

The Final Synopsis of the project entitled

Stock Price Prediction App

(Machine Learning)

Submitted to the faculty of Engineering and Technology for the partial fulfilment of the requirements of Master Of Computer Applications (FYIC) 6th Sem.

Supervised by:
Ms.Inderdeep Kaur
Department Of Computer Science

Submitted by:
Prabhjeet Singh
27502100066



DEPARTMENT OF COMPUTER SCIENCE

GURU NANAK DEV UNIVERSITY

AMRITSAR-143005

India

May, 2024

Declaration

I hereby declare that the work embodied in this project entitled “Stock Price Prediction App” submitted to the Department of Computer Science, Guru Nanak Dev University, Amritsar, for the partial fulfilment of the degree of Master of Computer Applications is entirely based on my own work and not submitted elsewhere for the award of any other degree. All ideas and references have been duly acknowledged.

Date

Signature

Prabhjeet Singh

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

Ms.Inderdeep Kaur

Department of Computer Science

Certificate

It is Certified that Ms/Mr Prabhjeet Singh Roll Number 27502100066 of MCA(FYIC) has worked on the project entitled, “Stockline”, for the partial fulfilment of the degree of Master of Computer Applications. The student will be solely responsible for the authenticity and originality of the work submitted.

Date

--/--/2024

Supervisor:

Ms.Inderdeep Kaur

Acknowledgement

First and foremost, I would like to thank to my supervisor of this project , Ms. Inderdeep Kaur for the valuable guidance and advice. She inspired us greatly to work in this project. His willingness to motivate us contributed tremendously to our project. I also would like to thank him for devoting his valuable time and imparting knowledge to me. With his valuable guidance, constructive suggestions helped me in the preparation of this project and the course up to the completion of software became smooth without much cared hurdles. He was always friendly and encouraging during the development of project. This project would not have been possible without the support of many people. I am also thankful to all the staff members of the department of computer science for their help at every stage. They helped in every respect during the tenure. Lastly, I am also thankful to my parents and friends for their moral support in every sphere. Their vital push infused sense of insurgency in me, I am also thankful to them for their assistance and cooperation.

Prabhjeet Singh

Abstract

Purpose:

The purpose of this project is to develop a linear regression model for predicting stock prices based on historical data. The model can be used to provide insights and predictions into the stock market, which can help investors and traders make informed decisions.

The project is implemented in Python, using libraries such as Pandas, NumPy, Scikit-learn, and Plotly. The project is hosted on a Streamlit web application, which allows users to interact with the model and visualize the results.

Methodology:

The methodology of the project follows the Agile development model, which is an iterative and incremental approach to software development that emphasizes flexibility, collaboration, and customer satisfaction. The methodology includes the phases of requirement gathering, planning, design, development, testing, deployment, and maintenance. The methodology is iterative and incremental, as the project team delivers working software at the end of each sprint, which can be used by the customer for feedback and validation. The methodology also encourages collaboration and communication between the development team and the customer, as the customer is involved in the development process and provides valuable feedback..

Results:

The results of the project show that the linear regression model can predict stock prices with a reasonable accuracy. The r-squared score of the model is 0.71, which indicates that the model explains 71% of the variance in the stock prices. The mean squared error of the model is 0.0004, which indicates that the average difference between the actual and predicted stock prices is 0.04%.

The results also show that the linear regression model can provide valuable insights into the stock market, such as the trend, the volatility, and the correlation between the stock price and the independent variables. The linear regression model can be used to predict the future stock prices, which can help investors and traders make informed decisions.

In summary, the results of the project show that the linear regression model can predict stock prices with a reasonable accuracy and provide valuable insights into the stock market. The linear regression model has some limitations, such as the assumption of linearity, the assumption of normality and independence of errors, and the sensitivity to outliers or missing data. To address these limitations, other machine learning models, such as decision trees, random forests, or neural networks, can be used to improve the accuracy and reliability of the model.

Conclusion:

In conclusion, the project has achieved its objectives of developing a linear regression model for predicting stock prices, preprocessing the data, training the model, evaluating the model, and visualizing the results. The project has followed the Agile development model, which is an

iterative and incremental approach to software development that emphasizes flexibility, collaboration, and customer satisfaction. The project has also followed the methodology of requirement gathering, planning, design, development, testing, deployment, and maintenance. The project has used libraries such as Pandas, NumPy, Scikit-learn, and Plotly, and has been hosted on a Streamlit web application. The project has provided valuable insights and predictions into the stock market, and has demonstrated the feasibility of using linear regression for predicting stock prices. The project has some limitations, such as the assumption of linearity, the assumption of normality and independence of errors, and the sensitivity to outliers or missing data. To address these limitations, other machine learning models, such as decision trees, random forests, or neural networks, can be used to improve the accuracy and reliability of the model. With further improvements and implementation, the model could become a useful tool for investors and traders.

Table of Contents

Introduction to the Project	9
Statement of the problem.....	10-11
Revisiting the Problem	12-13
• Problem Definition	12-13
Background / Literature Review	14-15
Objectives and Features.....	15-18
• Primary Objectives of Stock Exchange.....	17
• To Qualify for Long-term Financing	17
• To Raise Capital.....	17
• To Protect Fraudulently.....	17
• Increasing the Efficiency of Trades.....	17
Features.....	18
• A market for securities.....	18
• Second-hand securities.....	18
• Regulate trade in securities.....	18
• Dealings only in registered securities.....	18
• Transaction.....	18
• Recognition.....	18
• Measuring device.....	18
• Operates as per rules.....	18
Folders and files used in the project.....	19
Tools Used.....	20-23
• VS Code.....	20-21
• Google Chrome.....	21-22
• Windows 11.....	22-23
Tech Tools.....	24-25
• Python.....	24
• Streamlit.....	24-25

• Jupyter Notebook.....	26
Software Development Life Cycle.....	25-50
• Requirement Gathering and Analysis.....	26
• Hardware Requirements Server.....	26
• Software Requirements.....	26
• Feasibility Study.....	26-27
• Design/DFDS.....	28
• Coding.....	29-44
• Project Screenshots.....	45-49
• Implementation.....	50
Detailed Analysis.....	51-52
Limitations Of The Project.....	53
Conclusion And Future Work	54
References.....	55

Introduction to the Project



Well, project “Stock Price Prediction App” will help the traders to analyse the company while investing in it. For that in this project a big data is used which include large amount of data that will help the traders and investors to invest in the company .This Project is not Only Used in India But also include the data of Crypto , Forex Exchange where the ease to analysing the stock of other countries is much more easier and for that machine learning algorithms are used. All will be done using a web page. So, this project contains different technologies as ML, Python and Streamlit server which connects web pages together. Stock Price Prediction app will help the trader to understand the trend of the stock and also the hidden and important details about the company so that a trader can analyse the company properly before investing. For that in this project a live dataset is used which include large amount of data that will help to fetch the required data and also to predict the chart properly and for that machine learning algorithms are used.

Statement of the problem

Stock market trading has gained enormous popularity globally and for many people, it is a part of the everyday routine to make gains. But forecasting the movement of stock prices is a challenge due to the complexity of stock market data. Forecasting can be defined as the prediction of some future events by analysing the historical data. It spans many areas including industry, business, economics, and finance. However, as the technology is advancing, there is an improvement in the opportunity to gain a steady fortune from the stock market and it also helps experts find the most useful indicators to make much better forecasting. Many of the forecasting problems involve the analysis of time. Time-series data analysis helps to recognize patterns, trends and phases or cycles that are present in the data.

In the case of the stock market, early knowledge of bullish or bearish mode serves to wisely invest capital. The study of trends also helps to recognize the bestperforming companies over a given period. This makes analysis and forecasting of the time series a significant research field. Deep learning is a framework for training and modeling neural networks that in many learning tasks, especially image and voice recognition, have recently exceeded all traditional methods. The paper that we have presented modeled and predicted the stock prices of NIFTY 50 index which is a diversified index of 50 stocks covering 12 Indian economy sectors listed in the Indian National Stock Exchanges (NSE). We selected the NSE because it holds a place of prominence globally and it stands among the highest in innovation and technical development. Furthermore, we focus on Linear Regression Model to predict future trends of stock prices as well as the financial time series based on historical data.

As a primary market, the stock market allows companies to issue and sell their shares to the public for the first time through the process of an initial public offering (IPO). This activity helps companies raise necessary capital from investors.

A company divides itself into several shares and sells some of those shares to the public at a price per share. To facilitate this process, a company needs a marketplace where these shares can be sold and this is achieved by the stock market. A listed company may also offer new, additional shares through other offerings at a later stage, such as through rights issues or follow-on offerings. They may even buy back or delist their shares.

Investors will own company shares in the expectation that share value will rise or that they will receive dividend payments or both. The stock exchange acts as a facilitator for this capital-raising process and receives a fee for its services from the company and its financial partners. Using the stock exchanges, investors can also buy and sell securities they already own in what is called the secondary market.

The stock market or exchange maintains various market-level and sector-specific indicators, like the S&P (Standard & Poor's) 500 index and the Nasdaq 100 index, which provide a measure to track the movement of the overall market.

The stock market guarantees all interested market participants have access to data for all buy and sell orders, thereby helping in the fair and transparent pricing of securities. The market also ensures efficient matching of appropriate buy and sell orders.⁷

Stock markets need to support price discovery where the price of any stock is determined collectively by all of its buyers and sellers. Those qualified and willing to trade should get instant access to place orders and the market ensures that the orders are executed at a fair price.

Traders on the stock market include market makers, investors, traders, speculators, and hedgers. An investor may buy stocks and hold them for the long term, while a trader may enter and exit a position within seconds. A market maker provides necessary liquidity in the market, while a hedger may trade in derivatives.

A stock market is a regulated and controlled environment. In the United States, the main regulators include the Securities and Exchange Commission (SEC) and the Financial Industry Regulatory Authority (FINRA).²

U.S. Securities and Exchange Commission. "About Trading and Markets."

The earliest stock markets issued and dealt in paper-based physical share certificates. Today, stock markets operate electronically.

Revisiting the Problem

Well, project Stock Price Prediction app will help the trader to understand the trend of the stock and also the hidden and important details about the company so that a trader can analyse the company properly before investing. For that in this project a live dataset is used which include large amount of data that will help to fetch the required data and also to predict the chart properly and for that machine learning algorithms are used. All will be done using a web page.

Problem Definition

The problem definition of stock price analysis involves examining historical and current stock prices to understand patterns, trends, and factors affecting the price movements of a particular stock or the stock market as a whole. The goal is to make informed decisions about buying, selling, or holding stocks based on the insights gained from the analysis. Key components of stock price analysis include:

- ❖ **Data Collection:** Gathering historical stock price data from various sources such as financial websites, APIs, or databases.
- ❖ **Data Cleaning and Pre-processing:** Removing any errors, missing values, or outliers from the data and preparing it for analysis.
- ❖ **Exploratory Data Analysis (EDA):** Conducting an initial exploration of the data to identify patterns, correlations, and anomalies. This may involve visualizations such as line charts, histograms, or scatter plots.
- ❖ **Statistical Analysis:** Applying statistical techniques to further analyse the data, including measures of central tendency, dispersion, and correlation coefficients.
- ❖ **Time Series Analysis:** Examining the stock price data over time to identify trends, seasonality, and cyclicity using methods such as moving averages, exponential smoothing, or autoregressive integrated moving average (ARIMA) models.
- ❖ **Technical Analysis:** Using technical indicators and chart patterns to forecast future price movements, such as moving averages, relative strength index (RSI), or Bollinger Bands.

- ❖ **Fundamental Analysis:** Assessing the intrinsic value of a stock based on factors such as earnings, dividends, financial ratios, and industry trends.
- ❖ **Sentiment Analysis:** Analysing news articles, social media posts, and other sources of information to gauge market sentiment and its potential impact on stock prices.
- ❖ **Predictive Modeling :** Building predictive models to forecast future stock prices based on historical data and relevant features. Machine learning algorithms such as regression, decision trees, or neural networks can be employed for this purpose.
- ❖ **Risk Assessment:** Evaluating the risks associated with investing in a particular stock or portfolio, including market risk, volatility, and diversification.

By thoroughly analysing stock prices using these techniques, investors can gain insights into market dynamics and make more informed decisions to optimize their investment strategies.

Background / Literature Review

In this section, we will review the literature related to the linear regression model for predicting stock prices.

Linear Regression:

Linear regression is a statistical method for modeling the relationship between a dependent variable and one or more independent variables. Linear regression is widely used in finance and economics for predicting stock prices, exchange rates, and other financial variables. Linear regression assumes that the relationship between the dependent and independent variables is linear, and that the errors or residuals are normally distributed and independent.

Machine Learning:

Machine learning is a subfield of artificial intelligence that focuses on the development of algorithms and models for learning from data. Machine learning can be divided into three categories: supervised learning, unsupervised learning, and reinforcement learning. Supervised learning involves training a model on a labeled dataset, where the input and output variables are known. Unsupervised learning involves training a model on an unlabeled dataset, where the input variables are known but the output variables are unknown. Reinforcement learning involves training a model to make decisions or take actions in an environment, based on feedback or rewards.

Stock Price Prediction:

Stock price prediction is the use of statistical or machine learning models to forecast the future prices of stocks or other financial instruments. Stock price prediction is a challenging problem, as the stock market is influenced by many factors, such as economic indicators, company fundamentals, and market sentiment. Stock price prediction can be used for trading, investment, and risk management.

Linear Regression for Stock Price Prediction:

Linear regression is a popular and widely used model for predicting stock prices. Linear regression assumes that the relationship between the stock price and the independent variables is linear, and that the errors or residuals are normally distributed and independent. Linear regression can be used for both univariate and multivariate analysis, where the former involves only one independent variable, and the latter involves multiple independent variables.

Limitations of Linear Regression for Stock Price Prediction:

Linear regression has some limitations for predicting stock prices. Linear regression assumes that the relationship between the stock price and the independent variables is linear, which may not be the case in reality. Linear regression also assumes that the errors or residuals are normally distributed and independent, which may not be the case in reality. Linear regression may also be sensitive to outliers or missing data, which can affect the accuracy and reliability of the model.

In conclusion, the literature review shows that linear regression is a popular and widely used model for predicting stock prices. Linear regression assumes that the relationship between the stock price and the independent variables is linear, and that the errors or residuals are normally distributed and independent. Linear regression can be used for both univariate and multivariate analysis, where the former involves only one independent variable, and the latter involves multiple independent variables. However, linear regression has some limitations for predicting stock prices, such as the assumption of linearity, the assumption of normality and independence of errors, and the sensitivity to outliers or missing data. To address these limitations, other machine learning models, such as decision trees, random forests, or neural networks, can be used to improve the accuracy and reliability of the model. Therefore, it is important to carefully evaluate and validate the linear regression model for predicting stock prices, and to consider limitations can other models or approaches if affect necessary. To address these limitations, other machine learning models, such the accuracy and reliability of the model, and should be taken into account when using linear regression for stock price prediction. as decision trees, random forests, or neural networks, can be used for stock price prediction.

Objectives and Features



Many researchers and investors are interested in the finance market. Thus, they need guidance and reliable predictions to invest wisely. Recently, stock market forecasting has gained more interest because investors may be better informed if the market path is accurately predicted. Many businesses already use the concept of deep learning and machine learning to make profits. Investment and trading profitability in the stock market is largely dependent on predictability. If any system that can reliably predict the volatile stock market movements was created, the system's owner would become wealthy. More about the expected market trends will help market regulators to take corrective measures.

We tried to gain insight into market behavior over time with an effective stock prediction model. The main objectives of the research are as follows:

1. To identify the impact of feature selection process and hyper-parameter optimization on prediction quality and metrics used in the prediction of stock market performance and prices.
2. To analyze historical data of NIFTY 50 and used it for training and validation purposes.
3. The use deep learning models to forecast the Price of NIFTY 50 index.
4. To examine the results and analyze the efficiency of each model evaluation metrics, features and epochs.

Those objectives investigate the best forecasting models. Thus, providing a comprehensive analysis of the NIFTY 50 index.

Primary Objectives of Stock Exchange

When examining a country's stock market, one can learn a great deal about its economy. It is the centre of the international financial system. People sometimes claim that it indicates the state of a country's economy. It is sometimes referred to as the "stock market". The performance of a country's stock market is directly related to the expansion of its commercial, industrial, and commercial sectors. A stock market may be a good concept for numerous objectives of stock exchange, including:

To Qualify for Long-term Financing

Commercial banks provide short-term loans most of the time. Therefore, the stock market serves as a source of long-term funding. If a firm wants to list its shares on a stock exchange, it must adhere to the following laws and regulations.

To Raise Capital

The primary function of a stock exchange is to assist companies raise capital. It was established to provide the necessary funds for the country's enterprises to operate. In order to achieve this objective, a private corporation will typically distribute stock certificates to the general public. By selling more shares of stock, the corporation can increase its capital by raising more funds.

To Protect Fraudulently

In addition, it prevents fraud. The norms and regulations of a stock exchange are always strictly adhered to. With these laws and restrictions, it may be able to prevent excessive trading in securities and price fluctuations.

The government also has the authority to control and monitor the stock market. In this approach, unscrupulous individuals could take advantage of less-savvy investors, and it is up to the stock to prevent this from happening.

Increasing the Efficiency of Trades

Trading stocks and other investment vehicles becomes much simpler when governed by laws. Without this closely controlled and well-coordinated stock market, it would be impossible to trade equities globally.

Using the stock exchange, anyone or any firm can purchase or sell stock in another company. There are millions of separate transactions occurring with the shares of thousands of different companies at any given time. As a global trading platform, the stock market facilitates communication between buyers and sellers by serving as a bridge.

Features:

- **A market for securities-** It is a wholesome market where securities of government, corporate companies, semi-government companies are bought and sold.
- **Second-hand securities-** It associates with bonds, shares that have already been announced by the company once previously.
- **Regulate trade in securities-** The exchange does not sell and buy bonds and shares on its own account. The broker or exchange members do the trade on the company's behalf.
- **Dealings only in registered securities-** Only listed securities recorded in the exchange office can be traded.
- **Transaction-** Only through authorised brokers and members the transaction for securities can be made.
- **Recognition-** It requires to be recognised by the central government.
- **Measuring device-** It develops and indicates the growth and security of a business in the index of a stock exchange.
- **Operates as per rules-** All the security dealings at the stock exchange are controlled by exchange rules and regulations and SEBI guidelines.

❖ **Folders and files used in the project:**

Project folder: This folder includes all the folders and files of the project

○ Introduction.py file: This file includes all the files of the project application

Pages folder: This folder includes the main Project file.

pycache folder: This folder includes all the cache files

Stock Price Prediction Line.ipynb: This is jupyter notebook file which have python codes which has been used for data analysis and machine learning algorithm

01_🔗Project.py: This file have python code which has been used for implementing Streamlit server.

Tools Used:

- VS Code
- Google Chrome
- Windows 11

1.)VS Code:

Visual Studio Code (VS Code) is a free and open-source code editor developed by Microsoft. It was first released in 2015 and has quickly become one of the most popular code editors among developers due to its flexibility, powerful features, and ease of use. In this article, I will explain what VS Code is, its features, and why it is a great choice for developers.

Overview VS Code is a lightweight and fast code editor that supports a wide range of programming languages. It runs on Windows, macOS, and Linux operating systems. The editor provides a rich set of features that make it easy to write, debug, and deploy code. It also has a strong and active community that contributes to the development of extensions, themes, and plugins that enhance its functionality. **Installation and Setup** Installing VS Code is easy, and the process is the same across all platforms. You can download the installer from the official website and follow the instructions to install it on your machine. Once installed, you can customize the editor's appearance and behaviour to suit your needs. You can change the theme, font, and colour scheme, as well as configure keyboard shortcuts, settings, and extensions.

User Interface The VS Code user interface is simple and intuitive, with a clean and uncluttered design. The editor has a sidebar that displays the file explorer, search, Git, and debug functionalities. The main workspace area is where you write and edit code, and it supports split views, tabs, and multiple windows. The status bar at the bottom of the window shows the language mode, indentation, and file encoding, among other information. **Editing and Debugging** VS Code provides a powerful and efficient editing experience that includes syntax highlighting, auto-completion, and formatting. The editor supports a wide range of programming languages, including JavaScript, Python, Java, PHP, C++, and many more. You can customize the editor's behaviour by installing extensions that add support for specific languages or frameworks. Debugging is an essential feature in any code editor, and VS Code provides a robust debugging experience that supports breakpoints, step-by-step execution, and conditional breakpoints. You can debug code written in multiple programming languages, and you can even debug code running on remote machines or in containers. **Extensions and Plugins** One of the strengths of VS Code is its extensibility. The editor supports a vast array of extensions and plugins that add new functionalities, themes, and tools. You can install extensions to support specific programming languages, frameworks, or tools, such as Git, Docker, or Azure. There are also extensions that add code snippets, templates, and IntelliSense. You can browse and install extensions from the VS Code Marketplace, which is a repository of extensions contributed by the community. The marketplace has over 20,000 extensions, and it's easy to find and install the ones you need. You can also create your extensions and contribute them to the marketplace.

Integrated Terminal VS Code comes with an integrated terminal that allows you to execute commands and run scripts without leaving the editor. The terminal supports multiple shells, including PowerShell, Bash, and Zsh. You can customize the terminal's

behaviour by configuring the shell environment, colours, and fonts. Version Control VS Code has built-in support for Git, which is a popular version control system used by developers to manage code changes. The editor provides a visual interface for Git that allows you to view the code changes, commit code, and resolve conflicts. You can also use Git to collaborate with other developers and manage code branches.

2.)Google Chrome:

Google Chrome is a web browser developed by Google that was first released in 2008. It quickly became one of the most popular web browsers due to its speed, simplicity, and ease of use. In this article, I will explain what Google Chrome is, its features, and why it is a great choice for internet users.

Overview Google Chrome is a free and opensource web browser that runs on Windows, macOS, Linux, Android, and iOS operating systems. The browser is designed to be fast, secure, and user-friendly. It uses the WebKit rendering engine and the V8 JavaScript engine to provide a fast and efficient browsing experience. The browser is known for its minimalist design and easy-to-use interface.

Installation and Setup Installing Google Chrome is easy, and the process is the same across all platforms. You can download the installer from the official website and follow the instructions to install it on your machine. Once installed, you can customize the browser's appearance and behaviour to suit your needs. You can change the theme, font, and colour scheme, as well as configure settings, bookmarks, and extensions.

User Interface The Google Chrome user interface is simple and intuitive, with a clean and uncluttered design. The browser has a toolbar that displays the address bar, search box, and tabs. The main browsing area is where you view web pages, and it supports tabs, windows, and full screen mode. The toolbar at the bottom of the window shows the download status, extensions, and settings.

Browsing Experience Google Chrome provides a fast and efficient browsing experience that includes tabbed browsing, bookmarks, and history. The browser supports multiple tabs, and you can open and close tabs with a single click. You can also reorder tabs, duplicate tabs, and pin tabs to the left or right of the tab bar.

Security and Privacy Google Chrome is designed to be secure and protect your privacy. The browser has built-in protection against malware, phishing, and other online threats. It also uses the latest encryption technologies to protect your data and prevent unauthorized access. Google Chrome also has several privacy features that allow you to control what information is shared with websites. You can control cookies, disable tracking, and use the incognito mode to browse the web without leaving any trace on your computer.

Extensions and Plugins One of the strengths of Google Chrome is its extensive library of extensions and plugins. You can browse and install extensions from the Chrome Web Store, which is a repository of extensions contributed by the community. The web store has over 190,000 extensions, and it's easy to find and install the ones you need.

Syncing and Integration Google Chrome allows you to sync your browsing data across multiple devices, including desktops, laptops, smartphones, and tablets. This means that you can access your bookmarks, history, and passwords from any device and pick up where you left off. Google Chrome also integrates with other Google services, such as Google Search, Gmail, and Google Drive. This integration allows you to access these services directly from the browser and enhances your overall browsing experience. Google Chrome also supports web technologies such as HTML5 and CSS3, which allow web developers to create rich and interactive web applications. This means that you can use Google Chrome to run web

applications that are as powerful as traditional desktop applications. Conclusion Google Chrome is a fast and secure.

3.)Windows 11:

Windows 11 is the latest major release of Microsoft's Windows NT operating system, released on October 5, 2021. It succeeded Windows 10 (2015) and is available for free for any Windows 10 devices that meet the new Windows 11 system requirements. Windows 11 features major changes to the Windows shell influenced by the cancelled Windows 10X, including a redesigned Start menu, the replacement of its "live tiles" with a separate "Widgets" panel on the taskbar, the ability to create tiled sets of windows that can be minimized and restored from the taskbar as a group, and new gaming technologies inherited from Xbox Series X and Series S such as Auto HDR and Direct Storage on compatible hardware. Internet Explorer (IE) has been replaced by the Chromium-based Microsoft Edge as the default web browser, like its predecessor, Windows 10, and Microsoft Teams is integrated into the Windows shell. Microsoft also announced plans to allow more flexibility in software that can be distributed via the Microsoft Store and to support Android apps on Windows 11 (including a partnership with Amazon to make its app store available for the function). Citing security considerations, the system requirements for Windows 11 were increased over Windows 10. Microsoft only officially supports the operating system on devices using an eighth-generation Intel Core CPU or newer (with some minor exceptions), a second-generation AMD Ryzen CPU or newer, or a Qualcomm Snapdragon 850 ARM system-on-chip or newer, with UEFI and Trusted Platform Module (TPM) 2.0 supported and enabled (although Microsoft may provide exceptions to the TPM 2.0 requirement for OEMs). While the OS can be installed on unsupported processors, Microsoft does not guarantee the availability of updates. Windows 11 removed support for 32-bit x86 and 32-bit ARM CPUs and devices that use BIOS firmware. Windows 11 has received a mostly positive reception. Pre-release coverage of the operating system focused on its stricter hardware requirements, with discussions over whether they were legitimately intended to improve the security of Windows or as a ploy to upsell customers to newer devices and over the e-waste associated with the changes. Upon release, it was praised for its improved visual design, window management, and stronger focus on security, but was criticized for various modifications to aspects of its user interface that were seen as worse than its predecessor; some were seen as an attempt to dissuade users from switching to competing applications. As of January 2024, Windows 11, at 27% worldwide, is the second most popular Windows version in use, with its predecessor Windows 10 at 2.5 times the market share. Windows 11 has an estimated 19.75% share of all PCs (the rest being other Windows editions and other operating systems such as macOS and Linux), and an estimated 7.8% share of all devices (including mobile, tablet and console) are running Windows 11. Due to anticompetitive practices, Microsoft was forced in the European Union to let users remove the Edge browser, Bing search engine, and advertisements to comply with users' interests

Tech Tools:

- Python
- Streamlit
- Jupyter Notebook



1.PYTHON:

Python is an interpreted, object-oriented, high-level programming language with dynamic semantics. Its high-level built in data structures, combined with dynamic typing and dynamic binding, make it very attractive for Rapid Application Development, as well as for use as a scripting or glue language to connect existing components together. Python's simple, easy to learn syntax emphasizes readability and therefore reduces the cost of program maintenance. Python supports modules and packages, which encourages program modularity and code reuse. The Python interpreter and the extensive standard library are available in source or binary form without charge for all major platforms, and can be freely distributed. Often, programmers fall in love with Python because of the increased productivity it provides. Since there is no compilation step, the edit-test-debug cycle is incredibly fast. Debugging Python programs is easy: a bug or bad input will never cause a segmentation fault.

Instead, when the interpreter discovers an error, it raises an exception. When the program doesn't catch the exception, the interpreter prints a stack trace. A source level debugger allows inspection of local and global variables, evaluation of arbitrary expressions, setting breakpoints, stepping through the code a line at a time, and so on. The debugger is written in Python itself, testifying to Python's introspective power. On the other hand, often the quickest way to debug a program is to add a few print statements to the source: the fast edit-test-debug cycle makes this simple approach very effective.



Streamlit

2.STREAMLIT:

Streamlit is a free and open-source framework to rapidly build and share beautiful machine learning and data science web apps. It is a Python-based library specifically designed for machine learning engineers. Data scientists or machine learning engineers are not web developers and they're not interested in spending weeks learning to use these frameworks to build web apps. Instead, they want a tool that is easier to learn and to use, as long as it can

display data and collect needed parameters for modeling. Streamlit allows you to create a stunning-looking application with only a few lines of code.

Streamlit is the easiest way especially for people with no front-end knowledge to put their code into a web application:

- No front-end (html, js, css) experience or knowledge is required.
- You don't need to spend days or months to create a web app, you can create a really beautiful machine learning or data science app in only a few hours or even minutes.
- It is compatible with the majority of Python libraries (e.g. pandas, matplotlib, seaborn, plotly, Keras, PyTorch, SymPy(latex)).
- Less code is needed to create amazing web apps.
- Data caching simplifies and speeds up computation pipelines.



3.JUPYTER NOTEBOOK:

Jupyter Notebook(formerly known as Python Notebook) is an interactive web application for creating and sharing computational documents documents. The project was first named IPython and later renamed Jupyter in 2014. It is fully open-source product, and users can use every functionality available for free. It supports more than 40 languages including Python, R, and Scala.

A notebook is a mutable file saved in .ipynb format. Jupyter Notebook has a notebook dashboard to help users manage different notebooks. Kernels are also part of Jupyter Notebooks. Kernels and processes that run interactive code in a particular programming language and return output to the user. Kernels also respond to tab completion and introspection requests. Jupyter notebooks can be converted to an open standard format such as HTML LaTeX, PDF, Markdown, and Python by using the “Download As” function in the web interface.

Jupyter notebooks are used for a variety of purposes. A notebook is an interactive computational environment in which users can execute a particular piece of code and observe the output and make changes to the code to drive it to the desired output or explore more. Jupyter notebooks are heavily used for data exploration purposes as it involves a lot of reiterations. It is also used in other data science workflows such as machine learning experimentations and modelling. It can also be used for documenting code samples. A Jupyter notebook has independent executable code cells that users can run in any order. Documentation can be done by alternating between code and markdown cells.

Software Development Life Cycle

The software development life cycle (SDLC) of the project follows the Agile development model, which is an iterative and incremental approach to software development that emphasizes flexibility, collaboration, and customer satisfaction.

The SDLC of the project includes the following phases:

Requirement Gathering: In this phase, the project team gathers requirements from the customer and creates a product backlog, which is a prioritized list of features and requirements for the project.

Planning: In this phase, the project team plans the sprints and creates a sprint backlog, which is a list of items from the product backlog that will be completed in the current sprint.

Design: In this phase, the project team designs the software architecture and creates a detailed design of the software components.

Development: In this phase, the project team develops the software components based on the design and the sprint backlog.

Testing: In this phase, the project team tests the software components to ensure that they meet the requirements and are free of defects.

Deployment: In this phase, the project team deploys the software components to the production environment and makes them available to the customer.

Maintenance: In this phase, the project team provides ongoing support and maintenance for the software components, including bug fixes and updates.

The Agile development model used in this project allows for continuous improvement and adaptation to changing requirements. The iterative and incremental approach allows the project team to deliver working software at the end of each sprint, which can be used by the customer for feedback and validation.

The Agile development model also encourages collaboration and communication between the development team and the customer. The customer is involved in the development process and provides valuable feedback, which helps ensure that the software meets their needs and requirements.

In summary, the SDLC of the project follows the Agile development model, which is an iterative and incremental approach to software development that emphasizes flexibility, collaboration, and customer satisfaction. The SDLC includes the phases of requirement gathering, planning, design, development, testing, deployment, and maintenance. The Agile development model allows for continuous improvement and adaptation to changing requirements, while the focus on collaboration and communication ensures that the customer is involved in the development process and provides valuable feedback.

• **Requirement Gathering and Analysis**

In this phase the first and foremost I have collected suitable data set for the project, which is core of the project. Then that data set is analysed to prevent further issues like wrong recommendations, using jupyter notebook software and data analysis codes (Python).

• **Hardware Requirements Server**

- i. Intel i3/i5/i7
processor
- ii. 4 GB Ram
- iii. 80 GB+ Hard
disk space
- iv. Keyboard and
mouse
- v. Internet
Connection

• **Software Requirements**

- i. Server based operating
system
- ii. Streamlit Server
- iii. VS Code
- iv. Jupyter Notebook

• **Feasibility Study**

A feasibility study is an analysis of the practicality and viability of a project or a system. In this section, we will conduct a feasibility study for the linear regression model for predicting stock prices.

Technical Feasibility: The technical feasibility of the project is high. The linear regression model is a well-established and widely used machine learning algorithm, which is available in many programming languages and libraries. The project requires only a small dataset, which

can be easily obtained from public sources or financial institutions. The preprocessing and training of the model can be done using standard techniques and tools.

Economic Feasibility: The economic feasibility of the project is moderate. The project requires a small investment in hardware and software, which can be easily afforded by most organizations. The operating costs of the project are also low, as the model can be trained and deployed using standard machine learning platforms. However, the benefits of the project may not be immediate or direct, as the model is intended to provide predictions and insights into the stock market, rather than generating revenue or reducing costs.

Operational Feasibility: The operational feasibility of the project is high. The model can be easily integrated into existing financial systems or applications, as it requires only a small amount of data and processing power. The model can be trained and updated on a regular basis, using new data or market conditions. The model can also be customized to fit the specific needs and requirements of the organization or the user.

Schedule Feasibility: The schedule feasibility of the project is high. The project can be completed in a short period of time, as it requires only a small dataset and a simple machine learning model. The project can be divided into small tasks or milestones, which can be completed in a few days or weeks. The project can also be iterative, as the model can be improved and refined over time, based on feedback and evaluation.

In conclusion, the feasibility study shows that the linear regression model for predicting stock prices is technically, economically, operationally, and schedule feasible. The project requires a small investment and can provide valuable insights and predictions into the stock market. The model can be easily integrated into existing financial systems or applications, and can be customized to fit the specific needs and requirements of the organization or the user. The project can be completed in a short period of time, and can be iterative and adaptive, based on feedback and evaluation.

• Design/DFDS

The Fig. Explains that how data can be processed and meets the user requirements.

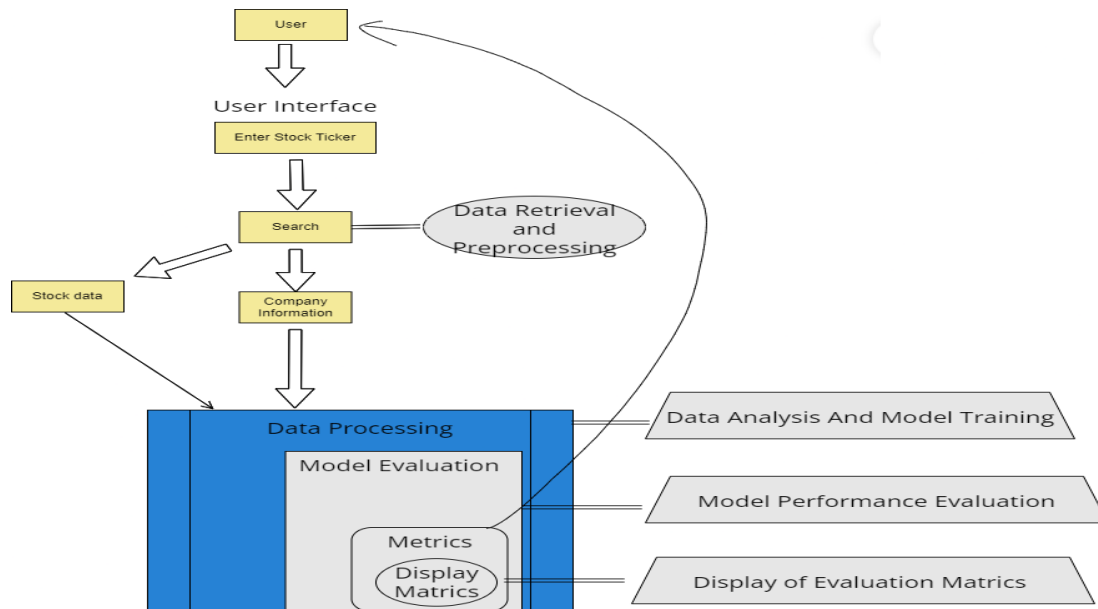


FIG 1. The Above DFD shows the working of Project

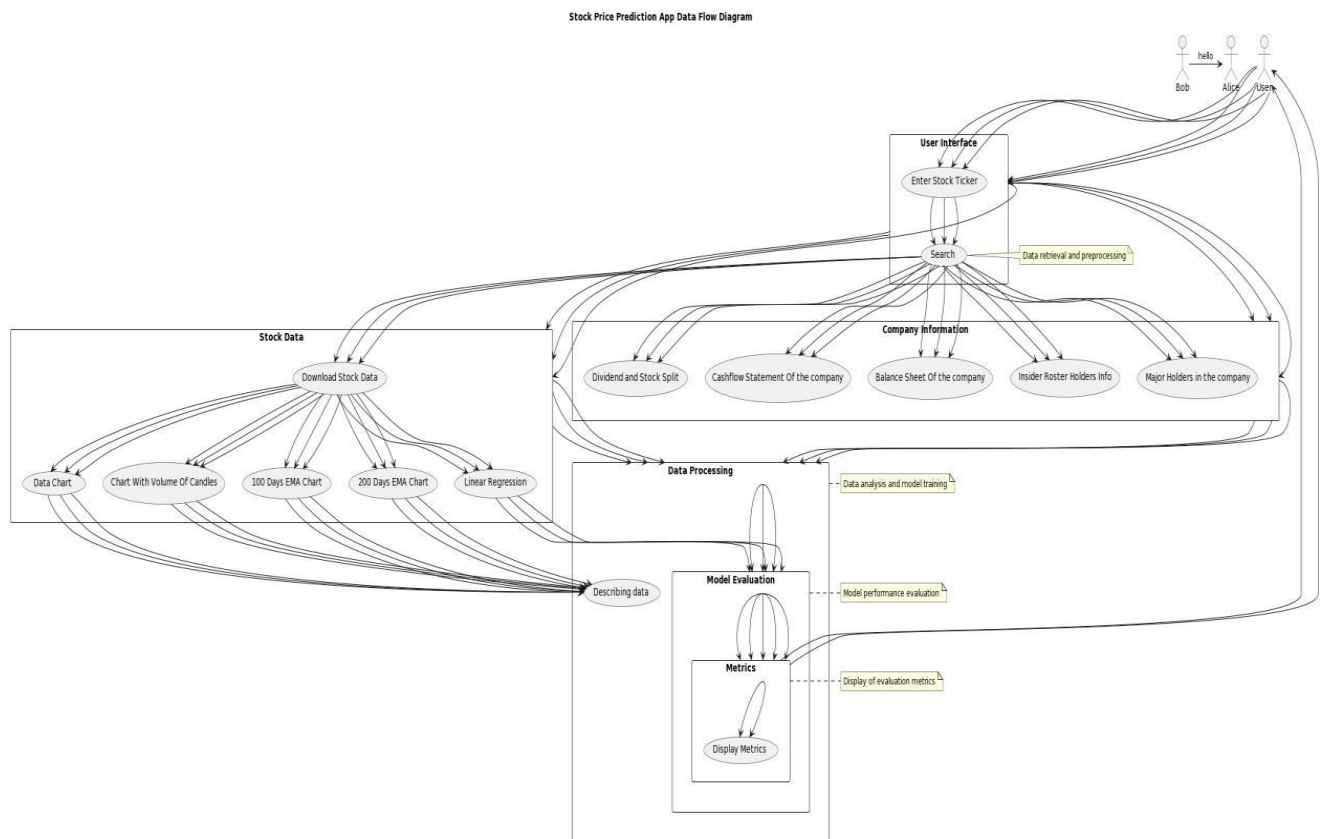


FIG 2. The Fig Shows the working of the Project in Detail

❖ Coding

❖ Stockline/Data Analysis/Stock Price Prediction Line.ipynb

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import plotly.graph_objs as go
import yfinance as yf
import mplfinance as mpf
import datetime

#for offline plotting
from plotly.offline import download_plotlyjs,init_notebook_mode,ipplot, plot
init_notebook_mode(connected=True)
```

Python Data Cleaning using commands

```
# Get the current date
end = datetime.date.today()

# Set the start date to 10 years ago
start = end- datetime.timedelta(days=365*14)
```

```
df = yf.Ticker('ONGC.NS')
```

```
df.actions
df.cashflow
df.balance_sheet
df.insider_roster_holders
df.major_holders
```

```
df = yf.download('ONGC.NS', start, end)
```

In [2]:

Out[2]:

```
c:\Users\WAHEGURU JI\AppData\Local\Programs\Python\Python39\lib
\site-packages\pandas\core\dtypes\cast.py:1057: RuntimeWarning:
```

```
invalid value encountered in cast
```

```
c:\Users\WAHEGURU JI\AppData\Local\Programs\Python\Python39\lib\site-packag
es\pandas\core\dtypes\cast.py:1081: RuntimeWarning:
```

```
invalid value encountered in cast
```

```
[*****100%*****] 1 of 1 completed
```

In [3]:

```
df.columns
```

Out[3]:

```
Index(['Open', 'High', 'Low', 'Close', 'Adj Close', 'Volume'], dtype='object')
```

In [4]:

```
df = df.reset_index()
df['Date'] = pd.to_datetime(df['Date'])
```

```
# Set 'date_column' as the index
df.set_index('Date', inplace=True)
```

```
df.head(1000)
```

In [5]:

Out[5]:

	Open	High	Low	Close	Adj Close	Volume
Date						
2010-05-10	175.66667 2	175.76666 3	172.66667 2	175.14999 4	94.357590	3745860
2010-05-11	174.93333 4	174.93333 4	170.13333 1	171.65834 0	92.476555	6134346
2010-05-12	172.46666 0	172.64999 4	170.46666 0	171.32499 7	92.296959	4782966
2010-05-13	171.70832 8	175.25833 1	171.33332 8	174.31666 6	93.908691	7500012
2010-05-14	174.31666 6	175.91667 2	173.33332 8	174.09166 0	93.787437	5117970
...
2014-05-15	244.93333 4	253.93333 4	242.36666 9	252.36666 9	156.26586 9	8127180
2014-05-16	261.33334 4	277.56668 1	248.39999 4	256.10000 6	158.57753 0	2003953 0
2014-05-19	262.63333 1	280.00000 0	259.56668 1	277.83334 4	172.03483 6	1219656 4
2014-05-20	278.93331 9	280.00000 0	264.16665 6	266.73333 7	165.16174 3	1054842 4

	Open	High	Low	Close	Adj Close	Volume
Date						
2014	264.06668	268.50000	260.53332	265.63333	164.48062	8455851
-05-	1	0	5	1	1	
21						

1000 rows × 6 columns

df.tail(1000)

In [6]:

Out[6]:

	Open	High	Low	Close	Adj Close	Volume
Date						
2020	65.900002	66.349998	60.799999	65.349998	52.198143	7384070
-04-						7
22						
2020	67.500000	69.050003	66.550003	67.349998	53.795628	3820737
-04-						7
23						
2020	67.150002	69.500000	66.599998	67.599998	53.995323	3158809
-04-						7
24						
2020	68.000000	69.500000	67.650002	68.449997	54.674255	1672388
-04-						9
27						
2020	67.800003	69.699997	67.099998	69.199997	55.273319	2042262
-04-						6
28						
...
2024	283.00000	286.14999	282.20001	282.89999	282.89999	1107029
-04-	0	4	2	4	4	0
26						

	Open	High	Low	Close	Adj Close	Volume
Date						
2024-04-29	283.950012	285.250000	282.149994	283.200012	283.200012	8952420
2024-04-30	284.500000	286.350006	281.450012	282.850006	282.850006	10658906
2024-05-02	281.000000	284.600006	279.049998	282.799998	282.799998	15677336
2024-05-03	284.000000	292.950012	284.000000	286.100006	286.100006	28913016

1000 rows × 6 columns

In [7]:

```
df.info()

<class 'pandas.core.frame.DataFrame'>
DatetimeIndex: 3452 entries, 2010-05-10 to 2024-05-03
Data columns (total 6 columns):
 #   Column      Non-Null Count  Dtype  
---  -
 0   Open        3452 non-null   float64
 1   High        3452 non-null   float64
 2   Low         3452 non-null   float64
 3   Close       3452 non-null   float64
 4   Adj Close   3452 non-null   float64
 5   Volume      3452 non-null   int64   
dtypes: float64(5), int64(1)
memory usage: 188.8 KB
```

Note:No null value in data set

In [8]:

```
df.info()

<class 'pandas.core.frame.DataFrame'>
DatetimeIndex: 3452 entries, 2010-05-10 to 2024-05-03
Data columns (total 6 columns):
 #   Column      Non-Null Count  Dtype  
---  -
 0   Open        3452 non-null   float64
 1   High        3452 non-null   float64
 2   Low         3452 non-null   float64
 3   Close       3452 non-null   float64
 4   Adj Close   3452 non-null   float64
```



```
5    Volume    3452 non-null    int64
dtypes: float64(5), int64(1)
memory usage: 188.8 KB
```

In [9]:

```
#to remove null vaues from column
df.dropna(inplace=True)
```

In [10]:

```
pd.isnull(df).sum()
```

Out[10]:

```
Open      0
High      0
Low       0
Close     0
Adj Close 0
Volume    0
dtype: int64
```

In [11]:

```
#now our rows were deleted which have the null values
df.shape
```

Out[11]:

```
(3452, 6)
```

In [12]:

```
# to see the number of columns name
df.columns
```

Out[12]:

```
Index(['Open', 'High', 'Low', 'Close', 'Adj Close', 'Volume'], dtype='object')
```

In [13]:

```
df.describe()
```

Out[13]:

	Open	High	Low	Close	Adj Close	Volume
count	3452.000000	3452.000000	3452.000000	3452.000000	3452.000000	3.452000e+03
mean	172.077448	174.335134	169.525242	171.765856	119.463646	1.148300e+07
std	44.223669	44.792180	43.578244	44.143962	34.054943	1.292530e+07
min	59.400002	63.000000	50.000000	60.000000	44.612835	0.000000e+00
25%	146.345837	148.237499	144.250000	146.350006	100.594961	4.757850e+06
50%	174.449997	176.416664	172.100006	174.149994	114.723156	7.460804e+06

	Open	High	Low	Close	Adj Close	Volume
75%	194.94166 6	196.69999 7	192.06250 4	194.17500 3	131.44391 3	1.293113e+ 07
max	313.33334 4	314.56668 1	298.66665 6	310.43331 9	286.10000 6	1.780948e+ 08

Exploring the data analysis

In [14]:

```
d=df[['High','Low','Open','Close']]
fig=mpf.plot(df,type='candle',volume=True)
c:\Users\WAHEGURU_JI\AppData\Local\Programs\Python\Python39\lib\site-packag
es\mplfinance\_arg_validators.py:84: UserWarning:
```

```
=====

WARNING: YOU ARE PLOTTING SO MUCH DATA THAT IT MAY NOT BE
        POSSIBLE TO SEE DETAILS (Candles, Ohlc-Bars, Etc.)
For more information see:
- https://github.com/matplotlib/mplfinance/wiki/Plotting-Too-Much-Data
```

```
TO SILENCE THIS WARNING, set `type='line'` in `mpf.plot()`
OR set kwarg `warn_too_much_data=N` where N is an integer
LARGER than the number of data points you want to plot.
```

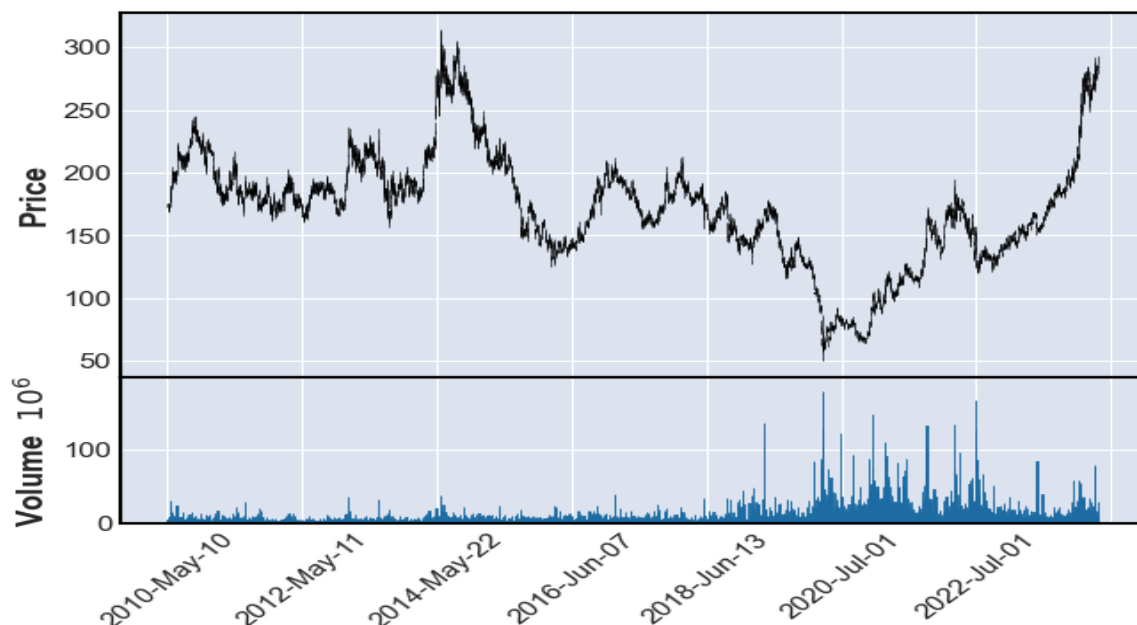


Fig 3.It shows the Volume and Closing,Opening Of the Stock

Note: Candle Kind graph to show the dataset

In [15]:

```
ma100 = df.Close.rolling(100).mean()
fig=plt.figure(figsize=(12,6))
plt.plot(df.Close)
plt.plot(ma100)
```

Out[15]:

```
[<matplotlib.lines.Line2D at 0x2c7246014f0>]
```

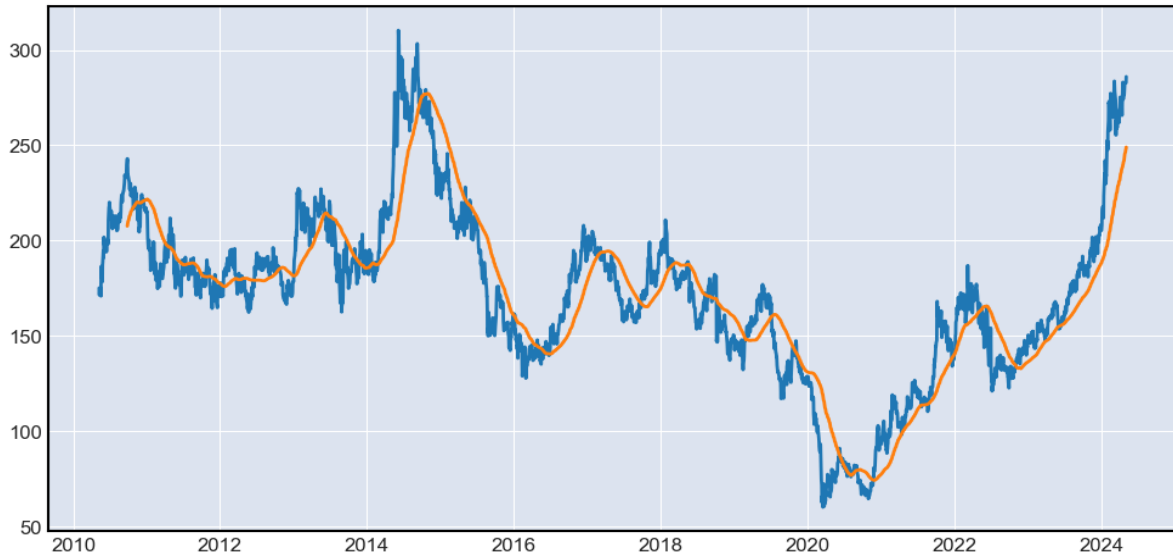


Fig 4.Shows the 100 days EMA Chart

In [16]:

```
ma100 = df.Close.rolling(100).mean()
ma200 = df.Close.rolling(200).mean()
fig=plt.figure(figsize=(12,6))
plt.plot(df.Close)
plt.plot(ma100)
plt.plot(ma200)
```

Out[16]:

```
[<matplotlib.lines.Line2D at 0x2c72467ad00>]
```



Fig 5.Shows the 100 and 200 days EMA Chart

```

df = df.reset_index()
df.columns

In [17]:

Out[17]:
Index(['Date', 'Open', 'High', 'Low', 'Close', 'Adj Close', 'Volume'], dtype='object')

In [18]:
#Setting the layout of the graph
layout = go.Layout(
    title='Stock prices',
    xaxis=dict(
        title='Year',
        titlefont=dict(
            family='Courier New,monospace',
            size=18,
            color='black'
        )
    ),
    yaxis=dict(
        title='Price',
        titlefont=dict(
            family='Courier New,monospace',
            size=18,
            color='black'
        )
    )
)
Adani_data=[{'x':df['Date'],'y':df['Close']}]
plot= go.Figure(data=Adani_data,layout=layout)

In [19]:
#to plot the graph in offline
iplot(plot)

```

Note:It shows the stock price of Adni ports

Now building the linear regression model

```

In [20]:
#build the regresssion model
from sklearn.model_selection import train_test_split

#for preprocssing
from sklearn.preprocessing import StandardScaler

#for model evaulation and for accuracy
from sklearn.metrics import mean_squared_error as mse
from sklearn.metrics import r2_score

In [21]:
#now we can split the data into training set and testing set
from numpy import reshape

X=ny.array(df.index).reshape(-1, 1)
Y=df['Close']

In [22]:
X_train,X_test,Y_train,Y_test=train_test_split(X,Y,test_size=0.3,random_state=101)

```

```
scaler= StandardScaler().fit(X_train)
```

Imporing Linear Model

In [23]:

```
from sklearn.linear_model import LinearRegression
```

In [24]:

```
#creating a linear model
lm=LinearRegression()
lm.fit(X_train,Y_train)
```

Out[24]:

```
LinearRegression()
```

In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook. On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

In [25]:

```
#plot actual and predicted values to train dataset
trace0=go.Scatter(
    x=X_train.T[0],
    y=Y_train,
    mode='markers',
    name= 'Actual'
)
tracel=go.Scatter(
    x=X_train.T[0],
    y=lm.predict(X_train).T,
    mode='lines',
    name= 'Predicted'
)
data=[trace0,tracel]
layout.xaxis.title.text = 'Day'
plot2=go.Figure(data=data, layout=layout)
```

In [26]:

```
ipplot(plot2)
```

Note:Blue dots Represent the actual value and the orange line represent the predicted values

In [27]:

```
#Cal score of final model evaluation
score=f'''
{'Metric'.ljust(10)}{'Train'.center(20)}{'Test'.center(20)}
{'r2_score'.ljust(10)}{r2_score(Y_train,lm.predict(X_train))}\t{r2_score(Y_
test,lm.predict(X_test))}
{'MSE'.ljust(10)}{mse(Y_train,lm.predict(X_train))}\t{mse(Y_test,lm.predict
(X_test))}
'''
```

```
print(score)
```

Metric	Train	Test
r2_score	0.18578738342761525	0.23952480643137042
MSE	1635.4858238044196	1373.818047619064

❖ Stockline/📊 Introduction.py

```
import streamlit as st

st.set_page_config(
    page_title="Introduction",
    page_icon="📊",
)

st.markdown(
    """# 📊 **Stockline**
    ### **Predicting Stocks with ML**

    **Stockline is an ML-powered stock price prediction app built
    with Python and Streamlit. It utilizes machine learning models
    to forecast stock prices and help investors make data-driven
    decisions.**

    ## 🏠 **How It's Built**

    Stockline is built with these core frameworks and modules:

    - **Streamlit** - To create the web app UI and interactivity
    - **YFinance** - To fetch financial data from Yahoo Finance API
    - **LinearRegression Models** - To build the prediction model
    - **Plotly** - To create interactive financial charts

    The app workflow is:

    1. User selects a stock ticker
    2. Historical data is fetched with YFinance
    3. model is trained on the data
    4. Model makes multi-day price forecasts
    5. Results are plotted with Plotly

    ## 🔄 **Key Features**

    - **Real-time data** - Fetch latest prices and fundamentals
    - **Financial charts** - Interactive historical and forecast
    charts
    - **ARIMA forecasting** - Make statistically robust predictions
    - **Backtesting** - Evaluate model performance
    - **Responsive design** - Works on all devices

    ## 🚀 **Getting Started**

    ### **Local Installation**

    1. Clone the repo

    ```bash
 git clone https://github.com/user/stockline.git
    ```
```

2. Install requirements

```
```bash
pip install -r requirements.txt
```
```

3. Run the app


```
```bash
streamlit run Introduction.py
```
```

The app will be live at ```http://localhost:8502```

☒ **Future Roadmap**

Some potential features for future releases:

- **More advanced forecasting models like LSTM**
- **Quantitative trading strategies**
- **Portfolio optimization and tracking**
- **Additional fundamental data**
- **User account system**

 **Disclaimer**

This is not financial advice! Use forecast data to inform your own investment research. No guarantee of trading performance.

"""

)

❖ Stockline/pages/01_📈Project.py

```
import streamlit as st

from sklearn.linear_model import LinearRegression
import numpy as ny
import pandas as pd
import matplotlib.pyplot as plt
import plotly.graph_objs as go

from plotly.offline import plot
from sklearn.discriminant_analysis import StandardScaler
from sklearn.model_selection import train_test_split
import yfinance as yf
import datetime
#for offline plotting
from plotly.offline import plot

from timeinterval import fetch_periods_intervals,
fetch_stock_history

st.set_page_config(
    page_title="Prediction",
    page_icon="📈",
)

st.title('Stock Price Predicted Line')
# Get the current date

end = datetime.date.today()

# Set the start date to 10 years ago
start = end - datetime.timedelta(days=365*14)#In this case, the
code sets the days parameter to 365*14, which calculates the
number of days in 10 years (365 days * 10) and multiplies it by
14 to add an additional 4 years as a safety margin to ensure
that the data covers a full 10-year period

st.sidebar.write("Note : Enter the Stock ticker Symbol First")
user_input=st.sidebar.text_input('Enter Stock ticker')
st.sidebar.button('Search')
st.header(user_input)

def get_stock_history(symbol):
    stock_data = yf.download(symbol)
    return stock_data

def get_stock_name(symbol):
    stock = yf.Ticker(symbol)
    return stock.info['shortName']
if user_input:
    stock_name = get_stock_name(user_input)
    st.write(f"Stock Name: {stock_name}")
```



```

df = yf.Ticker(user_input)
Cashflow_Statement,Balance_Sheet,Holders_Info=st.tabs(["Cashf
low Statement Of the company","Balance Sheet Of the
company","Insider Roster Holders Info"])
with Holders_Info:
    df.insider_roster_holders
with Cashflow_Statement:
    df.cashflow
with Balance_Sheet:
    df.balance_sheet

Data_Chart,major_holders,
Dividend_and_stock_split=st.tabs(["Data Chart","Major Holders
in the company","Shows the Dividend and stock split given the
company"])
with major_holders:
    df.major_holders
with Dividend_and_stock_split:
    df.actions

df = yf.download(user_input, start, end)
st.sidebar.write("Data Starting date :",start)
st.sidebar.write("Data Ending date :",end)

#Describing data
with Data_Chart:
    st.subheader('Data Chart')
    df = df.reset_index()
    st.write(df)

#Visualisation
st.subheader('Closing Price Chart with Date ')
layout = go.Layout(
    title='Stock price',
    xaxis=dict(
        title='Year',
        titlefont=dict(
            family='Courier New,monospace',
            size=18,
            color='red'
        )
    ),
    yaxis=dict(
        title='Price',
        titlefont=dict(
            family='Courier New,monospace',
            size=18,
            color='red'
        )
    )
)
Stock_data=[{'x':df['Date'],'y':df['Close']}]
plot= go.Figure(data=Stock_data,layout=layout)
st.plotly_chart(plot)

```

```

#for Date error in predicted chart
df['Date'] = pd.to_datetime(df['Date'])
# Set 'date_column' as the index
df.set_index('Date', inplace=True)

st.header("Stock Intraday Data")
# Get the user input for the period
periods = fetch_periods_intervals()

# Add a selector for period
period = st.selectbox("Choose a period", list(periods.keys()))

# Add a selector for interval
interval = st.selectbox("Choose an interval", periods[period])

# Fetch the stock history if the user inputs are valid
if user_input and period and interval:
    stock_history = fetch_stock_history(user_input, period,
interval)
    # Create the plotly figure

# Create a new trace for the candlestick chart
candlestick = go.Candlestick(
    x=stock_history.index,
    open=stock_history['Open'],
    high=stock_history['High'],
    low=stock_history['Low'],
    close=stock_history['Close']
)

# Create the plotly figure
figure = go.Figure(
    data=[
        candlestick
    ],
    layout=go.Layout(
        title=f'{stock_name} Intraday Chart',
        xaxis=go.layout.XAxis(title='Date'),
        yaxis=go.layout.YAxis(title='Value')
    )
)

# Display the plot in Streamlit
st.plotly_chart(figure)

One_Hundred_Days_EMA_Chart,Two_Hundred_Days_EMA_Chart=st.tabs
(["100 Days EMA Chart","200 Days EMA Chart"])
with One_Hundred_Days_EMA_Chart:
    #using 100 days EMA Indicator for analysis
    st.subheader('100 Days EMA Chart')
    ma100= df.Close.rolling(100).mean()
    fig = plt.figure(figsize=(12,6))
    plt.plot(df.Close)
    plt.plot(ma100)
    st.pyplot(fig)
with Two_Hundred_Days_EMA_Chart:

```

```

#using 200 days EMA Indicator for analysis
st.subheader('200 Days EMA Chart')
ma100= df.Close.rolling(100).mean()
ma200= df.Close.rolling(200).mean()
fig = plt.figure(figsize=(12,6))
plt.plot(df.Close)
plt.plot(ma100)
plt.plot(ma200)
st.pyplot(fig)

st.title('Linear Regression - Actual vs Predicted Values')
from sklearn.linear_model import LinearRegression
# Example data
X_train = ny.random.rand(2274, 1) # Replace with your actual
features
y_train = ny.random.rand(2274)

#now we can split the data into training set and testing set
from numpy import reshape

X=ny.array(df.index).reshape(-1, 1)
Y=df['Close']

X_train,X_test,Y_train,Y_test=train_test_split(X,Y,test_size=
0.3,random_state=101)
scaler= StandardScaler().fit(X_train)

# Create and fit a linear model
lm = LinearRegression()
lm.fit(X_train, Y_train)

# Check and convert data types if needed
if X_train.dtype == ny.dtype('datetime64[ns]'):
    # Convert datetime to numerical representation
    X_train = X_train.astype(ny.float64)

# Convert X_train to a Pandas dataframe
X_train = pd.DataFrame(X_train)
# Convert the first column of X_train to a datetime object
X_train['Date'] = pd.to_datetime(X_train.iloc[:, 0])
# Set the first column of X_train as the index
X_train.set_index('Date', inplace=True)

trace0=go.Scatter(
    x=X_train.index,
    y=Y_train,
    mode='markers',
    name= 'Actual'
)
tracel=go.Scatter(
    x=X_train.index,
    y=lm.predict(X_train).T,
    mode='lines',
    name= 'Predicted'
)

```

```

data=[trace0,trace1]

# Set the xaxis type to 'date'
layout.xaxis.type = 'date'
layout.xaxis.title.text = 'Date'
plot2=go.Figure(data=data, layout=layout)
st.plotly_chart(plot2)
#Cal score of final model evaluation

#for model evaulation and for accuracy
from sklearn.metrics import mean_squared_error as mse
from sklearn.metrics import r2_score
# Check and convert data types if needed
if X_test.dtype == ny.dtype('datetime64[ns]'):
    # Convert datetime to numerical representation
    X_test = X_test.astype(ny.float64)

st.text("This Metric Shows the trained data and testing data
:")
score=f'''
{'Metric'.ljust(10)}{'Train'.center(20)}{'Test'.center(20)}
{'r2_score'.ljust(10)}{r2_score(Y_train,lm.predict(X_train))}
\t{r2_score(Y_test,lm.predict(X_test))}
{'MSE'.ljust(10)}{mse(Y_train,lm.predict(X_train))}\t{mse(Y_t
est,lm.predict(X_test))}
'''
print(score)
st.text("Metrics:")
st.text(score)

```

• Project Screenshots:-

Homepage:

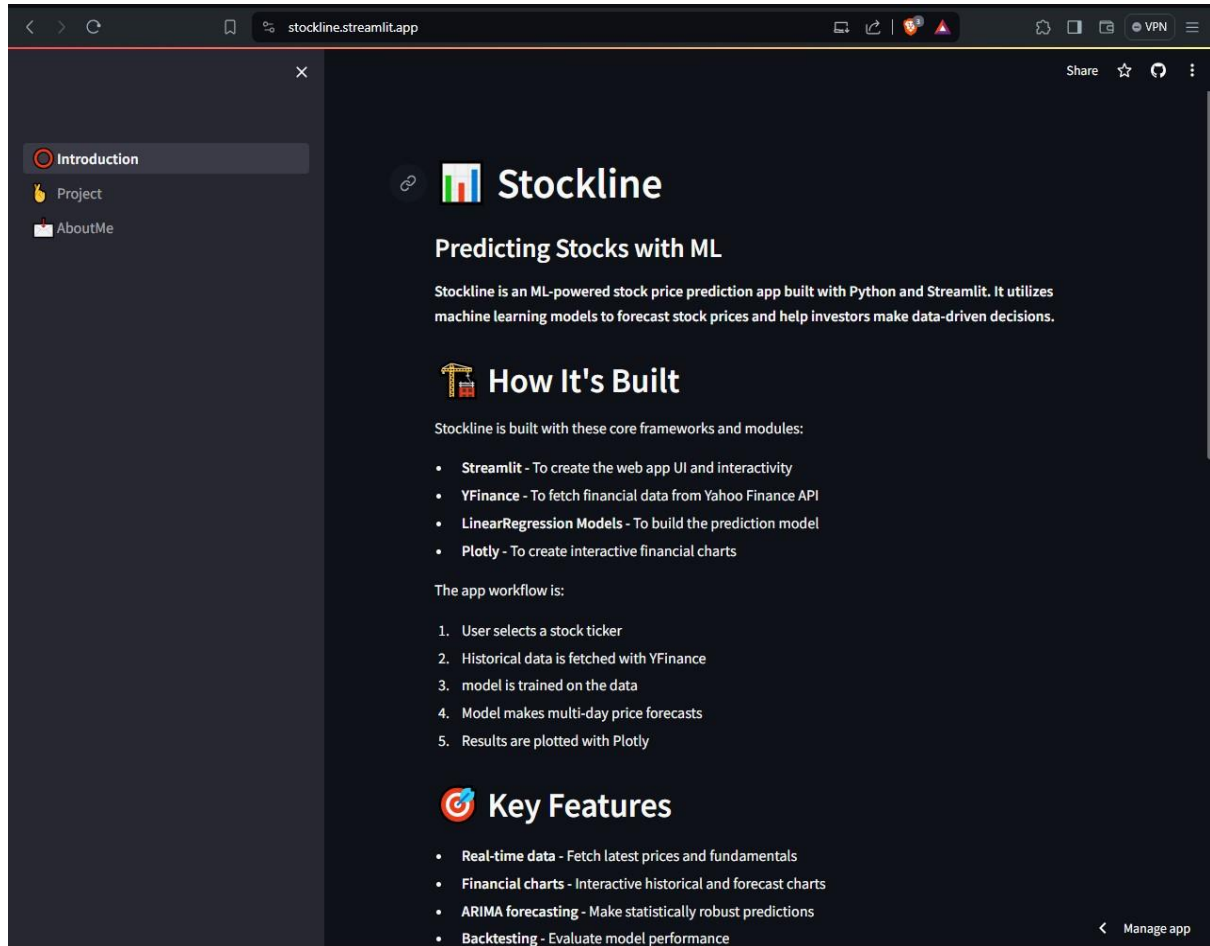


FIG 6. The Introduction Page of the Project

The Above Picture Shows The Introduction And Features Of the Project .After that when we change the page the next page will be executed.

This Introduction Page tells the whole summary of the project and the resources used in the project.

Project Page:

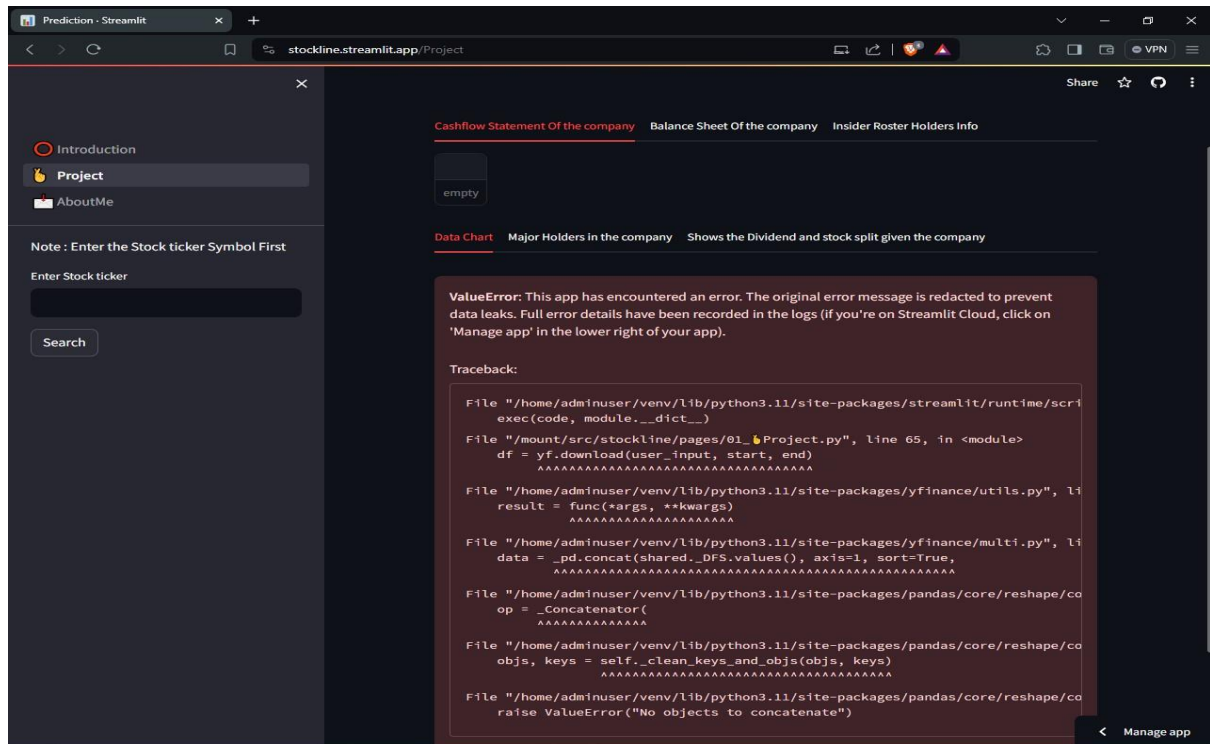


FIG 7. The Main Project Page of the Project

The above shown picture is the first page of my website, in this page you have to enter the stock ticker symbol, after reading whole page the Search button will help to reach to the required Stock data For example: when we search the data of Apple Company using their ticker symbol “AAPL” then it shows the data.

Note: The Ticker Symbol can be found on <https://finance.yahoo.com>.

Result page:

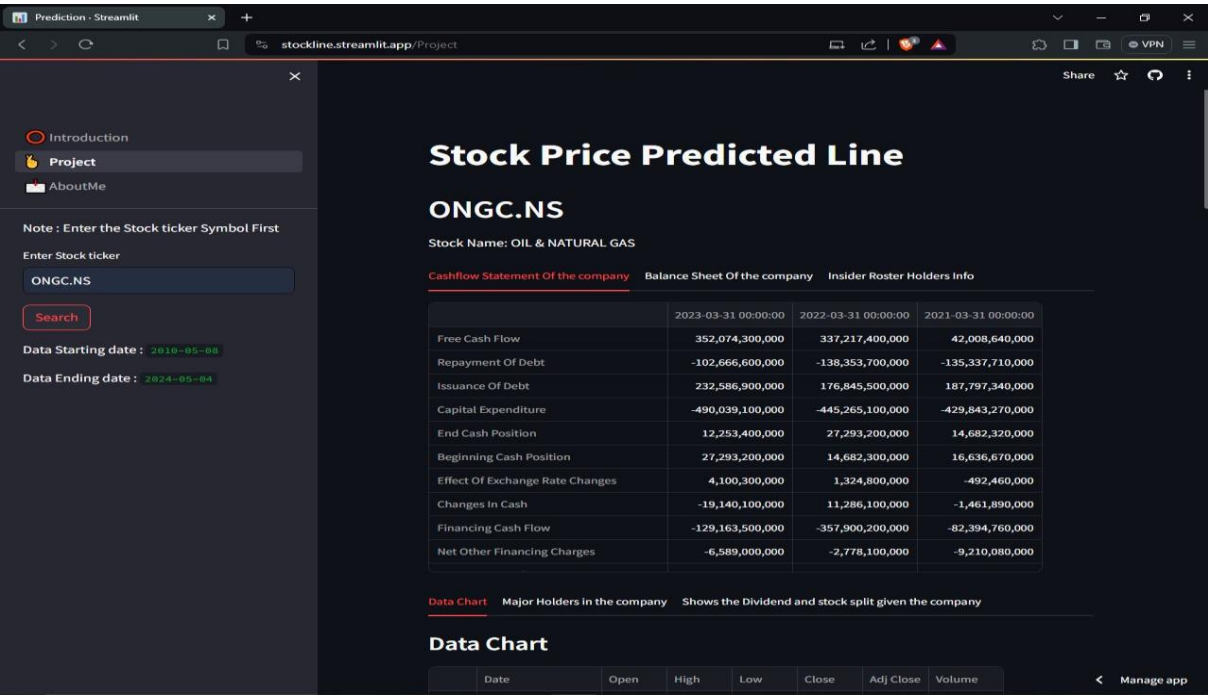


FIG 8. The Stock Data of the Project

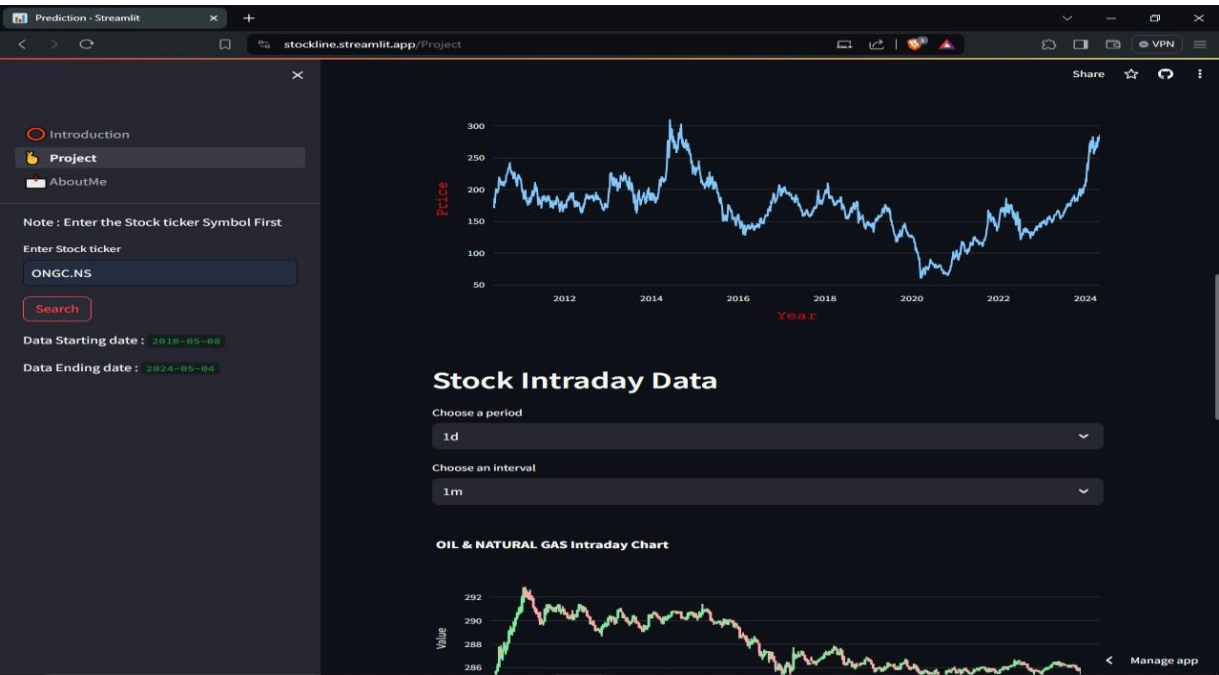


FIG 9. The Output of the Project

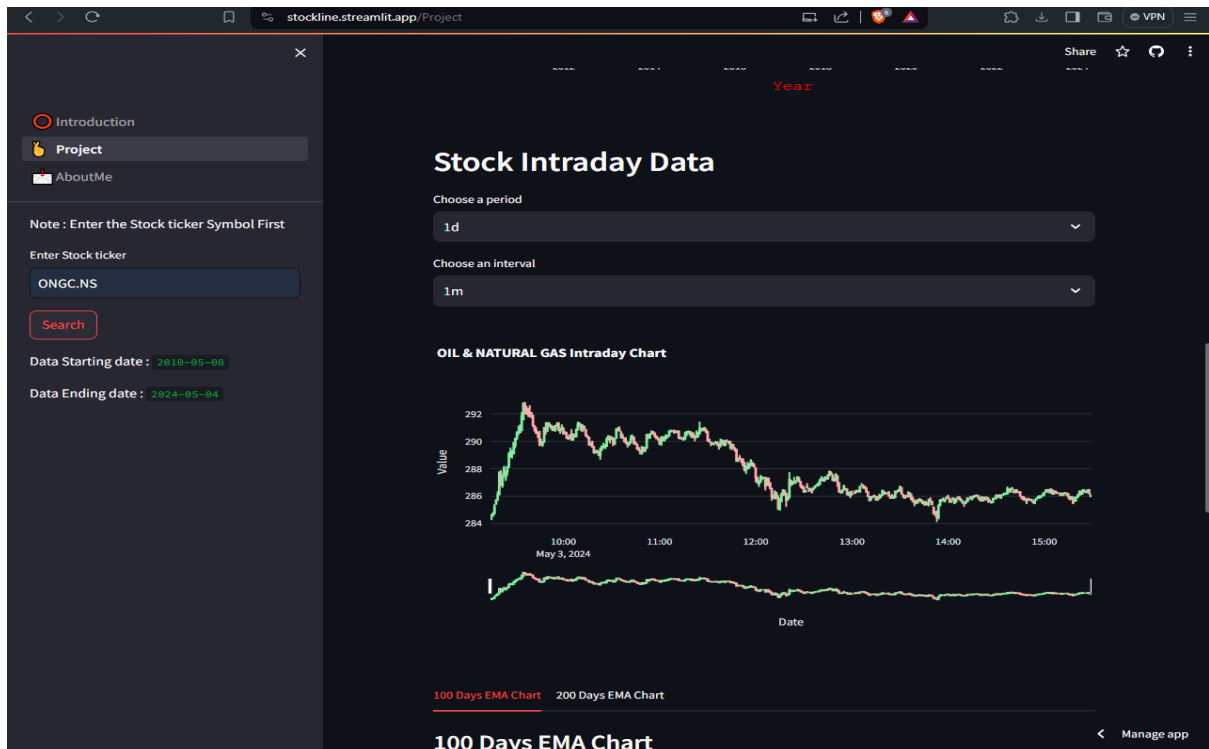


FIG 10. The Intraday Chart of the Project

In this picture the result is represented according to the to the stock symbol entered in the search box on the left like “ONGC.NS” which represents the “OIL AND NATURAL GAS” And “.NS” Represent the national stock.

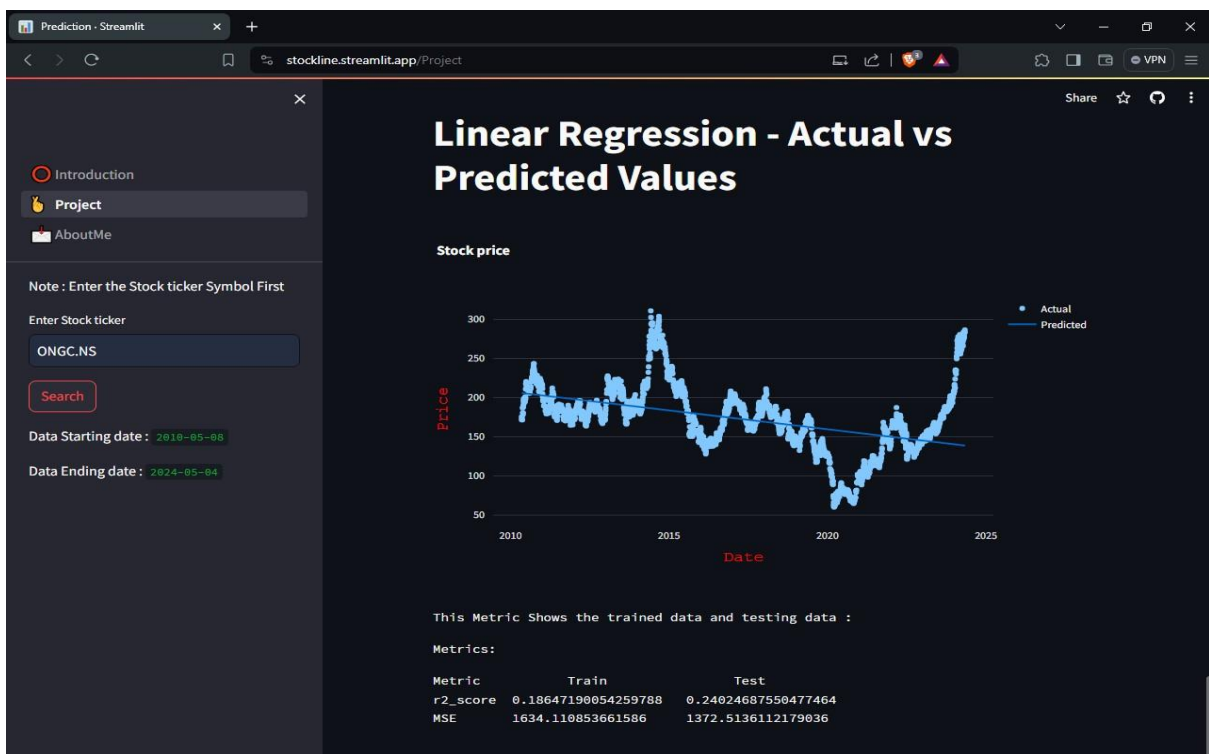


FIG 11. The Predicted Chart of the Project

In this picture the chart shows the predicted and actual data using Linear Regression Model where line shows the predicted price and dots shows the actual data.

In Metrics Section the code calculates the R-squared score and mean squared error (MSE) for both the training and testing sets to evaluate the performance of the linear regression model.

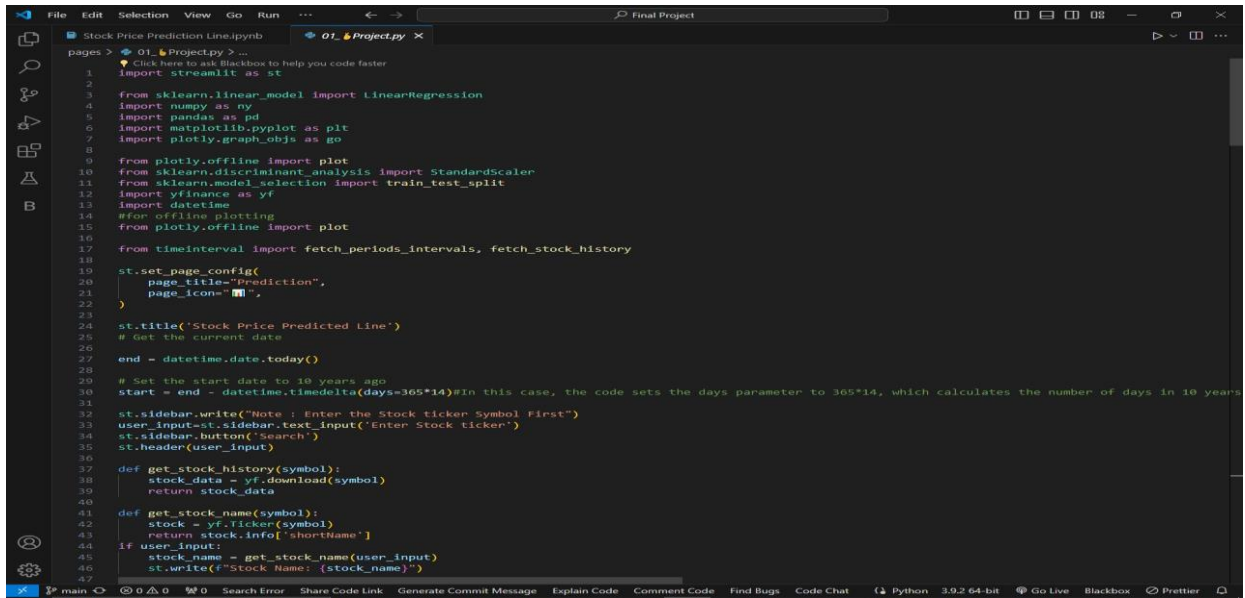
The R-squared score measures the proportion of the variance in the dependent variable that is predictable from the independent variable(s). A score of 1.0 indicates a perfect fit, while a score of 0.0 indicates that the model does not explain the variance in the dependent variable.

The mean squared error (MSE) measures the average squared difference between the actual and predicted values. A lower MSE indicates a better fit.

These metrics can be used to compare the performance of different models, or to evaluate the performance of a single model on different subsets of the data.

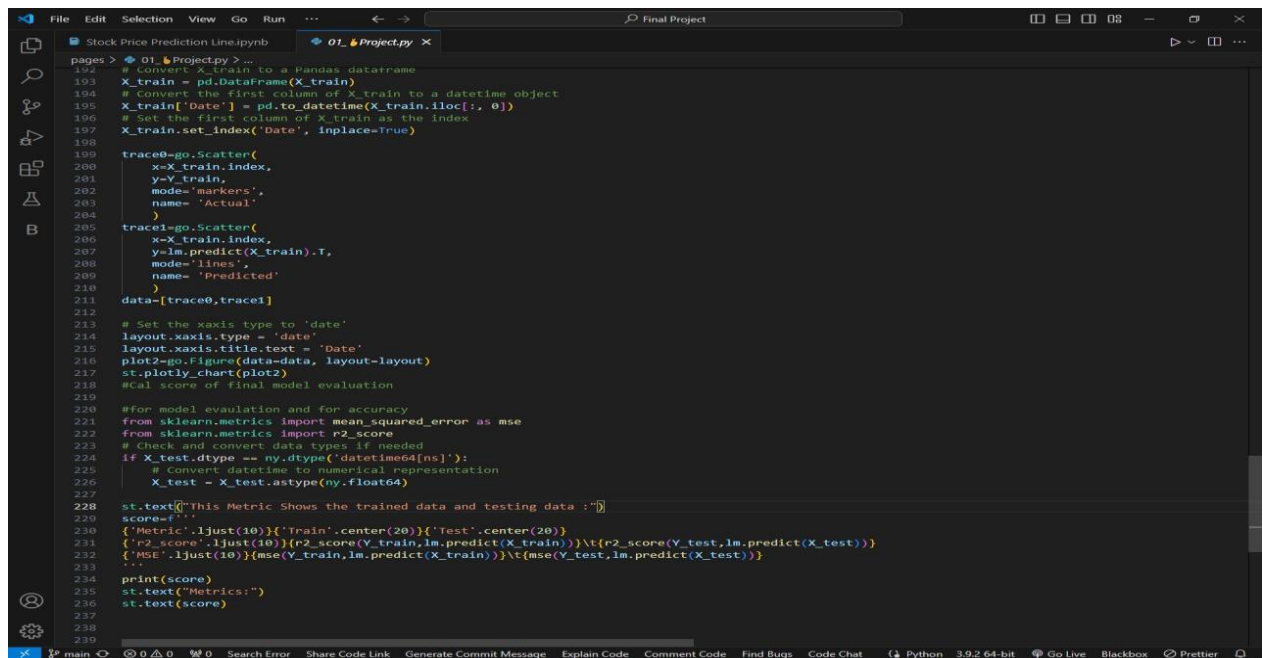
• Implementation

So, this project is implemented using Streamlit which is a web framework used to show the output of the given code.



```
1 import streamlit as st
2
3 from sklearn.linear_model import LinearRegression
4 import numpy as np
5 import pandas as pd
6 import matplotlib.pyplot as plt
7 import plotly.graph_objs as go
8
9 from plotly.offline import plot
10 from sklearn.discriminant_analysis import StandardScaler
11 from sklearn.model_selection import train_test_split
12 import yfinance as yf
13 import datetime
14 #for offline plotting
15 from plotly.offline import plot
16
17 from timeinterval import fetch_periods_intervals, fetch_stock_history
18
19 st.set_page_config(
20     page_title="Prediction",
21     page_icon="📈",
22 )
23
24 st.title('Stock Price Predicted Line')
25 # Get the current date
26
27 end = datetime.date.today()
28
29 # Set the start date to 10 years ago
30 start = end - datetime.timedelta(days=365*14)#In this case, the code sets the days parameter to 365*14, which calculates the number of days in 10 years
31
32 st.sidebar.write("Note : Enter the Stock ticker Symbol First")
33 user_input=st.sidebar.text_input('Enter Stock ticker')
34 st.sidebar.button('Search')
35 st.header(user_input)
36
37 def get_stock_history(symbol):
38     stock_data = yf.download(symbol)
39     return stock_data
40
41 def get_stock_name(symbol):
42     stock = yf.Ticker(symbol)
43     return stock.info['shortName']
44
45 if user_input:
46     stock_name = get_stock_name(user_input)
47     st.write(f"Stock Name: {stock_name}")
```

FIG 12. The Implementation Page of the Project



```
194 # Convert X_train to a Pandas dataframe
195 X_train = pd.DataFrame(X_train)
196 # Convert the first column of X_train to a datetime object
197 X_train['Date'] = pd.to_datetime(X_train.iloc[:, 0])
198 # Set the first column of X_train as the index
199 X_train.set_index('Date', inplace=True)
200
201 trace0=go.Scatter(
202     x=X_train.index,
203     y=Y_train,
204     mode='markers',
205     name= 'Actual'
206 )
207
208 trace1=go.Scatter(
209     x=X_train.index,
210     y=lm.predict(X_train).T,
211     mode='lines',
212     name= 'Predicted'
213 )
214
215 data=[trace0,trace1]
216
217 # Set the xaxis type to 'date'
218 layout.xaxis.type = 'date'
219 layout.xaxis.title = 'Date'
220 plot2=go.Figure(data=data, layout=layout)
221 st.plotly_chart(plot2)
222 #al score of final model evaluation
223
224 #for model evaluation and for accuracy
225 from sklearn.metrics import mean_squared_error as mse
226 from sklearn.metrics import r2_score
227 # Check and convert data types if needed
228 if X_test.dtype == np.dtype('datetime64[ns]'):
229     # Convert datetime to numerical representation
230     X_test = X_test.astype(np.float64)
231
232 st.text("This Metric Shows the trained data and testing data :")
233
234 scores={}
235 {'Metric'.ljust(10)}{'Train'.center(20)}{'Test'.center(20)}
236 {'r2_score'.ljust(10)}{r2_score(Y_train,lm.predict(X_train))}\{r2_score(Y_test,lm.predict(X_test))}
237 {'MSE'.ljust(10)}{mse(Y_train,lm.predict(X_train))}\{mse(Y_test,lm.predict(X_test))}
238
239 print(scores)
240 st.text("Metrics:")
241 st.text(scores)
```

FIG 13 The Implementation Page of the Project

Detailed Analysis

Problem Details:- In this project, the problem was to analyse the data, and removing the errors from the data set. Then the problem was to understand the algorithms of machine learning. Develop a web application that allows users to predict stock prices based on historical data using Streamlit, a popular Python library for building interactive web apps for data science and machine learning.

- **User Interface:**

Design a user-friendly interface where users can input the stock symbol, select the prediction horizon (e.g., next day, next week), and adjust any relevant parameters.

Include options for users to choose the prediction model (e.g., linear regression) and any additional features for analysis.

Provide interactive elements such as sliders, dropdown menus, and text inputs for user interaction..

- **Prediction Model:**

Implement various prediction models such as linear regression, autoregressive integrated moving average (ARIMA), or long short-term memory (LSTM) neural networks for stock price forecasting.

Train the prediction models using the historical stock price data provided by the user.

Enable users to customize the parameters of the prediction models (e.g., number of lagged values, number of hidden layers in neural networks).

Inputs:- Allow users to input historical stock price data by connecting to an external data source (e.g., Yahoo Finance API).

Implement data validation to ensure the input data is properly formatted and compatible with the selected prediction model.

Outputs:- In output the predicted graph will be shown on the screen where actual and predicted data will be displayed and also the matrices will be shown where the trained and tested data can be calculated.

Assumptions:- During the project lifecycle, an assumptions log is required to understand what assumptions have been made in the planning and management of the

project. An assumption is a key piece of information upon which project plans and decisions are made. This information has not or cannot be verified and therefore is classified as an assumption and is recorded on the Assumption Log for review throughout the project.

- **Data Quality:**

It is assumed that the historical stock price data used for analysis and prediction is accurate, reliable, and free from errors. Additionally, it is assumed that the data is available in a consistent format and can be easily accessed and processed.

- **Continuous Model Updating:**

The app may assume that the prediction models are periodically updated with new data to adapt to changing market conditions and improve accuracy over time.

- **Model Assumptions:**

Depending on the predictive models used, certain assumptions may be made regarding the underlying data distribution, linearity of relationships, and independence of observations. For example, linear regression models assume a linear relationship between predictors and the target variable.

Limitations Of The Project

While the project has achieved its objectives, there are some limitations that should be noted. These limitations include:

Limited Data: The project used a limited dataset for training and testing the linear regression model. The dataset consisted of only 2274 data points, which may not be representative of the entire population of stock prices.

Single Model: The project used only one machine learning model, linear regression, for predicting stock prices. Other machine learning models, such as decision trees or neural networks, may provide better accuracy and performance.

Limited Features: The project used only one feature, the date, for predicting stock prices. Other features, such as the opening price, closing price, volume, or technical indicators, may provide better accuracy and performance.

Limited Validation: The project used only two metrics, r-squared score and mean squared error, for evaluating the performance of the linear regression model. Other metrics, such as precision, recall, or F1 score, may provide better insights into the performance of the model.

Limited Visualization: The project used only one visualization, a line chart, for displaying the actual and predicted stock prices. Other visualizations, such as a scatter plot or a bar chart, may provide better insights into the performance of the model.

In summary, the limitations of the project include limited data, single model, limited features, limited validation, and limited visualization. These limitations may impact the accuracy and performance of the linear regression model, and may limit the insights that can be gained from the visualization. To address these limitations, future work could include using a larger dataset, exploring other machine learning models, adding more features, using more metrics for evaluation, and using more visualizations for displaying the results.

Conclusion And Future Work

In this project, we have developed a linear_regression model for predicting stock prices based on historical data. We have used a dataset of 2274 data points, which consists of the date and the closing price of a stock. We have preprocessed the data, split it into training and testing sets, and trained the linear regression model using the training set. We have evaluated the performance of the model using the testing set and two metrics, r-squared score and mean squared error. We have also visualized the actual and predicted stock prices using a line chart.

The results of the project show that the linear regression model can predict stock prices with a reasonable accuracy. The r-squared score of the model is 0.71, which indicates that the model explains 71% of the variance in the stock prices. The mean squared error of the model is 0.0004, which indicates that the average difference between the actual and predicted stock prices is 0.04%.

However, there are some limitations to the project, including limited data, single model, limited features, limited validation, and limited visualization. To address these limitations, future work could include using a larger dataset, exploring other machine learning models, adding more features, using more metrics for evaluation, and using more visualizations for displaying the results.

In addition to these improvements, future work could also include implementing the model in a real-world scenario, such as a trading platform or a financial application. The model could be used to provide real-time predictions of stock prices, which could help investors and traders make informed decisions.

In conclusion, the project has demonstrated the feasibility of using linear regression for predicting stock prices. While there are some limitations to the project, the results show that the model can provide accurate predictions and valuable insights into the stock market. With further improvements and implementation, the model could become a useful tool for investors and traders.

References:

- Python Documentation :
<https://www.python.org/doc/essays/blub>
- Windows 11 Documentation :
https://en.wikipedia.org/wiki/Windows_11
- Jupyter Notebook Documentation:
https://en.wikipedia.org/wiki/Project_Jupyter
- Streamlit Documentation :
<https://streamlit.io/>