Prediction of Churning Credit Card Customers

Kushalkumar Patel
College of Engineering
Northeastern University
Toronto, ON
patel.kushal@northeastern.edu

Gunjit Arora
College of Engineering
Northeastern University
Toronto, ON
arora.g@northeastern.edu

Adedamola Adesoye
College of Engineering
Northeastern University
Toronto, ON
adesoye.a@northeastern.edu

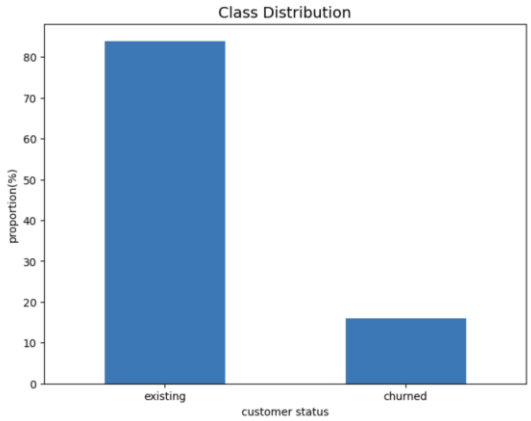
Abstract

This report investigates the performance of three tree-based algorithms for predicting customer churn in a credit card dataset. Specifically, we compare the results of Decision Tree, Random Forest, and XGBoost using a threshold of 0.5 for the positive class, and also explore the impact of lowering the threshold to arbitrary figures based on the precision-recall curves. Additionally, we analyze the most important features used by each model to split the data. Our findings provide insights into the strengths and weaknesses of each algorithm and their potential use in predicting churn for this particular dataset.

1 Dataset

The dataset used in this project contains 10,127 instances and 23 attributes with no missing values. Each data point represents a unique credit card account that has been on the bank's books for a period of 13 to 56 months. The features are a mix of categorical and numerical variables, with ten integer, seven float, and six object data types. The attributes include demographic data such as age, gender, marital status, and income category, as well as account-specific information like the card category (blue, silver, gold, platinum), credit limit, and the number of months the account has been active. The complete list of features and their meanings is provided in Appendix A.

The most important feature in this dataset is the *Attrition_Flag*, which indicates if an account is still active or if it has been churned. The dataset exhibits a high degree of class imbalance, with only 16.06% of accounts being classified as churned, as shown in the figure below.



1.1 Why is this dataset interesting?

Customer churn prediction is an important task for businesses as it helps them to retain existing customers and reduce customer acquisition costs. The credit card customer dataset is particularly interesting because it contains many features that can potentially help in predicting customer churn, such as demographics, account information, and transaction history. Additionally, the dataset is representative of real-world customer data, making it a valuable resource for developing and testing churn prediction models.

1.2 Preprocessing

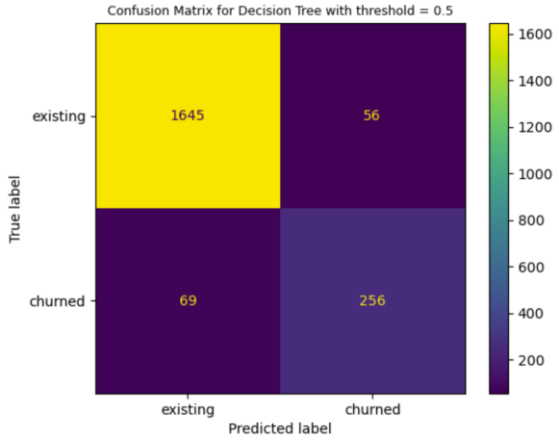
To prepare the data for modeling, we first dropped the last two features as they were highly correlated with the target column and would have biased our predictions. We also dropped the *CLIENTNUM* feature as it provided no useful information. The dataset had no null values, but we found that three categorical features (*Education_Level*, *Marital_Status*, and *Income_Category*) had Unknown label values. Instead of dropping the instances with Unknown values, which would have resulted in losing 30% of the data, we one-hot encoded the categorical variables and dropped the features representing the unknown labels (e.g., *Marital_Status_Unknown*). We then split the data into 80% training set and 20% test set while maintaining the same class proportions in both sets. Finally, we fit a standard scaler to the training set and used it to transform both the training and test set for modeling.

2 Model Building and Evaluation

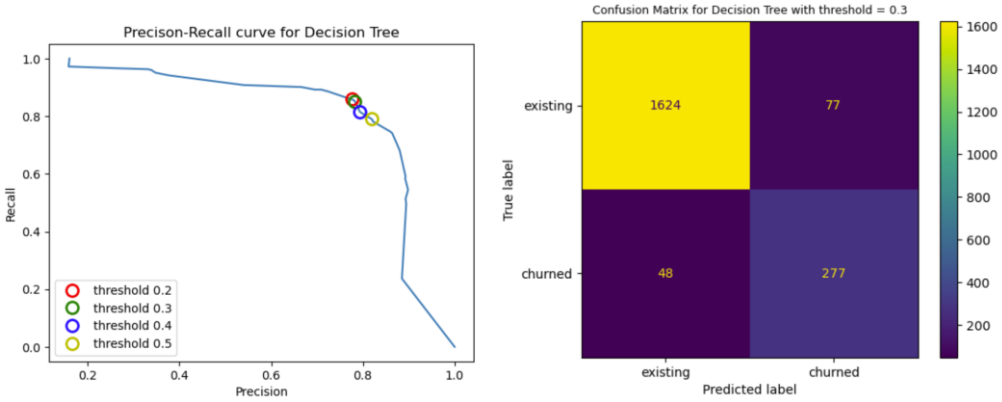
In building our models, we considered three tree-based algorithms namely, Decision Tree, Random Forest and XGBoost. All models we built posted accuracy scores of at least 0.94, which is a high level of accuracy. However, since the data is imbalanced, accuracy alone is not an adequate metric to evaluate our models. Therefore, we focused more on more suitable metrics like precision, recall and f1-score in our analysis. These metrics provide a better evaluation of how well our models are performing in predicting the minority class (i.e., attrited accounts).

2.1 Decision Tree

We evaluated the performance of a Decision Tree classifier for our churn prediction task. We first performed hyperparameter tuning using grid search with a stratified cross-validation of 5 to find the best hyperparameter for the model. The hyperparameter we tuned was max_depth testing 20 values: *None, 1, 2, 3…19*.
The best estimator based on f1-score was a DecisionTreeClassifier with max_depth of 7. We then used this estimator to fit our training set and made predictions on our test set. When the threshold for the positive class (churn) was set to the default value of 0.5, the model achieved the confusion matrix below with precision, recall, and f1-score of 0.82, 0.79, and 0.80, respectively.



To further improve the model, we plotted the precision-recall curve and noticed that lowering the threshold to 0.3 yielded the most significant increase in recall while sacrificing the least amount of precision. Setting the threshold to 0.3 resulted in the confusion matrix below with a precision, recall, and f1-score of 0.78, 0.85, and 0.82, respectively.



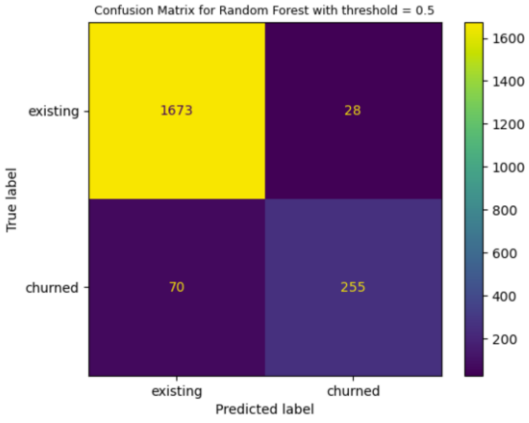
Comparing the confusion matrix with the threshold set to 0.5 and 0.3, we can see that increasing our true positives by 21 came at the cost of misclassifying 21 more existing customers as churned. Depending on the bank's intended offer for these customers, this trade-off may be acceptable. Next, we will evaluate the performance of a Random Forest classifier and compare it with the Decision Tree classifier.

2.2 Random Forest

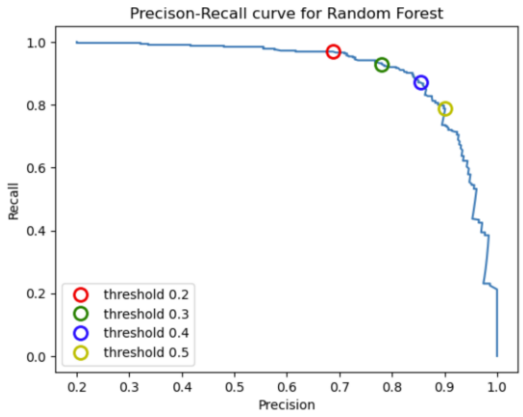
We optimized the Random Forest model by tuning the hyperparameters listed in the table and determined that the best estimator, based on the f1-score, was RandomForestClassifier(max_depth=18, n_estimators=100).

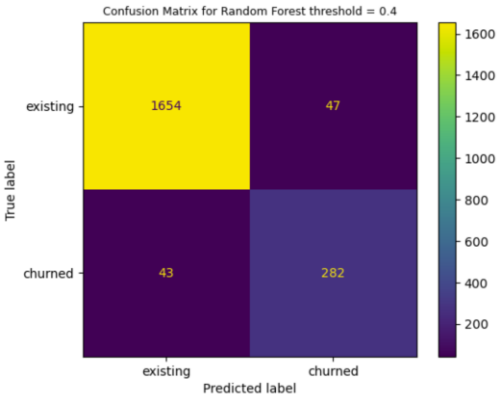
Hyperparameter	Tested Values	Best Value
Max_depth	None, 1, 2, 3…19	18
N_estimators	100, 200, 300	100

We evaluated the model's performance on our test set, and at a threshold of 0.5 for the positive class, the model achieved precision, recall, and f1-score of 0.90, 0.78, and 0.84, respectively.



Although the Random Forest model detected one less true positive than the Decision Tree model at the same threshold, it had a considerably higher precision. By lowering the threshold to 0.4, we could achieve a significantly higher recall for a fair trade-off in precision, resulting in new scores of precision: 0.86, recall: 0.87, and f1-score: 0.86, as shown in the precision-recall curve on the left.





The confusion matrix on the right displays a clearer representation of the effects of reducing the threshold to 0.4. The plot indicates that we can achieve a fairer trade-off (27 more true positives with 19 more false positives) than the Decision Tree model at a higher threshold.

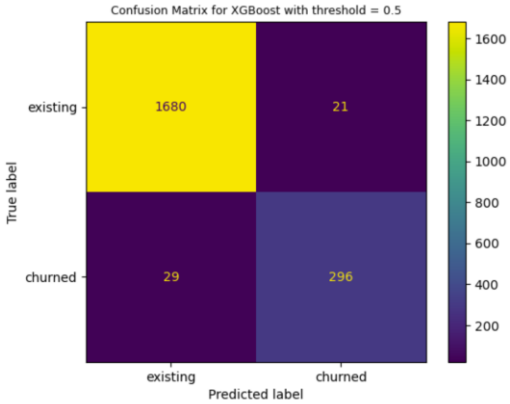
Next, we will explore the performance of the XGBoost model.

2.3 XGBoost

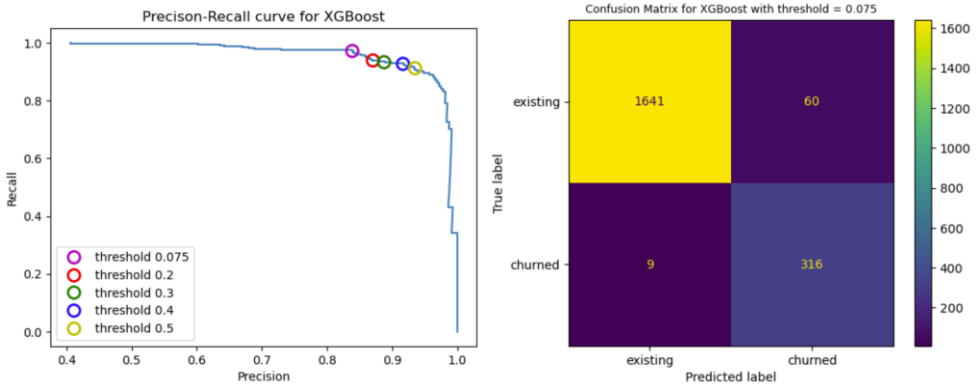
We performed hyperparameter tuning using the table below and found that the best estimator based on f1-score was XGBClassifier with a learning rate of 0.5, max depth of 3, and 200 estimators.

Hyperparameter	Tested Values	Best Value
Max_depth	3,5,7	3
Learning rate	0.01, 0.1, 0.5	0.5
N_estimators	50, 100, 200	200

Using this estimator, we predicted the test set and achieved precision, recall, and f1-score of 0.93, 0.91, and 0.92, respectively, when the threshold for the positive class was set to 0.5.



As shown in the confusion matrix above, XGBoost outperformed both the Decision Tree and Random Forest models at the two tested thresholds.

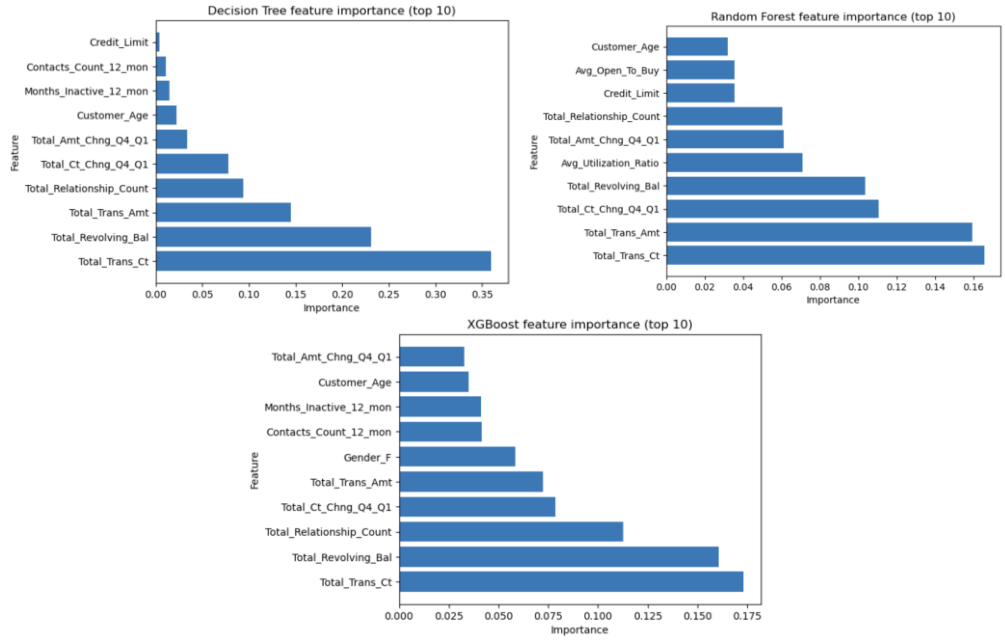


The precision-recall curve above indicates that increasing the recall at the expense of precision by setting the threshold at 0.2, 0.3, or 0.4 is not advisable as it would result in a significant loss of precision with little gain in recall. However, retaining a few high-paying customers who would otherwise have churned is important, and a small increase in recall could be beneficial.

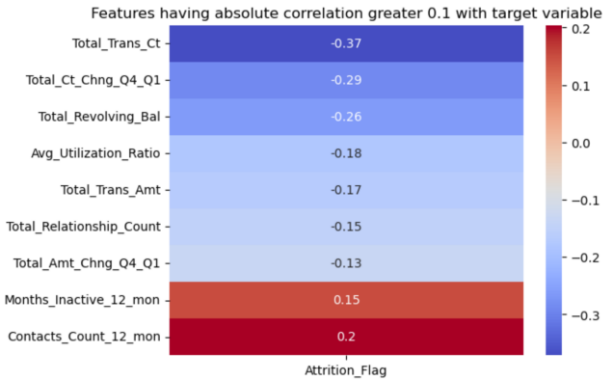
We found that setting the threshold at 0.075 resulted in the most significant increase in recall while maintaining a precision above 0.8. This threshold yielded the confusion matrix above with precision, recall, and f1-score of 0.84, 0.97, and 0.90, respectively. Overall, models built using XGBoost were superior to those built using the Decision Tree and Random Forest. Finally, we analyzed the features that were most important for the models in making their decisions.

2.4 Feature Importance

The top ten most important features used by the three models in their decision-making process are displayed in the figures below.



Interestingly, all three models agree that the total number of transactions made by each account is the most important feature. Other notable features include Total_Revolving_Bal (the total amount of revolving credit the customer is currently utilizing), Total_Relationship_Count (the total number of products the customer holds with the bank), and Total_Amt_Chng_Q4_Q1 (percentage change in the number of transactions made by a customer in the fourth quarter of the year compared to the first quarter).



Additionally, a heatmap was created to show the features that have an absolute correlation greater than 0.1 with the *Attrition_Flag*. As expected, *Total_Trans_Ct* had the highest absolute correlation with the *Attrition_Flag*, and it was negatively correlated. This indicates that the lower the number of transactions an account makes, the more likely a customer is to churn, which makes sense intuitively.

Overall, the feature importance analysis provided insight into the most important factors that drive customer churn, which can be used to inform future strategies aimed at reducing churn rates.

3 Conclusions

This report highlights the importance of predicting customer churn for a bank's credit card customers and the effectiveness of tree-based algorithms in addressing this problem. The analysis shows that the total transaction count of an account is the most critical indicator of an account most likely to churn. Moreover, the XGBoost algorithm outperforms the Decision Tree and Random Forest algorithms in predicting customer churn. By adjusting the threshold for these algorithms, we can capture more accounts likely to churn while acknowledging the increased number of false positives. However, the cost of the false positives is minimal in this business case, as it may lead to increased customer engagement and retention. Therefore, detecting more true positives at the expense of more false positives is a worthwhile trade-off. Overall, these findings can help the bank to identify accounts that are at risk of churn and take proactive measures to retain these customers, ultimately contributing to the bank's financial success.

Acknowledgments

The learning code is adapted from:
https://github.com/amueller/introduction_to_ml_with_python and references therein.

References

[1] Wiryaseputra, M., (2022, September 27). Bank Customer Churn Prediction Using Machine Learning. Analytics Vidhya. https://www.analyticsvidhya.com/blog/2022/09/bank-customer-churn-prediction-using-machine-learning/#:~:text=Customer%20Churn%20prediction%20means%20knowing,more%20than%20retaining%20existing%20ones.

A Appendix

CLIENTNUM: A unique identifier for each customer.

Attrition_Flag: Whether the customer is an existing customer (Existing Customer) or has left the bank (Attrited Customer).

Customer_Age: The age of the customer.

Gender: The gender of the customer (Male or Female).

Dependent_count: The number of dependents the customer has.

Education_Level: The customer's educational qualification (e.g. High School, Graduate, etc.).

Marital_Status: The customer's marital status (e.g. Married, Single, etc.).

Income_Category: The customer's income bracket (e.g. Less than $40K, $40K - $60K, etc.).

Card_Category: The type of credit card the customer holds (e.g. Blue, Silver, Gold, Platinum).

Months_on_book: The number of months the customer has been a credit card holder.

Total_Relationship_Count: The total number of products the customer holds with the bank.

Months_Inactive_12_mon: The number of months the customer has been inactive in the last 12 months.

Contacts_Count_12_mon: The number of times the customer has been contacted in the last 12 months.

Credit_Limit: The credit limit assigned to the customer.

Total_Revolving_Bal: The total amount of revolving credit the customer is currently utilizing.

Avg_Open_To_Buy: The average amount of credit available to the customer. (i.e.: Credit_Limit – Total_Revolving_Bal)

Total_Amt_Chng_Q4_Q1: The change in transaction amount (Q4 over Q1).

Total_Trans_Amt: The total dollar amount of transactions conducted by the customer.

Total_Trans_Ct: The total number of transactions conducted by the customer.

Total_Ct_Chng_Q4_Q1: The change in transaction count (Q4 over Q1).

Avg_Utilization_Ratio: The average utilization ratio of the customer's credit limit. (i.e.: Total_Revolving_Bal divided by Credit_Limit to 3 decimal places)

Naive_Bayes_Classifier_Attrition_Flag_Card_Category_Contacts_Count_12_mon_Dependent_count_Education_Level_Months_Inactive_12_mon_1: The probability that an account is attrited given Card_Category, Contacts_Count_12_mon, Dependent_count, Education_Level, Months_Inactive_12_mon

Naive_Bayes_Classifier_Attrition_Flag_Card_Category_Contacts_Count_12_mon_Dependent_count_Education_Level_Months_Inactive_12_mon_2: The probability that an account is existing given Card_Category, Contacts_Count_12_mon, Dependent_count, Education_Level, Months_Inactive_12_mon