**Problem Statement: To predict fraudulent credit card transactions with the help of machine learning models.**

**Steps:**

1. Load data.
   1. The data set includes credit card transactions made by European cardholders over a period of two days in September 2013. Out of a total of**2,84,807 transactions, 492 were fraudulent.**
2. Data understanding and EDA
   1. There are no **"Null"** values, so we don't have to work on ways to replace values.
   2. **Data Imbalance:** This data set is highly unbalanced, **with the positive class (frauds) accounting for just 0.172% of the total transactions.** If we use this data frame as the base for our predictive models and analysis, we might get a lot of errors and our algorithms will probably overfit since it will assume that most transactions are not fraud. We will have to balance the dataset using one of the balancing techniques (SMOTE, ADAYN, over sampling)
   3. **PCA Transformation:** The data set has been modified with Principal Component Analysis (PCA) to maintain confidentiality. Apart from ‘time’ and ‘amount’, all the other features**(V1, V2, V3, up to V28)** are the principal components obtained using PCA. The feature 'time' contains the seconds elapsed between the first transaction in the data set and the subsequent transactions. The feature 'amount' is the transaction amount. The **feature 'class' represents class labelling**, and it takes the value 1 in cases of fraud and 0 in others.
   4. **Distributions:** By looking at the distributions, we get an idea that the features are skewed. Skewness may affect model assumptions or may impair the interpretation of feature importance. We need to mitigate the skewness. **We can use PowerTransformer (method='yeo-johnson') package present in the preprocessing library provided by sklearn to make distribution more gaussian.**
3. Outlier Removal – to be done using one of the following methods
   1. Interquartile Range
   2. Boxplots
4. Splitting the data set: In a 80:20 ratio (train / test split) - to check the performance of your models with unseen data. Here, for validation, we can use the k-fold cross-validation method. You need to choose an appropriate k value so that the minority class is correctly represented in the test folds.
5. **Model-Building/Hyperparameter Tuning:** We will be trying different modelling techniques and will try to tune the hyper parameters until we reach the desired level of performance. Some of the modelling techniques are:
   1. Logistic Regression
   2. Decision Trees
   3. Random Forest
   4. XG Boost
   5. **For hyperparameter tuning, random and grid search**are the two methods available in**scikit-learn**in the form**of RandomiszedSearchCV and GridSearchCV,**respectively**.**
6. Model Evaluation : Few tests the evaluate the model will be
   1. Accuracy
   2. Precision
   3. Recall
   4. Confusion matrix
   5. F1 Score
   6. AUC – ROC Score

Because the ROC curve is measured at all thresholds, the best threshold would be one at which the **TPR is high and FPR is low,** **i.e., mis classifications are low.**

1. Cost - Benefit Analysis
   1. From your confusion matrix, find total fraud predictions that the model has made (TP + FP).
   2. For all the predictions, the bank has to call to verify whether the prediction was right or not. (Take ₹ 10/call for the bank)
   3. The savings will be the total amount of correct predictions made: TP x Cost of each transaction which is correctly predicted.
   4. The loss will be the total amount of incorrect predictions made: FN x Cost of each transaction which is incorrectly predicted.
   5. So the total savings= (TP x cost of each transaction (correct predictions) -[ (TP+FP) x 10 + FN x Cost of all transactions (incorrect predictions)] )