1. Introduction

In [ ]:
```
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 confusion_matrix, ConfusionMatrixDisplay
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
```

In [ ]:
```
pip freeze > requirements.txt
```

```
Note: you may need to restart the kernel to use updated packages.
WARNING: Ignoring invalid distribution -ensorflow (e:\miniconda3\envs\bootcamp\lib\site-packages)
WARNING: Ignoring invalid distribution -rotobuf (e:\miniconda3\envs\bootcamp\lib\site-packages)
```

In [ ]:
```
data = pd.read_csv('creditcard.csv')
data
```

Out[ ]:

	Time	V1	V2	V3	V4	V5	V6	V7	V8
0	0.0	-1.359807	-0.072781	2.536347	1.378155	-0.338321	0.462388	0.239599	0.098698
1	0.0	1.191857	0.266151	0.166480	0.448154	0.060018	-0.082361	-0.078803	0.085102
2	1.0	-1.358354	-1.340163	1.773209	0.379780	-0.503198	1.800499	0.791461	0.247676
3	1.0	-0.966272	-0.185226	1.792993	-0.863291	-0.010309	1.247203	0.237609	0.377436
4	2.0	-1.158233	0.877737	1.548718	0.403034	-0.407193	0.095921	0.592941	-0.270533
...	...	...	...	...	...	...	...	...	...
284802	172786.0	-11.881118	10.071785	-9.834783	-2.066656	-5.364473	-2.606837	-4.918215	7.305334
284803	172787.0	-0.732789	-0.055080	2.035030	-0.738589	0.868229	1.058415	0.024330	0.294869
284804	172788.0	1.919565	-0.301254	-3.249640	-0.557828	2.630515	3.031260	-0.296827	0.708417
284805	172788.0	-0.240440	0.530483	0.702510	0.689799	-0.377961	0.623708	-0.686180	0.679145
284806	172792.0	-0.533413	-0.189733	0.703337	-0.506271	-0.012546	-0.649617	1.577006	-0.414650

284807 rows × 31 columns

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

In [ ]:
```
# Statistical descriptions of the features

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

Out[ ]:

	Time	V1	V2	V3	V4	V5	
count	284807.000000	2.848070e+05	2.848070e+05	2.848070e+05	2.848070e+05	2.848070e+05	2.848(
mean	94813.859575	1.168375e-15	3.416908e-16	-1.379537e-15	2.074095e-15	9.604066e-16	1.487
std	47488.145955	1.958696e+00	1.651309e+00	1.516255e+00	1.415869e+00	1.380247e+00	1.332
min	0.000000	-5.640751e+01	-7.271573e+01	-4.832559e+01	-5.683171e+00	-1.137433e+02	-2.616(
25%	54201.500000	-9.203734e-01	-5.985499e-01	-8.903648e-01	-8.486401e-01	-6.915971e-01	-7.682
50%	84692.000000	1.810880e-02	6.548556e-02	1.798463e-01	-1.984653e-02	-5.433583e-02	-2.741
75%	139320.500000	1.315642e+00	8.037239e-01	1.027196e+00	7.433413e-01	6.119264e-01	3.985
max	172792.000000	2.454930e+00	2.205773e+01	9.382558e+00	1.687534e+01	3.480167e+01	7.330

8 rows × 30 columns

Objetivos do projeto

Objetivo principal:

Classificação de transações como autênticas ou fraudulentas. Para sermos precisos, dados os dados sobre **Time**, **Amount** e recursos transformados **V1** a **V28** para uma determinada transação, nossa meta é classificar corretamente a transação como **autêntica** ou **fraudulenta**. Empregamos diferentes técnicas para criar modelos de classificação e compará-los por meio de várias métricas de avaliação.

Objetivos secundários:

Responder às seguintes perguntas usando ferramentas e técnicas de aprendizado de máquina e estatísticas.

- Quando uma transação fraudulenta é feita, ela é seguida logo por uma ou mais transações fraudulentas? Em outras palavras, os invasores fazem transações fraudulentas consecutivas em um curto espaço de tempo?
- O valor de uma transação fraudulenta é geralmente maior do que o de uma transação autêntica?
- Há alguma indicação nos dados de que as transações fraudulentas ocorrem em um período de alta transação?
- Os dados mostram que o número de transações é alto em alguns intervalos de tempo e baixo em outros. A ocorrência de fraudes está relacionada a esses intervalos de tempo?
- Há alguns pontos de tempo que exibem um número alto de transações fraudulentas. Isso se deve ao alto número de transações totais ou a algum outro motivo?

Nesta parte, classificaremos as transações como autênticas ou fraudulentas com base nas informações disponíveis sobre os recursos independentes (tempo, valor e as variáveis transformadas V1-V28). Um problema com o conjunto de dados é que ele é altamente desequilibrado em termos da variável-alvo Classe. Assim, corremos o risco de treinar os modelos com uma amostra representativa de transações fraudulentas de tamanho extremamente pequeno. Empregamos diferentes abordagens para lidar com esse problema. O desempenho de cada modelo é verificado por meio de várias métricas de avaliação e está resumido em uma tabela.

2. Evaluation Metrics

Qualquer previsão sobre uma variável de destino categórica binária se enquadra em uma das quatro categorias:

- Verdadeiro positivo: O modelo de classificação prevê corretamente que o resultado é positivo
- True Negative (Verdadeiro negativo): O modelo de classificação prevê corretamente que o resultado será negativo.
- Falso positivo: O modelo de classificação prevê incorretamente que o output é positivo
- Falso negativo: O modelo de classificação prevê incorretamente que o resultado será negativo

Estado real / Estado previsto →	**Positivo**	**Negativo**	
Positivo	Verdadeiro positivo	Falso negativo	Negativo
Negativo	Falso positivo	Verdadeiro negativo	

Deixe que **TP**, **TN**, **FP** e **FN** denotem, respectivamente, o número de **verdadeiros positivos**, **verdadeiros negativos**, **falso positivos** e **falso negativos** entre as previsões feitas por um determinado modelo de classificação. A seguir, apresentamos as definições de algumas métricas de avaliação com base nessas quatro quantidades.

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

- **Métrica de recuperação de precisão**

$$\text{Precisão} = \frac{\text{Número de previsões positivas verdadeiras}}{\text{Número de previsões positivas totais}} = \frac{TP}{TP + FP}$$

$$\text{Recall} = \frac{\text{Número de previsões positivas verdadeiras}}{\text{Número total de casos positivos}} = \frac{TP}{TP + FN}$$

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

$$F_1\text{-Score} = \text{Média harmônica de precisão e recuperação} = \frac{2 \times \text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}$$

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

em que β é um fator positivo, escolhido de forma que o Recall seja β vezes mais importante que a Precisão na análise. As escolhas populares de β são $0, 5$, 1 e 2.

- Métricas de sensibilidade-especificidade**

$$\text{Sensibilidade} = \frac{\text{Número de previsões positivas verdadeiras}}{\text{Número total de casos positivos}} = \frac{TP}{TP + FN}$$

$$\text{Especificidade} = \frac{\text{Número de previsões negativas verdadeiras}}{\text{Número total de casos negativos}} = \frac{TN}{TN + FP}$$

$$\text{G-mean} = \text{Média geométrica de sensibilidade e especificidade} = \sqrt{\text{Sensibilidade} \times \text{Specificity}}$$

- **Métricas de área sob a curva (AUC)**

Considere as seguintes quantidades:

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

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

A curva ROC (Receiver Operating Characteristic, Característica de Operação do Receptor) é obtida plotando-se a TPR em relação à FPR para vários valores de probabilidade de limite. A área sob a curva ROC (ROC-AUC) serve como uma métrica de avaliação válida.

Da mesma forma, a curva Precision-Recall (PR) é obtida plotando-se a Precision em relação à Recall para um número de valores de probabilidade de limite. A área sob a curva PR (PR-AUC) também é uma métrica de avaliação válida. Outra métrica amplamente usada nesse sentido é a precisão média (Average Precision, AP), que é uma média ponderada de precisões em cada limite, com os pesos sendo o aumento na recuperação do limite anterior.

- **Outras métricas**

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

Diferentemente das métricas anteriores, **MCC** varia de -1 (pior cenário) a 1 (melhor cenário: previsão perfeita).

Observe que **Recall** e **Sensibilidade** são essencialmente a mesma quantidade.

Entre as métricas discutidas, algumas boas opções para avaliar modelos, em especial para conjuntos de dados desequilibrados, são **MCC** e F_1**-Score**, enquanto **Precision** e **Recall** também fornecem informações úteis. Não daremos muita importância à métrica **Accuracy** neste projeto, pois ela produz conclusões errôneas quando as classes não estão equilibradas. No problema em questão, o falso negativo (uma transação fraudulenta classificada como autêntica) é mais perigoso do que o falso positivo (uma transação autêntica classificada como fraudulenta), pois, no primeiro caso, o fraudador pode causar mais danos financeiros, enquanto no segundo caso o banco pode verificar a autenticidade da transação do usuário do cartão depois de tomar as medidas necessárias para proteger o cartão. Considerando esse fato, damos ao F_2**-Score** uma importância especial na avaliação dos modelos.

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

3. Train-Test Split

Splitting the data into training set and testing set

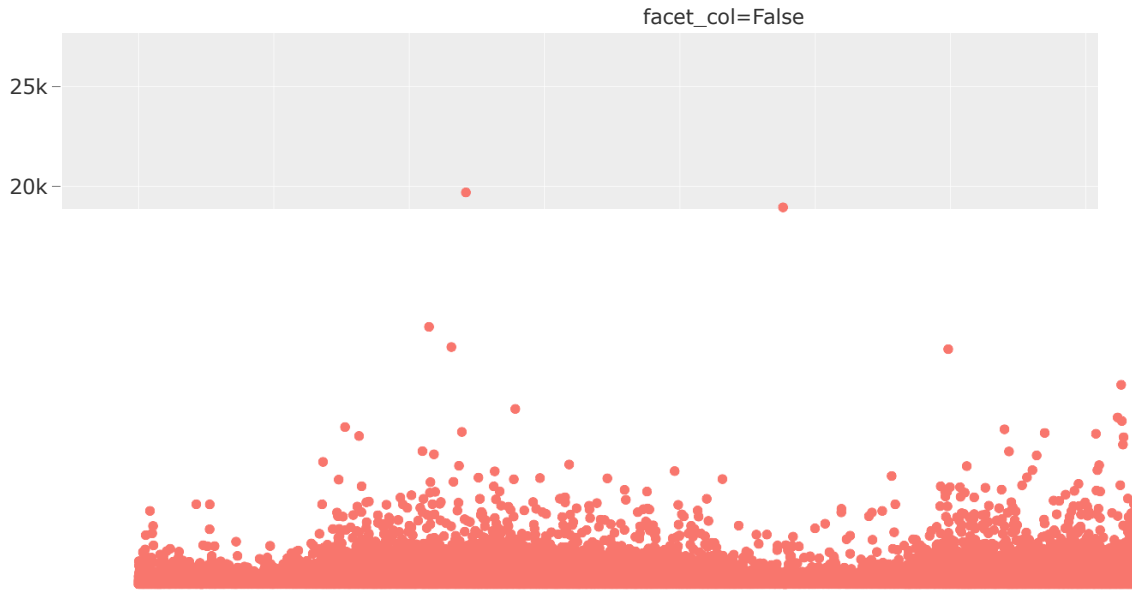
In [ ]:
```
# Separação das variáveis independentes e da variável-alvo

y = data['Class'] # alvo
X = data.drop('Class', axis = 1) # independente

# Construção do conjunto de treinamento e do conjunto de teste

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

Balancing the training set

Separating the training set by class

In [ ]:
```
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]
```

In [ ]:
```
# Valor vs. tempo para transações autênticas e fraudulentas no conjunto de treinamento

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()
```



Subamostragem aleatória (RUS)

Pegamos um subconjunto da classe majoritária para equilibrar o conjunto de dados de treinamento.

Vantagem: Melhora o tempo de execução e resolve qualquer problema de armazenamento devido ao grande conjunto de dados de aprendizado.

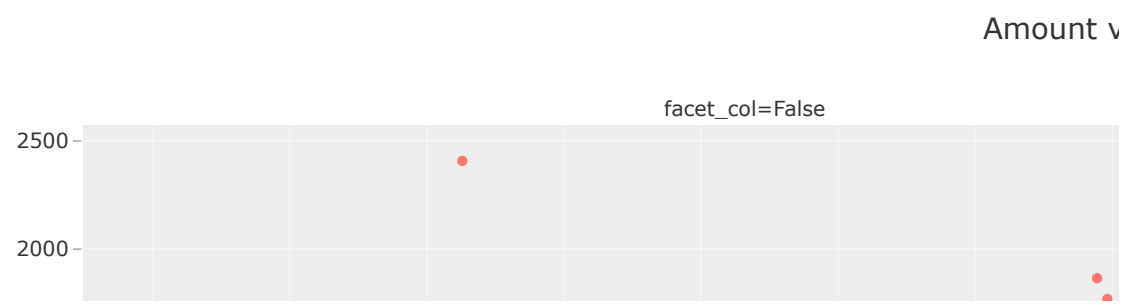
Desvantagem: Ignora um pedaço de informação que poderia ser impactante na análise e usa apenas uma amostra representativa da classe majoritária, o que não garante que reflita a mesma com precisão.

In [ ]:
```
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
```

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

In [ ]:
```
# Valor vs. tempo para transações autênticas e fraudulentas no conjunto de treinamento após suba

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()
```



Comparando com os mesmos gráficos do conjunto de dados completo, vemos que as transações autênticas de alto valor não são representadas na amostra retirada da classe majoritária. Isso indica uma desvantagem geral das técnicas de subamostragem, que jogam fora uma grande parte das informações da classe majoritária e não representam as mesmas com precisão.

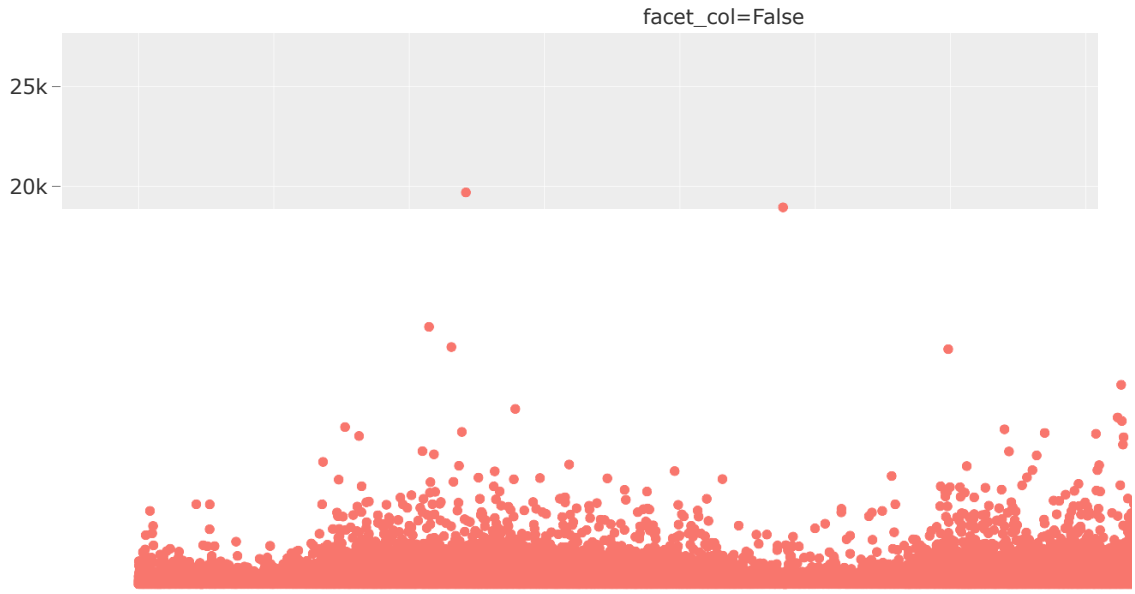
Random over-sampling (ROS)

In [ ]:
```
data_train_fraudulent_over = data_train_fraudulent.sample(len(data_train_authentic), replace = '
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
```

Out[ ]: <Axes: title={'center': 'Class frequencies after over-sampling'}>

In [ ]:
```
# Valor vs. tempo para transações autênticas e fraudulentas no conjunto de treinamento após amos

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()
```



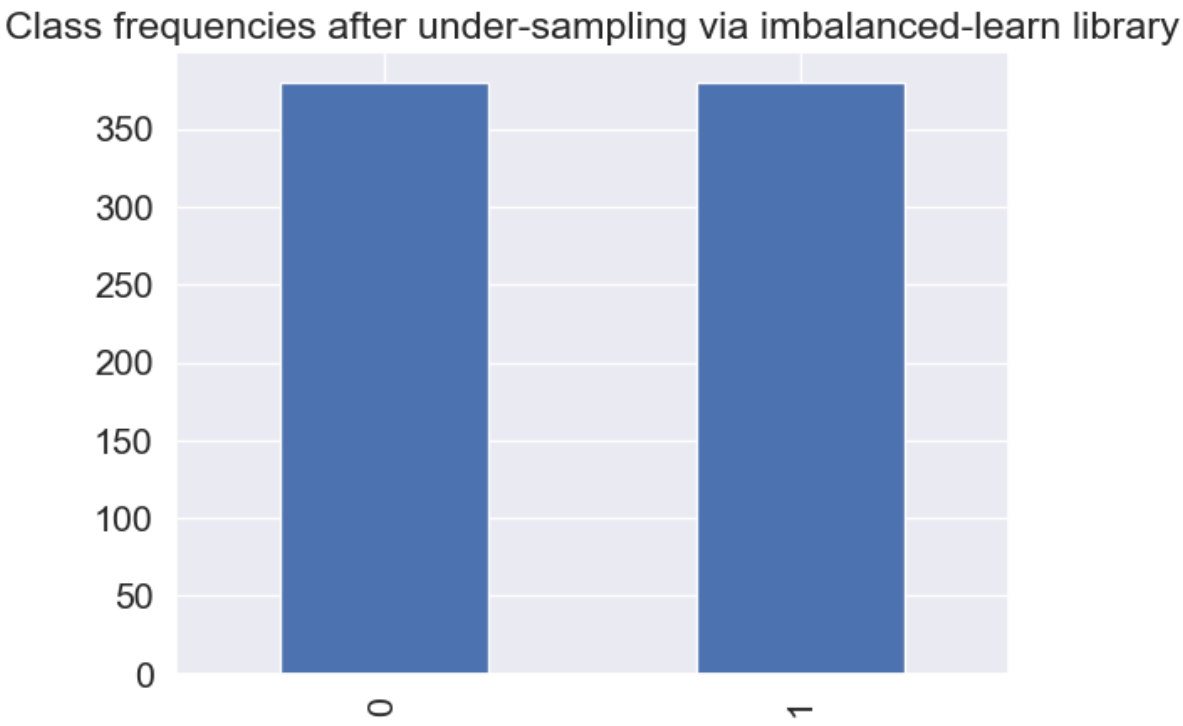
Esses gráficos se assemelham aos gráficos *Amount vs Time* correspondentes para o conjunto de dados completo com muito mais precisão do que os gráficos correspondentes para o conjunto de treinamento obtido da subamostragem aleatória.

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

In [ ]:
```
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 i
```

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

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



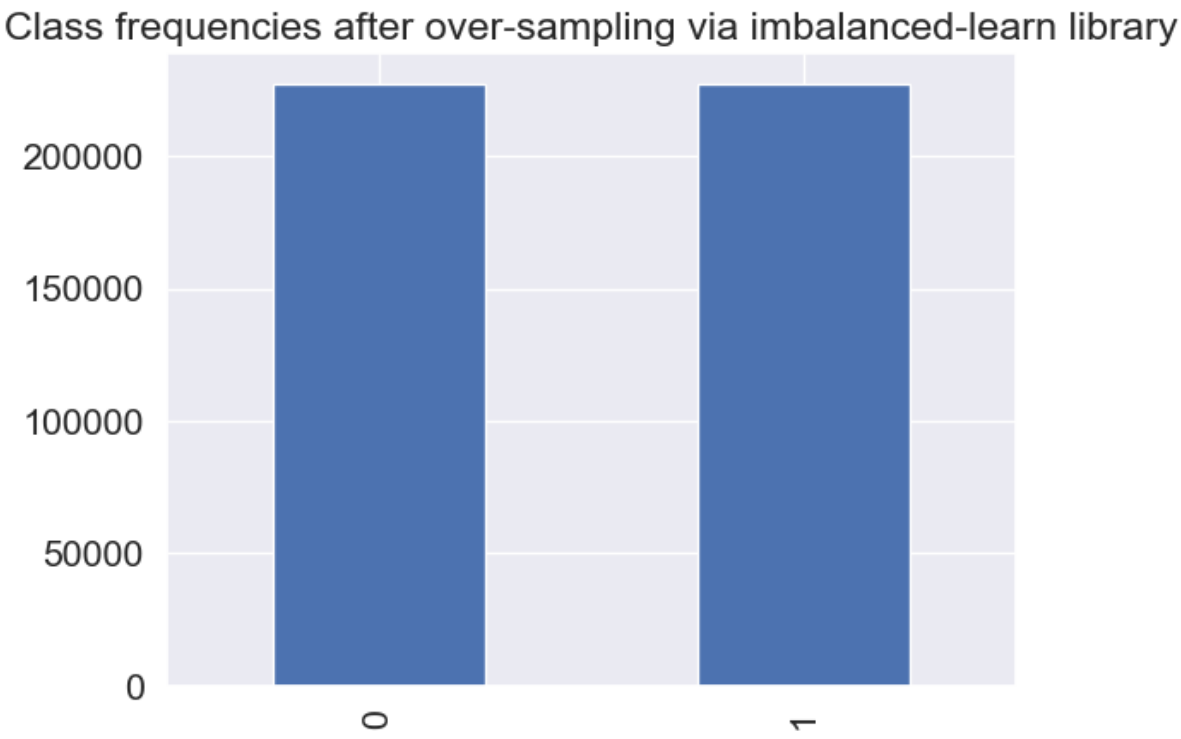
In [ ]:
```
# Valor vs. tempo para transações autênticas e fraudulentas no conjunto de treinamento após suba

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()
```

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

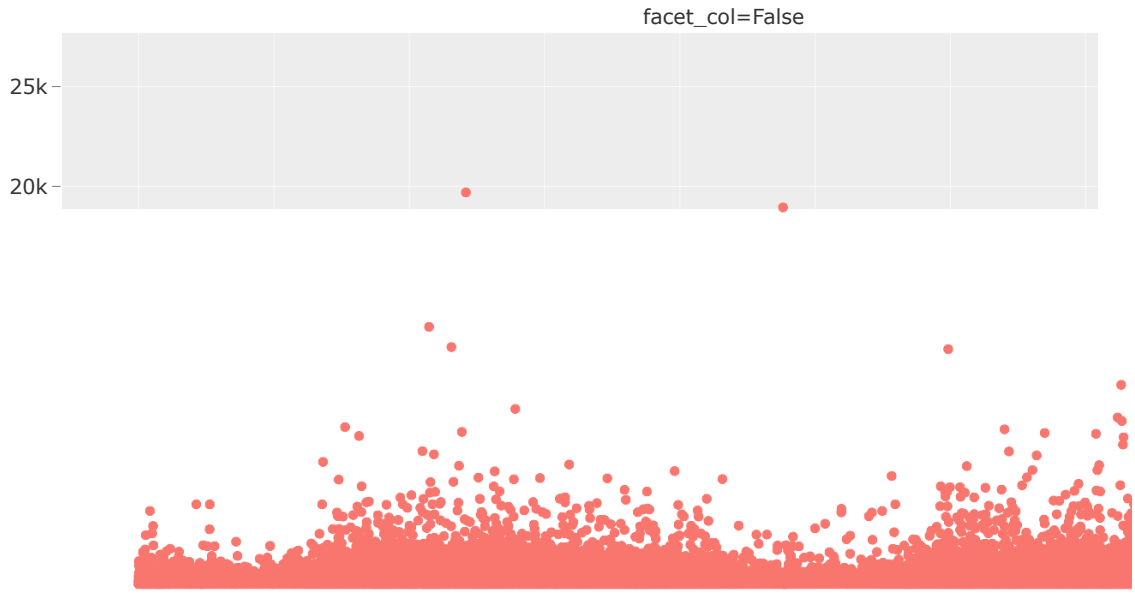
In [ ]:
```
imblearn_ros = RandomOverSampler(random_state = 40)
X_train_ros, y_train_ros = imblearn_ros.fit_resample(X_train, y_train)

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 imb
```

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

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



In [ ]:
```
# Valor vs. tempo para transações autênticas e fraudulentas no conjunto de treinamento após amos

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()
```

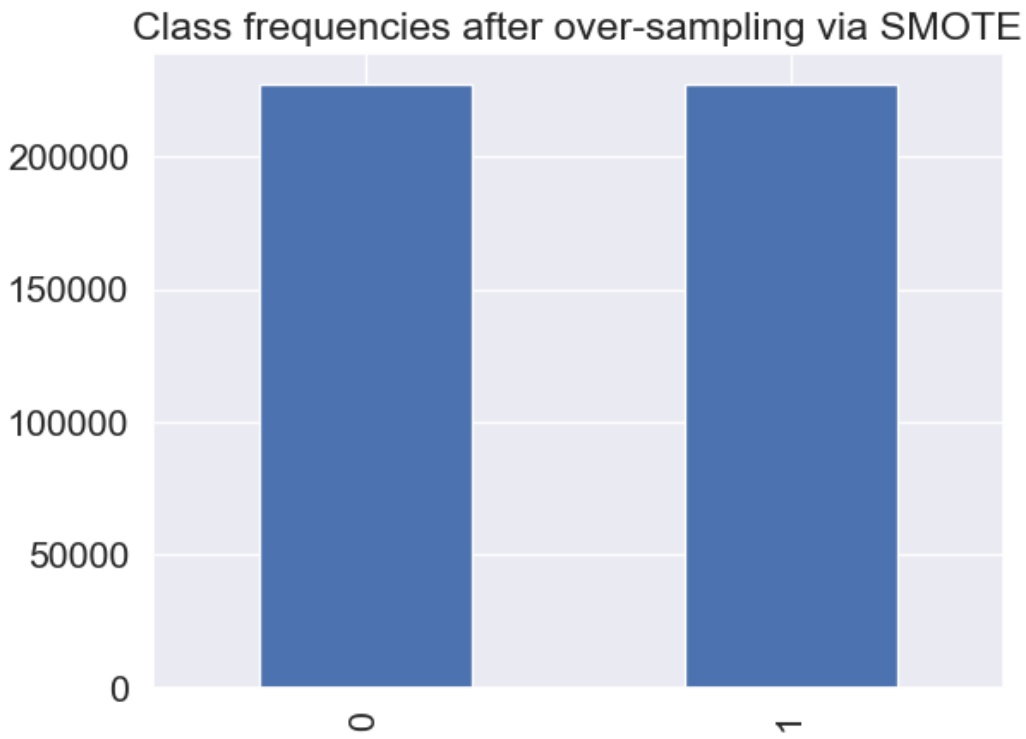


Synthetic minority over-sampling technique (SMOTE)

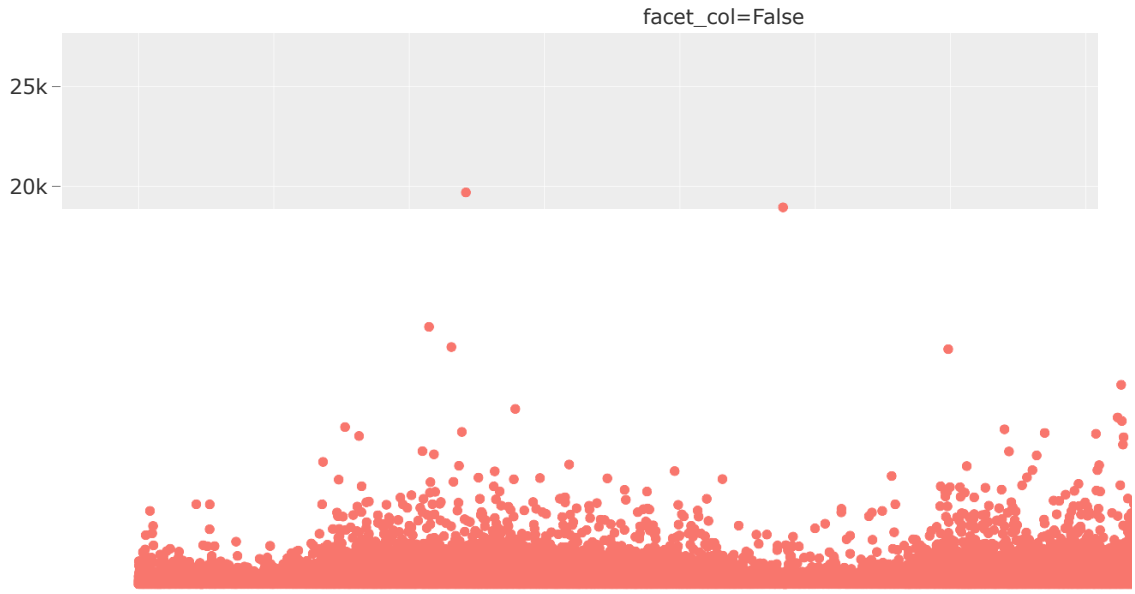
In [ ]:
```
smote = SMOTE()
X_train_smote, y_train_smote = smote.fit_resample(X_train, y_train)

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-sampl
```

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

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



In [ ]:
```
# Valor vs. tempo para transações autênticas e fraudulentas no conjunto de treinamento após a am

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()
```

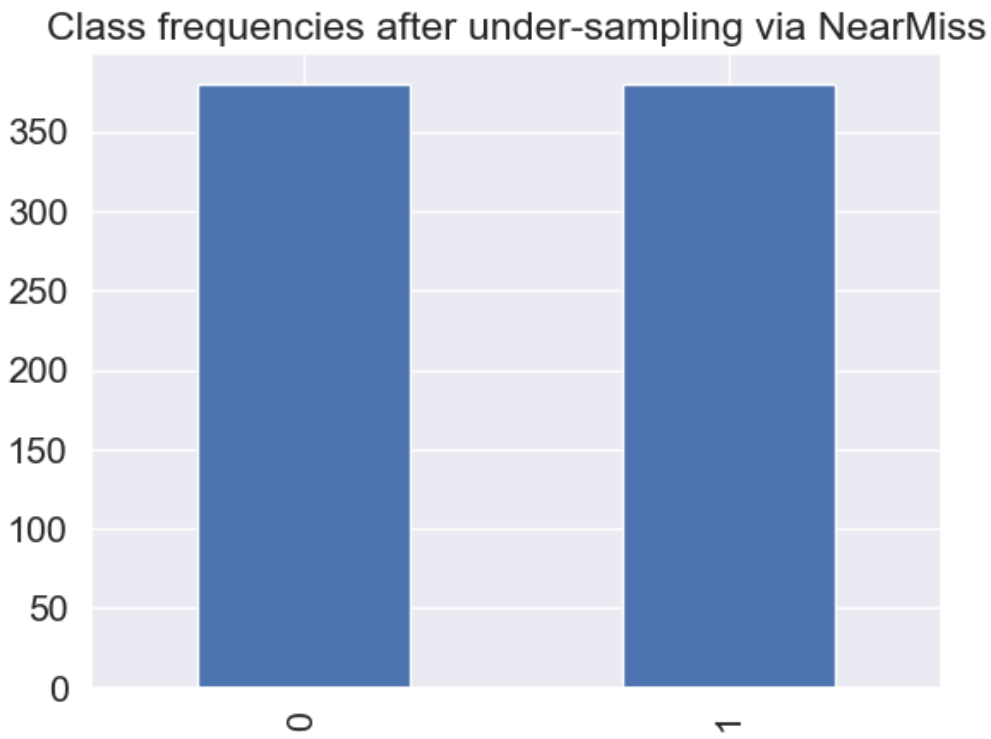


Under-sampling via NearMiss (NM)

In [ ]:
```
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-sampli
```

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

Out[ ]: <Axes: title={'center': 'Class frequencies after under-sampling via NearMiss'}>



In [ ]:
```
# Valor versus tempo para transações autênticas e fraudulentas no conjunto de treinamento após s

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()
```

No gráfico à esquerda, fica claro que a classe majoritária não é representada com precisão no esquema de subamostragem via NearMiss.

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

4. Feature Scaling

Pode ser natural que um dos recursos contribua mais para o processo de classificação do que outro. Mas, muitas vezes, isso é causado artificialmente pela diferença de intervalo de valores que os recursos assumem (geralmente devido às unidades em que os recursos são medidos). Muitos algoritmos, especialmente os baseados em árvores, como árvore de decisão e floresta aleatória, bem como classificadores baseados em modelos gráficos, como análise discriminante linear e Bayes ingênuo, são invariáveis ao dimensionamento e, portanto, são indiferentes ao dimensionamento de recursos. Por outro lado, os algoritmos baseados em distâncias ou semelhanças, que incluem k-vizinhos mais próximos, máquina de vetor de suporte e descida de gradiente estocástico, são sensíveis ao dimensionamento. Isso exige que o profissional dimensione os recursos adequadamente antes de alimentar os dados com esses classificadores.

In [ ]:
```
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

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

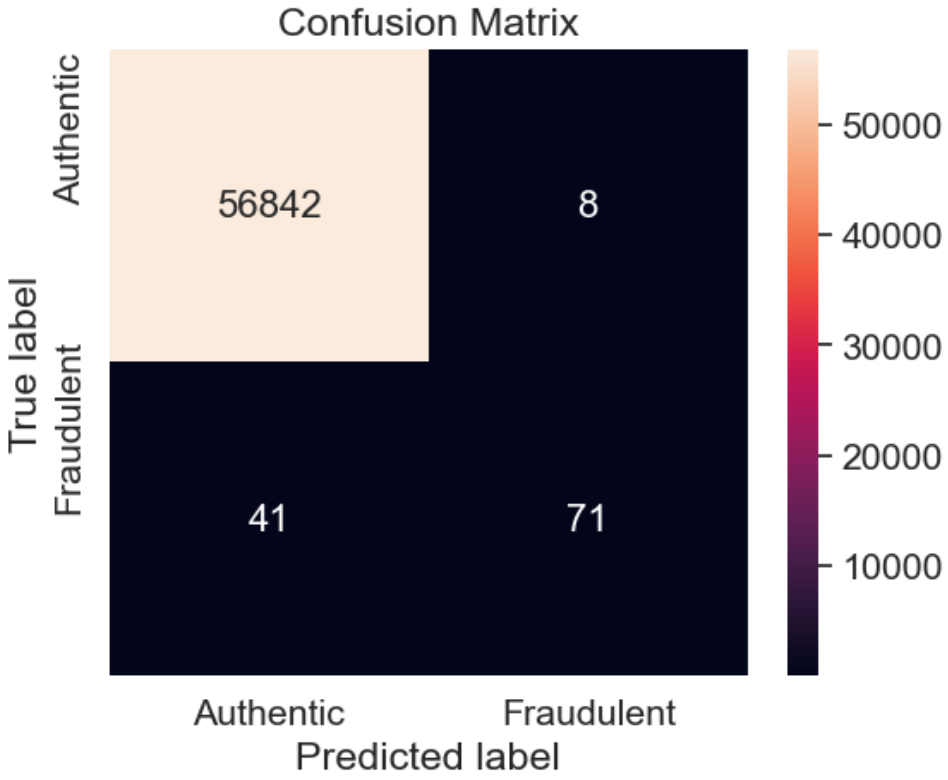
In [ ]:
```
#  matriz de confusão, métricas de avaliação e visualização de classes

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 s
    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)
```

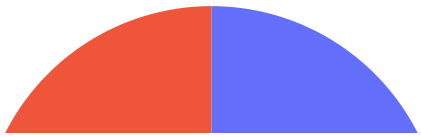
```
    MCC_num = (TN * TP) - (FP * FN)
    MCC_denom = np.sqrt((FP + TP) * (FN + TP) * (TN + FP) * (TN + FN))

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

   # Resumo

    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_mea

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

    # Desempenho do modelo por meio da matriz de confusão

    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()
```

Unaltered training set

In [ ]:

```
# Elementos da matriz de confusão

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

# Resumo das métricas de avaliação

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', ave
summary_logreg_unaltered_extended.loc[len(summary_logreg_unaltered_extended.index)] = ['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
```





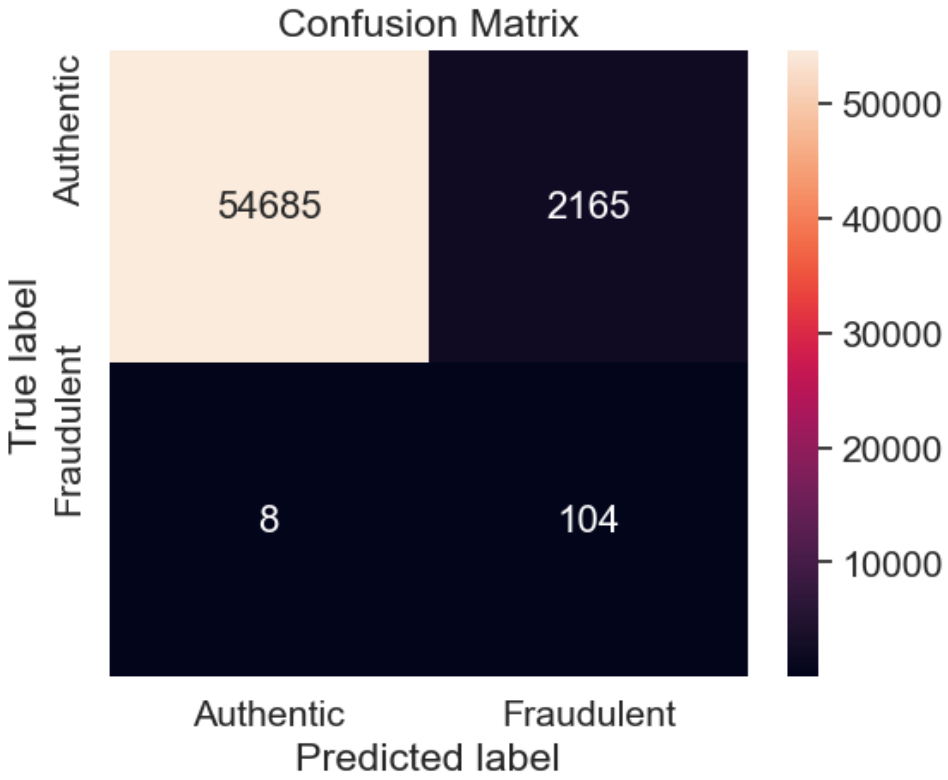
Out[ ]:

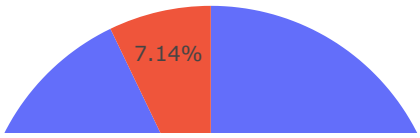
Metric	MCC	F1-Score	F2-Score	Recall	Precision	FM index	Specificity	G-mean	F0.5-Score	A
Performance score	0.75442	0.743455	0.673624	0.633929	0.898734	0.754807	0.999859	0.79614	0.829439	

Observação: Embora o modelo de regressão logística no conjunto de treinamento inalterado tenha um desempenho extremamente bom na classe negativa (transações autênticas), ele não funciona tão bem com a classe positiva crítica (transações fraudulentas), pois classifica erroneamente mais de um terço das transações nessa classe.

Random under-sampling

In [ ]:
```
# Elementos da matriz de confusão

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

# Resumo das métricas de avaliação

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_pre
summary_logreg_under_extended.loc[len(summary_logreg_under_extended.index)] = ['ROC-AUC', roc_au
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
```





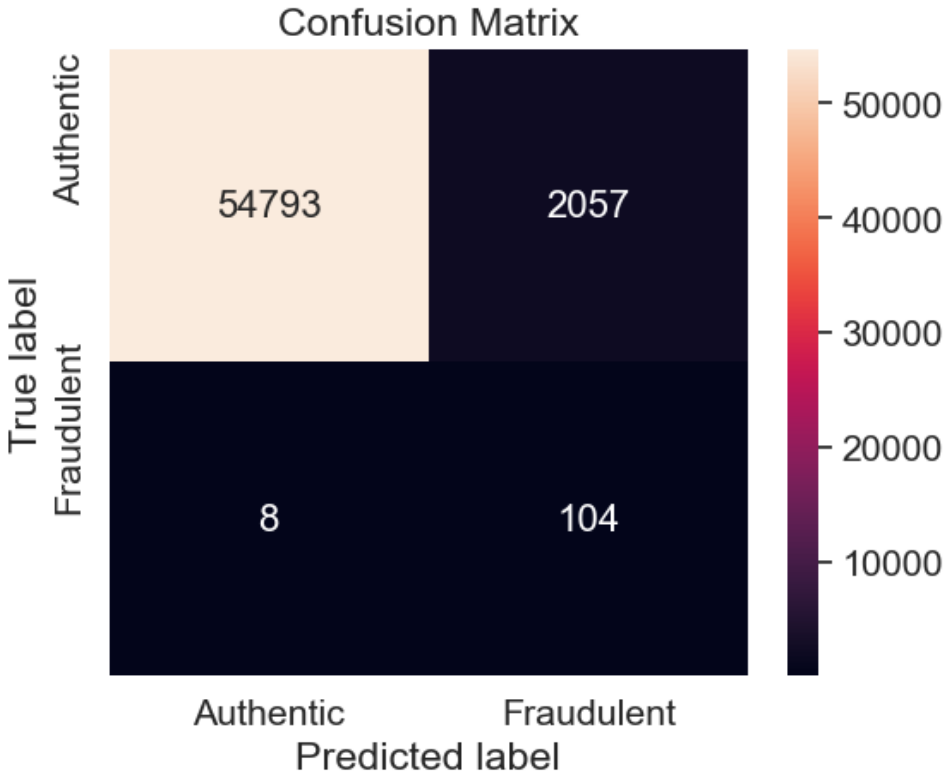
Out[ ]:

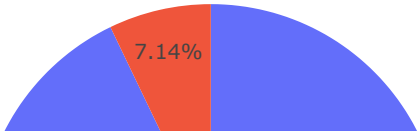
Metric	MCC	F1-Score	F2-Score	Recall	Precision	FM index	Specificity	G-mean	F0.5-Score
Performance score	0.201706	0.087358	0.191388	0.928571	0.045835	0.206304	0.961917	0.945097	0.056596

Random over-sampling

In [ ]:

```
# Elementos da matriz de confusão

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

# Resumo das métricas de avaliação

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_preci
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
```





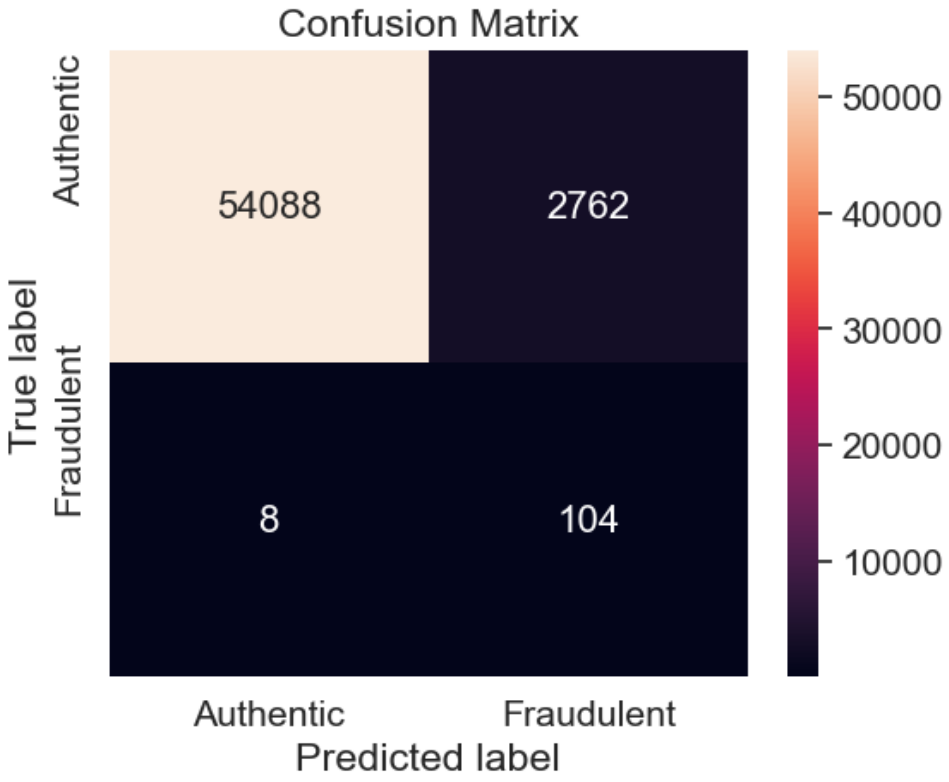
Out[ ]:

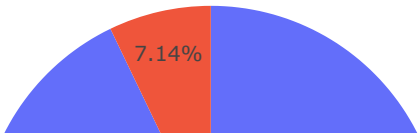
	Metric	MCC	F1-Score	F2-Score	Recall	Precision	FM index	Specificity	G-mean	F0.5-Score	A
Performance score		0.206922	0.091509	0.19931	0.928571	0.048126	0.211396	0.963817	0.94603	0.059388	0

Random under-sampling with imbalanced-learn library

In [ ]:
```
# Elementos da matriz de confusão

classification(logreg, X_train_under_imblearn, y_train_under_imblearn, X_test, y_test)
```

```
# Resumo das métricas de avaliação

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)] =
summary_logreg_under_imblearn_extended.loc[len(summary_logreg_under_imblearn_extended.index)] =
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 =
summary_logreg_under_imblearn_index
```



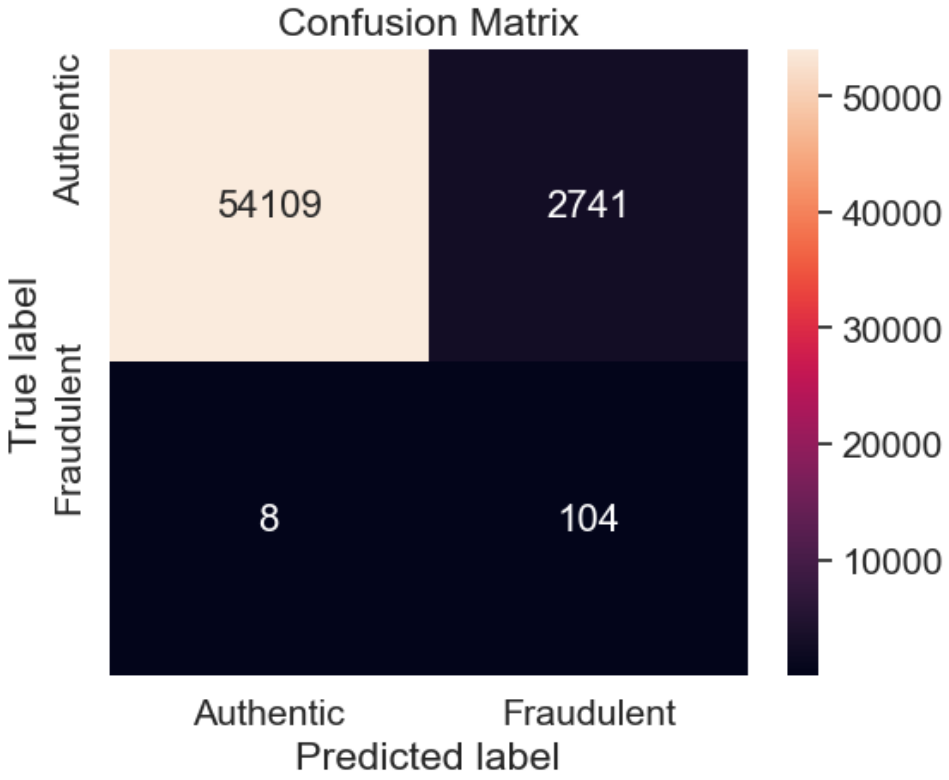


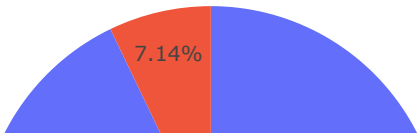
Out[ ]:

Metric	MCC	F1-Score	F2-Score	Recall	Precision	FM index	Specificity	G-mean	F0.5-Score	A
Performance score	0.178332	0.069846	0.15691	0.928571	0.036288	0.183563	0.951416	0.939924	0.044921	

Random over-sampling with imbalanced-learn library

In [ ]:
```
# Elementos da matriz de confusão

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

# Resumo das métricas de avaliação

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)] = ['
summary_logreg_over_imblearn_extended.loc[len(summary_logreg_over_imblearn_extended.index)] = ['
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 = T
summary_logreg_over_imblearn_index
```





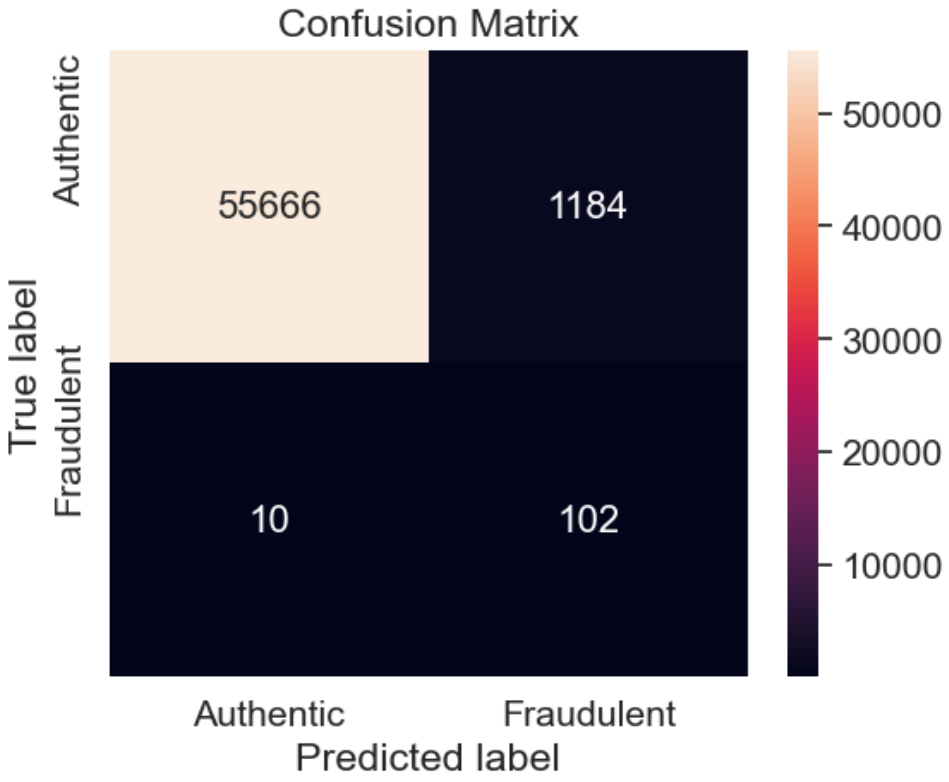
Out[ ]:

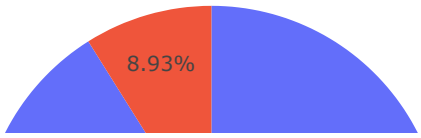
	Metric	MCC	F1-Score	F2-Score	Recall	Precision	FM index	Specificity	G-mean	F0.5-Score	A
Performance score		0.17903	0.070342	0.157911	0.928571	0.036555	0.18424	0.951785	0.940107	0.045249	

Synthetic minority over-sampling technique (SMOTE)

In [ ]:
```
# Elementos da matriz de confusão

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

```
# Resumo das métricas de avaliação

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', a
summary_logreg_over_smote_extended.loc[len(summary_logreg_over_smote_extended.index)] = ['ROC-AU
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
```





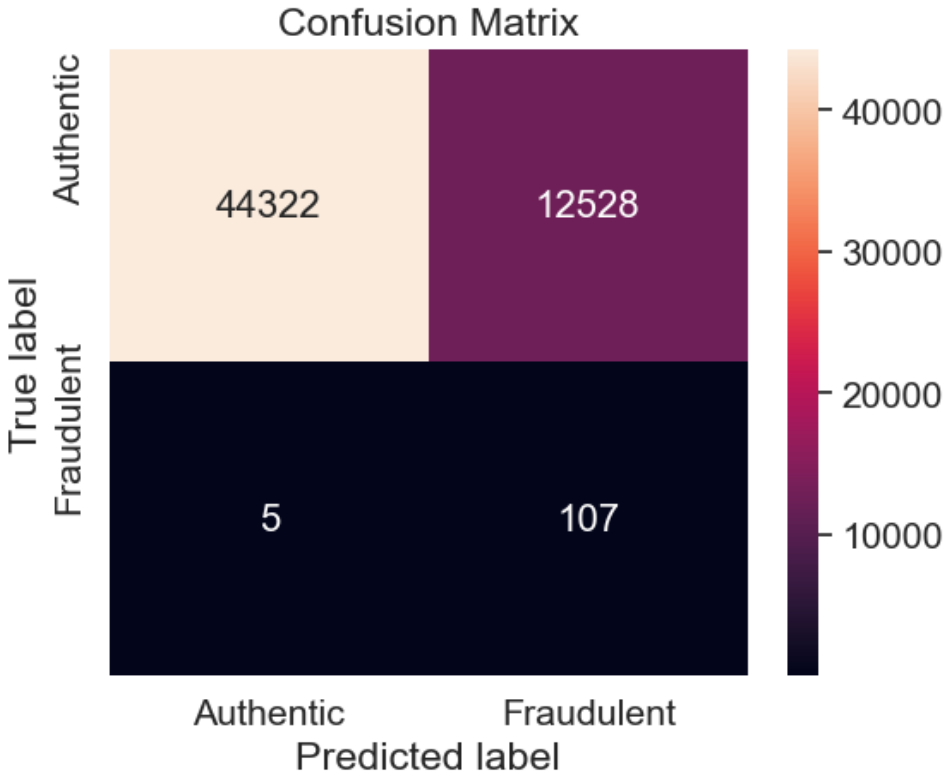
Out[ ]:

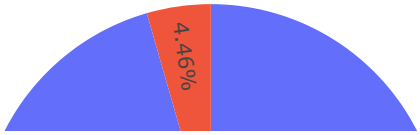
Metric	MCC	F1-Score	F2-Score	Recall	Precision	FM index	Specificity	G-mean	F0.5-Score
Performance score	0.265372	0.145923	0.294118	0.910714	0.079316	0.268764	0.979173	0.944324	0.097032

Under-sampling via NearMiss

In [ ]:

```
# Elementos da matriz de confusão

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

# Resumo das métricas de avaliação

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', avera
summary_logreg_under_nm_extended.loc[len(summary_logreg_under_nm_extended.index)] = ['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
```





Out[ ]:

Metric	MCC	F1-Score	F2-Score	Recall	Precision	FM index	Specificity	G-mean	F0.5-Score
Performance score	0.078367	0.016788	0.040893	0.955357	0.008469	0.089947	0.779631	0.863033	0.010562

Summary of logistic regression models

Tendo em mente que o conjunto de dados é altamente desequilibrado e que a classe positiva (transações fraudulentas) é mais importante do que a classe negativa (transações autênticas), informamos **MCC**, **F1-**

Score, **F2-Score** e **Recall** para cada modelo considerado. Além disso, informamos **Precisão**, **Índice FM**, **Exatidão** e **Especificidade**.métricas de avaliação aproximadas

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

summary_logreg['Metric'] = EvalMetricLabels
summary_logreg_list = [summary_logreg_unaltered, summary_logreg_under, summary_logreg_over, summ
                       summary_logreg_over_imblearn, summary_logreg_over_smote, summary_logreg_u

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
```

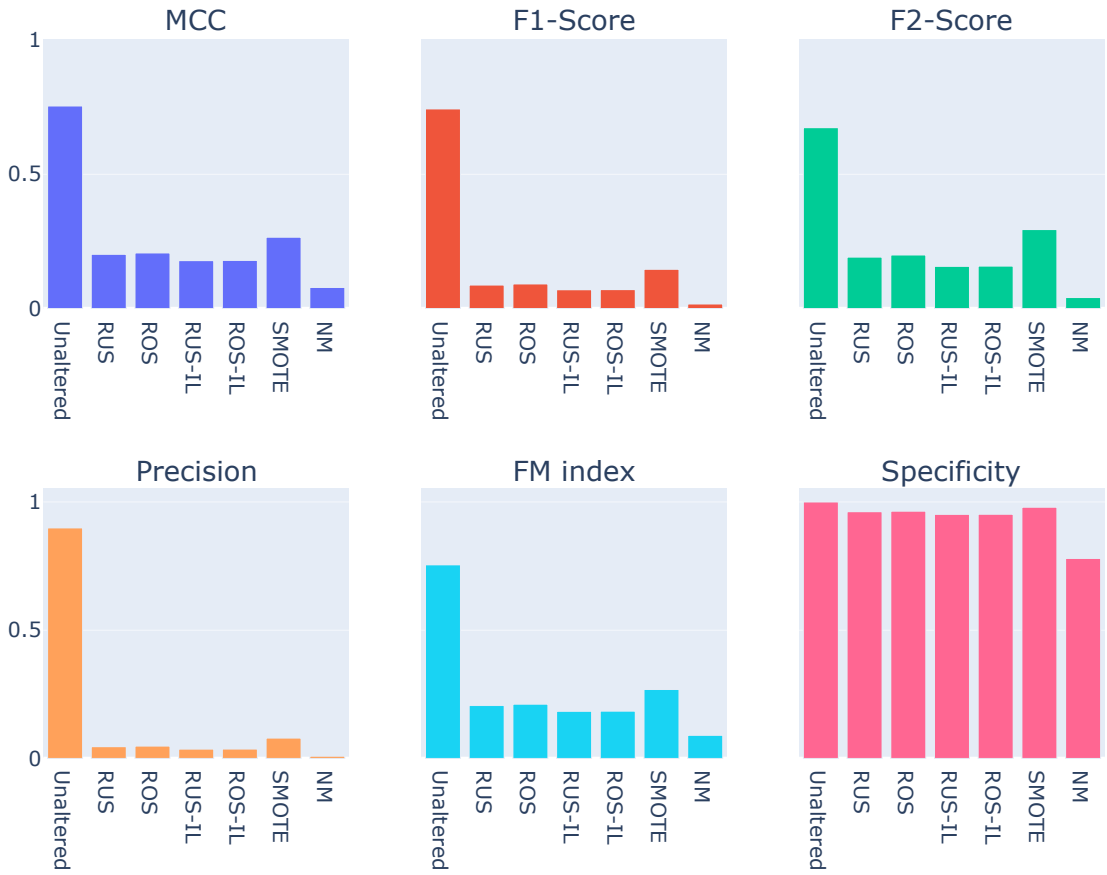
```
C:\Users\Diones\AppData\Local\Temp\ipykernel_14028\1296588952.py:8: FutureWarning:

Passing 'suffixes' which cause duplicate columns {'Performance score_x'} in the result is depreca
ted and will raise a MergeError in a future version.

C:\Users\Diones\AppData\Local\Temp\ipykernel_14028\1296588952.py:8: FutureWarning:

Passing 'suffixes' which cause duplicate columns {'Performance score_x'} in the result is depreca
ted and will raise a MergeError in a future version.
```

Out[ ]:

Metric	Unaltered	RUS	ROS	RUS-IL	ROS-IL	SMOTE	NM
MCC	0.754420	0.201706	0.206922	0.178332	0.179030	0.265372	0.078367
F1-Score	0.743455	0.087358	0.091509	0.069846	0.070342	0.145923	0.016788
F2-Score	0.673624	0.191388	0.199310	0.156910	0.157911	0.294118	0.040893
Recall	0.633929	0.928571	0.928571	0.928571	0.928571	0.910714	0.955357
Precision	0.898734	0.045835	0.048126	0.036288	0.036555	0.079316	0.008469
FM index	0.754807	0.206304	0.211396	0.183563	0.184240	0.268764	0.089947
Specificity	0.999859	0.961917	0.963817	0.951416	0.951785	0.979173	0.779631
G-mean	0.796140	0.945097	0.946030	0.939924	0.940107	0.944324	0.863033
F0.5-Score	0.829439	0.056596	0.059388	0.044921	0.045249	0.097032	0.010562
Accuracy	0.999140	0.961852	0.963748	0.951371	0.951740	0.979039	0.779976

In [ ]:
```
# Função para comparar visualmente o desempenho do modelo aplicado em diferentes conjuntos de tr

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

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

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

In [ ]:
```
# Comparação visual do modelo aplicado em diferentes conjuntos de treinamento por meio de métric

summary_visual(summary_logreg)
```



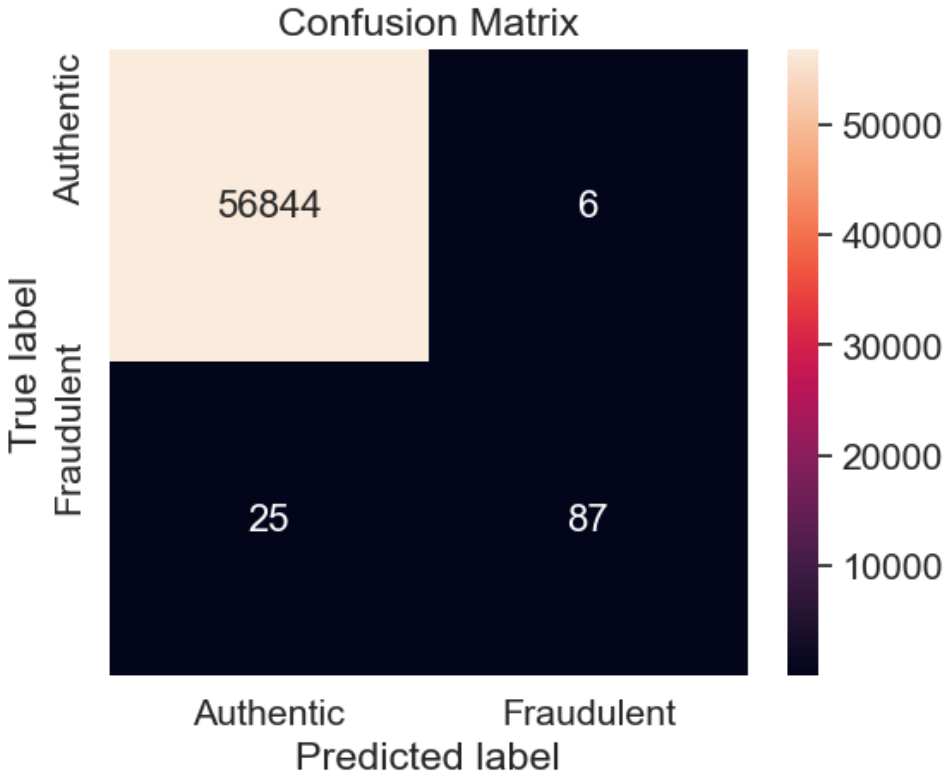
6. k-Nearest Neighbors (k-NN)

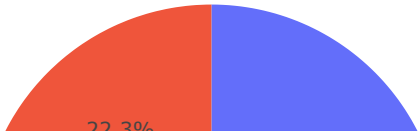
In [ ]:
```
k = 29
knn = KNeighborsClassifier(n_neighbors = k, n_jobs = -1)
```

Unaltered training set

In [ ]:
```
# Elementos da matriz de confusão

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

# Resumo das métricas de avaliação

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_
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
```





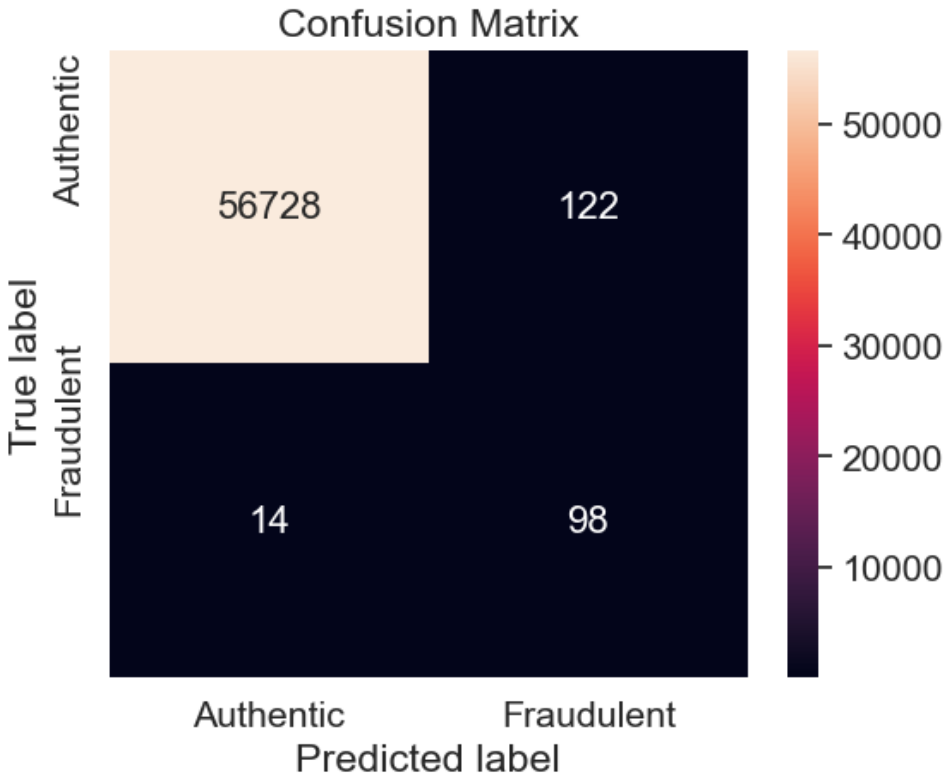
```
e:\miniconda3\envs\Bootcamp\lib\site-packages\sklearn\base.py:432: UserWarning:

X has feature names, but KNeighborsClassifier was fitted without feature names
```

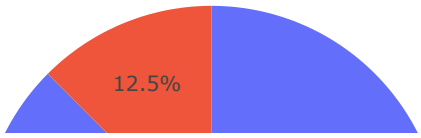
Out[ ]:

Metric	MCC	F1-Score	F2-Score	Recall	Precision	FM index	Specificity	G-mean	F0.5-Score	Ac
Performance score	0.852191	0.84878	0.804067	0.776786	0.935484	0.85245	0.999894	0.881308	0.89876	0.9

Random under-sampling

In [ ]:
```
# Elementos da matriz de confusão

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

# Resumo das métricas de avaliação

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
```






```
e:\miniconda3\envs\Bootcamp\lib\site-packages\sklearn\base.py:432: UserWarning:

X has feature names, but KNeighborsClassifier was fitted without feature names
```

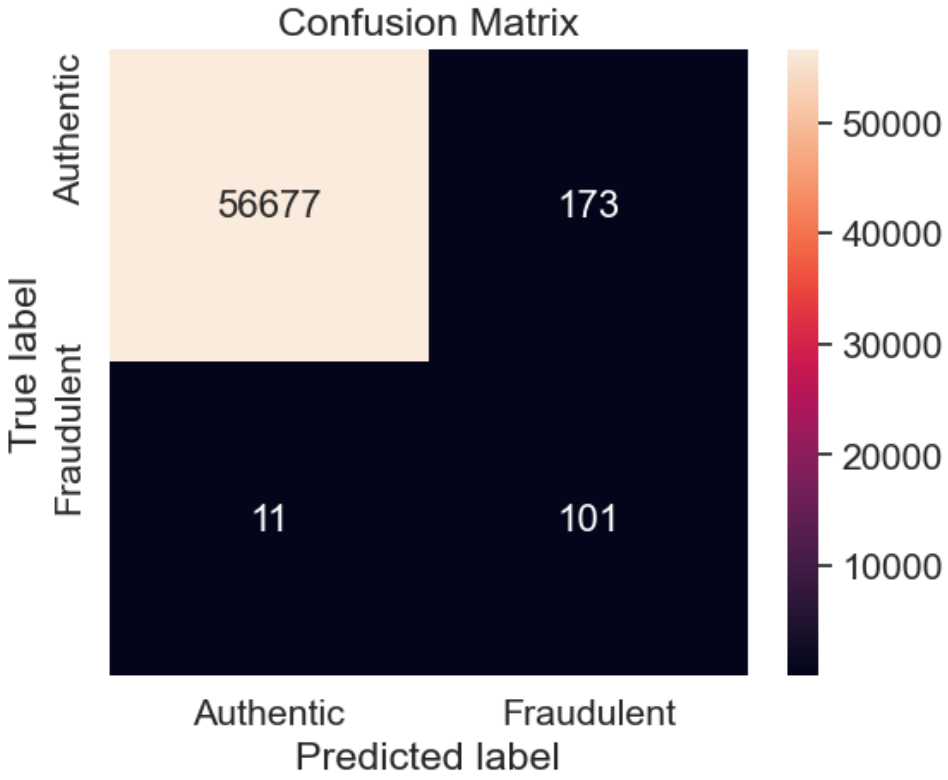
Out[ ]:

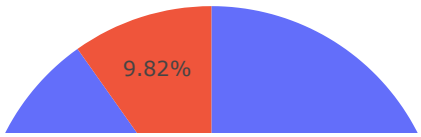
Metric	MCC	F1-Score	F2-Score	Recall	Precision	FM index	Specificity	G-mean	F0.5-Score	Acc
Performance score	0.623379	0.590361	0.733533	0.875	0.445455	0.624318	0.997854	0.93441	0.493952	0.9

Random over-sampling

In [ ]:

```
# Elementos da matriz de confusão

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

# Resumo das métricas de avaliação

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
```





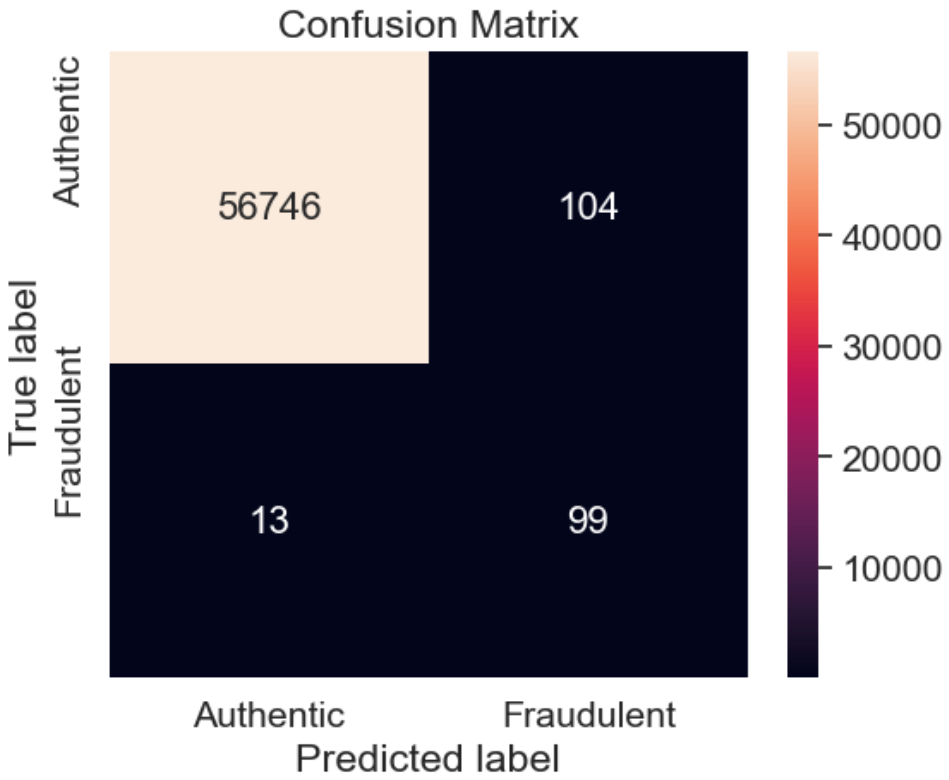

```
e:\miniconda3\envs\Bootcamp\lib\site-packages\sklearn\base.py:432: UserWarning:

X has feature names, but KNeighborsClassifier was fitted without feature names
```

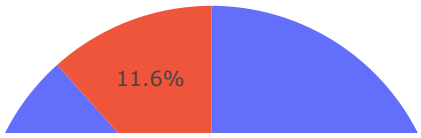
Out[ ]:

Metric	MCC	F1-Score	F2-Score	Recall	Precision	FM index	Specificity	G-mean	F0.5-Score	A
Performance score	0.575425	0.523316	0.699446	0.901786	0.368613	0.57655	0.996957	0.948178	0.418046	

Random under-sampling with imbalanced-learn library

In [ ]:
```
# Elementos da matriz de confusão

classification(knn, X_train_under_imblearn_scaled_minmax, y_train_under_imblearn, X_test_scaled_

# Resumo das métricas de avaliação

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-
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
```






```
e:\miniconda3\envs\Bootcamp\lib\site-packages\sklearn\base.py:432: UserWarning:

X has feature names, but KNeighborsClassifier was fitted without feature names
```

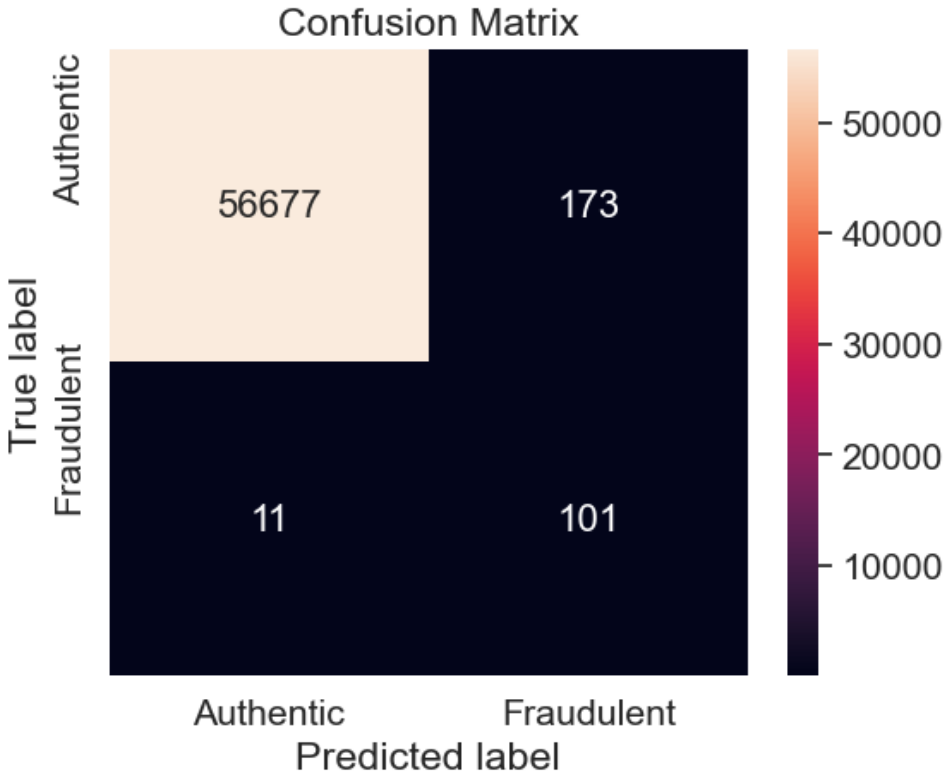
Out[ ]:

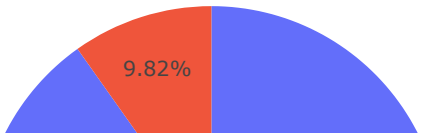
Metric	MCC	F1-Score	F2-Score	Recall	Precision	FM index	Specificity	G-mean	F0.5-Score
Performance score	0.655732	0.628571	0.760369	0.883929	0.487685	0.656566	0.998171	0.939314	0.535714

Random over-sampling with imbalanced-learn library

In [ ]:

```
# Elementos da matriz de confusão

classification(knn, X_train_over_imblearn_scaled_minmax, y_train_over_imblearn, X_test_scaled_mi

# Resumo das métricas de avaliação

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-AU
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
```





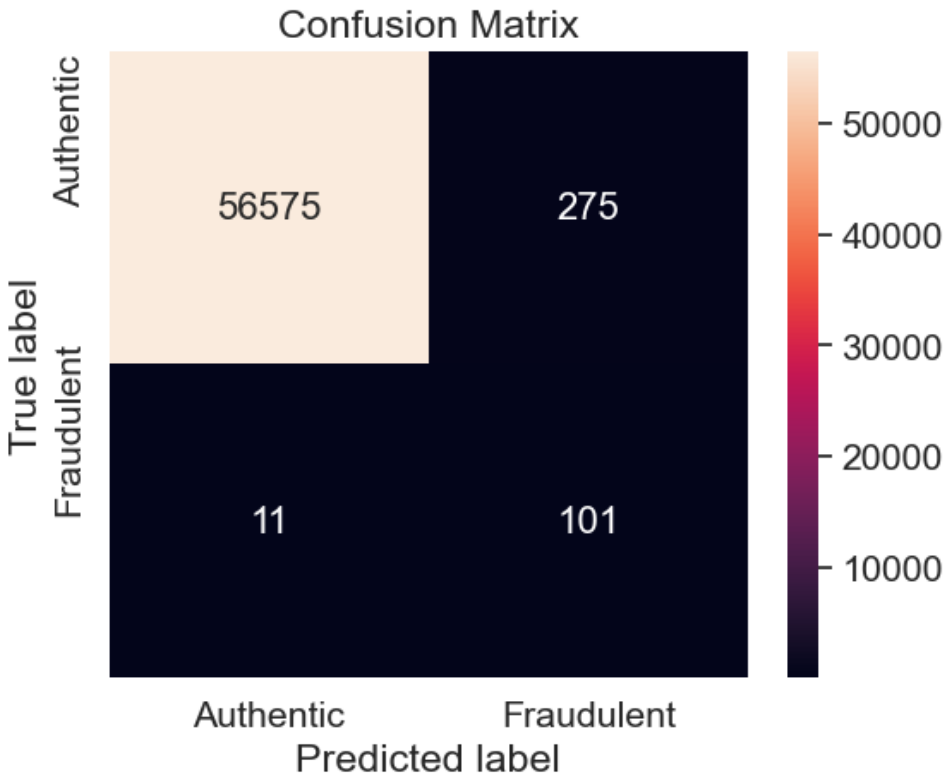

```
e:\miniconda3\envs\Bootcamp\lib\site-packages\sklearn\base.py:432: UserWarning:

X has feature names, but KNeighborsClassifier was fitted without feature names
```

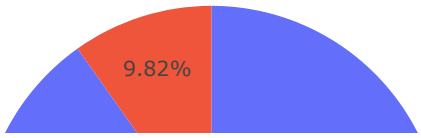
Out[ ]:

Metric	MCC	F1-Score	F2-Score	Recall	Precision	FM index	Specificity	G-mean	F0.5-Score
Performance score	0.575425	0.523316	0.699446	0.901786	0.368613	0.57655	0.996957	0.948178	0.418046

Synthetic minority over-sampling technique (SMOTE)

In [ ]:
```
# Elementos da matriz de confusão

classification(knn, X_train_over_smote_scaled_minmax, y_train_over_smote, X_test_scaled_minmax,

# Resumo das métricas de avaliação

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', ro
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
```






```
e:\miniconda3\envs\Bootcamp\lib\site-packages\sklearn\base.py:432: UserWarning:

X has feature names, but KNeighborsClassifier was fitted without feature names
```

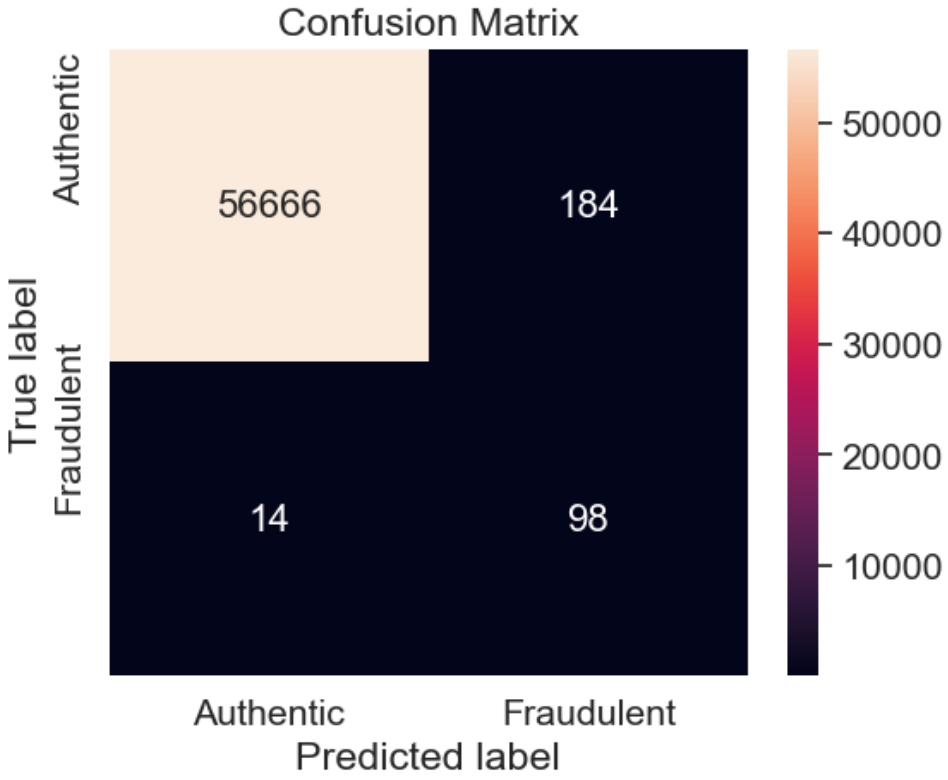
Out[ ]:

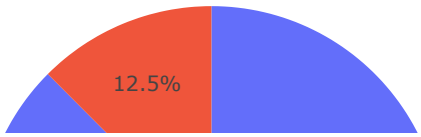
Metric	MCC	F1-Score	F2-Score	Recall	Precision	FM index	Specificity	G-mean	F0.5-Score	A
Performance score	0.490674	0.413934	0.612864	0.901786	0.268617	0.492174	0.995163	0.947324	0.3125	0

Under-sampling via NearMiss

In [ ]:

```
# Elementos da matriz de confusão

classification(knn, X_train_under_nm_scaled_minmax, y_train_under_nm, X_test_scaled_minmax, y_te

# Resumo das métricas de avaliação

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_au
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
```






```
e:\miniconda3\envs\Bootcamp\lib\site-packages\sklearn\base.py:432: UserWarning:

X has feature names, but KNeighborsClassifier was fitted without feature names
```

Out[ ]:

Metric	MCC	F1-Score	F2-Score	Recall	Precision	FM index	Specificity	G-mean	F0.5-Score	Ac
Performance score	0.550216	0.497462	0.671233	0.875	0.347518	0.551433	0.996763	0.933899	0.395161	0.

Summary of k-NN classification models

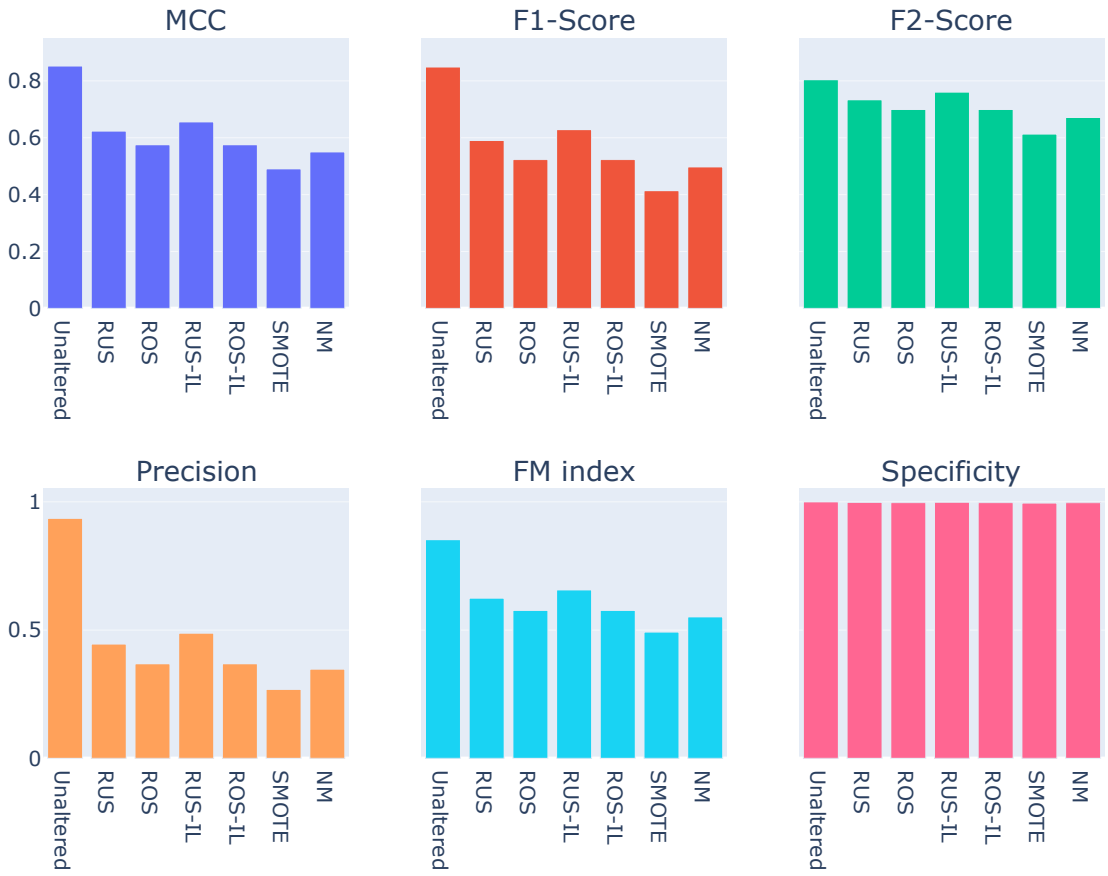
In [ ]:
```
summary_knn = pd.DataFrame(columns = ['Metric'])

summary_knn['Metric'] = EvalMetricLabels
summary_knn_list = [summary_knn_unaltered, summary_knn_under, summary_knn_over, summary_knn_unde
                    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
```

```
C:\Users\Diones\AppData\Local\Temp\ipykernel_14028\1710798844.py:8: FutureWarning:

Passing 'suffixes' which cause duplicate columns {'Performance score_x'} in the result is depreca
ted and will raise a MergeError in a future version.

C:\Users\Diones\AppData\Local\Temp\ipykernel_14028\1710798844.py:8: FutureWarning:

Passing 'suffixes' which cause duplicate columns {'Performance score_x'} in the result is depreca
ted and will raise a MergeError in a future version.
```

Out[ ]:

Metric	Unaltered	RUS	ROS	RUS-IL	ROS-IL	SMOTE	NM
MCC	0.852191	0.623379	0.575425	0.655732	0.575425	0.490674	0.550216
F1-Score	0.848780	0.590361	0.523316	0.628571	0.523316	0.413934	0.497462
F2-Score	0.804067	0.733533	0.699446	0.760369	0.699446	0.612864	0.671233
Recall	0.776786	0.875000	0.901786	0.883929	0.901786	0.901786	0.875000
Precision	0.935484	0.445455	0.368613	0.487685	0.368613	0.268617	0.347518
FM index	0.852450	0.624318	0.576550	0.656566	0.576550	0.492174	0.551433
Specificity	0.999894	0.997854	0.996957	0.998171	0.996957	0.995163	0.996763
G-mean	0.881308	0.934410	0.948178	0.939314	0.948178	0.947324	0.933899
F0.5-Score	0.898760	0.493952	0.418046	0.535714	0.418046	0.312500	0.395161
Accuracy	0.999456	0.997612	0.996770	0.997946	0.996770	0.994979	0.996524

In [ ]:
```
# Comparação visual do modelo aplicado em diferentes conjuntos de treinamento por meio de várias

summary_visual(summary_knn)
```



Nota: Um possível problema com os modelos de classificação k-NN, que é relevante neste projeto, é que eles são afetados pela **cursão da dimensionalidade**, bem como pela **presença de outliers nas variáveis de recursos**. Apesar disso, ele tem um desempenho bastante bom quando aplicado ao conjunto de treinamento inalterado (desequilibrado), especialmente com relação ao **MCC**, mas também ao **F2-score**.

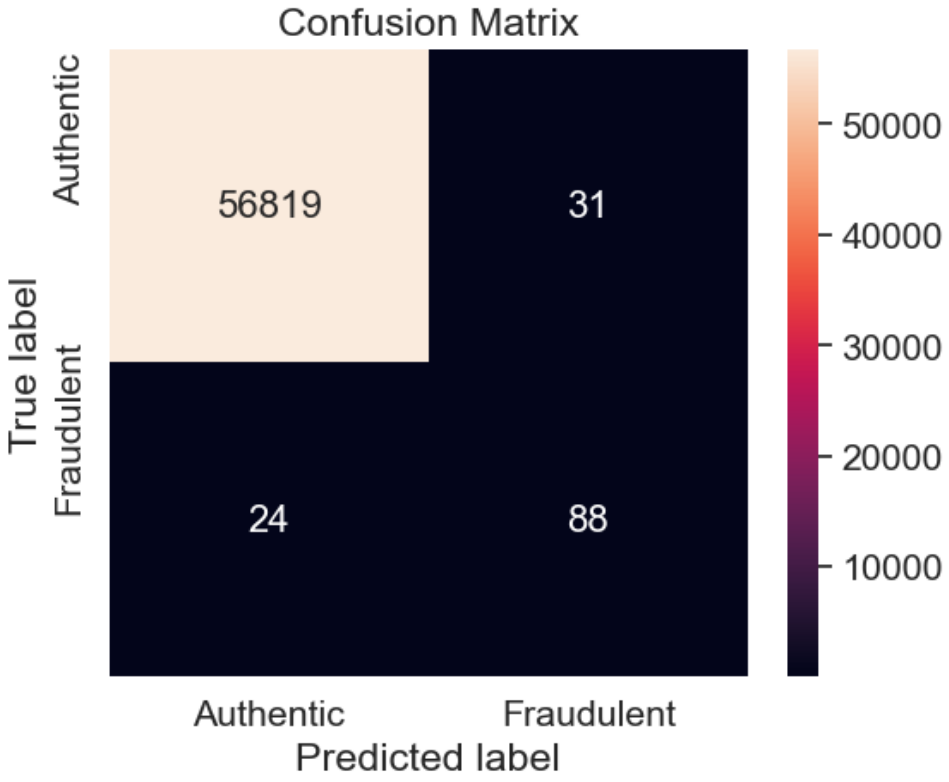
7. Decision Tree

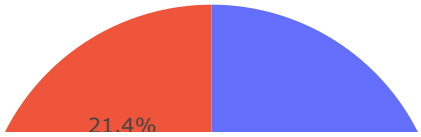
In [ ]:
```
dt = DecisionTreeClassifier()
```

Unaltered training set

In [ ]:
```
# Elementos da matriz de confusão

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

# Resumo das métricas de avaliação

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_au
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
```



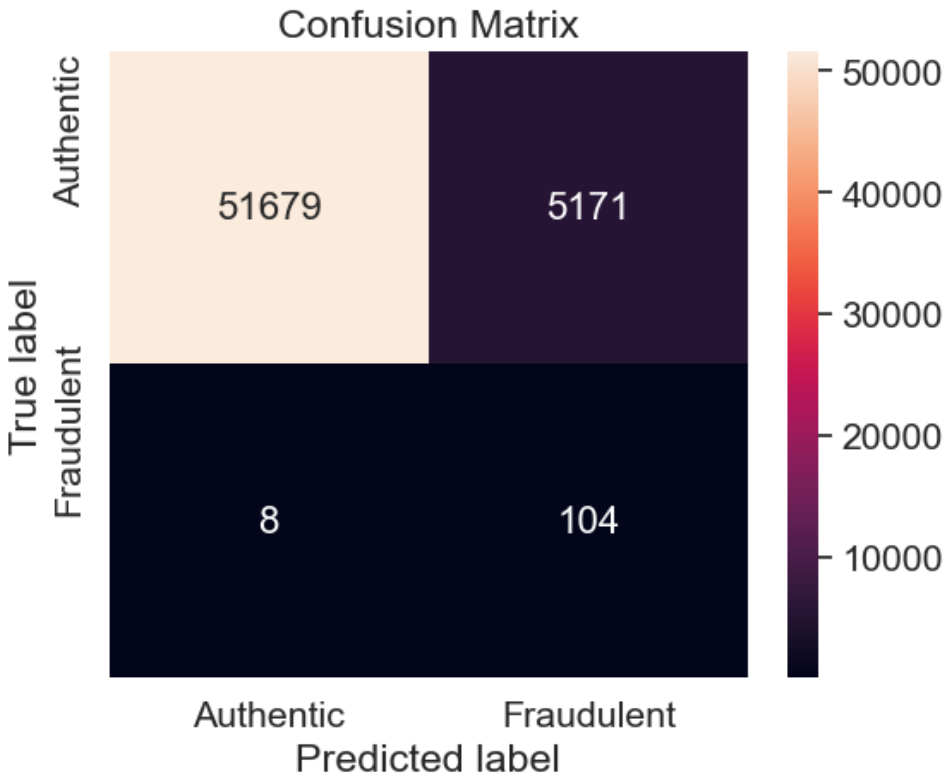


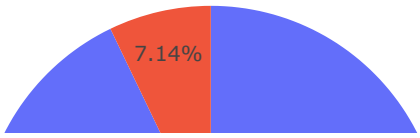
Out[ ]:

	Metric	MCC	F1-Score	F2-Score	Recall	Precision	FM index	Specificity	G-mean	F0.5-Score
Performance score		0.761773	0.761905	0.776014	0.785714	0.739496	0.762255	0.999455	0.886164	0.748299

Random under-sampling

In [ ]:
```
# Elementos da matriz de confusão

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

# Resumo das métricas de avaliação

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
```





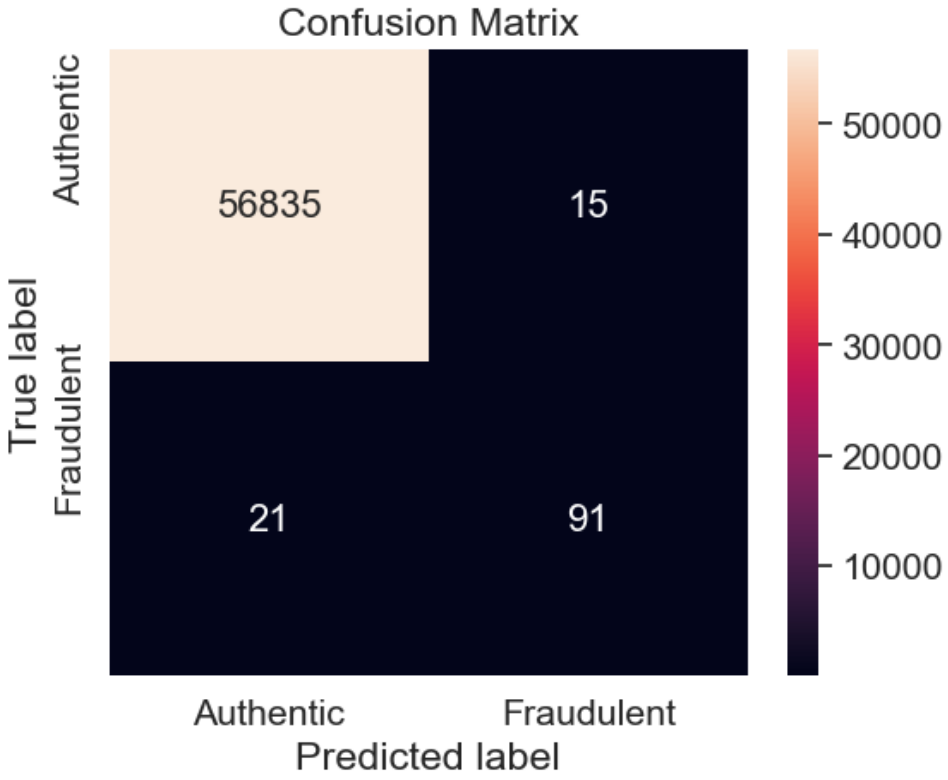
Out[ ]:

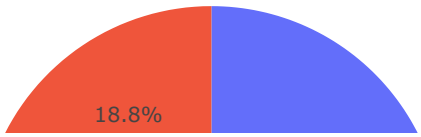
Metric	MCC	F1-Score	F2-Score	Recall	Precision	FM index	Specificity	G-mean	F0.5-Score
Performance score	0.128002	0.038611	0.090861	0.928571	0.019716	0.135305	0.909041	0.918754	0.024514

Random over-sampling

In [ ]:

```
# Elementos da matriz de confusão

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

# Resumo das métricas de avaliação

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
```





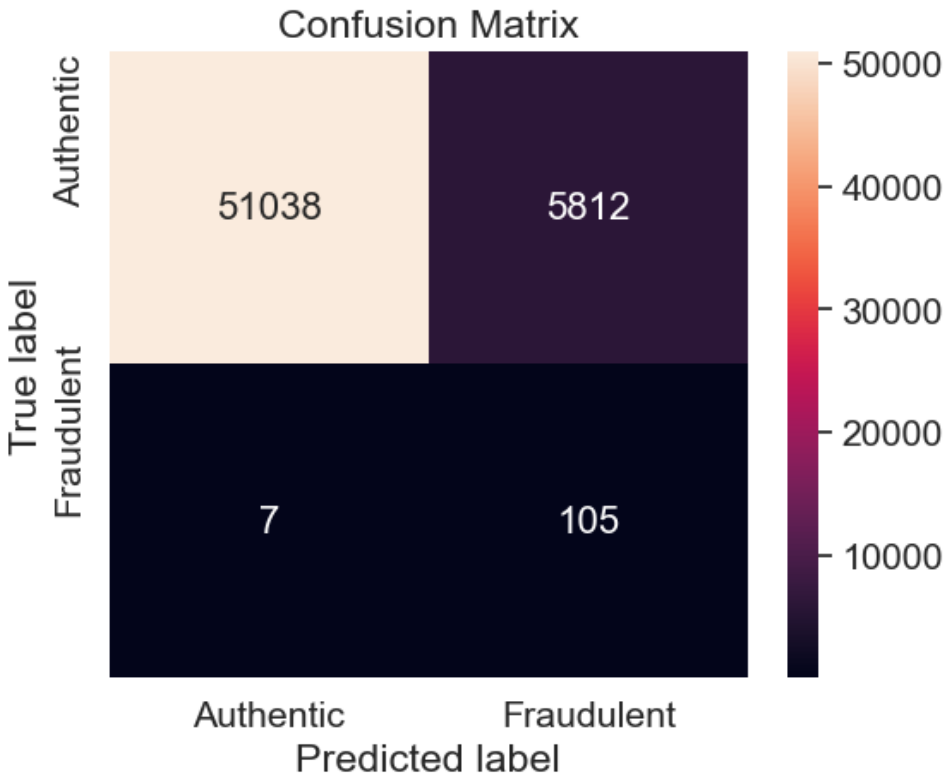
Out[ ]:

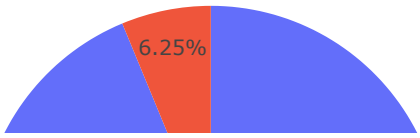
	Metric	MCC	F1-Score	F2-Score	Recall	Precision	FM index	Specificity	G-mean	F0.5-Score	Acc
Performance score		0.834864	0.834862	0.8213	0.8125	0.858491	0.835179	0.999736	0.901269	0.848881	0.9

Random under-sampling with imbalanced-learn library

In [ ]:
```
# Elementos da matriz de confusão

classification(dt, X_train_under_imblearn, y_train_under_imblearn, X_test, y_test)
```

```
# Resumo das métricas de avaliação

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-AU
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
```



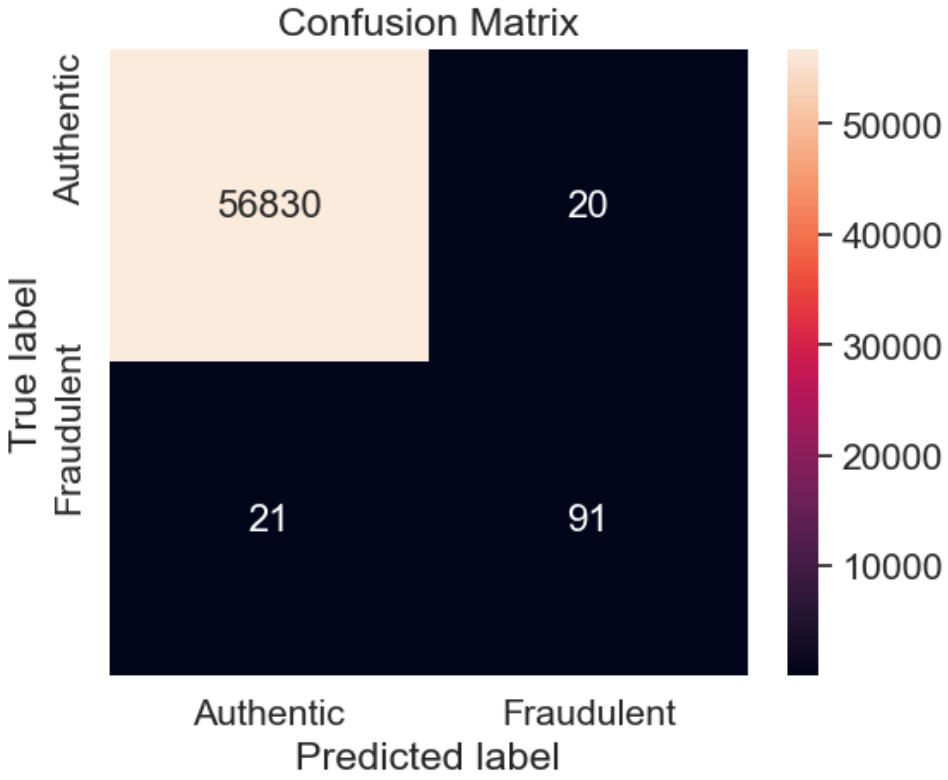


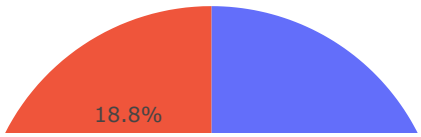
Out[ ]:

Metric	MCC	F1-Score	F2-Score	Recall	Precision	FM index	Specificity	G-mean	F0.5-Score	A
Performance score	0.121275	0.034832	0.082482	0.9375	0.017745	0.128982	0.897766	0.917418	0.022077	0

Random over-sampling with imbalanced-learn library

In [ ]:
```
# Elementos da matriz de confusão

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

# Resumo das métricas de avaliação

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'
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
```





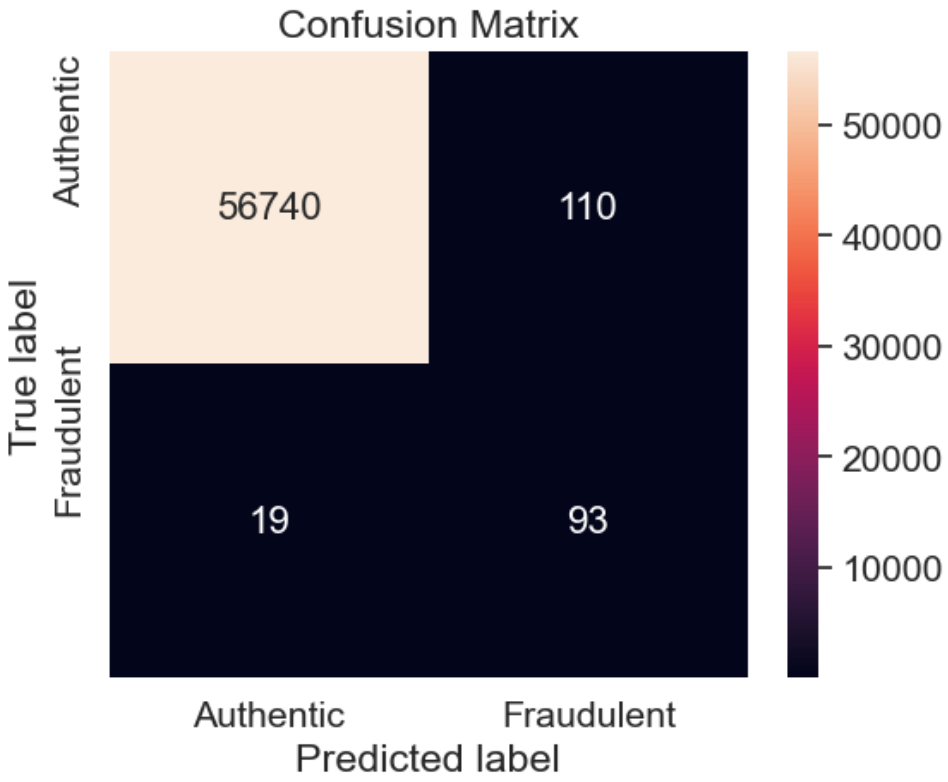
Out[ ]:

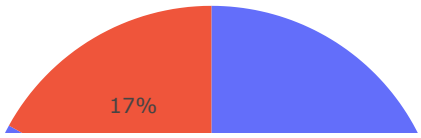
	Metric	MCC	F1-Score	F2-Score	Recall	Precision	FM index	Specificity	G-mean	F0.5-Score	A
Performance score		0.815791	0.816143	0.813953	0.8125	0.81982	0.816152	0.999648	0.901229	0.818345	

Synthetic minority over-sampling technique (SMOTE)

In [ ]:
```
# Elementos da matriz de confusão

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

```
# Resumo das métricas de avaliação

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_
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
```



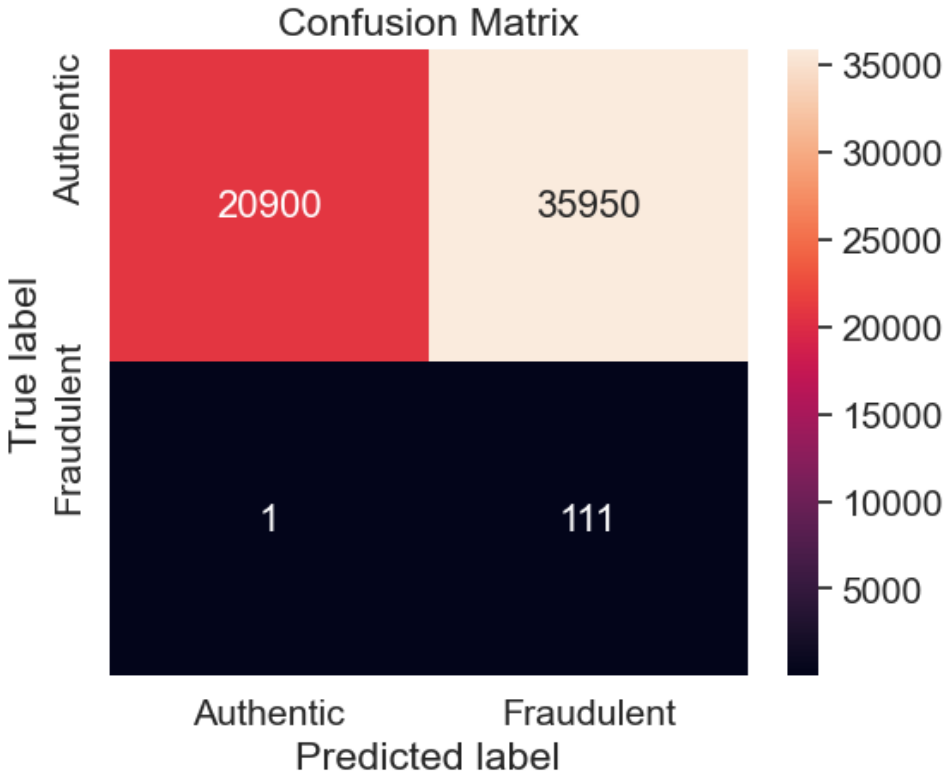


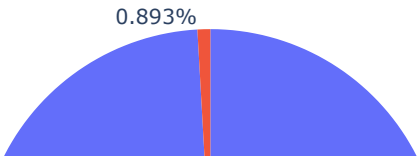
Out[ ]:

Metric	MCC	F1-Score	F2-Score	Recall	Precision	FM index	Specificity	G-mean	F0.5-Score	A
Performance score	0.61583	0.590476	0.714286	0.830357	0.458128	0.616774	0.998065	0.910357	0.503247	

Under-sampling via NearMiss

In [ ]:
```
# Elementos da matriz de confusão

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

# Resumo das métricas de avaliação

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
```





Out[ ]:

Metric	MCC	F1-Score	F2-Score	Recall	Precision	FM index	Specificity	G-mean	F0.5-Score
Metric	MCC	F1-Score	F2-Score	Recall	Precision	FM index	Specificity	G-mean	F0.5-Score
Performance score	0.032969	0.006137	0.015202	0.991071	0.003078	0.055233	0.367634	0.603616	0.003845

Summary of decision tree classification models

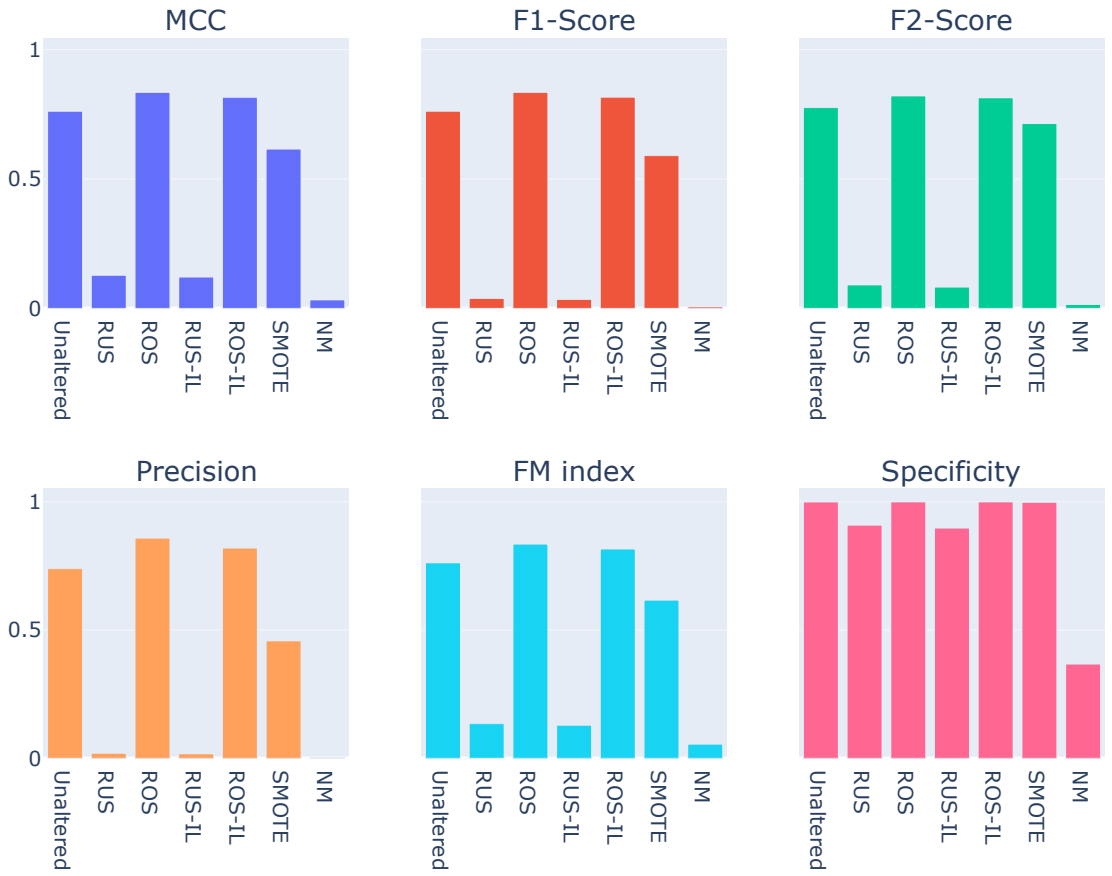
In [ ]:
```
summary_dt = pd.DataFrame(columns = ['Metric'])

EvalMetricLabels_dt = EvalMetricLabels
summary_dt['Metric'] = EvalMetricLabels
summary_dt_list = [summary_dt_unaltered, summary_dt_under, summary_dt_over, summary_dt_under_imb
                   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
```

```
C:\Users\Diones\AppData\Local\Temp\ipykernel_14028\1435390414.py:9: FutureWarning:

Passing 'suffixes' which cause duplicate columns {'Performance score_x'} in the result is depreca
ted and will raise a MergeError in a future version.

C:\Users\Diones\AppData\Local\Temp\ipykernel_14028\1435390414.py:9: FutureWarning:

Passing 'suffixes' which cause duplicate columns {'Performance score_x'} in the result is depreca
ted and will raise a MergeError in a future version.
```

Out[ ]:

Metric	Unaltered	RUS	ROS	RUS-IL	ROS-IL	SMOTE	NM
MCC	0.761773	0.128002	0.834864	0.121275	0.815791	0.615830	0.032969
F1-Score	0.761905	0.038611	0.834862	0.034832	0.816143	0.590476	0.006137
F2-Score	0.776014	0.090861	0.821300	0.082482	0.813953	0.714286	0.015202
Recall	0.785714	0.928571	0.812500	0.937500	0.812500	0.830357	0.991071
Precision	0.739496	0.019716	0.858491	0.017745	0.819820	0.458128	0.003078
FM index	0.762255	0.135305	0.835179	0.128982	0.816152	0.616774	0.055233
Specificity	0.999455	0.909041	0.999736	0.897766	0.999648	0.998065	0.367634
G-mean	0.886164	0.918754	0.901269	0.917418	0.901229	0.910357	0.603616
F0.5-Score	0.748299	0.024514	0.848881	0.022077	0.818345	0.503247	0.003845
Accuracy	0.999034	0.909080	0.999368	0.897844	0.999280	0.997735	0.368860

In [ ]:
```
# Comparação visual do modelo aplicado em diferentes conjuntos de treinamento por meio de várias

summary_visual(summary_dt)
```

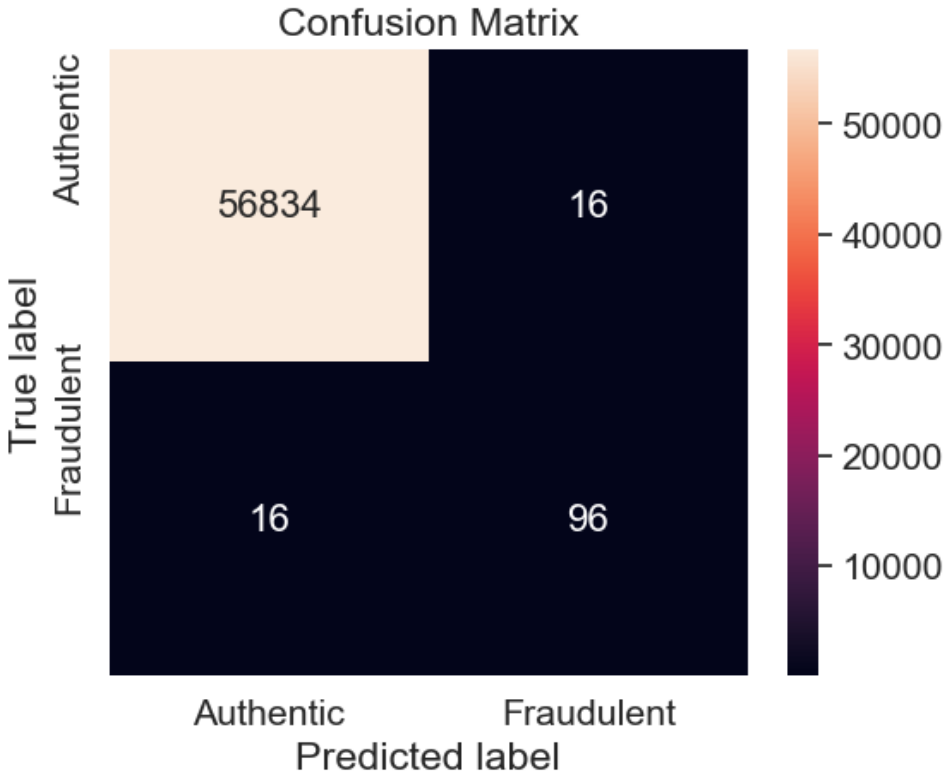


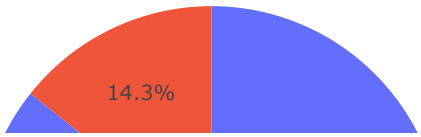
8. Support Vector Machine (SVM)

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

Unaltered training set

In [ ]:
```
# Elementos da matriz de confusão

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

# Resumo das métricas de avaliação

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 = T
summary_svm_linear_unaltered_index

# classification(svm_linear, X_train, y_train, X_test, y_test) # TP = 37, FN = 75, TN = 56840, F
```





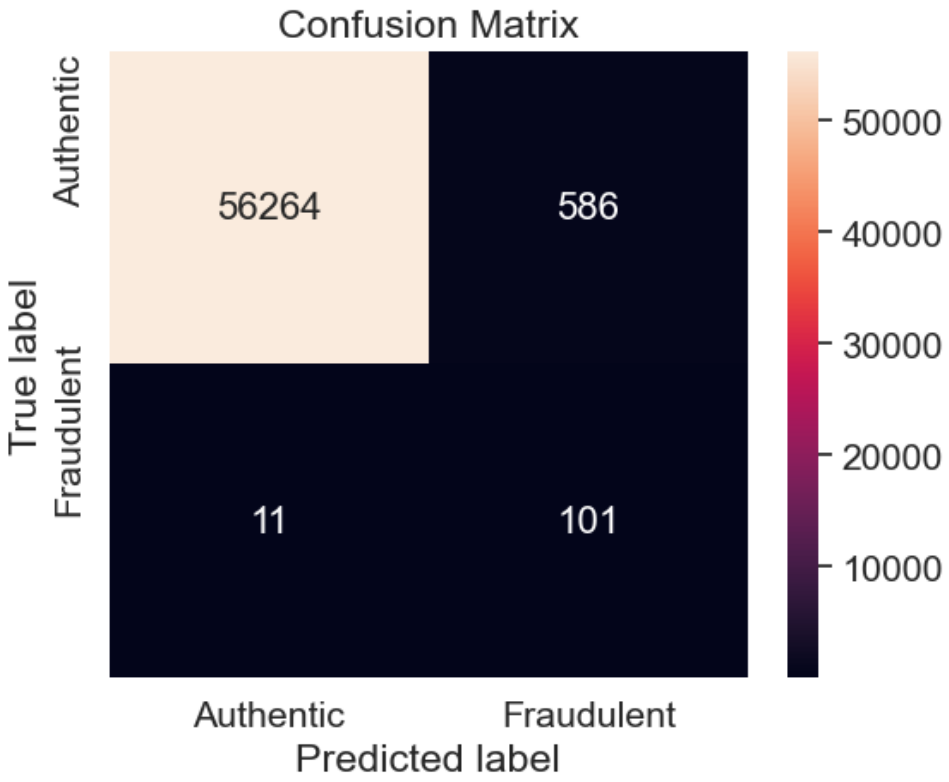
Out[ ]:

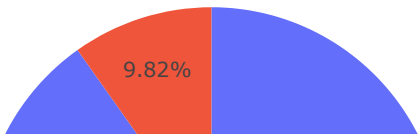
Metric	MCC	F1-Score	F2-Score	Recall	Precision	FM index	Specificity	G-mean	F0.5-Score	A
Performance score	0.856861	0.857143	0.857143	0.857143	0.857143	0.857143	0.999719	0.92569	0.857143	

Random under-sampling

In [ ]:
```
# Elementos da matriz de confusão

classification(svm_linear, X_train_under_scaled_minmax, y_train_under, X_test_scaled_minmax, y_t
```

```
# Resumo das métricas de avaliação

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
```



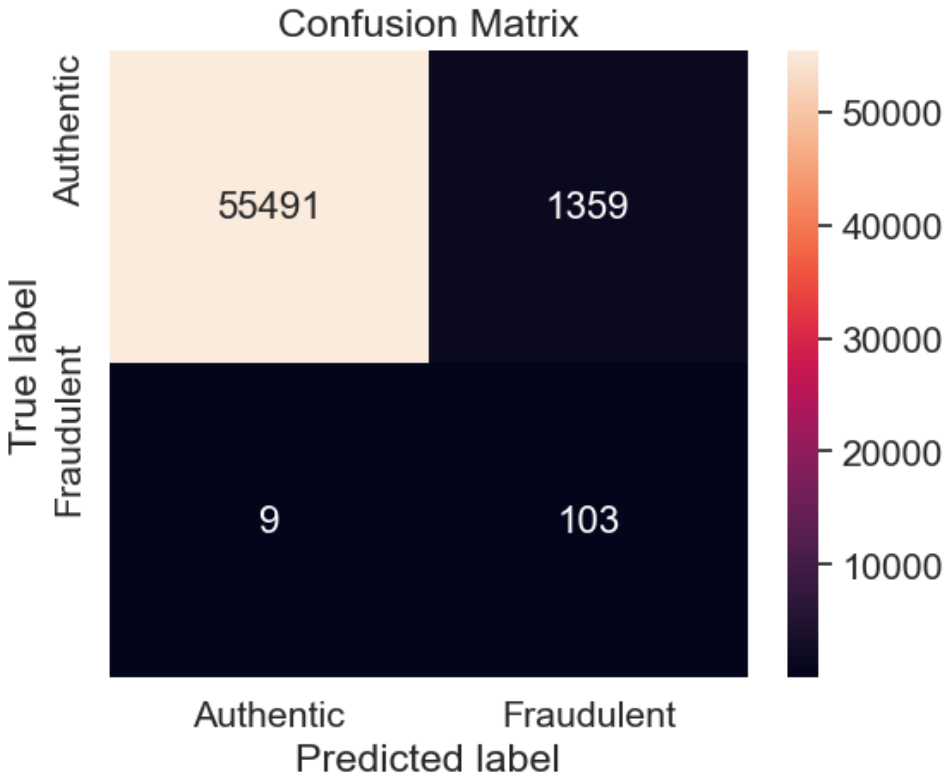


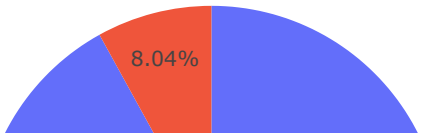
Out[ ]:

Metric	MCC	F1-Score	F2-Score	Recall	Precision	FM index	Specificity	G-mean	F0.5-Score
Performance score	0.361783	0.252816	0.444934	0.901786	0.147016	0.364111	0.989692	0.944717	0.176573

Random over-sampling

In [ ]:
```
# Elementos da matriz de confusão

classification(svm_linear, X_train_over_scaled_minmax, y_train_over, X_test_scaled_minmax, y_tes

# Resumo das métricas de avaliação

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
```



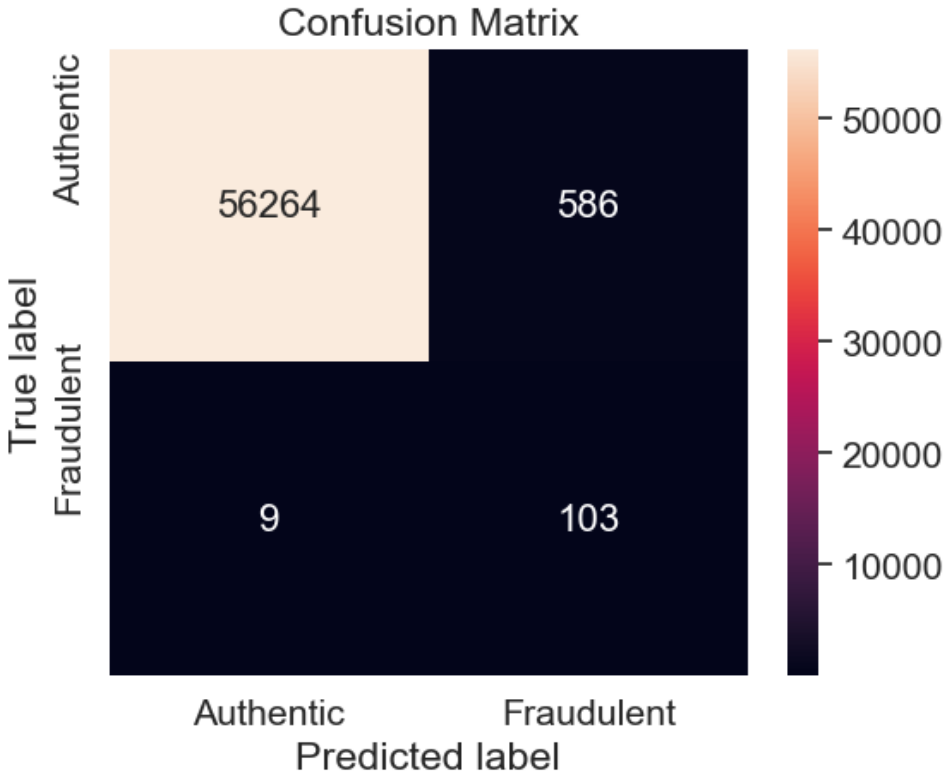


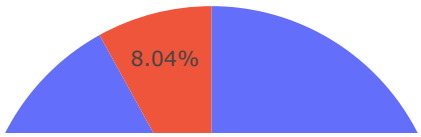
Out[ ]:

Metric	MCC	F1-Score	F2-Score	Recall	Precision	FM index	Specificity	G-mean	F0.5-Score	A
Performance score	0.25092	0.130877	0.269634	0.919643	0.070451	0.254539	0.976095	0.947449	0.086409	

Random under-sampling with imbalanced-learning library

In [ ]:
```
# Elementos da matriz de confusão

classification(svm_linear, X_train_under_imblearn_scaled_minmax, y_train_under_imblearn, X_test_

# Resumo das métricas de avaliação

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], i
summary_svm_linear_under_imblearn_index
```





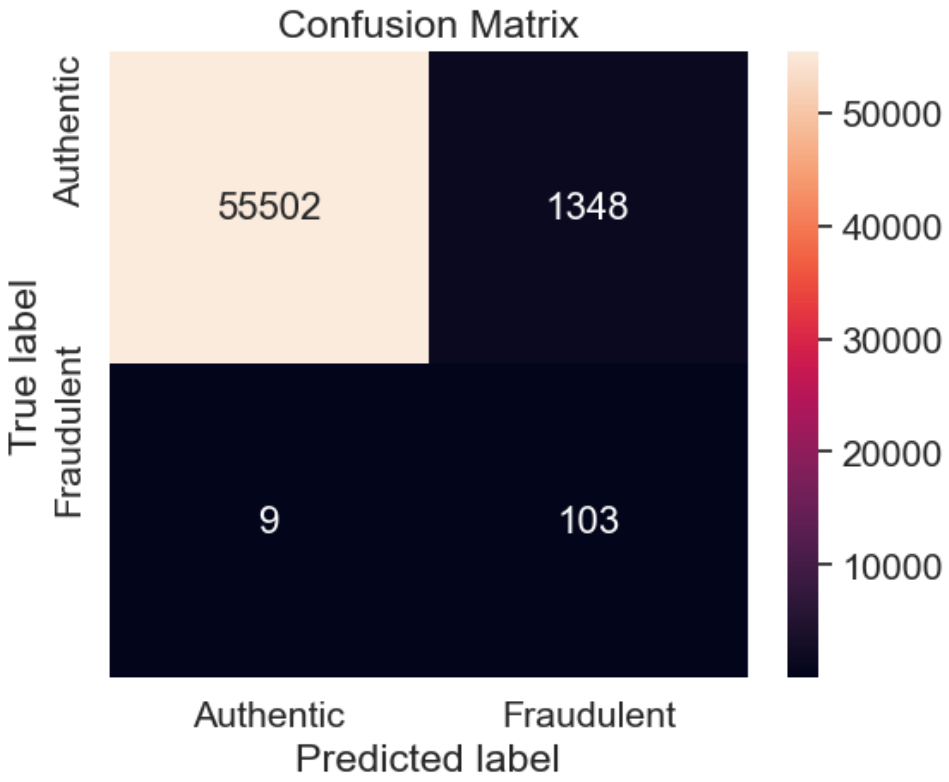
Out[ ]:

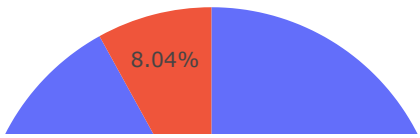
	Metric	MCC	F1-Score	F2-Score	Recall	Precision	FM index	Specificity	G-mean	F0.5-Score
Performance score		0.368501	0.257179	0.452946	0.919643	0.149492	0.370782	0.989692	0.954025	0.179568

Random over-sampling with imbalanced-learning library

In [ ]:
```
# Elementos da matriz de confusão

classification(svm_linear, X_train_over_imblearn_scaled_minmax, y_train_over_imblearn, X_test_sc
```

```
# Resumo das métricas de avaliação

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], inp
summary_svm_linear_over_imblearn_index
```



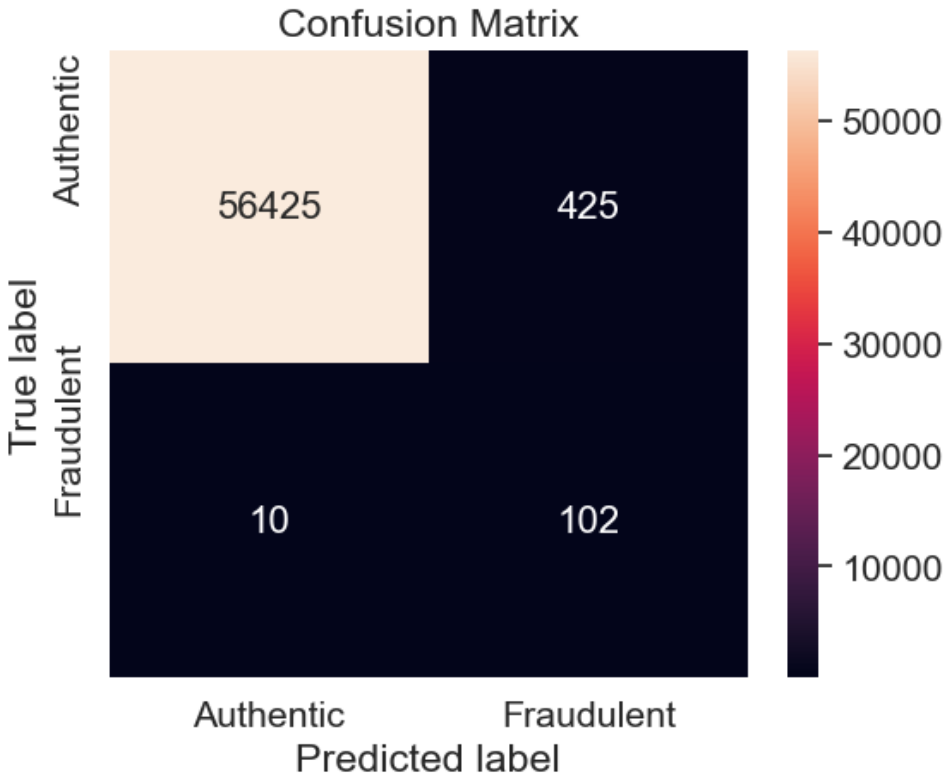


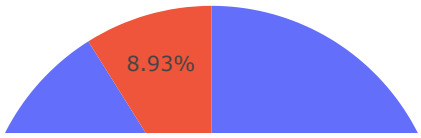
Out[ ]:

Metric	MCC	F1-Score	F2-Score	Recall	Precision	FM index	Specificity	G-mean	F0.5-Score
Performance score	0.251899	0.131798	0.271195	0.919643	0.070986	0.255502	0.976288	0.947542	0.087052

Synthetic minority over-sampling technique (SMOTE)

In [ ]:
```
# Elementos da matriz de confusão

classification(svm_linear, X_train_over_smote_scaled_minmax, y_train_over_smote, X_test_scaled_m

# Resumo das métricas de avaliação

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 =
summary_svm_linear_over_smote_index
```



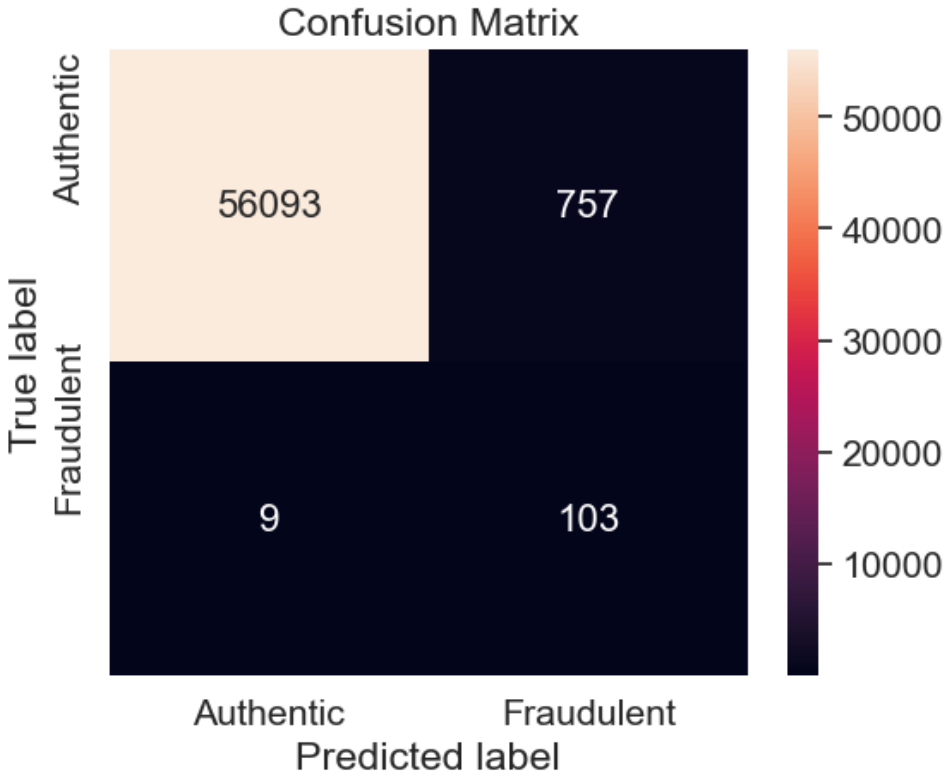


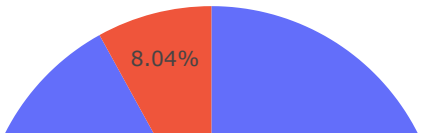
Out[ ]:

Metric	MCC	F1-Score	F2-Score	Recall	Precision	FM index	Specificity	G-mean	F0.5-Score	A
Performance score	0.417924	0.319249	0.523077	0.910714	0.193548	0.419842	0.992524	0.95074	0.22973	0

Under-sampling via NearMiss

In [ ]:
```
# Elementos da matriz de confusão

classification(svm_linear, X_train_under_nm_scaled_minmax, y_train_under_nm, X_test_scaled_minma

# Resumo das métricas de avaliação

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 = Tru
summary_svm_linear_under_nm_index
```





Out[ ]:

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

Summary of linear SVM classification models

In [ ]:
```
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_l
```

```
                          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
```

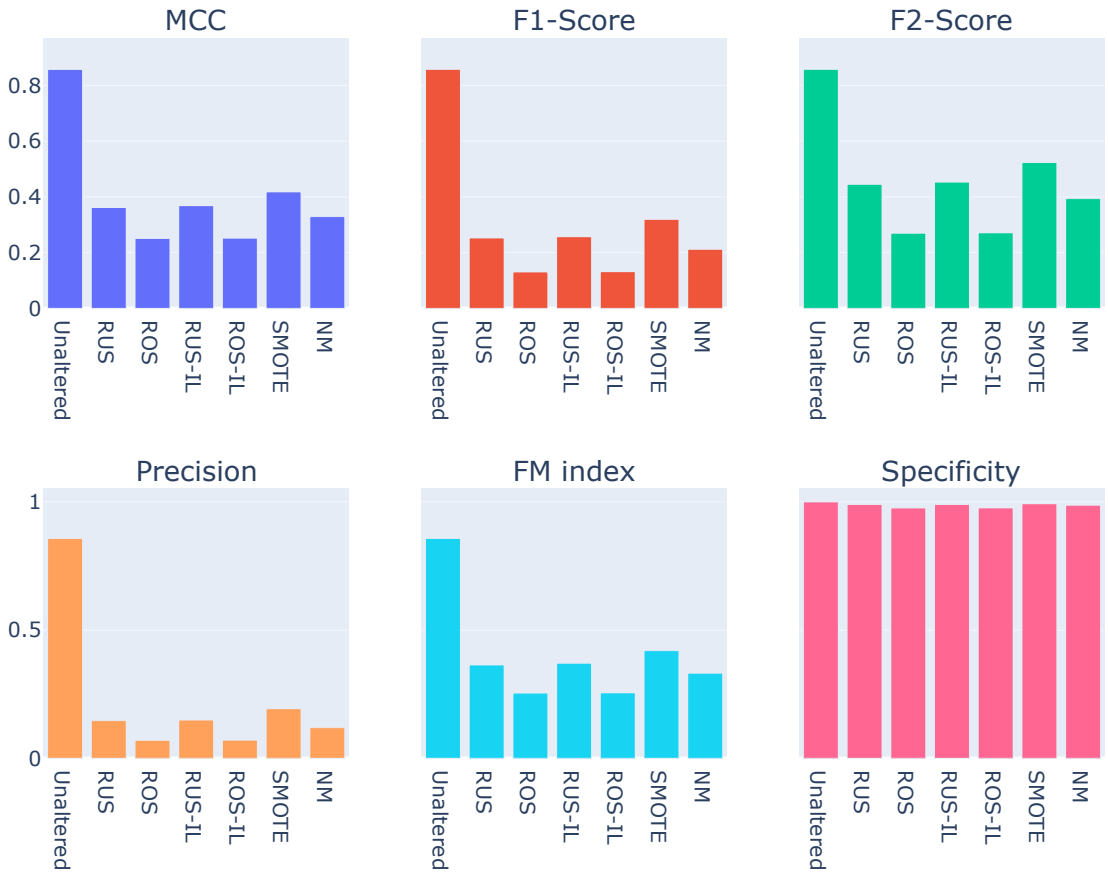
```
C:\Users\Diones\AppData\Local\Temp\ipykernel_14028\3997214041.py:9: FutureWarning:

Passing 'suffixes' which cause duplicate columns {'Performance score_x'} in the result is depreca
ted and will raise a MergeError in a future version.

C:\Users\Diones\AppData\Local\Temp\ipykernel_14028\3997214041.py:9: FutureWarning:

Passing 'suffixes' which cause duplicate columns {'Performance score_x'} in the result is depreca
ted and will raise a MergeError in a future version.
```

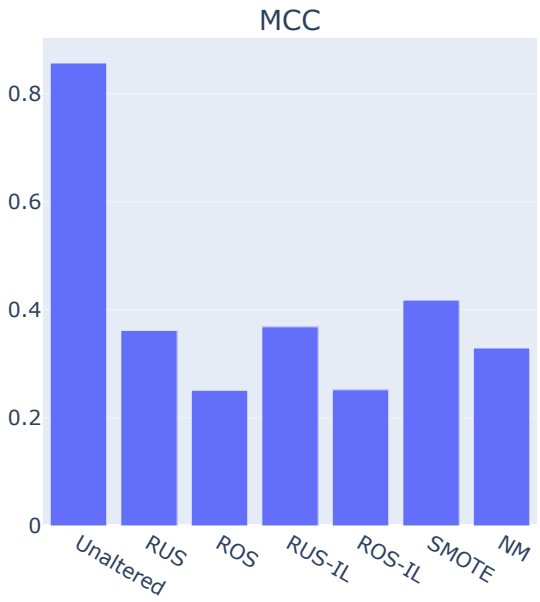
Out[ ]:

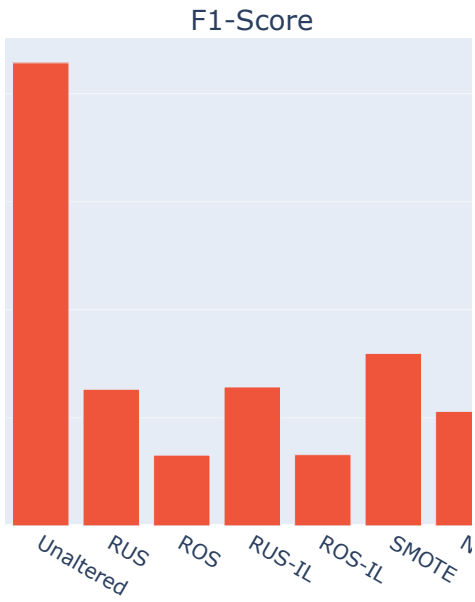
Metric	Unaltered	RUS	ROS	RUS-IL	ROS-IL	SMOTE	NM
MCC	0.856861	0.361783	0.250920	0.368501	0.251899	0.417924	0.329246
F1-Score	0.857143	0.252816	0.130877	0.257179	0.131798	0.319249	0.211934
F2-Score	0.857143	0.444934	0.269634	0.452946	0.271195	0.523077	0.393731
Recall	0.857143	0.901786	0.919643	0.919643	0.919643	0.910714	0.919643
Precision	0.857143	0.147016	0.070451	0.149492	0.070986	0.193548	0.119767
FM index	0.857143	0.364111	0.254539	0.370782	0.255502	0.419842	0.331878
Specificity	0.999719	0.989692	0.976095	0.989692	0.976288	0.992524	0.986684
G-mean	0.925690	0.944717	0.947449	0.954025	0.947542	0.950740	0.952574
F0.5-Score	0.857143	0.176573	0.086409	0.179568	0.087052	0.229730	0.144989
Accuracy	0.999438	0.989519	0.975984	0.989554	0.976177	0.992363	0.986552

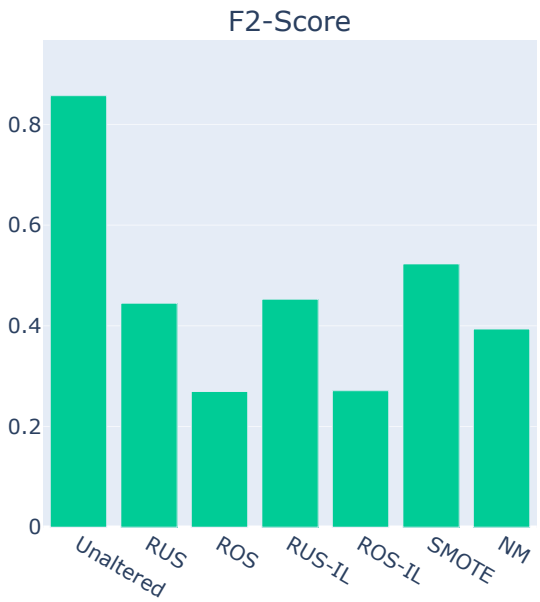
In [ ]:

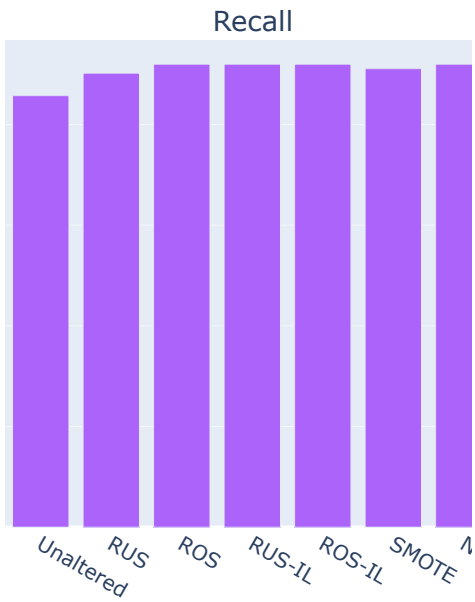
```
# Comparação visual do modelo aplicado em diferentes conjuntos de treinamento por meio de várias

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
fig1.add_trace(go.Bar(x = list(summary_svm_linear.columns), y = list(summary_svm_linear.loc['F1-
fig1.add_trace(go.Bar(x = list(summary_svm_linear.columns), y = list(summary_svm_linear.loc['F2-
fig1.add_trace(go.Bar(x = list(summary_svm_linear.columns), y = list(summary_svm_linear.loc['Rec
fig1.add_trace(go.Bar(x = list(summary_svm_linear.columns), y = list(summary_svm_linear.loc['Pre
fig1.add_trace(go.Bar(x = list(summary_svm_linear.columns), y = list(summary_svm_linear.loc['FM
fig1.add_trace(go.Bar(x = list(summary_svm_linear.columns), y = list(summary_svm_linear.loc['Acc
fig1.add_trace(go.Bar(x = list(summary_svm_linear.columns), y = list(summary_svm_linear.loc['Spe

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

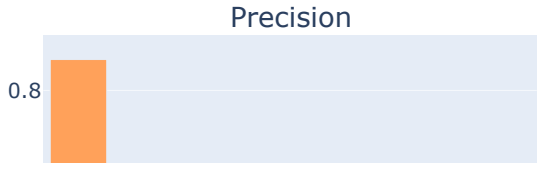

MCC
0.8
0.6
0.4
0.2
0
Unaltered
RUS
ROS
RUS-IL
ROS-IL
SMOTE
NM
F1-Score
Unaltered
RUS
ROS
RUS-IL
ROS-IL
SMOTE
NM
F2-Score
Unaltered
RUS
ROS
RUS-IL
ROS-IL
SMOTE
NM
Precision
1
0.5
0
Unaltered
RUS
ROS
RUS-IL
ROS-IL
SMOTE
NM
FM index
Unaltered
RUS
ROS
RUS-IL
ROS-IL
SMOTE
NM
Specificity
Unaltered
RUS
ROS
RUS-IL
ROS-IL
SMOTE
NM

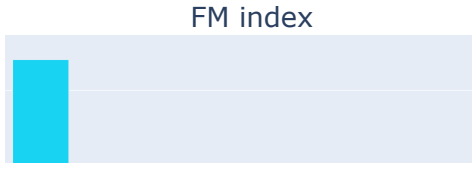

MCC
0.8
0.6
0.4
0.2
0
Unaltered
RUS
ROS
RUS-IL
ROS-IL
SMOTE
NM

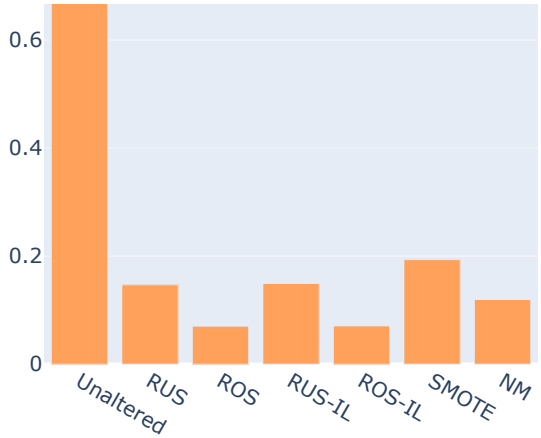

F1-Score
Unaltered
RUS
ROS
RUS-IL
ROS-IL
SMOTE

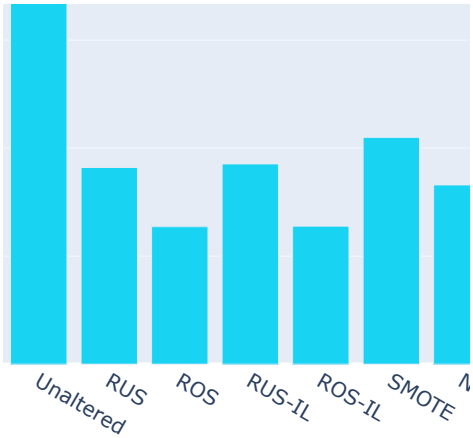

F2-Score
0.8
0.6
0.4
0.2
0
Unaltered
RUS
ROS
RUS-IL
ROS-IL
SMOTE
NM

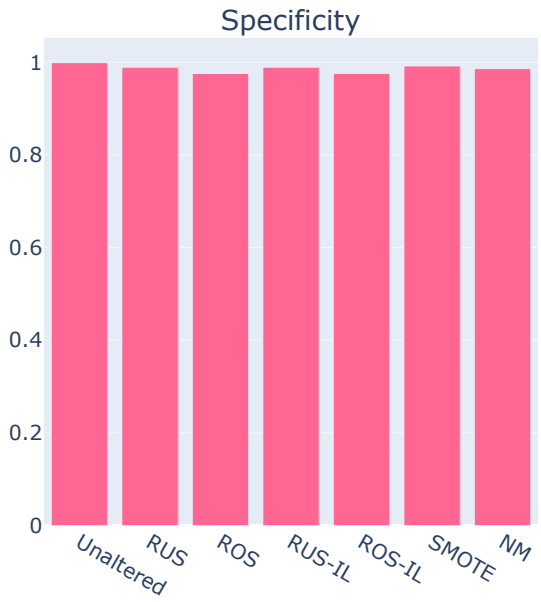

Recall
Unaltered
RUS
ROS
RUS-IL
ROS-IL
SMOTE

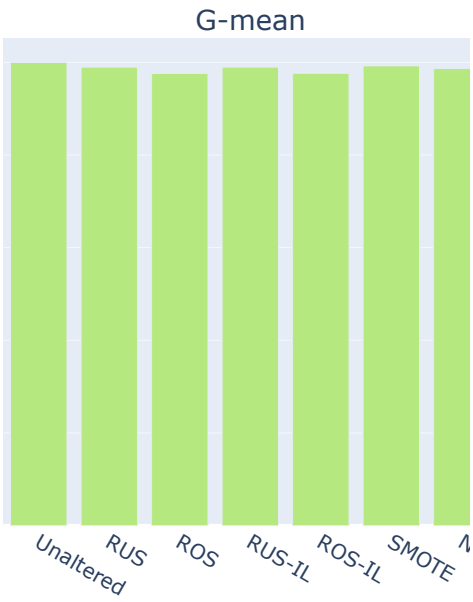

Precision
0.8


FM index









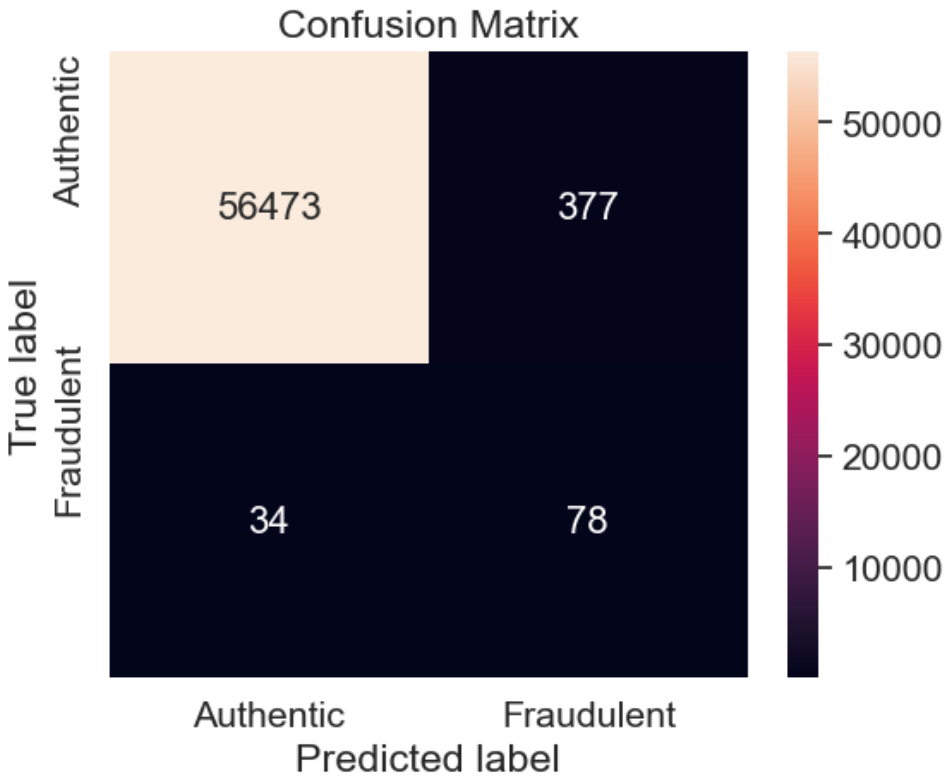
9. Naive Bayes

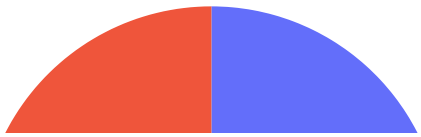
In [ ]:
```
nb = GaussianNB()
```

Unaltered training set

In [ ]:
```
# Elementos da matriz de confusão

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

# Resumo das métricas de avaliação

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_au
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
```



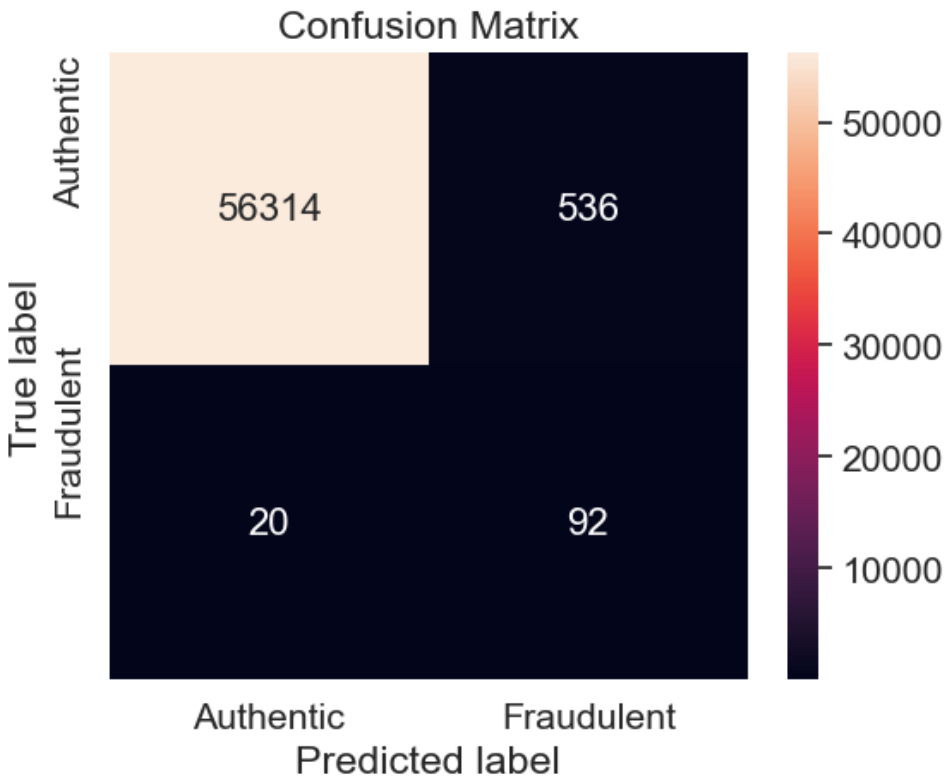


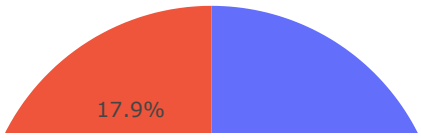
Out[ ]:

Metric	MCC	F1-Score	F2-Score	Recall	Precision	FM index	Specificity	G-mean	F0.5-Score
Performance score	0.343272	0.275132	0.431894	0.696429	0.171429	0.345525	0.993369	0.831751	0.201863

Random under-sampling

In [ ]:
```
# Elementos da matriz de confusão

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

# Resumo das métricas de avaliação

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
```





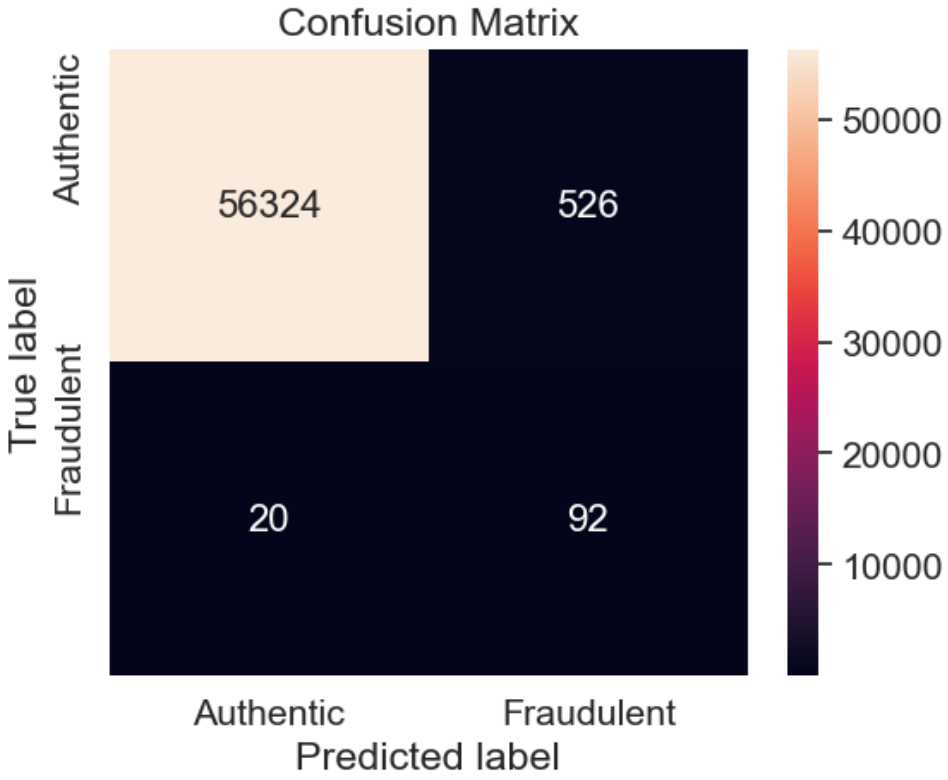
Out[ ]:

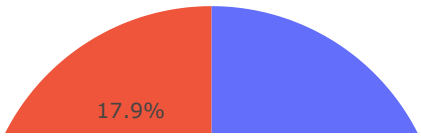
Metric	MCC	F1-Score	F2-Score	Recall	Precision	FM index	Specificity	G-mean	F0.5-Score
Performance score	0.344481	0.248649	0.427509	0.821429	0.146497	0.346896	0.990572	0.902044	0.175305

Random over-sampling

In [ ]:

```
# Elementos da matriz de confusão

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

# Resumo das métricas de avaliação

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
```





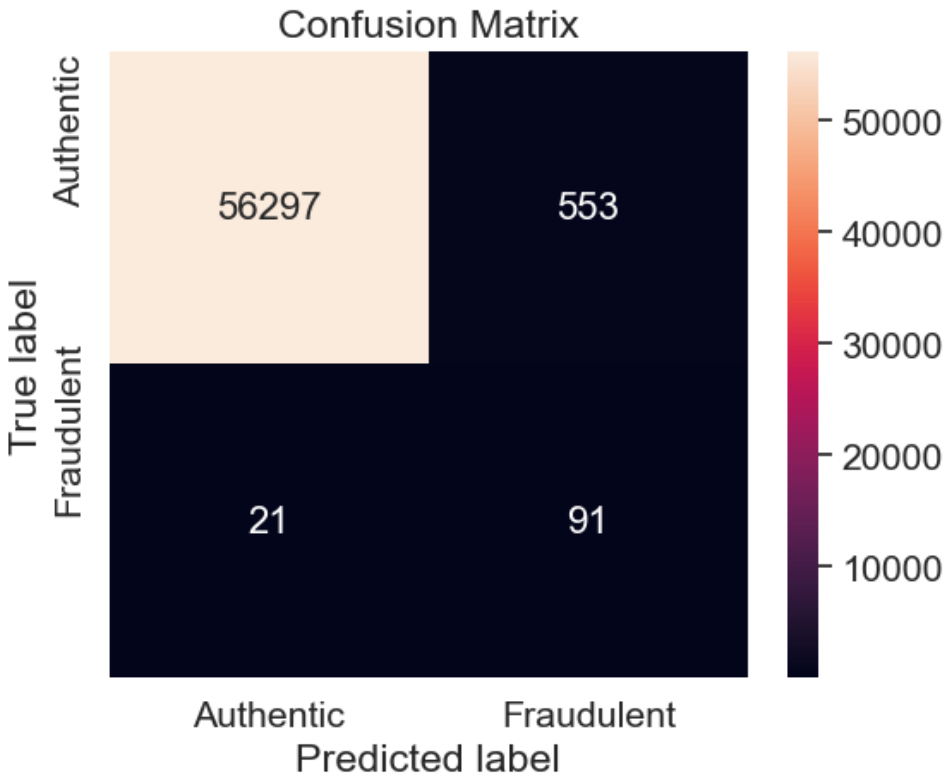
Out[ ]:

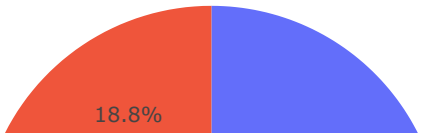
	Metric	MCC	F1-Score	F2-Score	Recall	Precision	FM index	Specificity	G-mean	F0.5-Score	A
Performance score		0.347301	0.252055	0.43152	0.821429	0.148867	0.349691	0.990748	0.902124	0.178019	

Random under-sampling with imbalanced-learn library

In [ ]:
```
# Elementos da matriz de confusão

classification(nb, X_train_under_imblearn, y_train_under_imblearn, X_test, y_test)
```

```
# Resumo das métricas de avaliação

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-AU
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
```



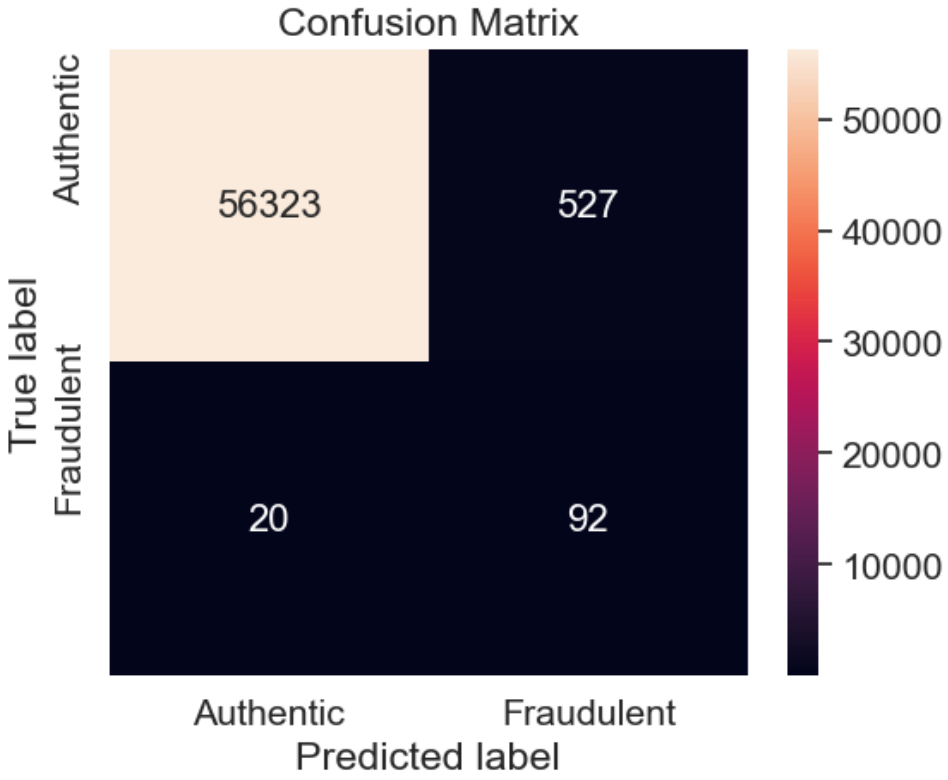


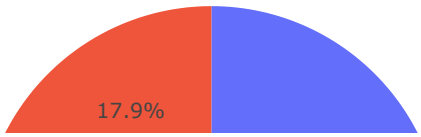
Out[ ]:

Metric	MCC	F1-Score	F2-Score	Recall	Precision	FM index	Specificity	G-mean	F0.5-Score	A
Performance score	0.336357	0.240741	0.416667	0.8125	0.141304	0.338836	0.990273	0.896993	0.169271	0

Random over-sampling with imbalanced-learn library

In [ ]:
```
# Elementos da matriz de confusão

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

# Resumo das métricas de avaliação

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'
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
```





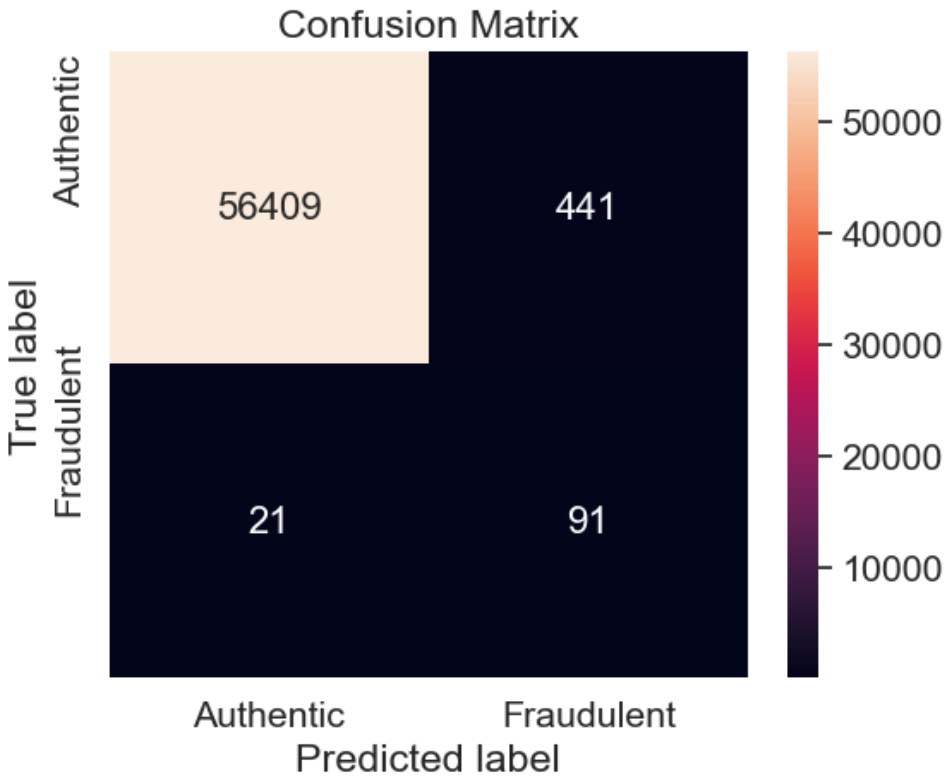
Out[ ]:

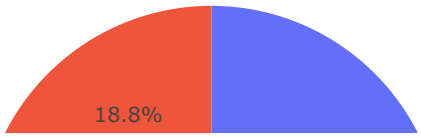
	Metric	MCC	F1-Score	F2-Score	Recall	Precision	FM index	Specificity	G-mean	F0.5-Score	A
Performance score		0.347016	0.25171	0.431115	0.821429	0.148627	0.349409	0.99073	0.902116	0.177743	

Synthetic minority over-sampling technique (SMOTE)

In [ ]:
```
# Elementos da matriz de confusão

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

```
# Resumo das métricas de avaliação

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_
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
```



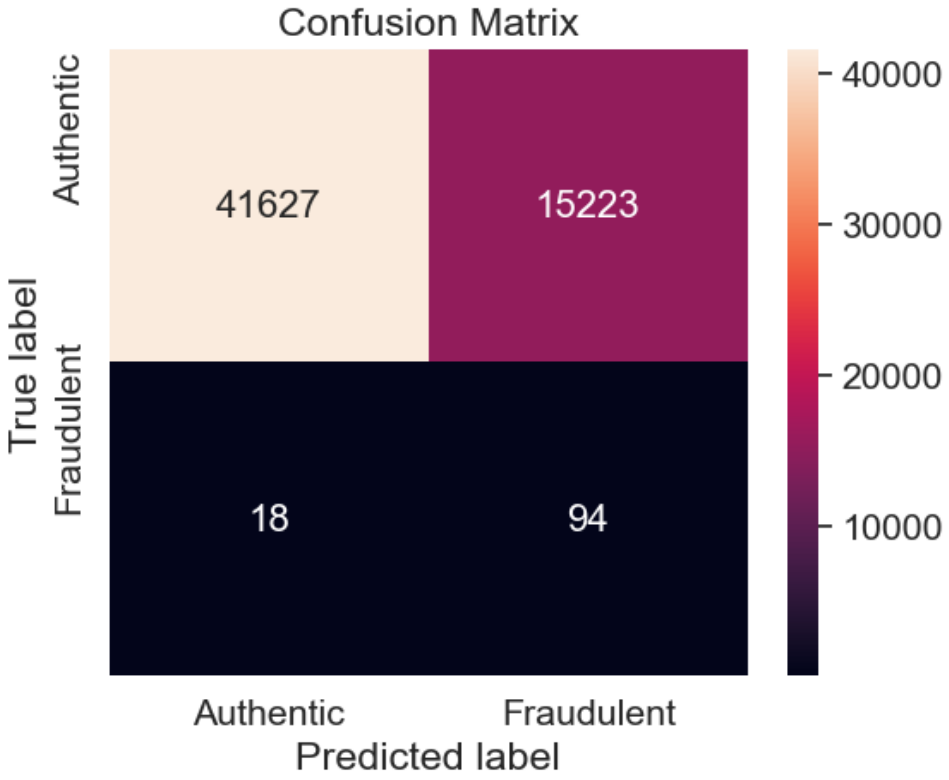


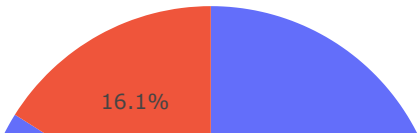
Out[ ]:

Metric	MCC	F1-Score	F2-Score	Recall	Precision	FM index	Specificity	G-mean	F0.5-Score	A
Performance score	0.370613	0.282609	0.464286	0.8125	0.171053	0.372801	0.992243	0.897885	0.203125	0

Under-sampling via NearMiss

In [ ]:
```
# Elementos da matriz de confusão

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

# Resumo das métricas de avaliação

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
```





Out[ ]:

	Metric	MCC	F1-Score	F2-Score	Recall	Precision	FM index	Specificity	G-mean	F0.5-Score
Performance score		0.057099	0.012185	0.029813	0.839286	0.006137	0.071768	0.732225	0.78393	0.007657

Summary of naive Bayes models

In [ ]:
```
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_imb
```

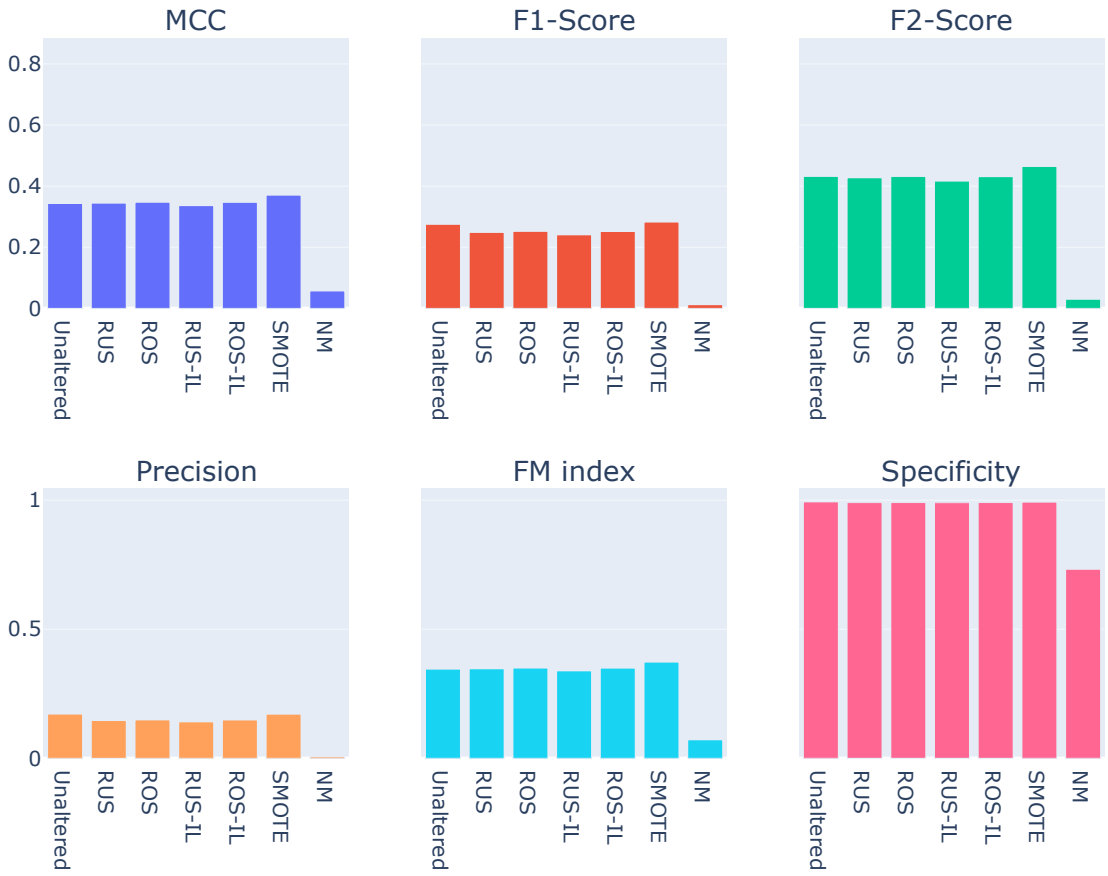
```
                        summary_nb_over_imblearn, summary_nb_over_smote, summary_nb_under_nm]

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
```

```
C:\Users\Diones\AppData\Local\Temp\ipykernel_14028\3110425635.py:8: FutureWarning:

Passing 'suffixes' which cause duplicate columns {'Performance score_x'} in the result is depreca
ted and will raise a MergeError in a future version.

C:\Users\Diones\AppData\Local\Temp\ipykernel_14028\3110425635.py:8: FutureWarning:

Passing 'suffixes' which cause duplicate columns {'Performance score_x'} in the result is depreca
ted and will raise a MergeError in a future version.
```

Out[ ]:

Metric	Unaltered	RUS	ROS	RUS-IL	ROS-IL	SMOTE	NM
MCC	0.343272	0.344481	0.347301	0.336357	0.347016	0.370613	0.057099
F1-Score	0.275132	0.248649	0.252055	0.240741	0.251710	0.282609	0.012185
F2-Score	0.431894	0.427509	0.431520	0.416667	0.431115	0.464286	0.029813
Recall	0.696429	0.821429	0.821429	0.812500	0.821429	0.812500	0.839286
Precision	0.171429	0.146497	0.148867	0.141304	0.148627	0.171053	0.006137
FM index	0.345525	0.346896	0.349691	0.338836	0.349409	0.372801	0.071768
Specificity	0.993369	0.990572	0.990748	0.990273	0.990730	0.992243	0.732225
G-mean	0.831751	0.902044	0.902124	0.896993	0.902116	0.897885	0.783930
F0.5-Score	0.201863	0.175305	0.178019	0.169271	0.177743	0.203125	0.007657
Accuracy	0.992785	0.990239	0.990415	0.989923	0.990397	0.991889	0.732436

In [ ]:
```
# Comparação visual do modelo aplicado em diferentes conjuntos de treinamento por meio de várias

summary_visual(summary_nb)
```



10. Random Forest

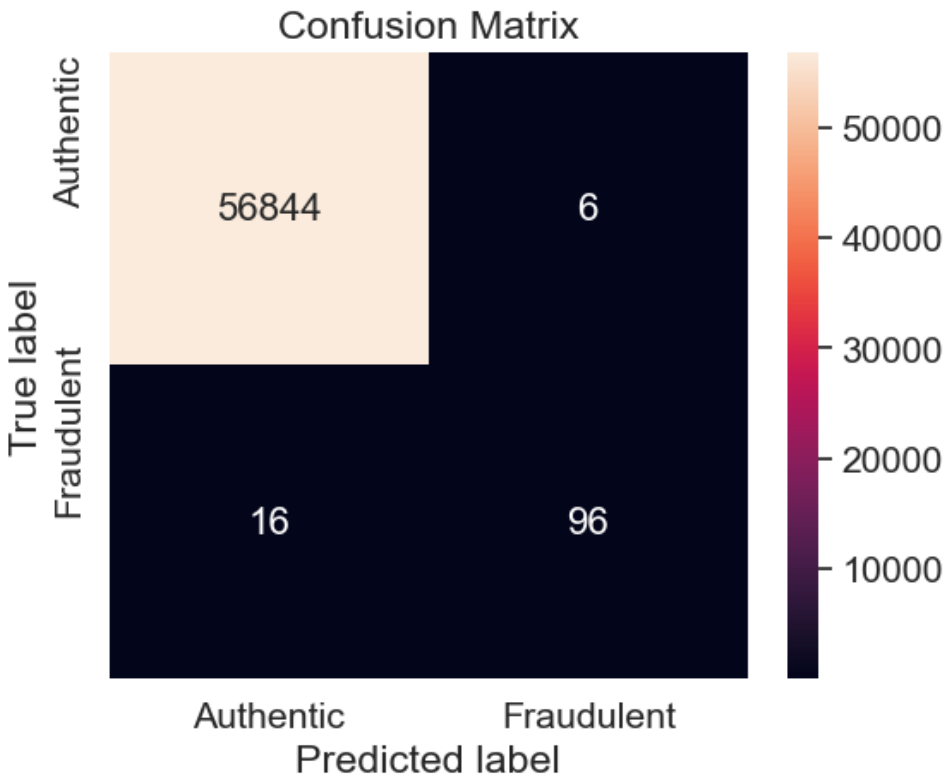
The Random Forest classifier employs multiple decision trees, thereby avoiding the reliance upon feature selection of a singular decision tree.

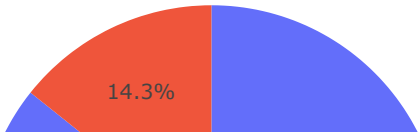
In [ ]:
```
rf = RandomForestClassifier(n_estimators = 100)
```

Unaltered training set

In [ ]:
```
# Elementos da matriz de confusão

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

# Resumo das métricas de avaliação

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_au
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
```



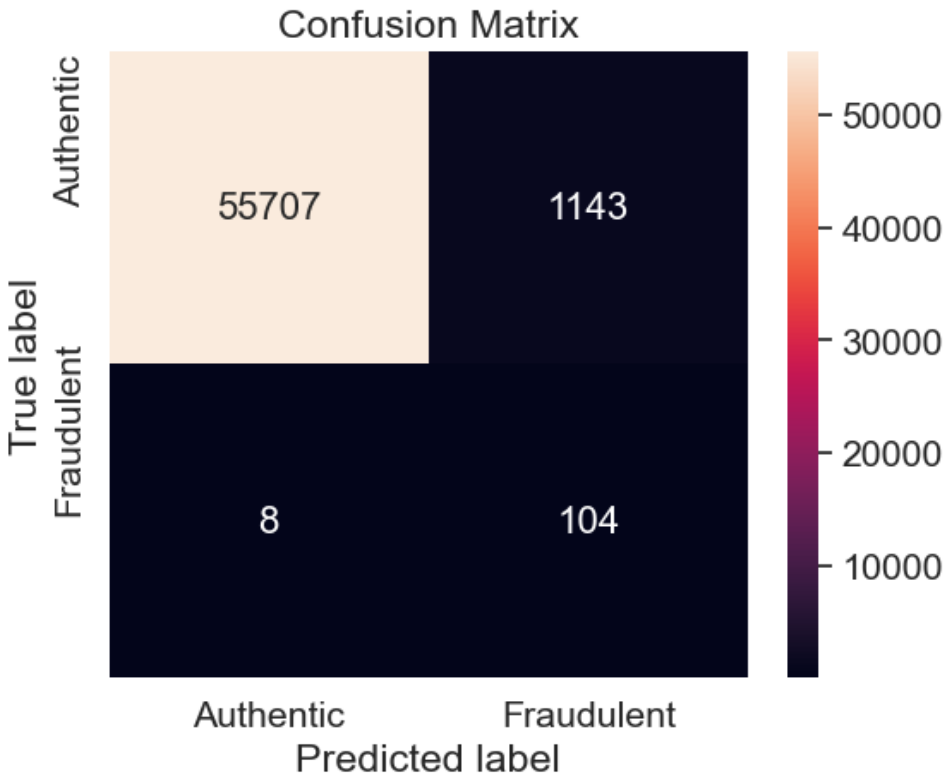


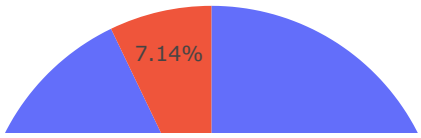
Out[ ]:

Metric	MCC	F1-Score	F2-Score	Recall	Precision	FM index	Specificity	G-mean	F0.5-Score
Performance score	0.897988	0.897196	0.872727	0.857143	0.941176	0.898177	0.999894	0.925771	0.923077

Random under-sampling

In [ ]:
```
# Elementos da matriz de confusão

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

# Resumo das métricas de avaliação

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
```



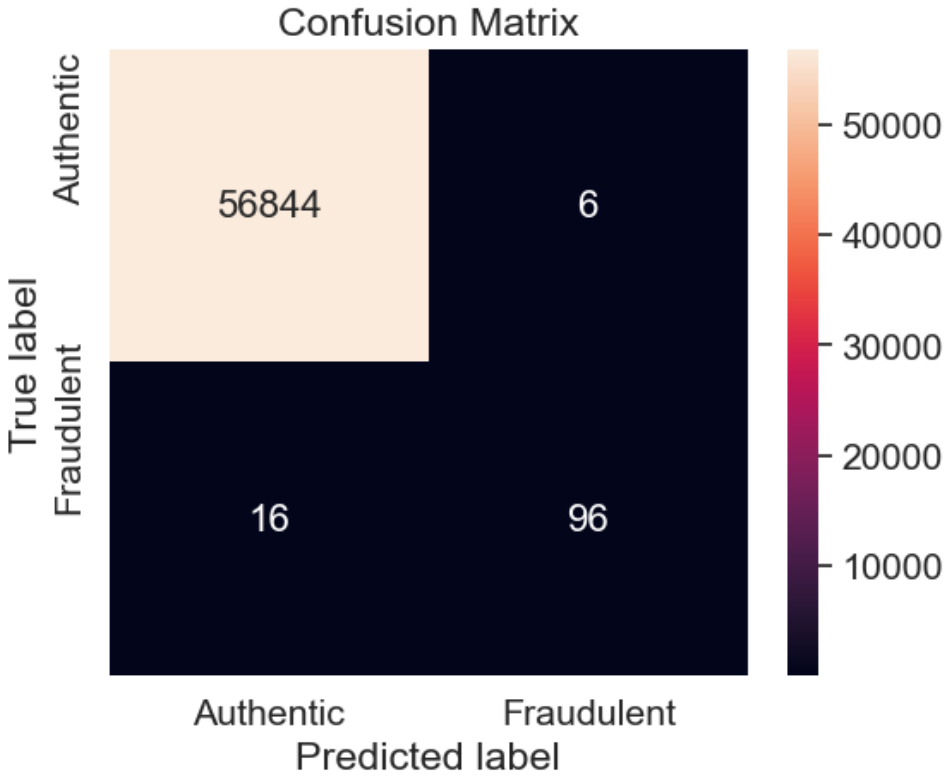


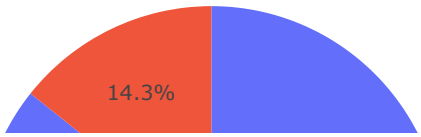
Out[ ]:

Metric	MCC	F1-Score	F2-Score	Recall	Precision	FM index	Specificity	G-mean	F0.5-Score	A
Performance score	0.27502	0.153054	0.306785	0.928571	0.0834	0.278286	0.979894	0.953888	0.101961	

Random over-sampling

In [ ]:
```
# Elementos da matriz de confusão

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

# Resumo das métricas de avaliação

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
```





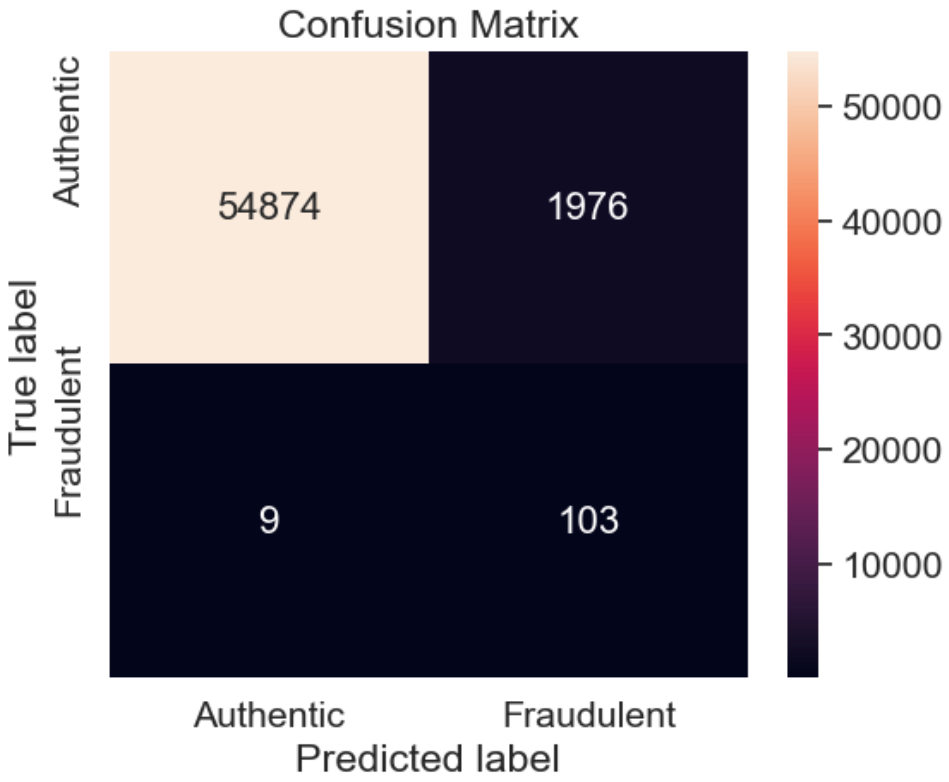
Out[ ]:

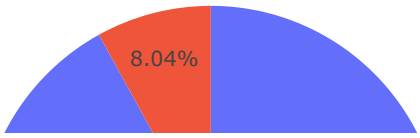
Metric	MCC	F1-Score	F2-Score	Recall	Precision	FM index	Specificity	G-mean	F0.5-Score
Performance score	0.897988	0.897196	0.872727	0.857143	0.941176	0.898177	0.999894	0.925771	0.923077

Random under-sampling with imbalanced-learn library

In [ ]:
```
# Elementos da matriz de confusão

classification(rf, X_train_under_imblearn, y_train_under_imblearn, X_test, y_test)
```

```
# Resumo das métricas de avaliação

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-AU
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
```





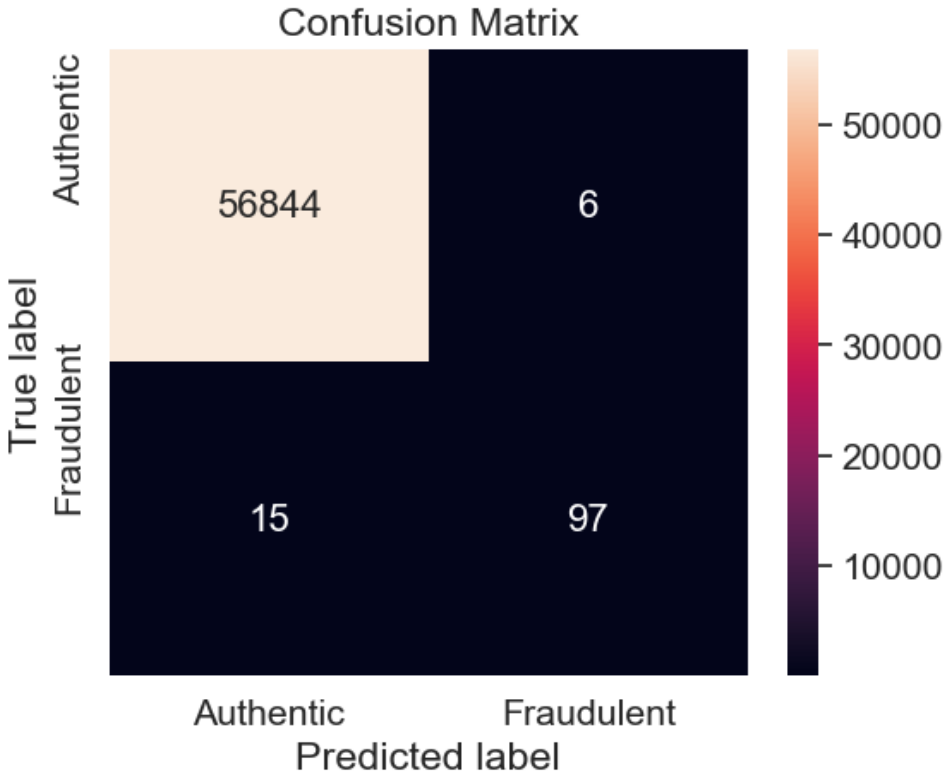
Out[ ]:

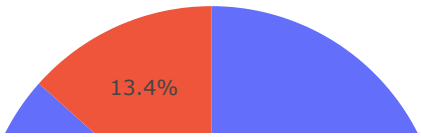
Metric	MCC	F1-Score	F2-Score	Recall	Precision	FM index	Specificity	G-mean	F0.5-Score
Performance score	0.209033	0.094021	0.203799	0.919643	0.049543	0.213452	0.965242	0.942167	0.061106

Random over-sampling with imbalanced-learn library

In [ ]:

```
# Elementos da matriz de confusão

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

# Resumo das métricas de avaliação

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'
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
```





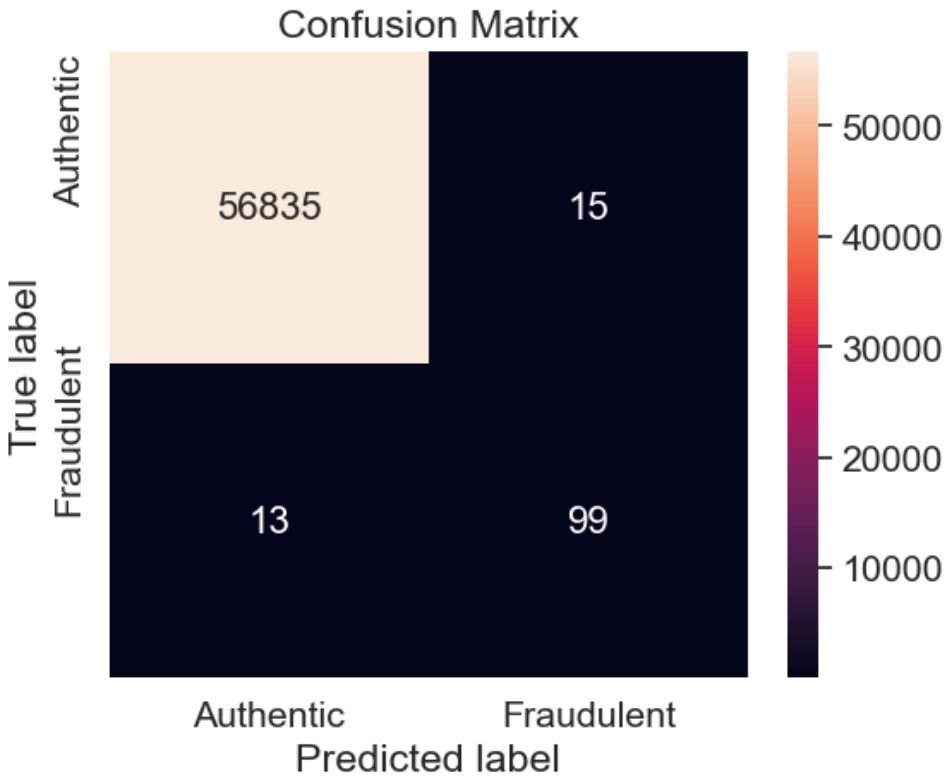
Out[ ]:

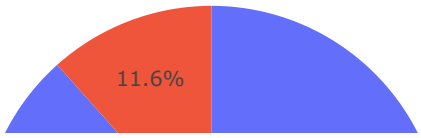
	Metric	MCC	F1-Score	F2-Score	Recall	Precision	FM index	Specificity	G-mean	F0.5-Score
Performance score		0.902936	0.902326	0.880218	0.866071	0.941748	0.903117	0.999894	0.93058	0.925573

Synthetic minority over-sampling technique (SMOTE)

In [ ]:
```
# Elementos da matriz de confusão

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

```
# Resumo das métricas de avaliação

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_
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
```





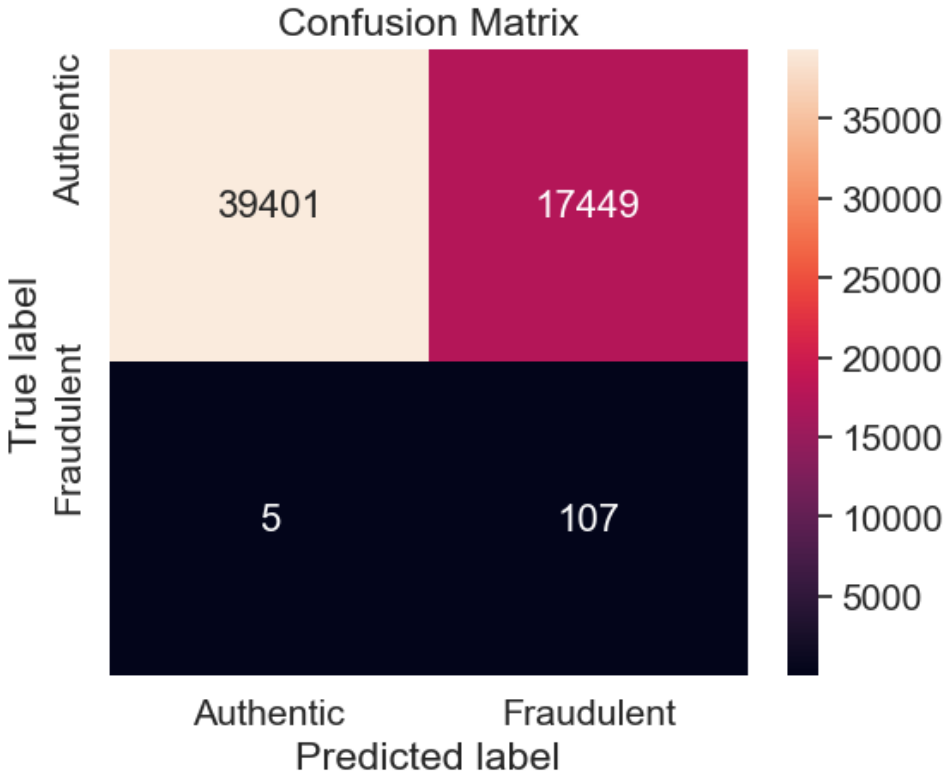
Out[ ]:

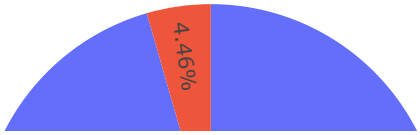
Metric	MCC	F1-Score	F2-Score	Recall	Precision	FM index	Specificity	G-mean	F0.5-Score
Performance score	0.875894	0.876106	0.880783	0.883929	0.868421	0.876141	0.999736	0.940051	0.871479

Under-sampling via NearMiss

In [ ]:

```
# Elementos da matriz de confusão

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

# Resumo das métricas de avaliação

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
```





Out[ ]:

Metric	MCC	F1-Score	F2-Score	Recall	Precision	FM index	Specificity	G-mean	F0.5-Score
Performance score	0.062207	0.012112	0.029716	0.955357	0.006095	0.076307	0.693069	0.813713	0.007606

Summary of random forest models

In [ ]:
```
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_imb
```

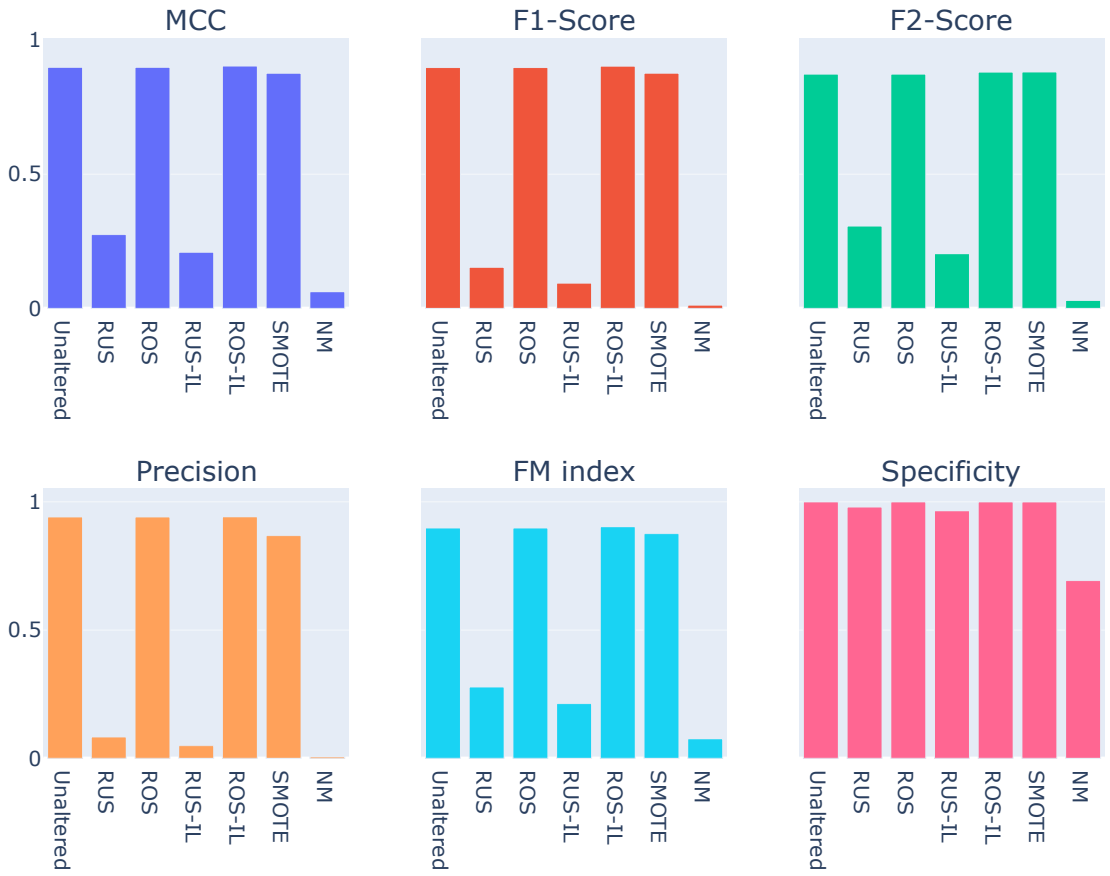
```
                    summary_rf_over_imblearn, summary_rf_over_smote, summary_rf_under_nm]

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
```

```
C:\Users\Diones\AppData\Local\Temp\ipykernel_14028\2240544619.py:8: FutureWarning:

Passing 'suffixes' which cause duplicate columns {'Performance score_x'} in the result is depreca
ted and will raise a MergeError in a future version.

C:\Users\Diones\AppData\Local\Temp\ipykernel_14028\2240544619.py:8: FutureWarning:

Passing 'suffixes' which cause duplicate columns {'Performance score_x'} in the result is depreca
ted and will raise a MergeError in a future version.
```

Out[ ]:

Metric	Unaltered	RUS	ROS	RUS-IL	ROS-IL	SMOTE	NM
MCC	0.897988	0.275020	0.897988	0.209033	0.902936	0.875894	0.062207
F1-Score	0.897196	0.153054	0.897196	0.094021	0.902326	0.876106	0.012112
F2-Score	0.872727	0.306785	0.872727	0.203799	0.880218	0.880783	0.029716
Recall	0.857143	0.928571	0.857143	0.919643	0.866071	0.883929	0.955357
Precision	0.941176	0.083400	0.941176	0.049543	0.941748	0.868421	0.006095
FM index	0.898177	0.278286	0.898177	0.213452	0.903117	0.876141	0.076307
Specificity	0.999894	0.979894	0.999894	0.965242	0.999894	0.999736	0.693069
G-mean	0.925771	0.953888	0.925771	0.942167	0.930580	0.940051	0.813713
F0.5-Score	0.923077	0.101961	0.923077	0.061106	0.925573	0.871479	0.007606
Accuracy	0.999614	0.979794	0.999614	0.965152	0.999631	0.999508	0.693585

In [ ]:
```
# Comparação visual do modelo aplicado em diferentes conjuntos de treinamento por meio de várias

summary_visual(summary_rf)
```



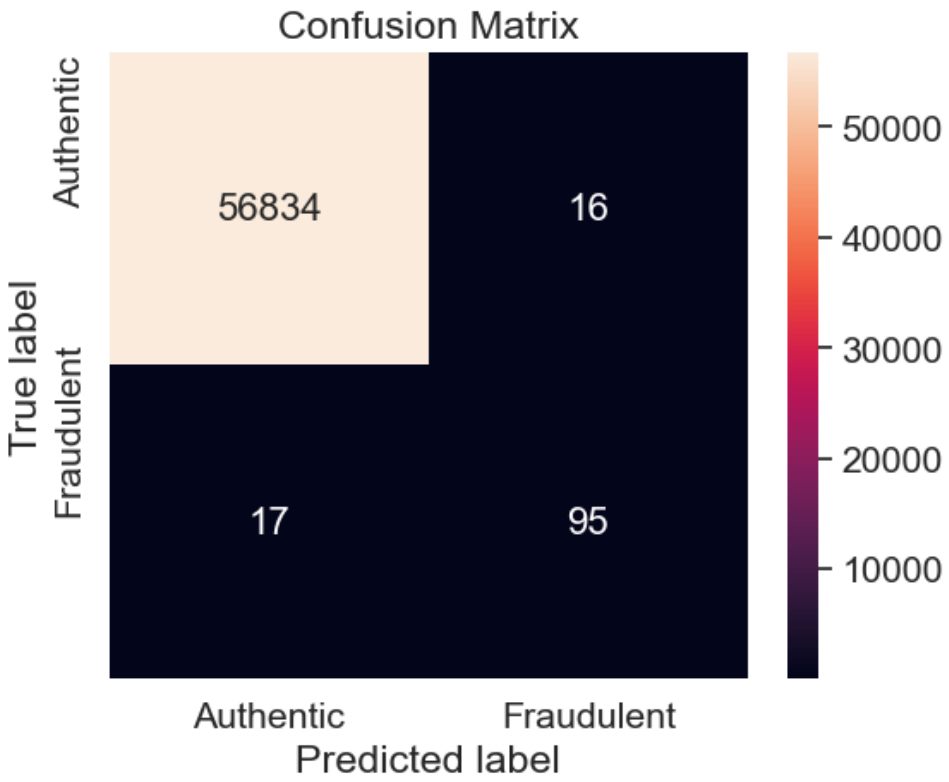
11. Linear discriminant analysis (LDA)

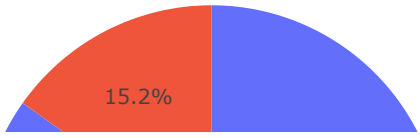
In [ ]:
```
lda = LinearDiscriminantAnalysis()
```

Unaltered training set

In [ ]:
```
# Elementos da matriz de confusão

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

# Resumo das métricas de avaliação

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_p
summary_lda_unaltered_extended.loc[len(summary_lda_unaltered_extended.index)] = ['ROC-AUC', roc_
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
```



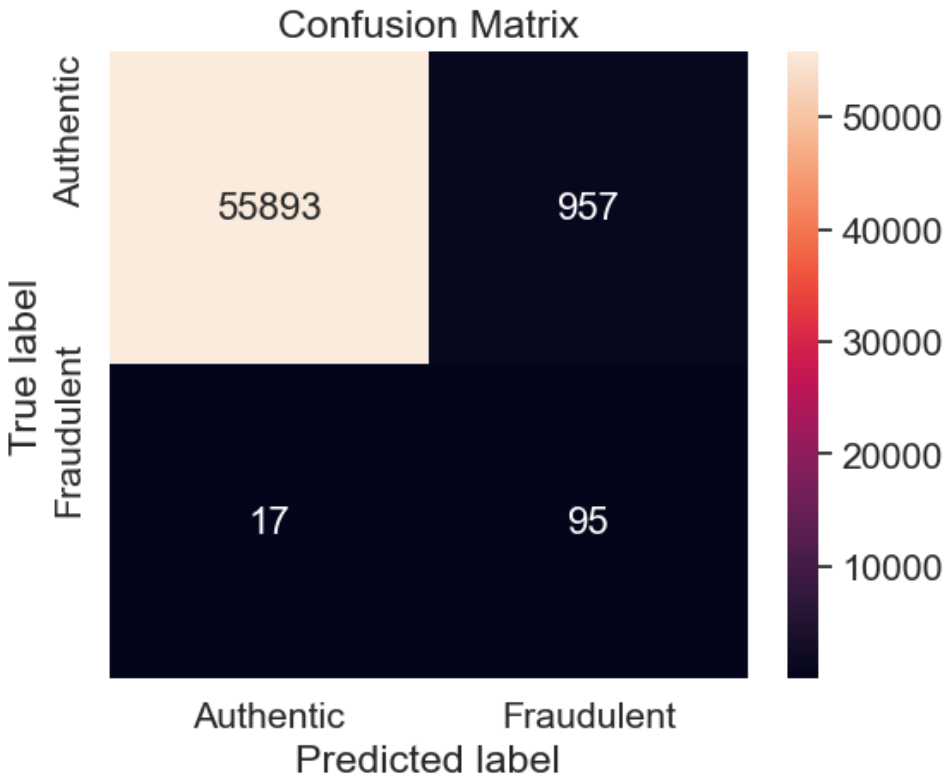


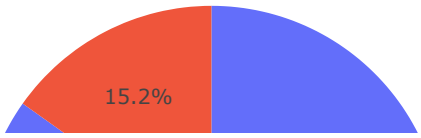
Out[ ]:

Metric	MCC	F1-Score	F2-Score	Recall	Precision	FM index	Specificity	G-mean	F0.5-Score
Performance score	0.851736	0.852018	0.849732	0.848214	0.855856	0.852027	0.999719	0.920856	0.854317

Random under-sampling

In [ ]:
```
# Elementos da matriz de confusão

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

# Resumo das métricas de avaliação

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
```



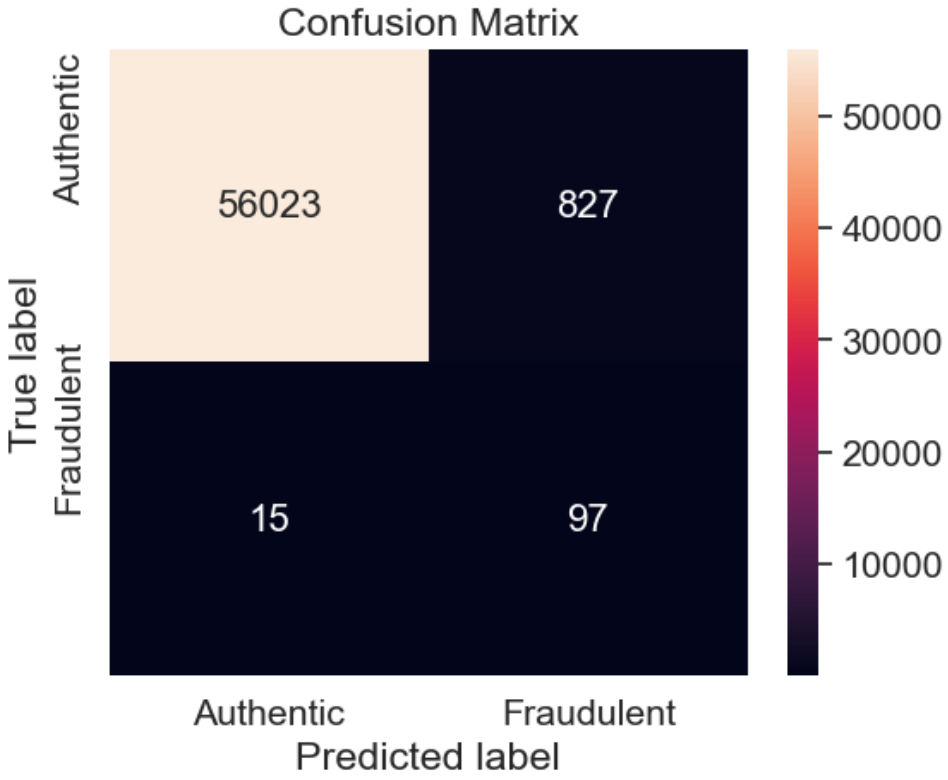


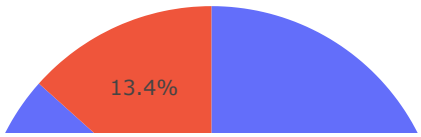
Out[ ]:

Metric	MCC	F1-Score	F2-Score	Recall	Precision	FM index	Specificity	G-mean	F0.5-Score	A
Performance score	0.27354	0.16323	0.316667	0.848214	0.090304	0.276762	0.983166	0.913201	0.109954	0

Random over-sampling

In [ ]:
```
# Elementos da matriz de confusão

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

# Resumo das métricas de avaliação

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
```





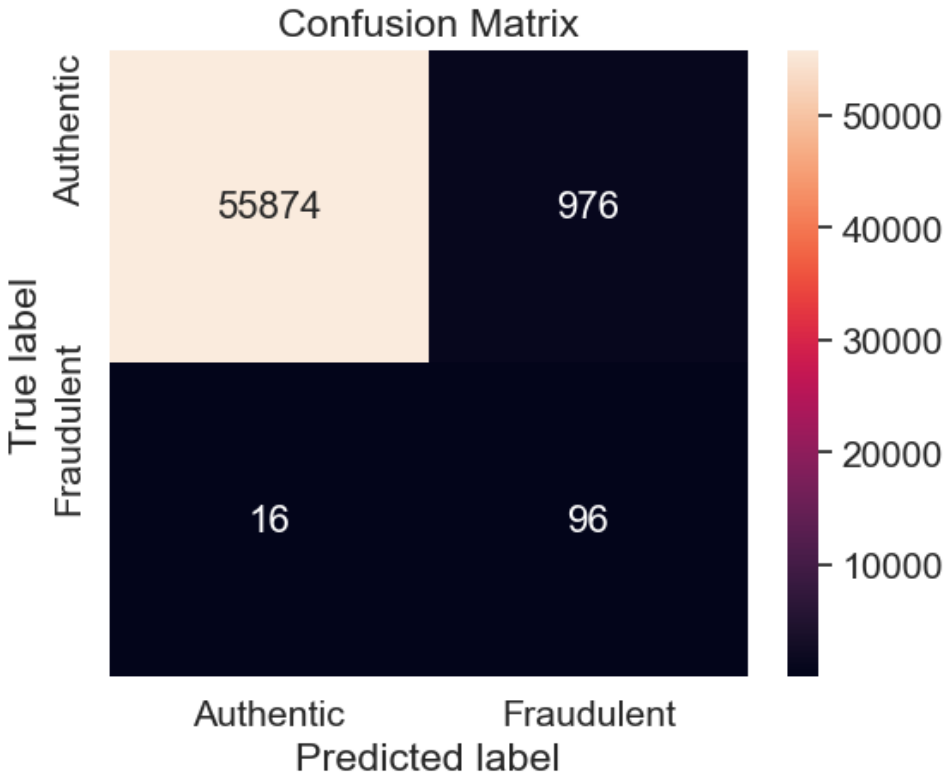
Out[ ]:

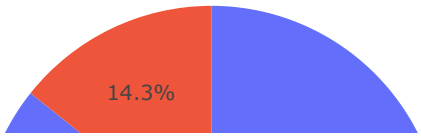
Metric	MCC	F1-Score	F2-Score	Recall	Precision	FM index	Specificity	G-mean	F0.5-Score
Performance score	0.298603	0.187259	0.353499	0.866071	0.104978	0.301527	0.985453	0.923836	0.127363

Random under-sampling with imbalanced-learn library

In [ ]:
```
# Elementos da matriz de confusão

classification(lda, X_train_under_imblearn, y_train_under_imblearn, X_test, y_test)
```

```
# Resumo das métricas de avaliação

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',
summary_lda_under_imblearn_extended.loc[len(summary_lda_under_imblearn_extended.index)] = ['ROC-
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
```





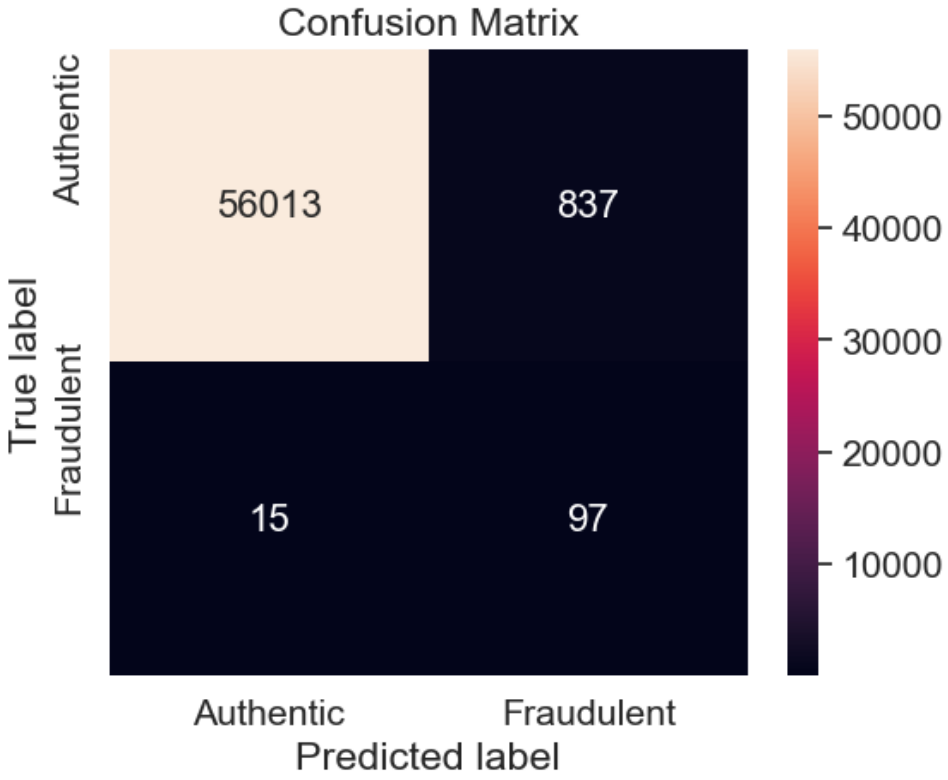
Out[ ]:

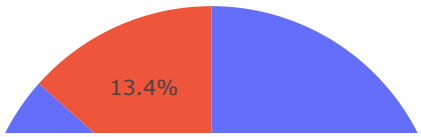
Metric	MCC	F1-Score	F2-Score	Recall	Precision	FM index	Specificity	G-mean	F0.5-Score
Performance score	0.273827	0.162162	0.315789	0.857143	0.089552	0.277054	0.982832	0.917838	0.109091

Random over-sampling with imbalanced-learn library

In [ ]:

```
# Elementos da matriz de confusão

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

# Resumo das métricas de avaliação

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', a
summary_lda_over_imblearn_extended.loc[len(summary_lda_over_imblearn_extended.index)] = ['ROC-AU
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
```





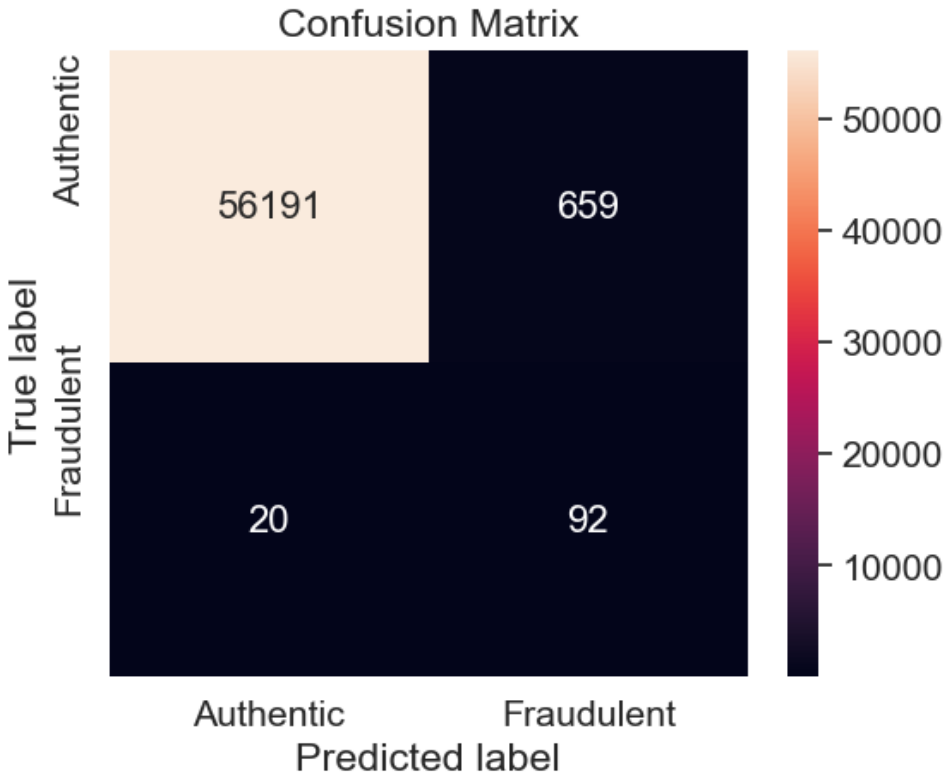
Out[ ]:

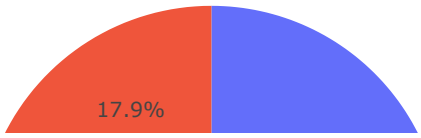
	Metric	MCC	F1-Score	F2-Score	Recall	Precision	FM index	Specificity	G-mean	F0.5-Score	A
Performance score		0.296965	0.185468	0.350941	0.866071	0.103854	0.299909	0.985277	0.923753	0.12604	

Synthetic minority over-sampling technique (SMOTE)

In [ ]:
```
# Elementos da matriz de confusão

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

```
# Resumo das métricas de avaliação

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
summary_lda_over_smote_extended.loc[len(summary_lda_over_smote_extended.index)] = ['ROC-AUC', ro
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
```



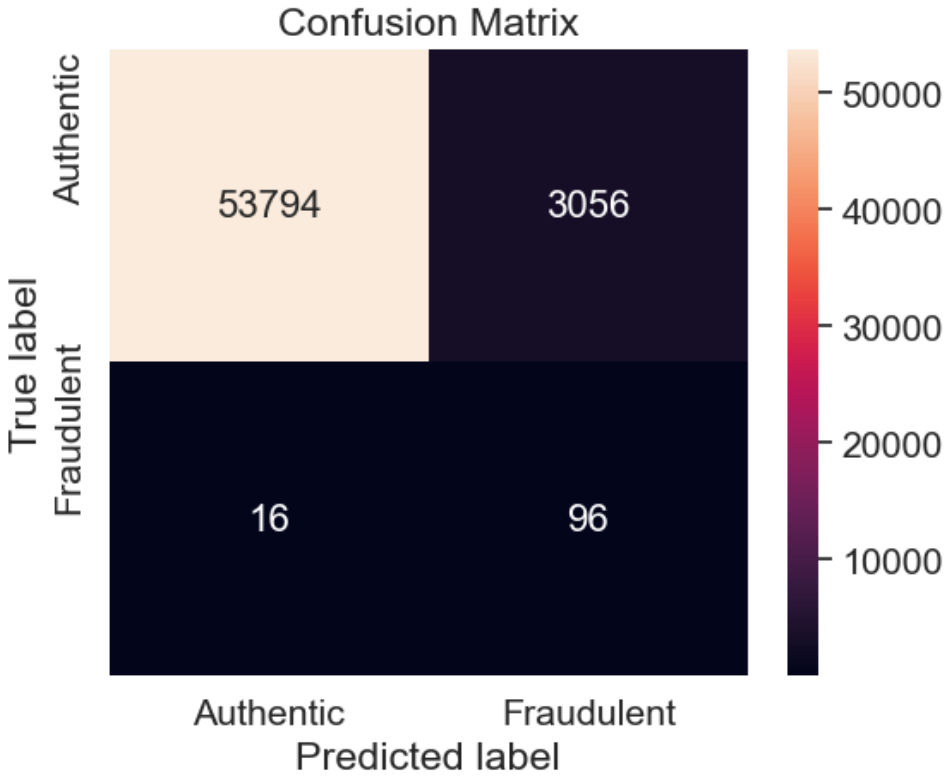


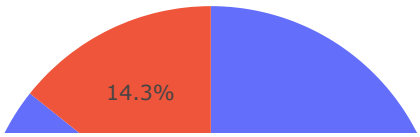
Out[ ]:

Metric	MCC	F1-Score	F2-Score	Recall	Precision	FM index	Specificity	G-mean	F0.5-Score	A
Performance score	0.314515	0.21321	0.383653	0.821429	0.122503	0.317219	0.988408	0.901059	0.147625	

Under-sampling via NearMiss

In [ ]:
```
# Elementos da matriz de confusão

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

# Resumo das métricas de avaliação

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_pre
summary_lda_under_nm_extended.loc[len(summary_lda_under_nm_extended.index)] = ['ROC-AUC', roc_au
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
```





Out[ ]:

	Metric	MCC	F1-Score	F2-Score	Recall	Precision	FM index	Specificity	G-mean	F0.5-Score
Performance score		0.155659	0.058824	0.133333	0.857143	0.030457	0.161573	0.946245	0.900592	0.037736

Summary of LDA models

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

summary_lda['Metric'] = EvalMetricLabels
summary_lda_list = [summary_lda_unaltered, summary_lda_under, summary_lda_over, summary_lda_unde
```

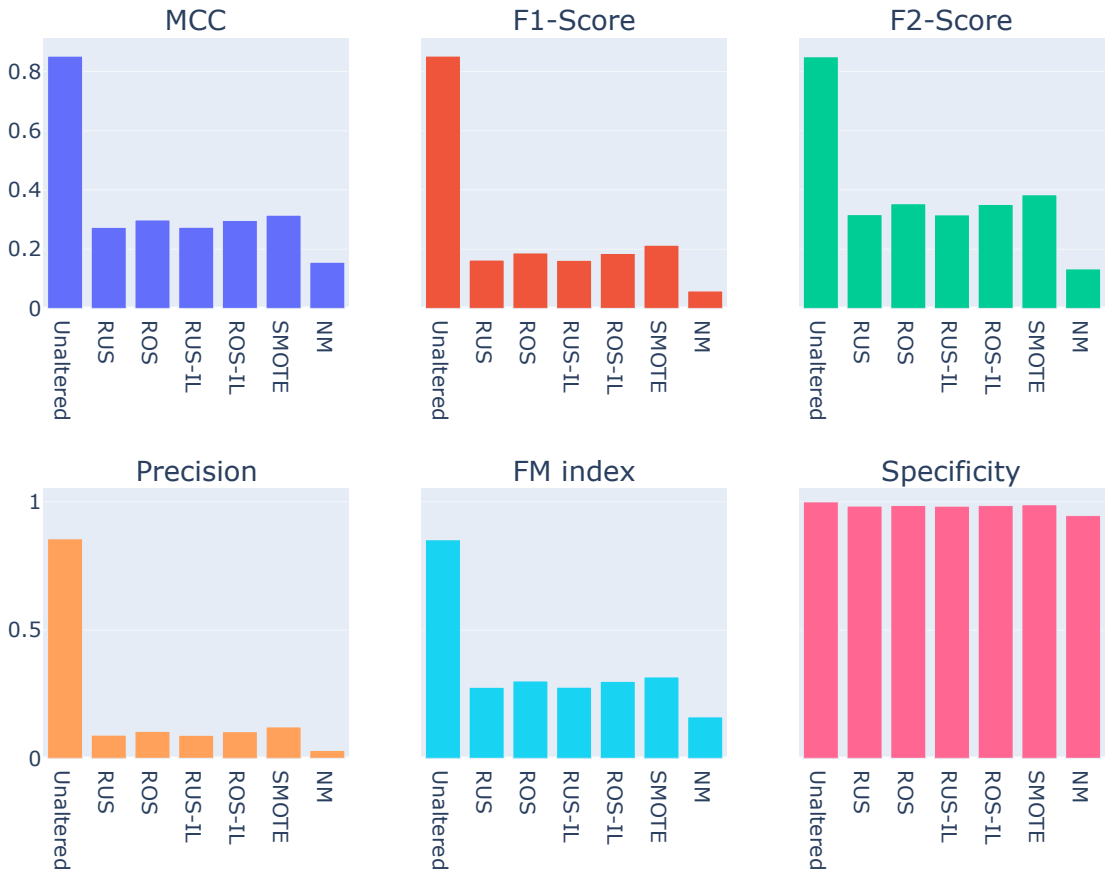
```
                    summary_lda_over_imblearn, summary_lda_over_smote, summary_lda_under_nm]

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
```

```
C:\Users\Diones\AppData\Local\Temp\ipykernel_14028\741047787.py:8: FutureWarning:

Passing 'suffixes' which cause duplicate columns {'Performance score_x'} in the result is depreca
ted and will raise a MergeError in a future version.

C:\Users\Diones\AppData\Local\Temp\ipykernel_14028\741047787.py:8: FutureWarning:

Passing 'suffixes' which cause duplicate columns {'Performance score_x'} in the result is depreca
ted and will raise a MergeError in a future version.
```

Out[ ]:

Metric	Unaltered	RUS	ROS	RUS-IL	ROS-IL	SMOTE	NM
MCC	0.851736	0.273540	0.298603	0.273827	0.296965	0.314515	0.155659
F1-Score	0.852018	0.163230	0.187259	0.162162	0.185468	0.213210	0.058824
F2-Score	0.849732	0.316667	0.353499	0.315789	0.350941	0.383653	0.133333
Recall	0.848214	0.848214	0.866071	0.857143	0.866071	0.821429	0.857143
Precision	0.855856	0.090304	0.104978	0.089552	0.103854	0.122503	0.030457
FM index	0.852027	0.276762	0.301527	0.277054	0.299909	0.317219	0.161573
Specificity	0.999719	0.983166	0.985453	0.982832	0.985277	0.988408	0.946245
G-mean	0.920856	0.913201	0.923836	0.917838	0.923753	0.901059	0.900592
F0.5-Score	0.854317	0.109954	0.127363	0.109091	0.126040	0.147625	0.037736
Accuracy	0.999421	0.982901	0.985218	0.982585	0.985043	0.988080	0.946069

In [ ]:
```
# Comparação visual do modelo aplicado em diferentes conjuntos de treinamento por meio de várias

summary_visual(summary_lda)
```

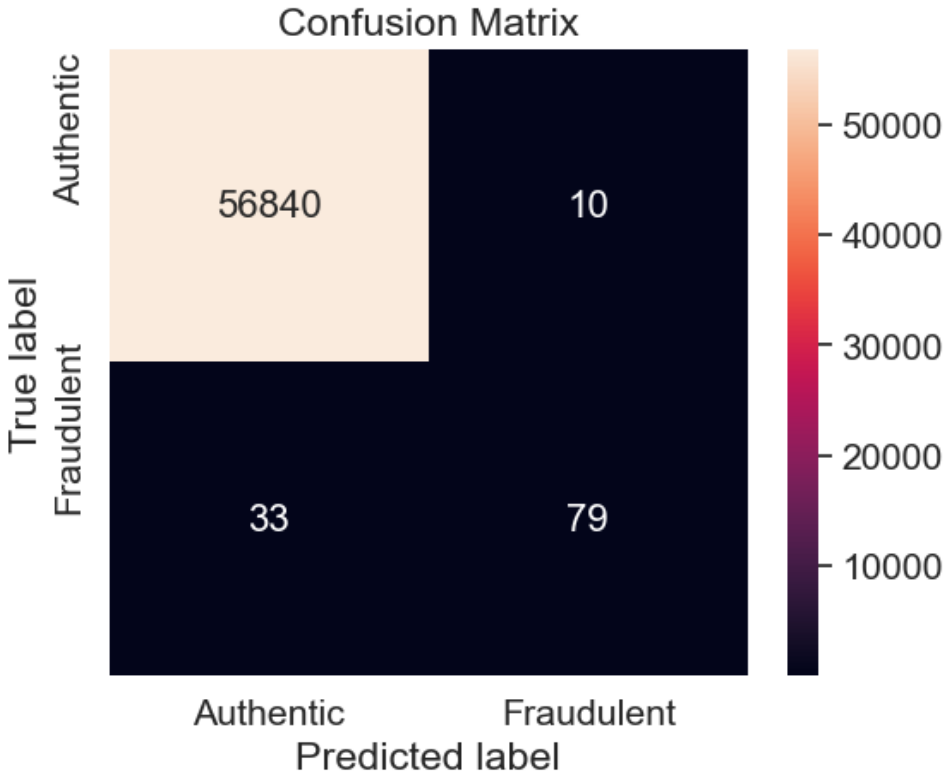


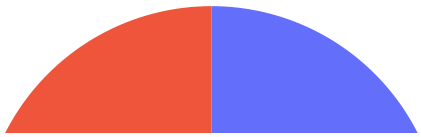
12. Stochastic Gradient Descent (SGD)

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

Unaltered training set

In [ ]:
```
# Elementos da matriz de confusão

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

# Resumo das métricas de avaliação

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_p
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
```





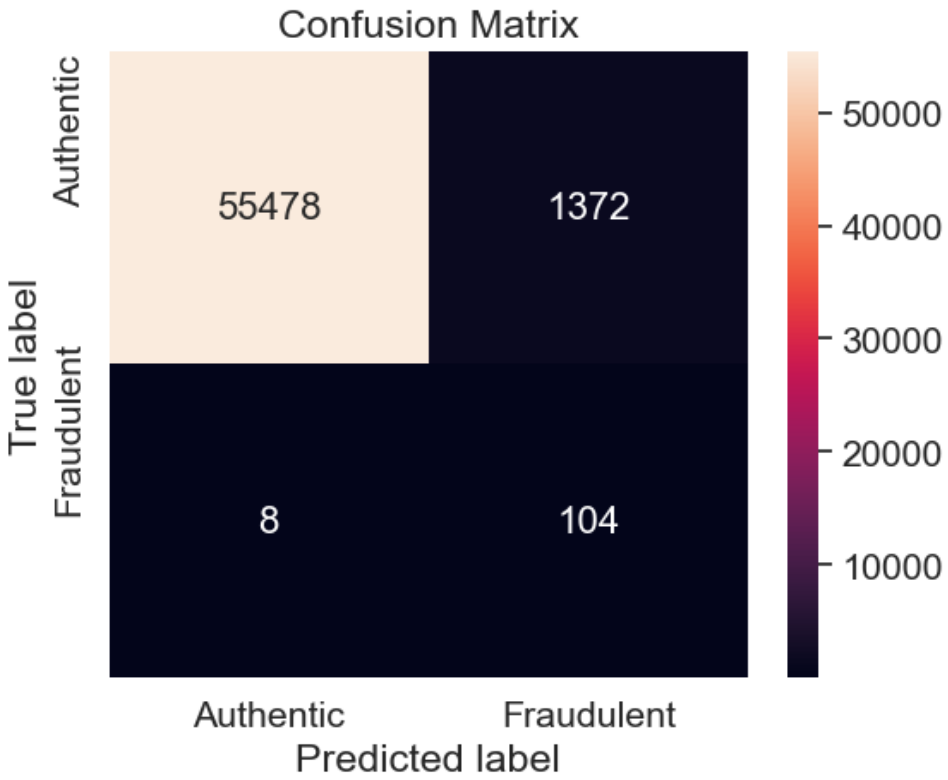
```
e:\miniconda3\envs\Bootcamp\lib\site-packages\sklearn\base.py:432: UserWarning:

X has feature names, but SGDClassifier was fitted without feature names
```

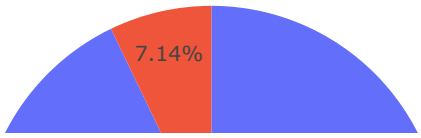
Out[ ]:

Metric	MCC	F1-Score	F2-Score	Recall	Precision	FM index	Specificity	G-mean	F0.5-Score	A
Performance score	0.79091	0.78607	0.735568	0.705357	0.88764	0.791267	0.999824	0.839782	0.844017	0

Random under-sampling

In [ ]:
```
# Elementos da matriz de confusão

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

# Resumo das métricas de avaliação

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
```






```
e:\miniconda3\envs\Bootcamp\lib\site-packages\sklearn\base.py:432: UserWarning:

X has feature names, but SGDClassifier was fitted without feature names
```

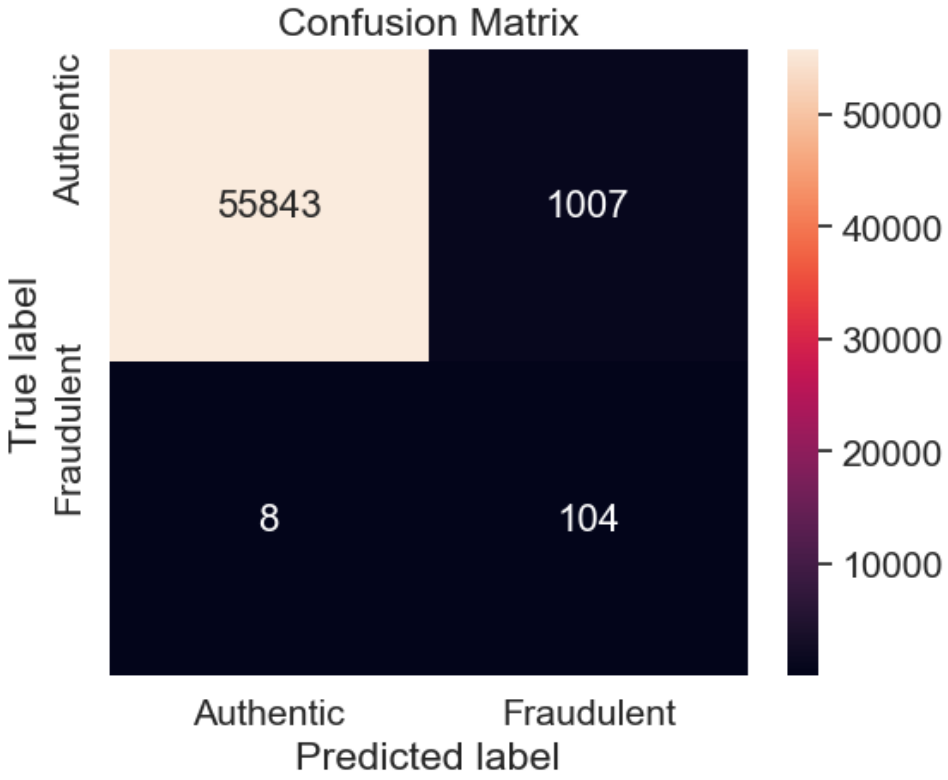
Out[ ]:

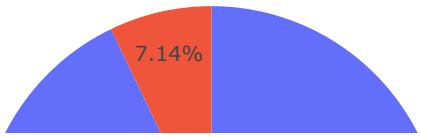
Metric	MCC	F1-Score	F2-Score	Recall	Precision	FM index	Specificity	G-mean	F0.5-Score	A
Performance score	0.252184	0.130982	0.27027	0.928571	0.070461	0.255789	0.975866	0.951925	0.086436	

Random over-sampling

In [ ]:

```
# Elementos da matriz de confusão

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

# Resumo das métricas de avaliação

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
```





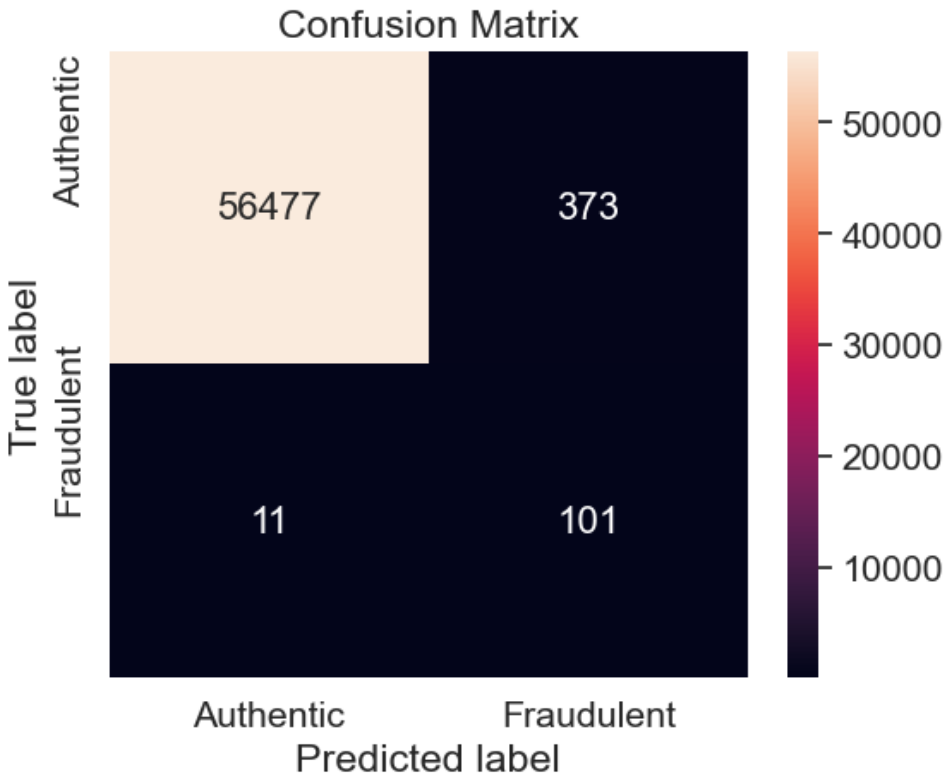

```
e:\miniconda3\envs\Bootcamp\lib\site-packages\sklearn\base.py:432: UserWarning:

X has feature names, but SGDClassifier was fitted without feature names
```

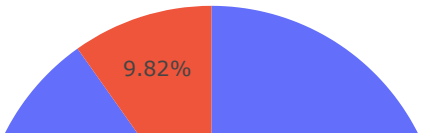
Out[ ]:

Metric	MCC	F1-Score	F2-Score	Recall	Precision	FM index	Specificity	G-mean	F0.5-Score
Performance score	0.291778	0.170074	0.333547	0.928571	0.093609	0.294827	0.982287	0.955052	0.114135

Random under-sampling with imbalanced-learn library

In [ ]:
```
# Elementos da matriz de confusão

classification(sgd, X_train_under_imblearn_scaled_minmax, y_train_under_imblearn, X_test_scaled_

# Resumo das métricas de avaliação

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',
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
```






```
e:\miniconda3\envs\Bootcamp\lib\site-packages\sklearn\base.py:432: UserWarning:

X has feature names, but SGDClassifier was fitted without feature names
```

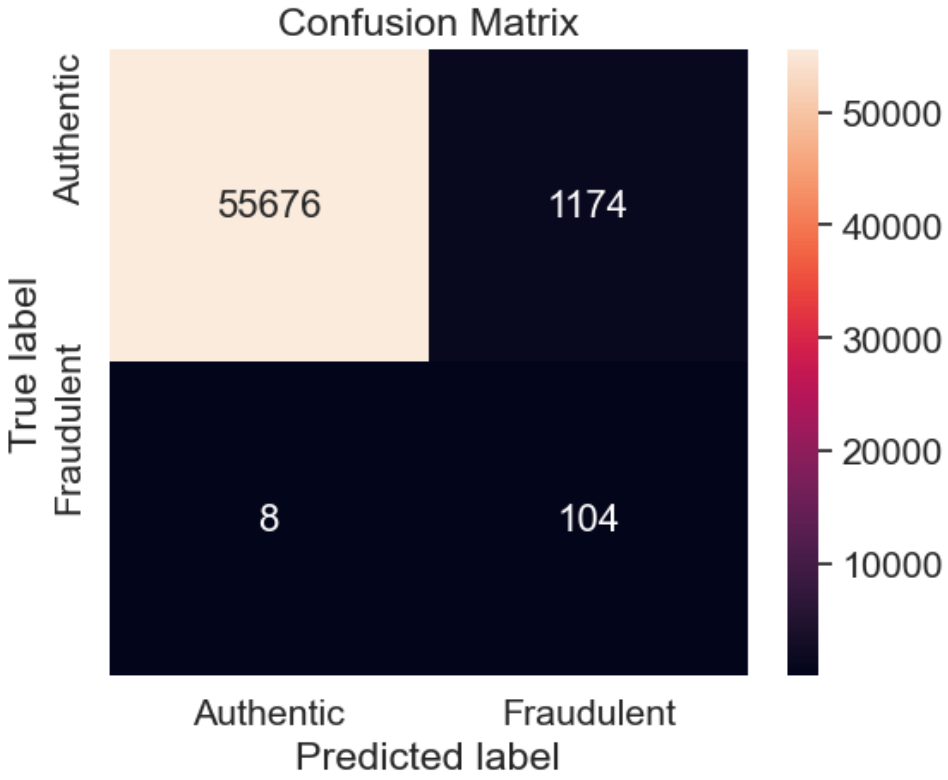
Out[ ]:

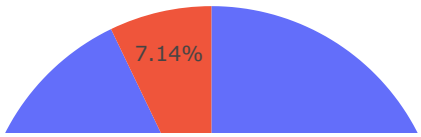
Metric	MCC	F1-Score	F2-Score	Recall	Precision	FM index	Specificity	G-mean	F0.5-Score	A
Performance score	0.436555	0.34471	0.547722	0.901786	0.21308	0.438352	0.993439	0.946504	0.251494	

Random over-sampling with imbalanced-learn library

In [ ]:

```
# Elementos da matriz de confusão

classification(sgd, X_train_over_imblearn_scaled_minmax, y_train_over_imblearn, X_test_scaled_mi

# Resumo das métricas de avaliação

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', a
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
```





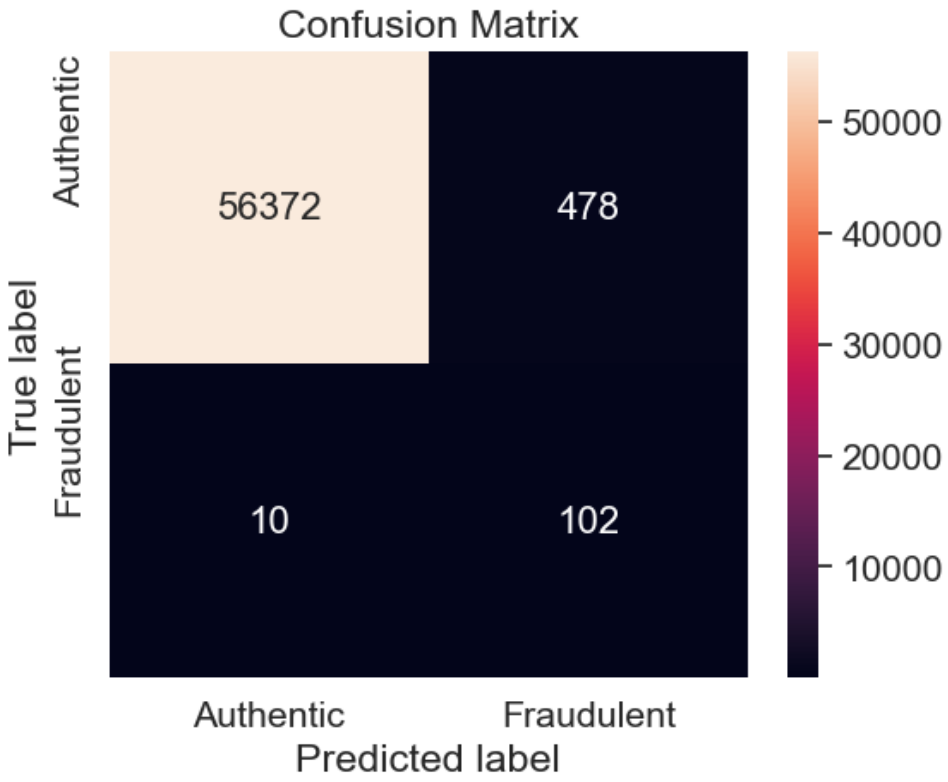

```
e:\miniconda3\envs\Bootcamp\lib\site-packages\sklearn\base.py:432: UserWarning:

X has feature names, but SGDClassifier was fitted without feature names
```

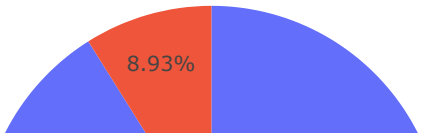
Out[ ]:

Metric	MCC	F1-Score	F2-Score	Recall	Precision	FM index	Specificity	G-mean	F0.5-Score	A
Performance score	0.271576	0.14964	0.301275	0.928571	0.081377	0.27489	0.979349	0.953622	0.099541	0

Synthetic minority over-sampling technique (SMOTE)

In [ ]:
```
# Elementos da matriz de confusão

classification(sgd, X_train_over_smote_scaled_minmax, y_train_over_smote, X_test_scaled_minmax,

# Resumo das métricas de avaliação

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
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
```






```
e:\miniconda3\envs\Bootcamp\lib\site-packages\sklearn\base.py:432: UserWarning:

X has feature names, but SGDClassifier was fitted without feature names
```

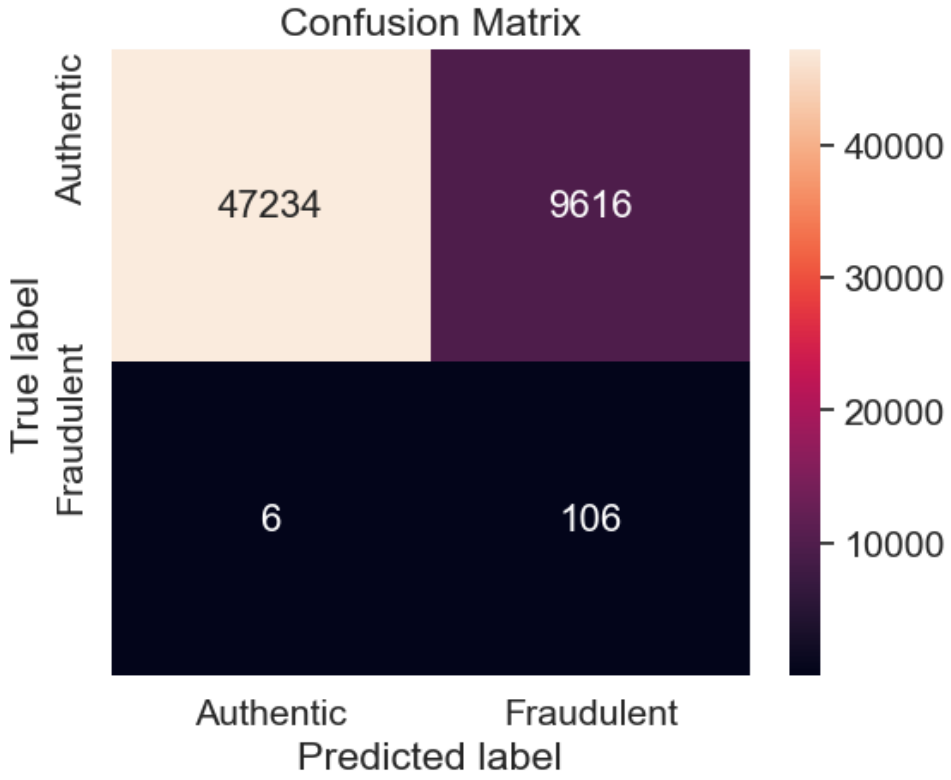
Out[ ]:

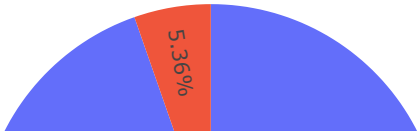
Metric	MCC	F1-Score	F2-Score	Recall	Precision	FM index	Specificity	G-mean	F0.5-Score	A
Performance score	0.398147	0.294798	0.496109	0.910714	0.175862	0.4002	0.991592	0.950293	0.209704	0

Under-sampling via NearMiss

In [ ]:

```
# Elementos da matriz de confusão

classification(sgd, X_train_under_nm_scaled_minmax, y_train_under_nm, X_test_scaled_minmax, y_te

# Resumo das métricas de avaliação

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_pre
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
```






```
e:\miniconda3\envs\Bootcamp\lib\site-packages\sklearn\base.py:432: UserWarning:

X has feature names, but SGDClassifier was fitted without feature names
```

Out[ ]:

Metric	MCC	F1-Score	F2-Score	Recall	Precision	FM index	Specificity	G-mean	F0.5-Score	A
Performance score	0.091521	0.021558	0.052114	0.946429	0.010903	0.101583	0.830853	0.88676	0.01359	

Summary of SGD models

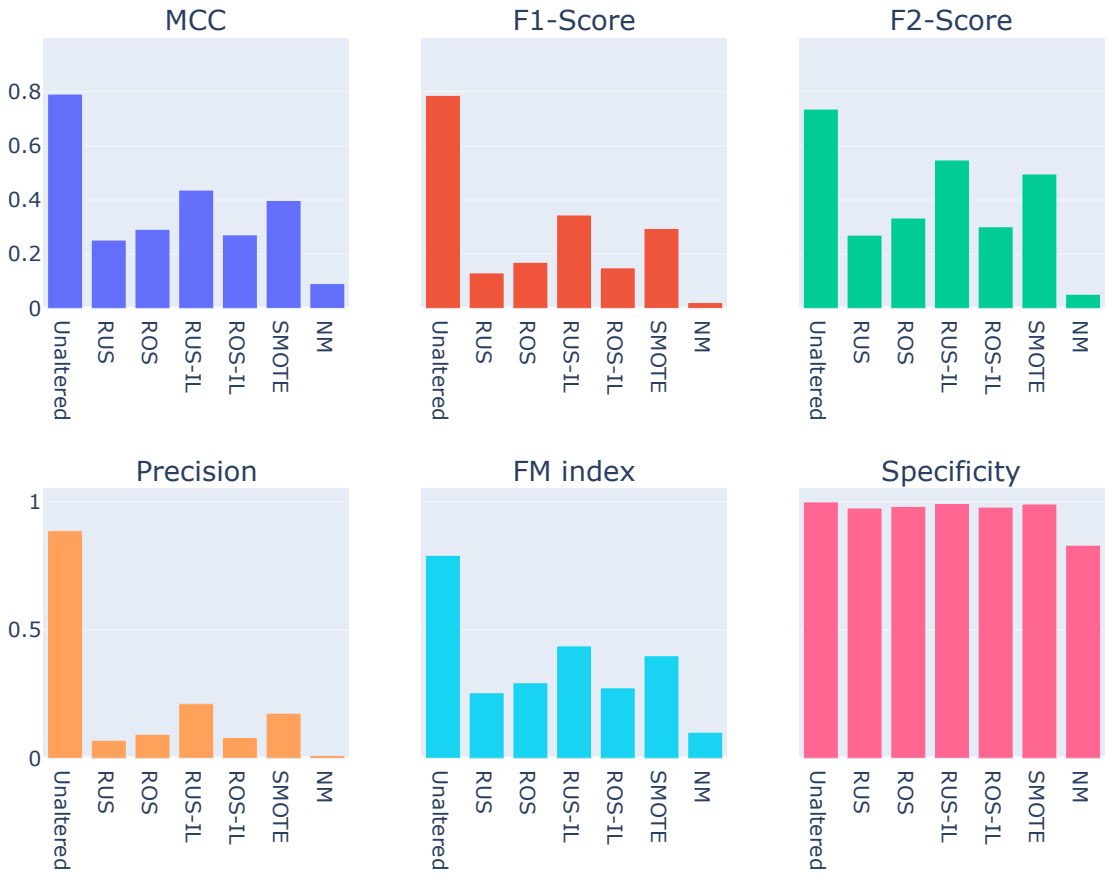
In [ ]:
```
summary_sgd = pd.DataFrame(columns = ['Metric'])

summary_sgd['Metric'] = EvalMetricLabels
summary_sgd_list = [summary_sgd_unaltered, summary_sgd_under, summary_sgd_over, summary_sgd_unde
                    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
```

```
C:\Users\Diones\AppData\Local\Temp\ipykernel_14028\4039655680.py:8: FutureWarning:

Passing 'suffixes' which cause duplicate columns {'Performance score_x'} in the result is depreca
ted and will raise a MergeError in a future version.

C:\Users\Diones\AppData\Local\Temp\ipykernel_14028\4039655680.py:8: FutureWarning:

Passing 'suffixes' which cause duplicate columns {'Performance score_x'} in the result is depreca
ted and will raise a MergeError in a future version.
```

Out[ ]:

Metric	Unaltered	RUS	ROS	RUS-IL	ROS-IL	SMOTE	NM
MCC	0.790910	0.252184	0.291778	0.436555	0.271576	0.398147	0.091521
F1-Score	0.786070	0.130982	0.170074	0.344710	0.149640	0.294798	0.021558
F2-Score	0.735568	0.270270	0.333547	0.547722	0.301275	0.496109	0.052114
Recall	0.705357	0.928571	0.928571	0.901786	0.928571	0.910714	0.946429
Precision	0.887640	0.070461	0.093609	0.213080	0.081377	0.175862	0.010903
FM index	0.791267	0.255789	0.294827	0.438352	0.274890	0.400200	0.101583
Specificity	0.999824	0.975866	0.982287	0.993439	0.979349	0.991592	0.830853
G-mean	0.839782	0.951925	0.955052	0.946504	0.953622	0.950293	0.886760
F0.5-Score	0.844017	0.086436	0.114135	0.251494	0.099541	0.209704	0.013590
Accuracy	0.999245	0.975773	0.982181	0.993259	0.979249	0.991433	0.831080

In [ ]:
```
# Comparação visual do modelo aplicado em diferentes conjuntos de treinamento por meio de várias

summary_visual(summary_sgd)
```



13. Ridge Classifier

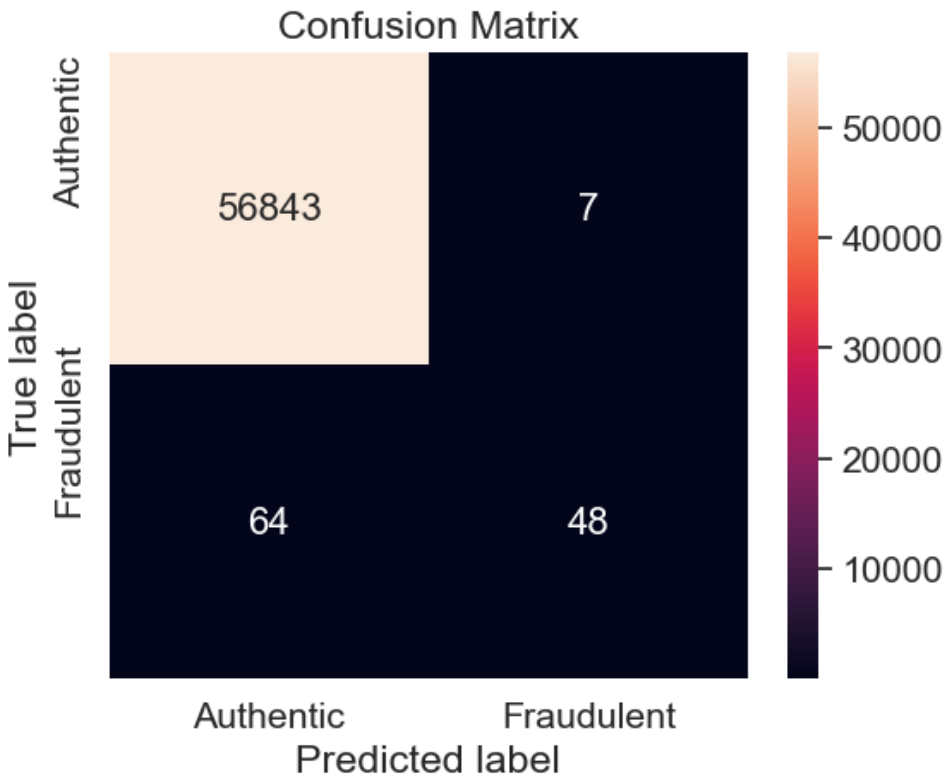
In [ ]:
```
ridge = RidgeClassifier()
```

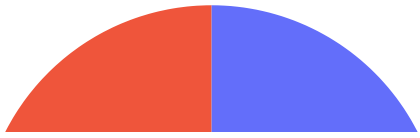
Usamos recursos normalizados, pois o classificador de cumeeira emprega a regularização l^2 por meio de um termo de penalidade aditiva na função objetiva.

Unaltered training set

In [ ]:
```
# Elementos da matriz de confusão

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

# Resumo das métricas de avaliação

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', avera
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
```





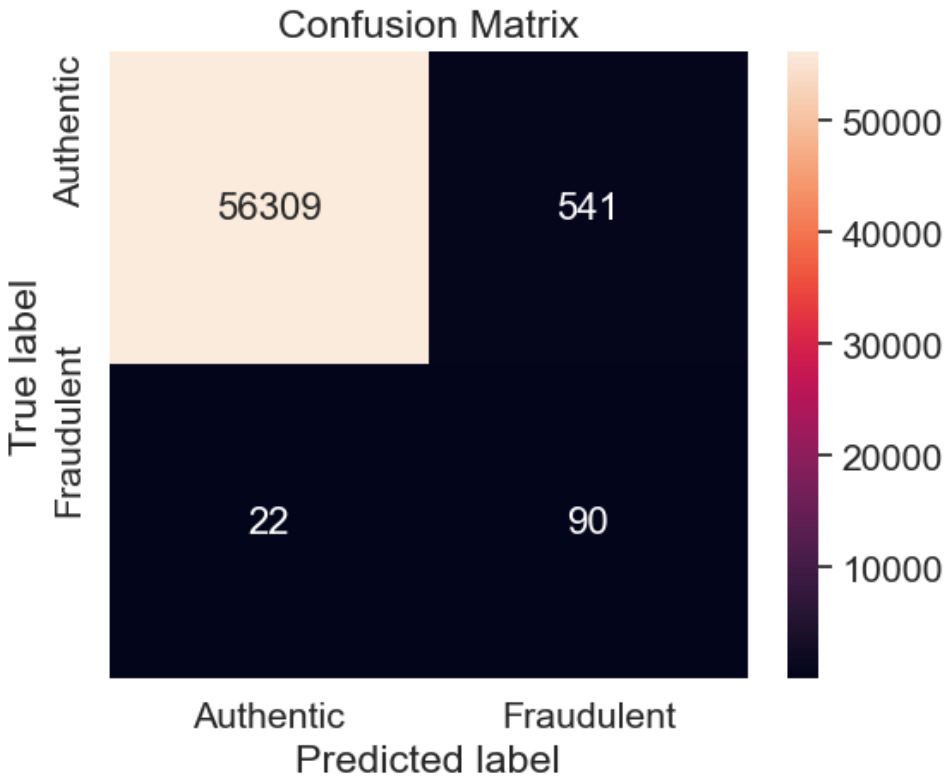
```
e:\miniconda3\envs\Bootcamp\lib\site-packages\sklearn\base.py:432: UserWarning:

X has feature names, but RidgeClassifier was fitted without feature names
```

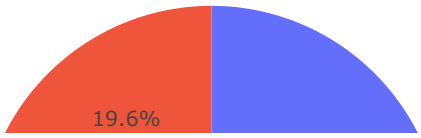
Out[ ]:

Metric	MCC	F1-Score	F2-Score	Recall	Precision	FM index	Specificity	G-mean	F0.5-Score	A
Performance score	0.611095	0.57485	0.477137	0.428571	0.872727	0.611577	0.999877	0.654613	0.722892	

Random under-sampling

In [ ]:
```
# Elementos da matriz de confusão

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

# Resumo das métricas de avaliação

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_preci
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
```






```
e:\miniconda3\envs\Bootcamp\lib\site-packages\sklearn\base.py:432: UserWarning:

X has feature names, but RidgeClassifier was fitted without feature names
```

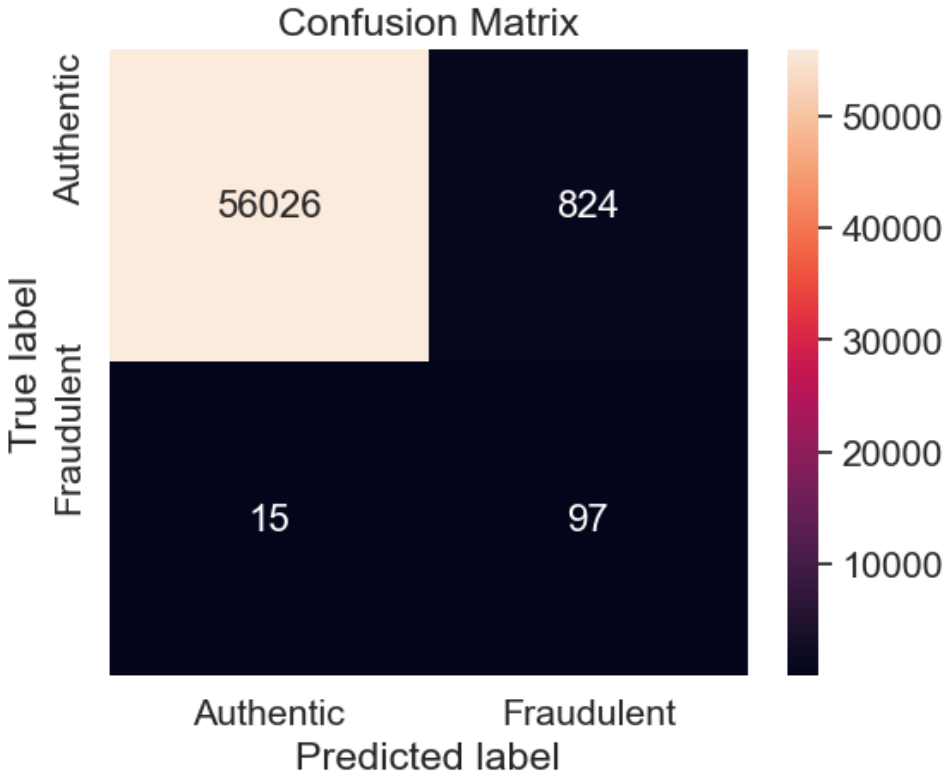
Out[ ]:

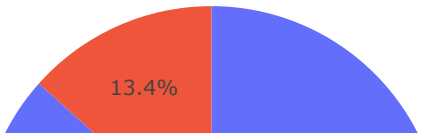
Metric	MCC	F1-Score	F2-Score	Recall	Precision	FM index	Specificity	G-mean	F0.5-Score
Performance score	0.336075	0.242261	0.417053	0.803571	0.142631	0.338547	0.990484	0.892146	0.170713

Random over-sampling

In [ ]:

```
# Elementos da matriz de confusão

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

# Resumo das métricas de avaliação

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_precisi
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
```





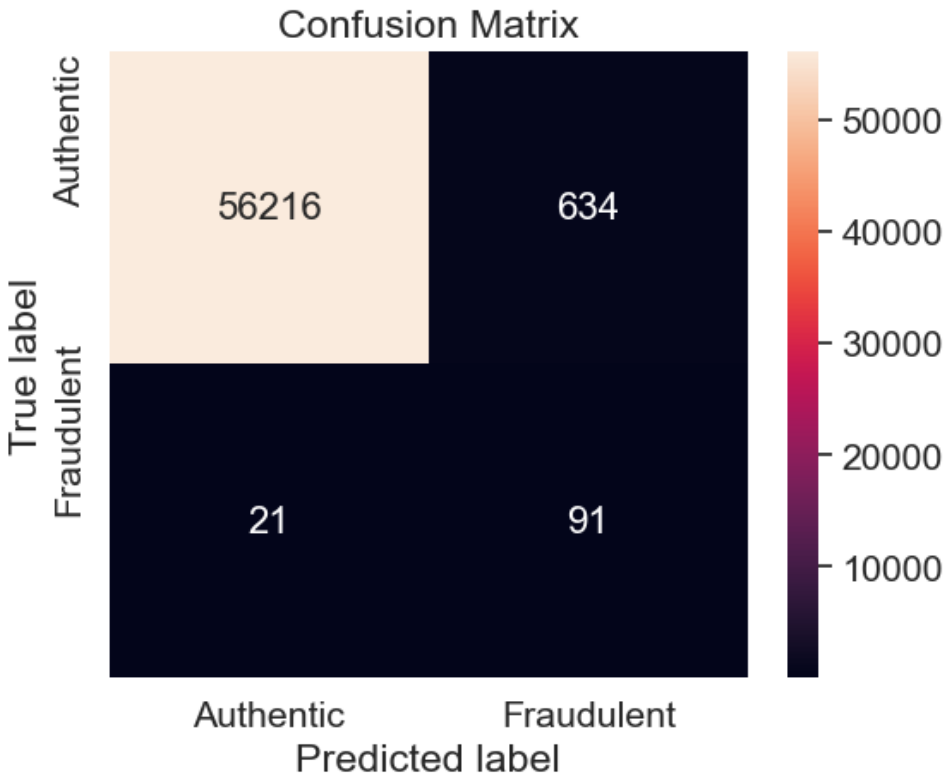

```
e:\miniconda3\envs\Bootcamp\lib\site-packages\sklearn\base.py:432: UserWarning:

X has feature names, but RidgeClassifier was fitted without feature names
```

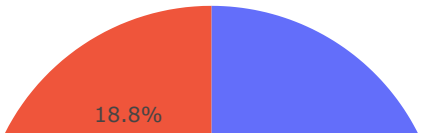
Out[ ]:

Metric	MCC	F1-Score	F2-Score	Recall	Precision	FM index	Specificity	G-mean	F0.5-Score
Performance score	0.299099	0.187803	0.354273	0.866071	0.10532	0.302018	0.985506	0.923861	0.127766

Random under-sampling with imbalanced-learn library

In [ ]:
```
# Elementos da matriz de confusão

classification(ridge, X_train_under_imblearn_scaled_minmax, y_train_under_imblearn, X_test_scale

# Resumo das métricas de avaliação

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)] = ['
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 = T
summary_ridge_under_imblearn_index
```






```
e:\miniconda3\envs\Bootcamp\lib\site-packages\sklearn\base.py:432: UserWarning:

X has feature names, but RidgeClassifier was fitted without feature names
```

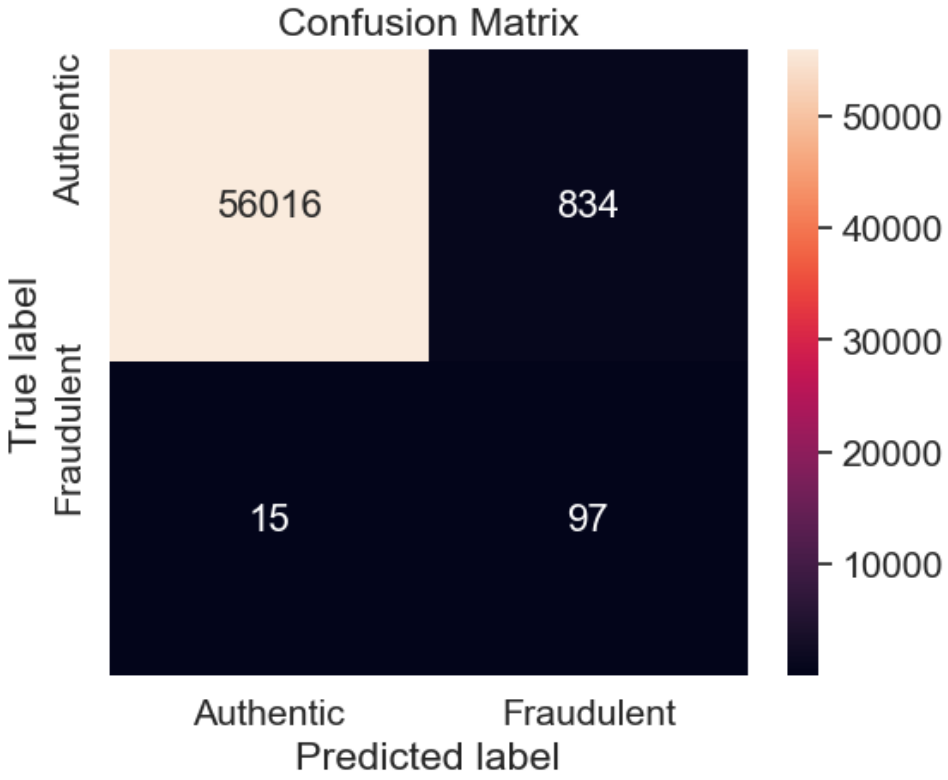
Out[ ]:

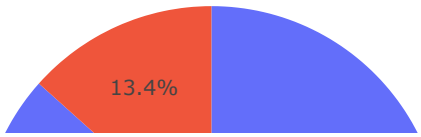
Metric	MCC	F1-Score	F2-Score	Recall	Precision	FM index	Specificity	G-mean	F0.5-Score	A
Performance score	0.316676	0.217443	0.387894	0.8125	0.125517	0.319347	0.988848	0.896348	0.151062	0

Random over-sampling with imbalanced-learn library

In [ ]:

```
# Elementos da matriz de confusão

classification(ridge, X_train_over_imblearn_scaled_minmax, y_train_over_imblearn, X_test_scaled_

# Resumo das métricas de avaliação

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
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 = Tru
summary_ridge_over_imblearn_index
```





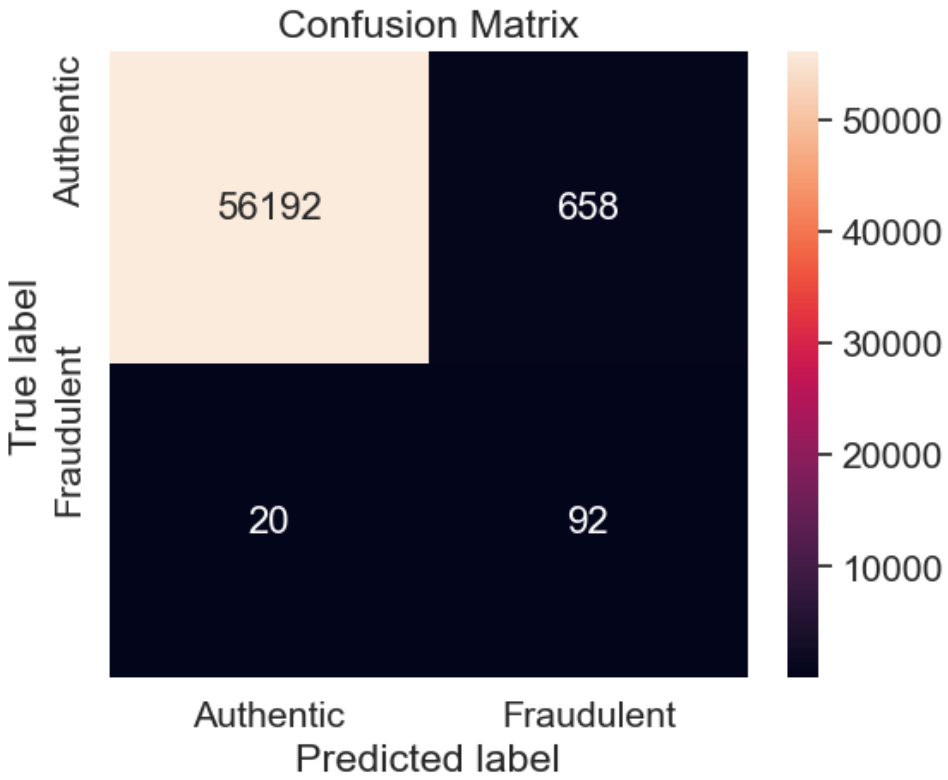

```
e:\miniconda3\envs\Bootcamp\lib\site-packages\sklearn\base.py:432: UserWarning:

X has feature names, but RidgeClassifier was fitted without feature names
```

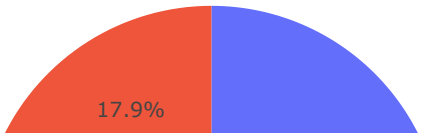
Out[ ]:

Metric	MCC	F1-Score	F2-Score	Recall	Precision	FM index	Specificity	G-mean	F0.5-Score
Performance score	0.297454	0.186002	0.351704	0.866071	0.104189	0.300392	0.98533	0.923778	0.126434

Synthetic minority over-sampling technique (SMOTE)

In [ ]:
```
# Elementos da matriz de confusão

classification(ridge, X_train_over_smote_scaled_minmax, y_train_over_smote, X_test_scaled_minmax

# Resumo das métricas de avaliação

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', ave
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
```






```
e:\miniconda3\envs\Bootcamp\lib\site-packages\sklearn\base.py:432: UserWarning:

X has feature names, but RidgeClassifier was fitted without feature names
```

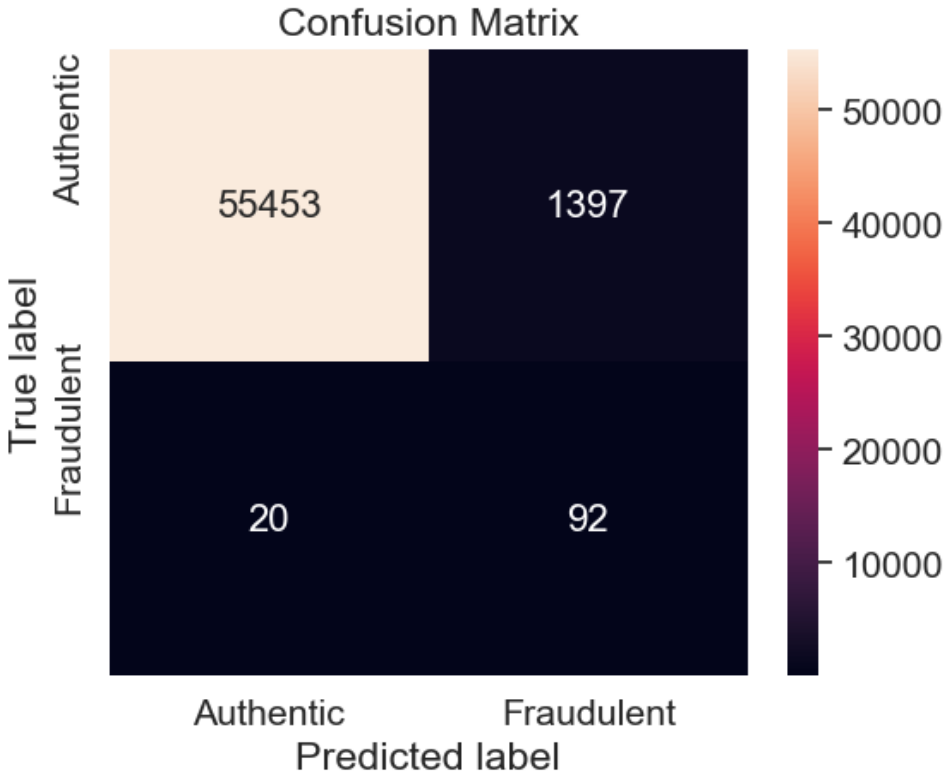
Out[ ]:

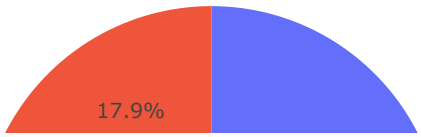
Metric	MCC	F1-Score	F2-Score	Recall	Precision	FM index	Specificity	G-mean	F0.5-Score	A
Performance score	0.314728	0.213457	0.383973	0.821429	0.122667	0.31743	0.988426	0.901067	0.147815	

Under-sampling via NearMiss

In [ ]:

```
# Elementos da matriz de confusão

classification(ridge, X_train_under_nm_scaled_minmax, y_train_under_nm, X_test_scaled_minmax, y_

# Resumo das métricas de avaliação

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
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
```






```
e:\miniconda3\envs\Bootcamp\lib\site-packages\sklearn\base.py:432: UserWarning:

X has feature names, but RidgeClassifier was fitted without feature names
```

Out[ ]:

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

Summary of ridge classifiers

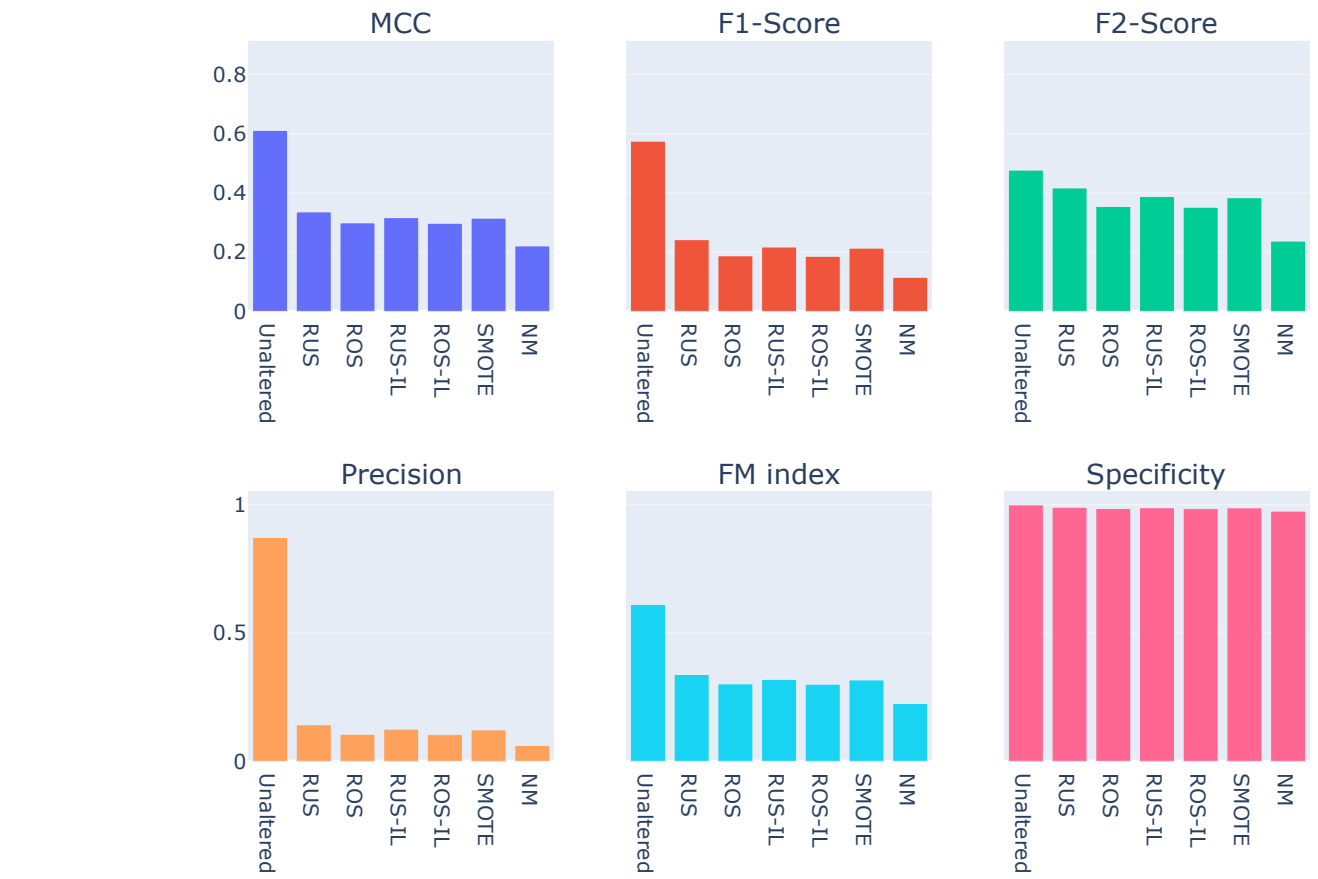
In [ ]:
```
summary_ridge = pd.DataFrame(columns = ['Metric'])

summary_ridge['Metric'] = EvalMetricLabels
summary_ridge_list = [summary_ridge_unaltered, summary_ridge_under, summary_ridge_over, summary_
                      summary_ridge_over_imblearn, summary_ridge_over_smote, summary_ridge_under

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
```

```
C:\Users\Diones\AppData\Local\Temp\ipykernel_14028\3761459512.py:8: FutureWarning:

Passing 'suffixes' which cause duplicate columns {'Performance score_x'} in the result is depreca
ted and will raise a MergeError in a future version.

C:\Users\Diones\AppData\Local\Temp\ipykernel_14028\3761459512.py:8: FutureWarning:

Passing 'suffixes' which cause duplicate columns {'Performance score_x'} in the result is depreca
ted and will raise a MergeError in a future version.
```

Out[ ]:

Metric	Unaltered	RUS	ROS	RUS-IL	ROS-IL	SMOTE	NM
MCC	0.611095	0.336075	0.299099	0.316676	0.297454	0.314728	0.221241
F1-Score	0.574850	0.242261	0.187803	0.217443	0.186002	0.213457	0.114928
F2-Score	0.477137	0.417053	0.354273	0.387894	0.351704	0.383973	0.237481
Recall	0.428571	0.803571	0.866071	0.812500	0.866071	0.821429	0.821429
Precision	0.872727	0.142631	0.105320	0.125517	0.104189	0.122667	0.061786
FM index	0.611577	0.338547	0.302018	0.319347	0.300392	0.317430	0.225285
Specificity	0.999877	0.990484	0.985506	0.988848	0.985330	0.988426	0.975427
G-mean	0.654613	0.892146	0.923861	0.896348	0.923778	0.901067	0.895122
F0.5-Score	0.722892	0.170713	0.127766	0.151062	0.126434	0.147815	0.075808
Accuracy	0.998754	0.990116	0.985271	0.988501	0.985095	0.988097	0.975124

In [ ]:
```
# Comparação visual do modelo aplicado em diferentes conjuntos de treinamento por meio de várias

summary_visual(summary_ridge)
```



14. Conclusion

Escolhemos o conjunto de treinamento de cada modelo com o melhor desempenho e tabulamos seu desempenho em termos de **F2-Score**, que considera os fatos de que o conjunto de dados é desequilibrado, a classe positiva (transações fraudulentas) é mais importante do que a classe negativa (transações autênticas) e também que os falsos negativos são mais caros do que os falsos positivos. Além disso, relatamos **MCC** (captura o desempenho geral em todas as classes) e **Recall** (concentra-se apenas na classe positiva crucial).

In [ ]:
```
# Comparação de modelos de classificação

"""
Na tabela final, os modelos são ordenados por ordem decrescente do seu desempenho no conjunto de

O conjunto de treino que é fornecido a um classificador é mencionado entre parêntesis a seguir a

Inalterado: conjunto de treino inalterado
ROS-IL: sobreamostragem aleatória da classe minoritária através da biblioteca de aprendizagem de
SMOTE: sobreamostragem da classe minoritária através da técnica de sobreamostragem de minorias s
"""

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_imb
                summary_svm_linear_unaltered_index, summary_nb_over_smote_index, summary_rf_over
                summary_lda_unaltered_index, summary_sgd_unaltered_index, summary_ridge_unaltere

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 = Fal
model_comparison_descending_F2
```

Out[ ]:

Classification model	F2-Score	MCC	Recall
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)	0.813953	0.815791	0.812500
KNN (Unaltered)	0.804067	0.852191	0.776786
SGD (Unaltered)	0.735568	0.790910	0.705357
Logistic Regression (Unaltered)	0.673624	0.754420	0.633929
Ridge Classifier (Unaltered)	0.477137	0.611095	0.428571
Naive Bayes (SMOTE)	0.464286	0.370613	0.812500

O algoritmo **Random Forest** aplicado ao conjunto de treinamento obtido após a sobreamostragem da classe minoritária (transações fraudulentas) por meio do **SMOTE** parece ser o melhor modelo de classificação para o problema em questão.

O **SMOTE** é uma das melhores opções para sobreamostragem da classe minoritária quando os dados são desequilibrados. Não é de surpreender que o **Random Forest** seja um dos classificadores mais adequados para o problema devido aos seguintes motivos:

- O algoritmo funciona bem ao lidar com grandes conjuntos de dados com dimensões elevadas.
- Ele é menos afetado pela presença de outliers nas variáveis de recursos em comparação com outros algoritmos.
- Ele não faz nenhuma suposição de distribuição nas variáveis de recursos.
- Ele lida com a colinearidade (dependência linear entre os recursos) implicitamente.
- Ignora automaticamente os recursos que não são úteis, fazendo efetivamente a seleção de recursos por conta própria.