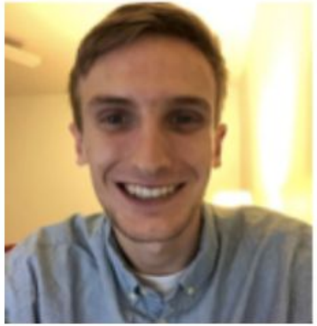
HCDR Team 1

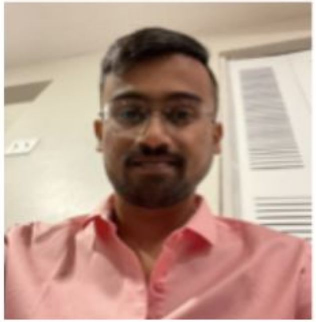
Phase 3

Zack Seliger



Keegan Moore

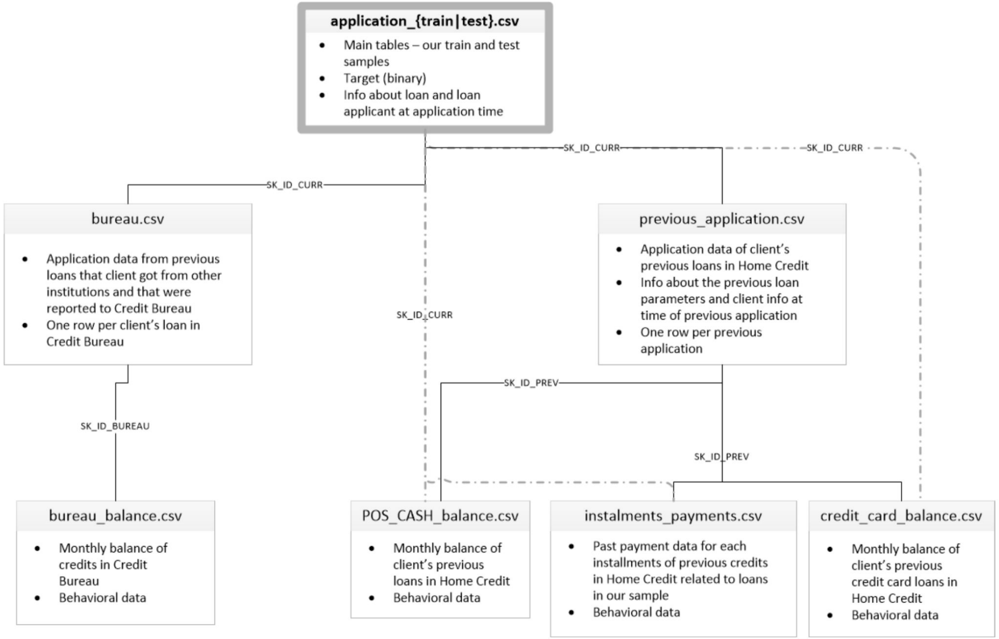
Jagan Lakku



Raja Simha Reddy

HCDR

The Kaggle HCDR problem has participants create machine learning models that can predict whether loan applicants will have trouble repaying a loan, based on some large datasets.



application_{train|test}.csv

- Main tables – our train and test samples
- Target (binary)
- Info about loan and loan applicant at application time

bureau.csv

- Application data from previous loans that client got from other institutions and that were reported to Credit Bureau
- One row per client's loan in Credit Bureau

previous_application.csv

- Application data of client's previous loans in Home Credit
- Info about the previous loan parameters and client info at time of previous application
- One row per previous application

bureau_balance.csv

- Monthly balance of credits in Credit Bureau
- Behavioral data

POS_CASH_balance.csv

- Monthly balance of client's previous loans in Home Credit
- Behavioral data

instalments_payments.csv

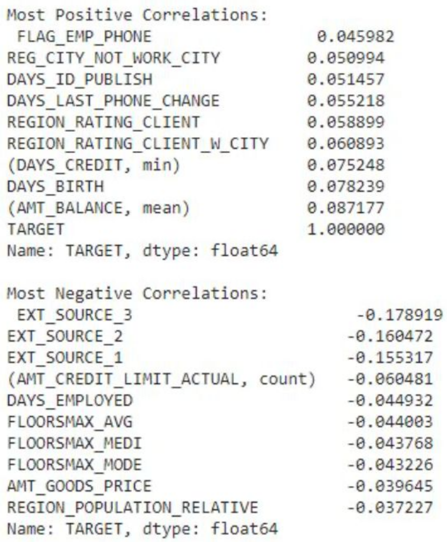
- Past payment data for each installments of previous credits in Home Credit related to loans in our sample
- Behavioral data

credit_card_balance.csv

- Monthly balance of client's previous credit card loans in Home Credit
- Behavioral data

In Earlier Phases...

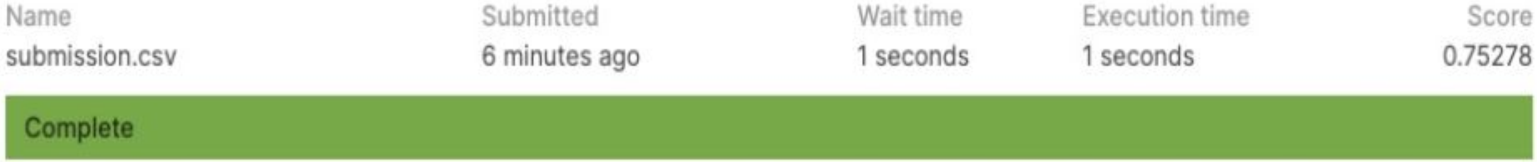
In phase 1 we grabbed data, did EDA, and made a baseline linear regression model that got a Kaggle score of .729.



```
Most Positive Correlations:
 FLAG_EMP_PHONE                  0.045982
REG_CITY_NOT_WORK_CITY          0.050994
DAYS_ID_PUBLISH                 0.051457
DAYS_LAST_PHONE_CHANGE          0.055218
REGION_RATING_CLIENT            0.058899
REGION_RATING_CLIENT_W_CITY     0.060893
(DAYS_CREDIT, min)              0.075248
DAYS_BIRTH                      0.078239
(AMT_BALANCE, mean)             0.087177
TARGET                          1.000000
Name: TARGET, dtype: float64

Most Negative Correlations:
 EXT_SOURCE_3                           -0.178919
EXT_SOURCE_2                           -0.160472
EXT_SOURCE_1                           -0.155317
(AMT_CREDIT_LIMIT_ACTUAL, count)       -0.060481
DAYS_EMPLOYED                          -0.044932
FLOORSMAX_AVG                          -0.044003
FLOORSMAX_MEDI                         -0.043768
FLOORSMAX_MODE                         -0.043226
AMT_GOODS_PRICE                        -0.039645
REGION_POPULATION_RELATIVE             -0.037227
Name: TARGET, dtype: float64
```

In phase 2 we did more rigorous feature engineering and EDA, as well as made tuned and untuned LGBM and XGBoost models. Our tuned LGBM model got a Kaggle score of .752.



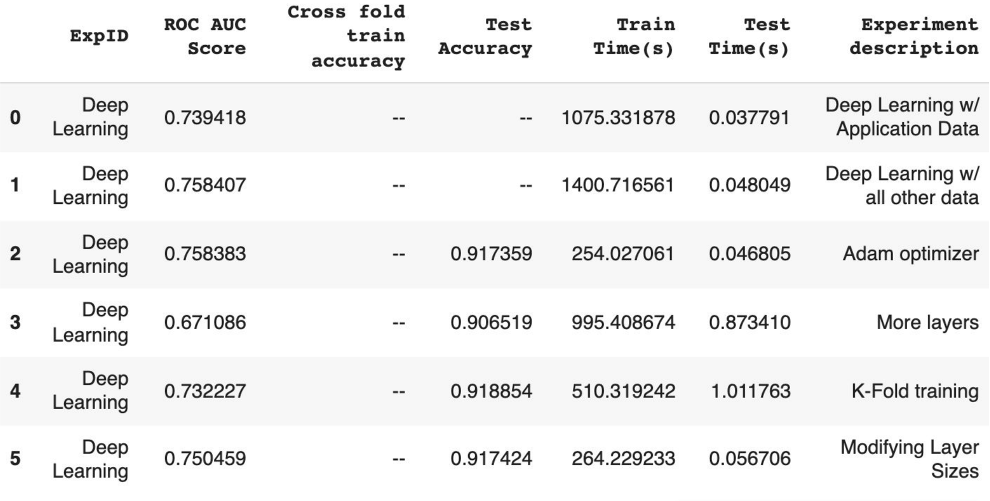
Name	Submitted	Wait time	Execution time	Score
submission.csv	6 minutes ago	1 seconds	1 seconds	0.75278

Complete

Phase 3 Models

In Phase 3, we tried many types of architectures, as well as changing optimizers and loss function.

ROC AUC Scores varied widely, as did training times.

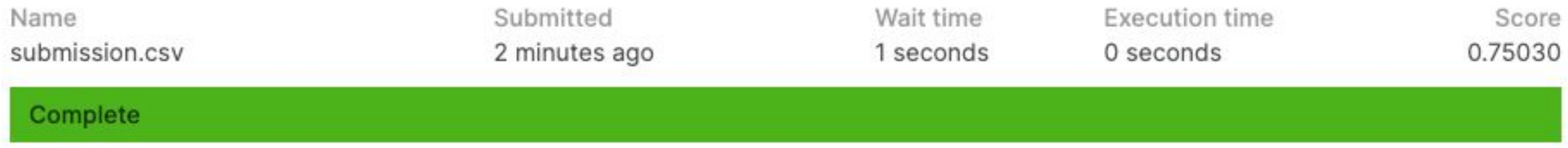


	ExpID	ROC AUC Score	Cross fold train accuracy	Test Accuracy	Train Time(s)	Test Time(s)	Experiment description
0	Deep Learning	0.739418	--	--	1075.331878	0.037791	Deep Learning w/ Application Data
1	Deep Learning	0.758407	--	--	1400.716561	0.048049	Deep Learning w/ all other data
2	Deep Learning	0.758383	--	0.917359	254.027061	0.046805	Adam optimizer
3	Deep Learning	0.671086	--	0.906519	995.408674	0.873410	More layers
4	Deep Learning	0.732227	--	0.918854	510.319242	1.011763	K-Fold training
5	Deep Learning	0.750459	--	0.917424	264.229233	0.056706	Modifying Layer Sizes

Best Model

Our best model used BCELoss and Adam optimizer. It had 173 input neurons, 1 layer of 20 neurons, and 2 output neurons,

Its Kaggle score was .750, making it better than our baseline of .729, but worse than our LGBM model of .752.



Name	Submitted	Wait time	Execution time	Score
submission.csv	2 minutes ago	1 seconds	0 seconds	0.75030

Complete

Conclusion

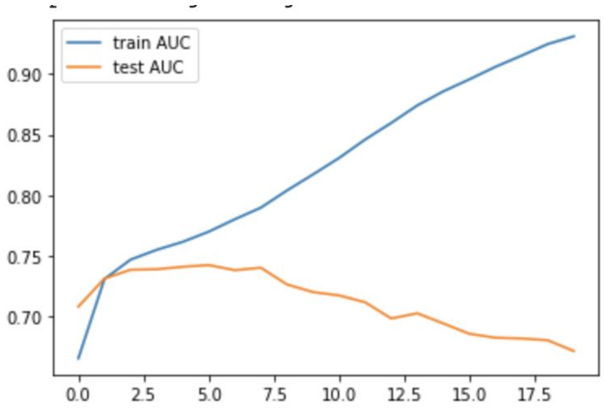
After implementing both single-layer and multi-layer neural networks, we found the following.

Challenges

Many of our models were prone to overfitting, especially those with more neurons and hidden layers.

Takeaways

We learned that sometimes simple is better. The more complicated we made our models, the worse it performed.



ROC AUC train vs test graph for large model without dropouts