Example: Fraud Detection in Credit Card Transactions

Goal: Identify fraudulent credit card transactions.
Data: Historical credit card transaction data.
Key Challenges:

1. Selecting appropriate machine learning models suitable for fraud detection.
2. Choosing the right evaluation metrics to assess classification performance, particularly in the context of highly imbalanced class labels.

Context

It is important that credit card companies are able to recognize fraudulent credit card transactions so that customers are not charged for items that they did not purchase.

Content

The dataset contains transactions made by credit cards in September 2013 by European cardholders. This dataset presents transactions that occurred in two days, where we have 492 frauds out of 284,807 transactions. The dataset is highly unbalanced, the positive class (frauds) account for 0.172% of all transactions.

It contains only numerical input variables which are the result of a PCA transformation. Unfortunately, due to confidentiality issues, we cannot provide the original features and more background information about the data. Features V1, V2, … V28 are the principal components obtained with PCA, the only features which have not been transformed with PCA are 'Time' and 'Amount'. Feature 'Time' contains the seconds elapsed between each transaction and the first transaction in the dataset. The feature 'Amount' is the transaction Amount, this feature can be used for example-dependant cost-sensitive learning. Feature 'Class' is the response variable and it takes value 1 in case of fraud and 0 otherwise.

Given the class imbalance ratio, we recommend measuring the accuracy using the Area Under the Precision-Recall Curve (AUPRC). Confusion matrix accuracy is not meaningful for unbalanced classification.

Steps to address this fraud detection problem

Step 1: Take an initial look at the data

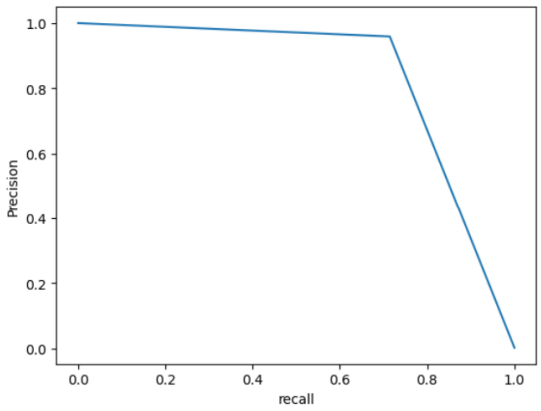
	Time	V1	V2	V3	...	V27	V28	Amount	Class
0	0.0	-1.359807	-0.072781	2.536347	...	0.133558	-0.021053	149.62	0
1	0.0	1.191857	0.266151	0.166480	...	-0.008983	0.014724	2.69	0
2	1.0	-1.358354	-1.340163	1.773209	...	-0.055353	-0.059752	378.66	0
3	1.0	-0.966272	-0.185226	1.792993	...	0.062723	0.061458	123.50	0
4	2.0	-1.158233	0.877737	1.548718	...	0.219422	0.215153	69.99	0

[5 rows x 31 columns]

Step 2: Prepare the data for machine learning models
(1).Get feature table X, and label y
(2).split train and test set
(3).Calculate class weights for handing class imbalance problem
(4).Create and train a Random Forest model with class weights

Random Forest was chosen for it interpretability, as it can identify which features contribute most to the classification.

(5).Make prediction on the test set
(6).Evaluate the model
(7).Plot precision-recall curve



(7).Understand the results.

	precision	recall	f1-score	support
0	1.00	1.00	1.00	56864
1	0.96	0.71	0.82	98
accuracy			1.00	56962
macro avg	0.98	0.86	0.91	56962
weighted avg	1.00	1.00	1.00	56962

Precision = 0.96: Of the transactions we predicted as fraud, 95% were actually fraud.
Recall = 0.71: We successfully caught 80% of all the fraudulent transactions in the test set

Given the class imbalance ratio (99.827% vs 0.173%), I recommend measuring the accuracy using the Area Under the Precision-Recall Curve (AUPRC). Confusion matrix accuracy is not meaningful for unbalanced classification.
AUPRC=0.685
The AUPRC is relatively low; for further analysis, more advanced models could be explored to improve it.

Data:
https://www.kaggle.com/datasets/mlg-ulb/creditcardfraud?resource=download

Code:

```
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.utils import class_weight
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import classification_report, precision_recall_curve,average_precision_score
import matplotlib.pyplot as plt
#load the dataset
df=pd.read_csv(r'creditcard.csv')
#take a look at the data (the first 5 samples)
print(df.head(5))

#check the class imbalance
print(df['Class'].value_counts())
print(df['Class'].value_counts(normalize=True)*100)

#get features X, and label y
X=df.drop('Class',axis=1)
y=df['Class']

#split train and test set
X_train,X_test,y_train,y_test=train_test_split(X,y,test_size=0.2,random_state=0,stratify=y)
```

```
# Handle class imbalance: calculate class weights
#Weight_i = N / (Number of Classes * N_i)
#a=df['Class'].value_counts()
#weight0,weight1=sum(a)/(2*a[0]),sum(a)/(2*a[1])
class_weights = class_weight.compute_class_weight('balanced', classes=np.unique(y_train), y=y_train)
class_weight_dict = dict(enumerate(class_weights))

#creast and train a Random Forest model with class weights
model=RandomForestClassifier(n_estimators=20,random_state=0,class_weight=class_weight_dict,n_jobs=-1)
model.fit(X_train,y_train)

#Make prediction on the test set
y_pred=model.predict(X_test)

#Evaluate the model
print(classification_report(y_test, y_pred))

#Understand the results
'''The classification_report will show high accuracy but, more importantly, it will show Precision and Recall for the
fraud class (Class 1).
              precision    recall  f1-score   support
           0       1.00      1.00      1.00     56864
           1       0.96      0.71      0.82        98
Precision = 0.96: Of the transactions we predicted as fraud, 95% were actually fraud.
Recall = 0.71: We successfully caught 80% of all the fraudulent transactions in the test set
'''
#average precision (AP) is a better evaluation metric than AUPR for imbalanced labels
print(average_precision_score(y_test, y_pred))
#plot precision-recall curve
precision, recall, thresholds = precision_recall_curve(y_test, y_pred)
plt.plot(recall,precision)
plt.xlabel('recall')
plt.ylabel('Precision')
plt.show()
```