ATM Fraud Detection System - Comprehensive Project Brief

Executive Summary

An Australian banking institution is experiencing significant financial losses and reputational damage due to fraudulent ATM transactions. PredCatch Analytics has been engaged to develop a real-time fraud detection system that can identify and prevent fraudulent transactions before they're completed.

Business Objective

Primary Goal: Build a predictive model capable of detecting fraudulent ATM transactions in real-time with high accuracy while minimizing false positives that would inconvenience legitimate customers.

Success Criteria:

- Maximize fraud detection rate (recall for fraudulent transactions)
- Minimize false alarms that block legitimate customers
- Enable real-time decision-making (low latency)
- Provide explainable predictions for regulatory compliance

Dataset Overview

Target Variable

- **Target**: Binary classification
    - 1 = Fraudulent transaction
    - 0 = Legitimate transaction

Key Features

The dataset includes masked transaction variables plus proprietary engineered features:

- **geo_scores**: Location-based risk indicators
- **Lambda_wts**: Proprietary risk weighting index
- **Qset_tats**: Network turnaround time metrics
- **instance_scores**: Vulnerability qualification scores
- **Additional masked variables**: Transaction-specific attributes (anonymized for privacy)

Critical Challenge: Class Imbalance

The dataset exhibits severe class imbalance with fraudulent transactions representing a small minority of all cases. This requires specialized modeling techniques and evaluation approaches.

Evaluation Framework

Primary Metrics (Ranked by Importance)

1. Recall (Sensitivity) for Fraud Class - MOST CRITICAL

Recall = True Positives / (True Positives + False Negatives)

- **Target**: ≥ 85-90%
- **Rationale**: Missing fraud is costly; we want to catch most fraudulent transactions
- **Business Impact**: Each missed fraud could mean $1,000s in losses

2. Precision for Fraud Class

Precision = True Positives / (True Positives + False Positives)

- **Target**: ≥ 30-40% (acceptable given extreme imbalance)
- **Rationale**: Too many false positives frustrate legitimate customers
- **Business Impact**: False positives lead to customer dissatisfaction and call center costs

3. F2-Score (Weighted F-Score favoring Recall)

F2 = 5 × (Precision × Recall) / (4 × Precision + Recall)

- **Target**: ≥ 0.70
- **Rationale**: Balances precision and recall while prioritizing fraud detection
- **Why F2 over F1**: Catching fraud is more important than avoiding false alarms

4. Area Under Precision-Recall Curve (AUPRC)

- **Target**: ≥ 0.50
- **Rationale**: More informative than ROC-AUC for imbalanced datasets
- **Use Case**: Evaluates model across all threshold settings

5. ROC-AUC Score

- **Target**: ≥ 0.85
- **Rationale**: Overall discrimination ability between classes
- **Limitation**: Can be misleading with severe imbalance

Secondary Metrics

6. Cost-Weighted Accuracy

Define costs:

- False Negative (missed fraud): $500 average loss
- False Positive (blocked legitimate): $5 customer service cost
- Calculate total cost and compare models

7. Detection Rate at Fixed False Positive Rate

- **Metric**: Recall when FPR = 1%
- **Business Use**: Real-world operating point constraint

Model Selection & Comparison

Recommended Models (Ranked)

1. XGBoost (Recommended)

Strengths:

- Excellent handling of imbalanced data with scale_pos_weight parameter
- Built-in feature importance for explainability
- Fast training and prediction (real-time capable)
- Robust to outliers and missing values
- Handles non-linear relationships

Configuration:

scale_pos_weight = count(negative) / count(positive)

max_depth = 6-8

learning_rate = 0.01-0.1

subsample = 0.8

Expected Performance: F2 ≥ 0.75, Recall ≥ 90%

2. LightGBM

Strengths:

- Faster training than XGBoost on large datasets
- Lower memory usage
- Excellent categorical feature handling
- Similar performance to XGBoost

When to use: Dataset > 100K rows, need faster training

3. Random Forest with Class Balancing

Strengths:

- Robust and interpretable
- Good baseline model
- Less hyperparameter tuning needed

Weaknesses:

- Slower prediction time
- Typically lower performance than boosting

4. Logistic Regression (Baseline)

Strengths:

- Fast, interpretable
- Good baseline to beat
- Regulatory-friendly (explainable coefficients)

Weaknesses:

- Assumes linear relationships
- Lower performance on complex patterns

Advanced Ensemble Approach

Stacking/Blending: Combine XGBoost + LightGBM + Random Forest

- Meta-learner: Logistic Regression
- Expected lift: +2-5% in recall

Handling Class Imbalance - Techniques

1. Algorithm-Level

- Use class_weight='balanced' or scale_pos_weight

2. Resampling Techniques

- **SMOTE** (Synthetic Minority Oversampling): Generate synthetic fraud examples

3. Threshold Optimization

- Default threshold = 0.5 may not be optimal
- Optimize threshold to maximize F2-score on validation set
- Typical optimal range: 0.2-0.4 for fraud

Implementation Roadmap

Phase 1: Model Development

1. **EDA & Feature Engineering**
    - Analyze fraud patterns
    - Create interaction features
    - Handle missing values
2. **Baseline Models**
    - Logistic Regression
    - Random Forest
3. **Advanced Models**
    - XGBoost with hyperparameter tuning

    - LightGBM comparison
4. **Ensemble & Threshold Optimization**

Phase 2: Model Validation

- K-fold cross-validation with stratification

Phase 3: Deployment Preparation

- Model serialization
- API development (FastAPI)
- Monitoring dashboard

Phase 4: Client Presentation

- Business impact analysis
- Live demo with frontend

Key Features of the Dashboard

1. Real-Time Transaction Testing

- Input transaction features for instant fraud prediction
- Visual risk scoring with color-coded alerts
- Feature importance breakdown

2. Model Performance Visualization

- Comparative bar charts across 4 algorithms
- Confusion matrix

3. Key Metrics Display

- Fraud detection rate (Recall)
- Precision
- F2-Score
- Average response time

Next Steps for Implementation

Technical Requirements

1. **Backend API** (Streamlit/FastAPI/Flask)
    - Model serving endpoint
    - Real-time prediction pipeline
    - Feature preprocessing