TABLE 1. VARIABLE DESCRIPTIONS.

Variable Name	Description
CLIENTNUM	Client number. Unique identifier for the customer holding one credit card
Attrition_Flag	This is the class variable or the target variable. If the account is closed then 1 else 0
Customer_Age	Customer's age in years.
Gender	Customer's gender
Dependent_Count	Number of dependents
Education_Level	Educational qualification of the account holder. For example, high school, college graduate, etc
Marital_Status	Married, Single, Divorced, Unknown
Income_Category	Annual income of the account holder
Card_Category	Type of the card that the customer holds or has held
Months_on_book	Period of relationship with the bank
Total_Relationship_Count	Total number of products of the bank held by the customer
Months_Inactive_12_mon	Number of months inactive in the last 12 months
Contacts_Count_12_mon	Number of contacts in the last 12 months
Credit_Limit	Credit limit on the credit card
Total_Revolving_Bal	Total revolving balance on the credit card
Avg_Open_To_Buy	The average of open to buy credit line in the last 12 months
Total_Amt_Chng_Q4_Q1	Change in transaction amount from Q4 to Q1
Total_Trans_Amt	Total transaction amount in the last 12 months
Total_Trans_Ct	Total transaction count in the last 12 months
Total_Ct_Chng_Q4_Q1	Change in transaction count Q4 over Q1
Avg_Utilization_Ratio	Average card utilization ratio over the last 12 months

2.2. Data Preprocessing

Data preprocessing is a crucial step to transform raw data into a format appropriate for machine learning modeling. The dataset used in this work is of decent quality. It does not have any missing values or duplications. Our data preprocessing focuses on three tasks. First, convert each categorical variables to binary variables. Second, split the dataset into a training set (80%) and a test set (20%), and then standardize all numerical variables. Third, we utilize SMOTE technique to resample the minority class (churns).

There are five categorical variables in the dataset.

- Gender: It has two categories: F or M.
- Education_Level: it has seven categories: 'College', 'Doctorate', 'Unknown', 'Graduate', 'High School', 'Post-Graduate', and 'Uneducated'.
- Marital_Status: it has four categories: 'Divorced', 'Unknown', 'Married', and 'Single'.

- Income_Category: it has six categories: '$120K +', '$40K - $60K', '$60K - $80K', '$80K - $120K', 'Unknown', and 'Less than $40K'.
- Card_Category: it has four categories: 'Blue', 'Silver', 'Gold' and 'Platinum'.

If k is the number of categories in a categorical variable, we convert this variable into k-1 binary variables. The only exception is 'Card_Category'. There are only 20 'Platinum' and 116 'Gold', out of 10,127 data points. We decided to merge 'Platinum' and 'Gold' into 'Gold+'. As a result, the 5 categorical variables in the original dataset become 17 binary variables.

A standard 80/20 split is then performed on the dataset. The training set has 8,101 records and the test set has 2,026 observations. We use StandardScaler from scikit-learn to standardize all 14 numerical variables in the training set into their corresponding z-values. Using the means and standard deviations of the numerical variables derived from the training set, we then standardize all the numerical variables in the test set. This procedure helps us avoid data leakage. The main reason for standardization is that regularization is applied by default in scikit-learn's logistic regression to avoid overfitting.

The original dataset has 1627 attrited or churned customers (positive and minority class) and 8,500 existing customers (negative and majority class). The churned customers account for 16.07% of all customers. The dataset is thus imbalanced between two classes. We choose Synthetic Minority Oversampling Technique (SMOTE) to address this issue. SMOTE is a widely used algorithm for handling imbalanced dataset and is proposed by Chawla et al. (2002). We apply SMOTE to oversample the minority class in the training set and this approach effectively forces the decision region of the minority class to become more general. We end up with the same number of positive examples as the number of negative examples.

At the end of data preprocessing, we have a resampled training set and a test set with 2,026 examples. Each set has 17 binary features and 14 standardized numerical features.

III. QUANTIFICATION OF IMPACT ON CUSTOMER CHURN

3.1. Mathematical Model of Logistic Regression Used in Our Study

Logistic regression is a standard algorithm in customer churn prediction problem because of its ability to produce decent and robust results, as well as excellent interpretability (Neslin et al. 2006, Coussement et al. 2017). We study and quantify the impact of different factors on customer churn using logistic regression, which is widely used in both academia and industry for its unique interpretability in classification problems. The version of logistic regression we use optimizes the following cost function:

$$\min_{w} C \sum_{i=1}^{n} [-y_i \log \hat{p}_i - (1 - y_i) \log(1 - \hat{p}_i)] + r(w).$$

The notations are defined as below:

- n: the number of examples or observations in the training set;
- y_i: the target value of the i^{th} observation. It is 1 or positive if the observation is a churned customer; 0, otherwise;
- w: the coefficient vector in logistic regression model; w_0, a real number not included in w, is the intercept term;
- $\hat{p}_i$: the predicted probability that the i^{th} observation is a churn;
- $r(w)$: a regularization term. In our work, we only consider ℓ_2-penalty and $r(w) = \frac{1}{2}\|w\|_2^2 = \frac{1}{2}w^T w$.
- C: a regularization constant, which is set to 1 for our study.

If x_i is the feature vector of the i^{th} observation, the predicted probability that the i^{th} observation is in positive class is:

$$\hat{p}_i = \frac{e^{(w \cdot x_i + w_0)}}{1 + e^{(w \cdot x_i + w_0)}}.$$

The odds that the i^{th} observation is in positive class is

$$\frac{\hat{p}_i}{1-\hat{p}_i} = e^{(w \cdot X_i + w_0)}.$$

The linear component represents the log-odds or logit:

$$\log\left(\frac{\hat{p}_i}{1-\hat{p}_i}\right) = w \cdot X_i + w_0.$$

Note that $\hat{p}_i^{y_i}(1-\hat{p}_i)^{1-y_i}$ is the probability that the model makes a correct prediction on the i^{th} observation. $\prod_{i=1}^{n} \hat{p}_i^{y_i}(1-\hat{p}_i)^{1-y_i}$, called likelihood, is the joint probability of making correct predictions on all observations. The key expression in the cost function, $\sum_{i=1}^{n}[-y_i \log \hat{p}_i - (1-y_i)\log(1-\hat{p}_i)]$, is thus the negative log-likelihood. The logistic regression model in our study therefore minimizes the regulated negative log-likelihood.

The purpose of regularization is to battle overfitting, a typical phenomenon in machine learning models. Adoption of regularized logistic regression enables us to corroborate the robustness of our logistic regression model such that the interpretation and the measurement of different factors' impacts on customer churn are more reliable.

TABLE 2. LOGISTIC REGRESSION MODEL WITH WALD TEST.

Feature	**Coef.**	**Std. Err.**	**z**	**P>|z|**	**95% CI**	
'Total_Trans_Ct'	-1.897	0.060	-31.510	0.000	-2.015	-1.779
'Total_Revolving_Bal'	-0.512	nan	nan	nan	nan	nan
'Total_Trans_Amt'	1.039	0.057	18.176	0.000	0.927	1.151
'Total_Relationship_Count'	-0.532	0.033	-15.912	0.000	-0.597	-0.466
'Total_Ct_Chng_Q4_Q1'	-0.545	0.036	-15.206	0.000	-0.615	-0.474
'Months_Inactive_12_mon'	0.362	0.030	12.214	0.000	0.304	0.420
'Contacts_Count_12_mon'	0.420	0.031	13.355	0.000	0.358	0.481
'Total_Amt_Chng_Q4_Q1'	-0.108	0.032	-3.332	0.000	-0.172	-0.044
'M'	-2.674	0.067	-40.014	0.000	-2.805	-2.543
'Credit_Limit'	0.192	nan	nan	nan	nan	nan
'Avg_Utilization_Ratio'	-0.131	0.051	-2.552	0.011	-0.232	-0.030
'Dependent_count'	0.080	0.030	2.635	0.008	0.020	0.139
'Single'	-1.285	0.059	-21.821	0.000	-1.400	-1.170
'Customer_Age'	-0.084	0.031	-2.751	0.006	-0.144	-0.024
'Avg_Open_To_Buy'	0.238	nan	nan	nan	nan	nan

3.2. Wald Test

Wald statistics are used to test the significant of the features. Then a variable selection is followed to remove insignificant features in steps, to achieve acceptable prediction performance. We first run a full model with all the features. Based on Wald statistics and p-value, we remove the features that are not significantly associated with the output (95% confidence interval). After several selection steps, we fine tune the logistic regression model with 15 features, which all pass the Wald Test. The results is presented in Table 2. The 'NaN' in the regression output is because, the fitted probability of 0 or 1 creates infinite values on the logit scale. It then propagates through all the other calculations, resulting in 'NaN'. This actually indicates that 'Total_Revolving_Bal', 'Credit_Limit', and 'Avg_Open_To_Buy' have strong relationships with customer churn.

3.3. Feature Selection

A feature selection framework consists a search engine used to determine the subset candidates, and a criterion used (Kotsiantis, 2011). When one feature does not affect describing the target, a redundant feature or a feature with high correlation does not add anything new to describe the target. This is the basic concept of using feature selection in the machine learning model. Kotsiantis (2011) states 4 main benefits of feature selection: i) reducing the cost of measurement and storage; ii) coping with the degradation of the classification performance because of finite training set; iii) reducing training and utilization time, and iv) improving data understanding and data visualization.

After data preprocessing, we have 31 features, some of which are highly correlated. Feature selection is thus in order. We consider two approaches in scikit-learn to feature selection: (1) SelectKBest and (2) permutation_importance.

Scikit-learn's SelectKBest selects features according to the k highest scores. The scoring function we used is the default: f_classif, which is the ANOVA F-value between label and feature. We tried three different k-values: 10, 15, and 20. Based on AUC, area under the ROC curve, we have found that the logistic regression model with k = 15 (AUC = 0.8340) outperforms when k = 10 (AUC = 0.8289) or when k = 20 (AUC = 0.8266).

Permutation importance measures feature importance by observing the effect of randomly shuffling each feature on model accuracy. It is defined to be the decrease in a model score (accuracy in this work) when a single feature value is randomly shuffled (Breiman, 2001). The feature permutation importance algorithm starts with a trained machine learning model. It then calculates the baseline prediction accuracy on the dataset. Next, it randomly shuffles each feature column in the dataset and calculates the prediction accuracy on the shuffled dataset. The difference between the baseline accuracy and the shuffled dataset accuracy is the feature importance score. The algorithm usually repeats the previous step multiple times and reports the average difference. Based on importance score of each feature, the algorithm can rank all the features in importance. Scikit-learn's permutation_importance function enables us to calculate feature permutation importance efficiently.

We select Extra Trees Classifier as the machine learning model fed into permutation importance function. Extra Trees Classifier, formally Extremely Randomized Trees Classifier, is a type of ensemble learning technique similar to Random Forest. It aggregates multiple de-correlated decision trees to generate classification result. It differs from Random Forest in how the decision trees are constructed in the forest. We combine Extra

Trees Classifier with permutation importance to select features. We will use Random Forest algorithm aiming for high prediction performance in the next section.

In a descending order, the figure below illustrates each feature's permutation importance with a horizontal box plot. Each box plot highlights the mean, median, quartiles, min, and max of decrease in prediction accuracy.

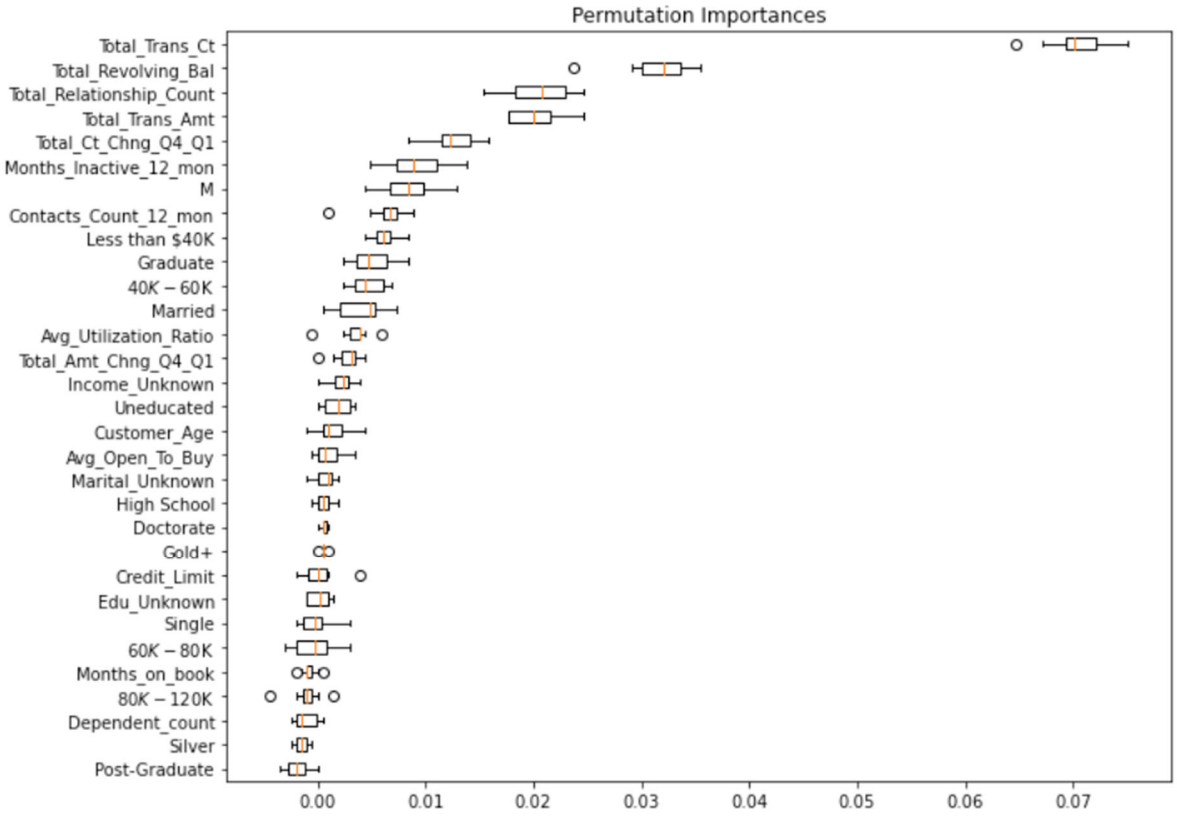

FIGURE 1: FEATURE PERMUTATION IMPORTANCE.

'Total_Trans_Ct' stands out as the most importance feature. When it is randomly shuffled, the model's prediction accuracy decreases by approximately 7%. 'Total_Revolving_Bal', reduces model accuracy by slightly above 3% when shuffled, is the second most important. 'Total_Relationship_Count' and 'Total_Trans_Amt' have similar importance score of around 2.1%. None of the other features scored above 1.5%.

Since SelectKBest methods find that choosing 15 features is near optimal, we decide to select 15 features based on permutation importance as well. The selected list starts with the most importance feature in 'Total_Trans_Ct' and ends with the 15th most important feature 'Avg_Open_To_Buy'. We ran a similar logistic regression model, which yields an even better AUC: 0.8309. We will use these 15 features in our ultimate logistic regression model.

To preserve as much interpretability as possible, we run a logistic regression model on the entire dataset without resampling. Numerical features are standardized. It is important to recall that in the entire dataset of 10,127 observations, there are 1,627 positive observations (churns) and 8,500 negative observations. As a result, the baseline churn probability is 1,627/10,127 = 16.07%, denoted

as p_0. Similarly, the baseline odds of churn is 1,627/8500 = 0.1914, denoted as r_0. In other words, the churn odds of an "average" customer is 0.1914 and the churn probability of this "average" customer is 16.07%. Table 3 below reports the coefficients of all 15 selected features in the final logistic regression model, as well as their corresponding odds and probabilities.

TABLE 3: COEFFICIENTS IN LOGISTIC REGRESSION MODEL AND INTERPRETATION.

Feature	**Coef.**	**odds ratio = exp(coef.)**	**odds (r) = odds ratio * r_0.**	**churn prob. (p) = $r/(1+r)$**	**% change = $(p-p_0)/p_0$**
'Total_Trans_Ct'	-1.897	0.150	0.029	2.79%	-82.63%
'Total_Revolving_Bal'	-0.512	0.599	0.115	10.29%	-35.96%
'Total_Trans_Amt'	1.039	2.825	0.541	35.10%	118.45%
'Total_Relationship_Count'	-0.532	0.587	0.112	10.11%	-37.08%
'Total_Ct_Chng_Q4_Q1'	-0.545	0.580	0.111	9.99%	-37.79%
'Months_Inactive_12_mon'	0.362	1.436	0.275	21.56%	34.20%
'Contacts_Count_12_mon'	0.420	1.522	0.291	22.55%	40.39%
'Total_Amt_Chng_Q4_Q1'	-0.108	0.898	0.172	14.66%	-8.74%
'M'	-2.674	0.069	0.013	1.30%	-91.89%
'Credit_Limit'	0.192	1.212	0.232	18.83%	17.22%
'Avg_Utilization_Ratio'	-0.131	0.877	0.168	14.37%	-10.54%
'Dependent_count'	0.080	1.083	0.207	17.17%	6.88%
'Single'	-1.285	0.277	0.053	5.03%	-68.70%
'Customer_Age'	-0.084	0.919	0.176	14.96%	-6.87%
'Avg_Open_To_Buy'	0.238	1.269	0.243	19.54%	21.65%

3.4. Quantification of Impacts on Customer Churn

Comparing the Wald test on a standard logistic regression and the logistic regression with feature selection, we find that the significant features are similar and hence the coefficient of the features. We use the result in Table 3 to quantify the impact of various features on Customer Churn.

In Table 3, 'Coef.' column collects the coefficients of the 15 selected features. In logistic regression, coefficients are logit or log-odds. The 'odds ratio' column converts coefficients into odds ratio, the exponential of coefficient. The 'odds' column turns exponentials of coefficient into actual odds of churn by multiplying the odds ratio and the baseline odds r_0. The 'churn probability' column calculates the actual churn probability using the odds in the previous column. The last column '% change' provides the percentage change in churn probability compared with the baseline churn probability p_0.

For example, the coefficient of 'Total_Trans_Ct' is -2.715 and exp(-2.715) =

0.066, the odds ratio. Recall that all the numerical features are standardized. If 'Total_Trans_Ct' is one standard deviation above the mean, the logit or log-odds of churn will decrease by 2.715, compared with an "average" customer. The mean number of total transaction count of last 12 months is 65 and the standard deviation is approximately 13. The exponential of -2.715, which is equal to 0.066, says that the odds of churn of this customer whose total transaction counts of last 12 months are 88 (1 standard deviation above the mean), is 0.066 times the churn odds of an "average" customer, assuming everything else remains the same. The churn odds of an "average" customer is our baseline odds: r_0 = 0.1914. Thus, 0.066*0.1914 = 0.013 is the churn odds of the customer with 88 transaction counts. The odds are in the column 'odds' in Table 3. We can easily convert odds into probabilities, odds/(1 + odds), which are in the 'churn prob.' column. Since 0.013/(1+0.013) = 1.25%, we know that, assuming everything else the same, the churn probability of a customer with 88 transactions is only 1.25%. Note that the churn probability of an "average" customer (with 65 transactions) is p_0 = 16.07%. This is a change of (1.25% - 16.07%)/16.07% = -92.21%, which is reported in the last column '% change'.

It is expected that 'Total_Trans_Ct' has significant negative effects on churn probability. We also expect 'Total_Revolving_Bal' and 'Total_Relationship_Count' have similar negative effects, but to lesser extent. We, however, need to be careful when interpreting the effect of 'Total_Ct_Chng_Q4_Q1', which reduces churn probability by 44.39%, if it is 1 standard deviation above its mean. The dataset does not specify the direction of the changes, whose values are all non-negative, indicating that they are likely the absolute values of the changes.

A surprising result is the effect of 'Total_Trans_Amt'. If a customer whose total transaction amount is 1 standard deviation above the mean, the churn probability rises dramatically from the baseline probability of 16.07% to 48.78%. This is counter-intuitive, worth further investigation if more data are available. The data on overdue balance can be helpful.

'Months_Inactive_12_mon' and 'Contacts_Count_12_mon' have strong positive impact on churn, which increase the churn probability by 49.67% and 55.32%, respectively. It is expected that if a client has been inactive longer, then this client is more likely to churn. What is intriguing is the effect of contact counts: the more contact a customer has, the more likely to churn. A possible explanation is that customers contact the bank usually when they are unsatisfied or have issues with certain transactions or services. Unsatisfied customers are more likely to churn.

Other numerical features, 'Total_Amt_Chng_Q4_Q1', 'Customer_Age', 'Credit_Limit', 'Avg_Utilization_Ratio', 'Avg_Open_To_Buy' and 'Dependent_count' have minor effects on churns, considering one standard deviation away from the mean is very significant. Among them, 'Dependent_count' has the largest impact: one standard deviation above the mean implies an increase of 16.80% in churn probability, from 16.07% to 18.76%. The mean 'Dependent_count' is 2.34 and the standard deviation is 1.30.

Only two binary features are selected in the top 15: 'M' and 'Single', both of which are rather influential on churn. The churn probabilities of male customers and single customers are 27.15% and 23.33%, respectively.

IV. VARIOUS CHURN PREDICTION MODELS

4.1. Random Forest

Random Forest is an ensemble learning method for classification, regression and other tasks through constructing a multitude of

decision trees at training time. In the context of binary output, decision trees are usually the tool for the easiness and interpretability (Hart, Stork, & Duda, 2000). Random forests was firstly introduced by Ho (1995). This technique uses a subset of *m* randomly chosen predictor to grow each tree on a bootstrap sample of the training data. Usually, this *m* is much lower than the total number of the features in the model. Then a large number of trees are generated. Each tree needs to decide the most popular class. Aggregating the decisions over different trees, we can predict a class label for each case (Breiman 2001,and Coussement & Van den Poel, 2008). We choose Random forests for the following reasons. First, it was proved by handful scholars that the predictive power is among the best of the available techniques (Luo et al, 2004). Second, the computation time is reasonable, based on a study of Buckinx & Van den Poel (2005). Lastly, we only need to set two free parameters, which are m and total number of the trees to be grown.

We start with the default RandomForestClassifier in scikit-learn. The number of trees in our Random forest model is thus 100. The maximum number of features in a subtree is sqrt(31), where 31 is the number of features after data preprocessing. In Section 4.3, we apply RandomizedSearchCV to perform cross validation and hyperparameter turning on the default Random forest model. The performance results on the test set of both models are provided in Table 4 in Section 4.5. The default Random forest model is listed as 'rf default' and the cross-validated Random forest model is listed as 'rf RandomizedSearchCV'. As we can see from Table 4, the default Random forest model has reasonable performance. Performance improvement is small after cross validation and hyperparameter tuning. Compared with their performances on the training set, the drop in performance is minimal. Therefore, overfitting is not of a major concern.

4.2. XGBoost

XGBoost is an algorithm based on tree learner. It serially generates trees to predict errors. During this process: (1) the predicted value gradually approaches the real errors; (2) the predicted errors of each tree is added up; and (3) the cumulative predicted value is the result. (Chen et al., 2015)

We select XGBoost for a couple of reasons. First, gradient boosting has algorithm that is easy to read and interpret, for classification and regression problems. It is a resilient and robust method which helps to prevent over-fitting. Second, we have relatively medium and structured dataset, with only 27 features. XGboost performs very well on medium, small data with structured dataset (Hachcham, 2022). Third and the most importantly, the most recent publication on this subject we have found (Shwartz-Ziv & Armon, 2021) demonstrated that XGBoost outperforms several deep learning models proposed in recent years across 11 widely used benchmark datasets. In addition to those advantages which encourage us to use this model, XGBoost is also a great choice for solution seeker requiring scalability. Cheb & Guestrin (2016) state that XGBoost runs 10 times faster than existing popular solutions on a single machine and scales to billions of examples in distributed or memory-limited settings. XGBoost exploits out-of-core computation and enables users to process a large dataset on a desktop.

If we use n to denote the number of observations (or customers) and m to denote the number of features. Each data point can be represented as $\{(x_i, y_i)\}, i = 1,2,\dots n$ and $x_i \in R^m, y_i \in R$. When training the t^{th} tree, we minimize the objective as in the following equation (Chen et al., 2021).

$$obj^t = \sum_{i=1}^{n} l\left(y_i, \hat{y}_l^{t-1} + f_t(x_i)\right) + \Omega(f_t) + C$$

Where $\hat{y}_l^{t-1}$ is the decision tree integration and $\hat{y}_l^{t-1} = \sum_{k=1}^{K} f_k(x_i)$, $f_k \in F$; K is the number of activation functions; F is the

collection of all CART trees; l is the loss function; $\Omega(f_t)$ is a regularization term and C is a constant.

Similarly, we first run an XGBoost model with default values. Then we once again conduct cross validation and hyperparameter tuning with RandomizedSearchCV. Their performances on the test set are in Table 4 in Section 4.5, listed under 'XGBoost default' and 'XGBoost RandomizedSearchCV', respectively. The default XGBoost model has excellent performance results on the test set. The cross-validated XGBoost model offers marginal improvement. Both models have similar performance results on the training and test sets. Thus, our models do not overfit the data.

4.3. Cross Validation and Hyperparameter Tuning with RandomizedSearchCV

Randomized parameter optimization is a powerful tool, enabling us to further improve model performance by way of finding better values to the hyperparameters. Randomized parameter optimization is realized with RandomizedSearchCV in scikit-learn. We specified a wide range of values on each parameter for each model (Random forest model and XGBoost model). When fine-tuning the Random forest model, we allow the number of trees to be as many as 2000, maximum tree depth to be between 10 and 110, and the maximum number of features to be determined either by 'sqrt' or 'auto'. In cross validation of XGBoost model, key hyperparameters extensively searched include the learning rate, maximum depth of tree, and maximum number of trees. We considered a uniform distribution on the learning rate between 0.1 and 0.5, randomized the maximum tree depth to be between 6 and 9, and accounted for the maximum number of subtrees to be between 100 and 300.

In addition, we employed k-fold (k=3) cross-validation on both models. We did not further explore cross-validation with $k > 3$ for the following reasons. Performing RandomizedSearchCV is computationally expensive. Each run took approximately 10 minutes on our computers. Additionally, we ran our models on different training/test set splits. The performance results of all models are extraordinarily consistent. All models had very similar performances on different training and test sets. Consequently, it is reasonable to believe that: (1) overfitting is not a major concern; (2) k-fold cross validation with $k > 3$ is unlikely to generate significantly better results.

4.4. Multi-Layer Perceptron Neural Networks

The universal approximation theorems establish that multi-player feedforward neural networks (MLP) can approximate any continuous functions defined on Euclidean spaces (Hornik, Stinchcombe, and White 1989). In theory, there exists an MLP neural network that can closely approximate the true mapping between the input and the output. However, the challenge is the choice of possible neural network structures is infinite. To the best of our knowledge, there is no known way of constructing such a neural network. It is also an unsettled debate if deep neural networks outperform tree ensemble models such as XGBoost on tabular datasets. The most recent publication on this subject we have found (Shwartz-Ziv & Armon, 2021) demonstrated that XGBoost outperforms several deep learning models proposed in recent years across 11 widely used benchmark datasets. They also showed that an ensemble of deep neural networks and XBGoost performs better than XGBoost.

In our exploration, we consider a simple neural network with one hidden layer of 100 neurons. We also take into account two different activation functions: ReLU and logistic. We adopted such a simple architecture because more complex networks are harder to converge and more likely to overfit. We experiment a

network with 200 neurons in the hidden layer, which underperforms. The results of both MLP models are provided in the next section, see Table 4 and Table 5.

4.5. Model Performance

In addition to prediction accuracy, we evaluate our models using precision, recall, f_β, and AUC (area under ROC curve). AUC is the key model selection metric.

If TP is the total number of true positive, FP is the number of false positive, and FN is the number of false negative, we can write down the formulas for precision, recall and f_β as follows:

- Precision = $\frac{TP}{TP+FP}$
- Recall = $\frac{TP}{TP+FN}$ (a.k.a. true positive rate)
- $f_\beta = \frac{1+\beta^2}{\frac{\beta^2}{Recall}+\frac{1}{Precision}}$ (β is a constant).

High precision implies that more predicted positives are true positives. In other words, among predicted positives, there are more true positives relative to false positives. Higher recall indicates that more true positives are predicted as positive. In other words, among true positives, there are more predicted positives relative to false negative. A model may have high precision but low recall or vice versa. f_β intends to strike a balance between precision and recall, where β is the times that recall is as important as precision. When β = 1, we consider recall and precision equally important. Since false negative (an actual churn is predicted to be no churn) is of a major concern, we consider three different β values: 1, 2, and 4, to give recall and thus false negatives a higher and higher weight. f_1 serves as a benchmark. All three f-measures are reported in Table 4 below.

ROC curve shows the performance of a classifier at all classification thresholds. The threshold is a probability value above which the classifier predicts positive, and negative, otherwise. ROC curve plots true positive rate on vertical axis against false positive rate on horizontal axis across all threshold values. True positive rate is the recall defined earlier. False positive rate is defined to be FP/(FP+TN), which tells us among true negatives, how many percent are incorrectly identified as positive. AUC is the area under this ROC curve, which is an aggregate measure of performance across all possible classification thresholds. A model has an AUC of zero if it always makes wrong prediction. A model has an AUC of one if its predictions are always correct.

From Table 4, it is clear that XGBoost with RandomizedSearchCV out beat all other methods, which includes random forest with or without RandomizedSearchCV, and default XGBoost. The default XGBoost is also a good option with considerable good performance in accuracy, precision, recall, f_1, and AUC. XGBoost becomes superior to Random Forest for the following reasons (Gupta, 2021): XGBoost straight away prunes the tree with a score called "Similarity score" before entering into the actual modeling purposes; XGBoost always gives more importance to functional space when reducing the cost of a model while Random Forest tries to give more preferences to Hyperparameters; XGBoost is a better option for unbalanced data sets as Random Forest might give more preferences to the class of a categorical variable with more participation. Both Random Forest and XGBoost perform better than the logistic regression; however, these two models also lose the power of interpretability. Such fact verifies the challenge of maintaining both higher performance and excellent interpretability in one model – there always is trade off.

TABLE 4: MODEL PERFORMACES ON TEST SET.

	accuracy	recall	precision	f_1	f-beta (beta = 2)	f-beta (beta = 4)	AUC
LogReg	84.0573	81.6514	50.3774	0.6231	0.7263	0.7877	0.8309
rf default	95.50839	83.48624	88.06452	0.85714	0.84363	0.83742	0.90654
xgb default	96.7917	89.2966	90.6832	0.8998	0.8957	0.8938	0.9377
rf Randomized SearchCV	95.8045	83.7920	89.5425	0.8657	0.8488	0.8411	0.9095
xgb Randomized SearchCV	97.1866	89.2966	92.9936	0.9111	0.9001	0.8951	0.9400
MLP - ReLU	93.5341	77.3700	81.6129	0.7943	0.7818	0.7761	0.8700
MLP - Logistic	94.4719	79.2049	85.4785	0.8222	0.8038	0.7955	0.8831

An interesting and importance observation from Table 4 is that, logistic regression model is the only one that has better f_β scores as β increases. Generally speaking, false negative (a churned customer incorrectly predicted to be non-churn) is more problematic than false positive for credit card issuers, who likely select high β value, thus making the logistic regression model more desirable.

A confusion matrix is a strong method of evaluating the performance of the churn models. It tell the degree of four important classification matrix: a) True Positive (TP) – the number of observations where the model predicted the customer would churn and they actually do churn; b) True Negative (TN) – the number of observation customer predicted to not churn and the actually do not churn; c) False Positive (FP) – number of the observations that model predicted the customer will churn but they do not actually; and d) False Negative (FN) - the number of observations where the model predicted the customer will not churn but they churn actually. From Table 5, FN would be more problematic in our context. Logistic Regression has the highest FN in both training sets and testing sets. Among other models, XGBoost (either default or cross validated) stands out with the lowest FN (0 for training set and 35 for testing set).

V. INSIGHTS AND CONCLUSION

Our work strikes a balance between model interpretability and prediction performance. Logistic regression model is a default choice for interpretability in binary classification. After carefully select 15 most important features, we have measured their effects on the customer churn probability. For example, an average customer has 65 transactions per year and the average churn probability is 16.07%. If a customer transacts 88 times per year, the churn probability of this customer drops to 1.25%. Other consequential features that negatively affect churn probability include total revolving balance and total relationship counts. It is counter intuitive that the churn probability of a customer with high total transaction amount is markedly higher. We recommend further investigation. Also interesting is that male and single customers are much more likely to churn.

TABLE 5: CONFUSION MATRICES.

Logistic Regression - Train	
TN = 5909	FP = 892
FN = 247	TP = 1053

Logistic Regress - Test	
1448	251
70	257

Random Forest Default - Train	
TN = 6801	FP = 0
FN = 0	TP = 1300

Random Forest Default - Test	
1662	37
54	273

XGBoost Default - Train	
TN = 6801	FP = 0
FN = 0	TP = 1300

XGBoost Default - Test	
1669	30
35	292

RF Cross Validated - Train	
TN = 6801	FP = 0
FN = 0	TP = 1300

RF Cross Validated - Test	
1667	32
53	274

XGBoost Cross Validated - Train	
TN = 6801	FP = 0
FN = 0	TP = 1300

Logistic Regress - Test	
1677	22
35	292

MLP - ReLU - Train	
TN = 6801	FP = 0
FN = 0	TP = 1300

MLP - ReLU - Test	
1642	57
74	253

MLP - Logistic - Train	
TN = 6801	FP = 0
FN = 0	TP = 1300

MLP - Logistic - Test	
1655	44
68	259

To gain better churn prediction accuracy, we tried other models such as Random Forest with and without RandomizedSearchCV, XGBoost with and without RandomizedSearchCV, and Multi-Layer Perceptron Neural Networks. Among them, XGBoost performs relative better, no matter with or without RandomizedSearchCV, in the metrics of accuracy, recall, precision, f1 and AUC. Logistic regression as a benchmark has the lowest but still decent level of performance. A well-prepared data set helps to improve the performance of the simplistic logistic regression model. Despite the advanced performance of Random Forest and XGBoost, these ensemble algorithms greatly increase model development time, because analyst must decide the multiple hyperparameters. Intuitively, more complex algorithms lack convincing and interpretable insights. Overall, the added complexity makes the churn prediction process more costly.

There are some limitations to the generalizability of our findings. First, we use a case study approach. Our analysis is limited to the dataset provided by one bank. The readiness of the data would greatly impact the prediction power of various models. It is interesting to test out the models over a more generalized data set in the industry. The credit card market continues to evolve and change over years. To consider the market characteristic, promising evaluation

metrics would be able to integrate cost and profit considerations. Future research should evaluate the models on the basis of such metrics. Lastly, since data preprocessing practices, such as class imbalance and outlier treatment can impact the result. We could revisit the impact of such practices on the algorithms presented in this study.

REFERENCES

Ahmed, A. A., & Maheswari, D. (2017). Churn prediction on huge telecom data using hybrid firefly based classification. *Egyptian Informatics Journal*, *18*(3), 215-220.

Amin, A., Anwar, S., Adnan, A., Nawaz, M., Alawfi, K., Hussain, A., & Huang, K. (2017). Customer churn prediction in the telecommunication sector using a rough set approach. *Neurocomputing*, *237*, 242-254.

Ali, Ö. G., & Arıtürk, U. (2014). Dynamic churn prediction framework with more effective use of rare event data: The case of private banking. *Expert Systems with Applications*, *41*(17), 7889-7903.

Ascarza, E., Neslin, S. A., Netzer, O., Anderson, Z., Fader, P. S., Gupta, S., ... & Schrift, R. (2018). In pursuit of enhanced customer retention management: Review, key issues, and future directions. *Customer Needs and Solutions*, *5*(1), 65-81.

Blattberg, R. C., Getz, G., & Thomas, J. S. (2001). *Customer equity: Building and managing relationships as valuable assets*. Harvard Business Press.

Breiman, L. (2001). Random Forests. *Machine Learning*, 45(1), 5-32.

Buckinx, W., & Van den Poel, D. (2005). Customer base analysis: partial defection of behaviourally loyal clients in a non-contractual FMCG retail setting. *European journal of operational research*, 164(1), 252-268.

Burez, J., & Van den Poel, D. (2008). Separating financial from commercial customer churn: A modeling step towards resolving the conflict between the sales and credit department. *Expert Systems with Applications*, *35*(1-2), 497-514.

Chawla, N. V., Bowyer, K. W., Hall, L. O., & Kegelmeyer, W. P. (2002). SMOTE: synthetic minority over-sampling technique. *Journal of artificial intelligence research*, 16, 321-357.

Chen, T., He, T., Benesty, M., Khotilovich, V., Tang, Y., Cho, H., & Chen, K. (2015). Xgboost: extreme gradient boosting. *R package version 0.4-2*, 1(4), 1-4.

Chen, T., & Guestrin, C. (2016, August). Xgboost: A scalable tree boosting system. *In Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining* (pp. 785-794).

Coussement, K., & Van den Poel, D. (2008). Churn prediction in subscription services: An application of support vector machines while comparing two parameter-selection techniques. *Expert systems with applications*, *34*(1), 313-327.

Coussement, K., & Van den Poel, D. (2009). Improving customer attrition prediction by integrating emotions from client/company interaction emails and evaluating multiple classifiers. *Expert Systems with Applications*, *36*(3), 6127-6134.

Coussement, K., Lessmann, S., & Verstraeten, G. (2017). A comparative analysis of data preparation algorithms for customer churn prediction: A case study in the telecommunication industry. *Decision Support Systems*, *95*, 27-36

De Bock, K. W., & De Caigny, A. (2021). Spline-rule ensemble classifiers with structured sparsity regularization for interpretable customer churn modeling. *Decision Support Systems*, *150*, 113523.

Dawes, J., & Swailes, S. (1999). Retention sans frontieres: issues for financial service retailers. *International Journal of Bank Marketing*.

Farquad, M. A. H., Ravi, V., & Raju, S. B. (2014). Churn prediction using comprehensible support vector machine: An analytical CRM application. *Applied Soft Computing*, *19*, 31-40.

Gallant, S. I. (1988). Connectionist expert systems. *Communications of the ACM*, *31*(2), 152-169.

Gupta, A. *XGBoost versus Random Forest*. (2021, April 26). Medium. https://medium.com/geekculture/xgboost-versus-random-forest-898e42870f30#:~:text=One%20of%20the%20most%20important,hyperparameters%20to%20optimize%20the%20model.sdf

Gupta, S., Lehmann, D. R., & Stuart, J. A. (2004). Valuing customers. *Journal of marketing research*, *41*(1), 7-18.

Hachcham, A. XGBoost: everything you need to know. (2022, November 14). Neptune.ai. https://neptune.ai/blog/xgboost-everything-you-need-to-know#:~:text=XGBoost%20performs%20very%20well%20on,XGBoost%20is%20the%20reigning%20king

Hart, P. E., Stork, D. G., & Duda, R. O. (2000). *Pattern classification*. Hoboken: Wiley.

Hill, S., Provost, F., & Volinsky, C. (2006). Network-based marketing: Identifying likely adopters via consumer networks. *Statistical Science*, *21*(2), 256-276.

Ho, T. K. (1995, August). Random decision forests. In *Proceedings of 3rd international conference on document analysis and recognition* (Vol. 1, pp. 278-282). IEEE.

Hornik, K., Stinchcombe, M., & White, H. (1989). Multilayer Feedforward Networks are Universal Approximators. *Neural Networks* (Vol. 2, pp. 359-366). Pergamon Press.

Huang, B., Kechadi, M. T., & Buckley, B. (2012). Customer churn prediction in telecommunications. *Expert Systems with Applications*, *39*(1), 1414-1425.

Idris, A., & Khan, A. (2014, December). Ensemble based efficient churn prediction model for telecom. In *2014 12th International Conference on Frontiers of Information Technology* (pp. 238-244). IEEE.

Japkowicz, N., & Stephen, S. (2002). The class imbalance problem: A systematic study. Intelligent data analysis, 6(5), 429-449.

Kirui, C., Hong, L., Cheruiyot, W., & Kirui, H. (2013). Predicting customer churn in mobile telephony industry using probabilistic classifiers in data mining. *International Journal of Computer Science Issues (IJCSI)*, *10*(2 Part 1), 165.

Kotsiantis, S. (2011). Feature selection for machine learning classification problems: a recent overview. *Artificial intelligence review*, 42(1), 157-176.

Lassar, W. M., Manolis, C., & Winsor, R. D. (2000). Service quality perspectives and satisfaction in private banking. *Journal of services marketing*.

Larivière, B., & Van den Poel, D. (2004). Investigating the role of product features in preventing customer churn, by using survival analysis and choice modeling: The case of financial services. *Expert Systems with Applications*, *27*(2), 277-285.

Luo, T., Kramer, K., Goldgof, D. B., Hall, L. O., Samson, S., Remsen, A., & Hopkins, T. (2004). Recognizing plankton images from the shadow image particle profiling evaluation recorder. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics),* 34(4), 1753-1762.

Martens, D., Vanthienen, J., Verbeke, W., & Baesens, B. (2011). Performance of classification models from a user perspective. *Decision Support Systems*, *51*(4), 782-793.

Moeyersoms, J., & Martens, D. (2015). Including high-cardinality attributes in predictive models: A case study in churn prediction in the energy sector. *Decision support systems*, *72*, 72-81.

Neslin, S. A., Gupta, S., Kamakura, W., Lu, J., & Mason, C. H. (2006). Defection detection: Measuring and understanding the predictive accuracy of customer churn models. *Journal of marketing research*, *43*(2), 204-211.

Peppers, D., & Rogers, M. (2004). Managing Customer Relationships, A Strategic Framework. John iley & Sons. *Inc., Hoboken, New Jersey*.

Prinzie, A., & Van den Poel, D. (2006). Incorporating sequential information into traditional classification models by using an element/position-sensitive SAM. *Decision Support Systems*, *42*(2), 508-526.

Sharma, R. R., & Rajan, S. (2017). Evaluating prediction of customer churn behavior based on artificial bee colony algorithm. *International Journal Of Engineering And Computer Science*, *6*(1), 20017-20021.

Shwartz-Ziv, R., & Armon, A. (2022). Tabular data: Deep learning is not all you need. *Information Fusion*, 81, 84-90.

Tsai, C. F., & Lu, Y. H. (2009). Customer churn prediction by hybrid neural networks. *Expert Systems with Applications*, *36*(10), 12547-12553.

Verbeke, W., Martens, D., Mues, C., & Baesens, B. (2011). Building comprehensible customer churn prediction models with advanced rule induction techniques. *Expert systems with applications*, *38*(3), 2354-2364.