**Credit Card Fraud Detection Using Machine Learning**

**1. Introduction**

In today's digital age, credit card transactions have become an essential part of financial systems, providing ease and convenience to consumers worldwide. However, this increase in digital transactions has also led to a significant rise in fraudulent activities, posing a major threat to both financial institutions and customers.

Credit card fraud can result in substantial financial losses and can severely affect the trust between consumers and service providers. Therefore, timely and accurate detection of fraudulent transactions is crucial to minimize risks and protect user data.

This project aims to develop a **Credit Card Fraud Detection system** using **Machine Learning algorithms** such as Logistic Regression, Decision Tree, and Random Forest. Due to the imbalanced nature of fraud datasets—where genuine transactions vastly outnumber fraudulent ones—**SMOTE (Synthetic Minority Over-sampling Technique)** is employed to balance the class distribution.

To provide a user-friendly interface, the project is deployed using **Streamlit**, enabling real-time predictions and visual insights into the model's performance.

**2. Objectives**

* To detect fraudulent credit card transactions using machine learning.
* To handle class imbalance using SMOTE.
* To compare the performance of different machine learning models.
* To deploy the model with a user-friendly interface using Streamlit.

**3. Problem Statement**

Credit card fraud detection is a challenging problem due to the rarity of fraudulent transactions compared to legitimate ones. Traditional rule-based systems often fail to adapt to new fraud patterns. The goal is to use machine learning techniques to create an adaptive and accurate fraud detection system that minimizes false negatives and false positives.

**4. Tools and Technologies Used**

* **Programming Language:** Python
* **Libraries:** Pandas, NumPy, scikit-learn, imbalanced-learn, Matplotlib, Seaborn
* **Models:** Logistic Regression, Decision Tree, Random Forest
* **Technique:** SMOTE (Synthetic Minority Over-sampling Technique)
* **Deployment:** Streamlit
* **IDE:** Jupyter Notebook / VS Code

**5. Methodology**

1. **Data Collection:**
   * The dataset used contains anonymized credit card transactions.
2. **Data Preprocessing:**
   * Handled missing values and scaled features.
   * Checked for class imbalance.
3. **SMOTE Application:**
   * Applied SMOTE to oversample the minority class.
4. **Model Building:**
   * Trained Logistic Regression, Decision Tree, and Random Forest models.
5. **Model Evaluation:**
   * Evaluated using Accuracy, Precision, Recall, F1-Score, and Confusion Matrix.
6. **Deployment:**
   * Developed a Streamlit interface to input transaction data and view predictions.

**6. Results and Evaluation**

After applying SMOTE, all three models showed improved performance. Among them:

* **Random Forest** achieved the highest accuracy and recall, making it the best model for fraud detection in this project.
* **Decision Tree** provided good interpretability.
* **Logistic Regression** performed well but was slightly less accurate than Random Forest.

Evaluation Metrics (sample results):

* **Random Forest:** Accuracy - 98.5%, Recall - 96%
* **Decision Tree:** Accuracy - 97%, Recall - 93%
* **Logistic Regression:** Accuracy - 95%, Recall - 90%

**7. Conclusion**

The project successfully demonstrates the use of machine learning for credit card fraud detection. By handling class imbalance with SMOTE and evaluating multiple models, we were able to identify Random Forest as the most effective algorithm for this task. The Streamlit deployment ensures accessibility and usability for real-time predictions.

**8. Future Work**

* Integrate deep learning models for improved accuracy.
* Implement real-time data streaming.
* Incorporate feedback loops for continuous learning.
* Add user authentication and logging in the Streamlit app for security.

**9. References**

* Credit Card Fraud Detection Dataset (Kaggle)
* scikit-learn documentation
* imbalanced-learn documentation
* Streamlit official docs

*End of Report*