Aalto University
School of Business

# EMPIRICAL ANALYSIS OF PREDICTION MODELS FOR CHURN AND CUSTOMER LIFETIME VALUE IN FREEMIUM MOBILE GAMES

Master's Thesis
Jani Kurki
Aalto University School of Business
Information and Service Management
Spring 2020

Aalto University
School of Business

Aalto University, P.O. BOX 11000, 00076 AALTO
www.aalto.fi

**Abstract of master's thesis**

**Author** Jani Kurki

**Title of thesis** Empirical analysis of prediction models for churn and customer lifetime value in freemium mobile games

**Degree** Master of Science in Economics and Business Administration

**Degree programme** Information and Service Management

**Thesis advisor(s)** Assistant professor Jukka Luoma & associate professor Pekka Malo

**Year of approval** 2020 **Number of pages** 64 **Language** English

Abstract

In mobile gaming, the freemium business model of downloading the game for free and paying for voluntary in-app purchases has become popular. However, this has made it hard to predict when players stop playing the game and how much revenue they will generate during their lifetime. These prediction tasks are called churn and customer lifetime value (CLV) predictions respectively.

The thesis aims at answering two questions. First, what the current academic state-of-the-art solutions are to predict churn and CLV in a noncontractual and continuous setting, and second if the suggested models work in a real business setting for freemium mobile games. The empirical analysis is performed with data from a Finnish mobile gaming company and the target variables are measured 30 days after a new player's registration.

Both probabilistic models and machine learning models are in scope. A family of probabilistic models called 'Buy-till-you-die' models are popular for noncontractual settings, and two of them are included in the both prediction tasks of the empirical analysis. Within machine learning, the ensemble methods of gradient boosting and random forests have been effective in both classification and regression tasks, and deep neural networks have been found suitable in some of the latest studies.

The empirical analysis is done in two parts, for churn and CLV separately, as they represent different prediction problems, classification and regression respectively. The achieved predictive performance is on par with earlier literature. For churn, no single model outperforms the rest, but logistic regression and random forest classifier are recommended due to their easy and transparent implementation. Linear regression and random forest regressor perform best in CLV prediction and they the recommended choice. An observation period of one week is suitable for predictor variables, and a longer period does not improve model performance.

Hence, it is recommended that practitioners utilize the aforementioned models with an observation period of one week. In addition, feature importances of the models indicate which features of the game affect churn and CLV, and companies can focus on improving these factors. In the analyzed game, sessions and sum of transactions from the observation period have by far most predictive power, indicating that in-game behavior would not affect outcomes. CLV prediction results were accurate enough to enable customer segmentation, but not sufficient to accurately predict future cash flows.

**Keywords** churn, customer lifetime value, CLV, LTV, freemium, mobile games, noncontractual, prediction, prediction models, machine learning, buy-till-you-die, BTYD, deep neural networks

**Tekijä** Jani Kurki

**Työn nimi** Asiakassuhteen loppumisen ja asiakkaan liikevaihdon ennustemallien empiirinen analyysi freemium-mobiilipeleissä

**Tutkinto** Kauppatieteiden maisteri

**Koulutusohjelma** Tieto- ja palvelujohtaminen

**Työn ohjaaja(t)** Apulaisprofessori Jukka Luoma & professori Pekka Malo

**Hyväksymisvuosi** 2020 **Sivumäärä** 64 **Kieli** englanti

Tiivistelmä

Mobiilipeleissä freemium-liiketoimintamalli on tullut yhä suosituimmaksi. Freemium-mallissa pelin voi ladata ilmaiseksi, ja vapaaehtoisesta maksusta saa lisäominaisuuksia peliin. Tämä on kuitenkin vaikeuttanut sen ennustamista, milloin pelaajat lopettavat pelaamisen ja kuinka paljon rahaa he tuottavat elinaikanaan. Näitä ennustamishaasteita kutsutaan asiakassuhteen loppumisen ja asiakkaan elinajan arvon (CLV) ennustamiseksi.

Tämä työ pyrkii vastaamaan kahteen kysymykseen. Ensinnäkin, mitkä ovat akatemiassa nykyiset huippuratkaisut kyseisten ennustamishaasteiden ratkaisemiseen ympäristössä, jossa asiakassuhdetta ei ole sopimuksella vahvistettu, se voi loppua milloin hyvänsä ja ostoksia voi tehdä milloin tahansa. Toiseksi, työ selvittää toimivatko löydetyt mallit oikean elämän liiketoimintaongelmassa freemium-mobiilipelissä. Empiirinen analyysi toteutetaan erään suomalaisen mobiilipeliyhtiön tarjoamalla oikealla datalla. Ennustukset tehdään 30 päivän päähän kunkin asiakkaan rekisteröitymisestä peliin.

Sekä todennäköisyys- että koneoppimismallit ovat mukana. 'Buy-till-you-die' (BTYD) -todennäköisyysmallit ovat suosittuja vastaavissa ympäristöissä, ja kaksi näistä malleista on mukana empiirisessä osuudessa. Koneoppimismallien joukossa gradient boosting sekä random forest -mallit ovat olleet tehokkaita sekä luokittelu- että regressiotehtävissä, ja lisäksi syviä neuroverkkoja on hyödynnetty uusimmissa julkaisuissa.

Empiirinen analyysi tehdään kahdessa osassa, asiakassuhteen loppumisen ja arvon ennustamiselle erikseen, sillä ne edustavat eri ennustamisongelmia, eli luokittelua ja regressiota. Mallien suoriutuminen on samalla tasolla aiemman kirjallisuuden kanssa. Asiakassuhteen ennustamisessa yksikään malli ei ole selkeästi muita parempi, mutta logistista regressiota ja random forest -luokittelumallia suositellaan niiden helppokäyttöisyyden ja läpinäkyvyyden takia. Arvon ennustamisessa lineaarinen regressio ja random forest -regressiomalli suoriutuvat parhaiten ja ovat siten suositeltu valinta. Viikon mittainen havainnointiaika on sopiva datan keräämiseksi ennustamista määrittäville muuttujille, eikä pidempi aikaikkuna paranna malleja.

Täten suosituksena toimialalle on käyttää mainittuja malleja viikon pituisella havainnointiajalla. Lisäksi tietoa mallien muuttujien painoarvoista ennusteisiin voidaan hyödyntää pelikehityksessä. Analysoidun pelin tapauksessa eniten painoa on sessioiden lukumäärällä sekä käytetyllä rahamäärällä, mikä viittaa siihen, että pelin sisäinen käyttäytyminen ei vaikuta lopputulokseen. Arvon ennustamisen tarkkuus mahdollistaa asiakassegmentoinnin, mutta kassavirtaennusteisiin summat ovat liian epätarkkoja.

**Avainsanat** asiakassuhde, asiakkaan elinajan arvo, freemium, mobiilipeli, ennustaminen, ennustemalli, koneoppiminen, buy-till-you-die, BTYD, syvät neuroverkot

# Table of Contents

1 Introduction ..... 5

1.1 Background ..... 5

1.2 Research questions ..... 6

1.3 Structure of the study ..... 6

2 Literature review ..... 8

2.1 Conceptual background ..... 8

2.1.1 Customer lifetime value ..... 8

2.1.2 Categorization of CLV analysis settings ..... 9

2.2 Churn and CLV prediction in gaming ..... 11

2.2.1 Churn prediction ..... 11

2.2.2 CLV prediction ..... 13

2.3 Churn and CLV prediction in other noncontractual settings ..... 16

2.3.1 Buy-till-you-die models ..... 16

2.3.2 Machine learning models ..... 20

3 Methodology ..... 22

3.1 Model selection ..... 22

3.1.1 Heuristics ..... 22

3.1.2 BTYD models ..... 23

3.1.3 Machine learning models ..... 24

3.2 Introduction to selected models ..... 25

3.2.1 Pareto/NBD (HB) ..... 25

3.2.2 CBG/NBD ..... 26

3.2.3 Pareto/GGG ..... 27

3.2.4 Linear regression ..... 28

3.2.5 Logistic regression ..... 28

3.2.6 Random forests ..... 29

3.2.7 Gradient boosting ..... 31

3.2.8 Conditional inference survival ensembles ..... 33

3.2.9 Deep multilayer perceptron ..... 34

3.3 Evaluation criteria ..... 34

4 Case introduction ..... 38

**4.1 Case company and game ..... 38**

**4.2 Introduction to data ..... 38**

**4.3 Choice of tools for analysis ..... 40**

**5 Analysis and results ..... 41**

**5.1 Data preparation ..... 41**

**5.2 Model-specific notes on implementation ..... 42**

5.2.1 BTYD and survival ..... 42

5.2.2 Machine learning models ..... 42

5.2.3 Deep-MLP ..... 43

**5.3 Results ..... 44**

5.3.1 Churn prediction ..... 44

5.3.2 CLV prediction ..... 46

**6 Discussion ..... 49**

**6.1 Key findings ..... 49**

6.1.1 Churn prediction ..... 49

6.1.2 CLV prediction ..... 51

**6.2 Managerial implications ..... 52**

**6.3 Limitations and future research ..... 55**

**6.4 Conclusion ..... 56**

**References ..... 57**

# List of Tables

Table 1: Summary of churn and CLV prediction in gaming literature ..... 15
Table 2: Summary of BTYD models..... 19
Table 3: Summary of machine learning models in other noncontractual settings..... 21
Table 4: Confusion matrix of binary classification ..... 35
Table 5: Summary of BTYD models..... 38
Table 6: Results of churn prediction by model and observation period..... 45
Table 7: Results of customer lifetime value prediction by model and observation period . 46
Table 8: Mean prediction error by model, player spending and observation period (USD)47

# List of Figures

Figure 1. Classifying customer relationships and examples (Fader & Hardie, 2009). ..... 9
Figure 2. Classifying differences in customer death processes (Jerath et al., 2011). ..... 18
Figure 3. Illustration of logistic regression with one predictor (Peng et al., 2002). ..... 29
Figure 4. Illustration of bagging process (modified from Maldonado et al., 2014). ..... 30
Figure 5. Illustration of a decision tree for classification. ..... 30
Figure 6. Illustration of boosting process (modified from Maldonado et al., 2014). ..... 32
Figure 7. Illustration of a ROC graph. ..... 36
Figure 8. Relative spend distribution among spending players (USD) ..... 39
Figure 9. Occasion of last session by player, cumulative distribution ..... 40
Figure 10. Relative feature importances in gradient boosting and random forest classifiers, 7-day period. ..... 45
Figure 11. Total prediction error (USD) by model, 7-day period. ..... 47
Figure 12. Relative feature importances in gradient boosting and random forest regressors, 7-day period. ..... 48
Figure 13. Illustration of a possible customer segmentation by player CLV (USD) ..... 54

# 1 Introduction

## 1.1 Background

In 2010s at latest, it became clear that mobile technology and the application ecosystem it enables have become an integral part of our daily lives. Applications are used for almost everything, from processing transactions to spend leisure time with. This development in mobile technology has also enabled the rise of mobile gaming. According to App Annie's mobile app report (2019), mobile gaming accounted for 74% of spent money in app stores in 2018, and simultaneously the industry is estimated to reach 60% of the total gaming market in 2019, leaving consoles and personal computers behind.

Within mobile gaming, the freemium business model of downloading the game for free and paying for voluntary in-app purchases has become popular. For instance, many of the global top-grossing games of the last few years, such as *PUBG Mobile*, *Clash of Clans*, *Fortnite* and *Pokémon Go*, have adopted the freemium model. The model, however, makes business planning more complicated, as incoming cash flows are harder to predict on both individual player and aggregate level. Marketing science uses the term "customer lifetime value" (CLV) to determine the profitability of a given customer or customer group in the timespan of the entire lifecycle of the customer relationship. More precisely, Fader et al. (2005a) define CLV as the present value of combined future cash flows of the customer.

For freemium mobile games, the task of determining CLV is especially tough. Following Fader's and Hardie's (2009) as well as Schmittlein et al's (1987) classifications of customer bases, freemium mobile games fall within the noncontractual and continuous category. "Noncontractual" refers to the situation, where customers can make purchases without a longer commitment to the company, hence there is no need for a signed contract to define the relationship, but the relationship can end at any point without the company knowing it. "Continuous" refers to transactions and that they can occur at any point in time, instead of in discrete intervals. This nature of the customer relationship in freemium mobile games makes the CLV hard to predict.

An accurate prediction of the CLV would, however, be very valuable for mobile gaming companies. First, it would help cash flow management within the company, as the incoming flow can be estimated more accurately. Second, non-contractual CLV takes into account customer churn, and the company could try to prevent churn preemptively by e.g. targeted marketing and offers. Third, changes in predicted CLV figures can provide insight

into games and their features. For instance, after an update is launched, new customer behavior might indicate whether the new features are having a positive effect in customer spend. For this third benefit, even an accurate prediction of any of the CLV components, especially churn, would be valuable.

These managerial implications of successful CLV prediction for freemium mobile games make the topic both interesting and relevant in business environment and worth further research.

## 1.2 Research questions

The managerial motivation for finding the best models for CLV prediction are clear, but the challenge is also interesting from the academic point of view. A thorough sweep through the literature from freemium mobile gaming perspective to determine state-of-the-art solutions has not been conducted recently.

The goal of this thesis is hence to perform just this and additionally create empirical results on top of it. The research problem therefore is split into two separate questions:

1) *What are the current academic state-of-the-art solutions for modeling customer churn and lifetime value in a noncontractual setting?*
2) *Do the proposed models work in a real business setting for freemium mobile games?*

The concrete aims of the thesis are linked to both research questions. The first concrete aim is to gather a short list of predictive models that would be most suitable for non-contractual business setting. The second aim is to create real-life results how the proposed models work in action in the context of freemium mobile games. To be more precise, the aim is to predict new players' churn and customer lifetime value after the first 30 days since registration by using data from a shorter observation period, e.g. only the first week of playing.

I am fortunate enough to receive real data from a Finnish mobile gaming company for the study, and the last aim of the thesis is thus to provide the company with an implementable model for their churn and CLV prediction challenges.

## 1.3 Structure of the study

The introduction is followed by the literature review, where both probabilistic and machine learning models are in scope. In addition to introducing the conceptual background, the review examines predictive models that directly predict CLV or churn, a component of

CLV, as well as machine learning models that predict these metrics outside gaming in other noncontractual settings.

The empirical part of the thesis begins after the literature review. First, I go through the methodology for the analysis, including selected models and evaluation criteria for them. Then introduction to the case, the analysis and results are presented. The analysis itself is performed separately for churn and CLV predictions. Through the empirical part, lifetime value is used as a synonym for their first 30 days. For spending players who in reality stay alive longer the actual lifetime value will of course be longer.

After the results, the section for discussion includes key findings, managerial implications, limitations of the study and ideas future research, and finally the final conclusion. These are lastly followed by references.

# 2 Literature review

## 2.1 Conceptual background

### 2.1.1 Customer lifetime value

Pfeifer et al. (2005) point out that several academic papers are inconsistent in the usage of the term *customer lifetime value* (CLV) and often use it and the term *customer profitability* as synonyms. Customer value and profit are mixed with each other, and the authors turn to academic definitions found in finance and accounting for widely accepted definitions.

Pfeifer at al. (2005) continue that the value should be understood as "valuation of an asset in determining its net present value". Therefore, CLV should not represent profit, but rather the present value of all future cash flows of a given customer during the total duration of the customer relationship. This definition also leaves out all costs related to the customer, otherwise it would become profit instead of value.

In this sense, the discipline of finance values a customer quite similarly to valuing a company with the help of the discounted cash flow (DCF) method. There are, however, essential differences between CLV and DCF calculations. In DCF, the total cash flows from all products and services are aggregated into one and the method contains the assumption of perpetuity, i.e. cash flows will not end in the future but will continue and possibly even with some growth rate. CLV, on the other hand, is calculated on an individual customer level and customer attrition, also known as *churn*, is taken into account (Gupta & Lehmann, 2003). The moment of churn determines the endpoint of the customer relationship, and no further cash flows are expected from the customer afterwards.

To conclude, this thesis sticks to the definition of CLV formulated by Pfeifer et al. (2005), which goes as follows:

*"The present value of the future cash flows attributed to the customer relationship"*

This formulation of CLV and the calculation of it can be achieved from a different angle as well. Many existing prediction models (e.g. Abe, 2009; Platzer & Reutterer, 2016) calculate the CLV on an individual customer level by estimating the total number of purchases multiplied by the monetary value of the purchases, which results in the predicted

future cash flows. Hence, CLV can also be seen as a function of transactions and their monetary value.

In practice, these predictive models utilizing this approach base the predictions on three factors that resemble the aforementioned function: recency, frequency and monetary value (e.g. Fader & Hardie, 2009; Platzer & Reutterer, 2016). *Recency* refers to the time since the previous transaction occurred, usually measured in days. *Frequency* is the total number of transactions during the given time period. *Monetary value* is self-explanatory, but it is worth noting that average spend is used as an approximation for all the transactions. Combined, these factors make up the so called *RFM-based framework* for CLV predictions (Platzer & Reutterer, 2016).

### 2.1.2 Categorization of CLV analysis settings

Calculation of customer lifetime value is heavily dependent on the type of relationship between the company and the customer. Fader and Hardie (2009) present a customer relationship matrix to classify distinct types of customer relationships based on observability of customer attrition and transaction occurrence, as shown in Figure 1.

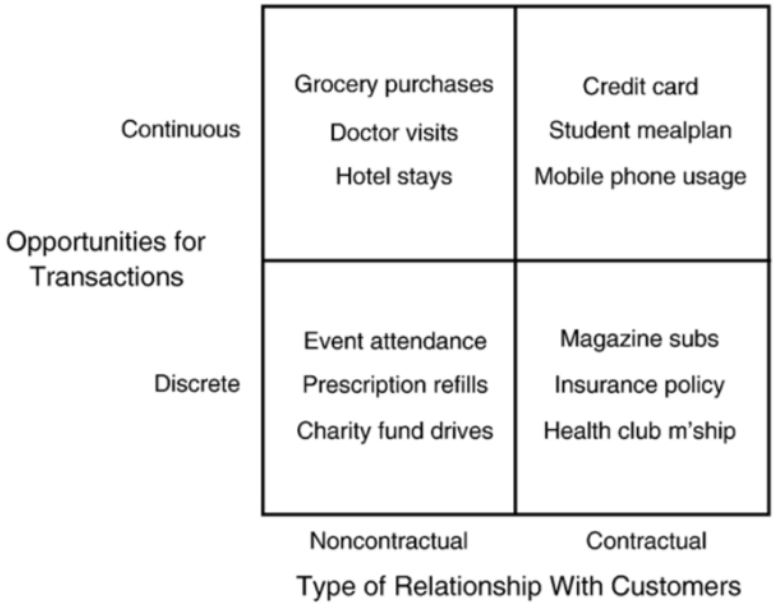


*Figure 1. Classifying customer relationships and examples (Fader & Hardie, 2009).*

In Figure 1, the x-axis of "Type of Relationship With Customers" refers to the aforementioned observability of customer attrition. In contract-based customer relationships, it is explicit when the customer becomes inactive, or in other words "dies".

A contract, for instance a 12-month subscription that is paid monthly, comes to the end of its lifetime and the contract is not renewed. This is an example of a *contractual* setting and businesses such as mobile phone operators and newspapers rely on it. In a *noncontractual* setting, on the other hand, customer attrition cannot be observed. If a customer has not made transactions for a while, the company does not know for sure whether the customer is "dead" or just "alive" but with an unusually long pause between transactions. The probability of death, naturally, increases as the period without transactions gets longer, but it is still not entirely certain. Grocery stores are an everyday-life example of this type of customer relationship. It is worth noting that models focusing on noncontractual settings assume that once a customer has died, he or she will not come back later on (e.g. Schmittlein et al., 1987; Fader et al., 2005a; Jerath et al., 2011).

The y-axis in Figure 1, "Opportunities for Transactions", refers to whether the transaction can occur at any point in time or within a determined time interval. Examples of limited availability include concert tickets, where the sales start at a fixed time before the event itself, and medicine prescriptions, where availability might be limited for patient safety.

The most complex quadrant of the matrix for predictions is the upper-left corner, where transactions occur at any point in time and customer attrition cannot be observed. One of the most popular examples from a business setting representing this corner are web stores, but freemium mobile games represent this setting as well. A player can make purchases in the app at any time and as games are always available without commitment, customer death cannot be known for sure.

As Fader and Hardie (2009) argue, prediction models should be designed for one quadrant of the matrix only. Hence, this thesis focuses on models that tackle the noncontractual and continuous business setting.

## 2.2 Churn and CLV prediction in gaming

While the literature in other noncontractual settings apart from gaming has been focusing on stochastic prediction models and transactional user data (more on that in Chapter 2.3), papers on prediction models for gaming have adapted a machine learning approach. This different approach probably stems from a better supply of data. In 2010s, the gaming industry has shifted focus on online gaming, which has enabled efficient and real-time data gathering on an individual player level. Consequently, proposed prediction models utilize behavioral data instead of just transactional data (e.g. player login times, session lengths and player level in game). Furthermore, the literature is richer in churn prediction, a component of CLV, than CLV prediction itself.

### 2.2.1 Churn prediction

In gaming, churn occurs when a player has stopped playing the game for good. As in other noncontractual settings, the exact time cannot be determined, but prediction models use a time interval as a proxy. The models may assume that, for instance, seven or 14 days without a login are needed to identify a churner (e.g. Runge et al., 2014; Milošević et al., 2017). The prediction problem for churn is often defined as a binary classification task, where each player is labeled either a churner or a non-churner (Runge et al., 2014; Hadiji et al., 2014).

For churn prediction, tested and suggested supervised machine learning algorithms range from simpler models, such as decision trees, to neural networks. Runge et al. (2014) as well as Hadiji et al. (2014) were the first ones to research churn prediction in freemium casual social games. Runge et al. (2014) compared the performance of four different algorithms in two separate games, and the most accurate results based on area under ROC curve (AUC) were clearly neural network and logistic regression, the former having a slight edge over the latter. Hadiji et al. (2014) compared four algorithms across five different games and, based on $F_1$-score (a.k.a. F-score), decision trees yielded best results, and they were clearly better than neural networks.

Kim et al. (2017) predicted churn for three casual games, two of them mobile and one online, utilizing play log data. They compared three “traditional” algorithms (logistic regression, gradient boosting and random forests) and two deep learning algorithms, convolutional neural networks (CNN) and long short-term memory (LSTM). Unlike in earlier papers, the authors performed feature engineering for the traditional algorithms to

determine the most meaningful features to be used by the algorithms. The results for the three different games varied, but generally gradient boosting gave best results. Using AUC as the criterion, LSTM and gradient boosting yielded best results for the first game, the two models having only a one thousandth difference. For the second game, gradient boosting was the best model and for the third game gradient boosting and logistic regression yielded exactly same results.

Milošević et al. (2017) honed the approach to suit casual games better by basing the classification task on players' behavioral data from the first day of activity. As lifetime in casual freemium games is low, fast predictions benefit the provider greatly (Milošević et al., 2017). Their findings were in line with previous literature: Based on both AUC and $F_1$-score, gradient boosting algorithm worked best, followed by random forest and logistic regression. As the AUC score was as high as 0.83, Milošević et al. (2017) proved that it is possible to predict churn after one day of player activity with decent performance.

Drachen et al. (2016) examined if churn prediction was possible with a smaller time window of data, an approach similar to Milošević et al. (2017). For a new cohort of players in a freemium mobile game, the authors tested prediction performance after one single session, the first day of playing and finally after an entire week of playing data. The results suggest that prediction after the first session and first day are decent, and that no single machine learning model performed clearly better than the rest. In addition, Drachen et al. (2016) benchmarked the machine learning models against simple predictive heuristics, which yielded almost as promising results as the models, indicating that easy-to-implement heuristics are an alternative in a commercial context where analytical resources are limited.

Periáñez et al. (2016) selected another approach to predict churn. Instead of handling it as a binary classification task, they incorporated the "temporal dimension" in the model, hence making it a survival analysis. In other words, the model gives a prediction when a player will churn. As the authors put it, all players get "a survival probability function that will let us know how the probability of churning is varying as a function of time". This model presented by Periáñez et al. (2016) worked best with survival ensemble method (more particularly with the conditional inference survival ensembles technique), which was compared with Cox regression based on integrated Brier score (IBS). In addition, the authors performed a binary classification analysis, where the same algorithm of conditional inference ensembles was put against several binary classifiers. In this analysis both the new algorithm and support vector machine (SVM) yielded excellent results, AUCs of 0.96

and 0.94 respectively. However, SVM is considered "a black box", because the relationship between its inputs and outputs in hardly visible (Periáñez et al., 2016).

Conditional inference survival ensembles were also found the most suitable method by Bertens et al. (2017). Their model is an extension of the aforementioned work by Periáñez et al. (2016) and instead of just providing a survival probability for each player, the model predicts when a given player is expected to churn.

A more hands-on approach was performed by Lee et al. (2019), who organized a data mining competition to predict churn and survival time for players of a game developed by the Korean game company NCSOFT. It is rare that game companies share their data with external researchers (Lee et al., 2019) and therefore this competition provides a unique setting. The competition consisted of two separate tracks, one for churn prediction and one for survival analysis. In addition, the dataset was separated into two parts, because the business model the game operated on changed during the time period from subscription to freemium. For the freemium business model churn prediction, the winning solution was extremely randomized trees, which is a model close to random forests. For survival analysis, the solution was consistent with earlier literature, namely conditional survival ensembles yielded best results. Feature selection was performed with the Lasso estimator.

### 2.2.2 CLV prediction

Sifa et al. (2015) were the first ones investigate purchase decisions in freemium mobile games with players' behavioral data. The authors approached the CLV prediction problem with a three-step approach. First, the paying players are predicted as a conventional binary classification task, where random forests performed best based on $F_1$-score and G-mean. Furthermore, Sifa et al. (2015) balanced the highly imbalanced dataset by synthetically generating paying players following Synthetic Minority Over-sampling Technique-Nominal Continuous (SMOTE-NC), which improved the accuracy. Second, the number of purchases is predicted as a regression task. The same features as in the classification task are used, but the target variable is changed from binary to integral range. The authors suggest using a Poisson regression tree and assume a Poisson distribution for the number of purchases, and the accuracy is measured with the root mean squared error (RMSE) metric. Third, the model becomes a CLV prediction model when an estimates of future purchase values are combined with the number of purchases, but the authors do not tackle this phase in more detail (Sifa et al., 2015).

A similar approach was chosen by Drachen et al. (2018), who first performed a classification task for paying and non-paying players. In addition, they used a dataset of a game with a social dimension, so the authors classified the players to social and non-social ones as well, which was new to the literature. For the paying player classification, random forests and gradient boosting worked best, with some variation depending on the time frame of the model training data. For the classification task, Drachen et al. (2018) also implemented the Synthetic Minority Over-sampling Technique (SMOTE) to balance the data. In the regression phase, gradient boosting outperformed random forest both in terms of normalized rooted mean squared error (NRMSE) and $R^2$.

Other papers utilize supervised machine learning algorithms to directly predict CLV on an individual player level without the aforementioned three-step approach. Sifa et al. (2018) compared the performance of four different algorithms with and without SMOTE. Based on NRMSE, their analysis found out that deep multilayer perceptrons (Deep-MLP) performed best, followed by random forest and logistic regression. Implementing SMOTE improved results for all these three algorithms.

Chen et al. (2018) use a deep perceptron multilayer network and convolutional neural networks (CNNs) for the prediction. Their paper is also the first to utilize CNNs for CLV prediction in the video games context, and also the first to compare deep learning techniques with probabilistic models (more on these in Chapter 2.3) for CLV prediction. The inputs of the CNNs were time series pictures illustrating the probability density function for each player. The predictive performance of the both deep learning algorithms were similar and significantly better than the performance of the probabilistic models based on four metrics: RMSLE, NRMSE, SMAPE and % error.

Table 1 summarizes these different approaches found in the literature.

*Table 1: Summary of churn and CLV prediction in gaming literature*

| Authors | Churn prediction | Churn definition | CLV prediction | Method | Observation period | Model validation | Suggested model(s) | Evaluation criteria | Feature selection | Dataset balancing | Non-spending players included | New users only |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Runge et al. (2014) | Yes | 14 days | No | Binary classification | 14 days | 10-fold cross-validation | Neural network & logistic regression | AUC | Experimentation | None | No | No |
| Hadiji et al. (2014) | Yes | 7 days | No | Binary classification | 7 days | 10-fold cross-validation | Decision trees | F-score | Manual extraction | None | Yes | No |
| Kim et al. (2017) | Yes | 10 days | No | Binary classification | 5 days | 10-fold cross-validation | Gradient boosting | AUC | Experimentation | None | Yes | Yes |
| Milošević et al. (2017) | Yes | 14 days | No | Binary classification | 1 day | 10-fold cross-validation | Gradient boosting, random forest, logistic regression | AUC & F-score | Manual extraction | Manual downsampling to balance churners and non-churners | Yes | No |
| Drachen et al. (2016) | Yes | Multiple | No | Binary classification | Multiple | 10-fold cross-validation | No clear winner | AUC & F-score | Manual extraction | None | Yes | No |
| Periáñez et al. (2016) | Yes | 10 days | No | Survival analysis & binary classification | 1 year 6 months | 1000 bootstrap cross-validation | Conditional inference survival ensembles & SVM | IBS for survival analysis, AUC for classification | Manual extraction | None | No | No |
| Bertens et al. (2017) | Yes | N/A | No | Survival analysis | Multiple | 1000 bootstrap cross-validation | Conditional inference survival ensembles | IBS | Manual extraction | None | No | No |
| Lee et al. (2019) | Yes | 5 weeks | No | Binary classification & survival analysis | 8 weeks | Cross-validation (number unknown) | Extremely randomized trees & conditional inference survival ensembles | F-score for classification, RMSLE for survival | Lasso regression | None | No | No |
| Sifa et al. (2015) | Yes | 7 days | Yes | Binary classification + regression | 1, 3 & 7 days | 10-fold cross-validation | Random forests for classification, Poisson regression tree for regression | F-score & G-mean for classification, RMSE for survival | Manual extraction | SMOTE-NC | Yes | No |
| Drachen et al. (2018) | Yes | 7 days | Yes | Binary classification + regression | 7 days | 10-fold cross-validation | Random forests & gradient boosting for classification, gradient boosting for regression | AUC & AUPR for classification, NRMSE & $R^2$ for regression | Manual extraction | SMOTE | Yes | No |
| Sifa et al. (2018) | No | N/A | Yes | Regression & deep learning | 7 days | 10-fold cross-validation | Deep-MLP | NRMSE | Manual extraction | SMOTE | Yes | No |
| Chen et al. (2018) | No | N/A | Yes | Deep learning | 32 months | Cross-validation (number unknown) | Deep-MLP & CNN | RMSLE, NRMSE, SMAPE, % error | Manual extraction | None | No | No |

## 2.3 Churn and CLV prediction in other noncontractual settings

### 2.3.1 Buy-till-you-die models

Churn and CLV prediction have been widely discussed in the marketing literature. For the noncontractual setting, a group of probabilistic models have been developed, which have been named 'Buy-till-you-die' models (BTYD) (Fader & Hardie, 2009), which are easy to implement and require a small number of datapoints. The models are stochastic instead of deterministic, i.e. the models assume that customer behavior is a stochastic process.

The BTYD models use customer-level transaction data as their input. More precisely, the models only require the information about the customers' recency and frequency following the aforementioned RFM-based framework (Wübben & Wangenheim, 2008). As the output, the models provide a prediction of individual-level churn and number of transactions before the churn.

In order to make the BTYD models CLV prediction models, the average monetary value of a customer's transactions is required. Two options for such a submodel are available in the literature: the normal-normal mixture by Schmittlein and Peterson (1994) and the gamma-gamma mixture by Colombo and Jiang (1999). As their names suggest, Schmittlein's and Peterson's (1994) model assumes that average monetary values follow a normal distribution, whereas Colombo's and Jiang's (1999) model follows a Gamma distribution. As Fader and Hardie (2013) point out, the normal distribution "is not bounded below by 0 and it results in a symmetric spend distribution". Therefore, the Gamma-Gamma model by Colombo and Jiang (1999) is a more suitable alternative for the context. When the average monetary value of the transactions is included, the BTYD models are comparable with machine learning models (Sifa et al., 2015). Fader et al. (2005b) were the first ones to present such a model that connected the RFM-based framework and CLV calculations. Their solution utilizes the aforementioned Gamma-Gamma model for predicting transaction value. Their analysis also shows that the monetary value can be calculated separately from the number of transactions.

Among the BTYD models, the first and probably the most cited one is the Pareto/NBD model developed by Schmittlein et al. (1987). The authors combine the negative binomial distribution (NBD) with a Pareto model to identify churn, hence the name. The model assumes that customer purchases follow a Poisson distribution around each customer's mean purchase rate and that customer lifetime follows an exponential distribution. The Pareto/NBD model was later modified and extended later by Ma and Liu

(2007) as well as Abe (2009). Ma and Liu (2007) present the Pareto/NBD (HB) alternative, which first revises the model from hierarchical Bayesian perspective and then proposes a Markov chain Monte Carlo (MCMC) algorithm for model estimation instead of the original maximum likelihood estimation. According to the authors, the new model is easier to implement and enables predictions on an individual level. Later on, Abe (2009) extended the model further by changing the independent gamma distributions for purchase and dropout rates to a multivariate lognormal distribution, which "permits a correlation between purchase and dropout processes". Abe's model also enables predictions on an individual customer level.

Fader et al. (2005a) developed a model that slightly varies from the Pareto/NBD model. Their model, BG/NBD, assumes that customers churn immediately after a purchase (if they do) instead of a possible churn at any point in time. This simplification makes the new model easier to implement, and according to Fader et al. (2005a), both models yield similar results, so the predictive performance does not suffer from the alteration.

Other alterations to the Pareto/NBD model have been developed later as well. Hoppe and Wagner (2007) developed a modification to the BG/NBD model. The new CBG/NBD model adds the possibility for churn at time zero, i.e. customers can churn immediately after trying the service. The model also worked slightly better than its predecessor (Hoppe & Wagner, 2007).

Hoppe's and Wagner's (2007) CBG/NBD model was again further developed by Platzer (2008), who kept the possibility of immediate churn, but replaced the Poisson distribution of transactions with Erlang-k distribution to "consider a certain extent of regularity. This resulted in the CBG/CNBD-k model (Platzer, 2008).

Jerath et al. (2011) developed the periodic death opportunity (PDO) model that also modifies the churn process. Unlike in the BG/NBD model, where the churn occurs right after a purchase, the PDO model periodically checks at discrete points in time whether a customer is churning or not. Figure 2 illustrates the differences of the churn processes in the three models. The prediction results of PDO were slightly better than for the two other models. However, Jerath et al. (2011) argue that the Pareto/NBD model should be used by managers when their primary goal is purchase forecasting instead of focusing on the churn process itself.

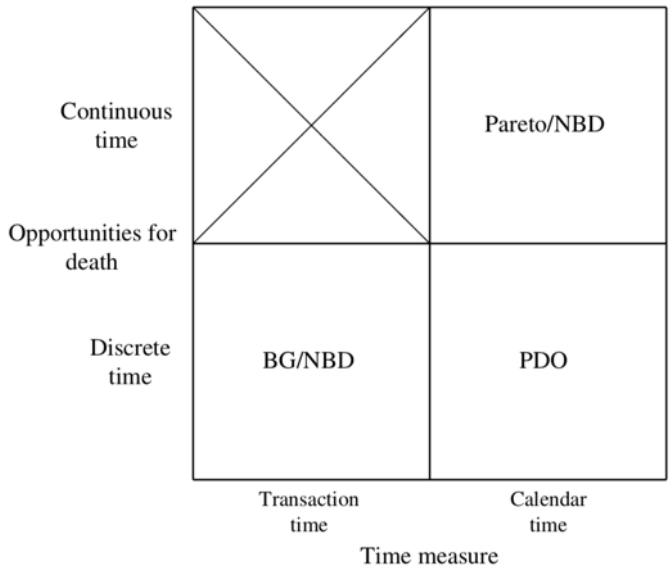


*Figure 2. Classifying differences in customer death processes (Jerath et al., 2011).*

Bemmaor and Glady (2012) propose a G/G/NBD model, which combines a gamma mixing of Gompertz distribution (G/G) with NBD. According to the authors, the model is more flexible than Pareto/NBD and it has two advantages. First, the probability density function can exhibit a mode at interior mode or zero. Second, the probability density function can be skewed to either right or left.

Platzer and Reutterer (2016) take into account the timing of a customer's transactions. Two customers with the same recency and frequency may time their purchases differently, resulting in different timing patterns. Their Pareto/GGG model considers these timing patterns in churn probability predictions. By replacing the Poisson purchase process found in the literature earlier with a mixture of gamma distributions, Platzer and Reutterer (2016) were able to outperform the Pareto/NBD model.

Table 2 summarizes the differences between the BTYD models.

*Table 2: Summary of BTYD models*

| Authors | Model | Purchase rate while alive | Heterogeneity in purchase rate | Churn rate | Purchase timing patterns considered |
|---|---|---|---|---|---|
| Schmittlein et al. (1987) | Pareto/NBD | Poisson distribution | Gamma distribution | Exponential distribution | No |
| Ma & Liu (2007 | Pareto/NBD (HB) | Poisson distribution | Gamma distribution | Exponential distribution | No |
| Abe (2009) | Pareto/NBD (Abe) | Poisson distribution | Multivariate lognormal distribution | Exponential distribution | No |
| Fader et al. (2005a) | BG/NBD | Poisson distribution | Gamma distribution | Right after purchase with some probability | No |
| Hoppe & Wagner (2007) | CBG/NBD | Poisson distribution | Gamma distribution | Right after purchase with some probability or at time zero | No |
| Platzer (2008) | CBG/CNBD-k | Erlang-k distribution | Gamma distribution | Right after purchase with some probability or at time zero | No |
| Jerath et al. (2011) | PDO | Poisson distribution | Gamma distribution | Checked at discrete intervals with some probability | No |
| Bemmaor & Glady (2012) | G/G/NBD | Poisson distribution | Gamma distribution | Gamma/Gompertz distribution | No |
| Platzer & Reutterer (2016) | Pareto/GGG | Gamma distribution | Gamma distribution | Exponential distribution | Yes |

### 2.3.2 Machine learning models

Although BTYD models have been successfully used with transaction data, the models do not make use of the vast amount of data that modern digital businesses generate. This has motivated machine learning approaches to the problem (Chamberlain et al., 2017).

Churn prediction was introduced in the gambling literature by Coussement and Bock (2013), who found random forests to perform best among the tested four alternatives in online gambling. Online gambling shares similarities to mobile gaming with its casual approach and game-like digital interface.

Wangperawong et al. (2016) utilized deep convolutional neural networks to predict churn in telecommunications. They examined prepaid customers' calls, which makes the context noncontractual. They found that deep convolutional neural networks performed better than simpler models, such as decision trees.

For new users and with just a week of user data, Dror et al. (2012) predicted churn for *Yahoo! Answers*, a community-based question answering (CQA) platform. In their tests, random forests yielded best performance, very closely followed by logistic regression, which is "much simpler to trains and easier to implement".

Vanderveld et al. (2016) were the first to employ customer engagement features for CLV prediction. They tackled the problem for a large e-commerce company and found random forest models to work best, and their model had two stages: First, churn is predicted as a binary classification task and then purchase value is estimated for the paying users.

Chamberlain et al. (2017) employed a model to predict CLV of an online retailer. The authors predicted churn as binary classification task utilizing random forests, and CLV with a random forest regression model, an approach similar to Vanderveld et al. (2016). In addition, Chamberlain et al. (2017) experimented churn prediction with deep neural networks, along with automated feature selection, but they argue that "the monetary cost required to perform such training outweighs the benefit of gain in performance".

Wang et al. (2019) proposed deep neural networks (DNNs) and, furthermore, they suggest that the standard mean squared error (MSE) loss function should be replaced by zero-inflated lognormal (ZILN) distribution. The authors point out that customer distribution can be heavy-tailed with a majority of customers never coming back. In such a setting, MSE does not perform well to take the outliers into account.

Table 3 summarizes the aforementioned machine learning models.

*Table 3: Summary of machine learning models in other noncontractual settings*

| Authors | Industry | Churn prediction | Churn definition | CLV prediction | Method | Observation period | Model validation | Suggested model(s) | Evaluation criteria | Feature selection | Dataset balancing | Non-spending players included | New users only |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Coussement & Bock (2013) | Gambling | Yes | 4 months | No | Binary classification | 18 months | Five times twofold cross-validation | Random forests | Top-decile lift & lift index | None | None | N/A | No |
| Wangperawong et al. (2016) | Tele-communications | Yes | 30 days | No | Deep learning | 14 days | Unknown | CNN | AUC | None | None | N/A | No |
| Dror et al. (2012) | CQA | Yes | Unknown | No | Binary classification | 7 days | 10-fold cross-validation | Random forests & logistic regression | AUC & F-score | Manual extraction | None | N/A | Yes |
| Vanderveld et al. (2016) | E-commerce | No | N/A | Yes | Binary classification + regression | 1 year | Cross-validation (number unknown) | Random forests | RMSE, Pearson and Spearman correlation | Manual extraction | Down-sampling | Yes | No |
| Chamberlain et al. (2017) | E-commerce | Yes | 1 year | Yes | Binary classification + regression | 1 year | 10-fold cross-validation | Random forests | AUC for churn, Spearman correlation for CLV | Manual extraction | None | Yes | No |
| Wang et al. (2019) | E-commerce & donations | No | N/A | Yes | Deep learning | 1 year | Unknown | DNN with ZILN loss function | Spearman correlation & normalized Gini coefficient | Automatic | None | Yes | No |

# 3 Methodology

## 3.1 Model selection

The analysis performed in this thesis consists of two parts. First, churn prediction is tested with different models, and then CLV prediction is tested separately from churn. Consequently, models are selected separately for each of the problems.

### 3.1.1 Heuristics

Even though prediction models have been studied widely in literature, they "have not found their way into managerial application" (Wübben & Wangenheim, 2008). The reasons are both technical and cultural. First, the implementation of complex models requires both time and money, so management will only implement them if clear positive results can be demonstrated. Second, many corporate executives rely on instinct in decision making rather than data, although they would argue otherwise (Wübben & Wangenheim, 2008). Furthermore, Wübben and Wangenheim (2008) compared simple heuristics with probabilistic models, Pareto/NBD and BG/NBD in particular. They conclude that there was no significant difference in the performance of probabilistic models and heuristics. For these same reasons some common heuristics are used as benchmark to validate the performance of more complex models.

In the aforementioned paper, Wübben and Wangenheim (2008) used the so-called hiatus heuristic to predict customer churn. It is a "simple recency-of-last-purchase analysis", where a customer is identified as churned after an arbitrarily selected time of inactivity. Persson and Ryals (2014) explored decision making among Nordic retail banks and they also noticed that the hiatus heuristic is a widely used rule of thumb in the context. Consequently, the hiatus heuristic is used as benchmark for churn prediction.

For CLV prediction, Wübben and Wangenheim (2008) used a simple heuristic, namely customers' past mean purchase frequency and extrapolated it to continue in perpetuity. This approach is the way to turn the churn prediction of the hiatus heuristic into a CLV prediction. This method will be used in this analysis as well to benchmark CLV predictions.

### 3.1.2 BTYD models

Even though previous work suggests that machine learning models outperform BTYD models (Chen et al., 2018), these probabilistic models should be included in the analysis. First, the study by Chen et al. (2018) is the only paper found explicitly comparing BTYD models and machine learning models in CLV prediction, and therefore there is room for further comparison in literature. Second, BTYD models are easier to implement, so if the results indicate that the predictive performance of BTYD models does not differ greatly from more complex models, they provide a decent alternative when resources are tight.

Pareto/NBD is considered the baseline in most papers comparing BTYD models, since it is the first model in the family (e.g. Jerath et al., 2011; Platzer & Reutterer, 2016). In this paper, the Pareto/NBD model is also used as the benchmark for classification task, but in order to make its implementation more efficient, the equivalent extension by Ma and Liu (2007) is used, which enables predictions at an individual customer level and utilizes MCMC-chains for model estimation.

In addition, Fader et al. (2005a) got similar predictive performance with their BG/NBD model and it is "vastly easier to implement". With a slight modification, this easy-to-implement model can yield even better results, which is what Hoppe and Wagner (2007) succeeded in with their CBG/NBD model. As mentioned earlier, CBG/NBD enables churn at time zero, which illustrates immediate customer churn. This assumption is highly relevant in the freemium game context, since a large share of players leave the game after giving it a short try. This fact combined with easy implementation and the industry-standard Poisson distribution for purchase rate make it the choice for the baseline model in this analysis.

It makes sense to select a more complex BTYD model as well. Jerath et al.'s (2011) PDO model did not perform any better than BG/NBD, and in addition it tends to predict a longer lifespan than BG/NBD. This may be unbeneficial for the freemium context. Platzer's (2008) CBG/CNBD-k model is fairly similar to the already chosen CBG/NBD, so examining it does not provide a different approach. The same applies to the G/G/NBD model by Bemmaor and Glady (2012).

The Pareto/GGG model by Platzer and Reutterer (2016) presents an alternative to the conventional Poisson purchase process. By replacing it with a gamma distribution, Platzer and Reutterer (2016) were able to yield better results. Furthermore, the model provides another new approach by taking customers' purchase timing patterns into account, thus

making individual customer predictions more accurate. The Pareto/GGG is thus a suitable second choice, providing clear differences to CBG/NBD. Both of the chosen BTYD models must, naturally, be combined with the Gamma-Gamma model of Colombo and Jiang (1999) to expand the prediction from number of purchases to CLV.

### 3.1.3 Machine learning models

There are numerous papers on churn prediction in gaming, as discussed earlier, and two different prediction methods are present: binary classification and survival analysis. In the papers focusing on classification, gradient boosting is found to perform best in both cases where they were tested (Kim et al., 2017; Milošević et al., 2017). Both Kim et al. (2017) and Milošević et al. (2017) speculate that a boosting algorithm performed so well due to the nature of the underlying data: behavioral gaming data is fairly unstructured and lacks patterns. Inspired by these papers, gradient boosting is included in the analysis.

Outside gaming, random forests have been found to perform well for classification (Coussement & Bock, 2013; Dror et al., 2012; Chamberlain et al., 2017) In addition, some papers on gaming also found that random forests performed well (Milošević et al., 2017; Drachen et al., 2016; Sifa et al., 2015). Having both gradient boosting and random forests in the analysis, two different meta-algorithms for generating multiple models within one are covered, namely boosting and bagging. The outcome and comparison of these two approaches is already interesting per se. Papers using the survival analysis method for churn prediction agree that conditional inference survival ensembles are a well-functioning technique (Periáñez et al., 2016; Bertens et al., 2017; Lee et al., 2019), and therefore it is included in the analysis as well.

To examine if model complexity improves results, two further classification models are included in the analysis. Logistic regression is used to illustrate how a very simple machine learning model performs, and it has been used in the literature comprehensively (e.g. Runge et al., 2014; Hadiji et al., 2014; Drachen et al., 2016). On the other end, deep neural networks are used to examine how a complex model works. The reviewed literature on churn prediction does not include deep learning, but deep multilayer perceptrons have been used in latest CLV prediction papers (Sifa et al., 2018; Chen et al., 2018). The model is modified to suit a classification task.

In CLV prediction, the literature is more limited, but both gradient boosting and random forests have been found to be suitable (Drachen et al., 2018; Vanderveld et al., 2016; Chamberlain et al., 2017). Hence, these models are tested in both churn and CLV

prediction. In addition to these models, recent papers have applied deep learning for CLV prediction in gaming. Deep multilayer perceptrons (Deep-MLP) have been used successfully by Sifa et al. (2018) and Chen et al. (2018). Convolutional neural networks (CNNs) were also used by Chen et al. (2018), but since the input data requires actual images of purchase time series per player, it is not applicable in this case. In both papers, the deep learning methods yielded better results than their more classical counterparts, e.g. random forests. Since the literature has not thoroughly researched these models in the context in question, it is only logical to include Deep-MLP in the analysis. Linear regression is used to compare if model complexity adds value, similarly to logistic regression in classification. Linear regression has also been used by Sifa et al. (2018) as a benchmark model.

## 3.2 Introduction to selected models

### 3.2.1 Pareto/NBD (HB)

The Pareto/NBD (HB) model was developed by Ma and Liu (2007), and its model assumptions are similar to the original model. However, it utilizes Markov chain Monte Carlo algorithm for model estimation and Monte Carlo simulations to calculate results. The model enables making predictions at an individual level, opposed to the original model, as well as acquiring a probability distribution instead of spot estimations. As mentioned, the other assumptions follow the Pareto/NBD model by Schmittlein et al. (1987):

- Individual customers make purchases according to a Poisson process with rate $\lambda$ while alive.
- The individual level purchase rates $\lambda$ are distributed gamma across customers with shape parameter $r$ and scale parameter $\alpha$.
- Customer lifetime follows exponential distribution at death rate $\mu$.
- Death rates $\mu$ are gamma distributed with shape $s$ and rate $\beta$ parameters across the population of different customers.
- Purchase process and lifetime process are independent of each other.

In practice, the model uses customers' transactional data to make predictions. A dataset contains a transaction log, where each transaction has data about its date and the customer who made the transaction. The *recency* and *frequency* values are calculated for each customer by the model. *Recency* refers to the total time between the first and the last purchase of the customer in the observation period. The time unit depends on the dataset

and it can be days or weeks, for instance. *Frequency* is the number repeat transactions, i.e. number of transactions in the observation period minus the initial one. Then the model utilizes MCMC-algorithm to draw parameters from the posterior distributions of the model's parameters. Monte Carlo simulation is used to predict future transactions with the help of the draws. The aforementioned assumptions are built into the predictions as well.

### 3.2.2 CBG/NBD

The CBG/NBD model was developed by Hoppe and Wagner (2007) and it is an extension of the classic Pareto/NBD and BG/NBD models of the BTYD family. The greatest difference to BG/NBD is that CBG/NBD allows customers to churn at time zero, i.e. immediately after they enter the service and before doing any purchases. Hoppe and Wagner (2007) also summarized the behavioral assumptions of the model:

- "Individual purchases of customers follow a Poisson process with rate $\lambda$."
- "The individual level purchase rates $\lambda$ are distributed gamma across customers with shape parameter $r$ and scale parameter $\alpha$."
- "Customers have a limited lifetime. At time zero and directly after each purchase there is a constant probability $p$ that the customer will become inactive. Therefore, dropouts follow a central geometric distribution with dropout probability $p$."
- "The individual level dropout probabilities $p$ are distributed beta across customers with the shape parameters $a$ and $b$."
- "Purchase process and lifetime process are independent of each other."

Parallel to Hoppe and Wagner, Batislam et al. (2007) were developing a similar model with similar modifications as in CBG/NBD, and they coined it MBG/NBD. "M" stands for *modified* in this case, as it was also derived from the BG/NBD model. In this analysis, the MBG/NBD model is used in practice instead of the similar CBG/NBD model, because there is a ready-made implementation of it in R's *BTYDplus* package.

The technical implementation is rather similar to Pareto/NBD (HB). Transactional data is used as input, but no draws from the distributions are done, and therefore MCMC-algorithm nor simulations are used in the prediction process. The aforementioned assumptions drive the probabilistic predictions, instead. Lifetime value is calculated by estimating the expected average transaction value, which is multiplied by the number of future transactions.

### 3.2.3 Pareto/GGG

Platzer and Reutterer (2016) pointed out that the Poisson purchase process utilized in BTYD models does not take purchase regularities into account. In the Poisson purchase process, the time between purchases is exponentially distributed, and therefore the highest probability of repurchase is immediately after the previous one. In addition, the memoryless property of exponential distribution does not consider the timing of the previous purchase. Even though the recency and frequency of two customers would be identical, the other is more likely to be churned if there has been a longer idle period after regular purchases (Platzer & Reutterer, 2016).

Hence Platzer and Reutterer (2016) developed a new model, Pareto/GGG, which replaces the negative binomial distribution (NBD) with a mixture of gamma distributions (GGG, gamma-gamma-gamma) to include regularity. Thus, the former focus on "count process" is abandoned to capture "timing process". In addition to recency and frequency statistics, the model only requires one additional summary statistic for historical intertransaction times (ITTs). Platzer and Reutterer (2016) also summarized the assumptions of the model:

- Transaction process: While alive, ITTs ($\Delta t_j = t_j - t_{j-1}$) follow a gamma distribution, with shape parameter $k$ and rate parameter $k\lambda$:

  $\Delta t_j \sim \text{Gamma}(k, k\lambda)$
- Dropout process: A customer remains alive for an exponentially distributed lifetime $\tau$:

  $\tau \sim \text{Exponential}(\mu)$
- Heterogeneity across customers: Individual-level parameters $\{k, \lambda, \mu\}$ follow gamma distributions across customers independently:

  $k \sim \text{Gamma}(t, \gamma)$

  $\lambda \sim \text{Gamma}(r, \alpha)$

  $\mu \sim \text{Gamma}(s, \beta)$

The implementation is similar to the other BTYD-models. In addition to recency and frequency, historical intertransaction times are used to from the transaction data. Then the model draws from the posterior distributions of the model's parameters, and these draws are used in predicting the number of future transactions. As in CBG/NBD, future lifetime value is predicted by estimating the future average transaction value and it is multiplied by the number of expected transactions.

### 3.2.4 Linear regression

The widest area of application for regression analysis is to investigate relationships between one variable and multiple other variable (Yan & Su, 2009), but regression can also be used for prediction and it is a simple method. Linear regression was developed at the beginning of the 19th century (Yan & Su, 2009). With one predictor variable, a linear regression model can be visualized as a two-dimensional graph, where Y-axis represents the values of the target values and X-axis represents predictor values. The assumption is that the values on both axes are continuous instead of discrete. The model then fits a line across the graph so that the distances between the line and the observation points are minimized. According to Yan and Su (2009), the typical, and oldest, loss function for this fit is *ordinary least squares* (OLS), where the distances are squared to ignore the direction of the error (below or above the line, minus and plus respectively). This loss function is also used in this analysis. New observations are then predicted by placing the observation on the line based on its predictor value on X-axis, and the equivalent Y-axis value can be taken.

### 3.2.5 Logistic regression

Logistic regression is an old model with roots in the 19th century, and it has been widely applied in statistics (Cramer, 2002). Despite its name, logistic regression is suited for classification tasks (Peng et al., 2002).

The simple supervised machine learning model can be thought of as a modification from the simple linear regression. In logistic regression, however, the target value is categorical instead of continuous. Therefore, observations are not spread across the Y-axis. Figure 3 illustrates this well in its simplest form with one continuous predictor. The red points are observations, where the target value is either zero or one. The predictor variable is continuous and takes values from 40 to 160. Logistic regression then fits a curve representing *probability* that a value on X-axis is one. When the curve crosses the 0.5 threshold on Y-axis, all observations that are right from it on the X-axis are classified as one (Peng et al., 2002).

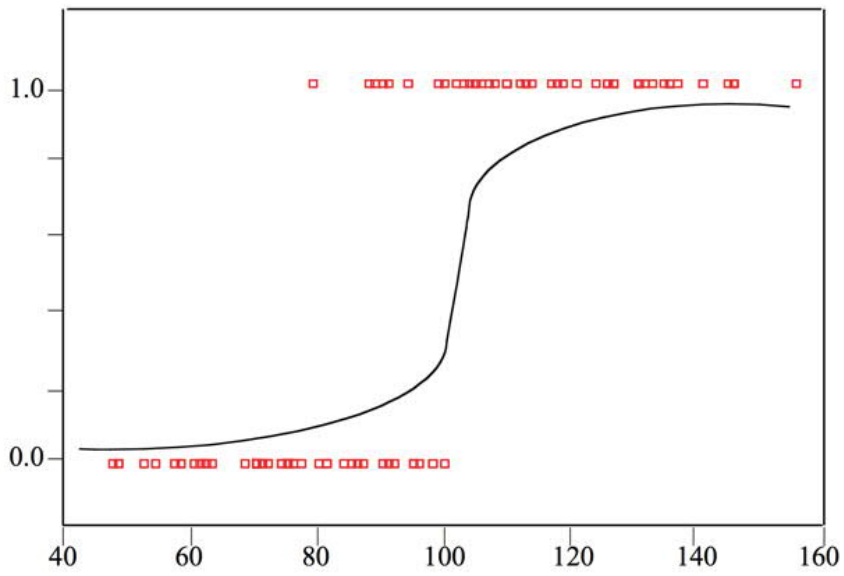


*Figure 3. Illustration of logistic regression with one predictor (Peng et al., 2002).*

### 3.2.6 Random forests

Random forests are an ensemble method that is based on bagging, short for 'bootstrap aggregating'. According to Breiman (1996), bagging "is a method for generating multiple versions of a predictor and using these to get an aggregated predictor". The multiple versions, or multiple training datasets, are generated through random selection with replacement. A separate model is then fitted to each of these samples. Usually, these models are decision trees, which is also the case in the literature reviewed in this thesis (Milošević et al., 2017; Sifa et al., 2015). Lastly, the final result is achieved by aggregating the results of the multiple models, either by determining the average for regression or by voting for classification (Lemmens & Croux, 2006). Figure 4 illustrates the logic of bagging visualized as diagram flow (Maldonado et al., 2014).

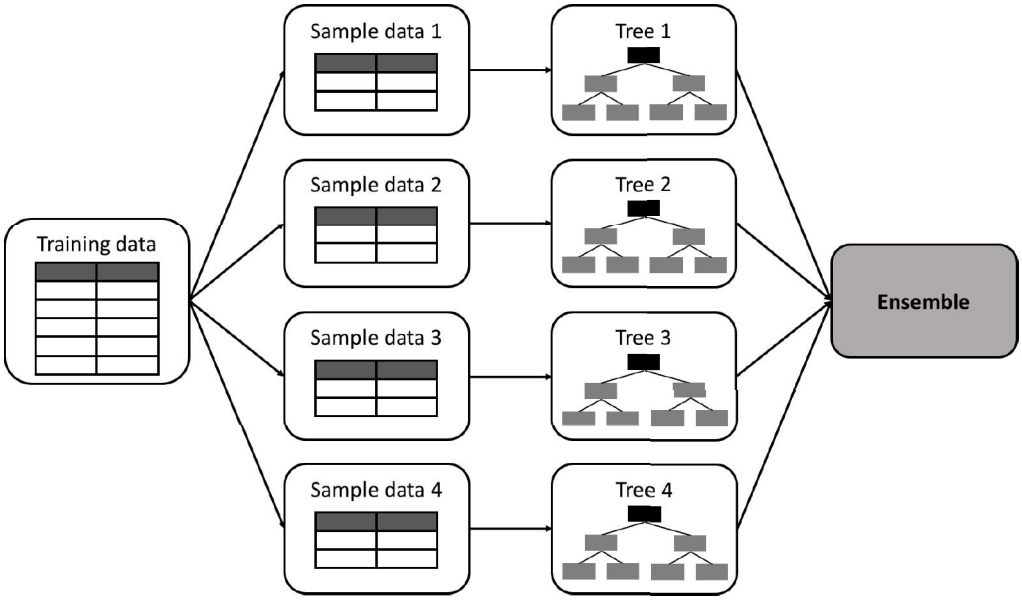


*Figure 4. Illustration of bagging process (modified from Maldonado et al., 2014).*

To elaborate on Figure 4, multiple different samples are drawn from the original dataset and each of these are given to one decision tree. Figure 5 illustrates a typical, but very small, decision tree. A decision tree is made of nodes that represent splits in dataset variables. For instance, an observation from the data is given to a decision tree, and the first node examines whether a given variable is below or above some threshold value. Depending on the answer, either the left or the right branch of the tree is used for further decisions. The branches do not have to be of equally deep. Eventually, the final prediction value depends on which node was the last one used for the observation.

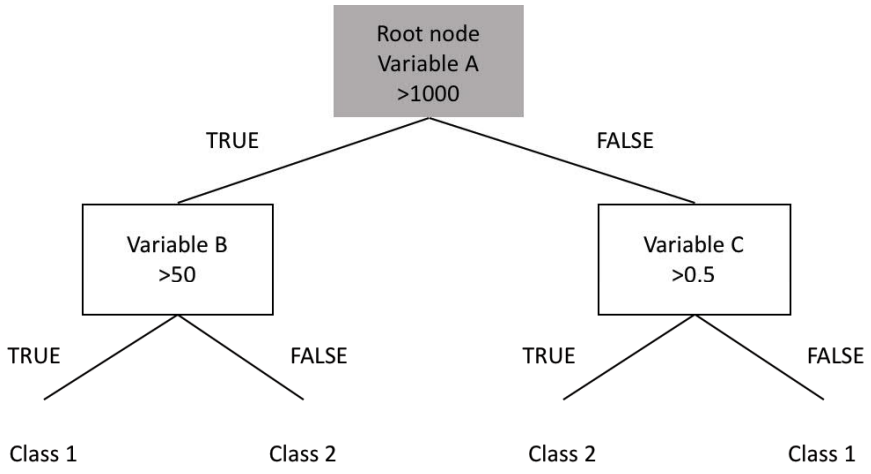


*Figure 5. Illustration of a decision tree for classification.*

Random forest creates multiple decision trees, and because every tree receives a different training dataset, the trees are different. Consequently, each tree predicts the final value in its own way. In classification problems, random forest selects the final value based on voting so that the value chosen is predicted by majority of the used trees, and each tree has an equal vote. In regression problems, the prediction values of the trees are averaged to one number.

Possibly the most common version of random forest is introduced by Breiman (2001) and his definition goes as follows:

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

Breiman (2001) proves in his work that the aggregate results of the model decrease the expected prediction error, also known as generalization error. As he states, "the generalization error for forests converges a.s. to a limit as the number of trees in the forest becomes large". This prevents the random forest from overfitting.

The random selection in Breiman's model is done by random feature selection. Hence, not all variables and not all data points are used to grow decision trees. The observations, that are left out, are used for generating out-of-bag estimates. The model runs the decision trees on the out-of-bag data to give an estimate for generalization error.

### 3.2.7 Gradient boosting

Gradient boosting is a machine learning model based on boosting. Boosting is another ensemble method, like bagging, in which multiple models are used sequentially instead of simultaneously (as in bagging). Boosting improves the performance of learning algorithms (Freund & Schapire, 1996) by creating a new training dataset based on the results (given weights) of the previous model. The final model is thus a result of multiple iterations of submodels. Figure 6 illustrates the process (Maldonado et al., 2014).

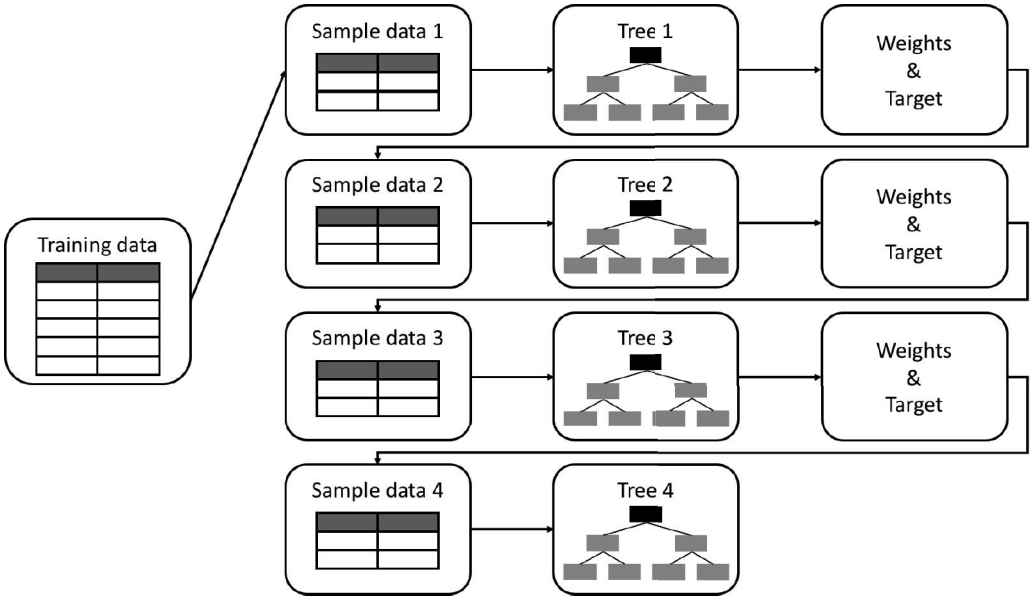


*Figure 6. Illustration of boosting process (modified from Maldonado et al., 2014).*

The popular gradient boosting model was developed by Friedman (2001; 2002). It creates the multiple models by "fitting a simple parameterized function (base learner) to current 'pseudo'-residuals by least squares at each iteration". These pseudo-residuals are then minimized by the loss function of the model at each step (Friedman, 2002). As is the case in random forests, the multiple models of gradient boosting are also usually decision trees.

In practical terms, how the model works is that the first decision tree receives a sample of the training data. The residuals, which are the differences between actual values and predicted values, are then used for the next decision tree. The new tree uses another sample from the training data, and it tries to predict the residuals of the previous tree, not the target variable. The difference between the first tree's residuals and the predictions of the second tree are the 'pseudo'-residuals. The third tree again tries to predict the pseudo-residuals of the second tree with a new sample etc. The errors decrease little by little, leading to fairly accurate end results. It is worth noting that only the first tree in the "chain" of trees directly tries to predict target variable values. Once the built model receives new data to make predictions on, each observation goes through the entire set of trees in order to get a prediction value.

In the function estimation, observations consist of a random output variable $y$ and a set of random explanatory variables $x = \{x_1, \ldots, x_n\}$ (Friedman, 2001). With the help of a training sample of the dataset, the "goal is to obtain an approximation of the function $F(\mathbf{x})$

mapping $\mathbf{x}$ to $y$, that minimizes the expected value of some specified loss function $L(y, F(\mathbf{x}))$ over the joint distribution of all $(y,\mathbf{x})$-values". Typical loss functions in use are squared-error and absolute error for regression tasks and negative binomial log-likelihood for classification tasks. Boosting stops when a predefined number of iterations is reached by the model.

### 3.2.8 Conditional inference survival ensembles

Conditional inference survival ensembles, first introduced by Hothorn, Hornik and Zeileis (2006), are a technique within survival analysis. Contrary to conventional classification task, the focus of survival analysis is the time until the expected event occurs (Periáñez et al., 2016). Periáñez et al. (2016) point out that a key characteristic of survival analysis is that the underlying data is censored, in other words the time of occurrence for the expected event is unknown. The expected event can take many forms, for example death, recovery, finding a job, or in our case churn. In survival analysis, only two datapoints are needed to predict survival time: observed time and whether the event happened within the observation period or not. If no event occurred, the observed time is usually equal to the length of the observation period. The output for each observation is a prediction how many days (or other units of time) it will take until the expected event will take place.

In more detail, conditional inference survival ensembles are a survival ensemble technique (Periáñez et al., 2016), where the "survival ensemble lies in growing a set of survival trees". A weighted Kaplan-Meier estimate is determined for each tree in order to distinguish survival characteristics in nodes (Bertens et al., 2017). Node split is done based on linear rank statistics so that the difference between nodes is maximized. The split is performed in two steps. The optimal variable for split is first chosen based on covariates and their relationships to each other, and then the split point is defined "by comparing two sample linear statistics for all possible partitions of the split covariate" (Bertens et al., 2017). Due to this two-step approach to node partition, conditional inference survival ensembles are robust to overfitting and, unlike random forests, they are not biased towards predictors with many splits or missing data (Periáñez et al., 2016). When used for classification task, the survival time is used to make predictions on survival status at selected time $T$.

### 3.2.9 Deep multilayer perceptron

Deep multilayer perceptron (deep-MLP) is one type of architecture for neural networks, which consists of an input and output layer and multiple hidden layers between them (Schmidhuber, 2015). What makes deep multilayer perceptron 'deep' is that it has multiple hidden layers instead of one, as in the more traditional multilayer perceptron architecture. Otherwise it is similar to the widely used multilayer perceptron structure.

When a new deep-MLP is constructed, the input layer usually consists of as many neurons as there are variables to be used in observation data. These values then travel through the deep layers and they get modified along the way by neurons and weights between them. Neurons use an *activation function* for tuning incoming values from the previous layer, and there are multiple options for these functions. The activation function and the multiple layer structure enable solving non-linear problems, and it performs supervised learning via backpropagation algorithm (Schmidhuber, 2015). In *backpropagation*, the weights between neurons are iteratively optimized based on a loss function in the output layer. The loss function is usually a minimization problem for least mean squares or a similar metric.

In other words, when the model is trained, the first prediction is done with randomly assigned weights between neurons, and once the observations from training data "travel" through the layers, the loss function (prediction error) is calculated. This is called *forward propagation*. Then the weights are changed in order to decrease prediction error starting from the last layer and all the way back to the input layer. This is the *backpropagation*. The process is performed multiple times to minimize the loss function.

The deep multilayer perceptron is also normally a fully connected network, meaning that a neuron in any layer is connected to each neuron in the following layer. In regression tasks, the output layer consists of one neuron to represent the one single value that is predicted. In classification tasks, the output layer has as many neurons as there are classification options, and each neuron calculates a probability that the observation belongs to the given class.

## 3.3 Evaluation criteria

The results of the analysis are evaluated separately for churn and CLV prediction, as the prediction problems are different. Churn prediction is a binary classification problem, where players are predicted to churn or not to churn. CLV prediction, on the other hand, is

a regression problem where a numerical value (monetary value in this case) is the output of the prediction. In both prediction tasks, data from a shorter observation period is used to predict churn status and monetary value after the first 30 days of the players.

In a binary classification problem, one class is labeled positive (in our case churner) and the other class negative (Burez & Van den Poel, 2009). Comparing the predicted classes to actual classes, a so-called *confusion matrix* can be drawn to illustrate the possible outcomes, which is visible in Table 4:

*Table 4: Confusion matrix of binary classification*

| | **Predicted positive** | **Predicted negative** |
|---|---|---|
| **Actually positive** | True positive (TP) | False negative (FN) |
| **Actually negative** | False positive (FP) | True negative (TN) |

In confusion matrix, true positives and true negatives are accurate predictions, i.e. the predicted class matches with the actual class. Numerous performance metrics can be derived from the matrix (Burez & Van den Poel, 2009). For instance, possibly the most common metric is *accuracy*, which tells what ratio of all the predictions were correctly predicted (TP + TN / all observations). The articles discussed in the literature review of this thesis mostly use two metrics derived from the confusion matrix, *F-score* and *AUC*.

The F-score is a metric that combines two further metrics from confusion matrix, *precision* and *recall* (Sokolova et al., 2006). Precision is a measure to determine how many of the positive predictions were accurate (TP/TP+FP), and recall how many of the relevant observations were selected accurately (TP/TP+FN). F-score aims at balancing these two metrics into one:

$$F-score = \frac{2}{1/precision + 1/recall} \quad (1)$$

The F-score varies between 0 and 1, and a higher score indicates a better result. The metric also has other used names in the literature, i.e. $F_1$-score and F-measure. The combination of precision and recall as well as the more sophisticated approach than accuracy make it a relevant choice to evaluate prediction results. In addition to F-score, precision and recall are however included to provide insight to the nature of prediction errors.

The F-score, however, has one downside. It may vary depending on the ratio of positive and negative observations. For instance, a dataset with 90% (actual) negative observations may not be predicted as accurately as a dataset with 50/50 distribution of positives and negatives. The AUC metric is independent of this ratio (Burez & Van den Poel, 2009). AUC is derived from the *ROC curve*, which furthermore is based on *ROC graph*.

ROC is short for Receiver Operating Characteristic. A ROC graph is a two-dimensional visualization that shows true positive rate and false positive rate on the axes (Burez & Van den Poel, 2009). Binary classifiers classify the observations based on a given probability threshold between 0 and 1. For example, if the threshold is put to 0.5 and the classifier gives an observation a probability of 0.7 of being positive, it is classified as positive. A ROC curve is consequently plotted in the ROC graph by putting a point in the graph for each threshold and their respective true positive and false positive rates. These points then make a line, or rather a curve, in the graph. Figure 7 shows a ROC graph and ROC curves in practice:

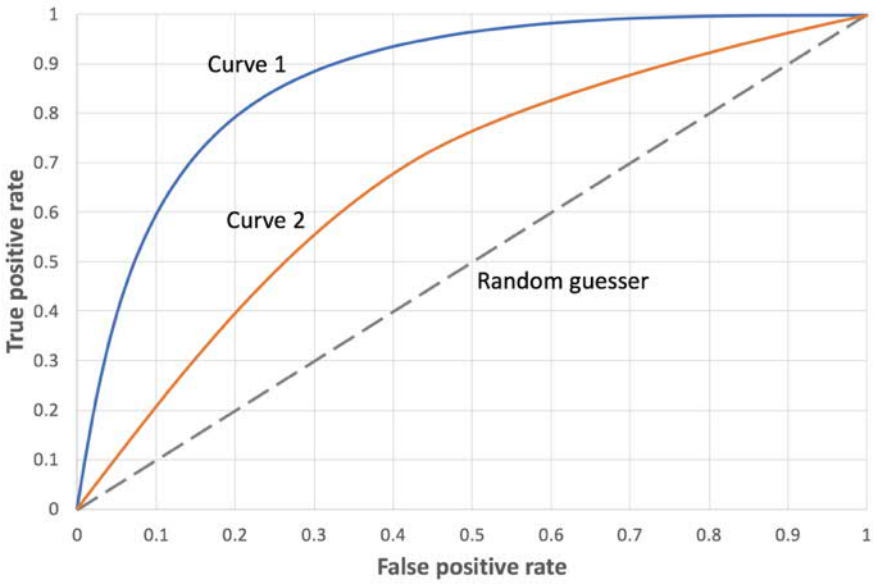


*Figure 7. Illustration of a ROC graph.*

In Figure 7, two different classifiers are presented by curves 1 and 2. Since the true positive rate of curve 1 is higher at all values of false positive rate, classifier 1 performs better than classifier 2 at all thresholds. This performance can be summarized in one metric, namely AUC. AUC is short for Area Under the ROC Curve (Sokolova et al., 2006) and it is commonly used, because it ignores the manual choosing of the threshold. AUC is a number between 0 and 1, which indicates what share of the ROC graph (the whole

square) is under the curve. The higher the number, the bigger the share of the graph under the curve. A good benchmark for classifiers is the so-called *random classifier*, aka random guesser, which is basically a classifier that guesses the class randomly. In random classification, the rate of true and false positives goes hand in hand, which is also visible in Figure 7. The AUC of a random classifier is 0.5, since half of the graph lies under the "curve". If a classifier's AUC is under 0.5, it performs worse than guessing the result randomly.

In practice, the desired point on the line to be used for the classifier is a business decision and it depends on the context. For instance, a cancer diagnosis classifier would want to maximize the true positive rate, because the consequences of missing a positive case can be lethal. The damage from a false positive diagnosis is smaller, so a higher false positive rate is acceptable in this example.

The papers on CLV prediction in gaming discussed in this thesis use MSE-based metrics to measure prediction performance. These include RMSE (Sifa et al., 2015) and NRMSE (Drachen et al., 2018; Sifa et al., 2018; Chen et al., 2018). MSE, short for mean-squared error, has been a dominant metric to measure accuracy of models (Wang & Bovik, 2009). It calculates the difference between actual values and the predicted values and takes the average of the squared differences. More formally, Wang & Bovik (2009) present MSE to be:

$$MSE = \frac{1}{N}\sum_{i=1}^{N}(x_i - y_i)^2 \tag{2}$$

Where $x_i - y_i$ is the difference between the actual and predicted values and $N$ is the number of observations. RMSE (rooted mean-squared error) is the square root of MSE, and NRMSE (normalized rooted mean-squared error) is the normalized version of RMSE, where the error is shown as percentage to actual values. RMSE is measured in the same unit as the dataset itself, so the results are not comparable between different datasets. NRMSE, on the other hand, enables the comparison of different datasets (Chen et al., 2018). Since the analysis of this thesis does not cover multiple datasets, RMSE is chosen to be used as the performance metric for the CLV prediction.

# 4 Case introduction

## 4.1 Case company and game

I receive the data for the empirical analysis from a Finnish mobile gaming company. The company has cumulatively reached over 100 million organic downloads and the data is retrieved from their most popular game. In the game, the player fights against other player(s) or CPU in order to achieve points, new accessories and tools to be used. Multiple different games modes are available to choose from. The game utilizes the freemium business model and money can be used to for example purchase new accessories and further advancement in the game.

## 4.2 Introduction to data

The dataset (received as a csv-file) represents a cohort of 63255 new players from one week in February 2020 from the USA and includes data from their first 37 consequent days so that the last new players of the cohort also have data from 30 days. Each row represents data from each player's each active day. For example, if a player logged in to the game twice on Monday, three times on Tuesday and not even once on Wednesday, this generates two rows of data, where data is aggregated on a daily level.

The dataset describes players' device, monetary spend, activity in different game modes, videos watched as well as the gains and spending of different in-game currencies. Table 5 summarizes the 41 parameters of the dataset.

*Table 5: Summary of BTYD models*

| Parameter type | # of parameters | Data type |
|---|---|---|
| User ID | 1 | String |
| Device and game version | 3 | String |
| Date of login and first registration | 2 | Date |
| In-app purchases per date and cumulative | 3 | Float & Integer |
| Number of sessions (logins) | 1 | Integer |
| Number of different game modes and success | 12 | Integer & Float |
| Number of different activities within the game (e.g. upgrades) | 4 | Integer |
| Number of different types of videos watched | 4 | Integer |
| Gains of game currencies | 5 | Integer |
| Spend of game currencies | 4 | Integer |
| Daily and cumulative in-game progress | 2 | Integer |

15% of the cohort stayed alive until the end the period, which in my opinion is a decent share for a freemium game. However, less than 1% of the cohort made purchases, making the data highly imbalanced. On the other hand, this represents the usual customer behavior pattern in freemium mobile games. Figure 8 shows the spend distribution among these few players, and it is visible that most players' CLV is quite low. Two thirds of these spending players spent at maximum 10 USD in total during the first 30 days.

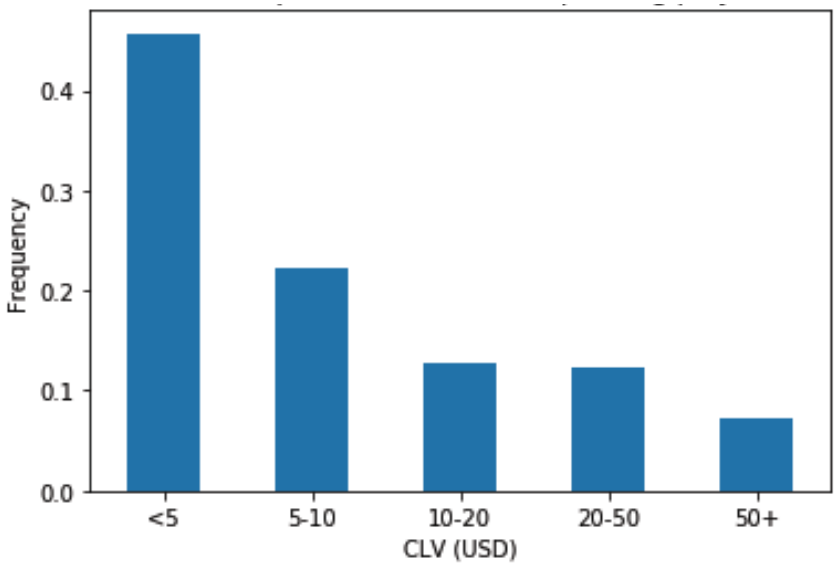


*Figure 8. Relative spend distribution among spending players (USD)*

While 15% of the cohort stays alive until the end of the 30-day period, the churners leave the game very quickly. Figure 9 illustrates the empirical cumulative distribution function (ECDF) of the players' last sessions measured by number of days since their initial registration. Because the graph showcases cumulative distribution, the counts of the next day are always added on top of the previous one. The graph shows that around 40% of all registered players have not played a single session after the day of registration, and additional 10% only played the game during the registration day and the following day after it. After that, the distribution is fairly even between different days until the end of the period. The observations in the graph do not form a line, because the underlying measurement value is discrete, namely number of days. As churn is defined to occur after seven days of inactivity, the players who have their last session on day 24 or later are interpreted to stay alive.

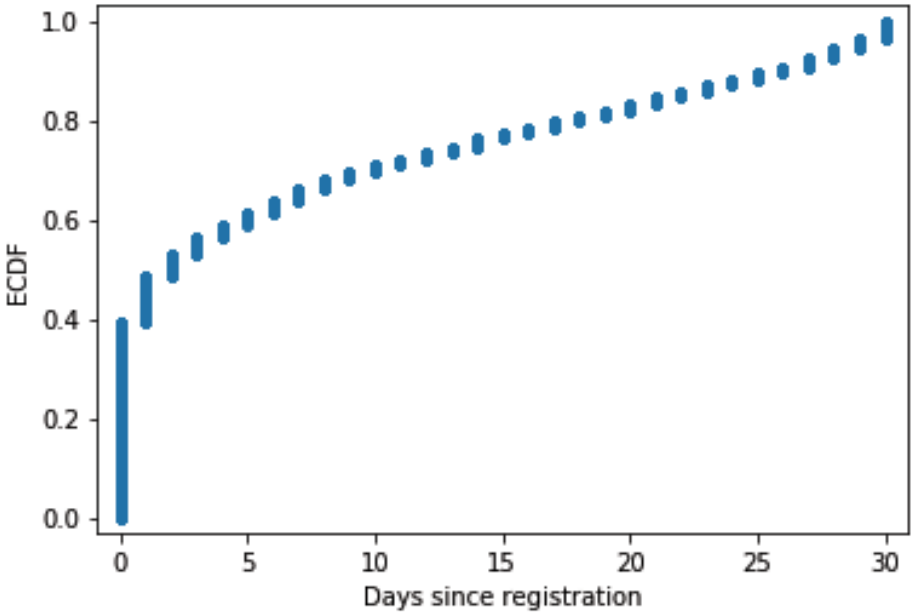


*Figure 9. Occasion of last session by player, cumulative distribution*

## 4.3 Choice of tools for analysis

Most of the work is done in Python, where the choice of software is JupyterHub, and the rest is done in R with RStudio as the choice of software.

R is used for testing the BTYD models as well as conditional inference survival ensembles. The packages *BTYD* and *BTYDplus* enable the implementation of BTYD models, and packages *survival* and *party* are needed for conditional inference survival ensembles.

In Python, packages *pandas* and *numpy* are used for data preparation and *matplotlib* for visualization. As the data is first prepared in Python for all models, the prepared dataset is transferred to RStudio as a csv-file. The building and measuring of gradient boosting and random forests are done with the *sklearn* package in both classification and regression problems. The neural networks are built using the *keras* package, but the models use the same train-test-split and performance measures from *sklearn* package as the other machine learning models.

# 5 Analysis and results

## 5.1 Data preparation

Two separate datasets are prepared from the received one. The BTYD models require granular data, i.e. each row represents a single transaction or a day of transactions (since the transactions from one day are merged later on if not done before). In order to prepare this dataset, the only preparation required are changing parameter datatypes (e.g. dates were strings) and sorting the data by user and date.

The other prepared dataset aggregates data so that each row represents one player. The target variables (churn and CLV) are calculated for the first 30 days of each player. To be consistent with previous literature, the most common definition for churn is used, which is seven days of inactivity. So, if a player has not logged in at all during the last seven days of the whole 30-day-period, they are labelled as churned. Categorical variables are encoded to dummy variables so that they are interpretable by the machine learning models.

Since the dataset includes 37 days so that even the last players of the cohort have 30 days of data, the period length is normalized. In other words, the days coming after the $30^{th}$ day for each player are ignored. The predictor variables are measured from a shorter observation period only. As the aggregation by player is performed, all data after the observation period (measured in days) is also ignored to simulate the real-life setting.

The survival analysis requires its own target variables, namely survival time and survival status for the entire 30 days. In survival analysis, the data is censored, when the event has not occurred during the period. This does not mean that the event could not occur later in the future after the period. In this case, the event is churn and its occurrence or absence is the event status. It is interpreted in a way that the churn itself occurs after the week of inactivity has passed.

Prediction performance is measured for three different lengths of observation period. In other words, the models predict results after knowing what the players have done during the first day, after one week and after two weeks. Consequently, three different versions of the aggregate dataset are produced to account for the observation periods. The first dataset includes predictor variables that have aggregated data from each player's first day only, the second from the first seven days and the last from the first 14 days.

Throughout the analysis, 70% of data is used to train the models and 30% is left to test their performance.

## 5.2 Model-specific notes on implementation

### 5.2.1 BTYD and survival

As mentioned, the dataset includes data from a total of 37 days in order to provide 30 days even for the last new players of the weekly cohort. The data is normalized so that each player has data from their first 30 days, disregarding the extra days for coherence. However, the implementation of BTYD models requires an actual cutoff date to determine the observation period, and therefore the observation period is a little longer for some players in BTYD models. The models are still comparable with each other, as the total length of the period does not change dramatically in absolute terms. The percentual change is of course rather large.

As any form of survival analysis, conditional inference survival ensembles predict the number of days (or weeks, months…) each person survives until the measured event takes place for them. As this analysis utilizes the survival predictions for binary classification, these integer values are manually turned into "churned/alive" predictions by setting a threshold to the end of the 30-day-period. Predictor variables are normalized, because their ranges vary widely. By normalizing the data, in other words putting them on the same scale, the performance can be improved significantly. The chosen scaling technique is to normalize each variable onto a scale from zero to one. This is performed in R with an own function that I built.

### 5.2.2 Machine learning models

In both classification and regression tasks, the hyperparameters of the gradient boosting and random forest models are tuned. The optimized hyperparameters for the gradient boosting are the learning rate (shrinking the contribution of each tree), number of estimators (i.e. trees in the forest), maximum number of predictor variables, and maximum depth of the built tree of the model. The same hyperparameters are optimized for random forest models as well, but without learning rate, which is not supported by random forests. The optimization itself is performed using *grid search cross-validation*, a technique where each combination of the possible hyperparameter values is fitted separately, and the combination with the best results is chosen. The cross-validation is tenfold, as in most of the previous literature, and AUC score and mean squared error are used to determine the best fit for classification and regression tasks respectively.

The synthetic minority over-sampling technique (SMOTE) is applied to machine learning classifiers in order to augment additional data on players who stay alive until the end of the 30-day period. Both Gradient boosting and random forest are therefore run with both original data and augmented data separately. This augmentation is not applied to the regression task, which is in line with previous literature. Only one of the reviewed articles (Sifa et al., 2018) utilized the technique, and even they developed their own SMOTE algorithm for the lifetime value prediction.

Data preparation for the algorithms also include normalizing predictor variables, because their ranges varied widely. By normalizing the data, in other words putting them on the same scale, the performance can be improved significantly. As for survival analysis, the chosen scaling technique is to normalize each variable onto a scale from zero to one, but with the help of *MinMaxScaler* from the *sklearn* package instead of an own function.

As mentioned, logistic regression is used as a machine learning baseline to examine how a simple machine learning model performs for classification. Linear regression serves the same purpose for regression. The highlight simplicity, SMOTE is not used for these models and only the variables with significant predictive power in gradient boosting and random forest, six and seven in total respectively, are included (more on feature importances in Chapter 5.3).

### 5.2.3 Deep-MLP

The structure of the deep multilayer perceptron model is optimized by manually changing the number of layers and nodes in each layer to find a version that performs best in terms of the evaluation criteria. This optimization is done manually and therefore only a dozen different options are tested. However, the optimization neither improves nor weakens performance, and therefore a similar structure to Chen et al's (2018) model is used in the final version. This includes an input layer with a number of nodes equal to predictor variables, three hidden layers with 300, 200 and 100 nodes respectively, and an output layer of one or two nodes for regression and classification respectively.

The activation function, optimization algorithm as well as kernel initializer are chosen by industry standards, and they are not questioned. The used activation function is *ReLU* (rectified linear unit), the optimizer is *Adam*, and the kernel initializer is *He uniform*. The last layers of the classification and regression models are different, as the regression model outputs one single continuous value with one node and the classification model

requires two nodes representing the probabilities of the binary values. The *softmax* activation function is used for the binary layer.

During training, mean squared error and categorical crossentropy are used for optimizing the loss function in regression and classification models respectively. A fifth of the training data is used to validate the loss function. The rounds of forward and backward propagations for optimization, called epochs, are stopped after three rounds of unimproved results with the help of an early stopping monitor.

Both SMOTE balancing and normalization of predictor variables are performed similarly to gradient boosting and random forest for the same reasons.

## 5.3 Results

### 5.3.1 Churn prediction

The results of the churn prediction models are presented in Table 6. The precision, recall, F-score and AUC were measured for each model and each observation period. The best value for each column is bolded for readability. Precision, recall and F-score are measured so that churn is the positive event in the underlying confusion matrix.

The table includes some N/A-values, especially for the one-day observation period. The hiatus heuristic, BTYD models as well as survival analysis require data from multiple days in order to make reasonable predictions. In the case of the hiatus heuristic, the observation length must be at least as long as the definition for inactivity. The lack of precision, recall and F-score values for BTYD models and deep multilayer perceptron stems from the predicted values themselves.

In both cases, the models only output a probability of being alive instead of a binary and absolute prediction. The confusion matrix used to calculate the F-score requires a binary prediction in order to be built. On the other hand, gradient boosting and random forest can produce both binary predictions and probabilities. Both binary predictions and predicted probabilities can be used to measure AUC, but predicted probabilities are used where applicable. The rationale behind this is to get probabilities before varying the threshold of the ROC curve, as explained in Chapter 3.3.

*Table 6: Results of churn prediction by model and observation period*

| Observation period | | 1 day | | | | 7 days | | | | 14 days | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **Model** | **Performance metric** | Precision | Recall | F-score | AUC | Precision | Recall | F-score | AUC | Precision | Recall | F-score | AUC |
| Hiatus heuristic | | N/A | N/A | N/A | N/A | **0.94** | 0.52 | 0.67 | 0.67 | **0.92** | 0.84 | 0.88 | 0.72 |
| Pareto/NBD (HB) | | N/A | N/A | N/A | N/A | N/A | N/A | N/A | **0.76** | N/A | N/A | N/A | **0.79** |
| Pareto/GGG | | N/A | N/A | N/A | N/A | N/A | N/A | N/A | 0.73 | N/A | N/A | N/A | 0.76 |
| Cond. inf. survival ensembles | | N/A | N/A | N/A | N/A | 0.87 | 0.98 | **0.92** | 0.55 | 0.86 | **0.98** | **0.92** | 0.54 |
| Logistic regression | | 0.86 | **0.99** | **0.92** | 0.62 | 0.86 | **0.99** | **0.92** | 0.74 | 0.87 | **0.98** | **0.92** | 0.78 |
| Gradient boosting | | 0.86 | **0.99** | **0.92** | 0.63 | 0.86 | 0.98 | **0.92** | **0.76** | 0.87 | **0.98** | **0.92** | **0.79** |
| Gradient boosting + SMOTE | | 0.86 | 0.97 | 0.91 | 0.53 | 0.88 | 0.94 | 0.91 | 0.70 | 0.89 | 0.94 | 0.91 | 0.75 |
| Random forest | | 0.86 | **0.99** | **0.92** | 0.63 | 0.86 | 0.98 | **0.92** | **0.76** | 0.87 | 0.98 | **0.92** | **0.79** |
| Random forest + SMOTE | | **0.88** | 0.80 | 0.84 | 0.62 | 0.91 | 0.80 | 0.85 | 0.75 | **0.92** | 0.80 | 0.86 | 0.78 |
| Deep-MLP | | N/A | N/A | N/A | **0.64** | N/A | N/A | N/A | **0.76** | N/A | N/A | N/A | **0.79** |
| Deep-MLP + SMOTE | | N/A | N/A | N/A | 0.63 | N/A | N/A | N/A | **0.76** | N/A | N/A | N/A | **0.79** |

The relative feature importances, i.e. the weights of predictor variables in the predictions, of gradient boosting and random forest classifiers are presented in Figure 10 for the seven-day observation period. The Y axis represents the weight magnitude (they add up to a total of one) and X axis represents the predictor variables. The names of the variables are hidden to keep the game unidentifiable. In both classifiers, only a few variables have predictive power. For gradient boosting, the first variable is number of sessions followed by number of games played. For random forest, the first two variables are the same, followed by usage and gains of different in-game currencies. Please note that the scale is slightly different in the two graphs.

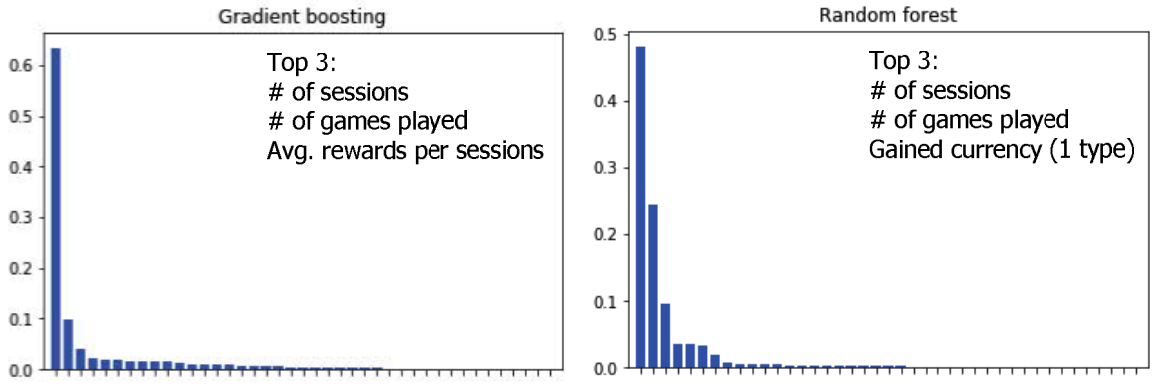


*Figure 10. Relative feature importances in gradient boosting and random forest classifiers, 7-day period.*

### 5.3.2 CLV prediction

Table 7 presents the results of the CLV prediction. As mentioned, the lifetime here is a maximum of 30 days, which is equal to the used dataset. The mean absolute error (MAE) and the root mean squared error (RMSE) are measured for each model and each observation period. The model performance is evaluated mainly by RMSE, but MAE is also provided to illustrate the magnitude of prediction errors. The MAE measurements are in U.S. dollars. The best value for each column is bolded for readability.

The N/A-values are due to the same reason as in churn prediction, namely that the heuristic and the BTYD models require data from multiple days.

*Table 7: Results of customer lifetime value prediction by model and observation period*

| Observation period | 1 day | | 7 days | | 14 days | |
|---|---|---|---|---|---|---|
| **Model** **Performance metric** | MAE | RMSE | MAE | RMSE | MAE | RMSE |
| Past mean purchase | N/A | N/A | 0.19 | 7.29 | 0.08 | 4.21 |
| CBG/NBD | N/A | N/A | 0.24 | 3.74 | 0.13 | **3.49** |
| Pareto/GGG | N/A | N/A | 0.19 | 3.71 | 0.11 | **3.49** |
| Linear regression | **0.11** | **1.87** | 0.06 | 1.13 | 0.07 | 6.09 |
| Gradient boosting | 0.28 | 3.29 | 0.24 | 1.98 | 0.09 | 6.12 |
| Random forest | 0.12 | 1.89 | **0.05** | **1.12** | **0.06** | 6.10 |
| Deep-MLP | 0.15 | 1.88 | 0.12 | 1.26 | 0.10 | 6.16 |

The MAE and RMSE metrics in Table 7 provide a standardized way to compare model performance. In order to get deeper insight into the nature of prediction errors, i.e. their magnitude and direction, the mean absolute error (MAE) is split to mean prediction error separately for all players, spending players and non-spending players. The results are presented in Table 8. For instance, the gradient boosting model predicts the spending players' lifetime value to be on average $5.55 smaller than in reality with a seven-day observation period. For non-spending players with the same observation period, gradient boosting predicts on average $0.09 too much spending. These metrics do not serve model comparison so well as MAE and RMSE, but they are valuable for practitioners to see whether models are rather too optimistic or pessimistic for both players with purchases and without purchases.

*Table 8: Mean prediction error by model, player spending and observation period (USD)*

| Observation period | | 1 day | | | 7 days | | | 14 days | | |
|---|---|---|---|---|---|---|---|---|---|---|
| **Model** | **Mean error (USD)** | All | Spend | No spend | All | Spend | No spend | All | Spend | No spend |
| Past mean purchase | | N/A | N/A | N/A | +0.14 | +39.05 | 0.00 | +0.04 | 12.43 | 0.00 |
| CBG/NBD | | N/A | N/A | N/A | +0.19 | +3.46 | +0.18 | +0.09 | +0.65 | +0.09 |
| Pareto/GGG | | N/A | N/A | N/A | +0.14 | +1.75 | +0.13 | +0.08 | +0.33 | +0.08 |
| Linear regression | | +0.03 | -4.11 | +0.05 | +0.01 | -5.52 | +0.03 | -0.04 | -14.22 | +0.01 |
| Gradient boosting | | +0.06 | -1.78 | +0.07 | +0.07 | -5.55 | +0.09 | -0.04 | -13.85 | +0.01 |
| Random forest | | +0.04 | -4.00 | +0.05 | +0.01 | -6.86 | +0.04 | -0.04 | -13.18 | +0.01 |
| Deep-MLP | | +0.04 | -13.82 | +0.09 | +0.05 | -10.92 | +0.09 | -0.06 | -21.39 | +0.02 |

Figure 11 builds on top of Table 8 and illustrates aggregated prediction errors. For the test data, the total USD value of all player-level predictions is summed and compared to the actual sum of the players' CLV in test data. Two models stand out, namely random forest and linear regression, which predict the total sum of all players to be 35% and 37% higher than in reality. The rest of the models predict the total sum to be around two or three times higher than it actually is.

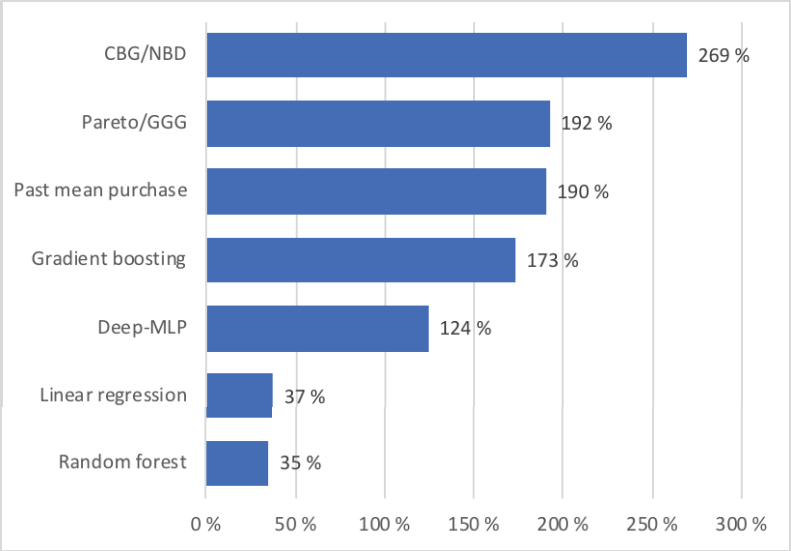


*Figure 11. Total prediction error (USD) by model, 7-day period.*

The relative feature importances for gradient boosting and random forest regressors are presented in Figure 12 for the seven-day observation period. The first variable in gradient boosting is the customer's lifetime value (cumulative) during observation period. It is followed by four variables of usage and gains of different in-game currencies and another four variables describing usage of different game modes and overall progress. For

random forest, the most important variables are the lifetime value and number of purchases (not monetary value). Hence, a larger number of variables have predictive power in the gradient boosting model. Please note that the scale is slightly different in these two graphs.

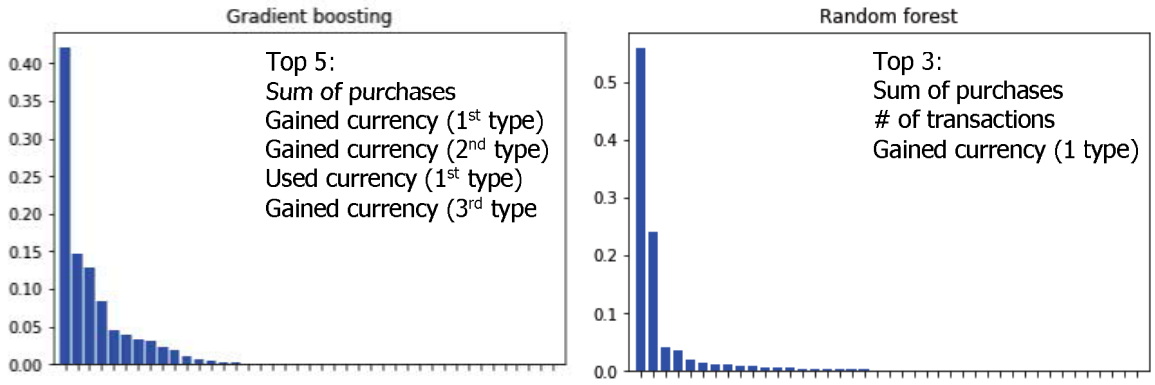


*Figure 12. Relative feature importances in gradient boosting and random forest regressors, 7-day period.*

# 6 Discussion

## 6.1 Key findings

### 6.1.1 Churn prediction

Analyzing classification metrics from Table 6, the results are quite promising. The scores are on par with results from previous literature, where game-specific best AUC scores range from 0.73 (Kim et al., 2017) to over 0.9 (e.g. Runge et al., 2014). The reason why the AUC scores of this analysis lie within the lower end of the range has most likely to do with the dataset. The dataset is smaller than its counterparts in earlier literature. Some of the earlier works also did not include non-paying players, and all but one (Kim et al., 2017) did not use a new cohort of registered players as was done in this analysis, but players who were apparently established ones. Most players in this dataset churned quickly. But all in all, the results are satisfying compared with earlier literature.

The F-score per se is hard to interpret, because it is a combination of two further metrics, precision and recall. However, on its own the F-score can be used to compare prediction results with each other. By splitting the F-score to precision and recall, it is possible to draw further conclusions. The overall very high recall scores indicate that the models are sensitive to identify churners. The lower precision scores, on the other hand, indicate that the high recall scores are achieved at the expense of classifying almost all non-churning players as churners. With precision scores of around 0.86, 14% of the positive predictions are false positive. As 15% of all players in the dataset stay alive, it means that nearly all of them are falsely classified as churners. So, how come are the AUC scores indicating a better performance than precision scores are? The answer lies within technical implementation. The models in Python's *sklearn* package calculate precision, recall and F-scores with a default probability threshold of 0.5. As explained in Chapter 3.3, ROC curves and AUC scores are derived from varying the threshold from zero to one. Hence, the threshold should be modified in real use cases to separate the classes more effectively from each other.

Only the hiatus heuristic with a seven-day observation period stands out with a low recall score. This happens because the heuristic is optimistic about players who continue playing after their first day, and therefore the recall score also increases with a longer observation period when the heuristic "notices" a longer idle time for players that had their last session on day 3 or 4, for instance.

AUC is an easier metric to interpret, because a value of 0.5 means randomly guessing the outcome (equivalent to flipping a coin). All the models outperform random guessing, and all of them, except conditional inference survival ensembles, outperform the hiatus heuristic, which was used as the benchmark. Overall, the performance does not vary dramatically between models and there is no single model that stands out from the pack. This also indicates that increasing model complexity, from logistic regression all the way to deep multilayer perceptron, does not provide significant advantage.

The failure of conditional inference survival ensembles certainly lies with different implementation from earlier literature. The initial reason of including the model in the analysis was the inspiration of the work by Periáñez et al. (2016), where conditional inference survival ensembles are compared with binary classifiers. The article does not explain the technical implementation, so my technique must be different from theirs.

SMOTE balancing did not improve prediction performance, and the reason might be player behavior. Unlike customers/users in some other noncontractual settings, mobile game players lose interest and churn quickly. So, a pattern of active playing might stop drastically, and there are no signs of growing intervals between sessions, for instance, which would indicate slowly losing interest. This means that synthetic samples are hard to build correctly, as the behavior does not greatly differ between churners and non-churners during observation period.

Another factor worth discarding is the 14-day observation period. The AUC scores for each model improve only slightly by including another week of data in observation. Predictions are more valuable the earlier they can be made and therefore the slight improvement does not justify doubling the length of the observation period. I suspect this lack of improvement to be linked to the nature of the game as well. As with many mobile freemium games, most players churn very quickly within days or even the first day. Hence, most churning patterns are captured by the shorter observation period, and the change during the second week is not significant. Based on AUC score, a one-day observation period yields unsatisfying results. With AUC scores around 0.63, the models do not predict so much better than a random guesser, which would be equivalent to an AUC score of 0.5.

The relative feature importances presented in Figure 10 showcase that most predictor variables are redundant for the prediction. What is interesting, however, is that the number of sessions and the number of games played overwhelm the other variables.

So, which model would be most suitable, given that their measured performance is so similar? First, the machine learning models are computationally lighter to run than

BTYD models. Second, within the machine learning models the deep multilayer perceptron does not shed light on feature importance and therefore does not provide insight into the factors affecting the prediction outcome in the first place. This is true to all deep neural networks, which are "black boxes" when implemented. Third, the random forest classifier utilizes more predictor variables than gradient boosting in this particular prediction task. This is visible in the wider distribution of relative feature importances. Lastly, even though logistic regression is also a model that is easy to implement, random forest yields slightly better results across the board. Thus, this particular analysis points towards using logistic regression and random forest classifiers.

### 6.1.2 CLV prediction

As in churn prediction, there is no clear choice of model. In terms of MAE and RMSE, linear regression and random forest are the best performing models and they yield surprisingly similar results, which is visible in Table 7. Some minor differences between the two models are observable in Table 8 where the mean prediction errors are presented. For the one-day observation period, random forest's prediction error for spending players is a little better. For the seven-day observation period on the other hand, linear regression's prediction error for spending players is clearly better with over a dollar's difference.

Overall, machine learning models outperform BTYD models and the heuristic, but the difference gets smaller with a longer observation period. As is visible in Table 8, machine learning models make rather pessimistic predictions, because the average prediction errors for spending players are smaller than actual values. BTYD models and the heuristic, on the other hand, are rather optimistic and predict larger spending than in reality. Figure 11 shows that all models are too optimistic on the aggregated level, and even the best performing model predicts that the total sum will be a third higher than in reality. Even though the spending players' lifetime value is underrated, the errors of non-spending players cancel this out on the aggregate level.

Another interesting phenomenon is that the machine learning models perform significantly better with a seven-day observation period compared to 14 days, which is contrary to the BTYD models and also counterintuitive. I believe this is due to the imbalanced dataset, where most players churn and stop spending quickly. Since most churns occur during the first week, the models are too optimistic and interpret the players who have made it to their second week to stay alive further than in reality.

The relative feature importances retrieved from gradient boosting and random forests in Figure 12 tell a similar tale as the classifiers, namely that the weights are heavily concentrated. In the regression task, the variables with most predictive power are logically related to spend, i.e. cumulative purchase value in the observation period and number of transactions.

But why do linear regression and random forest perform better than the rest? I believe that the strength of the simplest model, linear regression, can be explained by the heavy concentration of relative feature importances. With only a few predictor variables with actual predictive power, a simple model is enough. How about random forest versus the other two machine learning models? For the deep multilayer perceptron, it is hard to say, but the difference with gradient boosting may lie in high variance in the dataset. When data is noisy, boosting tends to overfit training data more easily. Trees are fitted to the training data until they reach their minimum value, but the ensemble does not generalize so well to new data. Random forests, on the other hand, train each tree independently and average the result in the end. This averaging reduces outcome variance and therefore risk of overfitting. The used dataset suffers from high variance because of the imbalance of spenders versus non-spenders as well as churners versus non-churners. The BTYD models seem to suffer from a short observation period, because they perform slightly better with a 14-day observation period. Since the gamma-gamma submodel used for extending BTYD models to regression models utilizes average spend for all predicted transactions, the models are too optimistic in churn prediction. Due to the large share of churn happening quickly, the models seem to have a blind spot for churn later on within the 30 days period.

## 6.2 Managerial implications

For the case company and other practitioners alike, the study clearly suggests that simple machine learning models and random forests work for both churn prediction and CLV prediction. In churn prediction, logistic regression and random forest perform practically equally well. In CLV prediction, linear regression and random forest also perform equally well. It is therefore suggested that two models are tried out in practice and the better performing model is chosen. As the simpler models are computationally lighter to run, logistic and linear regression models are advised to be used if computational capacity is a relevant concern. As a third alternative, deep multilayer perceptron can also be tested if wanted, because they yield relatively good results as well. However, they and other neural

networks are "black boxes" when implemented and more conventional machine learning models (including random forests) showcase feature importance.

The analysis also clearly shows that predictions can be done one week after player registration, and a longer observation period than that does not improve results. The effect can even be just the opposite. The one-day observation period also performs poorly. In churn prediction, the AUC scores indicate that predictions after the first day do not perform much better than a random guesser (random guesser's 0.5 vs. models' scores around 0.63). As described in Chapter 6.1.1, scores for recall are high, but it only illustrates that the models with default threshold tend to classify nearly all players as churners. Therefore, it is essential that the probability threshold is modified to improve classification performance with a seven-day observation period as well.

This information enables implementation of preventive measures after one week with fair confidence. Potentially, these churners could be incentivized to keep playing with e.g. push notifications, some in-game gifts or faster progress in the game. Because these measures are game-specific and may affect future likelihood to make purchases, commenting these measures on a detailed level is out of scope of this study. The exact probability threshold of the classification models to be chosen depends on these preventive measures. The false positive rate increases when the true positive rate increases (although not as much), and the false positive players are targeted as churners. These non-churning players would receive the same benefits of churn prevention as the players who are actually planning to stop playing. Whether targeting false positives causes any harm or not, depends on both the individual player and the general context. For instance, push notifications might annoy some players, and progressing too fast in the game could mean that the player stops playing the game earlier than otherwise.

Information about relative feature importances plays an important role for practitioners. By knowing what features of the game (if any) drive churn and CLV, the game developers can focus on improving these features. In the case of the game of this analysis, number of sessions and purchases made during the observation period have by far most predictive performance, indicating that players' different ways of playing the game do not affect the likelihood of churn and making purchases.

Predictions on customer lifetime value help segment players based on their worth to the company and, if accurate enough, the predictions can estimate future cash flows when the player-level predictions are aggregated to total sums of money. In this analysis, the total predicted CLV of all players combined was at best 35% higher than in reality, which

is visible in Figure 11. Hence, I do not recommend using it as a way to predict revenue, but the order of magnitude will probably be right and can therefore give some early indications on the future. On an individual player level, the prediction errors can be large as Table 8 illustrates. When the vast majority of spending players spend less than 10 dollars during the first 30 days, mean prediction errors of around 5 dollars are hard to accept.

However, the predictions are accurate enough to enable customer segmentation. Figure 13 illustrates an entirely made-up example how a segmentation with the used dataset could be performed. In this example, the observation period is seven days, only predictions from actual spending players are included and the actual counts of each category are hidden to maintain confidentiality. This example illustrates that the two best performing models can predict the category correct. The minor tendency to predict players to lower segments tells the same tale as Table 8 and negative mean prediction errors for spending players.

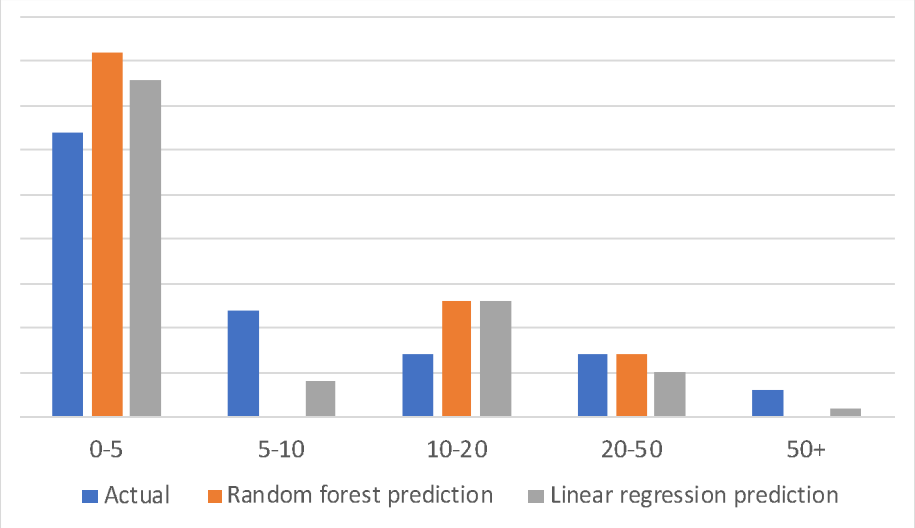


*Figure 13. Illustration of a possible customer segmentation by player CLV (USD)*

This kind of segmentation would help practitioners to cater to their most important players. By knowing which players will spend a little and which a lot, these groups could be kept happy e.g. by additional in-game content and gifts, and these players are probably more prone to make use of offers on purchases, which again could further increase revenues from these spending players.

## 6.3 Limitations and future research

The limitations of the study relate to data and restricted computational capacity. As stated earlier, the dataset included data from 63255 unique players, out of which less than one percent made any in-app purchases. A larger dataset could naturally give more accurate results and it would be interesting to test how the increase in number of observations affects model performance. How many observations are needed before performance does not improve anymore? Especially a larger number of observations from spending players would improve the regression task of predicting CLV. In addition, data from another game would shed light on cross-game model performance, which would reduce uncertainty of game-specific features affecting performance.

Computational capacity limits model optimization. For instance, when the time taken for hyperparameter tuning by grid search cross-validation takes several hours or even days on a regular laptop, it is unrealistic to hone the models close to perfection in a Master's thesis. The gain from such optimization would most likely not affect the key findings of the study, though.

The last clear limitation was the failed implementation of the conditional inference survival ensembles due to lack of clear reporting of its technical implementation in earlier literature. Investigating the implementation further would potentially lead to better predictions and relevant outcomes for the analysis.

In addition to these limitations regarding current research questions, I see that the research can be extended to multiple directions. The effect of spending patterns to lifetime value prediction performance would be interesting. The data could be retrieved from another game or potentially from the same game using a different nationality of players, since this study was done with U.S. players only.

Since the deep multilayer perceptron performed well in both classification and regression tasks, a deeper dive into their most suitable structure would be useful for practitioners. Of course, the final structure (layers and number of nodes) differs from setting to setting, but I suppose that the different options for compiling neural networks make a wider difference in model performance. These include at least activation function, optimizer, kernel initializer, validation split and early stopping.

The synthetic minority over-sampling technique (SMOTE) to augment data for imbalanced datasets was not included in the regression task of CLV prediction, because the reviewed literature does not use it, with one exception (Sifa et al., 2018). The whole idea

of SMOTE is to balance the data by nominal/factorial variables, even binary variables in its purest form. Balancing by a continuous variable, such as lifetime value, is a rather new field, and apparently the first article that modified the SMOTE algorithm to work for regression was written only seven years ago (Torgo et al., 2013). Investigating its performance in noncontractual and continuous settings, including mobile games, would be highly beneficial.

## 6.4 Conclusion

This thesis investigates what the current academic state-of-the-art models are for churn and customer lifetime value predictions in freemium mobile gaming and tests the suggested models in a real business setting. Both machine learning models and probabilistic models were researched and adjacent non-contractual and continuous fields outside mobile gaming are in scope of the investigation as well. The analysis is divided into two parts. Churn prediction is a binary classification task, where each prediction is a binary true/false categorization. CLV prediction, on the other hand, is a regression task where a continuous value is predicted.

In mobile games, the ensemble methods of gradient boosting and random forests have been popular and effective in both churn and CLV prediction, and CLV prediction has seen its first deep multilayer perceptrons (aka deep neural networks) with promising results. In similar noncontractual business settings, random forests were also popular. Probabilistic 'Buy-till-you-die' models have been popular in marketing science and e-commerce, and multiple different models are found in literature. For both prediction tasks, two probabilistic models were chosen, one following the footsteps of the conventional Pareto/NBD model and one with a fresher approach. A heuristic for both prediction tasks was used as a benchmark.

The achieved prediction performance is on par with earlier literature. There is no clear winner for churn prediction, but logistic regression and random forest classifier are recommended due to their easy and transparent implementation. In CLV prediction, linear regression and random forest regressor performed best with a clear difference to other models. In both prediction tasks, an observation period of one week is enough to make predictions for the first month of a player's lifetime.

# References

Abe, M. (2009) “Counting Your Customers” One by One: A Hierarchical Bayes Extension to the Pareto/NBD Model, *Marketing Science*, 28(3), pp. 541–553.

App Annie (2019) *The State of Mobile 2019*. Retrieved from https://www.appannie.com/en/go/state-of-mobile-2019/

Batislam, E. P., Denizel, M. & Filiztekin, A. (2007) Empirical validation and comparison of models for customer base analysis. *International Journal of Research in Marketing*, 24(3), pp. 201–209.

Bemmaor, A. C. & Glady, N. (2012) Modeling purchasing behavior with sudden "Death": A flexible customer lifetime model, *Management Science*, 58(5), pp. 1012–1021.

Bertens, P., Guitart, A. & Periáñez, Á. (2017) Games and Big Data: A Scalable Multi-Dimensional Churn Prediction Model, In *2017 IEEE Conference on Computational Intelligence and Games (CIG)*, IEEE, New York, USA, pp. 33–36.

Breiman, L. (1996) Bagging predictors. *Machine learning*, 24(2), pp. 123–140.

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

Burez, J. & Van den Poel, D. (2009) Handling class imbalance in customer churn prediction, *Expert Systems with Applications*, 36(3), pp. 4626–4636.

Chamberlain, B. P., Cardoso, A., Liu, C. H., Pagliari, R. & Deisenroth, M. P. (2017) Customer lifetime value prediction using embeddings. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 1753–1762, ACM.

Chen, P. P., Guitart, A., del Río, A. F. & Periáñez, Á. (2018) Customer Lifetime Value in Video Games Using Deep Learning and Parametric Models. In *2018 IEEE International Conference on Big Data*, pp. 2134–2140, IEEE.

Colombo, R. & Jiang, W. (1999). A stochastic RFM model, *Journal of Interactive Marketing*, 13(3), pp. 2–12.

Coussement, K. & De Bock, K. W. (2013) Customer churn prediction in the online gambling industry: The beneficial effect of ensemble learning, *Journal of Business Research*, 66(9), pp. 1629–1636.

Cramer, J. S. (2002) The origins of logistic regression, *Tinbergen Institute Working Paper*, 2002–119(4), pp. 167–178.

Drachen, A., Kung, Y., Klabjan, D. & Runge, J. (2016) Rapid Prediction of Player Retention in Free-to-Play Mobile Games, *arXiv.org*.

Drachen, A., Pastor, M., Liu, A., Fontaine, D. J., Chang, Y., Runge, J., Sifa, R. & Klabjan, D. (2018) To be or not to be... social: Incorporating simple social features in mobile game customer lifetime value predictions. In *Proceedings of the Australasian Computer Science Week Multiconference*, ACM.

Dror, G., Pelleg, D., Rokhlenko, O. & Szpektor, I. (2012) Churn prediction in new users of Yahoo! answers. In *Proceedings of the 21st International Conference on World Wide Web*, pp. 829–834, ACM.

Fader, P. S. & Hardie, B. G. S. (2009) Probability models for customer-base analysis, *Journal of interactive marketing*, 23(1), pp. 61–69.

Fader, P. S. & Hardie, B. G. S. (2013) The Gamma-Gamma model of monetary value.

Fader, P. S., Hardie, B. G. S. & Lee, K. L. (2005a) "Counting Your Customers" the Easy Way: An Alternative to the Pareto/NBD Model, *Marketing Science*, 24(2), pp. 275–284.

Fader, P., Hardie, B. & Lee, K. (2005b). RFM and CLV: Using iso-value curves for customer base analysis, *Journal of Marketing Research*, 42(4), pp. 415–430.

Freund, Y. & Schapire, R. E. (1996) Experiments with a new boosting algorithm. In *icml*, 96, pp. 148–156.

Friedman, J. H. (2001) Greedy Function Approximation: A Gradient Boosting Machine. *The Annals of Statistics*, 29(5), pp. 1189–1232.

Friedman, J. H. (2002) Stochastic gradient boosting. *Computational Statistics and Data Analysis*, 38(4), pp. 367–378.

Gupta, S. & Lehmann, D. R. (2003) Customers as assets, *Journal of Interactive marketing*, 17(1), pp. 9–24.

Hadiji, F., Sifa, R., Drachen, A., Thurau, C., Kersting, K. & Bauckhage, C. (2014) Predicting Player Churn in the Wild, In *2014 IEEE Conference on Computational Intelligence and Games*. IEEE.

Hoppe, D. & Wagner, U. (2007) Customer Base Analysis: The Case for a Central Variant of the Betageometric/NBD Model, *Marketing – Journal of Research and Management*, 29(2), pp. 75–90.

Hothorn, T., Hornik, K. & Zeileis, A. (2006) Unbiased Recursive Partitioning: A Conditional Inference Framework. *Journal of Computational and Graphical Statistics*, 15(3), pp. 651–674.

Jerath, K., Fader, P. S. & Hardie, B. G. S. (2011) New Perspectives on Customer "Death" Using a Generalization of the Pareto/NBD Model, *Management Science*, 30(5), pp. 866–880.

Kim, S., Choi, D., Lee, E. & Rhee, W. (2017) Churn prediction of mobile and online casual games using play log data, *PLoS ONE*, 12(7), e0180735.

Lee, E., Jang, Y., Yoon, D., Jeon, J., Yang, S., Lee, S., ... Kim, K. (2019). Game Data Mining Competition on Churn Prediction and Survival Analysis Using Commercial Game Log Data, *IEEE Transactions on Games*, 11(3), pp. 215–226.

Lemmens, A. & Croux, C. (2006) Bagging and boosting classification trees to predict churn. *Journal of Marketing Research*, 43(2), pp. 276–286.

Ma, S. H. & Liu, J. L. (2007) The MCMC approach for solving the Pareto/NBD model and possible extensions. In *Third international conference on natural computation (ICNC 2007)*, 2, pp. 505–512. IEEE.

Maldonado, M., Dean, J., Czika, W. & Haller, S. (2014) Leveraging ensemble models in sas® enterprise miner™. In *Proceedings of the SAS Global Forum 2014 Conference*.

Milošević, M., Živić, N. & Andjelković, I. (2017) Early churn prediction with personalized targeting in mobile social games, *Expert Systems With Applications*, 83, pp. 326–332.

Peng, C. Y. J., Lee, K. L. & Ingersoll, G. M. (2002) An introduction to logistic regression analysis and reporting. *The Journal of Educational Research*, 96(1), pp. 3–14.

Periáñez, Á., Saas, A., Guitart, A. & Magne, C. (2016) Churn Prediction in Mobile Social Games: Towards a Complete Assessment Using Survival Ensembles, *2016 IEEE International Conference on Data Science and Advanced Analytics*, pp. 564–573.

Persson, A. & Ryals, L. (2014) Making customer relationship decisions: Analytics v rules of thumb, *Journal of Business Research*, 67(8), pp. 1725–1732.

Pfeifer, P. E., Haskins, M.E. & Conroy, R. M. (2005) Customer lifetime value, customer profitability, and the treatment of acquisition spending, *Journal of Managerial Issues*, 17(1), pp. 11–25.

Platzer, M. (2008) Stochastic models of noncontractual consumer relationships, Master's thesis, Vienna University of Economics and Business Adminstration, Austria.

Platzer, M. & Reutterer, T. (2016) Ticking Away the Moments: Timing Regularity Helps to Better Predict Customer Activity, *Marketing Science*, 35(5), pp. 779–799.

Runge, J., Gao, P., Garcin, F. & Faltings, B. (2014) Churn prediction for high-value players in casual social games. In *2014 IEEE conference on Computational Intelligence and Games*. IEEE.

Schmidhuber, J. (2015) Deep learning in neural networks: An overview. *Neural Networks*, 61, pp. 85–117.

Schmittlein, D. C., Morrison, D. G. & Colombo, R. (1987) Counting Your Customers: Who Are They And What Will They Do Next? *Management Science*, 33(1), pp. 1–24.

Schmittlein, D. C. & Peterson, R. A. (1994) Customer Base Analysis: An Industrial Purchase Process Application, *Marketing Science*, 13(1), pp. 41–67.

Sifa, R., Hadiji, F., Runge, J., Drachen, A., Kersting, K. & Bauckhage, C. (2015) Predicting purchase decisions in mobile free-to-play games, In *Eleventh Artificial Intelligence and Interactive Digital Entertainment Conference*.

Sifa, R., Runge, J., Bauckhage, C. & Klapper, D. (2018) Customer lifetime value prediction in non-contractual freemium settings: Chasing high-value users using deep neural networks and SMOTE. In *Proceedings of the 51st Hawaii International Conference on System Sciences*.

Sokolova, M., Japkowicz, N. & Szpakowicz, S. (2006) Beyond accuracy, F-score and ROC: a family of discriminant measures for performance evaluation. In *Australasian joint conference on artificial intelligence*, pp. 1015–1021. Springer, Berlin, Heidelberg.

Torgo, L., Ribeiro, R. P., Pfahringer, B. & Branco, P. (2013) Smote for regression. In *Portuguese conference on artificial intelligence*, pp. 378–389, Springer, Berlin, Heidelberg.

Vanderveld, A., Pandey, A., Han, A. & Parekh, R. (2016) An engagement-based customer lifetime value system for e-commerce, In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 293–302, ACM.

Wang, Z. & Bovik, A. (2009) Mean squared error: Love it or leave it? A new look at Signal Fidelity Measures, *IEEE Signal Processing Magazine*, 26(1), pp. 98–117.

Wang, X., Liu, T. & Miao, J. (2019) A Deep Probabilistic Model for Customer Lifetime Value Prediction, *arXiv.org*.

Wangperawong, A., Brun, C., Laudy, O. & Pavasuthipaisit, R. (2016) Churn analysis using deep convolutional neural networks and autoencoders, *arXiv.org*.

Wübben, M. & von Wangenheim, F. (2008). Instant customer base analysis: Managerial heuristics often "get it right", *Journal of Marketing*, 72(3), pp. 82–93.

Yan, X., & Su, X. (2009) *Linear regression analysis: theory and computing*. World Scientific.