EDA process:

Getting column statistics

1. We started off with 31 columns of data. It contained one userID column containing userIDs and one churn column containing whether the customer churned or not as response variable.
2. We are going to perform analysis for 27000 users.

Data cleanup:

1. Below 3 columns were there where there was NaN values:
   age 4
   credit_score 8031
   rewards_earned 3227

2. Given our length of dataset its best to do the following:
   i) Drop the rows where age is nan. We have way more data than processing 4 records by imputer for the NaN age.
   ii) Drop the columns credit_score and rewards_earned as NaN values are too high.
   It means our client is not so much serious about these fields to be considered as metrics in our models
   Hence we are dropping these columns off.

Visualizing data

We visualized our data in the following fashion

a. **Bar charts:** Bar charts were made to get to know which were the categorical columns. We found out that there are 17 columns in our dataset with categorical variables.
b. **Pie charts:** We continued our EDA process and plotted Pie charts for categorical columns to see the distribution of categorical data and find the percentage distribution.
   Upon visualizing the pie charts we can see that there is a column waiting_4_loan which has a very odd distribution. 99.9% are with 0 and 0.1% with 1. We don't see a balanced distribution for both churn = 0 and churn = 1. Thus this may cause a bias in the model. thus we go to Exploring Uneven Features of 5 columns.
c. **Exploring uneven features:** here we are trying to find in dataset for the records where waiting_4_loan = 1 what is the distribution of churn variable we see the output as follows:
   0 27 for churn value = 0 there are 27 records with waiting_4_loan= 1
   1 8 for churn value = 1 there are 8 records with waiting_4_loan= 1
   This is done to check whether there is any fishy value like all are 1 or 0.
   The fishy records if fed into ML model will create a bias. Here there will be little or no bias. So we don't need to exclude them.
d. **Correlation with Response Variable:** This was done for the all the non – categorical columns. This is done to find out whether is there any variable for which there is no correlation with the response variable? If there is any variable with which there is no correlation there is no point of feeding them in our ML model.
e. **Correlation Matrix:** In the correlation matrix we can see there is a strong negative correlation between android user and ios user. This totally makes sense.
   In general if you are android user its least likely you buy ios.

The correlation is not 1 because there can be some people who bought both IOS and android. There are some people who use a different OS altogether
Also in the dataset if we observe carefully we can see that app_web_user is 1 only when both app_downloaded and web_user are both 1. Thus app_web_user is a f(app_downloaded, web_user). Thus we are removing it. In all we are removing two columns with highly correlated data with the other one.
This prevents highly correlated variables to enter into ML algorithms. ML algorithms work best with independent variables.

Model Building process

One Hot encoding

Using pandas we introduced new columns to our dataframe as part of OHE for housing, zodiac_sign, payment_type. These three columns were needed to be OHEd.

Train Test split

The splitting was done with test_size = 20% of the total data.

Balancing training set:

Before balancing:
No. of records with churn = 0 was 12656
No. of records with churn = 1 was 8940
This wouldn't have any adverse effect on the model building process in this case. But in other scenarios where the distribution if more uneven there would have been some problem.
By balancing we truncated the no. of records at random where churn equal 0 and brought it equal to the no. of records with churn = 1.
After balancing churn column distribution:
1 8940
0 8940

Standard Scaling

We performed standard scaling to normalize each input variable.

Model testing and trials

We performed 10 fold cross validations using different models and found out the below result.

Model	Average accuracy by K fold validation	Accuracy Standard deviation by K fold validation	Accuracy on our test set	Comments
LogisticRegression	0.64	0.0096	accuracy_score: 0.62 precision_score: 0.53 recall_score: 0.74 f1_score: 0.61	
KNeighborsClassifier	0.618	0.0087	accuracy_score: 0.57 precision_score: 0.49 recall_score: 0.59 f1_score: 0.54	
SVC linear	0.6254	0.0073	accuracy_score: 0.58 precision_score: 0.50 recall_score: 0.83 f1_score: 0.62	Very slow in fitting and doing validation
SVC Gaussian kernel RBF	0.67	0.01	accuracy_score: 0.63 precision_score: 0.53 recall_score: 0.74 f1_score: 0.62	Very slow in fitting and doing validation also there was a high variance
GaussianNB	0.6	0.006	accuracy_score: 0.60 precision_score: 0.51 recall_score: 0.59 f1_score: 0.55	
DecisionTreeClassifier	0.73	0.006	accuracy_score: 0.62 precision_score: 0.53 recall_score: 0.66 f1_score: 0.59	Very high accuracy on training set K fold validation. Remarkably lower accuracy on test set. Case of overfitting.

RandomForestClassifier	0.78	0.009	accuracy_score: 0.68 precision_score: 0.59 recall_score: 0.75 f1_score: 0.66	Very high accuracy on training set K fold validation. Remarkably lower accuracy on test set. Case of overfitting. We can see higher precision score and highest recall score in this classifier.

Using Gridsearch on Random forest classifier

Parameters to tune:
'n_estimators':[100,150,200,250],
'criterion':['gini','entropy']

Thus result: best parameters {'criterion': 'entropy', 'n_estimators': 250}
Conclusion: Retrained the model using the above parameters didn't show much improvement over n_estimators = 100.

Feature Selection

We will apply Recursive feature selection process to lessen the number of predictors from 40 to 20.
Recursive Feature Elimination (RFE) as its title suggests recursively removes features, builds a model using the remaining attributes and **calculates model accuracy**. RFE is able to work out the combination of attributes that contribute to the prediction on the target variable (or class).
Thus it was able to find 20 most important columns.

Evaluation:

After reducing the features we performed scoring on model's performance.
Before removal:

```
Confusion matrix:
[[1990 1176]
 [ 548 1686]]
accuracy_score: 0.68
precision_score: 0.59
recall_score: 0.75
f1_score: 0.66
```

After Removal:

```
Confusion matrix:
[[2025 1141]
```

```
 [ 575 1659]]
accuracy_score: 0.68
precision_score: 0.59
recall_score: 0.74
f1_score: 0.66
```

Thus we can see that accuracy precision f1_score all remained same even after removal of 20 unwanted features. Thus this makes our model more robust.

Again we performed K fold validation to find accuracy mean and STD

Before feature elimination:
Accuracy average: 0.78 Accuracy STD: 0.009

After feature elimination:
Accuracy average: 0.773 Accuracy STD: 0.011