**Step 6: ML model for alert triage (with MLOps)**.  
Goal: take the alerts you’ve generated (aml.gold.alerts), add investigator dispositions from your **case DB**, train a model that predicts “True Positive vs False Positive”, and integrate it back into your gold layer for prioritization.

**6A) Data prep for triage model**

1. **Investigator labels**
   * From aml.cases table in Azure SQL, enrich each alert with disposition = {SAR, NotSAR, Escalated, etc.}.
   * Expose this table to Databricks with **JDBC/AAD connector**.
2. **Feature engineering**
   * Start with engineered features you already have: typology, risk\_score, evidence JSON values.
   * Add transaction aggregates (velocity, corridor counts, round-trip flags).
   * Join with case disposition labels.

**Notebook snippet:**

import pandas as pd

from pyspark.sql import functions as F

alerts = spark.read.table("aml.gold.alerts")

cases = (spark.read

.format("jdbc")

.option("url", "jdbc:sqlserver://<server>.database.windows.net:1433;database=amlcases")

.option("dbtable", "aml.cases")

.option("authentication", "ActiveDirectoryPassword") # or token-based

.load()

.select("alert\_id","status","disposition"))

# Join alerts with case outcomes

labeled = (alerts.join(cases, "alert\_id")

.filter(F.col("disposition").isNotNull())

.withColumn("label", F.when(F.col("disposition")=="SAR",1).otherwise(0)))

**6B) Model training (XGBoost + MLflow)**

Use Databricks MLflow for experiment tracking.

import mlflow

import mlflow.sklearn

import xgboost as xgb

from sklearn.model\_selection import train\_test\_split

from sklearn.metrics import classification\_report, roc\_auc\_score

pdf = labeled.toPandas()

X = pdf[["risk\_score"]] # start simple; extend with parsed evidence fields

y = pdf["label"]

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X,y,test\_size=0.2,random\_state=42)

with mlflow.start\_run():

model = xgb.XGBClassifier(max\_depth=4, n\_estimators=100, learning\_rate=0.1)

model.fit(X\_train, y\_train)

preds = model.predict(X\_test)

auc = roc\_auc\_score(y\_test, model.predict\_proba(X\_test)[:,1])

mlflow.log\_metric("auc", auc)

mlflow.sklearn.log\_model(model, artifact\_path="model", registered\_model\_name="aml\_alert\_triage")

print(classification\_report(y\_test,preds))

* Model is registered as aml\_alert\_triage in MLflow Registry.

**6C) Batch scoring pipeline**

Create a **job** that periodically scores new alerts (e.g., daily).

from mlflow.tracking import MlflowClient

import mlflow.pyfunc

client = MlflowClient()

model = mlflow.pyfunc.load\_model("models:/aml\_alert\_triage/Production")

# Read new alerts (not yet in cases)

alerts\_new = spark.read.table("aml.gold.alerts").filter("created\_ts >= current\_date() - interval 1 day")

pdf\_new = alerts\_new.toPandas()

scores = model.predict\_proba(pdf\_new[["risk\_score"]])[:,1]

pdf\_new["triage\_score"] = scores

scored = spark.createDataFrame(pdf\_new)

scored.write.mode("append").saveAsTable("aml.gold.alerts\_scored")

**6D) CI/CD & promotion**

* **Dev → Test → Prod model lifecycle**
  + Train in dev with historical data.
  + Promote candidate model to “Staging” in MLflow.
  + Run evaluation notebooks; if KPIs met (AUC, precision@topK), promote to “Production”.
* **Automation** with Databricks Jobs + GitHub Actions / ADO:
  + Code in /ml/triage\_model/.
  + CI pipeline runs unit tests + model training with small sample.
  + CD pipeline triggers full training on Databricks, logs to MLflow, requests approval for Production.

**6E) Power BI integration**

Extend vw\_alerts in Synapse to include triage\_score from aml.gold.alerts\_scored.

CREATE OR ALTER VIEW vw\_alerts\_scored AS

SELECT a.\*, s.triage\_score

FROM vw\_alerts a

LEFT JOIN OPENROWSET(

BULK 'alerts\_scored',

DATA\_SOURCE = 'ds\_gold',

FORMAT='DELTA'

) AS s

ON a.alert\_id = s.alert\_id;

In Power BI:

* Add triage\_score to the **Investigator Queue** table.
* Sort descending → investigators see highest priority alerts first.
* Add measure “Precision @ Top 20%” for model monitoring.

**✅ MVP Complete after Step 6**

* Secure landing zone + Databricks workspace.
* Medallion storage + Unity Catalog.
* Baseline typology rules producing gold.alerts.
* Power BI dashboards with queues/trends.
* Case management loop in Azure SQL.
* ML triage model trained, logged, and used to prioritize alerts.