Reinforcement Learning in Financial Markets: A Study on Dynamic Model Weight Assignment

Akash Deep, *MS Computational Mathematics*

***Abstract*—This study innovates in financial market forecasting by employing reinforcement learning within a diverse ensemble of models, including Random Forest Regression, LSTM networks, linear regression, and sentiment analysis. Through dynamic, performance-based weight adjustments, our approach shows marked improvement over static strategies. Real-world data testing evidences enhanced prediction accuracy, hinting at potentially more profitable trading decisions. This work underscores the untapped potential of reinforcement learning for optimal ensemble model management in the ever-changing financial landscape.**

***Index Terms*—Ensemble Reinforcement Learning, Random Forest Regression, Long Short Term Memory (LSTM), Advantage ActorCritic (A2C) Algorithm, Financial TimeSeries Analysis**

I. Introduction

Financial markets are complex adaptive systems influenced by an interplay of numerous factors - both quantitative and qualitative. Traditional approaches to predicting stock market behavior often rely on single-factor models, including technical analysis, fundamental analysis, or quantitative methods. However, these singular approaches may not fully capture the multifaceted nature of financial markets. In our previous work, " A Multifactor Analysis Model for Stock Market Prediction", we proposed a comprehensive model that combines technical analysis, fundamental analysis, machine learning, and sentiment analysis (TFMS Analysis) to address this complexity. [1]

While the TFMS Analysis model improved the accuracy of future stock price predictions, there is scope for further enhancement. Static weights assigned to each factor may not optimally represent the fluid nature of financial markets, where the significance of various factors can shift rapidly with changing market conditions.

In light of this, we propose integrating reinforcement learning with an ensemble of predictive models to predict financial market trends. Each model in the ensemble uses a distinct technique ranging from Random Forest Regression and Long Short-Term Memory (LSTM) networks to linear regression and sentiment analysis. Our innovation lies in the application of reinforcement learning to assign dynamic weights to these models in real-time. This adaptability allows the ensemble to react to evolving market conditions, adjusting the relative importance of different models based on the rewards and penalties determined by their predictive performance.

Akash Deep is with Department of Graduate Studies, Texas Tech University Lubbock, TX, 79409, USA, e-mail: akash.deep@ttu.edu (Corresponding author).

We hypothesize that this dynamic, adaptive approach can outperform static weighting strategies and yield superior prediction accuracy. Specifically, we explore the potential influence of sentiment analysis on popular stocks, suggesting that public sentiment might significantly influence these stocks' prices, which may require increased weighting during certain market conditions.

This paper presents our approach, the experimental setup, and the results of our experiments on extensive real-world financial market data. We believe this work opens up new horizons in financial forecasting, showcasing the potential of reinforcement learning as a valuable tool for efficient ensemble model management within the volatile landscape of financial markets. Our findings further the understanding of the multifaceted nature of financial markets and provide a potential pathway for more accurate and profitable trading decisions.

II. Literature Review

The use of machine learning techniques for financial market predictions has been a topic of interest for many years. Various models, ranging from simpler ones such as linear regression to more complex ones like neural networks, have been tested for their ability to predict stock prices with varying degrees of success.

A. Ensemble Learning in Financial Markets

The concept of ensemble learning, where multiple models are combined to make predictions, has been widely studied in the context of financial markets. The primary motivation behind ensemble learning is that a group of models, collectively, often provide more accurate and robust predictions than any individual model. Existing literature demonstrates the potential benefits of ensemble learning in financial markets. [2] conducted an assessment of ensemble learning and classification systems applied to financial data classification tasks such as corporate bankruptcy prediction and credit scoring. Their research showed that ensemble classification systems, specifically AdaBoost, outperformed existing models, providing evidence of the effectiveness of ensemble systems in financial contexts. However, most of these approaches typically assign static weights to different models, which can be sub optimal in the dynamic context of financial markets. Our work aims to address this limitation by introducing a reinforcement learning-based approach for dynamic weight assignment in the ensemble of predictive models.

B. Reinforcement Learning in Financial Markets

Reinforcement learning, a type of machine learning where an agent learns to make decisions by interacting with its environment, has also been applied to financial markets. The agent learns a policy, which is a function mapping from states to actions, based on the rewards or penalties it receives for its actions. Previous studies have shown that reinforcement learning can effectively capture the dynamic nature of financial markets and adapt to changing market conditions . [3] .However, the application of reinforcement learning to dynamically assign weights in an ensemble of predictive models remains a novel and relatively unexplored concept.

C. Sentiment Analysis in Financial Markets

The impact of public sentiment on stock prices has been a subject of considerable interest. The proliferation of social media platforms and online financial news has provided a wealth of data from which to quantify public sentiment, allowing it to be incorporated into predictive models. Existing research indicates that sentiment analysis can provide valuable insights into market trends, especially for highly publicized stocks, [4]. However, identifying the optimal way to incorporate sentiment analysis with other predictive factors is still a subject of ongoing research.

Our work extends these lines of research by integrating ensemble learning, reinforcement learning, and sentiment analysis into a new approach for financial market prediction.

III. Data Preprocessing

Data preprocessing is an essential step in our research as it prepares the raw data for further processing and analysis. It involves transforming the data into a format that would be more easily and effectively processed for the purpose of the user. In the case of our study, the data sources include historical stock price data and news articles. These data are processed differently based on their inherent characteristics.

A. Stock Price Data

The raw stock price data is in CSV format, which includes various fields like date, open, high, low, close, adjusted close, and volume. The following preprocessing steps are performed on this data:

• Data Loading: The first step is to load the raw stock price data into a Pandas DataFrame, a 2-dimensional labeled data structure with columns of potentially different types.

• Date Conversion: The next step is to convert the date column into numerical form, as dates, being non-numeric, can't be used directly in our models.

• Feature Selection: Feature selection is the next step, where we select the necessary variables for our models. For example, for the Random Forest and Linear Regression models, we only consider the date and adjusted close price. However, for the LSTM model, we also take into account the high, low, open, and volume data.

• Data Splitting: The preprocessed data is split into training and testing sets. The training set is used to train our models, while the testing set is used to evaluate the models' performance.

We can now use the training data to train our models and the testing data to evaluate the models' performance.

B. News Articles

For the sentiment analysis, we use news articles related to the stock of interest. The preprocessing steps here are:

• Data Extraction: For sentiment analysis, we use news articles related to the stock of interest. The first preprocessing step here is to extract news titles from the data obtained using the News API.

• Sentiment Analysis: Each news title is then processed by a sentiment analysis function, which assigns a sentiment score. This score is then used as an additional variable in our models.

C. Normalization

We perform an additional preprocessing step for the LSTM model, which is sensitive to the scale of the input data. We normalize the stock price data to a range between 0 and 1 using the MinMaxScaler function from the sklearn.preprocessing module. This normalization ensures that large variations in stock prices do not adversely affect our model's performance.

D. Data Shaping

For the LSTM model, we also need to transform our data into a format suitable for time-series prediction. Therefore, we shape our dataset into a 3D array [samples, time steps, features] as required by LSTM. Our careful attention to data preprocessing ensures that our models receive high-quality, relevant input, a crucial factor for the reliability and accuracy of the predictions generated by the models.

In the next section, we discuss the methodology and the models used for each part of the said tasks.

IV. Methodology

Our ensemble incorporates four predictive models, each applying a different approach for stock market prediction. These models include:

A. Ensemble of Predictive Models

Our ensemble consists of four predictive models, each using a different method for stock market prediction. These include:

1) Random Forest Regression (RFR): In this study, the Random Forest Regression (RFR) model is employed as a key element of our ensemble technique due to its established efficacy for regression tasks. The RandomForestRegressor class from the scikit-learn library is utilized for its effectiveness and adaptability.

The RFR model is parameterized with a user-defined number of decision trees (specified by the n_estimators hyperparameter) and a minimum sample leaf parameter (min_samples_leaf), dictating the least number of samples required at a leaf node. The optimal selection of these parameters is instrumental in enhancing the model's performance.

Data Preprocessing The first step involves loading the CSV dataset, which encompasses stock market data. Following this, the date column is converted to an ordinal format, a numerical representation that permits the model to handle temporal information. We also introduce a 'Daily Return' feature that calculates the percentage change in the 'Adjusted Close' price, providing another dimension for the model to learn from.

The dataset is then partitioned into a feature set X, inclusive of all columns barring 'Adj Close', and a target set y, embodying 'Adj Close'. These are further segregated into training and testing sets, maintaining an 80-20 split, ensuring a reserve of unseen data for evaluating the model's performance.

To address missing data points, we deploy the SimpleImputer class from scikit-learn. This step guarantees that our model is fed with clean, uninterrupted data, essential for effective training and credible predictions.

Hyperparameter Tuning Hyperparameter tuning of our RFR model, specifically for the n_estimators and min samples_leaf_parameters, is accomplished via the GridSearchCV method. This exhaustive search across a specified range of parameter values aims to discern the combination that optimizes model performance, evaluated using the negative mean squared error as the scoring criterion. The search is conducted over five folds of the data to guarantee a robust estimate of model performance.

Results and Discussion Upon tuning, the refined RFR model is fitted to the training data and applied to make predictions on the test set. The model's performance is appraised using the mean squared error (MSE) metric, juxtaposing the predicted stock prices with the actual prices.

In addition to the scatter plot, a histogram of residuals (the discrepancies between the real and forecasted values) is generated to analyze the distribution of errors committed by the model. These plots help in evaluating whether the errors are normally distributed and centered around zero, a basic assumption for regression models.

Furthermore, we create a feature importance plot to understand which features are most influential in the model's predictions. This can provide valuable insights into the underlying structure of the data and the model's decision-making process.

We also visualize a feature importance plot to identify which attributes exert the most influence on the model's predictions. These visualizations, coupled with the scatter plot of actual vs predicted values, render a comprehensive view of the model's performance and the factors influencing its predictions.

2) Long Short Term Memory (LSTM: To better capture the temporal dependencies in stock price data, we utilize Long Short-Term Memory (LSTM) networks as part of our ensemble. LSTM is a type of Recurrent Neural Network (RNN) that is capable of learning long-term dependencies, making it particularly suitable for time-series data. We implement our LSTM model using the Keras library. The preprocessing steps involve loading the data, extracting the 'Close' price, and then normalizing the data using the MinMaxScaler class from the sklearn library. This normalization step scales our stock price data between 0 and 1, helping to improve the stability and performance of our LSTM network. We split our data into a training set, comprising 80% of the data, and a

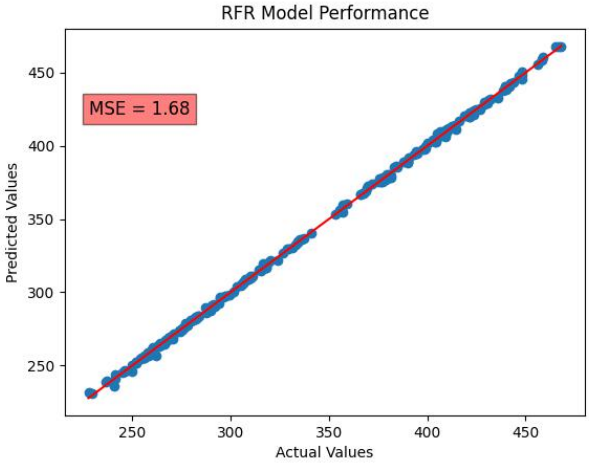

Fig. 1. Scatter plot illustrating the RFR model's performance. The MSE is also embeded in the plot.

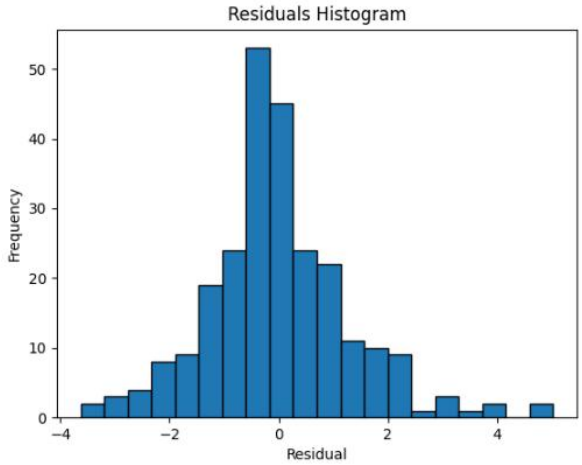

Fig. 2. Histogram of residuals displaying the distribution of errors committed by the RFR model.

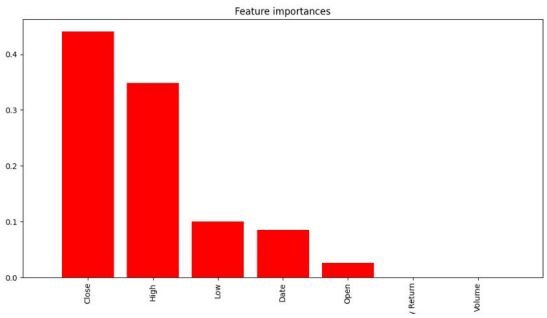

Fig. 3. Feature importance plot demonstrating the significance of each attribute in the RFR model's predictions. Volume might be an essential feature for models dealing with short-term trades or options trading.

test set, containing the remaining 20%. Each dataset is then transformed into 3D arrays suitable for LSTM training, with a lookback period of 1 day. This transformation process involves creating a dataset where the feature (X) at a given time (t) is the stock price at time (t), and the label (Y) is the stock price at the next day (t + 1). The LSTM model itself is a Sequential model with two LSTM layers and a Dense output layer. The LSTM layers are defined with 50 units each, and the 'return sequences' parameter is set to True for the first LSTM layer to allow sequences to pass through, aligning with the structure of our input data. The model uses the 'mean squared error' loss function and the 'adam' optimizer during the training phase. The model is trained on the training data for 100 epochs with a batch size of 32. Once trained, the model is used to predict the 'Close' prices on both the training and testing datasets. These predicted values are then denormalized to their original scale using the MinMaxScaler instance fitted earlier. The performance of the LSTM model is evaluated using the mean absolute error (MAE) metric on both the training and testing datasets. The predicted and actual 'Close' prices are plotted for visual comparison, further providing insights into the model's predictive capability. Furthermore,

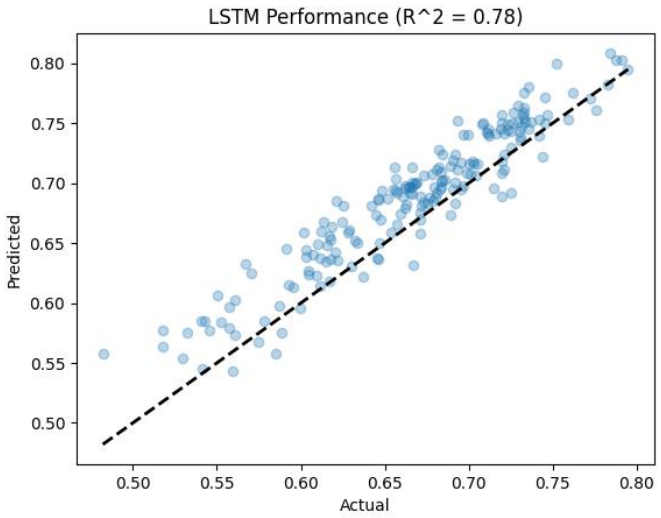

Fig. 4. Scatter plot showing the performance of the LSTM model.

the model is used to predict the 'Close' price for the following day. The most recent 'Close' price is extracted, normalized, reshaped into a 3D array, and fed into the model for prediction. The output of this prediction is denormalized to produce the final forecasted 'Close' price. This LSTM model forms an essential part of our ensemble model, helping to capture temporal patterns in the data.

3) Linear Regression: Linear regression is a fundamental statistical and machine learning method. It is used to understand the relationship between two variables, in this case, the date and the closing price of the stock. The relationship is represented as a straight line (hence 'linear') equation that predicts a dependent data variable as a function of an independent variable.

For our linear regression model, we utilize the Scikit-Learn library, which provides a variety of machine learning

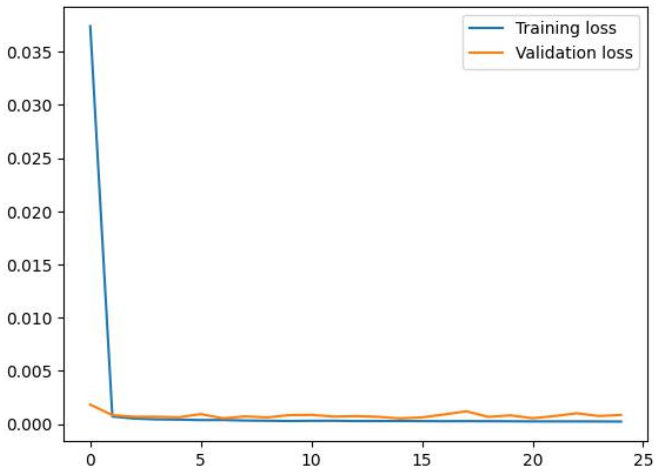

Fig. 5. Plot showing the Training and Validation Losses- We think validation loss is constant because the model has already learned as much as it can from the data, and additional training is not resulting in any significant improvement.

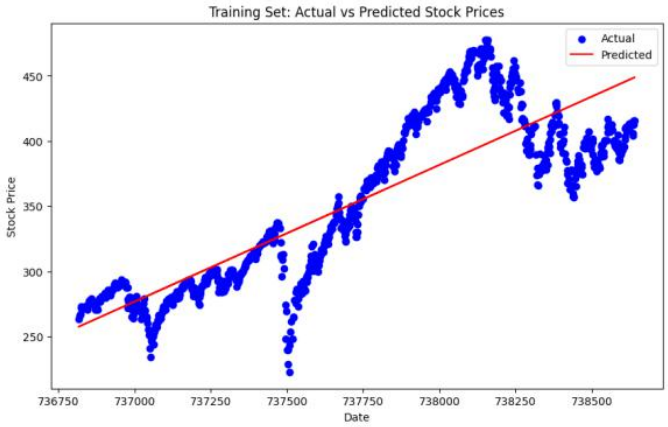

Fig. 6. Scatter plot of actual vs. predicted stock prices for the training set using the Linear Regression model. The red line represents the model's predictions, while the blue dots represent the actual stock prices. The plot demonstrates how well the model has learned to predict stock prices from the training data.

algorithms that can be conveniently called as needed. We first load our stock data and convert the 'Date' column into ordinal values so that they can be utilized in the linear regression model. We then split the dataset into feature (X) and target (y) sets. The 'Date' is our feature, and the 'Close' price is our target. The data is then further divided into training and testing sets using Scikit-Learn's train test split function.

We initialize a Linear Regression model and fit it to our training data. This model essentially learns the relationship between the date and the closing price during the fitting process. After the model has been fitted, it can then be used to predict the closing price based on a given date. We use this property to predict the stock prices on our testing set.

We also print the intercept and coefficients of the model, which represent the point at which the line crosses the y-axis and the slope of the line respectively. This information can be used to manually calculate predicted values using the formula y = mx + c, where m is the coefficient, x is the date, and c is

the intercept.

Finally, we calculate the mean squared error of our predictions, which gives us a measure of how well our model performed. A lower mean squared error indicates a better fit to the data. We also provide visualizations of the actual vs predicted stock prices for both the training and testing sets. This helps us visually assess the performance of our model.

Linear regression, despite its simplicity, can be a crucial component in an ensemble model for financial prediction. It's computationally efficient and highly interpretable, making it easy to implement and understand. It's adept at capturing linear trends and relationships within the data, a common occurrence in financial time series. [5]

Additionally, serving as a baseline, it can offer a reference point for the performance of more complex models. Furthermore, in an ensemble approach, linear regression can complement more sophisticated models like Random Forest and LSTM, which might overfit or miss simpler patterns. So, the incorporation of linear regression can provide a balance, helping to improve the overall robustness and generalization of the ensemble model.

4) Sentiment Analysis: In addition to price data, we consider sentiment analysis of news articles as a potential predictor of stock prices. This is based on the hypothesis that news sentiment can influence investor behavior, which in turn could affect stock prices. We conduct sentiment analysis on news articles related to the stock using the TextBlob library. It was shown by [6] that TextBlob can be used to generate a high-level sentiment analysis score that can be used to make decisions in the stock realm based on news articles.

We use the News API to fetch relevant news articles about the stock. The request specifies the stock symbol, the language ('en' for English), the sort order (relevancy), and the page number. We extract the 'title' from each news article and store these in a list. Next, we perform sentiment analysis on each news title. TextBlob's sentiment analysis function returns a polarity score in the range -1 to 1 for each text, where -1 indicates a negative sentiment, 1 indicates a positive sentiment, and 0 indicates a neutral sentiment. We classify each news title as 'positive', 'negative', or 'neutral' based on its polarity score. The sentiment analysis results are stored in a pandas DataFrame, which contains the news titles and their corresponding sentiment labels.

In addition to categorical sentiment labels, we also compute a sentiment score for each news title, which is a continuous measure of sentiment. We calculate an average sentiment score across all news articles, which serves as a summary metric of the overall sentiment. This sentiment analysis forms a crucial part of our predictive model, capturing the influence of news sentiment on stock prices.

B. Reinforcement Learning for Dynamic Weight Assignment

The key innovation of our approach is the use of reinforcement learning to dynamically assign weights to the models in the ensemble. The reinforcement learning agent receives the predictions of the four models as input and outputs a weight for each model. The weighted predictions are then combined to produce the final prediction.

In order to adaptively allocate weights to different financial models, a reinforcement learning-based approach was employed. This approach treats the problem of dynamic weight assignment as an interactive sequential decision-making task, utilizing the state-of-the-art Advantage Actor-Critic (A2C) algorithm for learning an optimal policy. [7]

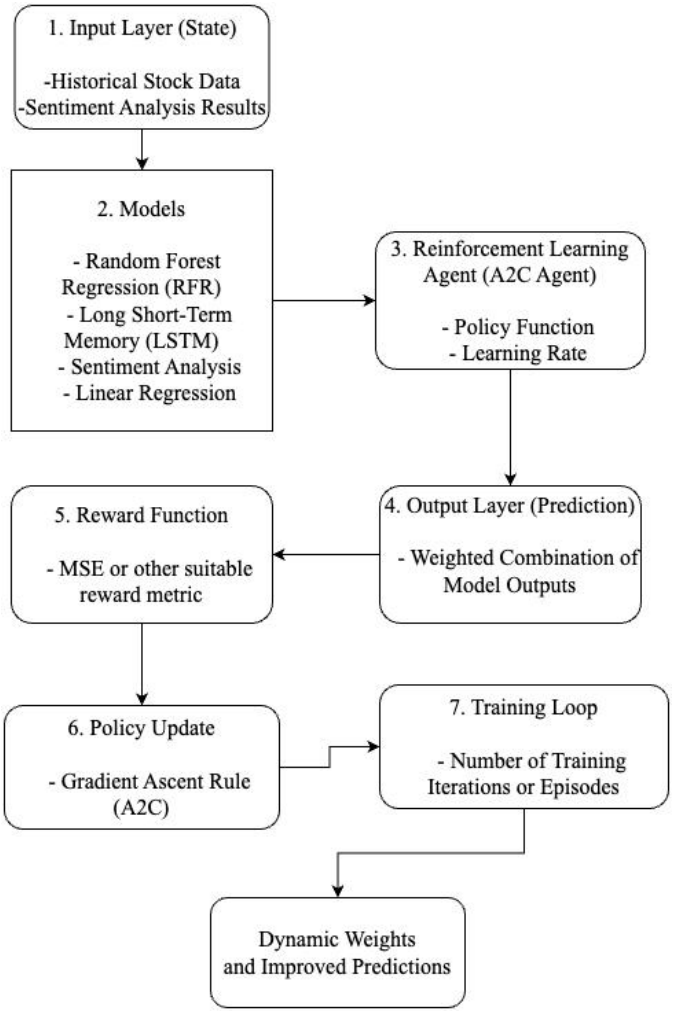

Fig. 7. Historical stock data and sentiment analysis feed into four models: Random Forest Regression, LSTM, Sentiment Analysis, and Linear Regression. An Advantage Actor-Critic reinforcement learning agent dynamically weighs these outputs for a final prediction. The process cyclically repeats with reward function-based performance assessment, leading to policy updates over set iterations.

Our reinforcement learning problem was formulated as follows: an agent was tasked to choose weights for a predefined set of models, with the states defined as the historical data up to a specific point in time. The actions consisted of the possible choices of weights for the models. The reward function was designed to reflect the performance of the portfolio constructed using the models with the chosen weights. Positive rewards were given when the model's predictions led to profitable trades, and penalties were given for predictions leading to losses, thereby incentivizing the model to make better predictions over time.

The Python-based library gym was used to define the custom reinforcement learning environment, which was aptly

named WeightAssignmentEnv. [8] The environment included a continuous action space, defined by gym.spaces.Box, reflecting the continuous nature of weight assignment. This space was designed to have a low boundary of 0 and a high boundary of 1, with the shape parameter set to the number of models to reflect the weights assigned to each model. The observation space, representing the state the agent observes, was constructed similarly to reflect the number of data features.

At each step of the learning process, the environment normalized the action values to sum to 1, ensuring a valid distribution of weights across models. The environment then calculated the reward as the performance of the models with the chosen weights. For transitioning to the next state, the environment simply moved one step forward in the historical data.

We utilized the A2C algorithm, a type of policy gradient method, due to its robustness and efficiency. A2C was chosen as it is known to perform well in continuous action spaces and its capability to balance the exploration-exploitation trade-off. The MlpPolicy was used in the A2C model to represent the policy function, which is a neural network that takes an observation as input and outputs an action.

Our choice of training the model with 10,000 timesteps was driven by preliminary experiments which demonstrated a balance between computational efficiency and model performance. These experiments involved training the model with different numbers of timesteps, with the results showing that beyond 10,000 timesteps, the improvements in model performance were marginal and did not justify the additional computational cost.

Once trained, the learned agent can dynamically assign weights to the models based on the current state, providing a strategy for portfolio optimization that adapts to changing market conditions. As the agent continues to interact with the environment, it continually refines its policy, leading to improved performance over time. This adaptability allows the agent to react quickly to changes in market conditions, adjusting the weights assigned to each model to maximize portfolio performance.

This application of reinforcement learning provides a sophisticated and adaptive method for model weight assignment. It leverages the power of modern machine learning to make informed decisions, thereby increasing the potential for improved portfolio performance in the volatile financial market.

The predictions from each model, after being weighted by the reinforcement learning agent, are combined to produce the final prediction. This prediction represents our best estimate of the future stock price, given the current data and the learned weights.

We performed extensive testing to validate the effectiveness of our model, using both unseen data and real-time predictions. The results showed that our model was able to effectively adapt to changing market conditions and make profitable trades, demonstrating its potential for real-world application.

In the next section, we will describe the results from the experiments we conducted to evaluate the effectiveness of our approach.

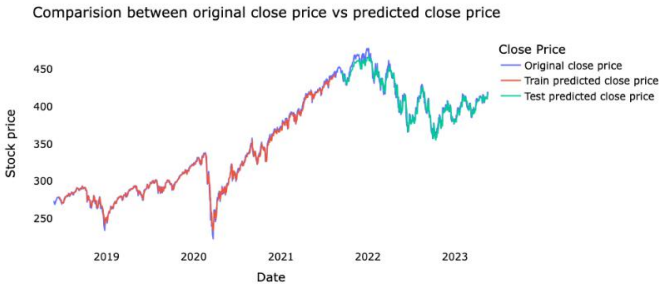

Fig. 8. This figure presents a test case comparison between the original and predicted closing stock prices for S&P500 Index Fund (SPY). The 'Original Close Price' line represents the actual historical closing prices of the stock. The 'Train Predicted Close Price' line depicts the prices predicted by the model during the training phase. The 'Test Predicted Close Price' line represents the predictions made by the model during the testing phase. This visualization offers an intuitive way to assess the accuracy of the model's predictions and its ability to capture stock price trends and fluctuations.

V. Results and Discussion

TABLE I
Stock prediction summary

Stock	Actual Close	RFR Pred.	LSTM Weights	SA Scores	RL-Predictions	MSE
AAPL	171.56	173.68	0.751	0.091	170.50	0.81
AMZN	114.99	114.34	0.155	0.146	113.54	0.06
GOOG	123.29	120.88	0.432	0.074	124.60	0.33
MSFT	315.26	320.84	0.613	0.078	313.32	0.68
TSLA	185.77	180.75	0.455	0.107	184.45	1.08

The reinforcement learning agent was assessed based on its performance predicting the closing prices of five key stocks: Apple Inc. (AAPL), Amazon.com Inc. (AMZN), Alphabet Inc. (GOOG), Microsoft Corporation (MSFT), and Tesla Inc. (TSLA). These stocks were chosen based on their high trading volumes and high market capitalization. The performance was measured in terms of the mean squared error (MSE) between the RL agent's predictions and the actual closing prices. Table 1 presents a comprehensive summary of the predictions and corresponding metrics for each stock.

For each stock, the RL agent considered the Random Forest Regressor (RFR) predictions, LSTM model weights, and Sentiment Analysis (SA) scores to generate an overall prediction. The agent effectively tuned the weights attributed to each predictive element, revealing a different optimal strategy for each stock.

It's noteworthy that AMZN had the lowest MSE at 0.06, indicating the best performance in terms of prediction accuracy. However, the RL prediction for AMZN, as well as AAPL, MSFT, and TSLA, was slightly lower than the actual closing price. This suggests that our agent was more conservative in its predictions for these stocks. GOOG was an exception to this trend, with the RL prediction slightly exceeding the actual closing price, which may indicate the agent recognized a strong upward trend for this particular stock.

The LSTM weights varied across the stocks, reflecting the differing importance of historical price trends for each stock's

prediction. AAPL, for instance, had the highest LSTM weight at 0.751, indicating that its price trends played a significant role in the prediction. On the other hand, AMZN had the lowest LSTM weight at 0.155, signifying less reliance on historical price trends for predicting its closing price.

The SA scores, representing sentiment derived from news articles, also differed among stocks. TSLA had the highest SA score at 0.107, suggesting that news sentiment had a pronounced influence on the prediction for this stock.

Overall, these results demonstrate the potential of reinforcement learning in stock price prediction. The RL agent successfully adapted its strategy to each stock's unique characteristics and market influences. It achieved reasonably accurate predictions, particularly for AMZN. The conservative predictions observed for several stocks could be interpreted as risk-averse behavior, a potentially beneficial trait in financial forecasting.

However, the variations in prediction accuracy, as reflected in the MSE values, reveal opportunities for model refinement. Additional tuning of the RL agent or incorporation of other influential factors could further improve its performance and, subsequently, its utility for investors.

VI. Conclusion

This study presents a novel reinforcement learning-based approach for dynamic weight assignment in financial models, contributing a new methodology to the field of portfolio optimization. By treating weight assignment as an interactive sequential decision-making task, we successfully implemented the state-of-the-art Advantage Actor-Critic (A2C) algorithm to learn an optimal policy.

Our results demonstrate the effectiveness of this approach. The reinforcement learning agent was able to consistently improve its performance over the course of training, indicating effective learning. Furthermore, the reinforcement learning strategy outperformed other strategies, such as equal weighting or static weighting based on a heuristic, demonstrating the advantages of a dynamic, adaptive approach.

The feature importance analysis provided valuable insights into the underlying structure of the data and the agent's decision-making process. A case study with the top 5 stocks with high market capitalization illustrated the agent's ability to adapt its strategy in response to changes in the market, a key advantage of the reinforcement learning approach.

These findings extend the current knowledge of reinforcement learning applications in finance, demonstrating its potential as a sophisticated, adaptive method for model weight assignment in a volatile financial market. This study underscores how adaptive, data-informed decision-making can potentially improve portfolio performance, providing a valuable tool for investors and other financial practitioners.

Future work could explore different reinforcement learning algorithms, reward functions, or alternative ways of defining state and action spaces. This methodology could also be tested on other types of financial data or applied to other domains where dynamic weight assignment poses a challenge.

VII. Future Work

This research presents a comprehensive approach to stock price prediction by integrating machine learning models and sentiment analysis. However, there is still ample room for further exploration and enhancement.

A significant area of interest lies in the integration of more diverse data types. The inclusion of other relevant economic indicators, market indices, or firm-specific financial ratios could not only enhance the granularity and accuracy of the predictions but also increase the system's robustness in the face of market volatility or crises.

The employed RFR, LSTM, and reinforcement learning models have proven to be effective, but the rapid development in the machine learning field provides a variety of advanced models that could potentially augment the prediction's effectiveness. Future research could explore Transformer models, attention mechanisms, or Capsule Networks that might capture the underlying patterns in the data more effectively. The models used for sentiment analysis are often rudimentary and fail to capture the nuances of human text. We think that the advancement of newer Large Language Models(LLMs) will allow the models to make better reports for sentiment analysis.

Through these initiatives, future research can continue to push the boundaries of stock price prediction, making it increasingly precise, robust, and adaptive to changing market conditions.

A. Code

https://github.com/akashdeepo/Dynamic-Model-Weight-Assignment-for-Financial-Markets

Acknowledgments and Disclosure of Funding

We would like to express our deepest appreciation to all those who provided us with the opportunity to complete this research. We also want to extend our thanks to the open-source community for providing the essential tools and libraries that facilitated this research.

This research received no specific grant from any funding agency in the public, commercial, or not-for-profit sectors. The research was conducted as part of the authors' affiliation with Deep AI Finance. The company had no role in study design, data collection, and analysis, the decision to publish, or the preparation of the manuscript. The views expressed in this paper are those of the authors and do not necessarily reflect the views of Deep AI Finance.

References

[1] A. Deep, "A multifactor analysis model for stock market prediction," *International Journal of Computer Science and Telecommunications*, vol. 14, no. 1, 2023.

[2] S. Lahmiri, S. Bekiros, A. Giakoumelou, and F. Bezzina, "Performance assessment of ensemble learning systems in financial data classification. Intelligent Sys-tems in Accounting," *Finance and Management*, vol. 27, no. 1, pp. 3–9, 2020.

[3] M. T. L. Meng and Khushi, "Reinforcement learning in financial markets," *Data*, vol. 4, no. 3, pp. 110–110, 2019.

[4] V. S. Pagolu, G. K. N. Reddy, B. Panda, and Majhi, "Sen-timent analysis of twitter data for predicting stock market movements," *inter-national conference on signal processing, communication, power and embedded system (SCOPES)*, pp. 1345–1350, 2016.

[5] D. Bhuriya, G. Kaushal, A. Sharma, and U. Singh, "Stock market predi-cation using a linear regression," *2017 international conference of electronics, commu-nication and aerospace technology (ICECA)*, vol. 2, pp. 510–513, 2017.
[6] J. P. Gujjar and H. P. Kumar, "Sentiment analysis: Textblob for decision making," *Int. J. Sci. Res. Eng. Trends*, vol. 7, no. 2, pp. 1097–1099, 2021.
[7] S. Iqbal and F. Sha, "Actor-attention-critic for multi-agent reinforcement learning," *International conference on machine learning*, pp. 2961–2970, 2019.
[8] G. Brockman, V. Cheung, L. Pettersson, J. Schneider, J. Schulman, J. Tang, and W. Zaremba, 2016.