IEEE Investment Ranking Challenge

Kirill Romanov
Saint-Petersburg
Russia
kirill.v.romanov@gmail.com

IEEE Investment Ranking Challenge. Publication date: 17 May 2018.

Abstract— This document describes my approach and methodology during IEEE Investment Ranking Challenge competition. I built a rigorous pipeline to test and validate different models and as a result at the end of stage 1 I get about 0.15 of spearman score on public leaderboard (0.17 on internal validation set). During the stage 2 I got the spearman correlation on internal validation set as much as more than 0.2

Keywords— data science, competition

Introduction

The main goal of the competition was to develop a model that will help identify the best-performing stocks in each time-period using the provided data sets of financial predictors and semi-annual returns. Two performance metrics were used: (1) Spearman correlation and (2) Normalized Discounted Cumulative Gain of Top 20%. The second metric estimate the quality of forecasting for top 20% the most performant stocks. Total period for data was 1996-2017. The participants should predict the performance from 2002 to first half of 2017. The total score was:

$$\text{Final score} = (A+B+C+D)/4 \qquad (1)$$

Where:

A = Spearman correlation on holdout data from 2002 – 2016
B = NDCG score on holdout data from 2002 – 2016
C = Spearman correlation on holdout data from 2017
D = NDCG score on holdout data from 2017

To deal with this challenge I built the four sets of linear models using the classical data science pipeline (see picture 1). My approach was to try different kinds of datasets and select the best model for each forecast period based on validation data. I tried different kinds of models (linear, tree, neural nets, knn) and found out that classical linear models with l2 regularization (ridge model) is the best choice. Due to this hereinafter, if not noted explicitly, I will describe my experiments with Ridge models. The sections of this documents correspond the stages of my pipeline (Data preparation, Modelling, Stacking).

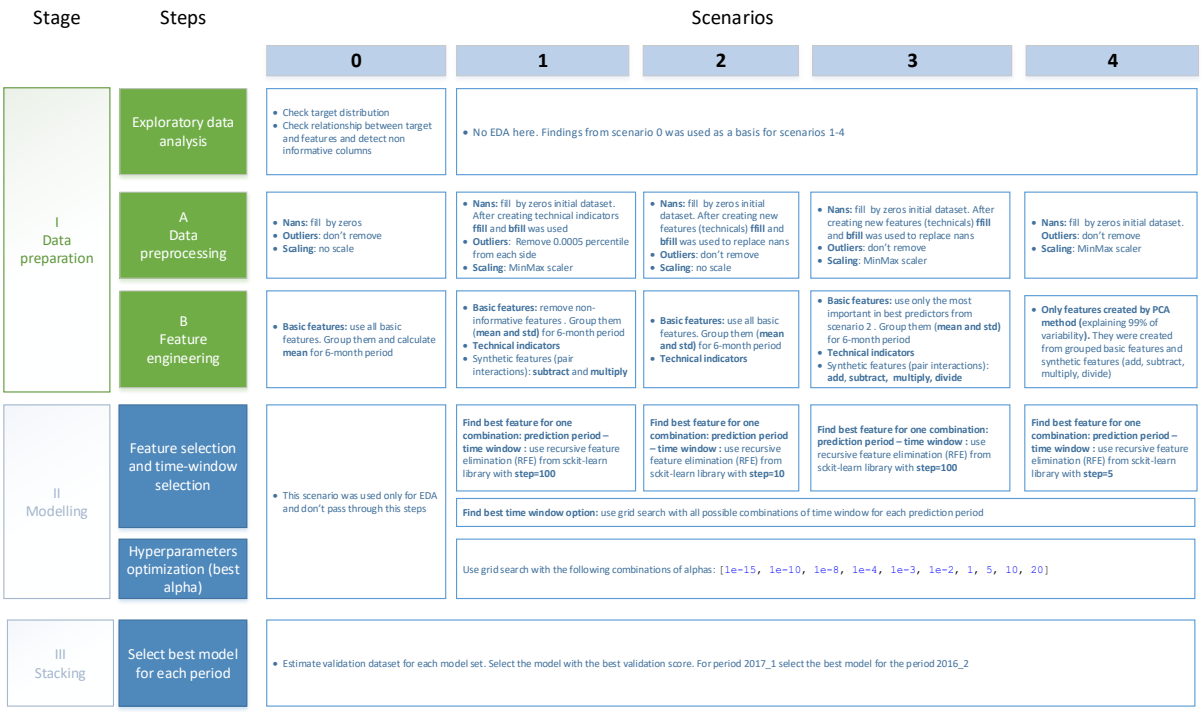

Fig. 1. Pipeline of different scenarios

I. Data preparation

A. Exploratory data analysis

Below I describe the main findings related with provided data:

- **All indicators are normalized and anonymized**. So our opportunities to do some fundamental analysis or use subject matter expertise are limited and we should rely more on technical and statistical methods.
- **All stock names are encoded and in each period different ids are used.** This condition is quite challenging because we lost a lot of useful information related with the dynamic of stocks.
- **All indicators have values for each month inside of half-year period**. However due to the fact that it is impossible to make a fair link of each stock between the periods (see previous note), this information is not very useful and can be aggregated
- **For period time 2002 – 2016 we have data for train and holdout data for prediction**. It imposes additional constraint because in model training process and during feature engineering we have to pay attention to don't see in the future.
- Data contain missing values and outliers
- Distribution of target values in training dataset is close to normal and don't need to be transformed for further modeling:

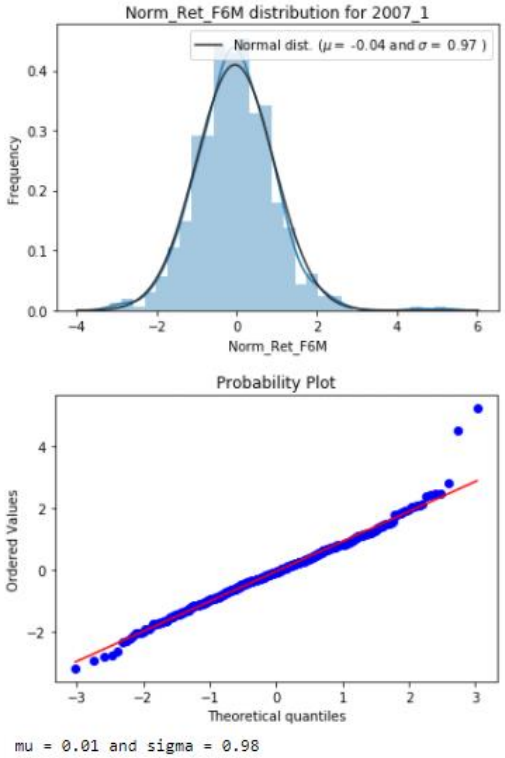

Fig. 2. Example of distribution of values for target in first half of 2008

B. Data preprocessing

As a result of data analysis, the following initial data preprocessing was made:

- Replace nan values by zero
- Aggregate the indicators by taking mean values for six month (as it was proposed in starter kit).
- Additionally calculate standard deviation for each indicators and add this to new dataset
- Create dataset without outliers and dataset with outliers. The methodology for excluding outliers is the following: for each indicator identify rows where the values are outside of 0.0005 percentile from each side. Then take an aggregated list and remove the set of these rows from dataset.
- After the preprocessing some indicators looks like noise (see figure below) these columns were identified by visual analysis and removed from dataset without outliers

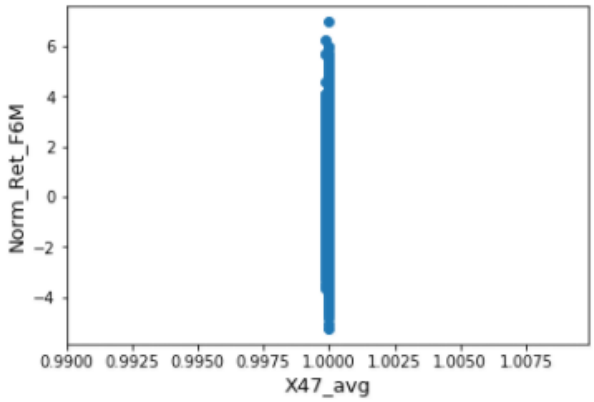

Fig. 3. Example of non-informative indicator

- Finally, after feature engineering stage, all features were scaled by sklearn MinMaxScaler

Feature engineering

There were three big groups of features that were created: (1) Technical indicator based on target value from previous period (don't have a big impact on final solution). The methodology was the following:

- Calculate the average value of normalized return for each period on training data.
- Shift these values for one period ahead and use them as a feature of next period. By this step we eliminate the effect of looking in the future (e.g in the period 2008_1 we have predictions of normalized return for period 2008_2 and cannot use these predictions in period 2008_1. But in period 2008_2 we already known these returns because it current period's return. Thus it's possible to use these information in period 2008_2).
- Calculate technical indicators for this features. After the indicators were created, fill nans by ffil and bfill methods. Below the description of these indicators:

TABLE I. Technical indicators created as a new features

Indicator	***Parameter***
Moving Average	Period size 2,3
Exponential Moving Average	
Momentum	
Rate of Change	
MACD, MACD Signal and MACD difference	n_fast =2, n_slow=3
KST Oscillator	r1, r2, r3, r4 = 1, 2, 3, 4 n1, n2, n3, n4 = 1, 2, 3, 4
True Strength Index	r=2, s=2
Coppock Curve	Period size 2,3

Indicator	***Parameter***
Standard Deviation	

(2) new features generated from the basic ones. The algorithm was the following:

- Take averaged indicators (ends with "_avg" mask)
- Do the following pseudocode:

Algorithm 1. Algorithm to create synthetic features from feature pairs (on the example of dataset 2)

```
for indicator2 in indicators_list:
        new_indicator1 = indicator1 - indicator2
        new_indicator2 = indicator1 * indicator2
```

As a result at the end of feature engineering we had the following datasets:

- **Dataset 1**: No outliers, manually excluded noise basic indicators, basic indicators (mean and std), technical indicators, generated features from basic mean indicators. Min-max scaling
- **Dataset 2**: Basic indicators (mean and std), technical indicators. Min-max scaling
- **Dataset 3**. Was created on the base of **Dataset 2**. This dataset had the following features:
    - The most frequent basic indicators that occurred in the best models of Dataset 2
    - New indicators derived from averaged basic indicators by the following pseudocode:

Algorithm 2. Algorithm to create synthetic features from feature pairs for Dataset 3

```
for indicator1 in indicators_list:
    for indicator2 in indicators_list:
        new_indicator1 = indicator1 - indicator2
        new_indicator2 = indicator1 +  indicator2
        new_indicator3 = indicator1 /  indicator2
        new_indicator4 = indicator1 *  indicator2
```

- **Dataset 4**. Contain only generated by PCA method features. The base for PCA was the following scaled features:
    - All basic indicators (mean)
    - Synthetic features generated from all basic mean indicators. I used algorithm 2 to generate them

II. Modelling

A. Model selection

For model selection I used the following approach:

- Use the whole dataset because different models demonstrated different performance on the same period and it was hard to estimate rough performance on small dataset size.
- Model selection was made on the base of Dataset1 and Dataset 2, because Dataset 3 had about 4000 features and it was technically impossible to calculate all models on this dataset
- "Second chance" for bad models with basic parameters. If model demonstrate bad results try to calculate this models with other parameters. For tree model I use hyperopt [1], for linear models – manual tuning.

The models that were tested but weren't selected:

- **Random forest.** Was proposed in starter kit. Demonstrate average spearman about 0.03 and was excluded from further experiments
- **KNN regressor from sklearn –** sklearn algorithm is slow and demonstrate very low performance (about 0). Was excluded from further experiments
- **Kernel Ridge from sklearn** (alpha=0.6, kernel='polynomial', degree=2, coef0=2.5). Was slow and demonstrate almost the same performance as a much faster basic Ridge
- **Elastic Net from sklearn –** performance about zero with the different parameters.
- **Gradient Boosting regressor from sklearn –** extremely slow, average performance (with 500 estimators)
- **Keras models (LSTM and simple feedforward networks).** Instable results between public leaderboard and internal validation. This class of models could provide high results but I found better and more stable models)
- **SVR from sklearn** slow and didn't demonstrate good performance
- **Catboost regressor [2].** Slow and results worse than lightgbm. This model works great when dataset contains a lot of categorical features because it can great encode them. However this dataset doesn't contain it and maybe due to this the results weren't good.
- **LGBM regressor.** Was quite fast and I got the good and consistent results during the internal validation and public leaderboard (hereinafter PLB). However comparing with Ridge model was worse from the point of view of speed and results.

The final model that I analyze deeply in the next stages:

- **Ridge from sklearn –** extremely fast and best result during internal validation and PLB

B. Model optimization (feature selection, time window selection and hyperparameters optimization). Ridge

At the first step I fixed random state for reproducibility. Then the algorithm was the following:

- Fix start training period equals to 1996_2
- Make recursive feature selection for the model with fixed parameters. I use RFECV library from sklearn for this [3]. If features number were more than 1000, step was 100 (dataset 1, 3). If the number of features was relatively small, I used step 10 (dataset 2) or step =5 (dataset 4)
- Select the best features for the prediction period

- Shift start training period and repeat up the algorithm up to the situation when start training period equals prediction period -1 (it means that we fit the model on one period and try to predict the next one)
- As a result we have the model for each prediction period with the features and time windows that demonstrate best score on validation dataset
- At final step I made grid search for finding the best alpha for the ridge models that demonstrate the best results for each prediction period

III. Ensembling/Stacking

- For each prediction period I selected the model with the highest spearman score on validation dataset.
- For the period 2017_1 I selected the parameters of the best model for the period 2016_2

Conclusion

During the competition was developed a rigorous pipeline for selecting, testing and evaluating the models. As a result this approach led to top 5 solution during the round 1. During the round 2 I fine-tuned it up to 0.2+spearman correlation on internal validation dataset.

References

[1] https://github.com/hyperopt/hyperopt
[2] https://github.com/catboost/catboost
[3] http://scikit-learn.org/stable/modules/generated/sklearn.feature_selection.RFECV.html