Machine Learning Engineer Nanodegree

Capstone Project

Ezhilvendhan V
January 13th, 2019

I. Definition

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, 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 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 below[5]:

In this competition, signed confidence value, $\hat{y}_{ti} \in [-1,1]$, 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:

$$x_t = \sum_i \hat{y_{ti}} r_{ti} u_{ti},$$

where r_{ti} is the 10-day market-adjusted leading return for day t for instrument i, and u_{ti} 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 x_t values:

$$\text{score} = \frac{\bar{x}_t}{\sigma(x_t)}.$$

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

II. Analysis

Data Exploration

Two types of data sets[3] will be used. The datasets can be accessed from only a Kaggle kernel.

1. Market data provided by Intrino

   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.

2. News data provided by Thomson Reuters

   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

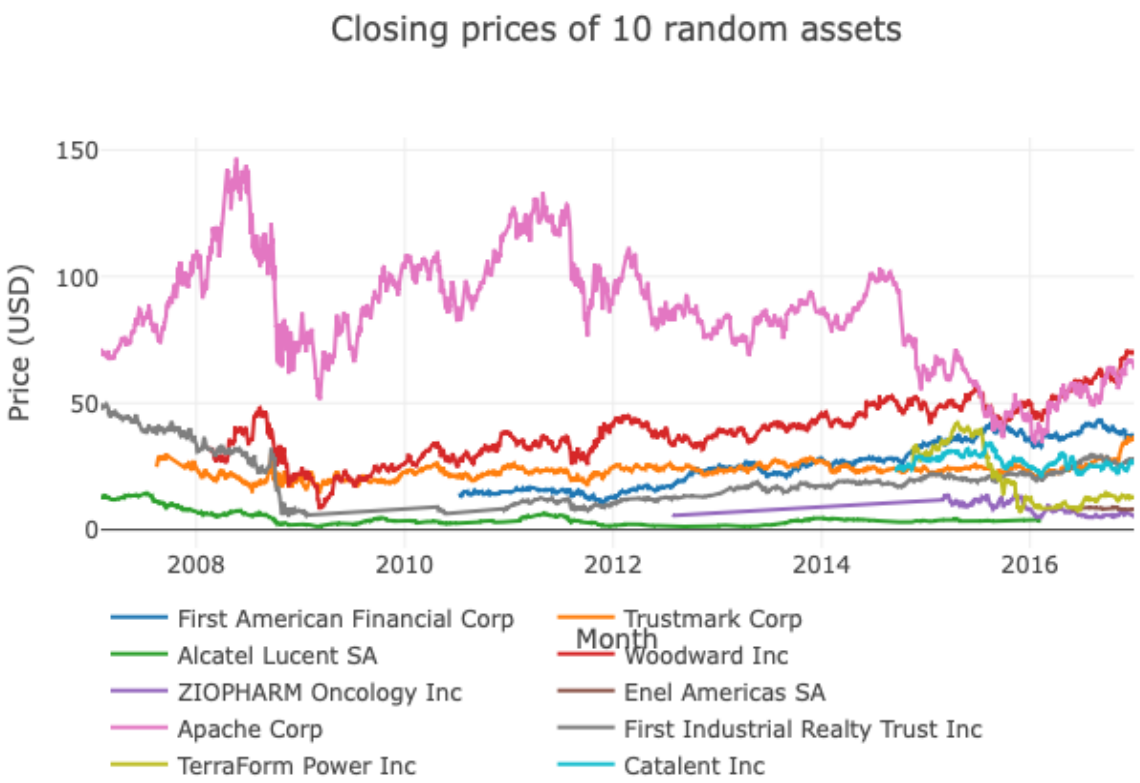
- volumeCounts12H(int16) - the 12 hour volume of news for each asset. A cache of previous news items is maintained and the number of news items that mention the asset within each of five historical periods is calculated.
- volumeCounts24H(int16) - same as above, but for 24 hours
- volumeCounts3D(int16) - 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

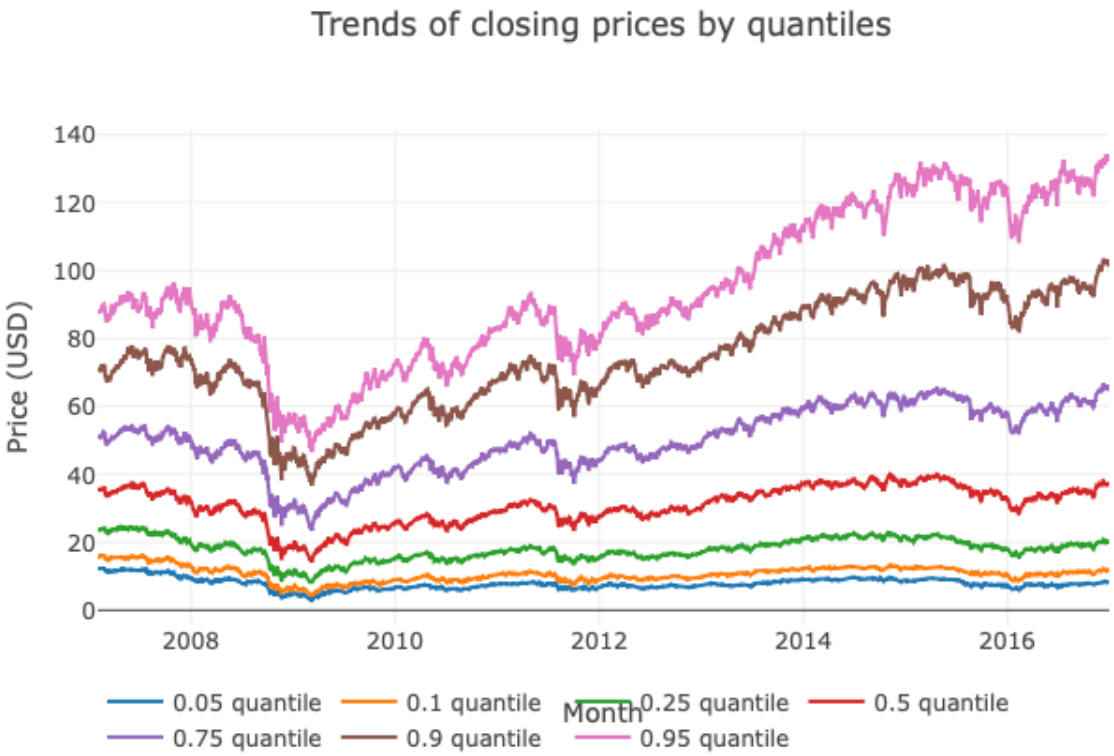
Closing prices of 10 random assets

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.



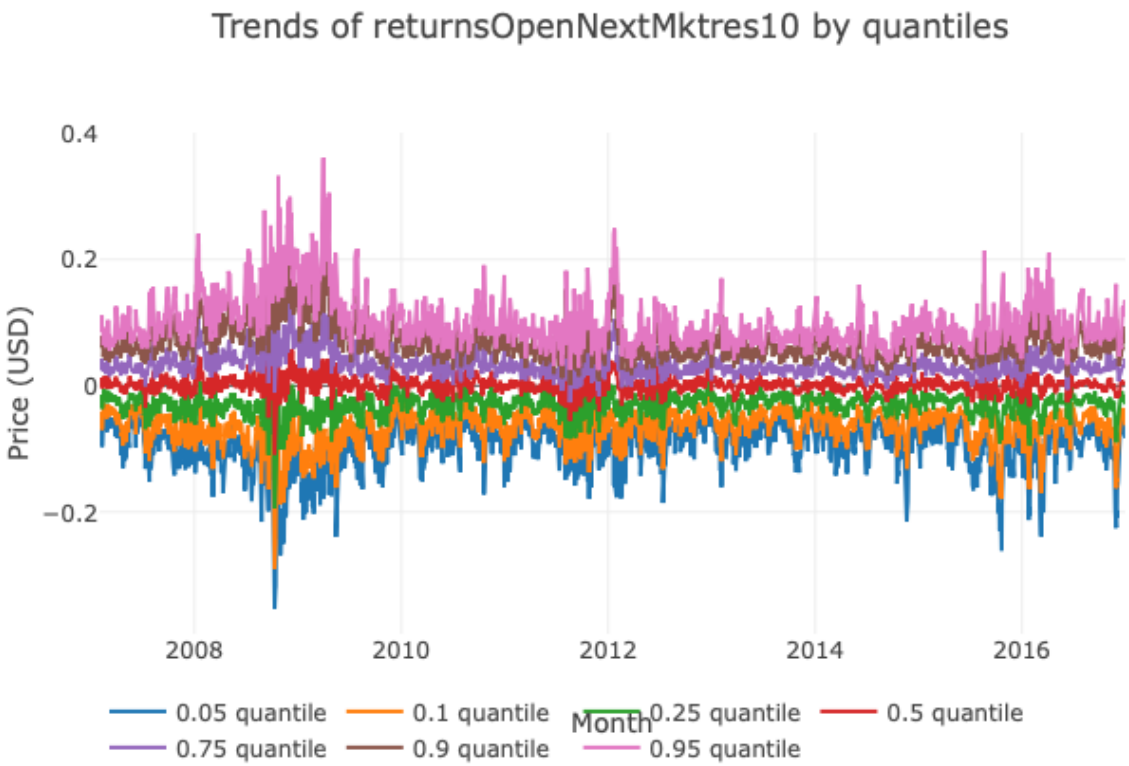
Trends of closing prices by quantiles

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

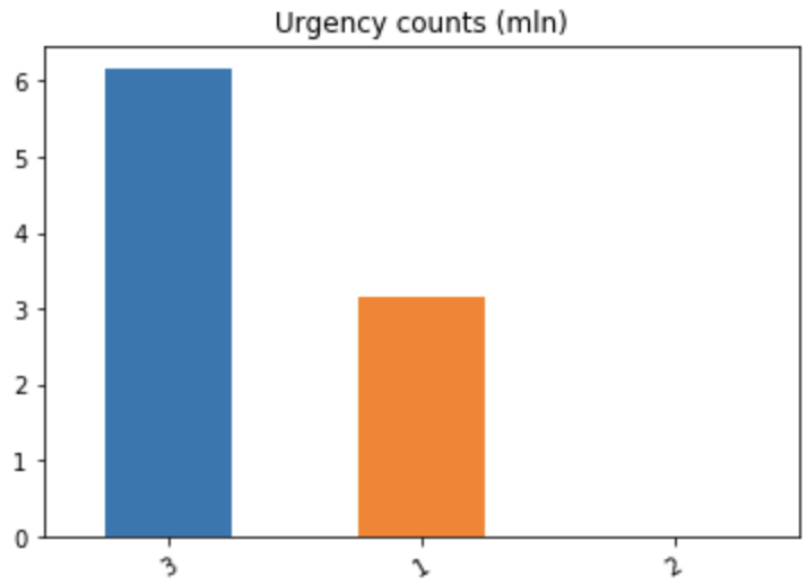


News Data

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 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'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
3. `objective = 'binary'`

Benchmark

ExtraTreesClassifier 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.

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 cleaned up 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`
- Imputed the missing data with mean values.
- Transformed the number columns except `returnsOpenNextMktres10` into float32.
- Finally, dropped the following columns from the merged data frame:
  - `returnOpenNextMktres10`
  - `date`
  - `universe`
  - `assetCode`
  - `assetName`
  - `time`

Implementation

Data is split in to test and train sets. 90% of the data goes to the training set while the remaining goes to the testing set.

Baseline model is first created using ExtraTreesClassifier. This model will be trained for 10 iterations. If the oob_score for the current iteration did not improve, then the training will be stopped. With the trained model, predictions will be tested against the testing set. R2 score, mean absolute error, mean squared error and root mean squared error are measured for this model.

The main model is created using Light GBM algorithm. Using skopt's gp_minimize library, optimal light GBM parameters are selected from a set of limits of these parameters. Mean squared error is used to decide on the optimal values.

With these optimal parameters, the light GBM model will be trained with the training set. The trained model is then used for predicting the testing set. The model is finally is measured for r2 score, mean absolute error, mean squared error and root mean squared error. These values are then compared with that of the baseline model.

Refinement

The following Light GBM parameters will be tuned:

1. min_data_in_leaf (range 260 to 300)
2. num_iteration (range 800 to 1000)
3. max_bin (range 200 to 220)
4. learning_rate (range 0.01 to 0.3)
5. num_leaves (range 3000 to 3500)

Based on trial and error, the ranges were selected and then fed to the gp_minimize library. Following optimum values of the parameters were found

```
{
 'learning_rate': 0.10415810253886905,
 'num_leaves': 3346,
 'min_data_in_leaf': 266,
 'num_iteration': 855,
 'max_bin': 217
}
```

IV. Results

Model Evaluation and Validation

The final Light GBM model performed better than our baseline model. This is evident from the metrics.

The final model was submitted to the Kaggle competition and it scored a value of 0.63. Due to competition restrictions, model could not be tested with custom or outside data. Also, the data used in scoring is not publicly available or revealed. So, I am not sure of the model generalization.

The model might not be robust enough as we eliminated the noisy 2009 recession data while training and testing. As a result, we cannot rely on it to predict all market conditions. However, the model does well under normal conditions.

Justification

Baseline metrics

```
R2 score:  -1.0101129310399442
Mean Absolute Error: 0.5021990402953773
Mean Squared Error: 0.5021990402953773
Root Mean Squared Error: 0.7086600315351341
```

Final Model metrics

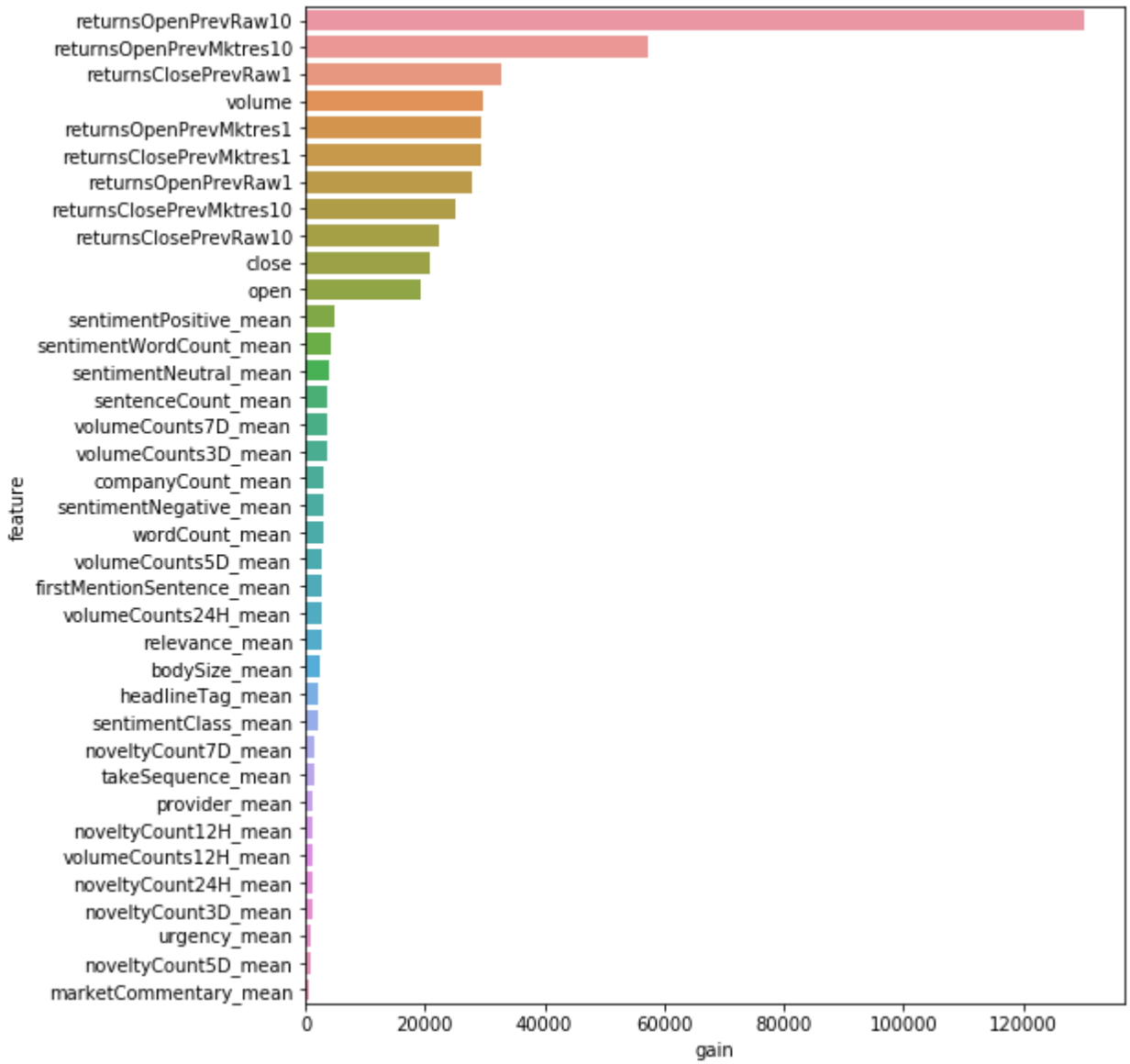
```
R2 score:  -0.015223480287422708
Mean Absolute Error: 0.49361855978573976
Mean Squared Error: 0.25363960880640946
Root Mean Squared Error: 0.5036264576116007
```

As we see, the final model performed better than the baseline model. The final model is more accurate than the baseline and more significant too.

V. Conclusion

Free-Form Visualization

With the final model, it is evident that the following variables play an important role in deciding stock prices. The model is able to make use of all these variables in predicting the target value.



Reflection

It was difficult for me to pre-process and merge the market and news data. Missing and noisy data like the recession data significantly affected the model performance. Also, it was quite challenging to decide on the important features. I had to iterate slowly to decide on them. Initially, I did some feature scaling but it did not work out well.

It was also tedious in deciding optimum parameters. gp_minimize was helpful but I was struggling with finding the right ranges of these parameters.

Improvement

There is more room for improvement as I can see from the leaderboard. I am curious to know the methodologies used by fellow competitors and understand them. May be I should try mixing multiple algorithms or multiple models with different parameters and return the average of values.

Also, I am curious to apply neural networks to solve this problem. However, I am not sure of the computing complexity.