Trading Strategy: Utilizing LSTM for Predictive Trading

Introduction

For this project, I explored a trading strategy that leverages LSTM neural networks to predict forex price movements. LSTM models are particularly suited for time series forecasting due to their ability to capture long-term dependencies and patterns in sequential data.

Objective

The primary objective of this strategy is to predict future returns based on historical price data and other relevant features, allowing for informed trading decisions that capitalize on anticipated price movements.

Data Collection

I would start by gathering historical stock price data, including:

- **Open, High, Low, Close (OHLC)** prices
- **Technical indicators** (e.g., moving averages, RSI, MACD)

Data Preprocessing

Before feeding the data into the LSTM model, I would perform the following preprocessing steps:

1. **Feature Engineering**: By reducing the number of features, PCA helps mitigate the risk of overfitting, especially in models like LSTM that can be sensitive to noise.
2. **Normalization**: Scale the data to a range suitable for LSTM training (e.g., 0 to 1) to improve model convergence.
3. **Time Series Splitting**: Split the dataset into training, validation, and test sets while preserving the time order.

LSTM Model Development

1. **Model Architecture**: I designed a multi-layer LSTM model with:
    - Input layer to accept the features.
    - Bidirectional LSTM layers to capture temporal dependencies.
    - Dropout layers to prevent overfitting.

- A dense output layer with a single neuron for price prediction.

2. **Training**: The model would be trained using the training dataset, employing techniques such as:
    - **The Huber Loss:** a robust loss function used in regression problems, particularly when dealing with outliers. It combines the properties of both the Mean Squared Error (MSE) and Mean Absolute Error (MAE), making it particularly effective for training models in financial contexts where data may contain anomalies.
    - **Adam optimizer** for efficient training.

Strategy Implementation

Once the model is trained and validated, I have implemented the trading strategy:

1. **Risk Management**: Establish risk management rules, such as stop-loss and dynamic position size.
2. **Backtesting**: Test the strategy using historical data to evaluate its performance, adjusting parameters as necessary based on results.

Further Development

The current model relies solely on past data for training, making it static and necessitating frequent retraining. In the future, I plan to implement a dynamic training feature that allows the model to evolve over time. Additionally, I aim to incorporate more complex attention mechanisms to enhance time series predictions. I also intend to explore the use of autoencoders for feature extraction as a replacement for PCA. Furthermore, the dataset lacks certain information relevant to the true market context, such as headwinds, which will require deeper investigation.