1. Project Overview

Fraud detection in financial transactions is a critical challenge due to the highly imbalanced nature of fraud vs. non-fraud cases. Dataset: Credit Card Fraud Detection dataset (Kaggle) with 284,807 transactions and 492 fraud cases (~0.172%). Objective: Build a system that flags potential fraud with high accuracy, focusing on high recall.

2. Problem Statement

We must design and deploy a machine learning pipeline to preprocess the imbalanced dataset, train and compare models, select appropriate metrics, deploy the best model as an endpoint in AWS SageMaker, and monitor its behavior. Expected solution: an XGBoost classifier tuned for PR-AUC and Recall, compared against Logistic Regression baseline.

3. Metrics

- Primary: PR-AUC (best for rare events)
- Secondary: ROC-AUC, Recall, Precision, F1-score
- Confusion Matrix for interpretability
- Benchmark: Logistic Regression (ROC-AUC ~0.90, PR-AUC ~0.60). Final model must improve these.

4. Data Exploration & Visualization

- Check class distribution
- Summary statistics for Amount, Time
- Visualizations: Fraud vs Non-fraud counts, histograms of Amount, correlation heatmap, boxplot of Time vs fraud probability
- Key abnormalities: extreme imbalance, PCA features, scaling needed for Amount and Time.

5. Algorithms & Techniques

- Baseline: Logistic Regression with class_weight='balanced'
- Advanced: XGBoost with hyperparameter tuning
- Optional: Random Forest for comparison
- Deployment: SageMaker endpoint with inference.py
- Monitoring: Debugger, Profiler, Model Monitor.

6. Methodology

- Data preprocessing: Stratified split (80/10/10), RobustScaler on Amount and Time
- Implementation: train.py (baseline), hpo.py (XGBoost HPO), evaluation.ipynb, deploy_model.ipynb

- Refinement: Compare Logistic Regression → XGBoost default → XGBoost HPO tuned, document improvements.

7. Results

- Report ROC-AUC, PR-AUC, Recall, Precision, F1
- Show ROC and PR curves, confusion matrix
- Threshold tuning plot to justify cut-off
- Validation: hold-out test, robustness from stratified split
- Justification: XGBoost tuned significantly outperforms Logistic Regression baseline.

8. Deployment

Final model deployed as a SageMaker endpoint. Inference script ensures correct input/output formatting. Endpoint tested with sample JSON requests. Auto-scaling enabled (min=1, max=3).

9. Monitoring

Debugger and Profiler enabled during training (profiler_report.pdf generated). Model Monitor captures inference data and checks for drift, with CloudWatch alerts configured.

10. Documentation

- proposal.pdf – project origin
- report.pdf – this document with results/plots
- README.md – setup & reproduction instructions
- Notebooks – EDA, training, evaluation, deployment
- Scripts – train.py, hpo.py, inference.py, utils.py
- Supporting: screenshots, profiler report.

Submission Checklist

- proposal.pdf
- report.pdf (this document with results and plots)
- README.md
- Python scripts
- Jupyter notebooks
- Screenshots and profiler_report.pdf