Real-Time Data Streaming and Processing

A major project report submitted in partial fulfillment of the requirement for the award of degree of

Bachelor of Technology

in

Computer Science & Engineering / Information Technology

Submitted by

Adarsh Pandey (201565) & Vijay Deep Jain (201265)

Under the guidance & supervision of

Dr. Hari Singh

Department of Computer Science & Engineering and Information Technology

Jaypee University of Information Technology, Waknaghat, Solan - 173234 (India)

CERTIFICATE

This is to certify that the work which is being presented in the project report titled **'Real-Time Data Streaming and Processing'** in partial fulfillment of the requirements for the award of the degree of **B.Tech in Computer Science And Engineering / Information Technology** and submitted to the Department of Computer Science And Engineering, Jaypee University of Information Technology, Waknaghat is an authentic record of work carried out by **Adarsh Pandey (201565) and Vijay Deep Jain (201265)** during the period from August 2023 to May 2024 under the supervision of **Dr. Hari Singh (Assistant Professor(SG))**, Department of Computer Science and Engineering, Jaypee University of Information Technology, Waknaghat)

I also authenticate that I have carried out the above mentioned project work under the proficiency stream Cloud Computing.

Submitted by:
Adarsh Pandey (201565)
Vijay Deep Jain (201265)

This is to certify that the above statement made by the candidate is correct to the best of my knowledge.

Supervised by:
Dr. Hari Singh
Assistant Professor
Computer Science & Engineering and Information Technology
Jaypee University of Information Technology, Waknaghat

Candidate's Declaration

We hereby declare that the work presented in this report entitled **'Real-Time Data Streaming and Processing'** in partial fulfillment of the requirements for the award of the degree of **Bachelor of Technology** in **Computer Science & Engineering / Information Technology** submitted in the Department of Computer Science & Engineering and Information Technology**,** Jaypee University of Information Technology, Waknaghat is an authentic record of my own work carried out over a period from August 2023 to May 2024 under the supervision of **Dr. Hari Singh** (Assistant Professor(SG), Department of Computer Science & Engineering and Information Technology).

The matter embodied in the report has not been submitted for the award of any other degree or diploma.

(Student Signature with Date)	(Student Signature with Date)
Adarsh Pandey	Vijay Deep Jain
201565	201265

This is to certify that the above statement made by the candidate is true to the best of my knowledge.

(Supervisor Signature with Date)
Dr. Hari Singh
Assistant Professor (SG)
Computer Science & Engineering / Information Technology
Dated:

ACKNOWLEDGEMENT

We are highly indebted to all the members of the Department of Computer Science & Engineering and Information Technology, Jaypee University of Information Technology for their steerage and constant supervision while providing necessary data related to the project and additionally for their support in finishing of the project. We would also like to express our gratitude towards Dr Hari Singh, Assistant Professor (SG), as our project Supervisor, for his kind cooperation and encouragement which helped us in the completion of this project and for giving us such attention and time.

TABLE OF CONTENT

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

LSTM: Long Short-Term Memory

RNN: Recurrent Neural Network

CNN: Convolutional Neural Network

Bi-LSTM: Bidirectional LSTM

GRU: Gated Recurrent Unit

RF: Random Forest

SVM: Support Vector Machine

GPU: Graphical Processing Unit

CPU: Central Processing Unit

RAM; Random Access Memory

RE-LU: Rectified Linear Unit

ABSTRACT

This study presents a comprehensive approach to real-time data streaming and processing, focusing on sentiment analysis for web-scraped data. Initially, a historical dataset is utilized to train nine distinct models, including recurrent neural networks (RNN), long short-term memory networks (LSTM), convolutional neural networks (CNN), CNN-LSTM hybrids, bidirectional LSTMs (Bi-LSTM), CNN-BiLSTM hybrids, gated recurrent units (GRU), support vector machines (SVM), and random forests (RF).

Following model training, the sentiment analysis framework is implemented on real-time web-scraped data streams. The proposed methodology aims to provide a robust and versatile solution for continuous sentiment analysis, facilitating timely insights into evolving trends and sentiments in web-based content.

Further, sentiment analysis is also applied on twitter based dummy dataset to show the applicability of our project. Using textblob analysis, polarity is obtained which further helps us get the sentiment score and assign positive and negative weightage which further will provide assistance in getting price affected by sentiment analysis.

CHAPTER 1: INTRODUCTION

1.1 INTRODUCTION

The Combination of deep learning with real time data entry and the use of modern algorithms has ushered in a new era of accurate and flexible currency price forecasting in the banking world, where milliseconds can mean a world of gains and losses. The integration of these technologies is becoming increasingly important for individuals who want to solve stock trading challenges with accuracy and predictability as market momentum increases rapidly.

Continuous and rapid analysis of incoming market data is what instantaneous data and analysis means in stock price forecasting. With this dynamic approach, market participants can eliminate static, batch-oriented analysis and quickly adapt to real time changes in the market. The ability to use sophisticated algorithms to make accurate and robust forecasts is as important as the speed of data processing in the paradigm shift.

Extracting important sites from ever growing macroeconomic data relies heavily on machine learning and the use of increasingly sophisticated algorithms.These algorithms can identify complex relationship patterns and anomalies that traditional analysis the path can be lost. These algorithms enable Trader and investors to constantly adjust real time models, learn from the past data and make intelligent choices based on the latest information.

Deep learning showed that it was most successful in explaining the complex market swings because of a neural network structure, which can handle complicated and unpredictable connections in the data. Deep learning models from recurring neural networks (RNN) that are excellent at capturing sequential dependencies to long- and short-term memory networks (LSTM) which are good at learning from time series data.

The deep learning tools are powered by machine learning algorithms, which can be as simple as random forest Like cluster method or as complex as support vector machine learning. These algorithms offer a flexible and adaptable methods of stock price prediction Since they may be customized to certain financial instruments and market situations.

Using different news platforms and social media to understand the market sentiments about bitcoin and use it in price prediction can help minimize the gap between actual price and predicted price.So combining these technologies keeps the model updated to reflect new facts and market movements. Economists can gain a competitive advantage by combining sophisticated learning algorithms with real time data to anticipate market movements and can make informed decisions. As the world of Finance continues to change, such technologies are destined to play a role in forecasting stock prices to redefine concepts of accuracy, flexibility and responsiveness.

1.2 PROBLEM STATEMENT

For businesses looking to stay ahead of the competition and make wise decisions, it is now critical to capture and analyze real time data streams in the Fast-changing technology landscape. To address the difficulties associated with the real time data streaming and processing, this project offers a complete solution that makes it easier to gather, process and derive useful information from the data as it changes dynamically and results in effective decision making.

Challenges of efficiently monitoring real time data input: Traditional batch processing fails when dealing with frequently changing data resulting in processing delays and potential inaccuracies. To overcome this, our project focuses on a reliable and scalable system that can handle real time data from different sources and systems in an accurate and flexible manner.

The challenge of effectively predicting stock prices: The dynamic nature of financial markets requires a model that can quickly detect trends and anomalies in the database, especially those associated with the sudden change in stock prices.

Another challenge of ensuring data reliability: When there is a network issue or system crashes while predicting stock prices, The system should be design in such a way that it can handle such situation gracefully. If there is a connectivity hiccup, there should have procedures in place to recover and deal with any loss data once connectivity is restored. Maintaining consistent accuracy is important especially when dealing with stock price forecasts.

In short, this project is a bold attempt that aims to meet the challenges of gathering and processing real time data. It goes further by including the complexities of real time stock price prediction. The aim is to provide a complete solution to enhance organizational decision-making capabilities in fast-moving financial markets. The system promises to drive features like scalability, reliability and real time insights that will enable organizations to make faster and more informed decisions.

1.3 OBJECTIVES

1) Realtime data infrastructure development: This project has several main objectives to build a real time data system. First, it aims to effortlessly gather real time market data from a variety of sources such as social media, stock exchanges and financial news. Then, it will use the data for computing the future price of bitcoin and also analyze the news opinion about bitcoin. The system design prioritizes scalability to handle large amounts of data and user requirements.

2) Predictive models implementation: Different Deep learning methods such as LSTM, RNN, CNN, or combination of two or more algorithms will help capture complex patterns in stock price volatility. Machine learning methods such as Random forest, Support vector learning (SVM), GRU will also provide precise prediction. It will be important to analyze temporal correlation in stock price data and measure market sentiments from financial , social media and online news platforms related to bitcoin .To achieve this, the project will use different libraries that can find the sentiments and polarity of news, continue to train the predictive model with historical dataset of bitcoin and social media or news website such as twitter(X) and bitcoin magazine, allowing them to adapt to market fluctuations in a real time.

3) User friendly mobile Android application: The last part of the project focuses on creating an intuitive user interface that improves the usability and accessibility of real time market analytics including the news sentiment part.The interface will offer a customized and user-friendly platform for obtaining real time data and projections, catering to both new investors along with the seasoned financial professions. The project will use data visualization tools such as active graphical representation and

visualization, to show complicated data, in an easily understood way in order to meet these design objectives.

In summary, every aspect of the project is well defined with precise goals and plans, including the construction of real time infrastructure, the application of productive models and user interface design. This all-encompassing strategy is to provide consumers with the means of navigating the intricacies of real time market information enabling timely and well-informed decision making in a constantly changing financial environment.

1.4 SIGNIFICANCE AND MOTIVATION OF THE PROJECT WORK

The importance of this process is emphasized by the following important factors:

1) Informed decision making: In the fast-paced financial world, quick decisions can have a huge impact, our project focuses on using intelligent forecasting and rapid data processing to help economists and investors make smarter choices in an ever-changing portfolio. Early access to accurate insights helps decision makers seize opportunities and effectively manage and effectively manage risk.

2) Competitive edge: Getting ahead in today's tough financial market means embracing the latest technology. Our project focuses on powerful framework, prediction models and smart trading algorithms that put organization at the forefront of innovation. By blending machine learning, deep learning, news sentiment analysis and real time analytics we give market players a tech edge to spot trends early and take smarter risks. It's about using sophisticated tools to stay one step ahead of the competitive investment game.

3) Risk management: In an unpredictable market, it's important to be proactive about risk. Our work takes steps in using real time data to quickly identify and access potential risks. Investors need to act quickly to protect their assets from market fluctuations and our project will help them to do just that. It's about having the tool to use timely strategies and protect the portfolio from unexpected market fluctuations.

4) Market adaptability: In an ever-changing financial market our project is committed to continuously improving trading strategies and forecasting models. We rely on real time feedback to remain flexible and adapt to rapid changes in market conditions. This flexibility is critical to long-term success in an environment of rapidly changing market dynamics.

5) Financial inclusion and accessibility: Creating an easy-to-use interface provides real time market information and views to a wider audience from new investors to experienced professionals. This shared global financial knowledge promotes a more inclusive and participatory financial environment, and powers individual to make informed financial investment decisions.

6) Technical innovation: In the financial services, the sector with easy-to-use interface, technologically advanced through the integration of cutting-edge technologies such as machine learning, deep learning and real time analytics etc. This research not only addresses current challenges but also it pushes the boundaries of what is possible when using data to make strategic decisions within the world of finance.

In conclusion, in our project we are creating a smart system that not only process real time data but also predicts stock prices, manages risk, analyse the ongoing sentiment of news and empower everyone from new investors to seasoned professionals, with easy-to-use interfaces, we enable everyone to access complex market information, encouraging financial inclusion. Our use of cutting-edge technologies such as deep learning, machine learning and real time data Analytics not only solves current problems but advanced financial technologies. We don’t just adapt to fast moving markets we stay ahead and empower decision makers by providing change in a world where every second counts our work is the key to making informed timely strategy investment choices.

1.5 ORGANIZATION OF PROJECT REPORT

The project report is organized in a standard format and is organized as follows:

Chapter 1: Introduction - Provide a summary of the research topic, including background, research questions, aim and study importance.

Chapter 2: Literature survey - Conducts a thorough Examination of available literature on the project issue. The literature reviews analysis of prior research studies, theories and frameworks relating to the issue and finds gaps in the literature that the present study attempts to fill.

Chapter 3: System Development - Describes the project’s research and research design. This chapter describes the study’s data-gathering procedures, data processing methodologies, data analysis, and assessment methods.

Chapter 4: Testing - Presents and analyzes the outcome of the experiments carried out. This chapter offers a full analysis of the acquired data, a description of the experimental setup and an Evaluation of the findings produced from the different signal processing and feature engineering approaches.

Chapter 5: Results - Provide results and outcomes of the project work.

Chapter 6: Conclusion and future scope - Provides a summary of the research finding, examines the study’s relevance and explains the research’s contributions. This chapter also discusses the study weaknesses and makes recommendations for further research.

CHAPTER 2: LITERATURE SURVEY

2.1 OVERVIEW OF RELEVANT LITERATURE

Iyyappan. M et al [1] proposed a model that aims to assist individuals in making informed invested decisions by forecasting the closing price of stocks. The model inquires about user's needs regarding investment amount, duration of the investment and risk tolerance to recommend where to invest to gain more profit and reduce the losses. The proposed model uses Machine Learning algorithms, specifically the Holt-Winters triple exponential algorithm and a type Recurrent Neural Network (RNN) i.e. Long Short-Term Memory (LSTM). The authors also included a recommendation system which based on predictions and user input generates recommendations on where to invest for a profit. The use of two machine learning algorithms combined, personalized recommendations and the user input regarding investments and risk tolerance gives this model an edge over normal prediction models giving more accurate predictions. However, the accuracy of the predictions depends heavily on the quality and the quantity of the historical data available and the risk of overfitting the models to historical data may result in poor generalization to new or unseen data.

Varun Dogra et al [2] provides a comprehensive overview of the latest techniques and tools used in text classification. It discusses the various algorithms or methods used in subtasks of classification, such as Naive Bayes, Support Vector Machines (SVM), Decision Trees, Random Forest, and Deep Learning algorithms. The authors also present the techniques for data collection from several online sources, including web scraping, APIs, and social media platforms. The paper representation techniques discussed in the paper include basic techniques such as Bag of Words (BoW), Term Frequency-Inverse Document Frequency (TF-IDF). The authors also present recent research in document representation for different areas of machine learning and natural language processing, such as Latent Dirichlet Allocation (LDA), Non-negative Matrix Factorization (NMF) and Convolutional Neural Networks (CNNs). To provide suitable and fast classifiers, the higher dimensional space of data was reduced to a lower space using feature selection and feature extraction methods. The authors discuss different algorithms that perform differently depending on the domain-specific data collections and train machine learning text classifiers. They used these algorithms based on the problem statement, and none of the algorithms has proven perfect for all types of problems and data dimensionality. It also discusses the challenges faced in implementing a text classification system, such as data shortage, difficulty in training deep neural networks, and failure to model contextual long-distance data and maintain sequential order in their representations. Finally, the authors present real-world scenarios where text classification can be applied, such as sentiment analysis, spam detection, and topic modelling.

Peiyuan Lee et al [3] proposes a model that focuses on using machine learning algorithms to predict the dynamic relationship of stock market volatility. The study basically compares the effectiveness of machine learning models with traditional market in forecasting stock trends. The model developed uses Linear Support Vector algorithm. Further, the data is collected and pre-processed followed by eigenvector solution to identify patterns and relationships in data. The data is further normalized to ensure consistency and comparability. Finally, the developed model is compared with traditional forecasting models to assess its accuracy, error rates and overall effectiveness in predicting stock market trends which increases the model efficiency compared to normal analysis models. Again, this model can be affected by overfitting which can lead to inaccurate predictions

Jiake Li [4] proposed a model to approach stock market forecasting using time series analysis, deep learning techniques and a network security to predict stock index changes. By incorporating investor sentiment information and utilizing convolutional neural networks (CNN) for emotional feature extraction, the model aimed to enhance the accuracy of stock market predictions. Then Recurrent Neural Network (RNN) is applied which is suitable for sequential time analysis and can capture suitable dependencies over time followed by Support Vector Machine (SVM) which is used to recognize and observe patterns in the data. By further adding network security elements in the prediction model, it increases the identification of security threats and risk mitigations due to which early detection of market irregularities can be caught. Furthermore, the complexity of integrating network security elements into the predictive model may increase the complexity of the analysis requiring additional expertise, resources to manage and interpret the combined data.

Junhao Zhang and Yefei Lei [5] proposes a model with the concept of Deep Reinforcement Learning (DRL) for stock market prediction. The model implies DRL, which combines deep learning techniques with reinforcement learning principles to make sequential decisions in an environment to maximize rewards. Further incorporating policy gradients in the proposed DRL method for stock prediction which are a class of reinforcement learning algorithms that directly optimize the policy parameters to maximize expected rewards. Deep Q Network (DQN) is utilized as the primary reinforcement learning algorithm. It combines Q-learning with deep neural networks to approximate the Q-value function, enabling the model to learn optimal action-selection policies based on historical stock price data. Deep neural networks like Convolution Neural network (CNN) is used for raw data extraction and then a Feed Forward neural network is used to recognize and capture patterns in data. Finally, a comparison is made between the basic DRL approach and the proposed DRL method based on policy gradients to evaluate their effectiveness in predicting daily stock price changes.

S Anveshrithaa et al [6] proposed the study involves analyzing and predicting vehicle traffic from streams of data from a third-party API that is streamed using Apache Kafka into the analytics model implemented in Spark using Long Short-Term Memory Networks (LSTM) and then storing the original as well as the predicted data in a NoSQL database like MongoDB. The data analytics model uses Apache Kafka to continuously stream data from the source. Kafka is a popular distributed streaming platform which adopts the ‘publish-subscribe’ messaging model to deal with the real-time volume of data. Kafka provides an API that facilitates applications to consume log events in real-time. Kafka maintains message categories called topics to which the producers publish data, and the consumers subscribe to, in order to consume data. Kafka uses Zookeeper for coordination between the Kafka brokers and the consumers. The performance of the system was tested on MNIST image datasets. The study implemented a traffic forecast model that leverages a deep learning model called Long-Short Term Memory network for predicting the flow of traffic on roads in real-time using Apache Spark. With the use of Apache Kafka and Spark streaming, real-time predictive analysis of the traffic data was carried out, and good performance of the model was observed in analyzing the data and predicting the flow of traffic.

Mehar Vijh et al [7] investigates the prediction of stock market movements using machine learning techniques, specifically Artificial Neural Network (ANN) and Random Forest (RF), to enhance forecast accuracy. It utilises financial datasets from Yahoo Finance, incorporating variables like Open, High, Low, and Close prices of stocks. The methodology involves collecting ten years of historical data for five companies, extracting daily closing prices, and creating new variables such as differences in stock prices and various moving averages and standard deviations. Two machine learning models, ANN and RF, are employed, with ANN utilising a three-layer architecture to predict stock closing prices based on the new variables, while RF combines multiple decision trees to minimise forecasting errors. The study compares the effectiveness of ANN and RF models, with ANN demonstrating superior accuracy, as evidenced by lower RMSE and MAPE values across multiple companies. Future research could involve comparing these models with other machine learning and deep learning algorithms to further enhance prediction capabilities.

Mostafa Seif et al [8] [8]proposed modeled model results in a sophisticated stock market real-time recommender system, which uses Hadoop Distributed File System (HDFS) for data storage and Apache Spark framework for efficient data processing delivery Resilient Distributed Datasets (RDD) Streamlined processes execute quickly and ensure predictable results. The method unfolds in three main phases: data acquisition, storage, and analysis. In the data acquisition phase, information is collected from various sources, including social media, news websites, and web scraping techniques. The collected data have been pre-processed to eliminate irrelevant data and to exclude items that are important for the analysis. Hadoop Distributed File System (HDFS) is used for data storage, which is designed to handle large amounts of data on distributed hardware, reducing storage costs. The data analysis phase involves Apache Spark and its scalable machine learning library, MiLB, in three phases: data preprocessing, feature extraction, data classification and model Support vector machine (SVM) and linear regression to forecast stock market trends based on extracted features and so on. using supervised machine learning the results show that the model is accurate and efficient, reaching 87.5% accuracy online and 82.5% in real time. The combination of Hadoop HDFS and Apache Spark greatly encourages the efficiency of the model, making it a powerful tool for real-time portfolio forecasting.

V Kranthi Sai Reddy et al [9] focused on stock market performance forecasting using artificial neural network (ANN) and three different support vector machine (SVM) algorithms The first example used was a single-layer perceptron, it represents the main system. The study dataset covers three months from September 2015 to January 2016, in which nine parameters were included as input to the prediction model. Notably, the NEWS and Twitter data were processed using analytical methods, and this work was facilitated by the Opinion Finder library. The results showed that the radial basis function (RBF) algorithm achieved 63% accuracy in the test set and 61% accuracy in the training set. In contrast, the SVM algorithm showed 100% accuracy in the training set but only 60% accuracy in the testing set. The one-dimensional perceptron model yielded about 60% accuracy, and struggled with best-square predictions in particular. Overall, the study highlighted the usefulness of machine learning for predicting the performance of banks, acknowledging the variation in accuracy depending on the systems used. There are 100 instances in the dataset for four months, of which 70 instances were used for training and the remaining 30 instances for the testing dataset. Nine factors, including oil prices, gold prices, silver prices, FEX, SMA, ARIMA, KIBOR, NEWS, and Twitter served as inputs to the prediction model Also, information mining techniques were important in the processing of NEWS and Twitter data, and helped develop and advance divisions.

Sreelekshmy Selvin [10] aims to leverage deep learning algorithms for stock price prediction, specifically focusing on NSE-listed companies, by employing a sliding window approach with data overlap to create a generalised model capable of utilising minute-wise data, particularly beneficial for high-frequency trading. Utilising a fixed window size of 100 minutes with 90 minutes of overlap, the study trains three deep learning architectures—RNN, LSTM, and CNN—on Infosys' stock price data from July to October 2014 and tests them on Infosys, TCS, and CIPLA data from October to November 2014. Following normalisation of data to a common range, models undergo training for 1000 epochs, with the one yielding the lowest RMSE chosen for prediction. Post-prediction, error percentage is computed using denormalized data to assess performance. CNN emerges as the superior model, demonstrating lower error percentages across Infosys, TCS, and Cipla. It effectively captures trends and dynamics, indicating its adeptness at handling the dynamic nature of stock market data. For future exploration, incorporating different hybrid algorithms could potentially enhance prediction accuracy further.

Mehak Usmani et al[11] proposed three types of artificial neural networks and support vector machine algorithms to predict stock market performance. In the original model, a single layer Perceptron, used nine input parameters from data spanning September 2015 to January 2016, including NEWS and Twitter data determined using text mining using Opinion Finder Results as a radial basis function (RBF) algorithm performed more than 63%, during the testing process. and an accuracy of 61% was achieved in the training group. In contrast, the support vector machine exhibited 100% accuracy in the training set but only 60% accuracy in the test set. The single-layer perceptron achieved about 60% accuracy, struggling with its good-class predictions. In conclusion, the study revealed the feasibility of using machine learning to forecast the stock market, although the accuracy varies depending on the selected algorithms.

Shipra Banik [12] aim of this paper is to forecast Dhaka share market movements using the Rough Set (RS) and Artificial Neural Network (ANN) forecasting models. The methodology involves gathering eight years of daily stock market data from the Dhaka Stock Exchange (DSE), including metrics like opening and closing prices, processing it and creating indicators such as moving averages and relative strength index (RSI) to understand market trends. The dataset is divided into training (70%) and testing (30%) subsets, and two models, RS and ANN are constructed. The RS model uses a specialised toolkit to analyse data and make predictions based on discovered rules, while the ANN model forecasts trends through backpropagation. The hybrid ANN RS model achieves high prediction accuracy of around 97.68%, leveraging the strengths of both approaches. However, the model's complexity and the risk of overfitting pose challenges, necessitating comparison with other machine learning and deep learning algorithms should also be done.

2.2 KEY GAPS IN THE LITERATURE

1. Limited discussion of the impact of external factors, such as economic indicators, geopolitical events and regulatory changes, on the accuracy of prediction of bitcoin in cryptocurrency trading.

2. The lack of discussion about the potential impact of sentiment analysis on the broader cryptocurrency market, such as the potential for increased volatility or the impact on long-term investment strategies.

3. The analysis considers only historical data to determine the closing prices of private equity firms. Future research should explore the use of other data sources, such as news media and social media like twitter, to improve the accuracy of forecasts.

4. The reviewed papers primarily focus on predicting stock market trends within specific timeframes or market conditions. There's limited discussion on the adaptability of models to changing market dynamics or their ability to incorporate real-time data for continuous learning and refinement.

5. The stock market prediction books often lack in-depth analysis of feature importance. Understanding the factors that contribute most to forecast accuracy can increase the explanatory power of models.

6. The text classification literature lacks a comprehensive survey of the latest developments in natural language processing (NLP), such as transformer-based models (e.g., BERT, GPT). Examining the application of these models in text classification tasks could be a valuable addition.

7. The sensitivity analysis literature often lacks a comprehensive analysis of metrics beyond accuracy. Metrics such as precision, recall and F1-score can provide a more nuanced understanding of model performance, especially in crypts.

CHAPTER 3: SYSTEM DEVELOPMENT

3.1 REQUIREMENTS AND ANALYSIS

The first step in the development process is an in-depth analysis. The main goal is to develop a system that could accurately predict the price of the cryptocurrencies based on historical data. This includes identifying critical elements such as data sources, target parameters, and forecast timelines.

FUNCTIONAL REQUIREMENTS

1. Data Ingestion-
    - The collecting, importing and processing raw data for storage and analysis.
    - In addition to this, the system will also integrate sentiment data from sources such as news websites and social media platforms. The sentiment data will provide insights for public sentiments towards bitcoin.
    - Ensure that the data is collected from a reliable resource to maintain integrity.

2. Data Preprocessing-
    - The system takes the dataset and goes to perform data cleaning to handle missing or inconsistent values in the historical data.
    - For preprocessing we use scaling techniques to normalize the historical data.
    - Data preprocessing basically generate data suitable for training and validating different models.
    - For sentiment analysis, the system will preprocess data like in the financial data. Techniques like normalization and tokenization is used for this.

3. Model Implementation-
    - Implementing different deep learning as well as machine learning models. The deep learning model used are LSTM, RNN, CNN, CNN-LSTM, Bi-LSTM, CNN-BiLSTM, GRU while the machine learning model used are RF and SVM.
    - Further implementing the algorithms and training the models using historical data.
    - For sentiment analysis, implementing lexicon-based sentiment analysis to the data collected from the websites.
    - Again, implementing sentiment analysis on our dummy data to get the updated price with positive and negative weightage.

4. Evaluation Metrics-
    - Calculating and analysing different evaluation metrics like Root Mean Squared Error (RMSE), R-squared, adjacent R-squared, Mean Absolute Error (MAE), Mean Absolute Percentage Error (MAPE).
    - Then selecting the best-performing model by analysing different evaluation metrics.

5. Visualization-
    - Using matplotlib we visualized our data in form of the graph.
    - Visualize sentiment trends alongside financial data to gain insights into the relationship between sentiment and price movements.

6. Prediction-
    - Finally, our trained deep learning and machine learning models predict the closing value based on historical data.
    - Compare and analyse the predicted price to the actual price.
    - In case of our models, Bi-LSTM model is the model that has predicted the most accurate value.
    - Use sentiment-aware models to forecast bitcoin prices based on historical data that consists of many sentiment features.
    - Analyze how sentiments affects forecasting results and identify situations in which sentiment driven models outperform traditional models.

NON-FUNCTIONAL REQUIREMENTS

1. Performance-
    - Our system should be able to train various machine as well as deep learning model in timely manners and provide the predictions.
    - The system should be able to handle a large amount of the historical data and generate predictions in the reasonable time frame.

2. Scalability-
    - The system should be scalable to handle an increasing amount of data for training, validation and prediction.
    - If the volume of the data unexpectedly grows, the system should be able to handle the resources of larger datasets without any problem in the model performance.

3. Reliability-
    - The deep learning and machine learning models should be robust and reliable, providing accurate predictions consistently.
    - Users rely more on accurate cryptocurrency price forecasts to know their options. The system should also be able to be more reliable in the face of fluctuations in the market.

4. Usability-
    - The system must have a user-friendly interface that allows interactions with trained models and visualization of forecasts.
    - The system should be easy to navigate and relevant, for users. In addition, simulation tools must be provided to better interpret and analyze forecasts.

5. Maintainability-
    - The code should be well written, modular and easy to maintain and update.
    - Systems may need to be updated as market conditions and user needs to evolve.

6. Portability-
    - The system must be compatible with different platforms and environments to meet deployment needs.
    - The flexibility in deployment options is important because users may need to use the system in different environments.

7. Data Quality-
    - The system should include procedures for handling missing or inconsistent data during data entry.
    - This is important because noisy and imperfect nature of data and it ensures efficient preprocessing and cleaning for accurate model training.

HARDWARE REQUIREMENTS

1. Processor (CPU)-
    - Multi-core processor for handling preprocessing tasks and managing data flow to support overall operations.

2. GPU(s)-
    - High-performance GPUs.

3. RAM –
    - At least 8gb of RAM is required to facilitate smooth training.

4. Storage-
    - Sufficient storage space (preferably SSD) is needed.

5. Networking-

- Good internet speed is needed to train and validate the models.

SOFTWARE REQUIREMENTS

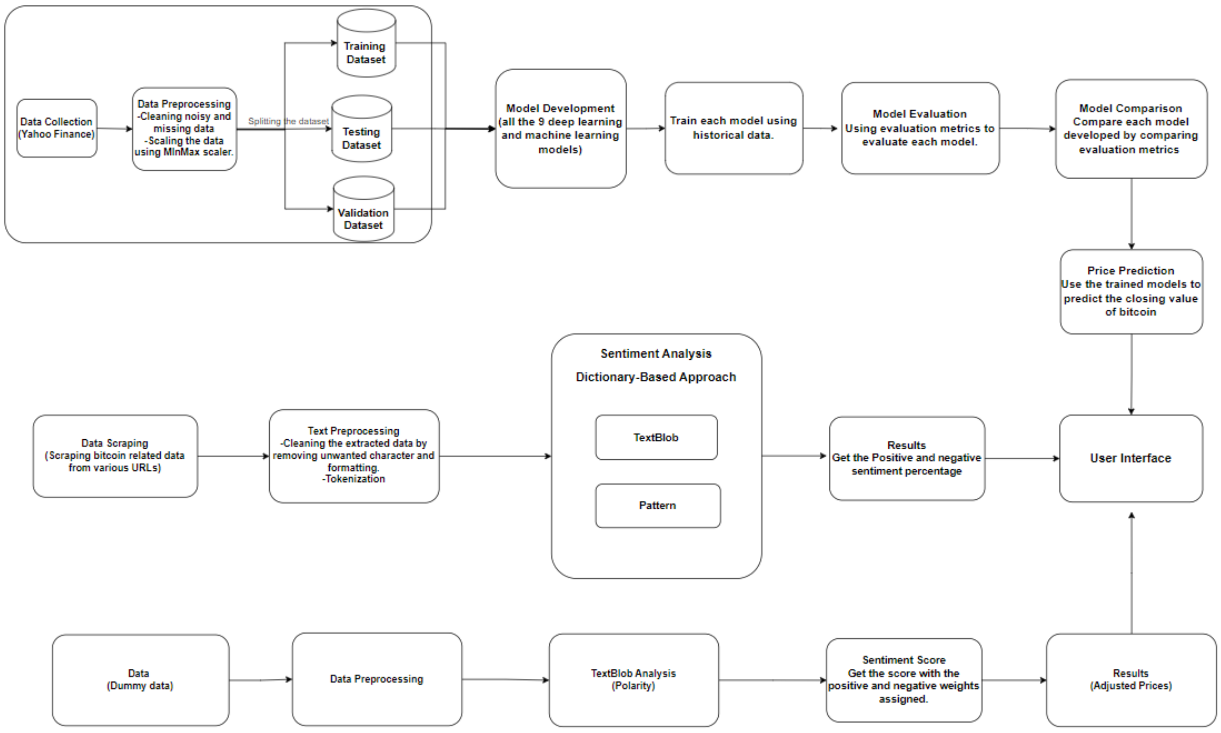
1. Operating System-
    - Linux or windows.
2. Python-
    - Python is used for implementation.
3. Development Environment-
    - An integrated development environment (IDE) like VSCode, Jupyter Notebook or google collab.
4. Version Control-
    - Git for version control.

3.2 PROJECT DESIGN AND ARCHITECTURE

The project design provides an overall view of how our project models for stock price prediction will be developed. Different methodologies are employed in various financial phases. Initially, we performed the collection of data from publicly available datasets on Yahoo Finance. There are various trends such as open, high and low, close, adjacent close, and volume. This dataset is the basic foundation for further development. Then we have implied sentiment analysis based from sources such as news websites and then on a dummy data.

Next, the financial dataset is divided into three sets with preprocessing techniques to improve the data and further normalizing the data using MinMaxScaler. The core phase of this project is training different deep learning and machine learning models in order to get the most accurate closing price. We have selected the best model on the basis of predicted value and evaluation metrics. Then the sentiment analysis helps us figure out people's sentiment towards bitcoin. We have cleaned the data from the sources and implemented tokenization for lexicon-based sentiment analysis. Further, the dummy data we made is to show the application that our model can give adjusted prices. Figure 1 represents project design.

FIGURE 3.1: PROJECT DESIGN



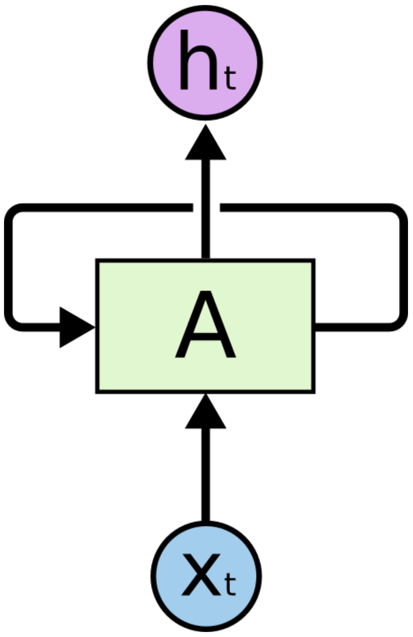
Data Collection (Yahoo Finance)
Data Preprocessing -Cleaning noisy and missing data -Scaling the data using MinMax scaler.
Splitting the dataset
Training Dataset
Testing Dataset
Validation Dataset
Model Development (all the 9 deep learning and machine learning models)
Train each model using historical data.
Model Evaluation Using evaluation metrics to evaluate each model.
Model Comparison Compare each model developed by comparing evaluation metrics
Price Prediction Use the trained models to predict the closing value of bitcoin
Data Scraping (Scraping bitcoin related data from various URLs)
Text Preprocessing -Cleaning the extracted data by removing unwanted character and formatting. -Tokenization
Sentiment Analysis
Dictionary-Based Approach
TextBlob
Pattern
Results Get the Positive and negative sentiment percentage
User Interface
Data (Dummy data)
Data Preprocessing
TextBlob Analysis (Polarity)
Sentiment Score Get the score with the positive and negative weights assigned.
Results (Adjusted Prices)

ARCHITECTURE

RNN Architecture

Recurrent neural network (RNN) is the type of neural network design for processing sequential data. The first layer is in the rnn architecture is rnn layer with 50 units function and returns sequence set to true with input shape based on our training data. The second layer is a dropout layer dropping at least 20% of our data to avoid overfitting . The third layer again is an RNN layer with 50 units with Relu activation functions but with return sequence set as false. The next are two dense layers with 25 and one units respectively.

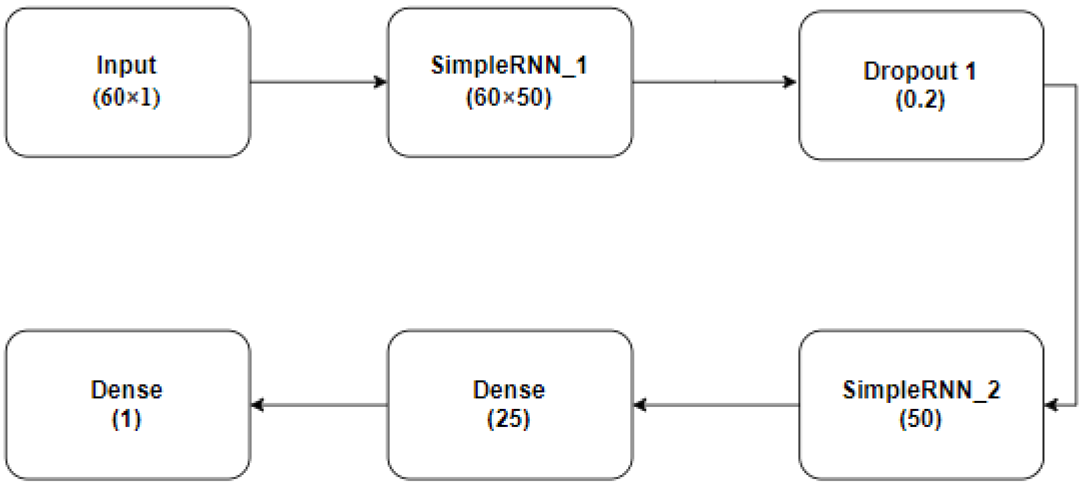
FIGURE 3.2: BASIC ARCHITECTURE OF RNN



For the RNN Model: Table 3.1

Parameter	**Value**
Activation function	ReLU
Number of LSTM layers	2
Number of neurons (1st LSTM)	50
Dropout rate	0.2

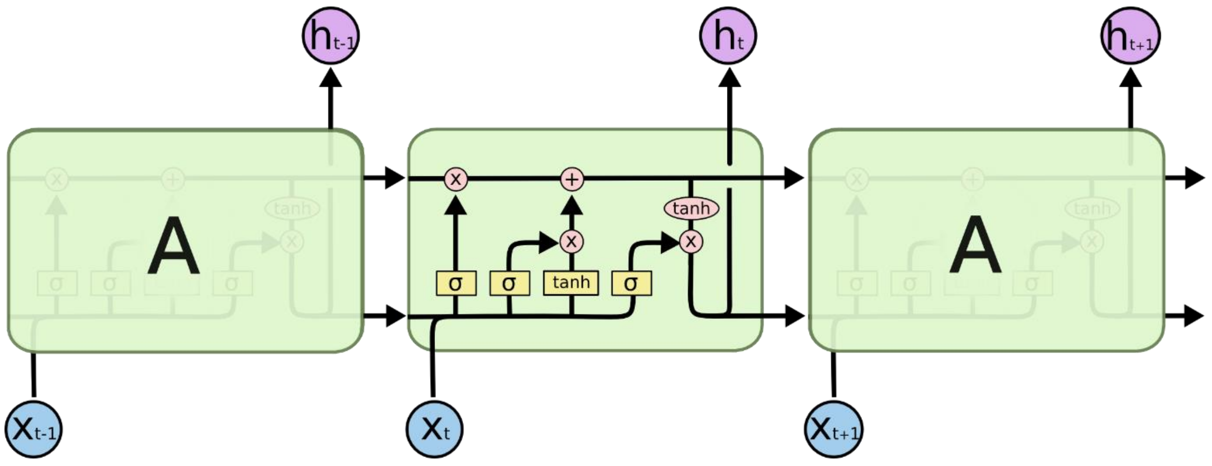
FIGURE 3.3: ARCHITECTURE OF RNN



LSTM Architecture

Long Short-Term Memomry (LSTM) architecture is a type of recurrent neural network made to identify and capture long term dependencies in sequential data. The first layer used is LSTM layer with 50 units, returning sequence for stacking and input shape based on our training data. The next layer is dropout layer with 0.2 which means to prevent overfitting it drops 20% of the data. The next layer is again an lstm layer but with return sequence false since it's the final layer, Followed by two fully connected dense layer with 25 and one units each respectively and then the model is compiled with Adam Optimizer and mse loss.

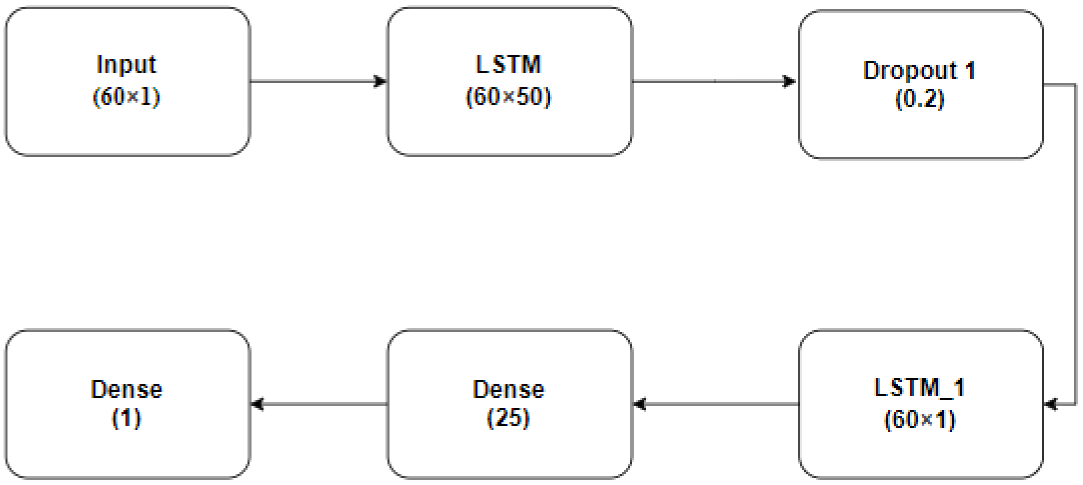
FIGURE 3.4: VISUALIZATION OF LSTM



For the LSTM Model: Table 3.2

Parameter	**Value**
Activation function	ReLU
Number of LSTM layers	2
Number of neurons (1st LSTM)	50
Dropout rate	0.2

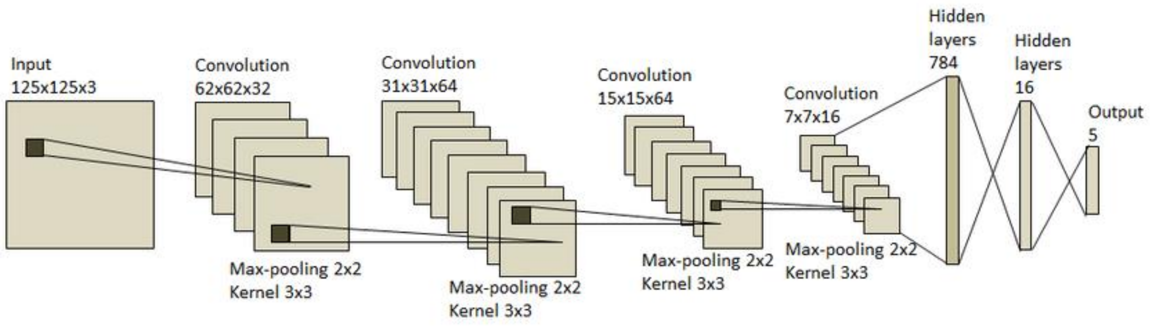
FIGURE 3.5: ARCHITECTURE OF LSTM



CNN Architecture

CNN is a kind of network architecture for deep learning algorithms And is specifically used for image recognition and tasks that involve the processing of pixel data. The first layer is a convolution layer with kernel size three ReLU activactation function and input shape same as the training data. The next layer is Max pooling layer with a size of two in order to control the spatial dimensionality followed by a dropout layer with 0.2 Input that is dropping 20% of data to reduce overfitting. The next layer is flatten layer followed by a dense layer of 50 and 1 units each respectively.

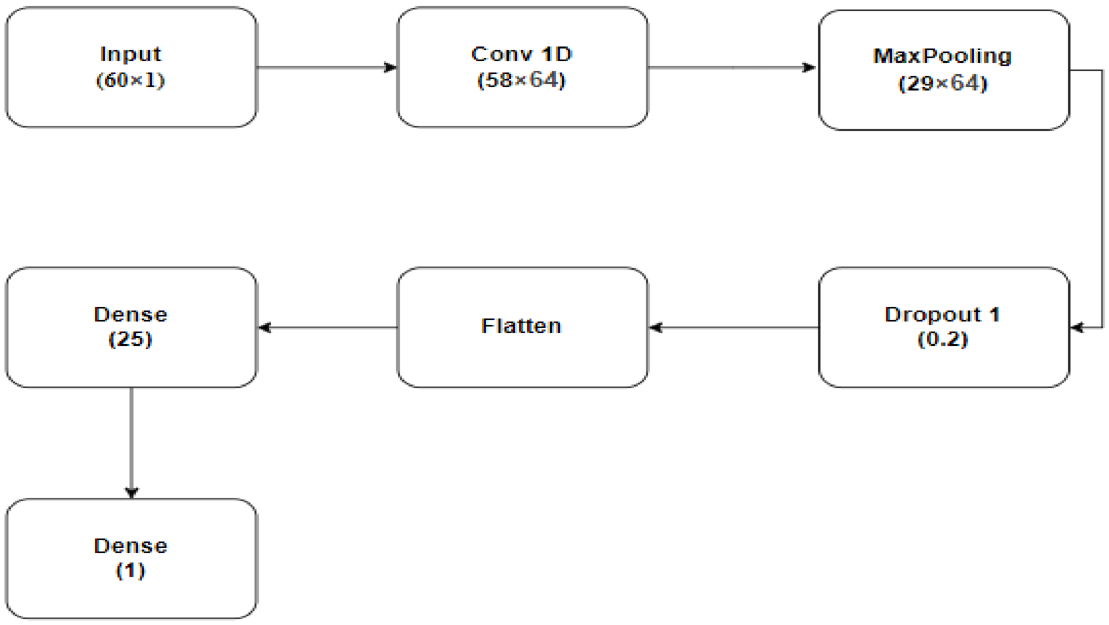
FIGURE 3.6: VISUALIZATION OF CNN



For the CNN Model: Table 3.3

Parameter	**Value**
Activation function	ReLU
Number of CNN layers	1
Number of filters	64
Kernel size	3
Max pooling size	2
Dropout rate	0.2

FIGURE 3.7: ARCHITECTURE OF CNN



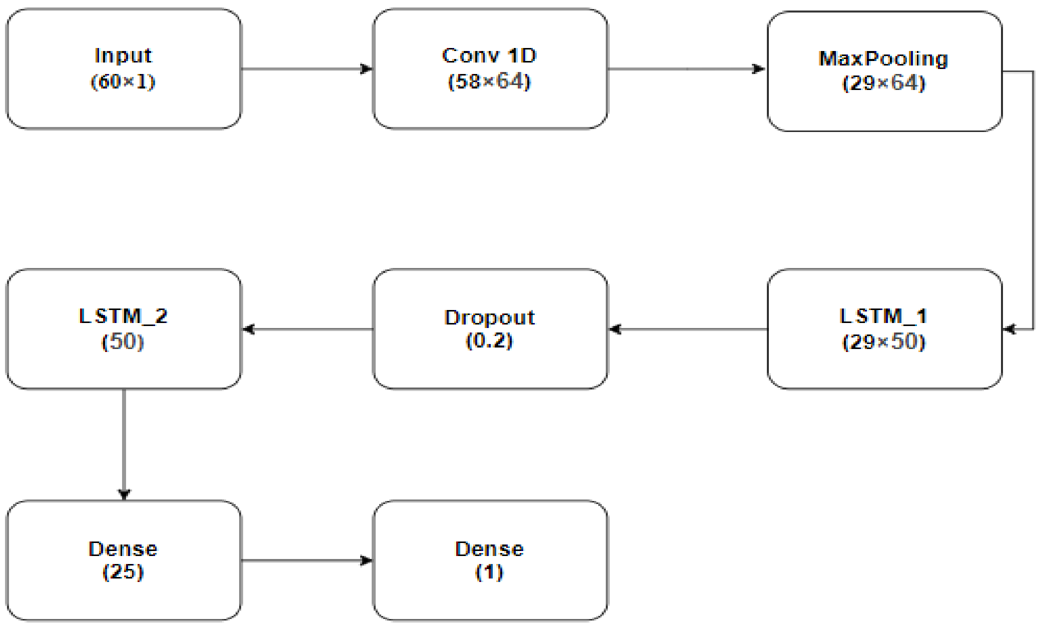
CNN-LSTM Architecture

CNN-LSTM is kind of a hybrid model between CNN and LSTM architecture. The first layer in this is convolution layer with 64 filtres and kernel size of three followed by a ReLU activation function and input shape as the training shape of the data the next layer is Max pooling layer to control the spatial diamensity. The next the whole architecture is same as the LSTM architecture that is two lstm layers followed by two dense layers then compiling the model using atom optimizer and mse loss.

For the CNN-LSTM Model: Table 3.4

Parameter	**Value**
Activation function	ReLU
Number of CNN layers	1
Number of LSTM layers	2
Number of filters	64
Kernel size	3
Max pooling size	2
Dropout rate	0.2

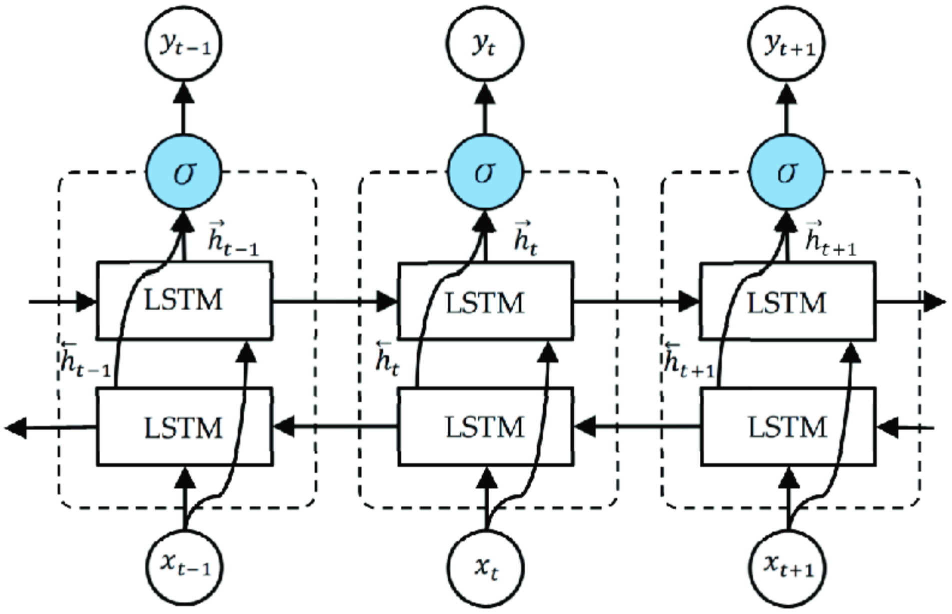
FIGURE 3.8: ARCHITECTURE OF CNN-LSTM



Bi-LSTM Architecture

A Bi-LSTM Architecture is basically an lstm architecture that learns bidirectional long term dependencies to capture long term Time sequence or sequential data. The first layer is a bidirectional LSTM layer with 50 units and a return sequence set as true with an input shape of training data. The next layer is a dropout layer to avoid overfitting. Again the next layer is bidirectional lstm with same 50 units and return sequence set as false followed by two dense layer of 25 and one units and compiling the model using atom optimizer and mse loss function.

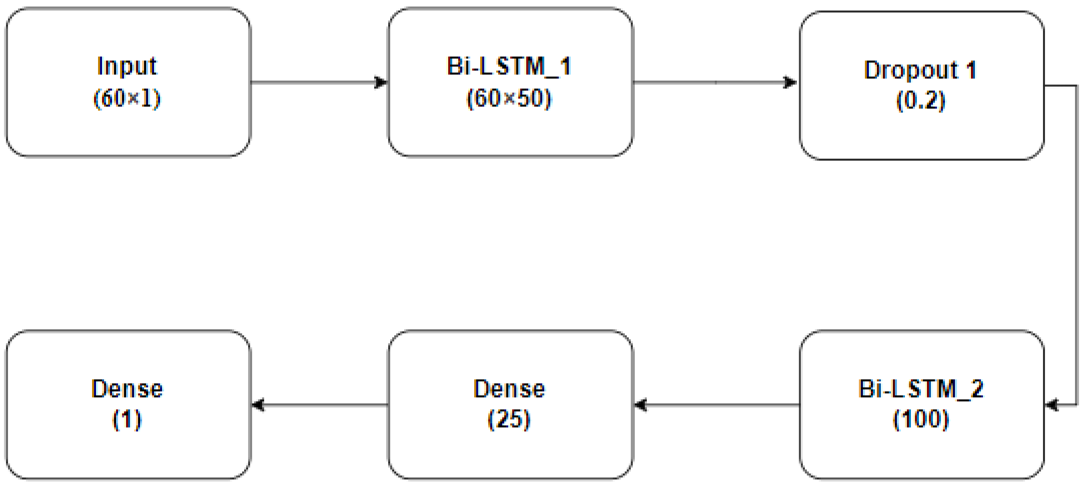
FIGURE 3.9: BASIC ARCHITECTURE OF Bi-LSTM



For the Bi-LSTM Model: Table 3.5

Parameter	**Value**
Activation function	ReLU
Number of Bi-LSTM layers	2
Number of neurons (1st Bi-LSTM)	50
Dropout rate	0.2

FIGURE 3.10: ARCHITECTURE OF Bi-LSTM



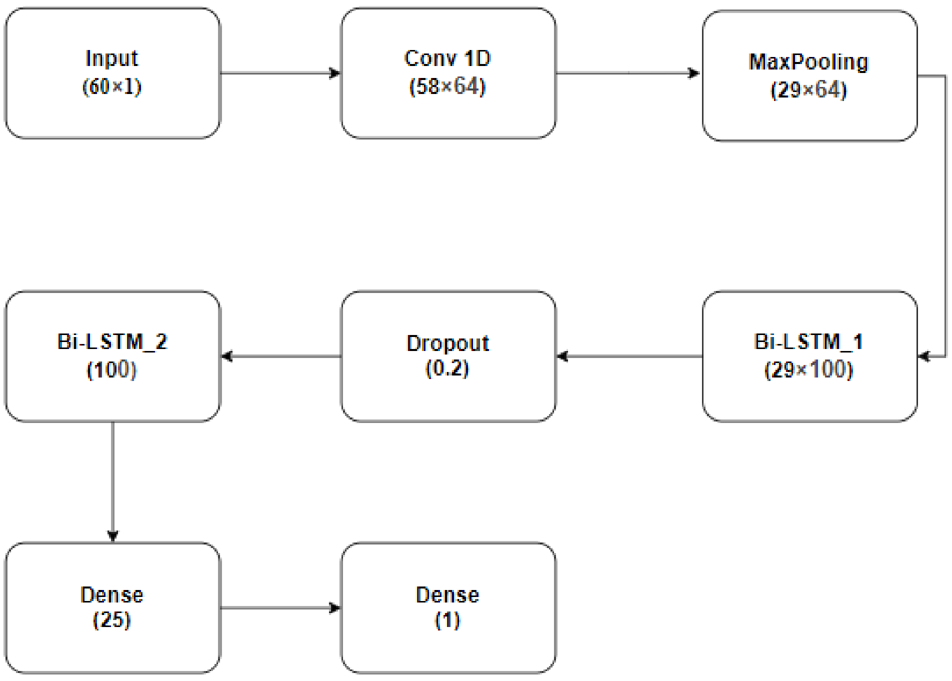
CNN-BiLSTM Architecure

CNN-BiLSTM Architecture is a combination of CNN and Bi-LSTM architecture model. The first layer in this is a convolution layer with 64 filtres, kernel-size 3, Relu function and input shape same size of training shape. The next layer is a max polling layer to reduce the spatial dimensionality.

For the CNN-BiLSTM Model: Table 3.6

Parameter	**Value**
Activation function	ReLU
Number of CNN layers	1
Number of Bi-LSTM layers	2
Number of filters	64
Kernel size	3
Max pooling size	2
Dropout rate	0.2

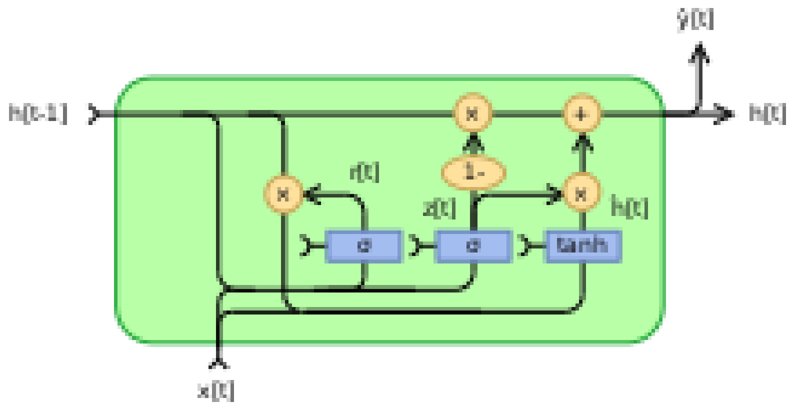
FIGURE 3.11: ARCHITECTURE OF CNN-BiLSTM



GRU Architecture

Gated recurrent unit is a type of recurrent neural network architecture which is a gating mechanism. The GRU is like a long short term memory lstm with a gating mechanism to input of forget certain features resulting in fewer parameters than LSTM. The first layer in this is a GRU layer with 50 inputs written sequence set to true and input shape similar to training shape followed by a dropout layer to drop 20% of our data to reduce overfitting then again a GRU layer with 50 units and return sequence set as false. Then the final two layers are dense layer with 25 and 1 unit respectively followed by compiling the model using atom optimizer and mse loss.

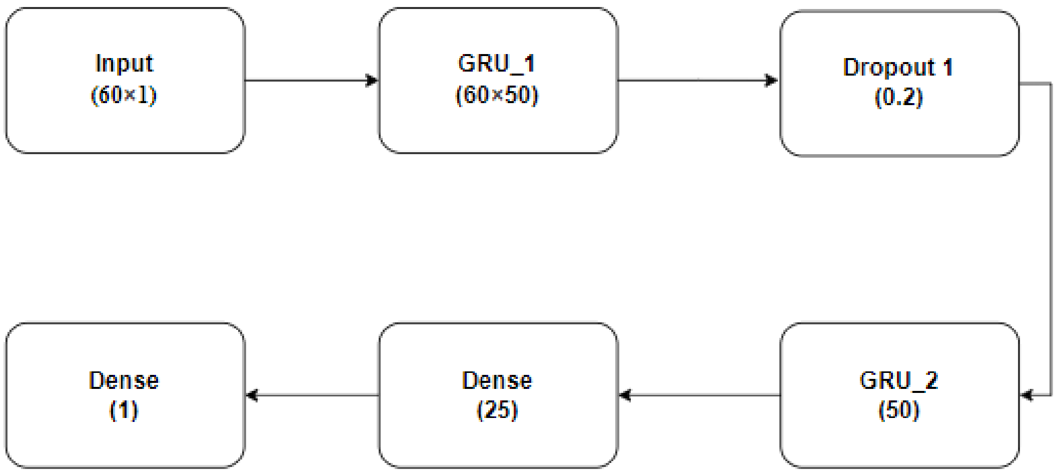
FIGURE 3.12: BASIC ARCHITECTURE OF GRU



For the GRU Model: Table 3.7

Parameter	**Value**
Activation function	ReLU
Number of GRU layers	2
Number of neurons (1st GRU)	50
Dropout rate	0.2

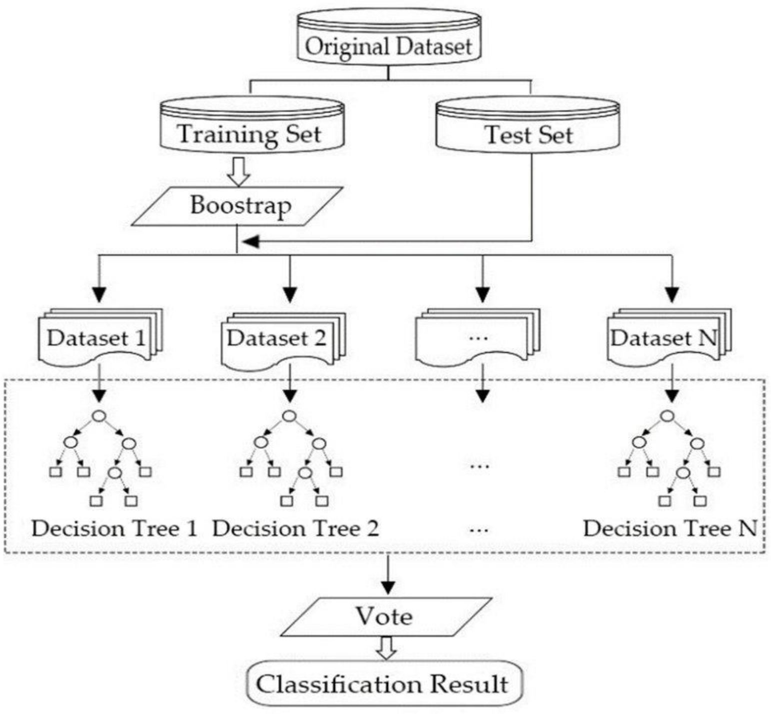
FIGURE 3.13: ARCHITECTURE OF GRU



Random Forest Architecture

A random forest is basically a commonly used machine learning algorithm which takes the output of multiple decision trees and combine them to reach a single result. The model uses random forest regressor class with an estimators to 100 which indicates the number of trees in the forest and random state as 42 which ensures reproductibility the model is trained using fit method with training data.

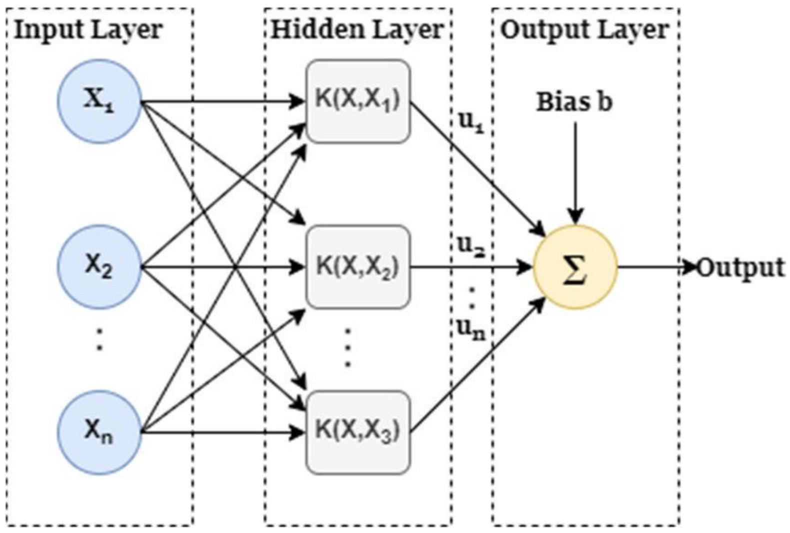
FIGURE 3.14: BASIC ARCHITECTURE OF RF



SVM Architecture

Support vector machine (SVM) is a type of supervised learning algorithm . The model uses SVR classes with a radial basis function RBF Kernel the model is trained using fit method with training data and prediction I made using predict method.

FIGURE 3.15: BASIC ARCHITECTURE OF SVM



Variations of RNNs

Recurrent neural networks (RNNs) are great for handling sequences of data, like text or audio. They're designed to remember what came before, which helps them understand context and relationships between different parts of the sequence. At each time step, they take an input and update their internal state based on the current input and the previous state. This allows RNNs to capture temporal dependencies in the data. However, traditional RNNs struggle to capture long-term dependencies due to the vanishing gradient problem.

Now, there are a couple of variations of RNNs out there, including long short-term memory (LSTM) networks and gated recurrent units (GRUs). Both LSTMs and GRUs are designed to address the problem of "vanishing gradients" in RNNs, which occurs when the gradients of the weights in the network become very small and the network has difficulty learning.

LSTMs are a type of RNN designed to address the vanishing gradient problem and better capture long-term dependencies. They introduce specialized memory cells and gating mechanisms (such as forget, input, and output gates) that regulate the flow of information within the network. These cells use gates to control the flow of information in and out, deciding what to remember and what to forget based on what's happening in the sequence.

Now, for bidirectional LSTM (Bi-LSTM). It's like having LSTMs but looking both forward and backward in the sequence. This means it can capture information from both past and future contexts, providing a better understanding of the data and potentially improving performance.

GRUs, on the other hand, are a simplified version of LSTMs. They use a single "update gate" to control the flow of information into the memory cell, rather than the three gates used in LSTMs. This makes GRUs easier to train and faster to run than LSTMs, but they may not be as effective at storing and accessing long-term dependencies.

CNN-BiLSTM models combine the strengths of CNNs and Bi-LSTMs for sequential data processing. CNNs are excellent at capturing spatial patterns in data, while Bi-LSTMs excel at capturing temporal dependencies. In CNN-BiLSTM architectures, CNN layers are typically used for feature extraction, capturing spatial patterns in the data, while Bi-LSTM layers process the output of CNNs, capturing temporal dependencies and contextual information. This integration allows CNN-BiLSTM models to learn hierarchical features in the data through convolutional layers and capture long-range dependencies through bidirectional recurrent layers, making them particularly effective for tasks such as video analysis, action recognition, and time series forecasting.

TABLE 3.8: COMPARISON BETWEEN RNN,LSTM,Bi-LSTM & GRU

Parameters	RNN	LSTM	Bi-LSTM	GRU
Structure	Simple	More complex	More complex than LSTM	Simpler than LSTM
Training	Can be difficult	Can be more difficult	Similar to LSTMs	Easier than LSTM
Performance	Good for simple tasks	Good for complex tasks	Good for tasks requiring context from both directions	Can be intermediate between simple and complex tasks
Hidden state	Single	Multiple (memory cell)	Multiple	Single
Gates	None	Input, output, forget	Input, output, forget	Update, reset
Long-term dependencies	Limited	Strong	Strong in capturing bidirectional dependencies	Intermediate between RNNs and LSTMs

Sentiment Analysis

Now, we have worked on sentiment analysis for two different parts. Firstly, we have implemented sentiment analysis on data collected through web scraping. Secondly, to reflect the application of our model we created a dummy dataset of twitter so that we can show the changes in the predicted prices.

For the first part, we have taken various news articles and bitcoin related websites and extracted html content. Further, the text is taken from the HTML content and preprocessed.

After the text is cleaned and set in a definite format, sentiment analysis is performed on it. We have used Dictionary-based approach for sentiment analysis. In this, we have used TextBlob and Pattern analysis libraries to get Subjectivity and Polarity. Now, subjectivity determines whether a statement is factual or based on personal opinions/feelings. It is represented in the numerical value of 0 to 1. Now the more the value is towards 0 or 0, it is a factual statement and the more the value is towards 1 or 1, it is a opinion. For polarity, it determines the positive and negative sentiments of the statement. It is also represented in numerical value i.e. -1 to +1. The more the value leans towards -1, the statement is of negative sentiment and the more the value leans towards +1, the statement is of positive sentiment. Now, polarity will help us in finding the negative and positive sentiments of the data collected through web scraping.

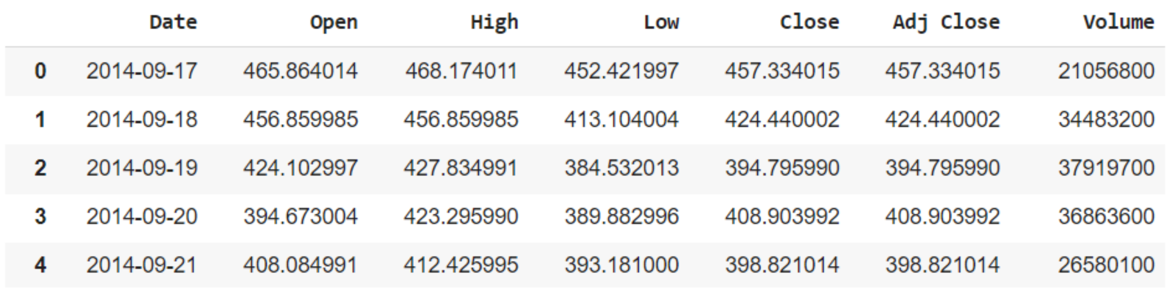
Now, for the second part of sentiment analysis. We have created a dummy dataset. The dataset consists of 5 columns which are date, tweet_id, username, tweets and followers. We have positive and negative sentiments both towards bitcoin in the data and we have filtered our data so as to only accounts with more than or equal to a million followers are used in the analysis. We used textblob analysis to get polarity of the filtered tweets. Then the positive and negative sentiments are obtained. Now based on this sentiment i.e. whether it is positive or negative, we assign weights to it. What weights does is it balances the variables to influence the positive or negative sentiments when calculating a sentiment score. After sentiment score is calculated, we get our adjusted price by taking 1% of the sentiment score of the predicted price and then if overall positive sentiment is more than by adding this price to original price or if overall price is negative sentiment, then subtract this part from the predicted price.

3.3 DATA PREPRATION

DATA COLLECTION

Data is collected from the website as historical data and then made sure that the dataset is from errors, noise and missing values. The dataset consists of many variables such as open, high and low, close, adjacent close and volume. The figure 9 is a small representation of our dataset.

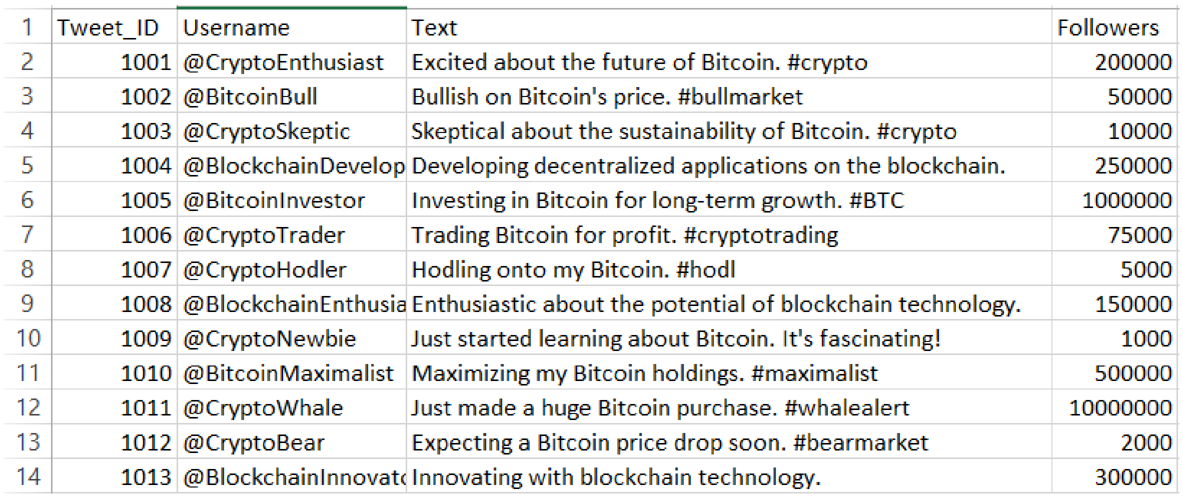
FIGURE 3.16: FINANCIAL DATASET



	Date	Open	High	Low	Close	Adj Close	Volume
0	2014-09-17	465.864014	468.174011	452.421997	457.334015	457.334015	21056800
1	2014-09-18	456.859985	456.859985	413.104004	424.440002	424.440002	34483200
2	2014-09-19	424.102997	427.834991	384.532013	394.795990	394.795990	37919700
3	2014-09-20	394.673004	423.295990	389.882996	408.903992	408.903992	36863600
4	2014-09-21	408.084991	412.425995	393.181000	398.821014	398.821014	26580100

For sentiment analysis, firstly the data is collected from different sources like news and social media websites. Then to show the application of our project, we created a dummy data of twitter with tweets_id, username, tweets and followers.

FIGURE 3.17: TWEETS DATASET



1	Tweet_ID	Username	Text	Followers
2	1001	@CryptoEnthusiast	Excited about the future of Bitcoin. #crypto	200000
3	1002	@BitcoinBull	Bullish on Bitcoin's price. #bullmarket	50000
4	1003	@CryptoSkeptic	Skeptical about the sustainability of Bitcoin. #crypto	10000
5	1004	@BlockchainDevelop	Developing decentralized applications on the blockchain.	250000
6	1005	@BitcoinInvestor	Investing in Bitcoin for long-term growth. #BTC	1000000
7	1006	@CryptoTrader	Trading Bitcoin for profit. #cryptotrading	75000
8	1007	@CryptoHodler	Hodling onto my Bitcoin. #hodl	5000
9	1008	@BlockchainEnthusia	Enthusiastic about the potential of blockchain technology.	150000
10	1009	@CryptoNewbie	Just started learning about Bitcoin. It's fascinating!	1000
11	1010	@BitcoinMaximalist	Maximizing my Bitcoin holdings. #maximalist	500000
12	1011	@CryptoWhale	Just made a huge Bitcoin purchase. #whalealert	10000000
13	1012	@CryptoBear	Expecting a Bitcoin price drop soon. #bearmarket	2000
14	1013	@BlockchainInnovat	Innovating with blockchain technology.	300000

DATA PREPROCESSING

The system takes the dataset and performs data cleaning to handle missing or inconsistent values in the historical data and the dummy data. Then it identifies and removes duplicate data in order to avoid bias. Then for the financial dataset, it uses scaling techniques to normalize the historical data and for the sentiment dataset, it uses preprocessing and tokenization for the dataset.

DATA ORGANIZATION

In the financial dataset, we structure the data for efficient model training and evaluation. Then further splitting the dataset into training and testing sets to asses model generalization. Normalize the entire dataset if needed, ensuring consistency between training and testing datasets. The table shows training and validation data.

TABLE 3.9

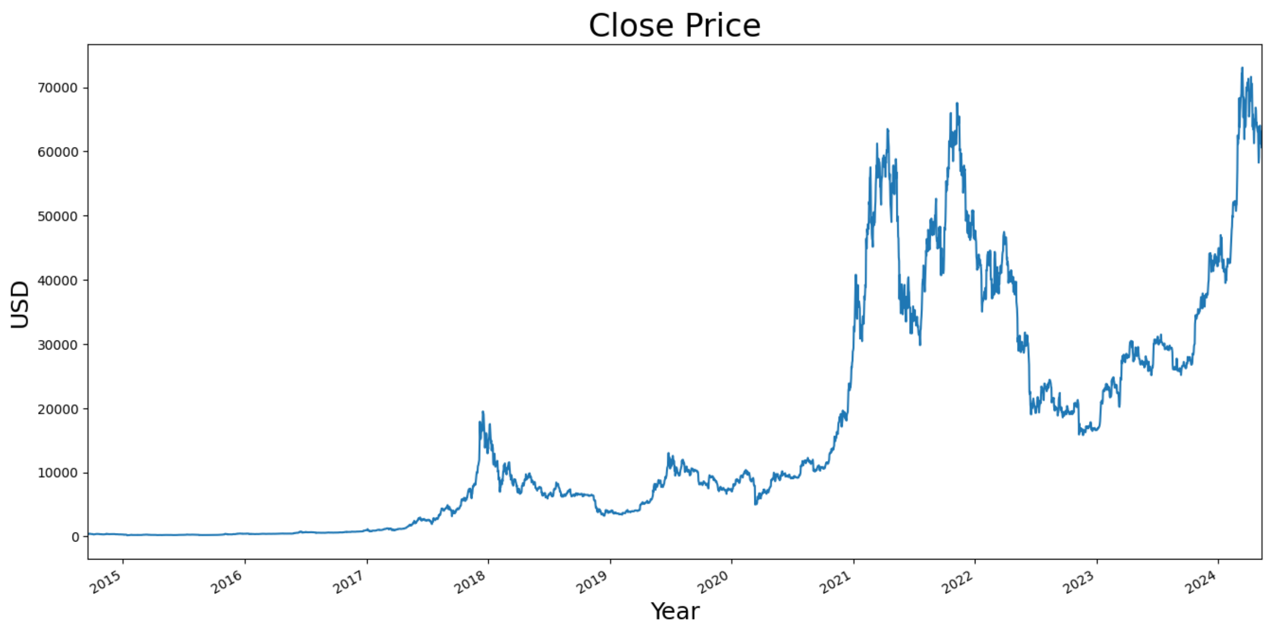
Training Set	**2292**
Validation Set	1067
Testing Set	230

In the sentiment dataset, we join all the data in one sentence to clean the sentence and further using Natural Language Processing (NLP) and spacy library to form sentences to further perform sentiment analysis.

DATA VISUALIZATION

Finally, the target value i.e., is closing price in represented in the graph using matplotlib library. Figure 10 represents the graph of closing price.

FIGURE 3.18: VISUALIZATION

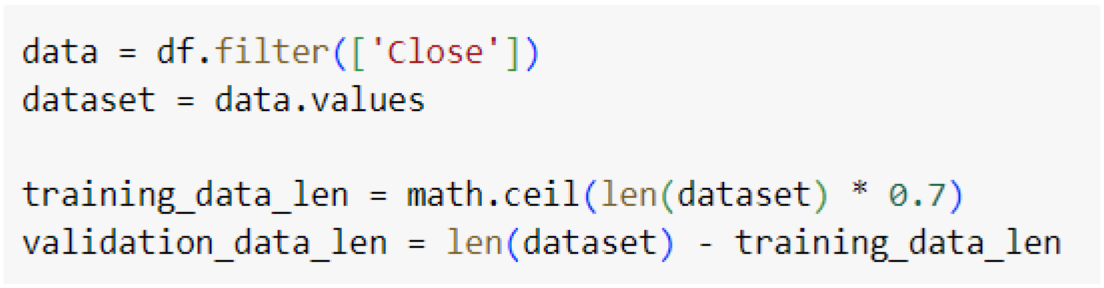


Finally, the split data for testing and validation is given to the models in order to predict the Bitcoin closing price.

3.4 IMPLEMENTATION

Division of the training and validation split

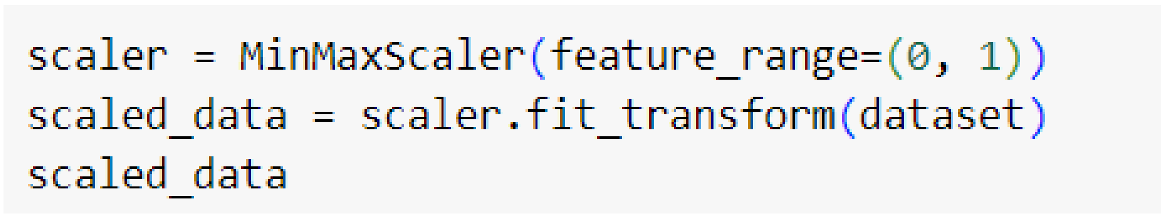
FIGURE 3.19: DIVISON OF DATA



```
data = df.filter(['Close'])
dataset = data.values

training_data_len = math.ceil(len(dataset) * 0.7)
validation_data_len = len(dataset) - training_data_len
```

Normalizing the data

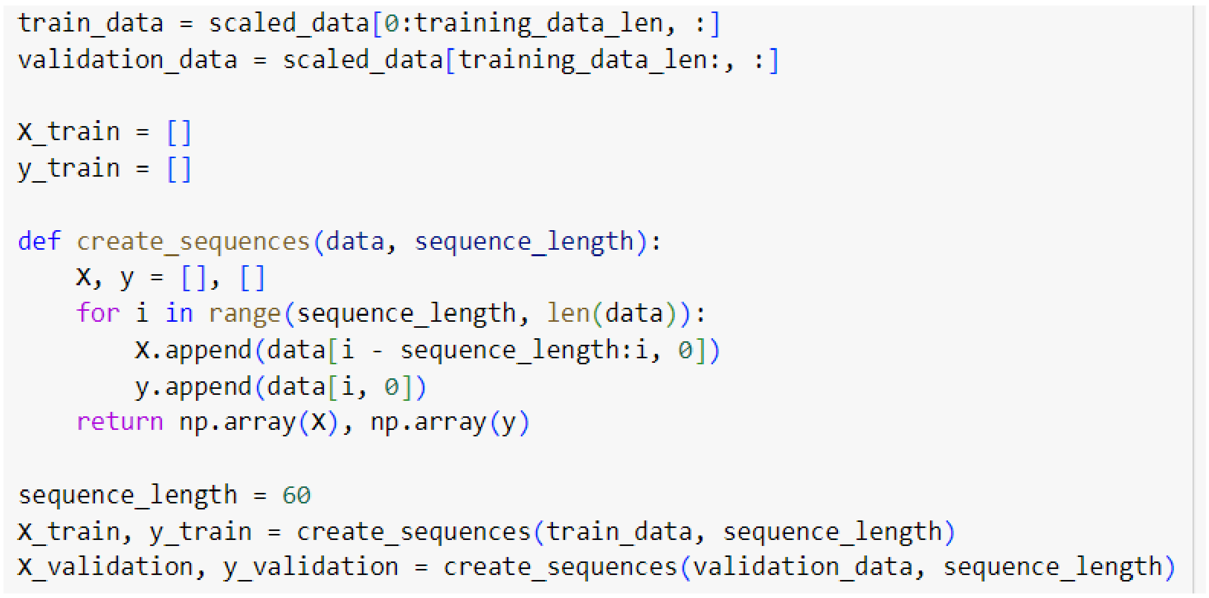
FIGURE 3.20: NORMALIZATION OF DATA



```
scaler = MinMaxScaler(feature_range=(0, 1))
scaled_data = scaler.fit_transform(dataset)
scaled_data
```

Sequence Function

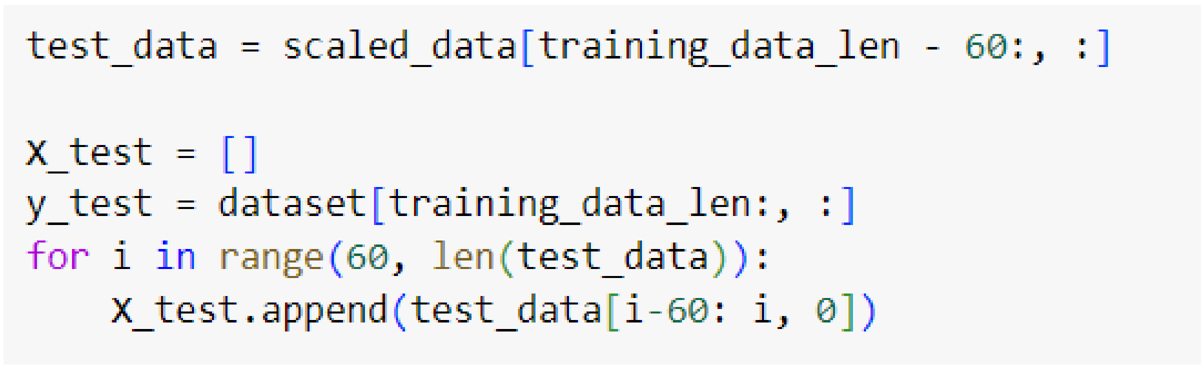
FIGURE 3.21: SEQUENCE FUNCTION



```
train_data = scaled_data[0:training_data_len, :]
validation_data = scaled_data[training_data_len:, :]

X_train = []
y_train = []

def create_sequences(data, sequence_length):
    X, y = [], []
    for i in range(sequence_length, len(data)):
        X.append(data[i - sequence_length:i, 0])
        y.append(data[i, 0])
    return np.array(X), np.array(y)

sequence_length = 60
X_train, y_train = create_sequences(train_data, sequence_length)
X_validation, y_validation = create_sequences(validation_data, sequence_length)
```

Training data

FIGURE 3.22: TRAINING DATA

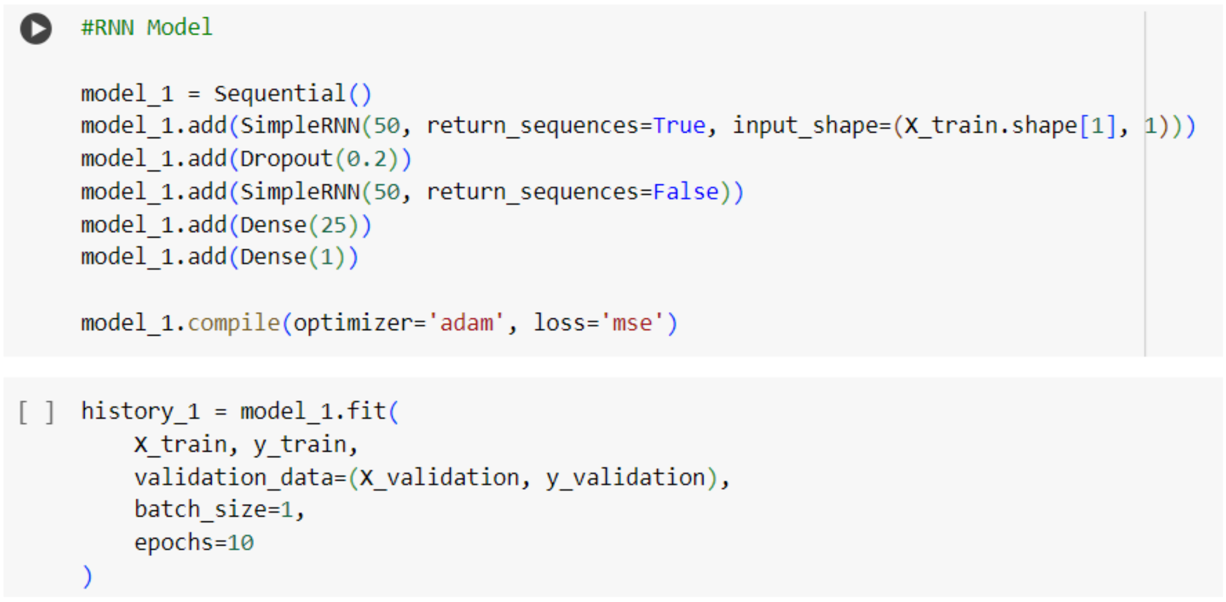


```
test_data = scaled_data[training_data_len - 60:, :]

X_test = []
y_test = dataset[training_data_len:, :]
for i in range(60, len(test_data)):
    X_test.append(test_data[i-60: i, 0])
```

Following are the model implemented in our study

1. RNN model

FIGURE 3.23

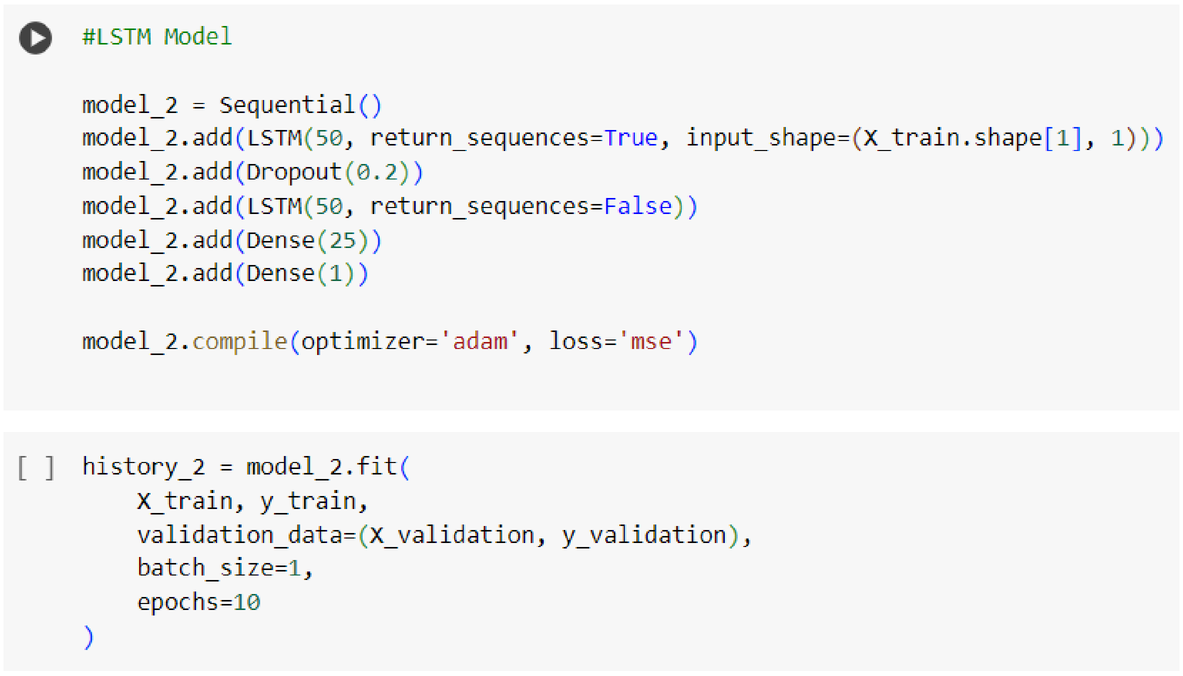


```
#RNN Model

model_1 = Sequential()
model_1.add(SimpleRNN(50, return_sequences=True, input_shape=(X_train.shape[1], 1)))
model_1.add(Dropout(0.2))
model_1.add(SimpleRNN(50, return_sequences=False))
model_1.add(Dense(25))
model_1.add(Dense(1))

model_1.compile(optimizer='adam', loss='mse')
```

```
[ ]  history_1 = model_1.fit(
         X_train, y_train,
         validation_data=(X_validation, y_validation),
         batch_size=1,
         epochs=10
     )
```

2. LSTM model

FIGURE 3.24

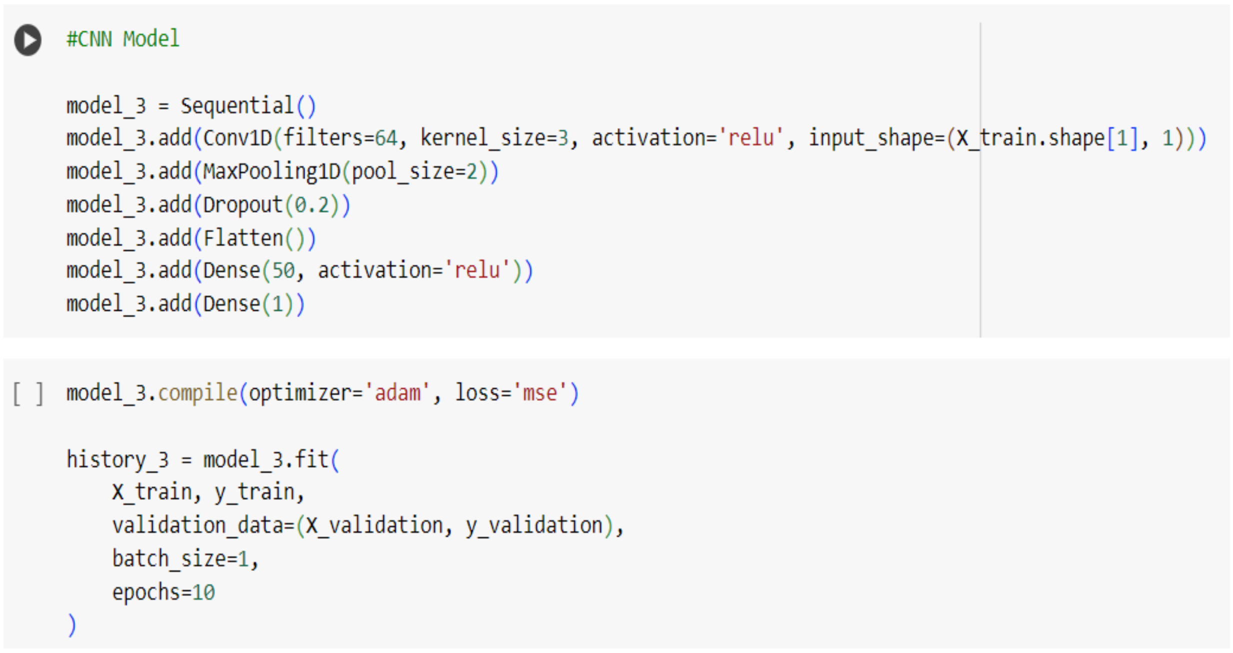


```
#LSTM Model

model_2 = Sequential()
model_2.add(LSTM(50, return_sequences=True, input_shape=(X_train.shape[1], 1)))
model_2.add(Dropout(0.2))
model_2.add(LSTM(50, return_sequences=False))
model_2.add(Dense(25))
model_2.add(Dense(1))

model_2.compile(optimizer='adam', loss='mse')
```

```
[ ]  history_2 = model_2.fit(
         X_train, y_train,
         validation_data=(X_validation, y_validation),
         batch_size=1,
         epochs=10
     )
```

3. CNN model

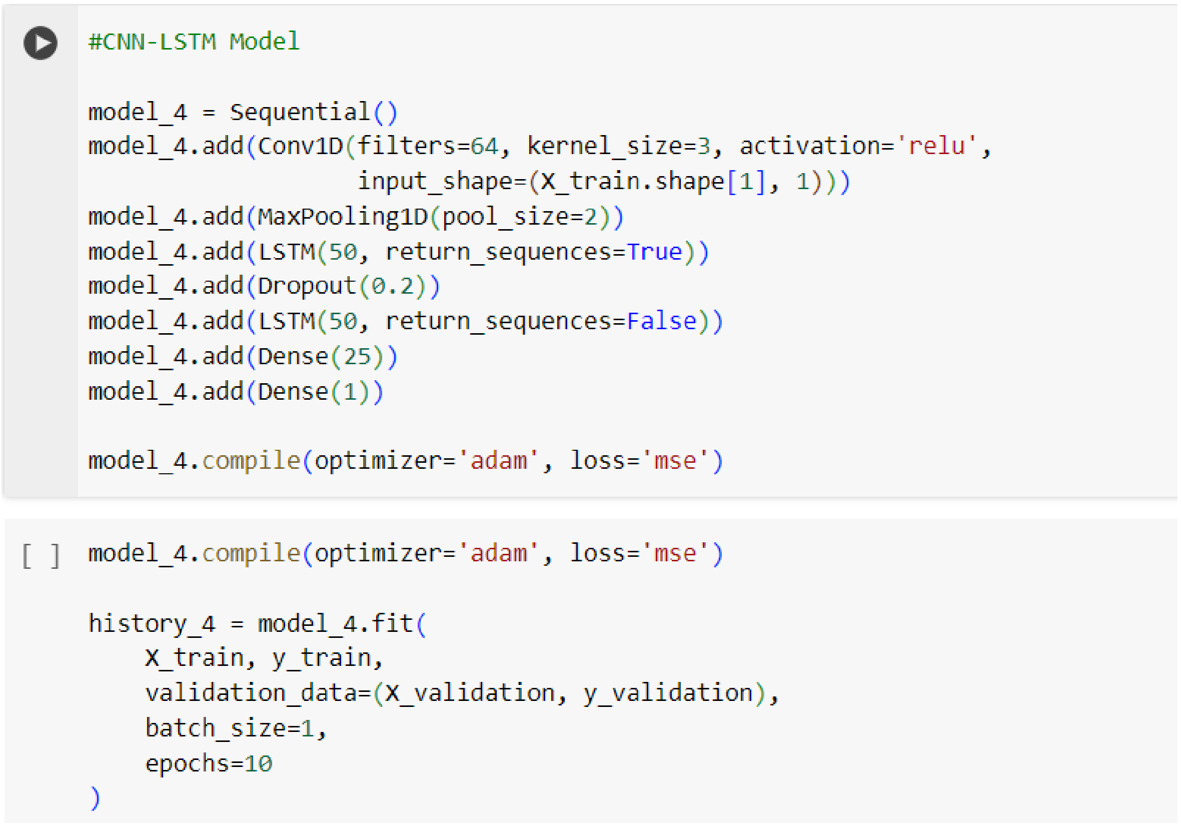
FIGURE 3.25


```
#CNN Model

model_3 = Sequential()
model_3.add(Conv1D(filters=64, kernel_size=3, activation='relu', input_shape=(X_train.shape[1], 1)))
model_3.add(MaxPooling1D(pool_size=2))
model_3.add(Dropout(0.2))
model_3.add(Flatten())
model_3.add(Dense(50, activation='relu'))
model_3.add(Dense(1))
```

```
[ ] model_3.compile(optimizer='adam', loss='mse')

    history_3 = model_3.fit(
        X_train, y_train,
        validation_data=(X_validation, y_validation),
        batch_size=1,
        epochs=10
    )
```

4. CNN-LSTM model

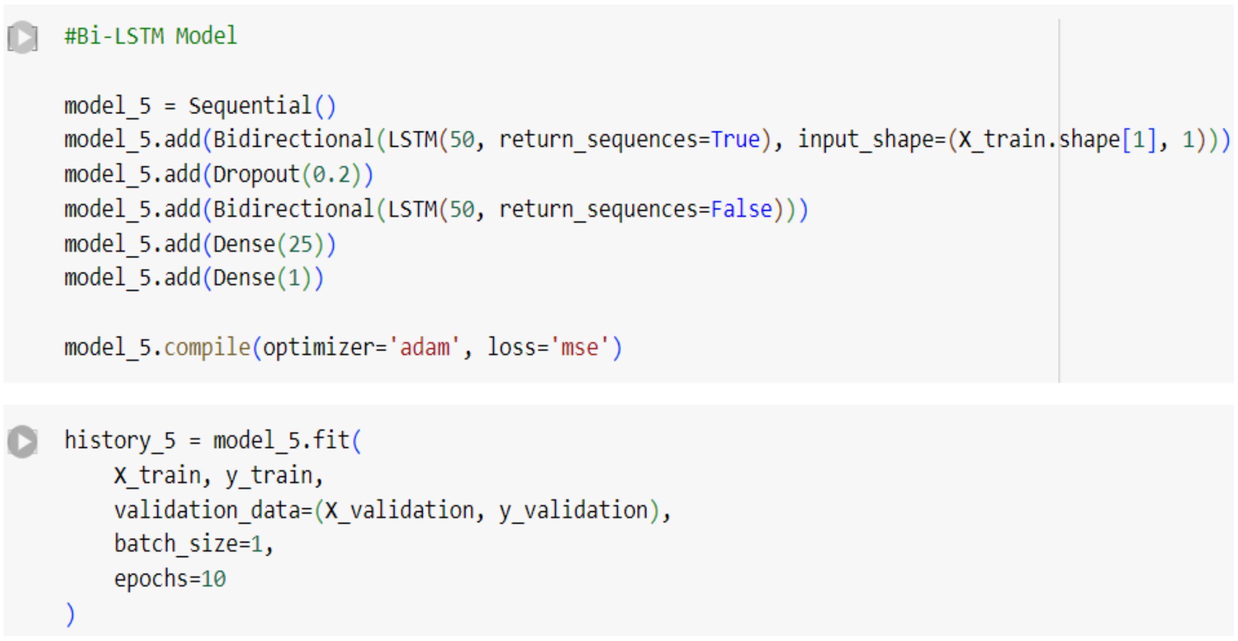
FIGURE 3.26


```
#CNN-LSTM Model

model_4 = Sequential()
model_4.add(Conv1D(filters=64, kernel_size=3, activation='relu',
                   input_shape=(X_train.shape[1], 1)))
model_4.add(MaxPooling1D(pool_size=2))
model_4.add(LSTM(50, return_sequences=True))
model_4.add(Dropout(0.2))
model_4.add(LSTM(50, return_sequences=False))
model_4.add(Dense(25))
model_4.add(Dense(1))

model_4.compile(optimizer='adam', loss='mse')
```

```
[ ] model_4.compile(optimizer='adam', loss='mse')

    history_4 = model_4.fit(
        X_train, y_train,
        validation_data=(X_validation, y_validation),
        batch_size=1,
        epochs=10
    )
```

5. Bi-LSTM model

FIGURE 3.27

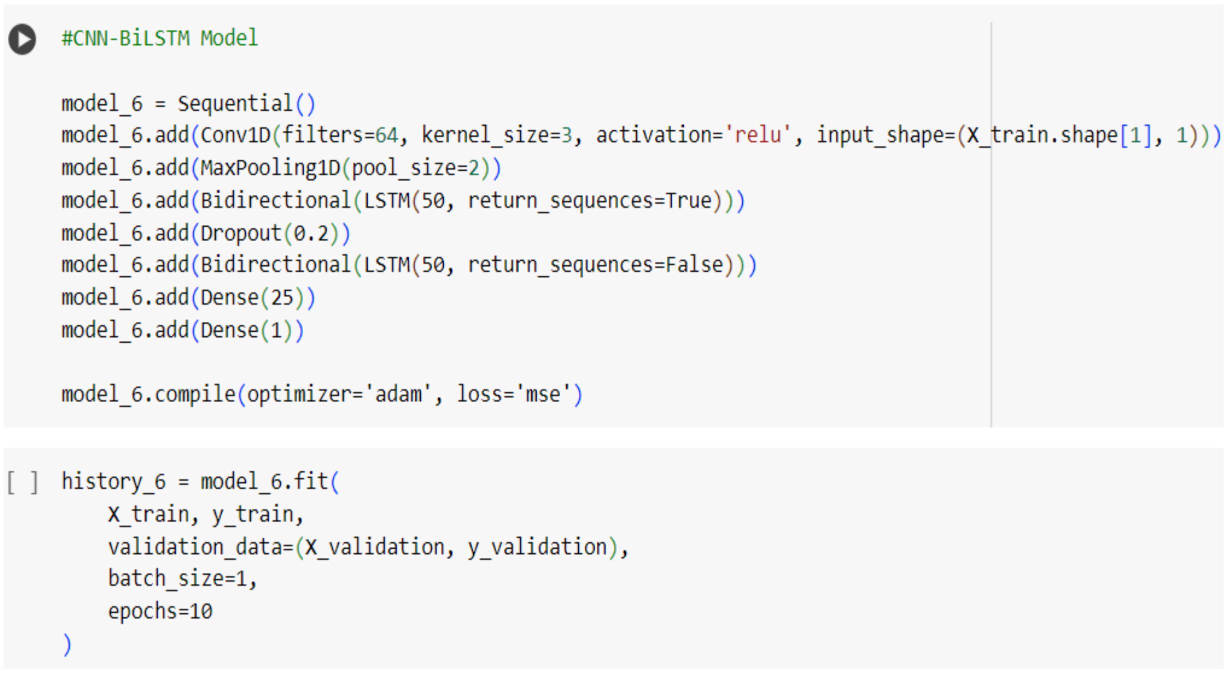


```
#Bi-LSTM Model

model_5 = Sequential()
model_5.add(Bidirectional(LSTM(50, return_sequences=True), input_shape=(X_train.shape[1], 1)))
model_5.add(Dropout(0.2))
model_5.add(Bidirectional(LSTM(50, return_sequences=False)))
model_5.add(Dense(25))
model_5.add(Dense(1))

model_5.compile(optimizer='adam', loss='mse')
```

```
history_5 = model_5.fit(
    X_train, y_train,
    validation_data=(X_validation, y_validation),
    batch_size=1,
    epochs=10
)
```

6. CNN-BiLSTM model

FIGURE 3.28

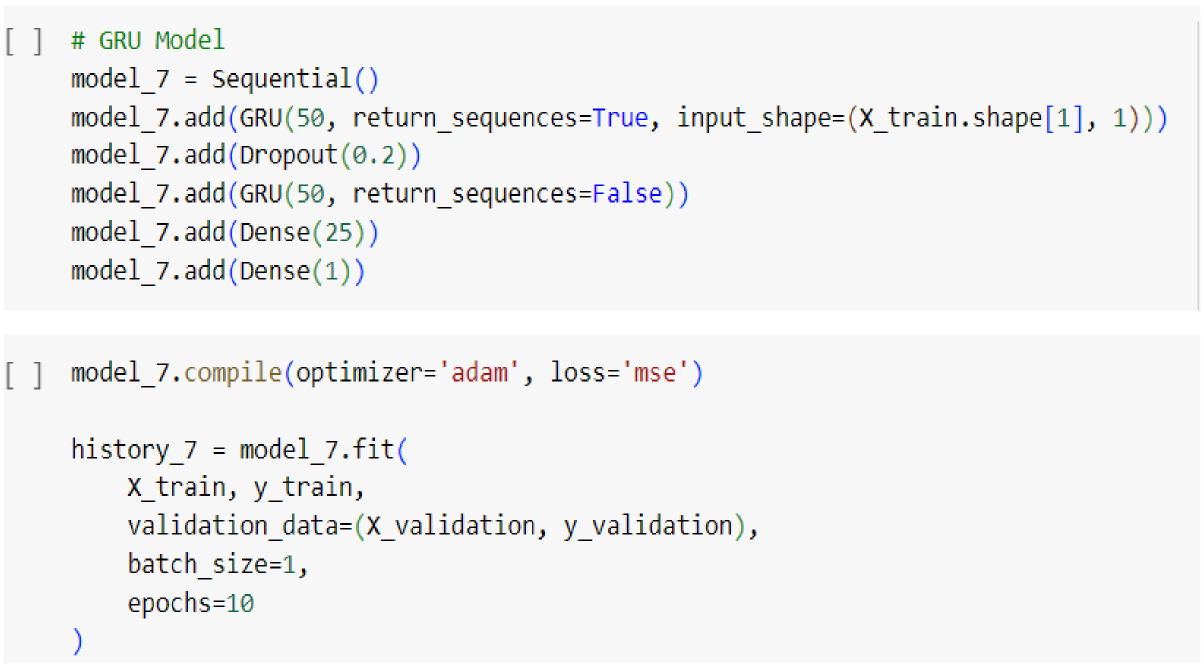


```
#CNN-BiLSTM Model

model_6 = Sequential()
model_6.add(Conv1D(filters=64, kernel_size=3, activation='relu', input_shape=(X_train.shape[1], 1)))
model_6.add(MaxPooling1D(pool_size=2))
model_6.add(Bidirectional(LSTM(50, return_sequences=True)))
model_6.add(Dropout(0.2))
model_6.add(Bidirectional(LSTM(50, return_sequences=False)))
model_6.add(Dense(25))
model_6.add(Dense(1))

model_6.compile(optimizer='adam', loss='mse')
```

```
[ ] history_6 = model_6.fit(
        X_train, y_train,
        validation_data=(X_validation, y_validation),
        batch_size=1,
        epochs=10
    )
```

7. GRU model

FIGURE 3.29

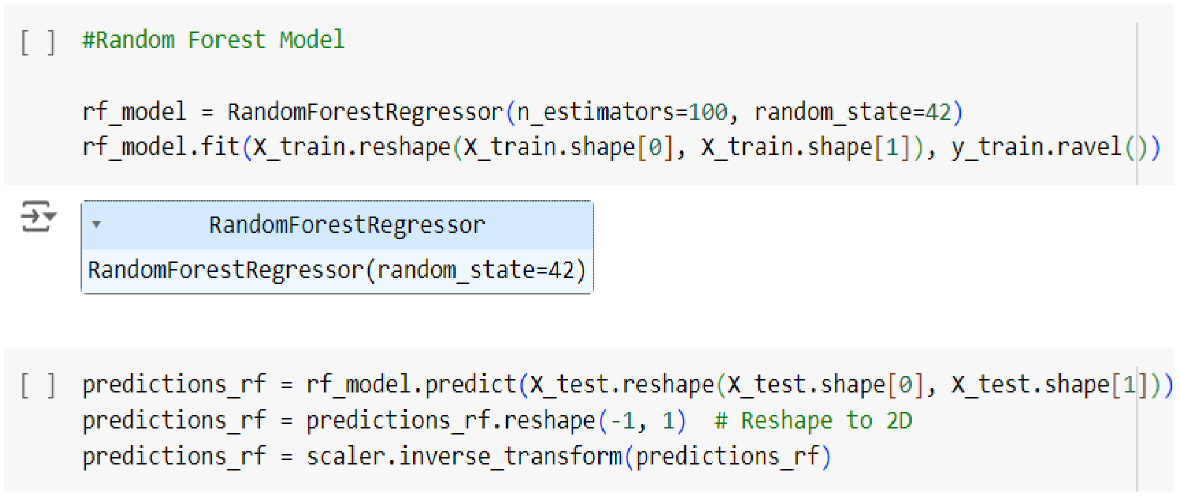


```
[ ] # GRU Model
    model_7 = Sequential()
    model_7.add(GRU(50, return_sequences=True, input_shape=(X_train.shape[1], 1)))
    model_7.add(Dropout(0.2))
    model_7.add(GRU(50, return_sequences=False))
    model_7.add(Dense(25))
    model_7.add(Dense(1))
```

```
[ ] model_7.compile(optimizer='adam', loss='mse')

    history_7 = model_7.fit(
        X_train, y_train,
        validation_data=(X_validation, y_validation),
        batch_size=1,
        epochs=10
    )
```

8. Random Forest model

FIGURE 3.30



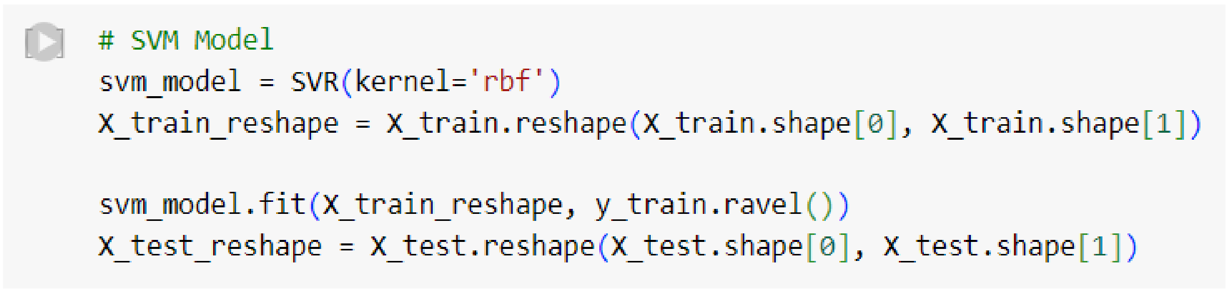
```
[ ] #Random Forest Model

    rf_model = RandomForestRegressor(n_estimators=100, random_state=42)
    rf_model.fit(X_train.reshape(X_train.shape[0], X_train.shape[1]), y_train.ravel())
```

```
▾     RandomForestRegressor
RandomForestRegressor(random_state=42)
```

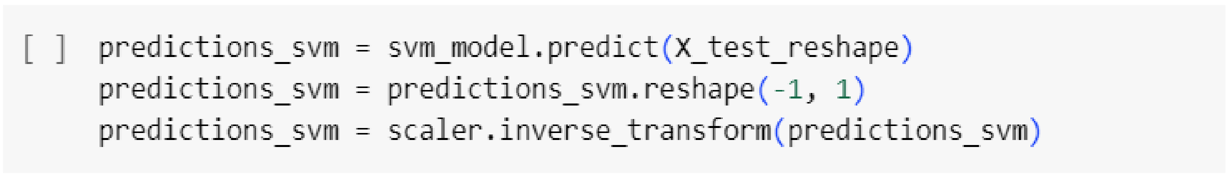
```
[ ] predictions_rf = rf_model.predict(X_test.reshape(X_test.shape[0], X_test.shape[1]))
    predictions_rf = predictions_rf.reshape(-1, 1)  # Reshape to 2D
    predictions_rf = scaler.inverse_transform(predictions_rf)
```

9. SVM model

FIGURE 3.31



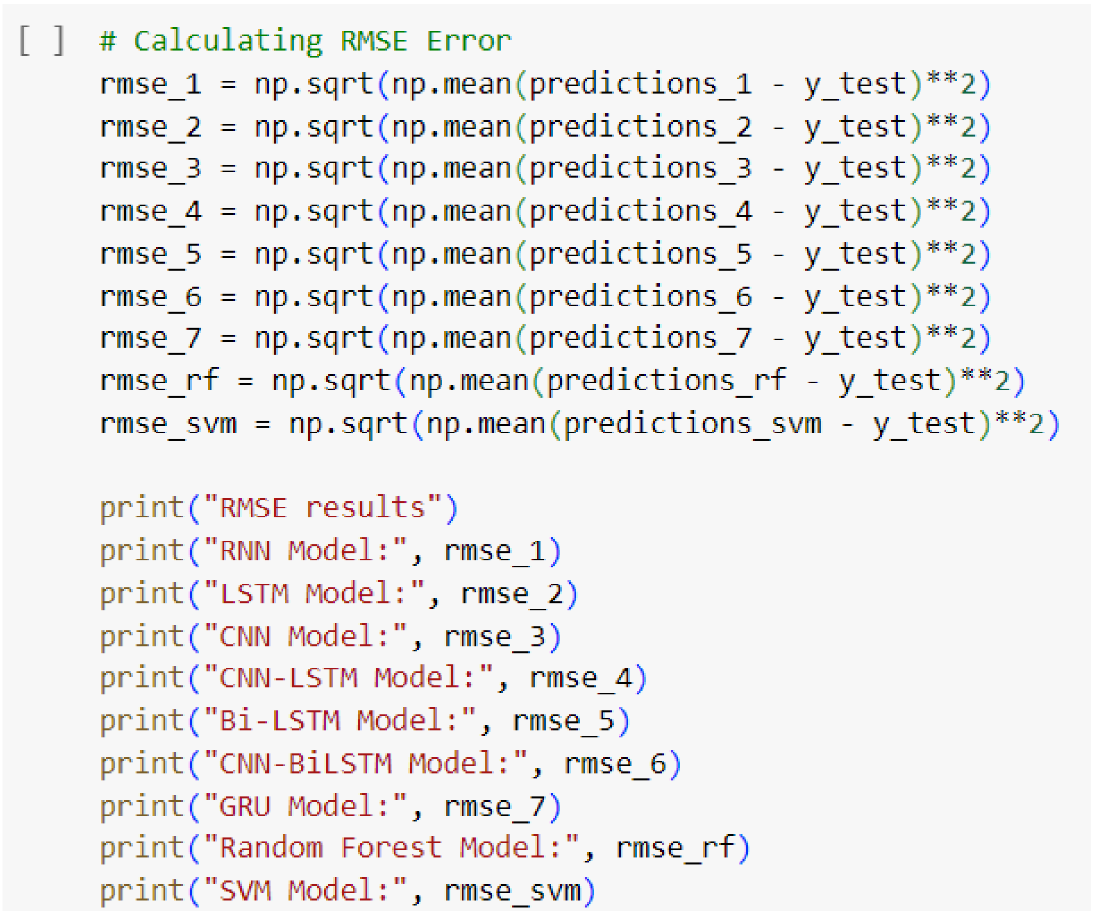
```
# SVM Model
svm_model = SVR(kernel='rbf')
X_train_reshape = X_train.reshape(X_train.shape[0], X_train.shape[1])

svm_model.fit(X_train_reshape, y_train.ravel())
X_test_reshape = X_test.reshape(X_test.shape[0], X_test.shape[1])
```



```
[ ] predictions_svm = svm_model.predict(X_test_reshape)
    predictions_svm = predictions_svm.reshape(-1, 1)
    predictions_svm = scaler.inverse_transform(predictions_svm)
```

Our study also has evaluations metrics for each model such as RMSE. Here is the code snippet of the following

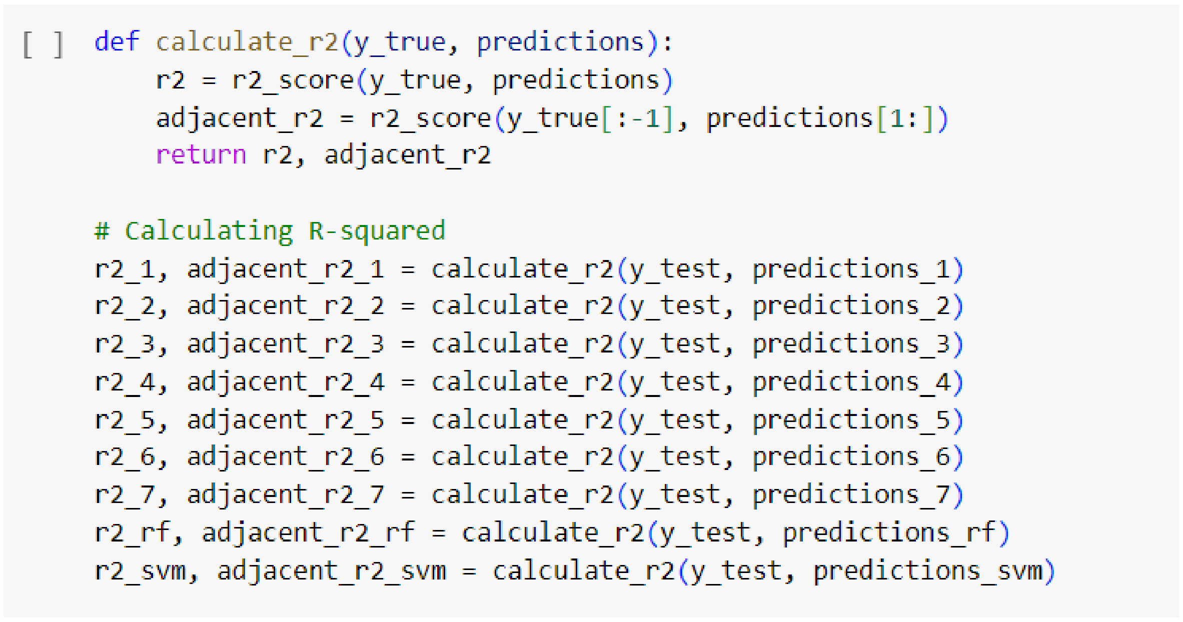
FIGURE 3.32



```
[ ] # Calculating RMSE Error
    rmse_1 = np.sqrt(np.mean(predictions_1 - y_test)**2)
    rmse_2 = np.sqrt(np.mean(predictions_2 - y_test)**2)
    rmse_3 = np.sqrt(np.mean(predictions_3 - y_test)**2)
    rmse_4 = np.sqrt(np.mean(predictions_4 - y_test)**2)
    rmse_5 = np.sqrt(np.mean(predictions_5 - y_test)**2)
    rmse_6 = np.sqrt(np.mean(predictions_6 - y_test)**2)
    rmse_7 = np.sqrt(np.mean(predictions_7 - y_test)**2)
    rmse_rf = np.sqrt(np.mean(predictions_rf - y_test)**2)
    rmse_svm = np.sqrt(np.mean(predictions_svm - y_test)**2)

    print("RMSE results")
    print("RNN Model:", rmse_1)
    print("LSTM Model:", rmse_2)
    print("CNN Model:", rmse_3)
    print("CNN-LSTM Model:", rmse_4)
    print("Bi-LSTM Model:", rmse_5)
    print("CNN-BiLSTM Model:", rmse_6)
    print("GRU Model:", rmse_7)
    print("Random Forest Model:", rmse_rf)
    print("SVM Model:", rmse_svm)
```

Code Snippet for evaluation metric R-squared and adjacent R-squared

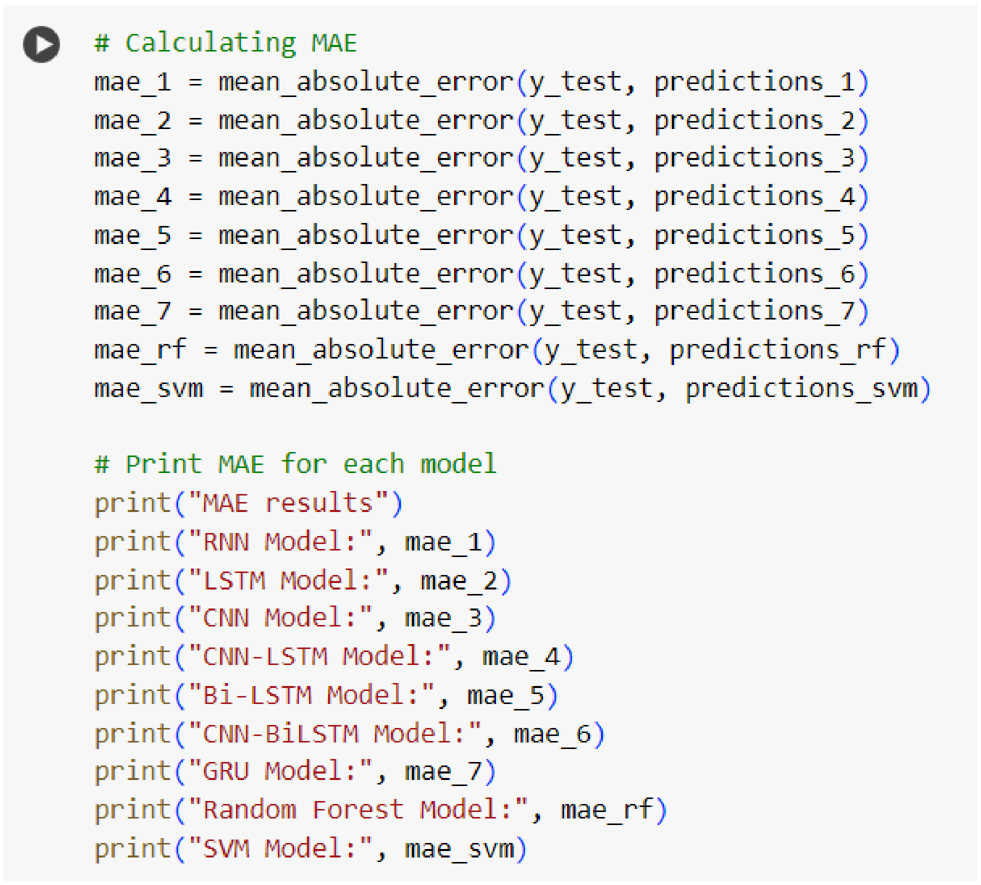
FIGURE 3.33



```
def calculate_r2(y_true, predictions):
    r2 = r2_score(y_true, predictions)
    adjacent_r2 = r2_score(y_true[:-1], predictions[1:])
    return r2, adjacent_r2

# Calculating R-squared
r2_1, adjacent_r2_1 = calculate_r2(y_test, predictions_1)
r2_2, adjacent_r2_2 = calculate_r2(y_test, predictions_2)
r2_3, adjacent_r2_3 = calculate_r2(y_test, predictions_3)
r2_4, adjacent_r2_4 = calculate_r2(y_test, predictions_4)
r2_5, adjacent_r2_5 = calculate_r2(y_test, predictions_5)
r2_6, adjacent_r2_6 = calculate_r2(y_test, predictions_6)
r2_7, adjacent_r2_7 = calculate_r2(y_test, predictions_7)
r2_rf, adjacent_r2_rf = calculate_r2(y_test, predictions_rf)
r2_svm, adjacent_r2_svm = calculate_r2(y_test, predictions_svm)
```

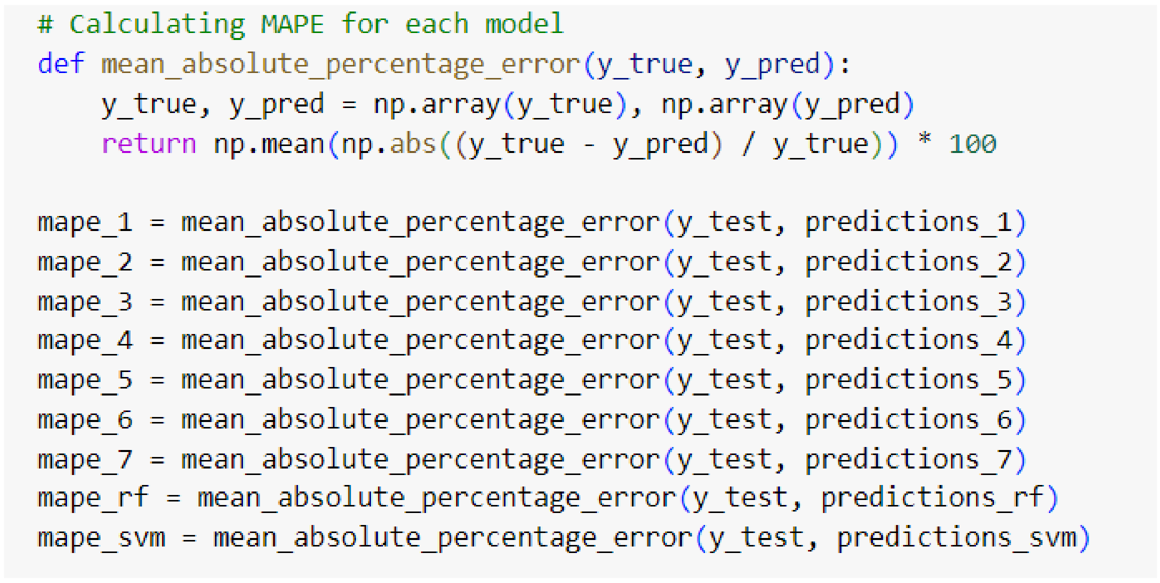
Code snippet for MAE (Figure 34) and MAPE (Figure 35)

FIGURE 3.34



```
# Calculating MAE
mae_1 = mean_absolute_error(y_test, predictions_1)
mae_2 = mean_absolute_error(y_test, predictions_2)
mae_3 = mean_absolute_error(y_test, predictions_3)
mae_4 = mean_absolute_error(y_test, predictions_4)
mae_5 = mean_absolute_error(y_test, predictions_5)
mae_6 = mean_absolute_error(y_test, predictions_6)
mae_7 = mean_absolute_error(y_test, predictions_7)
mae_rf = mean_absolute_error(y_test, predictions_rf)
mae_svm = mean_absolute_error(y_test, predictions_svm)

# Print MAE for each model
print("MAE results")
print("RNN Model:", mae_1)
print("LSTM Model:", mae_2)
print("CNN Model:", mae_3)
print("CNN-LSTM Model:", mae_4)
print("Bi-LSTM Model:", mae_5)
print("CNN-BiLSTM Model:", mae_6)
print("GRU Model:", mae_7)
print("Random Forest Model:", mae_rf)
print("SVM Model:", mae_svm)
```

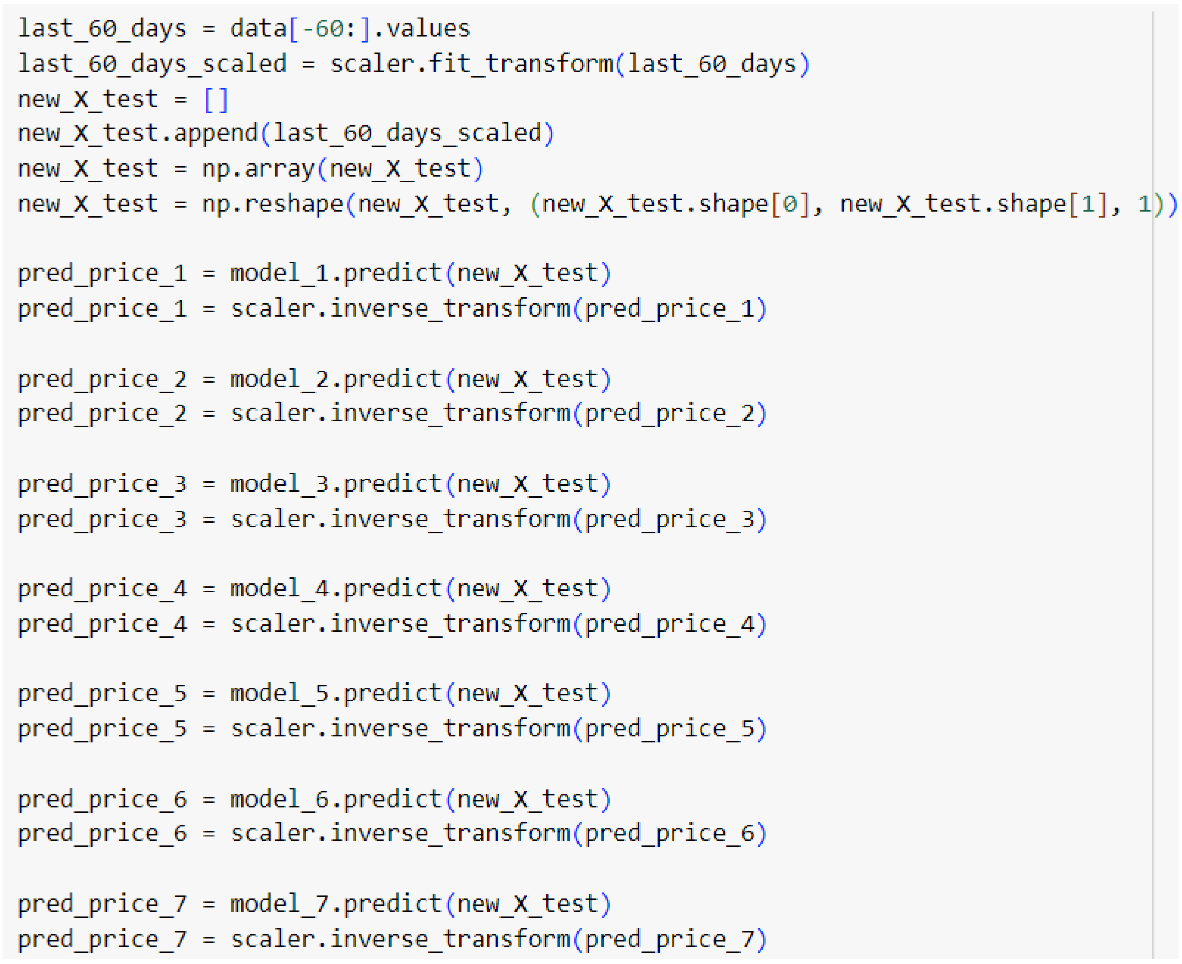
FIGURE 3.35



```
# Calculating MAPE for each model
def mean_absolute_percentage_error(y_true, y_pred):
    y_true, y_pred = np.array(y_true), np.array(y_pred)
    return np.mean(np.abs((y_true - y_pred) / y_true)) * 100

mape_1 = mean_absolute_percentage_error(y_test, predictions_1)
mape_2 = mean_absolute_percentage_error(y_test, predictions_2)
mape_3 = mean_absolute_percentage_error(y_test, predictions_3)
mape_4 = mean_absolute_percentage_error(y_test, predictions_4)
mape_5 = mean_absolute_percentage_error(y_test, predictions_5)
mape_6 = mean_absolute_percentage_error(y_test, predictions_6)
mape_7 = mean_absolute_percentage_error(y_test, predictions_7)
mape_rf = mean_absolute_percentage_error(y_test, predictions_rf)
mape_svm = mean_absolute_percentage_error(y_test, predictions_svm)
```

Following is the function created in order to last 60 for each model and predict the coding price values

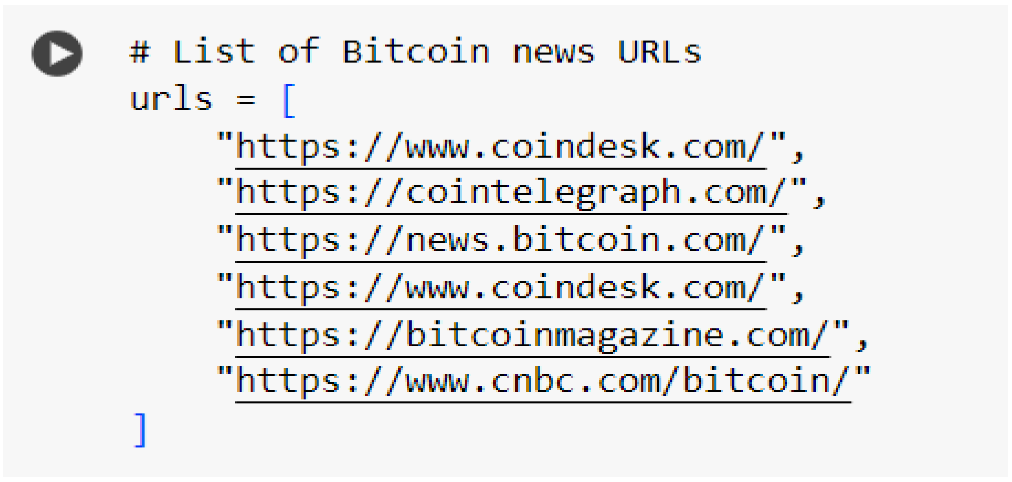
FIGURE 3.36



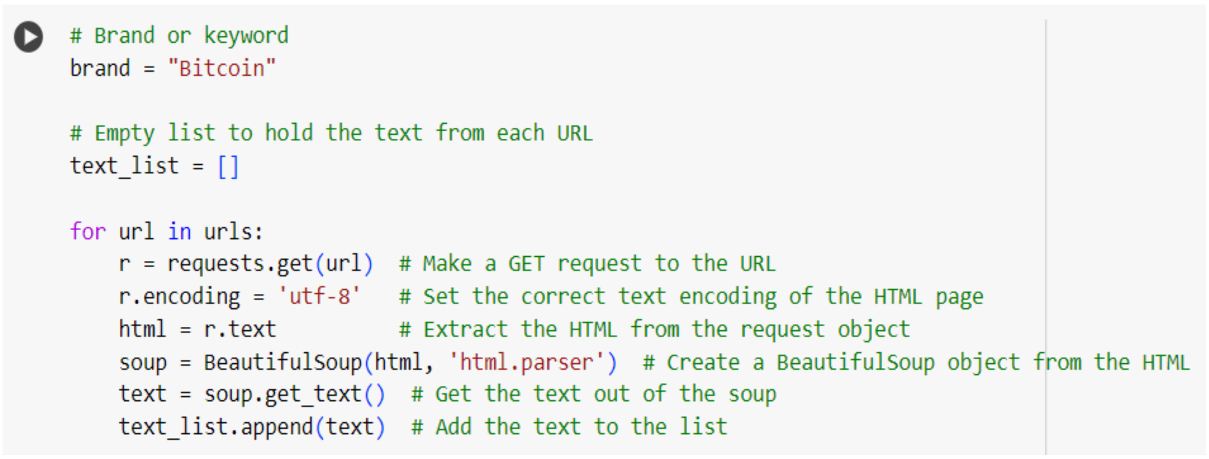
```
last_60_days = data[-60:].values
last_60_days_scaled = scaler.fit_transform(last_60_days)
new_X_test = []
new_X_test.append(last_60_days_scaled)
new_X_test = np.array(new_X_test)
new_X_test = np.reshape(new_X_test, (new_X_test.shape[0], new_X_test.shape[1], 1))

pred_price_1 = model_1.predict(new_X_test)
pred_price_1 = scaler.inverse_transform(pred_price_1)

pred_price_2 = model_2.predict(new_X_test)
pred_price_2 = scaler.inverse_transform(pred_price_2)

pred_price_3 = model_3.predict(new_X_test)
pred_price_3 = scaler.inverse_transform(pred_price_3)

pred_price_4 = model_4.predict(new_X_test)
pred_price_4 = scaler.inverse_transform(pred_price_4)

pred_price_5 = model_5.predict(new_X_test)
pred_price_5 = scaler.inverse_transform(pred_price_5)

pred_price_6 = model_6.predict(new_X_test)
pred_price_6 = scaler.inverse_transform(pred_price_6)

pred_price_7 = model_7.predict(new_X_test)
pred_price_7 = scaler.inverse_transform(pred_price_7)
```

Now, for sentiment analysis based on web scraping. The code snippet for accessing websites

FIGURE 3.37: URLs



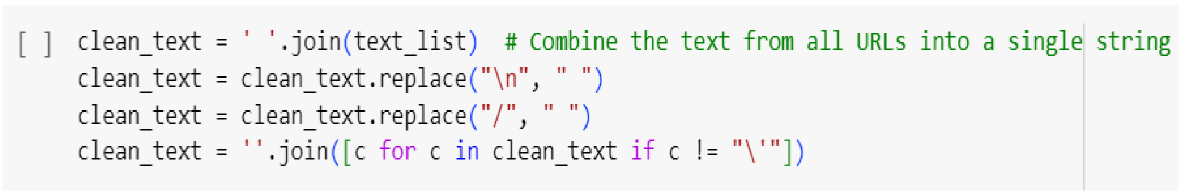
```
# List of Bitcoin news URLs
urls = [
    "https://www.coindesk.com/",
    "https://cointelegraph.com/",
    "https://news.bitcoin.com/",
    "https://www.coindesk.com/",
    "https://bitcoinmagazine.com/",
    "https://www.cnbc.com/bitcoin/"
]
```



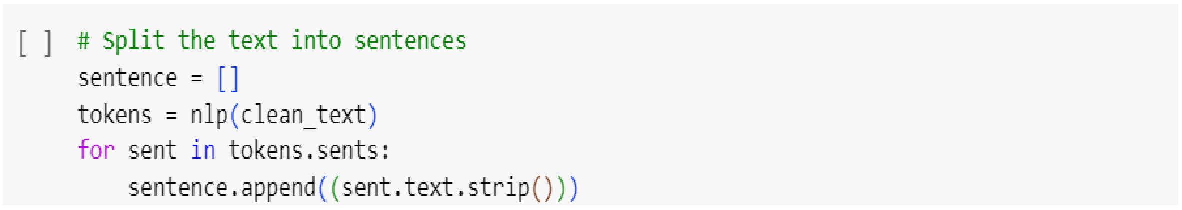
```
# Brand or keyword
brand = "Bitcoin"

# Empty list to hold the text from each URL
text_list = []

for url in urls:
    r = requests.get(url)  # Make a GET request to the URL
    r.encoding = 'utf-8'   # Set the correct text encoding of the HTML page
    html = r.text          # Extract the HTML from the request object
    soup = BeautifulSoup(html, 'html.parser')  # Create a BeautifulSoup object from the HTML
    text = soup.get_text()  # Get the text out of the soup
    text_list.append(text)  # Add the text to the list
```

Code snippet for combining and cleaning the texts and then forming the sentences.

FIGURE 3.38: TEXT PREPROCESSING



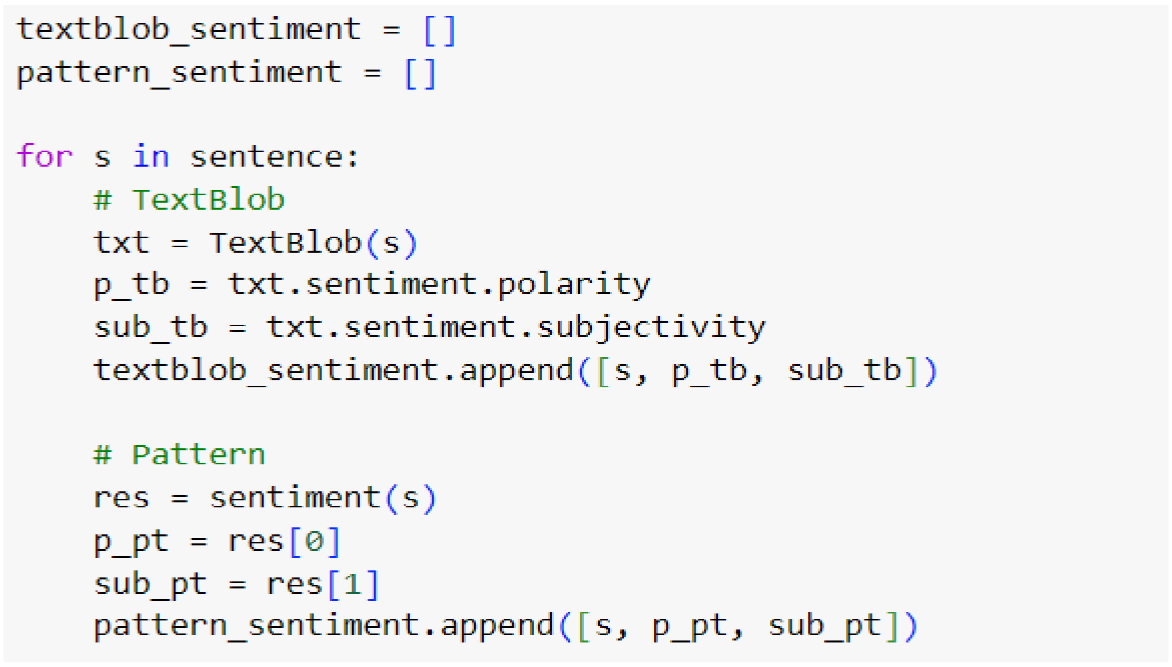
```
[ ]  clean_text = ' '.join(text_list)  # Combine the text from all URLs into a single string
     clean_text = clean_text.replace("\n", " ")
     clean_text = clean_text.replace("/", " ")
     clean_text = ''.join([c for c in clean_text if c != "\'"])
```



```
[ ]  # Split the text into sentences
     sentence = []
     tokens = nlp(clean_text)
     for sent in tokens.sents:
         sentence.append((sent.text.strip()))
```

Code snippet for Sentiment Analysis

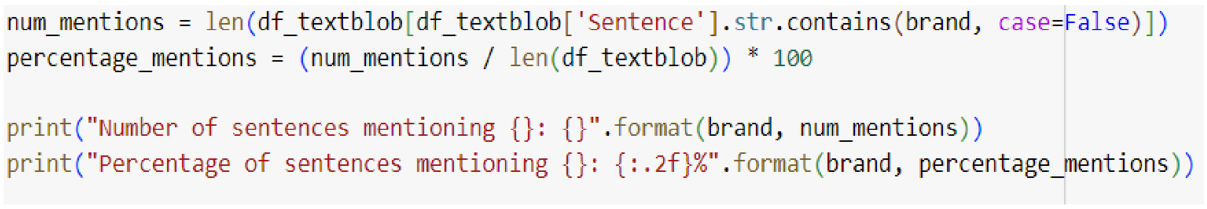
FIGURE 3.39: SENTIMENT ANALYSIS



```
textblob_sentiment = []
pattern_sentiment = []

for s in sentence:
    # TextBlob
    txt = TextBlob(s)
    p_tb = txt.sentiment.polarity
    sub_tb = txt.sentiment.subjectivity
    textblob_sentiment.append([s, p_tb, sub_tb])

    # Pattern
    res = sentiment(s)
    p_pt = res[0]
    sub_pt = res[1]
    pattern_sentiment.append([s, p_pt, sub_pt])
```

Code snippet for calculating number of sentences about bitcoin in the websites.

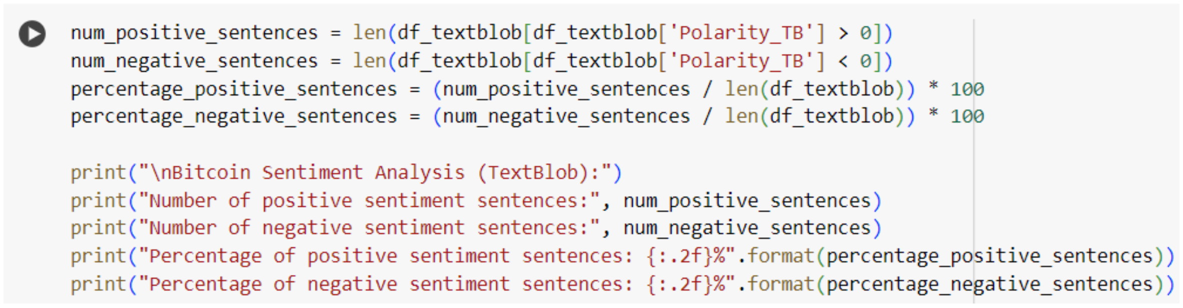
FIGURE 3.40: NUMBER OF BITCOIN SENTENCES



```
num_mentions = len(df_textblob[df_textblob['Sentence'].str.contains(brand, case=False)])
percentage_mentions = (num_mentions / len(df_textblob)) * 100

print("Number of sentences mentioning {}: {}".format(brand, num_mentions))
print("Percentage of sentences mentioning {}: {:.2f}%".format(brand, percentage_mentions))
```

Code snippet for calculating textblob positive and negative sentiments.

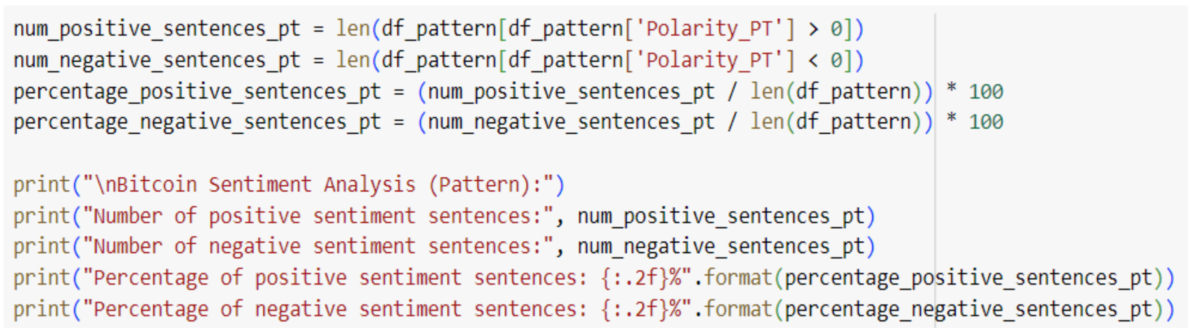
FIGURE 3.41: TEXTBLOB POSITIVE AND NEGATIVE SENTIMENTS



```
num_positive_sentences = len(df_textblob[df_textblob['Polarity_TB'] > 0])
num_negative_sentences = len(df_textblob[df_textblob['Polarity_TB'] < 0])
percentage_positive_sentences = (num_positive_sentences / len(df_textblob)) * 100
percentage_negative_sentences = (num_negative_sentences / len(df_textblob)) * 100

print("\nBitcoin Sentiment Analysis (TextBlob):")
print("Number of positive sentiment sentences:", num_positive_sentences)
print("Number of negative sentiment sentences:", num_negative_sentences)
print("Percentage of positive sentiment sentences: {:.2f}%".format(percentage_positive_sentences))
print("Percentage of negative sentiment sentences: {:.2f}%".format(percentage_negative_sentences))
```

Code snippet for calculating pattern positive and negative sentiments.

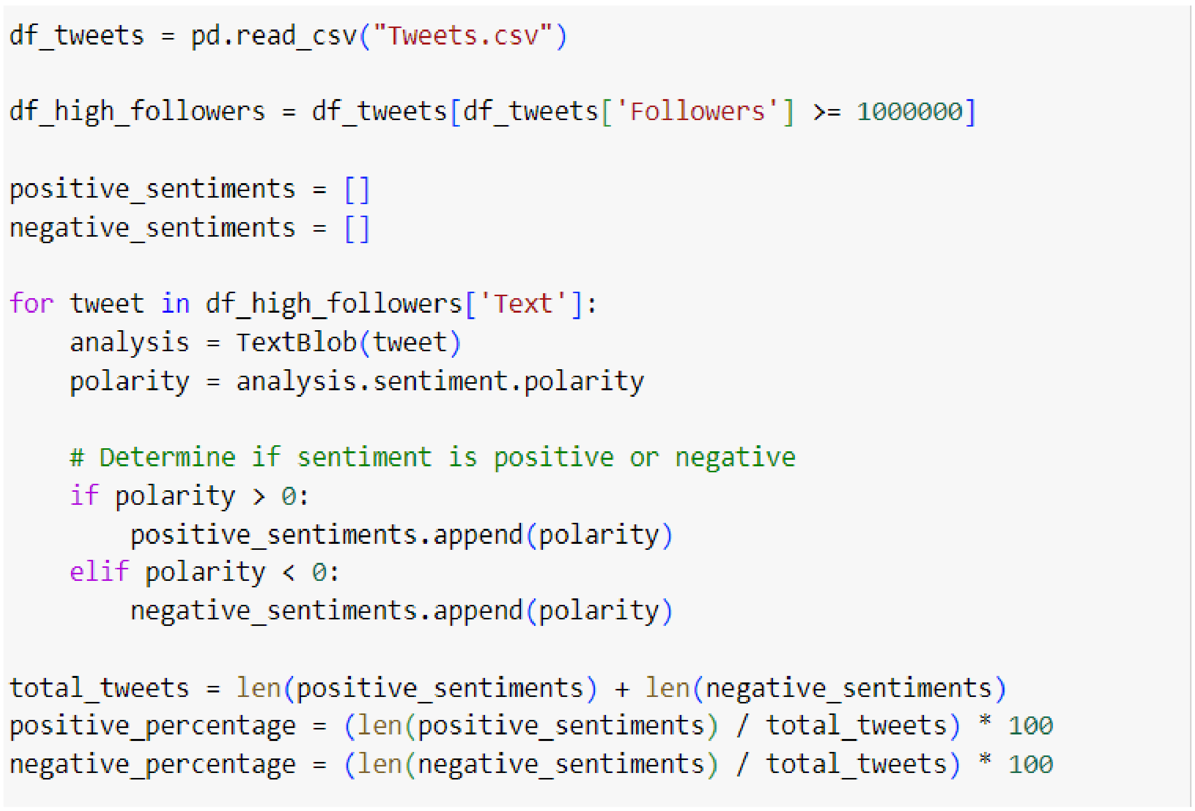
FIGURE 3.42: PATTERN POSITIVE AND NEGATIVE SENTIMENTS



```python
num_positive_sentences_pt = len(df_pattern[df_pattern['Polarity_PT'] > 0])
num_negative_sentences_pt = len(df_pattern[df_pattern['Polarity_PT'] < 0])
percentage_positive_sentences_pt = (num_positive_sentences_pt / len(df_pattern)) * 100
percentage_negative_sentences_pt = (num_negative_sentences_pt / len(df_pattern)) * 100

print("\nBitcoin Sentiment Analysis (Pattern):")
print("Number of positive sentiment sentences:", num_positive_sentences_pt)
print("Number of negative sentiment sentences:", num_negative_sentences_pt)
print("Percentage of positive sentiment sentences: {:.2f}%".format(percentage_positive_sentences_pt))
print("Percentage of negative sentiment sentences: {:.2f}%".format(percentage_negative_sentences_pt))
```

Further, for sentiment analysis for our dummy dataset. Code snippet for condition, positive and negative sentiment.

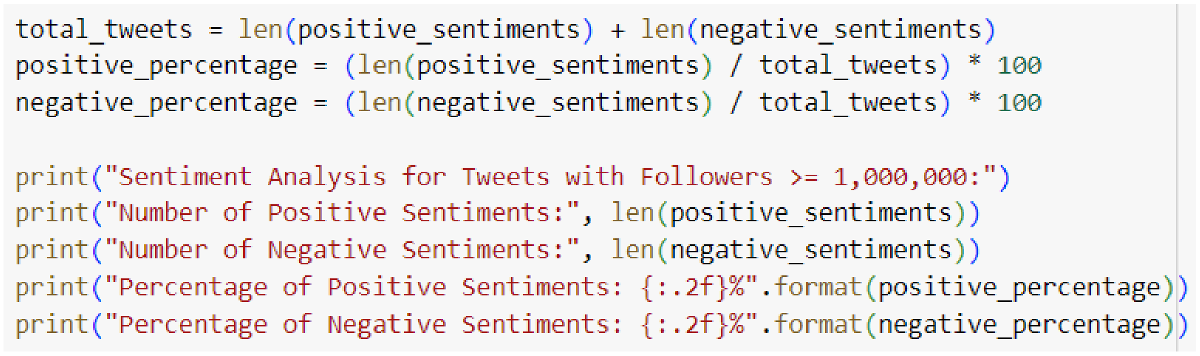
FIGURE 3.43: SENTIMENT ANALYSIS ON THE DUMMY DATASET



```python
df_tweets = pd.read_csv("Tweets.csv")

df_high_followers = df_tweets[df_tweets['Followers'] >= 1000000]

positive_sentiments = []
negative_sentiments = []

for tweet in df_high_followers['Text']:
    analysis = TextBlob(tweet)
    polarity = analysis.sentiment.polarity

    # Determine if sentiment is positive or negative
    if polarity > 0:
        positive_sentiments.append(polarity)
    elif polarity < 0:
        negative_sentiments.append(polarity)

total_tweets = len(positive_sentiments) + len(negative_sentiments)
positive_percentage = (len(positive_sentiments) / total_tweets) * 100
negative_percentage = (len(negative_sentiments) / total_tweets) * 100
```

Code snippet for positive and negative sentiment

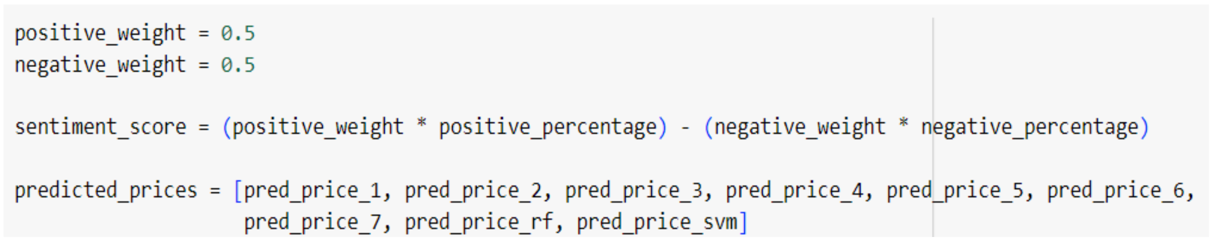
FIGURE 3.44: POSITIVE AND NEGATIVE SENTIMENTS



```
total_tweets = len(positive_sentiments) + len(negative_sentiments)
positive_percentage = (len(positive_sentiments) / total_tweets) * 100
negative_percentage = (len(negative_sentiments) / total_tweets) * 100

print("Sentiment Analysis for Tweets with Followers >= 1,000,000:")
print("Number of Positive Sentiments:", len(positive_sentiments))
print("Number of Negative Sentiments:", len(negative_sentiments))
print("Percentage of Positive Sentiments: {:.2f}%".format(positive_percentage))
print("Percentage of Negative Sentiments: {:.2f}%".format(negative_percentage))
```

Code snippet for sentiment score

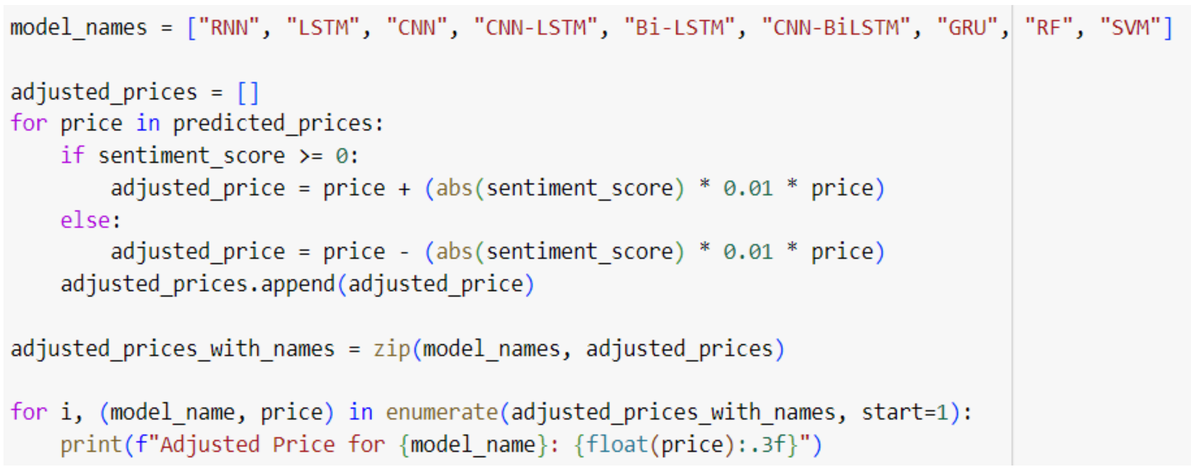
FIGURE 3.45: SENTIMENT SCORE



```
positive_weight = 0.5
negative_weight = 0.5

sentiment_score = (positive_weight * positive_percentage) - (negative_weight * negative_percentage)

predicted_prices = [pred_price_1, pred_price_2, pred_price_3, pred_price_4, pred_price_5, pred_price_6,
                    pred_price_7, pred_price_rf, pred_price_svm]
```

Code snippet for adjusted price

FIGURE 3.46: ADJUSTED PRICES



```
model_names = ["RNN", "LSTM", "CNN", "CNN-LSTM", "Bi-LSTM", "CNN-BiLSTM", "GRU", "RF", "SVM"]

adjusted_prices = []
for price in predicted_prices:
    if sentiment_score >= 0:
        adjusted_price = price + (abs(sentiment_score) * 0.01 * price)
    else:
        adjusted_price = price - (abs(sentiment_score) * 0.01 * price)
    adjusted_prices.append(adjusted_price)

adjusted_prices_with_names = zip(model_names, adjusted_prices)

for i, (model_name, price) in enumerate(adjusted_prices_with_names, start=1):
    print(f"Adjusted Price for {model_name}: {float(price):.3f}")
```

3.5 KEY CHALLENGES

1. Data Quality

    - The need for robust proactive procedures to deal with missing, noisy or inconsistent data.
    - Use robust data preprocessing pipelines.

2. Model Complexity
    - Properly assessing and fine-tuning complex neural architecture demanded careful considerations.
    - To overcome this challenge, it is important to thoroughly test and validate the models to balance complexity and performance.

3. Computational Resources
    - Deep learning models require significantly higher computational resources.
    - To mitigate this challenge, Collaboration with domain experts can simplify models while maintaining their predictive power.

4. Quality Assessment
    - The challenge of selecting appropriate assessment parameters for different models.
    - Optimizing the deep learning model architecture can also help balance performance and computational efficiency, using techniques like model compression to reduce resource requirements.

5. Scalability
    - Important to ensure that system can handle extensive amounts of data.
    - Can be implemented with scalable cloud-based solution role in demand.

6. Overfitting and Complexity
    - Price data for cryptocurrencies can contain misinformation, and advanced devices can overfit these errors, resulting in poor performance when applied to new data.
    - To overcome this challenge, it is necessary to regularize the models by incorporating different methods such L1 and L2 regularization.

7. Regulatory and Legal Considerations
    - The changing regulatory landscape impact cryptocurrency markets. Complying with regulatory requirements and following the regulatory changes is essential.

- To overcome this problem, it’s best to hire a good lawyer who specializes in cryptocurrency law.

8. Security Concerns
    - Cryptocurrencies face security threads such as hacking and fraud. System and model must be designed to withstand those security risks.
    - To meet this challenge, secure coding practices and encryption techniques are needed in order to protect system and instances from security threats.

9. Interpretability
    - Predictive model of the cryptocurrency prices can be complex, and understanding the reasoning behind the forecasts iv very hard to understand.
    - To solve this challenge, it is important ot use interpretable models whenever possible.

CHAPTER 4: TESTING

4.1 TESTING STRATEGY

In this section, we will discuss the experimental methodology we used in our study to see how well deep learning and machine learning models with sentiment analysis predicted Bitcoin prices. The main objective of this test is to evaluate the accuracy and reliability of these models under practical conditions. To guarantee a thorough analysis, we used a combination of time series classification.

TESTING TYPES

1. Unit Testing
2. Integration Testing
3. System Testing
4. Performance Testing
5. Validation Testing

TESTING TOOLS

1. Models

   To ensure diversity in our approach to time series forecasting, we use several deep leaning and machine learning models such as LSTM, RNN, CNN, Bi-LSTM, CNN-LSTM, CNN-BiLSTM, GRU, Random Forest and SVM were used.

2. Metrics

   We used several metrics such as Root Mean Square Error (RMSE), R-squared, adjacent R-squared, Mean Absolute Error (MAE) and Mean Absolute Percentage Error (MAPE) to evaluate the performances of the models successfully.

3. Training and Validation data

   We split the data into training and validation sets to speed up the learning process and test the performance of the models. To be precise, thirty percent of feedback was allocated for validation, and seventy percent was used for training. This split made it possible for the models to evaluate their performance on unseen data and learn from past data.

4.2 TEST CASES AND OUTCOMES

We executed a series of test cases to evaluate the models' ability to predict Bitcoin prices. The models were tested on the last 60 days of data, and their predictions were compared against the actual prices. Here are the outcomes of the test cases:

1. **RNN Model:**
   - **Predicted Price:** $61,062.543
2. **LSTM Model:**
   - **Predicted Price:** $61,675.043
3. **CNN Model:**
   - **Predicted Price:** $61,568.676
4. **CNN-LSTM Model:**
   - **Predicted Price:** $62,916.73
5. **Bi-LSTM Model:**
   - **Predicted Price:** $61,523.598
6. **CNN-BiLSTM Model:**
   - **Predicted Price:** $62,628.67
7. **GRU Model:**
   - **Predicted Price:** $61,332.453
8. **Random Forest Model:**
   - **Predicted Price:** $61,222.674
9. **SVM Model:**
   - **Predicted Price:** $65,294.410

Sentiment Analysis

Here, we represent sentiment analysis conducted on bitcoin related data obtained through web scraping. With TextBlob and Pattern libraries, we found out the sentiment polarity and subjectivity.

Number of sentences mentioning bitcoin: 72

Percentage of sentences mentioning bitcoin: 30.64%

1. **TextBlob**
    - **Polarity:**

        -Positive Sentiment: 39.57%

        -Negative Sentiment: 7.23%

2. **Pattern**
    - **Polarity:**

        -Positive Sentiment: 39.57%

        -Negative Sentiment: 7.23%

Now, for the second part of the sentiment analysis which is applied on our tweets dummy data, we have

1. **Polarity**

    -Positive Sentiment: 41.94%

    -Negative Sentiment: 58.06%

CHAPTER 5: RESULTS AND EVALUATION

5.1 RESULTS

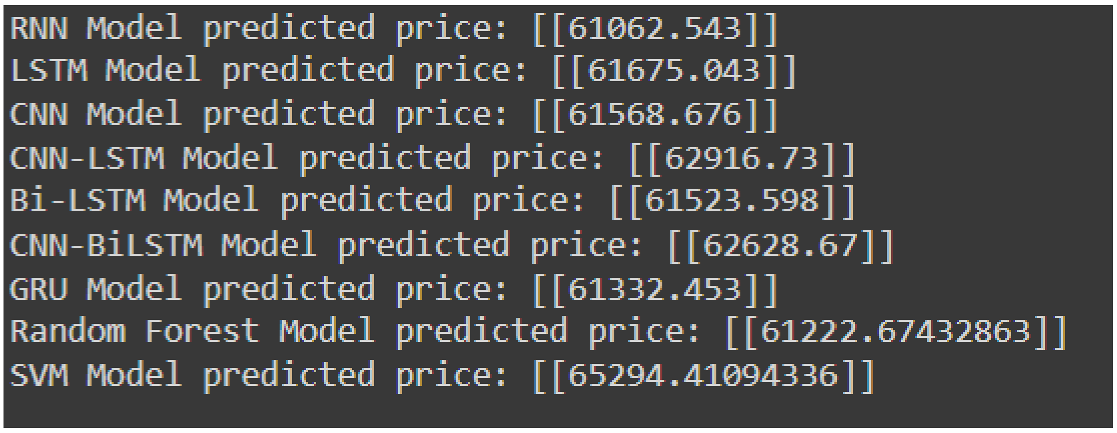
In this section we discuss the results of our predictive models for Bitcoin (BTC) closing price. All of our models have undergone training and validation using historical data of the Bitcoin, with later predictions applied to a testing subset. The models used in our projects are based on deep learning algorithms are Long Short-Term Memory (LSTM), Recurrent Neural Network (RNN), Convolution Neural Network (CNN), Convolution Neural Network-Long Short-Term Memory (CNN-LSTM), Bidirectional LSTM (Bi-LSTM), CNN-BiLSTM, Gated Recurrent Unit (GRU) and Machine Learning Models are Random Forest (RF), Support Vector Machine (SVM).

The dataset used in the models is of Bitcoin (BTC-USD). The historical daily prices of bitcoin are taken from yahoo finance. The dataset includes various values such as opening price, highest and lowest prices, closing price, adjusted closing price and trading volume for each day. The dataset gives the information enough for us to make and train models to predict the values. We are currently working with closing price of the dataset and we use time forecasting series to predict the closing value.

Seeing overall performance review, LSTM, CNN-LSTM and Bi-LSTM models have performed very well. LSTM model received a RMSE of 1624.75 while CNN-LSTM and Bi-LSTM models received even lower RMSE values i.e., 780.97 and 365.79 respectively. The R^2 value of the models are particularly high giving these models a strong essence to capture different patterns and variability. The predicted closing value of the Bitcoin of these models compared to the actual closing value of the Bitcoin are very close.

The actual closing price is 61448.39. Here is a figure to represent predicted price of all the models.

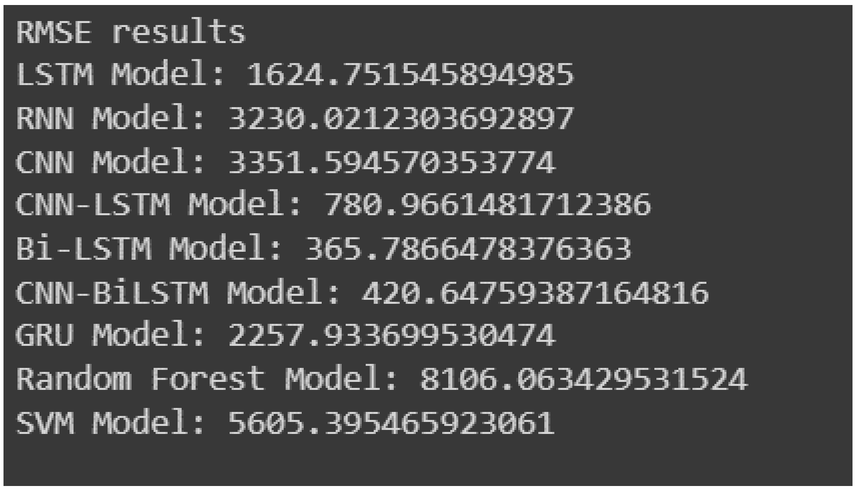
FIGURE 5.1: PREDICTED PRICES



```
RNN Model predicted price: [[61062.543]]
LSTM Model predicted price: [[61675.043]]
CNN Model predicted price: [[61568.676]]
CNN-LSTM Model predicted price: [[62916.73]]
Bi-LSTM Model predicted price: [[61523.598]]
CNN-BiLSTM Model predicted price: [[62628.67]]
GRU Model predicted price: [[61332.453]]
Random Forest Model predicted price: [[61222.67432863]]
SVM Model predicted price: [[65294.41094336]]
```

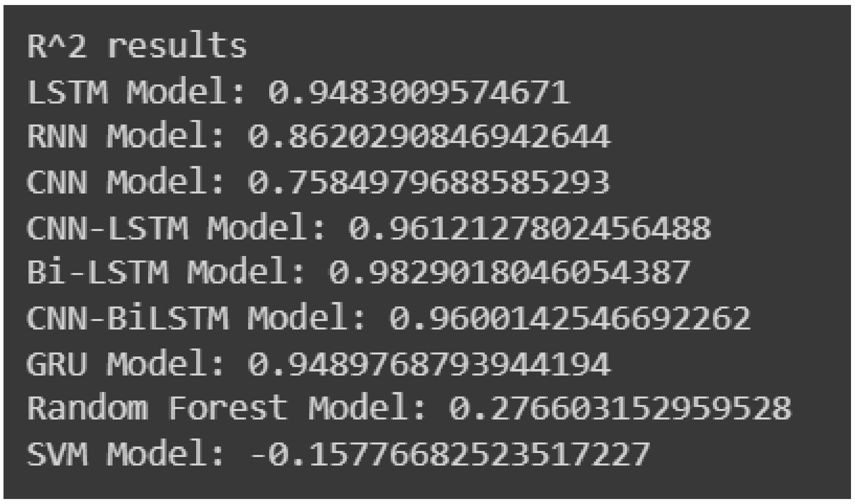
The following figures represent the values of evaluation metrics i.e., RMSE (Figure), R-squared (Figure), adjacent R-squared (Figure), MAE (Figure) and MAPE (Figure)

FIGURE 5.2: RMSE



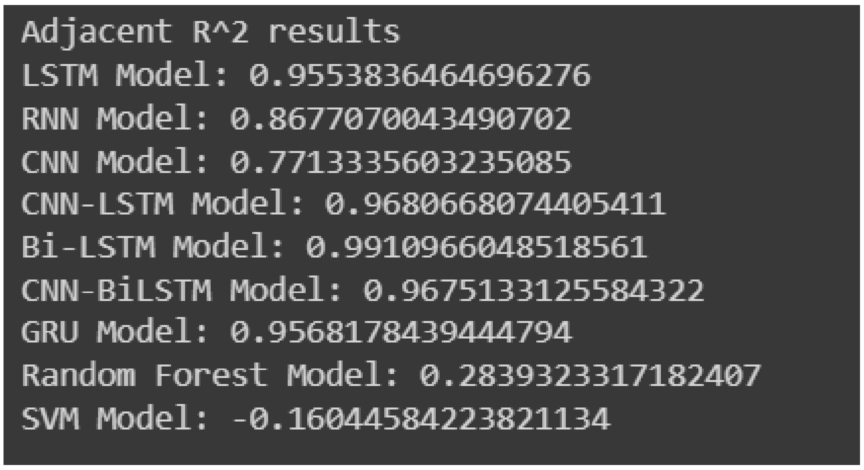
```
RMSE results
LSTM Model: 1624.751545894985
RNN Model: 3230.0212303692897
CNN Model: 3351.594570353774
CNN-LSTM Model: 780.9661481712386
Bi-LSTM Model: 365.7866478376363
CNN-BiLSTM Model: 420.64759387164816
GRU Model: 2257.933699530474
Random Forest Model: 8106.063429531524
SVM Model: 5605.395465923061
```

FIGURE 5.3: R-squared



```
R^2 results
LSTM Model: 0.9483009574671
RNN Model: 0.8620290846942644
CNN Model: 0.7584979688585293
CNN-LSTM Model: 0.9612127802456488
Bi-LSTM Model: 0.9829018046054387
CNN-BiLSTM Model: 0.9600142546692262
GRU Model: 0.9489768793944194
Random Forest Model: 0.276603152959528
SVM Model: -0.15776682523517227
```

FIGURE 5.4: Adjacent R-squared



```
Adjacent R^2 results
LSTM Model: 0.9553836464696276
RNN Model: 0.8677070043490702
CNN Model: 0.7713335603235085
CNN-LSTM Model: 0.9680668074405411
Bi-LSTM Model: 0.9910966048518561
CNN-BiLSTM Model: 0.9675133125584322
GRU Model: 0.9568178439444794
Random Forest Model: 0.2839323317182407
SVM Model: -0.16044584223821134
```

FIGURE 5.5: MAE

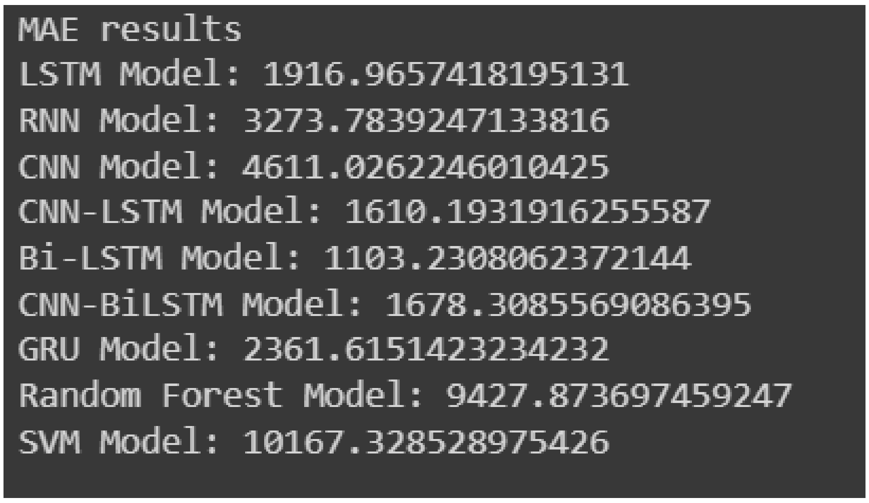
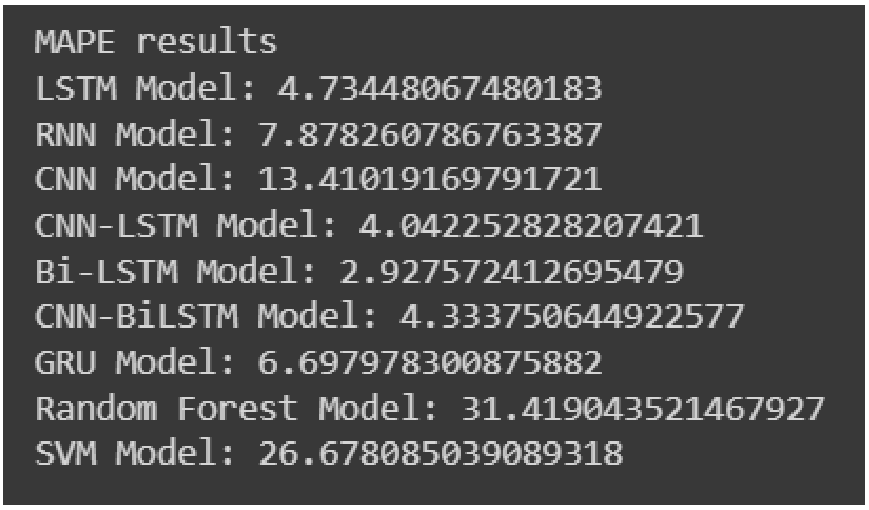

```
MAE results
LSTM Model: 1916.9657418195131
RNN Model: 3273.7839247133816
CNN Model: 4611.0262246010425
CNN-LSTM Model: 1610.1931916255587
Bi-LSTM Model: 1103.2308062372144
CNN-BiLSTM Model: 1678.3085569086395
GRU Model: 2361.6151423234232
Random Forest Model: 9427.873697459247
SVM Model: 10167.328528975426
```

FIGURE 5.6: MAPE


```
MAPE results
LSTM Model: 4.73448067480183
RNN Model: 7.878260786763387
CNN Model: 13.41019169791721
CNN-LSTM Model: 4.042252828207421
Bi-LSTM Model: 2.927572412695479
CNN-BiLSTM Model: 4.333750644922577
GRU Model: 6.697978300875882
Random Forest Model: 31.419043521467927
SVM Model: 26.678085039089318
```

While in GRU model, the RMSE value was 2257.93, the R^2 value of the model is 0.94, almost as same as LSTM model and the predicted value of the GRU model gives very close value of the Bitcoin to the actual value.

The other models like CNN and RNN although provided with close predictions but their results were not moderate as compared to LSTM-based models. CNN gave a RMSE value of 3351.59 while RNN gave a RMSE value of 3230.02, indicating not as good in predicting the prices.

Remaining Machine Learning Models presented higher RMSE values of 8106.06 and 5605.40 respectively. In time forecasting series, Machine Learning models showed limitation in discovering and capturing different data and patters, leading to most far off results from the actual closing price.

Table 5.1 shows an overall analysis of different models Root Mean Square Error (RMSE), R^2, Adjacent R^2, Mean Absolute Error (MAE), Mean Absolute Percentage Error (MAPE).

TABLE 5.1: METRICS COMPARISON

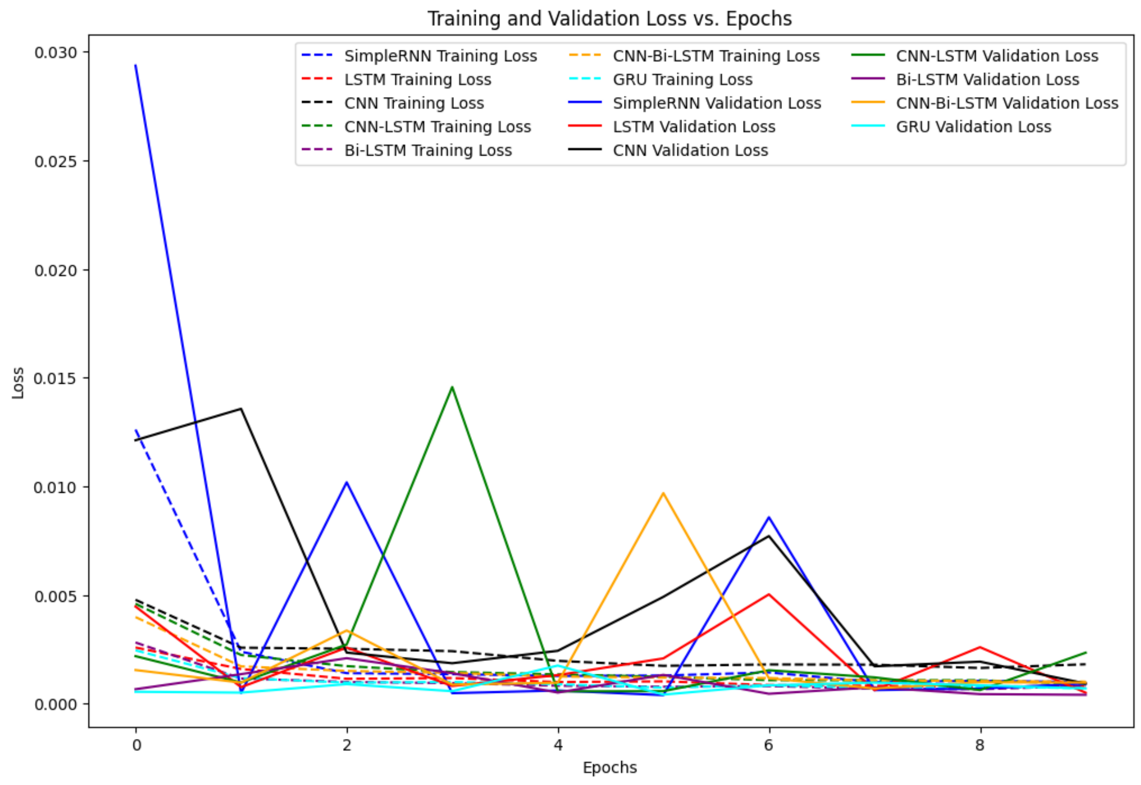
Models	**RMSE**	**R^2**	**Adjacent R^2**	**MAE**	**MAPE**
RNN Model	3230.02	0.8620	0.8677	3273.78	7.87
LSTM Model	1624.75	0.9483	0.9553	1916.96	4.73
CNN Model	3351.59	0.7584	0.7713	4611.02	13.41
CNN-LSTM Model	780.96	0.9612	0.9680	1610.09	4.04
Bi-LSTM Model	365.78	0.9829	0.9910	1103.23	2.92
CNN-BiLSTM Model	420.64	0.9601	0.9675	1678.31	4.33
GRU Model	2257.93	0.9489	0.9568	2361.61	6.69
RF Model	8106.06	0.2766	0.2839	9427.87	31.41
SVM Model	5605.39	0.1577	0.1604	10167.32	26.67

In last, traditional machine learning models i.e., Random Forest (RF) and Support Vector Machine (SVM) models, display limitations in predicting Bitcoin price while LSTM architecture-based models give the best predictions.

Furthermore, the approach of CNN-BiLSTM model i.e., combining convolution and bidirectional LSTM layers, demonstrated competitive results, allowing the hybrid model to capture more complex data and patterns.

Now, I’m going to show loss vs epoch graph for each models training as well as validation loss (Figure)

FIGURE 5.7: LOSS VS EPOCHS



Now, for the sentiment analysis part. It is divided in two parts. First is where we have textual data from various sources such news articles and websites. We have further applied sentiment analysis on it through TextBlob and Pattern libraries to find out the subjectivity and polarity of the sentences obtained. Now, for the graphs representing textblob subjectivity and polarity are

FIGURE 5.8: TEXTBLOB SUBJECTIVITY

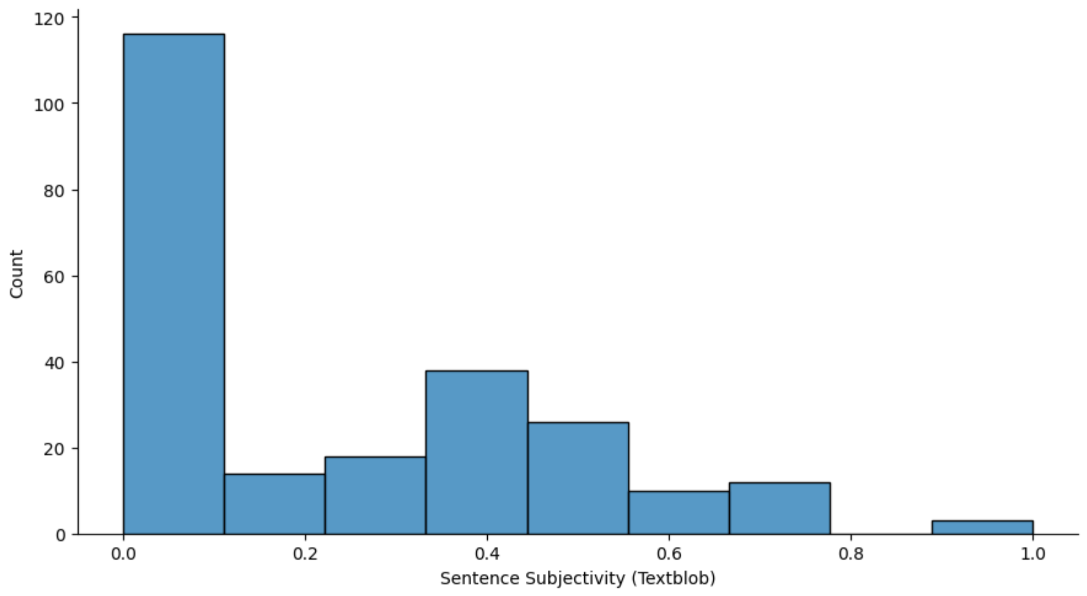
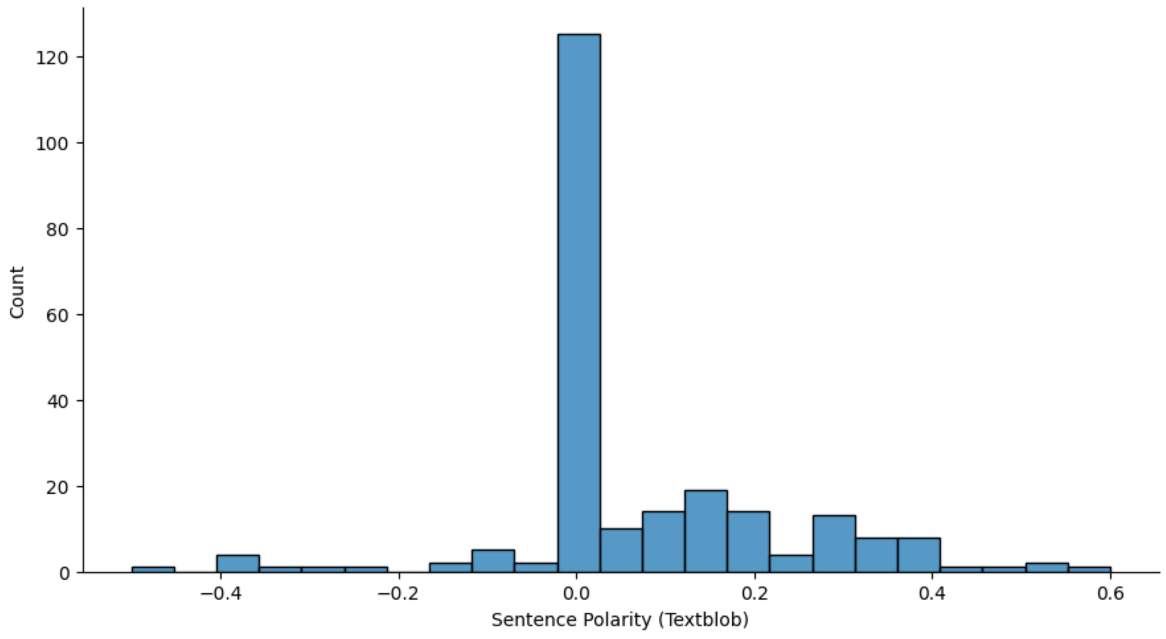
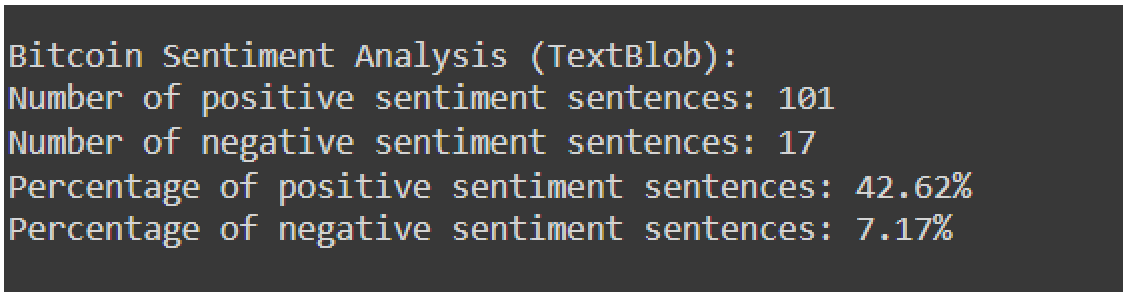


FIGURE 5.9: TEXTBLOB POLARITY



The total number of positive and negative sentences and percentage are

FIGURE 5.10: PERCENTAGE (TEXTBLOB)



```
Bitcoin Sentiment Analysis (TextBlob):
Number of positive sentiment sentences: 101
Number of negative sentiment sentences: 17
Percentage of positive sentiment sentences: 42.62%
Percentage of negative sentiment sentences: 7.17%
```

The graph representing the pattern subjectivity and polarity

FIGURE 5.11: PATTERN SUBJECTIVITY

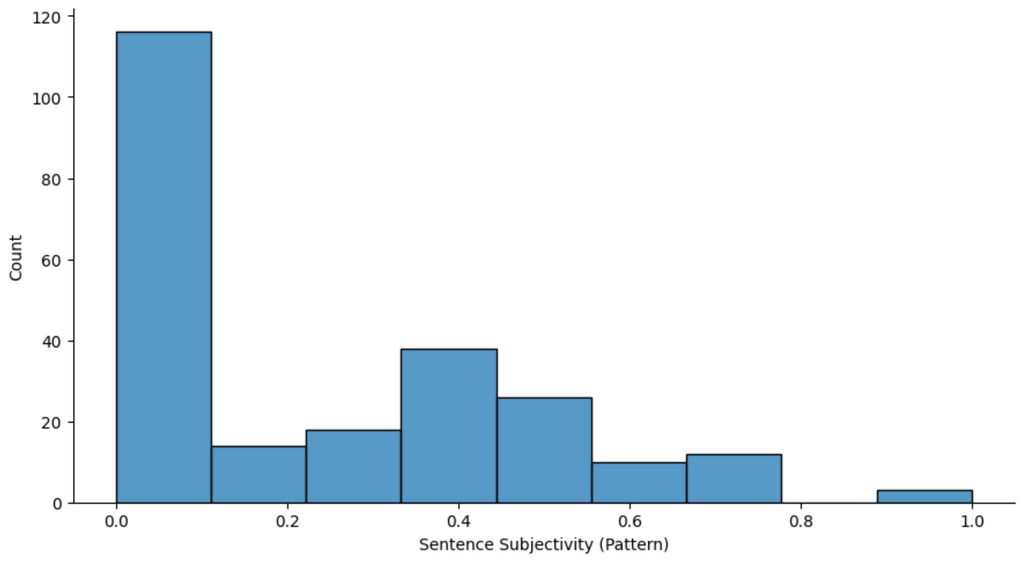
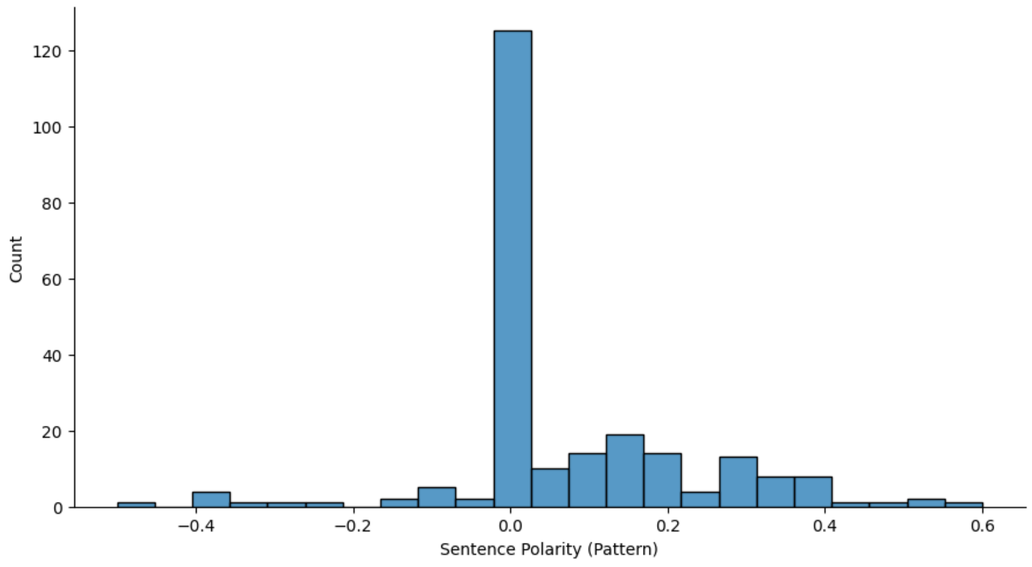
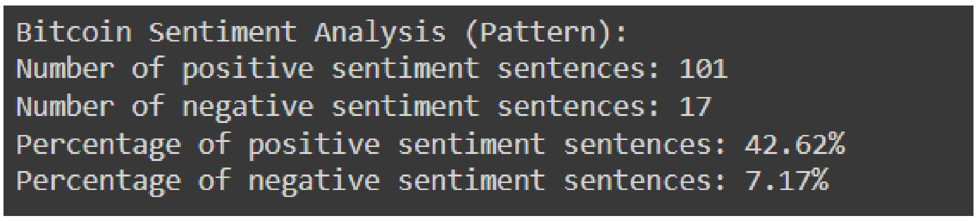


FIGURE 5.12: PATTERN POLARITY



The total number of positive and negative percentages are

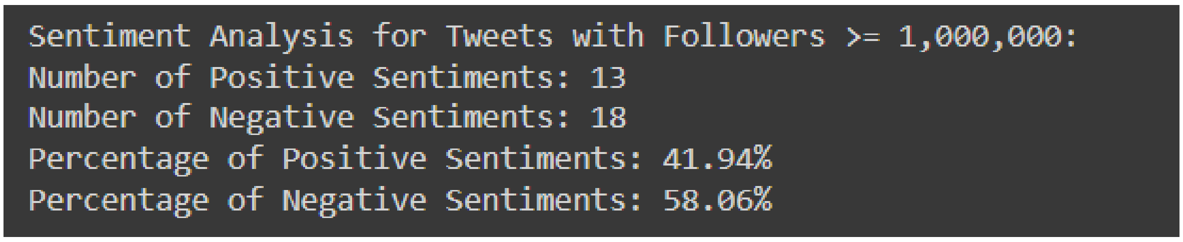
FIGURE 5.13: PERCENTAGE



```
Bitcoin Sentiment Analysis (Pattern):
Number of positive sentiment sentences: 101
Number of negative sentiment sentences: 17
Percentage of positive sentiment sentences: 42.62%
Percentage of negative sentiment sentences: 7.17%
```

Now for the second part of the sentiment analysis, we created a dummy data csv based on twitter. After that we applied sentiment analysis to the filter out tweets of high followers. After that we found out the sentiment score and based on sentiment score, we adjusted our prices based on sentiment analysis. The number of positive and negative sentences and percentage of sentiments are

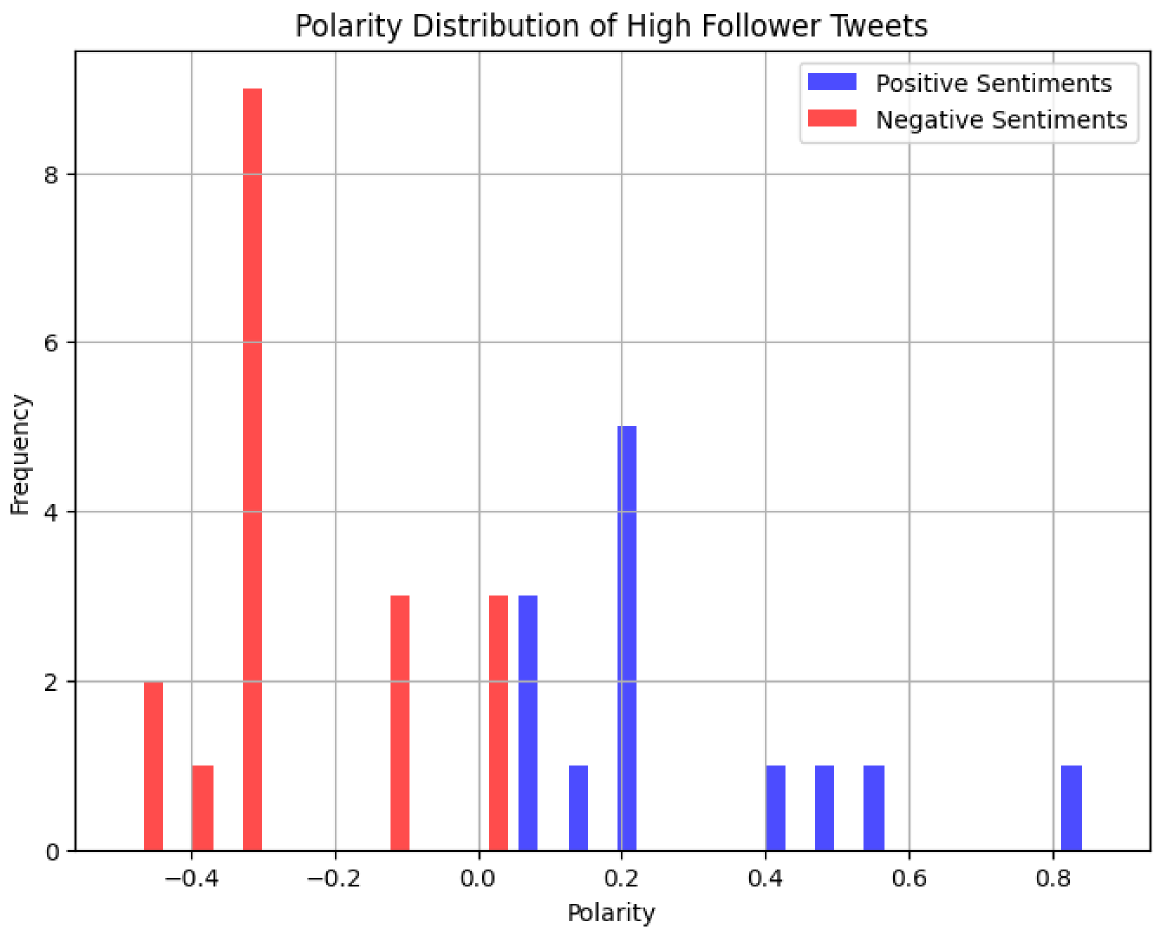
FIGURE 5.14: DUMMY DATA SENTIMENT



```
Sentiment Analysis for Tweets with Followers >= 1,000,000:
Number of Positive Sentiments: 13
Number of Negative Sentiments: 18
Percentage of Positive Sentiments: 41.94%
Percentage of Negative Sentiments: 58.06%
```

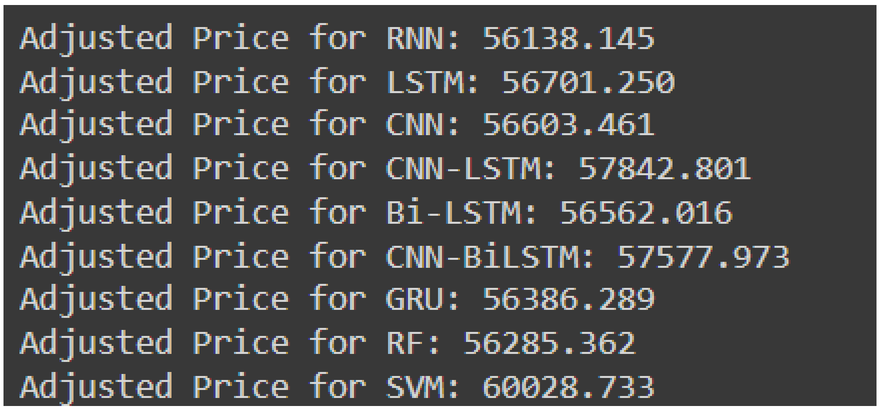
The graph shows the polarity distribution

FIGURE 5.15: POLARITY DISTRIBUTION



Now, the adjusted price for different models is

FIGURE 5.16: ADJUSTED PRICE



```
Adjusted Price for RNN: 56138.145
Adjusted Price for LSTM: 56701.250
Adjusted Price for CNN: 56603.461
Adjusted Price for CNN-LSTM: 57842.801
Adjusted Price for Bi-LSTM: 56562.016
Adjusted Price for CNN-BiLSTM: 57577.973
Adjusted Price for GRU: 56386.289
Adjusted Price for RF: 56285.362
Adjusted Price for SVM: 60028.733
```

Now, the data we used for predicting the new prices with sentiment analysis gives us highly compatible results which in turn shows to application of the program i.e. if the we replace the dummy data with real time tweets, we are highly capable of getting accurate prices in real time.

The reason we are not using real-time data is because now only v1 endpoints of twitter remains free for data collection which does not allow us to retrieve all the information we need for our analysis. To get that access we need to pay $160 per month. This is why we have shown our application on the dummy data that if anyone further wishes to use our project, they can get real-time tweets for real-time prices.

CHAPTER 6: CONCLUSIONS AND FUTURE SCOPE

6.1 CONCLUSION

In conclusion, our study contributes by providing a comprehensive evaluation and offers valuable insights for predicting bitcoin closing prices in both deep learning and machine learning models. The Bi-LSTM model has been most effective, showing it's potential in time series financial forecasting data. With an exceptional R^2 value of 0.9829, the Bi-LSTM outperformed every other model making it a high valuable tool for stakeholders.

When traditional machine learning models like random forest, support vector machine showed limitations in accurately predicting bitcoin closing price, the LSTM model, which is kind of recurrent neural network, showed commendable performance, emphasizing the importance deep learning algorithms in time forecasting series. Additionally, the CNN-BiLSTM hybrid model, using both convolution and bidirectional LSTM, showed competitive results.

Despite the great success of the models, it is important to acknowledge the limitations and contributions of including the inherent volatility of cryptocurrency market and the sensitivity of these models changing to market conditions.

Limitations:

1. Sensitivity to Market Conditions-
   Both deep learning as well as machine learning models are inherently sensitive to market data. The cryptocurrency market, known for its volatility, poses challenges for these models to adapt quickly to sudden and unexpected changes in the market.

2. Data Dependency-
   These models rely heavily on historical data for training. Sudden changes in market dynamics and unprecedented events can lead to suboptimal performance because the models may not be sensitive enough to unexpected situations.

3. Model Complexity-
   Deep learning models, especially those with multiple layers such as Bi-LSTM and CNN-BiLSTM, can be very expensive and require large amounts of resources. Their complexity may limit their application in real-time applications.

4. Interpretability-
   The black box nature of the deep learning models like LSTM and CNN-BiLSTM limits their interpretations. Understanding the reason behind a particular prediction can be very tough.

Contributions:

1. Performance of Deep Learning Model-
   The performance of deep learning model demonstrates the effectiveness in predicting the value, especially the Bi-LSTM algorithm. This contributes to the growing effectiveness of deep learning in capturing complex patterns and structures in time series forecasting data.

2. Hybrid Model-
   Exploring hybrid models such as CNN-BiLSTM highlights the potential importance of combining different models.

3. Benchmarking Traditional Models-
   The traditional machine learning models like random forest and support vector machine serves as a benchmark for comparison. This helps in valuable insights into the relative strengths and weakness of deep learning models.

4. Real-World Applicability-
   The study provides useful implications for stakeholders, investors and traders. This can guide decision making in cryptocurrency market.

5. Model Evaluations-
   The evaluation metrics like RMSE, R-squared, adjacent R-squared, MAE and MAPE provides some valuable insights to evaluate the performance of models in predicting the prices.

In conclusion, this study provides valuable insights into cryptocurrency forecasting, providing a basis for informed decision making and laying the groundwork for future developments in forecasting models.

6.2 FUTURE SCOPE

Following are the future scope of our study-

1. Enhancing the hybrid model:
   Previous studies have examined both independent models and hybrid architectures. To further improve the hybrid approach, future research could concentrate on amalgamating the strengths of different models to create more resilient hybrid architectures. For instance, the integration of deep learning models with traditional machine learning methods or the exploration of innovative combinations can significantly enhance prediction accuracy.

2. Inclusion of External Factors:
   Presently, models primarily rely on historical price data. However, future research could delve into the incorporation of external factors such as market sentiment analysis, macroeconomic indicators, and regulatory developments. By doing so, models would possess a greater ability to adapt to changing market conditions and external influences.

3. Dynamic model optimization:
   It is imperative to develop models that possess the capability to dynamically adapt to the ever-changing market dynamics and volatile fluctuations. This can be achieved by constructing models with adaptive coefficients or employing reinforcement learning techniques to adapt in real-time based on the evolving cryptocurrency markets.

4. Interpretation and interpretation:
   The interpretability of models, particularly in the realm of finance where stakeholders often require transparent and meaningful forecasts, needs to be enhanced. Future research should prioritize the development of models that are valuable to traders, investors, and decision-makers, while also providing detailed implications of their predictions.

5. Risk Assessment and Uncertainty Modelling:
   Addressing the uncertainty prevalent in cryptocurrency markets is a crucial area for future research. A model that not only predicts prices but also quantifies them and accounts for the associated uncertainties would be immensely valuable for effective risk management.

6. Scalability and real-time processing:
   Scalability and real-time processing are crucial factors in handling the increasing number of cryptocurrency transactions. To ensure efficient handling of large data, future research can focus on optimizing model architectures and providing timely predictions.

7. Cross-Cryptocurrency Analysis:
   A comprehensive understanding of market dynamics can be achieved by broadening the analysis of multiple cryptocurrencies simultaneously. Comparative analysis across different cryptocurrencies can reveal common patterns or unique characteristics specific to individual digital assets.

The growing impact of AI on investment decision making necessitates addressing ethical considerations. Future research could explore the ethical implications of applying predictive models to financial markets, ensuring fairness, accountability, and transparency. In conclusion, future research should aim to push the boundaries of current research by addressing limitations, exploring alternatives, adapting models to the evolving state of the cryptocurrency market, and ensuring that machine learning is developed responsibly.

References

1. M, Iyyappan. et al. (2022) ‘A novel AI-based stock market prediction using machine learning algorithm’, Scientific Programming, 2022, pp. 1–11. doi:10.1155/2022/4808088.

2. Dogra, V. et al. (2022) ‘A complete process of text classification system using state-of-the-art NLP models’, Computational Intelligence and Neuroscience, 2022, pp. 1–26. doi:10.1155/2022/1883698.

3. Lee, P., Huang, Z. and Tang, Y. (2022) ‘Trend prediction model of Asian stock market volatility dynamic relationship based on machine learning’, Security and Communication Networks, 2022, pp. 1–10. doi:10.1155/2022/5972698.

4. Li, J. (2021) ‘Research on Market Stock index prediction based on Network Security and Deep Learning’, Security and Communication Networks, 2021, pp. 1–8. doi:10.1155/2021/5522375.

5. Zhang, J. and Lei, Y. (2022) ‘Deep Reinforcement Learning for stock prediction’, Scientific Programming, 2022, pp. 1–9. doi:10.1155/2022/5812546.

6. Anveshrithaa, S. and Lavanya, K. (2020) ‘Real time vehicle traffic analysis using long short term memory networks in Apache Spark’, 2020 International Conference on Emerging Trends in Information Technology and Engineering (ic-ETITE) [Preprint].doi:10.1109/ic-etite47903.2020.97.

7. Vijh, M. et al. (2020) ‘Stock closing price prediction using Machine Learning Techniques’, Procedia Computer Science, 167, pp. 599–606. doi:10.1016/j.procs.2020.03.326.

8. Seif, M.M., Ramzy Hamed, E.M. and Abdel Ghfar Hegazy, A.E. (2018) ‘Stock market real time recommender model using Apache Spark Framework’, The International

Conference on Advanced Machine Learning Technologies and Applications (AMLTA2018), pp. 671–683. doi:10.1007/978-3-319-74690-6_66.

9. V Kranthi and Sai Reddy,"Stock Market Prediction Using Machine Learning", IRJET 2018.

10. Selvin,S. et al . (2017) ‘Stock price prediction using LSTM, RNN and CNN-sliding window model’, 2017 International Conference on Advances in Computing, Communications and Informatics (ICACCI) [Preprint].doi:10.1109/icacci.2017.8126078.

11. Usmani, M. et al. (2016) ‘Stock market prediction using Machine Learning Techniques’, 2016 3rd International Conference on Computer and Information Sciences (ICCOINS) [Preprint]. doi:10.1109/iccoins.2016.7783235.

12. Banik, S., Khodadad Khan, A.F. and Anwer, M. (2014) ‘Hybrid machine learning technique for forecasting Dhaka stock market timing decisions’, Computational Intelligence and Neuroscience, 2014, pp. 1–6. doi:10.1155/2014/318524.

13. Shah, A. et al. (2022) ‘A stock market trading framework based on Deep Learning Architectures’, Multimedia Tools and Applications, 81(10), pp. 14153–14171. doi:10.1007/s11042-022-12328-x.

14. Hoseinzade, E. and Haratizadeh, S. (2019) ‘CNNpred: CNN-based stock market prediction using a diverse set of variables’, *Expert Systems with Applications*, 129, pp. 273–285. doi:10.1016/j.eswa.2019.03.029.

15. Kara, Y., Acar Boyacioglu, M. and Baykan, Ö.K. (2011) ‘Predicting direction of stock price index movement using artificial neural networks and support vector machines: The sample of the Istanbul Stock Exchange’, *Expert Systems with Applications*, 38(5), pp. 5311–5319. doi:10.1016/j.eswa.2010.10.027.

16. Draw.io - free flowchart maker and diagrams online Flowchart Maker & Online Diagram Software. Available at: https://app.diagrams.net/ (Accessed: 17 March 2024).

17. Wikipedia. Available at: https://www.wikipedia.org/ (Accessed: 17 March 2024).

18. *Google colab*. Available at: https://colab.research.google.com/ (Accessed: 17 March 2024).

19. *Coindesk: Bitcoin, Ethereum, crypto news and Price Data* (no date) *CoinDesk Latest Headlines RSS*. Available at: https://www.coindesk.com/ (Accessed: 15 May 2024).

20. O'Sullivan, J. *et al.* (no date) *Bitcoin, Ethereum, Crypto News & Price indexes, Cointelegraph*. Available at: https://cointelegraph.com/ (Accessed: 15 May 2024).

21. *Bitcoin.com News* (2023) *Bitcoin News*. Available at: https://news.bitcoin.com/ (Accessed: 15 May 2024).

22. *Bitcoin magazine - Bitcoin News, articles and expert insights*. Available at: https://bitcoinmagazine.com/ (Accessed: 15 May 2024).

23. *Bitcoin* (2024) *CNBC*. Available at: https://www.cnbc.com/bitcoin (Accessed: 15 May 2024).

Real-Time Data Streaming and Processing

ORIGINALITY REPORT

14%	12%	8%	%
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

PRIMARY SOURCES

1	www.ncbi.nlm.nih.gov Internet Source	1%
2	S Anveshrithaa, K Lavanya. "Real-Time Vehicle Traffic Analysis using Long Short Term Memory Networks in Apache Spark", 2020 International Conference on Emerging Trends in Information Technology and Engineering (ic-ETITE), 2020 Publication	1%
3	isset.shiksha.com Internet Source	1%
4	www.mdpi.com Internet Source	1%
5	doctorpenguin.com Internet Source	1%
6	www.irjmets.com Internet Source	1%
7	dokumen.pub Internet Source	1%

8	upcommons.upc.edu Internet Source	<1 %
9	www.researchgate.net Internet Source	<1 %
10	"Advances in Computational Intelligence", Springer Science and Business Media LLC, 2017 Publication	<1 %
11	aclanthology.org Internet Source	<1 %
12	kit.pnu.edu.ua Internet Source	<1 %
13	Kamal Al-Malah. "Machine and Deep Learning Using MATLAB", Wiley, 2023 Publication	<1 %
14	discovery.researcher.life Internet Source	<1 %
15	arxiv.org Internet Source	<1 %
16	sifisheriessciences.com Internet Source	<1 %
17	www.transparencymarketresearch.com Internet Source	<1 %
18	fastercapital.com Internet Source	<1 %

19	support.sas.com Internet Source	<1 %
20	Jongkook Kim. "Synthetic shear sonic log generation utilizing hybrid machine learning techniques", Artificial Intelligence in Geosciences, 2022 Publication	<1 %
21	wikimili.com Internet Source	<1 %
22	www.scielo.br Internet Source	<1 %
23	Yavuz Eren, İbrahim Küçükdemiral. "A comprehensive review on deep learning approaches for short-term load forecasting", Renewable and Sustainable Energy Reviews, 2024 Publication	<1 %
24	lutpub.lut.fi Internet Source	<1 %
25	ai-event.ted.com Internet Source	<1 %
26	www.geeksforgeeks.org Internet Source	<1 %
27	www.process.st Internet Source	<1 %

28	peerj.com Internet Source	<1 %
29	S Soumya, K V Pramod. "Sentiment Analysis of Malayalam Tweets using Different Deep Neural Network Models-Case Study", 2019 9th International Conference on Advances in Computing and Communication (ICACC), 2019 Publication	<1 %
30	"Genetic and Evolutionary Computing", Springer Science and Business Media LLC, 2024 Publication	<1 %
31	Ammar Odeh, Anas Abu Taleb. "Ensemble-Based Deep Learning Models for Enhancing IoT Intrusion Detection", Applied Sciences, 2023 Publication	<1 %
32	m.people.bayt.com Internet Source	<1 %
33	medium.com Internet Source	<1 %
34	pure.umsa.bo Internet Source	<1 %
35	researcher.manipal.edu Internet Source	<1 %

36	Patthamanan Isaranontakul, Worapoj Kreesuradej. "A Study of Using GPT-3 to Generate a Thai Sentiment Analysis of COVID-19 Tweets Dataset", 2023 20th International Joint Conference on Computer Science and Software Engineering (JCSSE), 2023 Publication	<1 %
37	link.springer.com Internet Source	<1 %
38	thesai.org Internet Source	<1 %
39	Zehuan Hu, Yuan Gao, Luning Sun, Masayuki Mae, Taiji Imaizumi. "Self-learning dynamic graph neural network with self-attention based on historical data and future data for multi-task multivariate residential air conditioning forecasting", Applied Energy, 2024 Publication	<1 %
40	researchrepository.wvu.edu Internet Source	<1 %
41	www.igi-global.com Internet Source	<1 %
42	finbold.com Internet Source	<1 %
43	llmrisks.github.io	

Internet Source

<1%

44 www.coursehero.com
Internet Source

<1%

45 www.ijipr.latticescipub.com
Internet Source

<1%

46 www.researchtrend.net
Internet Source

<1%

47 Fang Wu, Jigang Wang, Jiqiang Liu, Wei Wang. "Vulnerability detection with deep learning", 2017 3rd IEEE International Conference on Computer and Communications (ICCC), 2017
Publication

<1%

48 Jaydip Sen, Saikat Mondal, Sidra Mehtab. "Analysis of Sectoral Profitability of the Indian Stock Market Using an LSTM Regression Model", Institute of Electrical and Electronics Engineers (IEEE), 2021
Publication

<1%

49 Kitsuchart Pasupa, Thititorn Seneewong Na Ayutthaya. "Thai sentiment analysis with deep learning techniques: A comparative study based on word embedding, POS-tag, and sentic features", Sustainable Cities and Society, 2019
Publication

<1%

50	downloads.hindawi.com Internet Source	<1 %
51	dspace.lboro.ac.uk Internet Source	<1 %
52	fmgvbojb.pescherillo.it Internet Source	<1 %
53	sersc.org Internet Source	<1 %
54	www.ijisrt.com Internet Source	<1 %
55	Junhao Zhou, Yue Lu, Hong-Ning Dai, Hao Wang, Hong Xiao. "Sentiment Analysis of Chinese Microblog Based on Stacked Bidirectional LSTM", IEEE Access, 2019 Publication	<1 %
56	"Advances in Knowledge Discovery and Data Mining", Springer Science and Business Media LLC, 2019 Publication	<1 %
57	en.wikipedia.org Internet Source	<1 %
58	ir.juit.ac.in:8080 Internet Source	<1 %

JAYPEE UNIVERSITY OF INFORMATION TECHNOLOGY, WAKNAGHAT

PLAGIARISM VERIFICATION REPORT

Date: ....

Type of Document (Tick): PhD Thesis | M.Tech Dissertation/ Report | B.Tech Project Report | Paper

Name: ______________________________ **Department:** _________________ **Enrolment No** _________

Contact No. ______________________________ **E-mail.** _______________________________________

Name of the Supervisor: ____

Title of the Thesis/Dissertation/Project Report/Paper (In Capital letters): _________________________

____

____

UNDERTAKING

I undertake that I am aware of the plagiarism related norms/ regulations, if I found guilty of any plagiarism and copyright violations in the above thesis/report even after award of degree, the University reserves the rights to withdraw/revoke my degree/report. Kindly allow me to avail Plagiarism verification report for the document mentioned above.

Complete Thesis/Report Pages Detail:

- Total No. of Pages =
- Total No. of Preliminary pages =
- Total No. of pages accommodate bibliography/references =

(Signature of Student)

FOR DEPARTMENT USE

We have checked the thesis/report as per norms and found **Similarity Index** at.................... (%). Therefore, we
are forwarding the complete thesis/report for final plagiarism check. The plagiarism verification report may be handed over to the candidate.

(Signature of Guide/Supervisor) **Signature of HOD**

FOR LRC USE

The above document was scanned for plagiarism check. The outcome of the same is reported below:

Copy Received on	**Excluded**	**Similarity Index (%)**	**Generated Plagiarism Report Details** (Title, Abstract & Chapters)	
	• All Preliminary Pages • Bibliography/Images/Quotes • 14 Words String		Word Counts	
Report Generated on			Character Counts	
		Submission ID	Total Pages Scanned	
			File Size	

Checked by
Name & Signature **Librarian**

....

Please send your complete thesis/report in (PDF) with Title Page, Abstract and Chapters in (Word File) through the supervisor at plagcheck.juit@gmail.com