CUSTOMER CHURN PREDICTION

Problem:

Before starting the project, it is important to understand the business use case. We might want to ask certain questions to get a clarity on the purpose of this project. Some queries to put forward are:

- What exactly is the business objective?
- Will this model be fed to another machine learning pipeline?
- How does the company expect to use and benefit from the model?

Answers to these will help us in model and its performance metrics selection.

The objective of this assignment is to build a model to predict whether a customer will churn or not. The performance metrics used is F1 score. F1 score is harmonic mean of recall and precision. Greater the F1 score better is the model's performance.

The provided dataset has the following parameters:

Variable	Description
ID	Unique Identifier of a row
Age	Age of the customer
Gender	Gender of the customer (Male and Female)
Income	Yearly income of the customer
Balance	Average quarterly balance of the customer
Vintage	No. of years the customer is associated with bank
Transaction_Status	Whether the customer has done any transaction in the past 3 months or not
Product_Holdings	No. of product holdings with the bank
Credit_Card	Whether the customer has a credit card or not
Credit_Category	Category of a customer based on the credit score
Is_Churn *(target variable)*	Whether the customer will churn in next 6 months or not

Solution:

We know that this is a binary classification problem. The dependent features are of different datatypes. We have 4 categorical and 6 numerical features (barring the ID as it is just for the identification purpose and won't be used for model building).

Approach:

We have followed the below steps:

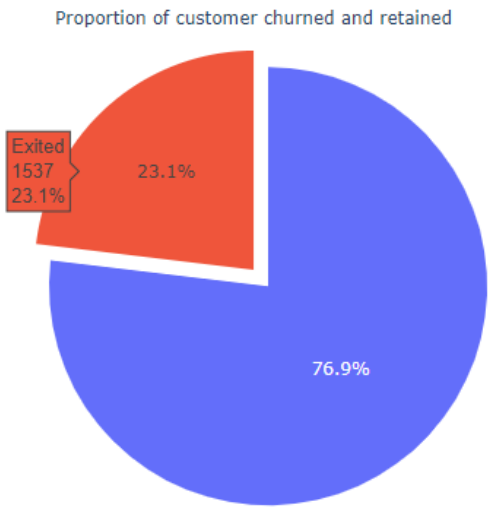
1. Loading and Understanding the data
2. Exploratory Data Analysis
3. Feature Engineering
4. Model Building
5. Model's Performance Comparison
6. Feature Importance
7. Conclusion
8. Next Steps

We have imported the required libraries as and when needed

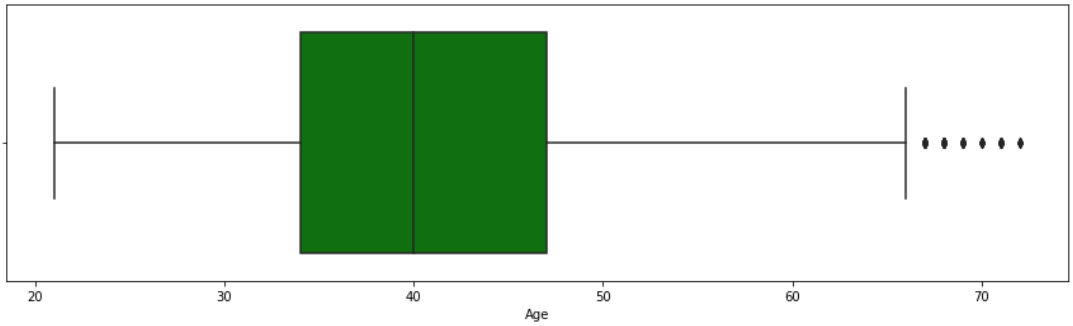
1. Load Data: As a very first step we have loaded the data using pandas read_csv. In this step we get a broad understanding of the train and test dataset. We were able to answer certain questions like the number of rows and columns in both the set, whether any missing values exist and datatypes of all the features.

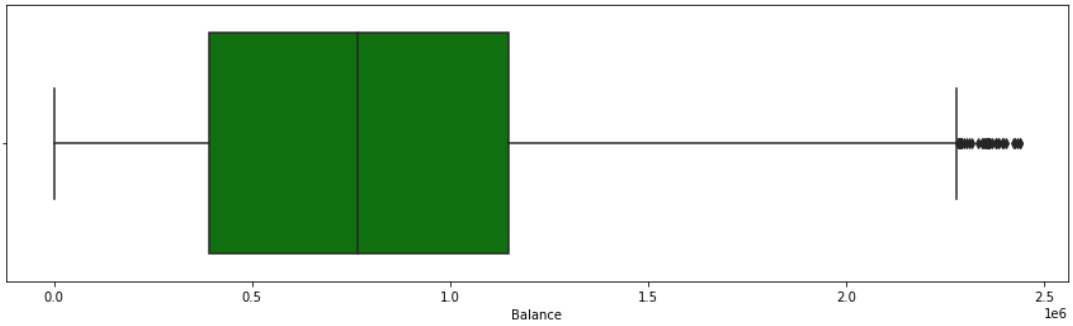
2. EDA: Under exploratory data analysis we explored each of the features closely. Categorical and numerical features were combined and both the groups were investigated separately. We performed both univariate and bivariate analysis to understand the spread and distribution of the data within a feature and also the correlation with other independent features and also with the target. We have used matplotlib and seaborn in abundance for visual interpretations along with plotly express. We drew a lot of important conclusions from the analysis. Key findings:

   Imbalanced Target Class

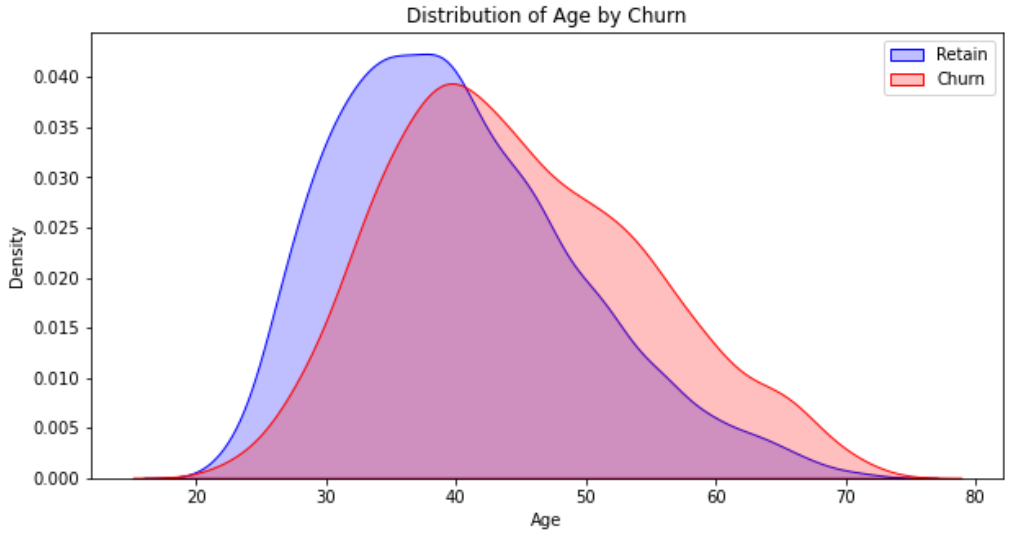


Outliers in the features Age and Balance





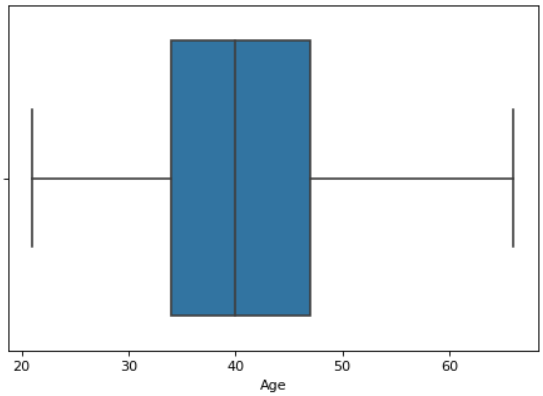
Elder customers are more likely to churn as compared to younger ones.

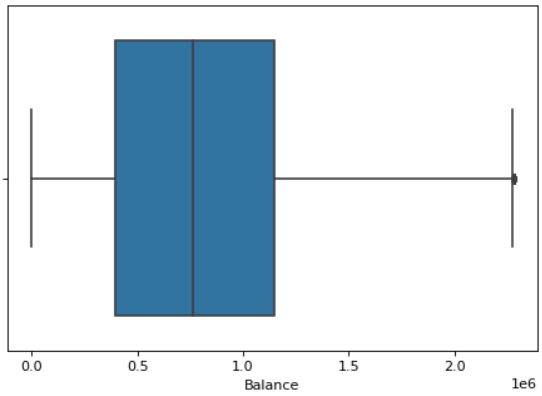


3. <u>Feature Engineering</u>: From the conclusions drawn in the previous section, we have performed different operations in the feature engineering step. Started with handling outliers for age and balance features followed by grouping the values of the same features into different categories. The latter is called binning (categorizing the values in different bins). Binning converted our numerical variables to categorical so we encoded all the non-numerical features using one-hot (features with more than 3 groups) and label encoding techniques.

   Illustration of binning and feature extraction:

Handled the outliers



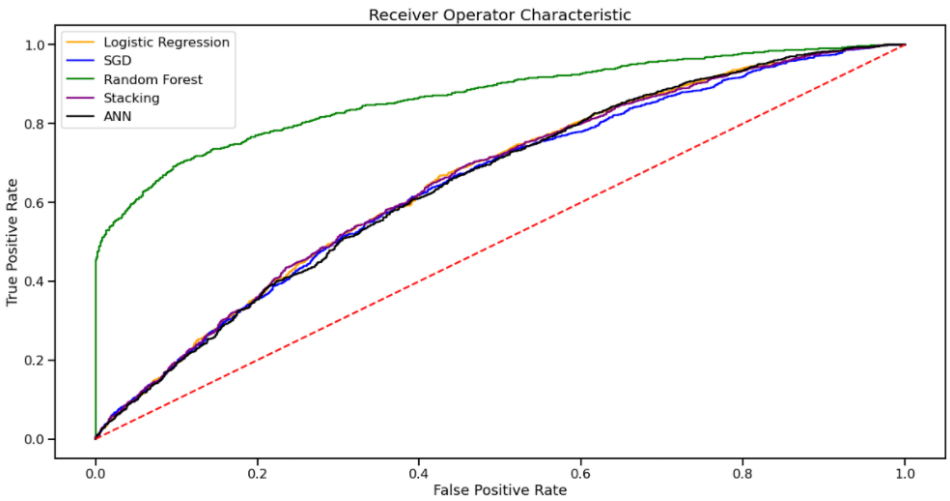


Scaling is an integral part of feature engineering process. It is done to scale down the features, which means if the features have different units the values may vary largely and to bring them to the same scale there are methods like Normalization and Standardization. We have used MinMaxScaler to scale down the vintage, age and balance features in our dataset.

4. Over-sampling: To balance the target variable classes, we have used SMOTE over-sampling technique because we had the not churned to churned ratio in the ratio of around 5:1. We also tried another over-sampling technique ADASYN which gave better results.

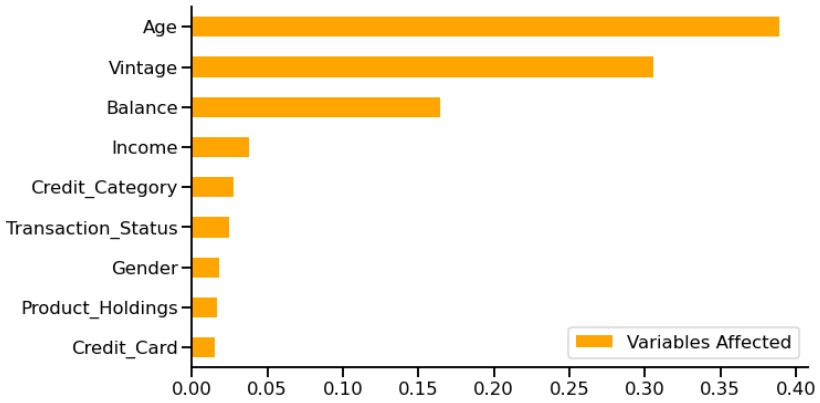
5. Base Model: Before starting off with our base model we divided our train data into train and test sets to evaluate our model's performance. We have taken the simple logistic regression algorithm for our base model.

6. Model Comparison: To improve the performance of our model and to ensure that we select the best one, we have taken into consideration ensemble techniques and compared their performance by trying out different hyperparameter tuning techniques. We chose the staking algorithm for the final submission.



7. Feature Importance

We can leverage random forest's feature scores function that shows the strength of each feature in determining the target variable. This helps business to focus more on the strong features and invest less time and money on the least important features.



8. Summary:

In this classification problem the dataset given was quite clean, it had no missing values and a lot of information regarding the customers was provided.
It was a binary classification problem with only 2 classes in the target feature (0: not churned, 1: churned). The classes were imbalanced and the models were predicting all 0s in the target feature. To resolve this, we had used over-sampling technique.
We used different encoding techniques depending on the feature type. We used Label Encoder for ordinal values whereas one hot encoding for the nominal ones.
The Age and Balance features were binned and encoded later on.

One important takeaway is new customers are more likely to churn and also customers with high income are hard to retain. The Bank can focus on these 2 groups to bring down the churn rate.

9. Next Steps:

To improve the model performance and to get higher accuracy the below things can be done:
A model's performance increases with increase in data (clean and relevant). If we can feed in more data we can achieve higher accuracy.
We can try extracting features like (Product_Holdings - Credit_Card) assuming credit card as one of the products we can fetch how many other products the customer is holding.
Experimenting with under-sampling to see whether there is any change in model performance
We can use hyperparameter tuning for the other algorithms like we have done for Random Forest

Trying other ensemble techniques like XGBoost and CatBoost might also help