**Project Approach and Execution Summary**

The objective of this project was to develop a robust strategy for predicting buy and sell signals using technical indicators and machine learning, specifically targeting the NIFTY index.

**Phase 1: Initial Exploration with XGBoost**

The project began with an attempt to use the XGBoost Classifier to predict buy and sell signals based on a range of technical indicators. This process not only helped in building a basic predictive framework but also offered valuable insights into which indicators held significant predictive power. Through this phase, I identified that momentum and volatility-based indicators had the most relevance for signal generation.

However, despite its popularity, the XGBoost model did not yield consistently reliable signals in my use case. Its results were difficult to interpret, and the model struggled with the temporal nature of financial time series data. This prompted a shift toward a more time-series-oriented modeling approach.

**Phase 2: Transition to LSTM for Time Series Prediction**

Given the limitations of XGBoost in this context, I moved to Long Short-Term Memory (LSTM) networks, which are better suited for sequential data and forecasting. I referred to Stephen Jansen's book on financial machine learning to guide model architecture and parameter optimization.

For training, I used 63 past observations to predict the next day's price, and the resulting predictions were added as an additional feature in the spot_signals_2023.csv dataset. These predictions later served as one of the signals in the overall trading strategy.

**Phase 3: Strategy Development and Indicator Selection**

After evaluating several indicators, I finalized a combination of momentum and volatility-based signals:

- **EMA 200 – RSI 14 Crossover**: Chosen for its stable performance in backtests under favorable conditions.
- **EMA 200 – EMA 50 Crossover**: Considered but ultimately not preferred due to comparatively lower performance.
- **MACD (Moving Average Convergence Divergence)**
- **ATR (Average True Range)**

To improve responsiveness, I modified the typical RSI thresholds (70/30) to a 50/50 range for signal generation. Additionally, if the LSTM model predicted a price higher than the current day's, a supporting buy signal was recorded.

**Phase 4: Signal Weighting and Final Logic**

Given the strength of the EMA 200 – RSI 14 signal in providing consistent returns with manageable risk, it was assigned the highest weight in the final signal aggregation logic. MACD and ATR, which acted as secondary confirmations, received moderate weights. The LSTM prediction signal, while informative, was given the lowest weight due to its higher uncertainty in real-time performance.

**Phase 5: Backtesting and Data Handling**

Managing the full dataset proved challenging in a Colab environment. To handle large-scale data efficiently, I split the primary dataset into 10 equal parts (each representing ~10% of the total data) and processed them individually. This helped streamline the backtesting process and optimize memory usage.

The backtesting engine was iteratively developed and refined. Some runs were computationally intensive—taking up to three hours—yet returned little to no output. Eventually, one dataset yielded a modest return of 0.8% per month, which became the baseline for further strategy refinement. I then adjusted some signal generation criteria to allow more flexibility and better opportunity capture during the backtest.

**Final Outcome and Reflection**

One of the major challenges I encountered was managing the full dataset within the memory constraints of Google Colab. To address this, I divided the dataset into 10 smaller parts and mounted them individually. This was a time-intensive task, but essential to make the backtesting process feasible. To maintain the chronological integrity of the trades, I wrote a loop within the backtesting engine to iterate through datasets 1 to 10 sequentially. This ensured the continuity of date and timestamp data across different splits, which is critical for time series analysis.

However, some datasets had structural or formatting issues that affected performance, and I had to drop them mid-process to ensure consistent results. After running the full backtest on the remaining datasets, the cumulative PnL stood at approximately -₹5,00,000, which indicated that the strategy was not yet viable in its current form.

Upon reviewing the output, I noticed an issue in how the backtesting engine was handling open trades. The PnL was being printed continuously until the trade was exited, which led to a biased representation of returns—especially toward the end of the dataset. To mitigate this, I developed a custom script to reduce the effect of this bias. The logic involved scanning for unique values in the PnL column and checking whether those values were repeated across the next 15 rows. If continuous repetition was detected (indicating an unclosed trade), I replaced those entries with zero, thereby adjusting the final PnL to reflect a more realistic outcome.

While the strategy did not yield profitable results, the project deepened my understanding of:

- Real-world data engineering constraints,
- The importance of clean and continuous time series data,
- Handling model bias and inaccuracies in backtesting,
- Debugging large-scale iterative systems, and
- Developing rule-based corrections for output anomalies.

This experience reinforced the fact that quantitative strategy development is inherently iterative. Even though the output wasn't what I initially aimed for, I gained practical, hands-on exposure to building and testing a trading framework end-to-end, which will prove valuable for future financial modeling and algorithmic trading projects.