Aspirevision Tech Education Pvt. Ltd.

Project Documentation for:
Minimizing Churn Rate Through Analysis of Financial Habits

Submitted by:
Daibik DasGupta
Arpit Falcon
Shivam Singh
Lila Rova
Juli Sharma

Under Guidance Of:
Md. Farmanul Haque

1. Acknowledgement:

I am very happy to compile this project, along with my talented group members. However, it would not have been possible without aid from multiple individuals.
I am highly indebted to **AspireVision Tech** for training me and providing me with all the necessary knowledge.
I would like to express my gratitude to **Md. Farmanul Haque**, whose advice and guidance has proven invaluable in bringing about the end product. It is through his constant encouragement that I have been able to see this project to its completion.
There was also a lot of support throughout the course of the project from **Mr. Brijesh Kumar** who oversaw the entire process and extended his aid whenever required.
I also have immense appreciation for my group members for their constant cooperation and help.

2. Abstract:

The objective of this project is to utilize the data of financial habits of a group of people using a certain online service related to a financial firm. The online service will have a subscription based offer and through this project, we will attempt to efficiently predict whether or not the user will be *churning (canceling subscription)* or not. This will help the company engage with the customer who is likely to churn. Identifying the *behavioral pattern* of the customer acts as a catalyst in identifying the reasons behind the disengagement of the customers. There are approximately 40+ attributes being considered while making predictions about churning, such as- number of deposits made, withdrawals made, purchases made, and whether user owns a house or rents one, and so on. This project seeks to utilize a variety of different Machine Learning algorithms to train using the provided data and make the predictions.
To conclude, the purpose of the project is to single out all the possible users who may choose to cancel their subscriptions.

3. Contents

4.Introduction

4.1 Problem Statement:
E-Companies are usually heavily reliant on paid subscriptions as a primary source of income, however there is always the issue of a certain customer losing interest in the service and eventually churning. This is a very common problem in the industry and can be remedied possibly by re-engaging the user before they churn.

4.2 Project Description:
The project is the implementation of *Machine Learning classification algorithm(s)* on a set of data giving some financial and behavioral details of the users of some subscription based service. The objective is to efficiently and accurately predict which customers may churn (cancel subscription), so that the e-company offering the subscription may engage with these customers and prevent future loss.

4.3 Objectives:
The objectives of the project will be discussed in this particular section. They are as follows:
-Analyze the financial data and behaviour of a set of users.
-Find the frequency of the variants of data and represent them visually.
-Find correlation of each attribute given.
-Clean the data to ensure it can be fit to any classification model.
-Finding accuracy of a variety of classification models to find the best classifier.
-Optimizing the best classification model to obtain further accuracy.
-Displaying the possibility of churning on a new set of independent variables.

System Requirements:

Operating System:	Windows XP SP3 and above
Processor:	2 x 64-bit 2.8 GHz 8.00 GT/s CPUs
RAM Requirement:	32 GB
Editing Tools:	Spyder or Jupyter

5. Libraries and Models Used

5.1 Libraries Used:

The following is a list of libraries used throughout the entirety of coding along with their basic purpose:

pandas	For data retrieval, manipulation, storage and analysis.
numpy	Adds support for large, multidimensional arrays and matrices, along with a large collection of high-level mathematical functions.
matplotlib	Provides an object-oriented API for embedding plots into applications.
seaborn	Data visualization library based on matplotlib, used for statistical data.
time	Provides various time-related functions

5.2 Functions Used:

The following is a list of functions used throughout the entirety of coding along with their basic purpose and their parent library:

read_csv( ) - Read a comma-separated values (.csv) file into a data frame. See below for how the data frame is represented in our editor:

```
6 dataset = pd.read_csv('churn_data.csv')
```

Index	user	churn	age	housing	credit_score
0	55409	0	37	na	nan
1	23547	0	28	R	486
2	58313	0	35	R	561
3	8095	0	26	R	567
4	61353	1	27	na	nan
5	3120	1	32	R	567
6	41406	0	21	na	475
7	67679	0	24	na	nan
8	21269	0	28	R	548
9	25788	0	23	na	658
10	13840	0	32	R	nan

Data set obtained after reading a .csv file

to_csv( ) - Write object to a comma-separated values (.csv) file.

head( ) - Displays first 5 rows of data, unless a number is specified in parameters, in which case, that many number of rows from start is displayed.

columns - Displays all the column heads of a given data set.

describe( ) - is used to view some basic statistical details like percentile, mean, std etc. of a data frame.

isna( ) - Returns a Boolean value true if missing values are present, otherwise returns false.

any( ) - Accepts iterable (list, tuple, dictionary etc.) as an argument and return true if any of the element in iterable is true.

```
In [6]: dataset.isna().any()
Out[6]:
user                    False
churn                   False
age                     False
housing                 False
```

Output of any( ) function

sum( ) - Adds the elements of an iterable and returns the sum.

```
In [7]: dataset.isna().sum()
Out[7]:
user                    0
churn                   0
age                     0
housing                 0
```

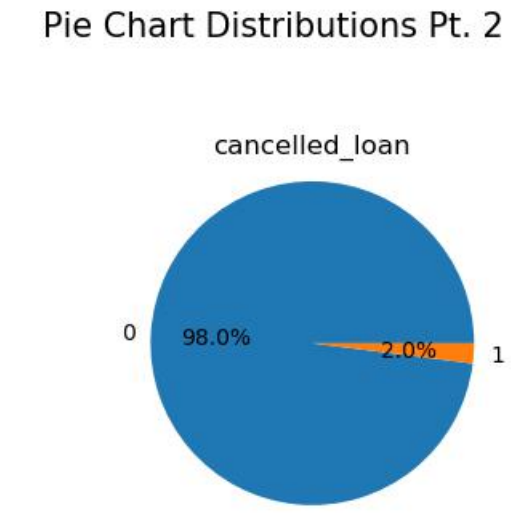
Output of sum( ) function

drop(columns = [ ]) - Deletes the column specified by column header name within the [ ] brackets.

figure ( ) - Create a new figure.

suptitle( ) - Add a centered title name to the figure.

```
plt.suptitle('Pie Chart Distributions Pt. 2', fontsize=15)
```



Title of pie chart obtained

subplot( ) - Add subplot to the current figure.

gca( ) - Stands for Get Column Access, iterates through all the values in a column.

get_yaxis( ) - Returns Y-Axis instance.

set_visible( ) - Set the artists' visibility.

set_title( ) - Sets a string value as the title of some data visualization instance.

iloc[ ] - Stands for index location. Used to select some rows of data from a column.

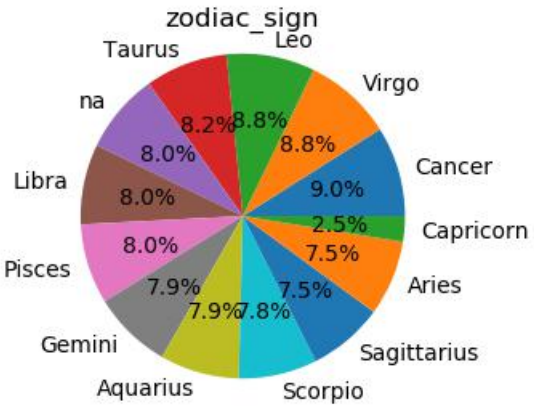
value_counts( ) - Returns object containing counts of unique values.

```
In [15]: dataset[dataset2.cancelled_loan == 1].churn.value_counts()
Out[15]:
0    194
1    187
Name: churn, dtype: int64
```

Output of value_counts( )

pie( ) - Produces a pie chart with the define specifications.

```
58    plt.pie(values, labels = index, autopct='%1.1f%%')
```



Data visualization through pie chart

tight_layout( ) - Automatically adjust subplot parameters to give specified padding.

corrwith( ) - Used to compute pairwise correlation between rows or columns of two data frame objects.

set( ) - used to convert any of the iterable to the distinct element and sorted sequence of iterable elements, commonly called Set.

corr( ) - used to find the pairwise correlation of all columns in the data frame.

Index	age	deposits	withdrawal
age	1	0.107722	0.0548934
deposits	0.107722	1	0.478407
withdrawal	0.0548934	0.478407	1
purchases_par…	-3.10162e-05	0.303484	0.201331

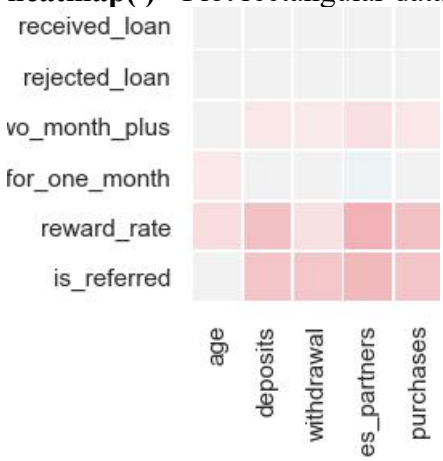
Snippet of correlation matrix

zeros_like( ) - Return an array of zeros with the same shape and type as a given array.

triu_indices_from( ) - Return the indices for the upper-triangle of two-dimensional array.

diverging_palette( ) - Make a diverging palette between two colors.

heatmap( ) - Plot rectangular data as a color-encoded matrix.



Snippet of heat map

nunique( ) - Return number of unique elements in the object.

reset_index( ) - Resets index of a Data Frame.

test_train_split( ) - Split arrays or matrices into random train and test subsets.

X_test - DataFrame

Index	age	deposits	withdrawal	urcha
3695	48	0	0	0
3913	29	0	0	140
13358	32	56	0	122
7118	29	0	0	0
12305	54	0	0	0
10592	24	0	0	25

X_train - DataFrame

Index	age	deposits	withdrawal	urcha
7624	36	1	0	17
12462	20	0	0	10
6666	23	12	1	56
538	47	2	0	24
3319	43	0	0	0
11435	37	0	0	0

Test data and train data

StandardScaler( ) - Standardize features by removing the mean and scaling to unit variance.

Index	age	deposits	withdrawal
3695	1.63678	-0.411784	-0.315782
3913	-0.339155	-0.411784	-0.315782
13358	-0.0271644	5.07732	-0.315782
7118	-0.339155	-0.411784	-0.315782
12305	2.26076	-0.411784	-0.315782
10592	-0.859138	-0.411784	-0.315782

Snippet of changed values after scaling

DataFrame( ) - Creates a two-dimensional size-mutable, potentially heterogeneous tabular data structure with labeled axes (rows and columns).

fit( ) - Sets given data according to a specified model.

predict( ) - Given a trained model, predict the label of a new set of data.

accuracy_score( ) - In multilabel classification, this function computes subset accuracy.

```
In [7]: acc = accuracy_score(y_test, y_pred)
   ...: acc
Out[7]: 0.659372528341682
```

precision_score( ) - Compute the precision. The precision is the ratio tp / (tp + fp) where tp is the number of true positives and fp the number of false positives.

```
In [9]: prec = precision_score(y_test, y_pred)
   ...: prec
Out[9]: 0.6026986506746627
```

recall_score( ) - Compute the recall. The recall is the ratio tp / (tp + fn) where tp is the number of true positives and fn the number of false negatives.

```
In [10]: rec = recall_score(y_test, y_pred)
    ...: rec
Out[10]: 0.5134099616858238
```

f1_score( ) - Compute the F1 score, also known as balanced F-score or F-measure. The F1 score can be interpreted as a weighted average of the precision and recall, where an F1 score reaches its best value at 1 and worst score at 0. The relative contribution of precision and recall to the F1 score are equal. The formula for the F1 score is: F1 = 2 * (precision * recall) / (precision + recall)

```
In [11]: f1 = f1_score(y_test, y_pred)
    ...: f1
Out[11]: 0.5544827586206896
```

append( ) - Adds its argument as a single element to the end of a list.

cross_val_score( ) - Evaluate a score by cross-validation.

GridSearchCV( ) - Exhaustive search over specified parameter values for an estimator.

confusion_matrix( ) - Compute confusion matrix to evaluate the accuracy of a classification.

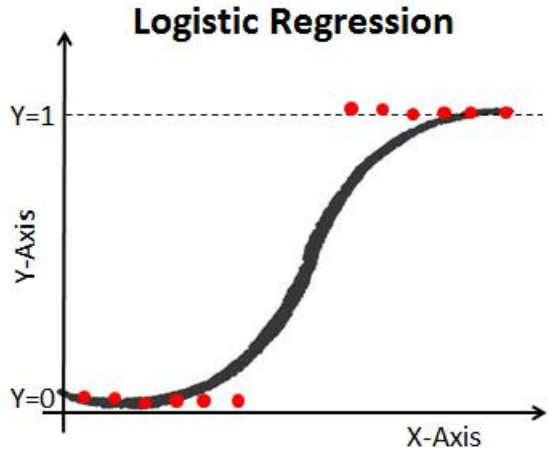
	0	1
0	1839	388
1	657	909

Confusion matrix obtained from data

time( ) - Returns the number of seconds passed since epoch.

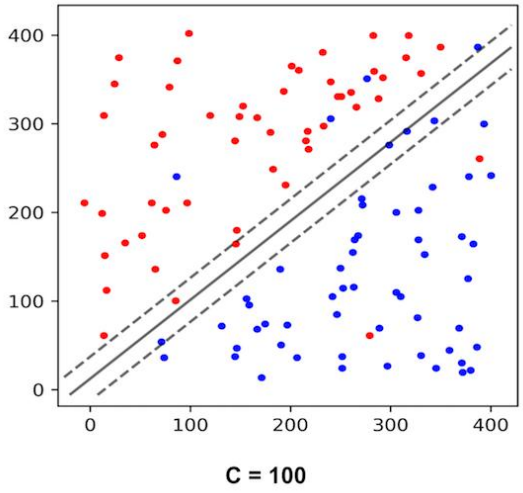
5.3 Models Used:
The following Machine Learning models have been used to predict the data:

Logistic Regression - It is a statistical method for analysing a data set in which there are one or more independent variables that determine an outcome. The outcome is measured with a dichotomous variable (in which there are only two possible outcomes). The goal of logistic regression is to find the best fitting model to describe the relationship between the dichotomous characteristic of interest (dependent variable = response or outcome variable) and a set of independent (predictor or explanatory) variables.
Equation: y = log[p / (1 - p)] where p is probability of success



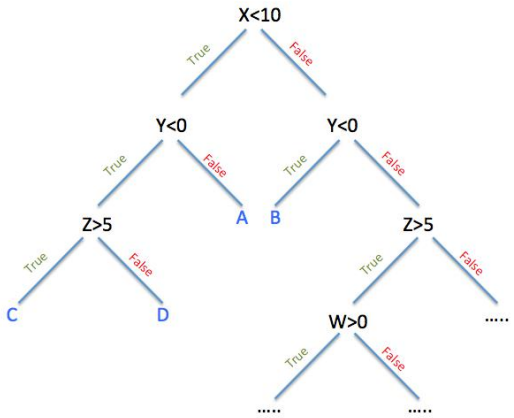
Logistic Regression Graph

SVM - It is a supervised machine learning algorithm capable of performing classification, regression and even outlying entity detection. The linear SVM classifier works by drawing a straight line known as a support vector to divide a plane representation into two hyper planes. All the data points that fall on one side of the vector will be labeled as one class and all the points that fall on the other side will be labeled as the second.



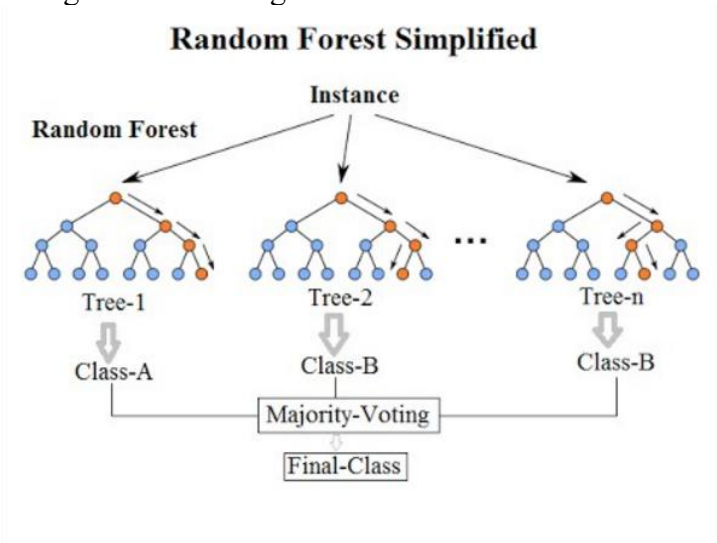
SVM Graph

Decision Trees - Decision tree builds classification or regression models in the form of a tree structure. It breaks down a data set into smaller and smaller subsets while at the same time an associated decision tree is incrementally developed. The final result is a tree with decision nodes and leaf nodes. A decision node has two or more branches and a leaf node represents a classification or decision. The topmost decision node in a tree which corresponds to the best predictor called root node. Decision trees can handle both categorical and numerical data.



Decision Tree Graphical Representation

Random Forest - Random forests or random decision forests are an ensemble learning method for classification, regression and other tasks, that operate by constructing a multitude of decision trees at training time and outputting the class that is the mode of the classes (classification) or mean prediction (regression) of the individual trees. Random decision forests correct for decision trees' habit of over fitting to their training set.



Random Forest Graphical Representation

6. Algorithm of Code

The following is the algorithm of the code and a list of all the operations performed on the data set to obtain the final prediction and best model:

6.1 Exploratory Data Analysis (EDA.py)

6.1.1 Importing Necessary Libraries and Data Set:

The aforementioned libraries of *numpy*, *matplotlib*, *pandas* are imported for mathematical operations on matrices, graphical representation of data and data retrieval and manipulation respectively. The pandas library is then used to import the given data set through *read_csv( )* function.

6.1.2 Exploring the Data:

The attributes present in the data set, the first 5 entities as well as a summary of some key values is presented to further get an overview of the data set.

```
In [2]: dataset.head()
Out[2]:
    user  churn   age  ... rewards_earned  reward_rate  is_referred
0  55409      0  37.0  ...            NaN         0.00            0
1  23547      0  28.0  ...           44.0         1.47            1
2  58313      0  35.0  ...           65.0         2.17            0
3   8095      0  26.0  ...           33.0         1.10            1
4  61353      1  27.0  ...            1.0         0.03            0

[5 rows x 31 columns]
```

```
In [3]: dataset.columns
Out[3]:
Index(['user', 'churn', 'age', 'housing', 'credit_score', 'deposits',
       'withdrawal', 'purchases_partners', 'purchases', 'cc_taken',
       'cc_recommended', 'cc_disliked', 'cc_liked', 'cc_application_begin',
       'app_downloaded', 'web_user', 'app_web_user', 'ios_user',
       'android_user', 'registered_phones', 'payment_type', 'waiting_4_loan',
       'cancelled_loan', 'received_loan', 'rejected_loan', 'zodiac_sign',
       'left_for_two_month_plus', 'left_for_one_month', 'rewards_earned',
       'reward_rate', 'is_referred'],
      dtype='object')
```

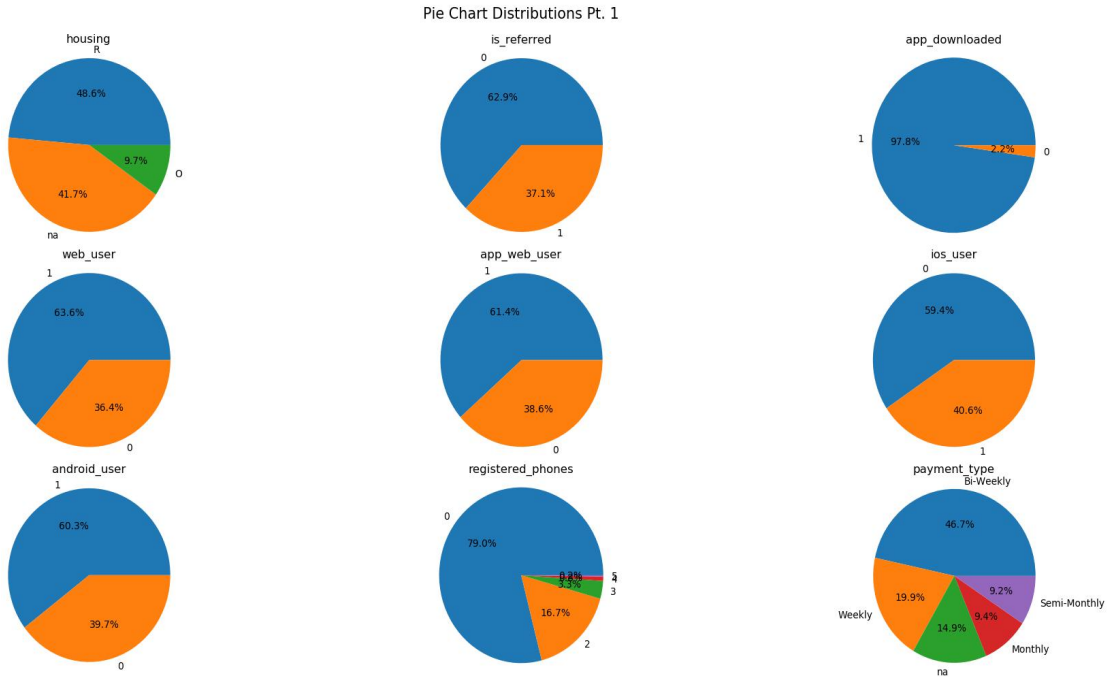
```
In [4]: dataset.describe()
Out[4]:
                user         churn  ...   reward_rate   is_referred
count  27000.000000  27000.000000  ...  27000.000000  27000.000000
mean   35422.702519      0.413852  ...      0.907684      0.318037
std    20321.006678      0.492532  ...      0.752016      0.465723
min        1.000000      0.000000  ...      0.000000      0.000000
25%    17810.500000      0.000000  ...      0.200000      0.000000
50%    35749.000000      0.000000  ...      0.780000      0.000000
75%    53244.250000      1.000000  ...      1.530000      1.000000
max    69658.000000      1.000000  ...      4.000000      1.000000

[8 rows x 28 columns]
```

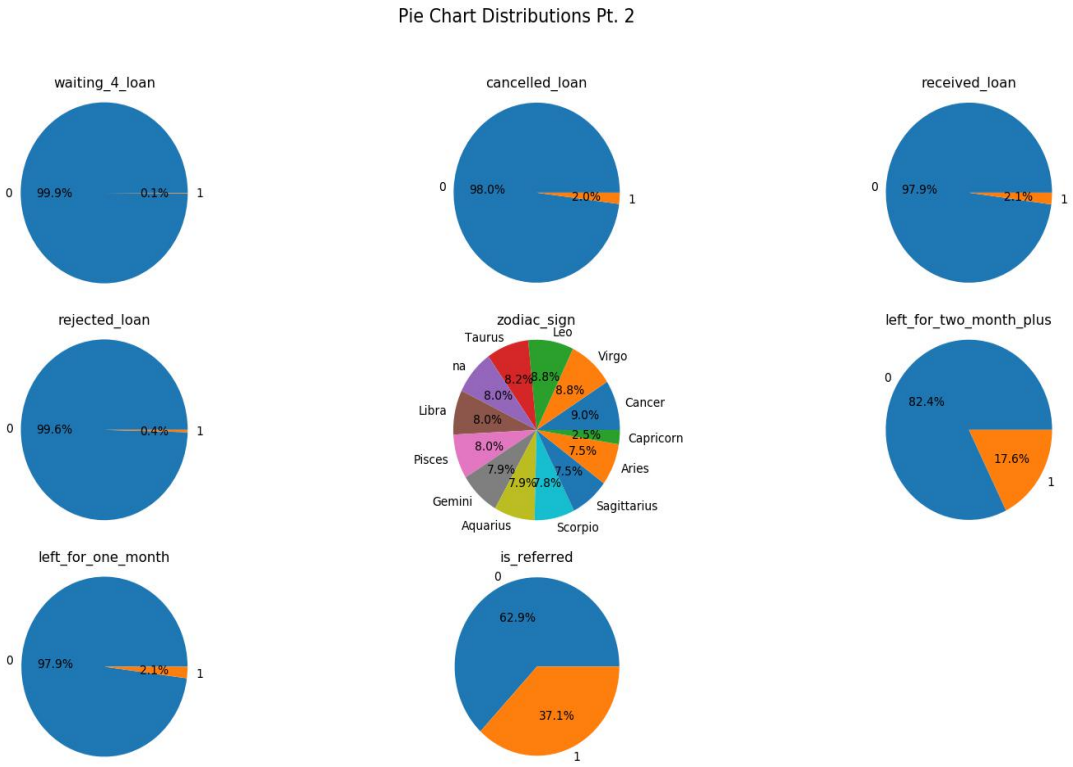
6.1.3 Early Data Cleaning:

We are working with the assumption that users with a credit score of less than 300 are not financially stable enough to continue the paid subscription, and those above are financially capable. The credit score is no more an affecting factor, as the financial behaviour exhibited in the remaining attributes will provide better information. Further, rewards earned is not a reliable enough feature as it has missing data for multiple entities (we used the *isna( )* function to show this) hence we can disregard that attribute as well. Thus we use the *drop( )* function to remove those columns from the data set.

6.1.4 Representing Frequency of Data through Pie Chart:

We produce a copy of the data set containing all the classifying attributes and we visually represent the frequency of occurrence of each classification type in our data set through percentages in a pie chart. There are 17 attributes including but not limited to *housing* situation, preferred *time slot of payment*, whether they are *waiting on a loan* or not, and so on. An individual pie chart is made for each of the attributes.
(See next page for Pie Chart Distributions)





Each pie chart represents a classifying feature and the distribution of each type of classification between all the entities. Each color represents a different classification criteria. For most of the pie charts, it is 0 for blue and 1 for orange, but in case of housing or zodiac_sign multiple classifying groups exist, where each color is so labeled.

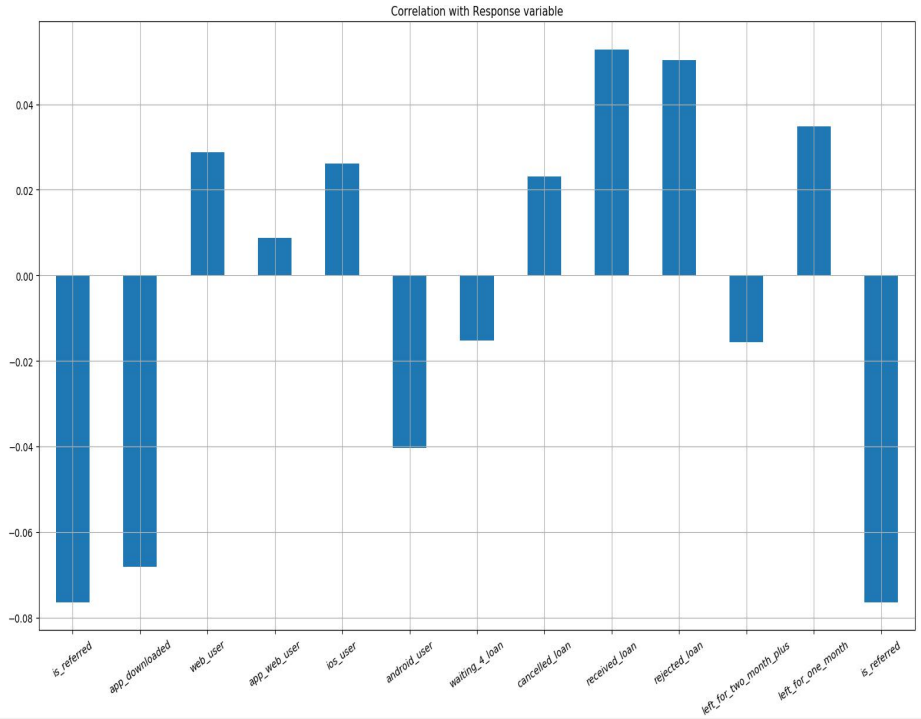
6.1.5 Exploring Uneven Features and Representing Correlation:

The number of users who churned and the number of users who didn't are counted based on the following 4 features individually - *whether user is waiting for a loan, whether their loan has been canceled, whether they have received a loan, whether their loan has been rejected, or whether they had churned for 1 month and returned*. The number of churning users and non-churning users for each of the above cases is then seen to see which of these features affects churning the most.

```
In [5]: dataset[dataset2.cancelled_loan == 1].churn.value_counts()
Out[5]:
0    194
1    187
Name: churn, dtype: int64

In [6]: dataset[dataset2.received_loan == 1].churn.value_counts()
Out[6]:
1    233
0    162
Name: churn, dtype: int64

In [7]: dataset[dataset2.rejected_loan == 1].churn.value_counts()
Out[7]:
1    64
0    17
Name: churn, dtype: int64

In [8]: dataset[dataset2.left_for_one_month == 1].churn.value_counts()
Out[8]:
1    207
0    184
Name: churn, dtype: int64
```

Frequency of churning and non-churning users based on each feature, where 0 represents frequency of non-churning users and 1 represents frequency of churning users.

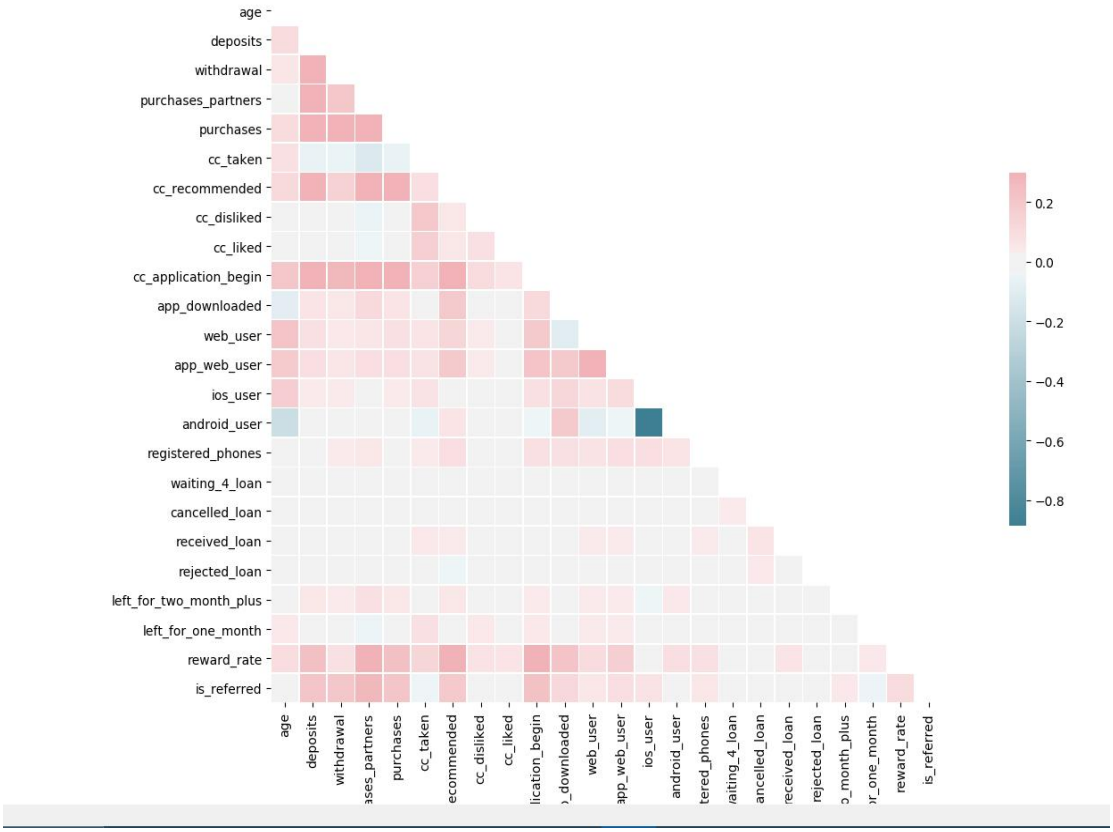
After this, the correlation of each attribute/feature (except housing situation, payment method, registered phone, and zodiac sign) is found and represented on a graph to show how strongly each feature affects the churning.



The X-Axis contains all the features and how strong their correlation is with whether or not user will churn. Y-Axis represents the degree of correlation. Bars going upwards represent positive correlation and bars going downwards represent negative correlation.

A heat map is a two-dimensional representation of information with the help of colors. Heat maps can help the user visualize simple or complex information.

Finally, a *heat map* is produced of all the features, so that the correlation of each feature to every other feature through the entire data set is represented in a single visual representation. We mask the upper half of the heat map as it is redundant due to being a mirror of the lower half. We do not include user ID and whether or not user has churned or not, as user ID is not an affecting feature and the the churn confirmation is the dependant variable.



Both the axes have all the attributes set along them. A darker shade of red on a cell represents a stronger positive correlation between the two attributes intersecting on that cell, and a darker shade of blue on a cell represents a stronger negative correlation.

6.1.6 Exporting Modified Data Set:

As we can see in the heat map, there are somewhat correlated fields, but they are not co-linear, that is, the features are not functions of each other, so they won't break the model, but they will not be able to aid much during prediction phase. With this, feature selection takes place and we export the data set with the required features out as "new_churn_data.csv" to fit to a model in our next program.

6.2 Model Fitting and Prediction (Model.py)

6.2.1 Data Importing, Cleaning Features and Data Splitting:

We import the modified data set through read_csv( ) function, and we store the user ID code in a separate array. Accordingly we remove the user ID feature (as we will use it later to identify the users and it has no purpose now), and the zodiac sign feature (as it does not impact the final result nearly as significantly as other features).

We obtain dummy variables for housing and payment type due to their being more than 2 classification groups, and we also delete the columns storing whether or not the features have missing or not applicable values to avoid the *dummy variable trap*.

We divide our data set on 8:2 ratio (80% train, 20% test) where our X (independent variable) constitutes of a feature matrix of all features (except churn status), and Y (dependent variable) will have churn status only in a vector.

Feature scaling is performed on the training and test sets of X, and we use Data Frame to ensure that our automatically imposed index column is stored safely so we can link the data.

6.2.2 Model Fitting and Optimization:

To find the best model for our data set, we will fit it to multiple different classification models and check the accuracy, precision, recall and f1 score for each of those models until we find the best model (using the appropriate score functions for each of them).
The following models were used in this project:

1. Logistic Regression
2. SVC (Linear Kernel)
3. SVC (RBF Kernel)
4. Decision Tree
5. Random Forest

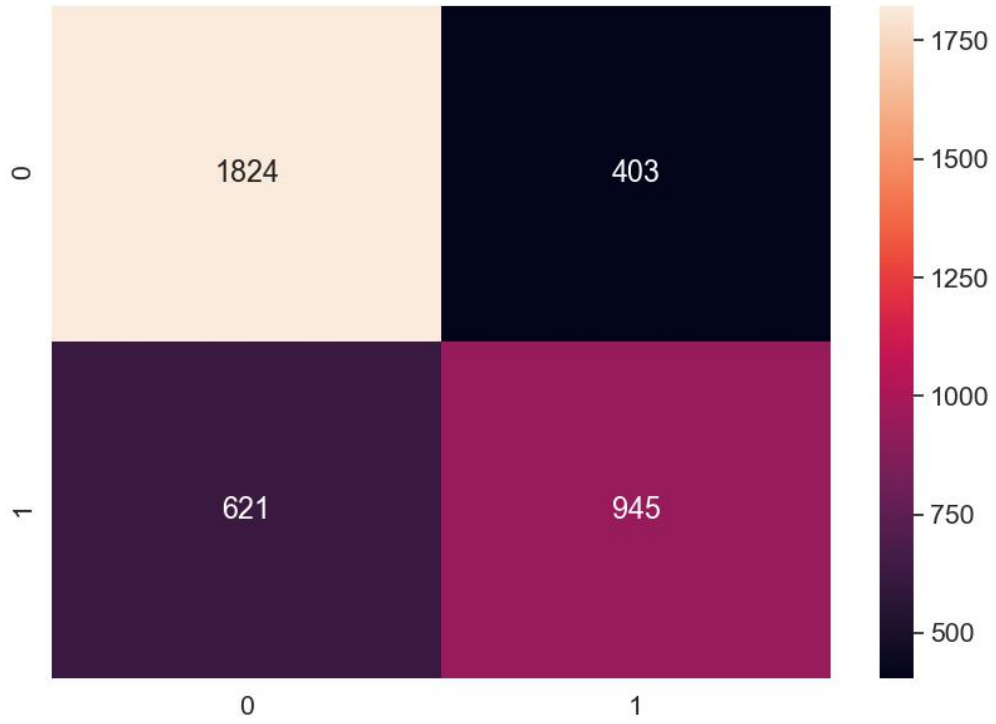
We have found that Random Forest gives the best accuracy, so we use *K-fold cross validation* and 2 rounds of *Grid Search (entropy criterion)* to further optimize our predictions from Random Forest.
K-Folds Cross Validation is a method of training a model by dividing the data set into n number of folds, or divisions. Now, the training of the data is done on (n-1) of those divisions with the n^{th} division being used for testing. This enhances the accuracy, Grid Search is an exhaustive search through each and every value for a given estimator. This also enhances the accuracy of the model.
We store the scores of each fit to obtain the final output:

Index	Model	Accuracy	Precision	Recall	F1 Score
0	Logistic Regression	0.659373	0.602699	0.51341	0.554483
1	SVM (linear)	0.651991	0.58965	0.516603	0.550715
2	SVM (RBF)	0.680464	0.634091	0.534483	0.580042
3	Decision Tree	0.661482	0.592037	0.579183	0.585539
4	Random Forest (n=100)	0.724492	0.700848	0.58046	0.634998
5	Random Forest (n=100, GSx2 + Entropy)	0.730029	0.701039	0.603448	0.648593

6.2.3 Confusion Matrix:

Now that we have finally obtained our best model and optimized it as well, we can represent the exact predictions and their correctness through a *Confusion Matrix*.

Confusion Matrix is a 2x2 matrix where each cell represents a prediction. The first cell [0,0] represents the number of positive predictions that we are correct (true positive), the second cell [0,1] represents number of positive predictions that are actually negative (false positive), the third cell [1,0] represents number of negative predictions that are actually positive (false negative), the fourth cell [1,1] represents number of correctly predicted false cases. Usually, the incorrect predictions of the third cell are fatal errors and must be minimized as much as possible.



This is the output confusion matrix we have obtained from our predictions, where we have 1824 true positive predictions, 403 false positive predictions, 621 false negative predictions, 945 true negative predictions. Darker color represents a lower value of number of predictions in that category.

7. Conclusion

After testing accuracy of multiple models, namely - Logistic Regression, SVM, Decision Tree and Random Forest and calculating their prediction accuracy, we have found that Random Forest provides the best accuracy (around 72.45%) which we have further optimized through Grid Search to obtain a final accuracy of 73.00%, which shows our algorithm is quite accurate, especially considering we are working with 27,000 entities. It has provided us with an indication of which of these 27,000 users are likely to churn. We have purposefully left the date of the expected churn open-ended because we are focused on only gauging the features that indicate the disengagement with the product, and not the exact manner(like time-frame) in which users will disengage. The aim is to distinctly single out those customers who are likely to churn so the company may engage with them again and rekindle their interest in the service. **In conclusion, we have obtained an accurate model (Random Forest) and predicted possible customers' churning out at 73.00% accuracy so that churn-rate can be effectively minimized by the company.**

8. Future Scope

With a final prediction accuracy of approximately 73.00% on a data set of 27,000 entities, **the selected model has proven to be considerably effective, especially so after optimization**. Therefore it can be used as the basis or framework for any such subscription based e-company to predict customer churning and preemptively interact with them to reduce the overall churn rate.

While the reduction of churn-rate is a long term plan, on a more immediate time frame, **the algorithm can also be used to test the success of any kind of subscription launched**, since the prediction made by our project, that is- whether or not an individual is likely to be subscribed or not is essentially a measure of prolonged success of a service.

The financial limitations of the users can also be studied so that the price range can accordingly be increased or decreased to **increase the number of customers purchasing the subscriptions or boost profit obtained from the current set of subscribed users**.

With only minor to moderate modifications, the algorithm can be utilized on multiple variants of subscription based services. Also worth noting is that since we test the accuracy of predictions by fitting on multiple models, **we can accommodate for a large range of scales of data and use the most appropriate model**. Therefore, this **project provides has a very wide scope for usage in any e-company looking to reduce churn-rate**.

9. Bibliography

References Used:

1. Basics of Python:
   - https://www.w3schools.com/python/python_reference.asp
2. Python Libraries:
   - https://numpy.org/
   - https://pandas.pydata.org/
   - https://matplotlib.org/3.1.1/tutorials/introductory/pyplot.html
3. Sci-Kit Learn:
   - https://scikit-learn.org/stable/getting_started.html
4. Seaborn:
   - https://seaborn.pydata.org/introduction.html