Predicting player churn using game-design-independent features across casual free-to-play games

Master thesis for Data Science: Business and Governance

Academic year 2016–2017

Name: Bastiaan van der Palen

ANR: 818913

Supervisor Tilburg University: Sander Bakkes

Supervisor Wooga: Julian Runge

Second reader: Pieter Spronck

Date: 3 July, 2017

Faculty: Tilburg School of Humanities

Preface

This thesis finalizes my education for the MSc specialization Data Science: Business and Governance at Tilburg University. With this preface I would like to thank the people involved during this project and Wooga for providing the datasets.

During the Data Science in Action thesis fair, the possibility of game analytics as a thesis topic immediately drew my attention. After consulting with Dr. Pieter Spronck, I was introduced to Dr. Anders Drachen who helped me by formalizing my research topic and introduced me to other researchers in the field of game analytics. Rafet Sifa, a PhD candidate focussing on in-game user behaviour using data science models, helped me during Skype calls with suggestions and input. Julian Runge, a former data scientist at Wooga who is also pursuing a PhD in predictive analytics, helped me to obtain the datasets from Wooga, when the plan of the initial datasets did not go through. This group of scientists helped me throughout this project. It would not have been possible to complete this thesis without these regular Skype meetings and feedback moments.

Finally, I would like to thank my thesis supervisor Dr. Sander Bakkes who supported me throughout the project. The quick e-mail responses, planned meetings and the mental support were essential for this thesis.

Bastiaan van der Palen
Eindhoven, June 2017

Summary

With the growth of the mobile casual game market, game developers have become more reliant on game analytics to stay ahead of competition. Predicting in-game churn behaviour has become essential for game developers to incentivize players who are predicted to leave the game.

While predicting churn has been the topic of several research studies, less research has been conducted on churn prediction across games. Developing a churn-predicting model that can be used across games is especially interesting for game developers who have recently launched a new game that has not yet generated enough data itself to train a prediction model. This thesis addresses this knowledge gap by predicting player churn across multiple free-to-play casual games using game-design-independent features and evaluating the model on out-of-sample data. The datasets provided by Wooga for this thesis contained player telemetry for three popular casual mobile games with different game designs.

The experiments conducted tested the performance differences between various models, namely, k-nearest neighbours, decision tree, random forest and logistic regression. The results indicate that random forest is overall the best classifier for predicting churn across games; however, the difference between its predication capability and that of the other classifiers was not substantial. Subsequently, the random forest model indicated *current absence time* is the most important feature, which is in accordance with the literature. Finally, the experiments with data covering a single-day feature window resulted in substantially lower prediction accuracies with little improvement compared to the majority baseline.

From the results, we can conclude that we can predict player churn across casual games with decent accuracy using game-design-independent features and data covering the first seven days of play behaviour. Moreover, the model generalizes excellently to out-of-sample data, allowing game developers to initiate churn prediction for recently launched games.

Contents

1. Introduction

1.1 Casual gaming:

The global market for casual games is estimated to grow from 75.5 billion dollars in 2013 to 102.9 billion dollars in 2017 (Casual Games Association, 2014). Revenues generated from casual games played on smartphones (22%) and tablets (14%) have seen the biggest growth in recent years, with an estimated 1.82 billion mobile players; mobile game spending will exceed $35 billion in 2017. With this growth in revenue and in the number of games, the industry has become more saturated and game developers have started to compete more heavily. To stay ahead of competition and keep players engaged longer, game developers have started learning from player data by analysing it (El-Nasr, Drachen, & Canossa, 2013).

Using player data analysis, game developers have succeeded in developing games for a broad audience and have learned what keeps casual gamers playing. This, in combination with the rise of global social network site Facebook, provided the industry with a new game segment, social networking games (SNGs) (Lewis, Wardrip-Fruin, & Whitehead, 2012).

> **Definition 1:** Social networking games (SNGs) are distributed and played on social networks such as Facebook.

SNGs use the data infrastructure of the social networking site to implement social functionalities (Alsén, Runge, Drachen, & Klapper, 2016). SNGs are able to spread fast and gain a substantial player base through viral acquisition techniques, for instance by friend recommendations and alerts. Surprisingly, the gameplay is often not social in terms of player interaction within the game. The social aspect of these games is mostly contributed by friend recommendations (Alsén et al., 2016).

The popularity of casual games and the emergence of the social games has led to the introduction of a new term in the literature: casual social games (CSGs) (Alsén et al., 2016).

> **Definition 2:** Casual social games (CSGs) are part of the social games industry and they require the user to engage socially while gaming.

A game is considered a CSG when five predefined elements are present in the game design. These elements are high accessibility and engagement, inclusion of viral/social features, free-to-play, strong sociability and genre agnosticism (Alsén et al., 2016). As the success of SNGs and CSGs spread through the industry, more game developers started developing free-to-play casual games.

> **Definition 3:** Free-to-play (F2P) games, also referred to as *freemium games*, can be downloaded and played free of charge. The revenue generated by F2P games is comprised of advertisement sales and/or in-game purchases.

The ability to develop games for a platform with a large user base gave birth to new companies, focussing on social networking games. Some of the biggest developers are Zynga, King, Rovio and Wooga (Lunden, 2012). The success of *Candy Crush* for social game developer King led to millions of dollars in revenue, and even today it is one of the most played games on Facebook's gaming platform (GameHunters, 2017).

1.2 Player churn for F2P social games

With the increase of competitors in the casual game industry, game developers have begun to compete heavily to keep players loyal to their games. Predicting player churn, in the F2P casual game industry, has become of increasing importance to game developers.

> **Definition 4:** Player churn is a metric, calculated as a function of time, that measures whether a player stops playing the game and is therefore considered a churner.

Churn rate is considered a key performance indicator in the field of online games and is one of the most important metrics in the F2P business model (El-Nasr et al., 2013). Being able to detect when and where players lose their interest in the game is very important. Predicted churners can be targeted with in-game rewards in order to keep them playing. This is often achieved by predicting future churners and offering tailored incentives to keep players loyal.

Investing in churn predicting can be profitable for companies. The cost of attracting new customers is five times higher than the cost of retaining current customers. Furthermore, long-term customers tend to generate more income (Verbeke, Martens, Mues, & Baesens, 2011). Therefore, improving retention increases revenues. Decision makers, concerned with customer retention, are interested in identifying future churners to use incentivization to prevent players from churning. Churn-prediction models have a certain degree of classification accuracy. False positives – i.e. players incorrectly classified as future churners – will be targeted in the retention campaign along with the actual future churners. This misclassification is a risk that companies are willing to take because the cost of attracting new customers is up to five times higher than the cost of retaining customers (Verbeke et al., 2011). The false negatives – i.e., actual future churners not predicted as such by the model – have a more negative financial impact on the company.

1.3 Research questions

In the area of game analysis for F2P games, churn prediction is a key challenge for game developers in order to succeed (Hadiji et al., 2014). Although, a sufficient amount of research has already been conducted on game-specific datasets, less is known about predicting churn across games. In Hadiji et al. (2014), the researchers addressed this knowledge gap for the first time. They were able to achieve high prediction accuracy by training and testing a decision tree, with universal input features. The researchers address this new area of cross-game analytics and recommend that future research investigate how cross-game data can make churn predictions for a game that has been recently launched and has not yet generated enough data itself. These early predictions can help game developers make data-driven decisions in an early stage in game development. This study will address this recommendation by analysing cross-game telemetry data to predict churn behaviour across games. This is of interest for the industry, which can use the insights of this project to analyse player churn across games. This objective is addressed in the first research question.

Research question 1 (RQ1): To what extent can we predict player churn across multiple F2P games with game-independent input features?

To train the prediction models to answer RQ1, input features will be selected that contribute to the correct classification of the future churners. These input features need to be independent of game-design and easily generalizable to similar games from other developers. Research question 2 aims to address this issue and give insight into the most important features.

Research question 2 (RQ2): What are the most important input features for predicting player churn across F2P games?

Apart from the input features, the churn time window significantly influences the results. An observation time window, which captures the game activities to check if the player churned, should accurately measure real churning behaviour. A short observation window could lead to misclassifying non-churners as churners and a long observation window could miss actual churners that open the game after a long period of inactivity. A feature window captures the game activities of players within a certain time-period. For this project we want to use different feature time- windows and compare the results. The last research question addresses this issue and aims to investigate the relation of different time windows per game on the churn-prediction performance.

Research question 3 (RQ3): What is the effect of different time windows on churn prediction across F2P games?

1.4 Structure

This thesis will be outlined according to the following structure. In Chapter 2 we will delve deeper into the literature on churn prediction and game-specific churn prediction. Chapter 3, the experimental set-up chapter, describes the dataset and the experiments in detail. Subsequently, Chapter 4 will present the results of these experiments. In Chapter 5, we discuss these results and in Chapter 6 we answer the research questions and present our recommendations for future research. The references used and the appendixes can be found on the last pages.

2. Related work

This chapter discusses the relevant literature in this field which will provide the foundation of this thesis. First, research into customer retention is explored in section 2.1. Second, in section 2.2 retention in the game industry is discussed. Subsequently, the literature on churn prediction is presented in section 2.3, followed by the literature on game-specific retention in section 2.4. This chapter is completed with a discussion of the literature on machine-learning algorithms as well as feature selection and time window selection.

2.1 Customer retention

In customer relationship management (CRM), the most important step after customer acquisition is customer retention (Kumar & Petersen, 2012). As discussed in section 1.2, retaining a customer is up to five times less costly than acquiring a new customer (Verbeke et al., 2011). Retaining the customer is dependent on several factors. How these factors relate to each other is illustrated in Figure 1. The dashed line in the lower right corner illustrates customer retention. It can be observed that customer satisfaction and product and service quality are the main influencing factors for customer retention.

In the literature, we can distinguish two approaches to investigate customer retention. One approach focuses on the effects of marketing efforts on customer retention and the second on researching statistical models that aim to predict customer retention (Kumar & Petersen, 2012). The main question in customer retention is "Will a customer stay or leave a company or service?" This question is often modelled with a logistic regression and as a binary classification problem. Retained customers are labelled with 1 and churners with 0.

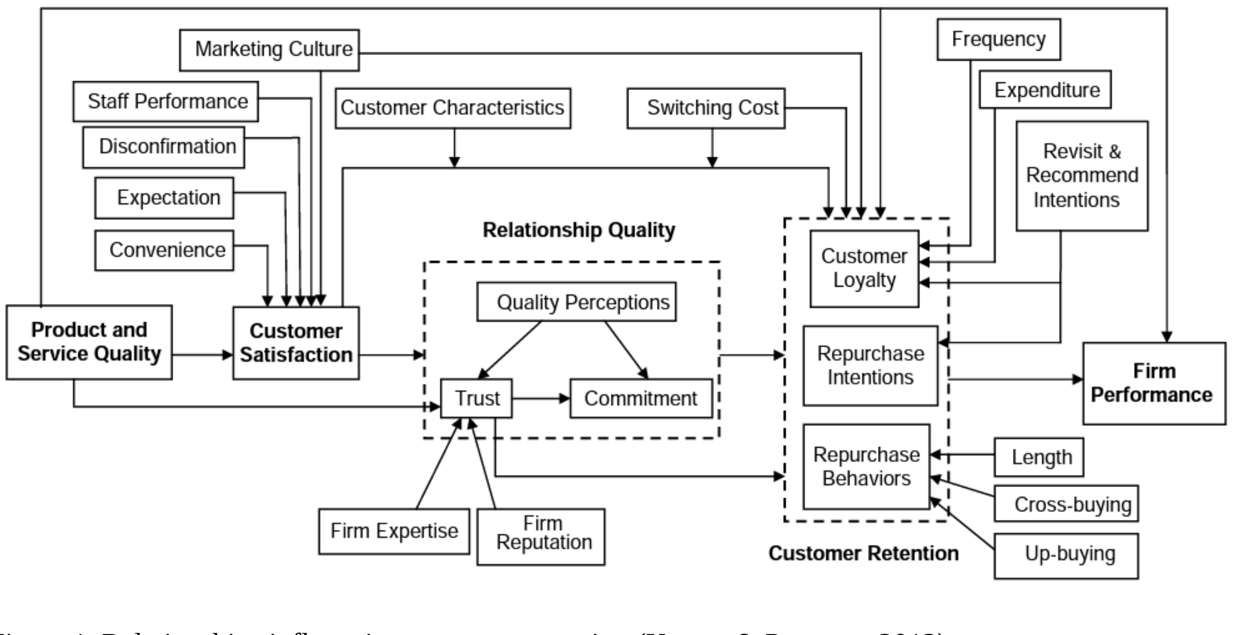

Figure 1. Relationships influencing customer retention (Kumar & Petersen, 2012).

2.2 Game-specific retention

When a game is being played, data is generated and sent to a collection server. These telemetry data are being analysed to obtain, for instance, insight into gaming patterns and to find bugs and drop out points. Metrics measuring revenue, gaming behaviour and game performance are important for determining the attention points and the direction of a company (El-Nasr et al., 2013).

In the industry, retention is measured using different methods. One method that is frequently used by developers in the industry is the DAU/MAU ratio (Hui, 2013). Dividing the Daily Active Users by the Monthly Active Users provides a broad idea of the number of players who are retained for a relatively long period. This ratio is also referred to as Retention Rate (El-Nasr et al., 2013). A retention rate between 0.2 and 0.3 for a relatively long period of time is considered necessary in order to be successful. During the last decade, this method of measuring player retention has been increasingly critiqued. The method does not measure retention on the individual level, but only on the entire player base. This measure does not provide insight into what proportion of players who played the game on day one actually returned later that month.

Why gamers, at some point during the game, decide to stop playing the game is of interest to game developers. A metric called XED (Exit Event Distribution) captures the last activity of the player before churning out of the game; this metric is increasingly being used by game developers (El-Nasr et al., 2013). It might be that a certain in-game event leads to churning behaviour. Analysing these so-called choke points will give game developers the opportunity to fix them. Using telemetry data of an online F2P strategy game, researchers were able to find these choke points by analysing player pathways (Gagné, Seif El-Nasr, & Shaw, 2012). Not only do choke points cause players to churn, all kinds of circumstances, not gathered within the datasets, lead to churning behaviour. Researchers were able to predict future churners based on data gathered in the first two levels of a game (Mahlman, Drachen, Canossa, Togelius, & Yannakakis, 2010).

Game developers started to share their data with computer scientists to predict why players lose interest in the game, and this resulted in several published papers. Researchers trained models that enable a game developer to predict how long a player remained interested in the game (Bauckhage et al., 2012). Subsequently, the researchers established that the distribution of player churn follows a power law distribution. This study, based on the data of five AAA games, showed that a majority of players play a game for a short period of time and a fraction of players maintain interest in the game for a considerably longer period. Players who are more engaged with the game, and thus are playing for a longer time period, are more willing to spend money on it.

2.3 Customer churn

While the widespread availability of data in the industry is put to use to increase the customer lifetime value, at some point, the customer will decide to leave the company and “churn”. Besides the retention

efforts of a company, attrition of customers is inevitable. Customer churn can be highly expensive for companies if they do not know when and why their customers churn; therefore companies can benefit financially by investing in churn prediction (Kumar & Petersen, 2012).
Customer churn, at the individual level, is the probability that the customer leaves the game in a given time period. At the company level, churn represents the percentage of customers who leave within a certain time period (Blattberg, Kim, & Neslin, 2008). Therefore, churn can be calculated as 1 – Retention Rate.

Blattberg et al. (2008) defines two types of customer churn, namely *voluntary* and *involuntary* churn. Involuntary churn occurs if the company decides to stop the relationship with the customer. When there is voluntary churn, the customer is either dissatisfied and considered a *deliberate* churner or the customer no longer needs the product and is an *incidental* churner. Predicting future churners is very interesting for companies, and with the revolution in data collection companies have started feeding customer data into models to predict churn rate with a certain level of accuracy. Because of the absence of data on what causes customers to churn directly, surrogate data is used which in itself does not measure churn. Computer models can find underlying patterns in this surrogate data to predict future churners.

Companies are interested in keeping the customer churn as low as possible. A high level of customer churn is especially threatening for companies that need to invest highly in acquisition in order to acquire a new customer. This is the case, for example, for telephone service providers. Researchers analysed data of telecom networks to determine if churning behaviour is related to social ties within the same network (Dasgupta et al., 2008). A social tie between two persons was found by analysing call frequency and call duration. The results of this study show that social ties influence churning behaviour. If someone in your network stops using the service, you are more likely to stop using the service. In the literature, a broad variety of papers have been published on predicting customer churn in the telecommunications. A study comparing prediction results for customer churn in telecommunication data found the highest accuracies for the SVM algorithm (Xia & Jin, 2008). Especially when the data is not linearly separable and the missing values in the dataset are not present excessively, the SVM is favourable above the other tested algorithms.

A meta-study, which compared the algorithms and results of multiple studies, concluded that the models need to be more comprehensive and justifiable (Verbeke et al., 2011). Two new techniques were introduced to tackle these shortcomings and compare the results with more classical data-mining approaches. While decision tree (C4.5) as a rule-based system is comprehensible, the comprehensibility decreases with each new rule. The justifiability of a model is related to the underlying rules the model makes. If domain knowledge dictates that the higher the difficulty of the game the higher the churn rate and your model violates this knowledge by predicting the opposite, the justifiably of your model decreases. The authors therefore suggest that the data mining model should have the possibility to impose constraints to safeguard the domain knowledge and the reliability of the model (Verbeke et al., 2011).

2.4 Churn prediction in the game industry

This approach of predicting player churn is a growing field in scientific research. There is, however, no predefined method of predicting customer churn. Researchers use different methods and data-mining algorithms to achieve the highest prediction accuracy scores on unseen data. Research into two casual F2P games showed the highest churn-prediction performance for a single hidden layer neural network (Runge, Gao, Garcin, & Faltings, 2014). Furthermore, the authors designed an A/B test to incentivize future churners by providing free in-game currency. This, however, did not yield lower churn rates. Where churn prediction often was associated with game specific features, the need for a game-independent churn prediction model surfaced in the industry.

A study on data containing five F2P games played on web-based and mobile platforms resulted in a game-independent churn prediction model with high prediction accuracy (Hadiji et al., 2014). This prediction model, however, predicts with mid-length observation and prediction data. A new contribution to the field of churn prediction for F2P games investigated the short-term prediction window as most players stop playing the game within a few days after installing the game (Drachen et al., 2016). The study also introduced a heuristic model with a prediction performance comparable to those of well-known traditional data-mining models. Additionally, the results indicated that prediction accuracies increase when the time window is increased.

Accurately predicting when a player becomes a churner does not necessarily mean that you get insight into what causes people to churn (Nozhnin, 2012). Researchers working on building models to predict player churn for the game Aion, for example, successfully tackled predicting player churn. However, why players stop playing was not discovered. Because this paper was not published, the results should be interpreted with caution; however, it gives an indication that accurately predicting churn does not necessarily mean one knows on what grounds an algorithm identifies that churn.

2.5 Applicable machine-learning algorithms

To answer the first research question – *To what extent can we predict player churn across multiple F2P games with game-independent input features?* – prediction algorithms used in similar studies are investigated. As discussed in section 1.2, player churn measures whether a player left the game and is calculated as a function of time. A player is either considered as a churner or a non-churner and therefore the prediction class is binary. A situation like this is best predicted using a classification algorithm instead of a regression model (Daumé, 2012). Another requirement that this classification model needs to address is the generalizability of the model, as this thesis aims to test the models using out-of-sample data. Finally, the logic behind the prediction models needs to be easily understandable for people without a background in machine learning. In this section we cover these requirements for these machine-learning algorithms by exploring the literature. How these models operate will be dealt with in the method chapter.

Before feeding data into a machine-learning algorithm, it is common practice to explore the data. Laursen (2011) strongly recommends using decision tree during this process. Within the early parts of data exploration and model development, decision tree can give insight into the decisions of a machine-learning algorithm. Additionally, decision tree help detect logical errors. In Drachen et al. (2016), a decision tree is used as a benchmark. The performance of other models is compared with this simple heuristic model. The authors also mention another benefit of using decision trees. The set of rules it generates can be easily understood by decision makers without a background in machine learning.

Throughout the literature on churn prediction for mobile games, different machine-learning models are used; however, one model was found in nearly all papers: logistic regression (Drachen et al., 2016; Hadiji et al., 2014; Runge et al., 2014). For all these papers, logistic regression is found to have one of the highest accuracies of all the models tested. In Drachen et al. (2016), logistic regression does especially well on data with a small feature window but the differences are minimal. Kumar and Petersen (2012) found logistic regression used in several domains of industry. Churning customers were predicted in the retail industry, telecommunication industry and financial services industry using logistic regression algorithm.

Support vector machine (SVM) is used by Runge et al. (2014) and Drachen et al. (2016). In Runge et al. (2014), the authors reflect on the performance of the SVM algorithm. Compared with the decision tree, the performance of the SVM is better for low false-positive rates. As discussed in section 1.2, a low rate of incorrectly classified churners is preferred and less harmful than incorrectly classifying real churners. Therefore, the authors prefer using the SVM to the decision tree.

In Hadiji et al. (2014), the neural networks model is used to predict player churn. In the literature, neural networks often yield high accuracies in classification tasks. In contrast to the results of Runge et al. (2014), the trained neural networks algorithm did not outperform the decision tree. These results might relate to the different datasets and input features. First, a dataset with telemetry data on five casual games instead of player data on one game is used. Subsequently, cross-game features instead of game-specific features are used. For this thesis, the data we use share more characteristics with the research done by Hadiji et al. (2014). Hence, neural networks is not likely to produce the best prediction accuracies for this thesis. Among the previously mentioned algorithms, the Naive Bayes algorithm (Hadiji et al., 2014) and the random forest algorithm (Drachen et al., 2016) were used in the literature with decent prediction results.

In conclusion, earlier research tested multiple machine-learning algorithms to address the classification task of predicting player churn. Logistic regression is used throughout different research industries. It is one of the most used algorithms for predicting player churn because it does especially well on binary classification problems. The decision tree is often used to get insight into the most important features. It is also put to use to check for logical errors. The rules it generates can easily be understood by decision makers without prior knowledge of machine learning. The SVM algorithm did well on predicting churn with low levels of misclassified churners. The neural networks algorithm performs above average on churn-prediction problems for datasets with player data on one game. As this thesis aims to build models based on data from multiple games, neural networks will be excluded from this research. The mentioned algorithms will be further explained in the classification algorithms section of the method chapter.

2.6 Input features

To answer RQ2 – *What are the most important features for predicting player churn across F2P games?* – we discuss the relevant literature on churn prediction for mobile games. The selection of input features is of the upmost importance for classification tasks. It is important that these features capture user behaviour across the game as well as user characteristics (Drachen et al., 2016).

2.6.1 Installation measures

In the literature, the installation metrics are almost always present as input features to the model. Some installation metrics are: device type, geographic location and whether or not the player was acquired by a marketing effort (Drachen et al., 2016).

2.6.2 Gameplay measures

The gameplay features used in the literature record players activities during playing sessions. Features such as total days, total sessions, total rounds, average session duration, average round duration, and total elapsed play time all contain information on how the game is being played (Drachen et al., 2016). These data, when analysed by the algorithms, reveal more information on what kind of player is playing the game. This information is highly useful for predicting future churners.

2.6.3 Intersession measures

Data on game activity is not the only data used for prediction purposes; data on non-activity can also be used as an input feature. In Hadiji et al. (2014), researchers used current absence time, which indicates the elapsed time since the last session of the user. Subsequently, Drachen et al. (2016) also used the current absence time and added average time between sessions to it as well. These two features were determined to be the strongest predictors of player churn. These features reveal how dedicated the gamer is to the game.

2.6.4 Round-specific measures

Much like the gameplay measures, the round-specific measures contain information on how the game is being played. This metric, however, contains information per round. This type of data is not captured by all mobile games, so this has to be taken into account when adding them into the model. The measures contain metrics such as average moves, average stars, and maximum level (Drachen et al., 2016). As our aim is to develop a cross-game churn-prediction model with game-design-independent features, these round-specific measures will not be used for this thesis.

2.7 Churn-prediction time windows

As discussed in the introduction, churn is a metric that is calculated as a function of time. To be able to answer the third research question – *What is the effect of different time windows on churn prediction across F2P games?* – it is important to investigate the literature on the time windows used for data collection and the evaluation period. The time window used to gather data about the player is called the feature window. The time window used to evaluate the predicted churn is called the observation window.

We can distinguish two types of churn-prediction models: single future period and time series. In single future period churn-prediction models, data is fed into the model for one or more time periods. When the churn period is reached, the customer is either observed as a churner or is not. For time series churn-prediction models, the data is gathered in combination with observing whether the customer has churned. The difference is that churn is continuously observed rather than a single point in the future. For the prediction models, you still use data from predefined periods to train the prediction model (Blattberg et al., 2008).

In the literature on F2P casual games, different time windows are used to measure player churn. Researchers, interested in predicting churn for high-value players found the highest prediction accuracies by using data gathered within 14 days before the churn event (Runge et al., 2014). In other literature, the researchers tested different feature windows and evaluation windows with the aim to predict churn on relatively short time periods. By comparing feature windows of one session, one day, 1-3 days and 1-7 days, the researchers found that data capturing one day of game activity can predict player behaviour up to one week into the future with acceptable accuracy (Drachen et al., 2016).

In Hadiji et al. (2014) the researchers distinguish two methods of churn data generation illustrated in Figure 2. In the classical approach one generates training examples by looking at all churned players and marking them as churners. Non-churners are selected from players who play more than one session and randomly pick a cut-off point in their recorded sessions. They are considered non-churners because the last session in the data that is selected is not the last session of the player. The second approach that the researchers tested better reflects real-world situations and therefore is more commonly used by game developers to predict future churners. For this approach, the data is generated by selecting sessions that fall in between a predefined range of time. Non-churners are labelled as such if they logged-in during the feature window and still played the game during the evaluation period. The churners are labelled as such if the players did not log-in during the evaluation period (Hadiji et al., 2014).

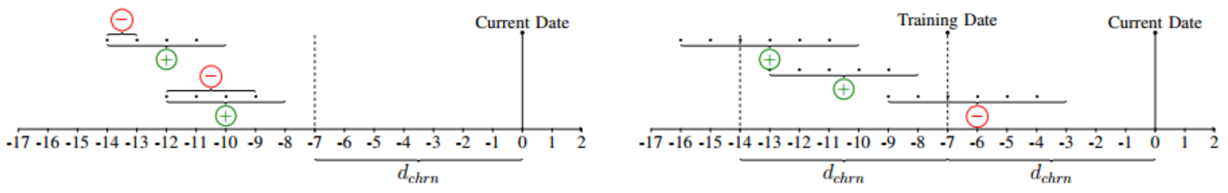

Figure 2: On the left the classical approach to churn data generation, on the right the real-world method. The + examples are players who churned, while the examples are the non-churners (Hadiji et al., 2014).

To conclude, there is no predefined method when it comes to deciding upon a time window to record churn behaviour. The choice depends on the aim of your research and the type of game. In the literature it is not uncommon to compare multiple time windows and reflect on the performance. This will be taken into account for this thesis.

2.8 Balancing classes

The literature on researching churn behaviour for F2P casual games takes notice of the imbalance of examples. All the published papers mention the over-representation of churners in the dataset. The over-representation of immediate churn after the first playing session of the game can lead to highly imbalanced datasets. Predicting player churn with imbalanced class labels could lead to misclassifying a minority class (Daumé, 2012). In Hadiji et al. (2014), this issue is addressed by defining a sliding-window to record churn behaviour. They use the feature current absence time and limit it to a particular number of days. The number is randomly chosen and represents the days before the test date. By implementing churn prediction this way, the model gets updated as new data come in. This is especially useful in a live system. For this thesis, it is something to investigate but because of the offline learning nature of this research, it is not necessary to implement.

Other techniques can be used to handle class imbalance, for instance sub-sampling. By sub-sampling the data, you omit part of the majority class until your classes are balanced. This is not preferable because if data is omitted, you have a less accurate representation of examples in your dataset (Daumé, 2012). Another method is to assign higher weights to the minority class. Unlike sub-sampling, weighing keeps the variance in the dataset and thus is preferred when handling class imbalance within the dataset.

2.9 Thesis contributions

2.9.1 Academic contribution

While a decent amount is known about predicting churn for individual F2P games, less is known about cross-game player churn. This thesis aims to target this knowledge gap by building prediction models that are applicable across games and only use input features that are game independent. To test the generalization performance of the prediction models, this thesis will use out-of-sample data as a test set. The prediction algorithms, time windows and features discussed in this section are the foundations upon which the experiments for this thesis are built.

2.9.2 Practical contribution

Predicting player churn in the game industry is every day's business. Retaining a player is up to five times less expensive than acquiring a new player. Thus, predicting future churners is very interesting for game developers in order to be successful. The ability to predict churn across games enables game developers to make predictions on newly launched games that have not yet generated enough data themselves. In the next chapters, the dataset and the experiments are described, followed by the results and the discussion of those results.

3. Method

This chapter describes the datasets, experiments and model evaluation of this thesis in detail. In section 3.1, the raw datasets provided by Wooga are discussed. The data pre-processing steps are presented in section 3.2 and the creation of the final datasets is described. In the data visualization section, the descriptive statistics of the final datasets are explored. Subsequently, in section 3.4 the experiments are described in detail. The classification algorithms used for these experiments are discussed in section 3.5. The software packages and programming languages are discussed in section 3.6. Finally, we conclude with the evaluation criteria for the classification algorithms.

3.1 Description of the datasets

In this section, we discuss the datasets and the metrics provided by Wooga for this research project.

Table 1. The number of players per game in the raw datasets

	IOS	**Android**
Pearl’s Peril	419,539	
Jelly Splash	634,085	968,326
Diamond Dash	538,443	
Total number of players	**1,592,067**	**968,326**

3.1.1 Raw datasets

The raw datasets provide by Wooga contained player data for about 2.6 million players. The datasets comprised data on three popular mobile casual games, namely, Jelly Splash, Diamond Dash and Pearl's Peril. The data of each game was represented in three datasets. The datasets contained data on installation, sessions and rounds per user. The numbers of players per dataset are displayed in Table 1. For all datasets, the data was recorded in a period of 274 observation days.

3.1.2 Metrics

The metrics of the raw datasets were divided over three datasets and belong to three categories of game telemetry data, namely installation measures, gameplay measures and round-specific measures (El-Nasr, 2013). The installation datasets cover data on each installation of the game on a device. The sessions datasets cover data on each time the game was opened on a device. The rounds contained data on the rounds a player played including the game-specific metrics. Table 2 displays all the metrics that were present in the raw dataset.

Table 2. Metrics of the raw datasets for Jelly Splash, Diamond Dash and Pearl's Peril

Dataset	**Number of metrics**	**Metrics**
Installation datasets	6	Device id, observation day, timestamp, country, device type and acquired by marketing effort.
Sessions datasets	8	Device id, Facebook id, observation day, timestamp, country, device type, operating system and language.
Diamond Dash rounds dataset	22	Device id, Facebook id, observation day, timestamp, country, device type, operating system, version, online, coin balance, gold balance, lives, xp, accuracy, crashed gems, final score, level number, time boost, bomb boost, colour boost, fireball and fire mode.
Jelly Splash rounds dataset	30	Device id, Facebook id, observation day, timestamp, country, device type, operating system, version, online, playing friends, coins, lives, stars, max level number, average snake length, max snake length, basic score, final score, game duration, level number, level outcome, mastery level, max merged supergems, moves count, objective details, objective done, objective name, difficulty level, splashed jellies.
Pearl's Peril	23	Device id, Facebook id, observation day, timestamp, country, device type, operating system, version, age in game, apptime, session count, badge count, energy, coins earned, score earned, scene, scene mastery and time to finish scene.

In Table 2, one can observe that each game shares the same metrics on installation and sessions data but the round metrics are game dependent. For this thesis, we aim to develop a cross-game churn-prediction model that can be generalized easily and therefore only a small selection of metrics will be used to compute the final features. This process of feature creation will be discussed in subsection 3.2.3.

3.1.3 Sample selection

In order to keep the data processing and classification tasks time efficient, it is recommended to reduce the number of players in the datasets. With sample reduction, you select users to form a new dataset containing a subsample of the original dataset. In Hadiji et al. (2014), where five F2P games were analysed to develop a game-design-independent churn-prediction model, the sample size per game was set to 50,000 randomly selected players. For this current research, we sample 100,000 randomly selected device ID numbers per game from the installation datasets that have more than one registered play session. This random selection was made with the knowledge that, after data cleaning and processing, the sample size will still contain a decent number of players to perform the research. By using a randomized technique of sample selection, we eliminate bias by giving all users an equal chance to be chosen for the sub-sample (Rashka, 2015).

3.2 Pre-processing

The pre-processing of the dataset entails cleaning the dataset, deriving features of the metrics and selecting feature windows. These processes will be described in the following subsections.

3.2.1 Data cleaning

Within the raw dataset, there were examples found of players who did not play a session after the initial installation of the game. To control for players who only installed the game but never played a session, players were only considered if they had played more than one session within the game. This step was necessary to make sure that the created features would not result in missing values.

As our aim is to train models on game-design-independent features, specific metrics were selected that could be used to calculate the features. Because the round metrics differ for each game, we selected the game-design-independent metrics for all games. For some games, the metric names were different but they were measuring the same activity. In Table 3 the metrics selected per dataset can be observed.

Table 3. Selected game-design-independent metrics per game dataset

Dataset	**Number of Features**	**Metrics**
Installation datasets	6	Device ID, observation day, timestamp, country, device type and acquired by marketing effort.
Sessions datasets	3	Device ID, observation day, timestamp.
Rounds for Diamond Dash	5	Device id, observation day, timestamp, coin balance and gold balance.
Rounds for Jelly Splash	5	Device ID, observation day, timestamp, coins and game duration.
Rounds for Pearl's Peril	6	Device ID, observation day, timestamp, apptime, coins earned, time to finish scene.

The selected metrics in Table 3 represent activities that are present in any casual F2P game. The metrics cover basic in-game activities that are not related to the game-design, thus ensuring the generalizability of the classification task.

Wooga indicated that duplicate rows in the data could be present. Because of a known issue in the database, duplicate rows could be present in the datasets for all games. These rows are identical on all values of the metrics and therefore could be easily spotted within the datasets. These rows were omitted from the sample datasets.

As can be observed in Table 3, the timestamp metrics in the datasets are sub-divided into timestamp and observation day. To be able to calculate new features and time windows from these timestamps, the data was converted to meaningful date time objects. From these date time objects, we could calculate useful metrics. For each row in the session and rounds data, a metric was added which indicated the date and time the game was installed. From this metric we could calculate the **time in game** metric. This metric indicates the time that has elapsed since the player played the game up till the current time of the session/round. This **time in game** metric is especially useful when we selecting data within a certain time range, for example the feature window or observation window.

3.2.2 Time windows

As described in definition 4 in the introduction, churn is calculated as a function of time. Choosing the time window that accurately measures churn is very important for research on churn prediction.

Feature window:

A feature window in churn prediction determines the amount of data captured to calculate the features. In the literature, a 7-day feature window is used because it captures a full week of playing behaviour (Drachen et al., 2016). Thus, a 7-day feature window captures data which is not exposed to day-by-day changes. As discussed in section 2.7, most churn happens within a short period after installation of the game. Drachen et al. (2016) trained models on a dataset capturing one full day of playing behaviour to compare prediction performance. For this thesis, we develop datasets using two different feature windows. We have chosen a dataset which captures the activity of a full week of playing behaviour (t-7) because this controls for day-to-day differences in playing behaviour. In addition, we have chosen datasets which capture a full day of playing behaviour (t-1). This decision was made to test whether the first day of game-activity can predict churning behaviour during the second week. The starting point for both feature windows is the moment when the game is installed on the player's device.

Observation window

An observation window in churn prediction observes if a player remains playing (no churn) or stops playing (churns). In section 2.7, different observation windows were discussed across the literature. For our current study, one approach was used to measure what observation window accurately captures churning behaviour. The approach by (Runge et al., 2014) generates examples of inactivity days between sessions. These examples are combined and plotted in a histogram with a cumulative distribution line indicating the percentage of players who are inactive. This method revealed that about 2% of the players exceed seven days of inactivity before playing a new session. A churn observation window of seven days captures 98% percent of actual churning behaviour. The plotted histogram with relative frequency of days of inactivity is presented in Figure 3. The cumulative percentages per game are displayed in Table 4.

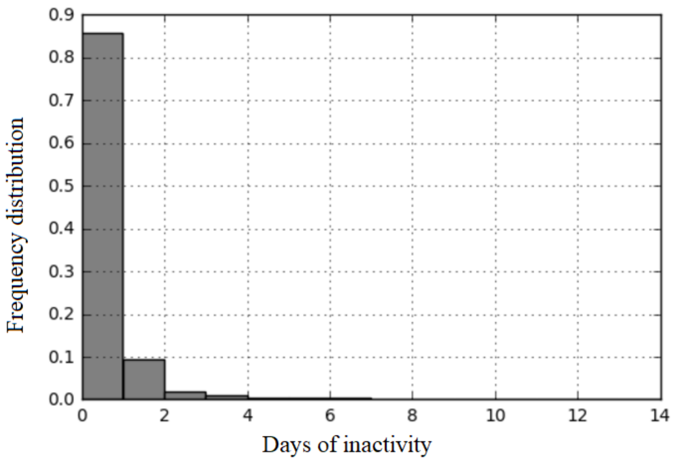

Figure 3. Histogram of Days of Inactivity for Diamond Dash

Table 4. Percentage of intersession times within 7-day time window

Game	Cumulative percent
Diamond Dash	98.3%
Jelly Splash	97.9%
Pearl’s Peril	99.2%

3.2.3 Features

After the sample selection, data cleaning and metric selection/transformation, we could start building the final datasets. These selected game metrics, as described in 3.1.2, can now be used to create features. For datamining purposes, the final datasets used for prediction contain a row for each player. We used the installation dataset for this. The sample of one hundred thousand players per game was reduced by only looking at unique device IDs. Apparently, some users reinstalled the game after playing it in the past and this resulted in duplicate device IDs.

With the installation dataset for each game as the foundation the final dataset, we kept the **acquired** metric as feature. This because it might hold information that can be used by the prediction algorithm for classifying churners.

The first computed features for the final datasets were **Number of Sessions** and **Number of Rounds**. They were easily computed by grouping the data by the device IDs of the players and counting the number of entries per device ID in the sessions dataset and rounds dataset. After calculating the feature for all games, a new feature was calculated by dividing the **Number of Rounds** by the **Number of Sessions**. This new **Average Rounds per Session** feature gives an indication of the player’s commitment to the game.

As discussed in section 2.6, the **Intersession Time** is mentioned as a very predictive feature for player churn in the literature. The feature measures the average time between sessions per player. If a player has a high **Intersession Time** value, the player is not committed to the game and will be more likely to churn. For the final dataset, we first transposed the timestamps of the sessions datasets to meaningful date time objects. Subsequently, we calculated the difference between the time stamps and average these results per player. This time delta object was transposed into float values by extracting the hours between sessions. For example, a time delta value of 0 days 03:59:39.268 was transposed into 3.994241. The final **Intersession Time** feature measured the time between sessions on average, displayed in hours. This transposing of time objects into floats was done to be able to easily use the feature in a prediction algorithm.

As observed in Table 3, the rounds datasets differed per game. Pearl's Peril recorded the *time to finish scene*, Jelly Splash the *game duration* and Diamond Dash did not record such data. To calculate the average round time per player, thus required a different approach per game. The fact that Diamond Dash does not record a round time is not surprising, as the aim of the game is to get the highest score in each round of 60 seconds. Each registered round took 60 seconds to complete which could be easily computed in the Round Time metric. This metric was used to compute the **Total Round Time** feature. The **Total Round Time** feature summed all the rounds per player within the feature window to reveal each player's time investment into the game.

The approach for Jelly Splash and Pearl's Peril slightly differed. For Jelly Splash the *game duration* metric was gathered for all rounds. The average per player was computed into the Round Time metric and the sum was computed into the **Total Round Time** feature. For Pearl's Peril the same algorithm was used but instead of using the *game duration* metric, the *time to finish scene* metric was used.

For all game datasets, the **Total Round Time** feature was used to calculate the **Average Playtime per Session.** This feature was easily computed by dividing the **Total Round Time** for each player by the **Number of Sessions** feature. This feature indicates on average how much time a player spends on each playing session.

The last computed feature of the final dataset is **Current Absence Time.** This feature is also mentioned in the literature as being highly predictive of player churn. This feature was computed by subtracting the length of the feature window by the date time object in the sessions dataset. This resulted in a time delta object which indicated the time period between the current session and the end of the feature window. Subsequently, the data was sorted and the last session of each player was extracted. The player's last session within the feature window was the start pointing of the **Current Absence Time** feature. Finally, this date time object was divided into hours. The same technique was used as with the **Intersession Time** to output a meaningful float type feature that is easily interpretable by a prediction algorithm.

3.2.4 Player churn

The aim of this thesis is to predict churn across games and thereby identify players who will leave the game in the future. Because of the structure of the datasets provided by Wooga, it was not possible to link session and rounds data to individual players. Because of this situation, the approach of this thesis is to measure all the activities by unique device IDs. Instead of predicting player churn, we predict the churn of devices. For most players this new approach does not differ from predicting churn based on unique players. However, some devices will be used by more than one player and thus player behaviour on the individual level is not measured.

As defined in the introduction section, player churn is often calculated as a binary classification problem. For this thesis, the same approach was taken. A player was considered a churner if there was no session recorded during the observation window. As discussed in subsection 3.2.2, by plotting the average absence time, we capture 98% of real churn behaviour with an observation window of seven days.

For each player the feature window started with the installation timestamp of the game. The feature window ended seven full days after the initial installation timestamp. After the 7-day feature window, the observation window started until the 14th day after installation of the game. This can be observed in Figure 4.

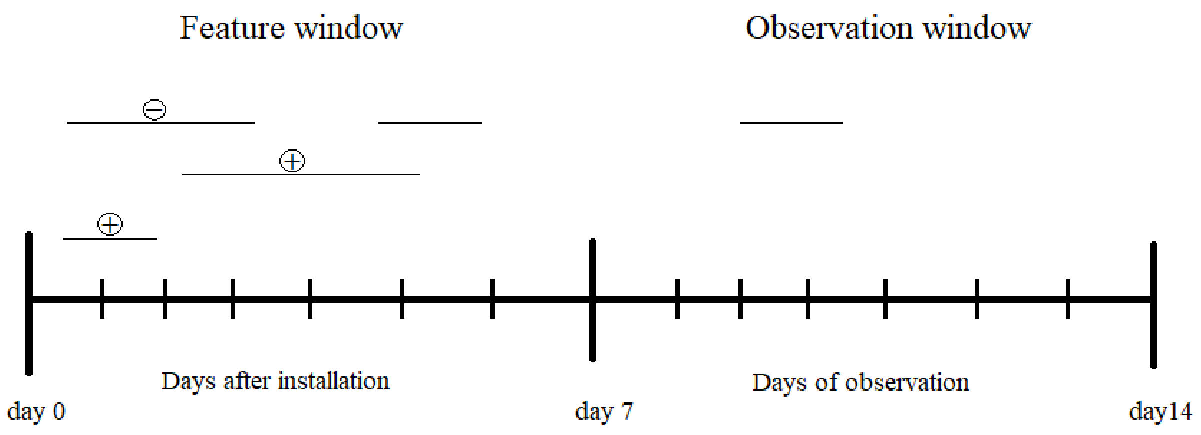

Figure 4. An example of player activity within the feature and observation window. The positive (+) examples are churned players, the negative example (-) is a non-churning player.

This approach of determining churning behaviour resulted in the assignment of class labels to the instances in the dataset.

3.2.5 Missing values

The data cleaning and feature creating resulted in some missing values in the data. The intersession time feature generated the most missing data points. This can be explained by players who did not have a second session within the feature window. Without this second session, there is no interval with which to compute the time between sessions. For these examples, missing values appeared in the intersession time feature. For the datasets with feature window t-7, this meant that 3.9% of instances contained missing values and were omitted from the dataset. For the datasets t-1, on average 8.0% of the data contained missing values and were omitted from the further calculations.

3.3 Data visualization:

With the creation of the features, the creation of the class labels and the omitting of the missing values, the final datasets were created. To get insight into the data, this section explores the datasets on which the algorithms will be trained for predicting player churn.

3.3.1 Descriptive statistics

In Tables 5 and 6 the dataset descriptive statistics are shown for the Diamond Dash datasets. For each feature in the dataset, the mean, standard deviation, minimum and maximum values are displayed. The acquired feature is a binary feature. Value 1 means that the player was acquired using a marketing effort.

Table 5. Diamond Dash dataset descriptive statistics for t-7

Features	**Mean**	**Std.**	**Min**	**Max**
Acquired	0.013	0.114	0	1
Number of Sessions	17.710	66.756	2	6691
Number of Rounds	34.304	149.083	1	18217
Average Rounds per Sessions	1.790	1.524	0.07	51.25
Intersession Time (hours)	6.375	10.897	0	167.791
Total Round Time (seconds)	2058.225	8944.982	60	109300
Current Absence Time (hours)	109.545	63.985	0	168
Average Playtime per Session (seconds)	107.255	91.449	0.444	3075

On average, a Diamond Dash player starts about 18 sessions in the first week after installation. The number of rounds per player is higher because one session can contain multiple rounds. The average rounds per session ratio is therefore 1.790. On average, players of Diamond Dash wait 6 hours before starting a new session. The total playtime is about 34 minutes on average within a 7-day feature window. The high standard deviation of current absence time can be explained by the fact that some players churn almost immediately after the first sessions and some players stay engaged with the game almost the entire week. The max value of current absence is 168, which is equal to 7 full days, which indicates the players who immediately churn.

Table 6. Diamond Dash dataset descriptive statistics for t-1

Features	**Mean**	**Std.**	**Min**	**Max**
Acquired	0.013	0.115	0	1
Number of Sessions	8.412	22.686	2	1808
Number of Rounds	14.738	46.198	1	3094

Average Rounds per Sessions	1.810	1.667	0.09	51.250
Intersession Time (hours)	1.180	2.103	0.044	23.999
Total Round Time (seconds)	884.270	2771.897	60	18564
Current Absence Time (hours)	17.293	8.872	0	24
Average Playtime per Session (seconds)	108.585	100.036	0.444	3075

The descriptive statistics in Table 6 indicate the values for Diamond Dash, which we recorded during the first 24 hours after installing the game (t-1). In these 24 hours, players play on average of 8 sessions with 14 rounds. The average playtime per session within the first 24 hours does not differ much with the average playtime per session within a week. The same count for the descriptive statistics of Jelly Splash which can be found in Appendix A. For Pearl's Peril, the average playtime per session increases with time.

3.1.2 Churn visualization

In Table 7 the distribution of player churn for the training datasets for t-7 can be observed. The numbers of instances per training set are comparable. As mentioned in section 3.2, the initial sample size of 100,000 instances per game was reduced after computing the features.

Table 7. Training sets distribution of churn and non-churn for t-7

	Diamond Dash training set	**Jelly Splash training set**	**Pearl's Peril training set**
Churners	0.652	0.559	0.484
Non-churners	0.348	0.441	0.516
Number of instances	87,069	90,084	87,979

When observing the distributions for class imbalance in the datasets, one can notice that the distribution of churners and non-churners is quite balanced. Especially compared to the distributions mentioned in the literature.

The distribution of the training sets for t-1 can be observed in Table 8. The number of instances per dataset is less than those for t-7. This can be explained by the fact that the feature window of t-1 yielded more missing values that were omitted from the final datasets.

Table 8. Training set distribution of churn and non-churn for t-1

	Diamond Dash training set	**Jelly Splash training set**	**Pearl's Peril training set**
Churners	0.658	0.490	0.474
Non-churners	0.342	0.510	0.526
Number of instances	83.666	88.061	71.310

In Tables 9 and Table 10, the test set distributions can be observed. These datasets will be used for evaluating the model on unseen data. The data is comprised of data from the other two games and therefore is considered out-of-sample data. The distribution of churners and non-churners is decently balanced, although there is a slightly imbalanced test dataset for Pearl's Peril.

Table 9. Test set distribution of churn and non-churn for t-7

	Diamond Dash test set	**Jelly Splash test set**	**Pearl's Peril test set**
Churners	0.522	0.568	0.605
Non-churners	0.478	0.432	0.395
Number of instances	180.737	175.048	179.827

For the Diamond Dash test set with a one-day feature window, the test set consists of more non-churners than churners. This could yield worse prediction performance because the algorithm will be trained on a dataset with more examples of churners, as can be observed in Table 7.

Table 10. Test set distribution of churn and non-churn for t-1

	Diamond Dash training set	**Jelly Splash training set**	**Pearl's Peril training set**
Churners	0.483	0.574	0.572
Non-churners	0.517	0.426	0.428
Number of instances	159.371	154.976	171.727

The difference in playing behaviour for churning and non-churning players can be observed in the following figures. Figure 5 plots the total number of sessions by churning and non-churning player within the first week of playing the game. One can notice that the number of sessions decreases after the first day of the installation of the game

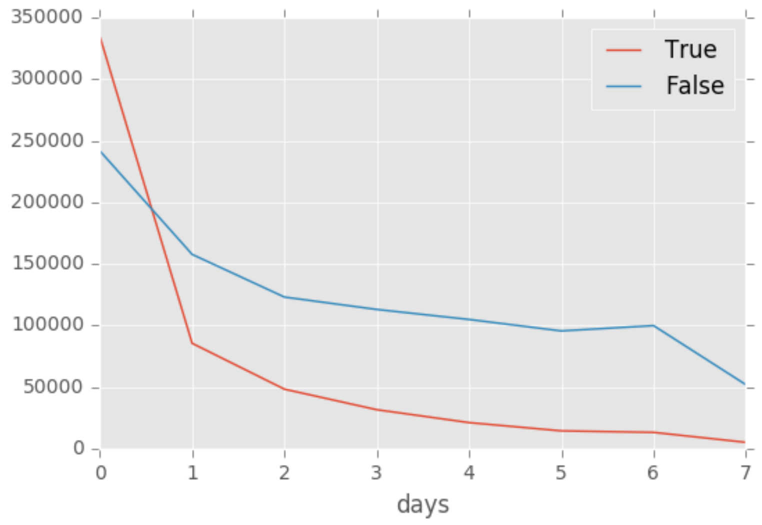

Figure 5: Comparing the total number of sessions played per day for Diamond Dash for churners (True) and non-churners (False) t-7.

This difference between the numbers of sessions can be observed in Figure 6, where the average number of sessions is plotted per group.

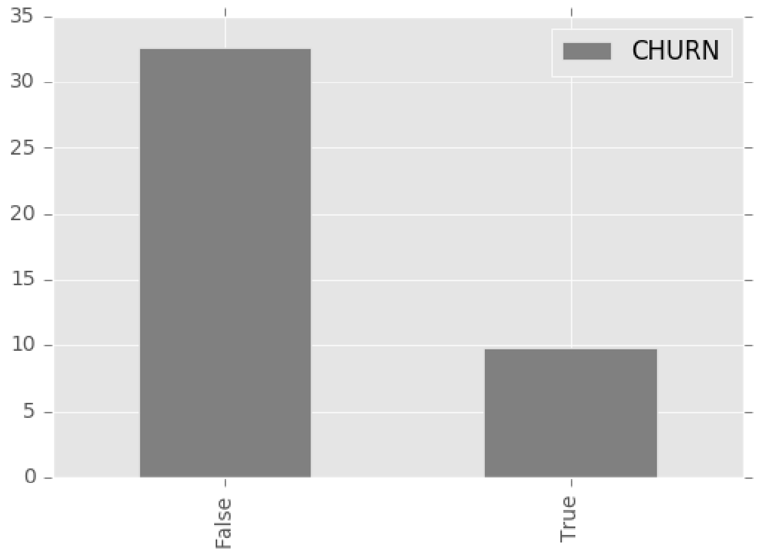

Figure 6. Average number of sessions for churners (True) and non-churners (False) t-7.

Both plots indicate that players who churn have different in-game behaviour than players who remain in the game. The experiments, designed in the next section, aim to separate both groups by analysing these behaviour differences in the datasets and thereby predict player churn.

3.4 Experimental procedure

This section will further elaborate on the design of the experiments conducted for this thesis. With the omitting of missing values and the creation of the class labels, four final datasets were created. For Jelly Splash two datasets were available, the dataset with data on Android devices and the dataset with data on IOS devices. In accordance with the data owners, the decision was made to leave the Android dataset out of the experiments. By excluding the Android dataset, the final datasets hold data solely on IOS devices for the games Jelly Splash, Diamond Dash and Pearl's Peril.

The experiments described in this section aim to answer the proposed research questions.

RQ1: *To what extent can we predict player churn across multiple F2P games with game-independent input features?*

RQ2: *What are the most important features for predicting player churn across F2P games?*

RQ3: *What is the effect of different time windows on churn prediction across F2P games?*

3.4.1 Cross-game Churn prediction:

The aim of this study is to develop a cross-game churn-prediction model with game-design-independent features. For prediction purposes, this means that the trained model needs to generalize to out-of-sample data with decent prediction performance. This will be tested in the proposed experiments.

The datasets of all games will be used individually to train prediction algorithms. Each dataset will be split into 10 different subsets for cross-validation. The model will be trained on 9 subsets and validated on 1 subset. This results in 10 validation scores, which are averaged into one cross-validated score. With the use of grid search, different sets of parameters are tested to develop a model that scores the highest on classification accuracy.

The model with the highest accuracy will be used to predict out-of-sample data of the test set. In cross-game churn prediction, one can use out-of-sample data to test the performance of the prediction model. This means that a trained model on data of Jelly Splash will be tested on the data of Diamond Dash and Pearl's Peril. This testing data does not originate from the trainings dataset, but can be used to test the model because of the game-design-independent features that were used across all datasets.

3.5. Classification algorithms

In order to predict whether a player will churn, the prediction algorithm needs to assign each new unseen instance to a class label True or False. "True" means that the player has a higher probability to churn during the observation window and "False" means the opposite. Algorithms used in the literature to classify churners are discussed in section 2.5. For this thesis, the following algorithms will be tested:

decision tree, random forest, k-nearest neighbours and logistic regression. Grid search is used for parameter optimization which results in the highest 10-fold cross-validated accuracy score. The best-performing model will be used for the out-of-sample test set. The area under the ROC curve (AUC-ROC) and the area under the precision and recall curve (AUC-PR) are also observed and reported.

3.5.1 Decision tree

The decision tree algorithm is a classical natural model of learning (Daumé, 2012). Similar to other algorithms, it uses past data to predict unseen data. Decision tree is suited for binary classification problems. It builds a model on binary questions, splitting the answers into two different outcomes, e.g. yes or no, true or false. When training the model, these questions form one rule-based tree which forms the trained model. To predict the class label of unseen data, the algorithm traverses the tree, using the questions and the features of the instances. Each instance ends up in a leaf, determining its class label. The order in which the questions are asked is the most important aspect of the decision tree. A question that separates the most instances into classes is asked first. These questions (features) are the features with the highest prediction power in the dataset.

The max depth parameter of a decision tree allows one to simplify the decision tree. When the depth parameter is set to a certain number, the model is limited to the number of features it can query. The shallow decision tree created makes class label guesses when the max depth parameter is exhausted (Daumé, 2012). This simplified tree tends to improve the generalizability of the model, and therefore will be useful for the experiments of this thesis.

3.5.2 Random forest

The random forest algorithm uses a set number of independently created decision trees with a certain depth (Daumé, 2012). The features used in the tree are chosen at random and can be used multiple times in the same branch. As with the decision tree, the leaves of individual trees are filled with training data. The random forest algorithm works best when individual features are similar regarding their importance. This can be explained by the fact that the trees built are only using a small set of features.

As with the decision tree, the random forest algorithm can be simplified with the use of the max depth parameter (Daumé, 2012). For this thesis, grid search will be implemented to train the parameters to optimal values. The aim is to build a model that will be generalizable to out-of-sample data.

3.5.3 K-nearest neighbours

The k-nearest neighbours algorithm converts feature values to feature vectors. Each feature represents its own dimension in the feature space. This results in a multidimensional array of feature vectors on which one can apply geometric concepts for machine-learning purposes (Daumé, 2012). A straightforward way is to compute the distances between each vector in the multidimensional array.

When the trained model gets tested on the validation set, the new unseen data is plotted in the same dimensional space without a known-class label attached to it. This vector gets the class label from its nearest neighbour. The *k* parameter tells the model what number of nearest neighbours to consider when determining the class label for the unseen data point. The class label, provided by the majority of the neighbours, gets assigned to the data point. These distances between data points can be computed with the use of different distance functions. For the k-nearest neighbours algorithm, Euclidean distance is used most often (Daumé, 2012).

The determination of the k-value is important because it allows to overfit (low k-value) and underfit (high k-value) the data. For this thesis, we will test the model on out-of-sample data. This means that the model needs to be generalizable to out-of-sample data. For this reason, a high value of *k* is preferred. With a high k-value, the model will be less sensitive for overfitting the training examples.

3.5.4 Logistic regression

As discussed in section 5 of the related work chapter, logistic regression is used in most churn-prediction literature. This simple, but powerful, algorithm can be applied to binary and linear classification problems (Rashka, 2015). The model uses the *odds ratio* to determine the probability of a certain event. For predicting the binary class label 'churn', the odds ratio determines for example the probability that a player will churn (y = True). The logit function of a logistic regression is the logarithm of the odds ratio. This logit function is displayed in equation 1.

Equation 1. The logit function of a logistic regression (Rashka, 2015).

$$logit(p) = log\frac{p}{(1-p)}$$

After this logit transformation, the logit values are fitted using linear regression analysis. These predicted values are converted back to odds using the inverse natural logarithm. The class label with the highest odds ratio is selected as class label (Rashka, 2015).

The λ parameter of the logistic regression is called the regularization parameter and it controls how well the model fits the training data (Rashka, 2015). For this thesis, it is important to keep this in mind. Overfitting the training data means worse performance on the out-of-sample test set.

3.6 Implementation of experiments

This section describes the packages used for data cleaning, sampling, visualization, and classification.

3.6.1 Programming language

All steps in the data analysis were carried out using programming language Python (Python Software Foundation, 2017). The language was developed by Guido van Rossum in 1991. The current version used for this thesis is 3.6.1, which was introduced on the 21st of March 2017. The programming language is free to use, as are the Python interpreters that are available for many operating systems. For this thesis, all steps were carried out in Jupyter, an open source, free to use environment for multiple programming languages for data science.

3.6.2 Python libraries

Programming language python allows for several libraries to be used in association with the python language. All the libraries used in this thesis are commonly used by data scientists for data mining purposes. Table 11 presents the used libraries for all steps in the data analysis.

Table 11. Used python libraries for the experiments with the corresponding version numbers

Python library	**Used for**	**Version**
Pandas	Data cleaning, manipulation, computing new features, random sampling	0.19.2
NumPy	Working with arrays, data manipulation	1.12.1
Datetime	Converting data into meaningful date time objects	2.7
Scikit-learn	Machine-learning algorithms, training and testing of the models, Grid Search cross-validation, classification report	0.18.1
Matplotlib	Plotting data to figures	2.0.2

3.7 Evaluation criteria

The evaluation methods used in the literature on churn prediction for F2P games differ per paper. In Drachen et al. (2016), models are evaluated on classification accuracy, precision, recall and F1 scores, whereas in Runge et al. (2014), models are evaluated on area under the ROC curve (AUC).

Where classification accuracy is a straightforward measure of what proportion of examples are accurately predicted, the AUC-ROC is used to calculate the area under the receiver operating characteristic curve (ROC) (Daumé, 2012). The AUC-ROC metric can be used to measure model performance. The AUC scores tend to be higher values compared to classification accuracy scores. This can be explained by the fact that an AUC of 0.5 indicates complete random chance and AUC of 1.0 the best possible performance. An alternative to the AUC-ROC is the AUC-PR (Davis & Goadrich, 2006). The area under the precession and recall curve can be used complimentary to the ROC curve for datasets with imbalanced classes. Davis and Goadrich (2006) indicate that a curve only dominates in ROC space if it dominates in PR space. For this thesis, we therefore report both.

For this thesis we will follow earlier papers in this field and report and evaluate the best-performing models both on cross-validated accuracy and F1 values as of AUC-ROC performance. With this approach, we will be able to evaluate not only the trained models, but also compare the results with the results from earlier research in this field. The area under the precision and recall curve will also be evaluated for each model (also known as *average precision*). This will give a broad sense of the trade-off for each model as precision will be plotted as a function of recall (Rashka, 2015).

4. Results

To answer the research questions proposed in the introduction of this thesis, this chapter provides the results of the experiments explained in Chapter 3. Each section of this results chapter presents the results for one of the research questions. Section 4.1 presents the results of the different prediction models used for cross-game prediction of player churn. Section 4.2 presents the results of feature importance analysis. Finally, we present the results for the time windows used for predicting player churn.

4.1 RQ1: Predicting cross-game player churn

As described in Chapter 3, the final datasets were created with the removal of the missing values, duplicate rows and duplicate device IDs. Each game was represented in two datasets, one with data on the first 24 hours of playing data (t-1) and one dataset containing the first seven days of playing the game (t-7). Each dataset was trained and optimized separately, yielding different optimal parameters. PCA was not performed because of the low number of features in the dataset. Because of the lack of class-imbalanced datasets, the decision was made not to use class imbalance techniques. The majority baseline scores for each dataset are displayed in Table 12. The majority baseline indicates the accuracy score a classifier produces when it assigns the majority class to each instance (Daumé, 2012).

Table 12. Majority baseline scores for training and testing set per game and feature window

Feature window	Dataset	MB training set	MB test set
t-1	Diamond Dash	0.658	0.517
	Jelly Splash	0.510	0.574
	Pearl’s Peril	0.526	0.572
t-7	Diamond Dash	0.652	0.522
	Jelly Splash	0.559	0.568
	Pearl’s Peril	0.516	0.605

4.1.1 K-nearest neighbour

As explained in subsection 3.5.3, K-NN is trained and tested on all datasets. The *k* parameter, which can be adjusted to adjust the level of overfitting and underfitting of the training dataset, was optimized using grid search. The k-parameter that yielded the highest 10-fold cross-validated accuracy score was chosen. An extra check was performed to check for other optimization points with a higher k-value. This was done to make the model more generalizable to unseen data.

The results for the KNN classifier on the training set can be observed in Table 13. The Pearl’s Peril dataset yields the best performance on accuracy and area under the ROC and precision and recall curve for both feature windows.

Table 13. 10-fold cross-validated accuracy and AUC scores for KNN classifier on training data

Feature window	**Dataset**	**Accuracy**	**AUC-ROC / AUC-PR**
t-1	Diamond Dash	0.703	0.708 / 0.543
	Jelly Splash	0.637	0.682 / 0.676
	Pearl’s Peril	0.705	0.765 / 0.765
t-7	Diamond Dash	0.799	0.844 / 0.765
	Jelly Splash	0.761	0.821 / 0.791
	Pearl’s Peril	**0.824**	**0.893** / **0.907**

Subsequently, the trained models were tested on the out-of-sample dataset containing data of the other two games. The results are reported in terms of accuracy/F1-Scores and AUC and are presented in Table 14.

Table 14. Accuracy and AUC scores for KNN classifier on testing dataset

Feature window	**Dataset**	**Accuracy / F1-score**	**AUC-ROC / AUC-PR**
t-1	Diamond Dash	0.620 / 0.574	0.678 / 0.669
	Jelly Splash	0.676 / 0.594	0.720 / 0.638
	Pearl’s Peril	0.658 / 0.616	0.703 / 0.620
t-7	Diamond Dash	0.783 / 0.767	0.848 / 0.836
	Jelly Splash	0.801 / 0.741	0.865 / 0.842
	Pearl’s Peril	0.773 / 0.713	0.829 / 0.766

The KNN classifier performs quite well on the data capturing the first seven days of playing behaviour. The performance of datasets containing one day of playing behaviour is not as good but it does outperform the results of the majority baseline. It can be observed that the performance on the testing set for Jelly Splash for both time windows exceeds the performance on the training set.

4.1.2 Decision tree

As described in subsection 3.5.1, the decision tree classifier is useful for data exploration and class prediction purposes. As parameters for grid search, max features and max depth were analysed to find the highest-scoring parameter combination in terms of accuracy. The produced decision tree rules yielded important information about the feature importance and decision rules of the model. These results will be analysed in section 4.2.

The performance of the decision tree classifier on the training data can be observed in Table 15.

Table 15. 10-fold cross-validated accuracy and AUC scores for decision tree on training data

Feature window	**Dataset**	**Accuracy**	**AUC-ROC / AUC-PR**
t-1	Diamond Dash	0.715	0.723 / 0.588
	Jelly Splash	0.648	0.696 / 0.694
	Pearl’s Peril	0.713	0.777 / 0.753
t-7	Diamond Dash	0.805	0.852 / 0.759
	Jelly Splash	0.770	0.837 / 0.783
	Pearl’s Peril	**0.831**	**0.903** / **0.834**

The data of Pearl’s Peril again yields the best accuracies and AUC-ROC scores. The difference with other classifiers is the biggest for t-7. The AUC-ROC score is with 0.903 optimal which means the classifier was able to accurately separate the classes.

Table 16. Accuracy / F1-scores and AUC scores for decision tree on testing dataset

Feature window	**Dataset**	**Accuracy / F1-score**	**AUC-ROC / AUC-PR**
t-1	Diamond Dash	0.660 / 0.642	0.699 / 0.692
	Jelly Splash	0.697 / 0.633	0.748 / 0.670
	Pearl’s Peril	0.661/ 0.574	0.695 / 0.577
t-7	Diamond Dash	0.798 / 0.777	0.867 / 0.769
	Jelly Splash	**0.810** / 0.759	**0.874** / **0.825**
	Pearl’s Peril	0.780 / 0.690	0.842 / 0.805

Table 16 contains the performance of the test set. Similar to the results of the KNN classifier, the decision tree yields better testing performance of Jelly Splash compared to the training performance. Feature windows t-7 yields decent prediction performance and t-1 does not perform as well.

4.1.3 Random forest

In subsection 3.5.2 the random forest classifier was discussed. Similar to the decision tree, random forest takes parameters max depth and max features to optimize the trained model. The depth of the individual trees determines the number of questions to ask before the decision for the class label is made. Simple trees with low depth values tend to generalize better to unseen data.

Table 17 displays the 10-fold cross-validated accuracy and AUC scores. The scores are decent for both time windows. The training performance for Jelly Splash yields worse accuracies and AUC-ROC scores.

Table 17. 10-fold cross-validated accuracy and AUC scores for random forest on training data

Feature window	**Dataset**	**Accuracy**	**AUC-ROC / AUC-PR**
t-1	Diamond Dash	0.717	0.727 / 0.577
	Jelly Splash	0.651	0.699 / 0.700
	Pearl's Peril	0.716	0.783 / 0.784
t-7	Diamond Dash	0.805	0.855 / 0.794
	Jelly Splash	0.772	0.839 / 0.825
	Pearl's Peril	**0.833**	**0.906** / **0.924**

The testing scores in Table 18 are overall the best seen so far. It seems random forest yields the best classification performance. With Jelly Splash, t-7 was the best performer in terms of accuracy and AUC-ROC. These results, however, to not differ much from the decision tree in 4.1.3.

Table 18. Accuracy / F1-scores and AUC scores for random forest on testing dataset

Feature window	**Dataset**	**Accuracy / F1-score**	**AUC-ROC / AUC-PR**
t-1	Diamond Dash	0.644 / 0.597	0.702 / 0.706
	Jelly Splash	0.701 / 0.644	0.753 / 0.686
	Pearl's Peril	0.676 / 0.597	0.714 / 0.645
t-7	Diamond Dash	0.799 / 0.781	0.870 / 0.876
	Jelly Splash	**0.816** / 0.764	**0.878** / **0.872**
	Pearl's Peril	0.808 / 0.761	0.845 / 0.807

4.1.4 Logistic Regression

Logistic regression, as discussed in subsection 3.5.4, is the most frequently used model for classifying churn. Logistic regression can be optimized by changing the parameter value λ. Using different values of this regularization parameter, grid search was implemented to find the best scores in terms of 10-fold cross-validated accuracy.

The training set scores can be observed in Table 19. As seen with the other classifiers, the results of t-7 are decent; the model can separate the classes with acceptable scores. For t-1 the scores are worse but still outperform the majority baseline.

Table 19. 10-fold cross-validated accuracy and AUC scores for Logistic Regression on training data

Feature window	**Dataset**	**Accuracy**	**AUC-ROC / AUC-PR**
t-1	Diamond Dash	0.711	0.711 / <u>0.566</u>
	Jelly Splash	<u>0.641</u>	0.693 / 0.691
	Pearl's Peril	0.712	0.777 / 0.777
t-7	Diamond Dash	0.804	0.849 / 0.788
	Jelly Splash	0.766	0.831 / 0.806
	Pearl's Peril	**0.825**	**0.897** / **0.909**

The test set scores of logistic regression, presented in Table 20, are consistent over the datasets. However, as with the training performance, the results of the test set are not improved compared to the results of random forest.

Table 20. Accuracy / F1-scores and AUC scores for logistic regression on testing dataset

Feature window	**Dataset**	**Accuracy / F1-score**	**AUC-ROC / AUC-PR**
t-1	Diamond Dash	<u>0.652</u> / 0.618	0.712 / 0.713
	Jelly Splash	0.705 / 0.645	0.754 / 0.663
	Pearl's Peril	0.672 / <u>0.578</u>	<u>0.710</u> / <u>0.634</u>
t-7	Diamond Dash	0.798 / 0.784	0.865 / **0.871**
	Jelly Splash	**0.808 / 0.761**	**0.874** / 0.847
	Pearl's Peril	0.778 / 0.696	0.825 / 0.774

4.1.5 Model comparison

The results from k-nearest neighbours, decision tree, random forest and logistic regression are combined in Table 21. This table allows us to compare the results per model in terms of accuracy and F1-scores.

Table 21. Cross-validated accuracy / F1-scores of different algorithms on the test set t-7

Classifier	**Diamond Dash**	**Jelly Splash**	**Pearl's Peril**
KNN	0.783 / 0.767	0.801 / 0.741	0.773 / **0.713**
Decision tree	0.798 / 0.777	0.810 / 0.759	0.780 / 0.690
Random forest	**0.799** / 0.781	**0.816** / **0.764**	**0.787** / 0.706
Logistic regression	0.798 / **0.784**	0.808 / 0.761	0.778 / 0.696

Table 21 presents random forest as the best overall performing model for classifying player churn on out-of-sample data. The performance of the algorithms is consistent. Surprisingly, the difference between the best-performing classifier (random forest) and the worst-performing classifier (KNN) is not substantial.

Also in terms of area under the curve, as Table 22 illustrates, random forest outperforms other classifiers. The AUC-PR scores also indicate that we can safely accept the results of the AUC. As discussed in section 3.7, Davis and Goadrich (2006) indicate that we can only appoint value to high AUC-ROC scores if the AUC-PR scores are high as well.

Table 22. AUC-ROC / AUC-PR performance of different algorithms on the test set t-7

Classifier	**Diamond Dash**	**Jelly Splash**	**Pearl's Peril**
KNN	0.848 / 0.836	0.865 / 0.842	0.829 / 0.766
Decision tree	0.867 / 0.769	0.874 / 0.825	0.842 / 0.805
Random forest	**0.870** / **0.876**	**0.878** / **0.872**	**0.845** / **0.807**
Logistic regression	0.865 / 0.871	0.874 / 0.847	0.825 / 0.774

These results indicate that random forest can separate the class labels of the unknown instances the best.

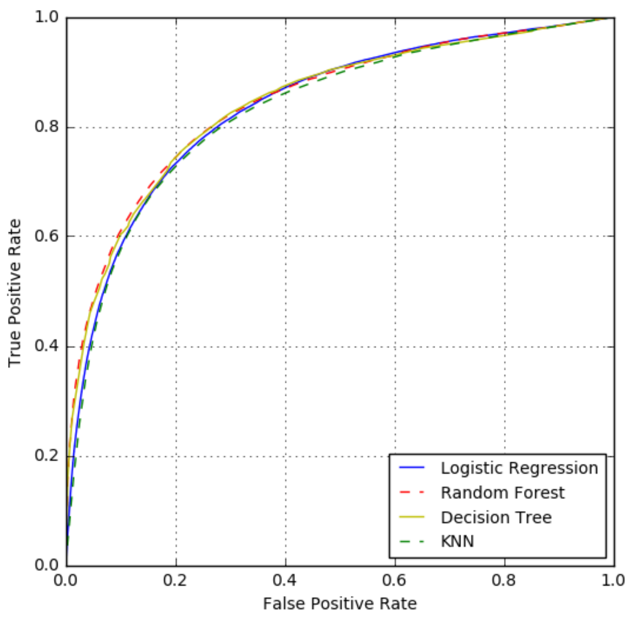

Figure 7. ROC Comparison for Pearl's Peril on test set t-7.

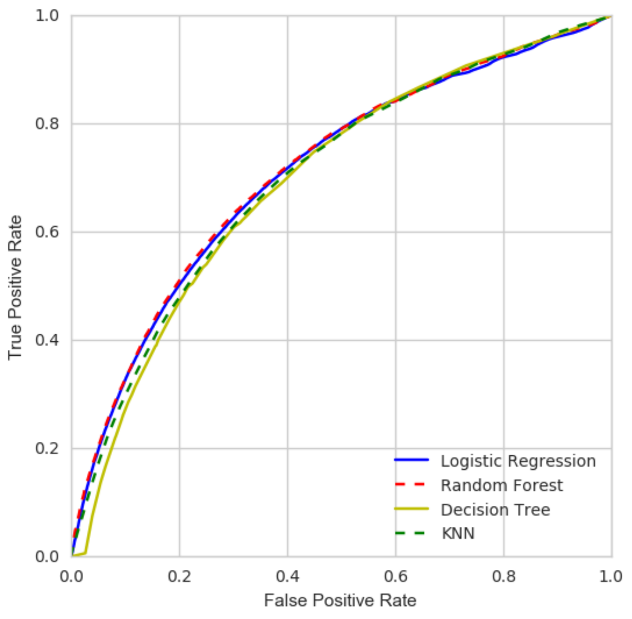

Figure 8. ROC Comparison for Pearl's Peril on test set t-1.

In Figures 7 and 8 the corresponding ROC curves for the Pearl's Peril dataset are compared. A ROC curve plots the true positive rate against the false positive rate. A straight diagonal ROC line indicates complete randomness in prediction performance. A line close to the 1.0 point of true positive rate at low levels of false positive rate indicates a nearly perfect classifier. Figure 7 indicates the steepest curve for random forest, meaning the model is the best at predicting the actual churners. The figure also visualizes the minor differences between the different models. The ROC curve in Figure 8 represents the ROC curves for Pearl's Peril with a 1-day feature window. The curve is less steep, indicating lower prediction performances. The corresponding ROC and PR curves for the remaining games can be observed in Appendix C.

When we observe the difference in training performance compared to testing performance, we can use a ratio which divides the AUC-ROC test set scores by the AUC-ROC validation set performance. These ratios are displayed in Tables 23 to 26. Table 23 displays the AUC-ROC ratios for the 7-day feature windows datasets. Values higher than 1.0 indicate that the test set outperformed the validation set. This is the case for Diamond Dash and Jelly Splash, but not for Pearl's Peril.

Table 23. Out-of-sample AUC-ROC divided by sample AUC-ROC ratio (CV-10) t-7

Classifier	**Diamond Dash**	**Jelly Splash**	**Pearl's Peril**
KNN	1.005	1.054	0.928
Decision Tree	1.018	1.044	0.932
Random Forest	1.018	1.046	0.933
Logistic Regression	1.019	1.052	0.920

For feature window t-1, the ratios differ, as can be observed in Table 24. Only Jelly Splash outperforms the validation scores.

Table 24. Out-of-sample AUC-ROC divided by sample AUC-ROC ratio (CV-10) t-1

Classifier	**Diamond Dash**	**Jelly Splash**	**Pearl's Peril**
KNN	0.958	1.056	0.913
Decision tree	0.967	1.075	0.894
Random forest	0.966	1.077	0.912
Logistic regression	1.001	1.088	0.914

Table 25 displays the results for the area under the precision and recall curve. These show similar results compared to Table 23 but the differences are increased. For Pearl's Peril, the test set performance is worse than the validation performance.

Table 25. Out-of-sample AUC-PR divided by sample AUC-PR ratio (CV-10) t-7

Classifier	**Diamond Dash**	**Jelly Splash**	**Pearl's Peril**
KNN	1.093	1.064	0.845
Decision tree	1.013	1.054	0.965
Random forest	1.103	1.057	0.873
Logistic regression	1.105	1.051	0.851

This difference in performance is even greater when analysing the results for the 1-day feature window. While the model trained on the Diamond Dash dataset seems to generalize well to unseen data, the models for Jelly Splash and Pearl's Peril do not. This difference can be observed in Table 26.

Table 26. Out-of-sample AUC-PR divided by sample AUC-PR ratio (CV-10) t-1

Classifier	**Diamond Dash**	**Jelly Splash**	**Pearl's Peril**
KNN	1.232	0.944	0.810
Decision tree	1.177	0.965	0.766
Random forest	1.224	0.980	0.823
Logistic regression	1.260	0.959	0.816

4.2 RQ2: Feature importance

This section reports the results for feature importance analysis. This will form the basis upon which the second research question – *What are the most important features for predicting player churn across F2P games?* – can be answered.

4.2.1 Random forest feature importance:

As discussed in subsection 3.5.1 and subsection 3.5.2, the decision tree and random forest models can be used to visualize important information about the dataset. The decision trees that are constructed provide information on logical classification steps and feature importance. A feature that separates the highest number of instances into classes is used first until all features are used.

These models, in the python library sklearn, have the ability to plot the relative feature importance to a graph. This visualizes each feature with its relative importance compared to other features. Figures 9 and 10 show these relative feature importances for two feature windows.

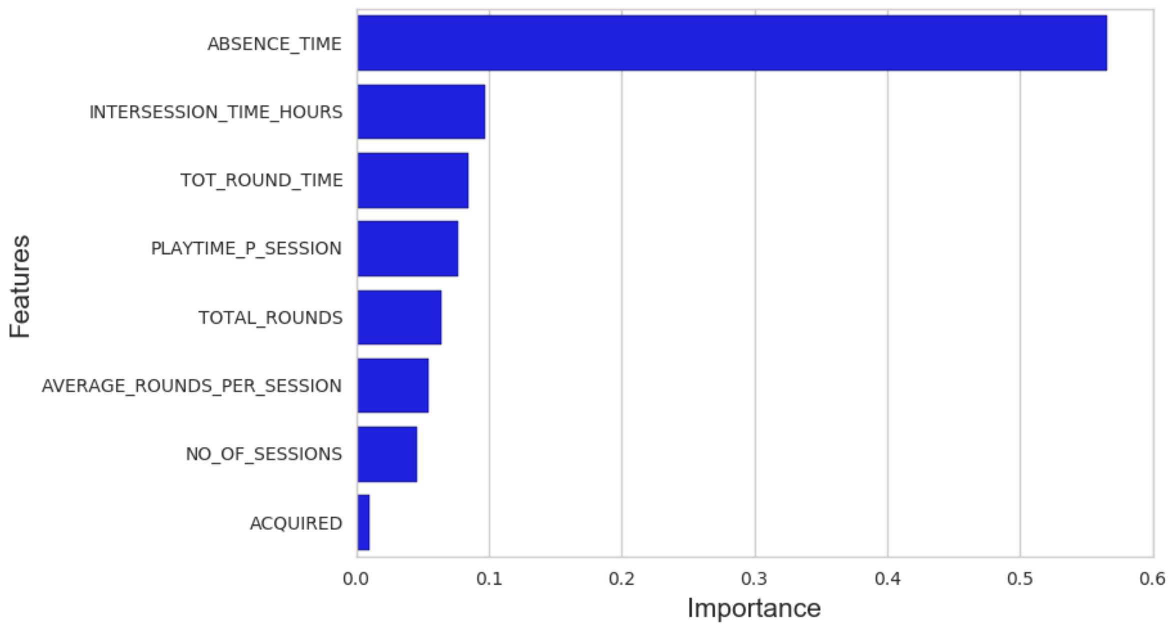

Figure 9. Feature importance decision tree classifier (Pearl’s Peril dataset t-7)

For the dataset with a 7-day feature window, the feature current absence time is the feature with the best prediction performance, followed by intersession time, total round time and playtime per session. The acquired feature contributes the least amount of classification performance to the model.

The prediction strength of current absence time is weakened when we observe the results for the 1-day feature window dataset in Figure 10.

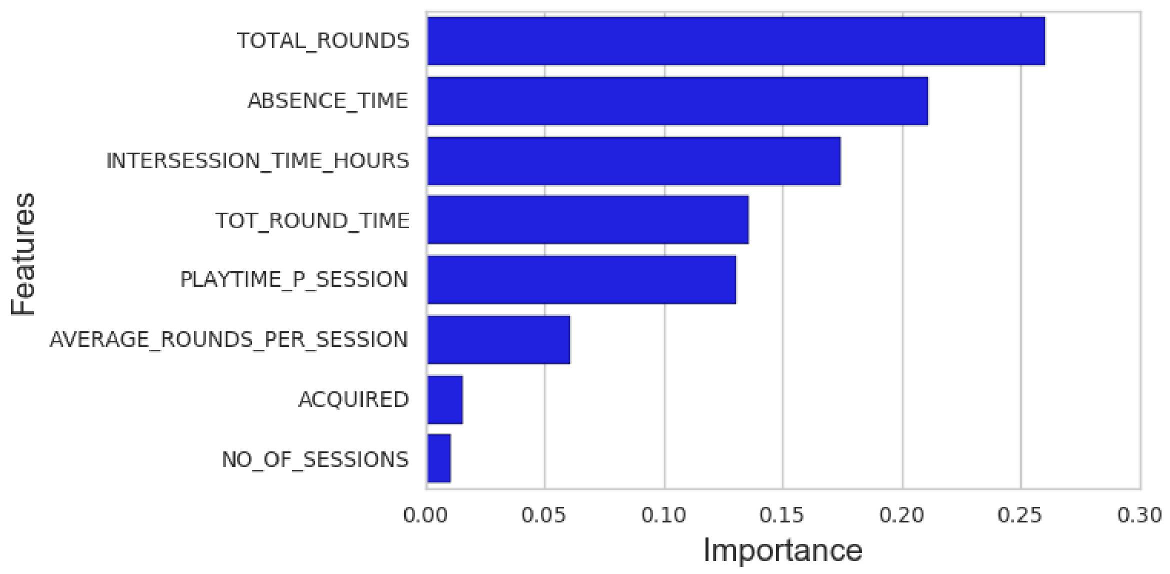

Figure 10. Feature importance decision tree classifier (Pearl's Peril dataset t-1)

This graph, showing feature importance for the same game with a different feature window, indicates a different feature as best predictor for player churn. For this dataset, the total number of rounds a player played is the best predictor, followed by current absence time.

For Diamond Dash and Jelly Splash, the feature importance results can be found in Appendix B. They show similar importance reduction for current absence time when comparing the feature windows. Surprisingly, the total number of rounds feature remains a relatively poor predictor in the 1-day feature window datasets. This might be related to the difference in game types. Jelly Splash and Diamond Dash are casual games compared to Pearl's Peril, which is a so-called hidden object game. As the way these games are played is different, so is the data.

4.3 RQ3: Time windows

In this section, we compare the effects of the feature window on player churn-prediction performance. This will contribute to answering the third proposed research question – *What is the effect of different time windows on churn prediction across F2P games?*

4.3.1 Comparing feature windows

In sections 4.1 and 4.2, the results for all models and time windows are presented. From these results we indicated that random forest yields the best prediction performance for both feature windows and all three datasets. This current section will compare the results from random forest by the two used feature windows.

As discussed in the previous sections, the 1-day feature window yields significantly lower prediction scores. Figure 11 visualizes this difference in terms of ROC curves and PR curves for the game Diamond Dash. The blue lines are the results for the best-performing random forest model trained on data capturing 7 days of game behaviour. The green lines indicate the results for the 1-day window.

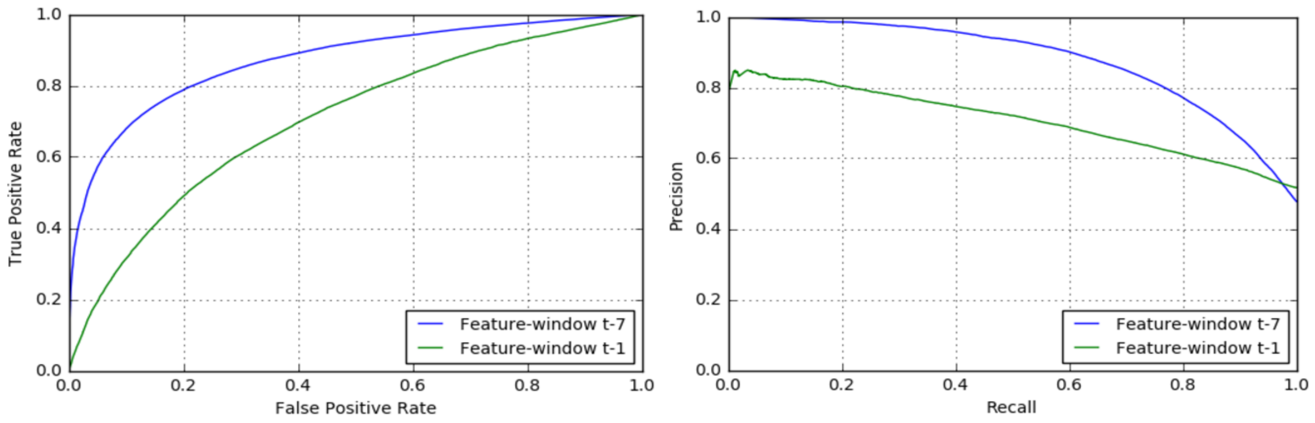

Figure 11. Random forest ROC curve (left) and PR curve (right) for Diamond Dash test set

When comparing the results of the ROC curve (left), it can be observed that the curve for the 1-day feature window is close to approaching the straight diagonal line, indicating a perfectly random classifier. The 7-day feature window, on the other hand, is approaching a perfect classifier. The curves of the PR curve indicate similar results and therefore we can accept the results of the ROC curve.

Figure 12 visualizes the different ROC curves and PR curves for Jelly Splash. Compared to the results for Diamond Dash, the 1-day feature window does not approach perfectly random classification performance. However, the 7-day feature window is still outperforming the 1-day feature window in terms of ROC and PR curves.

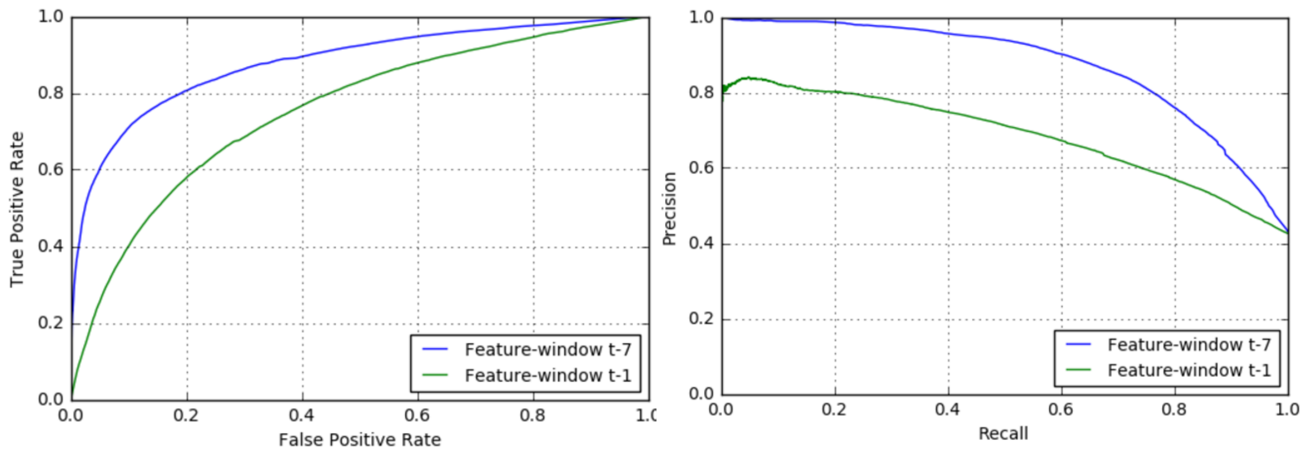

Figure 12. Random forest ROC curve (left) and PR curve (right) for Jelly Splash test set.

Finally, Figure 13 shows the ROC and PR curves for Pearl's Peril. Similar performance differences can be observed between the ROC curves of the 7-day feature window and the 1-day feature window. The ROC curve for the 7-day feature window performs worse compared to the curves of Diamond Dash and Jelly Splash. This might be explained by the difference in game genres. Pearl's Peril curves are trained on its own sample data of this hidden object game; however, it is tested on more casual game data of Jelly Splash and Diamond Dash. This results in minor performance decreases.

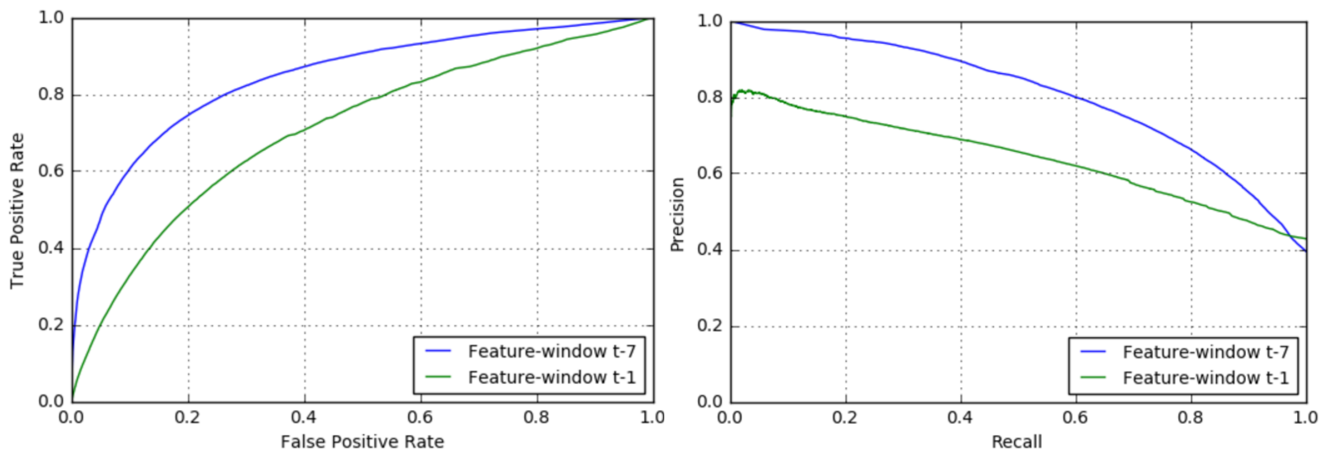

Figure 13. Random forest ROC-curve (left) and PR-curve (right) for Pearl's Peril test set

5. Discussion

In this chapter, we discuss the results presented in Chapter 4, with regard to the research questions stated in the introduction. The aim of this current research was to analyse cross-game telemetry data to predict churn behaviour across games using game-design-independent features. The following research questions were formulated in order to reach this goal:

RQ1: *To what extent can we predict player churn across multiple F2P games with game-independent input features?*

RQ2: *What are the most important features for predicting player churn across F2P games?*

RQ3: *What is the effect of different time windows on churn prediction across F2P games?*

5.1 Results per research question

RQ1: To what extent can we predict player churn across multiple F2P games with game-independent input features?

To answer this research question, the machine-learning models used in the literature were discussed in section 2.5. A selection of these models was put to the task of predicting player churn across games using three game datasets provided by Wooga. K-nearest neighbours, decision tree, random forest and logistic regression models were trained using game-design-independent features on each game separately. The best-performing model, in terms of accuracy, was used on the out-of-sample test set containing data of the other two games. Our results indicate that random forest is predicting player churn the best in terms of accuracy and AUC scores. These results were found across all games analysed. The performance difference between models, however, was not substantial.

Surprisingly, the performance on the out-of-sample test set yielded comparable performance results to the performance of the validation set. This is an indication that the features accurately capture game-design-independent game behaviour. In addition, the models trained on game A can predict player churn based on the data of game B and C. These results are in agreement with those obtained by Hadiji et al. (2014) where game-design-independent features were used to predict player churn across F2P games. The current research, however, found a different optimal classifier, using different datasets on a different player churn data generation method and tested the models on out-of-sample data.

RQ2: What are the most important features for predicting player churn across F2P games?

To answer the second research question, the literature was discussed in section 2.6. Drachen et al. (2016) found that current absence time and number of rounds played are the most important features for predicting player churn with data capturing 1-day of play behaviour. For a 7-day feature window, current absence time becomes more dominating in terms of relative importance.

For the current study, decision tree was used to plot the relative feature importance per feature window. The results are in agreement with those obtained by Drachen et al. (2016). For the data capturing one full day of play behaviour, the plots indicate that important features differ per game. For the hidden object game Pearl's Peril, the total number of rounds played and current absence time are considered the most important features, as opposed to current absence time and intersession time for the casual puzzle games Diamond Dash and Jelly Splash. The results for a 7-day feature window show the dominance of current absence time as most important feature, followed by intersession time. This dominance of current absence time can be explained by the fact that a large proportion of players churn soon after installing the game. These results are consistent with the literature.

RQ3: What is the effect of different time windows on churn prediction across F2P games?

In order to answer the third research question, the different time windows used in the literature were discussed in section 2.7. For this thesis, churn was measured in the way that best reflects real-world churn prediction problems (Hadiji et al., 2014). In subsection 3.2.2, the approach of Runge et al. (2014) was used to reveal that about 98% of real churn behaviour is captured using a 7-day observation-window. For this thesis, two feature windows were used to predict player churn for F2P games.

The 7-day feature window showed consistently decent performance and good ability to separate the different classes. The 1-day feature window showed less accuracy and lower AUC-ROC and AUC-PR scores. The latter did not show substantial improvement compared to the majority baseline scores and the ROC curves confirmed this by approaching the line of perfectly random classification. Subsequently, the results of the 1-day feature window tend to generalize worse to out-of-sample data, as was observed in subsection 4.1.5. This can be explained by the feature importance differences discussed in section 4.2. Current absence time is less dominant in the 1-day feature window datasets. Players who churn right after installation of the game do not impact the current absence time as heavily in the 1-day feature window datasets. This effect of immediate churn becomes more dominant in the 7-day feature window datasets.

5.2 Limitations

The data upon which these findings are based were limited to game data gathered from IOS devices. We did not have access to casual game data gathered via other devices or web-based platforms and therefore these data were not considered for this project. This means that the player population consisted only of players who own an Apple device. We did not find any literature on the in-game behaviour differences between users of different operating systems and can therefore not elaborate on the implications of this limitation for this thesis. Additionally, the sample consisted of players with more than one session. This was included intentionally to prevent creating missing values and to omit players who did not play the game. Furthermore, players who only played one session were not included in the dataset. A negative implication of this decision is that the datasets lack players who immediately churn after playing one session. A positive implication of this decision is that this resulted in surprisingly balanced classes.

5.3 Contributions and future research

The results of this thesis are in agreement with the findings of Hadiji et al. (2014), which showed that churn can be predicted using game-design-independent features. This thesis, however, deployed a new evaluation method by using out-of-sample data for testing purposes. Furthermore, the experiments conducted indicated random forest to be the best-performing algorithm for predicting player churn across games. The models, which are trained on datasets covering seven days of playing behaviour, show excellent generalization performance to out-of-sample data.

The practical relevance of these findings is that churn can be predicted across games. A newly launched casual game, which did not generate enough data itself to train prediction models, can now be tested on a comparable model. This can provide the game developer with useful information during the early stages after a game is released.

Future research could further investigate this relatively new domain of cross-game churn prediction by considering other game datasets, churn windows and by reviewing the prediction performance of other machine-learning algorithms. Moreover, additional features could be used that capture the in-game spending of coins and gold to derive valuable monetization information. Finally, more research can be conducted regarding why players churn and what keeps players playing.

6. Conclusion

The main goal of the current study was to develop a cross-game churn-prediction model with game-design-independent features. The following three research questions were formulated in order to research the topic.

RQ1: *To what extent can we predict player churn across multiple F2P games with game-independent input features?*

RQ2: *What are the most important features for predicting player churn across F2P games?*

RQ3: *What is the effect of different time windows on churn prediction across F2P games?*

To answer research question 1, predicting player churn across multiple F2P games using game-independent input features results in decent prediction performance. The best model found for this task is random forest. These results are consistent with the findings of Hadiji et al. (2014) on predicting player churn across games. Additionally, this study tested the prediction performance of the trained models on out-of-sample data. For 7-day feature window datasets, this yielded comparable results, indicating excellent generalization performance.

Regarding the second research question, the results indicated that the most important feature for predicting player churn across FTP games is current absence time. For a dataset which captures 7-days of playing behaviour, current absence time dominates other features in terms of relative importance. For a dataset with a 1-day feature window, the second-most-powerful feature, after current absence time, is game genre dependent; intersession time is more important for casual puzzle games, while total number of rounds is more important for hidden object games. These results are in line with the results found by Hadiji et al. (2014).

With regard to the third research question, the results indicated that the number of days captured within the features of the dataset influences the task of predicting player churn across games. A model trained on data capturing one day of game behaviour performs substantially worse than a model trained on a 7-day feature window. The performance scores of the 1-day feature window are close to the majority baseline scores, meaning the models only slightly outperform the results of random classification. These results are consistent with the literature. Subsequently, a model trained on one day worth of data generalizes worse to out-of-sample data.

References

Alsén, A., Runge, J., Drachen, A., & Klapper, D. (2016). Play With Me? Understanding and Measuring the Social Aspect of Casual Gaming.

Bauckhage, C., Kersting, K., Sifa, R., Thurau, C., Drachen, A., & Canossa, A. (2012). How players lose interest in playing a game: An empirical study based on distributions of total playing times. *Computational Intelligence and Games,* 139–146.

Blattberg, R. C., Byung-Do Kim, & Scott A. Neslin. (2008). Database Marketing: Analyzing and Managing Customers. *International Series in Quantitative Marketing.* New York, NY: Springer Science.

Buckinx, W., & Van den Poel, D. (2005). Customer base analysis: Partial defection of behaviourally loyal clients in a non-contractual FMCG retail setting. *European Journal of Operational Research,* 164, 252–268.

Casual Games Association. (2014). Towards the global games market in 2017: A broad look at market growth by screen & region. *Casual Games Sector Report.* Retrieved from https://issuu.com/casualconnect/docs/ccnewzoospringreport-pages

Dasgupta, K., Singh, R., Viswanathan, B., Chakraborty, D., Mukherjea, S., Nanavati, A. A., & Joshi, A. (2008). Social ties and their relevance to churn in mobile telecom networks. *Proceedings of the 11th international conference on Extending database technology: Advances in database technology,* 261, 668–677.

Drachen, A., Lundquist, E.T., Kung, Y., Rao, P.S., Klabjan, D., Sifa, R., & Runge, J. (2016). Rapid Prediction of Player Retention in Free-to-Play Mobile Games.

El-Nasr, M. S., Drachen, A., & Canossa, A. (2013). *Game analytics: Maximizing the value of player data.* London, England: Springer.

Gagné, A., Seif El-Nasr, M., & Shaw, C. (2012). Analysis of telemetry data from a real time strategy game: A case study. *Theoretical and Practical Computer Applications in Entertainment,* 10(3), 1–24.

GameHunters Club (2017, March 25). Retrieved from https://gamehunters.club/top-games/on-Facebook

Hadiji, F., Sifa, R., Drachen, A., Thurau, C., Kersting, K., & Bauckhage, C. (2014). Predicting player churn in the wild. *Conference on Computational Intelligence and Games,* 1–8.

Hui, S. K. (2013). Understanding Gamer Retention in Social Games using Aggregate DAU and MAU data: A Bayesian Data Augmentation Approach.

Kumar, V., & Petersen, J.A. (2012). Statistical Methods in Customer Relationship Management. Hoboken, NJ: John Wiley & Sons.

Laursen, G.H. (2011). Business analytics for sales and marketing managers: How to compete in the information age. Hoboken, NJ: John Wiley & Sons.

Lewis, C., Wardrip-Fruin, N., & Whitehead, J. (2012). Motivational game design patterns of ’ville games. *Foundations of Digital Games*, 12, 172–179.

Lunden, I., (2017, March 11). Retrieved from http://techcrunch.com/2012/08/14/facebook-says-itnow-has-235m-monthly-gamers-app-enter-hits-150m-monthlyusers

Mahlman, T., Drachen, A., Canossa, A., Togelius, J., & Yannakakis, G.N. (2010). Predicting player behavior in Tomb Raider: Underworld. *Proceedings of the 2010 IEEE conference on computational intelligence in games*, 178–185.

Nozhnin, D. (2017, April 10). Retrieved from http://www.gamasutra.com/view/feature/170472/predicting_churn_datamining_your_.php

R-project (2017, February 15). Retrieved from: https://www.r-project.org

Runge, J., Gao, P., Garcin, F., & Faltings, B. (2014). Churn prediction for high-value players in casual social games. *Conference on Computational Intelligence and Games*, 1–8.

Verbeke, W., Martens, D., Mues, C., & Baesens, B. (2011). Building comprehensible customer churn prediction models with advanced rule induction techniques. *Expert Systems with Applications*, 38(3), 2354–2364.

Weka. (2017, February 15). Retrieved from http://www.cs.waikato.ac.nz/ml/weka/index.html

Xia, G.E., & Jin, W.D. (2008). Model of customer churn prediction on support vector machine. *Systems Engineering-Theory & Practice*, *28*(1), 71–77.

Daumé, H. (2012). A Course in Machine Learning. C*hapter*, *5*, 69.

Rashka, S. (2015). Python Machine Learning. Birmingham, UK: Packt Publishing Ltd.

Python Software Foundation. Python Language Reference, version 2.7. Available at http://www.python.org

Davis, J., & Goadrich, M. (2006). The relationship between Precision-Recall and ROC curves. In *Proceedings of the 23rd international conference on Machine learning* (pp. 233 – 240). ACM.

Appendices and supplementary materials

Appendix A

Table 27. Jelly Splash dataset descriptive statistics for t-7

Features	**Mean**	**Std.**	**Min**	**Max**
Acquired	0.285	0.451	0	1
Number of Sessions	18.117	26.346	2	1367
Number of Rounds	43.107	61.766	1	1790
Average Rounds per Sessions	2.725	2.200	0.026	64.5
Intersession Time (hours)	9.640	16.283	0	168
Total Round Time (minutes)	5388.831	10865.969	0	203642
Current Absence Time (hours)	90.829	62.899	0	168
Average Playtime per Session (seconds)	282.237	268.025	0	11618.222

Table 28. Jelly Splash dataset descriptive statistics for t-1

Features	**Mean**	**Std.**	**Min**	**Max**
Acquired	0.269	0.444	0	1
Number of Sessions	8.172	9.121	2	266
Number of Rounds	20.595	24.242	1	592
Average Rounds per Sessions	2.766	2.345	0.018	54
Intersession Time (hours)	1.628	2.436	0	23.994
Total Round Time (minutes)	1931.063	2766.033	2	18555
Current Absence Time (hours)	14.664	9.364	0	24
Average Playtime per Session (seconds)	235.324	232.302	0.111	9680.333

Table 29. Pearl’s Peril dataset descriptive statistics for t-7

Features	**Mean**	**Std.**	**Min**	**Max**
Acquired	0.672	0.470	0	1
Number of Sessions	15.082	21.880	2	524
Number of Rounds	72.496	96.787	1	1483
Average Rounds per Sessions	4.857	4.068	0.007	55.400
Intersession Time (hours)	16.300	22.660	0	167.960
Total Round Time (seconds)	3689.874	4718.943	0	107576
Current Absence Time (hours)	74.927	64.166	0	168
Average Playtime per Session	281.230	299.115	0	13955

Table 30. Pearl’s Peril dataset descriptive statistics for t-1

Features	**Mean**	**Std.**	**Min**	**Max**
Acquired	0.672	0.470	0	1
Number of Sessions	4.983	4.764	2	272
Number of Rounds	25.533	25.496	1	507
Average Rounds per Sessions	5.600	5.210	0.007	69.800
Intersession Time (hours)	4.895	5.483	0	23.996
Total Round Time (seconds)	1495.953	1691.112	0	70139
Current Absence Time (hours)	11.424	8.962	0	24
Average Playtime per Session	335.377	414.449	0	35069.500

Appendix B: Feature Importance per dataset and feature window.

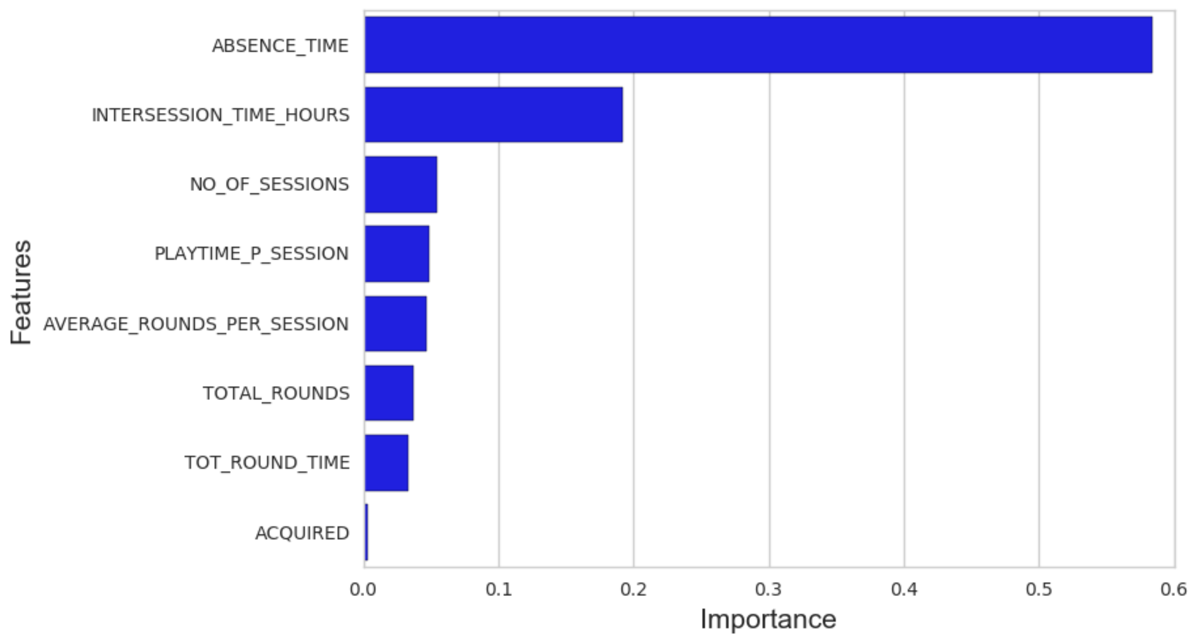

Figure 14. Feature importance decision tree classifier (Diamond Dash dataset t-7)

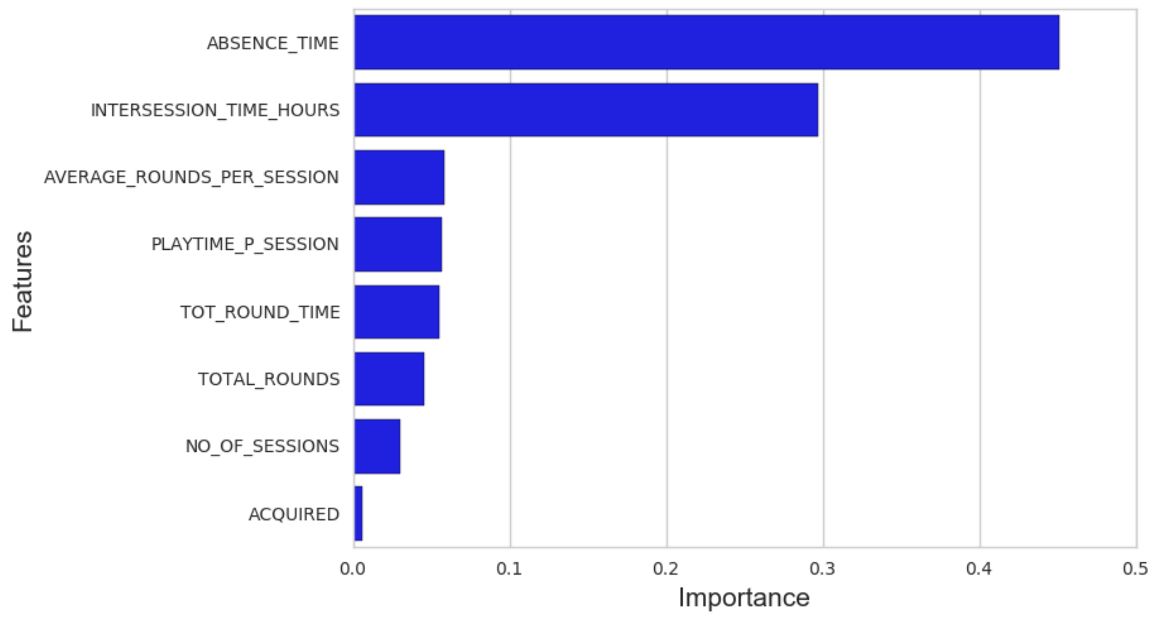

Figure 15. Feature importance decision tree classifier (Diamond Dash dataset t-1)

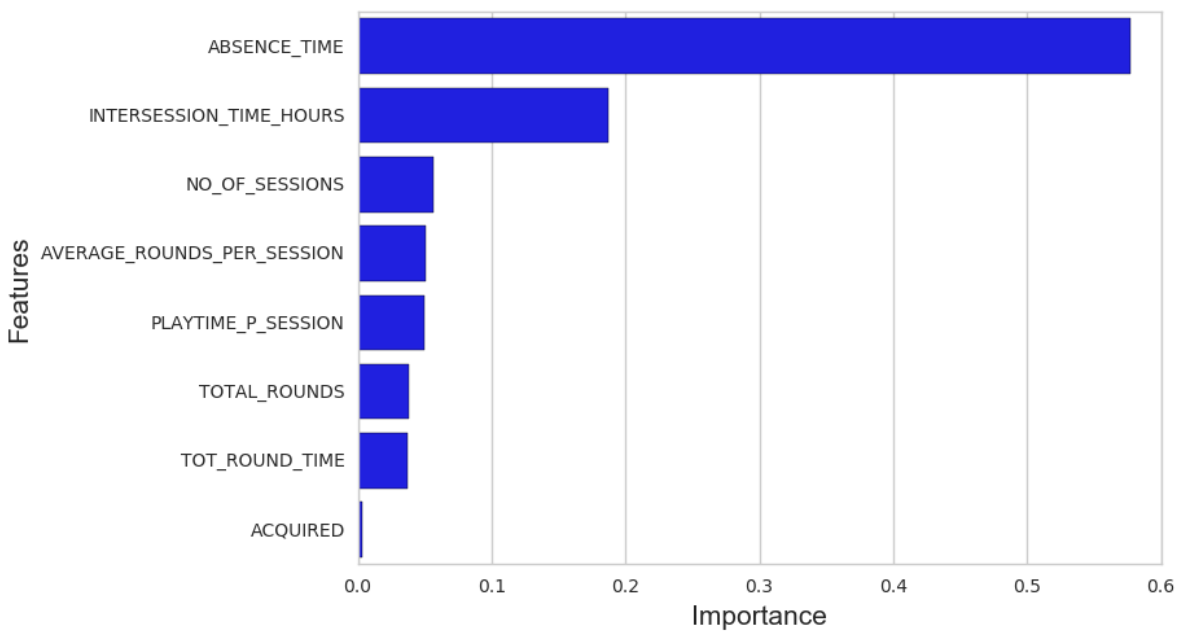

Figure 16. Feature importance decision tree classifier (Jelly Splash dataset t-7)

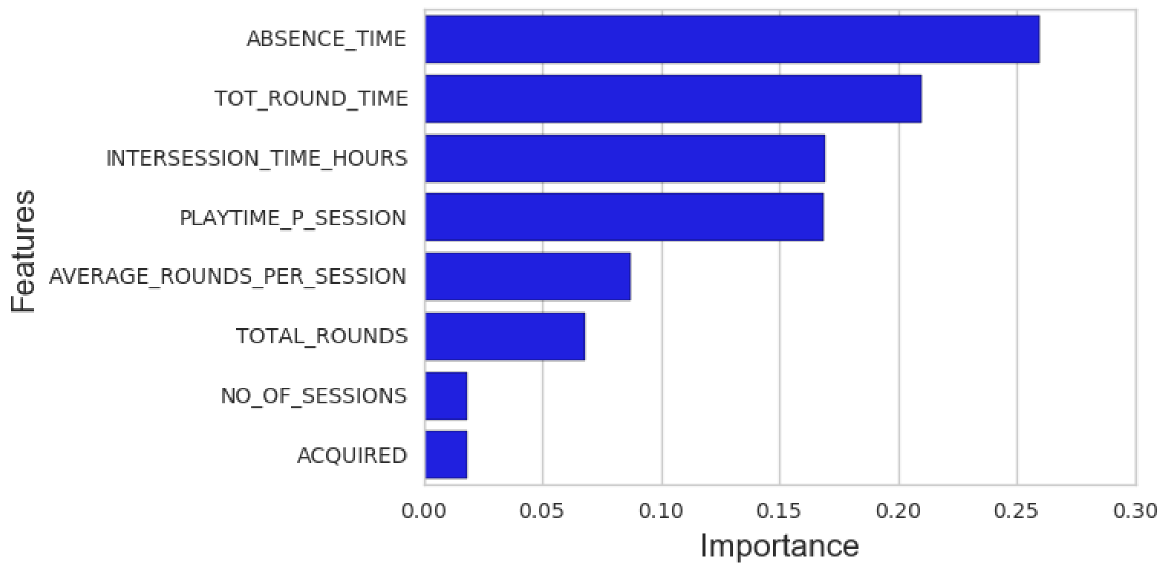

Figure 17. Feature importance decision tree classifier (Jelly Splash dataset t-1)

Appendix C: ROC & PR curves.

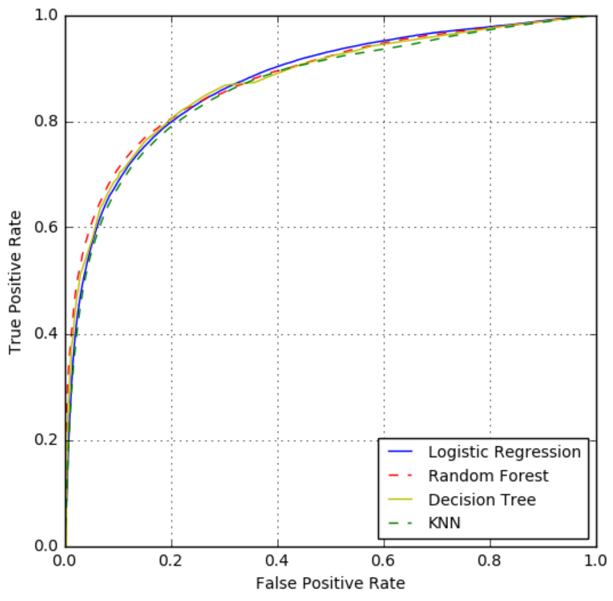

Figure 18. ROC Comparison on Jelly Splash Dataset t-7

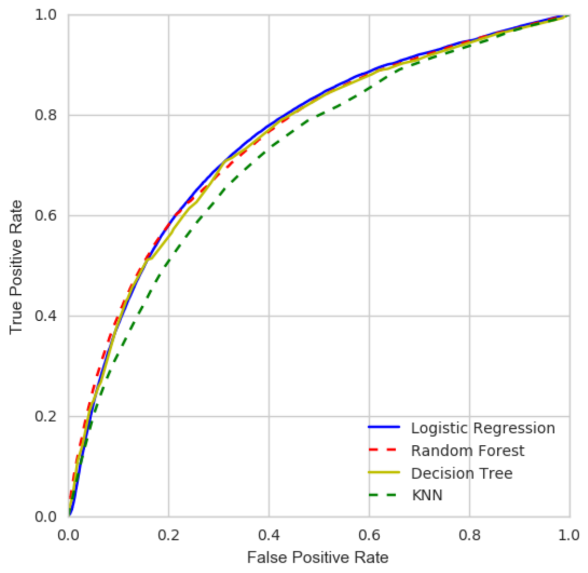

Figure 19. ROC Comparison on Jelly Splash Dataset t-1

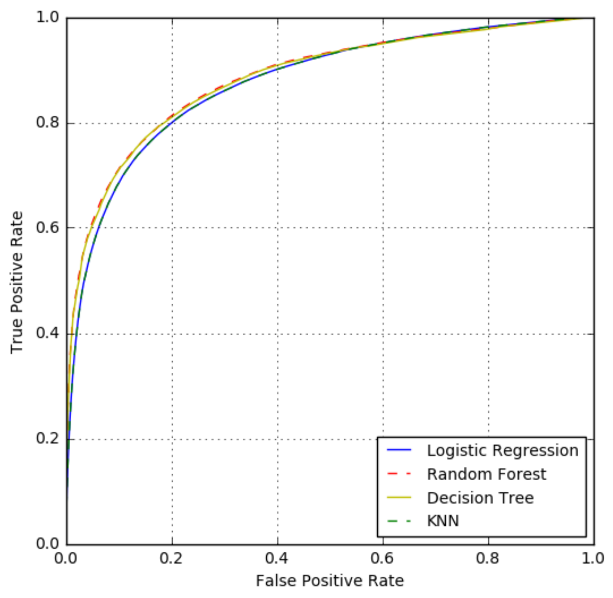

Figure 20. ROC Comparison on Diamond Dash Dataset t-7

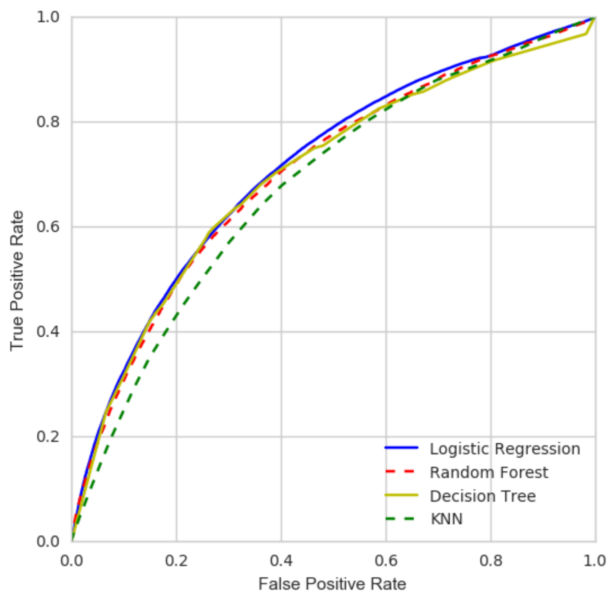

Figure 21. ROC Comparison on Diamond Dash Dataset t-1

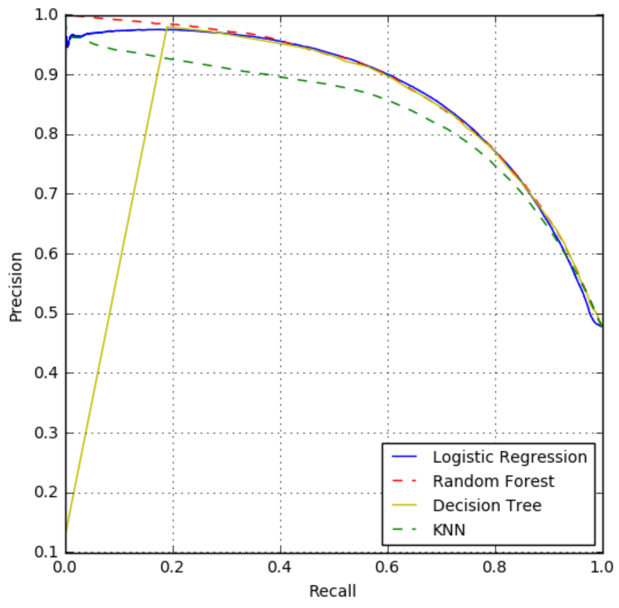

Figure 22. Precision-Recall curve comparison Diamond Dash Dataset t-7

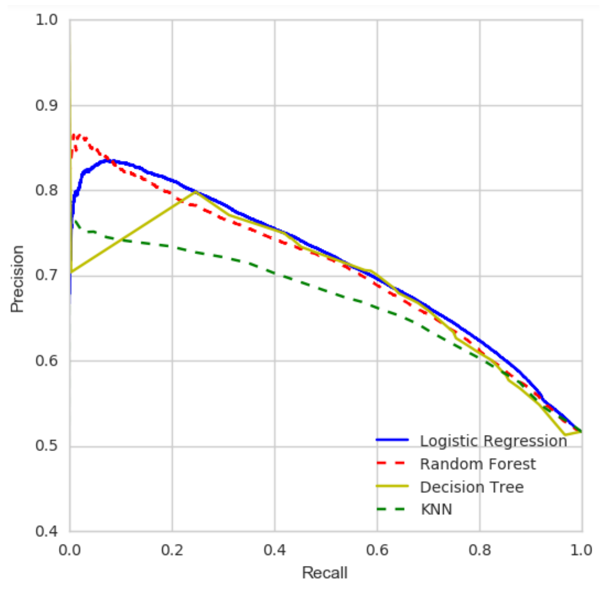

Figure 23. Precision-Recall curve comparison Diamond Dash Dataset t-1

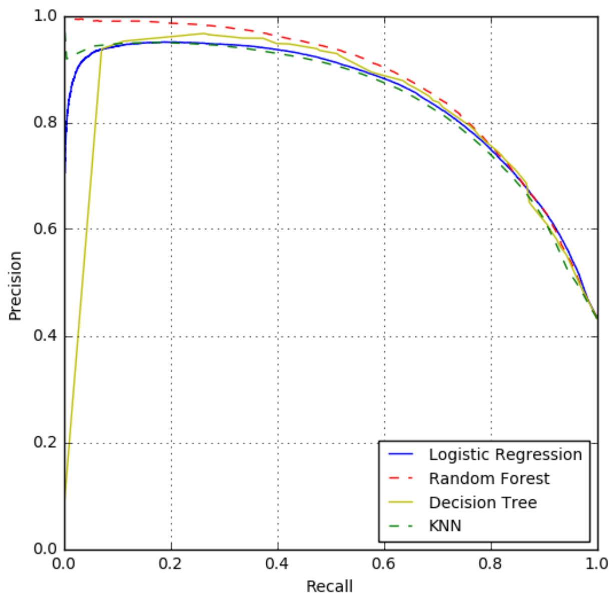

Figure 24. Precision-Recall curve comparison Jelly Splash Dataset t-7

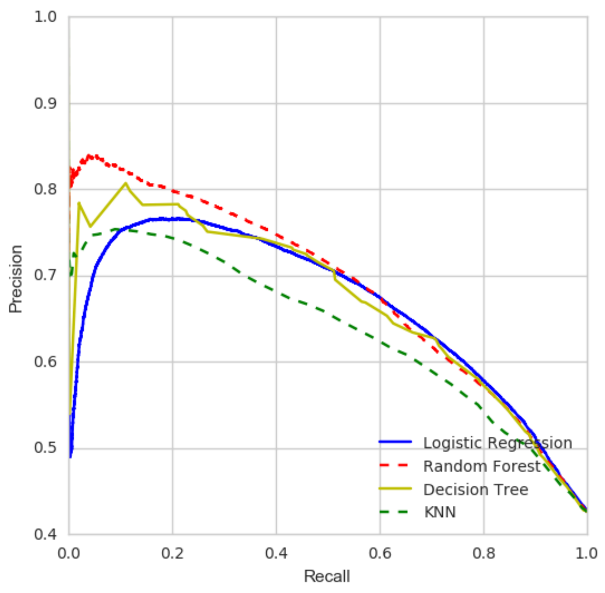

Figure 25. Precision-Recall curve comparison Jelly Splash Dataset t-7

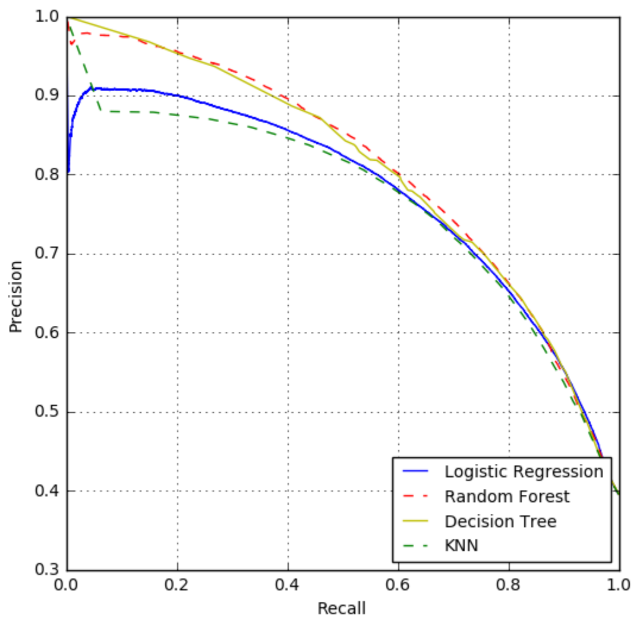

Figure 26. Precision-Recall curve comparison Pearl’s Peril Dataset t-7

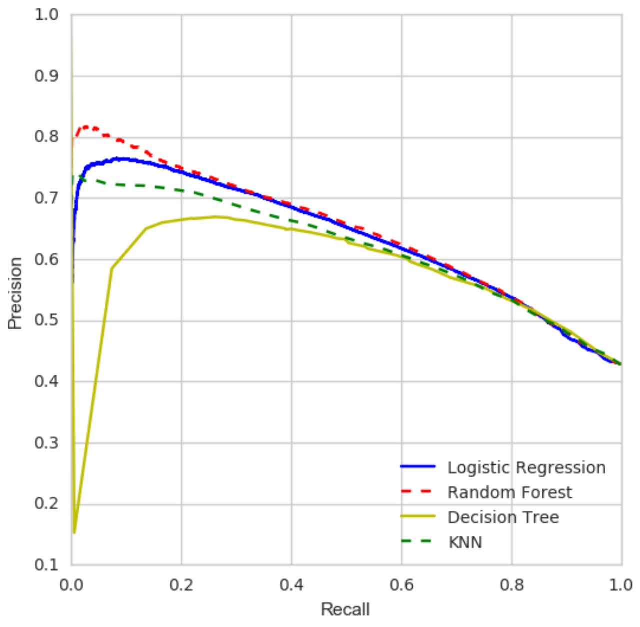

Figure 27. Precision-Recall curve comparison Pearl’s Peril Dataset t-1

Appendix D: Executive summary

This thesis provides an analysis and evaluation of a churn-prediction model using game-design-independent features across mobile casual games.

The datasets provided by Wooga contain data on three popular casual mobile games with different game-designs. The experiments conducted, tested the prediction performance differences between k-nearest neighbours, decision tree, random forest and logistic regression. The models were evaluated on classification accuracy, F1-scores and area under the ROC & PR curve.

The results show that predicting churn across games, using game-design-independent features, results in decent prediction performance using a 7-day feature window.

- The results indicate random forest as overall best classifier for predicting churn across games.
- The evaluation of the testing performance, conducted on out-of-sample data, shows excellent generalization of the model.
- The experiments indicate that *current absence time* is most important feature for predicting churn across games, which is in accordance with the literature.

This thesis also investigated the prediction performance of a single-day feature window. These results indicate substantial lower performance with only a minor improvement compared to the majority baseline. One limitation of the current study is that the data only contained owners of IOS devices.

The models described, clear the path for game developers to predict player churn in the early stage of game development. A game that has recently been launched, which has not generated enough data itself to train a machine-learning model, can use this model to predict churning players.