FINANCIAL MODELING - THE METEORIC RISE OF TESLA (TSLA)

Dhruv Srikanth

APPLIED DATA ANALYSIS PROJECT FINAL REPORT:

ABSTRACT

Tesla's stock has experienced a meteoric rise between the years 2010 and 2020. As most companies recovered slowly from the 2008 financial crisis, Tesla, along with other technology giants experienced tremendous growth and did not seem as adversely affected by the crisis as other companies did. Furthermore, Tesla, despite the dramatic effect the financial crisis had on the automobile industry, was able to rise above the rest to experience tremendous growth. With the objective of modeling Tesla's success between 2010 and 2020, I developed a dataset and model to perform forecasting on Tesla stock price (TSLA). This was accomplished by feature engineering predictors that are well-known financial indicators for stocks such as RSI and MACD, and leveraging Tesla CEO, Elon Musks Twitter activity to augment the dataset with sentiment information. Finally, a LSTM sequential model was developed to perform forecasting on the data.

INTRODUCTION

Tesla has experienced meteoric success between the years 2010 and 2020. Despite automobile and manufacturing companies experiencing large drops in stock price after the 2008 financial crisis, I wondered why Tesla, a car company, essentially an automobile and manufacturing company, experienced tremendous growth. A possible reason behind this is that Tesla (TSLA), is actually a technology stock or "tech" stock. Another possible reason is that factors external to traditional indicators for the company had a greater effect to play in the company's value. Companies of this size are often made up of thousands of employees, whose performance would logically dictate the trajectory of the company, thereby, influencing the stock price. It seems impossible that a single individual could hold such an influence over the company's stock. However, the value of the cryptocurrency DogeCoin appears to be highly influenced by Elon Musk's activity on Twitter. Therefore, in order to effectively and accurately model TSLA, feature engineering must be performed

thereby adding forms of what may previously have been non-traditional information, to the dataset, thereby enabling the model to learn from a more diverse feature set. Towards this, augmenting the stock data with the sentiment score of Elon Musk's tweets can provide an avenue to better model Tesla's stock. In addition, traditional financial indicators such as relative strength index (RSI) can be used to capture short, intermediate and long term trends that can augment the dataset with more feature rich information, thereby leading to a more accurate forecasting model. This has been the goal of the project and has been discussed in greater detail over the next few sections. The work described can be useful in determining optimal trading strategies, understanding what factors, previously not included in these strategies could now be incorporated (for example, a single person's tweets), and even in building automated trading bots.

RELATED WORK

I began researching the different kinds of machine learning models employed for the purpose of stock prediction and time series analysis. This led me to the following two papers -

1. Stock Prediction using Sentiment analysis and Long Short Term Memory
2. Harvesting social media sentiment analysis to enhance stock market prediction using deep learning

The first paper obtains information on stock data for Apple, Google and Microsoft through Yahoo finance and Twitter data by scraping tweets off the Twitter website, using Tweepy (Twitter API). The stock data is normalized and then split in a train-test 70-30 manner. The Twitter data is cleaned by removing all links and special characters. Finally, they are divided based on the polarity of the tweet (positive/negative). An LSTM architecture is employed for the prediction model due to its use of feedback connections. Root Mean Squared Error (RMSE) is used for evaluating the performance of the model. The authors of the paper achieved an RMSE of 1.2583.

The second paper approaches the problem of stock market prediction augmented with sentiment analysis on a more broad spectrum. News data was collected from Moneycontrol, IIFL, Economic Times, and Twitter. Stock data information was gathered from the National Stock Exchange of India (NSE). A similar approach to the previous paper was employed for cleaning the news data. For performing sentiment analysis, Naive Bayes, Maximum Entropy, Linear Support Vector Classifier, and Decision Tree classifier were used. The news data was then assigned polarities of -1 for negative sentiment, 0 for neutral sentiment and 1 for positive sentiment. For the stock data, a trading condition

was determined based on the close price of the stock for the trading day which was added as a predictor to the dataset along with the sentiment information. An LSTM network was trained and employed to perform predictions on the data. The sentiment classifiers achieved the following results for accuracy - Naive Bayes - 86.72%, Maximum Entropy - 88.93%, Linear SVC - 89.46, LSTM - 92.45%. For the sentiment classifier, the performance matrix detailing precision, recall and F1-score were utilized in determining the performance. Unfortunately, I was unable to determine the evaluation metric used for the LSTM model from the information provided in the paper.

Based on this, I learned that recurrent neural networks (RNN) are often used due to their ability to accurately model sequential data by understanding the temporal context (retaining memory). Upon diving deeper into the kinds of models and architectures used, I came across a type of RNN, the long-short term memory unit (LSTM), that helps to avoid the vanishing gradient problem and has also shown promise in the performance it is able to achieve when applied to time series problems like a stock market prediction. Upon exploring LSTM networks further, I came across two more kinds of networks -

1. GRU - Gated Recurrent Units
2. ESN - Echo State Network

GRUs are a form of LSTMs wherein in each LSTM unit, we remove the forget gate and keep the update and reset gates. This allows the network to train faster, have fewer parameters and utilize less resources than LSTMs. The theory suggests that LSTMs are capable of remembering a sequence for longer than GRUs.

ESNs utilize a reservoir computing framework in which there is a non-trainable dynamic reservoir to randomly sample from, which then connects to a series of trainable layers. The idea of ESNs is that for an input, a higher dimensional embedding is formed in the reservoir. This represents a state of the data and is then passed to the trainable layers of the ESN. By passing each state to the trainable layers, the model is able to generalize the results for different inputs over time, as each state from the reservoir “echo” through the trainable layers. The advantage of this sort of network is that the reservoir weights are randomly generated and fixed. Therefore, the model is able to learn fast and utilize less resources. In addition to this, like LSTMs and GRUs, ESNs are capable of avoiding the vanishing gradient problem.

I believe similar approaches to the network architecture of my model will be key in accurately predicting Tesla's stock price.

DATA

I used a Kaggle dataset that collates the stock price of Tesla (TSLA) between 2010 and 2020 and can be found here. Additionally, I amalgamated Elon Musk's tweets into the dataset. The required twitter data can be found here.

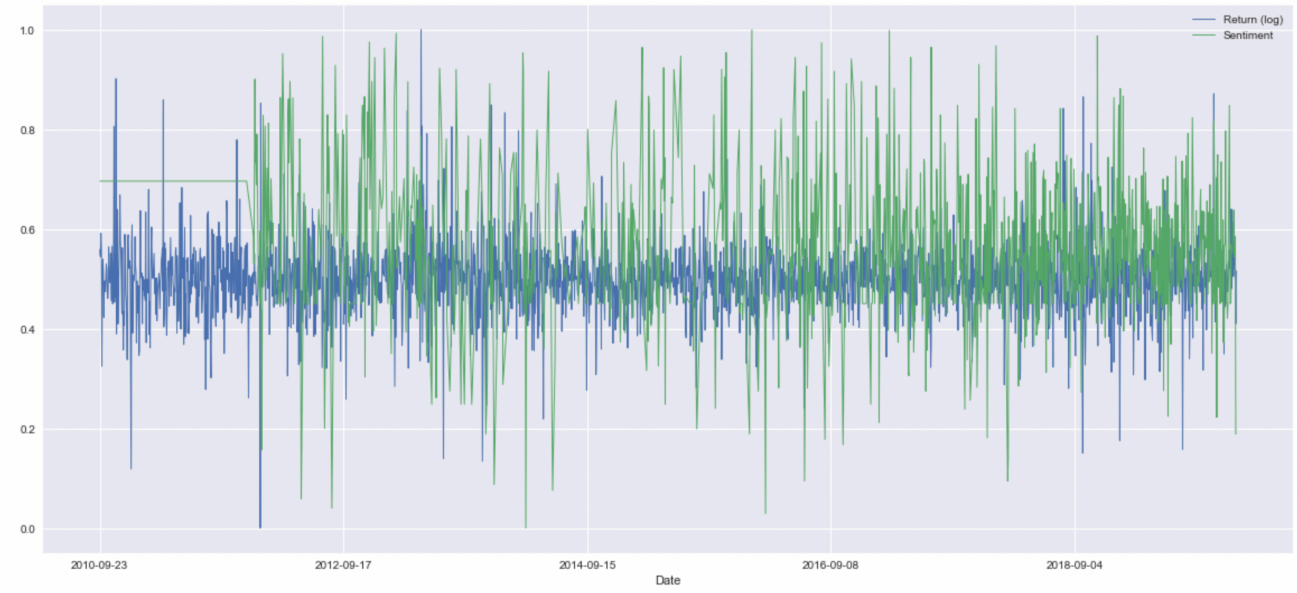
IMPLEMENTATION

Towards producing an effective forecasting model, I divided my implantation into two stages. The first stage focused on creating a feature rich dataset through feature engineering. The second stage involved creating, training and finally, evaluating the model. My time was evenly divided between the two stages. Whilst researching different financial indicators that could be feature engineered into the dataset, I spent most of my time on what each feature represents and how can it collectively contribute to the models learning. The implementation of each feature took far less time than deciding which features were best to implement. When developing the model, tuning the hyper parameters and testing different architectures took around the same time as researching which architectures to use and the evaluating tradeoffs between them. The libraries I have used (listed below) were already familiar to me, therefore, I spent the majority of time on this project, researching relevant features and models that are appropriate to my data. The approach taken for both of these stages can be seen below -

1. Feature Engineering - Features already present in the dataset were open price, day's high price, day's low price, closing price, volume traded that day and the adjusted closing price. In order to model the effect of the sentiment of the CEO, Elon Musk's tweets, I incorporated a sentiment score for each day. On days that had multiple sentiment scores (one for each tweet made), I found the mean of sentiment scores for the day. To account for days that provided no sentiment score, I performed linear forward and backward interpolation. This was under the assumption that at any given time, a person's sentiment is almost always representative of his immediately prior sentiments (from the previous day). Given that the window of time we are looking at is 10 years, interpolating under this assumption appears to average out any errors that may occur as exceptions to the assumption made prior. Apart from this, I computed the log of the return for each day. In addition, the relative strength index (RSI) was used as a short term indicator, moving average convergence divergence (MACD) was used as a long term indicator and the intermediate

momentum was computed to be an intermediate price indicator. The formula for RSI is provided here - RSI = 100 − 100/[1+(Average loss / Average gain)]. I computed the averages over 14 days to keep the RSI as a short term indicator. The formula for MACD is provided here - MACD = EMA (short period) - EMA (long period), where EMA is the exponential moving average. For these two periods, I have chosen 20 and 200 days respectively, in order to capture a trend over the long term. Finally, the intermediate momentum was computed by calculating the moving average over a 60 day period. By utilizing three different time windows for the trend, I focused on providing a diverse feature set to my model to learn. The short, intermediate and long time windows would respectively provide a different perspective on the trend of the stock price. Additionally, the return would provide a standardized perspective on the value of the stock. Furthermore, the sentiment aims to add a non-traditional indicator of a stocks performance, attributed to the sentiment of a prominent figure in the company. Here is how the indicators look visualized (indicators on the y-axis and date on the x-axis) -

As we can see in the above plot, the long and intermediate term indicators are follow a similar trend compared to the closing price of the stock. The y-intercept of the indicators are far higher than that of the close price, as expected, since the indicators consider a longer time window in which the TSLA stock does exceptionally well.



In the plot above, it is difficult to notice a trend between the return of TSLA stock and the sentiment, in the above plot. I am postulating that its helpfulness will be evident in the latent space.

From the above plot, we can see that the short term indicator (RSI) does provide some insights into the trend of the close price of TSLA stock, however, due to its short term nature, this is restricted to far smaller windows. In addition, we are able to see a correspondence between the return and the RSI. I am hoping that this will provide short term knowledge of the stock to my model.

Some of the challenges I encountered when trying to feature engineer a feature rich dataset was the lack of clarity on what time frames may be best for different indicators. In addition, the large variety of indicators made it difficult to decide between them, given my lack of experience in the financial industry. In order to overcome these challenges, I dedicated more time towards researching different financial indicators and decided to utilize the most widely used set of indicators. Furthermore, I chose time windows for different indicators by assigning a single type of trend to be captured in each indicator.

2. Model Implementation - When deciding the architecture of my model, I needed to weigh the tradeoffs between ESNs, GRUs, and LSTMs. Though ESNs and GRUs provide a more computationally efficient architecture for forecasting time series, LSTMs seem to be better at maintaining memory over longer sequences of time. Therefore, I decided to use an LSTM architecture, given that I was modeling 10 years of data. When developing the architecture I started with a single hidden layer. I increased the number of hidden units from 50 (what I initially started with) to 100 and then 150. Additionally, I gradually increased the number of hidden layers in order to accurately model the complexity of the dataset. After seeing a saturation in the accuracy with respect to the validation set, I decided to proceed with the following architecture. In the model, there is an input LSTM layer with 100 units, a hidden LSTM layer with 100 units, a hidden LSTM layer with 50 units and an output layer which was a dense layer (fully connected layer). Each LSTM layer is followed by a dropout layer with a value of 0.2, which helps to perform regularization and thereby reduce overfitting the model on the training data. Here, 0.2 indicates that 20% of the values will be randomly sampled and dropped, thereby preventing the model from "over" learning the features of the training data. The architecture in this model can be seen below -

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 lstm (LSTM)                 (None, 100, 50)           12200

 dropout (Dropout)           (None, 100, 50)           0

 lstm_1 (LSTM)               (None, 100, 50)           20200

 dropout_1 (Dropout)         (None, 100, 50)           0

 lstm_2 (LSTM)               (None, 50)                20200

 dropout_2 (Dropout)         (None, 50)                0

 dense (Dense)               (None, 1)                 51

=================================================================
Total params: 52,651
Trainable params: 52,651
Non-trainable params: 0
_________________________________________________________________
```

The training configuration I followed is given below -

1. Train-Test split = 70%-30%.
2. Validation split on the training data is 35%.
3. Sliding window over data = 100 days
4. Epochs trained for = 20
5. Batch size = 32

The training strategy is provided below -

1. Optimizer - Adam
2. Loss and metric - Root mean squared error (RMSE)

SOFTWARE

I used the following tools in the approach described above.

1. Language - Python was used.
2. Sentiment Analysis - Python package called vaderSentiment that has been trained specifically on twitter data.

3. Data Cleaning - NumPy, Pandas and tweet-preprocessor package for cleaning tweets and creating the dataset.
4. Utility Library - The package sklearn was used for normalizing the data and defining the loss function (RMSE).
5. Data Visualization - The packages matplotlib and seaborn were used to visualize the data and results.
6. Model Creation - TensorFlow was used to build the model.
7. Running Environment - I used jupyter notebooks for the purpose of running my experiments, analysis and investigation.
8. Model Training - I trained the model on Google Colab to take advantage of their GPUs.

EVALUATION

I evaluated my models performance on the root mean squared error on the test set. The quantitative results can be seen here - RMSE_test = 0.0651. This outperforms the results achieved by the papers I have referenced. The best RMSE I have come across in other papers is 1.2583, however, this was a generalized model on multiple stock datasets. Therefore, I believe the lower RMSE I have achieved is owing to the fact that this is only for the Tesla stock data.

In the plot below, the ground truth close price of the stock over 10 years is provided, along with the predictions of the model on the test set -

Given below is a magnified version of the same -

TAKEAWAYS

The model performs quite well on its predictions. It appears that Elon Musk's tweets sometimes dictate the swing of the stock as seen in the feature engineering graphs. Furthermore, the addition of these features adds to the flexibility and robustness of the model to accurately predict a stocks value. The high performance of the model is seen quantitatively, through the low RMSE and qualitatively, through the figures provided above. This illustrates the following in improving a models performance or the kinds of models to choose for the purpose of time series forecasting -

1. The effectiveness of LSTMs in time series forecasting.
2. Noting tradeoffs between similar architectures such as GRUs and LSTMs and their potential use based on the application of the model.
3. The importance of feature engineering as seen through the use of financial indicators such as RSI and MACD to provide additional knowledge to the model to learn from.
4. The use of non-traditional information such as sentiment to augment the dataset.

FUTURE WORK

Future work can include investigating optimal time windows for RSI, MACD and momentum in tandem with a perspective on what the model is supposed to accomplish. For example, if intraday trading is the objective, feature engineering would be focused towards developing short trend representative features for automated trading bots. Furthermore, based on the kind of trading, tradeoffs between different model architecture can be explored towards developing an efficient and efficacious model for the problem at hand. For example, ESNs and GRUs may be more helpful in the case of intraday trading due to its speed. Additionally, these architectures may be lighter to implement on edge devices given the resource constraint of the the edge device.

REFERENCES (BIBLIOGRAPHY)

Given below is a list of references that I have used in previous sections -

1. Tesla Stock Dataset - https://www.kaggle.com/timoboz/tesla-stock-data-from-2010-to-2020

2. Elon Musk Tweets Dataset - https://www.kaggle.com/ayhmrba/elon-musk-tweets-2010-2021

3. Stock Prediction using Sentiment analysis and Long Short Term Memory - https://ejmcm.com/pdf_3126_16b444b632c88db6fed0c6558dd6930a.html

4. Harvesting social media sentiment analysis to enhance stock market prediction using deep learning - https://www.ncbi.nlm.nih.gov/pmc/articles/PMC8053016/

5. Echo State Networks - https://towardsdatascience.com/gentle-introduction-to-echo-state-networks-af99e5373c68#:~:text=Echo%20state%20network%20is%20a,randomly%20assigned%20and%20not%20trainable

6. Gated Recurrent Units - https://towardsdatascience.com/understanding-gru-networks-2ef37df6c9be