**Innovation Project Title**

**Credit card fraud detection:**

**Summary:**

Credit card fraud detection is a set of methods and techniques designed to block fraudulent purchases, both online and in- store. This is done by ensuring that you are dealing with the right cardholder and that the purchase is legitimate.

When it comes to identifying the cardholder, credit card fraud detection relies on authentication techniques such as MFA (multi-factor authentication), 3DS, biometrics, and OTP (one-time passwords).

However, it is also possible to detect credit card fraud by looking at anomalies in the transaction. For instance, an IP address could point to a suspicious geolocation. Similarly, a device with a never-seen configuration of software and hardware could raise red flags.

* A credit card number (legitimate or stolen).
* A CNP purchase (card not present), for

instance at an online store.

* A request for a refund. This will be made by the victim whose stolen card was used in the fraudulent purchase or the legitimate cardholder.

Ideally, however, you will flag credit card fraud before the purchase goes through, ensuring you do not have to deal with the last step.

* Theft: criminals steal or gain access to physical cards and use them.
* Skimming and cloning: making unauthorized copies of credit card details with special equipment known as a skimmer that can be installed on top of a legitimate card reader. The card numbers are then reused for a cloned card.
* Account takeover: when a fraudster gains unauthorized access to someone else's account. With a credit card linked to it. The problem is even worse if the account acts as an ewallet (BNPL, crypto or neobank account, for instance)
* Phishing and social engineering: taking

advantage of people in order to extract key information. Credit card details may be stolen by sending emails or SMS, or by deploying entire fake online shops.

* Address Verification Service (AVS): A service designed to confirm the cardholder's identity by looking at their registered address. The address is confirmed against the bank's records.
* 3-D Secure (3DS): A security layer that prompts users to enter a code to complete a purchase. Different card operators offer the service under different names, such as Visa Secure (Visa), SecureCode (Mastercard), or SafeKey (American Express).
* CVV: A CVV, or Card Verification Value, is a three-digit number located on the card. It is designed to verify that the card is indeed in possession of the customer at the time of purchase.

It's worth noting that these card security features add a certain level of friction. This is why Amazon, for instance, doesn't ask for a CVV at the checkout stage, as the company has determined that it slows down the process, impacts the customer experience negatively, and has other defenses in place to make sure it is in fact you logging in.

**Dataset and its details:**

We used the below dataset for our project:

**Dataset:**

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

We have taken the dataset from [www.kaggle.com](http://www.kaggle.com).

**Columns to be used for credit card fraut detection:**

In credit card fraud detection, various columns or features are typically used to train machine learning models to identify fraudulent transactions. Some common columns or features include:

* Transaction Amount: The amount of the transaction can be an important indicator, as fraudulent transactions may often involve unusual or high amounts.
* Transaction Date and Time: Analyzing the timing of transactions can help identify anomalies, such as transactions occurring at odd hours.
* Merchant Information: Details about the merchant, including location and type of business, can be valuable for detecting fraud.
* Cardholder Information: Information about the cardholder, such as their location, transaction history, and spending patterns, can be used to spot unusual behavior.
* Card Information: Features related to the card itself, such as the card's issuing bank, type (credit or debit), and expiration date, can provide insights into potential fraud.
* IP Address: For online transactions, the IP address used for the transaction can help identify suspicious activity.
* Geographical Information: Geographic location data, including the location of the transaction and the cardholder's typical location, can be used to spot unusual activity.

These are just some examples, and the specific columns used can vary depending on the dataset and the machine learning approach employed for fraud detection. Additionally, feature engineering and selection are crucial steps in creating effective fraud detection models.

**Libraries to installed for credit card fraud detection:**

In credit card fraud detection, various columns or features are typically used to train machine learning models to identify fraudulent transactions. Some common columns or features include:

Transaction Amount: The amount of the transaction can be an important indicator, as fraudulent transactions may often involve unusual or high amounts.

Transaction Date and Time: Analyzing the timing of transactions can help identify anomalies, such as transactions occurring at odd hours.

Merchant Information: Details about the merchant, including location and type of business, can be valuable for detecting fraud.

Cardholder Information: Information about the cardholder, such as their location, transaction history, and spending patterns, can be used to spot unusual behavior.

Transaction Type: Identifying the type of transaction (e.g., online purchase, in-store, ATM withdrawal) can be useful in fraud detection.

Card Information: Features related to the card itself, such as the card's issuing bank, type (credit or debit), and expiration date, can provide insights into potential fraud.

IP Address: For online transactions, the IP address used for the transaction can help identify suspicious activity.

Device Information: Data about the device used for the transaction (e.g., device type, operating system, browser) can be analyzed for anomalies.

Historical Transaction Patterns: Information about the cardholder's past transaction history and spending habits can be compared to the current transaction to detect deviations.

Geographical Information: Geographic location data, including the location of the transaction and the cardholder's typical location, can be used to spot unusual activity.

Anomaly Scores: Some models generate anomaly scores based on various features to flag transactions that deviate significantly from the norm.

These are just some examples, and the specific columns used can vary depending on the dataset and the machine learning approach employed for fraud detection. Additionally, feature engineering and selection are crucial steps in creating effective fraud detection.

To perform credit card fraud detection using machine learning in Python, you can use various libraries and frameworks. Here are some commonly used libraries and how to download them using pip, a package manager for Python:

**Scikit Learn:**

Scikit-Learn is a popular machine learning library that provides tools for classification, regression, and clustering, making it suitable for building fraud detection models.

To install Scikit-Learn:

pip install scikit-learn

**Pandas:**

Pandas is a library for data manipulation and analysis. It's often used for data preprocessing and exploration before training machine learning models.

To install Pandas:

pip install pandas

**Numpy:**

NumPy is a fundamental library for numerical operations in Python. It's often used in conjunction with Pandas for data handling.

To install NumPy:

pip install numpy

**Matplotlib and seaborn:**

These libraries are useful for data visualization, which can be essential for understanding your data and identifying patterns.

To install Matplotlib and Seaborn:

pip install matplotlib seaborn

**XGBoost and LightGBM:**

Gradient boosting libraries like XGBoost and LightGBM are popular for building high-performance machine learning models, including fraud detection models.

To install XGBoost:

pip install xgboost

To install LightGBM:

pip install lightgbm

**Imbalanced-Learn (imbalanced-learn):**

This library provides various techniques for handling imbalanced datasets, which is a common issue in fraud detection where fraudulent transactions are often rare compared to legitimate ones.

To install Imbalanced-Learn:

pip install imbalanced-learn

**TensorFlow or PyTorch:**

If you want to use deep learning techniques for fraud detection, you can install either TensorFlow or PyTorch. These libraries are widely used for building neural networks.

To install TensorFlow:

pip install tensorflow

To install PyTorch (note that the command may vary depending on your system and CUDA version):

pip install torch

Remember to create a virtual environment for your project to manage dependencies effectively. You can create one using venv or conda if you prefer using Conda environments.

**Testing and Training of credit card fraud detection:**

# Import necessary libraries

import pandas as pd

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

from sklearn.ensemble import RandomForestClassifier

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

# Load your credit card fraud dataset (assuming it's in a CSV file)

data = pd.read\_csv('credit\_card\_fraud\_data.csv')

# Separate features (X) and target labels (y)

X = data.drop('Class', axis=1)

y = data['Class']

# Split the data into training (80%) and testing (20%) sets

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

# Initialize and train a Random Forest classifier

clf = RandomForestClassifier()

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

# Make predictions on the test data

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

# Evaluate the model's performance

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

classification\_rep = classification\_report(y\_test, y\_pred)

print(f'Accuracy: {accuracy}')

print('Classification Report:\n', classification\_rep)

```

This code assumes you have a dataset in a CSV file named 'credit\_card\_fraud\_data.csv' with features (columns) and a 'Class' column indicating fraud (1) or not fraud (0). It splits the data into a training set (80%) and a testing set (20%), uses a Random Forest classifier for training, and then evaluates the model's performance using accuracy and a classification report.

Make sure to replace 'credit\_card\_fraud\_data.csv' with the actual filename and adapt the code as needed for your specific dataset and requirements. Additionally, consider tuning hyperparameters and using techniques like feature engineering to improve the model's performance.

**Matrices used for the accuracy check:**

**1. Accuracy:**

Accuracy measures the overall correctness of the model's predictions, i.e., the ratio of correctly classified transactions (both fraudulent and non-fraudulent) to the total number of transactions. While accuracy is important, it may not be the best metric if the dataset is imbalanced (i.e., there are significantly more non-fraudulent transactions than fraudulent ones), as high accuracy can be achieved by simply classifying all transactions as non-fraudulent.

print(cl('ACCURACY SCORE', attrs = ['bold']))

print(cl('----------------------------------------', attrs = ['bold']))

print(cl('Accuracy score of the Decision Tree model is {}'.format(accuracy\_score(y\_test, tree\_yhat)), attrs = ['bold']))

print(cl('----------------------------------------', attrs = ['bold']))

print(cl('Accuracy score of the KNN model is {}'.format(accuracy\_score(y\_test, knn\_yhat)), attrs = ['bold'], color = 'green'))

print(cl('----------------------------------------', attrs = ['bold']))

print(cl('Accuracy score of the Logistic Regression model is {}'.format(accuracy\_score(y\_test, lr\_yhat)), attrs = ['bold'], color = 'red'))

print(cl('----------------------------------------', attrs = ['bold']))

print(cl('Accuracy score of the SVM model is {}'.format(accuracy\_score(y\_test, svm\_yhat)), attrs = ['bold']))

print(cl('----------------------------------------', attrs = ['bold']))

print(cl('Accuracy score of the Random Forest Tree model is {}'.format(accuracy\_score(y\_test, rf\_yhat)), attrs = ['bold']))

print(cl('----------------------------------------', attrs = ['bold']))

print(cl('Accuracy score of the XGBoost model is {}'.format(accuracy\_score(y\_test, xgb\_yhat)), attrs = ['bold']))

print(cl('----------------------------------------', attrs = ['bold']))

**2. Precision:**

Precision, also known as positive predictive value, measures the accuracy of the model's positive predictions (fraudulent transactions). It is the ratio of correctly classified fraudulent transactions to the total number of transactions predicted as fraudulent. High precision indicates that the model has a low rate of false positives, which means that when it flags a transaction as fraudulent, it's likely to be correct.

**3. Recall (Sensitivity):**

Recall, also known as sensitivity or true positive rate, measures the model's ability to correctly identify all actual fraudulent transactions. It is the ratio of correctly classified fraudulent transactions to the total number of actual fraudulent transactions. High recall means that the model is good at catching fraudulent transactions and has a low rate of false negatives.

**4. F1-Score:**

The F1-Score is the harmonic mean of precision and recall. It provides a balanced measure that takes both false positives and false negatives into account. It's particularly useful when you want to strike a balance between precision and recall.

print(cl('F1 SCORE', attrs = ['bold']))

print(cl('----------------------------------------', attrs = ['bold']))

print(cl('F1 score of the Decision Tree model is {}'.format(f1\_score(y\_test, tree\_yhat)), attrs = ['bold']))

print(cl('----------------------------------------', attrs = ['bold']))

print(cl('F1 score of the KNN model is {}'.format(f1\_score(y\_test, knn\_yhat)), attrs = ['bold'], color = 'green'))

print(cl('----------------------------------------', attrs = ['bold']))

print(cl('F1 score of the Logistic Regression model is {}'.format(f1\_score(y\_test, lr\_yhat)), attrs = ['bold'], color = 'red'))

print(cl('----------------------------------------', attrs = ['bold']))

print(cl('F1 score of the SVM model is {}'.format(f1\_score(y\_test, svm\_yhat)), attrs = ['bold']))

print(cl('----------------------------------------', attrs = ['bold']))

print(cl('F1 score of the Random Forest Tree model is {}'.format(f1\_score(y\_test, rf\_yhat)), attrs = ['bold']))

print(cl('----------------------------------------', attrs = ['bold']))

print(cl('F1 score of the XGBoost model is {}'.format(f1\_score(y\_test, xgb\_yhat)), attrs = ['bold']))

print(cl('----------------------------------------', attrs = ['bold']))

**5. Area Under the Receiver Operating Characteristic Curve (AUC-ROC):**

The ROC curve plots the true positive rate (recall) against the false positive rate at various thresholds. AUC-ROC measures the area under this curve and provides an aggregate measure of the model's ability to discriminate between fraudulent and non-fraudulent transactions. A higher AUC-ROC score indicates better model performance.

**6. Area Under the Precision-Recall Curve (AUC-PR):**

Similar to AUC-ROC, the AUC-PR measures the area under the precision-recall curve. This metric is especially useful when dealing with imbalanced datasets, where the number of non-fraudulent transactions significantly outweighs fraudulent ones.

**7. Confusion Matrix:**

While not a single metric, a confusion matrix provides a detailed break down of true positives, true negatives, false positives, and false negatives, which can be useful for understanding the model's performance at different thresholds.

lr\_cm\_plot = plot\_confusion\_matrix(lr\_matrix,

classes = ['Non-Default(0)','Default(1)'],

normalize = False, title = 'Logistic Regression')

plt.savefig('lr\_cm\_plot.png')

plt.show()

# 4. Support Vector Machine

svm\_cm\_plot = plot\_confusion\_matrix(svm\_matrix,

classes = ['Non-Default(0)','Default(1)'],

normalize = False, title = 'SVM')

plt.savefig('svm\_cm\_plot.png')

plt.show()

# 5. Random forest tree

rf\_cm\_plot = plot\_confusion\_matrix(rf\_matrix,

classes = ['Non-Default(0)','Default(1)'],

normalize = False, title = 'Random Forest Tree')

plt.savefig('rf\_cm\_plot.png')

plt.show()

# 6. XGBoost

xgb\_cm\_plot = plot\_confusion\_matrix(xgb\_matrix,

classes = ['Non-Default(0)','Default(1)'],

normalize = False, title = 'XGBoost')

plt.savefig('xgb\_cm\_plot.png')

plt.show()

The choice of which metrics to emphasize depends on the specific goals and priorities of the credit card fraud detection system. In many cases, a balance between precision and recall is sought to minimize both false positives and false negatives. Additionally, the choice of metrics should consider the business costs associated with fraud detection and prevention.