**Project Report: Indian Stock Trading Bot**

**1. Introduction**

The Indian Stock Trading Bot is an algorithmic trading solution designed to analyze Indian stock market data, generate buy and sell signals, and execute trades. This bot integrates financial indicators, machine learning models, and risk management techniques to provide trading recommendations. It is built using Python and includes advanced features like data fetching from TradingView, indicators like the Andean Oscillator and Trend Signal Indicator, risk-reward management, and Telegram notifications for real-time updates.

**2. Objectives**

The main objectives of this project are:

* Fetch historical stock data from the Indian stock market.
* Calculate technical indicators such as the Andean Oscillator and Trend Signal.
* Generate trading signals based on predefined conditions.
* Integrate machine learning models to predict future close prices.
* Execute trades based on risk-reward strategies (1:3 ratio).
* Provide real-time alerts through Telegram.

**3. Data Fetching**

To begin the analysis, the bot retrieves historical data for Indian stocks from TradingView using the tvDatafeed API. The stocks are sourced from a pre-defined list (indian\_companies). The function fetch\_data\_with\_retries ensures reliable data retrieval by attempting multiple retries in case of connection issues.

**Key Libraries Used**:

* tvDatafeed: For accessing stock data.
* pandas: To handle and process the stock data in a DataFrame format.
* numpy: For numerical operations.

**4. Technical Indicators**

Two technical indicators are computed:

**4.1 Andean Oscillator**

The Andean Oscillator calculates the momentum of stock prices over a rolling window. The momentum is the difference between the current closing price and the closing price ten periods earlier. The oscillator provides buy and sell signals based on its value:

* **Buy Signal**: If the oscillator value is positive.
* **Sell Signal**: If the oscillator value is negative.

**4.2 Trend Signal Indicator**

The Trend Signal Indicator detects upward or downward trends by calculating moving averages, RSI (Relative Strength Index), and CMO (Chande Momentum Oscillator). It identifies buy and sell signals by comparing these trends:

* **Buy Signal**: When a trend shifts from downward to upward.
* **Sell Signal**: When a trend shifts from upward to downward.

**5. Risk-Reward Management**

To manage risk, the bot calculates stop-loss and take-profit levels based on a 1:3 risk-reward ratio:

* **Stop-Loss**: Set at the lowest low of the last 4 candles for long positions and the highest high of the last 4 candles for short positions.
* **Take-Profit**: Set at three times the risk from the entry point to the stop-loss level.

This risk management approach aims to protect the trader from excessive losses while ensuring that profitable trades are maximized.

**6. Machine Learning Model**

The bot incorporates machine learning to predict the future close price of a stock:

A RandomForestRegressor is trained to predict future closing prices of stocks. The features include the opening price, highest price, lowest price, volume, and Andean Oscillator value.

* **Model Training**: The data is split into training and testing sets, with the model being trained on historical data.
* **Model Evaluation**: The model’s performance is evaluated using metrics like Mean Absolute Error (MAE) and accuracy.

The bot computes two predicted prices:

1. Historical Predicted Close Price: Prediction based on the historical data.
2. Real-Time Predicted Close Price: Prediction based on real-time data

**7. Real-Time Prediction**

The trained model predicts future close prices for each symbol in real-time. The predictions are then compared with actual close prices to calculate accuracy.

**8. Trade Information Logging**

The bot prints trading information to the console and sends real-time updates via Telegram. This information includes the date, time, symbol price, predicted close prices, and accuracy, as well as the generated buy/sell signals, stop-loss levels, and take-profit levels.

**9. Final Signal Generation**

After calculating both the technical indicators and machine learning predictions, the bot generates final trading signals:

* **Buy Signal**: Triggered when the Trend Signal and Andean Oscillator agree on upward momentum, and the predicted close price indicates further gains.
* **Sell Signal**: Triggered when both indicators show downward momentum, and the predicted close price suggests a decline.

The bot compares the real-time and historical predicted close prices, and the difference between these prices informs the final decision. The accuracy of the signal is calculated and included in the final message.

**10. Telegram Integration**

The bot sends updates on trading signals, predicted prices, and other relevant details through Telegram using the ‘python-telegram-bot’ library. The bot can notify users of buy and sell opportunities, along with other information like the stop loss, take profit, and prediction accuracy. Messages include detailed trade information such as:

* Date and time of the signal
* Stock symbol and price
* Predicted close prices
* Buy/sell signals, stop loss, and take profit levels
* Signal accuracy percentage

**11. Statistics Calculation**

The bot calculates various performance statistics to evaluate its predictions:

* **Mean Absolute Error (MAE)**: Measures the average magnitude of errors between predicted close prices and actual close prices, providing insight into prediction accuracy.
* **Accuracy**: The percentage of correctly predicted price directions (up or down), calculated as the ratio of correct predictions to total predictions.
* **Signal MAE**: Measures the accuracy of the buy/sell signals by comparing actual and predicted signals. This is expressed as a percentage, where a lower MAE indicates better signal accuracy.

The statistics help in evaluating the effectiveness of both the machine learning model and the trading strategy.

Here's a detailed explanation of the statistics calculated in the provided code:

**11. 1. Total Trades**

Python code

total\_trades = len(df.iloc[-1])

* **Explanation**: This calculates the total number of trades based on the last row of the dataframe df. This could either mean the total number of entries (candles) or trades evaluated at that point in time.

**11. 2. Total Buy Trades**

Python code

total\_buy\_trades = df['buySignal'].sum()

* **Explanation**: This sums up the number of buy signals generated in the dataframe. The buySignal column contains boolean values or 0/1 values indicating if a buy signal was generated for a particular row. The sum gives the total number of buy trades executed.

**11. 3. Total Sell Trades**

Python code

total\_sell\_trades = df['sellSignal'].sum()

* **Explanation**: Similar to buy trades, this calculates the total number of sell signals generated. The sum of the sellSignal column gives the total number of sell trades executed.

**11. 4. Winning Trades**

Python code

win\_trades = df['Take\_Profit'].notna().sum()

* **Explanation**: The bot assumes a trade is a win if the **take profit** level was hit. The .notna() function checks where the Take\_Profit column is not null (i.e., a take profit was executed), and .sum() gives the total count of winning trades.

**11. 5. Losing Trades**

Python code

loss\_trades = df['Stop\_Loss'].notna().sum()

* **Explanation**: Similar to winning trades, a trade is considered a loss if the **stop loss** was hit. The .notna() function checks where the Stop\_Loss column is not null, and .sum() gives the total count of losing trades.

**11. 6. Win Rate**

Python code

win\_rate = (win\_trades / total\_buy\_trades) \* 100 if total\_buy\_trades > 0 else 0

* **Explanation**: The win rate is the percentage of successful buy trades relative to the total number of buy trades. It's calculated as the ratio of winning trades to total buy trades, multiplied by 100 to express it as a percentage. If no buy trades were made, the win rate is set to 0 to avoid division by zero.

**11. 7. Loss Rate**

Python code

loss\_rate = (loss\_trades / total\_sell\_trades) \* 100 if total\_sell\_trades > 0 else 0

* **Explanation**: The loss rate is the percentage of losing sell trades relative to the total number of sell trades. It is calculated similarly to the win rate but for losing trades and sell signals. If no sell trades were made, the loss rate is set to 0.

**11. 8. Total Profit/Loss**

Python code

total\_profit\_loss = df['Take\_Profit'].sum() - df['Stop\_Loss'].sum()

* **Explanation**: This calculates the net profit or loss of the trading strategy by subtracting the sum of all losses (Stop\_Loss) from the sum of all profits (Take\_Profit). If the result is positive, the strategy was profitable overall; if negative, it incurred a loss.

**11. 9. Average Profit Per Trade**

Python code

average\_profit\_per\_trade = df[df['buySignal']]['Take\_Profit'].mean() if total\_buy\_trades > 0 else 0

* **Explanation**: This computes the average profit per buy trade by calculating the mean of the Take\_Profit column for rows where buySignal is True. If no buy trades were made, the average profit is set to 0.

**11. 10. Average Loss Per Trade**

Python code

average\_loss\_per\_trade = df[df['sellSignal']]['Stop\_Loss'].mean() if total\_sell\_trades > 0 else 0

* **Explanation**: This computes the average loss per sell trade by calculating the mean of the Stop\_Loss column for rows where sellSignal is True. If no sell trades were made, the average loss is set to 0.

**11. 11. Maximum Drawdown**

Python code

maximum\_drawdown = df['Stop\_Loss'].min() if not df['Stop\_Loss'].empty else 0

* **Explanation**: The maximum drawdown refers to the largest single loss incurred during trading. This is calculated by finding the minimum value in the Stop\_Loss column, which represents the most significant negative loss (largest drawdown). If there are no stop losses (empty column), the maximum drawdown is set to 0.

**11. 12. Profitable and Loss-Making Symbols**

Python code

profitable\_symbols = stats\_df[stats\_df['Total Profit/Loss'] > 0].shape[0]

loss\_making\_symbols = stats\_df[stats\_df['Total Profit/Loss'] < 0].shape[0]

* **Explanation**: This calculates the number of profitable and loss-making symbols. It checks the Total Profit/Loss column in stats\_df for each symbol and counts how many symbols had positive profits (profitable symbols) and how many had negative profits (loss-making symbols).

**11. 13. Accuracy Percentage**

Python code

accuracy\_percentage = (profitable\_symbols / len(symbols)) \* 100

* **Explanation**: This calculates the accuracy of the trading strategy as a percentage. It divides the number of profitable symbols by the total number of symbols and multiplies by 100 to express it as a percentage. This accuracy represents how many symbols resulted in a net profit out of all symbols traded.

**Summary of Calculated Statistics:**

* **Total Trades**: The number of trades executed.
* **Total Buy Trades**: Number of buy trades.
* **Total Sell Trades**: Number of sell trades.
* **Win Trades**: Number of winning trades (take profit hit).
* **Loss Trades**: Number of losing trades (stop loss hit).
* **Win Rate**: Percentage of successful buy trades.
* **Loss Rate**: Percentage of failed sell trades.
* **Total Profit/Loss**: Net profit or loss of the trading strategy.
* **Average Profit Per Trade**: Average profit for buy trades.
* **Average Loss Per Trade**: Average loss for sell trades.
* **Maximum Drawdown**: The largest single loss incurred.
* **Profitable Symbols**: Number of symbols with positive profits.
* **Loss-Making Symbols**: Number of symbols with negative profits.
* **Accuracy Percentage**: Percentage of symbols that were profitable.

**12. Final Implementation**

The bot operates on a loop, continuously fetching data for multiple Indian stocks, calculating indicators, and sending alerts. It uses a structured approach for each stock, ensuring that all calculations are accurate and timely.

**13. Conclusion**

This Indian Stock Trading Bot is a comprehensive solution that leverages technical indicators, machine learning, and real-time notifications to assist traders in making informed decisions. By integrating the Andean Oscillator and Trend Signal Indicator with robust risk management and prediction models, the bot aims to improve trading performance and reduce losses.