Implementation of RNNs using Long-Short term memory Network Model and Decision Tree Regressors for Stock Prediction

Sanjana Gombi
SXG210274
The University of Texas at Dallas

Namratha K Reddy
NXK220058
The University of Texas at Dallas

***Abstract*—In this study, we propose a hybrid approach combining Long Short-Term Memory (LSTM) and Decision Tree Regression for stock prediction. The LSTM model captures sequential dependencies in stock time series data, while Decision Tree Regression refines predictions with interpretable rules.The attempt is to combine the advantages of neural networks and classification-based decision tress. We present a comprehensive evaluation using various stock components, such as prices, volumes, and technical indicators. Our approach provides valuable insights for investors and analysts, aiding in making informed decisions in the complex and unpredictable stock market. The study employs rigorous formatting, style, and styling techniques to ensure clarity and readability.**

I. Introduction

Long Short-Term Memory (LSTM)'s unique design allows it to collect and recall long-term relationships in time-series data such as stock prices. It avoids the vanishing gradient problem of typical RNNs, allowing it to learn patterns over long periods of time. To handle sequential data, LSTMs employ memory cells, gates, and input/output links. LSTMs can discern complicated trends and patterns by studying previous stock prices and pertinent data, allowing them to make relatively accurate forecasts. However, stock markets are impacted by a variety of variables, and their inherent volatility makes any prediction model difficult to use. As a result, while LSTM may be a useful tool for stock prediction, it is critical to combine it with other approaches and exercise caution when making financial choices entirely based on its results. The Decision Tree Regressor is a strong machine-learning method that is used to predict stock prices. It operates by recursively segmenting historical stock data based on characteristics such as prior stock prices, trade volumes, and other pertinent indications. These breaks form a tree-like structure, allowing the model to generate predictions based on past trends in the data. Because the system can capture complicated non-linear interactions, it is useful for stock prediction jobs. Investors may acquire insights into anticipated future stock values by employing Decision Tree Regressor, allowing them to make more educated trading decisions.

II. Motivation

A. Big Data and Stock Market Predictions

In the realm of finance, stock trading stands as a pivotal activity, and seasoned professionals have honed a diverse array of analysis methods, including fundamental analysis, technical analysis, and quantitative analysis. These methods draw upon various sources, encompassing news and price data, all with the goal of accurately predicting future stock prices. Such informed predictions enable traders to make well-informed decisions in their trading endeavors.

Recent years have witnessed a significant rise in the application of machine learning techniques across various industries, and the financial sector is no exception. Many traders have recognized the potential of leveraging machine learning in their stock trading strategies, and some have achieved notable success.

Stock trading on the market is a major investment pursuit. Traditionally, investors have relied on various stock analysis methods to forecast the direction of stock price movements. Accurate modeling and prediction of equity prices, based on current financial information and news, hold immense value for investors seeking insights into the performance of their prospective investments. In this pursuit, financial analysts employ diverse analysis techniques, incorporating current and historical financial data along with other company-specific information, such as balance sheets and various ratios that depict a company's health. Such technical analysis serves as a basis for assessing and forecasting a company's future stock prices, with particular relevance to value investing.

In the modern landscape, the advent of big data has significantly impacted financial services, transforming organizational processes and profitability, among other aspects. For financial analysts and investors, big data presents a promising avenue for enhancing their services and investment strategies. The vast information garnered from stock message boards has become a valuable asset for approximately 71 percent of organizations, alongside the use of historical time series data to derive accurate predictions of the stock market. The application of big data technologies has yielded novel insights for financial organizations and investors alike, creating tangible value in the financial market. Consequently, this project delves into the

development of models utilizing predictive analytics on big data within the financial market, with the aim of achieving more accurate predictions.

III. DATA PRE-PROCESSING

In stock prediction, data preparation is essential for LSTM and Decision Tree Regression models to successfully assess time-series data and closing rates. Handling missing values, standardizing data, and constructing sequential input-output pairs are all part of this procedure. Long-term dependencies are well captured by LSTM, but non-linear connections are well handled by Decision Tree Regression. We want to improve the accuracy and resilience of our stock prediction models by employing these methodologies, which will provide significant insights for investors and analysts in the volatile financial market.

A. Data Source

We make use of Yahoo!, Y! Finance, and Yahoo! finance registered trademarks of Yahoo, Inc. which is an open-source tool that uses Yahoo's publicly available APIs, and is intended for research and educational purposes. yfinance offers a threaded and Pythonic way to download market data from Yahoo! finance.

B. Preprocessing for LSTM Implementation

The data we received from finance.yahoo.com have the daily stock price values as high value, low value, open value, close value, volume of the stock and the adjusted close value. For the program, we just need the closing value and the date for the closing value. The data received had date as index for this data. So, the first step was to create an extra column which can store the index which is the date column. Now the data was ready to be used in a PySpark Dataframe format.

C. Preprocessing for Decision Tree Implementation

- Data Preparation: The code downloads historical stock price data for the "QCOM" stock from July 16, 2018, to July 16, 2023, using the Yahoo Finance API. The data is stored in the variable "data."
- Feature Engineering: If necessary, provide extra features such as price discrepancies between open and close, high and low prices, or other technical indicators such as Moving Averages. All the one-hot encoded categorical features are assembled into a single vector column named "features" using VectorAssembler. This step is necessary as many machine learning algorithms in PySpark expect the input features to be in vector format.
- Label Generation: Create the target variable by comparing today's closing price to tomorrow's closing price. If the next day's price is higher, classify it as 'yes', otherwise 'no'.
- Pipelining: A pipeline is constructed with multiple stages that include all the data preprocessing steps. The pipeline is then fitted to the "qcomData" DataFrame, executing all the defined preprocessing transformations on the data. The resulting pipeline model can be used later for making predictions or training machine learning models on the preprocessed data.
- Data Splitting: To evaluate the model's performance, divide the dataset into training and testing sets.
- Building a Decision Tree: Use a Decision Tree Regressor method to recursively partition the data into nodes depending on the specified characteristics in order to minimize the mean squared error.
- Prediction: Make predictions using the trained Decision Tree on the test dataset.
- Evaluation: Use suitable regression metrics to assess the model's performance, such as Root Mean Squared Error (RMSE) or Mean Absolute Error (MAE).
- Optimization: To improve performance, fine-tune hyperparameters such as tree depth and minimum samples per leaf.

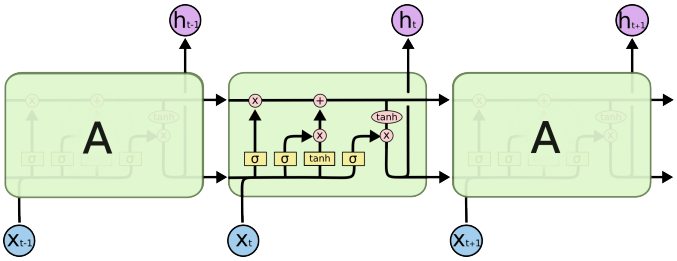

Fig. 1. LSTM structure

IV. MACHINE LEARNING MODELS

A. Working of LSTMs

Human thinking exhibits a sense of continuity and persistence, building upon prior knowledge as we process information. When reading this essay, we comprehend each word based on our understanding of the preceding words, avoiding the need to start from scratch with every momentary thought. This persistence characterizes our cognitive process.

However, traditional neural networks lack this capability, which can be perceived as a significant limitation. For instance, consider the task of classifying events in a movie at different time points. A conventional neural network struggles to utilize its understanding of previous events to influence its interpretation of subsequent events in the movie.

RNNs incorporate loops within their architecture, enabling the preservation and propagation of information across time steps. This loop structure grants RNNs the ability to process sequential data with a sense of temporal context and persistence, allowing them to consider past information while making predictions about subsequent data points.

B. LSTM structure and components

Long Short Term Memory networks, commonly known as "LSTMs," represent a specialized type of Recurrent Neural Network (RNN) capable of learning and handling long-term dependencies. Originally introduced by Hochreiter & Schmidhuber in 1997, LSTMs have undergone refinement and gained

popularity through subsequent research by various experts. Remarkably effective across a wide range of problem domains, LSTMs have become widely adopted in the field of machine learning.

LSTMs are explicitly engineered to address the challenge of preserving long-term information dependencies, making them inherently adept at retaining information over extended periods rather than struggling to learn such dependencies.

In general, all recurrent neural networks follow a sequential arrangement of repeating neural network modules. For traditional RNNs, these repeating modules are relatively simple in structure, often comprising only a single tanh layer.

LSTM (Long Short-Term Memory) networks contain 3 crucial components: the input gate, the forget gate, and the output gate. These components work together to enable LSTMs to capture and retain long-term dependencies in sequential data.

- Input Gate: The input gate informs what information from the current input and the previous hidden state should be modified and passed along to the next step. It uses a sigmoid activation function to decide which parts of the input and previous hidden state should be "remembered" for future use.
- Forget Gate: The forget gate is responsible for deciding which information from the previous hidden state should be discarded or forgotten. Like the input gate, it employs a sigmoid function to output a forget vector that scales the information from the previous hidden state.
- Output Gate: The output gate determines what information from the current input and the updated hidden state should be exposed as the final output. It uses a sigmoid function to regulate the information that should be exposed, and a tanh function to scale the output values to a range between -1 and 1.

By dynamically controlling the flow of information through these gates, LSTMs can retain relevant information over long sequences, effectively addressing the vanishing or exploding gradient problem that conventional RNNs often encounter. This ability makes LSTMs highly effective in capturing complex patterns and dependencies in sequential data, making them a popular choice for tasks such as natural language processing, speech recognition, and time series forecasting.

C. Decision Tree Regressor

A Decision Tree Regressor is a powerful supervised machine learning algorithm used for solving regression problems. Unlike classification tasks where the output is a discrete class label, regression tasks aim to predict continuous numeric values. Decision Tree Regressor works by recursively partitioning the data into subsets and fitting simple models to each subset, making it a popular and interpretable choice for regression tasks.

Working Principle of Decision Tree Regressor:

- Data Splitting: The first step in building a Decision Tree Regressor involves splitting the dataset into subsets based on the values of the features. The algorithm selects the feature and corresponding threshold that result in the best split, aiming to minimize the variance of the target values within each subset.
- Feature Selection: The decision tree algorithm evaluates each feature in the dataset to determine which one provides the most significant split in terms of reducing the variance of the target variable. It calculates various metrics, such as the mean squared error or the mean absolute error, to assess the quality of the split for each feature.
- Recursive Process: Once the initial split is made, the process repeats recursively for each subset created by the split. The algorithm further evaluates the features in each subset to find the best split again, generating more branches in the tree. This recursive process continues until a stopping criterion is met, such as reaching a maximum depth, having a minimum number of samples in the leaf nodes, or achieving a minimum reduction in variance.
- Tree Pruning: Decision trees can be prone to overfitting, where they capture noise in the data and perform poorly on unseen data. Tree pruning techniques help combat overfitting by removing branches that do not contribute significantly to the overall prediction performance. Pruning involves removing nodes with weak statistical significance and replacing them with simpler models, resulting in a more generalized and robust decision tree.
- Prediction: To make predictions with the Decision Tree Regressor, a new data point traverses the tree's branches based on the feature values. It ends up in a leaf node, and the target value for that leaf node becomes the predicted value for the new data point.

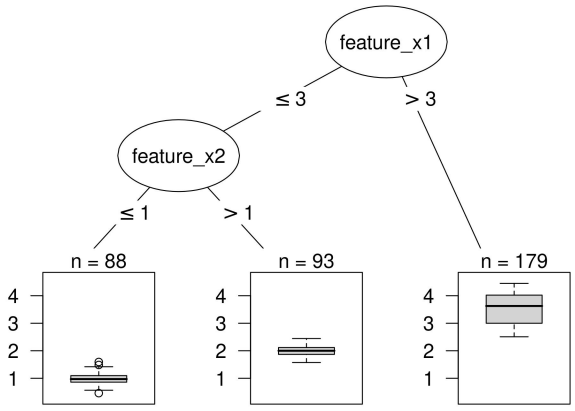

Fig. 2. LSTM structure

We apply machine learning algorithms to the Big Data files to make future predictions using Spark MLlib.

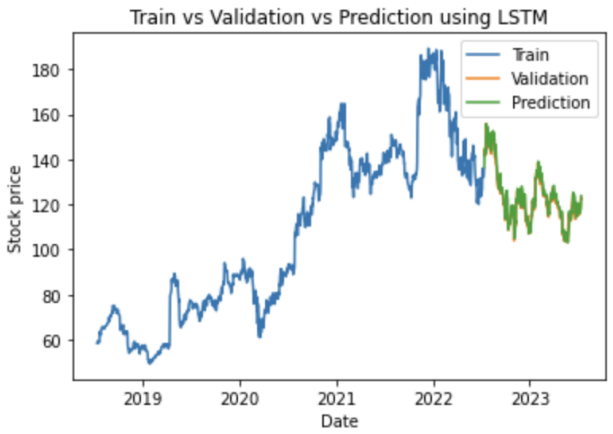

Fig. 3. Train vs Validation vs Prediction using LSTM.

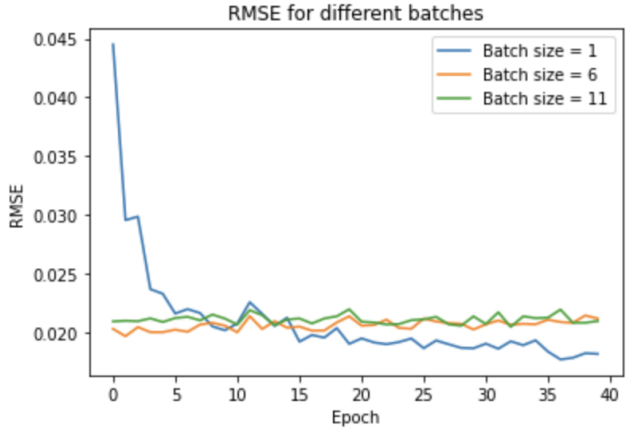

Fig. 4. RMSE for different batches.

V. RESULTS AND ANALYSIS

The prediction of Qualcomm stock values initially employed a batch size of 1 and a single epoch. However, this configuration resulted in suboptimal prediction accuracy for the model. To enhance the precision and accuracy of the predictions, the hyperparameters were fine-tuned to values of 6 and 11 for the batch size and 40 for the number of epochs. After training the model with these tuned hyperparameters, significant improvement in prediction accuracy was observed, enabling more accurate and reliable predictions of the Qualcomm stock values as seen in Fig 3.

The change in Root Mean Square Error (RMSE) with different batch sizes was analyzed for the LSTM model. The evaluation involved three batch sizes: 1, 6, and 11. As depicted in the graph below, it is evident that the model exhibits a higher RMSE when using a batch size of 1. However, as the batch size is increased, the RMSE decreases significantly. The batch size determines the number of samples that are processed together before updating the neural network's weights. With a larger batch size, more samples are utilized, resulting in a more data-rich training process. Consequently, the model learns from a larger pool of information, leading to improved performance. Based on the insights gained from these observations, a batch size of 11 was chosen for the stock market prediction model. This selection aims to strike a balance between computational efficiency and enhanced prediction accuracy, making the model well-suited for its intended application as seen in Fig 4.

We then perform a Decision Tree Classification on our dataset and measure the Root mean square value. Along with this, we try to plot the graph of actual data and the predicted values. Decision Tree Regressors often suffer from overfitting as we can see below in Fig 5. The Root Mean Squared obtained is 0.500053

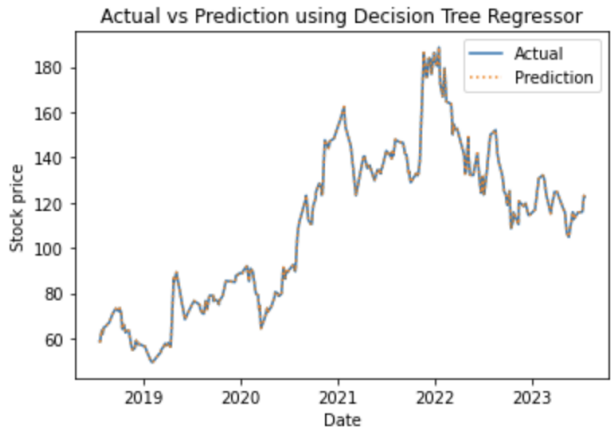

Fig. 5. Actual vs Prediction using Decision Tree Regressor.

CONCLUSION

In this study, we investigated the use of Long Short-Term Memory (LSTM) and Decision Tree Regression (DTR) models for stock price prediction. Both models displayed promising ability in collecting patterns and trends in historical stock data, resulting in significant insights for investors and analysts. As a strong form of Recurrent Neural Networks, the LSTM model excelled at capturing long-term relationships in sequential data. It handled the complicated and volatile nature of stock market time series successfully, leading in accurate stock price predictions. Its capacity to maintain important patterns over time makes it ideal for financial forecasting jobs. The Decision Tree Regression model, on the other hand, demonstrated its ability to handle non-linear correlations and feature significance. It provided interpretable findings, allowing us to comprehend the major elements driving stock values. Its ease of use and efficiency made it an appealing alternative for stock prediction assignments.

FUTURE WORK

Further research into LSTM topologies, hyperparameters, and training methodologies may be done to increase prediction accuracy. Using strategies like as attention mechanisms or

hybrid models that combine LSTM with different architectures might improve the model's performance. In the case of Decision Tree Regression, ensemble approaches such as Random Forest or Gradient Boosting Trees might be investigated to improve prediction accuracy and decrease overfitting. Experimenting with feature engineering and adding domain-specific information may further improve the model's efficacy. In addition, combining sentiment analysis from news and social media data to capture the influence of public mood on stock prices might provide useful insights. Investigating additional external elements such as economic data and market indices may further improve the models' robustness. Finally, rigorous backtesting and validation against real-world data may be carried out to evaluate the models' generalization capabilities and real-world applicability, assisting investors in making educated decisions in the volatile and unpredictable realm of financial markets.

REFERENCES

[1] Understanding LSTM Networks - https://colah.github.io/posts/2015-08-Understanding-LSTMs/

[2] Decision Trees - https://christophm.github.io/interpretable-ml-book/tree.html

[3] Machine Learning to Predict Stock Prices - https://towardsdatascience.com/predicting-stock-prices-using-a-keras-lstm-model-4225457f0233

[4] Evaluation Metrics - https://www.analyticsvidhya.com/blog/2021/05/know-the-best-evaluation-metrics-for-your-regression-model/

[5] Accuracy Class and Metrics -https://keras.io/api/metrics/accuracy_metrics/accuracy-class

[6] Stock Price Prediction and Forecasting using Stacked LSTM - https://www.analyticsvidhya.com/blog/2021/05/stock-price-prediction-and-forecasting-using-stacked-lstm/