credit-card-fraud-detection-part2-modeling

July 27, 2024

Credit Card Fraud Detection

Part 2. Classification Models

Yatharth Gautam

0.0.1 Notebook on Exploratory Data Analysis: Credit_Card_Fraud_Detection_Part1_EDA

1 Contents

1. Introduction

[1]:
```
# Importing necessary libraries

import numpy as np
import pandas as pd
import math
import matplotlib.pyplot as plt
import seaborn as sns
import plotly.express as px
from plotly.subplots import make_subplots
import plotly.graph_objects as go
from plotly.offline import init_notebook_mode, iplot
init_notebook_mode(connected=True)
```

```
import shutil
columns = shutil.get_terminal_size().columns

from sklearn.model_selection import train_test_split
from sklearn import metrics
from sklearn.preprocessing import MinMaxScaler

from sklearn.metrics import plot_confusion_matrix
from sklearn.metrics import average_precision_score

from sklearn.linear_model import LogisticRegression
from sklearn.neighbors import KNeighborsClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn import svm
from sklearn.naive_bayes import GaussianNB
from sklearn.ensemble import RandomForestClassifier
from sklearn.discriminant_analysis import LinearDiscriminantAnalysis
from sklearn.linear_model import SGDClassifier
from sklearn.linear_model import RidgeClassifier

import imblearn
from imblearn.under_sampling import RandomUnderSampler
from imblearn.over_sampling import RandomOverSampler
from imblearn.under_sampling import TomekLinks
from imblearn.over_sampling import SMOTE
from imblearn.under_sampling import NearMiss

from IPython.display import Javascript
from IPython.display import display
```

1.1 Data

Source: https://www.kaggle.com/mlg-ulb/creditcardfraud

The dataset contains information on the transactions made using credit cards by European cardholders, in two particular days of September 2013. It presents a total of 284807 transactions, of which 492 were fraudulent. Clearly, the dataset is highly imbalanced, the positive class (fraudulent transactions) accounting for only 0.173% of all transactions.

For a particular transaction, the feature **Time** represents the time (in seconds) elapsed between the transaction and the very first transaction, **Amount** represents the amount of the transaction and **Class** represents the status of the transaction with respect to authenticity. The class of an authentic (resp. fraudulent) transaction is taken to be 0 (resp. 1). Rest of the variables (**V1** to **V28**) are obtained from principle component analysis (PCA) transformation on original features that are not available due to confidentiality.

[2]:
```
# The dataset
```

```python
data = pd.read_csv('../input/creditcardfraud/creditcard.csv')
data
```

[2]:

```
            Time         V1         V2        V3        V4        V5  \
0            0.0  -1.359807  -0.072781  2.536347  1.378155 -0.338321
1            0.0   1.191857   0.266151  0.166480  0.448154  0.060018
2            1.0  -1.358354  -1.340163  1.773209  0.379780 -0.503198
3            1.0  -0.966272  -0.185226  1.792993 -0.863291 -0.010309
4            2.0  -1.158233   0.877737  1.548718  0.403034 -0.407193
…            …          …          …         …         …         …
284802  172786.0 -11.881118  10.071785 -9.834783 -2.066656 -5.364473
284803  172787.0  -0.732789  -0.055080  2.035030 -0.738589  0.868229
284804  172788.0   1.919565  -0.301254 -3.249640 -0.557828  2.630515
284805  172788.0  -0.240440   0.530483  0.702510  0.689799 -0.377961
284806  172792.0  -0.533413  -0.189733  0.703337 -0.506271 -0.012546

              V6        V7        V8        V9  …       V21       V22  \
0       0.462388  0.239599  0.098698  0.363787  … -0.018307  0.277838
1      -0.082361 -0.078803  0.085102 -0.255425  … -0.225775 -0.638672
2       1.800499  0.791461  0.247676 -1.514654  …  0.247998  0.771679
3       1.247203  0.237609  0.377436 -1.387024  … -0.108300  0.005274
4       0.095921  0.592941 -0.270533  0.817739  … -0.009431  0.798278
…            …         …         …         …  …         …         …
284802 -2.606837 -4.918215  7.305334  1.914428  …  0.213454  0.111864
284803  1.058415  0.024330  0.294869  0.584800  …  0.214205  0.924384
284804  3.031260 -0.296827  0.708417  0.432454  …  0.232045  0.578229
284805  0.623708 -0.686180  0.679145  0.392087  …  0.265245  0.800049
284806 -0.649617  1.577006 -0.414650  0.486180  …  0.261057  0.643078

             V23       V24       V25       V26       V27       V28  Amount  \
0      -0.110474  0.066928  0.128539 -0.189115  0.133558 -0.021053  149.62
1       0.101288 -0.339846  0.167170  0.125895 -0.008983  0.014724    2.69
2       0.909412 -0.689281 -0.327642 -0.139097 -0.055353 -0.059752  378.66
3      -0.190321 -1.175575  0.647376 -0.221929  0.062723  0.061458  123.50
4      -0.137458  0.141267 -0.206010  0.502292  0.219422  0.215153   69.99
…            …         …         …         …         …         …       …
284802  1.014480 -0.509348  1.436807  0.250034  0.943651  0.823731    0.77
284803  0.012463 -1.016226 -0.606624 -0.395255  0.068472 -0.053527   24.79
284804 -0.037501  0.640134  0.265745 -0.087371  0.004455 -0.026561   67.88
284805 -0.163298  0.123205 -0.569159  0.546668  0.108821  0.104533   10.00
284806  0.376777  0.008797 -0.473649 -0.818267 -0.002415  0.013649  217.00

        Class
0           0
1           0
2           0
3           0
```

```
4             0
…        …
284802        0
284803        0
284804        0
284805        0
284806        0

[284807 rows x 31 columns]
```

[3]:
```python
features = list(data.columns)
features.remove('Class')
```

[4]:
```python
# Statistical descriptions of the features

features_stat = data.drop(['Class'], axis = 1)
features_stat.describe()
```

[4]:
```
                Time            V1            V2            V3            V4  \
count  284807.000000  2.848070e+05  2.848070e+05  2.848070e+05  2.848070e+05
mean    94813.859575  3.918649e-15  5.682686e-16 -8.761736e-15  2.811118e-15
std     47488.145955  1.958696e+00  1.651309e+00  1.516255e+00  1.415869e+00
min         0.000000 -5.640751e+01 -7.271573e+01 -4.832559e+01 -5.683171e+00
25%     54201.500000 -9.203734e-01 -5.985499e-01 -8.903648e-01 -8.486401e-01
50%     84692.000000  1.810880e-02  6.548556e-02  1.798463e-01 -1.984653e-02
75%    139320.500000  1.315642e+00  8.037239e-01  1.027196e+00  7.433413e-01
max    172792.000000  2.454930e+00  2.205773e+01  9.382558e+00  1.687534e+01

                 V5            V6            V7            V8            V9  \
count  2.848070e+05  2.848070e+05  2.848070e+05  2.848070e+05  2.848070e+05
mean  -1.552103e-15  2.040130e-15 -1.698953e-15 -1.893285e-16 -3.147640e-15
std    1.380247e+00  1.332271e+00  1.237094e+00  1.194353e+00  1.098632e+00
min   -1.137433e+02 -2.616051e+01 -4.355724e+01 -7.321672e+01 -1.343407e+01
25%   -6.915971e-01 -7.682956e-01 -5.540759e-01 -2.086297e-01 -6.430976e-01
50%   -5.433583e-02 -2.741871e-01  4.010308e-02  2.235804e-02 -5.142873e-02
75%    6.119264e-01  3.985649e-01  5.704361e-01  3.273459e-01  5.971390e-01
max    3.480167e+01  7.330163e+01  1.205895e+02  2.000721e+01  1.559499e+01

       …           V20           V21           V22           V23  \
count  …  2.848070e+05  2.848070e+05  2.848070e+05  2.848070e+05
mean   …  5.126845e-16  1.473120e-16  8.042109e-16  5.282512e-16
std    …  7.709250e-01  7.345240e-01  7.257016e-01  6.244603e-01
min    … -5.449772e+01 -3.483038e+01 -1.093314e+01 -4.480774e+01
25%    … -2.117214e-01 -2.283949e-01 -5.423504e-01 -1.618463e-01
50%    … -6.248109e-02 -2.945017e-02  6.781943e-03 -1.119293e-02
75%    …  1.330408e-01  1.863772e-01  5.285536e-01  1.476421e-01
max    …  3.942090e+01  2.720284e+01  1.050309e+01  2.252841e+01
```

```
                V24           V25           V26           V27           V28  \
count  2.848070e+05  2.848070e+05  2.848070e+05  2.848070e+05  2.848070e+05
mean   4.456271e-15  1.426896e-15  1.701640e-15 -3.662252e-16 -1.217809e-16
std    6.056471e-01  5.212781e-01  4.822270e-01  4.036325e-01  3.300833e-01
min   -2.836627e+00 -1.029540e+01 -2.604551e+00 -2.256568e+01 -1.543008e+01
25%   -3.545861e-01 -3.171451e-01 -3.269839e-01 -7.083953e-02 -5.295979e-02
50%    4.097606e-02  1.659350e-02 -5.213911e-02  1.342146e-03  1.124383e-02
75%    4.395266e-01  3.507156e-01  2.409522e-01  9.104512e-02  7.827995e-02
max    4.584549e+00  7.519589e+00  3.517346e+00  3.161220e+01  3.384781e+01

              Amount
count  284807.000000
mean       88.349619
std       250.120109
min         0.000000
25%         5.600000
50%        22.000000
75%        77.165000
max     25691.160000

[8 rows x 30 columns]
```

1.2 Objectives of the project

1.2.1 Primary objective:

Classification of transactions as authentic or fraudulent. To be prcise, given the data on **Time**, **Amount** and transformed features **V1** to **V28** for a particular transaction, our goal is to correctly classify the transaction as **authentic** or **fraudulent**. We employ different techniques to build classification models and compare them by various evaluation metrics.

1.2.2 Secondary objectives:

Answering the following questions using machine learning and statistical tools and techniques.

- When a fraudulent transaction is made, is it followed soon by one or more such fraudulent transactions? In other words, do the attackers make consecutive fraudulent transactions in a short span of time?
- Is the amount of a fraudulent transaction generally larger than that of an authentic transaction?
- Is there any indication in the data that fraudulent transactions occur at high-transaction period?
- It is seen from the data that the number of transactions are high in some time intervals and low in between. Does the occurance of frauds related to these time intervals?
- There are a few time-points which exhibits high number of fraud transactions. Is it due to high number of total transactions or due to some other reason?

In this part we shall classify transactions as authentic or fraudulent based on the information available on independent features (time, amount and the transformed variables V1-V28). One issue with the dataset is that it is highly imbalanced in terms of the target variable Class. Thus we run into the risk of training the models with a representative sample of fraudulent transactions of extremely small size. We employ different approaches to deal with this problem. The performance of each model is checked through various evaluation metrics and is summarized in tabulated form.

2. Evaluation Metrics

Any prediction about a binary categorical target variable falls into one of the four categories: - True Positive: The classification model correctly predicts the output to be positive - True Negative: The classification model correctly predicts the output to be negative - False Positive: The classification model incorrectly predicts the output to be positive - False Negative: The classification model incorrectly predicts the output to be negative

↓ Actual state / Predicted state →	Positive	Negative
Positive	True Positive	False Negative
Negative	False Positive	True Negative

Let **TP**, **TN**, **FP** and **FN** respectively denote the number of **true positives**, **true negatives**, **false positives** and **false negatives** among the predictions made by a particular classification model. Below we give the definitions of some evaluation metrics based on these four quantities.

$$\text{Accuracy} = \frac{\text{Number of correct predictions}}{\text{Number of total predictions}} = \frac{TP + TN}{TP + TN + FP + FN}$$

- **Precision-Recall Metrics**

$$\text{Precision} = \frac{\text{Number of true positive predictions}}{\text{Number of total positive predictions}} = \frac{TP}{TP + FP}$$

$$\text{Recall} = \frac{\text{Number of true positive predictions}}{\text{Number of total positive cases}} = \frac{TP}{TP + FN}$$

$$\text{Fowlkes-Mallows index (FM)} = \text{Geometric mean of Precision and Recall} = \sqrt{\text{Precision} \times \text{Recall}}$$

$$F_1\text{-Score} = \text{Harmonic mean of Precision and Recall} = \frac{2 \times \text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}$$

$$F_\beta\text{-Score} = \frac{(1+\beta^2) \times \text{Precision} \times \text{Recall}}{(\beta^2 \times \text{Precision}) + \text{Recall}},$$

where β is a positive factor, chosen such that Recall is β times as important as Precision in the analysis. Popular choices of β are 0.5, 1 and 2.

- **Sensitivity-Specificity Metrics**

$$\text{Sensitivity} = \frac{\text{Number of true positive predictions}}{\text{Number of total positive cases}} = \frac{TP}{TP + FN}$$

$$\text{Specificity} = \frac{\text{Number of true negative predictions}}{\text{Number of total negative cases}} = \frac{TN}{TN + FP}$$

$$\text{G-mean} = \text{Geometric mean of Sensitivity and Specificity} = \sqrt{\text{Sensitivity} \times \text{Specificity}}$$

- **Area Under Curve (AUC) Metrics**

Consider the following quantities:

$$\text{True Positive Rate (TPR)} = \frac{\text{Number of true positive predictions}}{\text{Number of total positive cases}} = \frac{TP}{TP + FN}$$

$$\text{False Positive Rate (FPR)} = \frac{\text{Number of false positive predictions}}{\text{Number of total negative cases}} = \frac{FP}{FP + TN}$$

The Receiver Operating Characteristic (ROC) curve is obtained by plotting TPR against FPR for a number of threshold probability values. The area under ROC curve (ROC-AUC) serves as a valid evaluation metric.

Similarly, the Precision-Recall (PR) curve is obtained by plotting Precision against Recall for a number of threshold probability values. The area under PR curve (PR-AUC) is also a valid evaluation metric. Another widely used metric in this regard is the Average Precision (AP), which is a weighted mean of precisions at each threshold, with the weights being increase in recall from the previous threshold.

- **Other Metrics**

$$\text{Matthews Correlation Coefficient (MCC)} = \frac{(TP \times TN) - (FP \times FN)}{\sqrt{(TP + FP) \times (TP + FN) \times (TN + FP) \times (TN + FN)}}$$

Unlike the previous metrics, **MCC** varies from -1 (worst case scenario) to 1 (best case scenario: perfect prediction).

Note that **Recall** and **Sensitivity** are essentially the same quantity.

Among the discussed metrics, some good choices to evaluate models, in particular for imbalanced dataset are **MCC** and F_1**-Score**, while **Precision** and **Recall** also give useful information. We shall not give much importance to the **Accuracy** metric in this project as it produces misleading conclusion when the classes are not balanced. In the problem at hand, false negative (a fraudulent transaction being classified as authentic) is more dangerous than false positive (an authentic transaction being classified as fraudulent) as in the former case, the fraudster can cause further financial damage, while in the latter case the bank can cross-verify the authenticity of the transaction from the card-user after taking necessary steps to secure the card. Considering this fact, we give F_2**-Score** special importance in evaluating the models.

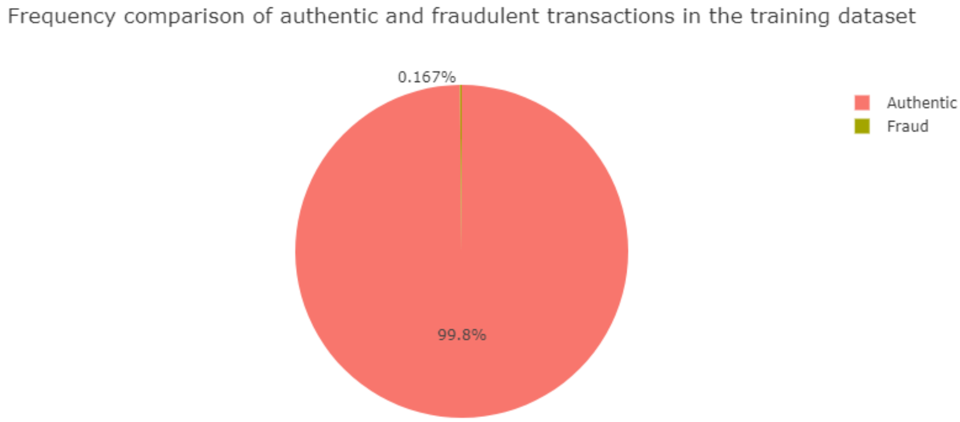
[5]:
```
EvalMetricLabels = ['MCC', 'F1-Score', 'F2-Score', 'Recall', 'Precision',
                    'FM index', 'Specificity', 'G-mean', 'F0.5-Score',␣
↪'Accuracy']
```

3. Train-Test Split

1.3 Splitting the data into training set and testing set

[6]:
```
# Separating independent variables and target variable

y = data['Class'] # target variable
X = data.drop('Class', axis = 1) # independent variables

# Constructing training set and testing set

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.2,␣
↪random_state = 25)
```

[7]:
```
z = list(y_train)

count0 = 0
count1 = 0

for i in z:
    if i == 0:
        count0 = count0 + 1
    elif i == 1:
        count1 = count1 + 1

# Class frequencies

class_label_train = ['Authentic', 'Fraud']
class_frequency_train = [count0, count1]

fig1 = px.pie(values = class_frequency_train,
             names = class_label_train,
             title = 'Frequency comparison of authentic and fraudulent␣
↪transactions in the training dataset',
             template = 'ggplot2'
            )
fig1.show()
```



The plot indicates extremely high imbalance between the two classes in the training set which may lead to producing misleading models.

1.4 Balancing the training set

1.4.1 Separating the training set by class

[8]:
```
data_train = pd.concat([X_train, y_train], axis = 1)
data_train_authentic = data_train[data_train['Class'] == 0]
data_train_fraudulent = data_train[data_train['Class'] == 1]
```

[9]:
```
# Amount vs Time for authentic and fraudulent transactions in the training set

class_list_train = list(y_train)
fraud_status_train = []
for i in range(len(class_list_train)):
    fraud_status_train.append(bool(class_list_train[i]))

fig1 = px.scatter(data_train,
                  x = 'Time',
                  y = 'Amount',
                  facet_col = fraud_status_train,
                  color = fraud_status_train,
                  title = 'Amount vs Time for the training set',
                  template = 'ggplot2'
                 )
fig1.show()
```

1.4.2 Random under-sampling (RUS)

We take a subset of the majority class to balance the training dataset.

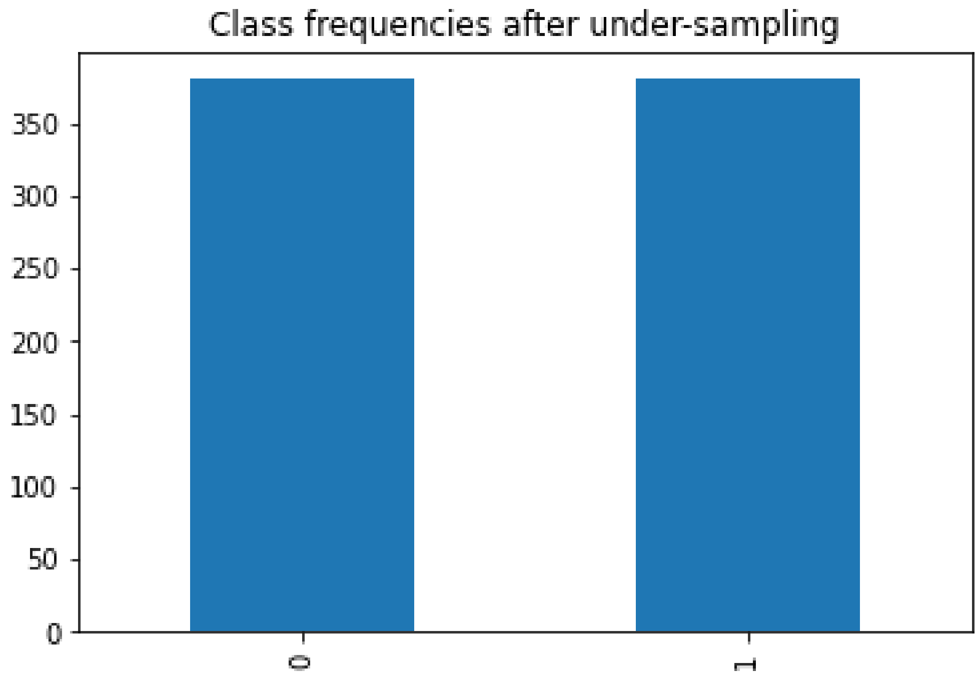
Advantage: Improves run-time and solves any storage issue due to large learning dataset.

Disadvantage: Ignores a chunk of information that could be impactful in the analysis and uses only a sample representative of the majority class which is not guarunteed to reflect the same accurately.

[10]:
```
data_train_authentic_under = data_train_authentic.
    ↪sample(len(data_train_fraudulent))
data_train_under = pd.concat([data_train_authentic_under,␣
    ↪data_train_fraudulent], axis = 0)

X_train_under = data_train_under.drop('Class', axis = 1)
y_train_under = data_train_under['Class']

print('Class frequencies after under-sampling:')
print(y_train_under.value_counts())
y_train_under.value_counts().plot(kind = 'bar', title = 'Class frequencies␣
    ↪after under-sampling')
```

```
Class frequencies after under-sampling:
0    380
1    380
Name: Class, dtype: int64
```

[10]: <AxesSubplot:title={'center':'Class frequencies after under-sampling'}>



[11]:
```
# Amount vs Time for authentic and fraudulent transactions in the training set␣
↪after random under-sampling

class_list = list(y_train_under)
fraud_status = []
for i in range(len(class_list)):
    fraud_status.append(bool(class_list[i]))

fig1 = px.scatter(data_train_under,
                 x = 'Time',
                 y = 'Amount',
                 facet_col = fraud_status,
                 color = fraud_status,
                 title = 'Amount vs Time for the training set after random␣
↪under-sampling',
                 template = 'ggplot2'
                )
fig1.show()
```

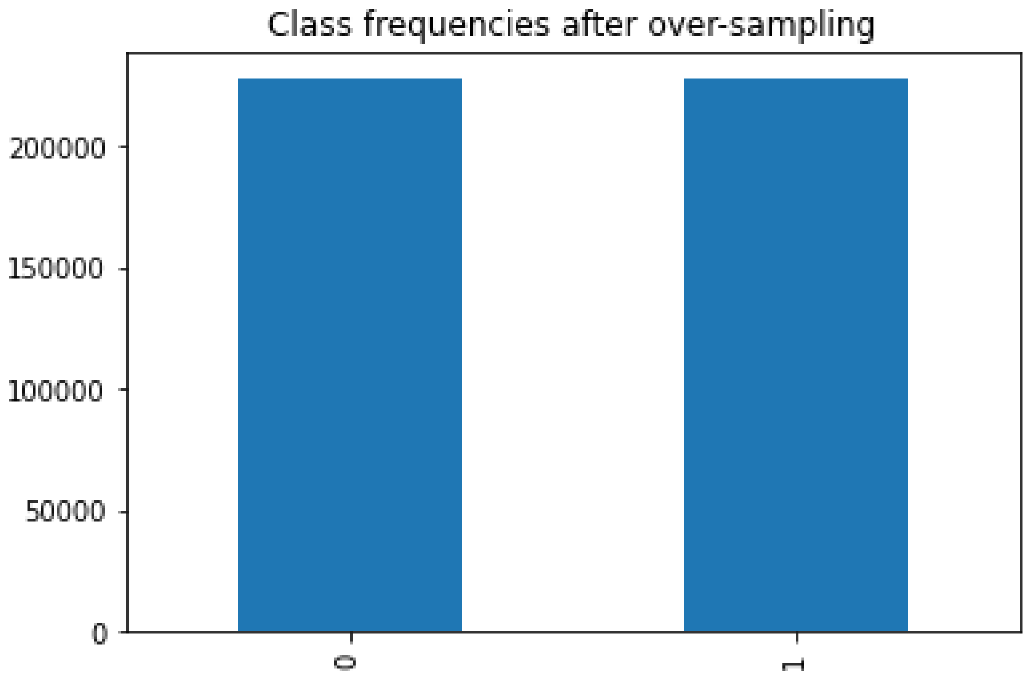
Comparing with the same plots for full dataset, we see that high-amount authentic transactions are not represented in the sample taken from the majority class. This indicates a general drawback of the under-sampling techniques which throws away a major chunk of information from the majority class and suffers from not representing the same accurately.

1.4.3 Random over-sampling (ROS)

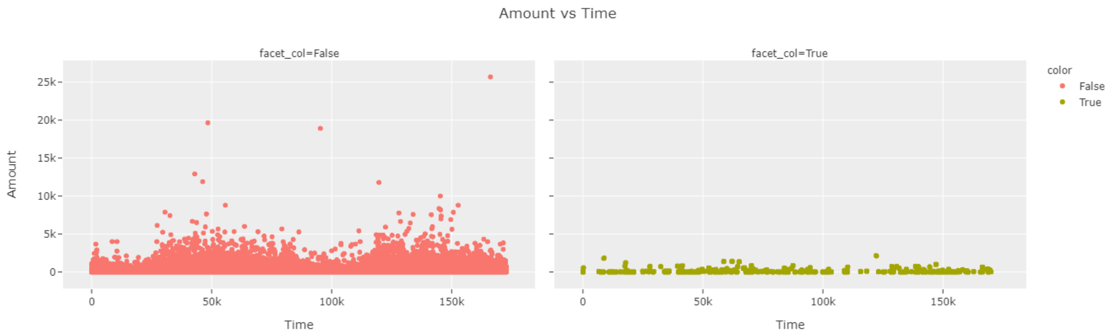
[12]:
```
data_train_fraudulent_over = data_train_fraudulent.
 ↪sample(len(data_train_authentic), replace = 'True')
data_train_over = pd.concat([data_train_authentic, data_train_fraudulent_over],␣
 ↪axis = 0)

X_train_over = data_train_over.drop('Class', axis = 1)
y_train_over = data_train_over['Class']

print('Class frequencies after over-sampling:')
print(y_train_over.value_counts())
y_train_over.value_counts().plot(kind = 'bar', title = 'Class frequencies after␣
 ↪over-sampling')
```

```
Class frequencies after over-sampling:
0    227465
1    227465
Name: Class, dtype: int64
```

[12]:
```
<AxesSubplot:title={'center':'Class frequencies after over-sampling'}>
```



[13]:
```
# Amount vs Time for authentic and fraudulent transactions in the training set␣
↪after random over-sampling

class_list = list(y_train_over)
fraud_status = []
for i in range(len(class_list)):
    fraud_status.append(bool(class_list[i]))

fig1 = px.scatter(data_train_over,
                  x = 'Time',
                  y = 'Amount',
                  facet_col = fraud_status,
                  color = fraud_status,
                  title = 'Amount vs Time',
                  template = 'ggplot2'
                 )
fig1.show()
```



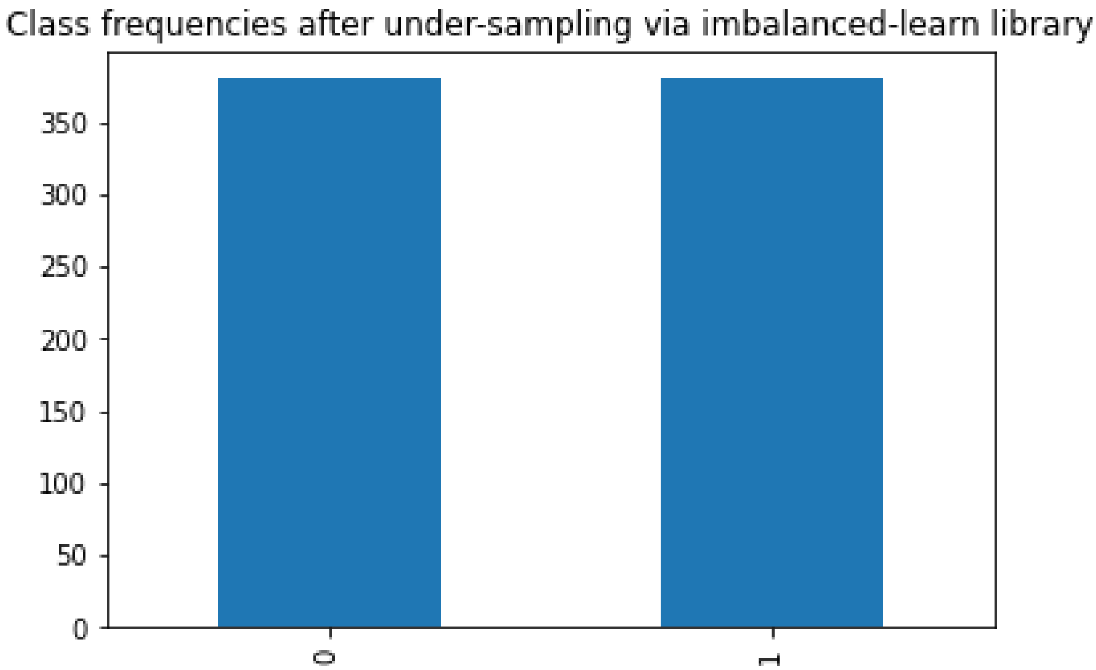
These plots resemble the corresponding *Amount vs Time* plots for the full dataset much more accurately than the corresponding plots for the training set obtained from random under-sampling.

1.4.4 Random under-sampling with imbalanced-learn library (RUS-IL)

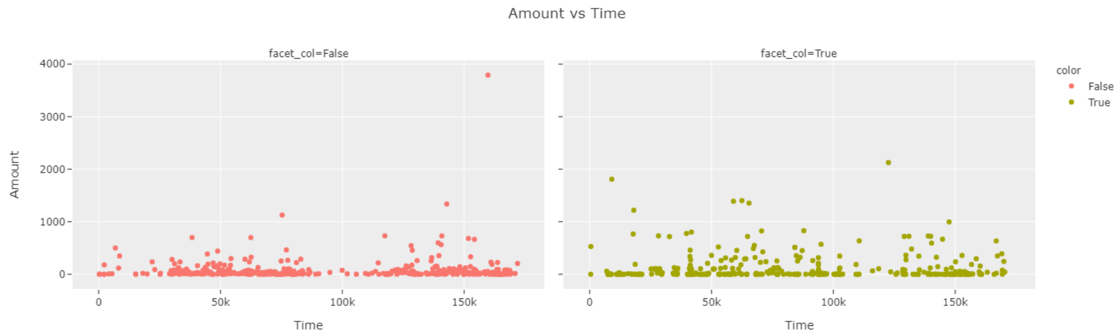
[14]:
```
imblearn_rus = RandomUnderSampler(random_state = 40, replacement = True)
X_train_rus, y_train_rus = imblearn_rus.fit_resample(X_train, y_train)

X_train_rus = pd.DataFrame(X_train_rus)
X_train_rus.columns = features

y_train_rus = pd.DataFrame(y_train_rus)
y_train_rus.columns = ['Class']

data_train_under_imblearn = pd.concat([X_train_rus, y_train_rus], axis = 1)

X_train_under_imblearn = data_train_under_imblearn.drop('Class', axis = 1)
y_train_under_imblearn = data_train_under_imblearn['Class']

print('Class frequencies after under-sampling via imbalanced-learn library:')
print(y_train_under_imblearn.value_counts())
y_train_under_imblearn.value_counts().plot(kind = 'bar',
                                           title = 'Class frequencies after␣
 ↪under-sampling via imbalanced-learn library')
```

```
Class frequencies after under-sampling via imbalanced-learn library:
0    380
1    380
Name: Class, dtype: int64
```

[14]:
```
<AxesSubplot:title={'center':'Class frequencies after under-sampling via
imbalanced-learn library'}>
```



[15]:
```
# Amount vs Time for authentic and fraudulent transactions in the training set␣
↪after random under-sampling via imblearn

class_list = list(y_train_under_imblearn)
fraud_status = []
for i in range(len(class_list)):
    fraud_status.append(bool(class_list[i]))

fig1 = px.scatter(data_train_under_imblearn,
                  x = 'Time',
                  y = 'Amount',
                  facet_col = fraud_status,
                  color = fraud_status,
                  title = 'Amount vs Time',
                  template = 'ggplot2'
                 )
fig1.show()
```

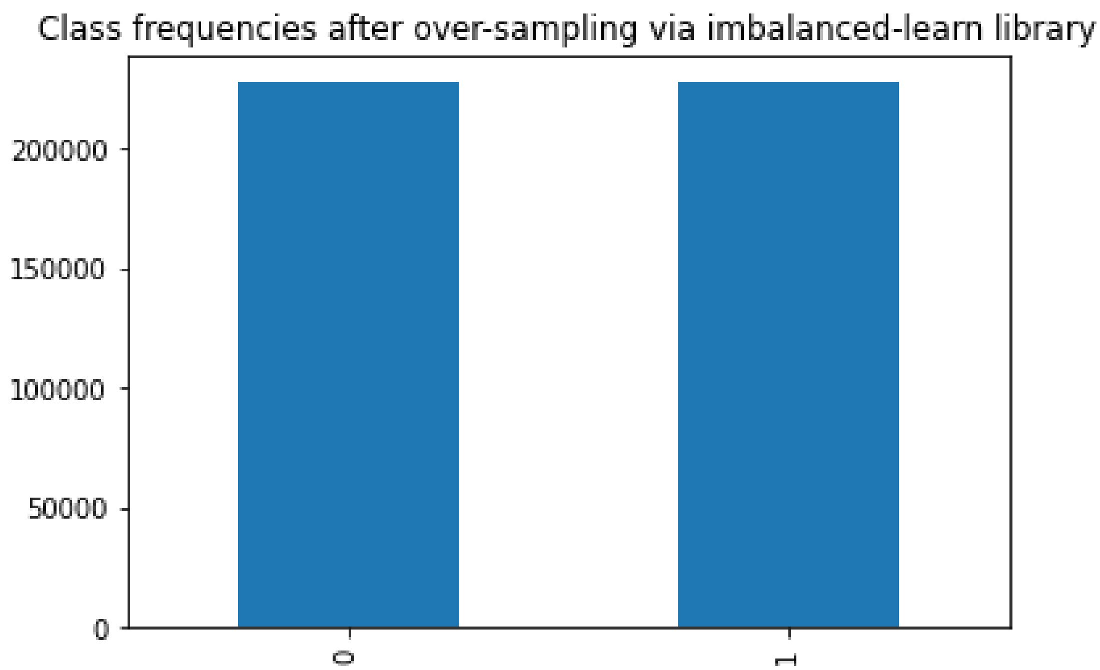


1.4.5 Random over-sampling with imbalanced-learn library (ROS-IL)

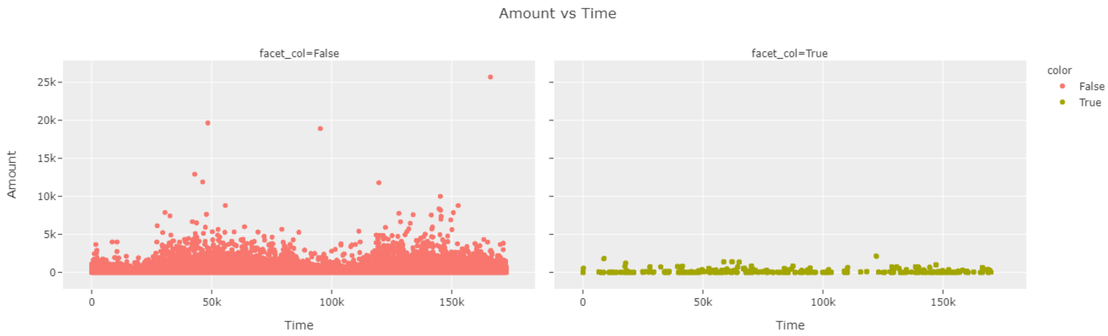
[16]:
```
imblearn_ros = RandomOverSampler(random_state = 40)
X_train_ros, y_train_ros = imblearn_ros.fit_resample(X_train, y_train)

X_train_ros = pd.DataFrame(X_train_ros)
X_train_ros.columns = features

y_train_ros = pd.DataFrame(y_train_ros)
y_train_ros.columns = ['Class']

data_train_over_imblearn = pd.concat([X_train_ros, y_train_ros], axis = 1)

X_train_over_imblearn = data_train_over_imblearn.drop('Class', axis = 1)
y_train_over_imblearn = data_train_over_imblearn['Class']

print('Class frequencies after over-sampling via imbalanced-learn library:')
print(y_train_over_imblearn.value_counts())
y_train_over_imblearn.value_counts().plot(kind = 'bar',
                                          title = 'Class frequencies after␣
↪over-sampling via imbalanced-learn library')
```

```
Class frequencies after over-sampling via imbalanced-learn library:
0    227465
1    227465
Name: Class, dtype: int64
```

[16]:
```
<AxesSubplot:title={'center':'Class frequencies after over-sampling via
imbalanced-learn library'}>
```



[17]:
```
# Amount vs Time for authentic and fraudulent transactions in the training set␣
↪after random over-sampling via imblearn

class_list = list(y_train_over_imblearn)
fraud_status = []
for i in range(len(class_list)):
    fraud_status.append(bool(class_list[i]))

fig1 = px.scatter(data_train_over_imblearn,
                  x = 'Time',
                  y = 'Amount',
                  facet_col = fraud_status,
                  color = fraud_status,
                  title = 'Amount vs Time',
                  template = 'ggplot2'
                 )
fig1.show()
```

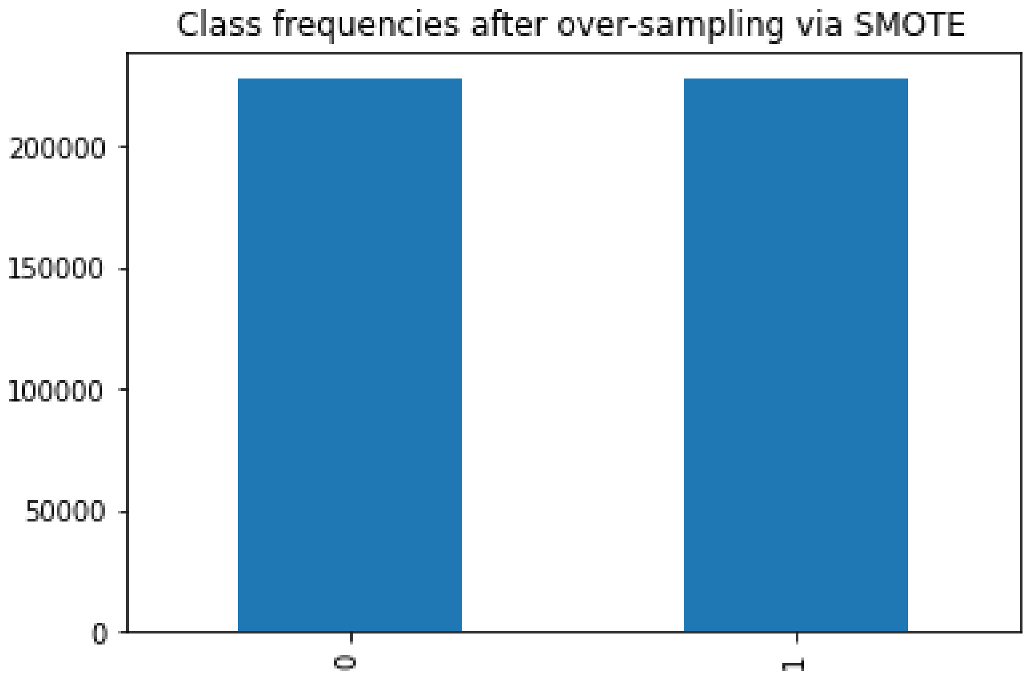


1.4.6 Synthetic minority over-sampling technique (SMOTE)

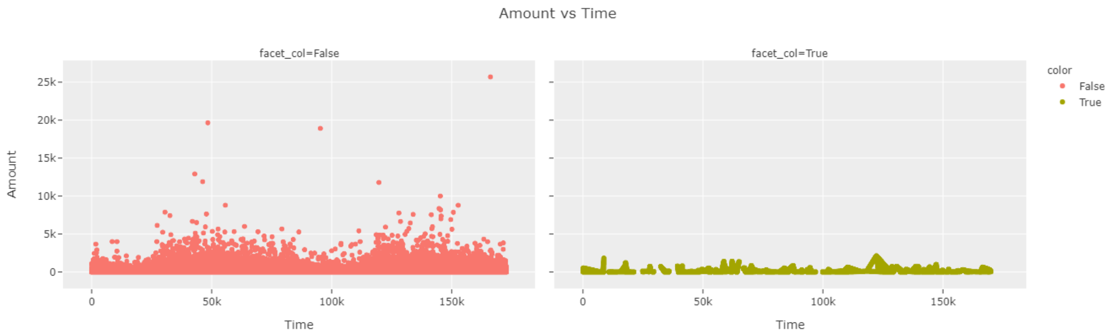
[18]:
```
smote = SMOTE()
X_train_smote, y_train_smote = smote.fit_resample(X_train, y_train)

X_train_smote = pd.DataFrame(X_train_smote)
X_train_smote.columns = features
X_train_smote

y_train_smote = pd.DataFrame(y_train_smote)
y_train_smote.columns = ['Class']
y_train_smote

data_train_over_smote = pd.concat([X_train_smote, y_train_smote], axis = 1)

X_train_over_smote = data_train_over_smote.drop('Class', axis = 1)
y_train_over_smote = data_train_over_smote['Class']

print('Class frequencies after over-sampling via SMOTE:')
print(y_train_over_smote.value_counts())
y_train_over_smote.value_counts().plot(kind = 'bar', title = 'Class frequencies␣
↪after over-sampling via SMOTE')
```

```
Class frequencies after over-sampling via SMOTE:
0    227465
1    227465
Name: Class, dtype: int64
```

[18]:
```
<AxesSubplot:title={'center':'Class frequencies after over-sampling via SMOTE'}>
```



[19]:
```
# Amount vs Time for authentic and fraudulent transactions in the training set␣
↪after over-sampling via SMOTE

class_list = list(y_train_over_smote)
fraud_status = []
for i in range(len(class_list)):
    fraud_status.append(bool(class_list[i]))

fig1 = px.scatter(data_train_over_smote,
                  x = 'Time',
                  y = 'Amount',
                  facet_col = fraud_status,
                  color = fraud_status,
                  title = 'Amount vs Time',
                  template = 'ggplot2'
                 )
fig1.show()
```

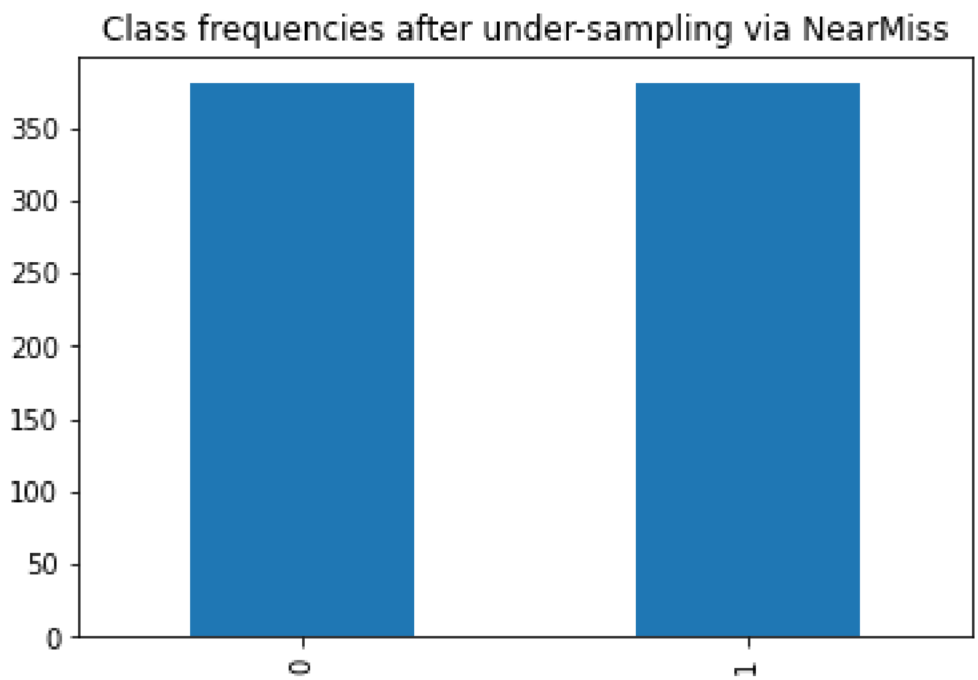


1.4.7 Under-sampling via NearMiss (NM)

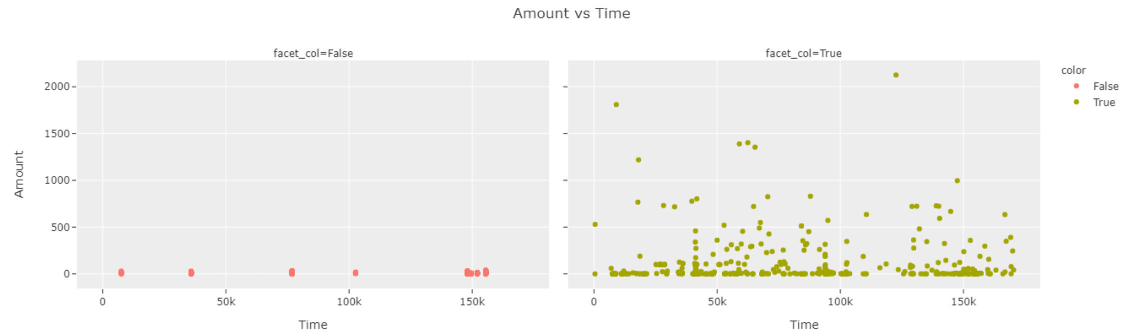
[20]:
```
nm = NearMiss()
X_train_nm, y_train_nm = nm.fit_resample(X_train, y_train)

X_train_nm = pd.DataFrame(X_train_nm)
X_train_nm.columns = features
X_train_nm

y_train_nm = pd.DataFrame(y_train_nm)
y_train_nm.columns = ['Class']
y_train_nm

data_train_under_nm = pd.concat([X_train_nm, y_train_nm], axis = 1)

X_train_under_nm = data_train_under_nm.drop('Class', axis = 1)
y_train_under_nm = data_train_under_nm['Class']

print('Class frequencies after under-sampling via NearMiss:')
print(y_train_under_nm.value_counts())
y_train_under_nm.value_counts().plot(kind = 'bar', title = 'Class frequencies␣
↪after under-sampling via NearMiss')
```

```
Class frequencies after under-sampling via NearMiss:
0    380
1    380
Name: Class, dtype: int64
```

[20]:
```
<AxesSubplot:title={'center':'Class frequencies after under-sampling via
NearMiss'}>
```



[21]:
```
# Amount vs Time for authentic and fraudulent transactions in the training set␣
↪after random under-sampling via NearMiss

class_list = list(y_train_under_nm)
fraud_status = []
for i in range(len(class_list)):
    fraud_status.append(bool(class_list[i]))

fig1 = px.scatter(data_train_under_nm,
                  x = 'Time',
                  y = 'Amount',
                  facet_col = fraud_status,
                  color = fraud_status,
                  title = 'Amount vs Time',
                  template = 'ggplot2'
                 )
fig1.show()
```



From the left plot it is clear that the majority class is not represented accurately in the undersampling scheme via NearMiss.

[22]:
```
TrainingSets = ['Unaltered', 'RUS', 'ROS', 'RUS-IL', 'ROS-IL', 'SMOTE', 'NM']
```

4. Feature Scaling

It may be natural for one of the features to contribute to the classification process more than another. But often this is caused artificially by the difference of range of values that the features take (often due to the units in which the features are measured). Many algorithms, especially the tree-based ones like decision tree and random forest, as well as graphical model-based classifiers like linear discriminant analysis and naive Bayes are invariant to scaling and hence are indifferent to feature scaling. On the other hand, the algorithms based on distances or similarities, which include k-nearest neighbours, support vector machine and stochastic gradient descent are sensitive to scaling. This necessitates the practitioner to scale the features appropriately before feeding the data to such classifiers.

[23]:
```
scaling = MinMaxScaler(feature_range = (-1,1)).fit(X_train)

X_train_scaled_minmax = scaling.transform(X_train)
X_train_under_scaled_minmax = scaling.transform(X_train_under)
X_train_over_scaled_minmax = scaling.transform(X_train_over)
X_train_under_imblearn_scaled_minmax = scaling.transform(X_train_under_imblearn)
X_train_over_imblearn_scaled_minmax = scaling.transform(X_train_over_imblearn)
X_train_over_smote_scaled_minmax = scaling.transform(X_train_over_smote)
X_train_under_nm_scaled_minmax = scaling.transform(X_train_under_nm)

X_test_scaled_minmax = scaling.transform(X_test)
```

5. Logistic Regression

[24]:
```
logreg = LogisticRegression(max_iter = 1000)
```

[25]:
```
# Computation of confusion matrix, evaluation metrics and visualization of␣
↪classes
```

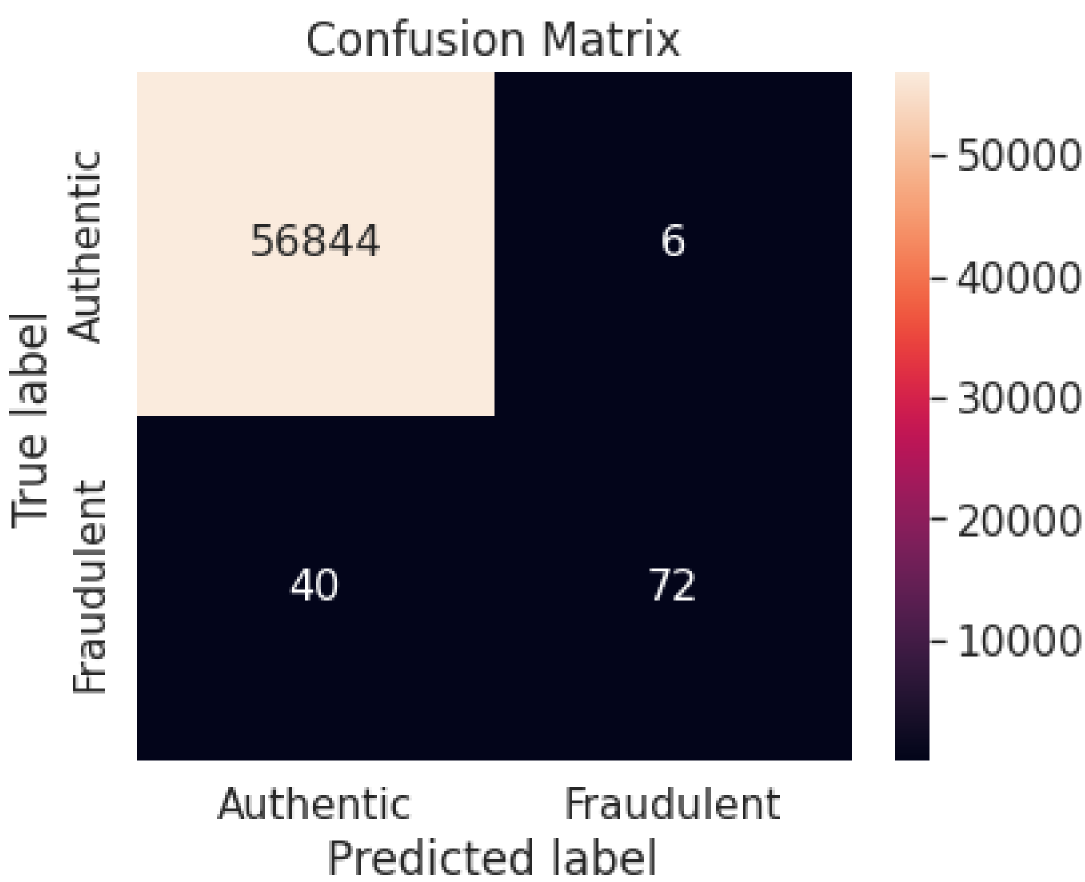
```python
def classification(model, X_train, y_train, X_test, y_test):
    model.fit(X_train, y_train)
    y_pred = model.predict(X_test)

    y_test = list(y_test)
    y_pred = list(y_pred)

    # Confusion matrix

    class_names = ['Authentic', 'Fraudulent']
    tick_marks_y = [0.25, 1.2]
    tick_marks_x = [0.5, 1.5]

    confusion_matrix = metrics.confusion_matrix(y_test, y_pred)
    confusion_matrix_df = pd.DataFrame(confusion_matrix, range(2), range(2))
    plt.figure(figsize = (6, 4.75))
    sns.set(font_scale = 1.4) # label size
    plt.title("Confusion Matrix")
    sns.heatmap(confusion_matrix_df, annot = True, annot_kws = {"size": 16}, fmt = 'd') # font size
    plt.yticks(tick_marks_y, class_names, rotation = 'vertical')
    plt.xticks(tick_marks_x, class_names, rotation = 'horizontal')
    plt.ylabel('True label')
    plt.xlabel('Predicted label')
    plt.grid(False)
    plt.show()

    # Evaluation metrics

    TN = confusion_matrix[0, 0]
    FP = confusion_matrix[0, 1]
    FN = confusion_matrix[1, 0]
    TP = confusion_matrix[1, 1]

    accuracy = (TP + TN)/(TP + FN + TN + FP)

    if (FP + TP == 0):
        precision = float('NaN')
    else:
        precision = TP/(TP + FP)

    if (TP + FN == 0):
        recall = float('NaN')
    else:
        recall = TP/(TP + FN)
```

```
    FM_index = np.sqrt(precision * recall) # Fowlkes-Mallows index

    if (TP == 0):
        F0_5_score = float('NaN')
        F1_score = float('NaN')
        F2_score = float('NaN')
    else:
        F0_5_score = (1.25 * precision * recall)/((0.25 * precision) + recall)
        F1_score = (2 * precision * recall)/(precision + recall)
        F2_score = (5 * precision * recall)/((4 * precision) + recall)

    if (TN + FP == 0):
        specificity = float('NaN')
    else:
        specificity = TN/(TN + FP)

    G_mean = np.sqrt(recall * specificity)

    MCC_num = (TN * TP) - (FP * FN)
    MCC_denom = np.sqrt((FP + TP) * (FN + TP) * (TN + FP) * (TN + FN))

    if (MCC_denom == 0):
        MCC = float('NaN')
    else:
        MCC = MCC_num / MCC_denom # Matthews Correlation Coefficient

    # Summary

    EvalMetricLabels = ['MCC', 'F1-Score', 'F2-Score', 'Recall', 'Precision',
                        'FM index', 'Specificity', 'G-mean', 'F0.5-Score', 'Accuracy']
    EvalMetricValues = [MCC, F1_score, F2_score, recall, precision, FM_index, specificity, G_mean, F0_5_score, accuracy]

    global summary
    summary = pd.DataFrame(columns = ['Metric', 'Performance score'])
    summary['Metric'] = EvalMetricLabels
    summary['Performance score'] = EvalMetricValues

    # Performance of the model through confusion matrix

    fig1 = make_subplots(rows = 1, cols = 2, specs = [[{"type": "pie"}, {"type": "pie"}]])

    fig1.add_trace(go.Pie(
        labels = ['TP', 'FN'],
        values = [TP, FN],
```

```
        domain = dict(x = [0, 0.4]),
        name = 'Positive Class'),
        row = 1, col = 1)

    fig1.add_trace(go.Pie(
        labels = ['TN', 'FP'],
        values = [TN, FP],
        domain = dict(x = [0.4, 0.8]),
        name = 'Negative Class'),
        row = 1, col = 2)

    fig1.update_layout(height = 450, showlegend = True)
    fig1.show()
```

1.5 Unaltered training set

[26]:
```
# Elements of confusion matrix

classification(logreg, X_train, y_train, X_test, y_test)

# Summary of evaluation metrics

summary_logreg_unaltered = summary.copy()
summary_logreg_unaltered.set_index('Metric')

y_score = logreg.decision_function(X_test)
average_precision = average_precision_score(y_test, y_score)
y_pred_proba = logreg.predict_proba(X_test)[::,1]
roc_auc = metrics.roc_auc_score(y_test, y_pred_proba)

summary_logreg_unaltered_extended = summary.copy()
summary_logreg_unaltered_extended.loc[len(summary_logreg_unaltered_extended.
 ↪index)] = ['AP', average_precision]
summary_logreg_unaltered_extended.loc[len(summary_logreg_unaltered_extended.
 ↪index)] = ['ROC-AUC', roc_auc]
summary_logreg_unaltered_extended.set_index('Metric')

summary_logreg_unaltered_index = summary_logreg_unaltered_extended.T
summary_logreg_unaltered_index.columns = summary_logreg_unaltered_index.iloc[0]
summary_logreg_unaltered_index.drop(summary_logreg_unaltered_index.index[0],␣
 ↪inplace = True)
summary_logreg_unaltered_index
```



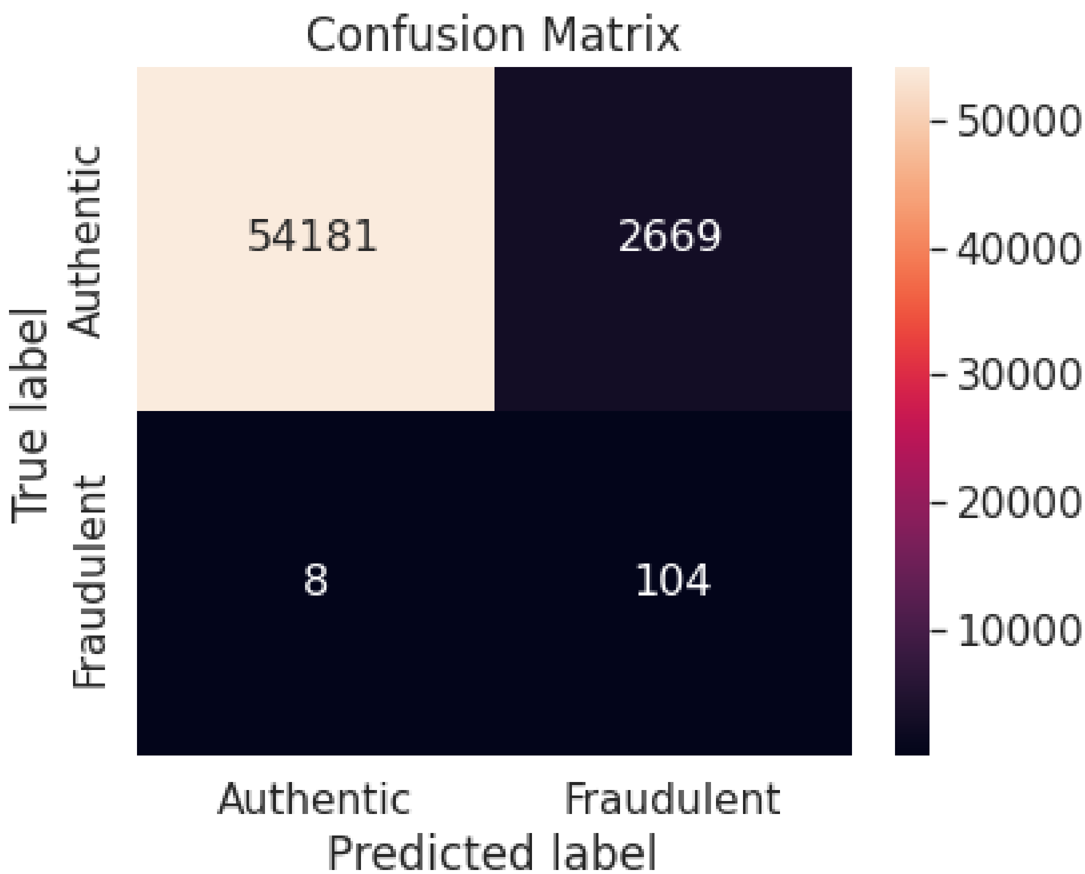

```
[26]: Metric                      MCC  F1-Score  F2-Score    Recall Precision  FM index  \
      Performance score  0.769972  0.757895  0.684411  0.642857  0.923077  0.770329

      Metric             Specificity    G-mean F0.5-Score  Accuracy        AP  \
      Performance score     0.999894  0.801741    0.849057  0.999192  0.735535
```

```
Metric               ROC-AUC
Performance score  0.959767
```

Observation: While logistic regression model on unaltered training set performs exceedingly well on the negative class (authentic transactions), it does not work so well with the critical positive class (fraudulent transactions) as it misclassifies more than one-third of the transactions in that class.

1.6 Random under-sampling

[27]:
```
# Elements of confusion matrix

classification(logreg, X_train_under, y_train_under, X_test, y_test)

# Summary of evaluation metrics

summary_logreg_under = summary
summary_logreg_under.set_index('Metric')

y_score = logreg.decision_function(X_test)
average_precision = average_precision_score(y_test, y_score)
y_pred_proba = logreg.predict_proba(X_test)[::,1]
roc_auc = metrics.roc_auc_score(y_test, y_pred_proba)

summary_logreg_under_extended = summary.copy()
summary_logreg_under_extended.loc[len(summary_logreg_under_extended.index)] =␣
 ↪['AP', average_precision]
summary_logreg_under_extended.loc[len(summary_logreg_under_extended.index)] =␣
 ↪['ROC-AUC', roc_auc]
summary_logreg_under_extended.set_index('Metric')

summary_logreg_under_index = summary_logreg_under_extended.T
summary_logreg_under_index.columns = summary_logreg_under_index.iloc[0]
summary_logreg_under_index.drop(summary_logreg_under_index.index[0], inplace =␣
 ↪True)
summary_logreg_under_index
```

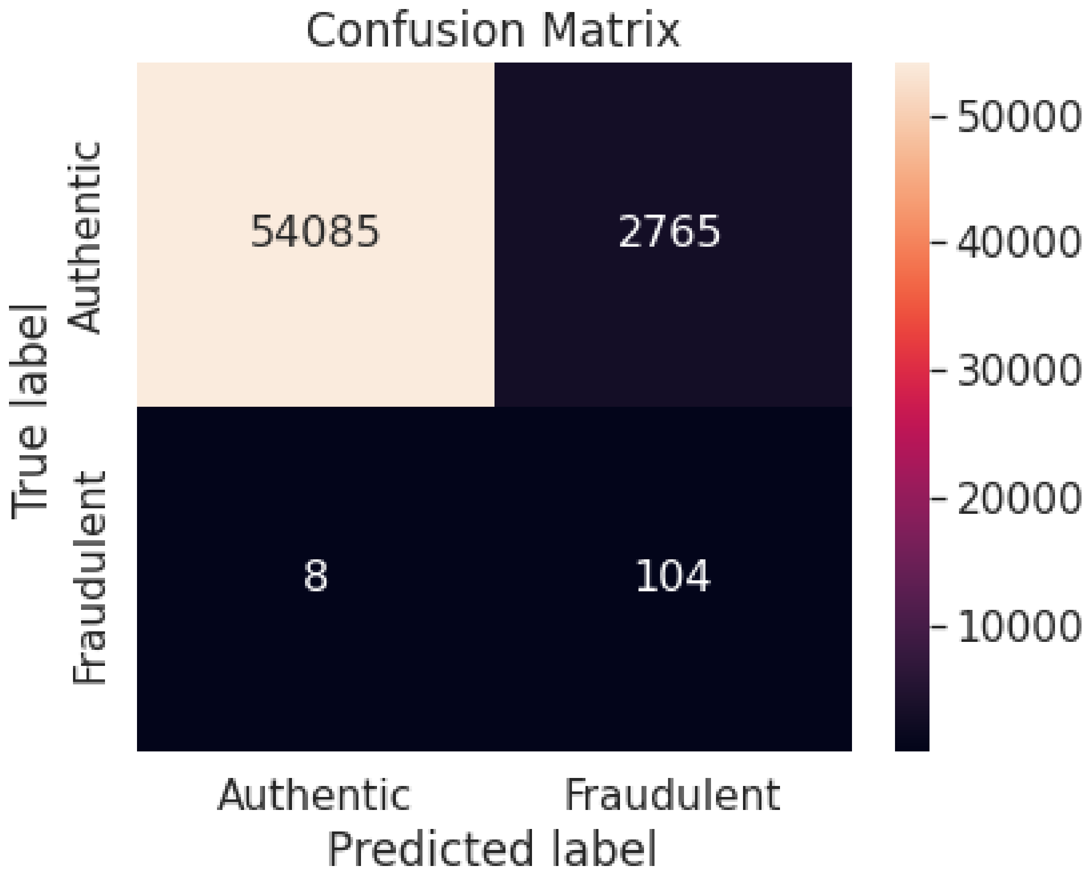


[27]:
```
Metric                  MCC  F1-Score  F2-Score    Recall Precision  FM index  \
Performance score  0.181479  0.072097  0.161441  0.928571  0.037505  0.186616

Metric             Specificity    G-mean F0.5-Score  Accuracy       AP  \
Performance score     0.953052  0.940732   0.046412  0.953004  0.79466
```

```
Metric              ROC-AUC
Performance score  0.978047
```

1.7 Random over-sampling

[28]:
```
# Elements of confusion matrix

classification(logreg, X_train_over, y_train_over, X_test, y_test)

# Summary of evaluation metrics

summary_logreg_over = summary
summary_logreg_over.set_index('Metric')

y_score = logreg.decision_function(X_test)
average_precision = average_precision_score(y_test, y_score)
y_pred_proba = logreg.predict_proba(X_test)[::,1]
roc_auc = metrics.roc_auc_score(y_test, y_pred_proba)

summary_logreg_over_extended = summary.copy()
summary_logreg_over_extended.loc[len(summary_logreg_over_extended.index)] =␣
 ↪['AP', average_precision]
summary_logreg_over_extended.loc[len(summary_logreg_over_extended.index)] =␣
 ↪['ROC-AUC', roc_auc]
summary_logreg_over_extended.set_index('Metric')

summary_logreg_over_index = summary_logreg_over_extended.T
summary_logreg_over_index.columns = summary_logreg_over_index.iloc[0]
summary_logreg_over_index.drop(summary_logreg_over_index.index[0], inplace =␣
 ↪True)
summary_logreg_over_index
```



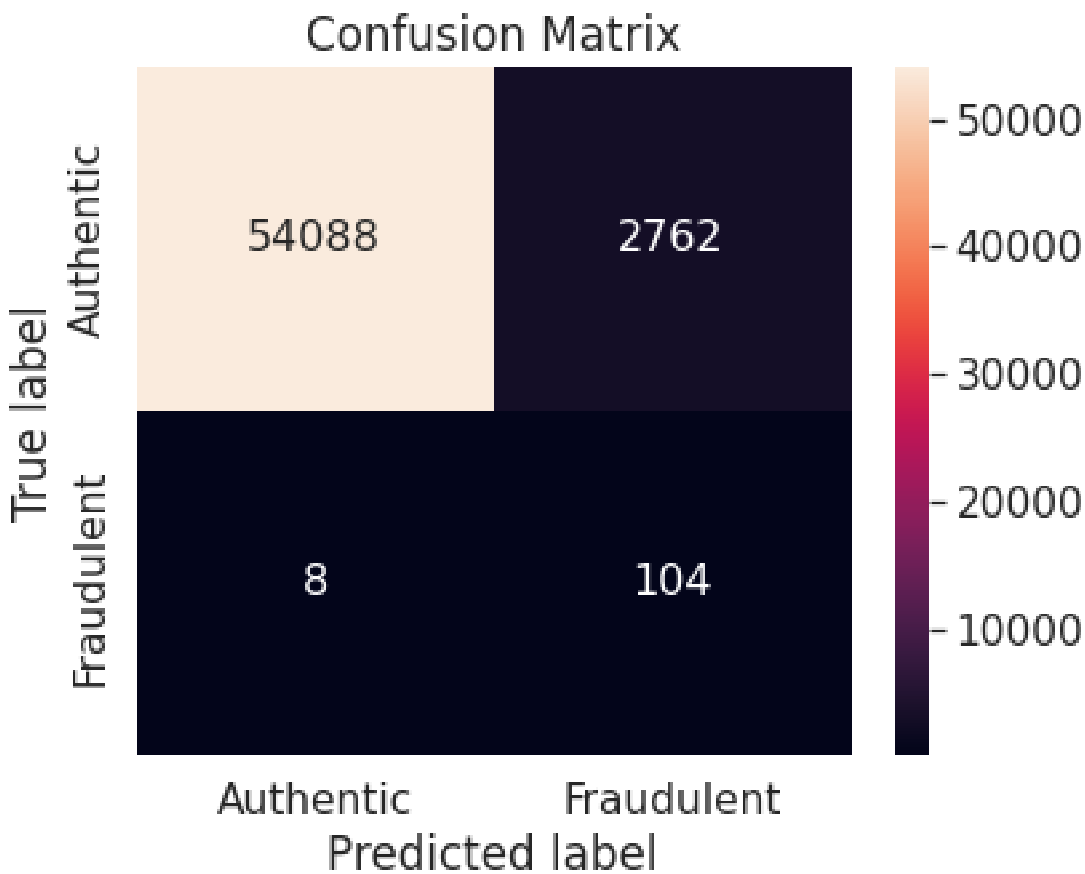
[28]:

```
Metric                  MCC  F1-Score  F2-Score    Recall Precision  FM index  \
Performance score  0.178233  0.069775  0.156768  0.928571   0.03625  0.183467

Metric             Specificity    G-mean F0.5-Score  Accuracy        AP  \
Performance score     0.951363  0.939898   0.044874  0.951318  0.749124
```

```
Metric                ROC-AUC
Performance score   0.971644
```

1.8 Random under-sampling with imbalanced-learn library

[29]:
```
# Elements of confusion matrix

classification(logreg, X_train_under_imblearn, y_train_under_imblearn, X_test,␣
 ↪y_test)

# Summary of evaluation metrics

summary_logreg_under_imblearn = summary
summary_logreg_under_imblearn.set_index('Metric')

y_score = logreg.decision_function(X_test)
average_precision = average_precision_score(y_test, y_score)
y_pred_proba = logreg.predict_proba(X_test)[::,1]
roc_auc = metrics.roc_auc_score(y_test, y_pred_proba)

summary_logreg_under_imblearn_extended = summary.copy()
summary_logreg_under_imblearn_extended.
 ↪loc[len(summary_logreg_under_imblearn_extended.index)] = ['AP',␣
 ↪average_precision]
summary_logreg_under_imblearn_extended.
 ↪loc[len(summary_logreg_under_imblearn_extended.index)] = ['ROC-AUC', roc_auc]
summary_logreg_under_imblearn_extended.set_index('Metric')

summary_logreg_under_imblearn_index = summary_logreg_under_imblearn_extended.T
summary_logreg_under_imblearn_index.columns =␣
 ↪summary_logreg_under_imblearn_index.iloc[0]
summary_logreg_under_imblearn_index.drop(summary_logreg_under_imblearn_index.
 ↪index[0], inplace = True)
summary_logreg_under_imblearn_index
```



[29]:

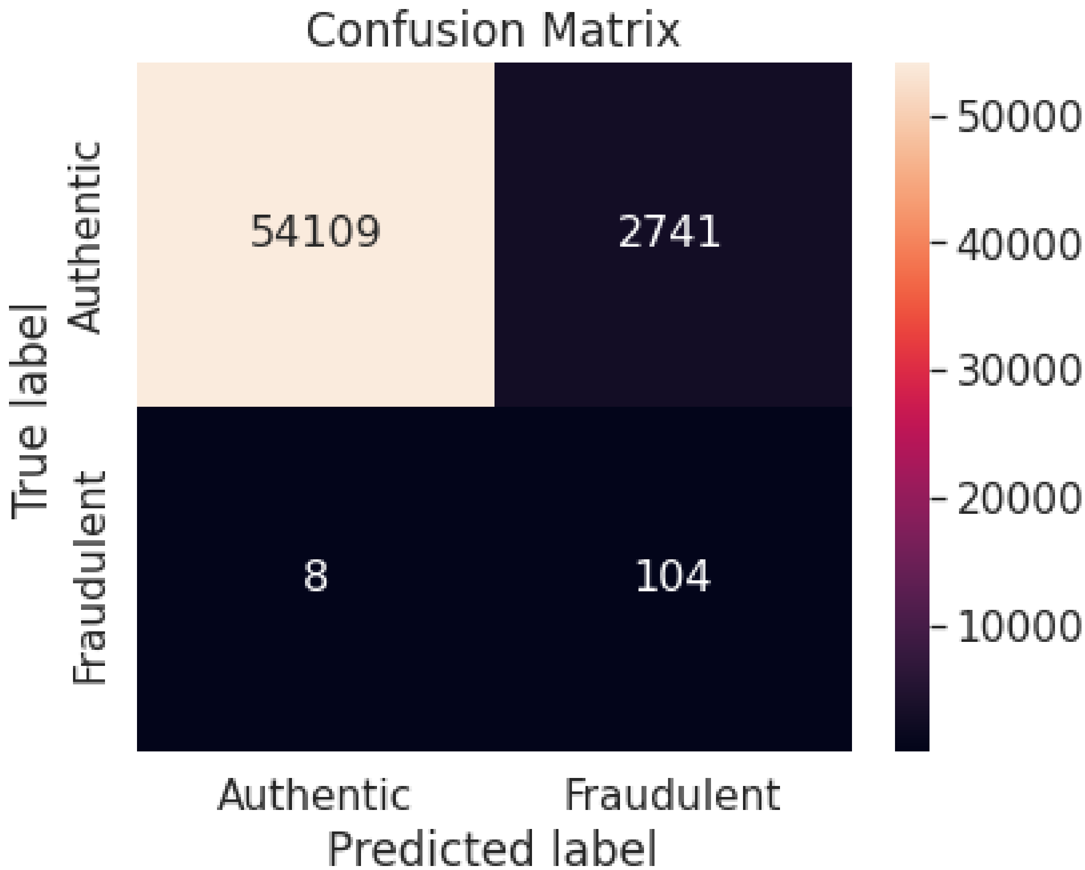
Metric	MCC	F1-Score	F2-Score	Recall	Precision	FM index
Performance score	0.178332	0.069846	0.15691	0.928571	0.036288	0.183563

Metric	Specificity	G-mean	F0.5-Score	Accuracy	AP
Performance score	0.951416	0.939924	0.044921	0.951371	0.767477

```
Metric                ROC-AUC
Performance score   0.980058
```

1.9 Random over-sampling with imbalanced-learn library

[30]:
```
# Elements of confusion matrix

classification(logreg, X_train_over_imblearn, y_train_over_imblearn, X_test,␣
↪y_test)

# Summary of evaluation metrics

summary_logreg_over_imblearn = summary
summary_logreg_over_imblearn.set_index('Metric')

y_score = logreg.decision_function(X_test)
average_precision = average_precision_score(y_test, y_score)
y_pred_proba = logreg.predict_proba(X_test)[::,1]
roc_auc = metrics.roc_auc_score(y_test, y_pred_proba)

summary_logreg_over_imblearn_extended = summary.copy()
summary_logreg_over_imblearn_extended.
↪loc[len(summary_logreg_over_imblearn_extended.index)] = ['AP',␣
↪average_precision]
summary_logreg_over_imblearn_extended.
↪loc[len(summary_logreg_over_imblearn_extended.index)] = ['ROC-AUC', roc_auc]
summary_logreg_over_imblearn_extended.set_index('Metric')

summary_logreg_over_imblearn_index = summary_logreg_over_imblearn_extended.T
summary_logreg_over_imblearn_index.columns = summary_logreg_over_imblearn_index.
↪iloc[0]
summary_logreg_over_imblearn_index.drop(summary_logreg_over_imblearn_index.
↪index[0], inplace = True)
summary_logreg_over_imblearn_index
```

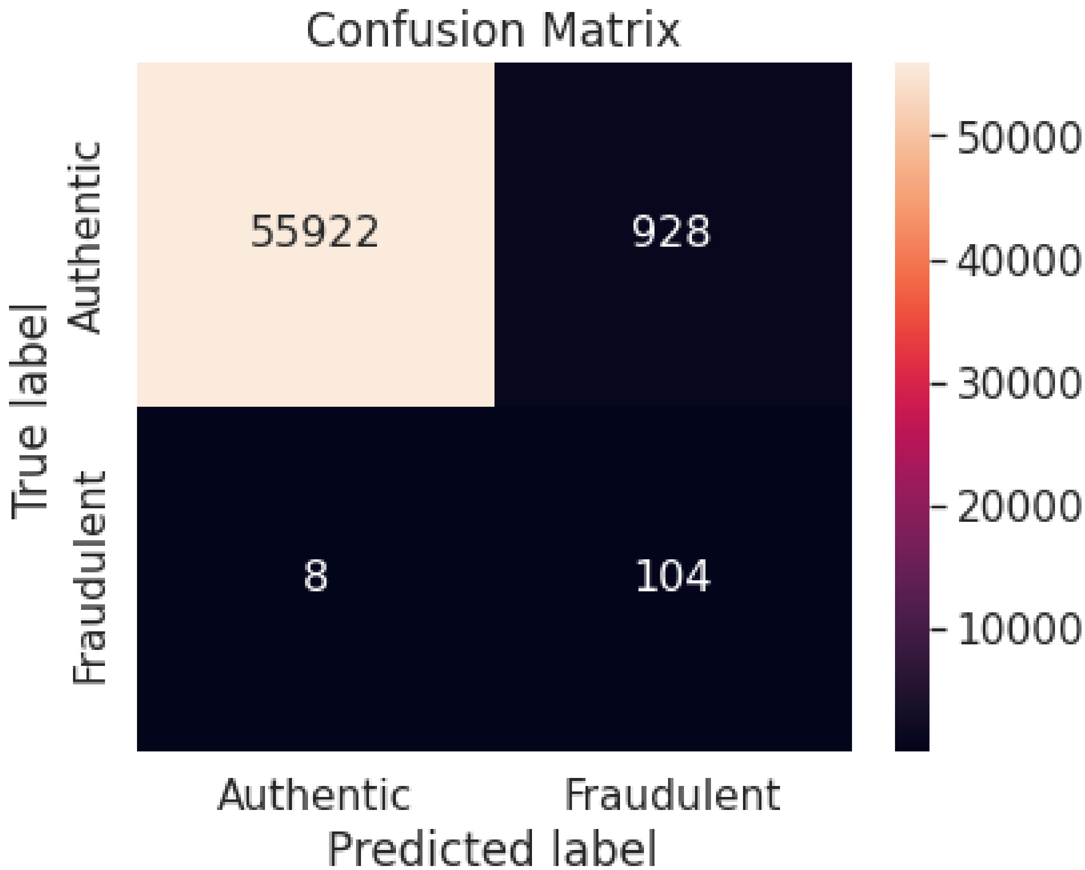


[30]:
```
Metric                  MCC  F1-Score  F2-Score    Recall Precision FM index  \
Performance score   0.17903  0.070342  0.157911  0.928571  0.036555  0.18424

Metric             Specificity    G-mean F0.5-Score Accuracy         AP  \
Performance score    0.951785  0.940107   0.045249  0.95174  0.749028
```

```
Metric              ROC-AUC
Performance score  0.971542
```

1.10 Synthetic minority over-sampling technique (SMOTE)

[31]:
```
# Elements of confusion matrix

classification(logreg, X_train_over_smote, y_train_over_smote, X_test, y_test)

# Summary of evaluation metrics

summary_logreg_over_smote = summary
summary_logreg_over_smote.set_index('Metric')

y_score = logreg.decision_function(X_test)
average_precision = average_precision_score(y_test, y_score)
y_pred_proba = logreg.predict_proba(X_test)[::,1]
roc_auc = metrics.roc_auc_score(y_test, y_pred_proba)

summary_logreg_over_smote_extended = summary.copy()
summary_logreg_over_smote_extended.loc[len(summary_logreg_over_smote_extended.
 ↪index)] = ['AP', average_precision]
summary_logreg_over_smote_extended.loc[len(summary_logreg_over_smote_extended.
 ↪index)] = ['ROC-AUC', roc_auc]
summary_logreg_over_smote_extended.set_index('Metric')

summary_logreg_over_smote_index = summary_logreg_over_smote_extended.T
summary_logreg_over_smote_index.columns = summary_logreg_over_smote_index.
 ↪iloc[0]
summary_logreg_over_smote_index.drop(summary_logreg_over_smote_index.index[0],␣
 ↪inplace = True)
summary_logreg_over_smote_index
```

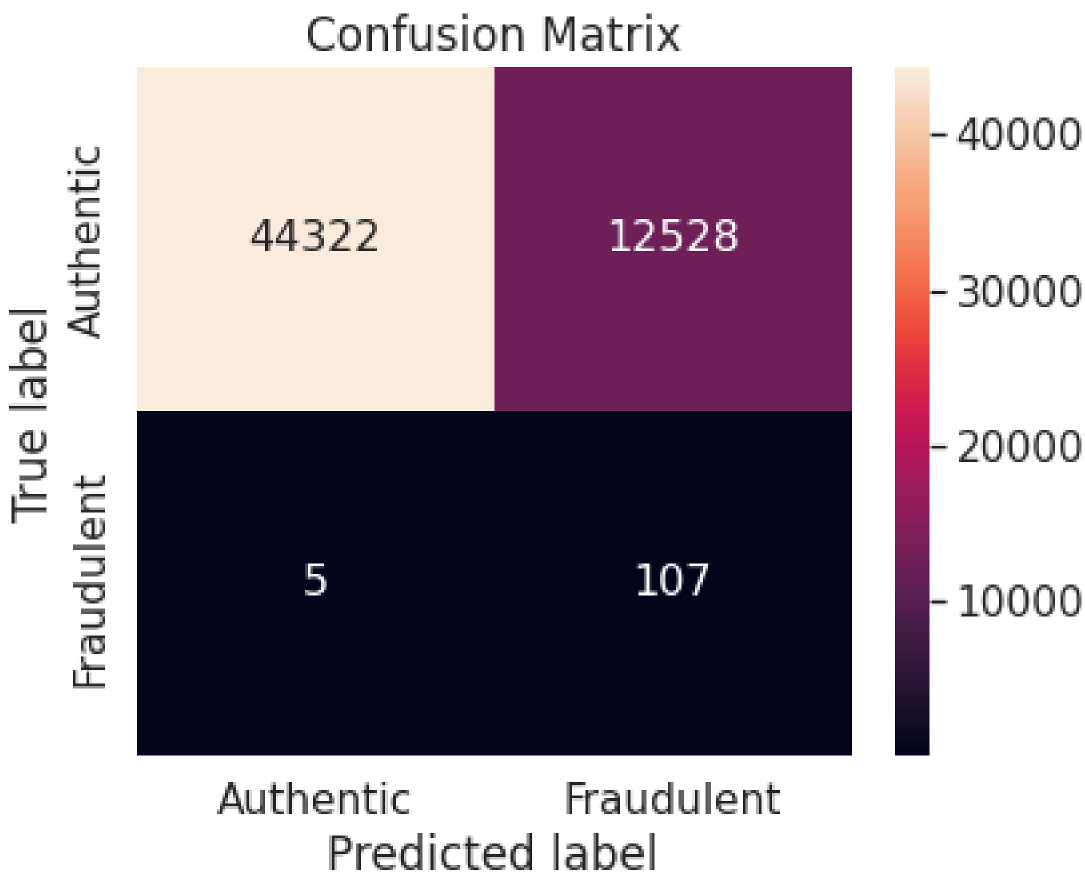


[31]:
```
Metric                  MCC  F1-Score  F2-Score    Recall Precision  FM index  \
Performance score  0.302988  0.181818  0.351351  0.928571  0.100775  0.305904

Metric             Specificity    G-mean F0.5-Score  Accuracy       AP  \
Performance score     0.983676  0.955727   0.122642  0.983568  0.82205
```

```
Metric                ROC-AUC
Performance score   0.978506
```

1.11 Under-sampling via NearMiss

[32]:
```
# Elements of confusion matrix

classification(logreg, X_train_under_nm, y_train_under_nm, X_test, y_test)

# Summary of evaluation metrics

summary_logreg_under_nm = summary
summary_logreg_under_nm.set_index('Metric')

y_score = logreg.decision_function(X_test)
average_precision = average_precision_score(y_test, y_score)
y_pred_proba = logreg.predict_proba(X_test)[::,1]
roc_auc = metrics.roc_auc_score(y_test, y_pred_proba)

summary_logreg_under_nm_extended = summary.copy()
summary_logreg_under_nm_extended.loc[len(summary_logreg_under_nm_extended.
 ↪index)] = ['AP', average_precision]
summary_logreg_under_nm_extended.loc[len(summary_logreg_under_nm_extended.
 ↪index)] = ['ROC-AUC', roc_auc]
summary_logreg_under_nm_extended.set_index('Metric')

summary_logreg_under_nm_index = summary_logreg_under_nm_extended.T
summary_logreg_under_nm_index.columns = summary_logreg_under_nm_index.iloc[0]
summary_logreg_under_nm_index.drop(summary_logreg_under_nm_index.index[0],␣
 ↪inplace = True)
summary_logreg_under_nm_index
```



[32]:
```
Metric                  MCC  F1-Score  F2-Score    Recall Precision  FM index  \
Performance score  0.078367  0.016788  0.040893  0.955357  0.008469  0.089947

Metric             Specificity    G-mean F0.5-Score  Accuracy        AP  \
Performance score     0.779631  0.863033   0.010562  0.779976  0.157375
```

```
Metric                  ROC-AUC
Performance score   0.963928
```

1.12 Summary of logistic regression models

Keeping in mind that the dataset is highly imbalanced and that the positive class (fraudulent transactions) is more important than the negative class (authentic transactions), we report **MCC**, **F1-Score**, **F2-Score** and **Recall** for each model considered. Additionally we report **Precision**, **FM index**, **Accuracy** and **Specificity**.

[33]:
```
summary_logreg = pd.DataFrame(columns = ['Metric'])

summary_logreg['Metric'] = EvalMetricLabels
summary_logreg_list = [summary_logreg_unaltered, summary_logreg_under,␣
 ↪summary_logreg_over, summary_logreg_under_imblearn,
                       summary_logreg_over_imblearn, summary_logreg_over_smote,␣
 ↪summary_logreg_under_nm]

for i in summary_logreg_list:
    summary_logreg = pd.merge(summary_logreg, i, on = 'Metric')

TrainingSetsMetric = TrainingSets.copy()
TrainingSetsMetric.insert(0, 'Metric')

summary_logreg.columns = TrainingSetsMetric
summary_logreg.set_index('Metric', inplace = True)
summary_logreg
```

[33]:
```
             Unaltered       RUS       ROS    RUS-IL    ROS-IL     SMOTE  \
Metric
MCC           0.769972  0.181479  0.178233  0.178332  0.179030  0.302988
F1-Score      0.757895  0.072097  0.069775  0.069846  0.070342  0.181818
F2-Score      0.684411  0.161441  0.156768  0.156910  0.157911  0.351351
Recall        0.642857  0.928571  0.928571  0.928571  0.928571  0.928571
Precision     0.923077  0.037505  0.036250  0.036288  0.036555  0.100775
FM index      0.770329  0.186616  0.183467  0.183563  0.184240  0.305904
Specificity   0.999894  0.953052  0.951363  0.951416  0.951785  0.983676
G-mean        0.801741  0.940732  0.939898  0.939924  0.940107  0.955727
F0.5-Score    0.849057  0.046412  0.044874  0.044921  0.045249  0.122642
Accuracy      0.999192  0.953004  0.951318  0.951371  0.951740  0.983568

                   NM
Metric
MCC          0.078367
F1-Score     0.016788
F2-Score     0.040893
```

```
Recall         0.955357
Precision      0.008469
FM index       0.089947
Specificity    0.779631
G-mean         0.863033
F0.5-Score     0.010562
Accuracy       0.779976
```

[34]:

```
# Function to visually compare performances of the model applied on different
↪training sets through evaluation metrics

def summary_visual(summary_model):
  fig1 = make_subplots(rows = 2, cols = 4, shared_yaxes = True, subplot_titles
↪= EvalMetricLabels)

  fig1.add_trace(go.Bar(x = list(summary_model.columns), y = list(summary_model.
↪loc['MCC'])), 1, 1)
  fig1.add_trace(go.Bar(x = list(summary_model.columns), y = list(summary_model.
↪loc['F1-Score'])), 1, 2)
  fig1.add_trace(go.Bar(x = list(summary_model.columns), y = list(summary_model.
↪loc['F2-Score'])), 1, 3)
  fig1.add_trace(go.Bar(x = list(summary_model.columns), y = list(summary_model.
↪loc['Recall'])), 1, 4)
  fig1.add_trace(go.Bar(x = list(summary_model.columns), y = list(summary_model.
↪loc['Precision'])), 2, 1)
  fig1.add_trace(go.Bar(x = list(summary_model.columns), y = list(summary_model.
↪loc['FM index'])), 2, 2)
  fig1.add_trace(go.Bar(x = list(summary_model.columns), y = list(summary_model.
↪loc['Specificity'])), 2, 3)
  fig1.add_trace(go.Bar(x = list(summary_model.columns), y = list(summary_model.
↪loc['Accuracy'])), 2, 4)

  fig1.update_layout(height = 600, width = 1000, coloraxis = dict(colorscale =
↪'Bluered_r'), showlegend = False)
  fig1.show()
```

[35]:

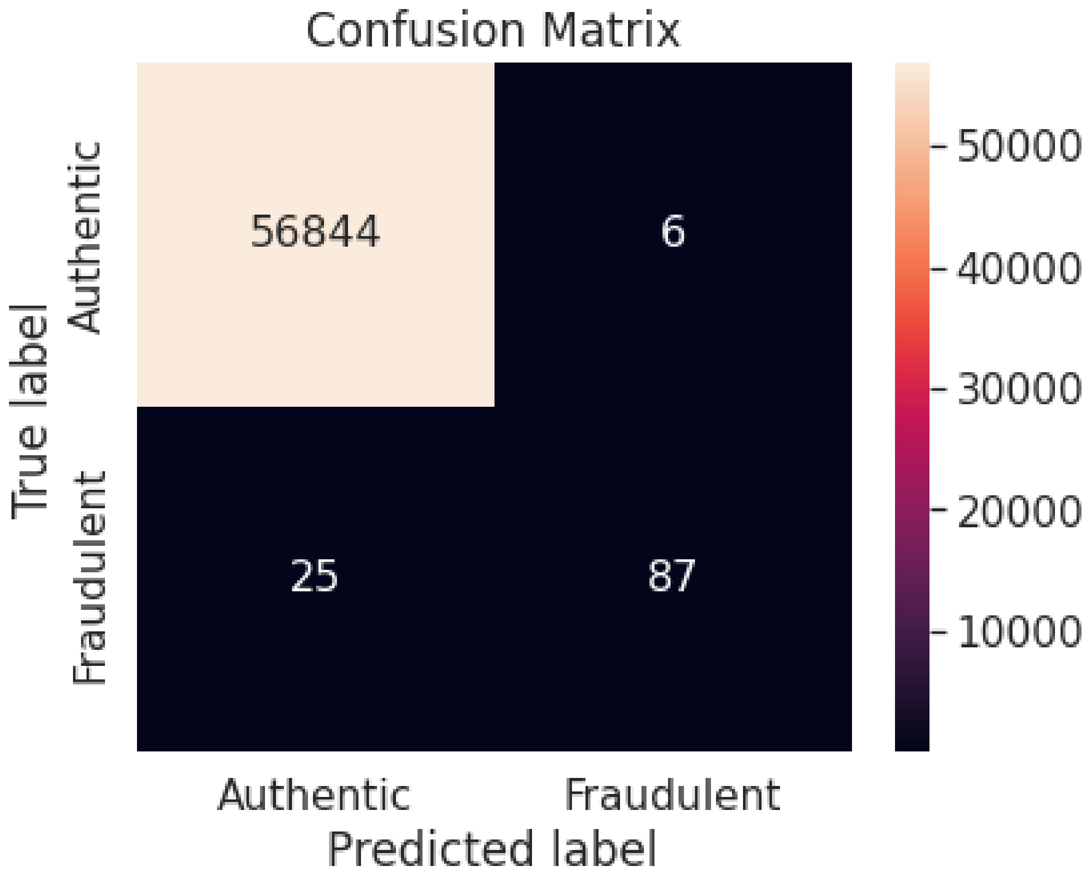
```
# Visual comparison of the model applied on different training sets through
↪evaluation metrics

summary_visual(summary_logreg)
```

6. k-Nearest Neighbors (k-NN)

[36]:
```
k = 29
knn = KNeighborsClassifier(n_neighbors = k, n_jobs = -1)
```

1.13 Unaltered training set

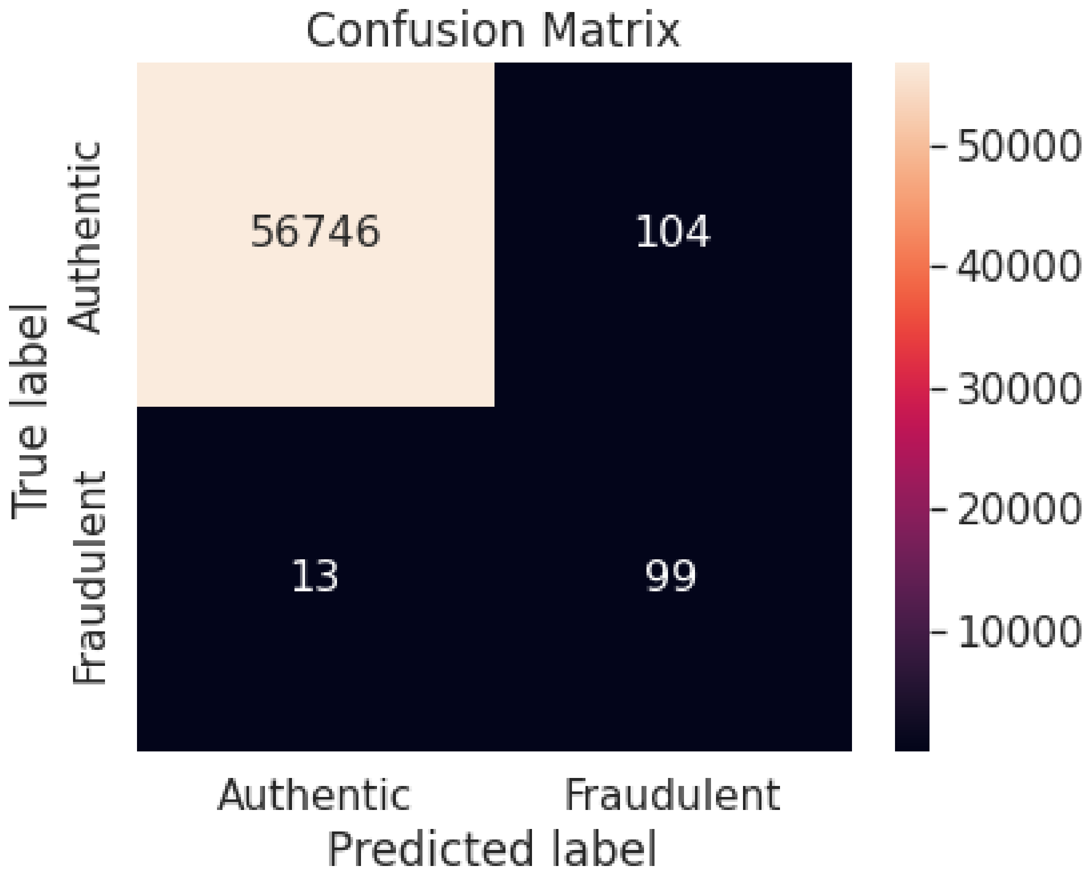
[37]:
```
# Elements of confusion matrix

classification(knn, X_train_scaled_minmax, y_train, X_test_scaled_minmax,␣
↪y_test)

# Summary of evaluation metrics

summary_knn_unaltered = summary
summary_knn_unaltered.set_index('Metric')

y_pred_proba = knn.predict_proba(X_test)[::,1]
roc_auc = metrics.roc_auc_score(y_test, y_pred_proba)

summary_knn_unaltered_extended = summary.copy()
summary_knn_unaltered_extended.loc[len(summary_knn_unaltered_extended.index)] =␣
↪['ROC-AUC', roc_auc]
summary_knn_unaltered_extended.set_index('Metric')

summary_knn_unaltered_index = summary_knn_unaltered_extended.T
summary_knn_unaltered_index.columns = summary_knn_unaltered_index.iloc[0]
summary_knn_unaltered_index.drop(summary_knn_unaltered_index.index[0], inplace␣
↪= True)
summary_knn_unaltered_index
```



[37]:
```
Metric                      MCC  F1-Score   F2-Score    Recall  Precision  FM index  \
Performance score  0.852191   0.84878   0.804067  0.776786   0.935484   0.85245

Metric             Specificity    G-mean  F0.5-Score  Accuracy  ROC-AUC
Performance score     0.999894  0.881308     0.89876  0.999456      0.5
```

1.14 Random under-sampling

[38]:
```
# Elements of confusion matrix

classification(knn, X_train_under_scaled_minmax, y_train_under,␣
↪X_test_scaled_minmax, y_test)

# Summary of evaluation metrics

summary_knn_under = summary
summary_knn_under.set_index('Metric')

y_pred_proba = knn.predict_proba(X_test)[::,1]
roc_auc = metrics.roc_auc_score(y_test, y_pred_proba)

summary_knn_under_extended = summary.copy()
summary_knn_under_extended.loc[len(summary_knn_under_extended.index)] =␣
↪['ROC-AUC', roc_auc]
summary_knn_under_extended.set_index('Metric')

summary_knn_under_index = summary_knn_under_extended.T
summary_knn_under_index.columns = summary_knn_under_index.iloc[0]
summary_knn_under_index.drop(summary_knn_under_index.index[0], inplace = True)
summary_knn_under_index
```

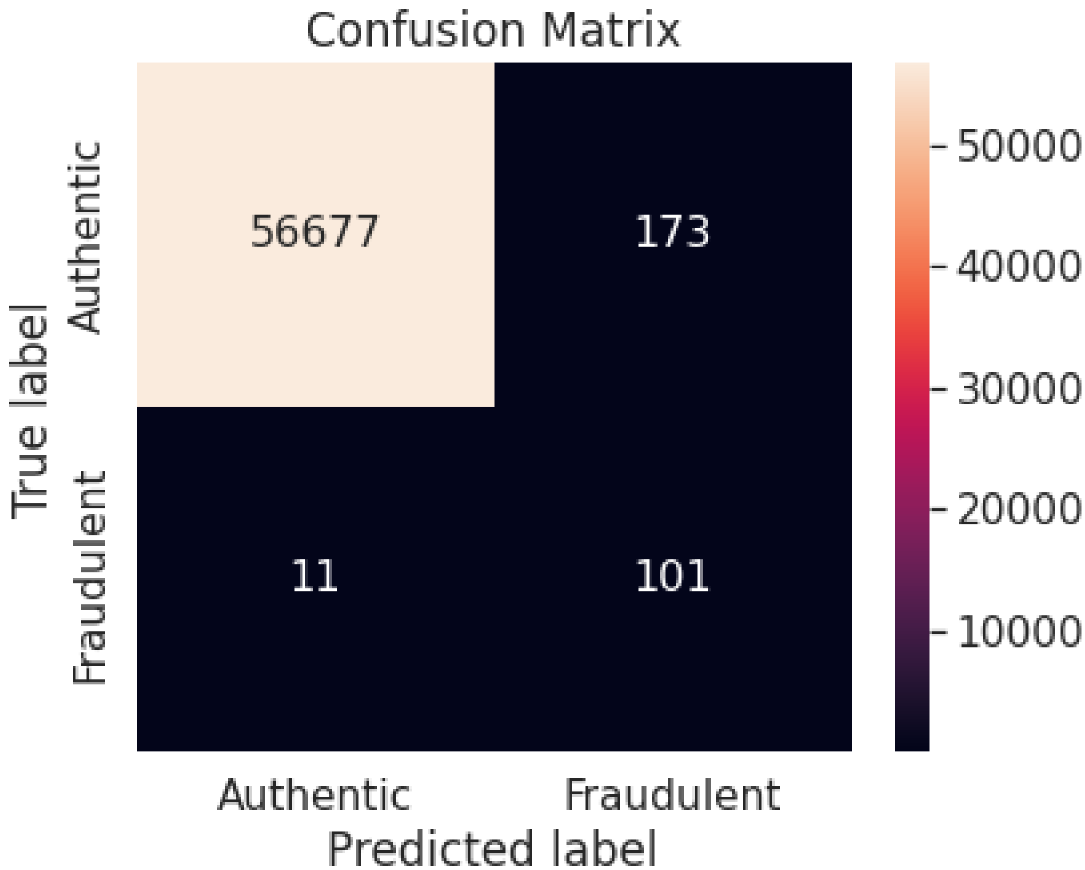


[38]:
```
Metric                    MCC  F1-Score  F2-Score    Recall Precision  FM index  \
Performance score  0.655732  0.628571  0.760369  0.883929  0.487685  0.656566

Metric            Specificity    G-mean F0.5-Score  Accuracy  ROC-AUC
Performance score    0.998171  0.939314   0.535714  0.997946  0.49978
```

1.15 Random over-sampling

[39]:

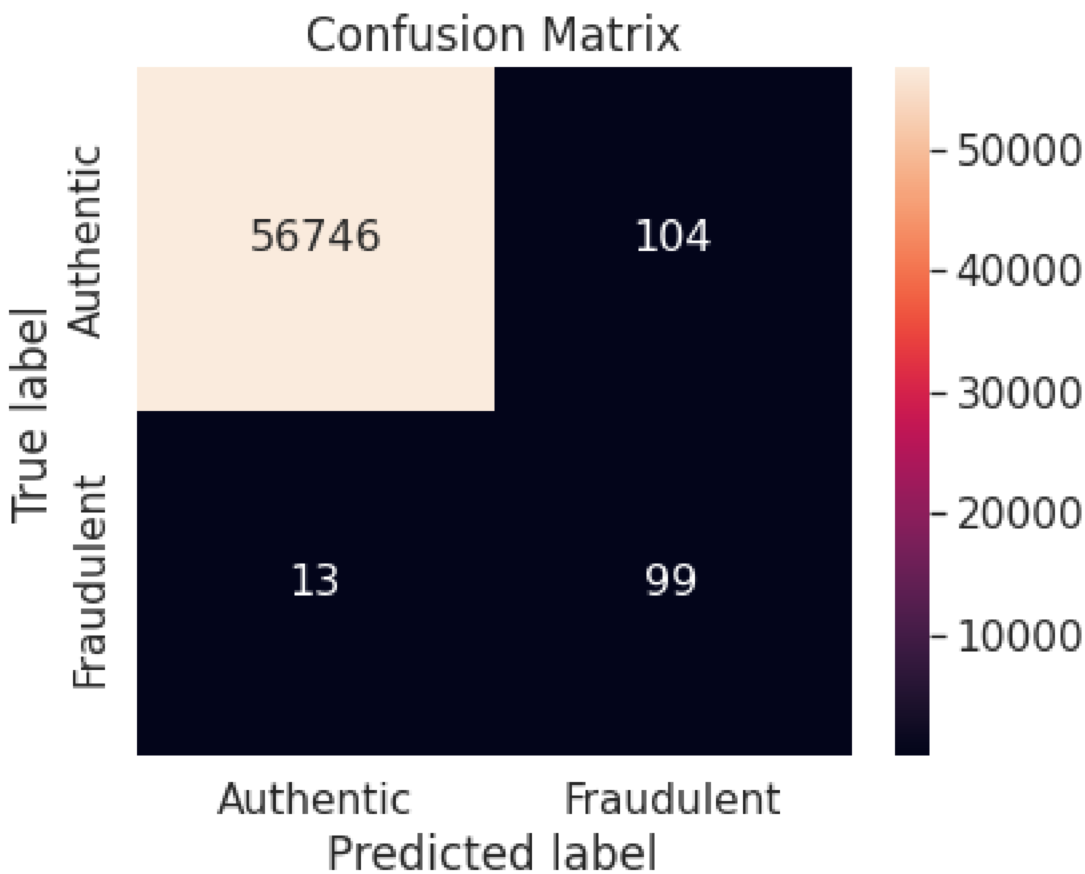
```
# Elements of confusion matrix

classification(knn, X_train_over_scaled_minmax, y_train_over,␣
 ↪X_test_scaled_minmax, y_test)

# Summary of evaluation metrics

summary_knn_over = summary
summary_knn_over.set_index('Metric')

y_pred_proba = knn.predict_proba(X_test)[::,1]
roc_auc = metrics.roc_auc_score(y_test, y_pred_proba)

summary_knn_over_extended = summary.copy()
summary_knn_over_extended.loc[len(summary_knn_over_extended.index)] =␣
 ↪['ROC-AUC', roc_auc]
summary_knn_over_extended.set_index('Metric')

summary_knn_over_index = summary_knn_over_extended.T
summary_knn_over_index.columns = summary_knn_over_index.iloc[0]
summary_knn_over_index.drop(summary_knn_over_index.index[0], inplace = True)
summary_knn_over_index
```



[39]:
```
Metric                  MCC  F1-Score  F2-Score    Recall Precision FM index  \
Performance score  0.575425  0.523316  0.699446  0.901786  0.368613  0.57655

Metric             Specificity    G-mean F0.5-Score Accuracy ROC-AUC
Performance score     0.996957  0.948178   0.418046  0.99677     0.5
```

1.16 Random under-sampling with imbalanced-learn library

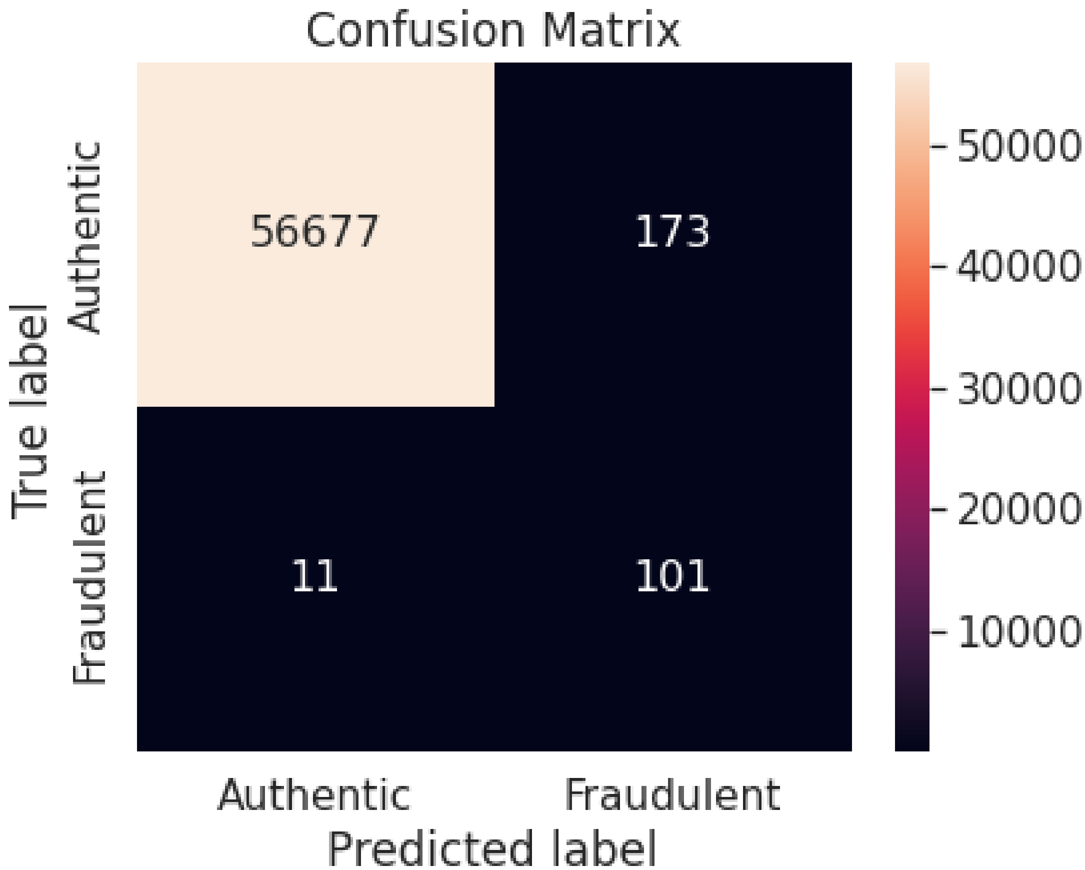
[40]:
```
# Elements of confusion matrix

classification(knn, X_train_under_imblearn_scaled_minmax,␣
↪y_train_under_imblearn, X_test_scaled_minmax, y_test)

# Summary of evaluation metrics

summary_knn_under_imblearn = summary
summary_knn_under_imblearn.set_index('Metric')

y_pred_proba = knn.predict_proba(X_test)[::,1]
roc_auc = metrics.roc_auc_score(y_test, y_pred_proba)

summary_knn_under_imblearn_extended = summary.copy()
summary_knn_under_imblearn_extended.loc[len(summary_knn_under_imblearn_extended.
↪index)] = ['ROC-AUC', roc_auc]
summary_knn_under_imblearn_extended.set_index('Metric')

summary_knn_under_imblearn_index = summary_knn_under_imblearn_extended.T
summary_knn_under_imblearn_index.columns = summary_knn_under_imblearn_index.
↪iloc[0]
summary_knn_under_imblearn_index.drop(summary_knn_under_imblearn_index.
↪index[0], inplace = True)
summary_knn_under_imblearn_index
```



[40]:
```
Metric                  MCC  F1-Score  F2-Score    Recall Precision  FM index  \
Performance score  0.655732  0.628571  0.760369  0.883929  0.487685  0.656566

Metric             Specificity    G-mean F0.5-Score  Accuracy   ROC-AUC
Performance score     0.998171  0.939314    0.535714  0.997946  0.499736
```

1.17 Random over-sampling with imbalanced-learn library

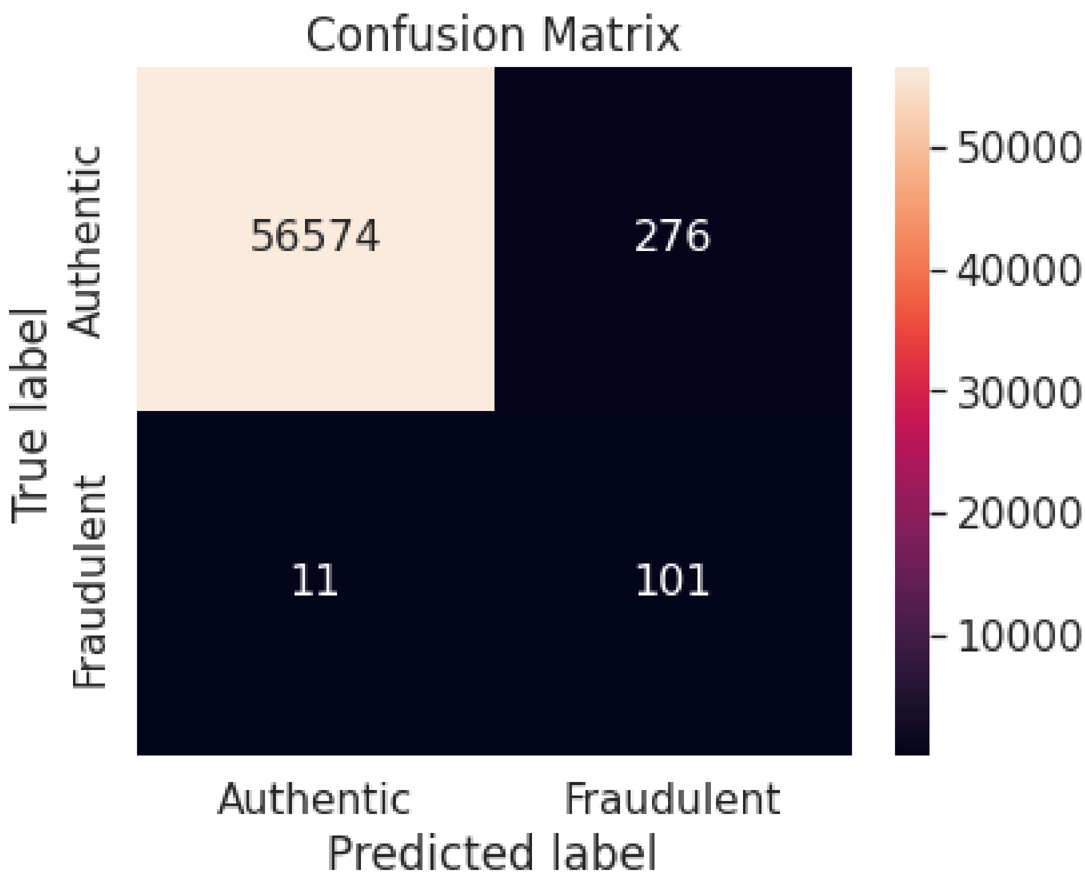
[41]:
```
# Elements of confusion matrix

classification(knn, X_train_over_imblearn_scaled_minmax, y_train_over_imblearn,␣
 ↪X_test_scaled_minmax, y_test)

# Summary of evaluation metrics

summary_knn_over_imblearn = summary
summary_knn_over_imblearn.set_index('Metric')

y_pred_proba = knn.predict_proba(X_test)[::,1]
roc_auc = metrics.roc_auc_score(y_test, y_pred_proba)

summary_knn_over_imblearn_extended = summary.copy()
summary_knn_over_imblearn_extended.loc[len(summary_knn_over_imblearn_extended.
 ↪index)] = ['ROC-AUC', roc_auc]
summary_knn_over_imblearn_extended.set_index('Metric')

summary_knn_over_imblearn_index = summary_knn_over_imblearn_extended.T
summary_knn_over_imblearn_index.columns = summary_knn_over_imblearn_index.
 ↪iloc[0]
summary_knn_over_imblearn_index.drop(summary_knn_over_imblearn_index.index[0],␣
 ↪inplace = True)
summary_knn_over_imblearn_index
```




```
[41]: Metric                    MCC  F1-Score  F2-Score    Recall Precision FM index  \
      Performance score  0.575425  0.523316  0.699446  0.901786  0.368613  0.57655

      Metric             Specificity    G-mean F0.5-Score Accuracy ROC-AUC
      Performance score     0.996957  0.948178   0.418046  0.99677     0.5
```

1.18 Synthetic minority over-sampling technique (SMOTE)

[42]:
```
# Elements of confusion matrix

classification(knn, X_train_over_smote_scaled_minmax, y_train_over_smote,␣
 ↪X_test_scaled_minmax, y_test)

# Summary of evaluation metrics

summary_knn_over_smote = summary
summary_knn_over_smote.set_index('Metric')

y_pred_proba = knn.predict_proba(X_test)[::,1]
roc_auc = metrics.roc_auc_score(y_test, y_pred_proba)

summary_knn_over_smote_extended = summary.copy()
summary_knn_over_smote_extended.loc[len(summary_knn_over_smote_extended.index)]␣
 ↪= ['ROC-AUC', roc_auc]
summary_knn_over_smote_extended.set_index('Metric')

summary_knn_over_smote_index = summary_knn_over_smote_extended.T
summary_knn_over_smote_index.columns = summary_knn_over_smote_index.iloc[0]
summary_knn_over_smote_index.drop(summary_knn_over_smote_index.index[0],␣
 ↪inplace = True)
summary_knn_over_smote_index
```

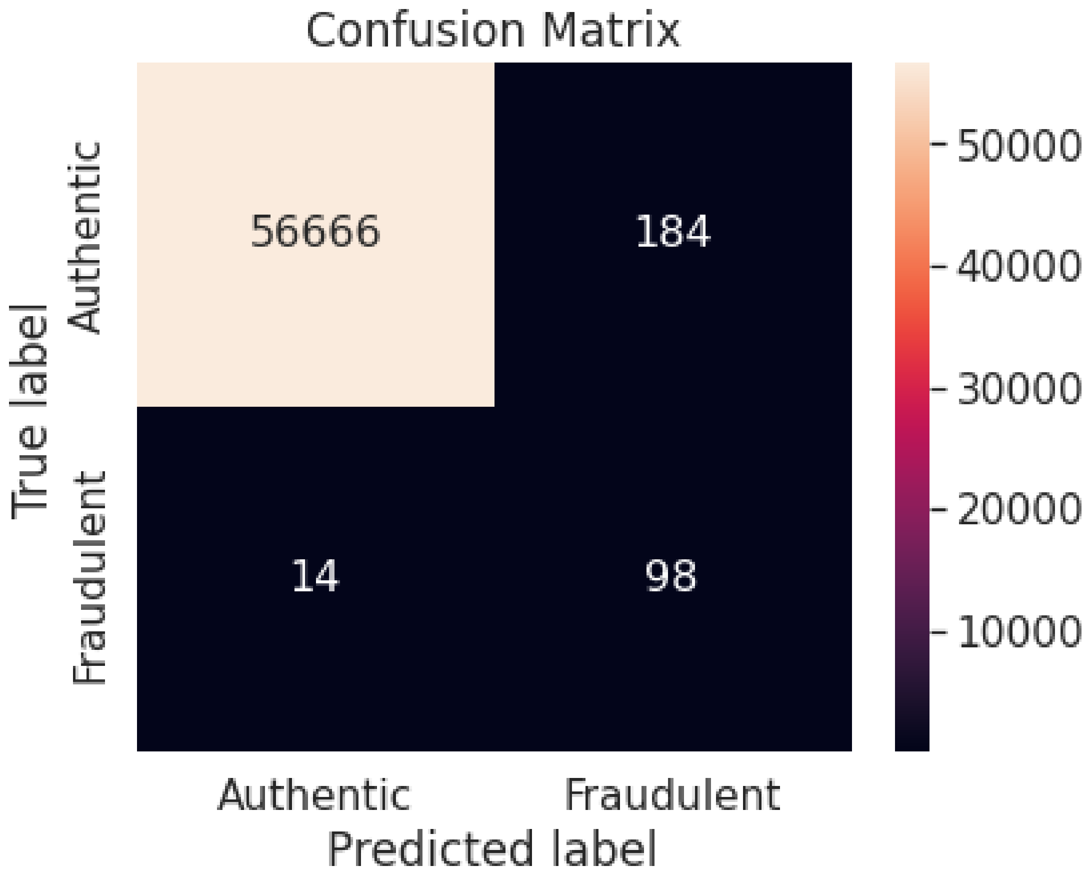


[42]:
```
Metric                  MCC  F1-Score  F2-Score    Recall Precision  FM index  \
Performance score  0.490018  0.413088  0.612121  0.901786  0.267905  0.491521

Metric            Specificity    G-mean F0.5-Score  Accuracy ROC-AUC
Performance score    0.995145  0.947316   0.311728  0.994962     0.5
```

1.19 Under-sampling via NearMiss

[43]:

```
# Elements of confusion matrix

classification(knn, X_train_under_nm_scaled_minmax, y_train_under_nm,␣
 ↪X_test_scaled_minmax, y_test)

# Summary of evaluation metrics

summary_knn_under_nm = summary
summary_knn_under_nm.set_index('Metric')

y_pred_proba = knn.predict_proba(X_test)[::,1]
roc_auc = metrics.roc_auc_score(y_test, y_pred_proba)

summary_knn_under_nm_extended = summary.copy()
summary_knn_under_nm_extended.loc[len(summary_knn_under_nm_extended.index)] =␣
 ↪['ROC-AUC', roc_auc]
summary_knn_under_nm_extended.set_index('Metric')

summary_knn_under_nm_index = summary_knn_under_nm_extended.T
summary_knn_under_nm_index.columns = summary_knn_under_nm_index.iloc[0]
summary_knn_under_nm_index.drop(summary_knn_under_nm_index.index[0], inplace =␣
 ↪True)
summary_knn_under_nm_index
```



[43]:
```
Metric                  MCC  F1-Score  F2-Score Recall Precision  FM index  \
Performance score  0.550216  0.497462  0.671233  0.875  0.347518  0.551433

Metric            Specificity    G-mean F0.5-Score  Accuracy   ROC-AUC
Performance score    0.996763  0.933899   0.395161  0.996524  0.657411
```

1.20 Summary of k-NN classification models

[44]:
```
summary_knn = pd.DataFrame(columns = ['Metric'])

summary_knn['Metric'] = EvalMetricLabels
summary_knn_list = [summary_knn_unaltered, summary_knn_under, summary_knn_over,␣
 ↪summary_knn_under_imblearn,
                    summary_knn_over_imblearn, summary_knn_over_smote,␣
 ↪summary_knn_under_nm]

for i in summary_knn_list:
    summary_knn = pd.merge(summary_knn, i, on = 'Metric')

TrainingSetsMetric = TrainingSets.copy()
TrainingSetsMetric.insert(0, 'Metric')

summary_knn.columns = TrainingSetsMetric
summary_knn.set_index('Metric', inplace = True)
summary_knn
```

[44]:
```
             Unaltered       RUS       ROS    RUS-IL    ROS-IL     SMOTE  \
Metric
MCC           0.852191  0.655732  0.575425  0.655732  0.575425  0.490018
F1-Score      0.848780  0.628571  0.523316  0.628571  0.523316  0.413088
F2-Score      0.804067  0.760369  0.699446  0.760369  0.699446  0.612121
Recall        0.776786  0.883929  0.901786  0.883929  0.901786  0.901786
Precision     0.935484  0.487685  0.368613  0.487685  0.368613  0.267905
FM index      0.852450  0.656566  0.576550  0.656566  0.576550  0.491521
Specificity   0.999894  0.998171  0.996957  0.998171  0.996957  0.995145
G-mean        0.881308  0.939314  0.948178  0.939314  0.948178  0.947316
F0.5-Score    0.898760  0.535714  0.418046  0.535714  0.418046  0.311728
Accuracy      0.999456  0.997946  0.996770  0.997946  0.996770  0.994962

                   NM
Metric
MCC          0.550216
F1-Score     0.497462
F2-Score     0.671233
Recall       0.875000
Precision    0.347518
FM index     0.551433
Specificity  0.996763
G-mean       0.933899
F0.5-Score   0.395161
Accuracy     0.996524
```

[45]:
```
# Visual comparison of the model applied on different training sets through
↪various evaluation metrics

summary_visual(summary_knn)
```

Note: A potential issue with k-NN classification models, which is relevant in this project is that, they are affected by **curse of dimensionality**, as well as **presence of outliers in the feature variables**. Despite that, it performs fairly well when applied on the unaltered (imbalanced) training set, in particular with respect to **MCC**, but **F2-score** as well.

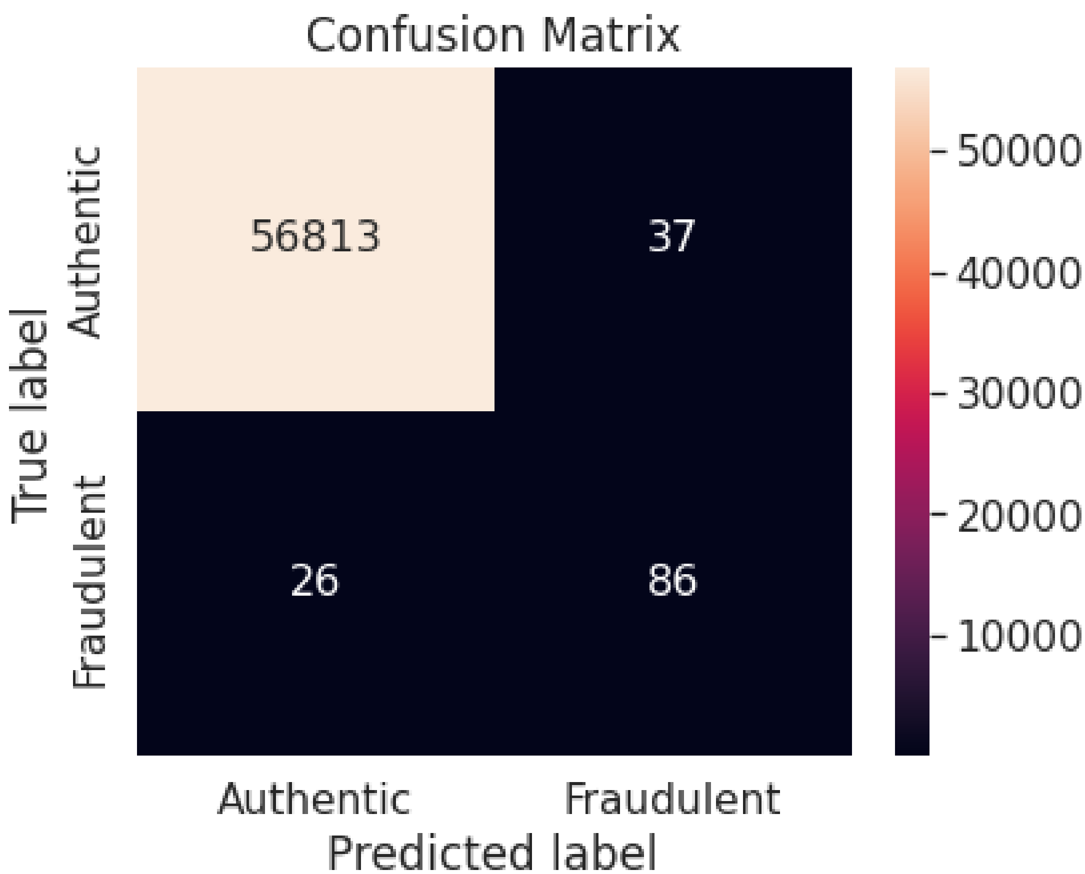
7. Decision Tree

[46]:
```
dt = DecisionTreeClassifier()
```

1.21 Unaltered training set

[47]:
```
# Elements of confusion matrix

classification(dt, X_train, y_train, X_test, y_test)

# Summary of evaluation metrics

summary_dt_unaltered = summary
summary_dt_unaltered.set_index('Metric')

y_pred_proba = dt.predict_proba(X_test)[::,1]
roc_auc = metrics.roc_auc_score(y_test, y_pred_proba)

summary_dt_unaltered_extended = summary.copy()
```

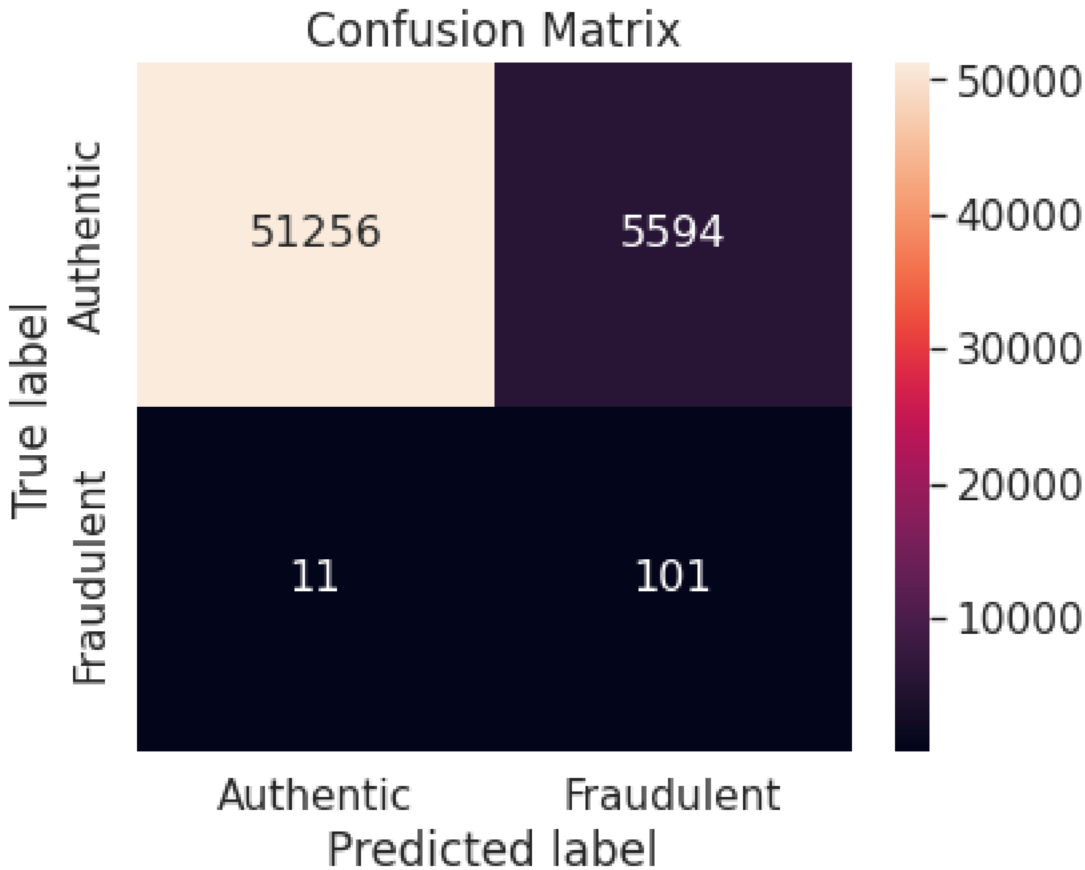
```
summary_dt_unaltered_extended.loc[len(summary_dt_unaltered_extended.index)] =
    ['ROC-AUC', roc_auc]
summary_dt_unaltered_extended.set_index('Metric')

summary_dt_unaltered_index = summary_dt_unaltered_extended.T
summary_dt_unaltered_index.columns = summary_dt_unaltered_index.iloc[0]
summary_dt_unaltered_index.drop(summary_dt_unaltered_index.index[0], inplace =
    True)
summary_dt_unaltered_index
```




```
[47]: Metric                    MCC  F1-Score  F2-Score    Recall Precision  FM index  \
      Performance score  0.732168  0.731915  0.753065  0.767857  0.699187  0.732718

      Metric             Specificity    G-mean F0.5-Score  Accuracy   ROC-AUC
      Performance score     0.999349  0.875989    0.711921  0.998894  0.883603
```

1.22 Random under-sampling

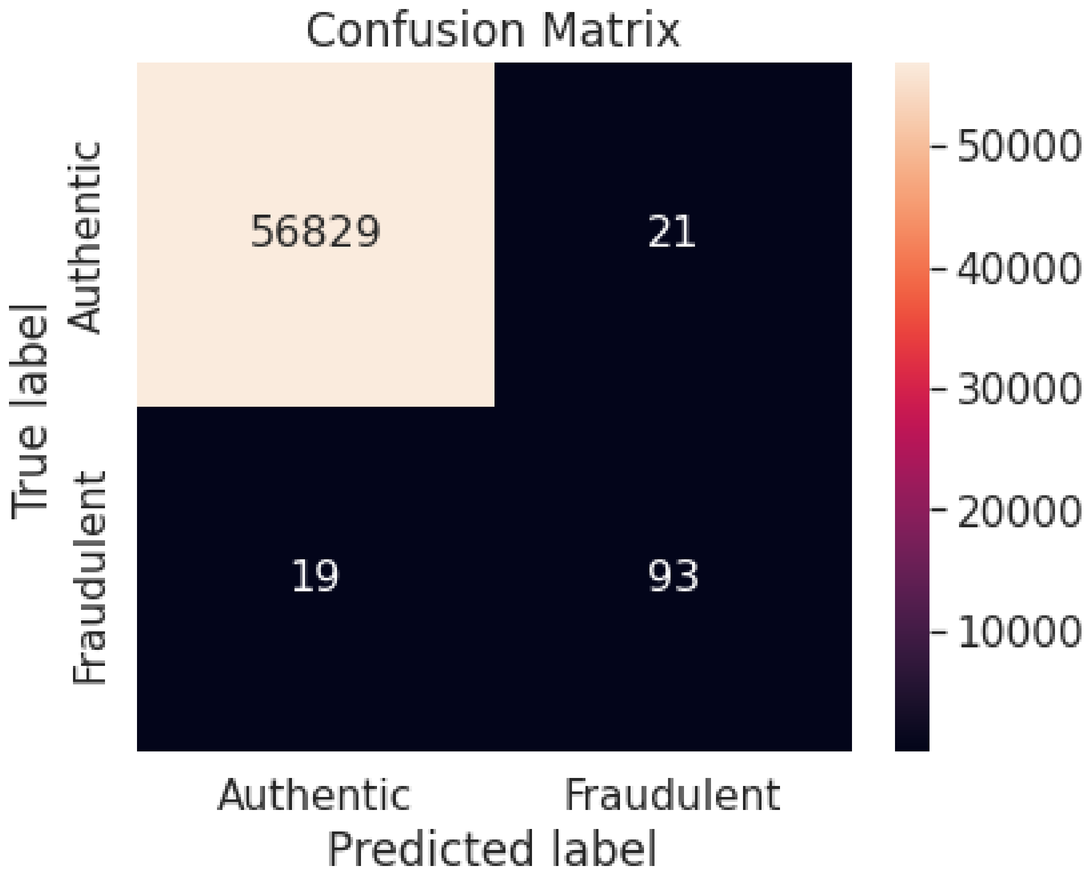
```
[48]: # Elements of confusion matrix

classification(dt, X_train_under, y_train_under, X_test, y_test)

# Summary of evaluation metrics

summary_dt_under = summary
summary_dt_under.set_index('Metric')

y_pred_proba = dt.predict_proba(X_test)[::,1]
roc_auc = metrics.roc_auc_score(y_test, y_pred_proba)

summary_dt_under_extended = summary.copy()
summary_dt_under_extended.loc[len(summary_dt_under_extended.index)] =␣
  ↪['ROC-AUC', roc_auc]
summary_dt_under_extended.set_index('Metric')

summary_dt_under_index = summary_dt_under_extended.T
summary_dt_under_index.columns = summary_dt_under_index.iloc[0]
summary_dt_under_index.drop(summary_dt_under_index.index[0], inplace = True)
summary_dt_under_index
```



[48]:
```
Metric                     MCC  F1-Score  F2-Score    Recall Precision  FM index  \
Performance score  0.11864  0.034786  0.082207  0.901786  0.017735  0.126464

Metric             Specificity    G-mean F0.5-Score  Accuracy   ROC-AUC
Performance score     0.901601  0.901693    0.02206  0.901601  0.901693
```

1.23 Random over-sampling

[49]:
```
# Elements of confusion matrix

classification(dt, X_train_over, y_train_over, X_test, y_test)

# Summary of evaluation metrics

summary_dt_over = summary
summary_dt_over.set_index('Metric')

y_pred_proba = dt.predict_proba(X_test)[::,1]
roc_auc = metrics.roc_auc_score(y_test, y_pred_proba)

summary_dt_over_extended = summary.copy()
summary_dt_over_extended.loc[len(summary_dt_over_extended.index)] = ['ROC-AUC',␣
 ↪roc_auc]
summary_dt_over_extended.set_index('Metric')

summary_dt_over_index = summary_dt_over_extended.T
summary_dt_over_index.columns = summary_dt_over_index.iloc[0]
summary_dt_over_index.drop(summary_dt_over_index.index[0], inplace = True)
summary_dt_over_index
```

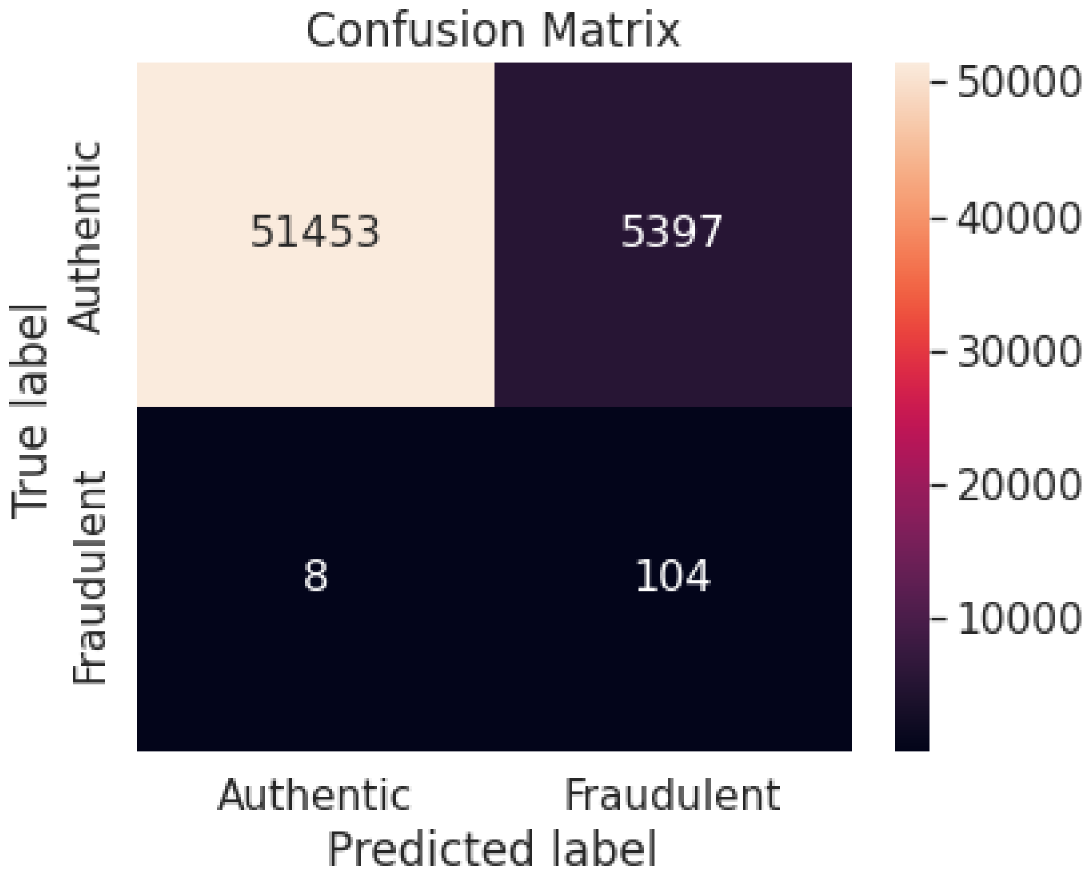


[49]:

```
Metric                  MCC  F1-Score  F2-Score    Recall Precision  FM index  \
Performance score  0.822689  0.823009  0.827402  0.830357  0.815789  0.823041

Metric             Specificity    G-mean F0.5-Score  Accuracy   ROC-AUC
Performance score     0.999631  0.911071   0.818662  0.999298  0.914994
```

1.24 Random under-sampling with imbalanced-learn library

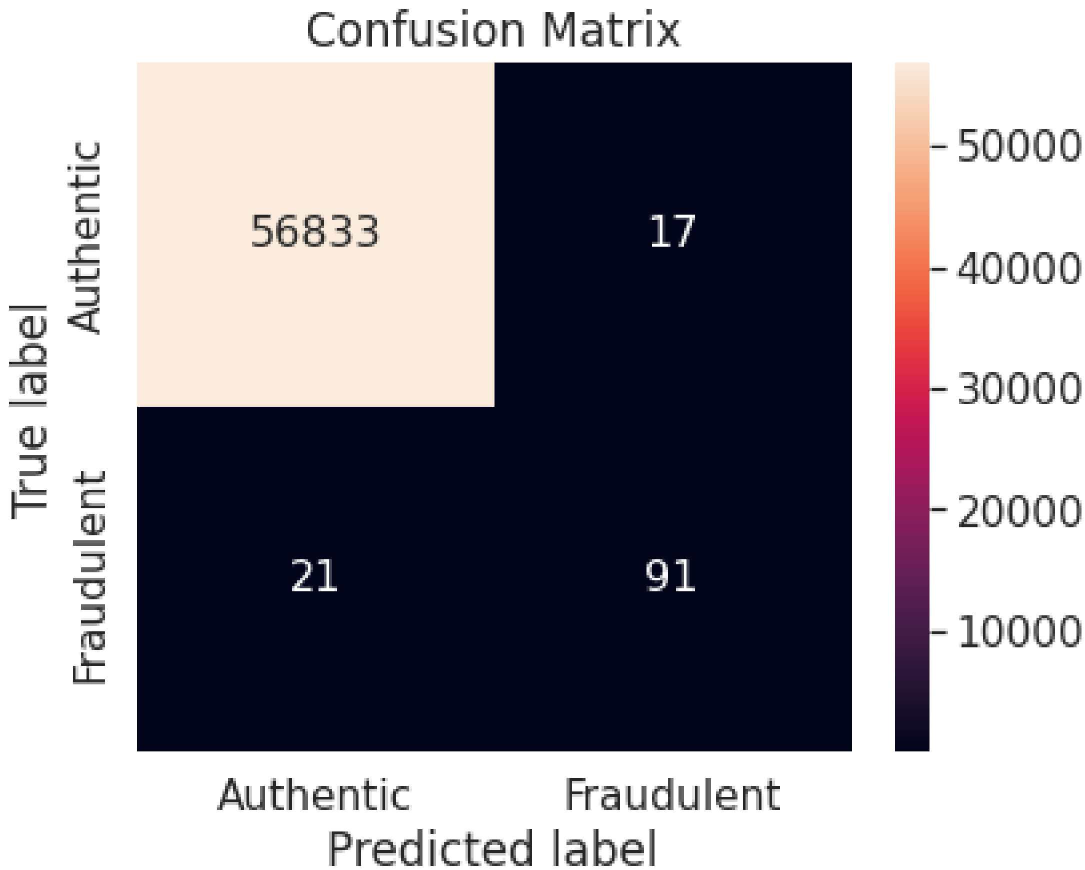
[50]:
```
# Elements of confusion matrix

classification(dt, X_train_under_imblearn, y_train_under_imblearn, X_test,␣
 ↪y_test)

# Summary of evaluation metrics

summary_dt_under_imblearn = summary
summary_dt_under_imblearn.set_index('Metric')

y_pred_proba = dt.predict_proba(X_test)[::,1]
roc_auc = metrics.roc_auc_score(y_test, y_pred_proba)

summary_dt_under_imblearn_extended = summary.copy()
summary_dt_under_imblearn_extended.loc[len(summary_dt_under_imblearn_extended.
 ↪index)] = ['ROC-AUC', roc_auc]
summary_dt_under_imblearn_extended.set_index('Metric')

summary_dt_under_imblearn_index = summary_dt_under_imblearn_extended.T
summary_dt_under_imblearn_index.columns = summary_dt_under_imblearn_index.
 ↪iloc[0]
summary_dt_under_imblearn_index.drop(summary_dt_under_imblearn_index.index[0],␣
 ↪inplace = True)
summary_dt_under_imblearn_index
```



[50]:
```
Metric                      MCC  F1-Score F2-Score    Recall Precision  FM index  \
Performance score  0.125023  0.037057  0.08741  0.928571  0.018906  0.132496

Metric             Specificity    G-mean F0.5-Score  Accuracy   ROC-AUC
Performance score     0.905066  0.916743   0.023512  0.905112  0.916819
```

1.25 Random over-sampling with imbalanced-learn library

[51]:
```
# Elements of confusion matrix

classification(dt, X_train_over_imblearn, y_train_over_imblearn, X_test, y_test)

# Summary of evaluation metrics

summary_dt_over_imblearn = summary
summary_dt_over_imblearn.set_index('Metric')

y_pred_proba = dt.predict_proba(X_test)[::,1]
roc_auc = metrics.roc_auc_score(y_test, y_pred_proba)

summary_dt_over_imblearn_extended = summary.copy()
summary_dt_over_imblearn_extended.loc[len(summary_dt_over_imblearn_extended.
 ↪index)] = ['ROC-AUC', roc_auc]
summary_dt_over_imblearn_extended.set_index('Metric')

summary_dt_over_imblearn_index = summary_dt_over_imblearn_extended.T
summary_dt_over_imblearn_index.columns = summary_dt_over_imblearn_index.iloc[0]
summary_dt_over_imblearn_index.drop(summary_dt_over_imblearn_index.index[0],␣
 ↪inplace = True)
summary_dt_over_imblearn_index
```

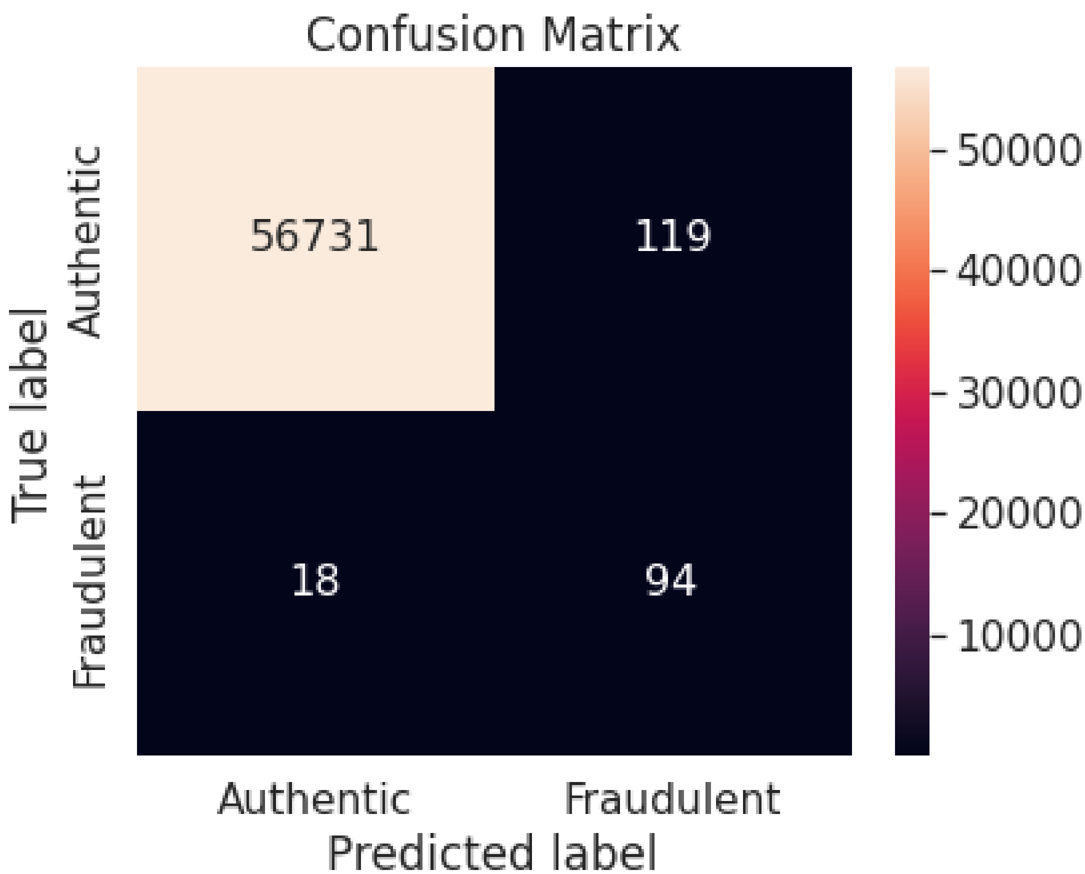


[51]:

```
Metric                    MCC  F1-Score  F2-Score  Recall Precision FM index  \
Performance score  0.827076  0.827273  0.818345  0.8125   0.842593   0.82741

Metric             Specificity    G-mean F0.5-Score  Accuracy ROC-AUC
Performance score     0.999701  0.901253   0.836397  0.999333   0.9061
```

1.26 Synthetic minority over-sampling technique (SMOTE)

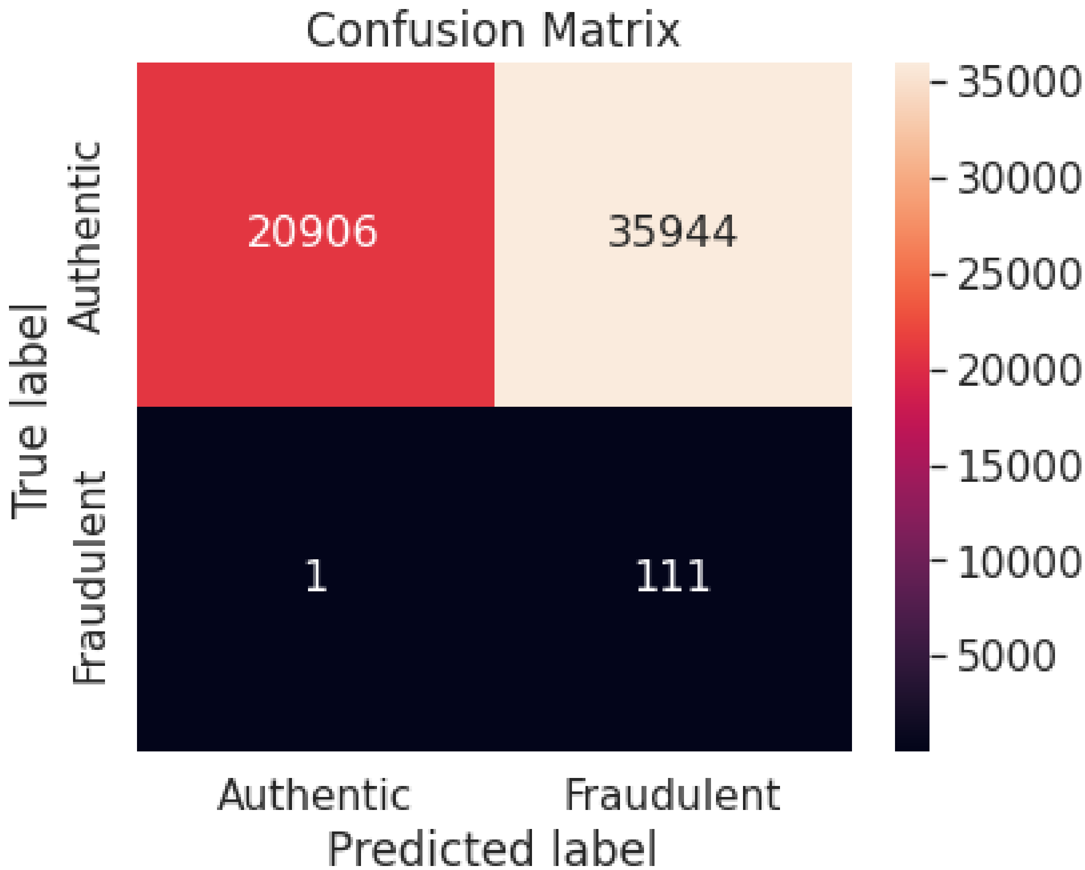
[52]:
```python
# Elements of confusion matrix

classification(dt, X_train_over_smote, y_train_over_smote, X_test, y_test)

# Summary of evaluation metrics

summary_dt_over_smote = summary
summary_dt_over_smote.set_index('Metric')

y_pred_proba = dt.predict_proba(X_test)[::,1]
roc_auc = metrics.roc_auc_score(y_test, y_pred_proba)

summary_dt_over_smote_extended = summary.copy()
summary_dt_over_smote_extended.loc[len(summary_dt_over_smote_extended.index)] =␣
↪['ROC-AUC', roc_auc]
summary_dt_over_smote_extended.set_index('Metric')

summary_dt_over_smote_index = summary_dt_over_smote_extended.T
summary_dt_over_smote_index.columns = summary_dt_over_smote_index.iloc[0]
summary_dt_over_smote_index.drop(summary_dt_over_smote_index.index[0], inplace␣
↪= True)
summary_dt_over_smote_index
```

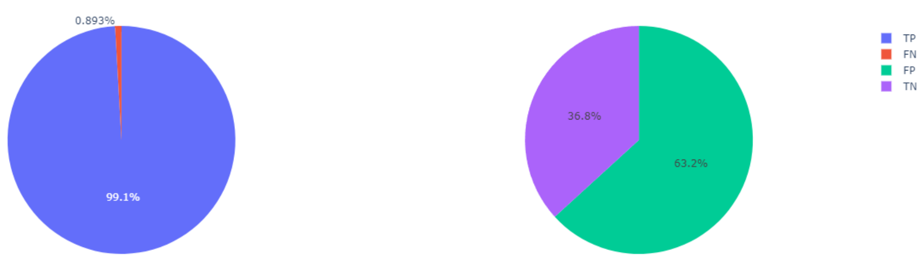



```
[52]: Metric                    MCC   F1-Score  F2-Score    Recall Precision  FM index  \
      Performance score  0.607618  0.578462  0.711044  0.839286  0.441315  0.608596

      Metric             Specificity    G-mean F0.5-Score  Accuracy   ROC-AUC
      Performance score     0.997907  0.915166   0.487552  0.997595  0.918596
```

1.27 Under-sampling via NearMiss

[53]:
```
# Elements of confusion matrix

classification(dt, X_train_under_nm, y_train_under_nm, X_test, y_test)

# Summary of evaluation metrics

summary_dt_under_nm = summary
summary_dt_under_nm.set_index('Metric')

y_pred_proba = dt.predict_proba(X_test)[::,1]
roc_auc = metrics.roc_auc_score(y_test, y_pred_proba)

summary_dt_under_nm_extended = summary.copy()
summary_dt_under_nm_extended.loc[len(summary_dt_under_nm_extended.index)] =␣
↪['ROC-AUC', roc_auc]
summary_dt_under_nm_extended.set_index('Metric')

summary_dt_under_nm_index = summary_dt_under_nm_extended.T
summary_dt_under_nm_index.columns = summary_dt_under_nm_index.iloc[0]
summary_dt_under_nm_index.drop(summary_dt_under_nm_index.index[0])
summary_dt_under_nm_index
```





[53]:
```
Metric                   MCC   F1-Score  F2-Score    Recall  Precision  \
Metric                   MCC   F1-Score  F2-Score    Recall  Precision
Performance score   0.032977   0.006138  0.015204  0.991071   0.003079

Metric              FM index  Specificity    G-mean  F0.5-Score  Accuracy  \
```

```
Metric             FM index  Specificity    G-mean  F0.5-Score  Accuracy
Performance score  0.055237      0.36774  0.603702    0.003845  0.368965

Metric              ROC-AUC
Metric              ROC-AUC
Performance score  0.679406
```

1.28 Summary of decision tree classification models

[54]:
```python
summary_dt = pd.DataFrame(columns = ['Metric'])

EvalMetricLabels_dt = EvalMetricLabels
summary_dt['Metric'] = EvalMetricLabels
summary_dt_list = [summary_dt_unaltered, summary_dt_under, summary_dt_over,␣
 ↪summary_dt_under_imblearn,
                    summary_dt_over_imblearn, summary_dt_over_smote,␣
 ↪summary_dt_under_nm]

for i in summary_dt_list:
    summary_dt = pd.merge(summary_dt, i, on = 'Metric')

TrainingSetsMetric = TrainingSets.copy()
TrainingSetsMetric.insert(0, 'Metric')

summary_dt.columns = TrainingSetsMetric
summary_dt.set_index('Metric', inplace = True)
summary_dt
```

[54]:
```
             Unaltered       RUS       ROS    RUS-IL    ROS-IL     SMOTE  \
Metric
MCC           0.732168  0.118640  0.822689  0.125023  0.827076  0.607618
F1-Score      0.731915  0.034786  0.823009  0.037057  0.827273  0.578462
F2-Score      0.753065  0.082207  0.827402  0.087410  0.818345  0.711044
Recall        0.767857  0.901786  0.830357  0.928571  0.812500  0.839286
Precision     0.699187  0.017735  0.815789  0.018906  0.842593  0.441315
FM index      0.732718  0.126464  0.823041  0.132496  0.827410  0.608596
Specificity   0.999349  0.901601  0.999631  0.905066  0.999701  0.997907
G-mean        0.875989  0.901693  0.911071  0.916743  0.901253  0.915166
F0.5-Score    0.711921  0.022060  0.818662  0.023512  0.836397  0.487552
Accuracy      0.998894  0.901601  0.999298  0.905112  0.999333  0.997595

                   NM
Metric
MCC          0.032977
F1-Score     0.006138
F2-Score     0.015204
```

```
Recall         0.991071
Precision      0.003079
FM index       0.055237
Specificity    0.367740
G-mean         0.603702
F0.5-Score     0.003845
Accuracy       0.368965
```

[55]:
```
# Visual comparison of the model applied on different training sets through␣
↪various evaluation metrics

summary_visual(summary_dt)
```

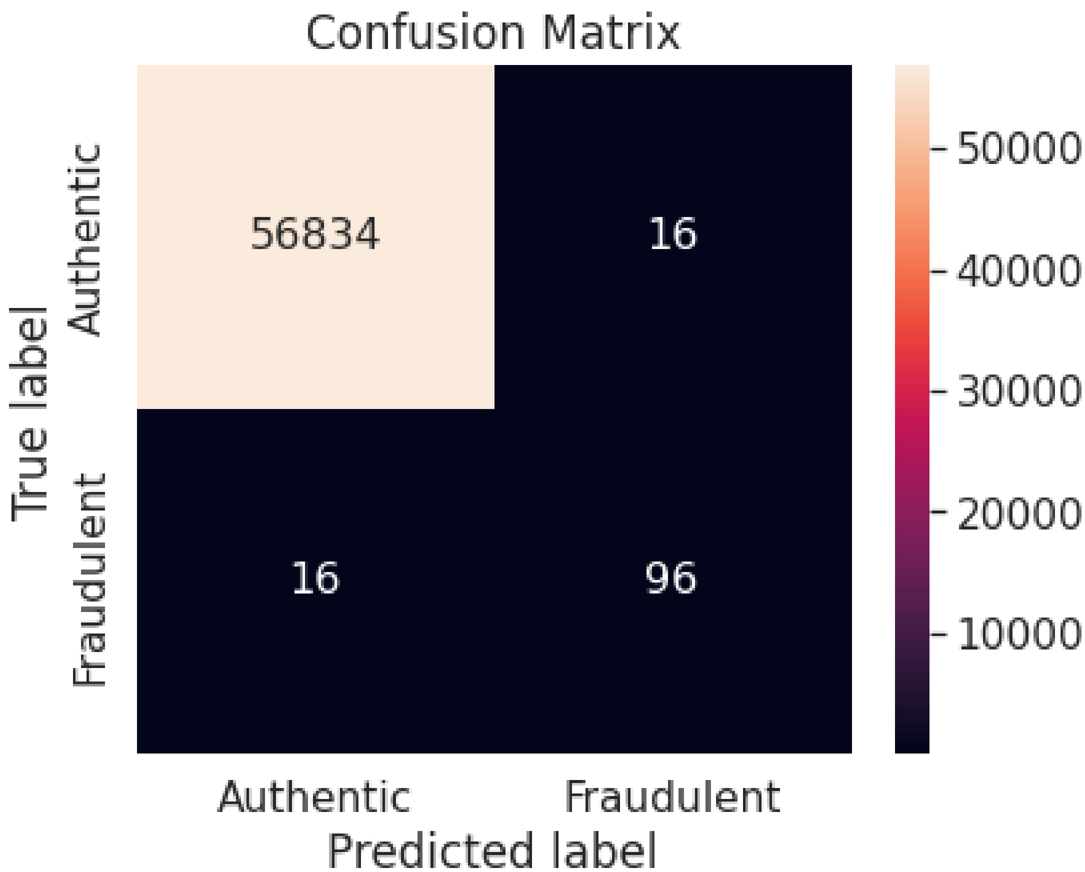
8. Support Vector Machine (SVM)

[56]:
```
svm_linear = svm.SVC(kernel = 'linear')
```

1.29 Unaltered training set

[57]:
```
# Elements of confusion matrix

classification(svm_linear, X_train_scaled_minmax, y_train,␣
↪X_test_scaled_minmax, y_test)

# Summary of evaluation metrics

summary_svm_linear_unaltered = summary
summary_svm_linear_unaltered.set_index('Metric')

summary_svm_linear_unaltered_index = summary_svm_linear_unaltered.T
```

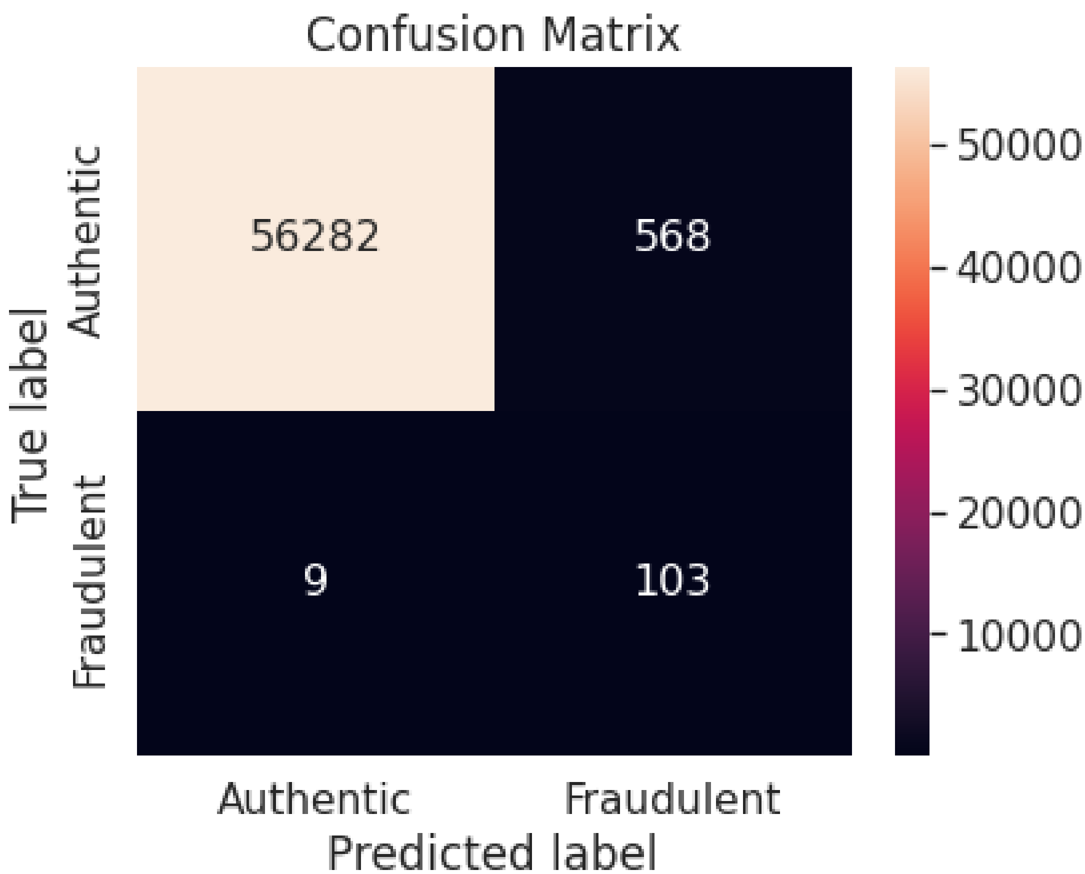
```
summary_svm_linear_unaltered_index.columns = summary_svm_linear_unaltered_index.
  ↪iloc[0]
summary_svm_linear_unaltered_index.drop(summary_svm_linear_unaltered_index.
  ↪index[0], inplace = True)
summary_svm_linear_unaltered_index

# classification(svm_linear, X_train, y_train, X_test, y_test) # TP = 37, FN =␣
  ↪75, TN = 56840, FP = 10
```



[57]:
```
Metric                    MCC  F1-Score  F2-Score    Recall Precision  FM index  \
Performance score  0.856861  0.857143  0.857143  0.857143  0.857143  0.857143

Metric            Specificity   G-mean F0.5-Score  Accuracy
Performance score    0.999719  0.92569   0.857143  0.999438
```

1.30 Random under-sampling

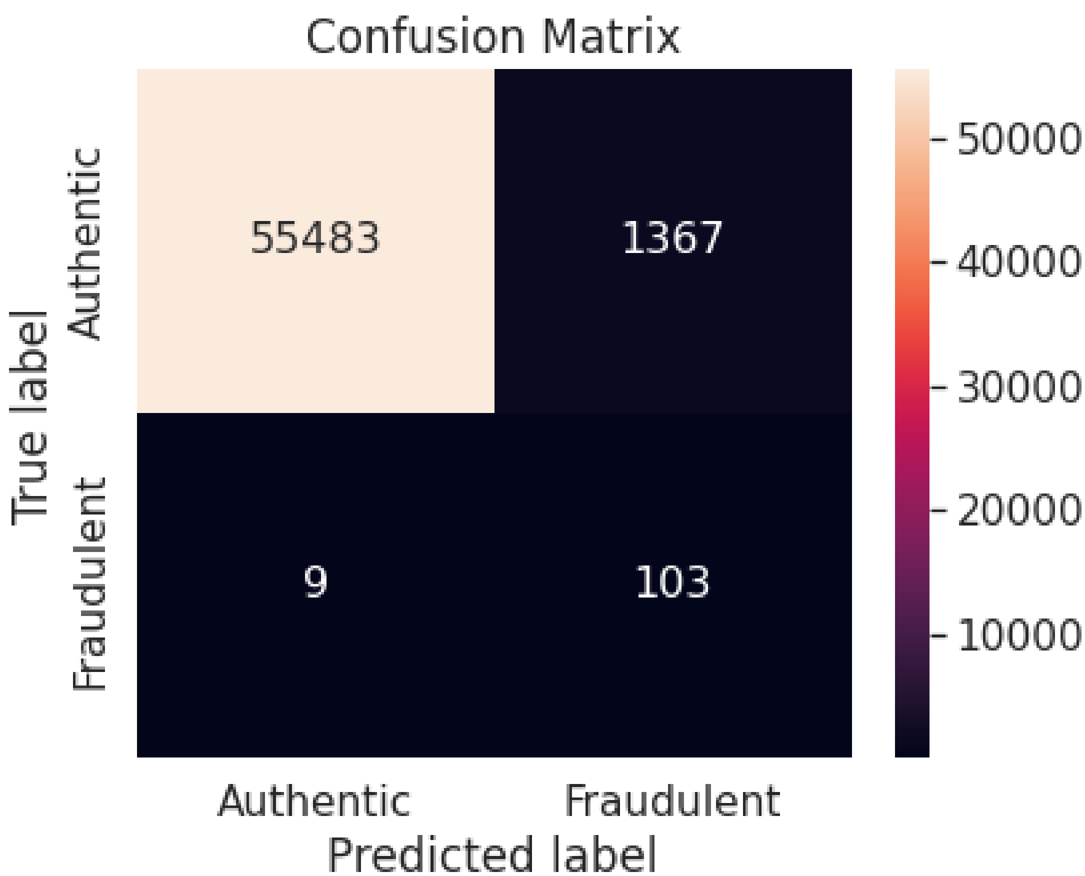
[58]:
```python
# Elements of confusion matrix

classification(svm_linear, X_train_under_scaled_minmax, y_train_under,␣
 ↪X_test_scaled_minmax, y_test)

# Summary of evaluation metrics

summary_svm_linear_under = summary
summary_svm_linear_under.set_index('Metric')

summary_svm_linear_under_index = summary_svm_linear_under.T
summary_svm_linear_under_index.columns = summary_svm_linear_under_index.iloc[0]
summary_svm_linear_under_index.drop(summary_svm_linear_under_index.index[0],␣
 ↪inplace = True)
summary_svm_linear_under_index
```



[58]:
```
Metric                  MCC  F1-Score  F2-Score    Recall Precision  FM index  \
Performance score  0.373481  0.263091  0.460232  0.919643  0.153502  0.375722

Metric             Specificity    G-mean F0.5-Score Accuracy
Performance score     0.990009  0.954177   0.184192  0.98987
```

1.31 Random over-sampling

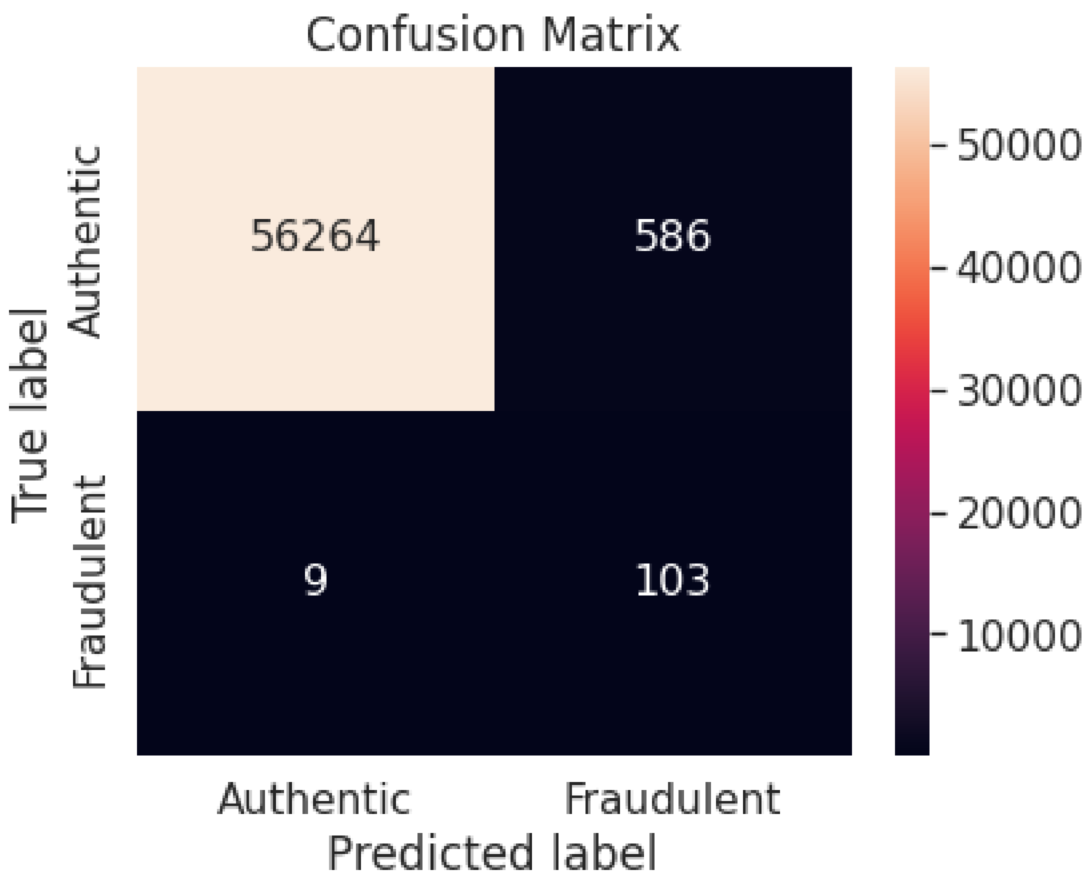
[59]:
```
# Elements of confusion matrix

classification(svm_linear, X_train_over_scaled_minmax, y_train_over,␣
↪X_test_scaled_minmax, y_test)

# Summary of evaluation metrics

summary_svm_linear_over = summary
summary_svm_linear_over.set_index('Metric')

summary_svm_linear_over_index = summary_svm_linear_over.T
summary_svm_linear_over_index.columns = summary_svm_linear_over_index.iloc[0]
summary_svm_linear_over_index.drop(summary_svm_linear_over_index.index[0],␣
↪inplace = True)
summary_svm_linear_over_index
```



[59]:
```
Metric                    MCC  F1-Score  F2-Score    Recall Precision  FM index  \
Performance score  0.250215  0.130215  0.268509  0.919643  0.070068  0.253846

Metric            Specificity   G-mean F0.5-Score  Accuracy
Performance score    0.975954  0.94738   0.085948  0.975844
```

1.32 Random under-sampling with imbalanced-learning library

[60]:
```python
# Elements of confusion matrix

classification(svm_linear, X_train_under_imblearn_scaled_minmax,␣
 ↪y_train_under_imblearn, X_test_scaled_minmax, y_test)

# Summary of evaluation metrics

summary_svm_linear_under_imblearn = summary
summary_svm_linear_under_imblearn.set_index('Metric')

summary_svm_linear_under_imblearn_index = summary_svm_linear_under_imblearn.T
summary_svm_linear_under_imblearn_index.columns =␣
 ↪summary_svm_linear_under_imblearn_index.iloc[0]
summary_svm_linear_under_imblearn_index.
 ↪drop(summary_svm_linear_under_imblearn_index.index[0], inplace = True)
summary_svm_linear_under_imblearn_index
```



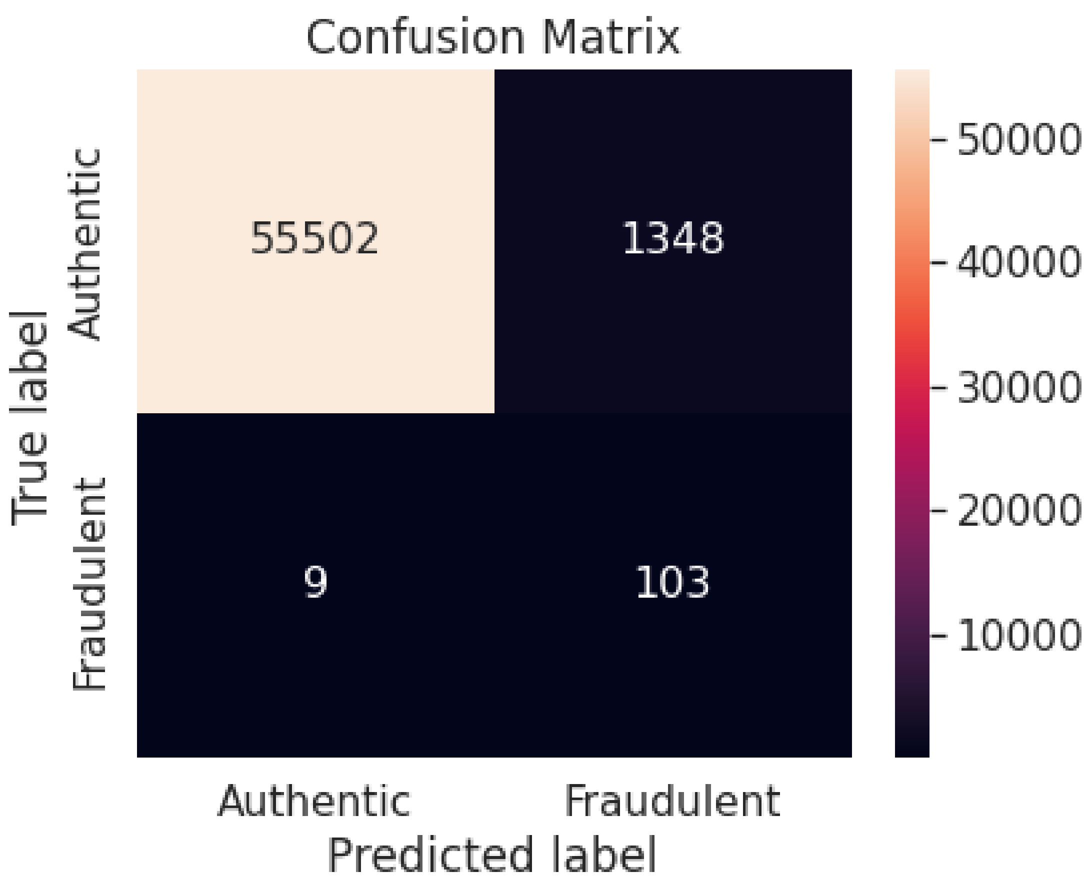
[60]:

Metric	MCC	F1-Score	F2-Score	Recall	Precision	FM index
Performance score	0.368501	0.257179	0.452946	0.919643	0.149492	0.370782

Metric	Specificity	G-mean	F0.5-Score	Accuracy
Performance score	0.989692	0.954025	0.179568	0.989554

1.33 Random over-sampling with imbalanced-learning library

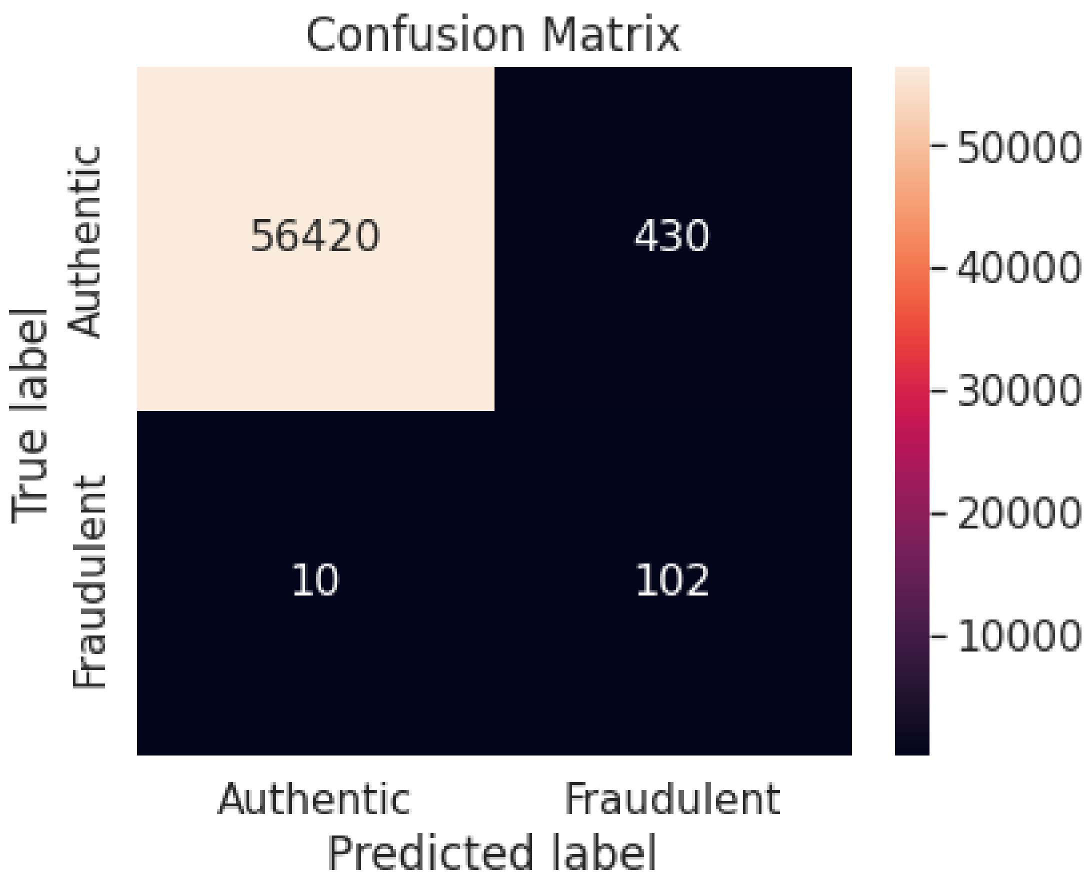
[61]:
```
# Elements of confusion matrix

classification(svm_linear, X_train_over_imblearn_scaled_minmax,␣
↪y_train_over_imblearn, X_test_scaled_minmax, y_test)

# Summary of evaluation metrics

summary_svm_linear_over_imblearn = summary
summary_svm_linear_over_imblearn.set_index('Metric')

summary_svm_linear_over_imblearn_index = summary_svm_linear_over_imblearn.T
summary_svm_linear_over_imblearn_index.columns =␣
↪summary_svm_linear_over_imblearn_index.iloc[0]
summary_svm_linear_over_imblearn_index.
↪drop(summary_svm_linear_over_imblearn_index.index[0], inplace = True)
summary_svm_linear_over_imblearn_index
```



[61]:
```
Metric                    MCC  F1-Score  F2-Score    Recall Precision  FM index  \
Performance score  0.251899  0.131798  0.271195  0.919643  0.070986  0.255502

Metric             Specificity    G-mean F0.5-Score  Accuracy
Performance score     0.976288  0.947542    0.087052  0.976177
```

1.34 Synthetic minority over-sampling technique (SMOTE)

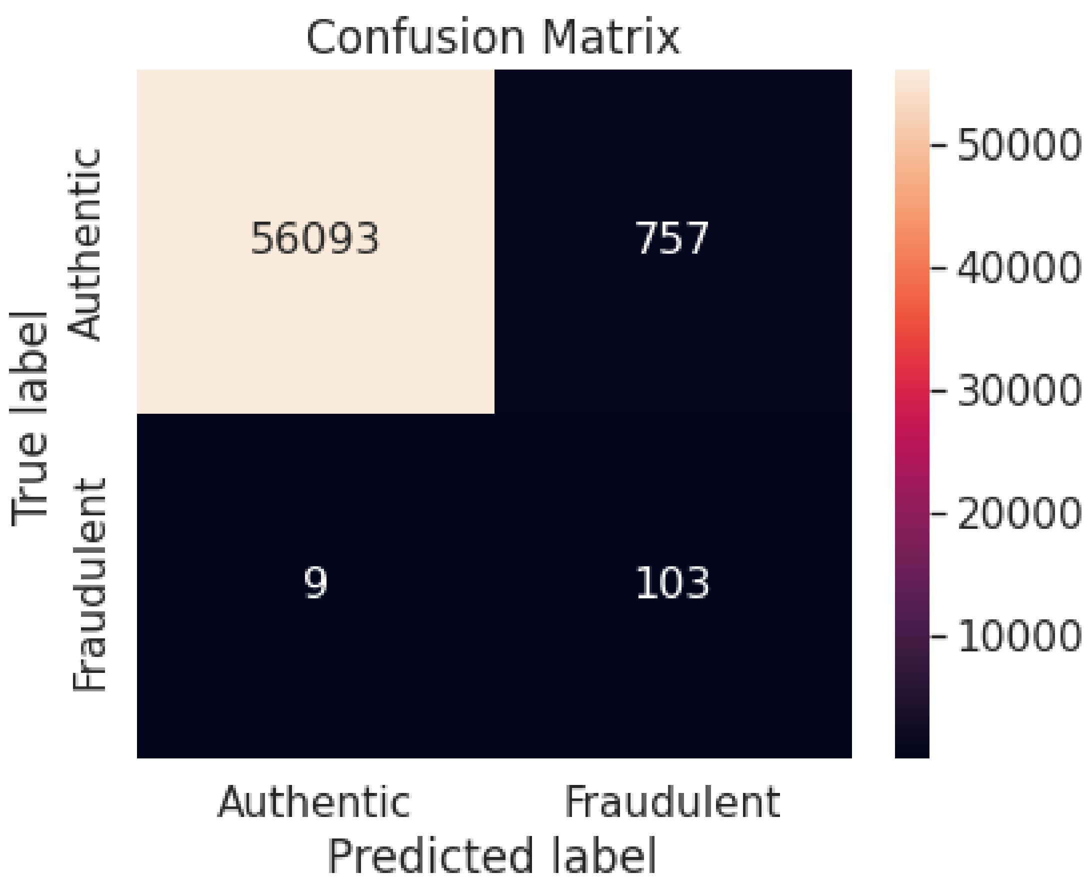
[62]:
```python
# Elements of confusion matrix

classification(svm_linear, X_train_over_smote_scaled_minmax,
    y_train_over_smote, X_test_scaled_minmax, y_test)

# Summary of evaluation metrics

summary_svm_linear_over_smote = summary
summary_svm_linear_over_smote.set_index('Metric')

summary_svm_linear_over_smote_index = summary_svm_linear_over_smote.T
summary_svm_linear_over_smote_index.columns =
    summary_svm_linear_over_smote_index.iloc[0]
summary_svm_linear_over_smote_index.drop(summary_svm_linear_over_smote_index.
    index[0], inplace = True)
summary_svm_linear_over_smote_index
```



[62]:
```
Metric                           MCC F1-Score  F2-Score    Recall Precision  FM index  \
Performance score  0.415933  0.31677  0.520408  0.910714  0.191729  0.417864

Metric             Specificity    G-mean F0.5-Score  Accuracy
Performance score     0.992436  0.950698   0.227679  0.992276
```

1.35 Under-sampling via NearMiss

[63]:
```
# Elements of confusion matrix

classification(svm_linear, X_train_under_nm_scaled_minmax, y_train_under_nm,␣
↪X_test_scaled_minmax, y_test)

# Summary of evaluation metrics

summary_svm_linear_under_nm = summary
summary_svm_linear_under_nm.set_index('Metric')

summary_svm_linear_under_nm_index = summary_svm_linear_under_nm.T
summary_svm_linear_under_nm_index.columns = summary_svm_linear_under_nm_index.
↪iloc[0]
summary_svm_linear_under_nm_index.drop(summary_svm_linear_under_nm_index.
↪index[0], inplace = True)
summary_svm_linear_under_nm_index
```



[63]:
```
Metric                    MCC  F1-Score  F2-Score    Recall Precision  FM index  \
Performance score  0.329246  0.211934  0.393731  0.919643  0.119767  0.331878

Metric             Specificity    G-mean F0.5-Score  Accuracy
Performance score     0.986684  0.952574    0.144989  0.986552
```

1.36 Summary of linear SVM classification models

[64]:
```python
summary_svm_linear = pd.DataFrame(columns = ['Metric'])

summary_svm_linear['Metric'] = EvalMetricLabels
summary_svm_linear_list = [summary_svm_linear_unaltered,␣
 ↪summary_svm_linear_under, summary_svm_linear_over,
                           summary_svm_linear_under_imblearn,␣
 ↪summary_svm_linear_over_imblearn,
                           summary_svm_linear_over_smote,␣
 ↪summary_svm_linear_under_nm]

for i in summary_svm_linear_list:
    summary_svm_linear = pd.merge(summary_svm_linear, i, on = 'Metric')

TrainingSetsMetric = TrainingSets.copy()
TrainingSetsMetric.insert(0, 'Metric')

summary_svm_linear.columns = TrainingSetsMetric
summary_svm_linear.set_index('Metric', inplace = True)
summary_svm_linear
```

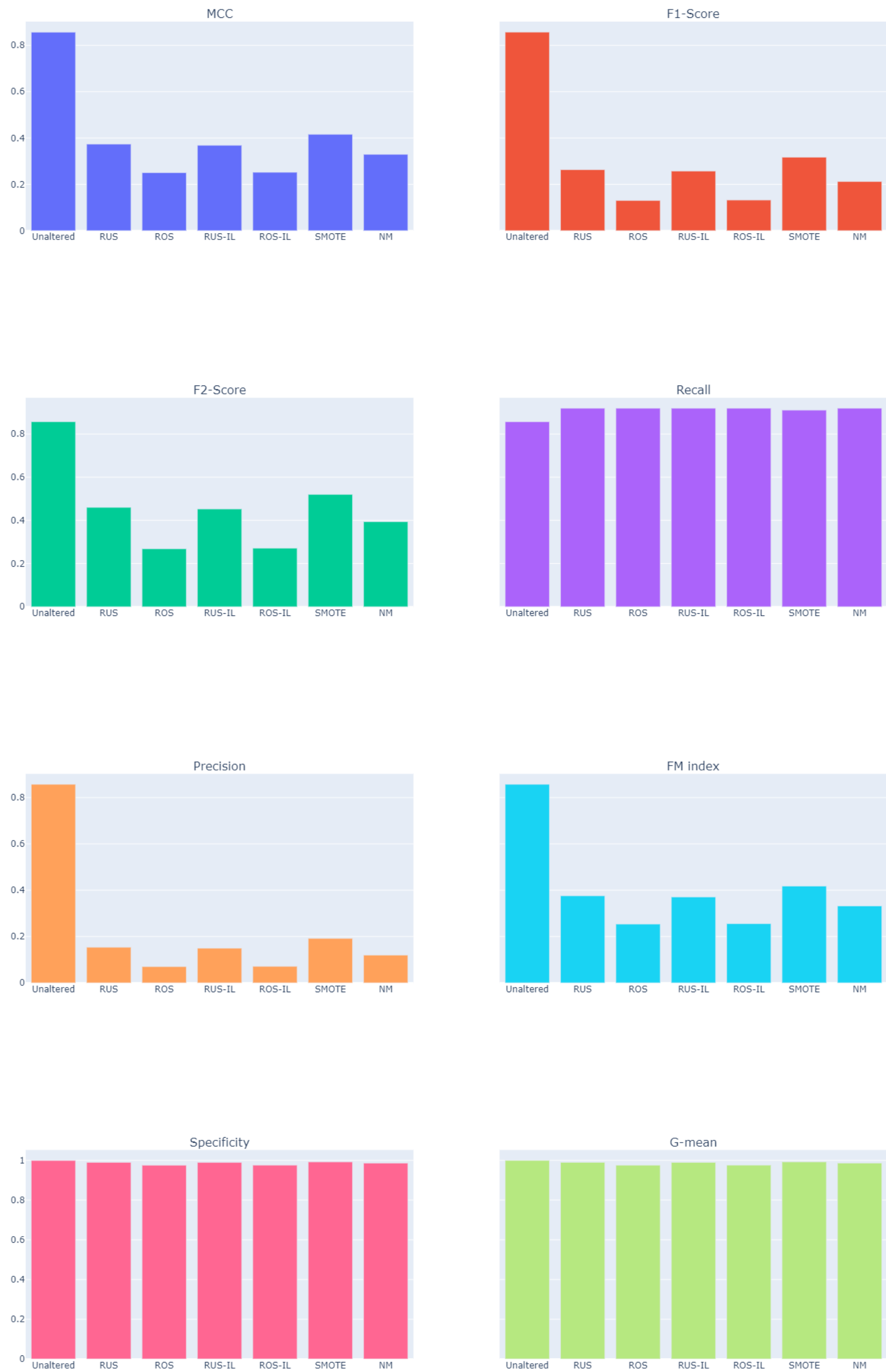
[64]:
```
             Unaltered       RUS       ROS    RUS-IL    ROS-IL     SMOTE  \
Metric
MCC           0.856861  0.373481  0.250215  0.368501  0.251899  0.415933
F1-Score      0.857143  0.263091  0.130215  0.257179  0.131798  0.316770
F2-Score      0.857143  0.460232  0.268509  0.452946  0.271195  0.520408
```

```
Recall         0.857143  0.919643  0.919643  0.919643  0.919643  0.910714
Precision      0.857143  0.153502  0.070068  0.149492  0.070986  0.191729
FM index       0.857143  0.375722  0.253846  0.370782  0.255502  0.417864
Specificity    0.999719  0.990009  0.975954  0.989692  0.976288  0.992436
G-mean         0.925690  0.954177  0.947380  0.954025  0.947542  0.950698
F0.5-Score     0.857143  0.184192  0.085948  0.179568  0.087052  0.227679
Accuracy       0.999438  0.989870  0.975844  0.989554  0.976177  0.992276

                     NM
Metric
MCC            0.329246
F1-Score       0.211934
F2-Score       0.393731
Recall         0.919643
Precision      0.119767
FM index       0.331878
Specificity    0.986684
G-mean         0.952574
F0.5-Score     0.144989
Accuracy       0.986552
```

[65]:

```python
# Visual comparison of the model applied on different training sets through␣
↪various evaluation metrics

summary_visual(summary_svm_linear)

fig1 = make_subplots(rows = 4, cols = 2, shared_yaxes = True, subplot_titles =␣
↪EvalMetricLabels)

fig1.add_trace(go.Bar(x = list(summary_svm_linear.columns), y =␣
↪list(summary_svm_linear.loc['MCC'])), 1, 1)
fig1.add_trace(go.Bar(x = list(summary_svm_linear.columns), y =␣
↪list(summary_svm_linear.loc['F1-Score'])), 1, 2)
fig1.add_trace(go.Bar(x = list(summary_svm_linear.columns), y =␣
↪list(summary_svm_linear.loc['F2-Score'])), 2, 1)
fig1.add_trace(go.Bar(x = list(summary_svm_linear.columns), y =␣
↪list(summary_svm_linear.loc['Recall'])), 2, 2)
fig1.add_trace(go.Bar(x = list(summary_svm_linear.columns), y =␣
↪list(summary_svm_linear.loc['Precision'])), 3, 1)
fig1.add_trace(go.Bar(x = list(summary_svm_linear.columns), y =␣
↪list(summary_svm_linear.loc['FM index'])), 3, 2)
fig1.add_trace(go.Bar(x = list(summary_svm_linear.columns), y =␣
↪list(summary_svm_linear.loc['Accuracy'])), 4, 1)
fig1.add_trace(go.Bar(x = list(summary_svm_linear.columns), y =␣
↪list(summary_svm_linear.loc['Specificity'])), 4, 2)
```

```
fig1.update_layout(height = 2000, width = 800, coloraxis =
 ↪dict(colorscale='Bluered_r'), showlegend = False)
fig1.show()
```

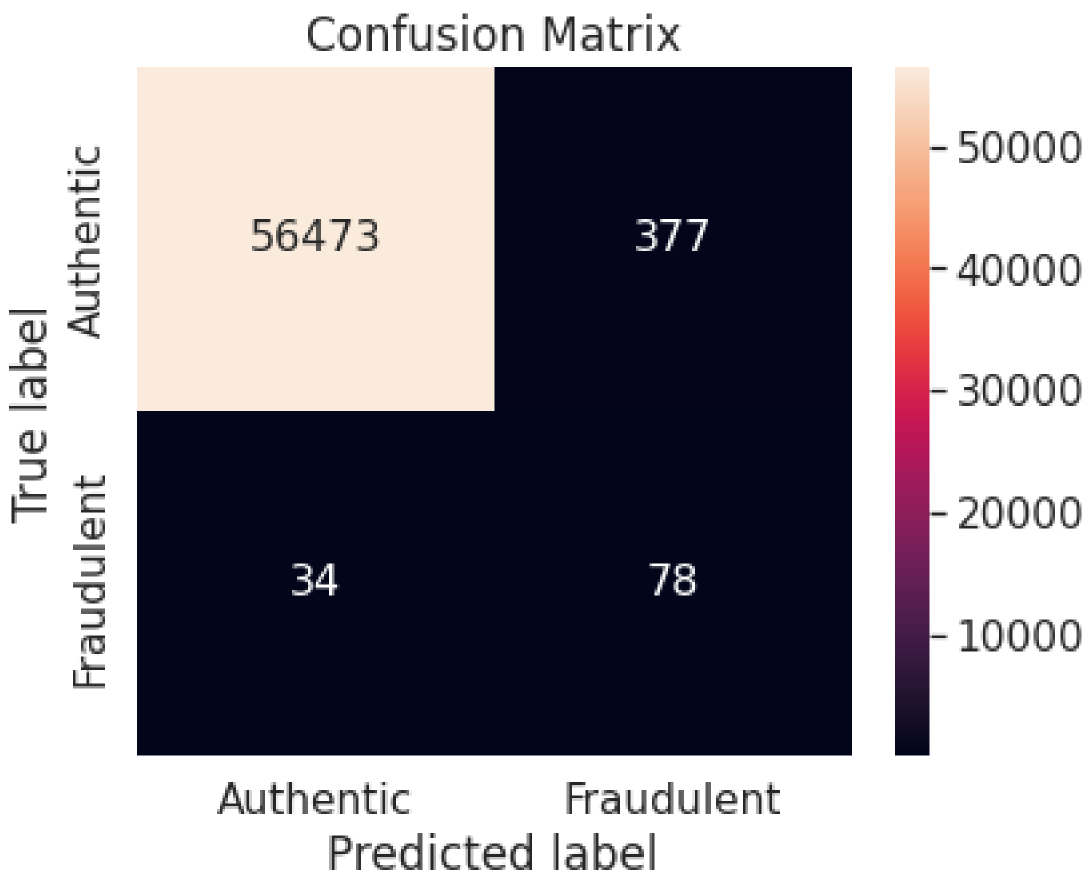

MCC
F1-Score
F2-Score
Recall
Precision
FM index
Specificity
G-mean
0
0.2
0.4
0.6
0.8
1
Unaltered
RUS
ROS
RUS-IL
ROS-IL
SMOTE
NM

9. Naive Bayes

[66]:
```
nb = GaussianNB()
```

1.37 Unaltered training set

[67]:
```
# Elements of confusion matrix

classification(nb, X_train, y_train, X_test, y_test)

# Summary of evaluation metrics

summary_nb_unaltered = summary.copy()
summary_nb_unaltered.set_index('Metric')

y_pred_proba = nb.predict_proba(X_test)[::,1]
roc_auc = metrics.roc_auc_score(y_test, y_pred_proba)

summary_nb_unaltered_extended = summary.copy()
summary_nb_unaltered_extended.loc[len(summary_nb_unaltered_extended.index)] =␣
 ↪['ROC-AUC', roc_auc]
summary_nb_unaltered_extended.set_index('Metric')

summary_nb_unaltered_index = summary_nb_unaltered_extended.T
summary_nb_unaltered_index.columns = summary_nb_unaltered_index.iloc[0]
summary_nb_unaltered_index.drop(summary_nb_unaltered_index.index[0], inplace =␣
 ↪True)
summary_nb_unaltered_index
```



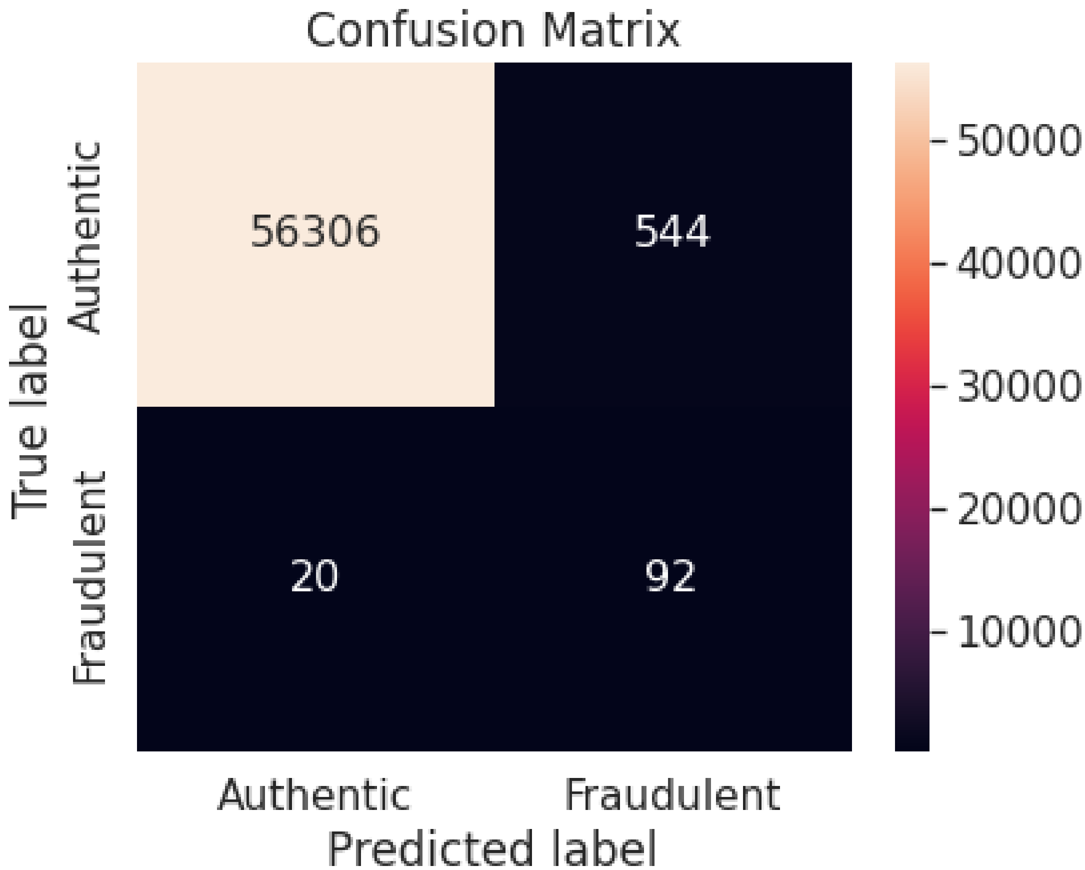
[67]:

Metric	MCC	F1-Score	F2-Score	Recall	Precision	FM index
Performance score	0.343272	0.275132	0.431894	0.696429	0.171429	0.345525

Metric	Specificity	G-mean	F0.5-Score	Accuracy	ROC-AUC
Performance score	0.993369	0.831751	0.201863	0.992785	0.97362

1.38 Random under-sampling

[68]:
```
# Elements of confusion matrix

classification(nb, X_train_under, y_train_under, X_test, y_test)

# Summary of evaluation metrics

summary_nb_under = summary.copy()
summary_nb_under.set_index('Metric')

y_pred_proba = nb.predict_proba(X_test)[::,1]
roc_auc = metrics.roc_auc_score(y_test, y_pred_proba)

summary_nb_under_extended = summary.copy()
summary_nb_under_extended.loc[len(summary_nb_under_extended.index)] =␣
 ↪['ROC-AUC', roc_auc]
summary_nb_under_extended.set_index('Metric')

summary_nb_under_index = summary_nb_under_extended.T
summary_nb_under_index.columns = summary_nb_under_index.iloc[0]
summary_nb_under_index.drop(summary_nb_under_index.index[0], inplace = True)
summary_nb_under_index
```



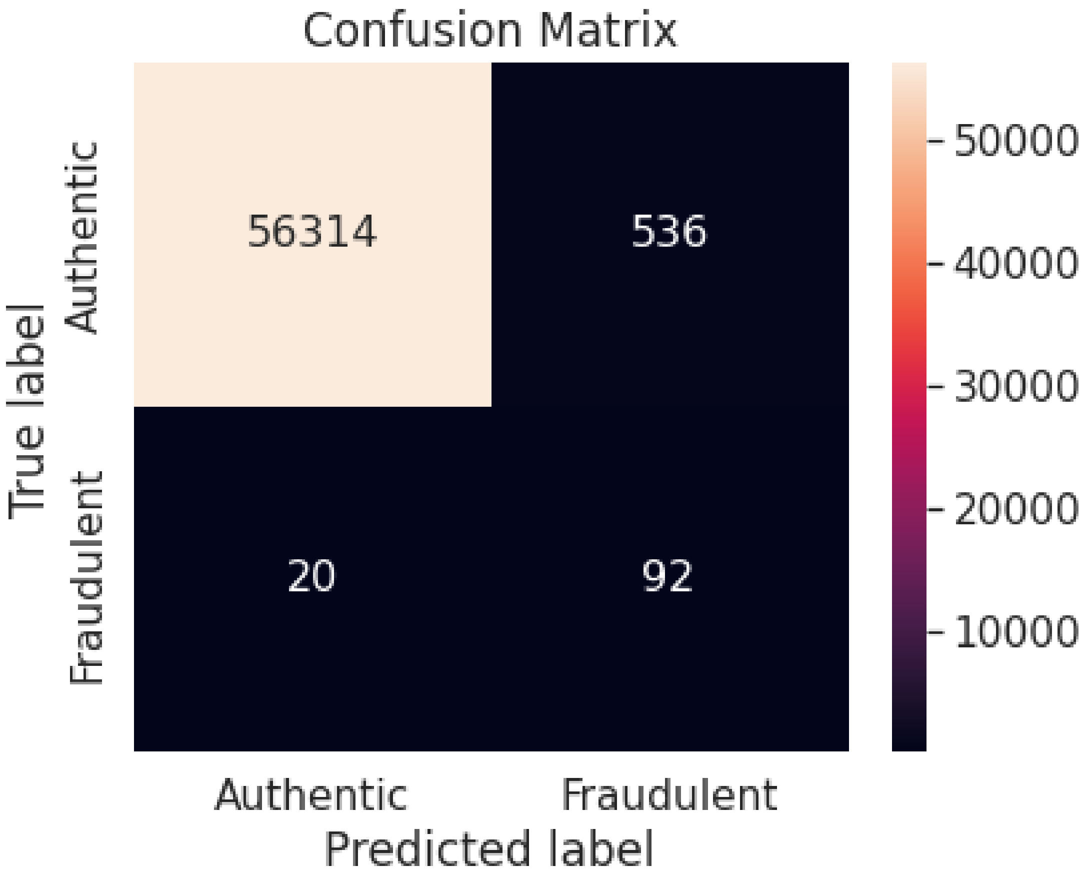
[68]:

Metric	MCC	F1-Score	F2-Score	Recall	Precision	FM index
Performance score	0.342273	0.245989	0.424354	0.821429	0.144654	0.344707

Metric	Specificity	G-mean	F0.5-Score	Accuracy	ROC-AUC
Performance score	0.990431	0.90198	0.173193	0.990099	0.973194

1.39 Random over-sampling

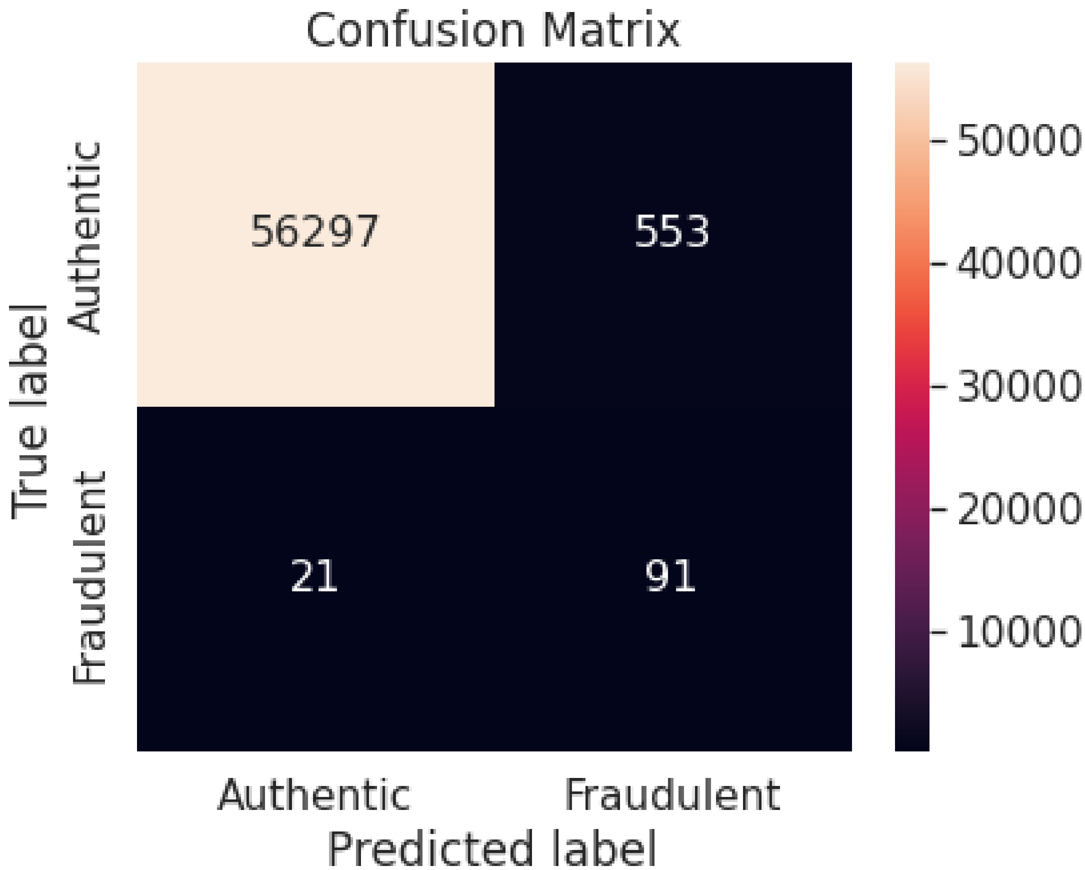
[69]:
```
# Elements of confusion matrix

classification(nb, X_train_over, y_train_over, X_test, y_test)

# Summary of evaluation metrics

summary_nb_over = summary.copy()
summary_nb_over.set_index('Metric')

y_pred_proba = nb.predict_proba(X_test)[::,1]
roc_auc = metrics.roc_auc_score(y_test, y_pred_proba)

summary_nb_over_extended = summary.copy()
summary_nb_over_extended.loc[len(summary_nb_over_extended.index)] = ['ROC-AUC',␣
↪roc_auc]
summary_nb_over_extended.set_index('Metric')

summary_nb_over_index = summary_nb_over_extended.T
summary_nb_over_index.columns = summary_nb_over_index.iloc[0]
summary_nb_over_index.drop(summary_nb_over_index.index[0], inplace = True)
summary_nb_over_index
```




```
[69]: Metric                   MCC   F1-Score  F2-Score    Recall  Precision  FM index  \
      Performance score  0.344481  0.248649  0.427509  0.821429   0.146497  0.346896

      Metric             Specificity    G-mean  F0.5-Score  Accuracy   ROC-AUC
      Performance score     0.990572  0.902044    0.175305  0.990239  0.973634
```

1.40 Random under-sampling with imbalanced-learn library

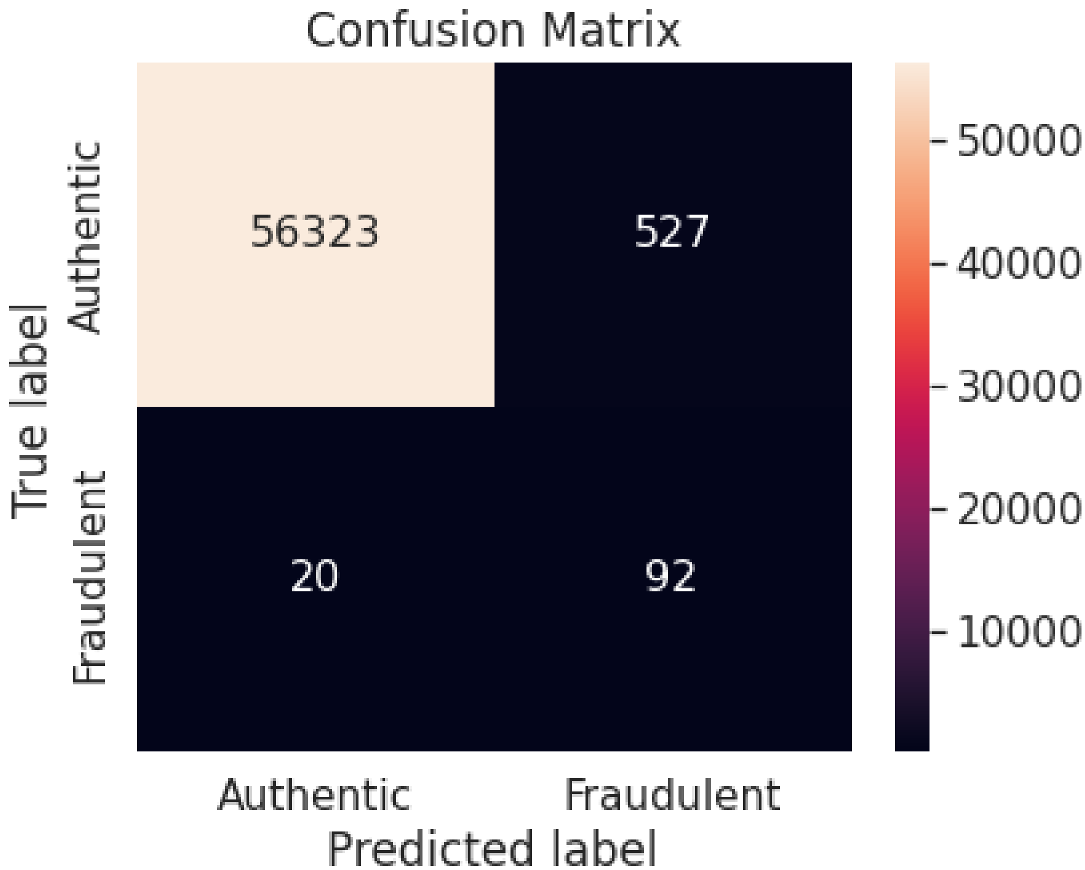
[70]:
```
# Elements of confusion matrix

classification(nb, X_train_under_imblearn, y_train_under_imblearn, X_test,␣
↪y_test)

# Summary of evaluation metrics

summary_nb_under_imblearn = summary.copy()
summary_nb_under_imblearn.set_index('Metric')

y_pred_proba = nb.predict_proba(X_test)[::,1]
roc_auc = metrics.roc_auc_score(y_test, y_pred_proba)

summary_nb_under_imblearn_extended = summary.copy()
summary_nb_under_imblearn_extended.loc[len(summary_nb_under_imblearn_extended.
↪index)] = ['ROC-AUC', roc_auc]
summary_nb_under_imblearn_extended.set_index('Metric')

summary_nb_under_imblearn_index = summary_nb_under_imblearn_extended.T
summary_nb_under_imblearn_index.columns = summary_nb_under_imblearn_index.
↪iloc[0]
summary_nb_under_imblearn_index.drop(summary_nb_under_imblearn_index.index[0],␣
↪inplace = True)
summary_nb_under_imblearn_index
```




```
[70]: Metric                   MCC  F1-Score  F2-Score  Recall Precision  FM index  \
      Performance score  0.336357  0.240741  0.416667  0.8125  0.141304  0.338836

      Metric             Specificity    G-mean F0.5-Score  Accuracy   ROC-AUC
      Performance score     0.990273  0.896993   0.169271  0.989923  0.973777
```

1.41 Random over-sampling with imbalanced-learn library

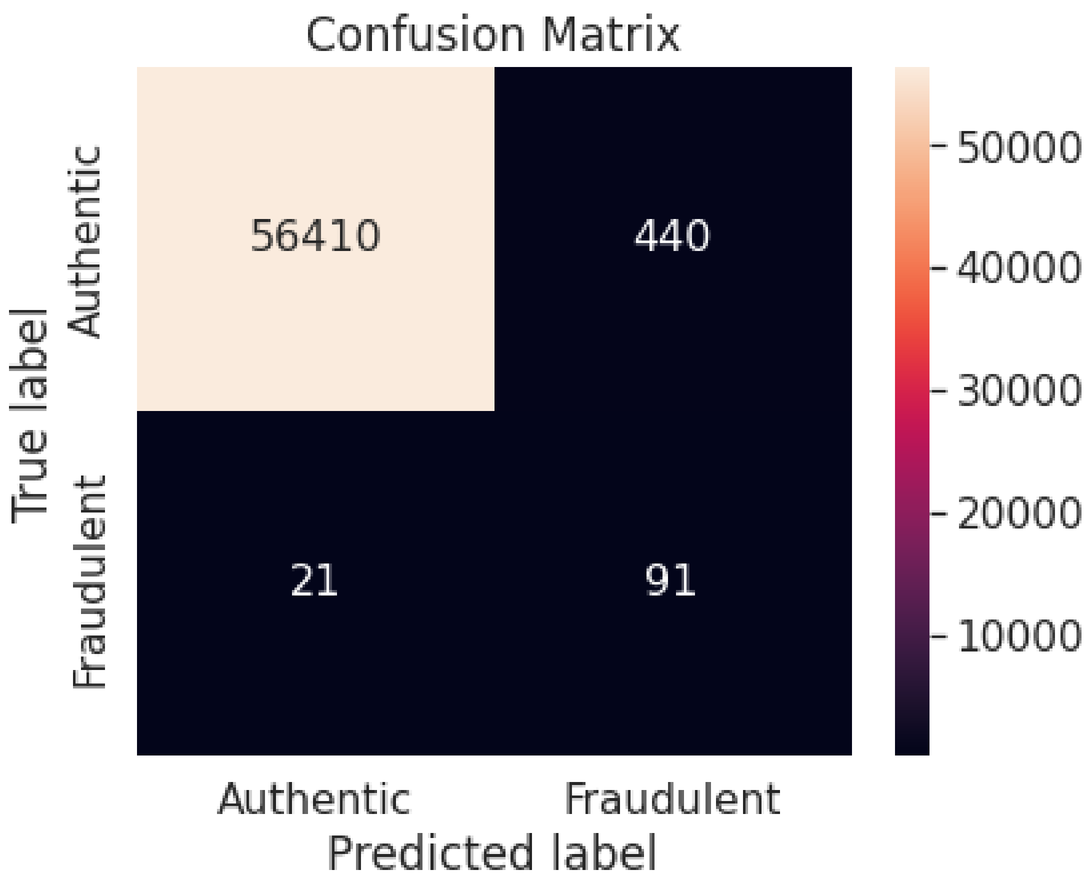
[71]:
```
# Elements of confusion matrix

classification(nb, X_train_over_imblearn, y_train_over_imblearn, X_test, y_test)

# Summary of evaluation metrics

summary_nb_over_imblearn = summary.copy()
summary_nb_over_imblearn.set_index('Metric')

y_pred_proba = nb.predict_proba(X_test)[::,1]
roc_auc = metrics.roc_auc_score(y_test, y_pred_proba)

summary_nb_over_imblearn_extended = summary.copy()
summary_nb_over_imblearn_extended.loc[len(summary_nb_over_imblearn_extended.
 ↪index)] = ['ROC-AUC', roc_auc]
summary_nb_over_imblearn_extended.set_index('Metric')

summary_nb_over_imblearn_index = summary_nb_over_imblearn_extended.T
summary_nb_over_imblearn_index.columns = summary_nb_over_imblearn_index.iloc[0]
summary_nb_over_imblearn_index.drop(summary_nb_over_imblearn_index.index[0],␣
 ↪inplace = True)
summary_nb_over_imblearn_index
```




```
[71]: Metric                      MCC F1-Score  F2-Score    Recall Precision  FM index  \
      Performance score  0.347016  0.25171  0.431115  0.821429  0.148627  0.349409

      Metric             Specificity    G-mean F0.5-Score  Accuracy   ROC-AUC
      Performance score      0.99073  0.902116   0.177743  0.990397  0.973635
```

1.42 Synthetic minority over-sampling technique (SMOTE)

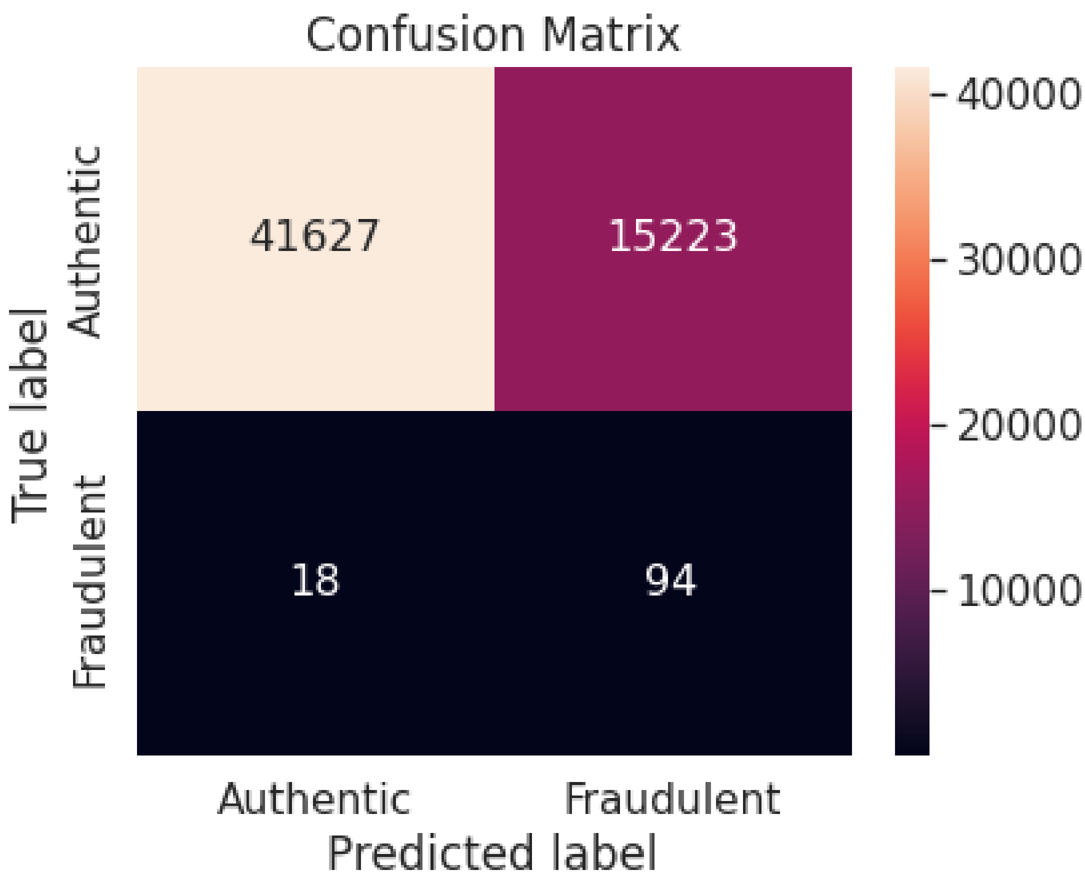
[72]:
```
# Elements of confusion matrix

classification(nb, X_train_over_smote, y_train_over_smote, X_test, y_test)

# Summary of evaluation metrics

summary_nb_over_smote = summary.copy()
summary_nb_over_smote.set_index('Metric')

y_pred_proba = nb.predict_proba(X_test)[::,1]
roc_auc = metrics.roc_auc_score(y_test, y_pred_proba)

summary_nb_over_smote_extended = summary.copy()
summary_nb_over_smote_extended.loc[len(summary_nb_over_smote_extended.index)] =␣
↪['ROC-AUC', roc_auc]
summary_nb_over_smote_extended.set_index('Metric')

summary_nb_over_smote_index = summary_nb_over_smote_extended.T
summary_nb_over_smote_index.columns = summary_nb_over_smote_index.iloc[0]
summary_nb_over_smote_index.drop(summary_nb_over_smote_index.index[0], inplace␣
↪= True)
summary_nb_over_smote_index
```

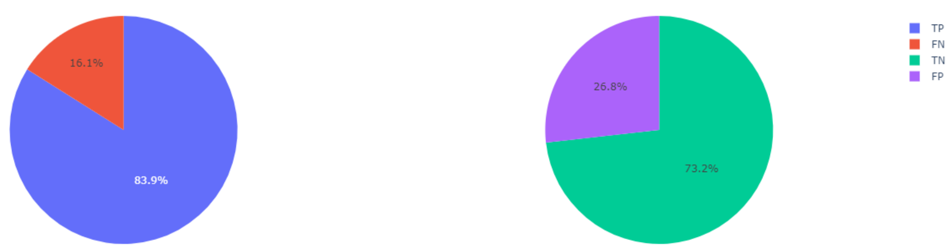



```
[72]: Metric                  MCC  F1-Score F2-Score  Recall Precision  FM index  \
      Performance score  0.370966  0.283048  0.46476  0.8125  0.171375  0.373151

      Metric             Specificity    G-mean F0.5-Score  Accuracy   ROC-AUC
      Performance score      0.99226  0.897893    0.203488  0.991907  0.969033
```

1.43 Under-sampling via NearMiss

[73]:
```
# Elements of confusion matrix

classification(nb, X_train_under_nm, y_train_under_nm, X_test, y_test)

# Summary of evaluation metrics

summary_nb_under_nm = summary.copy()
summary_nb_under_nm.set_index('Metric')

y_pred_proba = nb.predict_proba(X_test)[::,1]
roc_auc = metrics.roc_auc_score(y_test, y_pred_proba)

summary_nb_under_nm_extended = summary.copy()
summary_nb_under_nm_extended.loc[len(summary_nb_under_nm_extended.index)] =␣
↪['ROC-AUC', roc_auc]
summary_nb_under_nm_extended.set_index('Metric')

summary_nb_under_nm_index = summary_nb_under_nm_extended.T
summary_nb_under_nm_index.columns = summary_nb_under_nm_index.iloc[0]
summary_nb_under_nm_index.drop(summary_nb_under_nm_index.index[0], inplace =␣
↪True)
summary_nb_under_nm_index
```





[73]:

Metric	MCC	F1-Score	F2-Score	Recall	Precision	FM index
Performance score	0.057099	0.012185	0.029813	0.839286	0.006137	0.071768

Metric	Specificity	G-mean	F0.5-Score	Accuracy	ROC-AUC
Performance score	0.732225	0.78393	0.007657	0.732436	0.831196

1.44 Summary of naive Bayes models

[74]:
```
summary_nb = pd.DataFrame(columns = ['Metric'])

summary_nb['Metric'] = EvalMetricLabels
summary_nb_list = [summary_nb_unaltered, summary_nb_under, summary_nb_over,␣
 ↪summary_nb_under_imblearn,
                   summary_nb_over_imblearn, summary_nb_over_smote,␣
 ↪summary_nb_under_nm]

for i in summary_nb_list:
    summary_nb = pd.merge(summary_nb, i, on = 'Metric')

TrainingSetsMetric = TrainingSets.copy()
TrainingSetsMetric.insert(0, 'Metric')

summary_nb.columns = TrainingSetsMetric
summary_nb.set_index('Metric', inplace = True)
summary_nb
```

[74]:
```
             Unaltered       RUS       ROS    RUS-IL    ROS-IL     SMOTE  \
Metric
MCC           0.343272  0.342273  0.344481  0.336357  0.347016  0.370966
F1-Score      0.275132  0.245989  0.248649  0.240741  0.251710  0.283048
F2-Score      0.431894  0.424354  0.427509  0.416667  0.431115  0.464760
Recall        0.696429  0.821429  0.821429  0.812500  0.821429  0.812500
Precision     0.171429  0.144654  0.146497  0.141304  0.148627  0.171375
FM index      0.345525  0.344707  0.346896  0.338836  0.349409  0.373151
Specificity   0.993369  0.990431  0.990572  0.990273  0.990730  0.992260
G-mean        0.831751  0.901980  0.902044  0.896993  0.902116  0.897893
F0.5-Score    0.201863  0.173193  0.175305  0.169271  0.177743  0.203488
Accuracy      0.992785  0.990099  0.990239  0.989923  0.990397  0.991907

                    NM
Metric
MCC           0.057099
F1-Score      0.012185
F2-Score      0.029813
Recall        0.839286
Precision     0.006137
FM index      0.071768
Specificity   0.732225
G-mean        0.783930
F0.5-Score    0.007657
Accuracy      0.732436
```

[75]:
```
# Visual comparison of the model applied on different training sets through␣
↪various evaluation metrics

summary_visual(summary_nb)
```

10. Random Forest

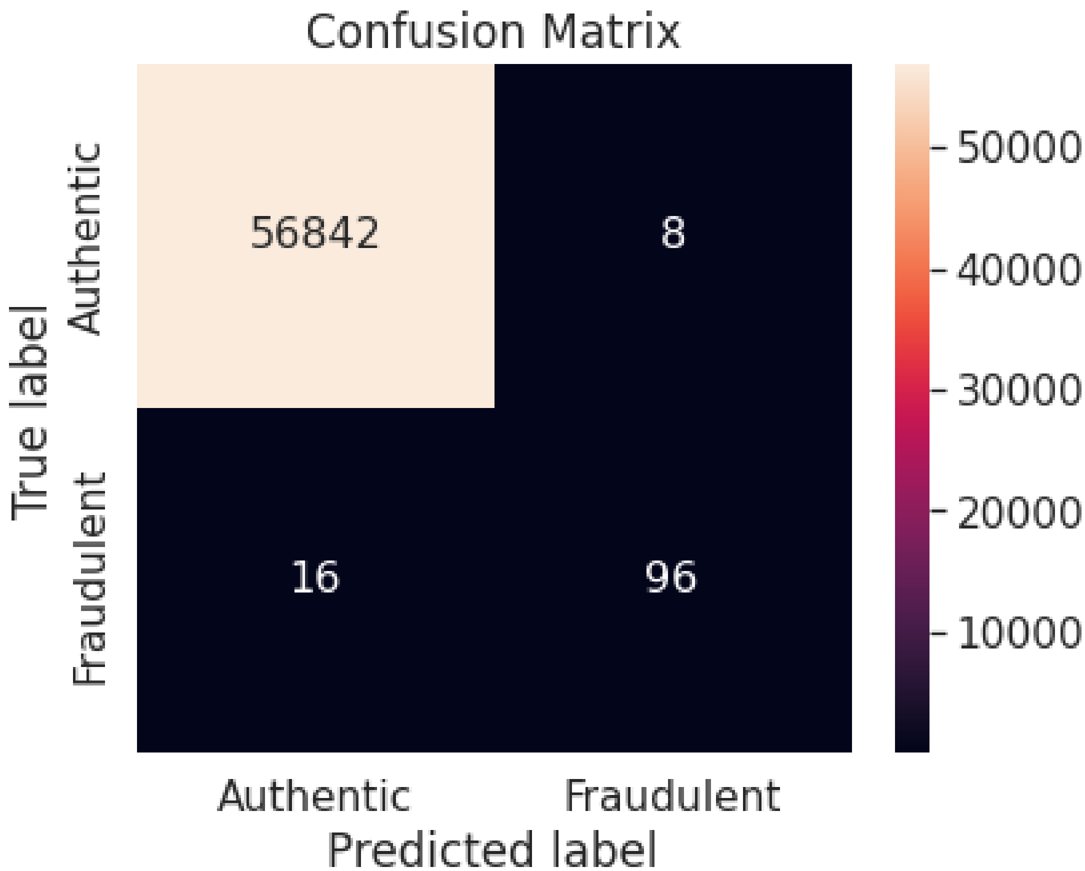
The Random Forest classifier employs multiple decision trees, thereby avoiding the reliance upon feature selection of a singular decision tree.

[76]:
```
rf = RandomForestClassifier(n_estimators = 100)
```

1.45 Unaltered training set

[77]:
```
# Elements of confusion matrix

classification(rf, X_train, y_train, X_test, y_test)

# Summary of evaluation metrics

summary_rf_unaltered = summary.copy()
summary_rf_unaltered.set_index('Metric')

y_pred_proba = rf.predict_proba(X_test)[::,1]
roc_auc = metrics.roc_auc_score(y_test, y_pred_proba)

summary_rf_unaltered_extended = summary.copy()
summary_rf_unaltered_extended.loc[len(summary_rf_unaltered_extended.index)] =␣
↪['ROC-AUC', roc_auc]
summary_rf_unaltered_extended.set_index('Metric')
```

```
summary_rf_unaltered_index = summary_rf_unaltered_extended.T
summary_rf_unaltered_index.columns = summary_rf_unaltered_index.iloc[0]
summary_rf_unaltered_index.drop(summary_rf_unaltered_index.index[0], inplace = True)
summary_rf_unaltered_index
```



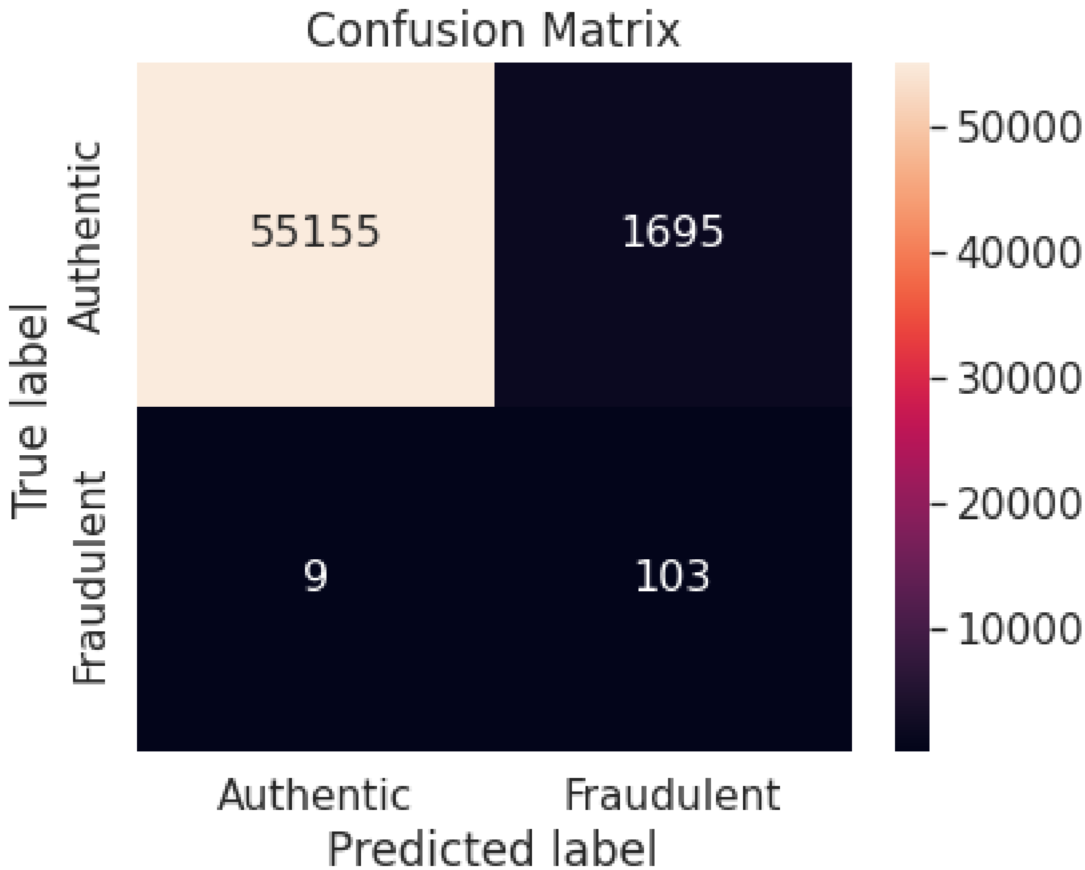
[77]:

```
Metric                   MCC  F1-Score  F2-Score    Recall Precision  FM index  \
Performance score  0.889291  0.888889  0.869565  0.857143  0.923077  0.889499

Metric             Specificity    G-mean F0.5-Score  Accuracy   ROC-AUC
Performance score     0.999859  0.925755   0.909091  0.999579  0.958735
```

1.46 Random under-sampling

[78]:

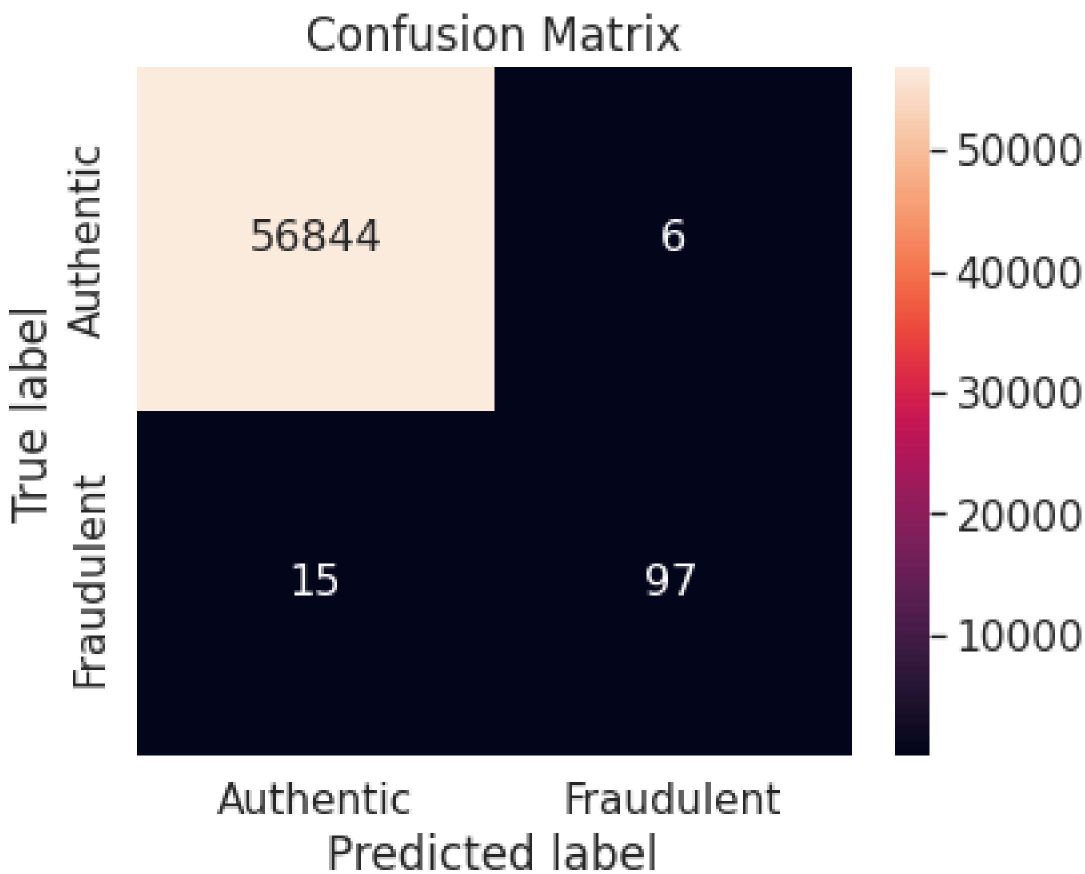
```
# Elements of confusion matrix

classification(rf, X_train_under, y_train_under, X_test, y_test)

# Summary of evaluation metrics

summary_rf_under = summary.copy()
summary_rf_under.set_index('Metric')

y_pred_proba = rf.predict_proba(X_test)[::,1]
roc_auc = metrics.roc_auc_score(y_test, y_pred_proba)

summary_rf_under_extended = summary.copy()
summary_rf_under_extended.loc[len(summary_rf_under_extended.index)] =␣
 ↪['ROC-AUC', roc_auc]
summary_rf_under_extended.set_index('Metric')

summary_rf_under_index = summary_rf_under_extended.T
summary_rf_under_index.columns = summary_rf_under_index.iloc[0]
summary_rf_under_index.drop(summary_rf_under_index.index[0], inplace = True)
summary_rf_under_index
```



[78]:
```
Metric                  MCC  F1-Score  F2-Score    Recall Precision  FM index  \
Performance score  0.225454  0.107853  0.229297  0.919643  0.057286  0.229527

Metric             Specificity    G-mean F0.5-Score  Accuracy   ROC-AUC
Performance score     0.970185  0.944576   0.070509  0.970085  0.981881
```

1.47 Random over-sampling

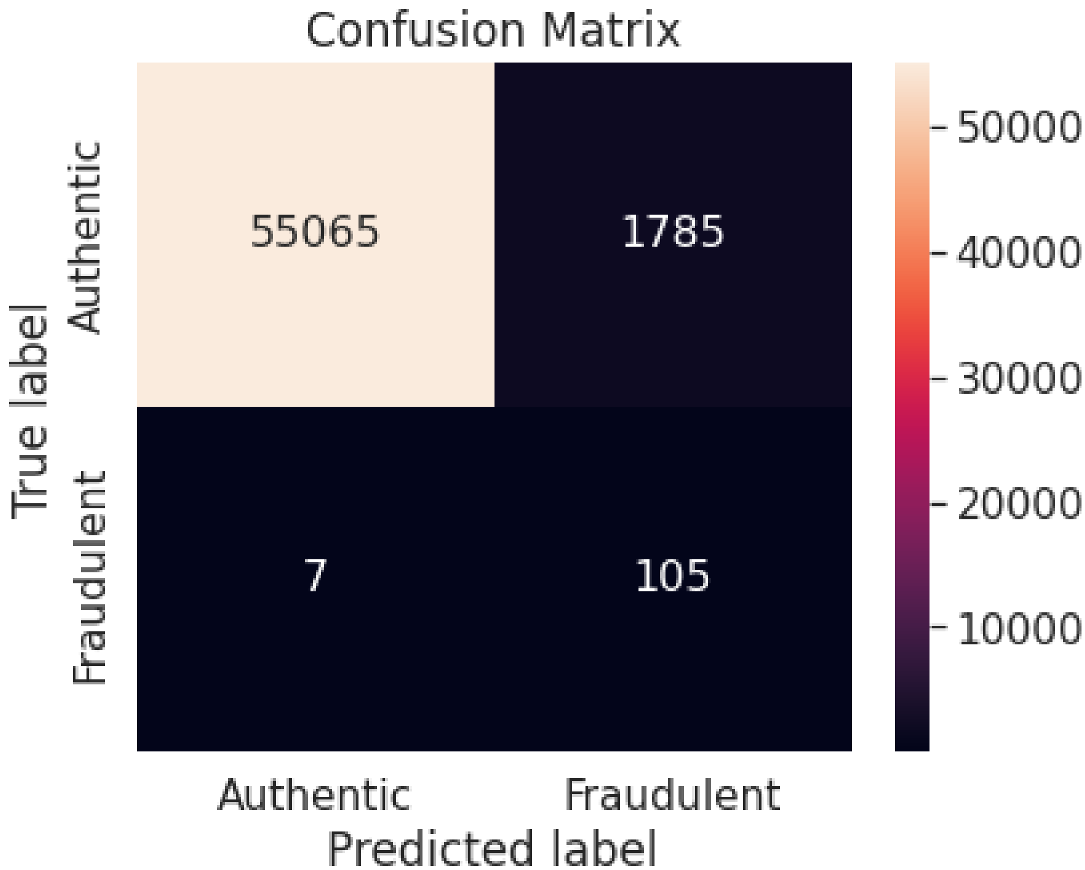
[79]:
```
# Elements of confusion matrix

classification(rf, X_train_over, y_train_over, X_test, y_test)

# Summary of evaluation metrics

summary_rf_over = summary.copy()
summary_rf_over.set_index('Metric')

y_pred_proba = rf.predict_proba(X_test)[::,1]
roc_auc = metrics.roc_auc_score(y_test, y_pred_proba)

summary_rf_over_extended = summary.copy()
summary_rf_over_extended.loc[len(summary_rf_over_extended.index)] = ['ROC-AUC',␣
↪roc_auc]
summary_rf_over_extended.set_index('Metric')

summary_rf_over_index = summary_rf_over_extended.T
summary_rf_over_index.columns = summary_rf_over_index.iloc[0]
summary_rf_over_index.drop(summary_rf_over_index.index[0], inplace = True)
summary_rf_over_index
```



[79]:
```
Metric                  MCC  F1-Score  F2-Score    Recall Precision  FM index  \
Performance score  0.902936  0.902326  0.880218  0.866071  0.941748  0.903117

Metric             Specificity   G-mean F0.5-Score  Accuracy   ROC-AUC
Performance score     0.999894  0.93058   0.925573  0.999631  0.958542
```

1.48 Random under-sampling with imbalanced-learn library

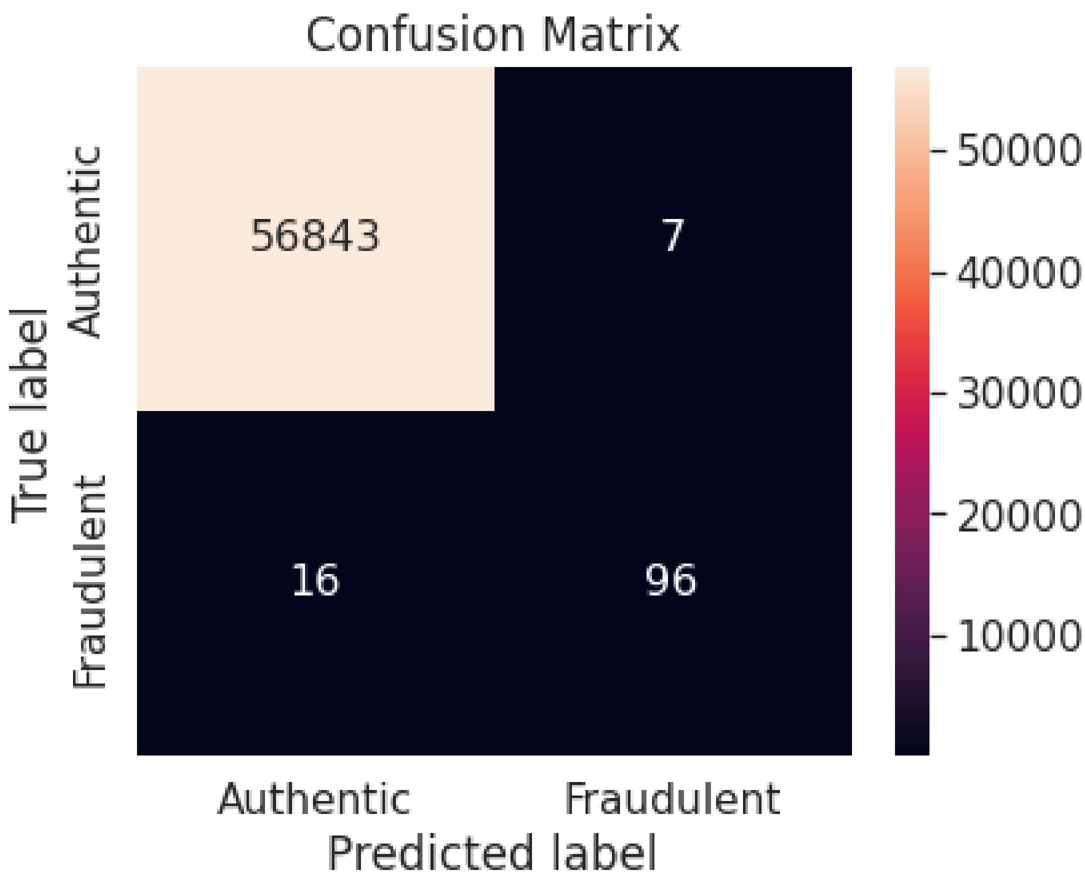
[80]:
```
# Elements of confusion matrix

classification(rf, X_train_under_imblearn, y_train_under_imblearn, X_test,␣
 ↪y_test)

# Summary of evaluation metrics

summary_rf_under_imblearn = summary.copy()
summary_rf_under_imblearn.set_index('Metric')

y_pred_proba = rf.predict_proba(X_test)[::,1]
roc_auc = metrics.roc_auc_score(y_test, y_pred_proba)

summary_rf_under_imblearn_extended = summary.copy()
summary_rf_under_imblearn_extended.loc[len(summary_rf_under_imblearn_extended.
 ↪index)] = ['ROC-AUC', roc_auc]
summary_rf_under_imblearn_extended.set_index('Metric')

summary_rf_under_imblearn_index = summary_rf_under_imblearn_extended.T
summary_rf_under_imblearn_index.columns = summary_rf_under_imblearn_index.
 ↪iloc[0]
summary_rf_under_imblearn_index.drop(summary_rf_under_imblearn_index.index[0],␣
 ↪inplace = True)
summary_rf_under_imblearn_index
```



[80]:
```
Metric                  MCC  F1-Score  F2-Score  Recall Precision  FM index  \
Performance score  0.224107  0.104895  0.224551  0.9375  0.055556  0.228218

Metric            Specificity    G-mean F0.5-Score Accuracy   ROC-AUC
Performance score    0.968602  0.952924   0.068431  0.96854  0.981963
```

1.49 Random over-sampling with imbalanced-learn library

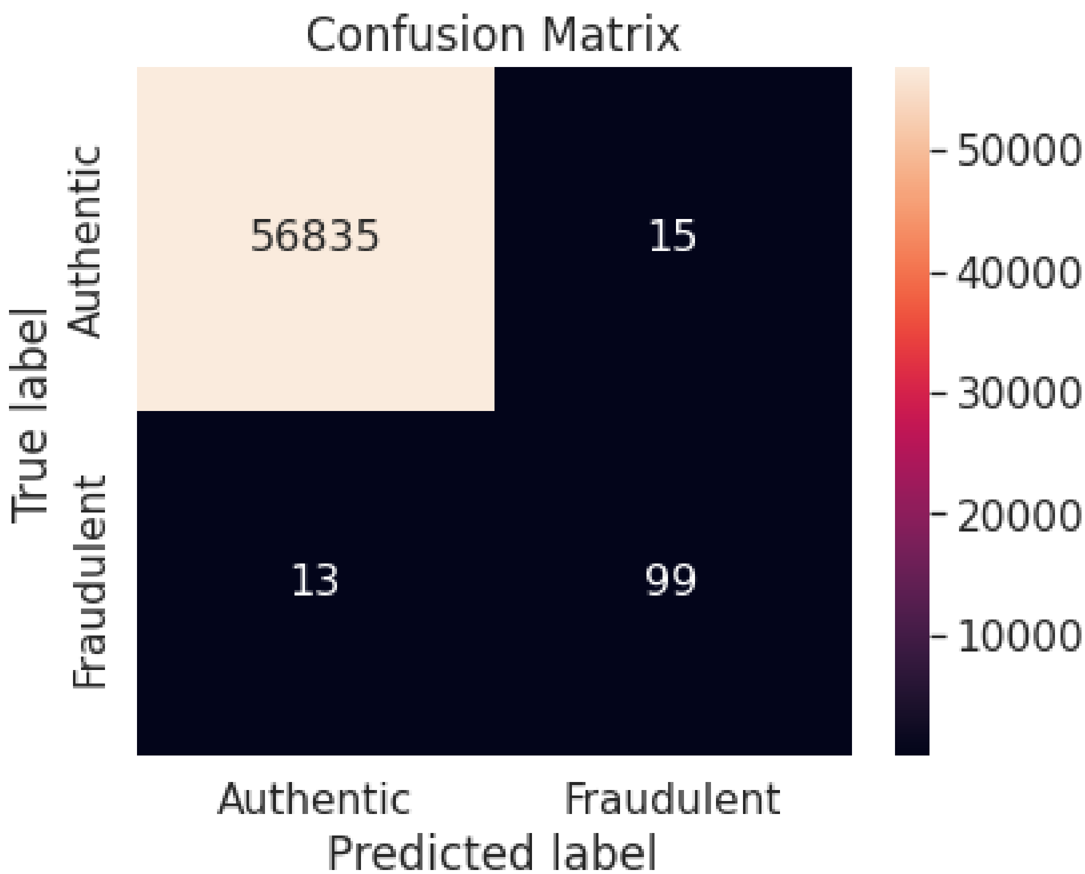
[81]:
```
# Elements of confusion matrix

classification(rf, X_train_over_imblearn, y_train_over_imblearn, X_test, y_test)

# Summary of evaluation metrics

summary_rf_over_imblearn = summary.copy()
summary_rf_over_imblearn.set_index('Metric')

y_pred_proba = rf.predict_proba(X_test)[::,1]
roc_auc = metrics.roc_auc_score(y_test, y_pred_proba)

summary_rf_over_imblearn_extended = summary.copy()
summary_rf_over_imblearn_extended.loc[len(summary_rf_over_imblearn_extended.
 ↪index)] = ['ROC-AUC', roc_auc]
summary_rf_over_imblearn_extended.set_index('Metric')

summary_rf_over_imblearn_index = summary_rf_over_imblearn_extended.T
summary_rf_over_imblearn_index.columns = summary_rf_over_imblearn_index.iloc[0]
summary_rf_over_imblearn_index.drop(summary_rf_over_imblearn_index.index[0],␣
 ↪inplace = True)
summary_rf_over_imblearn_index
```



[81]:
```
Metric                    MCC  F1-Score  F2-Score    Recall Precision  FM index  \
Performance score  0.893608  0.893023  0.871143  0.857143  0.932039  0.893807

Metric            Specificity    G-mean F0.5-Score  Accuracy   ROC-AUC
Performance score    0.999877  0.925763   0.916031  0.999596  0.962946
```

1.50 Synthetic minority over-sampling technique (SMOTE)

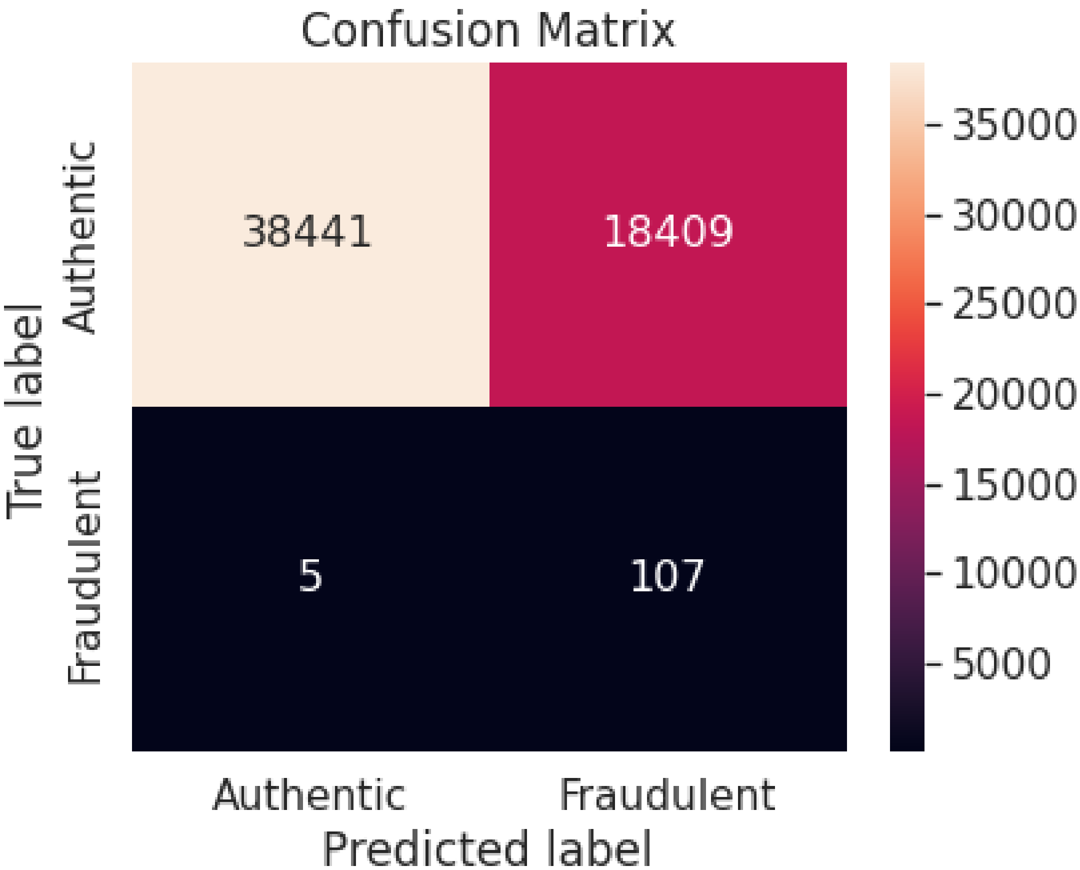
[82]:
```
# Elements of confusion matrix

classification(rf, X_train_over_smote, y_train_over_smote, X_test, y_test)

# Summary of evaluation metrics

summary_rf_over_smote = summary.copy()
summary_rf_over_smote.set_index('Metric')

y_pred_proba = rf.predict_proba(X_test)[::,1]
roc_auc = metrics.roc_auc_score(y_test, y_pred_proba)

summary_rf_over_smote_extended = summary.copy()
summary_rf_over_smote_extended.loc[len(summary_rf_over_smote_extended.index)] =
 ↪['ROC-AUC', roc_auc]
summary_rf_over_smote_extended.set_index('Metric')

summary_rf_over_smote_index = summary_rf_over_smote_extended.T
summary_rf_over_smote_index.columns = summary_rf_over_smote_index.iloc[0]
summary_rf_over_smote_index.drop(summary_rf_over_smote_index.index[0], inplace
 ↪= True)
summary_rf_over_smote_index
```

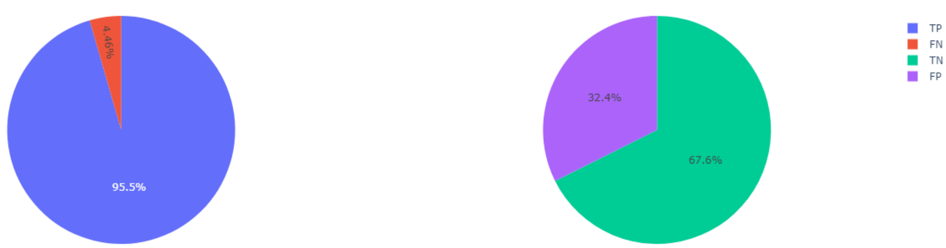


[82]:
```
Metric                   MCC  F1-Score  F2-Score    Recall Precision  FM index  \
Performance score  0.875894  0.876106  0.880783  0.883929  0.868421  0.876141

Metric             Specificity    G-mean F0.5-Score  Accuracy   ROC-AUC
Performance score     0.999736  0.940051    0.871479  0.999508  0.973488
```

1.51 Under-sampling via NearMiss

[83]:
```
# Elements of confusion matrix

classification(rf, X_train_under_nm, y_train_under_nm, X_test, y_test)

# Summary of evaluation metrics

summary_rf_under_nm = summary.copy()
summary_rf_under_nm.set_index('Metric')

y_pred_proba = rf.predict_proba(X_test)[::,1]
roc_auc = metrics.roc_auc_score(y_test, y_pred_proba)

summary_rf_under_nm_extended = summary.copy()
summary_rf_under_nm_extended.loc[len(summary_rf_under_nm_extended.index)] =
 ↪['ROC-AUC', roc_auc]
summary_rf_under_nm_extended.set_index('Metric')

summary_rf_under_nm_index = summary_rf_under_nm_extended.T
summary_rf_under_nm_index.columns = summary_rf_under_nm_index.iloc[0]
summary_rf_under_nm_index.drop(summary_rf_under_nm_index.index[0], inplace =
 ↪True)
summary_rf_under_nm_index
```





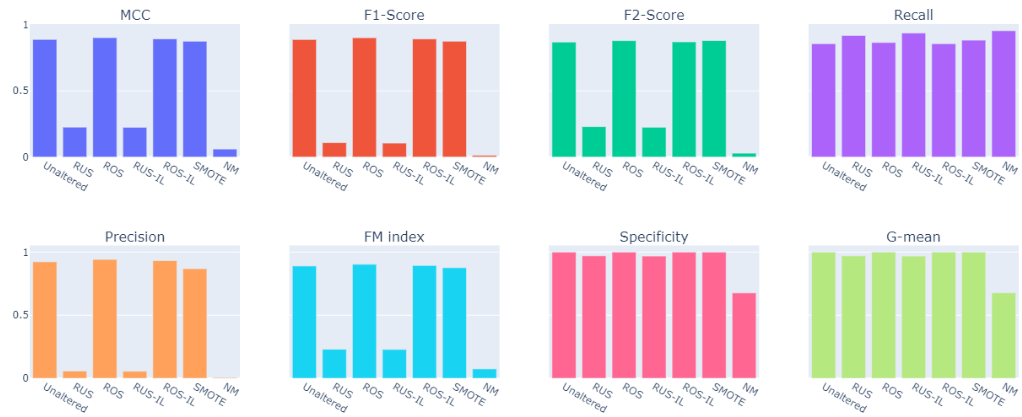
[83]:
```
Metric                  MCC  F1-Score  F2-Score    Recall Precision  FM index  \
Performance score  0.059728  0.011488  0.028211  0.955357  0.005779  0.074302

Metric             Specificity    G-mean F0.5-Score  Accuracy   ROC-AUC
Performance score     0.676183  0.803739   0.007213  0.676732  0.969865
```

1.52 Summary of random forest models

[84]:
```
summary_rf = pd.DataFrame(columns = ['Metric'])

summary_rf['Metric'] = EvalMetricLabels
summary_rf_list = [summary_rf_unaltered, summary_rf_under, summary_rf_over,␣
 ↪summary_rf_under_imblearn,
                   summary_rf_over_imblearn, summary_rf_over_smote,␣
 ↪summary_rf_under_nm]

for i in summary_rf_list:
    summary_rf = pd.merge(summary_rf, i, on = 'Metric')

TrainingSetsMetric = TrainingSets.copy()
TrainingSetsMetric.insert(0, 'Metric')

summary_rf.columns = TrainingSetsMetric
summary_rf.set_index('Metric', inplace = True)
summary_rf
```

[84]:
```
             Unaltered       RUS       ROS    RUS-IL    ROS-IL     SMOTE  \
Metric
MCC           0.889291  0.225454  0.902936  0.224107  0.893608  0.875894
F1-Score      0.888889  0.107853  0.902326  0.104895  0.893023  0.876106
F2-Score      0.869565  0.229297  0.880218  0.224551  0.871143  0.880783
Recall        0.857143  0.919643  0.866071  0.937500  0.857143  0.883929
Precision     0.923077  0.057286  0.941748  0.055556  0.932039  0.868421
FM index      0.889499  0.229527  0.903117  0.228218  0.893807  0.876141
Specificity   0.999859  0.970185  0.999894  0.968602  0.999877  0.999736
G-mean        0.925755  0.944576  0.930580  0.952924  0.925763  0.940051
F0.5-Score    0.909091  0.070509  0.925573  0.068431  0.916031  0.871479
Accuracy      0.999579  0.970085  0.999631  0.968540  0.999596  0.999508

                   NM
Metric
MCC          0.059728
F1-Score     0.011488
F2-Score     0.028211
Recall       0.955357
Precision    0.005779
FM index     0.074302
Specificity  0.676183
G-mean       0.803739
F0.5-Score   0.007213
Accuracy     0.676732
```

[85]:
```
# Visual comparison of the model applied on different training sets through
    ↪various evaluation metrics

summary_visual(summary_rf)
```



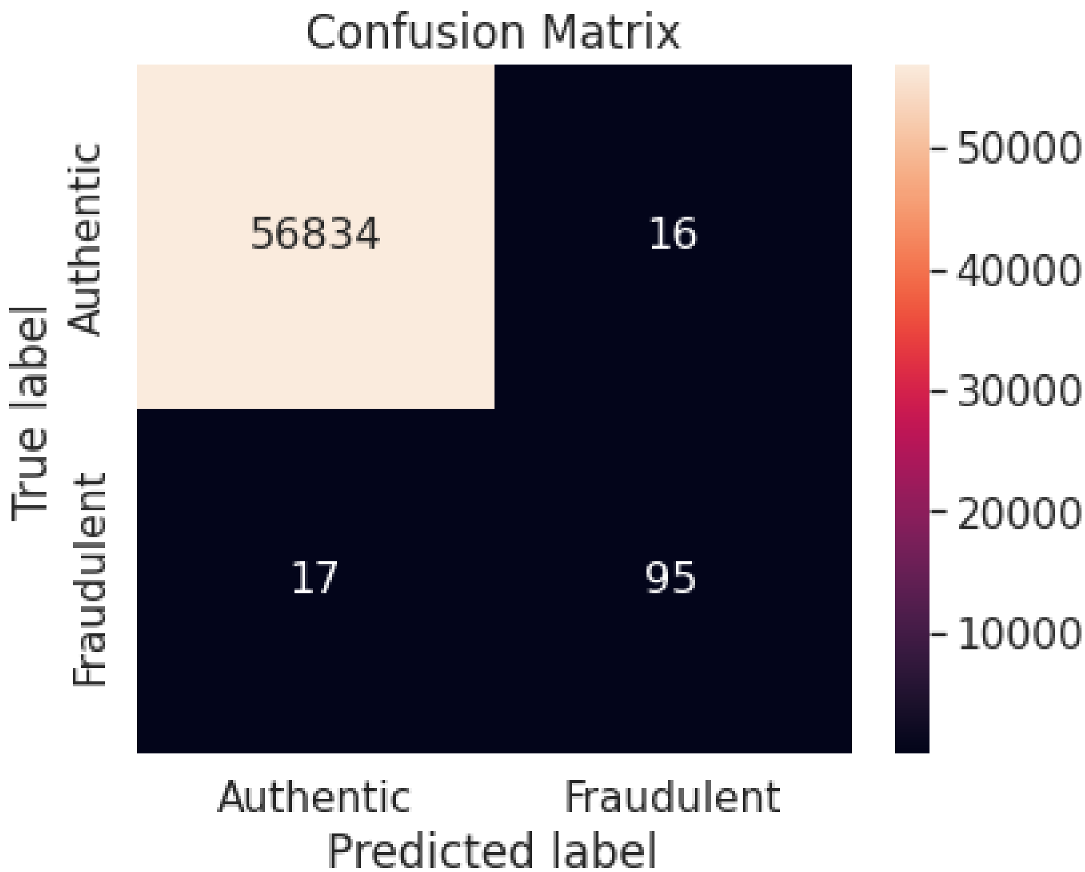
11. Linear discriminant analysis (LDA)

[86]:
```
lda = LinearDiscriminantAnalysis()
```

1.53 Unaltered training set

[87]:
```
# Elements of confusion matrix

classification(lda, X_train, y_train, X_test, y_test)

# Summary of evaluation metrics

summary_lda_unaltered = summary.copy()
summary_lda_unaltered.set_index('Metric')

y_score = lda.decision_function(X_test)
average_precision = average_precision_score(y_test, y_score)
y_pred_proba = lda.predict_proba(X_test)[::,1]
roc_auc = metrics.roc_auc_score(y_test, y_pred_proba)

summary_lda_unaltered_extended = summary.copy()
summary_lda_unaltered_extended.loc[len(summary_lda_unaltered_extended.index)] =
    ↪['AP', average_precision]
summary_lda_unaltered_extended.loc[len(summary_lda_unaltered_extended.index)] =
    ↪['ROC-AUC', roc_auc]
```

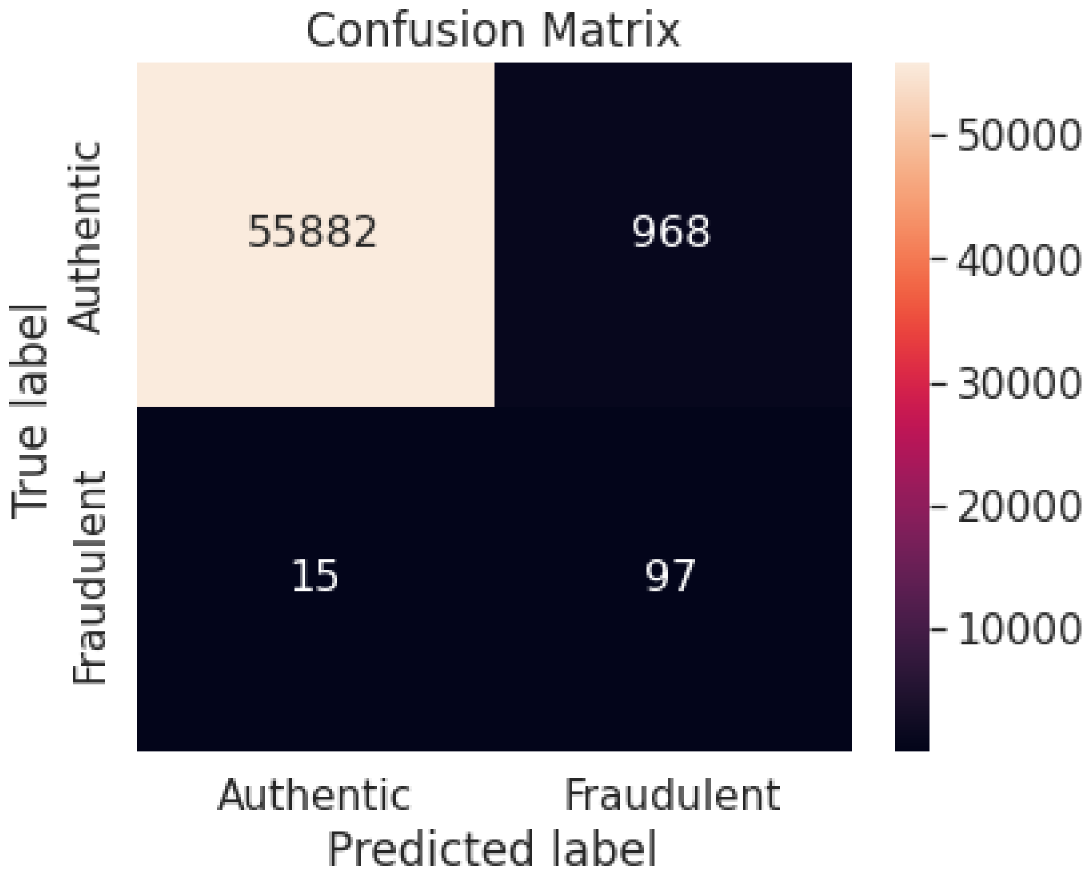
```
summary_lda_unaltered_extended.set_index('Metric')

summary_lda_unaltered_index = summary_lda_unaltered_extended.T
summary_lda_unaltered_index.columns = summary_lda_unaltered_index.iloc[0]
summary_lda_unaltered_index.drop(summary_lda_unaltered_index.index[0], inplace= True)
summary_lda_unaltered_index
```



[87]:
```
Metric                   MCC  F1-Score  F2-Score    Recall Precision  FM index  \
Performance score  0.851736  0.852018  0.849732  0.848214  0.855856  0.852027

Metric            Specificity    G-mean F0.5-Score  Accuracy        AP  \
Performance score    0.999719  0.920856   0.854317  0.999421  0.801019

Metric              ROC-AUC
Performance score  0.981249
```

1.54 Random under-sampling

[88]:
```python
# Elements of confusion matrix

classification(lda, X_train_under, y_train_under, X_test, y_test)

# Summary of evaluation metrics

summary_lda_under = summary
summary_lda_under.set_index('Metric')

y_score = lda.decision_function(X_test)
average_precision = average_precision_score(y_test, y_score)
y_pred_proba = lda.predict_proba(X_test)[::,1]
roc_auc = metrics.roc_auc_score(y_test, y_pred_proba)

summary_lda_under_extended = summary.copy()
summary_lda_under_extended.loc[len(summary_lda_under_extended.index)] = ['AP',␣
 ↪average_precision]
summary_lda_under_extended.loc[len(summary_lda_under_extended.index)] =␣
 ↪['ROC-AUC', roc_auc]
summary_lda_under_extended.set_index('Metric')

summary_lda_under_index = summary_lda_under_extended.T
summary_lda_under_index.columns = summary_lda_under_index.iloc[0]
summary_lda_under_index.drop(summary_lda_under_index.index[0], inplace = True)
summary_lda_under_index
```

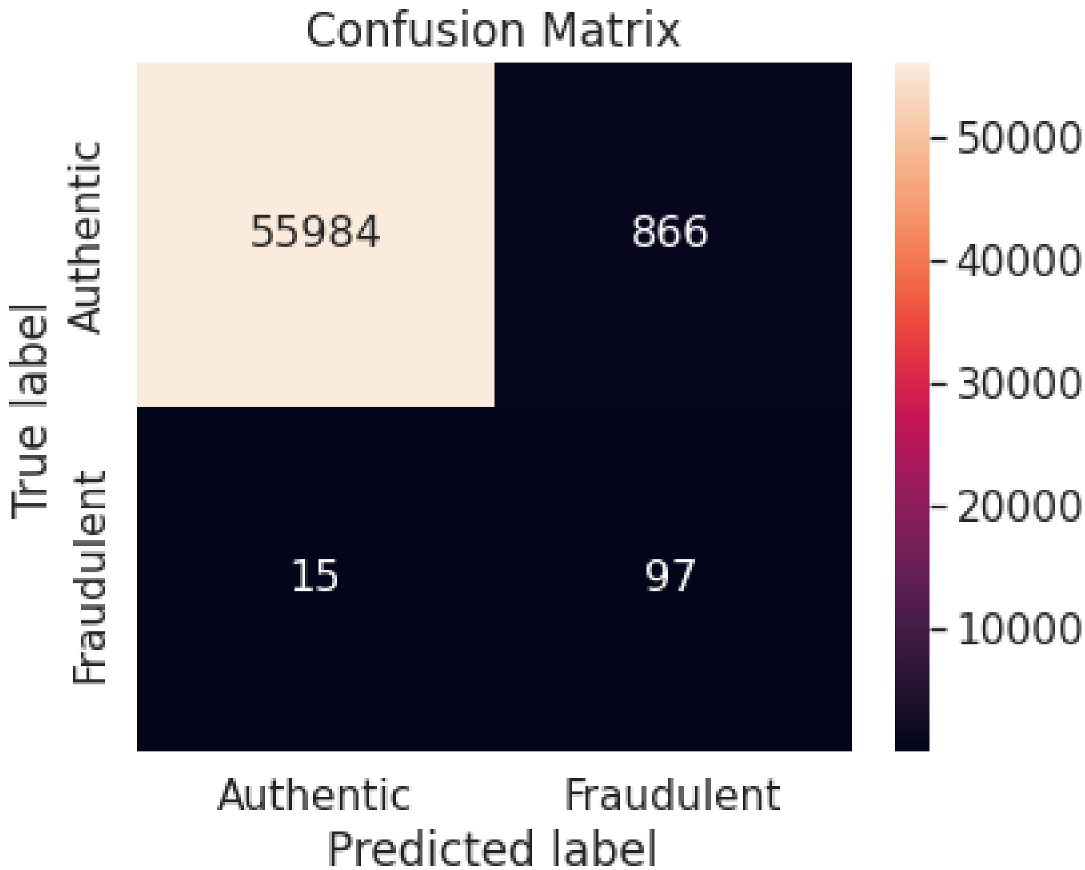


[88]:
```
Metric                   MCC  F1-Score  F2-Score    Recall Precision  FM index  \
Performance score  0.277674  0.164826  0.320555  0.866071   0.09108  0.280859

Metric             Specificity    G-mean F0.5-Score  Accuracy        AP  \
Performance score     0.982973  0.922673   0.110933  0.982743  0.276675
```

```
Metric                ROC-AUC
Performance score   0.960779
```

1.55 Random over-sampling

[89]:
```
# Elements of confusion matrix

classification(lda, X_train_over, y_train_over, X_test, y_test)

# Summary of evaluation metrics

summary_lda_over = summary
summary_lda_over.set_index('Metric')

y_score = lda.decision_function(X_test)
average_precision = average_precision_score(y_test, y_score)
y_pred_proba = lda.predict_proba(X_test)[::,1]
roc_auc = metrics.roc_auc_score(y_test, y_pred_proba)

summary_lda_over_extended = summary.copy()
summary_lda_over_extended.loc[len(summary_lda_over_extended.index)] = ['AP',␣
 ↪average_precision]
summary_lda_over_extended.loc[len(summary_lda_over_extended.index)] =␣
 ↪['ROC-AUC', roc_auc]
summary_lda_over_extended.set_index('Metric')

summary_lda_over_index = summary_lda_over_extended.T
summary_lda_over_index.columns = summary_lda_over_index.iloc[0]
summary_lda_over_index.drop(summary_lda_over_index.index[0], inplace = True)
summary_lda_over_index
```

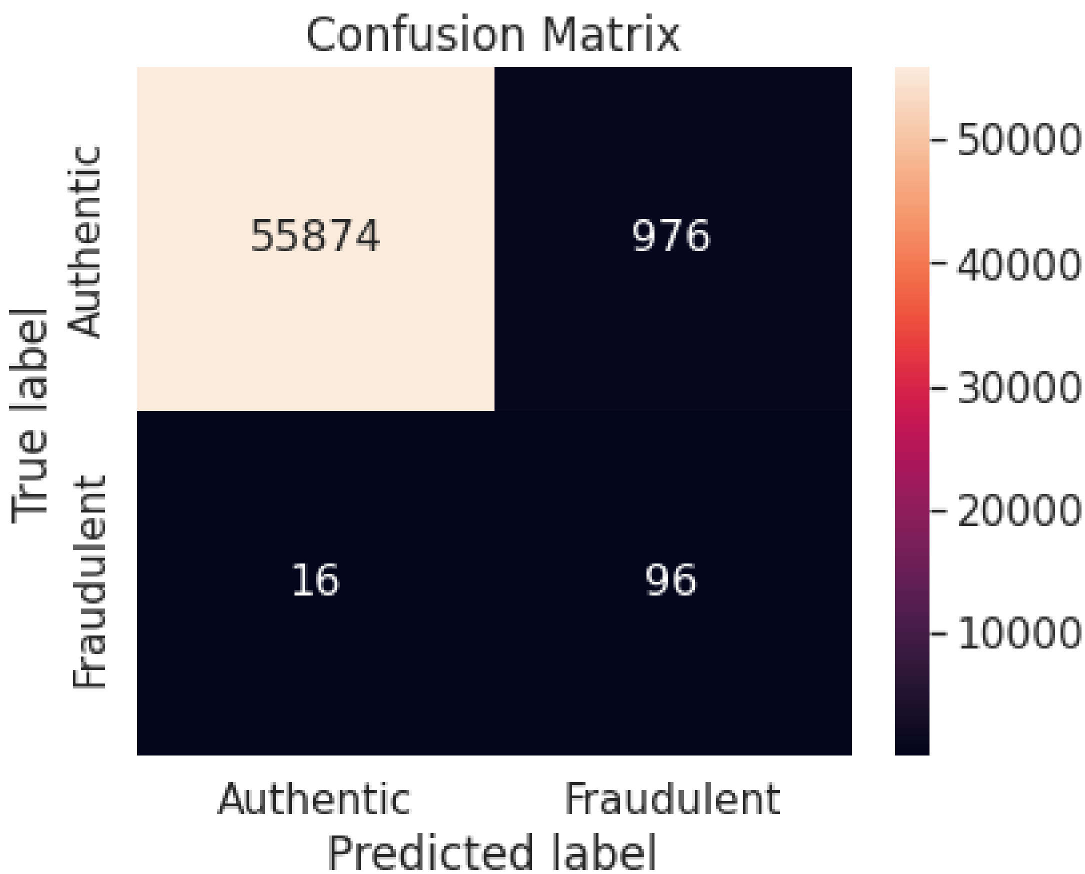



```
[89]: Metric                    MCC  F1-Score  F2-Score    Recall Precision  FM index  \
      Performance score  0.29236  0.180465  0.343728  0.866071  0.100727  0.295359

      Metric             Specificity    G-mean F0.5-Score  Accuracy       AP  \
      Performance score     0.984767  0.923514   0.122351  0.984534  0.28972
```

```
Metric                ROC-AUC
Performance score   0.962346
```

1.56 Random under-sampling with imbalanced-learn library

[90]:
```
# Elements of confusion matrix

classification(lda, X_train_under_imblearn, y_train_under_imblearn, X_test,␣
 ↪y_test)

# Summary of evaluation metrics

summary_lda_under_imblearn = summary
summary_lda_under_imblearn.set_index('Metric')

y_score = lda.decision_function(X_test)
average_precision = average_precision_score(y_test, y_score)
y_pred_proba = lda.predict_proba(X_test)[::,1]
roc_auc = metrics.roc_auc_score(y_test, y_pred_proba)

summary_lda_under_imblearn_extended = summary.copy()
summary_lda_under_imblearn_extended.loc[len(summary_lda_under_imblearn_extended.
 ↪index)] = ['AP', average_precision]
summary_lda_under_imblearn_extended.loc[len(summary_lda_under_imblearn_extended.
 ↪index)] = ['ROC-AUC', roc_auc]
summary_lda_under_imblearn_extended.set_index('Metric')

summary_lda_under_imblearn_index = summary_lda_under_imblearn_extended.T
summary_lda_under_imblearn_index.columns = summary_lda_under_imblearn_index.
 ↪iloc[0]
summary_lda_under_imblearn_index.drop(summary_lda_under_imblearn_index.
 ↪index[0], inplace = True)
summary_lda_under_imblearn_index
```

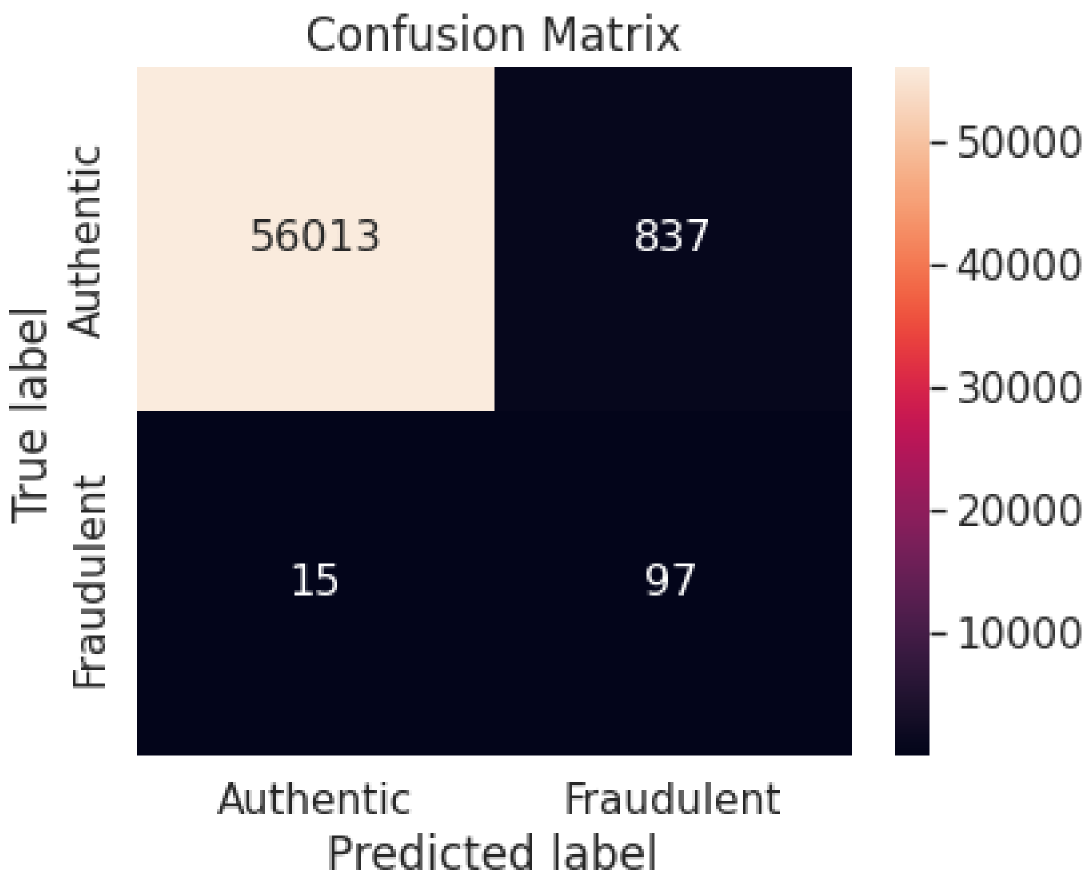



```
[90]: Metric                  MCC  F1-Score  F2-Score    Recall Precision  FM index  \
      Performance score  0.273827  0.162162  0.315789  0.857143  0.089552  0.277054

      Metric             Specificity    G-mean F0.5-Score  Accuracy        AP  \
      Performance score     0.982832  0.917838   0.109091  0.982585  0.195096
```

```
Metric               ROC-AUC
Performance score  0.962084
```

1.57 Random over-sampling with imbalanced-learn library

[91]:
```
# Elements of confusion matrix

classification(lda, X_train_over_imblearn, y_train_over_imblearn, X_test,␣
 ↪y_test)

# Summary of evaluation metrics

summary_lda_over_imblearn = summary
summary_lda_over_imblearn.set_index('Metric')

y_score = lda.decision_function(X_test)
average_precision = average_precision_score(y_test, y_score)
y_pred_proba = lda.predict_proba(X_test)[::,1]
roc_auc = metrics.roc_auc_score(y_test, y_pred_proba)

summary_lda_over_imblearn_extended = summary.copy()
summary_lda_over_imblearn_extended.loc[len(summary_lda_over_imblearn_extended.
 ↪index)] = ['AP', average_precision]
summary_lda_over_imblearn_extended.loc[len(summary_lda_over_imblearn_extended.
 ↪index)] = ['ROC-AUC', roc_auc]
summary_lda_over_imblearn_extended.set_index('Metric')

summary_lda_over_imblearn_index = summary_lda_over_imblearn_extended.T
summary_lda_over_imblearn_index.columns = summary_lda_over_imblearn_index.
 ↪iloc[0]
summary_lda_over_imblearn_index.drop(summary_lda_over_imblearn_index.index[0],␣
 ↪inplace = True)
summary_lda_over_imblearn_index
```

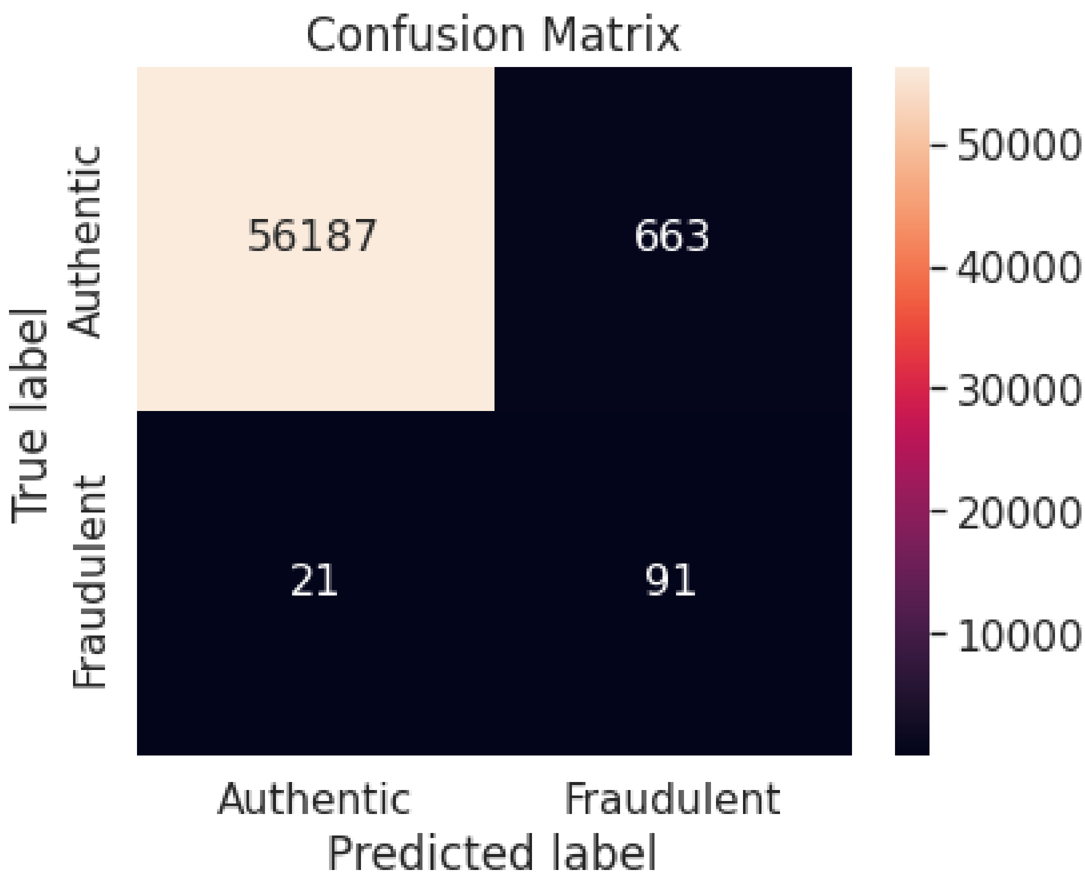



```
[91]: Metric                   MCC   F1-Score   F2-Score     Recall  Precision   FM index  \
      Performance score   0.296965   0.185468   0.350941   0.866071   0.103854   0.299909

      Metric             Specificity     G-mean F0.5-Score   Accuracy         AP  \
      Performance score     0.985277   0.923753    0.12604   0.985043   0.294876
```

```
Metric              ROC-AUC
Performance score  0.962222
```

1.58 Synthetic minority over-sampling technique (SMOTE)

[92]:
```python
# Elements of confusion matrix

classification(lda, X_train_over_smote, y_train_over_smote, X_test, y_test)

# Summary of evaluation metrics

summary_lda_over_smote = summary
summary_lda_over_smote.set_index('Metric')

y_score = lda.decision_function(X_test)
average_precision = average_precision_score(y_test, y_score)
y_pred_proba = lda.predict_proba(X_test)[::,1]
roc_auc = metrics.roc_auc_score(y_test, y_pred_proba)

summary_lda_over_smote_extended = summary.copy()
summary_lda_over_smote_extended.loc[len(summary_lda_over_smote_extended.index)]␣
 ↪= ['AP', average_precision]
summary_lda_over_smote_extended.loc[len(summary_lda_over_smote_extended.index)]␣
 ↪= ['ROC-AUC', roc_auc]
summary_lda_over_smote_extended.set_index('Metric')

summary_lda_over_smote_index = summary_lda_over_smote_extended.T
summary_lda_over_smote_index.columns = summary_lda_over_smote_index.iloc[0]
summary_lda_over_smote_index.drop(summary_lda_over_smote_index.index[0],␣
 ↪inplace = True)
summary_lda_over_smote_index
```



[92]:

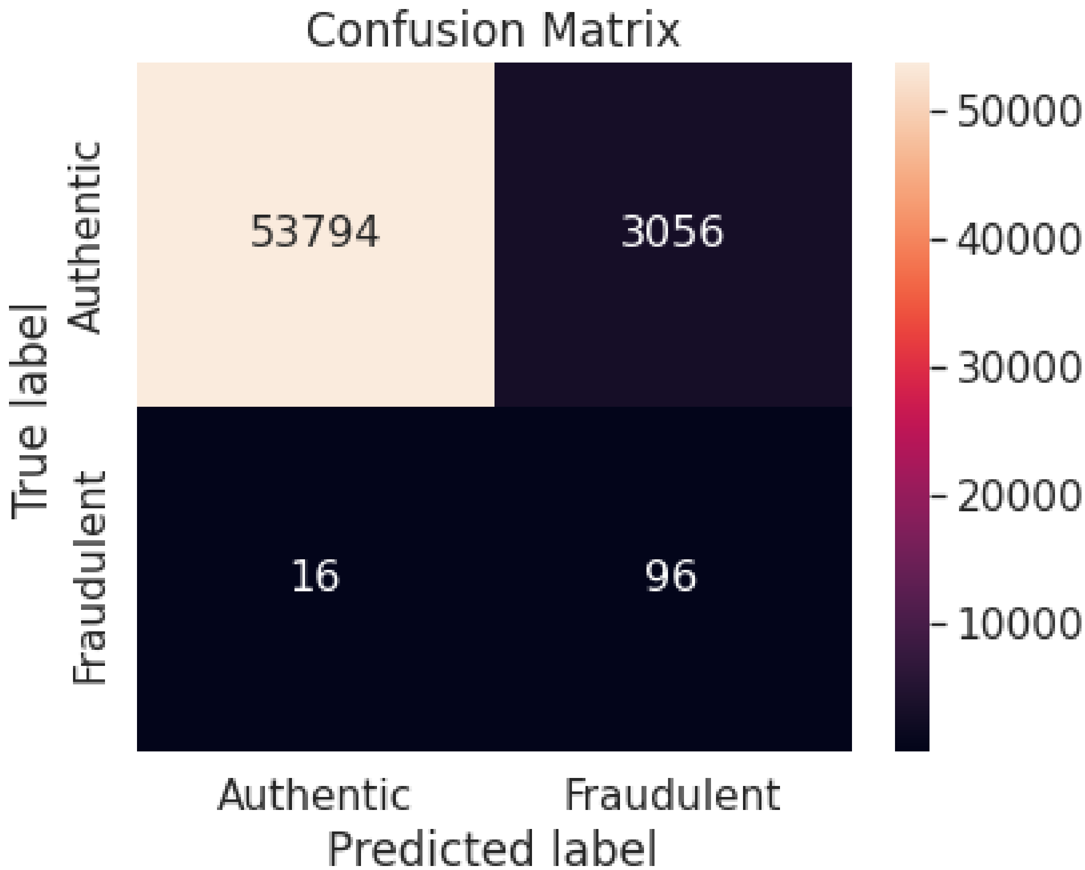
Metric	MCC	F1-Score	F2-Score	Recall	Precision	FM index	\
Performance score	0.310409	0.210162	0.378536	0.8125	0.12069	0.313146	

Metric	Specificity	G-mean	F0.5-Score	Accuracy	AP	\
Performance score	0.988338	0.896116	0.14546	0.987992	0.345685	

```
Metric              ROC-AUC
Performance score   0.96569
```

1.59 Under-sampling via NearMiss

[93]:
```
# Elements of confusion matrix

classification(lda, X_train_under_nm, y_train_under_nm, X_test, y_test)

# Summary of evaluation metrics

summary_lda_under_nm = summary
summary_lda_under_nm.set_index('Metric')

y_score = lda.decision_function(X_test)
average_precision = average_precision_score(y_test, y_score)
y_pred_proba = lda.predict_proba(X_test)[::,1]
roc_auc = metrics.roc_auc_score(y_test, y_pred_proba)

summary_lda_under_nm_extended = summary.copy()
summary_lda_under_nm_extended.loc[len(summary_lda_under_nm_extended.index)] =␣
 ↪['AP', average_precision]
summary_lda_under_nm_extended.loc[len(summary_lda_under_nm_extended.index)] =␣
 ↪['ROC-AUC', roc_auc]
summary_lda_under_nm_extended.set_index('Metric')

summary_lda_under_nm_index = summary_lda_under_nm_extended.T
summary_lda_under_nm_index.columns = summary_lda_under_nm_index.iloc[0]
summary_lda_under_nm_index.drop(summary_lda_under_nm_index.index[0], inplace =␣
 ↪True)
summary_lda_under_nm_index
```



[93]:

Metric	MCC	F1-Score	F2-Score	Recall	Precision	FM index	\
Performance score	0.155659	0.058824	0.133333	0.857143	0.030457	0.161573	

Metric	Specificity	G-mean	F0.5-Score	Accuracy	AP	\
Performance score	0.946245	0.900592	0.037736	0.946069	0.052498	

```
Metric              ROC-AUC
Performance score  0.948673
```

1.60 Summary of LDA models

[94]:
```python
summary_lda = pd.DataFrame(columns = ['Metric'])

summary_lda['Metric'] = EvalMetricLabels
summary_lda_list = [summary_lda_unaltered, summary_lda_under, summary_lda_over,␣
 ↪summary_lda_under_imblearn,
                    summary_lda_over_imblearn, summary_lda_over_smote,␣
 ↪summary_lda_under_nm]

for i in summary_lda_list:
    summary_lda = pd.merge(summary_lda, i, on = 'Metric')

TrainingSetsMetric = TrainingSets.copy()
TrainingSetsMetric.insert(0, 'Metric')

summary_lda.columns = TrainingSetsMetric
summary_lda.set_index('Metric', inplace = True)
summary_lda
```

[94]:
```
             Unaltered       RUS       ROS    RUS-IL    ROS-IL     SMOTE  \
Metric
MCC           0.851736  0.277674  0.292360  0.273827  0.296965  0.310409
F1-Score      0.852018  0.164826  0.180465  0.162162  0.185468  0.210162
F2-Score      0.849732  0.320555  0.343728  0.315789  0.350941  0.378536
Recall        0.848214  0.866071  0.866071  0.857143  0.866071  0.812500
Precision     0.855856  0.091080  0.100727  0.089552  0.103854  0.120690
FM index      0.852027  0.280859  0.295359  0.277054  0.299909  0.313146
Specificity   0.999719  0.982973  0.984767  0.982832  0.985277  0.988338
G-mean        0.920856  0.922673  0.923514  0.917838  0.923753  0.896116
F0.5-Score    0.854317  0.110933  0.122351  0.109091  0.126040  0.145460
Accuracy      0.999421  0.982743  0.984534  0.982585  0.985043  0.987992

                   NM
Metric
MCC          0.155659
F1-Score     0.058824
F2-Score     0.133333
Recall       0.857143
Precision    0.030457
FM index     0.161573
Specificity  0.946245
```

```
G-mean        0.900592
F0.5-Score    0.037736
Accuracy      0.946069
```

[95]:
```
# Visual comparison of the model applied on different training sets through␣
↪various evaluation metrics

summary_visual(summary_lda)
```

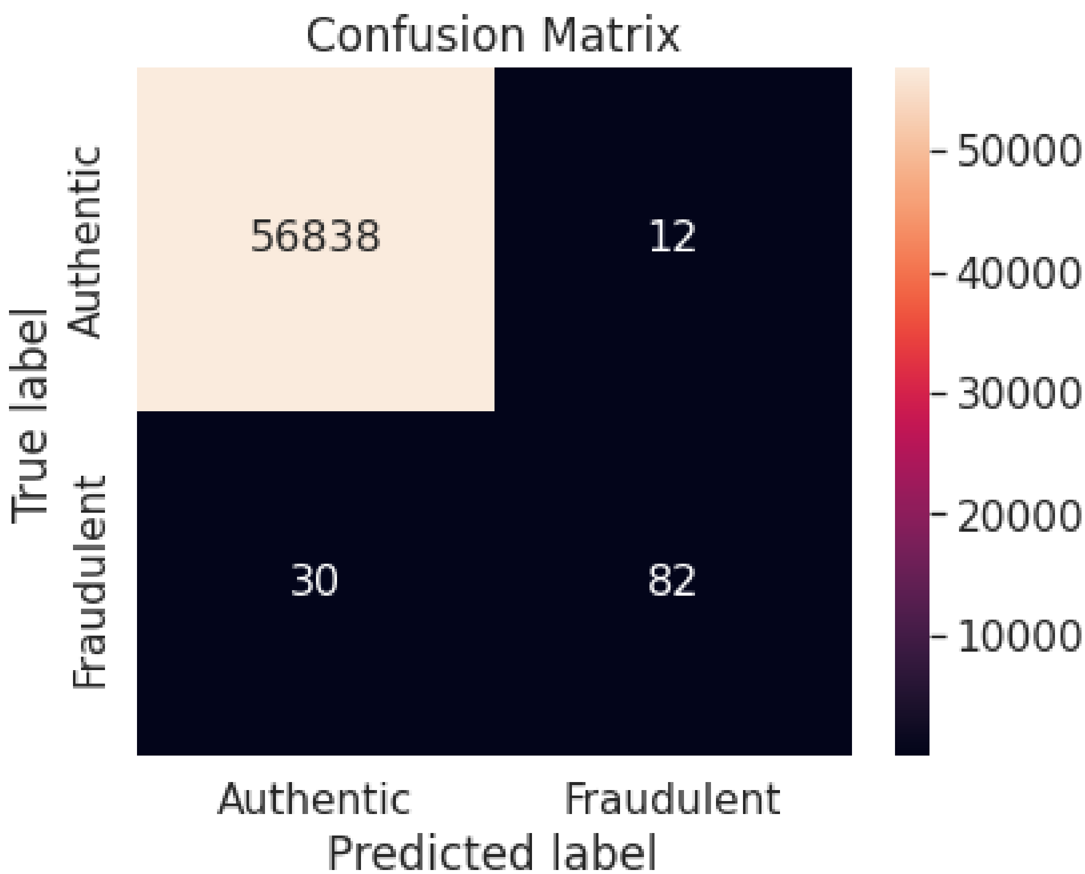
12. Stochastic Gradient Descent (SGD)

[96]:
```
sgd = SGDClassifier(loss = 'hinge')
```

1.61 Unaltered training set

[97]:
```
# Elements of confusion matrix

classification(sgd, X_train_scaled_minmax, y_train, X_test_scaled_minmax,␣
↪y_test)

# Summary of evaluation metrics

summary_sgd_unaltered = summary.copy()
summary_sgd_unaltered.set_index('Metric')

y_score = sgd.decision_function(X_test)
average_precision = average_precision_score(y_test, y_score)

summary_sgd_unaltered_extended = summary.copy()
```

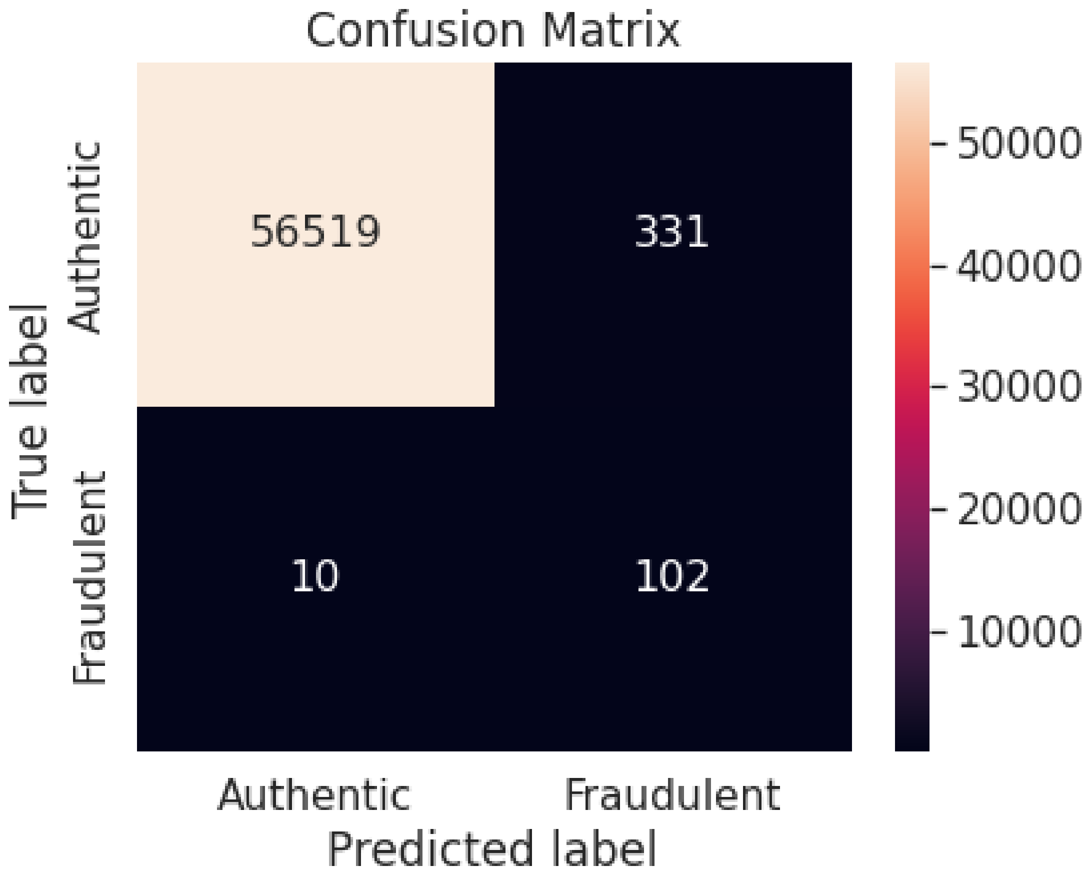
```
summary_sgd_unaltered_extended.loc[len(summary_sgd_unaltered_extended.index)] =␣
 ↪['AP', average_precision]
summary_sgd_unaltered_extended.set_index('Metric')

summary_sgd_unaltered_index = summary_sgd_unaltered_extended.T
summary_sgd_unaltered_index.columns = summary_sgd_unaltered_index.iloc[0]
summary_sgd_unaltered_index.drop(summary_sgd_unaltered_index.index[0], inplace␣
 ↪= True)
summary_sgd_unaltered_index
```



[97]:
```
Metric                    MCC  F1-Score  F2-Score    Recall Precision  FM index  \
Performance score  0.798816  0.796117  0.756458  0.732143   0.87234  0.799173

Metric             Specificity    G-mean F0.5-Score  Accuracy        AP
Performance score     0.999789  0.855563   0.840164  0.999263  0.043155
```

1.62 Random under-sampling

[98]:
```
# Elements of confusion matrix

classification(sgd, X_train_under_scaled_minmax, y_train_under,␣
↪X_test_scaled_minmax, y_test)

# Summary of evaluation metrics

summary_sgd_under = summary
summary_sgd_under.set_index('Metric')

y_score = sgd.decision_function(X_test)
average_precision = average_precision_score(y_test, y_score)

summary_sgd_under_extended = summary.copy()
summary_sgd_under_extended.loc[len(summary_sgd_under_extended.index)] = ['AP',␣
↪average_precision]
summary_sgd_under_extended.set_index('Metric')

summary_sgd_under_index = summary_sgd_under_extended.T
summary_sgd_under_index.columns = summary_sgd_under_index.iloc[0]
summary_sgd_under_index.drop(summary_sgd_under_index.index[0], inplace = True)
summary_sgd_under_index
```



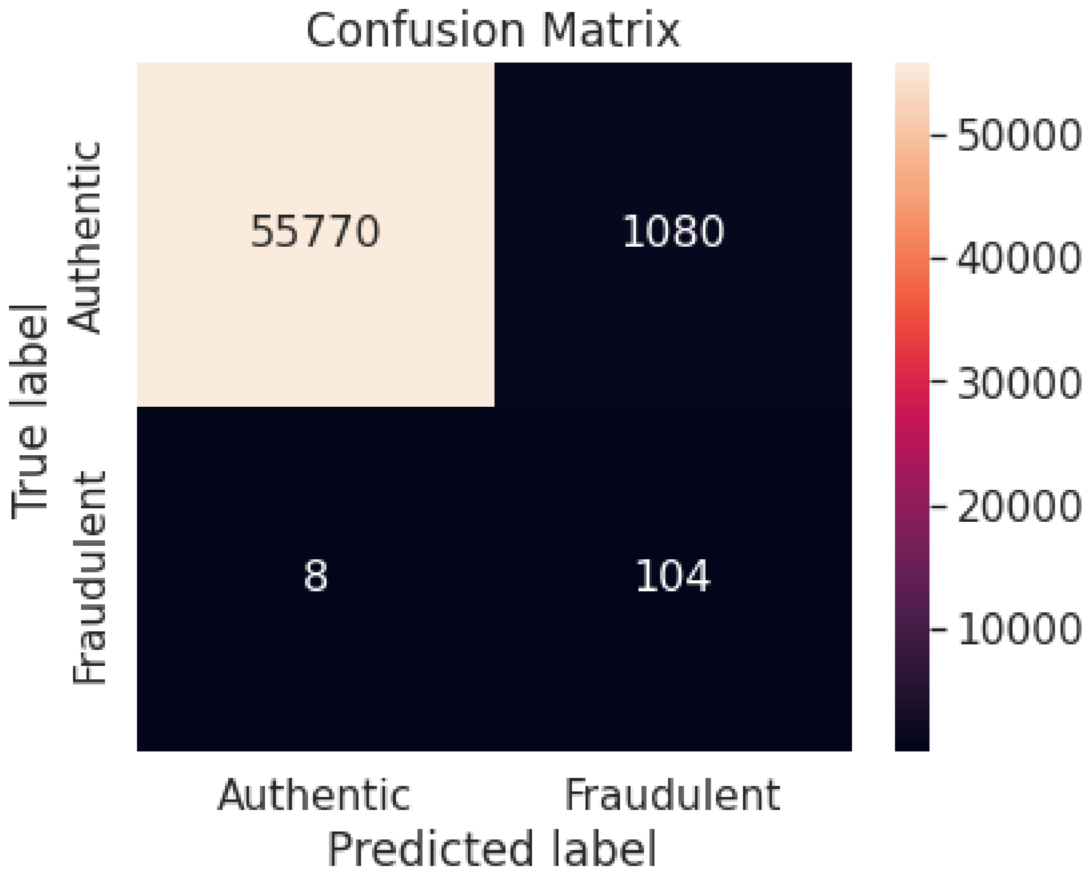
[98]:

Metric	MCC	F1-Score	F2-Score	Recall	Precision	FM index	\
Performance score	0.461521	0.374312	0.578888	0.910714	0.235566	0.463177	

Metric	Specificity	G-mean	F0.5-Score	Accuracy	AP
Performance score	0.994178	0.951531	0.276573	0.994014	0.00282

1.63 Random over-sampling

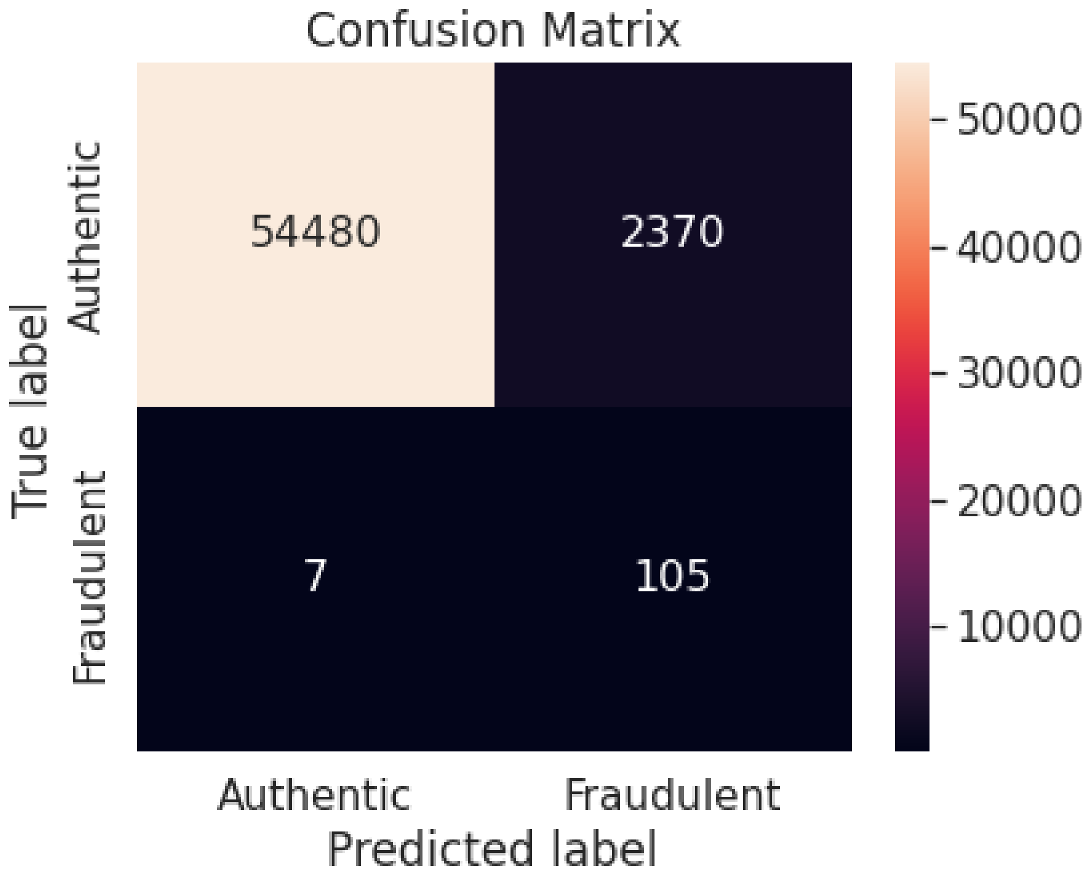
[99]:
```
# Elements of confusion matrix

classification(sgd, X_train_over_scaled_minmax, y_train_over,␣
 ↪X_test_scaled_minmax, y_test)

# Summary of evaluation metrics

summary_sgd_over = summary
summary_sgd_over.set_index('Metric')

y_score = sgd.decision_function(X_test)
average_precision = average_precision_score(y_test, y_score)

summary_sgd_over_extended = summary.copy()
summary_sgd_over_extended.loc[len(summary_sgd_over_extended.index)] = ['AP',␣
 ↪average_precision]
summary_sgd_over_extended.set_index('Metric')

summary_sgd_over_index = summary_sgd_over_extended.T
summary_sgd_over_index.columns = summary_sgd_over_index.iloc[0]
summary_sgd_over_index.drop(summary_sgd_over_index.index[0], inplace = True)
summary_sgd_over_index
```



[99]:
```
Metric                  MCC  F1-Score  F2-Score    Recall Precision  FM index  \
Performance score  0.282426  0.160494  0.318627  0.928571  0.087838  0.285594

Metric             Specificity    G-mean F0.5-Score Accuracy        AP
Performance score     0.981003  0.954427   0.107261   0.9809  0.002837
```

1.64 Random under-sampling with imbalanced-learn library

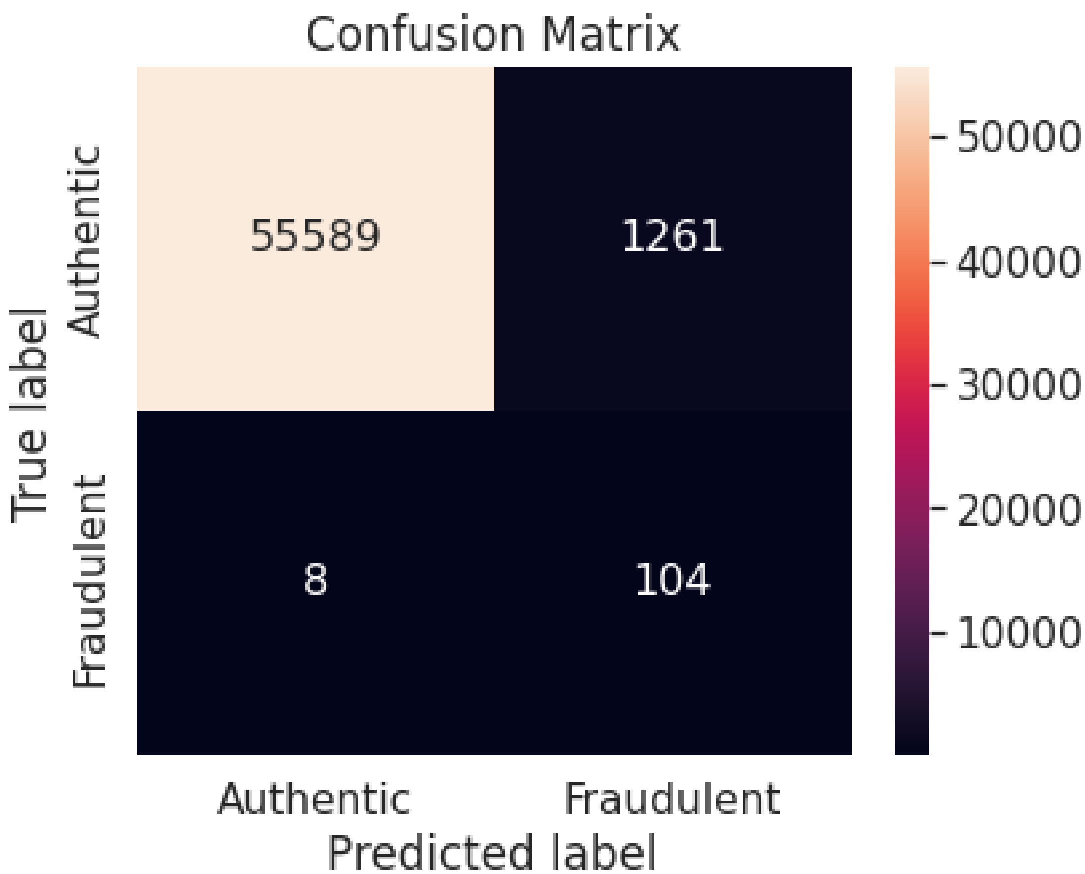
[100]:
```
# Elements of confusion matrix

classification(sgd, X_train_under_imblearn_scaled_minmax,␣
 ↪y_train_under_imblearn, X_test_scaled_minmax, y_test)

# Summary of evaluation metrics

summary_sgd_under_imblearn = summary
summary_sgd_under_imblearn.set_index('Metric')

y_score = sgd.decision_function(X_test)
average_precision = average_precision_score(y_test, y_score)

summary_sgd_under_imblearn_extended = summary.copy()
summary_sgd_under_imblearn_extended.loc[len(summary_sgd_under_imblearn_extended.
 ↪index)] = ['AP', average_precision]
summary_sgd_under_imblearn_extended.set_index('Metric')

summary_sgd_under_imblearn_index = summary_sgd_under_imblearn_extended.T
summary_sgd_under_imblearn_index.columns = summary_sgd_under_imblearn_index.
 ↪iloc[0]
summary_sgd_under_imblearn_index.drop(summary_sgd_under_imblearn_index.
 ↪index[0], inplace = True)
summary_sgd_under_imblearn_index
```




```
[100]: Metric                     MCC   F1-Score F2-Score   Recall Precision   FM index  \
       Performance score  0.194651  0.081175  0.17961  0.9375  0.042424  0.199431

       Metric             Specificity    G-mean F0.5-Score Accuracy        AP
       Performance score     0.958311  0.947849   0.052437  0.95827  0.00281
```

1.65 Random over-sampling with imbalanced-learn library

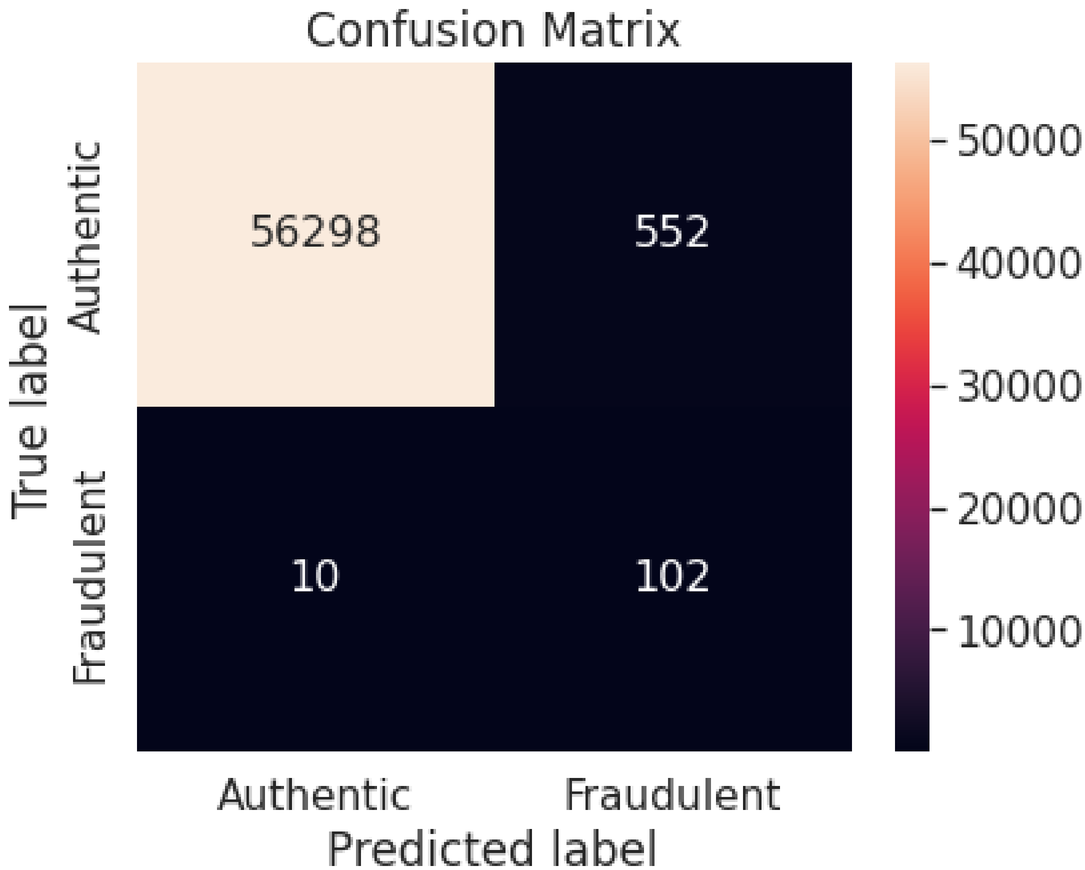
[101]:
```
# Elements of confusion matrix

classification(sgd, X_train_over_imblearn_scaled_minmax, y_train_over_imblearn,␣
 ↪X_test_scaled_minmax, y_test)

# Summary of evaluation metrics

summary_sgd_over_imblearn = summary
summary_sgd_over_imblearn.set_index('Metric')

y_score = sgd.decision_function(X_test)
average_precision = average_precision_score(y_test, y_score)

summary_sgd_over_imblearn_extended = summary.copy()
summary_sgd_over_imblearn_extended.loc[len(summary_sgd_over_imblearn_extended.
 ↪index)] = ['AP', average_precision]
summary_sgd_over_imblearn_extended.set_index('Metric')

summary_sgd_over_imblearn_index = summary_sgd_over_imblearn_extended.T
summary_sgd_over_imblearn_index.columns = summary_sgd_over_imblearn_index.
 ↪iloc[0]
summary_sgd_over_imblearn_index.drop(summary_sgd_over_imblearn_index.index[0],␣
 ↪inplace = True)
summary_sgd_over_imblearn_index
```



[101]:
```
Metric                        MCC  F1-Score  F2-Score    Recall Precision  FM index  \
Performance score  0.262541  0.140826  0.286817  0.928571   0.07619  0.265986

Metric             Specificity    G-mean F0.5-Score  Accuracy        AP
Performance score     0.977819  0.952877   0.093324  0.977722  0.00283
```

1.66 Synthetic minority over-sampling technique (SMOTE)

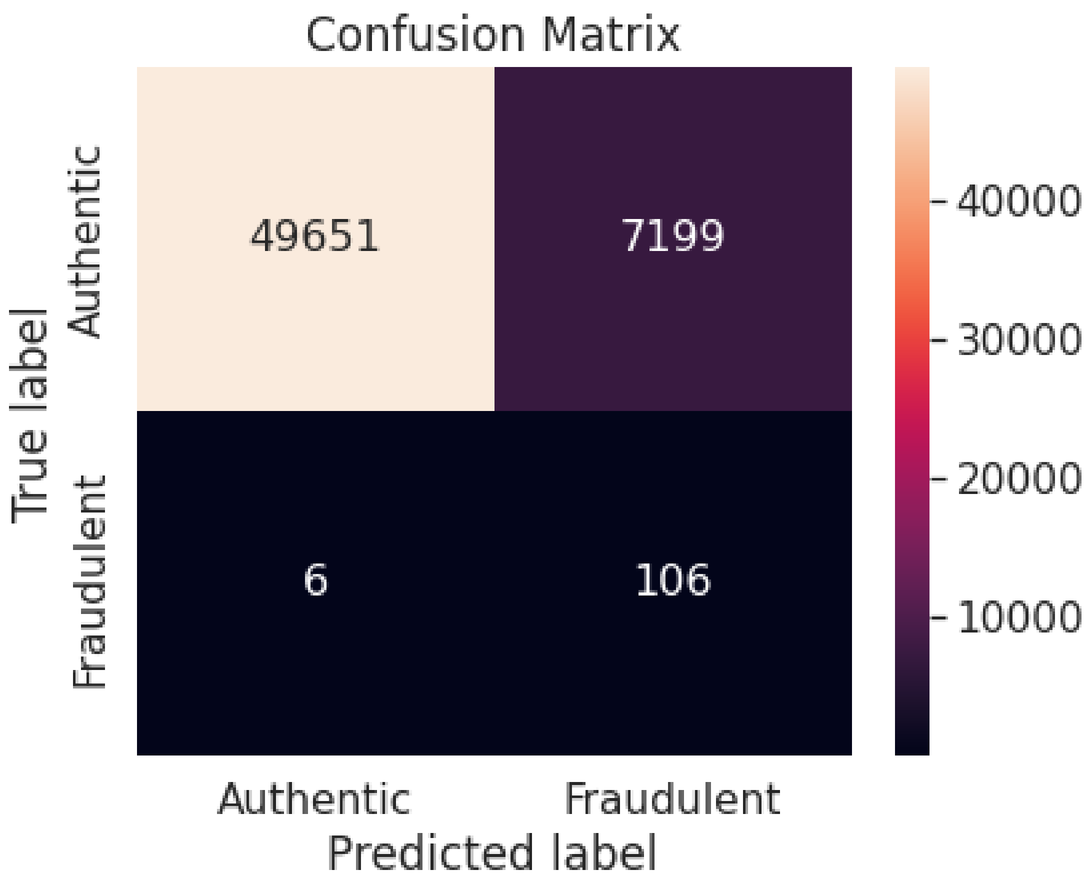
[102]:
```
# Elements of confusion matrix

classification(sgd, X_train_over_smote_scaled_minmax, y_train_over_smote,␣
↪X_test_scaled_minmax, y_test)

# Summary of evaluation metrics

summary_sgd_over_smote = summary
summary_sgd_over_smote.set_index('Metric')

y_score = sgd.decision_function(X_test)
average_precision = average_precision_score(y_test, y_score)

summary_sgd_over_smote_extended = summary.copy()
summary_sgd_over_smote_extended.loc[len(summary_sgd_over_smote_extended.index)]␣
↪= ['AP', average_precision]
summary_sgd_over_smote_extended.set_index('Metric')

summary_sgd_over_smote_index = summary_sgd_over_smote_extended.T
summary_sgd_over_smote_index.columns = summary_sgd_over_smote_index.iloc[0]
summary_sgd_over_smote_index.drop(summary_sgd_over_smote_index.index[0],␣
↪inplace = True)
summary_sgd_over_smote_index
```



[102]:
```
Metric                    MCC  F1-Score  F2-Score    Recall Precision  FM index  \
Performance score  0.374651  0.266319  0.462795  0.910714  0.155963  0.376879

Metric             Specificity    G-mean F0.5-Score  Accuracy        AP
Performance score      0.99029  0.949669    0.18695  0.990134  0.002994
```

1.67 Under-sampling via NearMiss

[103]:
```
# Elements of confusion matrix

classification(sgd, X_train_under_nm_scaled_minmax, y_train_under_nm,␣
↪X_test_scaled_minmax, y_test)

# Summary of evaluation metrics

summary_sgd_under_nm = summary
summary_sgd_under_nm.set_index('Metric')

y_score = sgd.decision_function(X_test)
average_precision = average_precision_score(y_test, y_score)

summary_sgd_under_nm_extended = summary.copy()
summary_sgd_under_nm_extended.loc[len(summary_sgd_under_nm_extended.index)] =␣
↪['AP', average_precision]
summary_sgd_under_nm_extended.set_index('Metric')

summary_sgd_under_nm_index = summary_sgd_under_nm_extended.T
summary_sgd_under_nm_index.columns = summary_sgd_under_nm_index.iloc[0]
summary_sgd_under_nm_index.drop(summary_sgd_under_nm_index.index[0], inplace =␣
↪True)
summary_sgd_under_nm_index
```



[103]:
```
Metric                   MCC  F1-Score  F2-Score    Recall Precision  FM index  \
Performance score  0.108613  0.028583  0.068361  0.946429  0.014511  0.117189

Metric             Specificity    G-mean F0.5-Score  Accuracy        AP
Performance score     0.873369  0.909165   0.018069  0.873512  0.002816
```

1.68 Summary of SGD models

[104]:
```python
summary_sgd = pd.DataFrame(columns = ['Metric'])

summary_sgd['Metric'] = EvalMetricLabels
summary_sgd_list = [summary_sgd_unaltered, summary_sgd_under, summary_sgd_over,␣
↪summary_sgd_under_imblearn,
                    summary_sgd_over_imblearn, summary_sgd_over_smote,␣
↪summary_sgd_under_nm]

for i in summary_sgd_list:
    summary_sgd = pd.merge(summary_sgd, i, on = 'Metric')

TrainingSetsMetric = TrainingSets.copy()
TrainingSetsMetric.insert(0, 'Metric')

summary_sgd.columns = TrainingSetsMetric
summary_sgd.set_index('Metric', inplace = True)
summary_sgd
```

[104]:

Metric	Unaltered	RUS	ROS	RUS-IL	ROS-IL	SMOTE
MCC	0.798816	0.461521	0.282426	0.194651	0.262541	0.374651
F1-Score	0.796117	0.374312	0.160494	0.081175	0.140826	0.266319
F2-Score	0.756458	0.578888	0.318627	0.179610	0.286817	0.462795
Recall	0.732143	0.910714	0.928571	0.937500	0.928571	0.910714
Precision	0.872340	0.235566	0.087838	0.042424	0.076190	0.155963
FM index	0.799173	0.463177	0.285594	0.199431	0.265986	0.376879
Specificity	0.999789	0.994178	0.981003	0.958311	0.977819	0.990290
G-mean	0.855563	0.951531	0.954427	0.947849	0.952877	0.949669
F0.5-Score	0.840164	0.276573	0.107261	0.052437	0.093324	0.186950
Accuracy	0.999263	0.994014	0.980900	0.958270	0.977722	0.990134

Metric	NM
MCC	0.108613
F1-Score	0.028583
F2-Score	0.068361
Recall	0.946429
Precision	0.014511
FM index	0.117189
Specificity	0.873369
G-mean	0.909165
F0.5-Score	0.018069
Accuracy	0.873512

[105]:
```
# Visual comparison of the model applied on different training sets through␣
↪various evaluation metrics

summary_visual(summary_sgd)
```

13. Ridge Classifier

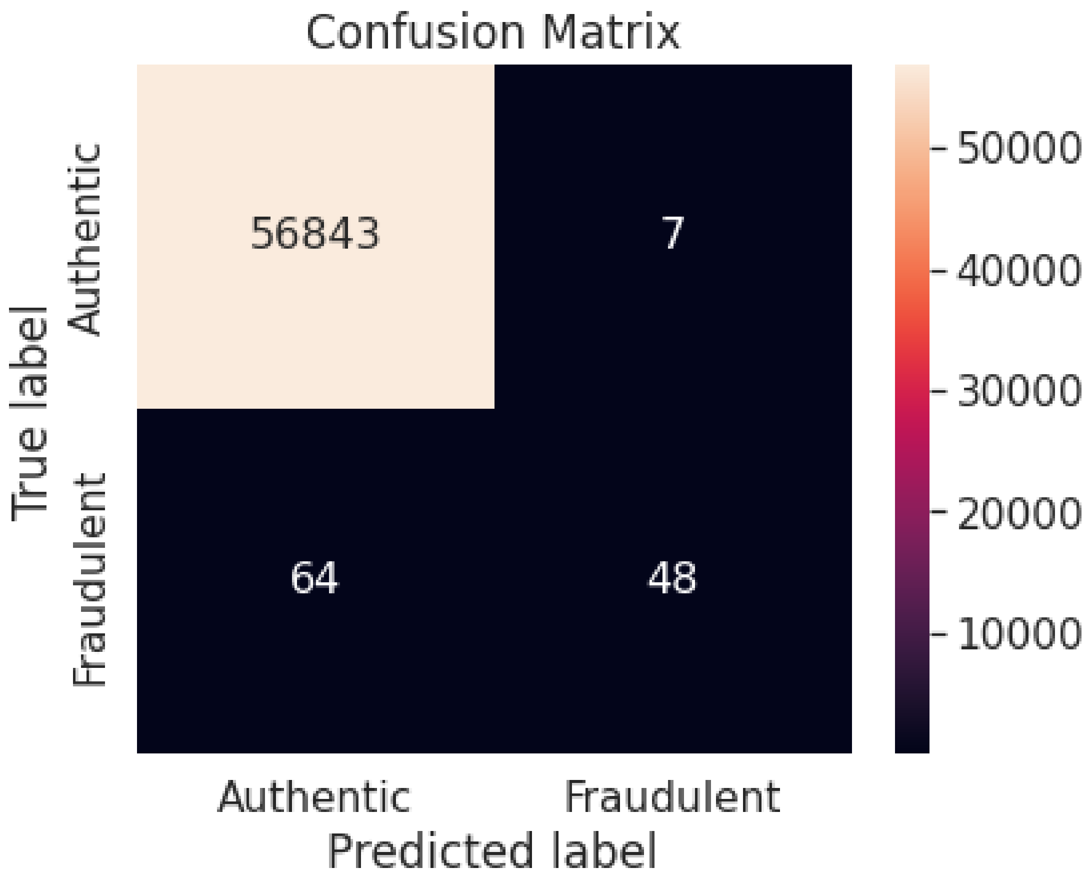
[106]:
```
ridge = RidgeClassifier()
```

We use normalised features as the ridge classifier employs l^2 regularization through an additive penalty term in the objective function.

1.69 Unaltered training set

[107]:
```
# Elements of confusion matrix

classification(ridge, X_train_scaled_minmax, y_train, X_test_scaled_minmax,␣
↪y_test)

# Summary of evaluation metrics

summary_ridge_unaltered = summary.copy()
summary_ridge_unaltered.set_index('Metric')

y_score = ridge.decision_function(X_test)
average_precision = average_precision_score(y_test, y_score)

summary_ridge_unaltered_extended = summary.copy()
summary_ridge_unaltered_extended.loc[len(summary_ridge_unaltered_extended.
↪index)] = ['AP', average_precision]
```

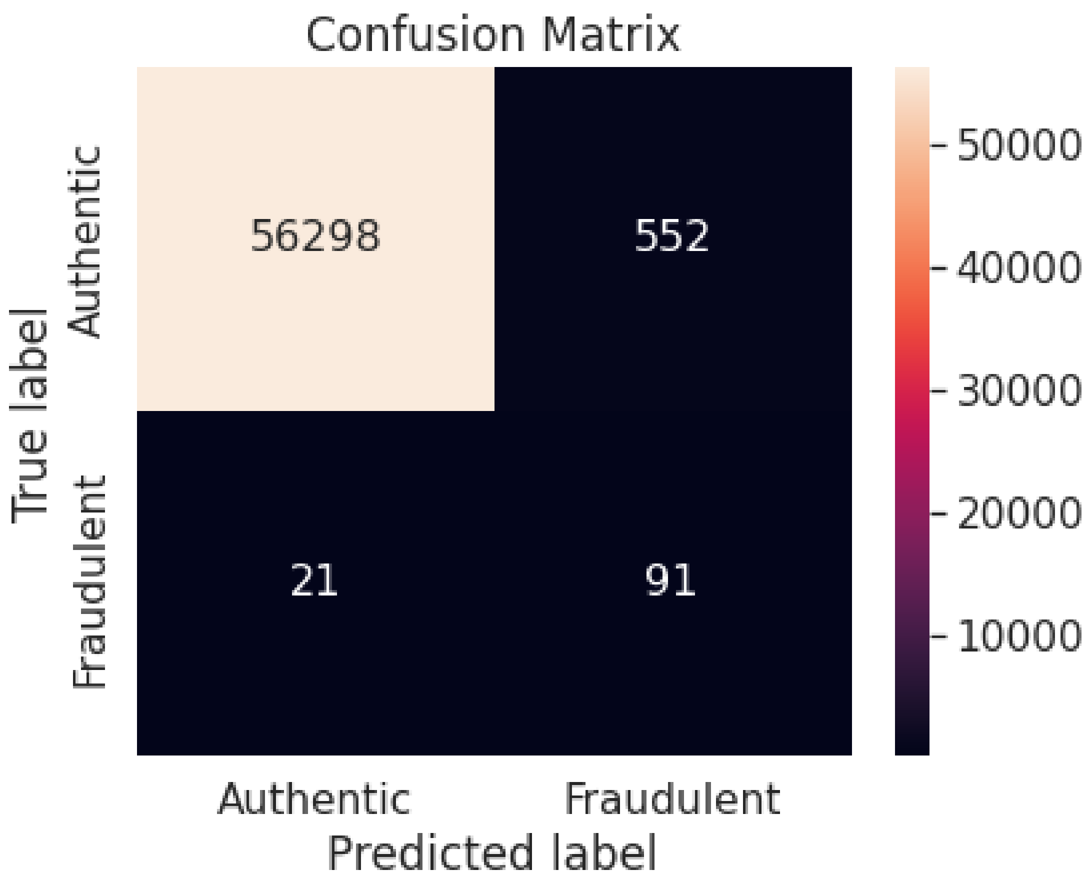
```
summary_ridge_unaltered_extended.set_index('Metric')

summary_ridge_unaltered_index = summary_ridge_unaltered_extended.T
summary_ridge_unaltered_index.columns = summary_ridge_unaltered_index.iloc[0]
summary_ridge_unaltered_index.drop(summary_ridge_unaltered_index.index[0],␣
 ↪inplace = True)
summary_ridge_unaltered_index
```



[107]:
```
Metric                   MCC F1-Score  F2-Score    Recall Precision  FM index  \
Performance score  0.611095  0.57485  0.477137  0.428571  0.872727  0.611577

Metric            Specificity    G-mean F0.5-Score  Accuracy        AP
Performance score    0.999877  0.654613   0.722892  0.998754  0.010148
```

1.70 Random under-sampling

[108]:
```
# Elements of confusion matrix

classification(ridge, X_train_under_scaled_minmax, y_train_under,␣
↪X_test_scaled_minmax, y_test)

# Summary of evaluation metrics

summary_ridge_under = summary.copy()
summary_ridge_under.set_index('Metric')

y_score = ridge.decision_function(X_test)
average_precision = average_precision_score(y_test, y_score)

summary_ridge_under_extended = summary.copy()
summary_ridge_under_extended.loc[len(summary_ridge_under_extended.index)] =␣
↪['AP', average_precision]
summary_ridge_under_extended.set_index('Metric')

summary_ridge_under_index = summary_ridge_under_extended.T
summary_ridge_under_index.columns = summary_ridge_under_index.iloc[0]
summary_ridge_under_index.drop(summary_ridge_under_index.index[0], inplace =␣
↪True)
summary_ridge_under_index
```

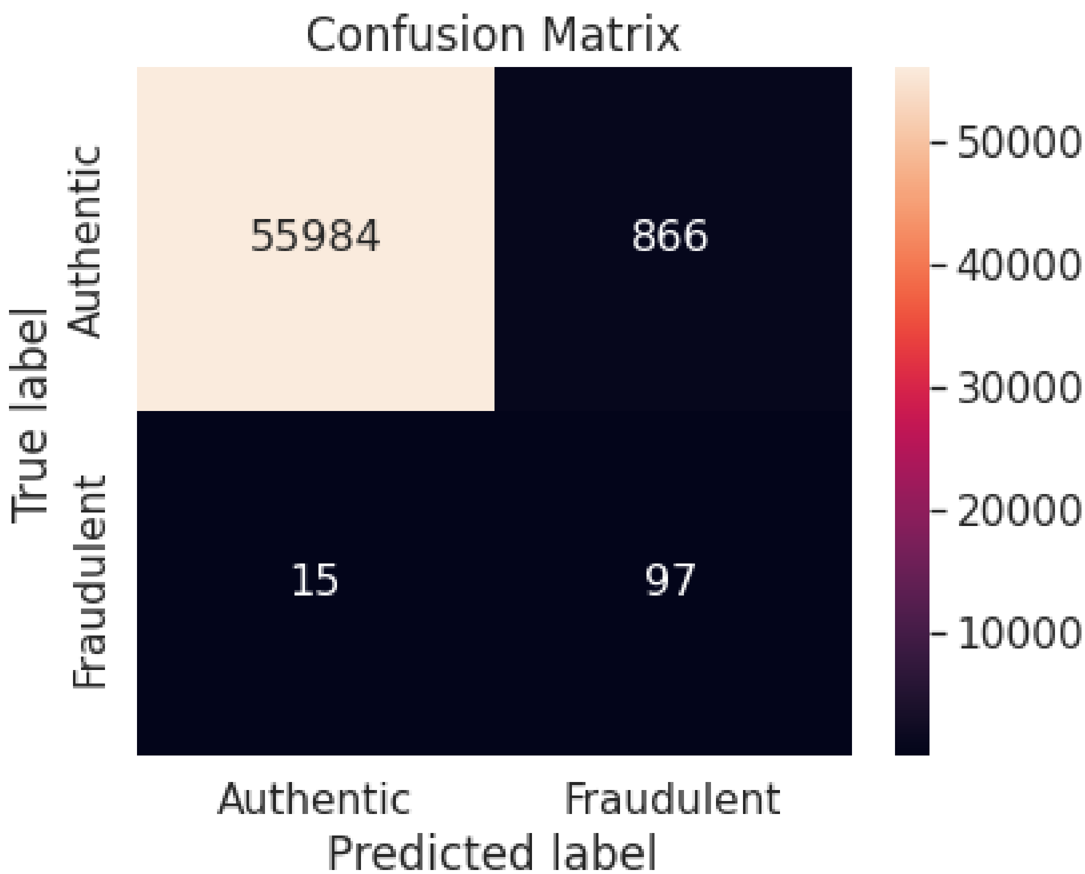


[108]:

```
Metric                  MCC F1-Score  F2-Score  Recall Precision  FM index  \
Performance score  0.336623  0.24106  0.417049  0.8125  0.141524  0.339099

Metric            Specificity    G-mean F0.5-Score  Accuracy        AP
Performance score     0.99029  0.897001   0.169523  0.989941  0.002821
```

1.71 Random over-sampling

[109]:
```
# Elements of confusion matrix

classification(ridge, X_train_over_scaled_minmax, y_train_over,␣
 ↪X_test_scaled_minmax, y_test)

# Summary of evaluation metrics

summary_ridge_over = summary.copy()
summary_ridge_over.set_index('Metric')

y_score = ridge.decision_function(X_test)
average_precision = average_precision_score(y_test, y_score)

summary_ridge_over_extended = summary.copy()
summary_ridge_over_extended.loc[len(summary_ridge_over_extended.index)] =␣
 ↪['AP', average_precision]
summary_ridge_over_extended.set_index('Metric')

summary_ridge_over_index = summary_ridge_over_extended.T
summary_ridge_over_index.columns = summary_ridge_over_index.iloc[0]
summary_ridge_over_index.drop(summary_ridge_over_index.index[0], inplace = True)
summary_ridge_over_index
```



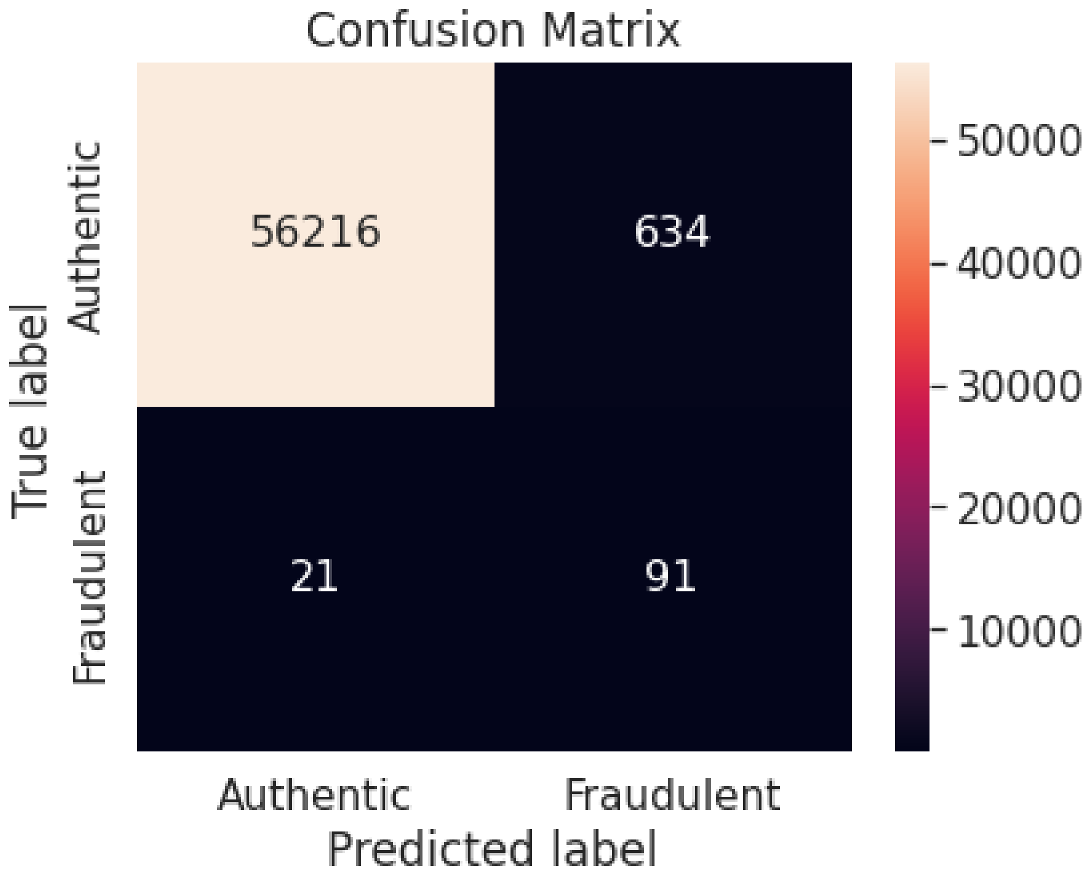
[109]:

Metric	MCC	F1-Score	F2-Score	Recall	Precision	FM index
Performance score	0.29236	0.180465	0.343728	0.866071	0.100727	0.295359

Metric	Specificity	G-mean	F0.5-Score	Accuracy	AP
Performance score	0.984767	0.923514	0.122351	0.984534	0.002775

1.72 Random under-sampling with imbalanced-learn library

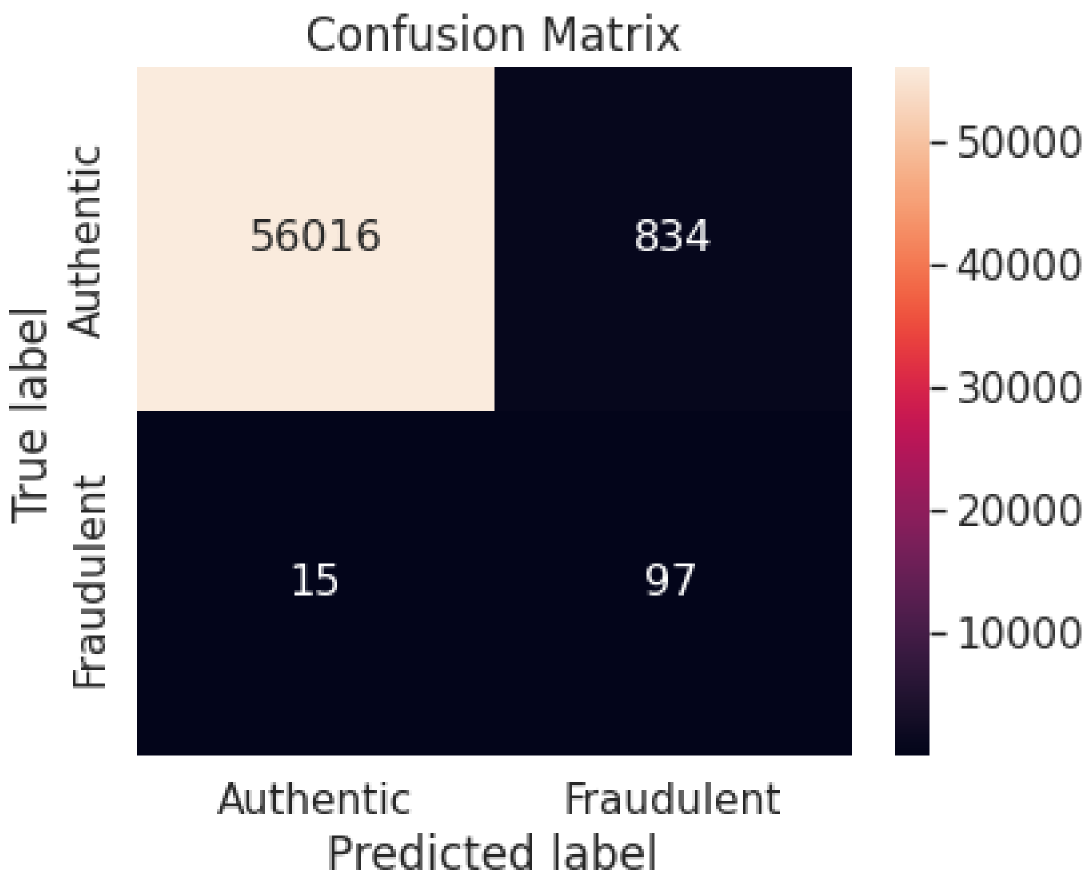
[110]:
```
# Elements of confusion matrix

classification(ridge, X_train_under_imblearn_scaled_minmax,␣
 ↪y_train_under_imblearn, X_test_scaled_minmax, y_test)

# Summary of evaluation metrics

summary_ridge_under_imblearn = summary.copy()
summary_ridge_under_imblearn.set_index('Metric')

y_score = ridge.decision_function(X_test)
average_precision = average_precision_score(y_test, y_score)

summary_ridge_under_imblearn_extended = summary.copy()
summary_ridge_under_imblearn_extended.
 ↪loc[len(summary_ridge_under_imblearn_extended.index)] = ['AP',␣
 ↪average_precision]
summary_ridge_under_imblearn_extended.set_index('Metric')

summary_ridge_under_imblearn_index = summary_ridge_under_imblearn_extended.T
summary_ridge_under_imblearn_index.columns = summary_ridge_under_imblearn_index.
 ↪iloc[0]
summary_ridge_under_imblearn_index.drop(summary_ridge_under_imblearn_index.
 ↪index[0], inplace = True)
summary_ridge_under_imblearn_index
```



[110]:
```
Metric                  MCC  F1-Score  F2-Score  Recall Precision  FM index  \
Performance score  0.316676  0.217443  0.387894  0.8125  0.125517  0.319347

Metric             Specificity    G-mean F0.5-Score  Accuracy        AP
Performance score     0.988848  0.896348   0.151062  0.988501  0.002833
```

1.73 Random over-sampling with imbalanced-learn library

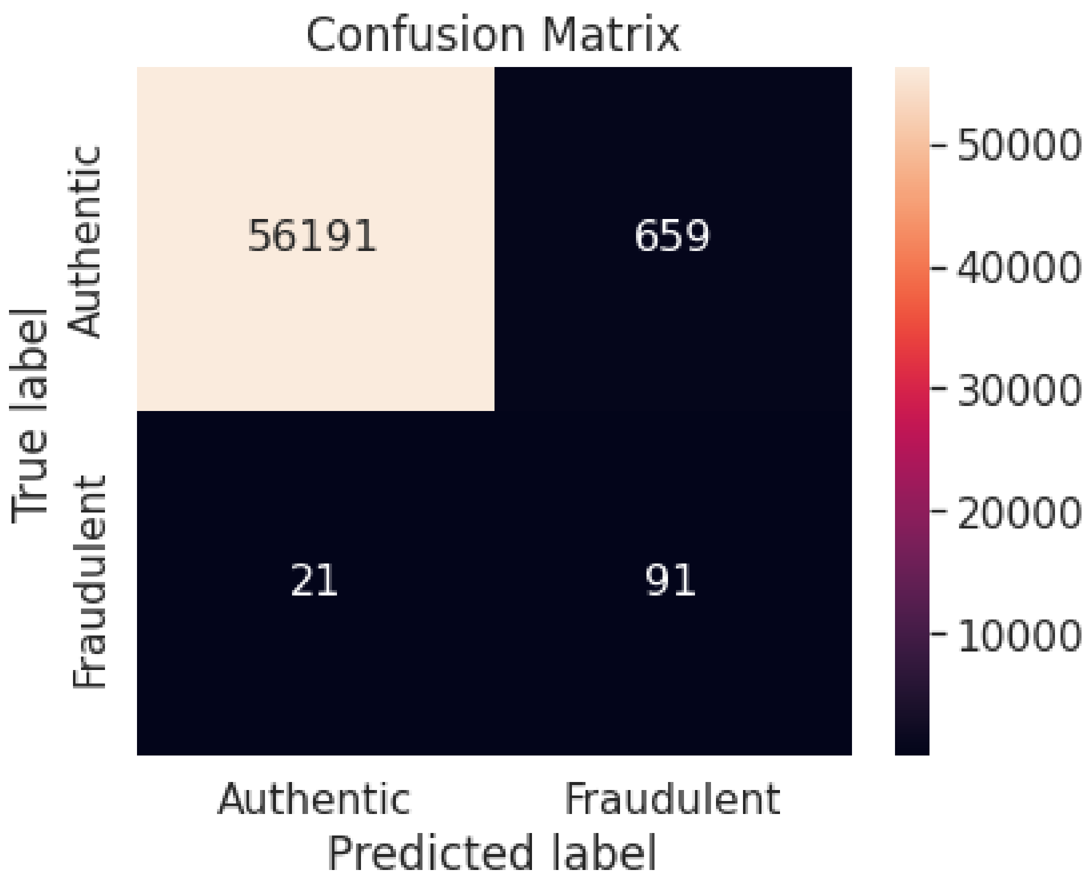
[111]:
```
# Elements of confusion matrix

classification(ridge, X_train_over_imblearn_scaled_minmax,␣
↪y_train_over_imblearn, X_test_scaled_minmax, y_test)

# Summary of evaluation metrics

summary_ridge_over_imblearn = summary.copy()
summary_ridge_over_imblearn.set_index('Metric')

y_score = ridge.decision_function(X_test)
average_precision = average_precision_score(y_test, y_score)

summary_ridge_over_imblearn_extended = summary.copy()
summary_ridge_over_imblearn_extended.
↪loc[len(summary_ridge_over_imblearn_extended.index)] = ['AP',␣
↪average_precision]
summary_ridge_over_imblearn_extended.set_index('Metric')

summary_ridge_over_imblearn_index = summary_ridge_over_imblearn_extended.T
summary_ridge_over_imblearn_index.columns = summary_ridge_over_imblearn_index.
↪iloc[0]
summary_ridge_over_imblearn_index.drop(summary_ridge_over_imblearn_index.
↪index[0], inplace = True)
summary_ridge_over_imblearn_index
```




```
[111]: Metric                    MCC   F1-Score  F2-Score    Recall Precision  FM index  \
       Performance score  0.297454  0.186002  0.351704  0.866071  0.104189  0.300392

       Metric             Specificity    G-mean F0.5-Score  Accuracy        AP
       Performance score      0.98533  0.923778   0.126434  0.985095  0.002772
```

1.74 Synthetic minority over-sampling technique (SMOTE)

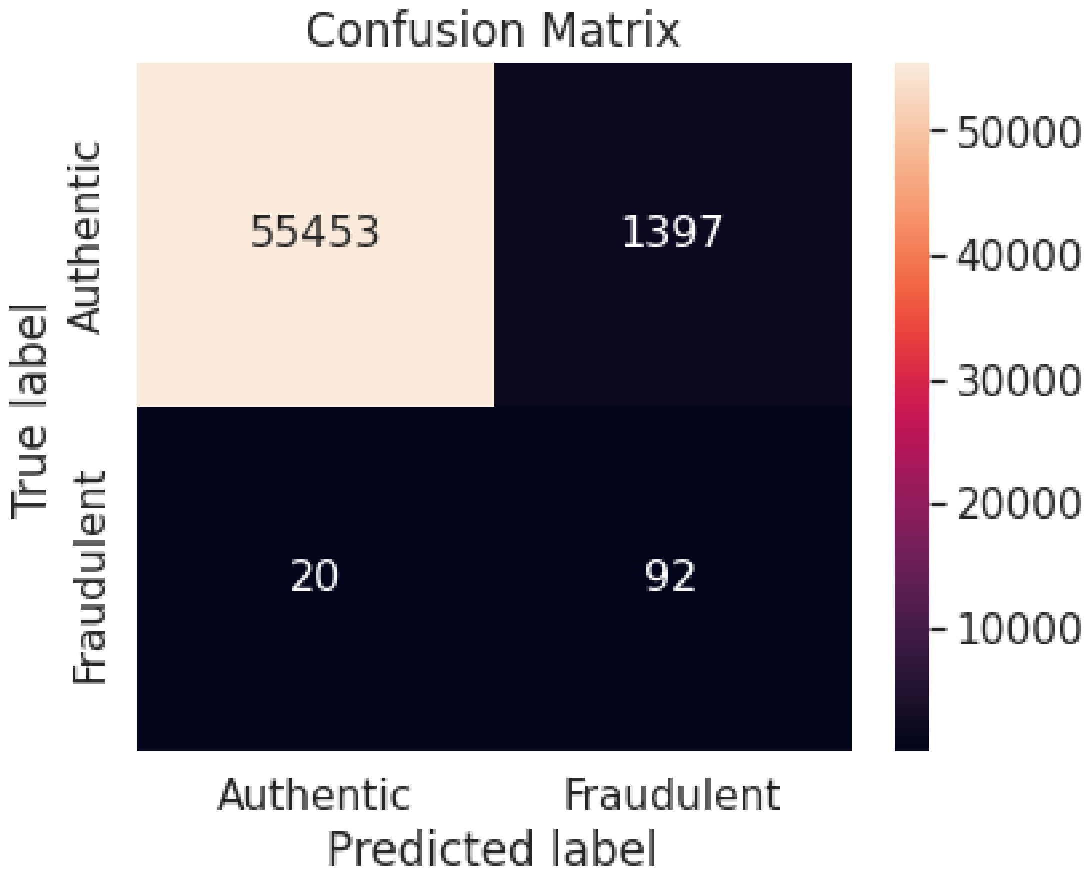
[112]:
```
# Elements of confusion matrix

classification(ridge, X_train_over_smote_scaled_minmax, y_train_over_smote,␣
↪X_test_scaled_minmax, y_test)

# Summary of evaluation metrics

summary_ridge_over_smote = summary.copy()
summary_ridge_over_smote.set_index('Metric')

y_score = ridge.decision_function(X_test)
average_precision = average_precision_score(y_test, y_score)

summary_ridge_over_smote_extended = summary.copy()
summary_ridge_over_smote_extended.loc[len(summary_ridge_over_smote_extended.
↪index)] = ['AP', average_precision]
summary_ridge_over_smote_extended.set_index('Metric')

summary_ridge_over_smote_index = summary_ridge_over_smote_extended.T
summary_ridge_over_smote_index.columns = summary_ridge_over_smote_index.iloc[0]
summary_ridge_over_smote_index.drop(summary_ridge_over_smote_index.index[0],␣
↪inplace = True)
summary_ridge_over_smote_index
```



[112]:
```
Metric                  MCC  F1-Score F2-Score  Recall Precision FM index  \
Performance score  0.311252  0.211137   0.3798  0.8125  0.121333  0.31398

Metric            Specificity    G-mean F0.5-Score  Accuracy        AP
Performance score    0.988408  0.896148   0.146208  0.988062  0.002765
```

1.75 Under-sampling via NearMiss

[113]:
```
# Elements of confusion matrix

classification(ridge, X_train_under_nm_scaled_minmax, y_train_under_nm,␣
 ↪X_test_scaled_minmax, y_test)

# Summary of evaluation metrics

summary_ridge_under_nm = summary.copy()
summary_ridge_under_nm.set_index('Metric')

y_score = ridge.decision_function(X_test)
average_precision = average_precision_score(y_test, y_score)

summary_ridge_under_nm_extended = summary.copy()
summary_ridge_under_nm_extended.loc[len(summary_ridge_under_nm_extended.index)]␣
 ↪= ['AP', average_precision]
summary_ridge_under_nm_extended.set_index('Metric')

summary_ridge_under_nm_index = summary_ridge_under_nm_extended.T
summary_ridge_under_nm_index.columns = summary_ridge_under_nm_index.iloc[0]
summary_ridge_under_nm_index.drop(summary_ridge_under_nm_index.index[0],␣
 ↪inplace = True)
summary_ridge_under_nm_index
```



[113]:
```
Metric                    MCC  F1-Score  F2-Score    Recall Precision  FM index  \
Performance score  0.221241  0.114928  0.237481  0.821429  0.061786  0.225285

Metric             Specificity    G-mean F0.5-Score  Accuracy        AP
Performance score     0.975427  0.895122   0.075808  0.975124  0.002798
```

1.76 Summary of ridge classifiers

[114]:
```
summary_ridge = pd.DataFrame(columns = ['Metric'])

summary_ridge['Metric'] = EvalMetricLabels
summary_ridge_list = [summary_ridge_unaltered, summary_ridge_under,␣
 ↪summary_ridge_over, summary_ridge_under_imblearn,
                      summary_ridge_over_imblearn, summary_ridge_over_smote,␣
 ↪summary_ridge_under_nm]

for i in summary_ridge_list:
    summary_ridge = pd.merge(summary_ridge, i, on = 'Metric')

TrainingSetsMetric = TrainingSets.copy()
TrainingSetsMetric.insert(0, 'Metric')

summary_ridge.columns = TrainingSetsMetric
summary_ridge.set_index('Metric', inplace = True)
summary_ridge
```

[114]:
```
             Unaltered       RUS       ROS    RUS-IL    ROS-IL     SMOTE  \
Metric
MCC           0.611095  0.336623  0.292360  0.316676  0.297454  0.311252
F1-Score      0.574850  0.241060  0.180465  0.217443  0.186002  0.211137
F2-Score      0.477137  0.417049  0.343728  0.387894  0.351704  0.379800
Recall        0.428571  0.812500  0.866071  0.812500  0.866071  0.812500
Precision     0.872727  0.141524  0.100727  0.125517  0.104189  0.121333
FM index      0.611577  0.339099  0.295359  0.319347  0.300392  0.313980
Specificity   0.999877  0.990290  0.984767  0.988848  0.985330  0.988408
G-mean        0.654613  0.897001  0.923514  0.896348  0.923778  0.896148
F0.5-Score    0.722892  0.169523  0.122351  0.151062  0.126434  0.146208
Accuracy      0.998754  0.989941  0.984534  0.988501  0.985095  0.988062

                   NM
Metric
MCC          0.221241
F1-Score     0.114928
F2-Score     0.237481
Recall       0.821429
Precision    0.061786
FM index     0.225285
Specificity  0.975427
G-mean       0.895122
F0.5-Score   0.075808
Accuracy     0.975124
```

[115]:
```
# Visual comparison of the model applied on different training sets through␣
↪various evaluation metrics

summary_visual(summary_ridge)
```

14. Conclusion

We choose the training set for each model on which it performs best and tabulate their performance in terms of **F2-Score**, which considers the facts that the dataset is imbalanced, the positive class (fraudulent transactions) is more important than the negative class (authentic transactions) and also that false negatives are more costly than false positives. Additionally, we report **MCC** (captures all-round performance across classes) and **Recall** (focuses only on the crucial postive class).

[116]:
```
# Comparison of classification models

"""
In the final table, models are sorted in decreasing order of their performance␣
↪on the testing set, measured in F2-Score

The training set which is fed to a classifier is mentioned in parenthesis␣
↪following the name of that classifier

Unaltered: unaltered training set
ROS-IL: random over-sampling of minority class via imbalanced-learn library
SMOTE: Over-sampling of minority class via synthetic minority over-sampling␣
↪technique (SMOTE)
"""
```

```
models = ['Logistic Regression (Unaltered)', 'KNN (Unaltered)', 'Decision Tree␣
↪(ROS-IL)',
          'Linear SVM (Unaltered)', 'Naive Bayes (SMOTE)', 'Random Forest␣
↪(SMOTE)',
          'LDA (Unaltered)', 'SGD (Unaltered)', 'Ridge Classifier (Unaltered)']
metrics = ['F2-Score', 'MCC', 'Recall']
cols = ['Classification model'] + metrics

model_comparison = pd.DataFrame(columns = cols)
model_comparison['Classification model'] = models

summary_list = [summary_logreg_unaltered_index, summary_knn_unaltered_index,␣
↪summary_dt_over_imblearn_index,
                summary_svm_linear_unaltered_index,␣
↪summary_nb_over_smote_index, summary_rf_over_smote_index,
                summary_lda_unaltered_index, summary_sgd_unaltered_index,␣
↪summary_ridge_unaltered_index]

F2_score = []
MCC = []
Recall = []

for i in summary_list:
  F2_score.append(float(i['F2-Score']))
  MCC.append(float(i['MCC']))
  Recall.append(float(i['Recall']))

model_comparison['F2-Score'] = F2_score
model_comparison['MCC'] = MCC
model_comparison['Recall'] = Recall

model_comparison.set_index('Classification model', inplace = True)
model_comparison_descending_F2 = model_comparison.sort_values(by =␣
↪['F2-Score'], ascending = False)
model_comparison_descending_F2
```

[116]:

```
                                 F2-Score       MCC    Recall
Classification model
Random Forest (SMOTE)            0.880783  0.875894  0.883929
Linear SVM (Unaltered)           0.857143  0.856861  0.857143
LDA (Unaltered)                  0.849732  0.851736  0.848214
Decision Tree (ROS-IL)           0.818345  0.827076  0.812500
KNN (Unaltered)                  0.804067  0.852191  0.776786
SGD (Unaltered)                  0.756458  0.798816  0.732143
Logistic Regression (Unaltered)  0.684411  0.769972  0.642857
Ridge Classifier (Unaltered)     0.477137  0.611095  0.428571
```

```
Naive Bayes (SMOTE)              0.464760  0.370966  0.812500
```

The **Random Forest** algorithm applied on the training set obtained after oversampling the minority class (fraudulent transactions) via **SMOTE** appears to be the best classification model for the problem at hand.

SMOTE is one of the best choices to oversample the minority class when the data is imbalanced. It is not surprising that **Random Forest** turns out to be one of the most suitable classifiers for the problem due to the following reasons:

- The algorithm works well in dealing with large datasets with high dimensions.
- It is less affected by the presence of outliers in feature variables compared to other algorithms.
- It does not make any distributional assumption on the feature variables.
- It handles collinearity (linear dependence among features) implicitly.
- It automatically ignores the features which are not useful, effectively doing feature selection on its own.