**Credit Card Fraud Detection Case Study**

**Introduction**

Finding credit card fraud is important. There's more credit card fraud happening, and we need better ways to catch it. We'll use machine learning to help us find fraud because they're good at handling lots of data and identifying tricky patterns.

**Aim:**

The aim of this study is to explore the effectiveness of machine learning algorithms in detecting credit card fraud. We want to use a machine learning model to see whether it helps us at identifying fraudulent transactions.

**Dataset:**

We have a dataset containing information about credit card transactions, including features like transaction time, amount, and various anonymized features (V1-V28). These are the principal components obtained with PCA. The exact meaning of these features is not provided due to privacy reasons but they represent transformed features from the original data.The dataset also includes a target variable, "Class", indicating whether a transaction is fraudulent (Class = 1) or not (Class = 0). This dataset allows us to train machine learning models to learn patterns associated with fraudulent transactions and make predictions on new, unseen data.

**Dataset Link :** [**https://www.kaggle.com/code/gpreda/credit-card-fraud-detection-predictive-models/input**](https://www.kaggle.com/code/gpreda/credit-card-fraud-detection-predictive-models/input)

**Code:**

import pandas as pd

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

from sklearn.ensemble import RandomForestClassifier

from sklearn.metrics import accuracy\_score, classification\_report, confusion\_matrix

from sklearn.impute import SimpleImputer

data = pd.read\_csv("creditcard.csv")

data.dropna(subset=['Class'], inplace=True)

X = data.drop(columns=['Class'])

y = data['Class']

# 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)

# Impute missing values with the mean for features

imputer = SimpleImputer(strategy='mean')

X\_train\_imputed = imputer.fit\_transform(X\_train)

X\_test\_imputed = imputer.transform(X\_test)

# Initialize the Random Forest classifier

clf = RandomForestClassifier(n\_estimators=100, random\_state=42)

# Train the classifier

clf.fit(X\_train\_imputed, y\_train)

y\_pred = clf.predict(X\_test\_imputed)

accuracy = accuracy\_score(y\_test, y\_pred)

print("\nAccuracy:", accuracy)

# Print classification report

print("\nClassification Report:")

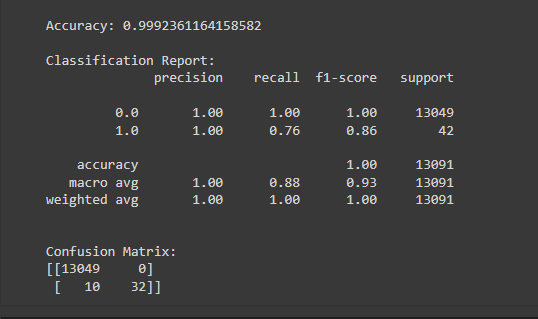
print(classification\_report(y\_test, y\_pred))

# Print confusion matrix

print("\nConfusion Matrix:")

print(confusion\_matrix(y\_test, y\_pred))

**Output:**

****

**Result Interpretation**

The results for the credit card fraud detection model look effecient. The accuracy is of approximately 99.92%, which states that the model demonstrates a good performance in accurately classifying transactions.

Looking at the classification report, we see that the precision and recall for detecting fraudulent transactions (class 1) are both perfect (1.00), indicating that the model correctly identifies nearly all instances of fraud while minimizing false positives.

The F1-score, which balances precision and recall, is also high at 0.86, suggesting strong overall performance in detecting fraud.

Upon examining the confusion matrix, we observe that the model has correctly classified the vast majority of transactions as either non-fraudulent or fraudulent. Specifically, it accurately identified all non-fraudulent transactions (true negatives) and most fraudulent transactions (true positives), while only misclassifying a small number of fraudulent transactions as non-fraudulent (false negatives).

Overall, these results indicate that the model is highly effective in detecting credit card fraud, with a negligible number of false positives and a high level of accuracy in identifying fraudulent transactions.