Week 5 – IoT Integration and Data Pipeline Setup

Introduction:

Continuous sensor data is necessary for smart manufacturing systems to make defensible judgments. Week 5 creates a simulated real-time data pipeline that replicates real-world IoT scenarios in order to test, debug, and evaluate predictive algorithms in a realistic manner. This prepares for live deployment in Week 6 and for thorough integration testing and anomaly detection exploration. With the possibility of anomaly injection and potential Kafka compatibility, the simulation simulates data streaming from manufacturing sensors.

Objective:

- Create a real-time sensor data stream simulation using old JSON files.
- To evaluate the resilience of downstream failure detection methods, introduce controlled anomalies.
- Allow for two ingestion routes:
  - batch simulation based on local JSON.
  - placeholder for upcoming streaming of Kafka.
- To trace data lineage and troubleshoot flows, create and save stream manifest logs.

Methodology:

- **Setting up and starting up:**
  - Important runtime parameters are defined by the script:
    - SOURCE_FILE: The location of factory_sensor_data.json, which contains sensor records from the past.
    - BATCH_SIZE: The quantity of records that are transmitted throughout each cycle.
    - STREAM_INTERVAL: To replicate real-world flow, the time interval (in seconds) between data pushes.
    - ENABLE_ANOMALY: Turns on the injection of fake anomalies into stress test models.
    - USE_KAFKA: Regulates the output mode (Kafka topic or local file).

    Before streaming starts, sensor recordings are loaded and saved in memory.
- **Function of Anomaly Injection:**

  To replicate authentic manufacturing flaws:

    - Data points are modified with a 5% probability (anomaly_chance = 0.05).
    - The following people introduce anomalies artificially:
        - The vibration field is multiplied by a factor (1.5x to 3.0x).
        - The failure flag is being forced to 1.
        - Adding a comment ("anomaly_injected") for tracking.

  In Weeks 6–7, this stage guarantees that machine learning pipelines are verified against noisy inputs and edge cases.

- **Function of Data Streaming:**
  A live sensor feed is simulated by the data_streamer() thread:
    - Bits of BATCH_SIZE data are read.
    - The current system time is used to timestamped each record.
    - Anomalies are injected if desired.
    - Records that have been processed are added to a thread-safe queue in Python.Queue(), which simulates ingress from sensors in real time.
- **Data Ingestion: Simulation Based on Files:**
  Writing streaming records is handled by the file_data_ingestor():
    - The queue's batches are gathered in a buffer.
    - Each buffer is written to the stream_batch_<index> file, which has a unique name, when it is full.JSON.
    - A log of a stream manifest records:
        - filename for the batch.
        - quantity of records that were kept.
        - creation time stamp.

  Transparency and batch traceability are provided by serializing this metadata as stream_manifest.json.
- **Ingestion of Data: Kafka Placeholder:**
  To guarantee scalability in the future:
    - The function kafka_data_ingestor() is defined as a placeholder.
    - sends JSON-encoded records to the smartfactory-stream topic using KafkaProducer.
    - When Kafka streaming is enabled (USE_KAFKA=True) in Week 6, this block will become active.

  The script currently uses local file-based streaming by default.
- **Multiple-Threaded Performance:**
  Two distinct threads are started at the same time:
    - stream_thread: Generates the data stream by using data_streamer().
    - ingest_thread: Utilizes the USE_KAFKA flag to execute the ingesting logic.

Complete data flow simulation is ensured by the concurrent operation and graceful termination of both threads.

Results & Observations:

- **Streaming Pipeline Emulation:**
    - By pushing records in predetermined bursts, the script effectively simulates streaming behavior.
    - System logs provide unambiguous feedback on files saved and records processed, confirming batch generation.
- **Anomaly Simulation Functional:**
    - Throughout several batches, anomalies naturally arise.
    - By tagging injected abnormalities, anomaly detection logic and downstream validation may be compared.
- **Manifest Logging:**
    - A verifiable synopsis of the complete streaming session is provided by stream_manifest.json.
    - Debugging, auditing, and evaluating data drift or batch quality over time all depend on this.
- **Kafka-Ready Design:**
    - This block guarantees future extensibility for enterprise-level deployment, even though it is not operational in Week 5.
    - Allows for minimum modification as the system transitions from development simulation to real-time production ingestion.

Summary:

The operating environment of a smart manufacturing floor is simulated in Week 5, with an emphasis on data flow as opposed to inference. Among the principal achievements are:

- A sensor with a data stream that functions in real time.
- Anomaly injection embedded for reliable testing.
- Two ingestion routes (Kafka-ready and local file).
- Tracking metadata using manifest logs.

In Week 6, real-time anomaly detection, decision delay, and alert thresholds will be assessed thanks to this environment, which permits thorough testing of prediction models.

Conclusion:

In order to prepare SmartFactory.AI for real-time inference, anomaly detection, and reactive control systems, the Week 5 pipeline allows for a regulated yet adaptable simulation of sensor data flow. Now that this framework is established, Week 6 can concentrate on integrating prediction models with real-time data to close the gap between data creation and wise decision-making.

URL To [Week 5 – IoT Integration and Data Pipeline Setup.ipynb]