lpotato case study

Neal Fultz
Apr 11

Executive Summary

- Decomposed problem into two parts
  - Approved or not
  - Value, given they are approved or not
- Fit model for each part
- Put it back together using high school math
- Worked fine, reasonably well calibrated, reasonably fast
- Implemented wrapper class and package for easy usage / deployment.

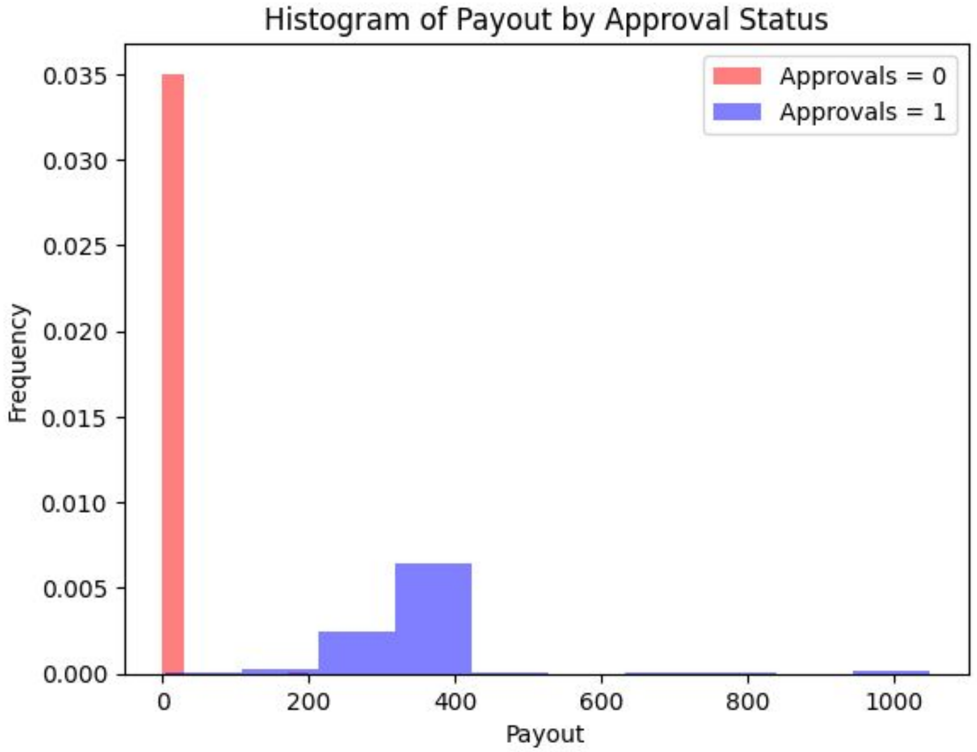
Data

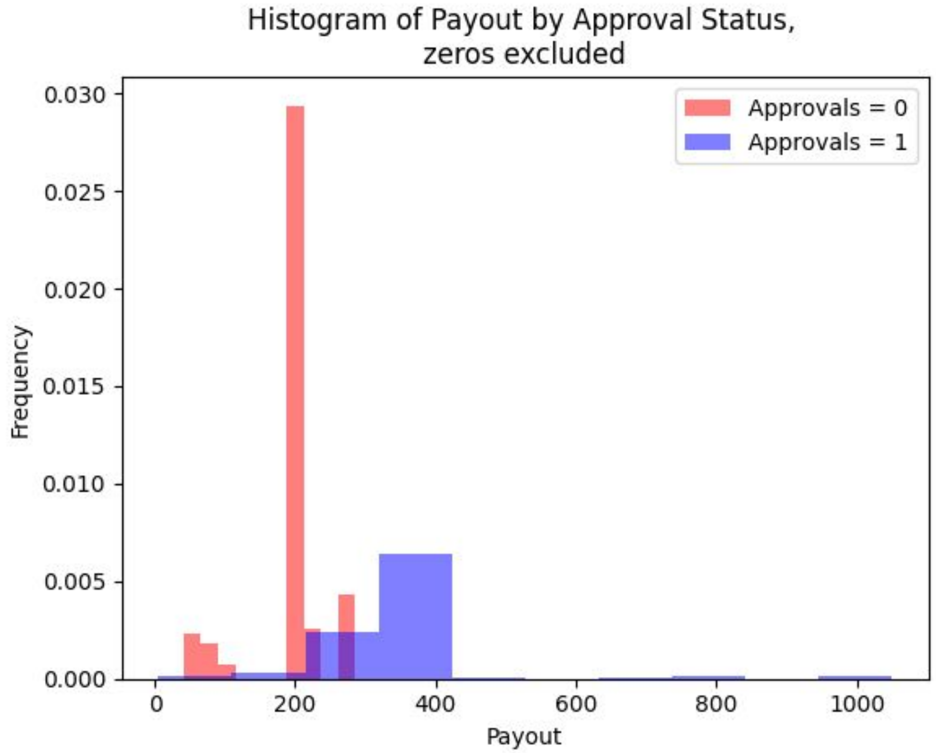
Predictors

- campaign
- keyword
- keyword_match_type
- device
- device_model
- page_slug
- offer_position
- user_session_count
- average_household_income
- median_age
- offer_id
- advertiser_id
- card_category
- date
- hour

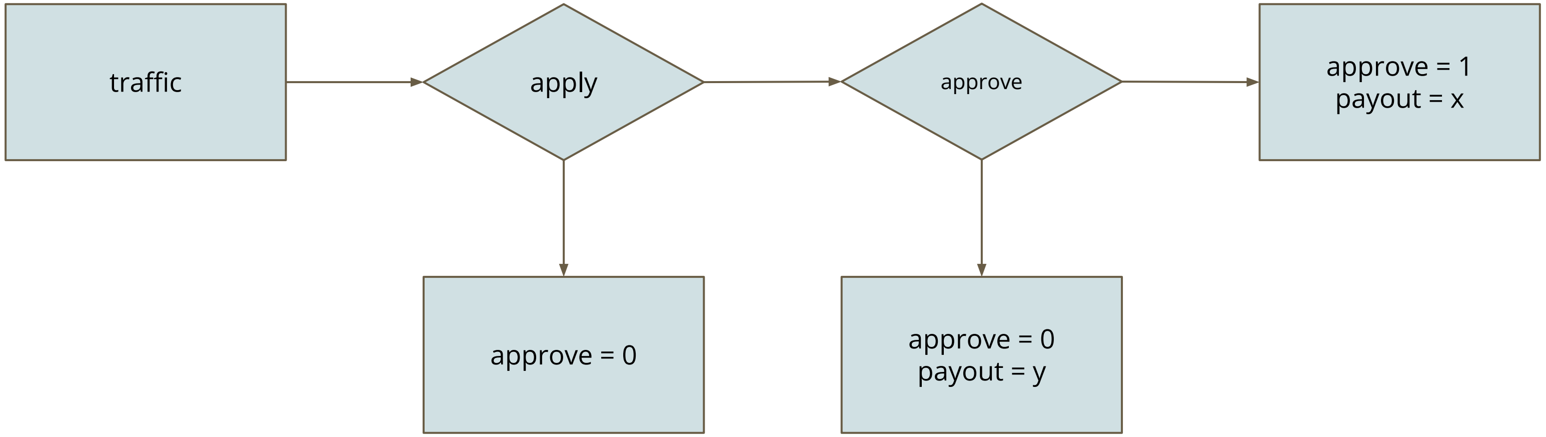
Outcomes

- approvals
- payout


Histogram of Payout by Approval Status
Approvals = 0
Approvals = 1
Frequency
0.035
0.030
0.025
0.020
0.015
0.010
0.005
0.000
0
200
400
600
800
1000
Payout


Histogram of Payout by Approval Status,
zeros excluded
Approvals = 0
Approvals = 1
Frequency
0.000
0.005
0.010
0.015
0.020
0.025
0.030
0
200
400
600
800
1000
Payout

DGP


traffic
apply
approve
approve = 1
payout = x
approve = 0
approve = 0
payout = y

Model

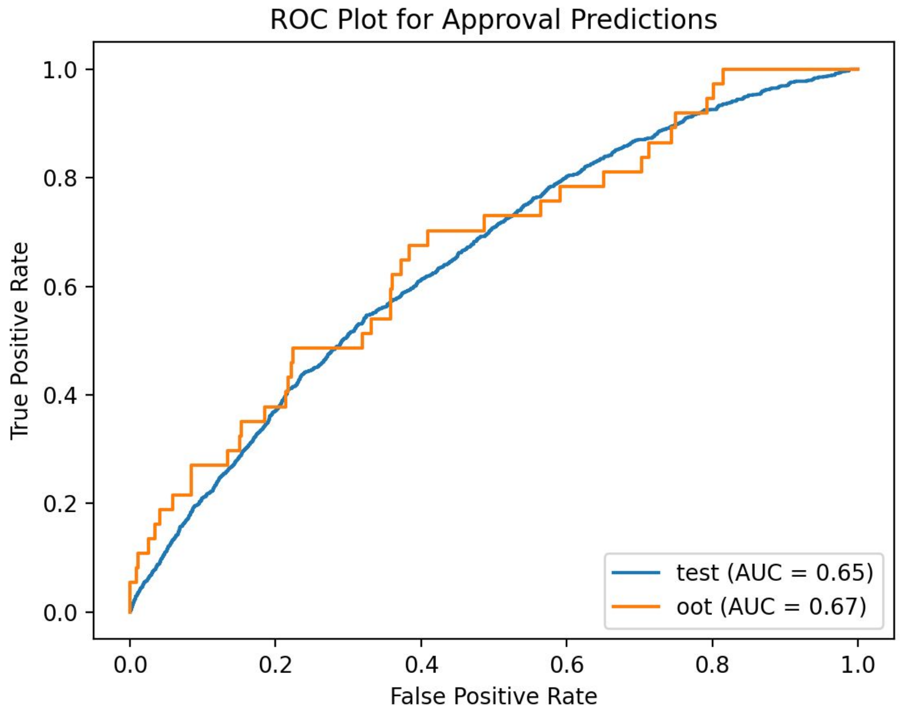
- Apply is a hurdle
- But not directly observed
  - Eg can’t tell difference between ‘didn’t apply’ and ‘not paid upon approval’
- Therefore, model the marginal
- But, approved is behind it
- Marginalize over that

- Build model of Approved
- Build model of Payout given Approve
- Combine
  - $E(P) = E(P|A)\ P(A) + E(P \mid {\sim}A)\ P({\sim}A)$

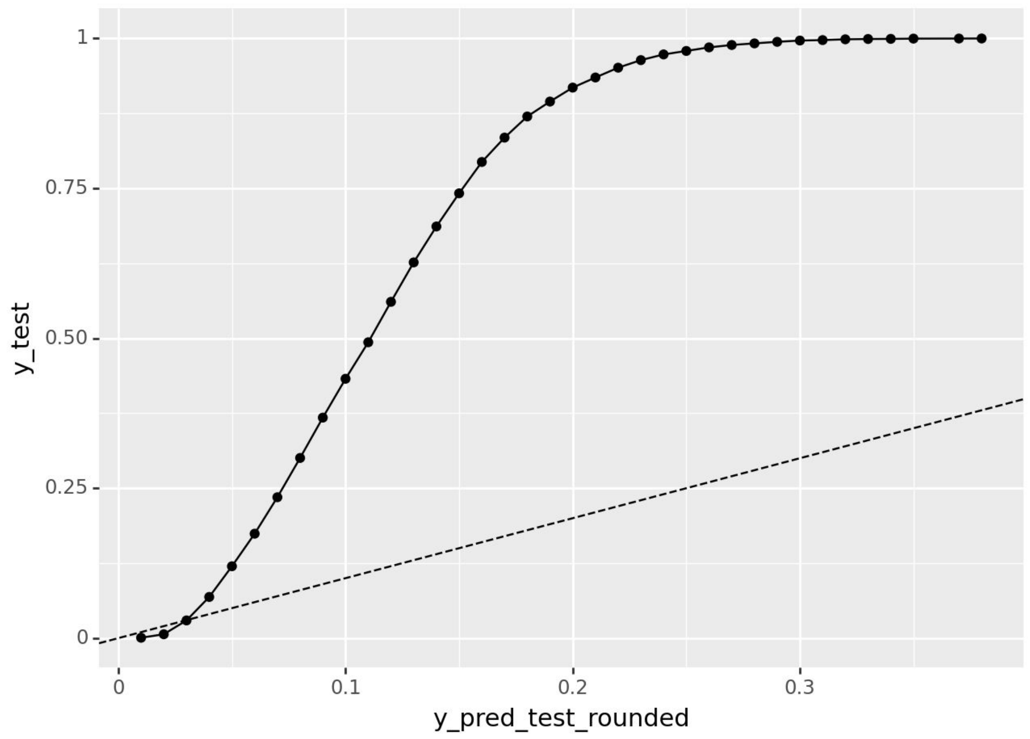
Model cont’d

- Fit each component using xgb
- Use hyperopt to tune
    - ~30 iterations each, in real world do a few thousand

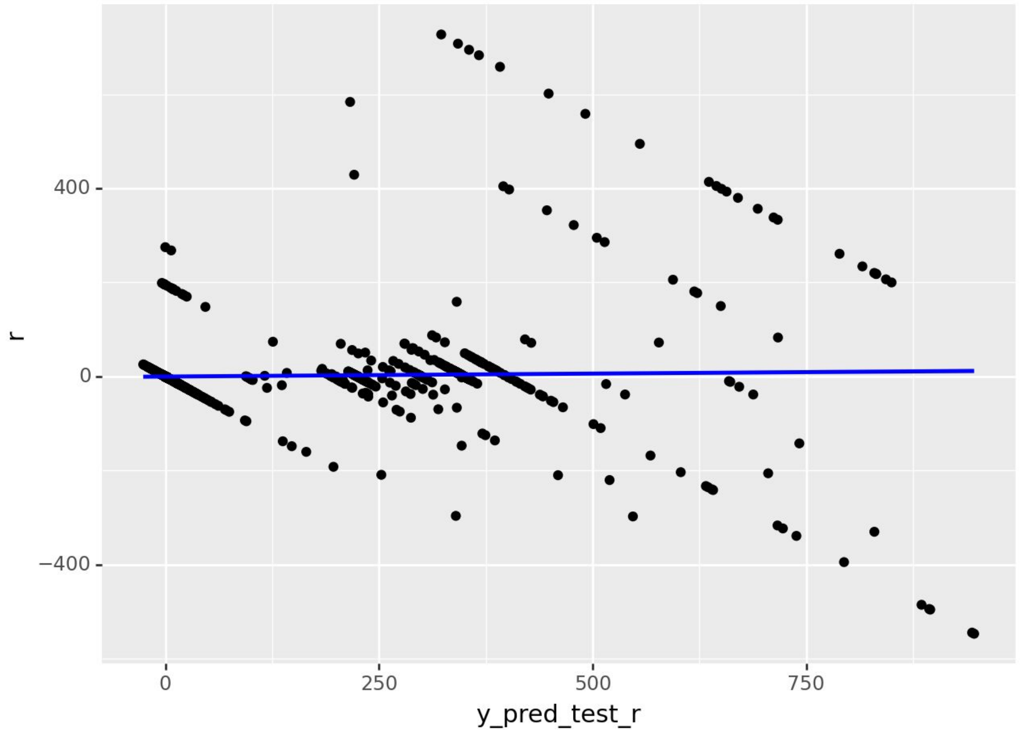
Model - P(Approve)



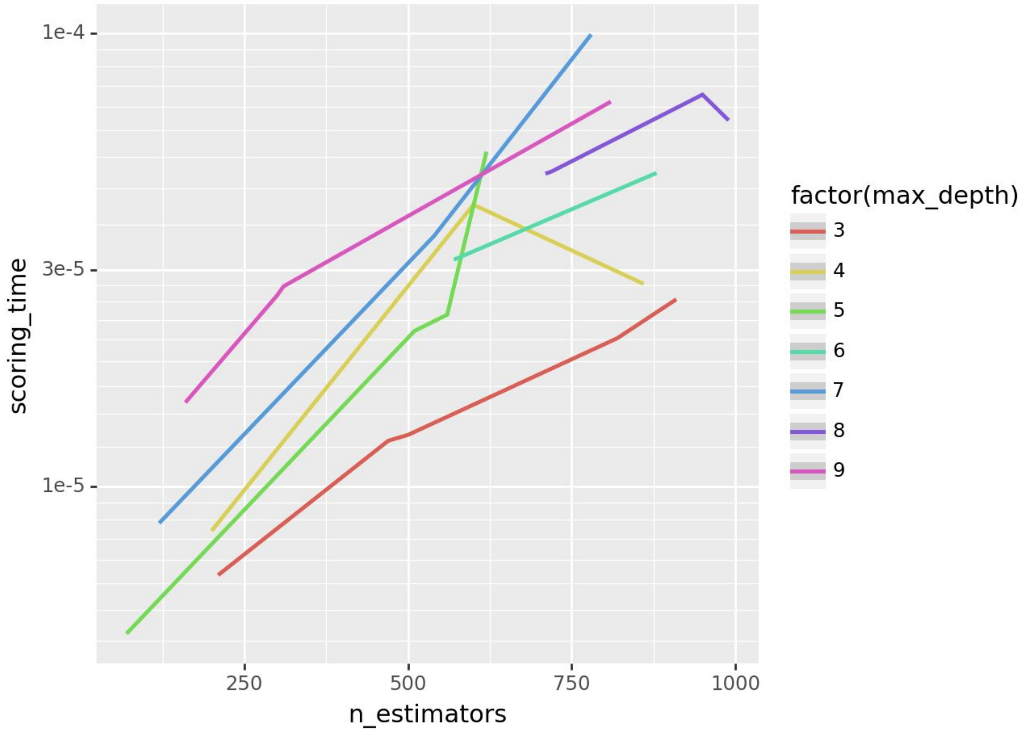
Model P(Approve)



Model - E(Payout | Approve)

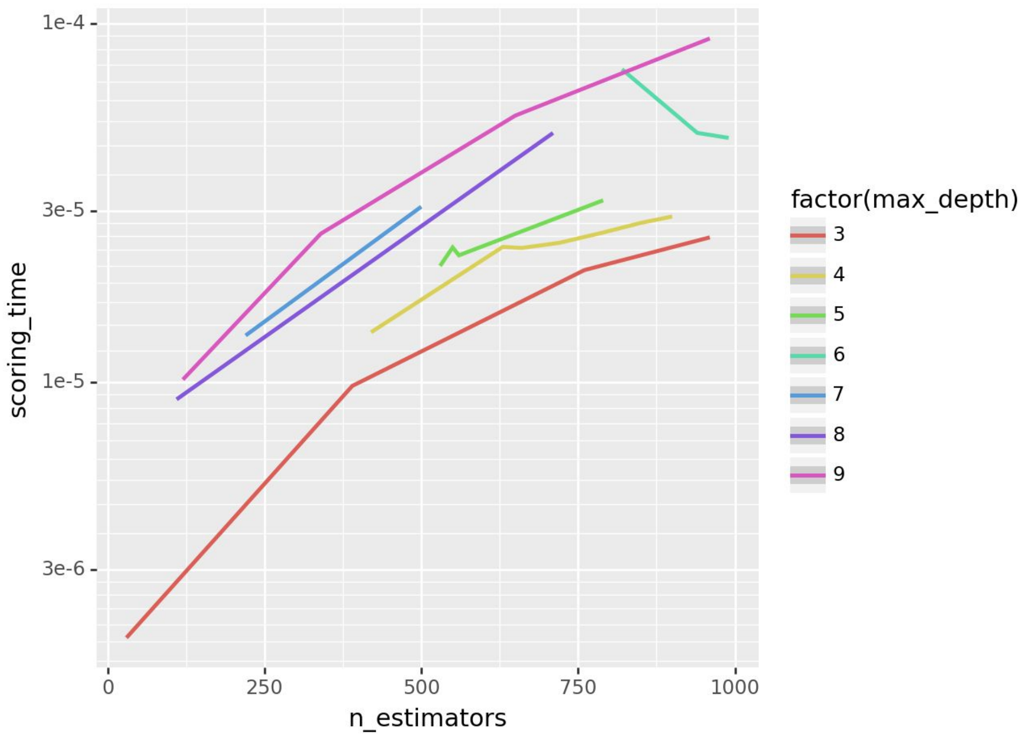


Timing - P(A)



1e-4
3e-5
1e-5
scoring_time
250
500
750
1000
n_estimators
factor(max_depth)
3
4
5
6
7
8
9

Timing - E(P | A)



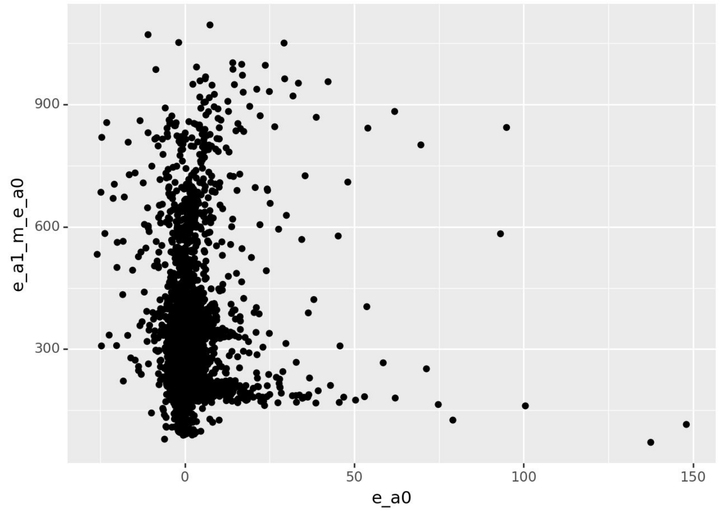
1e-4
3e-5
1e-5
3e-6
scoring_time
0
250
500
750
1000
n_estimators
factor(max_depth)
3
4
5
6
7
8
9

Q1. What other features could you think of to improve the performance of the models?

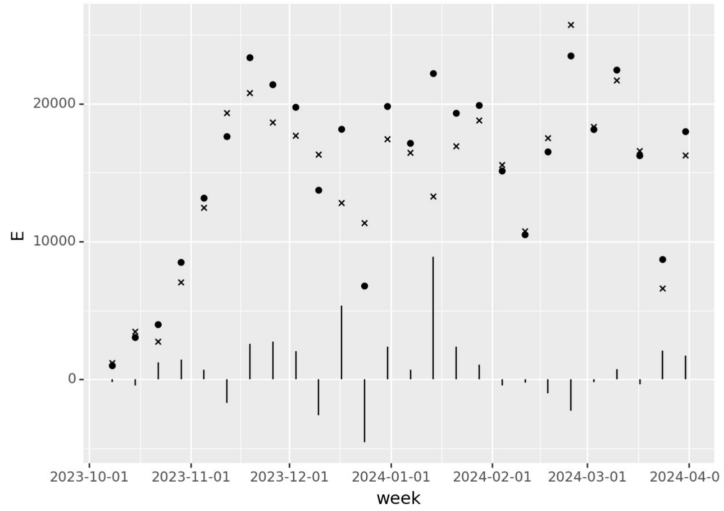
- Pingback event on “application completed”
- CPM vs CPC
- FICO score
    - Or upstream Experian, TU data
- Demogs / identities eg age/gender/race etc
    - NB regulations eg Equal Credit Opportunity Act, Fair Credit Reporting Act
    - Caveat emptor
- Other HTTP headers, referer metadata, etc
- Content of lander (advertorial copy, etc)
- Time / seasonality variables
    - Need timezone of user, full year of data, etc

Q2. On a weekly basis, how does the total predicted revenue compare to the actual revenue?

A) The clicks that generate revenue are assigned a higher value than the clicks that do not convert.



B) The total sum of the predicted click values is close to the actual revenue.



Python API

```
my_model = lpotato(‘p_xgb.json’, ‘e_xgb.json’, ‘meta.pkl’)

score = my_model.score(data)
```

NB: Not a full web service

demo

```
~$mkdir /tmp/test
~$cd /tmp/test
test$python3 -m venv venv
test$source venv/bin/activate
(venv) test$pip3 install git+https://github.com/nfultz/lpotato
Collecting git+https://github.com/nfultz/lpotato
  Cloning https://github.com/nfultz/lpotato to /tmp/pip-req-build-z4mavs0t
<SNIP>
Successfully installed joblib-1.3.2 lpotato-1.0.0 numpy-1.26.4 pandas-2.2.1
python-dateutil-2.9.0.post0 pytz-2024.1 scikit-learn-1.4.1.post1 scipy-1.13.0
six-1.16.0 threadpoolctl-3.4.0 tzdata-2024.1 xgboost-2.0.3

(venv) test$cd ..
(venv) tmp$cat line.txt
81207,JP-FBZ-SEM-Credit Cards-Balance Transfer-tCPA,best balance transfer
cards,b,iOS,iPhone,/top-balance-transfer-credit-cards,7.0,1,60 to 69,20 to
29,18313,2077,Rewards/Cash Back,2024-01-21,19,0,0.0

(venv) tmp$python3 -m lpotato p_xgb.json e_xgb.json df.pkl <line.txt
[24.017654]
```

Executive Summary

- Decomposed problem into two parts
  - Approved or not
  - Value, given they are approved or not
- Fit model for each part
- Put it back together using high school math
- Worked fine, reasonably well calibrated, reasonably fast
- Implemented wrapper class and package for easy usage / deployment.

Questions?