8-1-2024

Mining Gambling Data for Modeling Gambling Behavior Patterns

Piyush Aniruddha Puranik

MINING GAMBLING DATA FOR MODELING GAMBLING BEHAVIOR PATTERNS

By

Piyush Aniruddha Puranik

Bachelor of Engineering - Computer Engineering
Savitribai Phule Pune University
2015

Master of Science - Computer Science
University of Nevada, Las Vegas
2019

A dissertation submitted in partial fulfillment
of the requirements for the

Doctor of Philosophy - Computer Science

Department of Computer Science
Howard R. Hughes College of Engineering
The Graduate College

University of Nevada, Las Vegas
August 2024

UNLV | GRADUATE COLLEGE

Dissertation Approval

The Graduate College
The University of Nevada, Las Vegas

July 12, 2024

This dissertation prepared by

Piyush Aniruddha Puranik

entitled

Mining Gambling Data for Modeling Gambling Behavior Patterns

is approved in partial fulfillment of the requirements for the degree of

Doctor of Philosophy –Computer Science
Department of Computer Science

Kazem Taghva, Ph.D.
Examination Committee Chair

Laxmi Gewali, Ph.D.
Examination Committee Member

Mingon Kang, Ph.D.
Examination Committee Member

Fatma Nasoz, Ph.D.
Examination Committee Member

Brett Abarbanel, Ph.D.
Graduate College Faculty Representative

Alyssa Crittenden, Ph.D.
Vice Provost for Graduate Education & Dean of the Graduate College

Abstract

Understanding player behavior for responsible gambling research is a difficult task due to the lack of data on players' activities. Past studies in this area are largely limited to publicly available behavioral data or aggregated players data. Problem gambling in gamblers is typically identified only after they have already been addicted or have already been engaging in problematic gambling behavior. Furthermore, "risky" gambling behavior has historically been difficult to define due to the varying patterns of gambling activity that could potentially be attributed to it.

In this dissertation we illustrate the methodology and algorithms used to engineer financial data for further analysis. We demonstrate the use of time-series analysis and anomaly detection using statistical and unsupervised machine learning methods to detect anomalous individual player behavior. This is accomplished using a custom metric (delta index) for statistical analysis and the spectral residual algorithm for machine learning based anomaly detection. We also demonstrate the use of longitudinal clustering using Gaussian Mixture Models to identify changes in player behavior at 30 day time intervals. This method groups players based on changes in behavior over time rather than their existing behavior in any given time slice. Additionally, we demonstrate the use of this model to track individual player behavior over time.

The behavior analysis methods illustrated in this dissertation can be generalized for accepting additional behavioral features for further research in this area. Additionally, these methods can be further adapted for use in other behavioral studies.

Acknowledgements

"I am grateful to Dr. Kazem Taghva for his unwavering patience, guidance, and support throughout my journey as a Ph.D. student. His pursuit of excellence has been a source of motivation, pushing me to continually enhance my work and research methods.

To my esteemed committee members: Dr. Laxmi Gewali, Dr. Mingon Kang, Dr. Fatma Nasoz, and Dr. Brett Abarbanel, I extend the deepest appreciation for your support and the invaluable time that you have dedicated to evaluating my research. Special thanks to Kasra and Dr. Brett Abarbanel for their valuable insights into problem gambling, which have helped me realize the significance and impact of my research contributions.

My heartfelt gratitude goes to Ashkan for being not only an incredible friend but also a dedicated research colleague. To my mother and sister, who have always inspired me with their hard work and determination, I owe my deepest thanks; I would not be the person I am today without their unwavering support.

Lastly, I want to express my boundless gratitude to my wife, Amruta, who has been my rock and source of strength. Her unwavering belief in me, even in times when I doubted myself, has been a beacon of hope and encouragement. Thank you, Amruta, for always standing by my side."

Piyush Aniruddha Puranik

University of Nevada, Las Vegas

August 2024

Table of Contents

List of Tables

List of Figures

List of Algorithms

Chapter 1

Introduction

Responsible gambling as a concept is essentially regulations and guidelines that are followed by gambling service providers and governments to ensure that gambling remains a safe and enjoyable experience. The objective is to prevent gambling from turning into an addiction and ensuring players are aware of the risks involved with gambling and its addictive potential. Since the concept of "risky behavior" is difficult to define, we require some quantifiable metric or some statistical measure to identify this type of behavior. The most widely accepted method of categorizing players based on their problem gambling severity is using the Problem Gambling Severity Index (PGSI) [FW01]. This is a psychometric analysis that requires players to take a short quiz to assess their gambling. However, requiring players to take a test periodically to assess problem gambling severity is neither practical nor something that all players would agree to. In order to overcome this limitation, an automated approach to behavior evaluation may be an ideal methodology.

The primary concern regarding the application of data science and machine learning to gambling behavior is the lack of ground truth that identifies "problem gambling". Therefore training a supervised machine learning model to determine such behavior is unfeasible.

Within the defined scope of this dissertation, we address three research questions:

1. Can we extract relevant gambling behavior features from structured financial data?

2. Can we model a time-series on these extracted features to obtain insight into behavior patterns for individuals? Additionally, can we model machine learning based anomaly detection techniques to identify spikes or fluctuations in individual player behavior?

3. Can we use longitudinal clustering to identify and monitor gambling behavior patterns among groups of players?

Due to the limited availability of quantitative behavioral data on gambling financial transactions and player behavior, statistical modeling and machine learning applications in this area are severely limited. In our recent review on data science application to responsible gambling [GAP+22], we found a limited number of studies. Most of these use some form of supervised machine learning and statistical modeling on Behavioral Tracking Data (BTD) for responsible gambling. The majority of reported BTD studies are based on online European gamblers with lack of representation for offline gamblers and those in non-European jurisdictions. The dearth of publicly available data is further highlighted by the fact the BTD reports are largely based on the same dataset from online European gamblers [CG17].

This dissertation is divided into 7 chapters including this introduction.

- In Chapter 2, we discuss gambling related harm and the applications of data science and machine learning in the areas of problem gambling and responsible gambling. We also briefly discuss our findings from a recent scoping review on the applications of data science for responsible gambling.

- Chapter 3 provides a theoretical background of the tools and methods that are employed in this dissertation.

- In Chapter 4 we introduce the dataset that is used for player behavior modeling. The methods and algorithms used for cleaning and obfuscation along with a descriptive analysis of the data is also discussed here.

- In Chapter 5 we demonstrate the use of time-series for further analysis of the overall data. Subsequently, we extract player behavior features from this data and model them as a time series for identifying anomalies in player behavior. We demonstrate the use of *delta index* and the machine learning based spectral residual algorithm for anomaly detection in a time series.

- Chapter 6 demonstrates the use of longitudinal analysis using Gaussian Mixture Model clustering on a sample of players. These methods enable us to observe common be-

havior patterns among groups of players and identify changes in these patterns over time.

- Finally, in Chapter 7 we summarize our research and discuss additional avenues for further development of this area of research.

Chapter 2

Gambling Related Harm

To understand the significance of this research, we must first understand the socio-economic impact of gambling related harm. In this chapter, we define industry terms such as problem gambling and responsible gambling and how they are related to gambling related harm. We discuss methods and industry practices for gambling harm prevention and applications of data science for responsible gambling. The following three aspects of gambling are discussed in this chapter:

1. Problem Gambling (PG)
2. Responsible Gambling (RG) and gambling harm prevention
3. Data science and Machine Learning (ML) based solutions for responsible gambling and harm prevention.

2.1 Problem Gambling (PG)

Any form of repetitive gambling that negatively impacts an individual's day to day activity, mental health, and financial security is called problem gambling. This also includes gamblers who do not fulfil the threshold for some form of gambling mental disorder [AP00]. When a gambler transitions from a recreational gambler to a problem gambler, they typically require more severe forms of treatment including rehabilitation. However, problem gambling as a term only refers to people suffering from the harmful effects of excessive gambling. The

more clinically appropriate term for gamblers with a clinical gambling disorder is *pathological gambler* (not to be confused with *problem gambler*). Research shows that pathological gamblers and problem gamblers demonstrate similar sensitivity to short-term rewards and lower sensitivity to short-term losses [FBM17]. In other words, the psychological functioning of pathological and problem gamblers is different from that of the average person. Additionally, the research suggests that the motive for gambling also differs between problem gamblers and average gamblers. For example, people with relatively few or no observable symptoms of gambling disorders tend to gamble for fun, enjoy it with friends and family, or sometimes gamble for complimentary food and drinks. However, people with tendencies for severe gambling problems reportedly gamble either for financial gain or to escape other difficulties in their lives [GR20].

2.2 Responsible Gambling (RG) and Harm Prevention

To prevent gambling from turning into problem gambling, several systems and regulations to prevent problem gambling have been put in place. These systems and frameworks designed to prevent gambling related harm are broadly referred to as responsible gambling efforts. However, there is no established definition for what responsible gambling actually is [Gha22]. From a moral perspective, people who consider gambling as an activity only engaged in by the morally corrupt, deem the term 'responsible gambling' as an oxymoron [CBL$^+$15].

In 2004, a framework was proposed to define guidelines for minimizing gambling related harm called *The Reno Model* [BLS04]. This model aimed to describe a methodology for stakeholders (consumers, gambling operators, healthcare providers, governments, etc.) to address gambling related harm while still maintaining profitability. The authors emphasized that there is a lack of evidence to show that responsible gambling initiatives have shown any positive effect on problem gambling. This was their primary motive for proposing a more actionable framework for RG. This model was criticized by adversaries for focusing too strongly on individual responsibility towards problem gambling and not enough on the responsibility shouldered by the industry [HS17]. It was argued that *The Reno Model* pandered to operators and promoted regulatory-avoidance while advocating for a framework

that was backed by insufficient and contradictory evidence.

RG practices have broadly been argued as being focused on individual choice rather than on industry practice. The goal of RG is allegedly to support an individual's choice to gamble with access to resources and opportunities related to problem gambling rather than to push for regulatory reforms. Therefore, alternative frameworks to prevent gambling related harm from a public health perspective were proposed by researchers. These frameworks comprised the impact of gambling products and practices on a population rather than on individuals. Although the goal is the same, advocates of the public health approach to harm prevention argue that RG programs are not effective at mitigating problem gambling and need to be replaced with a more holistic alternative [LR20]. A study by Griffiths and Delfabbro shows that combining psychological, biological and sociological aspects of gambling provide a far better understanding for research and clinical intervention for gambling related harm [GD01]. Both RG and public health approaches provide different perspectives towards addressing gambling related harm with RG focusing more on the individual and public health focusing more on regulation and consumer protection.

Although modern consumer spending is largely dominated by cashless digital payments, land-based gambling primarily relies on cash. Since the COVID-19 pandemic, several Electronic Gaming Machines (EGM) have implemented cashless payment systems for gamblers enabling data collection and user tracking within these machines. User tracking systems present further opportunities for understanding players' gambling activity and consequently implementing systems for harm prevention [STC+23]. Player tracking can not only be utilized in understanding gambling behavior, but also motivate the implementation of regulations specifically for harm prevention [NS24].

2.3 Machine Learning (ML) for Responsible Gambling

With the modernization of payment systems used for online and land-based gambling, the use of digital payments has increased significantly. Furthermore, regulatory changes in the US have made digital gambling products more readily available to the public and to gambling providers to align with the guidelines that were put in place for the COVID-19 pan-

demic [SR21]. This opened the door to data collection and mining from active gamblers not only on online platforms but also on casino floors where digital payment systems had begun to proliferate. As player activity tracking increased, the interest and potential in utilizing this data for gambling harm prevention by researchers in data science, public health and hospitality also increased. The results for a scoping review that we performed on the use of machine learning applications for RG are summarized in this section [GAP+22]. The next two sub-sections discuss the use of machine learning in RG with the first sub-section focusing on the use of supervised ML and the second sub-section focusing on unsupervised ML. From this scoping review we found that ML studies for RG are limited in their scope as far as generalization and reproducibility are concerned.

2.3.1 Studies using Supervised ML

The scoping review showed that supervised ML based studies on problem gambler classification used a variety of different definitions for their target variable [GAP+22]. Some used 'self-exclusion' as a target variable to act as a proxy for problem gamblers. 'Self-exclusion' is a system that allows gamblers to voluntary ban themselves from any further gambling. However, self-exclusion acts as a very weak indicator of PG because problem gamblers are not guaranteed to use this system even when they are aware of being a problem gambler thereby reducing its efficacy as a target variable for PG [Gai14]. Furthermore, not all participants of the self-exclusion program are problem gamblers and may have other reasons for self-exclusion [Phi14]. Several other studies have arbitrary definitions for PG using a combination of factors such as cutoff scores on PGSI, changes in gambling limits, self-exclusion, etc.

Only one out of the reviewed supervised ML studies used quantitative data obtained from Electronic Gaming Machines (EGMs) from land-based betting offices [EBW+14]. However, the data used in this study is not publicly available. All other studies relied on Behavioral Tracking Data (BTD) based on online gamblers. This points to a lack of available data for machine learning and AI applications for RG.

Commonly used ML models used in these studies are random forest, logistic regression, neural networks, bayesian networks, and support vector machines. Some studies such as the

one by Philander [Phi14] compared 9 different supervised ML models to test their efficacy for detecting problem gambling. The author of this study acknowledged that the use of self-exclusion as a proxy for PG is inadequate at conclusively determining PG. Characteristics such as session count, game types, betting amounts, and timestamps of gambling activity were used to extract features using a variety of statistical computations applied to them. The computations include but are not limited to standard deviation, mean, frequency, and summation. The outcomes from these ML implementations were non-conclusive with either insufficient metrics to accurately represent their performance or insufficient information to accurately reproduce the models.

2.3.2 Studies using Unsupervised ML

All of the unsupervised ML studies in the scoping review, except for one, used similar BTD as the supervised ML studies. Of these studies, half employed the k-means clustering algorithm. Several other methods such as Density Based Spatial Clustering of Applications with Noise (DBSCAN), SPPS two-step clustering, Hidden Markov Model (HMM), Chi-square Automatic Interaction Detector (CHAID) were used in the other studies. The variables used for clustering did not differ greatly from the supervised ML studies, however two of the studies [PHGBCB18, CBHT$^+$20] employed Latent Class Analysis (LCA) with auxiliary variables to define the classes of players obtained from cluster analysis.

Chapter 3

Background on Machine Learning

Machine Learning (ML) is an area of artificial intelligence (AI) that specifically deals with development of algorithms to mimic the human reasoning process. Broadly speaking, there are two categories under which ML algorithms can be classified:

1. Supervised

2. Unsupervised

In Chapter 2, we discussed various applications of ML algorithms for responsible gambling and harm prevention. In this chapter, we discuss the theory behind these ML algorithms and how they can be utilized to develop AI based solutions in the gambling domain.

3.1 Supervised Machine Learning

Given a data set that consists of inputs and their corresponding outputs, a human can learn to identify patterns within these inputs that contributes to that output. Even though a direct mathematical relationship between the inputs and outputs may not exist, a human can learn to estimate an output based on prior experience showing a certain set of inputs mapped to a certain output. For example, a music connoisseur can identify different instruments in a musical concert despite the inability to mathematically quantify the difference between the sound of a piano, and that of a cello. In the case of a machine, the data containing the inputs and desired outputs is called training data [RN16]. We use this training data to

provide examples to a machine in order to train it to estimate the output for unseen set of inputs.

Mathematically, let us assume that we have an input $X = \{x_1, x_2, x_3, ...x_n\}$ that maps to an output Y. We assume that there exists a function f that can define this relationship:

$$f(X) \implies Y \tag{3.1}$$

The objective of supervised machine learning is to find this function f such that it satisfies the relationship in Equation 3.1. However, a single set of inputs and output is not sufficient for a machine to understand this relationship. A dataset with labeled data containing k inputs and their corresponding outputs can be represented as follows

$$\begin{matrix} X_1 \\ X_2 \\ X_3 \\ \vdots \\ X_k \end{matrix} \begin{bmatrix} x_{11} & x_{12} & x_{13} & \dots & x_{1n} \\ x_{21} & x_{22} & x_{23} & \dots & x_{2n} \\ x_{31} & x_{32} & x_{33} & \dots & x_{3n} \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ x_{k1} & x_{k2} & x_{k3} & \dots & x_{kn} \end{bmatrix} \begin{bmatrix} Y_1 \\ Y_2 \\ Y_3 \\ \vdots \\ Y_k \end{bmatrix}$$

Formally, we can define our entire data D as:

$$D = \{(X_1, Y_1), (X_2, Y_2), (X_3, Y_3), \dots (X_k, Y_k)\}$$

Now that we have a formal definition of what our data is, we can proceed to mathematically formalize the idea of supervised machine learning. The following is the standard definition of supervised learning following Dorian Brown's post on introduction to ML [Bro19].

Let R and C denote the input and output domain, i.e. $X \in R$ and $Y \in C$, then the our data set is:

$$D = \{(X_1, Y_1), (X_2, Y_2), (X_3, Y_3), \dots (X_k, Y_k)\} \subseteq R \times C \tag{3.2}$$

Since we are interested in a function f that approximates the relationship between X_i and Y_i, we can formally define this as:

$$f : R \implies C$$

The function f is not necessarily unique. Suppose F represents the set of all such functions; the ML is trying to find a function that minimizes the errors associated with the mapping of the input to the output. This is done using a loss function L which computes the loss for every $f \in F$. The value of L allows us to gauge the validity of the chosen f. Generally, the loss function indicates how well the chosen f reproduces the data D. Therefore, we are interested in minimizing the loss function L.

$$\min_{f \in F} \frac{1}{n} \sum_{i=1}^{k} L(X_i, f(X_i)) \tag{3.3}$$

Where $f(X_i)$ denotes the output obtained from the chosen f function. Equation 3.3 denotes the core optimization problem that signifies the learning process for any supervised ML algorithm.

3.1.1 Loss Functions

There are several different loss functions that can be applied to the optimization problem depending on the architecture of ML in specific applications. Many historical statistical learning tools such as regression use Mean Square Error (MSE) function as defined in Equation 3.4. This loss function is also called the L2 loss function and is applicable to regression based algorithms. This function calculates the arithmetic mean of the square of the difference between predicted values $\hat{Y}$ and true values Y.

$$L = \frac{\sum_{i=1}^{k} (Y_i - \hat{Y}_i)^2}{n} \tag{3.4}$$

The term $Y_i - \hat{Y}_i$ is also called *error*.

Mean Absolute Error (MAE): MAE is a variation of MSE where absolute values of the difference between the predicted and true value are calculated. This loss function is also called L1 loss and is similarly applicable to regression-based ML algorithms. This function is mathematically defined as:

$$L = \frac{\sum_{i=1}^{k} |Y_i - \hat{Y}_i|}{n} \tag{3.5}$$

The most important difference between MAE and MSE is the fact that MAE does not square the error term meaning small errors and large errors are treated equally. Due to this nature of MAE, it is well suited to noisy data since outliers will not disproportionately affect the loss function.

Huber Loss (Smooth MAE): This is a combination of the above 2 loss functions where MAE is applied when the error is high but MSE is applied when the error is low. This ensures that large errors are not heavily penalized (in case of outliers) while still maintaining the high sensitivity of MSE for errors not caused by outliers. Mathematically, the Huber Loss function is defined as:

$$L = \begin{cases} \frac{1}{2}(Y - \hat{Y})^2 & |Y - \hat{Y}| \leq \delta \\ \delta|Y - \hat{Y}| - \frac{1}{2}\delta^2 & |Y - \hat{Y}| > \delta \end{cases} \tag{3.6}$$

Where δ is a hyperparameter that determines the boundary at which an output is considered an outlier. This parameter can either be tuned iteratively or can be pre-determined based on the nature of outliers in the data.

Cross-Entropy Loss: This loss function is typically used in deep learning approaches for prediction and classification. In case of the **Binary Cross-Entropy** loss function where the target output is binary (between 0 and 1) then the following loss function is used:

$$L = -\frac{1}{k}\sum_{i=1}^{k}(Y_i \cdot \log(\hat{Y}_i) + (1 - Y_i) \cdot \log(1 - \hat{Y}_i)) \tag{3.7}$$

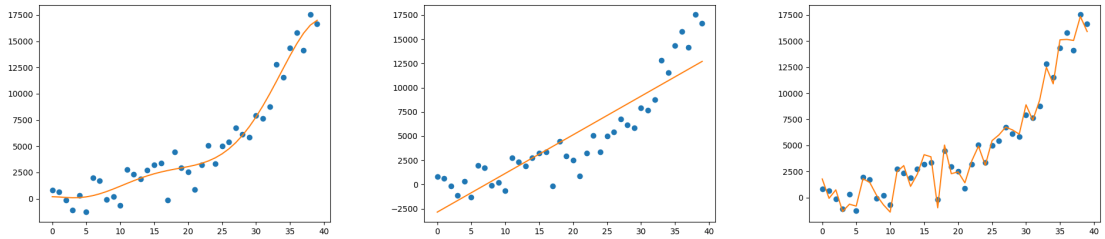
If the target is categorical and non-binary, then the loss function used is called **Categorical Cross-Entropy** and is represented as follows:

$$L = -\frac{1}{k}\sum_{i=1}^{k} Y_i \cdot \log(\hat{Y}_i) \tag{3.8}$$

3.1.2 Model Fit

A fundamental problem with this type of optimization is that it can potentially learn the data *too* well. In other words, the model will perform extremely well when given the data

that it was trained on but will do poorly when it encounters data that it has not encountered. This concept is called over-fitting. The opposite of over-fitting is called under-fitting wherein the function obtained from the optimization problem does not sufficiently represent the data.



(a) Good Fit. (b) Under Fit. (c) Over Fit.

Figure 3.1: Example of an estimated function f fitting the training data.

A visual example of this is given in Figure 3.1 where the line represents an estimated function f and the points represent the data D that the function represents. Note that the estimated f in Figure 3.1b is linear while the ones in Figures 3.1a and 3.1c are polynomial.

To prevent this phenomenon, we split D into three parts: D_{TR}, D_{VA}, D_{TE} where D_{TR} is the training set, D_{VA} is the validation set and D_{TE} is the test set. The typical split used is 80% training, 10% validation and 10% testing [Bro19]. The validation set is not strictly required for all types of supervised learning models. It is only relevant when a model has certain hyperparameters (rate of learning, training iterations, etc.) that need to be tweaked. The idea is that we test various hyperparameters for the training a model (or automatically adjust them) by using the validation set to evaluate the model performance. Hyperparameters that perform the best are selected for that specific model type. The test set is used to compare the performance of various ML algorithms when applied to a certain dataset.

K-Fold Cross Validation: Cross-validation is a technique often used in machine learning to test the efficacy of a model when used with unseen data. This is typically done to ensure that a good fit is obtained regardless of the way the dataset is split and to validate the "skill" of the model.

In this method, the data is split into K folds or segments of about equal size. One of these

segments is marked as the validation set, and the remaining $K-1$ segments are considered as the training set to train the model. This process is repeated K times, each time with a different segment being picked as a validation set [JWHT21]. Empirically, $K = 10$ has been found to be an optimal value to use [KJ13] to maintain reasonable bias and variance.

3.1.3 Artificial Neural Networks (ANN)

The idea of an Artificial Neural Network (ANN) comes from the way animal/human brains function. It consists of several interconnected artificial neurons, which propagate signals from input to output based on an activation function. This interconnected structure is typically laid out as layers with the first layer being the input layer, the last layer being the output layer, and one or many layers of neurons between these two layers called hidden layers [Dre05]. ANNs with 2 or more hidden layers are also called *deep neural networks*.

Training an artificial neural network is done by passing some input into the network and using a *loss function* to compare the output to a known output (ground truth). The value of the loss function is then used to reconfigure the neural network to obtain a more accurate output. This process is repeated iteratively until the network *converges* or stops changing.

The most commonly used loss function for ANNs is cross-entropy loss as described in Section 3.1.1. Each neuron in the ANN uses an *activation function* to compute its output. Let the input to a neuron z_i in the network be a vector $X \in \mathbb{R}$ with multiple inputs $(x_1, x_2, x_3, ...x_n)$, and the activation function used to compute the output be σl, which is a nonlinear function as defined by the following equation:

$$z_i = \sigma(W_i X + b_i)$$

Where $W \in \mathbb{R}$ is a set of weights $(w_1, w_2, w_3, ...w_n)$ and b_i is the *bias*. Both are changed iteratively through the training process. We can expand this equation as follows:

$$z_i = \sigma(\sum_{j=1}^{N}(w_{ji}x_j + b_i)) \tag{3.9}$$

Here, w_{ji} is the weight for the node i for input number j. We can think of w_{ji} as the weight being assigned to an *edge* in the example graph shown in Figure 3.2.

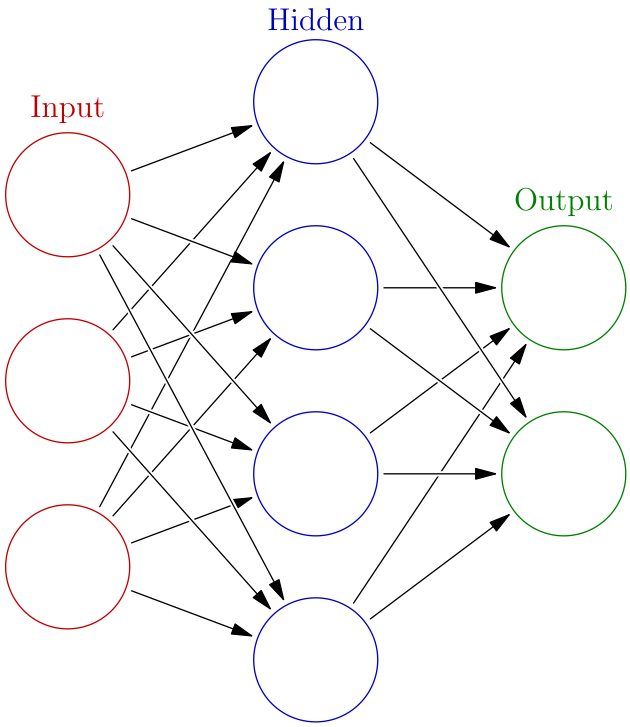


Figure 3.2: Artificial neural network with single hidden layer (by Glosser.ca on commons.wikimedia.org).

The parameters of a network represented by the weights W and biases b are formally denoted as θ. Therefore, the loss function used to calculate the error in the ANN is denoted as a function of the dataset D and the parameters θ of the network: $L(D, \theta)$.

During the iterative training process using gradient descent algorithm to find a minimum, the parameters of the network are tuned or adjusted based on the partial derivative of the loss function with respect to the parameters in the previous training iteration. We can mathematically denote this as:

$$\theta_{t+1} = \theta_t - \alpha \frac{\delta L(D, \theta_t)}{\delta \theta} \tag{3.10}$$

Where θ_t is the value of the parameters of the ANN at iteration t. Learning rate α of the ANN is a hyperparameter that is typically set before the training process begins. This can be used to control the rate at which the parameters of the network change with each iteration in the gradient descent process. Therefore, for any given weight in the network, the change in weight at each iteration can be denoted as:

$$\Delta w_{ji} = -\alpha \frac{\delta L(D, \theta)}{\delta w_{ji}} \tag{3.11}$$

This method of updating the network parameters using gradient descent is called *backpropogation* because the loss function propogates backwards from the output end of the network towards the input end. The expectation is that the parameters of the network will converge to a point that minimizes the errors. When the training is done, the network can be used to predict outputs for new input data points.

3.2 Unsupervised Machine Learning

In Section 3.1, we examine the application of machine learning for learning a mapping between a given set of inputs and their corresponding outputs using supervised learning. Unlike supervised learning, where the data has predefined labels, unsupervised learning algorithms are used to discover the inherent structure, relationships, and patterns present in the data without any explicit guidance or labeled data. The primary goal of unsupervised learning is to uncover hidden patterns, group similar data points together, or reduce the dimensionality of the data for further analysis or visualization [Rom19].

Unsupervised learning algorithms are particularly useful when dealing with large, complex datasets where manually labeling or annotating the data is impractical or infeasible. They can uncover hidden patterns, structures, and insights that may not be immediately apparent, providing a better understanding of the underlying data and supporting further analysis or decision-making processes.

3.2.1 K-Means Clustering

K-Means is an unsupervised ML method for partitioning data into k clusters based on their similarity to each other [SY20]. Similarity is measured using distance metric such as Euclidean distance between data points. The algorithm can be generally broken up into the following steps:

1. **Initialization:** The algorithm starts by randomly selecting k data points as the initial cluster centers called centroids.

2. **Assignment Step:** Each data point is assigned to the cluster whose centroid is closest

to it. The distance between a data point and a centroid is calculated using Euclidean distance or other metrics such as the Manhattan distance.

3. **Update Step:** After all data points have been assigned to clusters, the centroids of the clusters are recalculated by taking the mean of all the data points in each cluster.

4. **Iteration:** Steps 2 and 3 are repeated until the cluster assignments no longer change, or a predefined maximum number of iterations is reached.

Mathematically, the objective of k-means clustering is to minimize the sum of squared distances between each data point and its assigned cluster centroid. This objective function called Sum of Squared Errors (SSE) can be mathematically represented as follows:

$$J = \sum_{i=1}^{k} \sum_{x \in C_i} |x - \mu_i|^2$$

Where k is the number of clusters, C_i is the set of data points in the cluster i, x is a data point, μ_i is the centroid of the cluster i and $|x - \mu_i|^2$ is the squared Euclidean distance between x and μ_i. The algorithm works by iteratively minimizing this objective function. This is done by reassigning data points to their nearest centroids, and recalculating the centroids as the means of the data points in their respective clusters in every iteration.

One limitation of K-Means clustering is that it assumes that the clusters are roughly spherical and have similar variance. Therefore, this clustering method is not optimal for clusters with different shapes or distributions.

3.2.2 Clustering with Gaussian Mixture Model

A Gaussian Mixture Model (GMM) is a probabilistic model used for clustering, which assumes that the data is generated from a mixture of several Gaussian distributions with unknown parameters. Unlike K-Means clustering which assumes that the clusters are spherical and have similar variance, GMM can model clusters having different shapes, sizes, and densities [RBBL16]. Therefore, GMM is more flexible in representing non-spherical and heterogeneous clusters. Additionally, GMM also assigns each data point a probability of belonging to a certain component (cluster) allowing for uncertainty in assignments. GMM

can also handle outliers better than K-Means by assigning low probabilities to data points that are far from all Gaussian components.

In a GMM, we assume that the data points are generated from a mixture of K Gaussian distributions, where each Gaussian distribution represents a cluster. The probability density function of a GMM is given by:

$$p(x|\theta) = \sum_{k=1}^{K} \pi_k \mathcal{N}(x|\mu_k, \Sigma_k) \tag{3.12}$$

Where x is the data point, $\theta = \pi_1, \ldots, \pi_K, \mu_1, \ldots, \mu_K, \Sigma_1, \ldots, \Sigma_K$ is the set of parameters, K is the number of Gaussian components (clusters) and π_k is the mixing coefficient or weight of the k-th Gaussian component, such that $\sum_{k=1}^{K} \pi_k = 1$.
$\mathcal{N}(x|\mu_k, \Sigma_k)$ is the k-th Gaussian component with mean μ_k and covariance matrix Σ_k, given by the equation:

$$\mathcal{N}(x|\mu_k, \Sigma_k) = \frac{1}{(2\pi)^{d/2}|\Sigma_k|^{1/2}} \exp\left(-\frac{1}{2}(x-\mu_k)^T \Sigma_k^{-1}(x-\mu_k)\right)$$

Where d is the dimensionality of the data.

The goal is to find the parameters θ that maximize the likelihood of the data, given by:

$$L(\theta|X) = \prod_{n=1}^{N} p(x_n|\theta) \tag{3.13}$$

Where $X = x_1, \ldots, x_N$ is the set of N data points.

This maximization problem is typically solved using the Expectation-Maximization (EM) algorithm, which iteratively updates the parameters θ until they converge. The EM algorithm for GMM consists of two steps:

1. **Expectation Step (E-step):** In this step, the algorithm computes the probability that each data point belongs to each Gaussian component, given the current parameter estimates. This is done using Bayes' theorem:

$$\gamma_{nk} = p(z_{nk} = 1|x_n, \theta) = \frac{\pi_k \mathcal{N}(x_n|\mu_k, \Sigma_k)}{\sum_{j=1}^{K} \pi_j \mathcal{N}(x_n|\mu_j, \Sigma_j)}$$

Where γ_{nk} is the posterior probability that data point x_n belongs to the k-th Gaussian component, and z_{nk} is a latent variable that indicates the component that x_n belongs to.

2. **Maximization Step (M-step):** In this step, the algorithm updates the parameters θ to maximize the expected log-likelihood of the data, given the posterior probabilities computed in the previous step.

$$\pi_k = \frac{1}{N} \sum_{n=1}^{N} \gamma_{nk}$$
$$\mu_k = \frac{1}{N_k} \sum_{n=1}^{N} \gamma_{nk} x_n$$
$$\Sigma_k = \frac{1}{N_k} \sum_{n=1}^{N} \gamma_{nk} (x_n - \mu_k)(x_n - \mu_k)^T$$

Where $N_k = \sum_{n=1}^{N} \gamma_{nk}$ is the effective number of data points assigned to the k-th component.

The EM algorithm iterates between the E-step and M-step until convergence, typically when the log-likelihood or the parameter values change by a small or insignificant amount. After convergence, each data point is assigned to the Gaussian component with the highest responsibility γ_{nk}, effectively clustering the data.

Chapter 4

Dataset Creation and Analysis

Numerous studies have employed data science and machine learning techniques to aid in the prevention and reduction of gambling-related harm, often referred to as responsible gambling. These studies have shown promise in utilizing self-reported survey data and betting-related behavioral tracking data to model the behavior of gamblers and proactively identify at-risk groups or individuals [GAP+22]. However, the use of payment-related behavioral tracking data has been less prevalent. Traditionally, financial institutions have used financial data for fraud detection [AHM21] and customer risk assessment [CRC16] from a data analytics perspective. These methods can potentially be adapted to provide insights into problematic gambling behavior and risky financial behavior within the context of responsible gambling.

The lack of publicly available quantitative data on customer transactions has restricted academic research in the application of machine learning for problem gambling. To alleviate this issue, we obtained data from an online digital payments provider to facilitate further research in this area. In this chapter, we discuss this data in detail and the steps taken to formulate a dataset for player behavior modeling. Furthermore, we provide a descriptive analysis of the data and its contents which was published in 2023 [PTG23].

The data discussed in this chapter were obtained from a single payments technology provider (PTP) whose identity is not disclosed. The dataset consists of two types of transactions: financial and non-financial. Non-financial transactions include user account-related activities such as updating addresses, phone numbers, and other personally identifiable information associated with the customer's account. However, due to the sensitive nature of

this information, it is not included as part of our research and will not be disclosed. Instead, the contents of the data and the descriptive analysis presented here focus exclusively on the financial transactions found within the data.

We can describe a Payments Technology Provider (PTP) as an entity that serves as a facilitator, connecting merchants with a customer's financial institutions, such as banks or credit cards. This relationship is better visualized in Figure 4.1.

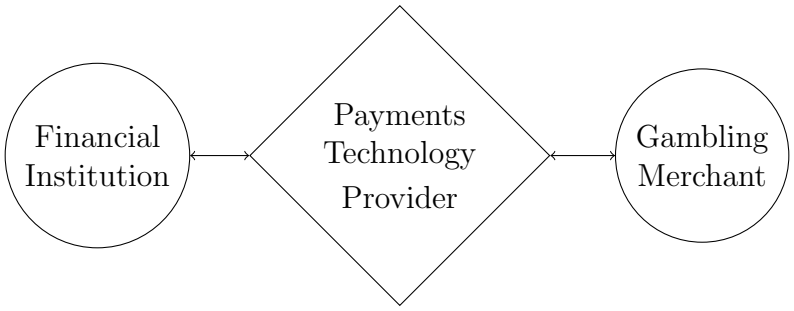


Figure 4.1: Illustrating the role of PTP in financial transactions.

4.1 Technology Tools

We leverage Apache Spark for data operations and processing and Apache Parquet for data storage. In this section we provide a description of these technologies and the motivation behind using them for data cleaning, data obfuscation and dataset creation for big data.

4.1.1 Apache Spark

Apache Spark is a framework built in 2009 at the University of California, Berkeley to deal with the computational challenges of big data. It leverages Resilient Distributed Datasets (RDD) and Map-Reduce to distribute a computation workload over a cluster of multiple nodes. The flavor of Apache Spark used in this research is Hadoop which was developed by Yahoo. The Apache Hadoop framework is an open-source framework for distributed computing which implements various systems such as a Hadoop distributed file system (HDFS) for data storage, Hadoop YARN for job scheduling and resource management, and Hadoop MapReduce which is a computation paradigm in the distributed environment. Apache Spark

abstracts the Apache Hadoop framework to provide a general-purpose API to replace traditional data manipulation pipelines with distributed computing [ZXW+16].

One of the key features of Spark is its ability to perform in-memory computing. By storing data in memory instead of reading from the disk every time, Spark can process data much faster than traditional big data processing frameworks like Apache Hadoop MapReduce. This in-memory capability allows Spark to efficiently perform iterative algorithms, such as those used in machine learning and graph processing, which require multiple passes over the same data [KKWZ15]. Additionally, it also facilitates data cleaning, transformation, and aggregation for big data, which is what we leverage to formulate a dataset from this data.

4.1.2 Apache Parquet

Apache Parquet is an open-source, column-oriented data storage format designed for efficient storage and retrieval of large datasets. Its use in big data processing is primarily preferred due to its high performance in data storage and retrieval as compared to other row-oriented formats such as CSV and JSON [Kie17].

Due to the storage of columns in contiguous memory locations, Parquet allows efficient column wise operations and better compression owing to the singular datatype requirement for each column in structured data [Voh16]. Parquet's file structure consists of three main components:

1. **Header:** Contains metadata about the file, such as the version, schema, and column information.

2. **Row groups:** The data is divided into row groups, which are horizontal partitions of the data. Each row group contains a subset of the rows and is stored independently.

3. **Column chunks:** Within each row group, the data is further divided into column chunks. Each column chunk contains the values for a specific column within the row group.

A simplified illustration of the parquet file structure is given in the following Figure 4.2

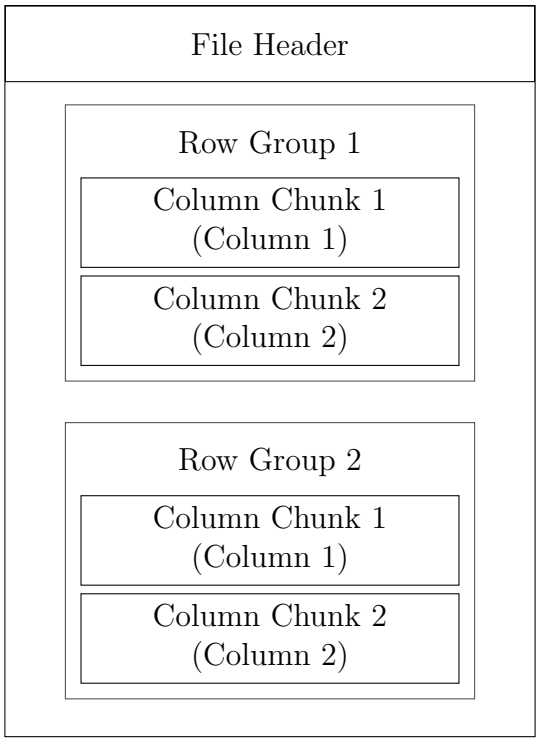


Figure 4.2: Illustration of the parquet file structure as a block diagram.

This structure allows for efficient reading and writing of data, as well as enabling parallel processing of row groups across multiple nodes in a distributed computing environment.

4.2 Data Cleaning & Obfuscation

In this section we describe the methods and technologies used for obfuscating and cleaning the data before formulating a dataset to be used for analysis.

Initially, this data was obtained in the form of comma-seperated values (CSV) files. The data contained personally identifiable information related to customers and merchants which was only partially sanitized. Additionally, numerous invalid and null entries were also found. Cleaning and analysis of this data was done on an Apache Spark cluster at the University of Nevada, Las Vegas, after converting and storing the data in Apache Parquet format.

4.2.1 Conversion to Parquet

The data was split into several CSV files containing extraneous lines in the beginning of the file, which broke compatibility with the CSV format. While analyzing the files, we found that there were exactly 7 rows of extraneous lines, which were first removed using Algorithm 1.

Algorithm 1: Clean transaction CSV

Input : input_file.csv
Output: Number of rows cleaned

```
1  CleanTransactionFile(input_file)
2  counter ← 0;
3  open input_file in read mode as infile;
4  reader ← Open infile with CSV Reader;
5  index ← 7;
6  open "output.csv" in write mode as outfile;
7  writer ← Open outfile with CSV Writer;
8  foreach row in reader do
9      counter ← counter + 1;
10     index ← index - 1;
11     if index > 0 then
12         continue;
13     end
14     else if index = 0 then
15         writer.writerow(header) ;        # header is a list of column headers.
16         continue;
17     end
18     writer.writerow(row);
19 end
20 return counter;
```

Subsequently, each cleaned output file is then converted into parquet format using the Spark API using Algorithm 2. Note that since there are multiple CSV files, each file is first imported as a Spark dataframe and then concatenated using the union operation:
$CSV_1 \cup CSV_2 \cup \ldots CSV_n$

4.2.2 Data Contents

On performing a preliminary analysis of the data we found the following information:

- The recorded data contains 53,445,398 financial transactions of 251,043 unique cus-

Algorithm 2: Convert and combine CSV to Parquet

Input : List of CSV Files
Output: Parquet File

```
ConvertToParquet(input_files)
df ← None ;
foreach file in input_files do
    if df = None then
        df ← Read file into Spark dataframe ;
    end
    else
        df1 ← Read file into Spark dataframe ;
        df ← df ∪ df1
    end
end
df.write("output.parquet")
```

tomers and 206 unique merchants between January 01, 2015 to December 31, 2021.

- Out of these, 51,542,349 transactions were approved, 1,886,572 transactions were declined, 15,648 transactions failed due to technical issues, and 829 transactions were corrupted due to system errors. Corrupted data entries were automatically discarded from the data by the CSV Reader from Algorithm 1, using exception handling.

- There are a total of 24,182,509 transactions labeled as deposits to the PTP digital wallet and 28,954,493 transactions labeled as withdrawals from the digital wallet. Withdrawals in this case can be assumed to be deposits to a gambling merchant. 308,396 transactions were wallet balance queries.

The columns present in the data are described in Table 4.1.

4.2.3 Data Obfuscation

The `MerchantID` and `AccountID` columns in the obtained data contained personally identifiable information that could be used to directly identify merchants and PTP account holders, respectively. To create a usable and publishable dataset, it was necessary to obfuscate this data. This was accomplished using a combination of SparkSQL and PostgreSQL.

Table 4.1: Data columns and their descriptions

Column	Description
ReqTimeUTC	Timestamp of transaction
MerchantID	Unique merchant ID
ServiceMethod	Indicates, deposit/withdrawal/balance check
AccountID	Unique customer account ID
MerchantTransactionID	Unique transaction ID assigned by merchant
EncodedTransactionID	Unique transaction ID assigned by PTP
TransactionType	Indicates credit or debit for loyalty card
TransactionAmount	Amount of transaction
Status	Approved/Declined/Failed/Unknown
ErrorMessage	Error shown for Declined/Failed/Unknown Status
FriendlyMessage	Human readable version of ErrorMessage

Data is stored as a table called `servicelog`. The following steps were performed to obfuscate the data:

1. Create a table containing the account IDs and the obfuscated account ID of the players in the original data. The SQL queries for this step are as follows:

```
1 CREATE TABLE accountid(
2   account_id    SERIAL PRIMARY KEY,
3   accountidentifier VARCHAR(100) );
4 INSERT INTO accountid(accountidentifier) SELECT accountidentifier FROM
  servicelog GROUP BY accountidentifier;
```

   Note that the `account_id` column is an auto-generated column that starts at 1 and increments by 1 every time a new entry is added to `accountidentifier`. This `account_id` column acts as the obfuscated ID.

2. Create a similar table for merchant IDs as the previous step. The SQL queries used are as follows:

```
1 CREATE TABLE merchantcode(
2   merchantid   SERIAL PRIMARY KEY,
3   merchantcode CHAR(20) );
4 INSERT INTO merchantcode(merchantcode) SELECT merchantid FROM
  servicelog GROUP BY merchantid;
```

In this table, the `merchantid` column acts as the obfuscated ID for the corresponding merchant ID in the original table.

3. Create a new table `servicelog_public` from the original `servicelog` table schema but using the new obfuscated IDs as a foreign key:

```
1  CREATE TABLE servicelog_public (
2    reqtimeutc      TIMESTAMPTZ,
3    merchantid      INTEGER,
4    servicemethod   VARCHAR(100),
5    accountid       INTEGER,
6    merchanttransactionid VARCHAR(100),
7    encodedtransactionid VARCHAR(100),
8    transactiontype   VARCHAR(50),
9    transactionamount  NUMERIC,
10   status      VARCHAR(50),
11   errormessage   VARCHAR(256),
12   friendlymessage   VARCHAR(256),
13   FOREIGN KEY (merchantid) REFERENCES public.merchantcode (merchantid)
14   ON DELETE CASCADE,
15   FOREIGN KEY (accountid) REFERENCES public.accountid (account_id)
16   ON DELETE CASCADE );
```

4. Now we insert data into the new table while joining it with the ID obfuscation tables to generate a table with no personally identifiable information.

```
1  INSERT INTO servicelog_public
2    SELECT reqtimeutc,
3         merchantid,
4         servicemethod,
5         account_id,
6         merchanttransactionid,
7         encodedtransactionid,
8         transactiontype,
9         transactionamount,
10        status,
11        errormessage,
12        friendlymessage
13   FROM servicelog
14   INNER JOIN merchantcode ON merchantcode.merchantcode = servicelog.
     merchantcode
15   INNER JOIN accountid ON accountid.accountidentifier = servicelog.
     accountidentifier
16   ORDER BY reqtimeutc;
```

The final `servicelog_public` table is what we use for dataset creation. This table is exported to JSON format [PRS+16] for easy portability to other systems.

4.3 Descriptive Analysis

To understand the characteristics of the data, we performed and published a descriptive analysis of the whole dataset [PTG23]. This descriptive analysis comprises the distribution of the data, cash flow patterns and values, and stationarity analysis to identify any seasonality patterns in the overall population of the dataset.

From the descriptive analysis, we found 3 main features that can be used to further engineer relevant features for a machine learning approach to modeling:

1. **Approved/Declined Transactions:** The dataset comprises both approved and de-

clined transaction. This information holds value not only for isolating approved transactions but also for discerning patterns behind the declination of transactions.

2. **Aggregated Value of Transactions:** The value of transactions conducted over an arbitrary period of time facilitates the identification of spending patterns for groups of customers. Here, the value of the transaction refers to its monetary value indicating a cash flow. Since we are only interested in cash flow, declined or failed transactions are disregarded.

3. **Frequency of Transactions:** Frequency refers to the number of transactions conducted over an aggregated period of time. For example, considering the frequency of transactions over a month can provide insight into consumer activity within that month for the PTP.

4.3.1 Distribution

By plotting the frequency of transactions per month from 2015 to the end of 2021 (entire period of the data) in Figure 4.3, it is evident that most of the data in this dataset is present in the period after 2019. This indicates that the PTP was either less active in the period before 2019, or that this specific PTP's business grew in 2019.

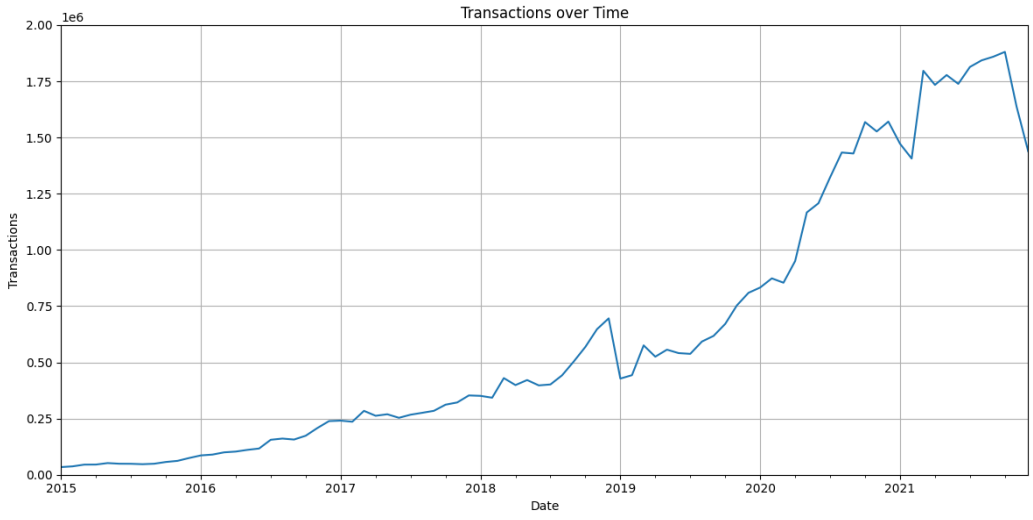


Figure 4.3: Frequency of transactions over the entire dataset [PTG23].

Of all approved transactions higher than 0, 90% of the transactions were under $200. The percentile figures for these transactions are illustrated in Table 4.2a.

Table 4.2: Percentile figures for all approved transactions [PTG23].

(a) Before outlier removal.

Index	**Amount (USD)**
Minimum	0.01
10%	10.00
25%	20.00
50%	50.00
75%	100.00
90%	200.00
Maximum	100000.00

(b) After outlier removal.

Index	**Amount (USD)**
Minimum	0.01
10%	10.00
25%	20.00
50%	40.00
75%	94.00
90%	108.95
Maximum	220.00

The maximum transaction amount is $100,000, while the minimum is $0.01, indicating that although the transactions have a very wide range, the majority of transactions are still under $200. In order to better understand this distribution, we use the interquartile range rule to find and remove outliers from the data [Ros17]. The data after outlier removal is represented in Table 4.2b. This does not mean that the outliers in this data are irrelevant; only that removing outliers in this specific scenario better represents the majority population in this dataset. This is also evident from the histogram presented in Figure 4.4, which shows that the majority of transactions in the dataset are in the $10 to $30 range.

In the following Chapter 5 we continue the analysis of the data from the perspective of a time-series.

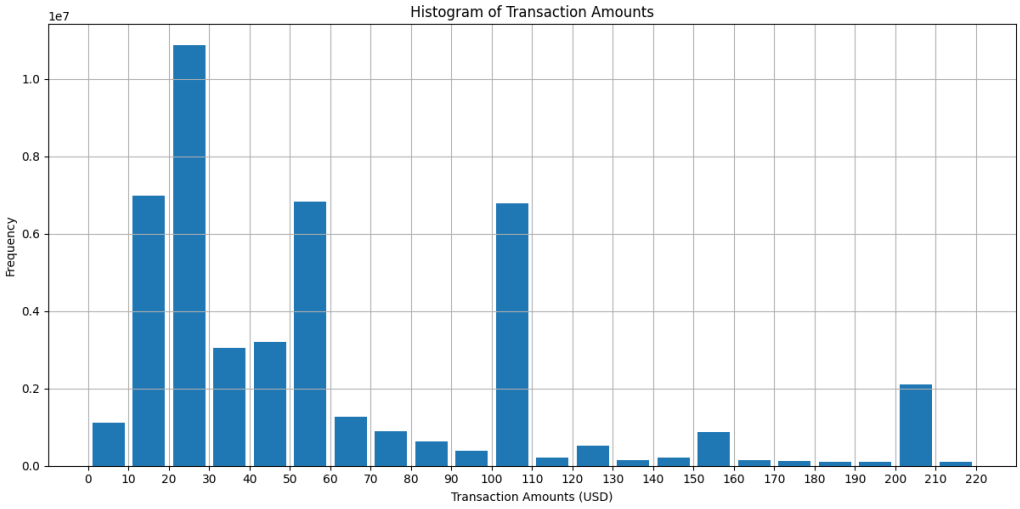


Figure 4.4: Histogram of transactions amounts without outliers [PTG23].

Chapter 5

Time Series Modeling

From Figure 4.3 it is evident that this dataset can be viewed as a time series. A time series is a sequence of data points indexed by time, typically measured at successive points in time spaced at uniform intervals. It is a collection of observations made sequentially over time. Mathematically, a time series can be denoted as a stochastic process y_t, where t represents the time index and y_t represents the observed value at time t.

$$\{y_t\}_{t=1}^{T}$$

We can also define a time series as a mapping between time domain and real number domain as follows:

$$x : T \to \mathbb{R}^k$$

where $T \subseteq \mathbb{R}$ and $k \in \mathbb{N}$ [Auf21].

Time series analysis helps describe and understand the behavior of data over time which makes it widely applicable in various fields such as finance, economics, engineering, environmental sciences and many others. Following are some historical and current examples of the uses of time series analysis.

- **Forecasting:** Time series analysis is extensively used for forecasting future values based on the patterns and trends observed in historical data. This is particularly useful in areas such as
  - Finance for forecasting stock prices, exchange rates and economic indicators [Tsa05].

  - Retail for forecasting sales and demand for inventory management [LVRK20, Win60].
  - Energy for forecasting energy consumption and production [PR22].

- **Signal processing:** Time series analysis is used in signal processing for tasks such as noise removal, signal denoising, and signal compression [SM+05] particularly in fields like telecommunications, audio/video processing, and radar systems.

- **Process Control:** In manufacturing and industrial processes, time series analysis is used to monitor and control process variables over time. By analyzing the patterns and deviations in the data, corrective actions can be taken to keep the process within desired limits [FMG+23, OSBJ20].

- **Environmental Studies:** Time series analysis is used to study and model environmental phenomena such as air pollution, water quality, and climate change patterns based on historical data [DAS20, MPAM22, SSC23].

- **Econometrics:** Time series analysis is a fundamental tool in econometrics for analyzing and modeling economic data, such as GDP, inflation, and unemployment rates [FH21, LL22].

- **Financial Risk Management:** Time series analysis is used in risk management for modeling and forecasting financial risks, such as value-at-risk (VaR) and expected shortfall [KBHG24].

In this chapter we discuss the use of time-series analysis and anomaly detection to understand the overall structure of the PTP data and to identify anomalous behavior patterns in individual gamblers.

5.1 Stationarity of Data

Time series can be classified into two distinct types based on their behavior and characteristics:

1. **Stationary Time Series:** A time series is considered stationary if its statistical properties, such as mean and variance, remain constant over time. Mathematically, a stationary time series can be represented as:

$$y_t = \mu + \varepsilon_t$$

where μ is the constant mean, and ε_t is a random error term (or white noise) with a normal distribution with mean of zero and constant variance σ^2.

2. **Non-stationary Time Series:** A time series is non-stationary if its statistical properties change over time. Non-stationarity can manifest in various forms, such as trend, seasonality, or a combination of both.

   (a) **Trend:** Trend: A time series exhibits a trend if there is a consistent upward or downward pattern over time. The trend component can be modeled using a function of time, such as a linear or polynomial function.

$$y_t = f(t) + \varepsilon_t$$

   where $f(t)$ represents the trend component.

   (b) **Seasonality:** A time series exhibits seasonality if there are recurring patterns or fluctuations within a fixed period (e.g., yearly, quarterly, or monthly). The seasonality component can be modeled using periodic functions, such as sine or cosine waves.

$$y_t = S_t + \varepsilon_t$$

   where S_t represents the seasonal component.

In the case of our data set, we want to measure if there are any data points that are susceptible to variations due to trends. A time-series may also be stationary around a deterministic trend and can therefore be made stationary by removing this trend.

5.1.1 Augmented Dickey-Fuller (ADF) Test

The ADF test is a statistical test used to determine if a given time series data is stationary or not [Mus11]. It is based on the following regression equation:

$$\Delta y_t = \alpha + \beta t + \gamma y_{t-1} + \sum_{i=1}^{p-1} \delta_i \Delta y_{t-i} + \epsilon_t \tag{5.1}$$

Where yt is the time series being tested, $\Delta y_t = y_t - y_{t-1}$ is the first difference of the series, t is the time trend, p is the number of lagged difference terms, α is the constant term also known as drift, and ϵ_t is the error term.

The null hypothesis of the ADF test is that the time series is non-stationary (has a unit root), which is represented by $\gamma = 0$ in Equation 5.1. The alternative hypothesis is that the series is stationary, or $\gamma < 0$. We calculate the test statistic using the following equation:

$$\text{ADF} = \frac{\gamma}{\text{se}(\gamma)}$$

Where $\text{se}(\gamma)$ is the standard error of the estimated coefficient γ.

The calculated ADF statistic is then compared to critical values from the Dickey-Fuller distribution. If the ADF statistic is less than the critical value, we reject the null hypothesis and conclude that the time series is stationary.

5.1.2 Kwiatkowski–Phillips–Schmidt–Shin (KPSS) Test

Unlike the ADF test, which tests the null hypothesis of non-stationarity (unit root), the KPSS tests the null hypothesis of stationarity [KPSS92]. The KPSS test is based on the following equation:

$$y_t = \beta_t + r_t + \epsilon_t \tag{5.2}$$

Where y_t is the time series being tested, β_t is a deterministic trend (a constant or a constant plus time trend), r_t is a random walk: $r_t = r_{t-1} + u_t$, with $u_t \sim iid(0, \sigma_u^2)$, ϵ_t is a stationary error term.

The null hypothesis of the KPSS test is that the series is stationary, which implies that $r_t = 0$. The alternative hypothesis is that the series is non-stationary, or $r_t \neq 0$. The

following equation is used to calculate the test statistic:

$$\text{KPSS} = \frac{\sum_{t=1}^{T} S_t^2}{T^2 \hat{\sigma}_\epsilon^2}$$

Where $S_t = \sum_{i=1}^{t} \hat{r}_i$ is the partial sum of the residuals $\hat{r}_t$ from the regression of y_t on a constant or a constant and time trend, T is the sample size, and $\hat{\sigma}_\epsilon^2$ is an estimator of the long-run variance of ϵ_t. This can be computed using any non-parametric estimator.

The calculated KPSS statistic is then compared to critical values from the KPSS distribution. If the KPSS statistic is greater than the critical value, we reject the null hypothesis and conclude that the time series is non-stationary. The KPSS test complements the ADF test, as it tests the opposite null hypothesis. Using both tests can provide more robust conclusions about the stationarity of a time series.

5.1.3 Results of ADF & KPSS

For testing the stationarity of the the distribution in Figure 4.3, we apply both the ADF and KPSS tests and compute the p-value for both. We use a threshold value of 0.05 for rejecting the null hypothesis for both tests. The p-value, test statistic and critical values for both tests are given in Table 5.1.

Table 5.1: ADF and KPSS results for frequency of transactions [PTG23].

(a) ADF Test

Index	**Value**
Test Statistic	-0.299447
p-value	0.925583
Lags Used	1.000000
No. of Observations	82.000000
Critical Value (1%)	-3.512738
Critical Value (5%)	-2.897490
Critical Value (10%)	-2.585949

(b) KPSS Test

Index	**Value**
Test Statistic	0.313977
p-value	0.010000
Lags Used	5.000000
Critical Value (10%)	0.119000
Critical Value (5%)	0.146000
Critical Value (2.5%)	0.176000
Critical Value (1%)	0.216000

The p-value for the ADF tests is 0.926 which means we can safely accept the null hypothesis that the time-series plot of the data has a unit-root and is non-stationary. Additionally, the p-value for the KPSS test is 0.01 which also suggests that the data is not stationary around a constant mean or a deterministic trend.

5.2 Player Behavior as Time Series

Previously, we discussed the problem of defining problem gambling due to the lack of quantitative evidence or metrics to define it. In this section, we discuss the modeling player behavior variables or features as a time series. By applying statistical and machine learning based analytical tools, we observe changes in player behavior and identify actionable anomalous characteristics.

5.2.1 Univariate Behavior Features

A univariate time series consists of a single variable or quantity measured over time. Modeling behavior variables as a time series enables observing changes in the variable with respect to time. This also allows us to detect trends and anomalies in the variable. From the variables available in the data as illustrated previously in Table 4.1, there are certain distinctive features that can be analyzed using time series. For this research we focus on three of these features to demonstrate the use of time series analysis.

Table 5.2: Features from PTP for time series analysis.

Feature	**Description**
Timestamp	This is obtained from ReqTimeUTC and indicates the time that the transaction was recorded.
Service Method	Indicates whether a transaction was a withdrawal or deposit which acts as a proxy for bets and winnings.
Transaction Amount	Indicates the value of the transaction, i. e. money transferred (bet or winning).

These three features from the data can allow use to model a time series for net loss, frequency of betting and transaction amounts. However, before we attempt to plot these variables as a time series, we need to address a fundamental problem:

Problem: A time series is a sequence of data points in chronological order typically taken at equally spaced time intervals (e.g. sensor readings, stock market data, audio/video signals). However, our data does not have equally spaced transactions that can be plotted as a continuous time series.

Solution: To make our data points continuous, we can aggregate them by equally spaced time periods such as by day or by month. For example, we cane aggregate the data for any arbitrary player by day, to have daily readings of their betting behavior. On days that the player does not play, the aggregate number of transactions and monetary values of the transactions can be considered to be zero. Similar aggregation can be performed by week, month, year, or any desired period.

The solution is implemented using the Rapids library. The Rapids library, developed by NVIDIA, is an open-source suite of software libraries and APIs designed to execute end-to-end data science and analytics pipelines entirely on GPUs. Its goal is to accelerate data science workflows using the power of GPUs, offering substantial speedups over CPU-based implementations [Tea23]. Specifically, we employ the cuDF component of Rapids that is designed for loading, joining, aggregating, filtering and manipulating data by leveraging the Nvidia CUDA toolkit.

Algorithm 3: Read dataset and filter player data.

Input: "dataset.json", accountID
Output: Player Dataframe

```
# Read dataset
df ← DaskDataframe(dataset, blocksize=2^28, orient='records') ;
# Filter player by accountID
filtered ← df WHERE df.accountid = accountID;
filtered ← filtered WHERE filtered.servicemethod = 'WITHDRAWAL' AND
  filtered.status = 'APPROVED' ;
# Filter features
playerDF ← SELECT 'reqtimutc', 'transactionamount' FROM filtered ;
```

Algorithm 3 describes the process for extracting a single player's data from the dataset. Note that the pseudocode for filtering and column selection has been converted to equivalent SQL queries for readability. The algorithm first reads the dataset from the json file, then filters it by `accountID` to get the intended player, and finally filters the features that are relevant for plotting a time series. Once we have the player data, we first need to aggregate the data by a fixed time period. In this case, we shall aggregate by day with all transaction

amounts summed together and number of transactions conducted in a day counted.

Algorithm 4: Aggregate data and impute values.

Input: Player Dataframe
Output: Aggregated Dataframe

```
1 agg_data ← PlayerDF.groupby(PlayerDF['reqtimeutc'].year,
    PlayerDF['reqtimeutc'].month,
    PlayerDF['reqtimeutc'].day).aggregate('reqtimeutc':count,
    'transactionamount':sum);
  # Rename Columns
2 agg_data.rename_columns('reqtimeutc':'freq', 'transactionamount':'debitamt');
  # Create new column for date
3 agg_data['date'] ← agg_data.year + agg_data.month + agg_data.day;
  # Select date, freq and amt columns
4 agg_data ← agg_data['data', 'debitamt', 'freq'];
5 agg_data.sort_by ('date');
```

Algorithm 4 first aggregates a player by year, month, and day which is extracted from the `reqtimeutc` column. This column is formatted as a timestamp and allows addressing of each individual time component. While using the `groupby` method, we define the aggregate functions to be applied to the `reqtimeutc` and the `transactionamount` columns. In this case, we want to count the `reqtimeutc` column to compute the number of transactions that are conducted in a day, and sum the `transactionamount` column to get the total amount of money spent in the day. An example of the data generated by Algorithm 4 for an arbitrary player is shown in Table 5.3.

Table 5.3: Example output for player aggregation using Algorithm 4

date	**debitamt**	**freq**
2015-11-25	30.0	2
2016-01-16	80.0	4
2016-01-17	150.0	9
2016-01-18	250.0	12
2016-01-19	20.0	1
2016-01-20	150.0	7
2016-01-21	110.0	8
⋮	⋮	⋮

Note that in this output, there are no records for days when there is no player activity. However, to plot this data as a time series, we require readings at equally spaced intervals. To do this, we impute the values for missing days to be zero (no amount debited and no transactions made) using Algorithm 5. We only consider 4 years or 365×4 days of data for this time series.

Algorithm 5: Impute missing values for aggregate data.

Input: AggregateData
Output: Periodic AggregateData

```
1  start_date ← agg_data.index(0); # First row
   # Generate array of dates
2  foreach d in range(start_date + 4 * 365) do
3  |  new_index.append(d);
4  end
5  Convert new_index to Pandas Index;
6  agg_data.reindex(new_index);
   # Insert 0 in place of null
7  agg_data['debitamt'].fillna(0);
8  agg_data['freq'].fillna(0);
```

The resultant data from Algorithm 5 now can be considered as a time series with equally spaced time intervals. If we apply this algorithm to the example from Table 5.3, then a sample of the resultant output is given in Table 5.4. Note that days between 2015-11-25 and 2016-01-16 with no activity have been imputed.

Table 5.4: Example output for imputation using Algorithm 5

date	**debitamt**	**freq**
2015-11-25	30.0	2.0
2015-11-26	0.0	0.0
2015-11-27	0.0	0.0
2015-11-28	0.0	0.0
2015-11-29	0.0	0.0
2015-11-30	0.0	0.0
2015-12-01	0.0	0.0
⋮	⋮	⋮

The example in Table 5.4 does not show all values since the resultant output is quite

large. A plot of this example has been illustrated in Figure 5.1. In this example plot we are considering a single variable, i. e. the debit/withdrawal amount of the player per day.

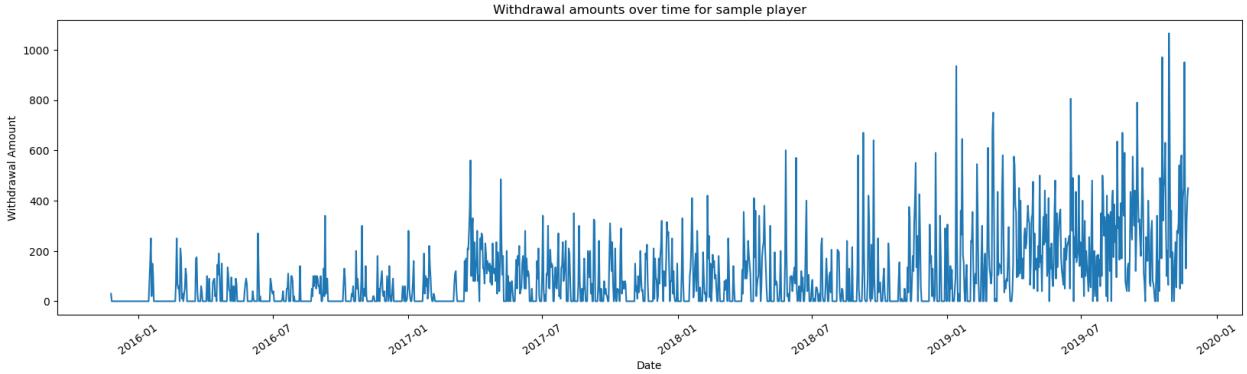


Figure 5.1: Example plot of debit amount for output of Algorithm 5.

We can further redefine the aggregation to weekly, monthly, yearly or any arbitrary time period using the *Grouper* function provided by Pandas [pdt23]. This is useful to identify general trends or the overall behavior of a player for specific time periods. For example, if we plot behavior by monthly aggregation, we can more easily observe trends in the behavior.

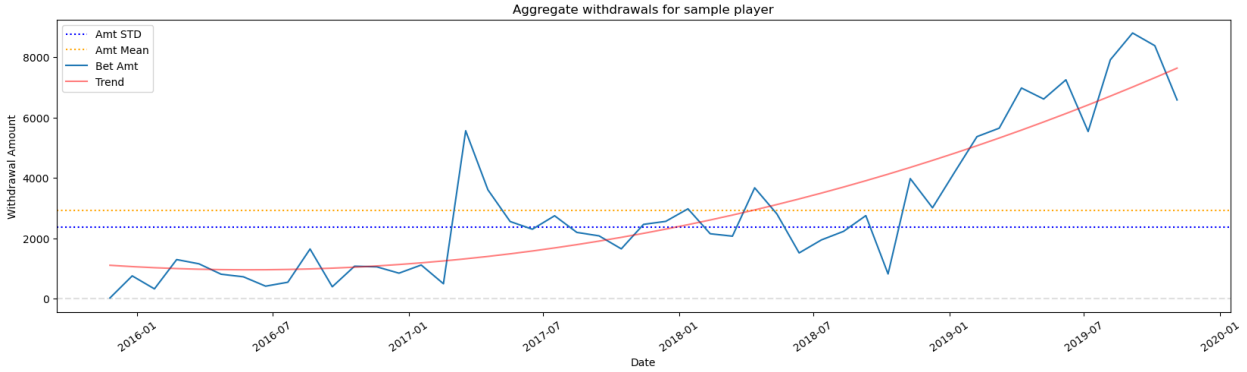


Figure 5.2: Example plot of withdrawals aggregated monthly.

The plot in Figure 5.2 shows the monthly aggregated withdrawals for the sample player observed previously in Figure 5.1. In this figure we can clearly observe a trend in this variable which is not evident in Figure 5.1. The trend here is calculated using the least squares polynomial fit algorithm with a specified degree of 2. Spikes in this graph also visually indicate drastic changes in player behavior.

5.2.2 Anomaly Detection with Delta Index

To establish a statistical baseline for change in player behavior, we define a metric called *Delta Index*. The mathematical representation of delta index is as follows.

$$\text{delta\_index} = \frac{\delta v_t}{v_{t-1}} \tag{5.3}$$

Where δv_t is

$$\delta v_t = v_t - v_{t-1}$$

Delta index is used to compute the rate of change of a variable without the amplitude scaling of the observed variable having a significant impact on the result. In other words the range of delta index is not dependent on the magnitude of the domain. The range only represents the factor by which the input variable has changed. For example, we can plot the delta index of the monthly aggregate withdrawals of a player to observe significant changes or spikes in the variable.

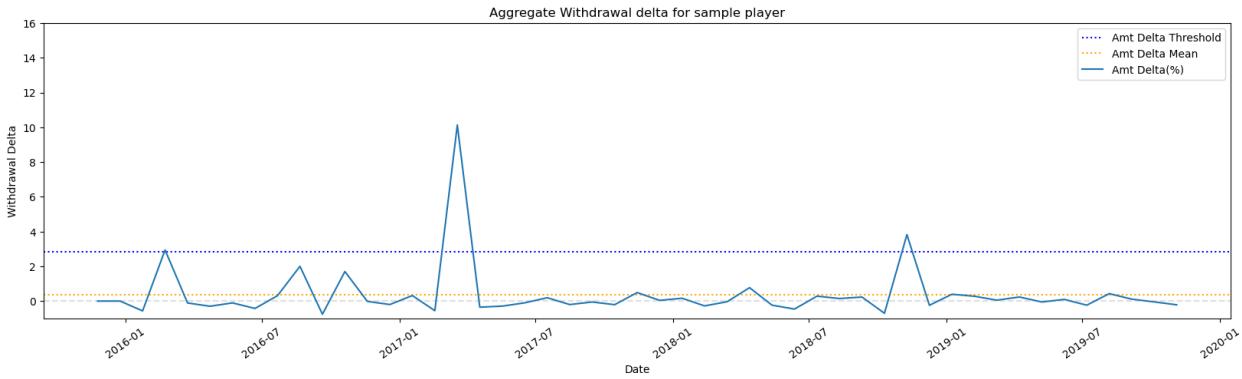


Figure 5.3: Delta index plot of the withdrawals of a sample player.

Figure 5.3 shows the delta index of the monthly aggregated withdrawals of a sample player that was previously observed in Figure 5.2. The `Amt Delta Threshold` line in this plot indicates the threshold at which delta index is considered significantly different from the normal plot. In this example, we consider the threshold to be $1.5 * \sigma_{DI}$ where σ_{DI} is the standard deviation of the delta index.

The following steps are used to implement this method of anomaly detection for the PTP dataset:

1. **Sampling:** For this experiment, we consider 2 years of player activity from January 2019 to December 2020 (inclusive). From this time period, we extract the top 10% of players based on transaction frequency over this entire period. The motivation behind selecting this time period is based on the observation in Figure 4.3 where the years 2019 and 2020 show maximum player activity. We consider only the top 10% of gamblers in order to only account for frequent gamblers from this dataset. This is based on the assumption that 90% of the players using the payments platform provided by PTP are not frequent gamblers.

2. **Aggregation:** We apply Algorithm 4 and Algorithm 5 to generate a time series of each player's daily activity.

3. **Normalization:** From preliminary analysis, we found that normalizing the data results in a more consistent result for threshold calculation. Therefore, we normalize the desired variable to a value between 0 and 3. The normalized value for any given variable v at time t is given by:

$$\text{Normalized } v_t = \frac{v_t - v_{min}}{v_{max} - v_{min} + \alpha} \tag{5.4}$$

   Where v_{min} is the minimum value of v for the player in the stipulated time frame, v_{max} is the maximum value, and α is a very small number (typically 10^{-5}) added to the denominator to prevent divide by zero error.

4. **Delta Index (DI):** In order to compute DI based on Equation 5.3 we need to know v_{t-1} for the desired variable. For example, the DI of withdrawal amount w_t for any given time instant t for a player is given by:

$$w_{DI} = \frac{w_t - w_{t-1}}{w_{t-1}}$$

   From this equation, it is clear that we need to get w_{t-1} for each data point in the time series. However, if w_{t-1} is 0 then the above equation will fail due to divide by zero error. Therefore, we use the following equation to compute DI:

$$w_{DI} = \begin{cases} \frac{w_t - w_{t-1}}{w_{t-1}} & w_{t-1} > 0 \\ w_t & w_{t-1} = 0 \end{cases} \tag{5.5}$$

5. **Anomaly Detection:** After computing the delta index for the variable, we set a threshold at which the data point is considered an anomaly. For this experiment, we use the standard statistical method of $\mu + (1.5 * \sigma)$ to determine the anomaly threshold. We label all data points in the original time series as anomalies where the delta index at that point exceeds the established threshold. This results in an output with 3 columns: timestamp, DI of variable, and label. The label is a boolean with true for anomaly, and false otherwise.

For a random player selected from the pool of sampled players, we can plot a time series for any given behavior variable with the overlaid anomalies.

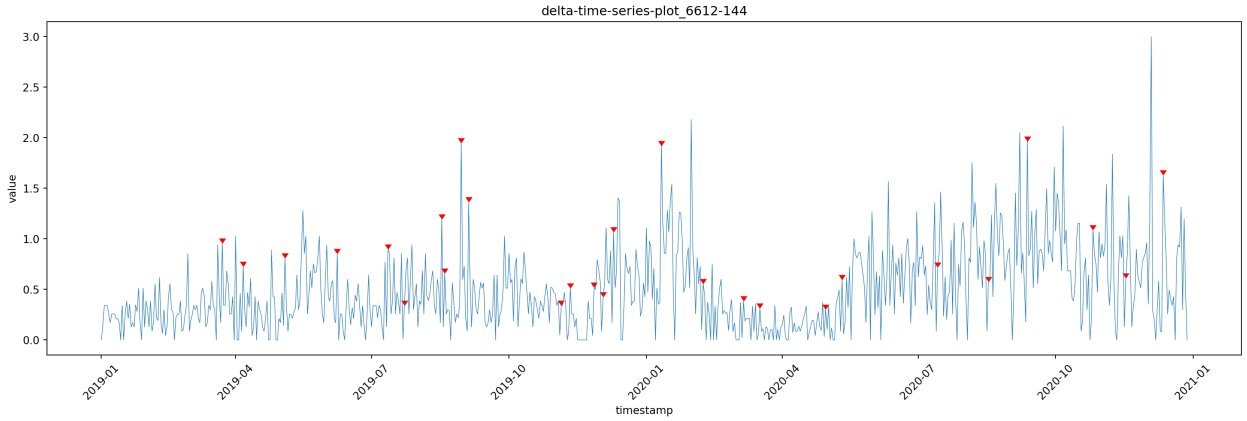


Figure 5.4: Withdrawals/bets for player 6612 with anomalies using DI algorithm.

We can see the behavior patterns of player 6612 based on their normalized withdrawal amounts in Figure 5.4. This is based on daily aggregated withdrawals. The red markers at various time points are the anomalies that were detected by the delta index method.

There are some limitations to this method for anomaly detection in a time series.

- This algorithm only considers a single prior value to determine the delta index which limits its efficacy for longer time periods.
- Determining the optimum threshold for anomaly detection is difficult and statistical methods only provide a general estimate over the entire range of the data which is sensitive to data skewness.

- Positive changes to a variable have significantly higher weight than negative changes since negative changes are in the range 0 to -1.0 whereas positive changes do not have an upper limit. For example, if a variable goes from 10 to 110 then the DI will be 10.0 whereas if it goes from 110 to 10 then the DI will be -0.9 which is a significantly lower magnitude for a similar change.

5.2.3 Anomaly Detection with Spectral Residual

The Spectral Residual (SR) algorithm is an unsupervised ML technique used for anomaly detection in time series data. It leverages the Fourier Transform to analyze frequency components of the time series and detect deviations that signify anomalies. Historically, this algorithm has been used for saliency detection in images by mimicking the way humans recognize salient features of an image [HZ07]. It is particularly useful for scenarios in which labeled anomalies are scarce or unavailable, and it provides a robust method for detecting unusual events based on the own structure of the data.

In essence, the SR algorithm transforms the time series to the frequency domain to isolate unusual components (anomalies), then transforms it back to the time domain to identify when these anomalies occur. The process involves filtering out expected patterns and focusing on unexpected deviations, making it effective for anomaly detection in time series data [LGC+23]. The steps used in this are as follows.

1. **Discrete Fourier Transform (DFT):** First we apply DFT to the input time series. The input time series is in the format $\{x_t\}_{t=1}^{T}$ where x_t represents the value of the time series at time t. The equation for DFT is given below:

$$X(f) = \sum_{t=1}^{T} x_t e^{-2\pi i f t}, \quad \text{for } f = 0, 1, \ldots, T-1 \tag{5.6}$$

   This step decomposes the time series into its frequency components, allowing us to analyze periodic patterns.

2. **Log Amplitude Spectrum:** Next, we compute the log amplitude spectrum of the transformed time series. The amplitude spectrum is denoted by $|X(f)|$ and is given

by:

$$|X(f)| = \sqrt{\text{Re}(X(f))^2 + \text{Im}(X(f))^2} \tag{5.7}$$

The log amplitude spectrum is:

$$\log|X(f)|$$

The log amplitude spectrum emphasizes the differences between significant frequencies and noise, making it easier to identify unusual patterns.

3. **Spectral Residual (SR):** Now, we obtain the spectral residual by subtracting a smoothed version of the log amplitude spectrum from itself. Let $|X(f)|_{smooth}$ denote the smoothed log amplitude spectrum obtained using a moving average filter. The spectral residual $\hat{X}(f)$ is given by:

$$\hat{X}(f) = \log|X(f)| - \log|X(f)|_{\text{smooth}} \tag{5.8}$$

This step isolates the unexpected or anomalous components in the frequency domain by removing the expected (smoothed) components.

4. **Inverse Fourier Transform:** The spectral residual $\hat{X}(f)$ is converted back to the time domain using inverse DFT function:

$$\hat{x}_t = \sum_{f=0}^{T-1} \hat{X}(f) e^{2\pi i f t} \tag{5.9}$$

Transforming back to the time domain allows us to see the anomalies in the original time series context.

5. **Anomaly Score Calculation:** We then calculate the anomaly score S_t at each time point. This can be done by computing the point-wise deviation between the original time series and the reconstructed time series from the spectral residual:

$$S_t = |x_t - \hat{x}_t| \tag{5.10}$$

The anomaly score quantifies the extent of deviation at each time point, highlighting potential anomalies.

6. **Anomaly Detection:** Determining anomalies is done by setting a threshold for the anomaly scores. This can be done using a fixed threshold, or a dynamic threshold using statistical methods on the distribution of the scores. For our experiment, we use a fixed threshold. This step identifies the specific time points in the time series where anomalies occur based on their scores. The output of this algorithm has 3 columns: timestamp, anomaly score, label.

The use of this algorithm for anomaly detection in a time-series was proposed by Microsoft. They also further proposed implementing a Convolutional Neural Network (CNN) at the output of the SR algorithm to further improve efficacy of the SR algorithm [RXW+19]. However, we found that we could not replicate the results claimed by Microsoft on our dataset. We suspect that this is due to the lack of a labeled dataset. In the paper published by Microsoft, there are three datasets that were tested:

1. **KPI** The KPI dataset is provided by the AIOPS data competition. Various labeled KPI curves are included that have been collected from Internet Companies such as Sogou, Tencent, eBay, etc.

2. **Yahoo** The Yahoo dataset, released by Yahoo Labs is also used for time series anomaly detection. This dataset includes both synthetic and real time-series curves. The synthetic curves are simulated, while the real-time-series curves are based on actual traffic from Yahoo services. Anomalies in the synthetic data are procedurally generated, whereas anomalies in the real data are manually labeled. All time-series in this dataset have a 1-hour interval.

3. **Microsoft** This dataset comes from Microsoft's internal anomaly detection service. A random selection of time-series, reflecting various KPIs such as revenues, active users, and page views, is used for evaluation. Anomalies in this dataset are manually labeled by customers or editors, and the time-series have a 1-day interval.

The results for these datasets as published by Microsoft are given in Table 5.5. Due to the lack of labeled anomalies in our dataset, we cannot obtain a metric for testing the

efficacy of this anomaly detection technique. Therefore, the results are subjective and require interpretation by experts in the field of problem gambling.

Table 5.5: Results for SR algorithm as published by Microsoft [RXW+19].

	KPI	Yahoo	Microsoft
F1-Score	0.666	0.529	0.484
Precision	0.637	0.404	0.878
Recall	0.697	0.765	0.334

Sampling, aggregation and normalization techniques for implementing the SR algorithm are the same as the steps for anomaly detection using delta index as discussed in Section 5.2.2. With the SR algorithm we can compute a *saliency map* or anomaly scores for any given time series on which we apply a threshold. An example of anomaly scores for a random player plotted over the transaction amount is given in Figure 5.5.

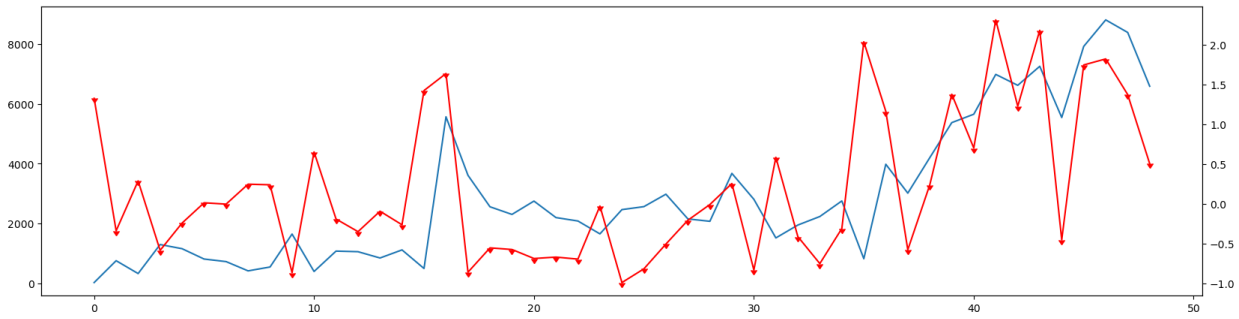


Figure 5.5: Anomaly score plot for monthly aggregate of player withdrawals.

The red plot line in Figure 5.5 corresponds to the anomaly score assigned to a specific data point with the score value represented on the right-hand side y-axis. The original withdrawal amount plot line is represented in blue with the values represented on the left hand side y-axis. The x-axis represents the timestamp. This figure is plotted for monthly aggregated player withdrawals for readability.

Finally, we plot the results of anomaly detection over the time series plot of the behavior variable.

We illustrate the output of the SR based anomaly detection of a sample player 6612 in Figure 5.6. This is the same sample player that was previously show in Figure 5.4. This serves as a comparison metric between the baseline Delta Index (DI) algorithm and

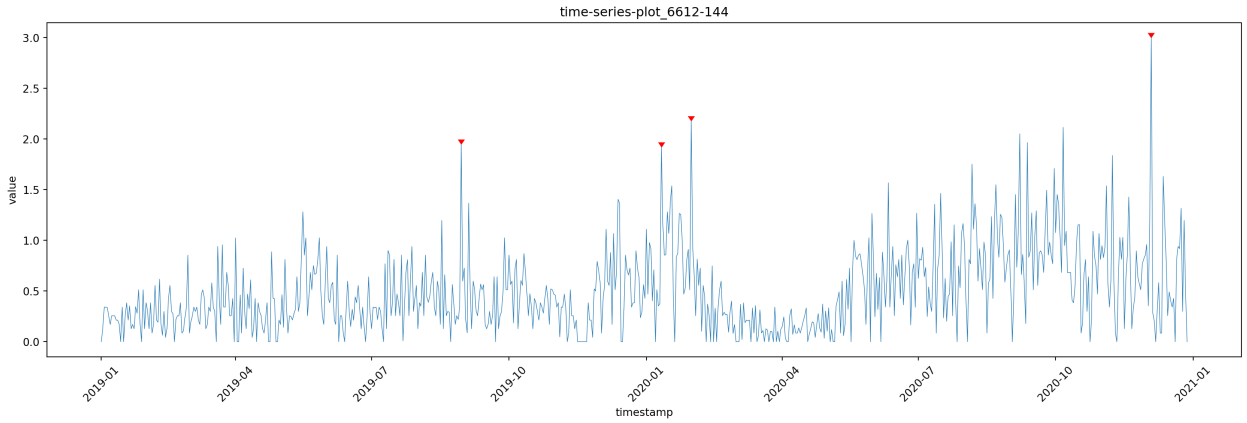


Figure 5.6: Withdrawals/bets for player 6612 with anomalies using SR algorithm.

the Spectral Residual (SR) algorithm. Similar to Figure 5.4, the red markers mark the anomalies that were detected by the SR algorithm. Although the interpretation of these results is subjective, from this example, we can observe that the SR algorithm is better at recognizing data points which show a more pronounced deviation from the rest of the data. Whereas the DI algorithm is potentially more aggressive at marking anomalies based on minor deviations from the data.

There are some limitations to the SR approach for time series anomaly detection.

- The SR algorithm can be sensitive to noise in the data, as it relies on transforming the time series to the frequency domain and identifying residuals. Noisy data can lead to false positives or missed anomalies.

- Although the SR algorithm is generally fast, its performance can degrade with extremely large datasets or very high-frequency data, as the computational cost of the Fourier transform increases.

- The SR algorithm identifies points in the time series that are anomalous but does not provide explanations or reasons for why a particular point is considered an anomaly. This can be a limitation in applications where understanding the cause of anomalies is important such as in identifying problem gambling behavior.

Chapter 6

Longitudinal Clustering

Longitudinal clustering is a statistical technique used to identify groups of subjects or observations that follow similar trajectories over time. This method is particularly useful in analyzing data collected at multiple time points, enabling the understanding of patterns of change or development within different sub-populations.

Various methods for longitudinal clustering include Group-Based Trajectory Modeling (GBTM), Growth Mixture Modeling (GMM), and Longitudinal K-Means (KML). These methods differ in their approaches to modeling and clustering trajectories. For example, GBTM focuses on identifying distinct trajectory groups within the data, while GMM allows for variability within clusters by modeling individual trajectories as mixtures of different distributions [TPvdH21].

In practical applications, longitudinal clustering is often used in medical research to explore how different patient groups progress over time and how these trajectories are associated with health outcomes. For instance, it has been used to study the relationship between health behaviors and multimorbidity in older adults, identifying distinct behavioral patterns that predict long-term health risks [SWH24]. It has also shown promise in observing the effects of treatments for various ailments and diseases in clinical trials [Alb99].

In this chapter we discuss an application of GMM clustering on longitudinal slices of data to not only observe behavior pattern changes among groups of gamblers, but also facilitate tracking the behavior trajectory of individual players.

6.1 Methodology

We perform longitudinal clustering on the PTP dataset using the following steps that will be detailed in this section:

1. **Filtering & Sampling:** The data is filtered to include only approved transactions. We are not interested in declined or failed transactions for this experiment as they do not contribute to cash flow between player and merchant and therefore do not represent betting behavior. To effectively cluster players based on their betting behavior, it is also necessary to eliminate extraneous factors that may affect player behavior. For example, location, area, income, age, and gender are all variables that may affect our results. Since we do not have access to this information, we need to choose a subset of players that can limit the effects of these factors on our result. Moreover, we also limit the time frame between which players are considered. Sampling methodology is explained in detail in Section 6.1.1.

2. **Feature Selection:** Features relevant to betting/gambling behavior are selected based on prior published research in this area. The feature selection strategy and methodology are explained in Section 6.1.2

3. **Model Selection & Clustering:** Prior research in clustering players based on gambling behavior has shown that models with 4 components provide an optimal result [WTEL20, PFM+21]. We test this theory for our data using the Akaike Information Criterion (AIC) score. We then cluster player behavior for every 30-day period of time to observe overall changes in player behavior patterns over time. The methodology and code for this process is explained in Section 6.1.3.

6.1.1 Filtering & Sampling

To restrict the effect extraneous variables may have on our result, we limit the analysis to one merchant at a time. From each merchant, we filter all withdrawals that have been approved. This process has been described in Algorithm 6.

Algorithm 6: Filter and sample player data for clustering.

Input: "dataset.json", merchantID
Output: Merchant Dataframe

```
# Read dataset
df ← DaskDataframe(dataset, blocksize=2^28, orient='records') ;
# Filter by specified merchant ID
filtered ← df WHERE df.merchantid = merchantID;
filtered ← filtered WHERE servicemethod = 'WITHDRAWAL' AND status ==
  'APPROVED' ;
```

Longitudinal clustering is performed for discrete time intervals with data aggregated for each interval. For this experiment, we chose an interval of 30 days starting from the first transaction of each player from the specified merchant. However, we restrict our time frame from January 2019 to December 2020. Since we are performing longitudinal clustering for 6 periods (each period being 30 days) we are only interested in extracting roughly 6 months of transactions for each player.

Algorithm 7: Get players with first transaction between 2019-01-01 and 2020-06-30.

Input: Filtered Dataset
Output: Sampled Dataframe

```
# Set start and end date for first transaction
start_date ← 2019-01-01 ;
end_date ← 2020-06-30 ;
merchant_data ← Filtered Dataset from Algorithm 6 ;
# Get all first transaction dates for players.
first_transactions ← merchant_data.groupby('accountid').aggregate('reqtimeutc' :
  'min');
first_transactions.rename('reqtimeutc' : 'first_transaction_time');
# Get player accounts with first transaction between start and end
  date.
player_accounts ← first_transactions WHERE reqtimeutc ≥ start_date AND
  reqtimeutc ≤ end_date ; # SparkSQL
merchant_data ← merchant_data WHERE accountid IN player_accounts ;
  # SparkSQL
sampled_data ← merge(merchant_data, first_transactions, on='accountid') ;
```

Following is a description of the steps taken by Algorithm 7:

1. Set the start date to 2019-01-01 and the end date to 2020-06-30. Since we want to limit the range of all transactions to between 2019-01-01 and 2020-12-31, setting the end date to 2020-06-30 ensures that the last transaction conducted by any player in a 6 month period will never exceed 2020-12-31.

2. The first transaction dates for all players are first calculated and stored in a dataframe.

3. Account IDs for all player accounts having their first transaction between start date and end date are filtered to create the `player_accounts` dataframe.

4. A filter is applied to the `merchant_data` dataframe to only keep players with account IDs present in the `player_accounts` dataframe.

5. The `merchant_data` and `first_transaction` dataframes are then merged (INNER JOIN) to create a new column in the original merchant data to indicate the first transaction date of each of the players. We call the resultant dataframe `sampled_data`.

Following is a description of the steps taken by Algorithm 8:

1. A new column is generated in the sampled dataframe containing the start date for each 30-day period. Note that the column for `6_months_after` will contain the start date for the 7th overall period (days 181-210) which we do not be consider for this experiment.

2. The `filtered_transactions` dataframe is created which contains all transactions between the first transaction of a player and the end of the last period.

3. For each 30 day period we filter transactions that lie between the start and end of that period by referencing the month columns for each account ID and generate a dataframe for that period. This dataframe is then appended to a list.

6.1.2 Feature Selection

As discussed in Section 2, the lack of publicly available quantitative gambling data has historically proven to be a limiting factor for behavioral research in problem gambling. This has

Algorithm 8: Create a dataframe for each 30-day period.

Input: Sampled Dataframe
Output: Dataframe for each period

```
# Generate columns for each 30 day period
foreach i in range(1, 7) do
    column_name ← "{i}_months_after";
    sampled_data[column_name] ← sampled_data[first_transaction_time] +
     DateOffset(days← 30 ∗ i);
end
# Get transactions between start date and 6 months later.
filtered_transactions ← sampled_data WHERE reqtimeutc ≥
 'first_transaction_time' AND reqtimeutc ≤ '6_months_after' ;
filtered_transactions.rename('first_transaction_time' : '0_months_after');
monthly_split ← empty list ;
foreach i in range(1, 7) do
    each_month ← filtered_transactions WHERE reqtimeutc ≥
     '{i − 1}_months_after' AND reqtimeutc < '{i}_months_after' ;
    monthly_split.append(each_month) ;
end
```

also contributed to the lack of research related to identifying features relevant to gambling behavior obtainable from quantitative data. However, a recent white paper published by Playtech discussed features that they found to effectively quantify problem gambling behavior using internal metrics on their proprietary problem gambling detection product [Pla21]. Due to the proprietary nature of this study and the data used in it, it is difficult to establish the validity of these findings.

Based on the Playtech study and consultation with experts in the field of responsible gambling, we identified 5 features that can be extracted from our dataset. Since this dataset is from a payment provider and not from a gambling merchant, using withdrawals toward a gambling merchant is the best option for a proxy for bets in this study. The assumption is that whenever a withdrawal is made towards a gambling merchant, the full withdrawal amount will be used for betting; in either the short or the long term. However, this also has limitation on accuracy. For example, a withdrawal of $100 could be used as two $50 bets. A player may also make additional bets that are not captured if they use their account balance from a prior win; thus not requiring a withdrawal.

Table 6.1: Description of features selected for clustering

Feature	Description	Column Name
Frequency	Number of bets.	freq
Mean Frequency	Average bets made per day.	mean_freq
STD Frequency	Standard deviation of bets per day.	std_dev_freq
Mean Bet Amount	Average bet amount per day.	avg_bet_amount
STD Bet Amount	Standard deviation of bet amount per day.	std_dev_bet

Table 6.1 describes the features that we extract from the dataset. More features can potentially be extracted from a richer dataset, which will lend more robustness to the model.

Algorithm 9 is used to extract the features described in Table 6.1 from the dataset. Once feature selection and extraction are complete, we perform model selection and clustering which is demonstrated in the next subsection.

6.1.3 Model Selection & Clustering

At the beginning of this chapter we discuss the use of 4 components for clustering, which was found to be an optimal number by previous studies. To verify this claim, we use the Akaike Information Criterion (AIC) to score clustering performed with components ranging from 2 to 7. AIC is a statistical method used for model selection that balances the complexity of a model against the goodness of its fit [BA98]. Mathematically, AIC is defined as:

$$\text{AIC} = 2k - 2\ln(L) \tag{6.1}$$

Therefore, a lower AIC score indicates a better model.

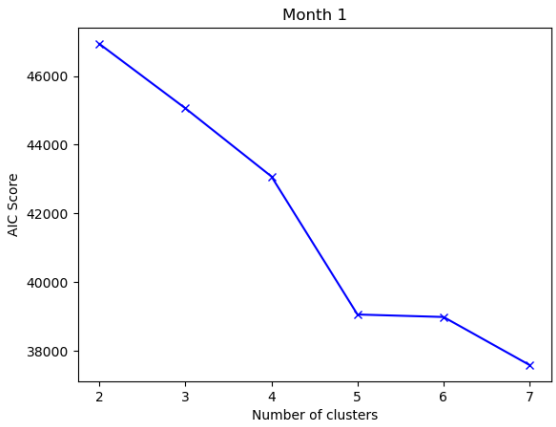
From the AIC scores plotted for each period in Figure 6.1, we find that the optimal model clusters vary significantly for each period. However, either 4 or 5 components is generally a good number. Although 6 or 7 components would be better, we want to keep the number of components as few as possible to maintain the resolution power of the model. Notably in Figures 6.1b and 6.1c the AIC score for 4 and 5 components is quite low.

For this study, we assume a component number of 4 for all periods of the GMM clustering.

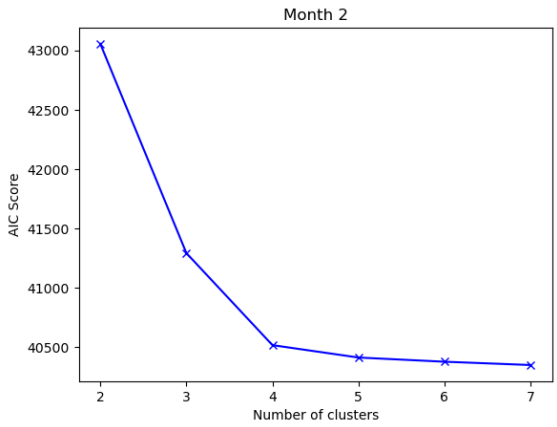
Algorithm 9: Extract features from the dataframe generated by Algorithm 8.

Input: Dataframes for each period (monthly_split)
Output: Feature Dataframes for each period

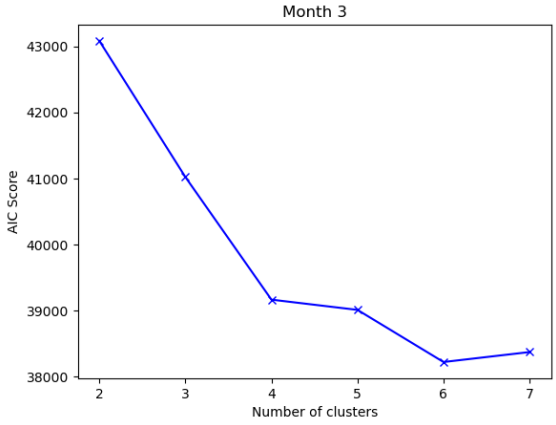
```
stats ← empty list;
foreach month in monthly_split do
    withdrawals ← month WHERE servicemethod = 'WITHDRAWAL';
    # Frequency
    freq ← withdrawals.groupby(accountid).size() ;
    # Mean Frequency
    mean_freq ← freq ;
    foreach row in mean_freq do
        row[freq] ← row[freq] /30 ;
    end
    # STD Frequency
    daily_freq ← withdrawals.groupby(accountid, reqtimeutc.year, reqtimeutc.month,
      reqtimeutc.day).aggregate(transactionid : 'count');
    daily_freq.rename('transactionid' : 'daily_freq');
    std_dev_freq ← daily_freq.groupby(accountid);
    std_dev_freq[daily_freq] ← std_dev_freq[daily_freq].std();
    std_dev_freq.rename('daily_freq' : 'std_dev_freq');
    # Mean Bet Amount
    avg_bet_amt ← withdrawals.groupby(accountid)[transactionamount].mean();
    avg_bet_amt.rename('transactionamount' : 'avg_bet_amt') ;
    # STD Bet Amount
    std_dev_bet ← withdrawals.groupby(accountid)[transactionamount].std();
    std_dev_bet.rename('transactionamount' : 'std_dev_bet') ;
    # Create dataframe for period
    stats_month ← merge(freq, mean_freq, std_dev_freq, avg_bet_amount, std_dev_bet,
      on=accountid);
    stats.append(stats_month) ; # Append to stats
end
```



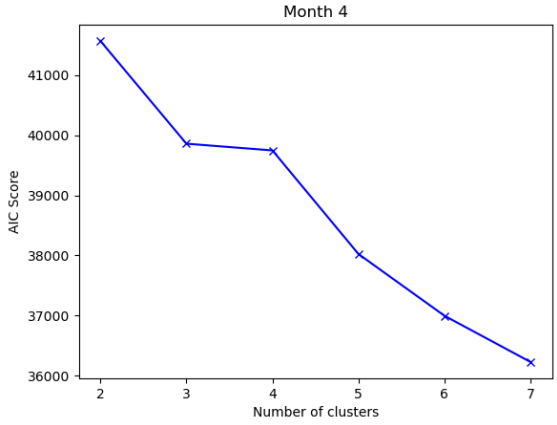
(a) AIC for 0-30 days.



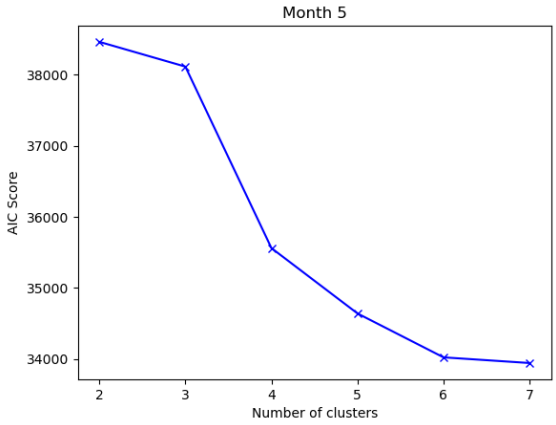
(b) AIC for 31-60 days.



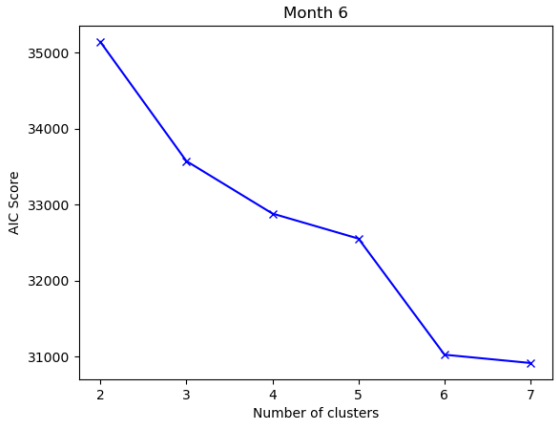
(c) AIC for 61-90 days.



(d) AIC for 91-120 days.



(e) AIC for 121-150 days.



(f) AIC for 151-180 days.

Figure 6.1: AIC scores for GMM clustering with 2-7 components for each time period.

6.2 Results

Using the features from Section 6.1.2 we cluster players for each 30 day period using the GMM clustering method described in Section 3.2.2. We use the implementation of GMM provided by the Scikit-learn library [PVG+11] for leveraging GPU computation and faster deployment.

As discussed in Section 6.1.1, we restrict our analysis to one merchant at a time. For this experiment, we picked the player with the highest number of unique account IDs to represent a large population of players. The assumption made here is that picking players from a single gambling merchant will limit the effect of extraneous variables related to the demographics of the players.

6.2.1 Longitudinal Analysis

The selected gambling merchant for this study comprised 2283 unique players. We observed that each subsequent 30-day period of play had fewer players than were observed in the first 30. This is illustrated in Table 6.2.

Table 6.2: Number of players in each period

Period	**Players**
Month 1 (0-30 days)	2283
Month 2 (31-60 days)	2002
Month 3 (61-90 days)	1981
Month 4 (91-120 days)	1898
Month 5 (121-150 days)	1745
Month 6 (151-180 days)	1596

Note that we refer to each period as *Month* for the sake of brevity, but the actual period length is 30 days.

For each period, we compute the statistics of each cluster and plot them using a box plot to visualize the differences between the clusters. The box plot visualizations are plotted using Z-score standardization of the features. Z-Score standardization is used to bring all features in the data to a common scale, which makes them easier to visualize on a box plot.

The mathematical formula for Z-Score is:

$$Z = \frac{X - \mu}{\sigma} \tag{6.2}$$

Where Z is the Z-score, X is the data point, μ is the mean of the variable for that period, and σ is the standard deviation of that variable for that period.

Month 1

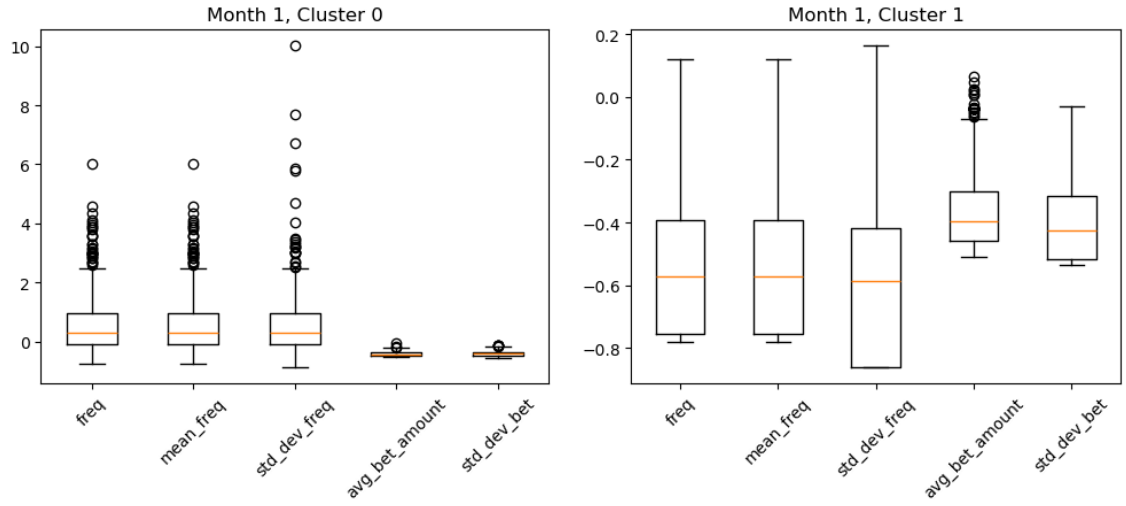
The cluster statistics for Month 1 are given in Table 6.3. In this table, the features are represented by their column names as previously explained in Table 6.1. Here, μ for each feature represents the average value of that feature for each cluster and σ indicates the standard deviation of that feature within the cluster. We follow the same labeling pattern throughout this analysis. The cluster numbers shown here do not represent clusters with similar characteristics across each period. For example, the characteristics exhibited by Cluster 2 in Month 1 may not be the same as the characteristics exhibited by Cluster 2 in Month 2 or any other Month. The cluster numbers are simply used to differentiate between clusters for each period.

Table 6.3: Cluster statistics for Month 1 of longitudinal clustering.

Cluster		0	1	2	3
freq	μ	53.21	10.42	13.58	37.38
	σ	39.98	8.43	11.34	29.68
mean_freq	μ	1.77	0.35	0.45	1.25
	σ	1.33	0.28	0.38	0.99
std_dev_freq	μ	2.32	0.42	0.73	1.62
	σ	1.7	0.42	0.66	1.19
avg_bet_amount	μ	29.94	39.21	395.1	108.82
	σ	14.99	25.04	257.55	52.69
std_dev_bet	μ	15.07	14.07	190.14	73.36
	σ	9.07	13.13	154.11	40.08
Count		523	911	305	544

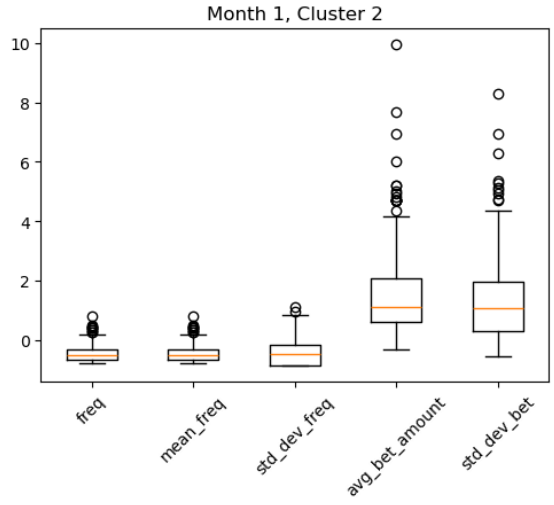
Table 6.3 shows that the highest number of players are in Cluster 1. This cluster is characterized by an average player betting frequency of 10.42 and an average bet amount of 39.21. However, the standard deviation of the bet amount is quite high at 25.04, meaning

these are players with relatively average betting amounts, and about 10 transactions per month. Cluster 2 contains the lowest number of players with relatively similar betting frequency but very high average bet amounts. This indicates a cluster with high value betting players. It is also characterized by a high standard deviation for bet amounts. Clusters 0 and 3 are characterized by players having high betting frequency with about 2 withdrawals per day on average. Cluster 3 characterizes players with high betting amounts and high frequency, which could potentially be characterized as "risky". The characteristics of these clusters can be visualized in Figure 6.2.

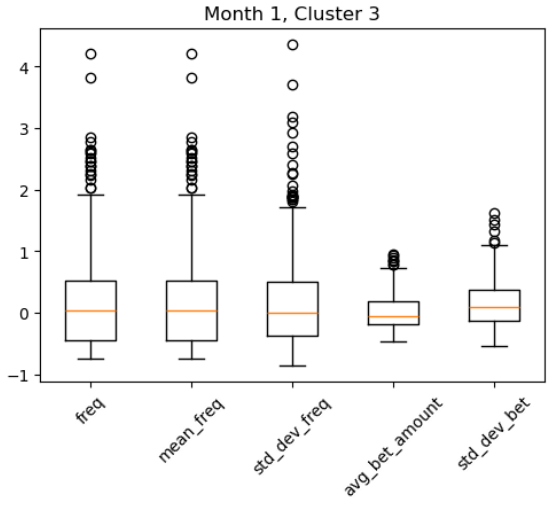


(a) Month 1, Cluster 0

(b) Month 1, Cluster 1



(c) Month 1, Cluster 2



(d) Month 1, Cluster 3

Figure 6.2: Box plot visualization of clusters for Month 1 of longitudinal clustering.

Note that in Figure 6.2, the y-axis of the box plot indicates the Z-Score of the feature values calculated using Equation 6.2. Therefore, 0 on the y-axis indicates the mean of that feature, and positive or negative values indicate the distance of a point from the mean as a factor of its standard deviation.

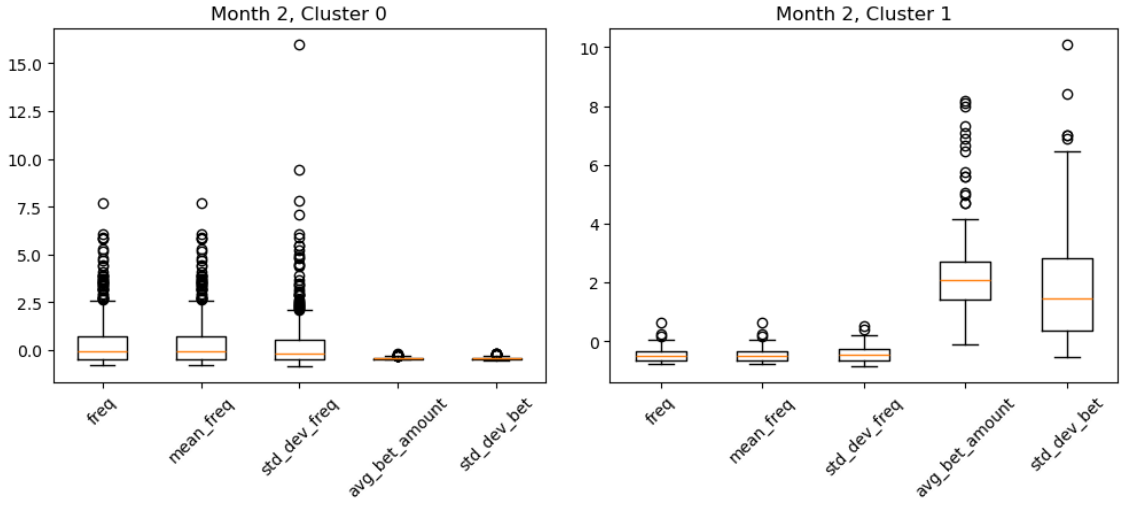
Month 2

Statistics for the clustering solution for Month 2 are given in Table 6.4.

Table 6.4: Cluster statistics for Month 2 of longitudinal clustering.

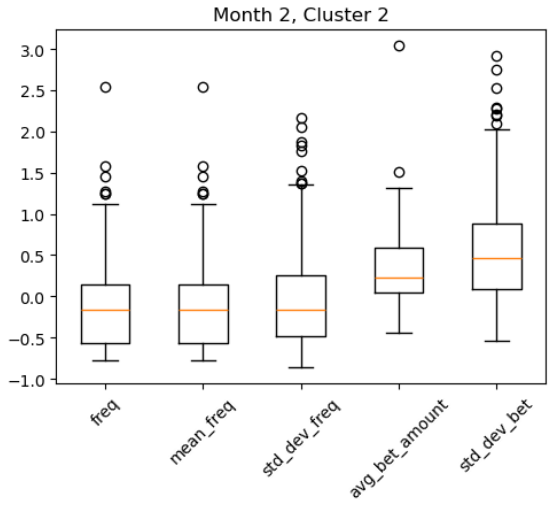
Cluster		0	1	2	3
freq	μ	43.91	11.96	26.22	35.68
	σ	47.05	8.79	19.23	32.73
mean_freq	μ	1.46	0.4	0.87	1.19
	σ	1.57	0.29	0.64	1.09
std_dev_freq	μ	1.7	0.65	1.27	1.51
	σ	2.11	0.49	0.95	1.37
avg_bet_amount	μ	21.28	565.27	170.18	59.77
	σ	8.94	309.93	77.33	26.18
std_dev_bet	μ	9.67	252.5	111.77	32.28
	σ	6.7	220.41	68.35	21.1
Count		696	137	379	790

It is evident when we compare Tables 6.4 and 6.3 that the cluster representation for both these periods is similar, but the average bet amounts are relatively higher. Cluster 3, containing 790 players, is characterized by players with approximately 1 transaction per day with betting amounts around $60. Clusters 1 and 2 have players with very high average betting amounts; however, Cluster 1 has a lower average frequency of betting, and Cluster 2 exhibits relatively higher betting frequency. It is also worth noting that the standard deviation for Cluster 2 is closer to the mean betting amount than any other cluster. Cluster 0 in this period represents players with relatively high betting frequencies but low average betting amounts. The characteristics of these clusters as box plots are visualized in Figure 6.3.

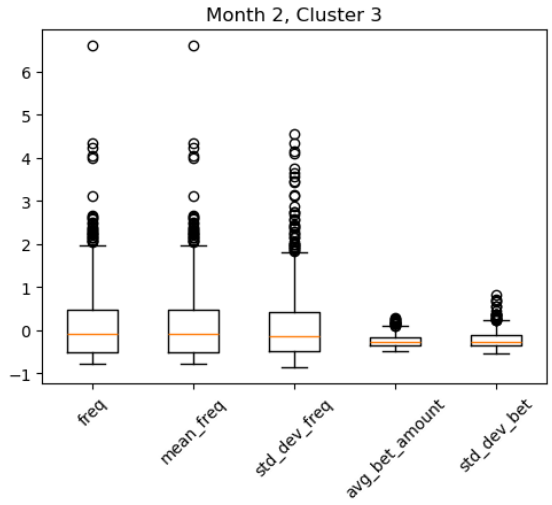


(a) Month 2, Cluster 0

(b) Month 2, Cluster 1



(c) Month 2, Cluster 2



(d) Month 2, Cluster 3

Figure 6.3: Box plot visualization of clusters for Month 2 of longitudinal clustering.

Month 3

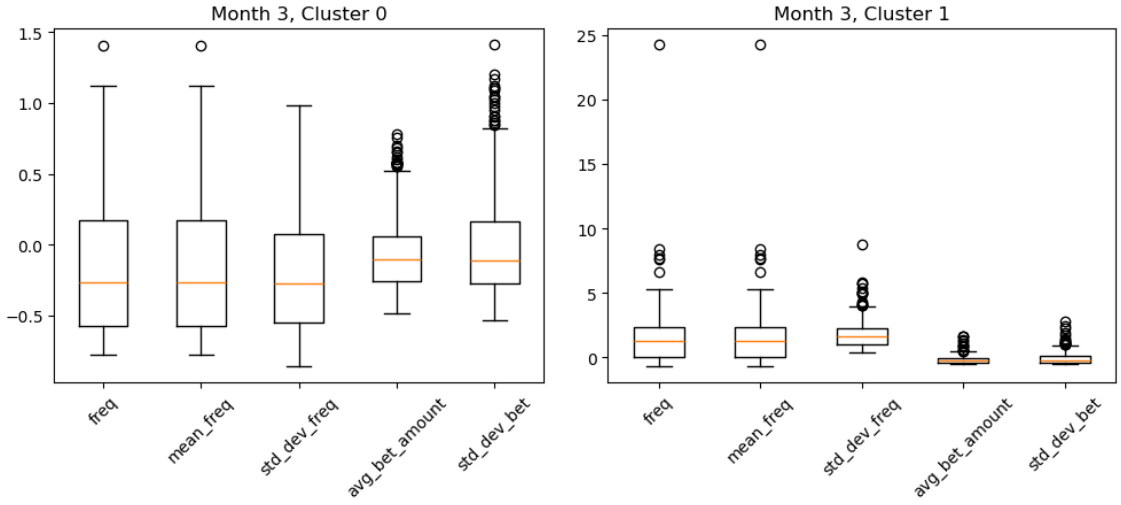
Cluster statistics for Month 3 are given in Table 6.5.

Table 6.5: Cluster statistics for Month 3 of longitudinal clustering.

Cluster		0	1	2	3
freq	μ	24.9	93.41	31.08	18.38
	σ	18.54	88.61	25.27	14.45
mean_freq	μ	0.83	3.11	1.04	0.61
	σ	0.62	2.95	0.84	0.48
std_dev_freq	μ	1.01	4.36	1.18	0.94
	σ	0.71	1.93	0.88	0.79
avg_bet_amount	μ	96.19	81.16	25.34	485.28
	σ	48.56	74.24	11.81	343.25
std_dev_bet	μ	54.6	55.5	12.33	248.35
	σ	41.03	58.96	8.83	250.33
Count		642	227	896	216

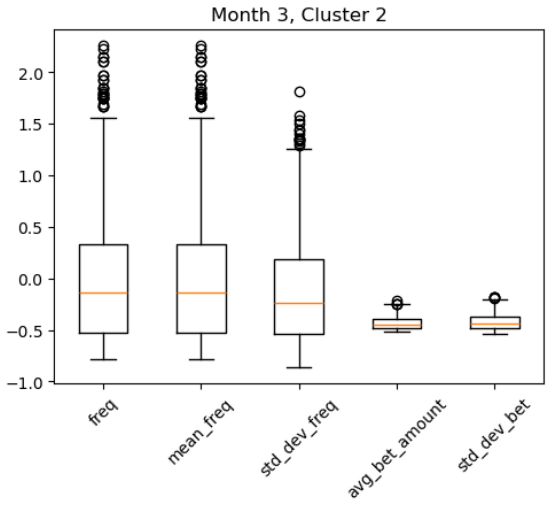
As compared to the previous two periods, in this period, Cluster 1 exhibits an unusually high betting frequency for players with an average daily frequency of 3.11 and average overall frequency higher than 93. Cluster 2, having the maximum number of players, exhibits similar behavior to the previous clusters with an average of 1 transaction per day and an average bet amount around $25. Cluster 3 represents players with unusually high betting amounts with a low average frequency of play, whereas Cluster 0 represents players betting around $90 to $100 with an average frequency of about 25. Overall, the most interesting clusters are Clusters 1 and 3, with Cluster 1 exhibiting very high transaction volume, potentially highlighting gambling problems. The characteristics of these clusters as box plots are visualized in Figure 6.4.

The most unique box plot of note is that of Cluster 1 in Figure 6.4b. This cluster shows players with the highest deviation from mean values of features, whereas Clusters 0 and 3 in Figures 6.4a and 6.4d show the least deviation of features from the mean. This potentially indicates high fluctuation in player behavior within the cluster.

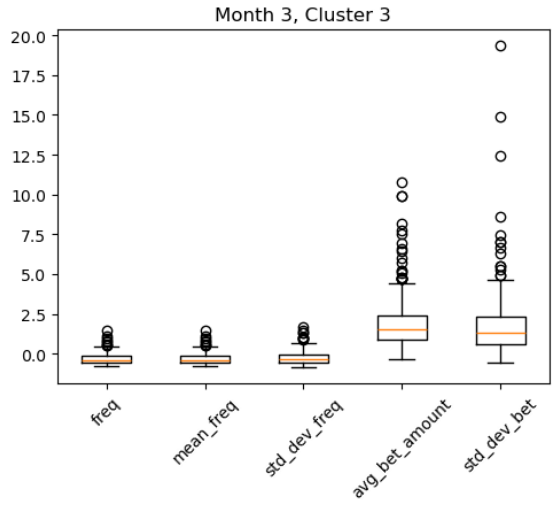


(a) Month 3, Cluster 0

(b) Month 3, Cluster 1



(c) Month 3, Cluster 2



(d) Month 3, Cluster 3

Figure 6.4: Box plot visualization of clusters for Month 3 of longitudinal clustering.

Month 4

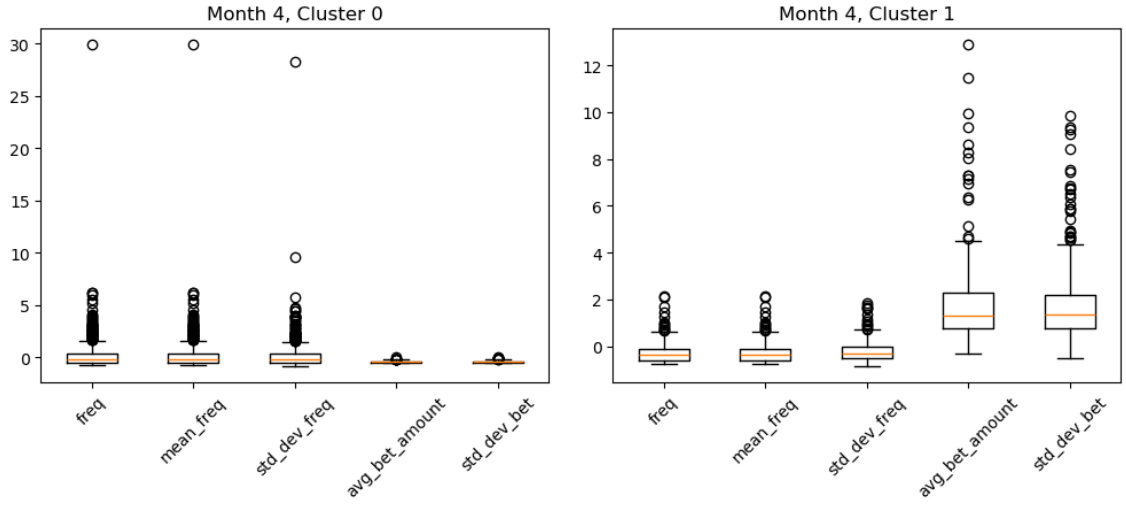
Cluster statistics for the 4th month/period are shown in Table 6.6. This month shows a significant shift in cluster statistics from prior months.

Table 6.6: Cluster statistics for Month 4 of longitudinal clustering.

Cluster		0	1	2	3
freq	μ	36.3	20.87	30.52	6.0
	σ	53.21	18.48	26.96	0.0
mean_freq	μ	1.21	0.7	1.02	0.2
	σ	1.77	0.62	0.9	0.0
std_dev_freq	μ	1.52	1.01	1.38	1.0
	σ	2.13	0.81	1.26	0.0
avg_bet_amount	μ	26.38	464.52	94.69	3750.0
	σ	13.29	370.56	54.46	0.0
std_dev_bet	μ	12.56	251.8	54.65	1369.31
	σ	9.75	205.95	40.02	0.0
Count		935	251	711	1

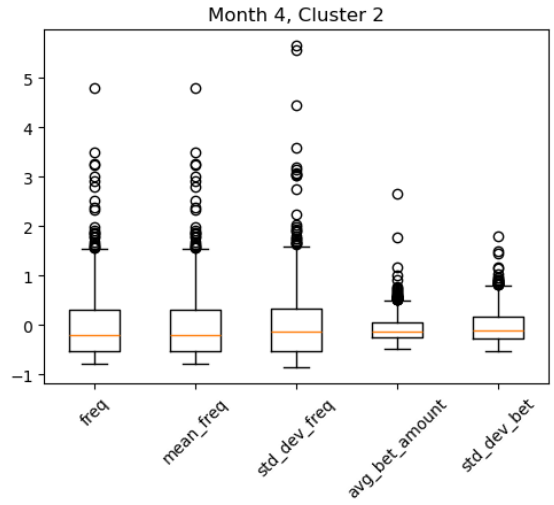
Cluster 3 in Table 6.6 shows exactly 1 player belonging to the cluster with a very high average betting amount of $3750 and an average betting frequency of 6. This indicates that this player conducted 6 withdrawals this month. The σ for all variables here is 0, since there are no other players in this cluster. This clustering behavior is indicative of either one of two possibilities: 1) The behavior exhibited by this player is an outlier and requires investigation by a problem gambling expert. This is also indicative of the outlier detection capabilities of this model. 2) The limitations of this dataset with regards to information about players is misrepresenting the behavior of this player. Due to the lack of ground truth or labels in this dataset, it is difficult to determine which of these possibilities is valid.

Similar to the previous periods, Cluster 0, with the highest number of players, exhibits an average frequency of 36, which translates to approximately 1-2 transactions per day. The average betting amount here is also relatively low and is the lowest among all clusters for this period. Cluster 1 encompasses players with very high betting amounts and lower betting frequencies than Cluster 0. Cluster 2 comprises players with betting frequencies similar to Cluster 0 but with relatively higher betting amounts of around $94-$95. The characteristics of these clusters as box plots are visualized in Figure 6.5.

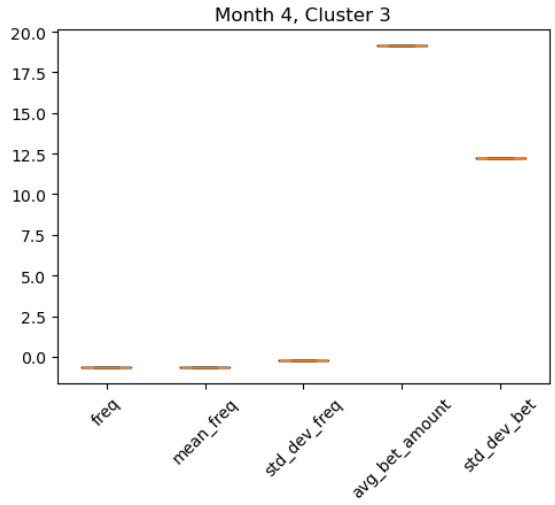


(a) Month 4, Cluster 0

(b) Month 4, Cluster 1



(c) Month 4, Cluster 2



(d) Month 4, Cluster 3

Figure 6.5: Box plot visualization of clusters for Month 4 of longitudinal clustering.

Note that the box plot for Cluster 3 in Figure 6.5d is indicative of a single player.

Month 5

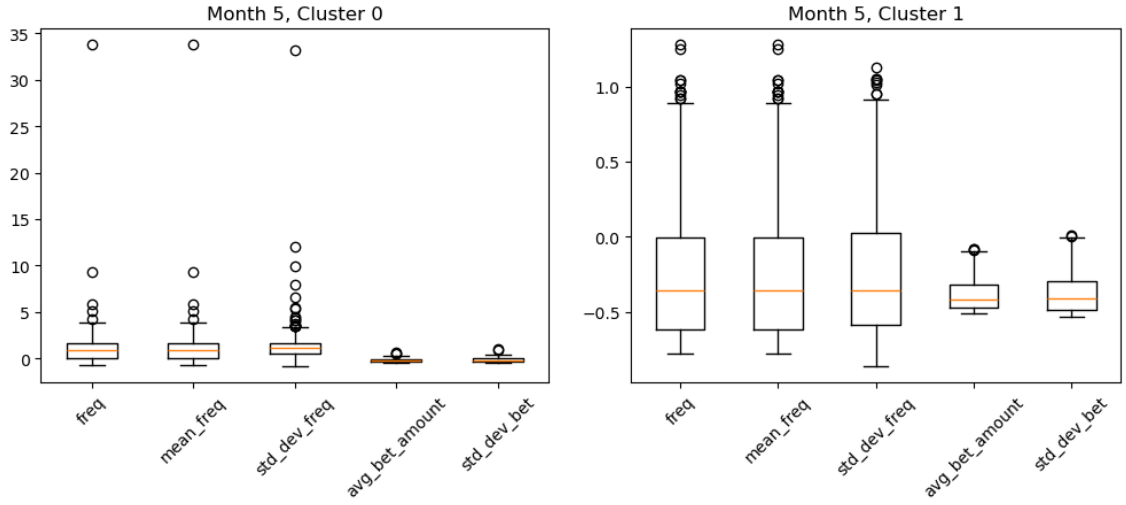
The clusters for Month 5 are similar to the clusters in Month 4, with one cluster having just 2 players with very high average bet amounts. Cluster statistics for this month are illustrated in Table 6.7.

Table 6.7: Cluster statistics for Month 5 of longitudinal clustering.

Cluster		0	1	2	3
freq	μ	75.54	21.43	14.0	22.31
	σ	91.74	17.19	14.14	17.77
mean_freq	μ	2.52	0.71	0.47	0.74
	σ	3.06	0.57	0.47	0.59
std_dev_freq	μ	3.69	0.95	0.36	1.12
	σ	3.89	0.75	0.51	0.9
avg_bet_amount	μ	62.92	33.06	3458.33	264.33
	σ	39.45	19.38	766.03	258.63
std_dev_bet	μ	36.26	16.13	1313.33	146.69
	σ	27.9	13.79	224.34	155.8
Count		268	954	2	521

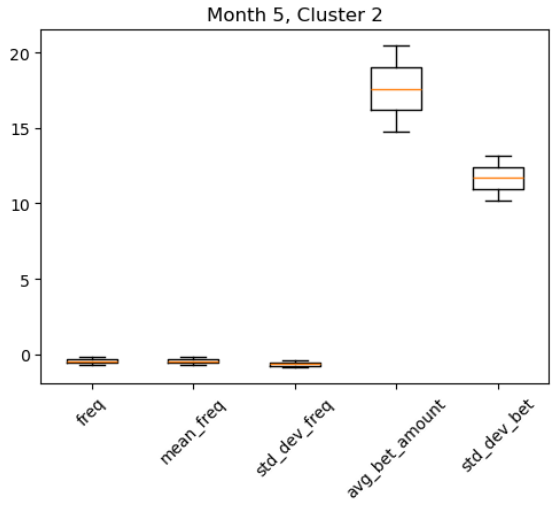
Cluster 2 in Table 6.7 is very similar to Cluster 3 from the previous month shown in Table 6.6. However, this time there are 2 players exhibiting very high withdrawal amounts with an average withdrawal frequency of 14. Cluster 1, having the highest number of players, has a lower mean withdrawal frequency this time as compared to Cluster 0 in the previous month which also had the highest number of players. The average bet amount for this cluster also does not show any significant changes. Cluster 0 indicates players with very high betting frequency and approximately double the average bet amount as compared to Cluster 1. Cluster 3 comprises players with relatively higher average bet amounts when compared to Cluster 0 and 1 but with an average mean frequency nearly matching that of Cluster 1. Figure 6.6 represents these clusters as box plots.

Cluster 2 in Figure 6.6c represents the behavior of the 2 outlier players with average bet amounts and standard deviations of the bet amounts showing a very wide deviation from the overall mean of the data. Similar to Month 4, this potentially represents outliers in the

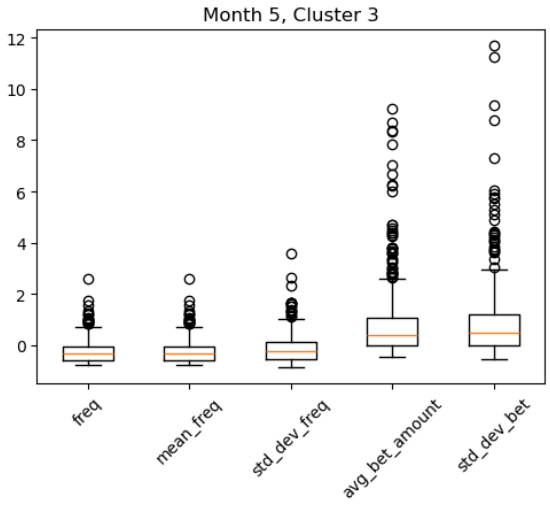


(a) Month 5, Cluster 0

(b) Month 5, Cluster 1



(c) Month 5, Cluster 2



(d) Month 5, Cluster 3

Figure 6.6: Box plot visualization of clusters for Month 5 of longitudinal clustering.

data.

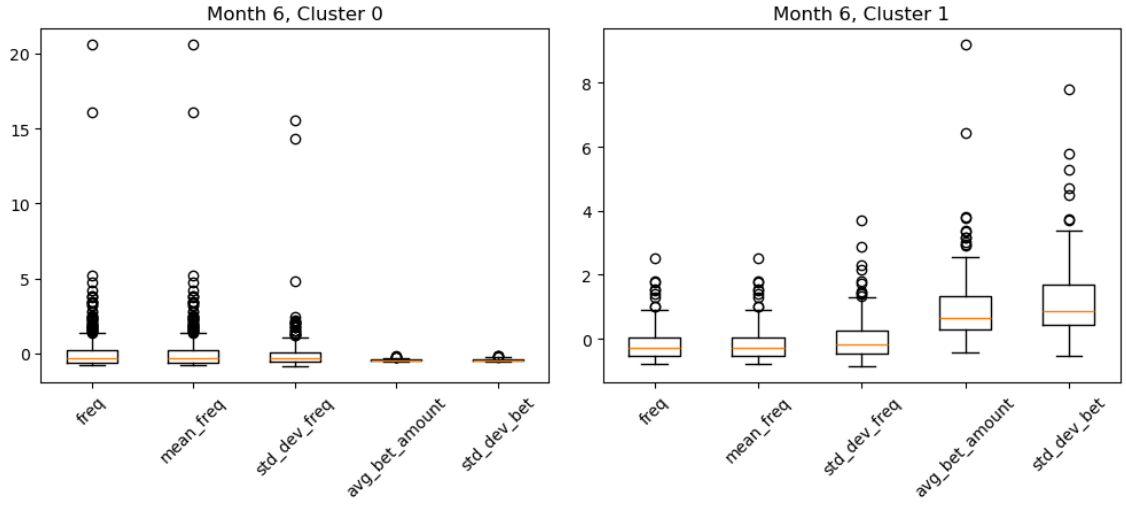
Month 6

The clustering statistics for Month 6 are found in Table 6.8. As compared to Months 4 and 5, Month 6 does not contain a cluster with outlier-like characteristics.

Table 6.8: Cluster statistics for Month 6 of longitudinal clustering.

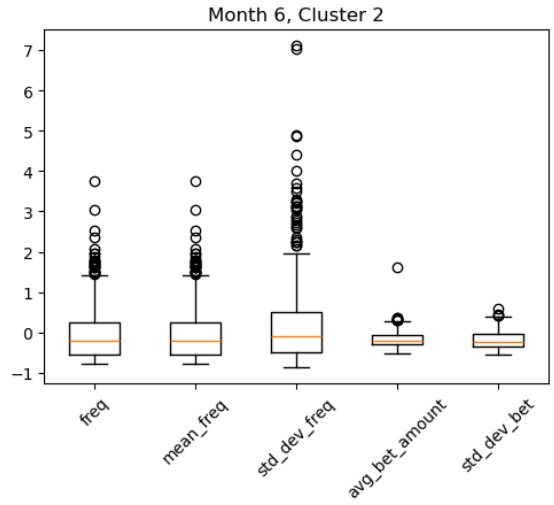
Cluster		0	1	2	3
freq	μ	31.69	24.64	29.37	8.62
	σ	52.96	19.54	25.82	6.88
mean_freq	μ	1.06	0.82	0.98	0.29
	σ	1.77	0.65	0.86	0.23
std_dev_freq	μ	1.16	1.31	1.63	0.53
	σ	1.69	1.03	1.6	0.45
avg_bet_amount	μ	23.29	283.29	73.82	977.33
	σ	10.31	193.26	35.89	615.87
std_dev_bet	μ	11.26	174.19	36.7	449.26
	σ	8.23	114.56	24.04	470.27
Count		621	280	648	47

Cluster 3 for Month 6 as shown in Table 6.8 represents the lowest number of players having the highest average bet amount and the lowest average frequency. However, the standard deviation for the bet amounts is also quite high, indicating that the range of bets for this category of players varies significantly. Clusters 1 and 2 exhibit relatively similar withdrawal frequency with Cluster 1, showing higher average bet amounts than Cluster 2. Cluster 2 also has the highest number of players; however, the number is very close to Cluster 0, which has a withdrawal frequency quite close to Cluster 2, but exhibits lower average bet amounts. We can infer that Clusters 0 and 2 signify the average player, with Cluster 3 representing high value players with fewer play sessions and Cluster 1 representing high value players with frequent gambling sessions. However, the bet values for Cluster 1 are still significantly lower than Cluster 3. Box plots of these clusters are shown in Figure 6.7.

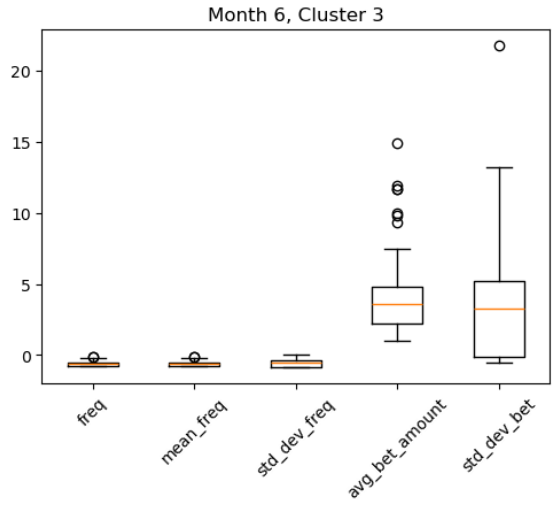


(a) Month 6, Cluster 0

(b) Month 6, Cluster 1



(c) Month 6, Cluster 2



(d) Month 6, Cluster 3

Figure 6.7: Box plot visualization of clusters for Month 6 of longitudinal clustering.

6.2.2 Player Behavior Tracking

In addition to observing changes in overall player behavior across longitudinal time slices, longitudinal clustering also facilitates identifying changes in individual player behaviors across these slices. For example, if we look at the singular player in Month 4, Cluster 3, we can obtain the `accountID` of this player to track their behavior from Months 1 to 6.

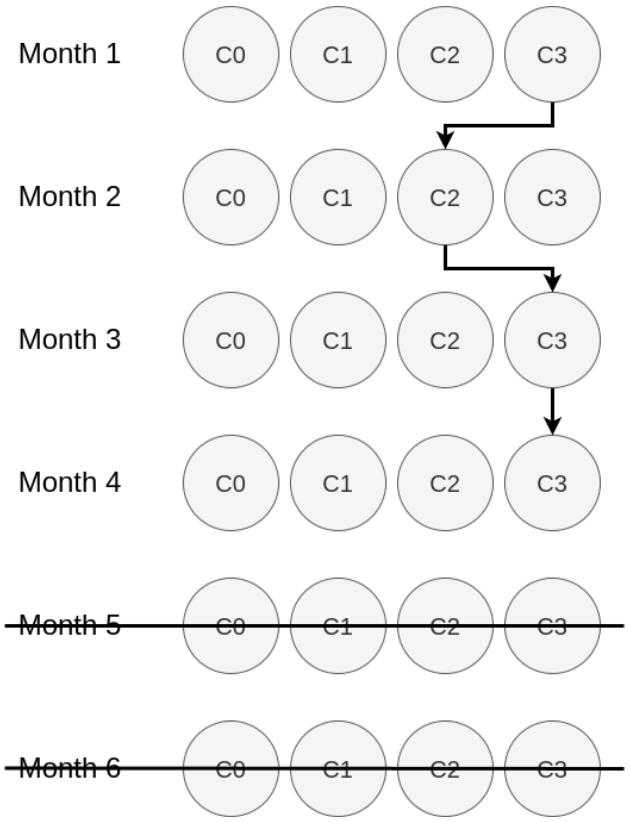


Figure 6.8: Cluster transition diagram for Player 28886.

Figure 6.8 shows the movement of Player 28886 from Month 1 to 4, after which they have no activity for subsequent months. Based on Table 6.3, Cluster 3 in Month 1 is characterized by relatively higher average bet amounts of about $108 with an average of 1.25 bets per day. Subsequently, Cluster 2 in Month 2 is characterized by average bet values of around $170 with an average of 0.87 bets per day. Cluster 3 in Month 3 is the cluster with the highest average bet amounts of $485 and average betting frequency per day around 0.61. Finally, we see the player in Cluster 3 for Month 4 as an outlier with a bet amount of $3750 for that month with 6 total withdrawals made.

For another example, we consider a random player that has played from Months 1 through 6 and observe their trajectory.

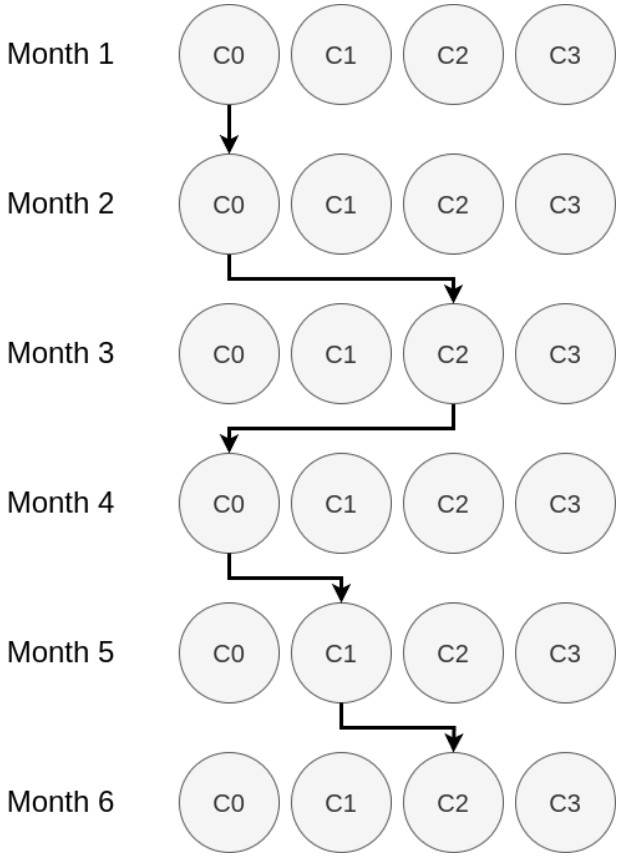


Figure 6.9: Cluster transition diagram for Player 226.

Player 226 starts at Cluster 0 for Month 1, characterizing players with high betting frequencies of about 53 bets in the period and amounts averaging about $30. Subsequently, this player moves to Cluster 0 for Month 2, also characterized by high betting frequencies of 43 bets per period and low average betting amounts of about $21. Month 3 Cluster 2 shows a betting frequency of about 31 bets per period and average bet amount of $25. Month 4 Cluster 0 again shows very similar characteristics to Month 3 Cluster 2 and comprises the highest number of players for that month. Month 1 Cluster 5 is also the majority player cluster characterized by about 21 withdrawals per period with a mean bet amount of $33. Finally, in Month 6, the player is part of Cluster 2, which is the majority player cluster, characterizing about 30 bets per period and a relatively higher average bet amount of $73. Based on this player's trajectory, we can say that this player consistently belongs to the

majority player class and exhibits average gambling behavior.

Based on the above two examples, we can consider this behavior transition between periods a state transition diagram. With more available data, this can potentially enable future researchers to predict a player's behavior trajectory using ML based forecasting techniques.

Chapter 7

Conclusions and Future Work

In this chapter, we summarize the contents of this dissertation and discuss avenues for future research. We provide an overview of what was discussed in the preceding chapters, our studies and contributions, and the applicability of these contributions.

Chapters 2 focuses on our research in gambling-related harm and the applicability of data science (primarily machine learning) in contributing towards this area. We discuss our published scoping review and highlight the gaps in applications of supervised and unsupervised ML in the areas of responsible gambling. Chapter 3 introduces the theory behind machine learning and how it relates to our contributions in this dissertation. We discuss the mathematics and theory behind supervised and unsupervised machine learning and review the importance behind clustering data when faced with a lack of ground truth.

7.1 Data Engineering & Time Series Analysis

Lack of publicly available quantitative data for machine learning applications for problem gambling are discussed briefly in Chapter 4. To alleviate this issue, we use data obtained from an online digital payment provider (identity undisclosed) to facilitate research. The data and analysis discussed in this chapter were also published in 2023. We describe the contents of the data obtained from the Payments Technology Provider and the tools used for cleaning and analysis.

The use of Apache Spark for distributed data processing is briefly discussed along with its applicability for our dataset. Additionally, we describe the utility of Apache Parquet:

an open-source, columnar data storage format, for our data. Data cleaning was started by converting the obtained dataset to parquet format where a detailed algorithm of the process was illustrated. Subsequently, we discuss the importance of data obfuscation and how it pertained to the received dataset. The process followed for obfuscation along with the SQL queries are discussed in detail.

The descriptive analysis of the data reviews its contents and overall distribution. We demonstrated how a majority of the data contents are focused in the 2019-2021 time frame. Furthermore, we establish the general distribution and percentile figures for the values of all the transactions in the data.

In Chapter 5, we introduce the concept of time series and discussed its various applications with a few examples. The concept of stationarity is also discussed along with the ADF and KPSS tests, which can be used to identify the stationarity of data. We illustrate and analyze these results of the test application on our dataset.

The problems related to modeling player behavior as a time series are discussed in this section with the primary problem being the non-continuous nature of player transactions. We devised a solution to this problem using aggregation at periodic time intervals to ensure continuity in the data. The algorithms used for this process are also discussed here.

A statistical metric called delta index is introduced, which can be used to measure changes in a time series. We discuss the use of this metric to detect anomalies in time series and also its limitations.

Furthermore, we applied the unsupervised machine learning based Spectral Residual algorithm to our time series data to detect anomalies in player behavior. The theory behind this algorithm is discussed in detail, and we provide evidence from prior research of this algorithm's efficacy for anomaly detection. Due to the lack of labeled anomalies in our data, we were unable to verify its efficacy for anomaly detection in player behavior modeled as a time series, and we emphasize the subjectivity of the obtained result.

7.2 Application of Longitudinal Clustering

In Chapter 6, we introduce the concept of longitudinal analysis and discuss its practical application in the fields of medicine and clinical trials. The idea of applying clustering to longitudinal slices of data is also discussed with different applicable clustering algorithms. Furthermore, we discuss the methodology and results of our application of longitudinal clustering to the PTP data.

Our methodology was broadly based on 3 steps: filtering and sampling, feature selection, model selection and clustering. In the filtering and sampling section step, we first describe the steps taken to filter data from a single merchant and for a specified time frame. In this case we use a time frame of 2019-01-01 to 2020-06-30. The algorithm followed for this step is discussed here in detail. Then we proceed to split this period into 6 slices of 30-days each to get distinct 30-day periods for longitudinal clustering.

The lack of studies on quantitative gambling data is highlighted in the feature selection step, resulting in a dearth of information on the optimal features to be used for quantifying gambling behavior. We selected features based on a recent white paper published by PlayTech on features they deem to provide optimal performance on their proprietary problem gambling software. We emphasize the proprietary nature of this study and the difficulty in establishing the efficacy of their method. Advice from industry experts in problem gambling was also considered for the feature selection process. Five features were described and extracted from the PTP dataset. The algorithm for this selection and extraction process is discussed.

Based on prior research on clustering players based on their gambling behavior, we decided to use 4 components for our clustering process. To verify our decision, we employed the AIC method to score GMM clustering performance with different numbers of components ranging from 2 to 7. The results for each period were evaluated individually.

Finally, we discuss the results obtained from the application of these steps and provid a detailed description of the cluster characteristics for each time slice. Furthermore, we also demonstrate how the behavior of individual players can be tracked using our longitudinal clustering method to determine changes in player behavior over time.

7.3 Future Research

One of the most important contributions that can be made for improving anomaly detection in player behavior modeled as a time series is the availability of labeled data. However, the process of identifying real anomalies in player behavior is not an easy task and will require experts in problem and responsible gambling to compile such data. Availability of labeled data can significantly improve anomaly detection using machine learning models especially since data of this nature can enable the use of supervised learning techniques as opposed to the unsupervised technique discussed in this dissertation.

The generalized nature of our proposed longitudinal clustering and time series anomaly detection methods broadens the possibilities for further analysis of these methods using different or additional player behavior features. Due to the limitations of our current dataset, we could not include more granular features in our analysis, which provides significant room for improving the efficacy of our methods. This includes optimizing the number of components, and modeling with dynamic components for each period. Further testing could also include changing the size of the time slices to determine changes in behavior over shorter or longer periods of time.

Our longitudinal clustering method can be further expanded to include player trajectory forecasting by modeling the months and clusters as a finite state machine. However this will require a significant amount of data to accomplish.

Bibliography

[AHM21] Khaled Gubran Al-Hashedi and Pritheega Magalingam. Financial fraud detection applying data mining techniques: A comprehensive review from 2009 to 2019. *Computer Science Review*, 40:100402, 2021.

[Alb99] Paul S. Albert. Longitudinal data analysis (repeated measures) in clinical trials. *Statistics in Medicine*, 18(13):1707–1732, 1999.

[AP00] Association AP. Diagnostic and statistical manual of mental disorders (dsm-iv-tr), fourth edition, text revision, 2000.

[Auf21] B. Auffarth. *Machine Learning for Time-Series with Python: Forecast, predict, and detect anomalies with state-of-the-art machine learning methods*. Packt Publishing, 2021.

[BA98] Kenneth P. Burnham and David R. Anderson. Practical Use of the Information-Theoretic Approach. In Kenneth P. Burnham and David R. Anderson, editors, *Model Selection and Inference: A Practical Information-Theoretic Approach*, pages 75–117. Springer, New York, NY, 1998.

[BLS04] Alex Blaszczynski, Robert Ladouceur, and Howard J. Shaffer. A Science-Based Framework for Responsible Gambling: The Reno Model. *Journal of Gambling Studies*, 20(3):301–317, September 2004.

[Bro19] Dorian Brown. What is Supervised Learning? A Mathematical Perspective, September 2019.

[CBHT+20] Gaëlle Challet-Bouju, Jean-Benoit Hardouin, Elsa Thiabaud, Anaïs Saillard, Yann Donnio, Marie Grall-Bronnec, and Bastien Perrot. Modeling Early Gambling Behavior Using Indicators from Online Lottery Gambling Tracking Data: Longitudinal Analysis. *Journal of Medical Internet Research*, 22(8):e17675, August 2020. Company: Journal of Medical Internet Research Distributor: Journal of Medical Internet Research Institution: Journal of Medical Internet Research Label: Journal of Medical Internet Research Publisher: JMIR Publications Inc., Toronto, Canada.

[CBL+15] Peter Collins, Alex Blaszczynski, Robert Ladouceur, Howard, Howard Shaffer, Davis Fong, and Jean-Luc Venisse. Responsible Gambling: Conceptual Considerations. *Gaming Law Review and Economics*, 19:594–599, October 2015.

[CG17] Bernardo T Chagas and Jorge FS Gomes. Internet gambling: A critical review of behavioural tracking research. *Journal of Gambling Issues*, 36:1–27, 2017.

[CRC16] Ning Chen, Bernardete Ribeiro, and An Chen. Financial credit risk assessment: a recent review. *Artificial Intelligence Review*, 45:1–23, 2016.

[DAS20] Tripti Dimri, Shamshad Ahmad, and Mohammad Sharif. Time series analysis of climate variables using seasonal arima approach. *Journal of Earth System Science*, 129:1–16, 2020.

[Dre05] Gérard Dreyfus. *Neural Networks: Methodology and Applications*. Springer Science & Business Media, November 2005.

[EBW+14] D Excell, G Bobashev, H Wardle, D Gonzalez-Ordonez, T Whitehead, R Morris, and P Ruddle. Predicting problem gamblers: analysis of industry data. In *responsible gambling trust conference, harm minimisation: investigating gaming machines in licensed betting offices. London*, volume 10, 2014.

[FBM17] M. Fauth-Bühler and K. Mann. Neurobiological correlates of internet gaming disorder: Similarities to pathological gambling. *Addictive Behaviors*, 64:349–356, January 2017.

[FH21] Chad Fulton and Kirstin Hubrich. Forecasting US Inflation in Real Time. *Econometrics*, 9(4):36, December 2021. Number: 4 Publisher: Multidisciplinary Digital Publishing Institute.

[FMG+23] Mojtaba A. Farahani, M. R. McCormick, Robert Gianinny, Frank Hudacheck, Ramy Harik, Zhichao Liu, and Thorsten Wuest. Time-series pattern recognition in Smart Manufacturing Systems: A literature review and ontology. *Journal of Manufacturing Systems*, 69:208–241, August 2023.

[FW01] Jacqueline Ann Ferris and Harold James Wynne. *The Canadian problem gambling index*. Canadian Centre on substance abuse Ottawa, ON, 2001.

[Gai14] Sally M. Gainsbury. Review of Self-exclusion from Gambling Venues as an Intervention for Problem Gambling. *Journal of Gambling Studies*, 30(2):229–251, June 2014.

[GAP+22] Kasra Ghaharian, Brett Abarbanel, Dylan Phung, Piyush Puranik, Shane Kraus, Alan Feldman, and Bo Bernhard. Applications of data science for responsible gambling: a scoping review. *International Gambling Studies*, pages 1–24, 2022.

[GD01] M. D. Griffiths and P. Delfabbro. The biopsychosocial approach to gambling: contextual factors in research and clinical interventions. *Journal of Gambling Issues (JGI)*, (5), 2001.

[Gha22] Kasra Ghaharian. Payment Profiles: Using Payment Transaction Data to Identify and Characterize Gamblers. *UNLV Theses, Dissertations, Professional Papers, and Capstones*, December 2022.

[GR20] Joshua B. Grubbs and Joseph A. Rosansky. Problem gambling, coping motivations, and positive expectancies: A longitudinal survey study. *Psychology of Addictive Behaviors*, 34(2):414–419, 2020. Place: US Publisher: American Psychological Association.

[HS17] Linda Hancock and Garry Smith. Critiquing the Reno Model I-IV International Influence on Regulators and Governments (2004–2015)— the Distorted Reality of "Responsible Gambling". *International Journal of Mental Health and Addiction*, 15(6):1151–1176, December 2017.

[HZ07] Xiaodi Hou and Liqing Zhang. Saliency Detection: A Spectral Residual Approach. In *2007 IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–8, June 2007. ISSN: 1063-6919.

[JWHT21] Gareth James, Daniela Witten, Trevor Hastie, and Robert Tibshirani. *An Introduction to Statistical Learning: with Applications in R*. Springer Texts in Statistics. Springer US, New York, NY, 2021.

[KBHG24] Kanon Kamronnaher, Andrew Bellucco, Whitney K. Huang, and Colin M. Gallagher. Estimating Value at Risk and Expected Shortfall: A Brief Review and Some New Developments, May 2024. arXiv:2405.06798 [stat].

[Kie17] R. Kienzler. *Mastering Apache Spark 2.x*. Packt Publishing, 2017.

[KJ13] Max Kuhn and Kjell Johnson. *Applied Predictive Modeling*. Springer New York, New York, NY, 2013.

[KKWZ15] H. Karau, A. Konwinski, P. Wendell, and M. Zaharia. *Learning Spark: Lightning-Fast Big Data Analysis*. O'Reilly Media, 2015.

[KPSS92] Denis Kwiatkowski, Peter CB Phillips, Peter Schmidt, and Yongcheol Shin. Testing the null hypothesis of stationarity against the alternative of a unit

root: How sure are we that economic time series have a unit root? *Journal of econometrics*, 54(1-3):159–178, 1992.

[LGC+23] Tianyang Lei, Chang Gong, Gang Chen, Mengxin Ou, Kewei Yang, and Jichao Li. A novel unsupervised framework for time series data anomaly detection via spectrum decomposition. *Knowledge-Based Systems*, 280:111002, November 2023.

[LL22] Xitao Liu and Lihui Li. Prediction of Labor Unemployment Based on Time Series Model and Neural Network Model. *Computational Intelligence and Neuroscience*, 2022:7019078, June 2022.

[LR20] C. Livingstone and A. Rintoul. Moving on from responsible gambling: a new discourse is needed to prevent and minimise harm from gambling. *Public Health*, 184:107–112, July 2020.

[LVRK20] Balakrishnan Lakshmanan, Palaniappan Senthil Nayagam Vivek Raja, and Viswanathan Kalathiappan. Sales Demand Forecasting Using LSTM Network. In Subhransu Sekhar Dash, C. Lakshmi, Swagatam Das, and Bijaya Ketan Panigrahi, editors, *Artificial Intelligence and Evolutionary Computations in Engineering Systems*, pages 125–132, Singapore, 2020. Springer.

[MPAM22] Evgeniy Marinov, Dessislava Petrova-Antonova, and Simeon Malinov. Time Series Forecasting of Air Quality: A Case Study of Sofia City. *Atmosphere*, 13(5):788, May 2022. Number: 5 Publisher: Multidisciplinary Digital Publishing Institute.

[Mus11] Rizwan Mushtaq. Augmented Dickey Fuller Test, August 2011.

[NS24] Philip Newall and Thomas B. Swanton. Beyond 'single customer view': Player tracking's potential role in understanding and reducing gambling-related harm. *Addiction*, 119(7):1156–1163, 2024.

[OSBJ20] Tim Offermans, Ewa Szymańska, Lutgarde M. C. Buydens, and Jeroen J. Jansen. Synchronizing process variables in time for industrial process monitoring and control. *Computers & Chemical Engineering*, 140:106938, September 2020.

[pdt23] The pandas development team. pandas-dev/pandas: Pandas, June 2023.

[PFM+21] Fernando Peres, Enrico Fallacara, Luca Manzoni, Mauro Castelli, Aleš Popovič, Miguel Rodrigues, and Pedro Estevens. Time Series Clustering of Online Gambling Activities for Addicted Users' Detection. *Applied Sciences*, 11(5):2397, January 2021.

[PHGBCB18] Bastien Perrot, Jean-Benoit Hardouin, Marie Grall-Bronnec, and Gaëlle Challet-Bouju. Typology of online lotteries and scratch games gamblers' behaviours: A multilevel latent class cluster analysis applied to player account-based gambling data. *International Journal of Methods in Psychiatric Research*, 27(4):e1746, 2018. _eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1002/mpr.1746.

[Phi14] Kahlil S. Philander. Identifying high-risk online gamblers: a comparison of data mining procedures. *International Gambling Studies*, 14(1):53–63, January 2014. Publisher: Routledge.

[Pla21] Playtech. Behavioural markers of harm. *Industry Research Brief Vo2. (1) - Markers of Harm*, December 2021.

[PR22] M. C. Pegalajar and L. G. B. Ruiz. Time Series Forecasting for Energy Consumption. *Energies*, 15(3):773, January 2022. Number: 3 Publisher: Multidisciplinary Digital Publishing Institute.

[PRS+16] Felipe Pezoa, Juan L Reutter, Fernando Suarez, Martín Ugarte, and Domagoj Vrgoč. Foundations of json schema. In *Proceedings of the 25th International Conference on World Wide Web*, pages 263–273. International World Wide Web Conferences Steering Committee, 2016.

[PTG23] Piyush Puranik, Kazem Taghva, and Kasra Ghaharian. Descriptive Analysis of Gambling Data for Data Mining of Behavioral Patterns. In Kevin Daimi and Abeer Al Sadoon, editors, *Proceedings of the Second International Conference on Innovations in Computing Research (ICR'23)*, pages 40–51, Cham, 2023. Springer Nature Switzerland.

[PVG+11] Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, et al. Scikit-learn: Machine learning in python. *Journal of machine learning research*, 12(Oct):2825–2830, 2011.

[RBBL16] Yordan P. Raykov, Alexis Boukouvalas, Fahd Baig, and Max A. Little. What to Do When K-Means Clustering Fails: A Simple yet Principled Alternative Algorithm. *PLOS ONE*, 11(9):e0162259, September 2016. Publisher: Public Library of Science.

[RN16] Stuart J. Russell and Peter Norvig. *Artificial intelligence : a modern approach*. Pearson, 2016.

[Rom19] Victor Roman. Unsupervised Machine Learning: Clustering Analysis, April 2019.

[Ros17] Sheldon M. Ross. *Introductory Statistics*. Academic Press, January 2017. Google-Books-ID: c838DAAAQBAJ.

[RXW+19] Hansheng Ren, Bixiong Xu, Yujing Wang, Chao Yi, Congrui Huang, Xiaoyu Kou, Tony Xing, Mao Yang, Jie Tong, and Qi Zhang. Time-Series Anomaly Detection Service at Microsoft. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 3009–3017, July 2019.

[SM+05] Petre Stoica, Randolph L Moses, et al. *Spectral analysis of signals*, volume 452. Pearson Prentice Hall Upper Saddle River, NJ, 2005.

[SR21] Sasha Stark and Janine Robinson. Online gambling in unprecedented times: Risks and safer gambling strategies during the covid-19 pandemic. *Journal of Gambling Issues*, (47), 2021.

[SSC23] Sanghamitra Sanyal, Sanchari Sarkar, and Moitreyee Chakrabarty. Time series analysis of groundwater quality at selected sites of Purba and Paschim Burdwan, West Bengal, India. *Environmental Monitoring and Assessment*, 195(9):1039, August 2023.

[STC+23] Thomas B. Swanton, Stephanie Tsang, Sharon B. Collard, Ellen Garbarino, and Sally M. Gainsbury. Cashless gambling: Qualitative analysis of consumer perspectives regarding the harm minimization potential of digital payment systems for electronic gaming machines. *Psychology of Addictive Behaviors*, pages No Pagination Specified–No Pagination Specified, 2023. Place: US Publisher: American Psychological Association.

[SWH24] Alisha Suhag, Thomas L. Webb, and John Holmes. Longitudinal clustering of health behaviours and their association with multimorbidity in older adults in England: A latent class analysis. *PLOS ONE*, 19(1):e0297422, January 2024.

[SY20] Kristina P. Sinaga and Miin-Shen Yang. Unsupervised K-Means Clustering Algorithm. *IEEE Access*, 8:80716–80727, 2020. Conference Name: IEEE Access.

[Tea23] RAPIDS Development Team. *RAPIDS: Libraries for End to End GPU Data Science*, 2023.

[TPvdH21] Niek Den Teuling, Steffen Pauws, and Edwin van den Heuvel. Clustering of longitudinal data: A tutorial on a variety of approaches, November 2021.

[Tsa05] Ruey S Tsay. *Analysis of financial time series*. John wiley & sons, 2005.

[Voh16] Deepak Vohra. Apache Parquet. In Deepak Vohra, editor, *Practical Hadoop Ecosystem: A Definitive Guide to Hadoop-Related Frameworks and Tools*, pages 325–335. Apress, Berkeley, CA, 2016.

[Win60] Peter R. Winters. Forecasting Sales by Exponentially Weighted Moving Averages. *Management Science*, 6(3):324–342, 1960. Publisher: INFORMS.

[WTEL20] Rhiannon C. Wiley, Matthew A. Tom, Timothy C. Edson, and Debi A. LaPlante. Behavioral Markers of Risky Daily Fantasy Sports Play. *Journal of Sport and Social Issues*, 44(4):356–371, August 2020.

[ZXW+16] Matei Zaharia, Reynold S. Xin, Patrick Wendell, Tathagata Das, Michael Armbrust, Ankur Dave, Xiangrui Meng, Josh Rosen, Shivaram Venkataraman, Michael J. Franklin, Ali Ghodsi, Joseph Gonzalez, Scott Shenker, and Ion Stoica. Apache Spark: a unified engine for big data processing. *Communications of the ACM*, 59(11):56–65, October 2016.

Curriculum Vitae

Graduate College
University of Nevada, Las Vegas

Piyush Aniruddha Puranik
piyushpuranik@gmail.com

Degrees:

Master of Science in Computer Science 2019
University of Nevada, Las Vegas

Bachelor of Engineering in Computer Engineering 2015
Savitribai Phule Pune University, India

Dissertation Title: Mining Gambling Data for Modeling Gambling Behavior Patterns

Dissertation Examination Committee:

Chairperson, Dr. Kazem Taghva, Ph.D.
Committee Member, Dr. Fatma Nasoz, Ph.D.
Committee Member, Dr. Mingon Kang, Ph.D.
Committee Member, Dr. Laxmi Gewali, Ph.D.
Graduate College Representative, Dr. Brett Abarbanel, Ph.D.

Work Experience

- **Graduate Assistant** at UNLV - *Aug 2017 to Aug 2024*
  - Designed and taught CS 448 (Computer Security)
  - Taught CS 135 (Computer Science I) and CS 202 (Computer Science II)
- **Embedded Software Engineer** at Power Control Engineers - *Jun 2016 to Apr 2017*
  - Improved control panel UI responsiveness by 80% by optimizing firmware code.
  - Enabled quick deployment of control panel firmware by designing driver code for custom micro-controller boards.
  - Designed Qt5 based applications for streamlined on-site control panel diagnostics and configuration.
  - Designed a custom low cost data logging system for PLC control panels.

Publications

- Puranik, P., Taghva, K., Ghaharian, K., 2023. Descriptive Analysis of Gambling Data for Data Mining of Behavioral Patterns, in: Daimi, K., Al Sadoon, A. (Eds.), Proceedings of the Second International Conference on Innovations in Computing Research (ICR'23). Springer Nature Switzerland, Cham, pp. 40–51. https://doi.org/10.1007/978-3-031-35308-6_4
- Ghaharian, K., Puranik, P., Abarbanel, B., Taghva, K., Kraus, S.W., Singh, A., Feldman, A., Bernhard, B., 2023. Payments transaction data from online casino players and online sports bettors. Data in Brief 48, 109077. https://doi.org/10.1016/j.dib.2023.109077
- Ghaharian, K., Abarbanel, B., Phung, D., Puranik, P., Kraus, S., Feldman, A., Bernhard, B., 2023. Applications of data science for responsible gambling: a scoping review. International Gambling Studies 23, 289–312. https://doi.org/10.1080/14459795.2022.2135753

- Choi, S., Puranik, P., Dahal, B., Taghva, K., 2022. How to generate data for acronym detection and expansion. Adv. in Comp. Int. 2, 23. https://doi.org/10.1007/s43674-021-00024-6

- Hua, D., Gao, J., Mayo, R., Smedley, A., Puranik, P., Zhan, J., 2020. Segregating Hazardous Waste Using Deep Neural Networks in Real-Time Video, in: 2020 10th Annual Computing and Communication Workshop and Conference (CCWC). Presented at the 2020 10th Annual Computing and Communication Workshop and Conference (CCWC), pp. 1016–1022. https://doi.org/10.1109/CCWC47524.2020.9031194

- Puranik, P.A., 2019. Static Malware Detection using Deep Neural Networks on Portable Executables. UNLV Theses, Dissertations, Professional Papers, and Capstones. https://doi.org/10.34917/16076285