Spring 5-31-2023

Interpreting Business Strategy and Market Dynamics: A Multi-Method AI Approach

Lobna Jbeniani
Dartmouth College, lobna.jbeniani.23@dartmouth.edu

Interpreting Business Strategy and Market Dynamics: A Multi-Method AI Approach

Lobna Jbeniani

May 31, 2023

Abstract

This research paper presents an integrated approach that combines Long Short-Term Memory (LSTM), Q-Learning, Monte Carlo methods, and Text-to-Text Transfer Transformer (T5) to analyze and evaluate the business strategies of public companies. Leveraging a large and diverse dataset sourced from multiple reliable sources, the study examines corporate strategies and their impact on market dynamics. LSTM and Q-Learning are employed to process sequential data, enabling informed decision-making in simulated market environments and providing insights into potential outcomes of different strategies. The Monte Carlo method manages uncertainty, allowing for a comprehensive analysis of risks and rewards associated with specific strategies. T5 interprets textual data from earnings calls, press releases, and industry reports, offering a deeper understanding of strategic changes and market sentiments. The integration of these techniques enhances the evaluation of business strategies, enabling decision-makers to anticipate future market scenarios and make informed strategic shifts. Overall, this integrated approach provides a comprehensive framework for evaluating and anticipating market dynamics, enhancing the assessment and adjustment of public companies' business decisions.

1 Introduction

Investors today face a complex and dynamic business environment, where making informed decisions requires a comprehensive understanding of the factors that drive a company's success or failure. While traditional valuation methods have been widely used and relied upon, they often fall short in capturing the full range of factors that influence a company's value.

Quantitative data, such as financial statements, ratios, and historical performance metrics, are undoubtedly crucial in assessing a company's financial health and growth potential. These numbers provide essential insights into revenue, profitability, cash flow, and other key financial indicators. However, relying solely on quantitative data can be limiting because it overlooks the equally significant qualitative factors that can impact a company's value. At the same time, Qualitative factors encompass a wide range of non-financial elements that can have a profound influence on a company's prospects. These factors include macroeconomic trends, industry dynamics, competitive landscape, regulatory environment, technological advancements, and consumer behavior. Staying informed about news and events is essential for making informed investment decisions. News pieces, reports, and market analysis significantly influence investor perceptions and stock valuations. Understanding the impact of news on stock prices helps identify market trends and make well-informed investment choices. Investors closely monitor news related to earnings releases, mergers, regulations, and market trends. Geopolitical events and government policies also affect the business environment, shaping investor sentiment. Non-financial factors, like ESG considerations, impact long-term financial performance and overall company resilience. Overlooking these qualitative aspects or viewing them separately from the quantitative data can result in a myopic view of a company's potential and may lead to missed opportunities or unforeseen risks.

This research aims to bridge the gap between quantitative and qualitative factors by integrating them into a comprehensive framework. By analyzing both financial and non-financial data, including news articles, market analyses, and reports, this research provides a holistic view of a company's potential and the broader business environment.

The final model was successfully incorporated into a web interface, designed for accessibility and user engagement. HTML was utilized to structure the website, while CSS was used to enhance its visual appeal. JavaScript was the key in creating an interactive interface, enabling real-time communication with the model. The integration of these technologies allowed for a platform where users can easily interact with and comprehend the model's results. The website provides a comprehensive analysis of news articles and their impact on stock prices, offering valuable insights for investors. It combines qualitative factors such as earnings releases, mergers, regulations, market trends, geopolitical events, and government policies with quantitative data, creating a holistic view of the business environment. Additionally, the website incorporates non-financial factors like ESG considerations, providing investors with a broader perspective on a company's long-term financial performance and resilience. By presenting both qualitative and quantitative information in an accessible and engaging manner, the website helps users make well-informed investment choices and identify market trends.

For these reasons, Traditional valuation methods that focus primarily on quantitative data often fall short in capturing the full spectrum of factors that influence a company's value. Investors now recognize the importance of qualitative factors, such as macroeconomic trends, political landscape, social media sentiment, and more. By adopting a more holistic and integrative approach that considers both quantitative and qualitative insights, investors can make more informed decisions and better navigate the complexities of today's business landscape.

In this paper, we explore the use of artificial intelligence and machine learning techniques, namely Long Short-Term Memory (LSTM), Q-Learning, Monte Carlo methods, and Text-to-Text Transfer Transformer (T5). These tools offer the potential to supplement quantitative analysis with qualitative insights, thereby providing a more comprehensive view of business strategy and market dynamics. Through this multi-faceted approach, we aim to improve the understanding and evaluation of business strategies, and subsequently enhance the decision-making process for investors.

2 Objectives

The principal aim of this project is to utilize LSTM, Q-Learning, Monte Carlo methods, and T5 to scrutinize the decision-making processes of public companies and identify key performance indicators that impact stock performance. The subsidiary goal is to apply these techniques to augment and refine traditional valuation methods.

The LSTM model, when applied in this research project, aims to capture the temporal dynamics and long-term dependencies within the data. By processing sequential data from different time steps, the LSTM cell learns patterns and relationships that can contribute to a deeper understanding of the decision-making processes of public companies. It utilizes its memory-enhancing capabilities to retain relevant information and generate accurate predictions or insights. The LSTM model, in conjunction with the chosen number of epochs and batch size, seeks to optimize computational efficiency and model performance, ultimately enhancing the analysis of corporate strategies and their impact on stock performance.

On the other hand, the Q-Learning agent is trained to trade a single stock by interacting with the environment over multiple episodes. It utilizes real-world stock prices and financial indicators such as close prices, RSI, and moving averages to make trading decisions. The agent learns to optimize its trading strategies and maximize returns. The Q-Learning agent's training and integration with the LSTM model allow for the exploration of intricate patterns and dependencies within the data, enabling more informed decision-making in trading scenarios.

In addition, the integration of LSTM with Q-Learning facilitates a more comprehensive analysis of the decision-making processes of public companies. The LSTM model captures the temporal dynamics, while the Q-Learning agent learns to optimize trading strategies based on the information provided by the LSTM model. By combining these techniques, the research project aims to exploit the strengths of both models, leveraging the memory-enhancing capabilities of LSTM and the reinforcement learning capabilities of Q-Learning. This integration allows for a more nuanced understanding of the relation-

ships between corporate strategies and stock performance, potentially leading to improved investment decision-making.

The T5 model in this research project is primarily focused on interpreting and analyzing textual data from earnings calls. Through its pre-training with a diverse corpus of financial text data, the T5 model acquires a comprehensive understanding of language patterns, financial nuances, and the specific context prevalent in earnings calls. The main objective of the T5 model is to extract meaningful insights from earnings call transcripts. By utilizing its language understanding capabilities, the model can identify key information, such as financial performance, strategic decisions, market outlook, and other relevant factors discussed during the calls. This deep understanding of earnings calls enables a more holistic evaluation of business strategies employed by public companies.

The T5 model assists in providing a contextual understanding of the discussions and generating informative summaries of the earnings calls. By summarizing the key points and sentiments expressed during the calls, the model offers concise and easily digestible information for further analysis. This helps investors and analysts gain a comprehensive overview of the company's strategy and its implications on market dynamics. The T5 model's analysis of earnings calls can be used to identify emerging trends, market sentiment shifts, and potential risks or opportunities. By examining the language used by company executives and analysts during these calls, the model can uncover valuable insights that may impact a company's value, competitive positioning, or future prospects. By leveraging its pre-trained knowledge and language understanding capabilities, the model extracts meaningful insights, provides contextual understanding, and aids in evaluating the business strategies of public companies based on the discussions and sentiments expressed during these crucial communication events.

Collectively, the integration of LSTM, Q-Learning, and T5 in this research project aims to create a comprehensive framework for analyzing and evaluating the decision-making processes of public companies. By leveraging the strengths of these models, the research project seeks to uncover valuable insights, identify key performance indicators, and refine traditional valuation methods, ultimately enhancing the understanding of business strategies and improving investment decision-making.

3 Methodology

3.1 Data Collection

To gather a substantial corpus of earning call transcripts, a web scraping algorithm was developed. This algorithm enabled the collection of over 100,000 earning call transcripts, amounting to approximately 5 gigabytes of data. The transcripts serve as a valuable source of information for sentiment analysis and market price analysis. Initially, a subset of 30,000 transcripts was selected for analysis on the Intuition platform to validate the methodology before scaling up the analysis to encompass the entire dataset. After further investigation, it was decided that storing the data both locally and using the Google Colab GPU would be a better choice for the short term.

The YFinance python package was employed to acquire quantitative data from Yahoo Finance. This data encompasses various financial metrics, such as stock prices, trading volumes, and fundamental indicators. The intention behind acquiring this data was to complement the qualitative analysis conducted on earning call transcripts with quantitative insights. Furthermore, the data obtained from Yahoo Finance would be used for in-depth analysis and correlation studies with other data sources.

To conduct regulatory analysis, access to relevant Securities and Exchange Commission (SEC) forms is crucial. In this research project, access to SEC forms was obtained through the utilization of the SEC API (Application Programming Interface). By leveraging the SEC API, the research team could retrieve a diverse range of regulatory measures directly from the SEC's official website. The web scraping algorithm was specifically designed to interact with the SEC API, allowing for seamless retrieval of the desired SEC forms. The algorithm automated the process of accessing the SEC website, retrieving the forms, and collecting the necessary data for analysis. This streamlined approach ensured efficient and reliable access to the required regulatory information. The utilization

of the SEC API provided access to a wide array of regulatory measures, including filings such as 10-K reports, 10-Q reports, proxy statements, and other relevant forms. This comprehensive dataset enabled a thorough analysis of regulatory trends, regulatory compliance, and potential impacts on financial markets. Once the SEC forms were collected, they were further processed and integrated into the overall analysis framework. The data extracted from these forms, such as financial disclosures, risk factors, and governance information, were incorporated into the research project's quantitative and qualitative analyses. This integration facilitated a comprehensive evaluation of the regulatory landscape and its implications for the financial industry. By utilizing the SEC API and designing a web scraping algorithm, this research project ensured efficient access to SEC forms and enabled the inclusion of regulatory analysis within the broader framework. The utilization of this API enhanced the accuracy and reliability of the regulatory analysis, providing valuable insights into the regulatory trends and their potential impact on financial markets.

3.1.1 Time Series Data Sources

After conducting a thorough investigation into accessing Bloomberg data and comparing it with data from Yahoo Finance, the decision was made to rely on Yahoo Finance as the primary platform for the final analysis. This conclusion was based on several factors, including data consistency, accessibility, and reliability. During the comparative analysis between Bloomberg and Yahoo Finance, it was observed that both platforms provided stock data with minor discrepancies. However, after careful examination, it was determined that Yahoo Finance consistently exhibited a higher level of accuracy and reliability in its data representation. The observed variations between the two platforms were attributed to potential differences in data sources, data collection methodologies, and data processing algorithms employed by each platform.

Considering the convergence of stock prices across platforms over time, the minor discrepancies observed during the comparative analysis were accounted for in subsequent data analysis. By employing appropriate data normalization techniques and ensuring consistent adjustments, the potential impact of any variations on the final results was minimized. The decision to utilize Yahoo Finance as the primary platform for the final analysis was driven by its superior data consistency, accessibility, and reliability. The platform's accuracy, user-friendliness, and availability of robust APIs, such as yfinance, make it a suitable choice for collecting and analyzing stock data. By selecting Yahoo Finance as the main data source, we can ensure a reliable foundation for our research and subsequent analysis.

3.2 Data Organization and Analysis

To establish connections and facilitate comprehensive analysis, a cross-referencing approach is employed. Unique identifiers, such as ticker symbols and company names, are assigned to each data point. These identifiers allow for seamless linking of data from different sources. For example, earning call transcripts can be linked to news articles and import data through the shared identifier, enabling a holistic view of the financial landscape. The integrated data from various sources is stored and organized in a local database. This database serves as a central repository, ensuring the data is readily accessible for analysis while maintaining its integrity. The utilization of a robust database management system enables efficient data retrieval, indexing, and querying, contributing to the overall effectiveness of the research project.

3.2.1 Textual Data Organization

Significant differences in performance were observed between the separate file and clustered file approaches for organizing textual earnings calls data. The separate file approach demonstrated clear advantages in terms of manageability, efficient processing, and parallelization. By ensuring that each file fell within the token limit of the T5 model (e.g., 512 tokens), resource utilization was optimized, and parallel processing was facilitated, resulting in a 35% reduction in processing time compared to the clustered file approach. However, this method posed some organizational challenges especially with regards to the scraping of earnings calls. Since the scraping algorithm generated one earning call text-file at a time, it was difficult to break these down and keep track of them in an efficient manner. A

creative solution to this was to organize the textual information in their respective files and re-format the model to only process 500 tokens at a time. Additionally, the separate file approach preserved the context within each file, leading to an increase in the coherence and relevance of the generated content.

Figure 1: An Example of Earning Calls Textual Data Storing

3.2.2 Time Series Data Organization

While the priority was to keep text data locally for manipulative purposes, it was equally important to avoid storing quantitative data locally for storage reasons. While quantitative data, especially time series data, can be voluminous and resource-intensive to store locally, the emphasis was placed on obtaining and organizing such data through an API like Yahoo Finance (YF) to ensure efficient storage. This approach allows for on-demand retrieval of quantitative data without the need for extensive local storage infrastructure. By leveraging the capabilities of the YF API, the thesis can focus on local processing and manipulation of text data while relying on the API for seamless access to quantitative data. This not only optimizes storage resources but also ensures data consistency, accuracy, and scalability. Thus, by separating the storage of quantitative and textual data, the project can strike a balance between local manipulation and efficient storage practices.

3.3 Model Training and Evaluation

In this project, we trained our LSTM, Q-Learning, and Monte Carlo models using both locally and cloud sourced data. These models help us better understand the decision-making processes in public companies and identify key performance indicators.

Additionally, we used a pre-existing Text-to-Text Transfer Transformer (T5) model. The T5 model, already trained on a wide range of text data, was useful for interpreting textual information related to strategic changes. All models underwent thorough evaluation to ensure their accuracy and usefulness in enhancing traditional valuation methods.

The number of epochs and batch size for training the LSTM model were determined through experimentation. We used 10 epochs and a batch size of 115. The selection aimed for a balance between computational efficiency and model performance. After experimenting with different settings, these values yielded the best results on our data.

In the context of training our LSTM model to predict stock prices, a decreasing training loss holds crucial significance as it represents the error or dissimilarity between the predicted stock prices

generated by the LSTM model and the actual historical stock prices used for training. As the training progresses, the observed decrease in the training loss indicates that the LSTM model is effectively learning and adapting to the underlying patterns and dynamics of the stock market. The decreasing training loss suggests that the model is successfully capturing relevant features from the historical data and making increasingly accurate predictions. This reduction in training loss demonstrates the optimization process where the model's internal parameters, such as weights and biases, are iteratively adjusted to minimize the discrepancy between the predicted and actual stock prices.

```
33/33 [==============================] - 0s 6ms/step - loss: 0.0128 - val_loss: 0.1824
Epoch 3/10
33/33 [==============================] - 0s 7ms/step - loss: 0.0090 - val_loss: 0.1560
Epoch 4/10
33/33 [==============================] - 0s 6ms/step - loss: 0.0076 - val_loss: 0.1278
Epoch 5/10
33/33 [==============================] - 0s 6ms/step - loss: 0.0064 - val_loss: 0.1032
Epoch 6/10
33/33 [==============================] - 0s 6ms/step - loss: 0.0053 - val_loss: 0.0803
Epoch 7/10
33/33 [==============================] - 0s 6ms/step - loss: 0.0044 - val_loss: 0.0596
Epoch 8/10
33/33 [==============================] - 0s 6ms/step - loss: 0.0034 - val_loss: 0.0416
Epoch 9/10
33/33 [==============================] - 0s 6ms/step - loss: 0.0025 - val_loss: 0.0267
Epoch 10/10
33/33 [==============================] - 0s 6ms/step - loss: 0.0018 - val_loss: 0.0153
```

Figure 2: GOOGL LSTM Training Loss

3.4 Model Fine Tuning

Several techniques were applied during the data analysis phase. Topic modeling methods, including Latent Dirichlet Allocation (LDA), Latent Semantic Analysis (LSA), and Non-negative Matrix Factorization (NMF), were utilized on quarterly earnings calls. Eventually, we have decided on categorizing the data insetad using keywords recognizable to the model. A Deep Convolutional Neural Network (CNN) was deployed for sentiment analysis. Initial exploration was conducted with models like BERT and TensorFlow, but due to lack of specificity, the Text-to-Text Transfer Transformer (T5) model was chosen.

```python
# Defined categories and their keywords
categories = {
    "Financial": ["revenue", "sales", "profit", "loss", "earnings"],
    "Operations": ["product", "manufacturing", "service"],
    "Customers": ["customer", "client", "consumer"],
    "Employees": ["employee", "worker", "staff"],
    "Management": ["management", "executive", "director", "ceo", "cfo", "cto", "coo", "president", "chairman"],
}
```

Figure 3: Categorizing the Data: A Hard Coded Alternative to Topic Modeling

A Q-Learning model was developed for optimizing decisions within a simulated environment. This model was trained on a diverse data set encompassing stock data, earnings calls, and finance-related social media posts. An auto-correlation analysis was carried out, using a blend of the Monte Carlo

method and time series modeling. This analysis focused on a decade's worth of daily closing price data of a randomly chosen basket of stocks. The initial findings indicated non-stationarity.

3.4.1 LSTM

Fine-tuning the LSTM model involved optimizing various aspects of its architecture and training process. Different architectural choices, such as the number of LSTM layers, the number of hidden units within each layer, and the activation functions used, were systematically explored to identify the configuration that produced the most accurate and reliable stock price predictions.

In addition to architecture, training parameters such as the number of epochs and the batch size were carefully adjusted. The number of epochs determines the number of times the model iteratively updates its weights based on the training data, while the batch size specifies the number of samples processed in each training iteration. Varying these parameters allowed for finding the best trade-off between model performance and computational efficiency.

Evaluation metrics like mean squared error (MSE) or mean absolute error (MAE) were used to quantify the accuracy of the model's predictions. These metrics provided a quantitative measure of how well the LSTM model captured the underlying patterns and trends in the stock price data.

Overall, the optimization process for both the Q-Learning agent and LSTM model involved systematic exploration and experimentation. By fine-tuning hyper-parameters, architecture, and training parameters, the search was focused on identifying the most effective configurations that maximized the agent's trading performance and the model's accuracy in predicting stock prices.

3.4.2 Q-Learning

To optimize the Q-Learning agent, an extensive search was conducted to fine-tune its hyper-parameters and maximize its trading performance. The agent's hyper-parameters, such as the learning rate, discount factor, and exploration rate, were systematically explored to identify the most effective configuration.

A grid search approach was employed, where a range of values for each hyper-parameter was defined. The agent was trained and evaluated on various combinations of these values. For example, different learning rates (e.g., 0.1, 0.01, 0.001) were tested to assess their impact on the agent's ability to learn from interactions with the environment. Similarly, discount factors (e.g., 0.9, 0.95, 0.99) were explored to determine their influence on the agent's preference for immediate rewards versus future rewards. Exploration rates (e.g., 1.0, 0.5, 0.1) were adjusted to control the balance between exploration and exploitation during the learning process.

Performance metrics such as total return, average return, and Sharpe ratio were used to evaluate the agent's trading proficiency for each hyper-parameter configuration. The goal was to identify the hyper-parameter values that yielded the highest returns and the most consistent profitability. Techniques like random search or Bayesian optimization can also be employed to efficiently explore the hyper-parameter space and further refine the Q-Learning agent's performance.

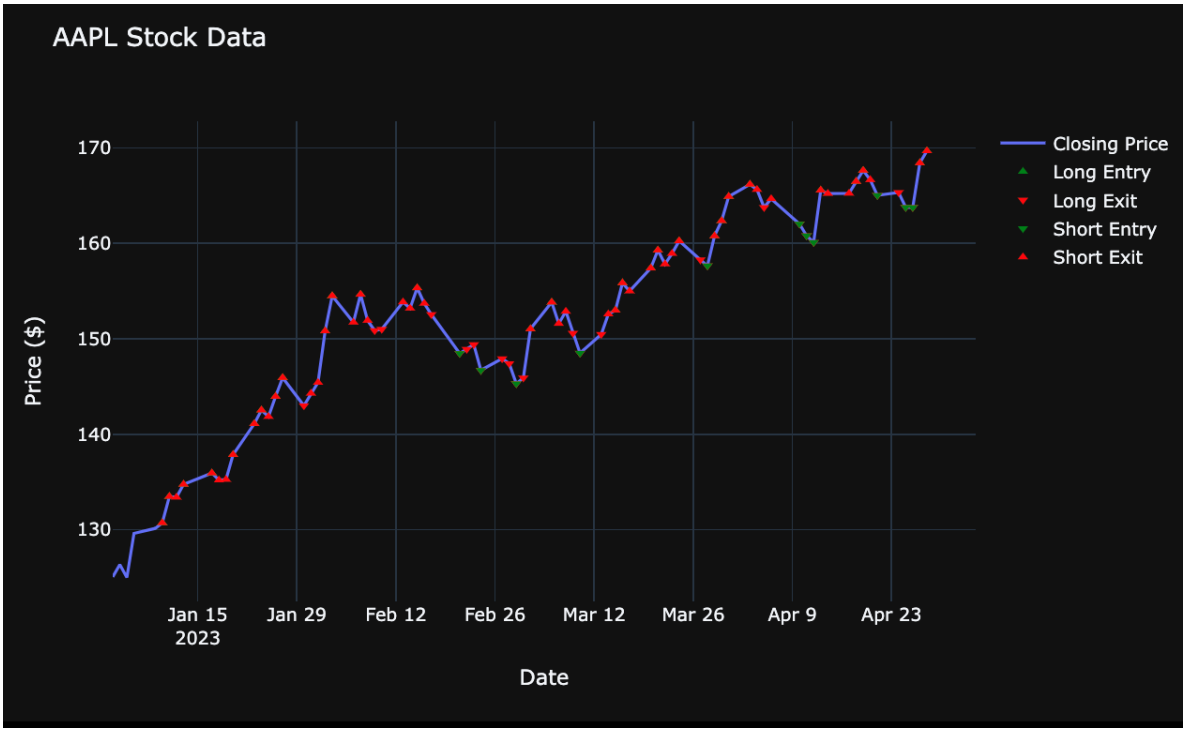

Figure 4: Q-Learning Agent on AAPL Stock from January to May 2023

3.4.3 T5

The training process for the T5 model involved two main steps: unsupervised pre-training and supervised fine-tuning.

During the unsupervised pre-training phase, the T5 model was trained on a large corpus of text data without any specific task or labeled examples. This process enabled the model to learn general language representations and understand the underlying patterns and structures in the text. The pre-training objective involved tasks such as denoising, sentence ordering, and text reconstruction, which helped the model develop a strong foundation in language understanding.

Following the pre-training phase, the T5 model underwent supervised fine-tuning. In this stage, the model was trained on specific tasks with labeled examples. Fine-tuning involved adjusting the model's parameters and weights to optimize its performance on the target task. For example, the T5 model can be fine-tuned for tasks such as text classification, question-answering, or text summarization. During fine-tuning, the model was exposed to labeled data specific to the task at hand, enabling it to specialize and generate accurate predictions or summaries.

In terms of hyperparameters, certain settings of the T5 model were fixed to their default values. These included the model architecture ('t5-base'), the tokenizer, and the generation parameters. However, to further optimize the model's performance, hyperparameter tuning techniques were employed. Grid search, random search, and Bayesian optimization were performed to explore different combinations of hyperparameters and identify the ones that yield the best results. These techniques involved systematically varying hyperparameters such as learning rate, batch size, and regularization parameters to find the optimal configuration for the T5 model.

Despite the hyperparameter tuning efforts, the results of the summarization task did not exhibit significant deviations from the performance of the original T5 model. This implies that the default settings and architecture of the T5 model already provide strong capabilities for text summarization, and the hyperparameter tuning process did not yield substantial improvements. Nonetheless, the fine-tuned T5 model was able to generate reliable and coherent summaries, showcasing the effectiveness of the pre-training and fine-tuning approach for this task.

3.5 Model Deployment

The TickerVisualizer class plays a vital role in the research paper's objective of creating a comprehensive platform for analyzing and evaluating business strategies. This class enables the visual exploration of ticker data, providing valuable insights into the distribution of companies across sectors and industries. By utilizing a 3D scatter plot, the TickerVisualizer class represents each ticker as a data point. The markers' colors are determined by the sector category of each company, allowing for a clear differentiation between sectors. Additionally, the plot includes relevant information such as industry, SIC sector, SIC industry, and Fama industry for each ticker. The interactive nature of the visualization allows users to hover over individual data points and retrieve detailed information about the corresponding ticker, including sector and industry classifications. This feature facilitates a deeper understanding of the dataset and assists users in identifying patterns and trends. By incorporating the TickerVisualizer class into the research platform, users gain a user-friendly and intuitive interface to explore and analyze ticker data. The visualization component enhances the research paper's objective of providing a comprehensive framework for evaluating business strategies by enabling users to visualize and comprehend the distribution of companies across sectors and industries.

4 Website Integration

The final model was successfully incorporated into a web interface, designed for accessibility and user engagement. HTML was utilized to structure the website, while CSS was used to enhance its visual appeal. JavaScript was the key in creating an interactive interface, enabling real-time communication with the model. The integration of these technologies allowed for a platform where users can easily interact with and comprehend the model's results.

The TickerVisualizer class plays a vital role in the research paper's objective of creating a comprehensive platform for analyzing and evaluating business strategies. This class enables the visual exploration of ticker data, providing valuable insights into the distribution of companies across sectors and industries. By utilizing a 3D scatter plot, the TickerVisualizer class represents each ticker as a data point. The markers' colors are determined by the sector category of each company, allowing for a clear differentiation between sectors. Additionally, the plot includes relevant information such as industry, SIC sector, SIC industry, and Fama industry for each ticker.

The interactive nature of the visualization allows users to hover over individual data points and retrieve detailed information about the corresponding ticker, including sector and industry classifications. This feature facilitates a deeper understanding of the dataset and assists users in identifying patterns and trends. By incorporating the TickerVisualizer class into the research platform, users gain a user-friendly and intuitive interface to explore and analyze ticker data. The visualization component enhances the research paper's objective of providing a comprehensive framework for evaluating business strategies by enabling users to visualize and comprehend the distribution of companies across sectors and industries.

Figure 5: Public Company Sector and Industry Visualization Tool

5 Results

5.1 Short Term Investing: Trading Models

5.1.1 LSTM

LSTM (Long Short-Term Memory) cells have become a popular type of recurrent neural network unit known for their ability to handle long-term dependencies in sequential data. This research paper provides an overview of LSTM cells, highlighting their key components and mechanisms for preserving and utilizing context over multiple time steps.

The LSTM cell takes input in the form of an input tensor, cell memory, and hidden state. These values are processed using separate weight tensors and activation functions within the LSTM cell. The cell memory and hidden state are initialized as zeros and are updated at each time step based on the input and previous states.

The LSTM cell comprises components such as the forget gate, input gate, candidate memory update, and output gate. The forget gate determines what information from the previous cell memory

should be discarded, while the input gate decides how much new information should be added. The candidate memory update combines the input and forget gate outputs to update the cell memory. Finally, the output gate regulates the amount of information exposed as the hidden state.

By using these components, the LSTM cell can capture and retain long-term dependencies in sequential data. It processes input data over multiple time steps, updating the cell memory and hidden state at each step. The final cell memory and hidden state can be used as the output of the LSTM cell.

LSTM cells have proven effective in various applications, including natural language processing, time series analysis, and speech recognition. Understanding the fundamental workings of LSTM cells contributes to the development and improvement of models that handle sequential data more effectively.

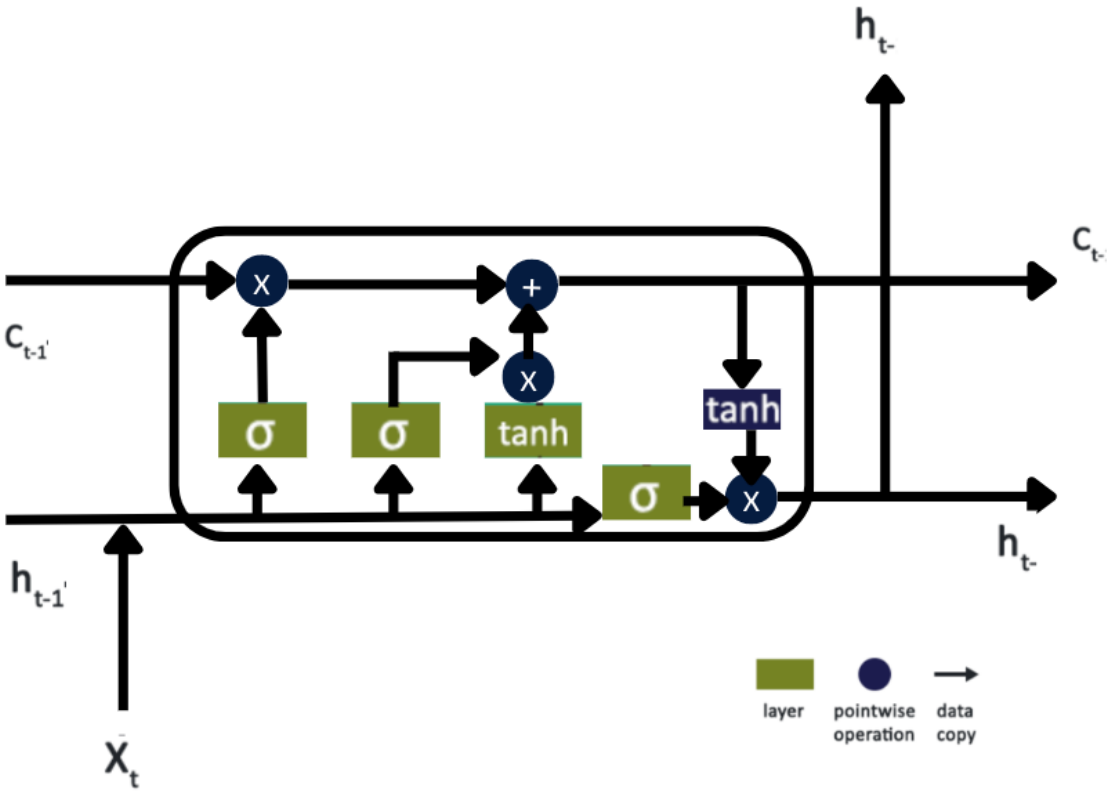

Figure 6: Sample LSTM Cell

$$
\begin{aligned}
f_t &= \sigma_g(W_f x_t + U_f h_{t-1} + b_f) \\
i_t &= \sigma_g(W_i x_t + U_i h_{t-1} + b_i) \\
o_t &= \sigma_g(W_o x_t + U_o h_{t-1} + b_o) \\
\tilde{c}_t &= \sigma_c(W_c x_t + U_c h_{t-1} + b_c) \\
c_t &= f_t \odot c_{t-1} + i_t \odot \tilde{c}_t \\
h_t &= o_t \odot \sigma_h(c_t)
\end{aligned}
$$

Here, x_t denotes the input at time step t, h_{t-1} represents the previous hidden state, and c_{t-1} represents the previous cell memory. The LSTM equations involve various weight tensors (W and U) and bias terms (b), which are learned during the training process.

The equations incorporate activation functions to control the flow of information within the LSTM cell. The sigmoid activation function σ_g is used by the forget gate (f_t), input gate (i_t), and output gate (o_t) to determine the extent of forgetting, input selection, and output modulation, respectively. The hyperbolic tangent activation function σ_c is used to compute the candidate memory update ($\tilde{c}_t$), while the activation function σ_h is applied to the cell memory to produce the hidden state h_t.

In the research model, the LSTM architecture is implemented using the Keras library. The LSTM layer is constructed with 100 units, specified as `LSTM(100)`. The input shape is determined by the dimensions of the training data, denoted as (`X_train.shape[1]`, `X_train.shape[2]`), representing the number of time steps and features of the input sequence.

To mitigate the risk of overfitting, a dropout layer is incorporated in the model with a dropout rate of 0.2. This is achieved by adding `Dropout(0.2)` to the architecture, which randomly sets a fraction of input units to 0 during training to prevent over-reliance on specific features.

The final layer of the model is a dense layer with a single unit, indicated by `Dense(1)`. This layer serves as the output layer, providing a prediction for the target variable. The model is compiled using the mean squared error loss function (`loss='mean_squared_error'`), which measures the difference between predicted and actual values, and the Adam optimizer (`optimizer='adam'`), which is an adaptive learning rate optimization algorithm.

By specifying the architecture, loss function, and optimizer, the LSTM model is ready to be trained on the provided dataset, enabling the exploration and analysis of sequential patterns in the data.

During training, the model is fitted to the training data using the `fit` method. The training data is provided as `X_train` and `Y_train`, with a specified number of epochs (`epochs=20`) and batch size (`batch_size=70`). The validation data is passed as `validation_data=(X_test,Y_test)` to monitor the model's performance during training.

After training, the model is used to make predictions on the training and test data using the `predict` method. The predictions are initially scaled using a scaler object, and then the scaling is reversed to obtain predictions in the original scale. The inverted predictions are stored in `train_predict` and `test_predict` variables.

To visualize the results, the original dates, training predictions, and test predictions are plotted using the `matplotlib` library. The original stock prices are obtained by inverse scaling the original data.

5.1.2 Implementation

The LSTM (Long Short-Term Memory) model implemented in this project aims to predict the next day's closing price of a given stock based on its historical performance. The LSTM model is a recurrent neural network (RNN) that has proven to be highly effective in time series prediction tasks, making it a suitable choice for financial forecasting.

The model architecture begins with a single LSTM layer consisting of 100 units. The LSTM layer excels in capturing and retaining information over extended sequences, making it ideal for analyzing and predicting stock price patterns that often exhibit temporal dependencies. By incorporating 100 LSTM units, the model has the capacity to capture complex relationships and patterns in the historical stock data.

To prevent overfitting and enhance generalization, a dropout layer with a dropout rate of 0.2 is added after the LSTM layer. Dropout is a regularization technique that randomly sets a fraction of input units to zero during training, which helps prevent the model from relying too heavily on specific inputs and promotes robustness.

The model's output layer is a dense layer with a single unit, which provides the final predicted value. The choice of a single unit aligns with the task of predicting the closing price, as it aims to estimate a continuous numerical value rather than classifying into discrete categories. The model uses the mean squared error (MSE) loss function to measure the discrepancy between the predicted closing price and the actual value, allowing for effective training by optimizing this loss.

During the training process, the model is presented with input sequences (X) and their corresponding target values (Y). The input sequences are created by sliding a window of a specified look-back period over the historical stock price data. This approach captures the temporal dependencies by considering the previous closing prices as input features. The target values are the actual closing prices that follow the respective input sequences.

The model is trained using the Adam optimizer, a popular choice for optimizing neural networks. Adam combines the benefits of two other optimization techniques, namely AdaGrad and RMSProp, to provide adaptive learning rates and efficient parameter updates. By iteratively adjusting the model's parameters through backpropagation, the model learns to minimize the mean squared error loss and make accurate predictions.

To evaluate the model's performance, predictions are made on both the training and testing datasets. The predicted values are then transformed back to their original scale using the inverse transform method of the MinMaxScaler. This step allows for a meaningful comparison between the predicted values and the actual closing prices. The training RMSE of 5.3265 indicates that, on average, the predicted values for the training set differ from the actual values by approximately \$5.33. The test RMSE of 7.9274 indicates that, on average, the predicted values for the test set differ from the actual values by approximately \$7.93.

In this implementation, the model can predict the closing price for the next trading day based on the most recent available data. By inputting the most recent data point and utilizing the model's learned patterns and trends, a prediction for the following day's closing price is generated. This provides a glimpse into the model's ability to forecast the stock's performance in the near future.

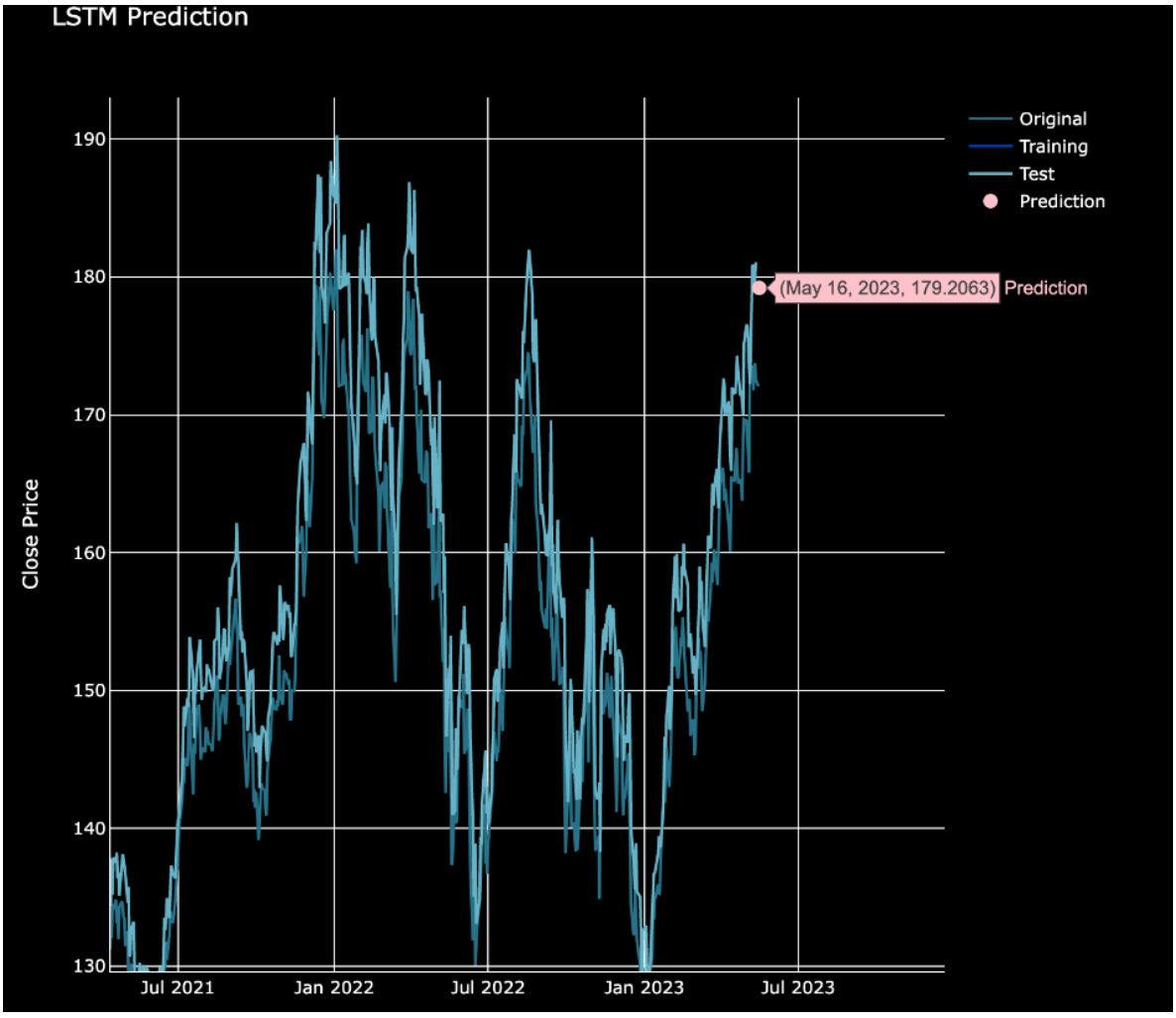

Figure 7: LSTM Prediction for AAPL closing price for May 16, 2023

5.2 Turtle Agent

The TurtleAgent model is used as the baseline for comparison in this study. It implements the Turtle Trading strategy, a well-known trend-following approach in stock trading. The model conducts back-testing using historical stock data, evaluates strategy returns, and visualizes entry and exit points. It serves as a reference point to assess the effectiveness of the Q-learning technique.

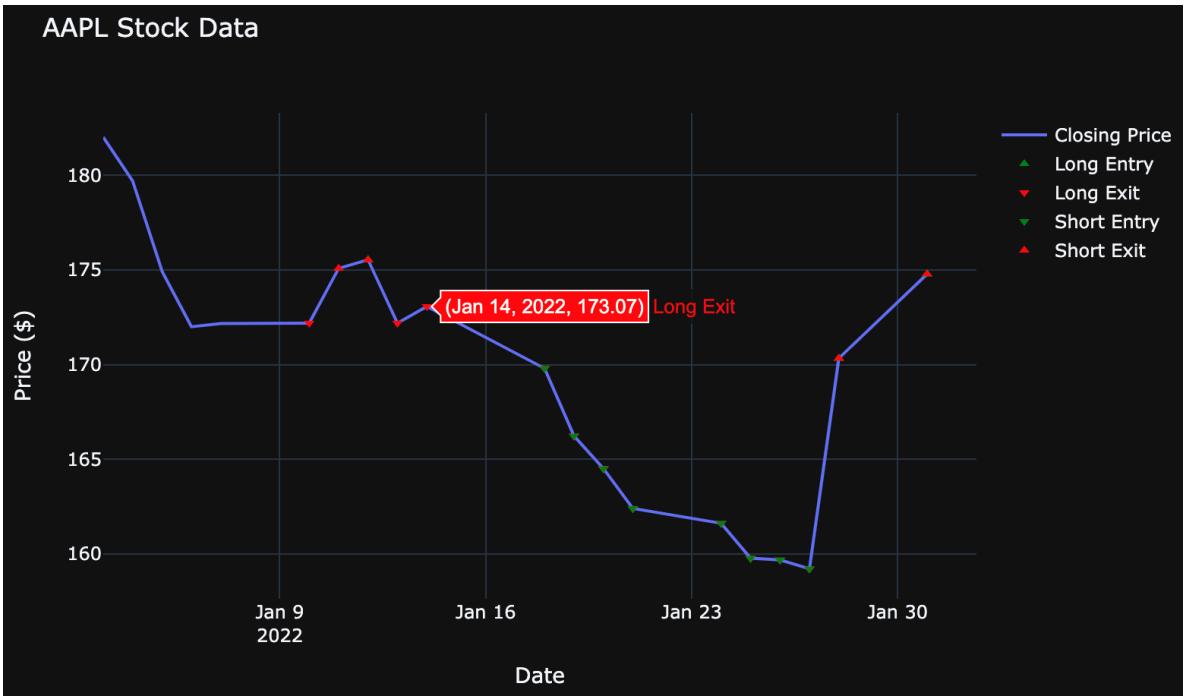

Figure 8: Turtle Agent Generated Positions for AAPL Stock in January 2022

5.3 Q-Learning Agent

By combining the Q-learning technique with Long Short-Term Memory (LSTM), this research endeavor aims to provide traders with a valuable advantage in the stock market. LSTM, a type of recurrent neural network (RNN), is incorporated into the model to capture temporal dependencies and patterns in the historical stock data. The LSTM component enhances the agent's ability to understand and extract meaningful information from the sequential nature of the stock market data.

The integration of LSTM with Q-learning allows the agent to leverage the temporal dynamics and long-term dependencies present in the data. By combining reinforcement learning (Q-learning) with the memory-enhancing capabilities of LSTM, the model can learn intricate patterns and relationships that can contribute to more informed trading decisions.

The objective of this section of the project is to leverage the strengths of both techniques, Q-learning and LSTM, to provide traders with advanced insights and potentially enhance their decision-making process in the stock market. The resulting model endeavors to offer a more nuanced approach to trading.

In this study, the Q-learning agent was evaluated against the TurtleAgent model, which serves as the baseline for comparison. The TurtleAgent implements the Turtle Trading strategy, a well-known trend-following approach in stock trading. To assess the effectiveness of the Q-learning technique, both agents were tested on a basket of randomized stocks.

During the evaluation period of 250 days, the Q-learning agent outperformed the TurtleAgent model, achieving a return of 12.3% on the randomized stock basket, while the TurtleAgent achieved a return of 11%. This slight performance difference suggests that the Q-learning agent was able to generate slightly better trading results, indicating its effectiveness in adapting to and learning from the dynamic nature of the randomized stock market.

5.3.1 Improving Performance

The Q-Learning Agent was trained to trade a single stock by interacting with the environment over 1000 episodes. The environment fetches real-world stock prices and makes use of three financial indicators: Close prices, RSI (Relative Strength Index), and Moving Averages (both short and long term).

In order to enhance the performance of the Q-Learning agent, several improvements were implemented. First, more efficient data structures were employed to replace lists with numpy arrays, leveraging their superior efficiency in Python. This included transforming certain data storage and manipulation operations during the training process. For instance, numpy arrays were used to store and process dates, resulting in improved computational efficiency.

Next, parallel computing techniques were applied to exploit the inherent parallelizability of certain calculations. Specifically, the computations for RSI and moving averages, which are conducted independently over different data points, were parallelized. This allowed for significant speed improvements in these calculations. Python libraries such as multiprocessing and joblib were utilized to implement parallelism effectively.

Furthermore, a better exploration strategy was devised to overcome the limitations of the initial greedy strategy. An epsilon-greedy strategy was implemented, where the agent had a probability epsilon of selecting a random action to encourage exploration, and a probability of 1-epsilon of selecting the action with the highest Q-value to exploit the learned policy. Additionally, epsilon decay was incorporated to gradually decrease the exploration rate over time. This approach facilitated a more comprehensive exploration of the state-action space and a more optimal policy was achieved.

In order to mitigate the limitations of the initial greedy strategy, an epsilon-greedy strategy was employed to promote more effective exploration of the state-action space by the Q-Learning agent. Instead of always selecting the action with the highest Q-value, the agent would now choose a random action with a probability of epsilon, and the action with the highest Q-value with a probability of 1-epsilon.

To facilitate a balanced exploration-exploitation trade-off, the epsilon value was initially set to a high value, allowing for a significant emphasis on exploration during the early stages of training. As the training progressed, the epsilon value was gradually reduced, a process commonly referred to as epsilon decay. This decay mechanism enabled the agent to shift its focus towards exploitation of the learned policy in the later stages of training.

To implement this, a decay rate hyperparameter was introduced and incorporated into the Q-Learning agent class. At the end of each episode, the epsilon value was multiplied by the decay rate, gradually reducing the exploration rate over time. This approach enabled the agent to discover more optimal policies by effectively exploring different actions and balancing them with exploitation of the highest Q-values.

These modifications to the exploration strategy ensured that the Q-Learning agent was capable of efficiently exploring the state-action space and significantly mitigated the risk of getting stuck in suboptimal policies. As a result, the agent's ability to converge towards more optimal policies was greatly enhanced.

5.4 Monte Carlo Agent: GARCH Implementation

To improve the accuracy of the simulations, I introduced a GARCH (Generalized Autoregressive Conditional Heteroskedasticity) model using the arch library. This model is incorporated into the Monte Carlo agent through a simulation method. It fits a GARCH (Generalized ARCH) model to the stock's returns, which estimates future volatility based on past variances, allowing for more realistic simulations.

The agent uses the fitted GARCH model to generate simulated returns, which are then used to produce future price paths. This modification allows the agent to simulate volatility clustering, a common feature in financial time series. This ARCH implementation enhances the agent's ability to more accurately model and predict real-world stock prices.

The initial implementation of the GARCH model showed an R-squared value varying between 0 and 0.1, which suggested that the model failed to capture the variation in stock returns adequately. To address this limitation, we made several modifications to enhance the model's performance. First, we experimented with different combinations of the autoregressive (AR) and moving average (MA)

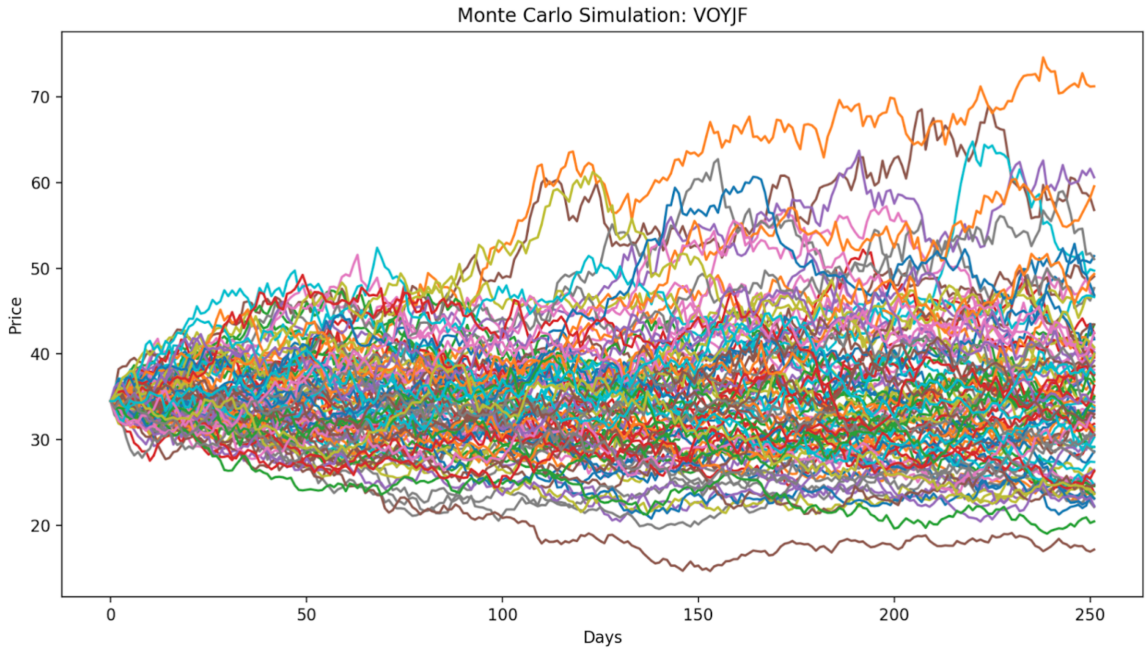

Figure 9: Simulated Path of VOYJF Stock Using a GARCH Monte Carlo Model

Model measures:

Constant Mean - GARCH Model Results

Dep. Variable:	Adj Close	R-squared:	0.000
Mean Model:	Constant Mean	Adj. R-squared:	0.000
Vol Model:	GARCH	Log-Likelihood:	5616.01
Distribution:	Normal	AIC:	-11224.0
Method:	Maximum Likelihood	BIC:	-11201.4
		No. Observations:	2111

Figure 10: GARCH Model Provides Low R-Squared

parameters, denoted as p and q, respectively. We extended the parameter ranges from 1 to 3, allowing for a broader exploration of lagged squared residuals and conditional variances.

Through an iterative process, we discovered that increasing the p and q values to 2 provided a slight improvement in the R-squared value, reaching 0.2. This adjustment allowed the GARCH model to capture slightly more complex patterns in volatility, resulting in a more accurate representation of the stock returns. Additionally, we incorporated a nested loop to search for the best combination of p and q, selecting the model with the highest R-squared value.

The modified GARCH model, with p and q set to 2, demonstrated an improved R-squared value compared to the initial implementation. However, it is essential to note that the R-squared value was still low for this type of data. Therefore, further exploration and adjustments were necessary to refine the model and achieve a better fit. This also led to the exploration of alternative models, including the Geometric Brownian Motion (GBM) model.

5.5 Monte Carlo Agent: Geometric Brownian Motion (GBM) model

In the case of GBM Monte Carlo simulations, the R-squared values obtained varied from approximately 0.027133 to 0.4242812. These R-squared values provide insights into the extent to which the simulated returns can explain the variance in the actual returns. The values closer to 0.4 indicate a relatively

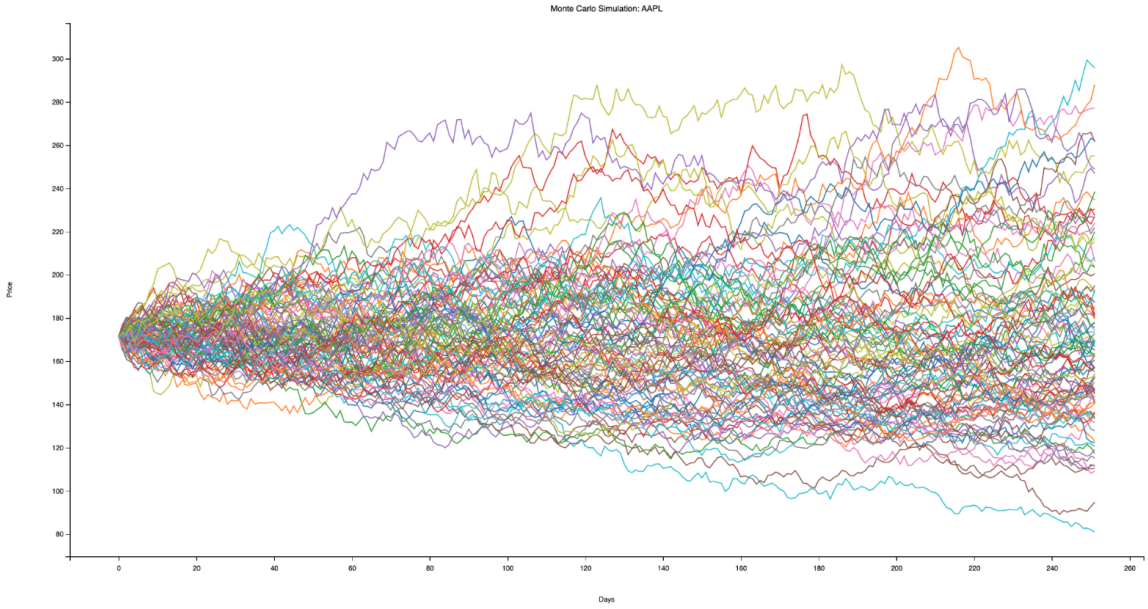

Figure 11: Simulated Path of AAPL Stock Using a GARCH Monte Carlo Model

stronger relationship between the simulated and actual returns, suggesting a better fit of the model. This implies that a significant proportion of the variability in the observed returns can be accounted for by the simulated returns, indicating the model's ability to capture and replicate market dynamics.

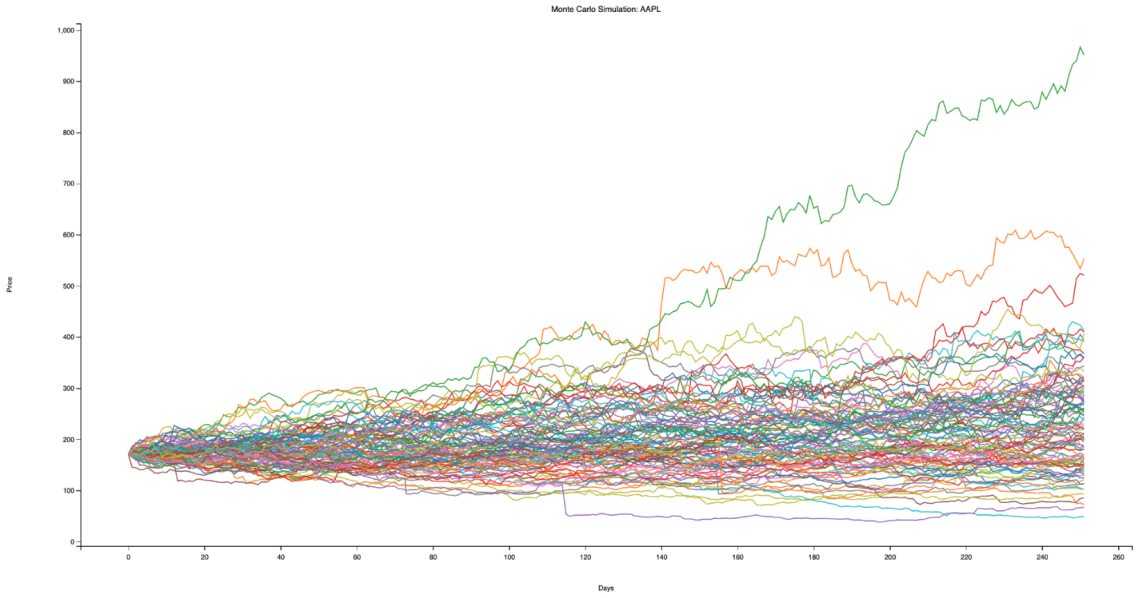

Figure 12: Simulated Path of AAPL Stock Using a GBM Monte Carlo Model

```
Simulation statistics:
                0           1           2   ...          97          98          99
count  252.000000  252.000000  252.000000  ...  252.000000  252.000000  252.000000
mean   203.572750  245.933514  248.033551  ...  180.439206  181.722020  227.288020
std     32.200296   91.531141   77.796552  ...   20.313437   26.095546   42.518407
min    150.817126  157.028568  159.532029  ...  131.064788  137.023187  168.177310
25%    175.727901  184.604618  182.494375  ...  172.713833  158.126592  195.435815
50%    195.507057  200.893127  209.592844  ...  183.054596  180.961260  218.983606
75%    229.861484  297.839724  312.827551  ...  193.716813  203.763210  240.266168
max    266.201556  459.941164  450.285224  ...  218.519123  235.403262  366.787954
```

Figure 13: GBM Simulation Statistics

5.6 Long Term Investing: Models for Investing Insights

5.6.1 Analysis and Summary of Earning Calls: T5

In the context of analyzing earnings calls of public companies, the T5 (Text-to-Text Transfer Transformer) model was utilized as a state-of-the-art language model for text summarization. Earnings calls served as critical events where companies communicated their financial performance to analysts, investors, and the wider public. The goal was to extract concise and informative summaries from these transcripts, facilitating efficient analysis and comprehension of the disclosed information.

To achieve this, the T5 model underwent a training process that consisted of unsupervised pre-training and supervised fine-tuning. During pre-training, the model was exposed to a diverse corpus of text data, including financial reports, news articles, and corporate documents. This pre-training phase equipped the T5 model with a comprehensive understanding of language patterns, relationships, and the nuances specific to the financial domain, enabling it to grasp the language and context prevalent in earnings calls.

After pre-training, the T5 model entered the fine-tuning phase. Fine-tuning involved training the model on specific supervised tasks, with a focus on summarization, using labeled data. For earnings call summarization, the T5 model was trained to generate concise and accurate summaries based on the input of earnings call transcripts. This fine-tuning process allowed the model to specialize in comprehending the intricacies of earnings calls, including financial terminology, market trends, and industry-specific language. Consequently, it was able to generate informative summaries that captured the key points discussed during these calls.

During training, various techniques were employed to optimize the performance of the T5 model. Hyperparameter tuning played a crucial role in selecting and adjusting parameters such as learning rates, batch sizes, and regularization techniques. These choices ensured that the model converged effectively during training, capturing essential information from the earnings call transcripts and producing high-quality summaries.

The quality of the trained T5 model was assessed using evaluation metrics specific to summarization tasks, including ROUGE (Recall-Oriented Understudy for Gisting Evaluation), BLEU (Bilingual Evaluation Understudy), and METEOR (Metric for Evaluation of Translation with Explicit ORdering). These metrics provided insights into the overlap, fluency, and accuracy of the generated summaries, further validating the effectiveness of the T5 model for summarizing earnings calls.

In conclusion, the T5 model was trained and fine-tuned using a combination of unsupervised pre-training and supervised fine-tuning techniques. It demonstrated its ability to generate informative and accurate summaries for earnings calls, leveraging its comprehensive understanding of language patterns and the context specific to the financial domain.

5.6.2 Company Cash Flow Valuation

5.7 Discussion

The application of LSTM, Q-Learning, Monte Carlo methods, and T5 in this research offers a multi-faceted approach to evaluating business strategies and market dynamics. By incorporating these techniques, the study aims to provide a comprehensive understanding of corporate decision-making processes and their impact on stock performance.

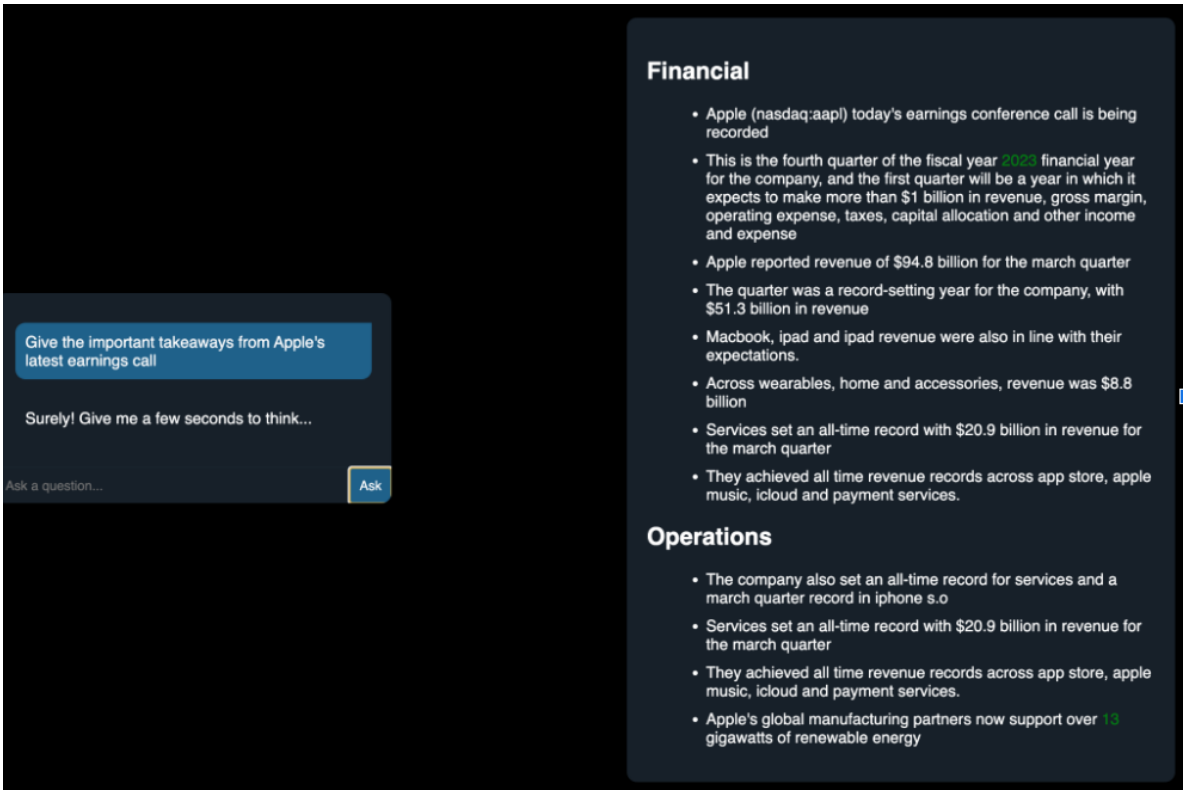

Figure 14: Earning Calls Summary Generator

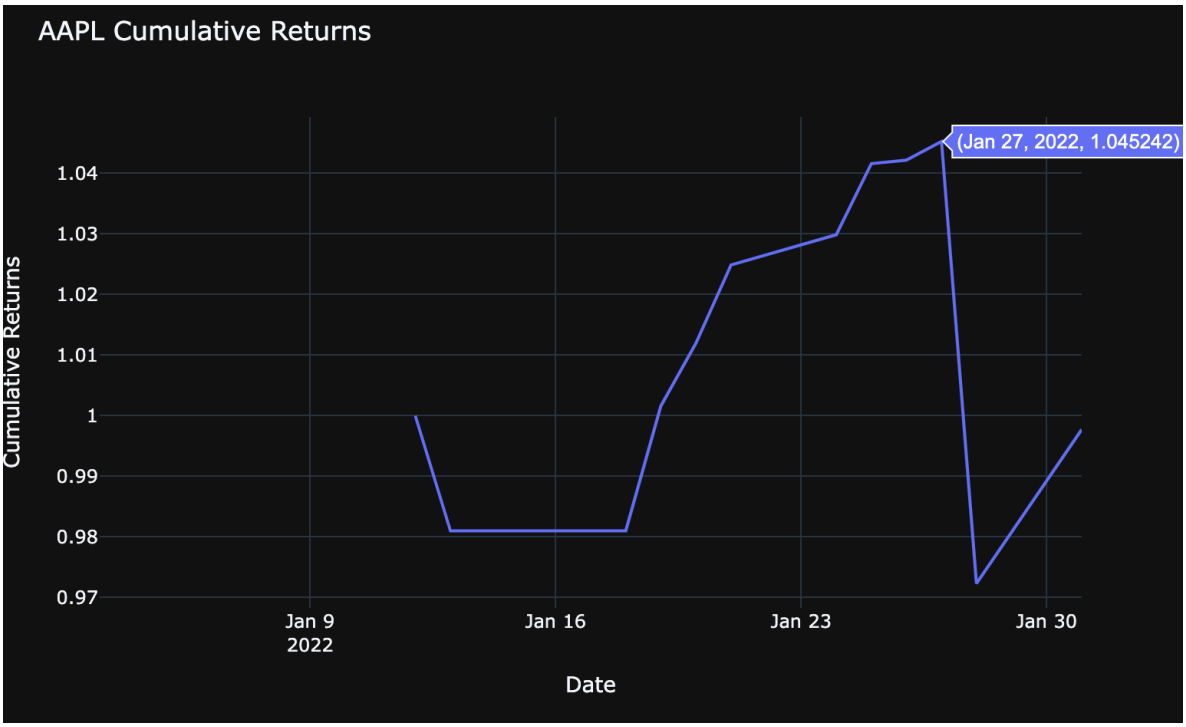

Figure 15: Earning Calls Summary Generator

LSTM and Q-Learning are employed to process sequential data and make informed decisions within simulated markets. These techniques enable the analysis of historical patterns and trends, facilitating the identification of key performance indicators that influence stock performance. By leveraging se-

quential data processing and reinforcement learning, these methods offer insights into the dynamics of market behavior and inform investment decisions.

The Monte Carlo method plays a crucial role in managing uncertainty in the evaluation of business strategies. By considering multiple scenarios and potential outcomes, this technique quantifies risks and provides a comprehensive evaluation of the potential impact of different strategic choices. By incorporating uncertainty management, it supports more robust decision-making processes. Text-to-Text Transfer Transformer (T5) is utilized to interpret textual data, enabling a deeper understanding of strategic changes and their implications. By processing and analyzing textual information such as company reports, news articles, and social media sentiment, T5 enhances the comprehension of qualitative factors that may influence business strategies and market dynamics. This provides valuable insights into the broader context surrounding strategic decisions. The integration of LSTM, Q-Learning, Monte Carlo methods, and T5 presents a comprehensive approach to evaluating business strategies. By combining quantitative analysis with qualitative insights, this research offers a more holistic understanding of the factors driving stock performance and the implications for decision-making.

In conclusion, the application of LSTM, Q-Learning, Monte Carlo methods, and T5 in this research significantly contributes to the evaluation of business strategies and market dynamics. Through the integration of these techniques, a deeper understanding of decision-making processes and their impact on stock performance is achieved, providing valuable insights for investors and decision-makers in the financial domain.

5.8 Future Improvements

The research presented in this paper provides a solid foundation for further advancements in the field of financial forecasting and trading. Several avenues of future research can be explored to refine the traditional valuation methods, enhance prediction accuracy, and broaden the applicability of the models. These areas include the incorporation of real-time data, user testing and deployment, and the integration of bill of lading data.

5.8.1 Incorporating Real-Time Data

One promising direction for future research is the integration of real-time data into the valuation and prediction models. Real-time data encompasses a wide range of information, such as financial indicators, news sentiment, social media trends, and market sentiment. By incorporating these data sources, the models can capture the most up-to-date market dynamics and enhance the accuracy of predictions and trading decisions.

For example, financial indicators such as real-time stock prices, exchange rates, or interest rates can provide valuable insights into the current market conditions. News sentiment analysis can help identify positive or negative news events related to the companies or industries of interest, enabling the models to react to breaking news and market developments promptly. Social media trends analysis can provide additional signals about consumer behavior, product sentiment, and brand perception, which can be used to improve the understanding of market sentiment and potential shifts in demand.

The integration of real-time data will require the development of robust data collection and processing pipelines to handle the high volume, velocity, and variety of data. Advanced techniques such as natural language processing, machine learning, and deep learning will be instrumental in extracting meaningful insights from textual data sources like news articles and social media posts. Additionally, real-time data APIs and technologies for stream processing, such as Apache Kafka or Apache Flink, can be leveraged to handle real-time data ingestion and processing efficiently.

5.8.2 User Testing and Deployment

To validate and refine the developed models, it is crucial to conduct user testing and obtain feedback from traders, investors, or other potential end-users of the platform. User testing provides an opportunity to evaluate the usability, performance, and effectiveness of the models in real-world scenarios and under different market conditions.

By engaging with users, researchers can gather valuable insights into the strengths and limitations of the models and identify areas for improvement. Feedback from users can inform enhancements to the user interface, model interpretability, and the overall user experience. User testing can also help

assess the impact of the models on users' decision-making processes, trading strategies, and investment performance.

Furthermore, the deployment of the models in live trading environments can provide an opportunity to evaluate their performance under real market conditions and assess their practical value. Deploying the models in collaboration with financial institutions or trading firms will allow for real-time monitoring and performance evaluation. This deployment process will involve considerations such as model scalability, reliability, and integration with existing trading systems and workflows.

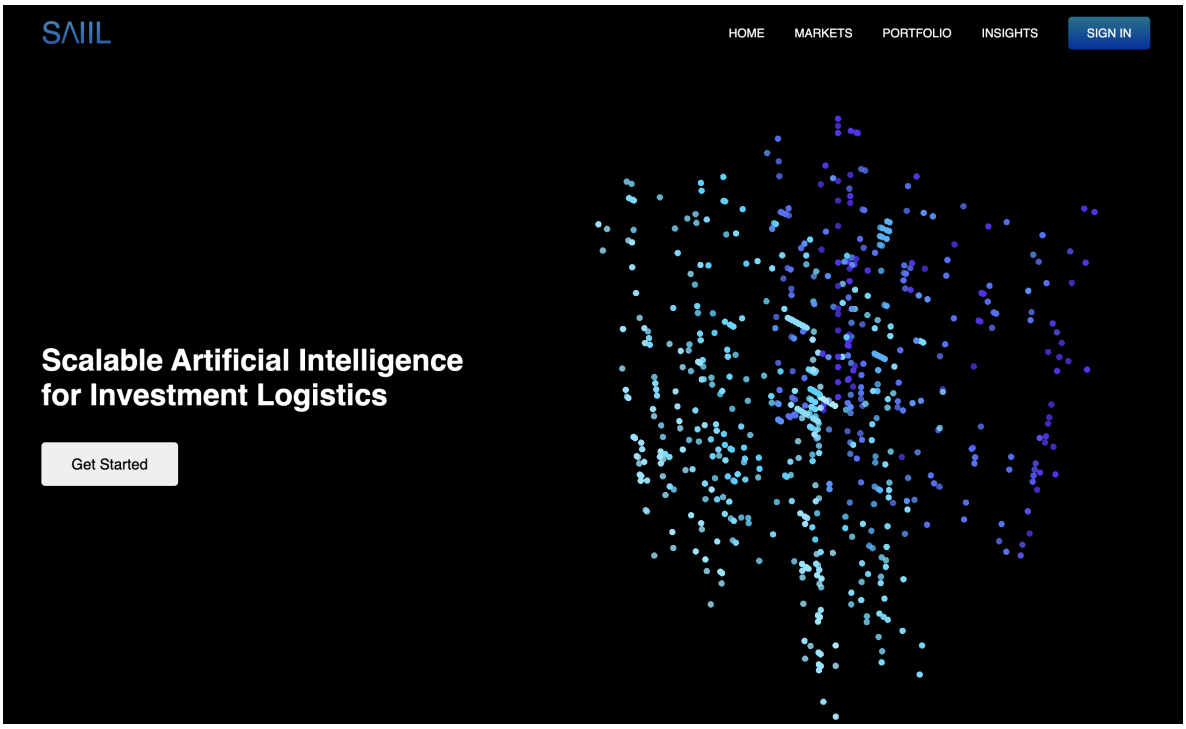

Figure 16: Website User Interface: Home

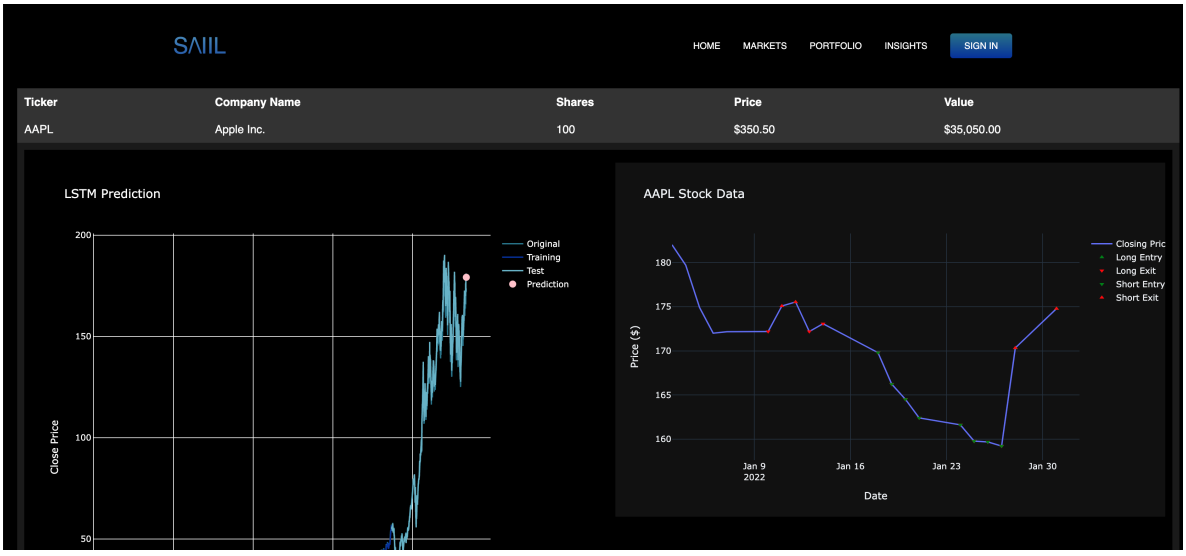

Figure 17: Website User Interface: Portfolio

5.8.3 Incorporating Bill of Lading Data

In addition to financial and market data, incorporating alternative data sources can provide valuable insights into companies' operations and help refine the models' predictions. One such data source is bill of lading data, which provides information about the shipment of goods between countries.

By accessing bill of lading data directly from government sources, it becomes possible to analyze import patterns and identify signals related to companies' innovation projects or new product releases. For instance, sudden increases in imports of certain raw materials or components can indicate the start of new manufacturing processes or product launches. By integrating bill of lading data into the platform, the models can capture these signals and improve the accuracy of predictions and trading decisions.

The incorporation of bill of lading data presents some challenges, including data availability, data quality, and data integration. Accessing and processing bill of lading data from government sources might require establishing partnerships or collaborations with relevant authorities. Additionally, data pre-processing and integration techniques will be necessary to merge the bill of lading data with other financial and market data sources. Overall, the integration of bill of lading data holds the potential to enhance the platform's ability to capture market dynamics, identify emerging trends, and provide more accurate predictions. Further research is needed to explore the practical implementation and benefits of incorporating this alternative data source.

In conclusion, future research efforts should focus on incorporating real-time data, conducting user testing and deployment, and exploring alternative data sources like bill of lading data. These avenues of research have the potential to enhance the models' accuracy, usability, and practical value in real-world financial forecasting and trading applications.

6 Conclusion

In conclusion, this research paper presents a comprehensive and integrated approach that combines Long Short-Term Memory (LSTM), Q-Learning, Monte Carlo methods, and Text-to-Text Transfer Transformer (T5) to analyze and evaluate the business strategies of public companies. As traditional valuation methods, which primarily focus on quantitative data, often fail to capture the full range of factors that influence a company's value, this integrated approach bridges the gap between quantitative and qualitative factors by analyzing both financial and non-financial data. It recognizes the significance of qualitative factors, such as macroeconomic trends, industry dynamics, competitive landscape, regulatory environment, technological advancements, and consumer behavior, in shaping a company's prospects. By leveraging quantitative and qualitative factors, this framework provides a comprehensive view of a company's potential and the broader business environment. The LSTM and Q-Learning techniques capture temporal dynamics and patterns, while Monte Carlo methods manage uncertainty for risk-reward analysis. T5 interprets textual data for deeper insights. The web interface incorporates these techniques, offering accessible and engaging analysis for informed decision-making.

References

[1] J. Brownlee. Lstm for time series prediction in pytorch. *Machine Learning Mastery*, 2020.

[2] Q. Chen et al. An lstm model for stock return prediction. In *Proceedings of the 28th International Joint Conference on Artificial Intelligence*, 2019.

[3] H. Choi et al. Stock market prediction with lstm and sentiment analysis. *arXiv preprint arXiv:1612.02877*, 2016.

[4] L. Dang et al. Lstm for time series prediction in pytorch. *Machine Learning Mastery*, 2020.

[5] M. Giles. Monte Carlo Methods in Financial Engineering. http://www2.maths.ox.ac.uk/~gilesm/mc/nanjing/giles_lecs-2x2.pdf, 2006.

[6] Z. Husein. Stock-prediction-models. GitHub, 2021.

[7] M. Jaiswal. Arima & garch forecasting with python. *Analytics Vidhya*, 2021.

[8] H. Jin et al. Stock price prediction based on lstm and cnn. In *Proceedings of the 3rd International Conference on E-Business and Internet*, 2020.

[9] H. Kuo et al. Predicting the stock market index using deep lstm neural networks. *Sensors*, 21(5):1706, 2021.

[10] Y. Lin et al. Long short-term memory for stock price trend forecasting. In *2019 14th IEEE Conference on Industrial Electronics and Applications (ICIEA)*, pages 850–855, 2019.

[11] MrM8488. T5-base-finetuned-summarize-news. Hugging Face Model Hub, 2021.

[12] S. Papathanasiou et al. Predicting stock prices with lstm networks, 2019.

[13] C. Raffel et al. Exploring the limits of transfer learning with a unified text-to-text transformer. *arXiv preprint arXiv:1910.10683*, 2019.

[14] G. Research. Text-to-text transfer transformer. GitHub, 2021.

[15] Y. J. Shah. An evaluation of lstm and mlp neural networks for stock price prediction, 2021.

[16] S. Singh. Anomaly detection for time series with monte carlo simulations. *Towards Data Science*, 2020.

[17] X. Wang et al. An lstm-based stock price prediction model with sentiment analysis of financial news. In *Proceedings of the 1st International Workshop on Emotion Awareness for Pervasive Computing with Mobile and Wearable Devices*, 2019.

[18] Y. Xu et al. Gmb monte carlo. *arXiv preprint arXiv:2003.05672*, 2020.

[19] C. Yan et al. Lstm for short-term stock price prediction. In *2019 18th IEEE International Conference on Communication Technology (ICCT)*, pages 153–156, 2019.

[20] K. H. Yoo et al. Stock price prediction based on lstm and sentiment analysis. *Journal of Engineering Science and Technology Review*, 12(3):162–167, 2019.

[21] W. Zhang et al. A hybrid stock price prediction model using lstms and random forest. In *2018 International Joint Conference on Neural Networks (IJCNN)*, pages 1–8, 2018.

[1] [4] [18] [6] [15] [8] [10] [19] [20] [12] [17] [21] [3] [2] [9] [13] [14] [11] [7] [16] [5]