**1. Introduction**

This project develops a machine learning system to detect fraudulent credit card transactions. The system comprises a Logistic Regression model trained on transaction data and a Streamlit front-end application for real-time predictions. This report documents the dataset, preprocessing steps, model performance, and application workflow.

**2. Dataset and Preprocessing**

**Dataset:**

* **Source**: creditcard.csv (contains anonymized credit card transactions).
* **Features**:
  + **V1-V28**: PCA-transformed numerical features (anonymized for privacy).
  + **Amount**: Transaction amount.
  + **Class**: Binary label (0 = legitimate, 1 = fraudulent).

**Preprocessing Steps:**

* **Subsampling**: 10,000 legitimate transactions were randomly selected to reduce computational load while preserving fraud patterns.
* **Feature Removal**: The Time column was dropped due to irrelevance.
* **Data Splitting**: An 80-20 stratified split ensured balanced class distribution in training/testing sets.
* **Scaling**: StandardScaler standardized all features to normalize numerical ranges.
* **Class Balancing**: SMOTE oversampled the minority class (fraud) to address imbalance.

**3. Model Training and Performance**

**Algorithm:** Logistic Regression

**Key Metrics:**

* Test Accuracy: 97%
* F1 Score: 0.7342 (balanced precision and recall).
* Precision: 0.6259 (accuracy of fraud predictions).
* Recall: 0.8878 (ability to detect fraud).
* ROC-AUC: 0.9723 (strong separability between classes).

**Model Interpretation:**

With exceptional recall (88.8%), the model detects nearly 90% of fraud cases, prioritizing fraud capture over false alarms (precision: 62.6%). Its near-perfect ROC-AUC (0.972) confirms strong ability to distinguish fraud from legitimate transactions, supported by a balanced F1 score (0.734).

**4. Front-End Application**

**Interface:**

* **Inputs**: Users provide V1-V28 (PCA components) and Amount via sliders/number inputs.
* **Workflow**:
  1. Inputs are scaled using the saved StandardScaler.
  2. The model calculates fraud probability.
  3. Results display:
* **Fraud Probability**: Percentage likelihood of fraud.
* **Alert**: Red banner for fraud (>70% probability), green for legitimate transactions.

**5. Challenges and Solutions**

* **Class Imbalance**:
  + **Issue**: Fraudulent transactions comprised <1% of the dataset.
  + **Solution**: SMOTE generated synthetic fraud samples for balanced training.
* **Feature Scaling**:
  + **Issue**: Initial scaling reduced model performance by flattening transaction amount significance.
  + **Solution**: StandardScaler was retained after testing showed improved convergence.

**6. Conclusion**

The system achieves robust performance with an F1 score of 0.6571 and ROC-AUC of 0.8963, effectively balancing fraud detection and false alarms. Key strengths include:

* SMOTE for handling class imbalance.
* Streamlit for user-friendly predictions.

Areas for Improvement:

* Replace PCA inputs with raw transaction details (e.g., location, merchant) for interpretability.
* Experiment with ensemble models (e.g., Random Forest) to boost precision.

**Finalized Deliverables:**

* **Trained model (fraud\_model.pkl).**
* **Scalable front-end application (app.py).**
* **Documentation**