NYU Mathematics in Finance final project

Cross Sectional Alpha Prediction

Xin Cui, Qian Shen, Xiang Yao

December 22, 2017

1 Abstract

Searching through large set of fundamental, technical and mixed features to forecast future returns is proved to be a very difficult task due to the low S/N ratio. To achieve this, we need the efficient computing power and regression techniques. For the hardware, we are using GPU, cloud computing, own implemented python libraries and several APIs to accelerate the algorithm. For the algorithm, among many machine learning methodologies, we find two best models: First: Linear Regression (after applying 7 feature engineering methodologies); Second: Extra tree Regression Model. In the end, we use Meta Regression Model to mix the Linear Regression and Extree Model. From this research, we learned that GPU is not a silver bullet, especially on the sparse noisy financial data; Feature engineering and combining model can significantly improve the prediction power of the model.

2 Literature Review

Cross-sectional regression has been widely used in the quantitative equity strategy to either forecast asset returns or explain the risk of the portfolio. The one that is most well-known is the Barra Factor Mode. Nearly all equity quantitative hedge fund is using Barra factor model (Barra Factor Model,

2017) to explain the daily asset return. The factors are classified into three different types: Market Factor, Industry Factors and Style Factors. The most interesting and research intensive ones are the Style Factors (Value, Size, Liquidity, Beta, Risk, etc.). Many style factors are transformed from several financial data points. The methodology of the combination involves feature engineering using either domain knowledge of the fundamental reasoning or the quantitative approaches. The research objective for Barra, similar to this research, is also trying to improve the explanatory power of the regression. Throughout its development, this cross-sectional regression model has evolved from the US to global with better feature engineering of the datasets.

Many computational intensive financial calculations are relying on large infrastructure for paralleled computing. The industry is starting to apply these extra powers to the financial dataset in Lin and Wehkamp and Kanniainen (2017). In the Quantitative Trading section, it is mentioned that the market is mature enough that the competition for alpha becomes the competition of the size of dataset and computing power. Using the cloud computing framework, there are two major benefit. First, it is the speed of the computation. The computation and backtesting works that normally takes overnight or days now takes less than hours and even minutes. Secondly, the cloud computing framework is more API friendly and can adapt to more existing and the latest development tools. The research on large data set is no longer limited by the size of the memory in the local development computer.

From the Heaton (2017), we see that the academic community are already applying feature engineering to not only to the linear regression but also empirical testing on the effect on the machine learning algorithms. The feature engineering, like simple mathematics transformation, can have large effect on the regression and machine learning algorithm. It has the effect of transforming the shape of distribution, de-noising the data set and rotating the data. The feature engineering itself is widely applied in the science, ex. Medical for various test on human being. The common feature engineering types are: Logarithms and Power Function, Differences and Ratio, Counts, Polynomials, Ratio Differences and Polynomials, The Quadratic Equation. From this research, we see that different feature engineering types are best suited for different machine learning algorithms.

In this research, we have reviewed many machine learning approached to apply to the alpha prediction model. Random Forest (Leo, Adele, 2017) is an algorithm that uses the randomly drawn trees votes for the final classification. The algorithm runs very efficient in large dataset and are able to show which features are more important than others. Boosting Algorithm (Buhlmann, Hothorn, 2007) is an algorithm that applies the votes from the

individual classification. It is used as a way to further improve the classification ability by combining different classification together. Extra Tree (Geurts, Ernst, Wehenkel, 2005) or Extremely randomized trees is another variation of random forest algorithm. It further improves the random forest algorithm by building completely randomized trees. It could further improve the accuracy of the algorithm. Meta Regression (Wolpert, 1992) is a method for forming linear combinations of different predictors to give improved prediction accuracy. Similar to combining orthogonal strategies improve the sharpe ratio, diversified model can potentially improve the predicting power of individual models.

From the literature reviews, we see that the financial data is very noisy data set. In order to extract signal from this noisy dataset, we will apply the following techniques in this research. 1. Build Pipeline to have the most efficient use of the cloud computing API 2. Take Benefits from the Google Cloud for its speed and computing power 3. Feature Engineering to apply to the linear model. 4. Meta Regression. This is the most interesting aspect of this research, not only did we use feature engineering, at the end, we combine two best performing models through Meta Regression. This new direction can be another future direction for Heaton (2017). The research can be further expanding to combining different machine learning approaches with different feature engineering techniques.

3 Introduction

3.1 Project Description

How can we use the world's tools and intelligence to forecast economic outcomes that can never be entirely predictable? This question is at the core of countless economic activities around the world. Economic opportunity depends on the ability to deliver singularly accurate forecasts in a world of uncertainty. By accurately predicting financial movements, we will learn about scientifically-driven approaches to unlocking significant predictive capability.

In this project, we use the data set of Two Sigma Financial Modelling Competition. This dataset contains anonymized features pertaining to a time-varying value for a financial instrument. Each instrument has an id. Time is represented by the 'timestamp' feature and the variable to predict is 'y', which is tomorrow's return. The independent variables are 100 factors, which have been classified as derived, fundamental and technical. Fundamental signals are locally constant and technical signals change more frequently. One difficulty

is that fundamental signals are hard to use as it usually updates quarterly.

The goal of this project is using Machine Learning skills to predict the label 'y' through these anonymized features, which is also predicting the cross sectional alpha. As for the evaluation of models, we use R value between the predicted and actual values. The R value similar to the R squared value, also called the coefficient of determination. R squared can be calculated as:

$$R^2 = 1 - \frac{\sum_i (y_i - \hat{y}_i)^2}{\sum_i (y_i - \mu)^2}.$$

To calculate R, we then use:

$$R = sign(R^2)\sqrt{R^2}.$$

where y is the actual value, μ is the mean of the actual values, and $\hat{y}$ is the predicted value. It is common that R values are really low; in finance, given the high ratio of signal-to-noise, even a small R can deliver meaningful value.

In this project, we have tried many Machine Learning skills, and applied 6 models. They are

- Linear Regression (Feature Engineering)
- Random Forest
- Boosting Algorithm
- Extra Tree
- Meta Regression.

We have found the most meaningful way to improve the predicted R square is through feature selection and engineering technique in the linear model. Our predicted R reaches 2%, and it is the largest improvement comparing to the plain regression and other methodologies. The second-best algorithm is Extra Tree regression. With Linear Regression and Extra Tree regression demonstrating different predictive characteristics, we are using the Meta Regression to combine Linear Regression and Meta Regression to come up with the best predictive result.

3.2 Hardware

3.2.1 Cloud

Since in this project, we have tried lots of Machine Learning skill and the data set is large, computation power really maters. So we use Google Cloud to perform the computation task. Google Cloud is a convenient platform to perform big data analysis. Some tools like 'BigQuery' are "nearly magical" because of their performance. Queries that used to take hours or days now take minutes or hours. In cloud side, 64 Virtual cores and 400GB memory are used.

3.2.2 Local

For our hardware setting, in local side, we use Threadripper 1950X, 16 core, 32 Threads, GTX 1080, and 32GB memory.

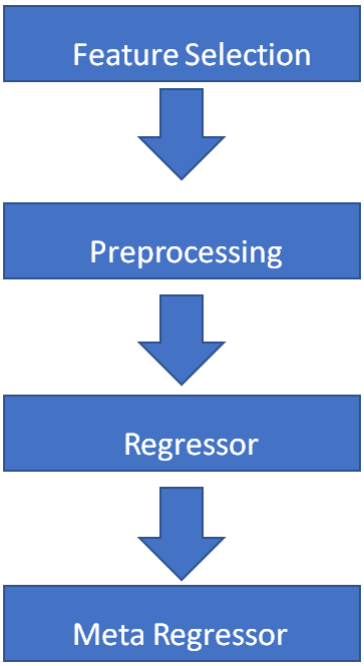
3.3 Kaggle GYM API

The 'kagglegym' API is based on OpenAI's Gym API, a toolkit for developing and comparing reinforcement learning algorithms. You could read OpenAI's Gym API documentation for more details. Note that ours is named 'kagglegym' and not 'gym' to prevent possible conflicts with OpenAI's 'gym' library.

First, we need to create an "environment". This will be our primary interface to the API. The kagglegym API has the concept of a default environment name for a competition, so just calling make() will create the appropriate one for the project. Each observation also has a 'features' dataframe which contains features for the timestamp you'll be asked to predict in the next 'step'. Each step takes the predicted y of next timestamp as input, and returns four things: 'observation', 'reward', 'done', and 'info'. The 'done' variable tells us if finished the training or not. The 'info' variable is just a dictionary used for debugging. Perhaps most interesting is the 'reward' variable. This tells you how well you're doing. The goal in reinforcement contexts is that you want to maximize the reward.

There two main benefits of using Kaggle Gym API. Firstly, we can avoid look-ahead bias. When making prediction at one timestamp, we will not look at information of later timestamps. Secondly, the APT is convenient for developing and comparing reinforcement learning algorithms.

3.4 Pipeline



There are standard workflows in applied machine learning. Standard because they overcome common problems like data leakage in your test harness. Python scikit-learn provides a Pipeline utility to help automate machine learning workflows. Pipelines work by allowing for a linear sequence of data transforms to be chained together culminating in a modeling process that can be evaluated. The goal is to ensure that all of the steps in the pipeline are constrained to the data available for the evaluation, such as the training dataset or each fold of the cross validation procedure.

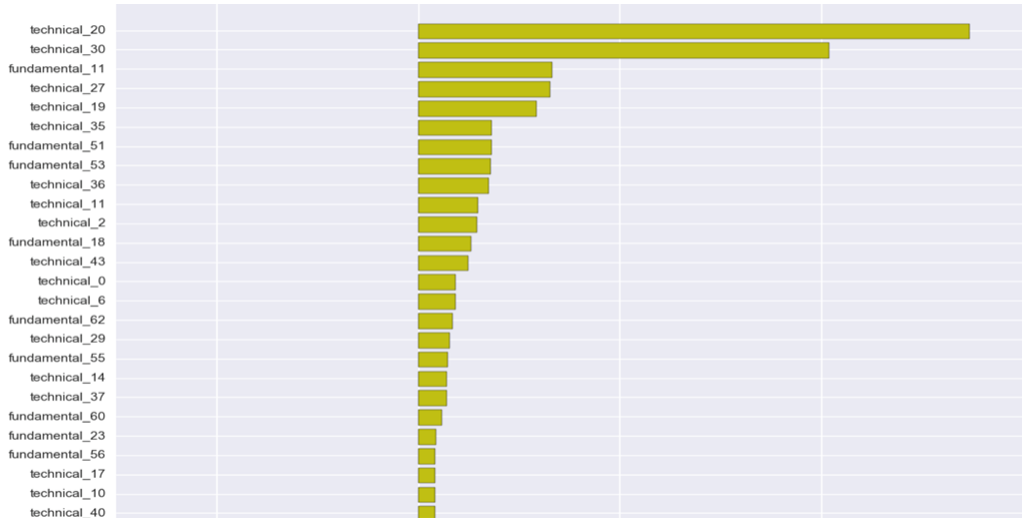
In this project, there have two main advantages for using pipeline to organize our codes. The first advantage is more easy to modify our model. Each part in a pipeline is like a LEGO block. Therefor when you have a new idea, just replace with the new block. The second advantage is also avoiding look-ahead basis. For example, if you fit transform (fillna/winsorize/minmax scaling) for the full DataFrame at the beginning, it implies you know the future properties which introduce a look ahead bias. With pipeline the winsorize threshold only depends on the train set.

4 Experiment Methodology and Analysis

4.1 Feature Engineering

Feature engineering is the process of using domain knowledge of the data to create features that make machine learning algorithms work. Feature engineering is fundamental to the application of machine learning, and is both difficult and expensive. The need for manual feature engineering can be obviated by automated feature learning.

We start from basic ordinary least square linear regression. We regress y on the 100 fundamental, technical, and derived features. The plain linear regression has a very low R value, basically less than 1%. But we can pick strong features through plain linear regression. We can compare the R value reduction if we remove a specific feature from the regressors. The following figure the R value reduction for top 27 factors. Based on the result, we can conclude that technical-20 and technical-30 are two strongest factors.



Plain linear regression is not enough to predict y. We have to some feature engineering based on the features we already have. We do the following feature engineering:

- k lags (k = 1, 2, 3, 4, 5, 6)
- Difference between current observation and the previous observation
- Absolute value of difference

- Exponential Weighted Moving Average (EMWA) along time of a feature
- DeEMWA of a feature
- Average (among IDs) level of a feature
- DeAverage.

After feature engineering, including the original features, we have 1300 features in total.

4.2 Model 1: Linear Regression

The first model we try is linear regression. But including all 1300 features as regressors in linear regression is time-consuming. Therefore, we select 110 features which hive highest correlation with 'y' series as regressors in our linear regression model.

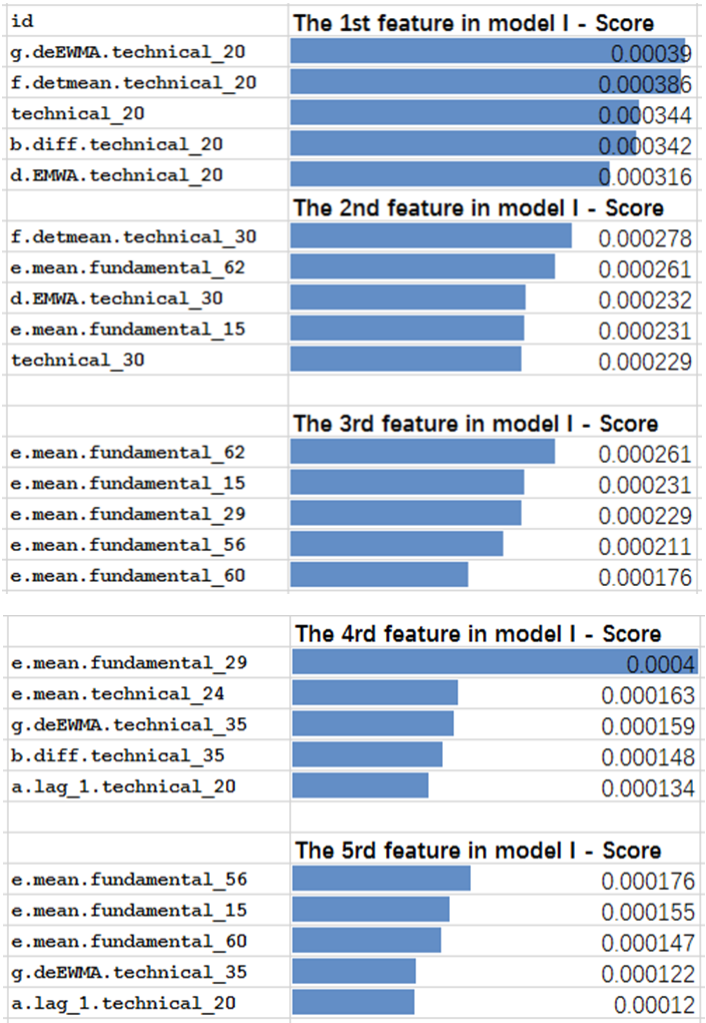
In order to found out most influential 5 features, we apply forward selection procedure. At each step, each variable that is not already in the model is tested for inclusion in the model. The most significant of these variables is added to the model. The top 5 influential features we found are,

- deEWMA feature of technical_20, which explained 0.039% R^2
- demean (fluctuation) feature of technical_30, 0.0278% R^2
- mean (trend) feature of fundamental_62, 0.0261% R^2
- mean (trend) feature of fundamental_29, 0.04% R^2
- mean (trend) feature of fundamental_56, 0.0176% R^2

We noted that the top 2 influential features are derived features of technical_20 and technical_30, which is consistent with the result we get from plain linear regression.

Feature	deEWMA.technical_20	detmean.technical_30	mean.fundamental_62	mean.fundamental_29	mean.fundamental_56
Explained R^2	0.000390	0.000278	0.000261	0.000400	0.000176

The 5 figures are results in our forward selection procedure. It is worthwhile to note that in the first regression of our procedure, beside the deEWMA.technical_20, the strongest 4 features in the remaining variables are deEWMA.technical_20, detmean.technical_20, technical_20, diff_technical_20, and EMWA.technical_20. But, it is interesting we find out, in our second regression, none of the above 4 features show up, which means there exist strong correlation between the above 4 features with deEWMA.technical_20. Hence, it is very meaningful to perform forward selection procedure to avoid the correlation between regressors.

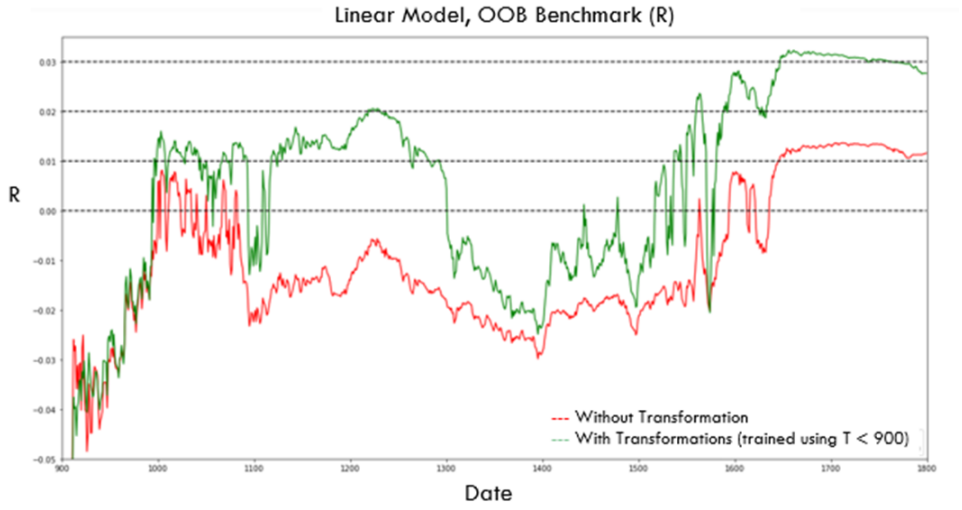


Now, we want to see how feature engineering transformation will affect the out-of-bag R value.

Model 1: Original features without transformation. we select original features technical_20, technical_30, and fundamental_62 which have been proved to have strong prediction power in our forward selection procedure.

Model 2: Original features and features derived from feature engineering transformation. technical_30, technical_20, detmean.technical_30, deEWMA. technical_20, mean.fundamental_62.

The results of out-of-bag R value of Model 1 and Model 2 are as follow:



From the the figure above, we can see that linear regression with figure engineering transformation show great prediction power. The out-of-bag R value reach 3%. Comparing the plain linear regression without any transformation, it is a big improvement.

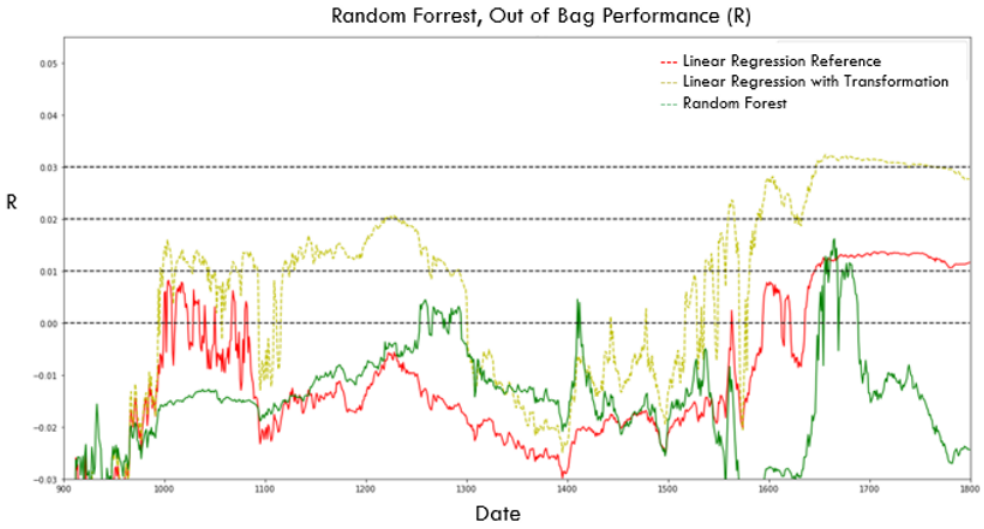
4.3 Model 2: Random Forest

In linear regression model we mention before, we have explored the linear relationship between features and dependent variable y. To further improve the prediction power of our model, we want to look at the nonlinear relationship between features and y through tree model. The first tree model we try is Random Forest. Random Forest constructs a multitude of decision trees at training time and outputting the mean prediction (regression) of the individual trees. Random decision forests correct for decision trees' habit of overfitting to their training set.

A random forest is a meta estimator that fits a number of regression decision trees on various sub-samples of the dataset and use averaging to improve

the predictive accuracy and control over-fitting. The sub-sample size is always the same as the original input sample size but the samples are drawn with replacement.

In our Random Forest model, we applied GridSearchCV to pick the best hyperparameter using cross validation. Also we notice that shrink feature pool and reduce the tree depth may increase oob performance in this dataset.



From the above figure, we can see that actually, the performance of Random Forrest model is not so good. Basically, the performance of Random Forrest is even weaker than plain linear regression

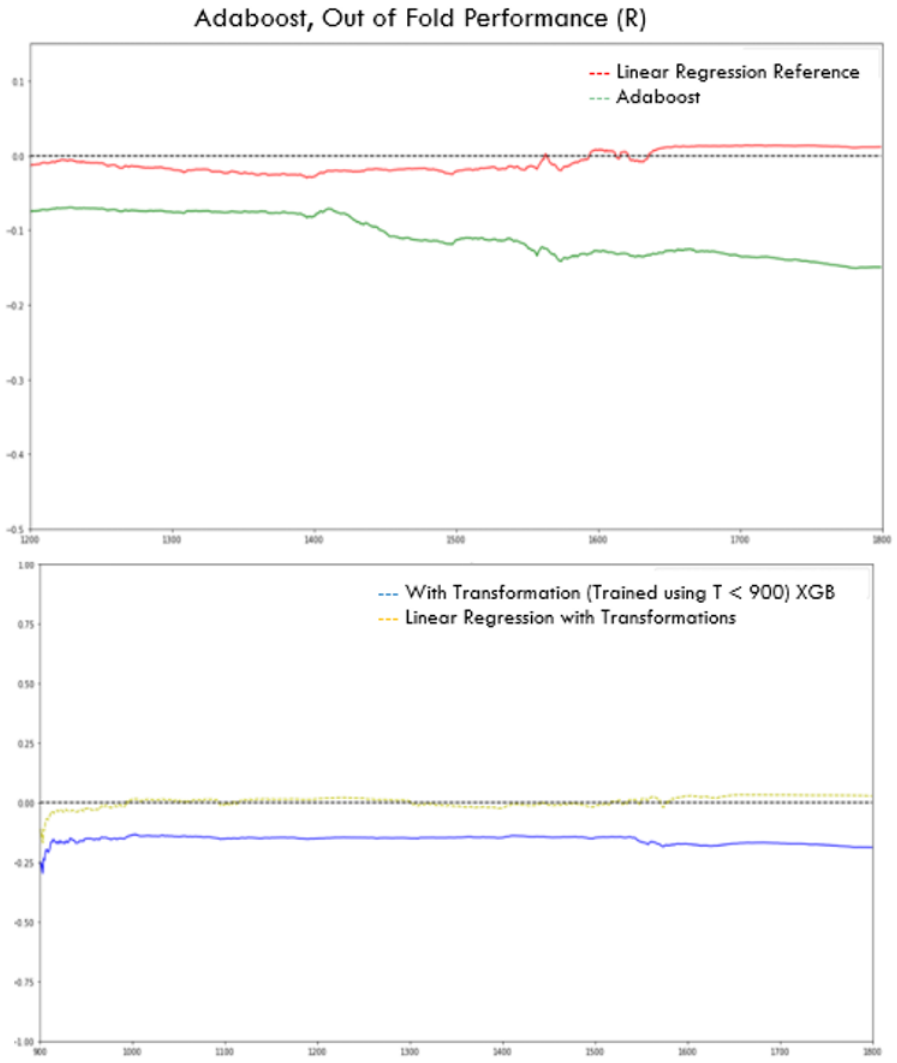
4.4 Model 3: Boosting Algorithm

The second tree model we try is boosting algorithm, especially Adaboost. An AdaBoost regressor that begins by fitting a regressor on the original dataset and then fits additional copies of the regressor on the same dataset but where the weights of instances are adjusted according to the error of the current prediction. As such, subsequent regressors focus more on difficult cases.

The core principle of AdaBoost is to fit a sequence of weak learners (i.e., models that are only slightly better than random guessing, such as small decision trees) on repeatedly modified versions of the data. The predictions from all of them are then combined through a weighted majority vote (or sum) to produce the final prediction.

We found 12 features has high feature importance. They are deEWMA. technical_20, detmean.technical_20, technical_20, diff.technical_20, EMWA.tech nical_20, detmean.technical_30, e.mean.fundamental_62, EMWA.technical_30, technical_30, e.mean.fundamental_29, mean.fundamental_15. And we decide to remove the bottom 0.01% and top 0.01% outliers by winsorizing based on cross validation.

The performance of our Adaboost model is as following. The result is a miserable and much worse than plain linear regression and linear regression with feature engineering transformation. The R value are negative in both training set and test set. Actually, it is expected since the financial time series are so noisy.

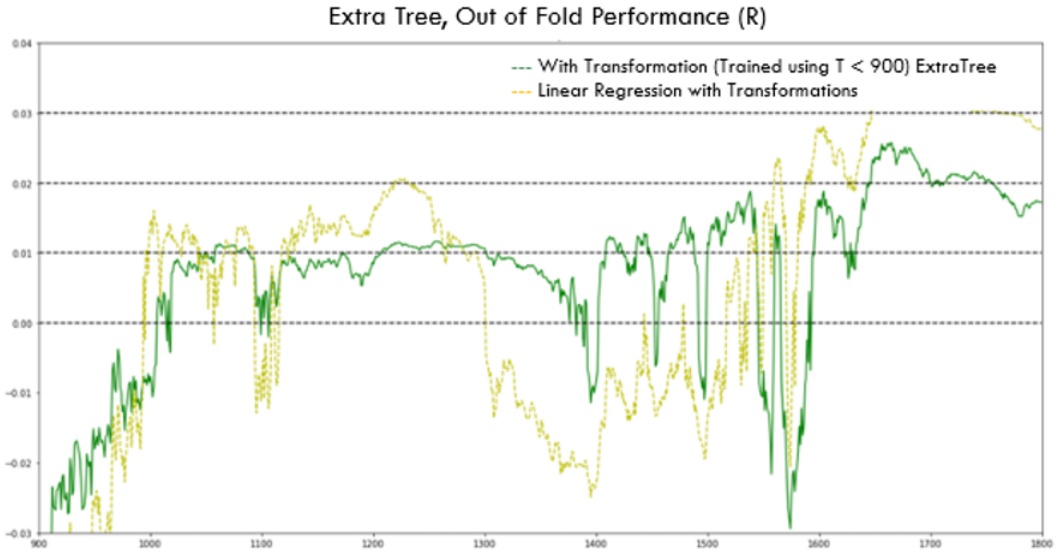


4.5 Model 4: Extra Tree Regression

The next tree model we try is Extra Tree regression. In Extra Tree model, randomness goes one step further in the way splits are computed. As in random forests, a random subset of candidate features is used, but instead of looking for the most discriminative thresholds, thresholds are drawn at random for each candidate feature and the best of these randomly-generated thresholds is picked as the splitting rule. This usually allows to reduce the variance of the model a bit more, at the expense of a slightly greater increase in bias.

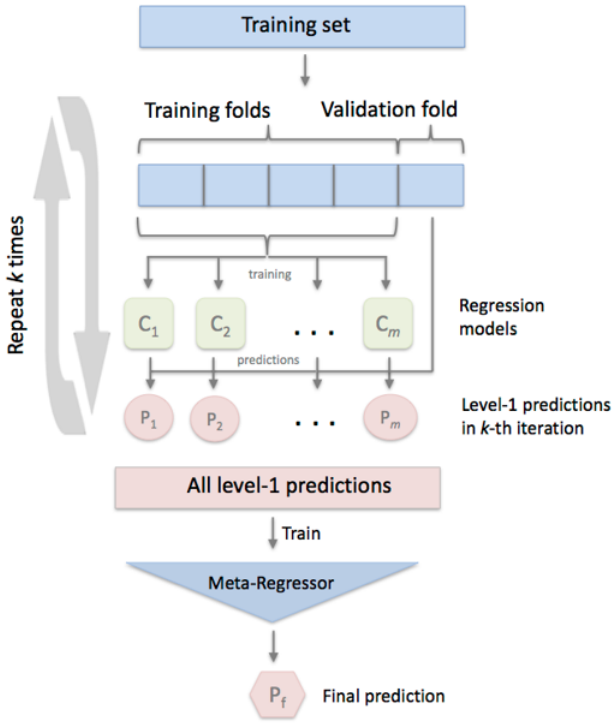
We set there are 30 trees in the forest. Basically, the number of tree is the larger the better, but also the longer it will take to compute. We also set the size of the random subsets of features to consider when splitting a node is 4. The lower the greater the reduction of variance, but also the greater the increase in bias.

The performance of Extra Tree model is as follow. Compare to linear regression with feature engineering transformation, Extra Tree performs not bad. Actually, Extra Tree is also the best among the tree model we have tried. The R value achieves 2%. What is more, in timestamp 1300 – 1500 it performs much better than the linear regression, however for other timestamps it is beated by the linear model.



4.6 Model 5: Meta Regression

In order to capture both the linear and nonlinear relationship between dependent variables y and features. We want to used a Meta Regression to combine linear model (Elastic Net) with the Extra Tree model.



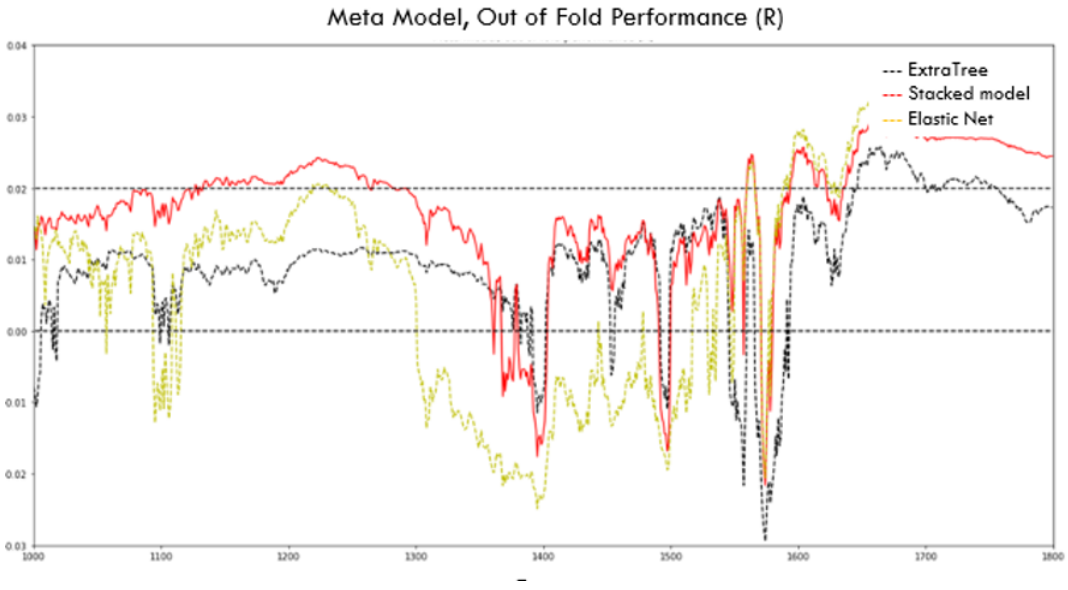
Meta regression data flow by mlxtend [9]

Meta regressions or stacking regression in this case, is a method for forming linear combinations of different predictors to give improved prediction accuracy. The idea is to use cross-validation data and least squares under non negativity constraints to determine the coefficients in the combination. Its effectiveness is demonstrated in stacking regression trees of different sizes and in a simulation stacking linear subset and ridge regressions. The idea of stacking originated with (Wolpert 1992).

We use the StackingRegressor API in python to conduct this meta regression model. Stacking regression is an ensemble learning technique to combine multiple regression models via a meta-regressor. The individual regression models are trained based on the complete training set; then, the meta regres-

sor is fitted based on the outputs – meta-features –of the individual regression models in the ensemble.

In our model, we stack Elastic Net and Extra Tree we trained above together, and fit them by a ridge regression to construct a new regressor. The following graph shows the performance of the new meta regressor. The out-of-bag performance of meta regressor is really good and quite stable. Even though, the prediction power is weaker in timestamp range 1600-1800 than linear regression with transformation, but it is stronger in timestamp range 1300-1500, which makes meta regression is a stable model.



4.7 Model Analysis and Comparison of the Result

We observe that the learn model have a a relative high overall prediction power $(OOB, R,\ 3\%)$ however during timestamp 1300-1600, it suffers a long bad prediction priod. The extra tree method captures the nonlinear dependency, it has more consistent performance however the overall prediction power is relative low. $(OOB, R,\ 1.7\%)$. We applied meta regression scheme to mix these two models and get a stacked model with good overall prediction power $(OOB, R,\ 2.5\%)$ and consistent OOB performance.

For other models like random forrest, boosting algorithm, genetic programming (gplearn), they all surfers from the poor oob performance in this dataset. It might caused by the low informantion and high noise.

4.8 Conclusion and Future Work

We believe that this research has a lot of potential as a forecasting model for the alpha research. On the hardware side, we are trying to accelerate the algorithm through GPU and cloud computing. Moreover, we compile python libraries to be more efficient. For the algorithm, among many machine learning methodologies, we find two best models: Linear Regression (after applying 7 feature engineering methodologies) and Extree Regression Model. Then, we use Meta Regression Model to mix the Linear Regression and Extree Mode, which gives us the best predicting power.

We are able to draw the following conclusion:

- It is important to optimize the python module for scripts with heavy matrix compuations. Anaconda delivers the numpy with BLAS and LAPACK, however currently the MKL (Math Kernel Library by Intel) could speed up the matrix computation up to 50% more. (Matlab ships MKL by default)

- GPU is not a silver bullet. We compiled xgboost and lightgbm with GPU label on. (It is OFF by default.) We expected a massive speed boost. However the speed boost is less than 20%. The reason might cased by the implementation or the noisy low information dataset.

- To improve the model, feature engineering and combining model can significantly improve the prediction. The hardware could support us a massive GridSearch on hyperparameters. However the increase is roughly neglectable. However using feature engineering, the prediction power increased more than 100% on our dataset.

There are many potential future research areas that we could explore. We are listing some interesting ones below as references. Machines learning and its applications are the most active research area and have a lot of potentials:

- Many financial datasets are too large to fit into the memory. Therefore, we should come up a model stacking package with SGD.

- Time series models can always be formulated using the gym interface. Therefore, we might be able to train a meta-model using reinforcement learning.

- In this research we are only applying feature engineering on the linear approach, and we can further expand the feature engineering to the machine learning approaches and see how machine learning methodology adapt to different feature engineering techniques.

- In this research, we are focusing on forecasting for future return. One interesting research area is to integrate the forecast signals to the portfolio construction and risk model; Then, we are able to run the backtest to see the alpha generating ability of the model and how stable the model through different market environment.

References

[1] Heaton, Jeff, An Empirical Analysis of Feature Engineering for Predictive Modeling (Jan, 2017).

[2] Barra Factor Model E3, http://www.alacra.com/alacra/help/barra_handbook_US.pdf.

[3] Lin, Binghuan and Wehkamp, Rainer and Kanniainen, Juho, Practitioner's Guide on the Use of Cloud Computing in Finance (November 6, 2017).

[4] Heaton, J.B. and Polson, Nick and Witte, Jan Hendrik, Deep Learning for Finance: Deep Portfolios (September 5, 2016).

[5] Kumar, Manish and Thenmozhi, M., Forecasting Stock Index Movement: A Comparison of Support Vector. Machines and Random Forest. Breiman, L. Machine Learning (2001) 45: 5.

[6] Tang, J., S. Alelyani, and H. Liu. "Data Classification: Algorithms and Applications." Data Mining and Knowledge Discovery Series, CRC Press (2015): pp. 498-500.

[7] Wolpert, David H. "Stacked generalization." Neural Networks 5.2 (1992): 241-259.

[8] Breiman, Leo and Cutler, Adele. Random Forests, https://www.stat.berkeley.edu/ breiman/RandomForests /cc_home.htm

[9] https://rasbt.github.io/mlxtend/user_guide/regressor/StackingRegressor/