Hidden Markov Model and Stock Price Prediction

Bibhu Sapkota

2019

Advisors

Stephen R. Wassell
Giancarlo Schrementi

Department of Mathematics and Statistics
Hollins University

Contents

1 Abstract

Stock markets are very unpredictable. There are several uncertain states such as macroeconomic policies, demand and supply, launch of new products, etc., that affect the stock price of a company. In this project, we will learn about the Hidden Markov Model, Forward Algorithm, Backward Algorithm, Viterbi Algorithm, and Baum-Welch Algorithm. We will focus on using a Hidden Markov Model to predict the state of the stock price of several companies and compare it with the actual price movements to see how accurate the model is in predicting the stock prices. We will then come up with investment strategies using the states to see what profit can be made using this model. We compare the profit from our model with the baseline profit to check the effectiveness of our model. We will use the concept of Machine Learning where we will generate the training set and testing set for the Hidden Markov Model and then use it to predict the states of historical daily stock price of these companies. We will be using the Python hmmlearn package to implement the Hidden Markov Model.

2 Acknowledgements

I would like to express my deep and sincere gratitude to my research advisor, Dr. Stephen R. Wassell, for invaluable guidance and encouragement throughout this research process and paper creation. I am also very grateful to Dr. Giancarlo Schrementi for sharing expertise, and guidance throughout the research process. I would also like to thank Dr. Julie M. Clark and the entire Mathematics and Statistics department for their unfailing support and continuous encouragement throughout my years of study at Hollins University. Any of my accomplishments at Hollins and beyond would not have been possible without them. Thank you.

3 Introduction

3.1 Hidden Markov Model

A Markov chain is a model that tells us something about the probabilities of sequences of random variables, states, each of which can take on values from some set. These sets can be words, or tags, or symbols representing anything, like the weather. A Markov chain makes a very strong assumption that if we want to predict the future in the sequence, all that matters is the current state. A Markov chain is useful when we need to compute a probability for a sequence of observable events. In many cases, however, the events we are interested in are hidden: we don't observe them directly. A Hidden Markov model (HMM) allows us to talk about both observed events and hidden events that we think of as causal factors in our probabilistic model. [5]

The basic elements of Hidden Markov Model are:

- Length of observation data, T
- Number of states, N
- Number of symbols per state, M
- Observation sequence, $O = \{O_t, t = 1, 2, \ldots, T\}$
- Hidden state sequence, $Q = \{q_t, t = 1, 2, \ldots, T\}$
- Possible values of each state, $\{S_i, i = 1, 2, \ldots, N\}$
- Possible symbols per state, $\{v_k, k = 1, 2, \ldots, M\}$
- Transition matrix, $A = (a_{ij})$, where $a_{ij} = P(q_t = S_j | q_{t-1} = S_i)\,,\ i, j = 1, 2, \ldots, N$
- Vector of initial probability of being in state (regime) S_i at time $t = 1$, $p = (p_i)$, where $p_i = P(q_1 = S_i)\,,\ i = 1, 2, \ldots, N$
- Observation probability matrix, $B = (b_{ik})$, where $b_{ik} = P(O_t = v_k | q_t = S_i)\,,\ i = 1, 2, \ldots, N$ and $k = 1, 2, \ldots, M$.

In summary, the parameters of an HMM are the matrices A and B and the vector p. For convenience, we use a compact notation for the parameters, given by:

$$\lambda \equiv \{A, B, p\}.$$

In summary, the parameters of an HMM are the matrices A and B and the vector p. For convenience, we use a compact notation for the parameters, given by:

$$\lambda \equiv \{A, B, p\}.$$ [1]

We can better try to understand these basic elements using the figure below:

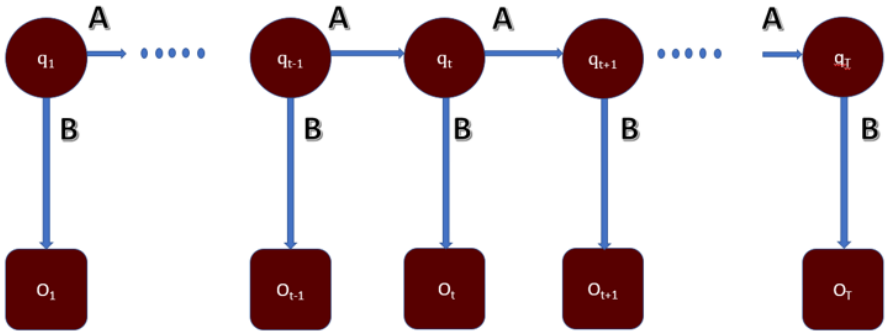

Figure 1: Hidden Markov Model

When working with Hidden Markov Model, we come across three fundamental problems:

1. Given the observation data and the model parameters, how do we compute the probabilities of the observations?

2. Given the observation data and the model parameters, how do we choose the best corresponding state sequence?

3. Given the observation data, how do we calibrate HMM parameters?

These fundamental problems are solved using forward algorithm, Viterbi algorithm and Baum–Welch algorithm respectively.

3.2 Forward Algorithm

We define the joint probability function as $\alpha_t(i) = P(O_1, O_2, \ldots, O_t, q_t = S_i|\lambda)$, then calculate $\alpha_t(i)$ recursively. The probability of observation $P(O|\lambda)$ is just the sum of the $\alpha_T(i)\,s$.

The forward algorithm.

1: Initialization: for i=1,2,..., N

$$\alpha_{t=1}(i) = p_i b_i(O_1).$$

2: Recursion: for $t = 2, 3, \ldots, T$, and for $j = 1, 2, \ldots, N$, compute

$$\alpha_t(j) = \left[\sum_{i=1}^{N} \alpha_{t-1}(i)\, a_{ij}\right] b_j(O_t).$$

3: Output:

$$P(O|\lambda) = \sum_{i=1}^{N} \alpha_T(i).$$

[1]

The forward algorithm solves the first problem of Hidden Markov Model.

3.3 Backward Algorithm

Similar to the forward algorithm, we define the conditional probability $\beta_t(i) = P(O_{t+1}, O_{t+2}, .., O_T | q_t = S_i, \lambda)$, for $i = 1, \ldots, N$. Then, we have the following recursive backward algorithm.

[1]

The backward algorithm.

1: Initialization: for $i = 1, \ldots, N$

$$\beta_T(i) = 1.$$

2: Recursion: for $t = T-1, T-2, \ldots, 1$, for $i = 1, \ldots, N$

$$\beta_t(i) = \sum_{j=1}^{N} a_{ij} b_j(O_{t+1}) \beta_{t+1}(j).$$

3: Output:

$$P(O|\lambda) = \sum_{i=1}^{N} p_i b_i(O_1) \beta_1(i).$$

3.4 Viterbi Algorithm

The Viterbi Algorithm helps to find the best sequence of states when the observation data and the model parameters are provided. The second problem has many solutions and among these solutions, we need to find the one with the "best fit". We define:

$$\delta_t(i) = \max_{q_1, q_2, \ldots, q_{t-1}} P(q_1, q_2, \ldots, q_t = S_i, O_1, \ldots, O_t|\lambda).$$

By induction, we have:

$$\delta_{t+1}(j) = \max_i \left[\delta_t(i) a_{ij}\right] b_j(O_{t+1}).$$

Using $\delta_t(i)$, we can solve for the most likely state q_t, at time t, as:

$$q_t = \text{argmax}_{1 \le i \le N} \left[\delta_t(i)\right], \ 1 \le t \le T.$$

Thus, the Viterbi algorithm is given below.

The Viterbi algorithm.

1: Initialization:

$$\delta_1(j) = p_j b_j(O_1), \quad j = 1, 2, \ldots, N;$$
$$\phi_1(j) = 0.$$

2: Recursion: for $2 \leq t \leq T$, and $1 \leq j \leq N$

$$\delta_t(j) = \max_i \left[\delta_{t-1}(i)\, a_{ij}\right] b_j(O_{t+1})$$
$$\phi_t(j) = arg\ \max_i \left[\delta_{t1}(i)\, a_{ij}\right]$$

3: Output:

$$q_T^* = \mathbf{argmax}_i \left[\delta_T(i)\right]$$
$$q_t^* = \phi_{t+1}\left(q_{t+1}^*\right),\ t = T-1, \ldots, 1$$

[1]

3.5 Baum-Welch Algorithm

Baum-Welch algorithm helps us find the answer to the third problem of Hidden Markov Model. We have to find the parameters to maximize the probability of the observation data. Unfortunately, with the given observation data, there is no way to find the global maximum of probability. However, we can choose the parameters such that probability is locally maximized using Baum-Welch iterative method, which used the maximum likelihood estimator to train the model parameters. [1]

We define:

$$\gamma_t(i) = P(q_t = S_i|O,\lambda) = \frac{\alpha_t(i)\beta_t(i)}{P(O,\lambda)} = \frac{\alpha_t(i)\beta_t(i)}{\sum_{i=1}^{N}\alpha_t(i)\beta_t(i)}.$$

The probability of being in state S_i at time t and state S_j at time $t+1$, $\xi_t(i,j)$ is defined as:

$$\xi_t(i,j) = P(q_t = S_i, q_{t+1} = S_j|O,\lambda) = \frac{\alpha_t(i)a_{ij}b_j(O_{t+1})\beta_{t+1}(j)}{P(O,\lambda)}.$$

Clearly,

$$\gamma_t(i) = \sum_{j=1}^{N}\xi_t(i,j).$$

The Baum–Welch algorithm

1: Initialization: input parameters λ, the tolerance tol, and a real number

2: Repeat until $\triangle < tol$

- Calculate $P(O,\lambda)$ using forward algorithm in Section 2.1
- Calculate new parameters λ^*: for $1 \le i \le N$

$$p_i^* = \gamma_1(i)$$
$$a_{ij}^* = \frac{\sum_{t=1}^{T-1}\xi_t(i,j)}{\sum_{t=1}^{T-1}\gamma_t(i)},\ 1 \le j \le N$$
$$b_{ik}^* = \frac{\sum_{t=1,O_t=v_k}^{T}\gamma_t(i)}{\sum_{t=1}^{T}\gamma_t(i)},\ 1 \le k \le M$$

- Calculate $\triangle = |P(O,\lambda^*) - P(O,\lambda)|$
- Update $\lambda = \lambda^*$

3: Output: parameters λ.

---

[1]

4 Methods

For the research, we use machine learning concepts to implement the Hidden Markov Model for predicting the states of the stock price of Coca Cola company. We code in Python and use hmmlearn, a built-in package, to train, test, and predict our Hidden Markov Model for our data. The data is comprised of the closing price of Coca Cola, Tesla, and Facebook from March 27, 2017 to March 26, 2019. This data is split into training and testing datasets. The training data for each company comprises the closing price from March 27, 2017 to August 29, 2018 and the testing data for each company comprises the closing price from August 20, 2018 to March 26, 2019.

Because our machine learning algorithm requires the data to be within a given range, whereas the stock market is optimistic so that prices tend to increase over time, we converted the prices to percent change. To smooth out the data somewhat in an attempt to get better results from the machine learning algorithm, we calculate a 5-day rolling standard deviation and 5-day rolling mean. The Hidden Markov Model is fitted into the training data and predicted in the testing data.

We use the model means to figure out the correspondence of predicted states in the model with the nature of change to come up with investment strategies from the model. Here, for each company, there are 4 different states which we try to match to the nature of change based on the model means of that model. For each company, there are 2 good states which consists of positive mean price change showing us a rise. Among these 2 good states, we have a best state with high volatility and high mean price change. The other 2 are bad states which consists of negative mean price change showing us a drop. We came up with two different investment strategies:

1. The investor will invest when the states change from a bad state to a good state and sells when the states change from a good state to a bad state.

2. The investor will only invest when the states change to the best state and sell at any other states.

For both of these investment strategies, the first buy of the stock occurs when the stock enters its necessary required state. After that the stock is sold and bought based on the criteria of the strategy. This leads to several buys and sells of the stock throughout the testing period. We also assume that we sell all our stock by the end for strategy comparison purpose.

5 Discussion

We began our research by studying various research projects conducted for analyzing the stock market/price using a Hidden Markov Model. During this process, we came across a Python implementation by authors Ayush Jain, Prateek Mehta and Ravi Teja Darba, Engineering candidates from Boston University, which predicted the last 100 days stock price for various stocks and compared it with their actual stock price. We used this code to understand the Hidden Markov Model and its efficiency in predicting stock price. We used their coding to predict Facebook, Inc.'s (FB) open, close, high, low price for the last 100 days using Hidden Markov Model and compared it with the actual price. The results we had were good as there was extensive overlap between the actual price and the predicted price, which can be seen in the figures below:

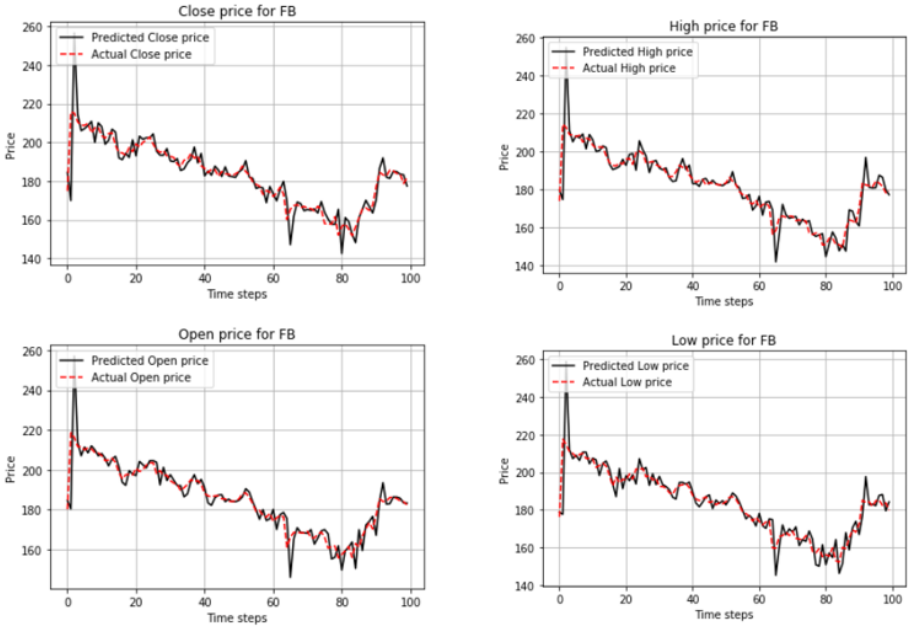

Figure 2: Predicted and Actual Price for Facebook, Inc.

We then decided to code our own simple version of stock state prediction using the hmmlearn package, where we used Coca Cola's (KO) closing price to train, test, and predict the states using the Hidden Markov Model. This initial version provided us with predicted states that only captured very drastic changes in the price. Below is a figure of the states predicted and percent change of price of Coca Cola. We can see that the states change only

when there is drastic change in the percent change.

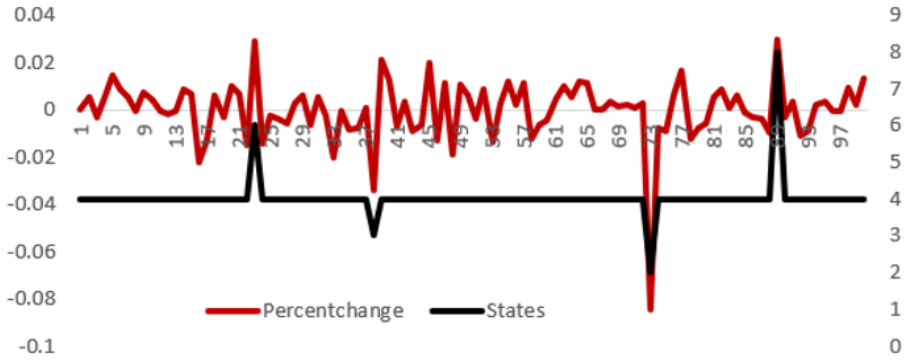

Figure 3: Percent Change and Predicted States for Coca Cola (Version 1)

To better fit the data and have better prediction for the states of the market, in the next version of coding we used 5-day rolling standard deviation and 5-day rolling mean for fitting and prediction of our Hidden Markov Model. We used 5-day rolling standard deviation and 5-day rolling mean (requiring a two-dimensional use of the Hidden Markov Model) to capture both the spread and the direction of change in the stock price for both training and testing dataset. We achieved better results for the predicted states.

In the figure below, we can see that our predicted states are better capturing the changes in the stock price. Though we have better predicted states, it is very difficult to comprehend what states correspond to what changes in price of the stock just looking at the graph below.

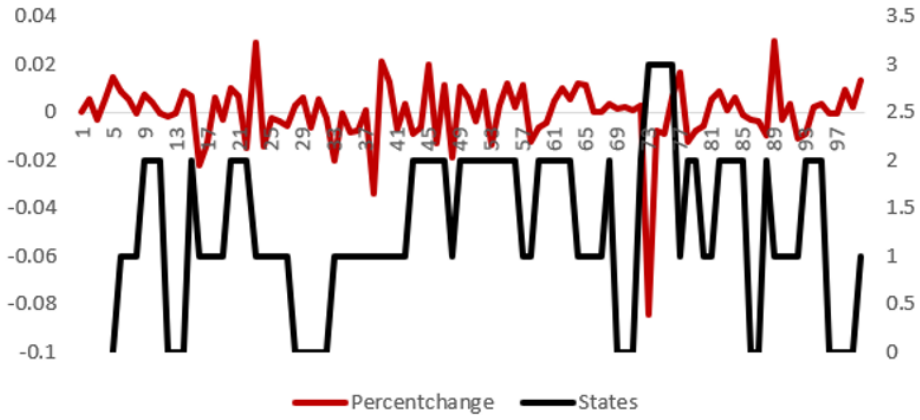

Figure 4: Percent Change and Predicted States for Coca Cola (Version 2)

To overcome this problem, we used the model means. The model means tells us what each state corresponds to. For example, if state 0 has model means whose 5-day rolling standard deviation is high and 5-day rolling mean is negative, we know that the given state will correspond to the phases when there is similar drop in price. Further, after getting the corresponding states with rise/fall phases, we use our investment strategies to see if our model will provide us profit. We also conduct the same process for other two stocks: Tesla and Facebook, to see the effectiveness of our model.

We can see the figures below for Tesla and Facebook with the percent change in price and the states predicted by our model

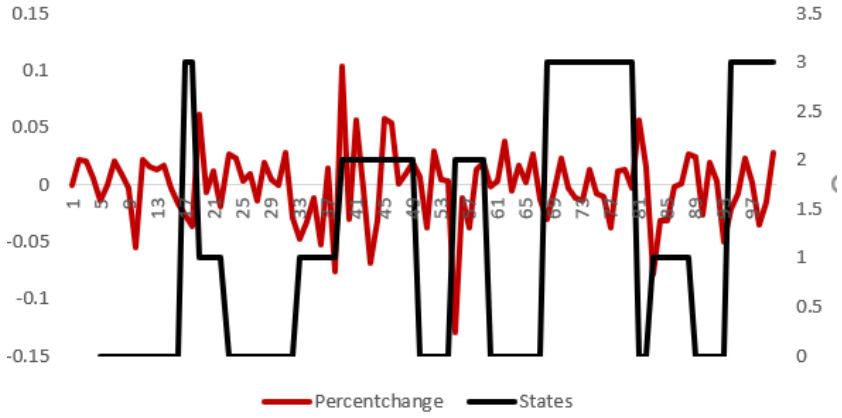

Figure 5: Percent Change and Predicted States for Tesla

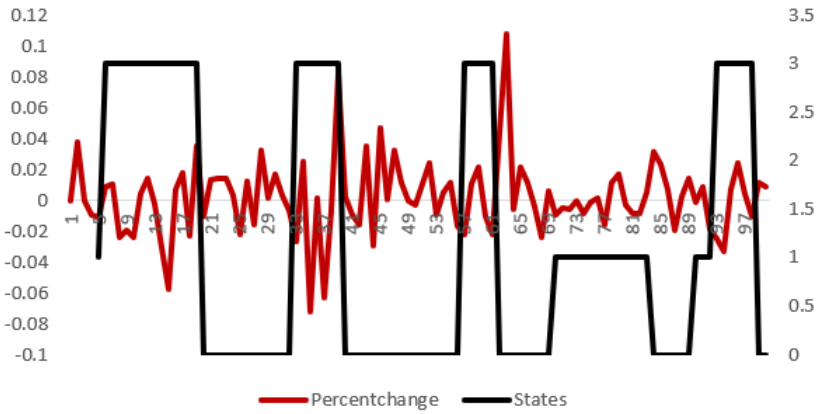

Figure 6: Percent Change and Predicted States for Facebook, Inc.

6 Results

Our model includes two strategies: Strategy 1 where the investor will invest when the states change from a bad state to a good state and sells when the states change from a good state to a bad state, and Strategy 2 where the investor will only invest when the states change to the best state and sell at any other states. The model means for Coca Cola are shown as below; we used this information to determine what each state corresponds to and develop appropriate Python code for our investment strategies.

5-day rolling Standard Deviation	5-day rolling Mean
0.83086712	0.08174457
0.44871797	0.27675396
0.45377605	-0.10517509
1.27097385	-0.32142345

Our python coding for investment strategies produced the following result for Coca Cola (KO)

	Training Data	Testing Data
Strategy 1 Profit	$8.27	$0.209
Strategy 2 Profit	$4.39	-$3.21
Strategy 1 Times Sold and Bought	17	4
Strategy 2 Times Sold and Bought	17	3
Baseline profit	$4.29	-$2.05

From the above table, we can see that our model is doing well in comparison to the baseline profit except for strategy 2 testing data. This tells us that our model is not perfect and will not always result in profit. We also expect to have frequent selling and buying in Strategy 1 compared to Strategy 2 as we will be buying and selling in two different states, respectively, which increases the frequency as compared to Strategy 2, which only buys at one state. It is also very logical to have less frequency of selling and buying in testing data as the data only consists of 1/3 of the training data.

We conduct the same strategies for our other two stocks, Tesla and Facebook.

Below are the results obtained from our code for Tesla and Facebook:

5-day rolling Standard Deviation(TSLA)	5-day rolling Mean (TSLA)
2.3141335	0.86237314
3.72875387	-1.46269884
6.57741304	1.68274287
1.81260506	-0.49052039

5-day rolling Standard Deviation(FB)	5-day rolling Mean (FB)
2.02950637	0.66539574
0.90240355	0.2416679
8.64992451	-3.63242561
2.01429819	-0.69923479

Our Python coding for investment strategies produced the following results for Tesla and Facebook.

Training Data	Tesla	Facebook, Inc.
Strategy 1 Profit	$103.19	$11.29
Strategy 2 Profit	$85.77	$25.37
Strategy 1 Times Sold and Bought	20	15
Strategy 2 Times Sold and Bought	6	13
Baseline profit	$10.2	$3.98

Testing Data	Tesla	Facebook
Strategy 1 Profit	-$62.019	$31.91
Strategy 2 Profit	$15.29	$36.27
Strategy 1 Times Sold and Bought	4	4
Strategy 2 Times Sold and Bought	2	5
Baseline profit	-$73.63	$19.0

From the above tables, we can see that our model is making profit for both strategies for both training and testing data.

7 Limitations and Further Scope for Research

We came across various limitations throughout the research process. First and foremost was manipulation of data for better training. Starting with one dimensional input which was either price or percent change of price, our scores for the test were very large telling us that the model wasn't very good at predicting the states. We overcame this limitation by using two-dimensional data, i.e. training and testing on the 5-day rolling mean and 5-day rolling standard deviation, so that our model accounts both the direction of change and the spread of the change.

Another main limitation was predicting the actual next price in the sequence. With four given states, there are four different model means and a 4 by 4 transition matrix. This will only give us four different stock prices, and with the change in states, the price will only fluctuate between these four stock prices rather than providing us the actual stock price. Further research lies in using other statistical concepts along with the Hidden Markov Model as authors of other articles have done to figure out the next observation (next day price of the stock) and compare it with the actual price to determine the effectiveness of the Hidden Markov Model.

S and P 500 was having its worst fourth quarter in 2018. Looking at our baseline profits, we can see that we trained in relatively in bear market and predicted on the bull market. Further research scope lies in using a training data in bull market and/or using a testing data on bear market. This might increase the score of our model and provide better predictions of the states of the stock price.

8 Conclusion

The mail goal of this research was to understand and learn Machine Learning concepts. Throughout the research project, we came across several limitations about manipulating the data to better fit the model and get better predictions. This strengthened my knowledge and confidence about data cleaning and manipulation. The project also taught us that a given model might work for one dataset but might not work for the other. We came up with two different investment strategies at the end of our research. This investment strategies do give us profit. This being said, our investment strategies are not 100 percent foolproof as the stock market is very volatile and our model isn't strong enough to capture all the volatility in the stock market.

References

[1] N. Nguyen and D. Nguyen, "Hidden market model for stock prediction," *Risks*, 2015.

[2] R. Hassan and B. Nath, "Stock market forecasting using hidden markov model: a new approach," *International Conference on Intelligent Systems Design and Applications*, 2005.

[3] Y. Zhang, "Prediction of financial time series with hidden markov models," Master's thesis, Simon Fraser University, 2004.

[4] A. Gupta and B. Dhingra, "Stocks market prediction using hidden markov models," *IEEE*, 2012.

[5] L. Baum and T. Petrie, "Statistical inference for probabilistic functions of finite state markov chains," *Ann. Math. Statist*, 1554.

[6] A. Jain and P. Mehta, "Stock forecasting using hidden markov models," *Boston University*, 2018.

9 Appendix

This section shows the python code that we used for our analysis with the data of Coca Cola. Exact same code was used for data of Tesla and Facebook for our analysis.

Manipulating the data to get 5-day rolling Standard Deviation and 5-day rolling Mean for training data:

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from hmmlearn import hmm
```

```
stocktrain = pd.read_csv("kotrain.csv")
stocktest = pd.read_csv("kotest.csv")
```

```
perchange = stocktrain.loc[:,"Percentchange"]
price = stocktrain.loc[:,"Price"]
perchange_std = perchange.rolling(window=5).std()
perchange_mean = perchange.rolling(window=5).mean()
price_std = price.rolling(window=5).std()
price_mean = price.rolling(window=5).mean()
newdf = pd.DataFrame({"Perchange Std" : perchange_std[4:]*100,
                      "Perchange Mean" : perchange_mean[4:]*100,
                      "Price Std" : price_std[4:],
                      "Price Mean" : price_std[4:]
                     })
newdf.head()
```

	Perchange Std	Perchange Mean	Price Std	Price Mean
4	0.305355	0.05700	0.100598	0.100598
5	0.310252	0.04286	0.079056	0.079056
6	0.406837	0.11820	0.122475	0.122475
7	0.439402	0.07138	0.112916	0.112916
8	0.408445	0.03350	0.125817	0.125817

Manipulating the data to get 5-day rolling Standard Deviation and 5-day rolling Mean for testing data:

```
perchange1 = stocktest.loc[:,"Percentchange"]
price1 = stocktest.loc[:,"Price"]
perchange1_std = perchange1.rolling(window=5).std()
perchange1_mean = perchange1.rolling(window=5).mean()
price1_std = price1.rolling(window=5).std()
price1_mean = price1.rolling(window=5).mean()
testdf = pd.DataFrame({"Perchange Std" : perchange1_std[4:]*100,
                       "Perchange Mean" : perchange1_mean[4:]*100,
                       "Price Std" : price1_std[4:],
                       "Price Mean" : price1_std[4:]
                      })
testdf.head()
```

	Perchange Std	Perchange Mean	Price Std	Price Mean
4	0.660137	0.44292	0.416616	0.416616
5	0.627348	0.61544	0.588837	0.588837
6	0.627188	0.61634	0.699835	0.699835
7	0.553398	0.65862	0.570964	0.570964
8	0.549806	0.69160	0.366715	0.366715

```
model = hmm.GaussianHMM(n_components=4)
fit = model.fit(newdf.iloc[:,0:2])

pred = model.predict(newdf.iloc[:,0:2])
pred1 = model.predict(testdf.iloc[:,0:2])

score = model.score(newdf.iloc[:,0:2])
score1 = model.score(testdf.iloc[:,0:2])

print(pred)
print(pred1)

print(score)
print(score1)
```

Fitting the Hidden Markov Model on our training data and predicting the states on both training and testing data:

```
[2 2 2 2 2 2 2 2 1 1 1 1 1 1 1 1 2 2 2 2 2 2 2 1 1 1 2 2 2 2 2 1 1 1 1 1 1
 1 1 1 1 1 1 1 1 1 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 1 1
 1 1 1 1 1 1 1 1 1 1 1 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 1 1 1 1
 1 1 1 1 0 0 0 0 0 2 2 2 2 0 0 0 0 0 1 1 1 1 1 1 1 1 1 1 1 2 2 2 2 2 2 2 2
 2 2 2 2 2 0 0 0 0 0 1 1 0 0 3 3 3 3 3 3 2 2 2 2 2 1 1 1 0 0 0 0 0 0 0 0 0 2 2
 2 2 2 2 2 2 2 0 0 0 0 0 1 1 1 1 1 1 1 1 1 1 1 3 3 3 3 3 3 3 3 3 3 3 3 3 3
 3 3 3 3 3 3 3 0 0 0 0 1 1 1 1 0 0 3 3 3 3 3 2 2 2 2 0 0 0 3 3 3 3 3 3 3 2
 2 1 1 1 0 0 0 0 3 3 3 3 3 3 3 3 3 3 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1
 0 0 0 0 0 1 1 0 0 3 3 3 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 0 0 0 0
 0 0 0 2 2 2 2 2 2 2 2 2 2 2 2 1 1 0 0 0 2 2 2 2 2 2 2 0 0 0 1 1 1 0 0 0 0
 0 0 0 0 0 0 0 0 1 1 0 0 0 0 0 0 3 3 3 3 3 0 0 0 0 0 0 0 0]
[1 1 1 1 1 1 1 1 1 1 1 3 3 3 3 3 3 3 3 3 3 3 3 2 2 2 2 2 0 3 3 3 3 3 3 3 3
 3 3 3 3 3 3 3 3 3 3 3 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 1 3 3 3 3 3 3
 3 3 3 0 0 1 1 1 1 0 3 3 3 3 3 2 2 2 2 1 1 1]
37.311824943354885
-273.02318598787815
```

Finding the model means and assigning states:

```
model.means_
```

```
array([[ 0.83086712,  0.08174457],
       [ 0.44871797,  0.27675396],
       [ 0.45377605, -0.10517509],
       [ 1.27097385, -0.32142345]])
```

```
high_std_highneg_mean = 2
high_std_lowpos_mean = 0
med_std_highpos_mean = 1
med_std_lowneg_mean = 3
```

Investment Strategy 1:

```
#Main Strategy 1

last_price = price[4]
profit = 0
bought = False
times_bought = 0
times_sold = 0
for i in range(1,len(pred)):
    curr_price = price[4+i]
    state = pred[i]
    prev_state = pred[i-1]
    if (state != med_std_highpos_mean and state != high_std_lowpos_mean)
        #sell
        profit += curr_price - last_price
        bought = False
        times_sold += 1
    elif (state == med_std_highpos_mean or state == high_std_lowpos_mean)
        #buy
        last_price = curr_price
        bought = True
        times_bought += 1
if bought:
 # sell if still holding on
    profit += curr_price - last_price
    bought = False
    times_sold += 1

print(times_bought)
print(times_sold)
profit
```

```
17
17

8.270000999999993
```

Investment Strategy 2:

```
#Main Strategy 2

last_price = price[4]
profit = 0
bought = False
times_bought = 0
times_sold = 0
for i in range(1,len(pred)):
    curr_price = price[4+i]
    state = pred[i]
    prev_state = pred[i-1]
    if state != med_std_highpos_mean and prev_state == med_std_highpos_mean and bought:
        #sell
        profit += curr_price - last_price
        bought = False
        times_sold += 1
    elif state == med_std_highpos_mean and prev_state != med_std_highpos_mean and not bought:
        #buy
        last_price = curr_price
        bought = True
        times_bought += 1
if bought:
    # sell if still holding on
    profit += curr_price - last_price
    bought = False
    times_sold += 1

print(times_bought)
print(times_sold)
profit
```

```
17
17

4.390008000000016
```

```
baselineprofit = price[399]-price[4]
baselineprofit
```

```
4.290000999999997
```