AI for Quant

Goal

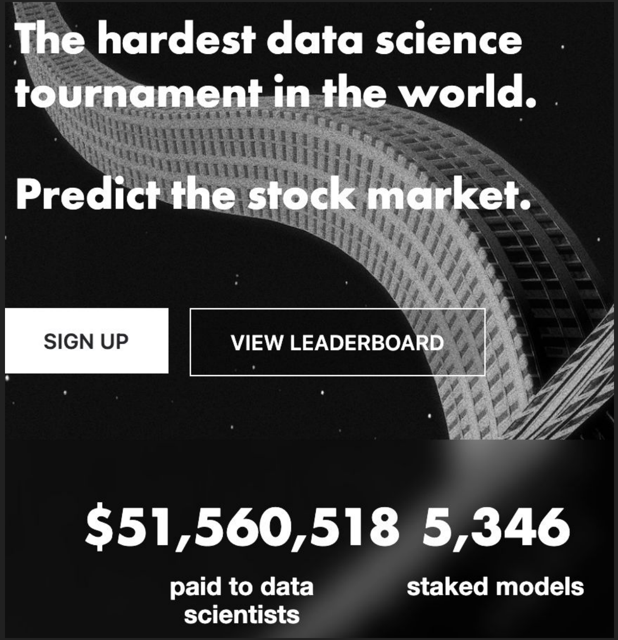
- Understand basics of Data Science in Quant
- Build our first model on Numerai
- Submit the result and start earning crypto
- Discuss how to go way further

Why you should follow along

- <2 minutes to get set up
- I'll give you $200 of stake credits for Numerai
- The best Numerai data scientists make 300% a year passively
- Data Science 101 crash course

Set Up Instructions

1. Go to Numer.ai and sign up



2. Open your Python IDE

```
!pip install numerapi
!pip install lightgbm
!pip install pandas
!pip install pyarrow
```

```python
from numerapi import NumerAPI
import pandas as pd
import lightgbm as lgbm
napi = NumerAPI()
```

Quant

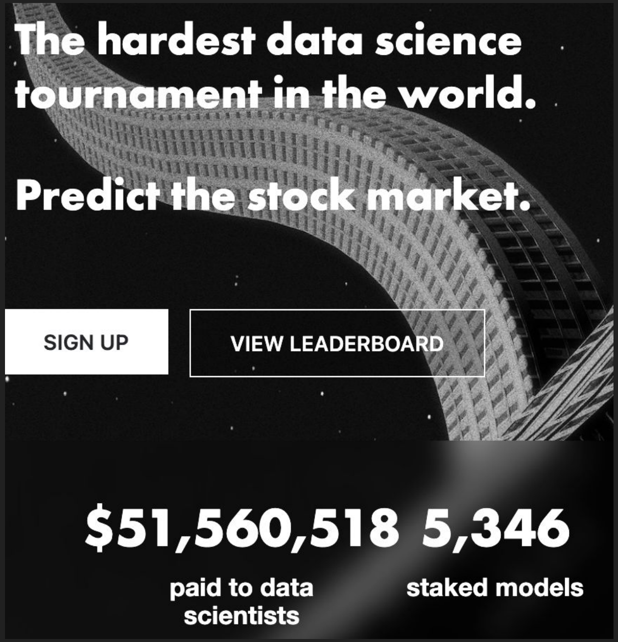
- Trading Stocks (or other instruments)
- Quant is just doing this using quantitative systematic methods
- A simple quant might devise simple rules: if PE ratio is low and growth is high, buy it
- That's 2 dimensions
- Today we're going to be using some more advanced methods to look at over 1000 dimensions

Numerai

- A "Decentralized AI Hedge Fund"
- Trade stocks, not correlated with the broader market
- ML models make all decisions
- None of our own models
- Trading 1bil every 2 months

Set Up Instructions

1. Go to Numer.ai and sign up



2. Open your Python IDE

```
!pip install numerapi
!pip install lightgbm
!pip install pandas
!pip install pyarrow
```

```
from numerapi import NumerAPI
import pandas as pd
import lightgbm as lgbm
napi = NumerAPI()
```

Getting the Data

```
napi.download_dataset("v4.1/train.parquet", "train.parquet")
napi.download_dataset("v4.1/live.parquet", "live.parquet")
```

```
df = pd.read_parquet("train.parquet")
df
```

id	era	data_type	feature_honoured_observational_balaamite	feature_polaroid_vadose_quinze	feature_untidy_withdrawn_bargeman
n003bba8a98662e4	0001	train	1.00	0.50	1.00
n003bee128c2fcfc	0001	train	0.50	1.00	0.25
n0048ac83aff7194	0001	train	0.50	0.25	0.75
n00691bec80d3e02	0001	train	1.00	0.50	0.50
n00b8720a2fdc4f2	0001	train	1.00	0.75	1.00
...	...	...	...	...	...
nffcc1dbdf2212e6	0574	train	0.00	0.25	0.00
nffd71b7f6a128df	0574	train	0.00	0.25	0.00
nffde3b371d67394	0574	train	0.25	0.25	0.50
nfff1a1111b35e84	0574	train	1.00	0.75	0.50
nfff2bd38e397265	0574	train	0.25	0.25	0.75

2420521 rows × 1617 columns

Downsample

605131 rows × 161 columns

```
# downsample to 1/10 as many feature cols
all_feature_cols = [c for c in df if c.startswith("feature_")]
keep_feature_cols = all_feature_cols[::10]

cols = ["era", "target"] + keep_feature_cols

# only keep every 4th row of data and our subset of cols
df = df.iloc[::4][cols]
```

```
df
```

	era	target	feature_honoured_observational_balaamite	feature_demolished_unfrightened_superpower
id				
n003bba8a98662e4	0001	0.25	1.00	0.50
n00b8720a2fdc4f2	0001	0.75	1.00	0.25
n018fc48e071e447	0001	0.50	0.50	0.00
n02fe92bf2c2a1b1	0001	0.75	1.00	0.00
n0393c0487c43940	0001	0.50	0.75	0.25
...	...	...	...	...
nfee9a1be7844c4d	0574	0.25	0.75	0.00
nff1243cde25232a	0574	0.50	0.25	1.00

Let's look at how good the features are alone

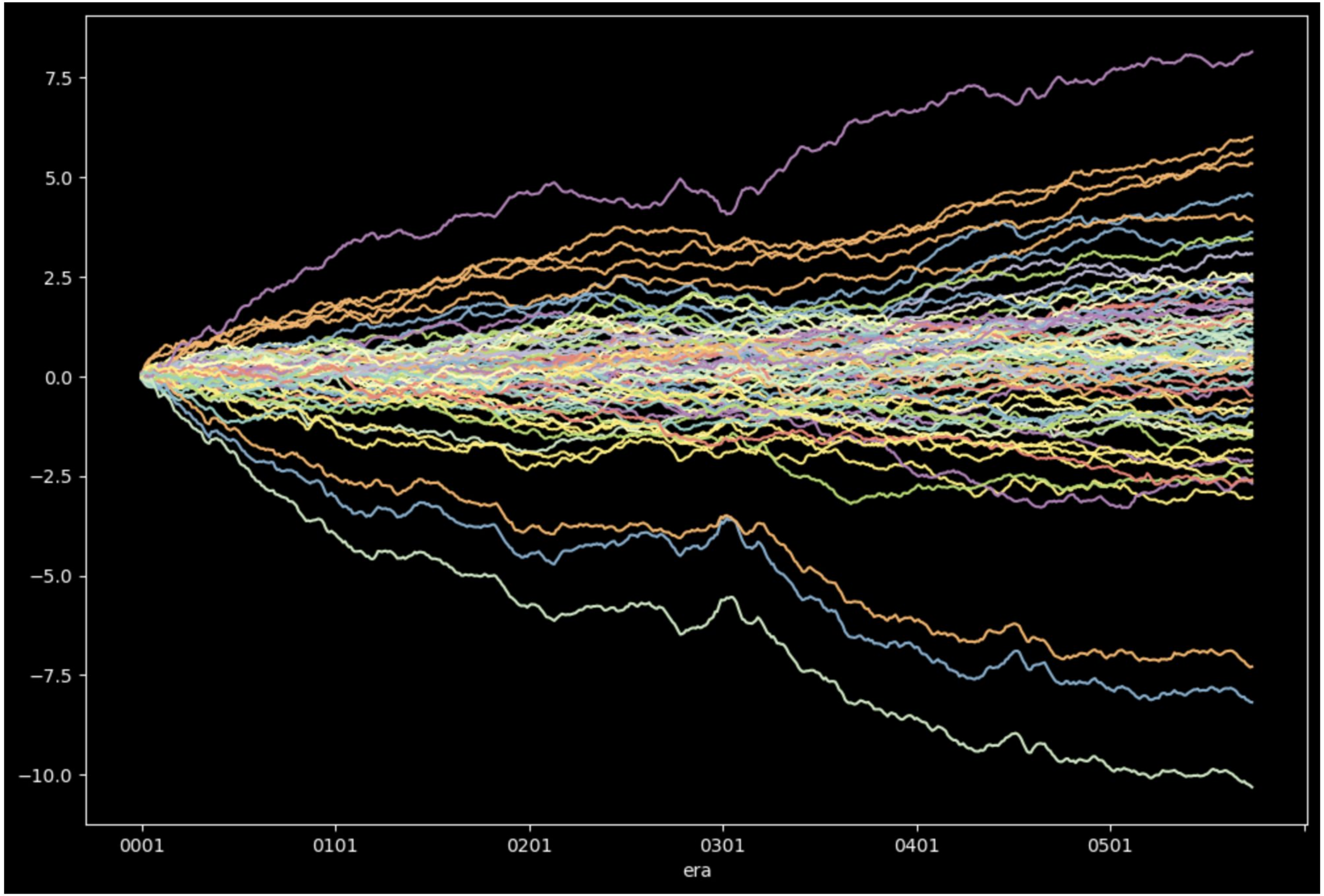
```
corrs_per_era = df.groupby("era").apply(lambda d: d[keep_feature_cols].corrwith(d["target"]))
```

```
corrs_per_era
```

	feature_honoured_observational_balaamite	feature_demolished_unfrightened_superpower	feature_cu
era			
0001	0.027187	-0.003769	
0002	0.011272	-0.023569	
0003	0.034478	-0.036151	
0004	0.007016	0.027885	
0005	-0.017088	-0.041120	
...	...	...	

```python
corrs_per_era.cumsum().plot(figsize=(12,8), legend=False)
```

```
<AxesSubplot: xlabel='era'>
```



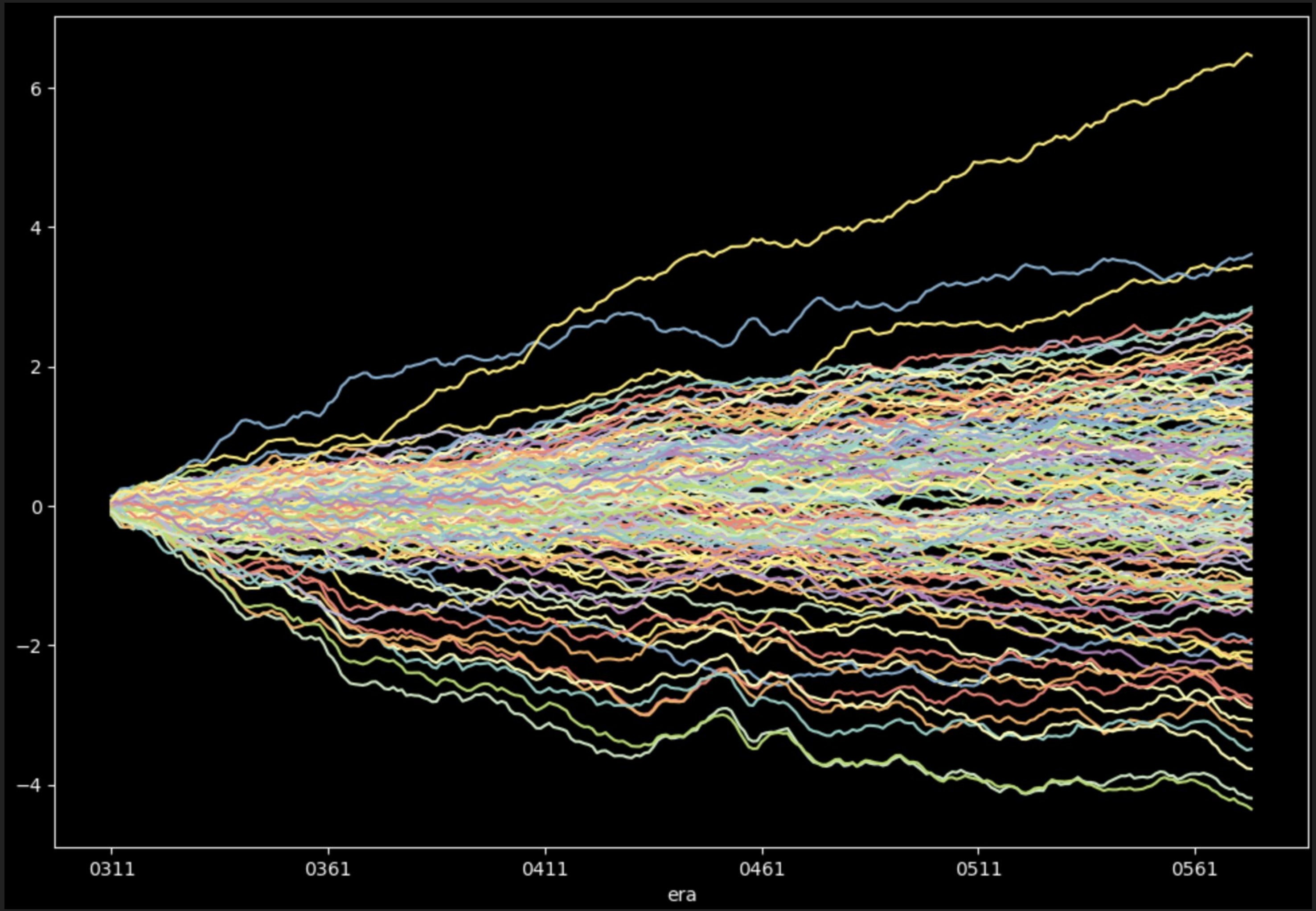
```python
# fit model
model = lgbm.LGBMRegressor(n_estimators=100)
model.fit(df[keep_feature_cols], df["target"])

# predict on live
live_df = pd.read_parquet("live.parquet")
live_preds = model.predict(live_df[keep_feature_cols])

# format and save for submission
submit_preds = pd.Series(live_preds, index=live_df.index)
submit_preds.to_frame("prediction").to_csv("live_preds.csv")
submit_preds
```

	prediction
id	
n00204dfabaef8e1	0.506670
n002101f62593be8	0.503350
n0022dc742690bba	0.508871
n0028d69d9082acb	0.483749
n0029e1bdff0b977	0.512874
...	...

Evaluating and Improving

```
def score_columns(df, columns):
    return df.groupby("era").apply(lambda d: d[columns].corrwith(d["target"]))
```

```
# split into train and test
train_df = df[df["era"] < "0300"]
test_df = df[df["era"] > "0310"]

# train model on train, predict on test
model.fit(train_df[keep_feature_cols], train_df["target"])
test_df["prediction"] = model.predict(test_df[keep_feature_cols])

# compare corrs per era for our prediction column vs each feature individually
oos_corrs_per_era = score_columns(test_df, keep_feature_cols+["prediction"])
```


6
4
2
0
−2
−4
0311
0361
0411
0461
0511
0561
era

```
oos_corrs_per_era.mean().sort_values()
```

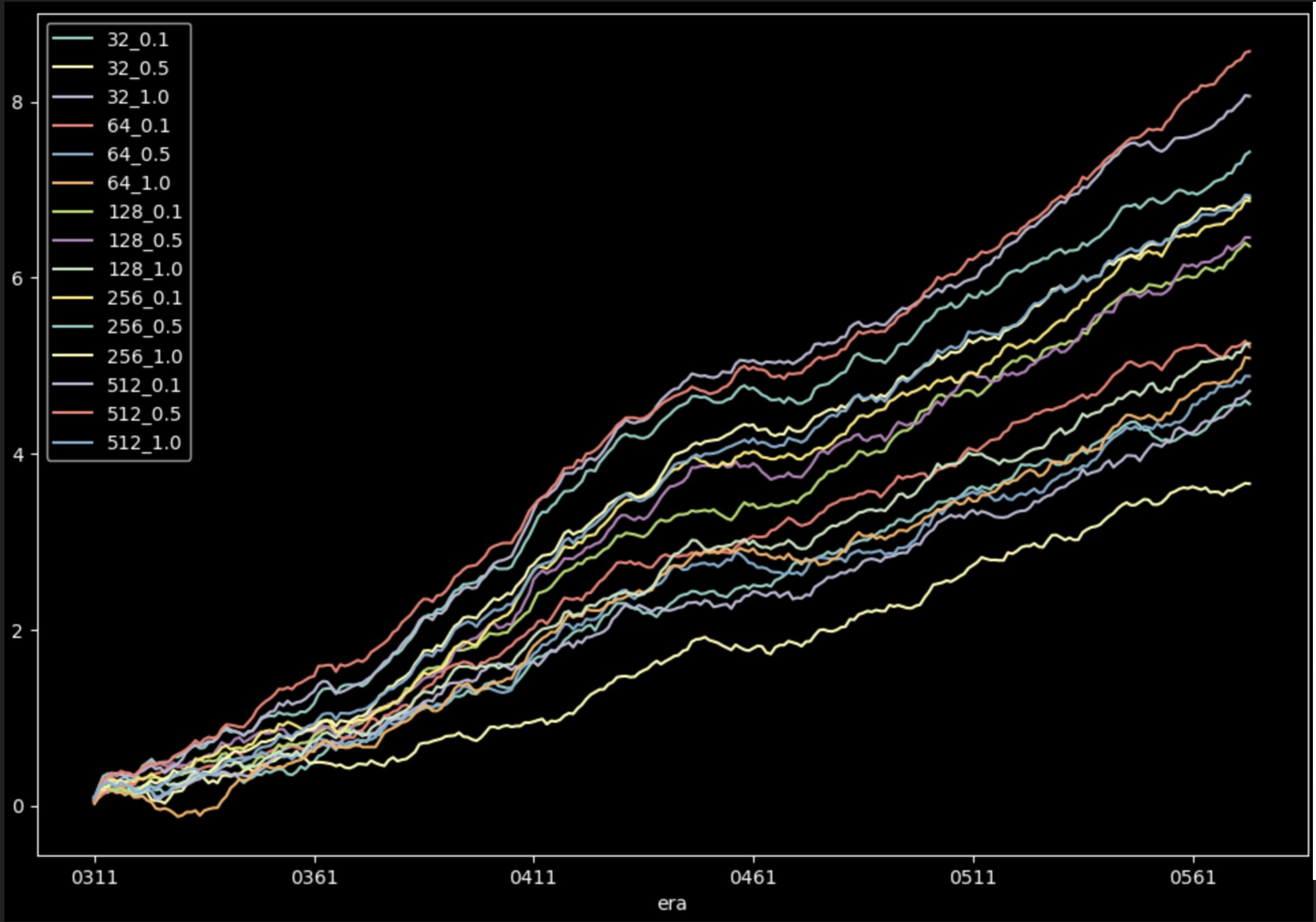
```
feature_encysted_conventionalized_dematerialization   -0.016484
feature_violated_telic_tuning                          -0.015893
feature_togate_unbailable_door                         -0.014302
feature_gaga_clinched_islamization                     -0.013223
feature_associate_unproper_gridder                     -0.012522
                                                          ...
feature_regurgitate_demolition_downstate                0.010617
feature_regrettable_liberating_crabber                  0.010785
feature_hunchbacked_unturning_meditation                0.013000
feature_nonnegotiable_errant_soya                       0.013672
prediction                                              0.024457
```

Run a few different models to compare

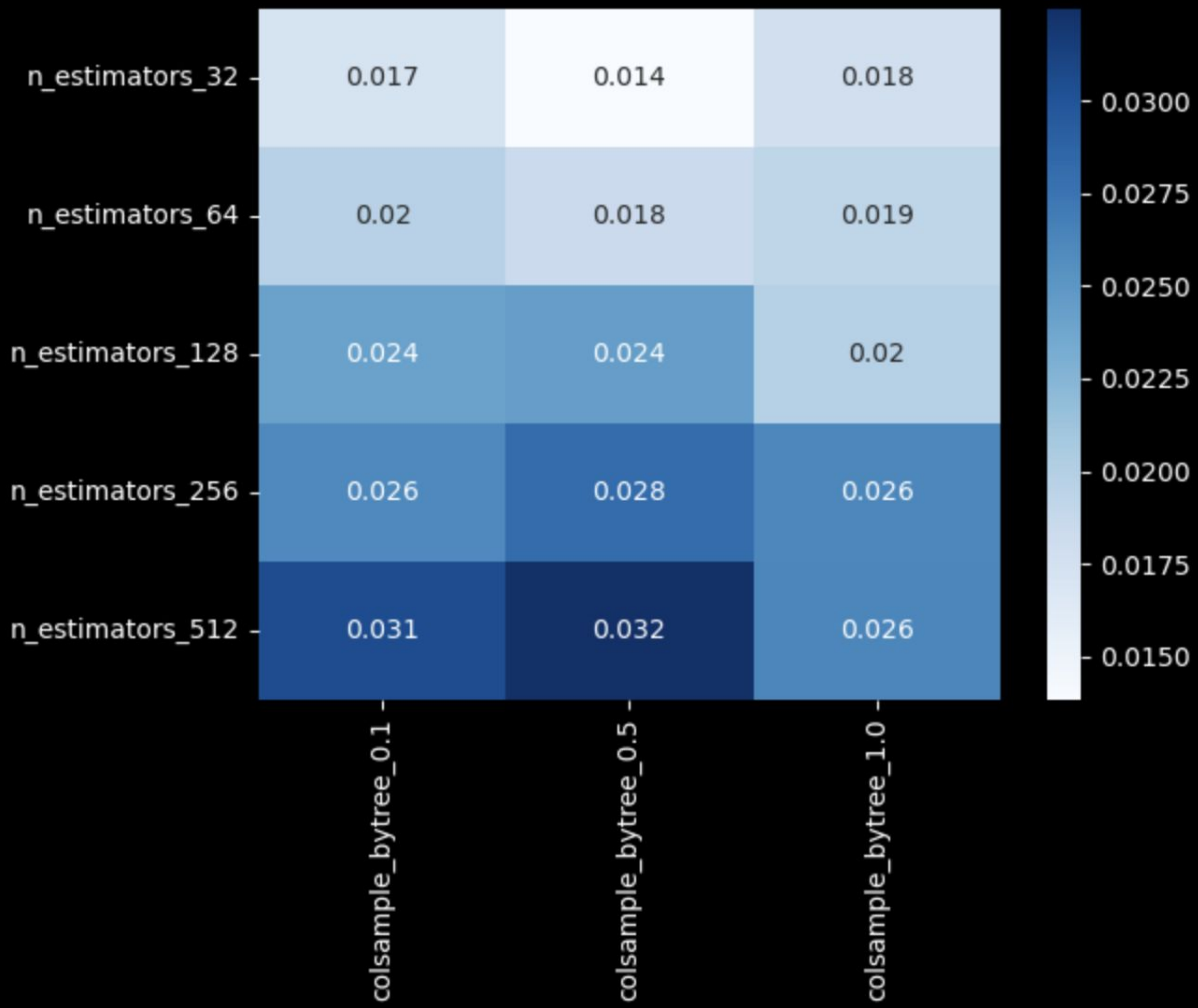
```python
def fit_predict_model(model, train_df, test_df, features):
    model.fit(train_df[features], train_df["target"])
    return model.predict(test_df[features])
```

```python
n_estimators_options = [32, 64, 128, 256, 512]
colsample_bytree_options = [0.1, 0.5, 1.0]

model_names = []
for n_estimators in n_estimators_options:
    for colsample_bytree in colsample_bytree_options:
        # name the model
        # build the model
        # run fit_predict_model, save predictions
```

```python
n_estimators_options = [32, 64, 128, 256, 512]
colsample_bytree_options = [0.1, 0.5, 1.0]

model_names = []
for n_estimators in n_estimators_options:
    # learning rate should scale inversely to number of estimators
    learning_rate = 100/n_estimators * 0.1
    for colsample_bytree in colsample_bytree_options:
        # name the model
        model_name = f"{n_estimators}_{colsample_bytree}"
        model_names.append(model_name)

        # build the model
        model_params = {"n_estimators": n_estimators,
                        "learning_rate": learning_rate,
                       "colsample_bytree": colsample_bytree}
        model = lgbm.LGBMRegressor(**model_params)

        # run fit_predict_model and save predictions
        test_df[model_name] = fit_predict_model(model, train_df, test_df, keep_feature_cols)
```



	correlation mean
512_0.5	0.032469
512_0.1	0.030536
256_0.5	0.028139
512_1.0	0.026251
256_1.0	0.026151
256_0.1	0.026026
128_0.5	0.024453
128_0.1	0.024078
128_1.0	0.019889
64_0.1	0.019742
64_1.0	0.019258
64_0.5	0.018482
32_1.0	0.017843
32_0.1	0.017289
32_0.5	0.013854


n_estimators_32
n_estimators_64
n_estimators_128
n_estimators_256
n_estimators_512
0.017
0.014
0.018
0.02
0.018
0.019
0.024
0.024
0.02
0.026
0.028
0.026
0.031
0.032
0.026
colsample_bytree_0.1
colsample_bytree_0.5
colsample_bytree_1.0
0.0300
0.0275
0.0250
0.0225
0.0200
0.0175
0.0150

Next steps for this

- Submit your best model
- Stake to earn
- Automate your submissions
- Improve your model
  - Use all features, no downsampling
  - Try much bigger models or different types of models
  - Do some clever feature selection

Numerai Signals

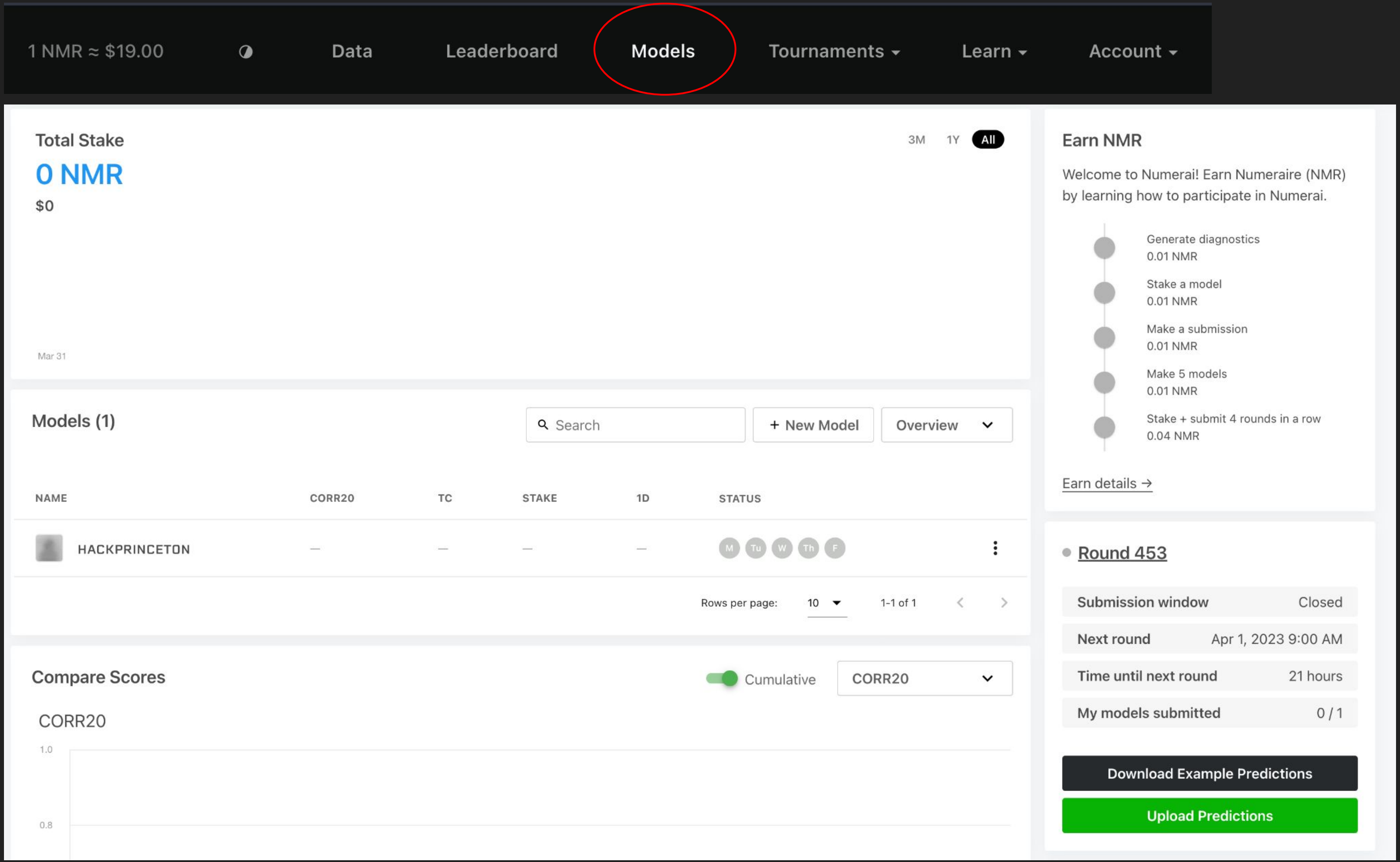
Claim your stake credits

- @SmoothMikeP
- Michael@numer.ai
- Come talk to me in the next couple of hours
- P.S. We are hiring for research positions

Appendix

Advanced

- Synthetic training data?
- Unique high performance model?
- Model Timing - which model to use when?


1 NMR ≈ $19.00
Data
Leaderboard
Models
Tournaments
Learn
Account
Total Stake
0 NMR
$0
3M
1Y
All
Mar 31
Models (1)
Search
+ New Model
Overview
NAME
CORR20
TC
STAKE
1D
STATUS
HACKPRINCETON
M Tu W Th F
Rows per page: 10
1-1 of 1
Compare Scores
Cumulative
CORR20
CORR20
1.0
0.8
Earn NMR
Welcome to Numerai! Earn Numeraire (NMR) by learning how to participate in Numerai.
Generate diagnostics
0.01 NMR
Stake a model
0.01 NMR
Make a submission
0.01 NMR
Make 5 models
0.01 NMR
Stake + submit 4 rounds in a row
0.04 NMR
Earn details →
Round 453
Submission window
Closed
Next round
Apr 1, 2023 9:00 AM
Time until next round
21 hours
My models submitted
0 / 1
Download Example Predictions
Upload Predictions