Stock Price Prediction using LSTM Neural Networks

Project Submitted By:
Name: Muhammad Bilal Sajid
Roll No.: SU92-BSAIM-F23-018
Section: BSAI 3A

Submitted To:
Instructor Name: Sir Rasikh Ali

Table of Contents

Introduction

Stock price prediction has always been a challenging domain due to its dynamic nature and sensitivity to market forces. This project utilizes the power of Artificial Intelligence, specifically **Recurrent Neural Networks (RNNs)** and **Long Short-Term Memory (LSTM)** networks, to predict stock prices based on historical data. It incorporates technical indicators to enrich the input dataset, providing a robust foundation for accurate predictions.

Objective

The primary goal of this project is to develop a predictive model capable of forecasting future stock prices with high accuracy by leveraging advanced AI techniques. The model will:

- Analyze historical stock data.

- Incorporate technical indicators such as SMA, EMA, and RSI.
- Provide predictions for future stock trends.
- Visualize actual vs. predicted prices to evaluate model performance.

Dataset Description

The project uses historical stock market data fetched from **Yahoo Finance** using the `yfinance` Python library. The dataset contains:

- **Features:** Open, High, Low, Close prices, and Volume.
- **Time Range:** 2010 to 2024.
- **Company:** Tesla Inc. (TSLA).

Key technical indicators calculated from the dataset include:

1. **Simple Moving Average (SMA):** Tracks price trends over 50 days.
2. **Exponential Moving Average (EMA):** A more dynamic average considering recent prices.
3. **Relative Strength Index (RSI):** Indicates overbought or oversold conditions.

Project Workflow

1. **Data Collection:** Fetch historical stock data using `yfinance`.
2. **Data Preprocessing:** Calculate technical indicators, normalize data, and prepare sequences for LSTM input.
3. **Model Building:** Design and train a Bi-Directional LSTM model.
4. **Prediction:** Predict stock prices for the test period.
5. **Evaluation:** Compare predicted prices with actual values using MAPE.
6. **Visualization:** Plot actual vs. predicted prices for better understanding.

Methodology

Technical Indicators

- **SMA:** Calculates the average of closing prices over the last 50 days.
- **EMA:** Applies a weighted average, giving more importance to recent prices.
- **RSI:** Measures the speed and change of price movements over a 14-day window.

Data Preprocessing

- Missing values were removed to ensure clean data.
- Features were scaled using `MinMaxScaler` to ensure uniformity.
- Sequences of 60-day windows were prepared for input into the LSTM model.

Model Architecture

- **Layers:**
  - Input Layer: Accepts 3D sequences of shape `(60, 4)`.
  - Two Bi-Directional LSTM layers (100 and 50 units).
  - Dropout layers (20%) to prevent overfitting.
  - Dense layer for output prediction.
- **Loss Function:** Mean Squared Error (MSE).
- **Optimizer:** Adam optimizer for faster convergence.

Technical Overview

Dependencies

- **Libraries:**
  - `yfinance`: Fetch stock data.
  - `pandas` and `numpy`: Data processing and manipulation.
  - `matplotlib`: Data visualization.
  - `tensorflow`: Building and training the LSTM model.
  - `sklearn`: Feature scaling and evaluation.

Code Workflow

1. **Data Fetching:**

   ```
   python
   Copy code
   stock_data = yf.download('TSLA', start='2010-01-01', end='2024-12-31',
   interval='1d')
   ```

2. **Technical Indicators:** RSI calculation:

   ```
   python
   Copy code
   delta = data.diff()
   gain = (delta.where(delta > 0, 0)).rolling(window=14).mean()
   loss = (-delta.where(delta < 0, 0)).rolling(window=14).mean()
   RS = gain / loss
   RSI = 100 - (100 / (1 + RS))
   ```

3. **Model Training:**

```
python
Copy code
model.fit(X_train, y_train, batch_size=64, epochs=100,
validation_split=0.2, callbacks=[early_stop])
```

4. **Prediction:** Inverse transformation to retrieve actual prices:

```
python
Copy code
predictions = scaler.inverse_transform(predicted_scaled)
```

Results and Evaluation

Performance Metrics

The model's accuracy is calculated using **Mean Absolute Percentage Error (MAPE)**:

```
python
Copy code
mape = mean_absolute_percentage_error(y_true, y_pred)
accuracy = 100 - mape
```

- **Accuracy:** ~85-90% (varies based on test data).

Visualization

The plot below illustrates actual vs. predicted stock prices for 2023-2024:

- **Blue Line:** Actual Prices
- **Red Line:** Predicted Prices

(The graph is generated during runtime using Matplotlib.)

Conclusion

This project successfully implements a Bi-Directional LSTM model for stock price prediction. By leveraging historical data and technical indicators, it demonstrates how AI can predict future trends in financial markets with reasonable accuracy.

Key Takeaways:

- AI models like LSTM are powerful tools for sequential data analysis.

- Incorporating technical indicators enhances prediction performance.
- Proper data preprocessing and scaling are crucial for model success.

Future Work

1. **Feature Enrichment:** Add more technical indicators or external factors like economic indices.
2. **Multi-Stock Prediction:** Expand the model to predict multiple stocks simultaneously.
3. **Real-Time Prediction:** Develop a real-time prediction system using live data streams.
4. **Hyperparameter Tuning:** Use techniques like Grid Search or Bayesian Optimization for improved model performance.