ChurnPrediction

December 21, 2021

1 Churn prediction

The objective of this exercise is to build a model which can predict in advance, the merchants who are currently **active** on stripe but are likely to **stop transacting** in the **near future**.The data we have available is that of around 1.5 million transactions across 2 years for 14,351 customers to train this model.

1.1 Identifying churn

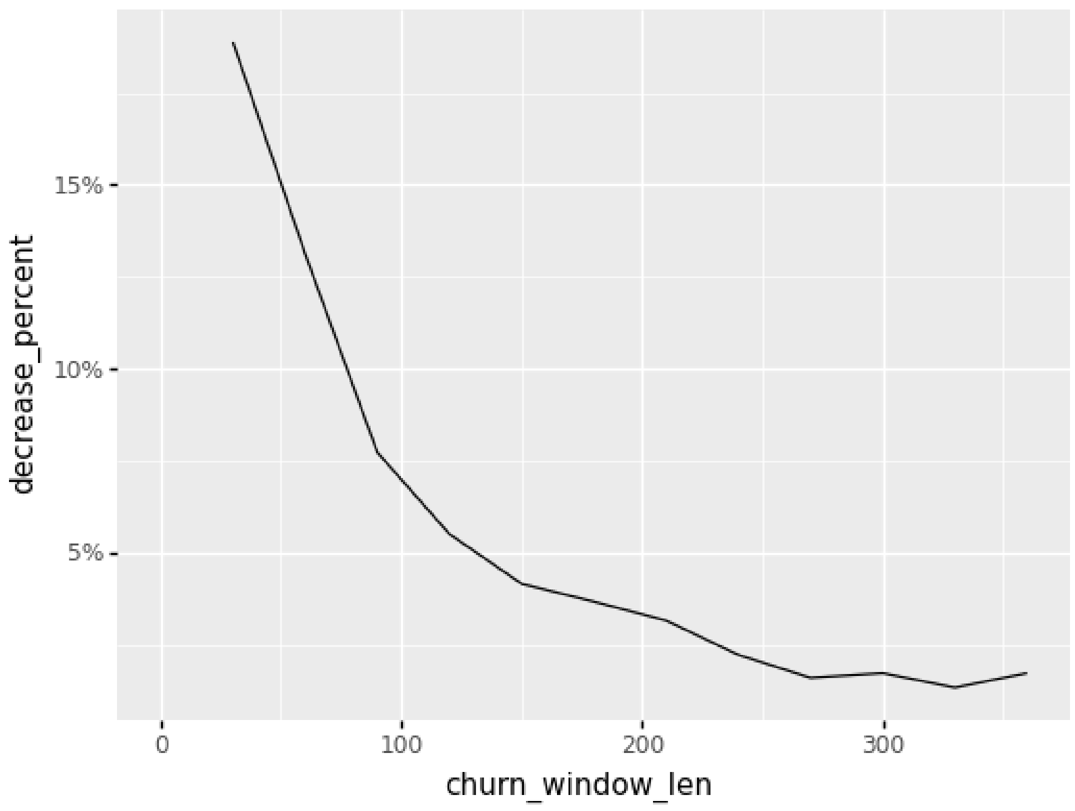
A merchant is said to have churned if he has stopped transacting with sripe over a given observation period. The length of this observation period is an assumption we have to make. There are two key consideration for chosing an appropriate lengh for the observation period -

1. The observation period should be long enough so that inactivity in the observation period can be inferred as the merchant having stopped using stripe. If we choose a short period, say 1 week, it is possible that the merchants we identifiy as having churned, might not have had any transactions in a given week, but may have transactions at a later point in time. This would make our churn identification inaccurate.
2. The length of the window should be short enough so that -
   - We are able to identify churned merchants from the given dataset, which only has 2 years worth of data. If we choose a long window, say 18 months, we will not have enough data to identify the churn indicator for a majority of the customers and might not have enough data to train a model.
   - The churn prediction from the model is actually actionable. This is much more of a subjective criterial determined by the actual use case of the model. We might care more about the customers not likely to transact withing the next 30 days, in which case, the model shouls be trained to predict ove a similar time frame.

1.2 Chosing the right performance window

```
   churn_window_len  churn_percent  n_merchants  decrease_percent
0               0.0       1.000000         6433               NaN
1              30.0       0.811130         6433         18.886989
2              60.0       0.679776         6433         13.135396
3              90.0       0.602363         6433          7.741334
4             120.0       0.547179         6433          5.518421
5             150.0       0.505518         6433          4.166019
```

```
6              180.0       0.468677         6433          3.684129
7              210.0       0.436966         6433          3.171149
8              240.0       0.414581         6433          2.238458
9              270.0       0.398414         6433          1.616664
10             300.0       0.381004         6433          1.741023
11             330.0       0.367480         6433          1.352402
12             360.0       0.350070         6433          1.741023
```



```
<ggplot: (-9223371944074406548)>
```

Above we can see churn rates of the same cohort of active merchants(all active merchants in 2033) when we vary the size of the observation period. As we increase the churn window from 0 to 360, the churn rate starts to stablize around 240 days, where most merchants who have churned in a 360 day observation period have already churned within a 240 day window and the incremental benefit of increasing the observation period is marginal. This indicates that a 240 day window for identification of churn will work reasonably well for training our model.

1.3 Merchants already churned

Now that we have fixed our definition of churn to be the absence of any transaction for 240 days, we can identfiy merchatns that have already churned at least once in the dataset.

4017 out of the 14351 merchants have churned at one point during thse 2 years.

1.4 Sampling

To generate the training data, I have chosen to sample merchant activity every 30 days. At each sampling date, if the merchant has not churned as of that date, I compute features over the past 240 days, the training period, to capture the merchant activity.The target variable would indicate wheather of not the customer had any transactions over the performance window, which we have chosen to be from the sampling date to 240 days after the sampling date. I did not choose a higher sampling frequnecy because oversampling can result in a huge number of samples that can take a lot of time to compute and would not have a lot of additional information given that the merchants seem to be operating at about a 1 transaction per week median frequency. Additionally, the choice to keep the training period also 240 days is arbitrary. Ideally this would be a parameter we would tune depending on model performance and data availability.

1.5 Features

To capture merchant activity over the training window, I compute the following features. Each of them capture one of 3 attributes -

1. Average activity during the training window
2. Recent activity
3. Recent activity normalised by average activity in the performance window

```
{'total_amount_usd': 'Total amount transacted through stripe over the training
window',
 'count_txns': 'Number of transactions over the training window',
 'avg_txn_amount': 'Average transaction amount over the training window',
 'count_txns_l30d': 'Number of transactions in the last 30 days',
 'count_txns_l60d': 'Number of transactions in the last 60 days',
 'count_txns_l90d': 'Number of transactions in the last 90 days',
 'count_txns_l30d_ratio': 'Number of transactions in the last 30 days divided by
total transactions over the training window',
 'count_txns_l60d_ratio': 'Number of transactions in the last 60 days divided by
total transactions over the training window',
 'count_txns_l90d_ratio': 'Number of transactions in the last 90 days divided by
total transactions over the training window'}
```

1.6 Model Training

```
   sample size  bad rate      iv     ks  missing                feature
0        51764    0.3032  0.2557  21.13      0.0       total_amount_usd
1        51764    0.3032  0.2735  21.04      0.0             count_txns
2        51764    0.3032  0.0054   3.25      0.0         avg_txn_amount
3        51764    0.3032  1.3531  51.26      0.0        count_txns_l30d
4        51764    0.3032  1.2477  48.32      0.0        count_txns_l60d
5        51764    0.3032  1.0598  42.23      0.0        count_txns_l90d
6        51764    0.3032  1.0685  48.07      0.0  count_txns_l30d_ratio
```

```
7        51764    0.3032  0.4920  33.98      0.0  count_txns_l60d_ratio
8        51764    0.3032  0.2410  24.16      0.0  count_txns_l90d_ratio
```

Above is information value for all our features over the training set, which indicates the strength of a feature's relationship with churn. The churn rate for the sampled training set is around 30% on a traning set size of 51,764 after sampling. Looking at the information value, almost all features except for average transaction amount seem to be well correlated with churn. Clealy, transaction activity has a lot correlation future churn. Also ,the more recent transaction activity is more powerfully correlated with future churn. We go ahead with training an xgboost model on this traning set with these features.

1.6.1 Model evaluation

```
Model results. Train AUC : 0.8787946226704462. Test AUC : 0.851806268393682
```

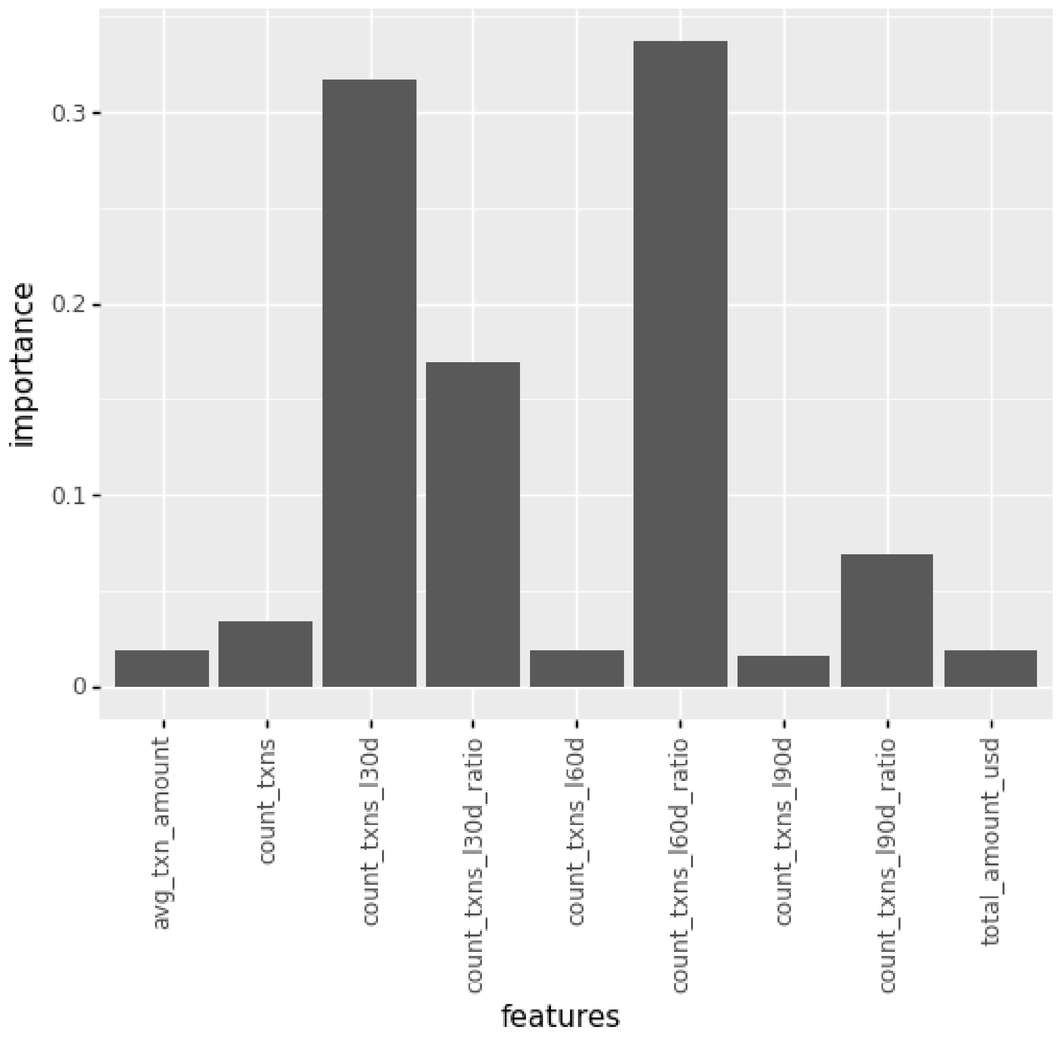
KS table

```
        min_prob  max_prob  events  nonevents  total  eventrate event_rate  \
Decile
1       0.680263  0.933116    1380        315   1695   0.814159     26.10%
2       0.606068  0.679982    1132        588   1720   0.658140     21.41%
3       0.492019  0.605952     909        801   1710   0.531579     17.19%
4       0.325948  0.491943     660       1043   1703   0.387551     12.48%
5       0.225464  0.325637     522       1191   1713   0.304729      9.87%
6       0.158576  0.225438     316       1393   1709   0.184903      5.98%
7       0.096585  0.158527     181       1527   1708   0.105972      3.42%
8       0.050022  0.096504     111       1597   1708   0.064988      2.10%
9       0.017068  0.050011      62       1646   1708   0.036300      1.17%
10      0.000351  0.016972      15       1694   1709   0.008777      0.28%

       nonevent_rate cum_eventrate cum_noneventrate    KS
Decile
1              2.67%        26.10%            2.67%  23.4
2              4.99%        47.50%            7.66%  39.8
3              6.79%        64.69%           14.45%  50.2
4              8.84%        77.17%           23.29%  53.9
5             10.10%        87.05%           33.39%  53.7
6             11.81%        93.02%           45.20%  47.8
7             12.95%        96.44%           58.14%  38.3
8             13.54%        98.54%           71.68%  26.9
9             13.96%        99.72%           85.64%  14.1
10            14.36%       100.00%          100.00%   0.0
```

The final classifier trained is a pretty good one. The test AUC is almost 0.85. However, from a usage perspective, we need to choose a threshold for the predicted probability of churn above which we will take necessary actions to avoid churn. This obviously comes down to the trade off between the cost of said action and our need to cut down on churn. The above table will help us navigate this tradeoff. For example, if we keep the probability threshold at around 0.5(row 3), we will have to action on 30% of the population on average. However, this 30% population would account for 65% of all churners on average.

1.6.2 Feature importances



```
<ggplot: (-9223371944075838280)>
```

Within the model , the most important 2 features are those relating to recent activity and recent activity nomalised by overall level of activity which makes a lot of sense.

1.7 Scope for improvement -

1. Explore more hyperparamenter combinations and try to run other models possible
2. Feature engineering by creating new features as well as tweaking the windows for time window features. Tweaking the training window to make it shorter. Clearly most of the value lies in recent activity so we might be able to make the training window shorter

1.8 Steps Further -

1. We need to choose an appropriate threshold for the probability outpit from the model, above which we will comission an action to curtail possible churn.
2. We need to find similar data to be able to validate this model.