README: LAB 3 PART 2

Model 1: Simple Moving Average (SMA):

Input:

Stock Names: The stock ticker symbol needs to be provided as input in the script**.**

- **Example:** 'AAPL' (Apple Inc.)

Execution:

- Run the script directly after setting the stock ticker in the __main__ block.
- The script will automatically download historical stock data, compute indicators, generate buy/sell signals, and predict future stock prices.

Dependencies:

Ensure the following Python libraries are installed:

- **yfinance:** For fetching historical stock data
- **pandas:** For data manipulation
- **numpy:** For numerical computations
- **matplotlib:** For plotting graphs and visualizations
- **scikit-learn:** For linear regression modeling

You can install these dependencies using below command:

- pip install yfinance pandas numpy matplotlib scikit-learn

About:

Data Loading and Preparation

- **Stock Data**: The script fetches historical stock price data for the last 10 years from Yahoo Finance using the yfinance library.
- **Exponential Moving Averages (EMA)**: The script calculates 50-day and 200-day EMAs to detect trends.

Trading Strategy

- The strategy is based on **EMA crossovers**:
- **Buy Signal**: Triggered when the 50-day EMA crosses above the 200-day EMA.
- **Sell Signal**: Triggered when the 50-day EMA crosses below the 200-day EMA.

Mock Trading Simulation

- Buy and sell signals generated by the EMA crossover strategy are used to simulate a basic trading approach.

Portfolio Management

- The code currently does not include portfolio management, but the signals could be used to design one based on historical performance.

Visualization

- The stock price along with the EMAs and buy/sell signals is plotted.
- Historical prices and predicted prices for the next 50 days are visualized separately.

Functions

a. get_stock_data(ticker, period="10y")

Fetches historical stock price data for a given stock ticker from Yahoo Finance.

b. calculate_ema(stock_data, short_window=50, long_window=200)

Calculates the 50-day and 200-day Exponential Moving Averages for the stock data.

c. generate_signals(stock_data)

Generates buy/sell signals based on the crossover of 50-day and 200-day EMAs.

d. plot_signals(stock_data, ticker)

Plots stock prices, EMAs, and buy/sell signals.

e. prepare_data(stock_data)

Prepares the stock data for linear regression by converting dates to numerical format.

f. train_linear_regression(X, y)

Trains a linear regression model using the stock data.

g. predict_future_prices(model, last_day, last_day_date, prediction_days=50)

Predicts future stock prices using the trained linear regression model.

h. plot_predictions_separately(stock_data, future_predictions, ticker)

Plots the historical and predicted stock prices separately.

Output:

- **Buy/Sell Signals**: The script generates buy/sell signals based on the EMA crossover strategy.
- **Stock Price Plot**: Visualization of historical stock prices, EMAs, and the generated buy/sell signals.
- **Predicted Prices**: Future stock price predictions for the next 50 days are displayed in a separate plot.

Model 2: Stock Price Prediction with Stacked LSTM and Mock Trading Environment

Input:

1. **Stock Data**: Time series data (e.g., daily closing prices) for a given stock ticker. In this case, AAPL (Apple Inc.) from 2014 to 2024.
2. **Technical Indicators**:
    - **MACD (Moving Average Convergence Divergence)**: Measures momentum and trends in stock prices.
    - **RSI (Relative Strength Index)**: Measures the speed and change of price movements.
3. **LSTM Model Input**: Sequences of past stock prices, typically spanning 100 days (time_step = 100), to predict the next closing price.

- Run the script directly after setting the stock ticker in the __main__ block.

Execution:

1. **Data Collection**:
    - Fetch stock data using yfinance and resample it to fill missing values.
2. **Feature Engineering**:
    - Compute MACD, RSI, and other relevant indicators.
    - Scale the features to [0, 1] range using MinMaxScaler to improve the performance of the LSTM model.
3. **Model Training:**

    The LSTM model used in this project is built using TensorFlow's Keras API. The model consists of stacked LSTM layers followed by a Dense output layer to predict future stock prices.

    1. **Normalization**: The stock data is scaled between 0 and 1 using MinMaxScaler for better model performance.
    2. **Data Splitting**: The dataset is split into training and test sets.

3. **Model Definition**: A Sequential model with multiple LSTM layers is defined, with dropout regularization to prevent overfitting.
4. **Callbacks**: EarlyStopping and ReduceLROnPlateau callbacks are used to optimize the training process by stopping early when no improvements are detected and reducing the learning rate when necessary.
5. **Evaluation**: The trained model is evaluated using the Root Mean Squared Error (RMSE) metric and plotted against the actual stock prices.

4. **Prediction**:
   - After training, the model predicts stock prices on both training and test datasets.
   - Predictions are inverse-transformed back to the original price scale for evaluation.
5. **Evaluation**:
   - **RMSE (Root Mean Squared Error)**: Used to evaluate the performance of the model on both scaled and original data.

Dependencies:

- **Python Libraries**:
  - numpy, pandas: For data manipulation.
  - scikit-learn: To scale data and compute errors.
  - tensorflow.keras: For building and training the LSTM model.
  - yfinance: To fetch stock data.
  - matplotlib, seaborn: For plotting and visualizing data.
  - pandas_datareader: For additional data collection.
  - datetime: For date and time handling.

You can install these dependencies using the below command:

! pip install numpy pandas scikit-learn tensorflow matplotlib yfinance pandas-datareader seaborn

About:

- **Model**: A stacked LSTM neural network is used for time-series prediction.
- **Purpose**: Predict future stock prices using technical indicators and past price sequences.
- **Prediction Horizon**: The model predicts one step (next closing price) based on the past 100 days.

Trading Strategy

This strategy uses a simple crossover mechanism with technical indicators like MACD and RSI:

- Buy Condition: The environment buys the stock when the predicted price is higher than the current price, signaling an upward trend.
- Sell Condition: The environment sells the stock when the predicted price is lower than the current price, signaling a downward trend.
- Profit Calculation: The environment tracks profit by keeping a record of each buy and sell action, allowing a rough estimate of the performance based on the predicted prices.

Mock Trading Simulation

A simple mock trading strategy can be applied based on the crossover points:

1. **Initial Capital**: Assume a starting capital, say $10,000.
2. **Buy Signal**: Buy shares when the crossover indicates "Buy."
3. **Sell Signal**: Sell shares when the crossover indicates "Sell."
4. **Tracking PnL**: Keep track of the portfolio value by adjusting the number of shares held and cash balance after each transaction.
5. **Cash and Stock Balances**: Track cash, stock holdings, and overall portfolio value over time.

Visualization

- **Actual vs. Predicted Prices**: A plot comparing the actual stock prices to the model's predicted prices for both the training and test datasets.

Functions

- **Data Preparation**:
  - **get_ticker_data():** Fetch stock data.
  - **clean_ticker_data():** Clean and format data.
  - **calculate_macd() & calculate_rsi():** Compute technical indicators.
  - **prepare_data():** Convert time-series data into sequences for LSTM.
- **Model Functions**:
  - **prepare_data():** Prepares the data in sequences for the LSTM model.
  - **LSTM model:** Sequential model with three LSTM layers.
- **Evaluation & Plotting**:
  - **plot_stock_predictions():** Plots the actual vs predicted stock prices.
  - **plot_correlation_matrix():** Plots the correlation matrix of features.

Output:

After running the notebook, you should see:

- A trained LSTM model predicting future stock prices.
- A plot comparing actual vs predicted prices.
- Simulated trading results based on the predicted prices to assess profitability.
- "Buy" and "Sell" signals generated from crossover points, plotted with stock prices.
- Heatmap showing correlations between features (MACD, RSI, etc.).