Christian Campbell **DSC550**

Bank Credit Card Churn

Introduction

For this project, I looked at market churn for a bank's credit card. The specific problem I wanted to solve was how to determine which customers were likely to churn by looking at certain data belonging to the customers. Solving a problem like this would help a bank intervene with various product options or better incentives for the customer before the customer gets an opportunity to churn. Thus, helping the bank reduce their rate of customer churn and subsequently stabilize or increase their revenue.

If I was pitching this problem to a group of stakeholders, I'd concentrate on the negative outcomes that are caused due to customer churn and then proceed to explain to them how we could eliminate those negative outcomes by being able to identify potential "churners" before they churn and intervene in a manner that is more favorable to the bank.

The dataset I used for this project was collected from the Kaggle website. It was specifically uploaded to help data scientists work on problems relating to customer churn.

Organized and detailed summary of Milestones 1-3

Milestone 1

My idea for the project was to look at a bank's dataset and determine how many customers decide to stay and how many decide to leave over a given period of time. Upon further research I discovered the term "churn" to refer to how many people stop using a product. As a result, I narrowed my focus to solving how a bank could identify customers before they churn. For the product I decided to focus on credit cards. The idea is that if the bank can predict these customers before they stop using the product, maybe they could offer them more enticing options to convince

them to stay. So basically, I needed a bank dataset that provided customer information, churn information and credit card information. More research led me to Kaggle which is where I found a dataset specifically for this exercise.

The dataset consisted of 10,000 customers and 18 features. The features are:

1. CLIENTNUM
2. Attrition_Flag
3. Customer_Age
4. Gender
5. Dependent_count
6. Education_Level
7. Marital_Status
8. Income_Category
9. Card_Category
10. Months_on_book
11. Credit_Limit
12. Total_Revolving_Bal
13. Average_Open_To_Buy
14. Total_Amt_Chng_Q4-Q1

15. Total_Trans_Amt
16. Total_Trans_Ct
17. Total_Ct_Chng_Q4_Q1
18. Avg_Utilization_Ratio

I created four diagrams to show certain information from the dataset. Fig 1 showed the customer attrition. It showed that there are 8,500 existing customers and 1,627 attrited customers. Attrited customers are those that have left the program. Fig 2 showed the same information as a pie chart. The pie chart showed that 83.9% of the customers are existing customers while 16.1% of the customers are attrited customers. Fig 3 shows the age of the customers and how many months they've been using the product for. As the age increases, so do the months on book. Fig 4 shows the credit limit vs total revolving balance. We can see that the majority of customers are below the $15,000 credit limit.

Milestone 2

- In this milestone, the very first thing that I did was delete the last two columns of the data frame. They were the "Naïve_Bayes_Classifier." I deleted them because they wouldn't be useful to what I intended to do. Additionally, the instructions on Kaggle where the data set was collected noted that those two columns were unnecessary and could be deleted.
- The next thing I did was check for null values so that they could be addressed. No null values were found.

- In step 2.3. I deleted the first column. This column was labelled “CLIENTNUM” and it was pretty much a client identifier. This feature will not be useful in this process, so it had to go.
- Next up, I checked the data type for each column. I did this to ensure that the right data type was in each column. For example, the “Customer_Age” column is supposed to be a numerical value and therefore contain the int64 data type. Upon checking the data type for that column, if object was listed, I would have known that there was data in that column that I needed to check, change or address. All the data types in each column were the right data types. Meaning nothing needed to be addressed.
- In step 2.5 I checked the categorical columns to see which values they contained. The values weren’t listed in the page that the data set was downloaded from. I found that “Education_Level,” “Marital_Status” and “Income_Category” contained a value named “unknown.” I equated this to a null value and tried to address it later on.
- In this step I wanted to see how many “unknown” values were in each of the three columns. The results were
    - Education_Level = 1,519,
    - Marital_Status = 749
    - Income_Category = 1,112

  Since the “unknown” value was in columns containing non-numeric categorical values, I couldn’t take the mean of the columns and replace the unknown values with that (as we had done in class exercises). I couldn’t just delete the rows containing the “unknown” values either. Deleting these rows would have meant deleting a possible 3,380 rows. The data set has a total of 10,000 rows. So deleting the unknown values

would have meant deleting almost a third of the data set. I had no intention of doing that.

- Finally, I created dummy variables and made sure that the data type foe the dummy variables was int64 just in case I used a model that only accepted numerical values.
- My final data frame, "bank_df3" contained only numerical values (int64 and float64).

Milestone 3

I chose to use a gradient boosting classifier model for the project. My reason for choosing this classifier model was because it was known for its high accuracy, its ability to handle missing data well, and its ability to model complex relationships.

So, after splitting the data into training and testing sets, I proceeded to train the gb_classifier model on the training set. This can be seen in step 3.3. Next was to test the accuracy of the model. After writing the code to test the accuracy, I got an accuracy of 1.00 for the Gradient Boosting Model. An accuracy of 1 was very much so ideal. It indicated that the model had achieved 100% accuracy on the test set. In other words, every single prediction made by the model on the test data was correct. However, I became very suspicious as I felt that a perfect accuracy value was more a sign of potential issues with the model. To test this, I did some further evaluations on the trained gradient boosting model. I generated a classification report and a confusion matrix. The result can be seen in Fig 5. This too showed that the model had perfectly predicted both classes on the test set. All 1699 instances of class 0 were correctly identified. All 327 instances of class 1 were correctly identified. The precision, recall, and F1-score for both classes are 1.00, indicating perfect performance. All of this seemed very suspicious, so I decided to generate the feature importance according to the model. The feature importance is shown in Fig 6 and Fig 7. It showed that

Attrition_Flag_Existing Customer had an importance value of 1.000000e+00 (or 1.0). This meant that the model found this feature to be the most important by far, with a normalized importance value of 1.0. This feature contributed the most to the model's decisions. All other features had importance values that were effectively zero or very close to zero (e.g., 2.229150e-14, 1.594734e-14). This suggested that these features were not significantly contributing to the model's decisions compared to the most important feature. Many features had an importance value of exactly 0.000000e+00. This indicated that these features did not contribute at all to the model's predictions. The model essentially ignored these features. All of this told me that the model relied almost entirely on the Attrition_Flag_Existing Customer feature to make predictions. Which meant that I had a "faulty" model. As a result of this, I decided to ignore this model and try another.

Next up I tried the simpler logistic regression model. I chose it because it was simple, interpretable, fast to train and predict and it required less computational power. So, I trained the logistic regression model and then evaluated it. The result can be seen in Fig 8. The accuracy was approximately 87.17%, meaning that the model correctly classified about 87.17% of the test samples. Fig 8 also showed that the model was quite good at predicting non-churn cases but needed improvement in predicting churn cases. The feature importance chat (Fig 9) for the logistic regression showed that Attrition_Flag_Existing Customer, Contacts_Count_12_mon, Months_Inactive_12_mon, Total_Relationship_Count and Dependent_count in that order, were the top most important features. Negative coefficients; features like Total_Relationship_Count, Months_on_book, and Total_Trans_Ct with negative coefficients indicate that higher values of these features reduce the likelihood of churn. Positive coefficients; features like Contacts_Count_12_mon, Months_Inactive_12_mon, and Dependent_count with positive coefficients indicate that higher values of these features increase the likelihood of churn.

Bank Credit Card Churn

Conclusion

My first model, the GB Classifier was a huge failure as it only used a single feature to predict churn. The second model, the logistic regression model, fared much better. However, even this model didn't perform as I would have hoped. As stated earlier, the logistic regression model was very good at predicting non-churn cases but it needed much more improvement in its prediction of churn cases. Which is ultimately what I want it to be good at predicting.

Ultimately, neither models are ready for deployment. If I had the opportunity, I would have loved to test other different models. Another factor that I think might have affected the models is the fact that in the data, the churn cases only accounted for 16% of the data. Maybe the models needed more "churners" for a better more effective training. At the very least, I can confidently say that contacts_count_12_mon, months_inactive_12_mon, total_relationship_count, dependent_count, customer_age, months_on_book and total_trans_ct are all features to focus on when trying to determine a possible churner.

Reference

- Goyal, S. (2020, November 19). Credit Card customers. Kaggle. https://www.kaggle.com/datasets/sakshigoyal7/credit-card-customers
- Product Plan. (2021, August 12). Churn. What is Churn? | Definition and Overview | Product Metrics. https://www.productplan.com/glossary/churn/#:~:text=Churn%20is%20the%20measure%20of,quarterly%2C%20or%20annual%20churn%20rate

Bank Credit Card Churn

Charts and Diagrams

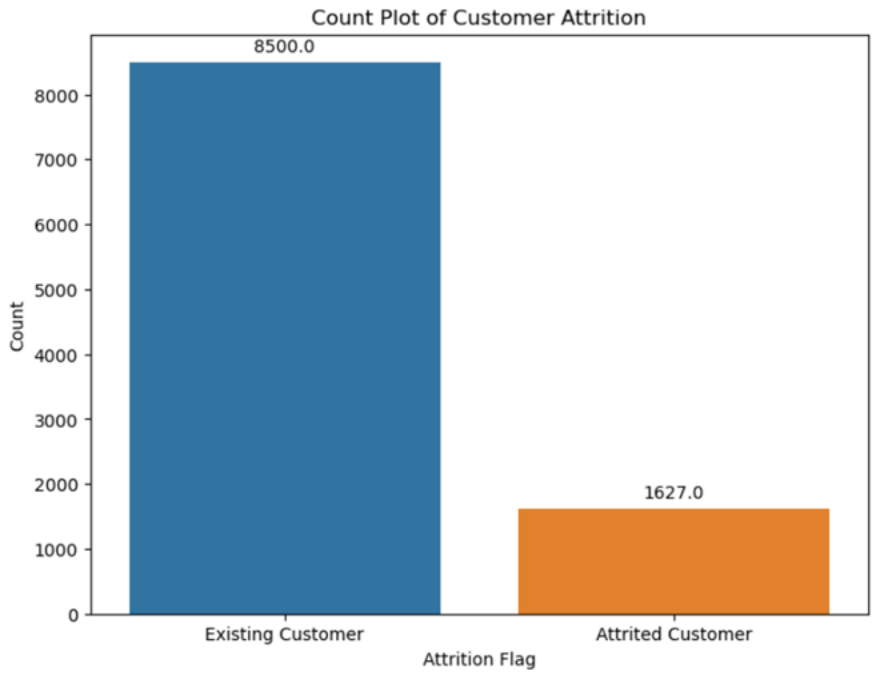

Fig 1

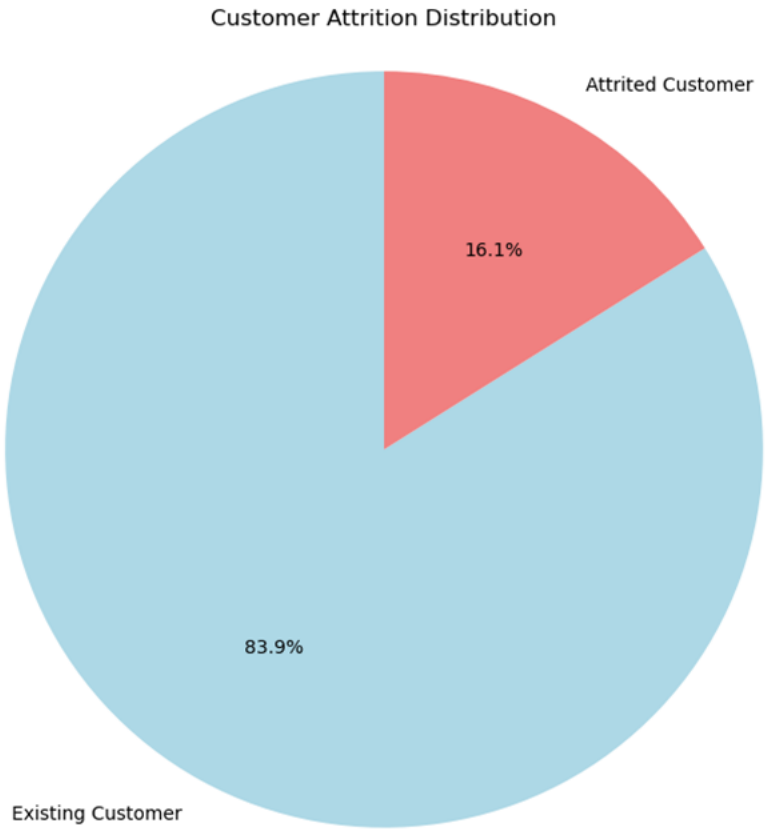

Fig 2

Bank Credit Card Churn

Fig 3

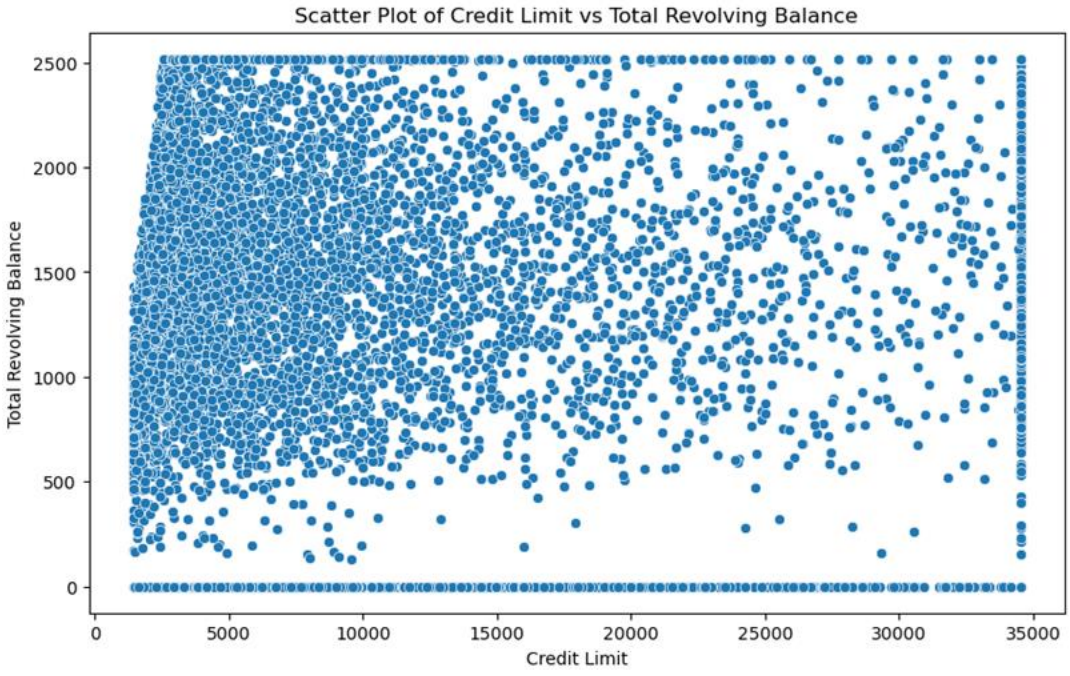

Fig 4

Bank Credit Card Churn

```
Classification Report:
              precision    recall  f1-score   support

           0       1.00      1.00      1.00      1699
           1       1.00      1.00      1.00       327

    accuracy                           1.00      2026
   macro avg       1.00      1.00      1.00      2026
weighted avg       1.00      1.00      1.00      2026

Confusion Matrix:
[[1699    0]
 [   0  327]]
```

Fig 5

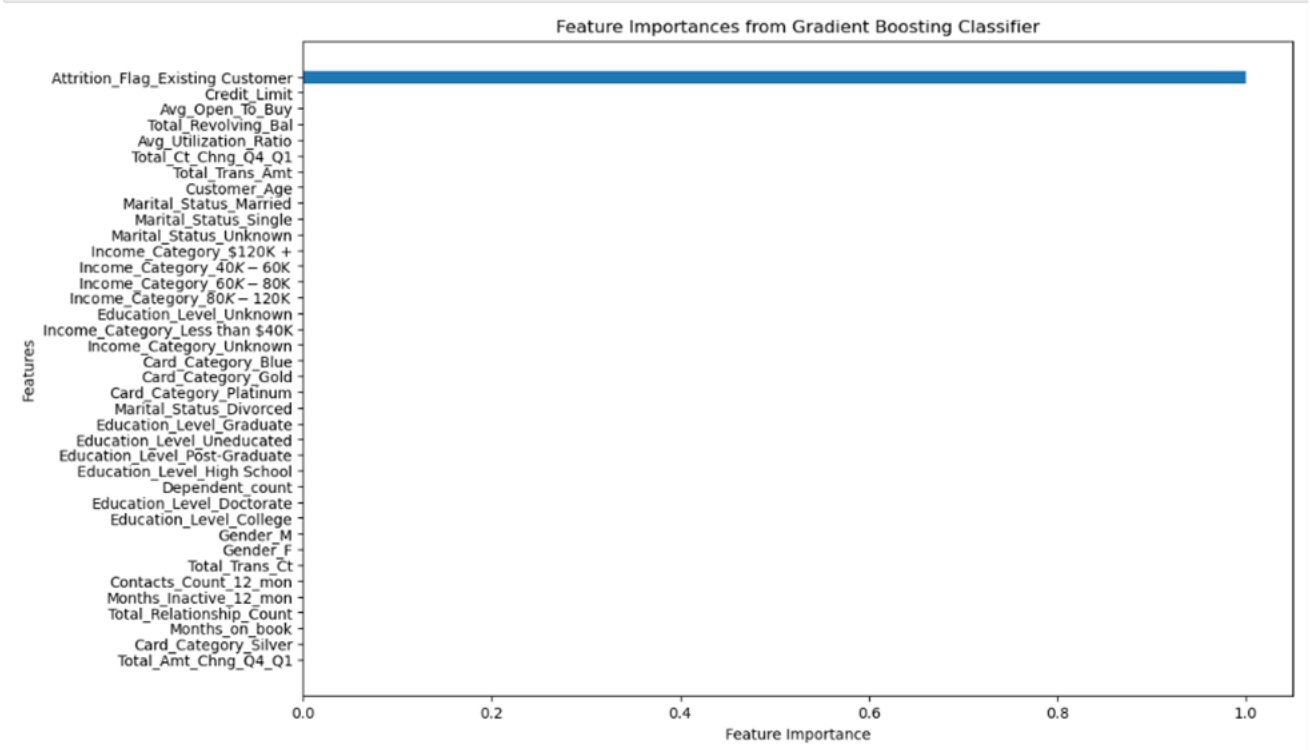

Fig 6

	Feature	Importance
14	Attrition_Flag_Existing Customer	1.000000e+00
6	Credit_Limit	2.229150e-14
8	Avg_Open_To_Buy	1.594734e-14
7	Total_Revolving_Bal	8.304532e-15
13	Avg_Utilization_Ratio	1.304907e-15
12	Total_Ct_Chng_Q4_Q1	9.323992e-16
10	Total_Trans_Amt	9.193127e-16
0	Customer_Age	0.000000e+00
25	Marital_Status_Married	0.000000e+00
26	Marital_Status_Single	0.000000e+00
27	Marital_Status_Unknown	0.000000e+00
28	Income_Category_$120K +	0.000000e+00
29	Income_Category_$40K - 60K	0.000000e+00
30	Income_Category_$60K - 80K	0.000000e+00
31	Income_Category_$80K - 120K	0.000000e+00
23	Education_Level_Unknown	0.000000e+00
32	Income_Category_Less than $40K	0.000000e+00
33	Income_Category_Unknown	0.000000e+00
34	Card_Category_Blue	0.000000e+00
35	Card_Category_Gold	0.000000e+00
36	Card_Category_Platinum	0.000000e+00
24	Marital_Status_Divorced	0.000000e+00
19	Education_Level_Graduate	0.000000e+00
22	Education_Level_Uneducated	0.000000e+00
21	Education_Level_Post-Graduate	0.000000e+00

Fig 7

```
Accuracy: 0.8716683119447186
Confusion Matrix:
[[1627   72]
 [ 188  139]]
Classification Report:
              precision    recall  f1-score   support

           0       0.90      0.96      0.93      1699
           1       0.66      0.43      0.52       327

    accuracy                           0.87      2026
   macro avg       0.78      0.69      0.72      2026
weighted avg       0.86      0.87      0.86      2026
```

Fig 8