Machine Learning Engineer Nanodegree

Capstone Project

Ezhilvendhan V  
January 13th, 2019

I. Definition

(approx. 1-2 pages)

Project Overview

Investors of stocks increasingly rely on news to buy and sell stocks. So, it naturally makes sense to correlate stock prices with news sentiment. However, not all the data that are available to us are relevant. We need to find out only data which strongly correlates with stock prices. Understanding this correlation has a huge potential in predicting financial outcomes and has tremendous financial impact.

This problem is from a Kaggle competition sponsored by [Two Sigma](https://www.twosigma.com/), a scientifically driven Investment Manager.

Problem Statement

This is a regression problem. From the news sentiment for a given stock, the corresponding stock price has to be predicted.

I intend to use [ExtraTreesClassifier](https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.ExtraTreesClassifier.html) algorithm as baseline to predict stock prices. Light Gradient Boosting Model (LightGBM) will be used as the main model to predict stock prices. Performance of these two models will be compared.

The following tasks are involved:

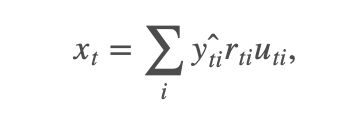
1. Exploratory data analysis of market and news data
2. Pre-processing of market and news data
3. Concatenation of market and new data frames
4. XGB base line modelling and performance evaluation
5. LightGBM modelling and performance evaluation
6. Comparison of LightGBM and the baseline model performance

Metrics

The metrics will be based as below5:

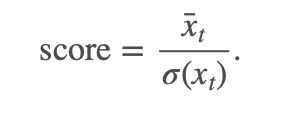
In this competition, signed confidence value, ŷti ∈ [−1,1], is predicted which is multiplied by the market-adjusted return of a given assetCode over a ten day window. For a stock which is expected to perform well, a positive confidenceValue (near 1.0) will be assigned. For low performing stocks, negative confidenceValue (near -1.0) will be assigned. If unsure, a value near zero will be assigned.

For each day in the evaluation time period, upon submission, the return will be calculated as:



where rti is the 10-day market-adjusted leading return for day t for instrument i, and uti is a 0/1 universe variable (see the data description for details) that controls whether a particular asset is included in scoring on a particular day.

Submission score is then calculated as the mean divided by the standard deviation of your daily xt values:



If the standard deviation of predictions is 0, the score is defined as 0.

II. Analysis

(approx. 2-4 pages)

Data Exploration

Two types of data sets3 will be used. The datasets can be accessed from only a Kaggle kernel.

1. Market data provided by [Intrino](https://intrinio.com/)

The data includes a subset of US-listed instruments. The set of included instruments changes daily and is determined based on the amount traded and the availability of information. This means that there may be instruments that enter and leave this subset of data. There may therefore be gaps in the data provided, and this does not necessarily imply that that data does not exist (those rows are likely not included due to the selection criteria).

The market data contains a variety of returns calculated over different timespans. All of the returns in this set of market data have these properties:

* 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.

Columns in the dataset:

* 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.

1. News data provided by [Thomson Reuters](https://www.thomsonreuters.com/en.html)

The news data contains information at both the news article level and asset level

* 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

Exploratory Visualization

Market Data

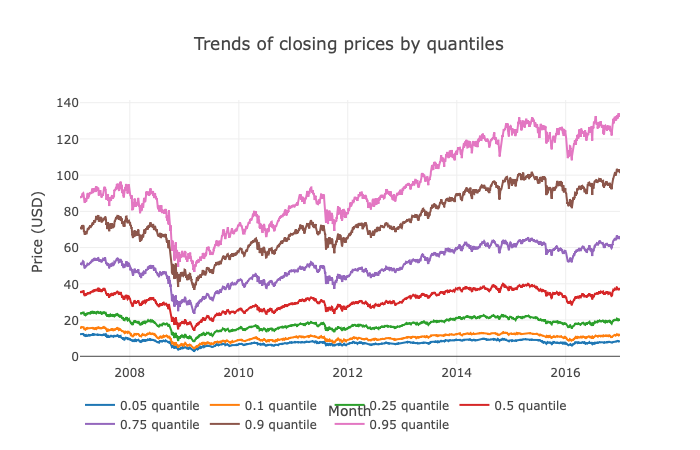
*Closing prices of 10 random assets*

This is to understand the long-term trends of random stocks. As we can see, some stocks started trading later while some others got discontinued.



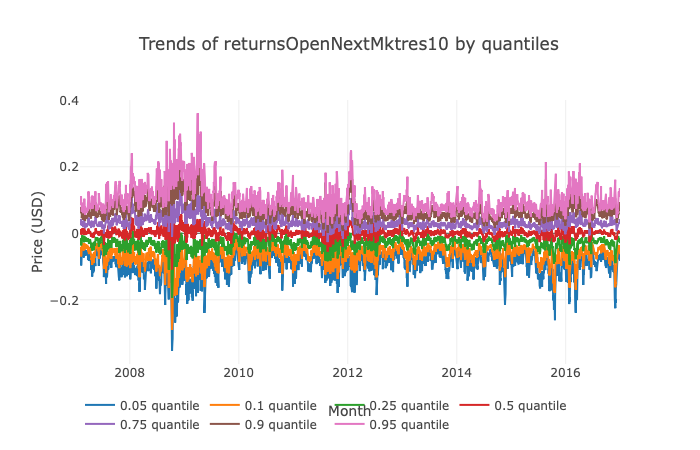
*Trends of closing prices by quantiles*

This plot shows four events where stock prices of all companies across the quantiles are affected.



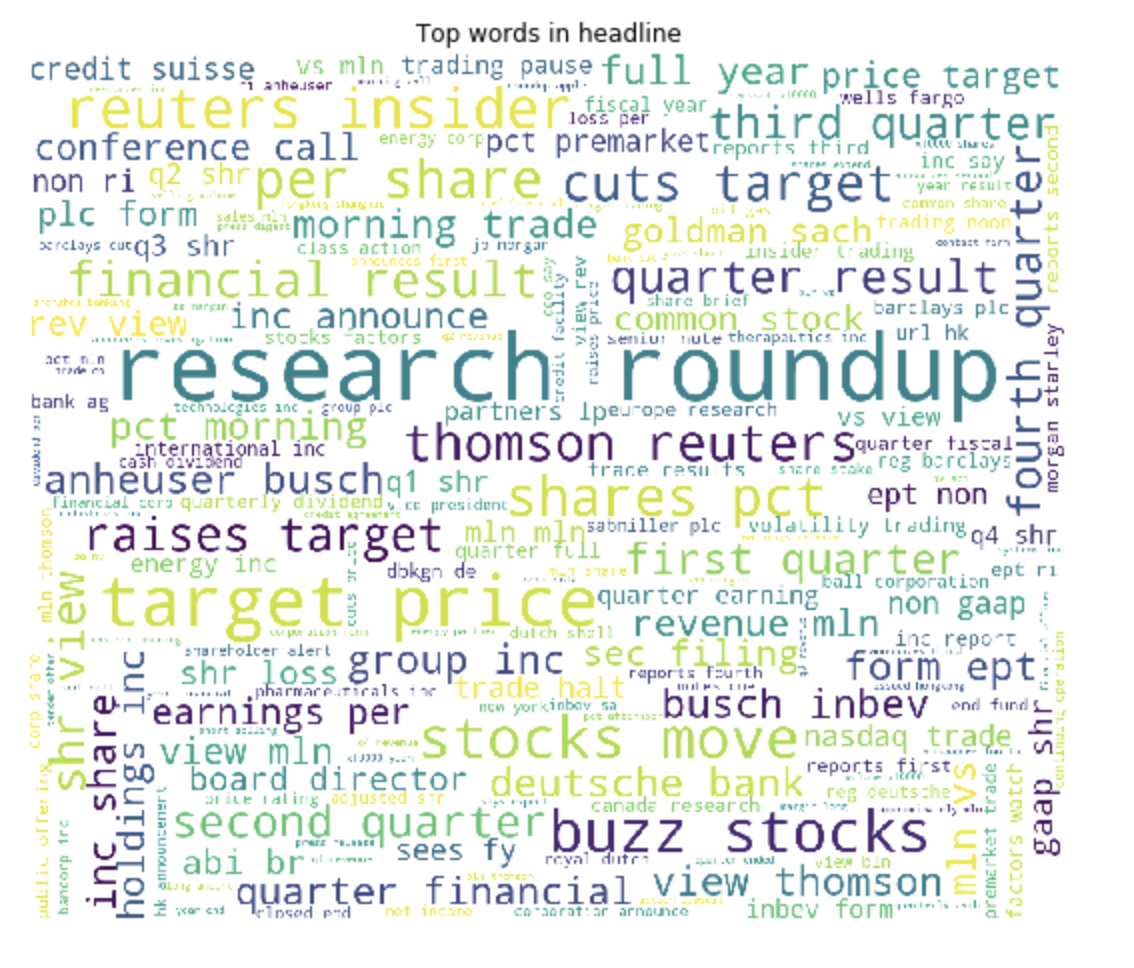
*Trends of returnsOpenNextMktres10 by quantiles*

As we can see, there is tremendous noise with the 2009 data due to recession. Rest of the data seem less noisy



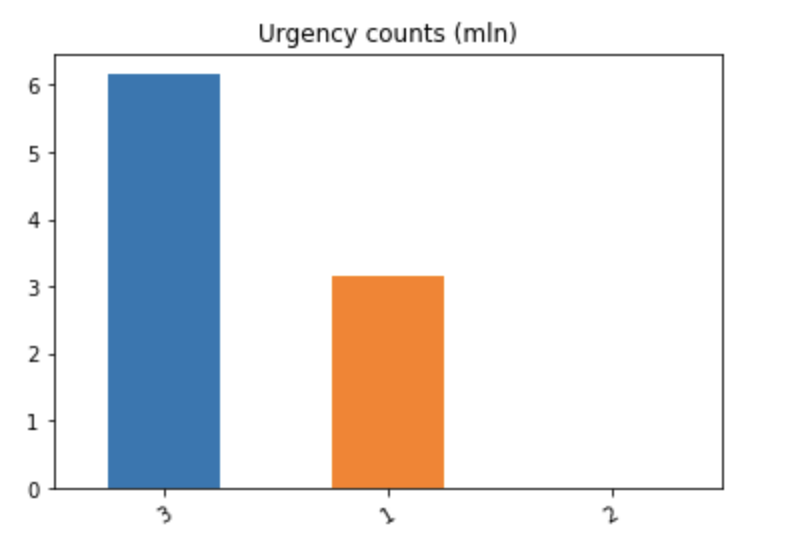
News Data

*Top words in headline*

**

*Urgency Counts*

Urgency 2 is never used as we can in the plot below



*Sentiment and companies*

Apple has the most positive and negative sentiments.

Top mentioned companies for negative sentiment are:

Apple Inc 22518

JPMorgan Chase & Co 20647

BP PLC 19328

Goldman Sachs Group Inc 17955

Bank of America Corp 17704

Name: assetName, dtype: int64

Top mentioned companies for neutral sentiment are:

HSBC Holdings PLC 19462

Credit Suisse AG 14632

Deutsche Bank AG 12959

Barclays PLC 12414

Apple Inc 10994

Name: assetName, dtype: int64

Top mentioned companies for positive sentiment are:

Apple Inc 19020

Barclays PLC 18051

Royal Dutch Shell PLC 15484

General Electric Co 14163

Boeing Co 14080

Algorithms and Techniques

[Light GBM](https://lightgbm.readthedocs.io/en/latest/Parameters-Tuning.html) is a gradient boosting framework that uses tree based learning algorithm. It is designed to be distributed and efficient with the following advantages:

* Faster training speed and higher efficiency.
* Lower memory usage.
* Better accuracy.
* Support of parallel and GPU learning.
* Capable of handling large-scale data.

The following parameters will be tuned and the optimum value will be found using [scikit-optimize](https://scikit-optimize.github.io/)’s gp\_minimize library.

1. min\_data\_in\_leaf – Minimum number of data in one leaf. This is a very important parameter to prevent over-fitting in a leaf-wise tree. Its optimal value depends on the number of training samples and num\_leaves. Setting it to a large value can avoid growing too deep a tree, but may cause under-fitting. In practice, setting it to hundreds or thousands is enough for a large dataset.
2. num\_iteration
3. max\_bin – Maximum number of bins that feature values will be bucketed in
4. learning\_rate
5. num\_leaves - This is the main parameter to control the complexity of the tree model. Denotes max number of leaves in one tree.

We will be using these parameters and their values as well:

1. task = ‘train’
2. boosting\_type = ‘dart’ [Dropouts meet Multiple Additive Regression Trees](https://arxiv.org/abs/1505.01866)
3. objective = ‘binary’

Benchmark

[ExtraTreesClassifier](https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.ExtraTreesClassifier.html) based model was considered as baseline model. R2 score, Mean absolute error, mean squared error and root mean squared error of this model is compared against that of the Light GBM model’s.

Baseline

R2 score: -1.0101129310399442

Mean Absolute Error: 0.5021990402953773

Mean Squared Error: 0.5021990402953773

Root Mean Squared Error: 0.7086600315351341

III. Methodology

Data Preprocessing

Due to the noise created by the 2009 recession data, I decided to take the market and news data only from 2010.

* The following columns were removed from the news data as they were not very helpful:
  + audiences
  + subjects
  + assetName
  + headline
  + firstCreated
  + sourceTimestamp
* assetCodes data was cleanedup to remove {} and single quotes.
* assetCodes column is comma separated. It’s unstacked and split in to individual codes. This will assist in merging the market and news data
* News Data is then grouped by assetCode and date columns and aggregated by mean.
* Market and news data are merged using assetCode and date

Implementation

In this section, the process for which metrics, algorithms, and techniques that you implemented for the given data will need to be clearly documented. It should be abundantly clear how the implementation was carried out, and discussion should be made regarding any complications that occurred during this process. Questions to ask yourself when writing this section:

* Is it made clear how the algorithms and techniques were implemented with the given datasets or input data?
* Were there any complications with the original metrics or techniques that required changing prior to acquiring a solution?
* Was there any part of the coding process (e.g., writing complicated functions) that should be documented?

Refinement

In this section, you will need to discuss the process of improvement you made upon the algorithms and techniques you used in your implementation. For example, adjusting parameters for certain models to acquire improved solutions would fall under the refinement category. Your initial and final solutions should be reported, as well as any significant intermediate results as necessary. Questions to ask yourself when writing this section:

* Has an initial solution been found and clearly reported?
* Is the process of improvement clearly documented, such as what techniques were used?
* Are intermediate and final solutions clearly reported as the process is improved?

IV. Results

(approx. 2-3 pages)

Model Evaluation and Validation

In this section, the final model and any supporting qualities should be evaluated in detail. It should be clear how the final model was derived and why this model was chosen. In addition, some type of analysis should be used to validate the robustness of this model and its solution, such as manipulating the input data or environment to see how the model’s solution is affected (this is called sensitivity analysis). Questions to ask yourself when writing this section:

* Is the final model reasonable and aligning with solution expectations? Are the final parameters of the model appropriate?
* Has the final model been tested with various inputs to evaluate whether the model generalizes well to unseen data?
* Is the model robust enough for the problem? Do small perturbations (changes) in training data or the input space greatly affect the results?
* Can results found from the model be trusted?

Justification

In this section, your model’s final solution and its results should be compared to the benchmark you established earlier in the project using some type of statistical analysis. You should also justify whether these results and the solution are significant enough to have solved the problem posed in the project. Questions to ask yourself when writing this section:

* Are the final results found stronger than the benchmark result reported earlier?
* Have you thoroughly analyzed and discussed the final solution?
* Is the final solution significant enough to have solved the problem?

V. Conclusion

(approx. 1-2 pages)

Free-Form Visualization

In this section, you will need to provide some form of visualization that emphasizes an important quality about the project. It is much more free-form, but should reasonably support a significant result or characteristic about the problem that you want to discuss. Questions to ask yourself when writing this section:

* Have you visualized a relevant or important quality about the problem, dataset, input data, or results?
* Is the visualization thoroughly analyzed and discussed?
* If a plot is provided, are the axes, title, and datum clearly defined?

Reflection

In this section, you will summarize the entire end-to-end problem solution and discuss one or two particular aspects of the project you found interesting or difficult. You are expected to reflect on the project as a whole to show that you have a firm understanding of the entire process employed in your work. Questions to ask yourself when writing this section:

* Have you thoroughly summarized the entire process you used for this project?
* Were there any interesting aspects of the project?
* Were there any difficult aspects of the project?
* Does the final model and solution fit your expectations for the problem, and should it be used in a general setting to solve these types of problems?

Improvement

In this section, you will need to provide discussion as to how one aspect of the implementation you designed could be improved. As an example, consider ways your implementation can be made more general, and what would need to be modified. You do not need to make this improvement, but the potential solutions resulting from these changes are considered and compared/contrasted to your current solution. Questions to ask yourself when writing this section:

* Are there further improvements that could be made on the algorithms or techniques you used in this project?
* Were there algorithms or techniques you researched that you did not know how to implement, but would consider using if you knew how?
* If you used your final solution as the new benchmark, do you think an even better solution exists?