# ML Engineer Coding Exercise

This is a production‑minded mini exercise. Keep it simple, readable and reproducible.

## You will build

1. Training command: reads config.yaml, trains a model, writes artifacts to ./artifacts

2. Prediction command: loads the saved model, scores the holdout file, writes predictions.csv

3. One unit test covering your preprocessing

4. Basic logging and clear error messages

## Inputs

• fraud\_train.csv — labelled, contains is\_fraud

• fraud\_holdout\_unlabelled.csv — unlabelled

• config.yaml — paths and feature lists

## Required outputs

• artifacts/model.joblib — persisted sklearn Pipeline with preprocessing and model

• artifacts/metrics.json — validation metrics as JSON

• predictions.csv — two columns: id, is\_fraud\_probability

## Functional requirements

• Use a scikit‑learn Pipeline combining preprocessing and a classifier

• Use a stratified split and report accuracy, precision, recall, F1, ROC AUC

• No hard‑coded paths or column names, read them from config.yaml

• Handle missing values sensibly

## Non‑functional requirements

• Reproducible: fixed random\_state

• Testable: add one pytest that asserts your preprocessing handles missing values

• Operable: log key steps and final artefact locations

## How to run

• python -m pip install -r requirements.txt

• python -m src.train --config config.yaml

• python -m src.predict --config config.yaml

## Bonus, pick one

A. Add a Dockerfile build and run instructions

B. Add the provided GitHub Actions workflow and make tests pass

C. Add a minimal SageMaker‑style inference module with model\_fn and predict\_fn signatures