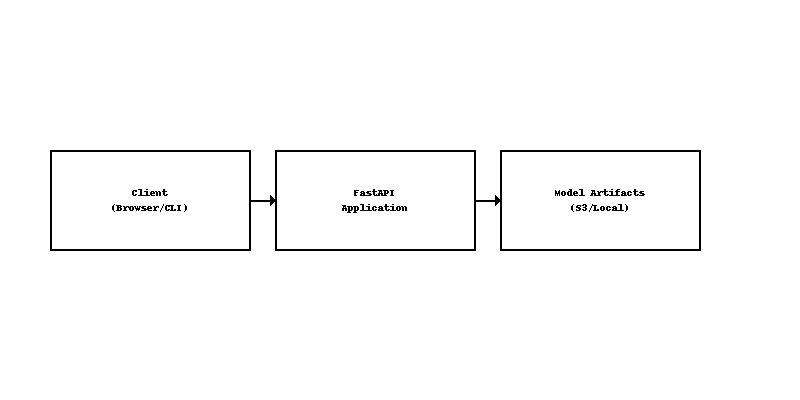
Fraud Detection FastAPI Application – Design Document

# High-Level Architecture



# 1. Introduction

## 1.1 Purpose

This document describes the architecture and design of the Fraud Detection FastAPI application, guiding developers and operations teams in understanding system components, data flows, and deployment considerations.

## 1.2 Scope

* • Summarize functional requirements: training, testing, inference, feature importance, and model ensemble.
* • Outline non-functional requirements: scalability, maintainability, security, and observability.
* • Define system boundaries, components, and interactions.

# 2. Functional Overview

## 2.1 Use Cases

* • Model Training: Train DNN and XGBoost models, persist artifacts.
* • Model Testing/Validation: Evaluate performance with metrics (AUC, accuracy, recall, precision).
* • Inference: Accept new claim data, preprocess, ensemble (soft-voting + threshold), return results.
* • Feature Importance: Compute mutual information and DNN-based importance for top-K features.

## 2.2 Actors & Roles

* • Data Scientist/Developer: Triggers training/testing, reviews metrics.
* • API Consumer: Sends JSON to /predict, receives responses.
* • Operations: Deploys, monitors, and scales the service.

# 3. Data Model & Storage

## 3.1 Input Schema (JSON)

{  
 "features": {  
 "Age": 45,  
 "VehiclePrice": "High",  
 ...  
 }  
}

## 3.2 Preprocessing

* • Numerical: Standard scaling.
* • Categorical: Target encoding, one-hot for low-cardinality, embeddings for high-cardinality.
* • Feature Selection: Top-K via mutual\_info + DNN ranking.

## 3.3 Artifact Storage

* • Preprocessor: Pickle (preprocessor.pkl).
* • DNN Model: Keras (.keras format).
* • XGBoost Model: Joblib (xgb\_model.pkl).

# 4. API Endpoints

|  |  |  |
| --- | --- | --- |
| **Path** | **Method** | **Description** |
| /train | POST | Trigger training; returns status and summary metrics. |
| /test | POST | Evaluate test set; returns metrics JSON. |
| /predict | POST | Score new data; returns probability and fraud flag. |
| /feature-importance | GET | Return top-K feature scores. |

# 5. Business Logic & Flow

* • Train Handler:
* - Load data, fit preprocessor, train DNN and XGBoost, save artifacts.
* • Predict Handler:
* - Load artifacts, preprocess input, average probabilities, apply threshold, return results.

# 6. Security & Validation

* • Validate JSON payloads with Pydantic schemas and type checks.
* • Use HTTPS and implement authentication/authorization for production.

# 7. Observability & Logging

* • Logging: Record requests, responses, metrics, and errors.
* • Metrics: Expose Prometheus metrics for monitoring.
* • Tracing: Optional OpenTelemetry integration.

# 8. Deployment & Scaling

* • Containerization with Docker (Python 3.11, Uvicorn/Gunicorn).
* • Kubernetes orchestration with HPA and CI/CD pipelines.

# 9. Testing Strategy

* • Unit Tests: Pytest for utils and service functions.
* • Integration Tests: FastAPI TestClient.
* • Performance Tests: Load testing with Locust.

# 10. Future Enhancements

* • Retraining scheduling (cron-based).
* • Incremental learning support.
* • Kafka integration for streaming inference.
* • Explainability with SHAP values.