CHALLENGE DATA ENS 2019 - 3 SOLUTIONS
CFM + NAPOLEON + NEURONS

L. DEBORDE – JAN. 2020

Methods

- Feature Engineering
- (mostly) Gradient Boosting
- (a bit of) Parameter Tuning

- I had more success on time series challenges

Tools

- Python (Jupyter)
- Pandas
- Scikit-learn
- LightGBM -> very fast & handle well categorical features
- Personal Computer (Windows, local small CPU)

Notebooks for the submissions are available here: https://github.com/ljmdeb

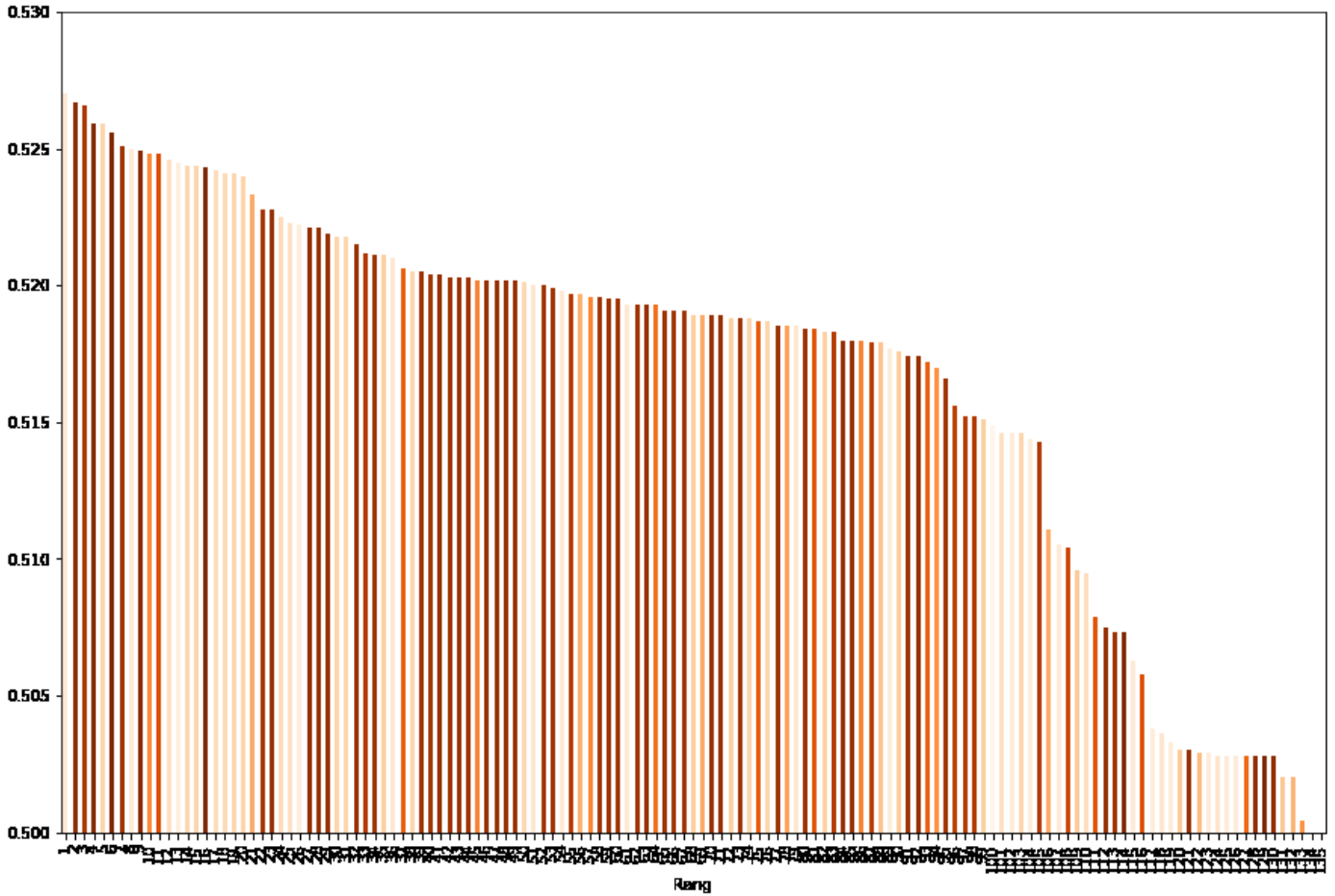
INSIGHT.DATA.CLARITY.

Prediction of daily stock movements on the US market

The goal of this challenge is to predict the sign of the returns (= price change over some time interval) at the end of about 700 days for about 700 stocks.

Ranked 2nd* with final score 0,5267

Ranked 2nd on the open leaderboard with the same entry during most of S2, score > 0,53



- CFM's was the most competitive of the 2019 « Challenge data » challenges*
- Competition lasted until year end (above, brown colors signify late submissions – 6 of the 10 best ranking final submissions where entered from november on)

*not counting deleted users

SAMPLES PERTAINING TO THE SAME EQUITY SHOULD BE CORRELATED
SAMPLES PERTAINING TO THE SAME DAY SHOULD BE CORRELATED

Equities : same equities in train and test

- We can use the same equities in train and validation
- Ligthgbm : just add Equity identifier as a categorical feature (converted to 0-N integers, as should always be done with Lightgbm)
- Without Lightgbm : need to build « Equity » features :
  - Must use only training values in the model
  - Possible Idea : use average of features for each equity, computed only on training set
  - Better idea :
    - compute average of sign(y) for each equity on the training set
    - Rational : tendency of a stock to overperform
    - Add this as a new sample feature -> **very powerful predictor (0,5%)**
    - Much better results than using one hot encoding of Equity Code
    - LightGBM is even more efficient however : Feature dropped after introducing Lightgbm

Days : different days in train and test

- Days must be different in validation and training : build train-valid-split and cross validation based on days
- How to build « Day » features ?
  - Compute Average (and Standard Deviation) of features on each day
  - Add those averages (and Std) as new features
  - Build also « equity normalised » feature (i.e. feature value minus average on day divided by std on day)

The Problem :

- We were asked to predict the sign of stock variations during the last half hour of a day, given an number of examples. It's a **binary classification problem** of supervised learning.
- For the examples, we were actualy given the exact amount of the last half hour variation (not only its sign). So we could have approached the problem as a regression problem (trying to predict the exact end day stock movement, then using the predicted variation to observe its sign). It's easily understood why this approach is literaly far-fetch (trying to do much to obtain less), hence not promizing. So we did stick to classification here.

The model :

- I did compute the following « simple » features for each sample (equity x day):
  - For each row, **Number of NaN** (NaN are filled by zeros after counting), **Number of Zero** variations (before imputing zeros for NaN)
  - For each row, **Sum of variations for the whole day**, **Sum of variations during last 30 m**inutes and **during penultimate 30 minutes**, **Last variation** (at 15:20)
  - For each row, Intraday **Standard deviation**, **Skewness** & **Kurtosis** for that day x equity
  - **Exponential moving average** length 6 for that day x equity , **Relative Strength Index** (sum of positive variation H divided by difference between H and sum of negative variations B)
- Then I computed the average of those features for each date, as well as Standard deviations of the features by dates, and normalised features (value minus average, divided by std).
- Keep also number of equity for each date.
- Add an equity feature : tendency of an equity to overperform at end-day
  - average of sign(y) on training set for this equity
  - **high eplanatory power (but superceded by proper use of categorical features in Lightgbm)**
- Tune Lgbm by cross validation (8 random split of date range) **(useful to avoid chance fit)**
- Try dropping each feature -> drop when cross validation score improve without feature -> a number of similarly performing results. Best with 6 features (see those in my notebook on github)
- Unsuccessfull attempts:
  - Taking into account (sectorial) correlation between equities (tried ACP data by date or global basis, no improvment on scores)
  - Stacking with Linear models

Didn't attempted:

- to recognize stocks based on external Stoxx500 data (it would have been cheating)
- Regression

How to improve?

- In real life : use exogeneous sructured (descriptive, financial, economic) or unstructured (news) data on company / sector / market /economy level

NOTA : A VERY SIMPLE « EXPERT »RULE BEATS THE BENCHMARK

Simple Expert Rule

- « if Market is higher at 15:20 than at opening, it will fall in the last 30 minutes ; if market is lower at 15:20 than at opening, it will rise in the last 30 minutes »
- Rules proves to predict correctly the sign of returns in more than 52% of cases (open test set)

July 23, 2019, 7:31 p.m.	Expert rule	Intraday Profit taking	0.520815

- Rational : intraday traders cut their positions just before market closes

Benchmark

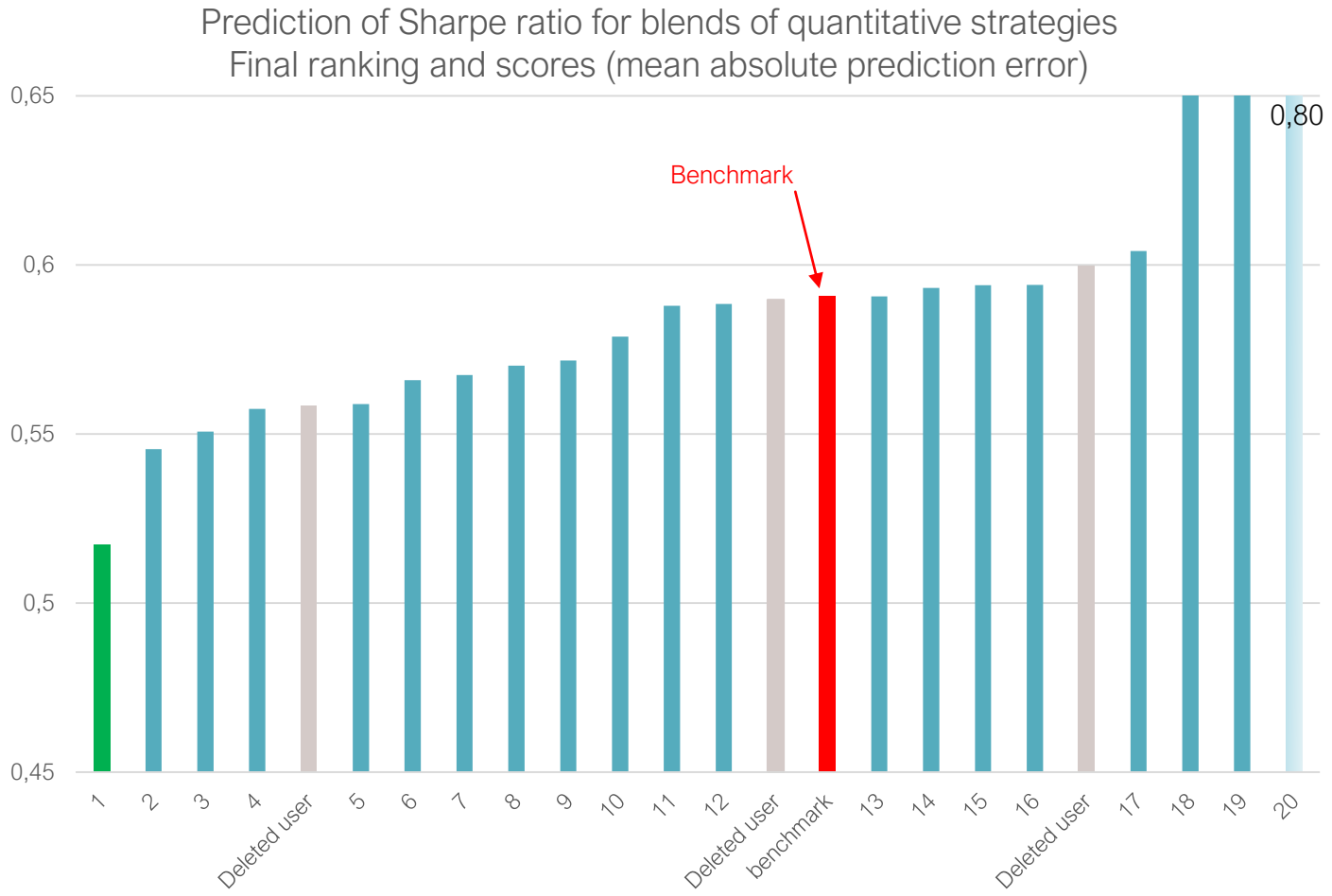
- "*The benchmark is a LightGBM boosted trees model, with the following [non default] parameters: objective: None, subsample_freq: 1, learning_rate: 0.05, n_estimators: 500, colsample_bytree: 0.8, subsample: 0.9*
- *This correctly predicts the sign of the returns in typically 51.8 % of the cases*."

52% w. simple rule, 53% w. complicated features and advanced algorithms -> meager reward for compexity
Complex approach is probably more rewarding with more data (financial variables, dates…)

Prediction of Sharpe ratio for blends of quantitative strategies
by Napoleon X

Ranked 1st - final score 0,517

(a bit of a lucky shot : given their score on the open leaderboard, my other late – more planned- submissions would probably have scored around 0,53, still ahead of #2)

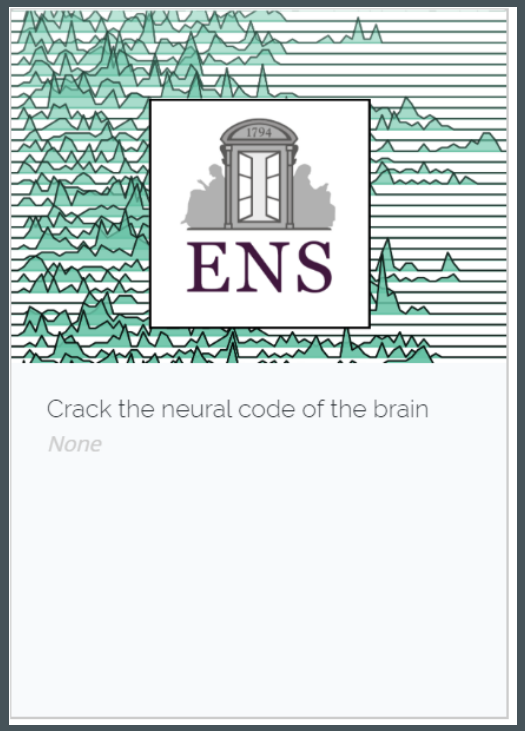


- Few (20*) competitors : the very technical statement probably deterred people with no finance or math background from participating
- Moderate improvement between benchmark and best scores

*not counting deleted users

MIND THE ARROW OF TIME !

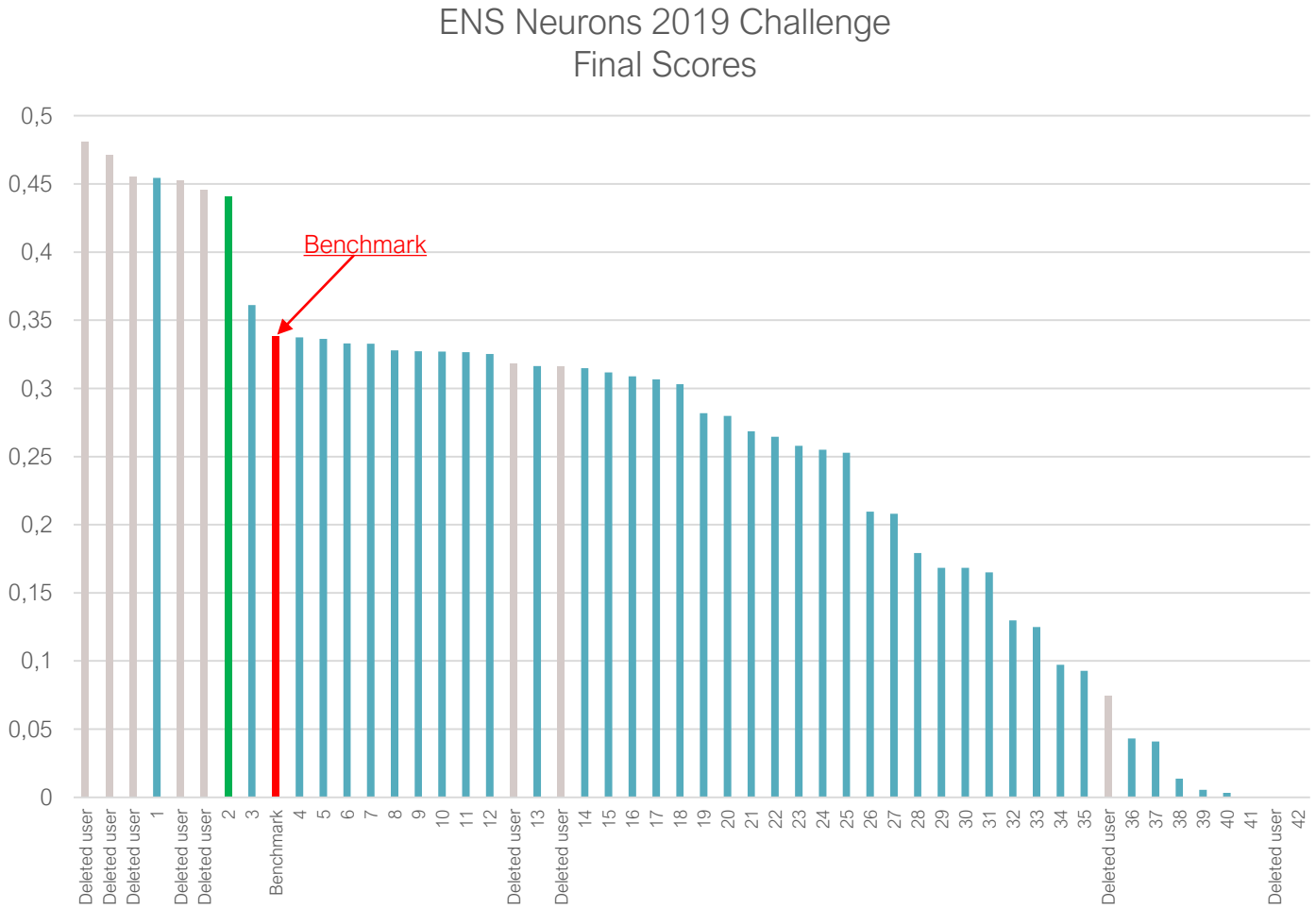
- The problem :
  - We were to predict Sharpe ratio (real value), given a set of example : it's a regression problem
- Optimisation :
  - to build a model on train, convert given target with the function $f(x) = sign(x) \times e^{-\frac{1}{abs(x)}}$
  - Then just optimise on l1 (absolute value of differences)
- Mind the date !
  - As told in the statement, a given time serie appear in up to 50 different lines (with different strategy weights w_i)
  - As NOT told in the statement (but as easily verified), time series are different in training set and test set
  - Data has to be split between different dates for training and validation or the model will be unable to forecast out of time (splitting between train and validation without taking care of that lead to spuriously good-looking models, with low performance on test)
- Feature used :
  - proxy for a sharpe ratio computed with the statement formula
  - Global performances on the whole time serie
  - Various usual time series features (taken from my CFM submission)
- An oddity :
  - I obtained the best score on test with a wide margin (> 2,5%) with a model that had scored much less on validation
  - I choose to select for the competition a less well performing model (still quite ahead of competitors on visible test, but with a lower margin(~1%), but with homogeneous performance in validation and test -> « select » feature on the Challenge site didn't function !



Ranked 2nd * - final score 0,44

*not counting deleted users

1st on the public leaderboard at S1 end



- Difficult challenge : few participants above benchmark
- Many high-ranking deleted users : same person ? Insiders ?

THE PROBLEM

- We are to predict a binary value : it's obviously a binary classification problem

DIFFERENT KINDS OF NEURONS

- Only one neuron is present in both training and test set -> Impossible to learn neurone-specific features -> to introduce neuron ID as a (categorical) feature would be a mistake
- But scientific litterature tell us there's (at least) 2 different kind of neurons regarding impulse responses (my thanks to Tristax for informing me of this very useful fact) -> Enabling the model to recognize those different kind of neurons would be highly useful. How to do that ?
  - Unsuccessfuly tried unsupervised approach (KNN) to identify neuron classes
  - Fallback plan : building alternate neuron features (by averaging feature values on all raw with the same neuron)
- Important notice : as neurons are different in train and test, one has to split by neurons when doing crossvalidation, if one expect to learn neuron features! Failure to recognize this necessity may be the reason why so many participants get bad results

MODEL :

- First we differentiate on each raw (as suggested in the benchmark)
- Then on each raw we compute (with numpy) : min, max, sum, standard deviation, skew, kurtosis, 20-quantiles.
- Feature importance analysis on a random forest built this way lead us to select 20-quantiles 1,5,10,14,17 (besides min, max, sum, std, kurtosis, for a total of 11 features).
- Besides, as explained earlier, we want to build neuron-specific feature. To this end, for each neuron, we compute average of the 11 features for each neurons
- … for a total of 22 features.
- The random forest was soon replaced by the better performing (sklearn) Gradient boosting.
- This model secured a 1st place on the public rank at end of June.

- Under pressure from late competition, we later (hastily) :
- Replaced the sklearn Gradient Boosting by the better performing and much quicker Lightgbm (handy for optimization)
- Added « normalized » features, i.e. row feature values divided by their average on the neuron (idea taken from our participation in the CFM challenge), for a total of 33 features

- Albeit those hasty additions had very significantly improved our submissions, we have probably indulged along the way in some measure of guilty glances at test results (aka overfitting), as our best entry loses 3% between public and final score, whereas other good competitors typicaly lose only 1%
- a good lesson learned : better to look for a new idea than to overoptimize parameters and feature selection !