Capstone Report

January 14, 2019

1 Capstone Project

1.1 Machine Learning Engineer Nanodegree

1.1.1 Manish Yathnalli

2 Definition

2.1 Project Overview

Stock prediction problem is one of the most interesting applications of machine learning. Entire finincial technology world is focused on solving the problem of predicting the stock value reliably. My goal originally was to use a LSTM to predict the confidance value of the stock going up or down next ten days. Given that this was a compitition in Kaggle and the restictions were that we cannot use GPU, the LSTM approach did not work that well, however I did my best.

2.2 Problem Statement

The problem is forcasting stock prices. We get historical stock and news data from 2007 to present. We need to predict a signed confidence value

$$y_{ti} \in [-1, 1]$$

which is multiplied by a market adjusted return of a given asset over a ten day window. If the algorithm thinks that the stock will go up and it is very confident, it will return a value close to 1.

2.3 Metrics

Given the above predicted $ y_{ti} $, the return will be calculated as

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

Where,
r_{ii} is the 10 day market adjusted return for day t for the instrument i
u_{ti} is a variable that controls whether a particular asset is included in scoring on a particular day. The score for submission, hence the performance of the model is decided by the standard deviation for the daily x_t values

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

Given that this is a real competition the results will be out after six months and who ever got closest to the future price in the 6 month duration will be the winner.

2.4 Data exploration

We have market data and news data. The data has to be extracted from special kaggle environment that is created for this particular competition. Market data contains usual market data like open, close, high, low, volume etc along with raw returns, previous returns and a special variable called `returnsOpenNextMktres` which is our target variable. News data contains categorical values like sentimentNegative, sentimentPositive etc and novelty values, commentry, word count etc.

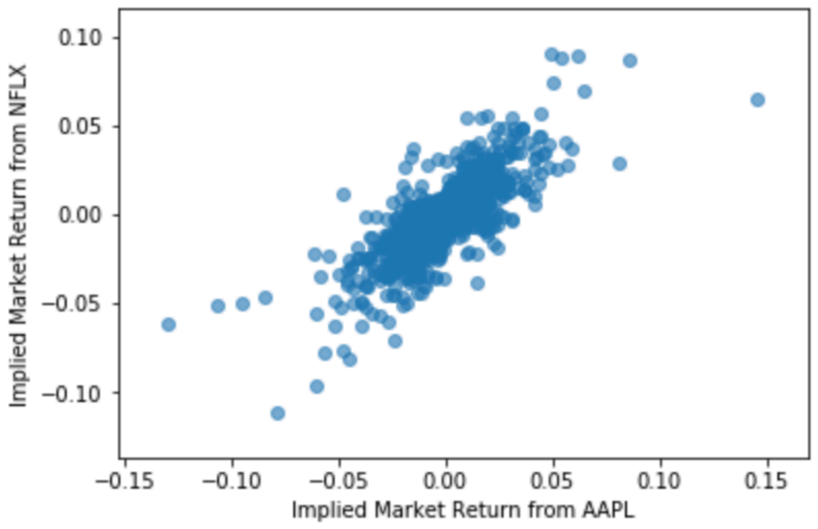
2.4.1 Understanding market data

Along with the normal open, close, high, low and volume data, there are several data columns what provide other information on the stock. I will highlight some of them in order to understand how it can impact our predictions. 1. assetCode - A unique id for an asset for example APPL.O for Apple Inc 2. universe - A Boolean indicating whether the asset will be used for scoring that day. trading universe change daily and only those which are true are used for scoring that day. 3. returnsClosePrevRaw1 - Non market adjusted return for 1 day close to close 4. returnsClosePrevMktres1 - Market adjusted return for 1 day (see target variable below) 4. returnsPrevRaw10 - 10 day non market adjusted returns.

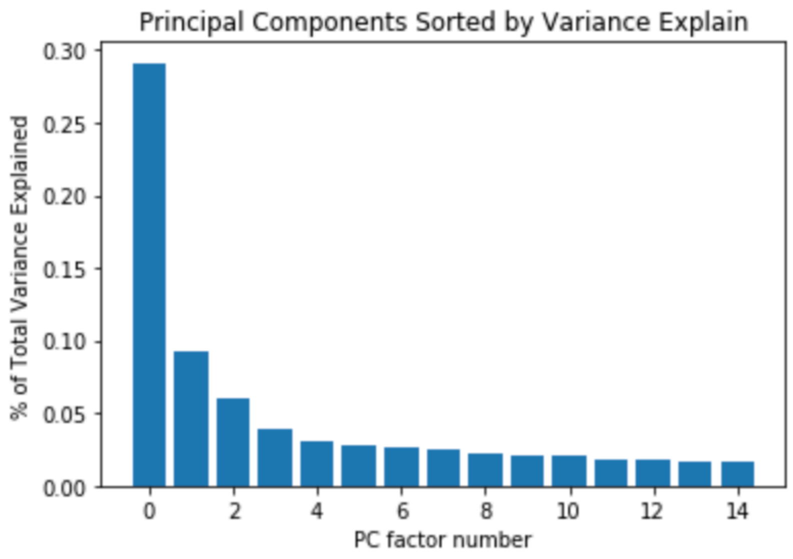
There are other returns which are along the same lines. I have used all the market data for my predictions.

2.4.2 Understanding the target variable.

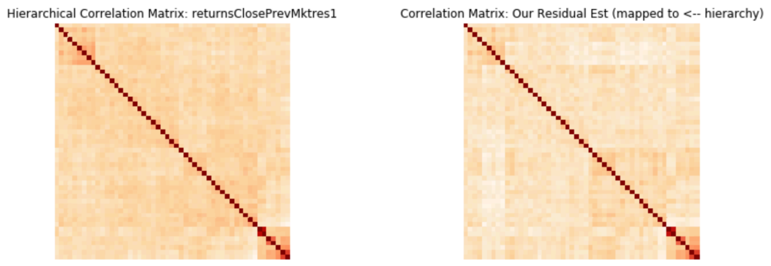
I had to first understand what is the target variable. It was not raw daily returns, which is represented by `returnsClosePrevRaw1` It was some kind of sharpe ratio per asset. It was not linearly related to raw returns. Here is a plot of raw returns vs target variable for Ap-



ple Inc. To understand what is the target variable, I did a principle component analysis of the target variable on 200 stocks over 5 years. Below are the principle components of target variable

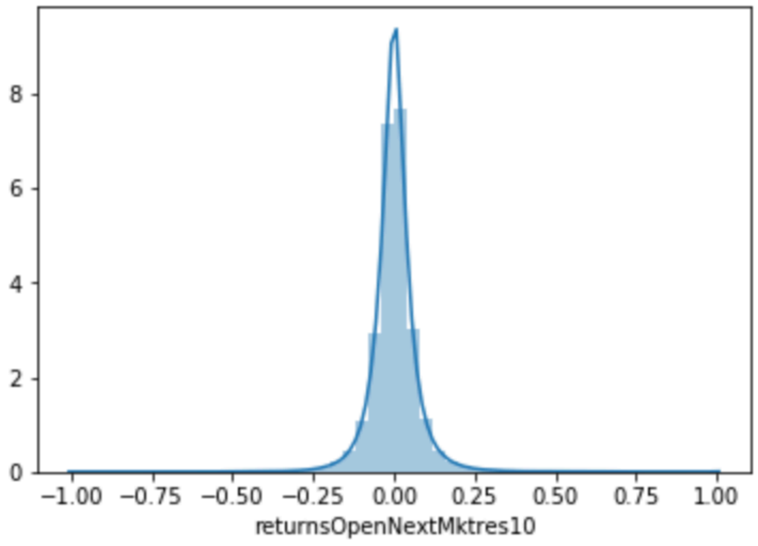


Below is the graph of co-relation of target variable after removing per stock beta adjusted market return from raw



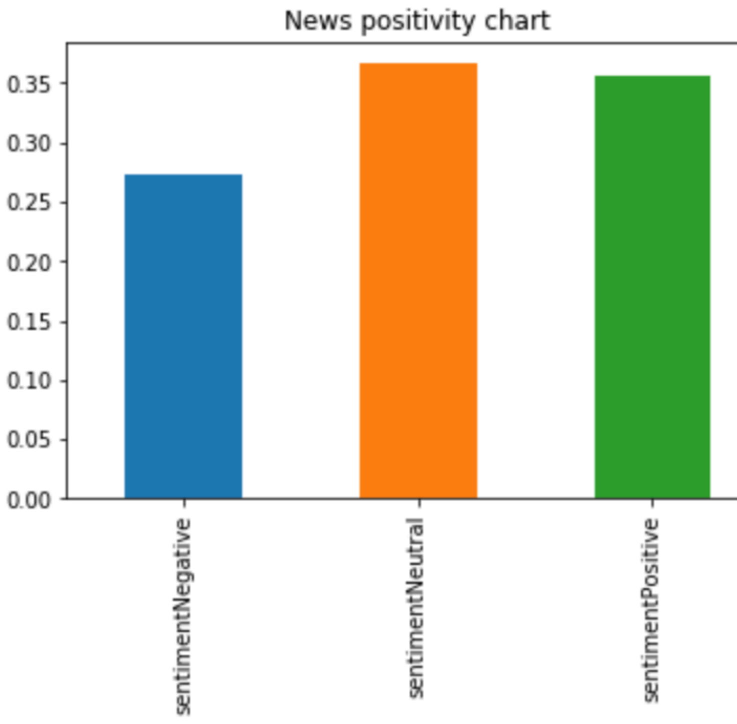
return From PCA I understood that target variable is substraction of per stock beta adjusted market return from the raw return.

We also want to look at distplot of the target variable to understand how it is spread.



2.4.3 Understanding news data.

News data has all normal news data colums like headline, provider, subjects, body size etc. Along with these, it has also some quantative alalytical data. I have listed some fo the important columns below. 1. sourceTimestamp - Time at which the news was created. 1. firstCreated - Time at which first version of the news was created. 1. urgency - 1: alert, 3: article 1. sentimentClass - Predomninantly positive or negative 1. sentimentNegative - Value of how negative the news is 1. sentimentPositive - Value of how positive the news is 1. sentimentNeutral - Value of how neutral the news is 1. noveltyCount12H - How new is the news in past 12 hours. 1. other timestamp novelty counts



Below is the bar plot of sentiment analysis.

2.5 Data preprocessing

2.5.1 Preprocessing market data

To preprocess market data I followed following steps 1. Encode the asset code 2. Split time into year, week, day, day of the week 3. pick numeric columns. (Volume, returns, open, close etc) 4. Pick features along with numeric columns 5. Scale the data using sklearn's StandardScalar 6. *Remove bad data - There were some unusally high returns in one day which were wrong.* 7. Remove target column from features. This made data sensible for a neural network to work on.

To further explain point 6, to remove unusually bad returns, I have simply dropped the rows where the price of the stock doubled and clipped the 0.1 percentile and 0.99 percentile values.

2.5.2 Preprocessing news data

1. Use numeric columns

2. scale and encode data.
3. Drop non numeric data.

It would have been better if we could do some natural language processing to convert news data into numeric data, however, that itself would have become a project

2.5.3 Merging news and market data into a single data frame

I merged the market and news data into a single data frame using time and asset code and using pandas left merge feature.

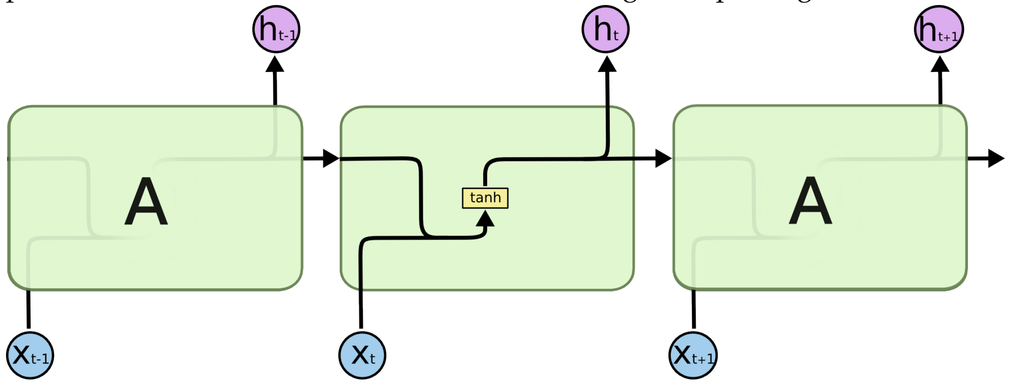
2.5.4 Creating a data generator.

Given that my plan was to use a LSTM network for predictions, I created a data generator for keras. This would feed the data batch wise into LSTM network

$$LSTMAlgorithm$$

2.6 Algorithms

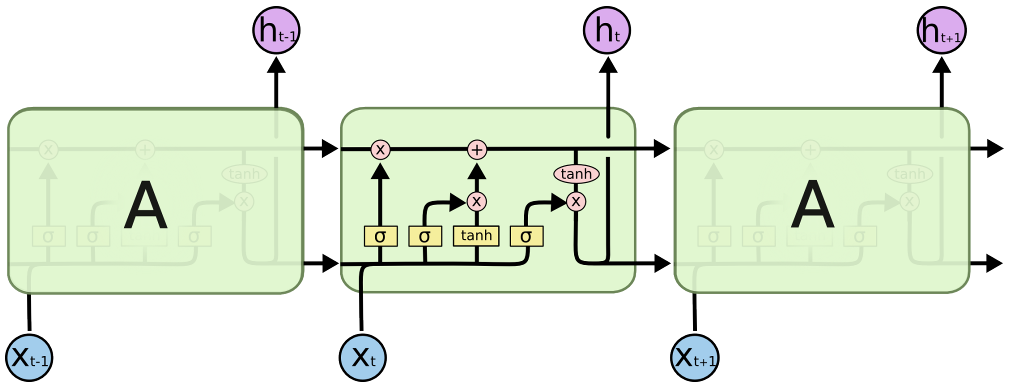
We will use LSTM Network to predict the stock prices. LSTM are special kind of Reccurant Neural Networks what work well on remembering long term dependencies. An RNN contains a single repeating neural network like below.



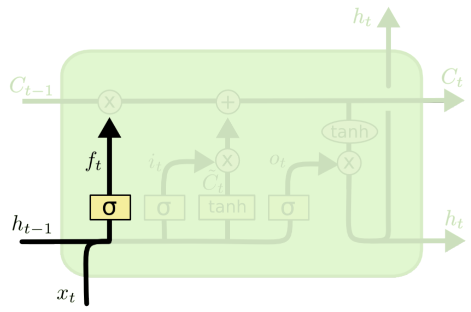
LSTMs also have this structure, however with 4 neural networks instead of single one, which act in a combined way to learn long term dependencies

The top line in the diagram is a cell state. The neural networks act as gates to control the flow of information in the top line. The x symbol indicates a point wise multiplication. The value sigmoid layer outputs is used for the point wise multiplication. If the sigmoid value is small, then very little information from the input flows forward, if the sigmoid is close to 1, then most of the input flows forward, hence the gate analogy.

The first gate decides what information must be thrown away from the cell state.



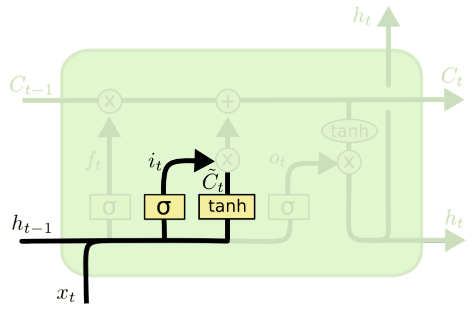
LSTM



$$f_t = \sigma\left(W_f \cdot [h_{t-1}, x_t] \; + \; b_f\right)$$

It looks at hidden outputs of previous predection and the current input and outputs a number between 0 and 1, which will be multipled with the cell state of previous prediction in order to *Forget* some of the previous cell state.

Next two layers are sigmod and tanh layers, which are called as Input Gate Layer, which decide which parts of hidden and input should be used as input to the next layer.



$$i_t = \sigma\left(W_i \cdot [h_{t-1}, x_t] \; + \; b_i\right)$$
$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] \; + \; b_C)$$

Finally, we will run a sigmoid on hidden state and scale it by output of sigmoid of input gate layer and pass it through tanh layer and combine it with the cell state that we got as output from applying the forget gate.

The reason that this works well in remembering the long term dependencies is that we learn what to forget from previous states and what to use from the new input every time.
This explaination is just to set the context, we will just use LSTM cell provided by tensorflow.

2.7 Benchmark

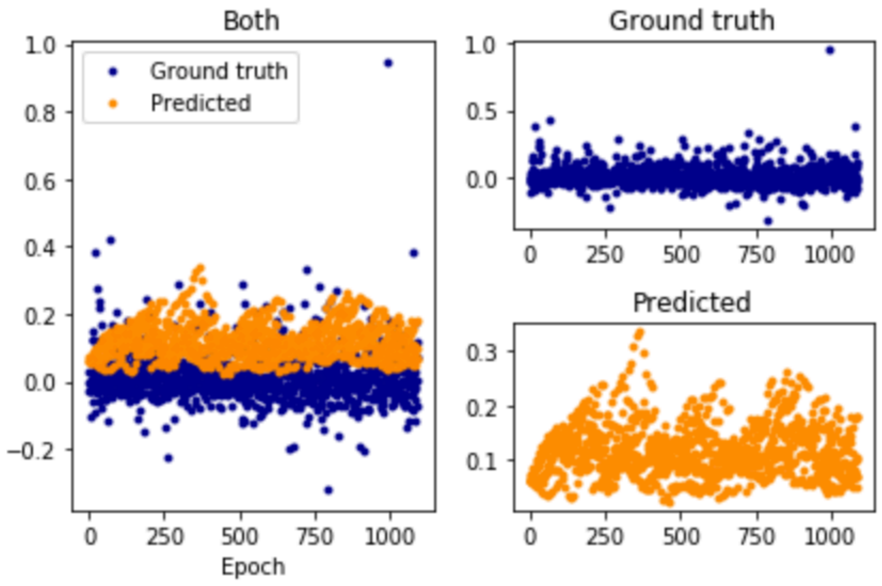
Since this is a returns prediction model, any benchmark would be good. The simplest benchmark model was given by the getting started kernel itself, where we simple create a normal distribution of range -1 to 1 and submit it as prediction. That model would score 0.00614 on the public leaderboard.

I also created a simple neural network with 3 fully connected layers working only on the numeric features (11 features) of the market data ignoring news data entirely. This model gave a public leaderboard score of 0.47.

2.8 Model Evaluation and Validation

2.8.1 Predictions for validation

I had divided data into train, validation and test datasets. After training the model on the training data, I evaluated the model on the validation set. The best score I got according to the formula mentioned in the metrics section for validation set for my model was 0.6. Below figure represents predicted versus the real values for validation set.



I created a predictor to predict and validate the predictions for the competition. Given that the competition was kernel only and had to be run in kaggle environment, it was not possible with current implementation of LSTM to get a high score. The best I could do was to get 0.65 in public leaderboard, which put me in top 40% of in the competition.

Considering that I only got into top 40% instead of top 10%, I am not satisfied about the approach or the model I have chosen. Given the constriants of this competition, I think it would have been better if I had gone for a gradient boosting algorithm rather than LSTM approach. However, though I did not score highly in the competition, I learnt a lot on processing stock data, reccurrent neural networks, LSTM etc.

2.9 Improvements

There are lot of areas where the code and data processing could have been improved. I have listed them below. 1. Natural language processing for news data - I have only used numeric values in news data. I am sure the winner of this competition would have to do some sort natural language processing for the news data to get more valuable insights into stock movements based on news. 2. Different models - If this were not a kernel only competition without GPU, I would have tried different model layers and sizes. Given that my kernel would take 8 hours to run once, it was very difficult to run more experiments. On the hindsight, I think I would have done better if I had chosen a competition where data would be available for offline download. 3. Using market indicators along with market data - I could have tried indicators like bollinger bands etc to add more insight into stock data. 4. Gradient boosting algorithms - Given the constriants of this competition, I would think the winner would use some sort of gradient boosting model rather than LSTM, since that would give you much more time to experiment with different kinds of models. My intention was to learn something new, hence I chose to do a LSTM instead of GBM models.

2.10 Conclusion

Given that this competition had rigid rules on not using GPU and not using any external data source as well as being kernel only competition, LSTM was probably not the best solution for this competition. With all the modification I could think of for this model, I could only get to 0.65 public leaderboard score. This just puts me in the top 40%. However, this also demonstrates that if this were a real project, LSTM would still be the best choice to predict stock prices from the news and market data.

2.11 References

Exploratory data analysis
LSTM concepts
LSTM for stock market
Combining news and market data
LSTM baseline for this competition
Benchmark