

Configuration Manual

MSc Research Project
Master's In Data Analytics

Jennifer Ullasa Kumar
Student ID: x23286091

School of Computing
National College of Ireland

Supervisor: Eric Gyamfi

National College of Ireland
MSc Project Submission Sheet
School of Computing



Student Name: Jennifer Ullasa Kumar
Student ID: x23286091
Programme: Master In Data Analytics **Year:** 2025
Module: Research Practicum
Lecturer: Eric Gyamfi
Submission Due Date: 15/09/2025
Project Title: Hybrid Real Fraud Detection
Word Count: 764 **Page Count: 7**

I hereby certify that the information contained in this (my submission) is information pertaining to research I conducted for this project. All information other than my own contribution will be fully referenced and listed in the relevant bibliography section at the rear of the project.

ALL internet material must be referenced in the bibliography section. Students are required to use the Referencing Standard specified in the report template. To use other author's written or electronic work is illegal (plagiarism) and may result in disciplinary action.

Signature: Jennifer Ullasa Kumar

Date: 11/08/2025

PLEASE READ THE FOLLOWING INSTRUCTIONS AND CHECKLIST

Attach a completed copy of this sheet to each project (including multiple copies)	☐
Attach a Moodle submission receipt of the online project submission, to each project (including multiple copies).	☐
You must ensure that you retain a HARD COPY of the project, both for your own reference and in case a project is lost or mislaid. It is not sufficient to keep a copy on computer.	☐

Assignments that are submitted to the Programme Coordinator Office must be placed into the assignment box located outside the office.

Office Use Only	
Signature:	
Date:	
Penalty Applied (if applicable):	

Configuration Manual

Jennifer Ullasa Kumar
x23286091

1 System Configuration

- OS: macOS (works on Linux/Windows too)
- RAM: 8–16 GB
- Python: 3.10+ in a virtual env
- Docker: Desktop installed (for Kafka/Zookeeper via docker compose)
- Kafka: brought up by docker-compose.yml
- Required Python packages

`pip install kafka-python streamlit pandas numpy scikit-learn joblib river altair`

- Project files (root folder):
 - `docker-compose.yml` – Kafka + Zookeeper
 - `start_all.sh` – helper to launch everything (Mac: opens new Terminal tabs)
 - `kafka_producer.py` – streams creditcard.csv into Kafka
 - `kafka_consumer.py` – scores messages, writes logs, sends alerts
 - `app.py` – Streamlit dashboard
 - Models: `rf_model.joblib`, `scaler.joblib`
 - Data: `creditcard.csv`
 - Outputs (created at runtime):

`stream_results.csv`, `kafka_alerts.csv`, `metrics_window.csv`, `kafka_drift_log.csv`

2 Execution Steps

One Shot Launcher

```
chmod +x start_all.sh
./start_all.sh
```

Manual:

Terminal 1:

```
docker compose up -d && docker compose ps
```

Terminal 2: Consumer

```
source .venv/bin/activate
python3 kafka_consumer.py
```

Terminal 3: Producer

```
source .venv/bin/activate
python3 kafka_producer.py
```

Terminal 4: Dashboard
source .venv/bin/activate
streamlit run app.py

3 Section 3

3.1 kafka_producer.py

```
kafka_producer.py x kafka_consumer.py app.py $ start_all.sh
kafka_producer.py > ...
1  import pandas as pd
2  from kafka import KafkaProducer
3  import json, time
4
5  # ----- CONFIG -----
6  CSV      = "creditcard.csv"      # path to your dataset
7  TOPIC    = "transaction_topic"
8  BOOTSTRAP = "localhost:9092"
9  SLEEP_SEC = 0.01                # 10ms between messages to simulate streaming
10
11 # Columns your consumer expects
12 FEATURES = ['Time'] + [f'V{i}' for i in range(1, 29)] + ['Amount']
13 LABEL    = 'Class'
14
15 # ----- LOADING DATA -----
16 df = pd.read_csv(CSV)
17
18 # Validate columns
19 missing = [c for c in FEATURES + [LABEL] if c not in df.columns]
20 if missing:
21     raise ValueError(f"Missing columns in {CSV}: {missing}")
22
23 # Ensure numeric types (prevents JSON serialization issues)
24 for c in FEATURES + [LABEL]:
25     df[c] = pd.to_numeric(df[c], errors='coerce')
26
27 # ----- KAFKA PRODUCER -----
28 producer = KafkaProducer(
29     bootstrap_servers=BOOTSTRAP,
30     value_serializer=lambda v: json.dumps(v).encode('utf-8'),
31     linger_ms=5
32 )
33
34 print("\n Producer connected. Streaming transactions...\n")
35
36 # Stream rows in original (chronological) order
37 for idx, row in df.iterrows():
38     # Build message
39     msg = {
40         'tx_id': int(idx),
41         'send_ts': time.time(), # used by consumer for end-to-end latency
42     }
43
44     # Features + label
```

Purpose: read creditcard.csv and publish each transaction to Kafka.

Core flow:

- Config: TOPIC="transaction_topic", BOOTSTRAP="localhost:9092".
- Validates columns ['Time', V1...V28, 'Amount', 'Class'].
- Serializes each row as JSON: includes tx_id, timestamps, features, and label.
- Sends to Kafka with a tiny sleep (0.01s) to simulate real-time.
- Why it matters: gives a controlled, replayable stream to test the online pipeline.

3.2 kafka_consumer.py

```
kafka_producer.py  kafka_consumer.py X  app.py  $ start_all.sh
kafka_consumer.py > ...
1  from kafka import KafkaConsumer, KafkaProducer
2  import json, joblib, os, csv
3  import pandas as pd
4  from collections import deque
5  from datetime import datetime, timezone
6
7  from sklearn.metrics import precision_recall_fscore_support, average_precision_score
8  from river.drift import ADWIN
9
10 # ----- Config -----
11 TOPIC_IN  = "transaction_topic"
12 TOPIC_OUT = "alerts"
13 BOOTSTRAP = "localhost:9092"
14
15 FEATURES = ['Time'] + [f'V{i}' for i in range(1, 29)] + ['Amount']
16 LABEL    = 'Class'
17
18 THRESHOLD = 0.20      # decision threshold for labeling as fraud
19 WINDOW_SIZE = 1000    # rolling window for metrics (recommended by dataset note)
20 METRIC_EVERY = 200    # compute and log metrics every N events
21 # -----
22
23 # Load model + scaler
24 model = joblib.load('rf_model.joblib') # must support predict_proba
25 scaler = joblib.load('scaler.joblib')
26
27 # Kafka
28 consumer = KafkaConsumer(
29     TOPIC_IN,
30     bootstrap_servers=BOOTSTRAP,
31     auto_offset_reset='earliest',
32     value_deserializer=lambda m: json.loads(m.decode('utf-8'))
33 )
34 alert_producer = KafkaProducer(
35     bootstrap_servers=BOOTSTRAP,
36     value_serializer=lambda v: json.dumps(v).encode('utf-8')
37 )
38
39 # Files
40 ALERT_CSV = 'kafka_alerts.csv'
41 DRIFT_CSV = 'kafka_drift_log.csv'
42 RESULTS_CSV = 'stream_results.csv' # per-transaction log
43 METRICS_CSV = 'metrics_window.csv' # rolling metrics for dashboard
44
```

Purpose: consume messages, score fraud probability, emit alerts, log metrics.

Core flow:

- Config: input topic `transaction_topic`, output topic `alerts`, `THRESHOLD=0.20`.
- Loads `rf_model.joblib` and `scaler.joblib`, uses `predict_proba` on features.
- Writes every prediction to `stream_results.csv` with columns:

`tx_id`, `pred`, `label`, `p_fraud`, `ts`.

- If `p_fraud >= THRESHOLD`, sends an alert to Kafka and appends to `kafka_alerts.csv`.
- Maintains a rolling window (`WINDOW_SIZE=1000`) and every `METRIC_EVERY=200` events logs precision, recall, F1, and average precision to `metrics_window.csv`.
- Runs ADWIN drift detection on errors; on change, logs a timestamp to `kafka_drift_log.csv`.

- Why it matters: closes the real-time loop with decisions, observability, and drift flags.

3.3 C. Dashboard app.py

```

kafka_producer.py kafka_consumer.py app.py $ start_all.sh
app.py > ...
1 import os
2 import time
3 import pandas as pd
4 import numpy as np
5 import streamlit as st
6 import altair as alt
7
8 # Prefer average_precision_score (AUPRC) but fall back gracefully
9 try:
10     from sklearn.metrics import average_precision_score, precision_recall_curve, auc, precision_score, recall_score
11     HAVE_SK = True
12 except Exception:
13     HAVE_SK = False
14
15 # ----- Config -----
16 RESULTS_CSV = "stream_results.csv" # tx_id,pred,label,p_fraud,ts
17 ALERTS_CSV = "kafka_alerts.csv" # tx_id,score,amount,time
18 DRIFT_CSV = "kafka_drift_log.csv" # DriftAt
19
20 st.set_page_config(
21     page_title="Real-Time Fraud Detection - Live Ops",
22     layout="wide",
23     page_icon="🔍",
24 )
25
26 # ----- Helpers -----
27 @st.cache_data(show_spinner=False)
28 def safe_read_csv(path: str, nrows: int | None = None) -> pd.DataFrame:
29     """Read a CSV that is being appended to, skipping malformed lines and
30     forcing the exact 5 columns we expect for stream_results.csv.
31     """
32     if not os.path.exists(path) or os.path.getsize(path) == 0:
33         return pd.DataFrame(columns=["tx_id", "pred", "label", "p_fraud", "ts"])
34
35     try:
36         df = pd.read_csv(
37             path,
38             header=None,
39             names=["tx_id", "pred", "label", "p_fraud", "ts"],
40             on_bad_lines="skip",
41             engine="python", # more forgiving for trailing writes
42             nrows=nrows,
43         )
44     except Exception:

```

Purpose: live operations view in Streamlit.

Core flow:

- Page config: wide layout, title “Real-Time Fraud Detection – Live Ops”.
- Safely reads append-only CSVs with a cached loader that skips malformed lines.
- Panels:
 - Recent Alerts table (kafka_alerts.csv) – last few high-risk transactions.
 - Performance chart (metrics_window.csv) – precision, recall, F1 over time (Altair line chart).
 - Latest transactions table (stream_results.csv) – tail view with p_fraud.
- Why it matters: gives you immediate feedback on model behavior and alert volume without digging into logs.

References

1. adimov, E. and Birihanu, E. (2025) 'Real-time suspicious detection framework for financial data streams', *International Journal of Information Technology*.
2. Alang, K. S. and Kushwaha, A. S. (2025) 'Stream processing with Apache Kafka: Real-time data pipelines', *International Journal of Research in Modern Engineering & Emerging Technology*.
3. Bifet, A. and Gavaldà, R. (2007) 'Learning from time-changing data with adaptive windowing', in *Proceedings of the 2007 SIAM International Conference on Data Mining*.
4. Bian, S., Li, C., Fu, Y., Ren, Y., Wu, T., Li, G.-P. and Li, B. (2021) 'Machine learning-based real-time monitoring system for smart connected worker to improve energy efficiency', *Journal of Manufacturing Systems*.