Stock Prediction using ML Algorithms

*Note: Comparision of various machine learning algorithms for stock prediction

1st Akshata Deo
Department of Computer Engineering
San Jose State University
San Jose, USA
akshata.deo@sjsu.edu

2nd Charlie Brayton
Department of Computer Engineering
San Jose State University
San Jose, USA
charles.brayton@sjsu.edu

3rd Dikshita Borkakati
Department of Computer Engineering
San Jose State University
San Jose, USA
dikshita.borkakati@sjsu.edu

4th Priyanshi Jajoo
Department of Software Engineering
San Jose State University
San Jose, USA
priyanshi.jajoo@sjsu.edu

***Abstract*—AIML techniques are being used with data mining to solve a plethora of real-world problems, and the prediction of the stock market is one of them. The stock market is one of the most complicated and lucrative businesses. Many other businesses depend on the stock market to generate revenue. Today, not only traders but Computer Engineers are also highly interested in predicting stock price. This can be done in two ways. First, by using previous stock data, and second, by analysing social media behaviour. However, this paper proposes to use various machine learning algorithms like Naive Bayes, ARIMA, neural network, and SVR to predict future stock prices using previous stock data. These ML algorithms use different open-source libraries and preexisting algorithms to make this highly unpredictable stock market, little more predictable. In comparison to the traditional approach, this technique has proven to be more accurate, and it saves time and money also. The implementation of these algorithms shows some good results at the end. Further, this paper suggests some measures that can be taken to improve the accuracy of the models.**

***Index Terms*—stock market, share market, ML algorithms, data mining, Naive Bayes, ARIMA, neural network, SVR.**

I. Introduction

The stock market or financial market is a system where people buy or sell stocks. A stock is commonly known as shares and it represents ownership claims on a business by a particular person. Predicting stock trends is one of the cumbersome tasks, as the stock market is very dynamic and composed of various factors. The main factors involved are Physical, Psychological, Rational, Irrational behavior, etc. These factors can be grouped into fundamental factors, technical factors and market sentiments like [1]:

- Supply and demand. For example, if traders tend to buy a particular stock more than selling it, this will affect stock price.
- Stock prices can show unexpected outcomes because of an announcement which keeps a stock artificially high or low.
- Global economical trends also affect stock prices. The more liquidity in the market the more it will be dynamic.
- Global disasters, pandemic, and any emergency conditions affect stock prizes. For an example, in recent days, pandemic COVID-19 is severely affecting stock market.
- Stock historical prices. Each stock value varies in between a particular range of data and this allows investors to understand the pattern of the stock price.
- Inflation and deflation are other technical factors that affect the global economy and indirectly affect stock prices.

Stock prediction needs to be accurate and efficient, in order to help people invest in the stock market. This paper employs different Machine Learning Algorithms on Stock Market data to analyze the trends of the stock market. The stock market prediction performs better when we consider it as a regression problem as well as a classification problem. Considering these facts, our goal is to design a model that gains insight from the market information utilizing machine learning strategies, then predict the future patterns of stock value. This paper discusses Support Vector Machines analysis[2] models which are linear, polynomial, and radial basic function; Tensor Flow that is a Neural Network computation framework[3]; Auto ARIMA (Auto-regressive integrated moving average) model[4] and Naive-Bayes clustering model[5], in which both Gaussian and Bernoulli versions are used. The rest of the sections of this paper will provide a detailed description of data-sets, methods, and results.

II. DATASET

The data-set is downloaded from Alpha Vantage for Microsoft and Adobe Stocks for the last 20 years. All the required libraries like request, JSON, CSV, etc have been imported for fetching the data from Alpha Vantage.The historical stock data is fetched into a csv file using an API key and passing parameters such as the ticker symbol for the company stock, 'TIMESERIESDAILY' for the function parameter and 'csv'

for the datatype parameter. The csv data is then converted into dataframe using pandas library. The dataframe has six features: timestamp, open, high, low, close and volume. Fig 1 shows the code snippet and the raw data after fetching.

```
# Collecting stock data of Adobe using Alpha Vantage API
CSV_URL ="https://www.alphavantage.co/query?function=TIME_SERIES_DAILY&symbol=ADBE&apikey=NR8HXOOB06DUO2M3
&datatype=csv&outputsize=full"
with requests.Session() as s:
    download = s.get(CSV_URL)
    print(download)
    decoded_content = download.content.decode('utf-8')

    cr = csv.reader(decoded_content.splitlines(), delimiter=',')
    df = pd.DataFrame(cr)
```

	timestamp	open	high	low	close	volume
1	2020-04-27	348.8000	349.9600	345.1200	348.5000	2361714
2	2020-04-24	338.0000	344.7000	333.5011	344.1000	2146162
3	2020-04-23	338.9000	341.7500	334.8000	335.3700	2120251
4	2020-04-22	337.4100	339.3800	332.8006	336.7700	2231819
5	2020-04-21	340.9000	342.7630	325.2100	328.9900	3548202

Fig. 1. Dataset from Alpha Vantage

III. METHODS

A. Neural Network

a) Importing Dataset: Fig2 shows the raw dataset after fetching from the Alpha Vantage.

	timestamp	open	high	low	close	volume
1	2020-04-27	59.3300	60.9200	59.0900	60.0500	286271
2	2020-04-24	58.5000	59.0900	57.2100	58.8000	747760
3	2020-04-23	56.7900	58.7100	56.7900	58.0500	585525
4	2020-04-22	55.7600	57.5150	55.1000	56.8700	583727
5	2020-04-21	56.6400	56.9000	54.3000	54.3600	458041

Fig. 2. Raw Dataset

b) Data Preprocessing: Dataset has been checked for duplicate, null and missing values. It gave false output, which means the dataset was already clean. One extra column has been added in the dataset with the average of high and low price of the stock. 'timestamp' column has been deleted, since it does not add any value while predicting stock price. Besides, 'timestamp' has dtype as a string, which might create errors while training the model, so it is better to drop this column. Now the data frame has been converted to a NumPy array to divide the whole dataset into training and testing dataset. The dataset has been divided sequentially like the first 80 percent of the data has been reserved for training purposes, and the rest of the 20 percent data has been reserved for testing purposes. Then, data scaling has been performed on the input and output data by importing the MinMaxScaler function from sklearn.preprocessing library. Data scaling should not be performed before categorizing training and testing dataset. Sequence should be like categorization of the data, then scaling of traning data and then scaling of testing data. Doing so helps model to avoid overfit. Finally, X and Y axis for training and testing datasets have been built, where X shows the input data, which indicates the price of the stock at a time 't' and Y shows the output data, which indicates the price of the stock at a time 't+1'.

c) Data Visualization: Fig3 shows the visualization of the dataset after preprocessing and before implemention of the model.

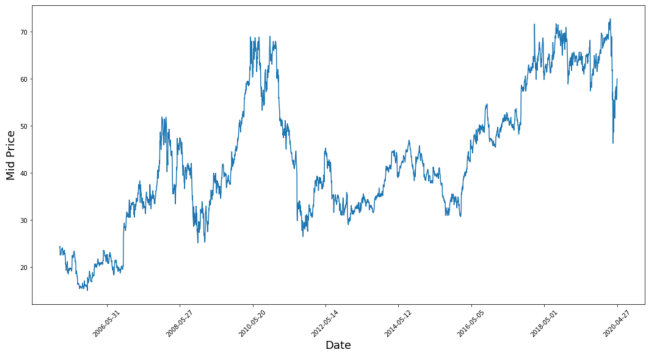

Fig. 3. Visualization of the Dataset

d) Training of Model using Tensorflow: First, TensorFlow has been installed, and all the required libraries have been imported. Now the TensorFlow session starts. Here, X, input data is a 2D matrix, and Y, output data is a 1D vector. At this point, Neural Network is not sure about the number of observations it will take. Therefore, it will consider 'None' value for unsure variables. Initialization of the variables is required before the training of the model starts. Tensorflow uses variance scaling initializer zeros initializer methods to initialize the weight and bias variables, respectively. In the feedforward network, Neural Network has one input, four hidden, and one output layer. The input layer consists of five neurons, first hidden layer consists of 16 neurons, second hidden layer consists of 8 neurons, third hidden layer consists of 4 neurons, fourth hidden layer consists of 2 neurons, and finally output layer consists of 1 neuron. In each subsequent hidden layer number of neurons, get half of the previous layer to compress the information. Then Matrix Multiplication is performed between different layers of the neural network model by using the matmul method. Hidden layers are then transformed by using the relu activation function, which also introduces non-linearity to the model. Transpose of the output layer will be taken. MSE (mean squared error) function has been used as a cost function to measure the deviation between the predicted and the actual value. AdamOptimizer function, which is the combination of two other popular optimizers AdaGrad and RMSProp, has been used as an optimizer function to choose weight and bias variables during training to minimize the cost function. Finally, model training starts by running the neural network session.

e) Architecture: Fig4 shows the architecture of the feedforward neural network adopted to train the model. In this, the flow of data is unidirectional that is from left to right. The three main components of the architecture are the input layer, hidden layer, and output layer.

f) Interactive Plot: Fig5 shows the interactive plot

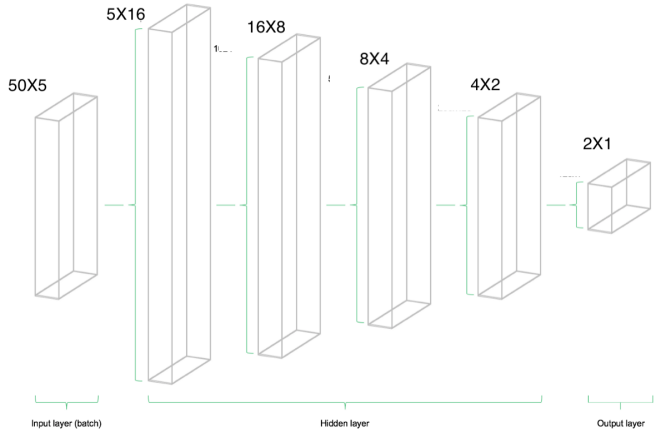

Fig. 4. Feedforward Neural Network Architecture

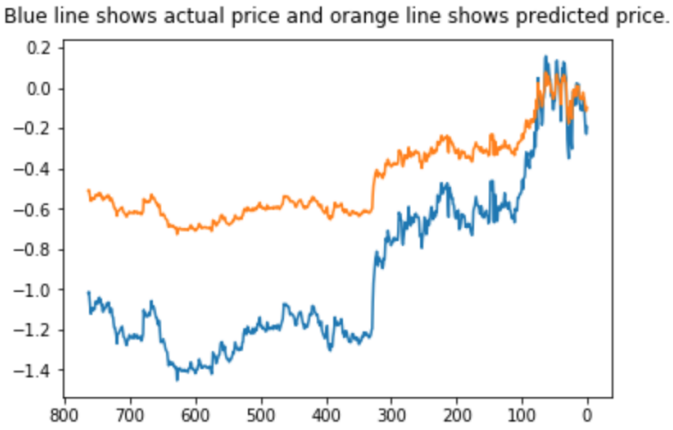

Fig. 5. Interactive Plot

g) Results: To fit the neural network, 10 number of epochs and 50 batch size has been considered. With this measure, the neural network is giving final mse of 0.018

B. Time Series Analysis using ARIMA

Time series in simple terms means data collected over time and is dependent on it. Forecasting stock prices is a very difficult and challenging task in the financial market because the trends of stock prices are non-linear and non-stationary time-series data. ARIMA is a class of models that 'explains' a given time series based on its own past values, that is, its own lags and the lagged forecast errors, so that equation can be used to forecast future values. The parameters of the ARIMA model are defined as

- p: the number of lag observations included in the model, also called the lag order.
- d: The number of times that the raw observations are differenced, also called the degree of differencing
- q: The size of the moving average window, also called the order of moving average.

a) Data Visualization: Time series data is composed of a combination of Trend, Cycles, Seasonality and Residuals. Statsmodels library has a function which decomposes time series into Trend, Seasonality and Residuals. Fig 6 shows the seasonal decomposition of Microsoft stocks from 2010 to 2020. There is an upward trend to the data but seasonal effect in insignificant. The Residuals component is left after other components have been calculated and removed from time series data. It is randomly, identically and independently distributed.

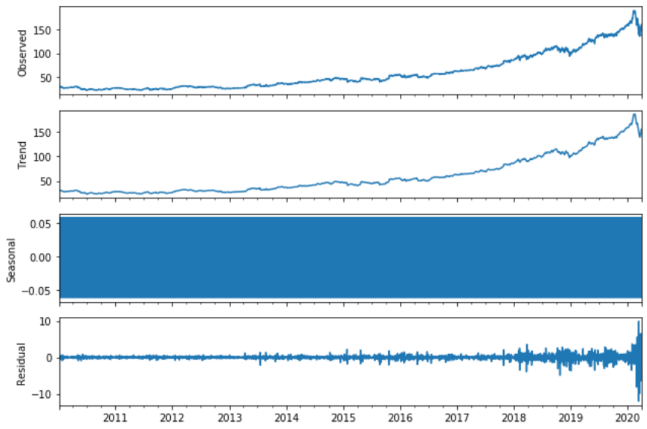

Fig. 6. Seasonal Decomposition of Microsoft Closing Price

b) Make Data Stationary: When we have trend and/or seasonality in a time series data we call it non-stationary. Stationarity means the statistical properties of data, such as mean, variance and standard deviation remain constant over time.

There are two ways we can check the stationarity of a time series. By visualizing the data, it should be easy to identify a changing mean or variation in the data. For a more accurate assessment there is the Augmented Dickey-Fuller test, which provides test statistic and P value. P value greater than or equal to 0.05 means the data is not stationary, otherwise, we reject the null hypothesis and say data is stationary. Rolling mean and standard deviation is plotted for the data to check stationarity. From fig 7, we can see that the data is not stationary.

To make the data stationary, we take a first difference of the data and the number of times the differencing was performed to make the series stationary is the value of the 'd' parameter for the model. Fig 8 shows the stationary data after differencing was performed.

c) Plot ACF and PACF and Find the Optimal Parameters: The next step is to determine the tuning parameters of the model by looking at the auto-correlation and partial auto-correlation graphs. The big issue as with all models is that we do not want to over-fit our model to the data by using too many terms. Fig3 shows the plot of ACF and PACF. The grey lines represent the confidence level. If we look at Fig 9 carefully, we will see that the upper confidence level line has been crossed at lag1. Therefore, the order of AR would be 1 and p=1. Similarly, in Fig 8 for PACF, the the upper

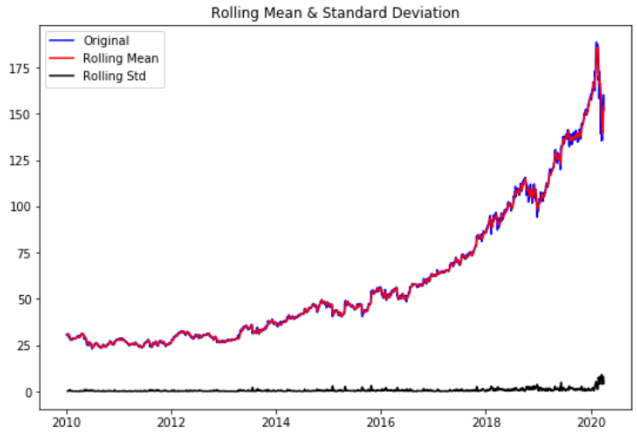

Fig. 7. Non Stationary Time Series

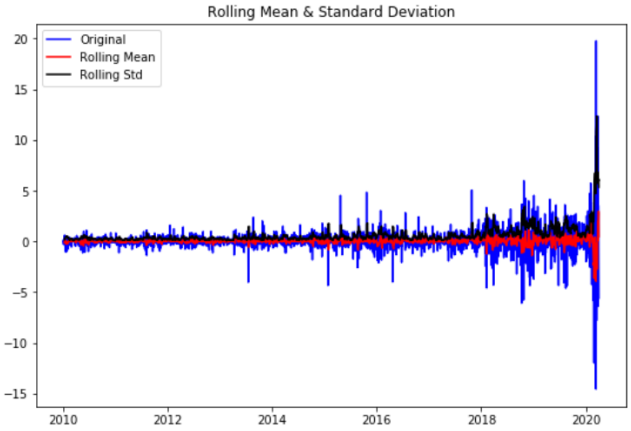

Fig. 8. Stationary Time Series

confidence level line has been crossed at lag1. Therefore, the order of MA would be 1 and q=1.

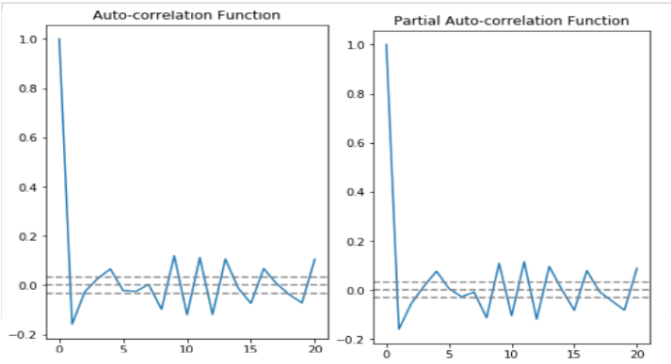

Fig. 9. ACF and PACF

d) Build Model and Predict: After determining the parameters p,d and q, now we are ready to build the model.Split the data-set for training and testing. Each time step of the test set is iterated. A prediction is made each iteration and stored in a list. This is so that at the end of the test set, all predictions can be compared to the list of expected values and an error score calculated. In this case, a mean squared error score is calculated and returned. Fig 10 shows the predicted price vs the actual price.

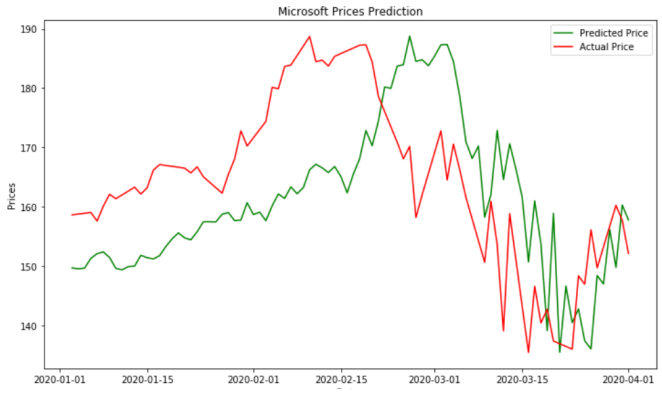

Fig. 10. Prediction vs Actual Price

e) Model Tuning: A suite of parameters p,d and q are evaluated to find the best fit for the model. A grid of p,d and q parameters are specified to iterate. A model is created for each parameter and its performance evaluated by keeping track of the lowest error score observed and the configuration that caused it. Grid search for parameters were performed for Microsoft stocks where the parameters p,d and q were specified values in the range (0,2) and MSE calculated. After performing grid search, the best best parameters for our model was (0,1,1) with MSE 3.694.

C. SVR: Support Vector Regression

For this method we used Adobe's stock data and Fig11 shows Adobe's data fetched from Alpha Vantage API. In

	timestamp	open	high	low	close	volume
1	2020-04-27	348.8000	349.9600	345.1200	348.5000	2361714
2	2020-04-24	338.0000	344.7000	333.5011	344.1000	2146162
3	2020-04-23	338.9000	341.7500	334.8000	335.3700	2120251
4	2020-04-22	337.4100	339.3800	332.8006	336.7700	2231819
5	2020-04-21	340.9000	342.7630	325.2100	328.9900	3548202

Fig. 11. Raw Dataset of Adobe

this project we analyse stock data of a particular company using Support Vector Regression with sci-kit-learn. For this all the required dependencies are imported. After data pre-processing (similar as mentioned in Neural Networks section), curve between timestamp and close value is plotted and it is shown in the Fig12. We employed three models on default SVR (Support Vector Regression) kernel,

- Linear Regression. Linear Regression is a way to find the best linear relationship between two given variables. The goal of linear regression is to find the line of best fit for our data that will result in the predicted values to be as close to our actual values.

$$Y' = A + B * Xeq \quad (1)$$

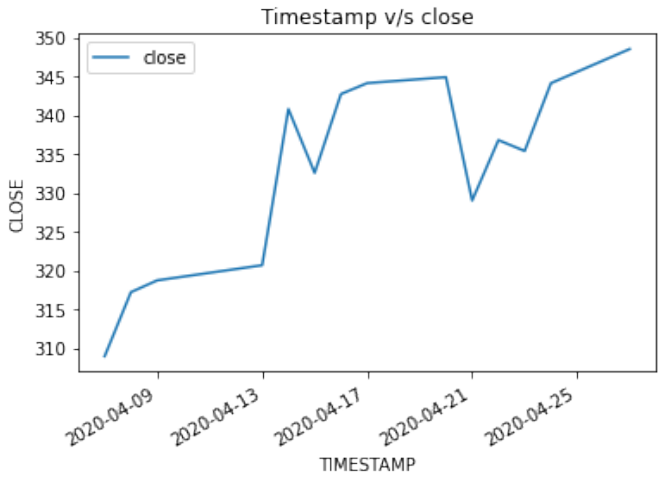

Fig. 12. Timestamp v/s Close Value curve

where, X is Predictor (Present in data); B is coefficient (Estimated by regression); A is intercept (estimated by regression); Y' is predicted value.

- Radial Basis Function. RBF stands for radial basis function. RBF transferred our 2D space into a higher dimension to help better fit our data. The function takes the squared euclidean distance between 2 samples and divides by some sigma value.
- Polynomial Regression. Polynomial Regression is a form of linear regression in which the relationship between the independent variable x and dependent variable y is modeled as an nth degree polynomial.

After applying SVR models on the dataframe of current 14 days, we find out the most accurate regression values. The Fig13 shows the Support Vector regression curve for the three different models. The preliminary results show accurate

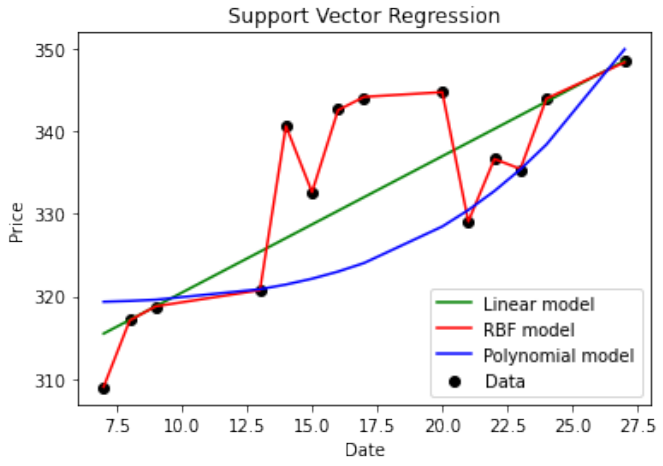

Fig. 13. Support Vector Regression

prediction of the stock price. Here, we observed that Linear regression model and Radical Basic function model shows the best results. The Fig14 represents a snapshot of the output of the regression models.

```
Actual Value:  348.5
Predicted value from Radical Basic Model:  348.4001955092228
Predicted value from Linear Regression Model:  348.60000000012303
Predicted value from Polynomial Model:  349.9899665829195
Root Mean Square Error value:  0.1000000001230319
```

Fig. 14. Predicted values from SVR Models

D. Naive Bayes

a) Data Preprocessing: The initial data was processed to generate a daily change in the market, given by close-open. This daily change value was used for all future calculations. The market data for analysis with Naive Bayes was modified to associate each day's change in value with the previous n days change. This sliding window provides a consistent scope for each days' prediction, and the variable size allows for the model's scope to be adjusted.

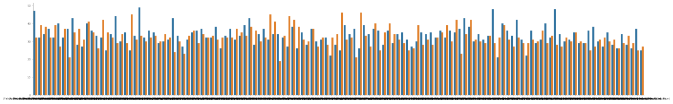

Fig. 15. 7 Day Combinations

b) Data Visualization: Data visualization (Fig 15) was also performed on this windowed data set to visualize how the windows could affect the predictions. This visualization was done with a window size of 7 and shows all 256 combinations of those 7 binary attributes. Each bar represents the number of occurrences of that combination that resulted in an increase (blue) or decrease (orange). By observing these bars in pairs, the influence of the past 7 days on the current day can be estimated. For example a tall blue bar and a short orange bar indicates that that particular sequence of days is generally followed by an uptick in stock value.

c) Training: The training of the model was done by splitting the windowed dataset into testing and training data based on a 50/50 split. The training data was then input in sklearn's naive bayes method, using both the Gaussian and Bernoulli options. For the Bernoulli option, the data is binarized into either a gain or loss for the day rather than the actual numeric change.

d) Algorithm: The Naive-Bayes algorithm is based off of Bayes Theorem:

$$P(C \mid X) = \frac{P(X \mid C)\,P(C)}{P(X)}$$

, where the C is the class assignments and X is the data observations. The denominator remains constant across all classes, so the numerator is the decider between which class is predicted. Using the naive assumption that all the attributes of X are conditionally independent, the attributes can be com-

puted individually and multiplied together, so the numerator becomes:

$$= p(C_k) \prod_{i=1}^{n} p(x_i \mid C_k). \tag{2}$$

Training the model is done by determining the probabilities associated with each attribute and class. Once these relationships and values are determined, the class of new data is determined by simply picking the class with the highest probability given the associated attributes.

REFERENCES

[1] M. Moukalled, W. El-Hajj, M. Jaber, "Automated Stock Price Prediction Using Machine Learning", Computer Science Department, American University of Beirut. Available [Online]: https://www.aclweb.org/anthology/W19-6403.pdf.

[2] K. Hiba Sadia, A. Sharma, A. Paul, SarmisthaPadhi, S. Sanyal., "Stock Market Prediction Using Machine Learning Algorithms," *International Journal of Engineering and Advanced Technology (IJEAT)*, Vol. 8 Sno. 4, pp. 25-31, April 2019. Available [Online]: https://www.ijeat.org/wp-content/uploads/papers/v8i4/D6321048419.pdf

[3] M. A. Hossain, R. Karim, R. Thulasiram, N. D. B. Bruce, Y. Wang, "Hybrid Deep Learning Model for Stock Price Prediction," *IEEE Symposium Series on Computational Intelligence (SSCI)*, Bangalore, India, pp. 1837-1844, January 2018. Available: 10.1109/SSCI.2018.8628641

[4] A. A. Ariyo, A. O. Adewumi, C. K. Ayo, "Stock Price Prediction Using the ARIMA Model," *UKSim-AMSS 16th International Conference on Computer Modelling and Simulation*, Cambridge, pp. 106-112, February 2015. Available [Online]: https://ieeexplore.ieee.org/document/7046047

[5] L. Ryll, S. Seidens. "Evaluating the Performance of Machine Learning Algorithms in Financial Market Forecasting: A Comprehensive Survey," July 2019. Available [Online]: https://arxiv.org/pdf/1906.07786.pdf