|  |  |  |
| --- | --- | --- |
|  | G H Patel College of Engineering and  Technology |  |

**COMPUTER ENGINEERING DEPARTMENT**

**Project Report on**

**STOCK PRICE PREDICTOR**

**Submitted By**

**Name of Student-1: Shaunak Lad**

**Enrolment Number: 12202040501057**

**Name of Student-2: Dwij Prajapati**

**Enrolment Number: 12202040501071**

**Guided By: Dr. Priyang Bhatt**

# A.Y. 2024-25 EVEN TERM

**Objective:**

The primary objective of this project is to build a robust forecasting system that can predict future stock prices using historical data. Utilizing a Long Short-Term Memory (LSTM) neural network and Random Forest Algorithm, we aim to capture temporal patterns in Microsoft’s stock prices and forecast future “Close” values. The system is useful for traders, investors, and data scientists to understand and anticipate market trends.

**Dataset Used:**

The project uses the historical stock prices of **Microsoft Corporation**, including features like:

* Date
* Open
* High
* Low
* Close
* Volume

**Source:**

Public Microsoft Stock Data (CSV format)

**Key Features:**

* **open**: Price at the beginning of the trading day
* **high**: Highest price of the day
* **low**: Lowest price of the day
* **close**: Price at the end of the trading day (target variable)
* **volume**: Number of shares traded

**Model Chosen: LSTM (Long Short-Term Memory Network) & Random Forest Algorithm:**

LSTM networks are an advanced type of RNNs that are effective at learning order dependence in time series data. The LSTM model here takes 60 past days of “close” prices to predict the next day's closing price.

**Why LSTM?**

* Capable of learning long-term dependencies
* Prevents vanishing gradient problem
* Suitable for sequential/temporal data like stock prices

**Model Architecture-**

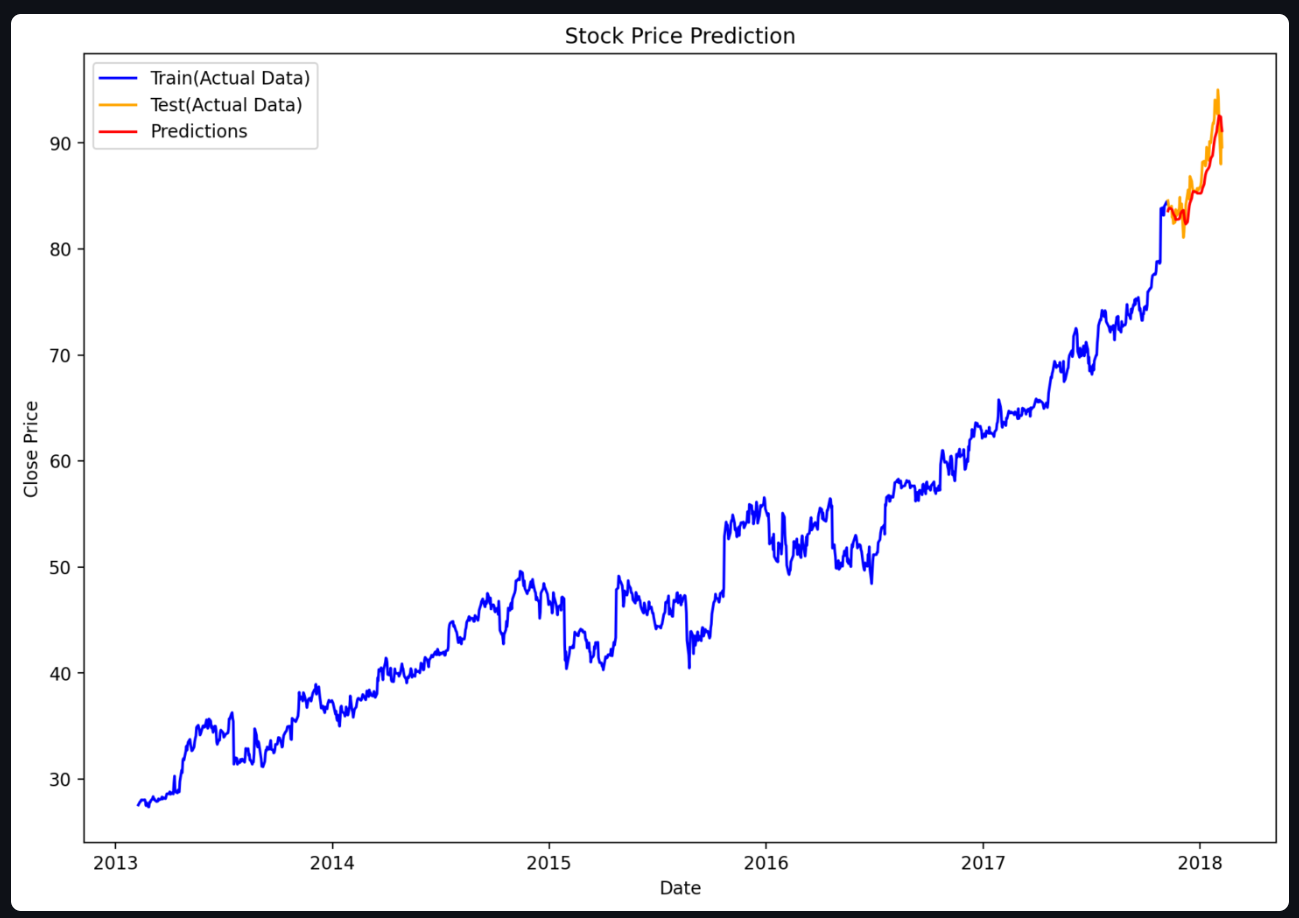
* Input Shape: (60, 1) (past 60 days of closing prices)
* 1st LSTM Layer: 64 units, return\_sequences=True
* 2nd LSTM Layer: 64 units
* Dense Layer: 128 units with ReLU activation
* Dropout Layer: Dropout rate = 0.5
* Output Layer: Dense with 1 unit (forecasted price)

**Performance Metrics-**

* **Loss Function**: Mean Absolute Error (MAE): 1.46%
* **Evaluation Metric**: Root Mean Squared Error (RMSE): 0.78

**Dataset Size-**

* **Total Records**: *1259 rows*
* **Training Data: 95%**
* **Test Data: 5%**



**Random Forest Regressor**

In addition to LSTM, a Random Forest Regressor was implemented to compare traditional machine learning techniques with deep learning for time series forecasting.

Why Random Forest?

* Handles non-linear data patterns well
* Less sensitive to outliers and noise
* Performs efficiently on small to medium-sized datasets
* Offers feature importance insight, which helps in model interpretability

**How Random Forest Works for Regression?**

Random Forest builds an ensemble of decision trees during training and outputs the average of predictions from individual trees.

Steps:

1. Random Sampling: Multiple subsets are randomly drawn from the dataset.
2. Decision Trees: A separate regression tree is trained on each subset.
3. Averaging: Predictions from all trees are averaged to get the final forecast.

**Model Configuration**

from sklearn.ensemble import RandomForestRegressor

model = RandomForestRegressor(n\_estimators=100, random\_state=42)

model.fit(X\_train\_rf, y\_train\_rf)

predictions\_rf = model.predict(X\_test\_rf)

* n\_estimators: 100 trees
* criterion: MSE (Mean Squared Error)
* Feature Used: Past 60 days closing prices

**Performance Metrics**

* MAPE: 3.82%
* R² Score: *-0.74*

These metrics were used to evaluate and compare the forecasting capability of Random Forest against LSTM.

A graph with blue lines

AI-generated content may be incorrect.

**Visualization-**

* Time Series of Open and Close Pices
* Stock Volume Trends
* Correlation Heatmap between features
* Prediction vs Actual Close Price Plot

A screenshot of a heat map

AI-generated content may be incorrect.

A green line graph with numbers

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

**Challenges & Learnings:**

**Challenges Faced**

* High volatility in stock prices
* Hyperparameter tuning of LSTM
* Preventing overfitting during training

**Key Learnings**

* Feature scaling is critical for neural networks
* LSTM architecture tuning improves forecast accuracy
* Visualization helps in model evaluation and debugging

**Conclusion:**

This project successfully demonstrates the potential of deep learning (LSTM) in financial forecasting. It offers a predictive model capable of learning from sequential stock price patterns and generating accurate forecasts. Future work may include the use of external features such as news sentiment or macroeconomic indicators.

**Tools Used-**

* **Languages**: Python
* **Libraries**: TensorFlow, Keras, Pandas, NumPy, Scikit-learn, Matplotlib, Seaborn
* **IDE**: Jupyter Notebook / VS Code

**References-**

1. Hochreiter, S., & Schmidhuber, J. (1997). Long Short-Term Memory. Neural Computation.
2. Brownlee, J. (2017). Deep Learning for Time Series Forecasting.
3. Microsoft Historical Stock Data – Yahoo Finance