ML System Design Document

AAI-540 ML Design Document – Fraud Detection in Financial Transactions

Team Info

Project Team Group : 4

Authors: Balubhai Sukani, Anwesha Sarangi, Soumi Ray

Business Name: Naturecon

Publication Date: October 18, 2025

GitHub Organization / User: Naturecon

Primary Environment: AWS SageMaker (JupyterLab)

Team Workflows

GitHub Project Link:

https://github.com/Naturecon/Fraud-Detection-in-Financial-Transactions

Asana Board Link:

https://app.asana.com/1/952672460738672/home

Team Tracker Link:
https://docs.google.com/document/d/1qofAxfuphcOQq7hDjpywJZcLZWMrW7p8V7HbUe37wCE/edit?usp=sharing

AWS And Data Locations:

Bucket (primary): bsukanisagemaker-afresh

Alternative bucket: bsukanisagemaker-aai-540

Dataset (S3): s3://bsukanisagemaker-afresh/aai-540-labs/lab-2-1-fresh/creditcard.csv

Model Artifacts:

Logistic Regression → `s3://bsukanisagemaker-afresh/aai-540-labs/lab-4-1-fraud-model/logistic_regression_fraud.pkl`

XGBoost → `s3://bsukanisagemaker-afresh/aai-540-labs/lab-4-1-fraud-model/xgboost_model.pkl`

Project Scope

Project Background

Financial institutions must detect fraudulent transactions in near real-time to minimize financial loss and preserve customer trust. The objective of this project is to develop a machine-learning system that classifies credit-card transactions as fraudulent or legitimate using historical data. This is a **supervised binary-classification** problem.

The dataset (Kaggle Credit-Card Fraud Detection) contains 284,807 transactions, of which 492 are labeled as fraud (~0.17 %). Features V1–V28 are PCA-transformed to protect privacy, along with Time and Amount. The system aims to deliver high recall to catch most frauds while controlling false positives through threshold tuning and secondary review mechanisms.

Technical Background:

The model will be evaluated using metrics like Recall, Precision, F1-score, ROC-AUC, and PR-AUC. The primary dataset comes from Kaggle (creditcard.csv) and contains 284,807 transactions with 492 labeled as fraud, making it highly imbalanced. Data preprocessing includes handling missing values, feature scaling, and stratified train/test split. Key features include V1–V28 (PCA-transformed), Time, and Amount. We plan to use Logistic Regression initially, with XGBoost as a potential enhancement.

Goals vs Non-Goals:

Goals:

Maximize recall for fraud detection.

Minimize false positives with secondary verification.

Deploy a functional ML model for near real-time detection.

Non-Goals:

Creating a full-scale production system (beyond MVP scope).

Incorporating demographic data or sensitive personal features.

Extensive deep learning models (not feasible in current timeline).

Solution Overview:

The ML system stores data in S3, preprocesses and engineers features, trains models using SageMaker, and deploys via SageMaker endpoints. Monitoring involves CloudWatch metrics, feature drift detection, and performance alerts.

Impact Measurement

The project measures impact through both **technical metrics** and **business KPIs**:

Technical Metrics:

Recall – minimize missed frauds (critical).

Precision – reduce false-positive alerts.

F1-Score – harmonic balance of precision and recall.

ROC-AUC and *PR-AUC* – overall discrimination power on imbalanced data.

Operational Metrics:

Manual review load per day.

Cost savings from fraud prevention.

Mean time to detect (MTTD).

Goal trade-off: Maintain recall > 0.90 for frauds while improving precision through secondary verification or ensemble models.

Solution Overview

The end-to-end system includes:

Data ingestion from S3 into SageMaker.

Data preprocessing and feature engineering in pandas and scikit-learn.

Model training and evaluation for two baselines: Logistic Regression and XGBoost.

Model deployment to SageMaker and local endpoint simulation for testing.

Model monitoring and CI/CD integration using CloudWatch and CodePipeline.

The architecture ensures reproducibility, scalability, and version tracking through **SageMaker Model Registry**. Both models are versioned as artifacts (v1.0 for Logistic Regression and v2.0 for XGBoost).

Data Sources

Dataset: Credit-Card Fraud Detection Dataset (Kaggle)

Transactions: 284,807 rows × 31 columns

Fraud Cases: 492 (0.17 %)

Storage: AWS S3 (`creditcard.csv`)

Risk: Extreme class imbalance; features anonymized via PCA.

The dataset is anonymized and contains no PII, ensuring compliance with privacy regulations.

Data Engineering

Loaded dataset directly from S3 into pandas.

Verified no missing values or type inconsistencies.

Separated features (X) and target (y = Class).

Stratified train/test split (80/20) to preserve fraud distribution.

StandardScaler applied to numeric features (Time, Amount).

Managed imbalance via `class_weight='balanced'` (Logistic Regression) and `scale_pos_weight` (XGBoost).

Feature Engineering

Base Features: V1–V28 (PCA-transformed), Time, Amount.

Engineered Features: `Amount_scaled`, time-of-day buckets, and transaction velocity.

Considered hour-of-day and aggregate patterns for future work.

Feature importance from both models shows V14, V17, and Amount as key predictors of fraud.

Model Training and Evaluation

1. Baseline Model – Logistic Regression

Configuration: `max_iter=1000`, `class_weight='balanced'`, `random_state=42`

Environment: AWS SageMaker Notebook

Key Results:

Accuracy: 0.98

Recall (Fraud): 0.92

Precision (Fraud): 0.06

F1-Score (Fraud): 0.11

AUC: 0.97

PR-AUC: 0.72

Insights: High recall ensures few missed frauds; low precision requires manual review or threshold tuning.

2. Comparative Model – XGBoost

Configuration: `scale_pos_weight = len(y == 0) / len(y == 1); eval_metric='aucpr'.`

Key Results:

Accuracy: ≈ 1.00

Recall (Fraud): 0.83

Precision (Fraud): 0.89

F1-Score (Fraud): 0.86

AUC: 0.96

PR-AUC: 0.88

Insights: XGBoost offers better balance between recall and precision than the logistic model. Fewer false positives improve operational efficiency.

Comparative Interpretation

Metric	Logistic Regression	XGBoost	Observation
Recall (Fraud)	0.92	0.83	LR detects more frauds
Precision (Fraud)	0.06	0.89	XGBoost reduces false alarms
F1-Score (Fraud)	0.11	0.86	XGBoost achieves balance
PR-AUC	0.72	0.88	XGBoost superior discrimination

Hence, **XGBoost becomes the preferred baseline for production**, while Logistic Regression remains the interpretable benchmark model.

Model Artifact and Storage

Local Paths: `fraud_model/logistic_regression_fraud.pkl`, `fraud_model/xgboost_model.pkl`

S3 Paths: Versioned and registered in `fraud-detection-model-group` under SageMaker Model Registry.

Serialization: Joblib for scikit-learn and XGBoost.

Versioning: Logistic Regression (v1.0); XGBoost (v2.0).

Model Deployment

Deployment performed using SageMaker model objects and inference scripts (`inference.py`).

Due to LabRole restrictions, inference tested locally via a Python endpoint simulation function (`local_endpoint_predict_df`).

Supports single or batch predictions with JSON responses.

Integration ready for future SageMaker endpoints with IAM-controlled access.

Model Monitoring

Monitoring via **AWS CloudWatch** and custom metrics includes:

Latency, 4xx/5xx error rates, and invocation counts.

Feature drift using Population Stability Index (PSI).

Model performance tracking (Recall, Precision, F1) over time.

Retraining trigger if recall drops below 0.85 or precision drops below 0.70.

Sample log outputs and alerts configured for continuous auditability.

Model CI/CD Pipeline

The CI/CD process follows a standard MLOps pattern:

Source: GitHub (Naturecon repository).

Build: Unit tests and static code analysis.

Train: Automated SageMaker job execution.

Deploy: Staging → production via AWS CodePipeline.

Validate: Canary deployments and integration tests.

Monitor: Pipeline dashboard with alerts for failed stages.

This design supports future automation for model retraining and A/B testing.

Recommended CI/CD pipeline elements:

Source: GitHub (Naturecon organization repository for this project).
Build: automated tests and static code analysis (unit tests for preprocessing and inference).
Train: trigger SageMaker training job (if model changes).
Deploy: automated deployment to staging and production endpoints via AWS CodePipeline / SageMaker Projects.
Validation: integration tests and canary deployments.
Observability: pipeline-level dashboards showing successful/failed runs (demonstrate both states in video).

Security Checklist and Ethical Considerations

Category	Safeguards and Actions
Data Security	Data stored in encrypted S3 buckets; IAM roles limit access to authorized users only.
PII Handling	Dataset fully anonymized (PCA features); no direct user information stored.
Bias Monitoring	Evaluate model for proxy bias if additional features introduced.
Explainability	Future use of SHAP for interpreting XGBoost outputs.
Ethics and Fairness	False positives reviewed by humans to reduce customer impact.
Compliance	Aligns with GDPR and HIPAA-like privacy principles.

Limitations and Risks

Severe class imbalance (0.17 % fraud).

Logistic Regression has low precision; XGBoost requires careful threshold tuning.

Fraud patterns evolve → concept drift risk necessitating periodic retraining.

Real-time deployment restricted by LabRole permissions (academic environment limitation).

Future Enhancements

Incorporate SMOTE or ensemble stacking to improve minority class detection.

Enable streaming scoring via AWS Kinesis + Lambda.

Build a feature store for reproducibility.

Integrate SHAP for model explainability to support auditing and trust.

Extend CI/CD with automated retraining triggers based on drift metrics.

Team Roles and Timeline

Balubhai Sukani (Lead): Model development, training, deployment, documentation.

Anwesha Sarangi: CI/CD pipeline integration, monitoring setup.

Soumi Ray: Documentation, demo video, QA and review.

Final Week Tasks: Finalize code, verify S3 artifacts, demonstrate CI/CD flow, compile design document, and record demo video.

References

Kaggle: Credit Card Fraud Detection Dataset (https://www.kaggle.com/mlg-ulb/creditcardfraud)

Scikit-learn Documentation (https://scikit-learn.org/)

XGBoost Documentation (https://xgboost.readthedocs.io/)

AWS SageMaker Documentation (https://docs.aws.amazon.com/sagemaker/)

Appendix A: Key Code Snippets

```
# Load dataset from S3 into pandas

import pandas as pd

s3_uri = 's3://bsukanisagemaker-afresh/aai-540-labs/lab-2-1-fresh/creditcard.csv'

data = pd.read_csv(s3_uri)


# Train/test split and scaling

from sklearn.model_selection import train_test_split

from sklearn.preprocessing import StandardScaler

X = data.drop(columns=['Class'])

y = data['Class']

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, stratify=y,
random_state=42)

scaler = StandardScaler()

X_train_scaled = scaler.fit_transform(X_train)

X_test_scaled = scaler.transform(X_test)
```

Logistic Regression Training and Evaluation

```
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import classification_report, roc_auc_score

log_model = LogisticRegression(max_iter=1000, class_weight='balanced',
random_state=42)
log_model.fit(X_train_scaled, y_train)
y_pred = log_model.predict(X_test_scaled)

print(classification_report(y_test, y_pred))
print("AUC:", roc_auc_score(y_test,
log_model.predict_proba(X_test_scaled)[:,1]))
```

```
# Train logistic regression and save model

from sklearn.linear_model import LogisticRegression

import joblib

model = LogisticRegression(max_iter=1000, class_weight='balanced', random_state=42)

model.fit(X_train_scaled, y_train)

joblib.dump(model, 'fraud_model/logistic_regression_fraud.pkl')
```

```
# EDA_and_Tuning.ipynb
# Fraud Detection - Enhanced EDA, Model Tuning, and Comparison
# Setup And Loading The Data

import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split, GridSearchCV
from sklearn.preprocessing import StandardScaler
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import classification_report, confusion_matrix, roc_auc_score, average_precision_score
from xgboost import XGBClassifier
import joblib

# S3 dataset path
s3_uri = "s3://bsukanisagemaker-afresh/aai-540-labs/lab-2-1-fresh/creditcard.csv"
print(f"Loading dataset from {s3_uri} ...")
data = pd.read_csv(s3_uri)
print("Dataset loaded successfully!")
print(data.shape)
data.head()


# Exploratory Data Analysis (EDA)

print("\n--- Basic Info ---")
print(data.info())
print("\n--- Missing Values ---")
print(data.isnull().sum().sum())
```

```
# Class distribution
plt.figure(figsize=(6,4))
sns.countplot(x='Class', data=data)
plt.title('Class Distribution (0 = Non-Fraud, 1 = Fraud)')
plt.show()

# Distributing the Amount
plt.figure(figsize=(6,4))
sns.histplot(data['Amount'], bins=50)
plt.title('Transaction Amount Distribution')
plt.show()

# Correlation heatmap
plt.figure(figsize=(10,8))
sns.heatmap(data.corr(), cmap='coolwarm', center=0)
plt.title('Correlation Heatmap of Features')
plt.show()

# Summarizing insight
fraud = data[data['Class'] == 1]
non_fraud = data[data['Class'] == 0]
print(f"Fraud Mean Amount: {fraud['Amount'].mean():.2f}")
print(f"Non-Fraud Mean Amount: {non_fraud['Amount'].mean():.2f}")

# Train; Test Split and Scaling

X = data.drop(columns=['Class'])
y = data['Class']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, stratify=y, random_state=42)

scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)
print("Data split and scaled successfully!")

#  Baseline Model-Logistic Regression

log_model = LogisticRegression(max_iter=1000, class_weight='balanced', random_state=42)
log_model.fit(X_train_scaled, y_train)

# Predicting and evaluating
y_pred = log_model.predict(X_test_scaled)
```

```
print("\n--- Logistic Regression Evaluation ---")
print(classification_report(y_test, y_pred))
print("AUC:", roc_auc_score(y_test, log_model.predict_proba(X_test_scaled)[:,1]))
print("PR-AUC:", average_precision_score(y_test, log_model.predict_proba(X_test_scaled)[:,1
]))

# Feature importance from coefficients
importance = log_model.coef_[0]
feature_names = X.columns
sorted_idx = np.argsort(np.abs(importance))[::-1][:10]
print("\nTop 10 Important Features:")
for i in sorted_idx:
    print(f"{feature_names[i]}: {importance[i]:.4f}")


#  Hyperparameter Tuning (GridSearchCV)

param_grid = {
    'C': [0.01, 0.1, 1, 10],
    'solver': ['liblinear', 'lbfgs']
}

print("\nRunning GridSearchCV (optimize for Recall)...")
grid = GridSearchCV(LogisticRegression(max_iter=1000, class_weight='balanced', random_s
tate=42),
                    param_grid, scoring='recall', cv=3, n_jobs=-1, verbose=1)
grid.fit(X_train_scaled, y_train)

print("Best Params:", grid.best_params_)
print("Best Recall:", grid.best_score_)

best_log_model = grid.best_estimator_

# Evaluating the tuned model
y_pred_best = best_log_model.predict(X_test_scaled)
print("\n--- Tuned Logistic Regression Evaluation ---")
print(classification_report(y_test, y_pred_best))


#  XGBoost Classifier (Comparative Model)

xgb_model = XGBClassifier(
    scale_pos_weight=len(y[y==0]) / len(y[y==1]),
    eval_metric='aucpr',
    random_state=42
)
xgb_model.fit(X_train_scaled, y_train)

# Evaluating the  XGBoost model
y_pred_xgb = xgb_model.predict(X_test_scaled)
print("\n--- XGBoost Evaluation ---")
print(classification_report(y_test, y_pred_xgb))
print("AUC:", roc_auc_score(y_test, xgb_model.predict_proba(X_test_scaled)[:,1]))
print("PR-AUC:", average_precision_score(y_test, xgb_model.predict_proba(X_test_scaled)[:,1]))

# Saving Artifacts

import os
os.makedirs('fraud_model', exist_ok=True)
joblib.dump(best_log_model, 'fraud_model/logistic_regression_best.pkl')
joblib.dump(xgb_model, 'fraud_model/xgboost_model.pkl')

print("Models saved to ./fraud_model/")

# Further Actions Or Monitoring (to be extended)

print("\nNext steps: Integrate with SageMaker Model Registry, CloudWatch monitoring, and CI/CD pipeline.")
```