Phase 3

Customer Churn Analysis Prediction Priyadrashini engineering college

Introduction

Financial institutions have many clients close their accounts or migrate to other institutions. As a result, this has made a significant hole in sales and may significantly impact yearly revenues for the current fiscal year, leading stocks to plummet and market value to fall by a decent percentage. A team of business, product, engineering, and data science professionals has been assembled to halt this decline.

The objective of this tutorial is that we want to build a model to predict, with reasonable accuracy, the customers who are going to churn soon.

A customer having closed all their active accounts with the bank is said to have churned. Churn can be defined in other ways as well, based on the context of the problem. A customer not transacting for six months or one year can also be defined as churned based on the business requirements.

The Dataset

In the customer churn modeling dataset, we have 10000 rows (each representing a unique customer) with 15 columns: 14 features with one target feature (`Exited`). The data is composed of both numerical and categorical features:

`Exited` — Whether the customer churned or not.

- `CustomerId`: A unique ID of the customer.
- `CreditScore`: The credit score of the customer,
- `Age`: The age of the customer,
- `Tenure`: The number of months the client has been with the firm.
- `Balance`: Balance remaining in the customer account,
- `NumOfProducts`: The number of products sold by the customer.
- `EstimatedSalary`: The estimated salary of the customer.

Categorical Features:

- `Surname`: The surname of the customer.
- `Geography`: The country of the customer.
- `Gender`: M/F
- `HasCrCard`: Whether the customer has a credit card or not.
- `IsActiveMember`: Whether the customer is active or not.

If you don't have a Kaggle account, the dataset can be downloaded here or here.

Questioning the Data

Here, the objective is to understand the data further and distill the problem statement and the stated goal. In the process, if more data/context can be obtained, that adds to the result of the model performance:

- No date/time column. A lot of useful features can be built using date/time columns.
- When was the data snapshot taken? There are certain customer features like `Balance`, `Tenure`, `NumOfProducts`, `EstimatedSalary`, which will have different values across time.
- Are all these features about the same single date or spread across multiple dates?
- How frequently are customer features updated?
- Will it be possible to have the values of these features over some time as opposed to a single snapshot date?
- Some customers who have exited still have a balance in their account or a non-zero `NumOfProducts`. Does this mean they had churned only from a specific product and not the entire bank, or are these snapshots just before they churned?
- Some features like number and kind of transactions can help us estimate the degree of activity of the customer, instead of trusting the binary variable `IsActiveMember`.
- Customer transaction patterns can also help us ascertain whether the customer has churned or not. For example, a customer might transact daily/weekly vs. a customer who transacts annually.

We don't have answers to all of these questions, and we will only use what we have.

Once the dataset is downloaded, put it in the current working directory.

Let's install the dependencies of this tutorial:

```
$ pip install ipython==7.22.0 joblib==1.0.1 lightgbm==3.3.1 matplotlib
numpy pandas scikit_learn==0.24.1 seaborn xgboost==1.5.1
```

Copy

Let's import the libraries:

Phase 3

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.preprocessing import LabelEncoder, OneHotEncoder, StandardScaler
from sklearn.feature_selection import RFE
from sklearn.linear_model import LogisticRegression
from sklearn.svm import SVC
from sklearn.decomposition import PCA
from sklearn.tree import DecisionTreeClassifier, export_graphviz
from sklearn.pipeline import Pipeline
from sklearn.model_selection import train_test_split
from lightgbm import LGBMClassifier
from sklearn.metrics import roc_auc_score, recall_score, confusion_matrix, classification_report
import subprocess
import joblib
# Get multiple outputs in the same cell
from IPython.core.interactiveshell import InteractiveShell
InteractiveShell.ast_node_interactivity = "all"
# Ignore all warnings
import warnings
warnings.filterwarnings('ignore')
warnings.filterwarnings(action='ignore', category=DeprecationWarning)
pd.set_option('display.max_columns', None)
pd.set_option('display.max_rows', None)
```

Copy

Reading the dataset:

```
# Reading the dataset
dc = pd.read_csv("Churn_Modelling.csv")
dc.head(5)
```

Copy

Phase 3

	RowNumber	CustomerId	Surname	CreditScore	Geography	Gender	Age	Tenure	Balance	NumOfProducts	HasCrCard	IsActiveMember	EstimatedSalary	Exited
0	1	15634602	Hargrave	619	France	Female	42	2	0.00	1	1	1	101348.88	1
1	2	15647311	Hill	608	Spain	Female	41	1	83807.86	1	0	1	112542.58	0
2	3	15619304	Onio	502	France	Female	42	8	159660.80	3	1	0	113931.57	1
3	4	15701354	Boni	699	France	Female	39	1	0.00	2	0	0	93826.63	0
4	5	15737888	Mitchell	850	Spain	Female	43	2	125510.82	1	1	1	79084.10	0

Copy

Related: *Recommender Systems using Association Rules Mining in Python*.

Exploratory Data Analysis

Let's see the dimension of the dataset:

```
# Dimension of the dataset

dc.shape
```

Copy

```
(10000, 14)
```

Copy

Let's see some statistics of the data. The first line describes all numerical columns, where the second describes categorical columns:

```
dc.describe(exclude= ['O']) # Describe all numerical columns

dc.describe(include = ['O']) # Describe all categorical columns
```

Copy

Phase 3

	RowNumber	CustomerId	CreditScore	Age	Tenure	Balance	NumOfProducts	HasCrCard	IsActiveMember	EstimatedSalary	Exited
count	10000.00000	1.000000e+04	10000.000000	10000.000000	10000.000000	10000.000000	10000.000000	10000.00000	10000.000000	10000.000000	10000.000000
mean	5000.50000	1.569094e+07	650.528800	38.921800	5.012800	76485.889288	1.530200	0.70550	0.515100	100090.239881	0.203700
std	2886.89568	7.193619e+04	96.653299	10.487806	2.892174	62397.405202	0.581654	0.45584	0.499797	57510.492818	0.402769
min	1.00000	1.556570e+07	350.000000	18.000000	0.000000	0.000000	1.000000	0.00000	0.000000	11.580000	0.000000
25%	2500.75000	1.562853e+07	584.000000	32.000000	3.000000	0.000000	1.000000	0.00000	0.000000	51002.110000	0.000000
50%	5000.50000	1.569074e+07	652.000000	37.000000	5.000000	97198.540000	1.000000	1.00000	1.000000	100193.915000	0.000000
75%	7500.25000	1.575323e+07	718.000000	44.000000	7.000000	127644.240000	2.000000	1.00000	1.000000	149388.247500	0.000000
max	10000.00000	1.581569e+07	850.000000	92.000000	10.000000	250898.090000	4.000000	1.00000	1.000000	199992.480000	1.000000

	Surname	Geography	Gender
count	10000	10000	10000
unique	2932	3	2

```
╟────────┼─────────┼────────────┼────────╢
╟ top    ┼ Smith   ┼ France     ┼ Male   ╢
╟────────┼─────────┼────────────┼────────╢
╟ freq   ┼ 32      ┼ 5014       ┼ 5457   ╢
╚════════╧═════════╧════════════╧════════╝
```

Copy

Checking the number of unique customers:

```
# Checking number of unique customers in the dataset
dc.shape[0], dc.CustomerId.nunique()
```

Copy

```
(10000, 10000)
```

Copy

This means each row corresponds to a customer.

Let's see the churn distribution:

```
# churn value Distribution
dc["Exited"].value_counts()
```

Copy

```
0    7963
1    2037
Name: Exited, dtype: int64
```

Copy

The data set is imbalanced from the above result, with a significant chunk of existing customers relative to their churned peers.

Below, we group by `Surname` to see the average churn value:

```
dc.groupby(['Surname']).agg({'RowNumber':'count', 'Exited':'mean'}

).reset_index().sort_values(by='RowNumber', ascending=False).head()
```

Copy

```
╔═══════╤═══════════╤═══════════╤═══════════╗
```

	Surname	RowNumber	Exited
2473	Smith	32	0.281250
1689	Martin	29	0.310345
2389	Scott	29	0.103448
2751	Walker	28	0.142857
336	Brown	26	0.192308

Copy

Or grouping by `Geography`:

```
dc.groupby(['Geography']).agg({'RowNumber':'count', 'Exited':'mean'}

).reset_index().sort_values(by='RowNumber', ascending=False)
```

Copy

	Geography	RowNumber	Exited
0	France	5014	0.161548
1	Germany	2509	0.324432
2	Spain	2477	0.166734

Copy

From what we see above, customers from Germany have a higher exiting rate than average.

Univariate Plots of Numerical Variables

- Plotting `CreditScore` as a boxplot:

Data Preprocessing

So, in summary, here's what we're going to do:

Phase 3

- We will discard the `RowNumber` column.
- We will discard `CustomerID` as well since it doesn't convey any extra info. Each row pertains to a unique customer.
- Features can be segregated into non-essential, numerical, categorical, and target variables based on the above.

In general, `CustomerID` is a handy feature based on which we can calculate many user-centric features. Here, the dataset is not sufficient to calculate any extra customer features.

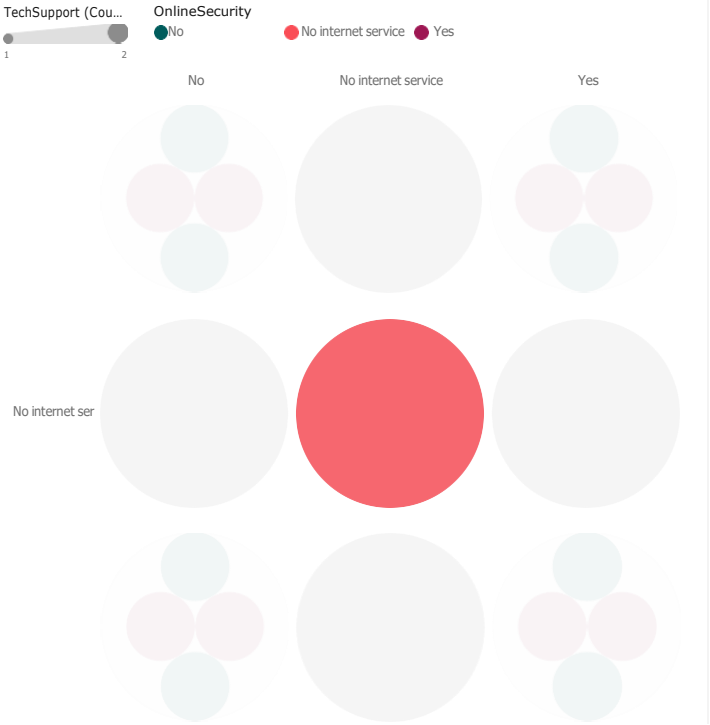
contact

1) tharun(team leader)
2)lokesh
3)sasi kumar
4)selva
5)michal

Tab 1

MultipleLines by customerID colored by gender

InternetService hierarchy colored by OnlineSecurity and sized by TechSupport



PaymentMethod, Churn and PaperlessBilling

PaymentMethod	Churn	PaperlessBilling
Bank transfer (automatic)	No	No
		Yes
	Yes	No
		Yes
Credit card (automatic)	No	No
		Yes
	Yes	No
		Yes
		No