2. Benchmark Bond Trade Price Challenge

2.1 Overview

Goal:

- The aim of this competition: **To Predict the next price that a US corporate bond might trade at.** Contestants are given information on the bond including current coupon, time to maturity and a reference price computed by Benchmark Solutions. Details of the previous 10 trades are also provided.

Submission Format:

- For each observation, a contestant should provide the expected trade price. In the data section, please see random_forest_sample_submission.csv for an example submission:

```
In [2]: submission_template =
pd.read_csv('random_forest_sample_submission.csv')

In [3]: submission_template
Out[3]:
           id  trade_price
0      762679    98.064530
1      762680   116.611906
2      762681   104.496657
3      762682   106.858986
...       ...          ...
61143  823822   123.218163
61144  823823    85.358722
61145  823824    96.918183

[61146 rows x 2 columns]
```

Evaluation Method:

- Mean Absolute Error *with Weighted Factors(as shown in the dataset + Normalization)*: so as called WEPS(Weighted Error in Price per Sample)

$$WEPS = \frac{\sum_{i=1}^{m} w_i(|y_{true} - y_{predict}|)}{\sum_{i=1}^{m} w_i}$$

2.2 Data

The host mainly provide train.csv&test.csv, and a R to generate the submission file.

- There are 61 features of 762678 stocks .

```
In [33]: train_set.index
Out[33]: RangeIndex(start=0, stop=762678, step=1)

In [34]: train_set.columns
Out[34]:
Index(['id', 'bond_id', 'trade_price', 'weight', 'current_coupon',
       'time_to_maturity', 'is_callable', 'reporting_delay',
'trade_size','trade_type', 'curve_based_price', 'received_time_diff_last1',
       'trade_price_last1', 'trade_size_last1', 'trade_type_last1',
       'curve_based_price_last1', 'received_time_diff_last2',
       'trade_price_last2', 'trade_size_last2', 'trade_type_last2',
       'curve_based_price_last2', 'received_time_diff_last3',
       'trade_price_last3', 'trade_size_last3', 'trade_type_last3',
       'curve_based_price_last3', 'received_time_diff_last4',
       'trade_price_last4', 'trade_size_last4', 'trade_type_last4',
       'curve_based_price_last4', 'received_time_diff_last5',
       'trade_price_last5', 'trade_size_last5', 'trade_type_last5',
       'curve_based_price_last5', 'received_time_diff_last6',
       'trade_price_last6', 'trade_size_last6', 'trade_type_last6',
       'curve_based_price_last6', 'received_time_diff_last7',
       'trade_price_last7', 'trade_size_last7', 'trade_type_last7',
       'curve_based_price_last7', 'received_time_diff_last8',
       'trade_price_last8', 'trade_size_last8', 'trade_type_last8',
       'curve_based_price_last8', 'received_time_diff_last9',
       'trade_price_last9', 'trade_size_last9', 'trade_type_last9',
       'curve_based_price_last9', 'received_time_diff_last10',
       'trade_price_last10', 'trade_size_last10', 'trade_type_last10',
       'curve_based_price_last10'],
      dtype='object')

In [37]: train_set.columns.size
Out[37]: 61
```

Column details:

- id: The row id.
- bond_id: The unique id of a bond to aid in timeseries reconstruction. (This column is only present in the train data)
- trade_price: The price at which the trade occured. (This is the column to predict in the test data)
- **weight**: The weight of the row for evaluation purposes. This is calculated as the square root of the time since the last trade and then scaled so the mean is 1.
- current_coupon: The coupon of the bond at the time of the trade.
- **time_to_maturity**: The number of years until the bond matures at the time of the trade.
- **is_callable**: A binary value indicating whether or not the bond is callable by the issuer.

- **reporting_delay**: The number of seconds after the trade occured that it was reported.
- trade_size: The notional amount of the trade.
- **trade_type**: 2=customer sell, 3=customer buy, 4=trade between dealers. We would expect customers to get worse prices on average than dealers.
- curve_based_price: A fair price estimate based on implied hazard and funding curves of the issuer of the bond.
- received_time_diff_last{1-10}: The time difference between the trade and that of the previous {1-10}.
- trade_price_last{1-10}: The trade price of the last {1-10} trades.
- trade_size_last{1-10}: The notional amount of the last {1-10} trades.
- trade_type_last{1-10}: The trade type of the last {1-10} trades.
- curve_based_price_last{1-10}: The curve based price of the last {1-10} trades.

2.3 Selected Solutions

a. Forum Discussion

Key Methods: **Random Forest，Gradient Boosting Machine**

- Sergey Yurgen... (2nd in this Competition)
  - Initial progress was based on RF model with minor data preprocessing. Actually, we had RF with private score of ~0.727 submitted on Feb 29.
  - Then Bruce found what we thought was the "secret" ingredient - GBM. Now we can see that it was not so secret :). We made a good run using GBMs only , however our best submission was ensemble of RFs and GBMs.

- Glen • (7th in this Competition)
  - I ended up using a combination of locally weighted non-linear regression, random forests and gradient boosting. I only used variables up to curvebasedprice_last4.

- desertnaut • (10th in this Competition)
  - We used random forest for a crude initial forward feature selection, and when we (thought we had) found our feature set we proceeded to modelling with gradient boosting.
  - This gave very competitive results early on, so we proceeded with detailed parameter tuning and averaging some of our best models' outputs.
  - no clustering, no test set usage, no outlier detection, only some handling of the missing values.

    - Tried different approaches to variables modelling (averages and differentials), but it proved that nothing could surpass the non-transformed variables input. We tried to remove from the training set some price value ranges that were not present in the test set, but again this gave inferior results...

- Halla Yang • (11th in this Competition)
    - I used a random forest, with a severely limited set of features: thirteen predictive variables. I found most of the variables to be unhelpful, probably because there were intuitive sufficient statistics.
    - It was clear from the data that corporate bonds are very illiquid and that they trade in bursts: weeks or months might elapse with no trade, and then you might see a flurry of matched trades in quick succession as customers and dealers pass the bonds around like hot potatoes. I found it essential to clean/filter my data, and this is one area where I wish I had spent more time.

- Wayne Zhang (13th in this Competition)
    - I did have the same experience of overfitting RF to training data. That's why I turned to linear regression. I agree with Halla, so there may be some normalization.
    - I also used time weighted VWAP, but I found std not that helpful.

- ivo • (14th in this Competition)
    - Generated some stats from the curve_based_prices_lastx and trade_price_lastxs (sd, average, median, linear extrapolation) and from some other features like present value of a bond with a given maturity and coupon with 10% reference rates.
    - I only modeled the trading price of a bond, where the received_time_difference_last1 > 300, because the trade_price_last1 was a sufficient predictor on average for those with rtdl1<300. (That may have been a mistake.
    - Tried some regression techniques: linear regression, PACE regression, Regression trees (bagged), neural nets, local linear regression. PACE was great overall, neural nets were good where bonds were callable (handled the the callable non callable bonds separately). I clustered the dataset with the training set and reached the highest accuracy by voteing together 69 different models.

b. Insight Papers by Stanford Graduates

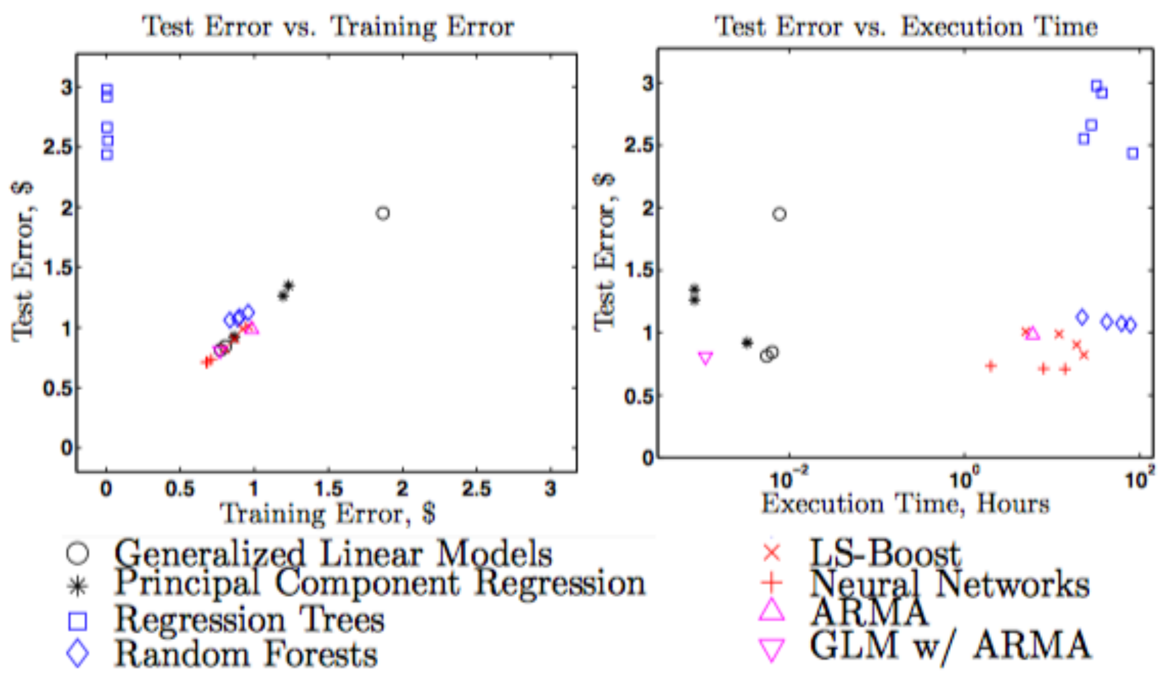
b.Results:

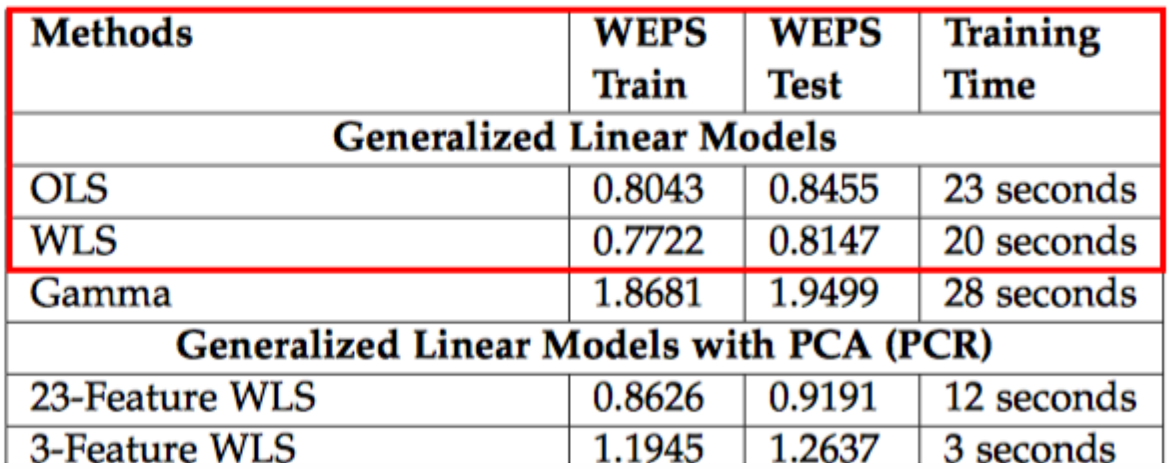
- evaluate the performance of various supervised learning algorithms for regression followed by ensemble methods, with feature and model selection considerations being treated in detail.
- we further evaluate all methods on both accuracy and speed. Finally, we propose **a novel hybrid time-series aided machine learning method** that could be applied to such

datasets in future work.

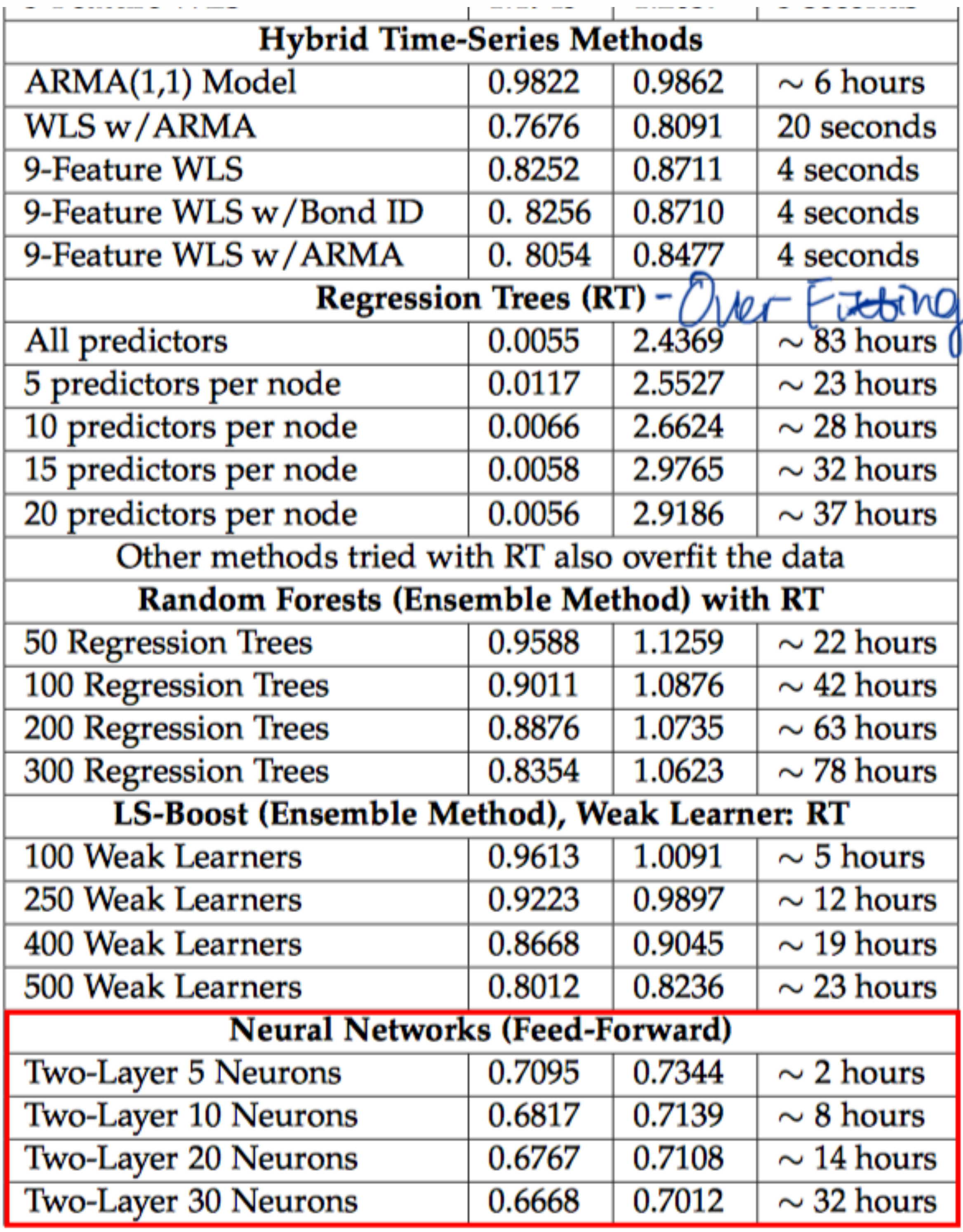
b. Work Flow(mainly talked about hybrid time series method)

- **Step1 : data exploration:** We observe from the correlation matrices that attributes Price of the Last Trade and Curve-Based Price of the Last Trade are strongly correlated at all time points; dataset is **class-balanced**;
- **Step2: Feature Generation and Selection:** *Correlation Analysis*(No attributes supplied are strongly correlated); *PCA in Supervised Learning*; *Scoring Function for Ensemble Methods*(Random Forests (RF) are used for feature ranking. If feature X appears in 25% of the trees, then score it. Otherwise, we do not consider ranking the feature because we do not have sufficient information about its performance. We then assign the performance score of every tree in which X appears to X and average the score.)
- **Step3: Model Implementation**： Generalized Linear Model(with PCA), Hybrid Time Series Mothods, Regression Trees, Random Forests(including Ensemble Methods), LS - Boost with RT(more details can be seen in the paper)
- **Comparation and Conclusion:**





Methods	WEPS Train	WEPS Test	Training Time
Generalized Linear Models			
OLS	0.8043	0.8455	23 seconds
WLS	0.7722	0.8147	20 seconds
Gamma	1.8681	1.9499	28 seconds
Generalized Linear Models with PCA (PCR)			
23-Feature WLS	0.8626	0.9191	12 seconds
3-Feature WLS	1.1945	1.2637	3 seconds



Hybrid Time-Series Methods			
ARMA(1,1) Model	0.9822	0.9862	~ 6 hours
WLS w/ARMA	0.7676	0.8091	20 seconds
9-Feature WLS	0.8252	0.8711	4 seconds
9-Feature WLS w/Bond ID	0. 8256	0.8710	4 seconds
9-Feature WLS w/ARMA	0. 8054	0.8477	4 seconds
Regression Trees (RT) - Over Fitting			
All predictors	0.0055	2.4369	~ 83 hours
5 predictors per node	0.0117	2.5527	~ 23 hours
10 predictors per node	0.0066	2.6624	~ 28 hours
15 predictors per node	0.0058	2.9765	~ 32 hours
20 predictors per node	0.0056	2.9186	~ 37 hours
Other methods tried with RT also overfit the data			
Random Forests (Ensemble Method) with RT			
50 Regression Trees	0.9588	1.1259	~ 22 hours
100 Regression Trees	0.9011	1.0876	~ 42 hours
200 Regression Trees	0.8876	1.0735	~ 63 hours
300 Regression Trees	0.8354	1.0623	~ 78 hours
LS-Boost (Ensemble Method), Weak Learner: RT			
100 Weak Learners	0.9613	1.0091	~ 5 hours
250 Weak Learners	0.9223	0.9897	~ 12 hours
400 Weak Learners	0.8668	0.9045	~ 19 hours
500 Weak Learners	0.8012	0.8236	~ 23 hours
Neural Networks (Feed-Forward)			
Two-Layer 5 Neurons	0.7095	0.7344	~ 2 hours
Two-Layer 10 Neurons	0.6817	0.7139	~ 8 hours
Two-Layer 20 Neurons	0.6767	0.7108	~ 14 hours
Two-Layer 30 Neurons	0.6668	0.7012	~ 32 hours

Figure 4: *Summary of Results.*

-
  - GLM Perform with low computational cost
  - Ensemble methods do not substantially improve the performance
  - Neural networks give very accurate results without overfitting in a reasonable amount of time.