Stock Prediction & Visualizer

A project report submitted in partial fulfillment of the
requirements for the award of the degree of

Master of Computer Applications

In

Computer Applications

By

Yashraj Jain (205121107)

DEPARTMENT OF COMPUTER APPLICATIONS

NATIONAL INSTITUTE OF TECHNOLOGY, TIRUCHIRAPPALLI 620015

JUNE 2024

<u>BONAFIDE CERTIFICATE</u>

This is to certify that the project **“Stock prediction & Visualizer”** is a project work successfully done by

Yashraj Jain (205121107)

in partial fulfillment of the requirements for the award of the degree of Master of Computer Applications from the National Institute of Technology, Tiruchirappalli, during the academic year 2023-2024 (6th Semester – CA750 Major Project Work).

Dr. U. Srinivasulu Reddy — **Internal Guide**

Dr. Michael Arock — **Head of the Department**

Project viva-voce held on …………………………….

Internal Examiner

External Examiner

Table of Contents

1. Introduction

1.1 Definition and History:

Stock Prediction: Stock prediction involves forecasting future stock prices based on historical data and various analytical methods. It utilizes statistical models and machine learning techniques, such as Long Short-Term Memory (LSTM) networks, to identify patterns and trends within stock market data. The goal of stock prediction is to provide investors and traders with insights that can aid in making informed decisions about buying or selling stocks.

Visualizer: A visualizer in the context of stock prediction is a tool or system that graphically represents stock data and prediction results. It uses charts and graphs to display historical stock prices alongside predicted prices, allowing users to easily interpret and compare the data. Visualization tools enhance the understanding of model performance and the accuracy of predictions, making the analysis more accessible and actionable.

1.2 Evolution of Stock Prediction Technology:

Below is an overview of the major milestones in the evolution of stock prediction and forecasting:

Early Methods (Pre-1970s):
Fundamental Analysis: Initially, stock prediction relied heavily on fundamental analysis, which involved evaluating a company's financial statements, management, competitive advantages, and market conditions to estimate its intrinsic value.
Technical Analysis: Concurrently, technical analysis emerged, focusing on historical price movements and trading volumes to identify patterns and trends. Analysts used charts and indicators like moving averages and the relative strength index (RSI) to make predictions.

Statistical Methods (1970s-1980s):
Time Series Analysis: The 1970s saw the introduction of time series analysis techniques, such as autoregressive integrated moving average (ARIMA) models. These models used historical data to predict future values based on identified patterns in the time series.

Efficient Market Hypothesis (EMH): During this period, the Efficient Market Hypothesis, proposed by Eugene Fama, gained prominence. It suggested that stock prices fully reflect all available information, making it impossible to consistently achieve higher returns through prediction.

2. Problem Statement and Working ?

2.1 Problem Statement:

The financial markets are complex and volatile, making accurate stock price predictions difficult. Investors need reliable tools to forecast price movements and visualize trends to make informed decisions. This project aims to develop a tool that uses machine learning algorithms to predict stock prices based on historical data. It will also provide detailed visualizations of market trends, enabling users to understand and act on these predictions effectively. The goal is to enhance prediction accuracy and offer a user-friendly platform for real-time market analysis. Here's a detailed step-by-step explanation of how these algorithms typically work:

2.2 Data collection and Preprocessing:

The first step is gathering relevant data. This includes:

Historical Stock Prices: Daily, weekly, or monthly stock prices (open, high, low, close, and volume).

Market Indicators: Indices like S&P 500, Dow Jones Industrial Average.

Company Financial Data: Earnings reports, balance sheets, etc.

External Factors: News articles, social media sentiment, economic indicators.

Raw data is rarely in a form suitable for direct use in algorithms. Preprocessing steps include:

Normalization: Scaling data to a specific range (e.g., [0, 1]) using techniques like MinMaxScaler.

Feature Engineering: Creating new features or selecting relevant ones to improve model performance.

2.3 Training and Prediction:

Training the chosen model involves several steps:

Data Splitting: Dividing the data into training, validation, and test sets to evaluate model performance.

Hyperparameter Tuning: Adjusting model parameters to improve accuracy and avoid overfitting.

Optimization: Using optimization algorithms like Adam, which adjusts learning rates during training for faster and more efficient convergence.

3. Short Methodologies

3.1 LSTM :

LSTMs Long Short-Term Memory is a type of RNNs Recurrent Neural Network that can detain long-term dependencies in sequential data. LSTMs can process and analyze sequential data, such as time series, text, and speech. They use a memory cell and gates to control the flow of information, allowing them to selectively retain or discard information as needed and thus avoid the vanishing gradient problem that plagues traditional RNNs.

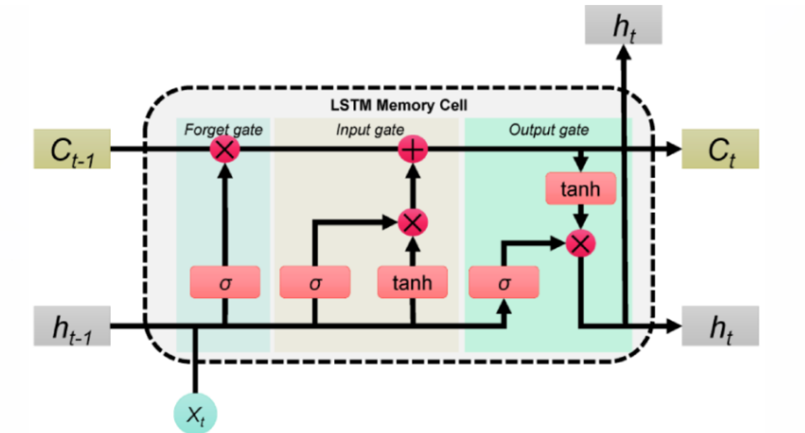

Fig 3.1 (a) LSTM Gates

There are three types of gates in an LSTM: the input gate, the forget gate, and the output gate.

The input gate controls the flow of information into the memory cell. The forget gate controls the flow of information out of the memory cell. The output gate controls the flow of information out of the LSTM and into the output.

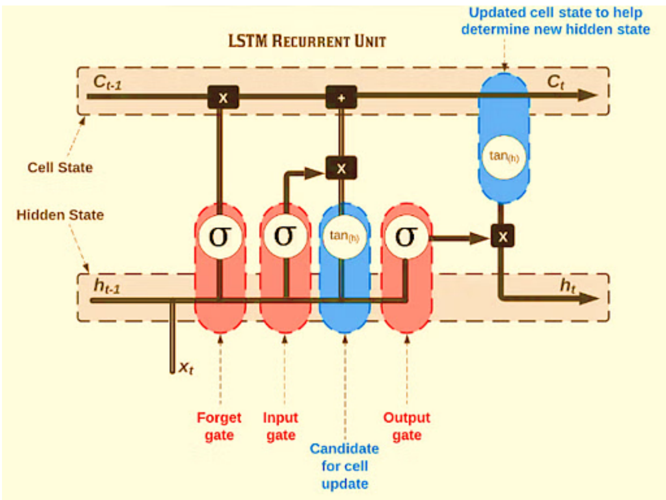

Fig 3.1 (b) Structure Of LSTM

The structure of an LSTM network consists of a series of LSTM cells, each of which has a set of gates (input, output, and forget gates) that control the flow of information into and out of the cell. The gates are used to selectively forget or retain information from the previous time steps, allowing the LSTM to maintain long-term dependencies in the input data.

3.2 Adam Optimizer:

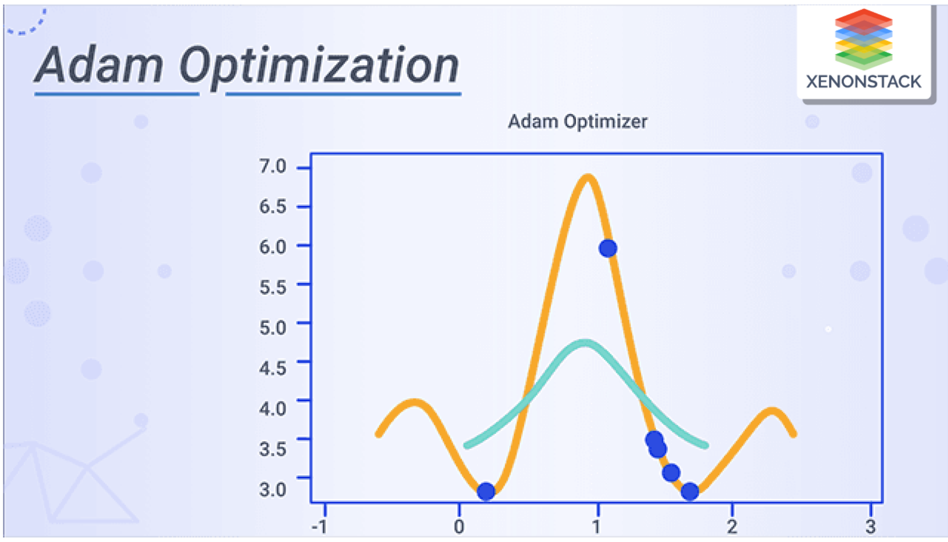

Fig 3.2 (a) Adam Optimizer

Adaptive Moment Estimation is an algorithm for optimization technique for gradient descent. The method is efficient when working with large problem involving a lot of data or parameters. It requires less memory and is efficient. Intuitively, it is a combination of the 'gradient descent with momentum' algorithm and the 'RMSP' algorithm.

Adam optimizer involves a combination of two gradient descent methodologies:

Momentum:

This algorithm is used to accelerate the gradient descent algorithm by taking into consideration the 'exponentially weighted average' of the gradients. Using averages makes the algorithm converge towards the minima in a faster pace.

Root Mean Square Propagation (RMSP):

Root mean square prop or RMSprop is an adaptive learning algorithm that tries to improve AdaGrad. Instead of taking the cumulative sum of squared gradients like in AdaGrad, it takes the 'exponential moving average'.

4. Project Implementation and Explanation

1. **Data Collection:**

Obtain historical stock price data from various sources like Yahoo Finance, Alpha Vantage, or Quandl. You'll typically need data on open, high, low, close prices (OHLC), and volume. Consider factors like dividends, stock splits, and other corporate actions that may affect the data.

2. **Data Pre-processing:**

Clean the data by handling missing values, removing outliers, and adjusting for stock splits or dividends. You might also need to normalize or scale the data to improve model performance.

3. **Feature Engineering:**

Extract relevant features from the raw data that could potentially influence stock prices. These could include technical indicators (e.g., moving averages, RSI), fundamental data (e.g., earnings per share, P/E ratio), and sentiment analysis from news or social media.

4. **Model Selection & Training:**

Choose an appropriate machine learning or deep learning model for stock prediction. Common choices include linear regression, ARIMA, LSTM, or more advanced algorithms like Gradient Boosting Machines (GBM) or Long Short-Term Memory (LSTM) networks.

Split your data into training and testing sets. Train your model on the training data and tune hyperparameters to optimize performance. Use techniques like cross-validation to prevent overfitting and ensure generalization.

5. **Model Evaluation:**

Evaluate your model's performance using metrics like Mean Absolute Error (MAE), Mean Squared Error (MSE), or accuracy. Compare the predicted values against actual values on the test set to assess how well your model generalizes to unseen data.

6. **Visualization:**

Develop a user-friendly visualization interface to display stock prices, predicted values, and any relevant features or indicators. Tools like Matplotlib, Plotly, or Bokeh can be used to create interactive charts and dashboards.

7. **Monitoring and Maintenance:**

Continuously monitor your model's performance in the real world and update it as needed. Markets are dynamic, so your model may need periodic retraining or adjustment to remain accurate.

5. Results

```
Model: "sequential_3"
_________________________________________________________________
Layer (type)                Output Shape              Param #
=================================================================
lstm_7 (LSTM)               (None, 100, 50)           10400
_________________________________________________________________
lstm_8 (LSTM)               (None, 100, 50)           20200
_________________________________________________________________
lstm_9 (LSTM)               (None, 50)                20200
_________________________________________________________________
dense_3 (Dense)             (None, 1)                 51
=================================================================
Total params: 50,851
Trainable params: 50,851
Non-trainable params: 0
_________________________________________________________________
```

Fig 5.1 (a) Sequential_3 Model for Predicting AAPL

The "sequential_3" model for predicting AAPL stock prices utilizes three Long Short-Term Memory (LSTM) layers to capture temporal dependencies in the data, followed by a dense layer for final prediction. The first two LSTM layers, each with 50 units, return sequences and capture complex patterns over time, while the third LSTM layer distills these patterns into a single output. The dense layer then produces the final stock price prediction. With a total of 50,851 trainable parameters, this architecture is designed to effectively learn from historical data and make accurate predictions, even in volatile market conditions like those observed in 2020.

```
Epoch 1/100
12/12 [==============================] - 6s 487ms/step - loss: 0.0206 - val_loss: 0.0505
Epoch 2/100
12/12 [==============================] - 4s 309ms/step - loss: 0.0035 - val_loss: 0.0046
Epoch 3/100
12/12 [==============================] - 4s 300ms/step - loss: 0.0014 - val_loss: 0.0040
Epoch 4/100
12/12 [==============================] - 3s 287ms/step - loss: 8.1361e-04 - val_loss: 0.0073
Epoch 5/100
12/12 [==============================] - 3s 290ms/step - loss: 6.6860e-04 - val_loss: 0.0062
Epoch 6/100
12/12 [==============================] - 3s 255ms/step - loss: 6.4653e-04 - val_loss: 0.0062
Epoch 7/100
12/12 [==============================] - 3s 291ms/step - loss: 6.6186e-04 - val_loss: 0.0062
Epoch 8/100
12/12 [==============================] - 4s 300ms/step - loss: 6.2498e-04 - val_loss: 0.0049
Epoch 9/100
12/12 [==============================] - 4s 297ms/step - loss: 6.2745e-04 - val_loss: 0.0042
Epoch 10/100
12/12 [==============================] - 4s 303ms/step - loss: 6.0206e-04 - val_loss: 0.0050
Epoch 11/100
12/12 [==============================] - 4s 298ms/step - loss: 5.9884e-04 - val_loss: 0.0061
Epoch 12/100
12/12 [==============================] - 4s 304ms/step - loss: 6.1458e-04 - val_loss: 0.0044
Epoch 13/100
...
Epoch 99/100
12/12 [==============================] - 3s 288ms/step - loss: 1.4087e-04 - val_loss: 9.8092e-04
Epoch 100/100
12/12 [==============================] - 3s 285ms/step - loss: 1.4775e-04 - val_loss: 9.3230e-04
```

Fig 5.1 (b) Training Results and Performance Evaluation of the LSTM Model

The training process involved 100 epochs to predict AAPL stock prices using an LSTM model. Initially, the model's performance was suboptimal, with higher losses on the validation data compared to the training data. However, as training progressed, both training and validation losses steadily decreased, indicating the model's learning and improved ability to generalize. Fluctuations in validation loss during middle epochs suggested occasional difficulties in handling unseen data patterns. Ultimately, the model converged to low training and validation losses, around 0.0001 and 0.0009 respectively, demonstrating its accuracy and robustness in predicting AAPL stock prices.

5.2 Graph:

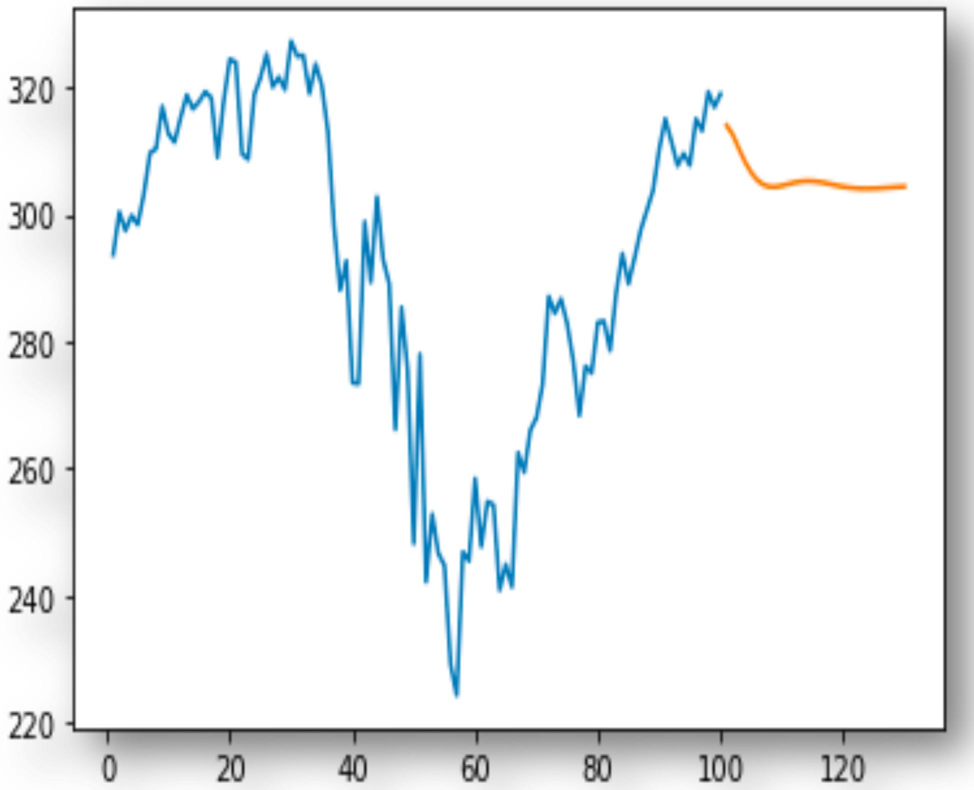

Fig 5.2 (a) Comparison of Actual and Predicted AAPL Stock Prices

Interpretation:

Actual Data (Blue Line): The blue line represents the actual AAPL stock prices from the dataset, starting from index 278 onwards. This portion of the graph shows the historical trends and fluctuations in the AAPL stock prices.

Predicted Data (Orange Line): The orange line represents the predicted AAPL stock prices generated by the LSTM model. It illustrates the model's forecast for future stock prices based on the historical data provided. Comparing the orange line with the blue line allows for an assessment of the model's predictive accuracy. A close alignment between the two lines suggests that the model effectively captures the underlying patterns in the data and makes accurate predictions.