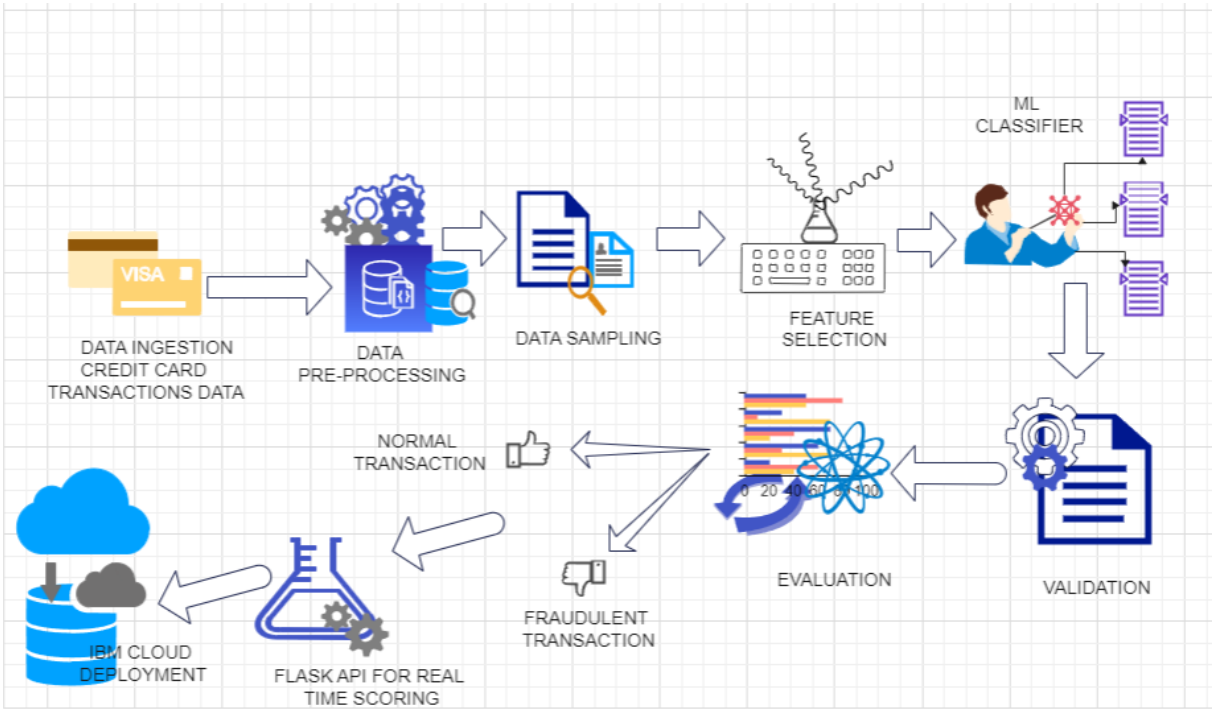
**Project Design Phase-I**
**Solution Architecture**

| | |
|---|---|
| **Date** | **07-11-2023** |
| **Team ID** | **Team-593025** |
| **Project Name** | **Project - Online Payments Fraud Detection Using ML** |
| **Maximum Marks** | **4 Marks** |

**Solution Architecture for Credit/Debit Card Fraud Detection**




**Data Ingestion:**

Data from credit/debit card transactions is collected from various sources, including databases, real-time streams, and external data providers.
Data connectors and APIs specific to the data sources are used to fetch the data.
The data is stored in a centralized data storage system, such as a data lake or data warehouse, for further processing and analysis.

**Data Preprocessing:**

Data preprocessing includes cleaning, normalization, handling of missing values, and data transformation.
Feature engineering is performed to create new relevant features from the raw data.
Python libraries like Pandas, NumPy, and Scikit-learn are used for data manipulation and preprocessing.

**Data Sampling:**

Data sampling techniques are applied to address class imbalance and create a balanced dataset.
Strategies such as oversampling, undersampling, or Synthetic Minority Over-sampling Technique (SMOTE) are used.

**Feature Selection:**

Feature selection methods are employed to identify the most relevant features for fraud detection.
Techniques like Recursive Feature Elimination (RFE), feature importance from tree-based models, and correlation analysis are applied.

**Machine Learning Classifier:**

Various classification algorithms, including Decision Trees, Random Forest, SVM, Extra Tree Classifier, and XGBoost, are used to train models on the preprocessed and feature-selected data.

**Model Validation:**

The trained models are validated using techniques such as k-fold cross-validation to assess their generalization performance.
The data is divided into training, validation, and testing sets to ensure robust model evaluation.

**Model Evaluation:**

Model evaluation metrics, including precision, recall, F1-score, and ROC AUC, are used to assess the model's performance in detecting fraudulent transactions.
Confusion matrices and receiver operating characteristic (ROC) curves are generated to visualize model performance.

**Model Serialization and Management:**

The best-performing model is serialized and saved in a format such as ".pkl" using Python's joblib or pickle.

Model management tools are employed to version, track, and monitor model performance.

**Flask API for Real-time Scoring:**

A Flask-based RESTful API is created to expose the trained model for real-time predictions.
Clients, including e-commerce platforms or payment gateways, can send transaction data to the API for fraud detection.

**IBM Cloud Deployment:**

The entire system, including the Flask application and the serialized model, is deployed on the IBM Cloud platform.
IBM Cloud provides scalability, infrastructure management, and hosting services.