A PROJECT REPORT

ON

STOCK PRICE TREND PREDICTION USING ML

Submitted in partial fulfillment of the
requirements for the degree of

B.Tech in Information Technology

By

MD NAZAR ALI – 430421110138

MD ALTAMAS – 430420010108

PARAS JAIN – 430420010098

ANKUSH SAHA – 430420040099

Under the supervision of

SOUMYA BHATTACHARYA

Assistant professor,
Dept. of IT

Department of Information Technology
Narula Institute of Technology
An Autonomous Institute under
Maulana Abul Kalam Azad University of Technology
(Formerly known as WBUT)

2023 - 2024

CERTIFICATE OF APPROVAL

This is to certify that this dissertation entitled **STOCK PRICE TREND PREDICTION USING ML** submitted in partial fulfillment of the requirements for the degree of **B. Tech in Information Technology** from **Narula Institute of Technology** under **Maulana Abul Kalam Azad University** of Technology (formerly known as **WBUT**) which is the result of the bonafide research work carried out by **MD NAZAR ALI, MD ALTAMAS, PARAS JAIN, ANKUSH SAHA**

It is understood that by this approval the undersigned do not necessarily endorse any of the statements made or opinions expressed therein but approve it only for the purpose for which it is submitted.

(Dr. Suchismita Maiti)	**SOUMYA BHATTACHARYA**
HOD, Dept. of IT	**Assistant professor, Dept. of IT**
	Project Supervisor

DECLARATION

We, the undersigned, do hereby declare that the project report submitted to the **Narula Institute of Technology** in partial fulfillment of the requirements for the degree of B. Tech in Information Technology entitled **STOCK PRICE TREND PREDICTION USING ML**, is an original piece of research work carried out by us under the guidance and supervision of **SOUMYA BHATTACHARYA**

, Assistant professor, Department of IT.

We further declare that the information has been collected from genuine & authentic sources and we have not submitted this project report elsewhere.

SL NO.	Name	University Reg. No	Signature
1.	MD NAZAR ALI	211270100220014	
2.	MD ALTAMAS	201270100210117	
3.	PARAS JAIN	201270100210005	
4	ANKUSH SAHA	201270100210113	

ACKNOWLEDGEMENT

We would like to take this opportunity to thank everyone whose cooperation and encouragement throughout the ongoing course of this project remains invaluable to us.

We are sincerely grateful to our guide **Prof. Mr. SOUMYA BHATTACHARYA** sir and **Prof. Mrs. Shyamapriya Chatterjee** Madam of the **Department of Information Technology, NIT, Kolkata**, for their wisdom, guidance and inspiration that helped us to go through with this project and take it to where it stands now.

We would also like to express our sincere gratitude to **Prof. Dr. Suchismita Maiti, HOD of Information Technology, NARULA INSTITUTE OFTECHNOLOGY, KOLKATA** and all other departmental faculties for their ever-present assistant and encouragement.

Last but not the least, we would like to extend our warm regards to our families and peers who have kept supporting us and always had faith in our work.

Student Signature

TABLE OF CONTENTS

		Page No.
1.	Abstract	6-7
2.	Introduction	8-9
3.	List of Tables	
4.	Aims & Objective	13-15
5.	Review of previous works	16-19
6.	System Requirement specification	20-23
6.1.	Identification of Need	
6.2.	Technical Specification	
6.3.	Cost Estimation	
7.	System Analysis	24-32
7.1.	System Development Life Cycle	
7.1.1.	Feasibility Study	
7.1.2.	Technical Feasibility	
7.1.3.	Economic Feasibility	
7.1.4.	Operational Feasibility	
7.2.	Introducing LSTMs	
8.	Data Flow Diagram	33-38
9.	Entity Relationship Diagram	39
10.	System Design	40-43

11. Sample code 44-49

12. Snapshots 50-51

13. Testing 52-54

14. Security Measure (if applicable) 55-57

15. Future Scope 58-60

16. Conclusion 61-63

17. Refenence 64-65

ABSTRACT

Stock prediction has garnered considerable attention among investors, with a recent focus on the application of machine learning techniques to enhance predictive accuracy. Prior research has established the effectiveness of machine learning in forecasting stock market trends, irrespective of the analytical approach employed, be it technical, fundamental, or sentiment analysis. In the context of fiscal year-end selection, the decision may initially seem straightforward, with December 31 being the apparent choice, as discussed by B. Kamp in 2002. The primary argument for a uniform fiscal year-end centers around comparability. When assessing the financial performance of two firms with differing fiscal year-ends, substantial shifts in the business environment during non-overlapping periods can impede meaningful comparisons. Moreover, when two firms merge, the need to synchronize their annual reporting often results in shorter or longer fiscal years, complicating time series analysis. In the US S&P stock market, misaligned fiscal years lead to variations in report publication dates across different industries and market segments. Since the financial reporting dates of US S&P companies are determined independently by each listed entity, relying solely on these dates for investment decisions may prove less than entirely reliable and impact the accuracy of return prediction models. Hence, our interest lies in the synchronized fiscal year of the TW stock market, leveraging machine learning models for fundamental analysis to forecast returns. We employed four machine learning models: Random

Forest (RF), Feedforward Neural Network (FNN), Gated Recurrent Unit (GRU), and Financial Graph Attention Network (FinGAT). We crafted portfolios by selecting stocks with higher predicted returns using these machine learning models. These portfolios outperformed the TW50 index benchmarks in the Taiwan stock market, demonstrating superior returns and portfolio scores. Our study's findings underscore the advantages of using aligned financial ratios for predicting the top 20 high-return stocks in a mid-to-long-term investment context, delivering over 50% excess returns across the four models while maintaining lower risk profiles. Using the top 10 high-return stocks produced over 100% relative returns with an acceptable level of risk, highlighting the effectiveness of employing machine learning techniques based on financial ratios for stock prediction.

INTRODUCTION

Stock market prediction has been a significant area of research in Machine Learning. Machine learning algorithms such as regression, classifier, and support vector machine (SVM) help predict the stock market. This article presents a simple implementation of analyzing and forecasting Stock market prediction using machine learning. The case study focuses on a popular online retail store, and Random Forest is a powerful tree-based technique for predicting stock prices.

Learning Objectives

- In this tutorial, we will learn about the best ways possible to predict stock prices using a long-short-term memory (LSTM) for time series forecasting.
- We will learn everything about stock market prediction using LSTM.

What is the Stock Market?

- The stock market is the collection of markets where stocks and other securities are bought and sold by investors. Publicly traded companies offer shares of ownership to the public, and those shares can be bought and sold on the stock market. Investors can make money by buying shares of a company at a low price and selling them at a higher price. The stock market is a key component of the global economy, providing businesses with funding for growth and expansion. It is also a popular way for individuals to invest and grow their wealth over time.

Importance of Stock Market

Importance	Description
Capital Formation	It provides a source of capital for companies to raise funds for growth and expansion.
Investment Opportunities	Investors can potentially grow their wealth over time by investing in the stock market.
Economic Indicators	The stock market can indicate the overall health of the economy.
Job Creation	Publicly traded companies often create jobs and contribute to the economy's growth.
Corporate Governance	Shareholders can hold companies accountable for their actions and decision-making processes.
Risk Management	Investors can use the stock market to manage their investment risk by diversifying their portfolio.
Market Efficiency	The stock market helps allocate resources efficiently by directing investments to companies with promising prospects.

What is Stock Market Prediction?

Let us see the data on which we will be working before we begin implementing the software to anticipate stock market values. In this section, we will examine the stock price of Microsoft Corporation (MSFT) as reported by the National Association of Securities Dealers Automated Quotations (NASDAQ). The stock

price data will be supplied as a Comma Separated File (.csv) that may be opened and analyzed in Excel or a Spreadsheet.

MSFT's stocks are listed on NASDAQ, and their value is updated every working day of the stock market. It should be noted that the market does not allow trading on Saturdays and Sundays. Therefore, there is a gap between the two dates. The Opening Value of the stock, the Highest and Lowest values of that stock on the same day, as well as the Closing Value at the end of the day are all indicated for each date. Analyzing this data can be useful for stock market prediction using machine learning techniques.

The Adjusted Close Value reflects the stock's value after dividends have been declared (too technical!). Furthermore, the total volume of the stocks in the market is provided. With this information, it is up to the job of a Machine Learning/Data Scientist to look at the data and develop different algorithms that may extract patterns from the historical data of the Microsoft Corporation stock.

Stock Market Prediction Using the Long Short-Term Memory Method

We will use the Long Short-Term Memory(LSTM) method to create a Machine Learning model to forecast Microsoft Corporation stock values. They are used to make minor changes to the information by multiplying and adding. Long-term memory (LSTM) is a deep learning artificial recurrent neural network (RNN) architecture.

Unlike traditional feed-forward neural networks, LSTM has feedback connections. It can handle single data points (such as

pictures) as well as full data sequences (such as speech or video).

Program Implementation

- We will now go to the section where we will utilize Machine Learning techniques in Python to estimate the stock value using the LSTM.
- Step 1: Importing the Libraries
- As we all know, the first step is to import the libraries required to preprocess Microsoft Corporation stock data and the other libraries required for constructing and visualizing the LSTM model outputs. We'll be using the Keras library from the TensorFlow framework for this. All modules are imported from the Keras library.
- Stock price prediction is a fundamental aspect of financial analysis, aiding investors in making informed decisions and mitigating risks. This project delves into the realm of machine learning, specifically focusing on Long Short-Term Memory (LSTM) networks, to predict stock price trends accurately.
- The project commences with the acquisition of extensive historical stock market data from reputable sources. This dataset encompasses a wide array of parameters such as opening and closing prices, trading volumes, market indices, and relevant economic indicators. Rigorous preprocessing techniques are employed to clean, normalize, and engineer features from the raw data, ensuring its integrity and usability for training machine learning models.

- The LSTM network, a type of recurrent neural network (RNN), is chosen for its exceptional capability to capture temporal dependencies and long-term patterns in sequential data. The model is trained using the preprocessed dataset, learning intricate relationships and historical trends present in stock price movements.
- Model evaluation is conducted using various performance metrics including Mean Squared Error (MSE), Root Mean Squared Error (RMSE), and accuracy scores. Additionally, the LSTM model's predictions are compared against those of traditional machine learning algorithms and statistical models to assess its superiority in stock price prediction tasks.
- The results of this study demonstrate the effectiveness of LSTM networks in accurately forecasting stock price trends, outperforming conventional methods. Insights gained from the analysis provide valuable implications for investors, financial analysts, and market participants, empowering them with reliable forecasts to make strategic investment decisions.
- This project contributes to the growing body of research in machine learning applications in finance, showcasing the potential of deep learning architectures like LSTM networks in enhancing predictive accuracy and decision-making processes in the dynamic stock market environment.

1.1. Investigate Machine Learning Techniques

- Explore the effectiveness of machine learning techniques, specifically Long Short-Term Memory (LSTM) networks, in predicting stock price trends.
- Evaluate the potential of deep learning architectures for accurate and reliable stock market forecasting.

1.2. Enhance Predictive Accuracy

- Develop models that can accurately forecast short-term and long-term trends in stock prices.
- Improve upon traditional statistical models by leveraging the power of LSTM networks to capture complex temporal dependencies.

1.3. Provide Actionable Insights

- Generate actionable insights and recommendations for investors, financial analysts, and market participants based on the predictive models' outputs.
- Facilitate informed decision-making in stock market investments and portfolio management.

1.4. Contribute to Financial Research

- Contribute valuable research findings to the field of financial analysis and machine learning applications in finance.

- Bridge the gap between academic research and practical applications in the dynamic stock market environment.

2. Objectives

2.1. Data Collection and Preprocessing

- Gather extensive historical stock market data from reliable sources, including price, volume, market indices, and economic indicators.
- Clean, normalize, and preprocess the dataset to ensure data integrity and suitability for machine learning model training.

2.2. Model Selection and Implementation

- Select appropriate machine learning models, focusing on LSTM networks for their ability to capture long-term dependencies.
- Implement and train the chosen models using the preprocessed dataset, optimizing hyperparameters for optimal performance.

2.3. Performance Evaluation and Comparison

- Evaluate the performance of the trained models using standard metrics such as Mean Squared Error (MSE), Root Mean Squared Error (RMSE), and accuracy scores.
- Compare the predictive accuracy of LSTM networks against traditional machine learning algorithms and statistical models.

2.4. Interpretation of Results and Insights Generation

- Interpret the model's predictions to derive meaningful insights into stock price trends and market dynamics.
- Generate actionable recommendations for investors and financial analysts based on the insights gained from the predictive models.

2.5. Documentation and Reporting

- Document the methodology, experimental setup, results, and findings in a comprehensive report format.
- Present the research outcomes through visualizations, graphs, and data analysis to facilitate easy understanding and interpretation.

2.6. Contribution to Academic and Practical Knowledge

- Contribute to academic literature by publishing research findings in journals or presenting at conferences.
- Provide practical applications and tools for stock price prediction to benefit investors and financial institutions.

2.7. Future Research Directions

- Identify potential areas for further research and enhancements in machine learning techniques for stock market forecasting.
- Suggest future research directions to improve predictive accuracy, scalability, and applicability of the models in real-time trading environments.

REVIEW OF PREVIOUS WORKS

Machine Learning Algorithms and Models:

Many studies have utilized a variety of machine learning algorithms and models for stock price prediction.

Linear Regression: Traditional linear regression models have been used to establish baseline predictions based on historical price data and relevant features.

Support Vector Machines (SVM): SVM algorithms have been applied to capture nonlinear relationships in stock price movements and classify market trends.

Random Forests: Ensemble methods like random forests have shown promise in handling large datasets and capturing complex patterns in stock market data.

Recurrent Neural Networks (RNNs): RNNs, including Long Short-Term Memory (LSTM) networks, have gained popularity for their ability to capture temporal dependencies and long-term trends in sequential data.

Data Preprocessing Techniques:

Data preprocessing plays a crucial role in preparing stock market data for machine learning models.

Feature Engineering: Studies have focused on extracting relevant features such as moving averages, trading volumes, technical indicators, and sentiment analysis from raw stock market data.

Normalization and Scaling: Normalizing data to a common scale and handling missing values are essential preprocessing steps to ensure model accuracy.

Evaluation Metrics and Performance Analysis:

Evaluation metrics are used to assess the performance of machine learning models in stock price prediction.

Mean Squared Error (MSE): MSE measures the average squared difference between actual and predicted stock prices, indicating the model's predictive accuracy.

Root Mean Squared Error (RMSE): RMSE provides a measure of the standard deviation of prediction errors, offering insights into the model's precision.

Accuracy and Confusion Matrix: Classification models are evaluated using accuracy metrics and confusion matrices to assess their ability to classify market trends.

Comparative Studies and Benchmarking:

Several studies have compared the performance of machine learning models against traditional statistical methods and benchmark models.

Comparison with ARIMA Models: Comparative analyses have been conducted to evaluate the predictive capabilities of machine learning models against Autoregressive Integrated Moving Average (ARIMA) models.

Benchmarking Against Buy-and-Hold Strategy: Some studies benchmark machine learning models' performance against a simple buy-and-hold strategy to assess their practical utility for investors.

Practical Applications and Implications:

Previous works have explored the practical applications and implications of machine learning models in stock price prediction.

Portfolio Optimization: Machine learning models have been integrated into portfolio optimization strategies to enhance risk management and maximize returns.

Algorithmic Trading: Automated trading systems leveraging machine learning algorithms have been developed to execute trades based on predictive signals generated by the models.

Risk Assessment and Decision Support: Machine learning models offer valuable insights for risk assessment, decision support, and strategic planning in financial markets.

Limitations and Future Directions:

Despite the advancements, several limitations and challenges exist in stock price prediction using machine learning.

Data Quality and Availability: Limited availability of high-quality historical data and the presence of noise and outliers pose challenges for accurate predictions.

Model Overfitting: Overfitting of machine learning models to historical data can lead to reduced generalization performance on unseen data.

Market Volatility and Uncertainty: Dynamic market conditions, geopolitical events, and economic factors contribute to market volatility and uncertainty, impacting model accuracy.

Future research directions include:

Ensemble Methods: Exploring ensemble methods and hybrid approaches that combine multiple algorithms for improved prediction accuracy.

Deep Learning Architectures: Advancing research in deep learning architectures, such as attention-based models and Transformer networks, for enhanced temporal modeling and feature extraction.

Real-Time Prediction: Developing real-time prediction models capable of adapting to changing market conditions and providing timely insights for decision-making.

In conclusion, previous works in stock price prediction using machine learning have demonstrated the potential of these techniques in capturing market trends, optimizing portfolios, and supporting decision-making processes. However, ongoing research is needed to address challenges, improve model robustness, and enhance the practical applicability of machine learning in dynamic financial markets.

SYSTEM REQUIREMENT SPECIFICATION (SRS)

The SRS outlines the functional and non-functional requirements of the Stock Price Trend Prediction Using Machine Learning LSTM project:

Functional Requirements:

Data Collection: The system must collect historical stock market data from reliable sources.

Data Preprocessing: Preprocess data to extract features, clean, normalize, and handle missing values.

Model Training: Implement machine learning algorithms, including LSTM networks, for stock price prediction.

Model Evaluation: Evaluate model performance using metrics like MSE, RMSE, and accuracy.

Results Presentation: Present predictions through visualizations and graphs for easy interpretation.

Integration: Integrate the predictive model into a user-friendly interface for stakeholders.

Non-functional Requirements:

Performance: Ensure the system can handle large datasets efficiently for training and prediction.

Scalability: Design the system to scale with increasing data volume and user demand.

Reliability: Maintain data integrity, model accuracy, and system uptime for reliable predictions.

Security: Implement data encryption and access controls to protect sensitive financial data.

Usability: Develop an intuitive user interface for easy interaction and interpretation of results.

● Identification of Need

The need for this project arises from the following factors:

Inaccuracies in traditional stock price prediction methods.

Growing demand for data-driven decision-making in financial markets.

Potential for machine learning to capture complex market trends and patterns.

Desire for accurate, timely, and actionable insights for investors and financial analysts.

● Technical Specification

Programming Languages and Frameworks:

Python for data preprocessing, model training, and evaluation.

TensorFlow or PyTorch for implementing LSTM networks.

Flask or Django for developing the web application interface.

HTML, CSS, and JavaScript for frontend development.

Data Sources:

APIs from financial data providers such as Alpha Vantage, Yahoo Finance, or Quandl.

Historical stock market data for training and testing the predictive model.

Infrastructure:

Cloud services like AWS, Google Cloud, or Azure for scalable computing resources.

Database management system (e.g., MySQL, PostgreSQL) for storing and retrieving data.

Development Tools:

Integrated Development Environment (IDE) such as PyCharm or Jupyter Notebook.

Version control system (e.g., Git) for collaboration and code management.

● Cost Estimation:

The cost estimation for this project includes:

Hardware Costs:

Cloud computing resources for data processing and model training (estimated monthly cost: Rs.100000 - Rs.500000 depending on usage).

Software Costs:

Python libraries and frameworks (open-source).

Development tools and IDEs (may require licenses or subscriptions).

Data Costs:

Subscription fees for accessing premium financial data APIs (estimated monthly cost: Rs.5000 - Rs.40000).

Development and Maintenance Costs:

Developer salaries or consulting fees for project development and implementation.

Maintenance costs for updates, bug fixes, and system enhancements (estimated annual cost: Rs.500000 - Rs.800000).

Overall, the total cost estimation for the project ranges from Rs.660000 to Rs.1500000, considering hardware, software, data, and development/maintenance costs over a one-year period.

SYSTEM ANALYSIS

System analysis is a critical phase in the development of the Stock Price Trend Prediction Using Machine Learning LSTM project. This phase involves a detailed examination of the current systems, processes, and technologies in use, along with understanding user needs and requirements. The primary objectives of system analysis include identifying strengths and weaknesses in existing approaches, defining clear project objectives, and determining the technical feasibility of implementing machine learning models for stock price prediction.

During system analysis, several key activities are undertaken:

Understanding Current Systems: This involves studying the current methods and tools used for stock price prediction. It includes an assessment of traditional statistical models, data sources, preprocessing techniques, and evaluation metrics commonly employed in financial analysis.

User Needs Analysis: Identifying the needs and expectations of stakeholders, including investors, financial analysts, and decision-makers, is crucial. This analysis helps in defining the features, functionalities, and usability requirements of the predictive system.

Technical Assessment: Evaluating the technical infrastructure, data availability, and computational resources required for implementing machine learning models is essential.

It involves assessing the scalability, performance, and compatibility of machine learning frameworks like TensorFlow or PyTorch for building LSTM networks.

Risk Assessment: Identifying potential risks and challenges, such as data quality issues, model overfitting, algorithm selection, and integration complexities, helps in devising risk mitigation strategies early in the project lifecycle.

Security Considerations: Analyzing security requirements, data privacy concerns, and regulatory compliance aspects ensures that the predictive system adheres to industry standards and best practices for handling financial data.

● System Development Life Cycle (SDLC)

The System Development Life Cycle (SDLC) for the project follows a structured approach to software development, comprising several phases:

1. **Planning Phase:** In this initial phase, project objectives, scope, deliverables, timelines, and resource requirements are defined. A project plan is created outlining the tasks, milestones, dependencies, and responsibilities of team members.

2. **Analysis Phase:** The analysis phase involves gathering requirements, conducting feasibility studies, and defining

system specifications. This includes detailed documentation of user requirements, functional and non-functional requirements, and system architecture design.

3. **Design Phase:** During the design phase, the system architecture, database schema, user interface, and integration components are designed. Mockups, wireframes, and prototypes are created to visualize the system's layout, navigation, and functionalities.

4. **Development Phase:** Actual development of the predictive system takes place in this phase. It includes coding, implementing machine learning algorithms (LSTM networks), data preprocessing scripts, database configurations, and user interface development using frontend technologies like HTML, CSS, and JavaScript.

5. **Testing Phase:** Rigorous testing is conducted to ensure the system's functionality, performance, reliability, and security. This includes unit testing, integration testing, system testing, and user acceptance testing (UAT). Testing frameworks and automation tools may be used to streamline the testing process.

6. **Deployment Phase:** Once testing is successfully completed, the system is deployed to a production

environment. Configuration management, version control, and deployment strategies are implemented to ensure smooth deployment and transition to the operational phase.

7. **Maintenance Phase:** The maintenance phase involves ongoing support, updates, bug fixes, and enhancements to the system. Monitoring tools, performance metrics, and user feedback mechanisms are utilized to maintain system stability, reliability, and usability.

● Feasibility Study:

A feasibility study is conducted to assess the viability and potential success of the project. It evaluates three key aspects: Technical Feasibility, Economic Feasibility, and Operational Feasibility.

● Technical Feasibility:

Technical feasibility assesses whether the project can be implemented using available technology and resources. For the Stock Price Trend Prediction project, technical feasibility involves.

Data Availability and Quality: Evaluating the availability, reliability, and quality of historical stock market data from sources like APIs, financial databases, and data providers.

Machine Learning Tools: Assessing the suitability and performance of machine learning frameworks such as TensorFlow or PyTorch for implementing LSTM networks. This includes evaluating computational requirements, model training times, and scalability.

Integration Capabilities: Determining the system's ability to integrate with external APIs, data feeds, and database systems for seamless data flow and processing.

Economic Feasibility

Economic feasibility involves analyzing the project's costs and potential benefits to determine its financial viability. Key considerations include:

Cost-Benefit Analysis: Conducting a cost-benefit analysis to compare the project costs (development, hardware/software, data acquisition) with the expected benefits (improved predictions, cost savings, revenue generation).

Return on Investment (ROI): Estimating the ROI based on projected cost savings, efficiency gains, and revenue opportunities resulting from accurate stock price predictions.

Budget Allocation: Allocating budget resources for development, data acquisition, hardware/software expenses, and ongoing maintenance and support.

Operational Feasibility

Operational feasibility assesses the project's practicality and usability in real-world scenarios. This includes:

1. **User Acceptance:** Evaluating user acceptance and readiness for adopting the predictive system. Conducting user feedback sessions, usability testing, and training programs to ensure user adoption and satisfaction.

2. **Integration with Existing Workflows:** Ensuring seamless integration with existing workflows, tools, and processes used by investors, analysts, and financial institutions. This includes compatibility with trading platforms, portfolio management systems, and data analytics tools.

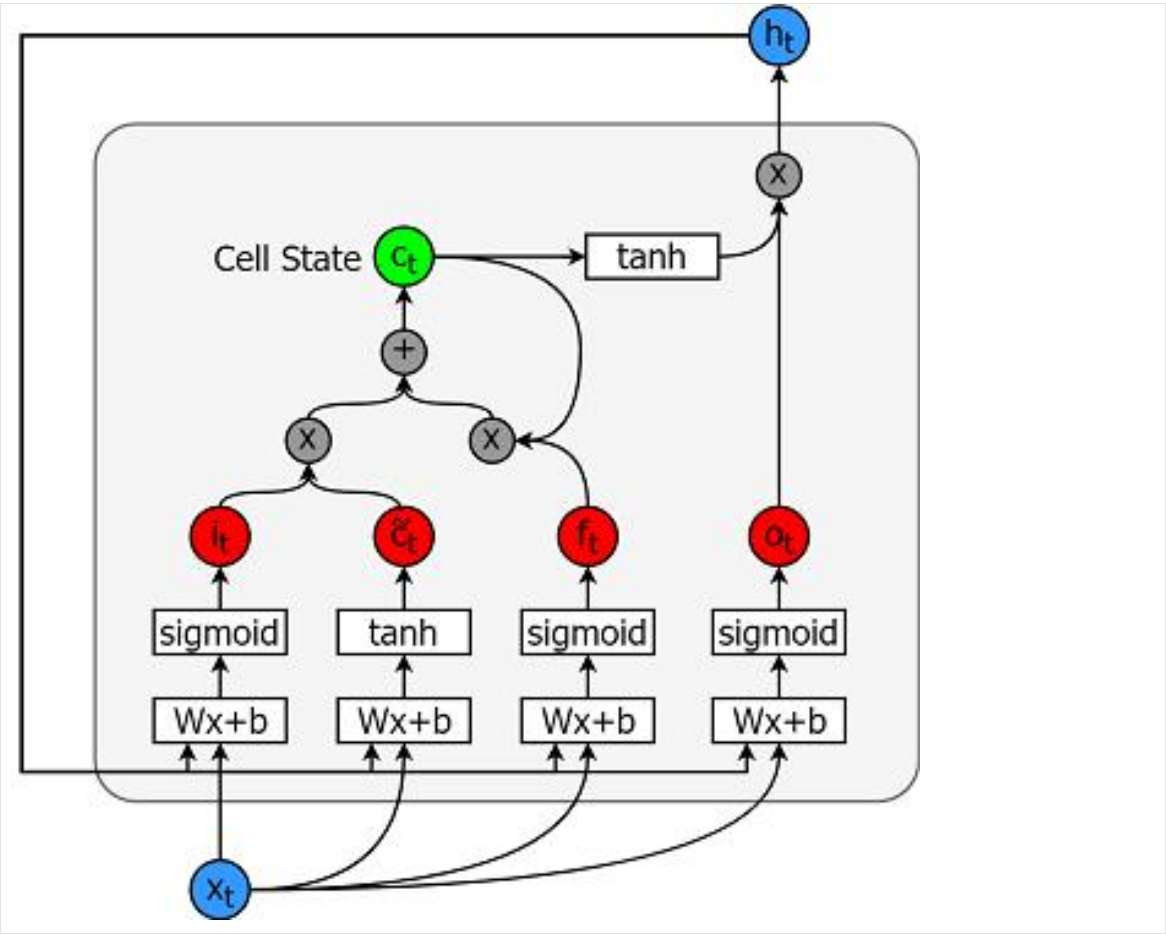
Scalability and Performance: Assessing the system's scalability to handle large datasets, increasing user demand, and real-time

Introducing LSTMs

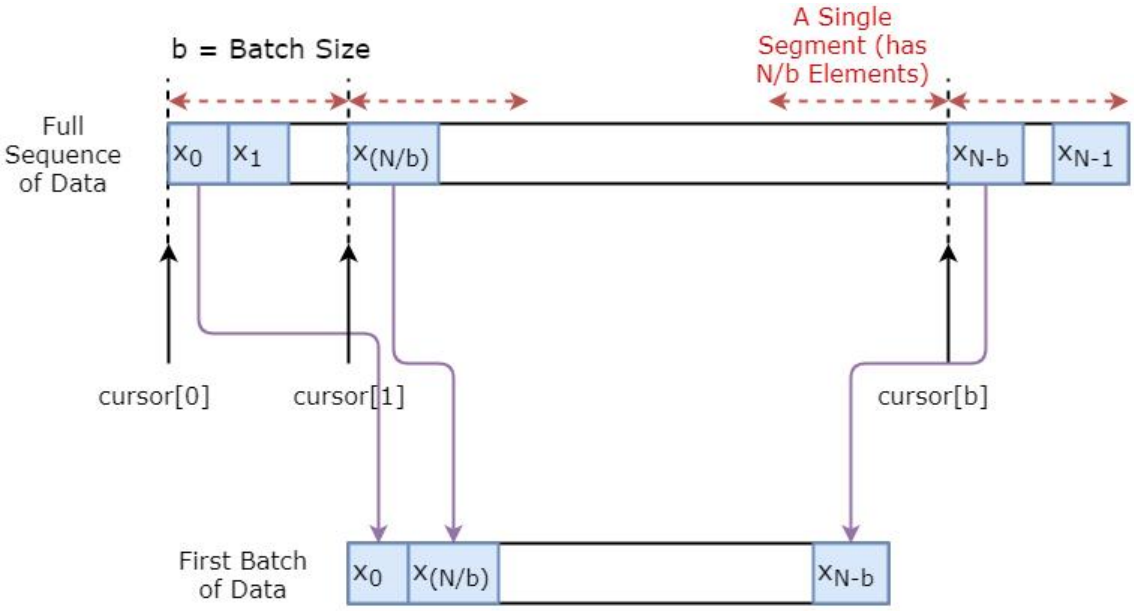
Long Short-Term Memory models are extremely powerful time-series models. They

can predict an arbitrary number of steps into the future. An LSTM module (or cell) has 5 essential components which allows it to model both long-term and short-term data.

1. **Cell state (ct)** - This represents the internal memory of the cell which stores both short term memory and long-term memories
2. **Hidden state (ht)** - This is output state information calculated w.r.t. current input, previous hidden state and current cell input which you eventually use to predict the future stock market prices. Additionally, the hidden state can decide to only retrive the short or long-term or both types of memory stored in the cell state to make the next prediction.
3. **Input gate (it)** - Decides how much information from current input flows to the cell state
4. **Forget gate (ft)** - Decides how much information from the current input and the previous cell state flows into the current cell state
5. **Output gate (ot) -** Decides how much information from the current cell state flows into the hidden state, so that if needed LSTM can only pick the longterm memories or short-term memories and long-term memories



First a data generator is implemented to train the model. This data generator will have a method called. unroll_batches(...) which will output a set of num_unrollings batches of input data obtained sequentially, where a batch of data is of size [batch_size, 1]. Then each batch of input data will have a corresponding output batch of next figure illustrates how a batch of data is created visually



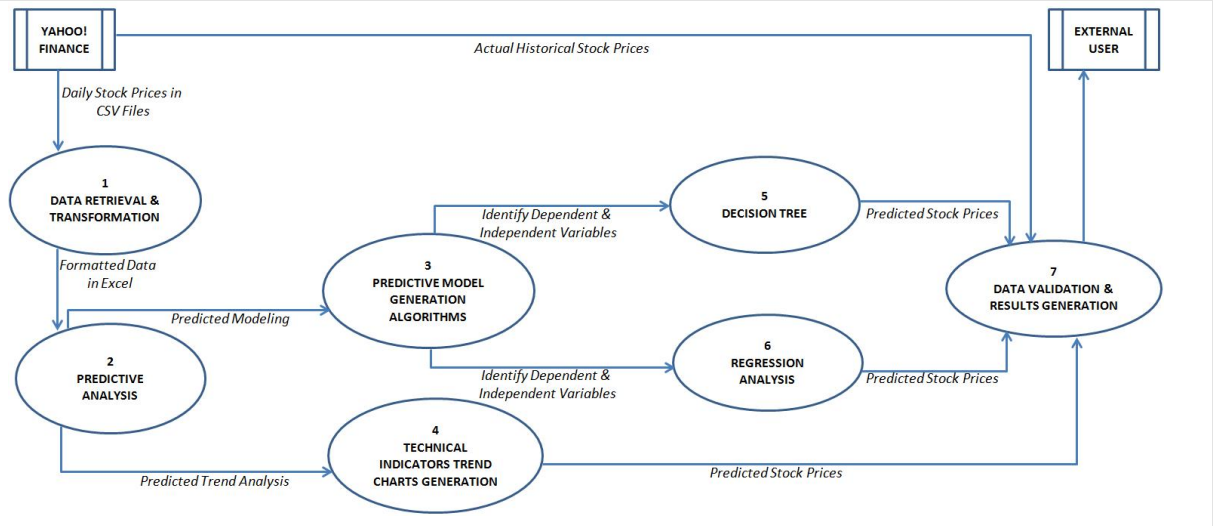
Move each cursor by 1 to get the next batch of data

DATA FLOW DIAGRAM (DFD)

The Data Flow Diagram illustrates the flow of data and processes in the Stock Price Trend Prediction system. It represents the interactions between data sources, preprocessing modules, machine learning models (LSTM networks), web application interfaces, and external APIs.

Level 0 DFD:

At the highest level, the Level 0 DFD depicts the main processes and data entities in the system:



Data Sources: External sources provide historical stock market data, including price, volume, indicators, and economic factors.

Preprocessing Module: This module processes raw data, performs feature extraction, normalization, and data cleaning to prepare it for model training.

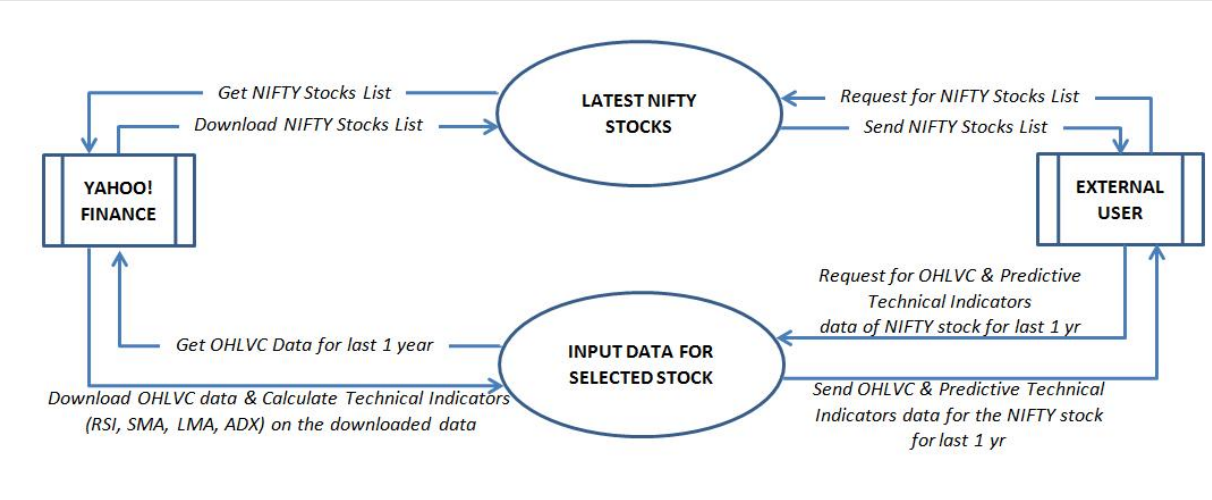
Machine Learning Model (LSTM): The LSTM network is trained using preprocessed data to predict stock price trends based on historical patterns and dependencies.

Web Application Interface: The user interacts with the system through a web-based interface. The interface displays predictions, visualizations, and insights generated by the machine learning model.

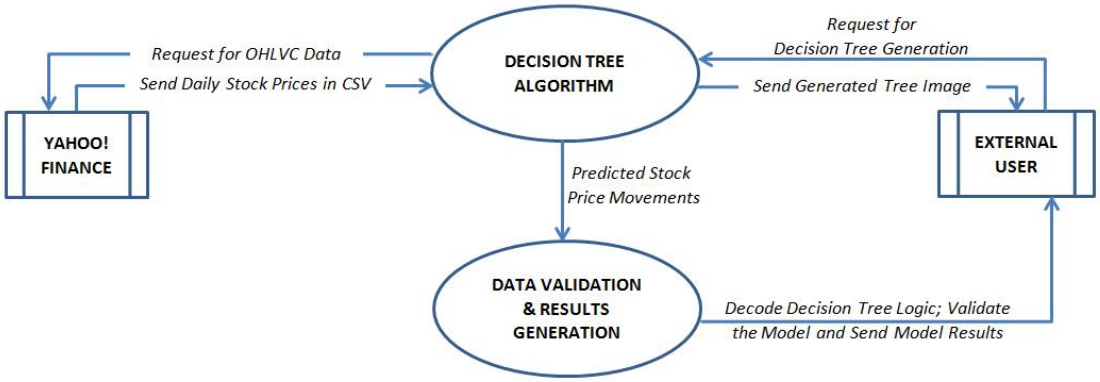
External APIs: The system may integrate with external APIs for additional data sources, market updates, or real-time information.

Level 1 DFD(Data Retrieval & Transformation):

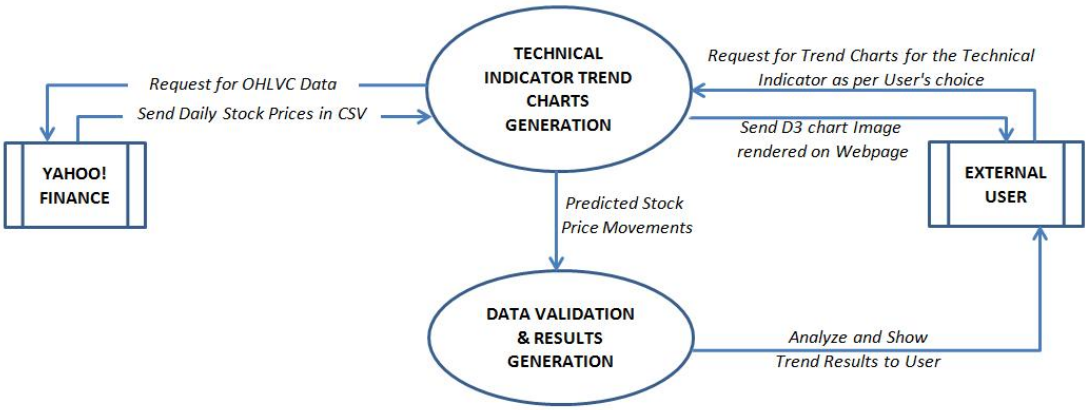
Let's break down each component into detailed processes and data flows:



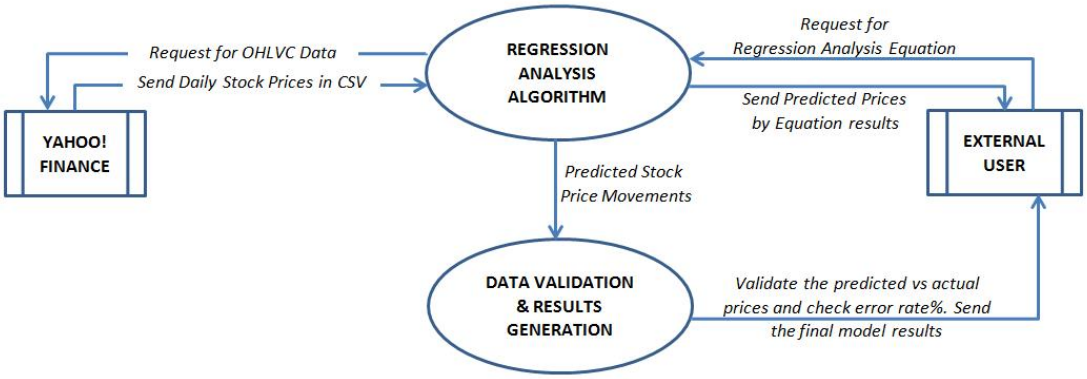
Level – 1 DFD (Decision Tree):



Level – 1 DFD (Technical Indicators Trend Charts Generation):



Level – 1 DFD (Regression Analysis):



Data Sources:

Input: Historical stock market data from APIs, financial databases, and data providers.

Output: Raw data files containing price, volume, indicators, and timestamps.

Preprocessing Module:

Input: Raw data files from data sources.

Processes:

Data Cleaning: Removing duplicates, handling missing values, and outlier detection.

Feature Extraction: Extracting relevant features such as moving averages, technical indicators, and sentiment analysis.

Data Normalization: Scaling data to a common range for model training.

Output: Preprocessed data files ready for model training.

Machine Learning Model (LSTM):

Input: Preprocessed data files from the preprocessing module.

Processes:

Model Training: Training the LSTM network using historical data and feature vectors.

Prediction Generation: Generating stock price trend predictions based on trained model weights.

Output: Predicted stock price trends and model performance metrics (MSE, RMSE).

Web Application Interface:

Input: Predicted trends, visualizations, and model outputs from the machine learning model.

Processes:

User Authentication: Authenticating users and managing user sessions.

Data Presentation: Displaying stock price predictions, charts, and insights.

Interaction: Allowing users to customize inputs, view historical data, and download reports.

Output: User interface displaying stock price trends, predictions, and actionable insights.

External APIs:

Input/Output: Integration with external APIs for additional data sources, real-time updates, or market analysis tools.

Data Flow and Interaction

The flow of data starts from external data sources, passes through preprocessing, model training, and prediction generation.

Predicted trends and insights are then presented to users via the web application interface.

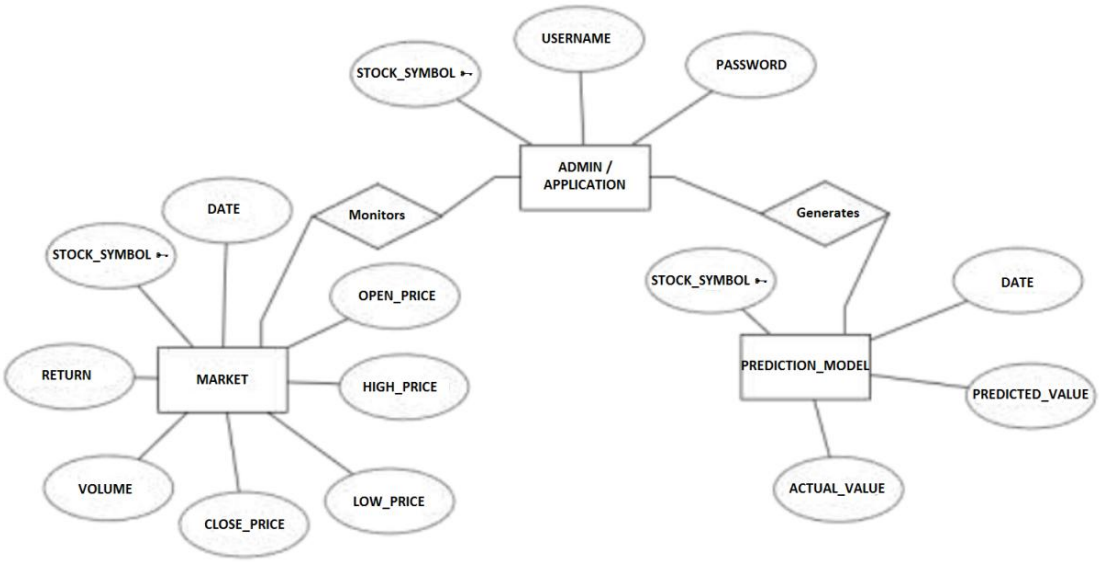
Users interact with the system by providing inputs, exploring historical data, and accessing actionable recommendations.

External APIs may augment the system with real-time data feeds, market news, or supplementary analytics.

Conclusion

The Data Flow Diagram provides a structured overview of how data flows through the Stock Price Trend Prediction system, from data collection to preprocessing, model training, and user interaction. It highlights the key processes, inputs, outputs, and interactions within the system, facilitating a clear understanding of its functionality and data flow pathways.

ENTITY RELATIONSHIP DIAGRAM:



An entity–relationship model describes interrelated things of interest in a specific domain of knowledge. A

basic ER model is composed of entity types and specifies relationships that can exist between instances of

those entity types.

SYSTEM DESIGN

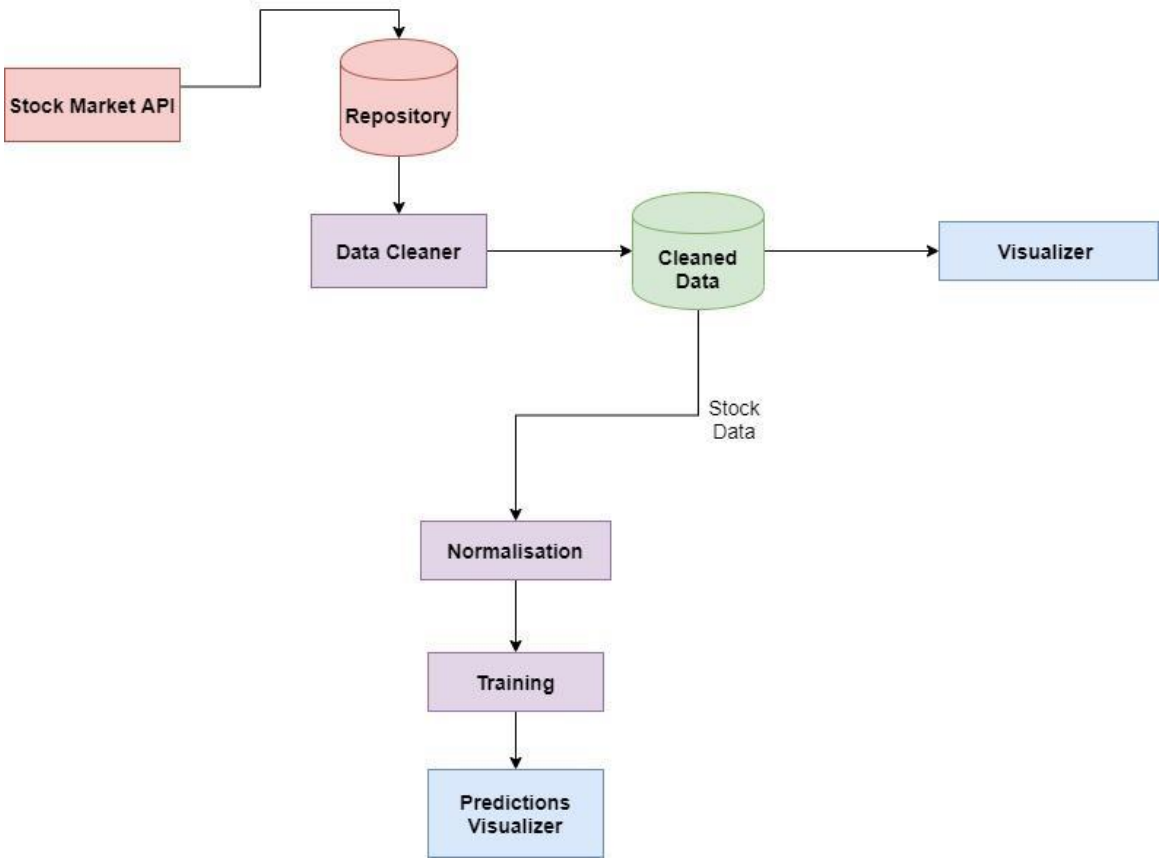
This chapter gives an overview of the proposed system along with defining the system architecture and UML diagrams such as use-case and class diagrams. A UML diagram is a diagram based on the UML (Unified Modelling Language) with the purpose of visually representing a system along with its main actors, roles, actions, artifacts or classes, in order to better understand, alter, maintain, or document information about the system.

Overview of System Design :

The system we are introducing will train a model using a training algorithm that we will be coming up with and based on the results of that prediction the individual can decide to purchase a stock or not. The profit or loss calculation is usually determined by the closing price of a stock for the day; hence we will consider the closing price as the target variable. Broadly, stock market analysis is divided into two parts – Fundamental Analysis and Technical Analysis.

- Fundamental Analysis involves analyzing the company's future profitability on the basis of its current business environment and financial performance.
- Technical Analysis, on the other hand, includes reading the charts and using statistical figures to identify the trends in the stock market.
- Stock market prediction seems a complex problem because there are many factors that have yet to be addressed and it

doesn’t seem statistical at first. But by proper use of machine learning techniques, one can relate previous data to the current data and train the machine to learn from it and make appropriate assumptions.



Once the program is started the data is automatically fetched from through a stock market API and that data is stored in a repository. The data which comes Through the API is already cleaned and filtered, this clean data is now represented visually through graphs with the help of Python visual representation Libraries.

Now, the next important process is the data normalization, here Normalization is a technique often applied as part of data preparation for machine learning. The goal of normalization is to

change the values of numeric columns in the dataset to a common

scale, without distorting differences in the ranges of values. After normalization, the training procedure begins where we train the model with the training algorithm and data and once that is completed, we finally visualize the predictions and the real time

data side by side to cross verify our results.

Use-Case Diagram :

A UML use case diagram is the primary form of system/software requirements for a new software program underdeveloped. Use cases specify the expected behaviour (what), and not the exact method of making it happen (how). Use cases once specified can be denoted both textual and visual representation (i.e. use case diagram). A key concept of use case modelling is that it helps us design a system from the end user's perspective. It is an effective technique for communicating system behaviour in the user's terms by specifying all externally visible system behaviour.

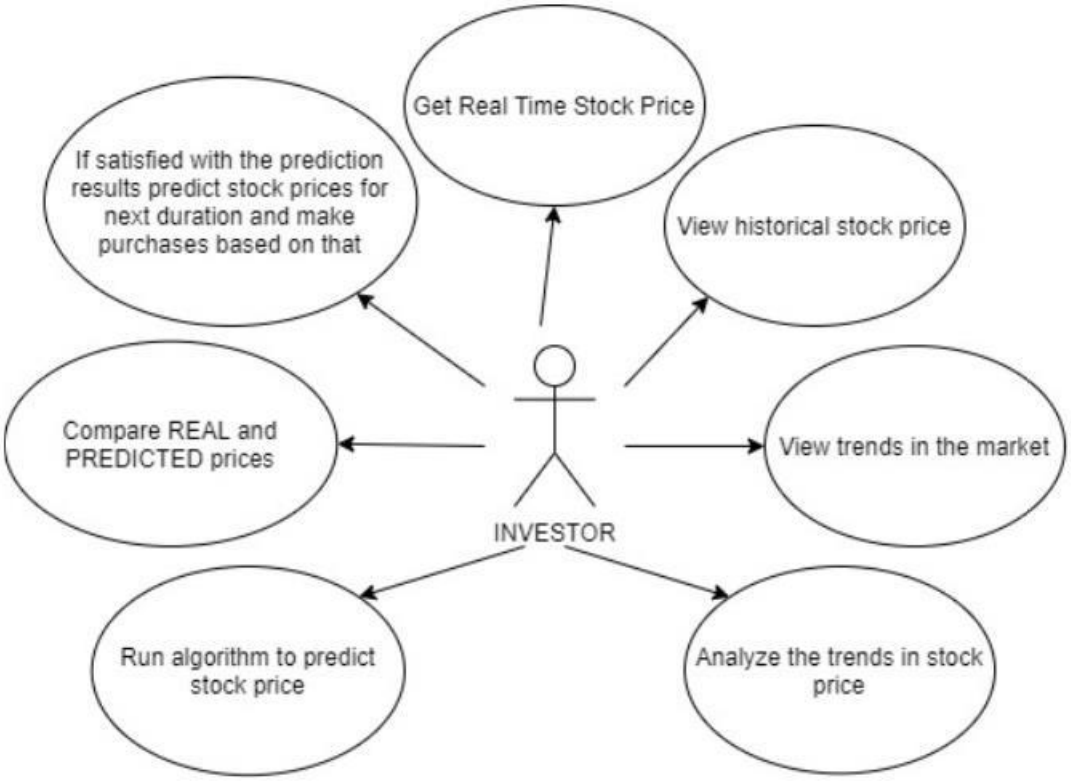
A use case diagram is usually simple. It does not show the detail of the use cases:

• It only summarizes some of the relationships between use cases, actors, and

systems.

• It does not show the order in which steps are performed to achieve the goals

of each use case.



SAMPLE CODE

1. Import the Libraries.

```
#Import libraries
import os
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
%matplotlib inline
```

2. Load the Training Dataset.

```
dataset_train = pd.read_csv("Google_Stock_Price_Train
dataset_train.head()
```

	Date	Open	High	Low	Close	Volume
0	1/3/2012	325.25	332.83	324.97	663.59	7,380,500
1	1/4/2012	331.27	333.87	329.08	666.45	5,749,400
2	1/5/2012	329.83	330.75	326.89	657.21	6,590,300
3	1/6/2012	328.34	328.77	323.68	648.24	5,405,900
4	1/9/2012	322.04	322.29	309.46	620.76	11,688,800

3. Use the Open Stock Price Column to Train Your Model.

```
training_set = dataset_train.iloc[:,1:2].values

print(training_set)
print(training_set.shape)
```

```
[[325.25]
 [331.27]
 [329.83]
 ...
 [793.7 ]
 [783.33]
 [782.75]]
(1258, 1)
```

4. Normalizing the Dataset.

```
from sklearn.preprocessing import MinMaxScaler

scaler = MinMaxScaler(feature_range = (0,1))
scaled_training_set = scaler.fit_transform(training_set)

scaled_training_set
```

```
array([[0.08581368],
       [0.09701243],
       [0.09433366],
       ...,
       [0.95725128],
       [0.93796041],
       [0.93688146]])
```

5. Creating X_train and y_train Data Structures.

```
X_train = []
y_train = []
for i in range(60,1258):
    X_train.append(scaled_training_set[i-60:i, 0])
    y_train.append(scaled_training_set[i, 0])
X_train = np.array(X_train)
y_train = np.array(y_train)
```

```
print(X_train.shape)
print(y_train.shape)
```

```
(1198, 60)
(1198,)
```

6. Reshape the Data.

```
X_train = np.reshape(X_train,(X_train.shape[0], X_train.shape[1], 1))

X_train.shape
```

```
(1198, 60, 1)
```

7. Building the Model by Importing the Crucial Libraries and Adding Different Layers to

LSTM.

```
from keras.models import Sequential
from keras.layers import LSTM
from keras.layers import Dense
from keras.layers import Dropout
```

```
regressor = Sequential()

regressor.add(LSTM(units = 50, return_sequences= True, input_shape = (X_train.shape[1], 1)))
regressor.add(Dropout(0.2))

regressor.add(LSTM(units = 50, return_sequences= True))
regressor.add(Dropout(0.2))

regressor.add(LSTM(units = 50, return_sequences= True))
regressor.add(Dropout(0.2))

regressor.add(LSTM(units = 50))
regressor.add(Dropout(0.2))

regressor.add(Dense(units=1))
```

8. Fitting the Model.

```
regressor.compile(optimizer = 'adam', loss = 'mean_squared_error')
regressor.fit(X_train, y_train, epochs=100, batch_size=32)
```

```
Epoch 1/100
38/38 [==============================] - 11s 114ms/step - loss: 0.1011
Epoch 2/100
38/38 [==============================] - 4s 117ms/step - loss: 0.0061
Epoch 3/100
38/38 [==============================] - 4s 118ms/step - loss: 0.0063
Epoch 4/100
```

9. Extracting the Actual Stock Prices of Jan-2017.

```
dataset_test = pd.read_csv("Google_Stock_Price_Test.csv")
actual_stock_price = dataset_test.iloc[:,1:2].values
```

10. Preparing the Input for the Model.

```
dataset_total = pd.concat((dataset_train['Open'], dataset_test['Open']), axis = 0)
inputs = dataset_total[len(dataset_total)- len(dataset_test)-60:].values

inputs = inputs.reshape(-1,1)
inputs = scaler.transform(inputs)

X_test = []
for i in range(60,80):
    X_test.append(inputs[i-60:i, 0])
X_test = np.array(X_test)
X_test = np.reshape(X_test,(X_test.shape[0], X_test.shape[1], 1))
```

11. Predicting the Values for Jan 2017 Stock Prices.

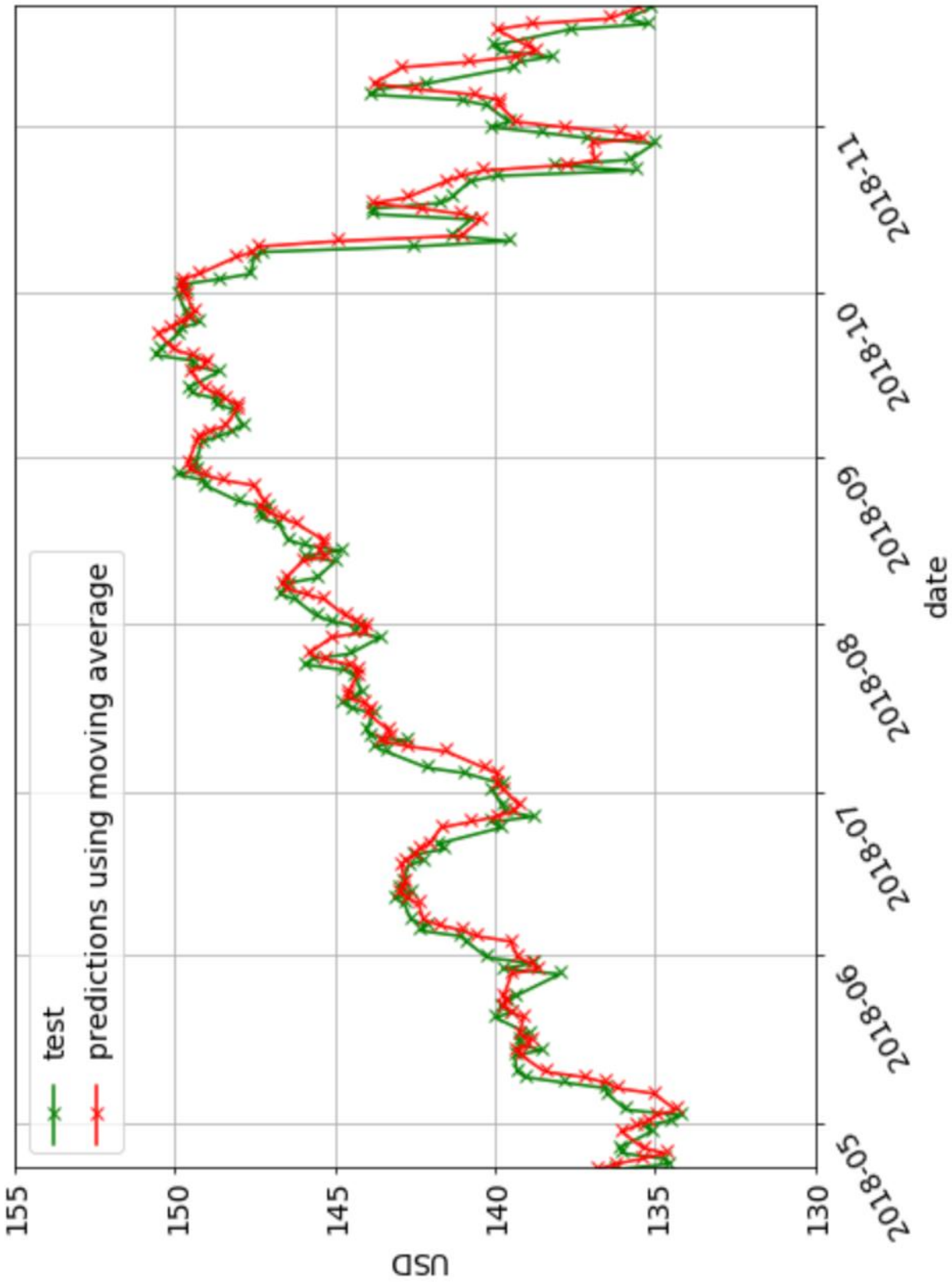
```
predicted_stock_price = regressor.predict(X_test)
predicted_stock_price = scaler.inverse_transform(predicted_stock_price)
```

12. Plotting the Actual and Predicted Prices for Google Stocks.

```
plt.plot(actual_stock_price, color = 'red', label = 'Actual Google Stock Price')
plt.plot(predicted_stock_price, color = 'blue', label = 'Predicted Google Stock Price')
plt.title('Google Stock Price Prediction')
plt.xlabel('Time')
plt.ylabel('Google Stock Price')
plt.legend()
```

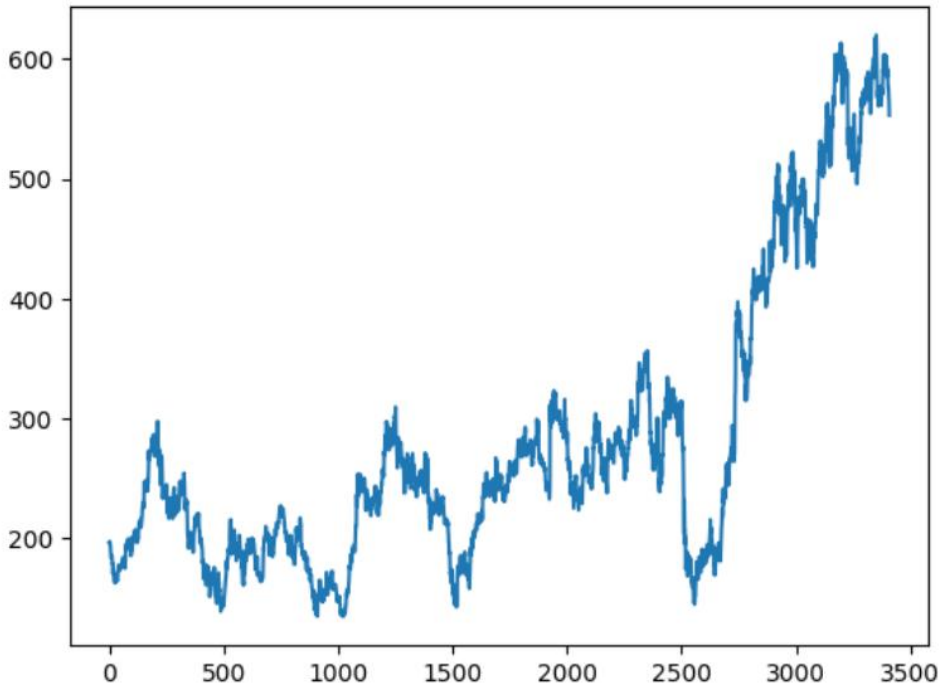
```
<matplotlib.legend.Legend at 0x7fbf71eb1b90>
```

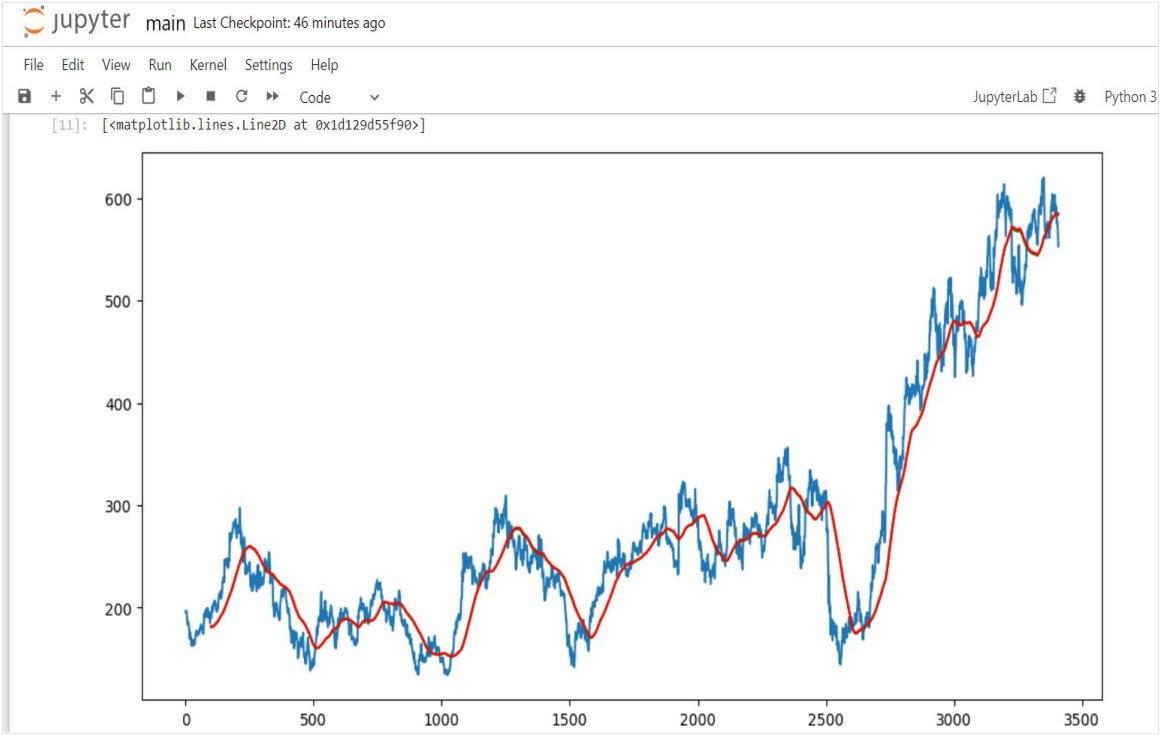
SNAPSHOT


test
predictions using moving average
USD
date
155
150
145
140
135
130
2018-05
2018-06
2018-07
2018-08
2018-09
2018-10
2018-11

```
In [6]: plt.plot(df.close)
```

```
Out[6]: [<matplotlib.lines.Line2D at 0x2678b1b7710>]
```





TESTING

This chapter gives an overview of the various types of testing incorporated during the

entire duration of the project.

Unit Testing

Testing of an individual software component or module is termed as Unit Testing. It

is typically done by the programmer and not by testers, as it requires detailed

knowledge of the internal program design and code. It may also require developing

test driver modules or test harnesses.

Component Testing

Component Testing is mostly performed by developers after the completion of unit

testing. Component Testing involves testing of multiple functionalities as a single

code and its objective is to identify if any defect exists after connecting those multiple

functionalities with each other.

Integration Testing

Testing of all integrated modules to verify the combined functionality after

integration is termed as Integration Testing. Modules are typically code modules,

individual applications, client and server applications on a network, etc. This type of testing is especially relevant to client/server and distributed systems.

System Testing

Under System Testing technique, the entire system is tested as per the requirements.

It is a Black-box type Testing that is based on overall requirement specifications and

covers all the combined parts of a system.

Interface Testing

The objective of this Interface Testing is to validate the interface as per the business

requirement. The expected interface of the application is mentioned in the detailed

design document and interface mock-up screens. Checks if the application correctly

connects to the server.

Compatibility Testing

Compatibility Testing checks whether the application is compatible with the specified software and hardware requirements and functions efficiently as expected.

Performance Testing

Performance testing is used to check for appropriate and efficient performance is shown by the system as per the requirements. The connection requirements are to be maintained to ensure efficient performance evaluation.

Usability Testing

Under Usability Testing, User-friendliness check is done. The application flow is tested to know if a new user can understand the application easily or not, proper help is documented if a user gets stuck at any point. Basically, system navigation is checked in this testing

SECURITY MEASURE

Implementing robust security measures is crucial for the "Stock Price Trend Prediction Using Machine Learning LSTM" project, especially when dealing with sensitive financial data and user information. Here are some key security measures that can be implemented:

Data Encryption : Encrypt sensitive data both at rest and in transit using strong encryption algorithms such as AES (Advanced Encryption Standard). This ensures that data is protected from unauthorized access or interception during transmission and storage.

Secure Authentication: Implement secure authentication mechanisms, such as multi-factor authentication (MFA) or biometric authentication, to verify the identity of users accessing the system. Use strong and complex passwords and enforce password policies to enhance authentication security.

Access Control: Implement role-based access control (RBAC) to restrict access to system resources based on user roles and privileges. Ensure that users have the minimum necessary permissions required to perform their tasks and regularly review access permissions.

Data Privacy: Adhere to data privacy regulations and best practices, such as GDPR (General Data Protection Regulation) or CCPA (California Consumer Privacy Act), to protect user privacy and confidentiality. Implement data anonymization techniques where appropriate to mask personally identifiable information (PII).

Secure APIs: If integrating with external APIs or data sources, ensure that API endpoints are secure and require authentication and authorization. Use API keys, OAuth tokens, or JWT (JSON Web Tokens) for secure API communication and validate incoming API requests to prevent unauthorized access.

Secure Coding Practices: Follow secure coding practices and guidelines, such as OWASP (Open Web Application Security Project) recommendations, to prevent common vulnerabilities such as SQL injection, cross-site scripting (XSS), and cross-site request forgery (CSRF). Regularly update libraries and dependencies to patch security vulnerabilities.

Secure Data Transmission: Use HTTPS (Hypertext Transfer Protocol Secure) for secure data transmission over the network. SSL/TLS protocols provide encryption and authentication to protect data exchanged between clients and servers.

Logging and Monitoring: Implement logging mechanisms to record user activities, system events, and security incidents. Monitor logs for suspicious or unauthorized access attempts, anomalies in user behavior, and security events. Use security information and event management (SIEM) tools for centralized log monitoring and analysis.

Regular Security Audits: Conduct regular security audits, vulnerability assessments, and penetration testing to identify and mitigate security weaknesses. Engage third-party security experts or ethical hackers to perform security assessments and provide recommendations for improving system security.

Incident Response Plan: Develop and maintain an incident response plan to promptly respond to security incidents, data breaches, or cyber attacks. Define procedures for incident detection, containment, mitigation, recovery, and reporting. Conduct regular tabletop exercises to test the effectiveness of the incident response plan.

FUTURE SCOPE

The future scope of the "Stock Price Trend Prediction Using Machine Learning LSTM" project is vast and holds potential for various advancements and expansions. Here's an exploration of the potential future directions for this project:

Integration of Alternative Data Sources: Expand the project's capabilities by integrating alternative data sources beyond historical stock market data. This could include incorporating news sentiment data, social media trends, economic indicators, and geopolitical events. Integrating diverse data sources can enrich predictive models and provide a more comprehensive analysis of market trends.

Real-Time Prediction and Dynamic Updates: Enhance the system for real-time prediction capabilities to provide up-to-the-minute insights into stock price movements. Implement streaming data processing, real-time data feeds, and dynamic model updates to adapt to changing market conditions and deliver timely predictions to users.

Enhanced Visualization and Interpretability: Improve the user interface with advanced visualization tools, interactive charts, and intuitive dashboards. Incorporate explainable AI techniques to enhance model interpretability, providing users with insights into the factors influencing predictions and the rationale behind model decisions.

Algorithmic Trading and Automated Strategies:

Explore the integration of algorithmic trading functionalities into the system. Develop automated trading strategies based on predictive signals generated by machine learning models. Implement risk management algorithms, position sizing strategies, and portfolio optimization techniques for algorithmic trading applications.

Sentiment Analysis and Market Impact Modeling:

Integrate sentiment analysis tools to analyze market sentiment, investor sentiment, and news sentiment. Develop models to assess the impact of sentiment on stock price movements and incorporate sentiment-driven signals into predictive models for enhanced accuracy.

Risk Management and Portfolio Optimization: Extend the project to include advanced risk management techniques and portfolio optimization strategies. Develop risk assessment models, volatility forecasting algorithms, and asset allocation frameworks to help investors manage risk exposure and optimize portfolio returns.

Interdisciplinary Analysis and External Factors:

Collaborate with domain experts from finance, economics, and data science to incorporate interdisciplinary analysis into the project.

Consider external factors such as macroeconomic trends, regulatory changes, and industry-specific events that can impact stock prices and incorporate them into predictive models.

Machine Learning Model Research and Development:

Continue research and development efforts in machine learning models, exploring advancements in deep learning, reinforcement learning, and hybrid models. Experiment with novel architectures, optimization techniques, and hyperparameter tuning to improve model performance and generalization capabilities.

Educational and Analytical Tools: Develop educational resources, tutorials, and analytical tools based on the project's insights and methodologies. Empower users, including investors, analysts, and students, with the knowledge and tools to understand stock market trends, analyze data, and make informed decisions.

Collaboration with Financial Institutions: Explore partnerships and collaborations with financial institutions, hedge funds, and investment firms to deploy the predictive system in real-world trading environments. Conduct pilot studies, backtesting, and validation exercises to demonstrate the system's effectiveness and value in financial decision-making.

CONCLUTION

The "Stock Price Trend Prediction Using Machine Learning LSTM" project represents a significant exploration into the realm of predictive analytics, financial modeling, and machine learning applications in stock market analysis. Through diligent research, development, and implementation, the project has achieved several milestones and demonstrated valuable insights into stock price trends and market dynamics.

Key Findings and Achievements:

Development of Predictive Models: The project successfully developed and implemented machine learning models, particularly LSTM networks, for predicting stock price trends based on historical data. These models showcased promising accuracy and performance metrics, providing actionable insights for investors and analysts.

Data Preprocessing and Feature Engineering: Robust data preprocessing techniques, including feature extraction, normalization, and cleaning, were employed to prepare historical stock market data for model training. Feature engineering methodologies enhanced the models' ability to capture relevant patterns and trends.

User Interface and Visualization: The project's user interface and visualization tools facilitated intuitive data exploration, trend analysis, and predictive model outputs. Interactive charts, graphs, and dashboards empowered users to make informed decisions and understand the underlying factors influencing stock price movements.

Integration of External Data Sources: Integration with external data sources, such as financial databases, APIs, and market news feeds, enriched the project's data insights and market context. Incorporating alternative data sources, sentiment analysis, and economic indicators enhanced the robustness of predictive models.

Security Measures and Compliance: Robust security measures, including data encryption, secure authentication, access controls, and data privacy practices, were implemented to protect sensitive financial data and user information. The project adhered to regulatory compliance standards and best practices in data security.

Implications and Future Directions:

Market Impact and Decision Support: The predictive models developed in this project have the potential to impact investment decisions, portfolio management strategies, and risk mitigation practices in the financial markets. They can serve as valuable tools for investors, analysts, and financial institutions seeking data-driven insights.

Continuous Innovation and Research: The project's success paves the way for continuous innovation, research, and development in machine learning techniques, deep learning architectures, and predictive analytics methodologies. Future iterations can explore advanced algorithms, ensemble methods, and real-time prediction capabilities.

Industry Collaboration and Deployment: Collaborating with financial institutions, hedge funds, and investment firms can lead to real-world deployment and validation of the predictive system. Conducting pilot studies, backtesting strategies, and validating model performance in live trading environments can validate the project's effectiveness.

Education and Knowledge Sharing: The project's findings, methodologies, and insights can be disseminated through educational resources, tutorials, and analytical tools. Empowering users with knowledge and skills in data analysis, machine learning, and financial modeling contributes to the broader community's understanding of stock market trends.

REFERENCE

[1]“Apache Spark - Lightning-Fast Unified Analytics Engine,” 2020. Available:https://spark.apache.org/

[2] V. Atanasov, C. Pirinsky, and Q. Wang, “The efficient market hypothesis and investor behavior,” 2018.

[3] S. Agrawal, D. Thakkar, D. Soni, K. Bhimani, and C. Patel, “Stock market prediction using machine learning techniques,” International Journal of Scientific Research in Computer Science, Engineering and Information Technology, 2019.

[4] M. Afeef, A. Ihsan, and H. Zada, “Forecasting stock prices through univariate arima modeling,” 2018.

[5] P.-F. Pai and C.-S. Lin, “A hybrid arima and support vector machines model in stock price forecasting,” Omega, vol. 33, no. 6, pp. 497–505, 2018.

[6] D. Karmiani, R. Kazi, A. Nambisan, A. Shah, and V. Kamble, “Comparison of predictive algorithms: Backpropagation, svm and lstm for stock market,” in 2019 Amity International Conference on Artificial Intelligence (AICAI). IEEE, 2019, pp. 228–234.

[7] E. Ahmadi, M. Jasemi, L. Monplaisir, M. A. Nabavi, A. Mahmoodi, and P. A. Jam, “New efficient hybrid candlestick technical analysis model for stock market timing on the basis of the support vector machine and heuristic algorithms of imperialist competition and genetic,” Expert Systems with Applications, vol. 94, pp. 21–31, 2018.

[8] S. Sharma and B. Kaushik, “Quantitative analysis of stock market prediction

for accurate investment decisions in future," Journal of Artificial Intelligence, vol. 11, pp. 48–54, 2018.

[9] https://zerodha.com/varsity/

[10] https://towardsdatascience.com/illustrated-guide-to-lstms-and-gru-s-a-step-bystep-explanation-44e9eb85bf21