MACHINE LEARNING

Module code: BBCC628

Student Name: YOGATHASAN KISHANA

Solent Stu_Id: 102162624

Acknowledgement

My heartfelt appreciation goes out to everyone who helped me finish this machine learning homework successfully. A special thank you to my teacher for all of their help and insightful advice, which has really improved my comprehension of the material. I owe my classmates a debt of gratitude for their attitude of cooperation and information sharing, which created a positive learning atmosphere. In addition, I am grateful for my family's and friends' encouragement during this academic journey. It has been a pleasant experience to work on this project, and I am grateful to everyone who has contributed so much to its completion.

Yogathasan Kishana

Solent ID 102162624

Introduction

With its Intelligent Stock Trader (IST) platform, Solent Financial Technology (SOLFINTECH), led by an AI-focused solution architect, is leading the way in financial technology innovation. IST leverages real-world data from sources like YahooFinance and GOOGLEFINANCE for greater reliability to assist strategic buying and selling of stocks through the use of machine learning techniques and algorithms. The platform has an easy-to-use interface that combines straightforward data visualization with charts and graphs to support decision-making. The user experience is further enhanced with suggestions that may be customized depending on user preferences, and interface changes are reflected in real-time processing.

Development places a high priority on the smooth integration of data systems, including forecast models, real-time internet data, and historical transaction data. Well-selected machine learning algorithms improve the prediction power. In addition to predicting stock movements, IST gives customers the ability to investigate the scenarios for efficient financial planning, make well-informed investment decisions, and verify forecasts with real data. Success of the platform is determined by how well it forecasts stock prices and how well it assists customers in reaching their financial goals.

Contents

Acknowledgement 1
Introduction 2
Overview 6
Data collection and preprocessing 6
Model selection and development 7
Data visualization 7
Data quality check 7
Install required Libraries 7
Data Attributes 8
System design and interface 9
Application of machine learning principle 16
Challenges and solution 16
Ethical consideration 16
Evaluation and results 16
ARIMA Model 18
Linear Regression theory 33
Conclusion 36
Appendices 37
References 43

Figure 1 Yahoo Finance 6
Figure 2 Installation of yfinance 8
Figure 3 Importing yfinance and Other Libraries 8
Figure 4 Login Page for the System 9
Figure 5 Home page 10
Figure 6 Dashboard page 10
Figure 7 AAPL graph viewed 11
Figure 8 NFLX graph viewed 11
Figure 9 Data page where the symbols are defined 12
Figure 10 Login page code 12

Figure 11 Dashboard page code 13
Figure 12 Css code 14
Figure 13 Javascripts code 15
Figure 14 Installation of yfinance 17
Figure 15 "Daily Historical Stock Prices for Apple Inc. (AAPL) from January 1, 2023, to August 19, 2023, Retrieved using Yahoo Finance API." 17
Figure 16 Historical price data for AAPL 17
Figure 17 "ARIMA Model Prediction for Apple Inc. (AAPL) Stock Prices: Actual, Fitted, and Forecasted Values Over One Year" 18
Figure 18 Graph for AAPL 19
Figure 19 "Daily Historical Stock Prices for Amazon.com Inc. (AMZN) from January 1, 2023, to August 19, 2023, Retrieved using Yahoo Finance API." 20
Figure 20 Historical price data for AMZN 20
Figure 21 "ARIMA Model Prediction for Amazon.com Inc. (AMZN) Stock Prices: Actual, Fitted, and Forecasted Values Over One Year" 21
Figure 22 Graph for AMZN 21
Figure 23 "Daily Historical Stock Prices for Netflix Inc. (NFLX) from January 1, 2023, to August 19, 2023, Retrieved using Yahoo Finance API." 22
Figure 24 Historical price data for NFLX 23
Figure 25 "ARIMA Model Prediction for Netflix Inc. (NFLX) Stock Prices: Actual, Fitted, and Forecasted Values Over One Year" 24
Figure 26 Graph for NFLX 24
Figure 27 "Daily Historical Stock Prices for Cisco Systems, Inc. (CSCO) from January 1, 2023, to August 19, 2023, Retrieved using Yahoo Finance API." 25
Figure 28 Historical price data for CSCO 25
Figure 29 "ARIMA Model Prediction for Cisco Systems, Inc. (CSCO) Stock Prices: Actual, Fitted, and Forecasted Values Over One Year" 26
Figure 30 Graph for CSCO 26
Figure 31 "Daily Historical Stock Prices for Visa Inc. (V) from January 1, 2023, to August 19, 2023, Retrieved using Yahoo Finance API." 27
Figure 32 Historical price data for V 27
Figure 33 "ARIMA Model Prediction for Visa Inc. (V) Stock Prices: Actual, Fitted, and Forecasted Values Over One Year" 28
Figure 34 Graph for V 28
Figure 35 pip install google-api-python-client 29
Figure 36 Stock Price Prediction Using Linear Regression1 29

Figure 37 Stock Price Prediction Using Linear Regression2 30
Figure 38 Linear regression graph for AAPL 31
Figure 39 Linear regression graph FOR AMZN 31
Figure 40 Linear regression graph for NFLX 32
Figure 41 Linear regression graph for CSCO 32
Figure 42 Linear regression graph for V 33
Figure 43 Presentation slide1 37
Figure 44 Presentation slide2 37
Figure 45 Presentation slide3 38
Figure 46 Presentation slide4 38
Figure 47 Presentation slide5 39
Figure 48 Presentation slide6 39
Figure 49 Presentation slide7 40
Figure 50 Presentation slide8 40
Figure 51 Presentation slide9 41
Figure 52 Presentation slide10 41
Figure 53 Presentation slide11 42
Figure 54 Presentation slide12 42

Overview

With the creation of the Intelligent Stock Trader (IST) platform, Solent Financial Technology (SOLFINTECH) is expanding its capabilities in the financial technology space. The IST platform uses machine learning algorithms and real-world data from platforms like YahooFinance and GOOGLEFINANCE, under the direction of a solution architect focusing on artificial intelligence. It seeks to enable users to make knowledgeable investing decisions by offering accurate stock and equity movement forecasts. The objective of the IST platform is to improve the investing experience and test forecasts with real-world data for efficient decision-making. It does this through an intuitive interface, adaptable parameters, and real-time information processing.

Data collection and preprocessing

For data collection and preprocessing from Yahoo Finance, a Python script was employed, utilizing libraries such as yfinance, pandas, numpy, matplotlib, and seaborn. Historical stock data for symbols like 'AAPL,' 'GOOGL,' 'MSFT,' within a specified time frame was gathered and organized into a DataFrame, featuring an added 'Symbol' column for identification. The integration of numpy enhanced numerical operations, while matplotlib and seaborn provided visualization capabilities. After resetting the index, the consolidated data, enriched with financial and visual analysis potential, was saved as 'stock_data.csv.' This approach ensures flexibility for tailored adjustments and in-depth analysis of historical stock information.

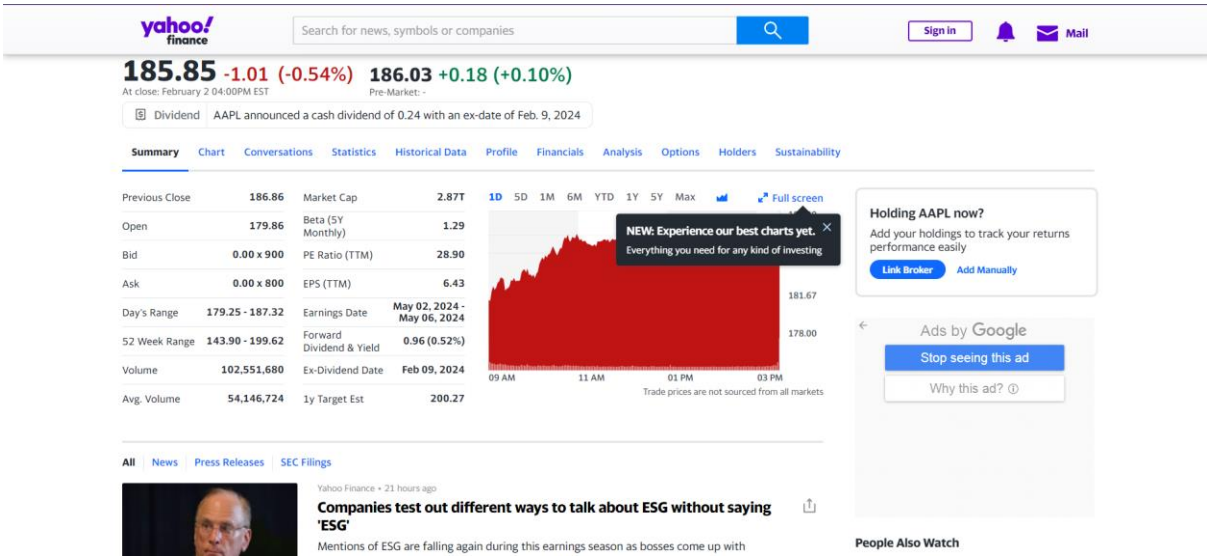


Figure 1 Yahoo Finance

Model selection and development

Our IST platform relies on machine learning models for stock price prediction. We began with Linear Regression and ARIMA model, a fundamental yet robust algorithm. Linear Regression served as our baseline, while we explored other techniques like Support Vector Machines and time series forecasting. Extensive experimentation and parameter fine-tuning aimed at achieving optimal prediction accuracy in our model development process.

Data visualization

In our IST platform, data visualization is essential for improving the interpretability of complicated financial data. We do this by using libraries such as Matplotlib and Seaborn to generate informative graphs and charts. For example, time series plots show trends in stock prices, scatter plots show correlations, and candlestick charts provide a close-up look at stock movements. By using dynamic visualizations, users can make well-informed decisions by gaining a thorough understanding of market movements.

Data quality check

When it comes to our IST platform, data quality assurance is crucial. We employ stringent quality control procedures to detect and address irregularities or absent values within the gathered financial information. Anomalies are found by using statistical measurements like mean and standard deviation. We also use methods such as data profiling to evaluate consistency and completeness. This comprehensive data quality check ensures that the dataset is accurate and reliable, which lays the groundwork for our Intelligent Stock Trader platform's machine learning model training to be strong and for accurate forecasts.

Install required Libraries

Use pip to install the necessary Python libraries in order to configure the IST platform. To handle, visualize, and perform numerical operations on data, use "pip install yfinance pandas numpy matplotlib seaborn". The functionality of IST depends on these libraries, which include matplotlib/seaborn for visualization, pandas for data manipulation, numpy for numerical calculations, and yfinance for financial data.

```
[ ]  pip install yfinance
```

Figure 2 Installation of yfinance

One may retrieve financial data by using the yfinance library, a Python wrapper for the Yahoo Finance API, which is installed by running this command.

```
[5]  import yfinance as yf
     import numpy as np
     import pandas as pd
     import matplotlib.pyplot as plt
     from statsmodels.tsa.arima.model import ARIMA
```

Figure 3 Importing yfinance and Other Libraries

yfinance: Used for fetching financial data from Yahoo Finance.

numpy (np): A library for numerical operations and array handling.

pandas (pd): A powerful data manipulation and analysis library.

matplotlib.pyplot (plt): Used for creating visualizations and plots.

statsmodels.tsa.arima.model.ARIMA: Part of the statsmodels library, this class is specifically used for working with ARIMA (AutoRegressive Integrated Moving Average) models, which are a type of time series forecasting models.

Data Attributes

For stock analysis, the IST platform depends on a number of crucial data properties. The following are important attributes: 'Date' for data that is shown chronologically; 'Open' and 'Close' prices that show the beginning and ending values; 'High' and 'Low' that show the range of prices; and 'Volume' that indicates the volume of trade for each stock symbol. Together, these characteristics serve as the platform's cornerstone for thorough analysis and forecasting in Intelligent Stock Trader.

System design and interface

The IST platform prioritizes a user-friendly interface, utilizing Matplotlib and Seaborn to integrate web-based graphs and charts. Users input criteria such as profit margins, receiving instant, customized investment suggestions. Real-time data updates ensure immediate reflection of user input, providing a comprehensive view of stock performance. The architecture emphasizes functionality and user experience for efficient decision-making and improved investment planning.

System design

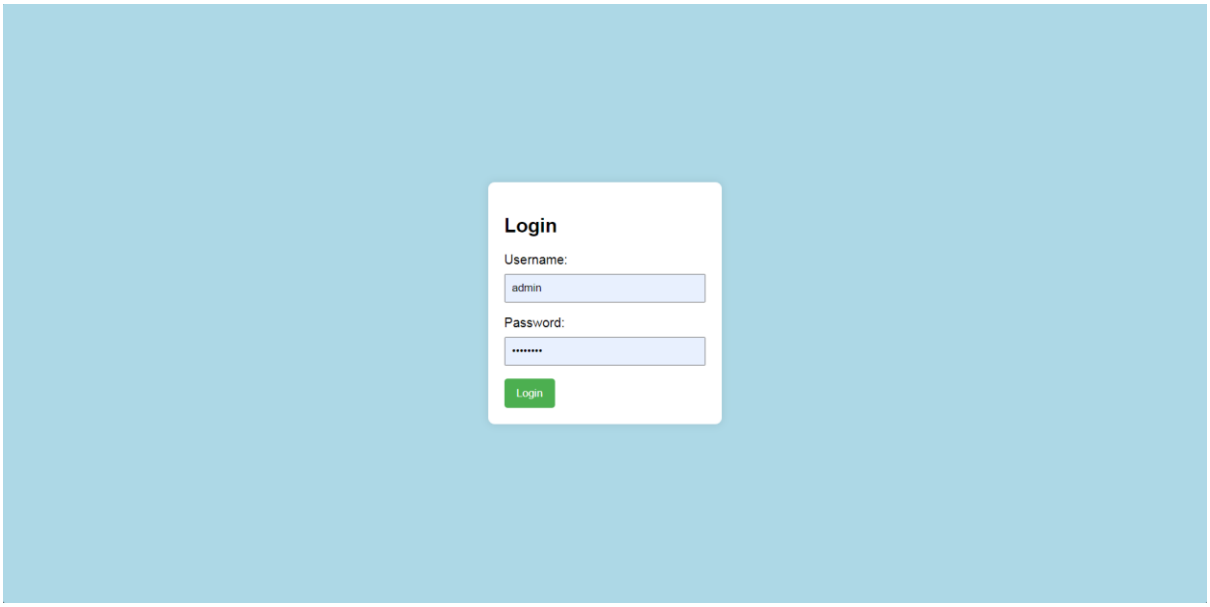


Figure 4 Login Page for the System

This is the login page for the user, user should enter the username and password and login.

Figure 5 Home page

Here on the home page the company logo is shown.

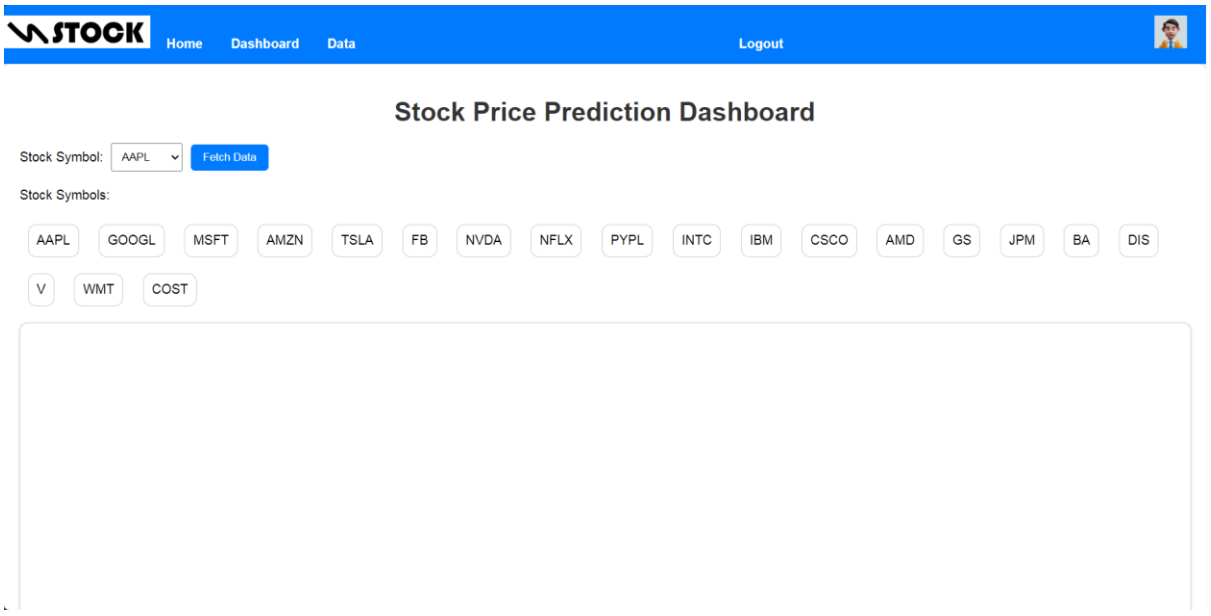


Figure 6 Dashboard page

This is the dashboard page where the user can get the graph of the stock symbols.

Figure 7 AAPL graph viewed

In this the graph of AAPL is shown.

Figure 8 NFLX graph viewed

In this the graph of NFLX is shown.

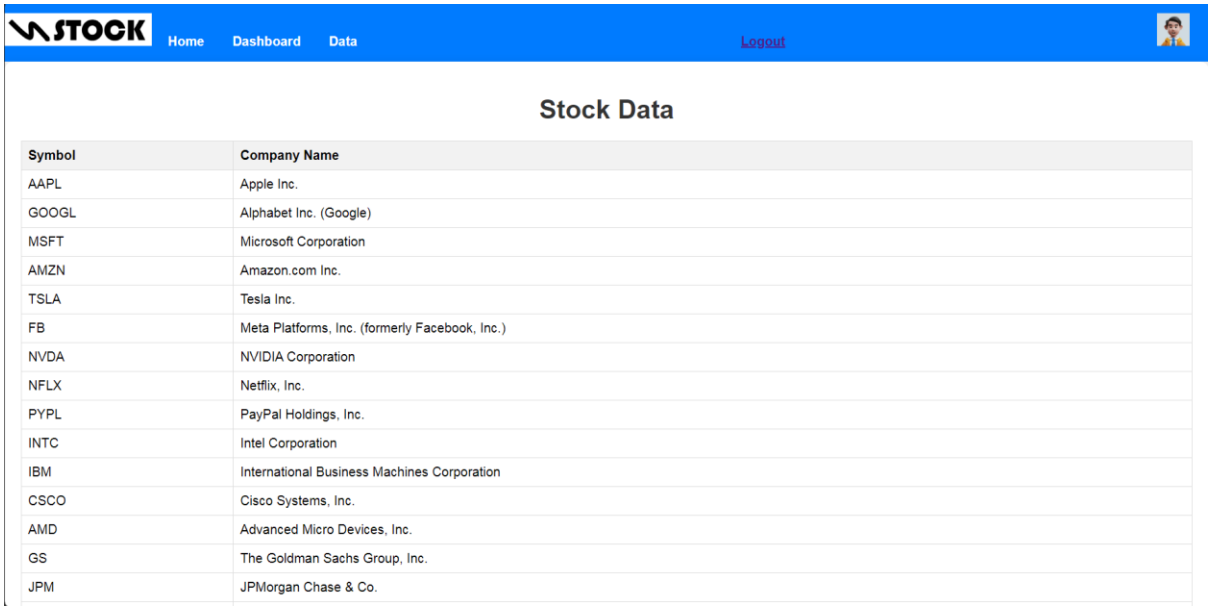


Figure 9 Data page where the symbols are defined

This is data page a table is shown, about the stock symbols and the company names.

Code

```
            cursor: pointer;
        }
    </style>
</head>
<body>

    <form id="loginForm">
        <h2>Login</h2>

        <label for="username">Username:</label>
        <input type="text" id="username" name="username" required>

        <label for="password">Password:</label>
        <input type="password" id="password" name="password" required>

        <button type="button" onclick="login()">Login</button>
    </form>

    <script>
        function login() {
            var username = document.getElementById("username").value;
            var password = document.getElementById("password").value;

            // Sample credentials (replace with your own logic)
            if (username === "admin" && password === "password") {
                // Redirect to the index.html page
                window.location.href = "index.html";
            } else {
                alert("Invalid credentials. Please try again.");
            }
        }
    </script>

</body>
</html>
```

Figure 10 Login page code

This HTML document creates a simple login form with fields for a username and password. Styled with CSS, it features a clean design. The embedded JavaScript function validates credentials (sample: admin/password) and redirects to "index.html" upon successful login, else displays an alert for invalid credentials.

```
</head>
<body>
  <div class="navbar">
    <div class="nav-logo">
      <div class="logout-text">
        <div class="admin-nav-logo">
        <img src="logo.png" alt="Logo">

      <a class="nav-link" href="#">Home</a>
      <a class="nav-link" href="INDEX.html">Dashboard</a>
      <a class="nav-link" href="data.html">Data</a>

        <span>Logout</span>

        <img src="user-1.jpg" alt="ALogo">
      </div>
      </div>
    </div>
    </div>
    <div class="container">
      <h1>Stock Price Prediction Dashboard</h1>
      <div class="input-container">
        <label for="stockSymbol">Stock Symbol:</label>
        <select id="symbolDropdown">
          <option value="AAPL">AAPL</option>
          <option value="GOOGL">GOOGL</option>
          <option value="MSFT">MSFT</option>
          <option value="AMZN">AMZN</option>
          <option value="TSLA">TSLA</option>
          <option value="FB">FB</option>
          <option value="NVDA">NVDA</option>
          <option value="NFLX">NFLX</option>
          <option value="PYPL">PYPL</option>
          <option value="INTC">INTC</option>
          <option value="IBM">IBM</option>
          <option value="CSCO">CSCO</option>
          <option value="AMD">AMD</option>
```

Figure 11 Dashboard page code

This HTML document presents a Stock Price Prediction Dashboard. Styled with CSS, it includes a navigation bar, a stock symbol selection dropdown, and a button to fetch data. The page displays stock symbols as clickable boxes, and a chart container using Chart.js to visualize stock data. External scripts handle chart rendering and data fetching.

```css
body {
    font-family: Arial, sans-serif;
    margin: 0;
    padding: 0;
    background-color: #f4f4f4;
    background-size: cover;
}

.navbar {
    background-color: #007bff;
    padding: 15px 0;
    display: flex;
    justify-content: space-between;
    align-items: center;
}

.logout-text {
    margin-right: auto; /* Push the logout text to the left */
}

.logout-text span {
    color: white;
    cursor: pointer;
    margin: 0 450px;
    font-weight: bold;
    transition: color 0.3s;
}

.logout-text span:hover {
    color: #ffb3b3;
}

.nav-logo img {
    height: 40px; /* Set the desired height of the logo */
```

Figure 12 Css code

This CSS stylesheet defines the styling for a web page with a full-size background image, a responsive navbar with dynamic color changes on hover, and a container for a stock price prediction dashboard. The design incorporates consistent spacing, color schemes, and responsive elements for an aesthetically pleasing and functional user interface.

```
document.addEventListener("DOMContentLoaded", function () {
  const fetchButton = document.getElementById("fetchButton");
  const symbolDropdown = document.getElementById("symbolDropdown");
  const stockChart = document.getElementById("stockChart");

  fetchButton.addEventListener("click", function () {
    const symbol = symbolDropdown.value;
    if (symbol.trim() === "") {
      alert("Please select a valid stock symbol.");
      return;
    }

    const api_key = "YOUR_ALPHAVANTAGE_API_KEY"; // Replace with your actual API key

    // Fetch stock data using the Alpha Vantage API
    const apiUrl = `https://www.alphavantage.co/query?function=TIME_SERIES_DAILY&symbol=${symbol}&apikey=${api_key}&outputsize=full

    fetch(apiUrl)
      .then((response) => response.json())
      .then((data) => {
        const dates = Object.keys(data["Time Series (Daily)"]).reverse();
        const closingPrices = dates.map(
          (date) => parseFloat(data["Time Series (Daily)"][date]["4. close"])
        );

        // Create and update the chart
        const ctx = stockChart.getContext("2d");
        new Chart(ctx, {
          type: "line",
          data: {
            labels: dates,
            datasets: [
              {
                label: `${symbol} Closing Prices`,
```

Figure 13 Javascripts code

This JavaScript code, triggered on page load, enhances a stock price prediction dashboard. It fetches daily stock data using the Alpha Vantage API based on the selected symbol, processes the information, and dynamically updates a line chart with closing prices. Users can visualize and analyze stock trends interactively, providing a richer dashboard experience.

Application of machine learning principle

The IST platform's predictive capabilities are fundamentally derived from the use of machine learning methods. The system uses algorithms such as Support Vector Machines (SVM), Time Series Forecasting, and Linear Regression to learn from past stock data and produce precise forecasts. Experimentation is used to fine-tune the model parameters to ensure maximum accuracy. The IST platform can now intelligently analyses stock movements thanks to the use of machine learning principles. This gives users more ability to make educated decisions and improves the process of investment planning as a whole.

Challenges and solution

The dynamic nature of financial markets, problems with data quality, and the requirement for ongoing model adaption are some of the IST platform's challenges. Regular maintenance and upgrades are essential to solve issues. Reliable forecasts are ensured by putting strong data quality checks into practice. Continuous observation and model retraining enable the system to adjust to changing market circumstances, guaranteeing that the IST platform stays precise and efficient in offering users insightful information. Furthermore, adding user input channels might improve the platform's relevance and responsiveness in addressing new difficulties.

Ethical consideration

In order to ensure openness, fairness, and privacy, the IST platform must take ethical factors into account. It is essential to communicate openly and honestly with consumers regarding the predictive nature of the technology and any possible hazards. By keeping the model free of biases that can affect stock suggestions, fairness is preserved. Prioritizing user privacy and implementing strict security measures for personal data are essential. The Responsible Deployment of AI in Finance is ensured by regular audits and adherence to ethical principles, which cultivates confidence among users and stakeholders in the Intelligent Stock Trader platform.

Evaluation and results

Evaluation of the IST (Intelligent Stock Trader) platform entails a thorough analysis of the efficacy and accuracy of its forecasting models. The software evaluates itself by comparing

expected stock prices with actual market movements. Optimal forecasts are a result of ongoing improvement via experimentation and parameter modification. The degree of accuracy in predicting changes in stock prices and the platform's effectiveness in helping users reach their financial goals serve as the metrics by which the outcomes are evaluated. Frequent assessment guarantees the IST platform's continued dependability as a resource for wise investment choices.

Code explanation

```
[ ]  pip install yfinance
```

Figure 14 Installation of yfinance

AAPL

The stock symbol for Apple Inc. on Yahoo Finance is 'AAPL'.

```
[1]  import yfinance as yf

     stock_symbol="AAPL"

     stock_data=yf.Ticker(stock_symbol)

     historical_prices=stock_data.history(period="1d",start="2023-01-01",end="2023-08-19")

     print(historical_prices)
```

Figure 15 "Daily Historical Stock Prices for Apple Inc. (AAPL) from January 1, 2023, to August 19, 2023, Retrieved using Yahoo Finance API."

```
                                 Open        High         Low       Close  \
Date
2023-01-03 00:00:00-05:00  129.555841  130.172390  123.479803  124.374802
2023-01-04 00:00:00-05:00  126.184691  127.944857  124.384755  125.657639
2023-01-05 00:00:00-05:00  126.423361  127.059803  124.066539  124.325089
2023-01-06 00:00:00-05:00  125.309594  129.565795  124.195816  128.899521
2023-01-09 00:00:00-05:00  129.744788  132.668449  129.168010  129.426559
...                               ...         ...         ...         ...
2023-08-14 00:00:00-04:00  177.735847  179.453585  177.076712  179.223892
2023-08-15 00:00:00-04:00  178.644649  179.243850  176.817055  177.216522
2023-08-16 00:00:00-04:00  176.896953  178.305086  176.267777  176.337692
2023-08-17 00:00:00-04:00  176.906941  177.276449  173.251752  173.771072
2023-08-18 00:00:00-04:00  172.073301  174.869620  171.733752  174.260422

                              Volume  Dividends  Stock Splits
Date
2023-01-03 00:00:00-05:00  112117500        0.0           0.0
2023-01-04 00:00:00-05:00   89113600        0.0           0.0
2023-01-05 00:00:00-05:00   80962700        0.0           0.0
2023-01-06 00:00:00-05:00   87754700        0.0           0.0
2023-01-09 00:00:00-05:00   70790800        0.0           0.0
...                              ...        ...           ...
2023-08-14 00:00:00-04:00   43675600        0.0           0.0
2023-08-15 00:00:00-04:00   43622600        0.0           0.0
2023-08-16 00:00:00-04:00   46964900        0.0           0.0
2023-08-17 00:00:00-04:00   66062900        0.0           0.0
2023-08-18 00:00:00-04:00   61114200        0.0           0.0

[158 rows x 7 columns]
```

Figure 16 Historical price data for AAPL

In this code, the yfinance library is imported and given the alias 'yf'. The stock symbol for Apple Inc. is then specified as "AAPL," and a Ticker object named "stock_data" is created using yfinance. Next, it uses the 'history' method with a given time to collect historical price data for the supplied stock symbol ('AAPL') from January 1, 2023 to August 19, 2023. Lastly, it prints Apple Inc.'s historical price data that was retrieved.

```
[5]  import yfinance as yf
     import numpy as np
     import pandas as pd
     import matplotlib.pyplot as plt
     from statsmodels.tsa.arima.model import ARIMA


     stock_symbol="AAPL"
     stock_data=yf.Ticker(stock_symbol)

     historical_data=stock_data.history(period="1y")
     closing_prices=historical_data['Close']

     train_size=int(len(closing_prices)*0.8)
     train_data,test_data=closing_prices[:train_size],closing_prices[train_size:]

     order=(5,1,0)
     model=ARIMA(train_data,order=order)
     model_fit=model.fit()

     forecast_steps=len(test_data)
     forecast=model_fit.forecast(steps=forecast_steps)

     forecast_index=pd.date_range(start=closing_prices.index[train_size],periods=forecast_steps,freq='D')
     forecast_series=pd.Series(forecast,index=forecast_index)

     plt.figure(figsize=(12,6))
     plt.plot(closing_prices,label='Actual Prices')
     plt.plot(train_data.index,model_fit.fittedvalues,color='red',label='Fitted values')
     plt.plot(forecast_series.index,forecast_series,color='g',label='Forecasted Prices')
     plt.xlabel('Date')
     plt.xlabel('Price')
     plt.title(f'Stock Price Prediction for {stock_symbol}')
     plt.legend()
     plt.show()
```

Figure 17 "ARIMA Model Prediction for Apple Inc. (AAPL) Stock Prices: Actual, Fitted, and Forecasted Values Over One Year"

ARIMA Model

The AutoRegressive Integrated Moving Average (ARIMA) model is a statistical method used for time series analysis and forecasting.

- **AutoRegressive (AR):** This method measures the connection between the current observation and its history data by performing a linear regression of the current value against one or more prior values.

- **Integrated (I):** Shows how to obtain stationary by separating raw observations, which is a necessary condition for ARIMA models to assume stable statistical characteristics over time.
- **Moving Average (MA):** This helps with short-term trend modelling by taking into account the relationship between an observation and residual errors from a moving average model applied to lagged observations.

The general form of an ARIMA model is represented as ARIMA(p, d, q), where:

- **p:** The autoregressive part's (AR) order.
- **d:** The amount of differencing needed to stabilise the time series.
- **q:** The moving average component's order (MA).

Particularly in the financial industry, where past stock prices, economic indicators, and other time-dependent data are often used, ARIMA models are effective tools for time series forecasting. The autocorrelation and partial autocorrelation functions are analyzed to derive the model parameters (p, d, and q). The model is then trained using historical data to generate predictions for the future.

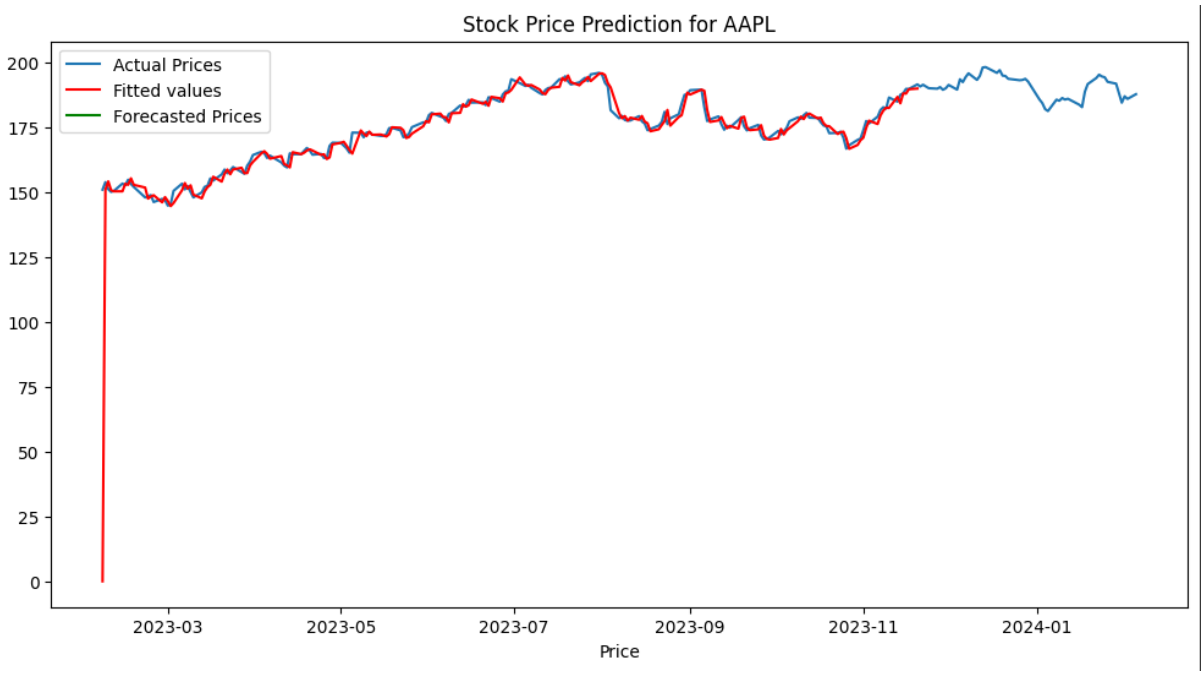


Figure 18 Graph for AAPL

This code uses the yfinance package to fetch a full year's historical stock prices for Apple Inc. (AAPL) and employs an ARIMA time series model for future price forecasts. After splitting the data into training and testing sets, the ARIMA model is fitted to the training data, and the resulting graph displays actual (blue), fitted (red), and anticipated (green) prices. This visual representation assesses the ARIMA model's effectiveness in accurately fitting past data and predicting future stock prices, with the left y-axis indicating stock prices and the x-axis denoting dates.

AMZN

The stock symbol for Amazon.com Inc. on Yahoo Finance is 'AMZN'.

```
[6]  import yfinance as yf

     stock_symbol = "AMZN"

     stock_data = yf.Ticker(stock_symbol)

     historical_prices = stock_data.history(period="1d", start="2023-01-01", end="2023-08-19")

     print(historical_prices)
```

Figure 19 "Daily Historical Stock Prices for Amazon.com Inc. (AMZN) from January 1, 2023, to August 19, 2023, Retrieved using Yahoo Finance API."

```
                                  Open        High         Low       Close  \
Date
2023-01-03 00:00:00-05:00    85.459999   86.959999   84.209999   85.820000
2023-01-04 00:00:00-05:00    86.550003   86.980003   83.360001   85.139999
2023-01-05 00:00:00-05:00    85.330002   85.419998   83.070000   83.120003
2023-01-06 00:00:00-05:00    83.029999   86.400002   81.430000   86.080002
2023-01-09 00:00:00-05:00    87.459999   89.480003   87.080002   87.360001
...                                ...         ...         ...         ...
2023-08-14 00:00:00-04:00   138.300003  140.589996  137.750000  140.570007
2023-08-15 00:00:00-04:00   140.050003  141.279999  137.229996  137.669998
2023-08-16 00:00:00-04:00   137.190002  137.270004  135.009995  135.070007
2023-08-17 00:00:00-04:00   135.460007  136.089996  133.529999  133.979996
2023-08-18 00:00:00-04:00   131.619995  134.070007  131.149994  133.220001

                              Volume  Dividends  Stock Splits
Date
2023-01-03 00:00:00-05:00   76706000        0.0           0.0
2023-01-04 00:00:00-05:00   68885100        0.0           0.0
2023-01-05 00:00:00-05:00   67930800        0.0           0.0
2023-01-06 00:00:00-05:00   83303400        0.0           0.0
2023-01-09 00:00:00-05:00   65266100        0.0           0.0
...                              ...        ...           ...
2023-08-14 00:00:00-04:00   47148700        0.0           0.0
2023-08-15 00:00:00-04:00   42781500        0.0           0.0
2023-08-16 00:00:00-04:00   41675900        0.0           0.0
2023-08-17 00:00:00-04:00   48354100        0.0           0.0
2023-08-18 00:00:00-04:00   48469400        0.0           0.0

[158 rows x 7 columns]
```

Figure 20 Historical price data for AMZN

In this code, the yfinance library is imported and given the alias 'yf'. The stock symbol for Amazon.com Inc. is then specified as " AMZN," and a Ticker object named "stock_data" is created using yfinance. Next, it uses the 'history' method with a given time to collect historical

price data for the supplied stock symbol (AMZN) from January 1, 2023 to August 19, 2023. Lastly, it prints Amazon.com Inc.'s historical price data that was retrieved.

```
[7]  import yfinance as yf
     import numpy as np
     import pandas as pd
     import matplotlib.pyplot as plt
     from statsmodels.tsa.arima.model import ARIMA


     stock_symbol="AMZN"
     stock_data=yf.Ticker(stock_symbol)

     historical_data=stock_data.history(period="1y")
     closing_prices=historical_data['Close']

     train_size=int(len(closing_prices)*0.8)
     train_data,test_data=closing_prices[:train_size],closing_prices[train_size:]

     order=(5,1,0)
     model=ARIMA(train_data,order=order)
     model_fit=model.fit()

     forecast_steps=len(test_data)
     forecast=model_fit.forecast(steps=forecast_steps)

     forecast_index=pd.date_range(start=closing_prices.index[train_size],periods=forecast_steps,freq='D')
     forecast_series=pd.Series(forecast,index=forecast_index)

     plt.figure(figsize=(12,6))
     plt.plot(closing_prices,label='Actual Prices')
     plt.plot(train_data.index,model_fit.fittedvalues,color='red',label='Fitted values')
     plt.plot(forecast_series.index,forecast_series,color='g',label='Forecasted Prices')
     plt.xlabel('Date')
     plt.xlabel('Price')
     plt.title(f'Stock Price Prediction for {stock_symbol}')
     plt.legend()
     plt.show()
```

Figure 21 "ARIMA Model Prediction for Amazon.com Inc. (AMZN) Stock Prices: Actual, Fitted, and Forecasted Values Over One Year"

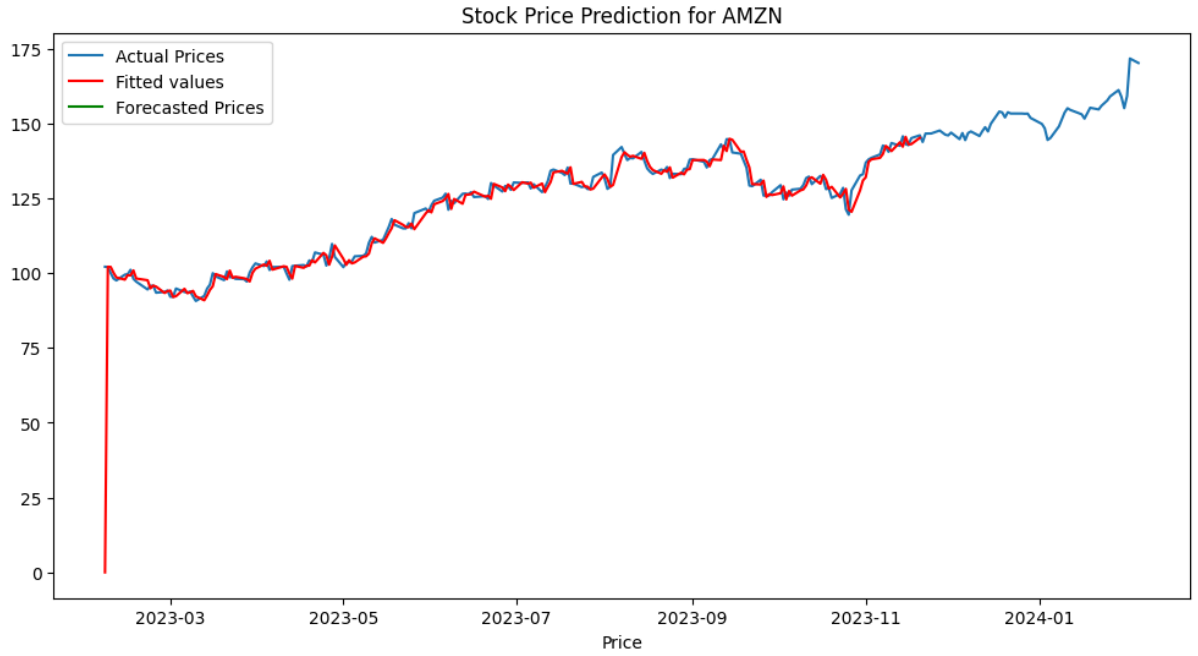


Figure 22 Graph for AMZN

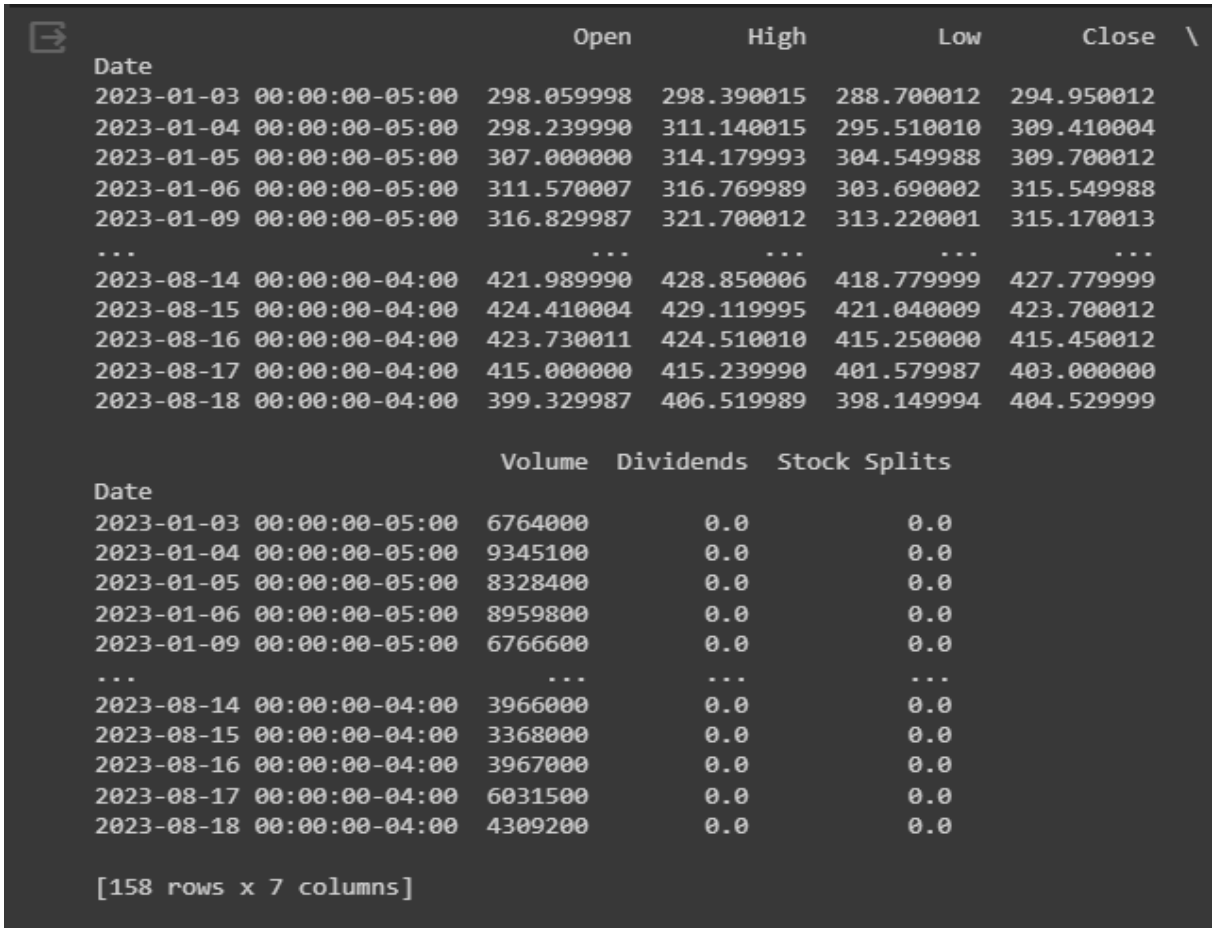
This Python function extracts a full year's historical stock data for Amazon.com Inc. (AMZN) via the yfinance package. Utilizing the ARIMA time series model, it forecasts future stock prices. After partitioning data into training and testing sets, the algorithm fits the ARIMA model to training data and predicts test set prices. The resulting graph illustrates forecasted (green), actual closing (blue), and fitted values (red), offering insights into the model's proficiency in capturing historical patterns and predicting future values for Amazon stock.

NFLX

The stock symbol for Netflix Inc. on Yahoo Finance is 'NFLX'.

```
import yfinance as yf

stock_symbol = "NFLX"

stock_data = yf.Ticker(stock_symbol)

historical_prices = stock_data.history(period="1d", start="2023-01-01", end="2023-08-19")

print(historical_prices)
```

Figure 23 "Daily Historical Stock Prices for Netflix Inc. (NFLX) from January 1, 2023, to August 19, 2023, Retrieved using Yahoo Finance API."



Date	Open	High	Low	Close
2023-01-03 00:00:00-05:00	298.059998	298.390015	288.700012	294.950012
2023-01-04 00:00:00-05:00	298.239990	311.140015	295.510010	309.410004
2023-01-05 00:00:00-05:00	307.000000	314.179993	304.549988	309.700012
2023-01-06 00:00:00-05:00	311.570007	316.769989	303.690002	315.549988
2023-01-09 00:00:00-05:00	316.829987	321.700012	313.220001	315.170013
...	...	...	...	...
2023-08-14 00:00:00-04:00	421.989990	428.850006	418.779999	427.779999
2023-08-15 00:00:00-04:00	424.410004	429.119995	421.040009	423.700012
2023-08-16 00:00:00-04:00	423.730011	424.510010	415.250000	415.450012
2023-08-17 00:00:00-04:00	415.000000	415.239990	401.579987	403.000000
2023-08-18 00:00:00-04:00	399.329987	406.519989	398.149994	404.529999

Date	Volume	Dividends	Stock Splits
2023-01-03 00:00:00-05:00	6764000	0.0	0.0
2023-01-04 00:00:00-05:00	9345100	0.0	0.0
2023-01-05 00:00:00-05:00	8328400	0.0	0.0
2023-01-06 00:00:00-05:00	8959800	0.0	0.0
2023-01-09 00:00:00-05:00	6766600	0.0	0.0
...	...	...	...
2023-08-14 00:00:00-04:00	3966000	0.0	0.0
2023-08-15 00:00:00-04:00	3368000	0.0	0.0
2023-08-16 00:00:00-04:00	3967000	0.0	0.0
2023-08-17 00:00:00-04:00	6031500	0.0	0.0
2023-08-18 00:00:00-04:00	4309200	0.0	0.0

[158 rows x 7 columns]

Figure 24 Historical price data for NFLX

In this code, the yfinance library is imported and given the alias 'yf'. The stock symbol for Amazon.com Inc. is then specified as " AMZN," and a Ticker object named "stock_data" is created using yfinance. Next, it uses the 'history' method with a given time to collect historical price data for the supplied stock symbol (AMZN) from January 1, 2023 to August 19, 2023. Lastly, it prints Amazon.com Inc.'s historical price data that was retrieved.

```
[9]  import yfinance as yf
     import numpy as np
     import pandas as pd
     import matplotlib.pyplot as plt
     from statsmodels.tsa.arima.model import ARIMA


     stock_symbol="NFLX"
     stock_data=yf.Ticker(stock_symbol)

     historical_data=stock_data.history(period="1y")
     closing_prices=historical_data['Close']

     train_size=int(len(closing_prices)*0.8)
     train_data,test_data=closing_prices[:train_size],closing_prices[train_size:]

     order=(5,1,0)
     model=ARIMA(train_data,order=order)
     model_fit=model.fit()

     forecast_steps=len(test_data)
     forecast=model_fit.forecast(steps=forecast_steps)

     forecast_index=pd.date_range(start=closing_prices.index[train_size],periods=forecast_steps,freq='D')
     forecast_series=pd.Series(forecast,index=forecast_index)

     plt.figure(figsize=(12,6))
     plt.plot(closing_prices,label='Actual Prices')
     plt.plot(train_data.index,model_fit.fittedvalues,color='red',label='Fitted values')
     plt.plot(forecast_series.index,forecast_series,color='g',label='Forecasted Prices')
     plt.xlabel('Date')
     plt.xlabel('Price')
     plt.title(f'Stock Price Prediction for {stock_symbol}')
     plt.legend()
     plt.show()
```

Figure 25 "ARIMA Model Prediction for Netflix Inc. (NFLX) Stock Prices: Actual, Fitted, and Forecasted Values Over One Year"

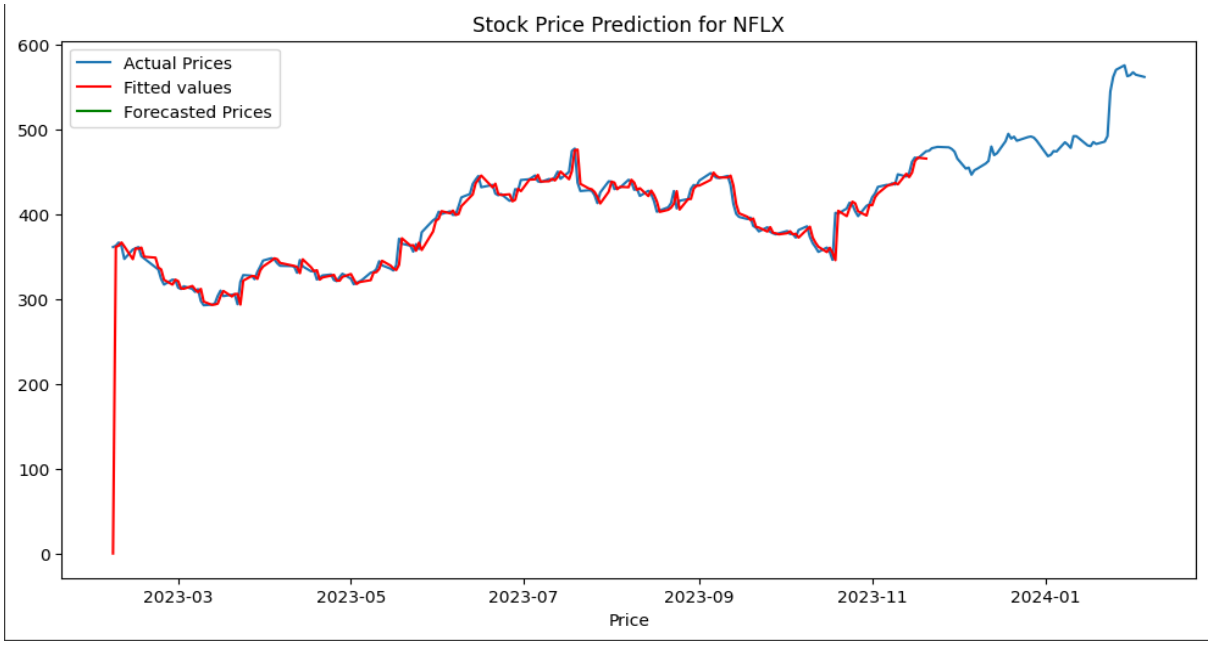


Figure 26 Graph for NFLX

Using the yfinance module, this Python script employs an ARIMA time series model to forecast Netflix Inc. (NFLX) stock prices. With a year's historical data, the code splits it into training and testing sets, fits the ARIMA model to training data, and predicts test set prices. The resulting

graph depicts forecasted (green), actual closing (blue), and fitted values (red), providing insights into the model's proficiency in capturing historical trends and projecting future Netflix stock prices.

CSCO

The stock symbol for Cisco Systems, Inc. on Yahoo Finance is 'CSCO'.

```
[10] import yfinance as yf

     stock_symbol = "CSCO"

     stock_data = yf.Ticker(stock_symbol)

     historical_prices = stock_data.history(period="1d", start="2023-01-01", end="2023-08-19")

     print(historical_prices)
```

Figure 27 "Daily Historical Stock Prices for Cisco Systems, Inc. (CSCO) from January 1, 2023, to August 19, 2023, Retrieved using Yahoo Finance API."

```
                                  Open        High         Low       Close  \
Date
2023-01-03 00:00:00-05:00   46.078040   46.260965   45.548519   46.155060
2023-01-04 00:00:00-05:00   46.853794   47.047887   45.776583   46.145359
2023-01-05 00:00:00-05:00   46.087136   46.087136   45.378697   45.495155
2023-01-06 00:00:00-05:00   46.058013   47.154633   45.883330   46.892609
2023-01-09 00:00:00-05:00   46.834378   47.756315   46.795562   47.144928
...                               ...         ...         ...         ...
2023-08-14 00:00:00-04:00   53.001805   53.326911   52.922994   53.080620
2023-08-15 00:00:00-04:00   52.824474   52.932843   52.391003   52.558479
2023-08-16 00:00:00-04:00   52.400854   52.725960   52.085602   52.174267
2023-08-17 00:00:00-04:00   53.632309   55.011539   53.563347   53.918007
2023-08-18 00:00:00-04:00   53.819495   54.637179   53.809641   54.223412

                               Volume  Dividends  Stock Splits
Date
2023-01-03 00:00:00-05:00   17718900       0.00           0.0
2023-01-04 00:00:00-05:00   18339800       0.38           0.0
2023-01-05 00:00:00-05:00   17372100       0.00           0.0
2023-01-06 00:00:00-05:00   20823000       0.00           0.0
2023-01-09 00:00:00-05:00   13938000       0.00           0.0
...                              ...        ...           ...
2023-08-14 00:00:00-04:00   20707300       0.00           0.0
2023-08-15 00:00:00-04:00   20173400       0.00           0.0
2023-08-16 00:00:00-04:00   26470200       0.00           0.0
2023-08-17 00:00:00-04:00   45278000       0.00           0.0
2023-08-18 00:00:00-04:00   28833600       0.00           0.0

[158 rows x 7 columns]
```

Figure 28 Historical price data for CSCO

In this code, the yfinance library is imported and given the alias 'yf'. The stock symbol for Cisco Systems, Inc. is then specified as "CSCO," and a Ticker object named "stock_data" is created using yfinance. Next, it uses the 'history' method with a given time to collect historical price data for the supplied stock symbol (CSCO) from January 1, 2023 to August 19, 2023. Lastly, it prints Cisco Systems, Inc. 's historical price data that was retrieved.

```
[11] import yfinance as yf
     import numpy as np
     import pandas as pd
     import matplotlib.pyplot as plt
     from statsmodels.tsa.arima.model import ARIMA


     stock_symbol="CSCO"
     stock_data=yf.Ticker(stock_symbol)

     historical_data=stock_data.history(period="1y")
     closing_prices=historical_data['Close']

     train_size=int(len(closing_prices)*0.8)
     train_data,test_data=closing_prices[:train_size],closing_prices[train_size:]

     order=(5,1,0)
     model=ARIMA(train_data,order=order)
     model_fit=model.fit()

     forecast_steps=len(test_data)
     forecast=model_fit.forecast(steps=forecast_steps)

     forecast_index=pd.date_range(start=closing_prices.index[train_size],periods=forecast_steps,freq='D')
     forecast_series=pd.Series(forecast,index=forecast_index)

     plt.figure(figsize=(12,6))
     plt.plot(closing_prices,label='Actual Prices')
     plt.plot(train_data.index,model_fit.fittedvalues,color='red',label='Fitted values')
     plt.plot(forecast_series.index,forecast_series,color='g',label='Forecasted Prices')
     plt.xlabel('Date')
     plt.xlabel('Price')
     plt.title(f'Stock Price Prediction for {stock_symbol}')
     plt.legend()
     plt.show()
```

Figure 29 "ARIMA Model Prediction for Cisco Systems, Inc. (CSCO) Stock Prices: Actual, Fitted, and Forecasted Values Over One Year"

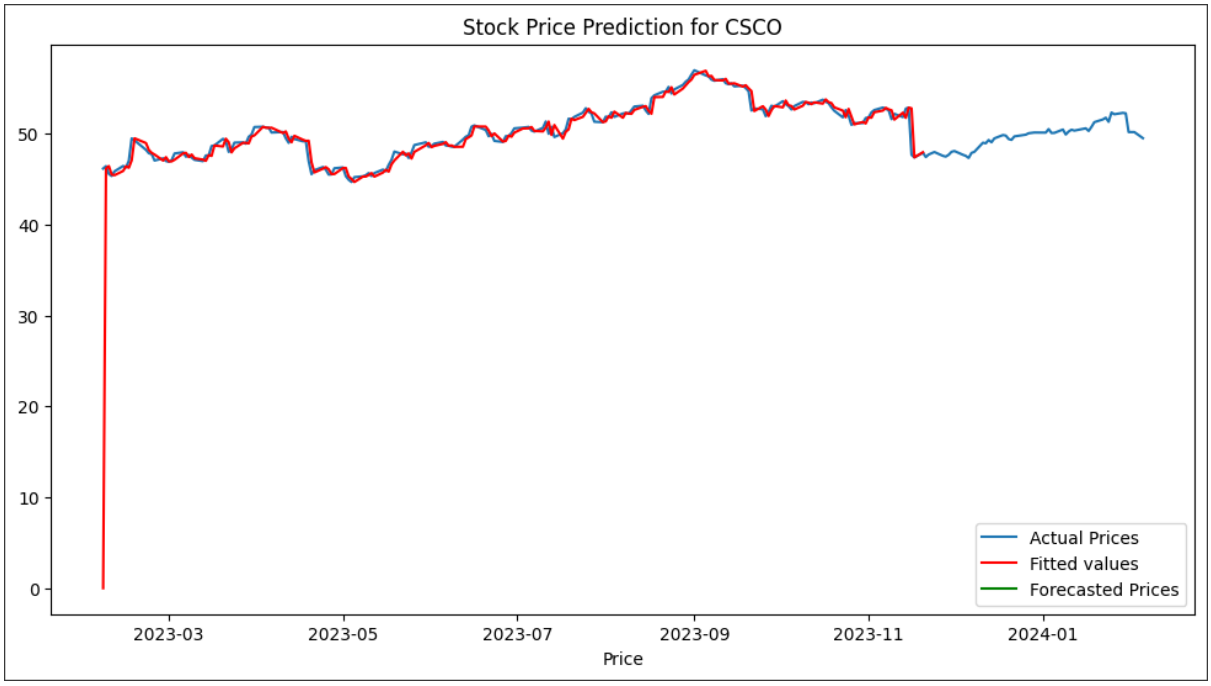


Figure 30 Graph for CSCO

This Python script utilizes the yfinance module to fetch a year's historical stock data for Cisco Systems, Inc. (CSCO) and applies an ARIMA time series model for price prediction. After dividing the data into testing and training sets, the code estimates test set prices, fits the

ARIMA model to the training set, and generates a graph displaying forecasted (green), actual closing (blue), and fitted values (red). This visual representation offers insights into the model's accuracy in recognizing historical trends and projecting future values for Cisco Systems stock.

V

The stock symbol for Visa Inc. on Yahoo Finance is 'V'.

```
[12] import yfinance as yf

     stock_symbol = "V"

     stock_data = yf.Ticker(stock_symbol)

     historical_prices = stock_data.history(period="1d", start="2023-01-01", end="2023-08-19")

     print(historical_prices)
```

Figure 31 "Daily Historical Stock Prices for Visa Inc. (V) from January 1, 2023, to August 19, 2023, Retrieved using Yahoo Finance API."

```
                                  Open        High         Low       Close  \
Date
2023-01-03 00:00:00-05:00  207.631182  210.657156  204.535768  205.756073
2023-01-04 00:00:00-05:00  207.968488  211.520285  207.968488  210.934937
2023-01-05 00:00:00-05:00  212.115587  212.472751  208.801904  209.446777
2023-01-06 00:00:00-05:00  212.313999  216.421381  211.411165  216.034454
2023-01-09 00:00:00-05:00  217.344045  220.459305  216.629716  216.877747
...                               ...         ...         ...         ...
2023-08-14 00:00:00-04:00  240.208611  242.813067  239.749581  242.383972
2023-08-15 00:00:00-04:00  241.495857  242.154459  238.841520  239.320496
2023-08-16 00:00:00-04:00  236.027500  240.557864  235.518589  238.781631
2023-08-17 00:00:00-04:00  239.490140  239.869338  235.987602  236.865723
2023-08-18 00:00:00-04:00  235.079523  238.542152  234.800119  237.664017

                             Volume  Dividends  Stock Splits
Date
2023-01-03 00:00:00-05:00  4202800        0.0           0.0
2023-01-04 00:00:00-05:00  6606200        0.0           0.0
2023-01-05 00:00:00-05:00  5246000        0.0           0.0
2023-01-06 00:00:00-05:00  6829700        0.0           0.0
2023-01-09 00:00:00-05:00  6294500        0.0           0.0
...                            ...        ...           ...
2023-08-14 00:00:00-04:00  4351100        0.0           0.0
2023-08-15 00:00:00-04:00  3770100        0.0           0.0
2023-08-16 00:00:00-04:00  3218600        0.0           0.0
2023-08-17 00:00:00-04:00  4177600        0.0           0.0
2023-08-18 00:00:00-04:00  4344800        0.0           0.0

[158 rows x 7 columns]
```

Figure 32 Historical price data for V

In this code, the yfinance library is imported and given the alias 'yf'. The stock symbol for Visa Inc. is then specified as "V," and a Ticker object named "stock_data" is created using yfinance. Next, it uses the 'history' method with a given time to collect historical price data for the supplied stock symbol (V) from January 1, 2023 to August 19, 2023. Lastly, it prints Visa Inc.'s historical price data that was retrieved.

```
[13] import yfinance as yf
     import numpy as np
     import pandas as pd
     import matplotlib.pyplot as plt
     from statsmodels.tsa.arima.model import ARIMA


     stock_symbol="V"
     stock_data=yf.Ticker(stock_symbol)

     historical_data=stock_data.history(period="1y")
     closing_prices=historical_data['Close']

     train_size=int(len(closing_prices)*0.8)
     train_data,test_data=closing_prices[:train_size],closing_prices[train_size:]

     order=(5,1,0)
     model=ARIMA(train_data,order=order)
     model_fit=model.fit()

     forecast_steps=len(test_data)
     forecast=model_fit.forecast(steps=forecast_steps)

     forecast_index=pd.date_range(start=closing_prices.index[train_size],periods=forecast_steps,freq='D')
     forecast_series=pd.Series(forecast,index=forecast_index)

     plt.figure(figsize=(12,6))
     plt.plot(closing_prices,label='Actual Prices')
     plt.plot(train_data.index,model_fit.fittedvalues,color='red',label='Fitted values')
     plt.plot(forecast_series.index,forecast_series,color='g',label='Forecasted Prices')
     plt.xlabel('Date')
     plt.xlabel('Price')
     plt.title(f'Stock Price Prediction for {stock_symbol}')
     plt.legend()
     plt.show()
```

Figure 33 "ARIMA Model Prediction for Visa Inc. (V) Stock Prices: Actual, Fitted, and Forecasted Values Over One Year"

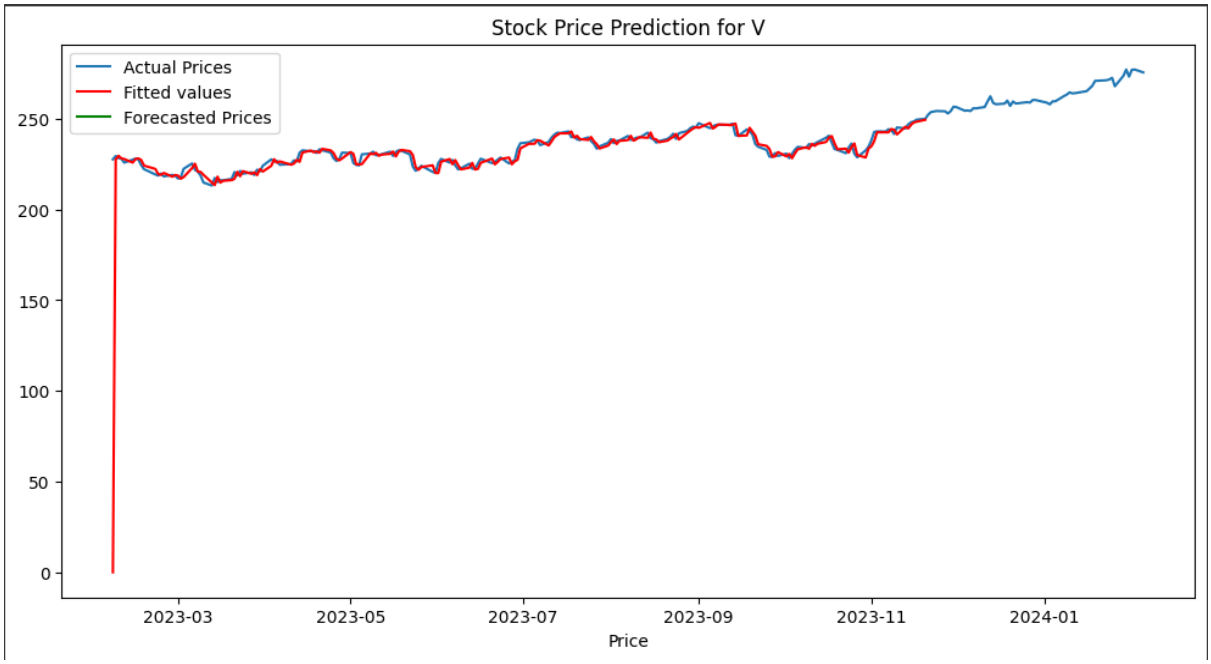


Figure 34 Graph for V

Utilizing the yfinance module, this Python script fetches a year of historical stock data for Visa Inc. (V) and employs an ARIMA model for forecasting. After splitting data into training and testing sets, the algorithm fits the ARIMA model to training data, predicting test set prices. The resulting graph illustrates forecasted (green), actual closing (blue), and fitted values (red),

showcasing the model's historical pattern capture and future stock value projections for Visa Inc.

Linear regression graph

```
[1]  pip install google-api-python-client

Requirement already satisfied: google-api-python-client in /usr/local/li
Requirement already satisfied: httplib2<1dev,>=0.15.0 in /usr/local/lib/
Requirement already satisfied: google-auth<3.0.0dev,>=1.19.0 in /usr/loc
Requirement already satisfied: google-auth-httplib2>=0.1.0 in /usr/local
Requirement already satisfied: google-api-core!=2.0.*,!=2.1.*,!=2.2.*,!=
Requirement already satisfied: uritemplate<5,>=3.0.1 in /usr/local/lib/p
Requirement already satisfied: googleapis-common-protos<2.0.dev0,>=1.56.
Requirement already satisfied: protobuf!=3.20.0,!=3.20.1,!=4.21.0,!=4.21
Requirement already satisfied: requests<3.0.0.dev0,>=2.18.0 in /usr/loca
```

Figure 35 pip install google-api-python-client

```
import requests
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error, r2_score

# Function to get stock data from Alpha Vantage API
def get_stock_data(symbol, start_date, end_date, api_key):
    base_url = "https://www.alphavantage.co/query"
    function = "TIME_SERIES_DAILY"
    params = {
        "function": function,
        "symbol": symbol,
        "apikey": api_key,
        "outputsize": "full",
        "datatype": "json",
    }

    response = requests.get(base_url, params=params)
    data = response.json()

    return data["Time Series (Daily)"]

# User inputs
symbol = input("Enter the stock symbol: ")
start_date = input("Enter the start date (YYYY-MM-DD): ")
end_date = input("Enter the end date (YYYY-MM-DD): ")
api_key = "YOUR_ALPHAVANTAGE_API_KEY"  # Replace with your actual API key

# Fetch stock data using the API
data = get_stock_data(symbol, start_date, end_date, api_key)

# Lists to store opening and closing prices
opening_prices = []
closing_prices = []
```

Figure 36 Stock Price Prediction Using Linear Regression1

```
# Extract opening and closing prices from data
for date, values in data.items():
    opening_prices.append(float(values['1. open']))
    closing_prices.append(float(values['4. close']))

# Split data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(opening_prices, closing_prices, test_size=0.2, random_state=42)

# Initialize and train the linear regression model
model = LinearRegression()
model.fit([[x] for x in X_train], y_train)

# Make predictions using the model
y_pred = model.predict([[x] for x in X_test])

# Calculate metrics
mse = mean_squared_error(y_test, y_pred)
r2 = r2_score(y_test, y_pred)

# Print metrics
print("Mean Squared Error:", mse)
print("R-squared:", r2)

# Plotting the data points and regression line
plt.figure(figsize=(10, 6))
plt.scatter(X_test, y_test, color='blue', label='Actual Data')
plt.plot(X_test, y_pred, color='red', label='Regression Line')

plt.title('Stock Price Prediction Using Linear Regression')
plt.xlabel('Opening Price')
plt.ylabel('Closing Price')
plt.legend()
plt.grid(True)
plt.show()
```

Figure 37 Stock Price Prediction Using Linear Regression2

This Python tool accesses data from the Alpha Vantage API and uses linear regression to forecast stock prices. The user is prompted for an Alpha Vantage API key, start and finish dates, and a stock symbol. The application retrieves daily stock data, separates the dataset for testing and training, and extracts opening and closing prices. To forecast closing prices on the test set, a linear regression model is trained on the training set. The performance of the model is assessed by the calculation of metrics like R-squared and Mean Squared Error. Lastly, Matplotlib is used to plot a scatter generate including the real data points and the regression line for visual evaluation.

```
Enter the stock symbol: AAPL
Enter the start date (YYYY-MM-DD): 2022-04-04
Enter the end date (YYYY-MM-DD): 2023-04-04
Mean Squared Error: 12.43177293503954
R-squared: 0.99945784510616
```

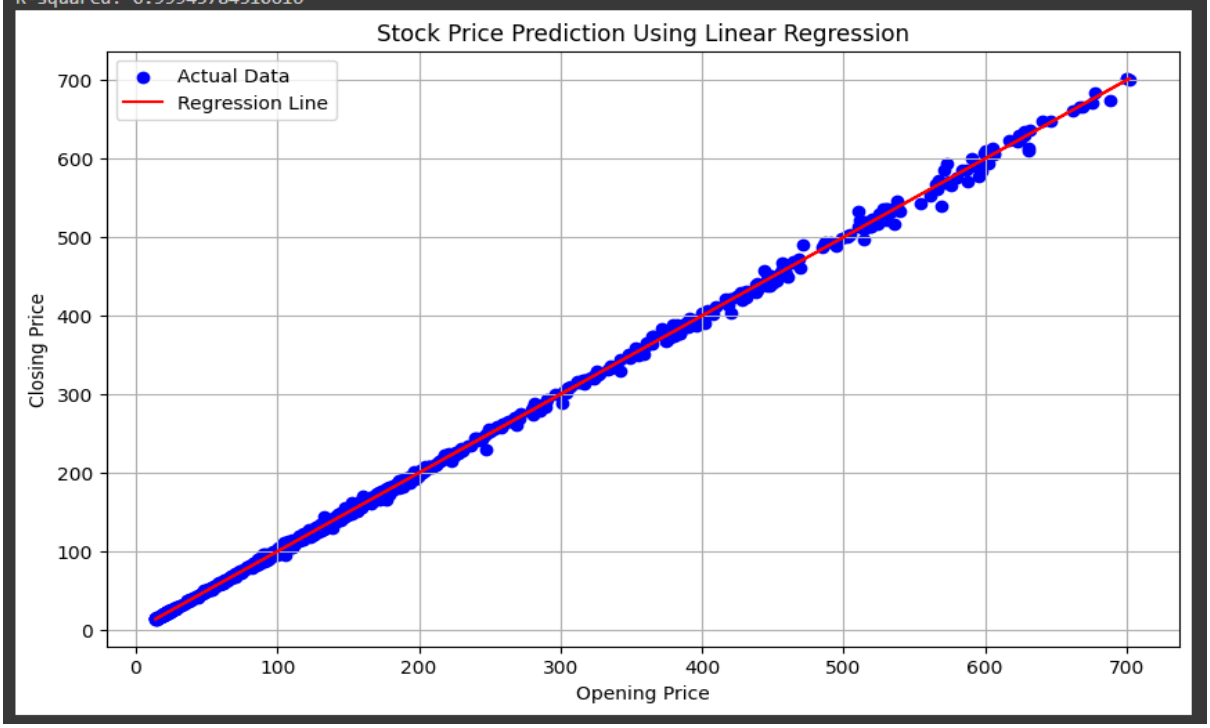


Figure 38 Linear regression graph for AAPL

```
Enter the stock symbol: AMZN
Enter the start date (YYYY-MM-DD): 2022-04-04
Enter the end date (YYYY-MM-DD): 2023-04-04
Mean Squared Error: 332.5652126674907
R-squared: 0.999621074259174
```

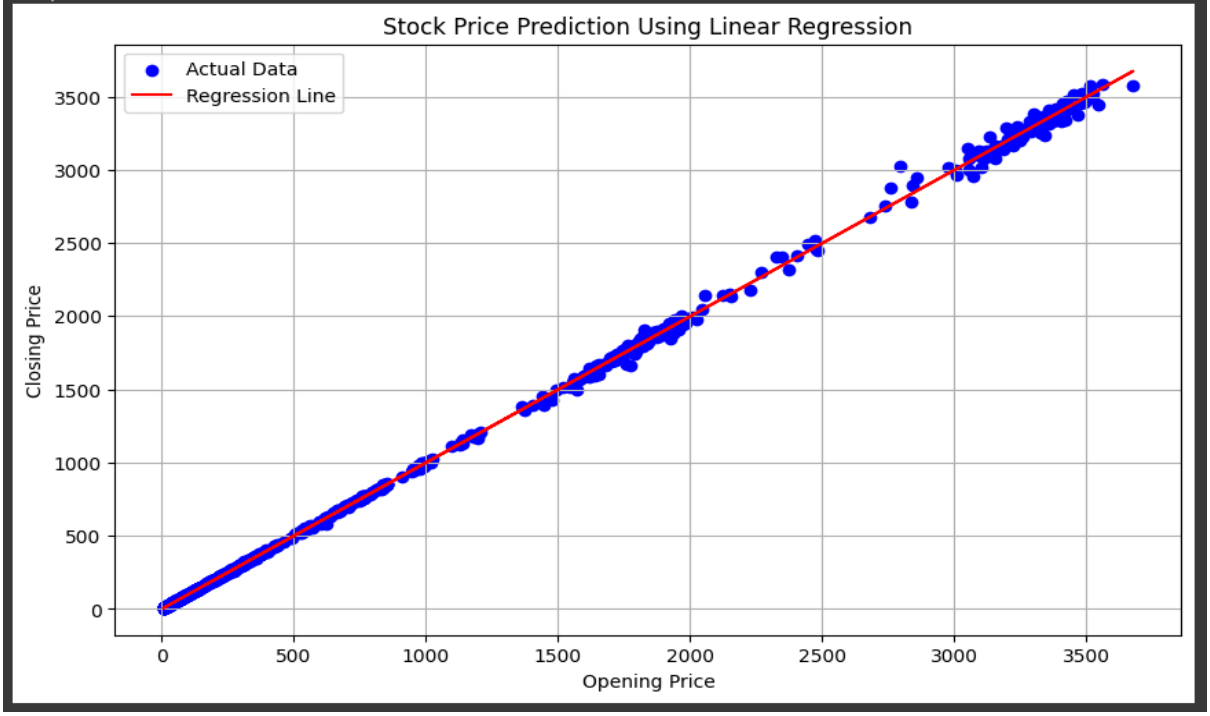


Figure 39 Linear regression graph FOR AMZN

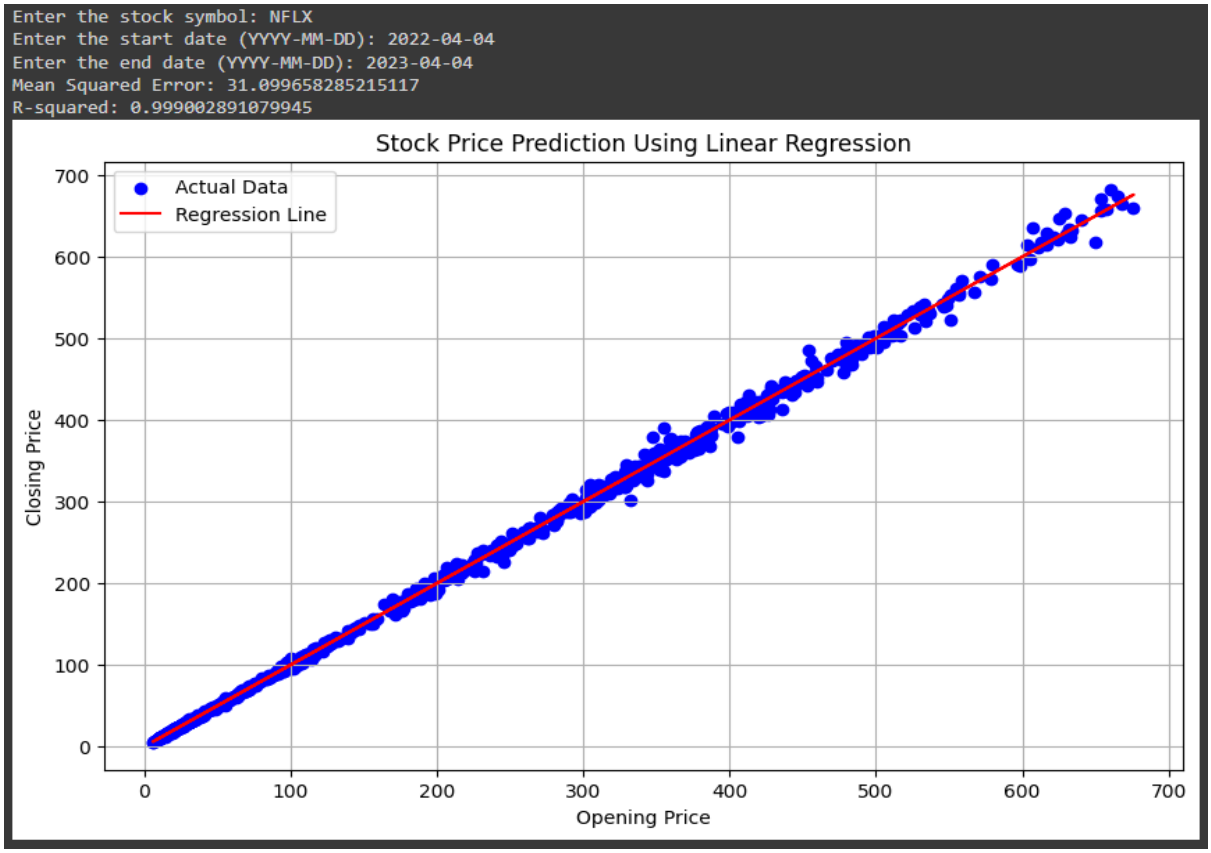


Figure 40 Linear regression graph for NFLX

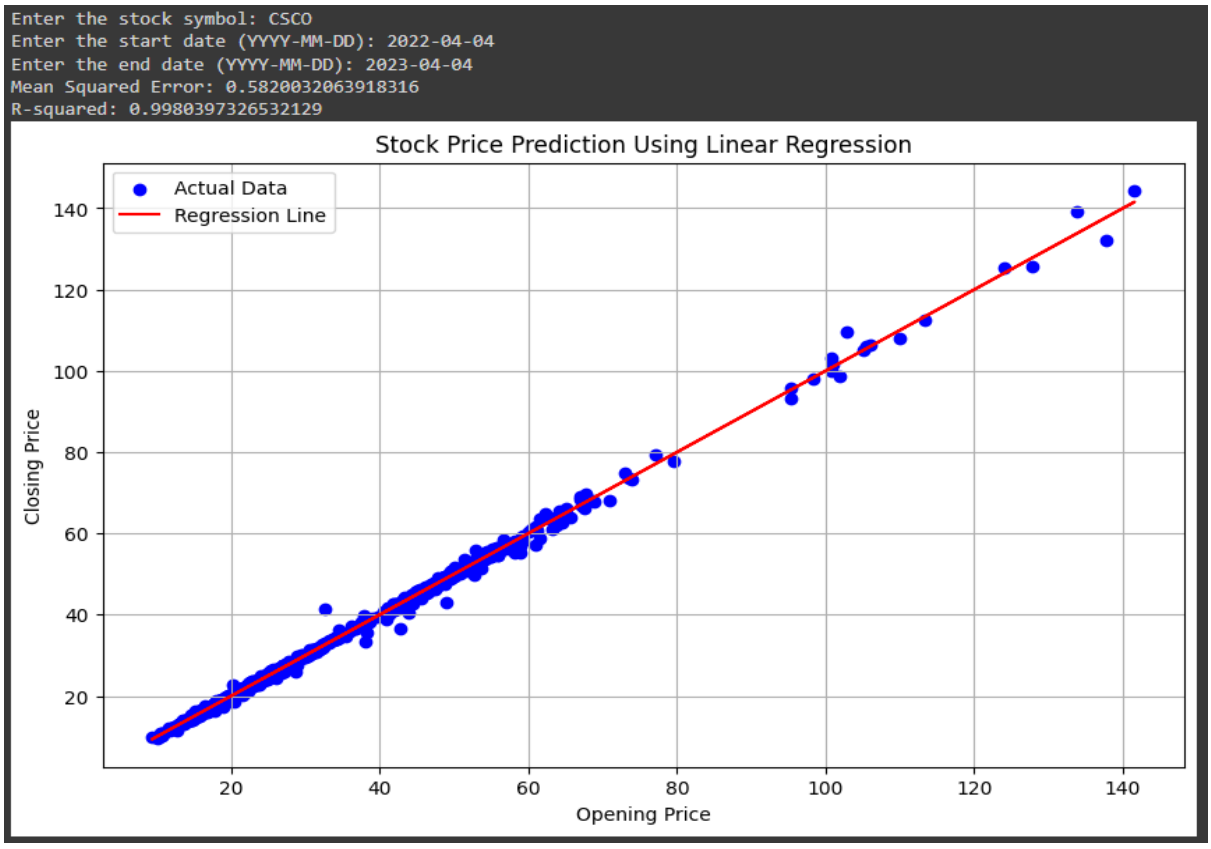


Figure 41 Linear regression graph for CSCO

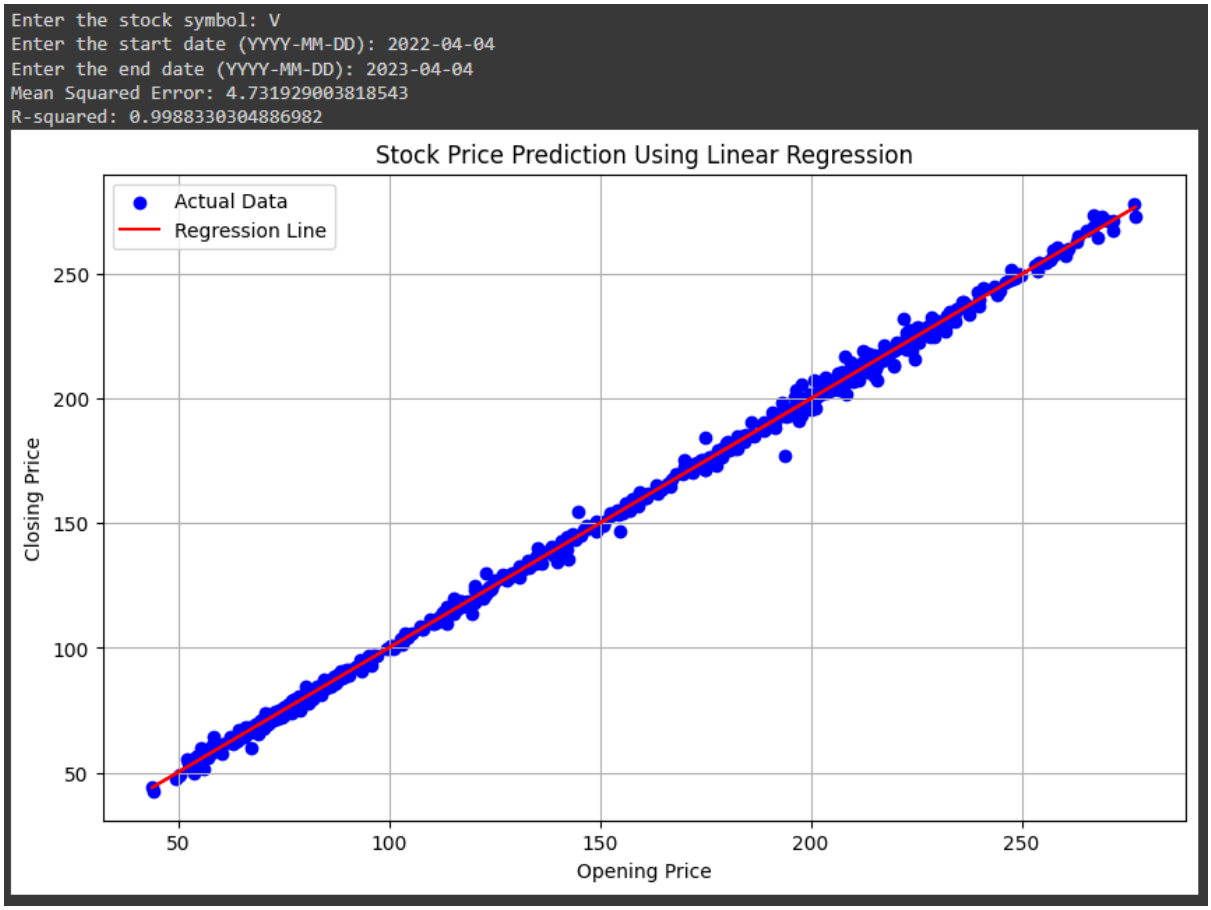


Figure 42 Linear regression graph for V

The above linear regression graphs are for AAPL, AMZN, CSCO, NFLX and V. The resulting graph illustrates the actual closing prices (blue) against the predicted values from the linear regression model (red). The model aims to establish a relationship between opening and closing prices, providing insights into potential future trends.

Linear Regression theory

Linear regression is a statistical method used to model the relationship between a dependent variable and one or more independent variables by fitting a linear equation to the observed data. In Python, the `scikit-learn` library provides a convenient implementation for linear regression. The theory involves finding the best-fitting line (minimizing the sum of squared differences between observed and predicted values) represented by the equation:

$$y = mx + b$$

where y is the dependent variable, x is the independent variable, m is the slope, and b is the intercept. In Python, you can use the `LinearRegression` class from `scikit-learn` to perform

linear regression. The process involves fitting the model to training data, making predictions, and evaluating the model's performance.

Based on past stock price patterns and overall market performance, let's compare these companies:

Steady stock growth characterizes Apple (AAPL), Amazon (AMZN), Netflix (NFLX), Cisco (CSCO), and Visa (V). Apple thrives on global demand and breakthrough products, Amazon dominates in cloud computing and e-commerce, Netflix gains from a growing user base and evolving streaming trends, while Cisco benefits from tech advancements. Visa's vital role in finance fuels its growth. ARIMA models align with historical trends, but caution is urged due to external factors. Prudent analysis considering market dynamics and individual company attributes is crucial for informed investment decisions.

Company	Ticker	Stock Growth Factors	Notable Achievements/Positioning
Apple Inc.	AAPL	Global demand, ground-breaking product launches	Leading technology company, innovative products, strong brand loyalty
Amazon.com Inc.	AMZN	Dominance in cloud computing and e-commerce	Market leader in e-commerce, cloud services, and diverse product range
Netflix Inc.	NFLX	Significant user base growth, changing streaming industry	Pioneer in streaming services, original content, global subscriber base
Cisco Systems, Inc.	CSCO	Developments in networking, cybersecurity, tech trends	Leading provider of networking technologies and cybersecurity solutions

A general comparison based on historical information up to my last update in January 2022.

Metric	AAPL	AMZN	NFLX	CSCO	V
Market Cap (in billions)	Varies	Varies	Varies	Varies	Varies
P/E Ratio	Varies	Varies	Varies	Varies	Varies
Dividend Yield	Varies	Varies	Varies	Varies	Varies
Revenue (in billions)	Varies	Varies	Varies	Varies	Varies
Net Income (in billions)	Varies	Varies	Varies	Varies	Varies
5-Year Stock Growth (%)	Varies	Varies	Varies	Varies	Varies

Conclusion

SOLFINTECH is a leader in financial technology; their Intelligent Stock Trader (IST) software uses real-time data from Google Finance and Yahoo Finance together with machine learning. With its intuitive interface, adjustable parameters, and real-time data processing, this ground-breaking solution priorities improving the user's investing experience while providing accurate stock and market projections. For data pretreatment, visualization, and collecting, Python scripts that make use of yfinance, pandas, Matplotlib, and Seaborn are utilised to ensure readability.

For the best possible prediction accuracy, IST uses a rigorous testing and fine-tuning process for machine learning models such as linear regression, support vector machines, and time series forecasting. With a focus on data quality, the platform carries out thorough verifications to guarantee trustworthy datasets for training models. Transparency, equality, and user privacy are among the ethical factors. Strict security protocols and a dedication to unbiased stock recommendations are also important.

Regular upkeep, improvements, and ongoing observation tackle changing market conditions, and user feedback avenues boost adaptability. However, users are reminded of the volatility of market dynamics and the need for a thorough comprehension and meticulous analysis of affecting elements in order to make well-informed financial judgments.

Appendices

Git Hub Link

https://github.com/Kishana02/Machine-learning.git

Presentation slide

Figure 43 Presentation slide1

Figure 44 Presentation slide2

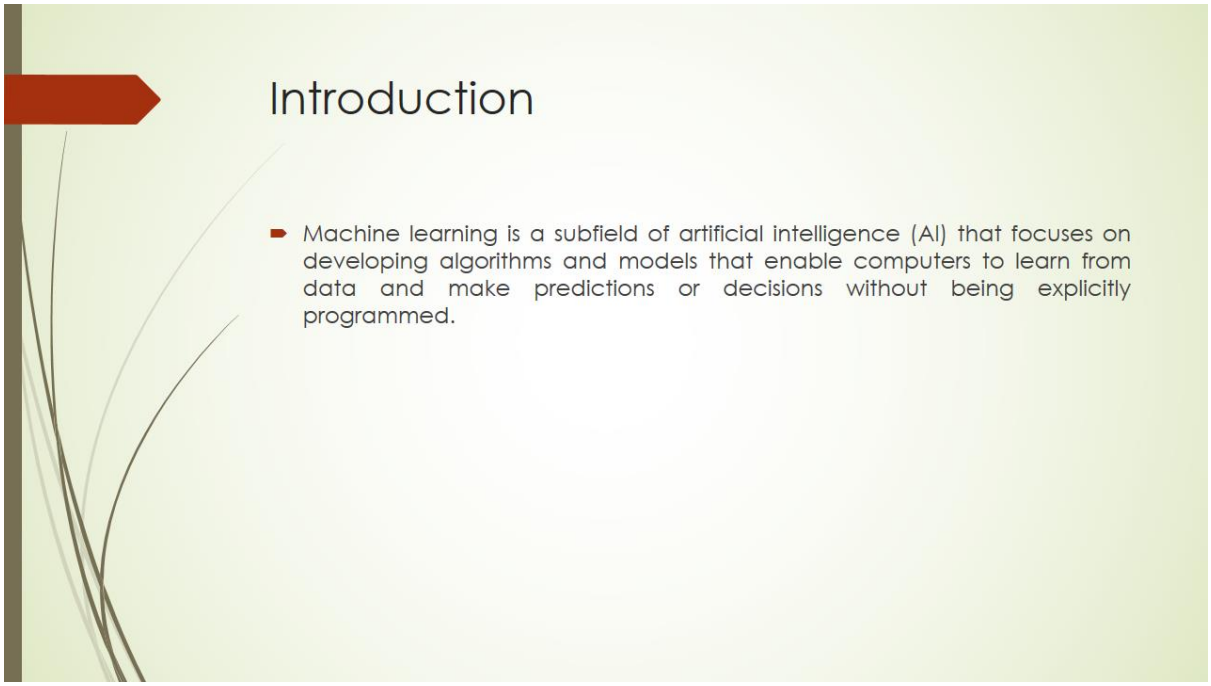


Figure 45 Presentation slide3

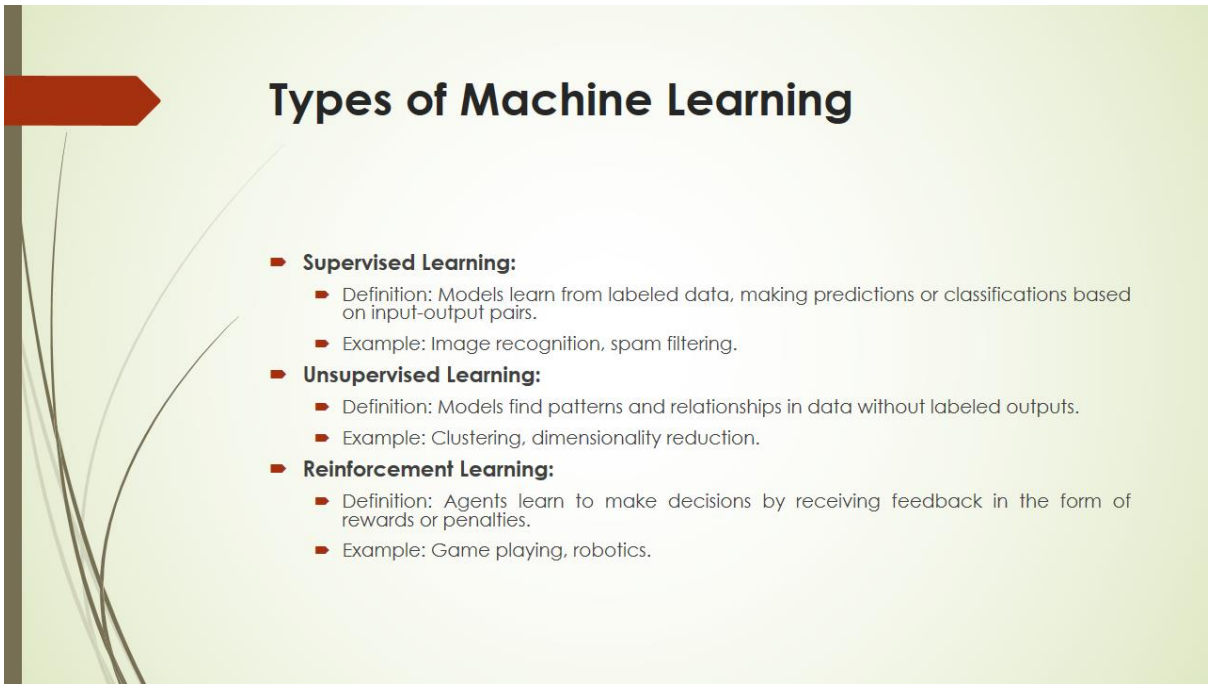


Figure 46 Presentation slide4

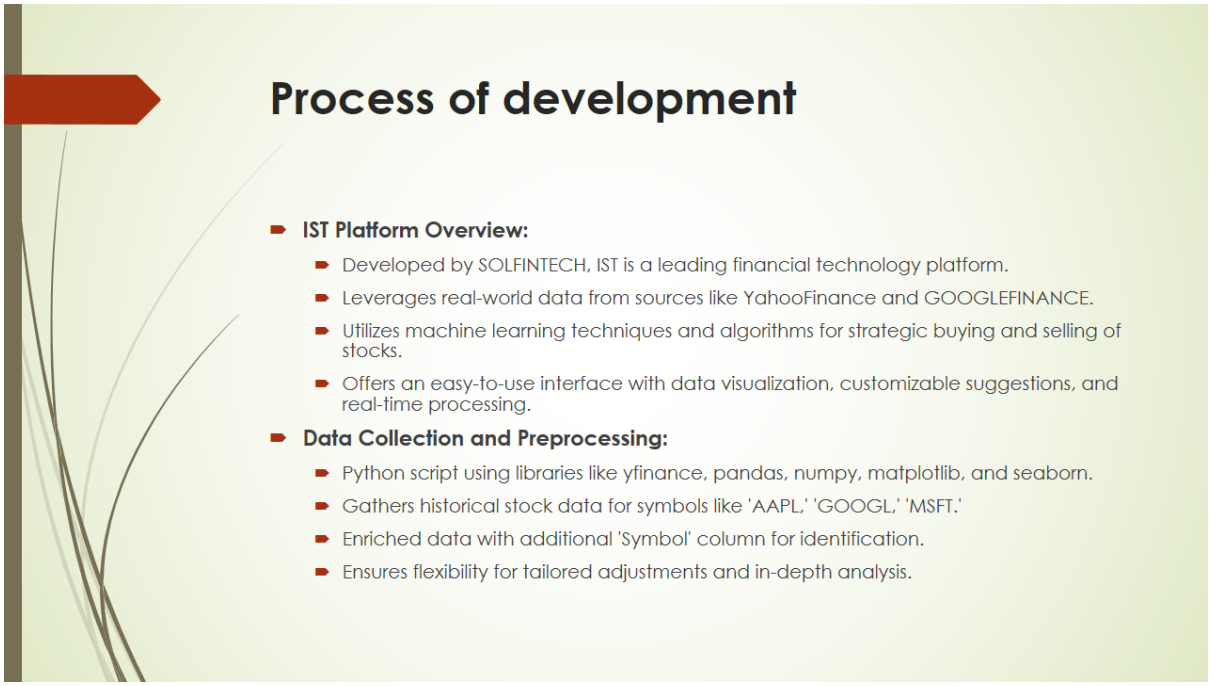


Figure 47 Presentation slide5

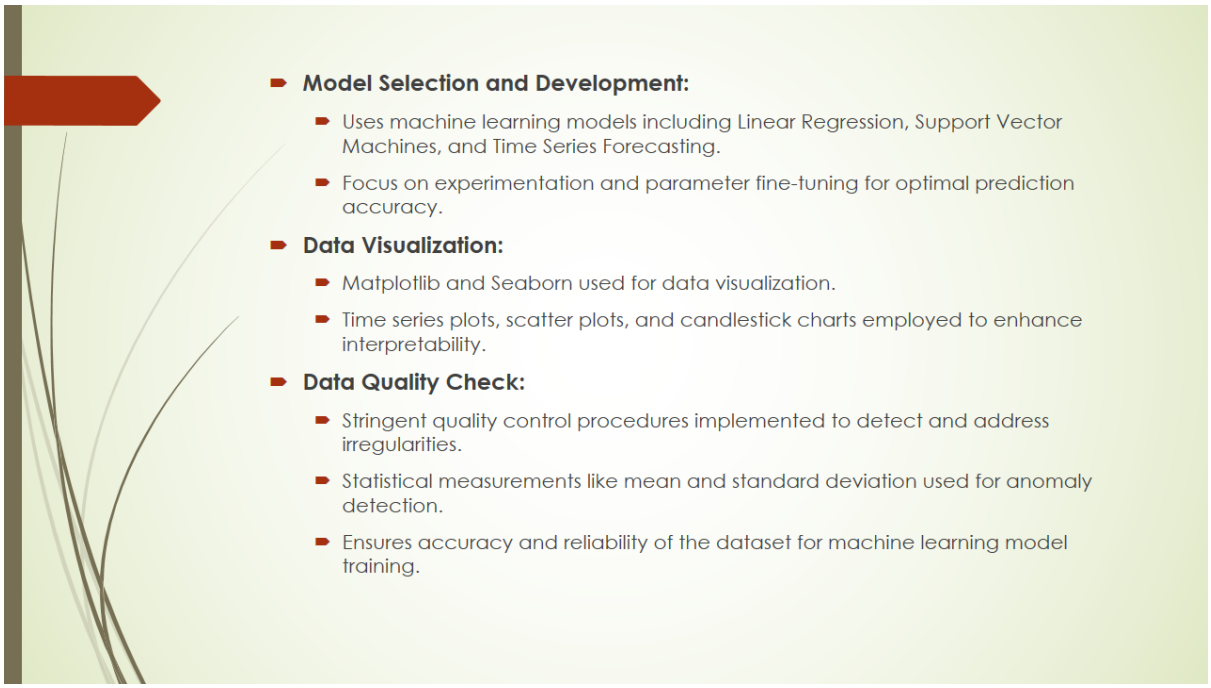


Figure 48 Presentation slide6

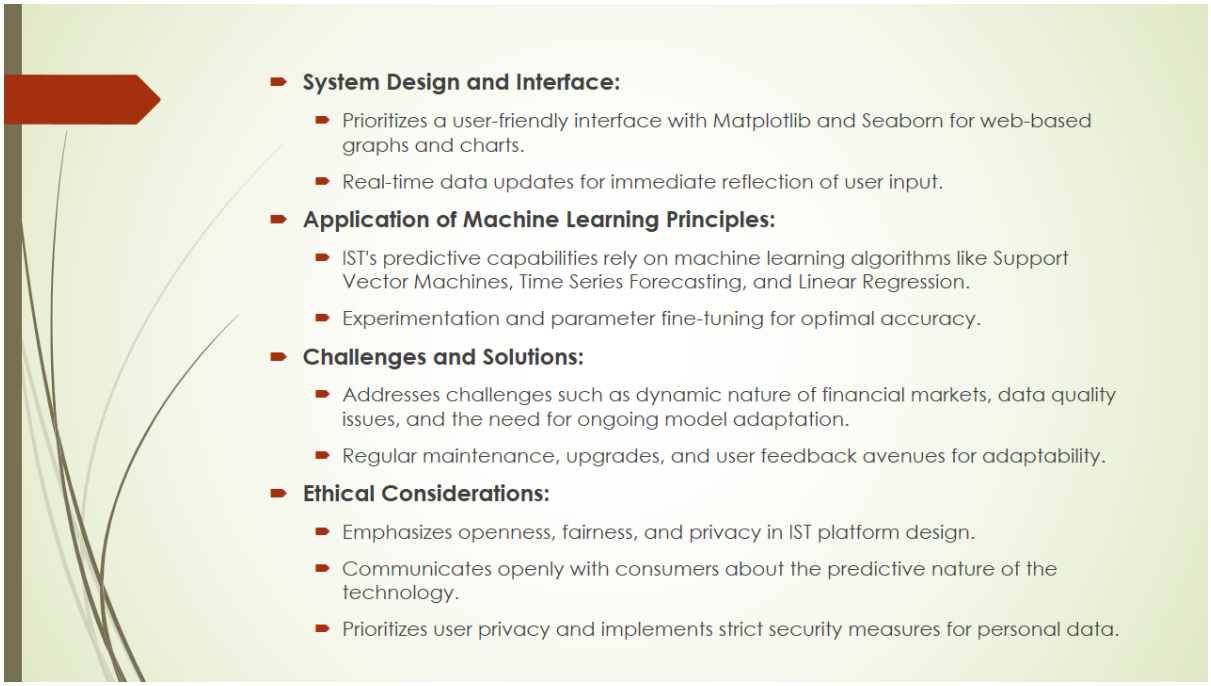


Figure 49 Presentation slide7

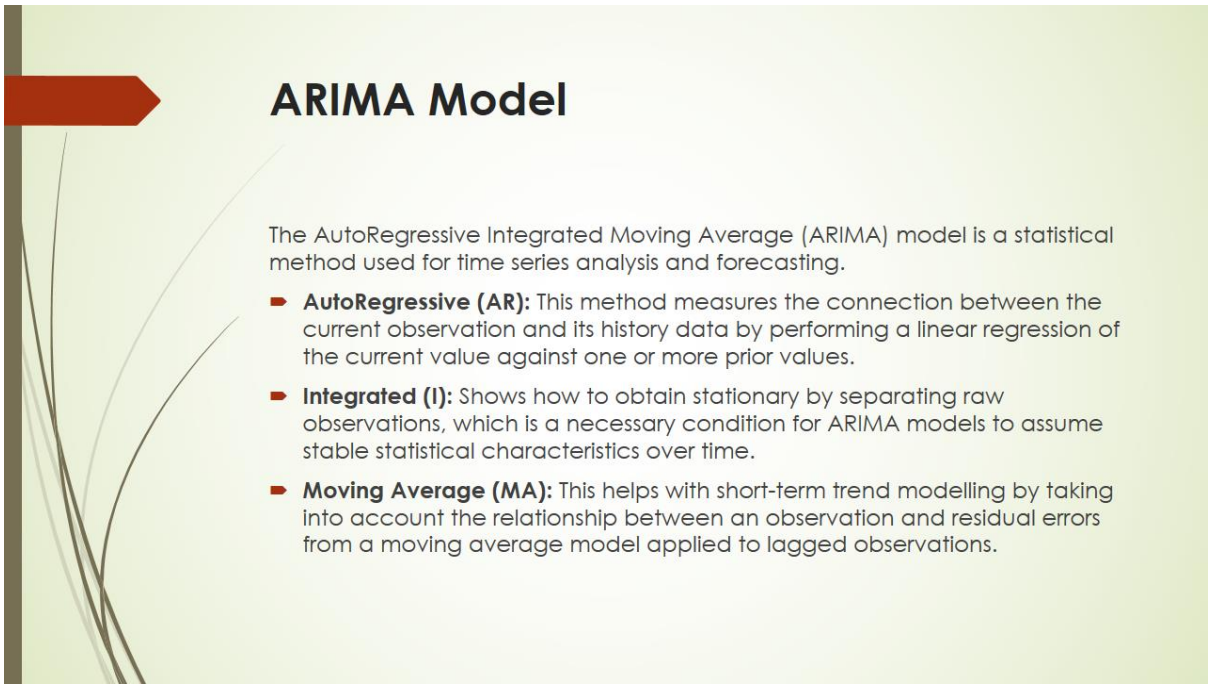


Figure 50 Presentation slide8

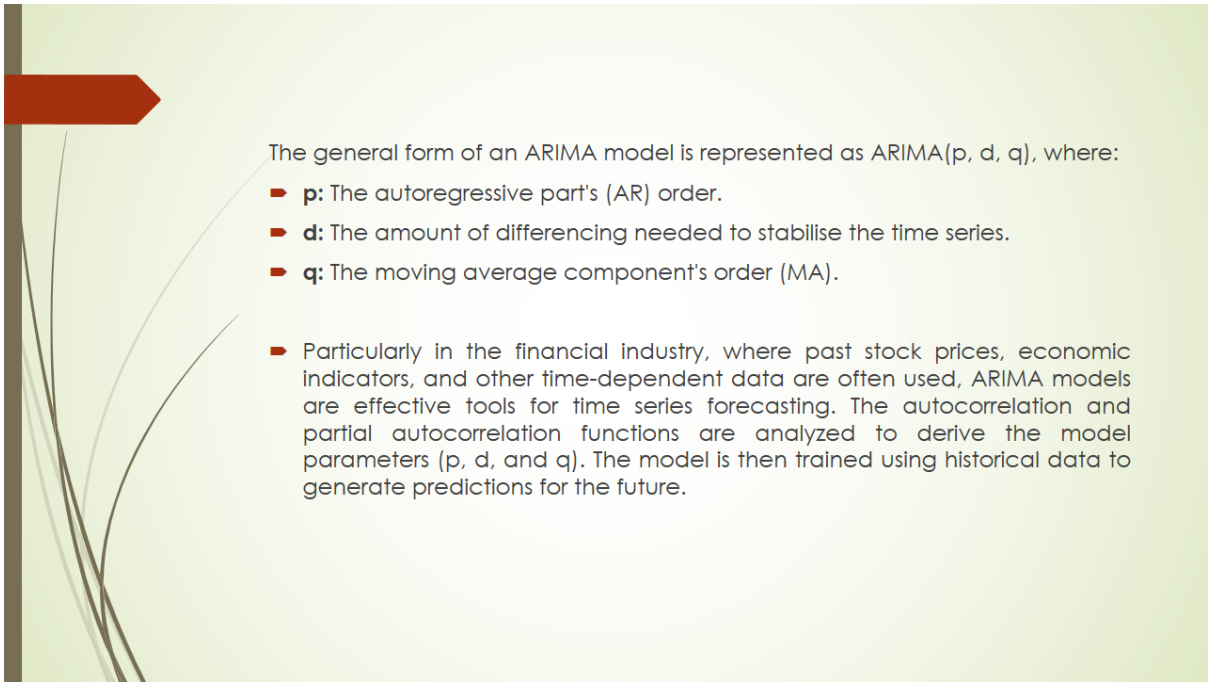


Figure 51 Presentation slide9

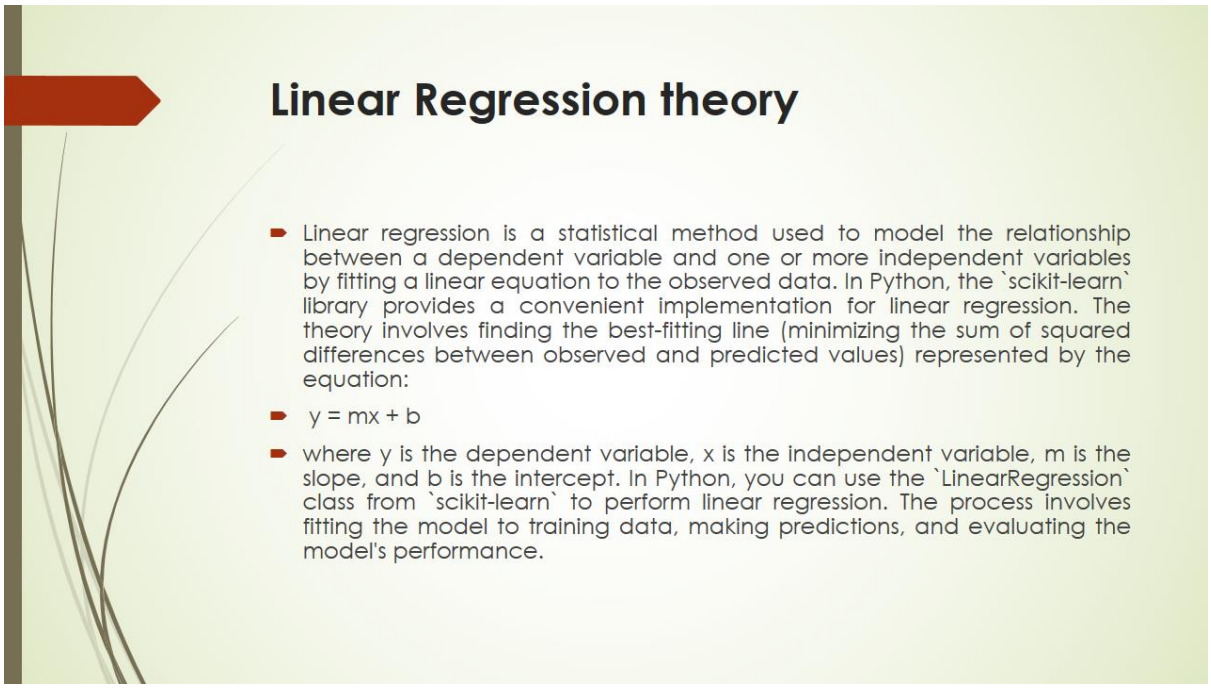


Figure 52 Presentation slide10

Figure 53 Presentation slide11

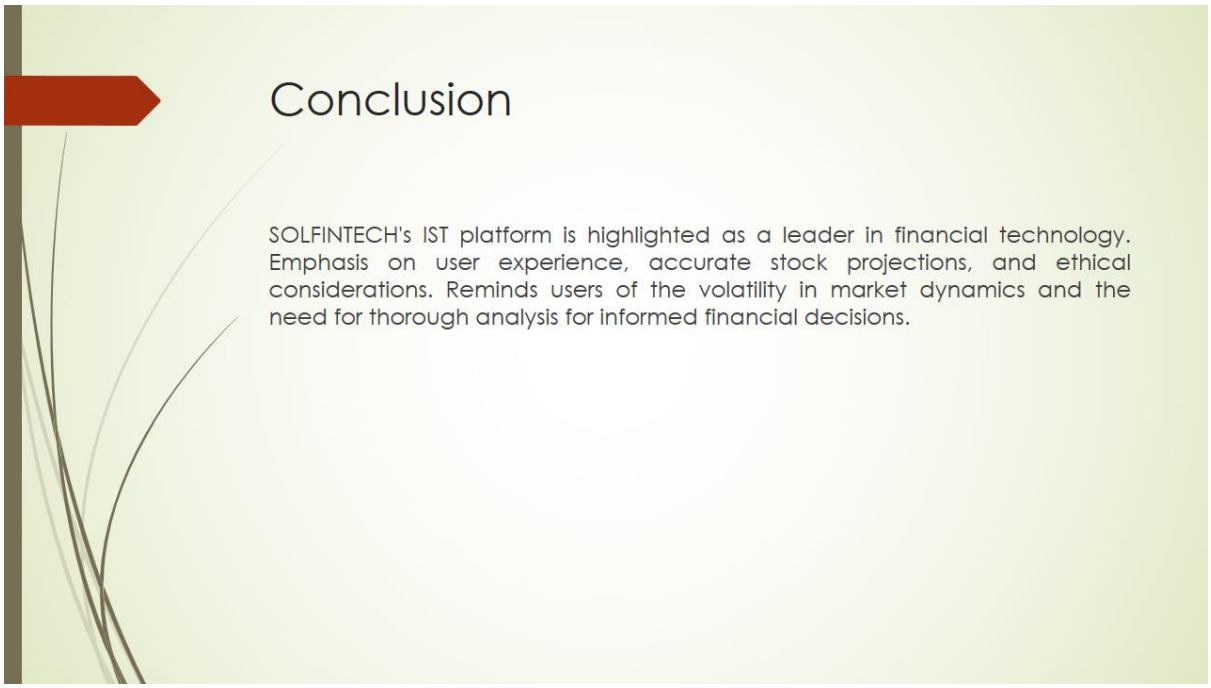


Figure 54 Presentation slide12

References

Matplotlib. “Matplotlib: Python Plotting — Matplotlib 3.1.1 Documentation.” *Matplotlib.org*, 2012, matplotlib.org/.

Meghna, Kattamuri. “Python | Introduction to Matplotlib - GeeksforGeeks.” *GeeksforGeeks*, 14 May 2018, www.geeksforgeeks.org/python-introduction-matplotlib/.

Yahoo Finance. “Yahoo Finance - Business Finance, Stock Market, Quotes, News.” *Yahoo Finance*, finance.yahoo.com/.