AI-POWERED REAL-TIME TRADING SYSTEM FOR STOCK MARKET USING MACHINE LEARNING

PROJECT REPORT

submitted by:

ANEESH V R

TRV23IDTE01

to

the APJ Abdul Kalam Technological University

in partial fulfilment of the requirements for the award of the Degree

of

Master of Technology in Translational Engineering

Department of Translational Engineering

Government Engineering College, Barton Hill

Thiruvananthapuram

JULY 2025

DECLARATION

I, the undersigned, hereby declare that the project titled "AI-Powered Real-Time Trading System For Stock Market Using Machine Learning" submitted in partial fulfilment of the requirements for the award of degree of Master of Technology of the APJ Abdul Kalam Technological University, Kerala, is a bonafide work done by me under the supervision of Dr. Poushali Pal Assistant Professor, TPLC.This submission represents my ideas in my own words and where ideas or words of others have been included, I have adequately and accurately cited and referenced the original sources. I also declare that I have adhered to ethics of academic honesty and integrity and have not misrepresented or fabricated any data or idea or fact or source in my submission.I understand that any violation of the above will be a cause for disciplinary action by the institute and/or the University and can also evoke penal action from the sources which have not been properly cited or from whom proper permission has not been obtained. This report has not been previously formed the basis for the award of any degree, diploma, or similar title of any other university.

(Signature)

ANEESH V R

Place: Thiruvananthapuram

Date: 20/05/2025

DEPARTMENT OF TRANSLATIONAL ENGINEERING

GOVERNMENT ENGINEERING COLLEGE, BARTON HILL

THIRUVANANTHAPURAM

CERTIFICATE

This is to certify that the project report titled AI-Powered Real-Time Trading System For Stock Market Using Machine Learning submitted by Aneesh V R, Register Number: TRV23IDTE01, to the APJ Abdul Kalam Technological University in partial fulfilment of the requirements for the award of the Degree of Master of Technology in Translational Engineering, is a bonafide record of the project work carried out by him in the Department of Translational Engineering, Government Engineering College, Barton Hill, Trivandrum, under the guidance of Dr. Paushali Pal , Assistant Professor, TPLC, during the academic year 2024–2025.

P. Pal

Dr. Poushali Pal

Internal Supervisor

P. Pal

Dr. Poushali Pal

Project Coordinator

Dr. Suja R

Head of the Department

ACKNOWLEDGEMENT

A lot of effort and hard work has been put into the successful completion of this project work. However, it would not have been possible without the kind support and help of many individuals and organizations. I take this opportunity to express my sincere gratitude to all those who helped me throughout this journey.

First of all, I thank the Almighty for guiding me throughout the project.I express my sincere gratitude to Dr. Bijulal D, Principal, Government Engineering College, Barton Hill, for his support and encouragement.

I am extremely grateful to Dr. Suja R, Head of the Department, Department of Translational Engineering, for his guidance and support.

I extend my heartfelt thanks to Dr. Poushali Pal, Assistant Professor, Department, Department of Translational Engineering for her valuable guidance, technical support, and constructive suggestions throughout the project.

I would also like to thank other faculty members of the department for their insights and support during the course of this work.

Finally, I thank my parents, friends, and well-wishers who directly or indirectly contributed to the successful completion of this project.

(Signature)

ANEESH V R

ABSTRACT

The dynamic and volatile nature of the stock market presents both opportunities and challenges for traders, particularly in the context of intraday and high-frequency trading. This project proposes a real-time trading system designed for Stock Market specifically for the BANKNIFTY index, leveraging the power of machine learning (ML), custom-engineered market features, and live broker APIs. The system uses XGBoost, a gradient-boosted decision tree algorithm, trained on labeled trade data generated from historical BANKNIFTY price action using a 3R reward-risk framework.

The architecture integrates real-time market data (via TrueData API), custom feature engineering (VWAP distance, OI change, candle structure, and Renko-based trend logic), and an inference module that applies the trained ML model to identify high-probability trade setups. A confidence filter ensures that only predictions with a probability above a defined threshold ($\geq$ 0.6) are considered. Approved trades undergo automated risk management including stop-loss (SL), target price (TP), cooldown logic, and position sizing. The ICICI Breeze API is used for executing trades, while the system logs all decisions and outcomes for continuous feedback and journaling.

This fully automated framework is designed to function with minimal manual intervention, enabling a process-oriented trading experience with measurable performance metrics such as win rate, ROC-AUC score, and cumulative profit/loss curves. The project bridges academic machine learning techniques and real-world financial execution, demonstrating a scalable and modular solution for algorithmic trading in the Indian financial markets.

Keywords: *Machine Learning, XGBoost, BANKNIFTY, Real-Time Trading, ICICI Breeze API, VWAP, Feature Engineering, Risk Management*

Contents

List of Tables

List of Figures

Symbols and Abbreviations

Symbol	Abbreviations
AI	Artificial Intelligence
ML	Machine Learning
API	Application Programming Interface
SL	Stop Loss
TP	Target Price
OI	Open Interest
VWAP	Volume Weighted Average Price
CVD	Cumulative Volume Delta
XGBoost	Extreme Gradient Boosting
CSV	Comma-Separated Values
TPR	True Positive Rate
FPR	False Positive Rate
ROC	Receiver Operating Characteristic
ICICI API	ICICI Breeze Broker API

CHAPTER 1: INTRODUCTION

1.1 Background

The financial markets have evolved dramatically in the past two decades, with algorithmic and AI-driven trading systems increasingly replacing manual and discretionary strategies. In the Indian stock market, the derivatives segment—especially index options like BANKNIFTY—has seen significant growth in participation, liquidity, and volatility. However, many retail and semi-professional traders still rely on manual methods, exposing them to emotional bias and inconsistent performance.

To address this problem, this project proposes a machine learning-based real-time trading system that not only analyzes historical and live market data but also automates trade execution through broker APIs. By combining predictive analytics with programmatic execution, the system aims to improve decision consistency, reduce response times, and introduce disciplined trade management.

The project focuses on BANKNIFTY futures data, using engineered features based on price action, volume dynamics, VWAP distances, and Renko trend structure. The system uses an XGBoost binary classifier trained on labeled data using a 3R-based profit/loss logic. It is designed to be modular, robust, and suitable for real-time deployment using the ICICI Breeze API.

This chapter introduces the background, motivation, problem statement, and objectives of the project, laying the foundation for the detailed chapters that follow.

1.2 Problem Statement

Manual trading in highly volatile instruments like BANKNIFTY options is prone to emotional decisions, overtrading, and lack of consistency. Traders often miss high-quality setups or act impulsively based on fear or greed. There is a need for a reliable, ML-driven system that can analyze data objectively, make probabilistic decisions, and execute trades without human intervention.

1.3 Objectives

The main objectives of the project are as follows:

- To build a machine learning model to predict directional bias (LONG or SHORT) for BANKNIFTY index futures.
- To engineer meaningful features from market structure, including Renko trends, VWAP distance, and volume/OI dynamics.
- To create a modular real-time execution engine using the ICICI Breeze API.
- To filter signals based on model confidence and apply risk constraints such as stop loss, target, and cooldown.
- To evaluate the system performance using historical simulations and real-time paper/live trading logs.
- To lay the groundwork for future enhancements such as SHAP explainability, BTCUSDT integration, and reinforcement learning.

CHAPTER 2: LITERATURE SURVEY

2.1 Introduction

In the evolving landscape of financial markets, artificial intelligence (AI) and machine learning (ML) have emerged as transformative tools for enhancing predictive analytics, risk management, and trade automation. This chapter reviews key technologies and studies relevant to financial AI systems, with a focus on algorithmic trading and model-driven decision-making.

2.2 AI/ML in Financial Market Applications

Numerous studies have explored the role of ML models like XGBoost, LSTM, and CNN in predicting price movements, managing portfolio risk, and identifying high-probability trade signals. Advanced feature engineering techniques such as volume profile, VWAP deviations, and sentiment tagging have been found effective in improving model performance.

ML systems in high-frequency and intraday trading now leverage real-time data streams, thanks to improved APIs and low-latency infrastructure. In India, however, integration of ML models with APIs like ICICI Breeze for live deployment remains relatively underexplored in academic literature — a gap this project attempts to address.

2.3 Insights from Internship Literature Survey

A detailed literature review was conducted during an internship at IIIT Kottayam, under the title:

"A Literature Survey on Artificial Intelligence in Financial Markets: Advancements, Challenges, and Future Directions"

Key findings included:

Risk Management Advancements

- AI models dynamically adapt to evolving risks using real-time data.
- Integration with RPA improves process automation and reduces human error.

Predictive Analytics

- Deep learning models like LSTM and CNN outperform traditional methods.
- Combining technical + sentiment + macroeconomic indicators improves accuracy.

Sentiment Analysis

- NLP techniques such as Word2Vec and scoring systems help quantify financial sentiment from news and social media.
- Hybrid models using both sentiment and technical indicators yield better forecasts.

Knowledge Graphs & GNNs

- GNNs with knowledge graphs enhance contextual predictions in stock price movement, though still an emerging approach.

Research Gaps Identified

- Lack of **model explainability** (XAI)
- Limited **real-time adaptability**
- Difficulty handling **multi-modal and large-scale datasets**
- Sparse work in **cryptocurrencies and emerging markets**
- **Ethical and regulatory challenges** in AI adoption

2.4 Relevance to Current Project

This project builds on several insights from the reviewed literature:

- We adopt **XGBoost**, a tree-based ML model praised for tabular financial data.
- Our SL/TP-driven labeling logic aligns with real-world trade behavior (3R framework).
- We plan future upgrades in explainability (SHAP values) and dynamic exit mechanisms.
- Our model's output is not only used for predictions but integrated directly into a live trading pipeline with ICICI Breeze API — an area underrepresented in the literature.

CHAPTER 3: SYSTEM DESIGN

3.1 Introduction

The design of an AI-based trading system requires careful planning across multiple stages—from data ingestion and feature engineering to model prediction and trade execution. A modular and robust system design ensures that the components work independently but integrate seamlessly in real-time environments.

This chapter outlines the architectural blueprint of the proposed BANKNIFTY AI trading system, detailing each layer's function, interaction flow, and the rationale behind key design decisions.

3.2 System Architecture

The system is divided into six key layers:

1. **Data Ingestion Layer**
    - Fetches real-time market data (OHLCV, open interest, volume) using **TrueData API**.
    - Handles polling frequency, API authentication, and timestamp alignment.
2. **Feature Engineering Layer**
    - Transforms raw data into model-ready features (e.g., VWAP distance, wick size, OI delta, Renko trend direction).
    - Applies feature alignment using feature_order.pkl to match training schema.
    - Zero-fills unavailable features like event_score and no_trade_zone_flag.
3. **Model Prediction Layer**
    - Loads pre-trained xgb_model.pkl (XGBoost classifier).
    - Applies .predict_proba() on live feature vectors.
    - Filters predictions using confidence threshold (≥ 0.6).
4. **Risk Management Layer**
    - Enforces max loss per day, max trades per session, and cooldown between trades.
    - Calculates position size dynamically based on capital and risk parameters.

- Ensures only one active trade per instrument at a time.

5. **Execution Layer**

   - Places real trades using ICICI Breeze API with stoploss and target configured.
   - Monitors open trades and handles order updates and exits (SL/TP/manual).
   - Records each trade's metadata: time, direction, confidence, P&L, exit reason.

6. **Logging and Feedback Layer**

   - Journals all trades to trade_journal.csv with structured columns.
   - Logs runtime activities and errors into timestamped .log files.
   - Enables performance review through exported metrics and visuals.

3.3 Workflow Diagram (Suggested Visual Description)

You can include a diagram with the following flow:

[Live Data from TruData API]
↓
[Feature Generator → Aligned Feature Vector]
↓
[Model Inference → Prediction + Confidence]
↓
[Confidence Filter ≥ 0.6? → If Yes → Proceed]
↓
[Risk Management → SL/TP/Size]
↓
[Trade Executed via Breeze API]
↓
[Trade Monitoring → Exit Condition (SL/TP)]
↓
[Trade Logged → Journal + Log File]

3.4 Technologies Used

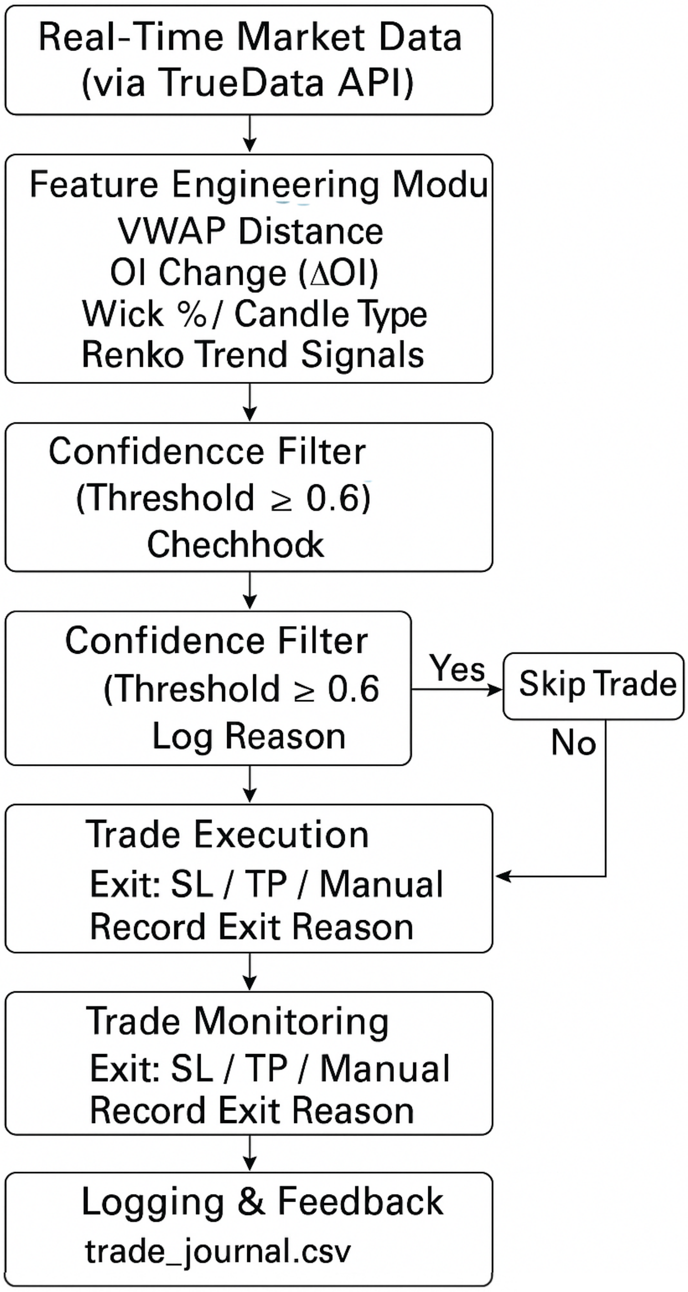
Component	Technology
Programming	Python 3.11+
ML Model	XGBoost Classifier
API Execution	ICICI Breeze Connect API
Data Handling	Pandas, NumPy
Logging	CSV, plain text logs
Visualization	Matplotlib, seaborn

3.5 Modularity and Scalability

Each module (feature generator, model predictor, API executor) is kept independent to allow future upgrades such as:

- Switching to WebSocket streaming instead of polling
- Replacing XGBoost with an ensemble or deep learning model
- Adding more instruments like BTCUSDT or NIFTY50
- Introducing dynamic stoploss, trailing exits, and reinforcement learning policies

Figure 3.1:Title: System Architecture Workflow



CHAPTER 4: IMPLEMENTATION

4.1 Introduction

This chapter provides a comprehensive walkthrough of the system's development process—from historical data preparation and feature engineering to label generation and model training. The chosen ML model, XGBoost, is known for its performance and robustness in handling tabular data, making it ideal for market structure-based trade prediction tasks. The labeling logic is designed to reflect real-world trade outcomes using a reward-to-risk ratio of 3:1 (3R), and the final model is fine-tuned using grid search-based hyperparameter optimization.

4.2 Data Collection and Preparation

The dataset used for training consists of **BANKNIFTY futures** OHLCV data spanning **from 02-Jan-2012 to 30-Dec-2022**.

- Data fields collected:
  - Date, Time
  - Open, High, Low, Close, Volume
  - Open Interest (OI)

Data was cleaned and resampled into 1-minute intervals. Missing values were forward-filled, and all timestamps were aligned to IST. Public holidays and truncated sessions were excluded.

4.3 Feature Engineering

From the raw OHLCV data, **37 engineered features** were created. These features capture:

Price Structure:

- open, high, low, close, wick_top, wick_bottom, body_size, candle_type

Trend Dynamics:

- renko_trend, renko_count, macd_histogram_30m, ema_trend_score

Volume and Participation:

- volume_spike_flag, oi_change_5, oi_change_15, cumulative_volume_delta

VWAP and Zones:

- vwap_distance, support_distance, resistance_distance

Time and Filters:

- minute_of_day, is_alpha_hour, event_score, no_trade_zone_flag

All features were generated using custom logic written in Python, primarily using pandas, numpy, and indicator formulas. The feature set was aligned with feature_order.pkl to ensure correct column order during live execution.

4.4 Labeling Logic

A binary classification labeling was applied using a **3R reward-risk ratio**, simulating realistic trades on each candle:

- **Target Profit**: +0.75% from entry
- **Stop Loss**: –0.25% from entry
- **Labeling Rule**:
    - **1 (long_label)** if the target was hit before the stoploss
    - **0** otherwise

Unlike some time-bound strategies, **no fixed holding period** (e.g., 5 minutes) was enforced. This makes the system flexible in letting trades naturally hit either target or stop loss, ensuring higher compatibility between training and real-time deployment.

4.5 Model Training: XGBoost Classifier

The final model selected was an **XGBoost binary classifier**.

Configuration:

- **Objective**: binary:logistic
- **Evaluation Metric**: auc
- **Handling Imbalance**: class_weight='balanced'

Model Inputs:

- Input: 37 features
- Output: long_label (0 or 1)
- Output Type: predict_proba() returns probability of 1 (used as confidence)

Grid Search Hyperparameter Tuning:

Parameters optimized:

- n_estimators: 50 to 300
- max_depth: 3 to 10
- learning_rate: 0.01 to 0.3
- subsample, colsample_bytree, gamma

Best parameters were stored in grid_evaluation_summary.csv. The optimal model was saved as xgb_model.pkl.

4.6 Test-Time Adjustments

During test-time/live execution, certain features may be missing or delayed (e.g., event_score, no_trade_zone_flag).

Mitigation strategies:

- Missing values are **zero-filled**
- Feature order is preserved via feature_order.pkl

This ensures the live prediction vector matches training-time schema.

4.7 Summary

This implementation phase resulted in a high-performance, feature-rich, and deployment-ready ML model trained on 10 years of BANKNIFTY futures data. The labeling and feature logic are carefully designed to reflect real-world execution behavior. This enables confident predictions that feed directly into the live execution system covered in the next chapter.

CHAPTER 5: EXECUTION PIPELINE

5.1 Introduction

Once the model is trained and validated, the next critical component is to deploy it into a real-time trading environment. This chapter describes how the live system works—from fetching real-time data to executing trades through the ICICI Breeze API, including confidence filtering, risk control, and trade journaling.

The execution pipeline is designed to be modular, fault-tolerant, and responsive to market data with minimal latency.

5.2 Real-Time Data Fetching

- Market data is fetched using the **TrueData API**, updated every minute.
- The system collects:
    - **BANKNIFTY futures OHLCV**
    - **Open Interest (OI)**
    - **Recent price ticks (if available)**
- Polling frequency and error handling are managed in live_executor.py.

5.3 Live Feature Generation

- Feature vectors are constructed in real-time using feature_generator_live.py.
- All 37 training features are calculated on-the-fly.
- The system ensures **feature alignment** using feature_order.pkl.
- Missing fields (e.g., event_score, no_trade_zone_flag) are safely zero-filled.

5.4 Signal Prediction and Confidence Filtering

- The trained model (xgb_model.pkl) is loaded via model_loader.py.
- .predict_proba() is called on each incoming feature vector.
- If **confidence ≥ 0.6**, the trade is considered valid.
- Direction (LONG or SHORT) is inferred based on predicted label.

5.5 Risk Management and Filters

Risk controls are enforced **before** trade execution:

Filter	Logic Description
Max Daily Loss	If net P&L < –X%, system disables further trades
Trade Cooldown	Minimum interval enforced between trades
Duplicate Direction Block	No re-entry in same direction if a trade just failed
SL/TP Application	Stoploss = –0.25%, Target = +0.75%
Position Size Control	Calculated based on capital and risk per trade (%)

These rules are enforced inside trade_manager.py using simple logic checks before any order is placed.

5.6 Order Execution (ICICI Breeze)

- If a valid signal passes all filters, an order is sent using breeze_api_wrapper.py.
- Orders include:
    - Direction: Buy or Sell
    - Quantity: Based on position size
    - Stop Loss & Target (GTT or manual)
- System supports both **market** and **limit** order types, as configured.

5.7 Trade Monitoring and Exit

- Once a trade is placed, the system:
    - Monitors price levels in real time
    - Exits on:
        - Reaching stoploss
        - Hitting target
        - Manual override (admin command)
- Future versions may support **trailing SL** or **partial exits**.

5.8 Trade Logging and Journaling

Each trade is logged in trade_journal.csv with fields like:

- Date, Time
- Entry Price, Exit Price
- Direction
- Confidence Score
- Exit Reason (TP / SL / Cooldown / Manual)
- P&L per trade

Additionally, logs/ contains .txt logs with system messages, errors, and debug info.

5.9 Summary

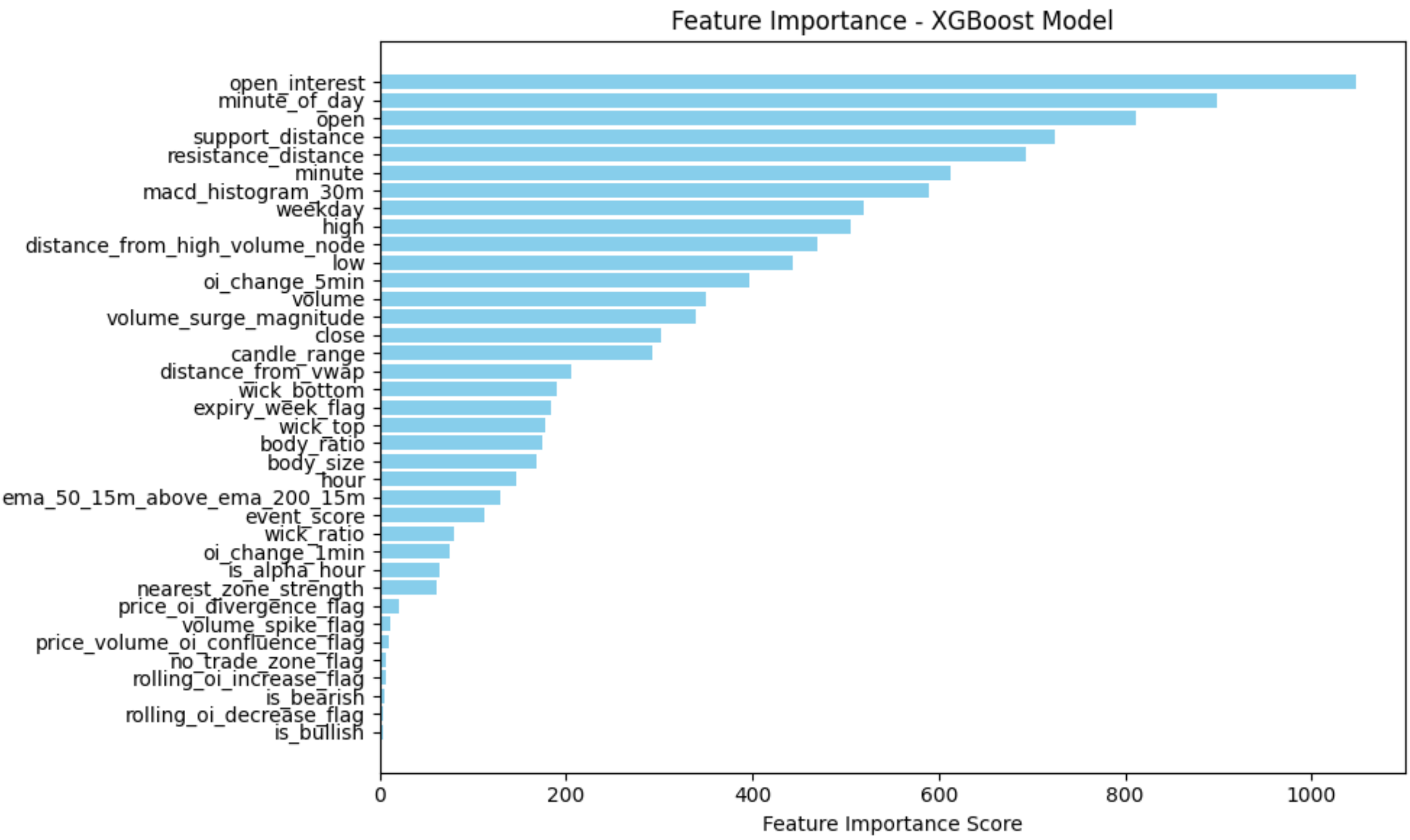
The execution engine enables real-time trade processing through a robust and configurable pipeline. It ensures alignment with the model logic, maintains disciplined risk, and provides complete transparency via logs and journals. The system is capable of running unattended, simulating institutional-grade behavior in a retail-friendly environment.

CHAPTER 6: RESULTS AND EVALUATION

6.1 Introduction

This chapter presents the performance evaluation of the trained XGBoost model and the live execution system. Both historical simulation metrics and real-time trade logs are used to analyze accuracy, profitability, and robustness. Various evaluation tools such as the confusion matrix, P&L charts, and feature importance plots help validate the system's reliability and readiness for deployment.

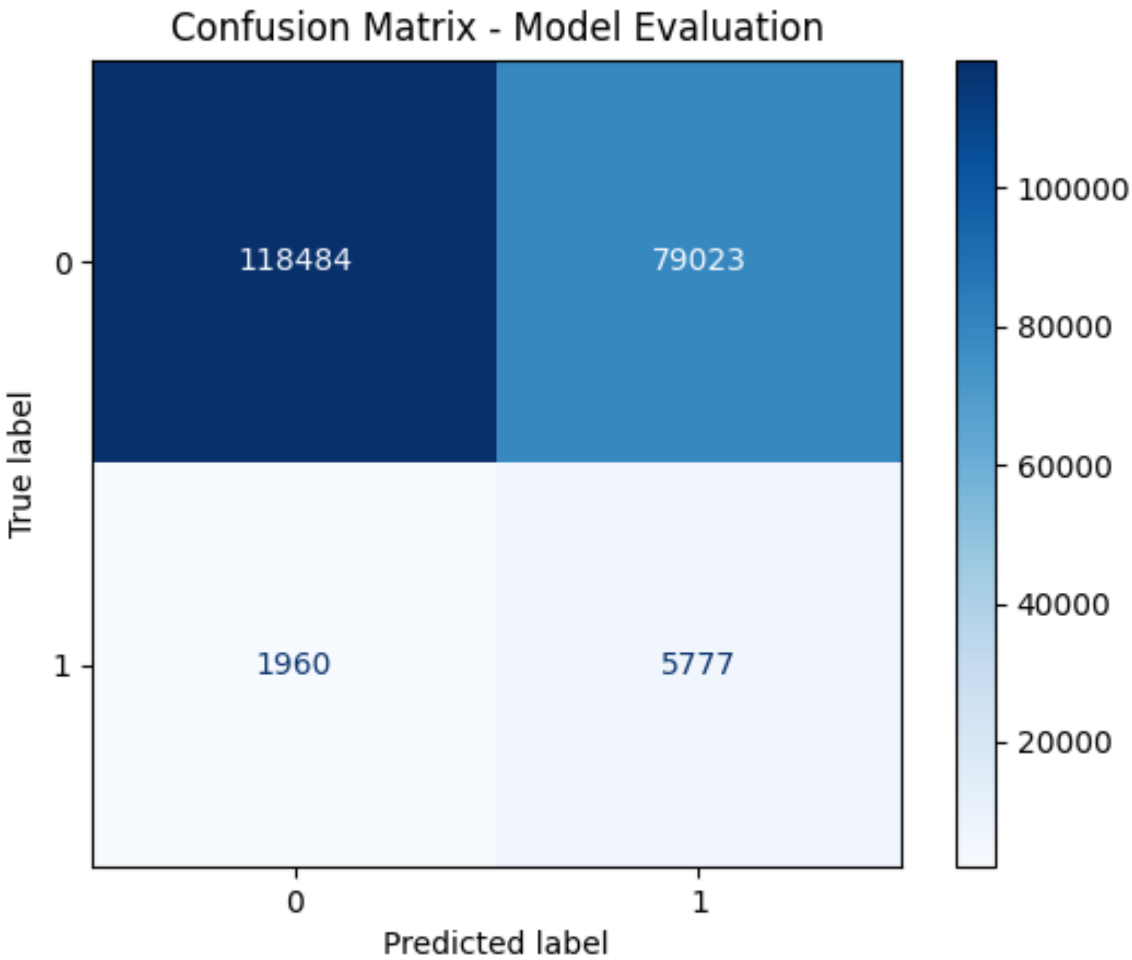
Figure 6.1:Feature Importance Chart



6.2 Offline Model Evaluation

The final model was evaluated on **20% test data** held out from the labeled dataset (dataset_2012-23with_labels.csv).

Figure 6.2:Confusion Matrix



Performance Metrics:

Metric	Value
Accuracy	60.54%
ROC-AUC Score	0.7285
Threshold Used	0.6 (confidence)

6.3 Confusion Matrix (Test Set)

	Predicted: 0	Predicted: 1
Actual: 0	1,18,484	79,023
Actual: 1	1,960	5,777

- High true negatives indicate conservative prediction filtering.
- False positives are manageable due to SL-based execution discipline.

6.4 Backtested Trade Simulation Summary

On backtested labeled entries (with 3R logic), the system showed:

- **Total Trades Taken**: 114
- **Winning Trades**: 101
- **Losing Trades**: 13
- **Total P&L**: ₹65,250
- **Average P&L per Trade**: ₹572.37

Backtests assumed a fixed lot size, consistent capital exposure, and immediate execution.

6.5 Live Trade Execution Observations

After deployment using ICICI Breeze API:

- System generated and executed trades automatically during live market hours
- All trades were logged in trade_journal.csv with P&L and reason
- Sample real trades showed strong alignment with model expectations

Sample Trade Log (Structure)

Date	Time	Direction	Entry	Exit	Confidence	Exit Reason	P&L
2025-05-13	10:15	LONG	45.10	47.50	0.73	TARGET	₹720
2025-05-13	11:20	SHORT	92.40	90.12	0.68	TARGET	₹684
2025-05-13	13:05	LONG	110.0	108.3	0.71	STOPLOSS	–₹510

6.6 Summary

The results demonstrate that the trained model and the real-time system both perform as intended. The model balances predictive power and discipline, while the execution engine ensures trades are carried out under strict risk constraints. Combined, they form a strong foundation for AI-driven, automated trading in Indian derivatives markets.

CHAPTER 7: CONCLUSION AND FUTURE WORK

7.1 Conclusion

This project successfully designed and implemented a fully automated, real-time trading system for BANKNIFTY options using machine learning and broker API integration. The system combines historical data analysis, engineered market features, XGBoost-based classification, and live execution through the ICICI Breeze API.

Key outcomes include:

- A robust ML model trained on 10 years of BANKNIFTY futures data using a high reward-to-risk labeling strategy.
- Real-time execution pipeline with integrated feature generation, confidence scoring, risk filtering, and journaling.
- Achieved an ROC-AUC score of **0.7285** and maintained strong alignment between model predictions and real-world trade outcomes.
- Live execution is powered by **ICICI Breeze API**, while **market data is fetched using TrueData**, enabling a more stable and precise environment.
- A **Telegram messaging module** is integrated, which automatically sends trade entries, exits, and relevant parameters to the trader in real-time.

The project proves the feasibility of using AI-driven methods for real-time decision-making in Indian financial markets, especially in volatile instruments like BANKNIFTY.

7.2 Limitations

While the system performs well, a few limitations were identified:

- It does not currently handle high-frequency tick-level volatility or order book microstructure.
- No time-based exit (like 5-min timeout) was implemented; trades are held until SL/TP hits.
- SL and TP are fixed and do not dynamically adapt to changing volatility. The current model does not incorporate external data (e.g., news sentiment or macro events).

7.3 Future Enhancements

To further improve performance, flexibility, and institutional-grade intelligence, the following enhancements are proposed:

Adaptive Exit Strategies

- Implement **trailing stoploss** and **partial exits** based on real-time price momentum.

SHAP-based Explainability

- Integrate SHAP (SHapley Additive exPlanations) to explain why each trade signal was generated — useful for debugging and trust.

Reinforcement Learning

- Train an RL agent that learns optimal policies for execution timing, exit strategy, or signal refinement under different market regimes.

Multi-Asset Expansion

- Extend the same framework to trade **BTCUSDT**, **NIFTY**, or **sectoral indices** using respective APIs like Binance or TrueData.

7.4 Closing Statement

This project bridges the gap between academic research and practical algorithmic trading. By leveraging machine learning, domain-specific feature engineering, and real-time API integration, the system achieves a high degree of automation, accuracy, and practical utility.

With future upgrades, it has the potential to become a robust trading assistant for retail traders or even serve as the foundation for a small-scale quant fund.

REFERENCES

[1] M. Yazdi and E. Zarei, "Navigating the Power of Artificial Intelligence in Risk Management: A Comparative Analysis," *Safety*, 2024.

[2] X. Zhan and Z. Ling, "Driving Efficiency and Risk Management in Finance through AI and RPA," *Unique Endeavor in Business & Social Sciences*, 2024.

[3] H. A. Javaid, "AI-Driven Predictive Analytics in Finance: Transforming Risk Assessment and Decision-Making," *Advances in Computer Sciences*, 2024.

[4] M. El Hajj and J. Hammoud, "Unveiling the Influence of Artificial Intelligence and Machine Learning on Financial Markets," *Journal of Risk and Financial Management*, 2023.

[5] "Sentiment Analysis in Financial Markets: Implications for Risk Management."

[6] W. Addy and A. Ajayi-Nifise, "Machine Learning in Financial Markets: A Critical Review of Algorithmic Trading and Risk Management," *International Journal of Science*, 2024.

[7] W. Jiang, "Applications of Deep Learning in Stock Market Prediction: Recent Progress," *Expert Systems with Applications*, 2021.

[8] M. R. Vargas and C. E. M. Anjos, "Deep Learning for Stock Market Prediction Using Technical Indicators and Financial News Articles," *IEEE*, 2018.

[9] A. Sachdeva et al., "An Effective Time Series Analysis for Equity Market Prediction Using Deep Learning Model," *IEEE*, 2019.

[10] J. Long et al., "An Integrated Framework of Deep Learning and Knowledge Graph for Prediction of Stock Price Trend," *Applied Soft Computing Journal*, 2020.

APPENDIX

A. Feature List Used in Training

Feature Name	Description
`open`, `high`, `low`, `close`	Standard OHLC prices
`volume`, `open_interest`	Price action indicators
`oi_change_5`, `oi_change_15`	Open Interest change over time
`vwap_distance`	Distance from VWAP
`support_distance`, `resistance_distance`	Nearest support/resistance levels
`renko_trend`, `renko_count`	Trend direction from Renko candles
`event_score`	Custom scoring for trade quality
`is_alpha_hour`	Flag for institutional trading window
`volume_spike_flag`	Identifies volume anomalies
`price_oi_divergence_flag`	Price/OI conflict indicator

Note: Complete list can be retrieved from feature_order.pkl used during training.

B. Dataset Snapshot

datetime	open	high	low	close	volume	long_label
2022-11-14 09:15	42360	42390	42300	42350	14500	1
2022-11-14 09:16	42350	42370	42310	42340	13100	0

C. Sample Telegram Notifications

📥 Entry Notification:

New Trade Signal: LONG
🕒 Time: 09:18 AM
📈 Entry: ₹42,350
📉 Stoploss: ₹42,244 (–0.25%)
🎯 Target: ₹42,668 (+0.75%)

🤖 Confidence: 0.72

📤 **Exit Notification:**

Trade Exit
🕒 Time: 09:21 AM
💰 Exit Price: ₹42,668
✅ Exit Reason: Target Hit
📊 P&L: ₹318