Machine Learning Engineer Nanodegree

Capstone Proposal

George Vargas October 2018

Proposal

Domain Background

Accurately predicting the financial markets has long been the holy grail for investors of all types. Individual investors seeking this grail often fail due to an underestimation of the extreme complexity of the financial markets. Though many fail, some individual investors do attain moderate success implementing trading systems that work for the highly focused conditions in which they find themselves trading. Likewise, some hedge funds, such as Renaissance Technologies, achieve great success building complex trading systems based on standard and alternative data whilst utilizing much more scientific methods for decision making.[1]

Approaching the problem as a hedge fund might, the success rate of any trade can be significantly increased by considering the system as components all feeding a decision engine, much like neurons in a neural network feed the output layer. While these systems are made of many complex components, there has been an increasing focus on alternative data and its use in sentiment analysis components. It has been proven that the data mining of news articles, blogs, and Twitter posts increases the probability of predicting market movements.[2][3]

My personal motivation for looking at possible solutions to this task is purely for my own financial independence. As a hobbyist day trader, I've been profitable but given the time constraints involved in working full-time, the creation of an autonomous trading system would be an extremely valuable step toward achieving financial independence.

Problem Statement

The problem with financial markets is that they are largely driven by emotions. Even in a time when a majority of the trades are being placed by sophisticated bots, emotion still rules the market. This can be seen by the effect that Donald Trump's tweets have on the stock of the particular companies he targets. Utilizing news headlines to reliably predict these types of market movements will be the problem this capstone attempts to tackle.

Datasets and Inputs

The datasets used are from *Kaggle*, specifically a dataset provided by a user in the community.[4] The datasets span approximately eight years from August 2008 to July 2016. Contained is a dataset of the Dow Jones Industrial Average price history with columns including Date, Open, High, Low, Close, Volume, and Adj Close; a dataset of Reddit news and a combined dataset with a Dow Jones Industrial Average label and a column for each one of the top twenty-five user upvoted headlines for that day. The label has a value of either "1" for the DJIA closing up and "0" for the DJIA closing down. The provider of the dataset recommends using data from 2008-08-08 to 2014-12-31 for training, and data from 2015-01-02 to 2016-07-01 for testing since this splits the overall dataset in an 80/20 fashion.

For our purposes, data points leading up to April 2009 will be excluded from training. The reason for excluding this data in training and testing is because the market was in a strong downward trend that was a direct result of the housing market crash. Aside from training and testing for correlation between news headlines and the DJIA closing up or down, the data leading up to April 2009 will be reserved for additional testing to determine if the model can generalize from being trained on only bull market data to succeed in a predominantly bear market.

Solution Statement

In this Capstone, an attempt will be made to predict whether the Dow Jones Industrial Average will close up or close down based on text classification tasks performed on the news headlines of that day. Text will be converted into embeddings which will then be fed to a supervised classification model. Our prediction value will be the label from the combined dataset denoting the upward or downward movement of the DJIA for a given day.

Benchmark Model

Although anecdotal, most individual investors will cite roughly a fifty to fifty-five percent accuracy when trading. This is mainly gathered from watching students in trading programs or from discussions with other individual investors. Anecdotal information isn't necessarily considered reliable, however, this researcher is inclined to believe such claims as most trading programs require students to methodically write down every trade(entry, exit, stop-loss, etc.) before placing the order. This is done to develop discipline and help remove emotion from the trades which is one of the most important factors for becoming a successful individual investor.

Now, fifty to fifty-five may seem like slim margins, but keep in mind that with just a fifty-one win percentage and proper risk management, card counters can effectively clean house at casinos. Also considering that the data spans a period of time where the market was particularly bullish, the dummy classifier provided by scikit-learn will be employed to benchmark against the multiple simple strategies provided by the dummy classifier.[5] This will help figure out if the bullish bias in the data can be exploited via simpler means.

Evaluation Metrics

To get a better idea of what is going on with the model, a few different metrics will be utilized to evaluate the performance against the benchmark model. Firstly, a confusion matrix and an accuracy score will be used to provide a general idea of the model's performance. This will be followed up by a dive into the precision, recall, and F1 score for the model. These will help us to gauge whether or not the model is overfitting or underfitting, and if its predictions are close to the actuals.

Project Design

First, the dataset will be split to create the aforementioned bearish holdout set. After which the rest of the data will be split into two portions; one for training and the other for testing.

In order to be able to feed the text from the dataset to a classification model, some preprocessing is required. Based on preliminary research, the most effective method currently used for text classification is the conversion of words and sentences to vectors.[6] While there are several libraries that are popular for this task and provide good performance, the one chosen to be utilized is the *InferSent* library with *fastText* used for the underlying word vectorization task.[7][8]

The reason for utilizing this library over *word2vec* or *gLoVe* is that fastText creates word representations from word n-grams, referred to as skip-grams, in a convolution-like manner instead of creating the representations from entire words or sentences. This helps piece together a much more specific representation for each word, and also facilitates the construction of vectors on words that are outside of the training corpus which will be useful when generalizing against the portion of the dataset that was held out for testing generalization to the bear market of 2008. InferSent will utilize fastText to generate a vector representation for the entire sentence based on the previously generated word vectors. Since the headlines in the data are mostly single sentences or phrases, this should make it much easier to train a model that provides a high predictive accuracy.

To aid in preprocessing, a pipeline will be built to feed a raw dataset with the same schema as the initial set. The pipeline will expand contractions and return two separate datasets; one which includes stop words, and one which has been stripped of stop words. This creation of the two separate datasets is to observe any difference in performance when stop words are removed.[9] Based on the documentation, InferSent with fastText seems to be both flexible and effective without the need for heavy preprocessing. If the results of the model are very poor, the pipeline may be revisited to add more of the preprocessing that is typically found in text classification tasks.

After preprocessing the training text, the datasets will be passed into two different InferSent models for vectorization. InferSent will output a vector for each headline which will be used to populate a new dataset where the headlines have been replaced with their vector representations. Once the headlines have been converted into word embeddings, cursory analysis of the data will be conducted. This will allow for the visualization of aspects such as trends over time, distribution, frequency, popularity of headlines, etc.

Upon completion of a cursory analysis, the datasets will be fed to a support vector classifier for training. The test set will then be preprocessed, vectorized independently of the previous InferSent models, and used to predict against the trained SVC. The predictions will be used to test for overfitting, underfitting, and general performance. If the performance of the trained model is adequate, the bearish holdout set will follow the same process to determine if the model is biased to bullish markets due to the data on which it was trained.

References

[1] A. Gneushev (2014) *Pure Alpha: Story of Renaissance Technologies*
[2] J. Bollen*, H. Mao*, and X. Zeng (2010) *Twitter mood predicts the stock market.*
[3] S. Colianni, S. Rosales, and M. Signorotti (2015) *Algorithmic Trading of Cryptocurrency Based on Twitter Sentiment Analysis*
[4] Aaron7sun (2016) *Dataset: Daily News for Stock Market Prediction*
[5] scikit-learn developers (2007-2018) *DummyClassifier*
[6] Y. Li and T. Yang (2017) *Word Embedding for Understanding Natural Language: A Survey*
[7] A. Conneau, D. Kiela, H. Schwenk, L. Barrault, A. Bordes (2018) *Supervised Learning of Universal Sentence Representations from Natural Language Inference Data*
[8] P. Bojanowski*, E. Grave*, A. Joulin, T. Mikolov (2017) *Enriching Word Vectors with Subword Information*
[9] H. Saif, M. Fernandez, Y. He, H. Alani (2014) *On Stopwords, Filtering and Data Sparsity for Sentiment Analysis of Twitter*