I have separated my script in two files, one to download the data and one to analyze the data. The download took a decent amount of memory, challenging the 32GB RAM I have. Also these days the Numerai server seems at times to have some problems handling requests, causing errors as well.

If you want to receive the data from the download script, feel free to send me a message and I will send it to you.

Code could also be a bit cleaner, depending on interest x complaints I can refactor it a bit.

Download.ipynb:

```
from multiprocessing.pool import Pool from numerapi import NumerAPI import pandas as pd import json
```

```
from tqdm import tqdm
```

```
from itertools import product from pathlib import Path
```

```
napi = NumerAPI()
```

```
START_ROUND = 285 END_ROUND = 304
```

```
round_query = """ query($roundNumber: Int!, $tournament: Int!) { RoundDetails(roundNumber: $roundNumber, tournament: $tournament) { openTime, roundResolveTime } } """
```

```
def download_round_info(round_number): data = napi.raw_query(round_query, {'roundNumber': round_number, 'tournament': 8})
```

```
data = data['data']['RoundDetails']
data['roundNumber'] = round_number
return data
```

```
def download_rounds_info(start_round, end_round): data = [download_round_info(round_number) for round_number in range(start_round, end_round + 1)] df = pd.DataFrame(data) df.to_csv("round_info.csv", index=False)
```

```
download_rounds_info(START_ROUND, END_ROUND)
```

```
def get_all_users(start_round, end_round): users = []
```

```
for round_number in tqdm(range(start_round, end_round + 1)):
    round_details = napi.round_details(round_number)
    users_round = pd.DataFrame(round_details).username.to_list()
    users.extend(users_round)

users = set(users)

return users
```

```
unique_users = get_all_users(START_ROUND, END_ROUND)
```

```
user_query = """ query($username: String!) { v2UserProfile(username: $username) { dailySubmissionPerformances { changeRequestActualAmount, changeRequestAmount, changeRequestType, corrMultiplier, corrPercentile, correlation, correlationWithMetamodel, date, fnc, fncPercentile, leaderboardBonus, mmc, mmcMultiplier, mmcPercentile, payoutPending, payoutPendingDelta, payoutSettled, roundNumber, roundOpenTime, roundPayoutFactor, roundResolveTime, roundResolved, selectedStakeValue, tc, tcPercentile, tournamentName, weekPayoutSelection } } } """
```

```
def download_username(username): data = napi.raw_query(user_query, {'username': username})
```

```
json_file_name = f'data/{username}.json'
with open(json_file_name, 'w') as f:
    json.dump(data, f)
```

```
with Pool(32) as p: p.map(download_username, unique_users)
```

```
data_path = Path('data')
```

```
all_performances = []
```

```
for file in tqdm(data_path.iterdir()):
```

```
username = file.name.split('.')[0]

try:
    with open(file, 'r') as f:
        data = json.load(f)
except:
    print(f'file not found {file}')
```

```
try:
    daily_performances = data['data']['v2UserProfile']['dailySubmissionPerformances']
except:
    continue

daily_performances_df = pd.DataFrame(daily_performances)
daily_performances_df = daily_performances_df[daily_performances_df.roundNumber > START_ROUND]
daily_performances_df['username'] = username

all_performances.append(daily_performances_df)

all_performances = pd.concat(all_performances) all_performances.to_csv('all_performances.csv', index=False)
```