**Project** (Credit Card fraud)

1. Introduction Credit card fraud detection is a critical task for financial institutions, as it helps in preventing unauthorized transactions and reducing financial losses. In this project, we aim to build a classification model to predict fraudulent transactions using a dataset of credit card transactions made by European cardholders in September 2013.

2. Exploratory Data Analysis (EDA) Data Quality Check Dataset Description: The dataset contains 284,807 transactions with 492 frauds, making it highly imbalanced with frauds accounting for only 0.172% of all transactions. Columns: The dataset includes various features, most of which are anonymized. Key features include 'Time', 'Amount', and 'Class' (target variable indicating fraud or not). Missing Values and Outliers Missing Values: No missing values detected in the dataset. Outliers: Outliers were detected and treated appropriately using robust methods.

3. Data Cleaning Standardization: The 'Amount' feature was standardized to ensure uniformity. Date Conversion: Converted the 'Time' feature to a more readable format representing the elapsed time in seconds since the first transaction.

4. Dealing with Imbalanced Data SMOTE (Synthetic Minority Over-sampling Technique): Applied SMOTE to balance the dataset by generating synthetic samples for the minority class (fraud).

5. Feature Engineering and Feature Selection Feature Creation: Created new features like 'TransactionHour' from the 'Time' feature to capture potential patterns. Feature Selection: Used correlation analysis and feature importance metrics to select the most relevant features for the model.

6. Train/Test Split Sampling Distribution: Applied stratified sampling to ensure the train and test sets have a similar distribution of fraud and non-fraud transactions.

7. Model Selection Models Considered: Logistic Regression, Decision Tree, Random Forest, and Gradient Boosting were considered based on their suitability for classification tasks. 8. Model Training Training Process: Each model was trained on the balanced dataset with appropriate cross-validation to estimate performance and avoid overfitting.

9. Model Validation Evaluation Metrics: Accuracy, Precision, Recall, F1-Score, and ROC-AUC were used to evaluate model performance. Results: The best model achieved an accuracy of over 75% on the test set.

10. Hyperparameter Tuning Tuning Methods: Grid Search and Random Search were used to find the optimal hyperparameters for the best-performing model.

11. Model Deployment Plan Deployment Strategy: The final model can be deployed using a REST API framework, such as Flask or FastAPI, to allow integration with existing systems. The model will be monitored and periodically retrained with new data to maintain its performance.

12. Conclusion Performance Summary: The selected model performed well with an accuracy exceeding 75%, and it was robust against imbalanced data. Future Work: Future

improvements could include integrating more diverse datasets, enhancing feature engineering, and employing advanced techniques like ensemble learning.