

# DOCUMENTATION

# PYTHON DEVELOPMENT

## WEEK-3

**PROJECT NAME:** Real-Time FinTech Fraud Detection System  
(SentinelStream)

**Week 3 –** Intelligence Layer (Machine Learning & Rules)

**Organization** -Zaalima Development Pvt. Ltd.

**CREATED BY** - PITCHE ESHWAR

## **1. Problem Statement:**

As transaction volume increases, static validation alone is insufficient to accurately detect fraud. Modern fraud patterns are dynamic and evolve over time, requiring intelligent decision-making mechanisms that can identify both known and unknown anomalies in real time.

To address this, an intelligence layer combining rule-based logic and machine learning-based anomaly detection is required while still maintaining strict latency constraints (<200 ms).

## **2. Objective:**

The objective of Week 3 is to implement the Intelligence Layer of SentinelStream by:

Integrating machine learning for anomaly detection

Implementing a dynamic rule engine

Combining rule results and ML scores for final fraud decisions

Enabling real-time alerts for high-risk transactions

Ensuring system latency consistently stays below 200 ms

## **3. Intelligence Layer Overview:**

The Intelligence Layer is responsible for evaluating transaction risk after basic validation and caching.

It acts as the core fraud detection logic of the system.

- Key Components:
- Rule Engine
- Machine Learning Scoring Engine
- Decision Engine
- Alerting System

## **4. Machine Learning Integration:**

### 4.1 Model Selection

- Integrated a pre-trained Isolation Forest model
- Used for anomaly detection in transaction patterns

### 4.2 Functionality

- The model analyzes transaction features such as:
  - Transaction amount
  - Frequency
  - Location deviation
- Produces an anomaly score indicating fraud likelihood

### 4.3 Implementation Details

- Model loaded using serialization (joblib / pickle)
- Inference performed in real time
- Optimized for low-latency scoring

## **5. Rule Engine Implementation:**

### 5.1 Rule Engine Design

- Implemented a dynamic Rule Engine class
- Allows fraud rules to be defined without code changes
- Supports configurable conditions such as:
  - High transaction amount
  - Unusual location
  - Abnormal frequency

### 5.2 Rule Execution

- Rules are evaluated before ML scoring
- Immediate flagging of obvious fraud cases
- Reduces unnecessary ML computation

## **6. Decision Engine:**

The Decision Engine combines:

Rule Engine output

Machine Learning risk score

Decision Outcomes:

- APPROVED – Low risk
- REVIEW – Medium risk
- REJECTED – High risk

This layered approach improves accuracy and reduces false positives.

## **7. Alerting Mechanism (Celery Workers):**

### **7.1 Asynchronous Processing**

- Implemented Celery workers for background tasks
- Prevents blocking the main transaction pipeline

### **7.2 Email Alerts**

- Email alerts are triggered for high-risk transactions
- Alerts sent asynchronously to ensure system performance

## **8. Latency Validation:**

Performance Requirement:

- End-to-end processing must remain below 200 ms

Validation:

- Latency measured across:
  - Rule evaluation
  - ML inference
  - Decision making
- System consistently met latency constraints

## **9. Technologies Used (Week 3):**

- FastAPI – Backend framework
- Scikit-learn – Isolation Forest ML model
- Celery – Background task processing
- Redis / RabbitMQ – Message broker
- Python – Core implementation

## **10. Deliverables:**

- Integrated Isolation Forest anomaly detection model
- Fully functional Rule Engine
- Combined decision logic
- Celery-based email alert system
- Latency-validated intelligence pipeline

## **11. Challenges & Mitigation:**

### Challenges

- Maintaining low latency with ML inference
- Balancing rule-based and ML-based decisions
- Avoiding blocking operations during alerts

### Mitigation

- Lightweight ML model
- Rules executed before ML
- Asynchronous alert handling

## **12. Learning Outcomes:**

By completing Week 3:

Gained practical experience in ML model integration

Understood real-time fraud detection strategies

Learned how to design rule-based systems

Implemented scalable background processing

Ensured performance under strict latency limits

## **13. Conclusion:**

Week 3 successfully implemented the Intelligence Layer of SentinelStream. By combining rule-based validation with machine learning-driven anomaly detection, the system now performs accurate, real-time fraud assessment while maintaining production-level performance and reliability.

This prepares the system for final deployment, security hardening, and testing in the next phase.