**Stock Price Prediction using LSTM**

**Project Overview**

This project is a stock price prediction tool that leverages historical stock data and an LSTM (Long Short-Term Memory) neural network to forecast future stock prices. It is designed to analyze past trends and predict prices for the next 30 days. The predictions are based on temporal patterns in the historical stock data.

**Features**

* **Fetch Data**: Retrieves historical stock prices using Yahoo Finance's API (yfinance library).
* **Data Preparation**: Scales and sequences the stock prices to prepare the data for time-series analysis.
* **LSTM Model**: Builds and trains an LSTM model to capture temporal dependencies.
* **Evaluation**: Evaluates the model using metrics such as MSE, RMSE, MAE, and R² score.
* **Prediction**: Predicts stock prices for a user-defined number of future days (default: 30 days).
* **Visualization**: Plots actual vs. predicted stock prices for better interpretation.
* **Error Logging**: Logs the process and errors in a file for debugging and monitoring.

**Technologies Used**

* **Programming Language**: Python
* **Libraries**:
  + Data Handling: pandas, numpy
  + Data Fetching: yfinance
  + Machine Learning: tensorflow, scikit-learn
  + Visualization: matplotlib
  + Logging: logging

**How It Works**

1. **Initialization**:
   * Specify the stock symbol (e.g., AAPL for Apple) and the number of past days to use for training (default: 60 days).
2. **Data Fetching**:
   * Fetches historical stock data (e.g., closing prices) from Yahoo Finance.
3. **Data Preparation**:
   * Scales the data using MinMaxScaler.
   * Generates sequences of past stock prices as features and the next day's price as the target.
4. **Model Training**:
   * Builds an LSTM-based neural network and trains it using the prepared data.
5. **Evaluation**:
   * Evaluates the model's performance on a test dataset.
6. **Prediction**:
   * Predicts future stock prices based on the trained model.
7. **Visualization**:
   * Plots the actual vs. predicted prices for better analysis.

**Example Usage**

if \_\_name\_\_ == "\_\_main\_\_":

# Initialize predictor for Apple stock

predictor = StockPredictor('AAPL')

# Fetch historical data

data = predictor.fetch\_data()

# Prepare data

X\_train, X\_test, y\_train, y\_test = predictor.prepare\_data(data)

# Build and train model

predictor.build\_model((predictor.prediction\_days, 1))

history = predictor.train\_model(X\_train, y\_train)

# Evaluate model

metrics = predictor.evaluate\_model(X\_test, y\_test)

print("Model Evaluation Metrics:")

for metric, value in metrics.items():

print(f"{metric.upper()}: {value:.4f}")

# Make future predictions

future\_predictions = predictor.predict\_future(days=30)

print("\nPredicted prices for next 30 days:")

for i, price in enumerate(future\_predictions, 1):

print(f"Day {i}: ${price:.2f}")

**Example Output**

Model Evaluation Metrics:

MSE: 0.0004

RMSE: 0.0200

MAE: 0.0150

R2: 0.9850

Predicted prices for next 30 days:

Day 1: $247.78

Day 2: $246.48

Day 3: $245.46

...

Day 30: $250.63

**Project Files**

* **stock\_predictor.py**: The main Python script containing the code.
* **stock\_predictions.log**: Log file for tracking processes and errors.

**Limitations**

* **Assumptions**: Assumes historical trends are indicative of future movements.
* **External Factors**: Does not account for market news, economic conditions, or company performance.
* **Smooth Predictions**: May not capture high volatility or sudden market changes.

**Future Enhancements**

* Incorporate external data like market news or economic indicators.
* Test and optimize the model for various stocks and timeframes.
* Implement hyperparameter tuning for better accuracy.
* Add a user interface for non-technical users.

**Conclusion**

This project demonstrates the application of LSTM networks for time-series forecasting in the stock market. While functional, it is recommended to use it as part of a broader financial analysis strategy.