Capstone Report

January 18, 2019

1 Capstone Project

1.1 Machine Learning Engineer Nanodegree

1.1.1 Manish Yathnalli

2 1. Definition

2.1 1.1 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.

Predicting the returns of a stock is an unsolved problem. There is lot of research being done in the field, however, it is simply not possible to solve it to a certian accuracy. Given the nature of stock prediction, it is very easy to peek ahead and show the back testing results. This is called look ahead bias. This is a blog which claims to have used natural language processing to predict future value of stock but suffers from look ahead bais. I have also refered to work done by Marcos Lopez de Prado on financial machine learning and kept a pdf of why machine learning based financial analysis companies fail. These are some of the other papers I have referred 1. Predicting the Stock Market with News Articles 1. Stock Movement Prediction Using Machine Learning on News Articles 2. STOCK TREND PREDICTION USING NEWS SENTIMENT ANALYSIS

2.2 1.2 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 1.3 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 1.4 An overview of the proposed solution.

My intention here is to use market and news data to predict the confidance score of the returns the stock might possibly give for next ten days. Here is what I intend to do to solve the problem. 1. Clean the data of any errors and outliers. 1. Combine the market and news data based on time and asset code. 1. Use quantative market and news data as my input. 1. Create a LSTM network and train it with training data. 1. The LSTM network can learn long term dependencies of the target variable has on the input and can give excellent prediction for time series data. 1. Experiment with the size of the network and how many time steps to unroll. 1. Find the best model I can build and publish the result.

3 2. Analysis

3.1 2.1 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.

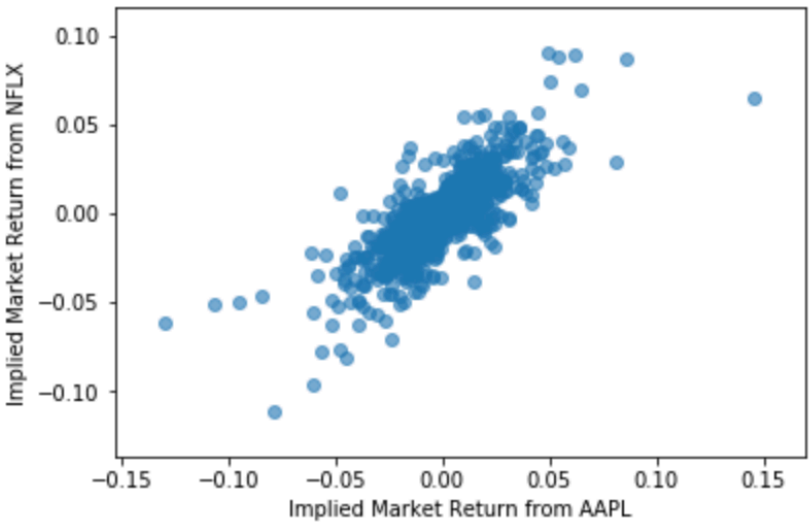
3.1.1 2.1.1 Understanding market data

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.

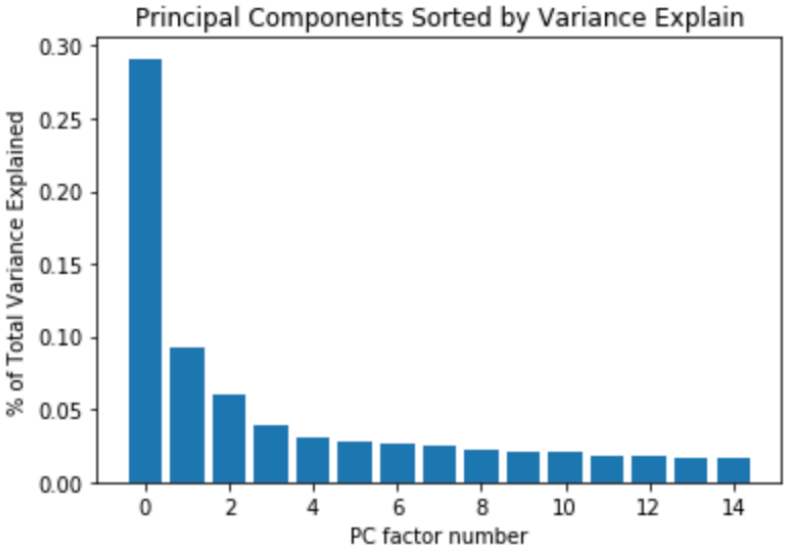
There are other returns which are along the same lines. I have used all the market data for my predictions.

3.1.2 2.1.2 Understanding the target variable.

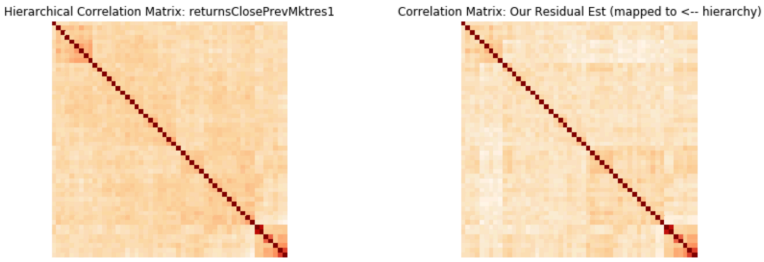
I had to first understand what is the target variable. It was not raw daily returns, which is represented by `returnsClosePrevRaw1` It was some kind of sharpe ratio per asset. It was not linearly related to raw returns. Here is a plot of raw returns vs target variable for Apple Inc.



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

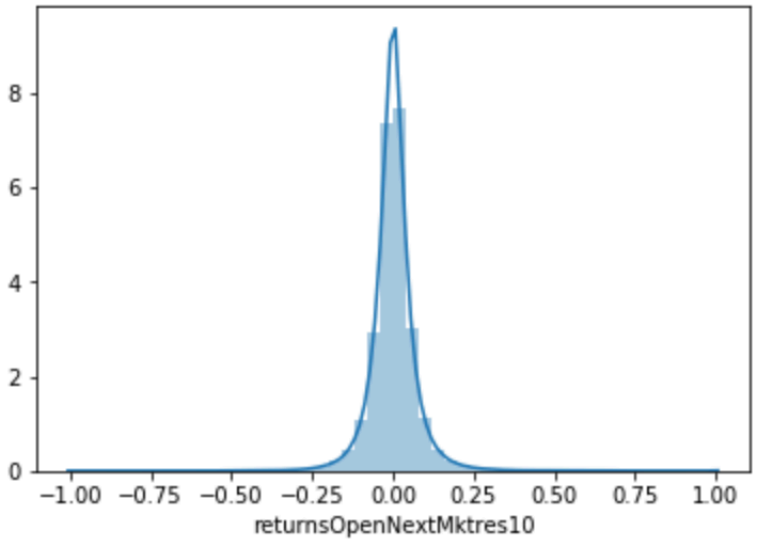


Below is the graph of co-relation of target variable after removing per stock beta adjusted market return from raw



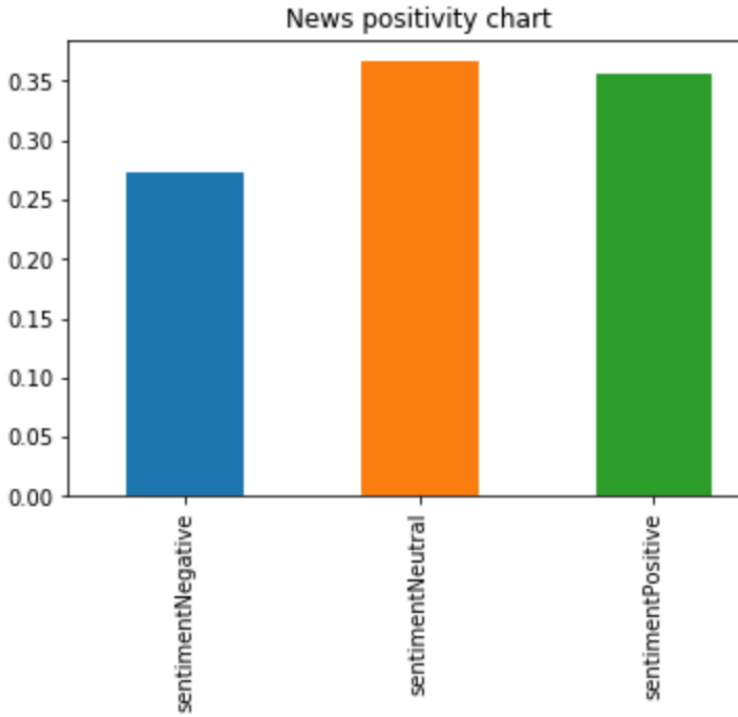
return From PCA I understood that target variable is substraction of per stock beta adjusted market return from the raw return.

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



3.1.3 2.1.3 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.

4 3. Analysis

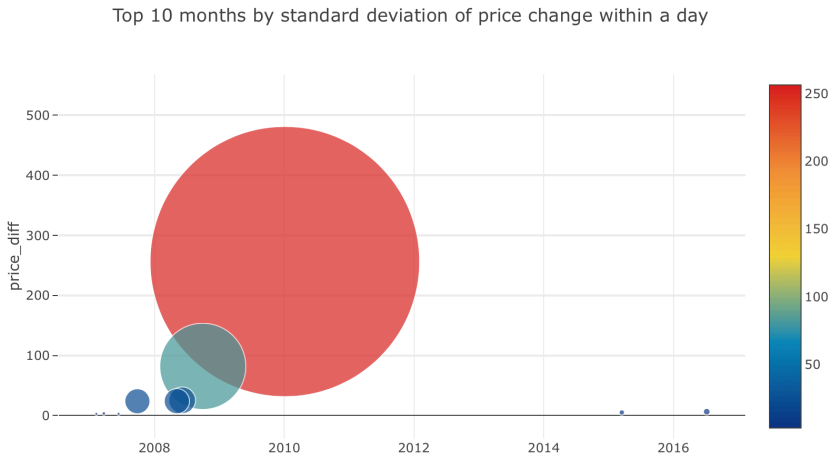
4.1 3.1 Data preprocessing

4.1.1 3.1.1 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.

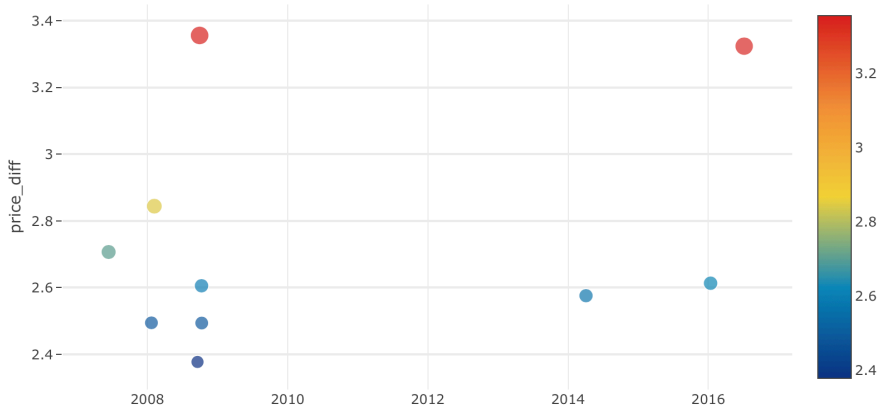
To further explain point 6, to remove unusually bad returns, I have simply dropped the rows where the price of the stock doubled and clipped the 0.1 percentile and 0.99 percentile values.

Here is a visualization of top standard deviation of price of stock within a day before cleaning



the data

As we can see there are huge deviations which could be data errors or because of stock splits and divident payouts etc. Below is the same plot after cleaning the data.



Another preprocessing step I took was to drop all the data before 2009, which had multiple financial crisis. I did not think that financial crisis before 2009 would affect the stock prices today. After preprocessing the market data, these were the columns left 1. Volume 2. Close 3. Open 4. returnsClosePrevRaw1 5. returnsOpenPrevRaw1 5. returnsClosePrevMktres1 5. returnsOpenPrevMktres1 5. returnsClosePrevRaw10 5. returnsOpenPrevRaw10 5. returnsClosePrevMktres10 5. returnsOpenPrevMktres10

I had these time columns. 1. year 1. week 1. day 1. dayofweek

Then there was the asset code. So overall there were 11 numeric, 4 time and 1 asset code. 16 columns in the market data.

4.1.2 3.1.2 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

After preprocessing this is what news data looked like. 1. urgency 1. takeSequence 1. wordCount 1. sentenceCount 1. companyCount 1. marketCommentary 1. relevance 1. sentimentNegative 1. sentimentNeutral 1. sentimentPositive 1. sentimentWordCount 1. noveltyCount12H 1. noveltyCount24H 1. noveltyCount3D 1. noveltyCount5D 1. noveltyCount7D 1. volumeCounts12H 1. volumeCounts24H 1. volumeCounts3D 1. volumeCounts5D 1. volumeCounts7D

There were 21 columns in the news data

4.1.3 3.1.3 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.

Once market data and news data were merged, I had 37 input features. ### 3.1.4 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

4.2 3.2 Challenges in data preprocessing and selecting features.

The preprocessing I have mentioned here is the final step in the experiments I did during this project. It was not this simple. Initially I started with just market data. It had 11 numeric features. I initially built a simple fully connected neural network model on market data. The performance was not good at all. It was just slightly better than random normal distribution result. Then I checked data for errors. There were several NANs and unreasonalbe changes in the stock prices. This is when I decided that I need to drop the 99 percentile and 1 percentile outliers. Once that was done the results improved ever so slightly. This is where I started to submit and I was in bottom third of the leaderboard.

The other problem with data preprocessing was when using StandardScalar. This requires the sum of all the data in a column must be representable without overflow in a double. Some of the columns were overflowing. I had to drop these columns to make StandardScalar fit the model.

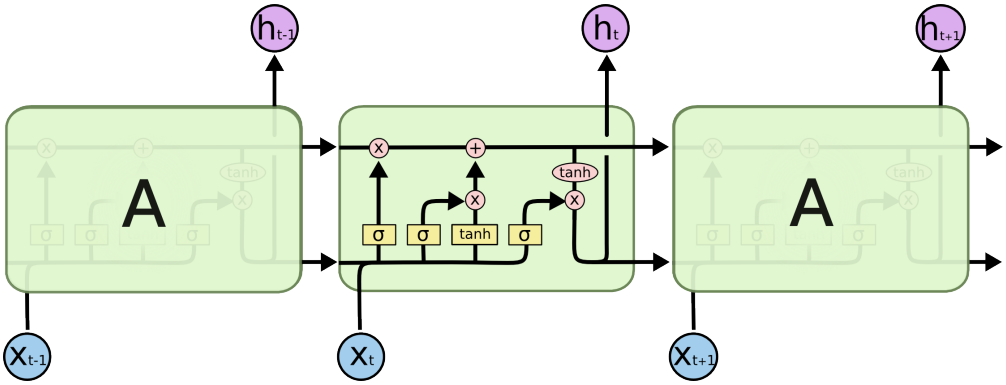
The other challenge was to figure out how to merge news and market data. For that I refered to this excellent guideline.

Once I had good preprocessing steps, the next task was to build a LSTM network.

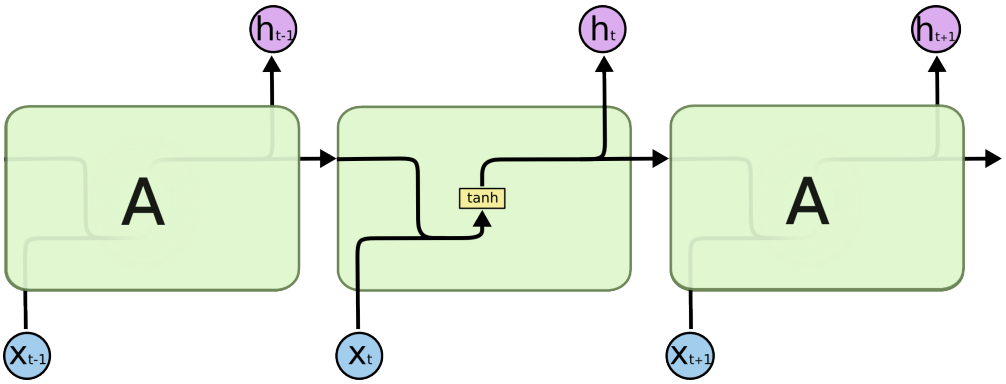
LSTMAlgorithm

4.3 3.3 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.



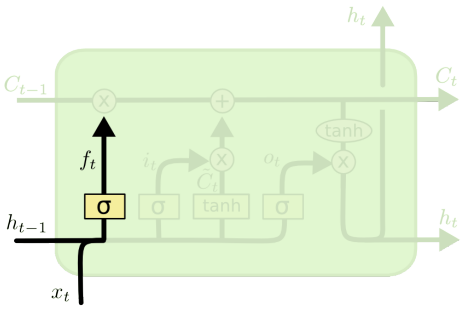
LSTM



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.

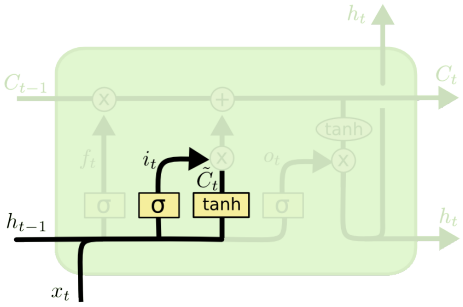
The first gate decides what information must be thrown away from the cell state.



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

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.

4.4 3.4 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.

5 4. Methodology

5.1 4.1 Model Evaluation and Validation

5.1.1 4.1.1 Parameter tuning and playing with number of layers.

I created my model as a class so that I can play with number of perceptrons in each layer. Initially I started with 1 LSTM layer of size 32 and one fully connected layer. I used SGD optimizer and a learning rate of 0.001. I trained it with 20 epochs. Since I had created LR decay and early stopper with patience of 3. This trained for 12 epochs and gave public leader board score of 0.43. I considered this as my base. It took about 6 hours to train for 12 epochs. This is because I had to run this as a kernel in Kaggle and could not activate GPU because of competition rules. Then I played around with the number of layers and activations per layer. The max I went was 4 layers with an architecture of 256, 128, 64 and 32 activations. This caused my kernel to crash because of memory overflow. Then I setteled to 128, 64, 32 architecture and it took about 6 hours to train for 6 epochs. I played around with other optimizers and learning rates. I found that adam with learning rate of 0.01 and ReduceLROnPlateau with a reduction of 10 times every time there is a plateau worked best.

Given the constraints of kaggle competition I could not do a lot of experimentation. If this was a GPU enabled competition or if data was available for download, I am sure I could have created a better model.

I do not presume to be an expert on how many activations to choose or what learning rate to choose or which optimizer to use. I played around with the architectures and experimented till I had time and ideas to tune. Then I picked the one that gave best public leaderboard score.

5.1.2 4.1.2 Final Model

This is what the final model looked like
Layer (type) Output Shape Param #
lstm_1 (LSTM) (None, None, 128) 84992
_____ lstm_2 (LSTM) (None, None, 64) 49408
_____ lstm_3 (LSTM) (None, 32) 12416
_____ dense_1 (Dense) (None, 1) 33
Total params: 146,849 Trainable params: 146,849 Non-trainable params: 0
_____

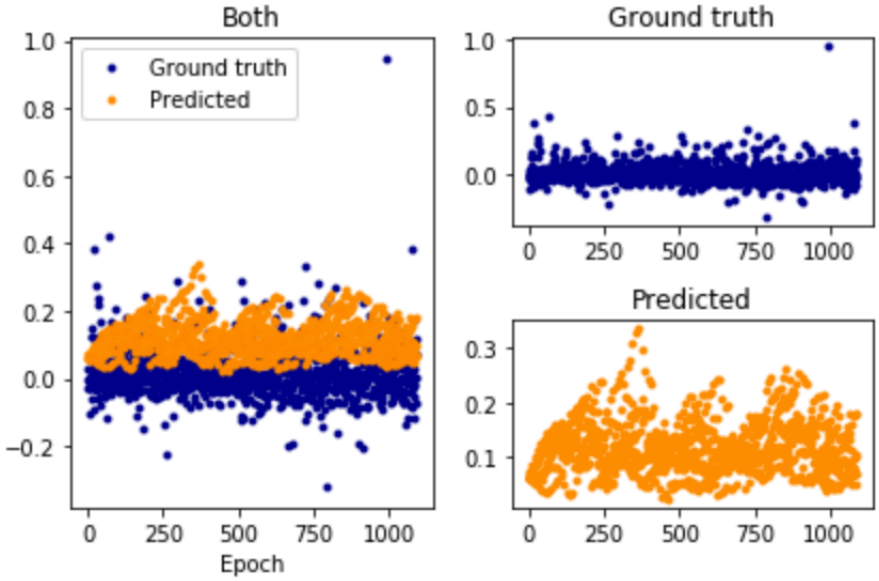
It had three LSTM layers of size 128, 64 and 32 and once fully connected sigmoid layer.

5.1.3 4.1.3 Robustness of the model's solution.

Even though this model did not give me the top 10% score, I am confident that this is a good model and generalizes well. The biggest problem of machine learning models for financial prediction is the peek ahead problem as I have emphasized earlier with citations. The biggest plus point for this model is although it is quite simple model, it does not have peek ahead bias. The reason is the the way this competition is built. You cannot see the data for t+10 days untill you have predicted the value for t. For this reason, I am confident that since the model scored well in public leaderboard, when the competition unfolds and private leader board is built over next 6 months as the market data comes in, my place in private leaderboard will still be in top 1/3. I could have used some dropouts to generalize well, but as opposed to image recognition, where the working of dropouts are very well studied and generalized, I did not find many studies of how dropouts would work on financial data.

5.1.4 4.1.4 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.

6 5. Conclusion

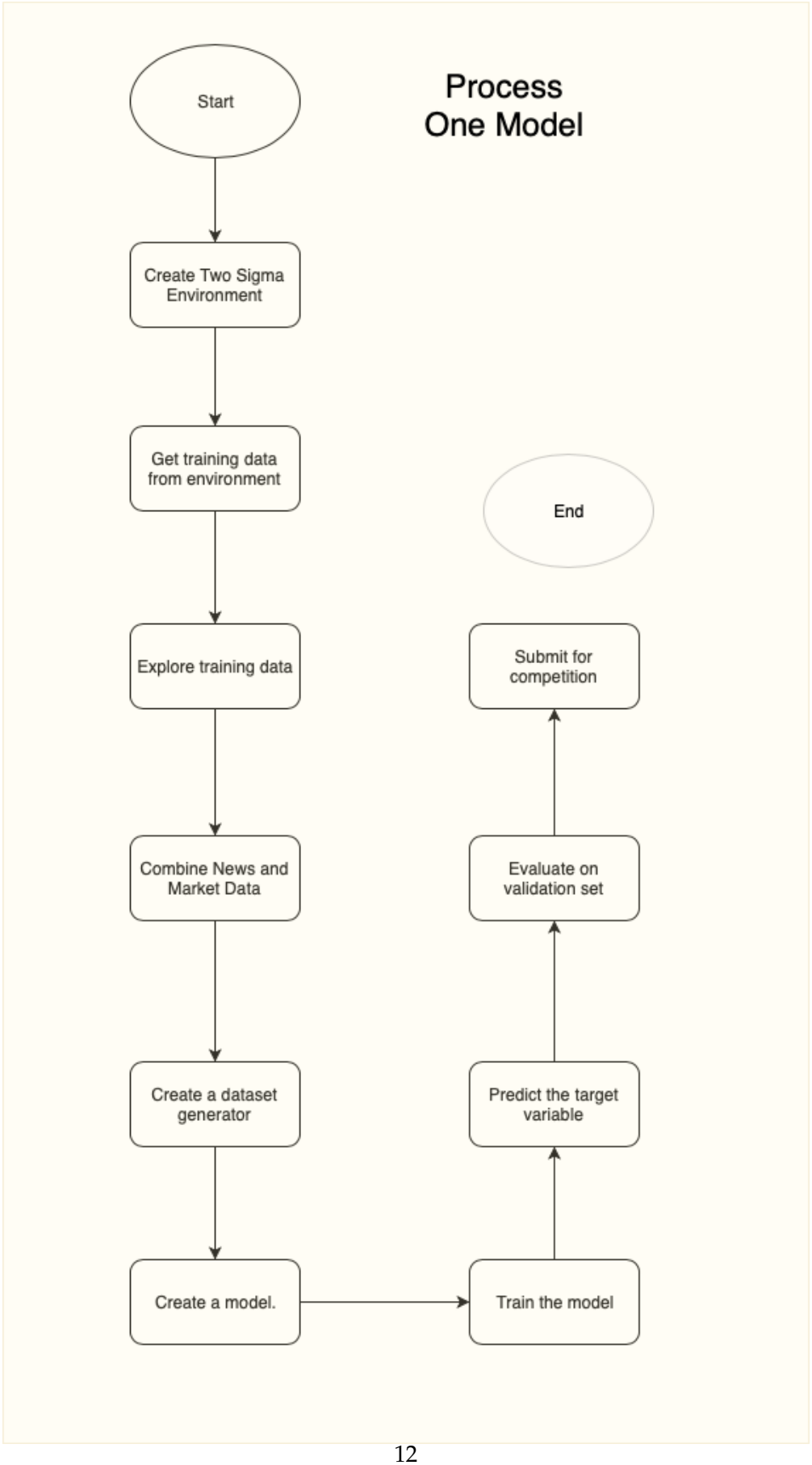
6.1 5.1 Free Form Visualization

I have no idea on what is required here. After procrastinating on this for few days, I am writing what ever I thought is required here. If this was a image processing project, I would have show how the model recognized the parts of image, but this is a stock prediction project. I have shown the predicted versus ground truth on validation data above. Test data is not yet available since this is future prediction problem. Hence I am showing a flowchart of what I did for this project. Here is the flow chart

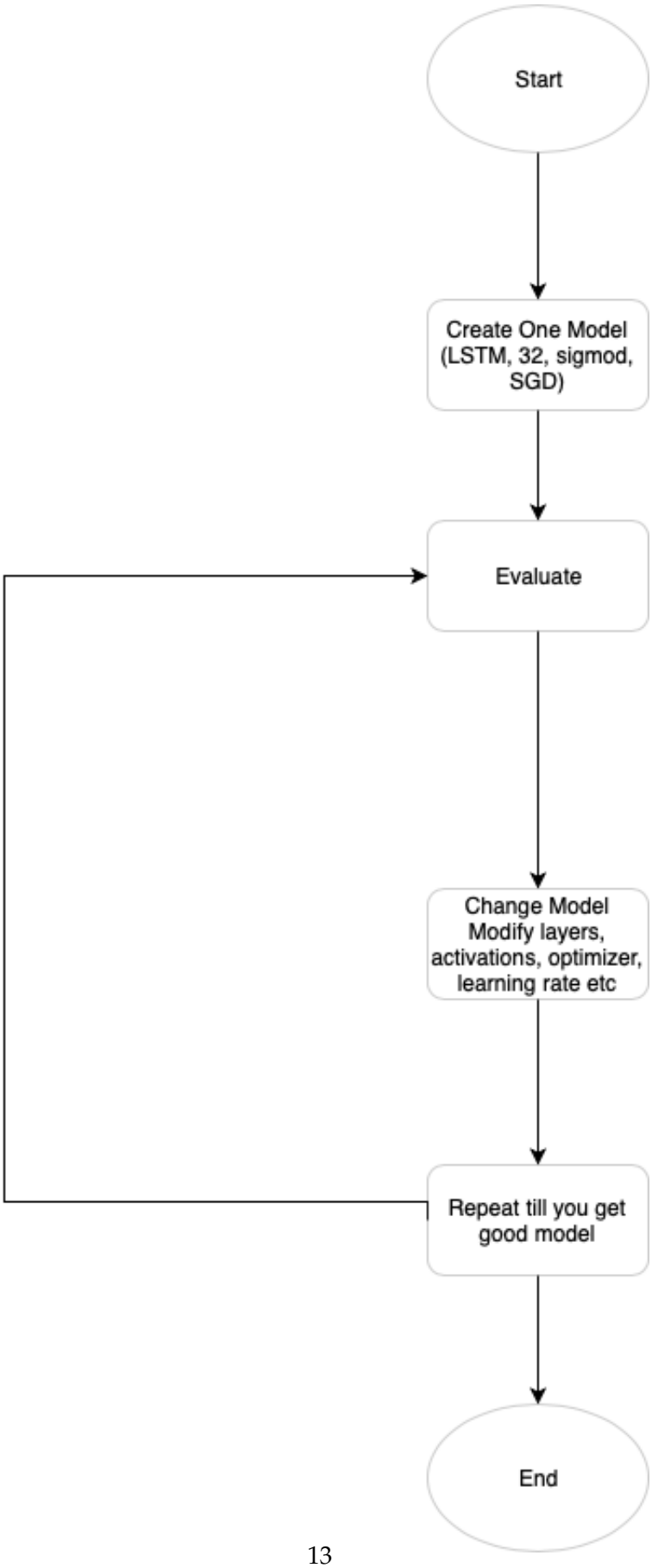
Considering the above process is for evaluating one model, given that an environment can be created only once per run of the kernel in this competition and once we call get next days, we cannot call it again, for every parameter, we need to run this whole process from scratch. Below is the process for doing that.

6.2 5.2 Recap

Here is a recap of what I have done in this project. 1. Understanding the competition - This was a unique competition with data not being available to download and having to calculate next ten days market adjusted return per stock. It took a while to even understand what this is about


Process
One Model
Start
Create Two Sigma Environment
Get training data from environment
Explore training data
Combine News and Market Data
Create a dataset generator
Create a model.
Train the model
Predict the target variable
Evaluate on validation set
Submit for competition
End

FlowChart


Start
Create One Model (LSTM, 32, sigmod, SGD)
Evaluate
Change Model Modify layers, activations, optimizer, learning rate etc
Repeat till you get good model
End

Flowchart 2

and how to go about the competition. 2. Processing the data - The data had so many errors and outliers. There were two different datasets, one for market data and one for news. I had to clean the data, combine the market and news data and create a data generator for LSTM algorithm 3. LSTM Algorithm - Neural networks can be considered as universal functions. By combining non linearity with linear functions and layers, they can model any problem. However, the limitation of neural networks are that they cannot remember state. They are like functions that map x to y. They cannot model historic data. The RNNs can model historic data by creating a loop, but they cannot remember long term dependencies. Hence LSTM were invented, which are combination of neural network and can remember long term dependencies. 4. Creating LSTM and training - This step involved with experimenting with optimizers, learning rates, activation functions, number of layers and activations per layer. Finding the best model that would accurately solve our problem. 5. Predicting the future with the trained model - This step involved creating the predictions and submitting for competition. It was very difficult given the nature of the competition. Any time I had to change anything, I had to re run the kernel completely including training set, since saving models was not allowed in the competition. Each iteration would take 8-12 hours. 6. Reporting the results.

6.3 5.3 Summary

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.

6.4 5.4 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.

7 6. References

Exploratory data analysis
LSTM concepts

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