ELECTRONIC GAMING MACHINE PLAYSTYLE DETECTION AND RAPID PLAYSTYLE CLASSIFICATION USING MULTIVARIATE CONVOLUTIONAL LSTM NEURAL NETWORK ARCHITECTURE

by

Soheil Latifi

Submitted in partial fulfillment of the requirements
for the degree of Master of Computer Science

at

Dalhousie University
Halifax, Nova Scotia
August 2021

I dedicate this to my family, who helped me get where I am today.

Table of Contents

List of Tables

List of Figures

List of Abbreviations Used

Acronyms

ANOVA Analysis of Variance

COTE Collective of Transformation-based Ensembles

DBSCAN Density-Based Spatial Clustering of Applications with Noise

DNN Deep Neural Network

DTC Decision Tree Classifier

DTW Dynamic Time Wrapping

EGM Electronic Gaming Machine

EMA Epectation Maximization

HC Hierarchical Clustering

HIVE-COTE Hierarchical Voting Collective of Transformation-based Ensembles

HSD Honestly Significant Difference

KNN K Nearest Neighbors

LSTM Long Short Term Memory

MCLSTM Multivariate Convolutional Long Short Term Memory

ML Machine Learning

RELU Rectified Linear Unit

RFT Random Forest Tree

RNN Recurrent Neural Network

Functions

$\lambda(p)$ Carmichael's function

$\max(x, y)$ Maximum between x and y

σ Standard Deviation

$\sqrt{x}$ Square Root

$\sum x$ Summation

$E(x)$ Entropy of x

Abstract

Electronic Gaming Machines (EGM) are common, anonymous, stateless gambling machines operated by a region's lottery and situated in licensed venues. Previous work have shown that problem gambling detection is possible using EGM data, however, real-time customer personae identification might be even more important for stopping problem gamblers or suspicious playing behaviors. The following clustering algorithms were used and analyzed for the task of identifying different behaviours and personae types based on play style: CURE, DBSCAN, K-means with random initialization, TK-means++ (TKMPP), BIRCH, EMA, OPTICS and BANG. The results show that K-means with random initialization is the most suitable method for this task since it can scale well for the immense player data, it is efficient enough for multiple tests and analysis, and, most importantly, the results by K-means (clusters) are interpretable and meaningful. The experiments indicate that DBSCAN can be used before K-means to refine the results as it can identify results that cannot be identified by K-means. Inferred personae are used as labels for the playing sessions, and this data is used to train classifiers for playstyle detection. We identified methods suitable for real-time customer analysis, and the minimal number of initial transactions needed to successfully conduct this task. The classic classification methods such as perceptron, decision trees, and random forests are compared to deep learning based methods, showing that the best performance is obtained with a Multivariate Convolutional LSTM neural network. An important results is that increasing the number of analyzed transactions to more than 40 does not result in a large increase in performance.

Acknowledgements

I would like to thank my family and friends Shahriar Enjelasi, Mammad Kakhal, Dr. Hussein, Haj Masood, Pouya Jemali and Khashi who always supported me through life. I also want to thank my colleagues and supervisor Dr. Keselj that heart warmingly accompanied me through my masters program.

Chapter 1

Introduction

Electronic Gaming Machines (EGMs) and their other variants such as Electronic Gaming Machines (EGMs) are the dominant games of chance present in casinos and other venues that offer such services. Like any game of chance, the use of EGMs can be entertaining, but can also facilitate undesirable behaviour, particularly among people who exhibit problem gambling. It is desirable to conduct customer analysis that can help EGM stakeholders understand users' behaviour. Customer analysis can take many forms and variations. In some cases, it can be a straightforward platform that asks for the customers' feedback like Amazon's customers' review system [1]. However, sometimes stakeholders look for latent traits, such as the psychological characteristics that predict customer loyalty, their interests, and other character traits that could predict customer preferences. In case of EGM machines, stakeholders and researchers are interested in behavioural patterns of the players and understanding their playstyles for many applications such as predicting warning signs of problematic behaviour, and providing interventions when necessary. The primary goal of this research is to investigate if it is possible to determine the player's personae in the early stages of EGM use, and if possible, determine how many transactions are needed to identify the personae. It is obvious that greater numbers of transactions can lead to improved classification predictions, however it is also desirable to detect personae with as few transactions as possible. However, EGM machines are stateless, meaning that they do not record anything but a list of transactions and the time they have been committed. Sessionizing the data is done with assumptions on how commonly a playing session starts and how it might end. With the playing sessions available, the K-means algorithm, an unsupervised learning technique that was previously used in similar projects, was applied to the playing sessions to detect the playstyles. Our work is not focused on detecting problem gambling, but instead on detecting behavioral patterns and playing personae. The personae are identified using unsupervised

learning, unlike supervised learning models, training is not done with the ground truth present, but the data points are grouped together based on how similar they are together. This similarity is usually obtained using distancing measure. Given a number of existing unsupervised learning methods, our first task is to choose the best clustering model for identifying meaningful groups and understanding limits and pitfalls of the clustering algorithms.

The first challenge of this research is to sessionize the gaming data. The only information provided in EGM logs is a list of tuples that explain the transactions placed on each machine ordered by time. Sessionizing refers to grouping set of transactions that are assumed to belong to a particular player. For doing this our hands are tight in terms of using statistical methods for identifying the sessions, therefore we would use a set of assumptions about sessions and identify them accordingly. Naturally, a gaming session starts with a cash-in and ends with a cash-out or the machine being left when it is empty for some period of time. Using these assumptions we were able to sessionize the data. The second task is choosing suitable clustering models for identifying sessions and testing its limitation. From each class of clustering algorithms, a method is used. From partition-based clustering algorithms, K-means and TKMPP, from grid-based clustering algorithms, BANG, from hierarchical clustering algorithms, CURE and BIRCH, from density-based algorithms, OPTICS and DBSCAN, and Gaussian mixture model from expectation maximization algorithms. Each algorithm is tested on both datasets. Tuning hyperparameters is a little bit trickier for unsupervised learning compared to supervised learning. Supervised learning is blessed with metrics and measures that reflect exactly how the model is performing. On the contrary, unsupervised learning metrics abstractly indicate how good are the clusters, and the question of whether the clusters are meaningful or not is yet to be investigated. Additionally, these measures sometimes show consistently better scores for some algorithms, although direct inspection of the clusters does not show better clusters for our application. Therefore, for tuning hyperparameters using multiple evaluation metrics is suggested. A configuration that yields better evaluation scores among most measures is usually chosen for further analysis. If that configuration also results in meaningful clusters it is chosen, and if not the next configuration is chosen.

Our results show that combination of DBSCAN and K-means achieves the best

results. K-means identified good playstyles on the original dataset. DBSCAN on the other hand identified good playstyles on the sample dataset but failed to scale to the original dataset. One of the clusters found by DBSCAN could not be associated with a playstyle. This cluster contains majority of transactions and obviously should be analyzide further. Therefore, it was used as a dataset for K-means algorithm. The result are less noisier version of K-means results on the original dataset, DBSCAN works as a preprocessing technique for K-means.

In this overview of our main findings, DBSCAN and K-means are better choices for the task since plenty of data is needed and efficiency plays a major rule. For identifying even more playstyles trying different combination of features and introducing new ones is generally suggested. Our work proves that using DBSCAN and K-means can make problem gambling and in general, playstyle detection easier with better results that experience absence of noise. In our second project, assessing playstyles of gamblers at playtime, labeling the gaming sessions is necessary as our dataset is not labeled. In this task, we only used K-means algorithm, despite the better performance of DBSCAN and K-means together. This combination of methods can only scale up to a few thousand records which is not enough for training deep neural network models. Using a set of statistical features extracted from the playing sessions, the K-means algorithm is used to detect the play-styles. Two sets of possible models are then proposed which may be able to use a limited number of transactions starting from the beginning of the playing sessions to determine the play-styles. The first set includes classic models such as random forest tree and perceptron, while the second set includes a multivariate LSTM convolutional neural network with an embedding layer. These models are trained with different types of data. Classic models are trained with statistical features extracted from the transactions while the neural network is trained with time series that describe a feature related to these transactions. It is believed that the classic models may face some limitations and lack flexibility to capture the elaborate patterns of this data while they are more straight forward and need less computational power and time to fit the data. The results show that around 40 transactions are needed to assess the play-style associated with the playing session, and that the Neural network model outperforms the classic models. Our proposed methodology has shown great potential for commercial use. Our neural network does

not require any explicit feature extraction and requires minimal inference time combined with the great performance, this methodology can be used for many practical purposes like playtime game recommendation, problem gambling detection and fraud detection.

1.1 Motivation

The gaming industry, in the sense of legalized gambling industry, is a popular and large industry, which is also sometimes associated with negative social implications such as problem gambling. Although it is heavily monitored and regulated by the governments in different ways, core differences between concerns of players, business owners and the government have caused dilemmas that make providing solutions for current menaces involving gambling almost impossible. Players want to remain as anonymous as possible when playing and want data privacy. On the other hand, operators expect a quality product that meets their customers needs that usually needs customer analysis that is highly data dependent. Many of the functionalities that can be added to EGMs rely on data and customer investigation for maximum efficiency. At the same time, Governments create legislation and laws designed to balance player expectations and sustainable industry practices. Our main goal is to develop a data driven solution powered by machine learning to answer some of these issues. Previous research shows abnormal play behavior can be detected through data analysis. [2, 3, 4, 5], meaning that the EGM data reflects some of the behavior traits that is sufficient to detect problem gambling. Our first stage of project is focused on identifying the best unsupervised learning method for detecting the behavior traits, a.k.a. playstyle, from the EGM data. Yet there is another question to answer, is it possible to detect playstyles from the EGM data? The playstyles should be witnessed in the venues played by real players, not a random grouping of playing sessions that are statistically similar. Identifying the correct playstyles will help with identifying different types of customers, opening new possibilities of business analysis. Machine learning methods are about adapting abstract statistical models to produce a desired outcome as accurate as possible.

Machine learning methods normally need substantial amounts of training data, and thanks to the popularity of EGM machines, this is not a big issue. On the other

hand, machine learning models also rely on clean data. This is proven to be an issue as the EGM data is not IDed or labeled, therefore, cleaning up the data is a must. Since the data is not labeled, we focused on using unsupervised learning to group playing sessions that are similar, together. If these groups represent a meaningful pattern that can be mapped to real data, then it can be assumed that the clustering algorithm can detect playstyles. With the help of unsupervised learning, we can label the data based on the groupings. The labeled data can be used to train a supervised learning model that uses the labeled data to train models capable of detecting players at play time. Identifying the playstyle at play time can be used to detect negative behavior such as problem gambling at the playtime. This is proven to bear great importance since the model can be used as a core part of a game recommender or fraud detection system. To make sure that the best model is found for playstyle classification at playtime we used different algorithms to find the best one. The results of this project will help with building advanced game recommender systems that can help with the EGM business. It can also come in handy in terms of problem gambling detection at playtime. Overall, all the potential social and economical benefits of the project gave us a motivational boost to deliver the work.

1.2 Contributions

This thesis makes three significant contributions to the field of behavioral analysis and data mining.

1. Investigating various unsupervised learning methods to find the best one suited for identifying playstyles.

2. Finding an optimal machine learning model for detecting the playstyle at playtime.

3. Finding the optimal number of transactions needed for detecting the playstyle at playtime.

There exist different classes of unsupervised learning algorithms. Each of them have their own pros and cons but finding the best one for our case needs actual experiments not just making decisions based on previous performances. The main classes

of unsupervised learning algorithms used are hierarchical, grid-based, centroid-based and density based. From these classes of the algorithms, we have chosen at least one algorithm that is the most suitable for our task. Previous research conducted by Mosquera and Keselj [5] and Adami et al [4] has shown that K-means algorithm with random initialization can be used to detect problem gamblers. Our experiments show that a combination of DBSCAN algorithm and randomly initialized k-means can work better than other algorithms. Although using DBSCAN before k-means results in a much cleaner result, we will be bounded by efficiency of the DBSCAN algorithm. This phenomenon results in using a smaller dataset, but cleaner results. For classifying the sessions, we used decision trees, perceptron, and random forest classifiers as the baseline models, and compared them to our neural network model, Multivariate convolutional LSTM model that surpasses all the mentioned models and achieves better F1-score.

As for numbers of transactions for detecting the playstyle, it can be said that with more transactions, a higher performance is expected, but at the same time detecting a playstyle is crucial at the early stages of playing so if an possible negative behavior is found, it is detected as soon as possible. The results show that the models' performance does not increase drastically after 40 transactions therefore it seems using 40 transactions is optimal for detecting the playstyle of the player.

1.3 Organization of the Thesis

The remainder of the thesis is organized as follows:

Chapter 2 delves into the background knowledge needed to understand the theses. First, we explain notions in unsupervised learning and introduce different techniques for that. Then we discuss different measures for evaluating unsupervised learning and tuning hyperparameters for each model. Then we discuss possible candidate machine learning models for identifying play-styles at playtime.

Chapter 3 discusses our approach on cleaning up EGM data and preparing unsupervised learning algorithms for personae detection along with our approach towards testing and evaluating different unsupervised learning on the play data. Inspecting

their superiority over each other. Experiments on the data are described broadly along with additional information on the most basic steps of preprocessing of the data to fine tuning each clustering model. By the end of the chapter a full comparison of the algorithms is conducted along with discussion on the future work and results.

Chapter 4 explains how we used unsupervised learning, mainly K-means algorithm to transform the unlabeled data to a labeled data that can be used for training sophisticated models that classify the players based on the assigned labels at playtime. In this chapter a complete comparison between the algorithms is shown along with analysis of results and possible future works and techniques.

Chapter 5 In this chapter conclusions are made based on results of experiments conducted for each of the projects. Future work is also mentioned with focus on practical potential and theoretical advances of Machine learning and data mining and business needs and interests.

Chapter 2

Background and Related Work

In this chapter we explain the keywords and concepts that are important to know before we can propose our methodology. First, we briefly explain the importance of research aimed at analysis of playing data. Then we jump into unsupervised learning and how and why it is applied to EGM data. We elaborate on measures and evaluation techniques used to compare performance of unsupervised learning algorithms and how we utilized them to determine the best algorithm for our data. Next, we briefly go through the concept of machine learning time series classification and candidate models.

2.1 Behavioral Analysis Research Aimed at Play Data

Research focused on problem gambling behavior based on the data obtained from casinos have the most overlap with this study. Clustering of game play data has shown to be a promising technique to identify play styles and identify potential problem gamblers. Problem gambling is a behavioral condition when the gamblers cannot maintain themselves from excessive gambling. This is often accompanied by other harmful behaviors such as eating disorders, substance abuse [6], etc. Electronic gambling machines are highly used by problem gamblers. They are stateless and only record minimal amount of the play data. Although there are venues that have introduced a new feature called loyalty cards [7]; that is used for recording costumers' data in the casino. This results in a well-formed data that includes playing sessions, games played, and money spent. This not only completely discards sessionizing task completely but also provides a history of customer play data. Using royal cards is not mandatory and it is not even implemented by most venues [8] so classic data handling is still widely used.

White et al. [9] did remarkable research in 2006 based on a questionnaire answered by problem gamblers, researchers in the field of problem gambling detection, gaming,

problem gambling specialists, problem gambling counselors and problem gamblers themselves. It is worthy to mention this research clarifies what kind of EGMs' characteristics are most intriguing to problem gamblers. It is found that even venues' and games' features played major role in attracting problem gamblers. An online survey conducted by Wood and Griffith's [10] shows that a common responsible gambling strategy obtained by individuals with gambling history is spending limit. Furthermore, Schwarz Bayesian Criterion was used for clustering survey participants. Finally, two clusters named "casual dreamers" and "thrill-seekers" were identified.

Adami and Benini [4] proposed indicators based on wager fluctuation throughout time and number of different games played on the website, resulting in the identification of a new class of players who likely developed a medium risk of disordered gambling behavior that was not identified by Shaffer [2]. It is worth mentioning that Adami used K-means algorithm for clustering, getting best results with K variable equal to 5. Even though K-means algorithm is sensitive to outliers, it can be resolved by replacing extreme values with their nearest neighbors. This method is called "winsorization" [11].

Mosquera and Keselj's work [5] focuses on EGM data. Since EGM logs contain gaming events, session detection is also required. Instead of an idle time-based sectioning method like Liu and Keselj's [12], an event-based method is used to identify each session. Each session is then mined and various aggregated data such as the number of bets, net loss, duration, bet per minute (intensity), vouchers and loss percentage are extracted. This research is of great importance to us for two reasons: first is identifying gaming sessions using EGMs' logs and second is adapting previously existing criteria like Shaffer's [3] to cluster EGMs' data. Mosquera also used the k-means algorithm to identify clusters, and for comparison between different clusters tests like ANOVA and Tukey's Honestly Significant Difference (HSD) were done.

2.2 Unsupervised Learning

Unsupervised learning is the task of assembling a model that groups the data points that are most like each other. These groupings are expected to represent a logical pattern. This pattern might be detected due statistical closeness or an explainable phenomenon. Unlike supervised learning, Unsupervised learning algorithms do not

require tagged data. Unsupervised learning algorithms use different measures to group data points based on which they are divided to different subclasses. In a more formal manner unsupervised learning can be described as Partitioning data into a certain number of groups which bear high internal homogeneity and high external separation from other groups [13, 14, 15]. Both similarity and the dissimilarity should be examinable in a clear meaningful way.

As mentioned, playing data are not usually labeled so researchers cannot use supervised learning to label the data. Also using focus groups, surveys and tests is expensive and sometimes even impossible. Some players might not take part in such monitored acts because they might be concerned about their privacy. To overcome this issue, researchers use unsupervised learning to group play data that allegedly show a particular behavior. To make sure that resulting clusters are problem gambling behavior, they extracted variables that are problem gambling indicators. For example, Keselj and Mosquera [5] used variables such as bet per minute, vouchers, loss percentage, net loss and duration that can indicate problem gambling. These features were used to train a K-means model to cluster the players and hopefully identifying the problem gamblers. Their approach towards identifying playstyles might come off as intuitive but Braverman and Schafer [3] extracted 23 variables that were later used with statistical analysis to detect problem gamblers previously tagged by a gambling website. The results show play data contains enough information for identifying the problem gamblers. Using K-means with randomly initialized centers is common practice; for example, Keselj and Mosquera [5], and Adami *et al.* [4] used this method to find different types of players. Our main goal for the first project was to use different types of unsupervised learning algorithms on our data to see if there are better candidates for extracting different or better playstyles.

2.2.1 Procedure of Clustering Analysis

The clustering procedure is not straight forward or agreed upon, but we tried to take a route that is at least scientifically accepted and practiced before [16].

1. **Feature Extraction:** as pointed out by numerous articles [17, 15, 18], feature extraction means choosing a group of features from a set of candidates. Feature extraction utilizes some transformation techniques to generate handful features

from raw ones. This task is essential since it affects clustering task immensely. Perfect features help with distinguishing patterns belonging to different clusters, immune to noise, easy to extract and interpret.

2. **Clustering Algorithm Design or Selection:** No clustering algorithm can give superior performance for all data and all applications—each algorithm has its own advantages and shortcomings. To address this issue, we should select a handful of criteria and proximity measures that can be used to identifying the most suitable model.

3. **Cluster Validation:** Each algorithm can divide a data set to different clusters. To provide some confidence in the results, we should use metrics that are likely unbiased towards all algorithms this process will result in finding the best clusters, best number of divisions and the most meaningful clusters. There are 3 different criteria usually used for evaluating the clusters: external indices, internal indices, and relative indices. A brief explanation of each testing criteria [19, 15, 20] is given below:

   (a) External indices are based on predefined structure, which is the reflection of prior assumptions on the data and used as a criterion to validate the clusters.

   (b) Internal tests, unlike external indices, are not dependent on prior knowledge and assumptions. On the contrary, they examine the clustering structure directly from the original data.

   (c) Relative criteria place the emphasis on the comparison of different clustering structures, in order to provide a reference, to decide which one may best reveal the characteristics of the objects.

4. **Result interpretation:** The ultimate goal of clustering is to provide clusters of similar meaningful playstyles; i.e., the typical playstyle behaviours that can be mapped into recognizable real-world player behaviour, based on insights from the original data.

Now that we have explained what is unsupervised learning, how it was used and

how it is commonly applied to problems, we now go through different sub-classes of clustering algorithms and how they are categorized.

2.2.2 Clustering Algorithms

Clustering algorithms can be divided into different sub classes based on different measures and criteria [21, 22, 14, 23, 15, 18, 24]. Most agreed upon frame is to classify clustering techniques as hierarchical clustering and partitional clustering, based on the properties of clusters generated [21, 18]. Hierarchical clustering groups data objects with a sequence of partitions, either from singleton clusters to a cluster including all individuals or vice versa, while partitional clustering directly divides data objects into some prespecified number of clusters without the hierarchical structure.

Hierarchical Clustering

Hierarchical clustering (HC) algorithms groups the data into a hierarchical structure that can be visualized using a tree like structure or a dendogram. HC algorithms fall into the category of agglomerative and divisive methods. Agglomerative clustering initializes N clusters with one data point followed by a series of merge operations. This operation is repeated until all objects are assigned to the same group. On the other hand Divisive clustering takes another route. At first, the entire data set is considered as a whole and then several split operations are applied to it until all clusters are set to belong to clusters with only one data point. For a cluster with N objects, there are $2^{N-1} - 1$ possible two-subset divisions, which is very expensive in computation [25]. Therefore, divisive clustering is not commonly used in practice. Most common distance methods for HC algorithms are single linkage [26] and complete linkage.

- **Single Linkage:** The distance between two closest objects in different clusters (AKA nearest neighbor)
- **Complete Linkage:** Computing the farthest distance of a pair of data points to define inter-cluster distance.

Both of these methodologies, single linkage and the complete linkage method, can be generalized by the recurrence formula proposed by Lance and Williams [27]. There

are also more complex versions of these algorithms, such as mean linkage and group average linkage. The common criticism for classical HC algorithms is that they lack robustness and are, hence, more naturally sensitive to the presence of noise and outliers in the dataset. They also tend to form spherical shapes and reversal phenomenon, in which the normal hierarchical structure is distorted. In recent years, with the requirement for handling large-scale data sets in data mining and other fields, many new HC techniques have appeared and greatly improved the clustering performance. Typical examples include CURE [28], ROCK [29], Chameleon [30], and BIRCH [31]. The two main characteristics of the BIRCH algorithm are its scalability in dealing with large datasets and robustness to outliners [31], and its computational complexity of $O(N)$, which is significantly more efficient compared to the counterparts named before.

Noticing the restriction of centroid-based HC, which is unable to identify arbitrary cluster shapes, Guha, Rastogi and Shim [28] developed an HC algorithm, called CURE, to explore more sophisticated cluster shapes. The most crucial strength of CURE lies in the usage of a set of well-scattered points to represent each cluster, which makes it possible to find richer cluster shapes other than hyperspheres and avoids both the chaining effect [21] of the minimum linkage method, and the tendency to favor clusters with similar sizes of centroid. CURE can also endure the sever effects of outliers. On the other hand, ROCK is developed to deal with categorical features, making it almost irrelevant to this project. Relative hierarchical clustering (RHC) is another category of HC algorithms that utilizes both the internal distance, distance between a pair of clusters which may be merged to yield a new cluster, and the external distance, distance from the two clusters to the rest. RHC uses the ratio of both distances to decide the proximities [32].

Partitional Clustering

Partitional clustering assigns a set of objects into K clusters with no hierarchical structure. Optimal K can be acquired through a brute force approach that is proven to be infeasible in practice. To overcome this issue, Heuristic algorithms have been taken in to use in order to approximate the optimal value for K. K-means algorithm is the best-known squared error-based clustering algorithm [33, 34]. The time

complexity of K-means is $O(NKd)$. Since K, number of clusters, and d, number of features, are usually insignificant compared to N, this algorithm can be considered linear computationally. Obviously it can scale well to large datasets, although there is almost no efficient way for identifying the initial partitions and the number of clusters K. The convergence centroids rely on the randomness of initial points. A general strategy for the problem is to run the algorithm many times with random initial partitions. An interesting technique, called ISODATA, developed by Ball and Hall [35], deals with the estimation of K. ISODATA can dynamically adjust the number of clusters by merging and splitting clusters according to some predefined thresholds (in this sense, the problem of identifying the initial number of clusters becomes that of K parameter (threshold) tweaking). The new is used as the expected number of clusters for the next iteration. The iteratively optimal procedure of K-means can not guarantee convergence to a global optimum. K-medoids is a variation of K-means algorithm that can deal with categorical data.

Density-Based Clustering

In density-based clustering [36], clusters are defined as dense areas that are farther from other data points. Data points in these sparse areas, that are required to separate clusters, are usually considered to be outliers and border points. There are two major approaches for density-based methods. The first approach pins density to a training data point and is reviewed in the sub-section Density-Based Connectivity. Technique density and connectivity both are measured in terms of local distribution of nearest neighbors in this clustering. So defined density-connectivity is a symmetric relation and all the points reachable from core objects can be factorized into maximal connected components serving as clusters. Representative algorithms include DBSCAN, GDBSCAN [37], OPTICS [38], and DBCLASD [39].

Grid-Based Algorithms

Grid-based clustering is clustering where the data space, and not the dataset, is quantized into limited cells which form the grid structure and perform clustering on the grids [40]. Grid based clustering maps the data points to finite numbers of grids. Grid based clustering has fast processing time that typically depends on the size of

the grid instead of the data. The grid-based methods use the single uniform grid mesh to partition the entire problem domain into cells. The data points that fall within a cell are represented by the cell using a set of statistical attributes from the objects. These algorithms have a fast processing time, because they go through the data set once to compute the statistical values for the grids and the performance of clustering depends only on the size of the grids, which is usually much less than the data objects. The grid-based clustering algorithms are STING, Wave Cluster, BANG and CLIQUE.

2.2.3 Mixture Densities-Based Clustering

In the probabilistic point of view, data points are believed to belong to several probability distributions. In other terms data points in different clusters are generated by different probability distributions. These can be derived from different types of density functions like Guassian distribution, or similar functions, but with different parameters. If the distributions are determined, finding the clusters of a given data set is equivalent to estimating the parameters of several generator models. Maximum likelihood (ML) estimation is an important statistical approach for parameter estimation [41] and it considers the best estimate as the one that maximizes the probability of generating all the observations. Unfortunately, since the solutions of the likelihood equations cannot be obtained analytically in most circumstances [42, 43], iteratively semi-optimal methods are needed to approximate the ML estimates. Among these methods, the expectation-maximization (EM) algorithm is the most popular [44]. The major disadvantages for EM algorithm are the sensitivity to the selection of initial parameters, the effect of a singular covariance matrix, the possibility of convergence to a local optimum, and the slow convergence rate [44]. Variants of EM for addressing these problems are discussed in research literature [43, 44].

2.2.4 Graph Theory-Based Clustering

The concepts of graph theory [45] make it very convenient to convert clustering problems to a graph problem. Nodes of a weighted graph correspond to data points in the pattern space and edges represent the proximities between each pair of data points. Since both single linkage and complete linkage are both present the problem can be

handled with HC algorithms. Single linkage clustering can be translated to finding maximaly connected subgraphs while complete linkage clustering can be translated to finding maximally complete subgraphs [15]. Graph theory can also be used for non hierarchical clustering. For example, CLICK is based on calculation of the minimum weight cut to form clusters [46]. Similarly, CAST considers a probabilistic model in designing a graph theory-based clustering algorithm [47].

2.2.5 Fuzzy Clustering

Algorithms mentioned till now are in the class of hard clustering algorithms meaning that they assign each data point to only one cluster. This does not hold for fuzzy clustering. In fuzzy clustering algorithms, a data point is assigned a membership value for each cluster. This value shows how similar the data point is to the other points in the same cluster, higher membership value shows a higher similarity [48]. This is particularly useful when the borderline points among clusters are ambiguous. Moreover, the memberships may help us discover more complex patterns among a given object and the disclosed clusters. Fuzzy C-means clustering (FCM) is on of the most popular fuzzy clustering algorithms. FCM suffers from the presence of noise and outliers and the difficulty to identify the initial partitions.

2.3 Clustering Candidates

Now that we have gone through each class of clustering algorithms, we proceed to briefly explain our candidate clustering algorithms. We mention their pros and cons, and briefly explain the intuition behind them and why we have chosen them.

2.3.1 K-means with Random Initialization

The K-means algorithm [49] is previously used in much similar research, due to its simplicity and good performance on big data. This algorithm generally has low computational and memory cost, making it suitable for vast amount of data. The algorithm is easy to understand: first, a number of random points are initialized as cluster centers and then points are assigned to the closest clusters. The centroids of these clusters are calculated by averaging coordinates and they are assigned as

new cluster centers. This process is repeated until the average distance from cluster centers is lower than a threshold.

Different variations of this algorithm exist; for example, K-means++ [50]. Instead of picking random points as cluster centers, only first cluster is picked randomly, then the next center is the farthest point from the first center, second center is the farthest point from both centers and so on.

This is implemented because if randomly picked centers are close to each other than the cluster results will not be satisfactory. But if the dataset has outliers then cluster centers are no longer samples from different dense points but can actually be outliers that again affects the results. Play data includes outliers that are results of abnormal behavior and cannot be discarded from the dataset. This being said, K-means worked well with similar data. The only main hyper-parameter that needs tuning for this algorithm is the number of clusters, which can be determined with a few experiments. This algorithm has a linear complexity and scales well to big data.

2.3.2 TK-means++

TKMEANS++ tries to improve weak points of KMEANS++ [51]. This algorithm does not select the farthest points. As mentioned in the previous section, KMEANS++ has disastrous results if dataset contains outliers. To prevent such thing TKMEANS++ discards points that are farther than a certain threshold. Then, the next center is chosen among remaining points, points that are farther have a bigger chance of getting selected. This feature can again impose similar problems to this algorithm. To fix this issue another hyper-parameter was introduced that gave this ability to the user to tune to what extent distance can increase the chance of becoming the next cluster center. So far this algorithm has three main hyperparameters two of which are continues meaning that are a bit hard to tune, which is the main issue of this algorithm. This algorithm is suitable for big data since it is very memory and computationally efficient. TK-means++ has 3 main hyper-parameters that need tuning, 2 of them determine the degree of freedom of choosing outliers as centers and one is asserting the number of clusters. Overally fine tuning TK-means++ needs more effort than K-means with random initialization.

2.3.3 DBSCAN

A density-based clustering algorithm [52] that finds clusters based on density measures. The algorithm works in the following manner:

- A point is a core point with at least *minpts* neighbors within the *eps* distance of it.
- A point directly reachable from a core point belongs to the same cluster.
- A point not reachable from other points is an outlier.

DBSCAN requires immense memory and has a power two computational complexity making this algorithm less scalable than k-means. It is advised [53] that minimum neighbors should be around two times more than number of features. A good feature of this algorithm is that there is no need to specify how many clusters exist in the data removing bias from the research. DBSCAN can find clusters with arbitrary shapes, making the results different from KMEANS. DBSCAN can also find and label the outliers making it less affected by the outliers.

This algorithm's main issue is its $O(n^2)$ memory complexity, making it really bad for big data. This is the main problem of the algorithm, although average time complexity is $O(n \log n)$, which is not very inefficient.

2.3.4 CURE

A hierarchical based clustering algorithm [28] can detect clusters with arbitrary shapes. This algorithm assigns each point to a cluster and puts all clusters in a priority queue. Then closest clusters are merged until desirable number of clusters are found. This algorithm has running time of $O(n^2 \log n)$ along with $O(n)$ memory cost. This algorithm is not computationally efficient, but it is somewhat robust to outliers. There is only one variable that determines the number of clusters making the algorithm easy to tune.

2.3.5 BIRCH

Another hierarchical based clustering algorithm [31], like CURE birch can detect clusters with arbitrary shapes. BIRCH is more efficient than CURE with computational

complexity of $O(n)$, but it has two main hyperparameters one of which is continuous and hard to tune. Compression rate and number of clusters determine how many clusters the model should detect. The smaller the compression rate the model is more likely to find desired clusters. Although if the compression rate is too small the clusters may not be well formed.

2.3.6 Gaussian Mixture Model

Coming from the family of expectation maximization clustering algorithms [54], this algorithm assumes that clusters are produced by the Gaussian models. A hyperparameter sets the number of Gaussian models, then points are randomly assigned to these models, based on these points' standard deviation, and means of these models are set. Every point is assigned to the most likely machine. Again, hyperparameters of the machines are computed and this process is repeated until a certain threshold is reached. If machine is left with no points, it is discarded. So, there is no guarantee that exact number of clusters are produced. This algorithm has computational complexity of $O(NKD^3)$, where D is the number of dimensions. So, this algorithm is somewhat efficient.

2.3.7 BANG

A clustering method from family of grid-based clustering [55]. The main advantage of grid-based algorithms is that they scale exceptionally well to big data. The computational complexity of these algorithms is dependent on the number of the partitions. BANG partitions the dimensional space of the variables to a hierarchical set of grid regions. Grids with highest density become the cluster centers. Next, they are merged with their neighbors to form bigger clusters. If the dimensional space is densely populated with points, then this algorithm will fail to distinguish clusters.

2.4 Time Series Classification (DTW)

Before discussing time series classification we first need to understand machine learning and time series thoroughly. Then we will explain how time series classification can be done using Deep Neural Networks (DNN) and non DNN methods.

2.4.1 Time Series

Time series is a set of data ordered by time. Time series show fluctuations a variable through time. Time series can be in very different forms, such as natural language data or stock market data. Although both natural language data and stock market data are time series but they are inherently disparate. Time series data are categorized in to two different groups based on the number of variables they have:

- Univariate Time Series: Time Series that have only one variable.
- Multivariate Time Series: Time Series including more than one variable.

2.4.2 Machine Learning

The technique of finding patterns in great amount of data using statistical model is called *machine learning*. Machine Learning is not a theoretical concept bounded to borders of academia but a thriving field that has already gotten in to commercial market. Today Machine Learning has evolved from doing simple classification on tabular data to generating human like text [56] to detecting complex objects in images [57]. Machine Learning is not restricted to image or language data, actually Machine Learning techniques are widely applied to time series data where the results are prominent [58].

2.4.3 Time Series Classification and Non-Deep-Learning Methods

Time series classification is one of the core focuses of machine learning [59, 58] like Image Processing, Speech Recognition and Natural Language Processing, but yet there are not that many models dedicated to this task. In many cases Time Series Classification is conducted using Dynamic Time Wrapping (DTW) combined with another classifier.

In most trivial cases, DTW is combined with K-nearest neighbour (KNN) method. DTW is a standard method used for measuring dissimilarity between 2 temporal sequences [60]. The main difference between Euclidean distance and DTW is that Euclidean distance is measured in a strict manner, no phase difference is taken into account while DTW, in exchange for some extra computation, computes the least difference by mapping the data points differently and applying a phase difference if

needed. KNN applies the most dominant label of K nearest neighboring points to the target data points.

DTW can be combined with more sophisticated models such as Support Vector Machines, Decision Trees, even Random Forests to achieve even better results. To achieve even better performance, DTW can be combined with a collection of classifiers. Collective Of Transformation-based Ensembles (COTE) uses 35 classifiers in for this task. An additional enhancement called hierarchical voting (HIVE-COTE) is also added to this design to achieve state of the art results [60]. Although HIVE-COTE shows great performance, it comes with extreme computational cost, after all training 35 classifiers should be taken seriously.

2.4.4 Time Series Classification and Deep Learning Methods

The concept of deep learning relies on designing minuscule learning units that are used together to mimic learning process done in human brain. At the beginning, training stacked layers of neurons was not efficient, but with recent innovations and introduction of powerful GPUs, attention has shifted from simple methods to neural networks. So far neural networks have revolutionized natural language processing and image processing. These milestones are achieved by introduction of Recurrent Neural Networks and Convolutional Neural Networks. In the next section we explain how different types of neural networks work and why they are better for answering different problems.

2.4.5 Artificial Neural Networks

Artificial Neural Networks use stacks of perceptrons to operate. A single Perceptron applies an activation function to a weighed sum of inputs and produces a result accordingly. Perceptrons process the input only once till they produce the target variable. There are many different activation functions that Perceptrons use like Rectified Linear Unit (RELU), Scaled Exponential Linear Units, Sigmoid, etc. Each of these activation functions have their own strengths and weaknesses and can be used accordingly. ANNs are used along other networks. ANNs can be applied to different types of problems, from tabular data to time series and images.

2.4.6 Convolutional Neural Networks

Convolutional Neural Networks (CNN) are first introduced for extracting features from image data [61]. Later on, with some modifications these networks' usage was expanded to other domains as well speech recognition [62], natural language processing [63], etc. Convolutional networks use a sliding window with trainable parameters that extract features from the input data. This approach is even used for time series classification by the architecture produced by Liu et al. [64].

2.4.7 Recurrent Neural Networks

Recurrent Neural Networks use a recursive construct to capture sequential information present in the data. This construct is widely used for conducting predictions and dealing with sequential data like voice data, time series data and natural language data. The recurrent structure allows feature extraction from temporal data but this comes with a cost called vanishing exploding gradient. This phenomena happens when extremely large or small gradients are multiplied in the process of training. This prolongs the process of training and usually happens when the sequence gets longer than the capacity of the recurrent network. A simple RNN network uses current input along with hidden state obtained from previous decisions. This network is highly prone to this issue.

2.4.8 Long Short Term Networks

Long short term memory units are developed to deal with vanishing exploding gradient problem [65]. An LSTM contains a cell, an input, an output gate and a forget gate. These gates work together to make the process of training smooth and optimized. Forget gate decides whether some information should be dropped or not. This is applied to the cell gate. The cell gate after that is updated by the hidden cell and the input gate. This process is then recursively done until the valuable information is extracted and kept. LSTM units in general work better than simple RNN units but still fall off if the input sequence is longer than their limits.

Chapter 3

Player Segmentation

3.1 Introduction

Analyzing and classification of EGM data is a challenging task that needs proper planning and immense focus on the details.The main intuition behind investigations targeted at EGM data is to get better insight on identifying problem gambling, playstyles and customer behavior in casinos without demanding more information and possibly invading privacy and breaking rules. The practical reason behind using clustering and data mining techniques on EGM data is not to mathematically prove that a playstyle exist or clusters are different but to see if player behavior can be segmented in a meaningful way with possibility of detection at playtime.

To be efficient with clustering and playstyle detection we have to understand the context, and target the playstyles that we want to detect and answer initial questions about alleged playstyles. What are the identifiers of such playstyles? What features can help with identifying these playstyles? To answer these questions we have to be precise and careful. In this section, we explain how we sessionized the data, how we conducted clustering, how the proper clustering method is used and how we trained machine learning models for classifying the player data.

3.2 Sessionization

EGM machines' logs contain transactions of all machines committed in a day. EGM logs consist of gaming data that are collected at the play time accompanied with additional meta data. These logs contain no user ID or any similar attribute that can be used for identifying users. These strict restrictions are imposed for respecting user's privacy, although these limitations add difficulty in the further analysis. The first thing that we had to overcome was to sessionize the data. In order to do so, we first have to define a gaming session. A gaming session is a list of transactions

that are assumed to belong to a player. With absence of player IDs this has to be conducted manually. First each log is grouped by the machine ID, ordered by time. The essential intuition is that a player only plays with one machine, although there are cases that some players play with two machines at the same time. We will not further look in to detecting these parallel sessions for the following reasons. First, we have no information on physical location of the machines. Second, using two machines at the same time is only possible in venues that do not require royalty cards for playing therefore it is reasonable to assume that this happens only rarely. Intuition of the sessionizing procedure is based on two strong assumptions. We first assumed that every gaming session starts with a cash-in. Although there are cases that a machine already has some credit inside it which results in cases that a player might start playing without cashing in at all. Obviously, our assumption discards such sessions. The second assumption is that a gaming session is terminated either by a cash-out that almost empties the machine or with player playing with almost all the credit in the machine and then leaving it in an idle state. We did not consider any threshold for the first assumption, but the second assumption need threshold values for minimum remaining cash and idle time threshold. With these two thresholds, termination of a session is declared when the time gap exceeds the idle time threshold while machine credit is less than minimum credit. Alternatively, if the cash-out leaves less money than the minimum money, session termination is declared. Our assumption allows multiple cash-outs with purpose of securing some wins to happen in a session. We computed the values for remaining cash and idle time experimentally by paying attention to how average idle time in sessions change by changing such values. If the idle time is increased to infinity the average idle time for each session is also increased so that a session might be as long as a few days. The same holds for minimum machine credit. Our experiments showed that a value around a few minutes for idle time and minim cash of approximately a couple of dollars cents result in sessions with logical lengths. Figure 3.1 shows how the sessionizing algorithm transforms the log data to sessions.

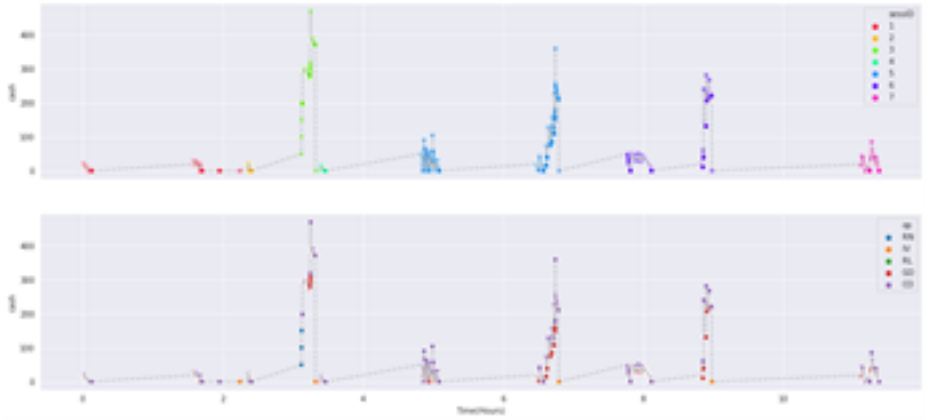

Figure 3.1: The current sessionizing method transforms raw data into gaming sessions

3.3 Feature Extraction

The next step is extracting features from these sessions. 30 initial features are extracted from each session. For each machine, a csv file is made with the same ID as that machine. Some of these features are explained in Table 3.3.

Feature	Explanation
MachineNum	Serial number dedicated to each EGM
Date	The exact date and time the session started
MeanRLpg	Average secondary games encountered per each game.
SumWinCo	Sum of primary wins
SumWinRL	Sum of secondary wins
SumInsertedCash	Sum of cash in money
SumPrimWager	Sum of primary wagers
NumBetsPrim	Number of primary bets
TimePlayedSec	Duration of session in seconds
SumCashOut	Sum of cash outs' money
NumGames	Number of times the primary game was changed
NumCashIn	Number of Cash-Ins
NumCashOut	Number of Cash-outs
STDSecWage	Standard deviation secondary wage

Table 3.1: List of features extracted for playstyle segmentation

Some of these features are the same as the ones Mosquera [5] found in her studies. The other ones are supposed to work as indicator for certain playstyles. For

example, in case of some players, high paced spins and large wagers were considered to be indicators of problem gambling. This acts as a prior to the experiments, as witnessed problem gamblers have shown such behavior, we can conclude that it can be detected by our methodology. Absence of ID accompanied with privacy issues makes it extremely hard if not impossible to validate our methodology. Thus we have to elaborate that our method groups players that are prone to the target behavior.

As mentioned, the features act like flags of such personae, meaning that if a certain playstyle exists then these indicator features will be used to identify them. Since this task is unsupervised, our main hypotheses is if an algorithm can identify playstyles. To answer this question the results of these algorithms should be examined carefully. If there are enough sessions in the same cluster with similar, extreme indicators then it can be concluded that the algorithm has successfully identified a playstyle. This part needs more clarification. Existence of enough sessions in a cluster is necessary unless, the generality of this playstyle can be questioned. Similar value means that sessions in the same cluster should be homogeneous meaning that the error rate should be relatively low for each variable unless the validity of the results may be at risk. Extremeness of the variables in each cluster is used as an indicator of a possible playstyle. A cluster with enough members and low error rate can be further investigated. If there is no variable with extreme value then the cluster cannot be interpreted. Therefore, 13 variables were selected for the final experiments. These features are shown in Table 3.2. These variables show possible playstyles, but it should be noted that the choice of variables is not limited to this list. Several different configurations can be tested to identify different playstyles.

Feature	Explanation
STDBetWage	Standard deviation of primary wagers
STDBetTime	Standard deviation of time Between two consecutive bets
MeanRLpg	Average of Secondary games faced per game.
TimePlayedSec	Duration of session in seconds.
NumGames	Number of times primary game was changed
SecPrimRate	Ratio of secondary wager mean over of primary wager mean
STDSecWage	Standard deviation of secondary wage

Table 3.2: The sample of selected variables selected for the final experiments

3.3.1 Dropping the Weak Sessions

This sessionizing method was applied on the raw data from 2016-01-02 to 2016-02-12. This resulted in 1,111,558 sessions that belongs to 5435 machines. These criteria seem to be solid but still the validity of this study relies on the validity of the sessions. Therefore, unreliable sessions are dropped. Unreliable sessions are results of unpredicted circumstances. IDLETIMEPREC is a feature that demonstrates how long the machine was in idle state throughout a session. A session that has been in an idle state for more than 50% of the time may not be valid and worthy of further analysis. 50% idle is not only hard to imagine in a real-life situation but has negative impact on other variables and damages their meaning. For example, bet per second does not reflect the betting pace anymore since there is a noticeable pause in the gaming sessions. Standard deviation of bet time is also affected, and will not show betting pace fluctuation anymore, because it contains pauses as well.

Another set of variables that are utilized to identify weak sessions are number of idle states, average machine credit during idle time and average idle times. We can easily imagine various situations where a player may interrupt the game and leave the machine with more than one dollar, so that the next player who uses this machine cashes-in and keeps playing. In this case our algorithm identifies only one session. To eliminate these sessions, all sessions with one idle period longer than four minutes with less than $1.20 is discarded. These safety measures result in discarding 397103 sessions. Note that it should not be concluded the rest of dataset is 100% valid, there can be even more measures to ensure the quality of the sessions.

3.4 Visualization of the Data

Visualization of the data bears great importance. It makes understanding the data easier and can even be helpful with finding a solution to various problems. Sometimes a plot can work just as good as a statistical method for ensuring validity of the results.

Histogram

A simple histogram shows the distribution of the data based on a variable. The maximum and the minimum value and the type of the distribution are common information

that can be taken from a histogram. It can also be used to detect outliers, especially global outliers. The histograms in Figure 3.2 depict how the data is distributed, it also gives us a better point of view on to what to expect from the data.

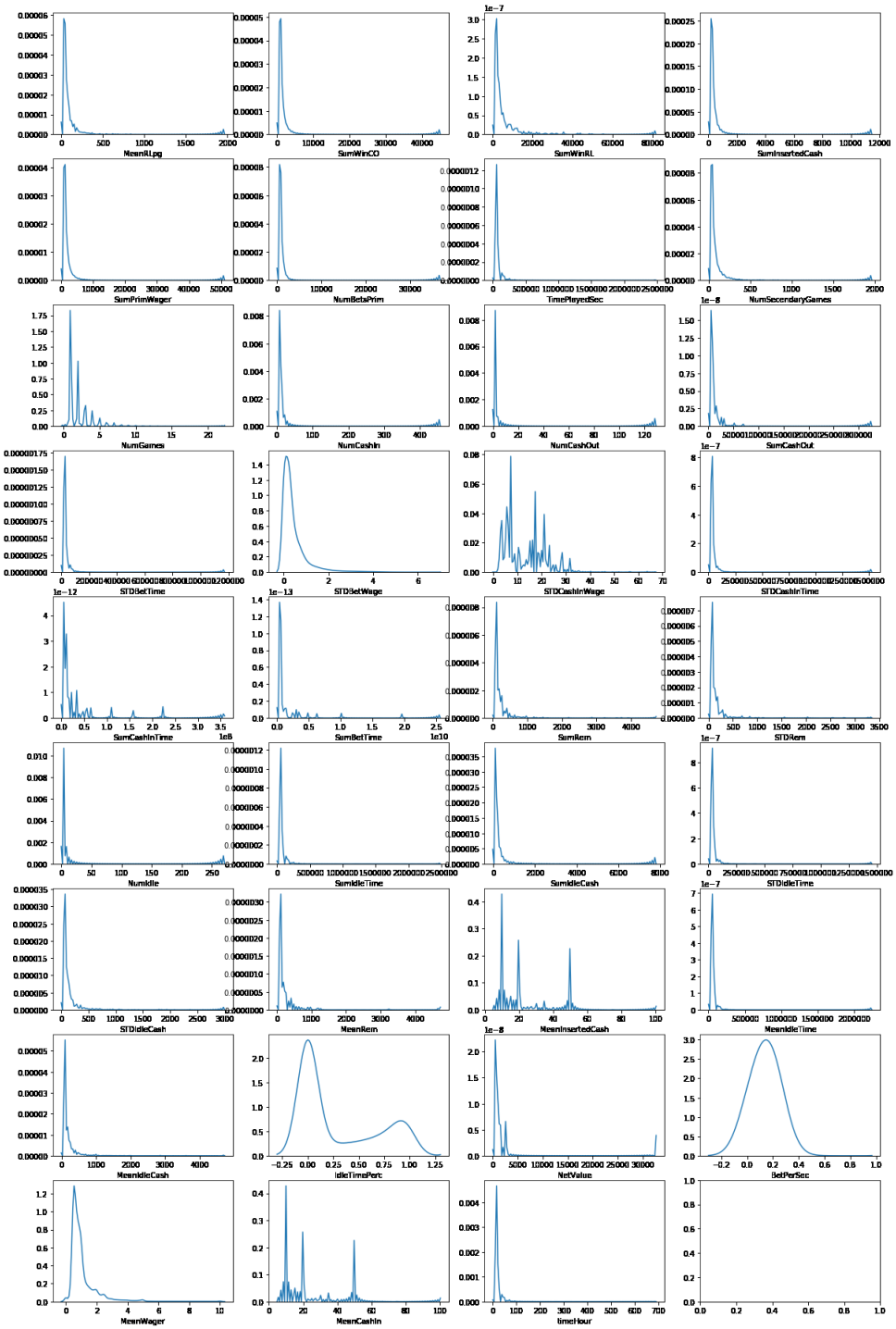

Figure 3.2: Histogram showing data distribution; used to help discard weak sessions

Box Plot

The Box Plot is better for finding outliers and understanding the distribution of the data. It gives additional information of the quantiles. As reflected in figure 3.3 the distribution of all features is somewhat skewed and the presence of outliers is prominent.

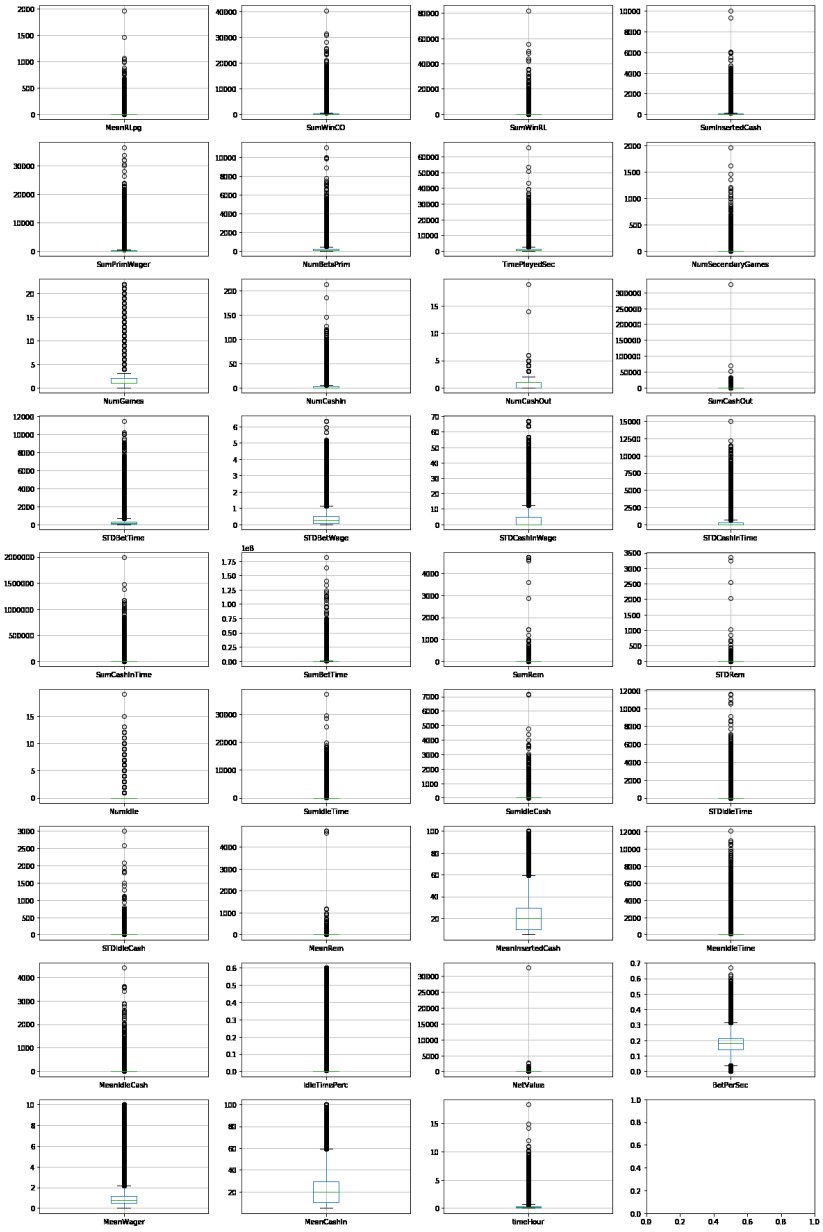

Figure 3.3: Clustering result illustrated

Scatter Plot

Another form of visualization is needed to discard contextual outliers. Outliers that are significantly different from data points present in the same context are called contextual outliers. With a minimal observation in figure 3.4 you can detect such anomalies and outliers. These outliers can be identified with scatter plot since it is hard to detect with respect to one variable. As an example, our analysis indicate that as the sessions get longer the playing speed expected to decrease but some anomalies were witnessed that showed extreme slow-paced betting further we found out these sessions have long idle pauses that made them useless for our research.

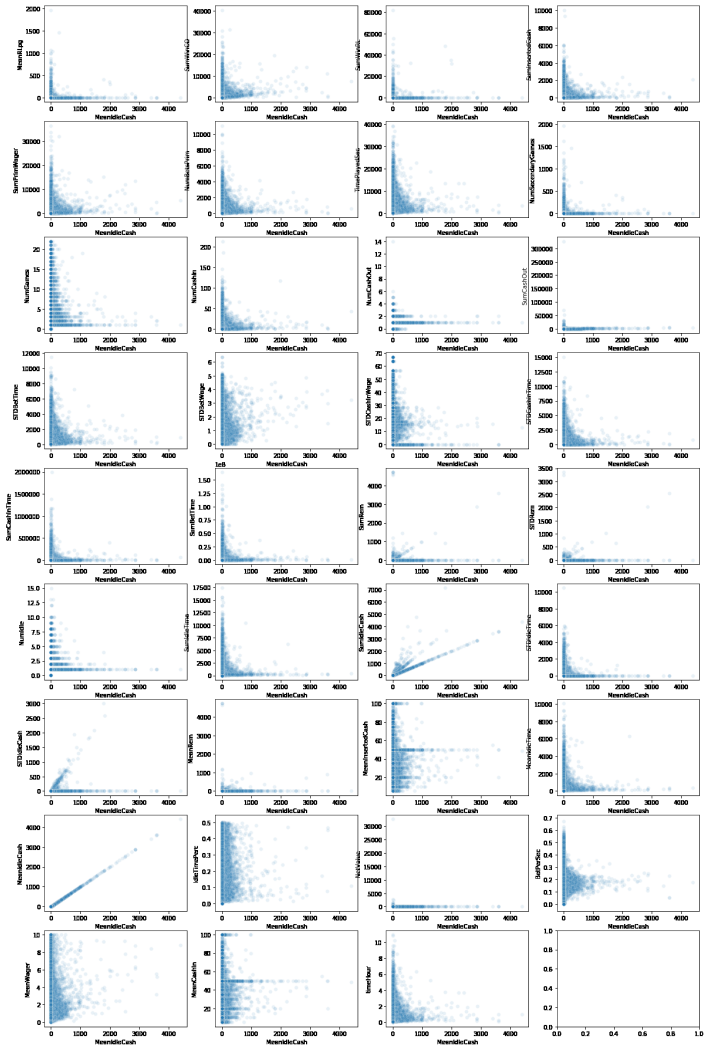

Figure 3.4: Data distribution in regards to average idle cash

Correlation Heatmap Plot

The correlation plot helps to understand variables relationship with each other. This helps to understand if the variables are linearly correlated. The data shows strong relationship between standard deviation of primary betting and play time duration.

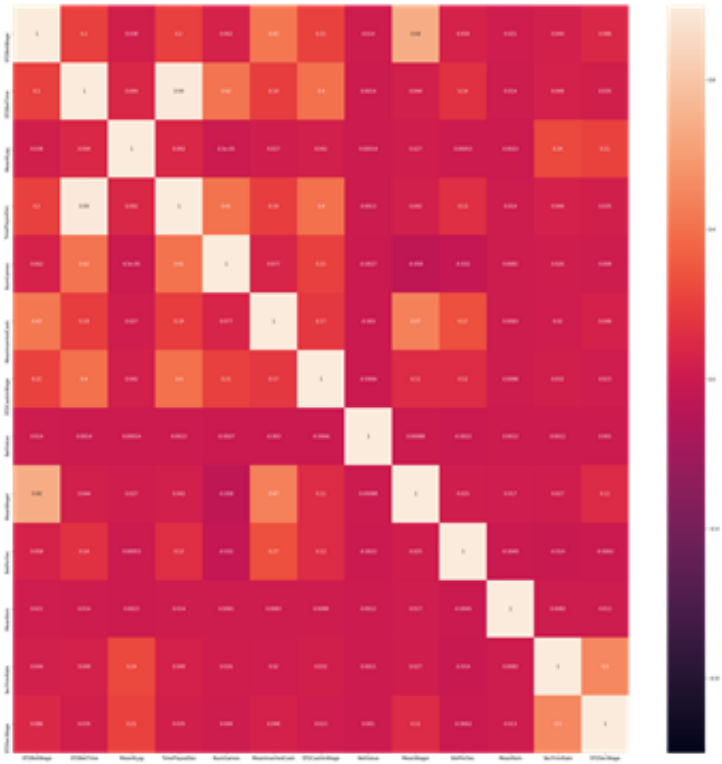

Figure 3.5: Clustering result illustrated as correlation heatmap

3.5 Normalization

Final step of preprocessing is normalization. Normalization is essential prior to clustering. Normalization scales the data to a range. It is necessary since usually variables are in different ranges, for example usually age has a bigger value than shoe size, if not normalized, clustering model prioritize age since it is greater in nature. It is explained in the next part how the normalizer is chosen.

Min-Max Method

This method scales the data to [0,1] range. This algorithm is highly affected by outliers. Outliers may compress the points and completely sabotage the distance between points. Making it a bad choice for this dataset since the presence of outliers is certain.

$$x_s = \frac{x - \min(x)}{\max(x) - \min(x)} \tag{3.1}$$

StandardScaler method

Standardize features by removing the mean and scaling to unit variance. Like Min-Max Normalization, skewed data can change the performance of this method and make it problematic.

$$x_s = \frac{x - \bar{x}}{\sigma(x)} \tag{3.2}$$

Normalizer Method

The normalizer scales each value by dividing each value by its magnitude in n-dimensional space for n number of features. The main advantage of this algorithm is that it does not alter the distribution of data. Making it worthy in this case.

$$x_n = \frac{x_i}{\sqrt{\sum_{i=1}^{n} x_i^2}} \tag{3.3}$$

3.6 Checking Scalability of the Clustering Algorithms

The original dataset is big; consequently, it is necessary to check scalability of the algorithms before using them on the original datase. Our main concern is that using the algorithms on the main dataset might cause the machine to crash or go on a never-ending execution. A never-ending run is not an empirical phenomena but rather a term we use to refer to any computation that lasts more than our expectation. In order to understand the scalability limits of each algorithm, chunks of datasets with different sizes are made. The smallest chunk has 50,000 data points, and we gradually increase the chunk's size 100,000 data points at a time until it reaches 850,000 data

points. Obviously most of these algorithms are not necessarily bounded by the size of the set but each hyper parameter can effectively increase or decrease the run time of the algorithm. A good example of such scenario is BANG algorithm that is only bounded by the number of grids or DBSCAN and OPTICS that require a spotless tuning. Note that OPTICS and DBSCAN should be initialized with good hyper parameters unless the algorithm might encounter a best-case scenario and assign all the data points to a single cluster with $O(N)$ complexity. Min distance was tuned on 700,000 and 150,000 data and the value was fixed on 0.0015. other algorithms where initialized to find 6 clusters. We anticipated that the not all algorithm can scale up to the original size of the data, one-month of play data containing more than 700,000 sessions. There are two possible workarounds, one is to only use k-means and similar computationally efficient methods on the dataset, which means discarding all other algorithms. The second approach is to come up with a form of sampling that can result in a secondary dataset that is smaller but still has enough sessions that models the original dataset with minimal data loss.

The second workaround is obviously a better logical choice. To do this, sessions are assigned to five minutes time periods, for example a session with 7 minutes play-time belongs to [5,10] bin, and so on. From each of these bins 20 % of the data were sampled. This is due to assumption that playstyles are distributed over time. Note that sessions are intact, and this sampling discards some sessions and does not summarize sessions.

At last, we ended up with 2 different datasets, the main dataset with 700,000 sessions, and the sample dataset with 140,000 sessions. The main idea is to test each algorithm on both datasets. But if one algorithm cannot scale then it is only tested on the sample dataset.

3.7 Evaluation of the Clusters

As mentioned in the previous sections, there is no absolute assumption about the outcomes and possible playstyles used, making this task an unsupervised learning task in from the machine learning perspective. It is mandatory to understand the limitations and challenges of unsupervised learning. First limitation unsupervised learning faces is lack of a solid evaluation metric. On the contrary, supervised learning

is blessed with different metrics like F1-measure or accuracy that precisely show how the model is performing. Also, the model focuses on optimizing these metrics. For unsupervised learning the evaluation metrics only show how good the clusters are separated and how similar cluster member are to each other, to this point this is not problematic, but if the clustering task is to identify some sort of behavior or pattern, then there is not an absolute correlation between the clustering results and actual existing patterns. Another consequence of this issue is that tuning the hyperparameters is extremely hard if not impossible. It is surprising that even if the metrics show a high score for a configuration it does not mean that result is going to show a meaningful pattern. In conclusion, the algorithms with less hyperparameters are preferred, also finding a good result should not be only limited to a good metric score but to human observation to confirm the results.

Before explaining how each algorithm works, general parameter tuning, and result evaluation schema is explained. If there is no way of hyper parameter tuning method for a certain algorithm, as a general rule each model is tested with a set of configurations, then the result with best score is analyzed with the hope of finding a good pattern. If no such thing exists another configuration is observed.

For this task 3 internal evaluation schemas are chosen. Davies-Bouldin index [66], Silhouette index [67], and Calinski-Harabasz [68] score. Note that the ground truth is not present in the data therefore evaluation metrics is restricted to the ones that do not require the ground truth. Ideally the configuration with best score in all 3 metrics is used. If no such situation happens then multiple configurations are examined and the one with the best meaningful clusters are chosen.

Davies-Bouldin Index

It is chosen when there is no ground truth present in the data. This index evaluates the cluster based on average closeness between clusters. The closeness measure is distance between clusters with the size of the clusters. Obviously zero is the best possible score. So lower Davies-Bouldin score indicates better clustering results. The index is defined as the average similarity between each cluster C_i for $i = 1, 2, \ldots, k$ and its more similar one C_j. The similarity is measured with R_{ij}. If we denonte variables

- s_i, is the average distance between each point of cluster i from the center of that cluster,
- $d_{i,j}$, is the distance between cluster centroids i and j,

then $R_{i,j}$ is calculated in this manner:

$$R_{i,j} = \frac{s_j + s_i}{d_{ij}} \tag{3.4}$$

and the DB score is:

$$DB = \frac{1}{k} \sum_{i=1}^{N} \max(R_{i,j}) \tag{3.5}$$

The DB score pros are:

- It is fast to compute.
- The index is computed only using inherent quantities and features.

The DB score cons are:

- DB score favors convex clusters over other concepts of clusters therefore K-means results usually get higher score.
- The usage of centroid distance limits the distance metric to the Euclidean space.

Silhouette index

If ground truth is not provided with the dataset this score can be used to assess goodness of the clustering method. The higher silhouette scores the better. The score ranges between -1 to $+1$ where a score close to $+1$ means good clustering while -1 means the opposite. And, a score close to 0 means that clusters have overlapping.

The Silhouette score is composed of two scores:

- The mean distance between a sample and all other points in the same cluster.
- The mean distance between a sample and all other points in the next nearest cluster.

The Silhouette score coefficient s for a single sample is computed in the following manner:

$$s = \frac{b - a}{\max(a, b)} \tag{3.6}$$

Silhouette score pros:

- The score is bounded and easy to interpret.
- The higher score indicates clusters are dense and well separated. Which relates to standard intuition of a good cluster.

Silhouette score cons:

- Silhouette score is higher in general for convex clusters therefore K-means results usually have a higher score.

.

Calinski-Harabasz Index

Again, if the ground truth is not provided this method is often a good choice. Also known as the variance ratio criterion, it can be used to evaluate clustering results. The index is the ratio of the sum of between-clusters dispersion and of inter-cluster dispersion for all clusters. Therefore, the higher Calinski-Harabasz score the better the clustering result.

For a dataset e with size N_e which is clustered into k clusters the Calinski-Harabasz score s is defined as the ratio of the between clusters dispersion mean and the within cluster dispersion:

$$S = \frac{tr(B_k)}{tr(W_k)} \times \frac{(N_e)}{k - 1} \tag{3.7}$$

Where tr(B_k) is trace the between group dispersion and tr(W_k) is the trace of the within-cluster dispersion matrix.

Calinski-Harabasz score pros are:

- The score is higher when clusters are dense and well separated that matches the common assumption about a good cluster.
- The score is fast to compute.

Calinski-Harabasz score cons are:

- Like other evaluation metrics this score is usually higher for convex clusters.

3.7.1 Final Notes on Evaluation Metrics

These evaluation metrics are good measures of assessing goodness of a good clustering results. They can also be used for comparing results of the same clustering methods but if the algorithm is changed then these methods are not good comparison criteria since they favor some algorithms over the other. Also, as it should be noted that high evaluation score does not indicate that a good pattern is observed. Therefore, no bias should be taken if the score is high or low since the final decision is made by human operator. Figure 3.6 shows how the scores help with choosing the right parameter.

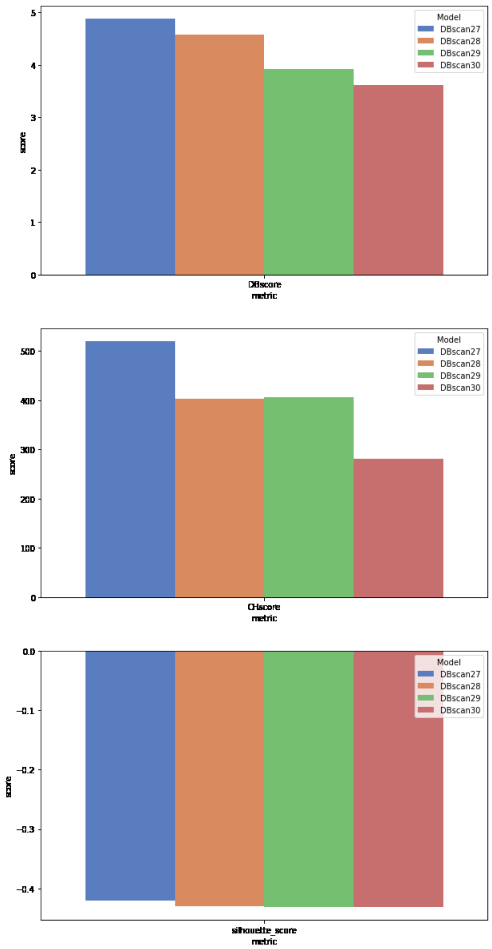

Figure 3.6: Overview of Clustering results

3.8 scalability Results

The following chart shows how long each of the algorithms need to find clusters for each chunk of the data. BANG is not included in the results since BANG is not bounded by the size of the data but by the number of grids. OPTICS is halted on 350,000 chunk as it fails to scale same holds for cure on 450,000 chunk. We face a different issue with DBSCAN as the computation fails to complete since it is memory intensive. As the results show BIRCH and K-means have the best performance time-wise, while CURE and OPTICS are much slower as they require almost one day to cluster 250,000 and 350,000 sessions. Although it might seem that 24 hours is not necessarily a unaffordable amount of time for a run, we require multiple runs to tune the hyperparameters as explained in previous chapters.

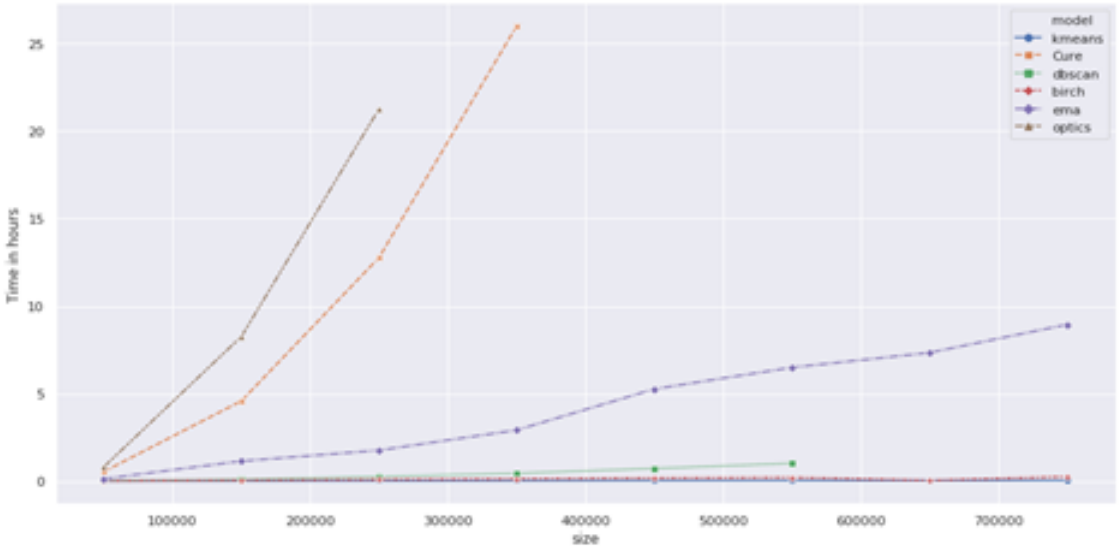

Figure 3.7: scalability Results of clustering algorithms

3.9 Clustering Results and Evaluation

We mentioned that not all clustering algorithms can be used on the preferred dataset as they fail to scale, but this does not stop us from running these algorithms on a smaller sample so that at least we know if it possible to extract meaningful patterns.

3.9.1 CURE

This algorithm is extremely inefficient and naturally cannot scale to the original data set. K was incrementally changed from 4 to 15. On the sample data set this algorithm finds 8 clusters but unfortunately 4 of the clusters contain only 1 member meaning that this algorithm is highly damaged by the outliers and cannot be used for this task. Even other tests showed the same problem suggesting that this algorithm and this data do not form a good combination.

3.9.2 BIRCH

This algorithm is tested on the sample data set and the result was hardly interpretable due to high error rate. This was due to low density that caused the algorithm to find clusters with unrelated sessions. The algorithm was used on the main data set that naturally is denser but now the low compression rate demanded more memory which caused a crash.

3.9.3 EMA

EMA was tested only on the sample data because it is somewhat inefficient. The number of clusters was incrementally changed from 4 to 15, although the algorithm found less than assigned clusters every time the clusters were almost identical. The main assumption of this algorithm is that data points are generated by gaussian distributions. Regardless of the hyperparameter this algorithm returned 3 cluster. Each distribution contains some similar sessions along with a few different ones. This causes the results to be similar without any extreme variable that would indicate a possible playstyle making the results unusable.

3.9.4 BANG

As mentioned in the introduction this algorithm can only find clusters if the sessions are not close and dense, in this scenario this algorithm only finds 1 cluster for all number of partitions from 2 to 20.

3.9.5 K-MEANS with Random Initialization

This algorithm is computationally efficient therefore it is tested on both datasets. K was changed from 4 to 20. The results on the sample datasets has a high error rate and this means that density of the data does not reach the desired amount. But on the original dataset the result shows some worthy playstyles. The algorithm was tuned, and the experiment shows that eight clusters is ideal.

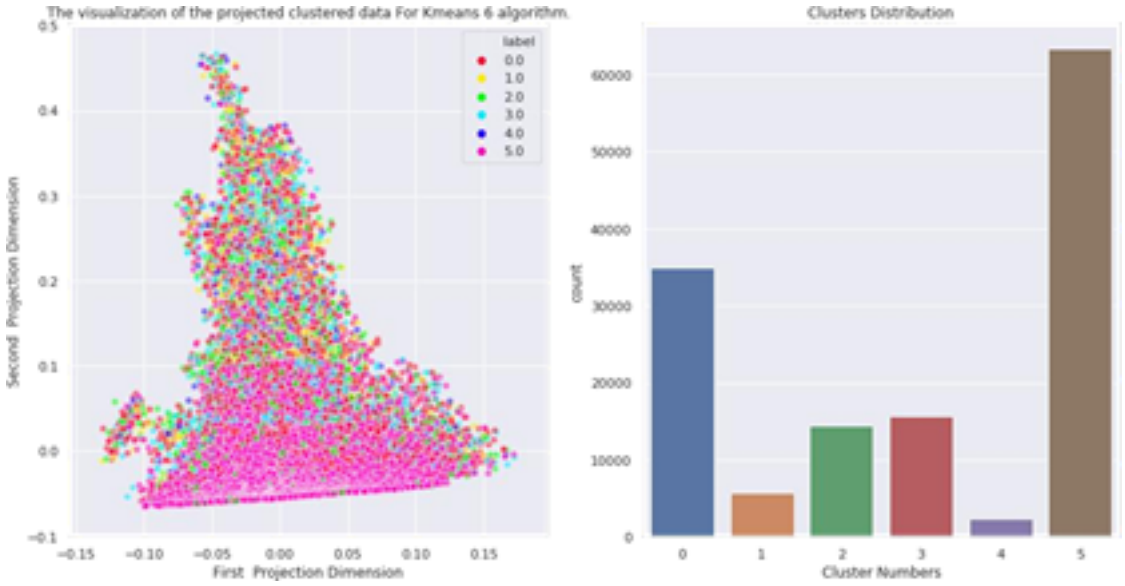

Figure 3.8: K-means results and distribution of the clusters.

Tables 3.4 and 3.5 show the average for the resulting eight clusters, for which we are giving a possible interpretation based on their summary data.

Cluster number	Playstyle name	Population	Population (Normalized)
Cluster 0	Casual gamer	82,254	11%
Cluster 1	Bet strategist	7,000	0.90%
Cluster 2	Bet strategist (Cons)	23,700	3%
Cluster 3	Multiple Cash in fast better	172,929	24%
Cluster 4	Just a few bucks	1,818	0.20%
Cluster 5	Intense gambler	359.111	50%
Cluster 6	Intense no time gambler	54,608	9%
Cluster 7	I win I leave	2464	0.30%

Table 3.3: Overview of play styles

Label	STD BetWage	STD BetTime	Mean Rlpg	Duration (sec)	Num Games	Mean Cash-in
Casual gamer	0.2	140	0.4	681	1.55	13.5
Bet Strategist	0.7	19	0.22	80	0.897	48
Bet Strategist (Conservative)	0.6	34	0.25	137	1.1	43
Multi cash-in fast better	0.35	79	0.43	301	1.4	22
Few Bucks guy	0.11	2	0.03	28	0.14	41
Intense gambler	0.42	372	1	1325	2	19
Intense no time gambler	0.48	51	0.3	201	1.2	32
I win I leave	0.01	1	0.002	52	0.03	12

Table 3.4: K-means results explanation

Cluster 0, Casual Gamer

Contains about 11% of the players, these players seem to enjoy the playing environment rather than wining. Therefor their focus is to not to lose much money and their net value and 80 cents average bet are good signs of this behavior. These people seem to have no rush in betting and show minimal interest in secondary games.

Cluster 1, Bet strategist

Practiced by about 0.9% of gamblers these people have relatively higher primary wagers than other and a prefer medium speed betting. Their net value is about 50% that indicates they are risky players. The dominant characteristic of such players is their primary and secondary wager fluctuation which is about 50 cents. The reason behind this behavior is not clear. The player although might decide to take a less risky approach to betting after a sequence of losses or the opposite. They might even do it when they encounter a set of games.

Cluster 2, Bet strategist (Conservative)

Same as Bet strategist (cluster 1), but with less risky approach to gambling which is reflected in the lower average wager and standard deviation of wager, but still relatively risker than other players. The majority of bet strategists have this approach to gambling. But this does not increase their net value significantly.

Label	STD Cash-in	Net Value	Mean Wager	Bet Per Second	Mean Residual	Secondary Primary Ratio	STD Secondary Wager
Casual gamer	2	0.99	0.8	0.145	0.19	0.14	0.09
Bet Strategist	0.11	0.53	3.8	0.141	0.002	0.12	0.46
Bet Strategist (Conservative)	0.32	0.56	2.69	0.177	0.007	0.1	0.28
Multi cash-in Fast better	1.61	0.79	1.13	0.187	0.03	0.12	0.14
Few Bucks guy	0.16	0.91	1	0.02	0.0029	0.01	0.07
Intense gambler	5	1.69	0.98	0.19	0.14	0.2	0.21
Intense no time gambler	0.8	0.7	1.75	0.18	0.034	0.12	0.22
I win I leave	0.27	22	0.08	0.003	0.02	0.0005	0

Table 3.5: K-means results explanation continued.

Cluster 3, Multiple Cash-in, Fast Better

A group of costumers bet intensely fast and spend around 5 minutes playing. Their average cash-in is about 22 dollars making which is least than their expenses meaning that these players cash-in multiple times. Their net value is less than 1 meaning that these people don't really have a good economical playstyle.

Cluster 4, Multiple Cash in Fast Better

A vary rare behavior that is only seen in 0.2% of players. The high average cash-in which is about 41 dollars does not get along with 28 seconds play time. Also, they do not bet many times and do it very slowly. This playstyle although is not pure, meaning that it can have a hidden playstyle inside it since the average games is less than 1. This indicates that some players did not play a game at all. Players who have this behavior might be newcomers who will halt gambling and have second thoughts.

Cluster 5, Intense Gambler

Majority of players. Fastest betters, they tend to be riskier than other players when they encounter secondary games. They spent more than anyone. The most prominent feature of these players is their net value which is higher than one, meaning that these

players leave casino with more money than they played. The only approach that may be responsible for this is their riskier approach to secondary games.

Cluster 6, Intense No Time Gambler

This group spend about 3 minutes playing with a very high pace. With a net value close to 0.7 it can be assumed that these players do not really care about losing or that these are players who might experience a bad sequence of loses and just withdraw from playing.

Cluster 7, I Win I Leave

A group of players who halt playing as soon as they win. The insanely high net value (22) suggests that these group play extremely safe. But 2 features make this playstyle a bit confusing. The players played less than 1 game meaning some people did not actually play. The same assumption can be made from low betting speed.

3.9.6 TKMPP

This algorithm shows great potential due to its technicality. Like K-means this algorithm can find clusters with spherical shape. Therefore, it is expected that the silhouette score should be high, but this is not the case. The silhouette score drops since tuning this algorithm is extremely hard. Figure 3.9 shows how diverse are the clusters and how bad is the Silhouette score.

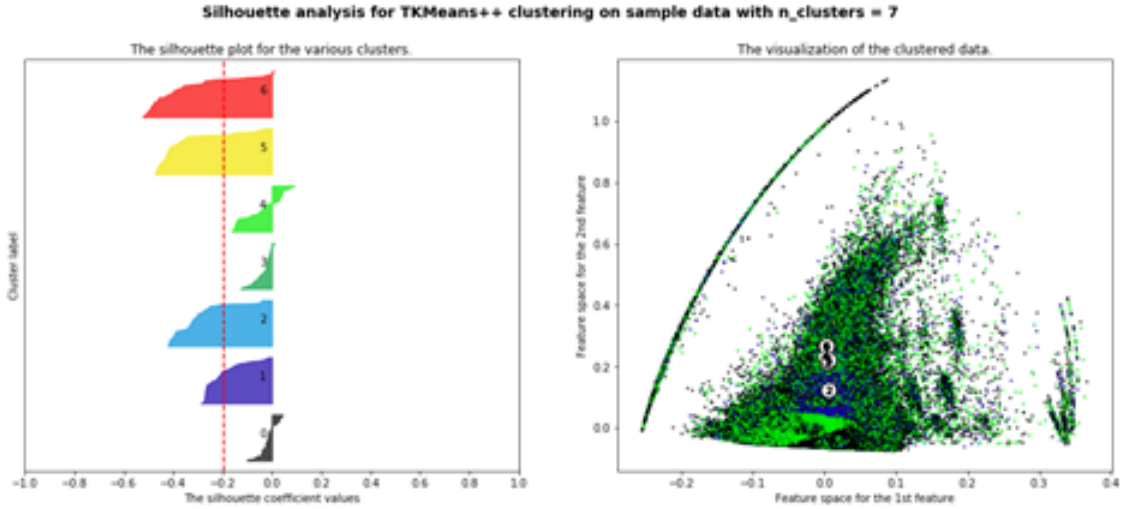

Figure 3.9: Finding the optimal value for clusters using Silhouette coefficient

The analysis shows that no matter how parameters are tuned the results do not seem to belong to a behavior, making the results useless.

3.9.7 DBSCAN

This algorithm fails to find cluster on the original dataset because it needs a huge amount of memory. Due to this limitation, the algorithm could be only tested on the sample dataset. The minimum distance is set to 0.012 and minimum neighbors is set to 28 after testing the following range for it [26, 28]. Minimum distance higher than 0.012 results in one monolithic cluster and less than 0.012 results in too many clusters. With specified hyperparameters the algorithm finds interesting results.

Cluster number	Playstyle name	Population	Population percentage
Cluster -1	Risky Better	6,280	4%
Cluster 0	Casual Gamer	136403	95%
Cluster 1	Cold Feet Cons	52	0.03%
Cluster 2	C	39	0.02%
Cluster 3	No Strategy Player	111	0.07%

Table 3.6: DBSCAN clusters overview

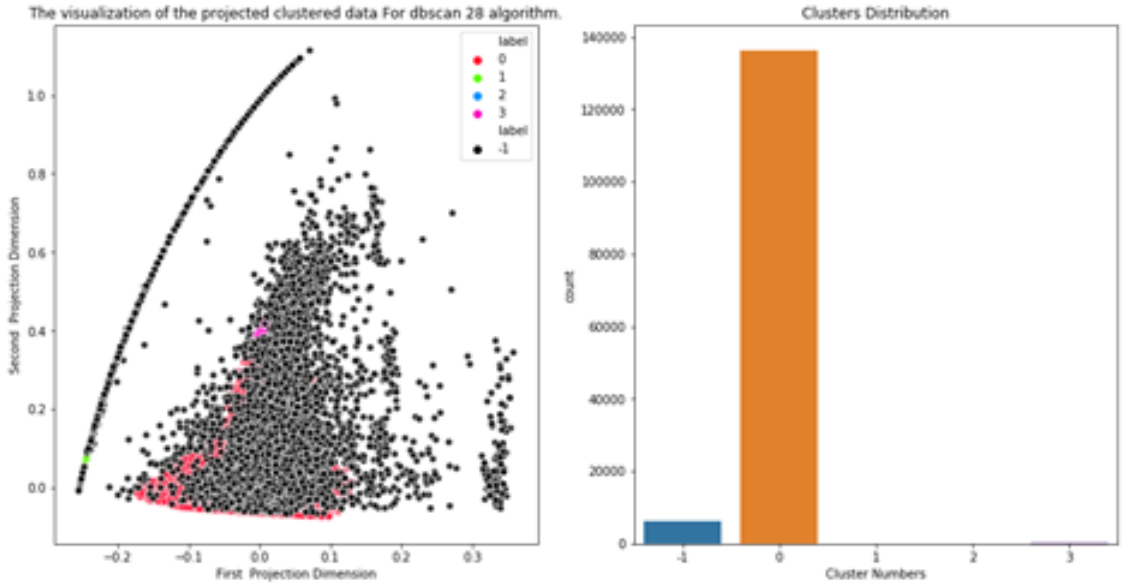

Figure 3.10: DBSCAN results illustrated.

As it is obvious from Table 3.6, the cluster 0 is extremely big and populated.

Table 3.7 shows the clusters and their properties: the clusters are again named, and their population properties are shown.

Label	STD Wager	STD BetTime	Mean SecToPrim	Duration (Seconds)	Game Count	Mean Cash-in
Risky Better	0.71	52	4.7	200	0.98	29
Casual Gamer	0.38	236	0.62	871	1.8	21
Money Launderer	0	0	0	84	0	9.2
Money Launderer (Cons)	0	0	0	80	0	14.6
No Strategy Player	0.2	21	0.05	98	1	51

Table 3.7: DBSCAN clustering: Variable Summary

Cluster -1, Risky Better

The outliers found by DBSCAN. The low error rate and meaningful variables allows us to label it as a playstyle. These players tend to play pretty risky, and actively

look for secondary games, when they encounter them, they place the same amount of wager on them unlike other players. They also have a high average wager and they tend to fluctuate their secondary wager a lot meaning that the risk taking differs from game to game or scenario to scenario.

Cluster 0, Casual Gamer

This is a tricky cluster, monolithic and densely populated. It does not reflect a certain playstyle. It was assumed that DBSCAN is initialized with parameters that results in such phenomena. To tackle this issue, the Min Distance was decreased. Predictably, the cluster was chunked into 300 mini clusters that did not resemble anything. Therefore, another trick was utilized. This cluster was fed to a KMEANS model that found 6 clusters in it that reflect a certain playstyle.

Cluster 2 and 3, Cold Feet

A group of people who target machines with some money left on them who later insert some cash inside them and cash out within 2 minutes making their net value slightly higher than 1. The reason why they do this is unknown, but it might belong to people who might get cold feet or people who just want to obtain a receipt for their money. The conservative Cold Feets insert less amount of cash to the machine while the other inserts bigger amounts.

Cluster 4, No strategy player

Average paced betters who do not really care about their net value. They would rather insert a big check in the machine and play a few rounds. They might be people who are trying to reduce their playtime or newcomers who just want to check out a few games and see how the machine work.

3.9.8 K-means Result on Cluster 0

K-means is tested on this monolithic resulting cluster.The exact same process was tested to tune hyperparameters and finally, six clusters were found suitable since the result had a high silhouette score. And the clusters seemed to be presenting

a meaningful playstyle. To compare the result of both tests resulting clusters are compared together with respect to their relative populations. Then a possible overlap can be detected. Figure 3.11 shows how the sessions are distributed among the clusters.

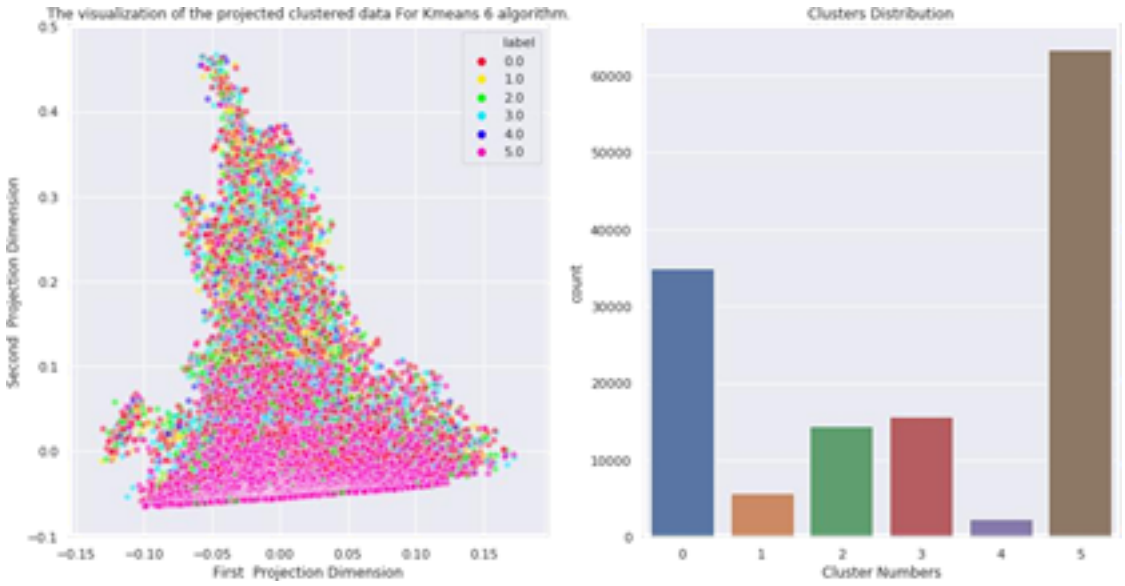

Figure 3.11: Kmeans results with 2D projection.

For convenience, the resulting clusters are not named because the results are like results of K-means on the original dataset.

Cluster number	Population	Population normalized
Cluster 0	34,731	25%
Cluster 1	5,651	5%
Cluster 2	14,191	10%
Cluster 3	14,689	11%
Cluster 4	2,541	2%
Cluster 5	63,341	46%

Table 3.8: K-means finds similar cluster on DBSCAN results

As it is portrayed in both tables there is strong similarity between playstyles found in both K-means results. This proves that sampling does not change the data so drastically that would cause in change of results. In the next sections each of the clusters are explained thoroughly.

Label	wager STD	Bet Time STD	RLPG Mean	Duration Count	Games Count Cash-in	Mean
Cluster 0	0.34	94	0.33	355	1.4	21
Multi-Cashin Fast better	0.35	79	0.43	301	1.4	22
Cluster 1	0.51	45	0.14	181	1.2	39.6
Bet strategist	0.7	19	0.22	80	0.897	48
Bet strategist cons	0.6	34	0.25	137	1.1	43
Cluster 2	0.23	145	0.38	711	1.5	13.44
Casual Gamer	0.2	140	0.4	681	1.55	13.5
Cluster 3	0.38	58	0.19	230	1.2	28
Intense no time gambler	0.48	51	0.3	201	1.2	32
Cluster 4	0.46	30	0.07	126	1.07	47
Cluster 5	0.43	404	1	1441	2.28	19
Intense gambler	0.42	372	1	1325	2	19

Table 3.9: K-means finds results on DBSCAN that can be viewed as refined clusters previously found by K-means used on the main data set

Cluster 1

Contains about 5% of the data, has strong similarity with bet strategist playstyles although bet strategy playstyles had about 2% population. It seems that if the dataset becomes bigger then there would be mini playstyles like conservative bet strategist and bet strategist that basically show different variation of the same playstyle. The high primary wager standard deviation and high primary wager are indicators of this playstyle.

Cluster 2

Contains about 10% of the data, almost same as casual gamer from original K-means playstyles. Strong similarity between variables show that again the same playstyle is identified. Low wager and betting pace along with net value close to 1 are indicators of this playstyle.

Cluster 3

This cluster contains about 11% of the data. Strong similarity with Intense no-time gambler variables show that the same playstyle is identified. Although population

wise, they are not the same since in the original K-means results this cluster contains about 1% of the sessions. High paced gambling, low play time are indicators of this playstyle.

Cluster 4

Contains about 2% of the data, this playstyle has no similarity with playstyles found by original K-means. The zero standard deviation of cash-in shows that there is only one cash-in happening during the playtime. High cash-in and high fluctuating primary wager along with fast paced gambling may belong to someone who has problem gambling but is trying to overcome it by setting a money limit for himself, since the only heavy gambler indicator that is missing is long duration gambling.

Cluster 5

Contains about 46% of the data, the biggest cluster in the data. almost THE same as intense gambler from original K-means playstyles. Strong similarity between variables show that again the same playstyle can easily be predicted since this playstyle is prevalent and dominant. Fast betting speed along with spending a long time playing are main indicators of this playstyle.

Chapter 4

Playstyle Prediction

4.1 Introduction

This part of the research is heavily dependent on results of the previous chapter. Our choice of unsupervised learning is based on results of K-means algorithm. We chose K-means because it is scalable, efficient and simple along with the fact it results in interpretable clusters. Similar to previous project, first we identified gaming sessions using the similar algorithm then we extracted some statistical features that helped us with detecting weak sessions and dropping them.

After extracting the features and dropping weak sessions, we trained K-means algorithm and fine tuned it to find 8 clusters. These clusters were then used as labels for the gaming data. Later on, we trained the candidate models on truncated sessions to have fine tuned classifies. The classifiers results were plotted to find optimal number of input transactions. Our results and plots show how 40 transactions is the optimal number of transactions needed for training a classifier both in terms of performance and number of input transactions. Our results show that MCLSTM architecture has the best performance compared to to other algorithms and shows great potential. Figure 4.1 shows a brief project flow and the order of tasks and their milestones.

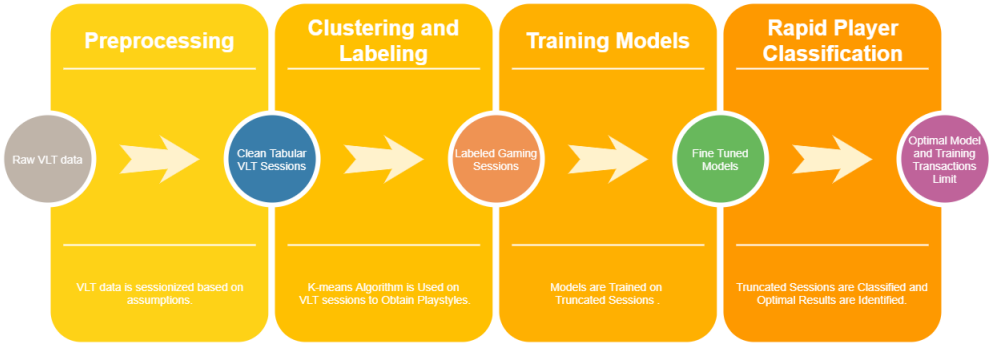

Figure 4.1: Project workflow

In the next sections we explain how labeling sessions and classification is done in more details.

4.2 Labeling Sessions

Since none of the playing sessions is labeled we need to come up with a way of assessing proper labels for them. Previous work focused on problem gambling detection and gaming data shows that K-means with random initialization can be used to detect players with risky gambling behavior. Also our results mentioned in previous chapter show that K-means is not only scalable and efficient but it can also be used to detect playstyles.

This inspired us to use the same methodology to detect possible customers. This methodology will allow us to use the same labels found by the clustering method as ground truth for our data. For this task, roughly one million playing sessions were extracted. To ensure that no weak playing sessions are included, we dropped every session that was in an idle state for more than most of its playing time or was too long. This results in 916 148 valid playing sessions. This methodology is thoroughly explained in previous chapter. These sessions were collected in 2020 and were slightly more fresh than the ones used in previous project. Again, using extracted features, we dropped sessions that made no contextual sense, sessions that last for weeks or sessions that have no bets placed or contain numerous pauses that last more than a few hours.

Next, we used k-means algorithm with eight clusters to find the alleged playstyles. The acquired playstyles are confirmed to exist and observed in venues. Needless to say, we used the same method as before to find optimal number of playstyles. As requested by the business partner we used a different set of features to investigate if using different features can facilitate us in finding the playstyles. A few of this features are mentioned in Table 4.1. The assumptions made about the clusters is just a way to interpret the playstyles and the associated features and values they represent. Using a subset of features mentioned in Table 4.1 and k-means playstyles in Table 4.2 were obtained, these clusters were verified by the business partner and the labels are used as ground truth for our classifiers.

Feature	Explanation
Wager Fluctuation	Relative fluctuation of the wager
Number of Secondary Games	How many bonus games were prompted in a session
Wager Risk	What percentage of wager is used when betting
Bet Intensity	Betting speed
Number of Games	Number of different games visited during a session
Number of cash-in	number of cash-ins during a session
Primary win	sum of primary win
Average secondary wager	Average wager on secondary games
Average bet time	How long the player waits to decide the next bet
Secondary to primary	How risky the player approaches the secondary games relatively

Table 4.1: Features and their explanations

4.2.1 Clustering Results

4.3 Classification of Labeled Truncated Sessions

The goal is to approximate how many transactions are needed for classifying the playing sessions. The number of transactions observed are explored in the following increments: 20, 40, 60, 80. Two sets of features are extracted from each playing session. First, the statistical features that describe each playing session are the same as features used for clustering, with a few additions such as the maximum wager for a few transaction periods, or average machine credit. Some of these features are included in Table 4.3. These features are extracted for every 20 transactions. The added features give more local information than session wide features. This is a quick fix for the lack of temporal features and the fact that DTW is almost impossible to use as a distance measures as it is inefficient compared to Euclidean distance. Ignoring the temporal nature of the EGM data is not advised since times of play decisions are closely related to the machine responses witnessed throughout the gaming sessions. It is worth mentioning that the feature space might be sparse and have a low variance when the whole playing session is not used. For example, the player might not do any cash-outs withing 20 transactions, making any cash-out related feature like net value meaningless.

Cluster	Nickname	Explanation	Population	Transactions (mean)
0	Casual Entertainers	Tends to change the game more than others, Don't chase secondary games, Avoids losing.	66%	148
1	Engaged Entertainers	Handles playing casually, Plays to spend time, tries not to lose, Plays conservatively.	13%	260
2	Less-engaged Players	Plays without caring about losing, Understands the function of bonus games, Plays cautiously after wining, Spends minor amount of time and money.	13%	31
3	Risk Seekers	Actively searches for secondary games, Usually cash-ins multiple times, Places risky bets on bonus games, Have a high net value.	3%	228
4	Cautious Players	Slow-paced player, Plays safe after losing, Tends to stop playing before losing a lot.	3%	15
5	Early Winners	Highest net value among players, decreases wager after losing, Places a few bets and leaves as soon as wins	1.9%	3
6	Outlier Events	Playing sessions that included faulty cash-ins, Try to make use of remaining credit in machines, Might be trying to get a receipt for their money	0.1	1

Table 4.2: Each playstyle is given a nickname based on interpretation of playstyle features

FEATURE NAME	EXPLANATION
AVERAGEWAGER_X-Y	Average wager placed starting from the Xth transaction to the Yth one.
MAXCREDIT_X-Y	Maximum machine credit from the Xth transactions to the Yth one.

Table 4.3: Each of these features is extracted for every 20 transactions to give model more information about momentum behavior of the player.

The second set of features are summations of the transactions. Each transaction is described with 4 sets of features. Game ID is the unique id corresponding the current game being played, wager shows the currently placed wager, timestamp is the time in seconds between two consecutive transactions and machine credit that show the amount of money available in the machine. Each of these variables are later treated as independent time series and are used to train the neural network model. For the classic models 30,000 sessions are used and for the neural network 820,000 sessions were used. 45,806 sessions are also used as test set for both sets of models. The set was kept large to maintain sufficient variance for the neural network model to be trained on.

It should be explained why we opted for this more complex clustering method, rather than just clustering the data with a restricted number of transactions. A challenge with this simpler method is that there is no direct mapping from clustering sessions with few transactions to clustering complete sessions. Some features do not scale linearly. For example most sessions usually have only one cash-out that happens in the end, given that most of these features are discrete, using a portion of the playing sessions only would lead to data loss. Whether there exists a set of features that can be used for clustering limited play sessions that ultimately results in the same results is one thing that is yet to be discovered. Another problem with this simpler approach is that K-means does not learn the representations but, like KNN algorithm, it groups the data relatively. This makes the inference task difficult if not impossible. If this task is to be conducted at playtime when the player is interacting with the machine, then the session should be added to the existing playing sessions and then be clustered again. Yet again there is no way to guarantee that K-means would produce the exact same clusters therefore it is again needed to be checked and validated. We were thus led to conclude that using unsupervised learning for assessing playstyles at run-time is inefficient and likely infeasible.

The first series of classifications models are non-deep models. For this purpose, K Nearest Neighbors (KNN), Decision Tree Classifier (DTC), Random Forest Tree (RFT) and Perceptron were used. Each model is fine-tuned for each set of transactions and the best result was used as measure of general performance for this data set. These models give a better comparative insight and act as a baseline for the neural network architecture. The second set of models include Neural network based models that are designed to capture complex temporal parents. We already knew that using neural networks means requiring heavier computational power and dealing with more parameters and generally more complexity, but we already know that this tradeoff is rather positive cause it offers us a model that requires data with the least amount of trimming, preprocessing and other feature extraction related tasks. Note that the feature extraction methods similar to what we described in previous section usually bring us unwilling data loss—a data loss that means lower performance for the model. Before presenting results and conclusions, we go through the models and name their weak points and strength in detail and explain why we are more keen on using these

models and why we have chosen them over other models. In the next section, we explain why each model is chosen and can be a possible good candidate, we also elaborate on how each of them work and what is the intuition behind.

4.4 Classification Models Explained

From algorithms we have named so far, KNN, Perceptron, DTC and RFT are used from scikit-learn package while neural network is provided from Keras and Tensorflow. Usual packages like Pandas and Numpy were also used for usual tasks like preprocessing and loading the data. For visualizations we used Matplotlib and Seaborn.

4.4.1 K Nearest Neighbours

KNN algorithm is the simplest algorithm we have chosen to use on our data. KNN assesses the label of data points based on the K closest neighbours to the data point. Closeness can be measured using a distance measure like the Euclidean distance. Therefore for each point waiting to be inferred the model has to find the K-nearest neighbors to it, depending on the size of the dataset this can be computationally expensive. The inference complexity of this algorithm is $O(N \times M)$, N being the size of the dataset and M being the number of features, on the other hand the training complexity is $O(1)$.

As mentioned, this method does not require the mainstream training phase but it needs the training data for inference. The reason we chose this algorithm is the simplicity of it and how it is similar to K-means in core. Therefore this algorithm has a high chance of showing a good performance.

4.4.2 Decision Tree Classifier

The Decision Tree Classifier (DTC) is based on a training algorithm which builds up a tree structure that has conditional nodes for accessing the flow of each data point at inference time. Nodes with no children are called leaves. At each node conditional phrases are based on a set of variables while roots serve as labels. Before explaining how the model is trained at training time, we first need to explain how the concept of entropy works, how it is measured and how it can be optimized. Note

that there exist different Decision Trees that are trained in a different manner with a different optimization technique. This particular type of decision tree works based on information gain concept.

Entropy is a measure used for showing how pure is a set of points with regards to their labels. Splits made in each nodes later causes the points to accumulate in a leaf, a pure leaf is a leaf that ideally contains only data points belonging to one class. Declaring that p_i is the frequency of data point in our dataset entropy is measured in the following schema.

$$Entropy = \sum_{i=1}^{n} -p_i \log_2 p_i \tag{4.1}$$

In order to minimize Entropy (E), we have to introduce another concept called Information Gain (IG). Information gain of Y given valaue X is calculated in the following way:

$$IG(Y|X) = E(Y) - E(Y|X) \tag{4.2}$$

IG shows how pure the nodes with regards to label Y will be based on a split on value X. In simple terms,information gain investigates how pure the leaves become after considering X as split condition. During training the model tries to finds the best splits for an optimal tree. This model can capture complex patterns and scale up to big data with training complexity of $O(N \times K \log_2 N)$, N being size of the dataset and K being number of features). DTC is prone to over fitting but is interpretable and understandable. These features make it a great candidate for our research.

4.4.3 Random Forest Tree

Random Forest tree trains N Decision trees that are differently trained with subsets of the dataset. Each tree is given a subset of features and a group of data points to be trained. These features can overlap, same for the training points, based on our preference at training time. RFT is not only more complex and can capture multiple patterns but also less prone to overfitting. These enhancements come with losing interpretablity and additional training time. We anticipated that if DTC overfits, or fails to get good results, RFT might be a good substitute.

4.4.4 Perceptron

Perceptron is a single neuron that is used in neural networks. Perceptron functions like a linear classifier. Perceptron applies a step function to linear weight sum of the features and determines the output class based on that. The step function can be a sigmoid function to a RELU function. Like linear classifier, Perceptron also uses gradient decent to optimize itself. Perceptron is the most basic neuron that can be used in neural networks, therefore it can be a perfect baseline for us.

4.4.5 MCLSTM with Embedding layer

The neural network architecture is inspired from Karim et al. [69]. The model proposed in this feature uses the transactions discussed in the previous section. The reason that Liu et al. [64] multivariate convolutional neural network (MCNN) model was not used is that it suffers from a few minor issues. First, MCNN lacks a mechanism to learn the long term dependencies from the transactions. Convolutional networks perform well on images but for the time series they lack a mechanism for understanding the order of data. Another problem this model faces is that it favors time series with numeric data, but each transaction also includes the game id, which either should be discarded, which is not practical, or it should be transformed to another form like one hot encoding or transforming it to embedding vector. Although it is possible to overcome these challenges with a few modifications to the model, these modifications transform the model to Karim's model.

We thus opted for the modified model described by Karim et al. [69]. The multivariate convolutional LSTM neural network proposed by Karim is enhanced with an extra embedding layer that is applied to game id. This modified model makes an embedding representation for the game id, trains the LSTM layer with the all the features and applies a convolutional layer followed by a pooling layer to timestamp, wager and machine credit. The result of all of these layers are concatenated and then fed to dense layers to conduct the classification. The convolutional layers and the LSTM layers make a new representation of the data. Treating each of these variables as independent variables is beneficial and makes using the convolutional layers possible, and also to minimize risks of this assumption, the LSTM layer is trained to make another representation that takes longer dependency of sessions into account rather

than extracting features that only rely on the filter size.

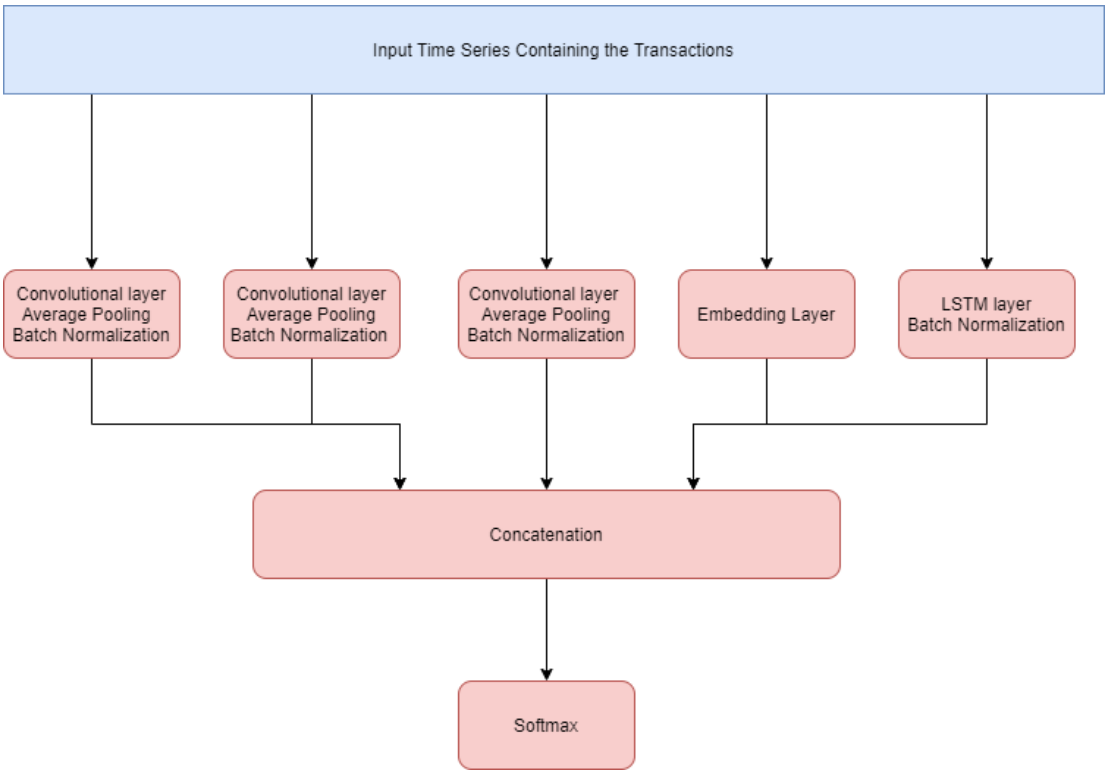

Figure 4.2: The Neural network architecture

4.5 Rapid Player Classification

4.5.1 Comparison of Classifiers

To understand which model is superior in terms of performance, an evaluation metric must be used that can demonstrate the performance better. For a class-wise comparison recall and precision were used but to compare the general performance of the models F1-measure issued. Table 4.4 shows the each models performance for each cluster while Figure 4.4 is used for detecting the possible optimal value for the number of transactions. To better judge the results, we have to take into account that class 0, 2 and 3 contain longer playing sessions so therefore the highest priority goes to the models that can identify them better. in Table 4.4 it is showed that MCLSTM outperforms all other models in terms of precision with a few exceptions while performing well in terms of Recall as well. Another surprising find is that KNN

with all simplictity performs relativly well compared to DTC and RFT making it a good candidate since it is much simpler and easier to train. Perceptron also can be a good choice if detecting Cluster 2, 5 and 6 is the main goal as it does not perform well on class 0, 1 and 2. Another interesting find is that Perceptron's performance peaks at 40 transactions and decreases as the data set size increases to 60 transactions showing how the model lacks scalability to detect more complex patterns. With regards to F1-measure KNN scores relatively well, but it should be noted that since both KNN and K-means use the same intuition it might be better to use RFT just to avoid biased results. As it can be seen in Figure 4.4, the neural network model on average outperforms the other models. Clearly the multivariate LSTM model is better in various ways. Multivariate convolutional LSTM (MCLSTM) is trained on more data and has more parameters, which means it can fit better given more data which in this scenario plenty is available.

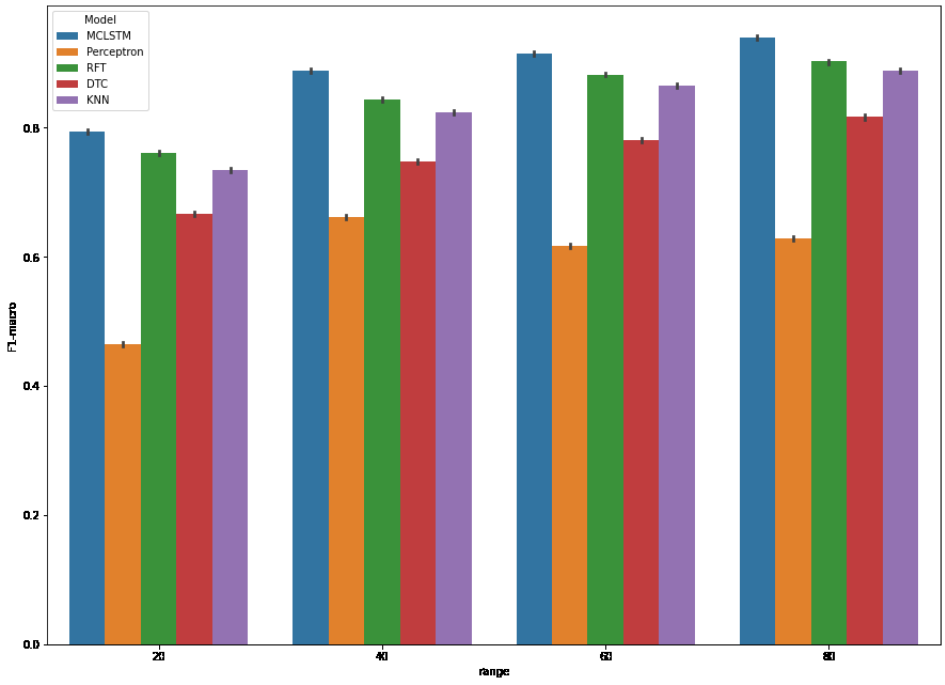

Figure 4.3: This bar plot with 95% confidence intervals provides additional insight on our results.

Also, MCLSTM can extract local features and can understand long term dependencies and features based on order of transactions. There are even minor things that MCLSTM improves upon, for example the inference time is significantly lower

than KNN model. It is also worth mentioning that KNN performs almost as good as random forest tree, which is fairly complex. The reason is that KNN can be seen as a supervised version of K-means algorithm. KNN uses Euclidean distance to find the closest neighbors and assigns the dominant label among those neighbours to the test point. Being inherently similar to K-means algorithm it can be justified that this model is matching an already identified pattern.

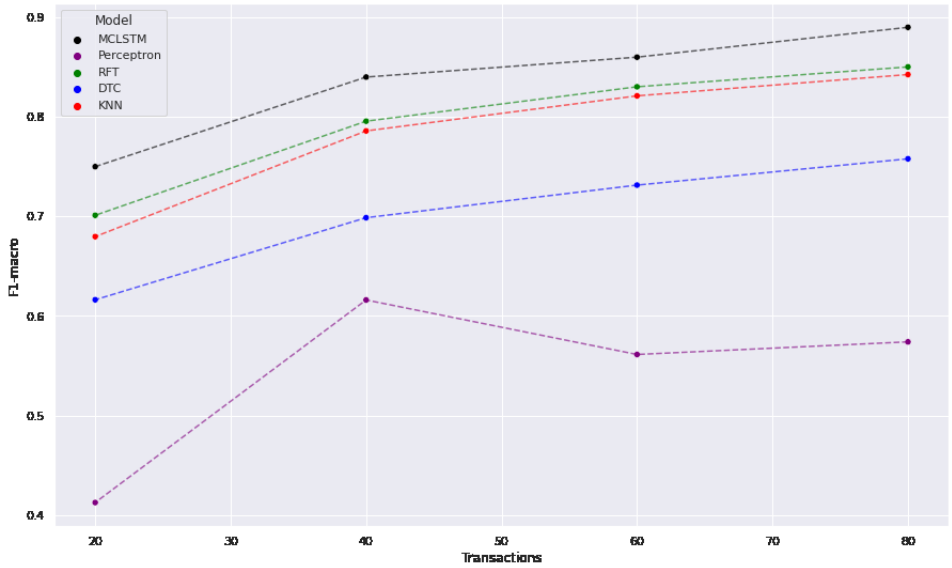

Figure 4.4: F1-macro visualized for finding the optimal value for number of transactions

Recall									
		Class							
Transactions	Model	C0	C1	C2	C3	C4	C5	C6	F_1
20 Transactions	KNN	0.49	0.56	0.78	0.58	0.83	0.98	1	0.68
	DTC	0.39	0.55	0.68	0.55	0.82	0.97	1	0.62
	RFT	0.51	0.6	0.8	0.66	0.86	0.99	1	0.7
	Preceptron	0.01	0	0.76	0.45	0.65	0.96	0.99	0.41
	MCLSTM	0.46	0.53	0.96	0.46	0.76	0.92	0.99	0.75
40 Transactions	KNN	0.63	0.76	0.85	0.68	0.93	0.99	1	0.79
	DTC	0.5	0.7	0.75	0.67	0.9	0.96	1	0.7
	RFT	0.66	0.78	0.84	0.76	0.94	0.99	1	0.8
	Preceptron	0.58	0.63	0.38	0.8	0.93	0.98	0.98	0.62
	MCLSTM	0.59	0.77	0.96	0.57	0.89	0.92	0.99	0.84
60 Transactions	KNN	0.7	0.83	0.87	0.72	0.94	0.99	1	0.82
	DTC	0.58	0.77	0.79	0.7	0.91	0.96	1	0.73
	RFT	0.74	0.85	0.86	0.8	0.95	0.99	1	0.83
	Preceptron	0.15	0.53	0.37	0.91	0.9	0.97	0.97	0.56
	MCLSTM	0.67	0.84	0.97	0.6	0.91	0.9	0.99	0.86
80 Transactions	KNN	0.76	0.86	0.88	0.76	0.95	0.99	1	0.84
	DTC	0.63	0.81	0.82	0.72	0.91	0.97	1	0.76
	RFT	0.78	0.88	0.88	0.82	0.96	0.99	1	0.85
	Preceptron	0.04	0.4	0.88	0.8	0.93	0.96	0.97	0.57
	MCLSTM	0.72	0.88	0.97	0.66	0.91	0.91	0.99	0.89
Precision									
20 Transactions	KNN	0.38	0.48	0.92	0.38	0.58	0.76	0.93	0.68
	DTC	0.28	0.44	0.91	0.21	0.51	0.68	0.97	0.62
	RFT	0.4	0.53	0.93	0.38	0.64	0.74	0.94	0.7
	Preceptron	0.15	0.75	0.91	0.59	0.57	0.83	0.01	0.41
	MCLSTM	0.55	0.72	0.86	0.64	0.85	0.95	0.99	0.75
40 Transactions	KNN	0.49	0.74	0.94	0.54	0.81	0.8	0.9	0.79
	DTC	0.37	0.61	0.93	0.29	0.68	0.73	0.98	0.7
	RFT	0.48	0.78	0.96	0.5	0.83	0.79	0.94	0.8
	Preceptron	0.2	0.38	0.98	0.34	0.58	0.83	0.84	0.62
	MCLSTM	0.66	0.89	0.91	0.74	0.92	0.95	0.98	0.84
60 Transactions	KNN	0.54	0.8	0.96	0.61	0.83	0.83	0.92	0.82
	DTC	0.43	0.71	0.95	0.34	0.72	0.68	0.94	0.73
	RFT	0.54	0.84	0.97	0.59	0.85	0.8	0.94	0.83
	Preceptron	0.05	0.81	0.83	0.25	0.81	0.86	0.38	0.56
	MCLSTM	0.71	0.92	0.93	0.78	0.91	0.97	0.98	0.86
80 Transactions	KNN	0.57	0.83	0.96	0.66	0.85	0.84	0.94	0.84
	DTC	0.47	0.75	0.95	0.4	0.71	0.73	0.95	0.76
	RFT	0.6	0.87	0.97	0.63	0.86	0.8	0.94	0.85
	Preceptron	0.39	0.88	0.95	0.57	0.84	0.83	0.02	0.57
	MCLSTM	0.75	0.92	0.94	0.8	0.91	0.96	0.98	0.89

Table 4.4: MCLSTM outperforms most of the models in precision recall and class-wise

Chapter 5

Conclusion and Future Work

In this chapter, we briefly go through every milestone we have achieved so far and elaborate on our contributions to the field of behavioral analysis and data mining. Furthermore, we explain how we can utilize and enhance these results and efficiently use them in real world scenarios.

5.1 Clustering of Play Sessions

5.1.1 Conclusions of Play Session Clustering

Our results and analysis show that customer segmentation and playstyle detection aimed on EGM data requires efficient unsupervised learning algorithms with the least number of hyperparameters. In case of TKMPP, we showed no matter how technically superior one clustering algorithm can be, fine tuning it might not be as easy as it sounds. We found out that the computational efficiency is also a strong requirement as some playstyles are rarely witnessed therefore using small datasets results in noisy clusters that contain similar but inherently different playstyles. Our research shows that one way of customer segmentation is to use K-means on immense datasets that results in playstyles and clusters that might need a bit of clean up afterwards. An alternative to this method for smaller datasets is to use DBSCAN prior to k-means to identify small clusters that would otherwise go undetected and add noise to other clusters. Our results indicate that unsupervised learning can not only be used for detecting problem gambling but can also be used detect more general playstyls and playing patterns.

5.1.2 Future Work in Play Session Clustering

For future work, we can use different features and consultation of experts to extract more playstyles and even use IDed data to track trajectory of a customer's playstyle

and check if the playstyle remains unchanged throughout time or customers may practice different playstyles at different times such analysis can help the gambling business and make it more efficient and promising. In technical aspect, Clustering EGM data can be done by using dynamic time wrapping as distance measure and extracting temporal features. Temporal features can either be learned using vector embedding or other representation learning methods or extracted using statistical methods. The combination of both methods can result in analyzing the EGM data in a slightly different manner. If we are interested in questions such as how can a player approach gaming throughout a session, how do they respond to different events of the game, and how they approach ending their gaming sessions, such questions can be answered easily with this methodology.

5.2 Rapid Playstyle Classification

5.2.1 Conclusions in Rapid Playstyle Classification

Comparison of classifiers and in depth analysis of their performances show that increasing number of transactions past 40 transactions does not drastically increase the evaluation metrics. Although if the goal is to have the best performance then it is advised to increase the input transactions as much as desired, on the other hand if the goal is to do the classification as early as possible we suggest to do it around 20 transactions as a decent performance is observed at that point. MCLSTM model showed promising result in terms of classifying customers in early stages of playing. Our results show that this model can surpass other models using temporal features and raw time series data to correctly classify customers. Although based on preferences and priorities one can use KNN for simplicity, Perceptron for ease of use, RFT for a good balance between complexity and performance, MCLSTM for the highest performance or Decision Tree for interpretablity. Current models can be used in different applications such as a game recommender system, a problem gambling detection system etc.

5.2.2 Future Work in Rapid Playstyle Classification

With raise of demand for interpretable neural network model we can also focus on deploying a model that is less abstract and black box but more intrepretable. Our MCLSTM model does not give us a proper idea on how the classification process is conducted and what is the most important feature that the model uses for the classification task. Although this might seem trifling, measuring the trustworthiness of the classification is highly dependent on interpretability. A model that makes crucial decisions based on irrelevant features may just abuse the randomness of the data and pure chance.

Apart from interpretabilty, we can also strengthen the basis of our work by introducing IDed data. The IDs can help us track the playing trajectory of players and further verify their playing patterns or even be used to manually label the players. If the labels are acquired under different circumstances, for example via a psychological test for assessing problem gambling, then we can conduct even more experiments aimed at finding out weather such classifiers can perform as prominent on the same data that is labeled in a different manner. We can also tweak the clustering process and use temporal features and dynamic time wrapping clustering to detect different playstyles and train classifiers that can detect the alleged behavior at playtime.

Bibliography

[1] A. Kanavos, S. A. Iakovou, S. Sioutas, and V. Tampakas, "Large scale product recommendation of supermarket ware based on customer behaviour analysis," *Big Data and Cognitive Computing*, vol. 2, no. 2, p. 11, 2018.

[2] J. Braverman and H. J. Shaffer, "How do gamblers start gambling: Identifying behavioural markers for high-risk internet gambling," *The European Journal of Public Health*, vol. 22, no. 2, pp. 273–278, 2012.

[3] J. Braverman, D. A. LaPlante, S. E. Nelson, and H. J. Shaffer, "Using cross-game behavioral markers for early identification of high-risk internet gamblers.," *Psychology of Addictive Behaviors*, vol. 27, no. 3, p. 868, 2013.

[4] N. Adami, S. Benini, A. Boschetti, L. Canini, F. Maione, and M. Temporin, "Markers of unsustainable gambling for early detection of at-risk online gamblers," *International Gambling Studies*, vol. 13, no. 2, pp. 188–204, 2013.

[5] M. G. Mosquera and V. Keselj, "Identifying electronic gaming machine gambling personae through unsupervised session classification," *Big Data & Information Analytics*, vol. 2, no. 2, p. 141, 2017.

[6] S. Sylvan *et al.*, "Gambling-productivity commission inquiry report," 2010.

[7] T. Schellinck and T. Schrans, "Intelligent design: How to model gambler risk assessment by using loyalty tracking data," *Journal of Gambling Issues*, no. 26, pp. 51–68, 2011.

[8] V. Keselj, "Use of data mining on electronic gaming machine session variables for responsible gaming support," *Dalhousie University, Faculty of Computer Science. (inner report)*, 2011.

[9] M. A. White, P. Mun, N. Kauffman, C. Whelan, M. Regan, and J. E. Kelly, "Responsible gambling council (RGC)," 2006.

[10] R. T. Wood and M. D. Griffiths, "Understanding positive play: An exploration of playing experiences and responsible gambling practices," *Journal of gambling studies*, vol. 31, no. 4, pp. 1715–1734, 2015.

[11] W. J. Dixon and K. K. Yuen, "Trimming and winsorization: A review," *Statistische Hefte*, vol. 15, no. 2-3, pp. 157–170, 1974.

[12] H. Liu and V. Kešelj, "Combined mining of web server logs and web contents for classifying user navigation patterns and predicting users' future requests," *Data & Knowledge Engineering*, vol. 61, no. 2, pp. 304–330, 2007.

[13] A. D. Gordon, "Classification," *Chapman & Hall, New York*, 1999.

[14] P. Hansen and B. Jaumard, "Cluster analysis and mathematical programming," *Mathematical programming*, vol. 79, no. 1, pp. 191–215, 1997.

[15] A. K. Jain and R. C. Dubes, *Algorithms for clustering data*. Prentice-Hall, Inc., 1988.

[16] R. Xu and D. Wunsch, "Survey of clustering algorithms," *IEEE Transactions on neural networks*, vol. 16, no. 3, pp. 645–678, 2005.

[17] C. M. Bishop *et al.*, *Neural networks for pattern recognition*. Oxford university press, 1995.

[18] A. K. Jain, M. N. Murty, and P. J. Flynn, "Data clustering: a review," *ACM computing surveys (CSUR)*, vol. 31, no. 3, pp. 264–323, 1999.

[19] P. IndiraPriya and D. Ghosh, "A survey on different clustering algorithms in data mining technique," *International journal of modern engineering research (ijmer)*, vol. 3, no. 1, pp. 267–274, 2013.

[20] C. H. Chen, *Handbook of pattern recognition and computer vision*. World Scientific, 2015.

[21] S. Landau and B. S. Everitt, *A handbook of statistical analyses using SPSS*. Chapman and Hall/CRC, 2003.

[22] E. Kolatch *et al.*, "Clustering algorithms for spatial databases: A survey," *PDF is available on the Web*, pp. 1–22, 2001.

[23] S. Oltedal and T. Rundmo, "Using cluster analysis to test the cultural theory of risk perception," *Transportation research part F: traffic psychology and behaviour*, vol. 10, no. 3, pp. 254–262, 2007.

[24] P. Berkhin, "A survey of clustering data mining techniques," in *Grouping multidimensional data*, pp. 25–71, Springer, 2006.

[25] M. B. Eisen, P. T. Spellman, P. O. Brown, and D. Botstein, "Cluster analysis and display of genome-wide expression patterns," *Proceedings of the National Academy of Sciences*, vol. 95, no. 25, pp. 14863–14868, 1998.

[26] P. H. Sneath, "The application of computers to taxonomy," *Microbiology*, vol. 17, no. 1, pp. 201–226, 1957.

[27] G. N. Lance and W. T. Williams, "A general theory of classificatory sorting strategies: 1. hierarchical systems," *The computer journal*, vol. 9, no. 4, pp. 373–380, 1967.

[28] S. Guha, R. Rastogi, and K. Shim, "Cure: An efficient clustering algorithm for large databases," *ACM Sigmod record*, vol. 27, no. 2, pp. 73–84, 1998.

[29] S. Guha, R. Rastogi, and K. Shim, "Rock: A robust clustering algorithm for categorical attributes," *Information systems*, vol. 25, no. 5, pp. 345–366, 2000.

[30] G. Karypis, E.-H. Han, and V. Kumar, "Chameleon: Hierarchical clustering using dynamic modeling," *Computer*, vol. 32, no. 8, pp. 68–75, 1999.

[31] T. Zhang, R. Ramakrishnan, and M. Livny, "Birch: an efficient data clustering method for very large databases," *ACM sigmod record*, vol. 25, no. 2, pp. 103–114, 1996.

[32] R. A. Mollineda and E. Vidal, "A relative approach to hierarchical clustering," *Pattern Recognition and Applications*, vol. 56, pp. 19–28, 2000.

[33] N. Frosst and G. Hinton, "Distilling a Neural Network Into a Soft Decision Tree," *arXiv:1711.09784 [cs, stat]*, Nov. 2017. arXiv: 1711.09784.

[34] J. MacQueen *et al.*, "Some methods for classification and analysis of multivariate observations," in *Proceedings of the fifth Berkeley symposium on mathematical statistics and probability*, vol. 1, pp. 281–297, Oakland, CA, USA, 1967.

[35] G. H. Ball and D. J. Hall, "Isodata, a novel method of data analysis and pattern classification," tech. rep., Stanford research inst Menlo Park CA, 1965.

[36] N. S. Gupta, B. S. Agrawal, and R. M. Chauhan, "Survey on clustering techniques of data mining," *Am Int J Res Sci Technol Eng Math*, vol. 5, no. 2, p. 227, 2015.

[37] J. Sander, M. Ester, H.-P. Kriegel, and X. Xu, "Density-based clustering in spatial databases: The algorithm gdbscan and its applications," *Data mining and knowledge discovery*, vol. 2, no. 2, pp. 169–194, 1998.

[38] M. Ankerst, M. M. Breunig, H.-P. Kriegel, and J. Sander, "Optics: Ordering points to identify the clustering structure," *ACM Sigmod record*, vol. 28, no. 2, pp. 49–60, 1999.

[39] X. Xu, M. Ester, H.-P. Kriegel, and J. Sander, "A distribution-based clustering algorithm for mining in large spatial databases," in *Proceedings 14th International Conference on Data Engineering*, pp. 324–331, IEEE, 1998.

[40] T. Sajana, C. S. Rani, and K. Narayana, "A survey on clustering techniques for big data mining," *Indian journal of Science and Technology*, vol. 9, no. 3, pp. 1–12, 2016.

[41] P. E. Hart, D. G. Stork, and R. O. Duda, *Pattern classification*. Wiley Hoboken, 2000.

[42] G. J. McLachlan, S. X. Lee, and S. I. Rathnayake, "Finite mixture models," *Annual review of statistics and its application*, vol. 6, pp. 355–378, 2019.

[43] M. A. T. Figueiredo and A. K. Jain, "Unsupervised learning of finite mixture models," *IEEE Transactions on pattern analysis and machine intelligence*, vol. 24, no. 3, pp. 381–396, 2002.

[44] G. J. McLachlan and T. Krishnan, *The EM algorithm and extensions*, vol. 382. John Wiley & Sons, 2007.

[45] J. A. Barnes and F. Harary, "Graph theory in network analysis," *Social networks*, vol. 5, no. 2, pp. 235–244, 1983.

[46] R. Sharan and R. Shamir, "Click: a clustering algorithm with applications to gene expression analysis," in *Proc Int Conf Intell Syst Mol Biol*, vol. 8, p. 16, 2000.

[47] A. Ben-Dor, R. Shamir, and Z. Yakhini, "Clustering gene expression patterns," *Journal of computational biology*, vol. 6, no. 3-4, pp. 281–297, 1999.

[48] L. A. Zadeh, "Fuzzy sets," in *Fuzzy sets, fuzzy logic, and fuzzy systems: selected papers by Lotfi A Zadeh*, pp. 394–432, World Scientific, 1996.

[49] E. Van Rest, "Proceedings of the third berkeley symposium on mathematical statistics and probability," 1958.

[50] D. Arthur and S. Vassilvitskii, "k-means++: The advantages of careful seeding," tech. rep., Stanford, 2006.

[51] A. Bhaskara, S. Vadgama, and H. Xu, "Greedy sampling for approximate clustering in the presence of outliers," *Advances in Neural Information Processing Systems*, vol. 32, pp. 11148–11157, 2019.

[52] M. Ester, H.-P. Kriegel, J. Sander, X. Xu, *et al.*, "A density-based algorithm for discovering clusters in large spatial databases with noise.," in *kdd*, vol. 96, pp. 226–231, 1996.

[53] H. Zhou, P. Wang, and H. Li, "Research on adaptive parameters determination in dbscan algorithm," *Journal of Xi'an University of Technology*, vol. 28, no. 3, pp. 289–292, 2012.

[54] D. A. Reynolds, "Gaussian mixture models.," *Encyclopedia of biometrics*, vol. 741, pp. 659–663, 2009.

[55] E. Schikuta and M. Erhart, "The bang-clustering system: Grid-based data analysis," in *International Symposium on Intelligent Data Analysis*, pp. 513–524, Springer, 1997.

[56] T. Iqbal and S. Qureshi, "The survey: Text generation models in deep learning," *Journal of King Saud University-Computer and Information Sciences*, 2020.

[57] Q. Zhao, *Cluster validity in clustering methods*. PhD thesis, Itä-Suomen yliopisto, 2012.

[58] H. I. Fawaz, G. Forestier, J. Weber, L. Idoumghar, and P.-A. Muller, "Deep learning for time series classification: a review," *Data mining and knowledge discovery*, vol. 33, no. 4, pp. 917–963, 2019.

[59] C. A. Ratanamahatana and E. Keogh, "Making time-series classification more accurate using learned constraints," in *Proceedings of the 2004 SIAM international conference on data mining*, pp. 11–22, SIAM, 2004.

[60] A. Bagnall, J. Lines, J. Hills, and A. Bostrom, "Time-series classification with cote: the collective of transformation-based ensembles," *IEEE Transactions on Knowledge and Data Engineering*, vol. 27, no. 9, pp. 2522–2535, 2015.

[61] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet classification with deep convolutional neural networks," *Communications of the ACM*, vol. 60, pp. 84–90, May 2017.

[62] D. Palaz, R. Collobert, *et al.*, "Analysis of cnn-based speech recognition system using raw speech as input," tech. rep., Idiap, 2015.

[63] A. M. Alayba, V. Palade, M. England, and R. Iqbal, "A combined cnn and lstm model for arabic sentiment analysis," in *International cross-domain conference for machine learning and knowledge extraction*, pp. 179–191, Springer, 2018.

[64] C.-L. Liu, W.-H. Hsaio, and Y.-C. Tu, "Time series classification with multivariate convolutional neural network," *IEEE Transactions on Industrial Electronics*, vol. 66, no. 6, pp. 4788–4797, 2018.

[65] A. Sherstinsky, "Fundamentals of recurrent neural network (rnn) and long short-term memory (lstm) network," *Physica D: Nonlinear Phenomena*, vol. 404, p. 132306, 2020.

[66] D. L. Davies and D. W. Bouldin, "A cluster separation measure," *IEEE transactions on pattern analysis and machine intelligence*, no. 2, pp. 224–227, 1979.

[67] P. J. Rousseeuw, "Silhouettes: a graphical aid to the interpretation and validation of cluster analysis," *Journal of computational and applied mathematics*, vol. 20, pp. 53–65, 1987.

[68] T. Caliński and J. Harabasz, "A dendrite method for cluster analysis," *Communications in Statistics-theory and Methods*, vol. 3, no. 1, pp. 1–27, 1974.

[69] F. Karim, S. Majumdar, H. Darabi, and S. Harford, "Multivariate LSTM-FCNs for time series classification," *Neural Networks*, vol. 116, pp. 237–245, 2019.