**Two Sigma: Using news to predict stock movements**

# Introduction

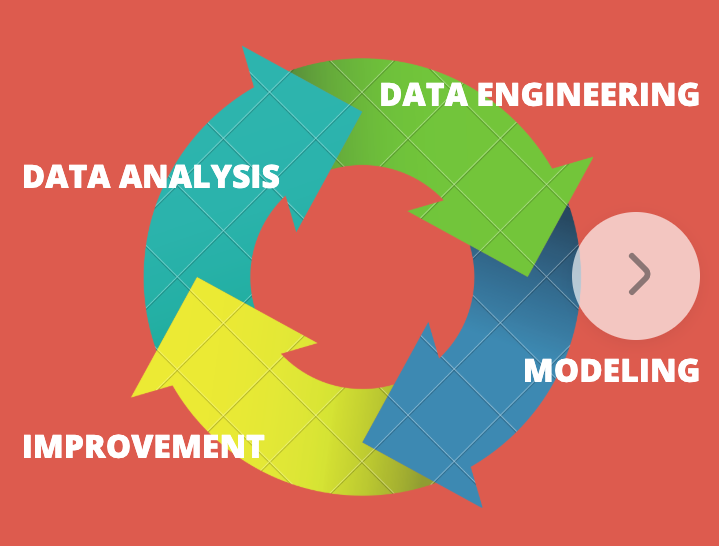
This project is a Kaggle competition that hosted by Two Sigma, the purpose of this project is to find a solution for predicting stock movements using news and market data. The competition use sigma score as evaluation metric to rank the result of the competition. This is a good chance to use industrial data and learn the way to build model and measure the result of a big finance company.

# Process

We follow process like this:

The process is a loop of these stages:

* Data Analysis
* Data Engineering
* Modeling
* Evaluation and improvement



Try stage of data analysis and data engineering fast to build simple model to make prediction and compute accuracy along with sigma score.

After that we have a framework and base on it to develop and improve

After measure the result we repeat the steps again, we pay more time on

# Challenges

The competition is closed 2 months a go and we could not download the original data. Fortunately, we downloaded the data some one had exported before. This data is also processed by remove all raw that have universal equals 0 and add some columns.

The original data has 6 million for market data and 9 million rows for news. This one had 2.4 million rows market data and 4 million rows news data.

Another challenges is stock prediction is a hard problem and data has many noisy.

There are many new definitions in stock market we have to cover and we are all new to stock market

Because data is so big so training time is also a very big problem.

We could not measure the result with the competitor on Kaggle because the test data is no longer available

# Data analysis

## Market data

About the market data description the details is on Kaggle. In general, each row of market data is a market history information of a stock (instrument) for a date include volume, open, close price previous change price of previous 10 days and 1 day of open and close, the same for next 1 and 10 days.

The market data is from 2007 to 2016

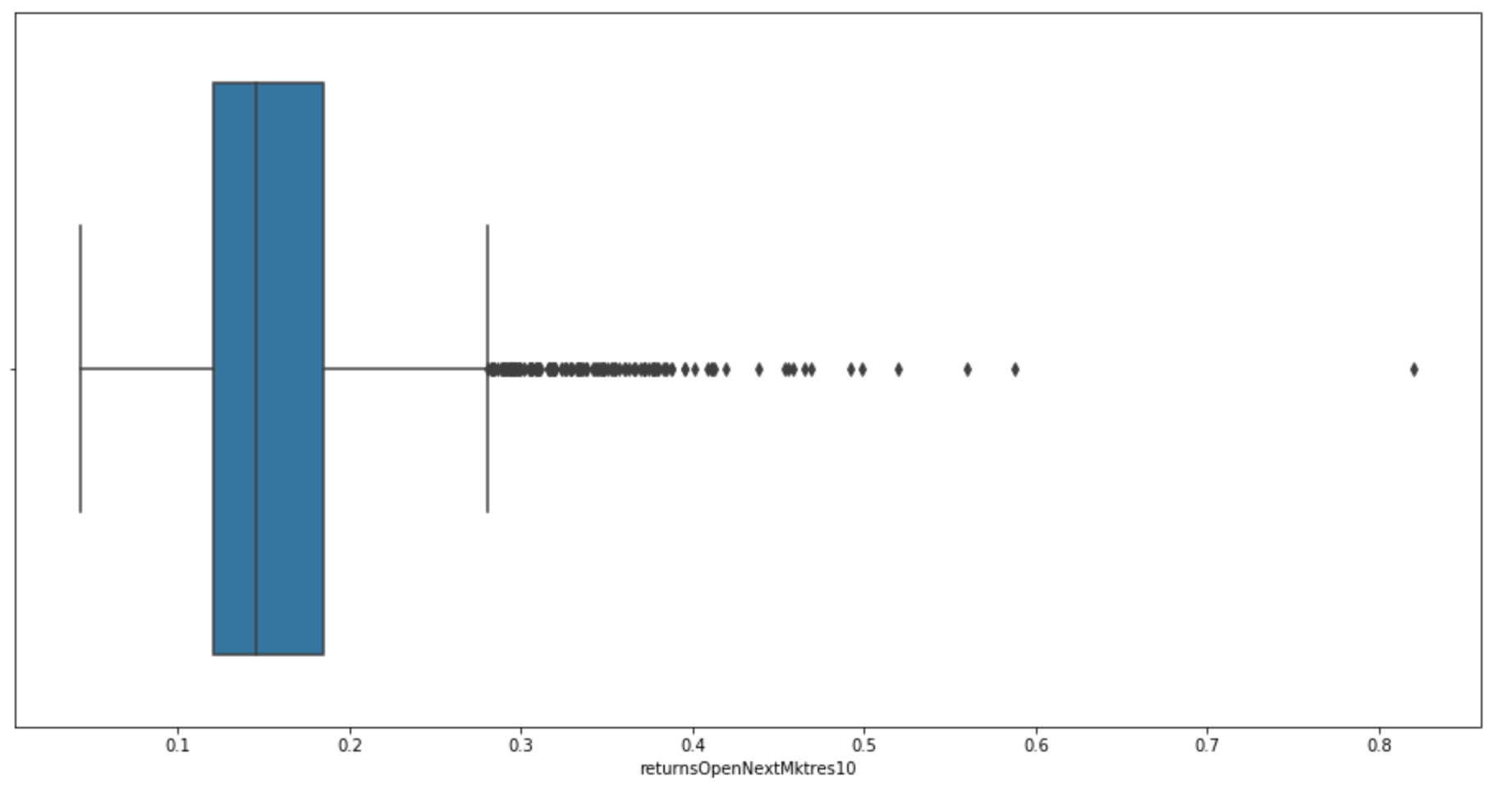
*Price of 5 random assets*



*returnOpenNextMktres10 of 5 random assets*



*Outlier of returnOpenNextMktres10*



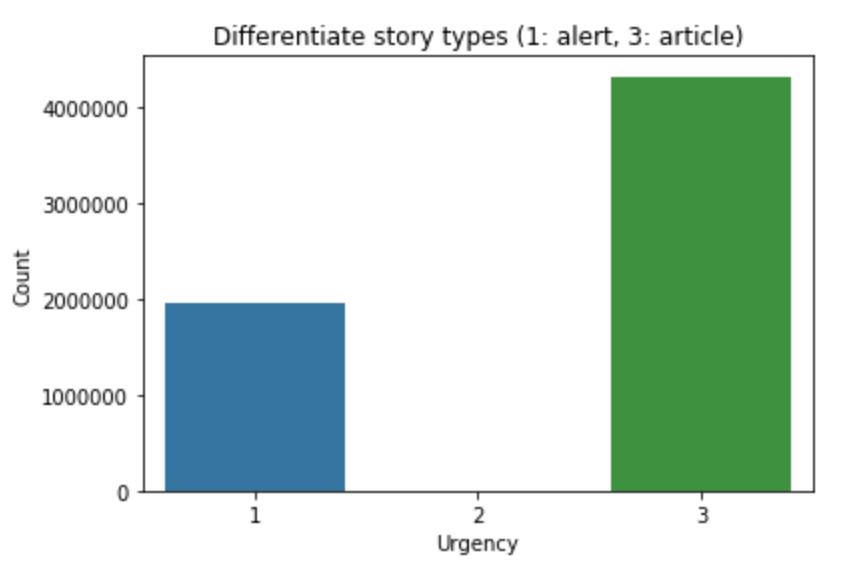
We also discover that the data before 2009 is not good so when training and testing we choose data from 2009

## News data

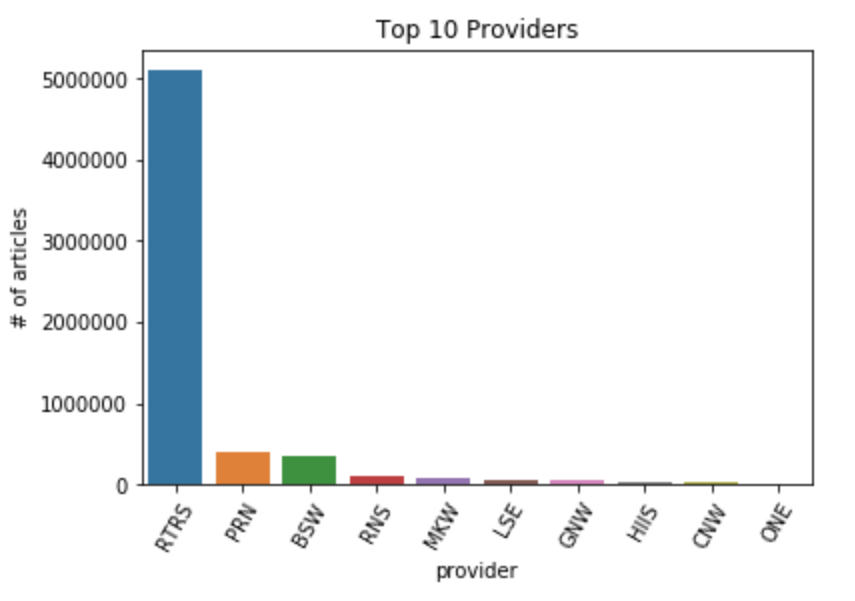
The details description about news is data on Kaggle. For each date, the news is provided by providers (Thomson for example). And that news related to some stocks, there are some important information like sentiment score and relevance score provided by two sigma.

Another field is firstMentionSentence, the good news for stock if it is mentioned at headline or beginning of the story body

New base on urgency



Providers:



## Evaluation

For each day t, we calculate the return as sum of all returns of all stocks

xt = sum of all stock(y\_hat\*r\*u) (1)

Inside ( and ) is a expression used to calculate for one particular stock

After that the score is calculated by mean divided by standard deviation of all daily xt

Score = mean(xt)/ standard deviation(xt) (2)

In (1), y\_hat is confident value that we have to predict

r as the return of next 10 days that provided by dataset

u is 0 or 1 indicate that the stock is calculated score or not

Data of market is very noisy so using regression to predict is not good so we choose classification for stock prediction.

From returnOpenNextMktres10 (10 day market-residualized return) we label data

*if returnOpenNextMktres10 larger than 0 label 1 else label 0*

The problem becomes the binary classification that to classify up (1) or down (0) base on the probability of prediction output in range [0,1] we map it to [-1,1] (y\_hat = y\_prob\*2 - 1). The purpose of mapping from [0,1] to [-1,1] is completion requirement of prediction range

So there are 4 cases: we predict right direction it mean (y\_hat>0 and r>0) or (y\_hat < 0 and r<0) the xt will be larger than 0 (we have revenue that day t) otherwise we have a lost at that day t

Assume we predict right direction and value r is big but y\_hat is very small so the x(t) will be small so the result is not good

# Data engineering

## Market data

Handle missing data

Add Technical Analysis data:

We are strongly believe that TA is very helpful for trader to make trade decision so adding these TA features make model more predictive capacity

There are some TA features that we use: Moving average, Log Return, 10 days percentage change, momentum and volatility

## News data

There are some features that is very important for our model is sentiment (negative, neutral and positive) and relevance another feature is headline.

# Modeling

We define the problem to binary classification because the data is noisy so it is not good to use regression. We make label of 0 mean the next 10 day the price not move up otherwise the label is 1.

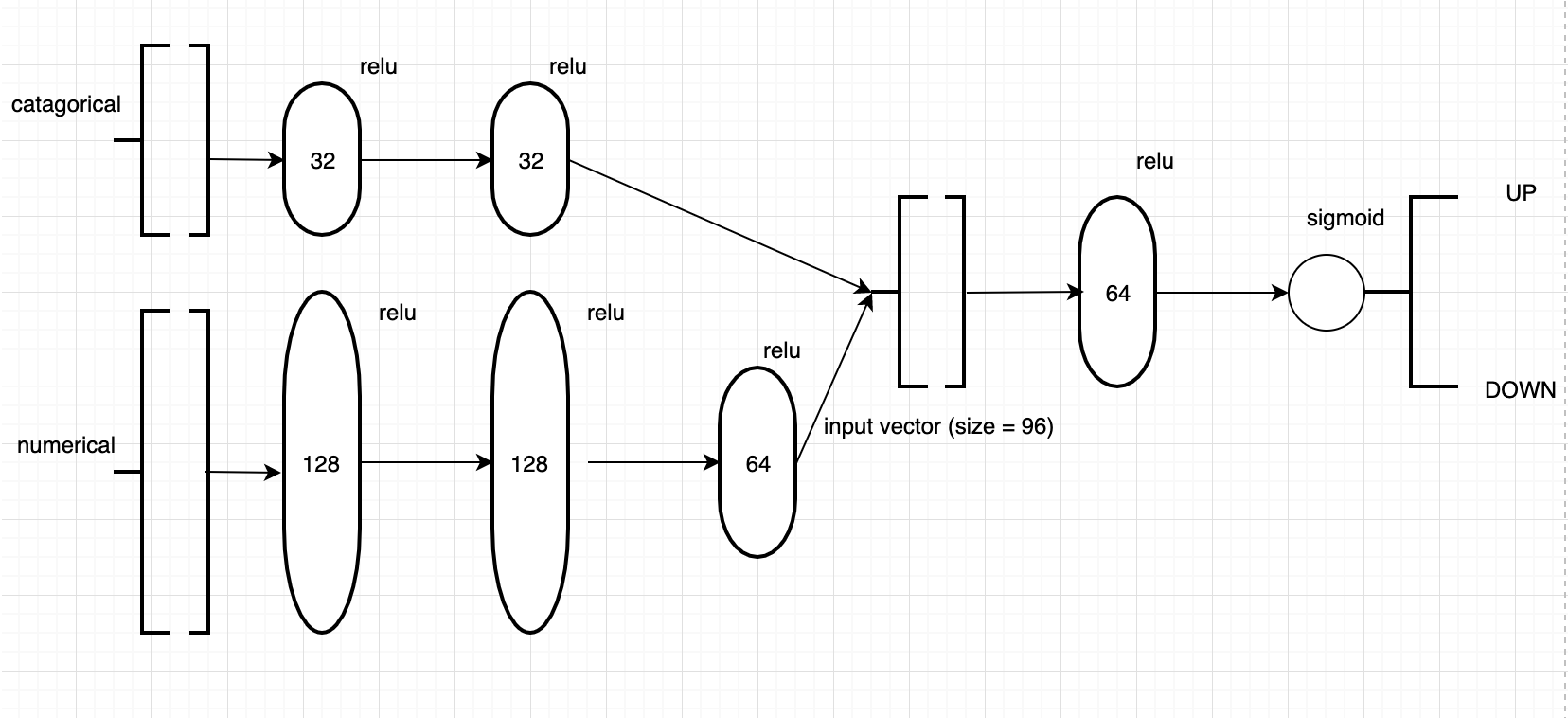
There is a trick is that the output of model is a probability of the price is likely to move up or move down. We use it to be confidence score as the prediction confidence value that two sigma required.

Data of stock is very noisy so we will use model can catch the weak signal of market using simple model random forest, logistic regression, linear classifier like Ridge, LGMB.

To know how marker or news effect to prediction we build model that just use market data and news data. The result of using market data is about 0.54 (random forest) and news is about 0.5 (random forest). It mean that news seem to be not effect to our prediction confidence value. The result is disappointed because news is not important as market data.

After marge market data and news data we build model using: Random Forest, XGBoot and LGBM (LGB is mentioned a lot on Kaggle so we have to try)

Neutral network also give a good result on sigma score



There are 2 measurements we use to evaluate: the first is accuracy that is up or down and the second is Two sigma

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Accuracy train | Accuracy test | Sigma score train | Sigma score test |
|  |  |  |  |  |
| Random Forest | 0.56 | 0.52 | -0.2 | -0.4 |
| XGBoost | 0.57 | 0.54 | -1 | -0.36 |
| LGBM | 0.6 | 0.59 | 1.4 | 0.84 |
| Neutral Network | 0.58 | 0.53 | 0.5 | 0.34 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Accuracy train | Accuracy test | Sigma score train | Sigma score test |
|  |  |  |  |  |
| Random Forest +XGBoost | 0.57 | 0.54 | 0.4 | 0.28 |
| ExtraTree + Ridge | 0.53 | 0.51 | 0.2 | 0.2 |
| LGBM + LGBM | 0.6 | 0.58 | 1.4 | 0.81 |
| Neutral Network +LGBM | 0.59 | 0.58 |  | 0.76 |

# Experiment

After Ensemble many weak models we got a good result on training and acceptable result on test, it make us surprise because combine some week models can make a better model in this case the evaluation metric is sigma score. It means that although the prediction of up or down was not good but the confidence value is better than using just one simple model.

# Improvement

* Transform news data to make it better for our prediction
* Make more market data features
* Using cross validation
* Handle high volatility strategy

# Appendix

### (From Kaggle)

### Market data

* Returns are always calculated either open-to-open (from the opening time of one trading day to the open of another) or close-to-close (from the closing time of one trading day to the open of another).
* Returns are either raw, meaning that the data is not adjusted against any benchmark, or market-residualized (Mktres), meaning that the movement of the market as a whole has been accounted for, leaving only movements inherent to the instrument.
* Returns can be calculated over any arbitrary interval. Provided here are 1 day and 10 day horizons.
* Returns are tagged with 'Prev' if they are backwards looking in time, or 'Next' if forwards looking.

Within the marketdata, you will find the following columns:

* time(datetime64[ns, UTC]) - the current time (in marketdata, all rows are taken at 22:00 UTC)
* assetCode(object) - a unique id of an asset
* assetName(category) - the name that corresponds to a group of assetCodes. These may be "Unknown" if the corresponding assetCode does not have any rows in the news data.
* universe(float64) - a boolean indicating whether or not the instrument on that day will be included in scoring. This value is not provided outside of the training data time period. The trading universe on a given date is the set of instruments that are avilable for trading (the scoring function will not consider instruments that are not in the trading universe). The trading universe changes daily.
* volume(float64) - trading volume in shares for the day
* close(float64) - the close price for the day (not adjusted for splits or dividends)
* open(float64) - the open price for the day (not adjusted for splits or dividends)
* returnsClosePrevRaw1(float64) - see returns explanation above
* returnsOpenPrevRaw1(float64) - see returns explanation above
* returnsClosePrevMktres1(float64) - see returns explanation above
* returnsOpenPrevMktres1(float64) - see returns explanation above
* returnsClosePrevRaw10(float64) - see returns explanation above
* returnsOpenPrevRaw10(float64) - see returns explanation above
* returnsClosePrevMktres10(float64) - see returns explanation above
* returnsOpenPrevMktres10(float64) - see returns explanation above
* returnsOpenNextMktres10(float64) - ***10 day, market-residualized return. This is the target variable used in competition scoring. The market data has been filtered such that returnsOpenNextMktres10 is always not null.***

### News data

* time(datetime64[ns, UTC]) - UTC timestamp showing when the data was available on the feed (second precision)
* sourceTimestamp(datetime64[ns, UTC]) - UTC timestamp of this news item when it was created
* firstCreated(datetime64[ns, UTC]) - UTC timestamp for the first version of the item
* sourceId(object) - an Id for each news item
* headline(object) - the item's headline
* urgency(int8) - differentiates story types (1: alert, 3: article)
* takeSequence(int16) - the take sequence number of the news item, starting at 1. For a given story, alerts and articles have separate sequences.
* provider(category) - identifier for the organization which provided the news item (e.g. RTRS for Reuters News, BSW for Business Wire)
* subjects(category) - topic codes and company identifiers that relate to this news item. Topic codes describe the news item's subject matter. These can cover asset classes, geographies, events, industries/sectors, and other types.
* audiences(category) - identifies which desktop news product(s) the news item belongs to. They are typically tailored to specific audiences. (e.g. "M" for Money International News Service and "FB" for French General News Service)
* bodySize(int32) - the size of the current version of the story body in characters
* companyCount(int8) - the number of companies explicitly listed in the news item in the subjects field
* headlineTag(object) - the Thomson Reuters headline tag for the news item
* marketCommentary(bool) - boolean indicator that the item is discussing general market conditions, such as "After the Bell" summaries
* sentenceCount(int16) - the total number of sentences in the news item. Can be used in conjunction with firstMentionSentence to determine the relative position of the first mention in the item.
* wordCount(int32) - the total number of lexical tokens (words and punctuation) in the news item
* assetCodes(category) - list of assets mentioned in the item
* assetName(category) - name of the asset
* firstMentionSentence(int16) - the first sentence, starting with the headline, in which the scored asset is mentioned.
  + 1: headline
  + 2: first sentence of the story body
  + 3: second sentence of the body, etc
  + 0: the asset being scored was not found in the news item's headline or body text. As a result, the entire news item's text (headline + body) will be used to determine the sentiment score.
* relevance(float32) - a decimal number indicating the relevance of the news item to the asset. It ranges from 0 to 1. If the asset is mentioned in the headline, the relevance is set to 1. When the item is an alert (urgency == 1), relevance should be gauged by firstMentionSentence instead.
* sentimentClass(int8) - indicates the predominant sentiment class for this news item with respect to the asset. The indicated class is the one with the highest probability.
* sentimentNegative(float32) - probability that the sentiment of the news item was negative for the asset
* sentimentNeutral(float32) - probability that the sentiment of the news item was neutral for the asset
* sentimentPositive(float32) - probability that the sentiment of the news item was positive for the asset
* sentimentWordCount(int32) - the number of lexical tokens in the sections of the item text that are deemed relevant to the asset. This can be used in conjunction with wordCount to determine the proportion of the news item discussing the asset.
* noveltyCount12H(int16) - The 12 hour novelty of the content within a news item on a particular asset. It is calculated by comparing it with the asset-specific text over a cache of previous news items that contain the asset.
* noveltyCount24H(int16) - same as above, but for 24 hours
* noveltyCount3D(int16) - same as above, but for 3 days
* noveltyCount5D(int16) - same as above, but for 5 days
* noveltyCount7D(int16) - same as above, but for 7 days
* volumeCounts12H(int16) - the 12 hour volume of news for each asset. A cache of previous news items is maintained and the number of news items that mention the asset within each of five historical periods is calculated.
* volumeCounts24H(int16) - same as above, but for 24 hours
* volumeCounts3D(int16) - same as above, but for 3 days
* volumeCounts5D(int16) - same as above, but for 5 days
* volumeCounts7D(int16) - same as above, but for 7 days