Technical Report: Automated Algorithmic Trading System with Telegram Integration

Abstract

This report presents the design, development, and implementation of an Automated Algorithmic Trading System. The system integrates stock data retrieval, technical analysis, machine learning predictions, backtesting, result logging to Google Sheets, and real-time Telegram alerts. The solution is built using Python and leverages free and widely accessible resources such as yfinance, Google Sheets API, and Telegram Bot API to ensure cost-effectiveness and scalability.

Table of Contents

---

1. Introduction

Algorithmic trading, also known as algo trading, is the process of using computer algorithms to execute trading strategies automatically. This system significantly improves trade execution speed, accuracy, and frequency. The primary goal of this project is to build a fully automated trading framework that can analyze stock data, identify trading signals, backtest strategies, predict market trends using machine learning, and send real-time alerts, all without manual intervention.

---

2. System Architecture

The system is modular and is composed of the following interconnected components: - **Data Ingestion Module:** Retrieves historical stock data. - **Strategy Module:** Calculates technical indicators and generates

buy signals. - **Backtesting Module:** Evaluates the performance of the trading strategy. - **Machine Learning Module:** Trains and tests predictive models on historical data. - **Google Sheets Module:** Stores trade logs for performance analysis. - **Telegram Alerts Module:** Sends real-time notifications for signals and errors. - **Main Orchestration Module:** Coordinates the execution of all other modules.

---

3. Modules Overview

Each module plays a specific role in the system and communicates with others to maintain an efficient workflow.

- **data_ingestion.py:** Handles data retrieval.
- **strategy.py:** Computes indicators and generates signals.
- **backtest.py:** Runs strategy performance simulations.
- **ml_model.py:** Applies machine learning predictions.
- **google_sheets.py:** Updates Google Sheets with trade results.
- **telegram_alerts.py:** Sends alerts and notifications.
- **main.py:** Runs the entire process in sequence.

---

4. Data Ingestion

The `data_ingestion.py` module uses the `yfinance` Python library to download daily historical stock data for specified tickers. It features error handling that sends a Telegram alert if the data retrieval fails or returns empty datasets.

Key Functions: - `fetch_stock_data(ticker, period='6mo', interval='1d')`: Downloads the required data and sends error alerts if the data is missing.

---

5. Trading Strategy Implementation

The `strategy.py` module implements technical analysis indicators such as: - **Relative Strength Index (RSI):** Measures the magnitude of recent price changes to evaluate overbought or oversold conditions. - **20-Day and 50-Day Moving Averages (DMA):** Tracks average prices over specified periods.

The strategy generates a buy signal when the RSI is below 30 and the 20DMA is above the 50DMA.

Key Functions: - `calculate_rsi(data)`: Computes the RSI values. - `calculate_moving_averages(data)`: Calculates 20DMA and 50DMA. - `generate_signals(data)`: Generates buy signals and sends Telegram alerts.

---

6. Backtesting Engine

The `backtest.py` module simulates trades based on the generated signals to evaluate the strategy's profitability.

Key Features: - Entry and exit points are defined by the presence and absence of buy signals. - Calculates trade profit or loss.

Key Function: - `backtest_strategy(data)`: Returns a list of all simulated trades.

---

7. Machine Learning Model Integration

The `ml_model.py` module uses a Decision Tree Classifier to predict price movements based on historical features like RSI, 20DMA, 50DMA, and volume.

Key Functions: - `train_ml_model(data)`: Trains the model and returns the prediction accuracy. - Sends Telegram alerts if the data is insufficient for model training.

---

8. Google Sheets Integration

The `google_sheets.py` module logs the backtest results into Google Sheets for tracking and analysis.

Key Features: - Automatically creates sheets if they don't exist. - Updates trade logs with entry/exit dates, prices, and profits. - Sends Telegram alerts for missing sheets or empty updates.

Key Function: - `update_google_sheet(trades, sheet_name)`

---

9. Telegram Alerts Integration

The `telegram_alerts.py` module connects the system with the Telegram API to deliver real-time alerts directly to the user's mobile device.

Key Features: - Sends alerts for: - Data fetch errors - Signal detections - Google Sheet updates - Model training accuracy - Process completions

Key Function: - `send_telegram_message(message)`

---

10. Workflow Pipeline

The entire process is orchestrated through `main.py`: 1. Loops through the list of stock tickers. 2. Fetches stock data. 3. Applies RSI and moving average strategies. 4. Generates buy signals and sends alerts. 5. Backtests trades. 6. Logs results to Google Sheets. 7. Trains the ML model and reports accuracy. 8. Sends a final Telegram notification upon completion.

---

11. Error Handling and Robustness

The system is designed to be highly robust: - Skips stocks with invalid or missing data. - Handles empty datasets gracefully. - Automatically creates missing Google Sheets. - Provides real-time feedback for every critical failure or success.

---

12. Advantages and Limitations

Advantages

- Fully automated pipeline.
- Real-time alerts through Telegram.
- Uses free, accessible APIs and tools.
- Modular, scalable, and easy to maintain.

Limitations

- Simple decision tree model; can be improved with advanced ML algorithms.
- Strategy based only on RSI and DMA; multi-indicator strategies could improve accuracy.
- Google Sheets API rate limits.

---

13. Future Scope

- Add support for more advanced technical indicators.
- Integrate more complex ML models (Random Forest, XGBoost).
- Expand to multi-asset trading.
- Incorporate live order execution via broker APIs.
- Add automated scheduling using cron jobs or cloud-based functions.

---

14. Conclusion

This Automated Algorithmic Trading System successfully integrates stock data analysis, technical strategy implementation, machine learning prediction, backtesting, Google Sheets logging, and Telegram alerts into

a cohesive workflow. It provides a strong foundation for further development into a fully operational trading bot that can execute live trades and adapt to more complex strategies.