Fraud Detection with Isolation Forest: Identifying Anomalous Credit Card Transactions

By Luis Mireles

This Python script implements an anomaly detection model using the Isolation Forest algorithm to identify fraudulent credit card transactions. The dataset is preprocessed by scaling the features and splitting it into training and testing sets. After training the model, predictions are made on the test set and various performance metrics, including the classification report, confusion matrix, and accuracy score, are calculated to evaluate the model's effectiveness in detecting anomalous transactions.

source data: https://www.kaggle.com/datasets/mlg-ulb/creditcardfraud

In [1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.ensemble import IsolationForest
from sklearn.metrics import classification_report, confusion_matrix, accuracy_score

data = pd.read_csv('/Users/fernando/Desktop/Code/Sources/creditcard.csv')
# columns = data.columns
# info = data.info()
# describe = data.describe()
# print(columns, info, describe)

# Separate features and labels
X = data.drop('Class', axis=1)
y = data['Class']

scaler = StandardScaler()
X = scaler.fit_transform(X)

# 30% of the dataset will be used for testing, while the remaining 70% will be used for training
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42)

outlier_fraction = len(y_train[y_train == 1]) / len(y_train)

classifier = IsolationForest(contamination=outlier_fraction, random_state=42)
classifier.fit(X_train, y_train)

# Isolation Forest returns 1 for inliers and -1 for outliers,
# so we need to convert it to 0 (normal) and 1 (anomaly)
y_pred = classifier.predict(X_test)
y_pred[y_pred == 1] = 0
y_pred[y_pred == -1] = 1

print("Classification Report:\n", classification_report(y_test, y_pred))
print("Confusion Matrix:\n", confusion_matrix(y_test, y_pred))
print("Accuracy Score:", accuracy_score(y_test, y_pred))
```

```
Classification Report:
              precision    recall  f1-score   support

           0       1.00      1.00      1.00     85307
           1       0.31      0.32      0.32       136

    accuracy                           1.00     85443
   macro avg       0.65      0.66      0.66     85443
weighted avg       1.00      1.00      1.00     85443

Confusion Matrix:
 [[85208    99]
 [   92    44]]
Accuracy Score: 0.9977645915990777
```

Classification Report: This report shows the main classification metrics for each class (0 - normal transactions, 1 - fraudulent transactions) and overall performance.

Precision: The ratio of true positive predictions to the total number of positive predictions (true positives + false positives). Precision measures the ability of the classifier to correctly identify positive instances.

Class 0 (normal transactions) has a precision of 1.00, meaning that almost all normal transactions were correctly classified.

Class 1 (fraudulent transactions) has a precision of 0.31, meaning that only 31% of the transactions classified as fraudulent were actually fraudulent. Recall: The ratio of true positive predictions to the total number of actual positive instances (true positives + false negatives). Recall measures the ability of the classifier to identify all the positive instances.

Class 0 (normal transactions) has a recall of 1.00, meaning that almost all normal transactions were identified by the classifier.

Class 1 (fraudulent transactions) has a recall of 0.32, meaning that the classifier identified 32% of the actual fraudulent transactions.

F1-score: The harmonic mean of precision and recall. It provides a single score that balances both precision and recall. A higher F1-score indicates better performance.

Class 0 (normal transactions) has an F1-score of 1.00, indicating excellent performance.

Class 1 (fraudulent transactions) has an F1-score of 0.32, indicating relatively poor performance.

Accuracy: The overall accuracy of the model, calculated as the ratio of correct predictions to the total number of instances. In this case, the accuracy is 0.9977, meaning that the model correctly classified approximately 99.77% of the transactions.

Macro avg: The unweighted mean of the metrics for each class. It gives equal importance to both classes, regardless of their relative frequencies. The macro average precision, recall, and F1-score are 0.65, 0.66, and 0.66, respectively.

Weighted avg: The weighted mean of the metrics for each class. The weights are determined by the number of instances for each class. The weighted average precision, recall, and F1-score are all close to 1.00, indicating that the model performs well on the majority class (normal transactions).

Confusion Matrix: A table that shows the number of true positive (TP), true negative (TN), false positive (FP), and false negative (FN) predictions. [[85208 99] [92 44]]

Top-left (85208): True Negatives (TN) - The number of normal transactions correctly classified as normal. Top-right (99): False Positives (FP) - The number of normal transactions incorrectly classified as fraudulent. Bottom-left (92): False Negatives (FN) - The number of fraudulent transactions incorrectly classified as normal. Bottom-right (44): True Positives (TP) - The number of fraudulent transactions correctly classified as fraudulent.

Accuracy Score: The overall accuracy of the model, calculated as the ratio of correct predictions (TP + TN) to the total number of instances (TP + TN + FP + FN). In this case, the accuracy is 0.9977, meaning that the model correctly classified approximately 99.77% of the transactions.

Comments: The results indicate that the model is executing well at detecting normal transactions but the model is struggling to correctly identify fraudulent transactions. This is a common issue with imbalanced datasets, where the majority class which in this case are (normal transactions) dominate the minority class (fraudulent transactions). In situations like this it is normal for the classifier to tend to perform well on the majority class but poorly on the minority class, as it learns to focus on the patterns within the majority class during training.