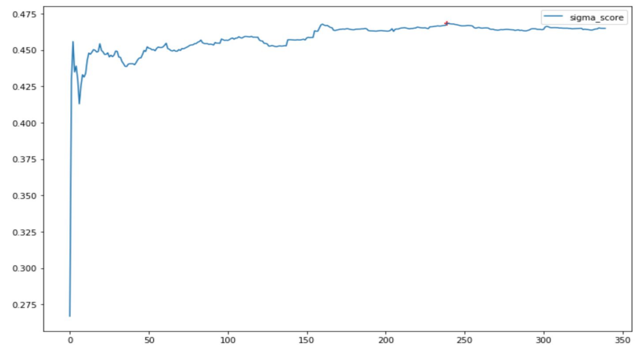
Using News to Predict Stock Movements

Group 11: *Jorge Nario, Duruvan Saravanan, Jon Hale, Nicole Mis*
{jmnario,durusarv,halej,misn15}@bu.edu

Finding optimal rounds using LGBM



1. Project Task

The goal of this Kaggle project is to predict, with an assigned confidence value [-1,1], the 10-day returns of various stocks using relevant news and sentiment features as input. Our goal is to maximize return and minimize price volatility by overweighting high confidence stocks and underweighting low confidence stocks. In January of 2019, Kaggle will use our model on live data to predict 10 day market returns. The final winners will be determined based on the results of the live testing period.

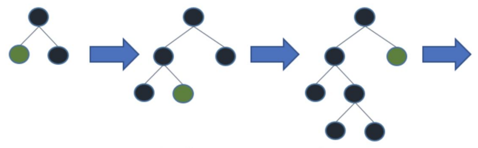
2. Related Work

Many recent studies on the stock market have implemented a wide range of machine learning techniques. One such study analyzed financial news articles and S&P 500 stock prices by applying a Support Vector Machine derivative, using Sequential Minimal Optimization [1]. The mean squared error between the predicted value and the actual stock price was used to evaluate the results. Another study also looked at the financial news but used a multi-layer perceptron with four hidden layers [2]. The ReLU activation function was used in each hidden layer and the softmax function was used in the output layer. Historical price was used as a baseline and various features from the financial news were then added to the model.

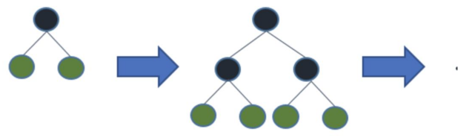
3. Approach

Originally, we tried to develop a long short-term memory model (LSTM) but this Kaggle competition has specific constraints that make this approach unfruitful (CPU, 16gb of RAM, and a 6 hour run time). From the LSTM that we created, we were getting a Sharpe ratio of about 0.55; this project has a custom evaluation metric so we used that from the given environment. This solution of 0.55 is not yielding that promising of results. So to increase our models' success, we would have to increase the depth of the network, and again the limitations are making that very complicated. The performance of the LSTM was also not optimal, taking almost the full allocated time (6 hours) to produce results, or the kernel crashes. Instead of using an LSTM, we decided to use a gradient boosting algorithm

We implemented LightGMB for our Kaggle challenge. LightGMB is a powerful, yet fast algorithm. Unlike most tree boosting algorithms, LightGBM splits the tree leaf wise while other algorithms tend to split the tree level wise or depth wise. This methodology helps to minimize loss at better rates than level-wise implementations. In addition, when the algorithm splits on levels the resulting tree requires more physical space. Thus, LightGBM tends to be faster, and more "light-weight" when compared to these similar boosting algorithms, like that of XGBOOST.



Explains how LightGBM works



The current LightGBM model we are using is yielding a Sharpe Score of approximately 0.619 (we have introduced a seed to keep it consistent). For LightGBMs, we are optimizing with the goal of creating a forward stagewise additive model. This will build a gradient descent latent space for the prediction of the future sequence, in this case, the next day. Therefore, our function is similar to that of the Black-Scholes (hopefully it will adjust to a correct gaussian distribution unlike the original Black-Scholes). This method is a greedy algorithm, which is why it's less computationally intensive. The algorithm we are using is:

$$F_m(x) = F_{m-1}(x) - \gamma_m \sum_{i=1}^{n} \nabla_{F_{m-1}} L(y_i, F_{m-1}(x_i)),$$

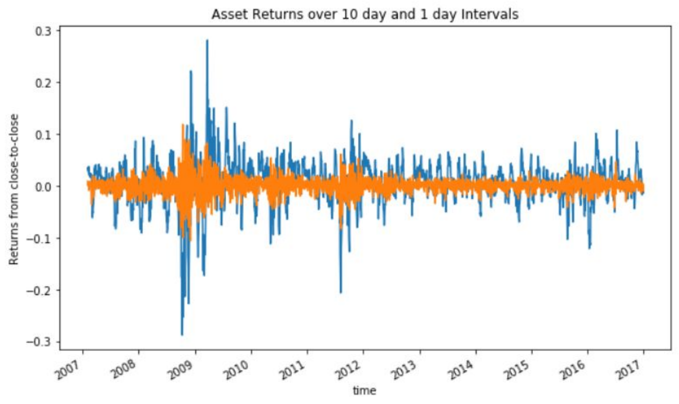
$$\gamma_m = \underset{\gamma}{\arg\min} \sum_{i=1}^{n} L\left(y_i, F_{m-1}(x_i) - \gamma \nabla_{F_{m-1}} L(y_i, F_{m-1}(x_i))\right)$$

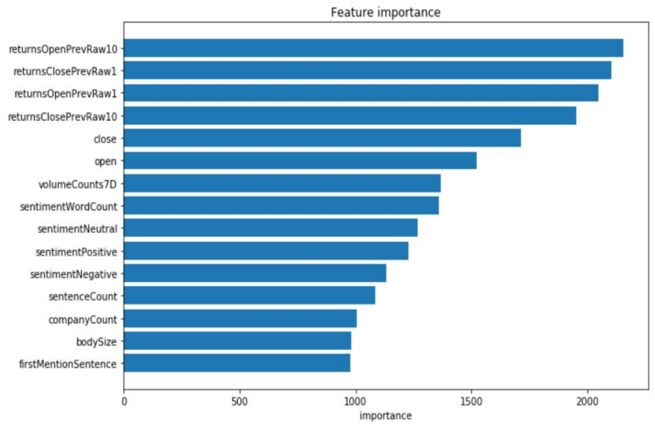
The algorithm begins with some weak model of the data ($F_{m-1}(x)$) and then builds its final prediction model by using an ensemble of these weak models. In our above equations, we compute the step size, gamma, by minimizing the predicted error of our previous model. We then plug gamma into our prediction model (first equation) to then create a new model. The simplicity of gradient boosting is that we can iteratively approximate a function that minimizes the loss of a previous weak model.

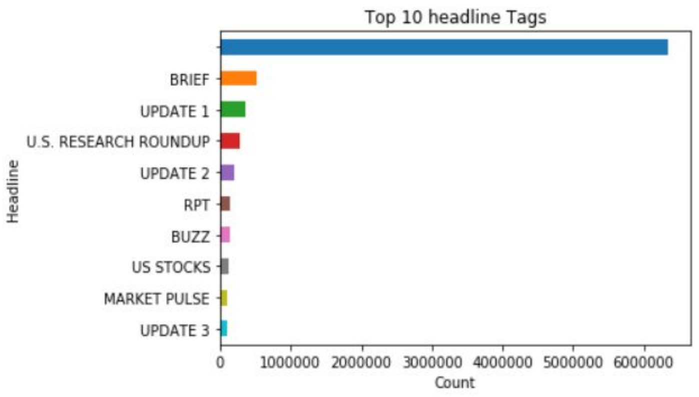
4. Dataset and Metric

Two Sigma will provide two sets of data per day (loop: predict → next day → predict …)for the time period: 01/01/2007 to present. For each historical day, we will be given one feature set containing 35 news features, including: relevant asset codes, market commentary (boolean), sentiment confidence probability functions (float32) etc. The second data set includes market return data (next 1-day and next 10-day returns). We will need to normalize the News Data where necessary. For example, "word count", "body count" and "first mention" will need to be normalized. The text features requires NLP. To do this efficiently, we are currently using the SciKit HashingVectorizer built in function to create reduced fixed dimension feature sets for each non numerical data frame column.

Our accuracy is determined by the Sharpe ratio (asset return / price volatility). In an ideal world, our goal would be to deliver a Sharpe ratio > 1, which would indicate a desirably positive risk/reward paradigm. The average S&P 500 passive Sharpe ratio (over the previous 25 years [3]) has been 1.0. Our approach is an active approach to investing and would only prove valuable if results are superior to the passive benchmark. However, given the limitations imposed by the competition and limited available data, our team's realistic goal is to best the average Sharpe score achieved in the Kaggle competition, which is currently ~0.60; this is our baseline.







5. Preliminary Results

We are in the process of running our model while varying parameters and features to see which is the most beneficial. We have visualized the data in different ways to help determine which features are most impactful. The above graph shows 10 day returns (blue) and 1 day returns (orange) for all assets from 2007 until 2017. We can see that during the financial crisis, there were stark differences in returns. Thus when building the model we would want to not include this data because these data points are outliers and we don't expect another financial crisis within the timeframe we are predicting. Additionally, we visualized some of the news features such as headline tags which is shown in the second graph above. This bar graph shows that most of the headline tags are unknown and therefore, it would probably not be helpful to include this feature in the model.

A useful feature of using gradient boosting is that the importance of each feature is calculated explicitly (graph 3) by taking the amount each attribute split point improves the performance, weighted by the number of observations the node is responsible for and then averaging this calculation across all decision trees. This is another way in which we can select features.
One of our biggest issues currently is operating within the 17GB RAM constraint. Our NLP functions require considerable memory, which causes our Kernels to crash. If we can not resolve this within the week, we may need to drop text features from the data frames.

6. Results & Analysis

LSTM

Due to the data set containing huge amounts of market and news data that the Sharpe ratio could depends on, we initially believed an LSTM model would work well due to the fact that they are capable of learning long term dependencies.

As stated earlier, the resource limitations of the competition made it difficult to adapt an LSTM model. We used three layers in our model, an LSTM layer over two dense layers. However, this model caused the kernel to crash due to overuse of the RAM capacity. Reducing the model to a single layer did not change the results.

In an effort to prevent the kernel from crashing, we decided to use just the market data and drop the news data entirely from training. This stopped the kernel from crashing, but as expected, the resulting Sharpe ratio was well below the Kaggle average.

Ultimately, we decided to drop our LSTM model altogether due to the hardware limitations that made it challenging to incorporate news data in our model.

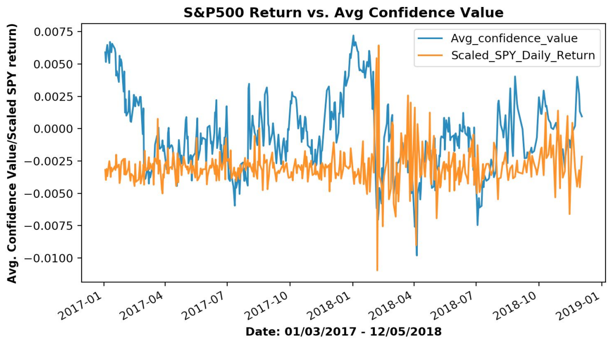
GBM

We achieved best results using the LightGBM model architecture. Our highest, and final, sigma evaluation metric and Sharpe ratio was **0.619**, which surpassed our initial self-imposed benchmark of **0.600** (the then avg. Kaggle score). However, we are currently placed 1,196 out of 2,165 Kaggle teams, which is a **55th** percentile rank. Evidently, though we managed to improve upon our initial models (which performed in the 0.40-0.55 Sigma score range), fellow teams improved their scores in lockstep.

Regardless, we are pleased with our model performance, especially given the strict competition limitations (single CPU, 6hr run time). While iterating our models, we discovered that the single CPU was the primary resource constraint. Given lack of computing power, our NLP processing continually failed, We were unable to resolve this issue and were forced to scratch advanced NLP from the final GBM model iteration.

To achieve our results, we fine tuned our Gradient Boosting Machine by manipulating parameters in an iterative process. We used a learning rate of 0.1, which believe provides an optimal blend of speed and accuracy. Though a slightly lower learning rate may improve accuracy, our CPU constraints kept us at >=0.1.

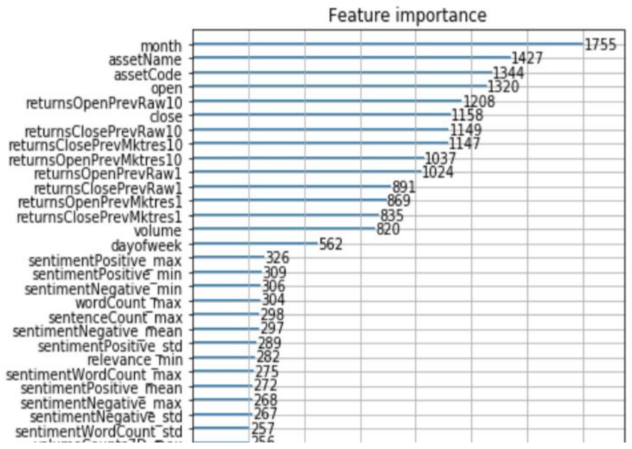
Our model's output data file consists of the set of our predicted confidence values for the investment universe (all assets) for the date range beginning 2017-01-03 to present. Using this output in conjunction with data acquired through the IEX Trading API, we constructed the following chart, which shows our average predicted confidence value vs. the daily market return..



The blue line is our average predicted confidence value while the orange line is the scaled S&P 500 daily market return (min-max scaled to coincide with the min-max of the avg. predicted confidence value). As can be seen, our predicted confidence value spikes often several months before market return spikes (specific example at 2017-12 and 2018-08). As desired, our model seems to predict market spikes before they occur and may be useful for investors (providing a holding period of >2-3 months is acceptable).

As can also be seen from this chart, a high average predicted confidence value is generally followed by market spikes in the 1-3 month range. Thus, we conclude that the market response to positive news is delayed. However, when our model predicts low confidence values, the market deteriorates apparently immediately. We accordingly conclude that the market is highly reactive to negative news and negative sentiment.

We generated the following chart to visualise our model's prescribed feature importance.



The most feature most correlated to short term 10 day market returns is the month in question. This could be due to seasonal patterns associated with corporate earnings calendars (when impactful news flow tends to increase). Previous 10-day returns are the fifth most important feature, which indicates that past short-term returns tend to persist. With regard to pure news flow data (as opposed to market and datetime data), sentiment metrics proved to be the most important model features, followed by headline word count.

7. Roles

Jon Hale - Finance Researcher/Data Pre-processing
Jorge - Model Architect I
Duruvan Saravanan - Model Architect II
Nicole Mis - Model Designer and Testing Lead

Task	**Deadline**	**Lead**
Finish tuning parameters for model	12/3/18	all
Final model results	12/5/18	all
Prepare report and presentation	12/9/18	all

References

1) R. Schumaker and H. Chen. Textual Analysis of Stock Market Prediction Using Breaking Financial News: The AZFinText System. ACM Digital Library, 27(2), 2009.
2) Y. Peng and H. Jiang, Leverage Financial News to Predict Stock Price Movements Using Word Embeddings and Deep Neural Networks. arXiv.org, 2015.
3) https://6meridian.com/2017/11/could-the-current-sharpe-ratio-for-the-sp-500-be-a-signal-of-things-to-come/