NHH - Norwegian School of Economics

STR459

ARTIFICIAL INTELLIGENCE AND ROBOTICS

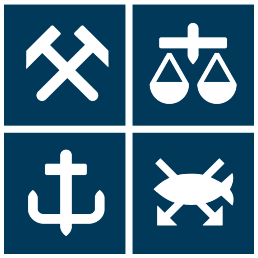
SPRING 2024 EXAM

April 18, 2024

Candidate 1:	*Candidate 2:*	*Candidate 3:*
17	36	48

Candidate 4:	*Candidate 5:*
54	92

Font: Calibri, size 12
Line spacing: 1.5
Word count: 4968
Pages: 26
Report to be used as an example: YES



Contents

1 Introduction

In subscription models with on-demand cancellation, customer retention is vital. These business models inherently rely on the longevity of customer subscriptions to maintain steady revenue streams. Therefore, the ability to retain customers becomes essential for the business's long-term financial health and stability. In this context, being able to accurately point out which customers are at a higher risk of canceling their subscriptions becomes a critical aspect of the overall business model.

This concept is particularly important in the financial services sector, and therefore applies to credit card issuers facing the challenge of predicting credit card churn. The loss of a credit card customer directly impacts revenue and weakens long-term profitability due to the high cost of acquiring new customers compared to retaining existing ones [18].

Therefore, this task seeks to aid businesses operating in the credit card industry by leveraging predictive analytics to analyze patterns in customer behavior, transaction activities, and credit utilization. With these insights and a thorough examination of customer data, companies can identify potential churn signals. This insight allows for targeted retention strategies, such as personalized incentives and improved customer service.

The strategies used in the financial sector to combat credit card churn can prove to be valuable for all subscription-based industries. By leveraging customer data to prevent churn, companies can not only retain valuable customers, but also get key insights to improve overall customer satisfaction. In essence, a proactive approach to customer retention, backed by predictive analytics, is fundamental to securing a stable future for subscription-based businesses.

2 Exploratory Data Analysis

To predict credit card customer churn effectively, our methodology begins with a thorough analysis of the dataset to examine the underlying patterns and behaviors. This is followed by data

preprocessing, which includes cleaning the data, performing feature engineering to improve predictive power, and transforming the data into a format suitable for machine learning algorithms. These actions ensure the dataset is optimized for analysis.

2.1 Introducing The Data

The data used in this task is collected from Kaggle.com, which provided us with data about customer churn in the credit card industry [21]. Before data cleaning and wrangling, the dataframe contained 10,127 data entries and 21 variables. These variables included the target variable `Attrition_Flag` and an unique customer ID represented as `CLIENTUM`, as well as 19 other features presenting various customer attributes. The individual variable descriptions used for this task are presented in Table 1 in the Appendix.

2.2 Data Manipulation

Following reading in the dataframe we observed that it included both numerical and categorical data. To adhere to subsequent analyses we therefore had to turn categorical data into numeric data, which we could analyze with the machine learning algorithms we will introduce later. For the categorical variables `Education_Level`, `Marital_Status`, and `Income_Category`, all contained missing values defined in the dataframe as *Unknown*. We would have to choose if we wanted to remove or impute the *Unknown* values. Although imputation with the mode is common for categorical variables, we decided not to do this as we could skew the data [11]. More sophisticated imputation methods were also considered, where we try to predict the imputation based on other variables; however, we deemed that introducing target-variable prediction in the EDA would be beyond the scope of this task since it will be introduced later. Therefore, removing the *Unknown* instances in categorical variables was the technique we decided upon.

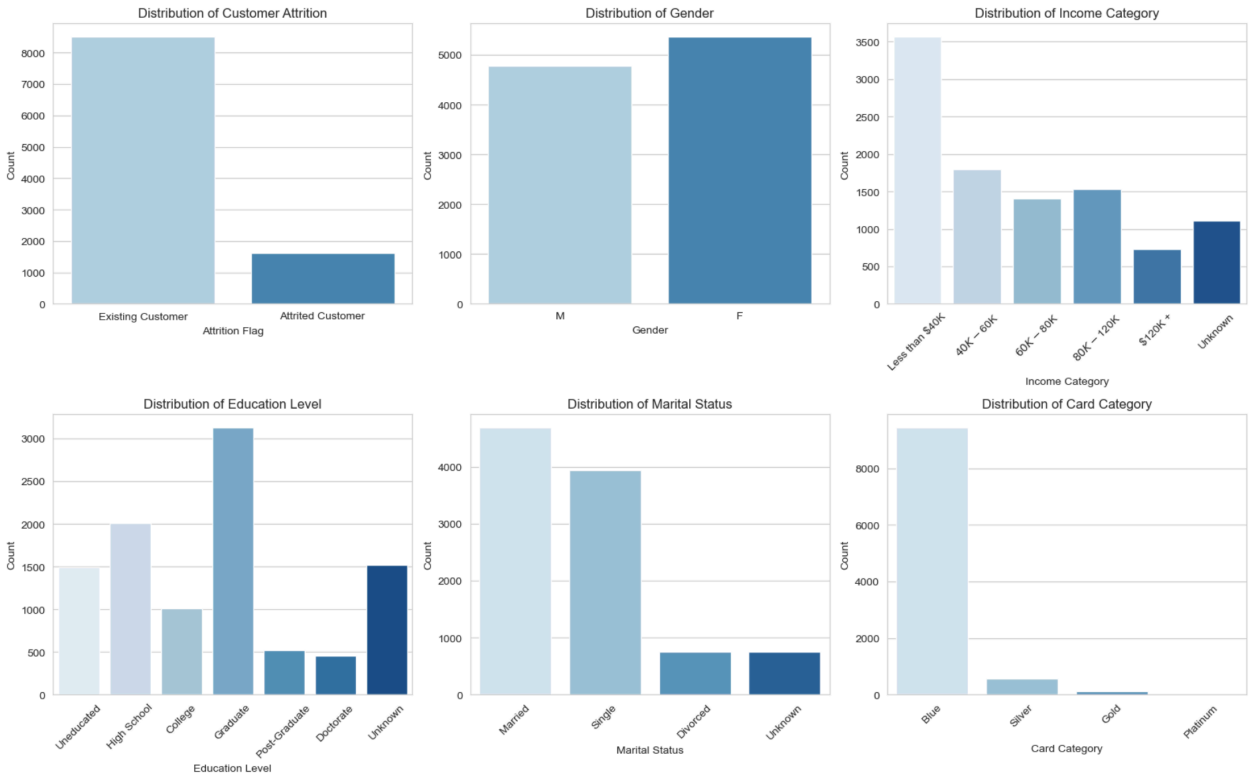


Figure 1: Distributions of Variables

After the removal of missing data, some additional insights regarding the distributions of the variables became evident. Firstly, the churn status is heavily imbalanced, with a much larger majority class of 5,968 existing customers compared to 1,113 attrited customers. The repercussions of how this affects model evaluation and data manipulation will be discussed later in 2.3. Furthermore, we can tell that the categorical variables are not yet suited for prediction as we need to map them to a numerical value. For binary outcomes, we mapped them to values 0 and 1. For ordinal data, we mapped each level to an integer starting from 0. For nominal data, we used one-hot encoding to convert one variable into multiple binary variables. This is to ensure certain categorical values are not interpreted as more valuable than others. We further ensured avoiding the dummy variable trap by dropping the first category for each nominal variable [15].

After transforming the data, we explore central tendencies and numerical descriptors to deepen our understanding of the dataset. This analysis additionally assists in identifying any potential anomalies that could influence our analysis. This can be found in Table 2 in the Appendix.

From the table, we can validate that we have no missing values left in the data, and the recording of categorical variables has been successfully implemented. The consistency of the minimum and maximum values indicates that our dataset is free from any false entries or mistyped values. Nonetheless, we still have to be aware of the potential presence of non-generalizable values that may impact particular analyses.

To further investigate the relationships within our data, creating a correlation matrix will be our next step. The correlation matrix gives us more insights into the interplay between variables, giving us narratives about customer behaviors and their implications on `Attrition_Flag`.

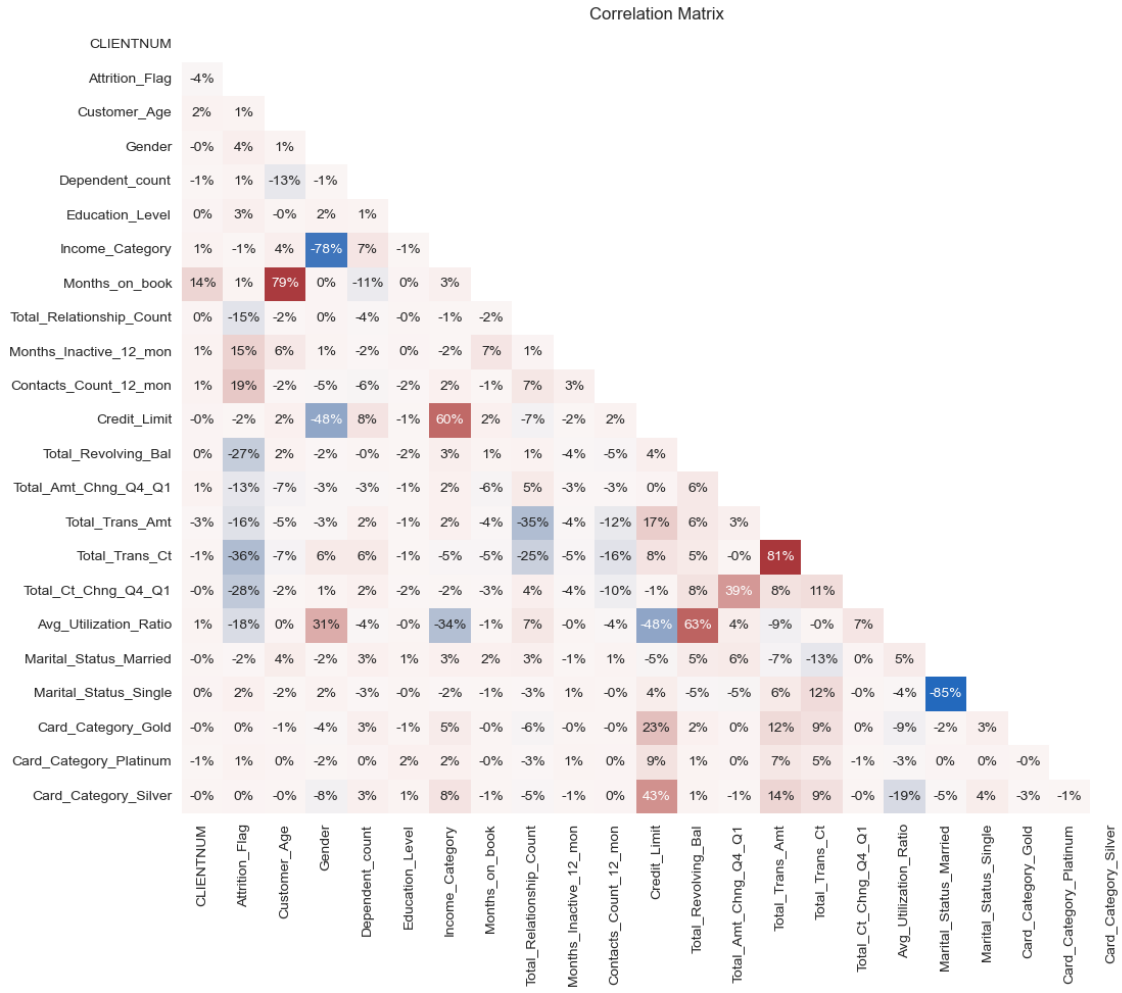


Figure 2: Correlation Matrix

The negative correlation between `Attrition_Flag` and `Total_Trans_Ct` at -0.36, gives us an idea that the tendency of highly transactional customers will stick with the bank. This trend is mirrored in the negative correlation with `Total_Relationship_Count`, which suggests that customers firmly established in a wider array of bank products and services exhibit a reduced likelihood to churn.

On the other hand, we see a positive correlation of 0.19 with `Contacts_Count_12_mon`, which gives us an interesting observation: more frequent interactions with customer service might foreshadow an increased risk of attrition, hinting at underlying dissatisfaction or unresolved issues. This observation specifically calls for a reassessment of the effectiveness of customer service interventions.

We also have variables that are highly or even perfectly correlated. This is something we have to take into account especially when it comes to models that are sensitive to multicollinearity. Additionally, removing `Avg_Open_To_Buy` since it is perfectly correlated with `Credit_Limit` was done as we deemed having both unnecessary.

Transitioning from the insights gained from the correlation matrix, we want to delve deeper into the dynamics of how churn is distributed across various feature variables. By leveraging box-plots to visually represent these relationships, we get a more detailed view of how specific factors correlate with customer churn.

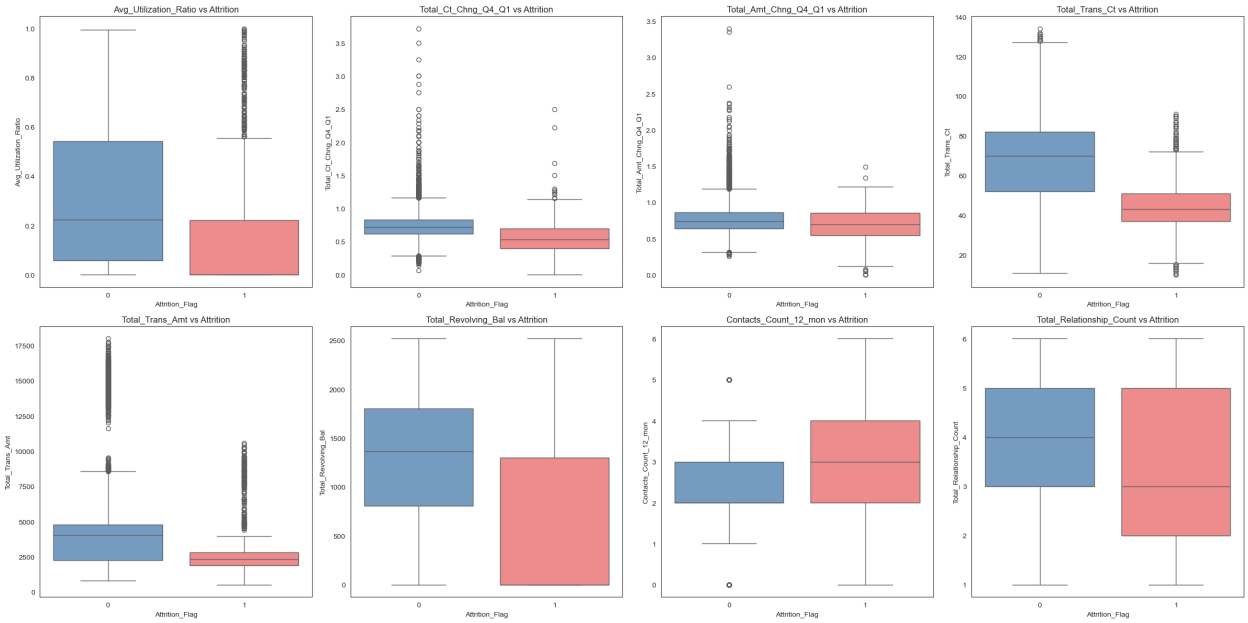


Figure 3: Box-plot

The box-plots reveal differences between churners and non-churners across selected attributes based on the correlation matrix. Notably, these plots also give us hints as to if there are outliers or anomalies within our dataset. Such outliers, while offering insights into extreme cases, might skew the analysis and potentially affect the generalizability of the predictive models we aim to develop. Here, we have to carefully consider the trade-off between generalizability and the data reflecting real-world data entries. One thing to consider is which mechanisms create these outliers. As the rule-of-thumb for this analysis, we follow the commonly used definition of an outlier defined by Hawkins: "An outlier is an observation which deviates so much from the other observations as to arouse suspicions that it was generated by a different mechanism." [2]. Our goal is therefore to refine our dataset in a way that balances the retention of meaningful information with the potential removal of data points that could distort our models' predictions on unseen data entries reflecting real-world data.

In light of this, we wanted to check the distributions of some selected variables where outliers at an univariate level inherently make sense, thus excluding binary and categorical variables.

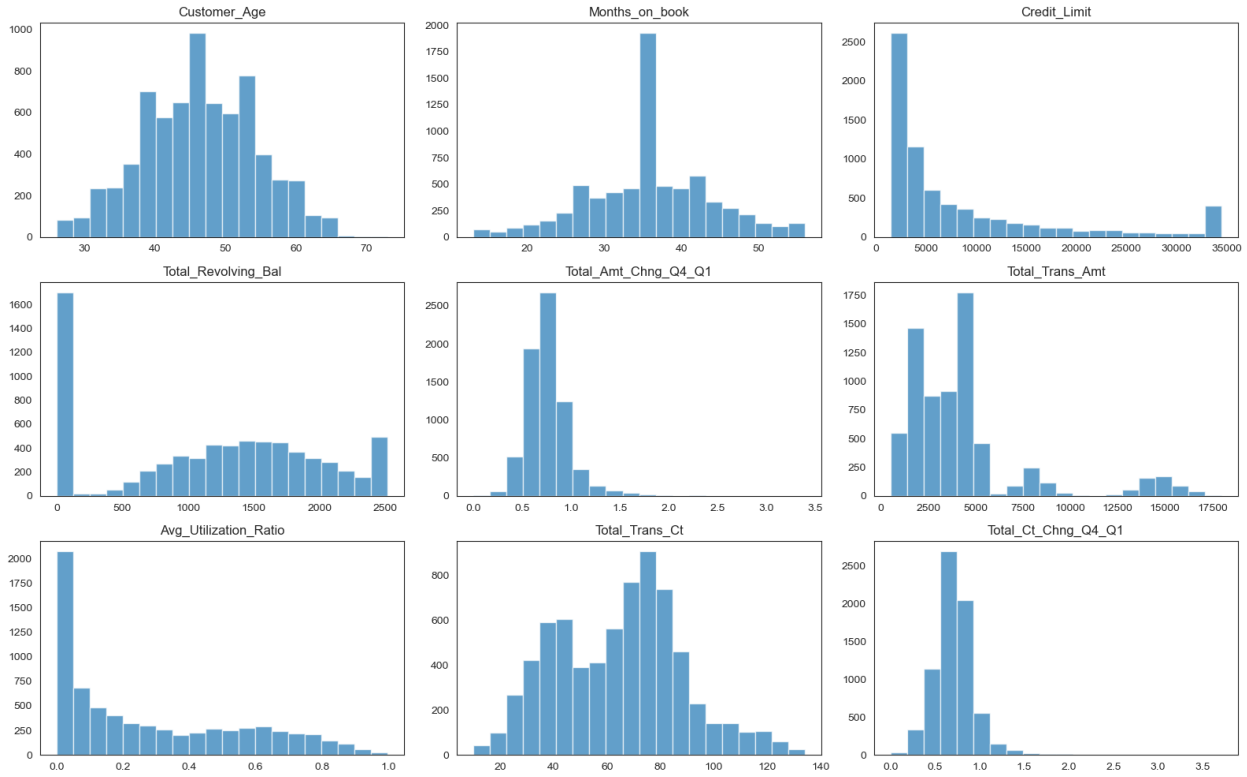


Figure 4: Histograms of numerical variables

The histograms indicate that although some variables display a normal distribution, there are notable deviations in skewness and kurtosis for many others. This suggests that quartile ranges are a suitable method for removing outliers at an univariate level. However, if outlier removal is based on the entire data entry as a whole, multivariate outlier handling algorithms like DBscan or isolation forest would be more appropriate [13].

Histograms provide a clear view of the variables' distributions at an univariate level. However, we can additionally assess the pair-plot below which extends this analysis to expose outliers in both univariate and bivariate relationships. This further visual indication of outliers is useful for examining the relationships between pairs of variables and identifying any data entries that might be caused by unnatural mechanisms. Nonetheless, upon examining the pair plot, we deemed that there were not any data entries created by any unnatural mechanisms and thus needed to be removed. We considered these data entries to hold valuable information that would enhance the model's performance in real-world applications.

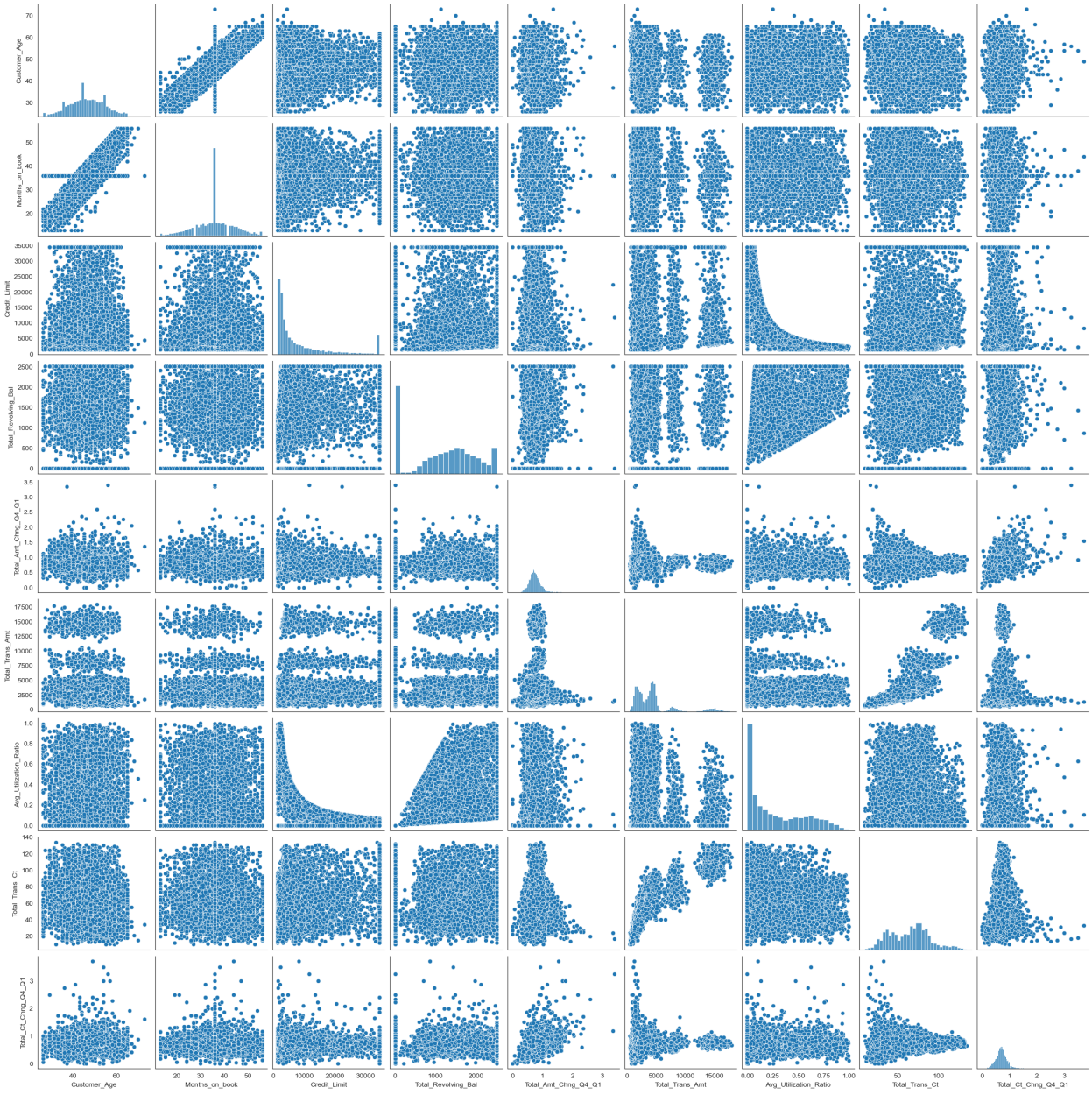


Figure 5: Pairplot

2.3 Splitting The Data

Following a thorough exploratory data analysis and data preparation, we divided our dataset using train-test-split to evaluate the model on unobserved data. As highlighted earlier in the EDA, the target variable `Attrition_Flag` exhibits a significant imbalance. To address this, we employed a resampling strategy to balance the classes within the training data. Specifically, we utilized an oversampling technique known as Synthetic Minority Over-sampling Technique (SMOTE).

This method synthesizes new examples from the minority class, thereby balancing the class distribution without losing important information or introducing significant bias. This step is important to make sure our models do not favor the majority class all the time [4].

To maintain the integrity of our models' predictive power, we confirmed the distributions of the key feature variables for the test and train data followed approximately the same distributions. The key variables were found with a simple non-fine-tuned random forest classifier - an ensemble method of a decision tree classifier less prone to overfitting. With that, we extracted feature importances and plotted histograms comparing test and train data. Here we saw that the oversampling technique came at a trade-off which was that key variable distributions in test and train data were not as closely mirrored. Nonetheless, we applied the SMOTE training data for model evaluation.

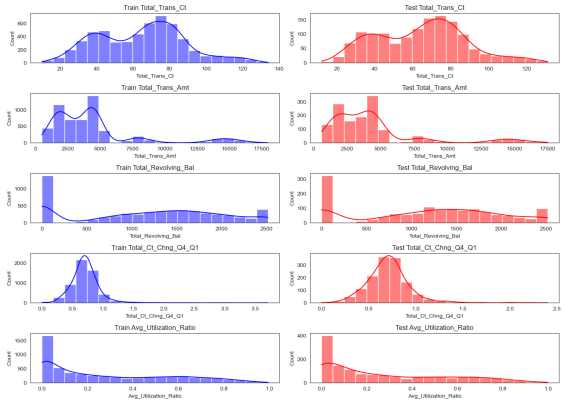


Figure 6: non-SMOTE train-test Comparison

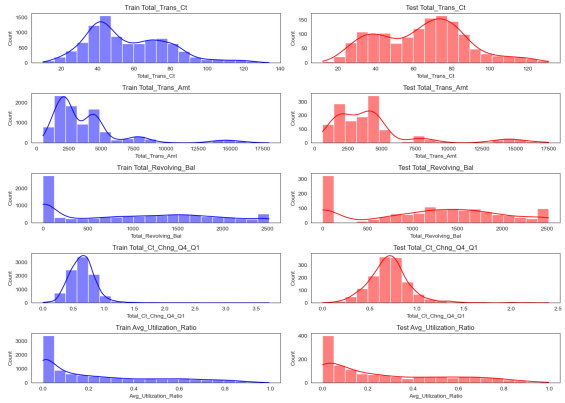


Figure 7: SMOTE train-test Comparison

2.4 Libraries

The following Python libraries are used to manage, visualize, tune, and model our data effectively:

- Pandas: Used for data manipulation and preprocessing, which helped us make the dataset handling and transformation suitable for modeling.
- Matplotlib and Seaborn: Used for data visualization, which were helpful to communicate results more clearly.

- Scikit-learn: This was our primary toolkit for machine learning as a way of evaluating, and deploying a variety of ML algorithms.
- Imblearn: Used for applying Synthetic Minority Over-sampling Technique (SMOTE) to balance class distribution in datasets.
- Optuna: A hyperparameter optimization framework that helped us fine-tune the models by systematically searching for the best hyperparameters, thereby enhancing model performance. Unlike traditional grid-based methods, which exhaustively search through a predefined set of hyperparameters, Optuna uses a Bayesian optimization approach to efficiently explore the search space, focusing on promising areas and significantly reducing the time and computational resources required to achieve desirable results.

3 Building AI Models

This section dives into the theoretical groundwork of three machine learning models used for classification tasks, namely: logistic regression, decision tree classifier, and extreme gradient boosting (XGboost). We will explore the core concepts of these algorithms and discuss the importance of feature engineering and model assumptions that underlie their successful implementation. In this context, we have already done feature engineering in the EDA, however, model-specific prepossessing will be accounted for in this section. Additionally, we will address the aspect of hyperparameter tuning. Hyperparameters are external configuration variables used to manage machine learning model training. They are different from parameters, which are internal parameters automatically derived during the learning process and not set externally [20]. Model-specific feature engineering and hyperparameter tuning are important additions to our strategy to optimize our models.

To compare models, a default and an enhanced model using feature engineering and tuned hyperparameters was created. The default model lays a baseline which we can optimize from. Getting anything worse than the baseline would indicate that some feature engineering processes should be reconsidered, or that the hyperparameter tuning space is poorly defined or not exten-

sive enough.

We aim to provide an overview of the models, by focusing on both the mathematical foundations, as well as the practical considerations. As we proceed, we will discuss the complexities of each model, ensuring a thorough understanding of each model.

3.1 Logistic Regression

3.1.1 Theoretical Overview

Logistic regression is a statistical analysis method used to predict the outcome of a categorical dependent variable based on predictor variables. It models the probability that each input belongs to a particular category, making it a useful tool for binary classification [14].

Central to the logistic regression is the logistic function, which is based on the sigmoid function. The logistic function is crucial for transforming the linear combination of the input variables into a probability that lies between 0 and 1, suitable for binary classification. The logistic function in the context of predicting an event (in our case churn) can be expressed as follows:

$$P(\text{Churn} = 1|X_1, X_2, \ldots, X_n) = \frac{1}{1 + e^{-(b_0 + b_1 X_1 + b_2 X_2 + \cdots + b_n X_n)}}$$

Here, $P(\text{Churn} = 1|X_1, X_2, \ldots, X_n)$ denotes the probability of churn given the customer attributes, b_0 represents the intercept term, and $b_1, b_2, \ldots, b_n$ are the coefficients that reflect the influence of each predictor on the likelihood of churn. The process of logistic regression involves estimating these coefficients b to best fit the model to the observed data, usually through maximum likelihood estimation (MLE) [16].

3.1.2 Model Specific implementations and Hyperparameters

The logistic regression model is sensitive to the scale of features. Large differences in the magnitudes of features can lead to slower convergence because features with larger scales disproportionately influence the model. Therefore, we scale the features to ensure that each feature

contributes approximately equally to the decision boundary. Here, we scale everything except binary variables, as they are already on a meaningful scale (0 and 1) [19].

The logistic regression model is additionally sensitive to variables that correlate with each other, a condition known as multicollinearity. Multicollinearity can make the model's estimates unstable and difficult to interpret. Here, regularization helps by introducing a penalty term to the model's loss function, which constrains the size of the coefficients and can help reduce overfitting. We employ three types of regularization: L1 (Lasso), L2 (Ridge), and Elastic Net. L1 regularization can drive some coefficients to zero, performing feature selection. L2 regularization tends to distribute the penalty across all coefficients, shrinking them but not setting them to zero. Elastic Net is a hybrid approach that combines the penalties of L1 and L2, making it versatile for various scenarios [17].

In terms of hyperparameter tuning, we focused on two parameters: the C parameter, which controls the strength of the regularization, and the 'penalty', associated with 'l1', 'l2', and 'elastic-net' hyperparameters. The choice of regularization affects the selection of the optimization solver, as not all solvers support all types of regularization. Specifically, the 'liblinear' solver does not support elastic-net regularization. Therefore, the solver was dynamically selected based on the chosen regularization method to ensure compatibility in model training.

3.2 Decision Tree Classifier

3.2.1 Theoretical Overview

A decision tree classifier is a supervised machine learning algorithm that segments data into classes based on decision rules, focusing on mathematical optimization to determine the best decision thresholds. It's structured with a root node for initial splits, internal nodes for further classifications, and leaf nodes for final class assignments, aiming for pure leaf nodes where possible. This model directly assigns classes, leveraging impurities to optimize data splits [1].

Mathematically, decision trees aim to minimize gini impurity, calculated as $Gini = 1 - \sum_{i=1}^{C}(p_i)^2$,

where p_i represents the probability of class i within a node, and C is the total number of classes. Similarly, they can use entropy, defined as $Entropy = -\sum_{i=1}^{C} p_i \log_2(p_i)$, to measure disorder or uncertainty, guiding the trees to make splits that result in the highest information gain or the most significant reduction in uncertainty. Both measures can be used for reducing variance among classes within nodes, directing the selection of splits that lead to subsets with lower impurity [7].

3.2.2 Model Specific implementations and Hyperparameters

Despite their straightforward nature and interpretability, decision trees are particularly susceptible to overfitting. This vulnerability has led to many tree-based ensemble methods for more robust models. For instance, the Random Forest Classifier, which was briefly introduced during the exploratory data analysis (EDA) phase for quick feature importance extraction, and XGboost which will be introduced in the next segment.

Due to the tendency of decision trees to overfit, primarily due to their sensitivity to irrelevant or redundant features, we adopted a strategy where the create_study instance in Optuna optimized feature selection as well - in a sense, treating each variable as a hyperparameter. To further safeguard against overfitting, we employed 10-fold cross-validation on the training data throughout the hyperparameter tuning process. Additionally, because of the decision tree's ability to effectively handle data at various levels, processes such as scaling were considered unnecessary [6].

For hyperparameter tuning, we tuned the following hyperparameters:

- *criterion*: Determines the function used to measure the quality of a split, we used options 'gini' and 'entropy'.
- *ccp_alpha*: Controls the cost complexity pruning to trim the tree and avoid overfitting.
- *max_depth*: Sets the maximum depth of the tree to prevent overly complex trees.
- *min_samples_split*: Specifies the minimum number of samples required to split an internal node.
- *min_samples_leaf*: Defines the minimum number of samples that must be present at a leaf node.

In choosing to use a decision tree classifier before progressing to more sophisticated models like XGBoost, we prioritized a strong understanding of the basics. Decision trees are the cornerstone for more advanced tree-based techniques, making them essential to get a good grasp on first. This foundational knowledge is crucial for effectively applying and interpreting more complex models that build upon the decision tree logic.

3.3 Extreme Gradient Boosting

3.3.1 Theoretical Overview

Extreme Gradient Boosting, or XGBoost, is a relatively new advanced supervised machine learning algorithm that enhances the traditional gradient boosting method by introducing a more efficient tree construction technique and integrating regularization parameters to prevent overfitting. This algorithm combines predictions from multiple decision trees to achieve higher accuracy than any individual tree could provide. Unlike conventional decision tree algorithms that may generate trees independently, XGBoost refines predictions iteratively by constructing a sequence of trees, with each one focusing on correcting the errors made by its predecessor. This section aims to delve deeper into the iterative tree-building process of XGBoost, explained through descriptive explanation and mathematical formulation: [5].

From a mathematical view, we can describe the algorithm as an iterative tree construction that relies on calculating the residuals or errors from previous predictions within the training data. These residuals are crucial in computing the similarity scores for observations, determining how trees are subsequently developed, and how to optimally subset the observations based on their characteristics. Similarity scores are calculated through the following formula:

$$\text{Similarity Score} = \frac{\left(\sum_{i=1}^{n} \text{residuals}_i\right)^2}{\sum_{i=1}^{n} P_{i-1} \cdot (1 - P_{i-1}) + \lambda}$$

Here, $Residual_i$ represents the difference between the observed and predicted values for the ith observation, and we use the probability to refine our predictions by assessing the deviation of the preceding guess. It quantifies the necessary adjustment for the subsequent guess, thus improving accuracy. The term λ is the regularization parameter that helps reduce model complexity and

prevent overfitting.

Calculating output values after the creation of each tree is the next step in the iterative process. The output values will be used to calculate the logarithmic odds for the prediction. Output values are calculated using the following formula:

$$\text{Output value} = \frac{\sum_{i=1}^{n} \text{residuals}_i}{\sum_{i=1}^{n} P_{i-1} \cdot (1 - P_{i-1}) + \lambda}$$

Continuing this iterative process, the logarithmic odds are updated by summing the product of the output values from each tree and the learning rate η to the logit of the initial prediction. The learning rate is a factor that controls the contribution of each tree to the final prediction, ensuring that the model learns gradually and avoids overfitting. After updating the odds, the next step involves transforming these log odds back into probabilities, using the formula: $p = \frac{e^{log(odds)}}{1+e^{log(odds)}}$. The process is repeated, with each iteration aiming to reduce residuals and improve the model's predictive performance until a stopping criterion is met [8].

3.3.2 Model Specific implementations and Hyperparameters

XGboost, being a tree-based model, is similar to the decision tree regarding its strengths and flaws. However, XGboost is a more robust model that dynamically adjust parameters in accordance with the data, therefore typically outperforming its decision tree counterpart. Nonetheless, it still can overfit if there are a lot of redundant or unnecessary variables [9]. Therefore, we employ the same method as with the decision tree by essentially treating feature selection as a hyperparameter in itself.

XGboost, being a more complex model overall, also has more hyperparameter options in the tuning process. The variables we have included in our hyperparameter tuning space are:

- *n_estimators*: Number of trees in the model.
- *max_depth*: Maximum depth of each tree.

- *learning_rate*: Adjusts the models sensitivity to the output values from each prediction tree.
- *subsample*: Fraction of samples used for fitting individual trees.
- *colsample_bytree*: Fraction of features used for each tree.
- *min_child_weight*: Minimum weight needed in a child node to keep splitting.
- *reg_alpha*: L1 regularization on weights.
- *reg_lambda*: L2 regularization on weights.

4 Evaluating AI Models

4.1 Evaluation Metrics

When considering evaluation metrics for machine learning models, the nature of what is valuable for the business is an important aspect. As discussed by Sabbeh [18], customer retention is 5 - 20 times more cost-effective than acquiring new customers. This highlights the importance of proper evaluation metrics when predicting which customers will churn or not.

Since customer churn is disproportionately costly compared to retention, choosing the correct evaluation metrics is vital to actually derive business value from the models. Given the imbalanced nature of the target variable, `Attrition_Flag`, we recognized that solely relying on accuracy might not provide the best measure of a model's performance, as it could bias the results towards the majority class, which is 0 (not churning). Even if our models were to predict only 0, the accuracy score would still be high because the majority of cases are 0 in the test data. This phenomenon is known as the Accuracy Paradox [10].

Considering the Accuracy Paradox, where high accuracy may not reflect true business value in our scenario, Recall Score, also known as Sensitivity or True Positive Rate (TPR) is a valuable metric. This can be defined as the fraction of true positive predictions among all actual positives in the data. Like mentioned previously, losing customers is costly, therefore, we rather want to predict non-churning customers as churning, than churning customers as non-churning. The formula for Recall Score can be defined as $\text{TPR} = \frac{\text{TP}}{\text{TP}+\text{FN}}$, where TP is the True Positives and FN is the False

Negatives.

The True Positive Rate (TPR) offers a useful evaluation metric. However, a more complete assessment of a model's limitations emerges when we consider both the TPR and the False Positive Rate (FPR). FPR can be defined as FPR $= \frac{FP}{FP+TN}$, where FP are False Positives and TN are True Negatives. In our business scenario, the balance between TPR and FPR is extremely vital in order to derive business value. Therefore, by presenting ROC-AUC as an evaluation metric, we are able to see how high we can get TPR, given a certain FPR, for all models. The ROC (Receiver operating characteristic) curve is a graphical representation that plots TPR against the FPR at various thresholds, and AUC is the area under the curve, which we are maximizing.

4.2 Cross-Validation

4.2.1 Train-test split

To estimate the performance of machine learning models, we use cross-validation. Like mentioned in the EDA, we splitted our data into a train set and a test set. By doing this, we can assess the models' performance by having the model perform on new, unseen data. To do this, we used the *train_test_split* method from the Sklearn library. One of the main concerns of Cross Validation is Data Leakage. Data leakage occurs when data from the train set is contaminated over to the test data, causing overly optimistic and misleading model performance [3].

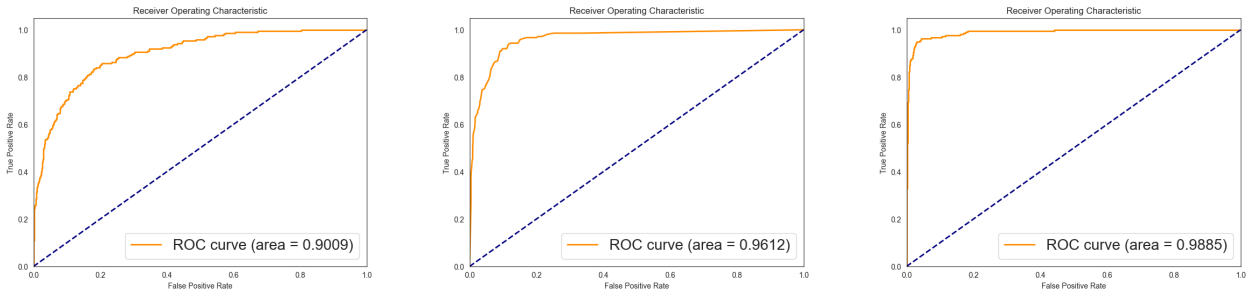
4.2.2 K-fold Cross-Validation

Beyond using a train-test split, we incorporated k-fold cross-validation into our model training approach, to ensure optimal hyper parameter tuning. This technique involves dividing the training data into 'k' subsets, where k = 10. During the cross-validation process, the model is trained on 'k-1' (9) of these subsets and validated on the remaining one, a procedure that is repeated 'k' (10) times with each subset serving as the validation set exactly once. By using Sklearn's cross_val_score function, we were able to find the optimal hyper parameters for each model, to ensure maximum AUC that does not have heavily overfitted hyperparamters.

4.3 Model Performance

The AUC score, as well as the confusion matrix, will be used to evaluate the performance of our three machine learning models.

4.3.1 Area Under the Curve



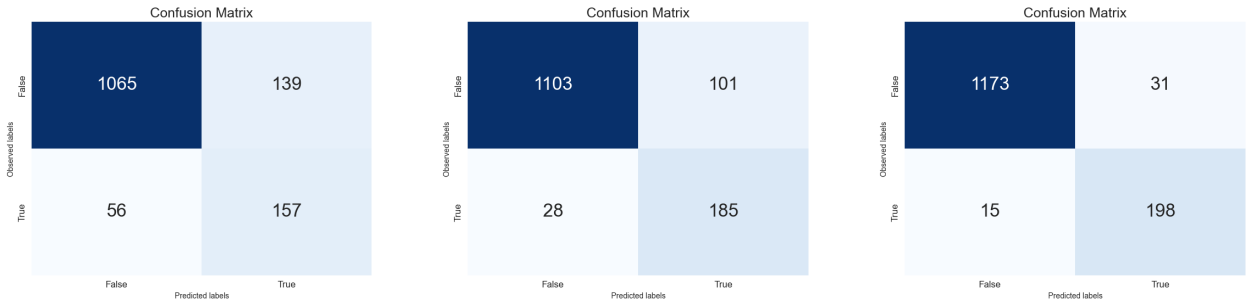
(a) Logistic Model (b) Decision Tree Classifier (c) XGBoost Model

Figure 8: Receiver Operating Characteristic (ROC) curves for different models

By analyzing the ROC curves of the logistic model, decision tree classifier, and XGBoost model, we can gain insights into their varying classification performance. The AUC of the Logistic Model is 0.9009, suggesting good predictive ability. The Decision Tree Model had an AUC of 0.9612, which is better than the logistic model, indicating a moderately strong capability in correctly classifying churn. The XGBoost model, with an AUC of 0.9885, has excellent performance. This not not surprising considering the XGBoost model has been a go-to algorithm for winning solutions in many Kaggle competitions, especially for classification-type problems [12].

4.3.2 Confusion matrix

The confusion matrices derived from the logistic model, decision tree classifier, and XGBoost model, present varying levels of classification ability. The logistic model has the highest count of false negatives and false positives, aligning with its lower AUC. This suggests that it will not classify instances as well as the other models. The Decision Tree Classifier was more accurate since it had fewer False Negatives than the Logistic model, again indicating good predictive power. The confusion matrix of the XGBoost model testifies to its predictive performance since it has very low numbers of False Negatives and False Positives, further asserting its robust nature mirroring the



(a) Logistic Model (b) Decision Tree Classifier (c) XGBoost Model

Figure 9: Confusion matrix for different models

result of the high AUC seen earlier. To be specific, the XGBoost model has only 15 False Positives out of 213 positive cases, translating to a recall value of approximately 93%.

4.4 Added Business Value

4.4.1 Insights

By leveraging Machine Learning methods to predict potential churning customers, we gain the ability to proactively address customer attrition. This will allow for a system where we can enhance customer retention by enabling the credit card company to pin-point which customers are at risk of attiring. This not only enhances operational efficiency in terms of classifying customer segments, but also helps understand the business's flaws and strengths. This will in return provide valuable information and actionable insights.

4.4.2 Measures to prevent churn

The information obtained from the churn prediction models offers useful information to the credit card business, from both a marketing and product development perspective. Although the feature importance of tree-models like XGBoost does not natively highlight the direction of the variables that affect churn we can decipher it by looking at the correlation matrix. From our XGBoost model, we can see that `Total_Trans_Ct` and `Total_Relationship_Count` are the most important features for predicting churn. Therefore, it seems that being a frequent user of the services provided in addition to being entangled in multiple services increases loyalty. A good measure to decrease

customer attrition would be to incentivize existing customers to extend their product portfolio within the company. Furthermore, by highlighting customers who are potential churners, we can deliver precise, personalized marketing - for example, offers on additional products, or newsletters highlighting the advantages of using their credit card.

In practice, we would implement a data pipeline within the business where customer data is regularly fed to the XGBoost model for continuous predictions. The output of the model should be actionable. This means setting up a system where the churn predictions are translated into lists or flags in Customer Relationship Management (CRM) systems, highlighting customers at risk of churning. For example, when a customer is flagged as at risk of churning, a tailored retention offer could be automatically generated, or a customer service representative could be prompted to reach out, personally offering discounts or special offers.

4.4.3 Value Proposition

The measures discussed do not come without costs and as a value proposition to a credit card company we lay out the following ground-work where a business' Customer Retention Costs (CRC) and Customer Acquisition Costs (CAC) can be accounted for. By doing this we can provide a quantitative proposal of added value to the credit card business. Here we create a formula that reflects the added value of the analysis performed for this task, we express the cost savings as:

$$\text{Cost Savings} = (\text{CAC} - \text{CRC}) \cdot \text{TP} \cdot \alpha \tag{1}$$

Here the parameter α represents the success-rate of outreach efforts in converting a churning customer into a retained one. While False Negatives also incur a hidden cost due to the necessity of offsetting them with customer acquisition costs, this factor is not considered in our formula. This is because our baseline scenario assumes the absence of any customer retention strategies, where such compensatory costs would be incurred in any case.

In addition to cost savings we have associated added costs because of the expense linked to reaching out to a customer or giving them an incentive to retain their subscription. We can express this

as follows:

$$\text{Added Costs} = \text{CRC} \cdot \text{FP} \tag{2}$$

Together these formulas form added value to the business expressed as:

$$\text{Added Value} = \text{Cost Savings - Added Costs} \tag{3}$$

This approach quantitatively creates a framework the credit card business can apply to determine the financial benefits of implementing targeted customer retention strategies with the use of machine learning.

5 Summary Of Findings

The goal for this project has been to provide key insights to a credit card company by applying machine learning. We have done this by carrying out an extensive exploratory data analysis followed by model specific feature engineering and hyperparameter tuning to create the best performing models. To provide business value to the credit card company we carefully considered the evaluation metrics we optimized towards. Here, the AUC emerged as the metric that was best suited for the value proposition we worked towards throughout this analysis. Overall, the XGboost model performed the best on this metric while additionally giving us key insights to what customer attributes influenced attrition. This model is therefore proposed to be used in a CRM pipeline where the credit card company continuously can monitor their customer portfolio, being alerted of potential churners. With this knowledge we created a framework from which we can track the added value this project provides the business. Overall, this comprehensive project forms a guide to how a subscription based business can address the crucial aspect of churning customers.

References

[1] *1.10. Decision Trees*. en. URL: https://scikit-learn/stable/modules/tree.html.

[2] C.C. Aggarwal. "An Introduction to Outlier Analysis". In: *Outlier Analysis*. Cham: Springer, 2017. DOI: 10.1007/978-3-319-47578-3_1. URL: https://doi.org/10.1007/978-3-319-47578-3_1.

[3] Jason Brownlee. *How to Avoid Data Leakage When Performing Data Preparation*. en-US. June 2020. URL: https://machinelearningmastery.com/data-preparation-without-data-leakage/.

[4] Nitesh V. Chawla et al. *SMOTE: Synthetic Minority Over-sampling Technique*. Tech. rep. Department of Computer Science, Engineering, University of South Florida; Department of Computer Science, and Engineering, University of Notre Dame; Biosystems Research Department, Sandia National Laboratories, 2002.

[5] Tianqi Chen and Carlos Guestrin. "XGBoost: A Scalable Tree Boosting System". In: *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. arXiv:1603.02754 [cs]. Aug. 2016, pp. 785–794. DOI: 10.1145/2939672.2939785. URL: http://arxiv.org/abs/1603.02754.

[6] *Cross-Validation in Machine Learning - Javatpoint*. URL: https://www.javatpoint.com/cross-validation-in-machine-learning.

[7] Shailey Dash. *Decision Trees Explained — Entropy, Information Gain, Gini Index, CCP Pruning..* en. Nov. 2022. URL: https://towardsdatascience.com/decision-trees-explained-entropy-information-gain-gini-index-ccp-pruning-4d78070db36c.

[8] Little Dino. *XGBoost (Classification) in Python*. en. July 2022. URL: https://medium.com/@24littledino/xgboost-classification-in-python-f29cc2c50a9b.

[9] Christina Ellis. *XGBoost overfitting*. en-US. Aug. 2022. URL: https://crunchingthedata.com/xgboost-overfitting/.

[10] F. Fischer. "The Accuracy Paradox of Algorithmic Classification". In: (May 2019). Ed. by G. Getzinger and M. Jahrbacher, pp. 105–120. DOI: 10.3217/978-3-85125-668-0-07. URL: https://doi.org/10.3217/978-3-85125-668-0-07.

[11] Chirag Goyal. *How to Handle Missing Values of Categorical Variables*. en-US. Aug. 2023. URL: https://www.analyticsvidhya.com/blog/2021/04/how-to-handle-missing-values-of-categorical-variables/.

[12] *How The Kaggle Winners Algorithm XGBoost Algorithm Works - Dataaspirant*. en-US. Nov. 2020. URL: https://dataaspirant.com/xgboost-algorithm/.

[13] Idil Ismiguzel. *Outlier Detection with Simple and Advanced Techniques*. en. Dec. 2022. URL: https://towardsdatascience.com/detecting-outliers-with-simple-and-advanced-techniques-cb3b2db60d03.

[14] Vijay Kanade. *Logistic Regression: Equation, Assumptions, Types, and Best Practices*. en-US. 2022. URL: https://www.spiceworks.com/tech/artificial-intelligence/articles/what-is-logistic-regression/.

[15] Fatih Karabiber. "Dummy Variable Trap". In: (2022).

[16] Marco Toboga. *Logistic regression - Maximum likelihood estimation*. 2021. URL: https://www.statlect.com/fundamentals-of-statistics/logistic-model-maximum-likelihood.

[17] Anuja Nagpal. *L1 and L2 Regularization Methods*. en. Oct. 2017. URL: https://towardsdatascience.com/l1-and-l2-regularization-methods-ce25e7fc831c.

[18] Sahar F. Sabbeh. "Machine-Learning Techniques for Customer Retention: A Comparative Study". In: *International Journal of Advanced Computer Science and Applications* 9.2 (2018), p. 273. URL: https://thesai.org/Downloads/Volume9No2/Paper_38-Machine_Learning_Techniques_for_Customer_Retention.pdf.

[19] Turing. *Effects of Normalization Techniques on Logistic Regression*. en. URL: https://www.turing.com/kb/effects-of-normalization-techniques-on-logistic-regression-in-data-science.

[20] *What is Hyperparameter Tuning?* URL: https://aws.amazon.com/what-is/hyperparameter-tuning/.

[21] Zhyli. *Prediction of Churning Credit Card Customers [Data set]*. https://doi.org/10.5281/zenodo.4322342. Also available at: https://www.kaggle.com/datasets/sakshigoyal7/credit-card-customers. 2020. DOI: 10.5281/zenodo.4322342.

A Appendix

Column/Variable	Description
CLIENTNUM	Integer value uniquely identifying each customer.
Attrition_Flag	Boolean value presented as a string that indicates whether or not the customer has churned out.
Customer_Age	Integer value representing the age of the customer.
Gender	String value denoting the gender of the customer.
Dependent_count	Integer value indicating the number of dependents the customer has.
Education_Level	String (Categorical/Ordinal) value representing the education level of the customer.
Marital_Status	String (Categorical/Nominal) value denoting the marital status of the customer.
Income_Category	String (Categorical/Ordinal) value indicating the income category of the customer.
Card_Category	String (Categorical/Nominal) value representing the type of card held by the customer.
Months_on_book	Integer value indicating how long the customer has been with the bank.
Total_Relationship_Count	Integer value representing the total number of relationships the customer has with the credit card provider.
Months_Inactive_12_mon	Integer value denoting the number of months the customer has been inactive in the last twelve months.
Contacts_Count_12_mon	Integer value indicating the number of contacts the customer has had in the last twelve months.
Credit_Limit	Floating point number representing the credit limit of the customer.
Total_Revolving_Bal	Integer value indicating the total revolving balance of the customer.
Avg_Open_To_Buy	Floating point number denoting the average open to buy ratio of the customer.
Total_Amt_Chng_Q4_Q1	Floating point number indicating the change in transaction amount, comparing Q4 to Q1.
Total_Trans_Amt	Integer value representing the total amount of transactions made by the customer.
Total_Trans_Ct	Integer value indicating the total count of transactions made by the customer.
Total_Ct_Chng_Q4_Q1	Floating point number denoting the change in transaction count, comparing Q4 to Q1.
Avg_Utilization_Ratio	Floating point number representing the average ratio of the line of credit used by the customer.

Table 1: Variable Description

Feature	Mean	Std	Min	Max	Unique	Missing
CLIENTNUM	739091922.52	36852441.97	708082083	828298908	7081	0
Attrition_Flag	0.16	0.36	0	1	2	0
Customer_Age	46.35	8.04	26	73	45	0
Gender	0.48	0.50	0	1	2	0
Dependent_count	2.34	1.29	0	5	6	0
Education_Level	2.07	1.40	0	5	6	0
Income_Category	1.34	1.36	0	4	5	0
Months_on_book	35.98	8.00	13	56	44	0
Total_Relationship_Count	3.82	1.54	1	6	6	0
Months_Inactive_12_mon	2.34	1.00	0	6	7	0
Contacts_Count_12_mon	2.45	1.10	0	6	7	0
Credit_Limit	8492.77	9126.07	1438.3	34516	4654	0
Total_Revolving_Bal	1167.50	812.32	0	2517	1821	0
Avg_Open_To_Buy	7325.27	9131.22	3	34516	5144	0
Total_Amt_Chng_Q4_Q1	0.76	0.22	0	3.4	1067	0
Total_Trans_Amt	4394.30	3468.46	510	17995	4194	0
Total_Trans_Ct	64.50	23.81	10	134	124	0
Total_Ct_Chng_Q4_Q1	0.71	0.24	0	3.71	771	0
Avg_Utilization_Ratio	0.28	0.28	0	1	946	0
Marital_Status_Married	0.50	0.50	0	1	2	0
Marital_Status_Single	0.42	0.49	0	1	2	0
Card_Category_Gold	0.01	0.11	0	1	2	0
Card_Category_Platinum	0.00	0.04	0	1	2	0
Card_Category_Silver	0.06	0.23	0	1	2	0

Table 2: Summary Statistics