Costco Stock Price Prediction Framework

Submitted by: Team Group #5
Manipal Sidhu
Mahpara Radmy
Ronald Saenz
Kanishka Dhir
Vikas Trivedi
Daniyal Ahmed

Submitted to: Hao Lac
ICET Department,
School of Engineering Technology and Applied Science
Progress Campus, Block A
Centennial College

Discipline: Software Engineering

Due Date: December 10, 2023

Declaration of Sole Authorship

We, Team Group #5, confirm that this work submitted for assessment is our own and is expressed in our own words. Any uses made within it of the works of any other author, in any form (ideas, equations, figures, texts, tables, programs), are properly acknowledged at the point of use. A list of the references used is included.

Signed: Manipal Sidhu, 3008XXXXX (Software Engineering Technology - AI)

Signed: Ronald Saenz, 3012XXXXX (Software Engineering Technology - AI)

Signed: Mahpara Radmy, 3011XXXXX (Software Engineering Technology - AI)

Signed: Kanishka Dhir, 3012XXXXX (Software Engineering Technology - AI)

Signed: Vikas Trivedi, 3012XXXXX (Software Engineering Technology - AI)

Signed: Daniyal Ahmed, 3011XXXXX (Software Engineering Technology - AI)

Date: December 10, 2023

Abstract

In today's dynamic global stock markets, investors continually seek reliable insights into specific brands with Costco being a prime example. The challenge lies not just in predicting Costco's stock trajectory, but also in understanding the countless factors that influence its market performance. Addressing this challenge is of paramount importance because with more precise predictions, investors can make informed decisions, potentially maximizing their returns on investments in brands like Costco. To confront this issue, our team proposes a solution: a specialized Stock Price Prediction application tailored for Costco. Using advanced machine learning techniques and leveraging a dataset that chronicles Costco's stock prices daily since January 2000. Our solution aims to provide unparalleled insights. By amalgamating segmentation, classification, and regression models. We intend to categorize Costco's stock based on historical data and other salient market factors. Furthermore, our application will offer future stock price predictions, underpinned by meticulous market sentiment analysis and regression techniques. This innovative approach is designed to furnish investors with real-time actionable insights specifically into Costco's stock performance.

Table of Contents

1 INTRODUCTION

The core technical challenge our team is addressing revolves around forecasting time-series data, specifically predicting Costco's stock prices. One of the challenging aspects is representing the stock data in a format suit. Choosing the optimal window size enough to cover the relevant patterns and not too big for the model to learn dependencies is crucial. Preprocessing the volume feature of stocks and using it for the prediction of future stock prices is another technical challenge. Therefore, transforming this data into sequences that capture historical trends while predicting future values like the closing price for the next week is complicated.

Inclusions:

- Historical stock data of Costco, emphasizing features like opening price, closing price, volume, highs, and lows.
- Application of a sliding window approach to transform time series data into sequences for training the LSTM model.
- Training and evaluation of an LSTM model tailored for weekly stock price predictions.

Exclusions:

- Real-time events or news sentiments which can influence stock prices.
- Integration with third-party visualization tools or platforms for displaying stock trends.
- External economic indicators or data from other brands/industries

Objectives:

- To design and implement an LSTM-based model capable of predicting the closing stock price of Costco for the upcoming week.
- To analyze and understand the historical trends and patterns in Costco's stock data.
- To provide stakeholders with a reliable forecasting tool that aids in investment decisions.

Unique Problems:

One particularly complicated issue was the determination of the optimal input size for the sliding window approach. Selecting the right window size is important because it defines the number of previous times steps the model considers while making a prediction. Too small a window might not capture enough historical context and causing the model to miss longer-term patterns. Conversely, an excessively large window might introduce noise and making the model overly complex and prone to overfitting.

Innovative Ideas:

Our strategy combines current deep learning algorithms with established time series forecasting methodologies. One such innovative approach is using of a sliding window to convert stock data into sequences appropriate for LSTM training.

2 METHODOLOGY AND RESULTS

2.1 Literature Review

The field of stock price prediction has witnessed a surge in interest and research in recent years, driven by the increasing availability of financial data and the

potential for leveraging machine learning algorithms to gain insights into the dynamic and complex nature of financial markets. Accurate stock price prediction holds significant implications for investors, financial institutions, and policymakers alike. This literature review aims to provide a comprehensive assessment of existing solutions for stock price prediction, shedding light on the strengths and weaknesses of different methodologies. In this section, we delve into two prominent approaches in the field: Support Vector Machines (SVMs) and Random Forest, as showcased in seminal works by Hsu and Lin (2003) and Li et al. (2014) respectively. By examining these approaches, we aim to offer valuable insights for researchers and practitioners seeking to navigate the ever-evolving landscape of stock price prediction algorithms.

Firstly, the paper called “Stock Price Prediction Using Support Vector Machines” written by Hsu and Lin highlights the application of Support Vector Machines in stock price prediction, emphasizing their robustness and non-linearity handling capabilities. While SVMs offer advantages in certain aspects, they may still require careful feature engineering and hyperparameter tuning, and they do not inherently capture temporal relationships in stock price data. Researchers and practitioners should consider these factors when choosing an algorithm for stock price prediction based on their specific requirements and data characteristics.

There are some strengths evaluated in this paper:

- **SVM Application:** This paper explores the application of Support Vector Machines (SVMs) to stock price prediction. SVMs are known for their strong generalization capabilities and ability to handle high-dimensional data.
- **Robustness:** SVMs are less prone to overfitting compared to some other

machine learning algorithms. They can provide stable and reliable predictions, especially when dealing with noisy financial data.

- **Non-Linearity Handling:** SVMs can efficiently handle non-linear relationships between input features and stock prices by using kernel functions.

There are some weaknesses evaluated in this paper:

- **Feature Engineering:** Like many traditional machine learning approaches, SVMs often require manual feature engineering to extract meaningful features from raw financial data.
- **Data Sensitivity:** The performance of SVMs can be highly sensitive to the choice of kernel function and hyperparameters. Selecting the right settings may require extensive tuning.
- **Lack of Sequential Modeling:** SVMs are not designed for sequential data, and they do not inherently capture temporal dependencies present in stock price time series.

Secondly, the paper called “Stock Price Prediction Using Random Forest for the SSE Composite Index in China” written by Li et al (2014) highlights the application of Support Vector Machines in stock price prediction, emphasizing their robustness and non-linearity handling capabilities. While SVMs offer advantages in certain aspects, they may still require careful feature engineering and hyperparameter tuning, and they do not inherently capture temporal relationships in stock price data. Researchers and practitioners should consider these factors when choosing an algorithm for stock price prediction based on their specific requirements and data characteristics.

The proposed solution in the paper, titled "Stock Price Prediction Using Random Forest for the SSE Composite Index in China" by Li et al. (2014), focuses on applying the Random Forest algorithm to predict stock prices, specifically for the SSE Composite Index in China. Random Forest, an ensemble learning method, is employed to harness the collective predictive power of multiple decision trees. This approach is known for its robustness, ability to handle large datasets, and feature importance analysis.

The study emphasizes the interpretability of Random Forest models, as they provide insights into the significance of each feature in influencing stock price changes. While Random Forest is not designed for sequential data like deep learning models, it offers a computationally efficient alternative, making it suitable for stock markets with limited historical data.

There are some strengths evaluated in this paper:

- **Ensemble Approach:** This article applies Random Forest, an ensemble learning technique, to predict stock prices for the SSE Composite Index in China. The ensemble approach often leads to robust and accurate predictions.
- **Feature Importance:** Random Forest provides insights into feature importance, allowing analysts to identify which variables have the most impact on stock price changes.
- **Scalability:** Random Forest models can be more scalable and require fewer computational resources compared to deep learning methods, making them suitable for large datasets.

There are some weaknesses evaluated in this paper:

- **Sequential Data Handling:** Random Forest is not designed to handle sequential data as effectively as deep learning models like LSTM. It may struggle to capture subtle temporal patterns.
- **Feature Engineering:** Unlike deep learning models that can learn features, Random Forest requires manual feature engineering, which can be time-consuming and may introduce biases.
- **Interpretability Limitations:** While Random Forest provides feature importance scores, the model's decision-making process can still lack transparency compared to some deep learning models.

Finally, these two studies demonstrate different approaches to stock price prediction. The first research project work leverages Machine Learning techniques like Support Vector Machines which can easily handle the robust and non-linear nature of data but face many issues when choices of feature engineering and kernels are considered. In contrast, Li et al.'s study uses Random Forest, an ensemble method, with a focus on feature importance and scalability.

In conclusion, the choice of approach should consider factors such as data availability, computational resources, and the specific requirements of the task. Researchers and practitioners in stock price prediction should weigh the strengths and weaknesses of these approaches when selecting the most appropriate algorithm for their needs. Additionally, exploring hybrid models that combine the strengths of both deep learning and ensemble methods may offer promising avenues for improving predictive accuracy and interpretability.

2.2 Proposed Solution

We propose a time series forecasting solution that utilizes Long Short-Term Memory (LSTM) networks which specifically designed to capture long-term dependencies in sequence data. Our goal is to predict the closing stock price of Costco for the next week based on historical data.

There are some strengths:

- LSTMs are designed to recognize and remember long-term patterns, making them appropriate for time series data like stock prices.
- The LSTM model can be easily scaled or modified by adding more layers or units to capture complex patterns.
- By using a sliding window approach, we can efficiently predict stock prices for the upcoming week and help investors in making informed decisions.

There are some weaknesses:

- The more input you have, it become more complex. While using sliding window approaches if we have more input and it will become more complex vector.
- LSTMs are computationally intensive and might require significant resources for training especially with large and complex timeseries datasets which.
- LSTMs have various hyperparameters for example, number of layers, units and learning rate that need fine-tuning for optimal performance.
- Due to their complexity, can be overfit to the training data if not regularized or trained properly.

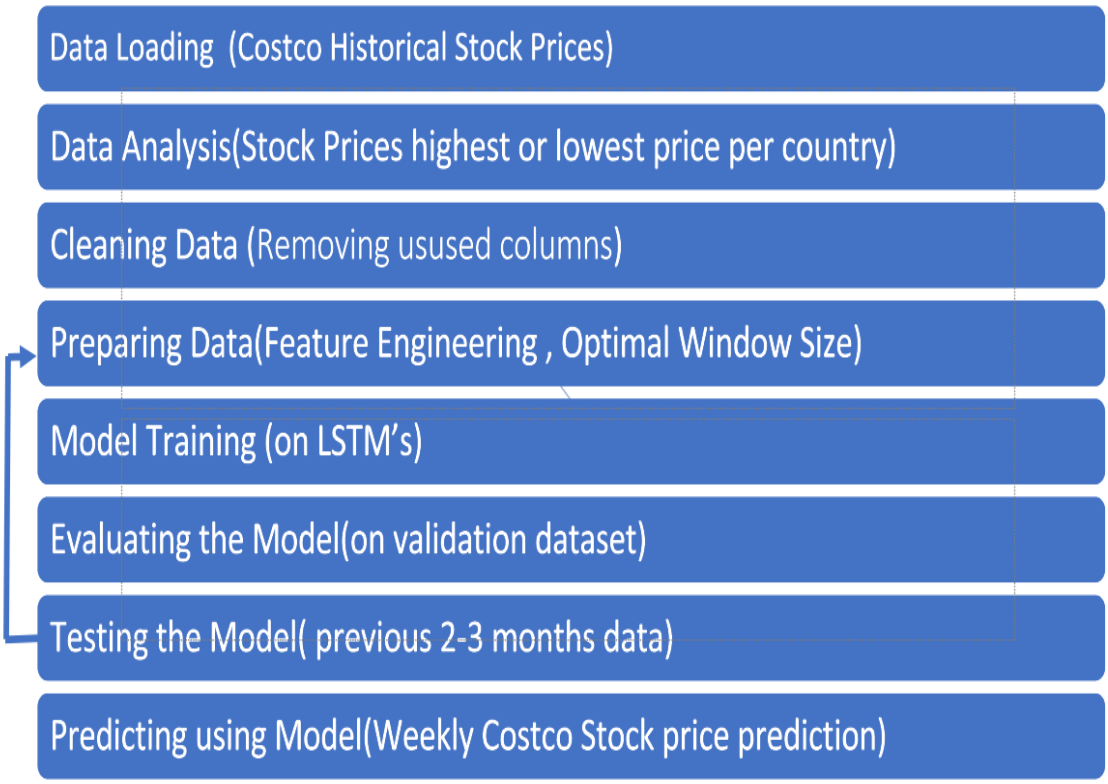

Figure 1. Machine Learning workflow

As shown in Figure 1, the diagram highlights Machine Learning workflow that would be used in the context of stock price prediction. There are 8 steps in out Machine Learning workflow:

1. **Data Loading:** Gathering and loading of the Costco Stock Prices Dataset that contains the following columns: Date, Time, Open, High, Low, Close, Volume, Brand_Name, Ticker, Industry_Tag, Country, Dividends and Stock Splits.
2. **Data Analysis:** Analyzing the dataset for highest and lowest stock price of Costco per country. Examining for Volume of stocks and how we could incorporate volume in prediction of new stock prices.

3. **Data Cleaning:** Checking for columns that very few values in them and removing them. Dividends columns has only 83 non-zero values and Stock splits has only a single nonzero value. These columns do not have sufficient data in them that could contribute to the prediction of our model.
4. **Preparing Data:** Performing feature engineering and preparing data in the suitable format to be consumed by LSTM model is one of the bigger challenges. Selecting the optimal window size for our time series data is critical.
5. **Model Training:** Training our model on Long Short-Term Memory model for our time series forecasting solution.
6. **Evaluating the model:** We are dividing the dataset into training and testing by keeping the data for last 3 months as our testing data. To evaluate and train the model better, we will divide the training data into training and validation datasets. Model will be evaluated upon the validation dataset.
7. **Testing the model:** Model will be tested upon the last 3 months data, to check for testing accuracy. For our model to perform better and get good test results, we would have to prepare our data again by making changes to parameters and train the model again.
8. **Predicting using Model:** The model will then be used for predicting the weekly Costco Stock Prices that will be of great value to Costco stock price shareholders and help them get an insight into the retail market.

2.3 User Role Modeling

Certainly, for a Costco stock market prediction system focused on simplicity and accessibility our user roles might be streamlined to just two primary categories: clients, who are the end-users of the system and administrators, who manage and maintain the system.

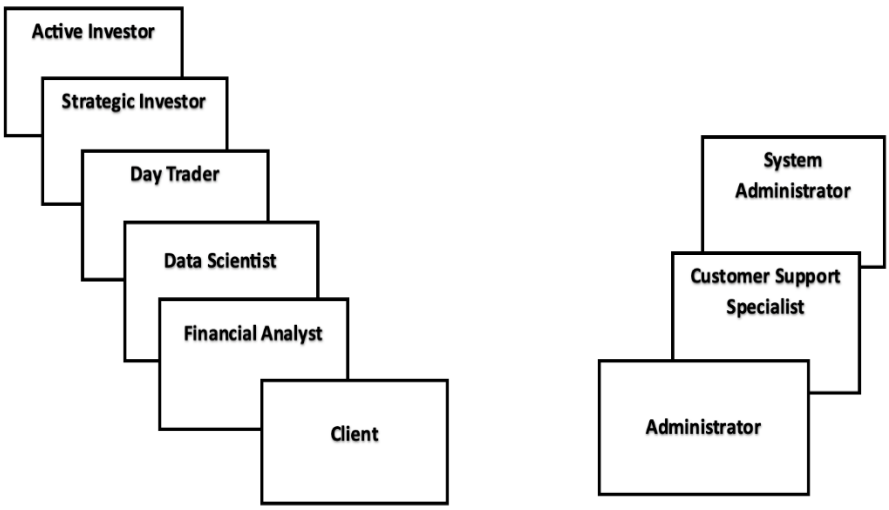

Figure: Organizing the user role cards on table

Figure 2. Organizing the user role cards on table

As shown in Figure 2, the diagram highlights the user role cards for **"Client"**:

- **Active Investor:** Active investors typically access the stock market prediction system multiple times a day to track market movements, get real-time analytics, and make timely trades.
- **Strategic Investor:** Strategic investors may not check the system as often as active traders, perhaps weekly or monthly, aligning with their long-term investment strategies. They tend to have a deep understanding of market

fundamentals, economic indicators, and financial forecasting, which allows them to plan out long-term investments.

- **Day Trader:** Highly active and requiring rapid, reliable information, the day trader uses the system frequently throughout the day for immediate data to inform swift stock trading decisions.
- **Data Scientist:** Analyzes and interprets complex digital data, such as the usage statistics of the system.
- **Financial Analyst:** An expert who inputs or verifies prediction data and algorithms in the system.

As shown in Figure 2, the diagram highlights the user role cards for **"Administrator"**:

- **System Administrator:** Responsible for the maintenance and overall functioning of the prediction system.
- **Customer Support Specialist:** Provides assistance to clients using the system for their stock market predictions.

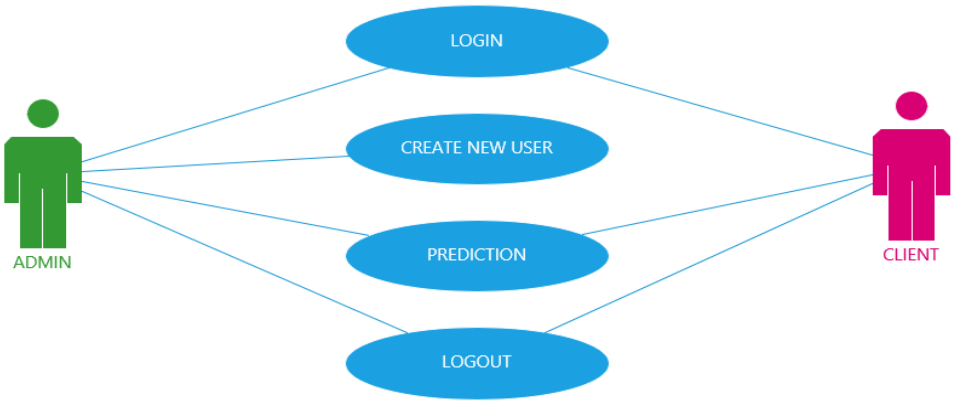

Figure 3. User Role Modelling – User Roles

As shown in Figure 3, the diagram highlights two types of user roles i.e., Admin and Client. Both roles have permission to “Login” and “Logout” of the system which are fundamental functions for access control. However, the Admin has additional privileges that include the ability to “Create New User” accounts which is suggesting administrative control over user management. Both the Admin and the Client can access the “Prediction” function which implies that both roles are permitted to view or interact with the Costco stock price (Close) predictions.

2.4 Frontend User Interface

There are some elements of the technology stack that used for the project:

- **JavaScript** - a scripting language that has built-in support for making HTTP requests and handling responses from RESTful APIs in a cloud environment.
- **HTML** - a standard markup language for creating web pages and web applications. It provides a way to organize text, images, links, and other elements, allowing for the creation of interactive and dynamic web pages.
- **CSS** - a style sheet language used for describing the presentation of a document written in HTML. It tells how elements should be displayed on a web page, including aspects such as layout, colors, fonts, and spacing.
- **Bootstrap** - simplifies the process of building responsive and mobile-first websites and web applications. It provides a comprehensive set of CSS and JavaScript components for creating user interfaces, including buttons, forms, navigation bars, and more.
- **React** - a JavaScript library to implement web and native user interfaces.

- **Heroku** – a container-based used for cloud Platform as a Service (PaaS) to implement our Frontend project.
- **Platform as a Service (PaaS)** - a service that support users into the web application lifecycle: building, testing, deploying, managing, and updating.

There are some mockup user interfaces that they were necessary to implement in the project:

2.4.1 Mockup 1: Home Page

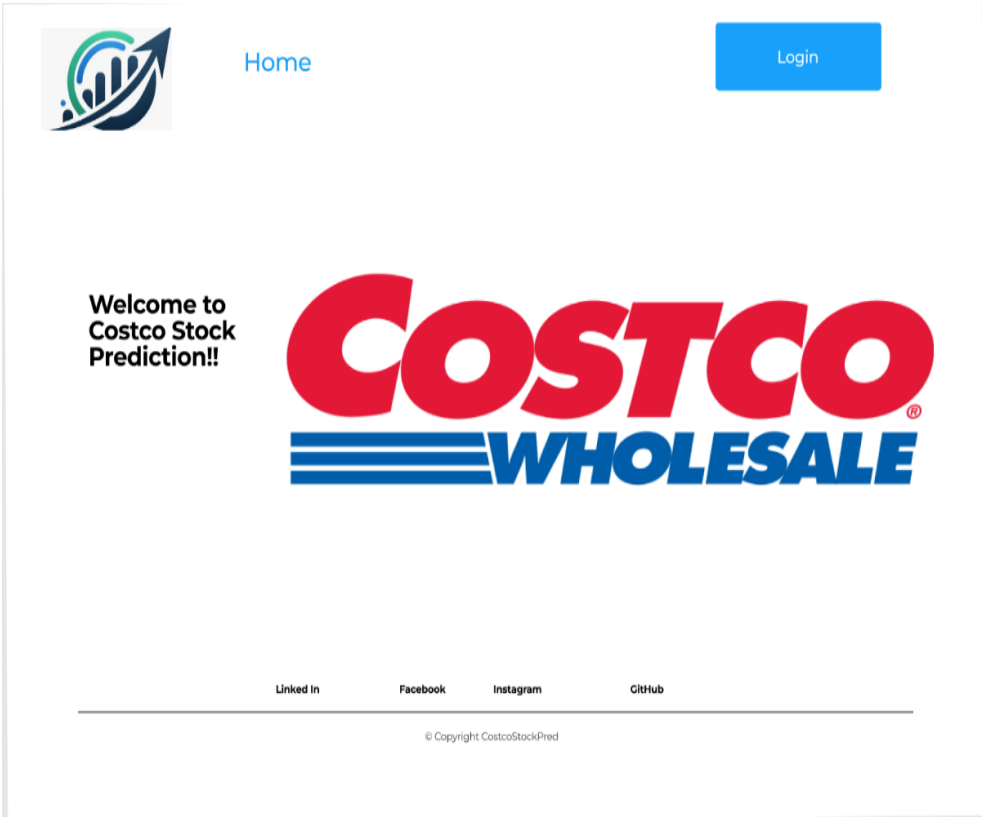

Figure 4. Home Page mockup

2.4.2 Mockup 2: Login Page

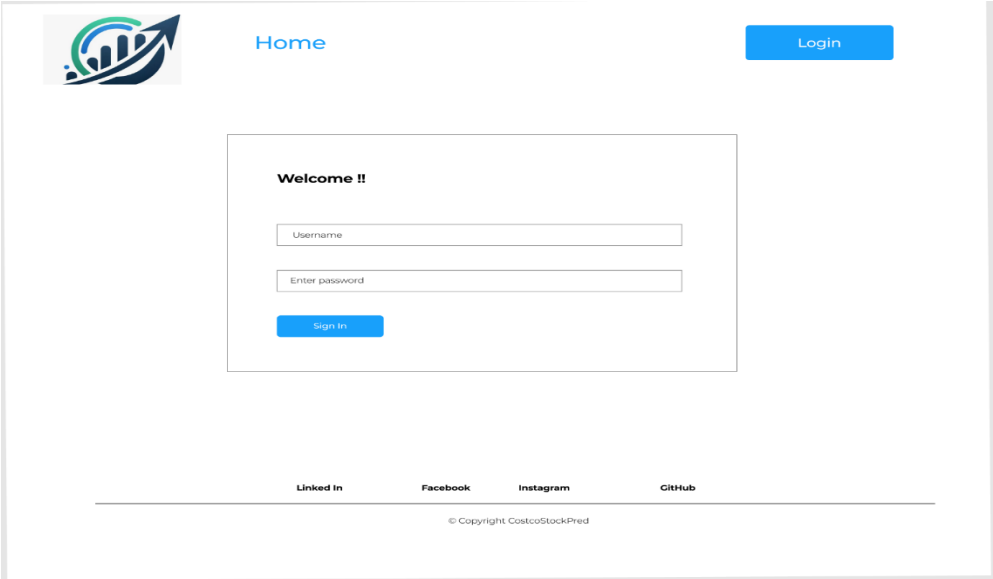

Figure 5. Login Page mockup

2.4.3 Mockup 3: Welcome Page

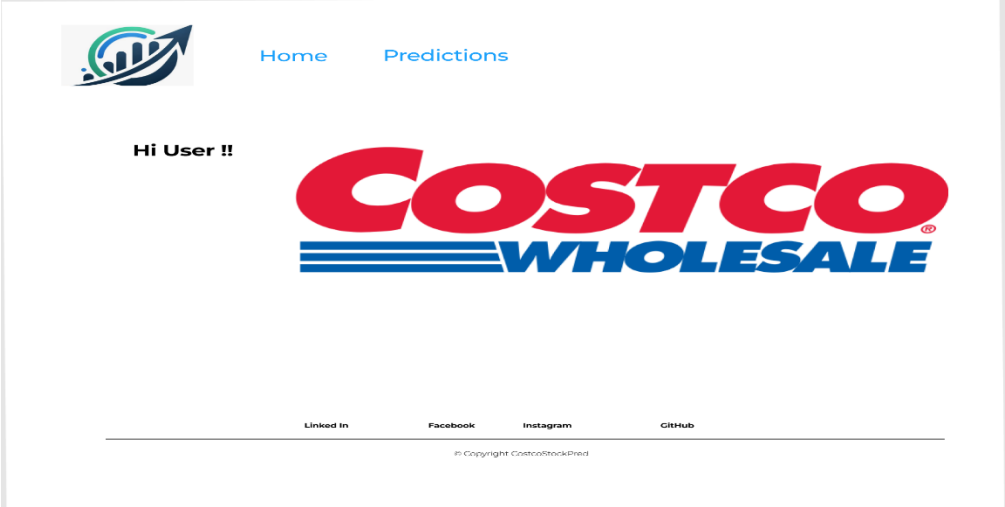

Figure 6. Welcome Page mockup

2.4.4 Mockup 4: Stock Price Predictions Page

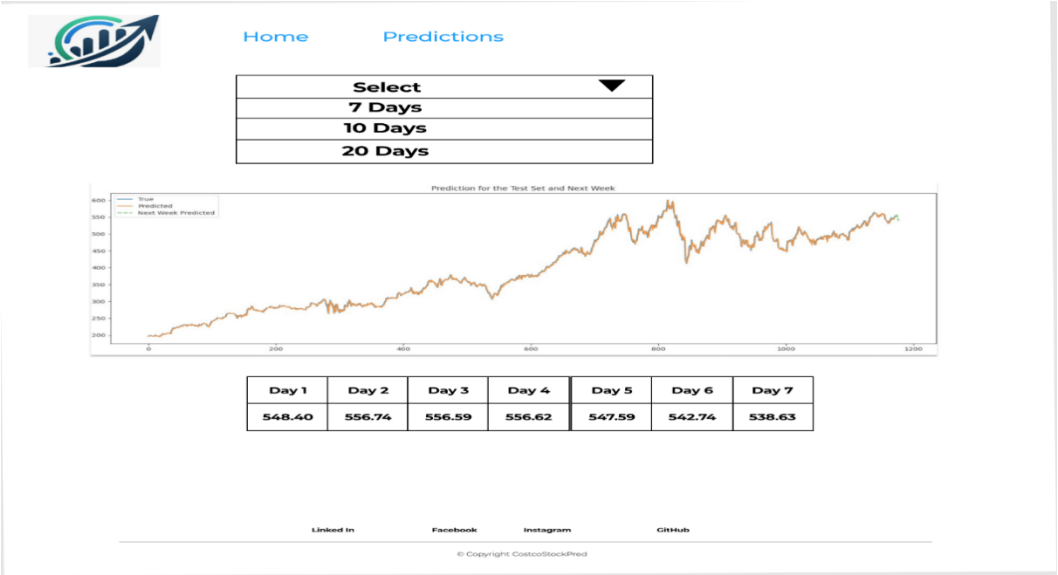

Figure 7. Stock Price Predictions Page mockup

2.4.5 Frontend Implementation

To implement the interface, we use the Heroku platform. Figure 8 shows the Stock Price Predictions web page.

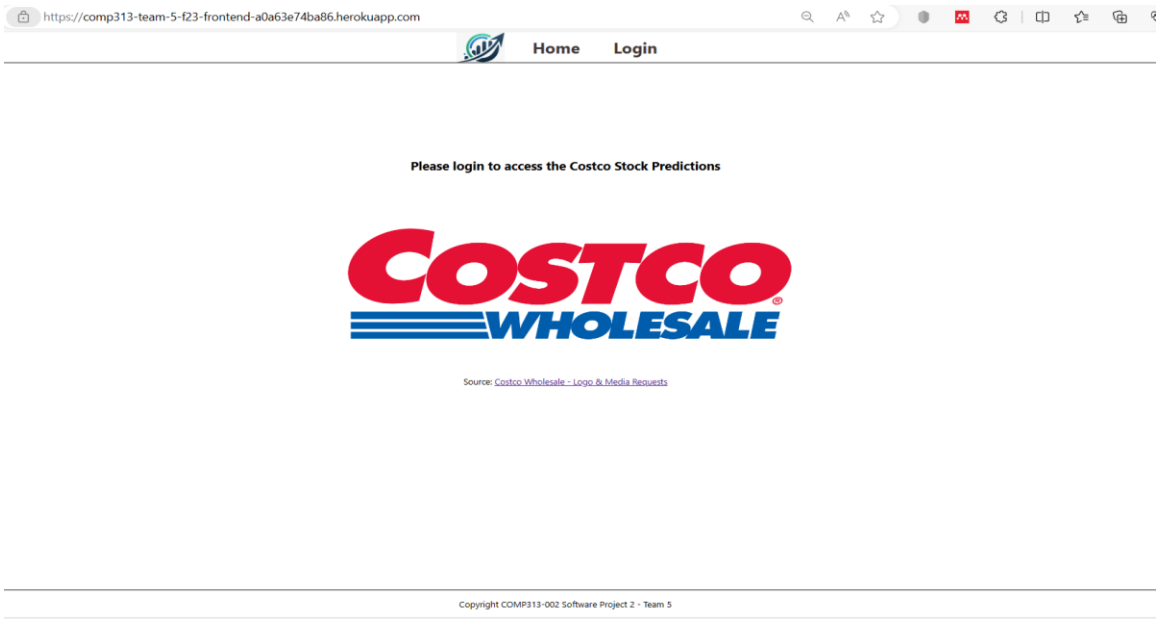

Figure 8. Frontend - Welcome Page

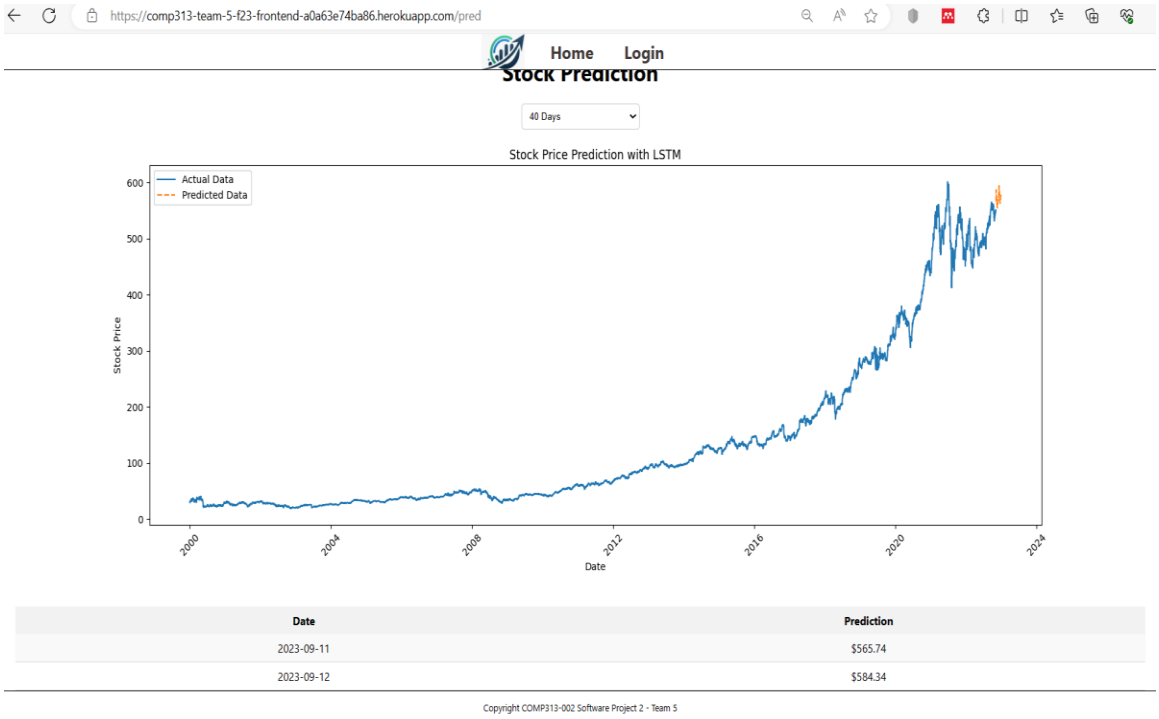

Figure 9. Frontend – Stock Price Predictions Page

2.5 Cloud Backend

There are some elements of the technology stack that used for the project:

- **Python** - a backend programming language that will be used to create the RESTful API, define functions, and integrate with Azure services.
- **Azure Blob Storage** - A scalable object storage service that will be used to store the H5 model.
- **Azure SQL Database** - A fully managed relational database service that is suitable to store the users, roles, historical predictions.
- **Azure Functions** - A serverless compute service that enables us to run even-driven code without managing infrastructure.
- **RESTful API** - A microservices architecture that will be used to create a scalable and flexible system for managing the predictions of stock prices.

- **Function as a Service (FaaS)** - a serverless architecture that allows for automatic scaling and cost optimization.

2.5.1 Architecture Design Graph – High Level

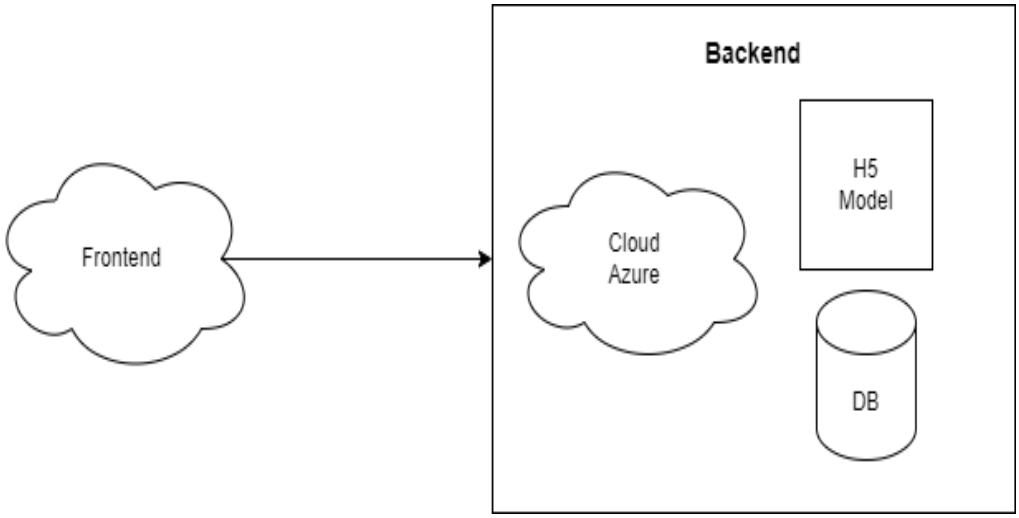

Figure 10. Cloud Backend Architecture Design – High Level

2.5.2 Architecture Design Graph – Low Level

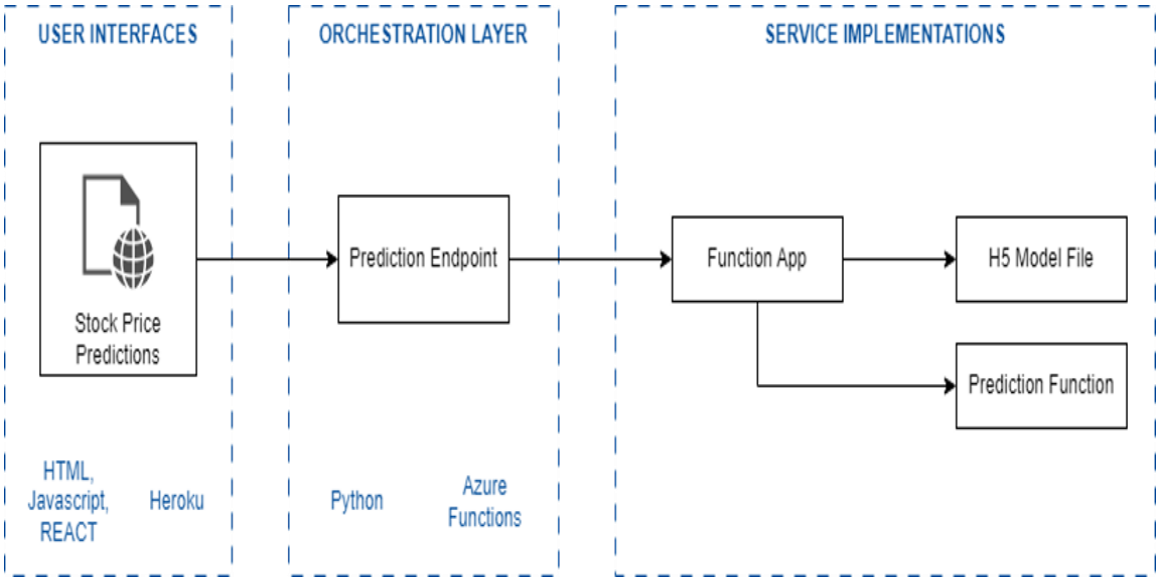

Figure 11. Cloud Backend Architecture Design – Low Level

2.5.3 Cloud Backend Implementation

To deploy the backend in the cloud, we developed a Python application using version 3 to create the API in the Azure Function. We use the Azure Function service to map our API to the Azure cloud. In this API, we define a function called **"get_predictions"** that returns an HTTP response with the expected results (prediction values, prediction graph, and model summary).

This is the expected HTTP response format:

```
result = {
    "status": "ok",
    "msg": "Excelent Job",
    "model_summary" : model_summary["output"],
    "prediction_values" : model_prediction["output"],
    "prediction_graph" : model_prediction["image"]
}

return func.HttpResponse(json.dumps(result), mimetype="application/json", status_code=200)
```

Figure 12. Cloud Backend – API Successful Response

If there is an unexpected error in the execution, this is the expected result:

```
        result = {
            "status": "error",
            "msg": "Wrong Job - You need to pass the 'days' variable to predict the new n days.",
            "model_summary" : None,
            "prediction_values" : None,
            "prediction_graph" : None
        }

        return func.HttpResponse(json.dumps(result), mimetype="application/json", status_code=200)

except Exception as e:
    result = {
        "status": "error",
        "msg": f"Wrong Job - Something happens. Error: {e}",
        "model_summary" : None,
        "prediction_values" : None,
        "prediction_graph" : None
    }

    return func.HttpResponse(json.dumps(result), mimetype="application/json", status_code=400)
```

Figure 13. Cloud Backend – API Failure Response

Figure 14 shows the file structure for deploying a function app to Azure. There are 3 important files:

- **final-model.h5:** This file is the model of our AI project.
- **function_app.py:** In this file, we defined the structure of the Function App in Azure. Furthermore, we defined the API methods that we needed to implement, in our case **“get_predictions”.**
- **requirements.py:** In this file, we defined the python libraries that we need to use in the execution of our python script.

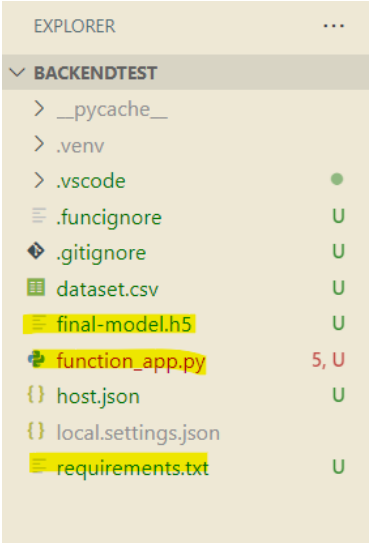

Figure 14. Cloud Backend – Function App structure

Figure 15 shows the response of the "get_predictions" API method. This output displays a JSON format with the prediction values and prediction graph in bytes (base 64) format. This API method is used in the Frontend to display the prediction values and plot the graph.

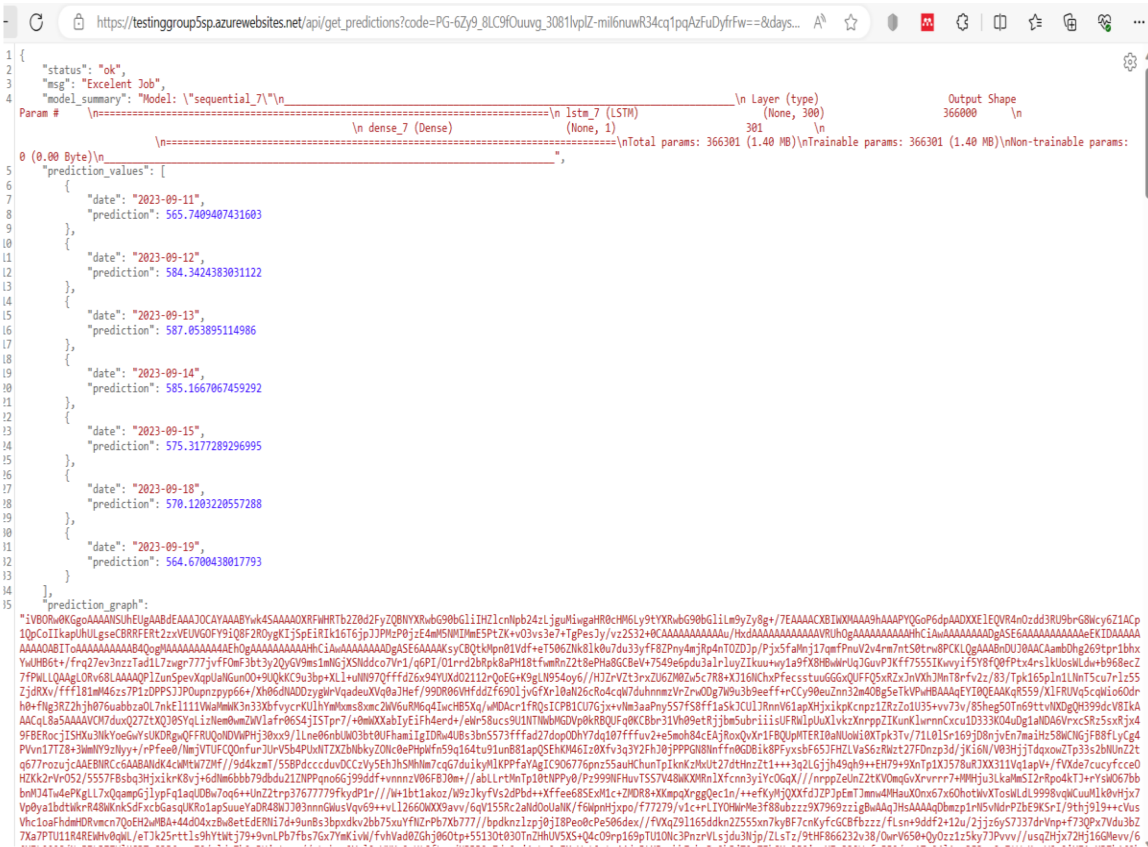

Figure 15. Cloud Backend – API Response

2.6 DataFrame Step

The dataset being used is sourced from Kaggle and is titled “World Stock Prices (daily updating)”.

Source: World stock prices (daily updating). Kaggle. https://www.kaggle.com/datasets/nelgiriyewithana/world-stock-prices-daily-updating/data

This dataset offers a rich historical record of stock prices for many globally recognized brands. The data spans from January 1st, 2000, to the present, which provides a deep and extensive timeline of stock market movements for various global entities.

Key features of the dataset include:

- **Date:** The specific day of the stock price data.
- **Open:** The opening price of the stock on that date.
- **High:** The highest price the stock attained during the trading day.
- **Low:** The lowest price the stock reached during the trading day.
- **Close:** The closing price of the stock on that date.
- **Volume:** The trading volume, indicating the number of shares traded on that date.
- **Dividends:** Information on dividends paid out on that date, if applicable.
- **Stock Splits:** Details about any stock splits occurring on that date.
- **Brand_Name:** The name of the brand or company.
- **Ticker:** The stock's ticker symbol.
- **Industry_Tag:** The industry category or sector the brand is associated with.
- **Country:** The country where the brand is primarily based or operates.

For our project, we are focusing on Costco's stock data extracted from this comprehensive dataset.

2.7 Transformer Step

2.7.1 Feature Extraction Approach

- **Correlation Heatmap Analysis:**

  Prior to feature selection, a correlation heatmap helped us to analyze the relationships between different variables in the dataset. This visual tool helps identify which features are most strongly correlated with the stock's

closing price which is our target variable in stock price prediction models. After careful examination of the heatmap, we made the decision to select ‘Open’, ‘High’, ‘Low’ and ‘Close’ as the primary features for our predictive model, given their strong intercorrelations and direct impact on stock price movements.

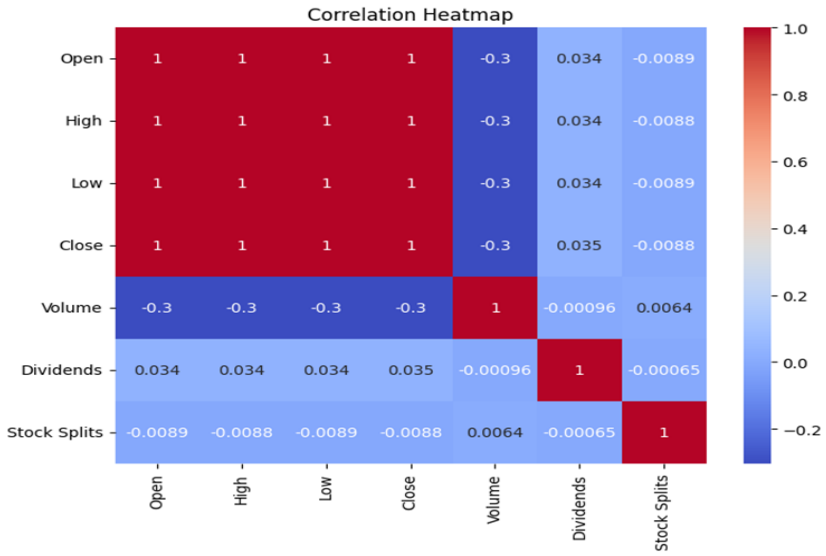

Figure 16. Correlation Heatmap

- **Rationale for Feature Removal:**
  - **Volume Column:**

    The decision to remove the ‘Volume’ column can be justified based on its dependency on real-time events and news sentiments. Stock trading volume can be significantly influenced by factors like tweets from influential figures, news events and statements from stock market experts. Since these elements are external and dynamic, they are not consistently captured in historical data, thus making the ‘Volume’ feature less reliable for predictive modeling in this context.
  - **Other Columns:**

Similarly for other columns like ‘Brand_Name’, ‘Ticker’, ‘Industry_Tag’, ‘Country’, ‘Dividends’ and ‘Stock Splits’, the heatmap analysis indicated a very weaker correlation with ‘Close’ as the target variable.

This approach ensures that the model focuses only on the most relevant features that have a more direct relationship with stock prices. It also helps in avoiding the noise and potential overfitting that could arise from including less relevant or more volatile features.

2.7.2 Data Cleaning Approach

- **Date Parsing and Indexing:**

  We begin by converting the ‘Date’ column into a proper datetime format and setting it as the index. This ensures chronological integrity and enables us to leverage time series functionalities within pandas.

- **Handling Missing Values:**

  Prior to normalization, we checked and addressed any missing values in our selected features. This step is critical to prevent any bias or inaccuracies in the model training phase.

- **Consistency and Integrity:**

  Throughout the cleaning process, we maintain consistency and integrity of the data by ensuring that all operations are uniformly applied to the entire dataset. This uniformity is crucial for the model to learn accurately from the data.

2.7.3 Data Pre-processing Approach

- **Data Normalization:**

  We apply the Min-Max scaler to normalize the data. This step is essential to ensure that the numerical range of our feature data is scaled to a uniform range of [0, 1]. This normalization helps in the acceleration of the model's progress during training and helps in preventing biases towards any outlier values.

- **Train-Test Split:**

  We divide the dataset into training and testing sets with the most recent data withheld as the test set. This is to simulate a realistic scenario where the model predicts future stock movements based on historical data it has not seen before.

- **Window Size:**

  We define a window size of 20 days based on team decision. This window size is considered ideal as it includes enough historical data to detect trends and patterns for model to process.

- **Dataset Structuring for Time Series:**

  The data is structured into a format where the input features are the stock prices of the past 20 days, and the target is the 'Close' stock price of the following day. This transformation is crucial for the LSTM network to understand and learn from the temporal structure of the data.

2.8 Estimator Step: Architecture

Below is the architecture of our model.

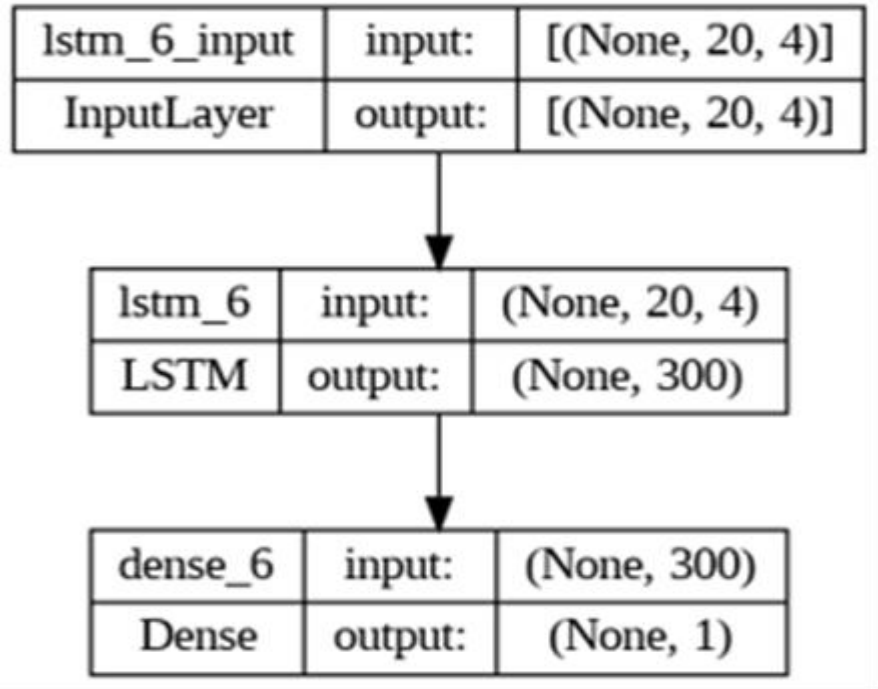

Figure 17. Model Architecture

The Input Layer has dimensions of 20 and 4 where 20 represents the number of time steps i.e. window size in each input sequence and 4 is the number of features for each time step. In this case, it corresponds to the number of columns in the dataset, which are 'Open', 'High', 'Low', and 'Close'. There is only one LSTM layer with 300 units. The LSTM layer is followed by a Dense layer with 1 unit. This Dense layer serves as the output layer and is responsible for producing the final prediction. The Dense layer has only 1 unit because the task is framed as a univariate regression problem, predicting a single value (Close price) for each input sequence. By default, the LSTM layer uses a hyperbolic tangent (tanh) activation function, and the Dense layer has no specified activation function, implying a linear activation.

Some of the strengths of this model are:

- **Long-Term Pattern Recognition:**

  LSTMs excel in recognizing and remembering long-term patterns, a crucial factor for time series data like stock prices. This strength enhances the model's ability to capture underlying trends and patterns in the data.

- **Scalability and Flexibility:**

  The LSTM model's flexibility allows for easy scaling or modification by adding more layers or units, adapting to the complexity of the data. This adaptability enhances the model's capacity to handle varying degrees of complexity, contributing to improved generalization across different datasets and market conditions.

- **Sliding Window Approach:**

  The sliding window approach efficiently predicts stock prices for upcoming periods, aiding investors in decision-making. By capturing temporal dependencies, this approach helps the model generalize well to future time steps, supporting informed decisions based on historical patterns.

Some of the weaknesses of this model are:

- **Input Complexity:**

  The model's complexity increases with more input features, especially using sliding window approaches. Higher input complexity may lead to challenges in generalization, as the model might struggle to extract relevant information from a more complex input vector, potentially impacting its performance on unseen data.

- **Computational Intensity:**

LSTMs can be computationally intensive, particularly with large and complex datasets. The resource-intensive nature of training may limit the model's ability to generalize quickly to new data, especially in real-time scenarios.

- **Hyperparameter Tuning:**

  LSTMs have various hyperparameters that require fine-tuning for optimal performance. Inaccurate hyperparameter tuning may hinder generalization.

- **Overfitting Risk:**

  Due to their complexity, LSTMs can be prone to overfitting if not regularized or trained properly.

In summary, while LSTMs offer powerful tools for time series prediction, careful consideration of their strengths and weaknesses is crucial to achieving robust generalization to unseen data.

2.8.1 Estimator Step: The Learning Algorithm & Cost Function

The **learning algorithm** used in this code is the **Long Short-Term Memory (LSTM)**, which is a type of Recurrent Neural Network (RNN). LSTM is specifically designed to avoid the long-term dependency problem, making it suitable for processing and predicting time series data.

The **cost function** optimized in this code is the **Mean Squared Error (MSE)**. It is a popular loss function for regression problems and is calculated as the average squared difference between the predicted and actual values. The goal of the optimization process is to minimize this error, resulting in a model that can make predictions as close as possible to the actual values.

The **optimizer** used in this code is **Adam**. Adam is an optimization algorithm that can be used instead of the classical stochastic gradient descent procedure to update network weights iteratively based on training data. Adam is a popular algorithm in the field of deep learning because it is straightforward to implement, computationally efficient, has little memory requirements, is invariant to diagonal rescale of the gradients, and is well suited for problems that are large in terms of data and/or parameters.

2.8.2 Estimator Step: Hyperparameters and Fine Tuning

The hyperparameters in this system include:

- **Number of LSTM units:** 300.

  This is the dimensionality of the output space of LSTM layer, or in other words, the number of LSTM cells or neurons in each LSTM layer.

- **Look-back period:** 20. This is the number of previous time steps to use as input variables to predict the next time period.

- **Batch size:** 64.

  This is the number of samples per gradient update, i.e., the number of training examples utilized in one iteration.

- **Number of epochs:** 100.

  An epoch is an iteration over the entire training data provided.

- **Learning rate:** 0.01.

  This is used in the Adam optimizer and determines the speed of learning.

- **EMA momentum:** 0.9.

  This is also used in the Adam optimizer and refers to the exponential decay

rate for the first moment estimates.

The approach used to train the dataset to maximize its generalization capability includes:

- **Normalization:**

  The data is normalized to a range between 0 and 1 using the *MinMaxScaler*. This ensures that all input features have the same scale, which can speed up learning and can lead to better performance.

- **Batch training:**

  The model is trained using a batch size of 64. This means that the model weights are updated after each batch of 64 samples.

- **Epochs:**

  The training process is run for a total of 100 epochs, meaning the whole dataset is passed forward and backward through the neural network 100 times.

- **Adam optimizer:**

  The Adam optimization algorithm is used to update network weights iteratively based on training data. It is computationally efficient and well suited for problems that are large in terms of data and/or parameters.

- **MSE loss function:**

  The mean squared error (MSE) loss function is used, which is suitable for a regression problem. It calculates the average squared difference between the predicted and actual values.

- **Prediction for next 7 days:**

  The model is used to predict the closing price for the next 7 days based on

the last known data. This is done by feeding the last known data back into the model and appending the new prediction to the end of the data.

2.9 Evaluator Step: Training and Testing

2.9.1 Visual Evaluation

The visual assessment involves plotting the actual 'Close' prices against the predicted prices for both the training and test (next 7 days) datasets. The plots serve as an intuitive gauge of the model's predictive accuracy.

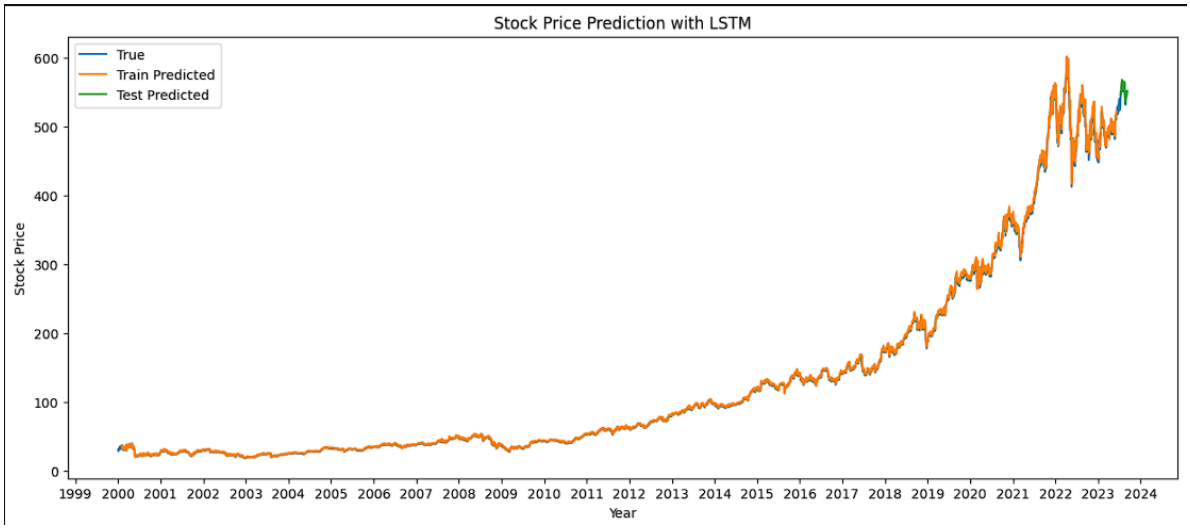

Figure 18. Prediction for the Test Set and Next Week

- **Training Set Visualization:**

  The model's training predictions are plotted with the actual prices. This plot is crucial to visually confirm that the model has learned the underlying patterns in the historical data.

- **Test Set Visualization:**

For the test set, which the model has not seen during training, the predictions are similarly plotted against the actual prices. The distribution of these points indicates how well the model generalizes to new, unseen data.

2.9.2 Quantitative Analysis

Quantitative evaluation is conducted using the Root Mean Squared Error (RMSE) metric, which provides a clear indication of the model's performance in numerical terms.

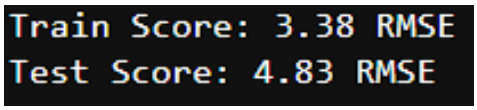

Figure 19. Accuracy Scores for Training and Testing Sets

- **Training Performance:**

  A low RMSE (3.38) on the training set suggests that the model has a good fit to the historical data.

- **Testing Performance:**

  The difference (4.83) between the training and testing RMSE provides insight into whether the model is overfitting to the training data or if it has learned general patterns that apply to the unseen data.

In our case the RMSE is relatively low, but in terms of stock prices this can cause quite a shift due to high volumes of stock being purchased. The model has learned general patterns that apply to unseen data, but also should not be entirely depended on due to the volatility of the real stock market.

3 Future Works

There are some future works that we could consider as improvements to our project:

- **Expanding the Dataset**

  We intend to add more diverse features to our dataset. This includes economic indicators, news about the company, and market sentiment data. These additional features can provide a deeper understanding of factors influencing stock prices, potentially improving the accuracy of our predictions.

- **Real-Time Data Integration**

  Our goal is to develop a system that can handle live data. This means our model will be able to make predictions based on the most current information. Stock markets are dynamic and change rapidly. Real-time data integration will allow our model to adapt to these changes, providing more timely and relevant predictions.

- **Risk Management**

  We aim to create strategies that can identify and reduce the risks associated with using our stock price prediction model. Predicting stock prices is inherently risky. By understanding and managing these risks, we can make our model safer and more reliable for users.

- **User Experience Enhancement**

  We plan to improve the interface of our application to make it more user-friendly and accessible to a wider audience. A better user experience means that more people can use our tool effectively, which is crucial for

both casual investors and professionals.

- **Cross-Industry Application**

  We want to test and adapt our model to predict stock prices for companies in different industries and sectors. This will show how versatile and robust our model is. If it works well across various sectors, it can be a more useful tool for a broader range of users.

4 References

- Bootstrap. (n.d.). Bootstrap. https://getbootstrap.com/
- Brownlee, J. (2021). A Gentle Introduction to Long Short-Term Memory Networks. Machine Learning Mastery. https://machinelearningmastery.com/gentle-introduction-long-short-term-memory-networks-experts/
- Costco Wholesale. (n.d.). Logo & media requests. https://www.costco.com/logo-media-requests.html
- CSS. (n.d.). CSS. https://developer.mozilla.org/en-US/docs/Web/CSS
- Heroku. (n.d.). Heroku. https://www.heroku.com/about
- Hsu, C.-W., & Lin, C.-J. (2003). Stock Price Prediction Using Support Vector Machines. In Proceedings of the International Joint Conference on Neural Networks (IJCNN).
- HTML. (n.d.). HTML. https://developer.mozilla.org/en-US/docs/Web/HTML
- JavaScript. (n.d.). JavaScript. https://developer.mozilla.org/en-US/docs/Web/JavaScript
- Kaggle. (2023). World Stock Prices (daily updating). https://www.kaggle.com/datasets/nelgiriyewithana/world-stock-prices-daily-updating/data
- Li, H., Hong, L., & He, X. (2014). Stock Price Prediction Using Random Forest for the SSE Composite Index in China. In Proceedings of the 2014 International Conference on Machine Learning and Cybernetics.
- Microsoft. (n.d.). Azure Blob Storage. https://azure.microsoft.com/en-

us/services/storage/blobs/

- Microsoft. (n.d.). Azure Functions. https://azure.microsoft.com/en-us/services/functions/
- Microsoft. (n.d.). Azure SQL Database. https://azure.microsoft.com/en-us/services/sql-database/
- Microsoft. (n.d.). What is PaaS?. https://azure.microsoft.com/en-ca/resources/cloud-computing-dictionary/what-is-paas
- Python. (n.d.). Python. https://www.python.org/
- React. (n.d.). React. https://react.dev/
- Techopedia. (n.d.). RESTful API. https://www.techopedia.com/definition/27178/restful-api
- TensorFlow. (2023). LSTM Layer. https://www.tensorflow.org/api_docs/python/tf/keras/layers/LSTM

5 CREDITS, LICENSE, AND REFERENCES

5.1 Credits

Provide any credits here. The following are examples:

Author of the template graphic layout: Hao Lac

5.2 License

State the license granted with your system. For example:

Permission is granted to copy, distribute and/or modify this document under the terms of the GNU Free Documentation License, Version 1.1 or any later version published by the Free Software Foundation; with no Invariant Sections, with no Front-Cover Texts, and with no Back-Cover Texts. A copy of the license is included in the appendix entitled "GNU Free Documentation License".

5.3 References

[1] TAC: Technology Accreditation Canada, https://www.technologyaccreditation.ca/.