CUSTOMER CHURN PREDICTION

December, 2021

Presentation outline

- Data validation and cleaning
- Data exploration and visualization
- Model Experimentation
- Model validation
- Conclusion

Data validation and cleaning

Data cleaning and validation

Observations:

1) No duplicate values
2) No missing values, we need to investigate deeper.

Check for duplicate values

```
# check for duplicate samples
customer_churn.duplicated().sum()
```

```
0
```

```
# check missing values
customer_churn.isnull().sum()
```

```
cust_id                                 0
income                                  0
debt_with_other_lenders                 0
credit_score                            0
has_previous_defaults_other_lenders     0
num_remittances_prev_12_mth             0
remittance_amt_prev_12_mth              0
main_remittance_corridor                0
opened_campaign_1                       0
opened_campaign_2                       0
opened_campaign_3                       0
opened_campaign_4                       0
tenure_years                            0
churned                                 0
dtype: int64
```

Data cleaning and validation

Observations:

1) Due to wrong data type, some missing values were hidden.
2) After correcting the data type, missing values were identified.

Detecting and correcting datatypes and rechecking missing values

```
# check missing values and datatype
customer_churn.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 7432 entries, 0 to 7431
Data columns (total 14 columns):
 #   Column                              Non-Null Count  Dtype  
---  ------                              --------------  -----  
 0   cust_id                             7432 non-null   int64  
 1   income                              7432 non-null   object 
 2   debt_with_other_lenders             7432 non-null   object 
 3   credit_score                        7432 non-null   object 
 4   has_previous_defaults_other_lenders 7432 non-null   int64  
 5   num_remittances_prev_12_mth         7432 non-null   int64  
 6   remittance_amt_prev_12_mth          7432 non-null   float64
 7   main_remittance_corridor            7432 non-null   object 
 8   opened_campaign_1                   7432 non-null   int64  
 9   opened_campaign_2                   7432 non-null   int64  
 10  opened_campaign_3                   7432 non-null   int64  
 11  opened_campaign_4                   7432 non-null   int64  
 12  tenure_years                        7432 non-null   float64
 13  churned                             7432 non-null   int64  
dtypes: float64(2), int64(8), object(4)
memory usage: 813.0+ KB
```

```
# check the data type to ensure its corrected
customer_churn.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 7432 entries, 0 to 7431
Data columns (total 14 columns):
 #   Column                              Non-Null Count  Dtype  
---  ------                              --------------  -----  
 0   cust_id                             7432 non-null   int64  
 1   income                              7199 non-null   float64
 2   debt_with_other_lenders             7137 non-null   float64
 3   credit_score                        7137 non-null   float64
 4   has_previous_defaults_other_lenders 7432 non-null   int64  
 5   num_remittances_prev_12_mth         7432 non-null   int64  
 6   remittance_amt_prev_12_mth          7432 non-null   float64
 7   main_remittance_corridor            7432 non-null   object 
 8   opened_campaign_1                   7432 non-null   int64  
 9   opened_campaign_2                   7432 non-null   int64  
 10  opened_campaign_3                   7432 non-null   int64  
 11  opened_campaign_4                   7432 non-null   int64  
 12  tenure_years                        7432 non-null   float64
 13  churned                             7432 non-null   int64  
dtypes: float64(5), int64(8), object(1)
memory usage: 813.0+ KB
```

Data cleaning and validation

Observations:

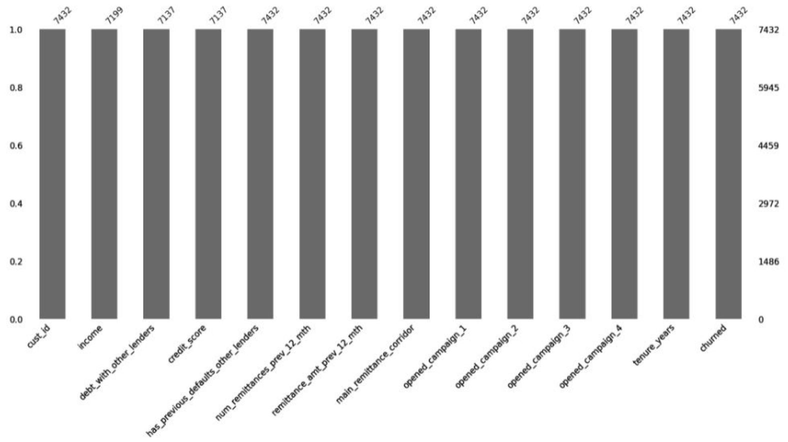
1) Three features possess missing values.

NB: The general rule of thumb is, if the missing value is less than 5% of the entire dataset, it can be dropped.

2) In this case, the missing value is around 3% of the entire dataset.

```
# Re-check missing values
customer_churn.isnull().sum()
```

```
cust_id                                0
income                               233
debt_with_other_lenders              295
credit_score                         295
has_previous_defaults_other_lenders    0
num_remittances_prev_12_mth            0
remittance_amt_prev_12_mth             0
main_remittance_corridor               0
opened_campaign_1                      0
opened_campaign_2                      0
opened_campaign_3                      0
opened_campaign_4                      0
tenure_years                           0
churned                                0
dtype: int64
```



Data cleaning and validation

Outlier detection:

1) Using cross table to check the outlier in income by comparing two features: ‘churned’, and ‘main remittance corridor’.

```
# detect outliers, if non then save
pd.crosstab(customer_churn['main_re
            values=customer_churn['
```

churned	0	1
main_remittance_corridor		
AE_IN	24779.29	10393.28
AE_PH	24088.89	8151.35
AE_PK	24031.26	8633.41

Data cleaning and validation

Summary of data:

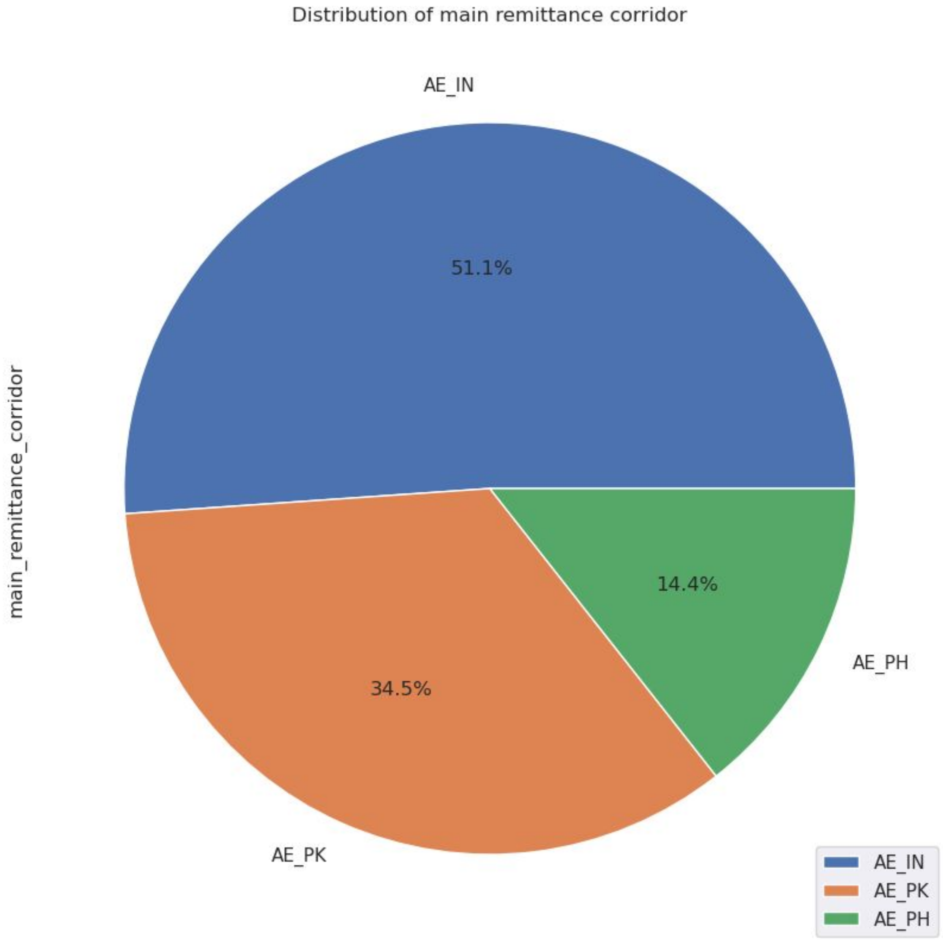
1) 7432 rows, 14 columns, 5 floats features, 8 integer features, and 1 string feature.

```python
# check data type, missing values, and the row counts
customer_churn.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 7432 entries, 0 to 7431
Data columns (total 14 columns):
 #   Column                             Non-Null Count  Dtype  
---  ------                             --------------  -----  
 0   cust_id                            7432 non-null   int64  
 1   income                             7432 non-null   float64
 2   debt_with_other_lenders            7432 non-null   float64
 3   credit_score                       7432 non-null   float64
 4   has_previous_defaults_other_lenders 7432 non-null   int64  
 5   num_remittances_prev_12_mth        7432 non-null   int64  
 6   remittance_amt_prev_12_mth         7432 non-null   float64
 7   main_remittance_corridor           7432 non-null   object 
 8   opened_campaign_1                  7432 non-null   int64  
 9   opened_campaign_2                  7432 non-null   int64  
 10  opened_campaign_3                  7432 non-null   int64  
 11  opened_campaign_4                  7432 non-null   int64  
 12  tenure_years                       7432 non-null   float64
 13  churned                            7432 non-null   int64  
dtypes: float64(5), int64(8), object(1)
memory usage: 813.0+ KB
```

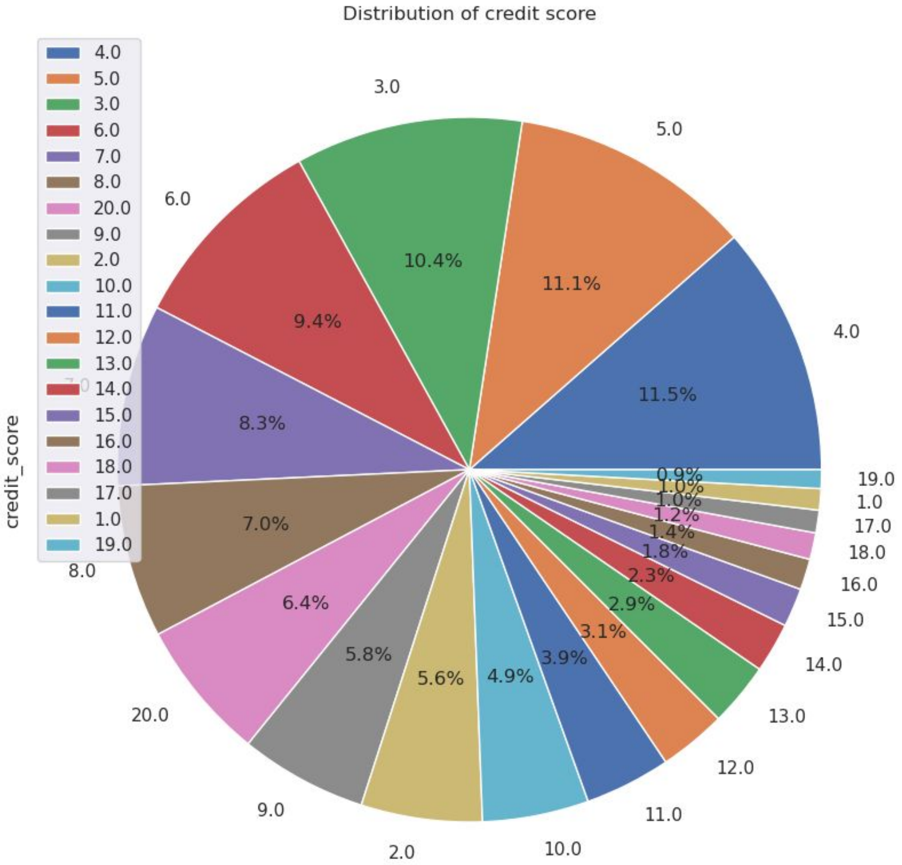
Data exploration and visualization

Data exploration and visualization



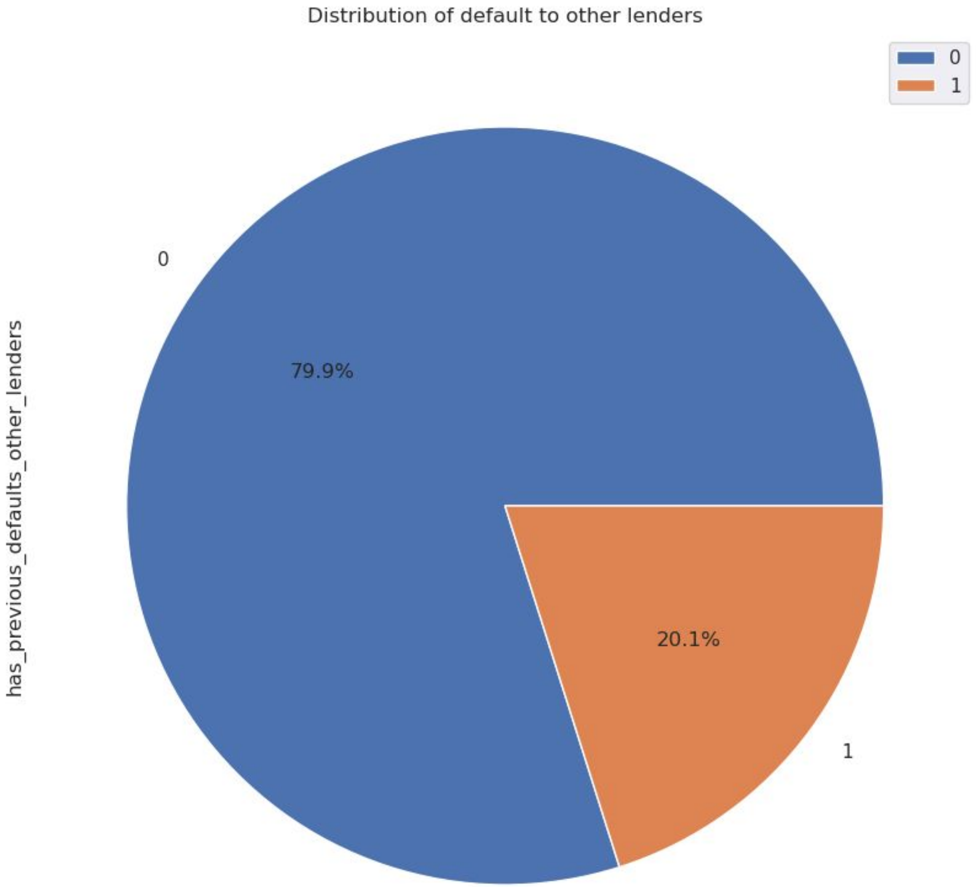
AE_IN corridor holds the dominance with over 50% of customer transaction was held through them.
While AE_PK and AE_PH represent the percentage left.

Data exploration and visualization



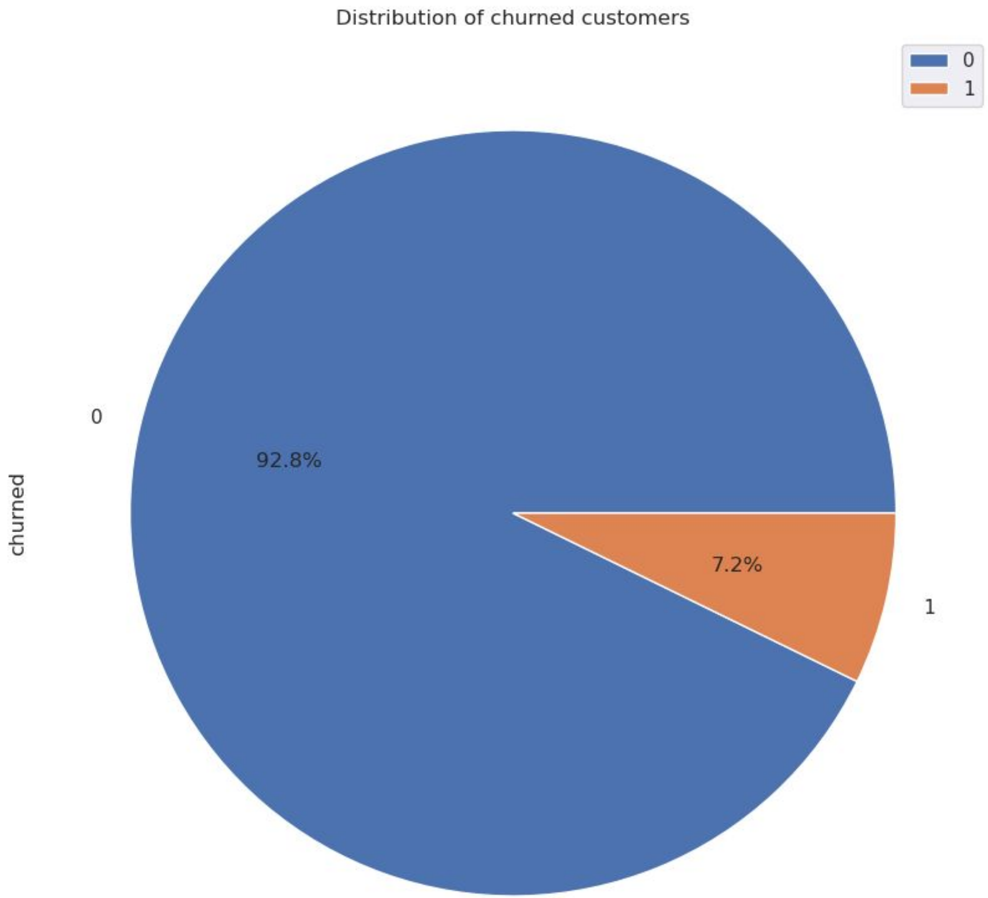
The credit score is dominated by: 3, 4, 5, 6, 7, and 8; taking over 50% of the dataset.

Data exploration and visualization



Around 20% of the customers have defaulted to other customers, while 80% has never defaulted to other customers.

Data exploration and visualization



The label 1 represents the customers that churned while 0, represents customer that didn’t churn.
Furthermore, only 7% percentage has churned, 92% of the customers have never churned.

Data exploration and visualization

```
# cross table on churn status and customers pr
pd.crosstab(ctr_churn['has_previous_defaults_o
```

churned	0	1
has_previous_defaults_other_lenders		
0	4950	353
1	1207	128

```
# Create the cross table from earlier and include tenure years
pd.crosstab(ctr_churn["churned"], ctr_churn["has_previous_defaul
```

	min		max	
has_previous_defaults_other_lenders	**0**	**1**	**0**	**1**
churned				
0	0.000322	0.003510	2.999704	2.998807
1	0.000643	0.004128	1.279770	1.118138

With the aid of cross table, around 5000 of our samples have never defaulted to other lenders and have never churned. While only 128 have defaulted and churned within the selected samples.

Considering the tenure years, customers who churned and defaulted to other lenders takes a maximum of around **a year** to act. While customers who never churned but has defaulted while dealing with other lenders take maximum of 3 years.

It takes around 3 years as well for customers who never churned or defaulted to other lenders to pay back.

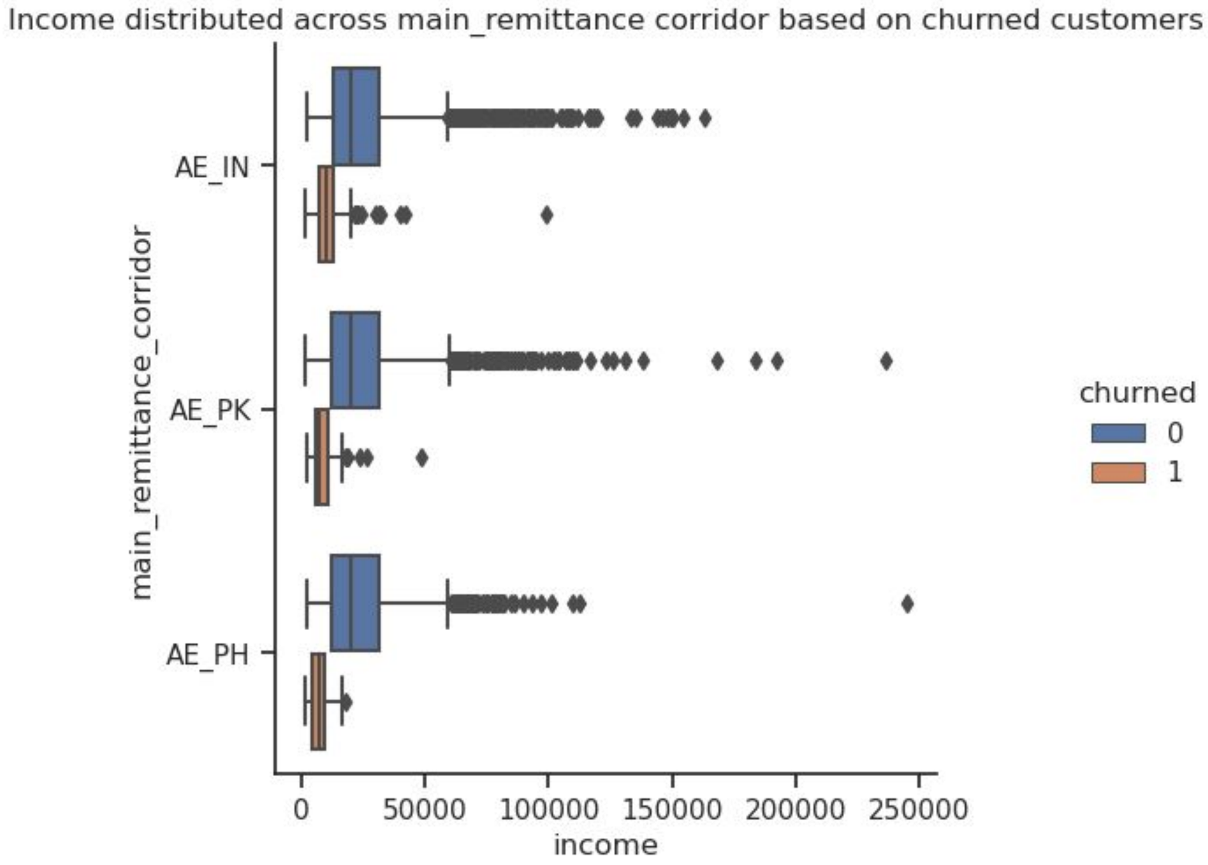
Data exploration and visualization

```
# Create the cross table from earlier and income
pd.crosstab(ctr_churn["churned"], ctr_churn["has_previous_defaults_
```

	min		max	
has_previous_defaults_other_lenders	**0**	**1**	**0**	**1**
churned				
0	2033.114127	1763.726965	244970.92610	65498.32692
1	1434.354208	2130.345902	98758.99741	21863.05765

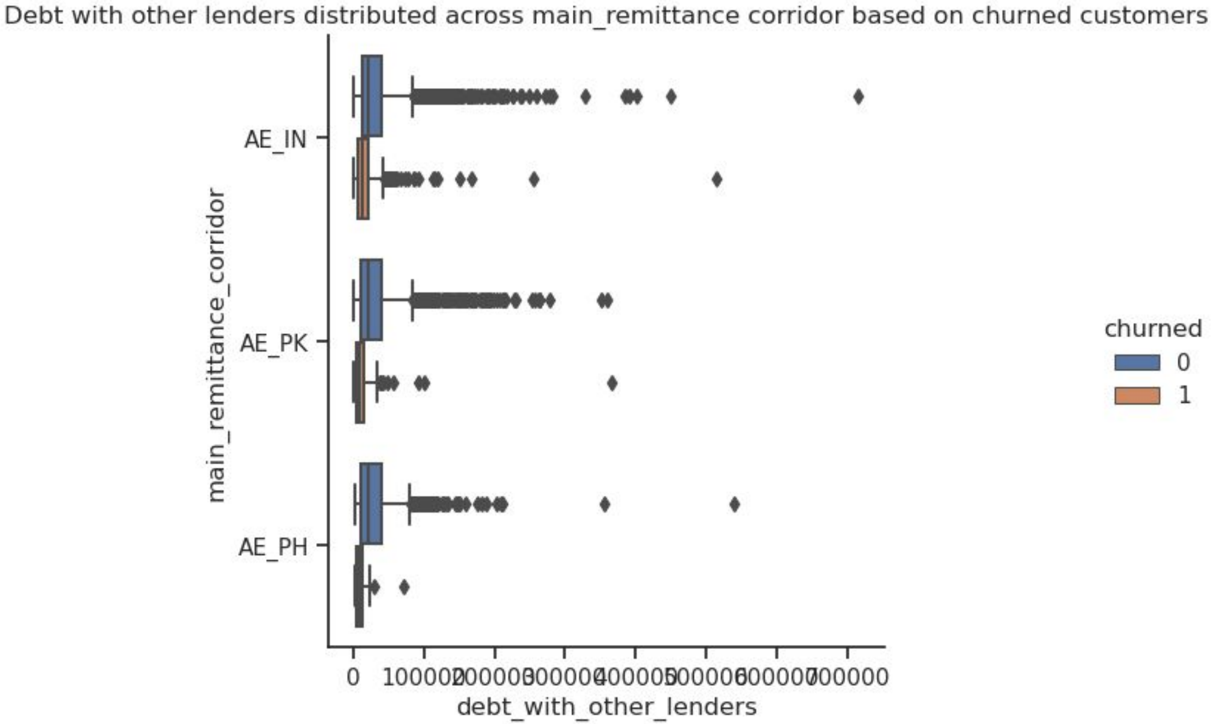
Customers who never churned or defaulted to other lenders, earns 11 times the average income of those that churned and has a previous record of defaulting.

Data exploration and visualization



For the three categories of remittance corridor, customers that never churn earns more compare to those that churned.

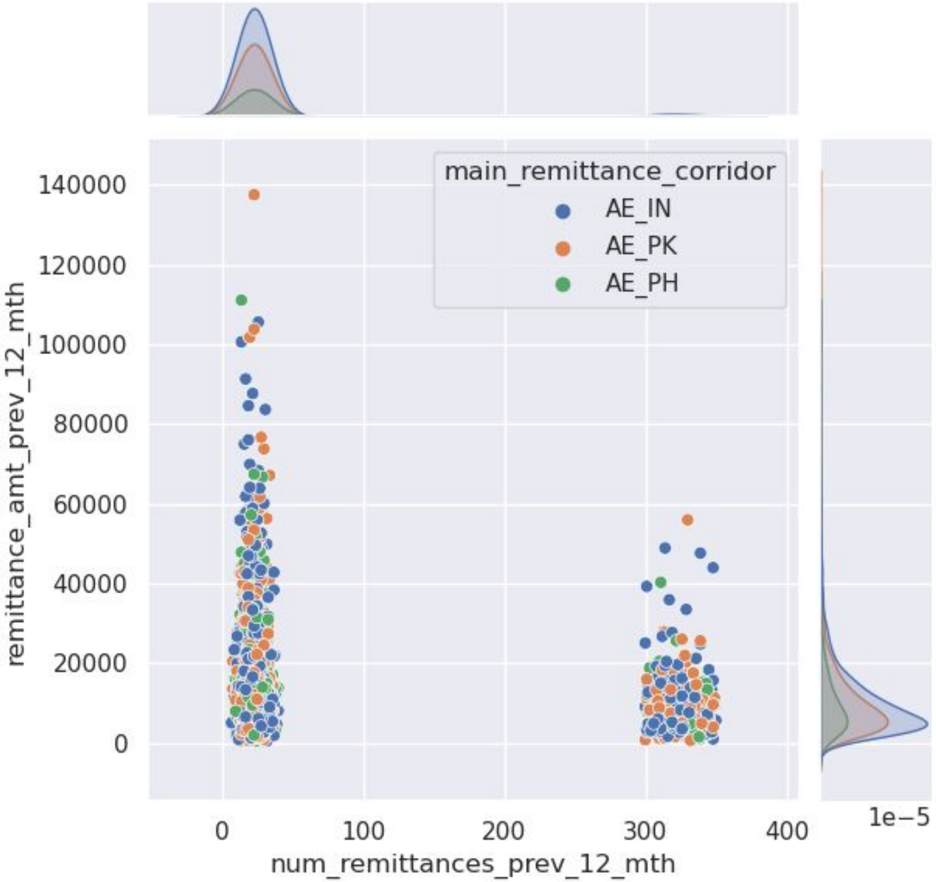
Data exploration and visualization



For the three categories of remittance corridor, customers that never churn also has more debt with lenders compared to those that churned.

Data exploration and visualization

A joint plot of amount remitted vs number of times of remitting funds in 12 months with respect to corridor

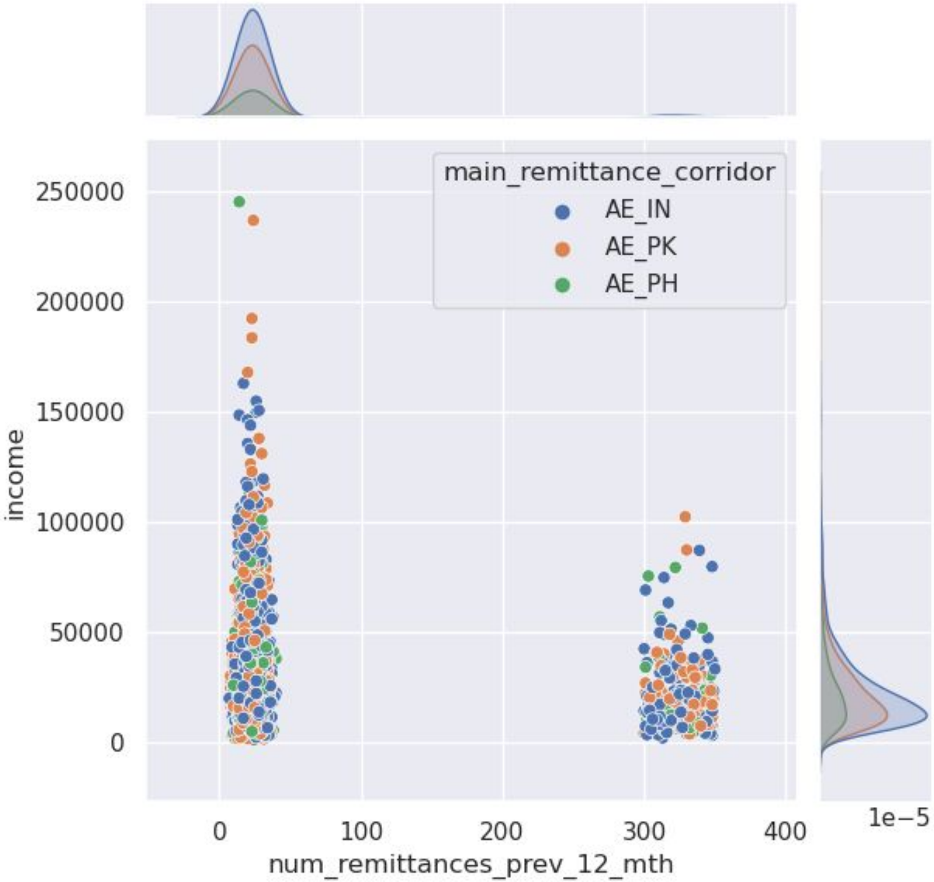


The number of remittance that falls between 0 and 100 remitted between 0 and 60,000. While the number of remittance between 300 and 400, falls between 0 and 20,000.

This indicates that, the lesser the amount to be remitted, the more likely are customers committed to remitting the amount.

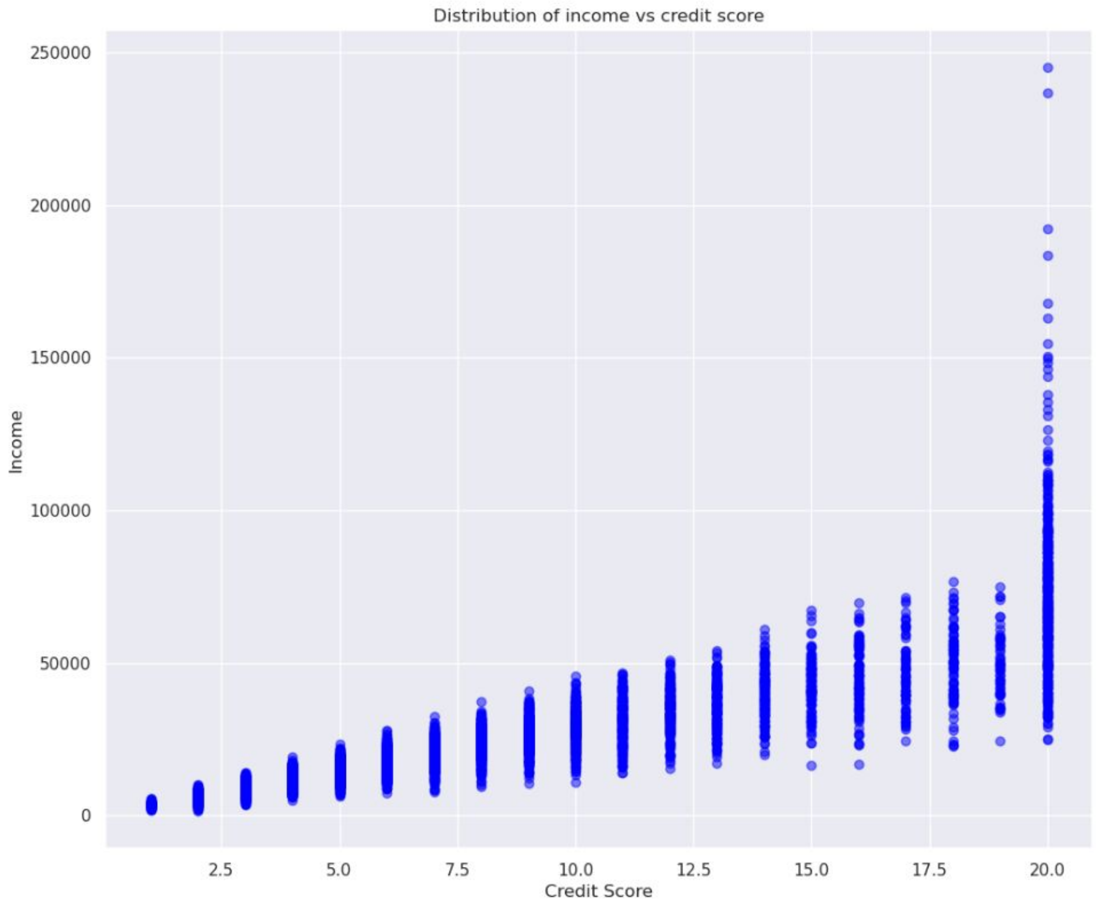
Data exploration and visualization

A joint plot of income vs number of times of remitting funds in 12 months with respect to corridor



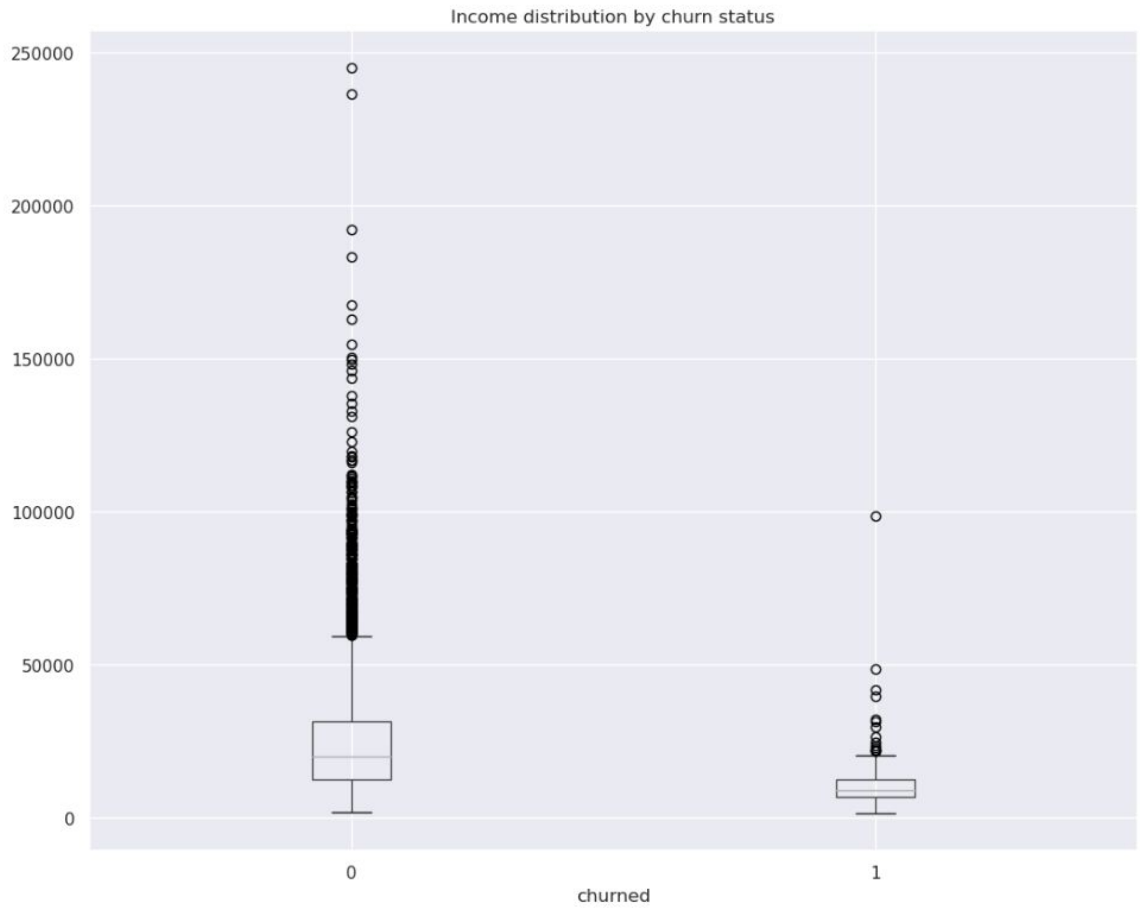
Those clients that earns between 0 and 50,000 are more likely to remit their loans compared to those that earns higher.

Data exploration and visualization



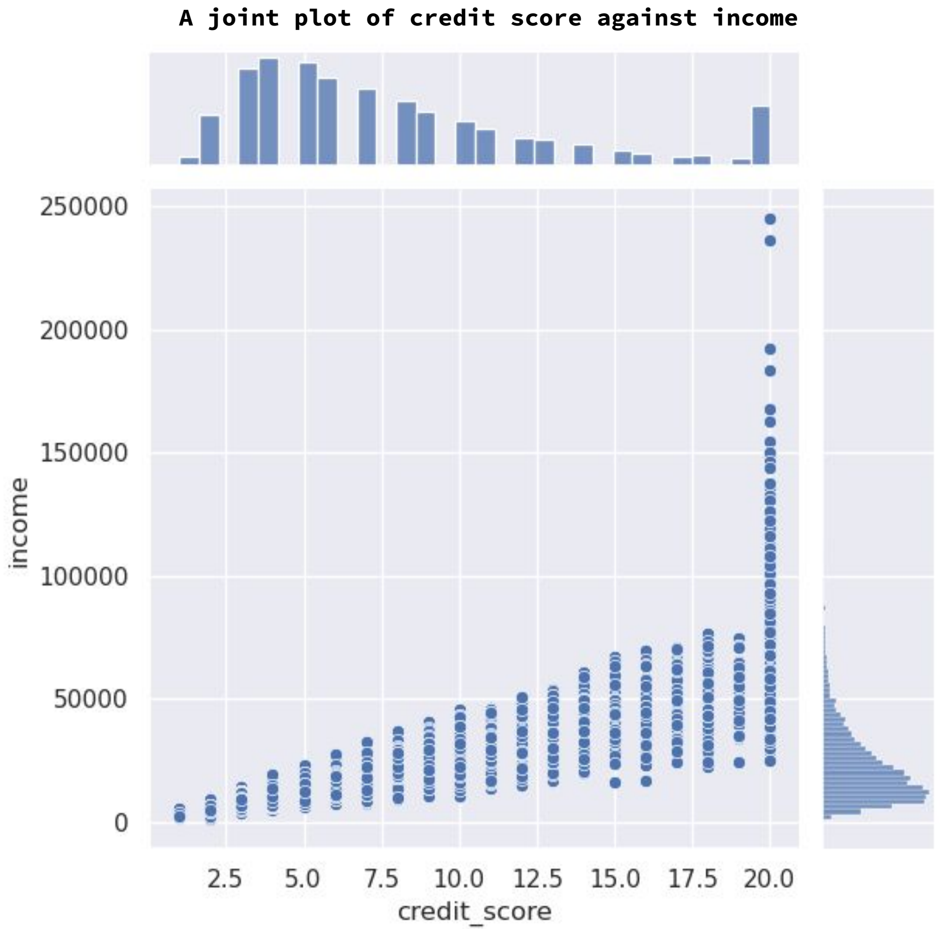
The credit score increases with increasing income, this further reinforce that those who earns more is more likely to pay back on time.

Data exploration and visualization



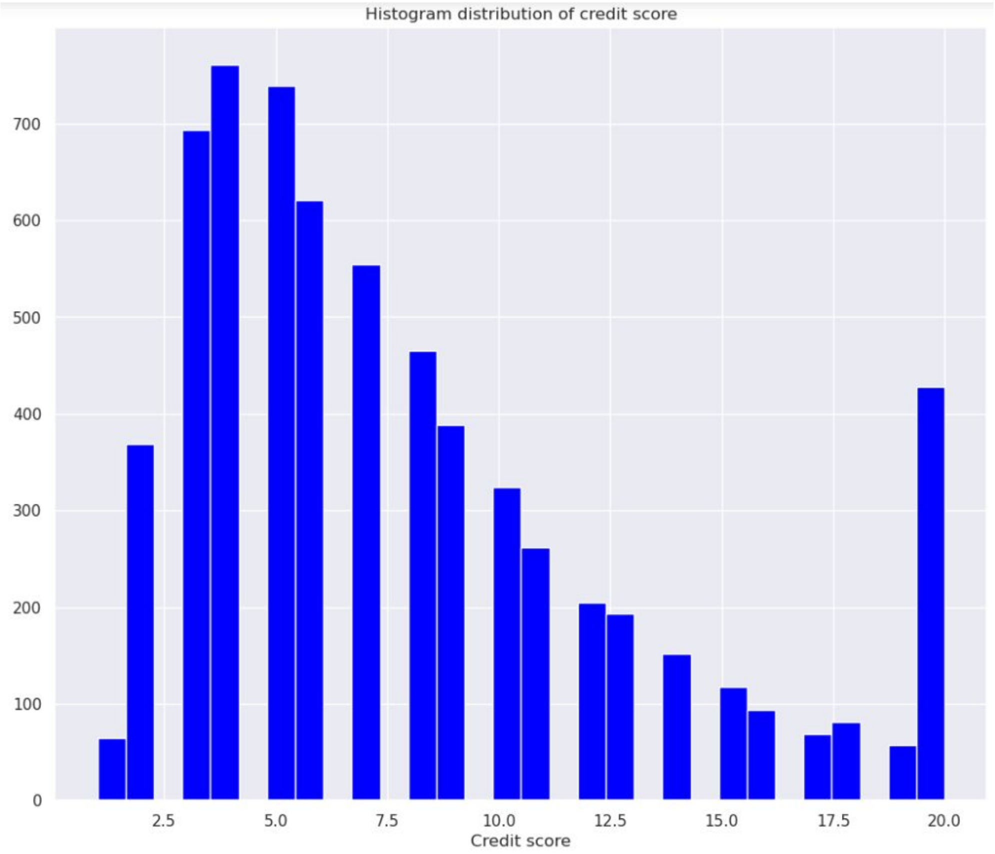
Another proof, that customers who earn less are most likely to churn. As its shown on the chart that clients who earns more fall into the category of those who never churned.

Data exploration and visualization



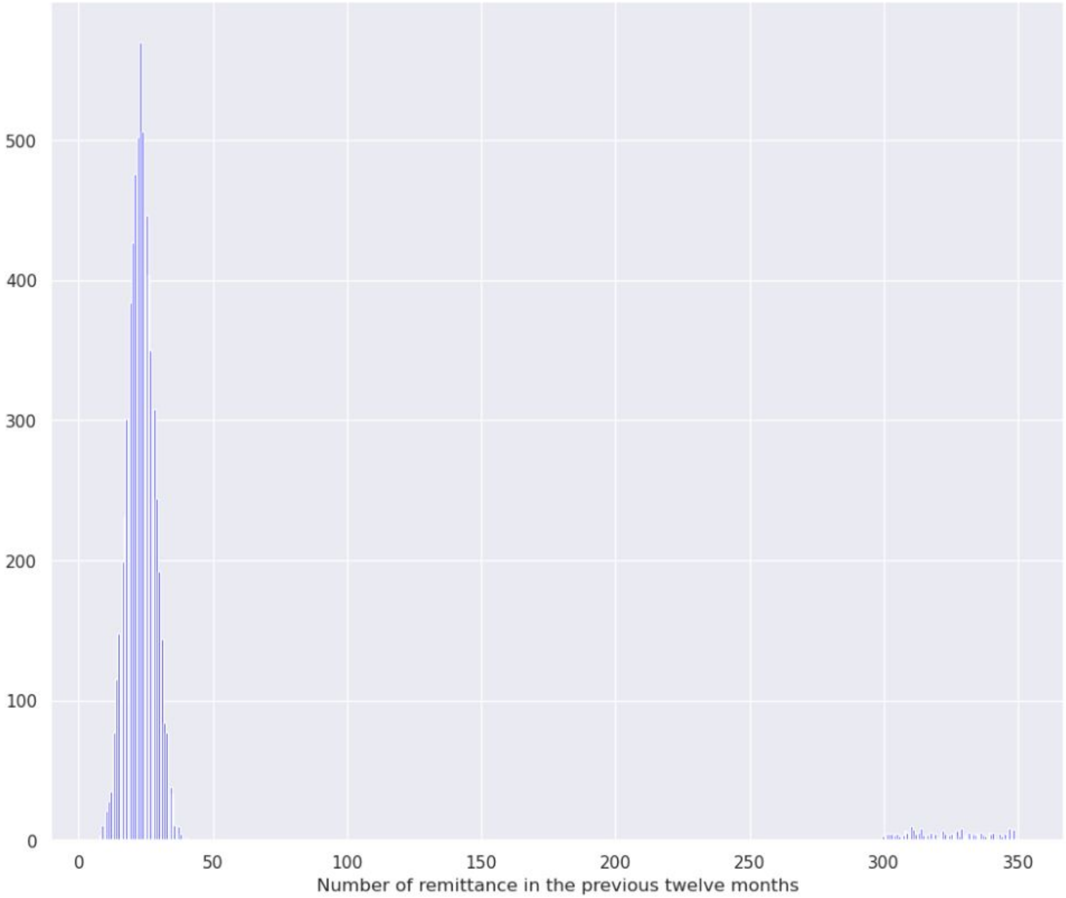
Histogram distribution of credit score against income shows that, most of the credit scores(2.5 to 18) fall into the income range of between 0 and 50,000.

Data exploration and visualization



The high density of credit score is recorded between 0 and 10.

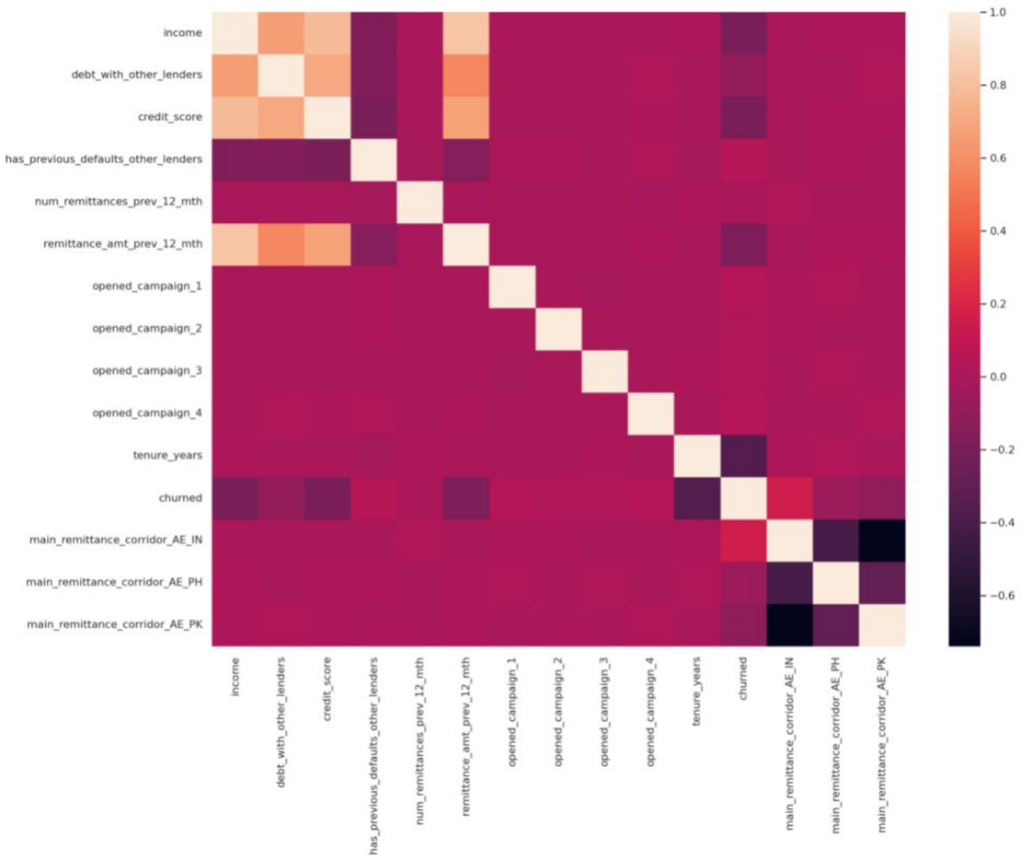
Data exploration and visualization



The high intensity of number of remittance between the last 12 months falls 10 and 30.

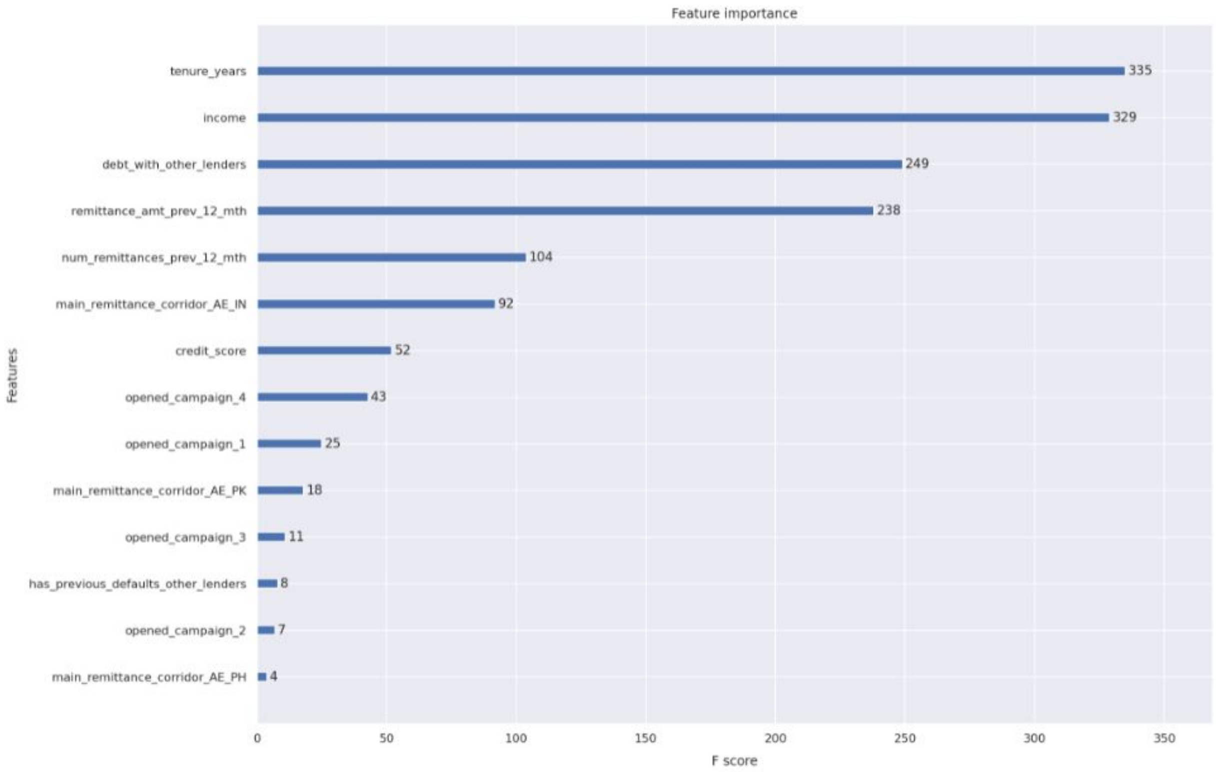
Model Experimentation

Feature Importance and Selection



All features have high correlation to the target feature, ‘churned’, except the *‘tenure year’.*

Feature Importance and Selection



The feature importance shows the relevance of each feature as related to the target feature, ‘churned’.

Feature Importance and Selection

```
print('Score list:', select_feature.scores_)
print('Feature list:', X_train.columns)
```

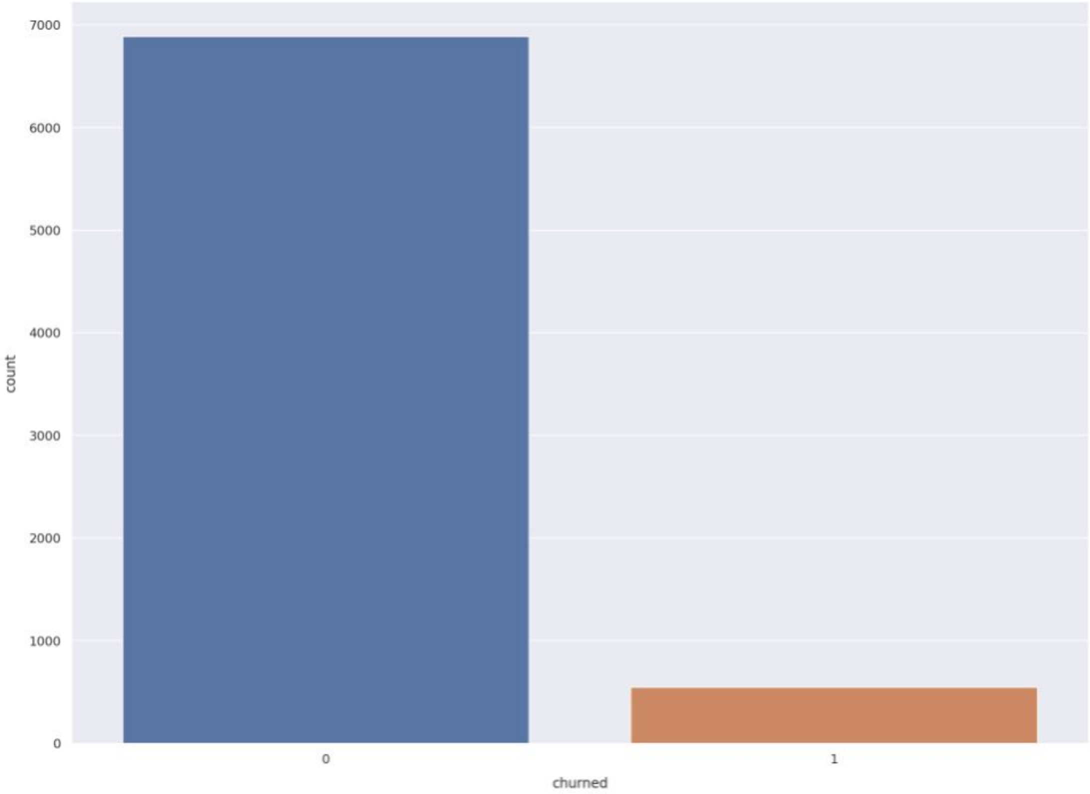
```
Score list: [3.04338274e+06 1.51691423e+06 6.08960654e+02 1.46454645e+01
 5.04335329e+01 1.56701787e+06 1.30601793e+01 2.48765319e+00
 2.00620720e+00 7.77457964e+00 3.41789047e+02 6.17396710e+01
 2.49801056e+01 3.92076226e+01]
Feature list: Index(['income', 'debt_with_other_lenders', 'credit_score',
       'has_previous_defaults_other_lenders', 'num_remittances_prev_12_mth',
       'remittance_amt_prev_12_mth', 'opened_campaign_1', 'opened_campaign_2',
       'opened_campaign_3', 'opened_campaign_4', 'tenure_years',
       'main_remittance_corridor_AE_IN', 'main_remittance_corridor_AE_PH',
       'main_remittance_corridor_AE_PK'],
      dtype='object')
```

```
print('Chosen best feature by rfe:', X_train.columns[rfe.support_])
```

```
Chosen best feature by rfe: Index(['income', 'debt_with_other_lenders', 'credit_score',
       'num_remittances_prev_12_mth', 'remittance_amt_prev_12_mth',
       'opened_campaign_1', 'opened_campaign_4', 'tenure_years',
       'main_remittance_corridor_AE_IN', 'main_remittance_corridor_AE_PK'],
      dtype='object')
```

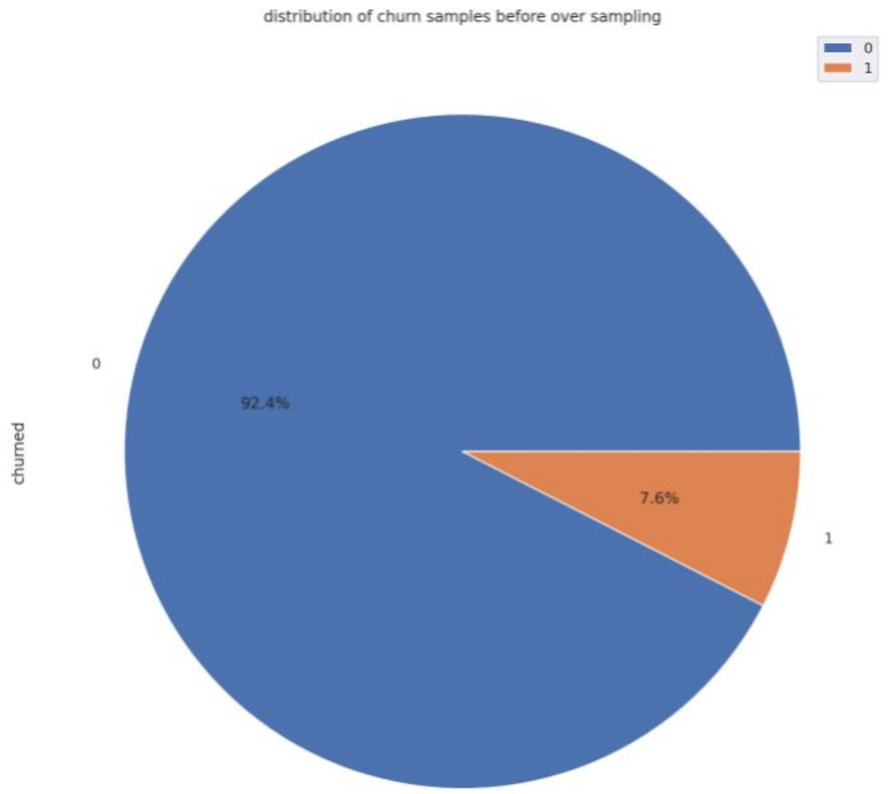
Further feature selection technique such as: Chi square, and RFE; proves that features such as income, debt with other lenders, and credit score are the most important features.

Model Experimentation



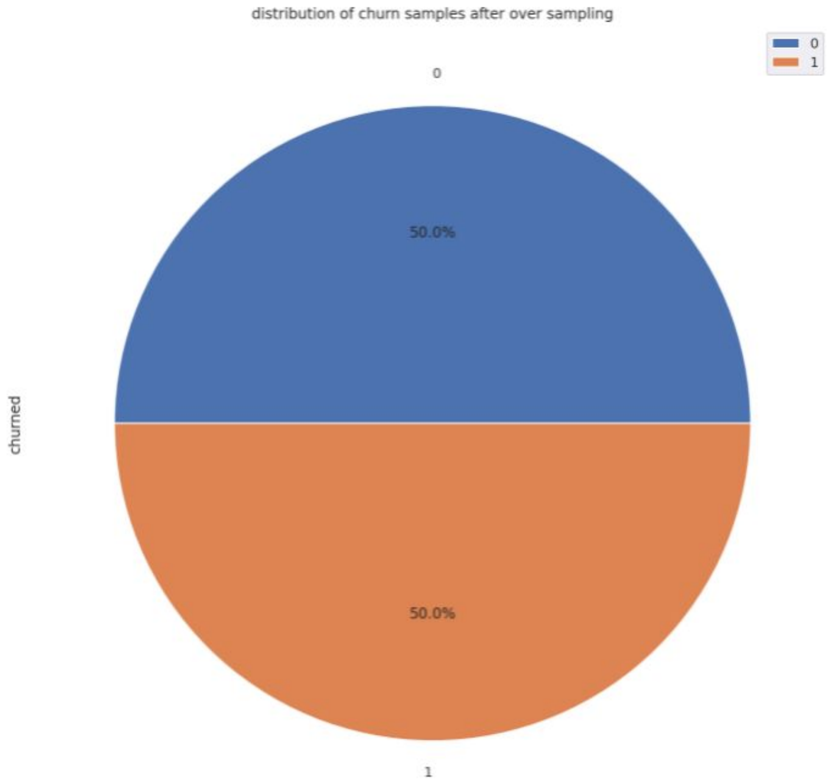
With around 7000 clients who never churned, there is less than 500 who have churned.

Model Experimentation



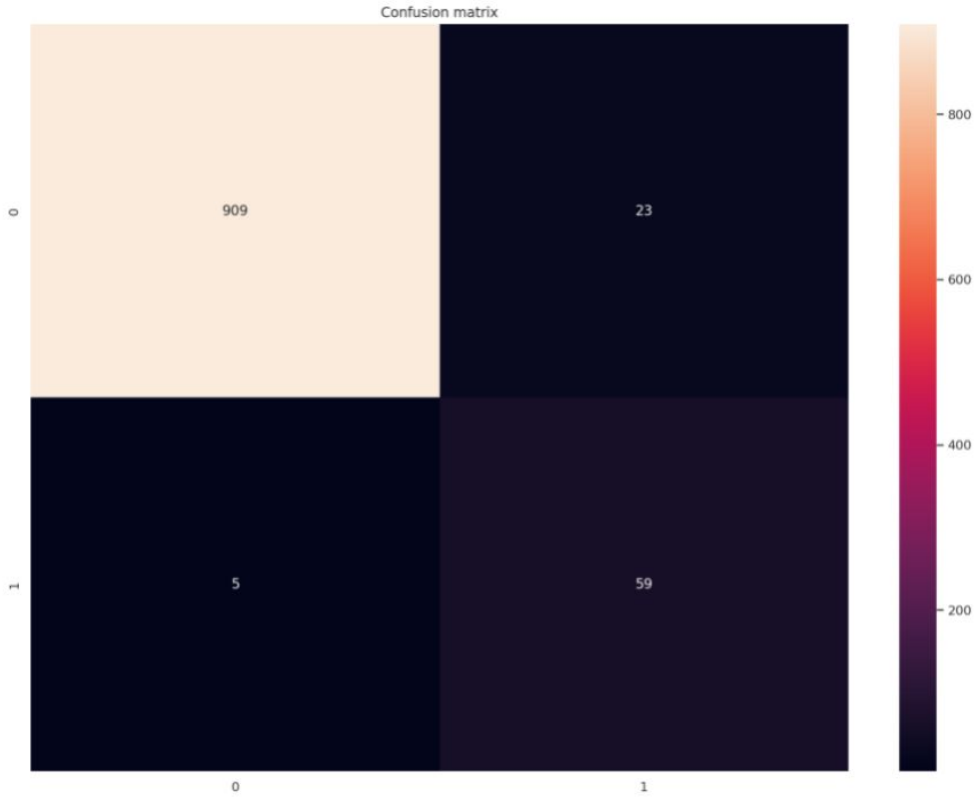
Around 8% have churned. This will force the algorithm to be biased, therefore Synthetic method for over sampling needs to be employed.

Model Experimentation



After the application of oversampling, the under represented samples(churned samples) are now represented equally as the over represented samples.

Model Experimentation- Logistic Regression



The predicted samples and actual samples that won't churn were over 900 samples while 59 samples of predicted and actual samples were to churn.

Model Experimentation- Logistic Regression

```python
# Compute metrics
print(classification_report(y_test, y_pred_lr))
```

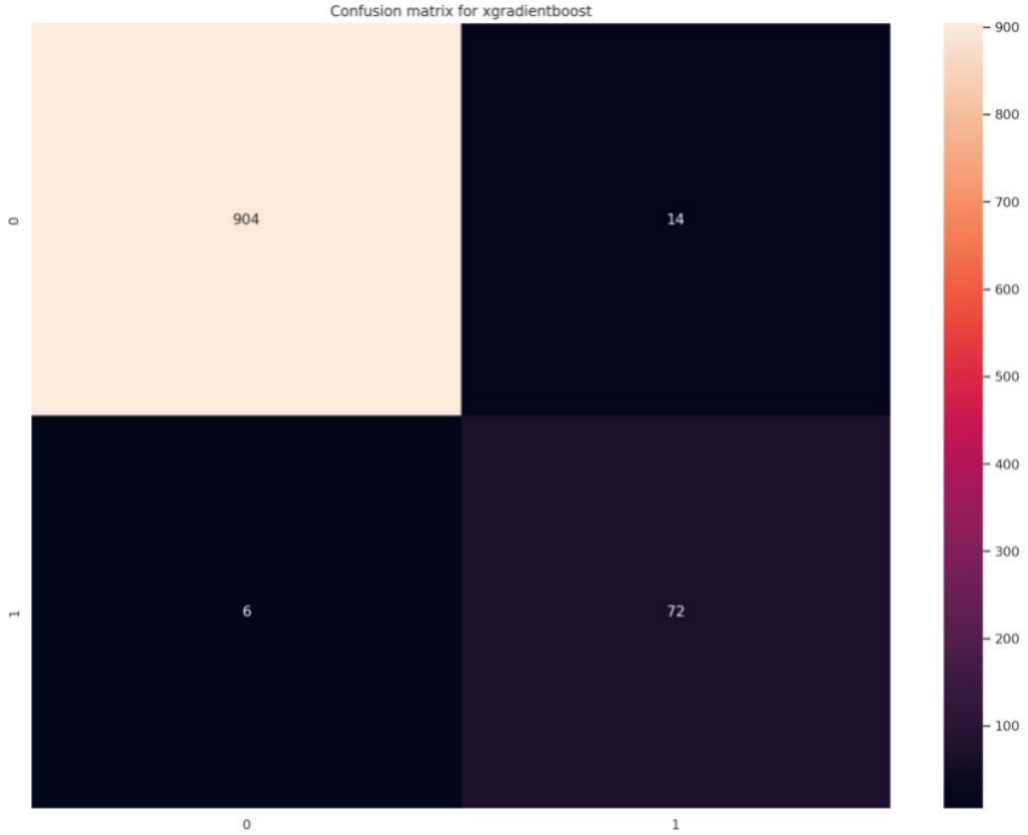
```
              precision    recall  f1-score   support

           0       0.99      0.98      0.98       932
           1       0.72      0.92      0.81        64

    accuracy                           0.97       996
   macro avg       0.86      0.95      0.90       996
weighted avg       0.98      0.97      0.97       996
```

With 81% accuracy from f1-score on test data for the customers that churned, and overall accuracy of 97%, signifies that logistic regression performs extremely well in this use case.

Model Experimentation- Xgradient boost Classifier



The predicted samples and actual samples that won't churn were over 900 samples while 72 samples of predicted and actual samples were to churn.

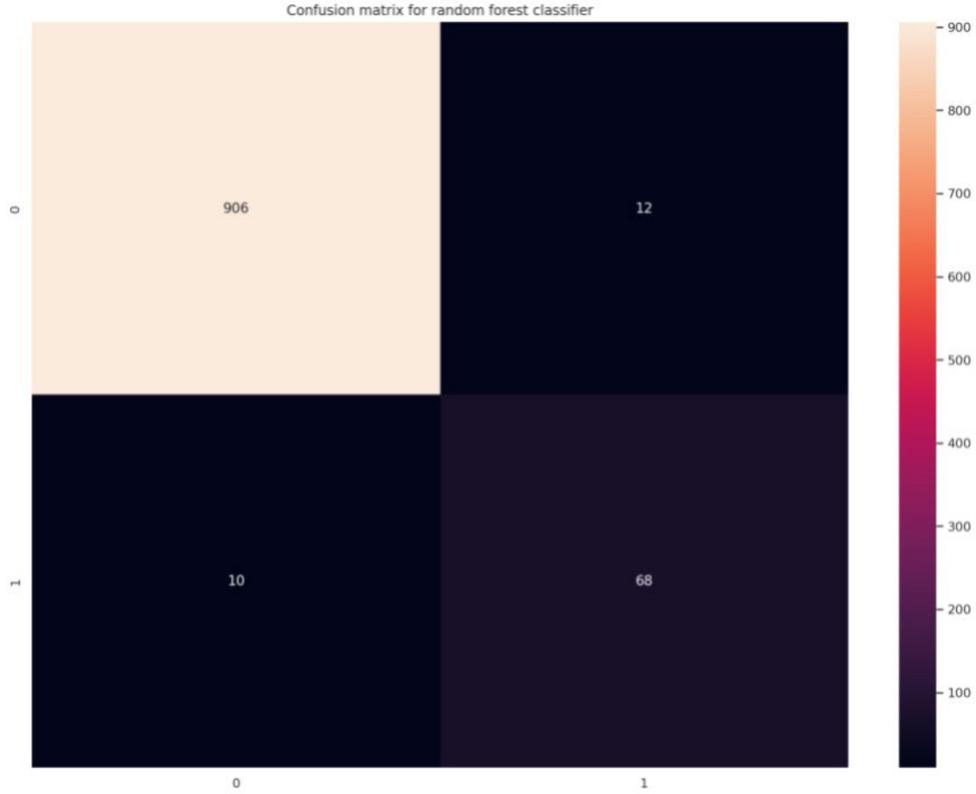
Model Experimentation- XGradient boost Classifier

```
: # Compute metrics
print(classification_report(y_test, y_pred_xgb))
```

	precision	recall	f1-score	support
0	0.99	0.98	0.99	918
1	0.84	0.92	0.88	78
accuracy			0.98	996
macro avg	0.92	0.95	0.93	996
weighted avg	0.98	0.98	0.98	996

This is not the case for extreme gradient boost were the the f1-score is 88% on the test data for the customers that churned, and overall accuracy of 98%. This has perform better than logistic regression.

Model Experimentation- Random Forest Classifier



The predicted samples and actual samples that won't churn were over 900 samples while 68 samples of predicted and actual samples were to churn.

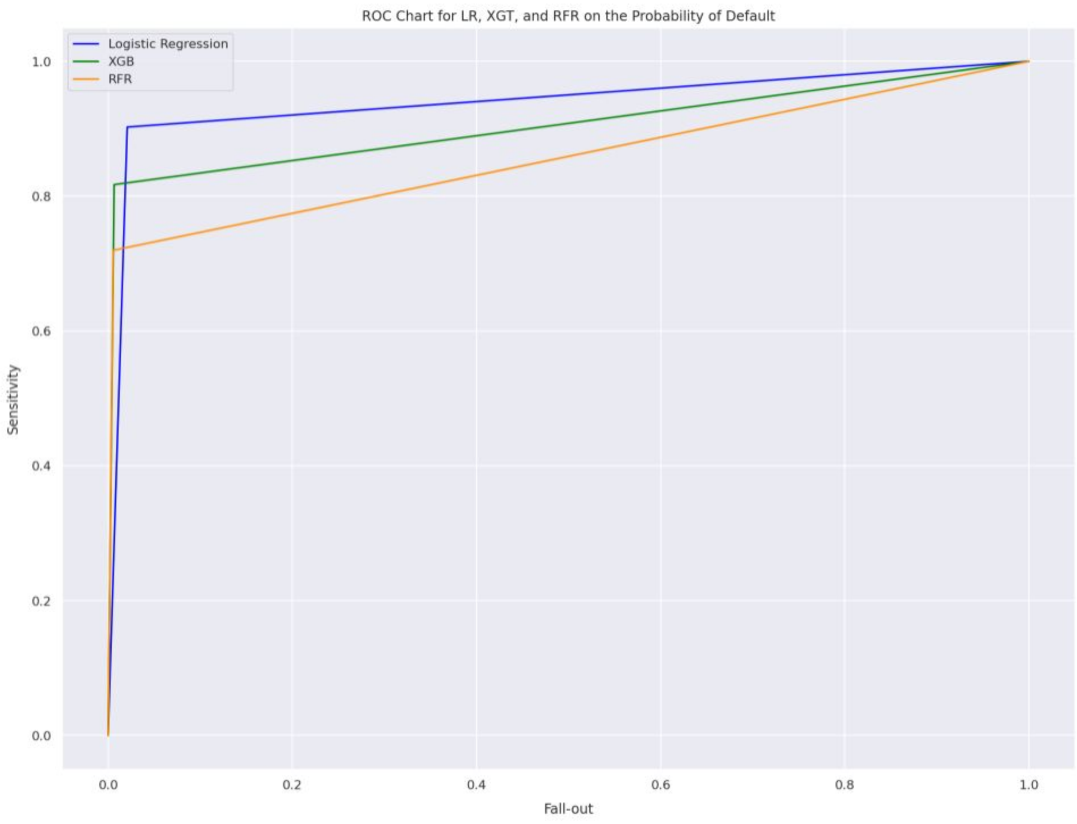
Model Experimentation- Random Forest Classifier

```
# Compute metrics
print(classification_report(y_test, y_pred_rfr))
```

	precision	recall	f1-score	support
0	0.99	0.99	0.99	918
1	0.85	0.87	0.86	78
accuracy			0.98	996
macro avg	0.92	0.93	0.92	996
weighted avg	0.98	0.98	0.98	996

This is not the case for extreme gradient boost were the the f1-score is 86% on the test data for the customers that churned, and overall accuracy of 98%.

Model Experimentation- Trained Model comparism



The closer the line graph is to the diagonal, the better the predicted probability. For this case, Logistic regression has the best sensitivity compared to others trained models.

Model Validation

Model Experimentation- Logistic Regression

```
# Compute metrics
print(classification_report(y_val, y_pred_lr_val))
```

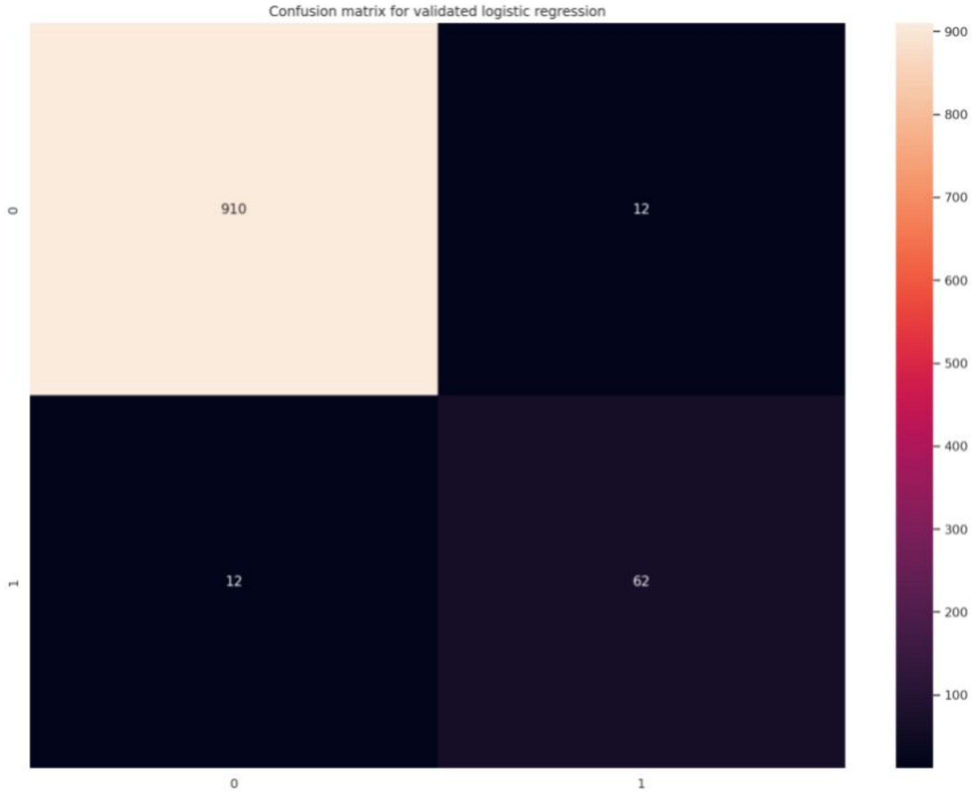
```
              precision    recall  f1-score   support

           0       0.99      0.99      0.99       922
           1       0.84      0.84      0.84        74

    accuracy                           0.98       996
   macro avg       0.91      0.91      0.91       996
weighted avg       0.98      0.98      0.98       996
```

The validated model has 84% accuracy from f1-score on validation data for the customers that churned, and overall accuracy of 98% for logistic regression.

Model Experimentation- Logistic Regression



The confusion matrix after validation shows that, the predicted samples and actual samples that won’t churn were over 900 samples while 62 samples of predicted and actual samples will churn.

Model Experimentation- XGradient Boost Classifier

```
# Compute metrics
print(classification_report(y_val, y_pred_xgb_val))
```

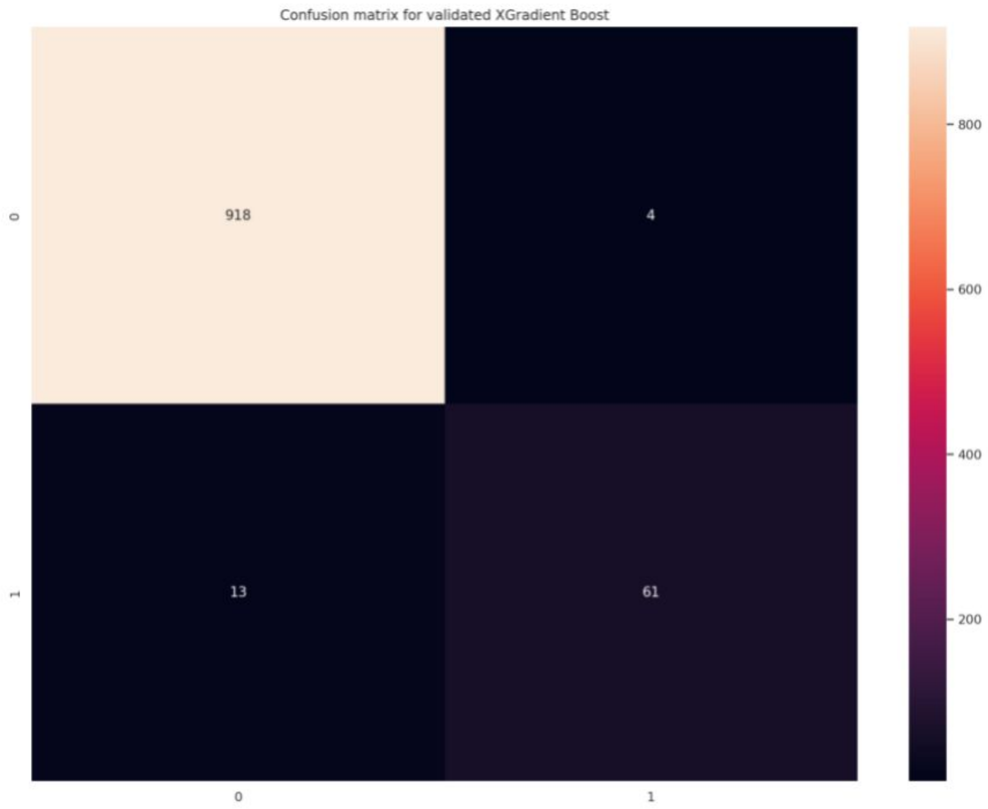
```
              precision    recall  f1-score   support

           0       0.99      1.00      0.99       922
           1       0.94      0.82      0.88        74

    accuracy                           0.98       996
   macro avg       0.96      0.91      0.93       996
weighted avg       0.98      0.98      0.98       996
```

The validated model has 88% accuracy from f1-score on validation data for the customers that churned, and overall accuracy of 98% for extended gradient boost.
Gradient boost has a close good performance to logistics regression.

Model Experimentation- XGradient Boost Classifier



For extreme gradient boost, the confusion matrix after validation shows that, the predicted samples and actual samples that won’t churn is 918 samples while 61 samples of predicted and actual samples represents those that churned.

Model Experimentation- Random Forest Classifier

```
# Compute metrics
print(classification_report(y_val, y_pred_rfr_val))
```

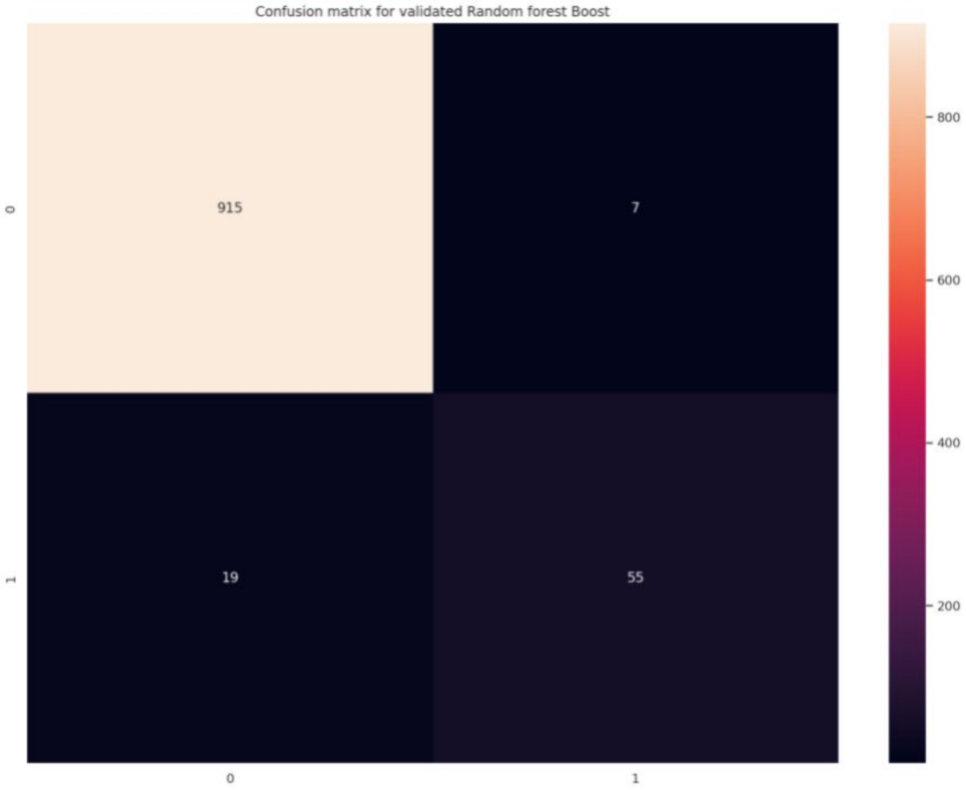
```
              precision    recall  f1-score   support

           0       0.98      0.99      0.99       922
           1       0.89      0.74      0.81        74

    accuracy                           0.97       996
   macro avg       0.93      0.87      0.90       996
weighted avg       0.97      0.97      0.97       996
```

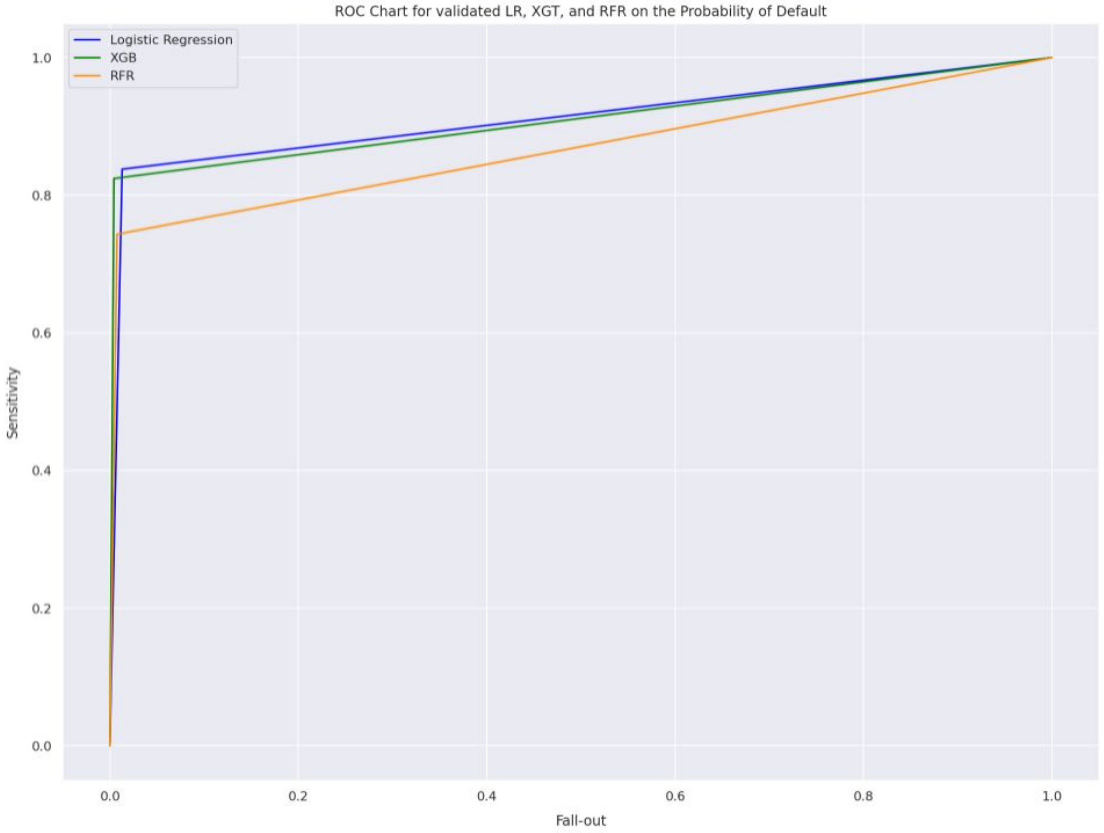
Random forest possess the good performance metrics but has the least f1-score and accuracy amongst the three models.
81% accuracy from f1-score on validation data for the customers that churned, and overall accuracy of 97% for logistic regression.

Model Experimentation- Random Forest Classifier



For random forest classifier, after validation the confusion matrix shows that, the predicted samples and actual samples that won’t churn is 915 samples but 55 samples of predicted and actual samples represents those that churned.

Model Experimentation- Validated Model comparism



Validation seems to prove otherwise, due to the closeness between logistic regression and extended gradient boost.
In addition to this, Logistic regression still performs the best compared to others trained models.

CONCLUSION

All three models have extremely good prediction metrics. However, during model testing and training, Logistic regression had the best performance.

The validated models present us with two best models on the validation dataset, these are logistic regression and extended gradient boost. Its then **recommended** that the machine learning engineer should continue with **logistic regression** for this business use case.