Stock Market Prediction based on News Impact Analysis

Aditee Vasant Jadhav, Kavyashree Chandrashekar, Krupa Dhirajlal Vadher, Pranav Sudhir Dixit
Computer Engineering Department
San José State University (SJSU)
San José, CA, USA
Email: {aditeevasant.jadhav, kavyashree.chandrashekar, krupadhirajlal.vadher, pranavsudhir.dixit}@sjsu.edu

***Abstract*—Stock price prediction is an important challenge for strategic investors as it involves great financial gain. The financial market of stock exchange has been seen as more of non-deterministic place of event. Traditional and most followed approach of general public for stock price prediction is monitoring time series of stock values. Machine learning approaches for time series prediction have been widely experimented. One area posing great impact on stock market prediction is information obtained from news or financial reports released regularly. This textual data affects stock price prediction greatly and this area has not been explored at its full potential impact. Accordingly, there is a need for embedding impact of news information while predicting stock price. In this project, we proposed an adaptive approach to predict stock price based on analysis of news impact. Our approach analyses the impact of news over stock price and accordingly makes prediction. The system then analyses time series and news information for potential concept drift detection and rebuilding model by adapting new concepts introduced by news articles. The outcome is realistic and efficient stock price prediction and potential drift detection.**

***Index Terms*—stock market, prediction, concept drift, adaptation, time series, news.**

I. Introduction

Stock market has gained attraction of investors and business firms. An intelligent investment in stock market may give a huge benefit, whereas a errors are costly. The goal of an investor is to maximize their profits by making smart decisions. Leveraging any patterns in the stock market behavior is a great way to do so. Previously stock market was considered as an unpredictable place. But, later they changed their theories to state that stock market is just a random walk of statistics. After more studies, stock market was defined as predictable and can be understood well if underlying patterns are extracted and examined well.

Stock market generates huge amount of data everyday and it is impossible to manually go through it for understanding and examining the patterns of stock market. Thus, use of machine learning techniques for gaining insights into huge stock market data have been proposed [1]. Many researchers have employed machine learning techniques to understand stock market and they understood that predicting stock market movement would be an interesting move towards helping investors determine the best time for selling or buying stocks in order to maximize profits. Stock market prediction not only helps the investors to make appropriate decisions, but it profits other disciplines like: financial sector, computer science, statistics, economics, etc.

Many machine learning models have been proposed to predict stock price movement and previously only numerical time series data has been taken into consideration. However, further in-depth study has shown that the volatility of the stocks is dependent not only on the time series data (numerical data of stock price) but also on textual data (news). Hence, it is essential for a stock market prediction system to consider time series data as well as news information together [2].

Previous projects on stock market prediction were based on the time series data only, which calculates moving average on the numerical stock data. Recently, an advance is made to consider the news information by following a bag-of-words approach. This technique structures the collected words into the form of a vector to analyze their polarity as positive or negative. Based on the polarity, the system predicts the direction of the future stock price movement as rise or fall [3]. However, it fails to predict the intensity up to which stock price will rise or fall. Moreover, they fail to analyze the relation between various news information and their impact on the stocks of the affected companies. In case the novel or unseen concepts appear in news data, the system may fail in predicting the stock value as there are no existing data points in the system representing the new concept. This introduces problem of concept drift into the system. Stock price may decrease or increase dramatically during the concept drift phase. Hence, there is a severe need of incorporating a functionality in the which detects the concept drift and adapts new concepts for efficient stock prediction. These all problems need to be attended and resolved into a single system. However, we could not find any existing solution for them all together.

To address these problems, we propose an adaptive prediction model which is trained using processed news data and stock price data. We get processed news data by extracting features using NLP techniques and then applying initial weights on it. The model performance will be continuously monitored using error rate to detect potential drift. Also, temporal data from news articles will be monitored to detect changes in feature vector over time and to detect drift in news articles. When drift is detected, the system will build new base classifier on drift data and update the ensemble. We will also revise weights of news features while retraining our model on drift data.

II. Problem Statement / Project Architecture

A. Problem Statement

Stock market adheres to the volatile nature, where factors like company's financial reports, investors' interest for the company, and political events affect the price of the stocks. Investors play a high-risk role of buying and selling the stocks on a daily basis. Volatility of the stocks is dependent on the time series data (numerical data of stock price) and textual data (news). Hence, it is essential for a stock market prediction system to consider time series data and news information. Stock market prediction not only helps the investors to make appropriate decisions, but it profits other disciplines like: financial sector, computer science, statistics, economics, etc.

Existing projects on stock market prediction is based on the time series data only, which calculates moving average on the numerical stock data. Recently, an advance is made to consider the news information by following a bag-of-words approach. This technique structures the collected words into the form of a vector to analyze their polarity as positive or negative. Based on the polarity, the system predicts the direction of the future stock price movement as rise or fall. However, it fails to predict the intensity up to which stock price will rise or fall. Moreover, they fail to analyze the relation between various news information and their impact on the stocks of the affected companies. In case the novel or unseen concepts appear in news data, the system may fail in predicting the stock value as there are no existing data points in the system representing the new concept. This introduces problem of concept drift into the system. Stock price may decrease or increase dramatically during the concept drift phase. Hence, there is a severe need of incorporating a functionality in the which detects the concept drift and adapts new concepts for efficient stock prediction. These all problems need to be attended and resolved into a single system. However, we could not find any existing solution for them all together.

To address these problems, we propose an adaptive ensemble model which is trained using processed news data and stock price data. We get processed news data by extracting features using NLP techniques and then applying initial weights on it. The ensemble performance is continuously monitored using error rate to detect potential drift. Also, temporal data from news articles is monitored to detect changes in feature vector over time and to detect drift in news articles. When drift is detected, the system builds new base classifier on drift data and updates ensemble. We also revise weights of news features while retraining our model on drift data.

B. System Architecture

The Fig. 1 gives the overall system architecture of our proposed model. The proposed system analyzes and identifies the impact of news articles on stock prices and then predicts stock prices using time series data of stock market and textual data from news articles. The system first gathers news articles and stock prices data for a given time period. The news articles are pre-processed for feature extraction and initially equal weights are assigned to feature vector obtained. Then an ensemble of base classifiers is trained by using stock price data and processed news data. The processed news data is aggregated into different sliding windows using timestamp of news articles and difference between two consecutive windows is monitored to detect concept drifts that can be seen through feature vectors. This process is termed as temporal data analysis in the above architecture diagram. Also, the performance of the ensemble classifier is monitored using error rate as metric to detect possible drifts in time series data of stock prices. If the drift is detected using either of mentioned techniques, then the data on which drift was detected is used for building a new base classifier and the new classifier replaces the worst performing base classifier from the ensemble. Simultaneously, weights of news data are updated using drift data and base classifiers are retrained. This way the system can maintain old concepts in base classifiers and new concepts in a newly added base classifier. The final output of the system is enhanced stock price prediction.

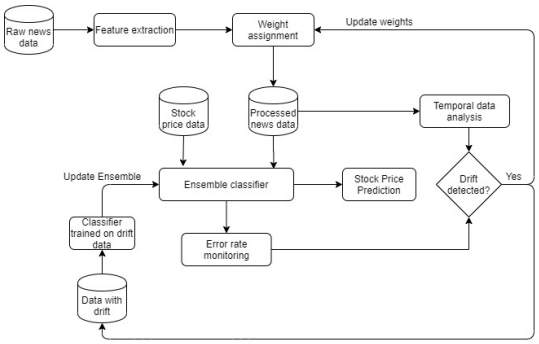


Fig. 1: System Architecture

C. System Design

The project architecture diagram shown in Chapter 5 gives the proposed system overview. Our proposed system makes the use of news data and stock price data to predict stock prices. The system preprocesses news data to extract relevant features and assigns initial weights to the base classifiers. This processed news data is incorporated along with historical stock price data to train ensemble classifier model and then it predicts stock price. The error rate of the proposed system is monitored and temporal news data is also analyzed to detect possible concept drift. If the drift is detected, the system then trains new base classifier on drift data and reassigns weights to the previous data and models. The worst performing base classifier will be replaced by newly trained model. The following subsections describes about the components and details of architecture using UML diagrams like use-case diagram, flowchart diagram, class diagram and sequence diagram.

1) Use case diagram: The use-case diagram as shown in Fig. 2 gives the list of actors and list of functions performed using those actors for our proposed system. The actor 'news and stock dataset' will first preprocess the dataset and provide the preprocessed dataset to the actor 'ensemble'. The actor 'ensemble' takes the preprocessed dataset and trains the models on it. Afterwards, it starts predicting stock prices for unseen dataset which is also called as test dataset. The actor

'analyzer' performs error rate monitoring and temporal data analysis for drift detection. Once drift is detected, the actors: 'analyzer' and 'drift data' will work on updating ensemble system by replacing worst performing base classifier with new base classifier trained on drift data.

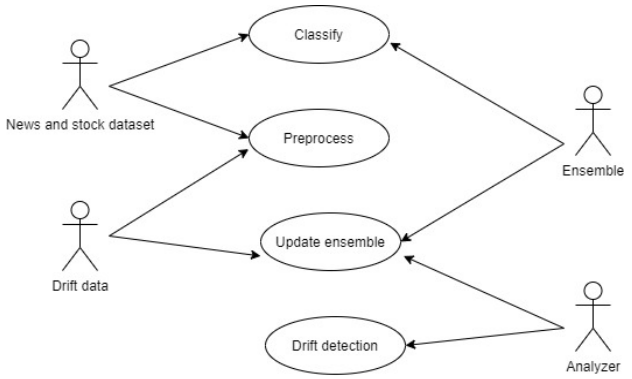


Fig. 2: System Use Case Diagram

2) Flowchart diagram: The flowchart diagram shown in Fig. 3 depicts the working flow on the proposed system as described previously.

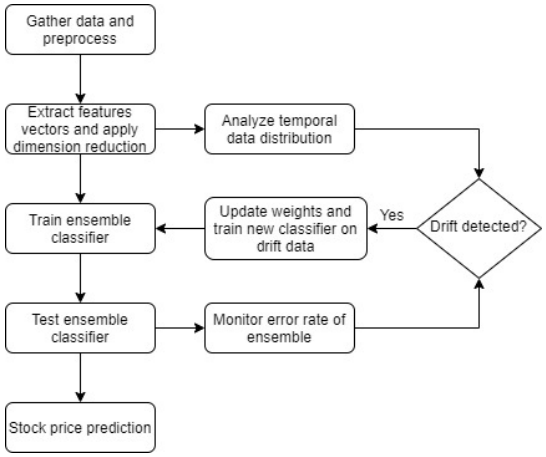


Fig. 3: System Flowchart Diagram

3) Class diagram: The class diagram as shown in Fig. 4 describes the important components of the proposed system in terms of classes and the functions of each component. The four main components of the proposed system are: data preprocessing, classification, analysis and drift handling. The component 'data preprocessing' uses news dataset and stock price dataset to performing functionalities like: feature extraction, data cleaning and dimension reduction. The component 'classification' takes 'processed dataset' and 'base classifiers' to perform functionalities like: training ensemble model and predicting stock prices. The next component is 'analysis' which uses 'processed news data', 'predicted stock prices' and 'base classifiers' to perform functionalities like: monitoring error rate of base classifiers, analyzing temporal data, detecting drift ad reporting drift to the system. The component 'drift handling' takes 'drift data' from component 'analysis' to perform functionalities like: updating weights, training new base classifier on drift data, updating ensemble model by replacing worst performing base classifier with newly trained base classifier and for analyzing drift and drift data to derive patterns of interest.

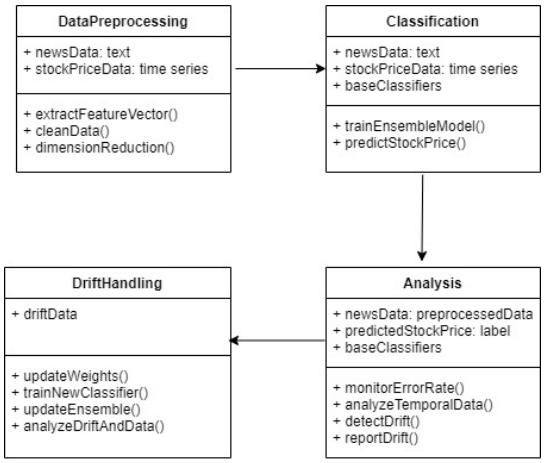


Fig. 4: System Class Diagram

4) Sequence diagram: The sequence diagram as shown in Fig. 5 describes the project architecture and functionalities in terms of sequence of actions performed between different modules. The sequence diagram is just nothing but a systematic representation of flow of the proposed system in terms of steps to be performed one after another. The figure above represents all the steps described by project architecture and other UML diagrams like class diagram, flowchart diagram and use-case diagram.

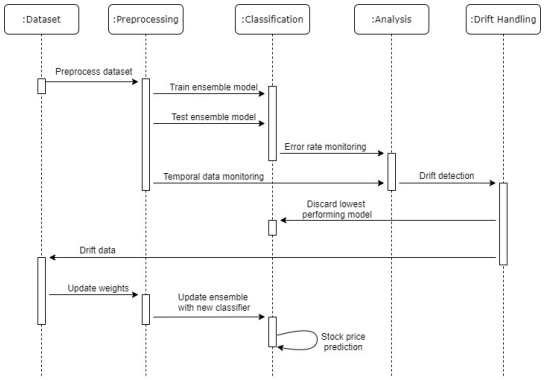


Fig. 5: System Sequence Diagram

III. Method(s) / System Design

Our stock prediction model as a whole can be divided into 3 different modules such as: stock prediction using time series stock data, stock prediction using news impact analysis, and concept drift detection & adaption. These modules can be described as:

A. Stock prediction using time series stock data

Our experiments are focused on predicting the stock prices by using various factors. One of the factors used to predict the future stocks' Closing price is the historical time

series stock data. We have gathered the daily stock quotes form Yahoo finance website for the four companies: Apple, AMD, Disney and Tesla. The stock quotes received included data like 'Date', 'High', 'Low', 'Close', 'Adj Close', and 'Volume'. We have filtered and converted the 'Close' column of a dataframe to a numpy array. We have applied numerous data-preprocessing and preparation techniques like train-test split ratio, windowing, and dataset corpus size on the entire corpus. After tweaking with the above mentioned data preparation techniques, we ran various machine learning models on our data to observe their performance. Models like 'Linear Regression', 'KNN', 'LSTM', 'ARIMA', 'RNN',and 'SVM' are trained on the dataset and their predictions are noted for different experimental data applied on the above mentioned three data preparation techniques. Furthermore, we have applied cross validation and hyper-parameter tuning on the models to get the better stocks' Closing price predictions of the test data.

B. Stock prediction using news impact analysis

After predicting the stock prices with time series stock data, we used news data as our other stock price prediction methodology. News data carry different sentiments and thereby affect the stock prices. Based on the positive/negative impact of the news data, there could be a rise/fall on the future stock prices. We have therefore collected the news headline and article data for the four companies from Thomson Reuters website by developing a data scraper in node.js. We performed three experiments on the news data to observe their impact on the stock prices movement. These are the Natural Language Processing techniques applied on the textual data to get various weights/importance of the words present in the text and thereby using the most weighted words for prediction models.

First experiment is the use of *sentiment analysis* on the news data. We calculated usability scores for six semantic features namely: Subjectivity, Polarity, Flesch index, entropy, Dale Chall index, and Lex diversity. All of these semantic scores are spread across various numbering limits. We normalized them using min-max normalization technique in order to bring them all within [0,1] scale. Afterwards, aggregation of these normalized scores is taken by averaging their score values based on the impact (positive/negative) of the semantic feature. Addition is done for the semantic features with positive impact whereas subtraction is done for the semantic features with negative impact. The averaging of all scores gave the final score which is noted into the dataframes' *finalscores* column. We plotted the correlation heat map for these scores to better understand their relation to the stock price movement.

Second experiment is using *Word2Vec* model to find the semantic meaning of each word in the news articles and generate a vector for each word. Vectors are generated for every word depending on the context they hold in the text. We used a pre-trained Word2Vec model provided by Google. This is the world's largest available Word2Vec model trained by Google on the Google news dataset. We used this model to generate a 300 dimensional word vectors for each word in the news article. Since this yielded a huge vector, we averaged the 300 dimensional vectors for each word in the news article to generate the single 300 dimensional word vector per news article.

Third and the final experiment is using *Sparse Auto Encoders* - the data compression technique. We used SAE to reduce the 300 dimensional vectors to 50 dimensional vectors. This is done by applying encoder function on the 300 dimensional data to generate the encoded version with lossy compression. 300 dimensional word embeddings from the dataframe are converted into a numpy array before feeding into the autoencoder function of the Keras library. This experiment is done to ensure that the unwanted words are removed from the data and proper encoded versions are used. Finally, various machine learning models are trained on the data generated from these three experiments and results are explained in the 'Results' section of this paper.

C. Concept drift detection and adaption

A prediction model is usually trained on certain data and then it starts predicting for unforeseen data. Sometimes, prediction model fails if the testing data possesses characteristics that were not seen by model even in training data. This may happen due to changes in probability distribution of dataset. This problem is termed as concept drift. A concept drift is said to have occurred if the probability distribution of data is replaced by data with different probability distribution. Sometimes, data from different probability distribution slowly starts mixing into existing probability distribution and then data from different probability distribution increases as compared to existing one. These two conditions are termed as sudden drift and gradual drift where sudden drift occurs suddenly while gradual drift is observed over longer period of time and is difficult to detect. The drift detection can be done either by monitoring error rate of the model or by performing density distribution analysis of the data. In this project, we have implemented drift detection using error rate monitoring method. We have used Mean Absolute Error (MAE) as a measure of error rate for given model. We observed error rate of the models over time and our system tells that a drift is detected when the error rate fall beyond threshold value set in model.

To further improve the performance of our model, we build an adaptive model where the base classifiers are retrained and their weights are updated according to their performance. The retraining phase helps our model in understanding the current trends in a better way and thus our model becomes adaptive model which has less chances of drift detection and a better performance is achieved.

IV. Evaluation Methodology / Materials

A. Environment setup

1) Data Collection:

- Time-series historical data
  - Data Provider: Yahoo finance
  - Runtime Environment: System Shell
- Historical News Data

- Data Provider: Thomson Reuters
- Requirements: Node.js
- Libraries: Nightmare and Cheerio
- Runtime Environment: System Shell

2) Baseline Setup:

- Concept Drift Baseline
  - Requirements: Python3, Jupyter Notebook, Colab Repository
  - Libraries: Sklearn, Numpy, Intertools, Scipy, Matplotlib, Pandas
- Stock Prediction Baseline
  - Requirements: Python3, Jupyter Notebook, VSCode
  - Libraries: Beautiful Soup, NLTK, Pandas, TensorFlow, Spacy

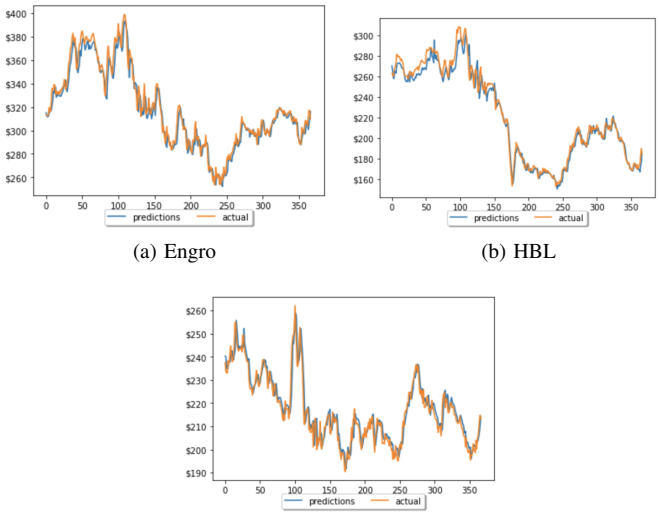
3) Proposed System setup:

- Requirements: Python3, Jupyter Notebook, Colab Repository, WinPython
- Libraries: Sklearn, Numpy, Scipy, Pandas, NLTK, Textblob, Textstat, Keras, Matplotlib, WedDataReader, Statsmodels, Gensim, Math, Datetime, React JS, Node JS, Express JS

B. Baseline implementation

1) Stock Prediction Baseline: In order to analyze time series data along with the textual data, it becomes vital to consider the features which reflect most of the information about the data. The features which give less information or no information about the data into consideration shall be removed. A Hybrid time-series predictive neural network model (HTPNN) proposed by [4] is used to extract important features from the news headlines and arranges them in the form of vectors. Those vectors contain the list of distributed words. They have used sparse automatic encoders to reduce the dimensionality of vectors, which ultimately removes unnecessary information/features. Sparse automatic encoders are used to learn features from the unlabeled data, which is helpful to extract important features from the data with a new concept. They considered the time series stock data which fell within a specific range of time of arrival of the news. Furthermore, they have used the Long Short-Term Memory (LSTM) layer for learning the stock price fluctuations. For extracting the text features, their HTPNN models use deep convolutional layers (CNN), which extracts the features with reduced dimensions. NLP is used to detect the effect of daily news headlines in terms of negativity, neutrality, positivity and compound values. According the model, current news data has effect on current more rather than the next day. By using current stock price information and next day's news information, next day's closing price can be predicted.

The experiment is conducted on three company's dataset, namely HBL, Engro and MCB. Data is pre-processed to eliminate empty values and to have only required data. NLP techniques were applied on the news headlines data to extract the negativity, positivity and polarity. These values combines with stock data and fed to RNN LSTM model for training. After testing around 88 to 92 percent accuracy is achieved.



(a) Engro (b) HBL

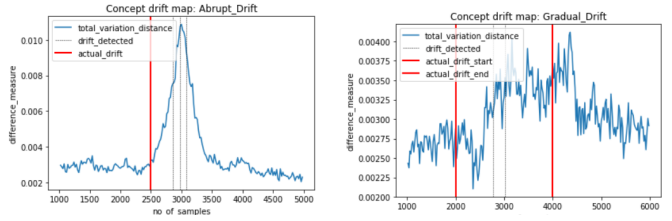
(c) MCB

Fig. 6: Stock Prediction Baseline results

2) Concept Drift Baseline: Stock price prediction fails immensely when the model is faced with the data which is previously unseen by the system. This change in the concept on which the data is collected and analyzed is known as concept drift. This requires the actual data to consist of unseen properties for which there exists no definition of their class labels. The data with a concept drift can be detected by the means of error rate monitoring. However, if the difference is detected due to the change in probability distribution of data, error rate monitoring under-performs in detecting those drifts. [5] proposed an ensemble model that considers the distance metrics between the probability distribution of data. These probability distributions are set at two distinct time windows. This is how a drift is detected when the summation of continuous values of the distance metrics exceeds the threshold value. When the data with concept drift is detected, the system adapts the new concept by learning the new model at a new time window and thereby adding a new classifier to the ensemble. The existing classifiers in the ensemble are updated with the new weights according to their performance in the new window. The least performing classifier is removed from the ensemble. This is how the system efficiently predicts stock prices due to the adaptive nature of the ensemble model for the recurrent concept drift data. Our project aims to consider the data with concept drift and correctly classify the data in our system, for accurate stock price prediction. The objective is to adapt the new concepts in the data and thereby remodel our system with the updated weights for those classifiers.

The code for this baseline approach was available and with error rectifications, we could reproduce their results. For this system, artificial neural network and decision tree algorithms are used. The experiments were performed using artificial dataset as well as real-time dataset (Australian new south wales electricity market dataset). The system was setup with 10,000 data instances for training the model and window size was kept as 500 data instances. The experiments were carried out in three different scenarios: abrupt drift, gradual drift and

recurring context. The drift map obtained for abrupt drift, gradual drift and recurring context on artificial dataset and drift detected on real-time dataset is shown in Fig. 7. This figure describe that all drift scenarios are captured correctly by the system and sometimes, drift is detected multiple times after it has occurred. The authors claim that this is acceptable since concept adaptation favors the relevant base model from the ensemble.

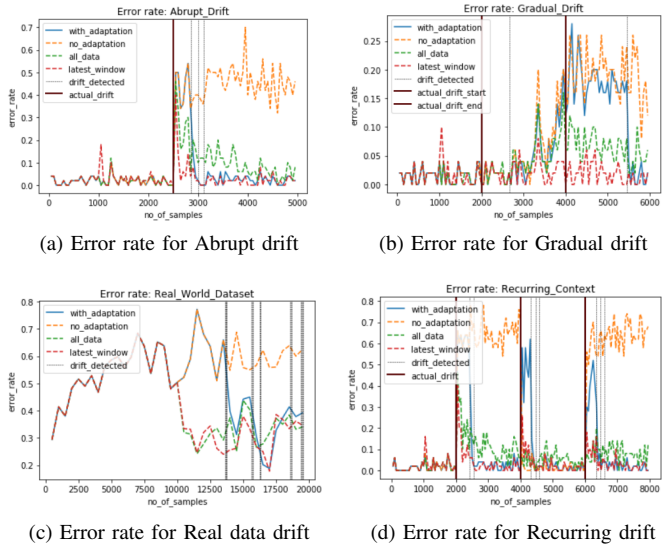


(a) Abrupt drift (b) Gradual drift

(c) Real data drift (d) Recurring drift

Fig. 7: Drift detection Baseline results

The error rate of the system is also captured during experiments to determine which approach works well. The error rates are compared for different algorithms in the error rate graphs as shown in Fig. 8. These graphs show that the proposed system of adaptation works far better than 'no adaptation' approach. It is also observed that algorithm trained on latest window of dataset works better than the algorithm trained on all dataset.



(a) Error rate for Abrupt drift (b) Error rate for Gradual drift

(c) Error rate for Real data drift (d) Error rate for Recurring drift

Fig. 8: Drift detection Baseline results

These experiments also show that the error rate of the system clearly drops down after drift is detected and adapted. Thus, it can be concluded that drift detection and drift adaptation is crucial task in dynamic environments.

C. Proposed System implementation

Proposed system for stock price prediction consists of two main parts. The first one is to combine both news data and stock data and train using multiple machine learning models and the second one is to detect the drift in the prediction and improve the models by fine tuning parameters to obtain minimum error at unforeseen situations.

In this experiment we are considering stock data and news data of four companies such as Apple, AMD, Tesla and Disney. Stock data is collected from the Yahoo finance and is pre-processed to eliminate the missing values and invalid values. News data is collected using Thomas reuters and NLP methods like semantic score calculation and word to vector conversion is applied in order to get the impact of the news data on the stock market prediction. These two data is combined and used for training the different models such as RNN, LSTM, Linear Regression, KNN, ARIMA and SVM models. Among these models will select top three models which performs best in predicting for ensemble model where we calculate the weighted average of the models prediction.

After stock market prediction, continuous analysis on error monitoring is carried out in order to detect the drift in the system. The drift in the system is detected when there is increase in the error value beyond the specified threshold value. This drift detection in the system indicates to fine tune the performance parameters of the ensemble model. Hence the proposed system becomes reliable in unforeseen parameters that affect the stock market.

V. Results

A. Data Preprocessing

1) Stock data preprocessing: The stock market time series data is available on many financial website resources. For this project, we have obtained time series stock price data from Yahoo Finance website. We have used stock data for 4 companies: Apple, AMD, Disney and Tesla. We chose these companies such that some stocks are stable and some stocks are volatile. This gave us variety of data for our experiments. To obtain updated stock price data, we have used web data reader which fetches data from given source for given company and for give date range. After time series stock data is scrapped down into a dataframe, we have filtered data on 'Close' column. The original fetched data contains different columns like: 'Date', 'High', 'Low', 'Close', 'Adj Close' and 'Volume'. Once data is filtered, we converted dataframe into Numpy array format as required by our base models. After that, we have converted data into normalized format using MinMaxScaler with value range between 0 and 1. After scaling data, we split dataset into training and testing dataset. Once dataset are split, we created feature set and class label sets using window technique of size 60. This data is further reshaped into the format as required by each model separately.

2) News data preprocessing: The news articles published by finance website are used for our project at different points for different purposes. Our news dataset was extracted using data collection technique called scraping. The dataset includes news headlines and news body stored in different files. We have collected news articles for four different companies: Apple, AMD, Disney and Tesla. We have implemented semantic scores computation, word2vec model and dimensionality reduction techniques over news articles as a part of NLP techniques of our proposed system. The details of data preprocessing required for each technique is as:

'Data preprocessing for semantic features': We performed many steps of data preprocessing for computing semantic features of news articles data. First we combined news headlines and news body from different files into one dataframe and we removed redundant column from combined dataset. We converted required column into datetime format and then we extracted only date from timestamp. There were multiple entries of news articles for each day, since multiple news may be published on same day. However we have only one entry for stock prices for each day. Thus, we first computed semantic features of all news dataset and then we combined scores and news for same day using groupby method and keeping average value for replacement. On some days, news articles may not be published and there may not be entries of semantic scores for those days. We believe that the impact or sentiment of news stays in environment until next news article is published. Using this principles, we created new entries for missing days and carry forwarded the previous news and semantic features until next entry is observed. The technique used for this purpose is ffill method. After computing all semantic scores, we normalized them using min-max normalization technique.

'Data preprocessing for Word2vec model': The Word2Vec model is used for vectorising textual data. Word2Vec models require minimal textual data preprocessing and can work with raw textual data. Currently, we are only using the news headlines in our models for predictions. For each stock we first generated a separate dataframe with timestamp and the news headline. Many news headlines had descriptive words added by the news source before the headline text like 'UPDATE', 'BRIEF' or 'US STOCKS'. Such words do not add any value to the news. So we removed such words. We used NLTK library to lemmatize each word and also preformed cleaning by removing special characters connected to words. Finally we converted the news headline into a list of separate words. The resultant data frame consisted of two columns - date and list of words, which serves as the input for word2vec model.

'Data preprocessing for dimensionality reduction': The dimensionality Reduction technique known as 'Sparse Autoencoders' (SAE) is a well known type of a deep neural network which is purely based on an unsupervised learning technique. SAE is used to reduce the dimensions of the features which are sparse in nature and thereby avoid unnecessary words from the text/images. For our project, Word2Vec model is used to generate 300 dimensional word embeddings known as vectors. In order to utilize those vectors in a more effective way, we have used SAE to reduce the 300 dimensional vectors to 50 dimensional vectors. There are no specific data-preprocessing steps applied on the 300 dimensional vectors. However, the 300 dimensional vectors are provided in the form of 300 float columns of a pandas dataframe. These columns are converted into a numpy array which is then fed into the autoencoder function of the Keras library for training of the 300 dimensional vectors and thereby encoding them into the lower dimensional vectors.

B. News data processing using NLP techniques

1) Semantic scores: Stock market is considered to be a very sensitive place where the stock market trends may be affected by sentiments of traders and investors. Along with previous history of stock prices, sentiment of people is also key factor that affects future stock price. The sentiment of investors may change due to changes in news articles related to specific companies of interest. Thus, to better understand the effect of news articles and their sentiment on stock price prediction, we decided to perform sentiment analysis of news articles in this project. We extracted the semantic features in terms of subjectivity, polarity, entropy, readability index, etc. from news articles. After computing all semantic scores, we normalized them using min-max normalization technique and this normalization can be described as: Consider x^{ij} be value of j^{th} semantic feature for i^{th} news in dataset, min^j be minimum value of j^{th} semantic feature and max^j be maximum value of j^{th} semantic feature. Then the normalized score value of x^{ij} will be given by:

$$normalizedScore^{ij} = \frac{x^{ij} - min^j}{max^j - min^j} \tag{1}$$

Finally we computed 'final semantic score' which can be defined as an aggregated value representing the overall effect of all extracted semantic features as: Consider $finalScore^i$ be value of final score for i^{th} news in dataset, sub^i, pol^i, dci^i, fli^i, ent^i be subjectivity score, polarity score, Dale Chall Index, Flesch Index and entropy score of i^{th} news. Then the final score value will be given by:

$$finalScore^i = \frac{sub^i + pol^i + dci^i - fli^i - ent^i}{5} \tag{2}$$

To understand the relation between stock price and news sentiment features, we plotted correlation heat map as shown in Fig. 9.

The heatmap shown in Fig. 9 shows that semantic features like subjectivity, polarity, entropy are positively correlated with stock closing price. The semantic features like flesch index and dale chall index are negatively correlated with stock closing price. Thus while computing final semantic score, we added positively correlated semantic features and subtracted negatively correlated semantic features and then we computed average value that represents the actual sentiment of the news articles. The Fig. 9 also shows that final semantic score is positively correlated with stock closing price. This concludes that sentiment of news articles can be useful for predicting stock prices.

From these results, we inferred a few interesting facts. The more is the subjectivity and entropy of the news article, the more informative data it contains and such news articles

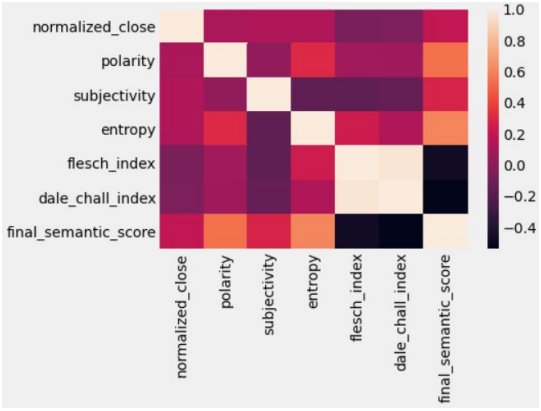


Fig. 9: Correlation heatmap between stock closing price and semantic features

contribute more towards decision making of investors. The more subjective news helps investors getting better insights into events happening around and as these events also contribute towards stock price movement, the investors should have good knowledge of those events. The polarity of news articles is another factor that affects sentiment and decision making of investors. If news article with negative content (negative polarity) is published, investors may get cautious and not perform many transactions. This results into reduction in volume and this reduction in volume contributes to increase in stock prices. Also if news article with positive content (positive polarity) is published, investors may get excited and may perform many transactions. This results into increase in volume and this increase in volume contributes to decrease in stock prices. This infers that polarity of news articles is strongly correlated to stock prices. The semantic features like flesch index and dale chall index are used as readability index which tells about the level of difficulty of the text. These readability indexes show if the text is difficult or easy to read and understand for moderately educated people. If the news article is difficult to understand, investors may become cautious while performing trading and this will result in increase in stock prices due to reduction in volume of stock trading. Inversely, if the news article is easy to understand, investors confidently perform trading and this results in decrease in stock prices due to increase in stock trading volume. This infers that readability indexes (flesch index, dale chall index) are negatively correlated to stock price.

2) Word2vec model: One of the biggest factors affecting stock prices is daily news. A news text can have deep meaning associated with it. The meaning interpreted by people after reading a news headline can indirectly have a very prominent impact on the stock market. Word2Vec models are designed to find the semantic meaning of each word in a text and convert it into its representative vector. These models generate vectors for each word based on the context of that word in the text. Different words appearing in similar context are usually synonyms and so are allocated vectors very similar to each other. Words which never appear together in the same context must be having completely different meanings and so are give different vectors.

Word2Vec models are neural networks which train on the each word in the text. There are two versions of Word2Vec models - Skip gram model and Continuous Bag of Words(CBOW). Both of them consists of one input layer, one hidden layer and one output layer. The input layer and the output layer have the same dimension. The length of the hidden layer defines the dimension of the final word vectors. Each word in the text is converted into a one-hot encoded vector. A sliding window is maintained which runs over the entire text. In skip gram model the center word in the sliding window is the input word while all the other words in the window are the output words. The neural network is trained for the center word, in every window, as the input of model and each of other words, in the window, as the output. The Continuous Bag of Words Model takes the opposite approach. It uses one word in the window as the output and all the other words as the input.

Finally after training the model on all the windows in the text, the hidden layer contains all the words vectors. Each row in the hidden layer matrix corresponds to vector for each word. Different methods, like subsampling and negative sampling, exist to make the training process efficient.

There were two options for using the word2vec model in the project. One approach is to train a new model from scratch on our data and use that to generate the vectors. But this required huge data and resources to train a large model. The second approach is to use a pretrained model. For our project, we decided to use Google's Pretrained Word2Vec model. This model is world's largest word2vec model which Google trained it on the Google News dataset. It has vocabulary of 3 million English words and is of size of 1.5GB. The model generates a vector of 300 dimension for every word. Due to the huge size of the model, normal versions of Jupyter notebook and 32-bit Python interpreters cannot load the model. Therefore, we installed WinPython on a Windows machine. Running Jupyter notebook using WinPython allows us to load large files like the model. We used gensim library's word2vec module to load and use the model.

To convert our news headlines, we first generated word vector for each word in the news headline. So the result of this operation was a 300 dimension vector for each word in the news. Then we averaged the vectors for each word in the news to generate a final single vector for each news. We carried out this procedure for all the news for each of the company.

3) Dimensionality Reduction: The news vectors created by using the Word2vec model consists of 300 dimensions, as explained in the above section. These vectors are the word embeddings applied over the large data corpus. There are few unnecessary words in the news text which can be neglected. Moreover, an effective way of using this word embeddings is to reduce its dimensionality for the feature learning process and application of machine learning and neural network models. We have used Sparse Autoencoders (SAE) to reduce the dimensionality of the news vectors from 300 to 50. Autoencoders is a data compression technique which uses encoder and decoder. It is a type of deep neural network which works on a principle of unsupervised learning. It uses back-propogation to learn the structure of data by encoding the

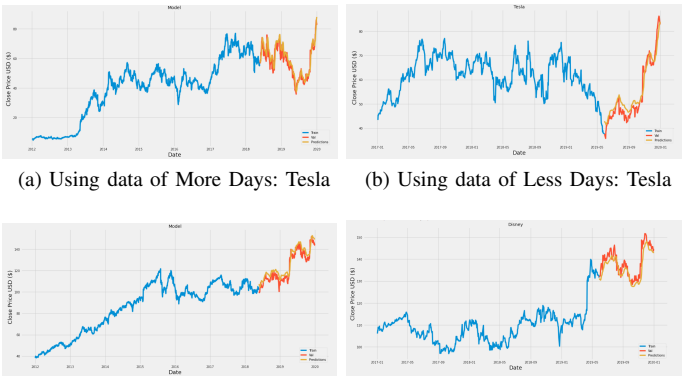
high dimension data into lower dimension. It has three layers: Encoder, Hidden Layer, Decoder. In Sparse Autoencoders, hidden layer is capable of controlling the activation of hidden layer neurons thereby making sure to produce output as close as to the input data.

The SAE is applied to 300 dimensional vectors by setting the encoding dimension size to 50. It will create an encoded vectors of size 50. This data compression technique is lossy by nature which means that the decoder will generate an output which is not exactly same as the input data. The loss function 'binary_crossentropy' is used while model configuration with Adadelta optimizer. After converting the 300 dimensional vector embeddings into the numpy array, auto encoder is applied for learning. This is followed by applying encoder and decoder on the learned numpy array of word embeddings. The encoder will generate a 50 dimensional numpy array of news data which will be used during the application of machine learning and neural network models.

C. Prediction models

1) LSTM model: LSTM is Long Short Term memory model which is a type of Recurring Neural Network(RNN) model. LSTM model is specially known for its best performance in time series prediction. LSTM model keep not only current data in memory for future prediction, but it also remembers the entire time series or sequence it has previously seen. Thus, LSTM tends to remember patterns and this makes it special in time series prediction. For our project, we have used LSTM model to predict stock closing price. We have performed different experiments with LSTM model and they can be described in details as:

'Number of days': In this experiment, we tried using stock data for last 8 years and last 3 years. The graphs with performance of LSTM model when less data is used and performance of LSTM model when more data is used is shown in Fig. 10. We observed that when very old data is also used for stock prediction, the graph looks smooth as compared to the graph when stock data for few recent years is used for training model. However, it is interesting to notice that RMSE(Root Mean Squared Error) value for both scenarios was same. When we investigated more into this, we observed that when we try to plot graph with lots of data, then graph does not reveal all skews that are present and it appears smooth. However, if graph shows all skews when less data is used. From RMSE and graphs, we learnt that LSTM does not keep whole previous history of stocks in memory and thus using huge training dataset does not improve performance of LSTM model.



(a) Using data of More Days: Tesla (b) Using data of Less Days: Tesla

(c) Using data of More Days: Disney (d) Using data of Less Days: Disney

Fig. 10: LSTM using Different size of data: Tesla & Disney

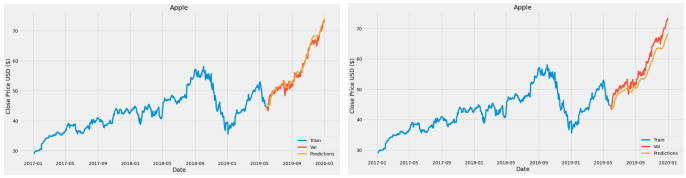
'Training data length': For this experiment we used different ratios for training and testing data split. We used 60:40, 70:30 and 80:20 ratio for training and testing data respectively. We observed that 80:20 ratio gives best performance. The studies have also shown that training model on large chunk of dataset gives better results, but this does not apply every time we simply keep increasing training dataset length. The ratio of 80:20 is optimal and should not be increased further.

'Window length': LSTM model is known for keeping in memory a data sequence from past and this allows model to remember patterns and to better predict future values. We experimented our LSTM model with different lengths of window, where window length is the number of previous days for which model remembers the patterns. We tried different length of window as 1 day, 30 days, 50 days, 60 days. We observed that 60 days length window gives optimal performance of model. We observed that LSTM tends to not remember all patterns if window length is extended beyond 60 and LSTM may not notice all patterns if window length is too small.

'Cross validation': When a model is trained and tested on same dataset, then model may fail to predict on unseen dataset. This problem is called as overfitting. To avoid this issue with our model, we used cross-validation technique. We experimented with different values for test split in cross validation method. While training LSTM model, we tested it on test size of 5% , 10%, 20%. We observed that when test split is less, model fails to capture all skews in details. As we increase test split size, all variations in unseen data are captured more accurately and RMSE decreases significantly. Cross validation technique gives better performing LSTM model than the LSTM model trained without cross validation.

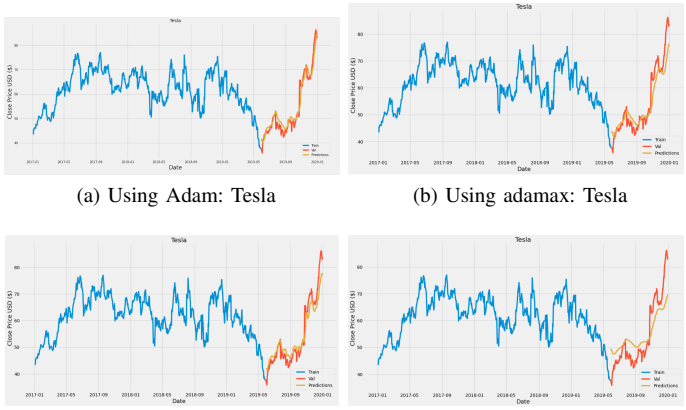
'LSTM parameter tuning': LSTM model is a kind of RNN model with many hidden layers and input-output layers. For this project, we experimented with many number of hidden layers. We observed that usually 1 hidden layer works well with simple problems, while 2 hidden layers show good results in case of complex problems. We used 2 hidden layers and 2 dense layers for our final LSTM model. For activation layer, we experimented with different loss functions such as Mean Absolute Error(MAE) and Mean Square Error(MSE). We observed that MSE gives better performance. The graph showing difference in performance of LSTM when different loss functions are used is shown in Fig. 11. We also experimented with different optimizers such as: Adam, Adamax, RmsProp and SGD. The graph showing difference in performance of LSTM when different optimizers are used is shown in Fig. 12. We observed that Adam works well practically. Thus, in our final LSTM model we have used 2 hidden layers, 2 dense layer, MSE as loss function and Adam as optimizer.

'Semantic scores': We experimented finely tuned LSTM



(a) Using MSE: Apple (b) Using MAE: Apple

Fig. 11: LSTM using Different loss functions: Apple



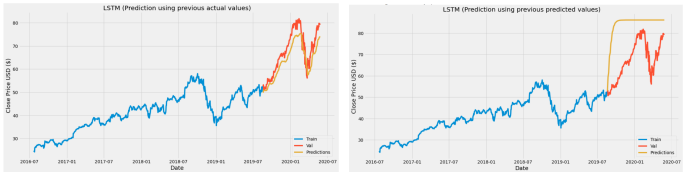
(a) Using Adam: Tesla (b) Using adamax: Tesla

(c) Using Rmsprop: Tesla (d) Using SGD: Tesla

Fig. 12: LSTM using Different Optimizers: Tesla

model first with only stock data and later with stock data and semantic scores of news articles for the same days. We observed that LSTM sometimes show better performance when semantic scores of news articles are incorporated along with stock price data. However, not all experiments with different companies show similar results. The results of LSTM model change when semantic scores are also used for prediction, but it does not always improve the performance of the LSTM model. Thus, for next experiments with LSTM model, we used only stock price data for prediction and we used semantic scores for temporal data analysis.

'Actual Vs Predicted': Our LSTM model is a sequence to one model. It means that our LSTM model considers previous 60 days stock data as X variable and 61st day stock data as y variable. The sequence of previous 60 days stock values is used for predicting 61st day stock value. Our LSTM model predicts only 1 day at a time and it needs actual values to predict the next day. To experiment with prediction for next 7 days or 5 days, we tried using predicted value as X instead of actual values. Thus with predicted value of 61st day, we predicted for 62nd day and with predicted value of 62nd day, we predicted for 63rd day and so on. The graphs showing difference between predictions done using actual values and predictions using predicted value are shown in Fig. 13. From these experiments, we observed that when LSTM model predicts based on previously predicted value, the model only follows direction of movement of stock prices. Thus, graph starts to grow linearly either upwards or downwards and after certain time, it saturates and shows horizontally linear predictions. Thus, RMSE is also increased to a great extent.



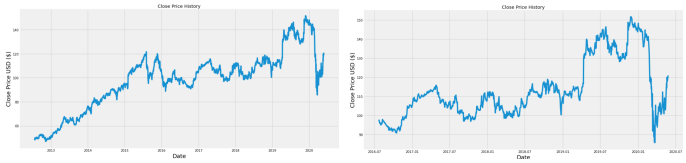
(a) Using actual values: Apple (b) Using predicted values: Apple

Fig. 13: Stock prediction using actual and predicted values: Apple

In contrast, when actual values are used for prediction, the model keeps predictions along with trends and RMSE is very low.

2) Linear Regression model: Linear Regression is one of the prediction techniques which is based on supervised learning. It learns multiple scalar features and predicts the dependent variable. For our project, we have tried various data-preprocessing methods before applying the Linear Regression model. These techniques includes varying the size of dataset, adjusting training and testing ratio, and windowing. Furthermore, we have tested Linear Regression model with multiple hyper-parameters and cross-validation function. The predictions are tested and analyzed by interpreting the weight vectors assigned during the training procedure by Linear Regression model.

'Number of Days': It is a length of a period set with the start and end date to get the stock data. We tried our experiment with two periods: 2012 to 2020 and 2016 to 2020. We decided to use the date range of 2016-2020, since it was precise to notice the stock's 'Close' price movements with that time frame as shown in Fig. 14.



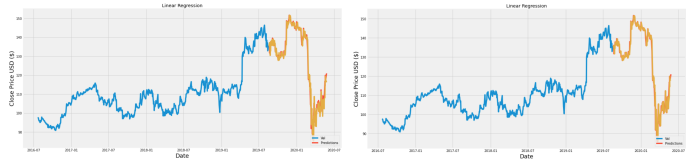
(a) Date Range: 2012-2020 (b) Date Range: 2016-2020

Fig. 14: Number of Days - Disney

'Training Data Length': It is a portion of the dataset used for training and testing our models. Our experiments are done for 80:20 and 70:30 ratio for train:test data respectively. It was found that 80:20 ratio gave precise prediction results for stock 'Close' price.

'Window Size': It is a size of window to capture the training data in the form of an array for the training vector X. We tried with three different window sizes: 60, 50, and 30. Window size with 60 works best where the 'Close' price of previous 60 records are kept in the X training vector and the 'Close' price of 61th record is kept in the Y training vector. Windowing is the most widely used technique lately, in queuing the training data for making predictions. By this manner, we use 60 previous 'Close' prices to predict 61th day's 'Close' price.

'Cross-Validation': We have used cross-validation with K-Folds which is then applied on the Linear Regression model. Number of splits with size 50 and a regression score function (R2) is applied as a cross-validation hyper parameters. Moreover, root-mean-square error is calculated for every experiment and the data with least rms is finalized for the model prediction. As shown in Fig. 15, Regression score - R^2 provided better prediction results compared to the standard Linear Regression model. More details are provided in the later section of 'Weight Vectors'.

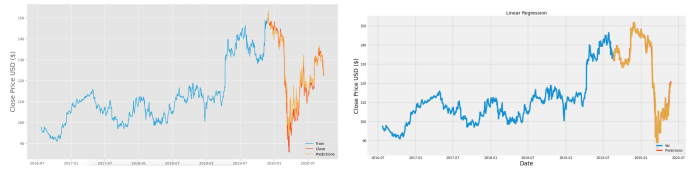


(a) LR - Standard model (b) R^2 - Regression score function

Fig. 15: LR: CrossValidation and parameter Tuning - Disney

'Data Scaling': This is a method used to scale the data values to fall under the range of [0 - 1]. Since we have multiple columns with different data ranges, we need to scale them to be consistent and uniform for their interpretation. We tried two different methods for data scaling. We used sklearns MinMaxScalar for these experiments.

For our experiment-01, we tried to scale the entire feature set during data pre-processing. We started by reading the stock data and scaling them directly using MinMaxScalar transform. After scaling the entire corpus, we split the data into train:test set with 80:20 ratio and proceeded to apply Linear Regression model prediction. This gave 100% accuracy with the precise stock market price prediction. After careful analysis, we noticed that we were performing data snooping on the dataset by scaling the corpus before splitting them into train:test sets. To verify further, we performed experiment-02 where we applied data scaling after splitting the dataset into train:test sets and the accuracy went down to 99% with the data curves of actual and predicted Close price more distinguishable as shown in Fig. 16 Linear Regression is a model which is based on the supervised learning technique where it learns scalar features for the target variable and predicts the dependant variable. By clipping the R^2 Regression Score function with the data-preprocessing techniques like windowing and data scaling done after train-test split, we got the best LR predictions upto 99% accuracy.



(a) LR - No Data Snooping (b) LR - Data Snooping

Fig. 16: LR Data Scaling

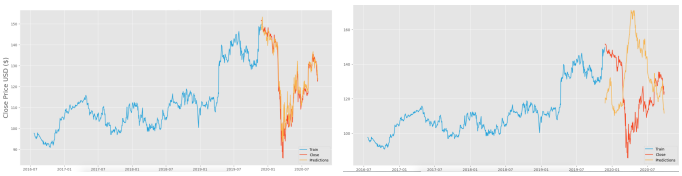
'Weight Vectors': In Linear Regression, the target value is the prediction made by the model by assigning weights to the input data/features. This is calculated as the weighted sum of the feature inputs. The weights assignment varies and the one with most precise predictions on the training dataset or yielding minimal error is considered during the test input predictions. The equation 3 is related to the predictions made by the Linear Regression model.

$$y = \beta_0 + \beta_1 * x1 + \beta_2 * x2 + ... + \beta_p * x_p + \in \quad (3)$$

y is the prediction made by the model which is calculated by taking the weighted sum of its feature vector. Weight vector $\{\beta_0, \beta_1, ..., \beta_p\}$ is the collection of weights which gave the best predictions during training of the feature set. $\{x1, x2, ..., xp\}$ is the vector of the input features. β_0 is known as the intercept which is not multiplied with the first item in the feature vector set. $\in$ is the error added as a form of penalty, which is a difference between the predicted and actual result value.

For our experiments, we have considered the window size of 60, which means there are total 60 items in the weight vectors. Hence, 60 weights are assigned to the 60 items in x_train respectively. Hence according to the equation 3, stock price is predicted based on the weights assigned to the training data. We observed that the highest weight is assigned to the last record of the items in feature vectors.

Furthermore, we used R^2 coefficient of determination as a regression score function. R^2 gave 99% accuracy in predicting future stock prices. For achieving this accuracy, we considered 'Close', 'Adjusted Close', and 'Volume' of the stock data as a feature set. Nevertheless, using just the 'Volume' column for stock price prediction gave 16% accuracy. Hence, we concluded that using the 'Volume' with other stock data yields better prediction rather than using just the 'Volume' data. The comparison is shown in Fig. 17



(a) Close, Volume Accuracy: 99% (b) Volume Accuracy: 16%

Fig. 17: LR: FeatureSet selection - Disney

3) KNN model: K-Nearest Neighbors is one of the simplest prediction techniques which is based on supervised learning. It finds k nearest neighbors of the new data point and assigns it the class to which the majority of the neighbors belongs. For our project, we have tried various data-preprocessing methods before applying the KNN model. Furthermore, we have tested the model with multiple values of nearest neighbors.

For data pre-processing, we applied methods like 'Number of Days', 'Training Data Length' , and 'Window Size' on the stock data. After applying data pre-processing methods on the stock data, we tested our model by tweaking its hyper parameters and applied cross-validation methods like n-neighbors size.

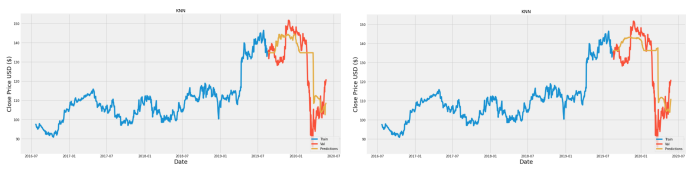
'Number of Days': In this method, we considered the stock data from the past eight and three years. We tried our

experiments with two periods: 2012 to 2020 and 2016 to 2020. Since the fluctuations of stock prices is clearly visible when we plot the stock data using Matplotlib library, we considered the date range from 2016 to 2020. Furthermore, we noticed that the Root Mean Squared Error (RMSE) remained same irrespective of the data size, which also supported our selection.

'Training Data Length': It is a portion of the dataset used for training and testing our models. Our experiments are done with 80:20 and 70:30 percentage ratio for train:test data respectively. It was found that 80:20 ratio gave precise prediction results for stock 'Close' price since training the models on large dataset gives better test predictions and increase the chance of accuracy.

'Window Size': We tried our experiments with three different window sizes: 60, 50, and 30. Window size with 60 works best where the 'Close' price of previous 60 records are kept in the X training vector and the 'Close' price of 61th record is kept in the Y training vector. Windowing is the most widely used technique lately, in queuing the training data for making predictions. By this manner, we used 60 previous 'Close' prices to predict 61th day's 'Close' price.

'Cross-Validation' and 'Hyper-Parameter Tuning' with n-neighbors is applied on the KNN model. Neighbors with the size of 1,5, and 20 are applied as the hyper parameters to the KNN model. Moreover, root-mean-square error is calculated for every experiment and the data with least rms is finalized for the model prediction. As shown in the Fig. 18, KNN with N-Neighbors set to 5 gave better predictions than any other number of neighbors.



(a) N-Neighbors: 5 (b) N-Neighbors: 20

Fig. 18: KNN: CrossValidation and parameter Tuning - Disney

4) ARIMA model: ARIMA is a combination of two different statistical techniques - AR and MA. Auto-Regressive (AR) Model uses a linear combination of past values to predict the next value. Along with past values a random error and constant is also added. This is different from Moving Average(MA) model, MA uses a linear combination of errors or white noise to generate the next error and add it to the mean value to generate the prediction. Both of these models can be combined to create an ARMA model. ARMA model works only on stationary time-series. Visually inspecting a stock price time-series reveals that it is non-stationary. For non-stationary data, ARIMA model is useful. It converts the non-stationary data into stationary before applying ARMA. It does this by finding difference in the data values between specified timespan. It is like a Z-transform for time-series data wherein it converts the original time-series into a difference time-series. Predictions are generated on the transformed time-series and then are converted back into the original time-series dimension. An ARIMA model has three important tuning parameters - p,q and d. p specifies the number of past data values to be used for forecasting. q specifies the number of past error or noise values to included in the model. While d specifies the order of difference applied during non-stationary to stationary conversion of the time-series.

For every experiment we divided the dataset into train and test using 80:20 split. We performed different experiments on the ARIMA model and they are described below-

'Prediction span': In this experiment, we tried two approaches for predicting stocks. In the first approach, we only predicted the next day stock value using the model. At every iteration, a new model was created to predict the stock price of the next day. Using this approach we were able to get good predictions. Fig. 19 shows the result of this experiment. The RMSE values for predictions ranged from 2.328 to 2.382 while the symmetric MAPE values ranged from 27.692 to 27.748.

In the second approach, we created a single model using the train data, and used it to predict all of the test data. The model generated a linear prediction without any noise or random shocks. Due to this, the model failed to forecast the jump, stock prices experienced, during the pandemic. The RMSE values ranged from 29.916 to 33.919 and the symmetric MAPE values from 29.691 to 35.185.

'Order of models': We experimented with different values of p,q and d. For the single day prediction approach, the results revealed that for different values of the hyper-parameter the RMSE showed a maximum difference of 0.06 and the symmetric MAPE of 0.056. Results show that for this approach, changing hyper parameters does not have any considerable effect on the prediction accuracy. The best accuracy was found for p,d,q as (1,1,1) while the worst was for (5,2,1).

With the second approach, we started by experimenting with the value p while keeping d and q to 1,1. Table I shows the result of the experiment for Apple stocks. It can be observed that initially the error rate decreases with the increase in the value of p, but start increasing after a threshold. We found that threshold to be 3. We also found similar trends for q with inflection point value being 4. Table II shows the error value with respect to change in q. For the value of d, experiments revealed that the best results were generated for higher order values. The implementation of ARIMA we used only supported the value up to 2, for which we got least error. Our results revealed that the best combination of results were observed for values (3,2,2) with RMSE and symmetric MAPE being 29.901 and 29.673 respectively. Fig. 20 shows the results for different stocks with different hyper parameters value.

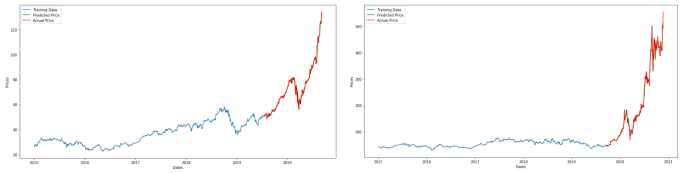
p,d,q	RMSE	Sym. MAPE
1,1,1	33.83	35.015
2,1,1	33.757	34.879
5,1,1	33.765	34.894
10,1,1	33.919	35.185

TABLE I: Change in error values with respect to p

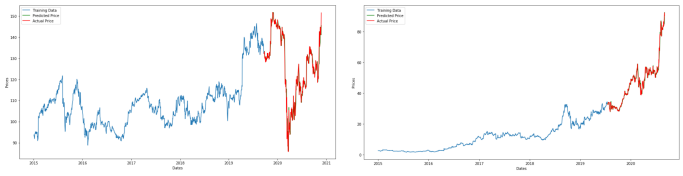
5) RNN model: Recurrent Neural Network(RNN) is a powerful neural network implemented to handle sequentially dependent data. We carried out many experiments with RNN for stock market prediction.

p,d,q	RMSE	Sym. MAPE
1,1,1	33.83	35.015
1,1,2	33.76	34.884
1,1,5	33.76	34.888
1,1,10	33.935	35.221

TABLE II: Change in error values with respect to q



(a) Apple (b) Tesla



(c) Disney (d) AMD

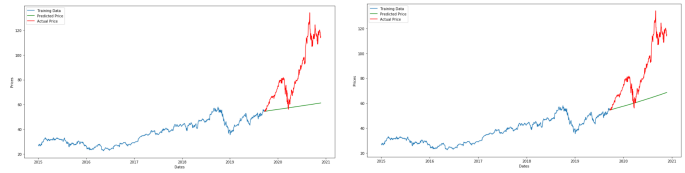
Fig. 19: Single day prediction using ARIMA

'Training data length': The first one is trying different amount of training and testing data split. Starting with 80:20 split for training testing we obtained least RMSE values, where as 70:30 and 85:15 RMSE value was increased by 50 percent to 75 percent as the training data was limited compared to 80:20 split. When the training data is increased to 85 percent, RMSE shows an increase in its value as the data was over fitting. The optimum training split is concluded 80:20 split with this experiment.

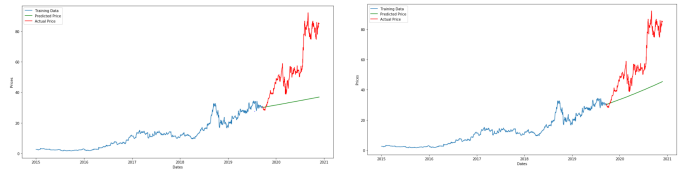
'Window size': Another experiment was to use different window sizes while training the model, starting with 50, then increased to 60 and finally for 70 experiment was conducted. Among these 60 window size showed optimum performance.

'Semantic scores And Vectors': Afterwards we tried to integrate the semantic scores and vectors from news data. Semantic scores with stock data improved the performance of the model, where as vectors showed dramatic decrease in the accuracy. This results were observed due to the number of vectors we used. Beginning we used 300 vectors which varies from -1 t0 +1, due to the sudden drop in accuracy, we decreased it to 50 vectors. This time performance was better compared to 300 vectors experiment as the 300 vectors were dominating and close value was given least preference.

'Batch processing' : All the experiments mentioned above were carried out without batch processing. We repeated all experiments listed above with batch training. Training data set is divided into 50 batches and trained with 10 epoch during training. Batch processing yields better accuracy and reduced RMSE for the model.

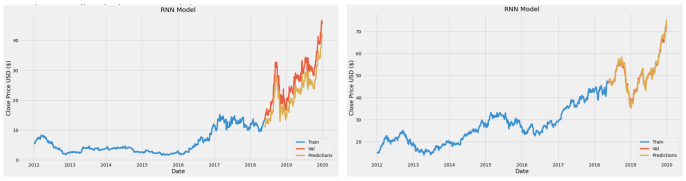


(a) Apple - 1,1,1 (b) Apple - 3,3,2

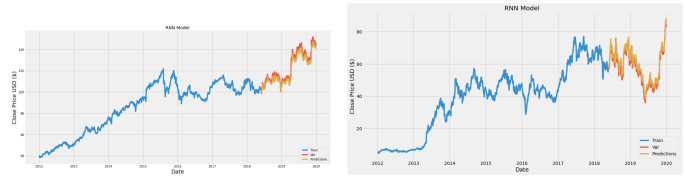


(c) AMD - 2,1,1 (d) AMD - 10,1,10

Fig. 20: Forecast generated by ARIMA with different values for p,d,q



(a) AMD (b) Apple



(c) Disney (d) Tesla

Fig. 21: RNN prediction

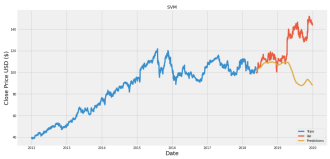
6) SVM model: Support Vector Machine(SVM) is a supervised machine learning algorithm which is basically used for classification problems. Stock data is continuous in nature, from experimenting with SVM we infer that continuous data needs regression models and classification models like SVM won't work well as it is designed for classification. We obtain RMSE of 24.06 for stock prediction with stock data alone and also with stock data with semantic scores and vectors.

We conducted all the experiments mentioned in RNN expect batch processing for the SVM Models but model failed to show improve in the prediction accuracy.

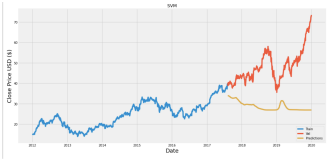
7) Ensemble model: Any prediction model may fail if it sees unforeseen patterns of data while testing. Some models may not perform well at a certain point, while some models may perform well at same time. The performance of system can be improved further if multiple finely tuned models predict together. Ensemble is a set of multiple base models where final outcome is calculated by aggregating the predictions of base models. Majority voting technique is most commonly used in ensemble, however this technique works well for classification problem. Stock market prediction problem is a regression problem and hence majority voting is not a great technique to use. Thus, in this project we have used an ensemble of multiple base models which predict stock value and final stock value is computed by taking the aggregate of all those predicted

(a) AMD

(b) Apple

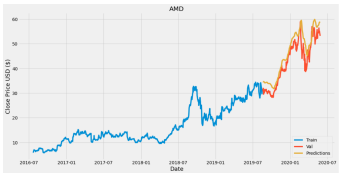


(c) Disney



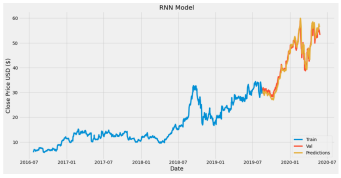
(d) Tesla

Fig. 22: SVM prediction



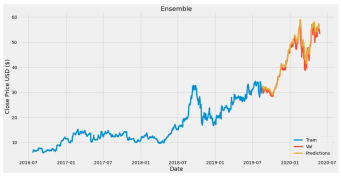
(a) LSTM: AMD

(b) KNN: AMD

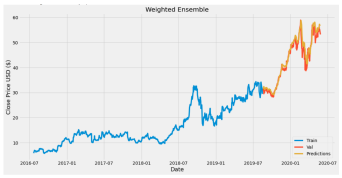


(c) RNN: AMD

(d) SVM: AMD



(e) Ensemble: AMD



(f) Weighted Ensemble: AMD

Fig. 23: Stock prediction using different models

values. We used RNN, LSTM and LR model as base models of our ensemble. The computations for final predicted value of ensemble model can be explained as: Consider $finalClose^i$ be predicted 'Close' price for i^{th} day in dataset. The $p1^i$, $p2^i$, $p3^i$, be predicted values of i^{th} day by LSTM, RNN and LR models respectively. Then the final predicted value of ensemble will be given by 4:

$$finalClose^i = \frac{p1^i + p2^i + p3^i}{3} \tag{4}$$

To extend these experiments, we further implemented weighted ensemble. In weighted ensemble, we assigned weights to models according to their training performance and final outcome is calculated by taking weighted average. Consider $weightedClose^i$ be predicted 'Close' price for i^{th} day in dataset. The $p1^i$, $p2^i$, $p3^i$, be predicted values of i^{th} day by LSTM, RNN and LR models respectively. The $w1$, $w2$, $w3$, be weights of LSTM, RNN and LR models respectively. Then the final predicted value of weighted ensemble will be given by 5:

$$weightedClose^i = \frac{w1 * p1^i + w2 * p2^i + w3 * p3^i}{w1 + w2 + w3} \tag{5}$$

The graphs of performance for each base model, our ensemble model, our weighted ensemble model are shown in Fig. 23.

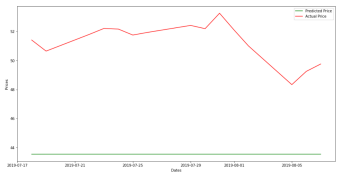
From these experiments, we observed that some base models may not work in every scenario, but ensemble model works always well. RMSE of ensemble is sometimes greater than some base models, however ensemble model always captured all details and trends very well than any model. We observed that weighted ensemble shows similar results as ensemble model, but RMSE is slightly increased when we use weighted ensemble. For weighted ensemble, we experimented with different metrics for assigning weights to the base models. We used RMSE, MAE as weights for base models and we observed that the use of MAE as weight gives better results than RMSE as weight.

8) Sequence to sequence model: A sequence to sequence model is a form of deep neural network which has to ability to take a sequence as input and generate another sequence as output. It consists of two separate networks - an encoder and a decoder. Both encoder and decoder are deep neural networks usually LSTM or GRU models. The state generated by the encoder is provided as input to the decoder. For our experiment, we converted data into sets of 15. Each set acts as the input sequence while the next sequence as the output sequence. We used GRU to implement the encoders and decoders. The number of neurons in the hidden layer was set to 35. Batch size, epoch, and epoch size were chosen as 15,200 and 15 respectively. Adam optimizer was used with loss function as MSE. The data was split train and test data with 80:20 split. We performed the following experiments-

'Cell type' - For the first experiment, we used GRU as the cell type. Fig. 24 shows the result. The model generate linear output with high error. In the second experiment, we used LSTM as the cell type in the model. We first trained the model using the train data. A sliding window approach was used for prediction. After training, the model was used to generate the first 15 day prediction after the train data. Then the actual values of the 15 day window were used to retrain the model. The new model was then used to generate the next 15 day window values. This process was continued to generate the predictions for all the test data. Fig. 25 shows the result of the experiment. The result clearly shows that the model generated decent results for Apple and AMD but failed to generate reliable predictions for Disney and Tesla.

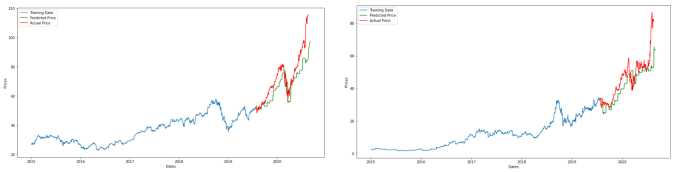
D. Drift detection

The prediction models face the problem of concept drift and performance degrades when the probability distribution of the dataset changes suddenly or gradually. The concept drift can be detected in two ways: by error rate monitoring or

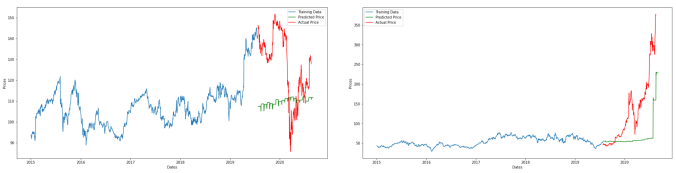


(a) 15 day prediction for Apple

Fig. 24: Encoder Decoder model result using GRU



(a) Apple (b) AMD

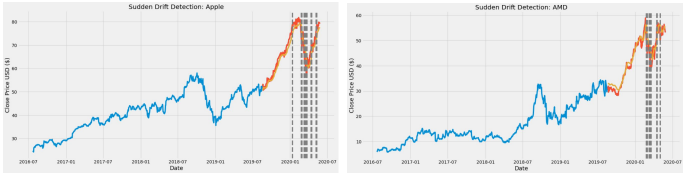


(c) Disney (d) Tesla

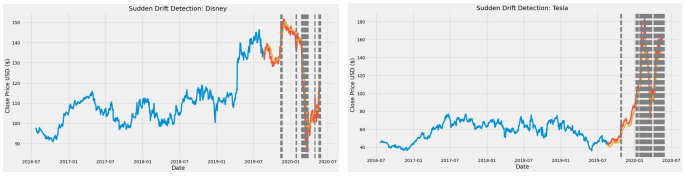
Fig. 25: Encoder Decoder Model result using LSTM

by monitoring density distribution. In this project, we implemented error rate monitoring technique for detecting drifts in dataset. The error rate can be measured in terms of any metrics like RMSE, MAE, MSE. For this project we experimented with all these metrics and we observed that use of MAE as metrics for error rate monitoring gives better results than other metrics. We used MAE as threshold value for models and then we monitored error rate to detect concept drift. From the experiments, we observed that whenever there are sudden spikes in stock price data, the model detects concept drift. This can be termed as sudden drift since sudden spikes are captured by drift detection module of our project. We also experimented our drift detection module using error rate monitoring over base models individually and over ensemble model and over weighted ensemble model. The graph of drift detection for LSTM model is shown in Fig. 26. The graph contains subgraphs for each company and vertical lines on each graph show the dates during which drift is detected.

The graph of drift detection for ensemble model is shown in Fig. 27. We observed that drift detection with ensemble and weighted ensemble gives better results than base models individually. This happens because ensemble and weighted ensemble are better at predicting stock prices and all fluctuations as compared to the predictions of the individual base models. From our experiments we observed that all the models detect concept drift during sudden spikes observed in stock price and this date range (during which drift is detected) can be used for further analysis to make our system adaptive.

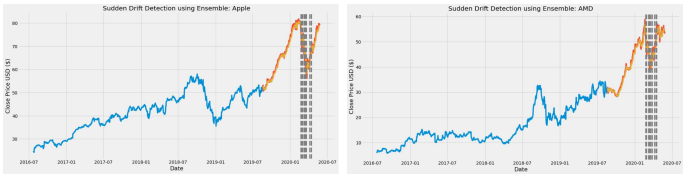


(a) Sudden drift: Apple (b) Sudden drift: AMD

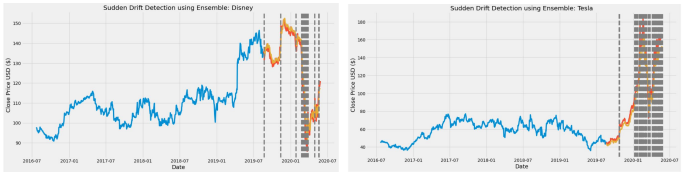


(c) Sudden drift: Disney (d) Sudden drift: Tesla

Fig. 26: Drift detection using LSTM model



(a) Sudden drift: Apple (b) Sudden drift: AMD



(c) Sudden drift: Disney (d) Sudden drift: Tesla

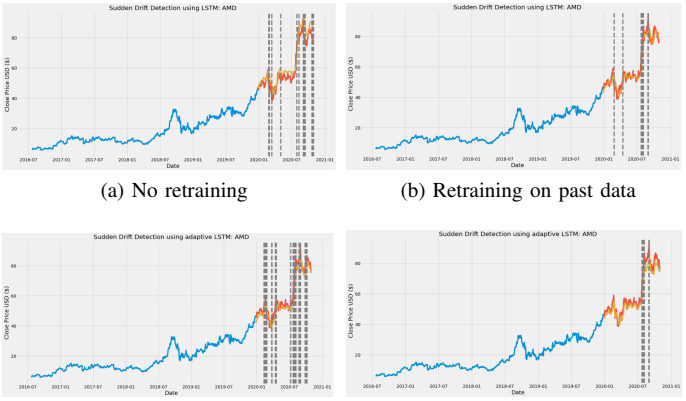
Fig. 27: Drift detection using ensemble model

E. Drift adaption

Once a drift detection module detects drifts in data, the crucial steps are drift analysis and model rebuilding. The concept drift represents the changes in underlying data which are not captured well by the classifier and hence the model performance degrades. To overcome this problem, we developed an adaptive module which retrains base classifiers and updates the weights assigned to those base classifiers depending on their performance. Thus recent trends are reflected into newly trained model and predictions are not affected by drift anymore unless a novel concept drift is observed. If another novel drift is detected, our system adapts it by retraining base classifiers and updating the weights. This is an iterative process and hence our system can be called as an adaptive system.

In this project, we performed different experiments with retraining base classifiers in adaption module. In first experiment, we did not retrain our base classifiers after drift was detected. In second experiment, we retrained our base classifiers on past data. In third experiment, we retrained base classifiers on recent only data. In fourth experiment, we retrained our module on past as well as recent data. The graphs of experiments are shown in Fig. 28.

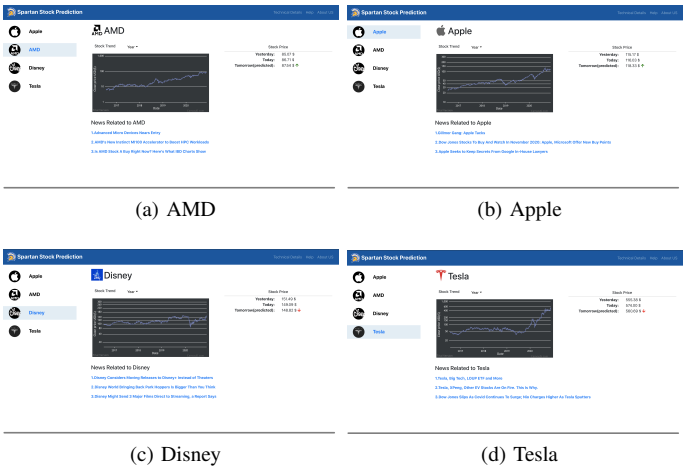
The graphs in Fig. 28 show that when prediction model



(a) No retraining (b) Retraining on past data

(c) Retraining on recent data (d) Retraining on recent and past data

Fig. 28: Drift adaption using ensemble model



(a) AMD (b) Apple

(c) Disney (d) Tesla

Fig. 29: User Interface

is not retrained after drift detection, the recent trends are not understood by model and model keeps capturing drifts and the model performance continues to degrade with each drift detected. When the model is retrained on past data, the drifts detection ratio reduces and model performed is slightly improved. This shows that previous trends are well reflected into model, while recent trends are not reflected. When the model is retrained on recent data during which drift was detected, then model tends to detect more drifts and model performance is degraded. This happens because model is well trained on recent data, while past patterns and trends are not captured by the model. Finally, when the model is retrained on past data as well as recent data during which drift was detected, the drift detection module shows least drifts and model performance is significantly improved. This happens because model is well trained and model captured all trends including recent trends and past trends. This shows us that, to accurately capture drift and improve model performance, the model must be adaptive model which keeps track of past as well as recent trends. Furthermore analysis of experiments shows that LSTM model is well known for keeping patterns from past in its memory and thus, adaptive LSTM gives better results than other adaptive base classifiers.

F. User Interface

To present the stock prediction to the user we developed user interface in terms of web application. User can able to see the stock trend of the companies and they can also get the predictions for the next day. User can see the stock trend for a year, month or a week, news affecting stock price and stock movement for the next day based on the predictions done using our ensemble model as shown in Fig. 29. We are using React JS and Node JS to build this application.

VI. Discussion

It can be observed from the results that time-series models and deep neural networks like LSTM, RNN, Arima performed better than general models like KNN, SVM with linear regression being an exception. Linear regression in spite of being a simple model showed very good results. Inspecting the trained linear regression coefficients revealed that the maximum weightage is given to last day stock price. Integrating news data, along with stock data showed promising results for LSTM and RNN experiments. For LSTM, integrating semantic scores improved the prediction for some stocks while for RNN it improved for all companies. We also observed that in order to use news vectors into our models, the dimension of the vectors must be a maximum of 50 or below. High dimension vectors tend to deteriorate the performance of the models. Due to the variety of models we experimented, ensemble methods provides a great opportunity.

For Arima, we found that model gave slightly better results for single day prediction when the hyper-parameters values were low. A small value for p and q indicates that the predictions are based on less historical values and more recent values. This could be due to the fact that single day predictions are more dependant on the earlier day prediction than on the past trend. For long predictions, it was observed that stocks which had a consistent trend, models for such stocks generate better predictions when value of p and q is higher. This confirms the idea that higher values indicate higher dependence on the past values.

Ensemble models with three base models gives good results in general. The technique of concept drift along with models is able to detect sudden spikes. Drift detection with ensemble methods with one of the base models being RNN and LSTM, which are using news data, show the best results so far. Thus the combination of drift detection along with news data shows great potential for accurate stock prediction.

VII. Related Works

Stock price forecasting is a great challenge for investors, as it drives their decision of buying or selling the stocks. Simply guessing the price involves a great financial risk and thus the stock market needs to be studied. The research in the field of stock price prediction started early with Efficient Market Hypothesis and Random Walk Theory [6], [7]. Both of these models stated that the stock prices are unpredictable. However,

later studies showed that stock prices can be analysed and predicted using machine learning techniques. Many researchers have studied and proposed different models for stock price prediction. There are two approaches available for processing the information used for stock price prediction. The first and traditional approach is to train models using only stock price data as time series data and predict stock values based on the current and past values of stocks. However, stock price is affected by many external factors like news articles, financial reports and political events happening around. The second approach uses news data along with stock price data to predict the future prices of stocks. Natural Language processing (NLP) and sentiment mining is performed on the news articles to extract knowledge required for predicting the stock prices. Furthermore, sometimes there may be changes in political laws, events happening, and this change may introduce a concept drift into data. The drift may degrade the performance of the system and hence, concept drift detection is a crucial task when working in a volatile environment. In this literature survey, subsection 'A' describes research work done to predict stock prices using time series data only. The subsection 'B' depicts the models based on the time series data along with textual data obtained from the news articles. The insights of NLP techniques that can be used is given in subsection 'C' and subsection 'D', gives a short description of how concept drift can be detected.

A. Times series data

Stock prices can be seen as a time series data since stock prices are tagged with a timestamp. Sometimes normal feature extraction from this data does not reflect the behaviour of the stock market. To address this issue of limited feature extraction, Chen et.al. [1] proposed a new trend prediction model (TPM) based on the long term and short-term features. The system extracts long term features from multiple points in time by using piecewise linear regression technique, and CNN is used to extract short term features from different data at the same point in time. Long term features reflect temporal span and short-term features reflect spatial span of the data. To map different features into uniform format, the system uses encoder-decoder framework, where encoder converts features into input vectors of fixed size and decoder processes the vectors into final output. However, encoder and decoder may fail if there are too many features or too much information extracted from data. This problem can be solved by incorporating attention mechanisms which derive hidden state from relevant features. TPM system uses attention mechanism during both encoder and decoder phase, which gives better predictions of stock market trends.

The model proposed by Wen.et.al. [8] aims at predicting stock prices just by using time series data. In the background research, the authors claim that current time series prediction methods depend on correlation from time series dependence, signal decomposition of time series and time series pattern detection. The paper suggests sequence reconstruction using motif-based CNN. Experiments by the authors revealed that this approach is almost better than all other approaches of time-series prediction. This is also computationally less expensive than RNN based systems.

Lee et.al. [9] proposed a very unique approach for stock prediction. They use stock chart images as an input to predict the stock prices. Closing price and volume of stock was gathered from Yahoo Finance. From this data visual charts were created. The chart had a height of 8 blocks. The block could be either filled or empty. Each column in the chart represented a day. The upper filled blocks represent the relative value of closing price and the lower filled block represents the relative closing volume. This data is used to train Deep Q Network with CNN function. The model gave the result of whether to buy or sell. The results showed that the model usually showed profitable output. They also showed that a model trained by using the US market works well for other world markets.

Chen et.al. [10] presented a state-of-art to compare the performance of multiple prediction models for predicting the value of Chinese stock index. Deep Learning, Back Propagation Neural Network, Extreme learning machine and radial basis function neural network are used for the comparison studies. They used CSI 300 dataset which is a Chinese stock market index. The dataset was divided into 3 groups. These three groups were used on 3 different ML models (Back Propagation Neural Network, Extreme learning machine and radial basis function neural network) to get their prediction accuracy result. These results were compared with their proposed system which combines deep learning with an autoencoder and restricted Boltzmann machine. The results indicate that this novel Deep Learning approach gives better prediction results than the other 3 ML models.

Aithal et.al. [11] proposed a model that aims at identifying the major economic factors affecting the stock index and then using them to predict the stock index. The authors used 44 macro-economic factors and tried to find their influence on BSE and NSE. The authors applied Correlation, KMO, Bartlett Test for Pre-data reduction and then used PCA, K1-Kaiser and Scree Test for data reduction. This step reduced the dimensionality of data from 44 to 7. PCA with Varimax was used for Data Grouping to find factors with maximum variation. To identify which factors influence the stock market, regression analysis was used. Finally, a simple NN with sigmoid and ReLU was used to train a model for prediction. The authors claim that this model had an accuracy of 92% for Nifty and 87% for SENSEX.

Prediction of Indian stock market: 'Sensex' and 'Nifty', by using time series data with financial forecasting is an approach of the system proposed by Idrees et.al. [12]. Since the prediction of future values based on the historical data is quantitative forecasting, Auto Regressive Integrated Moving Average (ARIMA) model is used for stock market prediction. ARIMA method is a time series analysis used to predict the future time series values based on the past values. In this research Indian stock market data from January 2012 to December 2016 has been gathered. The collected data is treated as stationary data which is considered as lacking trend and seasonality in nature. Furthermore, the estimation of Auto-correlation function (ACF), Partial auto-correlation function

(PACF) is done, which easily signifies the relation between the observations in a time series with one another. ARIMA method is applied in order to find the pattern and forecast the feature based on those patterns. In this model, both Primary data (does not have any previous existence) and secondary data (collected previously at some time) is collected and analyzed for the revealing patterns or trends in the data. Auto regressive model is used to identify the future behavior of the variable, where it includes linear combination of historical value of that variable. Moving Average model uses error terms for prediction instead of historical value. ARIMA Model is an integration of these two models where historical data and its associated error terms are considered for stock price prediction. With this method they have achieved only a deviation of 5% from Nifty and Sensex on an average.

Prediction of stock price requires study of a large number of technically influencing features called as technical indicators. Biclustering method for analyzing technical indicators along with KNN can be used in detecting trading patterns. But this method often becomes unreliable. Hence Huang et.al. [13] proposed the novel method where trading patterns from technical indicators are extracted using biclustering algorithm and then trading actions are determined with fuzzy inference system. Instead of defining the fuzzy rules by experts, they are automatically mined from the trading data. This algorithm was tested against six methods including BIC-KNN on three datasets and the BM-FM performed better. In this model, collected data is fed to Biclustering Mining, where data is projected into 2D data matrix and essential technical indicators are selected. The set of trading rules defined by the bicluster mining method, are fed to a fuzzy inference method where a threshold value is learnt based on which rule gives buy or sell trading signals.

Prediction of Stock market using Machine Learning is based on the input dataset and its dimensionality. Since the stock prediction can vary with N number of attributes, Alsubaie et.al. [14] proposed a system that focuses on achieving high cost sensitive accuracy with minimum technical indicators or input features of the data set. Numerous vector attributes which represent several TIs are calculated everyday for stock price prediction. This paper research on how the number of TIs influence the stock market prediction along with the closing price of a certain period. In this research authors used 9 different classifiers namely, NB, FTNB, ANN, SVM, Meta-Cost wrapper with NB (M-NB), MetaCost wrapper with FTNB (M-FTNB), MetaCost wrapper with ANN (M-ANN), MetaCost wrapper with SVM (M-SVM), and the newly proposed CSFTNB. For each classifier, accuracy, misclassification error and expected investment return are calculated. 50 most common TIs are used in the beginning and five different feature selection methods are applied to get these top most influencing TIs. The paper proposed Cost sensitive fine-tuning Naive Bayesian classifier to fine tune the probability values to minimize misclassification cost. Paper concludes that choosing a large number of TIs like 30 TIs will reduce the performance accuracy, 10 TIs are shown to have the highest performance. The CSFTNB model showed the highest balance between investment return and associated risk.

B. Time series data and news data

With years of work done to achieve better stock market prediction, researchers have found that external events and mood of people also affect stock price. Belief and mood of people is better captured in terms of sentiments from public opinion platforms like twitter. Batra et.al. [2] proposed a system that predicts sentiment score from public tweets related to stock market and then sentiment score is combined with stock market numerical data to predict stock price for next day. The system calculates aggregate sentiment score from tweets for a day and trains the SVM model to predict stock price for the next day and to predict investment decision. This system was trained using tweets from StockTwits API and Yahoo Finance data. The paper concludes that sentiment analysis also plays a key role in deciding whether investors should buy or sell stocks.

The different key factors that affect stock price are temporal effects of news items and representation type of information embedded in the news events. Traditional techniques worked on the bag-of-words aspect for news information. This may lead to the problem of sparsity of words. To solve the above problems, Vargas et.al. [3] proposed a deep learning-based model that makes use of sentence embedding technique for news information representation and only news titles were used to predict stock prices on a daily basis. The system generated word2vec models from news titles and extracts technical indicator factors that affect direction of stock price movement. The proposed RCNN model predicted directional movement of stock prices and outperformed CNN and RNN models. This paper concludes that sentence embedding is better than word embedding, news information has short temporal effect on stock prices and deep learning models can predict much better than other models.

Along with an increase in the number of information sources, stock market prediction based on sentiments of news articles are becoming popular. However, the availability of news articles to derive sentiments is one of the big challenges. The news articles distribution is uneven or imbalanced, which means that certain stocks may have a large number of news articles while some stocks may not have sufficient news articles to train the machine learning model. To address this problem, Li et.al. [15] proposed a sentiment transfer based system in which the knowledge learnt from news rich stocks is transferred to the news poor stocks. Two groups of stocks are created based on the availability of news articles as: news rich stocks and news poor stocks. The stocks in both of the groups are analyzed to derive correlation between them and knowledge from the models trained on news rich stocks is transferred to the models that will learn based on news poor stocks. The system correlates stocks in both groups using three comparison principles and the majority voting system is used to define the similarity between stocks from both groups. This helps in generating feature space for news poor stocks and thus, predicting stock prices more accurately. This work concludes that handling the issue of news articles imbalance and performing majority voting leads to better performance of the system.

Volatility of stocks is believed to be dependent on the two market factors namely: fundamental information and news reports. Gathering of data to predict future stock price is based on a multimodal data aspect, where information comes from various mediums. The primary challenge is to process the interrelated information present in these information bearer mediums (i.e. fundamental information and news). Traditional studies used to concatenate the information from various data sources into a compound vector. However, the problem is that they ignored the processing of the interrelated information. To overcome this problem, Li et al. [16] proposed a tensor representation approach to model the multimodal market information, which takes care of interrelated information. The secondary challenge to process a multimodal information is dealing with its heterogeneous sampling. Fundamental information consists of continuous data flow at fixed time intervals, whereas news contains randomly sampled data. Schumaker et.al. [17] addressed this problem by considering only that data within an interval of 20 minutes following the release of the news. Only a part of the data coinciding with the news was considered by discarding the rest. This caused partial miss of data and feature space distortion. Previous work focused on a one-to-one problem entailing that news only affects those stocks which are mentioned in its reports. Furthermore, they ignored the impact of news to the other related stocks. This article overcomes the above problem by using an event-based LSTM model which controls the memory in a neural network. In this manner, the authors addressed various challenges in processing the information used for predicting a stock price.

Kanungsukkasem et.al. [18] proposed a model which is based on the 'latent Dirichlet allocation' concept used to explain the similarity between the data. The proposed model known as Financial LDA (FinLDA) is used to discover the features from the textual data especially news. These extracted features are fed into machine learning algorithms to predict the time series data effectively. Two variants of FinLDA were proposed during their research, where one took discrete input data and other took continuous data describing changes. Final output from these two variants were fed into machine learning algorithms to compare the effectiveness of FinLDA features with other LDA features. The authors claimed that the extracted features from FinLDA performs better prediction than comparative features of common LDA.

Stocks information can be found from various sources like Wikipedia search pattern, media news, and social media. Two unique trading philosophies namely: fundamental and technical analysis, are used to predict stock market. Technical analysis focuses on the survey of charts to study the market actions, whereas fundamental analysis focuses on demand and supply of data affecting the stocks. These days, the use of Natural Language Processing (NLP) techniques is widespread amongst many research projects focusing on stock price prediction. Shah et.al. [19] proposed a model that retrieved, extracted, and analyzed the impact of news sentiments on the stock market. They have worked for developing the sentiment analysis dictionary for the financial sector, dictionary-based sentiment analysis model, and have evaluated that model to measure sentiment of the news. Furthermore, the authors suggested incorporating a word weighing approach for assigning the sentiment score.

Sentiment analysis on the news data plays a vital role in determining the stock price movements. Previous research projects have found the correlations between news information and financial data belonging to the particular stocks, to determine the future stock price. Descriptive task and Predictive task are the two categories of Data mining tasks. Khedr et.al. [20] proposed a system that implements predictive task strategy where stocks' historical data and news information is used for building a classification model. They have used Naive Bayes and KNN algorithms to build the classification model. To perform text mining on news data, authors started by determining polarity of the news. Polarity classifies the nature of news information as 'positive' or 'negative'. Moreover, they analyzed the numeric data of stocks' historical opening, high, low, and closing (OHLC) prices. Following these steps, they got two vectors of stocks' numerical data and labeled news articles respectively. The resulting concatenated feature vector is produced by combining two vectors based on the date. Finally, they used KNN algorithm to predict the future stock price behavior 'rise' or 'fall'. They concluded that KNN algorithm has a higher acceptance rate when numeric stock data is incorporated with textual news data.

Stock market prediction can be done using Time series prediction models. One such model is ARIMA. But due to the non-linear nature of stock prices such models are not good enough to capture the underlying pattern. Zhang et.al. [21] have proposed a technique to integrate news data into it. But before that NLP must be used to convert textual news into mathematical representation. The authors explored bag-of-words models like word2vec and GloVe to convert text into mathematical dimensions. For the final prediction the authors proposed a deep learning model called Extended Hybrid Attention Network (HAN). A general Hybrid Attention Learning Model is a type of RNN which learns important news and temporal patterns in them to predict the final value. The authors extended it by injecting sentiment scores and stock information into the network. Experiments by the authors showed that the Extended HAN outperforms all the baseline models like Naive Bayes, Logistic Regression and time series models.

Moukalled et.al. [22] proposed a model that combines mathematical calculations and news sentiment analysis with machine learning model to predict the stock movements. Many factors like historical prices, tweets from the president, public statement and social media etc. affect the stock price. Fundamental analysis and technical analysis of the company are taken into account during stock prediction. News sentiments and Historical prices are the two data sources. Around 10 years of data is collected from apple, google, Facebook and amazon. Tick data, which measure upward or downward movement of price includes details like open bid, close bid, high bid and low bid along with timestamp. This is used to predict intraday prices. The data is trained with RNN, FFNN, SVM and SVR to predict the direction of stock price. Out of all these, SVM performed best and achieved up to 85% accuracy. This includes the effect of news sentiments on a daily basis.

Aligning the tick data with news sentiments results in such improvements of prediction rate. Tick data is a measure of minimum upward or downward movement of stock price. In preprocessing these tick data, details are moved to MySQL database and sorted. Missing tick values are filled with nearest tick data. News sentiment data is aligned with this data, based on the time intervals of news and tick data. 8 different features are selected for predicting the end of the day price. These data features are normalized using MinMaxScaler since they are in different units. Then they used RNN, FFNN, SVM and SVR methods to train this data. SVM is found outperforming compared to other models.

C. NLP Techniques

Sentiment analysis has gained popularity as they help in extracting sentiments and opinions of the people from textual data like reviews, news articles, etc. In order to extract such knowledge, Natural Language Processing (NLP) techniques are used. The researchers have developed many algorithms for efficient and accurate knowledge discovery.

The research study done by Abdullah et.al. [23] aims to extract vital information from various sources of the news and analyze them to predict stock prices. The authors proposed a framework which uses their text parser and analyzer algorithm along with an open source NLP tool. This helped them to retrieve necessary information from the news using their text analyzer algorithm (if authentic information source) and NLP (if unauthentic information source). Following this, they analyzed the information by using machine learning and text mining techniques. Finally, they predicted the stock prices using the historical data. They provided rank/weight to the retrieved information based on the comparison made with the historical data. They claimed that their research is different from previous studies as they have combined text parsing techniques from two types of text data (patterned and scattered).

Stock price prediction based on time series data has been challenging. It has been noticed that the search volume (on any authenticated information sources) highly affects the stock price movement. Wang et.al. [4] proposed a Hybrid time-series predictive neural network model which combines news affection. They pulled out the features from the news headline and structured them in the form of vectors of distributed words. Those vectors' dimensions are reduced with the help of sparse automatic encoders which removed unnecessary information. They combined daily stock data with the news by analyzing its arrival timeline. Their proposed HTPNN model is used with deep convolutional layers for capturing text features and LSTM layer for learning stock price fluctuation. They claimed that the classification frequency of the HTPNN model is higher than other models.

Wang et.al. [24] proposed the idea of doing NLP using scope based CNNs. Currently it is done with the help of a sliding window technique. Current systems have a corpus of data (sentences) for training CNN. CNN used a sliding window to find the features and trained the system based on labels of the training dataset. But the problem is that we cannot predict window size. Sometimes languages can have discontinued sentences where related words are far away in the sentence. The authors proposed that scope-based systems will solve this issue. Experiments by authors found that it is better than other CNNs and LSTMs.

The system proposed by Maldonado et.al. [25] focuses on detecting technical debts in the code base using NLP. Technical debts are basically hacking or workarounds which developers have to do due to deadline pressure. Usually the developers leave some comments near that hack. The goal of this research was to find the debts. The authors first took the code of many open source projects. Following that, they filtered the comments to get the comments which will be useful for the purpose. Then they manually classified the comments into various technical debts categories. They used a maximum entropy classifier to train a classifier based on these comments. Such a classifier is like Naive-Bayes but the difference is that maximum entropy classifier can give good results when the features are dependent. Stanford classifier is an implementation of maximum entropy classifier which was used. It generates features (words) for each sentence which are associated with positive or negative votes for each class. It is basically a multiclass regression model. Experiments found that this approach has almost 90% accuracy.

Ding et.al [26] proposed a model that describes how the background knowledge will add value to the extracted events from the news in prediction of stock price. This paper is an improvised version of event embedded prediction model. The paper proposed to add the proper knowledge to the events from the knowledge graph, so that while comparing unrelated events are not put in the same bucket, even though they contain similar words. Neural network tensor is used for both event embedding and knowledge graph embedding. Joining event embedding and knowledge graph can get the similarity between the events with the knowledge how likely they are, so that event impact can be analyzed based on another. Structured events are extracted from the news, YOGO knowledge graph is used for relation extraction between the events. Paper compares this knowledge driven event embedding systems with Baseline method and discrete event vectors with semantic lexicons-based generalization method, with significant improvements in estimation. In this work, event embedding model is used to represent event tuples as small dimension dense vectors. This event embedding model is used as the base for the proposed model. Knowledge graph is used to get the relationships between the entities in the events. By combining these two, similarities between the two events are calculated. Prediction is based on how likely past similar events affect the stock price changes. After knowledge driven event embedding to the data, CNN model is applied to the prediction.

Liu et.al [27] proposed a temporal convolution network (TCN) for extracting information from the previous data to predict future results. As the TCN has proven efficiency with audio synthesis, text recognition, natural language processing, paper introduced modified version in order to better time series prediction. To facilitate the gradient flow Gated Linear Units (GLU) is added to the TCN. Multi-Channel Gated temporal convolution network(M-GTCN) is an improved ver-

sion of GTCN, which extract more features and improve the performance of the model. Temporal convolution network has a conventional structure and it doesn't use sequence modeling. Residual structure is added to TCN to overcome the mismatch between input and output using GLU forming new method GTCN. Multiple layer residual structure is embedded to extract features separately and fuse these features to get the prediction through connected layers. This model is proposed as a Multi-Channel Gated temporal convolution network(M-GTCN). Collected data is normalized and fed to M-GTCN model and prediction are better compared to LSTM and GRU.

D. Concept drift detection

The advancements in technology have created a large amount of data through applications like stock market, telecommunication and banking. These applications continuously generate a huge amount of data such that it is not feasible to first store such data and then process it like traditional data mining techniques. Such data is generally referred to as data streams and applications require quick responses based on it. Such continuous stream of data like stock price data is collected and processed on fly. This data comes from an environment which is not stationary and there may be evolution of data sources or data distributions. The non-stationary nature of such data streams may introduce a phenomenon called concept drift into data streams [28]. The drift is said to occur when the concept about which data is collected changes over a certain time period and this change degrades the performance of the prediction system. Since a machine learning model is trained on old concepts, it may fail to predict class labels of data from new concepts. Hence, it is necessary to detect such drifts from data streams. Many drift detection algorithms have been proposed over years of research and are discussed below.

The common problems in time series prediction are introduction of concept drift and class imbalanced data. Financial time series like stock market is a non-stationary environment where data instances with class imbalance may be observed and this may derive a biased result towards majority classes observed. To solve these both problems, Thalor et.al. [29] proposed a new ensemble-based learning model, which helps in detecting drift and balancing class labels. The system keeps an ensemble of classifiers trained on data at a certain point of time and after certain amount of time has passed, it monitors performance of the model. Based on error rate monitoring, drift is detected and the misclassified data instances are separately stored into a buffer. This buffer is combined with some correctly classified data instances to maintain class balance and then new classifier is trained on this data to learn old as well as new concepts. The newly trained classifier then replaces the poorly performing classifier from the ensemble.

Most of the recent work done for drift detection is based on error rate monitoring. However, if the drift is introduced due to change in probability distribution of data, then error rate monitoring may not efficiently detect drifts. To address this problem, Gamage et.al. [5] proposed an ensemble model that keeps track of distance metrics between probability distribution at two different time windows and detects drift when sum of continuous values of distance metrics exceeds threshold value. The system also adapts new concepts by learning a new model over new window of data and adding a new classifier to the ensemble. The existing base classifiers in ensemble are reassigned weights based on their performance over new window and the worst performing base classifier is omitted from ensemble. The paper concludes that the adaptive nature of ensemble gives efficient results when recurrent drift is observed.

E. State-of-the-Art Summary

Financial investors are greatly interested in stock price movements to determine the best time for selling or buying stocks in order to maximize their profits. Few decades back, the stock market was popularly known as an unpredictable place and investors were not able to predict the stock prices. Random Walk Theory [6], [7] proposed that the stock market is not predictive and stock prices are not dependent on any factor. However, later studies showed that stock market can be predicted if underlying patterns are extracted and understood well. Many machine learning models have been proposed over the past decade for stock price prediction. The study has shown that techniques of stock price prediction can be broadly categorized into two groups: techniques that use only time series data from stock market and techniques that use both time series and textual data from the sources like new articles, user reviews etc.

There are many systems proposed to predict stock prices based on time series data of stock prices. The most commonly used technique for keeping track of stock price movement is moving average technique. The moving average of stock prices determine the amount of stock price movement for next day in increasing or decreasing fashion. The averages calculated over sliding windows of time series data are generally used as moving average values. The most popular machine learning technique used for predicting stock prices based on time series data is Neural Networks algorithm. Many variations of neural networks have been proposed to efficiently predict stock prices. These systems also suggest that there are certain technical indicators/features extracted from time series data which contribute more while performing prediction tasks.

The recent studies have shown that stock prices are affected not only by current trend in stock price movements but also by external factors like news articles, political events, laws around the world. The sentiments of investors, news of incidents or financial reports also play a vital role in stock price determination. Thus, it is necessary to incorporate textual data from such sources along with time series data to better predict stock prices. Many news-based models have been proposed until this date and 'bag-of-words' technique is most-widely for extracting features from textual data. The sentiments of financial investors also contribute towards stock market trading and hence, sentiment analysis of news articles, financial reports play an important role in understanding stock market patterns. However, most of the work done in this area focuses on weightage of words rather than weightage of sentence as a whole. Further, the systems studied in this literature survey

show that these models use sentiment mining for determining the direction of stock price movements rather than the extent of effect on stock prices.

The sentiment mining can be achieved using Natural Language Processing (NLP) techniques by extracting feature vectors from textual data. The words from textual data sources are used to represent feature vectors and we can obtain these vectors by using parser and analyzer techniques. Many papers from this literature survey have assigned weights to vectors of words based on their TF-IDF (Term Frequency- Inverse Document Frequency). Some papers have also proposed techniques of dimensionality reduction since textual data may result in very large feature vectors. The features can further be categorized into two groups based on time period during which they can affect stock prices as: short-term features and long-term features. This distinction will help in discovering helpful patterns underlying stock market data.

The stock market is considered to possess a very dynamic environment and it can exhibit any kind of changes either suddenly or gradually. Such changes in data are often defined as concept drift and it is an important task to detect such changes since these changes may affect stock market prediction. Almost all papers on concept drift from this literature survey use error-rate monitoring method for detecting concept drift in data. Furthermore, studies conclude that concept drift detection and adaptation can be handled in the best way using ensemble classifier for stock price prediction.

Thus, this literature survey concludes that we can build a better prediction model by using time series data and textual data from the news articles. Sentiment mining should be done in a way to determine the magnitude of effect of textual data on stock prices, where magnitude of effect is the quantitative movement of stock prices. Since the stock market is a non-stationary place, we can further study if concept drift detection and adaptation help in efficiently predicting the non-deterministic nature of stock market.

VIII. Conclusions

This study is an experimental analysis of the effect of news data on stock market prediction and application of concept drift in identifying fluctuations in the stock market due to unpredictable situations around the world. One major experiment of the study is analysing implication of drift detection on the stock market, to provide better and reliable prediction algorithms. We found that RNN, LSTM and Linear regression model provides more than 88 percent accuracy and ensemble model with these algorithms can be used to achieve better prediction results.

We developed an ensemble model with LSTM, RNN and Linear regression to achieve the better prediction accuracy. This ensemble model provide promising results while analysing for drift detection. Analysis of drift in the data is helped to build the adaptive approach to learn new concepts. This ensemble model with drift analysis proved to be performed best compared to the other experiments conducted.

A web application is presented to user, so that they can see the stock trend, they can see our prediction for next day, predicted using our ensemble mode. We also give insights to the stock changes by providing information to the news impacting the stock of the particular company.

In future, our target is to implement a system which detects multiple kind of drifts, analysing those drifts to find novelty. Also targeting to analyse these drifts for authenticity, and design a model to detect virtual drifts.

Acknowledgment

Inspiration and guidance are invaluable in every aspect of life, especially in the fields of academics, which we have received from our respected project advisor Prof. Mahima Agumbe Suresh. We would like to thank her for her endless contributions of time, effort, valuable guidance and encouragement she has given us. We are grateful to the reviewers for their comments and suggestions that helped us to improve the report. At the last it is our immense pleasure to mention our families and friends for their support and motivation.

References

[1] Y. Chen, W. Lin, and J. Z. Wang, "A dual-attention-based stock price trend prediction model with dual features," *IEEE Access*, vol. 7, pp. 148 047–148 058, 2019.

[2] R. Batra and S. M. Daudpota, "Integrating stocktwits with sentiment analysis for better prediction of stock price movement," in *2018 International Conference on Computing, Mathematics and Engineering Technologies (iCoMET)*. IEEE, 2018, pp. 1–5.

[3] M. R. Vargas, B. S. De Lima, and A. G. Evsukoff, "Deep learning for stock market prediction from financial news articles," in *2017 IEEE International Conference on Computational Intelligence and Virtual Environments for Measurement Systems and Applications (CIVEMSA)*. IEEE, 2017, pp. 60–65.

[4] Y. Wang, H. Liu, Q. Guo, S. Xie, and X. Zhang, "Stock volatility prediction by hybrid neural network," *IEEE Access*, vol. 7, pp. 154 524–154 534, 2019.

[5] S. Gamage and U. Premaratne, "Detecting and adapting to concept drift in continually evolving stochastic processes," in *Proceedings of the International Conference on Big Data and Internet of Thing*, 2017, pp. 109–114.

[6] E. F. Fama, "The behavior of stock-market prices," *The journal of Business*, vol. 38, no. 1, pp. 34–105, 1965.

[7] ——, "Random walks in stock market prices," *Financial analysts journal*, vol. 51, no. 1, pp. 75–80, 1995.

[8] M. Wen, P. Li, L. Zhang, and Y. Chen, "Stock market trend prediction using high-order information of time series," *IEEE Access*, vol. 7, pp. 28 299–28 308, 2019.

[9] J. Lee, R. Kim, Y. Koh, and J. Kang, "Global stock market prediction based on stock chart images using deep q-network," *IEEE Access*, vol. 7, pp. 167 260–167 277, 2019.

[10] L. Chen, Z. Qiao, M. Wang, C. Wang, R. Du, and H. E. Stanley, "Which artificial intelligence algorithm better predicts the chinese stock market?" *IEEE Access*, vol. 6, pp. 48 625–48 633, 2018.

[11] P. K. Aithal, A. U. Dinesh, and M. Geetha, "Identifying significant macroeconomic indicators for indian stock markets," *IEEE Access*, vol. 7, pp. 143 829–143 840, 2019.

[12] S. M. Idrees, M. A. Alam, and P. Agarwal, "A prediction approach for stock market volatility based on time series data," *IEEE Access*, vol. 7, pp. 17 287–17 298, 2019.

[13] Q. Huang, J. Yang, X. Feng, A. W.-C. Liew, and X. Li, "Automated trading point forecasting based on bicluster mining and fuzzy inference," *IEEE Transactions on Fuzzy Systems*, 2019.

[14] Y. Alsubaie, K. El Hindi, and H. Alsalman, "Cost-sensitive prediction of stock price direction: Selection of technical indicators," *IEEE Access*, vol. 7, pp. 146 876–146 892, 2019.

[15] X. Li, H. Xie, R. Y. Lau, T.-L. Wong, and F.-L. Wang, "Stock prediction via sentimental transfer learning," *IEEE Access*, vol. 6, pp. 73 110–73 118, 2018.

[16] Q. Li, J. Tan, J. Wang, and H. Chen, "A multimodal event-driven lstm model for stock prediction using online news," *IEEE Transactions on Knowledge and Data Engineering*, 2020.

[17] R. P. Schumaker and H. Chen, "A quantitative stock prediction system based on financial news," *Information Processing & Management*, vol. 45, no. 5, pp. 571–583, 2009.

[18] N. Kanungsukkasem and T. Leelanupab, "Financial latent dirichlet allocation (finlda): Feature extraction in text and data mining for financial time series prediction," *IEEE Access*, vol. 7, pp. 71 645–71 664, 2019.

[19] D. Shah, H. Isah, and F. Zulkernine, "Predicting the effects of news sentiments on the stock market," in *2018 IEEE International Conference on Big Data (Big Data)*. IEEE, 2018, pp. 4705–4708.

[20] A. E. Khedr, N. Yaseen *et al.*, "Predicting stock market behavior using data mining technique and news sentiment analysis," *International Journal of Intelligent Systems and Applications*, vol. 9, no. 7, p. 22, 2017.

[21] Z. Zhang, X. He, and J. Ge, "News-oriented stock price trend prediction."

[22] M. Moukalled, W. El-Hajj, and M. Jaber, "Automated stock price prediction using machine learning," in *Proceedings of the Second Financial Narrative Processing Workshop (FNP 2019), September 30, Turku Finland*, no. 165. Linköping University Electronic Press, 2019, pp. 16–24.

[23] S. S. Abdullah, M. S. Rahaman, and M. S. Rahman, "Analysis of stock market using text mining and natural language processing," in *2013 International Conference on Informatics, Electronics and Vision (ICIEV)*. IEEE, 2013, pp. 1–6.

[24] J. Wang, Y. Li, J. Shan, J. Bao, C. Zong, and L. Zhao, "Large-scale text classification using scope-based convolutional neural network: A deep learning approach," *IEEE Access*, vol. 7, pp. 171 548–171 558, 2019.

[25] E. da Silva Maldonado, E. Shihab, and N. Tsantalis, "Using natural language processing to automatically detect self-admitted technical debt," *IEEE Transactions on Software Engineering*, vol. 43, no. 11, pp. 1044–1062, 2017.

[26] X. Ding, Y. Zhang, T. Liu, and J. Duan, "Knowledge-driven event embedding for stock prediction," in *Proceedings of coling 2016, the 26th international conference on computational linguistics: Technical papers*, 2016, pp. 2133–2142.

[27] Y. Liu, H. Dong, X. Wang, and S. Han, "Time series prediction based on temporal convolutional network," in *2019 IEEE/ACIS 18th International Conference on Computer and Information Science (ICIS)*. IEEE, 2019, pp. 300–305.

[28] H. Wang and Z. Abraham, "Concept drift detection for streaming data," in *2015 International Joint Conference on Neural Networks (IJCNN)*. IEEE, 2015, pp. 1–9.

[29] M. A. Thalor and S. Patil, "Learning on high frequency stock market data using misclassified instances in ensemble," *Learning*, vol. 7, no. 5, 2016.

Aditee Vasant Jadhav is graduate student of San Jose State University for Computer Engineering major. Her research interests include data mining, machine learning, recommendation systems. She has previously carried out research work in the field of data science and she has published 5 research papers.

Kavyashree ChandraShekar is a graduate student of SJSU for the Computer Engineering major. Her research interests lies in machine learning, web and mobile application development.

Krupa Dhirajlal Vadher is graduate student of San Jose State University for Computer Engineering major. Her research interests include data mining, machine learning, large scale analytics.

Pranav Sudhir Dixit received the B.E. degree from Savitribai Phule Pune University in India. He is currently pursuing an M.S. degree in Computer Engineering at San Jose State University. His research interests include Neural Network, Deep Learning, Computer Networks, and Computer Systems.