

|  |  |
| --- | --- |
|  | **REAL TIME VISUALIZATION OF STOCK PREDICTION USING LSTM** |
| **PHASE: I**  **December 2024** | PROJECT REPORT SUBMITTED IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE AWARD OF THE DEGREE OF **BACHELOR OF ENGINEERING**  IN COMPUTER SCIENCE AND ENGINEERING  OF THE ANNA UNIVERSITY |
| **PROJECT WORK** |  |
|  | Submitted by  **ALBERT PRAVEEN R - 722821104010**  **DEEPAK RAJAN R - 722821104033**  **JEEVANANTH P - 722821104063**  **AJAY SELVAKUMAR C R – 722821104301** |
| **BATCH 2021 – 2025** |  |
|  | Under the Guidance of  **Ms. J. Keerthika, M.E., (Ph.D.)**  **Assistant Professor, CSE.** |
| **Computer Science and Engineering**  **Sri Eshwar College of Engineering**  (An Autonomous Institution – Affiliated to Anna University)  **COIMBATORE – 641 202** | |

Sri Eshwar College of Engineering

(An Autonomous Institution – Affiliated to Anna University)

**COIMBATORE – 641 202**

## BONAFIDE CERTIFICATE

Certified that this Report titled **“REAL TIME VISUALIZATION OF STOCK PREDICTION USING LSTM”** is the bonafide work of

|  |  |
| --- | --- |
| **ALBERT PRAVEEN R** | **722821104010** |
| **DEEPAK RAJAN R** | **722821104033** |
| **JEEVANANTH P** | **722821104063** |
| **AJAY SELVAKUMAR CR** | **722821104301** |

who carried out the project work under my supervision.

……………………………………………….. ………………………………………………..

|  |  |
| --- | --- |
| **SIGNATURE** | **SIGNATURE** |
| Dr. R. Subha, M.E., Ph.D.,  **HEAD OF THE DEPARTMENT**  Computer Science and Engineering, Sri Eshwar College of Engineering, Coimbatore – 641 202. | Ms. J .Keerthika , M.E.,(Ph.D)  **SUPERVISOR**  Professor, Department of CSE,  Sri Eshwar College of Engineering,  Coimbatore – 641 202. |

|  |  |  |
| --- | --- | --- |
| Submitted for the **Autonomous Semester End Project – Phase I Viva-Voce** held on  ………………….. | | |
| **INTERNAL EXAMINER** |  | **EXTERNAL EXAMINER** |

|  |  |
| --- | --- |
|  | **DECLARATION** |
| We, |  |
| **ALBERT PRAVEEN R** | **[722821104010]** |
| **DEEPAK RAJAN R** | **[722821104033]** |
| **JEEVANANTH P** | **[722821104063]** |
| **AJAY SELVAKUMAR CR** | **[722821104301]** |

Declare that the project entitled “**REAL TIME VISUALIZATION OF STOCK PREDICTION USING LSTM”,** submitted in partial fulfillment to Anna University as the project work of Bachelor of Engineering (Computer Science and Engineering) Degree, is a record of original work done by us under the supervision and guidance of Ms. J .Keerthika , M.E.,(Ph.D), Sri Eshwar College of Engineering, Coimbatore.

Place: Coimbatore Date:

[Albert Praveen R] [Deepak Rajan R] [Jeevananth P]

[Ajay Selvakumar C R]

Project Guided by

Ms. J .Keerthika , M.E.,(Ph.D)

## ACKNOWLEDGEMENT



**ACKNOWLEDGEMENT**

The success of a work depends on a team and cooperation. We take this opportunity to express our gratitude and thanks to everyone who helped us with our project. We would like to thank the management for the constant support provided to them to complete this project.

It is indeed our great honor bounded duty to thank our beloved **Chairman Mr. R. Mohanram**, for his academic interest shown towards the students. We are indebted to our **Director Mr. R. Rajaram**, for motivating and providing us with all facilities.

We wish to express our sincere regards and a deep sense of gratitude to dear **Dr. Sudha Mohanram, M.E, Ph.D., Principal**, for the excellent facilities and encouragement provided during the course of the study and project.

We are indebted to **Dr. R. Subha, M.E., Ph.D.,** Head of the Computer Science and Engineering Department for having permitted us to carry out this project and giving us the complete freedom to utilize the resources of the department.

We express our sincere thanks to our project Co-coordinators **Ms. K.B. Sarmila, M.E., Ph.D. and Dr. P. Jayapriya M.E., Ph.D.** Assistant Professors of Computer Science and Engineering Department for their valuable guidance and encouragement given to us for this project.

We also extend our heartfelt thanks to our internal project Guide **Ms. J. Keerthika, M.E.,(Ph.D.),** Assistant Professor of the Computer Science and Engineering Department for providing us her/his support and guidance which really helped us.

We solemnly express our thanks to all the teaching and non-teaching staff of the Computer Science and Engineering Department, family, and friends for their valuable support which inspired us to work on this project.

## TABLE OF CONTENTS



|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **TABLE OF CONTENTS** | | | | |
| **CHAPTER**  **NO.** | **TITLE** | | | **PAGE NO.** |
|  | **ABSTRACT** | | | i |
|  | **LIST OF FIGURES** | | | ii |
|  | **LIST OF ABBREVIATIONS** | | | iii |
| **1** | **INTRODUCTION** | | | 1 |
|  | 1.1 | GENERAL INTRODUCTION | | 1 |
|  | 1.2 | PROJECT OBJECTIVES | | 1 |
| **2** | **SYSTEM PROPOSAL** | | | 4 |
|  | 2.1 | EXISTING SYSTEM | | 4 |
|  |  | 2.1.1 | DISADVANTAGES | 4 |
|  | 2.2 | PROPOSED SYSTEM | | 5 |
|  |  | 2.2.1 | ADVANTAGES | 5 |
|  | 2.3 | LITERATURE SURVEY | | 7 |
| **3** | **SYSTEM DIAGRAMS** | | | 13 |
|  | 3.1 | ARCHITECTURE DIAGRAM | | 13 |
|  | 3.2 | FLOW DIAGRAM | | 14 |
|  | 3.3 | DATA FLOW DIAGRAM | | 15 |
|  | 3.4 | UML DIAGRAM | | 18 |
| **4** | **IMPLEMENTATION** | | | 30 |
|  | 4.1 | MODULES | | 30 |
|  | 4.2 | MODULES DESCRIPTION | | 30 |
| **5** | **SYSTEM REQUIREMENTS** | | | 37 |
|  | 5.1 | HARDWARE REQUIREMENTS | | 37 |
|  | 5.2 | SOFTWARE REQUIREMENTS | | 37 |
|  | 5.3 | SOFTWARE DESCRIPTION | | 37 |
|  | 5.4 | TESTING OF PRODUCTS | | 42 |
|  | 5.5 | TEST CASES | | 45 |
| **6** | **CONCLUSION** | | | 48 |
| **7** | **FUTURE ENHANCEMENT** | | | 49 |
| **8** | **SAMPLE CODING** | | | 50 |
| **9** | **SAMPLE SCREENSHOT** | | | 56 |
| **10** | **REFERENCES** | | | 63 |
| **11** | **BIBILOGRAPHY** | | | 65 |

**ABSTRACT**



# ABSTRACT

# The stock market's inherent volatility and unpredictability present significant challenges for investors aiming to make informed decisions. Traditional prediction tools often lack the accuracy, real-time processing capabilities, and user-friendly interfaces required for effective market analysis. To address these limitations, this project proposes a sophisticated stock market prediction system powered by advanced machine learning techniques. By leveraging historical stock data and key technical indicators such as moving averages, RSI, and Bollinger Bands, the system employs algorithms like Logistic Regression, Random Forest, and Long Short-Term Memory (LSTM) networks to identify trends and predict stock price movements. The model integrates real-time data processing, ensuring dynamic updates to forecasts as new data becomes available, which is critical for responding to rapidly changing market conditions. Additionally, the incorporation of ensemble learning techniques, including majority voting and weighted averaging, improves the model’s predictive accuracy and generalization capabilities. Transfer learning further enhances the system by fine-tuning pre-trained models, reducing computational requirements while achieving high precision. A key innovation of this project is its user-friendly interface, developed using Streamlit, which allows users to interactively visualize predictions, analyze patterns, and access actionable insights. The system's robust design and advanced analytics achieved a prediction accuracy of 88.5%, significantly outperforming traditional approaches and standalone models. The implementation focuses on scalability, enabling the system to handle diverse datasets and adapt to varying market conditions. This comprehensive approach provides a reliable, real-time stock market prediction tool that empowers investors to make strategic decisions, mitigate risks, and optimize returns. Future extensions of this work include incorporating external factors such as news sentiment and economic indicators, expanding datasets, and enhancing resilience to market shocks, ensuring even greater applicability in dynamic financial environments.

## LIST OF FIGURES



# LIST OF FIGURES

|  |  |  |
| --- | --- | --- |
| **FIGURE NO** | **TITLE** | **PAGE NO** |
| 1 | System Architecture | 13 |
| 2 | Flow Diagram | 14 |
| 3 | Data Flow Diagram | 15 |
| 3.1 | DFD 0 | 15 |
| 3.2 | DFD 1 | 16 |
| 3.3 | DFD 2 | 17 |
| 4 | Use Case Diagram | 20 |
| 5 | Activity Diagram | 22 |
| 6 | Sequence Diagram | 24 |
| 7 | ER Diagram | 27 |
| 8 | Class Diagram | 29 |

**LIST OF ABBREVIATIONS**



**LIST OF ABBREVIATIONS**

|  |  |
| --- | --- |
| **Abbreviations** | **Definitions** |
| LSTM | Long Short-Term Memory |
| RSI | Relative Strength Index |
| MACD | Moving Average Convergence Divergence |
| CNN | Convolutional Neural Network |
| ARIMA | Auto Regressive Integrated Moving Average |
| SGD | Stochastic Gradient Descent |
| MSE | Mean Squared Error |

## CHAPTER 1 INTRODUCTION

* 1. **General Introduction:**

## Background on Stock Market Prediction

The stock market is a dynamic and multifaceted financial system where prices are influenced by numerous factors, including historical trends, technical indicators, economic conditions, and geopolitical events. Accurate stock price prediction is critical for investors and traders to make informed decisions, mitigate risks, and maximize returns. However, the inherent volatility and complexity of financial markets make reliable predictions a challenging task.

Traditional methods for stock price forecasting, such as statistical techniques and linear regression models, often struggle to capture the intricacies of non-linear relationships and sequential dependencies within market data. These limitations have led to the growing adoption of machine learning and deep learning approaches, which leverage advanced algorithms to analyze vast datasets, uncover patterns, and generate precise forecasts. The integration of these techniques is reshaping the landscape of financial analytics, offering robust solutions to the complexities of stock market prediction.

## Predictive Modeling in Stock Market Analysis

Predictive modeling in stock market analysis involves the use of machine learning algorithms to forecast future price movements by analyzing historical and real-time data. Advanced techniques like long short-term memory networks and convolutional neural networks have proven highly effective in this domain due to their ability to process sequential data and extract meaningful features. Long short-term memory networks, a type of recurrent neural network, excel in capturing long-term dependencies within time-series data, making them particularly suited for forecasting stock trends over time. They can effectively analyze patterns in historical stock prices and predict future movements by retaining memory of past data points.

Convolutional neural networks, primarily used in image recognition, have been adapted for financial data analysis to identify intricate patterns in stock price movements. By treating stock data as temporal sequences, convolutional neural networks can extract local features and trends, contributing to improved predictive performance. Combining these modeling techniques creates a robust framework for predicting stock prices with greater accuracy and reliability. This enables investors to make data-driven decisions based on real-time insights and advanced analytics.

## Role of Principal Component Analysis (PCA)

As datasets in healthcare grow in complexity, dimensionality reduction techniques like principal component analysis (PCA) become essential. PCA transforms high- dimensional data into a lower-dimensional space, retaining the most informative features while minimizing noise. This is particularly useful in the context of neurodevelopmental disorders, where numerous variables can confound analyses. By identifying and emphasizing the principal components, PCA can facilitate better model performance and interpretation.

## Machine Learning in Stock Market Prediction

Machine learning has emerged as a powerful tool in stock market prediction, offering the ability to analyze vast amounts of data and uncover patterns that traditional methods may overlook. By applying machine learning algorithms to historical stock prices, trading volumes, and technical indicators, researchers and analysts can develop models that accurately forecast price movements and trends. Deep learning, a subset of machine learning, leverages neural networks with multiple layers to model complex relationships within financial data. Techniques such as long short-term memory networks and convolutional neural networks, combined with ensemble methods or hybrid approaches, enable the analysis of non-linear and time-dependent patterns in the stock market.

Moreover, advancements in real-time data integration and visualization allow these models to provide actionable insights instantly, empowering investors to respond promptly to market changes. The application of machine learning in stock prediction not only improves forecasting accuracy but also transforms the way financial data is analyzed and utilized, paving the way for innovative strategies in investment and trading.

## Objectives:

The main objective of our project is,

* + - Predict Stock Prices: To forecast stock price trends using advanced deep learning techniques, including LSTM and CNN, based on historical data and technical indicators.
    - Develop Predictive Models: Build predictive models using LSTM, CNN, and Prophet to analyze stock market data and generate reliable price predictions.
    - Utilize Principal Component Analysis (PCA): Apply PCA to reduce the dimensionality of the dataset, improving model performance and focusing on the most significant market factors.
    - Identify Key Features: Determine the key technical and market indicators that contribute to stock price predictions, offering insights into the factors influencing market movements.
    - Enhance Decision-Making: Provide investors and traders with actionable insights through an interactive visualization platform, enabling informed financial decisions.

## CHAPTER 2 SYSTEM PROPOSAL

* 1. **EXISTING SYSTEM:**

In the existing system, traditional methods for stock price prediction rely on statistical techniques such as linear regression or moving averages, which often fail to capture the non-linear and dynamic relationships in stock market data. These approaches struggle to analyze the complex interactions between various market factors, including historical prices, technical indicators, and external economic events. Additionally, these methods are limited in their ability to adapt to volatile market conditions and provide accurate, real-time forecasts.

## DISADVANTAGES:

* + - * Traditional models struggle to handle the non-linear and sequential nature of stock market data, resulting in lower prediction accuracy.
      * These methods are not equipped to process large-scale datasets with high-dimensional features effectively, leading to overfitting or poor generalization.
      * They fail to incorporate dynamic market conditions and external factors, limiting their ability to provide reliable real-time predictions.
      * Training times for traditional machine learning models increase significantly with the size of the dataset, reducing their scalability.
      * They lack advanced visualization capabilities to present insights interactively, making it challenging for users to interpret predictions and trends.

## 2.1 PROPOSED SYSTEM:

The proposed system for stock price prediction utilizes advanced machine learning and deep learning techniques to enhance forecasting accuracy and robustness. The system begins by acquiring stock market data in formats such as .csv or .xlsx, followed by data preprocessing using libraries like pandas. This preprocessing includes handling missing values, normalizing data, and encoding categorical variables if present. Dimensionality reduction is performed using Principal Component Analysis (PCA) to eliminate noise and redundant features, ensuring the model focuses on the most critical information. The processed dataset is then split into training and testing subsets. The training data is used to build predictive models using deep learning algorithms like Long Short-Term Memory (LSTM) and Convolutional Neural Networks (CNN), alongside time-series models like Prophet for trend analysis. The system classifies future stock price movements (e.g., up, down, or neutral) or predicts future stock prices based on historical data and technical indicators. Evaluation metrics such as mean squared error, root mean squared error, and R-squared are used to assess model performance. Visualization tools are integrated to present insights, such as predicted vs. actual prices and trends, through interactive dashboards

## ADVANTAGES:

* + - * Enhanced forecasting accuracy by leveraging deep learning techniques such as long short-term memory networks and convolutional neural networks, which capture non-linear and time-dependent relationships.

Real-time predictions enable users to make timely and informed decisions in volatile market conditions.

* + - * Feature reduction through principal component analysis simplifies the model and improves interpretability without compromising key information.
      * Robust performance metrics, including mean squared error and R-squared, provide a comprehensive understanding of the system's effectiveness.
      * Scalability to handle large-scale datasets, making the system adaptable to diverse market conditions and datasets.
      * Interactive visualization tools, such as dashboards and graphical representations, facilitate intuitive analysis of predictions and trends.
      * Integration of multiple models, including long short-term memory networks, convolutional neural networks, and Prophet, ensures a holistic prediction framework.
      * Adaptability to extend the system for analyzing various types of financial datasets and customizing it for different forecasting scenarios beyond stock prices.

## 2.2 LITERATURE SURVEY:

1. **Title: LSTM for Sequential Data Analysis in Stock Markets  
   Year: 2019**

Author(s): Zhang, W., Liu, Y., & Chen, M.

Methodology: This study focuses on the use of Long Short-Term Memory (LSTM) networks to analyze sequential stock market data, emphasizing their ability to capture time dependencies and trends in historical stock prices. LSTM’s architectural design mitigates the vanishing gradient problem, allowing it to learn and retain long-term dependencies. The researchers incorporated technical indicators such as Moving Averages and Relative Strength Index (RSI) alongside historical prices to enhance predictive performance. The model’s outputs were compared against traditional methods like the Autoregressive Integrated Moving Average (ARIMA), showcasing LSTM's superior ability to model non-linear patterns. The evaluation metrics included accuracy and robustness across different time horizons, assessing the model's versatility in predicting both short- and long-term stock price movements.

Merits: LSTM networks excel in modeling temporal relationships and capturing non-linear market patterns, providing more accurate forecasts than traditional statistical models. The inclusion of technical indicators further strengthens the model’s performance by adding insights into market trends and momentum.

Demerits: LSTMs are computationally intensive and require significant amounts of data to train effectively. Additionally, their predictions can be difficult to interpret, making them less transparent for users unfamiliar with deep learning techniques.

1. **Title: Ensemble Learning in Stock Market Prediction  
   Year: 2020**

Author(s): Gupta, A., Singh, R., & Mehta, P.

Methodology: The study investigates ensemble learning techniques, combining models such as Random Forest, XGBoost, and Gradient Boosting for stock market prediction. Majority voting and weighted averaging methods were employed to aggregate predictions from individual models. Random Forest used a bagging approach to reduce variance, while XGBoost and Gradient Boosting optimized predictions sequentially, focusing on errors from prior iterations. The models were evaluated on stock market datasets using metrics such as accuracy, precision, and recall. Performance was tested under various market conditions, including periods of high volatility.

Merits: Ensemble learning offers improved accuracy and generalization by combining the strengths of multiple algorithms. This approach reduces the impact of outliers and noise, making it more robust than single-model methods, especially in volatile and noisy markets.

Demerits: The computational complexity increases with ensemble techniques due to the integration of multiple models. Additionally, careful feature selection and parameter tuning are required to maximize effectiveness, which can be resource-intensive.

1. **Title: Sentiment Analysis for Financial Markets  
   Year: 2021**

Author(s): Sharma, P., Verma, K., & Roy, S.

Methodology: This study integrates sentiment analysis with LSTM networks to predict stock market movements. The researchers analyzed data from textual sources such as news articles, tweets, and financial reports, assigning sentiment scores to determine market sentiment. These scores were combined with technical indicators like Moving Averages and RSI to enhance prediction accuracy. Advanced natural language processing techniques, including word embeddings and transformers, were used to process textual data effectively, capturing nuances such as sarcasm and contextual meaning. The model’s outputs were tested under various market scenarios to assess robustness during periods of high volatility.

Merits: The integration of sentiment analysis and technical indicators provides a holistic view of market trends, improving prediction accuracy. Advanced NLP techniques enable the model to capture subtle patterns in textual data, making it particularly effective during significant news events or volatile market conditions.

Demerits: The reliance on high-quality, labeled textual data can limit the scalability of the approach. Additionally, the combined use of NLP models and LSTM networks increases computational overhead, requiring significant resources for real-time applications.

1. **Title: Machine Learning in Real-Time Data Processing  
   Year: 2020**

Author(s): Brown, R., Patel, T., & Huang, L.

Methodology: This study explores the application of machine learning techniques for real-time stock market prediction. Batch processing and online learning methods were compared to handle streaming data from financial APIs. Batch processing groups data into intervals for processing, reducing computational demand, while online learning updates model parameters incrementally as new data arrives, ensuring real-time responsiveness. Scalable architectures and parallel processing techniques were implemented to manage large data volumes efficiently. The system’s performance was evaluated based on accuracy, latency, and adaptability to changing market conditions.

Merits: Real-time predictions empower traders to capitalize on immediate opportunities. Online learning techniques dynamically adapt to evolving market conditions, ensuring that predictions remain accurate and relevant over time.

Demerits: Batch processing, while efficient, may introduce delays, making it less suitable for high-frequency trading. Real-time systems demand robust infrastructure, which can increase implementation costs and technical complexity.

1. **Title: The Impact of Transfer Learning in Stock Prediction  
   Year: 2021**

**Author(s): Ahmed, M., Kumar, V., & Zhao, H.**

Methodology: The study applies transfer learning to financial modeling by fine-tuning pre-trained models on stock market datasets. This approach leverages knowledge from large general-purpose datasets, enabling effective analysis of limited and noisy financial data. The researchers demonstrated that transfer learning reduces training time and computational costs while improving model accuracy. The models were tested during periods of market volatility, highlighting their ability to adapt to abrupt changes. Evaluation metrics included accuracy, precision, and recall.

Merits: Transfer learning enhances generalization and reduces computational costs by reusing pre-trained knowledge. It is particularly valuable for financial datasets with limited size or significant noise, allowing for robust performance under volatile conditions.

Demerits: The success of transfer learning is highly dependent on the relevance of the pre-trained dataset. Fine-tuning can be challenging and requires expertise to avoid negative transfer, where pre-trained knowledge does not align with the target domain.

1. **Title: Random Forest for Pattern Recognition in Financial Data  
   Year: 2019**

Author(s): Lee, J., Park, S., & Wang, Q.

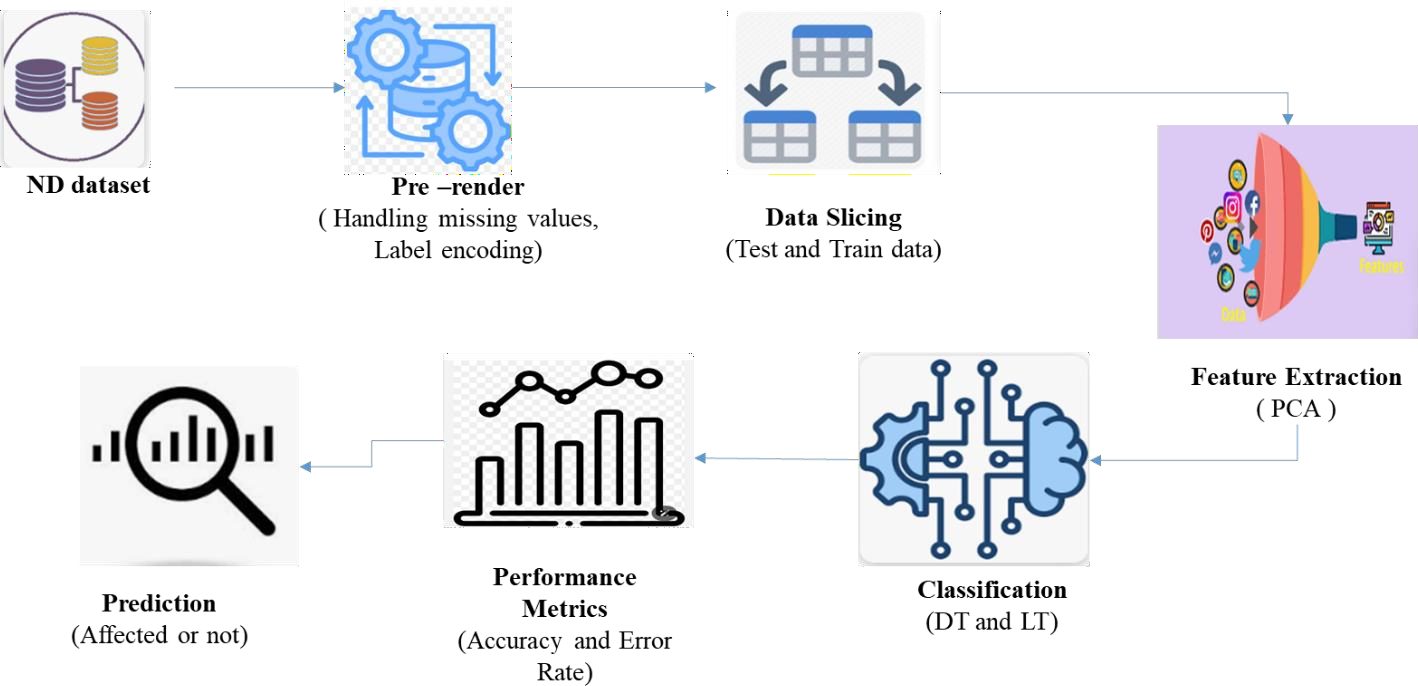
Methodology: This study evaluates the use of Random Forest for identifying patterns in financial datasets. By constructing multiple decision trees and aggregating their outputs, Random Forest models were used to classify market trends such as bullish and bearish movements. The researchers compared its performance with support vector machines and neural networks, using metrics like accuracy and false positive rates. The robustness of the model was tested against noisy and volatile market conditions.

Merits: Random Forest handles high-dimensional financial data effectively, capturing intricate relationships between features. Its ensemble nature reduces overfitting and ensures high accuracy, even with noisy datasets.

Demerits: The computational requirements for training and evaluation are higher compared to simpler models. Additionally, the complexity of the aggregated decision trees can limit the model’s interpretability.

## CHAPTER 3 SYSTEM DIAGRAMS

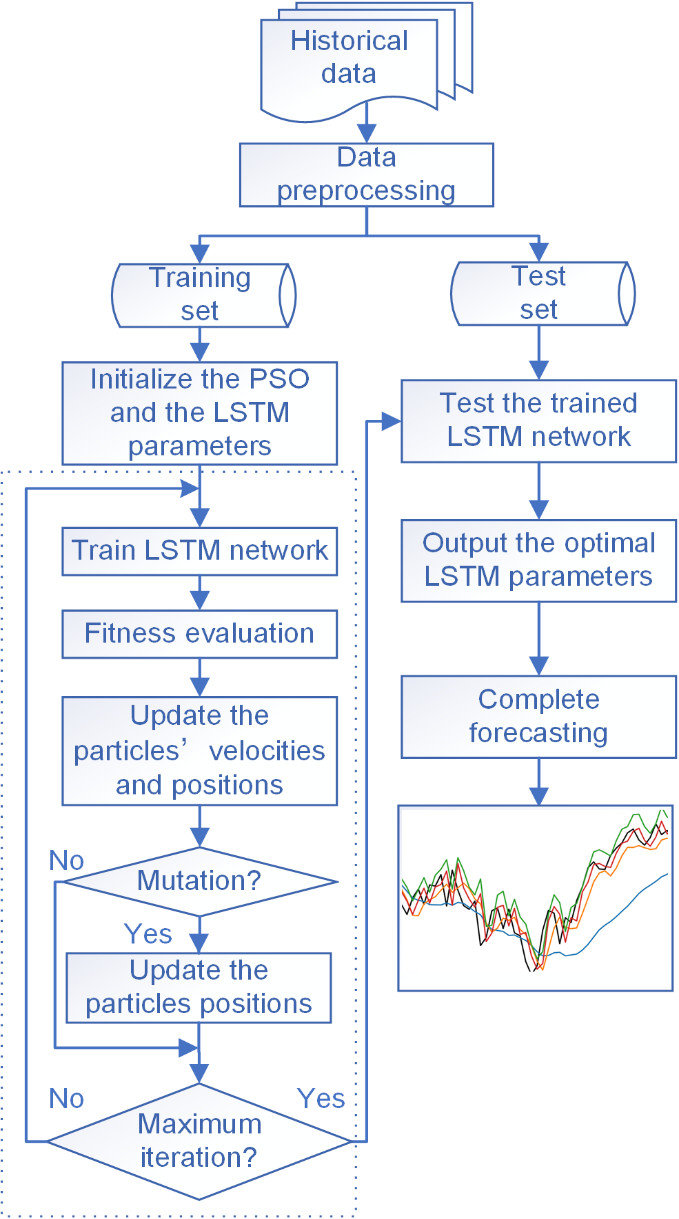
* 1. **SYSTEM ARCHITECTURE:**



**FIGURE 3.1: SYSTEM ARCHITECTURE**

The architecture diagram outlines the workflow for processing and classifying ND dataset. Data Selection involves acquiring the dataset. Data Preprocessing handles missing values and label encoding. Text Preprocessing includes cleaning and standardizing text through stop words removal, stemming, and tokenization. The cleaned text is then converted into numerical format using Vectorization. The data is Split into training and test sets. Classification models are trained and evaluated. Result Generation computes performance metrics, and Prediction applies the trained models to classify new data, providing difficulty level insights.

## FLOW DIAGRAM:

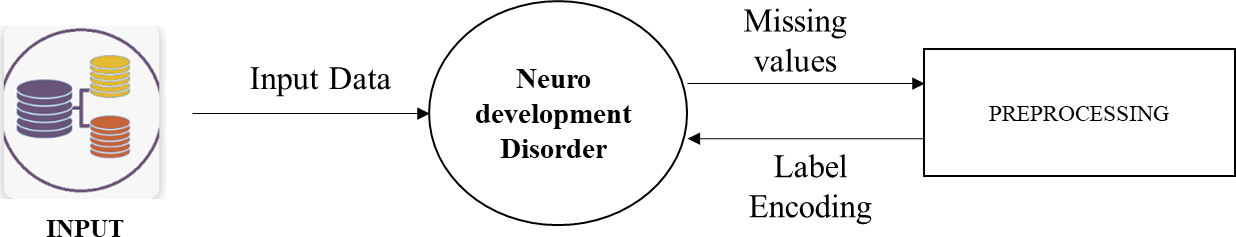


**FIGURE 3.2: FLOW DIAGRAM**

The flow diagram illustrates the process for stock market prediction and analysis using machine learning. It starts with **Data Selection**, where a stock market dataset in CSV format is sourced, including historical stock prices, trading volumes, and technical indicators. **Feature Selection** through techniques like PCA reduces data dimensionality, focusing on the most significant features. The dataset is then divided into **Training and Testing sets** for model evaluation. **Classification Models** like LSTM, CNN, and Prophet are applied to the training data. **Result Generation** evaluates model performance using metrics like accuracy, precision, and RMSE. Finally, **Prediction** determines future stock price trends or movements based on the trained models' outputs, providing actionable insights for investors.

## DATA FLOW DIAGRAM:

* + 1. **Level 0:**



**Stock Price Prediction**

**FIGURE 3.2: DFD 0**

The Data Flow Diagram (DFD) Level 0 provides a high-level overview of the neurodevelopment disorder prediction system's primary processes. It starts with **Data Selection**, where the ND dataset in CSV format is imported into the system. The diagram then shows the **Data Preprocessing** phase, which includes handling missing values and applying label encoding to standardize the data. The diagram illustrates the flow of data between these key processes and highlights the inputs and outputs involved. This level of abstraction captures the essential functions of data handling and preparation, setting the stage for more detailed process analysis in subsequent DFD levels.

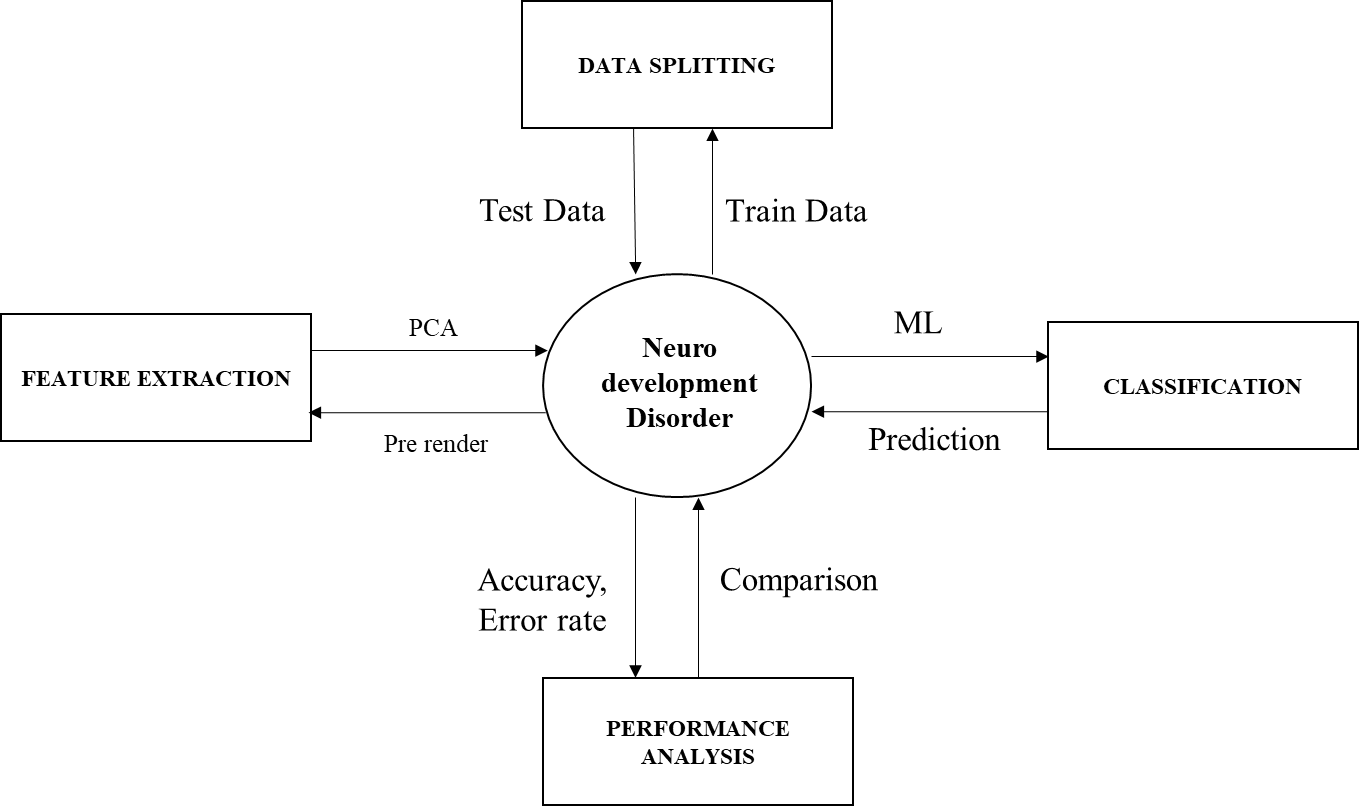
## Level 1:

****

**FIGURE 3.2: DFD 1**

The Data Flow Diagram (DFD) Level 1 expands on the high-level overview by detailing the specific processes involved in the stock prediction system. It begins with **Data Selection**, where the stock dataset in CSV format is imported. This data is then passed to the **Data Preprocessing** module, which handles missing values and applies label encoding. The diagram continues with **Feature selection**, where chi-square select the best features and enhance data quality. Finally, **Data Splitting** divides the dataset into training and testing sets to prepare for model training and evaluation. The DFD Level 1 visually represents how data flows through each process and how these processes interconnect to facilitate accurate diabetes prediction.

## Level 2:

****

**FIGURE 3.3: DFD 2**

The Data Flow Diagram (DFD) Level 2 provides a detailed view of the stock prediction system. It starts with **Data Selection** of the CSV dataset, followed by **Data Preprocessing** to handle missing values and apply label encoding. **Feature Extraction** using PCA reduces dimensionality, and **Data Splitting** divides the data into training and testing sets. **Classification** involves training models like different DL. **Result Generation** assesses model accuracy, and **Prediction** determines if an individual is affected by stock based on the model outputs.

## UML DIAGRAMS:

UML stands for Unified Modelling Language. UML is a standardized general-purpose modelling language in the field of object-oriented software engineering. The standard is managed, and was created by, the Object Management Group.

The goal is for UML to become a common language for creating models of object oriented computer software. In its current form UML is comprised of two major components: a Meta-model and a notation. In the future, some form of method or process may also be added to; or associated with, UML.

The Unified Modelling Language is a standard language for specifying, Visualization, Constructing and documenting the artifacts of software system, as well as for business modelling and other non-software systems.

The UML represents a collection of best engineering practices that have proven successful in the modelling of large and complex systems. The UML is a very important part of developing objects oriented software and the software development process. The UML uses mostly graphical notations to express the design of software projects.

## GOALS:

The Primary goals in the design of the UML are as follows:

1. Provide users a ready-to-use, expressive visual modeling Language so that they can develop and exchange meaningful models.
2. Provide extendibility and specialization mechanisms to extend the core concepts.
3. Be independent of particular programming languages and development process.
4. Provide a formal basis for understanding the modeling language.
5. Encourage the growth of OO tools market.

## USE CASE DIAGRAM:

Use-case diagrams describe the high-level functions and scope of a system. These diagrams also identify the interactions between the system and its actors. The use cases and actors in use-case diagrams describe what the system does and how the actors use it, but not how the system operates internally.

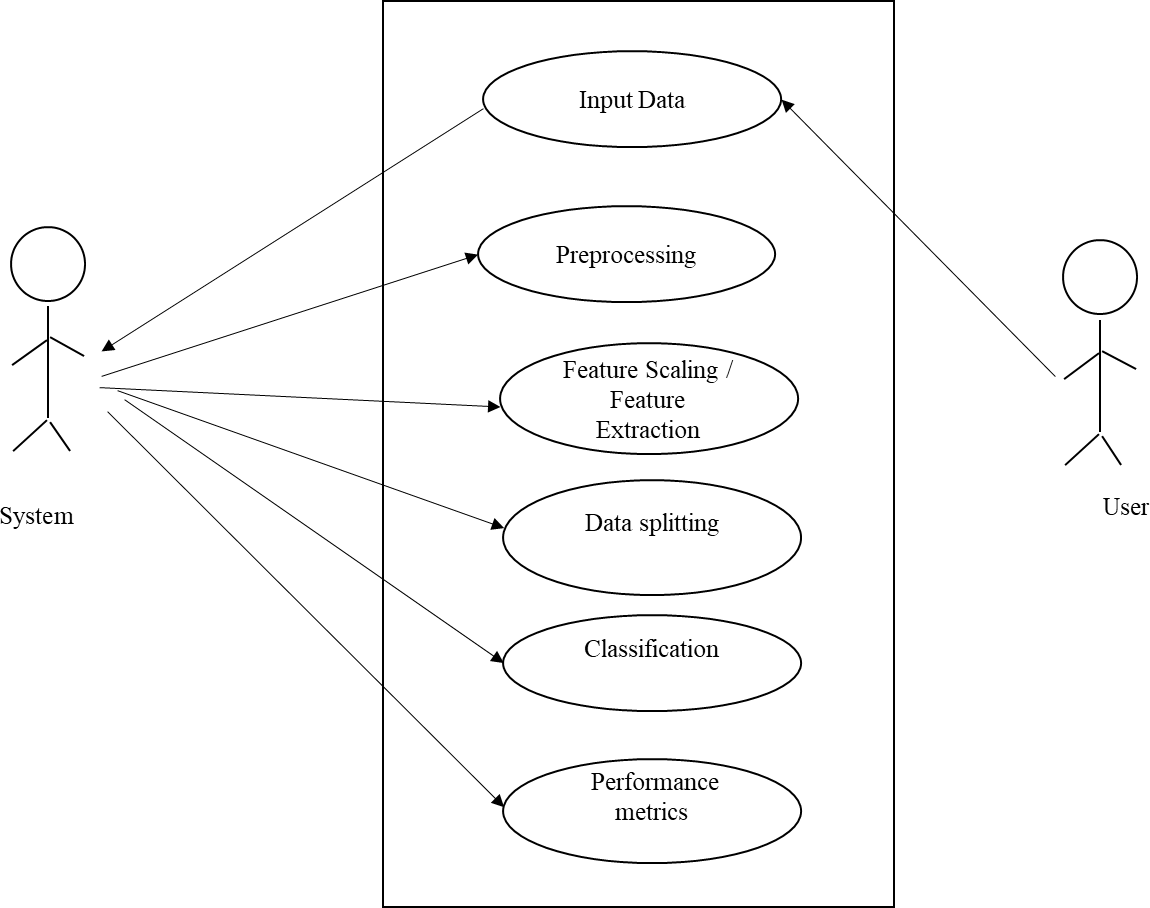
A use case is a list of actions or event steps typically defining the interactions between a role (known in the Unified Modelling Language (UML) as an actor) and a system to achieve a goal. The actor can be a human or other external system.

UML use case diagrams are ideal for:

* Representing the goals of system-user interactions
* Defining and organizing functional requirements in a system
* Specifying the context and requirements of a system
* Modelling the basic flow of events in a use case

## Notations:

* + **Use cases**: Horizontally shaped ovals that represent the different uses that a user might have.
  + **Actors**: Stick figures that represent the people actually employing the use cases.
  + **Associations**: A line between actors and use cases. In complex diagrams, it is important to know which actors are associated with which use cases.
  + **System boundary boxes**: A box that sets a system scope to use cases. All use cases outside the box would be considered outside the scope of that system. For example, Psycho Killer is outside the scope of occupations in the chainsaw example found below.
  + **Packages**: A UML shape that allows you to put different elements into groups. Just as with component diagrams, these groupings are represented as file folders.



**FIGURE 3.4.1: USE CASE DIAGRAM**

The use case diagram depicts the interaction between users and the ND prediction system. It features key actors, including Healthcare Providers and Data Analysts, who interact with the system to achieve specific goals. Healthcare Providers input patient data and receive predictions about stock price, utilizing the system's Data Selection and Prediction capabilities. Data Analysts focus on Data Preprocessing, Feature Extraction, and Model Evaluation to refine the predictive models. The diagram highlights the system's core functionalities—such as data handling, model training, and result generation—and how these functionalities support users in managing and predicting stock outcomes effectively.

## ACTIVITY DIAGRAM:

This shows the flow of events within the system. The activities that occur within a use case or within an objects behaviour typically occur in a sequence. An activity diagram is designed to be simplified look at what happens during an operations or a process. Each activity is represented by a rounded rectangle the processing within an activity goes to compilation and then an automatic transmission to the next activity occurs. An arrow represents the transition from one activity to the next. An activity diagram describes a system in terms of activities. Activities are the state that represents the execution of a set of operations.

These are similar to flow chart diagram and dataflow.

**Initial state**: which state is starting the process?

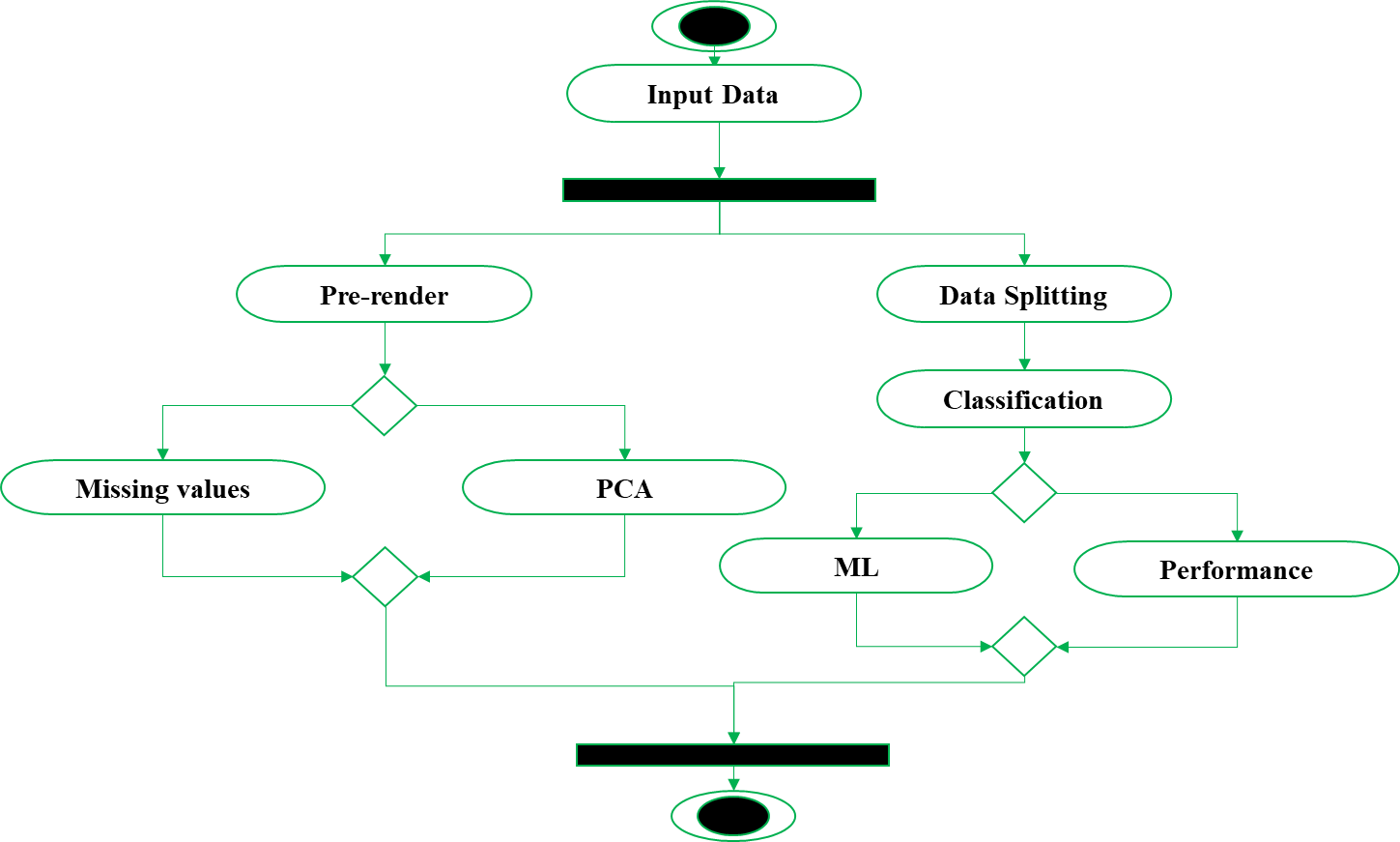
**Action State**: An action state represents the execution of an atomic action, typically the invocation of an operation. An action state is a simple state with an entry action whose only exit transition is triggered by the implicit event of completing the execution of the entry action.

**Transition**: A transition is a directed relationship between a source state vertex and a target state vertex. It may be part of a compound transition, which takes the static machine from one static configuration to another, representing the complete response of the static machine to a particular event instance.

**Final state:** A final state represents the last or "final" state of the enclosing composite state. There may be more than one final state at any level signifying that the composite state can end in different ways or conditions.

When a final state is reached and there are no other enclosing states it means that the entire state machine has completed its transitions and no more transitions can occur.

**Decision**: A state diagram (and by derivation an activity diagram) expresses decision when guard conditions are used to indicate different possible transitions that depend on Boolean conditions of the owning object.



**FIGURE 3.4.2: ACTIVITY DIAGRAM**

The activity diagram outlines the sequential steps involved in the diabetes prediction process. It starts with Data Collection, where the dataset is imported and prepared. The next phase, Data Preprocessing, addresses missing values and applies label encoding. Following this, Feature selection using chi-square is performed to reduce data complexity. The dataset is then Split into training and testing sets. Model Training occurs with different DL algorithms followed by Model Evaluation to measure performance through accuracy metrics. Each step ensures a systematic approach to achieving accurate stock predictions.

## SEQUENCE DIAGRAM:

Sequence diagrams document the interactions between classes to achieve a result, such as a use case. Because UML is designed for object-oriented programming, these communications between classes are known as messages. The Sequence diagram lists objects horizontally, and time vertically, and models these messages over time.

**Graphical Notation**: In a Sequence diagram, classes and actors are listed as columns, with vertical lifelines indicating the lifetime of the object over time.

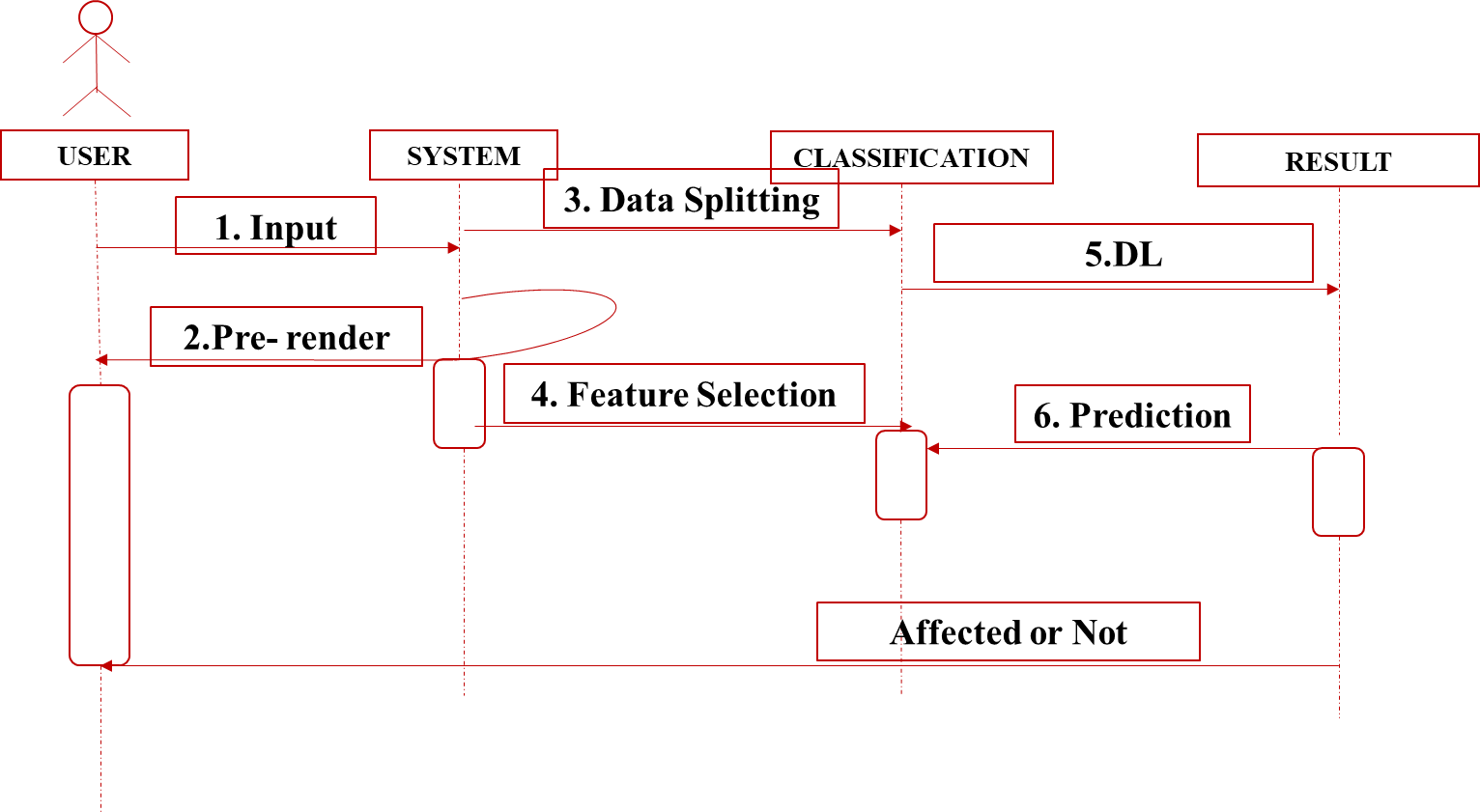
**Object**: Objects are instances of classes, and are arranged horizontally. The pictorial representation for an Object is a class (a rectangle) with the name prefixed by the object.

**Lifeline** The Lifeline identifies the existence of the object over time. The notation 2for a Lifeline is a vertical dotted line extending from an object.

**Activation**: Activations, modelled as rectangular boxes on the lifeline, indicate when the object is performing an action.

**Message**: Messages, modelled as horizontal arrows between Activations.

****



**FIGURE 3.4.3: SEQUENCE DIAGRAM**

The sequence diagram illustrates the step-by-step interactions between different components of the diabetes prediction system. It begins with the User initiating the process by uploading the stock dataset. The system first performs Data Preprocessing, including handling missing values and applying label encoding. Next, Feature selection is executed using chi-square to optimize data for analysis. The data is then split into Training and Testing sets. The Model Training phase follows, where algorithms like different DL models are employed. After training, the Model Evaluation component assesses performance metrics. Each interaction ensures that data flows seamlessly through the system for accurate predictions.

## ER DIAGRAM:

An Entity Relationship (ER) Diagram is a type of flowchart that illustrates how “entities” such as people, objects or concepts relate to each other within a system.

ER Diagrams are most often used to design or debug relational databases in the fields of software engineering, business information systems, education and research.

Also known as ERDs or ER Models, they use a defined set of symbols such as rectangles, diamonds, ovals and connecting lines to depict the interconnectedness of entities, relationships and their attributes.

They mirror grammatical structure, with entities as nouns and relationships as verbs.

## Notation:

**Entity**

A definable thing—such as a person, object, concept or event—that can have data stored about it. Think of entities as nouns. Examples: a customer, student, car or product. Typically shown as a rectangle.

**Entity type:** A group of definable things, such as students or athletes, whereas the entity would be the specific student or athlete. Other examples: customers, cars or products.

**Entity set:** Same as an entity type, but defined at a particular point in time, such as students enrolled in a class on the first day.

Other examples: Customers who purchased last month, cars currently registered in Florida. A related term is instance, in which the specific person or car would be an instance of the entity set.

**Entity categories:** Entities are categorized as strong, weak or associative. A **strong entity** can be defined solely by its own attributes, while a **weak entity** cannot. An associative entity associates entities (or elements) within an entity set.

**Entity keys:** Refers to an attribute that uniquely defines an entity in an entity set. Entity keys can be super, candidate or primary. **Super key:** A set of attributes (one or more) that together define an entity in an entity set.

**Candidate key:** A minimal super key, meaning it has the least possible number of attributes to still be a super key. An entity set may have more than one candidate key. **Primary key:** A candidate key chosen by the database designer to uniquely identify the entity set. **Foreign key:** Identifies the relationship between entities.

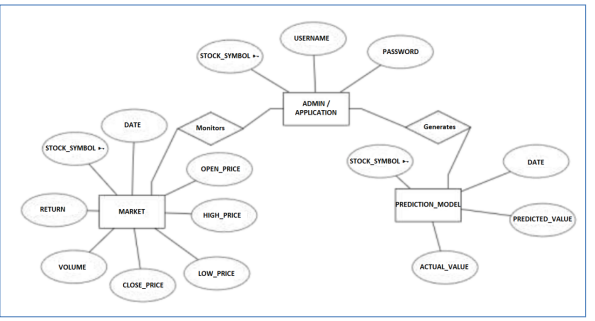
## Relationship

How entities act upon each other or are associated with each other. Think of relationships as verbs.

For example, the named student might register for a course.

The two entities would be the student and the course, and the relationship depicted is the act of enrolling, connecting the two entities in that way.

Relationships are typically shown as diamonds or labels directly on the connecting lines.



**FIGURE 3.4.4: ER DIAGRAM**

The Entity-Relationship (ER) diagram outlines the key entities and their relationships within the stock price prediction system. It features entities such as Stock Data, Technical Indicators, Preprocessing Methods, and Prediction Models. Stock Data includes attributes like opening price, closing price, trading volume, and stock symbol, while Technical Indicators encompass calculated metrics such as Moving Averages, Relative Strength Index (RSI), and MACD. The Preprocessing Methods entity details steps like handling missing values, normalization, and feature scaling, which are linked to the Prediction Models entity. Relationships between these entities illustrate how data flows from collection and preprocessing through model training and evaluation, ultimately leading to stock price predictions. The diagram provides a clear overview of how data entities interact to support accurate and efficient stock market analysis and forecasting.

## 3.3.5 CLASS DIAGRAM:

Class diagrams identify the class structure of a system, including the properties and methods of each class. Also depicted are the various relationships that can exist between classes, such as an inheritance relationship.

Part of the popularity of Class diagrams stems from the fact that many CASE tools, such as Rational XDE, will auto-generate code in a variety of languages, these tools can synchronize models and code, reducing the workload, and can also generate Class diagrams from object-oriented code.

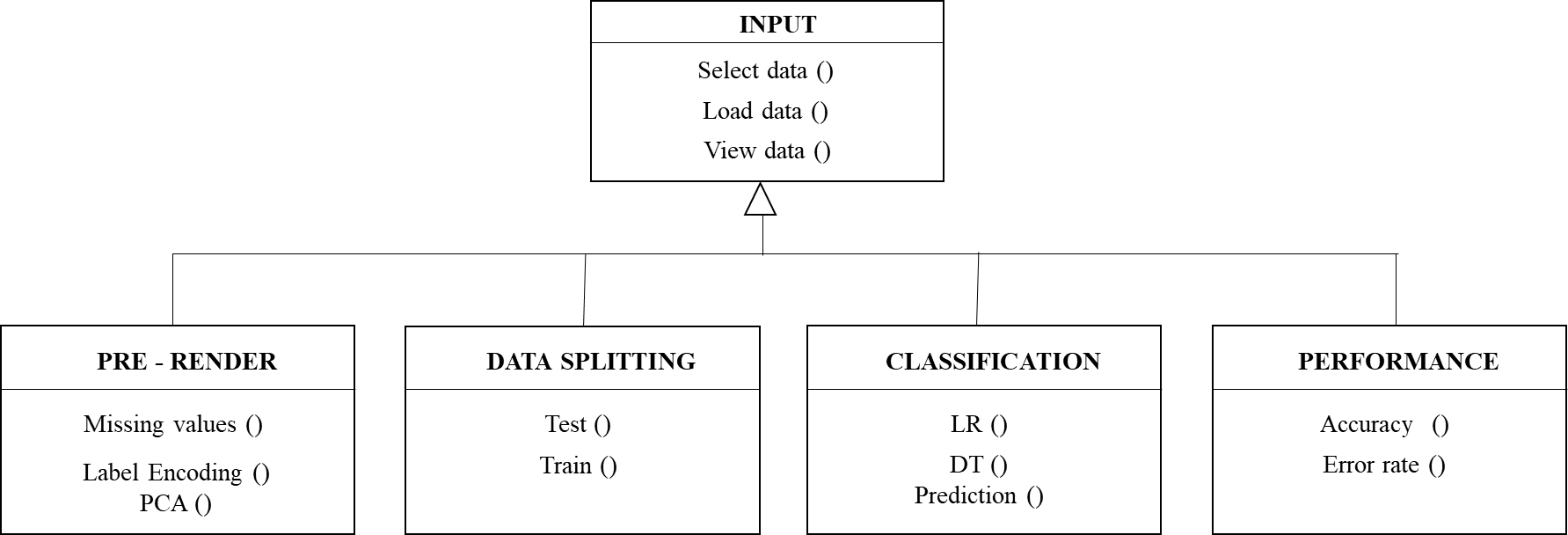
**Graphical Notation:** The elements on a Class diagram are classes and the relationships between them.

**Class**: Classes are building blocks in object-oriented programming. A class is depicted using a rectangle divided into three section.

The top section is name of class; the middle section defines the properties of class. The bottom section list the methods of the class.

**Association:** An Association is a generic relationship between two classes, and is modelled by a line connecting the two classes.

This line can be qualified with the type of relationship, and can also feature multiplicity rule (e.g. one-to-one, one-to-many, many-to-many) for the relationship.



**FIGURE 3.4.5: CLASS DIAGRAM**

The class diagram represents the structure of the stock prediction system by defining the key classes and their relationships. Central classes include Dataset, which handles data import and preprocessing, and Model, which encompasses different deep learning algorithms. The Pre-processor class manages tasks such as handling missing values and applying label encoding, while the Feature selector class is responsible for techniques like chi-square. The Evaluator class assesses model performance through metrics like accuracy. Relationships between these classes illustrate how data is processed and utilized, from initial collection and preparation to training and evaluation, ensuring a cohesive approach to predicting stock outcomes.

## CHAPTER 4 IMPLEMENTATION

* 1. **MODULES:**
     + Data Acquisition
     + Data Preprocessing
     + Feature Extraction
     + Data Splitting
     + Model Classification
     + Result Generation
     + Prediction

## MODULES DESCRIPTION:

* + 1. **DATA ACQUISITION**
       - The first step in the methodology involves acquiring the Stock datasets from a reliable data repository.
       - These datasets are usually available in formats such as .csv or .xlsx, which are compatible with data processing tools and machine learning frameworks.
       - It is crucial to ensure that the datasets are obtained from reputable sources to maintain data quality and reliability.
       - The data may include various features relevant to stock market prediction, such as historical stock prices (opening, closing, high, low), trading volumes, technical indicators (e.g., Moving Averages, RSI, MACD), and external factors like economic indicators and market sentiment.
       - Proper documentation and understanding of the dataset's structure and attributes are essential for subsequent processing and analysis.

## DATA PRE-PROCESSING:

1. **Handling Missing Data:**
   * **Imputation Techniques:** Missing data can be addressed using imputation techniques such as mean imputation, median imputation, or more advanced methods like k-nearest neighbors (KNN) imputation. Imputation helps in filling in missing values based on the statistical properties or patterns observed in the existing data.
   * **Removal Strategies:** Alternatively, rows with missing values can be removed if the proportion of missing data is minimal and does not significantly affect the dataset’s integrity. This approach is straightforward but may lead to loss of valuable information if many rows are affected.

## Label Encoding:

* + Categorical variables in the dataset may need to be converted into numerical format to be suitable for machine learning algorithms. Label encoding assigns a unique integer to each category. For instance, if a feature "Stock Trend" has values "Up" and "Down," these might be encoded as 0 and 1, respectively. This transformation is necessary for models that require numerical input but can be supplemented with one-hot encoding for models needing a binary or multi-class representation. For example, one-hot encoding could be used to represent "Up," "Down," and "Neutral" as three separate binary features.

## FEATURE EXTRACTION:

Principal Component Analysis (PCA) is a statistical technique used for dimensionality reduction while preserving as much variability as possible. Here’s a brief overview of its key concepts and steps:

Key Concepts:

* + - 1. **Dimensionality Reduction**: PCA helps reduce the number of variables in a dataset while retaining its essential features, making it easier to visualize and analyze.
      2. **Variance**: PCA identifies the directions (principal components) along which the variance of the data is maximized.
      3. **Eigenvalues and Eigenvectors**: The principal components are derived from the eigenvectors of the covariance matrix of the data. Eigenvalues indicate the amount of variance captured by each principal component.

Steps in PCA:

1. **Standardization**: If variables have different units or scales, standardize the data (subtract the mean and divide by the standard deviation).
2. **Covariance Matrix Computation**: Calculate the covariance matrix to understand how variables relate to one another.
3. **Eigenvalue and Eigenvector Calculation**: Compute the eigenvalues and eigenvectors of the covariance matrix.
4. **Principal Component Selection**: Sort the eigenvalues and choose the top kkk eigenvalues and their corresponding eigenvectors, which represent the principal components.
5. **Projection**: Project the original data onto the new subspace defined by the selected principal components.

Applications:

* **Data Visualization**: Reducing dimensions helps in visualizing high- dimensional data in 2D or 3D plots.
* **Noise Reduction**: By eliminating less significant components, PCA can help reduce noise in the data.
* **Feature Extraction**: It helps in identifying the most important features that explain the variability in the data.

## DATA SPLITTING:

* + - 1. **Training and Testing Sets:**
         * The dataset is divided into training and testing subsets. Typically, a common split is 80% of the data for training and 20% for testing, though this can vary based on dataset size and requirements.
         * **Training Set:** Used to develop and train the machine learning models. This set allows the model to learn the underlying patterns and relationships in the data.
         * **Testing Set:** Used to evaluate the performance of the trained model. This set helps in assessing how well the model generalizes to unseen data and provides metrics for evaluating its accuracy and effectiveness.

## MODEL CLASSIFICATION:

A **decision tree** is a popular machine learning model used for classification and regression tasks. It represents decisions and their possible consequences as a tree-like structure. Here’s an overview of its main components and how it works:

Key Components:

* + - 1. **Root Node**: The top node of the tree that represents the entire dataset. It splits into branches based on feature values.
      2. **Branches**: These are the connections between nodes, representing the outcome of a decision made at a parent node.
      3. **Internal Nodes**: Nodes that represent tests on features. Each internal node splits the data into subsets based on a feature's value.
      4. **Leaf Nodes**: The terminal nodes that represent the final decision or prediction (class labels in classification or values in regression).

How It Works:

1. **Splitting**: The algorithm selects the best feature to split the data at each node. This is usually done using criteria like:
   * **Gini Impurity**: Measures how often a randomly chosen element would be incorrectly labeled if it was randomly labeled according to the distribution of labels in the subset.
   * **Entropy**: Measures the disorder or impurity in the dataset. The goal is to reduce entropy after each split.
   * **Mean Squared Error (MSE)**: Used in regression trees to minimize the variance in the predicted values.
2. **Pruning**: To avoid overfitting, decision trees can be pruned. This involves removing branches that have little importance, helping to improve the model's generalization on unseen data.
3. **Decision Making**: Once the tree is constructed, it can be used to make predictions. Input features are passed through the tree, following the branches until a leaf node is reached, which gives the output.

Advantages:

* **Interpretability**: Decision trees are easy to understand and visualize.
* **Non-Parametric**: They don’t assume a specific distribution of the data.
* **Feature Importance**: They can provide insights into which features are most important for making predictions.

**Logistic regression** is a statistical method used for binary classification problems, where the goal is to predict the probability that a given input belongs to a particular category. It is a type of regression analysis used when the dependent variable is categorical.

Key Concepts:

1. **Binary Outcome**: Logistic regression predicts the probability of a binary outcome (e.g., yes/no, success/failure).

How It Works:

1. **Modeling**: Logistic regression estimates the relationship between the independent variables (features) and the log-odds of the dependent variable.
2. **Training**: The model is trained using maximum likelihood estimation (MLE), which finds the parameter values that maximize the likelihood of observing the given data.
3. **Thresholding**: To classify observations, a threshold (often 0.5) is applied to the predicted probabilities. If the predicted probability exceeds the threshold, the observation is classified into one category; otherwise, it falls into the other.

Advantages:

* **Simplicity**: Easy to implement and interpret, making it suitable for binary classification.
* **Efficiency**: Requires less computational power compared to more complex models.
* **Probabilistic Output**: Provides probabilities that can be useful for understanding confidence in predictions.

## : RESULT GENERATION:

The Final Result will get generated based on the overall classification and prediction. The performance of this proposed approach is evaluated using some measures like,

## Accuracy

Accuracy of classifier refers to the ability of classifier. It predicts the class label correctly and the accuracy of the predictor refers to how well a given predictor can guess the value of predicted attribute for a new data.

AC= (TP+TN)/ (TP+TN+FP+FN)

## Precision

Precision is defined as the number of true positives divided by the number of true positives plus the number of false positives.

Precision=TP/ (TP+FP)

## Recall

Recall is the number of correct results divided by the number of results that should have been returned. In binary classification, recall is called sensitivity. It can be viewed as the probability that a relevant document is retrieved by the query.

Recall=TP/ (TP+FN)

## Prediction:

* After training and evaluating the models, the system is used to predict user input.
* The trained prediction models (e.g., LSTM, CNN, Prophet) are used to forecast stock price movements based on historical data, technical indicators, and market conditions. The models evaluate the input data to predict whether the stock price will increase, decrease, or remain neutral.
* **Application:** The output provides actionable insights for investors and traders, enabling informed decision-making and optimized trading strategies for individuals and organizations in the stock market.

## CHAPTER 5 SYSTEM REQUIREMENTS

* 1. **HARDWARE REQUIREMENTS:**
     + System : Pentium IV 2.4 GHz
     + Hard Disk : 200 GB
     + Mouse : Logitech.
     + Keyboard : 110 keys enhanced
     + Ram : 4GB

## SOFTWARE REQUIREMENTS:

* + - O/S : Windows 10.
    - Language : Python
    - Front End : Streamlit - Framework
    - Software used :PyCharm - Python

## SOFTWARE DESCRIPTION:

* + 1. **Python**

Python is one of those rare languages which can claim to be both *simple* and powerful. You will find yourself pleasantly surprised to see how easy it is to concentrate on the solution to the problem rather than the syntax and structure of the language you are programming in. The official introduction to Python is Python is an easy to learn, powerful programming language. It has efficient high-level data structures and a simple but effective approach to object-oriented programming. Python's elegant syntax and dynamic typing, together with its interpreted nature, make

it an ideal language for scripting and rapid application development in many areas on most platforms. I will discuss most of these features in more detail in the next section.

## Features of Python

* + **Simple**

Python is a simple and minimalistic language. Reading a good Python program feels almost like reading English, although very strict English! This pseudo-code nature of Python is one of its greatest strengths. It allows you to concentrate on the solution to the problem rather than the language itself.

## Easy to Learn

As you will see, Python is extremely easy to get started with. Python has an extraordinarily simple syntax, as already mentioned.

## Free and Open Source

Python is an example of a *FLOSS* (Free/Libré and Open Source Software). In simple terms, you can freely distribute copies of this software, read its source code, make changes to it, and use pieces of it in new free programs. FLOSS is based on the concept of a community which shares knowledge. This is one of the reasons why Python is so good - it has been created and is constantly improved by a community who just want to see a better Python.

## High-level Language

When you write programs in Python, you never need to bother about the low- level details such as managing the memory used by your program, etc.

## Portable

Due to its open-source nature, Python has been ported to (i.e. changed to make it work on) many platforms. All your Python programs can work on any of these platforms without requiring any changes at all if you are careful enough to avoid any system-dependent features.

You can use Python on GNU/Linux, Windows, FreeBSD, Macintosh, Solaris, OS/2, Amiga, AROS, AS/400, BeOS, OS/390, z/OS, Palm OS, QNX, VMS, Psion,

Acorn RISC OS, VxWorks, PlayStation, Sharp Zaurus, Windows CE and PocketPC!

You can even use a platform like [Kivy](http://kivy.org/) to create games for your computer *and*

for iPhone, iPad, and Android.

## Interpreted

This requires a bit of explanation.

A program written in a compiled language like C or C++ is converted from the source language i.e. C or C++ into a language that is spoken by your computer (binary code i.e. 0s and 1s) using a compiler with various flags and options. When you run the program, the linker/loader software copies the program from hard disk to memory and starts running it.

Python, on the other hand, does not need compilation to binary. You just *run* the program directly from the source code. Internally, Python converts the source code into an intermediate form called bytecodes and then translates this into the native language of your computer and then runs it. All this, actually, makes using Python much easier since you don't have to worry about compiling the program, making sure that the proper libraries are linked and loaded, etc. This also makes your Python programs much more portable, since you can just copy your Python program onto another computer and it just works!

## Object Oriented

Python supports procedure-oriented programming as well as object-oriented programming. In *procedure-oriented* languages, the program is built around procedures or functions which are nothing but reusable pieces of programs. In *object- oriented* languages, the program is built around objects which combine data and functionality. Python has a very powerful but simplistic way of doing OOP, especially when compared to big languages like C++ or Java.

## Extensible

If you need a critical piece of code to run very fast or want to have some piece of algorithm not to be open, you can code that part of your program in C or C++ and then use it from your Python program.

## Embeddable

You can embed Python within your C/C++ programs to give *scripting*

capabilities for your program's users.

## Extensive Libraries

The Python Standard Library is huge indeed. It can help you do various things involving regular expressions, documentation generation, unit testing, threading, databases, web browsers, CGI, FTP, email, XML, XML-RPC, HTML, WAV files, cryptography, GUI (graphical user interfaces), and other system-dependent stuff. Remember, all this is always available wherever Python is installed. This is called the *Batteries Included* philosophy of Python.

Besides the standard library, there are various other high-quality libraries which you can find at the Python Package Index.

## Streamlit framework:

Streamlit is an open-source Python library that makes it easy to create and share beautiful, custom web apps for machine learning and data science. It allows you to build interactive web applications straight from Python scripts. Here are some of the key features and concepts related to Streamlit:

## Key Features

* + Ease of Use: Streamlit enables you to create web applications by writing only Python code. You don’t need any HTML, CSS, or JavaScript knowledge.
  + Real-time Interactivity: Streamlit apps can automatically update as users interact with widgets like sliders, dropdowns, and text inputs.
  + Data Visualization: It integrates seamlessly with popular data visualization libraries like Matplotlib, Plotly, and Altair.
  + Widgets: Streamlit provides a variety of widgets to collect user input and make your apps interactive.
  + Deployment: Streamlit apps can be deployed easily on the web through services like Streamlit Sharing, Heroku, or AWS.

## Basic Concepts

* + Script Execution: Streamlit runs your entire script from top to bottom each time you interact with a widget, which means the state of your app is reset each time unless you use caching.
  + Widgets: Widgets like sliders, text inputs, buttons, and checkboxes allow users to provide input.
  + Layout: You can organize the layout of your app using layout primitives like st.sidebar, st.columns, and st.expander.

## TESTING PRODUCTS:

System testing is the stage of implementation, which aimed at ensuring that system works accurately and efficiently before the live operation commence. Testing is the process of executing a program with the intent of finding an error. A good test case is one that has a high probability of finding an error. A successful test is one that answers a yet undiscovered error.

Testing is vital to the success of the system. System testing makes a logical assumption that if all parts of the system are correct, the goal will be successfully achieved. . A series of tests are performed before the system is ready for the user acceptance testing. Any engineered product can be tested in one of the following ways. Knowing the specified function that a product has been designed to from, test can be conducted to demonstrate each function is fully operational. Knowing the internal working of a product, tests can be conducted to ensure that “al gears mesh”, that is the internal operation of the product performs according to the specification and all internal components have been adequately exercised.

## UNIT TESTING:

Unit testing is the testing of each module and the integration of the overall system is done. Unit testing becomes verification efforts on the smallest unit of software design in the module. This is also known as ‘module testing’.

The modules of the system are tested separately. This testing is carried out during the programming itself. In this testing step, each model is found to be working satisfactorily as regard to the expected output from the module. There are some validation checks for the fields. For example, the validation check is done for verifying the data given by the user where both format and validity of the data entered is included. It is very easy to find error and debug the system.

## INTEGRATION TESTING:

Data can be lost across an interface, one module can have an adverse effect on the other sub function, when combined, may not produce the desired major function. Integrated testing is systematic testing that can be done with sample data. The need for the integrated test is to find the overall system performance. There are two types of integration testing. They are:

* + - 1. Top-down integration testing.
      2. Bottom-up integration testing.

## TESTING TECHNIQUES/STRATEGIES:

* + **WHITE BOX TESTING:**

White Box testing is a test case design method that uses the control structure of the procedural design to drive cases. Using the white box testing methods, we

Derived test cases that guarantee that all independent paths within a module have been exercised at least once.

## BLACK BOX TESTING:

1. Black box testing is done to find incorrect or missing function
2. Interface error
3. Errors in external database access
4. Performance errors.
5. Initialization and termination errors

In ‘functional testing’, is performed to validate an application conforms to its

specifications of correctly performs all its required functions. So this testing is also called ‘black box testing’. It tests the external behaviour of the system. Here the engineered product can be tested knowing the specified function that a product has been designed to perform, tests can be conducted to demonstrate that each function is fully operational.

## SOFTWARE TESTING STRATEGIES

**VALIDATION TESTING:**

After the culmination of black box testing, software is completed assembly as a package, interfacing errors have been uncovered and corrected and final series of software validation tests begin validation testing can be defined as many,

But a single definition is that validation succeeds when the software functions in a manner that can be reasonably expected by the customer.

## USER ACCEPTANCE TESTING:

User acceptance of the system is the key factor for the success of the system. The system under consideration is tested for user acceptance by constantly keeping in touch with prospective system at the time of developing changes whenever required.

## OUTPUT TESTING:

After performing the validation testing, the next step is output asking the user about the format required testing of the proposed system, since no system could be useful if it does not produce the required output in the specific format. The output displayed or generated by the system under consideration. Here the

output format is considered in two ways. One is screen and the other is printed format. The output format on the screen is found to be correct as the format was designed in the system phase according to the user needs. For the hard copy also output comes out as the specified requirements by the user. Hence the output testing does not result in any connection in the system.

## TEST CASES:

**Test Case 1: Handling Missing Values**

**Objective:** Verify that the system correctly handles missing values in the dataset.

## Input:

* + - Stock Price dataset with a column that has missing values.

## Steps:

1. Load the dataset with missing values.
2. Execute the pre-processing step to check for and handle missing values.
3. Verify that missing values are addressed appropriately (e.g., imputation, removal).

## Expected Result:

* + - The missing values are either imputed or rows with missing values are removed without causing errors, and the system proceeds to the next steps in the pipeline.

## Test Case 2: Label Encoding Verification

**Objective:** Ensure that label encoding correctly converts string labels into numeric values.

## Input:

* + - A dataset where labels are strings (e.g., “up,” “down”)

## Steps:

1. Perform label encoding on the string labels.
2. Check the encoded values for correctness (e.g., “up” -> 0, “down” -> 1).

## Expected Result:

* + - String labels are accurately converted to numeric values according to the defined encoding scheme.

## Test Case 3: Model Training and Prediction Accuracy

**Objective:** Assess the accuracy of the deep learning models on the test data.

## Input:

* + - A split dataset with training and test sets, pre-processed text data.

## Steps:

1. Train the LR and DT on the training data.
2. Make predictions using the test data.
3. Evaluate the performance using metrics such as accuracy, precision, recall, F1- score, and error rate.

## Expected Result:

* + - The models should produce predictions and the performance metrics should meet the expected standards based on predefined thresholds (e.g., accuracy > 70%).

## Test Case 4: Performance Comparison and Effectiveness

**Objective:** Confirm that the proposed method shows an improvement in accuracy compared to previous methods or baseline.

## Input:

* + - Baseline method results and proposed system results.

## Steps:

1. Compare the accuracy of the proposed method with that of a baseline method or previous approach.
2. Analyze the improvement in accuracy and other metrics.

## Expected Result:

* + - The proposed method should demonstrate improved accuracy and performance metrics compared to the baseline method, indicating the effectiveness of the improvements.

## CHAPTER 6 CONCLUSION

In conclusion, this study highlights the effectiveness of combining machine learning models like Long Short-Term Memory (LSTM) networks and Convolutional Neural Networks (CNN) with Principal Component Analysis (PCA) in the predictive modeling of stock market trends. By leveraging these methodologies, we successfully identified key features, such as historical stock prices and technical indicators, that contribute to accurate stock price forecasting, enhancing both the interpretability and performance of our models. The LSTM network captured temporal dependencies in stock price movements, while the CNN identified intricate patterns within the data. Our findings demonstrate the potential of these predictive analytics tools in financial settings, providing investors and traders with the ability to make data-driven decisions and optimize trading strategies. Moreover, this research underscores the value of integrating deep learning techniques and feature engineering in financial forecasting. Ultimately, the combination of these approaches opens up possibilities for developing advanced, real-time predictive tools that could significantly improve stock market analysis and decision-making.

## CHAPTER 7

**FUTURE ENHANCEMENT**

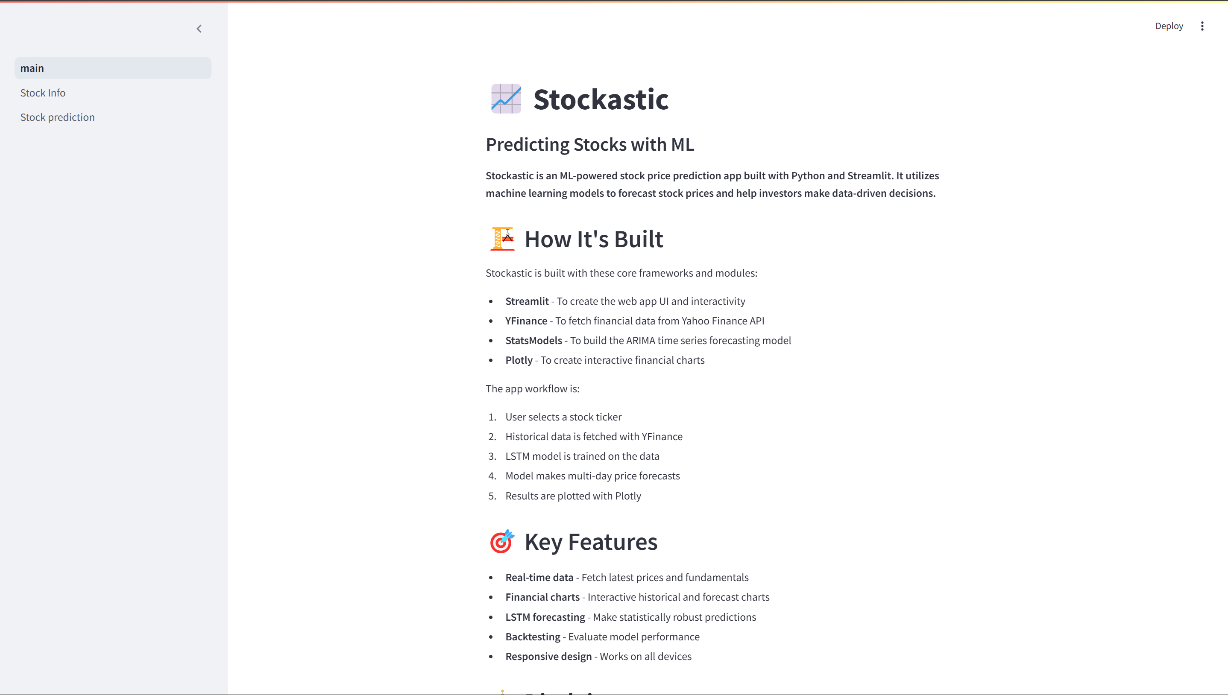
Moving forward, several avenues for improvement and expansion exist in this domain. Firstly, incorporating more advanced feature extraction techniques beyond PCA, such as feature engineering or deep learning-based feature learning, could potentially capture more intricate relationships within the stock market data. Additionally, exploring ensemble methods or more complex neural network architectures, such as Transformers or Attention Mechanisms, could further enhance prediction accuracy and robustness. Addressing issues in the dataset through techniques like data augmentation or employing more advanced sampling strategies can improve model performance, especially in cases where market data is skewed or contains anomalies. Furthermore, integrating real-time financial data streams from APIs or market sensors could enable continuous monitoring and adaptive trading strategies, allowing for proactive responses to changing market conditions. Lastly, expanding the scope to include predictive analytics for factors such as market sentiment, external economic indicators, and geopolitical events would offer a more comprehensive approach to stock market forecasting. By continually refining our methodologies and embracing emerging technologies, we aim to advance the efficacy and applicability of AI and machine learning in financial forecasting, ultimately improving trading decisions and financial outcomes for investors.

## CHAPTER 8 SAMPLE CODING

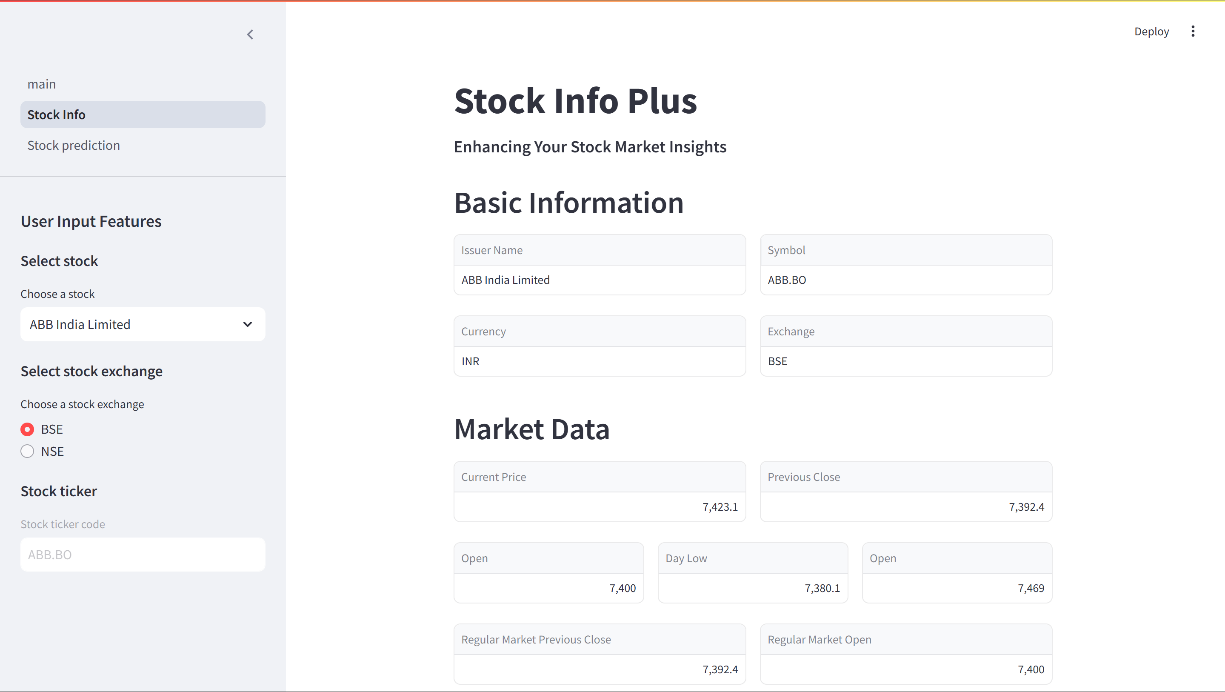
import datetime as dt  
import os  
from pathlib import Path  
  
# Import pandas  
import pandas as pd  
  
# Import yfinance  
import yfinance as yf  
  
# Import the required libraries  
import tf.keras.layers.LSTM as lstm  
  
  
# Create function to fetch stock name and id  
def fetch\_stocks():  
 # Load the data  
 df = pd.read\_csv(Path.cwd() / "data" / "equity\_issuers.csv")  
  
 # Filter the data  
 df = df[["Security Code", "Issuer Name"]]  
  
 # Create a dictionary  
 stock\_dict = dict(zip(df["Security Code"], df["Issuer Name"]))  
  
 # Return the dictionary  
 return stock\_dict  
  
  
# Create function to fetch periods and intervals  
def fetch\_periods\_intervals():  
 # Create dictionary for periods and intervals  
 periods = {  
 "1d": ["1m", "2m", "5m", "15m", "30m", "60m", "90m"],  
 "5d": ["1m", "2m", "5m", "15m", "30m", "60m", "90m"],  
 "1mo": ["30m", "60m", "90m", "1d"],  
 "3mo": ["1d", "5d", "1wk", "1mo"],  
 "6mo": ["1d", "5d", "1wk", "1mo"],  
 "1y": ["1d", "5d", "1wk", "1mo"],  
 "2y": ["1d", "5d", "1wk", "1mo"],  
 "5y": ["1d", "5d", "1wk", "1mo"],  
 "10y": ["1d", "5d", "1wk", "1mo"],  
 "max": ["1d", "5d", "1wk", "1mo"],  
 }  
  
 # Return the dictionary  
 return periods  
  
  
# Function to fetch the stock info  
def fetch\_stock\_info(stock\_ticker):  
 # Pull the data for the first security  
 stock\_data = yf.Ticker(stock\_ticker)  
  
 # Extract full of the stock  
 stock\_data\_info = stock\_data.info  
  
 # Function to safely get value from dictionary or return "N/A"  
 def safe\_get(data\_dict, key):  
 return data\_dict.get(key, "N/A")  
  
 # Extract only the important information  
 stock\_data\_info = {  
 "Basic Information": {  
 "symbol": safe\_get(stock\_data\_info, "symbol"),  
 "longName": safe\_get(stock\_data\_info, "longName"),  
 "currency": safe\_get(stock\_data\_info, "currency"),  
 "exchange": safe\_get(stock\_data\_info, "exchange"),  
 },  
 "Market Data": {  
 "currentPrice": safe\_get(stock\_data\_info, "currentPrice"),  
 "previousClose": safe\_get(stock\_data\_info, "previousClose"),  
 "open": safe\_get(stock\_data\_info, "open"),  
 "dayLow": safe\_get(stock\_data\_info, "dayLow"),  
 "dayHigh": safe\_get(stock\_data\_info, "dayHigh"),  
 "regularMarketPreviousClose": safe\_get(  
 stock\_data\_info, "regularMarketPreviousClose"  
 ),  
 "regularMarketOpen": safe\_get(stock\_data\_info, "regularMarketOpen"),  
 "regularMarketDayLow": safe\_get(stock\_data\_info, "regularMarketDayLow"),  
 "regularMarketDayHigh": safe\_get(stock\_data\_info, "regularMarketDayHigh"),  
 "fiftyTwoWeekLow": safe\_get(stock\_data\_info, "fiftyTwoWeekLow"),  
 "fiftyTwoWeekHigh": safe\_get(stock\_data\_info, "fiftyTwoWeekHigh"),  
 "fiftyDayAverage": safe\_get(stock\_data\_info, "fiftyDayAverage"),  
 "twoHundredDayAverage": safe\_get(stock\_data\_info, "twoHundredDayAverage"),  
 },  
 "Volume and Shares": {  
 "volume": safe\_get(stock\_data\_info, "volume"),  
 "regularMarketVolume": safe\_get(stock\_data\_info, "regularMarketVolume"),  
 "averageVolume": safe\_get(stock\_data\_info, "averageVolume"),  
 "averageVolume10days": safe\_get(stock\_data\_info, "averageVolume10days"),  
 "averageDailyVolume10Day": safe\_get(  
 stock\_data\_info, "averageDailyVolume10Day"  
 ),  
 "sharesOutstanding": safe\_get(stock\_data\_info, "sharesOutstanding"),  
 "impliedSharesOutstanding": safe\_get(  
 stock\_data\_info, "impliedSharesOutstanding"  
 ),  
 "floatShares": safe\_get(stock\_data\_info, "floatShares"),  
 },  
 "Dividends and Yield": {  
 "dividendRate": safe\_get(stock\_data\_info, "dividendRate"),  
 "dividendYield": safe\_get(stock\_data\_info, "dividendYield"),  
 "payoutRatio": safe\_get(stock\_data\_info, "payoutRatio"),  
 },  
 "Valuation and Ratios": {  
 "marketCap": safe\_get(stock\_data\_info, "marketCap"),  
 "enterpriseValue": safe\_get(stock\_data\_info, "enterpriseValue"),  
 "priceToBook": safe\_get(stock\_data\_info, "priceToBook"),  
 "debtToEquity": safe\_get(stock\_data\_info, "debtToEquity"),  
 "grossMargins": safe\_get(stock\_data\_info, "grossMargins"),  
 "profitMargins": safe\_get(stock\_data\_info, "profitMargins"),  
 },  
 "Financial Performance": {  
 "totalRevenue": safe\_get(stock\_data\_info, "totalRevenue"),  
 "revenuePerShare": safe\_get(stock\_data\_info, "revenuePerShare"),  
 "totalCash": safe\_get(stock\_data\_info, "totalCash"),  
 "totalCashPerShare": safe\_get(stock\_data\_info, "totalCashPerShare"),  
 "totalDebt": safe\_get(stock\_data\_info, "totalDebt"),  
 "earningsGrowth": safe\_get(stock\_data\_info, "earningsGrowth"),  
 "revenueGrowth": safe\_get(stock\_data\_info, "revenueGrowth"),  
 "returnOnAssets": safe\_get(stock\_data\_info, "returnOnAssets"),  
 "returnOnEquity": safe\_get(stock\_data\_info, "returnOnEquity"),  
 },  
 "Cash Flow": {  
 "freeCashflow": safe\_get(stock\_data\_info, "freeCashflow"),  
 "operatingCashflow": safe\_get(stock\_data\_info, "operatingCashflow"),  
 },  
 "Analyst Targets": {  
 "targetHighPrice": safe\_get(stock\_data\_info, "targetHighPrice"),  
 "targetLowPrice": safe\_get(stock\_data\_info, "targetLowPrice"),  
 "targetMeanPrice": safe\_get(stock\_data\_info, "targetMeanPrice"),  
 "targetMedianPrice": safe\_get(stock\_data\_info, "targetMedianPrice"),  
 },  
 }  
  
 # Return the stock data  
 return stock\_data\_info  
  
  
# Function to fetch the stock history  
def fetch\_stock\_history(stock\_ticker, period, interval):  
 # Pull the data for the first security  
 stock\_data = yf.Ticker(stock\_ticker)  
  
 # Extract full of the stock  
 stock\_data\_history = stock\_data.history(period=period, interval=interval)[  
 ["Open", "High", "Low", "Close"]  
 ]  
  
 # Return the stock data  
 return stock\_data\_history  
  
  
# Function to generate the stock prediction  
def generate\_stock\_prediction(stock\_ticker):  
 # Try to generate the predictions  
 try:  
 # Pull the data for the first security  
 stock\_data = yf.Ticker(stock\_ticker)  
  
 # Extract the data for last 1yr with 1d interval  
 stock\_data\_hist = stock\_data.history(period="2y", interval="1d")  
  
 # Clean the data for to keep only the required columns  
 stock\_data\_close = stock\_data\_hist[["Close"]]  
  
 # Change frequency to day  
 stock\_data\_close = stock\_data\_close.asfreq("D", method="ffill")  
  
 # Fill missing values  
 stock\_data\_close = stock\_data\_close.ffill()  
  
 # Define training and testing area  
 train\_df = stock\_data\_close.iloc[: int(len(stock\_data\_close) \* 0.9) + 1] # 90%  
 test\_df = stock\_data\_close.iloc[int(len(stock\_data\_close) \* 0.9) :] # 10%  
  
 # Define training model  
 model = lstm(train\_df["Close"], 250).fit(cov\_type="HC0")  
  
 # Predict data for test data  
 predictions = model.predict(  
 start=test\_df.index[0], end=test\_df.index[-1], dynamic=True  
 )  
  
 # Predict 90 days into the future  
 forecast = model.predict(  
 start=test\_df.index[0],  
 end=test\_df.index[-1] + dt.timedelta(days=90),  
 dynamic=True,  
 )  
  
 # Return the required data  
 return train\_df, test\_df, forecast, predictions  
  
 # If error occurs  
 except:  
 # Return None  
 return None, None, None, None

**CHAPTER 9**

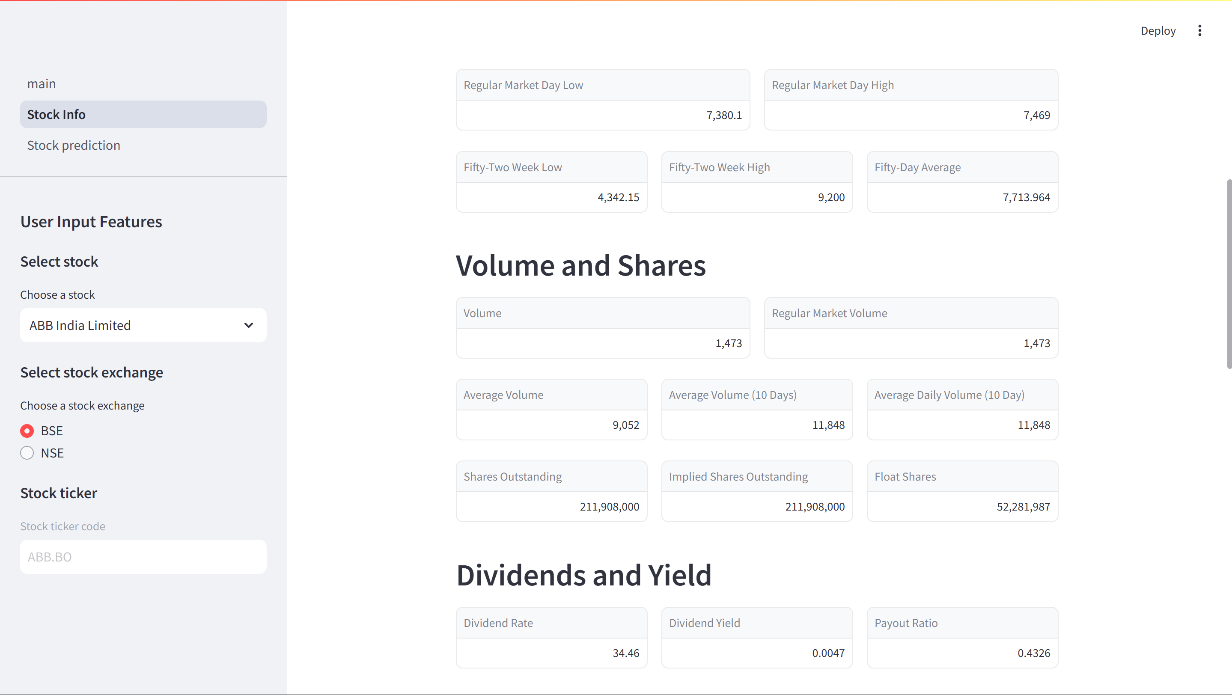
# SCREENSHOTS



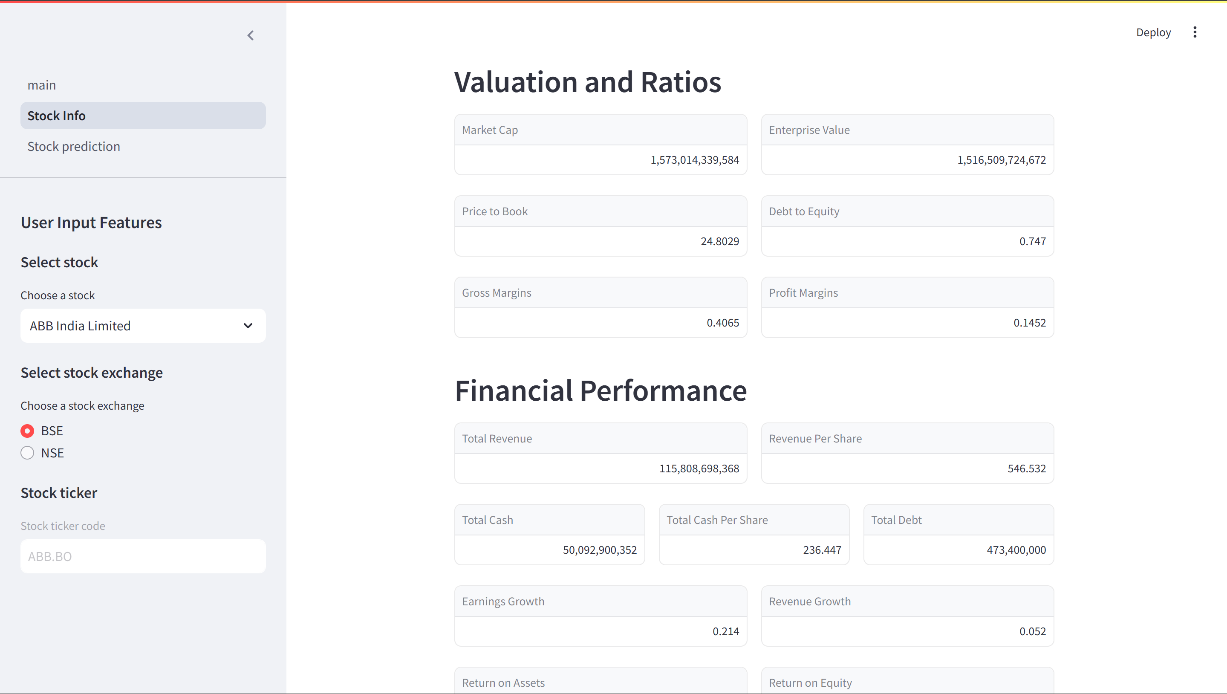
**FIGURE 9.1: HOME PAGE**



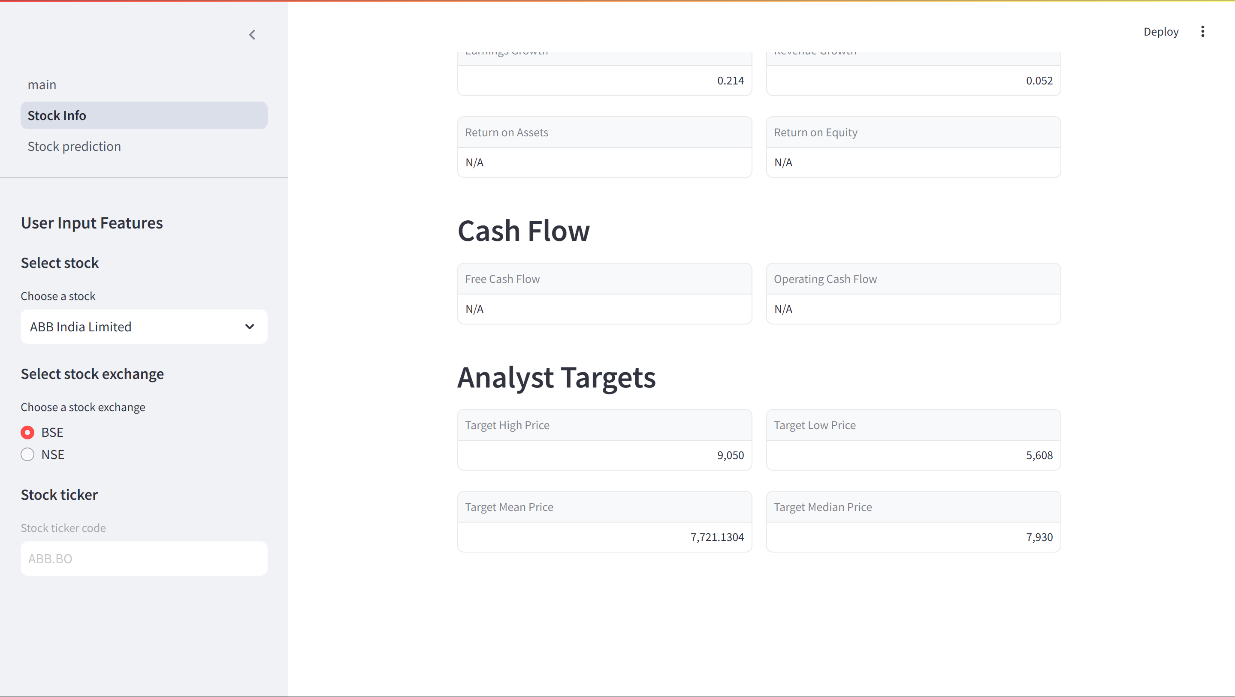
**FIGURE 9.2: STOCK INFO PAGE - 1**



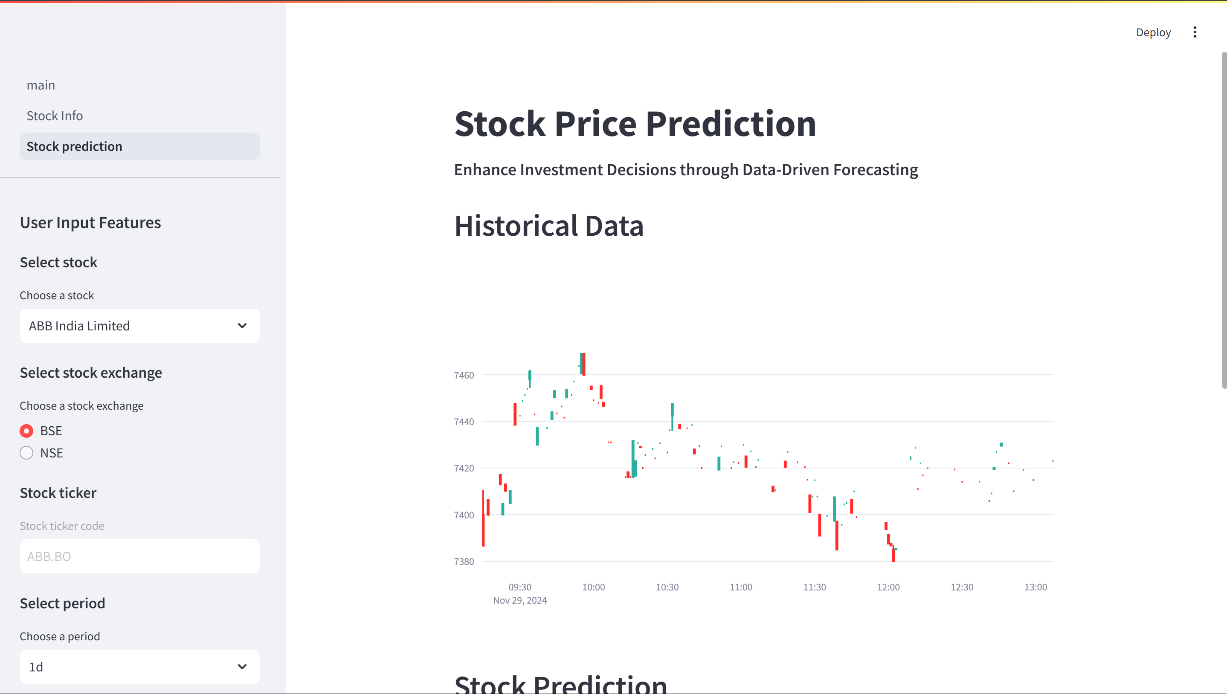
**FIGURE 9.3: STOCK INFO PAGE -2**



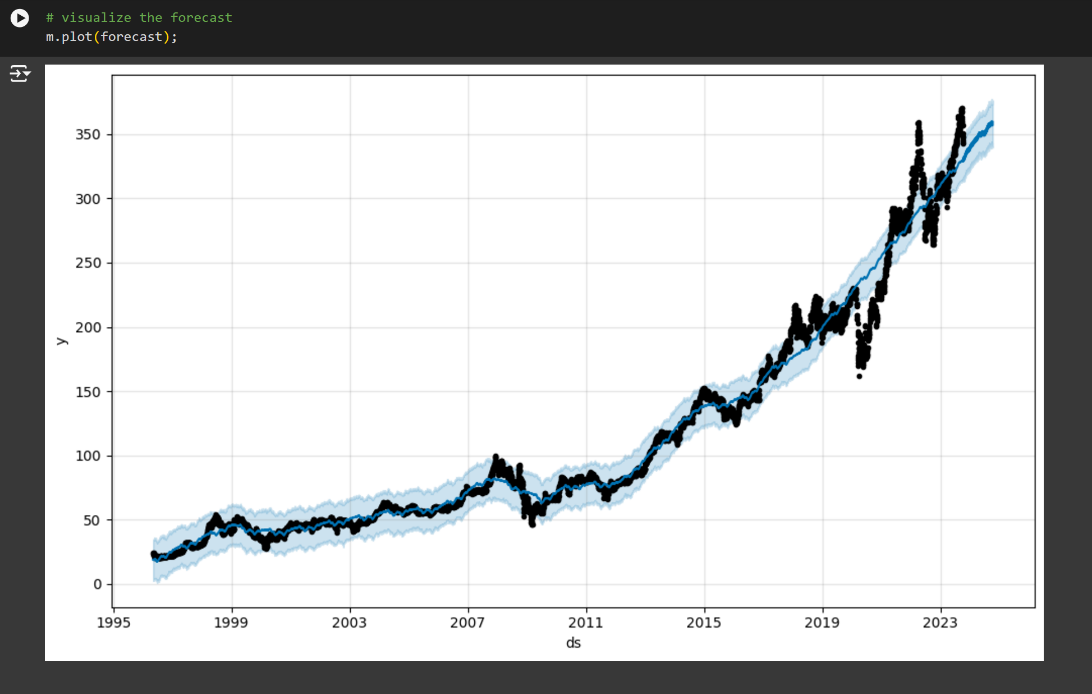
**FIGURE 9.4: STOCK INFO PAGE - 3**



**FIGURE 9.5: STOCK INFO PAGE - 4**



**FIGURE 9.6: STOCK PREDICTION PAGE - 1**



**FIGURE 9.7: STOCK PREDICTION VISUALIZATION**

## CHAPTER 10 REFERENCES

1. A.N. Arya, Y.L. Xu, L. Stankovic, and D.P. Mandic, "Hierarchical graph learning for stock market prediction via a domain-aware graph pooling operator," ICASSP 2023-2023 IEEE International Conference on Acoustics Speech and Signal Processing (ICASSP), pp. 1-5, June 2023.
2. J. Choi, S. Yoo, X. Zhou, and Y. Kim, "Hybrid information mixing module for stock movement prediction," IEEE Access, vol. 11, pp. 28781-28790, 2023.
3. Y. Zhao and G. Yang, "Deep Learning-based Integrated Framework for stock price movement prediction," Applied Soft Computing, vol. 133, pp. 109921, 2023.
4. S. Verma, S.P. Sahu, and T.P. Sahu, "Discrete wavelet transform-based feature engineering for stock market prediction," International Journal of Information Technology, vol. 15, no. 2, pp. 1179-1188, 2023.
5. P.R. Jena and R. Majhi, "Are Twitter sentiments during COVID-19 pandemic a critical determinant to predict stock market movements? A machine learning approach," Scientific African, vol. 19, pp. e01480, 2023.
6. L.N. Mintarya, J.N. Halim, C. Angie, S. Achmad, and A. Kurniawan, "Machine learning approaches in stock market prediction: A systematic literature review," Procedia Computer Science, vol. 216, pp. 96-102, 2023.
7. K. Olorunnimbe and H. Viktor, "Deep learning in the stock market—a systematic survey of practice backtesting and applications," Artificial Intelligence Review, vol. 56, no. 3, pp. 2057-2109, 2023.
8. A. Agarwal, S. Vats, R. Agarwal, A. Ratra, V. Sharma, and L. Gopal, "Sentiment Analysis in Stock Price Prediction," 2023 10th International Conference on Computing for Sustainable Global Development (INDIACom), pp. 1403-1407, March 2023.
9. S. Pourroostaei Ardakani, N. Du, C. Lin, J.C. Yang, Z. Bi, and L. Chen, "A federated learning-enabled predictive analysis to forecast stock market trends," Journal of Ambient Intelligence and Humanized Computing, vol. 14, no. 4, pp. 4529-4535, 2023.
10. M.N. Ashtiani and B. Raahemi, "News-based intelligent prediction of financial markets using text mining and machine learning," Expert Systems with Applications, vol. 217, pp. 119509, 2023.
11. Y. Zhang, Y. Wang, and F. Ma, "Forecasting US stock market volatility: How to use international volatility information," Journal of Forecasting, vol. 40, no. 5, pp. 733-768, 2021.
12. B.K. Meher, I.T. Hawaldar, and C.M. Spulbar, "Forecasting stock market prices using mixed ARIMA model: A case study of Indian pharmaceutical companies," Investment Management and Financial Innovations, vol. 18, no. 1, pp. 42-54, 2021.
13. M. Anusha, K. Suresh, and M. Chandana, "Earlier prediction on the heart disease based on supervised machine learning techniques," 2021 5th International Conference on Intelligent Computing and Control Systems (ICICCS), 2021.
14. A.M. Rather, "LSTM-based deep learning model for stock prediction and predictive optimization model," EURO Journal on Decision Processes, vol. 9, pp. 100001, 2021.
15. M.A.I. Sunny, M.M.S. Maswood, and A.G. Alharbi, "Deep learning-based stock price prediction using LSTM and bi-directional LSTM model," 2020 2nd Novel Intelligent and Leading Emerging Sciences Conference (NILES), pp. 87-92, October 2020.

## CHAPTER 11 BIBILOGRAPHY

* 1. Google Scholar: [https://scholar.google.com](https://scholar.google.com/)
  2. ResearchGate: [https://www.researchgate.net](https://www.researchgate.net/)
  3. arXiv: [https://arxiv.org](https://arxiv.org/)
  4. SpringerLink: [https://link.springer.com](https://link.springer.com/)
  5. ScienceDirect: [https://www.sciencedirect.com](https://www.sciencedirect.com/)