Sequential User Retention Modelling

HELDER MARTINS

Master in Machine Learning
Date: April 16, 2017
Supervisor: Hedvig Kjellström (KTH) and Sahar Asadi (Spotify)
Examiner: Patric Jensfelt
Swedish title: Detta är den svenska översättningen av titeln
School of Computer Science and Communication

Abstract

English abstract goes here.

Sammanfattning

Träutensilierna i ett tryckeri äro ingalunda en faktor där trevnadens ordningens och ekonomiens upprätthållande, och dock är det icke sällan som sorgliga erfarenheter göras ordningens och ekon och miens därmed upprätthållande. Träutensilierna i ett tryckeri äro ingalunda en oviktig faktor, för trevnadens ordningens och och dock är det icke sällan.

Contents

Chapter 1

Introduction

This chapter will be dedicated to introduce the topic of this thesis project, the motivations on why it is relevant to both academia and to the industry, the current context where this project is being developed on, the hypothesis that shall be explored, the methodology used to verify it, and the contributions of this work.

The rise of streaming services has in recent years revolutionized the way customers get access to digital content. The traditional model of media ownership, even though it still represents a significant share of revenue, is continuously losing ground to a new right-of-access model where users get access to content ether by paying a monthly subscription or by being exposed to advertisement. The customer base of said services is steadily growing, and with it the amount data collected tracking how they interact with the provided digital media. This information is of high-value to any provider who expects improve the user experience, increase its total number of clients and also avoid losing the current ones to the competition.

One application of a data set like this that has received attention from the industry is to predict the clients that are more likely to leave the service in the near future and anticipate it by performing a set of actions with the goal of avoiding it from happening. It is predicted that there is a set of features which are highly correlated with their desire of abandoning the application, and thus a model could be trained to leverage this information and classify users based on the probability of that event occurring. Such models are called *churn predictors*, and will be a central subject on this thesis project. Another interest will be to evaluate which set of features among the available ones actually correlates with churn and extract from it actionable insights that can be suggested to the provider as a form of improvement to their application, a task made harder by the extremely large number of attributes that is commonly captured.

To do Add temporal features paragraph (1)

The user behaviour data and the required computational resources were provided by *Spotify*, a well known music streaming service that joined this project in a tutoring partnership with the student and the university. Creating a suitable data set for training our proposed models is also part of the project, and it was performed by first exploring the data at our disposal and identifying the features that highly correlates with churn. While the available data is mostly related to the music domain, the methodology of this project and the conclusions reached can also be extended to other domains like video streaming with minor modifications.

1.1 Streaming services

The last decade has witnessed an overwhelming increase in popularity in all kinds of streaming services around the globe. Applications like Netflix and Spotify have changed the industry by offering a legal and affordable way of accessing content through a subscription-like contract instead of the classical pay-by-content commonly used by traditional media providers. This change in business model has proven to be successful by the sheer amount of customers that the most popular service providers have nowadays. Netflix for instance has reported to its shareholders a user base of 93 million worldwide at the end of 2016 [1], while Spotify recently reached the amazing feat of 50 million paying subscribers, 20 million of that only in one year [2]. Even though bureaucratic challenges like the dispute between the service providers and the labels regarding the value of licenses are still on debate, the adoption of this new model by consumers indicates no interest of going back to the old ways of getting access to digital content, like through physical or downloadable pay-by-content media.

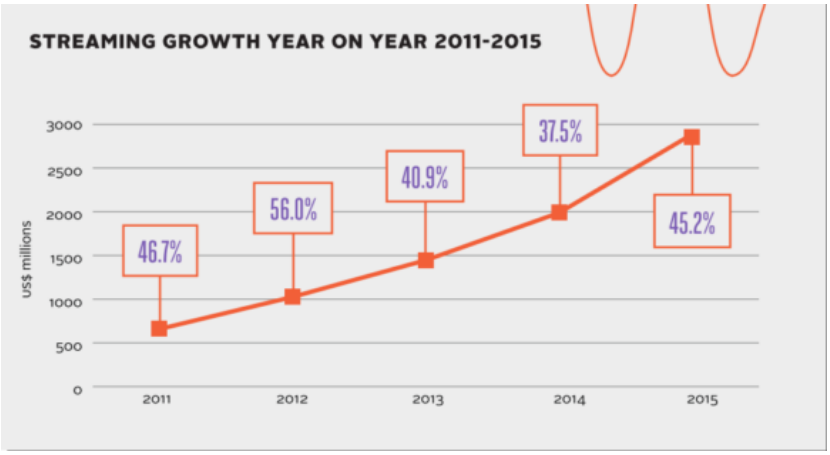

Figure 1.1: IFPI's streaming growth report

Music streaming follows this trend of growth on the latest economical reports. Over the five year period to 2015, the observed revenue from the industry grew four times to US $2.89 billion as can be seen in Figure 1.1 from the IFPI organisation [3]. The increase in the number of paying subscribers is also of notice, seeing an increase from 8 to 68 million people over the same period. This new model also helped bring some countries where licensed music market was losing ground to piracy like China and Mexico, and it corresponds to around 20% of the industry's revenue in the top five markets in the world.

1.2 Spotify and the growing competition

Spotify is the top music streaming service globally in terms of number of users, totalling 100 million of which 50 million are Premium paid subscriptions [2]. The content is delivered through applications available on several different operating systems like Windows, macOS and Linux, as well as devices like iOS and Android tablets and smartphones, an example is shown in Figure 1.2. Users may access over 30 million songs available in its catalog through two methods: a free membership where the client is exposed to advertisement, and a Premium paid subscription with additional features like offline downloads, increased music

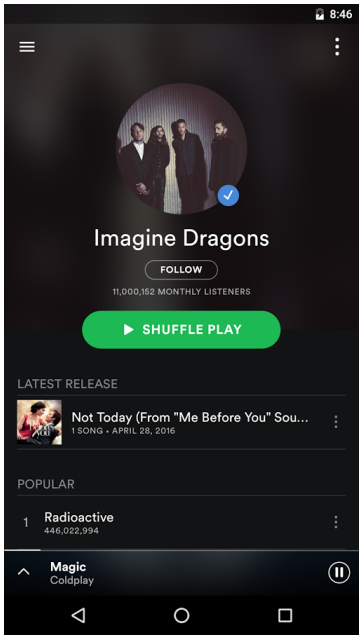

Figure 1.2: A screenshot of Spotify's Android app

quality and unlimited number of song skips. Spotify also is well known for its music recommendation subsystems like Discover Weekly and Daily Mix that provides playlists tailored to each user based on their listening history. The service is available in most of Europe, as well as in Australia, New Zealand, most of the Americas and several Asian countries.

The promise of growth in the area however has attracted well known brands to the dispute like Apple and Google, and their growth, while still not up to par with the top players, has been steady and may be significant in the following years [3]. The market is becoming saturated with competition, and any contender to the top position in the music streaming area must strive to keep its current user base at all costs. The loss of a customer, be it for the competition or not, is called in the industry as *churn* and is commonly seen as a metric that should be minimized in a customer relations framework (CRM) of a service provider. For that goal, an important step is to identify prematurely the candidate churning customers and to leverage actions that may avoid them from leaving, and this is commonly achieved by making use of the wealth of historical information that digital providers have about their users.

1.3 Churn prediction

Churn prediction, the task of identifying the most probable churning customers of a service provider, is an established topic in the literature and has seen a considerable amount of research specially on the mobile telecom industry [4] [5] [6]. Several classical techniques for predicting churn like decision trees and logistic regression have been used on recent work [7], however most of them make use of user behaviour data at a single point in time, normally the most recent one when the dataset was created. The user is commonly represented by summarizing or aggregating his behavioural feature values over a fixed time window, thus losing any latent sequential information that might be contained in the data set. This is a simplification that eases the burden of training a predictor model since the customer

data is commonly high-dimensional and difficult to work with, nonetheless latent attributes, when used properly, have been shown to improve significantly a predictor's performance for industries like mobile telecom [8].

Intuitively one can think that latent factors hidden in the temporal axis of the user behaviour data could yield a better prediction accuracy when comparing to a model which leverage a single and static point in time. For instance, a user that is gradually reducing its consumption of the service over time can be easily thought of a prospect churner. While this intuition is trivial to come up with, we are also interested in learning strong correlations between temporal aspects of the data and the churn rate which are not as clear to the eye. Our hypothesis is that making use of the properties hidden on user behaviour data over time, a churn rate predictor could improve its accuracy greatly when compared to methods which are static on time.

1.4 User retention

A churn rate predictor which can identify possible churners accurately is just a trigger on the user retention process that can span several different stages. One of these stages is to identify what actions can be taken by the service provider as to avoid the user abandoning the service. An important problem to solve beforehand is to identify which features of the data have a strong statistical correlation with the churn rate. With that information, the providers could for example set up automated actions as to influence the value on these features. Learning which are the most important features is also of relevance for the task of choosing what data to use for training the predictor models, since commonly service providers have a vast amount of data about the user but only a fraction of that is of importance for predicting churn: training models with a full dataset will commonly introduce error on the system and take a large amount of computing resources to train.

1.5 Objective

The goal of this project will be to research different models for churn prediction that can make use of the temporal aspect of user behaviour data at its fullest. Our hypothesis is that latent factors hidden on usage patterns may increase the accuracy of state-of-art predictors which considers only the state of the data at a single point in time. We shall experiment on different models that are known to perform well on time series data and compare their accuracy using different evaluation methods. Detecting possible churners, while important, does not improve user retention without an associated action performed by the service provider. To derive insights on what can done to improve this metric, a deep data exploration and feature analysis is also a goal of this project. Features shall be correlated to their influence on the churn rate, and feature selection techniques shall be researched and implemented in an attempt to reduce bias on the churn classifiers.

Chapter 2

Background

For a better understanding of this degree project, some concepts are required to be properly introduced. Some of them like the ones present in the Definitions section are mainly application-specific, while others pertains to our choice of methodology and evaluation metrics.

2.1 Definitions

Every service provider has a different definition of what churn is. Some of them use the explicit cancellation of a contract as a indicator of user abandonment, which has the advantage of being straightforward to calculate and having a clear correlation to revenue metrics, while others prefer to explore the user activity instead, which has the benefit of better leveraging his satisfaction level and working with the assumption that a satisfied customer has a lower chance of leaving the service for the competition.

In this degree project, the *user activity* will be used as indication of churn. Even though Spotify has a paid Premium version that could be used as a source of churn labels, a considerable fraction of Free users who generates revenue through advertisement would be excluded from this study. Thus their engagement will be measured instead disregarding for labeling purposes whether they are paying customers or not, however they may be split into different clusters for visualization purposes.

To build up the exact definition of what churn is for Spotify, some concepts need to firstly be introduced, which follows.

Definition 1 *A* session *is defined as the time period the user is engaged into the application, be it to stream music or to navigate through the available content. A session starts when a user interaction trigger is captured (eg. a button click), and the session ends when no such trigger is captured and no continuous activity is being performed (eg. listening to a song) for at least 15 minutes.*

The session is the basic measurement of activity of the user on the service. It incorporates not only the user music streaming behavior but also his interactions with other features of the application like playlist creation and social interactions between other users. Since Spotify is mainly interested in evaluating the user based on its main feature of content streaming, other user interactions were deemed by the service provider as a weak predictor of churn.

Definition 2 *The* consumption time *of a user is the time within a session that he is streaming any service-provided content. A user session is considered* active *if its consumption time is greater that*

zero, and inactive *otherwise.*

Active and inactive sessions are more constrained definitions that better reflect the goals of the service provider. Even though there may be a highly engaged user that makes use of a wide range of features on the application, every session of his will be considered inactive if no content is streamed.

Before we arrive in the definition of churn, a brief introduction to time windows will be given to understand further concepts in this section. These definitions will be thoroughly explored on chapter 4.

Definition 3 *The* observation window *is the time span used to observe user behavior and to train the predictor models. It is always followed by the* prediction window *which is exclusively used to predict whether a user in the observation window is going to churn in the future or not.*

Definition 4 *A user is defined to have* churned *if he has at least one active session in the observation window and no active session in the prediction window.*

Note that as mentioned before this definition is not concerned whether the user is a paying customer or not, but only if he is actually making use of the service. For example, a user may be subscribed to a Premium paid account and still be considered to have churned if no activity was registered in the prediction window. However, it should be noted that empirical observations by the service provider suggest that there is a significant behavioral difference between the two classes of users that will be taken into consideration when the experiments are performed.

2.2 Feature Selection

To do add feature selection techniques (Information gain, L1-norm, Recursive Feature Elimination, Variance Threshold...) (2)

2.3 Predictor Models

2.3.1 Logistic Regression

Logistic regression is a linear regression model used to predict the probability of occurrence of a categorical variable, like the churning and non-churning labels. First developed by mathematician David Cox [9], this technique has been used in wide range of domains like medical fields, marketing and economics **To do** add refs (3), and is as of today one of the most used models for predicting the churn rate of customers on the telecom industry [7]. Despite its name, logistic regression is used mostly as a classification algorithm, since its output is a probability score that together with a threshold can map the input to one of the supplied labels (churn and no-churn, for example).

The premise of logistic regression is that the output variable y can be modeled as a linear probability function dependent on a set of input feature values $\boldsymbol{x} = [x_1, x_2, ..., x_n]$ through a set of equations that follows:

$$p(y = 1|\boldsymbol{x}) = f(t) \tag{2.1}$$

$$f(t) = \frac{1}{(1 + e^{-t})} \tag{2.2}$$

$$t = w_0 + w_1 x_1 + ... + w_n x_n = \boldsymbol{w}^T \boldsymbol{x} \tag{2.3}$$

where y is the true output variable that can take the values 0 or 1 in a binary classification problem. t is a linear combination of the input $\boldsymbol{x}$ with a trainable weight vector $\boldsymbol{w}$. $f(t)$ is the logistic function that gives name to the model. This function has the property of "squashing" any real input to a value between 0 and 1, and thus it can be interpreted as a probability.

To do improve LR description (4)

2.3.2 Long Short-Term Memory

To do add models (Decision Trees, Random Forests, LSTMs, CNNs...) (5)

2.4 Evaluation Metrics

2.4.1 Confusion Matrix

A *confusion matrix* (also called contingency table) is a table layout representing the performance of a classifier's output, judging by its predictions against the actual true values. In a binary classification problem, the confusion matrix is a 2 x 2 table where commonly the rows represent the true labels while the columns the predicted classes. Each cell contains a count of how many samples were classified on that category, and the values in the diagonal represent the correctly classified samples (if the features are ordered). Figure 2.1 depicts a sample confusion matrix for a binary classifier.

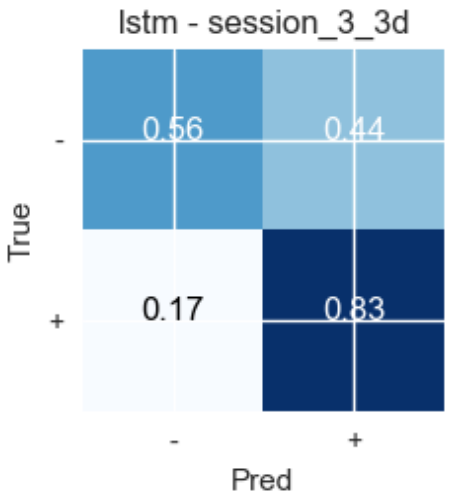

Figure 2.1: An example of a normalized confusion matrix

The confusion matrix is an excellent visualization tool to estimate the performance of a model. The values that should be maximized, the *true positives* (TP) and *true negatives* (TN), represent the samples that were correctly labeled as possessing the feature of interest or not, respectively. On the other hand, the *false positives* (FP) and *false negatives* (FN) are the misclassified samples where the algorithm predicted that the feature was present while in truth it was not and vice-versa. In this work, the positive class will always represent a churning user, and it follows that the negative class represents the non-churners.

2.4.2 Classification Accuracy

Several different metrics can be derived from the confusion matrix table. The *classification accuracy* (CA) of a model is a common metric that corresponds to the fraction of the correctly classified samples on the test set, and can be calculated as follows:

$$CA = \frac{TP + TN}{TP + TN + FP + FN} \tag{2.4}$$

While trivial to understand, this metric may lead to erroneous conclusions when class imbalance is present in the test set, which is a common occurrence on the churn prediction domain. For example, if 9 out of 10 users of a dataset are non-churners, any classifier that simply outputs a negative class for all samples will result in an accuracy of 90%, however its ability of detecting churners is non-existent. For a service provider, detecting churn cases is always more important than detecting the loyal users, and this metric by itself cannot represent this goal [10] [4].

2.4.3 Precision, Recall, Fall-out and False Alarm

To address the class imbalance problem of the classification accuracy, others metrics are also commonly used. The *positive predictive value* (PPV, also known as precision) is the proportion of the samples labeled as positive which are true positives, and describes the performance of the algorithm. The *true positive rate* (TPR, also called sensitivity and recall) of a model corresponds to the number of correctly predicted positive samples divided by all positive samples. *True negative rate* (TNR, also called specificity and fall-out) is the number of correctly predicted negatives divided by all true negatives. The *false positive rate* (FPR, also called false alarm ratio) is the probability of receiving a false positive as output of an experiment, and is calculated by dividing the number of false positives by the total number of positive samples. The *F1 score* (F1, also called F-score or F-measure) builds upon precision and recall by returning an harmonic mean between these two metrics.

$$PPV = \frac{TP}{TP + FP} \tag{2.5}$$

$$TPR = \frac{TP}{TP + FN} \tag{2.6}$$

$$TNR = \frac{TN}{TN + FP} \tag{2.7}$$

$$FPR = \frac{FP}{TN + FP} = 1 - TNR \tag{2.8}$$

$$F1 = 2 \times \frac{PPV \times TPR}{PPV + TPR} \tag{2.9}$$

Depending on the distribution of classes of the dataset, it is often difficult (although desirable) to maximize both metrics at the same time. A compromise must be reached that achieves the best trade-off between the two, which is commonly a business decision. For a music streaming service like Spotify, maximizing sensitivity is preferred due to the costs associated with a churning user, however a reasonable specificity is also a metric that should be strived for. **To do** Check with Sahar (6)

2.4.4 Receiver Operating Characteristic

The *receiver operating characteristic* (ROC) is a visualization tool that plots the relationship between the true positive rate (commonly the y-axis) and the false positive rate (the x-axis) of a binary classifier system. The curve is drawn by selecting different parameters of a model or levels of threshold for the decision boundary between the positive and negative classes. For example, when the output of a classifier is a probability value (like in logistic regression), different thresholds can be chosen to decide whether a user is a churner or not, depending if the goal is to minimize FPR or maximize TPR. An example of a ROC curve can be seen in Figure 2.2.

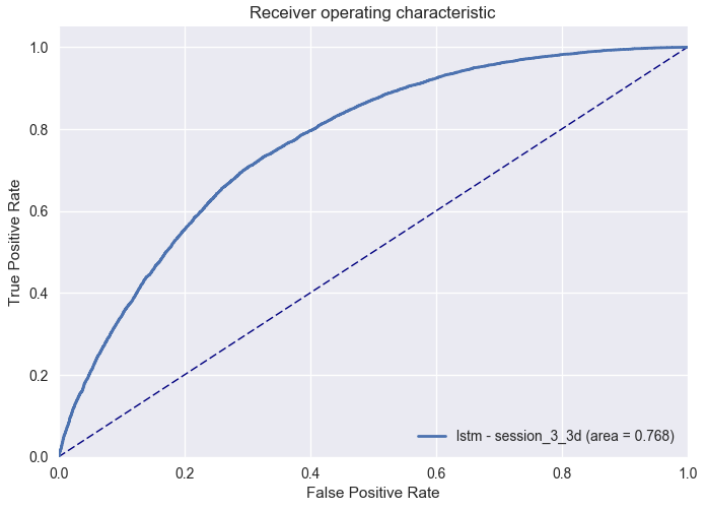

Figure 2.2: An example of a ROC curve

A good classifier would score values close to the upper-left corner of the plot, where the point (0,1) represents a perfect classifier with 100% TPR and 0% FPR. On the other hand, an algorithm that outputs a curve alongside the diagonal where TPR and FPR are almost the same at different threshold levels can be considered close to a random guess, like the flip of a coin. A classifier would underperform if its scores are closer to the bottom-right corner of the plot, however this result can always be mirrored by updating the model to simply invert the positive and negative labels of the classified samples.

ROC curves can be used to compare the performance of different models by measuring the *area under the curve* (AUC) of its plotted scores, which ranges from 0.0 to 1.0. The greater this area, the better the algorithm is to find a specific feature. Moreover, models with an area close to 0.5 can be assumed to perform not much better than random guess, since this is the total area under the diagonal line.

To do Add eval methods (Lift chart, Top Decile Lift (TDL)...) (7)

To do Talk about precision-recall curves and PR AUC (8)

Chapter 3

Related Work

There is considerable amount of work done on the churn prediction realm. The most common topic that is addressed by the literature is evaluating different types of predictor models trained on a domain-specific dataset, like the telecom and financial industries for example. Understanding what techniques are commonly used on the field is deemed crucial for the success of any churn rate predictor model.

The use of data represented as a time-series for predicting churn, although of high interest to this project, is a rather under researched topic as of today, the user features being commonly aggregated over a time window on the whole observation period. Recently though some interest has been sparked on the literature about trying to leverage temporal aspects of the user behaviour by representing the data as a sequential input, and some examples similar to the task at hand, while not abundant, can nevertheless be found.

Other research papers focus on answering how different properties inherent to the churn prediction problem affect the performance of classifiers, like for example how to deal with unbalanced data sets, how the evaluation procedure should be performed and how far back in the user history should models be trained on as to obtain a proper balance between accuracy and processor usage.

This chapter will introduce the literature on the fields of interest for this project, and will focus mainly on the following topics:

- Identifying the state-of-the-art for training, modeling and evaluating churn predictor classifiers, the algorithms being used, the methodology for the experiments and the observed results.
- Researching current techniques for creating models that leverage the properties of time-series data, applied for churn prediction (if any) or similar domains where we can map our problem to. It shall also be studied common features of sequential data as to better understand how to best exploit its properties as to improve the accuracy of our models.
- Understanding prevalent characteristics on data sets used for churn prediction as to improve the performance of models and also to avoid common pitfalls when training and evaluating their accuracy.

3.1 Churn Prediction

Identifying which users are most likely to churn is an important step in any customer relationship management (CRM) framework interested in improving user retention. It has been shown that several economical values are directly correlated with the customer lifetime metric that can be crucial in a competitive market, like how the costs of acquiring new customers surpass those of retaining existing ones and how loyal users usually spend more and can bring in new customers [8]. Thus, churn rate prediction is an established topic in the literature and has seen studies being performed on a wide range of domains like the telecommunication industry [5][6][4], social multiplayer games [11][12] and community-based question answering services (CQA) [13][14].

The very first aspect of the study that any researcher has to take into consideration is the formal definition of what churn is, which depends on the domain that the data set belongs to. [15] divides the definition into three types: *active* where the contract is officially terminated, *hidden* when the customer is inactive (not using the service) for a significant amount of time, or *partial* when the customer is not using the service at its fullest, opting for a competitor instead. Subscription-based services like mobile companies [5][4] and massive multiplayer games [11] commonly utilizes the active churning definition caused by the explicit termination of the agreement. Social games [12][16] and CQA services [13][14] however usually model the user interest by measuring its level of activity and engagement since no contract is formally established, opting for the hidden churn definition instead.

Even though Spotify has a contract-based Premium service, this project is interested in the clients who are inactive for some time, whether they are paying customers or not. Delving into the hidden churn definition, one could ask what is considered a "significant amount of time" as to judge a user to have churned or not, and different domains have distinct approaches depending on the way the interaction between user and service takes place. For example, for a mobile network like the one studied in [6] this definition can span months of inactivity since customers on that market are commonly more loyal, even though this paper focus on pre-paid accounts with "soft" contracts. On the other hand, rapid-changing markets like free-to-use services as social games and CQA sites prefer to define the churn period as small as possible (commonly days or weeks), since the absence of an agreement makes it easy for customers to switch for the competition, having the service the obligation to act faster if any sign of loss of users is identified [6][16]. On [14] the focus is set on new users so the the churn period is explicitly set to be their first week of activity. [13] however focus on both new and experienced users by considering the first days and posts (questions and answers) as parameters on the models trained, evaluating each independently.

Current literature for prediction of churn rate uses a wide range of machine learning technologies, such as decision trees [13][4] [17] [6], logistic regression [8], neural networks [12] and its convolutional variant [18], random forests [14] and support-vector machines [19]. The number of techniques is vast, however it can be mentioned that current research suffers from a difficulty to reach insights that can generalize well for other areas, since the applications that the data sets belong to are domain-specific, greatly influencing how the users interact with the service. Interpretability is a key component in any application since the end goal is always not only to detect the users most likely to defect but also the reasoning behind their departure, as to plan reasonable actions to increase retention. With that in mind, even though some models like weighted random forests [10] and neural networks [12] offer a higher accuracy for some domains, the difficulty in getting key drivers behind the con-

sumer's behaviour in some "black-box" type of models have led to researchers opting for a more interpretable technique like logistic regression which is commonly a close second place in terms of performance [12] [14]. For that reason, decision trees and logistic regression are as of today the most popular techniques for churn prediction, yet a consensus of the best model cannot be reached by researchers as stated by [7].

The telecom industry is by far the most well researched area for predicting churn, and a proper review would be incomplete without mentioning some examples. In [4], two popular models for predicting user churn were empirically compared: decision trees and logistic regression. By making use of the data set provided by a mobile operator, two models were created by independently training and selecting their best performing versions. Evaluation was executed by comparing the AUC of their ROC curves, as also their Lift score and overall accuracy. The conclusion is that decision trees consistently perform better when compared to logistic regression models, and thus should be a preferred choice. In [5], churn classification was performed on users in a rather unique way. The Gentle AdaBoost boosting algorithm was used alongside a logistic regression base learner as to train a model to do the separation between the churning and non-churning classes. However, one further step was performed where the users were split into two clusters based on the weights assigned by the boosting algorithm. One of the clusters was identified to have a considerably larger churn rate than the other, and thus another logistic model was trained using the clusters as labels, and its performance evaluated using ROC curves. The idea behind this approach is to use a model to mark users who have a high risk of leaving the service (and be the focus of user retention actions), instead of labelling them as churners and non-churners directly. On [6], billions of call detail records from a mobile phone operator was the focus of a churn prediction study and feature analysis. The initial data set was filtered to the calls made by 100.000 subscribers during a 6 month period. A brute-force approach to feature engineering was then performed to create 12.914 out of the initial 10 features by combining every feature from each of the manually split 8 different groups. Feature selection is then performed in two distinct ways: first, an individual Student's t-test score is computed for each individual feature to evaluate how well it can differentiate between the churners and non-churners sets. Second, a tree-based method was used as to estimate the the accuracy of a joint classifier by adding features one at a time. Features are then ordered by their statistical correlation with churn, and the top 100 features were selected to train several different classifiers. Evaluation was performed mainly by comparing the AUC of each model, where AdaBoost using a decision tree classifier performed the best, followed closely by a logistic regression model.

The rapid increase in Internet popularity has spawned a plethora of new services in the last decade, and thus a wider range of domains like CQA sites and gaming applications are being researched for churn prediction. In [14], an explorative study was made on the Yahoo! answers website as to discover an efficient churn predictor model and also the features that correlates with the user leaving the service, however differently from other works this paper focused on new users with less than a week of activity on the service. Features were grouped into Question, Answer and Gratification categories, and were used for training several different classifiers. For this dataset, random forests performed the best with logistic regression as close second. Features were also ordered by the amount of information gain that they provide, and the number of questions and answers are the top features on that regard (inversely correlating with churn), followed by the period of time the user is active and gratification features for answers given and questions made. In [13], the younger CQA service Stack Overflow was the focus of a study on user behaviour characterization and churn rate pre-

diction. An extensive data exploration was made as to correlate features to the chance of a user leaving a service, and with those insights classical modelling techniques were used, where the best performing one was a decision tree. The approach used for extracting and categorizing features (temporal, frequency, gratitude, etc) and the insights that follow the study (like the importance of temporal features for predicting churn) is of high value and can be mapped to concepts on a different domain like a music streaming service with minor modifications. In [12], players responsible for generating the most profit on two casual social games were the focus of a study on churn analysis and prediction. Formal definitions for active players and churning were made, and the problem was defined as a binary classification task where users were labelled as leaving the service or not in the following week of when the models were trained. Four different models were then trained, and a neural network classifier obtained the best area under the curve (AUC) score, with logistic regression as a close second. The formal definitions of "churn" and "activity" can be applied to a music streaming service in a similar way, as the evaluation of performance using a series of receiver operating characteristic (ROC) curves. In [11], the activity log of a subscription-based multiplayer online game was the source of an experiment focused on comparing two different methods for predicting the users with a higher tendency to leave the service, a theory-driven and data-driven approaches. For the modelling, an ensemble method was used with several different classifiers, where the choice of model was made based on whether the cluster of the data set (found with K-means) that will be used to train the classifier has a significant proportion of churners or not. Even though the performance of the data-driven model was superior, the difference was negligible when taking into consideration the complexity of the models and also its interpretability. Moreover, if marketing resources are constrained the theory-driven model was considerably superior for the first 40% of users on the lift chart.

3.2 Sequential Modelling for User Behaviour Data

Even though there is currently a wealth of research on churn prediction techniques, most of them treat the user behavioral features as static points in time by aggregating or summarizing their values over a time window [20]. It is a simplification that eases the hassle of training models with a data set which is commonly high-dimensional and difficult to work with, however information that could be used to improve the predictors' performance is lost in this process [8]. For example, there is no information as to when over the time window being analysed a user decides to churn, it could be that all defectors are grouped into a period where an external event (like the end of an offer) has taken place. Environmental variables, that is those that varies through time but are common to all users, are also omitted on static models, leaving it to the experts the task of manually integrating these features on their analyses. Newer models like [13] attempt to integrate some temporal data by manually engineering features related to time (like the duration of an event, for example), however since each user is represented as a sample in a single point in time the sequence of data transformations is inherently lost in this process.

It is natural to think that the human behaviour evolves through time. Our opinions and desires changes in accordance to the stimuli that we receive from the world around us, and summarizing our persona by looking at a single point in our life is an oversimplification that does not make justice to the path that leaded us to where we are. Dealing with temporal data is by no means trivial however, and to leverage fully its properties some careful considerations about this type of data must be done. On [21], an overview of the challenges on creating

models that make use of the time component on data sets are presented to the reader. Also, since hand-crafted features are generally domain-specific and difficult to create, a review of the current research on unsupervised feature learning applied for time-series data is the focus of this paper. From the challenges, it is worth of mention the uncertainty that there is enough information to understand the underlying process, its non-stationary characteristics like mean, variance and frequency, and its common high dimensionality and noise. The right representation is deemed then crucial for any model to be successful in its goal. Several different techniques currently being used for unsupervised feature learning are then presented in details to the reader, where it is worth of quick mention the conditional Restricted Boltzmann Machine (cRBM), Recurrent Neural Networks (RNN, with is Long Short-Term Memory extension), the Gated RBM, the Time-Delay Neural Networks, and the Space-Time Deep Belief Network. Finally, several classical time-series problems are reviewed alongside the best models currently being used for each domain.

The hypothesis that sequential patterns hidden on temporal data can improve the accuracy of predictor models is not an uncharted territory however, and has being recently explored by researchers on similar domains as churn prediction that could be mapped to with minor modifications. Sparked by positive results of recurrent neural networks on domains like speech recognition [22] and video classification [23], some papers attempt to recreate similar deep architectures on their own prediction problems. On [24], a technique was presented as to predict the next event and its timestamp on a running case of a business process (a help desk process, for example). Three different LSTM architectures were experimented on, one for each of the problems being tackled: estimating the next activity and its timestamp, all the remaining activities in a use case and the remaining time of a process. This technique could be used for churn prediction by interpreting the user interaction with the service as actions, and "churn" could be an event that may or may not exist in the process chain. The LSTM architecture was also present on [20] alongside a quantile regression (QR) model as to predict which of the past buyers of a store chain will return after acquiring a product on sale. The hypothesis is that a performance gain can be achieved by combining models that are know to perform well on both temporal and aggregate features, respectively. It was empirically shown that a mixture of experts approach between these two techniques can reach a significant mean-square error improvement when comparing to any of the models when evaluated independently.

Deep architectures like LSTMs could theoretically be applied in a similar way for the task of predicting which users are more likely to abandon a service, however the lack of research using techniques like these suggests that this mapping may not be as easy as it seems, nevertheless some examples can still be found in the literature. Inspired by results obtained on image recognition [25], a convolutional neural network (CNN) model was used as a churn predictor on the work by [18]. On it, users were represented as 2-dimensional images where columns are the tracked features and rows are days organized in sequences. Two different CNN models were used, and both outperformed a baseline decision tree model. On [8] the researchers propose a new data set generation framework that can better leverage the temporal aspect of user behaviour data for churn prediction models. The hypothesis being tested is that by boosting the data with features on different time periods instead of only the last one, a significant performance gain can be achieved. The main framework of this study is called Multiple Period Training Data (MPTD), which basically consists of aggregating to each user their features at each predefined time step along the time range of the data set, alongside their churn labels at each point in time and also environment variables which varies through

time but are common for all users. This framework was tested against a classical static data set, demonstrating a significant improvement (by *p-value* comparison) on AUC and TDL for both logistic regression and decision tree models for predicting if the user is going to churn on the next test period or nor. Another experiment was focused on predicting churn several periods on the future, and for that a common survival analysis method called Cox regression was used as a benchmark for comparison against logistic regression models trained on MPTD. It was empirically concluded that using several independently trained binary classifiers, each trained on a different parameter to the "churn within δ periods" (WδC) feature, a significant performance gain can be achieved when comparing their respective AUC and TDL.

3.3 Evaluating and Training Churn Prediction Models

Data sets from service industries often have a common characteristic that may be problematic while training models, which is the uneven distribution between the churning and non-churning classes. It is a common scenario for the number of non-churners to heavily outweigh the number of churning samples on real data sets. Class imbalance is thus a recurrent concern in almost all domains, however research in this area lacks the proper attention. [10] aims to solve this mystery by focusing entirely on how the uneven class distribution affects the performance of several different classifiers. The performance boost estimated to be received by using a cost-sensitive learning algorithm (where false negatives are assigned a greater cost than false positives) is also evaluated, and it has taken the form of a weighted random forest model which was compared to other classical baseline models like logistic regression and random forests. Under-sampling, where fewer samples from the majority class are incorporated in the training data as to artificially change the distribution between the labels, is proved to significantly improve the performance of the underlying models, however the exact ratio between churners and non-churners is confirmed to be case-dependant, not necessarily being the even distribution the perfect choice.

To deal with the problems brought by the unbalanced distributions mentioned above, proper evaluation metrics must be used as to avoid the predictors to be biased towards the majority class. It is a well know fact that the accuracy of a classifier is not an appropriate method when comparing the performance of different models [26], and so no paper reviewed used this metric by itself. The overwhelming majority of the literature abides by the use of AUC as a proper metric for predicting churn rate, [17] [8] [6] [5] to name a few. [10] delves into this topic and defends the use of AUC as a proper metric, while adding the cumulative gains chart as to graphically represent the percentage of customers has to be targeted to reach a percentage of churners. The F-score is also sometimes added as a metric of interest, normally in conjunction with AUC [14] [6]. In respect to graphical representations, ROC curves and lift charts dominates the literature as appropriate methods, and it is present in works like [5] and [10].

Another question that can be made is how far in the user history models must be trained on as to achieve the best trade-off between accuracy and computational burden. It is intuitive to imagine that user actions far in the past exerts little influence on predicting whether the user is going to churn or not in the near future, however the exact threshold that should be used is not trivial to find. [17] attempts to answer this question by experimenting with three different models (logistic regression, decision trees with and without bagging) on a newspaper data set consisting of 16 years of user history. The models were evaluated for

their performance by training with data ranging on this interval with a 1-year gap between each measurement. It has been concluded that after year 5 no gain in performance (measured by AUC) is statistically relevant enough to warrant the increase in computational power needed for the training. However, the researchers also question whether this result can be extended for other domains or not, leaving that for future work.

Current data sets from service providers have no lack of user historical information stored, however the probability that all available features have a significant correlation to churn rate is pretty slim. Using too many features may lead to undesirable effects like overfitting and falling into the "curse of dimensionality" problem, reducing the classifiers performance as cited by [27]. It is a common step to pre process the data in a way as to mitigate this risk, but still there is no single method in the literature that dominates the scene. Several papers do a manual subset selection of features based on theories, testing each using baseline classifiers and selecting the subset with the best performance [13][12]. Others works like [11] and [14] use information gain as a metric to classify the features on their expected reduction in entropy. One step further is taken by works like [5] and [6] where full decision trees are built specifically for feature selection. Another more exotic method was used by [18] with an autoencoder to discover which features influenced the churn rate the most.

Churn prediction has been explored in different kind of applications on recent literature, however music streaming services are as of today a quite under-researched area, even though there is a prominent feature that makes it significantly different from other more researched and similar areas like video streaming, which would be the absence of the constant attention factor that video demands. Studying the user behaviour in each application can possibly help optimizing parameters that are difficult to tune without expert knowledge on how the interaction between costumer and application occurs. The work by [28] aims to fill that gap by analysing patterns of usage of Spotify's Premium subscribers on features like session and playback arrival patterns, user behavior on single and multiple devices and favorite times of day for streaming. The main contributions of the study can be summarized as follows: First, daily patterns can be observed on features like session arrival, playback arrivals and session length. Second, there is a high probability of users to continue on the same device for consecutive sessions. Third, users have their unique times of day when they prefer to stream music on the platform. Fourth, a session length can be used as a good indicator of the next session length and also downtime.

Chapter 4

Methodology

In God we trust: all others bring data.

William Edwards Deming

4.1 The Dataset

Every experiment on this projects relies exclusively on the data provided by Spotify. Each day hundreds of terabytes of streaming information are collected through the provider's data pipelines, aggregating customer usage of the application over several different platforms and countries. Since it is technically impossible to train a single model with the data at its entirety, one of the challenges of this project will be to understand the information that is available to us and to choose a subset of features that are most likely to the probability of the user churning. This task can be extremely daunting by itself, so this project will make use of previous private studies performed by the service provider as a guidance on which features to choose.

4.1.1 Data description

The provider's dataset that is going to serve as a source for the creation of our own transformed data is called *Sessions* and contains for each row a set of features for one session and one user. To limit the influence of time cycles in the learning process, users were sampled from three countries which share a similar timezone between each other and encompass a large share of Spotify's userbase, that is Brazil, Mexico and United States.

The Spotify service is provided through a wide range of platforms like through Apple's iPhone or a desktop PC. To minimize the influence that different platforms may have on predicting churn, and also due to higher reliability on collected metrics, the dataset shall be filtered on streams performed on the Android platform, be it on a smartphone or tablet device.

4.1.2 Features description

The *session length* feature is the time in seconds the session lasted. *Consumption time* is the time in seconds the user is streaming content, for example by listening to a song. Since this action is always encompassed by a session, this feature is always at most as large as

the session length. *Skip ratio* is the percentage of streams the user skipped by for example pressing the Next button on the app, and *unique pages* is the number of unique pages like Artist, Album and Discover Weekly the user visited in his streams.

Using the above features as source, 8 features were derived by taking the mean and the standard deviation of each one of them. The correlation matrix between these features can be seen in Figure 4.1.

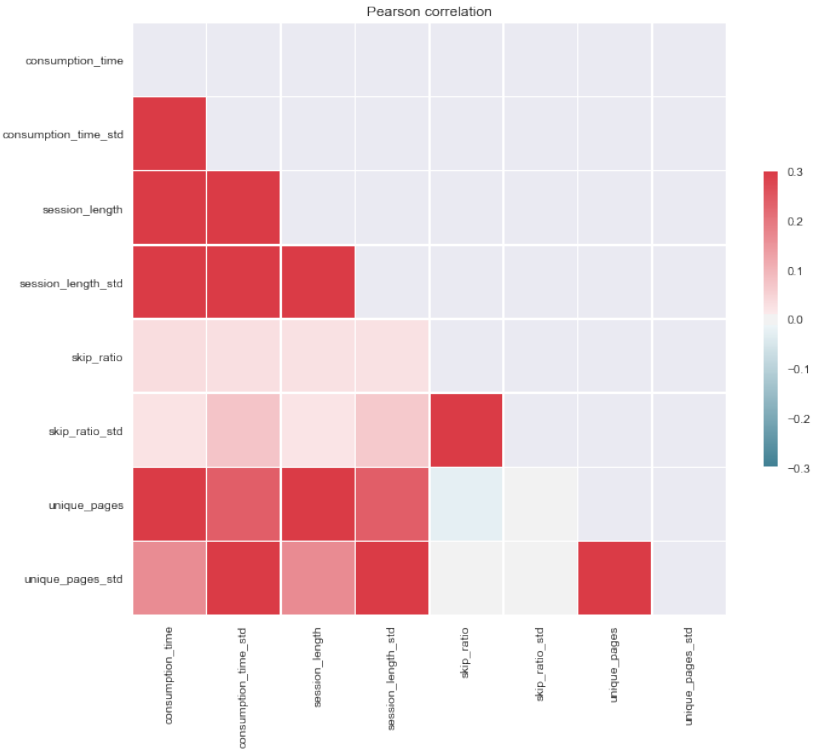

Figure 4.1: Pearson correlation matrix between features

4.1.3 Time windows

In order to predict whether a user is going to churn in the future or not and generate the churn binary labels, the time sequence is split into two time windows. The *observation window* is the time span where the user behavior is tracked, data which will be used for training the predictor models. It is sequentially followed by a *prediction window* that is exclusively used for generating the churn label. The windows can be better visualized in Figure 4.2. The size of each window is a parameter that shall be experimented on, but a standard length of 60 days for the observation window and 30 days for the prediction window were chosen by the provider as values that are suitable for the task at hand.

The user is considered to have churned if he has at least one active session during the observation window, but no session during the prediction window. This means that every user will have a single label for the whole sequence instead of one label per time step.

To do Read [8], [4], [18], [5], improve this section (9)

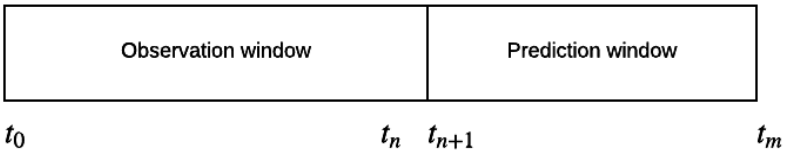

Figure 4.2: Observation and prediction windows

4.1.4 Sequential and Aggregated Datasets

Our source dataset is sequential by nature. Every interaction is logged and stored in tables representing all sessions of users in any given day. However, taking this data as it is is unsuitable to be used for training since most temporal models **To do** check this claim (10) requires (a) that the time step between events to be constant and (b) the it possess the same number of samples in every time step. Since there is no control as to when any given user will interact with the application, some data transformations must be performed.

To address these concerns, the work by [8] and its Multiple Period Training Data (MPTD) were the main focus of inspiration for the creation of our dataset. **To do** write about his approach (11)

First, a constant *time step length* was chosen as to split the observations equally across time. To capture the routine of users between every day-night cycles, a time step of 8 hours was deemed to be a suitable value. Second, the features of users that have no session in any single time step are set to zero as to keep the number of samples constant through time. **To do** Important! Go into more details here, maybe add a figure (12)

In order to train our time-invariant models, another dataset was generated where each user is represented by a single sample with its features aggregated over the whole observation window. The source data is exactly the same as the sequential dataset, however the features are aggregated by taking the mean of values over the window instead of splitting it into smaller time steps.

4.1.5 Data Exploration

(NOTE TO THE READER!! The Y axis values were intentionally hidden in the following plots due to the Non-Disclosure Agreement signed by me. Proper plots will be placed here after the internal review is performed by the company)

Class Distribution

A recurring problem in churn prediction is how the non-churning class dominates over the churners for almost all applications, a problem properly examined by [10]. Figure 4.3 plots the distribution of classes in our generated dataset. In it we can observe that indeed the non-churning class has approximately 5 times more samples than the churners, and even though the difference is significant users abandoning the service can hardly be considered as a "rare" event.

User Behavior through Time

Humans are creatures of habits. Our daily routines follow distinguishable patterns mainly influenced by day-night cycles that are far from being random. To better visualize these patterns, Figure 4.4 plots the feature values for churning and non-churning users over the

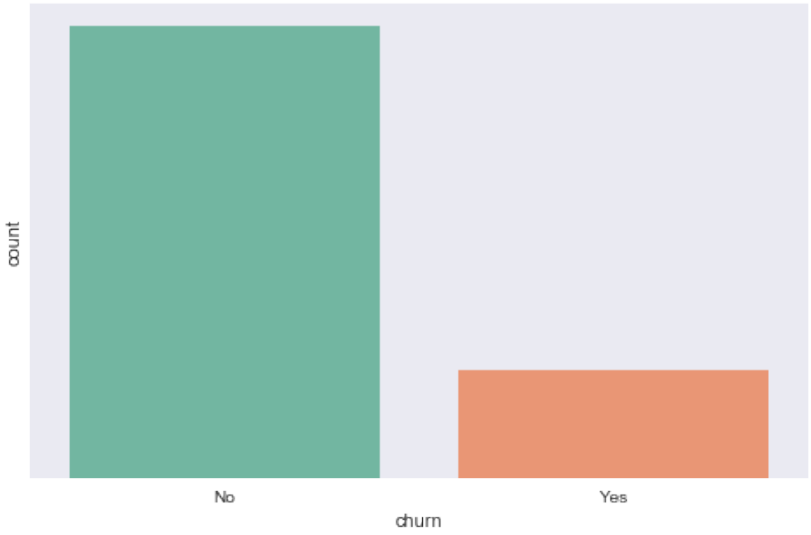

Figure 4.3: Count of users distributed in the churn and non-churn classes

course of the observation window sized at 60 days. The first pattern that can be noticed is that users that are labeled as churning have lower feature values for unique pages, consumption time and session length, and that it gradually decreases over time as users abandon the service.

The second trait that can be observed is that skip ratio values are way more intertwined between the classes when comparing to other features. This is due to the fact that churning users exhibits a higher skip ratio as was shown by empirical studies from the service provider, however since we fill with zeroes the feature values for the sessions where the user did not stream any content we end up conveying the exact opposite: that churning users have a low skip ratio. This suggests that if we want to make use of this feature properly, transforming this feature through a function like the negative logarithm might be beneficial.

Even though plotting the features over the whole observation window let us visualize the overall tendency between the churning and non-churning classes, this representation does not emphasize daily and weekly routines that are present in the data. To address that, the mean and standard deviation of features were plotted by clusters of week days and day-night periods, and this visualization can be seen in Figure 4.5. As expected, higher values can be seen during day time (8h-16h CST) for all features, and also a slightly lower mean on Sundays related to lower usage on this day.

4.2 Feature Preprocessing

While it is possible to use raw features as input to train predictor models, often there is a benefit on normalizing the values using some standard technique. [29] argues that for the backpropagation algorithm commonly used in neural networks, well-behaved features centered around and with unit variance usually provides a faster convergence time. This is mainly due to the way weights are updated in the network: if the input vector has all features with the same sign, the weights can only be increased or decreased together for a single input, zigzagging the values back and forth which turns to be extremely inefficient. In deep architectures, the output a network layer is used as input to the next layers, and for

Figure 4.4: Mean and standard deviation of features through the observation window

the same reasons it is also desirable that this input is centered around zero. Inputs with unit variance will keep the layer's output well behaved and thus speed up convergence.

Therefore, the following transformations were applied to the input before the training process:

$$\mu_i = \frac{1}{N}\sum_{n=1}^{N} x_n^i \tag{4.1}$$

$$\sigma_i^2 = \frac{1}{N}\sum_{n=1}^{N}(x_n^i - \mu_i)^2 \tag{4.2}$$

$$x_n^i = \frac{x_n^i - \mu_i}{\sigma_i^2} \tag{4.3}$$

where N is the total number of samples, x_n^i is a scalar value representing the feature i of the training sample n, feature which has mean μ_i and variance σ_i^2.

4.3 Handling the Class Imbalance Problem

For any relatively successful company the non-churning class heavily outweigh the amount of churners in the dataset. Depending on the estimator, this property can have a negative impact in performance where the models only can only learn how to distinguish the majority class and failing to detect the class of interest. However empirical observations have shown that a better performance can be achieved if the class ratio difference is reduced [10] [30]. Unless otherwise stated, the experiments in this project will be *randomly undersampled* to a class ratio of 1-to-1. This technique, while easy to implement, throws away several samples from the majority class that could be used for learning. Testing different ratios for the class distribution is however one of the experiments that will be performed in this project.

4.4 Cross-Validation

It is a common practice in supervised machine learning experiments to split the data sets into *train* and *test sets* as to obtain an unbiased estimate of the performance of the trained models on unseen data: evaluating a model on the data it was trained on can return an overly optimistic score that does not generalize well for data that is not included in the training, a phenomenon called *overfitting*. Therefore, every score in this project is reported using data that was not present in the training stage.

Almost all machine learning predictor models have arguments that needs to be chosen even before the training process starts, values that are commonly called *hyperparameters*. To fine tune these parameters, we need a method to evaluate how good a set of values are against all other (or a reasonably sized subset of) available options, and for that purpose we need a set of unseen data for the same reasons mentioned before. If the test set were used for such an evaluation, it could not be considered as unseen data anymore since it was used for the purpose of choosing the best model, thus our bias may have "leaked" to the prediction score. Therefore another set is needed to tune the hyperparameters, which is commonly called the *validation set* in the machine learning domain.

Dividing the dataset into three splits however drastically reduces the number of samples that are available for model learning. Another concern is that if the sets are unique, the overall performance metric will depend in a seemingly random split of validation and test sets. To mitigate this problem, a technique called *cross-validation* can be applied. In its most basic approach, instead of having a single validation set, the training set is instead divided into k folds, and at each hyperparameter tuning round the model is trained on $k-1$ folds and validated on the fold that was left out. For each set of parameters, k models are generated, and the final metric can then be an aggregation function over all estimators performance, like the mean. An example of this procedure can be seen in Figure 4.6.

The above technique helps to strengthen our performance metric against biases when selecting the best set of hyperparameters to train our model with. However we still have an unique test set where we evaluate our model to obtain a final score. When comparing different types of estimators, it may happen that one estimator may better learn from the samples that are on the training set while the other can better learn from the patterns contained on the samples of the test set, a split that was purely made by chance. The same procedure used for hyperparameter tuning can be applied for splitting the train and test sets, a technique called *nested cross-validation*. In it, the full dataset is split into k outer folds of training and test sets, and each training set is further split into l inner folds for hyperameter tuning. A diagram of

this procedure can be seen in Figure 4.7.

Even though this technique increases the confidence of our metric scores, there is a price to pay in terms of performance: training all these models may take a considerable amount of time. Spotify however has no lack of computational resources available, so this project makes use of this method. Inspired by [10], all our models were trained using *5x2 cross-validation*, with 5 outer folds for the train/test split and 2 inner folds for train/validation splits.

Figure 4.5: Mean and standard deviation of features through different weekdays and periods

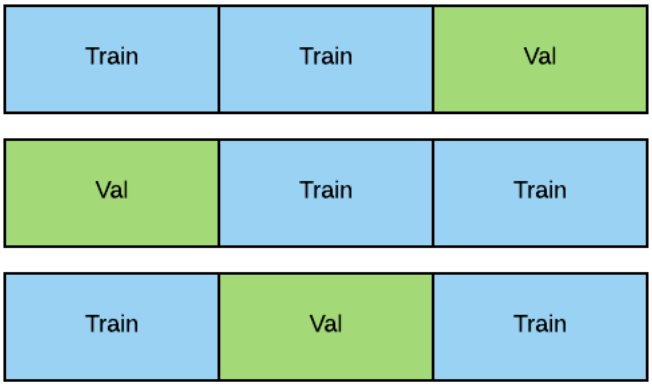

Figure 4.6: 3-fold cross validation split

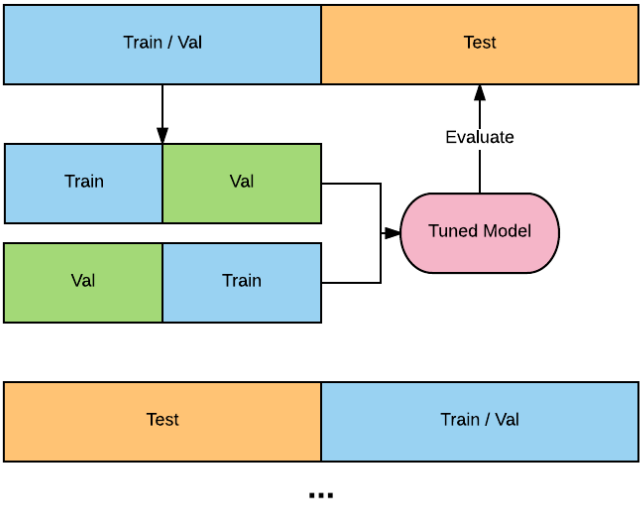

Figure 4.7: 2x2 nested cross validation

Chapter 5

Results

5.1 Temporal vs. Time-invariant Models

5.2 Experimenting on Different Window Sizes

The size chosen for the time windows may have an influence on the performance metrics of our trained estimators that is non-trivial to predict. Even though an intuitive guess can be made for the effect of the change, the set of values that maximizes the accuracy of the classifiers while being compliant to the company's goals and also the available computing resources is *a priori* unknown. This experiment attempts to answer this question by training several LSTM models while changing the observation and the prediction window sizes, and by plotting the best score obtained for each estimator.

5.2.1 Observation Window

Increasing the size of the observation window basically means that more data is going to be used for training. It is expected that with more data, a better accuracy can be achieved, however that comes with the cost of an ever increasing training time. So the question that this experiment is exploring is how far in the user history must we train our models on as to obtain a good trade-off between accuracy and training time, a topic thoroughly explored in the work of [17].

Our experiment involved increasing the size of the observation window roughly by 7 days while keeping the size of the prediction window constant in 30 days, the resulting scores can be seen in Figure 5.1. There is a noticeable gain in ROC AUC and F1 score when increasing the size of the observation window, however this gain visibly diminishes at larger windows. This suggests that there is a threshold where the performance gain is so insignificant that would not justify the increase in time for training these models.

5.2.2 Prediction Window

Modifying the size of the prediction window indicates how tolerant we are for considering a user a churner or not. Small windows are commonly used for services lacking a formal contract between the provider and the customer, which is similar approach used in the works of [14], [12] and [13]. Larger prediction window sizes are often seem in "classical" service providers like the telecom industry where a contract is commonly established, an approach that is widely seem in the churn prediction literature, [6] and [4] to name a few.

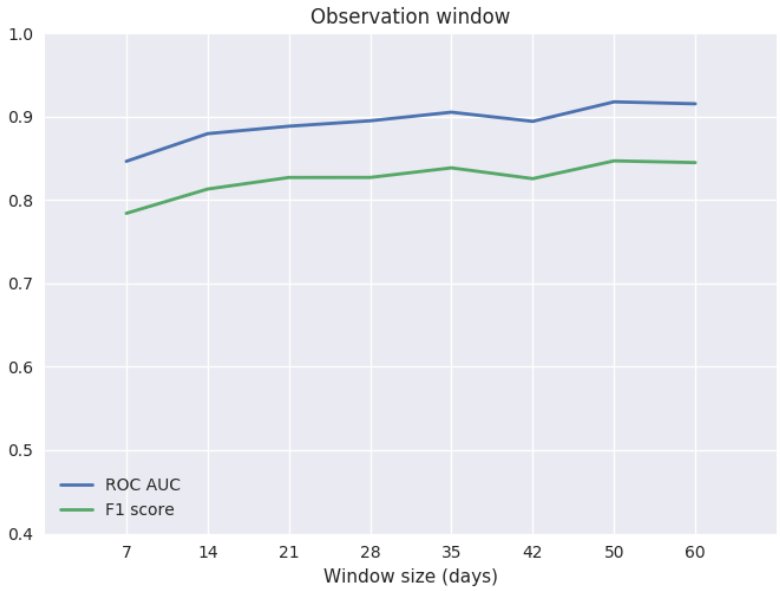

Figure 5.1: Effect on performance metrics for different observation window sizes

This experiment involves changing the size of the prediction window while keeping the observation window size constant at 30 days. It is important to note that smaller window sizes means that a larger share of users will be considered churners, and due to our under-sampling technique this also means that the number of samples used for training will be larger. However the effect on performance metrics hardly seems to be affected by it as can be seen in Figure 5.2. The effect in ROC AUC and F1 scores suffer only minor changes when the prediction window size increaases from 7 to 59 days.

5.3 Effect of Class Distribution

5.4 Experimenting on Different Subset of Features

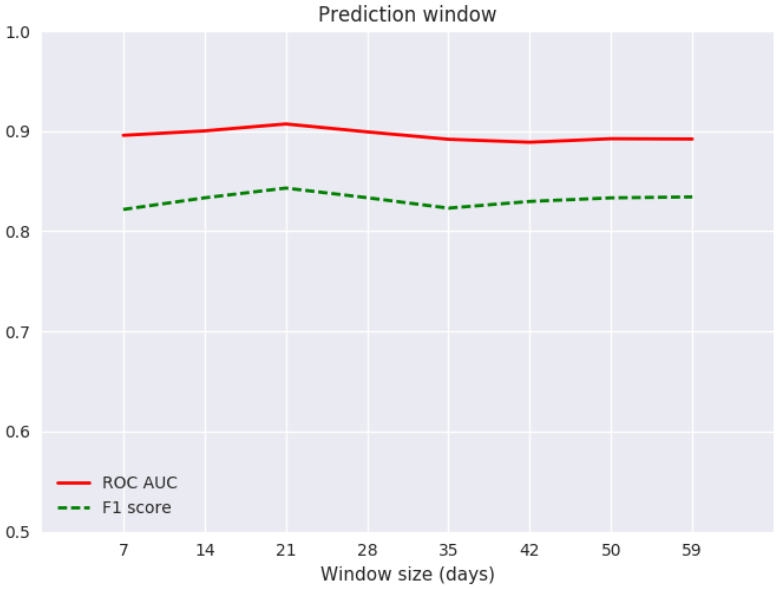

Figure 5.2: Effect on performance metrics for different prediction window sizes

Chapter 6

Conclusions and Future Work

Bibliography

[1] Netflix Quarterly Earnings. https://ir.netflix.com/results.cfm. Acessed: 2017-03-01.

[2] Spotify Press. https://press.spotify.com/us/about/. Accessed: 2017-03-01.

[3] IFPI Global Music Report 2016. http://www.ifpi.org/downloads/GMR2016.pdf. Accessed: 2017-03-01.

[4] Mohammed Hassouna, Ali Tarhini, Tariq Elyas, Mohammad Saeed Aboutrab, United Kingdom, United Kingdom, and Saudi Arabia. Customer Churn in Mobile Markets: A Comparison of Techniques. 8(6):224–237, 2015. doi: 10.5539/ibr.v8n6p224.

[5] Ning Lu, Hua Lin, Jie Lu, and Guangquan Zhang. A customer churn prediction model in telecom industry using boosting. *IEEE Transactions on Industrial Informatics*, 10(2): 1659–1665, 2014. ISSN 15513203. doi: 10.1109/TII.2012.2224355.

[6] Muhammad Raza Khan, Joshua Manoj, Anikate Singh, and Joshua Blumenstock. Behavioral Modeling for Churn Prediction: Early Indicators and Accurate Predictors of Custom Defection and Loyalty. *Proceedings - 2015 IEEE International Congress on Big Data, BigData Congress 2015*, pages 677–680, 2015. doi: 10.1109/BigDataCongress.2015.107.

[7] Vishal Mahajan, Richa Misra, and Renuka Mahajan. Review of data mining techniques for churn prediction in telecom. *Journal of Information and Organizational Sciences*, 39(2): 183–197, 2015.

[8] Özden Gür Ali and Umut Aritürk. Dynamic churn prediction framework with more effective use of rare event data: The case of private banking. *Expert Systems with Applications*, 41(17):7889–7903, 2014. ISSN 09574174. doi: 10.1016/j.eswa.2014.06.018.

[9] David R Cox. The regression analysis of binary sequences. *Journal of the Royal Statistical Society. Series B (Methodological)*, pages 215–242, 1958.

[10] J. Burez and D. Van den Poel. Handling class imbalance in customer churn prediction. *Expert Systems with Applications*, 36(3 PART 1):4626–4636, 2009. ISSN 09574174. doi: 10.1016/j.eswa.2008.05.027. URL `http://dx.doi.org/10.1016/j.eswa.2008.05.027`.

[11] Zoheb Borbora, Jaideep Srivastava, Kuo Wei Hsu, and Dmitri Wil Iams. Churn prediction in MMORPGs using player motivation theories and an ensemble approach. *Proceedings - 2011 IEEE International Conference on Privacy, Security, Risk and Trust and IEEE International Conference on Social Computing, PASSAT/SocialCom 2011*, pages 157–164, 2011. doi: 10.1109/PASSAT/SocialCom.2011.122.

[12] Julian Runge, Peng Gao, Florent Garcin, and Boi Faltings. Churn prediction for high-value players in casual social games. *IEEE Conference on Computatonal Intelligence and Games, CIG*, 2014. ISSN 23254289. doi: 10.1109/CIG.2014.6932875.

[13] Jagat Pudipeddi, Leman Akoglu, and Hanghang Tong. User Churn in Focused Question Answering Sites: Characterizations and Prediction. pages 469–474, 2014.

[14] Gideon Dror, Dan Pelleg, Oleg Rokhlenko, and Idan Szpektor. Churn prediction in new users of Yahoo! answers. *Proceedings of the 21st International Conference on World Wide Web*, pages 829–834, 2012.

[15] Vladislav Lazarov and Marius Capota. Churn prediction. *Bus. Anal. Course. TUM Comput. Sci*, 2007.

[16] Anders Drachen, Eric Thurston Lundquist, Yungjen Kung, Pranav Simha Rao, Diego Klabjan, Rafet Sifa, and Julian Runge. Rapid prediction of player retention in free-to-play mobile games. *CoRR*, abs/1607.03202, 2016.

[17] Michel Ballings and Dirk Van Den Poel. Customer event history for churn prediction: How long is long enough? *Expert Systems with Applications*, 39(18):13517–13522, 2012. ISSN 09574174. doi: 10.1016/j.eswa.2012.07.006. URL `http://dx.doi.org/10.1016/j.eswa.2012.07.006`.

[18] Artit Wangperawong, Cyrille Brun, Dr. Olav Laudy, and Rujikorn Pavasuthipaisit. Churn analysis using deep convolutional neural networks and autoencoders. pages 1–6, 2016.

[19] Kristof Coussement and Dirk Van den Poel. Churn prediction in subscription services: An application of support vector machines while comparing two parameter-selection techniques. *Expert systems with applications*, 34(1):313–327, 2008.

[20] Gaurangi Anand Auon, Haidar Kazmi, Pankaj Malhotra, Lovekesh Vig, Puneet Agarwal, and Gautam Shroff. Deep Temporal Features to Predict Repeat Buyers. (February 2016), 2015.

[21] Martin Längkvist, Lars Karlsson, and Amy Loutfi. A review of unsupervised feature learning and deep learning for time-series modeling. *Pattern Recognition Letters*, 42(1): 11–24, 2014. ISSN 01678655. doi: 10.1016/j.patrec.2014.01.008.

[22] Alex Graves, Abdel-rahman Mohamed, and Geoffrey Hinton. Speech recognition with deep recurrent neural networks. In *Acoustics, speech and signal processing (icassp), 2013 ieee international conference on*, pages 6645–6649. IEEE, 2013.

[23] Joe Yue-Hei Ng, Matthew Hausknecht, Sudheendra Vijayanarasimhan, Oriol Vinyals, Rajat Monga, and George Toderici. Beyond short snippets: Deep networks for video classification. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4694–4702, 2015.

[24] Niek Tax, Ilya Verenich, Marcello La Rosa, and Marlon Dumas. Predictive Business Process Monitoring with LSTM Neural Networks. 2016. URL `http://arxiv.org/abs/1612.02130`.

[25] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.

[26] David Martin Powers. Evaluation: from precision, recall and f-measure to roc, informedness, markedness and correlation. 2011.

[27] Isabelle Guyon and André Elisseeff. An introduction to variable and feature selection. *Journal of machine learning research*, 3(Mar):1157–1182, 2003.

[28] Boxun Zhang, Gunnar Kreitz, Marcus Isaksson, Javier Ubillos, Guido Urdaneta, Johan a. Pouwelse, and Dick Epema. Understanding user behavior in Spotify. *2013 Proceedings IEEE INFOCOM*, pages 220–224, 2013. ISSN 0743-166X. doi: 10.1109/INFCOM.2013.6566767. URL `http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=6566767`.

[29] Yann A LeCun, Léon Bottou, Genevieve B Orr, and Klaus-Robert Müller. Efficient backprop. In *Neural networks: Tricks of the trade*, pages 9–48. Springer, 2012.

[30] Charles X Ling and Chenghui Li. Data mining for direct marketing: Problems and solutions. In *KDD*, volume 98, pages 73–79, 1998.

Appendix A

Unnecessary Appended Material

To do...

- ☐ 1 (p. 1): Add temporal features paragraph
- ☐ 2 (p. 6): add feature selection techniques (Information gain, L1-norm, Recursive Feature Elimination, Variance Threshold...)
- ☐ 3 (p. 6): add refs
- ☐ 4 (p. 7): improve LR description
- ☐ 5 (p. 7): add models (Decision Trees, Random Forests, LSTMs, CNNs...)
- ☐ 6 (p. 8): Check with Sahar
- ☐ 7 (p. 9): Add eval methods (Lift chart, Top Decile Lift (TDL)...)
- ☐ 8 (p. 9): Talk about precision-recall curves and PR AUC
- ☐ 9 (p. 18): Read [8], [4], [18], [5], improve this section
- ☐ 10 (p. 19): check this claim
- ☐ 11 (p. 19): write about his approach
- ☐ 12 (p. 19): Important! Go into more details here, maybe add a figure