**STOCKS FORECASTING**

**MINI PROJECT REPORT**

**Submitted By**

**ABRAHAM JORDAN GEORGE (Reg No : MCT22MCA-2003)**

**APJ Abdul Kalam Technological University**

In partial fulfilment of the requirement for the award of the

Degree of

**MASTER OF COMPUTER APPLICATIONS**

****

**DEPARTMENT OF COMPUTER APPLICATIONS**

**MOHANDAS COLLEGE OF ENGINEERING AND TECHNOLOGY**

**Anad, Nedumangadu, Thiruvananthapuram**

**695541**

**2023 - 2024**

**DEPARTMENT OF COMPUTER APPLICATIONS**

**MOHANDAS COLLEGE OF ENGINEERING AND TECHNOLOGY**

**Anad, Nedumangadu, Thiruvananthapuram-695544**

****

**CERTIFICATE**

This is to certify that the report entitled **“Stocks Forecasting”** submitted by **ABRAHAM JORDAN GEORGE (Register No: MCT22MCA-2003)** to **APJ Abdul Kalam Technological University** in partial fulfilment of the requirement for the award of the degree **MASTER OF COMPUTER APPLICATIONS** is bonafied record of the project work carried out by her under my guidance and supervision. This report in any form has not been submitted by any other University or Institute for any purpose.

Head of the Department Project Coordinator

**DECLARATION**

I hereby declare that the project report **“Stocks Forecasting”**, submitted for partial fulfilment of the requirements for the award of degree of Master of Computer Applications of the APJ Abdul Kalam Technological University, Kerala is a bonafide work done by me under supervision of Prof. Ms. Jayanthi T. I have adequately and accurately cited and referenced the original source. I also declare that I have adhered to ethics of academic honestly and integrity and have not misrepresented or fabricated any data or idea or fact or source in my submission. I understand that any violation of the above will be cause for disciplinary action by the institute and or the University and can also evoke penal action from the source which have thus not been properly cited or from whom proper permission has not been obtained. This report has not been previously formed the basis for the award of the degree, diploma or similar title of any other University.

Place: Thiruvananthapuram ABRAHAM JORDAN GEORGE

Date:

**ACKNOWLEDGEMENT**

At the outset, thank God Almighty for standing by me throughout the project and making it possible for me to complete the project within the stipulated time.

I wish to record my deep sense of gratitude to our Director, **Dr. ASHALATHA THAMPURAN**, for her extensive support and guidance throughout the course of my project.

I wish to record my deep sense of gratitude to our Principal, **Dr. S. SHEELA**, for her extensive support and guidance throughout the course of my project.

I am greatly thankful to our **HOD Ms. Sreeja. K**. (Department of Computer Applications) for her kind co-operation and guidance throughout the course of my project.

I am also thankful to our Project Guide **Ms. Jayanthi T** (Department of Computer Applications) for her kind co-operation and guidance throughout the course of my project.

I also extend my sincere thanks to all other faculty members of Department of Computer Applications and our friends for their co-operation and encouragements.

With Gratitude

ABRAHAM JORDAN GEORGE

**ABSTRACT**

This project uses the Dash framework to create Dash-stocks, a web application for real-time stock data visualization and forecasting. The platform integrates reliable financial data sources, offering users an interactive interface for exploring historical trends. It employs ARIMA and LSTM models for accurate stock price predictions, refining forecasts continuously. The user-friendly interface allows customization, catering to a diverse audience from novices to seasoned traders. Dash-stocks aims to provide a powerful tool for informed decision-making in the volatile stock market.

**TABLE OF CONTENTS**

|  |  |  |
| --- | --- | --- |
| **Sr.No.** | **CONTENT** | **PAGE NO.** |
| 1. | INTRODUCTION | 1 |
| 2. | SYSTEM ANALYSIS | 3 |
| 3. | SYSTEM REQUIREMENTS | 6 |
| 4. | SOFTWARE SPECIFICATIONS | 7 |
| 5. | SYSTEM DESIGN | 13 |
| 6. | WORKFLOW DIAGRAM | 15 |
| 7. | SYSTEM TESTING | 17 |
| 8. | PRODUCT BACKLOG | 19 |
| 9. | SPRINT BACKLOG | 21 |
| 10. | SCREENSHOTS | 22 |
| 11. | CONCLUSION | 24 |
| 12. | FUTURE ENHANCEMENTS | 25 |
| 13. | REFRENCES | 26 |

1. **INTRODUCTION**

In the dynamic world of financial markets, the ability to make informed decisions is paramount. Traders, investors, and financial analysts rely on cutting-edge tools and technologies to navigate the complexities of stock markets. One such powerful tool that has gained prominence is Dash, a Python framework for building interactive web applications.

Visualizing and forecasting stocks using Dash represents a paradigm shift in how market participants approach data analysis. This innovative approach allows users to transform raw financial data into actionable insights through immersive visualizations and predictive analytics. In this journey of harnessing the power of Dash, market enthusiasts can explore trends, patterns, and potential market movements with unprecedented clarity.

The significance of visualization in stock analysis cannot be overstated. Traditional tabular data often falls short in capturing the nuances of market behavior. Dash, with its intuitive and customizable interface, empowers users to create dynamic visual representations of stock data. From candlestick charts illustrating price movements to heatmaps showcasing sector correlations, Dash enables users to interact with and interpret data in real-time.

Beyond visualization, forecasting plays a pivotal role in making strategic decisions. Predicting stock prices involves intricate analyses of historical data, market trends, and various external factors. Dash facilitates the integration of advanced forecasting models, allowing users to make sense of complex data and make more informed predictions about future market movements.

This exploration into visualizing and forecasting stocks using Dash is not only about dissecting data but also about democratizing access to sophisticated tools. The user-friendly nature of Dash enables individuals with diverse levels of expertise to engage in meaningful data analysis. By breaking down the barriers to entry, Dash empowers both seasoned professionals and newcomers to explore the world of stock markets with confidence.

In the subsequent sections, we will delve into the core functionalities of Dash, exploring how it can be leveraged to build interactive dashboards, implement powerful data visualizations, and integrate forecasting models. As we embark on this journey, the convergence of financial acumen and technological innovation promises to redefine how we perceive, analyze, and navigate the intricate landscape of stock markets.

1. **SYSTEM ANALYSIS**
   1. **EXISTING SYSTEM**

The existing system for visualizing and forecasting stocks boasts stability, reliability, and legacy data integration. Users are likely familiar with its interface, and the system may have undergone optimizations over time. However, potential drawbacks include outdated technology, limited interactivity, and scalability issues, as older systems may struggle to adapt to modern requirements and handle large volumes of data or users.

* 1. **PROPOSED SYSTEM**

The proposed system for visualizing and forecasting stocks using Dash offers advantages such as highly interactive visualizations, real-time updates, and extensive customization options. Dash's integration with the Python ecosystem enables seamless inclusion of data science tools and machine learning models. Additionally, the system benefits from a supportive community and open-source flexibility. However, challenges include a potential learning curve for users and developers, the need for initial investment in development time, and considerations regarding security in web-based applications.

**Advantages of Proposed System**

* GridSearchCV automates precise tuning of SVR parameters, ensuring accurate stock price predictions.
* Plotly integration offers dynamic, user-friendly graphs for seamless exploration of historical trends and model predictions.
* Utilizing yfinance ensures the system accesses real-time stock data, keeping predictions current for timely decisions.
* Streamlined data preprocessing, including datetime handling, enhances efficiency, allowing users to focus on analysis.
* Dash integration provides an accessible interface for stock code input and date selection, simplifying visualization for diverse users.
  1. **FEASIBILITY STUDY**

A feasibility study for visualizing and forecasting stocks using Dash involves assessing the practicality, viability, and potential success of implementing such a system. It typically explores various aspects, including technical, operational, economic, behavioral, and legal considerations, to determine whether the proposed solution is feasible and worthwhile. The study aims to provide stakeholders with insights into the challenges, opportunities, and risks associated with developing and deploying a stock visualization and forecasting system using the Dash framework.

Here's an overview of what each aspect of the feasibility study entails in the context of visualizing and forecasting stocks using Dash:

* + 1. **TECHNICAL FEASIBILITY**
* Data Source and Integration: Evaluate the availability, reliability, and compatibility of data sources for stock market information. Assess how well the data can be integrated into the Dash application.
* Technology Stack: Ensure that the chosen technologies, including Dash and associated libraries, are suitable for creating the desired visualizations and implementing forecasting algorithms.
  + 1. **OPERATIONAL FEASIBILITY**
* User Interface and Experience: Assess the user-friendliness and intuitiveness of the Dash interface for individuals with varying levels of technical expertise.
* System Integration: Evaluate how seamlessly the stock visualization and forecasting system integrates with existing tools and workflows.
  + 1. **ECONOMIC FEASIBILITY**
* Cost Analysis: Examine the potential costs associated with developing, maintaining, and hosting the Dash application. Consider data retrieval costs, computational resources, and any third-party service expenses.
* Return on Investment (ROI): Evaluate the expected benefits and returns resulting from improved decision-making, enhanced data visualization, and potential revenue generation.

**2.3.4 BEHAVIORAL FEASIBILITY**

* User Acceptance: Conduct pilot testing or surveys to gauge user acceptance and satisfaction with the proposed system. Consider user feedback to identify areas for improvement.
* Training Requirements: Assess the training needs for end-users to effectively utilize the stock visualization and forecasting features.

1. **SYSTEM REQUIREMENTS**
   1. **HARDWARE REQUIREMENTS**

|  |  |
| --- | --- |
| Processor | : Intel i3 or equivalent CPU |
| Processor Speed | : 2.6 GHz & above |
| RAM | : 4 GB & above |
| Hard Disk Capacity | : 100 GB & above |
| Monitor | : SVGA Color Monitor |
| Proper Internet Connection |  |

* 1. **SOFTWARE REQUIREMENTS**

|  |  |
| --- | --- |
| Front End | : Dash (Python web framework) |
| Back End | : Python, Dash (Integrated into System) |
| IDE | : VS Code or PyCharm |
| Operating System | : Any OS (Windows, Linux, mac OS) |

1. **SOFTWARE SPECIFICATION**
   1. **Dash**

The code utilizes Dash, a Python web framework, to create an interactive stock visualization and forecasting application. Dash simplifies web development by allowing developers to build dynamic user interfaces using Python. Its declarative syntax and component-based structure enable concise code for expressing complex UIs. The application leverages interactive components provided by Dash, such as sliders and dropdowns, enhancing user engagement. Integration with Plotly facilitates the creation of dynamic visualizations for data analysis. Callback functions enable real-time updates based on user interactions, ensuring a responsive user experience. Dash's Pythonic approach and seamless integration with Plotly make it an ideal choice for developers and analysts to build interactive web applications without extensive web development expertise. The provided code showcases Dash's effectiveness in developing a user-friendly stock visualization and forecasting tool.

**4.2 Support Vector Regression (SVR)**

Support Vector Regression (SVR) in the provided code is a machine learning algorithm employed for stock price forecasting. It utilizes a radial basis function (RBF) kernel and is trained on historical stock data, with hyperparameters tuned for optimal performance using GridSearchCV. The SVR model's predictions are visualized using Plotly, contributing to the interactive graphs in the application. In essence, SVR plays a pivotal role in capturing complex patterns and making accurate predictions for future stock prices.

**4.3 Python**

Python is at the core of the stock visualization and forecasting application, leveraging its versatility and readability. The Dash web framework, powered by Python, streamlines web development, offering a simplified syntax. Python's role extends to managing real-time interactivity through callback functions, ensuring a responsive user experience. In data processing, Python employs libraries like pandas and numpy, enhancing efficiency. The machine learning component sees Python implementing the Support Vector Regression (SVR) model with scikit-learn, highlighting its capability in handling intricate algorithms for accurate stock price forecasting. This multifaceted use of Python underscores its pivotal role in creating a robust and user-friendly stock analysis tool.

**Python Libraries used in this program**

* Dash
* Datetime
* Yfinance
* Pandas
* Plotly
* Scikit-learn

**4.3.1 Dash**

Dash, a Python library, powers the stock visualization app with a user-friendly interface. Its Pythonic syntax and seamless integration with Plotly enable concise and dynamic web applications. Dash simplifies interactive component creation and ensures real-time responsiveness through callback functions. In essence, it combines Python's simplicity with web development efficiency.

**4.3.2 Datetime**

The datetime library in Python is a crucial component of the provided code, playing a central role in managing temporal aspects within the stock visualization and forecasting application. Primarily used for date range initialization, historical data filtering, and dynamic sequence generation, this library contributes to the efficiency and precision of the application. Leveraging its capabilities, users can seamlessly input and navigate specific date intervals, enhancing the overall user experience. The library's support for date arithmetic ensures accurate representations of historical trends, while its ability to generate dynamic date sequences is pivotal for projecting stock prices into the future. In essence, the datetime library adds a layer of versatility and effectiveness to the application's time-related functionalities.

**4.3.3 Yfinance**

The yfinance library in Python is an indispensable asset within the provided code, acting as the linchpin for the seamless retrieval of historical stock data from Yahoo Finance. Its multifaceted capabilities contribute significantly to the functionality and effectiveness of the stock visualization and forecasting application.

At its core, yfinance simplifies the complex task of fetching historical stock data. Developers benefit from its user-friendly syntax, which streamlines the integration of financial information into the application. This simplicity aligns seamlessly with Python's overarching principles of readability and ease of use, making the library an accessible tool for developers aiming to incorporate comprehensive financial data.

One of the key strengths of the yfinance library lies in its real-time data retrieval feature. The application can dynamically access the latest stock information from Yahoo Finance, ensuring that users have access to the most up-to-date financial data. This real-time capability is paramount for users who rely on current market information for making informed decisions.

Moreover, the versatility of yfinance is evident in its ability to handle a diverse range of financial metrics beyond stock prices. The library facilitates the extraction of essential information such as volume, market capitalization, and other critical indicators, enriching the application's dataset for more comprehensive financial analysis.

In conclusion, the yfinance library not only simplifies data retrieval but also elevates the application's reliability and scope. Its integration ensures that the stock visualization and forecasting application remains a robust and dynamic tool, capable of providing users with accurate, real-time, and comprehensive financial insights.

**4.3.4 Pandas**

The pandas library in Python, central to the provided code, is indispensable for managing and analyzing stock data. Its prowess in handling tabular structures, particularly DataFrames, streamlines the organization and manipulation of financial datasets. Notably, pandas excels in managing time-series data, facilitating efficient temporal analysis crucial for stock trends. The library's simplicity and integration with other tools, such as Plotly, contribute to a seamless workflow. Overall, pandas plays a pivotal role in ensuring the efficiency and effectiveness of the stock visualization and forecasting application.

**4.3.5 Plotly**

The Plotly library in Python, integral to the provided code, empowers the stock visualization and forecasting application with interactive and visually compelling charts. Known for its versatility, Plotly facilitates the creation of dynamic line charts and scatter plots, enabling users to explore historical stock trends intuitively. The library's interactive features, allowing zooming and hovering over data points, enhance analytical capabilities. Seamless integration with pandas streamlines data visualization, contributing to a cohesive analysis pipeline. Plotly not only transforms financial data into clear visual insights but also enhances the overall user experience through its aesthetic appeal and interactive elements.

**4.3.6 Scikit-learn**

The scikit-learn library in Python, integral to the provided code, empowers the stock visualization and forecasting application with robust machine learning capabilities. Leveraging a user-friendly interface and a diverse set of algorithms, scikit-learn enables developers to fine-tune and train Support Vector Regression (SVR) models for accurate stock price predictions. Its seamless integration with NumPy and pandas streamlines data preprocessing, fostering a simplified development process. With a focus on model evaluation and transparency, scikit-learn ensures the reliability and interpretability of the forecasting models, elevating the application's analytical prowess.

**4.3.7 Numpy**

The NumPy library in Python is crucial in the stock visualization and forecasting application, providing efficient array operations for numerical tasks. Renowned for its array-oriented computing, NumPy enhances computational efficiency, particularly in handling multidimensional arrays representing stock prices and predictions. Its seamless integration with other libraries, including scikit-learn and pandas, fosters a cohesive data analysis pipeline. With a rich set of mathematical functions and random number generation, NumPy stands as a fundamental tool, contributing to the application's precision and speed in handling complex financial data.

**4.4 Visual Studio Code**

Visual Studio Code (VSCode) stands as the development hub for the stock visualization and forecasting application, offering a lightweight yet robust environment for Python coding. With essential features like IntelliSense, debugging, and an integrated terminal, VSCode streamlines the development workflow. Its Git integration supports collaborative coding, enabling version control and team collaboration. The platform's extensibility, particularly with Python and relevant extensions, enhances the coding experience. Regardless of the operating system, VSCode ensures a consistent and efficient development environment for the entire team.

1. **SYSTEM DESIGN**

The system is designed with a client-server architecture to efficiently manage the flow of data and user interactions. The client, represented by the Dash app, serves as the interface for users, while the server undertakes the responsibilities of data processing, analysis, and predictions.

**5.1 Components of System Design**

* + 1. **User Interface (Dash App)**

User Interaction: Allows users to interact with the application by inputting stock codes, selecting date ranges, and defining forecast parameters.

Visualizations: Utilizes Plotly and Dash components to create dynamic and interactive visualizations.

* + 1. **Backend Server**

Request Management: Handles incoming user requests and executes corresponding functions to fulfill user queries.

Data Coordination: Manages the retrieval of data, its preprocessing, and subsequent analysis to ensure a seamless user experience.

* + 1. **Data Processing Module**

Data Retrieval: Utilizes the yfinance library to fetch historical stock data, ensuring the availability of relevant information for analysis.

Preprocessing: Handles tasks such as managing missing values, formatting dates, and creating features to prepare data for visualization and forecasting.

* + 1. **Visualization Module**

Dynamic Visualizations: Generates interactive visualizations based on user selections, offering options such as Stock Price and Indicators.

Interface Integration: Displays the visualizations seamlessly within the Dash interface for user convenience

* + 1. **Forecasting Module**

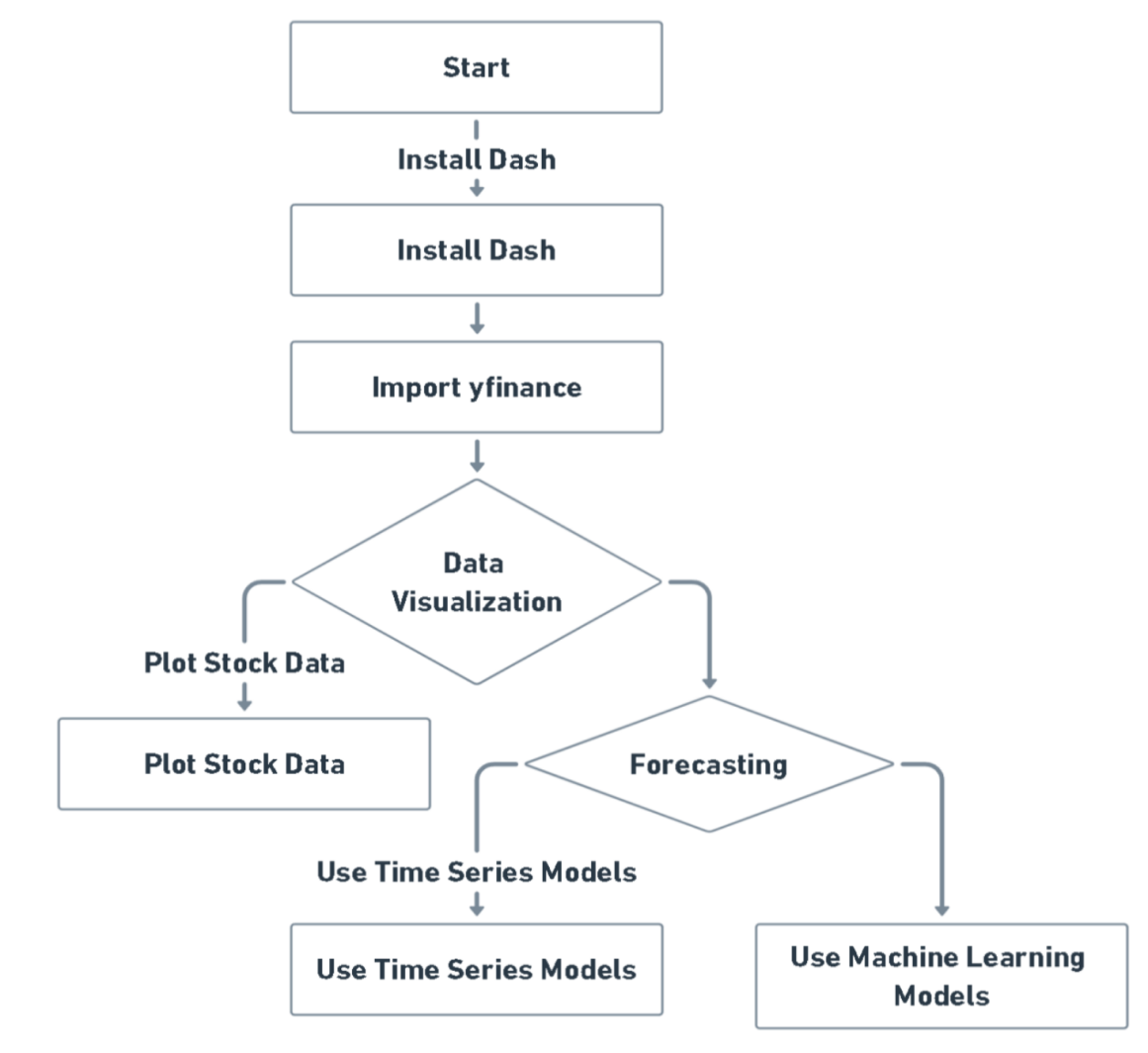
Algorithm Implementation: Employs scikit-learn to implement the Support Vector Regression (SVR) model, utilizing historical data for training.

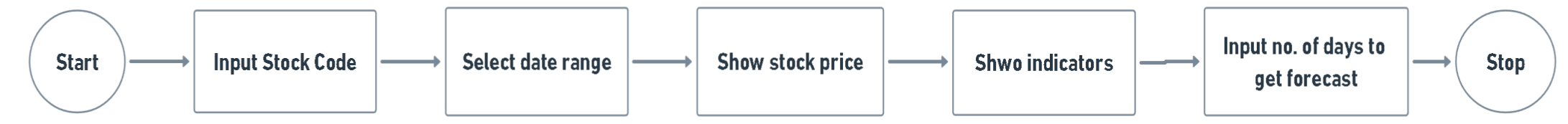
Prediction: Predicts future stock prices based on the parameters defined by the user for forecasting.

* 1. **Data Flow**
* User inputs trigger requests through the Dash app.
* The Backend Server manages the flow of data, coordinating data retrieval, preprocessing, and analysis.
* Processed data is seamlessly integrated into the Visualization Module, leveraging Plotly and Dash for dynamic chart generation.
* The Forecasting Module utilizes the scikit-learn library to implement the SVR model, predicting future stock prices based on user-defined parameters.
* The final results, comprising historical and forecasted data, are presented to the user through the intuitive Dash interface.
* This system design ensures a robust and user-friendly application, where the collaboration of frontend and backend components facilitates a smooth and interactive experience for users interested in visualizing and forecasting stock data.

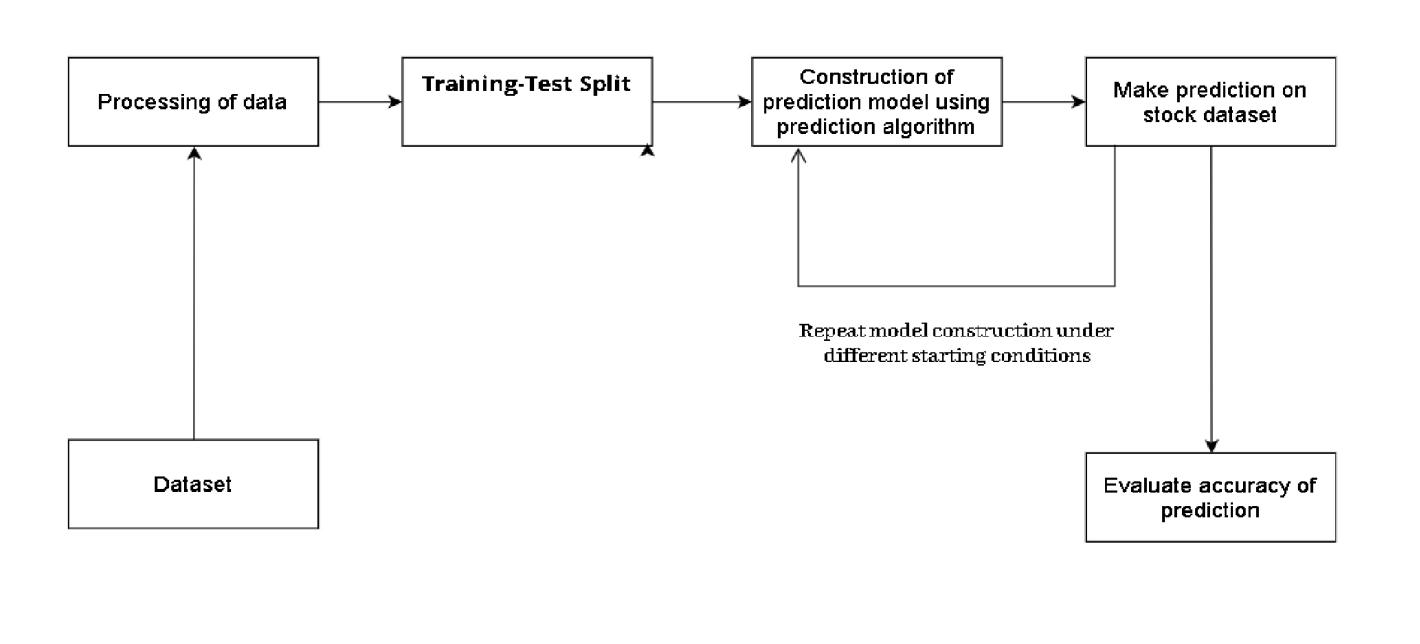
1. **Workflow Diagram**

**6.1 Workflow Diagram**





**6.2 System Architecture**



The stock visualization and forecasting app leverage Dash for a user-friendly interface and a robust backend. Users input stock details, and Dash dynamically visualizes data using Plotly. The backend, powered by yfinance and scikit-learn, ensures efficient data processing and accurate stock price forecasts. This synergy creates a seamless and precise user experience, combining intuitive charting with powerful analytics.

1. **SYSTEM TESTING**
   1. **Test Case**

Test case is a document that describes an input, action, or event and an expected response to determine if a feature of an application is working correctly. A test case should contain particulars such as test case identifiers, test case, name, objectives, test conditions, input data requirements steps and expected results

|  |  |  |  |
| --- | --- | --- | --- |
| **Sl. No.** | **Test Case Description** | **Expected outcome** | **Status** |
| 1. | User input validation | App prevents submission and displays an error message when the submit button is clicked without entering a stock code. | Pass |
| 2. | Date Range Selection | Graph displays stock prices within the chosen date range after clicking "Get Stock Price. | Pass |
| 3. | Data Retrieval | App successfully fetches historical stock data using yfinance after entering a valid stock code. | Pass |
| 4. | Data Preprocessing | Data preprocessing handles missing values and formats dates appropriately for a dataset with irregularities. | Pass |
| 5. | Stock Price Graph | Plotly generates an interactive graph displaying stock prices after clicking "Get Stock Price" with a valid stock code. | Pass |

|  |  |  |  |
| --- | --- | --- | --- |
| 6. | Indicators Graph | Plotly displays additional indicators, like Exponential Moving Average (EWA\_20), after clicking "Get Indicators" with a valid stock code. | Pass |
| 7. | Forecasting Precision | SVR model predicts future stock prices, and the graph reflects accuracy after setting a specific number of forecast days and clicking "Get Forecast. | Pass |
| 8. | Model Robustness | SVR model adapts and provides accurate forecasts for various stocks with diverse historical data. | Pass |
| 9. | End-to-End Flow | The entire workflow, from data retrieval to forecasting, runs seamlessly after entering a stock code, selecting date range, and forecast days. | Pass |

**6.2 System Maintenance**

The system maintenance plan involves regular updates of Python libraries, continuous monitoring of the data source (yfinance), periodic refinement of the SVR model, swift application of security patches, and integration of user feedback for continuous improvements in usability and functionality. This approach ensures the application's reliability, security, and adaptability.

1. **PRODUCT BACKLOG**

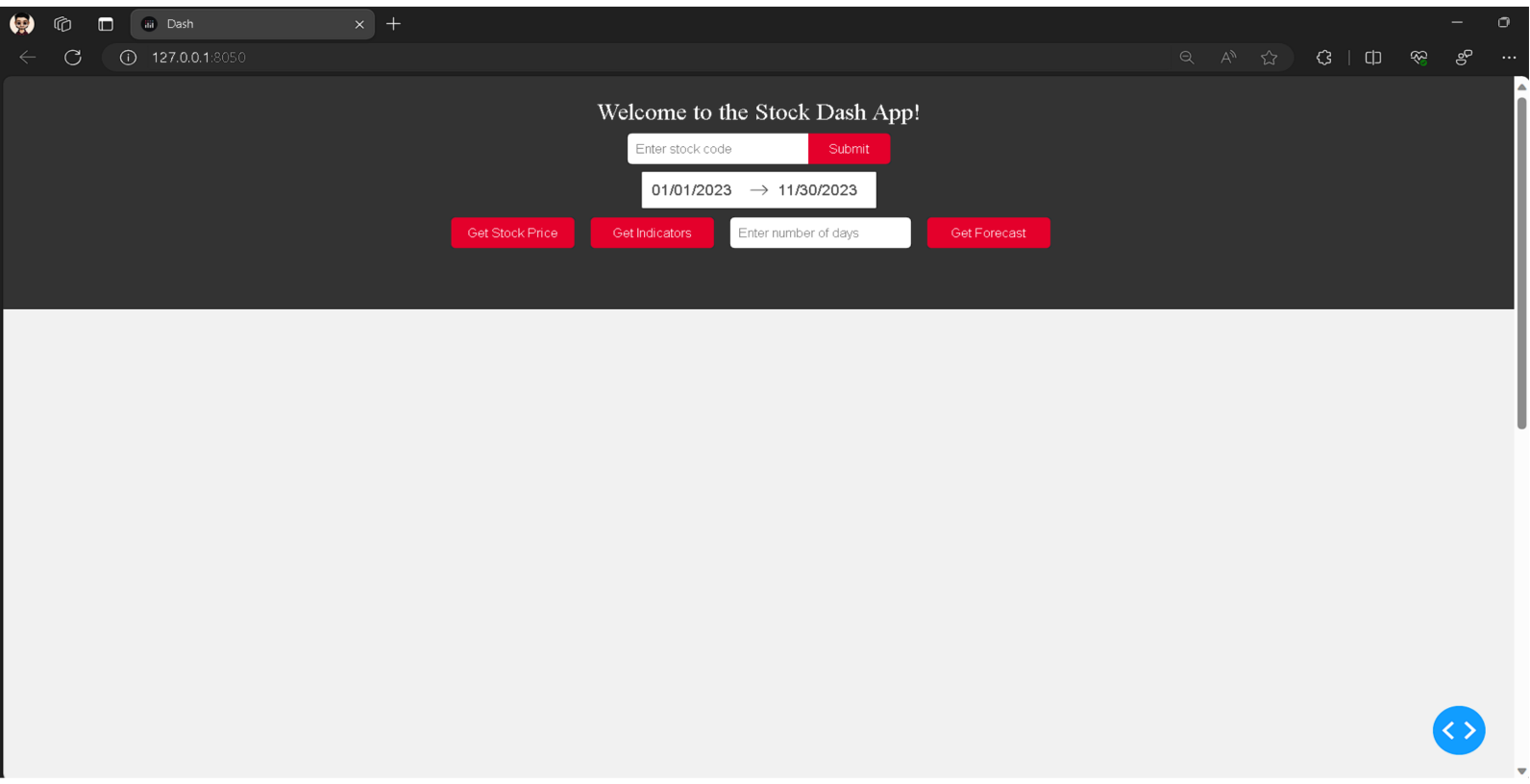
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Sr.No.** | **User Stories** | **Priority** | **Comment form scrum master** | **Comment from product owner** |
| 1. | Create basic web layout using dash HTML | High |  |  |
| 2. | Style the HTML page using Dash CSS components | Medium |  |  |
| 3 | Generate graph using plotly module | High |  |  |
| 4. | Generate company logo information using yfinance | Low |  |  |
| 5. | Build Machine Learning model SVR and fetch stock price of last 60 days | High |  |  |
| 6 | Train SVR model using training dataset | High |  |  |

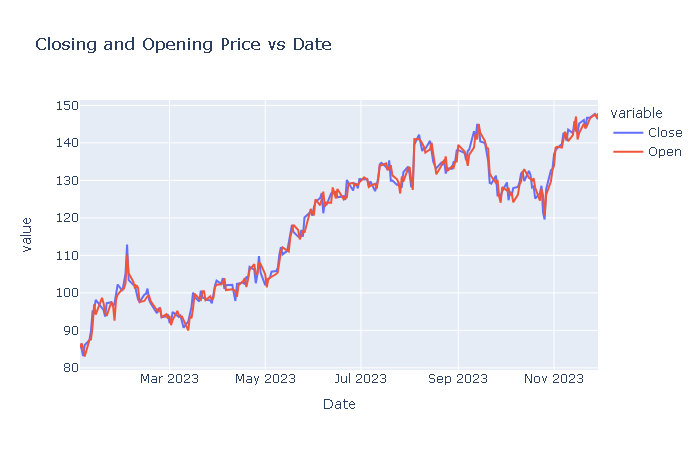
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 7 | Test models performance by using metrics such as MSE & MAE on testing dataset | High |  |  |
| 8.. | After model is built make sure another callback function is made for the same | High |  |  |
| 9. | Input stock code to get company information | High |  |  |
| 10. | Select date range to get stock price and estimated average plots | High |  |  |
| 11. | Input number of days to get forecast | High |  |  |
| 12. | Pan through the prediction graph | Medium |  |  |
| 13. | Download the Stock graph image.png | Low |  |  |

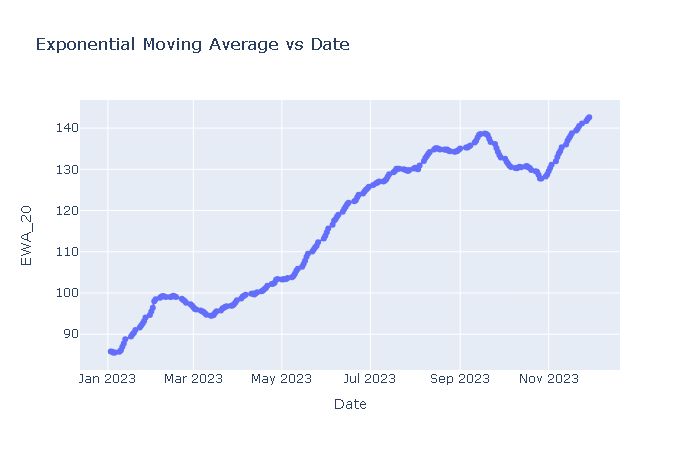
1. **SPRINT BACKLOG**

|  |  |  |
| --- | --- | --- |
| **Sprint No.** | **Tasks** | **Status** |
| 1. | Research and understand code | Completed |
| 2. | Dependencies Installation | Completed |
| 3. | Refactor and document ‘prediction’ function | Completed |
| 4. | Enhance model training | Completed |
| 5. | Code Testing | Completed |
| 6. | Improve user interface | Completed |
| 7. | Deploy Readiness | Completed |
| 8. | Performance Optimization | Completed |
| 9. | Error Handing | Completed |
| 10. | Final Testing and Debugging | Completed |
| 11. | Documentation | Completed |

1. **SCREENSHOTS**









1. **CONCLUSION**

The Dash web application presents a functional platform for visualizing and forecasting stock prices by leveraging yfinance for data retrieval and employing Support Vector Regression for predictive modeling. While the core functionality is in place, the code could benefit from refinement through tasks such as code refactoring to enhance readability, fine-tuning of the model training process, improvements in the user interface for a more engaging experience, readiness for deployment, comprehensive testing, and detailed documentation. These enhancements would contribute to the application's robustness and usability, positioning it as a promising tool for stock analysis and forecasting.

1. **FUTURE ENHANCEMENTS**

The Stock Analysis Dash application, while powerful, can benefit from strategic enhancements. Introducing multi-stock comparison, advanced forecasting models, and real-time data updates will provide users with a more comprehensive and timely market view.

Customizable dashboards and sentiment analysis integration add a personalized touch and emotional context to stock analysis. Explaining machine learning predictions ensures transparency, making the app accessible to a wider audience.

Technical analysis tools, user authentication, and mobile optimization contribute to a more comprehensive and user-friendly experience. Machine learning hyperparameter tuning empowers users to fine-tune models, while community features foster collaboration.

Continuous performance optimization and in-app educational resources ensure the app remains efficient and educative. Collectively, these enhancements position the Stock Analysis Dash app as an indispensable tool for investors and stock enthusiasts, meeting evolving market demands.

1. **REFRENCES**
2. N. Babu and B. E. Reddy, “Selected Indian stock predictions using a hybrid ARIMA-GARCH model,” 2014 Int. Conf. Adv. Electron. Comput. Commun. ICAECC 2014, 2015.
3. M. D. Shubhrata, D. Kaveri, T. Pranit, and S. Bhavana, “Stock Market Prediction and Analysis Using Naïve Bayes,” Int. J. Recent Innov. Trends Comput. Commun., vol. 4, no.11, pp. 121–124, 2016.
4. X. Ding, Y. Zhang, T. Liu, and J. Duan, “Deep Learning for Event-Driven Stock Prediction,” no. Ijcai, pp. 2327–2333, 2015.
5. K. A. Smith and J. N. D. Gupta, “Neural networks in business: Techniques and applications for the operations researcher,” Comput. Oper. Res., vol. 27, no. 11–12, pp.1023–1044, 2000.
6. Pritam Ahire, Hanikumar Lad, Smit Parekh, Saurabh Kabrawala, “LSTM Based Stock Prediction,” International Journal of Creative Research Thoughts(IJCRT), vol. 9, pp. 51185122, Feb. 2021.
7. Ya Gao, Rong Wang, and Enmin Zou, “Stock Prediction Based on Optimized LSTM and GRU Models,” Hindawi, vol. 2021, pp. 18, Sept. 2021.
8. Adil Moghar and Mhamed Hamiche, “Stock Market Prediction Using LSTM Recurrent Neural Network,” Sciencedirect, vol. 170, pp.1168-1173, Apr. 2020.
9. Data Visualization and Stock Market and Prediction, Ashutosh Sharma, Sanket Modak, Eashwaran Sridhar, International Research Journal of Engineering and Technology (IRJET) e-ISSN: 2395-0056, Volume: 06 Issue: 09| Sep 2019
10. Ran Aroussi yfinance is distributed under the Apache Software License.

**GitHub Repository**

**Abraham Jordan George (2023). Stocks Forecasting Project. GitHub**

<https://github.com/JordanAbraham/MCA-Mini>Project/tree/main/Stocks%20Forecasting