I ran all of the benchmark model predictions through my metrics and here are the results. I used numerai-tools v0.0.11 with the updated MMC calculation. teager60 had the best MMC but teager_plus_cyrus had the best CORR.

[

benchmarks

571×766 60.4 KB

](https://forum.numer.ai/uploads/default/original/2X/0/01e2469567d1feaf7677dd0e641cd03ba7ecf49b.png)

Here is the code I used. Comments appreciated!

""" benchmarks.py calculate metrics for benchmarks """

import pandas as pd import matplotlib.pyplot as plt import numpy as np from time import sleep from tools import printst from scoring import correlation_contribution, numerai_corr, filter_sort_index

printst('initialize') data_round = '624' scoring_target = 'target_cyrus_v4_20'

benchmarks = pd.read_parquet(f'd:/rain/{data_round}_v4_2_validation_benchmark_models.parquet') model_names = [m for m in benchmarks if m != 'era'] validation = pd.read_parquet(f'd:/rain/{data_round}_v4_2_validation_int8.parquet') del validation['data_type']

benchmarks, validation = filter_sort_index(benchmarks, validation) # remove NaNs

printst('read meta model data') metamodel = pd.read_parquet(f'd:/rain/{data_round}_v4_2_meta_model.parquet', columns= ['era', 'numerai_meta_model']) eras = metamodel['era'].unique() # eras to keep val_data = validation[validation['era'].isin(eras)]

for model_name in model_names: printst(f'{model_name}: calculate statistics') predictions = pd.DataFrame(validation.index.values, columns=['id']) validation['prediction'] = benchmarks[model_name] predictions.set_index('id', inplace=True) predictions['prediction'] = benchmarks[model_name] predictions['era'] = validation['era'].values results = pd.read_csv('benchmarks/benchmarks.csv', index_col='model_name')

```
printst(f'{model_name}: compute metrics')
per_era_corr = (validation.groupby('era')
                .apply(lambda x: numerai_corr(pd.DataFrame(x['prediction']),
                                              x[scoring_target])))
corr_values = per_era_corr['prediction'].values
corr = np.mean(corr_values)
std = np.std(corr_values)
sharpe = corr / std
consistency = np.sum([c >= 0.01 for c in corr_values]) / len(corr_values)
predictions = predictions[predictions['era'].isin(eras)]  # select eras in metamodel data
mmc = correlation_contribution(predictions,
                               metamodel['numerai_meta_model'],
                               val_data[scoring_target])['prediction']
title = ('corr: {0:8.6f}  mmc: {1:8.6f}  std: {2:8.6f}  sharpe: {3:8.6f}  consistency: {4:8.6f}'
         .format(corr, mmc, std, sharpe, consistency))
printst(f'{model_name}: {title}\n')

# update validation results
results.at[model_name, 'corr'] = corr
results.at[model_name, 'mmc'] = mmc
results.at[model_name, 'std'] = std
results.at[model_name, 'sharpe'] = sharpe
results.at[model_name, 'consistency'] = consistency

# Plot the per-era corr

per_era_corr.plot(kind='bar', title=f'Validation Correlation for {model_name}\n{title}',
                  figsize=(12, 6), xticks=[], snap=False)
plt.savefig(f'benchmarks/{model_name}_validation_corr.png')
plt.close()

printst(f'{model_name}: save results')
saved = False
bs = '\b' * 80
while not saved:
    try:
        results.to_csv('benchmarks/benchmarks.csv')
        saved = True
    except:
        print(bs + '*** Close benchmarks/benchmarks.csv to save new results ***', end='', flush=True)
        sleep(3)
print(bs, end='', flush=True)
```