Machine Learning Project Report

Topic: **Time Series Forecasting using LSTM for Stock Price Prediction**

Group Number: 4
Team Members:
Suzen Akhtar - 121CS0136
Khair Alanam - 121CS0137
Ramtul Ali - 121CS0138

Introduction….

Overview of the Problem

Stock price forecasting is a challenging and essential task in the field of financial markets. Predicting future prices can assist investors, financial analysts, and institutions in making informed investment decisions and managing risks. However, accurately predicting stock prices is complex due to the high volatility and noise associated with market data. Traditional time series methods have limited capacity to capture intricate patterns, especially in highly non-linear and dynamic environments like stock markets. In this project, we address this challenge by leveraging Long Short-Term Memory (LSTM) networks, a type of recurrent neural network (RNN) that can model sequential dependencies and handle long-term relationships within the data, making it particularly well-suited for time series forecasting tasks.

Objective of the Project

The primary objective is to utilize the LSTM model to analyze and forecast future stock prices by learning from past data. The steps involve preprocessing the time series data, tuning the model's hyperparameters to optimize performance, and validating model stability to ensure reliable forecasts. Successful implementation of this model can provide valuable insights into stock price movements, demonstrating the potential of deep learning techniques in financial forecasting.

About Dataset

The dataset used in this project is a historical stock price dataset for Microsoft (MSFT), Amazon (AMZN), Apple (AAPL), Netflix (NFLX), Google (GOOGL) and Facebook (META) obtained from Yahoo Finance. This dataset includes daily trading records for all the 6 companies' stock, with columns such as:

Date: The date of each trading day.

Open: The price of the stock at the beginning of the trading day.

High: The highest price reached during the day.

Low: The lowest price reached during the day.

Close: The price of the stock at the end of the trading day, which is typically used as the target variable for forecasting.

Adjusted Close: The closing price adjusted for corporate actions such as dividends and stock splits.

Volume: The total number of shares traded on that day.

These features allow us to analyze the stock's price patterns and trading volume over time. In this project, the Adjusted Close price will serve as the primary feature to forecast future prices. By preprocessing this dataset, including handling missing values, normalizing data, and transforming it into sequences for LSTM input, we aim to train the model on past data to predict the stock's future price, capturing trends and potential shifts in MSFT's market behavior.

Methodology….

Data Collection and Understanding

Collecting Data: Retrieve historical stock price data for Microsoft Corporation (MSFT), AMZN, GOOGL, AAPL, META and NFLX from Yahoo Finance. This dataset includes daily records such as Date, Open, High, Low, Close, Adjusted Close, and Volume.

Data Understanding: Identify key features for time series forecasting, focusing on the Adjusted Close price as the target variable.

Data Exploration

Summary Statistics: Compute descriptive statistics (e.g., mean, median, and standard deviation) to understand the overall distribution of data.

Visualization: Plot line graphs of stock price trends and volume over time to identify patterns, seasonality, and any anomalies in the dataset.

Data Preprocessing

Cleaning: Handle any missing or inconsistent values within the dataset to ensure the quality of inputs.

Feature Engineering: Create time-lagged sequences for the LSTM model input, generating “look-back” windows of historical prices to feed into the model.

Data Scaling: Normalize features using Min-Max scaling to improve model convergence by ensuring consistent value ranges across input data.

Model Development

Model Selection: Choose an LSTM model for its ability to capture sequential dependencies in time series data, ideal for financial forecasting.

Hyperparameter Tuning: Experiment with key parameters like the number of LSTM layers and units, learning rate, batch size, and dropout rate to optimize performance.

Training: Train the LSTM model on the preprocessed data, iterating through epochs while monitoring for overfitting and making adjustments to prevent it.

Model Evaluation

Evaluation Metrics: Assess model performance using metrics like Mean Squared Error (MSE), Root Mean Squared Error (RMSE), and Mean Absolute Error (MAE). These metrics help gauge accuracy and robustness, identifying areas for improvement.

Validation and Testing: Split the data into training and testing sets, and evaluate the model on unseen data to check generalization capability.

Optional GUI Development

User Interface Creation: Build a simple, interactive GUI using Tkinter or Streamlit, where users can select a stock (e.g., MSFT) and view real-time or historical predictions.

Deployment: Deploy the GUI as a web application using Flask or Streamlit, enabling user accessibility and ease of interaction.

Conclusion and Future Work

Model Assessment: Summarize the model's performance and its ability to generalize well on test data.

Improvements: Discuss potential enhancements, such as integrating additional market indicators, incorporating external datasets, or exploring alternative models like GRUs or Transformers for future work.

Model Architecture…..

lstm1 (First LSTM Layer):

- Type: Bidirectional LSTM
- Input: input_size
- Output: hidden_size * 2 for each direction, so hidden_size * 4 in total
- Purpose: Processes the input data, capturing information in both forward and backward directions to increase contextual awareness. The layer has num_layers stacked LSTM layers with dropout applied between them.

lstm2 (Second LSTM Layer):

- Type: Bidirectional LSTM
- Input: hidden_size * 4 (output from lstm1)
- Output: hidden_size for each direction, resulting in hidden_size * 2 in total
- Purpose: Further refines the feature representation, again learning bidirectional dependencies. This layer reduces dimensionality by using fewer hidden units than lstm1.

ln (Layer Normalization):

- Input: Output from lstm2
- Purpose: Normalizes the output along the feature dimension, helping to stabilize training by reducing internal covariate shift, which can speed up convergence and potentially improve generalization.

fc (Fully Connected Layer):

- Input: hidden_size * 2 (the output of layer normalization)
- Output: output_size
- Purpose: Maps the normalized output to the target output size. This layer acts as the final prediction layer for the model.

dropout (Dropout Layer):

- Purpose: Applied between LSTM layers during training to prevent overfitting by randomly setting some neuron activations to zero. The dropout rate is specified by the dropout parameter.

Significance….

1. **Enhanced Investment Decision-Making:** Accurately forecasting stock prices can help investors make informed decisions about buying, holding, or selling stocks, enabling better portfolio management and potentially increasing returns.

2. **Risk Management:** By predicting future price trends, investors can identify potential risks in stock movements, allowing for proactive decision-making to mitigate losses. The LSTM model can also aid financial institutions in risk assessment by providing insights into market trends.

3. **Advancements in Financial Forecasting:** This project contributes to the growing body of research on deep learning applications in financial markets, particularly the use of LSTM networks for time series forecasting. Success in this area demonstrates the value of advanced machine learning models over traditional time series methods, which often struggle with non-linear patterns in stock data.

4. **Scalability for Other Financial Markets**: The methodology and model developed here can be adapted to predict other financial indicators beyond stock prices, such as currency exchange rates, commodity prices, or interest rates, making it widely applicable in various financial forecasting domains.

5. **Educational Value**: For students and researchers, this project provides hands-on experience with complex concepts in time series analysis, LSTM model architecture, data preprocessing, and hyperparameter tuning. It serves as a practical case study in applying machine learning to real-world problems, enhancing skills in both technical and financial analysis domains.

Explanation of the Source Code….

1. **Data Collection:** In the first step we are collecting the dataset to retrieve historical stock price data for MSFT, AMZN, GOOGL, AAPL, META and NFLX from Yahoo Finance. Here the start date specifies the beginning of the date range for the data (2018-01-01) and dynamically uses today's date, ensuring the latest data is downloaded.This dataset includes daily records such as Date, Open, High, Low, Close, Adjusted Close, and Volume.

2. **Data preparation:** After generating raw data, this script likely prepares the datasets by cleaning, formatting, or organizing data.

3. **Data Preprocessing:** For each ticker, the raw CSV data is processed by normalizing or scaling the values, likely using MinMaxScaler from sklearn then the scaled data is saved in data/preprocessed. The scalar objects are saved in data/scalers (e.g., AAPL_scaler.pkl) to ensure consistent scaling for later use in testing or predictions. After preprocessing filler.py file will run which is used for scaling these predictions, integrates them with existing data, and outputs an updated CSV file for each ticker with the missing days filled.

4. **Sequence Generation:** The preprocessed data is transformed into sequences of input and output pairs (X and y) to feed into an LSTM model. This is done to create a time-series sequence of a fixed window (e.g., 60 steps) for the model to predict the next value(s). Each sequence (X and y) is saved in data/training_data as separate .npy files, like AAPL_X_train.npy, AAPL_y_train.npy, AAPL_X_test.npy, and AAPL_y_test.npy, to split the data for training and testing.

5. **Model Definition:** ltms.py defines the YahooLSTM class, a bidirectional LSTM model architecture with two LSTM layers, layer normalization, and a

final fully connected layer for output. This class serves as the core LSTM model used for training and prediction.

6. **Training the Model:** trainer.py handles the training process. Transfers data to the specified device, calculates the loss, backpropagation gradients, and updates model parameters.After each epoch, it evaluates the model on validation data. Then computes the average loss on validation data, which is used to assess model performance without backpropagation. The hyperparameter_tuning method performs grid search over specified hyperparameters (hidden_sizes, num_layers, dropouts) for a given stock ticker. At last, saves the best hyperparameter in the JSON file.

7. **Hyperparameter Tuning :** Defines a list of stock tickers for which the model will be tuned. Sets the number of epochs for training to 50. Define batch size and hyperparameter. For each stock ticker in tickers, it calls trainer.hyper- parameter_tuning with the ticker name and lists of hyper-parameter values (hidden_sizes, num_layers, and dropouts). This will train models for each combination of hyperparameters and select the best model for each ticker based on validation loss. In summary, this code performs hyperparameter tuning for each stock ticker, testing various configurations of the LSTM model to find the optimal set of parameters.

8. **Model Evaluation:** This code evaluates an LSTM model's accuracy on test data for various stock tickers. It loads test data, the trained model, and scaling parameters, then performs predictions. The predictions and actual values are rescaled to the original scale, focusing on 'Close' prices. Root Mean Square Error (RMSE) and Mean Absolute Error (MAE) are computed to quantify prediction accuracy. A plot is generated to visualize actual vs. predicted prices over time. Each ticker's evaluation, including metrics and plots, is printed to assess the model’s performance for each stock ticker on unseen data.

9. **Prediction of stock price:** Loads model parameters and initializes an LSTM model (YahooLSTM) with the optimal settings for the specified ticker. Loads the scaler to revert data to its original scale after prediction. Retrieves the last n_steps of training data to serve as input for generating future predictions. Predicts future values iteratively, using each prediction as the next input, and stores results. Rescales predictions back to their original scale, focusing on 'Close' prices.

Note: The commented-out lines at the bottom show how to predict the next 15 days of closing prices for Amazon (ticker 'AMZN', etc.).This approach is ideal for short-term stock price forecasts based on recent data patterns.

Result….

So here is Mean absolute error and Root Mean Square Error for each of the dataset which we have got after evaluating the models.

Apple:- AAPL - RMSE: 93.47, MAE: 91.68

Amazon:- AMZN - RMSE: 111.19, MAE: 108.56

Meta:- META - RMSE: 171.10, MAE: 155.66

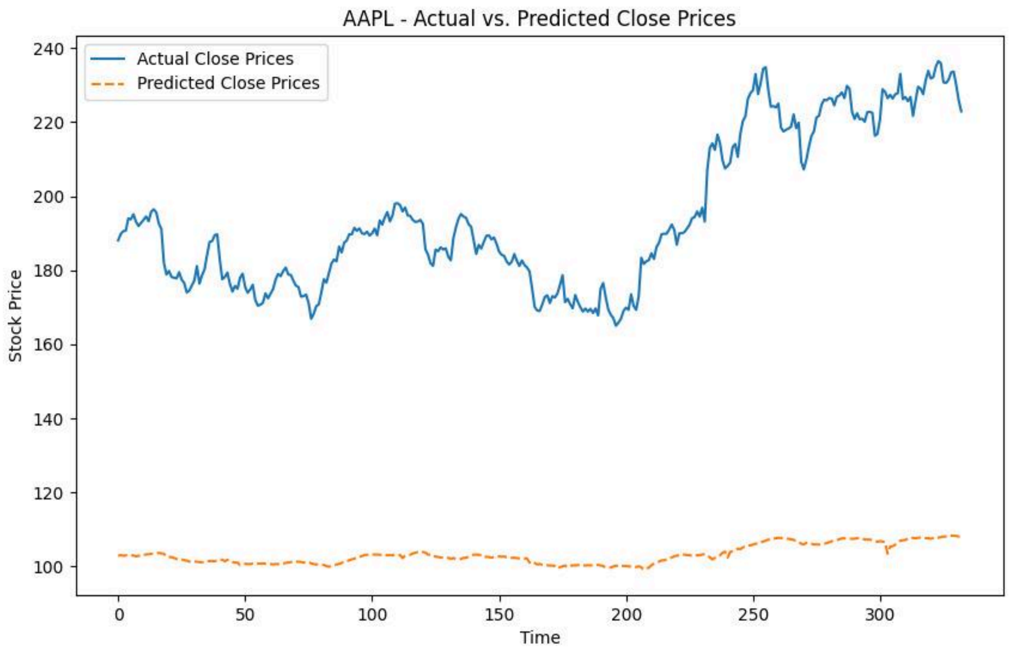
Google:- GOOGL - RMSE: 37.54, MAE: 32.68

Netflix:- NFLX - RMSE: 355.01, MAE: 337.92

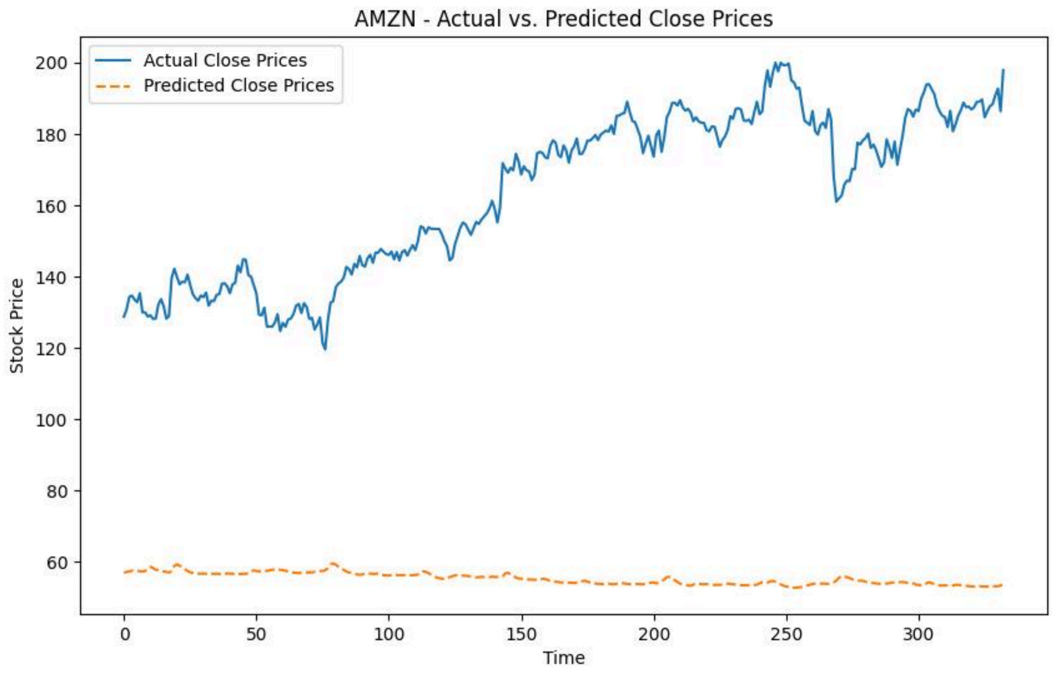
Microsoft:- MSFT - RMSE: 456.04, MAE: 454.16

Here is the time series plot comparing actual and predicted closing prices for each of the datasets.

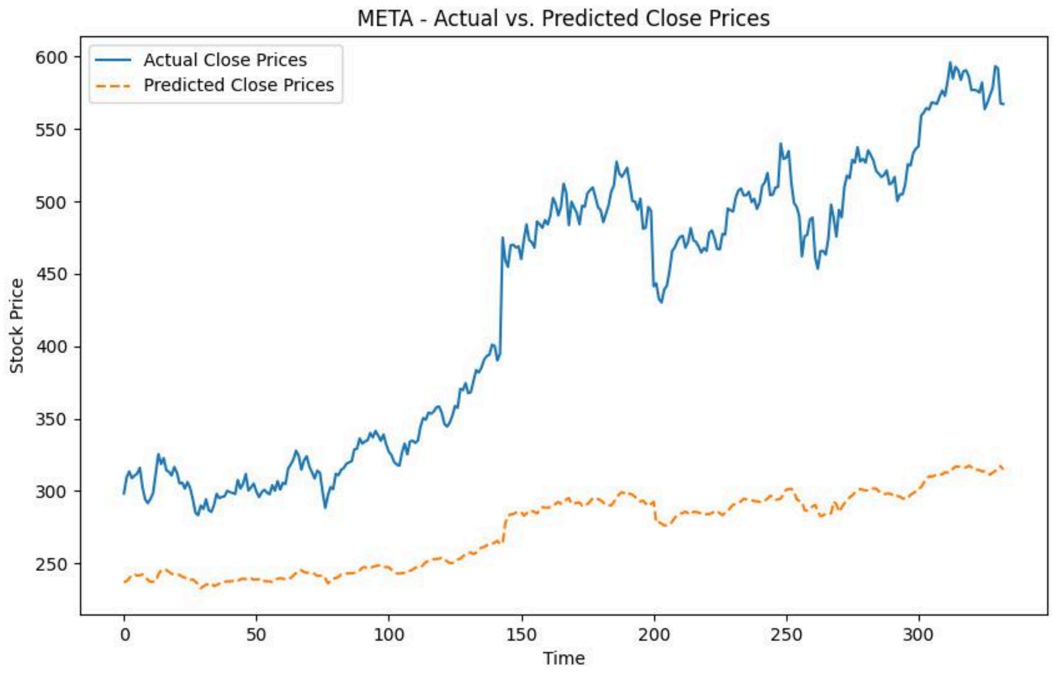
FOR APPLE:-



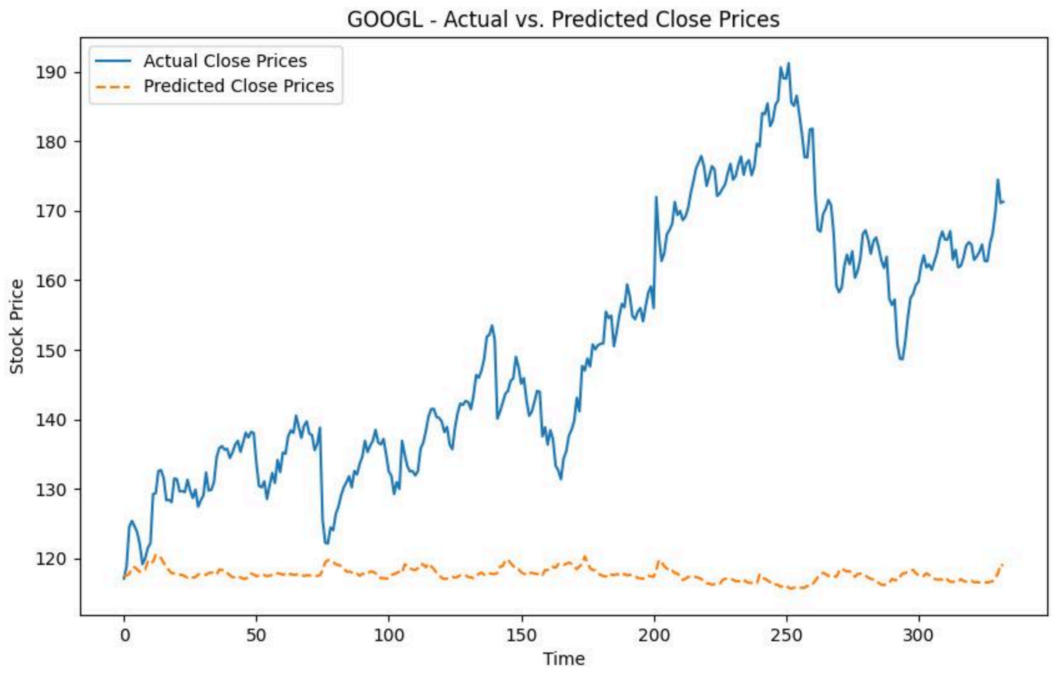
FOR AMAZON:-


AMZN - Actual vs. Predicted Close Prices
Actual Close Prices
Predicted Close Prices
Stock Price
Time
200
180
160
140
120
100
80
60
0
50
100
150
200
250
300

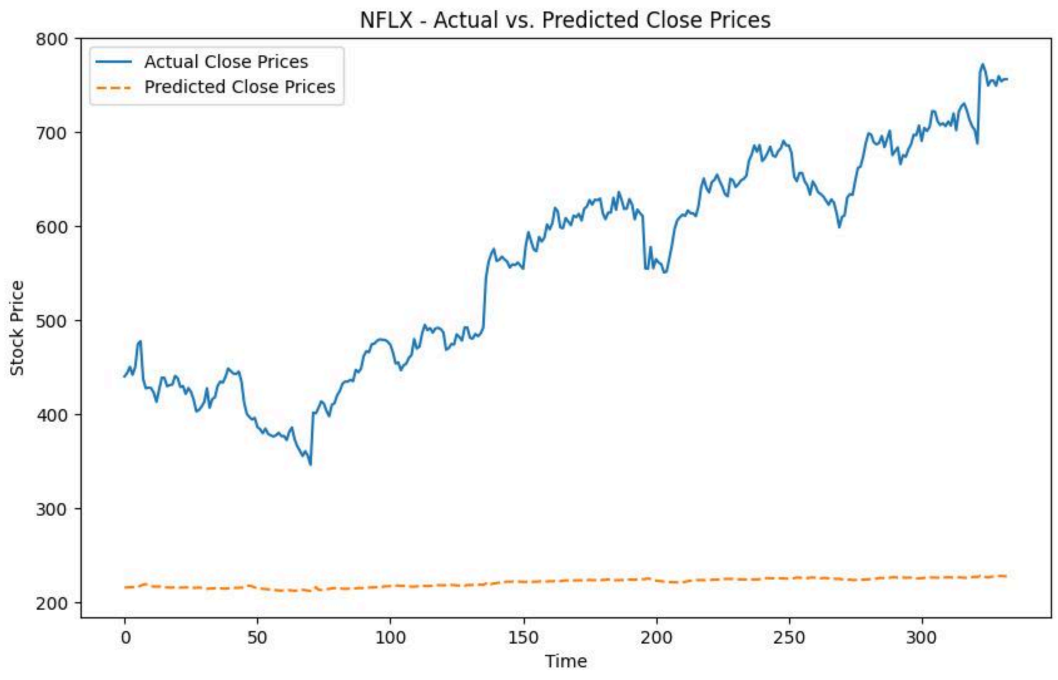
FOR META:-


META - Actual vs. Predicted Close Prices
Actual Close Prices
Predicted Close Prices
Stock Price
Time
600
550
500
450
400
350
300
250
0
50
100
150
200
250
300

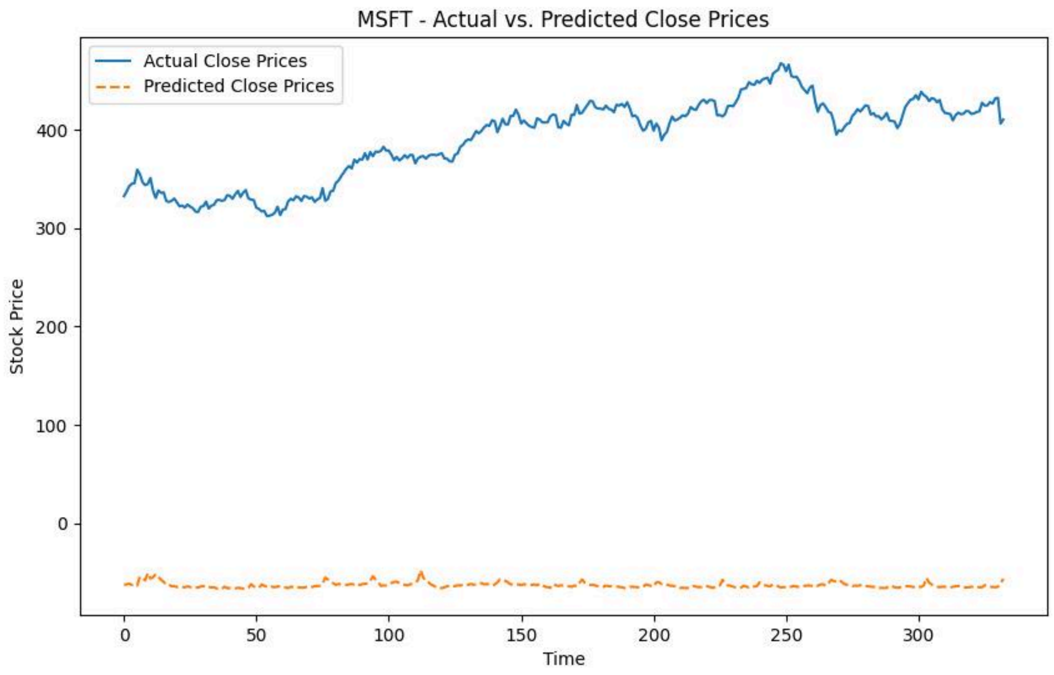
FOR GOOGLE:-



FOR NETFLIX:-

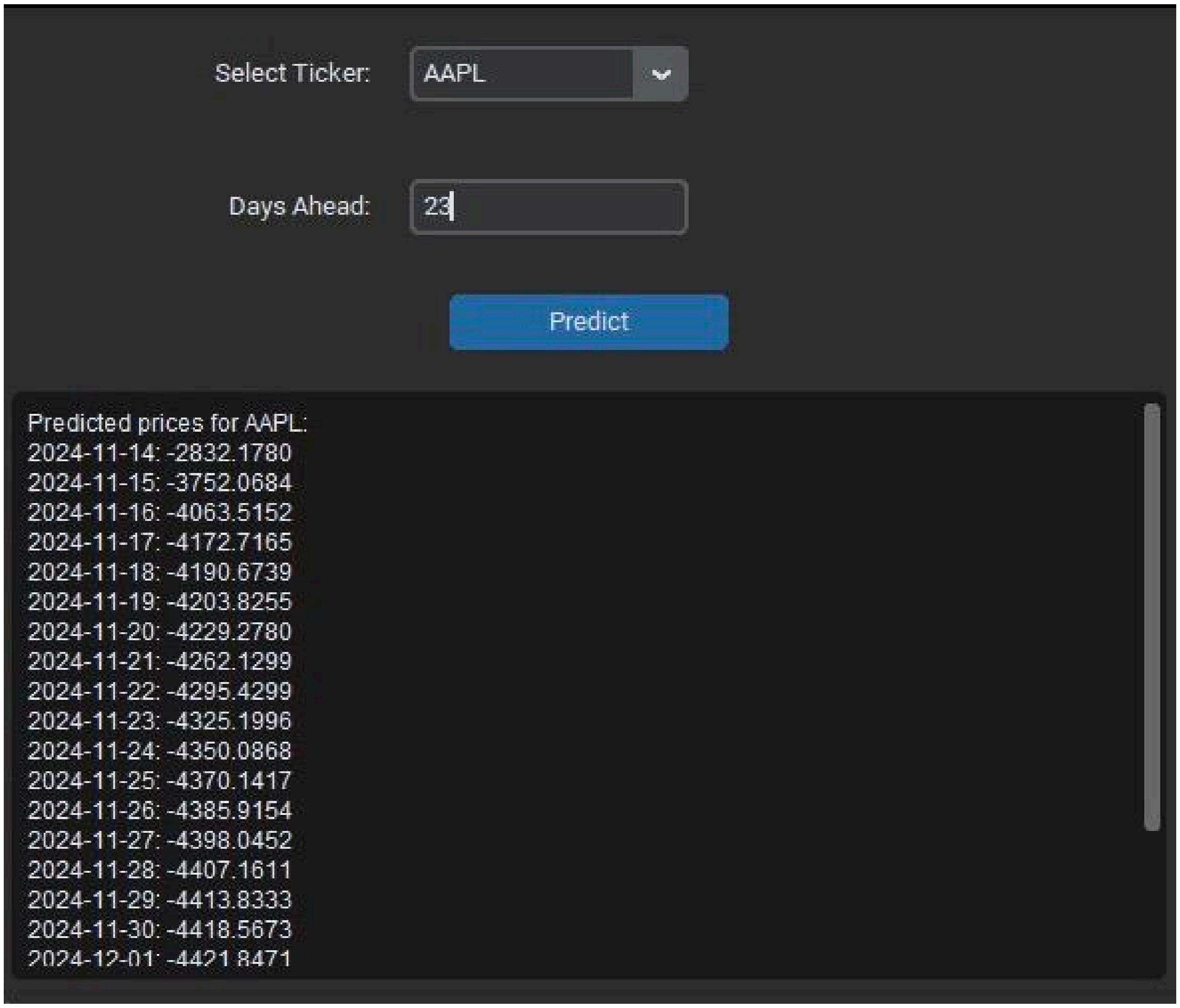


FOR MICROSOFT:-



GUI Interface of the whole implementation

This GUI interface is a stock price prediction tool. It allows the user to select a stock ticker, in this case "AAPL" (Apple Inc.), and input the number of days ahead to predict the price. After entering the days and clicking the "Predict" button, the tool displays a list of predicted prices for each future date. The predictions for Apple's stock price seem to show a consistent downward trend, as the prices are progressively decreasing over time. The interface is simple and user-friendly, with a dropdown for selecting the ticker, an input box for days ahead, and a button to initiate the prediction.



```
Predicted prices for AAPL:
2024-11-14: -2832.1780
2024-11-15: -3752.0684
2024-11-16: -4063.5152
2024-11-17: -4172.7165
2024-11-18: -4190.6739
2024-11-19: -4203.8255
2024-11-20: -4229.2780
2024-11-21: -4262.1299
2024-11-22: -4295.4299
2024-11-23: -4325.1996
2024-11-24: -4350.0868
2024-11-25: -4370.1417
2024-11-26: -4385.9154
2024-11-27: -4398.0452
2024-11-28: -4407.1611
2024-11-29: -4413.8333
2024-11-30: -4418.5673
2024-12-01: -4421.8471
```

Conclusion…..

1. LSTM Model Performance in Stock Price Prediction

The project allows us to evaluate the effectiveness of Long Short-Term Memory (LSTM) networks for time-series forecasting, specifically in the context of stock prices.By training LSTM models on historical data, we can determine how well these models can capture trends and patterns over time.

2. Importance of Data Preprocessing and Scaling

Preprocessing, especially scaling with MinMaxScaler, is crucial when working with neural networks on financial data, as it ensures that all input features are in a comparable range. This step helps avoid issues with vanishing or exploding gradients during training. The normalization process demonstrates the necessity of preparing financial data properly for model training, as unprocessed data can lead to poor model performance.

3. Hyperparameter Tuning's Role in Model Accuracy

The project's exploration of different hyperparameters (e.g., hidden_size, num_layers, dropout) reveals the impact of tuning on model accuracy and generalization. By testing various configurations, we observe how certain hyperparameters affect the LSTM's ability to learn patterns in the data, highlighting the importance of tuning to balance model complexity and performance.

4. Limitations of Stock Price Prediction Using Historical Data

While the LSTM model might achieve good accuracy on test data, predicting stock prices remains inherently challenging due to market volatility and unpredictability.This project likely illustrates that while machine learning models can identify patterns in historical data, they cannot account for external factors (e.g., economic events, news) that significantly impact stock prices.This limitation emphasizes the need for caution when using such models in real-world financial decision-making.

5. Scalability of Time-Series Forecasting Across Multiple Stocks

By creating a pipeline that processes and trains models on multiple tickers (AAPL, AMZN, GOOGL, etc.), the project demonstrates scalability. This approach can be applied to various stocks or other financial instruments, showing that with proper data preparation, the methodology is adaptable to different assets in the financial market.

Reference…..

https://www.simplilearn.com/tutorials/machine-learning-tutorial/stock-price-prediction-using-machine-learning

https://www.geeksforgeeks.org/stock-price-prediction-using-machine-learning-in-python/

For datasets - https://finance.yahoo.com/