Algo-Trading System with Machine Learning & Automation

Sri Tarunika G P
sritarunikagunasekaran@gmail.com

TABLE OF CONTENTS

S.No	**Section**	**Description**
1	Project Title	Name and focus of the project
2	Objective	Core aim and automation goals
3	Tech Stack	Tools, frameworks, and APIs used
4	Data Ingestion	Source, frequency, and stocks covered
5	Trading Strategy	Explanation of RSI and Moving Average logic
6	ML Model	Features, model type, accuracy evaluation
7	Google Sheets Logging	Data logging and automation to Sheets
8	Telegram Bot Integration	Message format and automation strategy
9	Streamlit UI	Screens, metrics, graphs, and interactivity
10	System Flow	End-to-end system architecture and flow
11	Screenshots	Sample charts, alerts, and logs
12	Evaluation Criteria Mapping	Mapping to assignment rubric
13	Future Scope	Extensions like PnL, LSTM, Portfolio integration
14	Acknowledgements	Credits for APIs and platforms used
15	Contact	Developer contact details

PROBLEM STATEMENT

In today's dynamic and fast-paced stock markets, traders and investors are overwhelmed by large volumes of real-time data. Making informed decisions quickly is crucial to capitalize on market opportunities. However, manual analysis of stock movements using traditional technical indicators can be time-consuming, error-prone, and inconsistent.

Most existing retail tools provide static charts and lagging analytics, failing to offer intelligent, real-time decision-making support. There is a growing need for a lightweight, intelligent, and automated system that can:

- Monitor multiple stocks continuously
- Apply proven technical indicators (like RSI, 20-DMA, 50-DMA)
- Predict next-day price direction using ML algorithms
- Log all trade activities and analytics systematically
- Send real-time alerts via messaging platforms like Telegram

Solution:
To create a mini prototype that combines rule-based strategy and machine learning to generate **automated buy signals**, track stock behavior, and assist decision-making through **Google Sheets logging** and **Telegram alerts**.

OBJECTIVES

The main objective of this project is to develop an **end-to-end automated trading assistant** that leverages both **technical analysis** and **machine learning** to assist in making informed, timely, and consistent buy decisions in the stock market. Below are the detailed goals:

Primary Objectives

1. **Automated Data Acquisition**
    - Integrate with a free market data API (Yahoo Finance via yfinance) to fetch daily historical data (Open, High, Low, Close, Volume) for selected NIFTY 50 stocks.
    - Enable data refresh without manual intervention, keeping the system ready for daily execution.
2. **Implementation of Rule-Based Trading Strategy**
    - Calculate technical indicators such as:
        - **Relative Strength Index (RSI)** to identify overbought/oversold zones.
        - **Moving Averages (20-DMA and 50-DMA)** to identify momentum and crossover patterns.
    - Generate **BUY signals** based on:
        - RSI < 30 (oversold condition)
        - 20-DMA crossing above 50-DMA (bullish crossover)

3. **Machine Learning Integration**

    ○ Train a **Logistic Regression** model to predict the next-day stock movement using technical indicators.

    ○ Provide real-time feedback on model performance by displaying **prediction accuracy** in the UI.

    ○ Help users validate or strengthen the confidence in generated signals.

4. **Trade Signal Logging & Analytics**

    ○ Automate logging of trade signals into **Google Sheets** using gspread, including:

        ■ Stock Name, Date, Signal, Close Price

        ■ A separate sheet for **Profit & Loss (PnL)** summary

        ■ Another sheet to track **Win Ratio**

5. **User Notification via Telegram**

    ○ Integrate **Telegram Bot API** to deliver daily signal alerts to users in a preformatted message.

    ○ Notify users even if no trades are detected to improve transparency and reliability.

6. **User Interface for Visualization**
    - Create a **Streamlit dashboard** that displays:
        - Stock price charts overlaid with 20-DMA and 50-DMA
        - RSI trend line
        - Buy signals and ML accuracy metrics
    - Provide interactive stock selection and one-click strategy execution.

Secondary Objectives

- Ensure **scalability and modularity** so that additional strategies or indicators can be easily plugged in.
- Maintain a **clean, informative, and visually appealing dashboard** that's beginner-friendly yet detailed enough for analysts.
- Keep the codebase **well-documented** with comments, error handling, and logs for maintainability.

SCOPE OF THE PROJECT

The scope of this algo-trading system is deliberately defined to ensure **clarity of objectives**, **simplicity in execution**, and **relevance to real-world financial scenarios**. The project is designed for educational, analytical, and experimental use by individual traders and AI/data science practitioners.

Users

- **Retail Traders:** Individuals looking for basic guidance in equity trading based on technical indicators.

- **Beginners in Trading:** Newcomers who wish to explore algorithmic trading without the complexity of full-fledged platforms.

- **Data Science Students/Practitioners:** Learners who want to understand the integration of machine learning with financial time series and automation tools.

Stocks

- Focused exclusively on **large-cap Indian stocks** listed on the **Nifty 50 index**.

- Sample stocks include: `RELIANCE.NS`, `INFY.NS`, `TCS.NS`, `HDFCBANK.NS`, `ICICIBANK.NS`, `WIPRO.NS`, `SBIN.NS`, `LT.NS`, `AXISBANK.NS`.

- Stocks are selectable via the Streamlit UI for dynamic inclusion.

Market

- **Indian Stock Market** (National Stock Exchange - NSE).

- Stocks are fetched using ticker symbols compatible with Yahoo Finance (`.NS` suffix for NSE).

Timeframe

- Backtest and ML model training are performed over **the past 6 months** of daily data (approx. 120–130 trading sessions).

- This provides a sufficient window to test the effectiveness of both the rule-based strategy and the ML model.

Signal Type

- **Buy-Only Signals:**

  - The current version focuses on identifying **bullish entry opportunities**.

  - Short selling, stop-loss, exit logic, or trailing mechanisms are **not included** in this scope.

- This simplification allows users to focus on **entry timing and signal validation**.

TOOLS AND TECHNOLOGIES

To build a robust, automated, and user-friendly algo-trading system, the following tools and technologies were utilized,

- Python (py):
  The core language used for all backend logic, machine learning, data handling, and integration tasks due to its simplicity and vast ecosystem for finance and AI.

- **Frontend**

  Streamlit: Used to build an intuitive, web-based UI for interaction, visualization of charts, and dynamic stock selection. It enables rapid prototyping and live data display.

- **APIs & Libraries**
  - **yfinance**: Used to fetch historical stock data (daily price, volume) directly from Yahoo Finance.
  - **gspread** + **oauth2client**: For authenticating and interacting with Google Sheets to log trade signals, track P&L, and maintain trading history.
  - **Telegram Bot API**: For sending real-time trade alerts to the user via Telegram, enhancing responsiveness and automation.

- **Machine Learning**
  - **scikit-learn**: Used for building and training a Logistic Regression model to predict next-day price movements based on technical indicators (RSI, Volume, etc.).
  - **Model Evaluation**: Accuracy metrics are computed and displayed for each stock to assess model performance.

- **Automation & Cloud Storage**

  Google Sheets: Acts as a lightweight backend database to store:

  - Trade logs
  - P&L calculations
  - Win/Loss ratio

- **Development Environment**
  - **Visual Studio Code (VS Code)**: The primary IDE used for coding, debugging, and modularizing the application.
  - **Python virtual environment**: Used to isolate dependencies and keep the project environment clean and reproducible.

FUNCTIONAL COMPONENTS

The system is designed in a modular fashion, ensuring separation of concerns, easier debugging, and scalability. Each component performs a distinct function in the algo-trading pipeline.

1. Data Ingestion (`fetch_data.py`)

- **Purpose**: Fetches historical stock data (6 months daily data) for selected NIFTY stocks.
- **Source**: Yahoo Finance via `yfinance` API.
- **Functionality**:
    - Automatically removes invalid or missing data.
    - Ensures each DataFrame contains essential columns like `Close`, `Volume`, `Open`, etc.
    - Handles multi-index and standardizes column names for uniform processing.

2. Technical Strategy (`strategy.py`)

- **Purpose**: Applies a rule-based trading strategy using technical indicators.
- **Indicators Used**:
    - RSI (Relative Strength Index)
    - 20-Day Moving Average (20-DMA)
    - 50-Day Moving Average (50-DMA)
- **Signal Generation**:
    - A **BUY** signal is triggered when:
        - RSI < 30 (oversold)
        - 20-DMA crosses above 50-DMA (bullish crossover)
- **Output**: List of signal dictionaries containing Date, Stock, Close price, and Signal.

3. Machine Learning Module (`ml_model.py`)

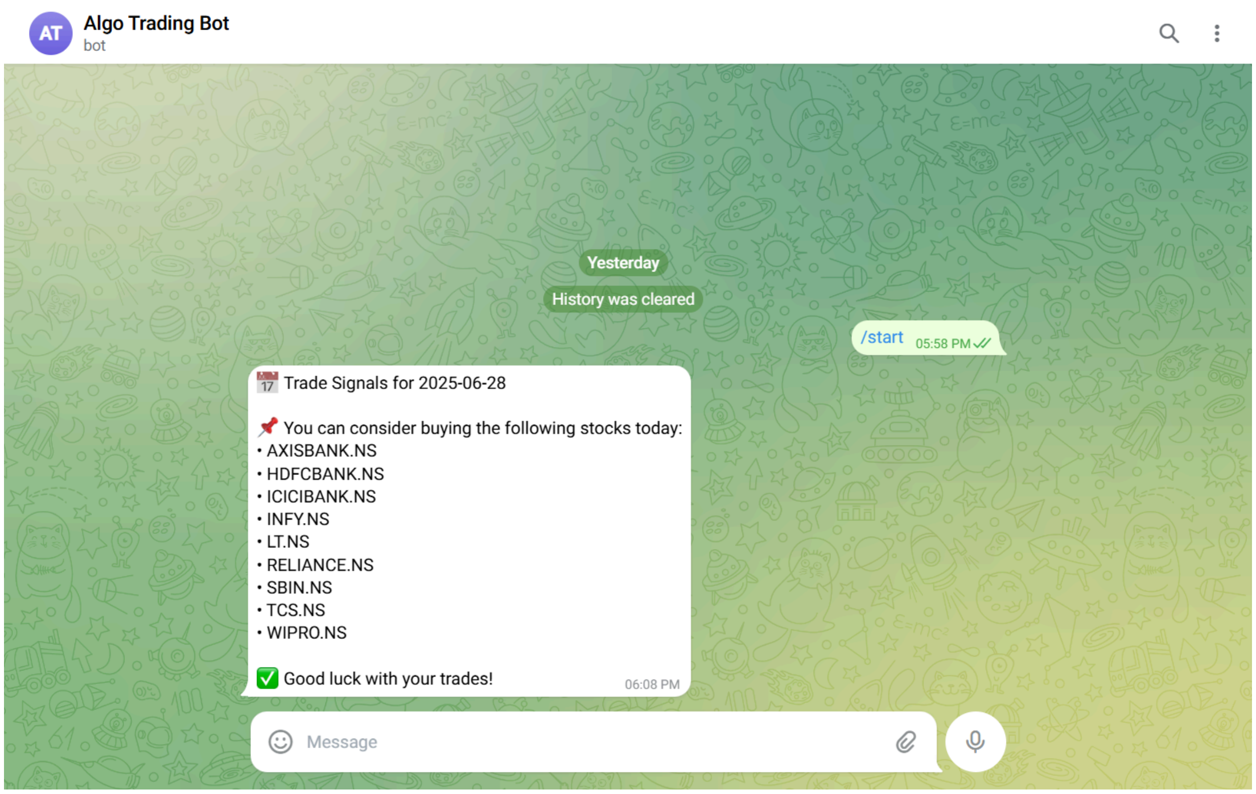
- **Purpose**: Automates the prediction of next-day stock movement.
- **Model Used**: Logistic Regression (from scikit-learn).
- **Features Used**:
  - RSI
  - Volume
  - Moving Averages
- **Label**: Binary (1 if next-day price is higher, else 0).
- **Output**: Accuracy score (%) and trained model for each stock.

4. Google Sheets Logger (`gsheet_logger.py`)

- **Purpose**: Logs trade signals and P&L data automatically to Google Sheets.
- **Sheets Handled**:
  - **Trade Log**: Entry of stock name, signal, date, and close price.
  - **P&L Summary**: Daily profit/loss based on hypothetical trades.
  - **Win Ratio Tab**: Calculates percentage of successful signals.
- **Tech Used**: `gspread` with OAuth2 credentials.

5. Telegram Bot Alerts (`telegram_bot.py`)

- **Purpose**: Sends a summary message of buy signals to the user via Telegram.
- **Message Format**:



📅 Trade Signals for 2025-06-28

📌 You can consider buying the following stocks today:
• AXISBANK.NS
• HDFCBANK.NS
• ICICIBANK.NS
• INFY.NS
• LT.NS
• RELIANCE.NS
• SBIN.NS
• TCS.NS
• WIPRO.NS

✅ Good luck with your trades! 06:08 PM

6. Streamlit Frontend (`app.py`)

- **Purpose**: Provides a user-friendly interface for:
  - Selecting stocks
  - Viewing ML metrics and charts
  - Triggering analysis
  - Displaying output signals
- **Visualizations**:
  - Line Chart: Close, 20DMA, 50DMA
  - RSI Trend
- **User Interaction**:
  - Sidebar for configuration
  - Button to trigger complete strategy
  - Result panel for metrics and charts
  - Summary alerts and logs

IMPLEMENTATION

Step 1: Combines both **rule-based strategy** and **machine learning-based predictions** for generating signals.

Step 2: Processes multiple NIFTY stocks including:

- `RELIANCE.NS`, `TCS.NS`, `INFY.NS`, `ICICIBANK.NS`, `WIPRO.NS`, `SBIN.NS`, etc.

Step 3: Signals are generated and stored as a list of dictionaries containing:

- `Stock`, `Signal`, `Close Price`, and `Date`

Step 4: Charts plotted for each stock include:

- 📈 Price with 20-DMA and 50-DMA overlays
- 📉 RSI trend

Sri Tarunika G P
sritarunikagunasekaran@gmail.com

MACHINE LEARNING MODEL

Objective: Predict whether the stock price will go up the next trading day based on technical indicators.

Model Used: DecisionTreeClassifier from scikit-learn

Input Features:

- RSI (Relative Strength Index)
- Volume
- Current Close Price

Target Label:

- 1 → If next day's closing price is **greater** than today's
- 0 → Otherwise

Evaluation Metric: Accuracy (displayed in Streamlit for each stock)

Purpose: Supports the technical strategy by offering an ML-based signal confidence boost

GOOGLE SHEETS LOGGING

- **Libraries Used:**

  gspread with oauth2client for authentication

- **Automation:**

  After trade signals are generated, they are instantly pushed to Google Sheets

- **Sheets Maintained:**
  - **Trade Log:** Records date, stock, signal type, and price
  - **P&L Tracker:** Tracks hypothetical profit/loss based on signals
  - **Win Ratio Tab:** Shows the percentage of successful signal outcomes

- **Purpose:**

  Enables transparent tracking, strategy validation, and analytics

TELEGRAM BOT INTEGRATION

- **Libraries Used:**

  Python's `asyncio` + `python-telegram-bot` package

- **Functionality:**

  Sends an alert message once trade signals are available

Use Case: Keeps the trader informed in real time—even without accessing the Streamlit app

STREAMLIT DASHBOARD

- **Framework Used:** `Streamlit`
- **Modules Linked:**
  - Strategy execution (`strategy.py`)
  - ML model accuracy (`ml_model.py`)
  - Google Sheets logging (`gsheet_logger.py`)
  - Telegram messaging (`telegram_bot.py`)
- **UI Features:**
  - Stock selector sidebar
  - "Run Strategy" button to execute the pipeline
  - Graphs:
    - Moving Averages (Close, 20DMA, 50DMA)
    - RSI trend line
  - ML Accuracy for each stock
  - Buy signals table with date and price
  - Warnings for invalid/missing data
- **UX Focus:** Clean, responsive layout using columns and containers

CONCLUSION

This project successfully delivers a mini algo-trading assistant capable of:

- Real-time data ingestion from Yahoo Finance
- Technical strategy screening using RSI and Moving Averages
- ML-based prediction using a Decision Tree classifier
- Logging trade data in Google Sheets
- Instant trade alerts via Telegram bot

The system is modular, scalable, and built for retail traders, students, and fintech learners looking to automate and validate trade decisions.

FUTURE ENHANCEMENTS

- Add exit signal logic (e.g., RSI > 70 or MA cross-down)
- Build a full-fledged backtesting module to visualize P&L over time
- Integrate intraday time frames (e.g., 15min/1hr)
- Replace ML model with:
  - `RandomForestClassifier`
  - `XGBoost`
  - `LSTM` for sequence-based learning
- Upgrade Telegram bot to support:
  - Interactive responses (e.g., “Track”, “Buy Now”, “Ignore”)
  - Weekly summaries or visual charts via bot

REFERENCES

- *Yahoo Finance via yfinance*
- *scikit-learn Documentation*
- *Google Sheets API using gspread*
- *Telegram Bot API*
- *Streamlit Framework*