TELECOM CUSTOMER CHURN PREDICTION

GROUP B

Eduardo Jr Morales- C0900536

Eduardo Roberto Williams Cascante- C0896405

Ezgi Tanyeli- C0877348

Flora Mae Villarin- C0905584

Haldo Jose Somoza Solis- C0904838

Marzieh Mohammadi Kokaneh- C0898396

Big Data Analytics, Lambton College

BDM 2053- Big Data Algorithms and Statistic

Meysam Effati

December 14, 2023

TABLE OF CONTENTS

TELECOM CUSTOMER CHURN PREDICTION

Predicting and preventing customer churn is of critical importance for businesses to gain a competitive advantage. This analysis aims to understand and predict customer churn, specifically examining the performance of models developed using XGBoost and LightGBM algorithms. Machine learning techniques prove effective in uncovering patterns in customer behaviors within extensive and complex datasets, providing valuable insights to businesses.

At the beginning of the analysis, the dataset was explored, descriptive statistics were examined, and potential factors influencing customer churn were identified. During the feature engineering stage, a comprehensive approach was adopted to better understand customer behaviors, and the goal was to enhance model training by extracting new features.

Subsequently, models were constructed using powerful boosting algorithms such as XGBoost and LightGBM. In this phase, there was a specific focus on parameter adjustments to handle the imbalanced dataset. Comprehensive hyperparameter tuning for the XGBoost and LightGBM models aimed to address overfitting issues and enhance performance on specific metrics regarding to customer churn problem.

1. About the Dataset

The dataset provides information about telecom company customers and aims to predict high-risk churn. Customer churn is defined as discontinuing business with the company. The analysis focuses on identifying characteristics and behaviors indicating a higher likelihood of churn, utilizing Exploratory Data Analysis (EDA) and predictive analytics.

Column Name	Definition	Type
customerID	Unique customer identifier	object
gender	Customer's gender (Male, Female)	object
SeniorCitizen	Indicates if the customer is 65 or older (1 for Yes, 0 for No)	object
Partner	Indicates if the customer is a partner (Yes, No)	object
Dependents	Indicates if the customer lives with any dependents (Yes, No)	object
tenure	Total months with the company	int64
PhoneService	Customer subscribes to home phone service (Yes, No)	object
MultipleLines	Customer subscribes to multiple telephone lines (Yes, No, No Phone Service)	object

InternetService	Customer subscribes to Internet service (No, DSL, Fiber Optic)	object
OnlineSecurity	Customer subscribes to additional online security service (Yes, No, No Internet Service)	object
OnlineBackup	Customer subscribes to additional online backup service (Yes, No, No Internet Service)	object
DeviceProtection	Customer subscribes to additional device protection plan (Yes, No, No Internet Service)	object
TechSupport	Customer subscribes to additional technical support plan (Yes, No, No Internet Service)	object
StreamingTV	Customer uses Internet service to stream TV programming (Yes, No, No Internet Service)	object
StreamingMovies	Customer uses Internet service to stream movies (Yes, No, No Internet Service)	object
Contract	Customer's current contract type (Month-to-Month, One Year, Two Year)	object
PaperlessBilling	Customer has chosen paperless billing (Yes, No)	object
PaymentMethod	How the customer pays their bill (Bank transfer, Credit card, Electronic check, Mailed check)	object
MonthlyCharges	Customer's current total monthly charge	float64

Table 1 - Independent Variables

The dataset consists of 21 columns and 7043 rows. The customerID column, being a non-variable during modeling, has been assigned as the index in the dataset. The train_test_split function has been used to split to dataset, with 30% of the total data set aside for testing. After the splitting the dataset into train and test, summary information about the datasets is provided in the table below:

Dataset	#Column	#Row
X_train	19	4930
X_test	19	2113
y_train	1	4930
y_test	1	2113

Table 2 - Shape of Train and Test Datasets

1.1.Target Variable

The 'Churn' column, which indicates whether customers have decided to discontinue their subscription, plays a crucial role in understanding customer behavior. In the training dataset, around 27% of customers have chosen to discontinue their engagement with the platform in the previous month. This implies a notable segment of the customer base is at risk of churning.

It's noteworthy that in the training dataset, the majority, specifically 73% of customers, do not churn. This distribution reveals a significant imbalance, where a considerable portion of customers tends to stay loyal. Recognizing this skewness is vital during the modeling process as it suggests a potential challenge in predicting churn accurately. A model that predicts non-churning instances more often may yield high accuracy but might overlook actual churn cases.

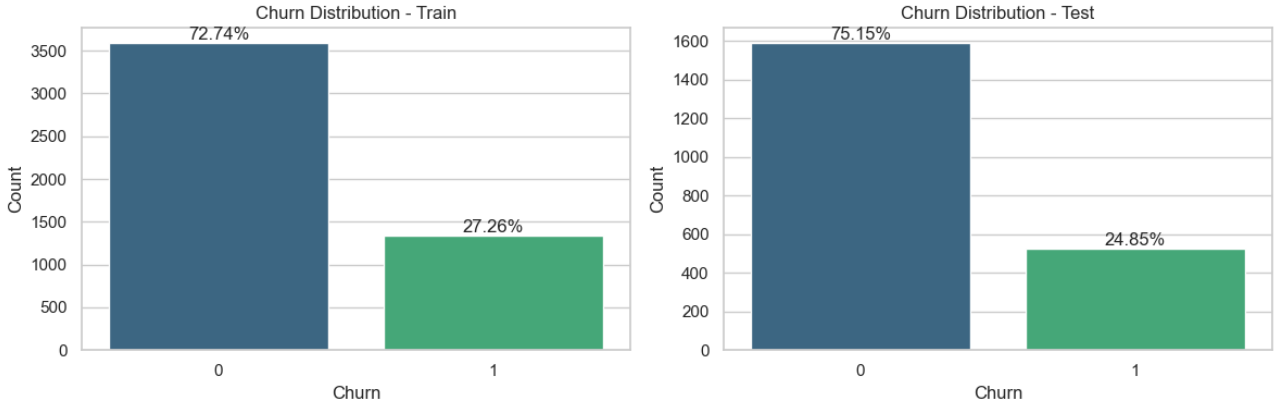


Table 3 - Target Variable Class Distribution in Train and Test Datasets

Strategies for addressing this skewness, such as using techniques like oversampling the minority class or employing different evaluation metrics, will be explored in the modeling section. Balancing the predictive performance for both churn and non-churn instances is crucial for developing a model that effectively identifies customers at risk of discontinuing their subscriptions.

2. Data Preprocessing

2.1.Handling Data Types

The 'TotalCharges' column is initially stored as an object type, but it should be of type float. To address this, we convert the column to numeric format, handling any potential errors during the process.

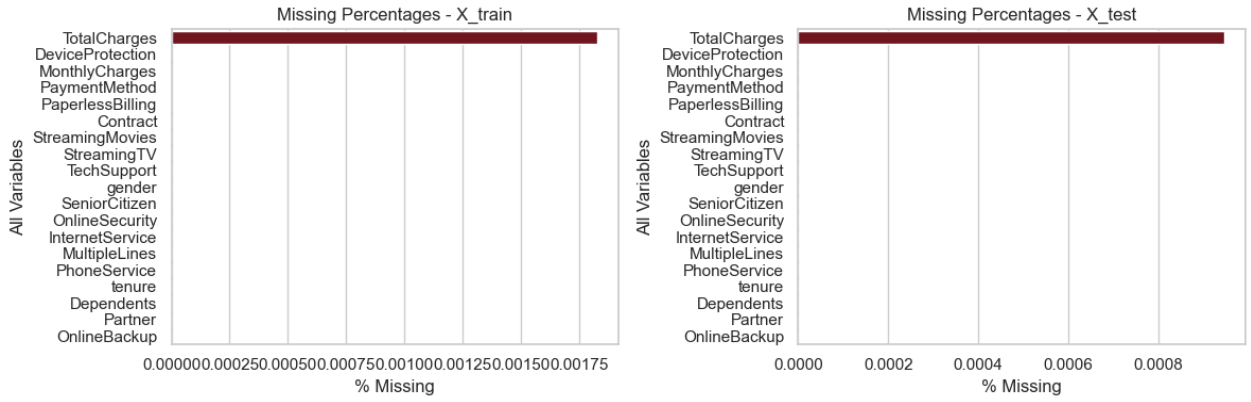


Table 4 - Percentage of Missing Values

The 'TotalCharges' column has 9 null values in the training set and 2 null values in the test set. We handle these null values by filling them with the median value of the 'TotalCharges' column in the training set.

Additionally, considering the nature of the 'SeniorCitizen' column, we convert it to a categorical variable, representing 'Yes' for 1 and 'No' for 0.

2.2.Outlier Detection

Boxplots visualize the distribution of numerical columns, highlighting potential outliers. Among the three columns (tenure, MonthlyCharges, TotalCharges), only the 'TotalCharges' column contains outliers. This suggests that the distribution of total charges has some extreme values that deviate from the typical range, while the other two columns exhibit a more consistent distribution without noticeable outliers.

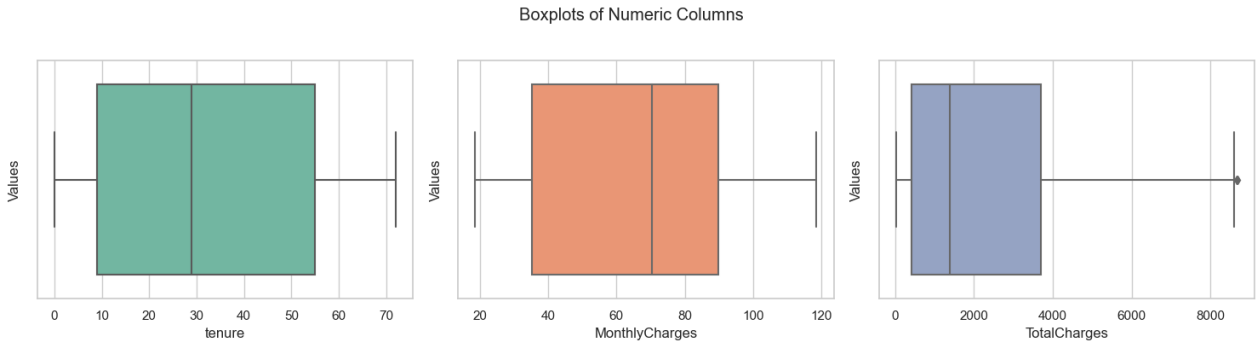


Table 5 - Quantile Ranges of Numeric Variables

To address outliers, the 'TotalCharges' column was capped at the 99.5% quantile. The boxplot after capping outliers demonstrates a more refined distribution below:

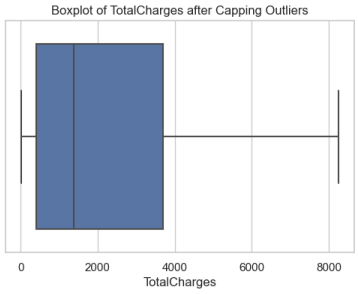
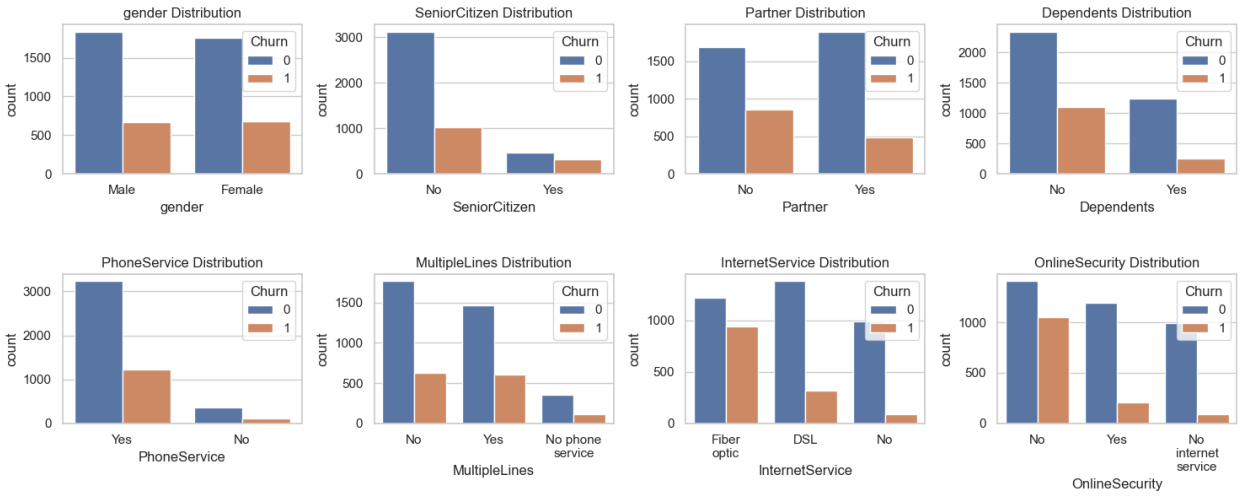


Table 6 - TotalCharges Column After Capping Outliers

3. Exploratory Data Analysis (EDA)

3.1.EDA on Categorical Features

In-depth analysis of categorical variables sheds light on nuanced patterns and trends within our dataset, offering valuable insights into the factors influencing customer churn. Let's delve into the details:



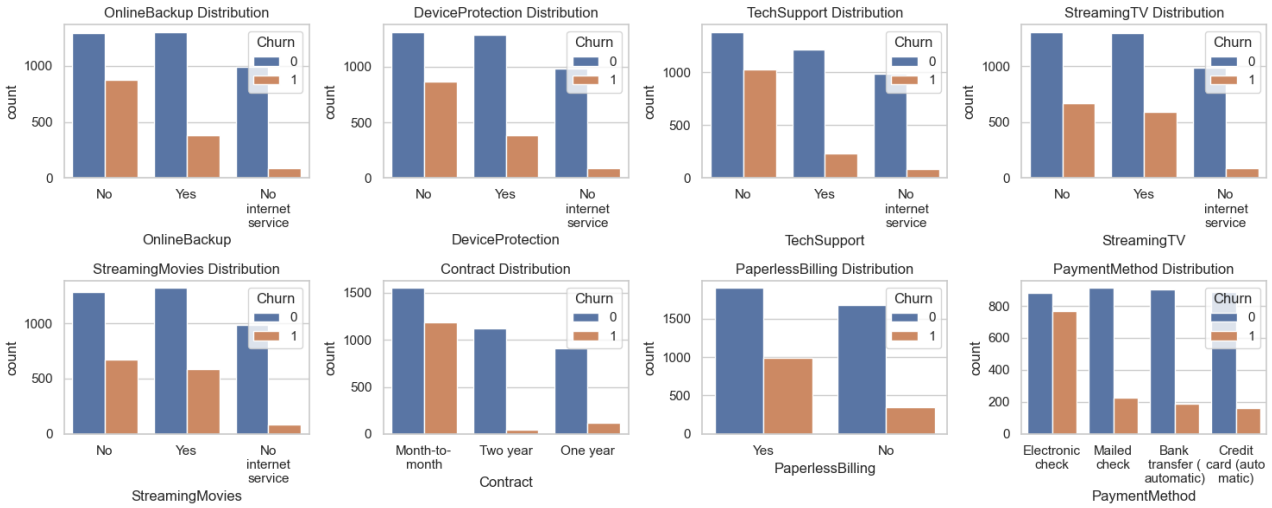


Table 7 - Distribution of Independent Variables

3.1.1. Gender Distribution and Churn

The dataset exhibits a balanced distribution between male and female customers, with each comprising approximately half of the customer base. Interestingly, churn rates appear to be comparable for both genders, indicating that gender alone may not be a significant predictor of churn.

3.1.2. Senior Citizen Status and Churn

A notable observation is that the majority of customers in our dataset fall into the non-senior citizen category. However, a closer examination reveals that senior citizens exhibit a higher likelihood of churn compared to their younger counterparts. This insight prompts further exploration into the unique needs and preferences of senior customers.

3.1.3. Partner and Dependents Influence on Churn

Examining customers' relationship status, around half of them have a partner, while only about 30% have dependents. Strikingly, customers with partners and dependents demonstrate a lower churn rate, suggesting that familial associations may contribute to increased customer loyalty and retention.

3.1.4. Internet Service and Churn

Diving into the realm of internet services, customers subscribed to Fiber Optic Internet Services stand out with a significantly higher churn rate. This raises questions about the quality or perceived value of Fiber Optic services and emphasizes the need for improvements or targeted retention efforts in this service category.

3.1.5. Essential Services and Churn

Customers lacking essential services such as OnlineSecurity, OnlineBackup, DeviceProtection, and TechSupport present a higher likelihood of churning. This underscores the importance of these services in enhancing customer satisfaction and loyalty.

3.1.6. Contract Type and Churn

The analysis of contract types reveals that a substantial portion of customers opts for month-to-month contracts, while an equal number chooses 1-year and 2-year contracts. Notably, customers with monthly subscriptions exhibit a higher churn rate, emphasizing the potential advantages of longer-term contracts in retaining customers.

3.1.7. Paperless Billing and Churn

The choice of paperless billing emerges as a notable factor influencing churn, with customers who opt for paperless billing showing a higher churn rate compared to those who do not. This prompts further investigation into the reasons behind this correlation and potential strategies for mitigating churn in this billing category.

3.1.8. Payment Method and Churn

A noteworthy finding is that customers using the Electronic Check payment method are more likely to churn than those using alternative payment options. Understanding the reasons behind this preference and addressing potential pain points could contribute to more effective churn prevention strategies.

3.2.EDA on Numerical Features

An in-depth exploration of numerical variables provides valuable insights into customer behavior, tenure, and spending patterns.

3.2.1. Tenure

Examining the distribution of customer tenure in months reveals that a significant number of customers have relatively short tenures, while a substantial portion exhibits longer tenures. The diversity in tenure lengths suggests a varied customer base with both recent and long-standing engagements with the platform.

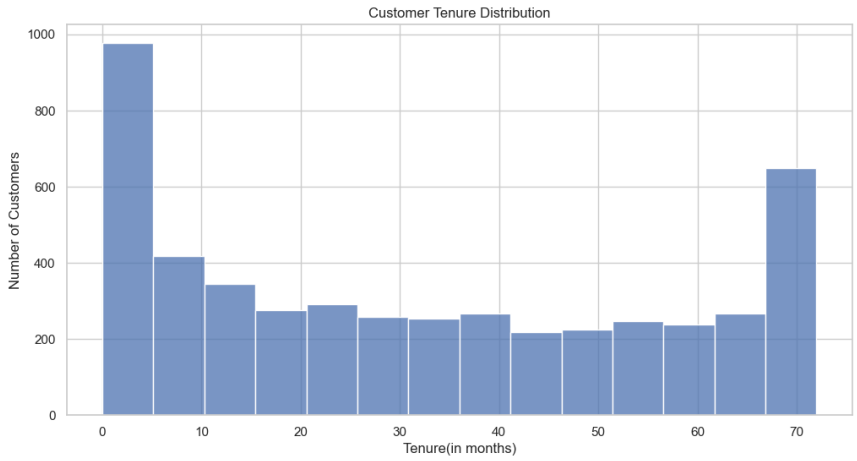
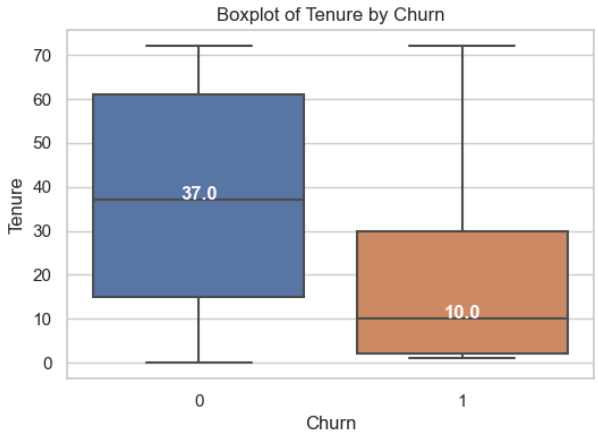


Table 8 - Customer Tenure Distribution in Months

Customers who churn tend to have a median tenure of around ten months. The boxplot below illustrates that customers with longer tenures are less likely to churn compared to those with shorter tenures. This observation suggests that customer loyalty, as reflected in longer tenures, may act as a mitigating factor against churn.



Understanding the relationship between tenure and churn is valuable for identifying potential risk factors. It indicates that newer customers, with shorter histories of engagement, may require special attention and targeted retention strategies. On the other hand, customers with longer tenures may have established a more stable connection with the platform, making them less prone to churning.

3.2.2. MonthlyCharges

The relationship between monthly charges and customer churn, indicating that customers with higher monthly charges are more likely to churn. Specifically, the median monthly charge for customers who decided to discontinue their subscription is approximately $80. This observation implies that customers with elevated monthly charges may be more prone to churning, and it could be a crucial factor to consider in the churn prediction model.

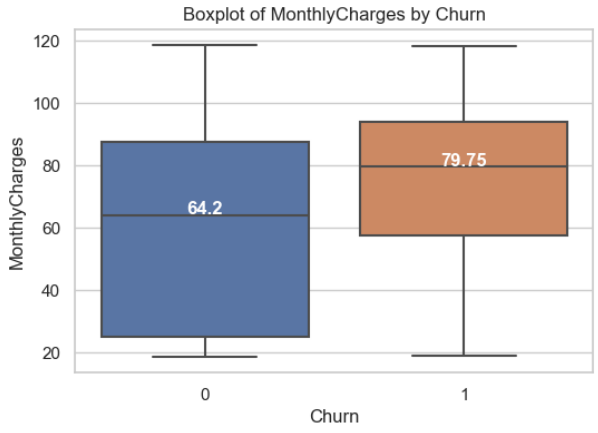


Table 9 - Median Values of MonthlyCharges By Churn Classes

3.2.3. TotalCharges

Comparing the median values of churn and not churn based on TotalCharges column, it is observed that customers who churn have lower total charges (around $718.78) than those who do not churn (1654.8). Interestingly, customers with lower total charges are more likely to churn. This finding suggests a potential correlation between the total charges incurred by a customer and their likelihood of churning.

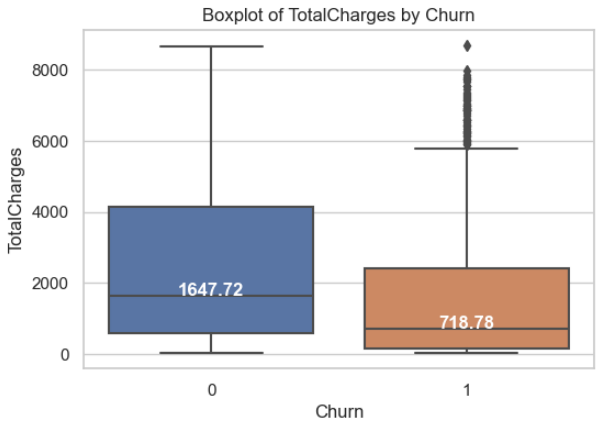


Table 10 - Median Values of TotalCharges By Churn Classes

The insight that customers with lower total charges are more prone to churn could be indicative of different customer segments or behaviors. It may imply that customers with lower spending patterns might be more sensitive to pricing changes or less engaged with additional services, making them more likely to discontinue their subscription.

4. Feature Engineering

In this section, our emphasis was on preparing and transforming the raw data into a format that is conducive to constructing a predictive model. This procedural step, commonly known as feature engineering, entails crafting meaningful representations of the original data, enabling the model to comprehend and utilize them effectively.

4.1.Extraction of New Features

As we embark on the extraction of new features, our primary objective is to elevate the depth of our analysis, unveiling intricate patterns within customer data. Each newly introduced feature serves as a strategic lens, offering unique perspectives into distinct aspects of customer behavior and interaction with the platform.

From categorizing customers based on tenure to identifying those with impending contract renewals, these features are meticulously designed to provide actionable insights for targeted retention efforts.

Furthermore, recognizing vulnerabilities through the absence of essential protection services and understanding the holistic engagement level via total active services contribute to a comprehensive view of customer satisfaction and commitment.

The introduction of features such as 'Any Streaming Service' and 'Auto Payment Flag' reflects a keen awareness of evolving customer preferences in the digital landscape.

By calculating average charges and monitoring increase factors, we not only gauge the financial commitment but also anticipate shifts in spending behavior, enabling us to proactively align our services with evolving customer needs.

The 'Average Service Fee per Active Service' feature adds an additional layer of understanding, shedding light on the perceived value of services.

The details for each newly derived variable are provided below:

- **Tenure-Year Extraction:** Categorizing customers based on tenure into distinct year groups allows for a more nuanced analysis. It can unveil patterns or behaviors specific to certain stages of the customer lifecycle. For example, newly acquired customers (0-1 Year) may have different considerations compared to those who have been with the platform for several years (5-6 Years).
- **Contract Due:** Identifying customers with contracts due can be crucial for targeted retention efforts. Customers under a contract might have a stronger commitment, but those nearing the end of their contract could be more susceptible to churn. This binary indicator helps flag such cases for proactive customer engagement.
- **Any Protection:** Recognizing whether a customer lacks essential protection services provides insights into potential vulnerabilities. Customers without backup, device protection, or tech support might be at a higher risk of churn, especially if these services are perceived as valuable.
- **Total Active Services:** Counting the total number of active services offers a holistic view of a customer's engagement with the platform. A higher count suggests a more

comprehensive utilization of services, potentially indicating a satisfied and committed customer.

- **Any Streaming Service:** Identifying customers who subscribe to streaming services is relevant in the current digital landscape. This information can be used to tailor promotions or content recommendations, addressing the specific preferences of this segment and potentially enhancing customer satisfaction.
- **Auto Payment Flag:** Knowing whether a customer uses automatic payment methods reflects convenience. Customers with auto payments might experience fewer interruptions in service due to billing issues, contributing to a smoother customer experience, and potentially reducing churn.
- **Average Monthly Charges:** Calculating average monthly charges over the entire tenure normalizes the charge data. This helps in understanding the consistent financial commitment of a customer, accounting for any fluctuations in monthly charges over time.
- **Charges Increase Factor:** The increase factor in charges provides insights into the dynamics of a customer's spending behavior. Sudden increases could signal dissatisfaction or the adoption of additional services. Monitoring this factor helps in understanding evolving customer needs.
- **Average Service Fee per Active Service:** Understanding the average cost per active service sheds light on the cost-effectiveness of services utilized by a customer. This can be valuable for tailoring offers or promotions, ensuring that customers perceive a fair value for the services they use.

In essence, these new features aim to empower our strategic decision-making by unraveling the intricacies of customer relationships, fostering personalized interactions, and ultimately fortifying our approach to customer retention.

4.2.One Hot Encoding

Machine learning algorithms often require numerical input, and one-hot encoding provides a numerical representation for categorical data. It is a technique used to convert categorical variables into a format that can be provided to machine learning algorithms to improve predictive performance. Sci-kit Learn provides the OneHotEncoder class to handle categorical inputs using One-Hot Encoding. The algorithm operates as follows:

id	color
1	red
2	blue
3	green
4	blue

id	color_red	color_blue	color_green
1	1	0	0
2	0	1	0
3	0	0	1
4	0	1	0

Table 11 - One Hot Encoding

To illustrate with a specific example, in the Churn dataset 'Contract' column initially contains categories like 'Month-to-month,' 'One year,' and 'Two years,' one-hot encoding results in three binary columns ('Contract_Monthly,' 'Contract_OneYear,' 'Contract_TwoYears'). If a data point corresponds to a particular category, the corresponding binary column is set to 1, otherwise, it is set to 0.

This process has been applied to all categorical columns in both the training set (X_train) and the test set (X_test). This prevents misinterpretation of categorical variables and ensures consistency in the input format for various algorithms.

5. Modelling

In this section, we will delve into the internal mechanisms of two powerful boosting algorithms—XGBoost and LightGBM—with a particular focus on how they effectively tackle the challenge of imbalanced data. Handling imbalanced class distributions is a common and critical aspect in machine learning, and boosting algorithms excel in such scenarios due to their adaptability and ability to prioritize learning from the minority class.

5.1.Controlling The Balance of Class Weights

Effectively balancing class weights in imbalanced data is a critical aspect of model development, playing a pivotal role in enhancing predictive accuracy and ensuring robust performance across diverse performance metrics.

In the context of handling imbalanced data, we calculated and set the same scale_pos_weight parameter for both LightGBM and XGBoost algorithms. According to the XGBoost documentation, a common approach to determining a typical value for the scale_pos_weight parameter is through the following function:

$$scale\_pos\_weight = \frac{\text{sum(negative instances)}}{\text{sum(positive instances)}}$$

By default, the scale_pos_weight hyperparameter is set to 1, implying equal weights for both the majority and minority classes. For example, in our training data, where there are 3586 non-churned customers and 1344 churned customers, the corresponding scale_pos_weight would be approximately 3586 / 1344, or around 2.5. This adjustment ensures that the minority class receives two and a half times more weight, thereby enhancing its impact and correction influence compared to errors made on the majority class.

While adjusting scale_pos_weight is a powerful tool for addressing class imbalance, two essential considerations emerge:

- **Effect on Performance Metrics:**

Changing the scale_pos_weight not only influences the model's treatment of class imbalance but also has ramifications on various performance metrics such as recall, accuracy, ROC-AUC, and precision scores. This adaptability allows practitioners to fine-tune the model based on the specific requirements of the task at hand.

- **Potential Overfitting Concerns:**

Caution must be exercised to avoid overfitting the minority class when adjusting scale_pos_weight. Extreme values of this hyperparameter can lead the model to overly prioritize the minority class, potentially resulting in poorer predictions and reduced generalization to unseen data. Striking the right balance is crucial for optimal model performance in imbalanced settings.

5.2.XGBoost

XGBoost, renowned for its efficacy with imbalanced datasets, introduces a vital hyperparameter known as scale_pos_weight. This hyperparameter plays a crucial role in directly controlling the balance between positive and negative weights, thereby emphasizing the minority class. Furthermore, XGBoost's gradient boosting framework inherently adjusts its learning process to allocate more resources to under-represented classes, contributing to its robust performance in imbalanced settings.

5.2.1. First Model With Default Parameters

In this stage of the analysis, the XGBoost algorithm has been implemented using both default parameters and a carefully adjusted scale_pos_weight to address the high-class imbalance present in the dataset. The primary motivation behind the using of scale_pos_weight parameter is to observe the true effect and performance of the model after hyperparameter tuning.

By employing the algorithm with default parameters allows us to observe the baseline performance of the model without any specific tuning. Default values include a learning rate of 0.3, max depth of 6, and a variety of other settings that XGBoost considers as a starting point.

The evaluation metrics of the default model as follows:

Metric	Value
Accuracy	78%
Precision	55%
Recall	**69%**
F1-Score	61%
Roc-AUC-Train	**99%**
Roc-AUC-Test	**84%**

Table 12 - First Run of The Default XGBoost Model

The disparity between the ROC-AUC value of 99% in the training set and 84% in the test set suggests an overfitting issue, indicating that the model has become overly tailored to the intricacies of the training data, thereby compromising its ability to generalize well on new, unseen data.

To address this overfitting challenge, adjustments will be made to the model parameters. Furthermore, during the parameter tuning process, a key focus will be on enhancing the recall parameter. This strategic approach aims not only to mitigate the overfitting problem but also to reduce the likelihood of potential losses for the company in case of customer churn. Recall, in this context, gauges the model's accuracy in predicting whether a customer will churn or not, specifically emphasizing the significance of minimizing false negatives.

5.2.2. Second Model With Default Parameters

The default model was initially run with all variables, providing a starting point to assess its overall performance. However, the feature importance analysis sheds light on the contributions of

different variables to the model. Particularly, the top-performing features are ranked based on their influence on the model's predictions.

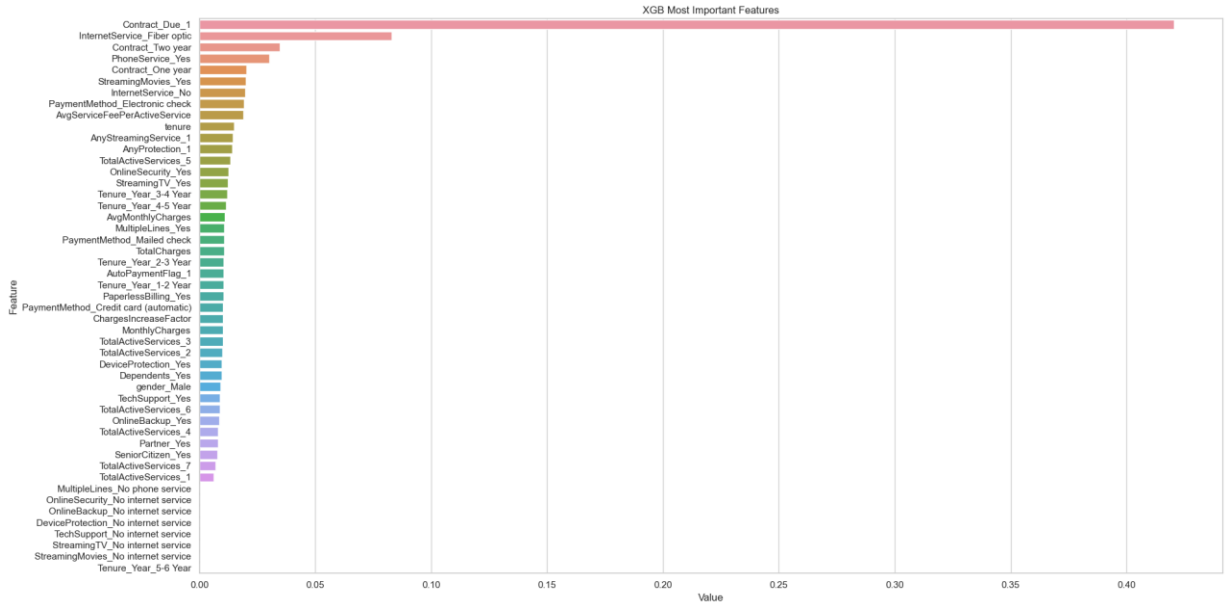


Table 13 - XGBoost Model Feature Importances

In this specific model with a total of 49 variables, the bar plot illustrates that the last 8 variables seem to have limited impact on the model compared to others. This observation suggests that these four variables may not play a significant role in the model's final outputs, or the model may not be very sensitive to these variables.

In light of these observations, it was considered that the contribution of these 8 variables to the model's performance is limited. Subsequent evaluations after removing these variables indicated that the changes were generally minimal or there was no significant alteration in some key parameters. In other words, removing these four variables did not have an effect on the overall performance of the model.

Metric	Value
Accuracy	78%
Precision	55%
Recall	**69%**
F1-Score	61%
Roc-AUC-Train	**99%**
Roc-AUC-Test	**84%**

Table 14 - Second Run of The Default XGBoost Model

5.2.3. Final Model with Hyperparameter Tuning

To extract the utmost potential from XGBoost, a technique renowned for its 'regularized boosting,' effective parameter tuning is imperative. XGBoost, standing for eXtreme Gradient Boosting, outshines Standard GBM in speed due to its implementation of parallel processing. This enables a remarkable acceleration in model training, contributing to its popularity in machine learning applications.

One distinctive feature of XGBoost is its capacity for parallel processing, rendering it significantly faster than traditional Gradient Boosting Machines (GBM). This speed advantage is a crucial factor in handling large datasets and complex models efficiently.

XGBoost's flexibility is further underscored by its provision for users to define custom optimization objectives and evaluation criteria. This capability empowers practitioners to tailor the model to specific use cases, opening a realm of possibilities without predefined limits.

XGBoost simplifies the process of determining the optimal number of boosting iterations by offering built-in cross-validation at each iteration. This feature facilitates a streamlined approach to finding the exact number of boosting iterations required for optimal model performance. This efficiency is particularly advantageous for achieving the right balance between model complexity and predictive accuracy in a single run.

XGBoost authors have categorized parameters into three distinct groups, each serving a specific purpose:

- **General Parameters:** These parameters guide the overall functioning of the XGBoost model. They play a crucial role in shaping the high-level characteristics of the boosting process.
- **Booster Parameters:** Responsible for guiding the behavior of individual boosters, which can be either trees or regression models, at each step of the boosting process. Fine-tuning these parameters contributes to the optimal construction of each booster.
- **Learning Task Parameters:** These parameters guide the optimization process performed by XGBoost for a specific learning task. They provide a means to customize the model's behavior based on the nature of the data and the goals of the analysis.

By incorporating parameter tuning, with a specific focus on booster parameters, adjustments have been made to the following key parameters, in addition to the scale_pos_weight parameter:

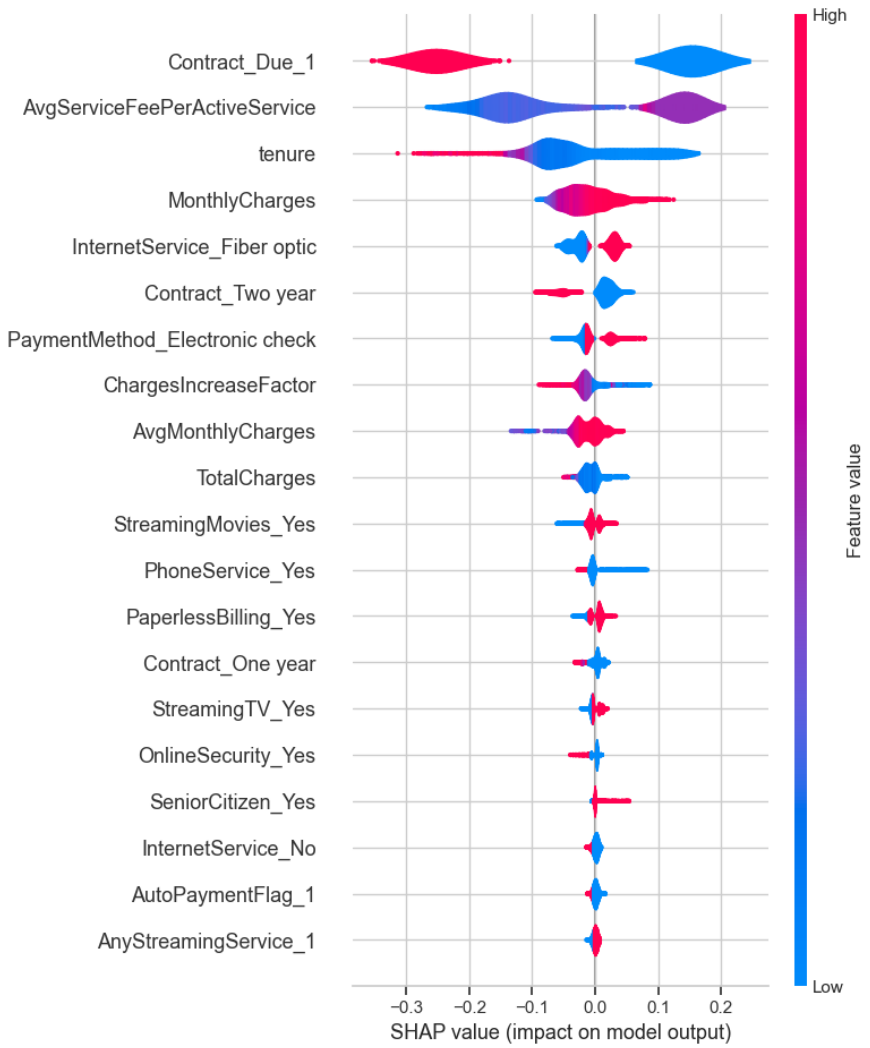
Parameter	Definition	Final Value
learning_rate	The rate at which the model adapts during training. A lower value makes the model more robust but may require more iterations.	0.01
n_estimators	The number of trees to be used in the model. More trees generally lead to better generalization but may increase training time.	43
max_depth	The maximum depth of a tree. Deeper trees can capture more complex relationships but may lead to overfitting.	5
min_child_weight	Minimum sum of instance weight (hessian) needed in a child. Controls the minimum amount of instances that can be assigned to a leaf.	5
gamma	Minimum loss reduction required to make a further partition on a leaf node.	0.84
subsample	The fraction of samples used for each tree. A lower value can make the model more generalizable.	0.8
colsample_bytree	The fraction of features used for each tree.	0.8
objective	Indicates a binary classification problem.	binary logistic'
n_jobs	Number of parallel threads used during training.	4
reg_alpha	L1 regularization term. Help prevent overfitting.	0.1
reg_lambda	L2 regularization term. Help prevent overfitting.	1
tree_method	Specifies the method used for tree learning, utilizing GPU acceleration for faster training.	'gpu_hist'
seed	Random seed for reproducibility.	27

Table 15 – XGBoost Model Hyperparameters Definitions and Final Values After Tuning

Shap values have been also examined to better interpret how variables are positioned in the model after tuning. SHAP (Shapley Additive Explanations), a game theory approach, is aimed at explaining the output of any machine learning model. The SHAP method assigns a value (referred to as SHAP value) to each variable, representing its contribution to the final result of the model. SHAP values can be calculated for tree ensemble models such as decision trees, random forests, and gradient boosting trees.

If the model is a binary classification model, shap.summary_plot() generates a density distribution plot of SHAP values specific to each variable to determine how much each variable's values influence the model. In this plot, the variables are sorted based on the sum of the magnitudes of SHAP values across the entire sample.

The Shapley plot for the tuned XGBoost model can be examined below:



In the above graph, the impact of variables on the model is interpreted with Shapley values on the X-axis. The colors of each variable on the plot depend on whether the values of the variable are low or high. For example, the transition of the Contract_Due_1 variable from red to blue parallel to the X-axis indicates that as the value of the variable decreases, the output value increases (indicating an inversely proportional relationship with the target). This variable essentially takes values of 1 or 0. If the customer has a 1 or 2 years contract, it gets 1; otherwise, it gets 0. This

means that customers with longer contract durations churn less, which aligns with the insight observed for this variable during exploratory data analysis (EDA).

Following a rigorous hyperparameter tuning, the XGBoost model underwent substantial improvements to address overfitting and enhance performance. The summary of the tuned model and default model evaluation metric comparison after tuning is presented in the table below.

Metric	First Default Model	Tuned Model
Accuracy	78%	76%
Precision	55%	51%
Recall	**69%**	**82%**
F1-Score	61%	63%
Roc-AUC-Train	**99%**	**%86**
Roc-AUC-Test	**84%**	**%85**

Table 16 - Comparison of Evaluation Metrics on First and Tuned XGBoost Models

Despite a marginal decline in precision, the tuned model exhibits a significant surge in recall. This noteworthy improvement suggests that the XGBoost model, after tuning, effectively fulfills our targeted objective by placing greater importance on correctly identifying positive instances.

Moreover, the comparison of ROC-AUC values indicates a substantial enhancement in the tuned model, particularly in the test set. The discernible reduction in overfitting underscores that the tuned model achieves a more equitable and dependable performance across both training and test datasets. This advancement is pivotal in ensuring the model's adaptability to new, unseen data, transcending a mere memorization of the intricacies within the training set.

In essence, navigating the trade-off between precision and recall is a customary challenge in machine learning. The decision to prioritize recall in this context has proven advantageous, contributing to an overall improved performance of the tuned XGBoost model. Not only did it successfully alleviate overfitting, but it also heightened its predictive capacities, rendering it a more resilient solution for the designated classification task.

5.3.LightGBM

LightGBM, another robust gradient boosting framework, adopts a unique approach to tree-based learning through its histogram-based method. This technique allows for faster training and efficient handling of large datasets. Similar to XGBoost, LightGBM incorporates a parameter called scale_pos_weight to address imbalanced data, ensuring the model's adaptability to varying class distributions.

5.3.1. First Model With Default Parameters

As a challenger, a default LightGBM model was developed, and its performance metrics are as follows:

Metric	Value
Accuracy	77%
Precision	52%
Recall	**74%**
F1-Score	61%
Roc-AUC-Train	**96%**
Roc-AUC-Test	**85%**

Table 17 - First Run of The Default LightGBM Model

Comparing these results with the default XGBoost model, it's evident that LightGBM achieved a higher Train AUC but a slightly lower Test AUC. This suggests that LightGBM may be more prone to overfitting in the training set, and the XGBoost model, after tuning, could potentially offer a more balanced and robust performance. The next step

5.3.2. Second Model With Default Parameters

The second default model of LightGBM underwent a meticulous feature importance analysis, revealing that 10 variables had negligible impact on the model. Subsequently, these variables were excluded, and the model's performance metrics were reassessed.

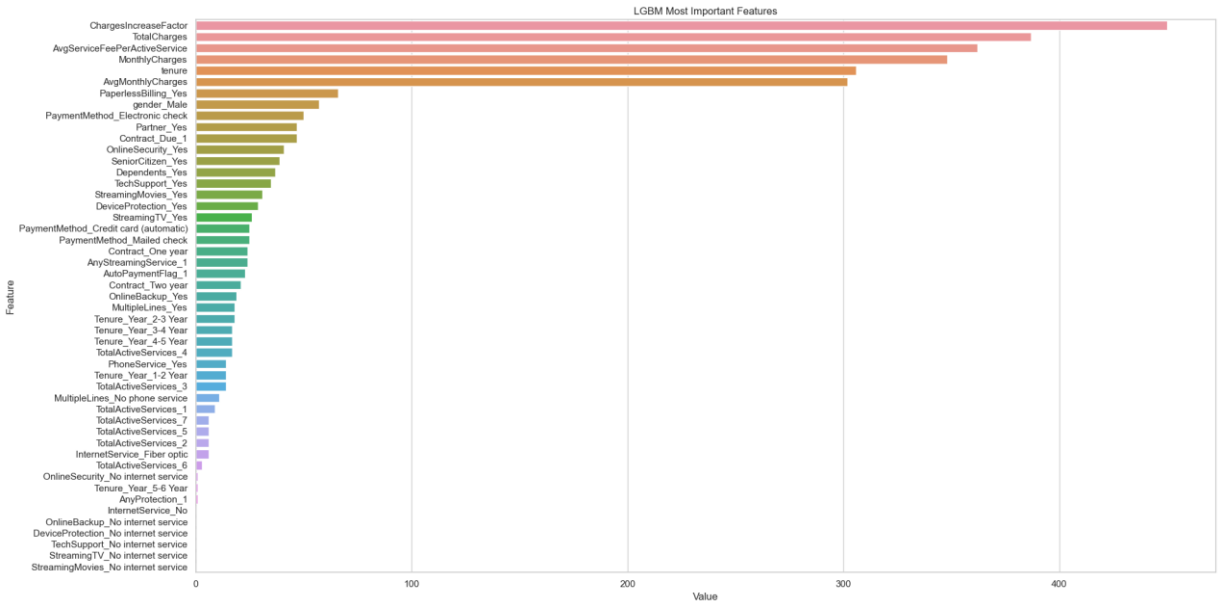


Table 18 LightGBM Feature Importances

Following the exclusion of the identified 10 variables with minimal impact on the LightGBM second default model, a comprehensive evaluation of the model's performance metrics was conducted. The results revealed that the removal of these variables had limited effect on the overall model performance, as indicated by the consistent metrics. The evaluation metrics after this strategic variable removal are presented below:

Metric	Value
Accuracy	77%
Precision	52%
Recall	**75%**
F1-Score	61%
Roc-AUC-Train	**96%**
Roc-AUC-Test	**85%**

Table 19 - Second Run of The Default LightGBM Model

As a result, the LightGBM model was refined to include only the top 39 most influential variables. This strategic feature selection process aimed to enhance the model's efficiency by focusing solely on the key predictors that contribute significantly to its predictive accuracy. The decision to trim down the feature set was guided by the principle of prioritizing the most impactful variables, streamlining the model for improved interpretability and computational efficiency.

The next step will involve hyperparameter tuning for the model to further enhance its respective performances.

5.3.3. Final Model with Hyperparameter Tuning

To unlock the full potential of LightGBM, a powerful technique renowned for its efficiency with large datasets and complex models, strategic parameter tuning is essential. LightGBM, representing 'Light Gradient Boosting Machine,' excels in speed and performance due to its implementation of parallel and gradient-based learning. This makes it particularly advantageous for machine learning applications, especially when dealing with extensive datasets and intricate models.

Similar to XGBoost, LightGBM features parallel processing, contributing to its accelerated training speed compared to traditional Gradient Boosting Machines. This characteristic is pivotal for efficiently handling substantial datasets and intricate model architectures.

LightGBM's adaptability is highlighted by its flexibility in allowing users to define custom optimization objectives and evaluation criteria. This empowers practitioners to tailor the model according to specific use cases, offering a wide range of possibilities without predefined constraints.

The model further simplifies the determination of the optimal number of boosting iterations through built-in cross-validation at each step. This streamlined approach facilitates finding the exact number of boosting iterations necessary for optimal performance in a single run, contributing to a balanced model between complexity and predictive accuracy.

Similar to XGBoost, LightGBM categorizes parameters into three groups, each serving a specific purpose:

- **General Parameters:** These guide the overall functioning of the LightGBM model, influencing high-level characteristics of the boosting process.
- **Booster Parameters:** Responsible for shaping the behavior of individual boosters, whether trees or regression models, at each step of the boosting process.
- **Learning Task Parameters:** These parameters guide the optimization process performed by LightGBM for a specific learning task, allowing customization based on the nature of the data and analysis goals.

Incorporating parameter tuning, with a specific focus on booster parameters, adjustments have been made to key parameters, including:

Parameter	Definition	Final Value
random_state	Random seed for reproducibility	1003
metric	Evaluation metric for binary classification tasks	'auc'
objective	Specifies the learning task and the corresponding objective	'binary'
subsample	Fraction of samples used for each tree	0.8
colsample_bytree	Fraction of features used for each tree	0.8
learning_rate	Rate at which the model adapts during training	0.01
max_depth	Maximum depth of a tree	5

Table 20 - LightGBM Model Hyperparameters Definitions and Final Values After Tuning

The Shapley plot for the tuned LightGBM model can be examined below:

Tuned LightGBM model's SHAP values reveal a change in the orders of some variables. Additionally, while the XGBoost model exhibited a clearer impact of variable values (shown in blue, purple, and red colors), here, certain values seem to intertwine with each other. However, in

terms of interpretability, the LightGBM model is considered sufficient. For further details, refer to the explanation of XGBoost Shapley Values.

Following an extensive hyperparameter tuning process, the LightGBM model underwent substantial enhancements to mitigate overfitting and improve overall performance. The comparison of evaluation metrics between the tuned model and the default model is summarized in the table below:

Metric	First Default Model	Tuned Model
Accuracy	77%	80%
Precision	52%	59%
Recall	**74%**	**65%**
F1-Score	61%	62%
Roc-AUC-Train	**96%**	**87%**
Roc-AUC-Test	**85%**	**86%**

Table 21 - Comparison of Evaluation Metrics on First and Tuned LightGBM Models

The tuned model exhibits various improvements and changes compared to the first default model. Firstly, there are enhancements in several key metrics of the tuned model, but these improvements come with some trade-offs.

In terms of recall, the tuned model shows a decrease of 9%. Recall represents the model's ability to correctly detect positive instances, indicating increased sensitivity. Since the recall value is higher in the default model, it can be said that it predicts more positive examples correctly compared to the tuned model. This outcome is an unexpected result for the tuned model.

However, this decrease comes with a benefit. The precision value has increased in the tuned model (from 52% to 59%). Precision measures how accurate the positive predictions of the model are. This increase indicates an improvement in the model's ability to predict positives.

There is a significant improvement in addressing overfitting, indicating that the tuned model is less prone to overfitting the training data and has increased generalization ability. The model demonstrates a more robust performance by avoiding overfitting.

In conclusion, an unexpected precision-focused improvement has been made, and the improvement in addressing overfitting indicates that the model is a more general and reliable solution.

6. Conclusion

Significant differences are observed between the final models obtained through hyperparameter tuning processes on XGBoost and LightGBM.

Metric	XGBoost Tuned Model	LightGBM Tuned Model
Accuracy	76%	80%
Precision	51%	59%
Recall	**82%**	**65%**
F1-Score	63%	62%
Roc-AUC-Train	**%86**	**87%**
Roc-AUC-Test	**%85**	**86%**

Table 22 - Comparison Between Tuned XGBoost and Tuned LightGBM Models

In the XGBoost model, although there is a slight decrease in precision, there is a significant increase in recall for the tuned model. This notable improvement suggests that the tuned XGBoost model effectively prioritizes correctly identifying positive instances. When comparing ROC-AUC values, a substantial enhancement is observed, particularly in the test set. Moreover, a noticeable reduction in overfitting indicates that the tuned model achieves a more balanced and reliable performance across both the training and test datasets. This advancement is crucial for ensuring the model's adaptability to new, unseen data, rather than merely memorizing intricacies within the training set.

For the LightGBM model, the tuned model shows improvements in several key metrics, but these improvements come with some trade-offs. There is a decrease in recall, indicating a reduction in the model's ability to correctly detect positive instances, and an increase in precision, suggesting improved accuracy in positive predictions. The significant improvement in addressing overfitting indicates that the tuned LightGBM model is less prone to overfitting the training data, demonstrating increased generalization ability.

Overall, it is believed that XGBoost outperforms in addressing the churn problem, justifying its selection for the final model. The strategic emphasis on recall, despite a slight decrease in precision, is intentional. This decision stems from the understanding that retaining existing customers is paramount for sustained business growth, and losing valuable customers may have a more significant impact than potential gains from new acquisitions. The choice of XGBoost, with

its improved recall and overall robust performance, reflects a commitment to effectively addressing the dynamic nature of the churn problem.

7.References

https://xgboost.readthedocs.io/en/stable/

https://lightgbm.readthedocs.io/en/stable/

https://machinelearningmastery.com/xgboost-for-imbalanced-classification/

https://christophm.github.io/interpretable-ml-book/shapley.html

https://www.analyticsvidhya.com/blog/2016/03/complete-guide-parameter-tuning-xgboost-with-codes-python/

https://towardsdatascience.com/building-a-one-hot-encoding-layer-with-tensorflow-f907d686bf39