

Important

This is a step-by-step guide for running this project. Follow each step as outlined here.

1. Prerequisites and dataset

Note

The project root directory refers to: `C:\Users\XXX\Desktop\realtime_fraud_detection` or similar to this

Device requirement

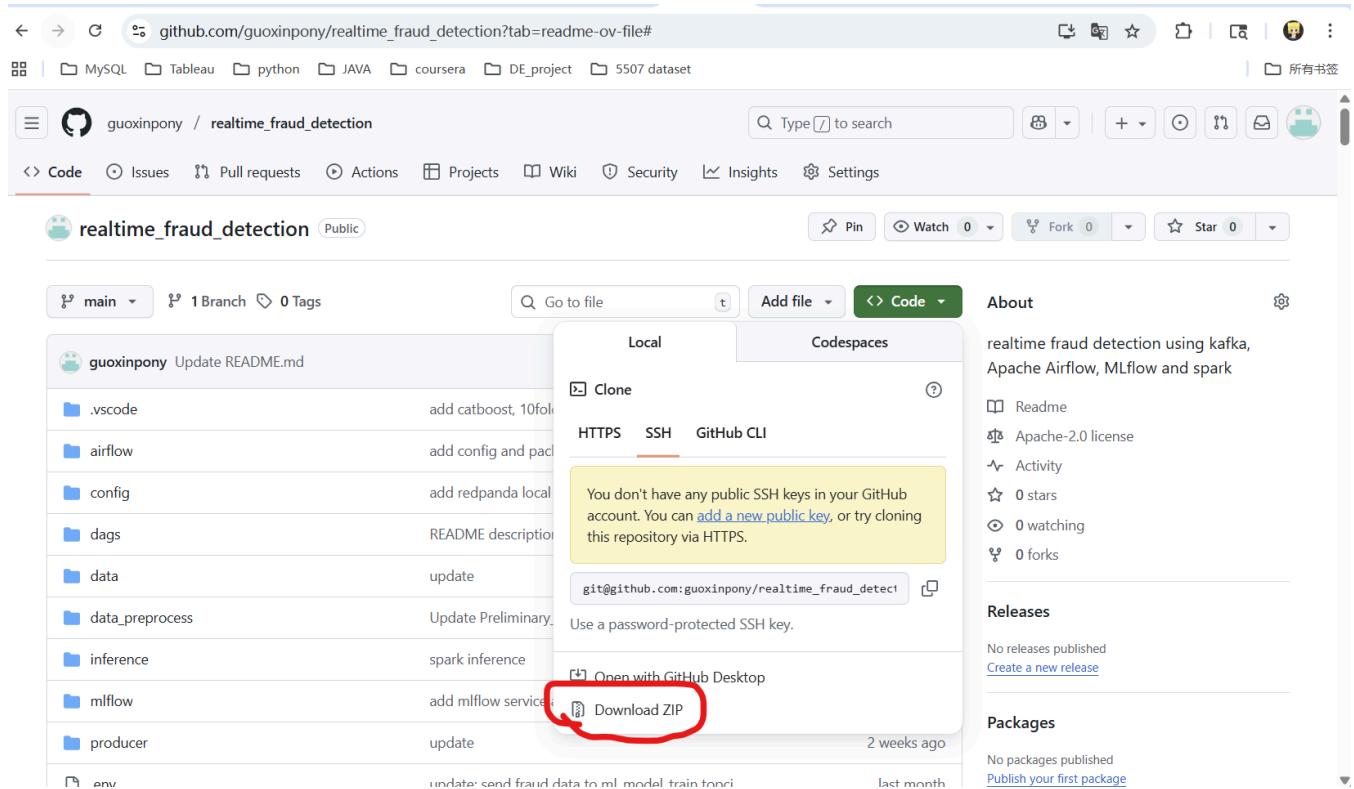
- **CPU:** 4+ cores recommended
- **Memory:** 8GB minimum, 16GB recommended
- **OS:** Linux, macOS, or Windows with WSL2

Clone or download project:

```
git clone https://github.com/guoxinpony/realtime_fraud_detection.git
```

OR

Download ZIP:



Install Docker Desktop:

1. download package from: <https://www.docker.com/products/docker-desktop/>
2. Install docker desktop. If you are using Windows system, the installation process will prompt you to enable WSL2. Follow the instructions to enable WSL2 and restart your computer.
3. Launch Docker Desktop

Download Dataset:

1. Download from: https://drive.google.com/file/d/1y1QqL1BdJKMpEu4dOB5OKANPeUxIkI3X/view?usp=drive_link, and place the dataset in the data directory within the project.

OR

2. Download from Kaggle: <https://www.kaggle.com/datasets/kartik2112/fraud-detection>, including two datasets: fraudTest.csv and fraudTrain.csv; and place the two datasets in the data directory within the project; and RUN in location of project root directory:

```
python ./data_preprocess/merge_data.py
```

Configure Environment Variables (.env file):

Make sure `.env` file in the project root directory with the following content:

```
# MinIO Configuration
AWS_ACCESS_KEY_ID=minio
AWS_SECRET_ACCESS_KEY=minio123
MINIO_USERNAME=minio
MINIO_PASSWORD=minio123

# Airflow UID
AIRFLOW_UID=50000

# kafka/Redpanda
KAFKA_BOOTSTRAP_SERVERS=kafka_broker:9092
KAFKA_TOPIC=m1_model_train
```

Note

The `.env` file is required for the system to connect to MinIO (object storage) and MLflow. The default values shown above match the MinIO login credentials used in step (5) of section 3. If you change the MinIO username or password, make sure to update both the `.env` file and the MinIO login credentials accordingly.

MAC OS/ Linux required: Change permission of Script in location of project root directory:

```
chmod +x wait-for-it.sh
```

2.Docker image build

The initial build takes some time, which includes downloading the official image, necessary packages, and the build process itself. The exact duration depends on your computer's performance and network connection.

In the project root directory, build the following four images airflow-webserver, mlflow-server, producer, and inference. Input in the terminal:

```
docker compose build airflow-webserver
```

```
docker compose build mlflow-server
```

```
docker compose build producer
```

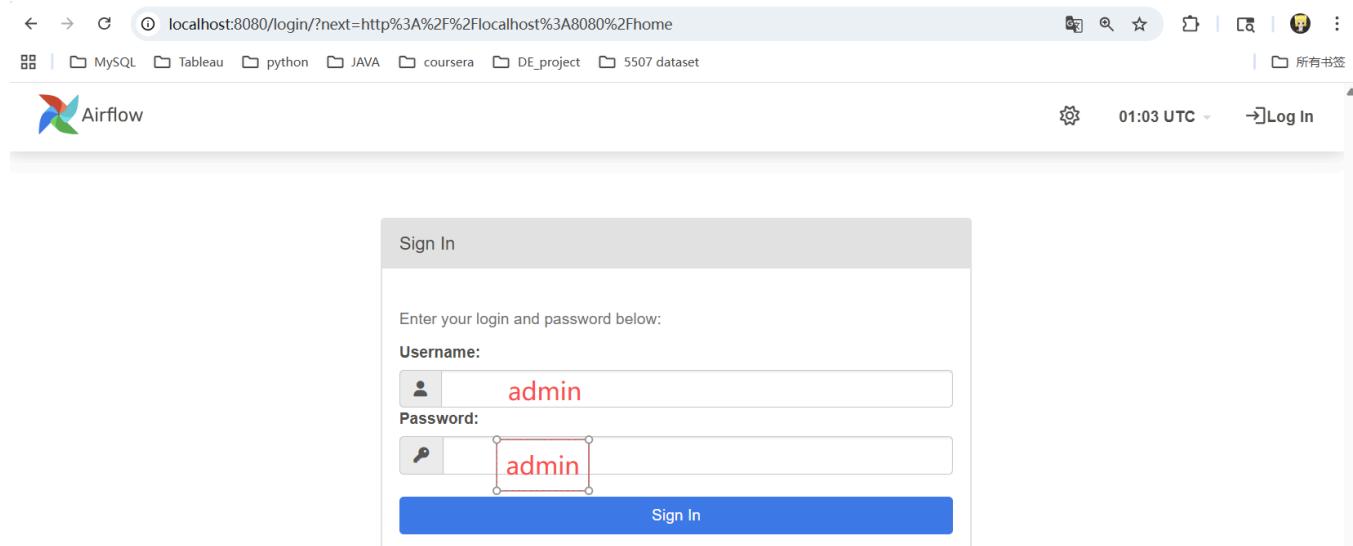
```
docker compose build inference
```

3.Start Services and Check

(1)Use the following command to start all services(containers), and waiting for all container run successfully:

```
docker compose up -d
```

(2) Open <http://localhost:8080/home> in your browser, then enter the username admin and password admin; If the page fails to open, it indicates that the service has failed to run:



You should be able to see:

The screenshot shows the Airflow web interface at localhost:8080/home. The top navigation bar includes links for MySQL, Tableau, python, JAVA, coursera, DE_project, and 5507 dataset. The main header has sections for Airflow, DAGs, Cluster Activity, Datasets, Security, Browse, Admin, and Docs. A status bar indicates 01:04 UTC and shows the AF logo.

The main content area is titled "DAGs". It displays a table of running DAGs:

DAG	Owner	Runs	Schedule	Last Run	Next Run	Recent Tasks
fraud_detection_catboost_training	fraud_detection_project	16	0 2 * * *	2025-11-05, 19:33:16	2025-11-05, 02:00:00	3
fraud_detection_decision_tree_training	fraud_detection_project	2	0 3 * * *	2025-11-05, 20:50:41	2025-11-05, 03:00:00	3
fraud_detection_extra_tree_training	fraud_detection_project	2	0 4 * * *	2025-11-05, 20:51:00	2025-11-05, 04:00:00	3
fraud_detection_lightgbm_training	fraud_detection_project	3	0 3 * * *	2025-11-05, 22:36:47	2025-11-05, 03:00:00	3
fraud_detection_logistic_training	fraud_detection_project	8	0 2 * * *	2025-11-04, 19:26:30	2025-11-04, 02:00:00	3
fraud_detection_mlp_training	fraud_detection_project	0	0 5 * * *		2025-11-04, 05:00:00	
fraud_detection_random_forest_training	fraud_detection_project	2	30 2 * * *	2025-11-05, 19:45:29	2025-11-05, 02:30:00	3

(3) Open <http://localhost:5500/> in your browser, If this is the first time opening it, this interface should be blank, and the `fraud_detection` directory does not exist:

ⓘ Note

After running a task that trains a model in Airflow, a `fraud_detection` directory will appear.

The screenshot shows the mlflow web interface at localhost:5500/#/experiments. The top navigation bar includes links for MySQL, Tableau, python, JAVA, coursera, DE_project, and 5507 dataset. The main header has sections for mlflow, GitHub, and Docs. A sidebar on the left shows options for Experiments, Models, and Prompts. The main content area is titled "Experiments".

The experiments table lists two entries:

Name	Time created	Last modified	Description	Tags
fraud_detection	09/23/2025, 09:46:58 PM	09/23/2025, 09:46:58 PM	-	
Default	09/21/2025, 01:19:31 AM	09/21/2025, 01:19:31 AM	-	

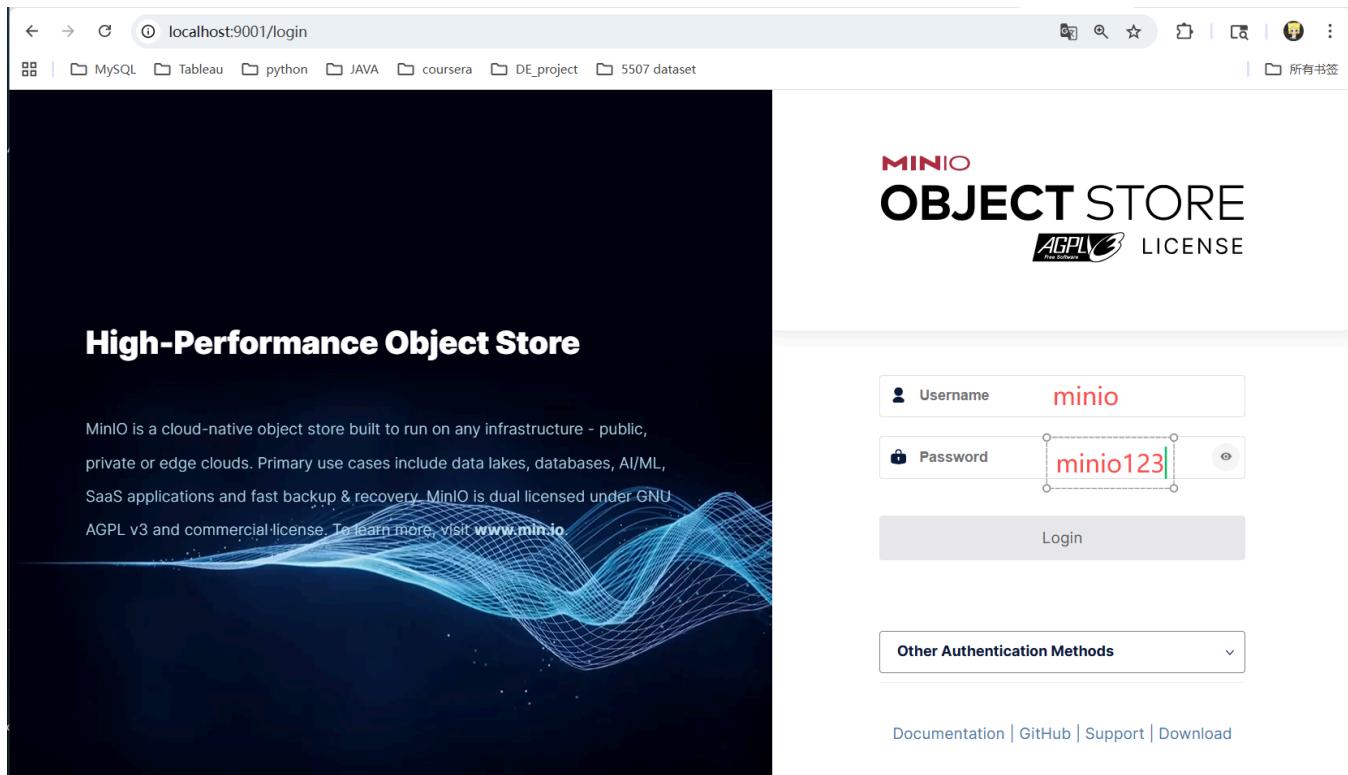
(4) Open <http://localhost:8090/overview> in your browser, you may notice the growing volume of data.

The screenshot shows the Redpanda cluster overview. On the left sidebar, there are several navigation options: Overview, Topics, Schema Registry, Consumer Groups, Security, Connectors, Transforms, and Reassign Partitions. The main area is titled 'Overview' and displays cluster status as 'Running'. A red circle highlights the 'Cluster Storage Size' field, which shows '676 MiB'. Below this, the 'Cluster Version' is listed as 'v0.11.0'. To the right, it shows '1 of 1 Brokers Online', '4 Topics', and '4 Replicas'. Under 'BROKER DETAILS', there is a table with one row showing ID 0, Status Running, and Size 676 MiB. The 'CLUSTER DETAILS' section includes sections for SERVICES (Kafka Connect Not configured, Schema Registry Healthy, 0 schemas), STORAGE (Total Bytes 676 MiB, Primary 676 MiB, Replicated 0 B), and SECURITY (Service Accounts Admin API not configured). At the bottom left, there is a 'Collapse sidebar' button.

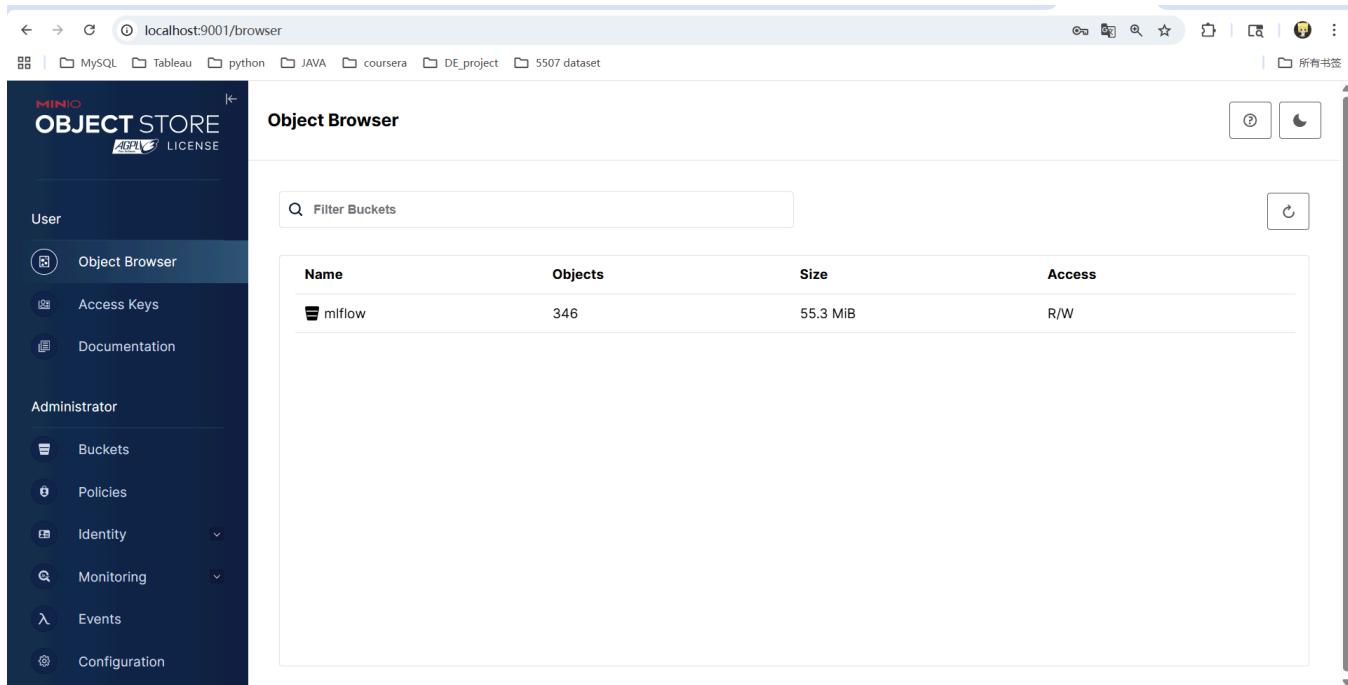
(5) Open <http://localhost:5555/> in your browser, You should be able to see two workers online.

The screenshot shows the Flower web interface at port 5555. The top navigation bar includes links for Worker, Tasks, Broker, and Documentation. The main content area has a search bar and a table titled 'Show 15 workers'. The table columns are: Worker, Status, Active, Processed, Failed, Succeeded, Retried, and Load Average. There are two entries: 'celery@e38e3c996c37' and 'celery@7d213b742818', both marked as 'Online'. The 'Total' row shows 0 Active, 3 Processed, 0 Failed, 3 Succeeded, and 0 Retried. The 'Load Average' column lists '0.38, 0.37, 0.37' for each worker. At the bottom, it says 'Showing 1 to 2 of 2 workers' and includes navigation buttons for Previous, Next, and a page number input field set to 1.

(6) Open <http://localhost:9001/> in your browser, Username is minio, password is minio123:



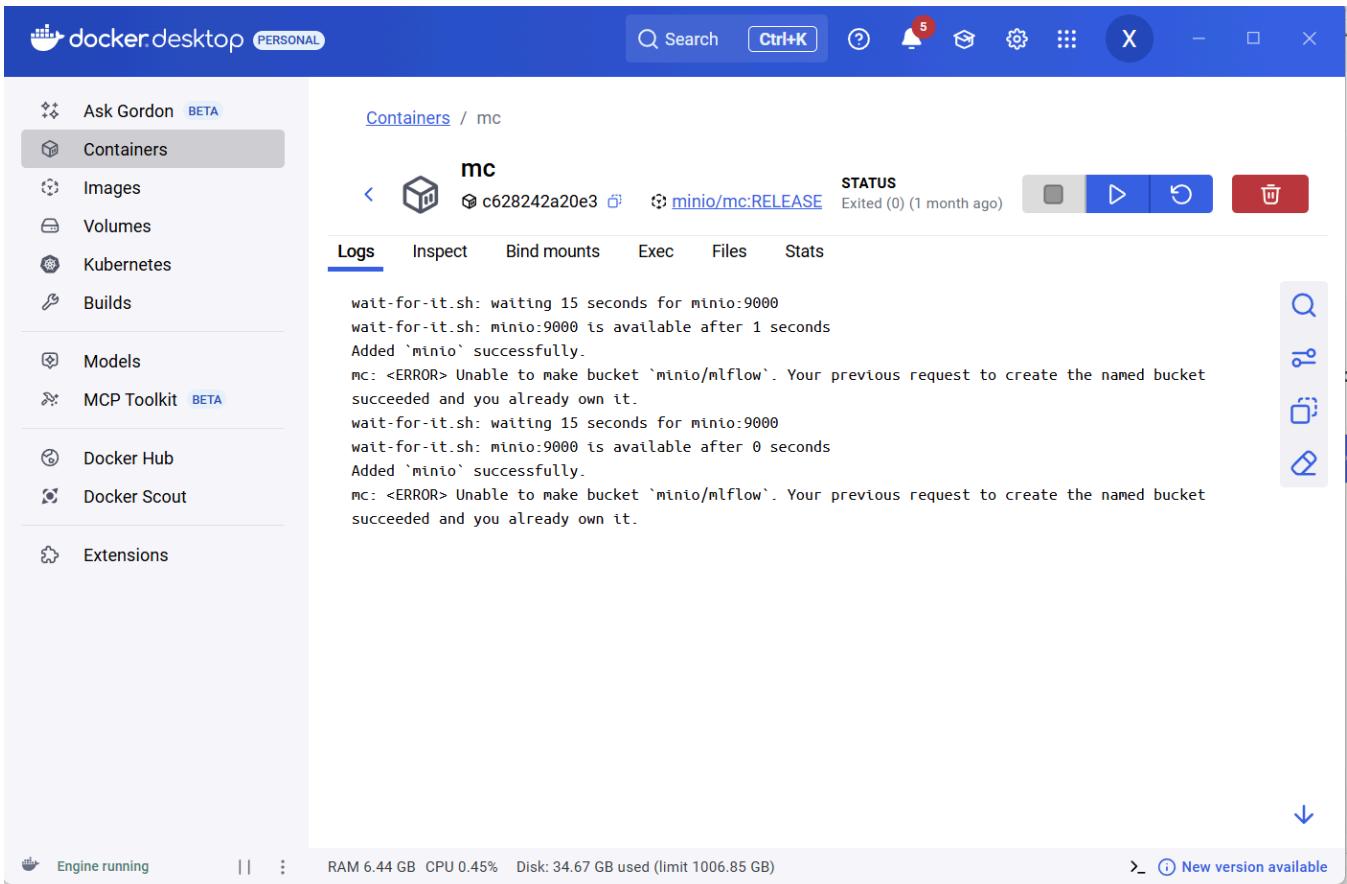
After logging in, you will see:



(7) In Docker Desktop, Open the container named `mc`.

i Note

If you see the prompt "Added `minio` successfully," it indicates you have successfully created a storage bucket named `mlflow` in Minio.



4. Model Training and Viewing Inference Results

(1) Access Airflow Web UI:

1. Open <http://localhost:8080/home> in your browser
2. Log in with username `admin` and password `admin`

(2) Find and Enable DAGs:

In the Airflow home page, you will see a list of DAGs (Directed Acyclic Graphs) for different model training tasks

1. Each DAG corresponds to a different machine learning model:

- o `fraud_detection_xgboost_training` - XGBoost model
- o `fraud_detection_lightgbm_training` - LightGBM model
- o `fraud_detection_logistic_training` - Logistic Regression model
- o `fraud_detection_catboost_training` - CatBoost model
- o `fraud_detection_mlp_training` - MLP (Neural Network) model
- o `fraud_detection_random_forest_training` - Random Forest model
- o `fraud_detection_decision_tree_training` - Decision Tree model
- o `fraud_detection_extra_tree_training` - Extra Trees model
- o `fraud_detection_tabnet_training` - TabNet model

2. By default, DAGs are paused. To enable a DAG:

- Click the toggle switch on the left side of the DAG name to turn it ON (Blue)
- The DAG will now be scheduled to run automatically according to its schedule

DAG Name	Description	Last Run	Run ID	State	Next Run	Progress
fraud_detection_decision_tree_training	decision-tree fraud	2025-11-05, 20:50:41	1	Running	2025-11-05, 03:00:00	0 3 ***
fraud_detection_extra_tree_training	extra-tree fraud	2025-11-05, 20:51:00	1	Running	2025-11-05, 04:00:00	0 4 ***
fraud_detection_lightgbm_training	fraud lightgbm	2025-11-05, 22:36:47	1	Running	2025-11-05, 03:00:00	0 3 ***
fraud_detection_logistic_training	fraud logistic_regression	2025-11-04, 19:26:30	1	Running	2025-11-05, 02:00:00	0 2 ***
fraud_detection_mlp_training	fraud mlp			Queued	2025-11-05, 05:00:00	0 5 ***
fraud_detection_random_forest_training	fraud random_forest	2025-11-05, 19:45:29	1	Running	2025-11-05, 02:30:00	30 2 ***
fraud_detection_tabnet_training	fraud tabnet	2025-11-05, 19:46:57	1	Running	2025-11-05, 03:00:00	0 3 ***
fraud_detection_training_TEST	fraud ML			Queued	2025-11-05, 03:00:00	0 3 ***
fraud_detection_xgboost_training	fraud xgboost	2025-11-05, 19:33:20	1	Running	2025-11-05, 02:00:00	0 2 ***

(3) Trigger Model Training:

Option A: Manual Trigger (Recommended for first-time users)

1. Click on the DAG name you want to run (e.g., `fraud_detection_xgboost_training`)
2. Click the "Play" button (▶) in the top right corner
3. Select "Trigger DAG" from the dropdown menu
4. The DAG run will start immediately

DAG: fraud_detection_xgboost_training Daily XGBoost fraud detection model training

Schedule: 0 2 * * * Next Run ID: 2025-11-06, 02:00:00 UTC

DAG Docs

2025/11/06 20:22:07 All Run Types All Run States Clear Filters

Press shift + / for Shortcuts

DAG fraud_detection_xgboost_training / Run 2025-11-05, 02:00:00 UTC Task train_xgboost_model

Details Graph Gantt Code Event Log Logs XCom Task Duration

All Levels All File Sources Wrap Download See More

```
7d213b742818
▶ log message source details
[2025-11-06, 20:20:13 UTC] {local_task_job_runner.py:123} ▶ Pre task execution logs
[2025-11-06, 20:20:17 UTC] {xgboost_model_train.py:253} INFO - Starting XGBoost fraud detection training workflow with 10-fold cross-validation.
[2025-11-06, 20:20:17 UTC] {conn.py:388} INFO - <BrokerConnection node_id=bootstrap-0 host=kafka_broker:9092 <connecting> [IPv4 ('172.18.0.14', 9092)]: connecting to kafka_broker
[2025-11-06, 20:20:17 UTC] {conn.py:1285} INFO - Probing node bootstrap-0 broker version
[2025-11-06, 20:20:17 UTC] {conn.py:418} INFO - <BrokerConnection node_id=bootstrap-0 host=kafka_broker:9092 <connecting> [IPv4 ('172.18.0.14', 9092)]: Connection complete.
[2025-11-06, 20:20:17 UTC] {conn.py:1267} INFO - Broker version identified as 2.5.0
[2025-11-06, 20:20:17 UTC] {conn.py:1267} INFO - Set configuration api_version=(2, 5, 0) to skip auto check_version requests on startup
[2025-11-06, 20:20:17 UTC] {consumer.py:118} WARNING - group_id is None: disabling auto-commit.
[2025-11-06, 20:20:17 UTC] {subscription_state.py:171} INFO - Updating subscribed topics to: ('ml_model_train',)
[2025-11-06, 20:20:17 UTC] {xgboost_model_train.py:177} INFO - Consuming messages from Kafka topic ml_model_train
```

Option B: Automatic Schedule

- If you enabled the DAG and it has a schedule, it will run automatically at the scheduled time
- You can check the schedule by clicking on the DAG and viewing its details

(4) Monitor Training Progress:

DAG: fraud_detection_xgboost_training / Run 2025-11-05, 02:00:00 UTC Task train_xgboost_model

Details Graph Gantt Code Event Log Logs XCom Task Duration

All Levels All File Sources Wrap Download See More

```
[2025-11-06, 20:20:44 UTC] {xgboost_model_train.py:461} INFO - Processing Fold 4/10
[2025-11-06, 20:20:50 UTC] {xgboost_model_train.py:554} INFO - Fold 4: F1=0.1429, Precision=0.0775, Recall=0.9167, ROC-AUC=0.9723
[2025-11-06, 20:20:50 UTC] {xgboost_model_train.py:461} INFO - Processing Fold 5/10
[2025-11-06, 20:21:00 UTC] {xgboost_model_train.py:554} INFO - Fold 5: F1=0.1424, Precision=0.0669, Recall=0.9167, ROC-AUC=0.9661
[2025-11-06, 20:21:00 UTC] {xgboost_model_train.py:461} INFO - Processing Fold 6/10
[2025-11-06, 20:21:06 UTC] {xgboost_model_train.py:554} INFO - Fold 6: F1=0.0775, Precision=0.0404, Recall=0.9167, ROC-AUC=0.9378
[2025-11-06, 20:21:06 UTC] {xgboost_model_train.py:461} INFO - Processing Fold 7/10
[2025-11-06, 20:21:12 UTC] {xgboost_model_train.py:554} INFO - Fold 7: F1=0.0848, Precision=0.0440, Recall=0.9167, ROC-AUC=0.9611
[2025-11-06, 20:21:12 UTC] {xgboost_model_train.py:461} INFO - Processing Fold 8/10
[2025-11-06, 20:21:18 UTC] {xgboost_model_train.py:554} INFO - Fold 8: F1=0.0984, Precision=0.0520, Recall=0.9167, ROC-AUC=0.9655
[2025-11-06, 20:21:18 UTC] {xgboost_model_train.py:461} INFO - Processing Fold 9/10
[2025-11-06, 20:21:26 UTC] {xgboost_model_train.py:461} INFO - Fold 9: F1=0.1164, Precision=0.0621, Recall=0.9167, ROC-AUC=0.9719
[2025-11-06, 20:21:35 UTC] {xgboost_model_train.py:554} INFO - Processing Fold 10/10
[2025-11-06, 20:21:35 UTC] {xgboost_model_train.py:554} INFO - Fold 10: F1=0.1262, Precision=0.0676, Recall=0.9444, ROC-AUC=0.9691
[2025-11-06, 20:21:35 UTC] {xgboost_model_train.py:580} INFO - Cross-validation completed. Mean F1=0.1221±0.0317, Mean Precision=0.0657±0.0185, Mean Recall=0.9190±0.0085
[2025-11-06, 20:21:35 UTC] {xgboost_model_train.py:580} INFO - Training final model on all training data...
[2025-11-06, 20:21:42 UTC] {xgboost_model_train.py:644} INFO - Final test set evaluation: F1=0.1527, Precision=0.0832, Recall=0.9222, ROC-AUC=0.9576
[2025-11-06, 20:21:43 UTC] {logging_mixin.py:190} WARNING - 2025/11/06 20:21:43 WARNING miflow.models.model: 'artifact_path' is deprecated. Please use 'name' instead.
[2025-11-06, 20:21:48 UTC] {credentials.py:1147} INFO - Found credentials in environment variables.
[2025-11-06, 20:21:49 UTC] {logging_mixin.py:190} WARNING - Registered model 'fraud_detection_xgboost' already exists. Creating a new version of this model...
[2025-11-06, 20:21:49 UTC] {logging_mixin.py:190} WARNING - 2025/11/06 20:21:49 INFO miflow.store.model_registry.abstract_store: Waiting up to 300 seconds for model version
```

- After triggering a DAG, you will see the DAG run appear in the "DAG Runs" view
- Click on the DAG run to see the task execution graph
- Each task will show its status:
 - Queued** (gray) - Task is waiting to be executed
 - Running** (light blue) - Task is currently executing

- **Success** (green) - Task completed successfully

- **Failed** (red) - Task encountered an error

4. Click on a task to view its logs and details

(5) Training Time and Resources:

Note

- Model training typically takes 5-30 minutes depending on:
 - The model type (XGBoost and LightGBM are faster, neural networks take longer)
 - The amount of training data
 - Your computer's CPU and memory
- Training uses data from the `m1_model_train` Kafka topic or falls back to the CSV file specified in `config.yaml`
- Trained models are saved to `/app/models/` directory and registered in MLflow

(6) Verify Model Training Success:

1. Check MLflow UI: <http://localhost:5500/>

- After successful training, you should see:
 - A new experiment run in the `fraud_detection` experiment
 - Model metrics (precision, recall, F1-score, ROC-AUC, etc.)
 - The trained model artifact

2. Check model files:

- In Docker Desktop, verify that model files are created in the `models` directory
- Model files are named like: `fraud_detection_model_xgboost.pkl`,
`fraud_detection_model_lightgbm.pkl`, etc.

3. Check Airflow task logs:

- If a task fails, click on it to view error logs
- Common issues: insufficient memory, missing data, or configuration errors

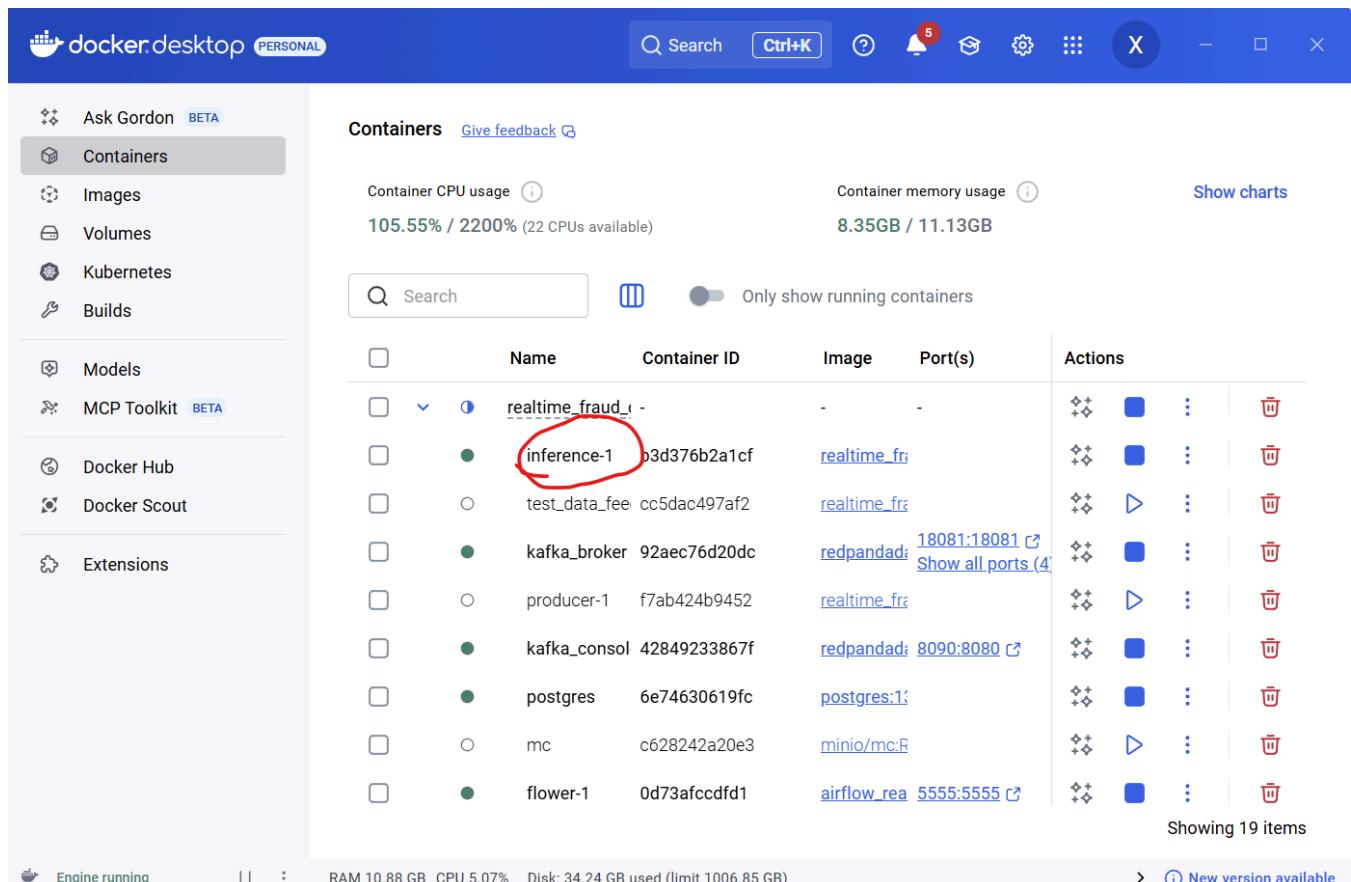
(7) Model Selection for Inference:

Note

The inference service uses the model specified in `config.yaml` under `models.xgboost.path` (default: `/app/models/fraud_detection_model_xgboost.pkl`). To use a different model:

1. Train the desired model first
2. Update `config.yaml` to point to the correct model file
3. Restart the inference container: `docker compose restart inference`

(8) In Docker Desktop, click the Containers panel on the left and navigate to the inference container:



The screenshot shows the Docker Desktop interface with the 'Containers' panel selected. The main area displays a list of running containers. One container, 'inference-1', is highlighted with a red circle. The table columns are: Name, Container ID, Image, Port(s), and Actions. The 'Actions' column includes icons for restarting, stopping, pausing, and deleting each container.

	Name	Container ID	Image	Port(s)	Actions
realtime_fraud_1	b3d376b2a1cf	realtime_fraud			[Actions]
inference-1	b3d376b2a1cf	realtime_fraud			[Actions]
test_data_fee	cc5dac497af2	realtime_fraud			[Actions]
kafka_broker	92aec76d20dc	redpanda:2	18081:18081	[Show all ports (4)]	[Actions]
producer-1	f7ab424b9452	realtime_fraud			[Actions]
kafka_console	42849233867f	redpanda:2	8090:8080	[Show all ports (4)]	[Actions]
postgres	6e74630619fc	postgres:12			[Actions]
mc	c628242a20e3	minio/mc:2			[Actions]
flower-1	0d73afccdf1	airflow_reactor	5555:5555	[Show all ports (4)]	[Actions]

Note

If you see information similar to the image below, don't worry—this indicates Spark is functioning normally and processing incoming credit card data.

The screenshot shows the Docker Desktop interface. On the left, a sidebar contains links like Ask Gordon (BETA), Containers (selected), Images, Volumes, Kubernetes, Builds, Models, MCP Toolkit (BETA), Docker Hub, Docker Scout, and Extensions. The main area displays the logs for the container 'realtime_fraud_detection-inference-1'. The container status is 'Exited (143) (27 minutes ago)'. The logs show several INFO messages related to checkpointing and streaming state management. At the bottom, a note indicates a new version is available.

```
25/11/06 19:06:11 INFO CheckpointFileManager: Writing atomically to file:/app/checkpoints/inference/offsets/21 using temp file file:/app/checkpoints/inference/offsets/.21.aafe9104-3f44-4050-96b9-c83a46917a68.tmp
25/11/06 19:06:11 INFO CheckpointFileManager: Renamed temp file file:/app/checkpoints/inference/offsets/.21.aafe9104-3f44-4050-96b9-c83a46917a68.tmp to file:/app/checkpoints/inference/offsets/21
25/11/06 19:06:11 INFO MicroBatchExecution: Committed offsets for batch 21. Metadata OffsetSeqMetadata(1546215154000,1762455971893,Map(spark.sql.streaming.stateStore.providerClass -> org.apache.spark.sql.execution.streaming.state.HDFSBackedStateStoreProvider, spark.sql.streaming.join.stateFormatVersion -> 2, spark.sql.streaming.stateStore.compression.codec -> lz4, spark.sql.streaming.stateStore.rocksdb.formatVersion -> 5, spark.sql.streaming.statefulOperator.useStrictDistribution -> true, spark.sql.streaming.flatMapGroupsWithState.stateFormatVersion -> 2, spark.sql.streaming.multipleWatermarkPolicy -> min, spark.databricks.sql.optimizer.pruneFiltersCanPruneStreamingSubplan -> false, spark.sql.streaming.aggregation.stateFormatVersion -> 2, spark.sql.shuffle.partitions -> 2))
25/11/06 19:06:11 INFO KafkaOffsetReaderAdmin: Partitions added: Map()
25/11/06 19:06:11 INFO KafkaOffsetReaderAdmin: Partitions added: Map()
25/11/06 19:06:11 INFO KafkaOffsetReaderAdmin: Partitions added: Map()
25/11/06 19:06:11 INFO BlockManagerInfo: Removed broadcast_21_piece0 on 1f6689446db5:34491 in memory
```

Note

Under the logs bar, if you see the log entry “`INFO MicroBatchExecution: Streaming query has been idle and waiting for new data more than 10000 ms,`” it indicates that the system's inference component is ready and waiting for new credit card data to be pushed, utilizing the machine learning model to make inferences!

To enable inference, simply start the `test_data_feeder` container, as Spark continuously monitors the `*fraud_test_data*` topic under Redpanda for incoming data.

Docker Desktop interface showing the `realtime_fraud_detection-inference-1` container. The logs tab is selected, displaying the following log entries:

```

25/11/06 01:23:42 INFO AdminClientConfig: These configurations '[key.deserializer, value.deserializer, enable.auto.commit, max.poll.records, auto.offset.reset]' were supplied but are not used yet.
25/11/06 01:23:42 INFO AppInfoParser: Kafka version: 3.6.1
25/11/06 01:23:42 INFO AppInfoParser: Kafka commitId: 5e3c2b738d253ff5
25/11/06 01:23:42 INFO AppInfoParser: Kafka startTimeMs: 1762392222684
25/11/06 01:23:52 INFO MicroBatchExecution: Streaming query has been idle and waiting for new data more than 10000 ms.
25/11/06 01:24:02 INFO MicroBatchExecution: Streaming query has been idle and waiting for new data more than 10000 ms.
25/11/06 01:24:12 INFO MicroBatchExecution: Streaming query has been idle and waiting for new data more than 10000 ms.
25/11/06 01:24:22 INFO MicroBatchExecution: Streaming query has been idle and waiting for new data more than 10000 ms.
25/11/06 01:24:32 INFO MicroBatchExecution: Streaming query has been idle and waiting for new data more than 10000 ms.
25/11/06 01:24:43 INFO MicroBatchExecution: Streaming query has been idle and waiting for new data more than 10000 ms.
25/11/06 01:24:53 INFO MicroBatchExecution: Streaming query has been idle and waiting for new data more than 10000 ms.
25/11/06 01:25:03 INFO MicroBatchExecution: Streaming query has been idle and waiting for new data more than 10000 ms.
25/11/06 01:25:13 INFO MicroBatchExecution: Streaming query has been idle and waiting for new data more than 10000 ms.
25/11/06 01:25:23 INFO MicroBatchExecution: Streaming query has been idle and waiting for new data more than 10000 ms.
25/11/06 01:25:33 INFO MicroBatchExecution: Streaming query has been idle and waiting for new data more than 10000 ms.
25/11/06 01:25:43 INFO MicroBatchExecution: Streaming query has been idle and waiting for new data more than 10000 ms.
25/11/06 01:25:53 INFO MicroBatchExecution: Streaming query has been idle and waiting for new data more than 10000 ms.
25/11/06 01:26:03 INFO MicroBatchExecution: Streaming query has been idle and waiting for new data more than 10000 ms.
25/11/06 01:26:13 INFO MicroBatchExecution: Streaming query has been idle and waiting for new data more than 10000 ms.
25/11/06 01:26:23 INFO MicroBatchExecution: Streaming query has been idle and waiting for new data more than 10000 ms.
25/11/06 01:26:33 INFO MicroBatchExecution: Streaming query has been idle and waiting for new data more than 10000 ms.
25/11/06 01:26:43 INFO MicroBatchExecution: Streaming query has been idle and waiting for new data more than 10000 ms.
25/11/06 01:26:53 INFO MicroBatchExecution: Streaming query has been idle and waiting for new data more than 10000 ms.
25/11/06 01:27:03 INFO MicroBatchExecution: Streaming query has been idle and waiting for new data more than 10000 ms.

```

Engine running | RAM 10.88 GB CPU 2.09% Disk: 34.24 GB used (limit 1006.85 GB) | > Terminal | New version available

(9) You have just started the `test_data_feeder` container. This container simulates banks and financial institutions sending data to this system.

ⓘ Note

If you stop the container as shown in the figure, the logs will display: `INFO - main2 - Shutting down dataset producer` and `INFO - main2 - Dataset streaming completed. Messages sent: 39`

Docker Desktop interface showing the `test_data_feeder` container. The logs tab is selected, displaying the following log entries:

```

2025-11-06 19:16:01,455 - INFO - main2 - Kafka producer ready (bootstrap=kafka_broker:9092)
2025-11-06 19:16:01,457 - INFO - main2 - Starting dataset streaming from /data/fraud_detection_data.csv (random selection mode)
2025-11-06 19:16:40,812 - INFO - main2 - Loaded 1852394 rows from CSV file
2025-11-06 19:16:42,190 - INFO - main2 - Randomly shuffled all 1852394 rows
2025-11-06 19:17:14,552 - INFO - main2 - Shutting down dataset producer
2025-11-06 19:17:16,300 - INFO - main2 - Dataset streaming completed. Messages sent: 17
2025-11-06 19:21:17,418 - INFO - main2 - Kafka producer ready (bootstrap=kafka_broker:9092)
2025-11-06 19:21:17,420 - INFO - main2 - Starting dataset streaming from /data/fraud_detection_data.csv (random selection mode)
2025-11-06 19:21:39,283 - INFO - main2 - Randomly selected and shuffled 10000 rows (max 10000, memory efficient)
2025-11-06 19:22:55,548 - INFO - main2 - Shutting down dataset producer
2025-11-06 19:22:57,361 - INFO - main2 - Dataset streaming completed. Messages sent: 39

```

Engine running | RAM 6.39 GB CPU 0.45% Disk: 34.67 GB used (limit 1006.85 GB) | > New version available

(10) Return to Redpanda: <http://localhost:8090/topics>

You'll find there are now three topics: ml_model_train, fraud_test_data, and fraud_predictions:

- **ml_model_train**: Store data sent by banks or financial institutions for model training
- **fraud_test_data**: Banks or financial institutions send data requiring inference to the fraud detection system
- **fraud_predictions**: Inference results from Spark; banks or financial institutions can read the results from this topic

(11) View Inference Results:

After starting the `test_data_feeder` container and data flows through the system, you can view the fraud detection predictions in several ways:

Method 1: View Predictions in Redpanda Console (Recommended)

The screenshot shows the Redpanda console interface. On the left, a sidebar menu includes 'Overview', 'Topics' (which is selected), 'Schema Registry', 'Consumer Groups', 'Security', 'Connectors', 'Transforms', and 'Reassign Partitions'. Below the sidebar is a 'Collapse sidebar' button. The main content area has a header bar with a back arrow, forward arrow, refresh icon, and search bar. The URL in the address bar is `localhost:8090/topics/fraud_predictions?p=-1&s=50&o=-1#messages`. The main table displays a single message. The table columns are 'TIMESTAMP', 'KEY', and 'VALUE'. The timestamp is 2025/11/6 13:05:34. The key is 0b242abb623afc578575680df30655b9 (TEXT - 32 B). The value is {"record_id":0,"transacted_at":"2019-01-01 00:00:18",...} (JSON - 984 B). Below the table, a detailed view of the JSON value is shown, highlighting the transaction record. At the bottom right of the main area are 'Copy Value' and 'Download Record' buttons.

1. Open Redpanda Console: <http://localhost:8090/topics>
2. Click on the `fraud_predictions` topic
3. Click on the "Messages" tab
4. You will see real-time prediction results with the following information:
 - Transaction details (transaction_id, amount, merchant, etc.)
 - Fraud probability (0.0 to 1.0)
 - Prediction (0 = legitimate, 1 = fraud)
 - Timestamp of the prediction

Note

The `fraud_predictions` topic contains the output from the inference pipeline. Each message represents a fraud detection prediction for a transaction.

Method 2: View Inference Logs in Docker Desktop

1. Open Docker Desktop
2. Navigate to the `inference` container
3. View the logs to see:
 - o Processing status of each transaction
 - o Any errors or warnings
 - o Throughput information

```
# View recent predictions
docker exec -it kafka_broker rpk topic consume fraud_predictions --num 10
```

Understanding Prediction Results:

- **fraud_probability:** A value between 0.0 and 1.0 indicating the model's confidence that the transaction is fraudulent
 - o Values closer to 1.0 indicate higher fraud likelihood
 - o Values closer to 0.0 indicate legitimate transactions
- **prediction:** Binary classification result
 - o `0` = Transaction is predicted to be legitimate
 - o `1` = Transaction is predicted to be fraudulent
- **decision_threshold:** The probability threshold used for classification (configurable in `config.yaml`)

Verifying System is Working:

1. Start the `test_data_feeder` container (if not already running)
2. Wait a few seconds for data to flow through the system
3. Check Redpanda Console - you should see:
 - o Messages appearing in `fraud_test_data` topic (input)
 - o Messages appearing in `fraud_predictions` topic (output)
4. Verify predictions contain reasonable values:
 - o Fraud probabilities should be between 0.0 and 1.0
 - o Predictions should be either 0 or 1
 - o Transaction IDs should match between input and output

Troubleshooting:

- If no predictions appear in `fraud_predictions` :
 - o Check that `test_data_feeder` is running and sending data
 - o Check inference container logs for errors
 - o Verify the model file exists in `/app/models/` directory

- Ensure the model path in `config.yaml` is correct
- If predictions seem incorrect:
 - Verify the model was trained successfully
 - Check that the model version matches the inference code
 - Review model metrics in MLflow to ensure good performance

5. Model Selection and Switching

Understanding Model Configuration:

The inference service uses the model specified in `config.yaml`. By default, it's configured to use the XGBoost model:

```
models:  
  xgboost:  
    path: "/app/models/fraud_detection_model_xgboost.pkl"
```

Available Models:

After training, the following model files are available in the `models` directory:

- `fraud_detection_model_xgboost.pkl` - XGBoost (default)
- `fraud_detection_model_lightgbm.pkl` - LightGBM
- `fraud_detection_model_logistic.pkl` - Logistic Regression
- `fraud_detection_model_catboost.pkl` - CatBoost
- `fraud_detection_model_mlp.pkl` - MLP (Neural Network)
- `fraud_detection_model_random_forest.pkl` - Random Forest
- `fraud_detection_model_decision_tree.pkl` - Decision Tree
- `fraud_detection_model_extra_tree.pkl` - Extra Trees
- `fraud_detection_model_tabnet.pkl` - TabNet

How to Switch Models:

Step 1: Train the Desired Model

1. Follow the model training steps in section 4
2. Ensure the model training completes successfully
3. Verify the model file exists in the `models` directory

Step 2: Update config.yaml

1. Open `config.yaml` in the project root directory
2. Locate the `models` section
3. Update the model path. For example, to switch to LightGBM:

```
models:  
  xgboost: # Note: The key name stays 'xgboost' but path changes  
    path: "/app/models/fraud_detection_model_lightgbm.pkl"
```

Or if you want to use a different model section (if available in config.yaml):

```
models:  
  lightgbm:  
    path: "/app/models/fraud_detection_model_lightgbm.pkl"
```

ⓘ Note

The inference service reads from `models.xgboost.path` by default. If your config.yaml has different model sections, you may need to update the inference code or ensure the path points to the correct model.

Step 3: Restart Inference Service

After updating the configuration:

```
# Restart the inference container to load the new model  
docker compose restart inference
```

Step 4: Verify Model Loaded

1. Check inference container logs:

```
docker compose logs inference | grep -i "model loaded"
```

2. You should see a message like:

```
INFO - Model loaded from /app/models/fraud_detection_model_lightgbm.pkl
```

Comparing Model Performance:

Before switching models, you can compare their performance in MLflow:

1. Open MLflow UI: <http://localhost:5500/>
2. Navigate to the `fraud_detection` experiment
3. Compare metrics across different model runs:
 - ROC-AUC (higher is better)
 - Precision (higher is better)
 - Recall (higher is better)
 - F1-Score (higher is better)

Model-Specific Considerations:

- **XGBoost/LightGBM/CatBoost:** Fast inference, good performance, recommended for production
- **Logistic Regression:** Very fast, interpretable, but may have lower accuracy

- **MLP/TabNet**: Slower inference, may require more memory
- **Tree-based models** (Random Forest, Decision Tree, Extra Trees): Good balance of speed and accuracy

Troubleshooting Model Switching:

- **Model file not found**: Ensure the model was trained successfully and the file exists
- **Model loading error**: Check that the model file is not corrupted and matches the inference code version
- **Inference errors**: Some models may require different feature engineering - verify the model was trained with compatible features
- **Performance issues**: Different models have different resource requirements - monitor CPU and memory usage

6. Stop Services

(1) Stop All Services:

To stop all running containers and services:

```
docker compose down
```

This command will:

- Stop all running containers
- Remove containers (but keep volumes and images)
- Stop the network

(2) Stop Specific Service:

To stop a specific service without affecting others:

```
# Stop a specific service (e.g., test_data_feeder)
docker compose stop test_data_feeder

# Or stop multiple specific services
docker compose stop test_data_feeder inference
```

(3) Stop and Remove Everything (Including Volumes):

Warning

This will delete all data including:

- Database data (PostgreSQL)
- MinIO storage data
- Kafka/Redpanda data
- Model files in volumes

```
# Stop and remove containers, networks, and volumes
docker compose down -v
```

(4) Stop Services but Keep Containers:

If you want to stop services but keep containers for quick restart:

```
# Stop containers but don't remove them
docker compose stop
```

Later, you can restart with:

```
docker compose start
```

(5) Restart Services:

To restart all services:

```
# Restart all services
docker compose restart

# Or restart a specific service
docker compose restart inference
```

(6) View Running Services:

To see which services are currently running:

```
docker compose ps
```

(7) View Service Logs:

To view logs of running services:

```
# View logs of all services
docker compose logs

# View logs of a specific service
docker compose logs inference

# Follow logs in real-time
docker compose logs -f inference

# View last 100 lines of logs
docker compose logs --tail=100 inference
```

(8) Clean Up Docker Resources (Optional):

If you want to free up disk space:

```
# Remove stopped containers
docker container prune

# Remove unused images
docker image prune

# Remove unused volumes (be careful - this deletes data)
docker volume prune

# Remove everything unused (containers, networks, images, volumes)
docker system prune -a --volumes
```

Note

- Use `docker compose down` for normal shutdown
- Use `docker compose stop` if you plan to restart soon
- Always backup important data before using `-v` flag with `docker compose down`