STOCK FORECASTING USING CONTINUOUS HIDDEN MARKOV MODEL VARIATIONS AND LSTMS

Rowan Lavelle, Ethan Behar, Brendan McShane, Jack McShane

Indiana University, Graduate Luddy School Of Informatics, Computing and Engineering

ABSTRACT

Forecasting and prediction of stock prices is a classic but recent problem within the quantitative financial community. Stock forecasting has been analyzed extensively using a variety of methods ranging from classic explainable models like regression, to new research using black box models like recurrent neural networks. This paper introduces the comparison of Gaussian Mixture Model Hidden Markov Models (GMMHMMs), Gaussian Hidden Markov Models (GHMMs), Long-Short Term Memory networks (LSTMs), Gated Recurrent Unit networks (GRUs), and Recurrent Neural networks (RNNs) as prediction systems for stock price forecasting. We also introduce a proof of concept for sentiment analysis to be used for next day price prediction, as well as next day trend prediction.

***Index Terms*—** Stock Forecasting, Hidden Markov Model, LSTM, GRU, RNN

1. INTRODUCTION

The stock market is a collection of markets where exchanges are made for buying and selling shares of publicly traded securities. The overall market can be viewed as a stochastic random walk with an upward trend, however due to the weak assumption of the Efficient Market Hypothesis (EMH) this problem is currently unsolvable, meaning it is impossible to exactly predict the movements of the market. With this problem posed, any knowledge that a financial institution can gain, with respect to what future prices will be for individual securities, gives an avenue for making large amounts of revenue.

This is a challenging problem to solve for a multitude of reasons, it may seem as though the price of a stock at time $(t + 1)$ is highly dependent on time (t). However many factors influence the price such as news, sentiment, and volatility. This causes many issues when attempting to forecast a prediction. For example, a stock can grow steadily and perform well, then a negative earnings report can come out and the price can tank the next day.

A common source for stock market data is the Yahoo Finance API which allows you to search for companies listed on the New York Stock Exchange (NYSE). The data used when attempting to forecast stock prices is primarily: *close price*: the price at which a stock closes when trading ends for the day, *open price*: the price at which a stock opens when trading starts for the day, *high price*: the highest price of a stock during the trading day, *low price*: the lowest price of a stock during the trading day, *volume*: the amount of shares bought and sold during the trading day. For this paper, we used the Fastquant [1] wrapper to pull stock data.

Previous techniques like regression [2] and decision trees [3] have been used as prediction systems due to their explainable nature. In recent studies, it has been shown that the use of more complex models such as LSTMs [4, 5, 6, 7] and GMMHMM's [8, 9] can be beneficial for accuracy of prediction. However, they have the drawback of not being explainable. Using models of this caliber are self explanatory, both are common models in academia for time series analysis.

2. RELATED WORKS

Due to the nature of capital gain, for financial institutions through knowing the future value of securities, stock market forecasting has been a popular study in both academia and industry. Within academic research there has been a plethora of models presented to solve this problem; LSTMs are one of the most popular ones. Given the variety of possible solutions, many different network architectures, preprocessing, and forecasting types have been presented.

The approach from [4] uses a shallow LSTM network that is two layers deep, yet the network uses 512 hidden recurrent nodes in each layer, which is a large amount when compared to [5]. This paper also normalizes their data per training window, each price point in the window is divided by the first price point in the window. Their predictions are also normalized, and not true closing prices. This presents a problem because the model is only tracking the movement of the security, there is no real relation to the actual price [4]. In most papers that deal with LSTMs for stock forecasting, the trend is that the inputs and outputs are all 1-dimensional. Networks are trained on open/close prices and then are used to predict open/close prices [4, 6]. However in [5], the paper shows strong results and proposes using a multidimensional observation of

$$O_t = (High, Low, Open, Close). \tag{1}$$

This is similar to [7, 8]. To compensate for the smaller number of hidden recurrent nodes, [5] uses an additional dense layer of 16 nodes, before feeding into a single node dense layer to predict the regressed value. This approach is similar to what is shown in [7]. In both [5, 7] there is no discussion of how changing the number of hidden parameters affected results.

The use of dropout layers is presented in [6]. It uses a 4 layer LSTM network with dropout layers between the LSTM layer. This is a useful addition that can help deal with overfitting, since LSTMs are prone to overfit when working with noisy signals.

Lastly, [4, 5, 6] lack key information for replication. This makes it particularly difficult to recreate their work.

The use of Continuous Hidden Markov Models (CHMMs) are also present in research with respect to stock forecasting [8, 9]. A strong model is presented in [8] which uses a CHMM for next day price forecasting through fractional change. The paper, similar to [5], uses a multidimensional observation for its predictions.

$$O_t = \left(\frac{close - open}{open}, \frac{high - open}{open}, \frac{open - close}{open} \right) \quad (2)$$

$$O_t := (fracChange, fracHigh, fracLow)\,. \quad (3)$$

Similar to how we will present our GMMHMM, [8] uses Maximum A Posteriori (MAP) estimation to calculate the most likely next day fractional change. The paper uses a set of 5,000 possible observations and chooses the one with the maximum log likelihood.

3. METHODS

3.1. Models for Fractional Change Prediction

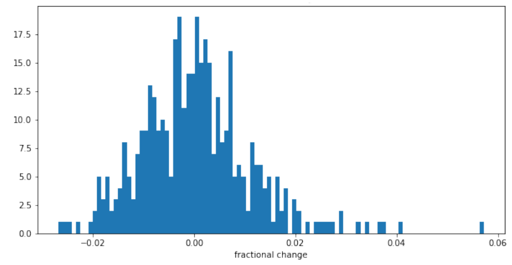
Our first proposed model is a CHMM which uses a Gaussian Mixture Model (GMM) as the emission probabilities. It is the same model that [8] presented but with different hyper-parameters. The model can be denoted as

$$\lambda = (\pi, A, E), \quad (4)$$

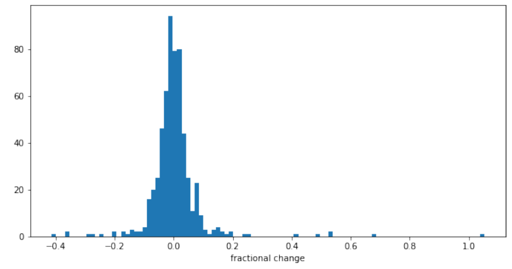
where π is the initial probability distribution, A is the transition matrix for the set of states Q, and E is the set of emission probabilities. $e_i(O_t)$ denotes the probability of observing observation O_t in state q_i. Since the emission probabilities are GMM's we denote this as

$$e_i(O_t) = \sum_{m=1}^{M} c_{im} N(O_t, \mu_{im}, \mathbf{\Sigma}_{im}). \quad (5)$$

Where M is the number of mixture components, c_{im} denotes the weight for mixture m in state q_i and last term is the probability of observing O_t within the multivariate normal distribution.



(1.1) IBM Fractional Change Distribution



(1.2) GME Fractional Change Distribution

Fig. 1. Differences in Ticker Fractional Change Distributions

Similar to [8], 3-dimensional observations, (2), are used. The model is trained using the baum-welch algorithm [10] which is a variation of expectation maximization (EM). All parameters of the model are learned from this algorithm. This enables the use of MAP estimation for fractional change of the next day. Given a set of observations $(O_1, O_2, ...O_d)$ and the model λ we can find

$$\hat{O}_{d+1} = \arg\max_{O_{d+1}} \left[P(O_{d+1} | O_1, O_2, ..., O_d, \lambda) \right]. \quad (6)$$

We can calculate $P(O_{d+1} | O_1, O_2, ..., O_d, \lambda)$ using the variable elimination (VE) algorithm [11]. We define the search space of O_{d+1} to be a set of 5,000 observations over the following ranges: *fracChange* fluctuates between -0.1 and 0.1 by a step size of 50, both *fracHigh* and *fracLow* fluctuate between 0.0 and 0.1 by a step size of 10, thus we have 50x10x10 candidates for O_t.

To implement this model we use the open source python library hmmlearn [12]. To perform the MAP estimation we compute the log likelihood values in parallel using 10 workers, this allows for a large speed increase from the model in [8] (15 minutes to test 90 points, down to 1 minute to test 90 points). Due to the speed increase, a grid search was performed over 784 possible sets of hyper-parameters. It was found that using 2 hidden states, 4 mixture components and 5 day latency resulted in the optimal model.

Our second proposed model is a CHMM. Following the principle of Occam's Razor, instead of using a GMM to represent the emission probabilities, a Gaussian distribution is used instead. Here we define the model λ as (4). The variables remain the same as described above, but now $e_i(O_t)$ is denoted as

$$e_i(O_t) = N(O_t, \mu_i, \sigma_i), \tag{7}$$

where $N(O_t, \mu_i, \sigma_i)$ is the probability of observing O_t in the normal distribution for state q_i. Instead of using a 3-dimensional observation, O_t, a 1-dimensional observation of the fractional change is used,

$$O_t = \left(\frac{close - open}{open}\right). \tag{8}$$

We found that the accuracy of the model was heavily dependent on the granularity of the observations being used in the MAP estimate. We validate that the GHMM is a good candidate by showing how the fractional change values are normally distributed in the training data, which can be seen in Figure 1. Notice in the figures that the spread of fractional change is heavily dependent on the stock. For this reason we found that using a hard coded set of observations as candidates for O_{d+1} was nonsensical. For this model we take the mean and standard deviation of the training data, and use this to sample 10,000 possible points from the distribution as candidates for O_{d+1}. Again, similar to the GMMHMM, we find the MAP estimation for O_{d+1} (6).

After finding the most likely next day observation O_{d+1}, we can use that days opening price to solve for our predicted closing price. For this model we use the following set of hyper-parameters: 2 hidden states and 5 day latency. The learning for the GHMM works the same way, we use the baum-welch algorithm [10]. Both the GMMHMM and GHMM are used for next day closing price predictions given the opening for a day.

Our third proposed model is a LSTM model (LSTM_ADV) that combines the strengths of papers [4, 5, 6] by using the 2 layer structure with 512 hidden recurrent nodes for each layer [4], the multidimensional input (2), along with a dense layer of 16 nodes feeding into the output dense layer [5]. Additionally we use dropouts with $p = 0.1$ in between our recurrent layers [6]. Data is normalized using minmax scaling

We also normalize our training y labels, meaning we have to inverse transform the prediction. The models hyper-parameters are the following: 20 day latency, minibatch size of 75, 50 epochs, MSE loss, Adam optimizer, and 0.001 learning rate. The LSTM layers use tanh activation and sigmoid recurrent activation. The first dense layer has sigmoid activation, and the output dense layer has linear activation.

For baseline accuracy comparison, three baseline models were created: a base RNN model, a base GRU model, and a base LSTM model. The network structure for these models is three recurrent layers, with 50 hidden units in each layer, feeding into a single dense layer that has the regression output. For these models, training is done using the same hyper-parameters as above but with 100 epochs, RMSprop optimizer, and 10-day latency. An additional deeper LSTM was also created with the same parameters above, however it has five recurrent layers (LSTM-DEEP). These models all use the fractional change of the day as the observation O_t (8).

3.2. Models for Raw Close Price Prediction

Our fourth proposed model is another LSTM that combines the strengths of papers [4, 5, 6]. Unlike the previous model, this model uses a 4 layer structure. The first 2 layers are LSTM layers with 128 and 64 hidden recurrent nodes, respectively. Then, a dense layer of 16 nodes feeds into the output dense layer. We include dropout layers with p = 0.3 after the first 3 layers. This model, along with our recreations of [4, 5, 6], are used for raw close price predictions.

The model's hyper-parameters are the following: 0.001 learning rate, 22-day latency, 500 epochs, minibatch size of 150, MAPE loss, and RMSProp optimizer. We use a validation split of 90/10%. Standardization scaling is used on the data. The LSTM layers use tanh activation and sigmoid recurrent activation. The 16 node dense layer uses Relu activation while the output dense layer uses linear activation. Our inputs for this model are the multidimensional inputs as seen in [5].

We recreated the models proposed in [4, 5, 6]. Unfortunately, information of the complete architecture is missing for each model. Thus, we had to use our best judgement to faithfully recreate these models. In the end, we believe we built them correctly to the best of our ability.

Majority of the hyper-parameters are the same as our fourth model. The few discrepancies include using 100 epochs instead of 500 for [4, 6]'s recreated models and using the input Open for [4] and Close for [6]. We kept these discrepancies as a way of remaining faithful to the papers.

3.3. Sentiment Analysis

Sentiment analysis of tweets, as a way to quantify mass sentiment for later use in uptick/downtick prediction, was also explored. A variety of neural network architectures utilizing word embedding encodings for input, as opposed to bag of word models, were built and tested. The use of word embedding provides better representation of tweets due to two primary factors: the vectorized floating point representation for words indicating a word's meaning as well as presence and the dissolution of sparse matrix representation for tweets. The Flair sentiment analysis library was used in generating labels for the data set fed to the models.

We produced two models that gave us fair results. One was a deep neural network (DNN) and the other was a convolutional neural network (CNN). The DNN was four layers deep. The first three layers had 100 nodes and Relu activation. The output had 2 nodes. All four layers were linear. The CNN used 128 filters at each convolutional layer with sizes of 3 and 8, respectively. The hidden layer had 128 nodes and 2 in the output layer.

Ticker	GMMHMM	GHMM	LSTM-ADV	LSTM-BASE	LSTM-DEEP	GRU-BASE	MAP HMM	ARIMA	ANN
AAPL	1.495	**1.504**	**1.561**	1.520	1.505	1.511	1.510	1.801	1.801
IBM	0.635	**0.647**	**0.638**	0.620	0.613	0.644	0.611	0.972	0.972

Table 1. Fixed Origin MAPE Model Comparison Fractional Change Forecasting

Ticker	GMMHMM	GHMM	LSTM-ADV	LSTM-BASE	LSTM-DEEP	GRU-BASE	RNN-BASE
AMZN	0.837	0.753	0.751	0.783	0.801	0.775	0.850
MSFT	0.752	0.677	0.696	0.719	0.685	0.716	0.700
GOOGL	0.754	0.688	0.696	0.791	0.728	0.720	0.763
DPZ	1.190	1.163	1.168	1.170	1.163	1.209	1.191
DIS	0.934	0.755	0.767	0.752	0.776	0.758	0.804
TMO	1.00	0.860	0.851	0.864	0.888	0.890	0.904
Avg MAPE:	0.911	**0.816**	**0.822**	0.855	0.840	0.845	0.869

Table 2. Rolling Window Average MAPE Model Comparison Fractional Change Forecasting

Ticker	Roondiwala Et Al	Pawar Et Al	Moghar Et Al	Behar
F	1.432	2.270	2.826	1.925
AMZN	8.542	4.006	11.751	7.264
MRK	3.538	2.301	7.391	3.753
Avg MAPE	4.503	**2.859**	7.323	**4.314**

Table 3. Rolling Window Average MAPE Model Comparison Raw Close Forecasting

Ticker	GHMM	Behar	Roondiwala et al	Pawar et al	Moghar et al	Buy_n_hold
AAPL	129.4	91.13	61.31	36.74	105.4	101.3
IBM	4.005	–	-07.87	-24.43	-05.85	.0841
AMZN	25.68	42.19	05.42	-17.79	43.02	58.69
MSFT	55.51	–	65.75	67.85	–	-44.82
GOOGL	15.86	03.06	15.98	-34.14	–	-26.43

Table 4. Backtesting results

4. RESULTS

4.1. Results for Fractional Change Prediction

The metric used to evaluate the models is Mean Absolute Percentage Error (MAPE), this is defined as

$$MAPE = \frac{1}{n}\sum_{i=0}^{n}\frac{|p_i - a_i|}{|a_i|} \cdot 100, \tag{9}$$

where p_i is our predicted closing price and a_i is the actual closing price. We test using two different methods, fixed origin and rolling window. Fixed origin tests use a static set of training data and testing data, then calculate the MAPE. Rolling window tests are a form of cross-validation for time series analysis. We define a large window, then use a chunk of it to train, and a smaller chunk directly after it, to test. We then shift the window by the size of the testing data, repeating this for 10 folds, and average the MAPEs.

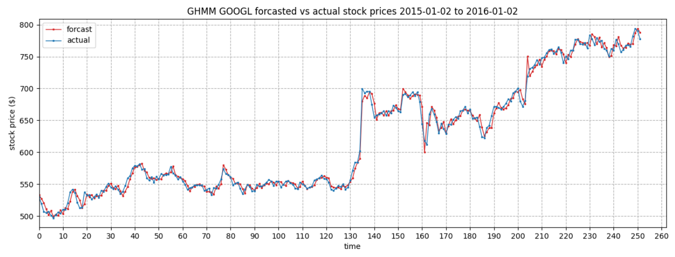
For comparison we use the MAPEs for the HMM, ARIMA, and ANN in [8, 13]. The fixed origin training dates are from 2003-02-10 to 2004-09-12, and the testing dates are from 2004-09-13 to 2005-01-22 for both AAPL and IBM. The dates used for the rolling window tests span from 2015-01-01 to 2020-01-01 for AMZN, MSFT, GOOGL, DPZ, DIS and TMO. Table 1 shows the MAPE results for the comparison and Table 2 shows the rolling window test results along with Average MAPE. Figure 2.1 shows the actual stock price along with the predicted price for the GHMM.

As seen in Table 1, the models generated in this paper perform similarly to the MAP HMM in Gupta et al [8] and well out perform the ARIMA and ANN in Ayodele et al [13]. With respect to the AAPL ticker, we see some minor improvements on accuracy from the GMMHMM, GHMM and LSTM-DEEP. However, when looking at the accuracy of IBM, we cannot make any statistically valid conclusions of improvement.

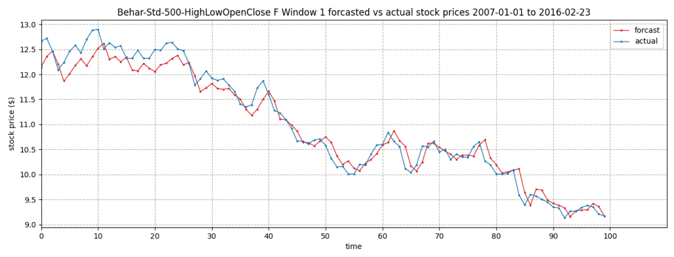
Assuming our GMMHMM version performs similar to the MAP HMM in [8] (based on Table 1), we can draw conclusions by looking at the MAPE in the rolling window test (Table 2). We see that the GHMM is significantly better than the GMMHMM, and we also see that our LSTM-ADV model which is built using the strengths from [4, 5, 6] also significantly out performs the GMMHMM. Our baseline models also perform comparably to the GMMHMM. With the results in Table 2 we can conclude that the GHMM and LSTM-ADV both perform better than the MAP HMM [8] and the ARIMA and ANN [13]. We believe the results for the GHMM are due to the intelligent choice of candidates for O_{d+1}.

4.2. Results for Raw Close Price Prediction

The rolling window tests for raw close price prediction range from 2007-01-01 to 2016-02-23 for F, AMZN, and MRK. As seen in Table 3 for F, [5]'s average MAPE score over 10 windows beats the other models. For AMZN and MRK, [4] is the clear winner. Taking the average, overall 3 tickers show us that [4]'s model performed the best. While our model, Behar, did not perform the best in any particular rolling window test, it performs second best overall.



(2.1) GHMM Forecasted v. Actual



(2.2) Behar Forecasted v. Actual

Fig. 2. Visual of model results.

Figure 2.2 is an example of what these raw close price predictions look like. At first glance, this looks like an excellent prediction. On further inspection, it should be noted there is an obvious horizontal shift within the graph between the prediction and actual. After performing due diligence, it was realized this is overfitting. Essentially, all four models learned to predict the price of (d_{t+1}) as (d_t)[1]. Out of 120 graphs created for all 4 models, many exhibited overfitting. We realized this problem early enough allowing us to implement some techniques to combat the overfitting. The Behar model implemented these techniques. Unfortunately, it was not enough and we still suffer from overfitting. In section 5, we discuss another possible method for reducing overfitting.

As we just stated, all 4 models overfitted. With that in mind, we question if [4, 5, 6] had overfitting issues that were not discussed in the papers. Or, we question if our recreations failed. Ultimately, there is not enough information in any of these papers to make a hard claim either way. Our intuition leads us to believe that the papers did suffer from overfitting.

4.3. Backtesting Models

The final test for our models was to utilize Fastquant's custom backtesting functionality. This allows us to use our models to buy and sell stocks. It works setting a threshold for buying and selling. If either threshold is met, the algorithm will buy or sell. If neither is met, the algorithm will hold the stocks. For our tests, we set the buy threshold to 3% and the sell threshold to 2%. That is, if the next day's price is higher than today's price by 3%, we will buy and if it is lower by 2%, we will sell. We did this for our GHMM and Behar models, along with the Moghar et al, Pawar et al, and Roondiwala et al model recreations. We used the following tickers: AAPL, IBM, AMZN, MSFT, GOOGL, along with the following dates range, 2019-0101 to 2020-05-31. The GHMM model consistently, withholding AMZN, beat the simply strategy of buying and holding. There were a few tests where the percentage gained was zero. We believe this happened because our model "flat lined." This happens when the training values are outside of the range of the testing values. Thus our model's predictions are capped at the maximum training value. See Table 5 for the full results.

4.4. Sentiment Analysis Results

Of the DNN architectures tested, the best resulting model achieved testing accuracy of 82.35%. The CNN architecture achieved testing accuracy of 89.56%. Because of limitations in data collection as the Twitter API limits the number of tweets that can be collected, this aspect of the project remains a proof of concept due to the lack of historical tweet data for comparison with stock prices.

5. CONCLUSION

Stock price forecasting is a challenging problem to solve. Any good solution that can give confidence in next day predictions is valuable. Our approach tackles attempts at forecasting next day prices using both fractional change prediction and raw close price forecasting. Fractional change prediction requires the knowledge of the day's opening prices, while raw close price forecasting requires the previous day's closing price. Our best models consist of the GHMM, the LSTM-ADV, and the Behar model. Each model is trained separately for each stock, since it is assumed that the price movement of one stock is independent from the other.

One main future improvement we briefly looked into was Gaussian smoothing over the data sets. This is an attempt to reduce the overfitting problem. The Gaussian smoothing will smooth out the volatility of the data. This should help the models learn the general trend of the data instead of the volatility. In our early tests with this idea, Guassian smoothing models beat Moghar et al [6]. We believe with more time and experimenting, we could improve this to outperform the other models. The use of Gaussian smoothing, as a technique to avoid overfitting in noisy signals, shows strong promise and should be researched further.

As a proof of concept, we used sentiment analysis to predict the closing price of the next day. Sentiment was found

[1]This realization of the models predicting the next days value as the previous day is in line with the assumptions of the efficient market hypothesis in its weak form. Due to incomplete information about a stock on a certain day, the best prediction for the next days price is the previous days price.

using the flair library, 1.4 million tweets were collected over a 12-day period and average sentiment for a ticker was calculated for each of the 12 days. Using a linear regression model, we attempted to predict the next day's fractional change using the previous day's average sentiment. However, due to a strong lack of data, no trend began to emerge. We believe that given more days of data, a trend would be learnable. This model was also used as a binary classifier for trend prediction, however again, due to lack of data strange results were found. Some tickers had 100% accuracy while others had 0% accuracy.

There are many other future improvements and research we would like to consider. One algorithm we did not explore, but shows promise, is Temporal Convolution Networks (TCN). TCNs use causal connections to weigh the importance of information depending on how long your input time series is (newer inputs hold more weight than older inputs) [14]. This could be advantageous because newer predictions can help the model learn the current volatility, while older predictions can help the model learn the overall trend of how the stock has moved in past years.

Another potential future improvement, could be to intelligently incorporate volume into the dimensions of the observations. An attempt for this was made but omitted from the paper due to poor results. The attempt used the fraction of total volume in the training set.

Recursive multistep forecasting for true prediction into the future is also a domain that would be interesting to see applied to stock price forecasting. However, it is still in active research [15, 16], and suffers from issues with compounding errors.

This project was a pleasure to work on. All of us learned a lot about best practices for time series analysis, along with gaining an intricate understanding of how the stock market works. It was interesting to learn how difficult this problem was to solve and we encountered many challenges as described throughout the paper. Overall, our project lines up well with our proposed approach, however after diving into the project and learning more about the domain we realized that some of the proposed methods were not valid methods for prediction. This in turn made us pivot to use some different models, i.e switching from Markov models with Monte Carlo simulations for inference to using CHMMs.

6. REFERENCES

[1] Lorenzo Ampil, "fastquant," GitHub.

[2] Muhammad Zubair Asghar, Fazal Rahman, Fazal Masud Kundi, and Shakeel Ahmad, "Development of stock market trend prediction system using multiple regression," Feb 2019, Springer US.

[3] Margaret Miró-Julià, Gabriel Fiol-Roig, and Andreu Pere Isern-Deyà, "Decision trees in stock market analysis: Construction and validation," in *Trends in Applied Intelligent Systems*, Nicolás García-Pedrajas, Francisco Herrera, Colin Fyfe, José Manuel Benítez, and Moonis Ali, Eds., Berlin, Heidelberg, 2010, pp. 185–194, Springer Berlin Heidelberg.

[4] Kriti Pawar, Raj Srujan Jalem, and Vivek Tiwari, "Stock market price prediction using lstm rnn," in *Emerging Trends in Expert Applications and Security*, Vijay Singh Rathore, Marcel Worring, Durgesh Kumar Mishra, Amit Joshi, and Shikha Maheshwari, Eds., Singapore, 2019, pp. 493–503, Springer Singapore.

[5] Murtaza Roondiwala, H. Patel, and Shraddha Varma, "Predicting stock prices using lstm," 2017.

[6] Adil Moghar and Mhamed Hamiche, "Stock market prediction using lstm recurrent neural network," 2020, vol. 170, pp. 1168–1173, The 11th International Conference on Ambient Systems, Networks and Technologies (ANT) / The 3rd International Conference on Emerging Data and Industry 4.0 (EDI40) / Affiliated Workshops.

[7] Sidra Mehtab, Jaydip Sen, and Abhishek Dutta, "Stock price prediction using machine learning and lstm-based deep learning models," 2020.

[8] Aditya Gupta and Bhuwan Dhingra, "Stock market prediction using hidden markov models," in *2012 Students Conference on Engineering and Systems*, 2012, pp. 1–4.

[9] Md. Rafiul Hassan, "A combination of hidden markov model and fuzzy model for stock market forecasting," 2009, vol. 72, pp. 3439–3446, Financial Engineering Computational and Ambient Intelligence (IWANN 2007).

[10] Stephen Tu, "Derivation of baum-welch algorithm for hidden markov models," 2015, Citeseer.

[11] L.R. Rabiner, "A tutorial on hidden markov models and selected applications in speech recognition," 1989, vol. 77, pp. 257–286.

[12] Sergei Lebedev, "hmmlearn," GitHub.

[13] Ayodele Ariyo Adebiyi, Aderemi Oluyinka Adewumi, and Charles Korede Ayo, "Comparison of arima and artificial neural networks models for stock price prediction," Mar 2014, Hindawi.

[14] Charlotte Pelletier, Geoffrey Webb, and François Petitjean, "Temporal convolutional neural network for the classification of satellite image time series," Mar 2019, vol. 11, p. 523, MDPI AG.

[15] Souhaib Ben Taieb and Gianluca Bontempi, "Recursive multi-step time series forecasting by perturbing data," in *2011 IEEE 11th International Conference on Data Mining*, 2011, pp. 695–704.

[16] Jie Liu and Enrico Zio, "Svm hyperparameters tuning for recursive multi-step-ahead prediction," Mar 2016, Springer London.