label the customers in the training set properly. In fact, the labels have to be chosen according to the event we want to predict, which definitely depends on the desired reaction time. For example, knowing far in advance if a customer will leave means to focus more intensively on forecasting "warning" events, which are small signals of potential churn (e.g. a complaint call to the customer service) [26]. This leads to label as churner

those who are involved in warning events, because of the changed prediction target (i.e. we are interested in predicting warning events, not the churns). Modelling the problem with labels, which represent customers who have effectively left the bank, leads to predict if a customer will leave the bank very soon, which is almost impossible to rectify (the customer has already decided definitively). However, if the bank pretends to know if a customer will leave too much in advance, the results will be poor. This happens because warning events does not necessary imply churn ones, hence a lot of false positive errors will arises (i.e. non-churner considered as churner).

3.5 Preprocessing

Preprocessing is an important phase of the KDD process, which prepares data for the modelling phase. Researches have shown that preprocessing has a significant impact on the predictive accuracy [34]. Actually, data analysts spend more time on preprocessing data than on modelling, highlighting the relevance of this phase. The preprocessing activity has many facets [16], but in the following we explain the most relevant ones, which are usually used in practice in forecasting customer churn.

3.5.1 Outlier Detection

An outlier is a record which significantly differs from typical records. This means that it has at least one feature with an atypical value. Outliers could be related to noisy data, but in some cases they are special records, which diverge from normality. This suggests that the outlier concept assumes dif-

ferent faces depending on the problem. The outlier detection phase concerns the search of anomalous records, which have to be removed if they represent noise.

Typically, in churn prediction there are different types of outliers. Sometimes a customer leaves for external reasons, which are out of bank control. Natural death or moving to a foreign country (or a different region) are examples of events that may result in customer churn [10, 12]. Hence, these records have to be removed from the dataset, because they will introduce noise.

Another type of outliers is represented by customers who open a new account just for a particular purpose and they close it as soon as their goal is achieved. These customers do not give additional useful information for the churn behavior, therefore they have to be removed. Since these customers are indistinguishable from the others, empirical methods are usually applied to remove them from the dataset. In particular, Devi Prasad et al. [23] suggest to ignore customers whose duration is less than six months and customers who have made less that fifty transactions, in order to solve this problem.

Removing outliers usually results in increasing predictive accuracy, because the dataset represents better the common churn patterns.

3.5.2 Missing Values Treatment

The treatment of missing values is a fundamental step in the preprocessing phase. However, some classifiers do not deal with missing data. Therefore it is necessary to apply countermeasures, in order to remove or replace missing

values. A missing value could have at least two meanings: the real value is indeed not known, or a problem has occurred during the data collection phase [24], leading to missing some data.

Since missing data are very common in data mining, many techniques have been developed by researchers. The less sophisticated approach simply removes all the records with at least one missing value, which is fine if the available data are abundant. If this is not the case, it is better to replace missing values imputing them. The most popular method is to replace missing data with the mean/median/mode, in order to leave untouched the feature distribution. In predicting customer churn, both removing [35] and imputing [10, 41] methods are used. Dirk Van den Poel et al. proposed a peculiar technique: the "tree imputation with surrogates" [10], which imputes missing data based on the other features values. This approach uses a decision tree (Section 2.5.1) to estimate the unknown values.

3.5.3 Feature Selection

As explained in Subsection 2.4.2, having a high dimensional dataset leads to a more difficult analysis. Fortunately, plenty of procedures were developed, in order to reduce the dimensionality. Usually, when data are collected, we do not really know which features will be significant and which ones will be irrelevant or noisy. Therefore, we can analyse the features importance in predicting the event of interest just after the data collection phase. This is the task of feature selection, whose aim is to remove irrelevant attributes and noise, in order to improve predictive accuracy. Feature selection is also a

useful procedure to detect the most relevant predictors. Moreover, it favors classifiers interpretability because they manage only the useful features and therefore the model will be simpler.

In the literature, there is not a peculiar procedure used for churn predictions, as far as we know. However, some feature selection techniques are shown to predict telecom customer churn. Since the problem is in common, even though for different fields, we show these results. Adnan Idris et al. [18] propose two main feature selection procedures. The first one is the Principal Component Analysis (PCA). PCA is a quite old statistical procedure [17], whose aim is to remove redundancy from data. The feature space is transformed into a set of linearly uncorrelated attributes, which are called the principal components. These components try to cover the maximum variance of the dataset, minimizing the loss of information. PCA works well if the dataset is redundant, because just a few of principals components are sufficient to fully describe the dataset.

The second procedure proposed is the minimum-Redundancy-Maximum-Relevance (mRMR). mRMR selects a subset of features which have a strong correlation with the outcome, but a low dependency on each other. Therefore, this procedure selects attributes which are mutually far away from each other, maintaining high correlation with class labels, thus resulting into minimum redundancy and maxim relevance. The results show that applying mRMR gives better results of PCA in average.

Anyway, features with too many missing values should be omitted, because they will introduce noise [28, 39].

3.5.4 Overcome Class Imbalance

In customer churn prediction, the number of churners is usually much smaller than the number of non-churners. This makes predicting customer churn more challenging, due to class imbalance (Subsection 2.4.1). For this particular classification task, the two most popular techniques used to endorse class imbalance are undersampling and SMOTE [1, 11] (Subsection 2.4.1). In fact, Verbeke et al. report that oversampling does not result in significant improvement of predictive accuracy, because it causes overfitting. However, undersampling throws away a lot of information. An efficient procedure used to exploit all the available data is the easy ensemble technique [42]. This method independently creates several balanced subsets from the original dataset with the undersampling technique. Then, for each subset it trains a classifier. Finally, it combines all the classifiers with a chosen criteria, such as the majority vote, in order to classify all the records. This approach allows to cover almost all the available data of the majority class, resulting in an increased predictive accuracy. Moreover, the algorithm can be easily parallelized to decrease computational time, because the samples are independent from each other. A weakness of this procedure is the lack of comprehensibility, because it combines a lot of models, which are difficult to interpret as a unique mechanism.

3.6 Models

In this section we report a number of standard classifiers, which were proposed in the literature to predict bank churn events. We also show some

ad hoc classification models, which have been developed to predict customer churn.

3.6.1 Standard Popular Models

In the past decades researchers' focus was set on training the best classifier to model a problem. As a matter of fact, a lot of well known classifiers were applied to predict customer churn. In particular, decision trees and logistic regression were highly used because of their comprehensibility [23, 19, 20, 27]. In fact, knowing which factors affect the customers' behavior is an additional insight which could be extracted from these simple classifiers. Another simple model which has been used is RIPPER [12, 36]. However, comprehensibility is usually sacrificed, in order to achieve a high predictive accuracy with the use of more complex classifiers. In fact, support vector machine (SVM) and random forest were tested, in the attempt to reach better results [37, 15, 41, 39].

In the following subsections we show a list of ad hoc proposed model, which have been studied to obtain additional insight and predictive accuracy.

3.6.2 Comprehensible Models

Comprehensible models are really important in churn prediction, because they give insight on which are the critical factors that affect customers' behavior. Knowing these characteristics allows to improve the CRM, thus preventing future attempts to churn. Unfortunately, comprehensible classifiers are usually very simple, hence their predictive accuracy may be below the

expectations. This issue motivates the development of more complex classifiers, which are still understandable. In the following subsection we show AntMiner+ and Active Learning Based Approach (ALBA), which are two algorithms proposed by Wouter Verbeke et al. [2] to cope with this problem.

Ant-Miner+

Ant-Miner+ is a rule induction technique based on the principles of Ant Colony Optimization (ACO), which is a metaheuristic inspired on the foraging behavior of real ant colonies. A single ant is a simple insect with limited capabilities, guided by straightforward decision rules. However, these simple rules are sufficient for the overall ant colony to find the shortest paths from the nest to the food source.

Ant-Miner+ artificially simulates the behavior of ants, which are used to find the shortest path to reach a solution. More precisely, the algorithm releases "artificial ants" in a directed acyclic graph (DAG), which is a representation of the problem we aim to solve. Each attribute has to be categorical (Section 2.1) to build a proper graph. In particular, every feature is mapped on a set of vertices, where each vertex represents one of the possible attribute values. Each graph edge connects two vertices related to different features, in order to create plausible paths which represent consistent decision rules. Each ant treads the graph and if it finds a solution, then it releases pheromone in the travelled path, in order to guide other ants movements. However, the pheromone evaporates over time, reducing its attractive strength if it is not readily remarked. The pheromone concentration indicates how many ants have chosen that specific trail recently. When an ant reaches a decision

point, it is more likely that it will choose the trail with the higher pheromone concentration. This stochastic process is also guided by an heuristic measure, which is problem dependent.

Ant-Miner+ uses the ACO algorithm to build a classification model. In particular, it is based on the max-min ant system, which is a modified version of the ACO. Through this system, an environment is initially defined for the ants which will walk through it, such as each path corresponds to a classification rule. The ACO procedure drives the ants towards good predictive rules. A peculiar characteristic of this algorithm is that it can accept monotonicity constraint, in order to incorporate domain knowledge within the model, making it more comprehensible. For example, the rule "if Bank Visits $<= 3$ then class = non-churner" is a poor rule, as we would expect that less frequently a customer visits the bank, most probably he will churn. Therefore, a more reasonable rule would be "if Bank Visits > 3 then class = non-churner" [2]. AntMiner+ allows to impose these types of constraints on the rules, in order to implement the domain knowledge in the classifier.

In particular, this can be done during the graph construction phase. In fact, it is possible to remove edges in such a way that the ants cannot tread all the graph paths, forbidding connections between vertices that would violate the domain knowledge constraints. In this way, we can make a more reasonable and comprehensible model, joining data driven patterns and domain knowledge to predict the customer churn better.

The literature shows that this classifier can reach a high accuracy level, maintaining the comprehensibility required [2].

ALBA

ALBA is another comprehensible classifier, which tries to achieve a high predictive accuracy. It trains a SVM in order to catch non-linear relationships in the data. However, the SVM creates a opaque model, which is absolutely incomprehensible. ALBA handles this issue, exploiting the support vectors used in the SVM in order to modify the training set in a certain way. The modified training set is then used to train a rule based classifier (such as RIPPER), in order to extract an accurate, but still comprehensible, rule set [13]. In particular, the training set is modified following the active-learning paradigm, which consists in controlling the learning algorithm over the input data on which it learns, focusing on the input space where the noise is the highest (i.e. the regions near the SVM decision boundary).

The procedure used to modify the training set is the following. First, ALBA trains an SVM to fit the training set and to discover the support vectors. Once the classifier is optimized (choosing the right parameters), it is exploited in order to change the training set labels to clean up data [13]. In particular, we replace the records labels with the predicted outcome of the black box classifier. In this way, all noise concerning class labels is removed from the data. Therefore, an artificial optimal decision boundary is created.

The next step consists in generating additional artificial records close to the support vectors, in order to incorporate the active learning approach. In particular, the distance of the new records is determined by the average distance from the training instances to the support vectors, scaled by a random factor. We label the new instances automatically, forecasting them with the

SVM previously trained.

Finally, a rule induction algorithm is run on the entire modified training set in order to extract a comprehensible rule set. ALBA results in increased predictive accuracy, outperforming the standard SVM classifier [2, 13]. The detailed procedure is shown in Algorithm 1.

Algorithm 1 ALBA

Let be N the training set size and n the number of features
Train SVM on training set
Replace the records labels with the SVM predicted outcome

*** Calculate the average distance $distance_k$
*** from records to support vector, in each dimension k
for k=1 to n **do**
    $distance_k = 0$
    for all support vectors sv_j **do**
        for all record **d** in training set **do**
            $distance_k = distance_k + |d_k - sv_j|$
        end for
    end for
    $distance_k = \frac{distance_k}{\#sv \times N}$
end for

*** Create T extra data instances
for i = 1 to T **do**
    Randomly choose one of the support vectors sv_j
    *** Randomly generate an extra data instance x_i close to sv_j
    for k = 1 to n **do**
        let be rand a random number in $[0, 1]$
        $x_{i,k} = sv(j, k) + [(rand - 0.5) * \frac{distance_k}{2}]$
    end for
    *** Provide a class label y_i using the trained SVM
    $y_i = SVM(x_i)$
end for
Run rule induction algorithm on the modified training set

3.6.3 Clustering Classification

The approach used to make predictions so far is the standard classification procedure, that is, building a model which learns from a labelled dataset, in order to infer a relationship between data and classes. In this section, instead, we explore another approach to learn from data: the clustering procedure. The clustering task is to discover patterns in an unlabelled dataset, that is, analysing features in the absence of a prespecified set of record classes. In particular, the aim is to discover groups of similar records, which are called clusters. Records can then be labelled according to which cluster they belong to. In churn prediction, we want to find two kind of clusters in data, which represents churners and non-churners.

Fuzzy C-Means (FCM)

Classical cluster analysis methods assign each instance to a single cluster. The FCM disrupts this concept, allowing records to belong to many clusters with different "degrees of membership" [3]. The FCM is a modified version of the popular K-Means clustering algorithm, which creates clusters based on the concept of distance between records and cluster centroids. In the FCM, a cluster centroid is represented by the mean of all points, weighted by their degree of membership.

The algorithm is an iterative procedure, which moves the centroids in the space, in order to minimize an optimization function. Every iteration updates the centroids position and gives to the instances a set of coefficients, which indicate the membership degrees for each cluster. The algorithm ends

if the centroids do not move more than a threshold value ϵ.

Džulijana Popović et al. [3] show that FCM outperforms the classical clustering algorithms in forecasting customer churn. In particular, their results prove that FCM is robust to outliers, which is always desirable.

3.6.4 Hybrid Neural Networks

Chih-Fong Tsai et al. [21] proposed a hybrid approach to predict customer churn, that is, a procedure based on two or more data mining techniques. In particular, they studied two different combinations: classification + classification and clustering + classification. They used as classifier an Artificial Neural Networks (ANNs), which is a model that attempts to simulate biological neural systems. In particular, these systems learn by changing the strength of the synaptic connection between neurons upon repeated stimulations by the same impulse [21]. Moreover, they employ a clustering technique based on ANNs called Self Organizing Maps (SOM).

Their strategy is selecting the best data to train a classifier. The first phase of the two combinations essentially selects the data which will be used in the next step.

The first hybrid procedure, that is, classification + classification, trains two standard ANN classifiers. First, a ANN is trained. This model will classify only a subset of the training set correctly. Therefore, the incorrect instances can be regarded as outliers, since the ANN model cannot predict them accurately. Then, the correctly predicted data are used to train another ANN, which will be used to make predictions.

The second hybrid procedure, that is, clustering + classification, makes a clusters analysis to select data and then builds an ANN on them. First, the SOM is run, in order to discover clusters in the training set. After this, the two clusters of SOM, which contain the highest proportion of the churner and non-churner groups respectively, are selected as the clustering result. Then, ANN is trained on these two clusters.

Chih-Fong Tsai et al. [21] evaluate the two different models on real customers' data. The results show that the Hybrid ANN outperforms the standard ANN for predicting customer churn, while the SOM does not improve the ANN performance.

3.6.5 Hierarchical Multiple Kernel Support Vector Machine (H-MK-SVM)

A H-MK-SVM [35] is an SVM-based classifier, which can directly deal with longitudinal attributes. Therefore, it is a useful model in forecasting customer churn, where the dynamic features are the most significant ones. It is an enhanced version of the multiple kernel SVM (MK-SVM), which is a SVM that uses a linear combination of basic kernels to approximate an optimal kernel. The linear combination coefficients, which weigh the kernels' importance, has to be determined in the training phase. More precisely, the MK-SVM is trained in two phases. In the first step the kernels coefficients are fixed and a normal SVM is trained in order to learn the standard parameters. In the second phase, the best kernel coefficients are determined by solving a linear program. In this way, we can classify new records through a

standard SVM, using as kernel the optimized linear combination of kernels.

The H-MK-SVM is an extension of MK-SVM, which has been developed to manage dynamic features directly. In particular, the H-MK-SVM uses two different multiple kernel types: one for the static features and the other for the longitudinal attributes. Both of them are described in the following paragraph.

Let m be the number of static features, M the number of dynamic attributes and T the number of samples belonging to each longitudinal feature. Concerning the static attributes, a simple linear combination of m kernels is used as kernel. For the dynamic ones, instead, a more complex kernel is used. More precisely, this kernel is obtained in two steps. First, for each longitudinal attribute a linear combination of T kernels, which map the T dynamic feature time values, is created. Then, these M combinations are linearly combined in turn to create the final kernel.

All the coefficients of each combination have to be determined, in order to train the H-MK-SVM. This could be done in three different steps. The first phase is the usual one, which trains a normal SVM as in the MK-SVM case. The second phase finds the optimal coefficients for the static and dynamic attributes. Finally, the last step determines the time series coefficients, that is, the values that weigh each point of every dynamic feature.

In practice, the H-MK-SVM outperforms the standard SVM and the MK-SVM in predicting customer churn.

3.6.6 Improved Balanced Random Forests (IBRF)

Random forest is one of the most popular ensemble learning algorithm in data mining. However, standard random forests do not work well on extremly imbalanced dataset, such as a typical customer churn prediction dataset. In the literature, there are two ways to adapt random forest for imbalanced datasets [5]: balanced random forests and weighted random forests. The first, simply preprocesses data, balancing them using the oversampling technique described in Subsection 2.4.1. Then it trains a random forest with the balanced dataset. The second, instead, assigns a weight to each class, which represents the cost of misclassification for that class. Thus, the rare class receives a larger weight, in order to heavily penalize its misclassification in the training phase.

Neither method was proved to be the best, therefore Yaya Xie et al. [39] proposed an algorithm which combines both strategies: the improved balanced random forests. In particular, it creates N different datasets as the standard algorithm (see Subsection 2.5.4), but each of them are generated in a supervised way (i.e. looking at the labels), in order to deal with the class imbalance. For each dataset, a weighted decision tree is trained, with some peculiar weights. To classify a new record, the standard voting procedure is used, that is, each tree votes for a class and the majority vote wins. Yaya Xie et al. [39] show that IBRF outperforms the standard random forests approaches, maintaining scalability and fast training time. For more details on the workings of this technique, one may refer to Yaya Xie et al. [39].

Chapter 4

Practical Problem Description and Data Preprocessing

In this chapter we present the customer churn analysis we made on a real dataset offered by global bank. In particular, we explain the data collection, the data aggregation and the statistical analysis used, following the same logic discussed in Chapter 3. The entire analysis was executed avoiding those optimizations considered not necessary, such as the extraction and computation of complex feature to describe the customers or an efficient outlier analysis, since the project is a *Proof Of Concept* (POC). A POC is a methodology used to demonstrate that some theoretical concepts can be efficiently applied to solve practical problems.

4.1 Introduction

The aim of this thesis is to demonstrate that forecasting customer churn with data mining techniques is feasible. In particular, we want to prove that these techniques are sufficiently effective to enhance the bank retention power, which relies only on traditional approaches (see Section 3.1).

The bank provided us the access to a large dataset for the data analysis, based on scrambled but consistent data. This dataset aggregated several customers' data, therefore we initially analyzed it generating proper SQL queries to make the entire process more efficient and manageable.

The extracted data were subsequently analyzed with R, which is an open source statistical software full of up-to-date data mining packages. R is a powerful tool which allows to perform a complete data analysis and reduce the POC costs thanks to its free license.

4.2 Dataset Structure

The bank has to track customers' data over time, in order to offer them a proper service. This means that each customer is described through many records within the database for tracking his/her operations over time. This implies that the data have to be aggregated in order to be analyzed, leading to additional preprocessing.

In fact, each customer is described by a huge amount of records, which is very complex to manage. For this reason, it is fundamental to decrease the number of records which describes a customer, in order to reduce the infor-

mation size and the computational time required to analyze it. Therefore, it is necessary to find a good compromise between the precision needed to describe customers' behavior and the quantity of data necessary for a proper description.

Hence, the customers' actions were initially aggregated by month to make database operations more manageable.

More precisely, we grouped all the information we needed (i.e. demographic, transactional and balance data) within a single convenient table, which has a composite primary key (that is, each record is uniquely identified by this field) defined by the customer's identifier (ID) and the month considered.

We collected the data related to the period from January 2015 to April 2016, hence each customer is described at most through sixteen records, since the period considered includes sixteen months.

The collected data included about a subset of about 730000 customers, 18300 of which are churners. The annual churn rate is about 2.5%, which means that every month approximately 1500 customers leave the bank.

4.3 Dataset Creation

In this section we present how the final dataset was generated from the raw data, the initial data quality check used to verify the correctness of the collected data and a naïve outlier analysis.

Furthermore, we show how we have aggregated, adapted and labelled the data, in order to create a standard dataset, which can be given as input to

data mining algorithms directly. In particular, we present a non-standard way to separate the data into training set and test set.

4.3.1 Data Quality Check

During the dataset creation phase it is important to avoid accidental and logic errors, otherwise the subsequent analysis will be corrupted by wrong data. Therefore, it is fundamental to verify that the data collected from the database are correct, since errors happen very frequently in a database of such size.

In particular, we verified some common mistakes that are usually present, such as:

- customers who leave the bank, but have opened their first account after the date of leaving;
- categorical variables characterized by different values, but identical meaning, as Male and Man for the Gender;
- transactions import of erroneous sign (e.g. a positive withdraw);
- unknown activation date (i.e. when a customer opens the first account);
- inconsistency between account number and the date of activation or leaving;
- the presence of records with the same primary key (i.e. the same customer ID and month).

It is important to accurately fix the mistakes in this phase, in order to avoid the errors propagation to the next steps. In fact, the errors are increasingly more difficult to detect and fix as the project phases go on.

4.3.2 Dimensionality Reduction

The customers' transactional data are described through about 200 attributes, one per operation type. Since many of them have similar meaning, they can be grouped in order to compress the table dimensions and reduce the complexity of the analysis. However, these attributes are difficult to interpret without any business expertise. Hence we grouped them with the help of domain knowledge to effectively generate meaningful clusters.

The similar attributes of each group were aggregated applying the sum operator, in order to identify the total amount of money moved related to the considered type of operations. This compression allows to reduce by about one order of magnitude the number of transactional variables, thus avoiding the feature overgrowth in the "multi-month" attributes generation phase.

The "multi-month" features are variables synthetically created to track customers' behavior over time (see Subsection 4.3.3 for further details).

4.3.3 Feature Extraction

Due to its nature, customers' behavior is a time varying peculiarity, hence it requires longitudinal features to be tracked over time (Subsection 3.4.1).

In this thesis, we followed a similar approach to the one used by Lee et al. [38] for the *SPTD+Lags* framework described in Subsection 3.4.2.

In particular, each customer is described through one record only, which captures both static and time varying information.

Since the customers' temporal behavior is not tracked with additional records, contrary to the original bank database in which customers' actions are described by several records, it is possible to describe this information with extra attributes. These new features track peculiar customers' characteristics over time, evaluating them during different periods of time.

We consider four different periods of time on which computing dynamic customers' features. More precisely, we considered:

- the last month available;
- the last three months available;
- the last six months available;
- the last twelve months available.

For these periods of time we separately computed the mean, sum, minimum and maximum of each dynamic attribute, causing an overgrowth of the features number. In fact, for each variable we generated sixteen new features, that is, four attributes for every period of time considered. This is the principal reason why we have aggregated the transactional features (Section 4.3.2). In fact this phase would have created too many variables, which are not easily manageable.

Since the granularity of our data is represented by one month, the four considered functions (mean, sum, minimum and maximum) will return the same result if computed on a single month. For this reason it is possible

to maintain only one of these four results, reducing considerably the overall number of features.

We called these new variables *multi-month attributes*. To clarify this procedure, we present the following example. Lets consider a customer who leaves in October 2015 and joined the bank during January 2015. Table 4.1 represents the available data for the considered customer (ID 0001) and highlights in gray the period of time used to compute multi-month attributes on six months. The card balance attribute is the time varying feature considered in the example. This attribute measures the withdrawals and the deposits made by the customer 0001 on a specific card. More precisely, the records describe the total amount of money moved by the customer every month, allowing to track his/her operations over time. Moreover, the "Churn" variable shows if the customer left the bank in the considered month, or not.

ID	**Month**	**Card Balance**	**Churn**
0001	Jan-15	-100	No
0001	Feb-15	-1000	No
0001	Mar-15	2000	No
0001	Apr-15	-150	No
0001	May-15	-150	No
0001	Jun-15	-20	No
0001	Jul-15	100	No
0001	Aug-15	0	No
0001	Sep-15	0	No
0001	Oct-15	-500	Yes

Table 4.1: Multi-month computation example

In this example, the maximum of card balance during the six months amounts to 100€, the minimum is -500€, the mean is -95€, and the sum amounts to -570€.

It is important to notice that, if the required data used to compute the considered functions on a certain number of months are not sufficient, then, as a convention, we approximate it using only the available data.

In this example it is not possible to compute the functions on the twelve months, since the available data concern only ten months.

If a customer does not leave the bank, the multi-month attributes are calculated starting from the available data related to the last reachable month, that is, the last month of the considered period. For example, consider a training set built with the data related to the period from January 2015 to December 2015. In this case, the last reachable month for the non-churners of this period would be December 2015.

We extracted some static features also, such as the customers' age and his/her seniority (i.e. the number of months during which the customers stay loyal to the bank).

4.3.4 Labelling the Records

The churn event is not well defined per se, therefore it is necessary to formally specify it to correctly label the records (as said in Section 3.2).

The standard definition posits that a customer is a "churner" in a specific month if he/she closed all the bank accounts in that month. However, observe that since the main need of a bank is to retain its customers as long as possible, it is useless to identify customers with too short notice with respect to their departure, since then the bank cannot rectify customers' behavior, as the available period of time is not sufficiently long. For this reason,

our analysis aims at training a classifier able to detect customers who leave with adequate advance. However, the standard definition of "churner" is not sufficient to satisfy this requirement, since labelling the records with this definition would create a classifier useful to predict the precise instant of time when the customer leaves, making behavioral rectification actions impossible to be applied (as discussed in Subsection 3.4.3). Thus, it is necessary to modify the standard definition of "churner", in order to make sure that the classifier will predict a time instant far enough from the real churn instant.

In this thesis we chose to predict the churn event one month in advance. Technically speaking, we shifted the churn event one month backward, in order to label as churners the customers who leave during the following month. This means that the churn flag (i.e. the variable which describes if a customer churns in a specific month) is turned on for the penultimate record, while the last record becomes irrelevant. In fact, the churners' data related to the last month would introduce noise in the analysis, because these data are not concerned with the event we want to predict (i.e. forecast which customers will leave during the following month). This means that it is necessary to remove those customers who join and then leave the bank in less than one month, because there is not a sufficient period of time available to apply the previous operations.

We present an example of the churn event shift Table 4.2. The considered customer (ID 0001) leaves the bank during May 2015, hence the churn event is shifted to April 2015 and the record related to May 2015 is removed from the data.

In particular, the table on the left shows the standard churn event, while

the table on the right presents the shifted churn event.

ID	Month	Churn
0001	Jan-15	No
0001	Feb-15	No
0001	Mar-15	No
0001	Apr-15	No
0001	**May-15**	**Yes**

becomes ⟹

ID	Month	Churn
0001	Jan-15	No
0001	Feb-15	No
0001	Mar-15	No
0001	**Apr-15**	**Yes**
-	-	-

Table 4.2: Churn Event Shift Example

The use of these new labels to train a classifier leads to the construction of a model capable of predicting if a customer will leave in the following month. In fact, a classifier will search the values of features which characterize the behavior of the customers labelled as "churner" with the new definition, that is, the customers who will leave during the following month.

Training and Test Set Partitioning

The churn prediction problem raises a non-trivial data partition issue. In fact, the standard validation technique (Section 2.2) is not appropriate, since it would not return a reliable estimate of the classifier performance. This happens because behavior of the customers who leave could change over time, that is, the main reasons which drive the customers to leave the bank vary over time.

Business people are interested in a model which predicts the future customers' behavior, not the present ones, therefore the standard validation approach does not work for this purpose. In fact, with the standard approach the classifier would be evaluated on customers characterized by the same behavioral features of the customers contained in the training set, hence

returning an erroneous model performance estimation.

The approach used to avoid this problem is to divide the data temporally. In particular, we used the available customers' information between January 2015 and December 2015 to create the training set and those of January 2016 to generate the test set. We explain the details of this subdivision after proving that the standard validation procedure is not appropriate for churn prediction problems.

We tested the differences between the two validation procedures evaluating a CART classifier trained with all the multi-month attributes and 70% of randomly chosen customers. Concerning the standard validation, we evaluated CART on the remaining 30% of the customers, while for the non-standard approach we evaluated the classifier on the test set generated by the temporal subdivision. Table 4.3 shows precision and recall performances obtained with both validation approaches.

	Standard	**Non-Standard**
Precision	0.9672	0.0301
Recall	0.7013	0.0022

Table 4.3: Estimation of CART performances with two validation approaches

The standard validation procedure leads one to think that this CART model is very accurate in predicting customer churn, since it correctly identifies 70.1% of the churners with 96.7% of precision. Actually, the non-standard procedure reveals that the model fails to generalize when data related to future periods of time are used. In fact, both precision and recall values collapse to near zero values! This proves that the standard validation approach is not appropriate for churn prediction problems. Moreover, the non-standard vali-

dation procedure is far more penalizing than the standard one, since it evaluates the model ability to manage with different churners' behaviors compared to those included in the training set. For this reason, the results we obtained in this thesis are much lower than those usually reported in the literature, but nevertheless they offer an honest characterization of the performance that a model would reach in real business applications. In any case, domain knowledge ensures us that our results (see Section 5.1.4) are considered standard in banking environments.

In the following we explain in details the procedure we used to create the training set and the test set. Since we considered two different periods of time (i.e. January 2015-December 2015 for the training set and January 2016 for the test set), we separately compute the churn labels and the multi-month attributes on both the sets of data.

Recall that the churn event is shifted one month backward (Subsection 4.3.4), in order to guarantee an early identification of churners by the bank. From here on out in this section we refer to this shifted churn event when we state that a customer leaves. For example, lets consider a customer who actually leaves during October 2015. We consider September 2015 as the month during which the customer leaves the bank. The general procedure is the following:

- A customer is labelled as non-churner if he/she does not leave the bank in the considered period of time.
- A customer is labelled as churner if he/she leaves the bank during the considered period of time.

- The multi-month attributes specified in Subsection 4.3.3 are computed for the time subinterval defined as follows:
  - up to the last month included in the considered period of time for those customers who do not leave during the same period,
    or
  - up to the specific month during which the shifted churn event happens, for those customers leaving the bank in the considered period.

In Figure 4.1 we show a concise plot, which clarifies how we treat the customers who leave, during the creation of the training set or test set. The initial point represents the customers arrival, while the final point the shifted churn event. In particular, in the considered period of time:

- Customer n.1 is considered as churner
- Customer n.2 is not considered at all
- Customer n.3 is not considered at all
- Customer n.4 is considered as non-churner
- Customer n.5 is considered as churner
- Customer n.6 is considered as non-churner

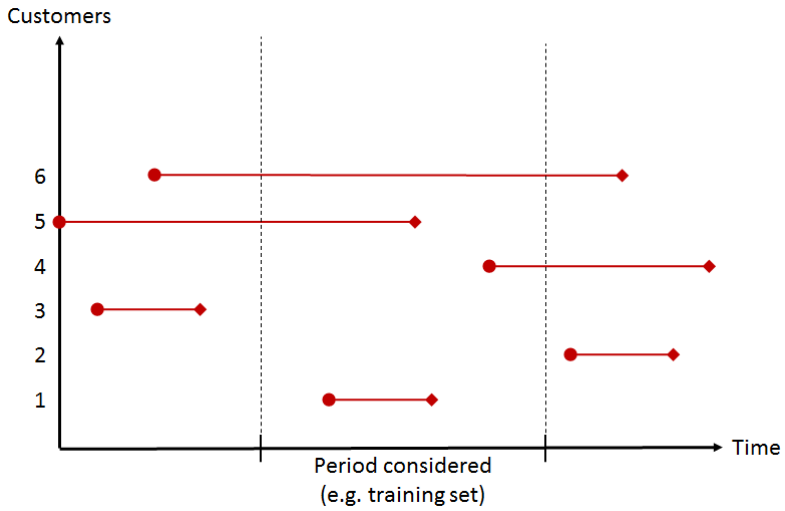

Figure 4.1: Possible churn events in a fixed period of time

The multi-month attributes related to the customers n.4 and n.6 (non-churners) have to be computed using as the last available month the last month of the considered period. The multi-month attributes related to the customers n.1 and n.5 (churners), instead, have to be computed up to the month during which the shifted churn event happens.

Customers who leave and subsequently rejoin the bank are not considered in the figure, because they need ad hoc reasoning, which is discussed in Subsection 4.4.3.

In the context of this project, it is important to notice that, as the churn event was shifted one month backward, the information related to the churners who left in January 2016 were included in the training set, while the data related to the churners who left in February 2016 were included in the test set. In fact, a customer who leaves during January 2016 would be labelled as a churner of December 2015 and thus must be included in the training set.

At the end of this phase, the training set contains 717422 customers and 15731 of them are churners. The test set, instead, contains 709213 customers and 1836 of them are churners. Both sets are described by 519 features and a binary class (churner/non-churner). Table A.1 shows a summary of the attributes contained in the dataset, highlighting their logical subdivision among demographics, transactional attributes and balance features. In particular, the demographics was composed of 7 attributes, the transactional features amounted to 496 attributes and the balance amounted to 16 attributes. A complete description of the attributes of the dataset can be found in Appendix A.

Dataset Structure
Customer Demographics
Age
⋮
Gender
Transactional multi-month attributes
Avg card payment - 1 month
Avg card payment - 3 months
Avg card payment - 6 months
Avg card payment - 12 months
Min card payment - 1 month
⋮
Sum withdrawals - 12 month
Balance multi-month attributes
Avg end balance - 1 month
Avg end balance - 3 months
Avg end balance - 6 months
Avg end balance - 12 months
Min end balance - 1 month
⋮
Sum end balance - 12 months
Class
Churn Flag

Table 4.4: Dataset Structure

4.4 Preprocessing Phase

In this section we present all the preprocessing actions we made on the training set, in order to improve the classifier performance. In particular, we show the feature selection phase, the sampling phase and the outlier analysis.

4.4.1 Feature Selection

The training set is quite affected by the curse of dimensionality (Subsection 2.4.2), which increases its complexity. Hence, it is necessary to select the most relevant features, in order to tackle this issue and make the final classifier simpler and more generalizable. However, due to the high data dimensions, standard feature selection approaches take too much computational time to be efficiently applied. Therefore, we used less sophisticated methods to speed up the pace of the analysis.

The first feature selection approach that we used consists in estimating the importance of each feature independently. More precisely, we trained an univariate model (i.e. a classifier that uses only one feature) for each attribute of the training set. Every model was then evaluated on the test set, in order to independently estimate the importance of each feature. We tested this technique using CART as a classifier, because its computational time with one feature was the lowest among the models evaluated. Indeed, it was necessary to train about 500 models to complete the process, since the training set contained about 500 features, and a slow classification algorithm could have taken too much time to be trained 500 times. Table 4.5 shows the average training time of univariate models depending on the algorithm

used.

Classification Algorithm	Average training time with one feature (seconds)
CART	**1.39**
Logistic Regression	5.18
C5.0	11.04
Random Forest	50.47

Table 4.5: Univariate models training time

However, CART was not able to separate the two classes using only one feature at a time. In fact, the decision tree always resulted in just a root node which classified all the records as non-churners. For this reason, we opted for another feature selection procedure, which takes into account the dependencies among the variables. In particular, we used the *Random Feature Selection* technique (Subsection 2.4.2), which iteratively evaluates subsets of randomly chosen features, in order to estimate the most promising one.

In this thesis we tested several subsets of five randomly chosen features, which were then used to train a C5.0 classifier (Subsection 2.5.1). The C5.0 was then evaluated on the test set, in order to estimate the importance of each subset. In particular, the classifier performances were evaluated seeking the best compromise between precision and recall (Section 2.3), in order to maximize the number of churners covered by the model and to minimize the false positive errors (i.e. classify a non-churner as churner, Table 2.2).

After this phase, we locked the most promising subset and iteratively added another feature chosen at random, in order to improve the just obtained results. We stopped the process when no additional feature provided a better separation of the two classes (i.e. it did not improve precision and

recall), resulting in a final subset with eight features in total.

4.4.2 Sampling

As said in Subsection 3.5.4, the number of churners is usually much smaller than the number of non-churners, leading to an imbalanced training set.

This issue was also present in our analysis, since the annual bank churn rate was about 2.5% only. For this reason, different sampling techniques were tested to balance the two classes (Subsection 2.4.1). In particular, the most effective method was *undersampling*, which outperformed the sampling techniques that replicates the rare class, since the latter ones resulted in overfitting the rare class, thus degrading the model performance significantly. In particular, the undersampling technique significantly improved the results only if combined with the outlier removal (Section 5.1.4). The best proportion between classes was chosen so to optimize the classifier performance on the test set.

4.4.3 Outlier Analysis

Subsection 3.5.1 explains why outlier analysis is useful to improve the classifier quality. Exploring the bank database, we detected some strange customers' behaviors, which have to be treated carefully. In particular, we noticed that some customers, who have already left the bank in the past, came back in subsequent time periods. These customers could be potentially characterized by a peculiar behavior, therefore they have to be managed separately. Since the amount of these customers was very low (about 100), we

simply chose to remove them from the analysis, in order to avoid ineffective efforts for their management.

The reasons which cause customers defection change over time quickly. This means that the customers, who left the bank a long time ago and were included in the training set, could potentially introduce noise in the model. In fact, they could show a complete different behavior compared to the churners considered in the test set, who left more recently. For example, a customer who left in March 2015 would probably show different reasons for leaving compared to a customer who left in January 2016.

This observation leads us to investigate how much the "freshness" of churners influences the model performance. In particular, we discovered that customers, who left more than seven months earlier from the considered test set starting instant, introduce noise in the classifier, resulting in lowered performance (Table 5.5). For this reason, we chose to maintain only the recent churners for training the model.

4.5 Final Summary

In this chapter we have described in details the practical problem that we faced. Moreover, we have shown different preprocessing actions, such as dimensionality reduction and the creation of multi-month attributes, which allowed us to create a standard and manageable dataset with one record per customer. Furthermore, we have formally defined the "churner" as a customer who leaves during the subsequent month. In particular, this new definition led us to change the customers labels, shifting the churn event

one month backward. Moreover, we have shown a non-standard validation approach, which estimates the ability of the model to identify the churners in periods of time subsequent to the period related to the training set. We have also presented the random feature selection technique, which we used to select the most relevant attributes, in order to decrease the training time of the classifiers and to increase their predictive accuracy. Eventually, we have shown both the sampling and the outlier removal actions that we applied to the training set, in order to increase the classifier performances.

Chapter 5

Churn Prediction Model Development and Experimental Validation

In this chapter we present several classification models for the churn prediction case study described in the previous chapter, and compare their relative performances in terms of standard metrics such as precision, recall, specificity, and AUC. These performances also allow us to evaluate the effectiveness of the outlier removal, undersampling and ensemble techniques. Recall that the results we obtained in this thesis are generally much lower than those reported in the literature, due to the more meaningful validation approach used to estimate the real business performances of the models (Subsection 4.3.4), which differs from the standard approaches but provides more meaningful estimates..

5.1 Modelling Phase

Customer churn turned out to be a very difficult event to predict. In fact, simple classifiers, as logistic regression or CART (Subsection 2.5.1, 2.5.3), did not reach very good results. Table 5.1 and 5.2 show the confusion matrices related to the predictions of CART and logistic regression on the test set. Notice that CART yields a lot of false positives compared to the number of true positives, while logistic regression almost totally failed to predict which customers will leave.

		True Classes	
		Non-Churner	Churner
Predictions	Non-Churner	706476	1681
	Churner	901	155

Table 5.1: Confusion Matrix of CART

		True Classes	
		Non-Churner	Churner
Predictions	Non-Churner	707248	1832
	Churner	129	4

Table 5.2: Confusion Matrix of Logistic Regression

The large size of the dataset made the use of models with high training times unfeasible for our purposes. In fact, algorithms such as SVM or Adaboost did not even finish the training process after five hours of computational effort. Table 5.3 shows the training time of several models we applied to the training set, using only the eight selected features (as explained in Subsection 4.4.1). For these reasons, we aimed at a tradeoff between predictive accuracy and training time.

Algorithm	Time (minutes)
Logistic Regression	0.15
C5.0	0.76
CART	0.80
RIPPER	8.4
Random Forest	12.5
Adaboost	NA
SVM	NA

Table 5.3: Computational training time of classifiers

We found that the best classification algorithm which satisfied these conditions was C5.0, the same algorithm used in the feature selection phase. In fact, C5.0 is suited to work with large datasets, therefore its training time is sufficiently low to be feasible for our purposes.

We used C5.0 to train several models on different preprocessed training sets. In particular, each model has been trained using only the eight features selected in the feature selection phase (Subsection 4.4.1).

First, we trained a C5.0 on the entire training set (i.e. not affected by any preprocessing actions), in order to get a baseline score useful to figure out whether the classifiers built on preprocessed training sets improve the results, or not. In particular, the baseline scored 0.2774 in precision and 0.0414 in recall (Table 5.8).

5.1.1 Undersampling and Oversampling

As discussed in Subsection 4.4.2, we applied sampling techniques to balance the classes. In particular, we applied both undersampling and oversampling (Subsection 2.4.1) to figure out which technique could potentially improve the baseline score. Both techniques require a parameter (called p), which specifies

the desired proportion between churners and non-churners in the training set. In particular, we tuned this parameter aiming at the best compromise between precision and recall, which were estimated on the test set. For this purpose, we chose $p = 0.03$ for both undersampling and oversampling. Table 5.4 shows the performances reached using these techniques.

Undersampling			**Oversampling**		
p	*Precision*	*Recall*	p	*Precision*	*Recall*
0.03	**0.2490**	**0.0563**	**0.03**	**0.0449**	**0.0387**
0.04	0.2102	0.0403	0.04	0.0242	0.0038
0.05	0.2215	0.0459	0.05	0.0283	0.0148
0.06	0.2276	0.0584	0.06	0.0265	0.0391
0.07	0.1970	0.0430	0.07	0.0272	0.0388
0.08	0.2346	0.0564	0.08	0.0225	0.0399
0.09	0.1816	0.0493	0.09	0.0202	0.0378
0.10	0.2182	0.0556	0.10	0.0183	0.0208

Table 5.4: Undersampling (left) and Oversampling (right) performances. The final choice of p is highlighted in bold.

As shown by the results, undersampling outperformed oversampling in predicting customer churn. In fact, oversampling led to overfitting the churner class, resulting in very low precision. The undersampling, instead, increased the recall, but lowered the precision compared to the baseline score (i.e. precision 0.2774, recall 0.0414). Hence, it is not trivial to assert if the classifier built with undersampling is a better model than the baseline. For this reason, the final judgment depends on the bank needs, that is, if the bank prefers better precision or better recall for its business.

We did not test SMOTE or a combination of undersampling with oversampling to balance the classes.

5.1.2 Outlier Removal

In Subsection 4.4.3 we observed that the churners who left the bank near the beginning of the time period considered, which we refer to as early churners, could introduce noise in the model, since the reasons which cause customers defection change over time quickly. For this reason, we investigated the impact of early churners on the performances of the classifier. Table 5.5 shows the C5.0 performances obtained removing from the training set the customers who left before the month indicated by the column "Month", and the baseline score. An NaN value of precision means that the model classified all the records as non-churners, resulting in zero recall and indeterminate precision.

The removal of those customers who left before June 2015 improved the overall results, outperforming the baseline score both in precision and recall measures. In particular, we removed 5894 churners. This confirms that customers who left a long time ago exhibit a different behavior compared to the recent churners.

Month (2015)	Precision	Recall
February	0.2763	0.0387
March	0.3029	0.0507
April	0.2857	0.0142
May	0.2766	0.0142
June	**0.3455**	**0.0566**
July	0.3436	0.0545
August	0.2778	0.0136
September	0.6061	0.0109
October	0.6061	0.0109
November	0.6061	0.0109
December	NaN	0
Baseline	0.2774	0.0414

Table 5.5: Old churner removal - performances

5.1.3 Easy Ensemble Technique

We continued the analysis combining the undersampling technique with the outliers removal. In particular, we removed the customers who left before June 2015 from the training set first, and then applied the undersampling technique with $p = 0.03$, thus using the parameters which yielded the best performances in the two preprocessing steps, when executed in isolation. The model scored 0.3140 in precision and 0.0735 in recall, as Table 5.8 shows (C5.0_US_CH).

The C5.0, combined with the undersampling technique and the outliers removal, reached quite good results and we studied if its performance could be improved using an ensemble technique.

The undersampling method throws away many non-churner records to balance the classes proportion, hence we implemented an ensemble technique which allows to maintain all the data and balance the classes proportion at the same time. In particular, we implemented the *Easy Ensemble* technique

proposed by Xu-Ying Liu et al. [42]. This method is divided in three steps:

1. Generate N different datasets through the application of the undersampling technique;

2. Train a classifier on each dataset;

3. Mediate arithmetically the classifiers predictions to make the final forecasts.

This procedure allows to cover all the available non-churner information, if a proper value for N is chosen. In fact, given the p undersampling parameter, it is possible to estimate a value N such that on average all the data are used. In particular, estimating a proper N can be modelled as an extension of the famous *Coupon Collector Problem*, which studies how many times we have to buy random coupons, in order to complete the coupon collection composed of n coupons. The original problem considers the case of buying only one coupon at a time, which requires on average to pick $\Theta(nlogn)$ coupons to complete the collection [4]. The extension considers the case of buying batches which contain several coupons [33], as it happens in our problem. In fact, the goal is to pick all the records at least once with N extractions from the entire training set. Since the formula used to compute the number of batches required to complete the coupon collection works only if applied with small numbers, we used a numeric solution to estimate the best N. In particular, we applied undersampling with $p = 0.03$ to the training set after the outlier removal, hence generating a dataset with about 328000 records. We simulated several random picking of 328000 records from the training set,

which contained 711528 observations after the outlier removal phase. The average number of iterations required to get all the records was about 22, with a minimum of 19 and a maximum of 30.

We tested the easy ensemble technique with different N values between 10 and 35. Table 5.6 shows the precision and recall measures, which were obtained evaluating on the test set several models obtained through the application of the easy-ensemble technique with different values of N. As expected, the best performances were achieved around the estimated average number of iterations (i.e. $N = 22$), while larger or smaller numbers of iterations caused loss of precision.

N	**Precision**	**Recall**
10	0.3123	0.0703
15	0.3244	0.0724
20	0.3475	0.0714
22	0.3483	0.0719
25	**0.3492**	**0.0719**
30	0.3251	0.0719
35	0.3192	0.0700

Table 5.6: Easy Ensemble performances with different number of iterations

5.1.4 Boosting

Another common method to improve the classifier performance is the boosting technique, as discussed in Subsection 2.5.4. The C5.0 implementation includes the boosting technique, which requires as parameter the number of *trials*. A number of trials equal to one means no boosting, while a number greater than one specifies how many boosting iterations are requested. We tested this method also, but the results showed that boosting led to overfit-

ting, hence the boosting technique was not employed. Table 5.7 shows the precision and recall obtained with boosting on the test set, where the first row indicates no boosting (i.e. the baseline score). Notice that boosting has never outperformed the baseline performances. We did not evaluate boosting with more than nine trials, because of an excessive training time for the classifier.

Trials	**Precision**	**Recall**
1	0.2774	0.0414
2	0.2774	0.0414
3	0.0221	0.0054
4	0.0562	0.0076
5	0.0188	0.0022
6	0.1247	0.0305
7	0.0239	0.0049
8	0.0641	0.0054
9	0.0397	0.0074

Table 5.7: Boosting performances

Let us summarize the above findings. We have evaluated the performances of C5.0 combined with undersampling, oversampling and outlier removal. Oversampling did not help to improve the results, while undersampling allowed to increase recall, sacrificing a little of precision. Moreover, the outlier removal was very effective in increasing the overall performances, since it removed a lot of noise from the model. We have also tested a mixed version of C5.0, applying both outlier removal and undersampling, resulting in quite good precision and recall. Moreover, we have successfully improved the overall results through the use of the Easy Ensemble technique, which allowed to apply undersampling while using all the available data. Eventually, we have tried to improve the results using the built-in boosting algorithm offered by the C5.0 software, but we have found that boosting led to overfitting.

5.2 Results

In this section we compare the results obtained with the classifiers described in Section 5.1. Table 5.8 summarizes the results obtained evaluating each model considered on the test set, which contains 709213 customers, 1836 of them are churners. The models considered are:

- Baseline: it is the direct application of the C5.0 algorithm, without particular preprocessing.
- C5.0_CH: it is the C5.0 trained using only the churners who left after May 2015.
- C5.0_US: it is the C5.0 trained on the undersampled dataset (with

$p = 0.03$).

- C5.0_US_CH: it is the C5.0 trained on the undersampled dataset (with $p = 0.03$), which contains only the churners who left after May 2015.

- EE C5.0: it is the classifier obtained through the easy ensemble technique (with $p = 0.03$ and $N = 25$), considering only the churners who left after May 2015.

	Baseline	**C5.0_CH**	**C5.0_US**	**C5.0_US_CH**	**EE C5.0**
Precision	0.2774	0.3455	0.2490	0.3140	**0.3465**
Recall	0.0414	0.0566	0.0563	**0.0735**	0.0719
Specificity	0.9997	0.9997	0.9996	0.9996	0.9996
AUC	0.6370	0.6720	0.6233	0.6560	0.6720

Table 5.8: Classifiers Performance

The baseline succeeded to predict the 4% of churners correctly with 27.74% of precision. These results were improved removing the old churners, that is, the customers who left before June 2015 (Subsection 5.1.1). In fact, the C5.0_CH scored higher precision and recall compared to the baseline, reaching 34.6% of precision and 5.7% of recall. As shown by the results, undersampling tended to increase the recall measure, sacrificing a little of precision (see baseline compared to C5.0_US and C5.0_CH compared to C5.0_CH_US). This was expected, since undersampling leads a classifier to suppose that the proportion of churners is higher than the real one, because several non-churners are removed from the training set. For this reason, a model tends to classify a customer as churner more easily, resulting in lower precision and, hopefully, higher recall.

The EE C5.0 addressed the disadvantages of undersampling, using all the available data to improve the results. In fact, the EE C5.0 recovered the precision lost with undersampling (C5.0_CH_US), maintaining a similar recall at the same time. EE C5.0 correctly classified 7.2% of the churners with 34.7% of precision.

All of the models considered featured a very high specificity (about 99.9%), which is the fraction of non-churners correctly predicted. This means that almost all the non-churners were precisely identified. In fact, due to the heavy classes disequilibrium, a generic model tended to classify all the records as non-churners, in order to minimize the errors, thus resulting in high specificity. However, a high specificity was fundamental to reduce the bank costs related to false positive errors. In particular, low recall combined with high specificity is reasonable in banking environments, since customers labelled as churners can be retained without wasting money on many non-churners, increasing the bank profits.

The AUC values of all models were also quite similar, reaching a maximum of 67.20% with C5.0_CH and EE C5.0.

Tables 5.9, 5.10 and 5.11 present the confusion matrices of the C5.0_CH, the C5.0_CH_US and the EE C5.0. The EE C5.0 resulted to be the best model, since it scored both high precision and high recall, compared to the other classifiers.

		True Classes	
		Non-Churner	Churner
Predictions	Non-Churner	707180	1732
	Churner	197	104

Table 5.9: Confusion Matrix of C5.0_CH

		True Classes	
		Non-Churner	Churner
Predictions	Non-Churner	707082	1701
	Churner	295	135

Table 5.10: Confusion Matrix of C5.0_CH_US

		True Classes	
		Non-Churner	Churner
Predictions	Non-Churner	707128	1704
	Churner	249	132

Table 5.11: Confusion Matrix of EE C5.0

It must be observed that the values of precision and recall featured by our best models are lower than those reported by churn prediction studies found in the literature. However, it is important to remark that unlike those studies, we employed a non standard validation approach which provides a much more meaningful estimate of the effectiveness of the model in realistic scenarios. In fact, Fan Li et al. [36] reached 88.3% of precision and 79.9% of recall using a Ten-fold Cross Validation (Section 2.2) to evaluate RIPPER. Dr. U. Devi Prasad et al. [23] reached 91.2 % of precision and 61.2% of recall using an Holdout approach (Section 2.2) to evaluate CART. Recall that we obtained similar results using an Holdout approach to evaluate CART in Subsection 4.3.4. These performances are useful to measure the ability of classifiers to understand the old reasons of abandonment, but they do not estimate the capability of models to predict future customers' behavior.

5.3 Improved Results

All the results showed in the previous section were calculated evaluating the models performances on the data related to the customers' behavior of January 2016, which included the information of the customers who left in February 2016, due to the churn event shift, as explained in Subsection 4.3.4. However, it is possible that some non-churners of January 2016 left in the subsequent months, namely, March and April 2016. This means that those customers potentially have already exhibit a churner behavior in the test set, because they left shortly after.

Since a bank is interested in predicting as soon as possible which customers will leave, the non-churners who left in March and April 2016, and labelled as churner by a classifier, could be considered as correct predictions. Thanks to this, the bank can spend more time in rectifying these customers' behavior, increasing the probability of retention. Recall that the original test set considered as churners only those customers who left in February 2016 even though, as explained before, they appear as they had left in January 2016.

We took into account the possibility that a customer shows the churner behavior many months before the period during which he/she actually leaves, and so we studied how many non-churners, classified as churners of February 2016, actually left during March or April 2016. We considered the EE C5.0 classifier only, since it was the most promising model.

It turned out that 107 of 249 non-churners classified by the EE C5.0 as churners actually left the bank during March or April 2016. This means

that these customers had already shown a churner behavior during February 2016, allowing the EE C5.0 to identify them earlier. This capability of the model is an important value added for the bank, because the bank applies the rectifying actions to all the customers identified as churners by the classifier. Hence the bank can make even more profits, retaining also the customers who have already shown a churner behavior, but will leave only in subsequent months.

Table 5.3 shows the precision, recall and AUC metrics obtained considering the customers who left in March and April 2016 as churners in the test set.

Precision	0.6273
Recall	0.1230
Specificity	0.99980
AUC	0.6970

Table 5.12: Rectified EE C5.0 Performance

The performances got significant improvements, resulting in about doubled precision and recall. In fact, the classifier reached about 62.7% of precision and 12.3% of recall.

Chapter 6

Conclusions

The objective of this thesis has been to assess, on a case study, to what extent customer churn in a banking environment can be predicted using data mining techniques. Specifically, we aimed at predicting in advance which customers are likely to leave a bank with quite good precision, avoiding costs related to false positive errors (i.e. non-churners classified as churners).

In the first chapter we have presented several data mining tools, which are already consolidated in the literature, and some typical issues related to classification tasks, such as the class imbalance and the curse of dimensionality.

In the second chapter we have explained why predicting customers' behavior is a fundamental process to increase bank profits. Moreover, we have addressed the most relevant issues related to the churn prediction problem and several ad hoc solutions proposed in the literature. Furthermore, we have presented several classifiers developed in the literature for predicting customer churn, such as ALBA and IBRF. However, these peculiar models

have not been developed as open softwares, therefore we could not use them in our project. In fact, an implementation from scratch would have taken too much time, leading to additional costs that a POC project cannot support.

In the third chapter we have described the structure of the dataset made available to us and the peculiar characteristics of the specific problem we faced during the POC project. In particular, we have addressed the critical points, such as the dataset creation and preprocessing phases, showing the implemented solutions also.

In the fourth chapter we have described which classifiers we trained to forecast customer churn, highlighting the best model found. Moreover, we have shown the reached results and compared the models performances among them. In particular, the EE C5.0 turned out to be a powerful classifier for forecasting customers' behavior, reaching a sufficient precision and recall for our purposes to confirm the POC. Therefore, further studies to explore the real model potential may be worth considering.

Future studies should focus on seeking new discriminating features, which might allow to characterize the customers' behavior and separate the two customers classes better. For example, one possibility is to create multi-month attributes on shorter periods of time, in order to describe customers movements more accurately. Another possibility is to compute value variations of each attribute over time, rather than using the static values per se. In fact, high variations may be very informative about relevant customers movements.

It is important to notice that in this thesis we have used only the balance, demographic and transactional data. Hence, a future study may collect more

customers' information to better understand the reasons of abandonment. Moreover, we have labelled a customer as churner in a given month if he/she left during the subsequent month. Since this event could turn out to be too difficult to be efficiently predicted, it would be interesting to test different churn event definitions, in order determine any significant differences among the various results.

In conclusion, we have seen that predicting customer churn is a very challenging task due to its temporal characteristic, which increases the overall data analysis complexity. In fact, a standard procedure to model time events in classification tasks does not exists yet, making time varying problems widely discussed and studied in the literature. However, we have proved that data mining tools can help banks to understand their customers' behavior, confirming that further studies may be worth considering.

Appendix A

Dataset Attributes

In this appendix we report the dataset we used in this thesis. In particular, Table A.1 shows the attributes and their domain (Section 2.1). For the sake of simplicity, each multi-months attribute stands for all the different versions of it, that is, those calculated on 1 month, 3 months, 6 months and 12 months. For example, Avg card payment represents the average of Card Payment computed on 1 month, 3 months, 6 months and 12 months.

Name	**Domain**
Customer Demographics	
Age	Quantitative
Gender	Qualitative
Seniority (years)	Quantitative
Postal province	Qualitative
Postal region	Qualitative
Birth province	Qualitative
Birth region	Qualitative
Transactional multi-month attributes	
Avg card payment	Quantitative

Avg salary pension	Quantitative
Avg forms	Quantitative
Avg titles	Quantitative
Avg bank transfer	Quantitative
Avg capital gain	Quantitative
Avg insurances	Quantitative
Avg closures	Quantitative
Avg payment fee	Quantitative
Avg services	Quantitative
Avg stamps	Quantitative
Avg consumptions	Quantitative
Avg prepaid recharge	Quantitative
Avg transfers	Quantitative
Avg deposits	Quantitative
Avg withdrawals	Quantitative
Avg checks emission	Quantitative
Avg promotion	Quantitative
Avg credit	Quantitative
Avg debit	Quantitative
Avg number of accounts	Quantitative
Avg number of transactions	Quantitative
Avg gcca	Quantitative
Avg gca	Quantitative
Avg cc	Quantitative
Avg ca	Quantitative
Avg to	Quantitative
Avg cp	Quantitative
Avg money moved	Quantitative
Avg salary flag	Quantitative
Avg transactions flag	Quantitative
Min card payment	Quantitative
Min salary pension	Quantitative
Min forms	Quantitative
Min titles	Quantitative
Min bank transfer	Quantitative
Min capital gain	Quantitative
Min insurances	Quantitative
Min closures	Quantitative
Min payment fee	Quantitative
Min services	Quantitative
Min stamps	Quantitative

Min consumptions	Quantitative
Min prepaid recharge	Quantitative
Min transfers	Quantitative
Min deposits	Quantitative
Min withdrawals	Quantitative
Min checks emission	Quantitative
Min promotion	Quantitative
Min credit	Quantitative
Min debit	Quantitative
Min number of accounts	Quantitative
Min number of transactions	Quantitative
Min gcca	Quantitative
Min gca	Quantitative
Min cc	Quantitative
Min ca	Quantitative
Min to	Quantitative
Min cp	Quantitative
Min money moved	Quantitative
Min salary flag	Quantitative
Min transactions flag	Quantitative
Max card payment	Quantitative
Max salary pension	Quantitative
Max forms	Quantitative
Max titles	Quantitative
Max bank transfer	Quantitative
Max capital gain	Quantitative
Max insurances	Quantitative
Max closures	Quantitative
Max payment fee	Quantitative
Max services	Quantitative
Max stamps	Quantitative
Max consumptions	Quantitative
Max prepaid recharge	Quantitative
Max transfers	Quantitative
Max deposits	Quantitative
Max withdrawals	Quantitative
Max checks emission	Quantitative
Max promotion	Quantitative
Max credit	Quantitative
Max debit	Quantitative
Max number of accounts	Quantitative

Max number of transactions	Quantitative
Max gcca	Quantitative
Max gca	Quantitative
Max cc	Quantitative
Max ca	Quantitative
Max to	Quantitative
Max cp	Quantitative
Max money moved	Quantitative
Max salary flag	Quantitative
Max transactions flag	Quantitative
Sum card payment	Quantitative
Sum salary pension	Quantitative
Sum forms	Quantitative
Sum titles	Quantitative
Sum bank transfer	Quantitative
Sum capital gain	Quantitative
Sum insurances	Quantitative
Sum closures	Quantitative
Sum payment fee	Quantitative
Sum services	Quantitative
Sum stamps	Quantitative
Sum consumptions	Quantitative
Sum prepaid recharge	Quantitative
Sum transfers	Quantitative
Sum deposits	Quantitative
Sum withdrawals	Quantitative
Sum checks emission	Quantitative
Sum promotion	Quantitative
Sum credit	Quantitative
Sum debit	Quantitative
Sum number of accounts	Quantitative
Sum number of transactions	Quantitative
Sum gcca	Quantitative
Sum gca	Quantitative
Sum cc	Quantitative
Sum ca	Quantitative
Sum to	Quantitative
Sum cp	Quantitative
Sum money moved	Quantitative
Sum salary flag	Quantitative
Sum transactions flag	Quantitative

Balance multi-month attributes	
Avg end balance	Quantitative
Min end balance	Quantitative
Max end balance	Quantitative
Sum end balance	Quantitative
Class	
Churn Flag	Qualitative

Table A.1: Dataset Structure

Bibliography

[1] Verbeke W. & Dejaeger K. & Martens D. & Hur J. & Baesens B., *New insights into churn prediction in the telecommunication sector: A profit driven data mining approach*, 2012.

[2] Wouter Verbeke & David Martens & Christophe Mues & Bart Baesens, *Building comprehensible customer churn prediction models with advanced rule induction techniques*, 2011.

[3] Džulijana Popović & Bojana Dalbelo Bašić, *Churn prediction model in retail banking using fuzzy c-means algorithm*, 2008.

[4] Vassilis G. Papanicolaou & George E. Kokolakis & Shahar Boneh, *Asymptotics for the random coupon collector problem*, 1998.

[5] C. Chen & A. Liaw & L. Breiman, *Using random forest to learn imbalanced data*, 2004.

[6] Francis Buttle, *Customer relationship management: Concepts and technologies*, 2009.

[7] Ling R. & Yen D. C., *Customer relationship management: An analysis framework and implementation strategies*, 2001.

[8] Prof. Nilima Patil & Prof. Rekha Lathi & Prof. Vidya Chitre, *Comparison of c5.0 & cart classification algorithms using pruning technique*, 2012.

[9] Tim Coltman, *Why build a customer relationship management capability?*, 2007.

[10] Dirk Van den Poel & Bart Larivière, *Customer attrition analysis for financial services using proportional hazard models*, 2004.

[11] J. Burez & D. Van den Poel, *Handling class imbalance in customer churn prediction*, 2009.

[12] M. Rajeswari & T. Devi, *Design of modified ripper algorithm to predict customer churn*, 2015.

[13] David Martens & Bart Baesens & Tony Van Gestel, *Decompositional rule extraction from support vector machines by active learning*, 2009.

[14] Aurélie Lemmens & Sunil Gupta, *Managing churn to maximize profits*, 2013.

[15] Shouwei Li & Mingliang Wang & Jianmin He, *Prediction of banking systemic risk based on support vector machine*, 2013.

[16] Elsevier Inc., *Data mining: Concepts and techniques*, 2012.

[17] Pearson K., *On lines and planes of closest fit to systems of points in space*, 1901.

[18] Adnan Idris & Muhammad Rizwan & Asifullah Khan, *Churn prediction in telecom using random forest and pso based data balancing in combination with various feature selection strategies*, 2012.

[19] M Chandar & A Laha & P Krishna, *Modeling churn behavior of bank customers using predictive data mining techniques*, 2006.

[20] Ding-An Chiang & Yi-Fan Wang & Shao-Lun Lee & Cheng-Jung Lin, *Goal-oriented sequential pattern for network banking churn analysis*, 2003.

[21] Chih-Fong Tsai & Yu-Hsin Lu, *Customer churn prediction by hybrid neural networks*, 2009.

[22] Tom Au & Shaomin Li & Guangqin Ma, *Applying and evaluating models to predict customer attrition using data mining techniques*, 2003.

[23] Dr. U. Devi Prasad & S. Madhavi, *Prediction of churn behavior of bank customers using data mining tools*, 2012.

[24] DJ Hand & HJ Adèr & GJ Mellenbergh, *Advising on research methods: A consultant's companion*, 2008.

[25] Ed Thompson & Scott D. Nelso, *How to develop a crm strategy*, 2004.

[26] NGData, *Predicting & preventing banking customer churn by unlocking big data*, 2013.

[27] Teemu Mutanen & Jussi Ahola & Sami Nousiainen, *Customer churn prediction - a case study in retail banking*, 2003.

[28] S.B. Kotsiantis & D. Kanellopoulos & P.E. Pintelas, *Data preprocessing for supervised leaning*, 2006.

[29] A. Kalja & H.M. Haav & T. Robal, *Modelling customer churn using segmentation and data mining*, 2014.

[30] Perlich C. & Provost F. & Simonoff J. S., *Tree induction vs. logistic regression: A learning-curve analysis*, 2003.

[31] Yoav Freund & Robert E. Schapire, *A short introduction to boosting*, 1999.

[32] Usama Fayyad & Gregory Piatetsky-Shapiro & Padhraic Smyth, *The kdd process for extracting useful knowledge from volumes of data*, 1996.

[33] Wolfgang Stadje, *The collector's problem with group drawings*, 1990.

[34] Sven F. Crone & Stefan Lessmann & Robert Stahlbock, *The impact of preprocessing on data mining: An evaluation of classifier sensitivity in direct marketing*, 2006.

[35] Zhen-Yu Chen & Zhi-Ping Fan & Minghe Sun, *A hierarchical multiple kernel support vector machine for customer churn prediction using longitudinal behavioral data*, 2012.

[36] Fan Li & Juan Lei & Ying Tian & Sakuna Punyapatthanakul & Yanbo J. Wang, *Model selection strategy for customer attrition risk prediction in retail banking*, 2011.

[37] Zhao Jing & Dang Xing-hua, *Bank customer churn prediction based on support vector machine: Taking a commercial bank's vip customer churn as the example*, 2008.

[38] Yen-Hsien Lee & Chih-Ping Wei & Tsang-Hsiang Cheng & Ching-Ting Yang, *Nearest-neighbor-based approach to time-series classification*, 2012.

[39] Yaya Xie & Xiu Li & E.W.T. Ngai & Weiyun Ying, *Customer churn prediction using improved balanced random forests*, 2009.

[40] Özden Gür Ali & Umut Arıtürk, *Dynamic churn prediction framework with more effective use of rare event data: The case of private banking*, 2014.

[41] Benlan He & Yong Shi & Qian Wan & Xi Zhao, *Prediction of customer attrition of commercial banks based on svm model*, 2014.

[42] Xu-Ying Liu & Jianxin Wu & Zhi-Hua Zhou, *Exploratory undersampling for class-imbalance learning*, 2009.

Ringraziamenti

Desidero ringraziare i Professori Andrea Pietracaprina e Geppino Pucci per avermi dato la possibilità di lavorare su un progetto innovativo e interessante, guidandomi con pazienza e attenzione nel percorso della tesi.

Voglio ringraziare Everis Italia per avermi permesso di lavorare su un progetto di scala nazionale, durante il quale ho imparato molto, sia per quanto riguarda la parte tecnica, sia su come funziona realmente un'azienda.

Ringrazio di cuore l'ing. Massimo Ferrari, che ha seguito il percorso della mia tesi dall'inizio alla fine con costante attenzione, guidandomi nella realizzazione di questo interessante progetto. Grazie davvero per la tua disponibilità.

Desidero ringraziare il dott. Riccardo Panizzolo per avermi insegnato molti strumenti importanti relativi all'analisi dei dati e applicati ad una realtà aziendale.

Ringrazio di cuore Martina per avermi aiutato nella realizzazione di questo progetto durante la mia permanenza in Everis, condividendo con me buona parte del suo tempo e facendomi sentire sempre a mio agio, sempre pronta ad aiutarmi. Alla fine, tra i vari imprevisti e i pranzi insieme, ne abbiamo passate tante insieme. Grazie Marti!

Voglio inoltre ringraziare tutto il team di Everis per il tempo passato insieme. In particolare, ringrazio Alessandro, Marianna, Luca, Quan e Silvia.

Con grande affetto desidero ringraziare tutta la mia famiglia, mamma Nicoletta, papà Stefano e mio fratello Nicola. Grazie per avermi permesso di studiare e per tutto il sostegno e amore che mi avete sempre dato. Vi voglio bene.