Challenge Data 2019

Prediction of daily equity stock movements (CFM)

L. DEBORDE – Nov. 2020

Contexte

- 1ère participation à des challenges
- Formation initiale (ancienne) d'ingénieur
- Professionnel de la gestion d'actifs
- Récente formation Data CES data scientist (TPT)

Méthodes

- pas de NN !
- Random Forest, **Gradient Boosting**
- Feature Engineering (plutôt manuel)
- Parameter Tuning

Outils

- Python (Jupyter)
- Pandas
- Scikit-learn
- LightGBM (very fast)
- Personal Computer (Windows, local CPU)

Notebooks for my submissions on 3 challenges are available at https://github.com/ljmdeb

CFM

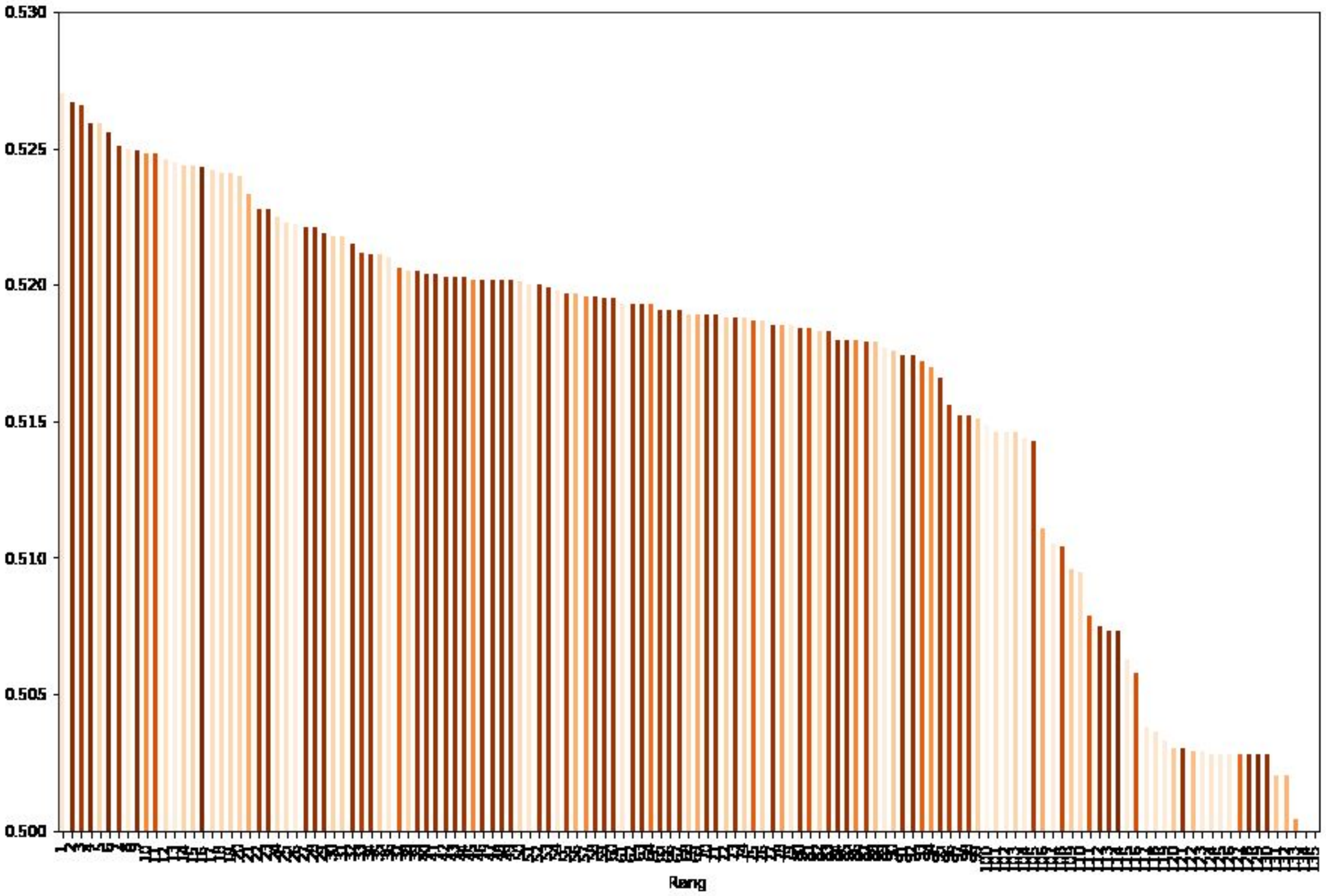
INSIGHT.DATA.CLARITY.

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.

I ranked 2nd* with final score 0,5267

(light overfit)



- CFM's was the most competitive of the 2019 ENS challenges (by number of participants*)
- 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

In [1]:
```python
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
```

In [2]:
```python
dbo=pd.read_csv("output_training_hTMcdrS.csv")
```

In [3]:
```python
dbo.head()
```

Out[3]:

	ID	end_of_day_return
0	0	0.964025
1	1	-0.583286
2	2	2.191065
3	3	0.406176
4	4	-0.053764

In [5]:
```python
dbi=pd.read_csv("input_training_CFM.csv")
```

In [6]:
```python
dbi.head()
```

Out[6]:

	ID	eqt_code	date	09:30:00	09:35:00	09:40:00	09:45:00	09:50:00	09:55:00	10:00:00	...	14:35:00	14:40:00	14:45:00	14:50:00	14:55:
0	0	975514820	2059277756	0.378421	0.062977	-1.699955	-0.253532	0.317230	-1.394081	-1.338014	...	-0.258339	-0.193950	0.129398	0.064666	0.3232
1	1	975514820	661006643	-0.588441	-0.393182	0.918810	0.392393	0.195901	-0.195754	0.000000	...	0.000000	-0.260940	0.000000	-0.130601	0.1306
2	2	975514820	2252275018	-0.974472	1.789703	-0.105310	-0.456042	0.894337	-0.341168	-0.373679	...	-0.331103	-0.236124	-0.256959	-0.229774	0.0639
3	3	975514820	901241465	-4.612982	1.746131	1.092367	-0.030626	0.996415	-1.391209	0.169799	...	0.336335	-0.185523	0.521719	-0.095356	-0.2079
4	4	975514820	1213815424	-2.642804	-0.772708	-0.115318	0.145272	-0.136105	0.450894	-0.046230	...	0.631221	0.090185	-0.169416	-0.081212	-0.2751

5 rows × 74 columns

The Problem :

- Predict the sign of stock variations during the last half hour of a day, given their moves during the rest of the day.
- The actual variation during the last half hour was given on a number of examples (training set)
- It’s a **binary classification problem** of supervised learning.

In [1]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
```

In [2]:
```
dbo=pd.read_csv("output_training_hTMcdrS.csv")
```

In [3]:
```
dbo.head()
```

Out[3]:

	ID	end_of_day_return
0	0	0.964025
1	1	-0.583286
2	2	2.191065
3	3	0.406176
4	4	-0.053764

In [5]:
```
dbi=pd.read_csv("input_training_CFM.csv")
```

In [6]:
```
dbi.head()
```

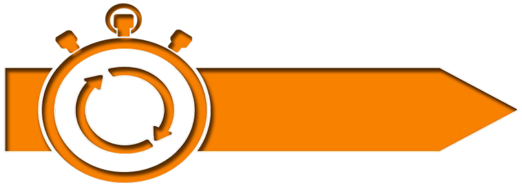
Out[6]:

	ID	eqt_code	date	09:30:00	09:35:00	09:40:00	09:45:00	09:50:00	09:55:00	10:00:00	...	14:35:00	14:40:00	14:45:00	14:50:00	14:55:
0	0	975514820	2059277756	0.378421	0.062977	-1.699955	-0.253532	0.317230	-1.394081	-1.338014	...	-0.258339	-0.193950	0.129398	0.064666	0.3232
1	1	975514820	661006643	-0.588441	-0.393182	0.918810	0.392393	0.195901	-0.195754	0.000000	...	0.000000	-0.260940	0.000000	-0.130601	0.1306
2	2	975514820	2252275018	-0.974472	1.789703	-0.105310	-0.456042	0.894337	-0.341168	-0.373679	...	-0.331103	-0.236124	-0.256959	-0.229774	0.0639
3	3	975514820	901241465	-4.612982	1.746131	1.092367	-0.030626	0.996415	-1.391209	0.169799	...	0.336335	-0.185523	0.521719	-0.095356	-0.2079
4	4	975514820	1213815424	-2.642804	-0.772708	-0.115318	0.145272	-0.136105	0.450894	-0.046230	...	0.631221	0.090185	-0.169416	-0.081212	-0.2751

5 rows × 74 columns

- Please note that we were 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 of day stock movement, then using the predicted variation to observe its sign). It’s easily understood why this approach is literally far-fetch (trying to do much to obtain less), hence not promising.

Mind the time arrow !



- each Eqt_code appears many times
- each Date appears many times
- each couple (Eqt_code x Date) appears only once
- Eqt_code are the same in the training set and the test set (except for one)
- **Dates are different in the training set and the test set !**
  - we are to predict… the future
  - don’t use the same dates in the training and validation set, or your model will (only) excel at predicting… the present

```
# pour se mettre dans les conditions du test, nous séparons les dates entre training et validation
ndate=dbi.date.unique()
```

```
for random_state in [7,42,210,666]:
    #tirons au hasard les dates retenues
    n_train, n_val =train_test_split(ndate,random_state=random_state)
    .........
```

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

Stocks : same stocks in train and test

- We can use the same stocks 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
  - Other Possibility : Embedding

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)

N.B. : A VERY SIMPLE « EXPERT »RULE BEATS THE BENCHMARK

CFM 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."*

Simple Expert Rule : intraday profit taking

- « **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** »
- This rule predicts correctly the sign of returns in more than 52% of cases (open test set)
- Rational : intraday traders close their trades just before market closes
- Shows importance of the total variation feature

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

The model

- A NaN Strategy
- A number of « simple » features for each sample (equity x day)
- Then average / Standard deviation / normalised feature for each date
- Kept also number of equity for each date.
- Before LGBM : added an equity feature : tendency of an equity to overperform at end-day (useless with LGBM)
- With LGBM : treated Eqt_code as a categorical feature
- Tuned Lgbm parameters
- Then reducing the number of features (a bit of overfitting here)

Unsuccessful attempts

- Taking into account (sectorial) correlation between equities
- Automatic feature building (TSFresh)
- Stacking with Linear models
- Expert rules

Didn’t attempted

- to recognize stocks based on external Stoxx500 data (it would have been cheating)
- RNN (lack of skill at the time)

How to improve?

- Reduce overfitting : average the predictions of a number of good models
- In real life : use exogenous / fundamental data

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).
- Kept also number of equity for each date.
- Added an equity feature : tendency of an equity to overperform at end-day
  - average of sign(y) on training set for this equity
  - **high explanatory power (but superseded by proper use of categorical features in Lightgbm)**
- Tuned 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 -> I probably should have taken the average prediction of a number of good models rather than only the best one
- Unsuccessfull attempts:
  - Taking into account (sectorial) correlation between equities (tried ACP data by date or global basis, no improvement on scores)
  - Stacking with Linear models

Didn't attempted:

- to recognize stocks based on external Stoxx500 data (it would have been cheating)
- Regression (not promising)
- RNN (lack of skill at the time)

How to improve?

- Reduce overfitting : average the predictions of a number of good models (rather than only taking the best one)
- In real life : use exogenous data, structured (descriptive, financial, economic) or unstructured (news), on company / sector / market /economy level

FEATURES IMPORTANCE

- 1) Sum (and variations taking into account daily average, i.e. market index)
- 2) Equity code
- 3) Volatility (Standard deviation of variations)
- 4) relative number of zeros variations (not counting NaN)
- 5) number of stocks with data this day (correction to other features)