**SRI SHANMUGHA COLLEGE OF ENGINEERING AND TECHNOLOGY**

**TEAM MEMBER: TS.JANANI**

***PHASE 5 SUBMISSION*:**

CREDIT CARD FRAUD DETECTION

**ABSTRACT:**

Credit card fraud is a major problem for financial institutions and consumers alike. Machine learning can be used to develop effective credit card fraud detection systems that can identify fraudulent transactions with high accuracy. This abstract will discuss the different modules involved in building a credit card fraud detection system using machine learning.

**Conclusion**:

Machine learning can be used to develop effective credit card fraud detection systems that can identify fraudulent transactions with high accuracy. The modules involved in building a credit card fraud detection system using machine learning include data collection and preparation, feature engineering, model selection and training, model evaluation, and model deployment

* Class imbalance: Credit card fraud datasets are typically imbalanced, meaning that there are many more non-fraudulent transactions than fraudulent transactions. This can make it difficult to train a machine learning model that can accurately identify fraudulent transactions. There are a variety of techniques that can be used to address class imbalance, such as oversampling the minority class or under sampling the majority class.
* Privacy: Credit card fraud detection systems often collect sensitive data about consumers, such as transaction amounts and locations. It is important to design and implement these systems in a way that protects the privacy of consumers. This may involve using techniques such as data anonymization and encryption.
* Explainability: It is important to be able to explain how a machine learning model makes its predictions. This is especially important for credit card fraud detection systems, as consumers need to be able to understand why a transaction was flagged as fraudulent. There are a variety of techniques that can be used to explain machine learning models, such as LIME and SHAP.

**Algorithm: Credit Card Fraud Detection**

**STEP 1: IMPORT LIBRARIES**

**Step 2: Load the Data**

* Load the credit card transaction data from a CSV file.

**Step 3: Handle Missing Values**

* Identify columns with missing values (NaN) in the dataset.
* Fill the missing values with the mean of the respective columns using SimpleImputer from scikit-learn.
* Convert the imputed data back into a pandas DataFrame.

**Step 3: Split the Data**

* Split the data into features (X) and the target variable (y).
* The features represent various attributes of transactions, and the target variable is binary (1 for fraud, 0 for non-fraud).
* Use train\_test\_split from scikit-learn to split the data into training and testing sets (typically 80% for training and 20% for testing).

**Step 4: Initialize and Train the Random Forest Classifier**

* Initialize a Random Forest Classifier, a machine learning algorithm suitable for classification tasks, with default parameters.
* Train the classifier using the training data (X\_train, y\_train).
* The classifier learns to distinguish between fraudulent and non-fraudulent transactions based on the provided features.

**Step 5: Handle NaN Values in Test Data**

* Apply the same SimpleImputer used for training data to fill missing values in the test data.
* Ensure that the test data (X\_test\_imputed) is a pandas DataFrame with appropriate column names.

**Step 6: Make Predictions**

* Use the trained Random Forest Classifier to make predictions on the imputed test data.
* The classifier predicts whether each transaction in the test set is fraudulent or non-fraudulent.
* Compare the predicted labels (y\_pred) with the actual labels (y\_test) to assess the model's performance.
* Calculate metrics such as precision, recall, and F1-score to quantify the model's accuracy in detecting fraud.
* Generate a confusion matrix to visualize true positive, true negative, false positive, and false negative predictions.
* Interpret the metrics and confusion matrix to understand the model's strengths and areas for improvement.
* Analyze the classification report, including precision, recall, and F1-score, to understand the model's accuracy for both fraudulent and non-fraudulent transactions.
* Review the confusion matrix to identify how well the model is performing in terms of true positives, true negatives, false positives, and false negatives.
* Use the results to make decisions about deploying the model for real-world credit card fraud detection, considering the trade-off between false positives and false negatives based on the application's requirements.
* This algorithm outlines the key steps involved in credit card fraud detection using a Random Forest Classifier and provides a structured approach to handling missing values, training the model, and evaluating its performance.

### Project Structure:

* **credit\_card\_fraud\_detection.ipynb**: GOGLE Notebook containing the code for data preprocessing, model training, and evaluation.
* **README.md**: Documentation explaining how to run the code, dependencies, dataset source, and pr oject details.

***# Credit Card Fraud Detection:This project aims to detect fraudulent credit card transactions using machine learning techniques. The dataset used for this project can be found at [Dataset***

**Dataset Link:**[**https://www.kaggle.com/datasets/mlg-ulb/creditcardfraud**](https://www.kaggle.com/datasets/mlg-ulb/creditcardfraud)

***## Dependencies***

***- Python 3.x***

***- Jupyter Notebook***

***- pandas***

***- scikit-learn***

***## How to Run the Code***

***1. Clone this repository:***

***```bash***

***git clone:***

***# Data Preprocessing***

***import pandas as pd***

***from sklearn.model\_selection import train\_test\_split***

***from sklearn.ensemble import RandomForestClassifier***

***from sklearn.metrics import classification\_report***

***# Load the dataset***

***df = pd.read\_csv('creditcard.csv')***

***# Data preprocessing steps...***

***# Split the data into features (X) and target variable (y)***

***X = df.drop(columns=['Class']) # Features***

***y = df['Class'] # Target variable***

***# Split the data into training and testing sets***

***X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=42)***

***# Train the Random Forest model***

***rf\_classifier = RandomForestClassifier()***

***rf\_classifier.fit(X\_train, y\_train)***

***# Make predictions***

***y\_pred\_rf = rf\_classifier.predict(X\_test)***

***# Evaluate the model***

***print(classification\_report(y\_test, y\_pred\_rf))***

