Architectural Decision Document:
Credit Card Churn Prediction

Alif Sussardi

January 18, 2021

1 Overview

Credit Card Churn Prediction was carried out as follow:

1. Dataset: Use and Case
2. Exploratory Data Analysis
3. Extract, Transform and Load
4. Feature Creation
5. Model Definition
6. Model Training
7. Model Evaluation
8. Model Deployment

2 Dataset and Use Case

2.1 Data Choice and Use Case

The purpose of this Project is to classify the churn of Credit Card customer given several parameters as shown in Column Name. The data was obtained from Kaggle: Churn Data. As a company who issue credit card, it is important that we know whether customer will be able to continue their transactions (not defaulted) and keep using our service. The pre-registered features such as Gender, Income Category, Total Relationship Count, can be an indicator whether a customer will be able to sustain our service. However, here, I will use all the parameters which are available for the customer that has already registered to see what type of customer will sustain, therefore once we spot a customer with such characteristic, we can reach to them, send them offer to keep using our services. The parameters that are available in the data is as follow:

1. CLIENTNUM = Client number. Unique identifier for the customer holding the account.
2. Attrition Flag = THIS IS THE TARGET PARAMETER.
3. Customer Age.

4. Gender = Demographic variable - M=Male, F=Female.
5. Dependent count = Demographic variable - Number of dependents.
6. Education Level.
7. Marital Status.
8. Income Category = Annual Income Category of the account holder (< 40K, 40K - 60K, 60K - 80K, 80K - 120K, > 120K, Unknown).
9. Card Category = Product Variable - Type of Card (Blue, Silver, Gold, Platinum).
10. Months on Book = Period of relationship with bank.
11. Total Relationship Count = Total no. of products held by the customer.
12. Months Inactive 12 mon = No. of months inactive in the last 12 months.
13. Contacts Count 12 mon = No. of Contacts in the last 12 months.
14. Credit Limit.
15. Total Revolving Bal = Total Revolving Balance on the Credit Card.
16. Avg Open To Buy = Open to Buy Credit Line (Average of last 12 months).
17. Total Amt Chng Q4-Q1 = Change in Transaction Amount (Q4 over Q1).
18. Total Trans Amt = Total Transaction Amount (Last 12 months).
19. Total Trans Ct = Total Transaction Count (Last 12 months).
20. Total Ct Chg Q4-Q1 = Change in Transaction Count (Q4 over Q1)
21. Avg Utilization Ratio = Average Card Utilization Ratio

2.2 Justification

This data was chosen because it is interesting and important to determine which feature and which Machine Learning or Neural Network are the best in predicting the churn.

3 Exploratory Data Analysis

From this exploratory data analysis, we can see how the data spread, and whether some algorithm might not be appropriate.

3.1 Correlation Matrix

From the correlation matrix below we can identify which features are the most important to explain the Attrition Flag. From the Figure 1 we can see that the Total Transaction, Total Count of Change Q4-Q1, Total Revolving Balance, Contact count, Total Transaction Amount are correlated to the Attrition Flag. Lets see What they are looks like later. For now We can see how these data behaved. The features, however, can be divided into Categorical and Numerical. And will be shown as follow. (Only important data are shown)

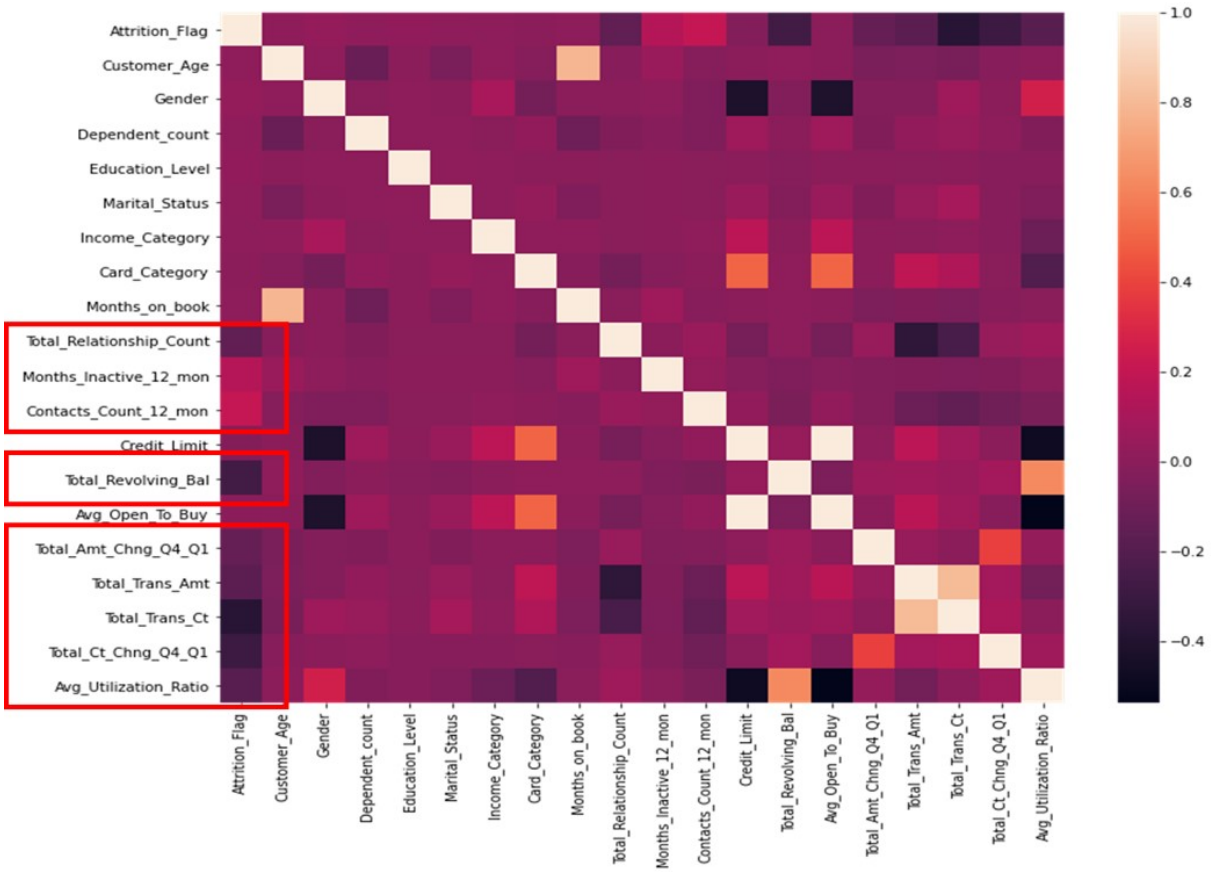

Figure 1: Correlation matrix

3.2 Features Outlook

3.2.1 Attrition Flag

It can be seen that the Attrition Flag distribution is not great, with the count of Existed cases 5 times the Attrited cases. This type of data might be bad for making a model as it will introduce bias towards the Existed, means the machine will predict Existed customer better than the Attrited due to having more training.

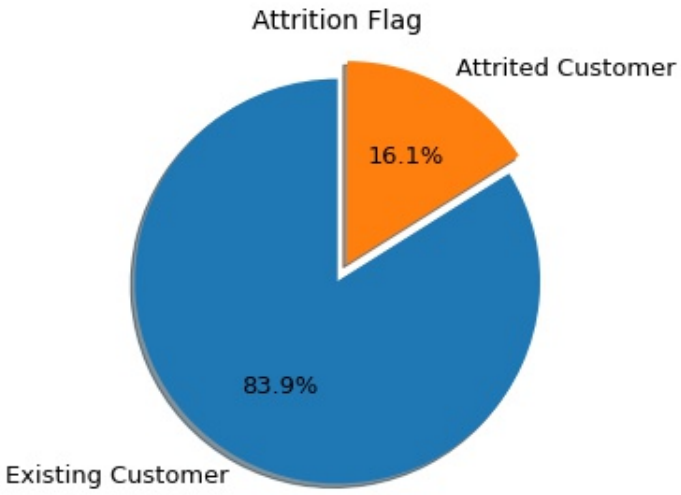

Figure 2: Attrition Flag

3.2.2 Total Transaction Count

The Total transaction count data is not normally distributed, which is not good if we want to use standard scaler. It looks like there are 3 main groups around 40, 80 and 110 transaction amount.

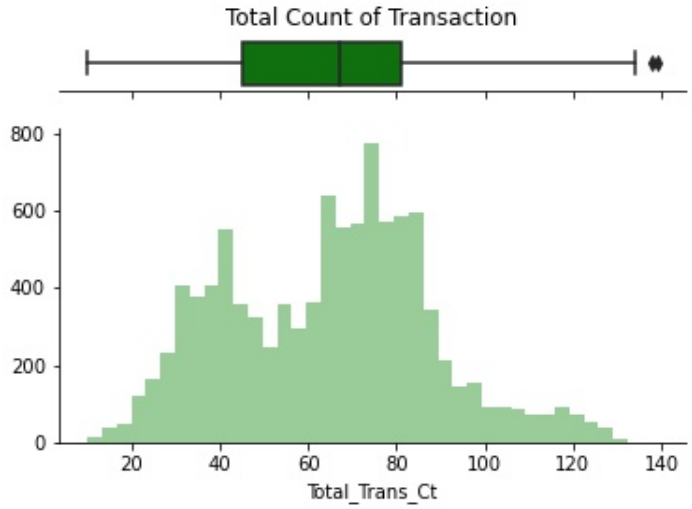

Figure 3: Gender Ratio

3.2.3 Average Utilization Ratio

The Average utilization ratio is also not normally distributed, with most of the user not using their card! This type of behaviour is also not good for StandardScaler scaling model.

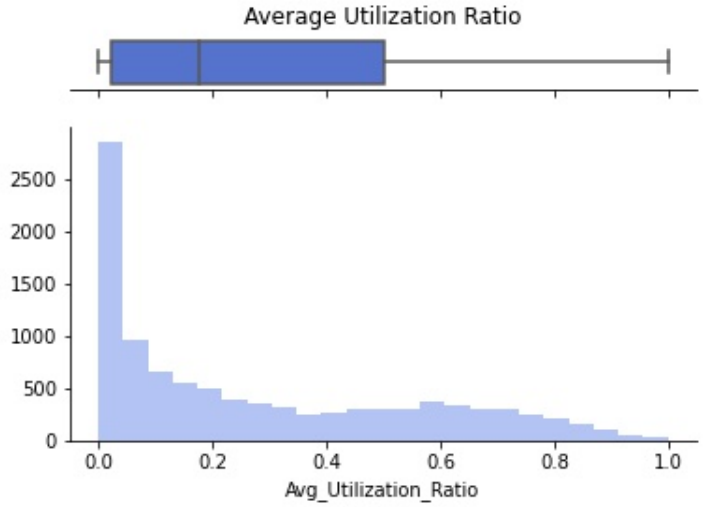

Figure 4: Utilization Ratio

3.2.4 Change from Q1 to Q4

The Total Amount of change is normally distributed, with a small kurtosis on the right hand side of the graph.

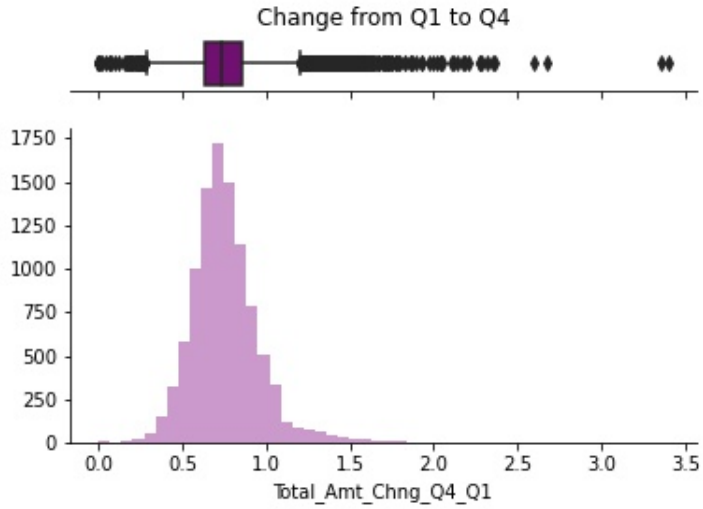

Figure 5: Total Change Q1-Q4

3.2.5 Total Relationship

The total number of relationship does not seem like have any pattern. But it can be seen that most of the customer has at least 3 relationships, meaning they have several cards at once.

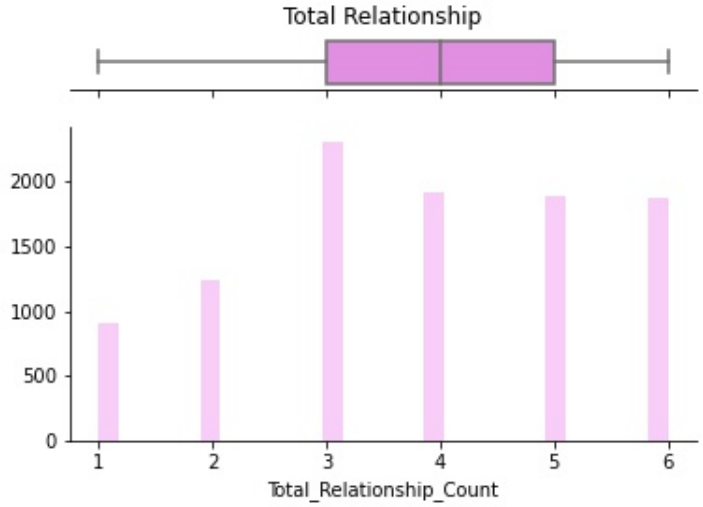

Figure 6: Total relationship

3.2.6 Total Transaction Amount

It can be seen clearly that there are at least 4 groups. One at 2000 USD,at 4500 USD, 8000 USD and 15000 USD. This data cannot be scaled using StandardScaler as well.

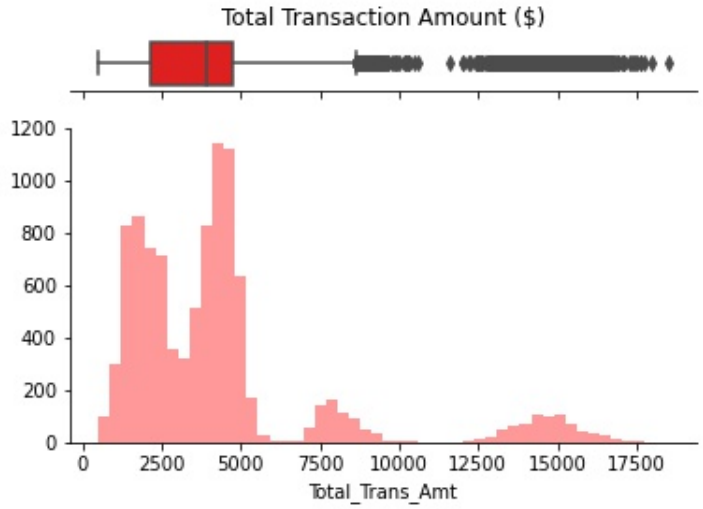

Figure 7: Total Transaction

3.2.7 Contact Count

Number of contact in 12 months are evenly distributed between 2 and 3. Meaning that, the customer mostly contacting our Credit Card company 2 or 3 times. Better be preparing for that Customer Service Call! If we want to add more customer!

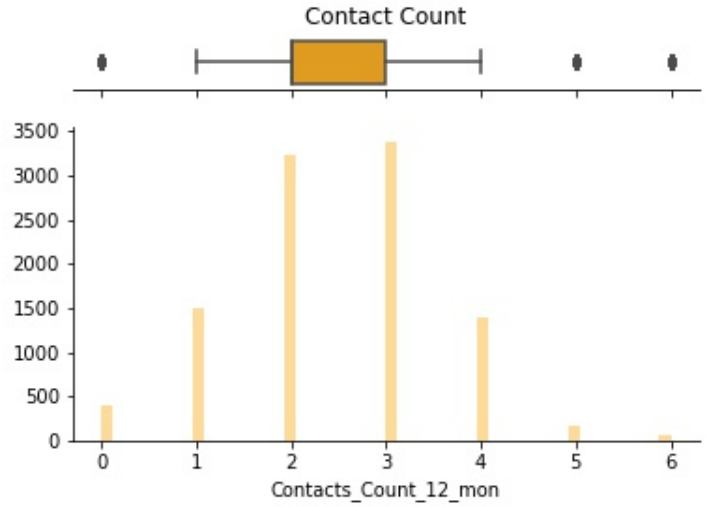

Figure 8: Contact Count

3.2.8 Total Revolving Balance

This quite strange, because most of the people does not have any balance, while the rest of the people has average of about 1500 USD with large variation from 500 to 2500, but the affluent customer are second to the one without balance.

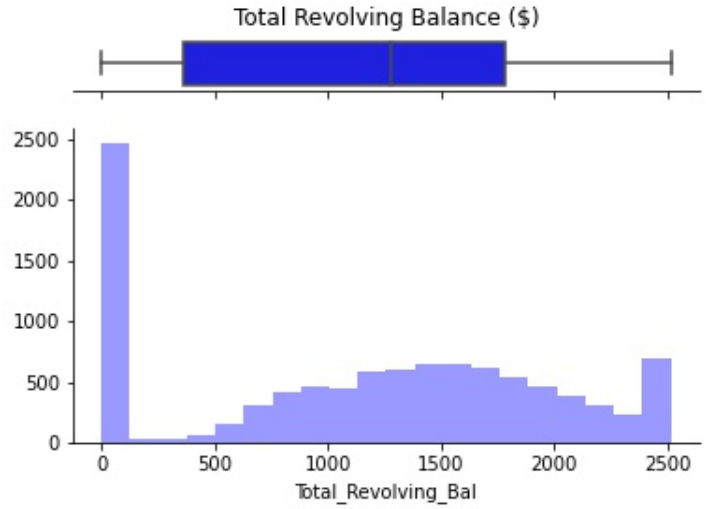

Figure 9: Total Revolving Balance

3.2.9 Months Inactivity

This is the weirdest amongst all. Lets take a look, there are seems to be a cut at 3 months inactivity here. Is it that marketing guy again, who did some campaign 3 months ago? or most customer just cancel their account after 3 months?

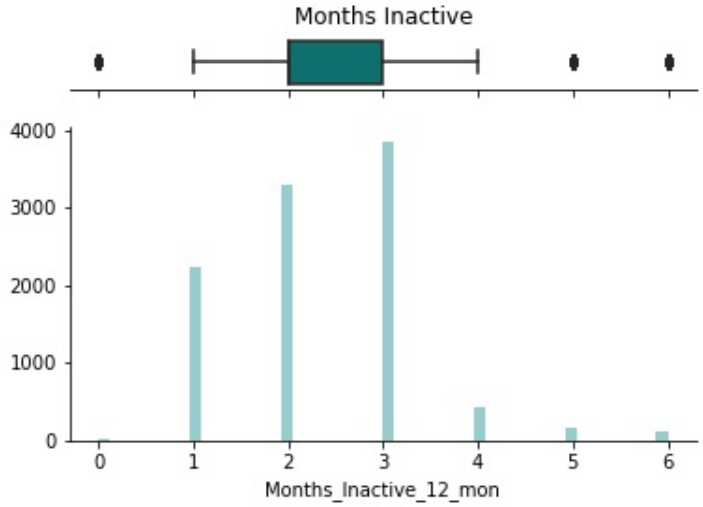

Figure 10: Month Inactivity

4 Extract, Transform & Load (ETL)

4.1 Technology Choice

All operations in this data analysis and modeling were done in **Jupyter Notebook**, using **Google Cloud Server**.
The data were imported from the BankChurner.csv file downloaded from kaggle, as mentioned in the introduction.

4.2 Feature Creation

All data transformation were done using **pandas** and **numpy** as they are the most commonly used tools and easy to use. The Categorical data were transformed with **pandas factorize** as follow:

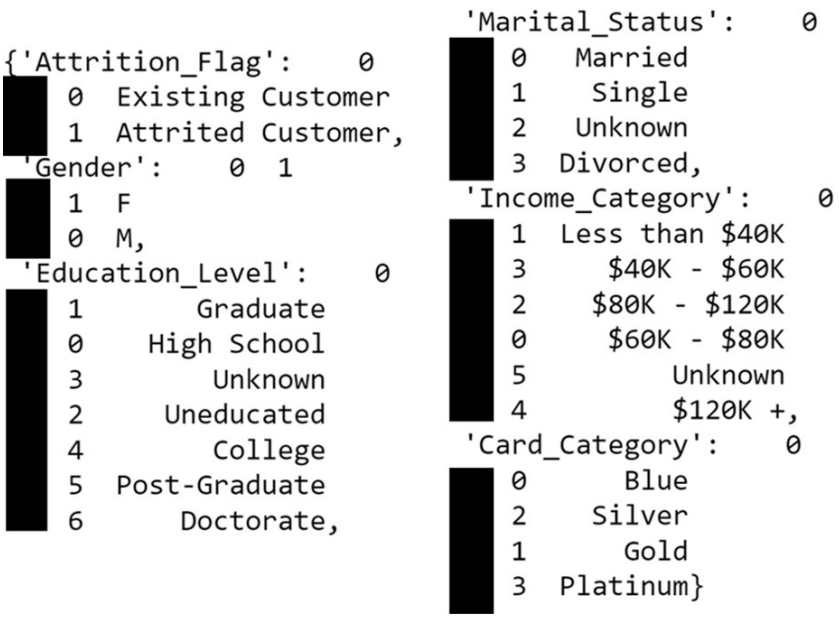

Figure 11: Categorization

4.3 Data splitting and Synthesis

The date were then transformed into array with **numpy** and scaled using **sklearn MinMaxScaler**.
After scaled, the data then split into 80% train and 20% test. Later the train data split again into train (80%) and validation (20%) data. This also means that the test data consist of 20%, the train is 64% and 16% of the validation from the total data provided.

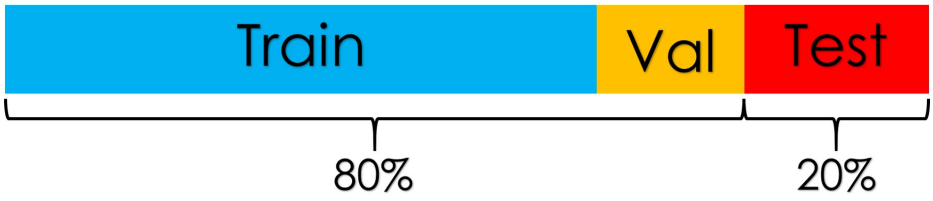

Figure 12: Train, Validation & Test data

Data were synthesized using **imbalanced-learn SMOTE** algorithm on the original data to obtain the same number of Attrited and Existed data:

```
Old df -> Existed:  8500 & Attrited 1627
New df -> Existed:  8500 & Attrited 8500
```

Figure 13: Old & New Data

4.4 Justification

Jupyter Notebook were used as it is easy and can be run line-by-line in a separate server. In fact, the server that I was using was **Google Cloud Server**, as it is significantly cheaper (three month promos with $300 balance) without limitation of CPU's usage or power, compared to IBM Watson.
Pandas factorize was used as opposed to **sklearn one-hot-encoder** as I want to do modeling fast, in a sacrifice of data clarity, the factorization or label encoder will introduced hierarchial bias on each category, but it will have significantly smaller feature than onehot encoder.
Test data of 20% was chosen as I think 20% is enough number to reveal whether the model will overfit or not.
sklearn MinMaxScaler was used to normalize the data as we can see from the Exploratory data analysis that the distribution are not normal/Gaussian, using StandardScaler (data divided by standard deviation) algorithm will make it bad, as the standard deviation cannot reflect the spread of the data from the mean.

5 Model Definition

5.1 Machine Learning

The supervised machine learning classifiers were used, including:

1. **Random Forest Classifier** by **scikit-learn**,
2. **Logistic Regression** by **scikit-learn**,
3. **XGBoost Classifier** from **xgboost** package,
4. **Light Gradient Boosting Machine** from **lightgbm** package.

Both original and synthesized data were fed into the machine learning on their default parameters.

5.2 Neural Network

Deep learning were carried out using **tensorflow keras** machine.

5.2.1 Long Neural Network

The long Neural Network is a Sequential neural network, consisting 6 hidden layers, as follow:

```
model1 = Sequential()
model1.add(Dense(256, kernel_regularizer=regularizers.l2(0.001), input_dim=19, activation='relu'))
model1.add(Dropout(rate=0.2))
model1.add(Dense(128, kernel_regularizer=regularizers.l2(0.001), activation='relu'))
model1.add(Dropout(rate=0.2))
model1.add(Dense(64, kernel_regularizer=regularizers.l2(0.001), activation='relu'))
model1.add(Dropout(rate=0.2))
model1.add(Dense(32, kernel_regularizer=regularizers.l2(0.001), activation='relu'))
model1.add(Dropout(rate=0.2))
model1.add(Dense(16, kernel_regularizer=regularizers.l2(0.001), activation='relu'))
model1.add(Dropout(rate=0.2))
model1.add(Dense(8, kernel_regularizer=regularizers.l2(0.001), activation='relu'))
model1.add(Dropout(rate=0.1))
model1.add(Dense(1, activation='sigmoid'))
```

Figure 14: Long Neural Network

5.3 Wide Neural Network

The wide neural network is also Sequential neural network, consisting of only 3 hidden layer, but with significantly more neuron, as shown below:

```
model2 = Sequential()
model2.add(Dense(256, kernel_regularizer=regularizers.l2(0.001), input_dim=19, activation='relu'))
model2.add(Dropout(rate=0.2))
model2.add(Dense(512, kernel_regularizer=regularizers.l2(0.001), activation='relu'))
model2.add(Dropout(rate=0.2))
model2.add(Dense(64, kernel_regularizer=regularizers.l2(0.001), activation='relu'))
model2.add(Dropout(rate=0.1))
model2.add(Dense(1, activation='sigmoid'))
```

Figure 15: Wide Neural Network

5.4 Justification

Random Forest and Logistic regression were chose as they are the most common machine learning, and also simplest. On the other hand, the XGBoost and LGBM were chosen as they are slightly more advanced.
The Deep learning were varied for Wide and Long Neural Network to compare which one is the more effective algorithms.

6 Model Training

6.1 Machine Learning

Machine Learning algorithms were trained using original train data and synthesized data using SMOTE.

6.2 Deep Learning

Binary Crossentropy were used to calculate the loss, using ADAM optimizer.

```
model1.compile(loss = "binary_crossentropy",
                  optimizer = 'adam',
                  metrics=['accuracy'])
```

Figure 16: Compile

The model were trained using scaled data, both the original and after synthesised using SMOTE, and validation data were used for the validation, with 150 epochs.

```
model1.fit(X_train_scaled, y_train,
           validation_data=(X_val_scaled, y_val),
           epochs=150, batch_size=32, verbose=1)
```

Figure 17: Fit

6.3 Justification

Algorighm were trained using Train and evaluated using Validation data (Original and SMOTEd) to avoid over-fitting. Binary Crossentropy were chosen as the target or predicted feature is binary, and it takes into account for both Attrited and Existed.
ADAM was used as it is the fastest optimizer with reasonable ability to reach least gradient with avoiding local minimum. It was excellent!
150 epochs were chosen as it was thought that after 150 cycle, the accuracy or loss will converge.

7 Model Evaluation

All Machine Learning models were evaluated using cross validation method on its training data with 5 cross validation batches. Both Machine Learning and Deep Learning methods were also evaluated using test data to see its evaluations scores (Accuracy, F1 and Recall).

7.1 Machine Learning Confusion Matrix

By comparing the two sets of Original and SMOTEd data, we can see clearly that the proportion of correct guess are better in the SMOTEd Data.

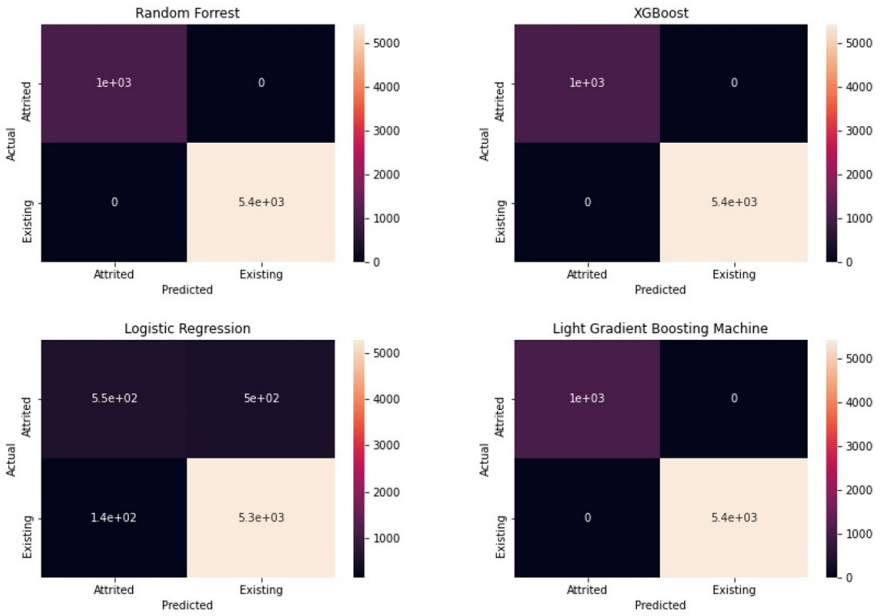

Figure 18: Confusion matrix: Machine Learning Original Data

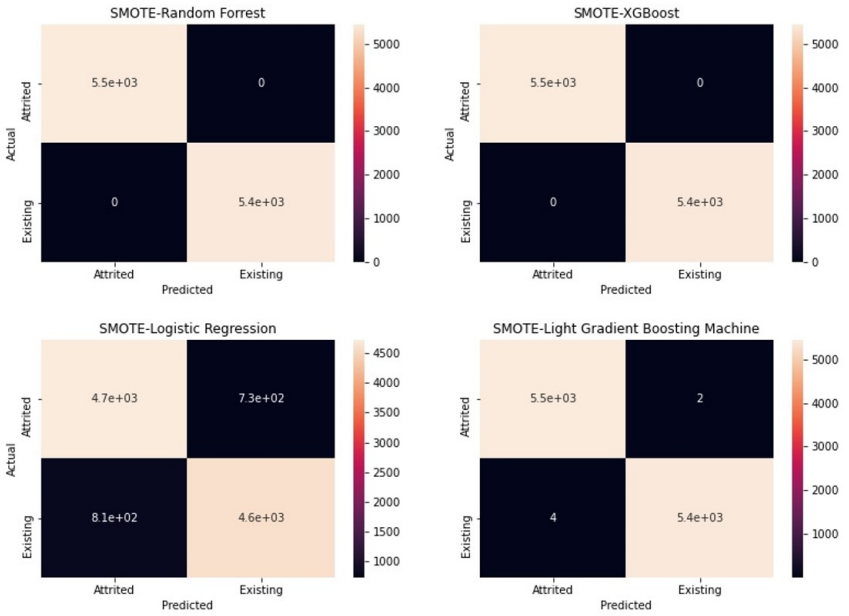

Figure 19: Confusion Matrix: Machine Learning SMOTEd Data

7.2 Neural Network Chronological Accuracy & Loss and Confusion Matrix

7.2.1 Long Neural Network

It can be seen that the SMOTEd data performed better with smaller fluctuation and also better distribution of confusion matrix.

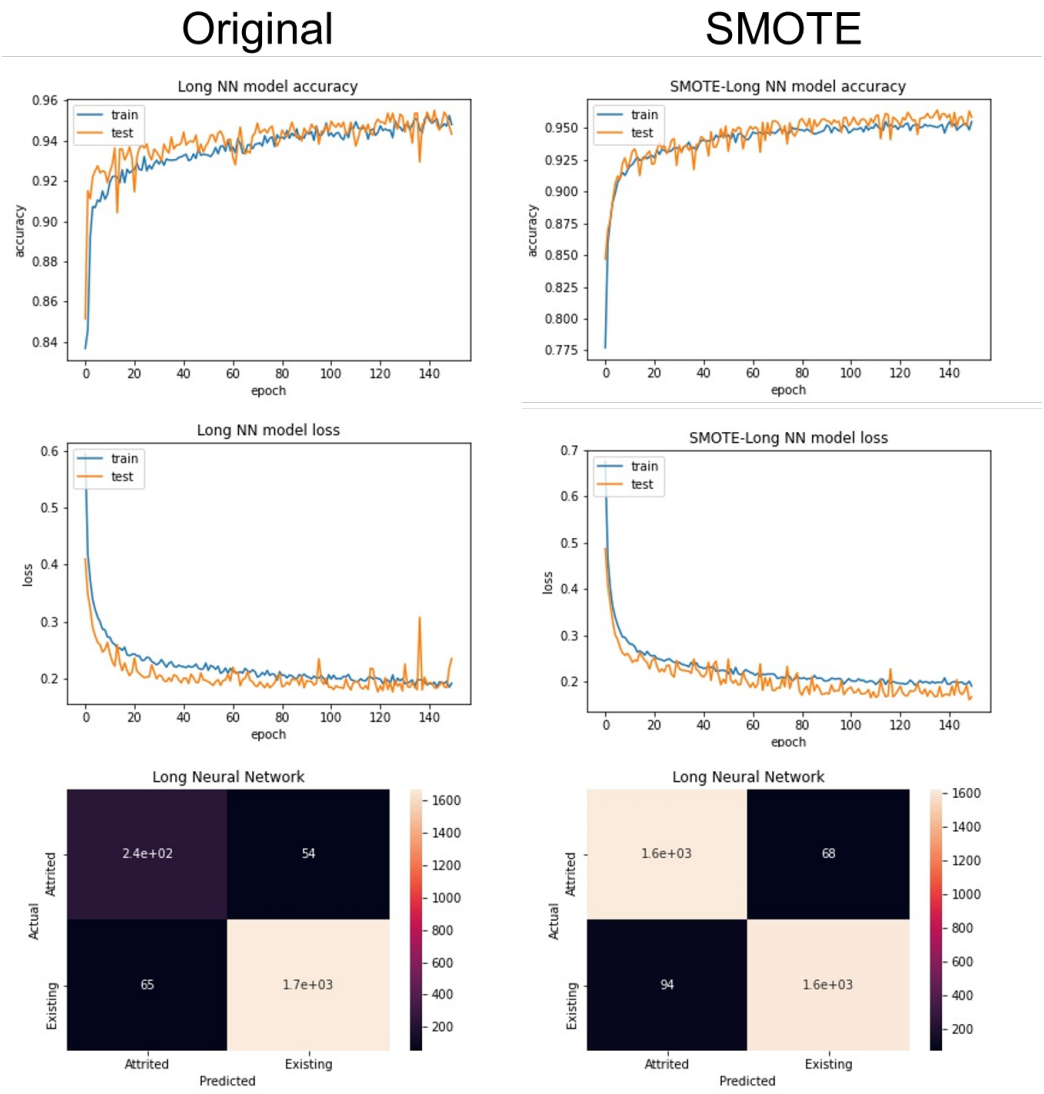

Figure 20: Long Neural Network

7.2.2 Wide Neural Network

It can be seen that the SMOTEd data performed better with smaller fluctuation and also better distribution of confusion matrix.

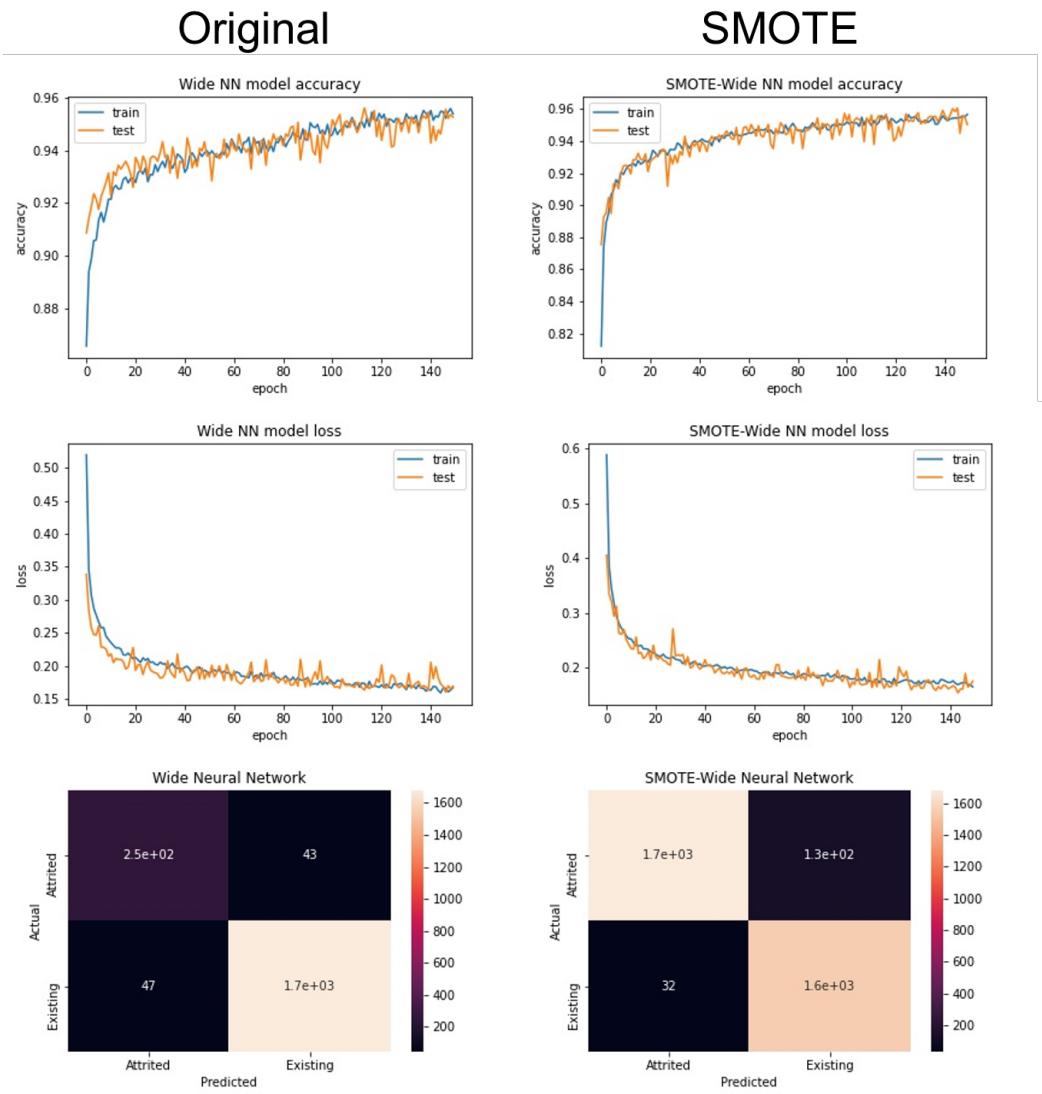

Figure 21: Wide Neural Network

7.3 Machine Learning and Deep Learning Accuracy, F1 Score and Recall

It can be seen that models trained with SMOTEd data are performing really well due to the absence of training frequency bias. Wide Neural Network can perform really well with high F1 score and Recall. LGBM Machine Learning Also performing excellently with overall 98% scores. Perfect!
Cross Validated accuracy are also very close to the Accuracy by test data, which is a good sign that the model are not over-fitted! Excellent!

Model	Accuracy	Cross Val Accuracy	f1 Score	Recall
SMOTE-Wide NN	94.71	NaN	95.34	98.89
SMOTE-LGBM	97.85	98.21	97.88	98.19
SMOTE-XGBoost	97.94	98.18	97.96	98.08
SMOTE-Random Forest	97.24	97.45	97.27	97.55
SMOTE-Long NN	94.82	NaN	94.38	92.89
LGBM	96.89	98.21	89.48	89.04
XGBoost	96.99	98.18	89.75	88.70
SMOTE-Logistic Regression	86.38	85.74	86.67	87.65
Wide NN	94.45	NaN	83.31	87.04
Random Forest	96.45	97.26	87.37	82.72
Long NN	94.94	NaN	78.95	69.77
Logistic Regression	91.12	85.74	64.43	54.15

Figure 22: Summary

7.4 Justification

Confusion Matrix can tell us directly the proportion of the Predicted and Actual data, whether there are any Falsely classified data. Accuracy parameters can tell us how well the Prediction matched with the Actual value. F1 and Recall can tell us how well the classification of the True and Predicted. The better the values reach 100% the better the model is.

8 Model Deployment

The machine Learning models were deployed and saved in "sav" files as follow:
Original

Machines	Name	Filename
Random Forest	rf	rf.sav
Logistic Regression	lr	lr.sav
XGBoost	xgb	xgb.sav
Light GBM	lgbm	lgbm.sav

SMOTEd

Machines	Name	Filename
Random Forest	rfsm	rfsm.sav
Logistic Regression	lrsm	lrsm.sav
XGBoost	xgbsm	xgbsm.sav
Light GBM	lgbmsm	lgbmsm.sav

The deep learning models were deployed and saved in a "json" files and the weights were saved in "h5" formats, as follow:

Machines	Name	model	weight
Long NN	nn_model	nn_model.json	nn_model.h5
Wide NN	nn_model2	nn_model2.json	nn_model2.h5
SMOTEd-Long NN	nn_model1sm	nn_model1sm.json	nn_model1sm.h5
SMOTEd-Wide NN	nn_model2sm	nn_model2sm.json	nn_model2sm.h5

8.1 Justification

The models were saved in files as it will be easier to distributed and saved in a local repository or GitHub. They can be easily loaded and used should there are any new upcoming data.
Jupyter Notebook file the "*.ipynb" file is also saved, but even without it, with the help of this ADD, the next data scientist will be able to easily replicate the analysis and model building process.

9 Summary

The model has been successfully built with LGBM is the best performing Machine Learning and Wide Neural Network as the best Deep Learning algorithm.
SMOTE tool is essential to synthesis unequal data, and to get a better training and remove bias. The worst is Logistic Regression. Do not use it!