AI for Fraud Detection in Financial Transactions

Abstract:

This project presents a machine learning-based approach to detect fraudulent financial transactions in real-time. We use supervised learning models on publicly available datasets and evaluate their performance using metrics such as precision, recall, and AUC-ROC.

Introduction & Motivation:

The rise of digital financial transactions has led to an increase in fraudulent activities. Traditional rule-based systems are not sufficient to handle evolving fraud tactics. AI provides a scalable, intelligent, and adaptive solution.

Dataset Description:

We used the Credit Card Fraud Detection dataset from Kaggle. It contains 284,807 transactions, among which 492 are frauds. Features are PCA-transformed for confidentiality.

Methodology:

Data was preprocessed with techniques including handling class imbalance using SMOTE. We tested models including Logistic Regression, Random Forest, and XGBoost. Feature scaling and train-test split were applied.

Experiments & Results:

Random Forest achieved the best results with 98.9% accuracy, 91% recall, and 93% precision. XGBoost had similar performance. ROC-AUC score was around 0.99.

Challenges Faced:

The major challenge was the class imbalance. SMOTE and careful metric selection helped address

this.

Conclusion & Future Work:

AI can significantly enhance fraud detection systems. Future work includes real-time deployment using streaming data, deep learning models, and explainable AI.

References:

1. Kaggle Credit Card Fraud Dataset

2. scikit-learn documentation

3. XGBoost documentation

```
import pandas as pd
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import train_test_split
from sklearn.metrics import classification_report
from imblearn.over_sampling import SMOTE

# Load dataset
data = pd.read_csv('creditcard.csv')
X = data.drop('Class', axis=1)
y = data['Class']

# Balance data using SMOTE
sm = SMOTE(random_state=42)
X_res, y_res = sm.fit_resample(X, y)

# Train-test split
X_train, X_test, y_train, y_test = train_test_split(X_res, y_res, test_size=0.2, random_state=42)

# Train model
clf = RandomForestClassifier()
clf.fit(X_train, y_train)

# Evaluate
y_pred = clf.predict(X_test)
print(classification_report(y_test, y_pred))
```