StockSplosion Report

May 26, 2015

Contents

Chapter 1

Project

This project tries to create models for StockSplosion simulated market in order to assist analysts to make their buy/sell decisions easier.

1.1 Cost

This project has been rated at *$150/hr*.

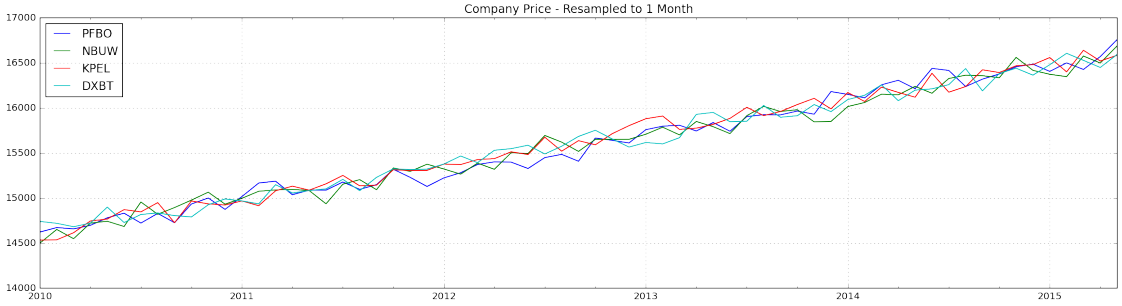
- Web Client Development: $1500
- Price Estimation Research: $7500
- Total Cost: $9000

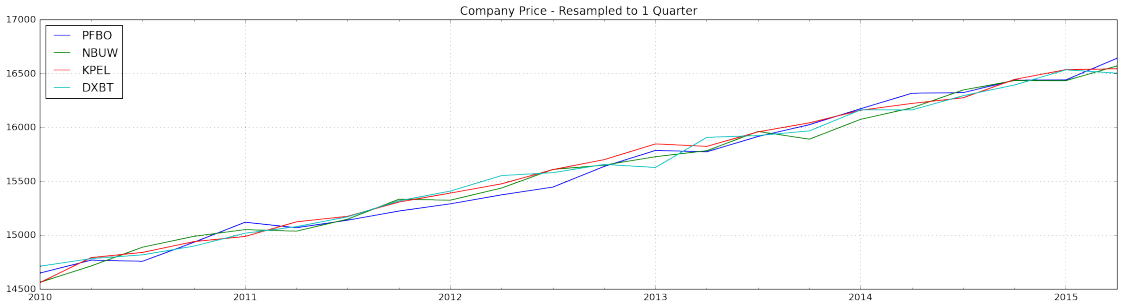
Chapter 2

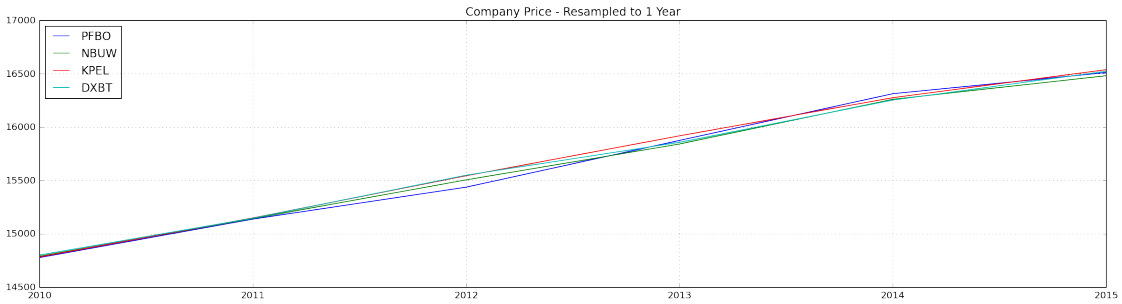
Analysis

2.1 Final Model

The dataset reacts in a specific way. Starting from Unix Epoch (1970-01-01), the price of any stock increases by 1.0 (+/- ˜10% of the rolling average). There are no seasonalities or any other anomalies in the dataset. ˜10% variation also appear to be random. In order to reduce the factor of randomness, the dataset has been resampled into 3 periods: 1 month, 1 quarter, and 1 year. As the length of a period increases, the pattern for the dataset becomes very apparent.







As it can be seen from the graphs above, the trend is always linearly upwards and there are no seasonality effects in place.

Since there are no seasonalities and the trend is always upwards linearly, the dataset fits perfectly for a *mean-reversion model*, that is the price will revert to its rolling mean over time. In real datasets, mean reversion may take a long time, thus it might not be attractive to investors. However in this dataset, the price fluctuates around and reverts to the mean all the time. Thus, we can calculate the trading range and the moving average price at any given time with very high accuracies using recent historical data.

After calculating the trading range and the moving average, we can use these to make suggestions to the analyst. If the current price is significantly higher than mean of last 30 days, it will fall in the following days, thus we suggest to *SELL*. If it is significantly lower than mean, it will rise soon, thus we suggest to *BUY*. However, if the price is close to mean, we suggest to *WAIT*.

2.2 Research

Regardless of whether the data was simulated or not, it has been treated as if it is real market data and the research has been conducted accordingly.

2.2.1 Data Preparion

Dataset Details

The dataset covers the dates between `2010-01-01` and `2015-05-15` inclusive.

The dataset has been downloaded from the provided API using 10-day range per request. Although API allows 30-day range, any errors in API would result in a loss of 30-day data. Hence, 10-day turned out to be a meaningful range, it is fast enough to download and 10-day loss in 5.5 years of data is negligible and not likely affect the outcome.

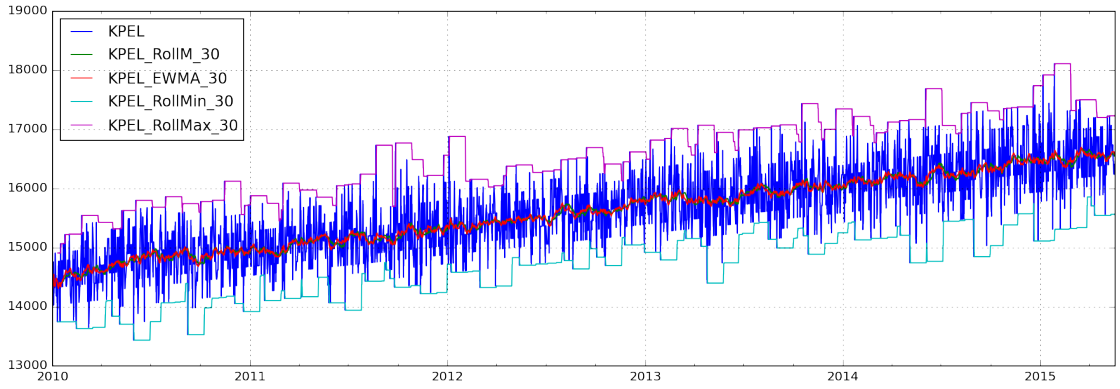
Features Extracted

Total market value (sum of all prices for any given day) has been added to dataset.

The following values are added for each company (and total market value):

	Generated Range
EWMA	3-day, 7-day, 5-day, 30-day
Lagged Price	1-day, 2-day...15-day
Lagged Price Change Direction	1-day
Percent Change	3-day, 7-day, 5-day, 30-day
Price Change	1-day, 2-day...15-day
Rolling Max	3-day, 7-day, 5-day, 30-day
Rolling Mean	3-day, 7-day, 5-day, 30-day
Rolling Min	3-day, 7-day, 5-day, 30-day
Rolling St. Dev	3-day, 7-day, 5-day, 30-day
Rolling Sum	3-day, 7-day, 5-day, 30-day
Rolling Variance	3-day, 7-day, 5-day, 30-day

The following graph from company *KPEL* visualises the price pattern. We can see the trading range, variation and the mean values of the price.



Train/Test Split

The dataset is split into two from the middle, this is because it's a time-series dataset and we haven't seen any seasonalities.

2.2.2 Cluster Analysis

During cluster analysis, no meaningful clusters have been formed. Further research might be useful.

2.2.3 Regression

Feature Selection

As a first filter to explore features and their effects on the prediction model, *SelectKBest*, *SelectPercentile* and *VarianceThreshold* models are used. This resulted that most important features are extracted features for the company and the total market. This supports previous findings on cluster analysis, no companies are acting together or affecting each other's behavior.

As a second filter to reduce the number of features for a more generalized model, *RFECV* model has been used. *RFECV*, i.e. recursive feature elimination with cross-validation, removes `n` features in each loop and decides the optimum number of features using cross-validation. For this dataset, `n` was set to 1.

Price Direction Estimation

For price direction estimation `LogisticRegression` model is used.

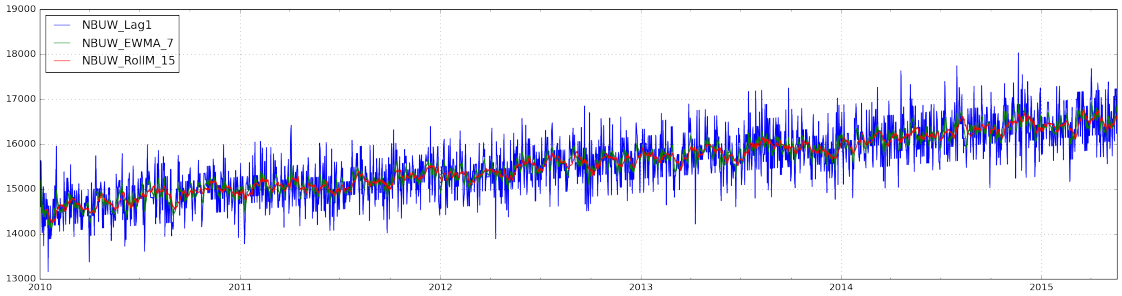
The model was able to predict with a score of `.724`.

The predicted directions were added to dataset for price estimation.

The features selected by *RFECV* are as follows:

	Feature
1	Company Lag 1-day
2	Company EWMA 7-day
3	Company Rolling Sum 15-day
4	Company Rolling Mean 15-day

The following graph shows the features chosen by *RFECV* (Rolling Sum feature has been omitted as it does not fit into this graph.)



Price Estimation

For price estimation, `LinearRegression`, `Ridge`, `RidgeCV` models are trained. Although the scores of models were pretty close, `Ridge` model with alpha `10` yielded more generalized results than the rest.

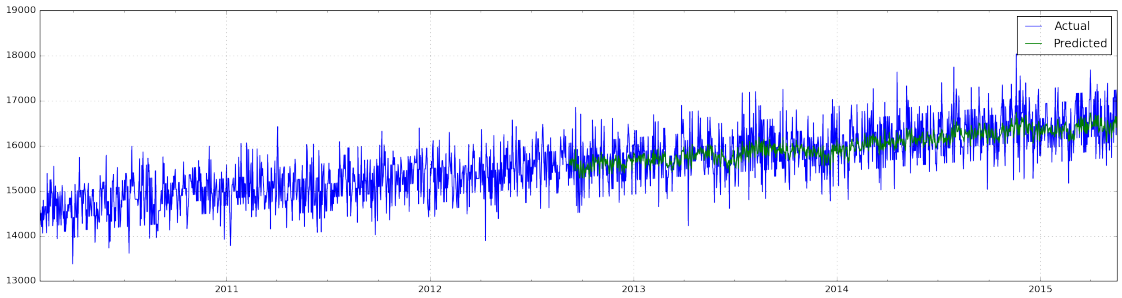
	Actual Stdev	Mean Sum of Square Errors	Score
NBUW	572.6	154635	0.528

The features that yielded these results are as follows:

	Feature
1	Company Percent Change 7-day
2	Company EWMA 7-day
3	Company EWMA 30-day
4	Total Percent Change 7-day
5	Total Rolling Sum 15-day
6	Total Rolling Variance 15-day
7	Total Percent Change 30-day
8	Total Rolling Sum 30-day
9	Total Lag 26

It is important to note that the predicted direction feature has been eliminated by *RFECV*. It might be because it does not affect the model outcome, but further research is needed for a conclusion.

The following prediction graph shows us the predicted values fluctuate around the moving average.



2.2.4 Conclusion

As we have seen from the examples and charts above, the price of a company as an upward trend with 10% randomness along its moving average mean.

In order to create a suggestion model for buy/sell indicators, we can fit our data in *mean-reversion model*. The price fluctuates around it's moving average , but it never goes beyond a certain limit.

Thus, we can rules for our model as follows (significance factor used in this model is `0.015`): - If the current price is significantly higher than its moving average, suggest *SELL* - If the current price is significantly lower than its moving average, suggest *BUY* - If the current price is insignificant / close its moving average, suggest *WAIT*.

Using this model, we can safely and confidently indicate BUY/SELL/WAIT for any company in the market.