Stellar Competition

Procedure

Train-valid split

Feature Engineering

“Divide-and-Conquer” Modeling

Stacking

Rounding

More-days lagging

How to “cheat”?

Although it is not valid to bundle current transactions or use future value, we can get some ideas of it in this way.

Unit price = fee_charged / (#ops + bool(fee_account)). It is a CONSTANT in one sequence. So the way to get yesterday’s unit price is very simple. You can achieve 0.000 error in this way.

Feature Engineering 0: Group data by sequence

If the unit prices by sequences are all the same, we just need to predict the unit price by sequences instead of transactions.

Aggregation methods are needed for some features and we will discuss that later.

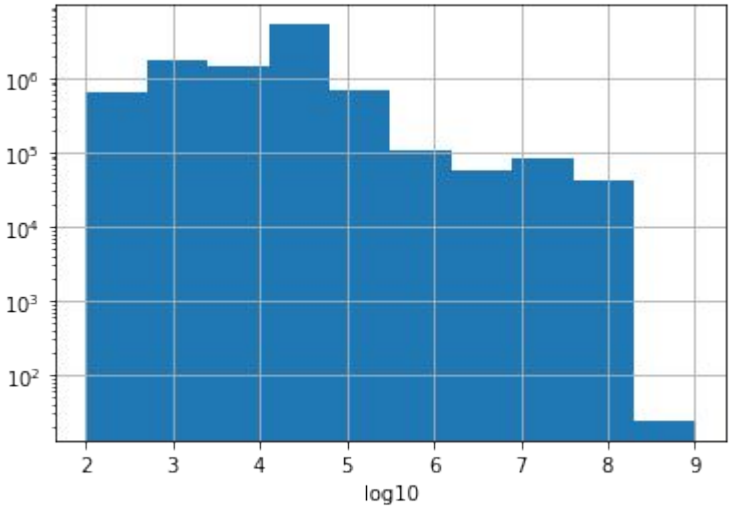
Feature Engineering 1: Last Unit Price

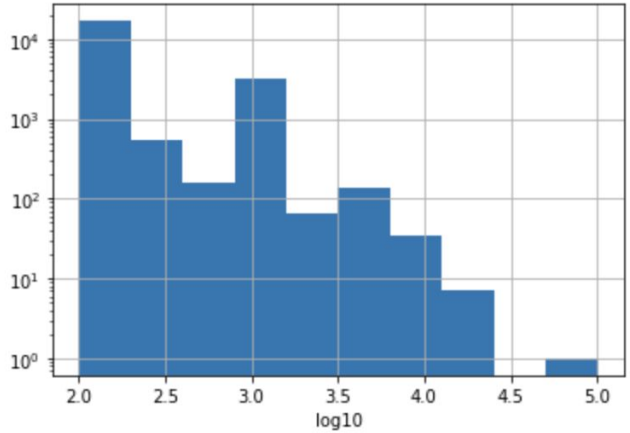
We cannot directly calculate the unit price in test set.

Unit Price = prior_min_fee_charged / min(#ops + bool(fee_account))

Feature Engineering 2: Price Binning

How to aggregate the price? We created 6 bins (left: unit_bid_price, right: unit_fee)





Feature Engineering: All columns

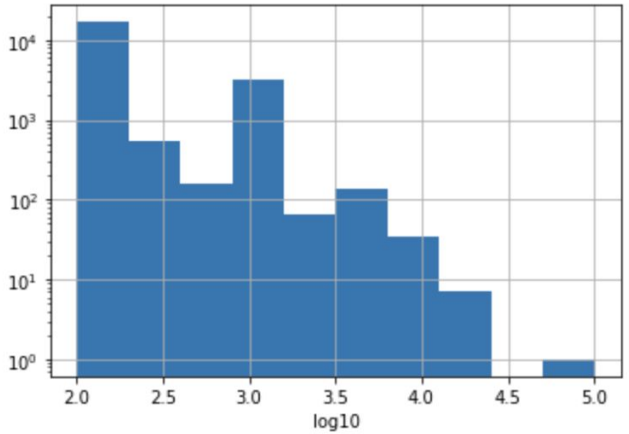
'transaction_operation_count', 'avg_fee', 'price2', 'price3', 'price4', 'price5', 'price6', 'price7', 'prior_successful_operation_count', 'prior_failed_operation_count', 'prior_successful_transaction_count', 'prior_failed_transaction_count'

All the columns without the word “prior” are shifted by 1 (which means last seq)

Modeling: Decision Tree

Why not neural network? Why decision tree?

```
100.0     15161
1000.0     2835
110.0      1365
200.0       457
121.0       129
          ...
202.0         1
213.0         1
722.0         1
750.0         1
405.0         1
Name: avg_fee, Length: 346, dtype: int64
```




```
100.0      15161
1000.0      2835
200.0        457
5000.0       124
1100.0        79
500.0         25
3000.0        20
10000.0       10
1200.0         8
20000.0        7
300.0          6
700.0          5
2000.0         5
1500.0         5
5200.0         4
400.0          4
900.0          4
3500.0         3
600.0          2
1300.0         1
3300.0         1
1400.0         1
Name: avg_fee, dtype: int64
```

Divide-and-Conquer

Decision Trees we learned in COMS 4995: Random Forest, Xgboost, LightGBM, Gradient Boosting, Histogram Gradient Boosting

We would like to add another model: ExtraTreeRegressor (This actually performs the best in the result.)

Stacking

0.2 * random_forest + 1.2 * ExtraTree + 0.5 * Xgboost - 0.9 * LightGBM

Discarded: Histogram Gradient Boosting, Gradient Boosting

Rounding

If x in [15000, 20000], x = 20000

If x in [10000, 15000), x = 10000

If x in [2000, 3000), x = 2000

If x in [1000, 2000), x = 1000

Multi-lagging

max(txn_count) < 950 (rolling 2-4 seqs ago): prediction would be 100

Thank you!