Machine Learning Engineer Nanodegree

Capstone Proposal

Ezhil Vendhan
October 17th, 2018

Using News to predict Stock movements (from Kaggle[1] competition)

Domain Background

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 eXtreme Gradient Boosting (XGB)[2] algorithm to predict stock prices. Feature importance can be inferred from an XGB Model. Thus, we can discard unimportant features. With the important correlated features, the model can be tweaked further for more prediction accuracy.

Datasets and Inputs

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

Solution Statement

The solution is to predict the stock price based on the news sentiment and other features which are deemed important using XGB Model. Feature selection and feature importance will be done using the XGB model. Then, the hyper parameters will be tuned appropriately to improvise the model using Grid Search Cross Validation[4].

Benchmark Model

The Benchmark model will be a random prediction model. I will try to surpass this prediction using XGB algorithm.

Evaluation 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.

Project Design

Firstly, I will try to understand the datasets by Explorative Data Analysis. This analysis will include but not limited to understanding the stock trends, missing data, feature

correlation, label encoding. I will have to plot for each of these analyses. This process will take up almost 50% of the time.

During the model development phase, I plan to try out XGB, Support Vector Machines and other models. I will compare the results of these models before deciding on important features. Then, with these correlated features, I will tweak the model hyper parameters to achieve the best possible result. These steps will take up the remaining 50% of time.

Bibliography

1. https://www.kaggle.com/c/two-sigma-financial-news
2. https://machinelearningmastery.com/gentle-introduction-xgboost-applied-machine-learning/
3. https://www.kaggle.com/c/two-sigma-financial-news/data
4. http://scikit-learn.org/stable/modules/generated/sklearn.model_selection.GridSearchCV.html
5. https://www.kaggle.com/c/two-sigma-financial-news#evaluation