1. Algorithmic Trading Challenge

1.1 Overview

Goal:

- The aim of this competition is to determine the relationship between recent past order book events and future stock price mean reversion following shocks to liquidity.

Submission Format(Things to be predicted) :

- For each observation a participant should provide 100 numbers describing the next 50 bid and ask prices(from t51 to t100) following a liquidity shock. There should be 50,001 rows in total (50,000 entries plus a header row) and 101 columns for each row.

Evaluation Method :

- Root Mean Square Error(RMSE)： $\sqrt{\frac{\Sigma_{t=1}^{n}(\hat{y}_t - y_t)^2}{n}}$, where $\hat{y}_t$ and y_t are the predicted/actual value respectively.

1.2 Data

The competition host provide 4 files for preparation， train.csv & test.csv, plus example_entry_example.csv&example_entry_naive.csv, the last two seems to be used as submission template, though I can't tell their difference. But it seems that these two files have similiar dimension, and differences seems to have patterns. Here's the comparison using pandas.DataFrame：

```
In [26]: ex_entry_linear == ex_entry_naive
Out[26]:
       row_id  bid51  ask51  bid52  ask52  bid53  ask53  bid54  ask54  bid55  \
0        True  False   True  False   True  False   True  False   True  False
1        True   True  False   True  False   True  False   True  False   True
2        True  False   True  False   True  False   True  False   True  False
3        True   True  False   True  False   True  False   True  False   True
4        True   True  False   True  False   True  False   True  False   True
5        True  False   True  False   True  False   True  False   True  False
6        True  False   True  False   True  False   True  False   True  False
7        True  False   True  False   True  False   True  False   True  False
8        True   True  False   True  False   True  False   True  False   True
9        True   True  False   True  False   True  False   True  False   True
10       True   True  False   True  False   True  False   True  False   True
11       True   True  False   True  False   True  False   True  False   True
12       True   True  False   True  False   True  False   True  False   True
13       True  False   True  False   True  False   True  False   True  False
14       True   True  False   True  False   True  False   True  False   True
```

Anyway, the meaning of provided labels in training/test set are shown as below：

Table 1 Data Schema

Variable Type	Predictors																						Responses				
Event Time								t=1				...	t=49				t=50				t=51	...	t=100				
Column Id	1	2	3	4	5	6	7	8	9	10	11	...	200	201	202	203	204	205	206	207	208	209	...	305	306		
Variable Name	row_id	security_id	p_tcount	p_value	trade_vwap	trade_volume	initiator	transtype<t>	time<t>	bid<t>	ask<t>	⋮	transtype<t>	time<t>	bid<t>	ask<t>	transtype<t>	time<t>	bid<t>	ask<t>	bid<t>	ask<t>	⋮	bid<t>	ask<t>		
													Trade						Quote								
													Liquidity Shock														

Table 2 Data Fields

Variable Name	Description	Type	Example
row_id	Unique row identifier	Integer	6
security_id	Unique stock identifier	Integer	24
p_tcount	Count of previous day's on-market trades in current security	Integer	670
p_value	Sum of previous day's on-market trade values in current security (£)	Integer	65,000
trade_vwap	Volume-weighted average price of the trade causing the liquidity shock (pence)	Double	4.250
trade_volume	Size of the trade causing the liquidity shock (number of shares)		1,500
initiator	Whether the trade is buyer or seller initiated ('B'=Buyer, 'S'=Seller)	String	B
transtype<t>	Whether the time-series event is a trade or a quote ('T'=Trade, 'Q'=Quote) at event time t	String	T
time<t>	Event time (HH:MM:SS.mmm) at event time t	String	10:05:36.488
bid<t>	Best buy price (pence) at event time t	Double	213.85
ask<t>	Best sell price (pence) at event time t	Double	214.10

The Size of each files are listed:

```
##train.csv
In [10]: training_set.index
Out[10]: RangeIndex(start=0, stop=754018, step=1)

##test.csv
In [22]: test_set.index
Out[22]: RangeIndex(start=0, stop=50000, step=1)

##example_entry_linear.csv
In [13]: ex_entry_linear.index
```

```
Out[13]: RangeIndex(start=0, stop=50000, step=1)

##example_entry_naive.csv
In [16]: ex_entry_naive.index
Out[16]: RangeIndex(start=0, stop=50000, step=1)
```

1.3 Selected Solutions

Key Methods: Time Partitioning Prediction, Random Forest, Ensemble Methods and Feature Extractions.

a. Top1 Solution

a. Results:

- Ranked 1st in the private leaderboard, 4th in the public leaderboard.

a. Work Flow:

- **Step1: Time Interval Partitioning Algorithm**, Due to identity between t=50 and t=51, the partition start from t=52 to t=100. Use greedy algorithm to make partition(when the error in algorithm step7 begins to rise, then stop and separate), and use different sub-models($M_{bit}(t)$ & $M_{ask}(t)$ for bit/ask prediction respectively). Furthermore, assume the length of later segmentation is larger than the earlier ones(length(C_{i+1}) > length(C_i)), this is derived from the main hypothesis of the model:

  ```
  'an always increasing prediction error may require averaging longer price time
  series to obtain a constant price prediction with an acceptable error'
                                                  -Cited from author
  ```

  Different $M_{bit,i}(t)$ & $M_{ask,i}(t)$ are used to describe each time periods(C_i):

$$M_{\text{bid}}(t) = \sum_{i=1}^{K} a_{i,t} M_{\text{bid},i}(t),\ M_{\text{ask}}(t) = \sum_{i=1}^{K} a_{i,t} M_{\text{ask},i}(t),$$

$$\text{where } a_{i,t} = \begin{cases} 1 \text{ if t} \in C_i, \\ 0 \text{ otherwise}, \end{cases} \quad t \in [52, 100].$$

Algorithm 1 Time interval partitioning algorithm

```
1: b ← e ← 52; P ← NULL; i ← 1
2: while b < 100 do
3:    C_i ← NULL; e ← b + length(C_i); bestError← ∞
4:    repeat
5:       C_i ← createTimeInterval(b,e)
6:       C_all ← createTimeInterval(e+1,100)
7:       error← evaluateModel(P,C_i,C_all)
8:       if bestError > error then
9:          bestError← error
10:      end if
11:      e ← e + 1
12:   until bestError≠ error
13:   addTimeInterval(P, C_i)
14:   i ← i + 1; b ← e
15: end while
```

- **Step2: Feature Extraction**, The author have extracted over 150 features(divided into 4 classes: *Price, Liquidity book, Spread, Rate*) from the original data, using a specified R module (I guess)

> *Price:* Price features provide information about the bid/ask normalized price time series (price values divided by the post liquidity shock price). Technical analysis [4] and statistical estimators are the fundamental instruments to compute these predictors (e.g., the detrended price oscillator, the exponential moving average of the last n [bid/ask] prices before the liquidity shock, the number of price increments during the last n [bid/ask] prices before the liquidity shock).
>
> *Liquidity book:* This class contains all features able to provide information about the depth of the liquidity book (e.g., liquidity book improvements in the last n time periods understood as bid/ask price increases between two consecutive quotes).
>
> *Spread:* Spread related features are meant to distill information about the bid/ask spread. As price features, technical analysis and statistical estimators are the fundamental instruments to compute these predictors. Before computing the predictor, spread time series were divided by the minimum price increment allowed for the particular *security_id* (e.g., exponential moving average of the last n spreads before the liquidity shock).
>
> *Rate:* This class of features provides information about the arrival rate of orders and/or quotes (e.g., number of quotes and/or trades during the last n events).

- **Step3: Modeling Approach Selection,** First, using MIC(Maximal Information Coefficient) to analyze the mutual infromation between features and the dependent variable, revealing a very low mutual information and non-functional relationships; then tried both Random Forest and Gradient Boosting Machines to select the data.(4-cross validation, and the **refined training set** consisted of 1.same size 2. same combination of security_id as the test set, then choose RF for lower RMSE cost.)

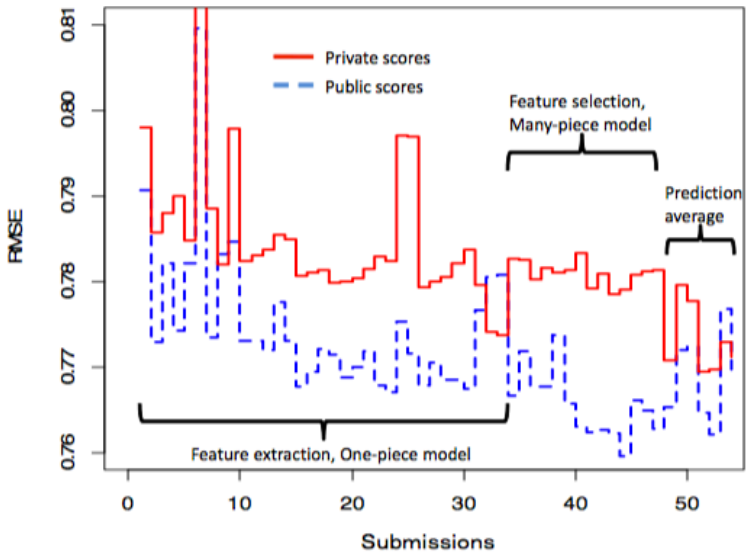
#

- **Step4: Feature Selection**, inspired by a similiar method applied to the *Kaggle*'s Heritage Health Prize dataset. The algorithm are divided into 2 parts(1. step1 - 8: to get the quasi-optimized S_f quickly, 2. Rest of the steps: to make local adjustments on these S_f feature set).

Algorithm 2 Feature selection algorithm

1: Train a single piece model using all S features
2: Compute model performance against the test set
3: Rank features importance (RF importance method)
4: **for** each subset size S_i=$S, S-1, \ldots, 1$: **do**
5: Retrain the model with only S_i most important features
6: Re-compute individual variable importance and re-rank
7: Fit the model to S_f features and rank individual features
8: **end for**
9: Determine which S_i yielded the smallest RMSE. Call this S_f
10: **repeat**
11: Choose a set of semantically similar features from S_f
12: Select the feature with less rank not selected before
13: Evaluate the model performance
14: If smaller RMSE, then remove the feature
15: **until** no improvement
16: **repeat**
17: Choose a feature set among the already removed in Steps 1)-9) considering only those semantically orthogonal with the already selected in Steps 1)-15)
18: If smaller RMSE, then we add the feature to S_f
19: **until** no improvement

- **Step5: Validation**:
  - **Feature Set** Validation: Ironically, the author used the same S_f set for both F_b and F_a (bit and ask feature set for bit and ask price respectively) at the final stage due to the lack of time for calcu dlation. **BUT, for the different time period, the F_b is different.**
  - **Optimal Time Segmentation**: The final partition of the time period is {t=52, t=53, t=54–55, t=56–58, t=59–64, t=65–73, t=74–100}, which I thought quite make sense: the nearest period has the highest weight, so is sepearted.
- **Model Performance via methods**:



a. Comments

- Pros: Innovated on both *Time Segmentation* and *Feature Selction* process.
- Cons: In reality, the company only cares about the most recent prediction on the price, input new data and genreate new prediction, so there won't be error accumulation, the *Time Segmentation* process seems to be meaningless, plus, the feature set for the F_a is not optimal.

b. Forum Disscussion

- Christopher Hef (4th in this Competition)
    - Observations on data
    1. Bids/asks from T=1...T=47 seemed to provide little predictive value;
    2. The error contribution right at the market open (at 8AM) was extremely large;
    3. Prediction accuracy varied across time. Using a holdout set & one of our models, I found that the error rose as you got farther from the liquidity-event trade;
    4. The "liquidity event" trades did not seem to impact prices very much.

- Sergey Yurgens(6th in this Competition)

  Here is the secret recipe for Linear Regression meal:
    - go to your friendly neighbor datastore and choose couple fresh pieces of data (day1 and last 50k)
    - cut out bones and extra fat (leave only columns 5 170 206 207)
    - cook separately "seller initiated" transactions and "buyer initiated" transactions using your favorite linear regression function (do it separately for each askN and bidN to be predicted)
    - use 200 created LRs to calculate required predictions and nicely plate them into submission file
    - Serve hot, because you do not want to miss 0.77590 public score and 0.77956 private score :)

- Cole Harris • (9th in this Competition)
    - I had separate models for t<60 & t>60.

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)

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

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	$\sim$ 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	$\sim$ 83 hours
5 predictors per node	0.0117	2.5527	$\sim$ 23 hours
10 predictors per node	0.0066	2.6624	$\sim$ 28 hours
15 predictors per node	0.0058	2.9765	$\sim$ 32 hours
20 predictors per node	0.0056	2.9186	$\sim$ 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.

3. The Winton Stock Market Challenge

3.1 Overview

Goal:

- To Find the hidden signal in the terabytes of noisy, non-stationary data via novel statistical modelling and data mining techniques. In this competition the challenge is **to predict the return of a stock, given the history of the past few days.**

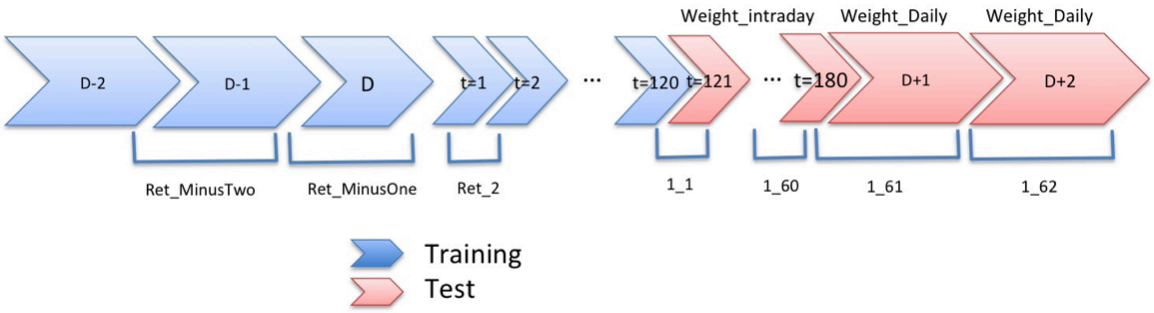
Evaluation Method:

- Provide 5-day windows of time, days D-2, D-1, D, D+1, and D+2. You are given returns in days D-2, D-1, and part of day D, and you are asked to predict the returns in the rest of day D, and in days D+1 and D+2.
- Weighted Mean Absolute Error *Weighted Factors is associated with the return*(similiar with the Benchmark competition):

$$WMAE = \frac{1}{n}\sum_{i=1}^{n} w_i \cdot \left|y_i - \hat{y}_i\right|,$$

3.2 Data

Basically just train.csv&test.csv, and a csv file for the submission template:



Provide 5-day windows of time, days D-2, D-1, D, D+1, and D+2. You are given returns in days D-2, D-1, and part of day D, and you are asked to predict the returns in the rest of day D, and in days D+1 and D+2.

During day D, there is intraday return data, which are the returns at different points in the day. We provide 180 minutes of data, from t=1 to t=180. In the training set you are given the full 180 minutes, in the test set just the first 120 minutes are provided.

For each 5-day window, we also provide **25 features**, Feature_1 to Feature_25. These may or may not be useful in your prediction.

Each row in the dataset is an arbitrary stock at an arbitrary 5 day time window.

- **train.csv** - the training set, including the columns of:
  - Feature_1 - Feature_25
  - Ret_MinusTwo, Ret_MinusOne
  - Ret_2 - Ret_120
  - Ret_121 - Ret_180: **target variables**
  - Ret_PlusOne, Ret_PlusTwo: **target variables**
  - Weight_Intraday, Weight_Daily
- **test.csv** - the test set, including the columns of:
  - Feature_1 - Feature_25
  - Ret_MinusTwo, Ret_MinusOne
  - Ret_2 - Ret_120

3.3 Selected Solution

Pretty Tricky this one...

3.4 Comment

4. Web Traffic Time Series Forecasting

4.1 Overview

Goal To forecast the future values of multiple time series. In this competition the challenge is **to predict the return of a stock, given the history of the past few days.**

Evaluation Method: SMAPE(Symmetric mean absolute percentage error)。

Highlight two stages of competition, second part is based on real-time data.

4.2 Data

train_*.csv - contains traffic data. This a csv file where each row corresponds to a particular article and each column correspond to a particular date. Some entries are missing data. The page names contain the *Wikipedia project (e.g. en.wikipedia.org), type of access (e.g. desktop) and type of agent (e.g. spider).*

In other words, each article name has the following format: 'name_project_access_agent' (e.g. 'AKB48_zh.wikipedia.org_all-access_spider').

key_*.csv - gives the mapping between the page names and the shortened Id column used for prediction

sample_submission_*.csv - a submission file showing the correct format

```
In [5]: train_set.index
Out[5]: RangeIndex(start=0, stop=145063, step=1)

In [7]: train_set.columns
Out[7]:
Index(['Page', '2015-07-01', '2015-07-02', '2015-07-03', '2015-07-04',
       '2015-07-05', '2015-07-06', '2015-07-07', '2015-07-08', '2015-07-09',
       ...
       '2016-12-22', '2016-12-23', '2016-12-24', '2016-12-25', '2016-12-26',
       '2016-12-27', '2016-12-28', '2016-12-29', '2016-12-30', '2016-12-31'],
      dtype='object', length=551)
```

so 550 daily traffic data of 145063 Wikepedia sites for the training set

4.3 Top Solution Top 1 Solution with Code, Top 2 Discussion, Top 6 With Code

Key Methods: RNN seq2seq, 就这些github上已经非常全了，以下主要概括Top 1的solution

4.3.1 Information Sources for Prediction

- Local features. If we see a trend, we expect that it will continue **(AutoRegressive model)**, if we see a traffic spike, it will gradually decay **(Moving Average model)**, if wee see more traffic on holidays, we expect to have more traffic on holidays in the future **(seasonal model)**.
- Global features. If we look to autocorrelation plot, we'll notice strong year-to-year autocorrelation and some quarter-to-quarter autocorrelation.

The good model should use both global and local features, combining them in a intelligent way.

4.3.2 Model Selection - RNN seq2seq

1. RNN can be thought as a natural extension of well-studied ARIMA(Autoregressive integrated moving average) models, but much more flexible and expressive.
2. RNN is non-parametric, that's greatly **simplifies learning**. Imagine working with different ARIMA parameters for 145K timeseries.
3. Any exogenous feature (numerical or categorical, time-dependent or series-dependent) can be easily injected into the model
4. seq2seq seems natural for this task: we predict next values, conditioning on joint probability of previous values, including our past predictions. Use of past predictions stabilizes the model, it learns to be conservative, because error accumulates on each step, and extreme prediction at one step can ruin prediction quality for all subsequent steps.
5. Deep Learning is all the hype nowadays —> RNN。。。

4.3.3 Feature Engineering

I tried to be minimalistic, because **RNN is powerful enough to discover and learn features on its own.** Model feature list:

- *pageviews* (spelled as 'hits' in the model code, because of my web-analytics background). Raw values transformed by log1p() to get more-or-less normal intra-series values distribution, instead of skewed one.
- *agent*, *country*, *site* - these features are extracted from page urls and one-hot encoded
- *day of week* - to capture weekly seasonality
- *year-to-year autocorrelation*, *quarter-to-quarter autocorrelation* - to capture yearly and quarterly seasonality strength.
- *page popularity* - High traffic and low traffic pages have different traffic change patterns, this feature (median of pageviews) helps to capture traffic scale. This scale information is lost in a *pageviews* feature, because each pageviews series independently normalized to zero mean and unit variance.
- ***lagged pageviews* - I'll describe this feature later**

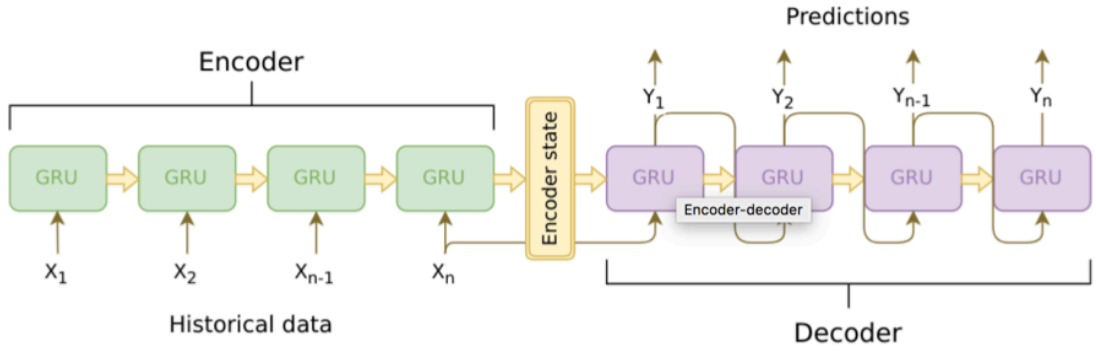
4.3.4 Feature Preprocessing

- All features (including one-hot encoded) are **normalized** to zero mean and unit variance. Each *pageviews* series normalized independently.
- Time-independent features (autocorrelations, country, etc) are "stretched" to timeseries length i.e. repeated for each day by `tf.tile()` command.
- Model trains on random fixed-length samples from original timeseries. For example, if original timeseries length is 600 days, and we use 200-day samples for training, we'll have a choice of 400 days to start the sample.

This sampling works as effective data augmentation mechanism: training code randomly chooses starting point for each timeseries on each step, generating endless stream of almost non-repeating data.

4.3.5 Model Core

Model has two main parts: encoder and decoder.

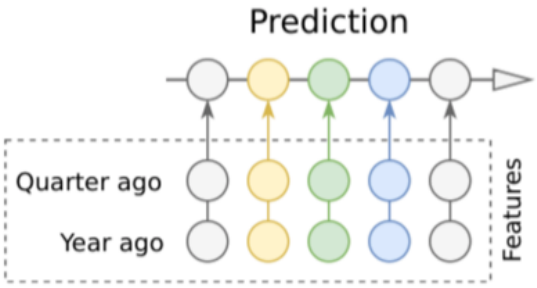


4.3.6 Working with long timeseries

- LSTM/GRU能够很好的处理短序列的问题(100-300items)，但比赛中涉及到超过700天的数据，因此作者采用了一些方法来'strengthen' GRU memory.
    - First trial is called 'attention' method:

I can just take encoder outputs from `current_day - 365` and `current_day - 90` timepoints, pass them through FC layer to reduce dimensionality and append result to input features for decoder. This solution, despite of being simple, **considerably lowered prediction error.**

- Then, to reduce the noise: `attn_365 = 0.25 * day_364 + 0.5 * day_365 + 0.25 * day_366`
- Then I realized that `0.25,0.5,0.25` is a 1D convolutional kernel (length=3) and I can automatically learn bigger kernel to detect important points in a past.（非常6）
- Ended up with a monstrous attention mechanism not classical method such as **(Bahdanau or Luong attention)**, because of calculation cost.
- 但其实最好的public score反倒是最简单的模型：



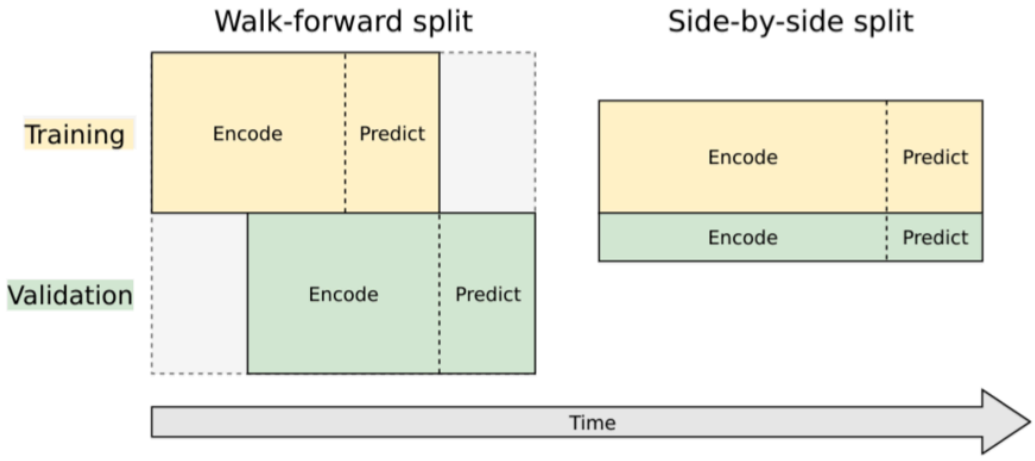
- 另外发现60-90day的encoder表现也还okay

4.3.7 Redefine Loss Function

- SMAPE (target loss for competition) can't be used directly, because of unstable behavior near zero values (loss is a step function if truth value is zero, and not defined, if predicted value is also zero).
- Candidates include *differentiable SMAPE* and *MAE*

4.3.8 Training and validation

- COCOB optimizer
- Prefer walk-forward split rather than side-by-side split



4.3.9 Reducing model variance

- Performance varies from training steps and seed.
- Solutions:
  - But I know approximate region where model is (possibly) trained well enough, but (possibly) not started to overfit. I decided to set this optimal region bounds to 10500..11500 training steps and save 10 checkpoints from each 100th step in this region.
  - Similarly, I decided to train 3 models on different seeds and save checkpoints from each model. So I have 30 checkpoints total.
  - One widely known method for reducing variance and improving model performance is SGD averaging (ASGD). Method is very simple and well supported in Tensorflow - we have to maintain moving averages of network weights during training and use these averaged weights, instead of original ones, during inference.
  - 然后 AVERAGE！！！

4.3.10 Hyperparameter Tuning

略略略。。。

4.4 Comment

solution set非常丰富，1st place有着详细的code和readme，用的是RNN，有一些地方feature extraction不清晰，可以先做。

5. Two Sigma Financial Modeling Challenge

5.1 Overview

Goal To forecast the future values of multiple time series. In this competition the challenge is **to predict the return of a stock, given the history of the past few days.**

Evaluation Method: R Value(signed square root of R^2 value), Negative R values are clipped at -1, i.e. the score you see will be max(−1,R)max(−1,R)

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

$$R = sign\left(R^2\right)\sqrt{\left|R^2\right|},$$

5.2 Data

train_*.csv 就这一个文件，我kaggle的api(应该是kaggle不支持python3.6?)出现了一些问题，暂时没下载下来数据，但是可以看到很多data interpretation在discussion session里面

5.3 Top Solution Top 12 Solution with Code, Some Exploration

Key Methods: **Ensemble Learning** Ridge 和 拉搜

Our solution (Giba and Xpeuler) is a basic blend of 7 models:

1. Ridge A - Selected Features trained on SignedExp(y+1). Some cleaning on features and filter on target instances.
2. Ridge B - Selected features and some cleaning on features and filter on target instances.
3. Ridge C - Selected features and some cleaning on features and filter on target instances.
4. Extra Trees - Selected features. 222 trees
5. XGB - Selected Features and tuned hyperparameters on "all" trainset.
6. Ridge online rolling fit - Trained every 100 steps on submission time. features: Some lags of [technical20-technical30]. Target used: lag1 of [technical20-technical30]
7. Variance by Step(day) - Simple variance calculated over all 'Id' per day

The final predictions are a weighted average of that 7 models.

Crossvalidation for model performance and feature selection was made using some approaches:

- 2 folds: timestamp > 906 and timestamp <= 906
- 5 kfolds
- rolling fit for ts> 906

5.4 Comment

solution set不是很丰富，我看他们的解答都各种说“black magic”。。。

6. Predict short term movements in stock prices using news and sentiment data provided by RavenPack

6.1 Overview

Goal To develop a model that predicts stock price movements using sentiment data provided by RavenPack.

Evaluation Method: mean absolute error

$$\mathrm{MAE} = \frac{1}{n}\sum_{i=1}^{n}|y_i - \hat{y_i}|$$

6.2 Data

data.zip - contains features for 510 days worth of trading, including 200 training days and 310 testing days **trainLabels.csv** - contains the targets for the 200 training days **sampleSubmission.csv** - shows the submission format

Each variable named O1, O2, O3, etc. (the outputs) represents a percent change in the value of a security. Each variable named I1, I2, I3, etc. (the inputs) represents a feature. *（所以这里还是把sentiment已经抽象成了feature）* The underlying securities and features represented by these anonymized names are the same across all files (e.g. O1 will always be the same stock)...

6.3 Top Solution Top 5% Solution,PDF

Key Methods: Feature Selection

6.4 Comment

solution set不是很丰富，披着nlp的外衣，其实还是在抽象的feature做选择，**高频交易**