IDENTIFYING FACTORS INFLUENCING ELECTRONIC GAMING MACHINE PLAYER BEHAVIOR USING INTERPRETABLE AI AND MIMICKING PLAYER BEHAVIOR USING REINFORCEMENT LEARNING

by

Gaurav Devendra Jariwala

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

at

Dalhousie University
Halifax, Nova Scotia
November 2022

This thesis is dedicated to my family and friends, who have supported me in every situation.

Table of Contents

List of Tables

List of Figures

Abstract

People in venues like casinos play games on Electronic Gaming Machines (EGMs). These machines do not record information about players, such as their playing experience. The gaming industry is keen on learning about the different types of player behaviors and what leads to these behaviors. In this thesis, we justify an assumption and define whether or not a player had a positive experience during the session based on play session attributes. Furthermore, we identify the factors and their importance that contribute to a positive experience using Interpretable AI. We classify player sessions using a Decision Tree, Logistic Regression, and Explainable Boosting Machine (EBM) to gain insights into the factors used for prediction. EBM gave a comparable performance to that of a state-of-art model with high interpretability and accuracy of 95%. This understanding will provide insights into game performance as well as responsible gaming behaviors. Moreover, to acquire a good evaluation of a machine learning model as a viable alternative for real-world players, we train models that mimic player behavior. We use K-means to cluster different playing behaviors and determine termination states for one of the playing behaviors for Reinforcement Learning. We implemented PPO and ACKTR models to generate the playing behavior, with the agents being rewarded based on their proximity to the termination states. ACKTR performed well as the playing behavior generated by this model were statistically matching the real-world players behavior within the selected cluster.

List of Abbreviations and Symbols Used

ACCI Average Credit to Cash In. 32

ACKTR Actor Critic using Kronecker-Factored Trust Region. 22, 57

AUC Area Under Curve. 35

COCI Cash Out to Cash In. 31

DRL Deep Reinforcement Learning. 8

DT Decision Tree. 10, 37, 57

EBM Explainable Boosting Machine. 7, 12, 16, 17, 39, 57

EGM Electronic Gaming Machine. 1, 5, 44

GAM Generalized Additive Model. 12

IG Information Gain. 11

IGR Information Gain Ratio. 11

IRL Inverse Reinforcement Learning. 8

K-FAC Kronecker-Factored Approximated Curvature. 22

LR Logistic Regression. 9, 36, 57

MDP Markov Decision Process. 20

NAM Neural Additive Models. 59

PPO Proximal Policy Optimization. 21, 57

PW Payout to Wager. 38, 40

RL	Reinforcement Learning. 8, 17, 44, 52, 57
ROC	Receiver Operating Characteristic. 35
TB-COCI	Time-based and Cash Out to Cash In ratio. 34, 57
TBO	Time-based Only. 34, 57
TRPO	Trust Region Policy Optimization. 21
WCI	Wager to Cash In. 40

Acknowledgements

During my Master's journey, I received lots of support from my family, friends, and professors. First and foremost, I am extremely grateful to my supervisor, Dr. Vlado Keselj, for his invaluable guidance and support at every step throughout the process. Without his insightful feedback, this thesis would not have been possible.

Words cannot express my gratitude to my parents, Mr. Devendra Jariwala and Mrs. Sumitra Jariwala. I am forever grateful to my family for their love and for always believing in me. Thanks for the constant encouragement, which has helped me pass through every situation. I would also like to extend my gratitude to my friends who helped me through the difficult times.

Finally, I am grateful to Mitacs for supporting this thesis with funding through the Mitacs Accelerate program.

Chapter 1

Introduction

This chapter explains the motivation behind this thesis, provides an overview of the research and elucidates the contributions of this thesis. This chapter will also provide an outline of this thesis.

1.1 Motivation

Electronic Gaming Machine (EGM), which are gambling machines installed in a range of locations such as casinos, bars, and hotels, have become more and more popular, attracting the attention of not just the gaming industry but also of the government and researchers. The availability of gambling activities has increased thanks to internet gaming [20]. According to the 2018 Canadian Community Health Survey (CCHS), nearly two-thirds (64.5%) of Canadians aged 15 or older (18.9 million) reported gambling at least once in the previous year [62]. These games are highly addictive as there is a chance of winning a high amount of money in a short duration. People continue to spend time and money on gambling even though it affects them mentally and financially, this is called problem gambling [42, 69]. The government is working to tackle this problem by promoting responsible gambling resources, while researchers have conducted studies [3, 49, 54, 15] to identify the gamblers at-risk of problem gambling. Most of the work in this field has been done on limiting problem gambling, however, the gambling industry is interested in learning the psychological traits that contribute to a positive gaming experience in order to enhance the gaming experience of those players who are not having fun. Due to privacy concerns of people, this can not be done by survey. Also, generating virtual players that mimic the real-world players can help industry and research test experiments and conduct a detailed behavioral analysis to understand the behavioral patterns in-depth without identifying the person. Future behavior in the particular situation of a real player could be anticipated using this virtual player.

1.2 Research Overview

The major challenge in working with EGM data is that they are anonymous, which means that these machines do not keep any identifying information about the players in their logs and that they do not distinguish between various players' sessions. In this research, we are using sessions that were sessionized based on balance, and the pauses taken while playing [34], however, these sessions do not contain how was the experience of the player.

We need to address a few research problems in order to categorize the sessions based on the player's experience, classify them, and simulate player behavior using only the characteristics recorded by EGM:

1. What reasonable assumption should be considered while attempting to define the player experience?
2. Are there any interpretable models that can justify the assumption taken in the previous question while also accurately classifying the sessions?
3. What strategy may be utilized to comprehend the EGM's complicated environment and mimic the behavior of a real-world player?

To categorize the sessions based on the player's experience, assumptions were made regarding the amount of time the player spent playing and the total amount of money taken out of and placed back into the machine. Obviously, the player who cashes out more than cash in are having a positive experience and we are also considering the fact that the player is playing for a longer duration and is having fun. To find the factors that are influencing the positive behavior and classify these labelled sessions, we used interpretable AI models such as Logistic Regression (LR), Decision Tree (DT), and Explainable Boosting Machine (EBM). LR gives the weightage of each factor contributing to the prediction, while DT gives the flow of feature selection based on decision rules to classify a session with a positive or a negative experience. To get the advantages of both models, we trained LR on the features selected by the DT. However, EBM not only performed better at classifying the sessions due to its boosting mechanics but also provided insights into all features contributing towards the final prediction.

Furthermore, we developed agents using Reinforcement Learning (RL) to produce such sessions in order to mimic the behavior of real-world EGM players. To mimic one particular player's behavior, we have to separate all types of behaviors. We used an unsupervised learning algorithm, namely K-means for grouping the sessions with similar behavior of players. This similarity is measured using Euclidean distance. Moreover, for replicating a particular behavior we are defining termination states using some selected features and values from one cluster. These termination states are used for reward calculation of the RL models, which get higher rewards as the RL agent approaches towards these termination states. The closeness of the current state of the agent to the termination states is measured by Euclidean distance. We trained two RL algorithms, PPO and ACKTR, to generate sessions with a particular behavior. Both the models performed well as they generated sessions with behavior similar to the selected player's cluster. ACKTR did extremely well at changing the wager considering all aspects of the environment which is a necessary step in order to mimic the behavior of a real-world player.

1.3 Contributions

The four most important contributions of this thesis are as follows:

1. Identifying the factors and their importance in influencing a positive experience of the player playing on EGM. This could help the industry build a better experience for the player.

2. Finding the best performing machine learning model to classify the experience of players.

3. Creating alternatives for real-world players by mimicking their behavior using reinforcement learning.

4. Finding the best reinforcement learning algorithm to generate playing sessions matching statistically with the real-player behavior.

1.4 Thesis Outline

The remaining parts of this thesis are outlined in this section as follows:

Chapter 2 discusses the background and related work. It starts by reviewing gambling behavior studies and explains the working of supervised and unsupervised learning algorithms along with the reinforcement learning algorithm used in this thesis. It further explains how to interpret supervised learning algorithms.

Chapter 3 explains the extraction of new features and removing the invalid sessions. It discusses labelling a session with positive or negative experiences sessions. Furthermore, it interprets models to find the most important factors responsible for a positive experience and compares the models to find the best model for classifying these sessions.

Chapter 4 discusses the transformation of skewed data. It then explains the grouping of players with similar behavior using the clustering technique. It then discusses the determining termination states for the calculation of reward in the reinforcement learning models. Lastly, it explains the generation of sessions and provides a comparison with the original sessions.

Chapter 5 concludes this thesis by highlighting the main finding of this thesis and providing directions for future work.

Chapter 2

Background and Related Work

In this chapter, we explain key concepts needed to understand the background and provide insights into the related work of the methods used in this study. To begin, we will discuss gambling behavioral studies and how it is applied to EGM data. Second, we will review the concepts of Interpretable AI and how it can provide insight into predictive models. Finally, we will go over the concept of Reinforcement Learning and explain how it works to learn gambling behavior.

2.1 EGM Player Behavioral Analysis

Electronic Gaming Machine (EGM) [39] are a common type of gambling machine found in casinos, clubs, and other public areas where people congregate for recreation. Although these devices, which use sophisticated technology, are actually computers, many of them still have reels that purport to spin and are evocative of earlier gambling machines. A random number generator is the base of every EGM. The computer retrieves the numbers created at that moment and transforms them into a display on the screen when a button or touch screen is pressed. The numbers represent a location on a reel map (the quantity and arrangement of symbols on each virtual reel) and a pay table (the payouts for any combination of symbols appearing on a line). For instance, the pay table will be used to map the random process's generation of three cherries to a payout of, say, two credits. These machines don't keep track of most of the play data and are stateless. Loyalty cards [65] is a major update that certain venues have implemented that are used to track customer information in the casino. As a result, well-formed data that takes into account playing sessions, games played, and money spent is produced. With this, the sessionizing task is entirely relinquished, as well as a history of user play data is also provided. Loyalty cards are not required and are not even used by the majority of venues [29], therefore typical data processing is still in use.

Latifi [34] did excellent work at sessionizing user datasets. EGM logs are made up of game data that is gathered during playtime along with additional meta data. No user ID or other attribute that can be used to identify users is included in these logs. Although there are instances where some players play with two machines simultaneously, the fundamental understanding is that a player only uses one machine. The sessionizing process is intuitively based on two reliable assumptions. The first assumption is that a cash-in is there before the start of each game session. The second assumption is that a gaming session ends either when a player cashes out almost all of their winnings or when they play almost all of their remaining credits before turning the machine off. The first assumption was not subject to any threshold consideration, but the second assumption required to take into account of minimum cash balance and idle time threshold values. When the time gap exceeds the idle time threshold and the machine credit is less than the minimum credit, the session is declared terminated using these two thresholds. In order to secure some wins, it is assumed that numerous cash-outs are permitted within a session. The researcher experiments show that sessions with the actual player session duration are derived when the idle time of the machine is about a few minutes and a minimal amount of money in the machine is around a couple of dollars and cents. Our research will use data that has been sessionized using this method, hence this study is important to us.

Intensive research and studies have been done on detecting the persona of gamblers and predicting gamblers at-risk of problem gambling [3, 49, 54, 15]. Typically problem gamblers struggle to control their urges to gamble excessively, regardless of the harm that their behavior may inflict on others (such as family, friends, or coworkers) and also frequently occurs in conjunction with other negative habits such as food disorders and substance abuse [51, 42, 10]. To detect problem gamblers, researchers have used unsupervised learning techniques like clustering to identify various gambling behaviors. The identification of a new group of players who probably developed a medium risk of disordered gambling behavior that was not recognized by Braverman and Shaffer [15] was made possible by the indicators proposed by Adami et al. [3] based on wager volatility over time and the number of different games played on the website.

Mosquera and Keselj [49] worked on EGM data and since player identity information is not contained in EGM data, they assumed sessions contain relevant gameplay classes, and the session starts with no money and ends with either winning or no winning to cash out. They also used the k-means algorithm and criteria like Braverman [14] to identify clusters. For comparing different clusters, they performed ANOVA and Tukey's Honestly Significant Difference (HSD) test. Latifi [34] suggested using DBSCAN, which may identify results that K-means cannot identify, and can be used prior to K-means to further refine the results. Although the research uses a Multivariate Convolutional LSTM neural network to quickly classify playstyle, it does not significantly improve performance when more than 40 transactions are analyzed. The inability of the models to be interpreted is a drawback of this study.

In order to determine the significance of knowledge extraction and algorithm interpretability, as described by Percy et al. [57], the researchers conducted a survey of the participants during a similar presentation at the 2016 New Horizons in Responsible Gambling conference. When asked which option they would prefer—a model that gave a 75% correct assessment that was completely interpretable and accountable, or a responsible gambling assessment algorithm that provided a 90% accurate assessment of problem gambling risk which they were unable to comprehend. Only 20% preferred the model with more accuracy, with 70% choosing to give up 15 percent more accurate model for better interpretability (10% were unsure or thought it depended on the situation). Sarkar et al. [64] work focuses on data obtained from the regulated Internet gambling jurisdiction of Ontario, Canada. TREPAN [17] is an algorithm that induces a decision tree that approximates neural network with a high degree of fidelity. Unlike Percy et al. [57], who only used TREPAN on neural networks, they also used it on random forests. Furthermore, they came to the conclusion that while random forests produce the most accurate predictions, a neural network produced the decision tree that performed the best. This appeared to offer the best accuracy-to-interpretability trade-off in this research since it was using a simpler form of rules than other best performing trees. Though TREPAN gave the promising results, we propose to use Explainable Boosting Machine (EBM) , which is relatively simpler, more accurate, and faster to execute the model. Harsh et al. [22] built a simulation of a financial trading app to help analysis risk-taking and enjoyment characteristics in investing app behaviors.

The advancement in the deep neural network has brought tremendous growth in the field of Reinforcement Learning (RL). Deep learning makes it possible for RL to scale to previously unsolvable decision-making issues; i.e., situations with high-dimensional state and action spaces [7]. Researchers have applied deep reinforcement learning (DRL) to a wide range of domains, such as robotics [35, 36] and video games [47, 73, 56, 74]. In the disciplines of psychology and neuroscience, human decision-making has been researched for years. Evidence that the brain uses reinforcement learning algorithms was found when it was discovered that there was a relationship between dopaminergic neuron activity in brain activity and reward prediction errors [68]. Inspired from Sergey et al. work [35], Wu and Izawa [76] studied incorporating the emotion of regret into reinforcement learning formulation and investigated how regret affects motivation and reinforcement learning in the application of problem gambling. They mathematically defined the term 'regret' as a maximum reward minus the current reward. By limiting the incentive for lower rewards, regret can increase learning for the optimal solution while decreasing learning for a sub-optimal solution. Their suggested algorithm, regret reinforcement learning, exhibited behavior similar to that of addicted gamblers by choosing a high-risk, high-reward option when a high reward was discovered by chance. Another approach, Inverse Reinforcement Learning (IRL) [53], which extracts the reward function of a problem given observation and optimal behavior, uses this learned reward function as is if the subject agent shares the same environment, actions, and goals as the other, otherwise it continues to provide a useful basis when the agent specifications differ mildly [5]. Although IRL has been used to infer human goals and modeling behavior [9, 72, 32, 33, 52, 30], it requires well-defined environment and optimal behavior trajectories which is difficult to produce from EGM.

2.2 Supervised Learning

Supervised learning is a form of machine learning where the algorithm needs labelled data, also known as tagged data, for training. The term "labelled data" refers to data that has been assigned a label or tag (target variable). When training is finished, only input data points are provided to the algorithm to check whether the predicted label matches the original label. This is done to assess how well the algorithm is working. This is also known as test data or unseen data. The objective of supervised learning algorithms is to train a function that can map input features to their appropriate labels and also generalize well to unseen data.

There are basically two types of supervised learning:

- Regression: It creates a function that demonstrates the connection between its continuous target variable and its input variables. It is commonly used for forecasting sales revenue, market trend, etc.
- Classification: The target variable used in classification are categorical in nature. These models take the input variables and predict the type of category it belongs to. For example, predicting an e-mail is spam or ham.

We will only briefly describe a few of the classification models that were used in this study due to its limited scope.

2.2.1 Classification

In this section, we will briefly go through the classification model used during this research.

Logistic Regression

Logistic Regression (LR) is a statistical model that estimates the probability of the target variable upon a given input. This algorithm is mainly used for binary classification. It is a linear regression model [70] extension for the categorization issue. The model only predicts values between 0 and 1, since the model gives the probability of a category. The main difference between linear regression and logistic regression is that it uses a natural logarithm of odds, which is calculated by dividing the probability of success by the probability of failure, as a coefficient. The logistic function is defined as:

$$logistic(x) = \frac{1}{1 + \exp(-x)} \tag{2.1}$$

As this function returns values between 0 and 1, we need to select a threshold value above which it is classified as category 1 otherwise classified as category 2. Commonly the threshold value is set to 0.5. This is also known as the decision boundary.

There are three types of logistic regression:

1. Binary logistic regression: In this method, there are only two possible categories for the target variable. For instance, the type of cancer is benign and malignant.
2. Multinomial logistic regression: In this type, the target variable contains three or more possible categories with no set rank. For example, a vehicle could be a car, truck, or bus.
3. Ordinal logistic regression: In this kind, the target variable could fall into three or more different categories, but in a particular order. For example, a review of a restaurant from 1 to 5.

The cost function for linear regression is mean squared error. This will be a non-convex function of parameters if this is used for logistic regression which may result in being stuck at local minima. To resolve this issue, we use log loss, also known as cross-entropy loss. The cost function of logistic function [50] is defined as:

$$J(\Theta) = \frac{1}{m} \sum [-y^{(i)} \log(h_\Theta(x^{(i)})) - (1 - y^{(i)}) \log(1 - h_\Theta(x^{(i)}))] \tag{2.2}$$

Decision Tree

Decision Tree (DT) is a non-parametric algorithm meaning it does not make strong assumptions about input and output mapping functions. It is used for both classification and regression tasks. It has a tree-like structure, as the name would imply, with a root node, internal nodes, branches, and leaf nodes. The root node and the internal nodes represent the features, while the branches show the rules or decisions, and the leaf nodes show the outcome of the decision tree. It finds the optimal split points using the divided and conquer method by conducting a greedy search. The process of dividing the tree is done in a recursive fashion. The purity of the leaf node is given by the number of categories present within that particular node. The node is called pure if all the data point in that node is classified in a single category. The purity depends largely on the complexity of the tree, but if the tree is too complex, it may lead to a problem known as overfitting [13]. Pruning, which is a technique that crops the branches which split the features that are not important, is used to lower the complexity of the tree and avoid overfitting. To get the best split and features, a decision tree can use multiple techniques like Information Gain, Information Gain Ratio, and Gini Index.

To understand Information Gain, we need to define Entropy. Entropy in simple words could be defined as a measure of the impurity of a set containing multiple categories of data points. In other words, given a node containing p positive and n negative data points then the entropy of that node is given by the formula:

$$Entropy = -p \log_2(p) - n \log_2(n) \tag{2.3}$$

This formula could be generalized to:

$$H(D) = -\sum_{i=1}^{c} p_i \log_2 p_i \tag{2.4}$$

where, entropy H is given by the probability of category i from the set of categories c in the given dataset D.

Information Gain (IG) measures the change in entropy based on the particular features before and after the splitting of the dataset. The split will be optimally determined by the feature that has the maximum information gain since it performs the best job of classifying the training data in accordance with its intended category. The Information Gain is represented by the formula:

$$IG(D, A) = H(D) - \sum_{v \epsilon Values(A)} \frac{|D_v|}{|D|} \times H(D_v) \tag{2.5}$$

where, v represent the value in the attribute A in the given dataset D.

Since Information Gain selects the features with multiple unique values, Information Gain Ratio (IGR) is used as it reduces bias by scaling Information Gain by the entropy.

$$IGR = \frac{IG}{Entropy} \tag{2.6}$$

Gini Index is the probability that a randomly chosen data point in the dataset would be wrongly classified. The formula of the Gini Index is given as:

$$GiniIndex = 1 - \sum_{i=1}^{n} (p_i)^2 \tag{2.7}$$

where, p_i is the probability of a data point being classified into a particular category.

Gini index value is in the range from 0 to 1, where 0 represents that all the data points belong to one particular category while 1 indicates the random distribution of data points among multiple categories. The Gini index of a value of 0.5 represents categories that are equally distributed.

Explainable Boosting Machine

Explainable Boosting Machine (EBM) [55] is a glass-box model, which has high interpretability along with accuracy as state-of-the-art machine learning algorithms like Random Forest and Boosted Trees. The idea behind the EBM was inspired by Generalized Additive Model (GAM) [26]. GAM is represented in the form:

$$g(E[y]) = \beta_0 + \sum f_i(x_j) \tag{2.8}$$

where, the GAM is adjusted to various models, such as regression or classification, through the link function g.

The problem with GAM is that it ignores the effects of co-linearity between features. To overcome this issue, GA^{2}M [40] was introduced with the functionality of pairwise interaction which is in a form

$$g(E[y]) = \beta_0 + \sum f_i(x_j) + \sum f_{ij}(x_i, x_j) \tag{2.9}$$

EBM learns this function f of GA^{2}M using the latest machine learning technique called gradient boosting. This is the major difference between these two algorithms. EBM is trained in the same way as gradient boosting, but with some notable differences. They are trained with many small trees, and each tree is trying to comprehend the mistakes done by the tree before it. The boosting method is meticulously limited to training on a single feature at a time in round-robin form with a very low learning rate, making the order of the features irrelevant. Due to this, more number of iterations are required compared to typical gradient boosting algorithms like XGBoost. The algorithm is parallelizable and is faster compared to GA^{2}M.

After collecting all the trees, we group them based on a feature to produce contribution plots or lookup tables for each feature. They keep track of how much each feature value contributed to the final prediction. They are basically the function f in the equation 2.9, learned by boosting. To make a prediction, each feature's lookup table is used to get their contribution, which is then added up and simply pass to the link function g to compute the final prediction. This makes EBM execute very fast at the prediction time.

2.3 Unsupervised Learning

Unsupervised learning is a type of machine learning that groups similar data points without any supervision or labelled data. Unlike supervised learning, it does not have any labels to differentiate one data point from another, it uses the underlining structure of the data and finds patterns to make groups with alike data points. Clustering and dimensionality reduction [23] are two extremely basic yet well-known examples of unsupervised learning.

2.3.1 Clustering

Clustering is the most common unsupervised learning technique used to segregate the data points into different groups, also called clusters, such the homogeneity of data points within the cluster is high. Although the clustering process is not simple or widely acknowledged, we try to explain the commonly procedure [78] taken. Firstly, the features are selected and some are transformed which affects clustering a lot [28, 12, 27]. Secondly, the clustering algorithm is designed and selected by defining criterion and proximity measures. The next step is to validate the cluster using some testing criteria [46, 27]. Finally, the clusters are analyzed to get some meaningful insights.

There are many types of clustering algorithms based on various measures and criteria [60, 21, 46, 27, 28]. We will briefly describe the K-means algorithm as it has shown promising results for detecting EGM player behaviors [34, 49].

K-means

K-means is the most frequently used clustering algorithm for grouping data points into clusters. The K in K-means stands for the number of clusters which is a hyper-parameter meaning it is a user defined number. There are multiple ways [59, 31] to come up with the optimal value of K, but the most common technique used is the elbow method [45, 18]. This algorithm first selects the K points randomly that acts as the centroids of clusters. It then calculates the distance, generally Euclidean distance, between each data point and the centroids and assigns the data points to the cluster with the smallest distance. New centroid values of each cluster are calculated by averaging all the data points within the cluster after they are assigned to a particular cluster, this describes the 'means' part of K-means. This procedure is repeated until there is no change in the centroids' value or there is a small change in the values of centroids from the previous iteration.

K-means is a very simple algorithm to implement and scales to large datasets, but the cluster depends largely on the initial values of centroids selected. To tackle this problem, K-means++ [6] was introduced that initialize only the first centroid randomly while selecting all other centroids such that they are farthest from each other. Despite the fact that it's computationally more expensive than K-means, but this results in a better selection of clusters.

2.4 Interpretable AI

Industries are not only looking for models with more accuracy but also are interested in understanding the reasons behind the results of the models. This is where Interpretable AI comes in, which explains the underneath working of the models in a form that is understandable by humans. The degree of interpretability is measured by how well the decision made by the model is comprehensible to humans. The need for interpretability arises from incompleteness in problem formalization [19]. Machine learning models are not always to be trusted as these models might be considering some features in another way that humans might assume those features should be used, this may lead models to be biased. Though some industries may not require the models to be interpretable, but in a field like healthcare, interpretability could be found very useful.

Interpretable AI has given promising results in understanding gambler behavior [57, 64]. Gambling companies and government can use this understanding of the behavior to tackle problem gambling and make sure of responsible gambling practices. Interpretable AI is used in this research to find the factors that are influencing player behavior to play for a longer duration which could help the gambling industry to identify factors to make the player play for a longer time to increase their revenue and also can help gamblers at-risk.

2.4.1 Accuracy-Interpretability Trade-off

It's very difficult to have models with high accuracy and high interpretability, for example, Neural Networks give high accuracy, but due to their complexity, they are very difficult to interpret, while on the other hand, Decision Trees are relatively simpler models, but with low accuracy. The trade-off between accuracy and interpretability has been shown by Figure 2.1.

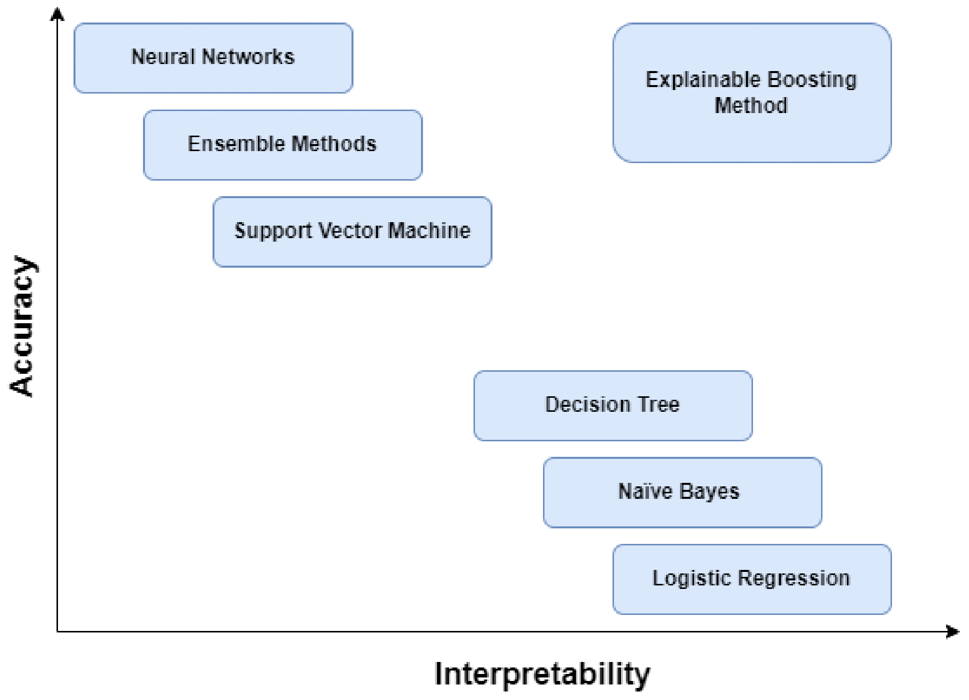


Figure 2.1: Accuracy-Interpretability Trade-off

There are some problems that require sophisticated deep neural networks to solve, such as text or image processing. Using techniques like LIME [61] and Shapley values [41] on a black-box model that has already been trained can help make it interpretable. LIME work as a surrogate model, which is trained on top of the black-box to approximate its predictions. It can only explain one prediction at a time as it uses interpretable models as surrogate models. Shapley values calculate the contribution of each feature toward the prediction. This gives a global interpretation of the model. Though both these methods are used extensively by researchers, a LIME approximation could be inaccurate resulting in bad interpretation and Shapley values are difficult and computationally expensive to compute with high numbers of features. So to overcome this trade-off, we can use Explainable Boosting Machine (EBM), which gives high accuracy along with high interpretability. We will shortly explain how this algorithm could be interpreted.

2.4.2 Interpretable Algorithms

In this section, we will briefly discuss how to interpret algorithms that have been used in this research.

Logistic Regression

As the logistic function (equation 2.1) output the probability, the coefficients or weights are not linearly associated with the prediction like linear regression. So in logistic regression, a change in a feature by one unit changes the odds ratio (multiplicative) by a factor of $\exp(\beta_j)$ [48]. The log odds ratio is increased by the value of the associated weight for each unit change in x_j. This could be defined as:

$$Odds\ Ratio = \exp(\beta_j). \tag{2.10}$$

Logistic regression is highly interpretable, but its lacks of performance produce a model with lower accuracy as depicted by Figure 2.1.

Decision Trees

Decision Trees are made of rules and features, so they are very easy to interpret. The branches indicate which sections you are looking at as you proceed through the nodes, starting with the root node. When you arrive at the leaf node, the node notifies you of the anticipated result. All the branches are connected by 'AND' [48].

In a decision tree, a feature's overall relevance can be calculated by checking the variance or Gini index in relation to the parent node for each split for which the feature was applied. All importance is scaled to a total of 100. As a result, it is possible to interpret each importance as a portion of the overall model importance.

Explainable Boosting Machine

Explainable Boosting Machine (EBM) are some of the exception models of accuracy-interpretability trade-off (Figure 2.1), as they are highly interpretable models with high accuracy. As discussed in section 2.2.1, EBM creates a lookup table for each feature, which is then plotted for visualization and interpretability. Each feature contributes separately toward the prediction and as EBM is an additive model, the importance of the features and their impact can be depicted.

2.5 Reinforcement Learning

Reinforcement Learning (RL) is a subfield of machine learning in which an agent (learner) acts in a specific circumstance to maximize reward. It learns the best policy (behavior) within an environment by taking the best action for a specific situation in order to obtain the biggest reward, but it must think about the fact that the future reward and situation are also dependent on it. The two most crucial distinguishing properties of reinforcement learning are trial-and-error search and delayed reward [71]. The Figure 2.2 shows the general flow of a reinforcement learning model. Here the action is the move that an agent can make from its available action space. The state represents the current circumstance that is given by the environment. Finally, the reward is the immediate gain earned by the agent upon action performed.

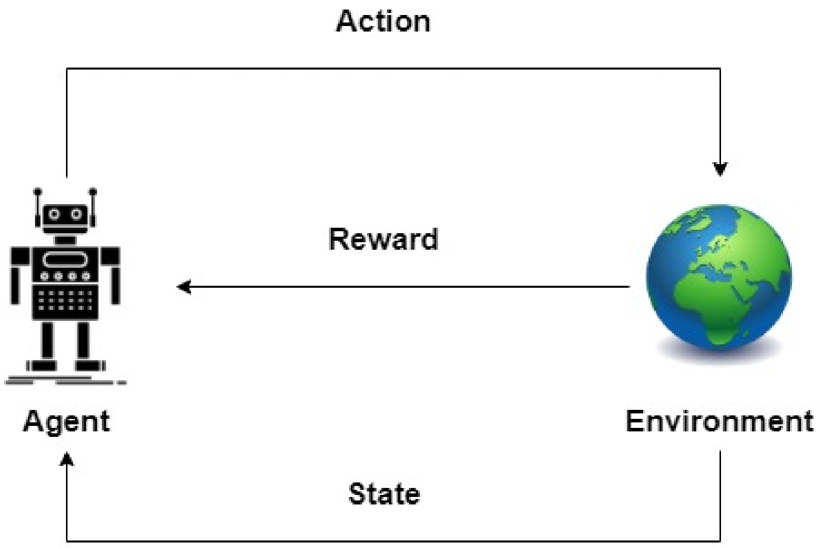


Figure 2.2: Reinforcement Learning Model

The trade-off between exploration and exploitation is one of the difficulties that reinforcement learning faces, as opposed to other types of learning. A reinforcement learning agent must favor actions that it has previously done and proven to be effective in creating rewards if it wants to gain a lot of rewards. However, it must try acts that it has never chosen before in order to find such actions. The agent must exploit the advantage of its past experiences in an attempt to benefit, but it must also explore making better decision-making in the future. The main goal of an RL algorithm is to find a balance between exploration and exploitation.

2.5.1 Elements of Reinforcement Learning

A reinforcement learning system primarily consists of four subelements: a policy, a reward signal, a value function, and, if present, a model of the environment [71].

1. **Policy:** A policy is a mapping between perceived environmental states and the actions that should be taken when those states are present. The policy may be as straightforward as a function or lookup table in some circumstances, while in others it may need complex computations like a search procedure. Policies may, in general, be stochastic, defining probability for every action. In simpler words, a policy defines the behavior of an agent at a specific moment.

2. **Reward signal:** The agent's primary goal is to increase its overall reward over the long term. The key driver behind changing the policy is the reward signal; if a policy selected action is followed by a little reward, the policy may be modified to choose a different action in that circumstance in the future. Reward signals could, in general, be stochastic functions of the environment's state and the actions conducted.

3. **Value function:** A value function explains what is beneficial over the long term, while the reward signal thinks for the current state. A state's value can be thought of as the total amount of reward that the agent can anticipate accumulating in the future, beginning from that state. Not the highest reward, but the highest value states are what we aim to achieve and select agent action on because they will ultimately yield the greatest rewards for us.

4. **Model:** This is something that imitates the behavior of the environment or, to put it more broadly, something that enables predictions about the future behavior of the environment. Models are used for planning, which we define as any method of selecting a course of action by factoring in potential future circumstances before they actually occur. Model-based approaches to reinforcement learning are distinguished from simpler model-free approaches that explicitly rely on trial-and-error learning by their use of models and planning.

2.5.2 How does Reinforcement Learning Work

When an agent interacts with an environment, at each time step t, the agent obtains a state s_t in a state space S and chooses an action at a_t from an action space A, following a policy $\pi(a_t|s_t)$, receives a reward r_t, and transitions to the next state s_{t+1}, according to the environment dynamics, or model, for reward function $R(s,a)$ and state transition probability $P(s_{t+1}|s_t, a_t)$ respectively. In an episodic problem, this process continues until the agent reaches a terminal state, and then it restarts. The return, equation 2.11, is the discounted, accumulated reward with the discount factor $\gamma \epsilon [0,1]$ [38].

$$R_t = \sum_{k=0}^{\infty} \gamma^k r_{t+k} \tag{2.11}$$

It is possible to model reinforcement learning problems as a Markov Decision Process (MDP) using Markov property [43], which states that the future states are only dependent on its current state; hence, given the present, the future is independent of the past. The topic is referred to as a finite MDP if the state and action spaces are finite. MDP can be defined as a 5 elements tuple (S, A, P, R, γ) [11]. It assumes that the outcome of an action taken in a given state depends only on the current state-action pair and not on the past states and actions [79], that is,

$$\mathbb{P}(S_{t+1}|S_t, a_t, S_{t-1}, a_{t-1}, ..., S_0, a_0) = \mathbb{P}(S_{t+1}|S_t, a_t). \tag{2.12}$$

The formulation of the policy is the aim of reinforcement learning algorithms. A policy π is a function that identifies the course of action to be taken in each state in order to accomplish the goal of maximizing the cumulative discounted reward (equation 2.11) [25]. To find the optimal policy π^*, we first need to find the value function. A value function is a prediction of the expected, accumulative, discounted, future reward, measuring how good is each state, or state-action pair [37]. The state value,

$$v_\pi(s) = \mathbb{E}[R_t|s_t = s] \ \textit{where} \ R_t = \sum_{k=0}^{\infty} \gamma^k r_{t+k}, \tag{2.13}$$

is the expected return for following policy π from state s. The action value,

$$q_\pi(s, a) = \mathbb{E}[R_t|s_t = s, a_t = a] \tag{2.14}$$

is the expected return for selecting action a in state s and then following policy π. Value function $v_\pi(s)$ decomposes into the Bellman equation:

$$v_\pi(s) = \sum_a \pi(a|s) \sum_{s',r} p(s', r|s, a)[r + \gamma v_\pi(s')]. \tag{2.15}$$

An optimal state value,

$$v_*(s) = \max_\pi v_\pi(s) = \max_a q_{\pi^*}(s, a), \tag{2.16}$$

is the maximum state value achievable by any policy for state s, which decomposes into the Bellman equation:

$$v_*(s) = \max_a \sum_{s',r} p(s', r|s, a)[r + \gamma v_*(s')]. \tag{2.17}$$

Action value function $q_\pi(s, a)$ decomposes into the Bellman equation:

$$q_\pi(s, a) = \sum_{s',r} p(s', r|s, a)[r + \gamma \sum_{a'} \pi(a'|s')q_\pi(s', a')]. \tag{2.18}$$

An optimal action value function,

$$q_*(s, a) = \max_\pi q_\pi(s, a), \tag{2.19}$$

is the maximum action value achievable by any policy for state s and action a, which decomposes into the Bellman equation:

$$q_*(s, a) = \sum_{s',r} p(s', r|s, a)[r + \gamma \max_{a'} q_*(s', a')]. \tag{2.20}$$

Optimal policies have the same optimal action value function q_*.

2.5.3 Reinforcement Learning Algorithms

Proximal Policy Optimization

Model-free policy search techniques, such as policy gradient approaches, are helpful for updating the policy [58], but the problem with policy gradient is finding the right step size for updation, as they are sensitive. To eliminate this problem, researchers came up with an approach called Trust Region Policy Optimization (TRPO) [66], which applied a trust region restriction to the objective function in order to reduce the KL divergence between the existing and new policies to make sure that the new policies are not too far from the old policies. Theoretically, this can be supported by demonstrating that improving the policy within the trust region results in a guaranteed improvement in monotonic performance. TRPO is computationally inefficient for large-scale tasks, and when applied to sophisticated network architectures, it is challenging to scale up for those situations [75]. By using a clipping technique to avoid totally imposing the hard restriction, Proximal Policy Optimization (PPO) [67] greatly decreases complexity and is able to employ a first-order optimizer, such as the Gradient Descent method, to optimize the objective function which is defined as:

$$L^{CLIP}(\theta) = \widehat{\mathbb{E}}_t[\min(r_t(\theta)\widehat{A}_t, clip(r_t(\theta), 1 - \epsilon, 1 + \epsilon)\widehat{A}_t)], \tag{2.21}$$

where θ is policy parameter, $\widehat{\mathbb{E}}_t$ is the expectation over time t, r_t is the probability ratio of old and new policies, $\widehat{A}_t$ is the estimated advantage at time t, and ϵ is a hyper-parameter. By attempting to remove the reward for moving the policy away from the previous one when the probability ratio between them is outside of a clipping range, this objective function eliminates the KL constraint of TRPO while maintaining the execution of a Trust Region update.

PPO performs better overall for a broad range of tasks and is relatively easy to implement and tune while preserving the stability and reliability of a TRPO.

Actor Critic using Kronecker-Factored Trust Region

Actor Critic using Kronecker-Factored Trust Region (ACKTR) [77] uses actor-critic methods in which the actor performs an action while the critic estimates the value function, distributed Kronecker factorization [44], and trust region optimization [66]. It creates a scalable approximation of the natural gradient using the Kronecker-Factored Approximated Curvature (K-FAC) [44, 24]. K-FAC uses a Kronecker-factored approximation to the Fisher matrix to perform efficient approximate natural gradient updates. It approximate small block F_l corresponding to layer l as $\widehat{F_l}$ by calculating:

$$F_l \approx \mathbb{E}[aa^T] \otimes \mathbb{E}[\nabla_s L (\nabla_s L)^T] := A \otimes S := \widehat{F_l} \tag{2.22}$$

By assuming that there is no correlation between the second-order statistics of the activations and the backpropagate derivatives, this approximation can be understood. The Fisher metric for RL objectives is defined as:

$$F = \mathbb{E}_{p(\tau)}[\nabla_\theta \log \pi(a_t|s_t)(\nabla_\theta \log \pi(a_t|s_t))^T], \tag{2.23}$$

where $p(\tau)$ is the distribution of trajectories stated as:

$$p(s_0) \prod_{t=0}^{T} \pi(a_t|s_t) p(s_{t+1}|s_t, a_t). \tag{2.24}$$

The Fisher matrix is used to update both the actor and the critic by approximating it by applying K-FAC. ACKTR then applies trust region formulation of K-FAC [8] to update the policy distribution. With both discrete and continuous action spaces, ACKTR is adaptable to learning the model's action probability distribution from an observation. It returns the probability mass for discrete action spaces whereas it returns the probability density for continuous action spaces [16].

Chapter 3

Identifying Importance of Enjoyment Factors

Many people play EGM every day; some of them play for longer periods of time, while others simply play for a brief duration. The gaming venues and companies and other stakeholders have an interest in discovering what motivates these players to play for longer periods of time. Though playing for a longer period of time does not always imply that the player is having fun; the player may simply be playing more to make up for the lost money. The goal of this study is to determine the significance of factors that promote a longer duration of playing that the player enjoys.

In this chapter, we will discuss the process of identifying the importance of enjoyment factors. Figure 3.1 shows the flow of this process. It proceeds by discussing how features were extracted from data and preprocessed, followed by a detailed data analysis. Following that, we will go over some of the assumptions that were used to determine whether or not the player enjoyed the session. Furthermore, it interprets the machine learning models and compares results using evaluation metrics.

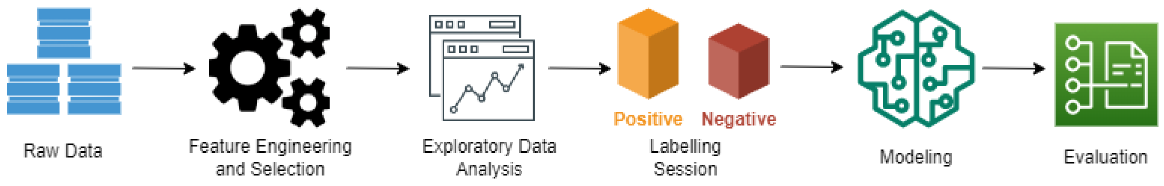


Figure 3.1: Factors Identification Flow

3.1 Dataset Description

The data collected by EGM is anonymous in order to safeguard the privacy of those who use it. This makes working with EGM data challenging. For this study, we will use game data that has been sessionized using the sessionization approach [34] explained in section 2.1. EGM data has a fairly limited set of information such as transaction type, transaction amount, transaction timestamp, and amount machine returns on that transaction, though additional features can be derived using these features, as explained in the next section.

3.1.1 Feature Engineering and Cleaning

Many characteristics have been extracted from the sessions' initial features. Some of these features are mentioned and explained in Table 3.1.

Attribute	**Explanation**
Reward	% of wins, where win is greater than wager, # of wins (>wager) divided by number of wagers
Losses Disguised as Wins	% of wins, where win is less than wager, # of wins (<wager) divided by number of wagers
Illusion of Control	Number of times the player changed wager
Bonus Round	Frequency of bonus rounds
Total cash in amount	The total amount of money inserted by the player in the machine during a play session
Average primary wager	The average primary wager in a play session
Number of distinct primary wagers	Number of distinct primary wagers in the session
Session length	Elapsed time is the amount of time that passes from the start of a session to the end of the session.
Average time between spins	The average of the time difference between consecutive spins
Total cash out	Total cash out in a session
Starting cash in	Total amount of cash in at the start of the session before he starts playing
Additional cash in	Total amount of additional cash in after a player starts playing
Loss Percentage	Total number of loss divided by total number of wagers
Intensity	Intensity (wagers/minute) in a session
Cash out occurrence flag	A flag to specify whether a player cashed out or ran out of money
Cash out to cash in ratio	The ratio of cash out to cash in
Number of cash in	Number of times a player inserted money in a machine
Pause to Session Duration Ratio	Sum of pause time divided by the total session duration
Payout to Wager (PW) ratio	Total payout divided by total wager in a session

Table 3.1: Features Explanation

These extra features, such as intensity, which defines how fast or slow a player plays, help in understanding player behavior.

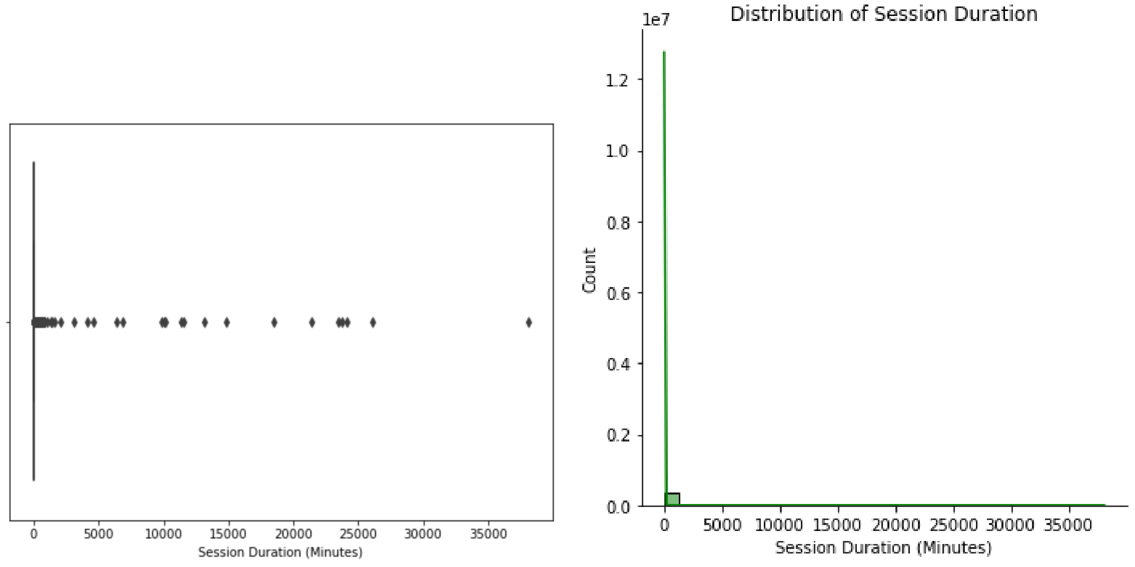
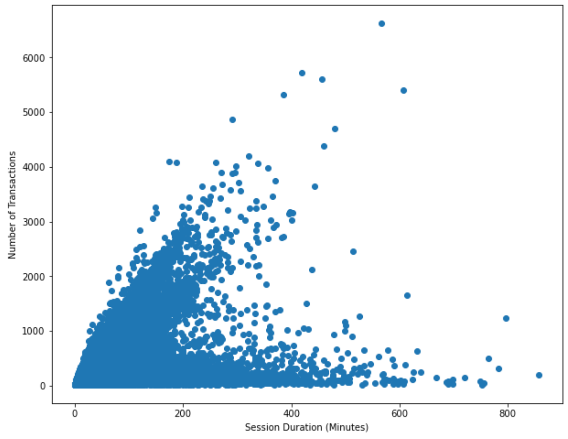


Figure 3.2: Distribution of Session Duration

Some of the sessions were classified with a very long duration which spans multiple days, which is evident from the Figure 3.2 which shows the distribution of the session duration. To identify such invalid sessions, we are assuming that if a session with a Pause to Session Duration (PS) ratio is more than equal to 0.5, meaning that if a person has taken pauses of more than 50% of the session duration, then that session is considered as an invalid session. From all the sessions, 5.2% of sessions have a PS ratio of more than equal to 0.5. All these sessions are considered to be invalid and removed from the dataset.

In an ideal scenario, the user would start with no money in the machine credit meter, but we'll assume the session starts with less than a $1 as there might be some money left in the machine by the previous user. Around 3360, or 11% of sessions, had a dollar or more at the start. These sessions were deemed invalid. Only a few sessions had more than 1000 transactions, while the majority of them had less than 500 transactions. Additionally, sessions with fewer than ten transactions were excluded because they provide little information on the user's behavior.



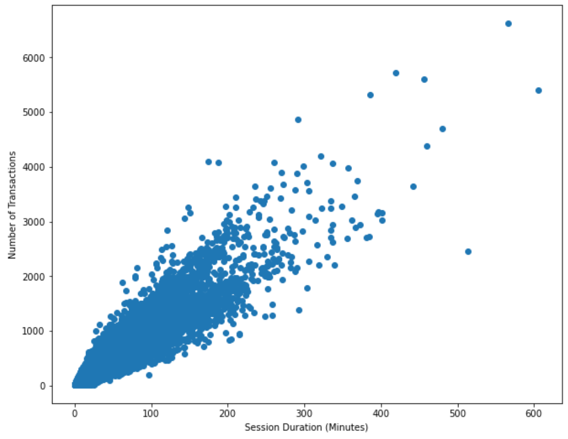


Figure 3.3: Number of Transaction with Session Duration

Figure 3.3 compares session time versus the number of transactions with invalid sessions, defined as sessions with a PS ratio of more than 0.5 and one or more dollars prior to the commencement of the session. The chart shows that, despite the low number of transactions, session time was excessive in several sessions. After excluding the invalid sessions, the right figure (genuine sessions) indicates a clear positive correlation between the number of transactions and session length.

3.2 Data Preparation

Generally, characteristics in the dataset have various scales of range; for example, cash in amount is normally greater than transaction number since both of these features are recorded in different units. As a result, if these features are fed into the predictive model as is, the model will be biased toward the characteristic with the larger scale. To address this issue, normalization is performed, which scales all features to the same range. Although there are several types of normalizing methods, the following two are the most commonly utilized.

1. **Min-Max Normalization:** The values are scaled from 0 to 1 using this technique. It maintains the relationships between the original data values, though it reduces the impact of outliers.

$$x_{scaled} = \frac{x - x_{min}}{x_{max} - x_{min}} \tag{3.1}$$

2. **Z-Score Normalization:** It normalize the data such that the mean (μ) of the feature distribution is 0 and the standard deviation (σ) is 1.

$$z = \frac{x - \mu}{\sigma} \tag{3.2}$$

We have chosen the z-score normalization as it does not suppress the outliers.

3.3 Data Analysis

Data Analysis is an important step to perform visualization of data to get insights into data.

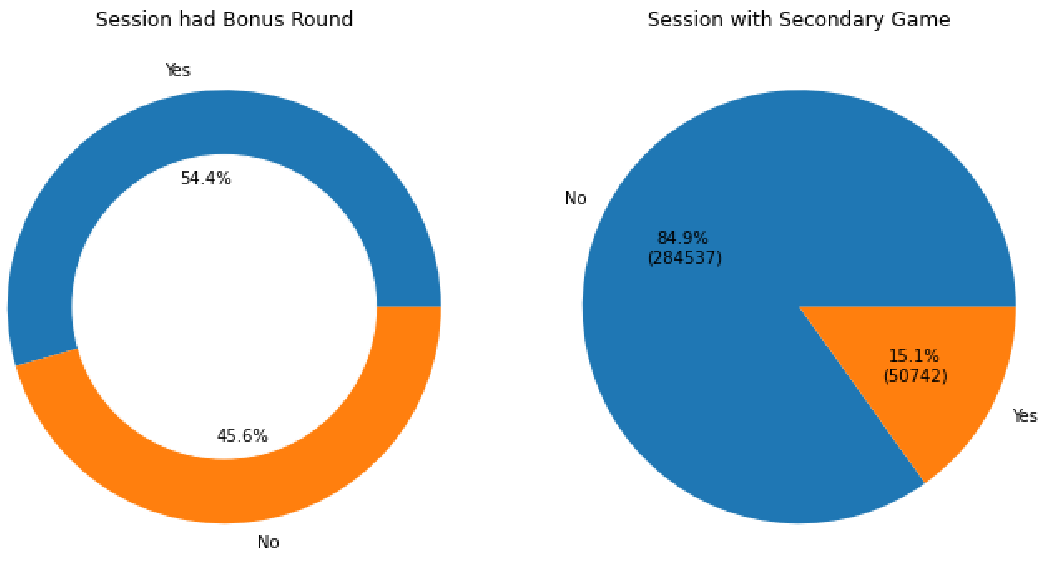


Figure 3.4: Bonus Round and Secondary Game Distribution

It is interesting to note that more than half of the sessions included bonus rounds. In addition, only 15.1% of sessions had participants playing a secondary game, while the rest of the sessions had no secondary games.

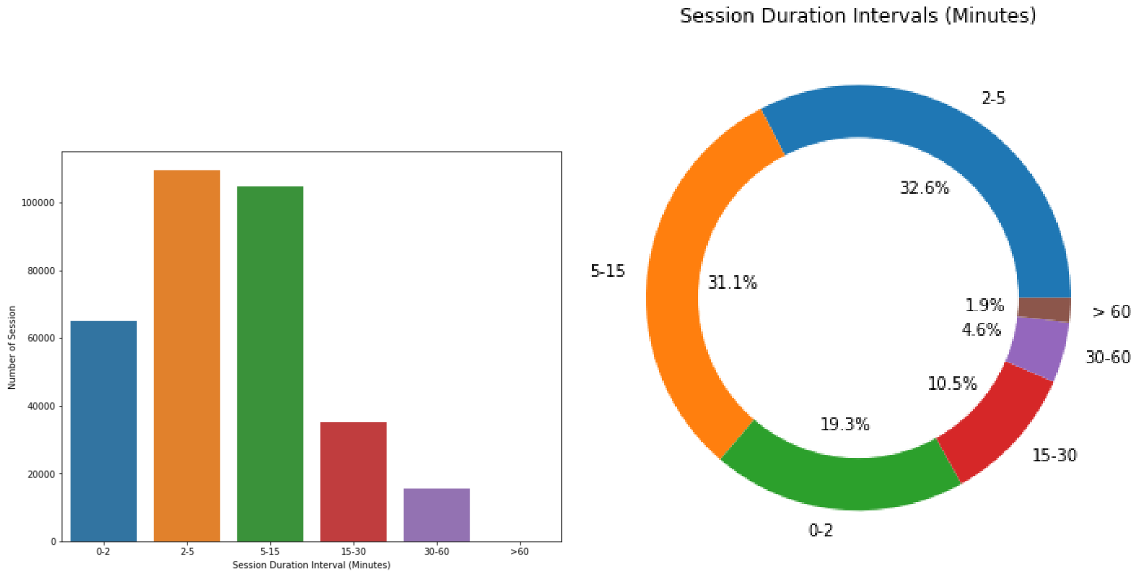


Figure 3.5: Session Intervals Distribution

Sessions were segmented into multiple session duration intervals (Figure 3.5) to have a better understanding of the factors impacting session duration. Intervals of 2–5 minutes and 5–15 minutes have nearly equal percentages of sessions, so we can estimate that roughly 60% of people play for 2–15 minutes, while 19.3play for only 0–2 minutes. About 4.6% and 1.9% of people, respectively, play lengthier sessions spanning 30–60 minutes and more than 1 hour.

Most individuals prefer to add cash to the machine with notes rather than coins or tickets, as seen in Figure 3.6. Furthermore, approximately 55% of people do not cash out at the end of the session, which could be because they have lost all of their money and the machine credit has zeroed out or because they have a few cents or dollars left in the machine and do not consider cashing out.

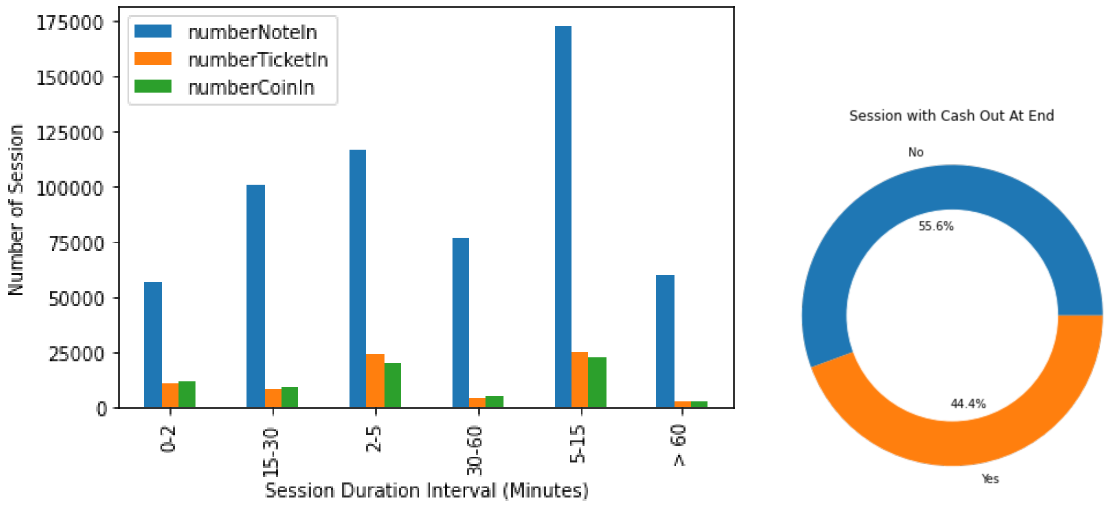


Figure 3.6: Distribution of Cash In and Cash Out

Figure 3.7 shows the correlation heatmap of all the features. This plot shows the pairwise linear relationship between features. There are many features having a high correlation with each other.

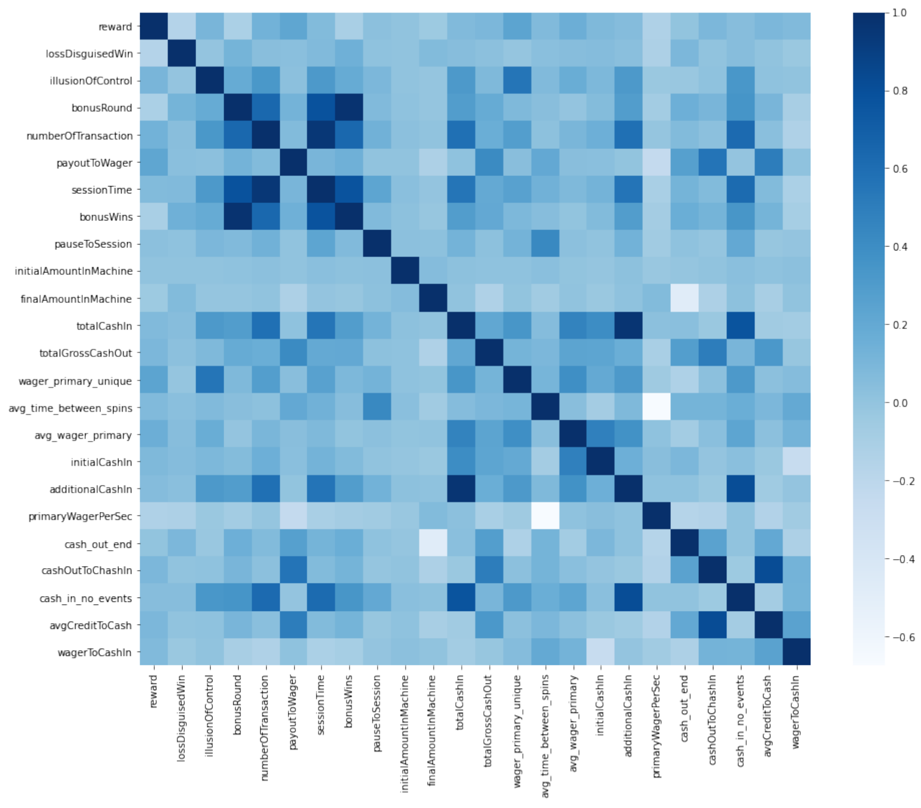


Figure 3.7: Correlation Heatmap

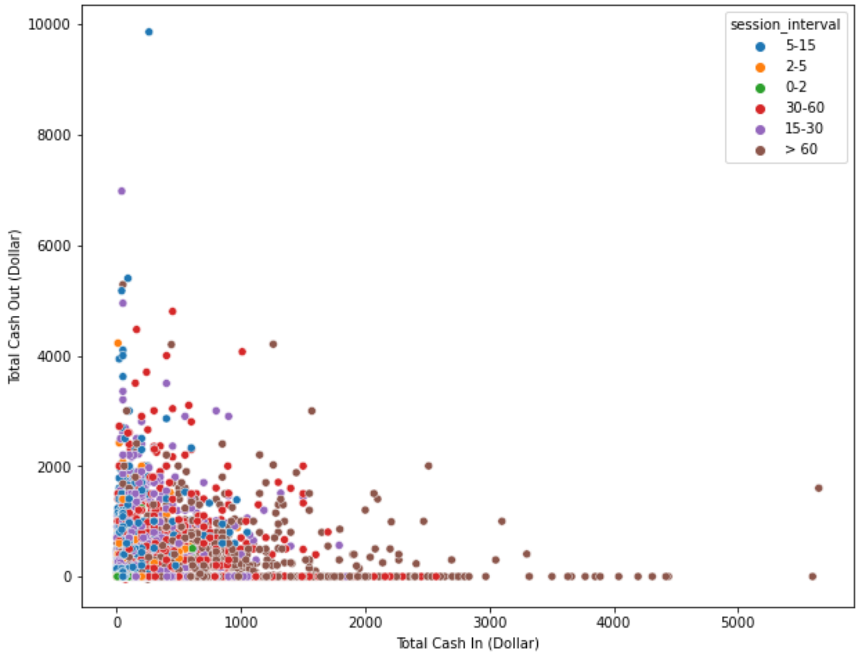


Figure 3.8: Total Cash In to Total Cash Out

We can see from the Figure 3.8 that players who have cashed in more than 1000 dollars have a lower chance of cashing out in that session. Furthermore, some people who played for over an hour and cashed in over 1000 dollars, lost all of their money in games or did not cash out at the end of the session.

The Cash Out to Cash In (COCI) ratio is an important element that tells us how much money the player took out of the machine at the end of the session in comparison to how much money was put into the machine. The Figure 3.9 illustrates that when the player plays for a longer period of time, the odds of receiving a larger return grow, but if the player plays for more than an hour, the COCI ratio declines.

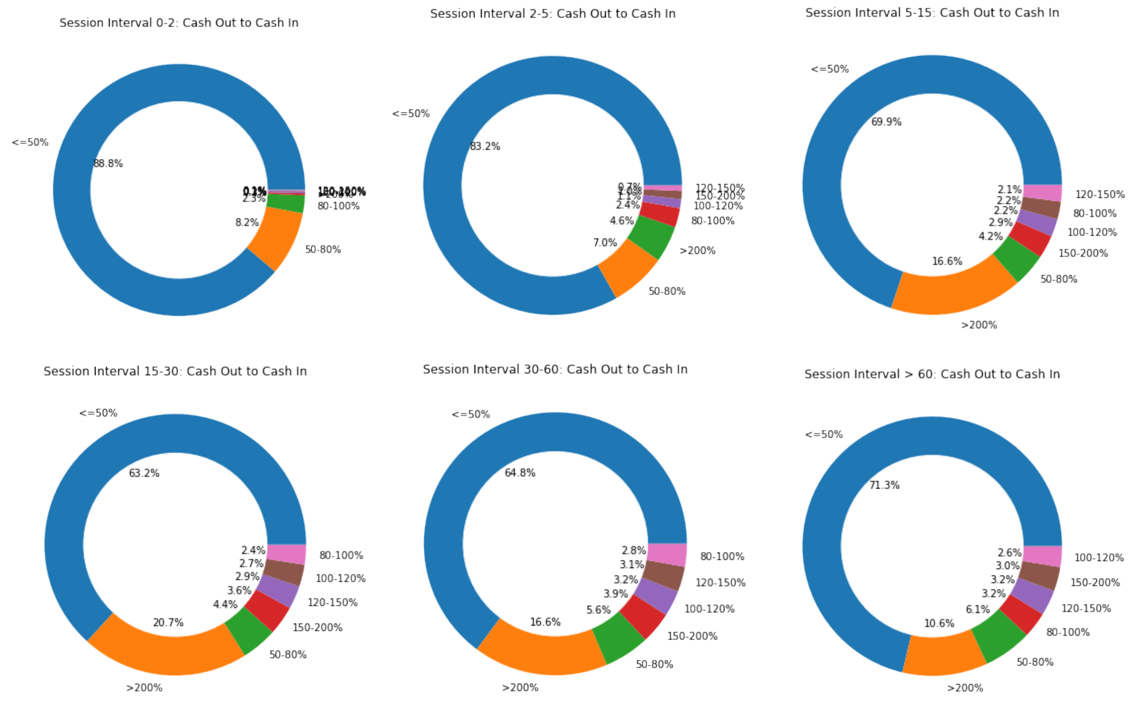


Figure 3.9: Cash Out to Cash In Ratio Distribution Over Session Interval

The Average Credit to Cash In (ACCI) ratio indicates how the player performs over time, as credit value may increase or decrease as the session progresses. According to Figure 3.10, players that fall into the middle session intervals of 5 to 30 minutes perform well overall.

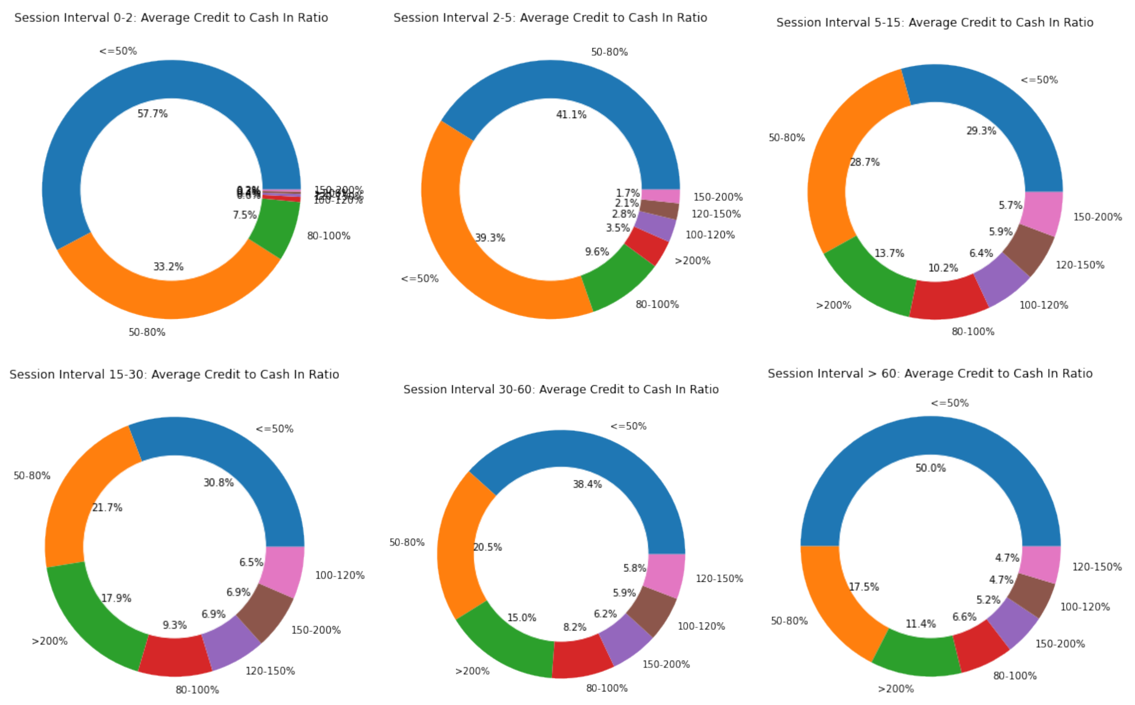


Figure 3.10: Average Credit to Cash In Ratio Distribution Over Session Interval

3.4 Labelling Sessions

EGM does not capture the player's experience of the session, that is, whether the player enjoyed it or not, while recording the logs. To identify each session as positive to indicate that the player enjoyed the session or negative to indicate that the player did not enjoy the session, we will make some assumptions, which we will discuss in the next section.

3.4.1 Segmenting Sessions

We consider two essential features in our assumptions to segment the sessions into positive and negative sessions: session length or duration and cash out to cash in (COCI) ratio, as these features can be considered indications of player enjoyment. We are considering these features in different ways in both the assumptions which are as follows.

Assumption 1: Time-based Only (TBO) Session Segmentation

In the assumption 1, all sessions with a session duration of more than 5 minutes would be labeled as positive, while all other sessions were labeled negative. This implies that the player who played for more than 5 minutes enjoyed the game, as the intuition behind this is that the person is playing for a longer period of time because they are enjoying the session. After applying the assumption, Figure 3.11 depicts the segmentation of sessions into positive and negative sessions. There are around 48.1% positive sessions and 51.9% negative sessions.

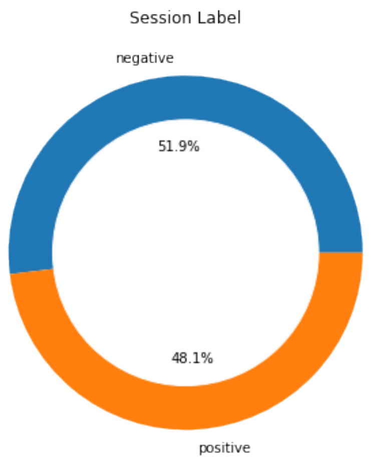


Figure 3.11: TBO sessions distribution

Assumption 2: Time-based and Cash Out to Cash In ratio (TB-COCI) Session Segmentation

Assumption 1 simply assumes that players who play for a longer period of time enjoy the session; however, this is not always the case, as some players tend to stop playing under 5 minutes upon receiving more money than they initially cashed in, which may be an enjoyable session for them. Along with session duration, we are considering the COCI ratio in this assumption. The assumption here is that sessions with a COCI ratio of more than 1, regardless of session duration, are labeled positive; additionally, sessions with a duration greater than 10 minutes are labeled positive; and finally, all other sessions are labeled negative. The COCI ratio ensures that the sessions with players earning more money than they invested are marked positive. Figure 3.12 illustrates the bifurcation of sessions into positive and negative sessions. There are about 33.8% sessions identified as positive, whereas negative sessions account for 66.2%.

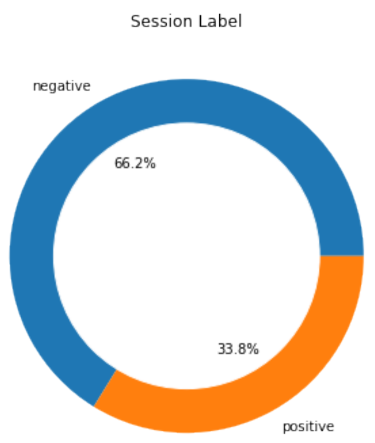


Figure 3.12: TB-COCI sessions distribution

Now that all the sessions are labelled, we can use these sessions in interpretable AI to determine what features are essential for classifying sessions.

3.5 Evaluation Metrics

There are numerous methods for assessing the quality of a classification model. Accuracy, F1-score, precision, recall, Receiver Operating Characteristic (ROC), or Area Under Curve (AUC) are all ways to quantify performance. These metrics can also be used to compare multiple models in order to determine which model performs the best. In this study, we will evaluate accuracy, precision, recall, and the F1-score to assess the performance of the models.

- **Accuracy:** It just evaluates how often the classifier predicts successfully. The accuracy of a prediction can be defined as the ratio of correct predictions to total predictions made. It is given by the equation:

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{3.3}$$

- **Precision:** It is a ratio of true positives (TP) among all positives predicted by the model that reflects how many of the accurately predicted occurrences were truly positive. When False Positives (FP) are more of a concern than False Negatives (FN), precision comes in handy. It is defined by the formula:

$$Precision = \frac{TP}{TP + FP} \tag{3.4}$$

- **Recall:** It is a ratio of true positives (TP) among all actual positives present in the dataset. It can be used to determine when a False Negative is more worrying than a False Positive. The equation to calculate recall is:

$$Recall = \frac{TP}{TP + FN} \tag{3.5}$$

- **F1-score:** It is the harmonic mean of precision and recall. The greater the F1 score, the better our model performs. The F1-score range is [0,1]. It is given by the equation:

$$F1 = 2 \times \frac{Precision \times Recall}{Precision + Recall} \tag{3.6}$$

3.6 Results and Discussion

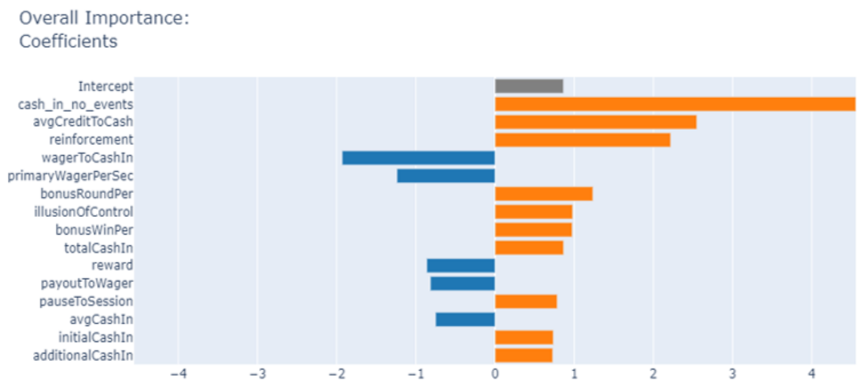
In this section, we will discuss the results and interpret machine learning algorithms on the assumptions discussed in section 3.4.1.

3.6.1 Interpreting Models

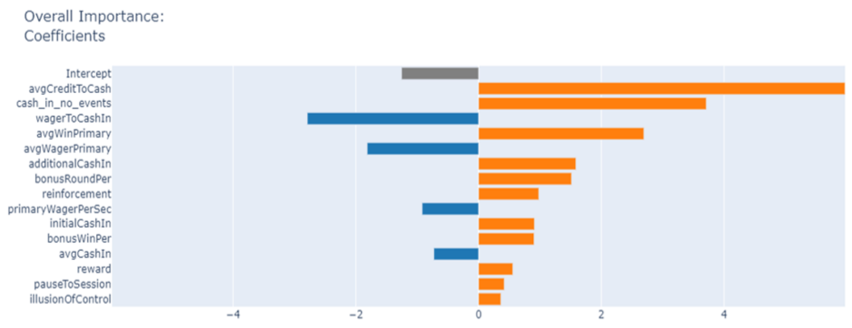
Due to the fact that we labeled sessions based on session duration and COCI ratio, features that had a high correlation with these features were not employed while training the models to prevent bias.

Logistic Regression

Logistic Regression (LR) is the simplest algorithm used to predict the probability of the session being positive or negative. The algorithm was implemented with the L2 regularization [63]. It gives high importance to the features that are impacting more on the outcome of the model. In TBO (Figure 3.13a), the number of times a player cashed in has the biggest positive coefficient, indicating that when players are having a good time, they put more money into the machine. The ACCI ratio, which reflects how well the player is performing during the session, has a positive effect on the player's mood as well. Players who wager more than they cash in may have a terrible overall experience because they may lose the money they may have won. In TB-COCI with the different assumption (Figure 3.13b), the ACCI ratio has been given more importance than the number of times player cash in, as it considers that overall performance defines enjoyment of the player rather than cash in numbers.



(a) TBO

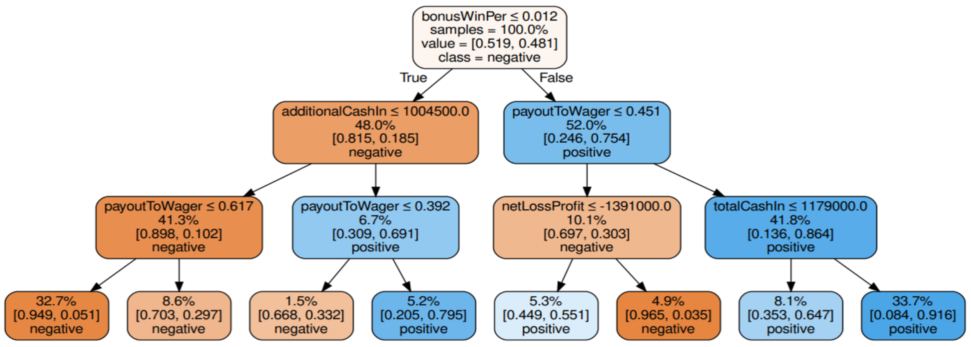


(b) TB-COCI

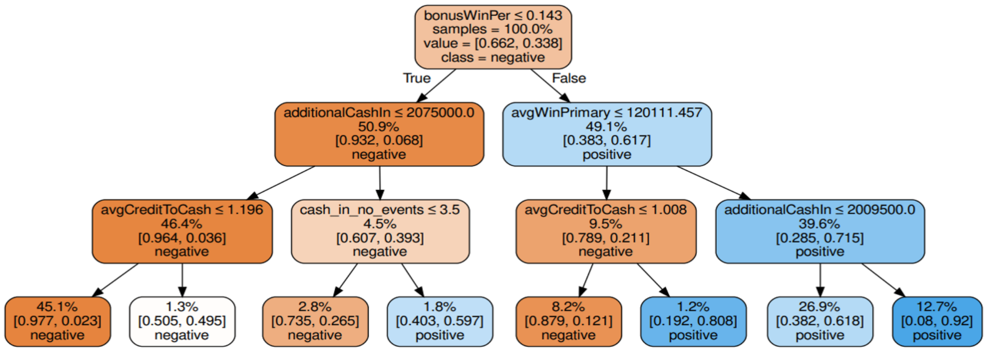
Figure 3.13: Feature Importance given by Logistic Regression

Decision Tree

Though LR gives the importance of the features to define the enjoyment of a player in a session, but it does not give more details about how the feature was used. Decision Tree (DT) infer simple decision rules from the features which makes them very easy to interpret. We trained DT with the max-dept of 3 to keep it relatively simple to interpret and we used gini as the criterion to measure the quality of the split. As shown in Figure 3.14a of TBO, DT uses bonus win percentage as the first feature to decide if the session is on the positive side or the negative. If the bonus win percentage is less than 1.2, the additional number of times the player cashed in is checked, else, the Payout to Wager (PW) ratio is checked to see if the player has won enough money that they have waged. Overall, if the player is winning more bonus rounds and getting more payout than they have waged in the other rounds that shows that the player is enjoying, while if the bonus win percentage is less and the player is not adding more cash in the machine shows that the player was not enjoying the session. The second assumption, TB-COCI, Figure 3.14b, also uses bonus win percentage as the initial feature for split, though the percentage value is 14.3 in the decision rule. DTs are very easy to follow to find out if the session will fall under the positive or negative category.



(a) TBO

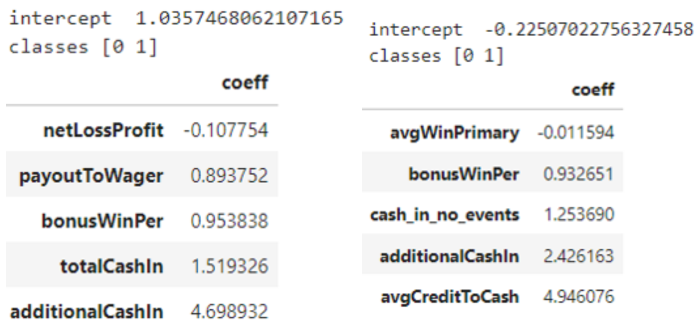


(b) TB-COCI

Figure 3.14: Decision Tree

Using Decision Tree Features in Logistic Regression

DT gives the flow of the feature used and how they are defined in the rule, but it does not give how much these features impact the decision of the model. To get the advantages of both DT and LR, we passed the feature used in the DT to the LR to get the weightage of each feature in predicting the enjoyability of a player.



```
intercept  1.0357468062107165
classes [0 1]
```

	coeff
netLossProfit	-0.107754
payoutToWager	0.893752
bonusWinPer	0.953838
totalCashIn	1.519326
additionalCashIn	4.698932

(a) TBO

```
intercept  -0.22507022756327458
classes [0 1]
```

	coeff
avgWinPrimary	-0.011594
bonusWinPer	0.932651
cash_in_no_events	1.253690
additionalCashIn	2.426163
avgCreditToCash	4.946076

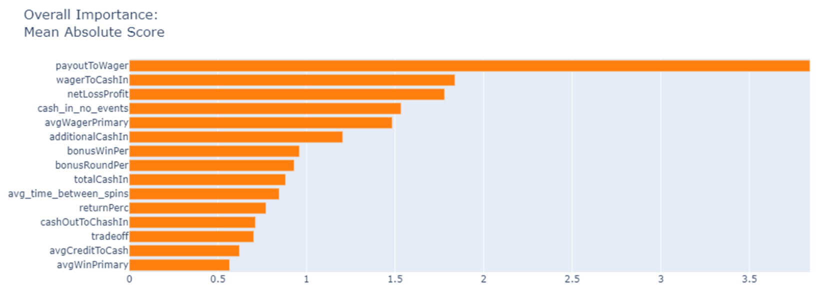
(b) TB-COCI

Figure 3.15: Coefficients of Features after applying Logistic Regression on Decision Tree Features

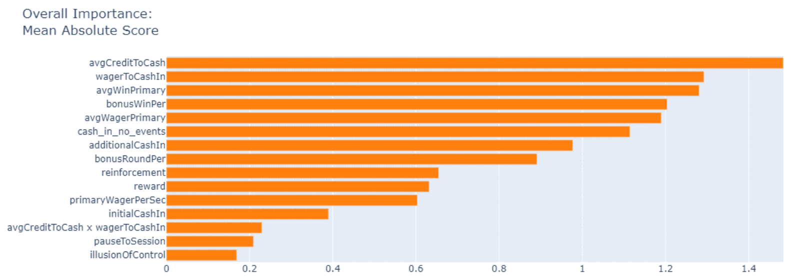
For TBO, the 5 features used by the decision tree were used to train logistic regression. Additional cashed in has the highest coefficient of around 4.70, followed by total cashed in with 1.52. This shows that if a player adds more cash to the machine, then that player is enjoying the game more. While on the other hand, in TB-COCI, the ACCI ratio is having high coefficient representing the overall performance of the player is affecting more to the mood of the player.

Explainable Boosting Machine

The previous algorithms are interpretable, but they do not perform as high as a black-box model. Explainable Boosting Machine (EBM) solves this problem by giving accuracy as state-of-art black-box models along with interpretability. We used interpretML API [1] to implement EBM, which produces nice visualization of each feature contribution towards the final prediction.



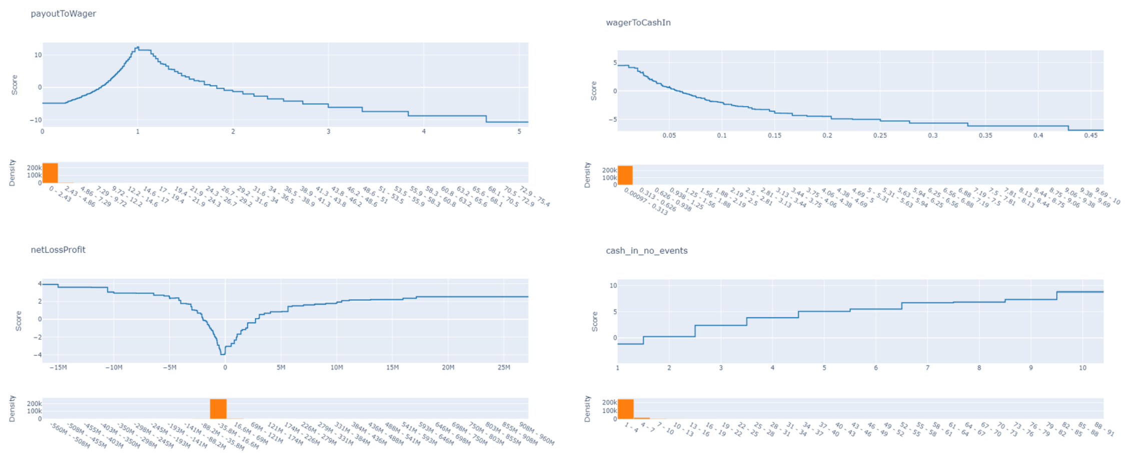
(a) TBO



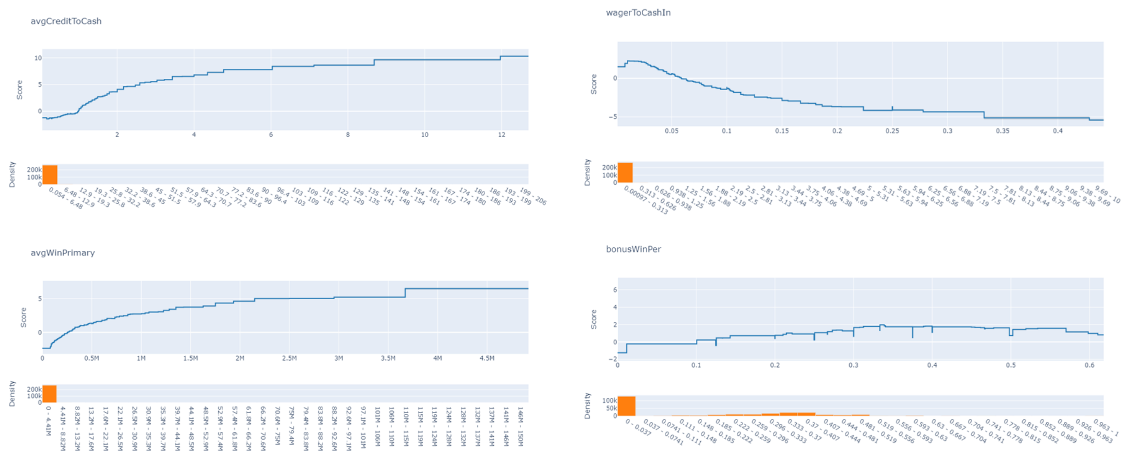
(b) TB-COCI

Figure 3.16: Feature Importance given by Explainable Boosting Machine

In TBO, lots of importance is given to the Payout to Wager (PW) ratio by the model, which is illustrated by Figure 3.16a. This means that a slight change in the PW ratio will highly affect the outcome of the model. The other features that were considered important by the model were the Wager to Cash In (WCI) ratio, net loss, number of times the player cashed in, and the average wager used in the primary game. Figure 3.17a shows how the final prediction of the model is affected by the top 4 most important features. We can notice a clear decrease and increase in the score with the increasing value of the WCI ratio and the numbers of cash in, respectively. There is a threshold value for the PW ratio and the net loss till which the score is increasing or decreases but after the threshold value, the score changes in the opposite direction than before.



(a) TBO



(b) TB-COCI

Figure 3.17: Top 4 Important Features

COCI ratio was identified as the most important feature by the EBM in TB-COCI for the prediction, as depicted in Figure 3.16b. The other features related to wagers and winning also contributed more to decide if the player was enjoying the session or not. How the change in the values of the top 4 important features is affecting the score of the model has been illustrated by Figure 3.17b. The ACCI ratio, average win percentage, and bonus win percentage show a positive correlation with the enjoyment of the player, while the increase in the WCI ratio is affecting the score negatively.

3.6.2 Model Comparison

To find out the best performing model and which assumption gave a better result, we compared the evaluation metrics. For evaluating how well the model predicted each session enjoyability and checking if the model was biased towards any class, we used precision, recall, and f1-score. For comparing the overall performance of the models, we used accuracy. We evaluated these models on test data, which was not used while training the models. Table 3.2 and Figure 3.18 show each model performance for both assumptions.

	TBO				TB-COCI			
	Precision	Recall	F1-score	Accuracy	Precision	Recall	F1-score	Accuracy
LR	0.89	0.89	0.89	0.89	0.92	0.91	0.92	0.93
DT	0.86	0.86	0.86	0.86	0.82	0.85	0.83	0.84
DT→LR	0.82	0.82	0.82	0.82	0.86	0.84	0.85	0.87
EBM	0.98	0.98	0.98	0.98	0.94	0.94	0.94	0.95

Table 3.2: Models Performance

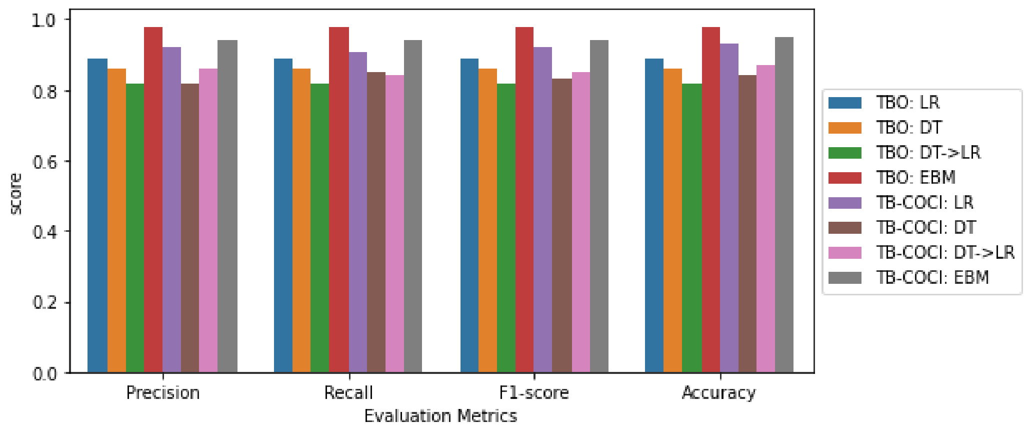


Figure 3.18: Models Performance

In both assumptions, EBM outperformed all other models, with an accuracy of 98% in TBO and 95% in TB-COCI. Even though LR was a simpler model than DT, it performed marginally better in both assumptions. After applying LR to the features used by the DT in the decision rules, the accuracy in TBO lowers by 4% to that of the DT, which was 86%, while the accuracy in TB-COCI increased by 3% to that of the DT. Based on the precision, recall, and F1-score, we can state that all of the models performed a good job of differentiating the enjoyability of the player's session, with

EBM showing the most potential for measuring player enjoyability solely on session data.

Chapter 4

Players Behavior Generation

In the previous chapter, we used interpretable AI to determine which features drive the enjoyment in the player experience, causing them to play for a longer period of time while playing on an EGM. Though the interpretable AI helped us to understand the reason behind the overall player behavior, but that's not enough to recreate the player behavior as the behavior of the player may depend on different situations.

In this chapter, we will discuss how we detected playstyles using the clustering technique. Furthermore, we will use one of the playstyle clustered data in our Reinforcement Learning (RL) model to mimic the behavior of players in that cluster. We will also discuss how we identified termination states where the players within this cluster end their session by deciding playing further is not worth it. Finally, the sessions generated by the RL are compared with the selected cluster session data. Figure 4.1 illustrated the flow of tasks performed for generating players behavior.

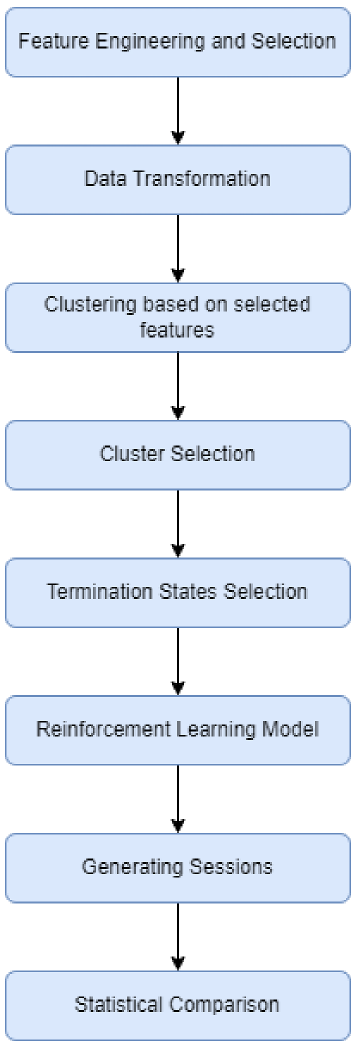

Feature Engineering and Selection
Data Transformation
Clustering based on selected features
Cluster Selection
Termination States Selection
Reinforcement Learning Model
Generating Sessions
Statistical Comparison

Figure 4.1: Player Behavior Generation Flowchart

4.1 Dataset Description

The data we will be using is from a less sophisticated EGM game compared to the game data which was used in the interpretable AI study in the previous chapter. This game does not have any bonus rounds or secondary game round, which makes the implementation of RL simpler. There are around 27,027 sessions in the dataset. Features were extracted from these session data as discussed in section 3.1.1.

4.2 Feature Selection and Data Transformation

For detecting the playstyles of the player, we have used only a few features similar to that were used by Mosquera and Keselj [49] in the clustering algorithms. Table 4.1 shows the selected features for clustering and their statistics measures. It is clear from Table 4.1 that the data is skewed to the right showing a non-normal sample distribution.

Features	Mean	Std. Dev.	Min	1st Qrt.	Median	3rd Qrt	Max
Win %	0.21	0.04	0.04	0.18	0.21	0.23	0.48
Loss Disguised as Win	0.10	0.03	0	0.08	0.10	0.11	0.44
Illusion of Control %	0.04	0.06	0	0.01	0.03	0.05	0.64
Number of Wagers	259.72	377.32	4	67	145	301	9945
Intensity	0.25	0.09	0.001	0.22	0.25	0.29	1.68

Table 4.1: Features Statistics Measures

To address the skewness of the data, there are several data transformation techniques such as:

- **Log Transformation:** It simply takes the logarithm of the values in the features, which is defined as:

$$y = \log{(y)}, \tag{4.1}$$

- **Square Root Transformation:** It takes the square root to transform the values in the features, which is represented as:

$$y = \sqrt{y}, \tag{4.2}$$

- **Box-Cox Transformation:** It transforms non-normal features to a normal distribution. The equation for transformation is defined as:

$$y(\lambda) = \begin{cases} \frac{y^{\lambda}-1}{\lambda}, & if\, \lambda \neq 0 \\ \log{(y)}, & if\, \lambda = 0 \end{cases} \tag{4.3}$$

  where λ is the transformation parameter.

We tried all the mentioned techniques to transform the feature, illustrated by Figure 4.2. With the Box-Cox transformation, the skewness coefficient value was closest to zero; as a result, it was chosen. The Q-Q plot analysis also shows the normal distribution of the data following the transformation. The z-score normalization technique was then used to normalize the transformed data.

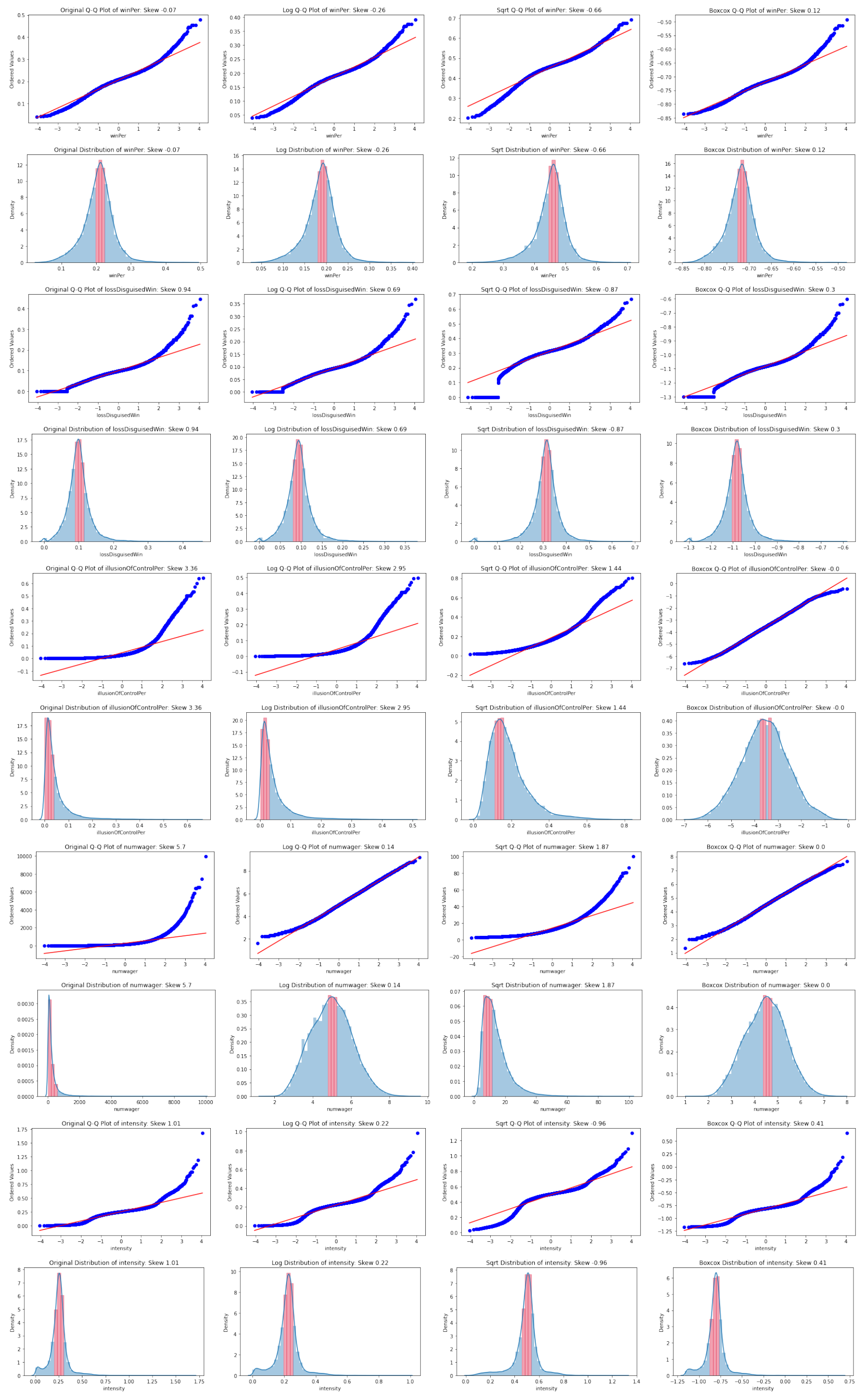


Figure 4.2: Data Transformation

4.3 Clustering

We used the K-means clustering method with random initialization to identify different types of gambling behavior or playstyles. Clustering algorithms, in general, divide data into k groups or clusters by analyzing cases in a data set; cases that appear similar to others are grouped together [69]. A dissimilarity function is used to define these clusters. There are numerous methods for clustering data, with k-means clustering being one of the most widely used.

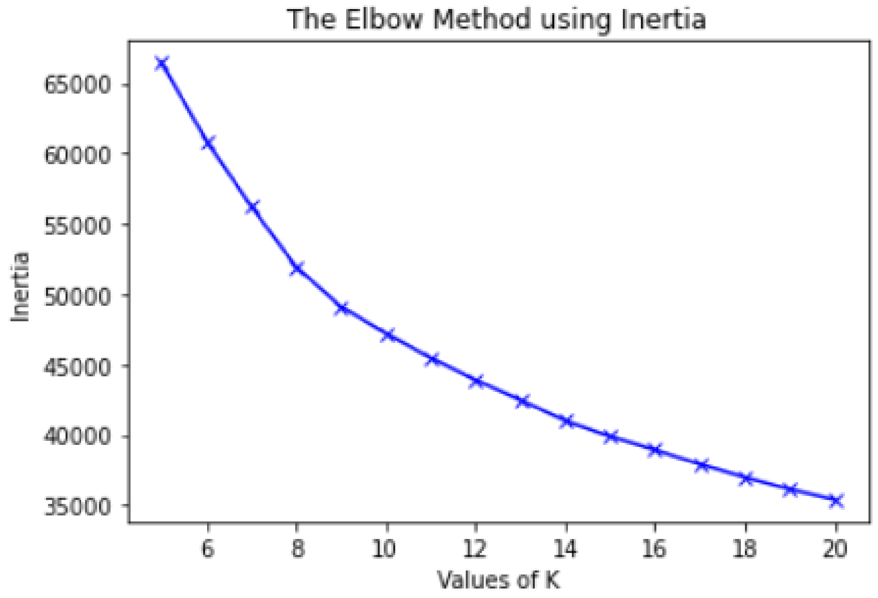


Figure 4.3: Elbow Method

The number of clusters is the only major hyper-parameter that needs to be tuned for this algorithm, which can be determined through Elbow method. This algorithm is linear in complexity and scales well to big data. The dataset was clustered by changing the value of k from 5 to 20 to identify a stable and suitable solution for k as shown in Figure 4.3. The optimal value of k was found to be 9 which was then verified using silhouette analysis (Figure 4.4). The dissimilarity of data objects was calculated in this study based on the distance between pairs of data objects using the Euclidean distance on the normalized dataset.

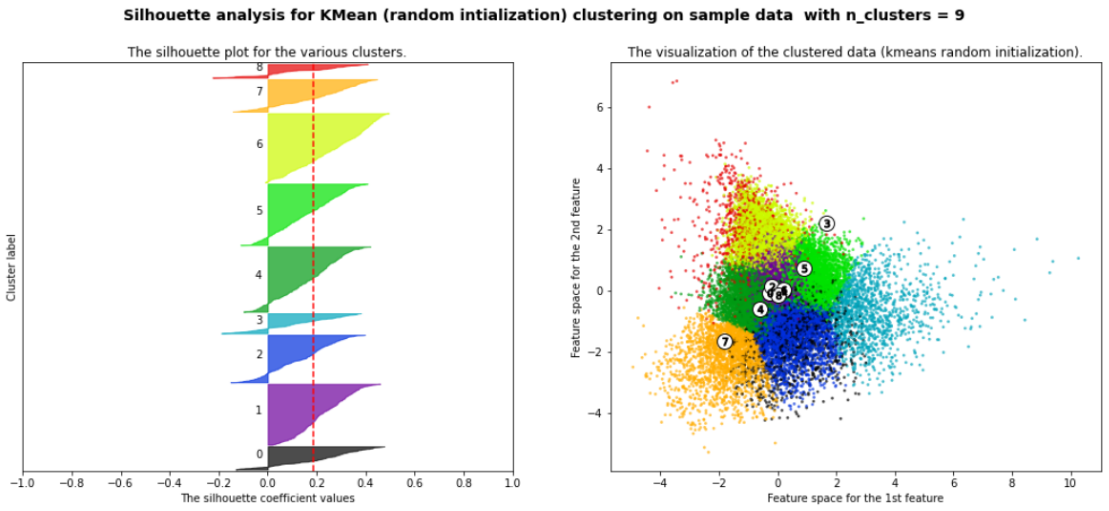


Figure 4.4: Silhouette Analysis

Figure 4.5 represents the overall distribution of sessions in each cluster. Clusters 1, 4, 5, and 6 have nearly identical numbers of player sessions, whereas other clusters have fewer sessions.

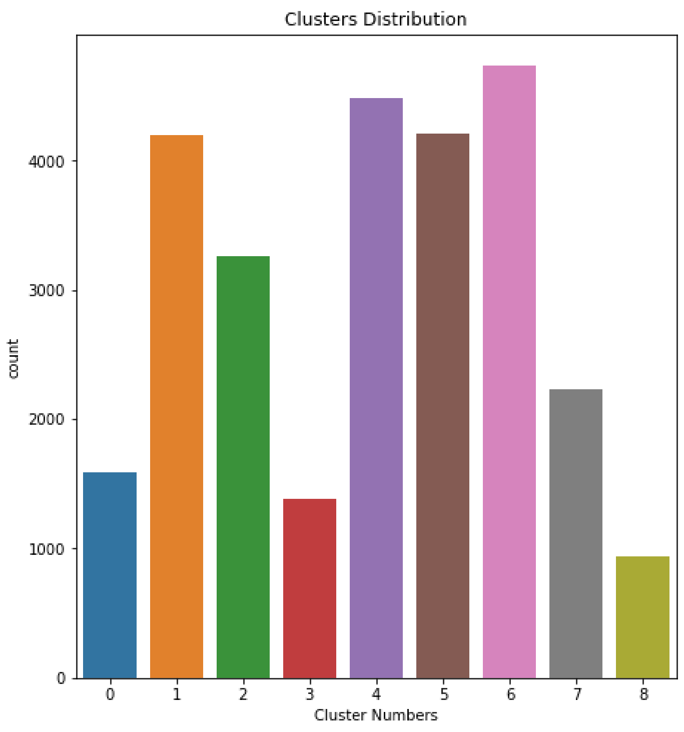


Figure 4.5: Distribution of Clusters

Figure 4.6 illustrates the distribution of all features in clusters. Many features have a low standard deviation (black bar on the plot), indicating that the values within the clusters are close to each other and the clusters are correctly formed.

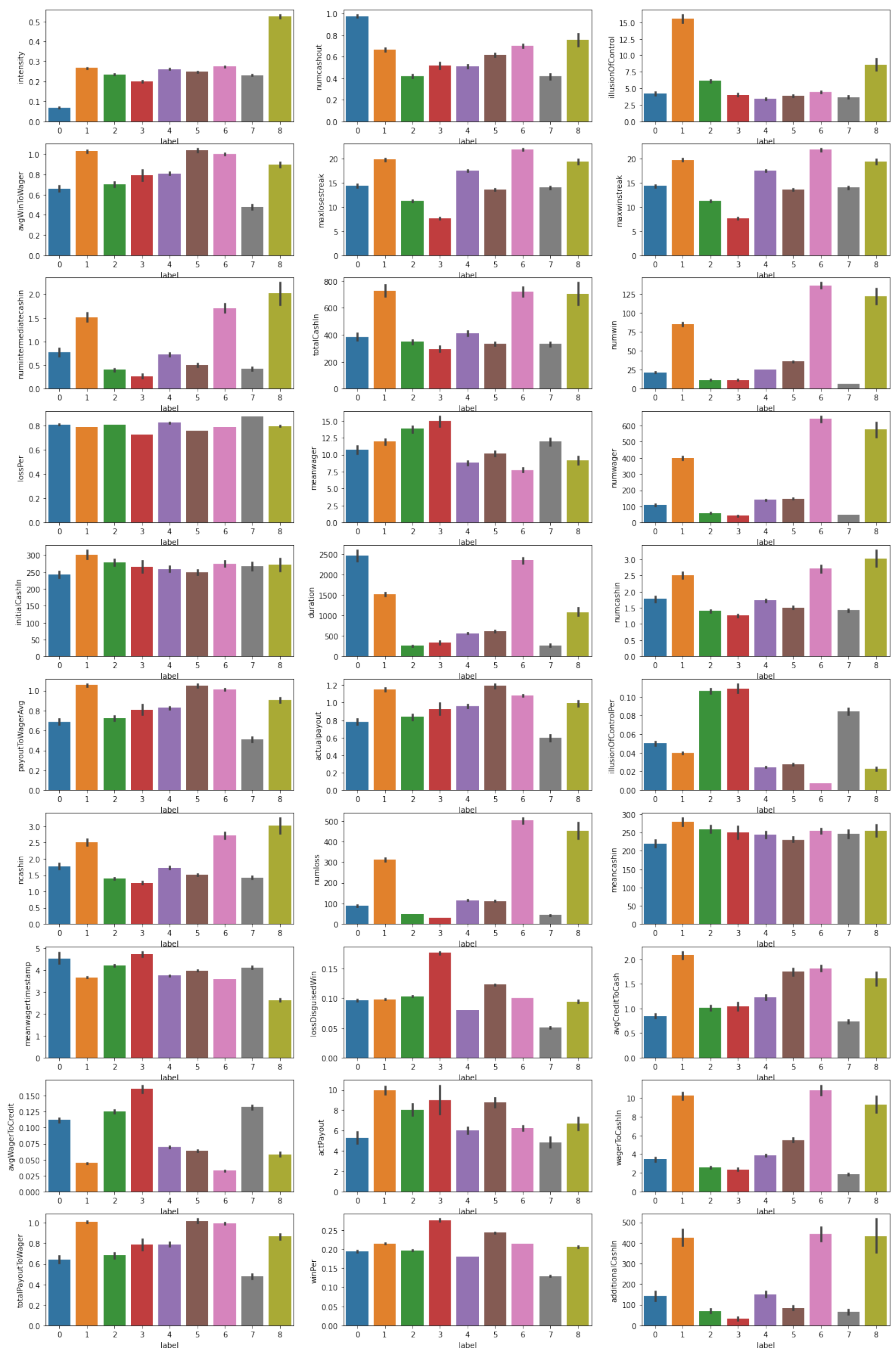


Figure 4.6: Distribution of Features

To mimic one of the playstyles, we selected a cluster from among all clusters that represent a group of players with similar playstyles. For this study, we chose cluster number 4, which contains 4483 sessions. This cluster represents intense gamblers, with a mean of 0.26 bets per second, who are unconcerned about losing a lot of money, as evidenced by an 82% loss.

4.4 Identifying Termination States

We want to find the termination states, where the player decides that continuing the session is not worth it, and train an RL model based on these states. The endpoints are the 25th, 50th, and 75th percentiles because they represent the majority of the distribution of the clusters and show the general playing style of the players within this cluster. The figures 4.7, 4.8, 4.9 show the distance between the value of a feature and its specific termination value that is 25th, 50th, and 75th percentiles, respectively. This distance is calculated by averaging 100 players' session features and then calculating the Euclidean distance for each transaction performed by the player with the termination value; i.e., the distance is calculated as the player progresses through the game. We chose win percentage, loss percentage, loss disguised as a win, PW ratio, and the illusion of control termination values in the termination states for training the RL model because they have a smooth decreasing curve, indicating a common playing style among players in this cluster.

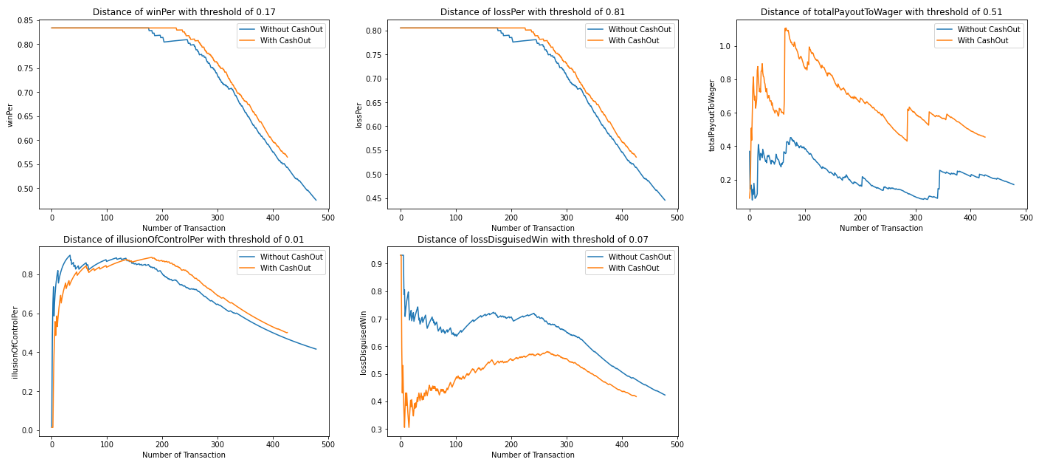


Figure 4.7: Distance with 25^{th} Percentile (Q1)

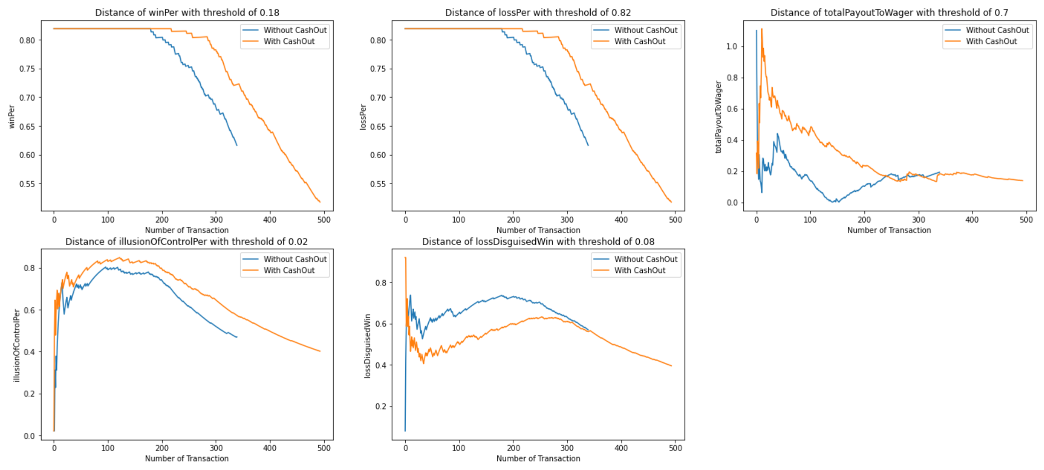


Figure 4.8: Distance with Median

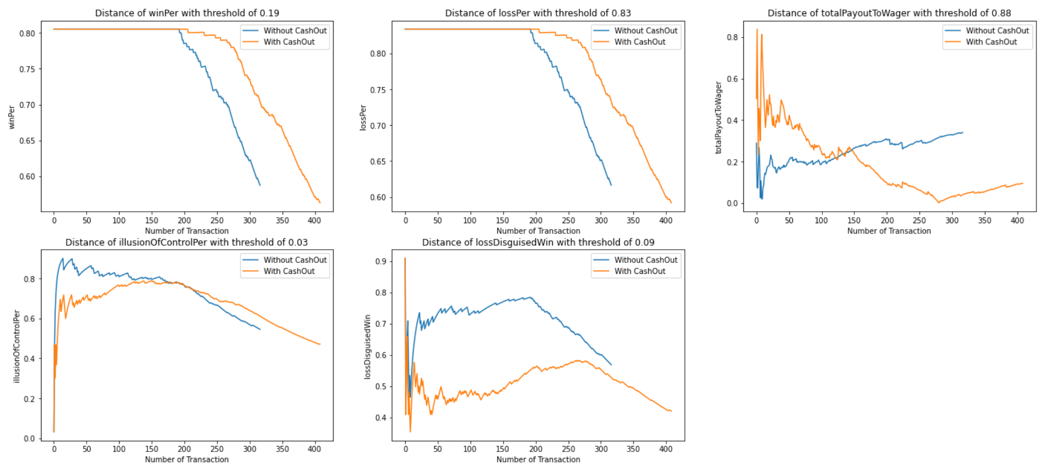


Figure 4.9: Distance with 75^{th} Percentile (Q3)

4.5 Reinforcement Learning

In this section, we will briefly discuss the action space, state space, and reward used in the RL model for generating player behavior sessions.

4.5.1 Action Space

The agent can mainly take two types of actions that are either it can make a wager or cash out. The agent also has to decide the amount of money to wager. As in the real EGM game, the agent can also only bet 2, 4, 8, 10, 15, 20, 25, 30, 35, 40, 45, 50, 60, 70, 80, 90, and 100 dollars. If the agent decides to cash out, it will cash out all of the money in the machine because the majority of the players in the original cluster only cashed out once.

4.5.2 State/Observation Space

The agent looks at the observation space in order to take action. Initially, the agent gets some random credit in the machine to start with as we have no cash in action. The agent considers the amount waged and received in the previous transaction, the current credit amount in the machine, the win percentage, the loss percentage, the loss disguised as a win, the payout to the wager, and the illusion of control. The agent uses these features to predict which actions will result in a higher reward and acts accordingly.

4.5.3 Reward

The agent's goal is to maximize the total reward of the game episode. If an agent makes an invalid move (for example, wagering more than credit) for each step, it will be penalized by 15. If the Euclidean distance between the termination state and the current state decreases, the agent receives a reward of 1. If the agent cashes out from the machine, he receives 15 as a reward. If the Euclidean distance between the current state and the termination state is less than 2, it receives a reward of 5.

4.6 Results and Discussion

The RL models were used to generate sessions simulating player behavior. We trained PPO2, which is implemented for GPU by OpenAI, and for multiprocessing, it uses vectorized environments compared to PPO1 which uses MPI [2], and ACKTR for around 1 million iterations.

We generated around 1000 sessions of agents playing the game by both models. These session data were then used to compute the win percentage, loss percentage, loss disguised as a win, and PW ratio (see description in Table 3.1). To evaluate the model's performance, we compared statistical measures like the minimum, 25th percentile, median, 75th percentile, and maximum values of the features and the real player cluster to see if the model's agent playstyle matched with the real player playstyle of the selected cluster.

Figure 4.10 shows that both ACKTR and PPO2 generated sessions that are very similar to those of real players. We can see that the 25th percentile, median, and 75th percentile values of both models are nearly identical to those of the original cluster for all the selected features. Figure 4.11 depicts the distribution comparison. We can see that the distribution of sessions generated by both models is similar to that of the original cluster. ACKTR depicted the wager features more accurately, such as the number of unique wagers, average wager, and the illusion of control, because the agent learned to change the wager, whereas the PPO2 agent played the entire session with only one wager value. ACKTR agent was intelligently changing the wager value depending on the losses and the wins to gain maximum reward out of the session. PPO2 was better than ACKTR in playing for a longer time, as PPO2 produced sessions with a number of wagers of around 200, while ACKTR produce sessions with a maximum of 100 numbers of wagers.

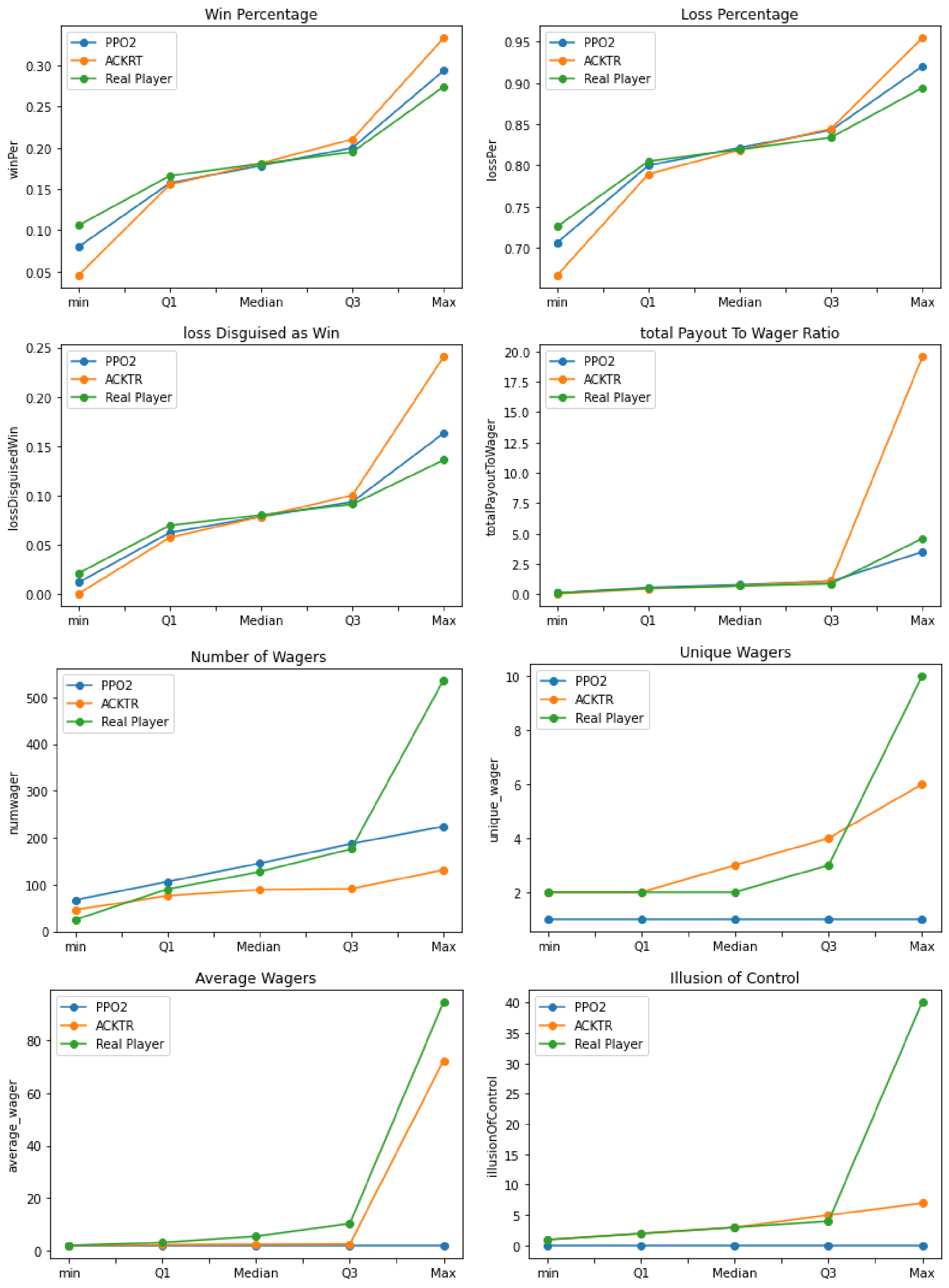


Figure 4.10: Statistical Measures Comparison

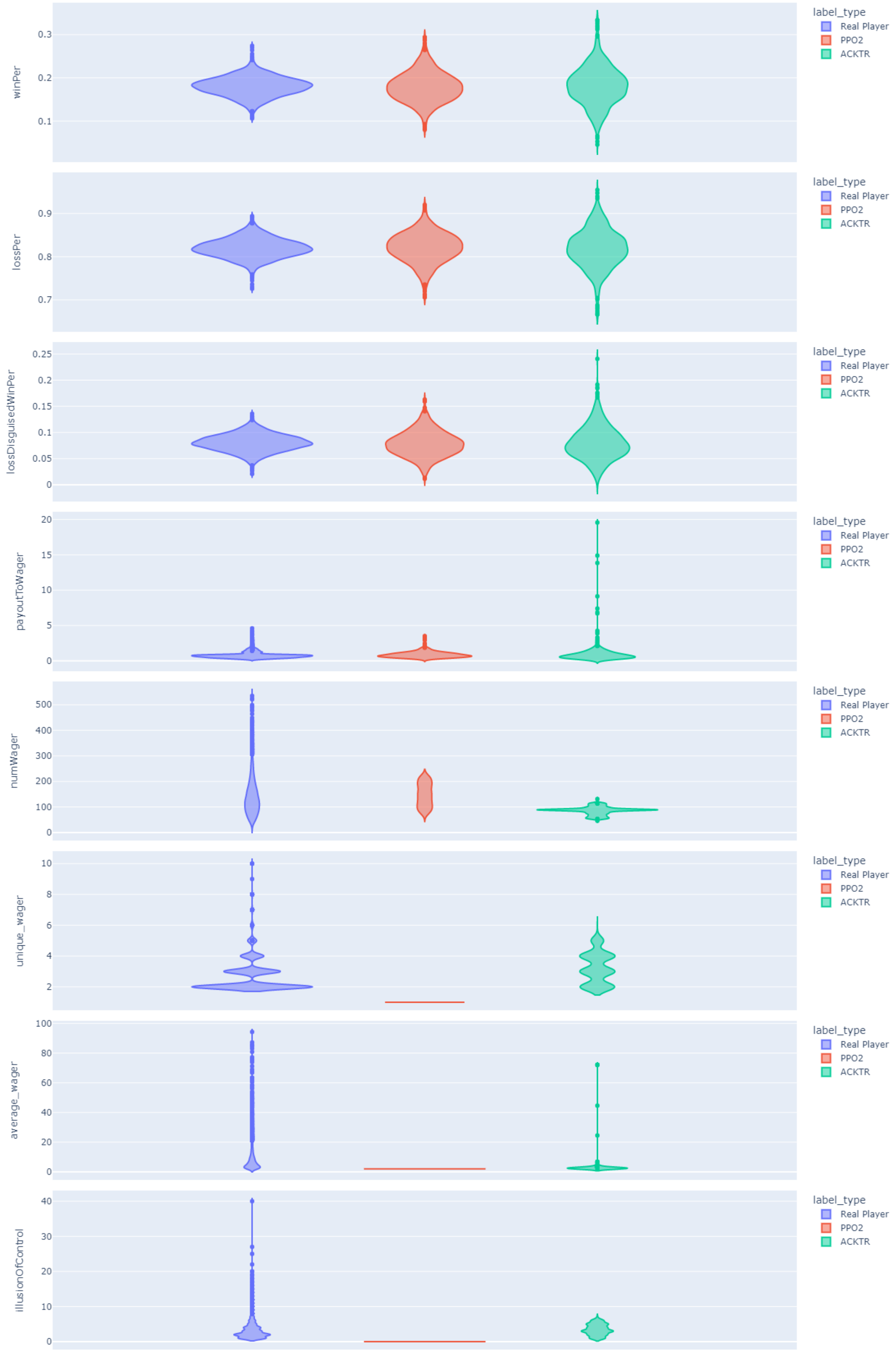


Figure 4.11: Features Distribution Comparison

Chapter 5

Conclusion and Future Work

In this chapter, we will conclude this research by discussing the main research outcomes in relation to the research objectives and questions, as well as discussing their significance and contribution. We will also briefly discuss the shortcomings of this research and provide suggestions for future work.

5.1 Addressing Research Problems

The following research problems were addressed by the analysis of session data and experiments implemented in this thesis:

1. What reasonable assumption should be considered while attempting to define the player experience?

   Given the characteristics of the players, two assumptions were used to categorize sessions with the players' experience: Time-based Only (TBO) and Time-based and Cash Out to Cash In ratio (TB-COCI).

2. Are there any interpretable models that can justify the assumption taken in the previous question while also accurately classifying the sessions?

   The Logistic Regression (LR), Decision Tree (DT), and Explainable Boosting Machine (EBM) provided insights into factors responsible for the positive experience and justified the TB-COCI assumption based on the importance of the features toward prediction. EBM performed efficiently in classifying the session experience.

3. What strategy may be utilized to comprehend the EGM's complicated environment and mimic the behavior of a real-world player?

   Reinforcement Learning (RL), Proximal Policy Optimization (PPO) and Actor Critic using Kronecker-Factored Trust Region (ACKTR), were used to train an agent to learn a simplified version of the EGM environment and generate playing behavior that mimics that of real-world players.

5.2 Importance of Factors in Player Positive Experience

5.2.1 Conclusion

This study aimed to justify an assumption for labelling whether or not a player enjoyed during the session and identifying the factors responsible for a positive experience while playing on EGM. Upon interpreting the models, it was found that assumption 2, which was TB-COCI, justifies the positive experience of a player as the importance given to the features by the models on TB-COCI was more cogent than TBO. The player experience is primarily influenced by factors such as bonus win rate, average wins in the primary game, additional cash in done by the player, and ACCI ratio. Furthermore, the DT and EBM performed better at classifying sessions in TBO than in TB-COCI, while the other two models performed better in TB-COCI. When LR was applied to the features chosen by DT, the overall performance in classifying the session improved, and it not only provides insights into the flow of the features, but also the weightage of each feature towards the predictions. The results indicate in both the assumptions, TBO and TB-COCI, EBM was giving a performance to that of state-of-art models along with a high interpretation of each feature contributing to the final prediction.

5.2.2 Limitations

The features chosen for the assumptions taken into account may not necessarily reflect actual enjoyment since it looks at behavioral patterns. Furthermore, the player experience may be influenced by external elements that are not taken into consideration in this study, such as the ambiance of the venues where the players are playing. Furthermore, the correlations between the characteristics and the player experience may be underdetermined by unforeseeable factors such as the player's motivations, like whether they play for money or for fun.

5.2.3 Future Work

For future research, labelling sessions with the experience that players had while playing the game could be done by collecting surveys at the end of the sessions, though this may be challenging because individuals desire privacy at places where EGMs are installed. Furthermore, in the assumption, a different set of behavioral attributes could be employed to label and classify sessions. Additionally, a linear function might be created utilizing the model insights to predict the real-time enjoyability of the players. Moreover, Neural Additive Models (NAM) [4] could be used which incorporate some of the expressivity of deep neural networks with the inherent intelligibility of generalized additive models. Lastly, this study could be used on the different kinds of EGM games, and it will be interesting to understand how the same player's experience changes with different games.

5.3 Mimicking Player Behavior

5.3.1 Conclusion

This study further mimics player behavior to obtain a good evaluation of a reinforcement learning model as a feasible substitute for real-world players. K-means gave an excellent separation of the player behaviors. We selected the cluster with intense gamblers and their behavioral attributes used to define the termination states for the RL model. The reward for the agents was calculated based on the Euclidean distance between the current state and the termination states. Both the models, PPO2 and ACKTR, succeeded in producing behavior similar to that of the selected cluster behavior. Though PPO2 was able to produce sessions with longer duration or more number of wagers, ACKTR slightly outperformed PPO2 as the ACKTR agent was intelligently able to change wager values during the session, which is an important attribute to say that the agent mimicked the player behavior.

5.3.2 Limitations

The RL algorithms employed to mimic player behavior in this study were only used for one playstyle, hence they may not be generalizable to other playstyles. Since the RL models have not been tested in production, they may be subject to unforeseen limitations.

5.3.3 Future Work

For future work, a different clustering algorithm could be used to have a better distinction of the behaviors of the player. We can include more features in the termination states, this might make the agent behave more like real-player as it will have more behavioral attributes to think about. It will be interesting to see how the agent will behave by tweaking the reward function from using the Euclidean distance to some other distance. Currently, the model does not have a cash in action as it starts with some random number credit in the machine, this action could be added so that it is performed by the agent, and also the agent could be modeled so it can perform some intermediate cash in and cash out based on the net loss of the session. Finally, it will be interesting to change this EGM environment to simulate it as a video game such as Atari games.

Bibliography

[1] Explainable boosting machine. `https://interpret.ml/docs/ebm.html`. Accessed: 2022-11-03.

[2] Ppo2. `https://stable-baselines.readthedocs.io/en/master/modules/ppo2.html`. Accessed: 2022-11-06.

[3] Nicola Adami, Sergio Benini, Alberto Boschetti, Luca Canini, Florinda Maione, and Matteo Temporin. Markers of unsustainable gambling for early detection of at-risk online gamblers. *International Gambling Studies*, 13(2):188–204, 2013.

[4] Rishabh Agarwal, Nicholas Frosst, Xuezhou Zhang, Rich Caruana, and Geoffrey E Hinton. Neural additive models: Interpretable machine learning with neural nets. *arXiv preprint arXiv:2004.13912*, 2020.

[5] Saurabh Arora and Prashant Doshi. A survey of inverse reinforcement learning: Challenges, methods and progress. *Artificial Intelligence*, 297:103500, 2021.

[6] David Arthur and Sergei Vassilvitskii. k-means++: the advantages of careful seeding. In *ACM-SIAM Symposium on Discrete Algorithms*, 2007.

[7] Kai Arulkumaran, Marc Peter Deisenroth, Miles Brundage, and Anil Anthony Bharath. Deep reinforcement learning: A brief survey. *IEEE Signal Processing Magazine*, 34(6):26–38, nov 2017.

[8] Jimmy Ba, Roger Baker Grosse, and James Martens. Distributed second-order optimization using kronecker-factored approximations. In *ICLR*, 2017.

[9] Chris L. Baker, Rebecca Saxe, and Joshua B. Tenenbaum. Action understanding as inverse planning. *Cognition*, 113(3):329–349, 2009. Reinforcement learning and higher cognition.

[10] Gary Banks, Robert Fitzgerald, and Louise Sylvan. Gambling: Productivity commission inquiry report, 2010.

[11] Richard Bellman. A Markovian decision process. *Indiana Univ. Math. J.*, 6:679–684, 1957.

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

[13] Max Bramer. Using j-pruning to reduce overfitting in classification trees. In Max Bramer, Frans Coenen, and Alun Preece, editors, *Research and Development in Intelligent Systems XVIII*, pages 25–38, London, 2002. Springer London.

[14] Julia Braverman, Debi A LaPlante, Sarah E Nelson, and Howard J Shaffer. Using cross-game behavioral markers for early identification of high-risk internet gamblers. *Psychol Addict Behav*, 27(3):868–877, September 2013.

[15] Julia Braverman and Howard J. Shaffer. How do gamblers start gambling: identifying behavioural markers for high-risk internet gambling. *European Journal of Public Health*, 22(2):273–278, 01 2010.

[16] Yunfei Chu, Zhinong Wei, Guoqiang Sun, Haixiang Zang, Sheng Chen, and Yizhou Zhou. Optimal home energy management strategy: A reinforcement learning method with actor-critic using Kronecker-factored trust region. *Electric Power Systems Research*, 212:108617, 2022.

[17] Mark Craven and Jude Shavlik. Extracting tree-structured representations of trained networks. *Advances in neural information processing systems*, 8:24–30, 1995.

[18] Mengyao Cui et al. Introduction to the k-means clustering algorithm based on the elbow method. *Accounting, Auditing and Finance*, 1(1):5–8, 2020.

[19] Finale Doshi-Velez and Been Kim. Towards a rigorous science of interpretable machine learning. *arXiv preprint arXiv:1702.08608*, 2017.

[20] Simo Dragicevic, George Tsogas, and Aleksandar Kudic. Analysis of casino online gambling data in relation to behavioural risk markers for high-risk gambling and player protection. *International Gambling Studies*, 11(3):377–391, 2011.

[21] Anuška Ferligoj and Vladimir Batagelj. Some types of clustering with relational constraints. *Psychometrika*, 48(4):541–552, 1983.

[22] Harsh M. Gawai, Colin D. Conrad, and Vlado Keselj. It's more than memes: User risk appetite and app enjoyment predict simulated mobile trading app behavior. *CONF-IRM 2022 Proceedings*, 16, 2022. `https://aisel.aisnet.org/confirm2022/16`.

[23] Zoubin Ghahramani. Unsupervised learning. In *Advanced Lectures on Machine Learning: ML Summer Schools 2003*, pages 72–112, Berlin, Heidelberg, 2004. Springer Berlin Heidelberg.

[24] Roger Grosse and James Martens. A kronecker-factored approximate fisher matrix for convolution layers. In *International Conference on Machine Learning*, pages 573–582. PMLR, 2016.

[25] Khaoula Hantous, Lilia Rejeb, and Rahma Hellali. Detecting physiological needs using deep inverse reinforcement learning. *Applied Artificial Intelligence*, 36(1):2022340, 2022.

[26] Trevor Hastie and Robert Tibshirani. Generalized additive models. *Statistical Science*, 1(3):297–310, 1986.

[27] A. K. Jain, M. N. Murty, and P. J. Flynn. Data clustering: A review. *ACM Comput. Surv.*, 31(3):264–323, sep 1999.

[28] Anil K. Jain and Richard C. Dubes. *Algorithms for Clustering Data*. Prentice-Hall, Inc., USA, 1988.

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

[30] Beomjoon Kim and Joelle Pineau. Socially adaptive path planning in human environments using inverse reinforcement learning. *International Journal of Social Robotics*, 8:51–66, 2016.

[31] Trupti M Kodinariya and Prashant R Makwana. Review on determining number of cluster in k-means clustering. *International Journal*, 1(6):90–95, 2013.

[32] Henrik Kretzschmar, Markus Spies, Christoph Sprunk, and Wolfram Burgard. Socially compliant mobile robot navigation via inverse reinforcement learning. *The International Journal of Robotics Research*, 35(11):1289–1307, 2016.

[33] Markus Kuderer, Shilpa Gulati, and Wolfram Burgard. Learning driving styles for autonomous vehicles from demonstration. In *2015 IEEE International Conference on Robotics and Automation (ICRA)*, pages 2641–2646, 2015.

[34] Soheil Latifi. Electronic gaming machine playstyle detection and rapid playstyle classification using multivariate convolutional LSTM neural network architecture. Master's thesis, Dalhousie University, 2021.

[35] Sergey Levine, Chelsea Finn, Trevor Darrell, and Pieter Abbeel. End-to-end training of deep visuomotor policies. *The Journal of Machine Learning Research*, 17(1):1334–1373, 2016.

[36] Sergey Levine, Peter Pastor, Alex Krizhevsky, Julian Ibarz, and Deirdre Quillen. Learning hand-eye coordination for robotic grasping with deep learning and large-scale data collection. *The International journal of robotics research*, 37(4-5):421–436, 2018.

[37] Yuxi Li. Deep reinforcement learning: An overview. *CoRR*, 2017. `http://arxiv.org/abs/1701.07274`.

[38] Yuxi Li. Deep reinforcement learning, 2018. `https://arxiv.org/abs/1810.06339`.

[39] Charles Livingstone. *How electronic gambling machines work*. Australian Gambling Research Centre - AIFS, July 2017. Published online on 11/7/17 by AGRC/AIFS.

[40] Yin Lou, Rich Caruana, Johannes Gehrke, and Giles Hooker. Accurate intelligible models with pairwise interactions. In *Proceedings of the 19th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '13, page 623–631, New York, NY, USA, 2013. Association for Computing Machinery.

[41] Scott M Lundberg and Su-In Lee. A unified approach to interpreting model predictions. *Advances in neural information processing systems*, 30, 2017.

[42] Vance V. MacLaren, Johnathan A. Fugelsang, Kevin A. Harrigan, and Michael J. Dixon. The personality of pathological gamblers: A meta-analysis. *Clinical Psychology Review*, 31(6):1057–1067, 2011.

[43] A.A. Markov and N.M. Nagorny. The theory of algorithms. In *Mathematics and its Applications*. Springer Dordrecht, 1954.

[44] James Martens and Roger Grosse. Optimizing neural networks with kronecker-factored approximate curvature. In *International conference on machine learning*, pages 2408–2417. PMLR, 2015.

[45] Dhendra Marutho, Sunarna Hendra Handaka, Ekaprana Wijaya, and Muljono. The determination of cluster number at k-mean using elbow method and purity evaluation on headline news. In *2018 International Seminar on Application for Technology of Information and Communication*, pages 533–538, 2018.

[46] Glenn W. Milligan and Martha C. Cooper. Methodology review: Clustering methods. *Applied Psychological Measurement*, 11(4):329–354, 1987.

[47] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller. Playing atari with deep reinforcement learning, 2013.

[48] Christoph Molnar. *Interpretable Machine Learning*. 2 edition, 2022. `https://christophm.github.io/interpretable-ml-book`.

[49] Maria Gabriella Mosquera and Vlado Keselj. Identifying electronic gaming machine gambling personae through unsupervised session classification. *Big Data & Information Analytics*, 2(2):141–175, 2017.

[50] Vladimir Nasteski. An overview of the supervised machine learning methods. *Horizons*, 4:51–62, 2017.

[51] National Research Council (US) Committee on the Social and Economic Impact of Pathological Gambling. *Pathological Gambling: A Critical Review*. National Academies Press (US), Washington (DC), 1999.

[52] Gergely Neu and Csaba Szepesvári. Apprenticeship learning using inverse reinforcement learning and gradient methods. In *Proceedings of the Twenty-Third Conference on Uncertainty in Artificial Intelligence*, UAI'07, page 295–302, Arlington, Virginia, USA, 2007. AUAI Press.

[53] Andrew Y Ng, Stuart Russell, et al. Algorithms for inverse reinforcement learning. In *Icml*, volume 1, page 2, 2000.

[54] Wenjia Ni. Application of clustering, logistic regression and decision tree induction on egm data for detection and prediction of at-risk and problem gamblers. Master's thesis, Dalhousie University, 2014.

[55] Harsha Nori, Samuel Jenkins, Paul Koch, and Rich Caruana. Interpretml: A unified framework for machine learning interpretability. *arXiv preprint arXiv:1909.09223*, 2019.

[56] OpenAI, :, Christopher Berner, Greg Brockman, Brooke Chan, Vicki Cheung, Przemyslaw Debiak, Christy Dennison, David Farhi, Quirin Fischer, Shariq Hashme, Chris Hesse, Rafal Józefowicz, Scott Gray, Catherine Olsson, Jakub Pachocki, Michael Petrov, Henrique P. d. O. Pinto, Jonathan Raiman, Tim Salimans, Jeremy Schlatter, Jonas Schneider, Szymon Sidor, Ilya Sutskever, Jie Tang, Filip Wolski, and Susan Zhang. Dota 2 with large scale deep reinforcement learning, 2019.

[57] Christian Percy, Artur S. d'Avila Garcez, Simo Dragicevic, Manoel Vitor Macedo França, Gregory G. Slabaugh, and Tillman Weyde. The need for knowledge extraction: Understanding harmful gambling behavior with neural networks. In *ECAI*, 2016.

[58] Jan Peters and Stefan Schaal. Reinforcement learning of motor skills with policy gradients. *Neural Networks*, 21(4):682–697, 2008. Robotics and Neuroscience.

[59] D T Pham, S S Dimov, and C D Nguyen. Selection of k in k-means clustering. *Proceedings of the Institution of Mechanical Engineers, Part C: Journal of Mechanical Engineering Science*, 219(1):103–119, 2005.

[60] Pradeep Rai and Shubha Singh. A survey of clustering techniques. *International Journal of Computer Applications*, 7(12):1–5, 2010.

[61] Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. "Why should i trust you?" Explaining the predictions of any classifier. In *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining*, pages 1135–1144, 2016.

[62] Michelle Rotermann and Heather Gilmour. Who gambles and who experiences gambling problems in canada, Aug 2022.

[63] Fariborz Salehi, Ehsan Abbasi, and Babak Hassibi. The impact of regularization on high-dimensional logistic regression. *Advances in Neural Information Processing Systems*, 32, 2019.

[64] Sanjoy Sarkar, Tillman Weyde, Artur S. d'Avila Garcez, Gregory G. Slabaugh, Simo Dragicevic, and Christian Percy. Accuracy and interpretability trade-offs in machine learning applied to safer gambling. In *CoCo@NIPS*, 2016.

[65] Tony Schellinck and T. Schrans. Intelligent design: How to model gambler risk assessment by using loyalty tracking data. *Journal of Gambling Issues*, 26:51–68, 01 2011.

[66] John Schulman, Sergey Levine, Pieter Abbeel, Michael Jordan, and Philipp Moritz. Trust region policy optimization. In *International conference on machine learning*, pages 1889–1897. PMLR, 2015.

[67] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint*, 2017.

[68] Wolfram Schultz, Peter Dayan, and P. Read Montague. A neural substrate of prediction and reward. *Science*, 275:1593 – 1599, 1997.

[69] Howard J. Shaffer and David A. Korn. Gambling and related mental disorders: A public health analysis. *Annual Review of Public Health*, 23(1):171–212, 2002. PMID: 11910060.

[70] Jeffrey M. Stanton. Galton, Pearson, and the Peas: A brief history of linear regression for statistics instructors. *Journal of Statistics Education*, 9(3), 2001.

[71] Richard S. Sutton and Andrew G. Barto. *Reinforcement Learning: An Introduction.* A Bradford Book, Cambridge, MA, USA, 2018.

[72] Tomer D. Ullman, Chris L. Baker, Owen Macindoe, Owain Evans, Noah D. Goodman, and Joshua B. Tenenbaum. Help or hinder: Bayesian models of social goal inference. In *Proceedings of the 22nd International Conference on Neural Information Processing Systems*, NIPS'09, page 1874–1882, Red Hook, NY, USA, 2009. Curran Associates Inc.

[73] Hado van Hasselt, Arthur Guez, and David Silver. Deep reinforcement learning with double q-learning. *Proceedings of the AAAI Conference on Artificial Intelligence*, 30(1), Mar. 2016.

[74] Oriol Vinyals, Igor Babuschkin, Wojciech M Czarnecki, Michaël Mathieu, Andrew Dudzik, Junyoung Chung, David H Choi, Richard Powell, Timo Ewalds, Petko Georgiev, et al. Grandmaster level in starcraft ii using multi-agent reinforcement learning. *Nature*, 575(7782):350–354, 2019.

[75] Yuhui Wang, Hao He, and Xiaoyang Tan. Truly proximal policy optimization. In Ryan P. Adams and Vibhav Gogate, editors, *Proceedings of The 35th Uncertainty in Artificial Intelligence Conference*, volume 115 of *Proceedings of Machine Learning Research*, pages 113–122. PMLR, 22–25 Jul 2020.

[76] Yihao Wu and Jun Izawa. The regret motivated reinforcement learning. In *2021 International Symposium on Micro-NanoMehatronics and Human Science (MHS)*, page 1–5. IEEE Press, 2021.

[77] Yuhuai Wu, Elman Mansimov, Roger B Grosse, Shun Liao, and Jimmy Ba. Scalable trust-region method for deep reinforcement learning using kronecker-factored approximation. *Advances in neural information processing systems*, 30, 2017.

[78] Rui Xu and D. Wunsch. Survey of clustering algorithms. *IEEE Transactions on Neural Networks*, 16(3):645–678, 2005.

[79] Changxi You, Jianbo Lu, Dimitar Filev, and Panagiotis Tsiotras. Advanced planning for autonomous vehicles using reinforcement learning and deep inverse reinforcement learning. *Robotics and Autonomous Systems*, 114:1–18, 2019.