Machine Learning in Marketing

Dipesh Yadav
B.tech Computer Science

A dissertation submitted for the degree of
Master of Science in Data Science

Supervised by
Williams, Bryan
Rolfe, Joe
Goring, Seb

School of Computing and Communications
Lancaster University

September, 2024

Declaration

I declare that the work presented in this dissertation is, to the best of my knowledge and belief, original and my own work. The material has not been submitted, either in whole or in part, for a degree at this, or any other university. Estimated word count is: **13696**

Name: **Dipesh Yadav**
Date: **September, 2024**

Machine Learning in Marketing

Dipesh Yadav, B.tech Computer Science.
School of Computing and Communications, Lancaster University

A dissertation submitted for the degree of *Master of Science* in Data Science.
September, 2024

Abstract

This dissertation explores the development of a machine learning model to predict customer deposit behavior on an online platform. The goal is to create an efficient predictive model that can assist in optimizing marketing strategies by identifying customers likely to make deposits, thereby enhancing engagement and profitability. The research focuses on analyzing the impact of various factors such as bonuses, transactional activity, and temporal behavior on customer decision-making. The study begins with extensive exploratory data analysis (EDA) to uncover key patterns and correlations between customer activities, bonuses, and deposit behavior. Binary and multiclass classification approaches are considered, with the objective of selecting the most appropriate target variable for the model. The research then tackles the challenges posed by class imbalance through statistical sampling techniques and applies advanced machine learning algorithms, such as LightGBM and Random Forest, to capture the complex interactions between features. Hyperparameter tuning through grid search is employed to optimize the model's performance, and additional evaluation metrics like Kolmogorov-Smirnov (KS), Gain, and Lift are explored to assess model effectiveness beyond traditional precision, recall, and accuracy scores. The dissertation further evaluates the model's generalizability by testing it on live data, providing insights into its real-world applicability. This research provides a comprehensive exploration of feature engineering, model selection, sampling strategies, and performance evaluation techniques to develop a robust solution for predicting customer behavior.

Acknowledgements

*I want to express my deep gratitude to my academic supervisor, **Bryan Williams**, for his support and guidance throughout the journey of the research and implementation of this project.*

*I want to extend my sincere thanks to the Flutter Entertainment company for this opportunity. I am very grateful to my professional supervisors **Joe Rolfe, Seb Goring** and the entire team of Flutter data science for their support, insights and guidance during this project.*

I want to acknowledge the use of course modules MATH403: Statistical Foundations I, MATH404: Statistical Learning, SCC 462: Distributed Artificial Intelligence, and SCC403: Data Mining for the successful completion of this project.

Contents

1 Introduction **1**

1.1 Section Name . . . 1

1.1.1 Motivation . . . 1

1.1.2 Objectives . . . 2

2 Background and Literature Review **4**

2.1 Data Analysis in Gaming and Gambling . . . 4

2.2 Sampling Techniques . . . 4

2.2.1 Hybrid Sampling and Gradient Boosting . . . 5

2.2.2 Algorithm-Level Modifications . . . 5

2.2.3 Data-Level Approach . . . 5

2.2.4 Cost-Sensitive Learning . . . 5

2.2.5 Ensemble Techniques . . . 5

2.2.6 Application in Digital Gaming and Betting . . . 6

2.3 Feature Creation . . . 6

2.3.1 Clickstream Data for Consumer Insight . . . 6

2.3.2 Predictive Framework for E-commerce . . . 6

2.3.3 Application in Digital Gaming and Betting . . . 7

2.4 Target Variable in Imbalanced Classification . . . 7

2.4.1 Multi-class Imbalanced Classification . . . 7

2.4.2 Binary Class Imbalance . . . 7

2.5 Model Considerations . . . 7

2.5.1 Random Forests . . . 7

2.5.2 Boosting Algorithms . . . 8

2.5.3 Application in Digital Gaming and Betting . . . 8

2.6 Model Tuning . . . 8

2.6.1 Grid Search, Random Search, and Genetic Algorithms . . . 8

2.6.2 Application in Digital Gaming and Betting . . . 9

2.7 Model Evaluation . . . 9

2.7.1 F1 Score, AUC-ROC, and Accuracy . . . 9

2.7.2 KS Statistic, Gain, and Lift 10

3 Methodology **11**

3.1 Justification for Ensemble Techniques 11

3.1.1 Ensemble methods vs Single predictor models 11

3.1.2 Latest Churn Models 11

3.1.3 Large Dataset Models 11

3.2 Bagging and Boosting 12

3.2.1 Bagging (Bootstrap Aggregating) 12

3.2.2 Boosting 12

3.3 Random Forest 12

3.3.1 Mathematical Representation 13

3.3.2 Cross-validation in Random Forest 13

3.3.3 K-fold Cross-validation 13

3.3.4 Hyperparameter Tuning 13

3.3.5 Key Hyperparameters 13

3.3.6 Optimization Techniques: Randomized Search and Grid Search . . . 14

3.3.6.1 Randomized Search 14

3.3.6.2 Grid Search 14

3.4 LightGBM 14

3.4.1 Gradient Boosting vs. Other Ensemble Methods 14

3.5 Key Innovations of LightGBM 15

3.5.1 Gradient-based One-Side Sampling (GOSS) 15

3.5.2 Exclusive Feature Bundling (EFB) 15

3.6 Model Fitting and Prediction 15

3.6.1 Mathematical Representation 15

3.6.2 Additive Strategy 16

3.7 Hyperparameter Tuning 16

3.8 Evaluation 16

3.8.1 Evaluation Strategy 17

3.8.2 Level 1 Evaluation 17

3.8.2.1 Model Assessment Techniques 17

3.8.2.2 Derived Metrics 17

3.8.3 Level 2 Evaluation 18

3.8.3.1 Advanced Comparative Metrics 18

3.8.3.2 Gains Table and Ks-Score 18

3.8.3.3 Visualization and Decision Metrics 18

3.8.4 Level 3 Evaluation 18

4 Experimental Results **19**
4.1 The Initial Data . . . 19
4.1.1 Data Collection and Structure . . . 19
4.1.2 Detailed Table Schema Analysis . . . 20
4.1.2.1 Bonus Information Table . . . 20
4.1.2.2 Stakes and Winnings Table . . . 21
4.1.2.3 Deposit and Withdrawal Information Table . . . 21
4.1.3 Analysis and Dataset Creation . . . 22
4.2 Data Preparation and Table Creation . . . 22
4.2.1 Initial Data Retrieval and Processing . . . 22
4.2.2 Creation of Intermediate Tables . . . 22
4.2.3 Class Assignment and Dataset Finalization . . . 23
4.2.4 Aggregation of Betting Data . . . 23
4.2.5 Final Tables for Binary and Multiclass Classification . . . 23
4.2.6 Schema of the Final Tables . . . 23
4.3 Exploratory Data Analysis . . . 25
4.3.1 Binary Target Variable Analysis . . . 25
4.3.2 Temporal Data Analysis . . . 26
4.3.3 Relationship Between Features . . . 27
4.3.4 Feature Selection for Modelling . . . 27
4.3.5 Handling Missing Values . . . 29
4.3.6 Insights and Justification for Model . . . 29
4.4 Binary vs Multi-class Model . . . 30
4.4.1 Binary Classification . . . 31
4.4.1.1 Conclusion on Binary Classification Performance . . . 31
4.4.2 Multi-class Classification . . . 32
4.4.2.1 Conclusion on Multi-class Classification Performance . . . 33
4.4.3 Observations and Justification for Choosing Binary Classification . . . 33
4.5 Random Sampling vs. Statistical Sampling . . . 34
4.5.1 Random Sampling . . . 34
4.5.2 Statistical Sampling . . . 34
4.5.3 Observations from Random Sampling . . . 34
4.5.4 Observations from Statistical Sampling . . . 35
4.5.5 Comparison of Random Sampling and Statistical Sampling . . . 36
4.5.6 Significance of Statistical Sampling in Model Training . . . 36
4.5.6.1 Addressing Class Imbalance . . . 36
4.5.6.2 Improved Class Separation . . . 37
4.5.6.3 Better Alignment with Business Objectives . . . 37
4.6 Model Selection . . . 37
4.6.1 Performance of Random Forest . . . 38

4.6.2 Performance of LightGBM 39
4.6.3 Key Findings and Justifications 40
4.7 Analysis of LightGBM Performance on Live Data 42
4.7.1 Performance on Live Data vs Validation Data 42
4.7.2 Conclusions from Live Data Performance 43

5 Discussion 45
5.1 Problem Context 45
5.2 Data Exploration and Initial Observations 46
5.3 Model Selection: Binary vs. Multiclass Classification 46
5.4 Sampling Techniques: Random vs. Statistical Sampling 47
5.5 Model Tuning and Evaluation 47
5.6 Performance on Live Data 47
5.7 Significance of Findings 48

6 Conclusions and Future Work 49
6.0.1 Problem Definition and Data Understanding 49
6.0.2 Model Selection: Binary vs. Multiclass Classification 49
6.0.3 Addressing Class Imbalance: Sampling Techniques 50
6.0.4 Model Tuning and Evaluation 50
6.0.5 Performance on Live Data 50
6.0.6 Key Findings 51
6.0.7 Practical Implications 51
6.0.8 Limitations and Future Work 51

References 52

Appendix A Motivation 54
A.1 Motivation 54
A.2 Project Aim 54
A.3 Data 55
A.4 Project Timeline 55
A.5 Deliverables 56

List of Figures

4.1 Database Schema 20
4.2 Database Schema 24
4.3 Database Schema 25
4.4 Reward Type 26
4.5 Deposit hourly Distribution 27
4.6 Weekly Distribution 28
4.7 Monthly Distribution 29
4.8 Variable Relation 30
4.9 Random Sample Distribution 35
4.10 Statistical Sample 35
4.11 Random Forest 38
4.12 LightGBM 40
4.13 ROC Curve 41
4.14 Feature Importance 43

List of Tables

4.1 Confusion Matrix for Binary Classification 31
4.2 Classification Report (Validation - LightGBM) 31
4.3 Confusion Matrix for Multi-class Classification 32
4.4 Classification Report (Validation) 32
4.5 Random Classification Report 34
4.6 Statistical Classification Report 36
4.7 Confusion Matrix - Random Forest 37
4.8 Classification Report - Random Forest 37
4.9 Confusion Matrix - LightGBM 38
4.10 Classification Report - LightGBM 38
4.11 KS, Gain, and Lift for Random Forest 39
4.12 KS, Gain, and Lift for LightGBM 40
4.13 Confusion Matrix 42
4.14 Classification Report 42

Chapter 1

Introduction

1.1 Section Name

Flutter owns some of the leading and most well-known online sports betting and iGaming brands globally, such as FanDuel, Sky Betting & Gaming, Sportsbet, PokerStars, Paddy Power, Sisal, Tombola, Betfair, Junglee Games, Adjarabet, and MaxBet. With an unrivaled collection of top-tier brands, extensive global reach, and a challenger mindset, this company delivers exciting and entertaining experiences to its customers.

In the competitive landscape of digital entertainment and gaming, understanding and influencing customer behaviour through targeted marketing efforts is pivotal for sustaining growth and profitability. Flutter Entertainment Ltd., a leader in this dynamic industry, leverages data-driven strategies to enhance customer engagement and optimize marketing expenditures. The company focuses on the application of advanced analytical techniques to predict customer responses to bonus-based promotions, aiming to identify the most valuable customers on a short-term basis and tailor promotions effectively.

This report aims to explore the most effective techniques for sampling, feature creation, feature selection, model comparison, model tuning, model evaluation, and model selection in predicting valuable customers on a short-term or daily basis. The goal is to optimize class separation of model for bonus-based promotions for improved return on investment. The analysis will be conducted using customer data provided by Flutter Entertainment Ltd. for this investigation.

1.1.1 Motivation

In the digital era, online gambling and gaming platforms have gained significant popularity, driven by technological advancements and ease of access. These platforms generate vast amounts of data related to user behavior, transactions, and interactions. Effectively analysing this data is critical for improving user engagement, identifying fraudulent activities, and optimizing business processes.

The integration of data science techniques into such platforms enables businesses to derive actionable insights from user interactions, such as bonus utilization, betting patterns, and financial transactions (deposits and withdrawals). Leveraging machine learning models, coupled with detailed exploratory data analysis (EDA), can help predict user behaviour, such as whether a user will make a deposit within 24 hours of receiving a bonus. Understanding this behaviour can drive personalized marketing strategies, enhance customer experience, and ensure regulatory compliance.

Research from the e-commerce sector underscores the efficacy of personalized marketing approaches, demonstrating substantial improvements in customer engagement and conversion rates. These insights offer valuable lessons for the gaming and entertainment sectors, where customer behaviours exhibit significant variability, and the impact of targeted promotions can be immediately quantifiable.

In the gaming and entertainment sector, where customer preferences and behaviours are rapidly evolving, the ability to swiftly adapt and customize promotional strategies can yield a decisive competitive edge. Recognizing the potential for improved return on investment (ROI) through effective bonus allocations, Flutter Entertainment Ltd. is committed to refining its promotional tactics based on robust analytical insights. This approach not only aims to maximize immediate financial returns but also strengthens long-term customer loyalty and engagement.

1.1.2 Objectives

The primary objectives of this report are outlined as follows:

1. **Advanced Sampling Techniques**: To implement and assess various sampling methods that ensure a representative and efficient subset of the entire customer dataset, facilitating more accurate and scalable predictive modelling.

2. **Feature Creation and Selection**: To develop and refine features that capture the essence of customer behaviours and their responsiveness to promotions, using advanced statistical and machine learning techniques to select the most predictive features for the models.

3. **Model Comparison and Tuning**: To evaluate multiple predictive models in terms of effectiveness and efficiency, fine-tuning model parameters to enhance their predictive accuracy and operational feasibility.

4. **Model Evaluation and Selection**: To rigorously assess the performance of each model using a suite of metrics, selecting the best-performing model based on its ability to predict short-term valuable customers and its applicability to real-world promotional strategies.

5. **Optimization of Promotional Strategies**: To apply the insights gained from the predictive models to optimize bonus-based promotions, aiming to increase the ROI by targeting customers predicted to have high short-term value.

The primary objective of this report is to explore advanced data analytics techniques to predict customer behaviour that signals high short-term value, particularly in response to bonus-based promotions. By using customer data provided by Flutter Entertainment Ltd., this analysis will identify which customers are most likely to respond to specific promotions. The ultimate goal is to improve promotional strategies to drive better engagement, long-term customer loyalty, and higher ROI for the business.

Chapter 2

Background and Literature Review

2.1 Data Analysis in Gaming and Gambling

The gaming and gambling industry generates massive amounts of data daily, encompassing user interactions, financial transactions, betting patterns, and engagement metrics. This data holds significant value for gaining insights into user behaviour and creating predictive models that can enhance business operations, tailor user experiences, and identify potentially fraudulent activities. The main goal of data-driven models in this sector is to analyse historical data and produce actionable insights that support better decision-making. With the advent of big data technologies, advanced machine learning methods, and cloud infrastructure, companies are increasingly turning to predictive analytics to stay competitive.

However, key challenges in this field include integrating various data sources—such as transactional records, user behaviour patterns, and promotional offers (like bonuses) as well as the necessity for real-time processing of large data volumes. Predictive models typically aim to address issues like user churn, bonus redemption prediction, stake estimation, and the detection of anomalies in financial transactions.

2.2 Sampling Techniques

The challenge of class imbalance in predictive modeling is well recognized across various fields, including transportation and digital gaming, where the objective is often to predict outcomes or behaviors based on historical data. Class imbalance occurs when some classes in the data are underrepresented, leading to biased predictions favoring the majority class. The literature presents several methodologies to handle this issue, improving the robustness and accuracy of predictive models.

2.2.1 Hybrid Sampling and Gradient Boosting

This approach combines gradient boosting with data resampling techniques to address class imbalances effectively. By integrating different sampling strategies, such as under-sampling the majority class and over-sampling the minority class, predictive models can better generalize across the actual class distribution in real-world scenarios. This is particularly effective in fields like transportation, where certain events (like specific types of commercial vehicle activities) may occur infrequently but are crucial for accurate predictions. The model described by Low demonstrates substantial improvement in predictive accuracy through a hybrid sampling approach that compensates for the imbalanced data distribution(Low, Cheah, and You, 2020).

2.2.2 Algorithm-Level Modifications

Lin, Lee, and Wahba discuss modifications to Support Vector Machines (SVM) that make them more sensitive to the minority class by adjusting the classification threshold or modifying the kernel function to emphasize the cost of misclassifications. This method is particularly useful when dealing with nonstandard situations where the cost associated with different types of misclassification varies significantly, a common scenario in disease diagnosis or fraud detection(Lin, Lee, and Wahba, 2002).

2.2.3 Data-Level Approach

Batista explore several methods for rebalancing class distributions, such as random over-sampling and under-sampling. Their research highlights the effectiveness of manipulating the training data to create a more balanced dataset, which in turn helps machine learning algorithms to perform better by not being biased towards the majority class(Batista, Prati, and Monard, 2004).

2.2.4 Cost-Sensitive Learning

This technique integrates cost considerations directly into the learning algorithm, making the model sensitive to the class distribution. Chawla extend this concept by embedding cost into the learning process, which is particularly effective when the costs of different types of errors vary. Such approaches are adaptive and can be tailored to specific needs of the problem at hand, providing a flexible framework for addressing class imbalance(Chawla et al., 2008).

2.2.5 Ensemble Techniques

Galar review several ensemble methods designed to improve predictions in the presence of class imbalance. These methods, including boosting and bagging, use multiple learning algorithms to obtain better predictive performance than could be obtained from any of the constituent learning algorithms alone. They discuss cost-sensitive boosting, which adjusts the weights of classifiers at each iteration to focus more on the minority class, and hybrid ensembles, which combine several of these techniques to benefit from their individual strengths(Galar et al., 2011).

2.2.6 Application in Digital Gaming and Betting

In the context of Flutter Entertainment, which operates several online gaming and betting platforms, these sampling and algorithmic strategies can be crucial. The digital gaming sector often deals with highly skewed datasets, where user behaviors such as response to promotions or engagement patterns may vary significantly across different segments of users. Effective application of these techniques can lead to more personalized user experiences, targeted marketing campaigns, and improved customer satisfaction.

2.3 Feature Creation

2.3.1 Clickstream Data for Consumer Insight

Schellong, Kemper, and Brettel (2017) explored the utilization of big data clickstream information to generate consumer insights, highlighting how such data can be linked to transaction-related shopping behaviors. Their study focused on using clickstream data to categorize consumer behaviors into distinct shopping types such as ”Buying”, ”Searching”, ”Browsing”, or ”Bouncing”. The authors employed clustering techniques to analyze off-site clickstream data, illustrating how different patterns of site engagement correlate with transaction behaviors. Their findings suggest that more goal-directed behaviors, as captured by clickstream data, are associated with higher purchase propensities(Schellong, Kemper, and Brettel, 2017).

2.3.2 Predictive Framework for E-commerce

Cirqueira presented a comprehensive framework for predicting customer purchase behaviors in e-commerce. Their conceptual model outlined three primary tasks: Predicting Customer Intent (PCI), Predicting Buying Session (PBS), and Predicting Purchase Decisions (PPD). This framework categorizes the predictive tasks and elucidates the methodologies employed for each, including the data types and predictive models used. The research highlights the application of various machine learning techniques and the significance of effectively constructed features to predict diverse aspects of customer behavior accurately(Cirqueira et al., 2019).

2.3.3 Application in Digital Gaming and Betting

Both papers emphasize the importance of integrating diverse data sources, including clickstream data and transaction histories, to develop robust predictive models. This integration allows for a more comprehensive understanding of customer behaviors and enhances the accuracy of predictions.

2.4 Target Variable in Imbalanced Classification

2.4.1 Multi-class Imbalanced Classification

Fernández provide a comprehensive examination of the challenges inherent in multi-class imbalanced classification. The study begins by outlining the general characteristics of multi-class imbalances and progresses through a discussion on various approaches to handle such complexities. The authors categorize the solutions into decomposition-based methods, which involve breaking down the multi-class problem into several binary problems, and ad-hoc methods specifically tailored for multi-class scenarios. Notably, they also review performance metrics suitable for evaluating models in this context, such as macro and micro averaging techniques, which are crucial for ensuring the models' fairness and effectiveness across all classes (Fernández et al., 2018).

2.4.2 Binary Class Imbalance

Lakshmi and Prasad (2014) focus on binary class imbalances, a common scenario in datasets where one class significantly outnumbers another. This study highlights various strategies to address class imbalance, such as resampling techniques, algorithmic adjustments, and cost-sensitive learning. The authors emphasize the use of Synthetic Minority Over-sampling Technique (SMOTE) combined with ensemble methods like Random Forest and Bagging to enhance the predictive performance of the minority class without compromising the majority class accuracy (Lakshmi and Prasad, 2014).

2.5 Model Considerations

2.5.1 Random Forests

Random Forests, an ensemble of Decision Trees, are particularly noted for their robustness and high accuracy, as they combine predictions from multiple trees to improve generalizability and prevent overfitting. Jha (2019) demonstrate the application of these models in classifying diverse datasets and highlight their strengths in handling both binary and multi-class data, making them highly suitable for customer segmentation and behavior prediction (Jha, Dave, and Madan, 2019).

2.5.2 Boosting Algorithms

Boosting is a sequential technique where each model attempts to correct the errors of its predecessors. The study by Low (2021) focuses on Gradient Boosting combined with hybrid sampling techniques to address class imbalances in predicting commercial vehicle activities. This approach iteratively adjusts the weights of incorrectly classified instances, making the model focus more on difficult cases in subsequent iterations. Boosting is particularly advantageous in scenarios with weak class separations, as it can enhance the decision boundary incrementally to achieve better classification performance(Low, Cheah, and You, 2020).

2.5.3 Application in Digital Gaming and Betting

Both reviewed studies address the challenge of imbalanced datasets, which is common in customer behavior prediction where certain behaviors or customer segments are underrepresented. The flexibility of boosting algorithms to integrate with different data sampling methods allows for tailored approaches in predictive tasks, suitable for dynamic and scalable environments like online gaming platforms where user behaviors constantly evolve. Decision Trees and Random Forests are praised for their robustness against noise and capacity to handle large feature spaces without severe overfitting, making them ideal for complex datasets in digital gaming. The ensemble nature of Random Forests inherently avoids overfitting and provides a more reliable predictive performance.

2.6 Model Tuning

2.6.1 Grid Search, Random Search, and Genetic Algorithms

Liashchynskyi and Liashchynskyi's study offers a comparative analysis of these three hyperparameter optimization techniques in the context of NAS. The research primarily focuses on building convolutional neural networks and evaluates the algorithms based on execution time and model accuracy using the CIFAR-10 dataset(Petro Liashchynskyi and Pavlo Liashchynskyi, 2019).

- **Grid Search**: This method involves an exhaustive searching through a manually specified subset of the hyperparameter space. Its main advantage is thoroughness, as it systematically works through all combinations, ensuring that no option is missed. However, it is computationally expensive and impractical for large datasets or complex models with numerous hyperparameters due to its brute-force nature.

- **Random Search**: Random Search improves over Grid Search by randomly sampling hyperparameters from a defined range. This approach is generally faster and can outperform Grid Search when only a few hyperparameters significantly influence the model's performance. It's particularly effective when used as a preliminary screening tool to narrow down the parameter space.

- **Genetic Algorithm**: This technique uses mechanisms inspired by biological evolution, such as selection, mutation, and crossover, to optimize the hyperparameters. It starts with a randomly generated population of parameter sets and iteratively evolves them towards better solutions. Genetic Algorithms are well-suited for complex optimization problems where the parameter space is vast and not well understood.

2.6.2 Application in Digital Gaming and Betting

Grid Search provides a solid foundation when model complexity and computational resources are manageable. Random Search offers a quicker alternative with reasonable outcomes, and Genetic Algorithms present a robust solution for complex and high-dimensional search spaces, typical in modern NAS applications. These techniques enable the development of highly optimized models that can drive personalized and engaging user experiences in digital platforms.

2.7 Model Evaluation

2.7.1 F1 Score, AUC-ROC, and Accuracy

Zhu, Zeng, and Wang (2010) provide a comprehensive examination of sensitivity, specificity, accuracy, and ROC analysis within medical diagnostics, which are directly applicable to predictive modeling in other fields(W. Zhu, Zeng, Wang, et al., 2010). They emphasize the importance of these metrics in evaluating the truthfulness of a model's predictions:

- **Sensitivity (Recall)** and **Specificity** offer measures of a model's ability to correctly predict positives and negatives, respectively.

- **Accuracy** provides a holistic view of the overall correctness of the model across all classes.

- **ROC Analysis** and the **AUC (Area Under Curve)** measure are highlighted as robust tools for assessing the trade-off between true positive rates and false positive rates, providing a single scalar value to compare different models.

2.7.2 KS Statistic, Gain, and Lift

Gu, Zhu, and Cai (2009) discuss these metrics in the context of evaluating classification performance on imbalanced datasets. Their focus on KS statistic, Gain, and Lift charts provides insights into the distribution of probabilities assigned to actual positive and negative classes and how well a model can distinguish between these two:

- **KS Statistic** measures the maximum distance between the cumulative true positive and false positive rates, helping in identifying the model's discriminatory power.
- **Gain and Lift Charts** are crucial for visualizing the effectiveness of a classification model at different threshold levels, showing how much better a model is compared to random guessing.

The combination of the aforementioned metrics offers a holistic framework for model evaluation, catering to both the nuances of imbalanced datasets and the general requirements for accurate predictive modeling(Gu, L. Zhu, and Cai, 2009).

Chapter 3

Methodology

3.1 Justification for Ensemble Techniques

3.1.1 Ensemble methods vs Single predictor models

This study emphasizes the superiority of ensemble methods over single predictor models in churn prediction within the online gambling sector. The research demonstrates that ensemble techniques, including Random Forests and generalized additive model ensembles (GAMens), provide more accurate and robust predictions compared to single models like CART decision trees or generalized additive models (GAMs). Ensemble methods benefit from aggregating multiple models, which helps in reducing the variance and improving the prediction accuracy, especially crucial in predicting customer behaviors that are not linearly separable (Coussement and De Bock, 2013).

3.1.2 Latest Churn Models

The research Grönros and Janér, 2018 explores various machine learning techniques for churn prediction in the iGaming industry, underscoring the effectiveness of Random Forest and logistic regression. They conclude that while traditional models like logistic regression are valuable, the ensemble models, particularly those that combine multiple predictors, consistently outperform single predictor models. This supports the integration of multiple learning strategies to enhance predictive performance in customer churn applications(Grönros and Janér, 2018).

3.1.3 Large Dataset Models

Sun's research further solidifies the argument for using sophisticated machine learning techniques like LightGBM, a type of gradient boosting framework that is part of the ensemble model family. LightGBM is specifically praised for its efficiency and performance in handling large datasets with many features, typical of the online gambling industry. The study details how LightGBM optimizes both speed and accuracy in predictions by utilizing a histogram-based algorithm that deals effectively with continuous and categorical data (Sun, 2024).

The selection of ensemble learning methods for predicting customer churn in the online gambling industry is justified by their enhanced ability to manage large and complex datasets, improve prediction accuracy through model aggregation, and offer deeper insights into the predictive importance of various customer behaviors. These methods align well with the industry's needs for robust analytical approaches to sustain competitive advantage through effective customer relationship management.

3.2 Bagging and Boosting

3.2.1 Bagging (Bootstrap Aggregating)

Bagging involves generating multiple versions of a predictor by creating random subsets of the training dataset, with replacement. Each subset is used to train a decision tree. The final prediction is typically made by averaging the predictions (for regression) or by majority voting (for classification). Random Forest employs this method where each tree in the forest votes for a class, and the class receiving the most votes becomes the model's prediction (*RandomForestClassifier* 2024).

3.2.2 Boosting

Unlike bagging, which trains models independently, boosting trains models sequentially with each new model focusing on correcting the errors made by the previous ones. The predictions are combined through a weighted majority vote (or sum) to produce the final prediction. Algorithms like LightGBM utilize this technique.

3.3 Random Forest

Random Forest specifically operates by constructing a multitude of decision trees at training time and outputting the class that is the mode of the classes (classification) or mean prediction (regression) of the individual trees (Breiman, 2017). Random Forest is particularly favored for its performance in diverse scenarios including, but not limited to, handling high-dimensional spaces and maintaining accuracy even when data has noticeable class imbalances—making it less likely to overfit compared to many other algorithms. This is accomplished through the randomness injected in both the selection of data points and features used to build individual trees within the forest. Each tree in a Random Forest works on a random subset of features to decide splits, reducing the variance without increasing the bias, which in turn gives Random Forest its characteristic of being a low bias and low variance model (Breiman, 2017).

3.3.1 Mathematical Representation

The ensemble prediction $ensemble(x)$ can be expressed as:

$$ensemble(x) = sign\left(\sum_{i=1}^{Q} w_i \times h_i(x)\right)$$

where $sign$ denotes the sign function, $h_i(x)$ represents the hypothesis of the i^{th} classifier, and w_i denotes the weight associated with that classifier, indicating its reliability. The ensemble method ensures that the collective decision reflects a balance between the predictions of all the individual trees, accounting for the possibility that some trees might be more accurate in certain areas than others.

3.3.2 Cross-validation in Random Forest

To maximize model reliability and accuracy, it is essential to ensure that the model performs well not just on the training data but also on unseen data. This is where cross-validation techniques come into play.

3.3.3 K-fold Cross-validation

K-fold Cross-validation enhances model validation by dividing the data into 'K' subsets. Each subset is used once as a test set while the other $K - 1$ subsets are used for training. This process rotates until each subset has been used for testing once. The average performance across all K trials is then used as the overall model performance estimate, helping in the identification and correction of any overfitting or underfitting issues (Brownlee, 2020).

3.3.4 Hyperparameter Tuning

3.3.5 Key Hyperparameters

- **n_estimators**: Number of trees in the forest.
- **max_depth**: Maximum number of levels in each decision tree.
- **min_samples_split**: Minimum number of data points placed in a node before the node is split.
- **min_samples_leaf**: Minimum number of data points allowed in a leaf node.

- **max_features**: Maximum number of features considered for splitting a node (Hancock and Khoshgoftaar, 2021).

3.3.6 Optimization Techniques: Randomized Search and Grid Search

Hyperparameter tuning can be approached through methods like Randomized Search and Grid Search, often combined with K-fold Cross-validation.

3.3.6.1 Randomized Search

This technique samples hyperparameter values from a distribution at random for a fixed number of iterations. This approach allows for a broad and random exploration of the hyperparameter space, providing a practical solution for narrowing down the range for more precise tuning methods like Grid Search Petro Liashchynskyi and Pavlo Liashchynskyi, 2019.

3.3.6.2 Grid Search

Following Randomized Search, Grid Search meticulously searches through a specified subset of hyperparameters, previously narrowed down by Randomized Search. It evaluates model performance for each combination of the hyperparameter values provided, ensuring that the optimal set of parameters is identified. This method is usually more computationally expensive due to its exhaustive nature but tends to find the most effective combination when paired with Cross-validation Petro Liashchynskyi and Pavlo Liashchynskyi, 2019.

3.4 LightGBM

LightGBM stands for Light Gradient Boosting Machine. LightGBM is a gradient boosting framework that uses tree-based learning algorithms. However, LightGBM is designed to be more efficient in terms of memory and speed, especially with large datasets. It achieves higher efficiency and lower memory usage by using a novel technique of Gradient-based One-Side Sampling (GOSS) and Exclusive Feature Bundling (EFB), which allows it to handle large amounts of data with significant acceleration and reduced computational resources Ke et al., 2017.

3.4.1 Gradient Boosting vs. Other Ensemble Methods

Gradient boosting is a method where new models are created that predict the residuals or errors of prior models and then added together to make the final prediction. It is an optimization of boosting techniques and stands in contrast to Random Forest, which is a bagging model. Boosting involves adjusting the weight of an incorrectly predicted instance so that the next decision tree prioritizes it more than the correctly predicted instances.

3.5 Key Innovations of LightGBM

3.5.1 Gradient-based One-Side Sampling (GOSS)

GOSS keeps all the instances with large gradients (i.e., poorly predicted instances) and performs random sampling on the instances with small gradients (i.e., well-predicted instances). This approach ensures that the training is more focused on the harder cases, which enhances the model's accuracy and speeds up the computation without significant loss of data integrity (Hancock and Khoshgoftaar, 2021).

3.5.2 Exclusive Feature Bundling (EFB)

EFB reduces the number of features in a dataset by bundling mutually exclusive features, i.e., features that rarely take nonzero values simultaneously. This reduction not only decreases the computational burden but also helps in handling high-dimensional data more efficiently, making LightGBM particularly useful in scenarios where the feature space is large but sparse (Hancock and Khoshgoftaar, 2021).

3.6 Model Fitting and Prediction

In LightGBM, trees are built vertically while other gradient boosting algorithms build trees level-wise. This means that LightGBM grows trees leaf-wise rather than level-wise, choosing the leaf with the maximum delta loss to grow. When this approach is compared with level-wise growth, it can reduce more loss, resulting in better model accuracy.

3.6.1 Mathematical Representation

For fitting a LightGBM model, the prediction score of each individual decision tree is considered:

$$y_i = f(x_i);\ f \in \mathcal{F}$$

where K is the total number of trees, and f represents the hypothesis of trees in $\mathcal{F}$, the set of all possible trees. The objective function is formulated as:

$$obj(\theta) = l(y, y_i) + \Omega(f)$$

where $l(y, y_i)$ represents the loss function and $\Omega(f)$ is the regularization term.

3.6.2 Additive Strategy

LightGBM applies an additive strategy, which involves minimizing the loss adjusted by the previous tree:

$$y_i^{(t)} = f_t(x) = y_i^{(t-1)} + f_t(x)$$

This sequential strategy enables the model to learn gradually, adjusting for the residuals of previous trees, thus refining the predictions iteratively.

3.7 Hyperparameter Tuning

To prevent overfitting and ensure robust performance, hyperparameters in LightGBM are meticulously tuned. Some of the major hyperparameters include:

- **Learning_rate** (or eta): Controls the step size shrinkage used to prevent overfitting. Smaller values make the learning slower and more robust.
- **N_estimators**: Determines the number of boosting rounds or trees to build.
- **Max_depth**: Controls the maximum depth of each tree. Deeper trees can model more complex patterns but can also lead to overfitting.
- **Subsample**: Specifies the fraction of samples to use for fitting the individual base learners.
- **Colsample_bytree**: Indicates the fraction of features to use for each tree.
- **Colsample_bylevel**: Controls the subsample ratio of columns for each level of each tree.

The tuning of these parameters is often performed using automated techniques like Grid Search and Randomized Search, in conjunction with K-fold cross-validation. These methods help in identifying the best parameter combination that yields the most generalizable model(Ke et al., 2017).

3.8 Evaluation

In this project, three primary machine learning algorithms are employed: Decision Trees, Random Forest, and LightGBM. To identify the most effective model, a systematic evaluation is conducted across three distinct levels. Each level of evaluation aims to progressively filter and refine the choice of the optimal model, ensuring that it meets specific criteria for accuracy, reliability, and applicability.

3.8.1 Evaluation Strategy

The evaluation process is tiered as follows:

- **Level 1 Evaluation**: This initial phase involves assessing various configurations of a single algorithm, including default settings, feature selection modifications, and models optimized through hyperparameter tuning.
- **Level 2 Evaluation**: The second phase compares the best-performing models from each algorithm to determine the superior approach.
- **Level 3 Evaluation**: The final evaluation compares the newly selected model against existing benchmarks to confirm improvements.

3.8.2 Level 1 Evaluation

3.8.2.1 Model Assessment Techniques

At this stage, multiple candidate models for each algorithm are evaluated primarily using the confusion matrix and its derived metrics. The confusion matrix is particularly useful in binary classification tasks as it provides a clear breakdown of true positives (TP), true negatives (TN), false positives (FP), and false negatives (FN), which are critical for understanding model performance in categorizing classes.

3.8.2.2 Derived Metrics

From the confusion matrix, several key performance metrics are calculated:

- **Accuracy**:
$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN}$$
- **Precision**:
$$Precision = \frac{TP}{TP + FP}$$
- **Recall (Sensitivity)**:
$$Recall = \frac{TP}{TP + FN}$$
- **Specificity**:
$$Specificity = \frac{TN}{TN + FP}$$

Additionally, the ROC (Receiver Operating Characteristic) curve and the AUC (Area Under the Curve) score are used for models optimized during hyperparameter tuning. The AUC score, in particular, indicates the model's ability to distinguish between classes—the higher the AUC, the better the model is at prediction W. Zhu, Zeng, Wang, et al., 2010.

3.8.3 Level 2 Evaluation

3.8.3.1 Advanced Comparative Metrics

In the second level of evaluation, more nuanced metrics such as the Ks-score, gain charts, and lift charts are employed to further discern the efficacy of each model beyond basic confusion matrix metrics. This is crucial in situations of class imbalance, where traditional metrics might not fully reveal the model's predictive strength.

3.8.3.2 Gains Table and Ks-Score

The Gains Table is a methodological approach where probabilities of positive and negative classes are segmented into deciles. This distribution helps in identifying optimal cut-points for classifying probabilities as positive or negative, enhancing model precision in predicting classes. The Ks-score, which measures the maximum divergence between the cumulative distribution functions of the positive and negative classes, is a critical indicator in this analysis. A higher Ks-score suggests a better model at distinguishing between classes.

3.8.3.3 Visualization and Decision Metrics

- **Gains Chart**: This chart plots the cumulative percentage of true positives against the total expected positives and is used to compare different models directly.
- **Lift Chart**: The lift chart illustrates how much better a model is at predicting positive classes compared to a random guess. It is an essential metric for gauging the practical effectiveness of a model in operational settings.

3.8.4 Level 3 Evaluation

The final evaluation phase involves a comparative analysis between the newly developed model and an existing baseline. This comparison is vital to ensure that the new model provides a tangible improvement over previous methodology.

Chapter 4

Experimental Results

4.1 The Initial Data

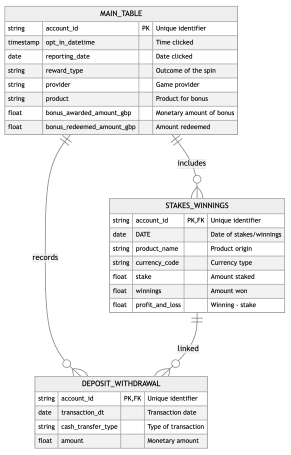
Flutter Entertainment plc operates a significant gaming and betting infrastructure, tracking and managing a vast amount of data daily. They collect information across multiple tables which detail customer interactions, transactions, and behavior patterns. The company relies on the integration of these tables to produce insights and decisions related to customer engagement, marketing strategies, and product development.

4.1.1 Data Collection and Structure

Flutter Entertainment utilizes a sophisticated database system to manage over 300,000 data entries daily. These data entries are essential for understanding customer habits, preferences, and the effectiveness of various promotions and games. The data is retrieved from the company's Redshift server, and the sampling process involves extracting information from three primary tables:

1. Bonus Information Table
2. Stakes and Winnings Table
3. Deposit and Withdrawal Information Table

These tables are designed to capture comprehensive data points that range from transaction details to customer betting outcomes, which can then be used to build predictive models and guide business strategy Fig 4.1.



MAIN_TABLE			
string	account_id	PK	Unique identifier
timestamp	opt_in_datetime		Time clicked
date	reporting_date		Date clicked
string	reward_type		Outcome of the spin
string	provider		Game provider
string	product		Product for bonus
float	bonus_awarded_amount_gbp		Monetary amount of bonus
float	bonus_redeemed_amount_gbp		Amount redeemed

STAKES_WINNINGS			
string	account_id	PK,FK	Unique identifier
date	DATE		Date of stakes/winnings
string	product_name		Product origin
string	currency_code		Currency type
float	stake		Amount staked
float	winnings		Amount won
float	profit_and_loss		Winning - stake

DEPOSIT_WITHDRAWAL			
string	account_id	PK,FK	Unique identifier
date	transaction_dt		Transaction date
string	cash_transfer_type		Type of transaction
float	amount		Monetary amount

Figure 4.1: Database Schema

4.1.2 Detailed Table Schema Analysis

4.1.2.1 Bonus Information Table

Objective: This table is pivotal for tracking promotional engagements, specifically through spins on the "Wonder Wheel." It helps Flutter Entertainment understand the effectiveness of different promotions and their impact on customer behavior.

Fields:

- **account_id**: The unique identifier for customers, essential for tracking individual activities without redundancy.
- **opt_in_datetime**: Timestamp indicating when a customer opts for a particular promotion by spinning the wheel.
- **reporting_date**: The exact date the customer participated in the promotion, useful for temporal analysis.
- **reward_type**: Categorizes the outcome of the spin, such as losing spins (VOID), free spins on slots (FREE_SPINS), or direct cash rewards (CASH).
- **provider**: Indicates the game provider when free spins are awarded, adding a layer of detail about which providers are most engaging for customers.
- **product**: Specifies the product category, like Gaming or Sports, where the bonus is applicable, aiding in targeted marketing strategies.
- **bonus_awarded_amount_gbp**: The monetary value of the bonus awarded, key for financial tracking and analysis.

- **bonus_redeemed_amount_gbp**: Shows how much of the awarded bonus was actually used by the customer, a direct indicator of promotion success.

Utility: This table serves as a foundational element for understanding promotional dynamics and customer engagement levels. By analyzing the bonus redemption rates and types, Flutter Entertainment can tailor future promotions more effectively.

4.1.2.2 Stakes and Winnings Table

Objective: To provide a daily snapshot of customer betting activities, revealing patterns in stakes, winnings, and overall profitability.

Fields:

- **DATE**: Records the date of betting activities, allowing for daily performance tracking.
- **account_id**: Links customer betting data with their identity for individualized tracking.
- **product_name**: Identifies the betting product, useful for segmenting data by product type.
- **currency_code**: The currency used for the transactions, necessary for financial consistency and conversions.
- **stake**: The amount staked by the customer on that particular day.
- **winnings**: The total amount won by the customer.
- **profit_and_loss**: Calculated as winnings minus stakes; a critical metric for understanding customer profitability and the financial health of the products.

Utility: This table is crucial for measuring the immediate financial engagement of customers with Flutter Entertainment's products. It aids in short-term financial planning and can be used to refine risk management strategies.

4.1.2.3 Deposit and Withdrawal Information Table

Objective: Tracks all customer financial transactions related to their accounts, including deposits and withdrawals, providing a clear picture of cash flow and customer liquidity.

Fields:

- **account_id**: Ensures transactions are accurately linked to the correct customer.
- **transaction_dt**: Date of the transaction, essential for chronological tracking.

- **cash_transfer_type**: Distinguishes between deposit and withdrawal transactions.
- **amount**: The monetary value of the transaction, vital for managing financial ledgers and understanding customer spending patterns.

Utility: This table is indispensable for financial oversight and operational management. It offers insights into the cash flow trends of customers, which can influence credit policies, promotional strategies, and overall financial health assessments.

4.1.3 Analysis and Dataset Creation

The following analysis elaborates on the process of creating training and validation datasets for a gaming and betting platform, focusing on customer interactions with bonus offerings and subsequent financial transactions. This entails building datasets for binary and multiclass classification tasks by querying a Redshift database and structuring the data accordingly. The end goal is to predict customer behaviors, specifically relating to deposits post-bonus interactions, classified into different categories based on the nature and result of those interactions.

4.2 Data Preparation and Table Creation

4.2.1 Initial Data Retrieval and Processing

The process begins by querying three principal tables from the company's Redshift server: *omni_bonus.pp_cpp_vw_wonder_wheel_summary*, *omni_payment.vw_payments_cash*, and *omni_accounting.vw_*
The data retrieved includes information about customer bonuses, transaction dates, transaction types, amounts, and detailed accounting of customer stakes, winnings, and losses.

4.2.2 Creation of Intermediate Tables

- **Recent Bonus Table**: Captures all details about the bonuses awarded to customers, including the type of bonus, the amount awarded and redeemed, and computes lagged features to capture previous bonuses and their types.
- **Transaction Data**: Filters to include only deposit transactions, crucial for linking bonus interactions to financial responses.
- **Daily Random Sample**: Implements random sampling to ensure diversity and manageability of the dataset, with limits set to 1,000 entries per day for training data (from January 1 to July 31) and 5,000 entries per day for validation data (from August 1 to August 31).

4.2.3 Class Assignment and Dataset Finalization

- **Binary Class Assignment**: For the binary classification task, a deposit made within 24 hours of receiving a bonus is labeled as 1, and no deposit as 0.
- **Multiclass Assignment**: Extends the binary classification by differentiating between:
  - 1: Bonus awarded and deposit made within 24 hours.
  - 0: Bonus awarded but no deposit made.
  - 2: No bonus awarded and no deposit made.
  - 3: No bonus awarded but a deposit made within 24 hours.

4.2.4 Aggregation of Betting Data

Seven-day aggregates of stakes, winnings, and profits/losses are computed to provide a recent snapshot of betting behaviours, which enrich the feature set for predictive modelling.

4.2.5 Final Tables for Binary and Multiclass Classification

The final tables for both binary and multiclass classification (*datascience_shared.dipesh_labels_final* for training and *datascience_shared.dipesh_labels_finalmulti* for multiclass) are structured to include:

- **Key Identifiers**: *account_id*, *account_sk* (a secure version of the account identifier).
- **Temporal Features**: *opt_in_datetime*, *reporting_date.*
- **Bonus and Transaction Details**: *bonus_awarded_amount_gbp*, *bonus_redeemed_amount_gbp*, *previous_bonus_awarded_amount_gbp*, *previous_reward_type.*
- **Aggregated Betting Data**: *stakes_7d*, *winning_7d*, *profit_loss_7d.*
- **Financial Transactions**: Aggregated sums and counts of deposits and withdrawals over the previous seven days.
- **Class Labels**: As defined by the binary or multiclass criteria.

4.2.6 Schema of the Final Tables

Core Attributes:

- **Account Information**: Account identifiers and timestamps of activities provide the linkage across different datasets and temporal dynamics.

- **Bonus Interaction**: Details of the bonus type, amounts awarded and redeemed, and historical bonus data offer insights into the promotional engagement of customers.
- **Financial Transactions**: Detailed transaction data including the type, amount, and timing of deposits and withdrawals, enriched with seven-day aggregates, support an understanding of customer financial behaviour in response to promotions.

The schema of the final table for both binary and multiclass classification tasks in a gaming and betting context is intricately designed to capture a comprehensive snapshot of customer interactions related to bonus and betting activities, as well as their subsequent financial behaviours Fig 4.2. At its core, the table includes several key attributes that uniquely identify each record and link it to specific customer activities. These include *account_id* and *account_sk*, the latter providing a secure identifier for the customer, which is crucial for maintaining privacy and data integrity. Temporal attributes such as *opt_in_datetime* and *re-*

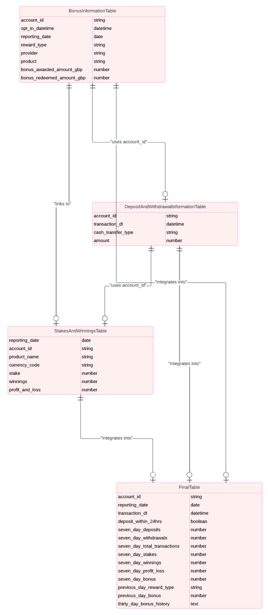


Figure 4.2: Database Schema

porting_date are essential for contextualizing each record within a specific timeframe, offering insights into when customers opt-in for bonuses and how their behaviors evolve over time. Bonus-related details are extensively covered through fields like *bonus_awarded_amount_gbp*, *bonus_redeemed_amount_gbp*, *previous_bonus_awarded_amount_gbp*, and *previous_reward_type*. These fields not only provide data on the current bonus interaction but also link back to previous interactions, helping in understanding patterns or trends in customer responses to promotional activities.

The table further incorporates comprehensive transactional data, including the type and amount of financial transactions (*transaction_dt*, *cash_transfer_type*, *amount*), which are pivotal in analysing the financial behavior of customers post-bonus engagement Fig 4.3. To deepen the analytical value, the table includes aggregated betting data over a specified period (typically the past seven days) such as *stakes_7d*, *winning_7d*, and *profit_loss_7d*. These aggregates offer a quick snapshot of a customer's betting activity, providing a measure of their engagement and profitability within the platform. Class labels, which are central to the use

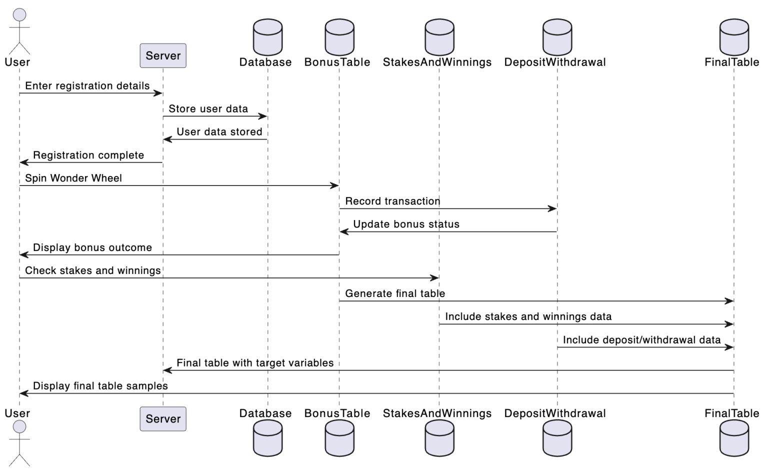


Figure 4.3: Database Schema

of this table in predictive modelling tasks, vary based on the classification type—binary or multiclass. In a binary schema, the label indicates whether a deposit was made within 24 hours after receiving a bonus, whereas, in the multiclass schema, the labels are more granular, indicating different scenarios of bonus awarded and subsequent customer actions (deposit made/not made with/without bonus awarded).

4.3 Exploratory Data Analysis

4.3.1 Binary Target Variable Analysis

One of the most important analyses was the distribution of the target variable, *deposit_within_24hrs*, in relation to the *reward_type*. From the reward-type distribution, it was evident that customers are more likely to deposit within 24 hours when they receive free spins as a reward, compared to other reward types like cash or wallet awarding. For instance, cash rewards exhibit a 50/50 probability of deposit, while wallet awarding ranks third. In contrast, when the reward type is "VOID," there is little incentive for customers to make deposits. This observation suggests that the *reward_type* is a key feature influencing the likelihood of a deposit and should be retained in the model Fig 4.4.

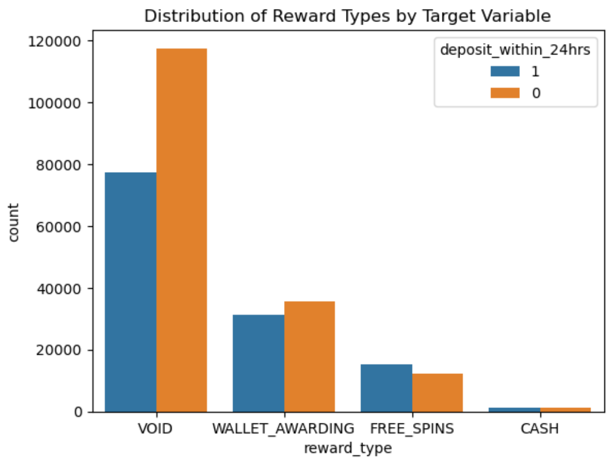


Figure 4.4: Reward Type

4.3.2 Temporal Data Analysis

Hourly Distribution: The analysis of hourly data reveals clear patterns in user behaviour. Most deposits (*deposit_7d*) occur during late evening hours around midnight and again at 23:00 Fig 4.5. These peak hours suggest that customers are more active and willing to make deposits at specific times of the day. Additionally, the hourly analysis of other features like *profit_loss_7d* and *transaction_7d* shows that users tend to experience losses throughout the day, with occasional peaks in transaction activity. Based on these insights, temporal features such as *opt_in_hour* (derived from the *opt_in_datetime*) should be included in the model as they capture the time-sensitive nature of customer deposits. **Weekly Distribution**: The weekly distribution of deposit activity showed consistency across the days, with slight dips midweek (Wednesday) and on Saturday. The most active days for deposits were Mondays and Sundays, suggesting a start-of-week and end-of-week pattern in user engagement Fig 4.6. Including *opt_in_day_of_week* as a feature will allow the model to capture these weekly variations in customer behaviour, adding more predictive power. **Monthly Distribution**: The monthly deposit distribution showed a decline in activity as we moved closer to July 2024. This decline in deposit activity could reflect seasonal changes or other external factors affecting user engagement. Additionally, profit and loss values remained relatively stable, while transaction values saw some variability over the months Fig 4.7. As seen in the temporal analysis, including *reporting_date* as a feature, alongside monthly variations, could help the model capture broader patterns in user behaviour over time.

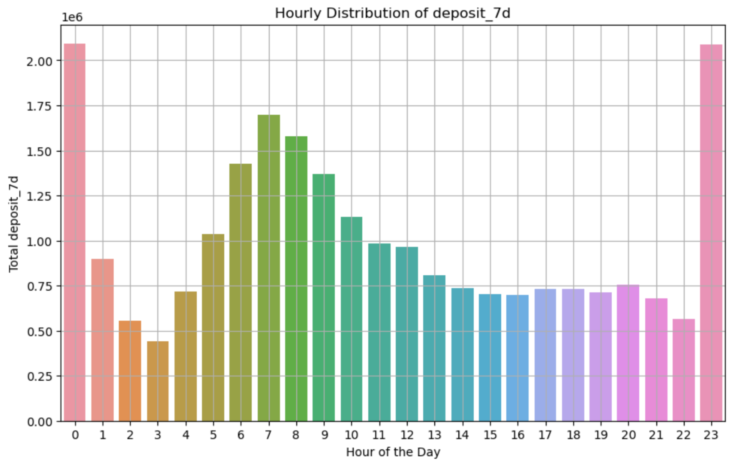


Figure 4.5: Deposit hourly Distribution

4.3.3 Relationship Between Features

Stakes vs. Winnings: A scatter plot of *stakes_7d* versus *winning_7d* revealed a strong linear relationship between the two variables Fig 4.8. As expected, users who stake larger amounts tend to win more frequently. This correlation also extends to deposit behaviour: users with the highest stakes and winnings are more likely to make deposits within 24 hours. This insight underscores the importance of including *stakes_7d* and *winning_7d* in the model, as they are strong indicators of user engagement and the likelihood of deposit behaviour.

4.3.4 Feature Selection for Modelling

Based on the EDA, several features stand out as being crucial for predictive modelling:

- **Time-based Features**:
  - *opt_in_hour*: The hourly analysis showed distinct spikes in deposit activity at midnight and late evening. Including this feature will help capture time-based behavior patterns in user engagement.
  - *opt_in_day_of_week*: Weekly trends suggest that users are more likely to deposit on Mondays and Sundays, making the day of the week an important predictor.
  - *reporting_date*: As monthly patterns revealed significant changes in deposit behaviour, particularly in July, including the reporting date will allow the model to capture long-term trends.
- **Transaction-based Features**:

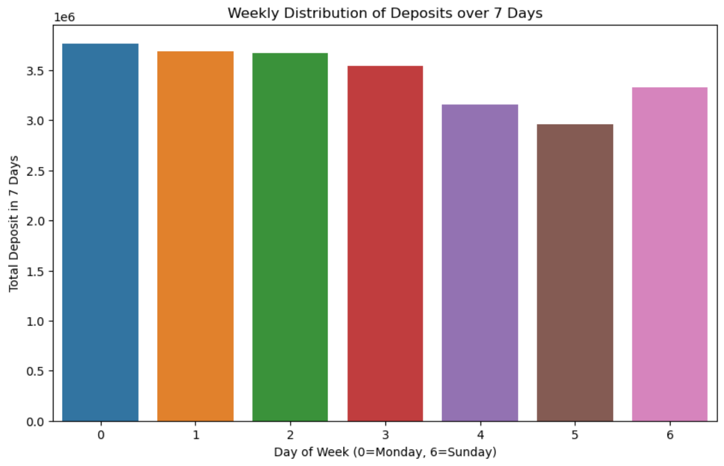


Figure 4.6: Weekly Distribution

- *stakes_7d* and *winning_7d*: These features measure user engagement and are positively correlated with deposit behavior. High-stakes users, especially those who win frequently, are more likely to continue interacting with the platform by making deposits.
- *transaction_7d* and *transaction_7d_count*: Total transactions and their frequency serve as proxies for user engagement. These features help capture how frequently users interact with the platform.

- **Bonus-related Features**:
  - *bonus_awarded_amount_gbp* and *bonus_redeemed_amount_gbp*: These features measure the bonuses awarded and redeemed by the user, potentially influencing deposit decisions. For example, users who receive high bonuses are more likely to deposit within 24 hours.
  - *bonus_7d* and *bonus_30d*: These features capture the user's bonus history, which is essential for understanding how previous bonuses impact current behaviour.

- **Profit and Loss**:
  - *profit_loss_7d*: This feature captures the user's profitability. Users who experience losses may be less likely to make deposits, while those who experience profits may be encouraged to reinvest their winnings. Including this feature will help the model understand the financial status of users and how it affects their future actions.

- **Target Variable**:

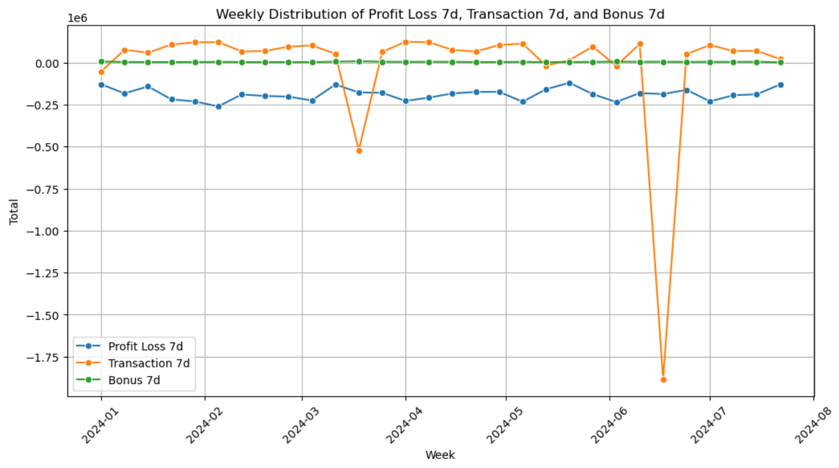


Figure 4.7: Monthly Distribution

– *deposit_within_24hrs*: The target variable, indicating whether the user deposits within 24 hours, is the focus of the model. All other features will be used to predict this outcome.

4.3.5 Handling Missing Values

In the cleaning process, several columns with a high percentage of missing values were removed (*provider*, *product*). For the remaining features, missing numerical values were filled with zeros. For instance, if no transactions occurred during a specific period, variables like *transaction_7d* and *transaction_7d_count* were set to zero. Categorical features such as *reward_type* were filled with "VOID," representing users who did not receive a reward.

4.3.6 Insights and Justification for Model

Temporal Variability: The temporal analysis shows that user behaviour is heavily influenced by the time of day, day of the week, and month. By incorporating time-based features like *opt_in_hour*, *opt_in_day_of_week*, and *reporting_date*, the model will be better equipped to capture these behavioural patterns, resulting in more accurate predictions.

User Engagement: Transaction-based features like *stakes_7d*, *winning_7d*, and *transaction_7d* showed a strong correlation with deposit behaviour. Users who stake more and win more are likely to make deposits, suggesting that these features will significantly improve the model's ability to predict deposit behaviour.

Bonus Influence: The analysis of bonus-related features suggests that bonuses play an important role in influencing user behaviour. Users who receive higher bonuses are more likely to deposit within 24 hours, particularly when the reward type is FREE_SPINS. Therefore,

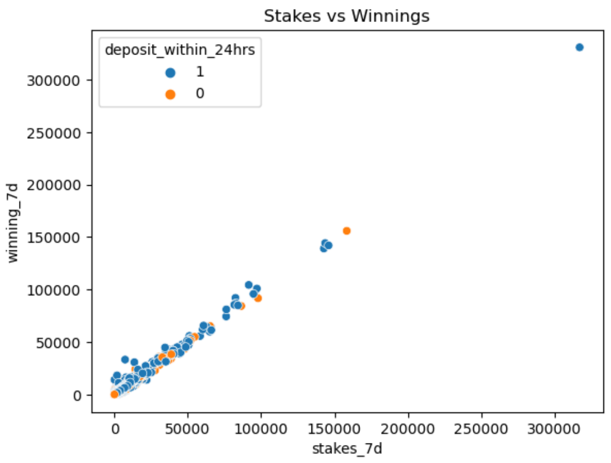


Figure 4.8: Variable Relation

including bonus-related features like *bonus_awarded_amount_gbp* and *bonus_7d* will help the model capture these effects.

Variable Selection: The EDA process revealed significant insights into user behaviour and helped identify key features for predictive modelling. Temporal features like *opt_in_hour* and *opt_in_day_of_week*, profit loss-related features, transaction, and bonus-related features all play crucial roles in predicting whether a user will deposit within 24 hours. By incorporating these features into the model, along with time-based patterns and user engagement metrics, we can create a robust predictive model that captures user behavior more accurately.

4.4 Binary vs Multi-class Model

In predictive modeling, the selection of the appropriate target variable and classification type is a crucial decision that significantly impacts the model's performance. In this case, the problem revolves around predicting customer deposits within 24 hours based on several features such as bonuses, transactions, stakes, and winnings. The primary objective is to decide whether to approach this problem as a binary classification (i.e., whether a customer will deposit or not) or a multi-class classification (i.e., incorporating scenarios where a customer might deposit depending on whether they receive a bonus).

To determine which approach provides better predictive performance, we conducted a comprehensive evaluation of both the binary and multi-class models using random sampling on the training data and validation testing on unseen data. Metrics such as confusion matrix, precision, recall, F1-score, and ROC-AUC were calculated to assess the strengths and weaknesses of each approach. The following sections outline the observations, results, and justifications for selecting the optimal classification dataset for this task.

4.4.1 Binary Classification

The binary classification model categorizes the target variable as a simple dichotomy: either the customer deposits within 24 hours (1) or they do not deposit (0). This approach narrows down the focus to a clear-cut prediction of user behavior based on their engagement with the platform.

	Predicted No Deposit	Predicted Deposit
Actual No Deposit (0)	86,614	26,975
Actual Deposit (1)	7,093	24,318

Table 4.1: Confusion Matrix for Binary Classification

The confusion matrix indicates that the model performed fairly well in predicting both classes. The true negatives (86,614) and true positives (24,318) suggest that the model is correctly identifying a large proportion of users who do not deposit and a reasonable number of users who do deposit. However, the model does exhibit a high number of false positives (26,975) and false negatives (7,093) Table 4.1.

Class	**Precision**	**Recall**	**F1-Score**	**Support**
Class 0	0.92	0.76	0.84	113589
Class 1	0.47	0.77	0.59	31411

Table 4.2: Classification Report (Validation - LightGBM)

The precision and recall for class 0 (no deposit) are significantly higher than those for class 1 (deposit). However, the recall for class 1 is higher than its precision, indicating that the model is more likely to predict deposits correctly when they occur (high recall), but it also produces many false positives (low precision). The weighted F1-score for the overall model stands at 0.78, while the ROC-AUC score of 0.8427 suggests that the model is relatively good at distinguishing between deposits and non-deposits Table 4.2.

4.4.1.1 Conclusion on Binary Classification Performance

The binary classification model demonstrates good performance overall, particularly with a strong ability to correctly identify users who do not deposit. However, the imbalance between precision and recall for deposits highlights potential issues in overestimating deposits, which could lead to unnecessary promotions or marketing efforts. The strong ROC-AUC score indicates that the model is still reliable in distinguishing the two classes.

4.4.2 Multi-class Classification

In the multi-class classification approach, the target variable is split into four categories based on the combination of bonuses and deposits. These classes are:

- 0: No bonus given, no deposit
- 1: No bonus given, deposit made
- 2: Bonus given, no deposit
- 3: Bonus given, deposit made

The multi-class approach adds more granularity to the prediction by incorporating information about the effect of bonuses on deposit behavior Table 4.3.

	Class 0	Class 1	Class 2	Class 3
Class 0 (No Bonus, No Deposit)	4,239	1,896	14,243	2,783
Class 1 (No Bonus, Deposit)	1,816	8,328	4,323	10,274
Class 2 (Bonus, No Deposit)	8,096	2,801	43,184	4,780
Class 3 (Bonus, Deposit)	2,547	10,114	7,261	15,630

Table 4.3: Confusion Matrix for Multi-class Classification

The confusion matrix for multi-class classification indicates mixed results. The model shows a high degree of confusion between classes 0 (no bonus, no deposit) and 2 (bonus, no deposit), as well as between classes 1 (no bonus, deposit) and 3 (bonus, deposit). This confusion suggests that while the model can predict the bonus-deposit interaction to some degree, the separation between these classes is not strong.

Class	**Precision**	**Recall**	**F1-Score**	**Support**
Class 0	0.25	0.18	0.21	23161
Class 1	0.36	0.34	0.35	24741
Class 2	0.63	0.73	0.68	58861
Class 3	0.47	0.44	0.45	35552

Table 4.4: Classification Report (Validation)

The model achieves its best performance in predicting class 2 (bonus, no deposit), with relatively high precision and recall. However, performance for other classes, especially class 0 (no bonus, no deposit) and class 1 (no bonus, deposit), is significantly lower. The weighted F1-score of 0.49 and the ROC-AUC score of 0.7195 indicate that the multi-class model is less effective than the binary model at distinguishing between classes Table 4.4.

4.4.2.1 Conclusion on Multi-class Classification Performance

The multi-class classification model adds more complexity to the problem by attempting to predict both bonus and deposit behavior. While this additional granularity could provide more insights into user engagement, the performance of the model is notably weaker compared to the binary classification approach. The model struggles to differentiate between the four classes, particularly in distinguishing between no-bonus scenarios and bonus scenarios. This leads to a significant drop in both precision and recall for several classes, making the model less reliable overall.

4.4.3 Observations and Justification for Choosing Binary Classification

Based on the results of the binary and multi-class classification models, the binary classification approach emerges as the better option for this task. The following observations and justifications support this decision:

- **Better Performance Metrics**: The binary model outperforms the multi-class model across almost all metrics. With an overall accuracy of 0.77, a weighted F1-score of 0.78, and a ROC-AUC of 0.8427, the binary model demonstrates better predictive power and reliability compared to the multi-class model's accuracy of 0.50 and ROC-AUC of 0.7195.
- **Simplified Problem Definition**: The binary classification problem simplifies the task by focusing solely on whether a deposit is made, irrespective of the bonus. The multi-class model introduces more confusion without significantly improving accuracy.
- **Reduced Class Imbalance Issues**: The binary model benefits from a more balanced class distribution. The multi-class approach faces a higher degree of class imbalance, leading to lower precision and recall for the minority classes.
- **Simpler Interpretability and Actionability**: A binary classification model is easier to interpret and act upon for business purposes. It allows for more straightforward marketing and customer retention strategies.
- **Targeting the Primary Goal**: The primary goal is to forecast customer deposits within 24 hours. A binary outcome (deposit or no deposit) stays aligned with this goal and simplifies the predictive modeling task.

4.5 Random Sampling vs. Statistical Sampling

4.5.1 Random Sampling

Random sampling is a straightforward technique that assumes each observation in the dataset has an equal chance of being selected. In many cases, this technique is used to ensure that the model receives a broad and diverse set of data points during training. However, random sampling can suffer from issues such as class imbalance, which can disproportionately affect the performance of the model, particularly when dealing with minority classes.

In this analysis, random sampling was applied to the dataset, and the model was trained and evaluated based on the randomly sampled data. The results show a significant imbalance in performance between the majority class (no deposit) and the minority class (deposit).

4.5.2 Statistical Sampling

In contrast, statistical sampling takes into account the distribution of the target classes within the population. By ensuring that the sampling process respects the actual distribution of the target variable in the population, statistical sampling can help the model achieve better class separation. This is particularly important when dealing with an imbalanced dataset, where the minority class may not receive sufficient representation in a randomly sampled dataset. Statistical sampling was applied in this case by calculating the average ratio of target variable occurrences over the sampled time frame for daily data.

4.5.3 Observations from Random Sampling

The random sampling technique shows a high precision and recall for the majority class (class 0), with an accuracy of 83%. However, the minority class (class 1), which represents customers who deposit within 24 hours, shows poorer performance with a precision of 65% and a recall of only 43%. This discrepancy indicates that the model struggles to predict the minority class accurately, which is a common issue when dealing with imbalanced datasets Table 4.5.

Class	**Precision**	**Recall**	**F1-Score**	**Support**
Class 0	0.86	0.93	0.89	113589
Class 1	0.65	0.43	0.52	31411

Table 4.5: Random Classification Report

The ROC-AUC score for random sampling is 0.839, which indicates that the model has a good ability to distinguish between the two classes. However, the overall performance metrics show that while the model does well for the majority class, it fails to provide a meaningful prediction for the minority class Fig 4.9.

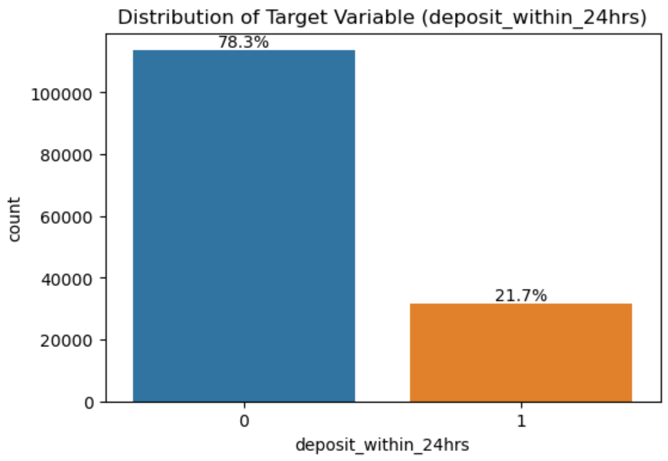


Figure 4.9: Random Sample Distribution

4.5.4 Observations from Statistical Sampling

Statistical sampling yielded different results from the random sampling approach. For the minority class (class 1), the recall increased from 43% to 70%, indicating that the model could better identify those customers who would make deposits within 24 hours. The precision for class 1, however, slightly decreased to 52%, compared to 65% in random sampling. This trade-off between precision and recall is expected in classification problems, particularly when the model shifts focus toward improving recall. The overall accuracy of the model is slightly

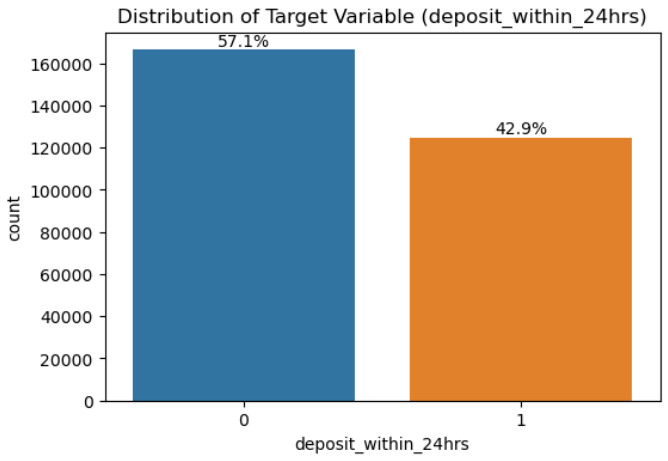


Figure 4.10: Statistical Sample

lower than with random sampling (79% compared to 83%). However, the improvement in the F1-score for the minority class (from 52% to 60%) Table 4.6 and the increased recall indicates that the model now has a better balance between predicting both classes Fig 4.14.

Class	Precision	Recall	F1-Score	Support
Class 0	0.91	0.82	0.86	113589
Class 1	0.52	0.70	0.60	31411

Table 4.6: Statistical Classification Report

The ROC-AUC score for statistical sampling is 0.841, which is slightly higher than the score for random sampling. This indicates that the statistical sampling approach yields a model that is better at distinguishing between customers who will and will not deposit within 24 hours.

4.5.5 Comparison of Random Sampling and Statistical Sampling

The results from both sampling techniques provide valuable insights into the strengths and weaknesses of each approach. Random sampling demonstrates better performance in terms of overall accuracy and precision for the majority class (class 0). However, it underperforms in terms of recall and F1-score for the minority class (class 1). This indicates that random sampling might not be the most effective approach for dealing with an imbalanced dataset, especially when it is critical to accurately predict the minority class.

Statistical sampling, on the other hand, demonstrates an improvement in recall and F1-score for the minority class (class 1), suggesting that the model is now better able to identify customers who will deposit within 24 hours. This is significant because the ability to predict this minority class is central to the business objective. Although the overall accuracy is slightly lower, the increased recall and improved class balance make statistical sampling a more suitable approach for this particular classification problem.

4.5.6 Significance of Statistical Sampling in Model Training

4.5.6.1 Addressing Class Imbalance

Class imbalance is a common issue in machine learning, particularly in binary classification problems where one class is significantly underrepresented. In this case, the minority class represents customers who deposit within 24 hours. Accurately predicting this behavior is crucial for developing effective marketing strategies, enhancing customer retention, and improving business outcomes. By adopting statistical sampling, the model's ability to identify the minority class is significantly improved, as reflected by the higher recall score and improved F1-score for class 1.

This improvement in recall means that the model is more capable of identifying customers who are likely to deposit within 24 hours, even if it sacrifices some precision. This trade-off is acceptable in cases where the goal is to maximize the identification of the minority class. In this scenario, a higher recall is preferable because it ensures that fewer customers who might deposit are missed by the model.

4.5.6.2 Improved Class Separation

One of the key benefits of statistical sampling is its ability to provide better class separation. In random sampling, the model is often biased toward the majority class because it dominates the training data. This leads to poor performance when predicting the minority class, as seen with the low recall in random sampling. Statistical sampling, by ensuring that the ratio of the classes in the sample reflects their actual distribution, helps the model achieve better separation between the classes. This is evident in the improved F1-score for the minority class.

4.5.6.3 Better Alignment with Business Objectives

The primary goal of the model is to identify customers who will deposit within 24 hours, as this behavior is closely tied to marketing and customer engagement strategies. While overall accuracy is an important metric, it is more important for the model to correctly identify the minority class, as these are the customers that the business is most interested in. By using statistical sampling, the model becomes more aligned with the business objective of targeting depositors, even if the overall accuracy is slightly lower than with random sampling.

4.6 Model Selection

Class	**Predicted 0**	**Predicted 1**
Actual 0	93352	20237
Actual 1	9636	21775

Table 4.7: Confusion Matrix - Random Forest

Class	**Precision**	**Recall**	**F1-Score**	**Support**
Class 0	0.91	0.82	0.86	113589
Class 1	0.52	0.69	0.59	31411

Table 4.8: Classification Report - Random Forest

When comparing machine learning models for performance evaluation, a variety of metrics are typically used to ensure that the model selected aligns best with the overall objective. In this case, two models—Random Forest and LightGBM—were evaluated based on standard classification metrics such as accuracy, precision, recall, F1-score, and ROC-AUC Table 4.8. To provide additional insight, more advanced metrics like KS (Kolmogorov-Smirnov statistic), Gain, and Lift were also examined, as the goal was to maximize the separation between customers likely to make a deposit and those who are not, while maintaining business-relevant lift for marketing actions.

Class	**Predicted 0**	**Predicted 1**
Actual 0	92856	20733
Actual 1	9278	22133

Table 4.9: Confusion Matrix - LightGBM

Class	**Precision**	**Recall**	**F1-Score**	**Support**
Class 0	0.91	0.82	0.86	113589
Class 1	0.52	0.70	0.60	31411

Table 4.10: Classification Report - LightGBM

4.6.1 Performance of Random Forest

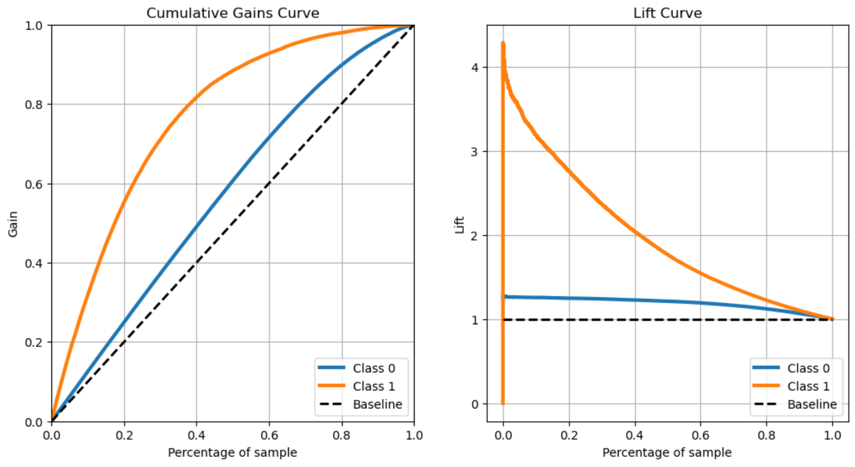


Figure 4.11: Random Forest

KS Statistics, Gain, and Lift for Random Forest:

- The KS statistic shows a maximum value of 52.35%, indicating how well the model is separating the two classes. This metric, which compares the cumulative distribution of true positives and false positives, helps in understanding the model's discriminatory power.

- Gain indicates how much better the model is performing compared to random guessing. In this case, the Gain reaches 100%, demonstrating that the model provides significant lift over random selection Fig 4.11, particularly in the highest decile, where the Gain is around 20.83% with a Lift of 2.08.
- The Lift of 1.00 in the final decile shows that the model is doing no better than random guessing for the lowest-scoring decile, as expected Table 4.11.

Decile Group	**KS Stats**	**Max KS**	**Gain**	**Lift**
(0.837, 0.959]	18.95		20.83	2.08
(0.756, 0.837]	33.63		39.22	1.96
(0.638, 0.756]	44.49		55.43	1.85
(0.508, 0.638]	51.28		69.31	1.73
(0.395, 0.508]	52.35	*****	79.92	1.60
(0.242, 0.395]	48.89		87.94	1.47
(0.173, 0.242]	39.69		92.67	1.32
(0.122, 0.173]	29.05		96.60	1.21
(0.0872, 0.122]	15.48		98.84	1.10
(0.0392, 0.0872]	0.01		100.00	1.00

Table 4.11: KS, Gain, and Lift for Random Forest

4.6.2 Performance of LightGBM

KS Statistics, Gain, and Lift for LightGBM

- The KS statistic shows a maximum value of 52.96%, slightly higher than Random Forest. This slight improvement indicates that LightGBM is slightly better at distinguishing between the positive and negative classes.
- Gain in the first decile for LightGBM is around 20.82%, and Lift is 2.08, similar to the Random Forest model. However, overall, LightGBM edges out Random Forest in the mid-decile ranges with better separation and Lift, providing a slight advantage in overall predictive performance Fig 4.12.
- The Lift is 1.00 for the final decile, showing that the model also behaves as expected, with random-like performance in the least significant group Table 4.12.

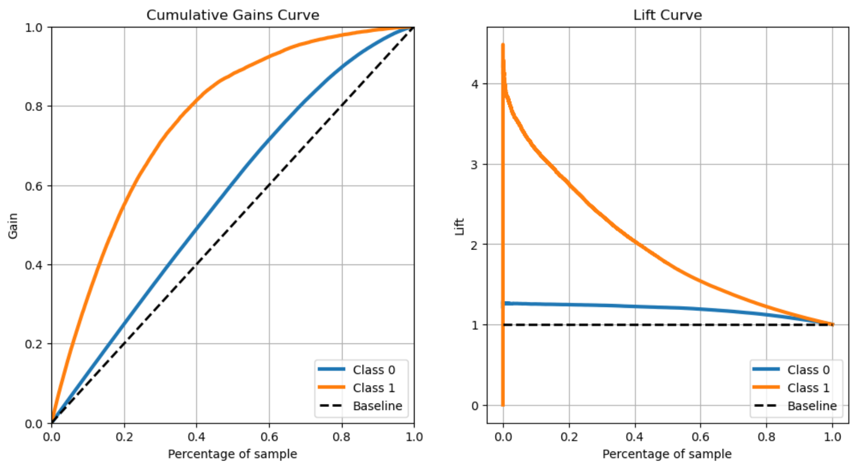


Figure 4.12: LightGBM

Decile Group	KS Stats	Max KS	Gain	Lift
(0.839, 0.989]	18.94		20.82	2.08
(0.755, 0.839]	33.72		39.27	1.96
(0.643, 0.755]	44.64		55.51	1.85
(0.52, 0.643]	51.42		69.38	1.73
(0.4, 0.52]	52.96	*****	80.26	1.61
(0.243, 0.4]	49.16		88.09	1.47
(0.185, 0.243]	40.16		92.95	1.33
(0.118, 0.185]	29.32		96.76	1.21
(0.0764, 0.118]	15.55		98.89	1.10
(0.0174, 0.0764]	-0.01		100.00	1.00

Table 4.12: KS, Gain, and Lift for LightGBM

4.6.3 Key Findings and Justifications

- **KS Statistic**: The KS statistic for LightGBM was slightly higher than for Random Forest, reaching 52.96% versus 52.35%. While the difference is small, it indicates that LightGBM has a marginally better capability in terms of class separation.
- **F1-Score and Recall**: LightGBM showed a slight improvement in recall for the minority class (class 1). The recall increased from 69% for Random Forest to 70% for LightGBM. The F1-score for the minority class is also marginally better in LightGBM (60%) compared to Random Forest (59%).
- **ROC-AUC Score**: The LightGBM model has a ROC-AUC score of 0.8414, which is

slightly higher than the 0.8385 for Random Forest Fig 4.13. Although the difference is minimal, ROC-AUC is an important metric when dealing with imbalanced datasets, as it reflects the model's ability to differentiate between classes.

- **Lift and Gain**: While both models provided strong Lift and Gain in the top deciles, LightGBM maintained better performance across the deciles in terms of cumulative gain. The Lift for the highest decile (2.08) was the same for both models, but LightGBM maintained its advantage across deciles, especially in mid-range deciles, demonstrating a more consistent ability to provide value over random selection throughout the score range.
- **Precision**: The precision for the minority class (52%) was the same for both models, but LightGBM had a higher recall, which resulted in a slightly better F1-score for class 1. This is crucial because identifying potential depositors (minority class) is the primary objective of this modeling task.
- **Business Impact (KS, Gain, and Lift)**: The higher KS statistic and better Gain and Lift for LightGBM suggest that it provides more business value in terms of identifying customers who are likely to deposit within 24 hours. The Lift of 2.08 in the top decile shows that the model doubles the efficiency of marketing campaigns compared to random targeting.

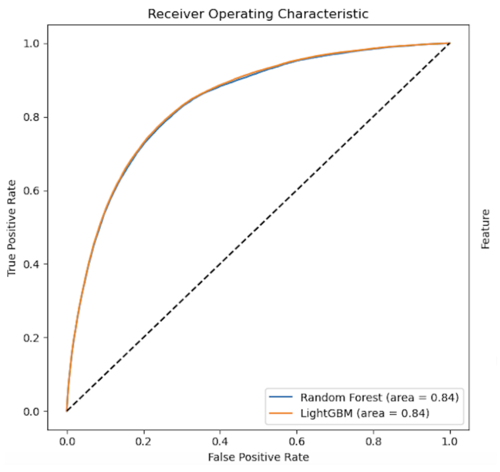


Figure 4.13: ROC Curve

In conclusion, both models perform very well and are highly comparable in terms of standard classification metrics. However, based on the KS statistic, Gain, and Lift, LightGBM has a slight edge over Random Forest. These additional metrics provide insight into how well the model is distinguishing between classes and the overall business value it can deliver in terms of targeted marketing campaigns. The higher KS statistic and ROC-AUC score, combined with improved Lift in mid-decile ranges, make LightGBM the better choice for this problem.

4.7 Analysis of LightGBM Performance on Live Data

The analysis of the LightGBM model on live daily data provides valuable insights into the model's performance in a real-world scenario. To assess whether the model's performance is satisfactory, we need to compare the results on live data against the results on the validation dataset. This comparison will help determine if the model is stable, reliable, and can generalize well to unseen data. Additionally, we'll examine whether any notable performance gaps exist and if further adjustments or fine-tuning are necessary.

4.7.1 Performance on Live Data vs Validation Data

Class	**Predicted 0**	**Predicted 1**
Actual 0	195398	43589
Actual 1	16429	41866

Table 4.13: Confusion Matrix

Class	**Precision**	**Recall**	**F1-Score**	**Support**
Class 0	0.92	0.82	0.87	238987
Class 1	0.49	0.72	0.58	58295

Table 4.14: Classification Report

- **Accuracy**: The overall accuracy has remained consistent between the validation dataset and live data. The model maintains an accuracy of around 80%, indicating that it generalizes well from the validation set to the live environment. This is a positive sign, as a significant drop in accuracy could indicate overfitting or model instability.

- **Precision for Class 0**: The precision for class 0 (non-depositors) has remained high and consistent at 91–92%, meaning the model continues to predict the majority class with high precision.

- **Precision for Class 1**: For class 1 (depositors), precision has dropped slightly from 52% in the validation set to 49% on live data. This small reduction suggests that the model is predicting more false positives (i.e., incorrectly predicting non-depositors as depositors) Table 4.14.

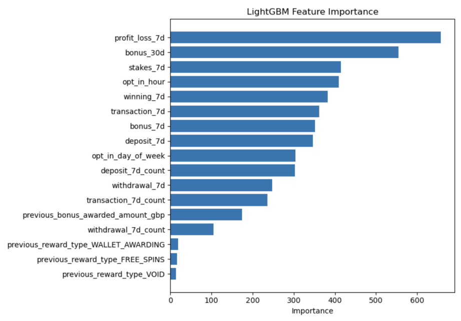


Figure 4.14: Feature Importance

- **Recall for Class 0**: Recall for class 0 (non-depositors) has remained steady at 82% on both live data and the validation set, showing that the model is consistent in correctly identifying non-depositors.
- **Recall for Class 1**: The recall for class 1 (depositors) has improved slightly on live data, increasing from 70% to 72%. This improvement means that the model is able to correctly identify a higher proportion of depositors in a live environment than during validation.
- **F1-Score**: The F1-score for class 0 (non-depositors) has improved slightly from 86% to 87%, indicating a well-balanced trade-off between precision and recall. However, for class 1 (depositors), the F1-score has decreased from 60% in the validation set to 58% on live data. This minor drop suggests that the model is facing challenges in balancing precision and recall for depositors, though it is still within an acceptable range.
- **ROC-AUC Score**: The ROC-AUC score has dropped from 0.8414 in the validation set to 0.7679 on live data. This decrease indicates that the model's ability to separate between depositors and non-depositors has diminished slightly in a live environment. While a drop in ROC-AUC is expected when moving from validation to live data due to real-world variability, the extent of the drop suggests that the model might be encountering more complex patterns in the live data that were not present in the validation set.

4.7.2 Conclusions from Live Data Performance

The model's performance on live data shows that it generalizes well to unseen data, especially considering that live environments are often more challenging due to real-world complexities and noise. The consistency in accuracy (80%) and recall for the majority class (82%) between the validation set and live data is a strong indicator that the model is stable and not overfitted to the training/validation set. The LightGBM model's performance on live data is generally satisfactory, with consistent accuracy and recall metrics compared to the validation set. While the precision for depositors dropped slightly, the recall improved, which is important for the business goal of identifying potential depositors. The model's ability to generalize from the validation set to live data is strong, but further tuning and monitoring will be necessary to optimize its performance, particularly for the minority class. The slight decline in the ROC-AUC score, while notable, is not drastic enough to indicate poor generalization but does suggest that further adjustments could help improve performance. We also notice that bonus, transaction and stakes based variable significantly affects model prediction as per the feature importance chart Fig **??**.

Chapter 5

Discussion

The focus of this dissertation is the analysis and development of a machine learning model aimed at predicting customer deposit behavior, with specific emphasis on how various features, such as bonuses, transactions, and profit and loss, influence this behavior. The study explores the predictive modeling process, including exploratory data analysis (EDA), model selection, tuning, and evaluation based on real-time and validation datasets. The primary goal is to create a model that can effectively identify customers likely to make a deposit within a specific time frame, allowing for more targeted marketing strategies and optimized resource allocation. This dissertation discusses in detail the choice between binary and multiclass models, the application of advanced sampling techniques, and the critical evaluation of model performance on live data to assess real-world applicability.

5.1 Problem Context

The overarching problem addressed in this dissertation revolves around identifying customers who are most likely to make a deposit on an online platform. The ability to predict such behavior is invaluable to businesses, especially in industries where customer deposits are tied to revenue streams, such as gaming and online gambling. Marketing campaigns often focus on incentivizing customers through bonuses, and understanding the correlation between these bonuses and customer deposits is essential to maximize return on investment (ROI). Therefore, the development of a machine learning model that can accurately predict deposits allows for a more efficient allocation of marketing resources and personalized customer engagement.

One critical element is the inclusion of features related to bonuses, stakes, winnings, and transactions over specific time periods, such as seven-day and thirty-day intervals. These features provide insight into how customer activity fluctuates over time and help capture patterns that could indicate an increased likelihood of making a deposit. The problem is further complicated by the inherent class imbalance, where a smaller percentage of users typically make deposits, requiring advanced techniques to improve model performance for the minority class.

5.2 Data Exploration and Initial Observations

In the early stages of the project, extensive exploratory data analysis (EDA) was conducted to uncover key trends and patterns in the dataset. The dataset included variables like bonus-awarded amounts, stakes, winnings, deposits, withdrawals, and transaction counts, all of which were aggregated over various time intervals (7 days and 30 days). These time-based aggregations are crucial for understanding customer behavior trends and serve as core features in the predictive model.

Through the EDA, several important insights were discovered:

- **Bonus and Deposit Behavior**: A significant relationship was found between bonuses awarded to customers and their likelihood of making a deposit. The frequency of deposits increased when a bonus was awarded, and deposit behavior showed a notable spike following the issuance of free spins and cash-based bonuses.

- **Hourly and Weekly Patterns**: Customer deposit behavior exhibited temporal patterns, with peaks at specific hours of the day and days of the week. Deposits were more likely in the early hours of the day and on certain days of the week.

- **Stake and Winnings Relationship**: A linear relationship between stakes and winnings was discovered, where customers who staked larger amounts tended to win more, and this increased engagement was correlated with a higher likelihood of making a deposit.

These observations informed feature selection, where variables such as *bonus_7d*, *profit_loss_7d*, *transaction_7d*, and *deposit_within_24hrs* were identified as key indicators of future deposit behavior. Time-based features were also extracted to capture temporal dependencies in the data.

5.3 Model Selection: Binary vs. Multiclass Classification

One of the major decisions in this project was the selection of the target variable. Two potential approaches were explored: a binary classification task and a multiclass classification task. The binary classification approach sought to predict whether or not a customer would make a deposit, while the multiclass approach aimed to predict different deposit behaviors based on the combination of bonuses awarded and deposits made.

Through comparative evaluation using validation datasets, it was evident that the binary classification approach was more effective. The binary classification model consistently demonstrated higher accuracy, precision, recall, and F1-scores. For instance, the LightGBM model achieved a precision of 92% for the majority class (non-depositors) and 49% for the minority class (depositors), with an overall accuracy of 80%. The multiclass model struggled with class separation and achieved lower performance metrics across all classes.

5.4 Sampling Techniques: Random vs. Statistical Sampling

During model training, class imbalance was observed, with the majority of customers not making deposits. This class imbalance posed a challenge, particularly in improving the model's performance for the minority class (depositors). Initially, random sampling was used, but it led to suboptimal precision for the minority class.

To address this, statistical sampling techniques were introduced, such as stratified sampling, which ensured that the ratio of the target variable remained consistent with the population dataset. This approach improved the model's precision, recall, and F1-scores. After applying statistical sampling, the LightGBM model achieved a recall of 72% for the minority class, compared to 70% with random sampling.

5.5 Model Tuning and Evaluation

Grid search and random search were employed to tune the hyperparameters of the LightGBM and Random Forest models. Grid search provided better performance metrics, and the tuned models were evaluated based on precision, recall, F1-scores, and ROC-AUC scores.

The evaluation revealed that both models performed similarly in terms of overall metrics. However, LightGBM had a slight edge based on additional metrics like KS (Kolmogorov-Smirnov), Gain, and Lift. The KS statistic indicated better separation between depositors and non-depositors, and the Gain and Lift metrics also favored LightGBM. Consequently, LightGBM was selected for deployment.

5.6 Performance on Live Data

The LightGBM model was tested on live daily data to verify its performance in a real-world scenario. The model's performance was consistent with the validation results, achieving an accuracy of 80% and an ROC-AUC score of 0.7679. While the recall for depositors improved slightly from 70% to 72%, the precision for depositors dropped from 52% to 49%.

Despite the drop in precision, the overall performance was deemed satisfactory, especially given the challenges of real-world variability. The improvement in recall for depositors suggests that the model is well-suited for identifying customers likely to make a deposit, which aligns with the business objective.

5.7 Significance of Findings

The findings of this dissertation demonstrate the importance of feature selection, sampling techniques, and model tuning in developing a robust machine learning model for predicting customer deposit behavior. Key findings include:

- **Influence of Bonuses**: Bonuses, especially free spins, significantly influence customer deposits, providing insights for targeted marketing strategies.
- **Temporal Patterns**: Incorporating temporal features helped capture hourly, weekly, and monthly patterns in customer behavior, improving prediction accuracy.
- **Class Imbalance**: Statistical sampling techniques improved the model's ability to identify depositors and can be applied to other imbalanced classification tasks.
- **Model Generalization**: The LightGBM model demonstrated strong generalization from validation data to live data, with stable performance metrics, indicating its reliability for deployment.

Chapter 6

Conclusions and Future Work

This dissertation aimed to develop a robust machine learning model to predict customer deposit behavior on an online platform, particularly focusing on the impact of bonuses, temporal activity, and transactional behavior. The ultimate objective was to improve marketing strategies by identifying customers who are more likely to make deposits, thus optimizing resources and enhancing business outcomes. The process involved several critical stages, including extensive exploratory data analysis (EDA), model selection, sampling techniques, hyperparameter tuning, and evaluating the models on validation and live data. The final conclusion outlines the key takeaways and insights gathered from this project.

6.0.1 Problem Definition and Data Understanding

The core problem addressed in this dissertation was to predict whether a customer would make a deposit within a given time frame, particularly after receiving a bonus or incentive. The dataset included transactional data, such as stakes, winnings, and deposits, as well as bonus data and timestamps, allowing us to capture both financial and temporal trends in customer behavior.

The data exploration phase revealed several key insights that shaped the feature selection and model development stages. Bonuses played a significant role in influencing deposit behavior, particularly free spins and cash bonuses. Temporal features, such as the hour of the day and day of the week, showed clear patterns in deposit frequency, indicating fluctuations in customer behavior depending on the time of interaction with the platform. Class imbalance, with fewer customers making deposits, posed a challenge and informed the decision to apply advanced sampling techniques.

6.0.2 Model Selection: Binary vs. Multiclass Classification

A critical decision in the project was determining the nature of the classification task. Two approaches were considered: binary classification, predicting whether a customer made a deposit, and multiclass classification, capturing bonus-related deposit actions. After evaluating both models, the binary classification model proved to be more effective and practical. The binary model consistently demonstrated higher accuracy, precision, recall, and F1-scores, and was easier to interpret for business applications.

In contrast, the multiclass model struggled to separate behaviors and resulted in lower performance metrics. Since the goal was to predict customer deposit behavior, the binary model, with better performance and interpretability, was selected for the final implementation.

6.0.3 Addressing Class Imbalance: Sampling Techniques

Class imbalance presented a significant challenge, with the majority of customers not making deposits. Initially, random sampling was used, leading to low precision for depositors. To address this, statistical sampling techniques were applied, specifically stratified sampling, which maintained the ratio of depositors to non-depositors in the training data.

This adjustment significantly improved model performance. For example, after applying statistical sampling, the recall for depositors in the LightGBM model increased, enabling the model to better identify customers likely to make deposits. This improvement highlighted the importance of addressing class imbalance in predictive modeling.

6.0.4 Model Tuning and Evaluation

Two machine learning models, LightGBM and Random Forest, were considered. Both models were tuned using grid search to optimize hyperparameters and evaluated on metrics such as precision, recall, F1-scores, and ROC-AUC scores. While both models performed well, LightGBM slightly outperformed Random Forest, particularly in additional evaluation metrics like KS (Kolmogorov-Smirnov), Gain, and Lift.

LightGBM's superior performance in class separation, gain, and lift made it the better choice for real-world applications. The LightGBM model achieved an ROC-AUC score of 0.8414, while Random Forest scored 0.8385.

6.0.5 Performance on Live Data

The LightGBM model was evaluated on live daily data to assess its performance in a real-world scenario. The results were consistent with the validation set, achieving an accuracy of 80% and a ROC-AUC score of 0.7679. The model generalized well to live data, indicating its reliability for production use.

However, there was a slight drop in precision for depositors, from 52% to 49%, indicating more false positives. Despite this, the model's overall recall and F1-scores remained strong, making it a reliable tool for predicting customer deposit behavior.

6.0.6 Key Findings

Several important findings emerged from this dissertation:

- **Significance of Bonuses**: Bonuses, especially free spins, were highly influential in predicting customer deposit behavior. This insight can inform targeted marketing strategies.
- **Temporal Patterns**: The model captured temporal trends, such as higher deposit activity during specific hours and days, emphasizing the importance of incorporating time-based features.
- **Class Imbalance**: Addressing class imbalance through statistical sampling significantly improved the model's ability to predict depositors, showcasing the importance of appropriate sampling techniques in imbalanced datasets.
- **Model Performance**: LightGBM outperformed Random Forest in terms of key metrics like precision, recall, KS, Gain, and Lift, demonstrating its suitability for handling complex, imbalanced data.

6.0.7 Practical Implications

The LightGBM model provides several practical benefits. It enables more targeted marketing efforts by identifying customers most likely to make deposits, supports personalized bonus offers, and optimizes resource allocation. The model's strong generalization to live data ensures reliability in production environments, driving informed business decisions.

6.0.8 Limitations and Future Work

Despite its strong performance, the LightGBM model showed a slight drop in precision for depositors on live data, suggesting that further refinement is needed to reduce false positives. Additionally, while the model captured temporal patterns well, incorporating more customer segmentation or behavioral trends could further enhance its predictive accuracy.

Future work could explore advanced techniques like SMOTE (Synthetic Minority Oversampling Technique) to address class imbalance or the use of ensemble methods that combine the strengths of multiple models. Expanding the dataset to include more granular customer information, such as demographic data, could provide deeper insights into customer behavior and improve the model's predictive power.

References

Batista, Gustavo EAPA, Ronaldo C Prati, and Maria Carolina Monard (2004). “A study of the behavior of several methods for balancing machine learning training data”. In: *ACM SIGKDD explorations newsletter* 6.1, pp. 20–29.

Breiman, Leo (2017). *Classification and regression trees*. Routledge.

Brownlee, Jason (Aug. 2020). *Repeated k-Fold Cross-Validation for Model Evaluation in Python*. en-US. URL: `https://machinelearningmastery.com/repeated-k-fold-cross-validation-with-python/` (visited on 09/13/2024).

Chawla, Nitesh V et al. (2008). “Automatically countering imbalance and its empirical relationship to cost”. In: *Data Mining and Knowledge Discovery* 17, pp. 225–252.

Cirqueira, Douglas et al. (2019). “Customer purchase behavior prediction in e-commerce: A conceptual framework and research agenda”. In: *International workshop on new frontiers in mining complex patterns*. Springer, pp. 119–136.

Coussement, Kristof and Koen W De Bock (2013). “Customer churn prediction in the online gambling industry: The beneficial effect of ensemble learning”. In: *Journal of Business Research* 66.9, pp. 1629–1636.

Fernández, Alberto et al. (2018). “Imbalanced classification with multiple classes”. In: *Learning from imbalanced data sets*, pp. 197–226.

Galar, Mikel et al. (2011). “A review on ensembles for the class imbalance problem: bagging-, boosting-, and hybrid-based approaches”. In: *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)* 42.4, pp. 463–484.

Grönros, Lovisa and Ida Janér (2018). *Predicting Customer Churn Rate in the iGaming Industry using Supervised Machine Learning.*

Gu, Qiong, Li Zhu, and Zhihua Cai (2009). “Evaluation measures of the classification performance of imbalanced data sets”. In: *Computational Intelligence and Intelligent Systems: 4th International Symposium, ISICA 2009, Huangshi, China, October 23-25, 2009. Proceedings 4*. Springer, pp. 461–471.

Hancock, John and Taghi M Khoshgoftaar (2021). “Impact of hyperparameter tuning in classifying highly imbalanced big data”. In: *2021 IEEE 22nd international conference on information reuse and integration for data science (IRI)*. IEEE, pp. 348–354.

Jha, Anupama, Meenu Dave, and Supriya Madan (2019). “Comparison of binary class and multi-class classifier using different data mining classification techniques”. In: *Proceedings of International Conference on Advancements in Computing & Management (ICACM).*

Ke, Guolin et al. (2017). “LightGBM: A Highly Efficient Gradient Boosting Decision Tree”. In: *Advances in Neural Information Processing Systems*. Vol. 30. Curran Associates, Inc. URL: `https://proceedings.neurips.cc/paper/2017/hash/6449f44a102fde848669bdd9eb6b76fa-Abstract.html` (visited on 09/13/2024).

Lakshmi, T Jaya and Ch Siva Rama Prasad (2014). “A study on classifying imbalanced datasets”. In: *2014 First international conference on networks & soft computing (ICNSC2014)*. IEEE, pp. 141–145.

Liashchynskyi, Petro and Pavlo Liashchynskyi (2019). “Grid search, random search, genetic algorithm: a big comparison for NAS”. In: *arXiv preprint arXiv:1912.06059*.

Lin, Yi, Yoonkyung Lee, and Grace Wahba (2002). “Support vector machines for classification in nonstandard situations”. In: *Machine learning* 46, pp. 191–202.

Low, Raymond, Lynette Cheah, and Linlin You (2020). “Commercial vehicle activity prediction with imbalanced class distribution using a hybrid sampling and gradient boosting approach”. In: *IEEE Transactions on Intelligent Transportation Systems* 22.3, pp. 1401–1410.

RandomForestClassifier (2024). en. URL: `https://scikit-learn/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html` (visited on 09/13/2024).

Schellong, Daniel, Jan Kemper, and Malte Brettel (2017). “Generating consumer insights from big data clickstream information and the link with transaction-related shopping behavior”. In.

Sun, Ranzhi (2024). “Applications of Machine Learning Algorithms in Predicting User’s Purchasing Behavior”. In: *Science and Technology of Engineering, Chemistry and Environmental Protection* 1.7.

Zhu, Wen, Nancy Zeng, Ning Wang, et al. (2010). “Sensitivity, specificity, accuracy, associated confidence interval and ROC analysis with practical SAS implementations”. In: *NESUG proceedings: health care and life sciences, Baltimore, Maryland* 19, p. 67.

Appendix A

Motivation

A.1 Motivation

In the competitive landscape of online platforms, customer retention and engagement are paramount to business success. Many companies offer incentives such as bonuses and free spins to encourage customers to make deposits. However, understanding which customers are likely to respond to these incentives and subsequently make a deposit remains a challenge. With a large dataset of customer transactions, bonuses, and temporal data, this project seeks to use machine learning techniques to predict deposit behavior. The outcome can help businesses optimize their bonus offers and marketing strategies by targeting customers who are most likely to deposit, thereby maximizing return on investment (ROI).

The motivation behind this project is to help businesses better allocate their resources, personalize marketing campaigns, and enhance customer engagement through predictive analytics. By accurately forecasting customer deposit behavior, online platforms can improve their bonus structures and enhance profitability through data-driven strategies.

A.2 Project Aim

The project aims to develop a machine learning-based predictive model that identifies the likelihood of a customer making a deposit within a given timeframe, particularly after receiving a bonus or incentive. The specific objectives are:

1. To analyze customer transaction and bonus data to identify key factors influencing deposit behaviour and build predictive models using various machine learning techniques. To assess the impact of bonuses, temporal data, and transactional patterns on the likelihood of making deposits on short term basis or daily data.

2. To provide actionable insights that businesses can use to optimize their marketing strategies.

3. To present a reliable and deployable model that can predict customer deposit behavior in real-time to the company.

A.3 Data

The dataset for this project consists of customer transactional and bonus-related data from an online platform. This data is obtained from the client's internal databases, either through direct access or anonymized samples. The data will be preprocessed to handle missing values, outliers, and other potential anomalies before feature engineering and model training.

A.4 Project Timeline

The following timeline outlines the key stages and milestones for developing the machine learning-based predictive model:

1. **Week 1-2: Project Setup and Data Exploration**
   - Set up the project environment and access the required dataset.
   - Conduct an initial exploratory data analysis (EDA) to understand the structure, quality, and potential of the dataset.
   - Visualize the data distributions and relationships using temporal plots (hourly, weekly, monthly) and other relevant charts.
2. **Week 3-4: Feature Engineering and Data Preprocessing**
   - Engineer new features based on temporal data (day of the week, time of day) and customer bonus/transactional data.
   - Handle missing data, normalize/scale features, and prepare data for modeling.
   - Address class imbalance using statistical sampling techniques to ensure that minority classes are well represented during model training.
3. **Week 5-6: Model Development and Initial Evaluation**
   - Train the first set of machine learning models (LightGBM and Random Forest).
   - Evaluate the models on the validation set using metrics like precision, recall, F1-score, and ROC-AUC.
   - Tune hyperparameters using grid search for both models to optimize performance.
4. **Week 7-8: Model Refinement and Final Evaluation**

  - Apply stratified sampling to balance the classes and improve model performance.
  - Finalize the model with the best performance and evaluate on unseen test data.
  - Conduct additional evaluations using KS, Gain, and Lift metrics to assess model utility for real-world deployment.

5. **Week 9-10: Reporting and Final Deliverables**
   - Generate a detailed project report with results, conclusions, and actionable insights for the business.
   - Submit the final dissertation.

A.5 Deliverables

A model which can identify the customers who will deposit the money after receiving bonus from promotions, to align promotion effectively to increase return of investment and customer engagement. It will help to retain new customers and better personalization to provide good user experience.

Note: Changes can be made in the above-mentioned timelines and aims during the project depending on the challenges, requirements, and time scale.