Nanotechnology Perceptions
ISSN 1660-6795
www.nano-ntp.com

Enhancing Player Engagement and Retention: Bonus Optimization and Churn Prediction in iGaming

Prathibha P.H[1], Shivada K.P[2]

[1]*Sree Sankara Vidyapeetom college, Valayanchirangara, Kerala, India*
[2]*Amirta Vishwapeetham, India*

The Bonus Optimization and and Churn Prediction works utilize advanced machine learning techniques, data analytics, and optimization algorithms to enhance player engagement, satisfaction, and retention in the iGaming industry. The Bonus Optimization initiative personalizes bonus distribution strategies by analyzing player behavior and responsiveness, leading to improved player retention and optimized marketing expenditures. In parallel, the Churn Prediction work focuses on identifying at-risk players through survival analysis models, enabling proactive interventions. The integration of predictive modeling and interactive tools facilitates data-driven decision-making, maximizing player lifetime value. Together, these research work represent a significant advancement in understanding and influencing player behavior in the iGaming sector.

Keywords: Bonus Optimization, Churn Prediction, Player Retention, Predictive Modeling, Streamlit Interface, iGaming Industry, Player Behavior.

1. Introduction

This work aimed to optimize bonus distribution using reinforcement learning models which suggest how the bonus is supposed to be distributed for each kind of customer and a churn model which is a statistical model that helps in predicting the churn of the customer using a survival model. The work starts by identifying the main reasons for inefficient bonus distribution to customers and patterns in customers that would in turn cause them to churn. Using machine learning and statistical analysis expertise, the data was comprehensively analyzed using Python's Polars and Pandas modules, the data analysis was done using python's polar and pandas modules apart from which visualization tools like Tableau and powerBI were also used, by using these tools the data was analyzed and underlying causes to the main problems were identified which would help reduce customer churn by identifying patterns and also optimize reinvestment by understanding bonus distributions among customers.

2. BONUS OPTIMIZATION USING REINFORCEMENT LEARNING

The objective of bonus optimization in the i-gaming industry is to strategically manage and allocate bonuses to maximize player engagement, retention, and revenue generation. Effective

bonus optimization involves identifying bonus abusers and determining the optimal bonus for each player, ensuring that the bonuses offered encourage positive player behavior and contribute to the company's profitability. In this work, two reinforcement learning models, Q-learning and SARSA, were implemented to achieve these objectives.

Bonus optimization is crucial for several reasons. First, it enhances player retention and decreases churn rates by incentivizing existing players to remain loyal and engaged with the platform. Second, it aids in player acquisition by offering attractive bonuses that serve as powerful marketing tools to attract new players. Third, it drives revenue generation by encouraging increased player spending and activity, which translates directly into higher earnings for the i-gaming company.

Two scenarios illustrate the impact of bonus optimization. In the first scenario, a player utilizes a bonus but does not achieve a positive return; nevertheless, their engagement with the bonus contributes to the company's revenue. In the second scenario, a player strategically utilizes a bonus and achieves a positive return, resulting in reduced revenue for the company due to the player's successful exploitation of the bonus. To address these scenarios and optimize bonus allocation, the work employs SARSA and Q-learning reinforcement learning algorithms, utilizing packages such as Gym, Stable Baselines3, and NumPy to build and train the models.

2.1 DATA RETRIEVAL

Data retrieval was streamlined by securely setting up AWS credentials using boto3 session and creating an S3 client for direct interaction with the data. The dataset was efficiently

accessed from the specified S3 bucket by leveraging AWS Wrangler, minimizing delays and ensuring smooth data pulling. This method facilitated effective and secure data retrieval and maintained the scalability required for handling large datasets.

The dataset used for this work comprised 2 million rows and 16 columns, providing a comprehensive foundation for analysis and model training. The preprocessing steps included handling missing values, selecting relevant features, and removing redundant or irrelevant columns. Handling missing values ensured the dataset was clean and reliable for analysis, while feature selection focused on identifying the most important variables for the reinforcement learning models. Removing redundant or irrelevant columns helped streamline the dataset, enhancing the performance and accuracy of the models.

2.2 DATA ANALYSIS AND VISUALIZATION USING TABLEAU

In the development of the reinforcement learning environment for bonus optimization, a multi-faceted approach was adopted to ensure alignment with business objectives and customer behavior patterns. First and foremost, the Bonus Redemption Rate was meticulously analyzed to gauge the effectiveness of bonus distribution. This metric, calculated as the ratio of customers actively redeeming bonuses to those offered, provided invaluable insights into the appeal and utility of the bonus system. The equation for bonus redemption rate (BRR) is represented as:

Secondly, using comprehensive analysis in Tableau, the Activity on Bonus Days vs. Cash Days was meticulously compared. This analysis delved into customer engagement levels before and after bonus allocation, offering vital insights into how bonuses influence customer

behavior and retention. This comparison was instrumental in understanding the impact of bonuses on player engagement and served as a cornerstone for optimizing bonus strategies.

Lastly, the Win Rate and Churn Rate were carefully examined to further refine bonus allocation strategies. The win rate, calculated as the percentage of successful outcomes upon receiving bonuses, provided insights into customer satisfaction and desired outcomes. Conversely, leveraging advanced predictive modeling, the churn rate was assessed to ascertain if bonuses acted as a mitigating factor. Additionally, a comprehensive churn factor, considering recency and latency metrics, was calculated to capture the essence of customer behavior. The churn factor equation is represented as:

$$BRR = \frac{\text{Number of customers redeeming bonuses}}{\text{Total number of customers offered bonuses}}$$

Lastly, the Win and Churn Rates were carefully examined to further refine bonus allocation strategies. The win rate, calculated as the percentage of successful outcomes upon receiving bonuses, provided insights into customer satisfaction and desired outcomes. Conversely, leveraging advanced predictive modeling, the churn rate was assessed to ascertain if bonuses acted as a mitigating factor. Additionally, a comprehensive churn factor, considering recency and latency metrics, was calculated to capture the essence of customer behavior. The churn factor equation is represented as:

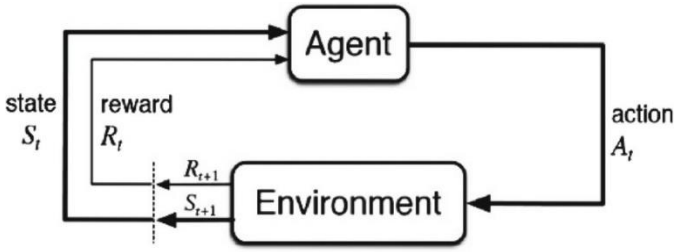
$$ChurnFactor = \frac{\text{Recency}}{\text{Latency}}$$

2.3 MODEL IMPLEMENTATION

Reinforcement learning (RL) is a sub-field of machine learning that enables AI-based systems to take actions in a dynamic environment through trial-and-error methods to maximize collective rewards based on feedback generated from respective actions. In RL, an agent interacts with an environment by performing actions and receiving rewards, aiming to learn the best strategies or policies to achieve the highest cumulative rewards over time.

On-Policy Learning: In this approach, the learning agent learns the value function according to the current action derived from the policy it is currently using. The agent follows the policy that it is trying to evaluate or improve, making decisions and updating the value function based on the actions taken. SARSA (State-Action-Reward-State-Action) is a common on-policy algorithm, where the value of the current policy is updated using the action chosen by the policy itself.

Off-Policy Learning: In contrast, off-policy learning involves the agent learning the value function according to actions derived from a different policy. This allows for the evaluation and improvement of one policy while following another. Q-learning is a popular off-policy algorithm, where the value function is updated based on the maximum reward of the next state-action pair, regardless of the policy currently being followed.



2.3.1 Q LEARNING MODEL

Q-learning is an off-policy reinforcement learning algorithm designed to learn the optimal action-selection policy for any given finite Markov decision process (MDP). For bonus optimization, Q-learning determines the best bonus allocation strategy that maximizes player engagement, retention, and revenue generation. The implementation begins with the setup of the environment, where the state space includes customer metrics like engagement levels, churn risk scores, and historical bonus usage, representing various scenarios for making bonus decisions. The action space encompasses different types and amounts of bonuses that can be allocated, with each action representing a specific bonus strategy.

The core of the Q-learning model is the reward function, which balances immediate customer engagement and long-term retention. Positive rewards are assigned for actions leading to increased engagement and retention, while negative rewards penalize actions that result in bonus abuse or reduced profitability. This reward mechanism is crucial for guiding the model towards strategies that maximize overall player satisfaction and business outcomes. The reward function can be mathematically expressed, ensuring clear criteria for the model's learning process.

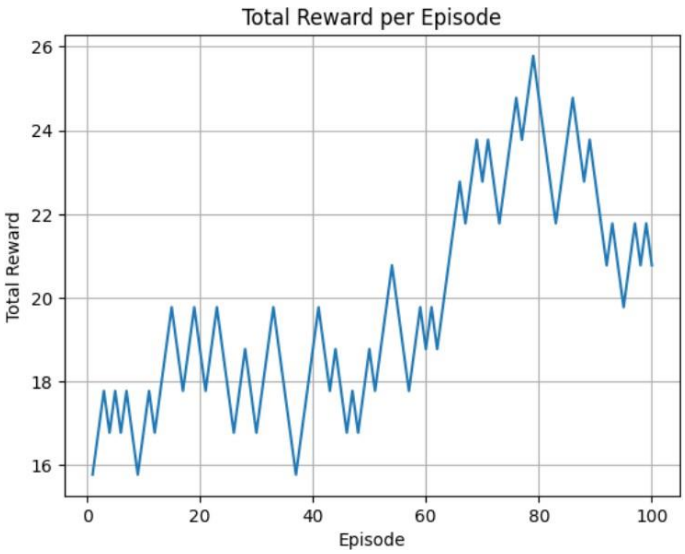
The algorithm uses the update rule:

$$Q(s,a) \leftarrow Q(s,a) + \alpha[r + \gamma \max_{a'} Q(s',a') - Q(s,a)]$$

where α is the learning rate, γ is the discount factor, r is the reward, s′ is the next state, and represents the optimal future reward. After sufficient training, the optimal

policy is extracted, dictating the best action in each state to maximize cumulative rewards.

The effectiveness of the Q-learning model is evaluated using metrics such as average reward, retention rates, and revenue impact, with cross-validation and A/B testing ensuring robustness and practical applicability.



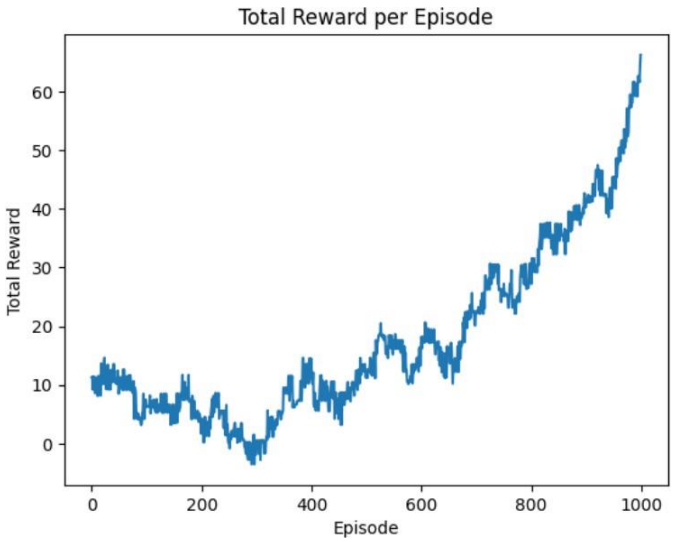
2.3.2 SARSA MODEL

SARSA (State-Action-Reward-State-Action) is an on-policy reinforcement learning algorithm that helps determine the best bonus allocation strategy by updating the policy based on the actions taken and rewards received. In the context of bonus optimization, the SARSA model is implemented to continuously refine the strategy for allocating bonuses to players, ensuring that the policies are updated based on actual player interactions. The state space includes customer metrics like engagement levels and churn risk scores, while the action space comprises different types and amounts of bonuses. This approach is particularly useful for real-time adjustments in bonus strategies, aligning with the dynamic nature of player behavior in the i-gaming industry.

The SARSA algorithm is designed to encourage actions that enhance immediate customer engagement and long-term retention. Positive rewards are assigned for actions that lead to increased player activity and retention, while negative rewards are given for actions resulting in bonus abuse or reduced profitability. The model uses specific parameters to ensure effective learning: a learning rate (α) of 0.1, a discount factor (γ) of 0.99, an epsilon (ϵ) of 0.1 for the epsilon-greedy policy, a threshold of 0.9 for bonus utilization, a replay buffer size of 10,000, and a batch size of 32 for experience replay. The SARSA update rule is as follows:

$$Q(s,a) \leftarrow Q(s,a) + \alpha[r + \gamma Q(s',a') - Q(s,a)]$$

where α controls the extent of new information overriding old information, and γ represents the importance of future rewards.



2.4 Model STORAGE AND PERSISTANCE

Following the successful training of the Q-learning and SARSA models for bonus optimization, it is imperative to establish a robust system for storing these models to ensure their preservation and future applicability. The storage of a Q-learning model entails the preservation of the learned Q-table, which maps state-action pairs to corresponding values. Similarly, SARSA mandates the preservation of learned Q-values for each state-action pair.

In this context, the Joblib library emerges as a suitable tool for the storage of these models. Joblib provides an efficient mechanism for serializing Python objects, rendering it apt for the preservation and retrieval of moderately large Q-tables. This choice aligns with the imperative of maintaining the integrity and accessibility of our trained models, facilitating their seamless integration into future applications or environments.

The utilization of Joblib enables the straightforward reloading of stored Q-tables, obviating the necessity for retraining the models ab initio. This approach not only conserves computational resources but also expedites the deployment of our bonus optimization strategies in real-world scenarios. It is worth noting that while Joblib serves as an optimal

solution for our current requirements, alternative methods such as pickle or database storage may be considered for specific use cases characterized by differing scalability or persistence demands.

3. CHURN PREDICTION USING COX PROPORTIONAL HAZARDS MODEL

3.1 INTRODUCTION

In the highly competitive i-gaming industry, understanding customer behavior and predicting player retention are crucial for optimizing revenue and enhancing player engagement. The objective of this work is to develop a robust churn prediction model using the Cox Proportional Hazards Model, a statistical technique widely used in survival analysis. This model identifies

key factors influencing player retention and predicts the likelihood of new players making a second deposit within the first 14 days of their initial deposit.

The focus is on new players first 14 days of activity, a critical period for determining long-term engagement. Players who make a second deposit within this period are likely to stay active, while those who do not are at risk of churning. To build the predictive model, historical data on player activity, including features such as deposit amount, return to player (RTP), and win amount, are analyzed. Players who made a second deposit within 14 days are marked with an event of one, indicating retention, while others are marked with an event of zero, indicating potential churn.

Feature selection is critical in model development. The correlation between features is examined, and various transformation techniques such as power, quantile, and spline transformers are employed to handle non-linear relationships. Partial correlation refines feature selection, ensuring only relevant predictors are included. The Cox Proportional Hazards Model is then fitted to the data, retaining features with a p-value less than 0.05 for their statistical significance. The model's performance is evaluated using concordance, and the probability of each player making a second deposit is calculated by subtracting the predicted hazard probability from one. This approach provides actionable insights into player retention, enabling targeted interventions to enhance customer loyalty and drive revenue growth.

3.2 MODEL IMPLEMENTATION

The Cox proportional hazards model, often referred to simply as the Cox model, is a statistical technique used in survival analysis to examine the association between the survival time of subjects and one or more predictor variables.

In survival analysis, the survival function and the hazard function are two key concepts used to describe the distribution of survival times. Here's a detailed explanation of each:

The survival function, denoted as S(t), represents the probability that an event (such as a player's second deposit) has not occurred by time t. In other words, S(t) gives the likelihood that a player will "survive" without making a second deposit up to day t. It is a decreasing function where S(0)=1 and as t increases, S(t) approaches 0, indicating that the probability of not making a second deposit diminishes over time.

The hazard function, denoted as $h(t)$, describes the instantaneous risk of the event occurring at time t, given that the individual has survived (not experienced the event) up to that time. It represents the rate at which players are making their second deposit at day t, conditional on not having done so before t. The hazard function is crucial for understanding the event occurrence over time and provides insights into the risk patterns associated with predictors.

Assumptions:

- the effects of the predictor variables upon survival are constant over time and are additive in one scale.

The coefficients in a Cox regression relate to hazard:

- a positive correlation indicates a worse priagnosis
- a negative coefficient indicates a protective effect of the variable with which it is

associated.

3.2.1 DATA EXTRACTION AND PREPROCESSING

The data used for this work is sourced from the i-gaming platform's transactional and interaction logs, capturing detailed player activities such as deposits, bets, wins, losses, and bonus redemptions. This comprehensive dataset, includes key features like deposit amount, return to player (RTP), win amount, and bet frequency. To efficiently retrieve and manage this extensive data, AWS S3 was utilized for storage due to its scalability and security. Data access was facilitated through boto3, the AWS SDK for Python, ensuring secure handling of AWS credentials and establishing a connection to S3. Additionally, AWS Wrangler was employed to read data directly from S3 into Pandas DataFrames, minimizing retrieval delays and ensuring efficient data handling

Missing values in the dataset were identified and handled using specific techniques to maintain data integrity. For numerical features, missing values were imputed using the median value of the respective feature to preserve the central tendency of the data, implemented using the SimpleImputer class from the sklearn.impute.

3.2.2 FEATURE ENGINEERING

For the churn prediction model, key features from the player's first day of activity were extracted, including deposit amount, Return to Player (RTP), win amount, and other relevant metrics. These features were selected based on their potential impact on player behavior and their ability to predict subsequent deposits.

To build the model, historical data was analyzed to observe player behavior over the first 14 days. Players who made a second deposit within the first 14 days were marked with an event label of one, indicating retention. Players who did not make a second deposit within this period were marked with an event label of zero, indicating potential churn. This labeling process provided a clear target variable for the survival analysis model.

In survival analysis, it's crucial to account for censoring. Censoring occurs when the event of interest (in this case, a second deposit) has not occurred for some players by the end of the observation period. These players are still active but have not yet made a second deposit. To handle censoring, each player's observation period was tracked, and censoring was appropriately marked in the dataset.

Certain features were transformed to better capture their relationships with the target variable. Various transformations were applied to stabilize variance and ensure normal distribution of features, enhancing the model's predictive accuracy:

- Power Transformer: Applied to features with skewed distributions to make them more Gaussian-like. This helps stabilize variance and reduce skewness, ensuring that the feature values are more evenly spread out.
- Quantile Transformer: Used to transform features to follow a uniform or normal distribution. This method is particularly useful for handling outliers and making the feature distribution more consistent.
- Spline Transformer: Applied to capture non-linear relationships by transforming the

data into a higher-dimensional space where linear models can perform better.

Partial correlation analysis was employed to refine feature selection, ensuring that only the most relevant predictors were included in the model. Partial correlation helps in understanding the direct relationships between features and the target variable by controlling for the effects of other variables. This method allows for the identification of features that have a direct and significant impact on the target, improving the model's robustness and accuracy. Features with high partial correlations to the target were retained, while those with low relevance were excluded. Additionally, partial correlation analysis was conducted to refine feature selection, ensuring that only the most relevant predictors were included in the model.

3.2.3 MODEL CONSTRUCTION

Following feature selection, the Cox Proportional Hazards model was employed for churn prediction, leveraging its robustness in handling time-to-event data. This model is well-suited for survival analysis tasks, as it estimates the hazard function, representing the instantaneous risk of the event (churn) occurring at a given time, conditioned on survival up to that time. The Cox model is semi-parametric, meaning it does not assume a specific baseline hazard function, making it flexible and powerful for various survival data scenarios.

To implement the model, the lifelines package in Python was used, particularly the CoxPHFitter class, which facilitates fitting the Cox Proportional Hazards model to the data. The preprocessed dataset, containing the selected features and labeled events (churn or no churn), was fed into the model. The process accounted for censoring, where the event of interest (second deposit) had not yet occurred for some players by the end of the observation period.

The fitting procedure involved estimating the regression coefficients for each feature, which indicate the direction and magnitude of their impact on the hazard rate. The model's coefficients were derived by maximizing the partial likelihood, a method that focuses on the ordering of event times rather than their specific values. This ensures robust estimation even in the presence of censored data.

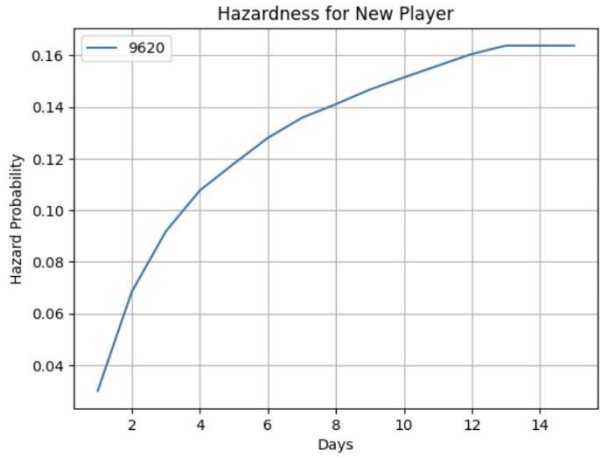
After fitting the model, the significance of each feature was assessed using p-values. Features with p-values less than 0.05 were considered statistically significant and retained for further evaluation. This threshold ensures that the included features have a meaningful impact on the model's predictions, reducing noise and improving interpretability.

The performance of the fitted model was evaluated using the concordance index (C-index), which measures the model's discriminative ability to correctly rank the event times. A higher

C-index indicates better predictive performance. Additionally, while the Cox model predicts the probability of the event (churn) not happening, the desired output was the probability of a player making a second deposit. This was achieved by subtracting the predicted hazard rate from one, providing the probability of retention for each player over time.By leveraging the Cox Proportional Hazards model and advanced preprocessing techniques, the churn prediction model effectively identified players at risk of churn and estimated the probability of subsequent deposits, facilitating targeted retention strategies and optimizing player engagement for the i-gaming platform.

3.3 MODEL LIMITATION AND SOLUTION

One disadvantage of the Cox Proportional Hazards model is its assumption of proportional hazards, which assumes that the hazard rate ratio between different groups remains constant over time. This assumption might not hold true in all situations, particularly when the effect of covariates on the hazard rate changes over time. When this assumption is violated, the model's estimates and predictions can be biased and inaccurate.




```
Probability of making a second deposit within 14 days: 0.16
```

The Accelerated Failure Time (AFT) model addresses this limitation by modeling the survival times directly, rather than the hazard rates. The AFT model assumes that covariates accelerate or decelerate the life time of an event, rather than altering the hazard rate proportionally. This allows for a more flexible approach in scenarios where the effect of covariates changes over time, providing a potentially better fit for the data.

The Accelerated Failure Time (AFT) model is another approach used in survival analysis, providing an alternative to the Cox proportional hazards model. While the Cox model focuses on the hazard function, the AFT model directly models the survival times. The key idea of the AFT model is that it assumes the effect of covariates accelerates or decelerates the life time of an individual or item by a constant factor.

The AFT model describes how covariates influence the survival time by accelerating or decelerating it. Specifically, it posits that the logarithm of the survival time is a linear function of the covariates:

T is the survival time.

X1,X2,…,XpX_1, X_2, \ldots, X_pX1,X2,…,Xp are the predictor variables.

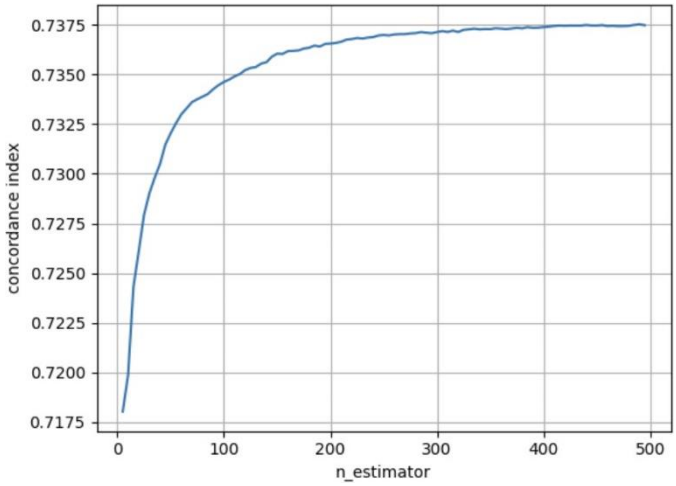
β0,β1,…,βp are the coefficients.

ϵ is the error term

The AFT model directly models the survival time rather than the hazard function, providing an intuitive understanding of how predictors impact the duration until the event.The coefficients β represent the acceleration factors. A positive β means that the covariate extends

the survival time (deceleration), while a negative β means it shortens the survival time

he error term ϵ can follow different distributions, making the AFT model flexible to fit various types of survival data.



3.4 INTEGRATING MODEL TO STREAMLIT

By integrating the survival analysis models with Streamlit, we have created an interactive tool that helps visualize the performance of different regularization strategies. This tool provides valuable insights into which regularization method yields the best predictive performance for player retention in the iGaming industry. The use of Streamlit ensures that the application is accessible and user-friendly, allowing stakeholders to make data-driven decisions.

4. CONCLUSION

The Bonus Optimization work successfully enhanced player engagement and retention by leveraging advanced machine learning techniques and data analytics to optimize bonus distribution strategies. Through in-depth analysis of player behavior and responsiveness to various bonus types, we were able to gain critical insights into what motivates different segments of our player base. The implementation of predictive models allowed for the personalization of bonus offers, leading to improved player satisfaction and retention. Moreover, the use of optimization algorithms ensured that our bonus distribution was both effective and efficient, maximizing player engagement while optimizing marketing expenditures.

The Churn Prediction research work aimed to identify players at risk of leaving the platform by predicting the likelihood of a second deposit within the first 14 days—a key indicator of long-term engagement. By employing survival analysis models, such as the Cox proportional hazards model and the Accelerated Failure Time (AFT) model, and incorporating regularization techniques, we were able to create highly accurate predictive models. The careful selection and transformation of features, alongside the development of an interactive Streamlit interface, allowed for clear visualization and interpretation of the model results. This

enabled stakeholders to make informed, data-driven decisions to proactively retain at-risk players.

Both the Bonus Optimization and Churn Prediction work have significantly advanced our capabilities in understanding and influencing player behavior. By integrating sophisticated analytical tools and machine learning models, we have developed effective strategies to optimize player retention and maximize lifetime value.

References

[1] Fullerton, Andrew S., and Kathryn Freeman Anderson. "Ordered regression models: A tutorial." Prevention Science (2023): 1-13.

[2] Carmona, René, Mathieu Laurière, and Zongjun Tan. "Model-free mean-field reinforcement learning: mean-field MDP and mean-field Q-learning." The Annals of Applied Probability 33.6B (2023): 5334-5381.

[3] Moradimaryamnegari, Hoomaan, Marco Frego, and Angelika Peer. "Model Predictive Control-Based Reinforcement Learning Using Expected Sarsa." IEEE Access 10 (2022): 81177-81191.

[4] Khorasani, Mohammad, Mohamed Abdou, and J. Hernández Fernández. "Web Application Development with Streamlit." Software Development (2022): 498-507

[5] Cox, David R. "Regression Models and Life-Tables." Journal of the Royal Statistical Society: Series B (Methodological), vol. 34, no. 2, 1972, pp. 187–220.

[6] Pedregosa, Fabian, et al. "Scikit-learn: Machine Learning in Python." Journal of Machine Learning Research, vol. 12, 2011, pp. 2825–2830.

[7] Streamlit Documentation. "Streamlit: The Fastest Way to Build Data Apps." Streamlit, https://docs.streamlit.io/.

[8] Sutton, Richard S., and Andrew G. Barto. Reinforcement Learning: An Introduction. MIT Press, 2018.

[9] Zhao, Yi, and George Karypis. "Data-Driven Strategies for Enhancing User Engagement and Retention." ACM Transactions on Knowledge Discovery from Data (TKDD), vol. 12, no. 4, 2018, pp. 1–28.