

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 [ ]: # #Configura a GPU para funciona com o TensorFlow

# physical_devices = tf.config.list_physical_devices('GPU')
# tf.config.experimental.set_memory_growth(physical_devices[0], True)
# print(physical_devices)
# print("Número de GPU's : ", len(tf.config.list_physical_devices("GPU")))
```

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

```
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
	mean	94813.859575	1.168375e-15	3.416908e-16	-1.379537e-15	2.074095e-15
	std	47488.145955	1.958696e+00	1.651309e+00	1.516255e+00	1.415869e+00
	min	0.000000	-5.640751e+01	-7.271573e+01	-4.832559e+01	-5.683171e+00
	25%	54201.500000	-9.203734e-01	-5.985499e-01	-8.903648e-01	-8.486401e-01
	50%	84692.000000	1.810880e-02	6.548556e-02	1.798463e-01	-1.984653e-02
	75%	139320.500000	1.315642e+00	8.037239e-01	1.027196e+00	7.433413e-01
	max	172792.000000	2.454930e+00	2.205773e+01	9.382558e+00	1.687534e+01

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**

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

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

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

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

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

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**

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

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

G-mean = 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**

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

O Matthews Correlation Coefficient (MCC) é calculado pela fórmula:

$$\text{MCC} = (TP * TN - FP * FN) / \sqrt{(TP + FP) * (TP + FN) * (TN + FP) * (TN + FN)}$$

O Matthews Correlation Coefficient (MCC) é calculado pela fórmula:

$$MCC = (TP \times TN - FP \times FN) / \sqrt{(TP + FP) \times (TP + FN) \times (TN + FP) \times (TN + FN)}$$

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

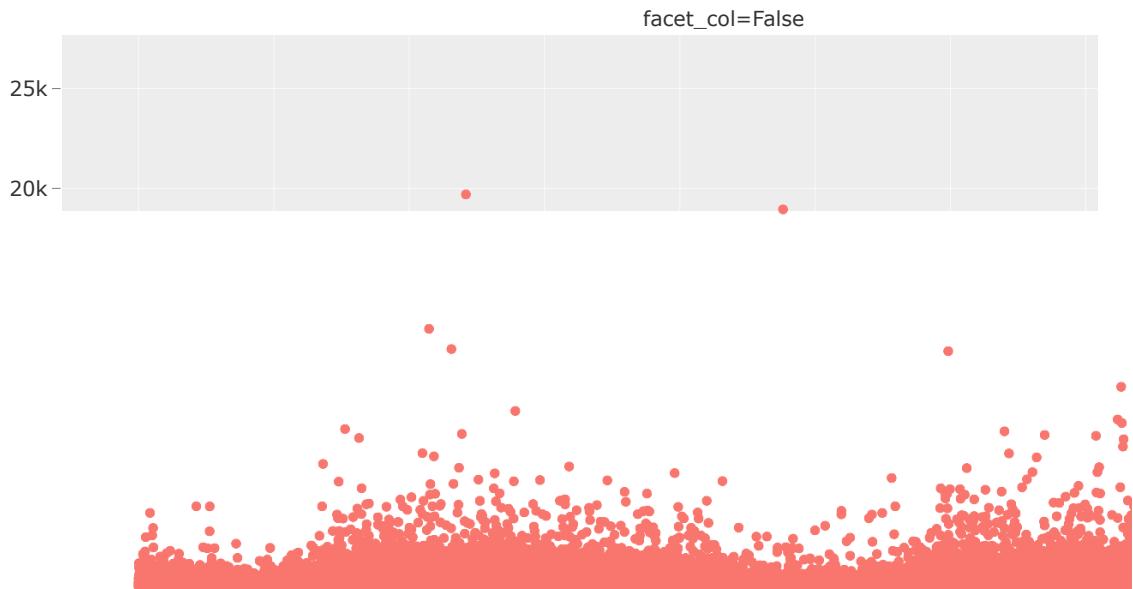
Separating the training set by class

```
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)

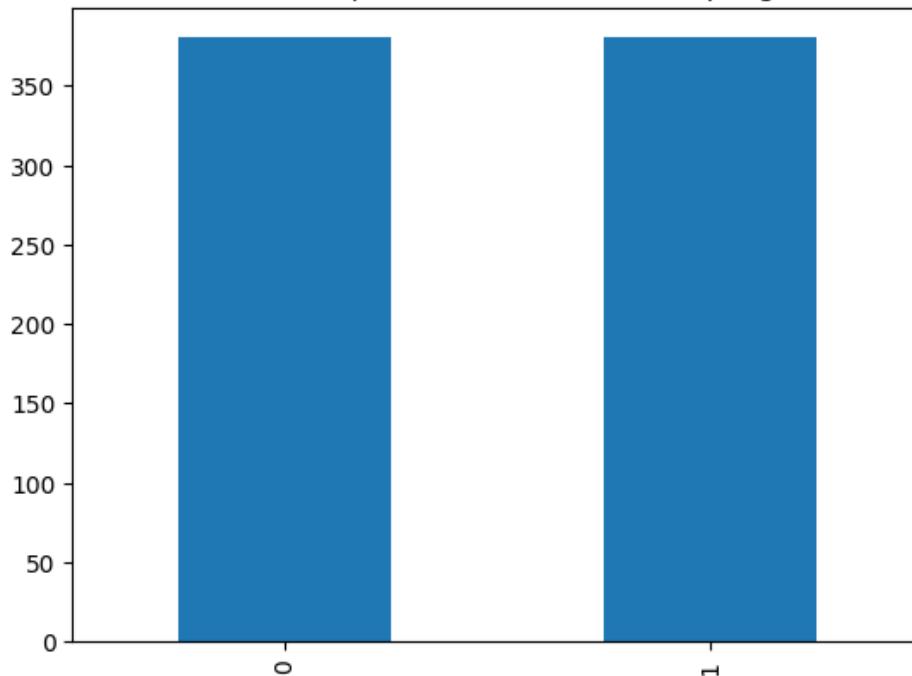
X_train_under = data_train_under.drop('Class', axis = 1)
y_train_under = data_train_under['Class']

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'}>
```

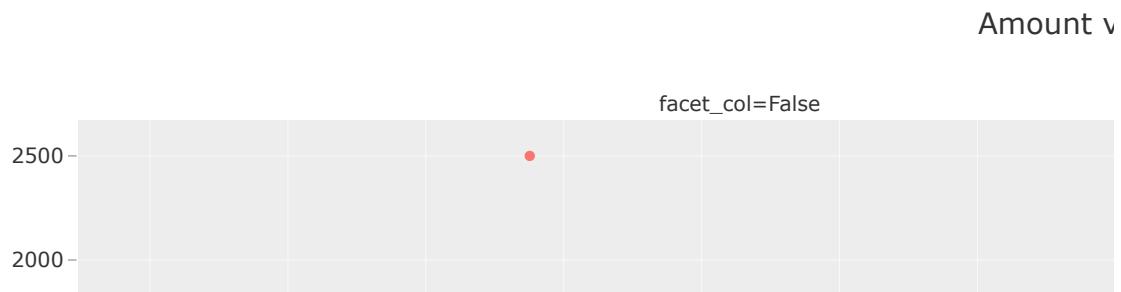
Class frequencies after under-sampling



```
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 = True)
data_train_over = pd.concat([data_train_authentic, data_train_fraudulent_over], axis = 0)

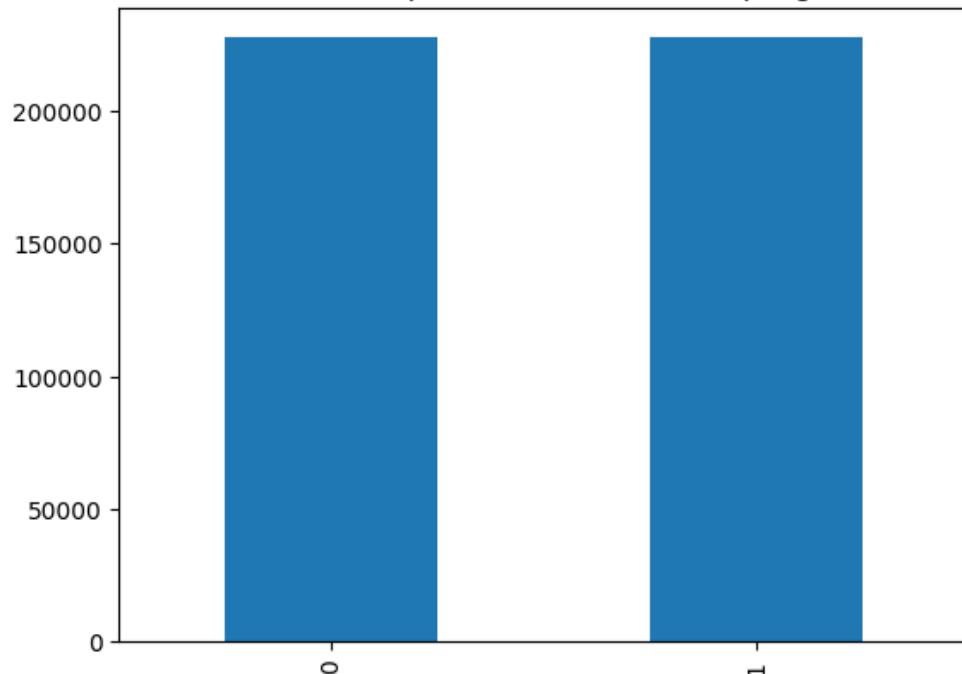
X_train_over = data_train_over.drop('Class', axis = 1)
y_train_over = data_train_over['Class']

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

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

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

Class frequencies after over-sampling



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

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)

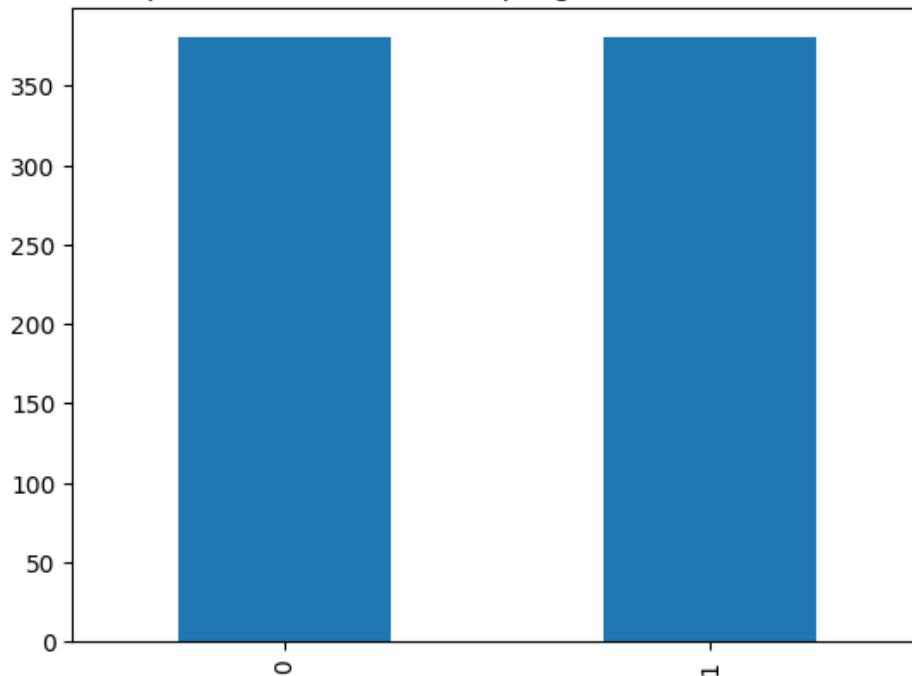
X_train_under_imblearn = data_train_under_imblearn.drop('Class', axis = 1)
y_train_under_imblearn = data_train_under_imblearn['Class']

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'}>
```

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']

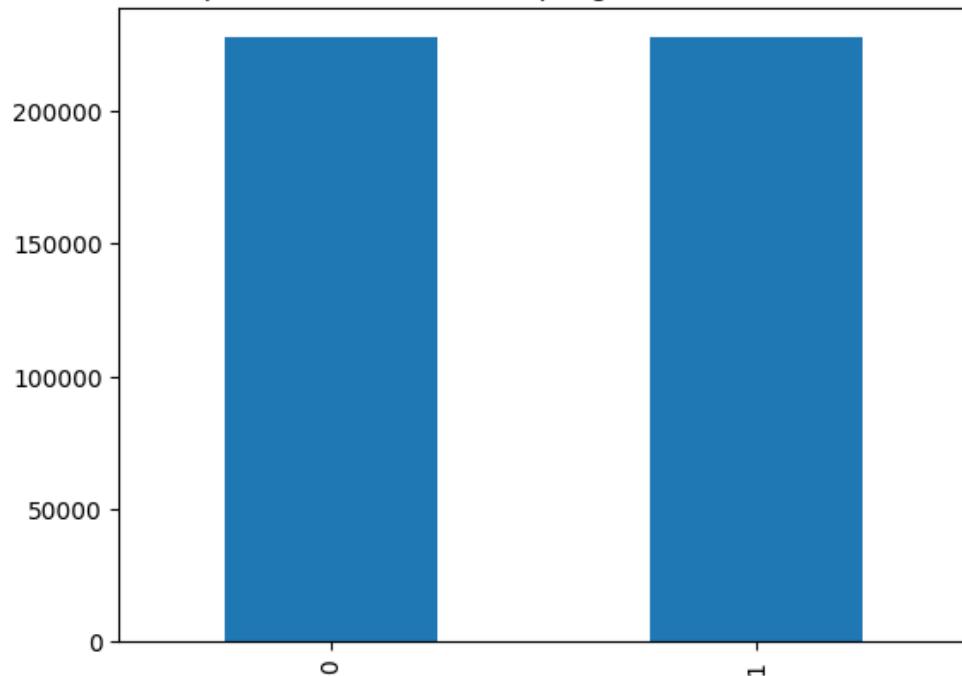
data_train_over_imblearn = pd.concat([X_train_ros, y_train_ros], axis = 1)

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 imbal')

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'}>
```

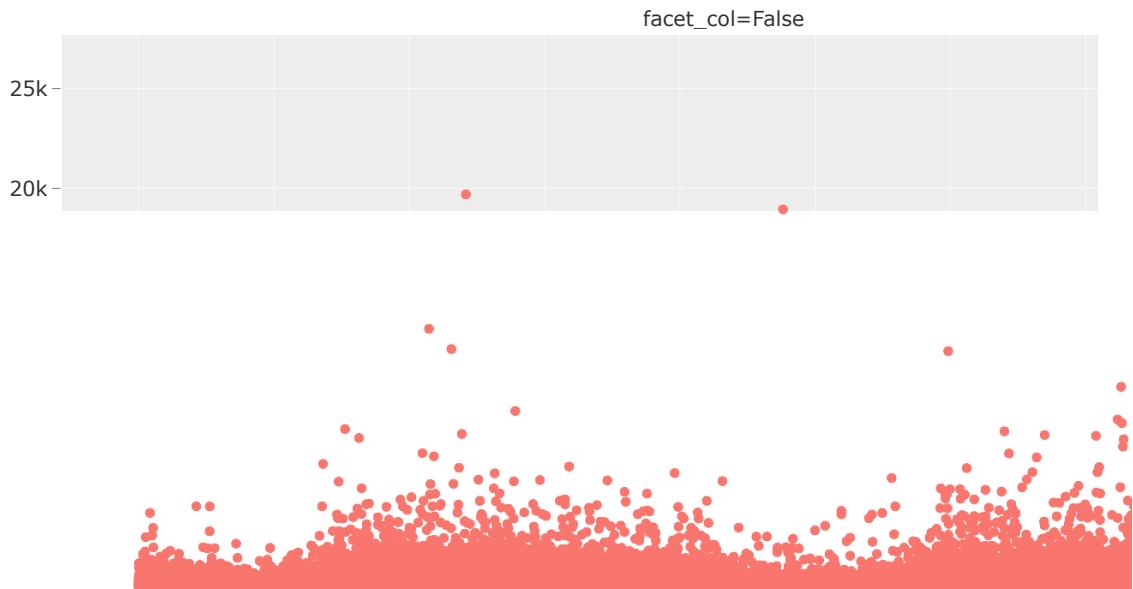
Class frequencies after over-sampling via imbalanced-learn library



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

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)

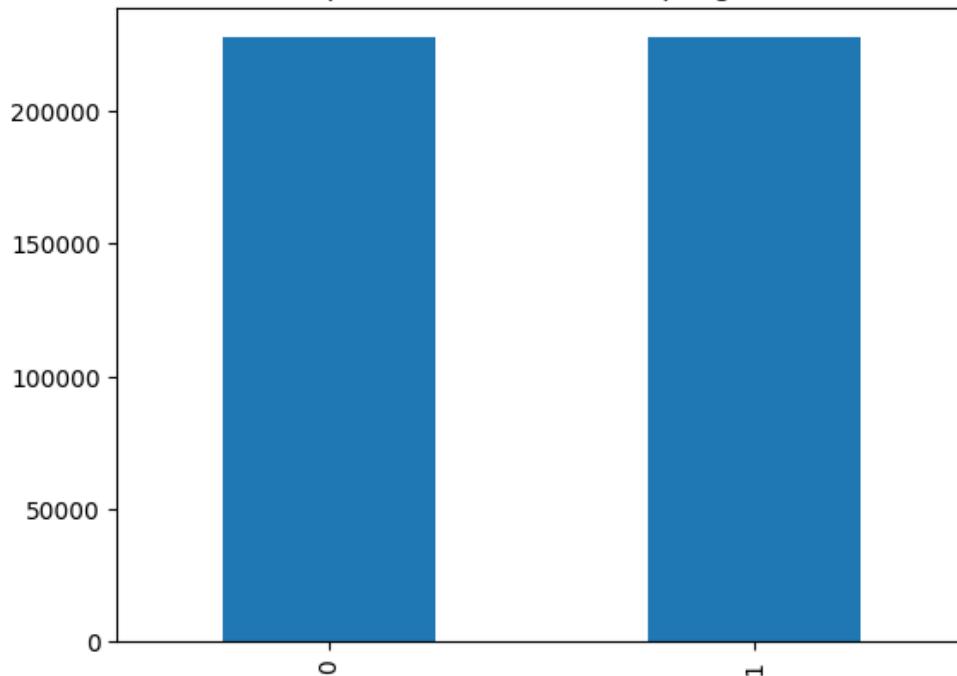
X_train_over_smote = data_train_over_smote.drop('Class', axis = 1)
y_train_over_smote = data_train_over_smote['Class']

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'}>
```

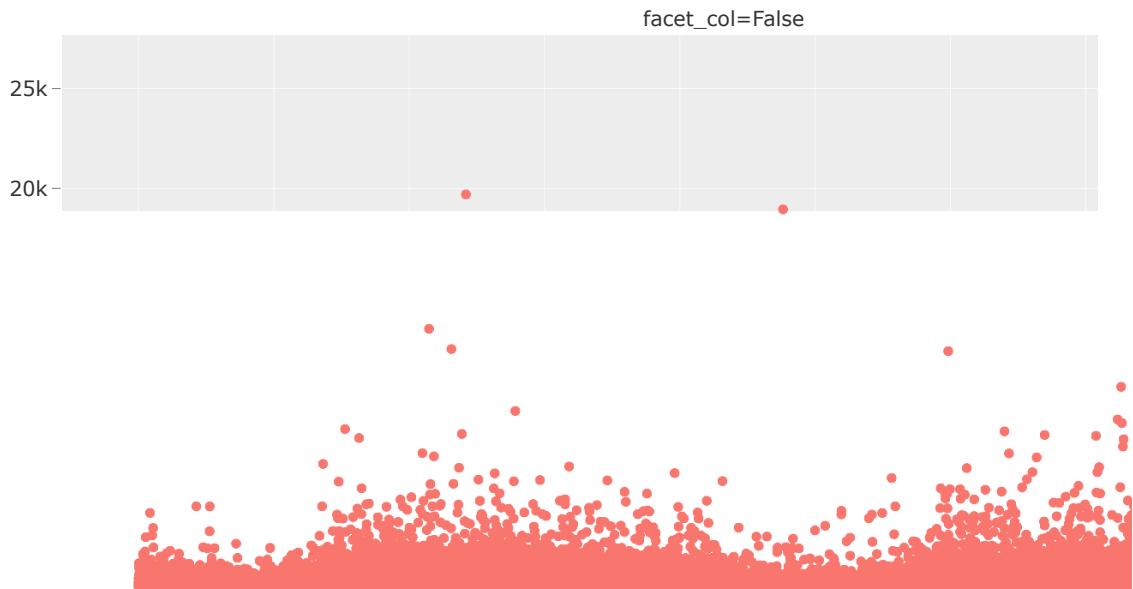
Class frequencies after over-sampling via SMOTE



```
In [ ]: # Valor vs. tempo para transações autênticas e fraudulentas no conjunto de treinamento após a amostragem com reposição via SMOTE

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

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



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

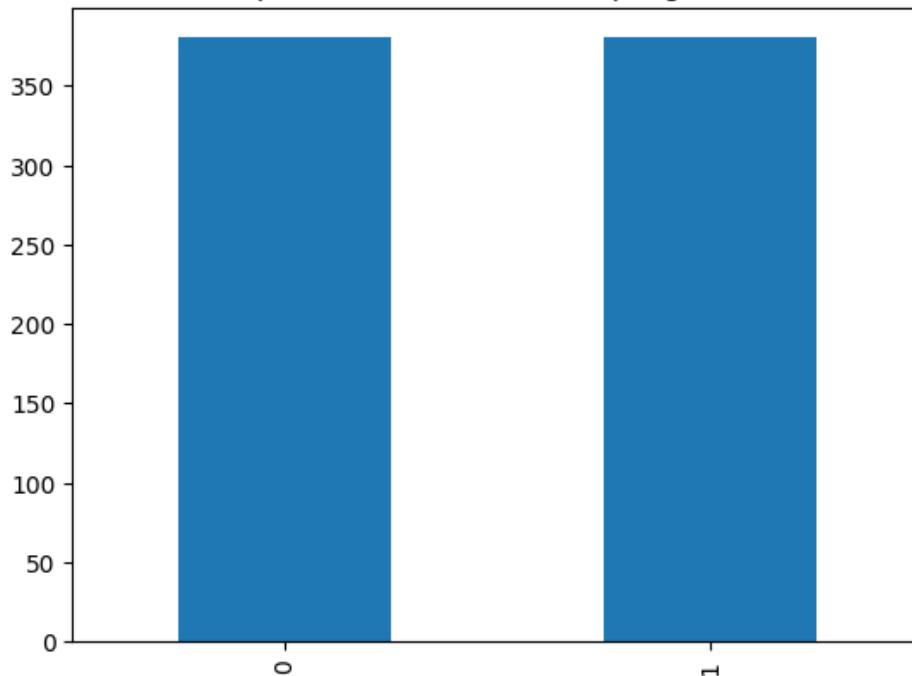
y_train_nm = pd.DataFrame(y_train_nm)
y_train_nm.columns = ['Class']
y_train_nm

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'}>
```

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 size
    plt.yticks(tick_marks_y, class_names, rotation = 'vertical')
    plt.xticks(tick_marks_x, class_names, rotation = 'horizontal')
    plt.ylabel('True label')
    plt.xlabel('Predicted label')
    plt.grid(False)
    plt.show()

    # Evaluation metrics

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

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

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

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

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

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

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

    G_mean = np.sqrt(recall * specificity)
```

```

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

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

# 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_mean]

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

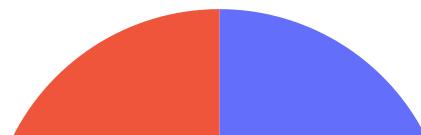
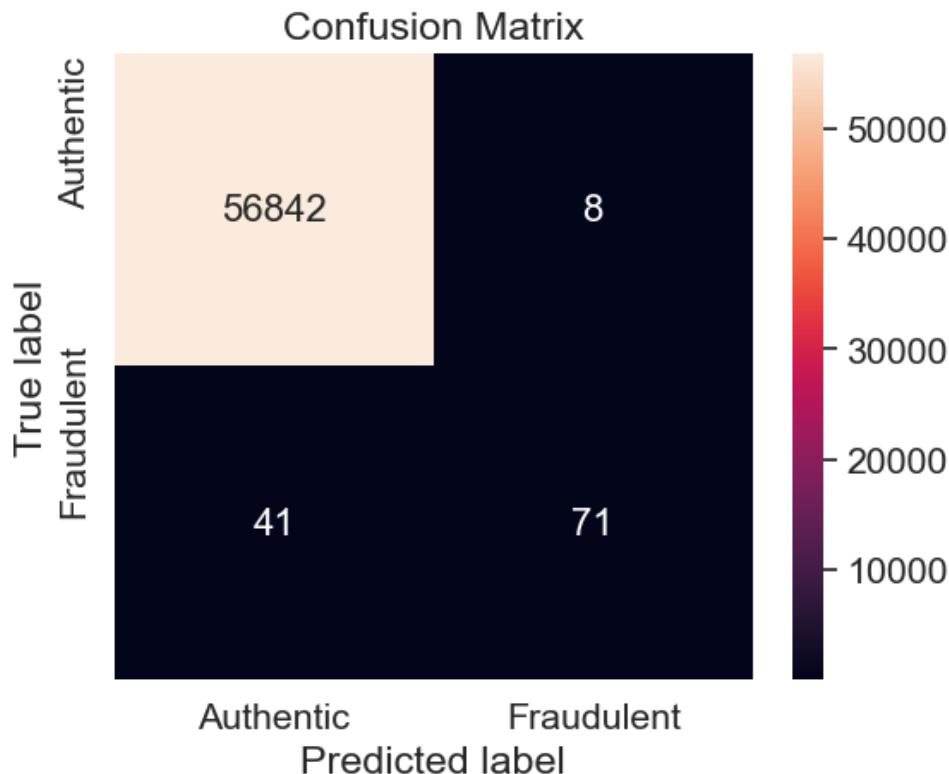
summary_logreg_unaltered = summary.copy()
summary_logreg_unaltered.set_index('Metric')

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

summary_logreg_unaltered_index = summary_logreg_unaltered_extended.T
summary_logreg_unaltered_index.columns = summary_logreg_unaltered_index.columns.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)

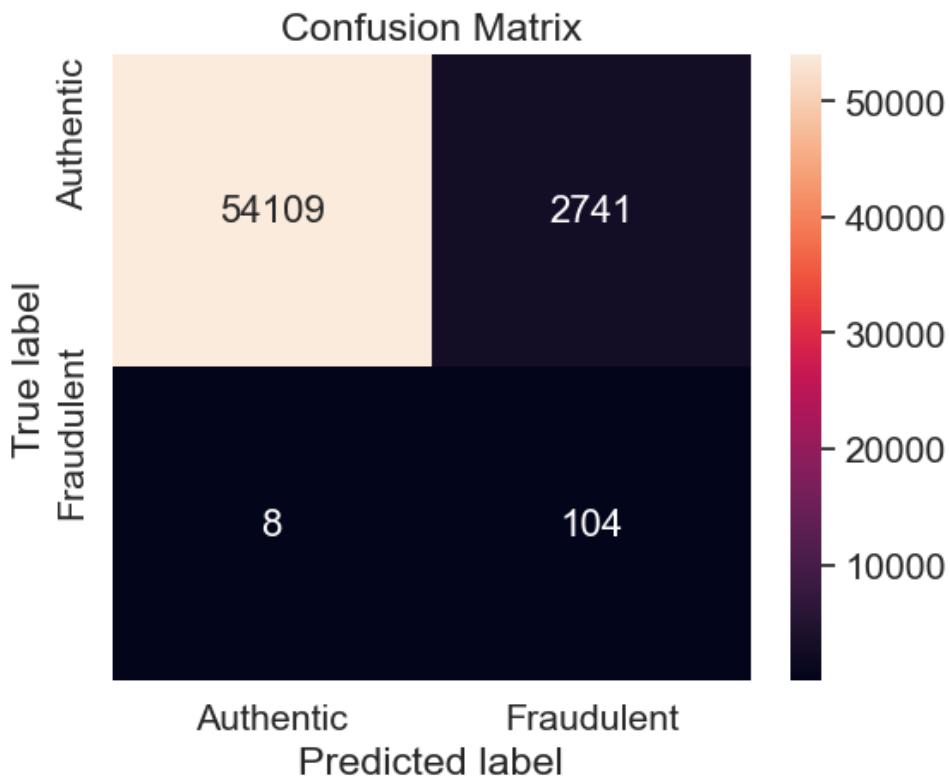
# Resumo das métricas de avaliação

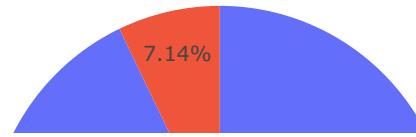
summary_logreg_under = summary
summary_logreg_under.set_index('Metric')

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

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





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	

Random over-sampling

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

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

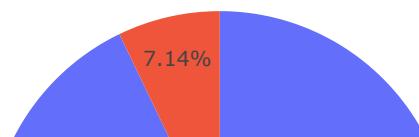
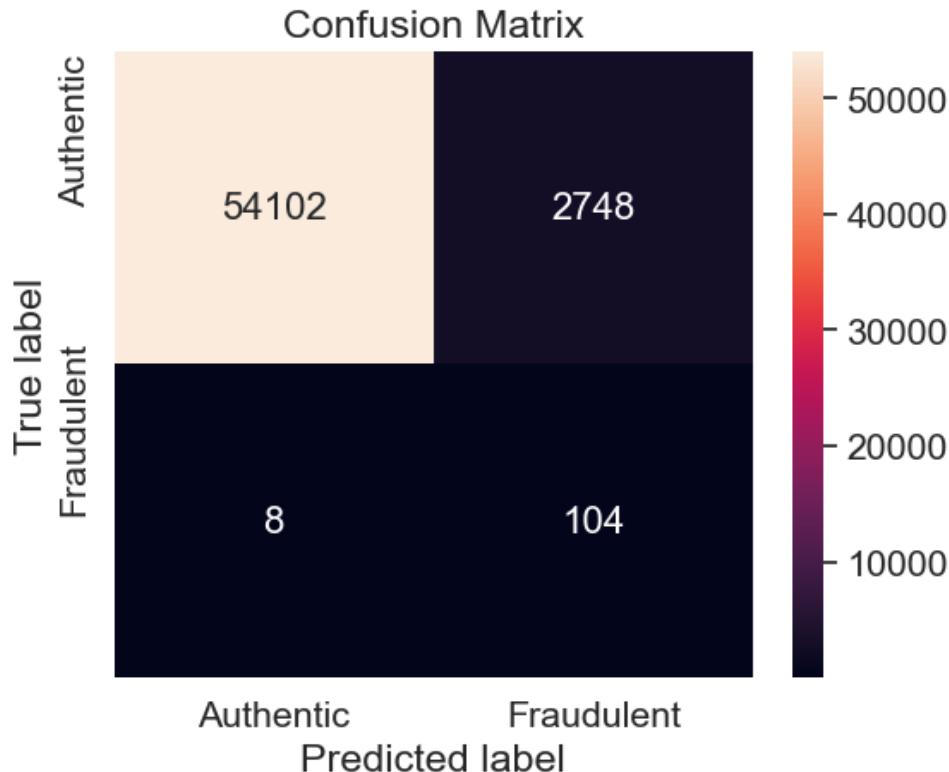
# Resumo das métricas de avaliação

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

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



Out[]:	Metric	MCC	F1-Score	F2-Score	Recall	Precision	FM index	Specificity	G-mean	F0.5-Score
Performance score		0.178796	0.070175	0.157576	0.928571	0.036466	0.184013	0.951662	0.940046	0.045139

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)
```

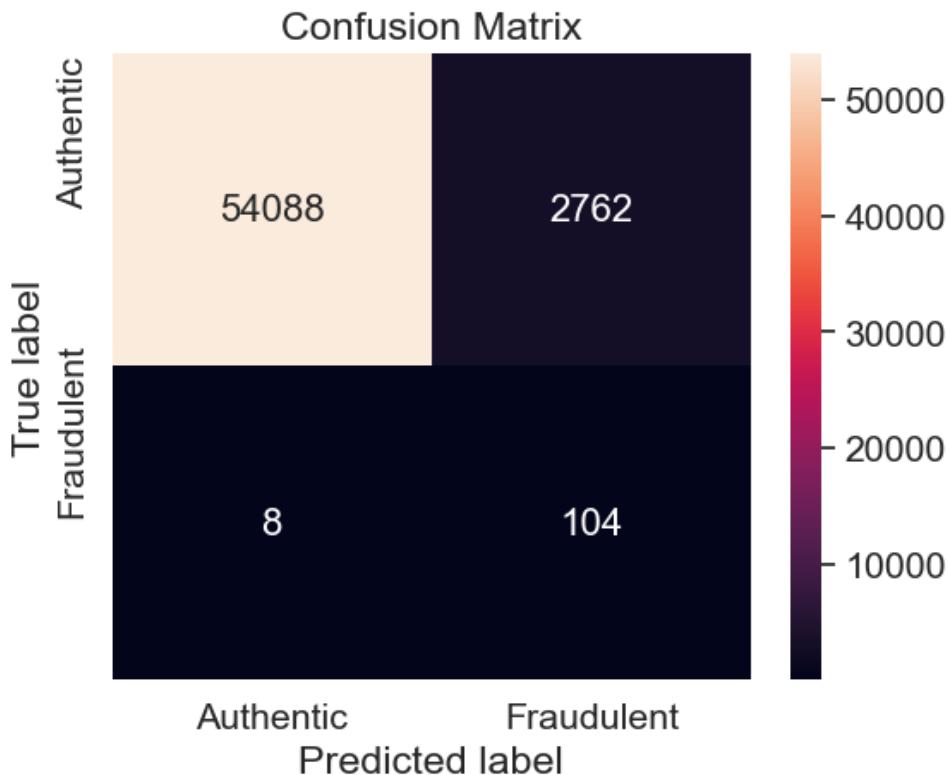
```
# Resumo das métricas de avaliação

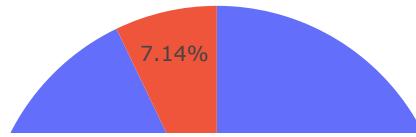
summary_logreg_under_imblearn = summary
summary_logreg_under_imblearn.set_index('Metric')

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 = True)
```





Out[]:	Metric	MCC	F1-Score	F2-Score	Recall	Precision	FM index	Specificity	G-mean	F0.5-Score
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)

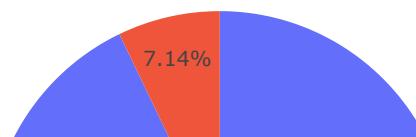
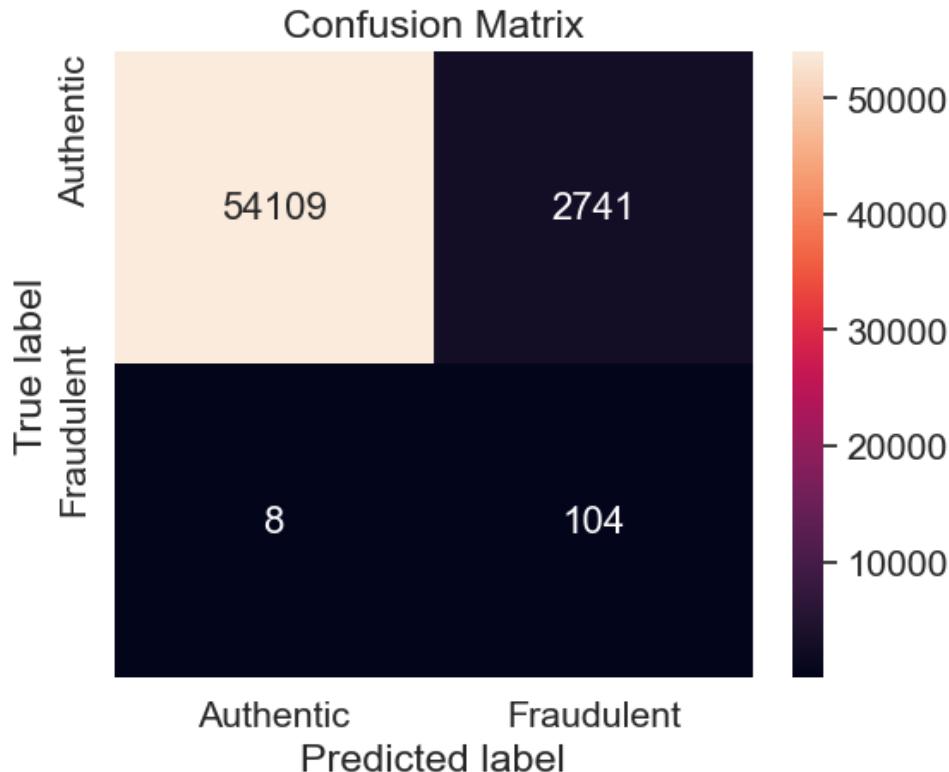
# Resumo das métricas de avaliação

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 = True)
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)
```

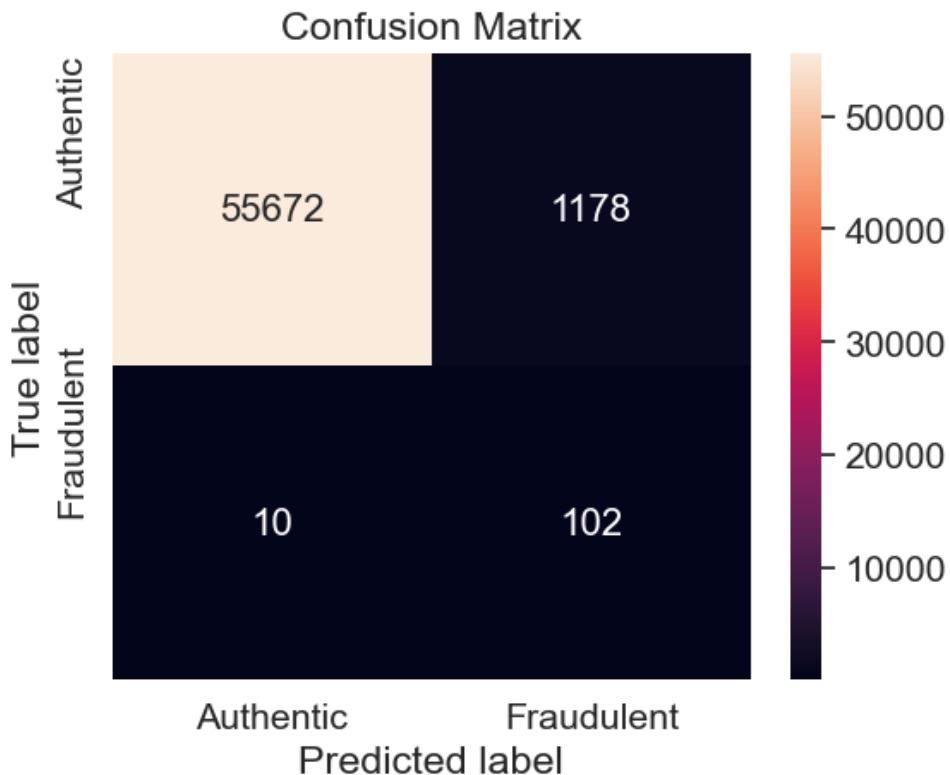
```
# Resumo das métricas de avaliação

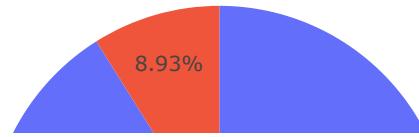
summary_logreg_over_smote = summary
summary_logreg_over_smote.set_index('Metric')

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', average_precision]
summary_logreg_over_smote_extended.loc[len(summary_logreg_over_smote_extended.index)] = ['ROC-AU', roc_auc]
summary_logreg_over_smote_extended.set_index('Metric')

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





Out[]:	Metric	MCC	F1-Score	F2-Score	Recall	Precision	FM index	Specificity	G-mean	F0.5-Score
Performance score	0.26601	0.146552	0.295139	0.910714	0.079687	0.269393	0.979279	0.944375	0.097477	

Under-sampling via NearMiss

```
In [ ]: # Elementos da matriz de confusão
classification(logreg, X_train_under_nm, y_train_under_nm, X_test, y_test)

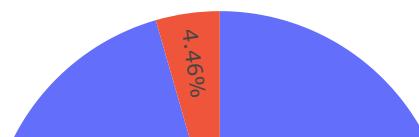
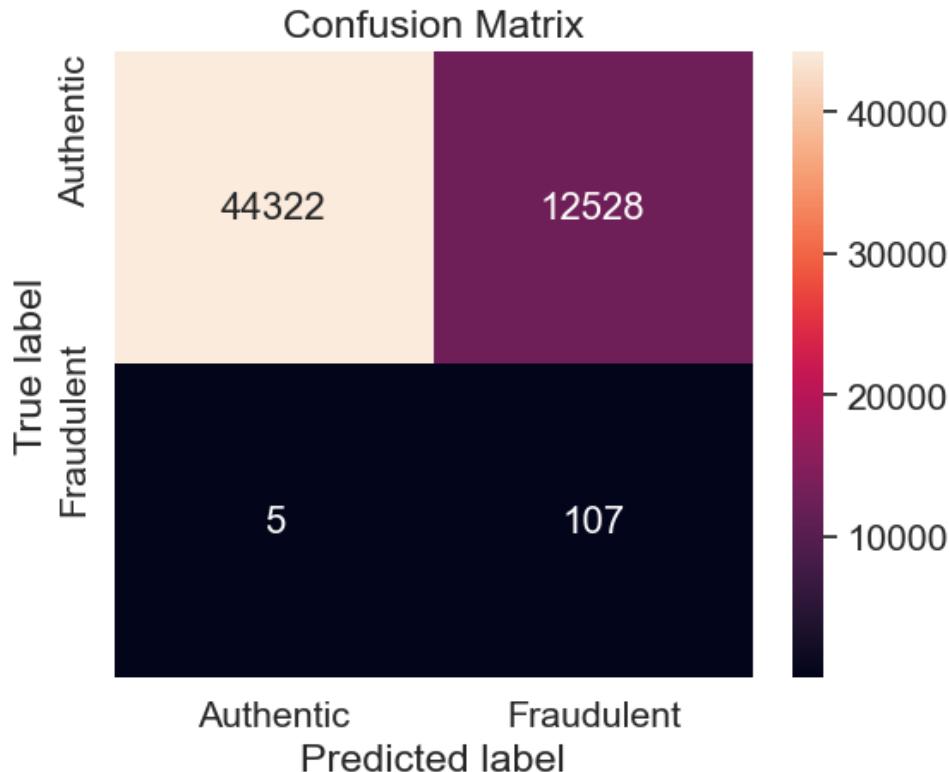
# Resumo das métricas de avaliação

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

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



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**.

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, summary_logreg_over_imblearn, summary_logreg_over_smote, summary_logreg_over_imblearn_smote]

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_18520\1296588952.py:8: FutureWarning:
Passing 'suffixes' which cause duplicate columns {'Performance score_x'} in the result is deprecated and will raise a MergeError in a future version.

C:\Users\Diones\AppData\Local\Temp\ipykernel_18520\1296588952.py:8: FutureWarning:
Passing 'suffixes' which cause duplicate columns {'Performance score_x'} in the result is deprecated and will raise a MergeError in a future version.

	Unaltered	RUS	ROS	RUS-IL	ROS-IL	SMOTE	NM
Metric							
MCC	0.754420	0.179030	0.178796	0.178332	0.179030	0.266010	0.078367
F1-Score	0.743455	0.070342	0.070175	0.069846	0.070342	0.146552	0.016788
F2-Score	0.673624	0.157911	0.157576	0.156910	0.157911	0.295139	0.040893
Recall	0.633929	0.928571	0.928571	0.928571	0.928571	0.910714	0.955357
Precision	0.898734	0.036555	0.036466	0.036288	0.036555	0.079687	0.008469
FM index	0.754807	0.184240	0.184013	0.183563	0.184240	0.269393	0.089947
Specificity	0.999859	0.951785	0.951662	0.951416	0.951785	0.979279	0.779631
G-mean	0.796140	0.940107	0.940046	0.939924	0.940107	0.944375	0.863033
F0.5-Score	0.829439	0.045249	0.045139	0.044921	0.045249	0.097477	0.010562
Accuracy	0.999140	0.951740	0.951617	0.951371	0.951740	0.979144	0.779976

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

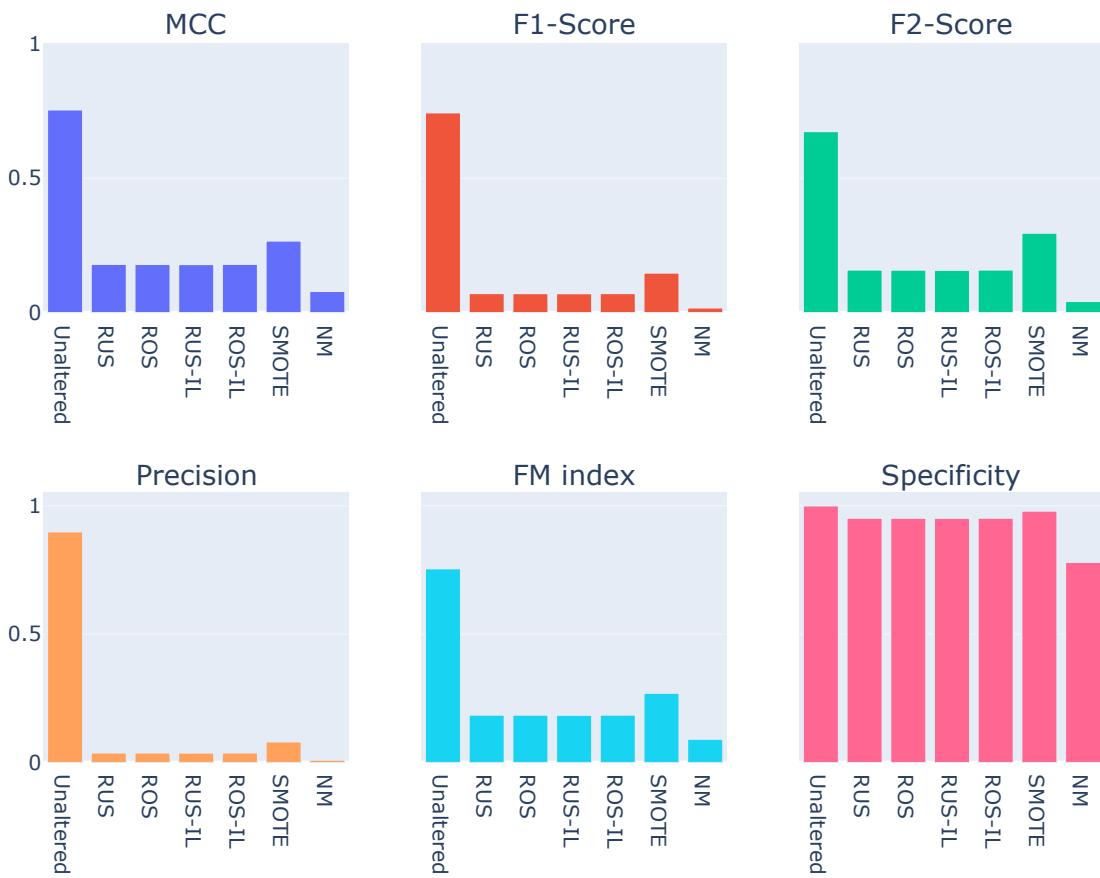
def summary_visual(summary_model):
    fig1 = make_subplots(rows = 2, cols = 4, shared_yaxes = True, subplot_titles = EvalMetricLabel)

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

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

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

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)

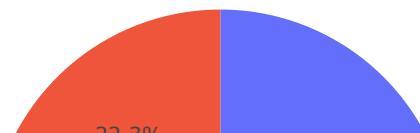
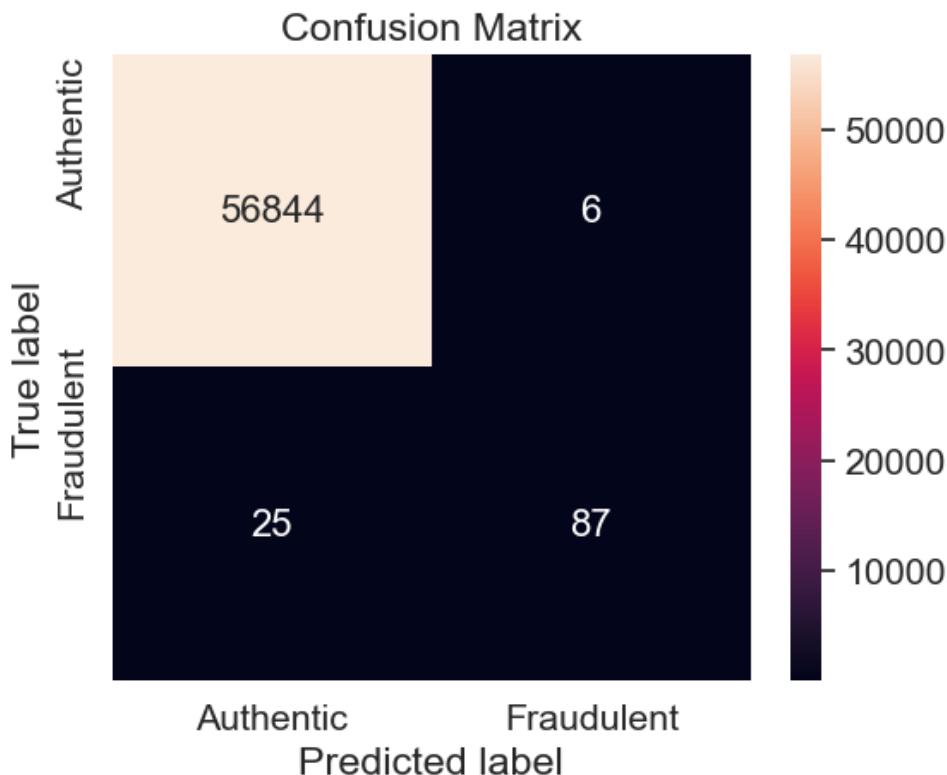
# Resumo das métricas de avaliação

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	Ace
Performance score		0.852191	0.84878	0.804067	0.776786	0.935484	0.85245	0.999894	0.881308	0.89876	0.5

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)

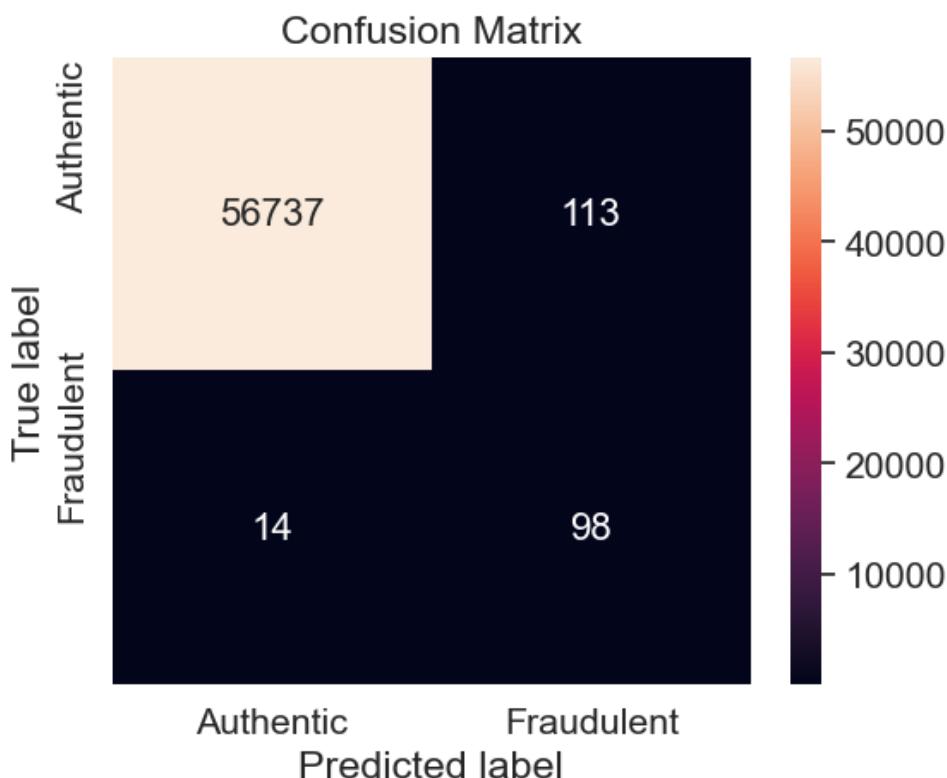
# Resumo das métricas de avaliação

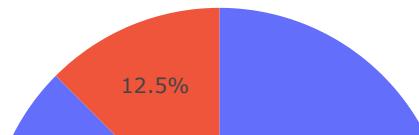
summary_knn_under = summary
summary_knn_under.set_index('Metric')

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	Accu
	Performance score	0.6366	0.606811	0.743551	0.875	0.464455	0.637494	0.998012	0.934484	0.512552	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)

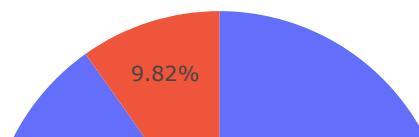
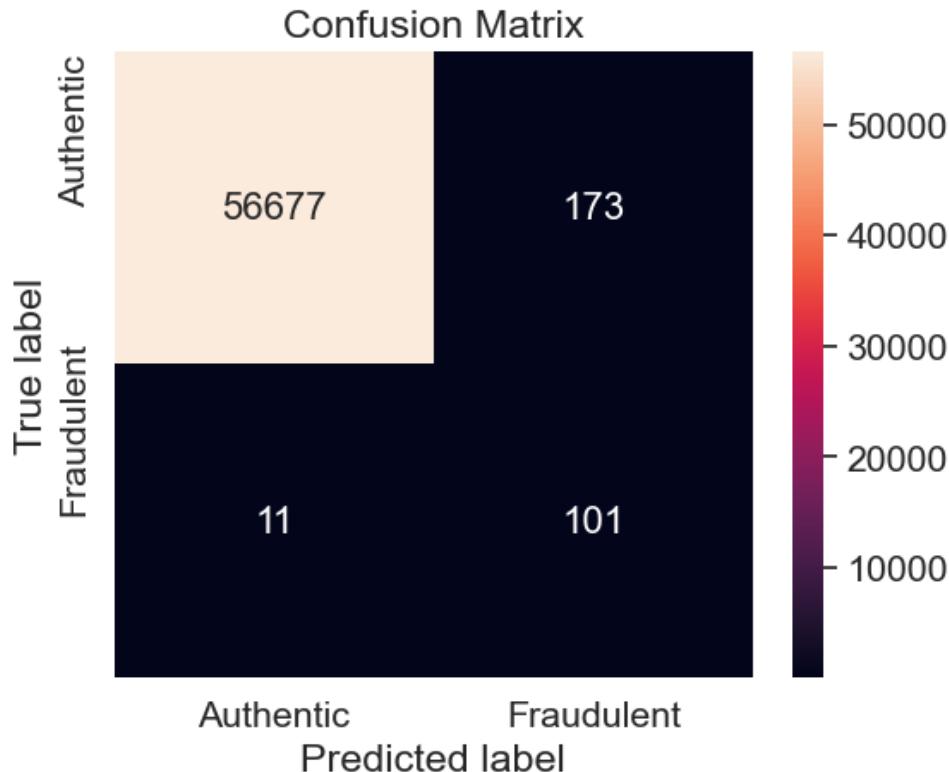
# Resumo das métricas de avaliação

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.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
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)

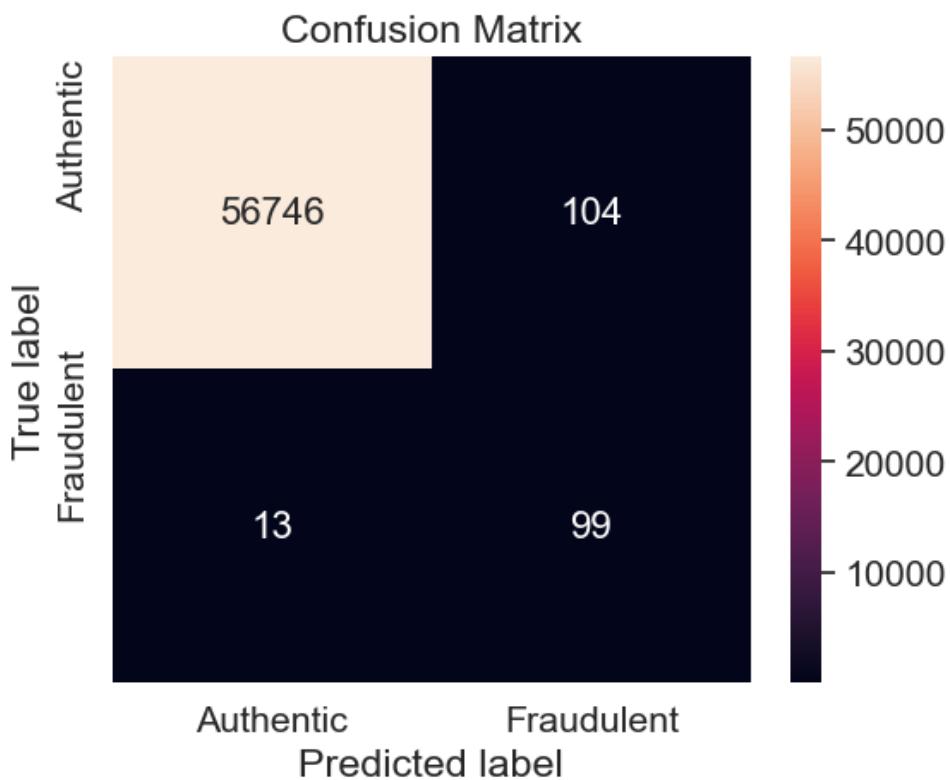
# Resumo das métricas de avaliação

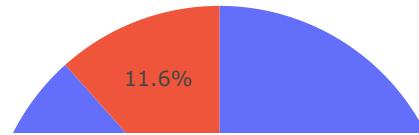
summary_knn_under_imblearn = summary
summary_knn_under_imblearn.set_index('Metric')

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

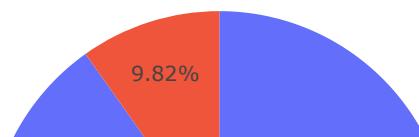
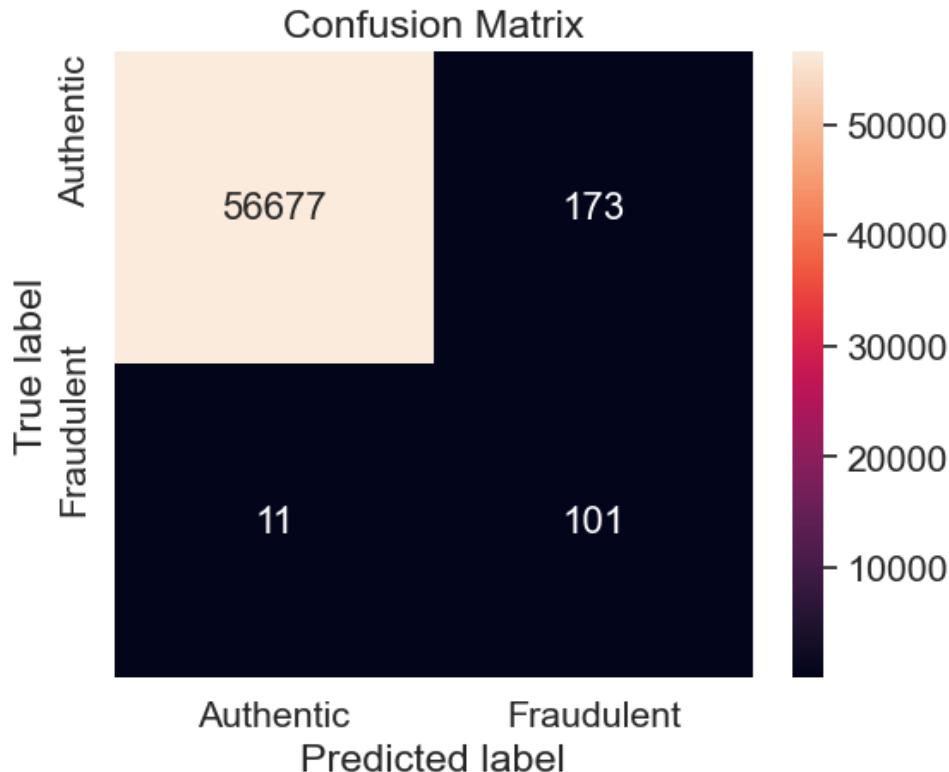
# Resumo das métricas de avaliação

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.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

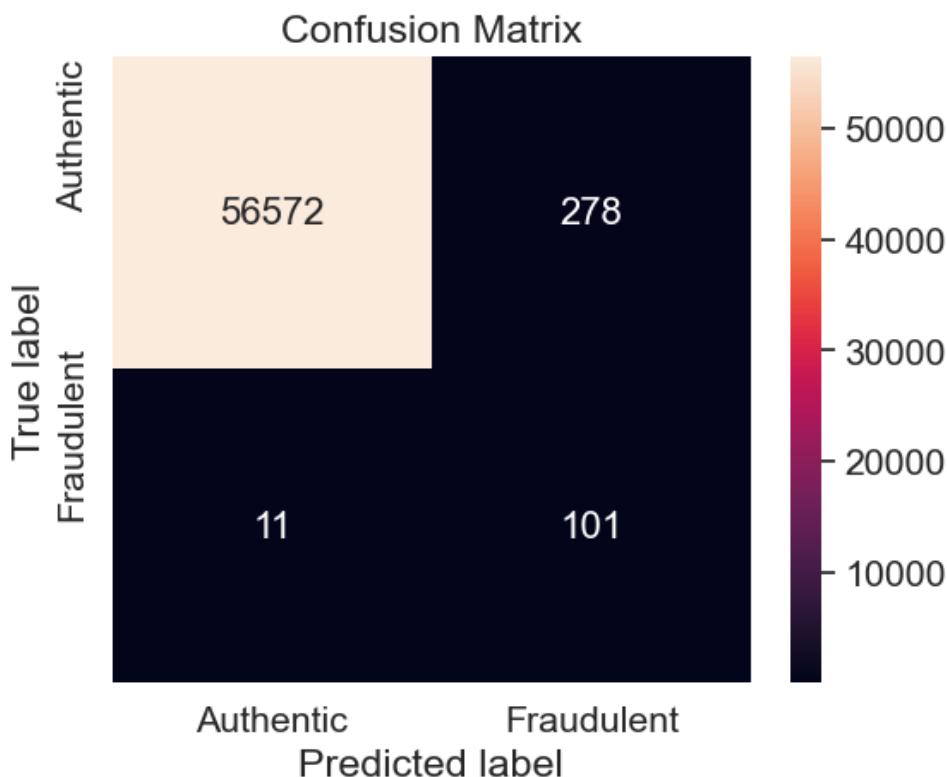
classification(knn, X_train_over_smote_scaled_minmax, y_train_over_smote, X_test_scaled_minmax,
# Resumo das métricas de avaliação

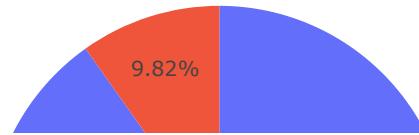
summary_knn_over_smote = summary
summary_knn_over_smote.set_index('Metric')

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
	Performance score	0.488713	0.411405	0.610641	0.901786	0.266491	0.490222	0.99511	0.947299	0.310197

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

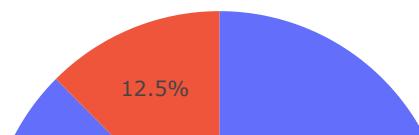
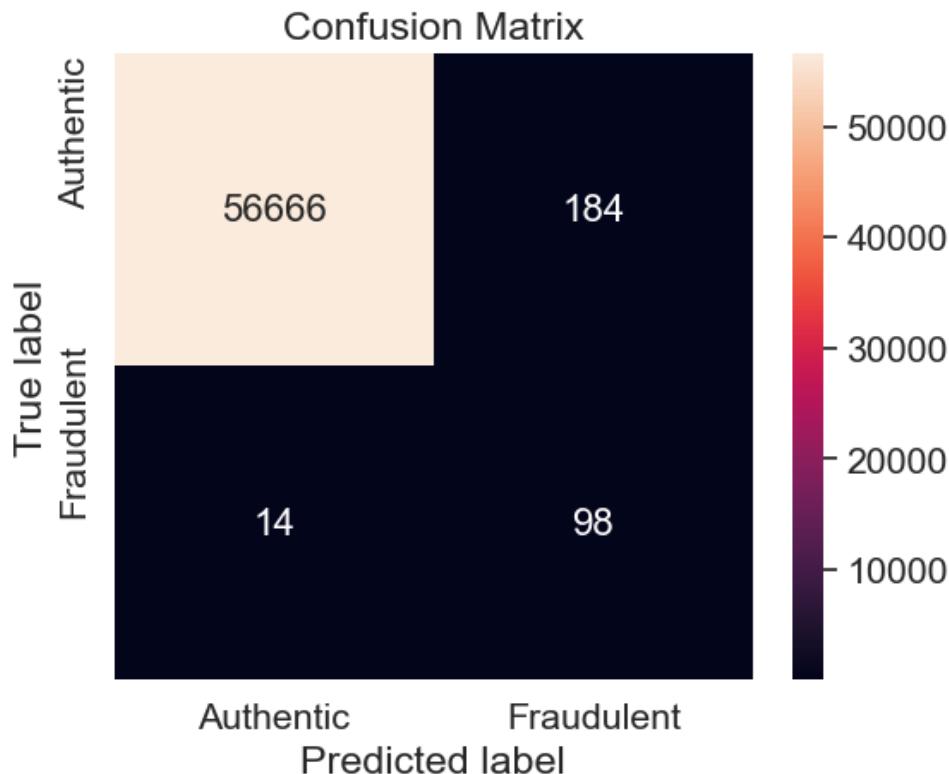
# Resumo das métricas de avaliação

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_auc]
summary_knn_under_nm_extended.set_index('Metric')

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



```
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		0.550216	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\Diônes\AppData\Local\Temp\ipykernel_18520\1710798844.py:8: FutureWarning:

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

C:\Users\Diônes\AppData\Local\Temp\ipykernel_18520\1710798844.py:8: FutureWarning:

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

Out[]:

Metric	Unaltered	RUS	ROS	RUS-IL	ROS-IL	SMOTE	NM
MCC	0.852191	0.636600	0.575425	0.655732	0.575425	0.488713	0.550216
F1-Score	0.848780	0.606811	0.523316	0.628571	0.523316	0.411405	0.497462
F2-Score	0.804067	0.743551	0.699446	0.760369	0.699446	0.610641	0.671233
Recall	0.776786	0.875000	0.901786	0.883929	0.901786	0.901786	0.875000
Precision	0.935484	0.464455	0.368613	0.487685	0.368613	0.266491	0.347518
FM index	0.852450	0.637494	0.576550	0.656566	0.576550	0.490222	0.551433
Specificity	0.999894	0.998012	0.996957	0.998171	0.996957	0.995110	0.996763
G-mean	0.881308	0.934484	0.948178	0.939314	0.948178	0.947299	0.933899
F0.5-Score	0.898760	0.512552	0.418046	0.535714	0.418046	0.310197	0.395161
Accuracy	0.999456	0.997770	0.996770	0.997946	0.996770	0.994926	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

```
# Elementos da matriz de confusão

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

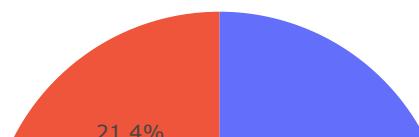
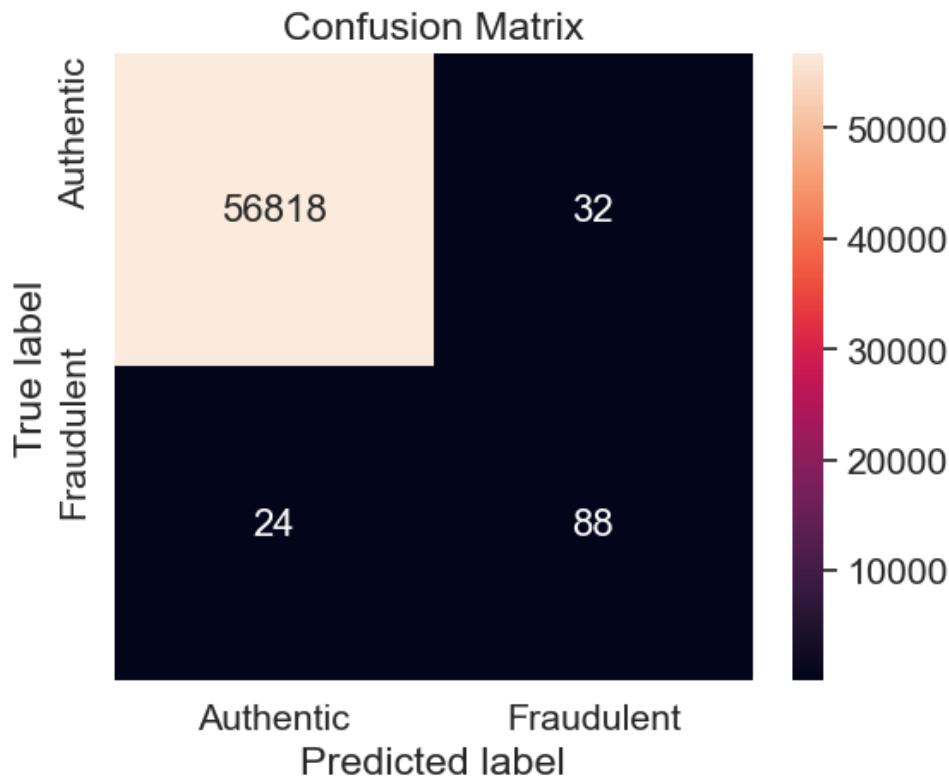
# Resumo das métricas de avaliação

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_auc]
summary_dt_unaltered_extended.set_index('Metric')
```

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



Out[]:	Metric	MCC	F1-Score	F2-Score	Recall	Precision	FM index	Specificity	G-mean	F0.5-Score
Performance score	0.758582	0.758621	0.774648	0.785714	0.733333	0.759072	0.999437	0.886156	0.743243	

Random under-sampling

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

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

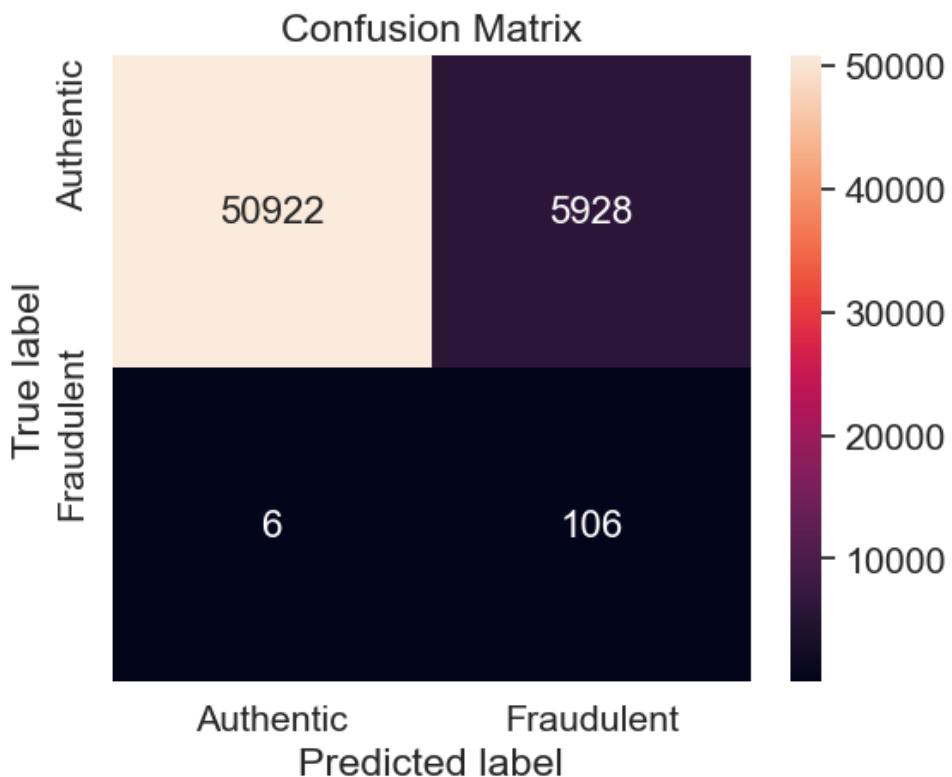
# Resumo das métricas de avaliação

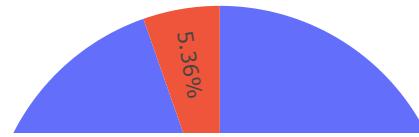
summary_dt_under = summary
summary_dt_under.set_index('Metric')

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	0.121223	0.034494	0.081765	0.946429	0.017567	0.128942	0.895726	0.920728	0.021857

Random over-sampling

```
In [ ]: # Elementos da matriz de confusão
classification(dt, X_train_over, y_train_over, X_test, y_test)

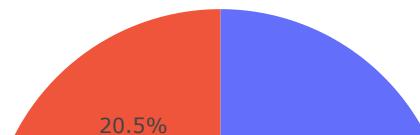
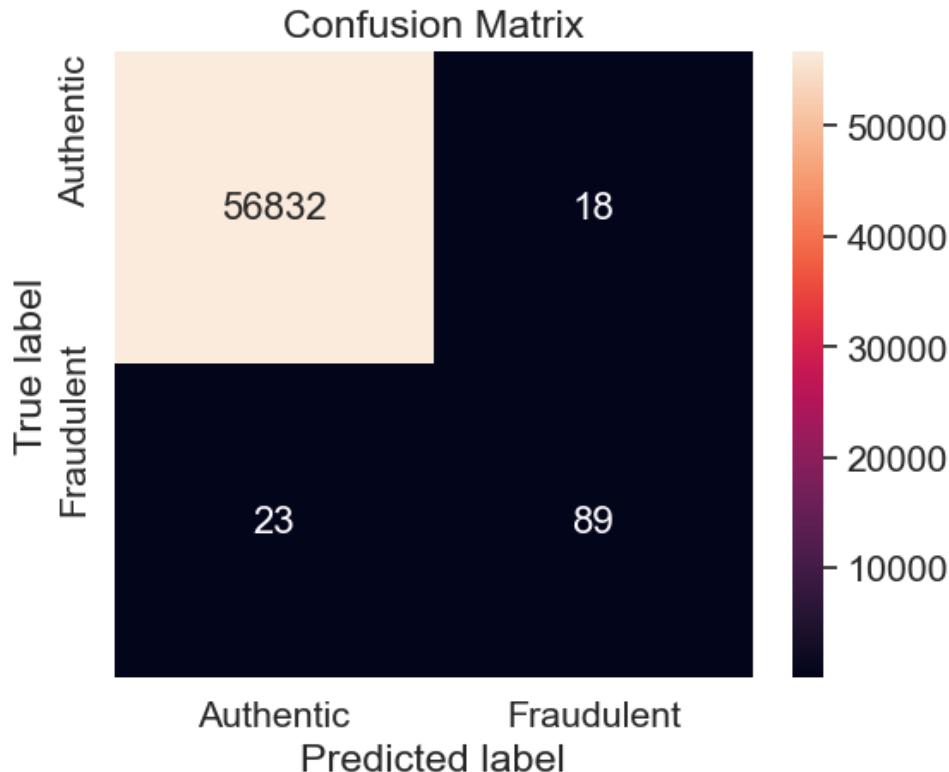
# Resumo das métricas de avaliação

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
Performance score		0.812638	0.812785	0.801802	0.794643	0.831776	0.812997	0.999683	0.891286	0.824074

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)
```

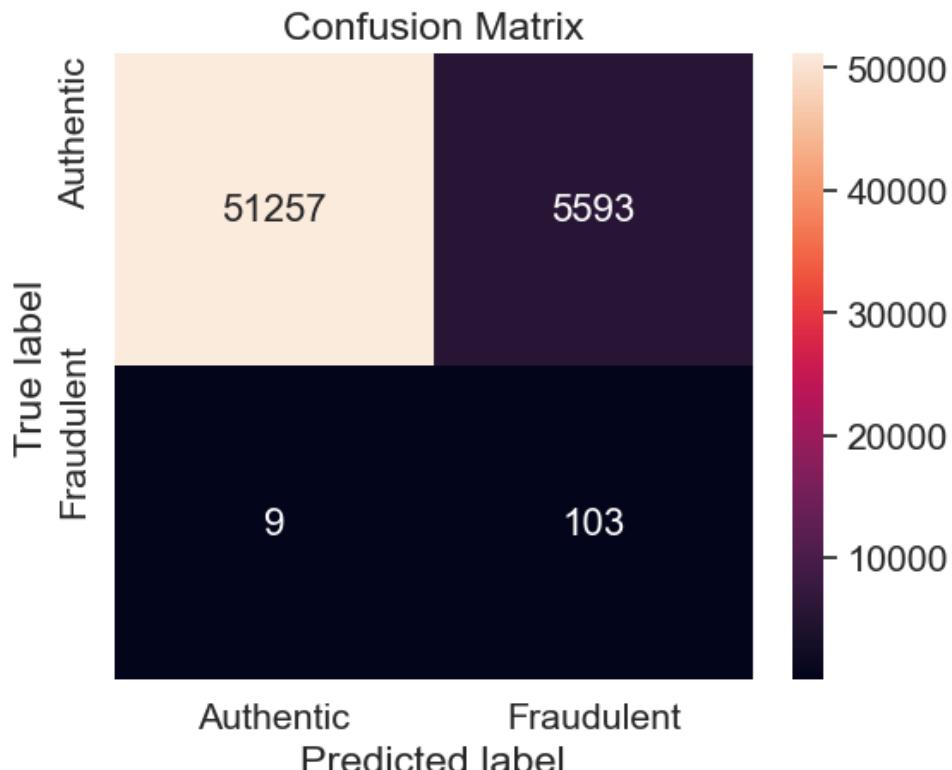
```
# Resumo das métricas de avaliação

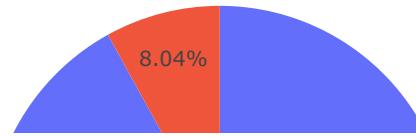
summary_dt_under_imblearn = summary
summary_dt_under_imblearn.set_index('Metric')

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
	Performance score	0.121271	0.035468	0.083822	0.919643	0.018083	0.128956	0.901618	0.910586	0.022493

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)

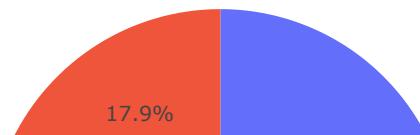
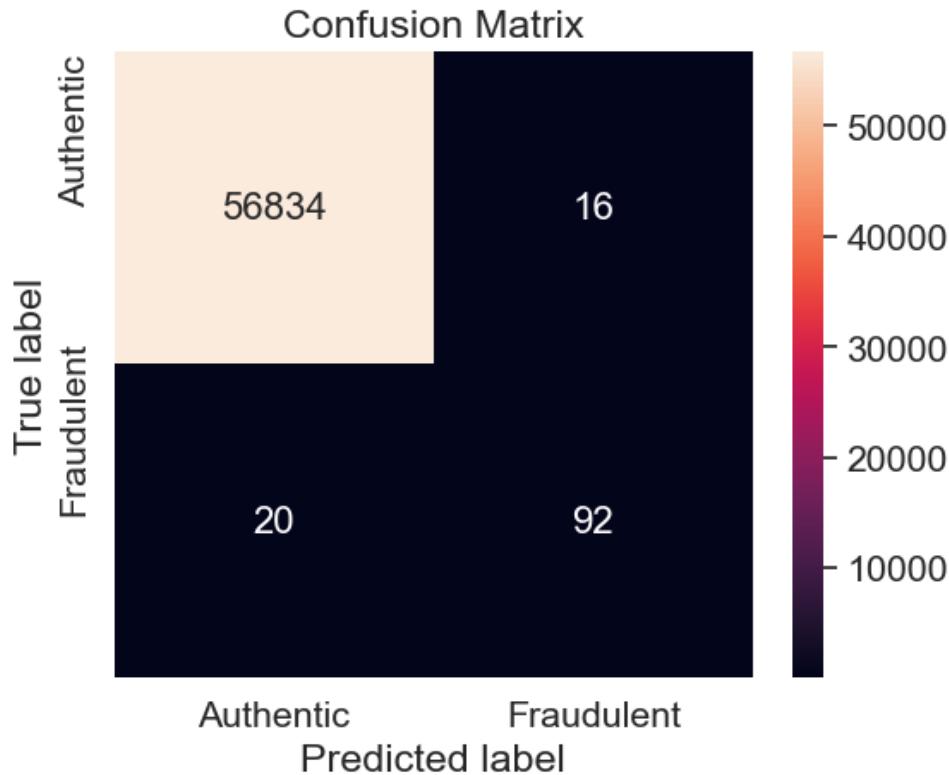
# Resumo das métricas de avaliação

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
Performance score		0.836186	0.836364	0.827338	0.821429	0.851852	0.836502	0.999719	0.906199	0.845588

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)
```

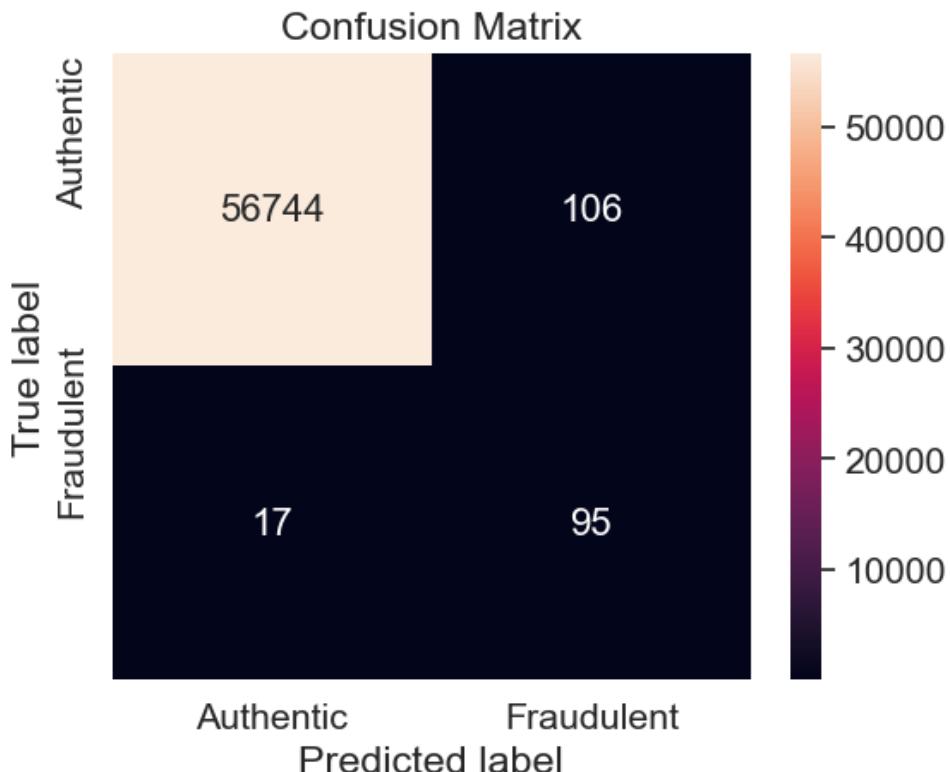
```
# Resumo das métricas de avaliação

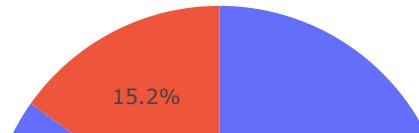
summary_dt_over_smote = summary
summary_dt_over_smote.set_index('Metric')

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_auc]
summary_dt_over_smote_extended.set_index('Metric')

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





Out[]:	Metric	MCC	F1-Score	F2-Score	Recall	Precision	FM index	Specificity	G-mean	F0.5-Score
	Performance score	0.632268	0.607029	0.731895	0.848214	0.472637	0.633165	0.998135	0.920126	0.518559

Under-sampling via NearMiss

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

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

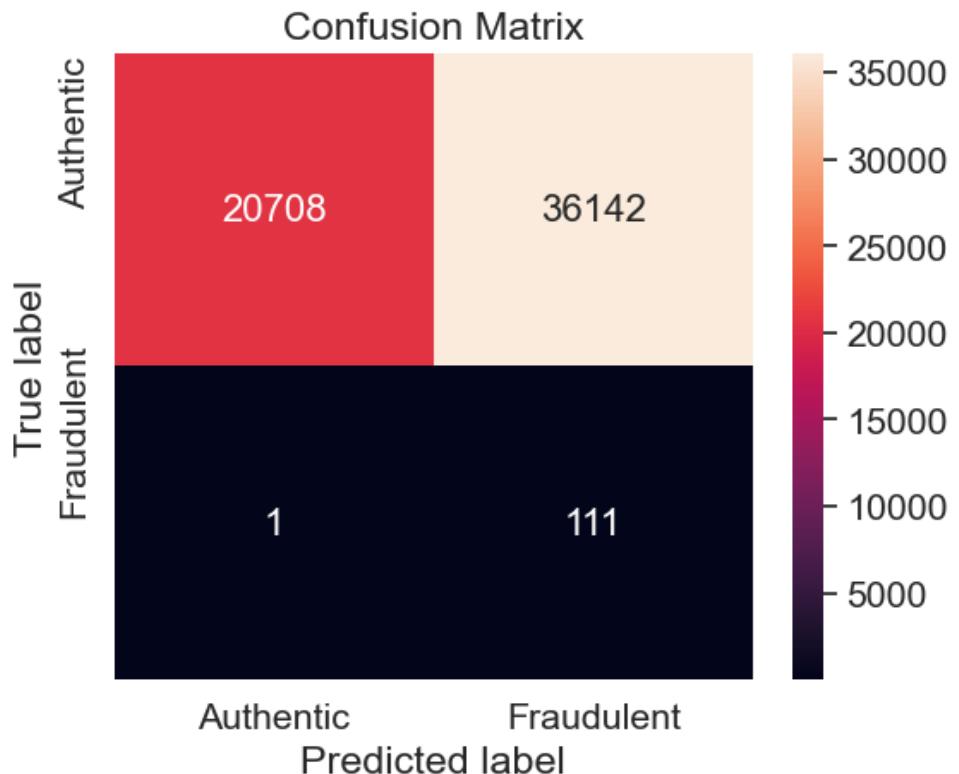
# Resumo das métricas de avaliação

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	0.032723	0.006105	0.015122	0.991071	0.003062	0.055086	0.364257	0.600837	0.003824	

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')

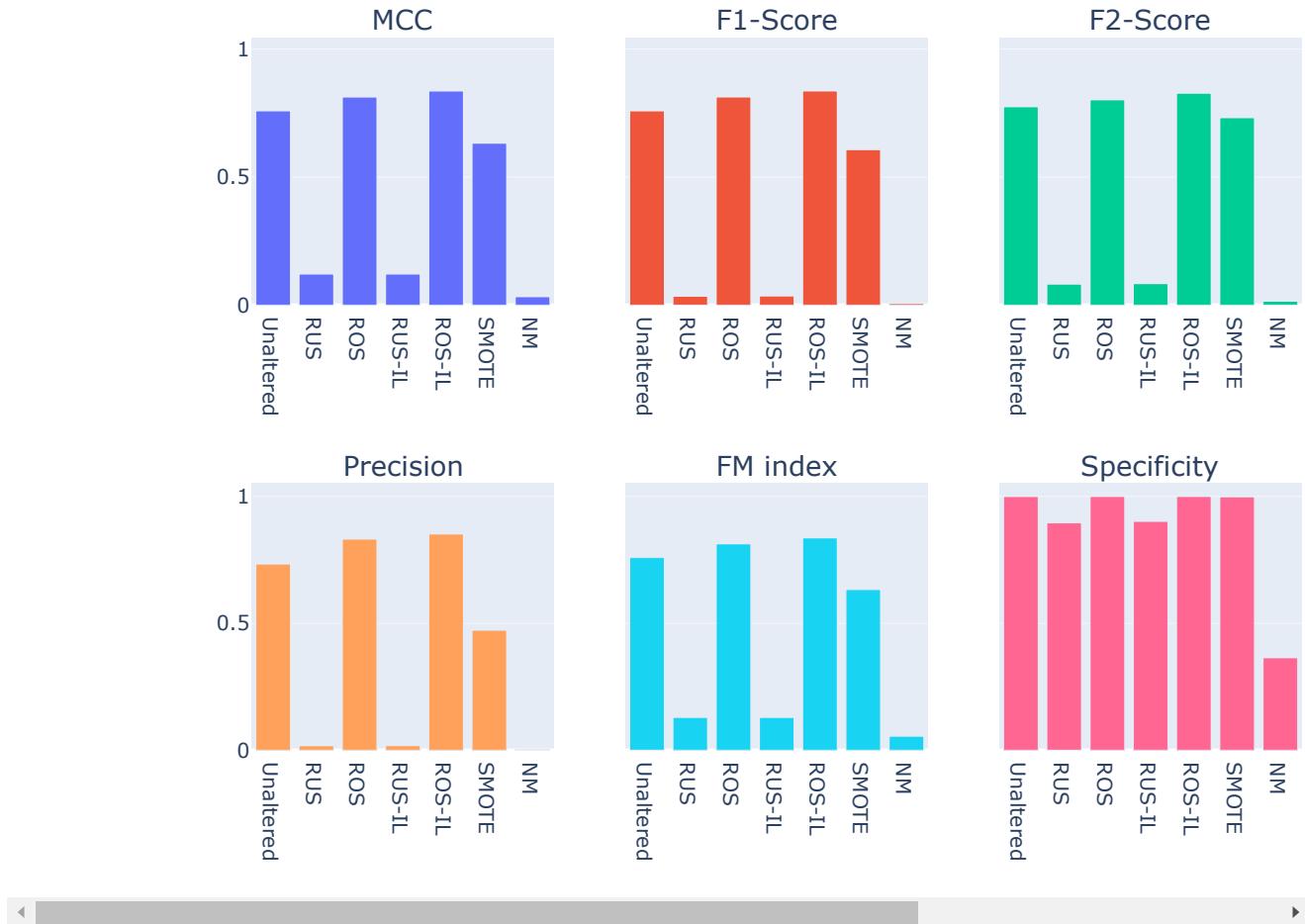
summary_dt.columns = TrainingSetsMetric
summary_dt.set_index('Metric', inplace = True)
summary_dt
```

C:\Users\Diônes\AppData\Local\Temp\ipykernel_18520\1435390414.py:9: FutureWarning:
 Passing 'suffixes' which cause duplicate columns {'Performance score_x'} in the result is deprecated and will raise a MergeError in a future version.

C:\Users\Diônes\AppData\Local\Temp\ipykernel_18520\1435390414.py:9: FutureWarning:
 Passing 'suffixes' which cause duplicate columns {'Performance score_x'} in the result is deprecated and will raise a MergeError in a future version.

	Unaltered	RUS	ROS	RUS-IL	ROS-IL	SMOTE	NM
Metric							
MCC	0.758582	0.121223	0.812638	0.121271	0.836186	0.632268	0.032723
F1-Score	0.758621	0.034494	0.812785	0.035468	0.836364	0.607029	0.006105
F2-Score	0.774648	0.081765	0.801802	0.083822	0.827338	0.731895	0.015122
Recall	0.785714	0.946429	0.794643	0.919643	0.821429	0.848214	0.991071
Precision	0.733333	0.017567	0.831776	0.018083	0.851852	0.472637	0.003062
FM index	0.759072	0.128942	0.812997	0.128956	0.836502	0.633165	0.055086
Specificity	0.999437	0.895726	0.999683	0.901618	0.999719	0.998135	0.364257
G-mean	0.886156	0.920728	0.891286	0.910586	0.906199	0.920126	0.600837
F0.5-Score	0.743243	0.021857	0.824074	0.022493	0.845588	0.518559	0.003824
Accuracy	0.999017	0.895825	0.999280	0.901654	0.999368	0.997841	0.365489

```
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

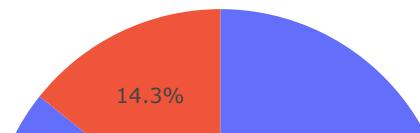
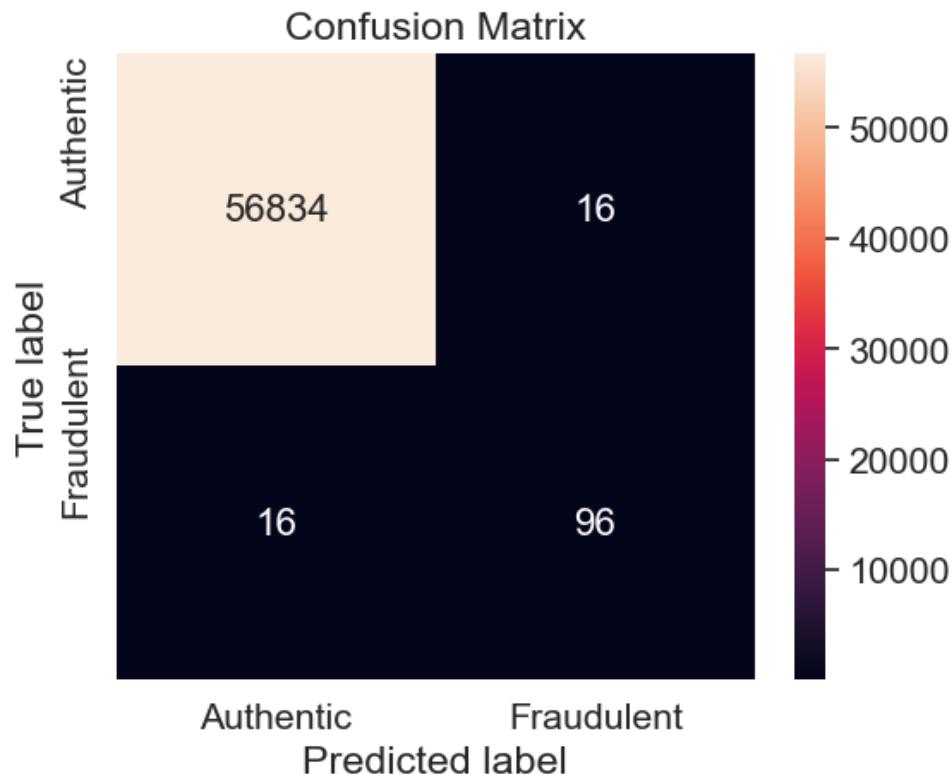
```
In [ ]: # Elementos da matriz de confusão
classification(svm_linear, X_train_scaled_minmax, y_train, X_test_scaled_minmax, y_test)

# 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 = True)
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
Performance score	0.856861	0.857143	0.857143	0.857143	0.857143	0.857143	0.999719	0.92569	0.857143	0.857143

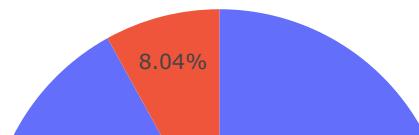
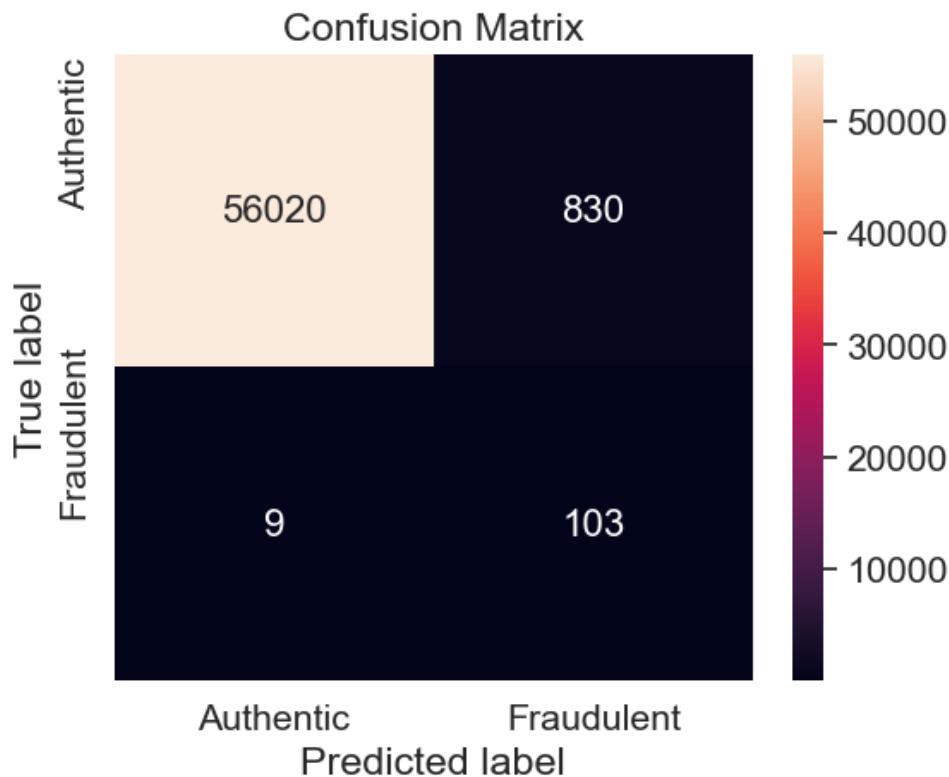
Random under-sampling

```
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.index
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	0.315861	0.197129	0.372918	0.919643	0.110397	0.318631	0.9854	0.951954	0.133975

Random over-sampling

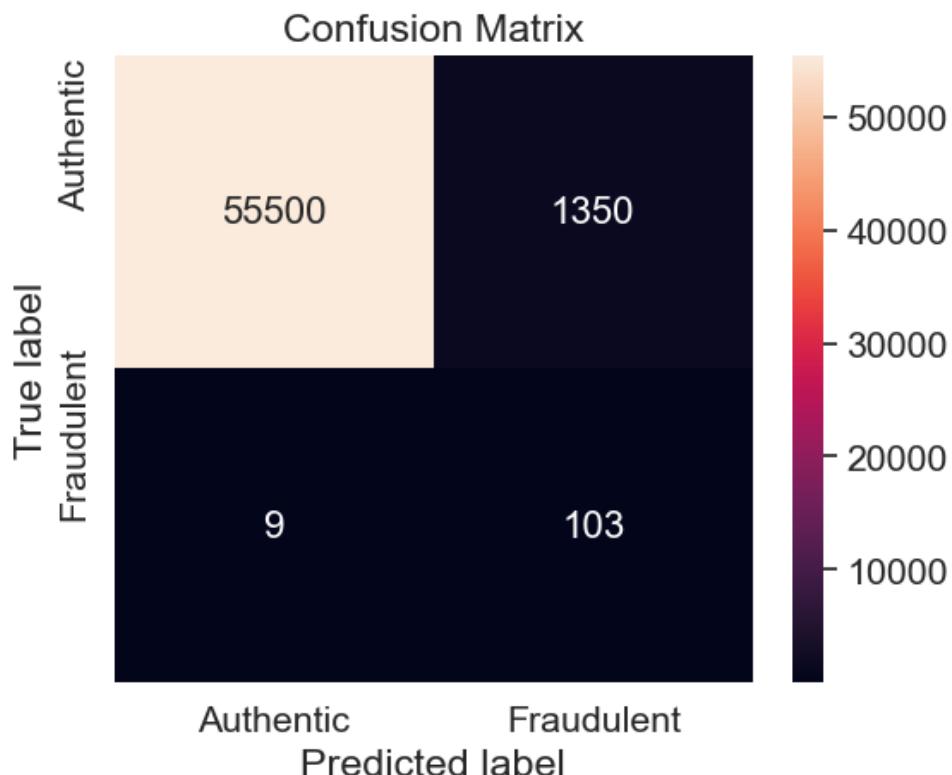
```
In [ ]: # Elementos da matriz de confusão

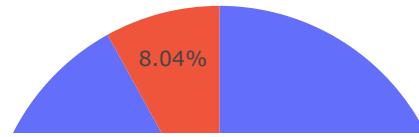
classification(svm_linear, X_train_over_scaled_minmax, y_train_over, X_test_scaled_minmax, y_test_over)

# 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	0.25172	0.131629	0.27091	0.919643	0.070888	0.255326	0.976253	0.947525	0.086935	0

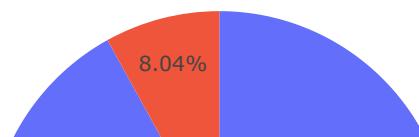
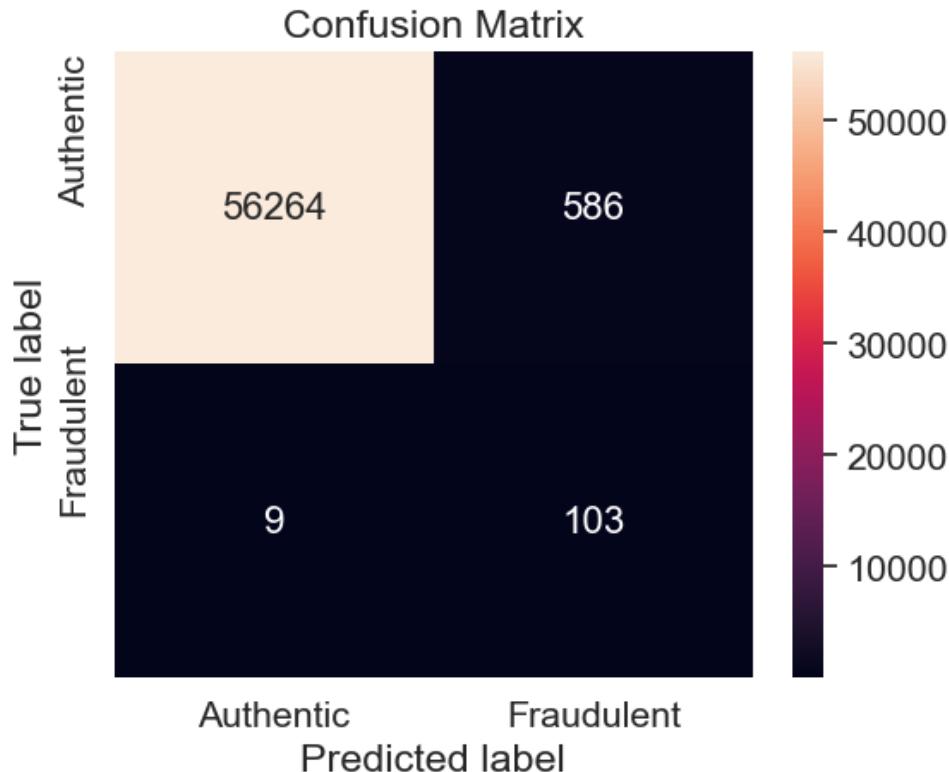
Random under-sampling with imbalanced-learning library

```
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

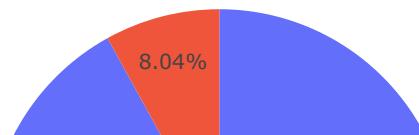
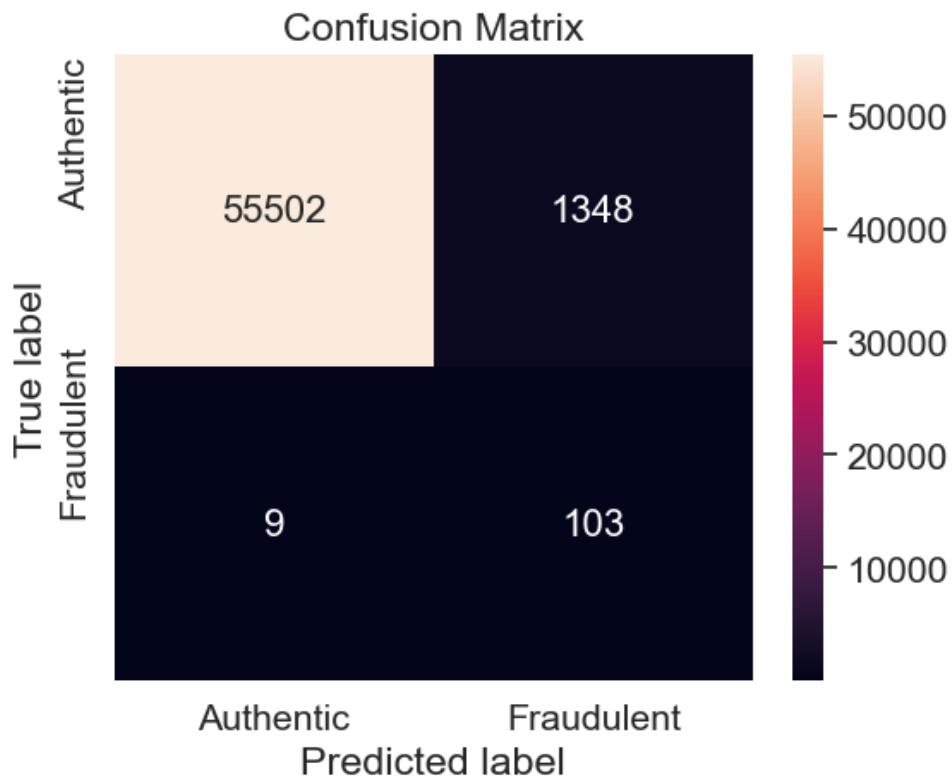
In []: # Elementos da matriz de confusão

```
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], inplace=True)
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)

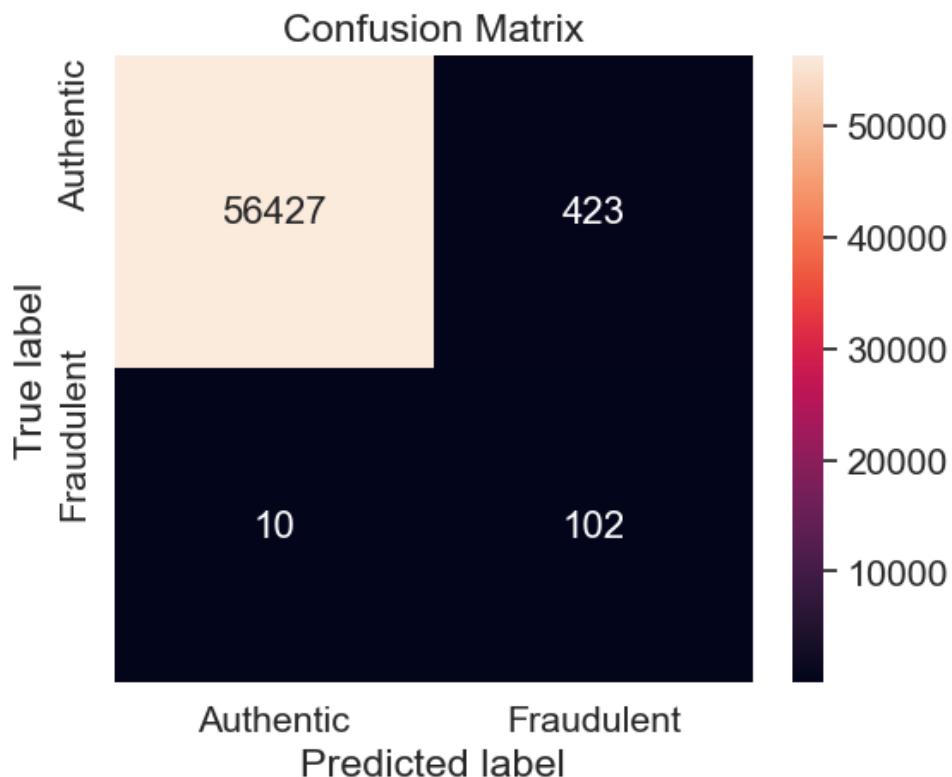
```
In [ ]: # Elementos da matriz de confusão

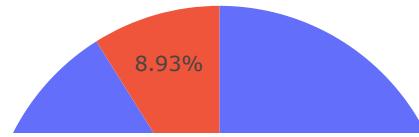
classification(svm_linear, X_train_over_smote_scaled_minmax, y_train_over_smote, X_test_scaled_m

# 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
	Performance score	0.418728	0.320251	0.524152	0.910714	0.194286	0.420641	0.992559	0.950757	0.230561

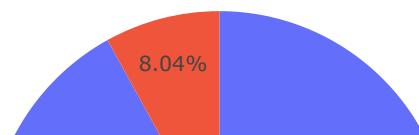
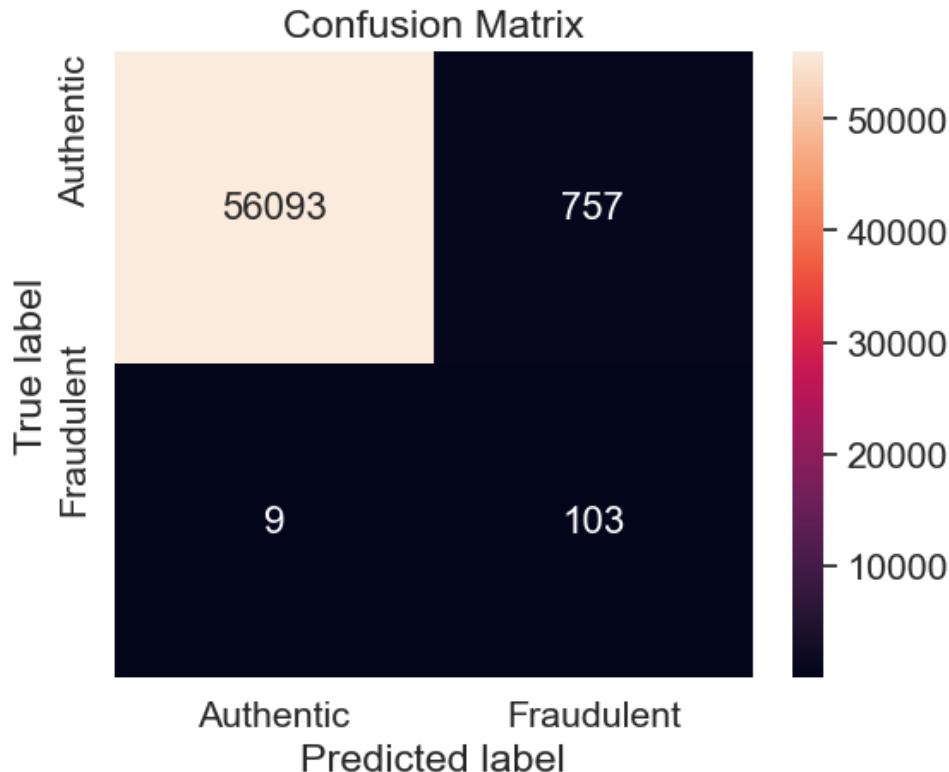
Under-sampling via NearMiss

```
In [ ]: # Elementos da matriz de confusão
classification(svm_linear, X_train_under_nm_scaled_minmax, y_train_under_nm, X_test_scaled_minmax)

# 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 = True)
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_18520\3997214041.py:9: FutureWarning:

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

C:\Users\Diones\AppData\Local\Temp\ipykernel_18520\3997214041.py:9: FutureWarning:

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

Out[]:

	Unaltered	RUS	ROS	RUS-IL	ROS-IL	SMOTE	NM
Metric							
MCC	0.856861	0.315861	0.251720	0.368501	0.251899	0.418728	0.329246
F1-Score	0.857143	0.197129	0.131629	0.257179	0.131798	0.320251	0.211934
F2-Score	0.857143	0.372918	0.270910	0.452946	0.271195	0.524152	0.393731
Recall	0.857143	0.919643	0.919643	0.919643	0.919643	0.910714	0.919643
Precision	0.857143	0.110397	0.070888	0.149492	0.070986	0.194286	0.119767
FM index	0.857143	0.318631	0.255326	0.370782	0.255502	0.420641	0.331878
Specificity	0.999719	0.985400	0.976253	0.989692	0.976288	0.992559	0.986684
G-mean	0.925690	0.951954	0.947525	0.954025	0.947542	0.950757	0.952574
F0.5-Score	0.857143	0.133975	0.086935	0.179568	0.087052	0.230561	0.144989
Accuracy	0.999438	0.985271	0.976142	0.989554	0.976177	0.992398	0.986552

In []:

```

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

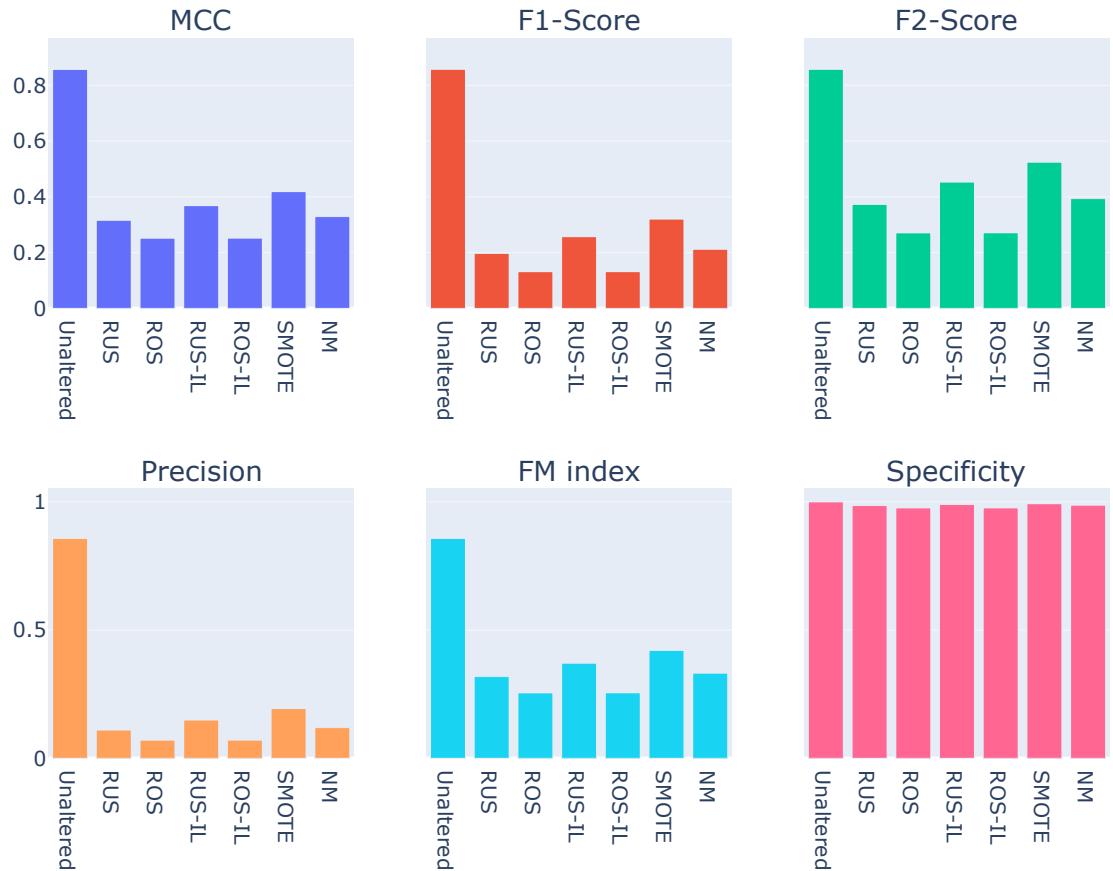
summary_visual(summary_svm_linear)

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

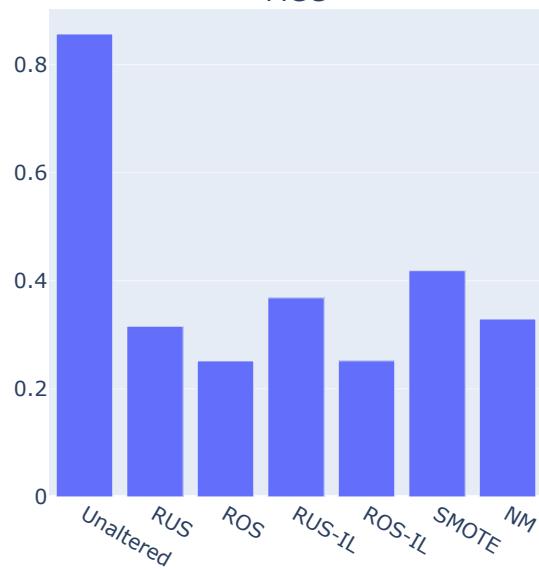
fig1.add_trace(go.Bar(x = list(summary_svm_linear.columns), y = list(summary_svm_linear.loc['MCC']))
fig1.add_trace(go.Bar(x = list(summary_svm_linear.columns), y = list(summary_svm_linear.loc['F1-']))
fig1.add_trace(go.Bar(x = list(summary_svm_linear.columns), y = list(summary_svm_linear.loc['F2-']))
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()

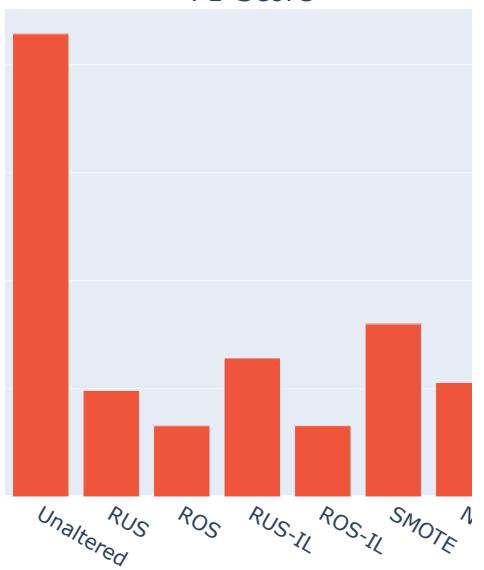
```



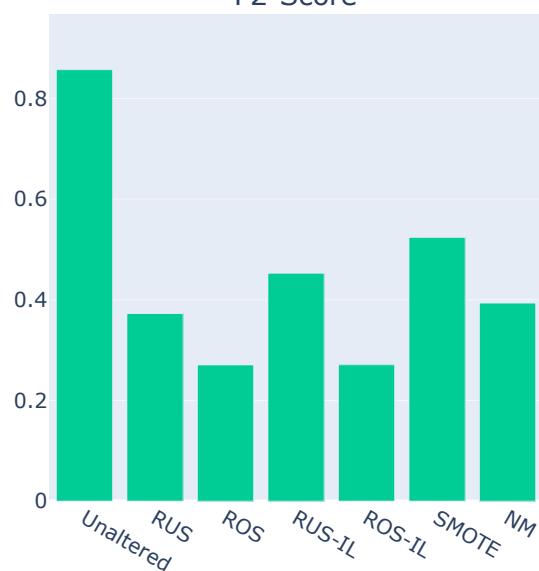
MCC



F1-Score



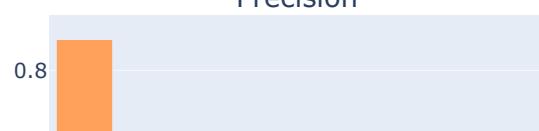
F2-Score



Recall

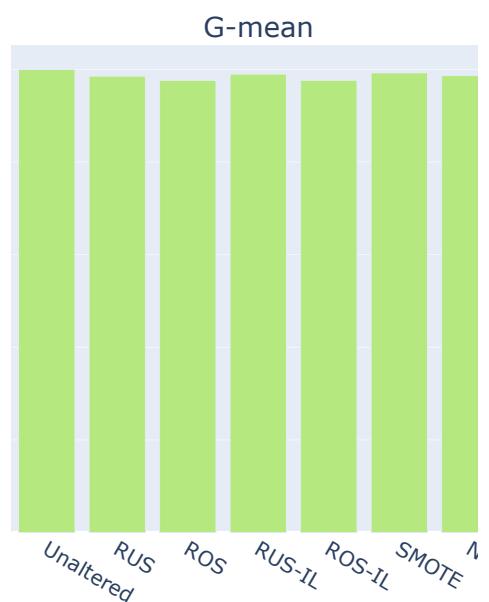
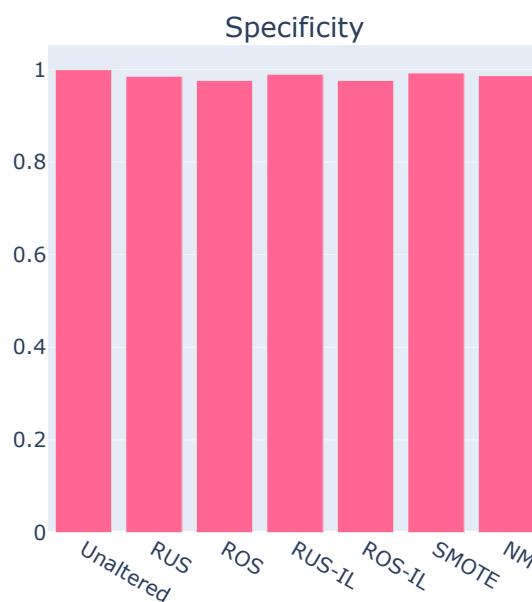
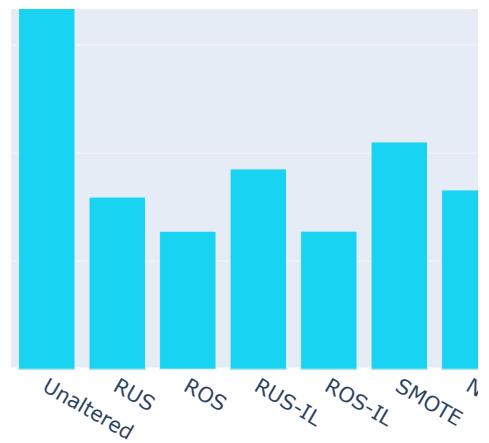
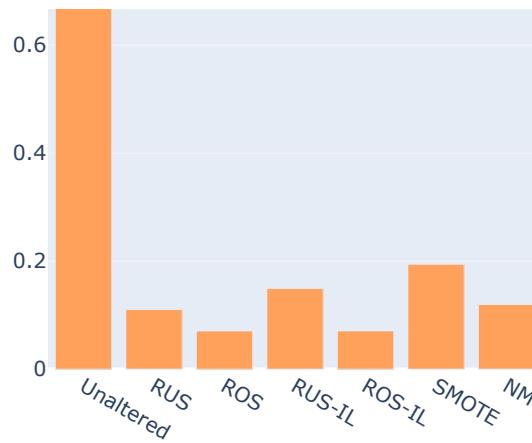


Precision



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

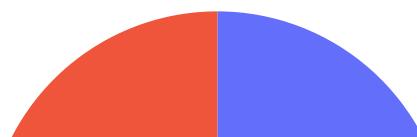
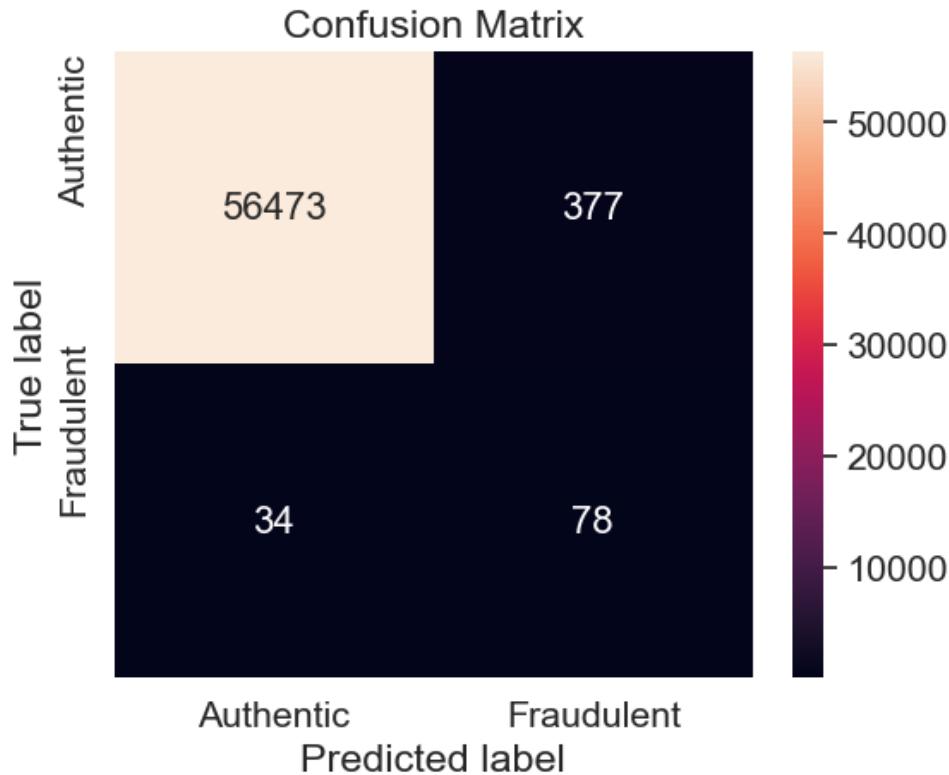
summary_nb_unaltered = summary.copy()
summary_nb_unaltered.set_index('Metric')

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_auc]
summary_nb_unaltered_extended.set_index('Metric')

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



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)

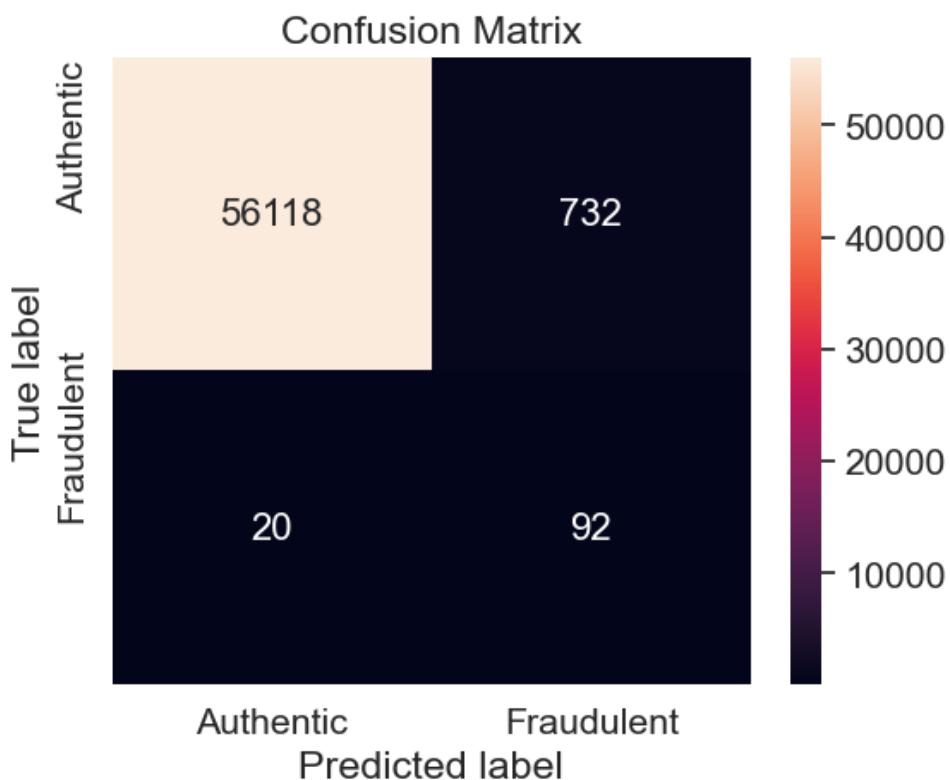
# Resumo das métricas de avaliação

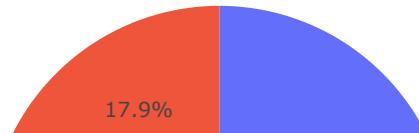
summary_nb_under = summary.copy()
summary_nb_under.set_index('Metric')

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	0.299979	0.196581	0.361635	0.821429	0.11165	0.302841	0.987124	0.900473	0.134977

Random over-sampling

```
In [ ]: # Elementos da matriz de confusão
classification(nb, X_train_over, y_train_over, X_test, y_test)

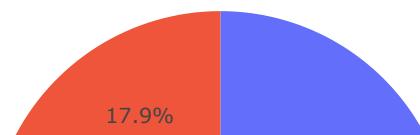
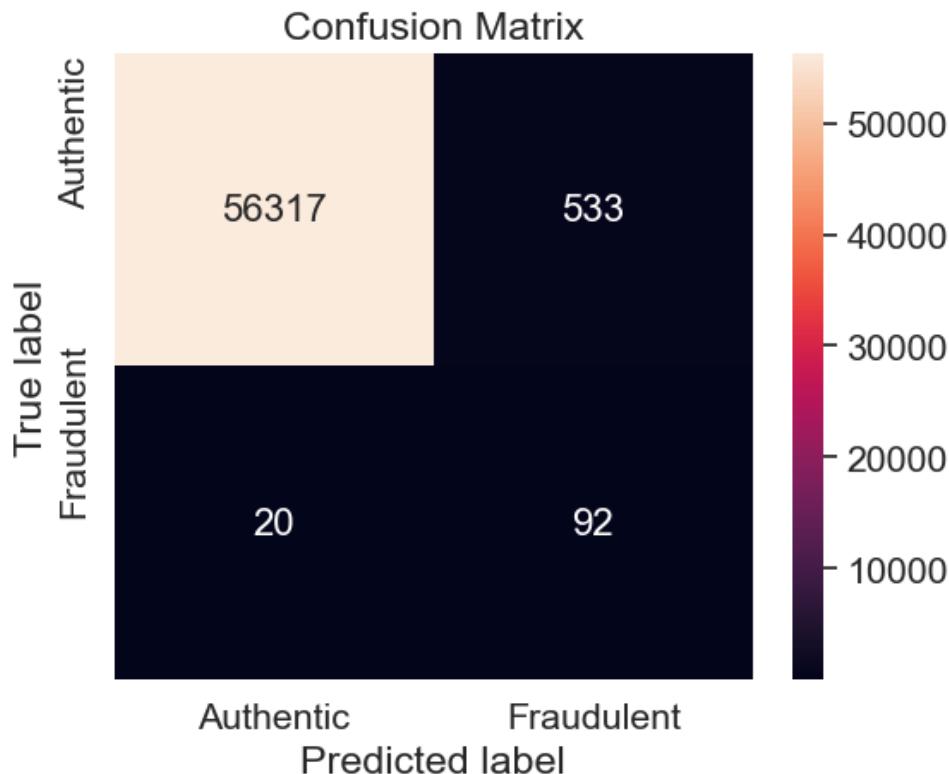
# Resumo das métricas de avaliação

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	0.34532	0.249661	0.428705	0.821429	0.1472	0.347727	0.990624	0.902068	0.17611	0	

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)
```

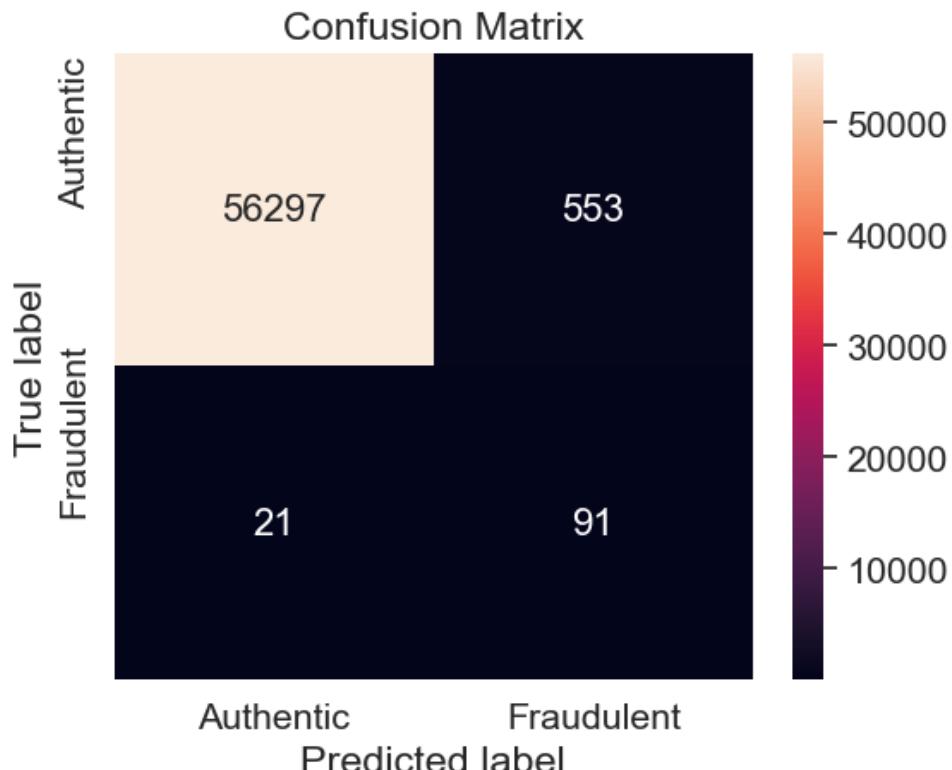
```
# Resumo das métricas de avaliação

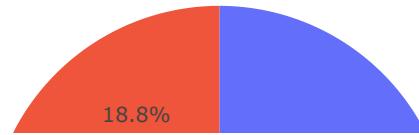
summary_nb_under_imblearn = summary.copy()
summary_nb_under_imblearn.set_index('Metric')

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)

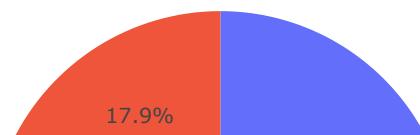
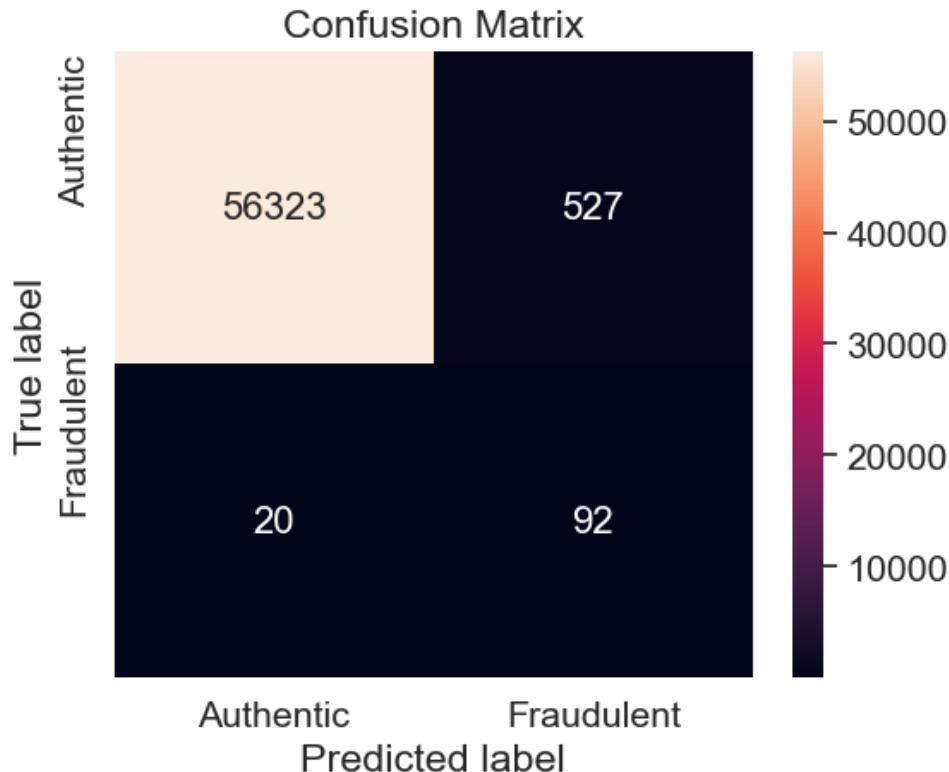
# Resumo das métricas de avaliação

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
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)
```

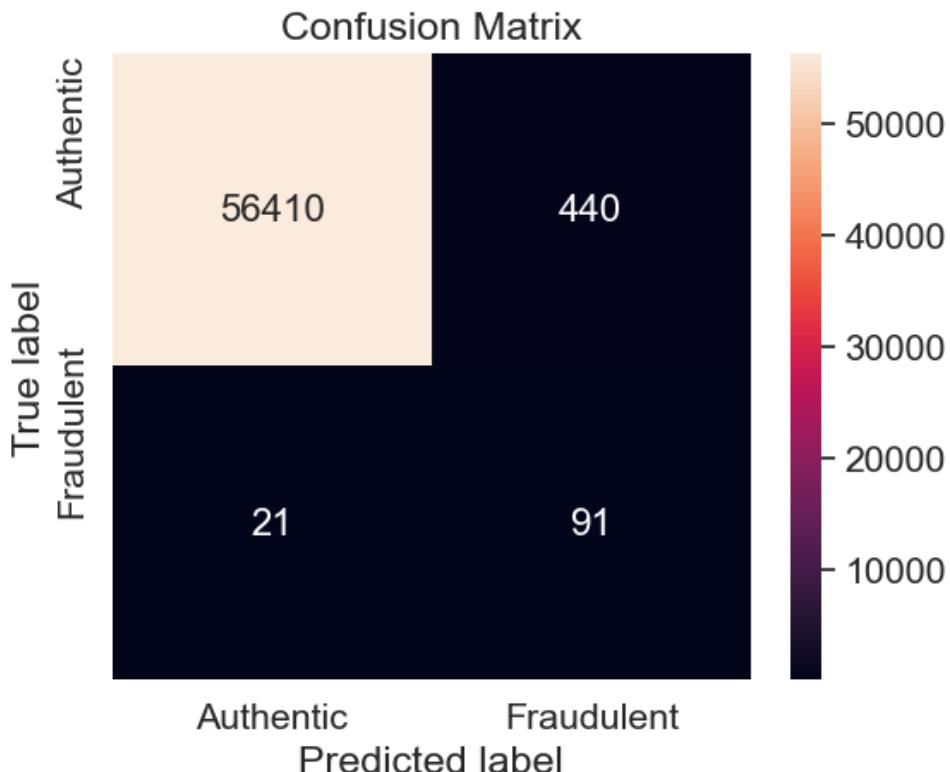
```
# Resumo das métricas de avaliação

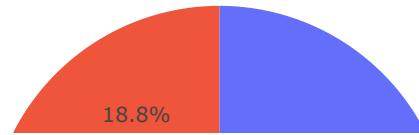
summary_nb_over_smote = summary.copy()
summary_nb_over_smote.set_index('Metric')

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_auc]
summary_nb_over_smote_extended.set_index('Metric')

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





Out[]:	Metric	MCC	F1-Score	F2-Score	Recall	Precision	FM index	Specificity	G-mean	F0.5-Score	Auc
Performance score	0.370966	0.283048	0.46476	0.8125	0.171375	0.373151		0.99226	0.897893	0.203488	0.5

Under-sampling via NearMiss

```
In [ ]: # Elementos da matriz de confusão
classification(nb, X_train_under_nm, y_train_under_nm, X_test, y_test)

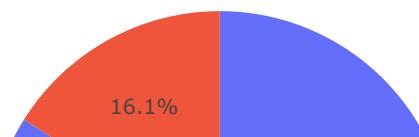
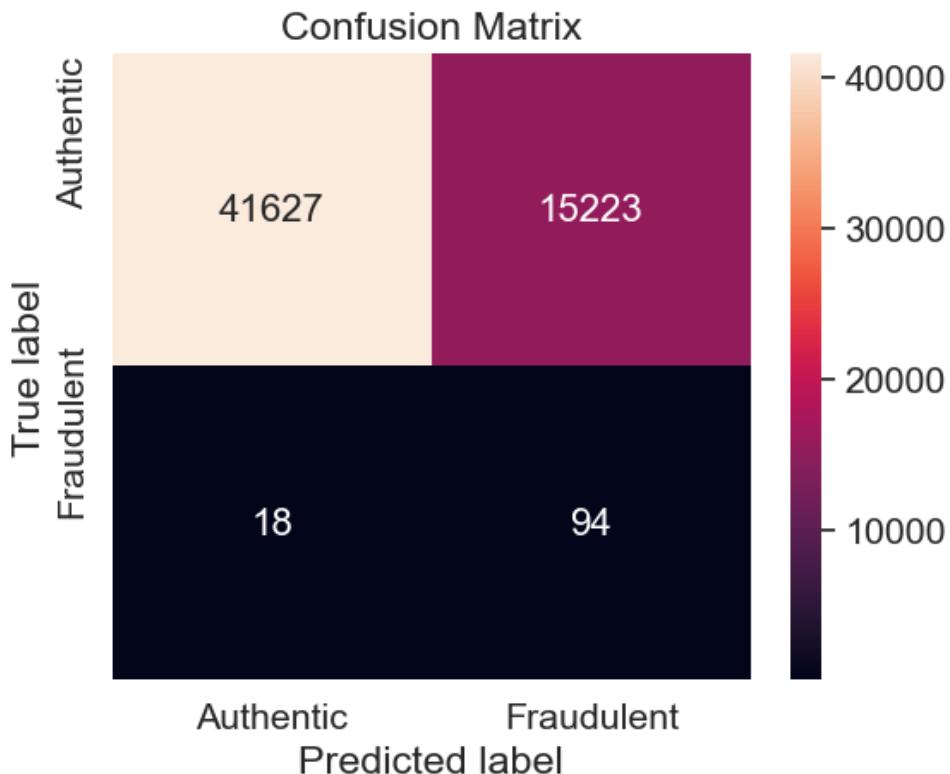
# Resumo das métricas de avaliação

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
```



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_18520\3110425635.py:8: FutureWarning:

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

C:\Users\Diones\AppData\Local\Temp\ipykernel_18520\3110425635.py:8: FutureWarning:

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

Out[]:

	Unaltered	RUS	ROS	RUS-IL	ROS-IL	SMOTE	NM
Metric							
MCC	0.343272	0.299979	0.345320	0.336357	0.347016	0.370966	0.057099
F1-Score	0.275132	0.196581	0.249661	0.240741	0.251710	0.283048	0.012185
F2-Score	0.431894	0.361635	0.428705	0.416667	0.431115	0.464760	0.029813
Recall	0.696429	0.821429	0.821429	0.812500	0.821429	0.812500	0.839286
Precision	0.171429	0.111650	0.147200	0.141304	0.148627	0.171375	0.006137
FM index	0.345525	0.302841	0.347727	0.338836	0.349409	0.373151	0.071768
Specificity	0.993369	0.987124	0.990624	0.990273	0.990730	0.992260	0.732225
G-mean	0.831751	0.900473	0.902068	0.896993	0.902116	0.897893	0.783930
F0.5-Score	0.201863	0.134977	0.176110	0.169271	0.177743	0.203488	0.007657
Accuracy	0.992785	0.986798	0.990292	0.989923	0.990397	0.991907	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

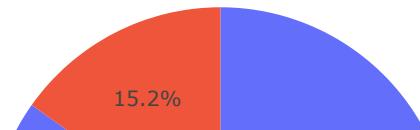
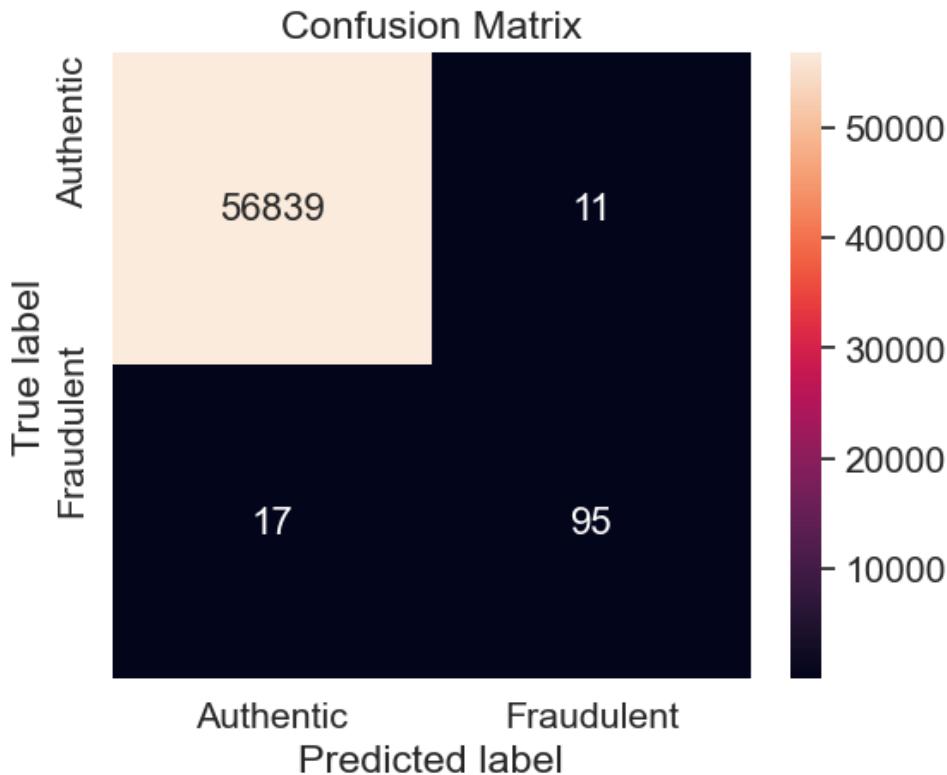
summary_rf_unaltered = summary.copy()
summary_rf_unaltered.set_index('Metric')

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_auc]
summary_rf_unaltered_extended.set_index('Metric')

summary_rf_unaltered_index = summary_rf_unaltered_extended.T
summary_rf_unaltered_index.columns = summary_rf_unaltered_index.iloc[0]
```

```
summary_rf_unaltered_index.drop(summary_rf_unaltered_index.index[0], inplace = True)
summary_rf_unaltered_index
```



Out[]:	Metric	MCC	F1-Score	F2-Score	Recall	Precision	FM index	Specificity	G-mean	F0.5-Score	A
Performance score		0.871645	0.87156	0.857401	0.848214	0.896226	0.87189	0.999807	0.920896	0.886194	0

Random under-sampling

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

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

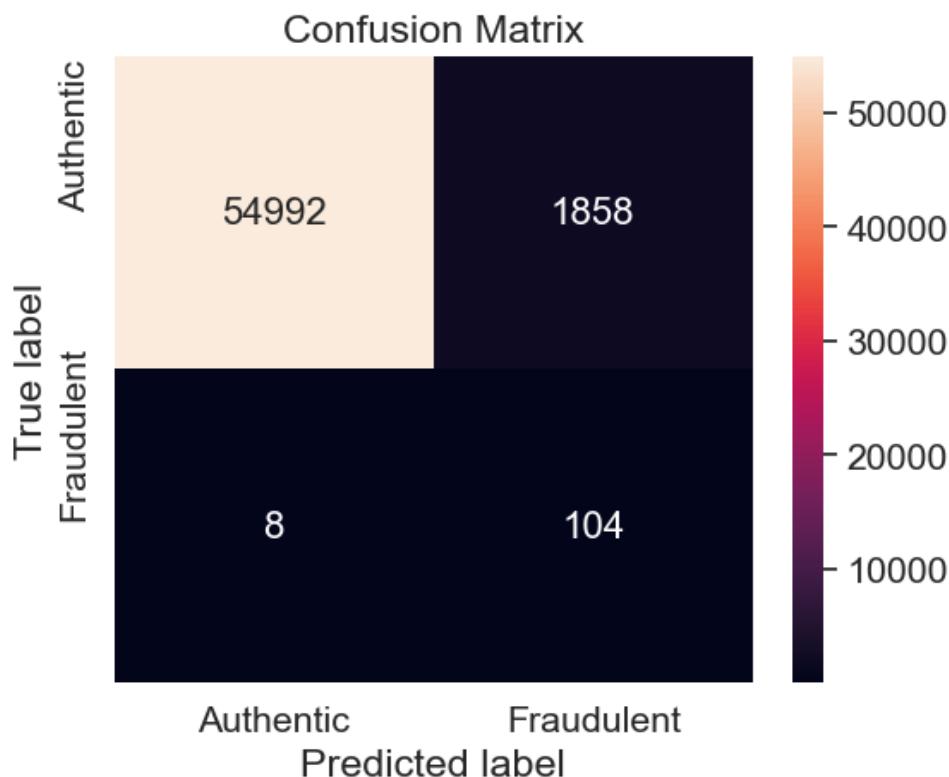
# Resumo das métricas de avaliação

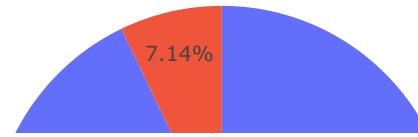
summary_rf_under = summary.copy()
summary_rf_under.set_index('Metric')

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
	Performance score	0.217619	0.100289	0.215768	0.928571	0.053007	0.221858	0.967318	0.947746	0.065327

Random over-sampling

```
In [ ]: # Elementos da matriz de confusão
classification(rf, X_train_over, y_train_over, X_test, y_test)

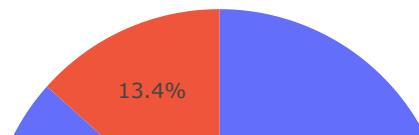
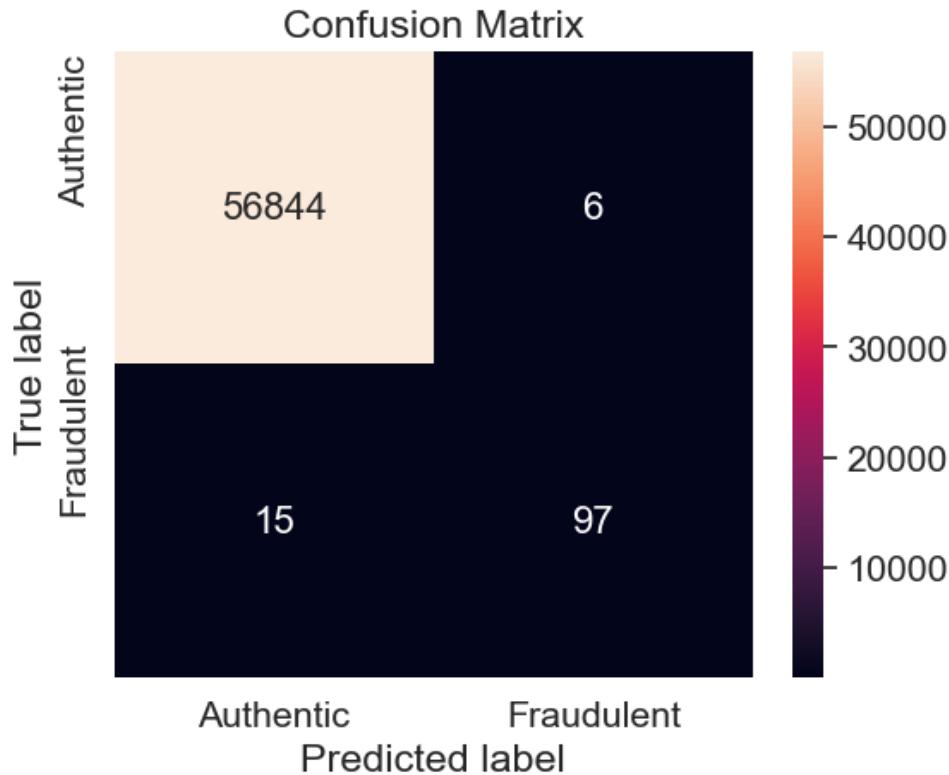
# Resumo das métricas de avaliação

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	0.902936	0.902326	0.880218	0.866071	0.941748	0.903117	0.999894	0.93058	0.925573	

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)
```

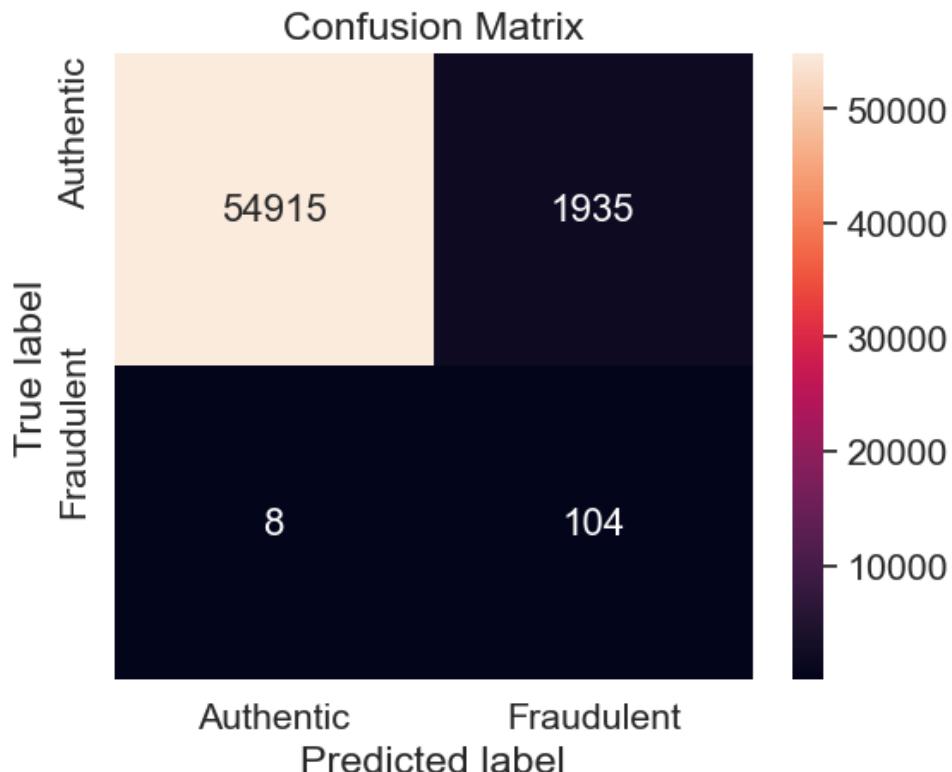
```
# Resumo das métricas de avaliação

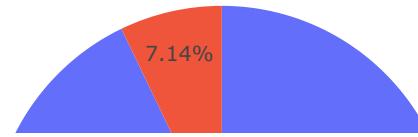
summary_rf_under_imblearn = summary.copy()
summary_rf_under_imblearn.set_index('Metric')

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	0.213297	0.096699	0.209087	0.928571	0.051005	0.217628	0.965963	0.947083	0.062893

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)

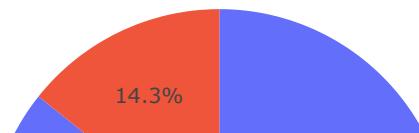
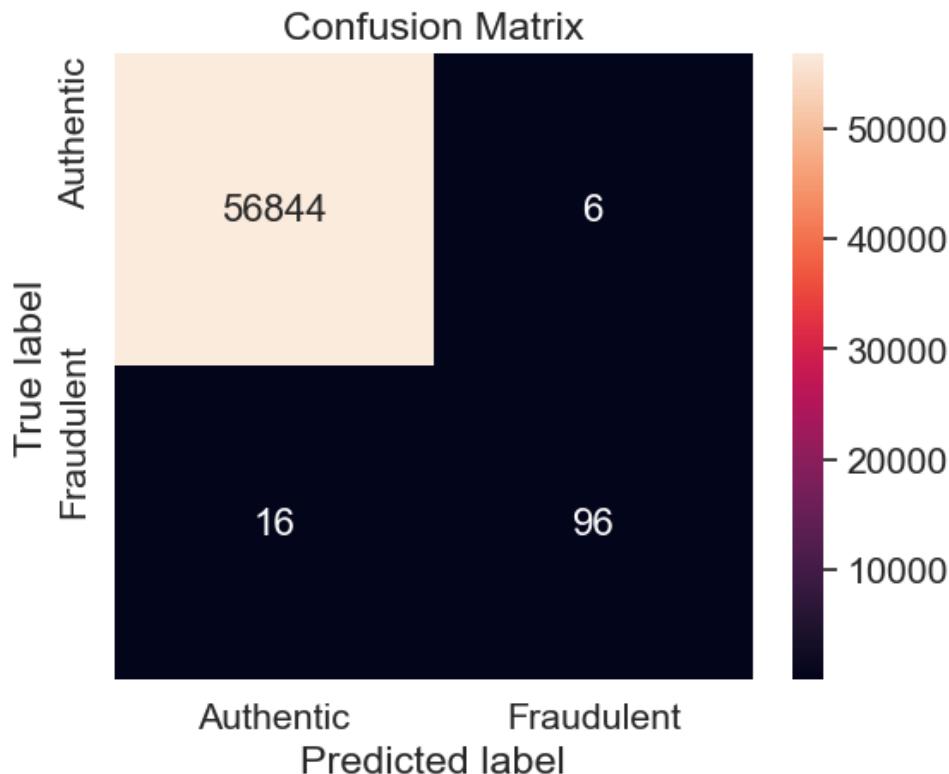
# Resumo das métricas de avaliação

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		0.897988	0.897196	0.872727	0.857143	0.941176	0.898177	0.999894	0.925771	0.923077

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)
```

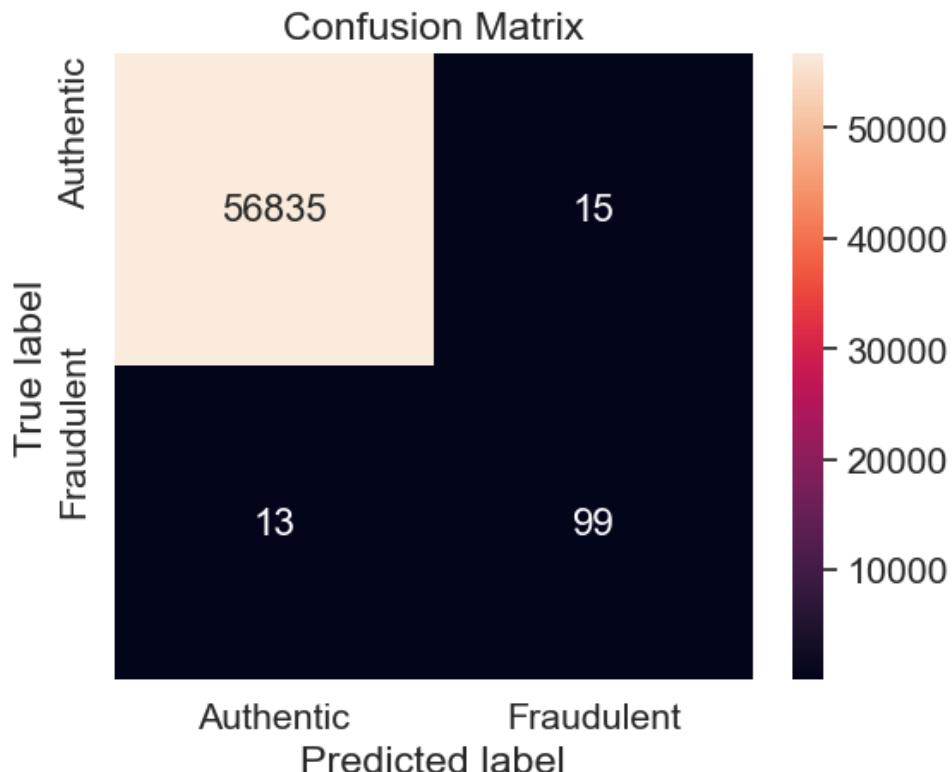
```
# Resumo das métricas de avaliação

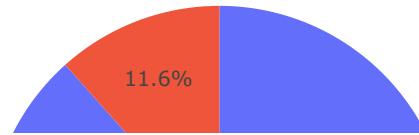
summary_rf_over_smote = summary.copy()
summary_rf_over_smote.set_index('Metric')

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_auc]
summary_rf_over_smote_extended.set_index('Metric')

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





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)

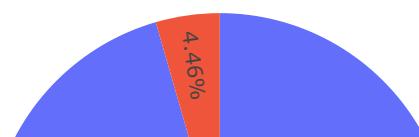
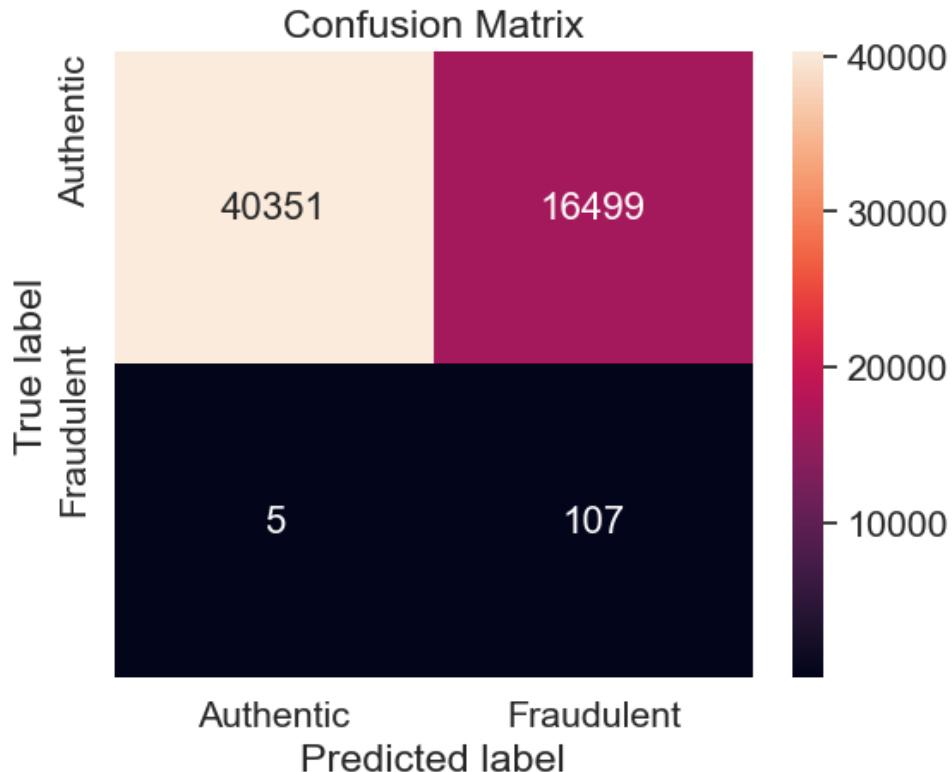
# Resumo das métricas de avaliação

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	0.064833	0.012801	0.031371	0.955357	0.006443	0.078459	0.70978	0.823464	0.008041	

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_18520\2240544619.py:8: FutureWarning:

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

C:\Users\Diones\AppData\Local\Temp\ipykernel_18520\2240544619.py:8: FutureWarning:

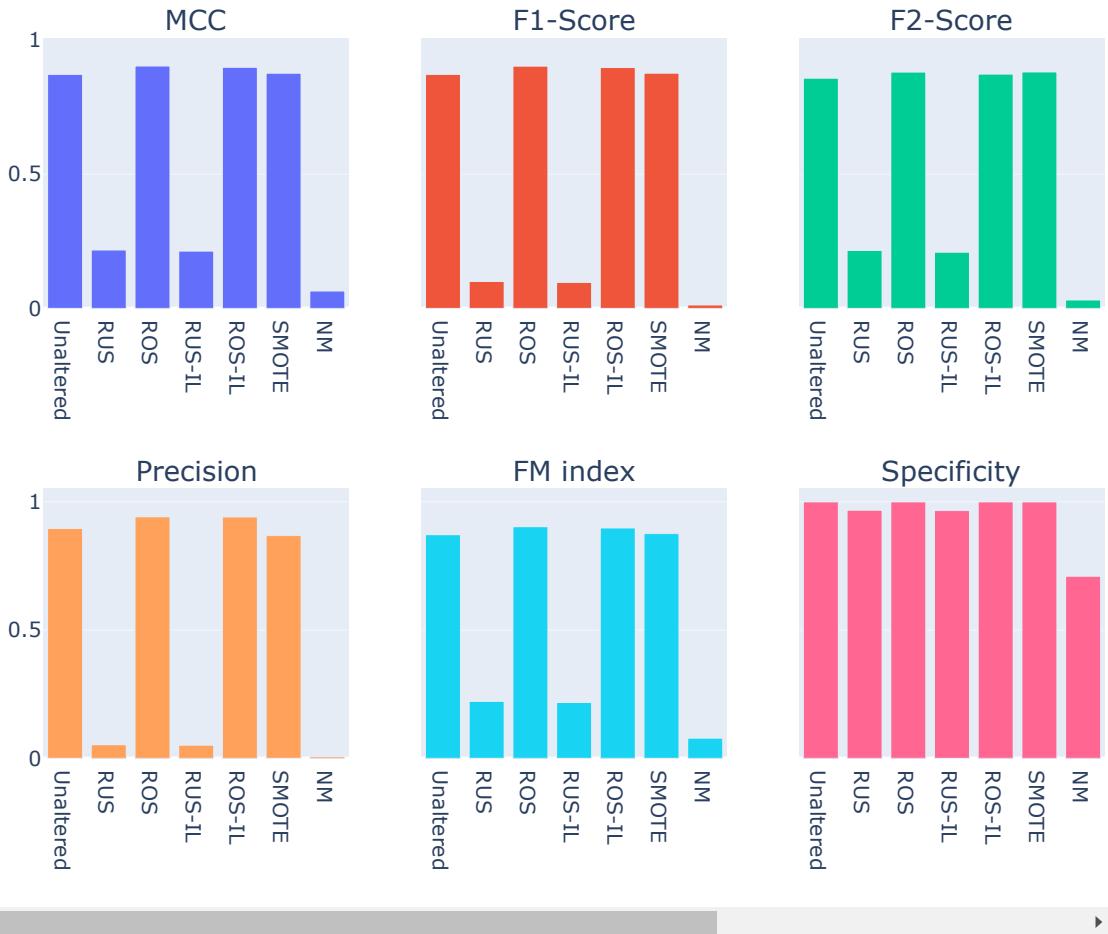
Passing 'suffixes' which cause duplicate columns {'Performance score_x'} in the result is deprecated and will raise a MergeError in a future version.

Out[]:

	Unaltered	RUS	ROS	RUS-IL	ROS-IL	SMOTE	NM
Metric							
MCC	0.871645	0.217619	0.902936	0.213297	0.897988	0.875894	0.064833
F1-Score	0.871560	0.100289	0.902326	0.096699	0.897196	0.876106	0.012801
F2-Score	0.857401	0.215768	0.880218	0.209087	0.872727	0.880783	0.031371
Recall	0.848214	0.928571	0.866071	0.928571	0.857143	0.883929	0.955357
Precision	0.896226	0.053007	0.941748	0.051005	0.941176	0.868421	0.006443
FM index	0.871890	0.221858	0.903117	0.217628	0.898177	0.876141	0.078459
Specificity	0.999807	0.967318	0.999894	0.965963	0.999894	0.999736	0.709780
G-mean	0.920896	0.947746	0.930580	0.947083	0.925771	0.940051	0.823464
F0.5-Score	0.886194	0.065327	0.925573	0.062893	0.923077	0.871479	0.008041
Accuracy	0.999508	0.967241	0.999631	0.965890	0.999614	0.999508	0.710263

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

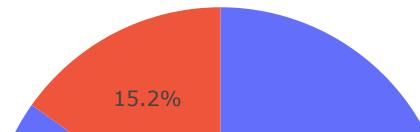
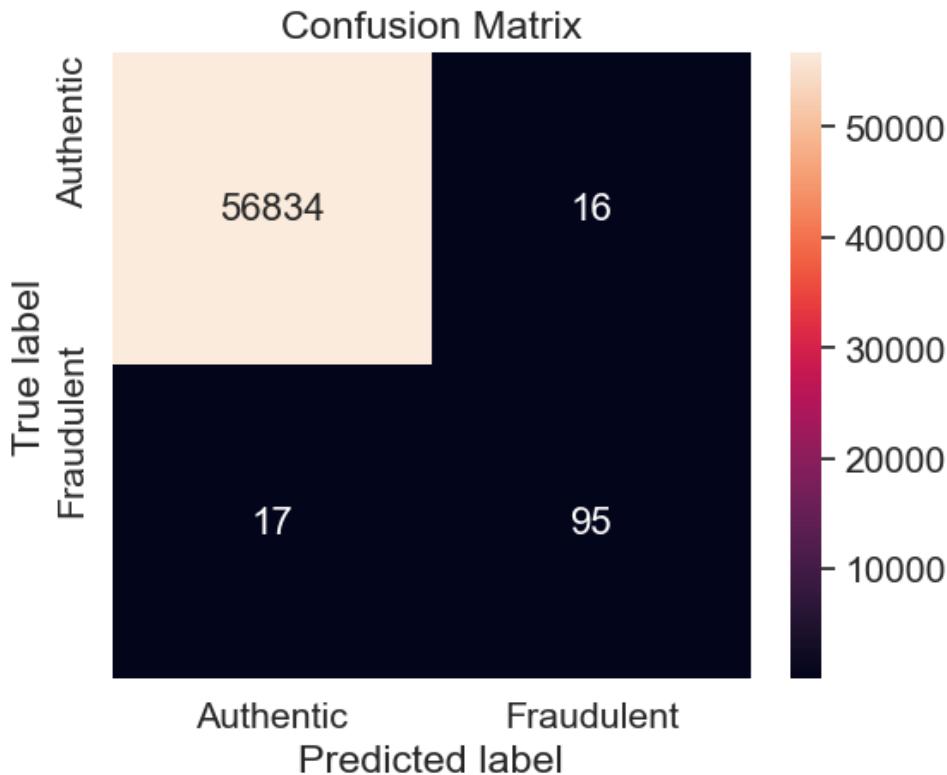
summary_lda_unaltered = summary.copy()
summary_lda_unaltered.set_index('Metric')

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)

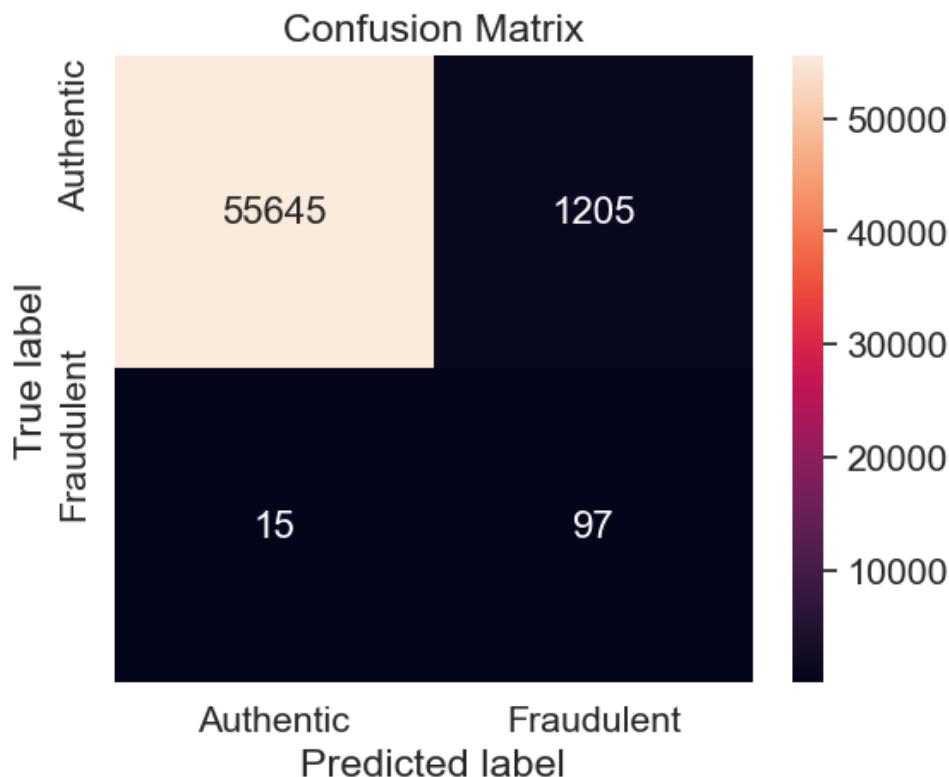
# Resumo das métricas de avaliação

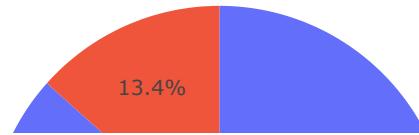
summary_lda_under = summary
summary_lda_under.set_index('Metric')

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
Performance score	0.250432	0.137199	0.277143	0.866071	0.074501	0.254014	0.978804	0.920714	0.091165	

Random over-sampling

```
In [ ]: # Elementos da matriz de confusão
classification(lda, X_train_over, y_train_over, X_test, y_test)

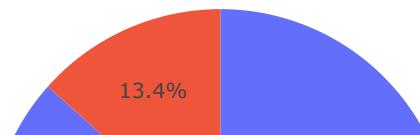
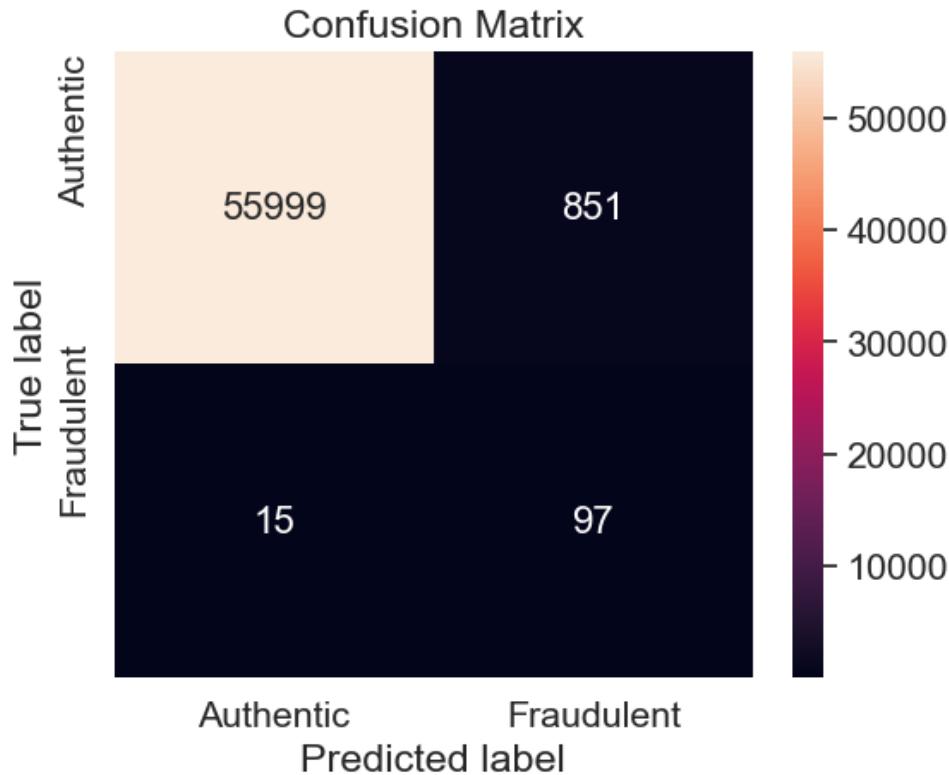
# Resumo das métricas de avaliação

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	0.294716	0.183019	0.347421	0.866071	0.102321	0.297686	0.985031	0.923638	0.124232	

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)
```

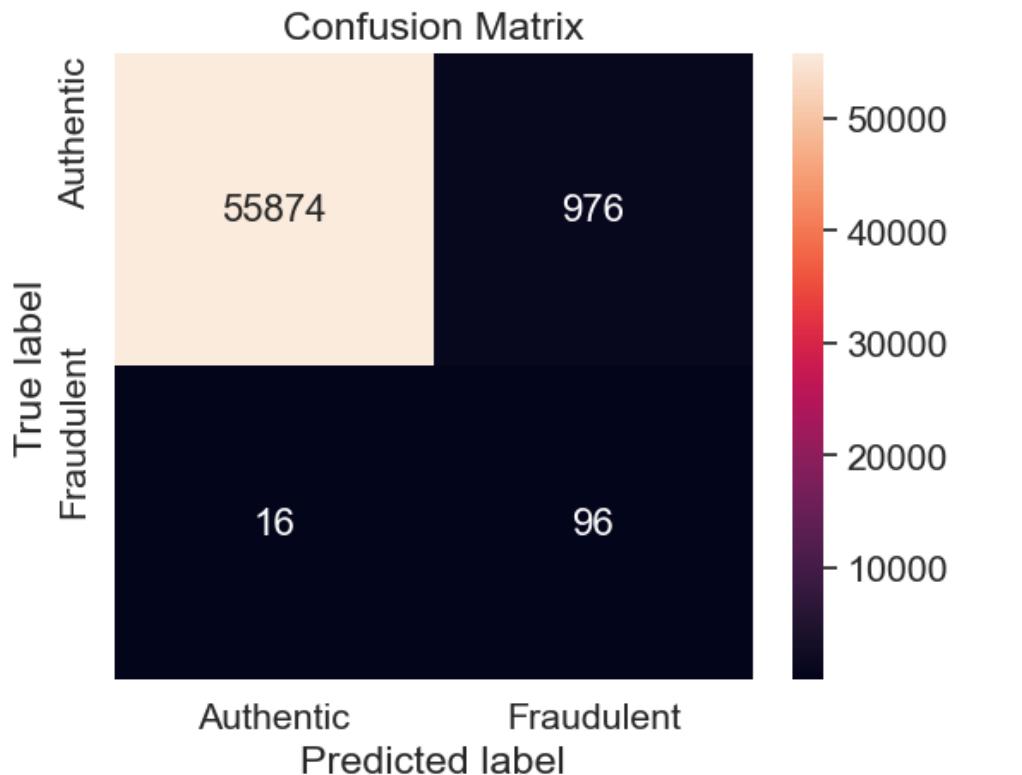
```
# Resumo das métricas de avaliação

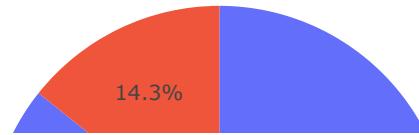
summary_lda_under_imblearn = summary
summary_lda_under_imblearn.set_index('Metric')

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', 'Precision']
summary_lda_under_imblearn_extended.loc[len(summary_lda_under_imblearn_extended.index)] = ['ROC-AUC', 'AUC']
summary_lda_under_imblearn_extended.set_index('Metric')

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





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)

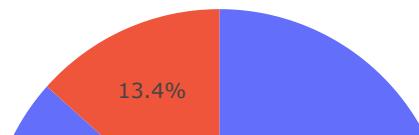
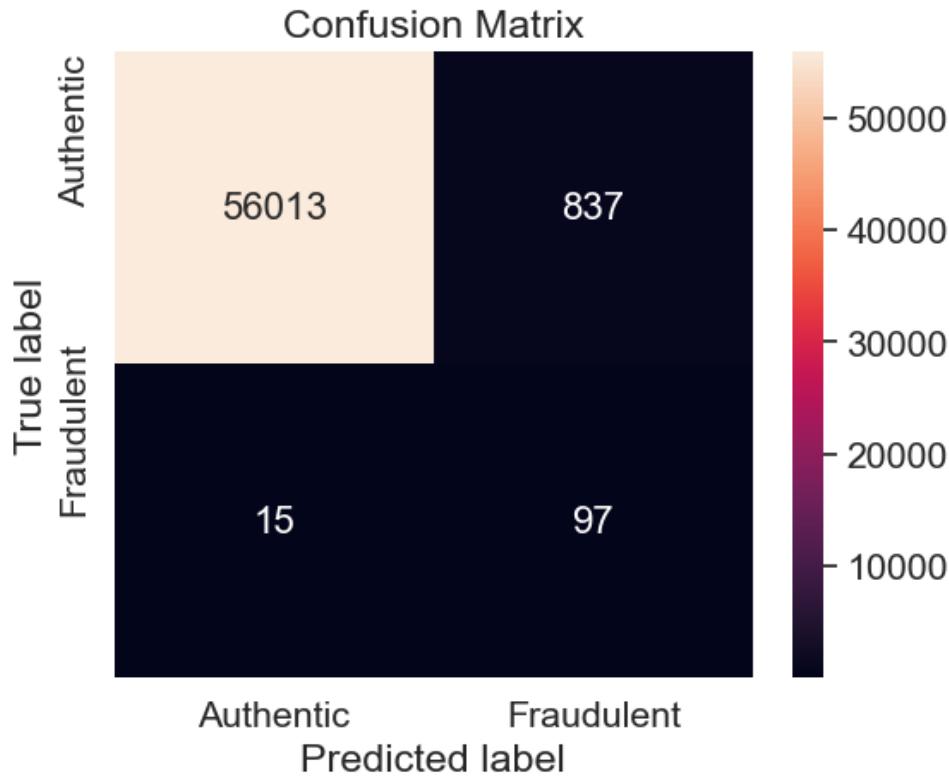
# Resumo das métricas de avaliação

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', average_precision]
summary_lda_over_imblearn_extended.loc[len(summary_lda_over_imblearn_extended.index)] = ['ROC-AU', roc_auc]
summary_lda_over_imblearn_extended.set_index('Metric')

summary_lda_over_imblearn_index = summary_lda_over_imblearn_extended.T
summary_lda_over_imblearn_index.columns = summary_lda_over_imblearn_index.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
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)
```

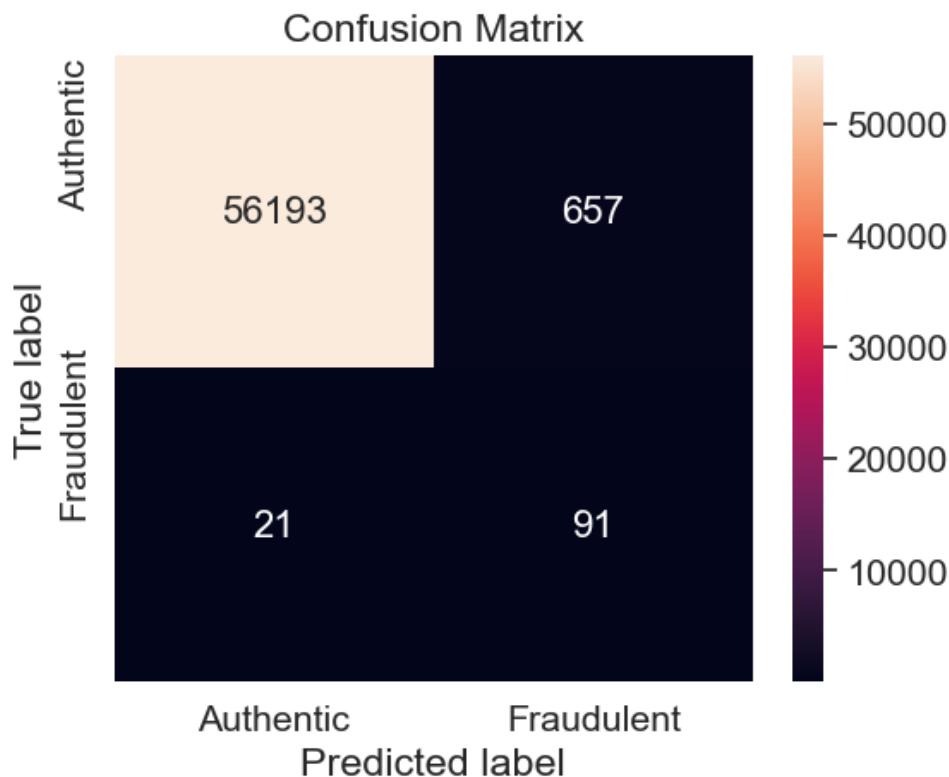
```
# Resumo das métricas de avaliação

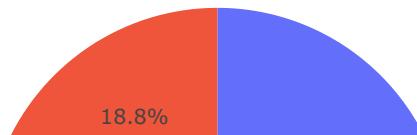
summary_lda_over_smote = summary
summary_lda_over_smote.set_index('Metric')

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

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





Out[]:	Metric	MCC	F1-Score	F2-Score	Recall	Precision	FM index	Specificity	G-mean	F0.5-Score	A
	Performance score	0.311676	0.211628	0.380435	0.8125	0.121658	0.314399	0.988443	0.896164	0.146585	0

Under-sampling via NearMiss

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

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

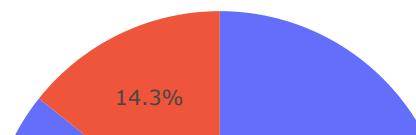
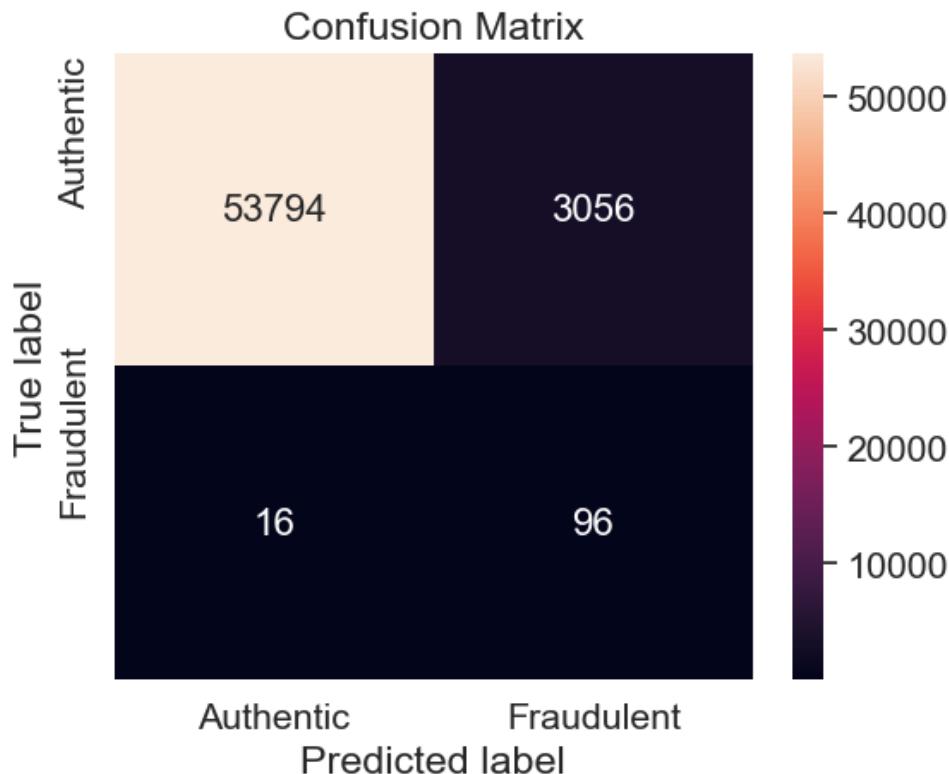
# Resumo das métricas de avaliação

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

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



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_18520\741047787.py:8: FutureWarning:

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

C:\Users\Diones\AppData\Local\Temp\ipykernel_18520\741047787.py:8: FutureWarning:

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

Out[]:

	Unaltered	RUS	ROS	RUS-IL	ROS-IL	SMOTE	NM
Metric							
MCC	0.851736	0.250432	0.294716	0.273827	0.296965	0.311676	0.155659
F1-Score	0.852018	0.137199	0.183019	0.162162	0.185468	0.211628	0.058824
F2-Score	0.849732	0.277143	0.347421	0.315789	0.350941	0.380435	0.133333
Recall	0.848214	0.866071	0.866071	0.857143	0.866071	0.812500	0.857143
Precision	0.855856	0.074501	0.102321	0.089552	0.103854	0.121658	0.030457
FM index	0.852027	0.254014	0.297686	0.277054	0.299909	0.314399	0.161573
Specificity	0.999719	0.978804	0.985031	0.982832	0.985277	0.988443	0.946245
G-mean	0.920856	0.920714	0.923638	0.917838	0.923753	0.896164	0.900592
F0.5-Score	0.854317	0.091165	0.124232	0.109091	0.126040	0.146585	0.037736
Accuracy	0.999421	0.978582	0.984797	0.982585	0.985043	0.988097	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)

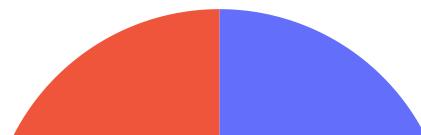
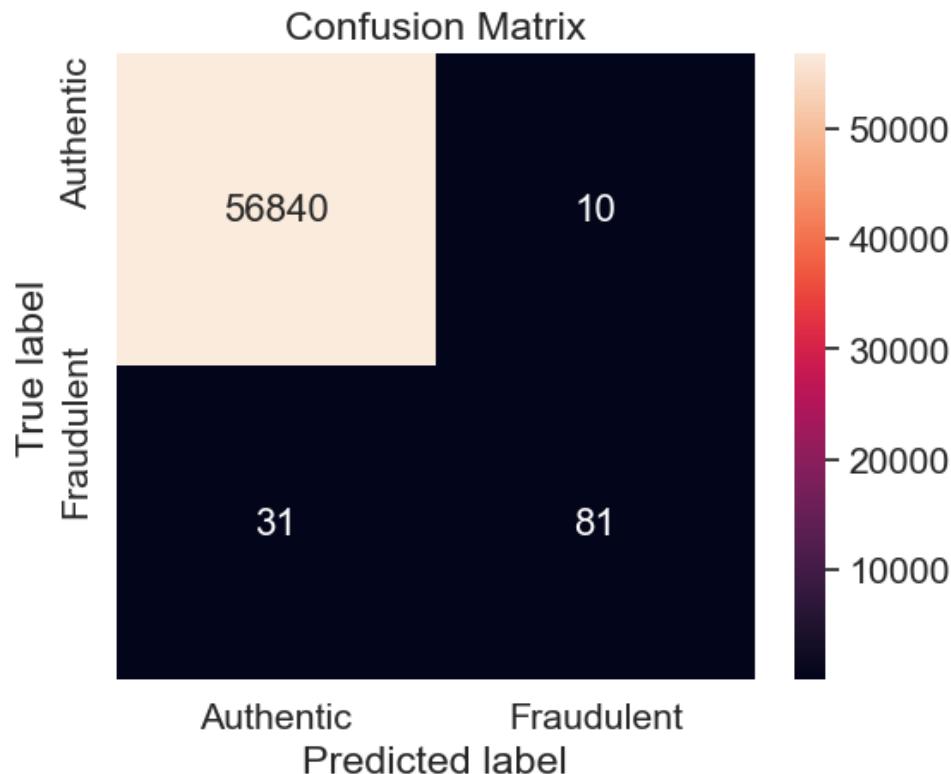
# Resumo das métricas de avaliação

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		0.801991	0.79803	0.751391	0.723214	0.89011	0.802334	0.999824	0.850345	0.85084	

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)

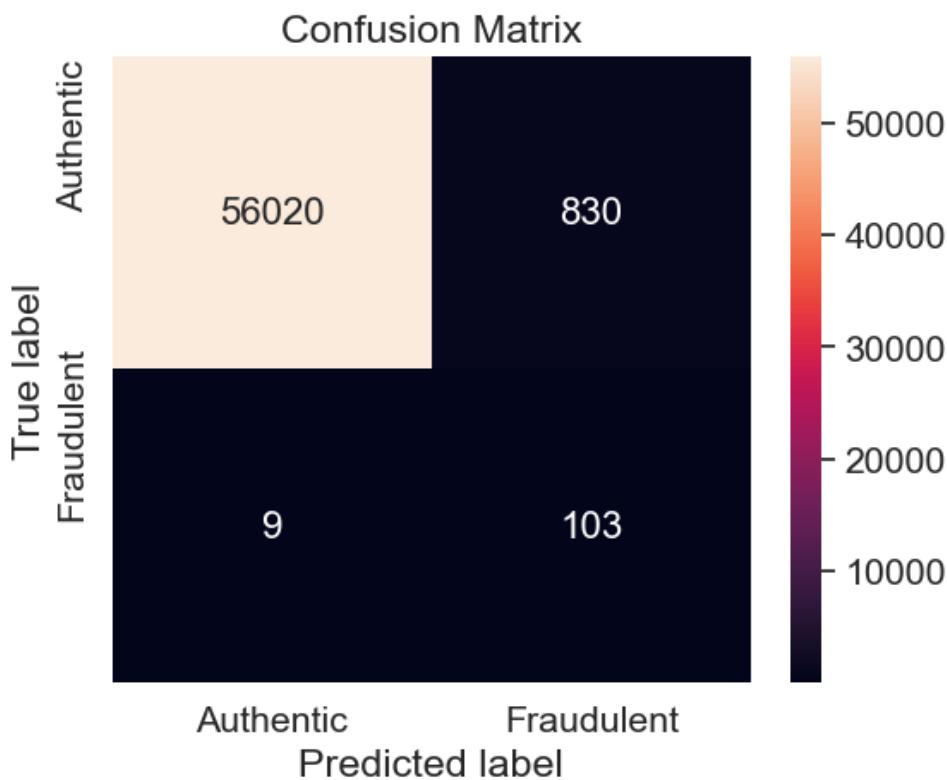
# Resumo das métricas de avaliação

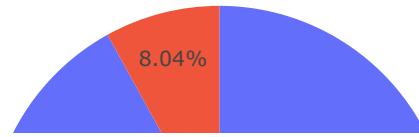
summary_sgd_under = summary
summary_sgd_under.set_index('Metric')

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
	Performance score	0.315861	0.197129	0.372918	0.919643	0.110397	0.318631	0.9854	0.951954	0.133975

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)

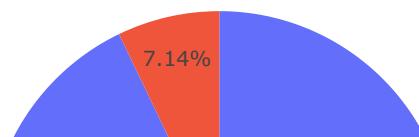
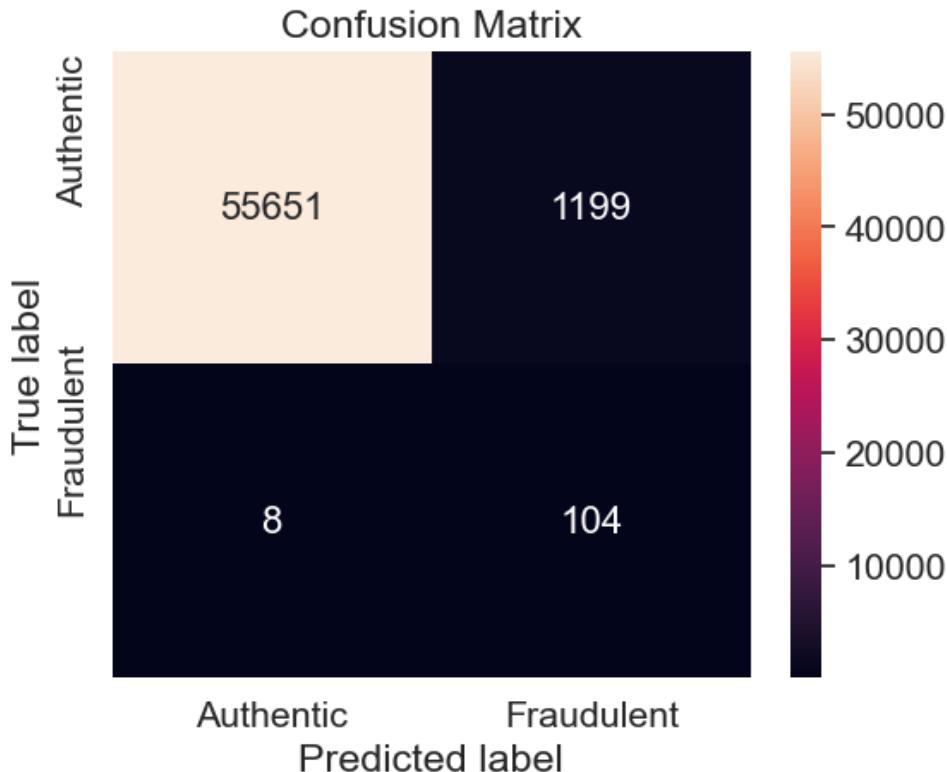
# Resumo das métricas de avaliação

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		0.268888	0.146996	0.296973	0.928571	0.079816	0.27224	0.978909	0.953408	0.097671

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)

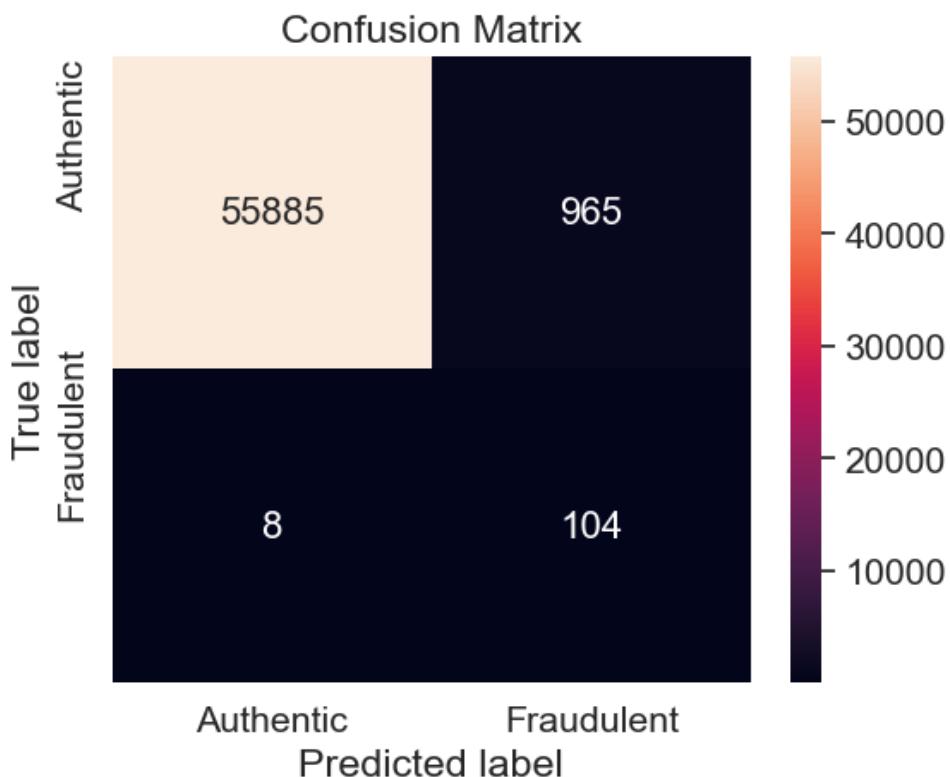
# Resumo das métricas de avaliação

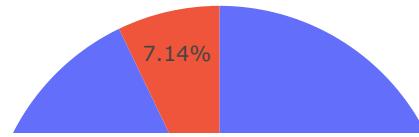
summary_sgd_under_imblearn = summary
summary_sgd_under_imblearn.set_index('Metric')

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
	Performance score	0.297584	0.176122	0.342782	0.928571	0.097287	0.300563	0.983026	0.955411	0.118505

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

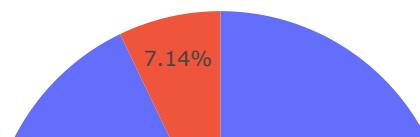
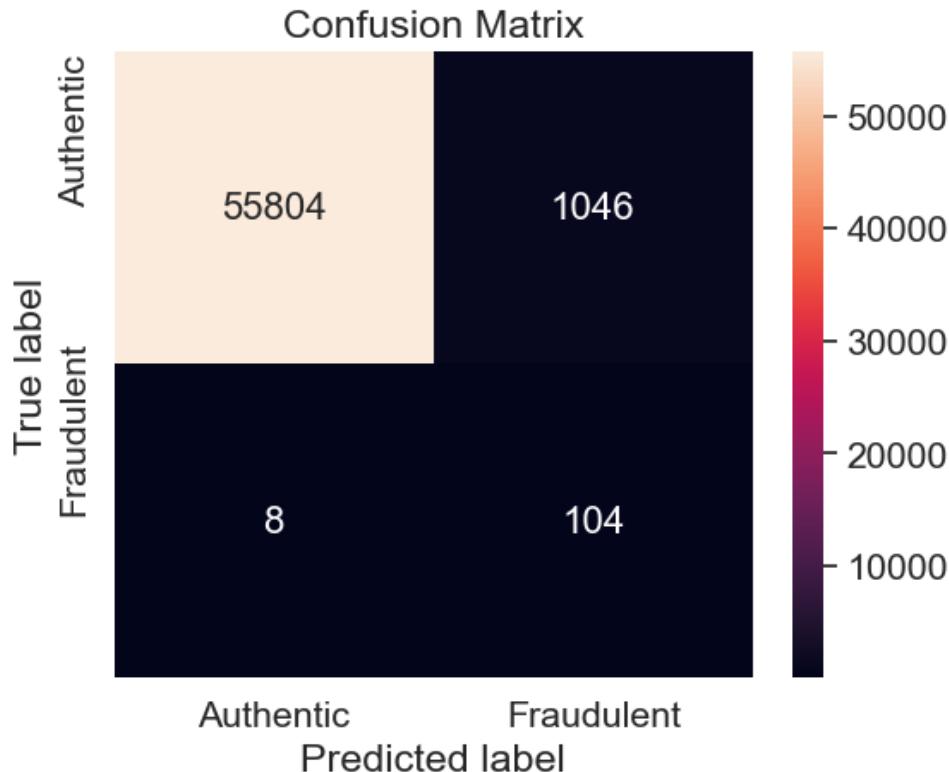
# Resumo das métricas de avaliação

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.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
Performance score		0.286672	0.164818	0.325407	0.928571	0.090435	0.289785	0.981601	0.954718	0.110357

Synthetic minority over-sampling technique (SMOTE)

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

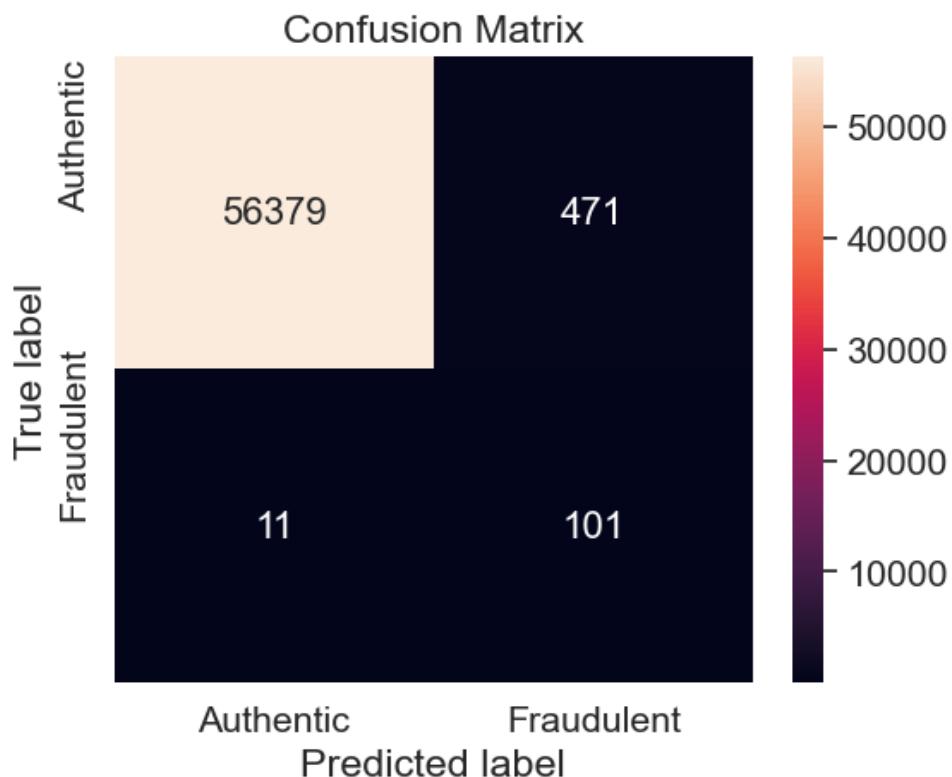
classification(sgd, X_train_over_smote_scaled_minmax, y_train_over_smote, X_test_scaled_minmax,
# Resumo das métricas de avaliação

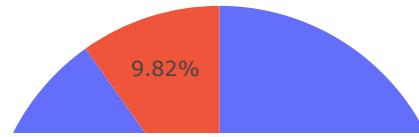
summary_sgd_over_smote = summary
summary_sgd_over_smote.set_index('Metric')

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
	Performance score	0.396981	0.295322	0.495098	0.901786	0.176573	0.399038	0.991715	0.945682	0.210417

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

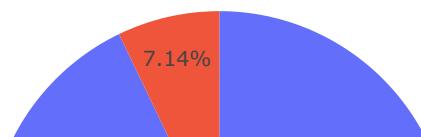
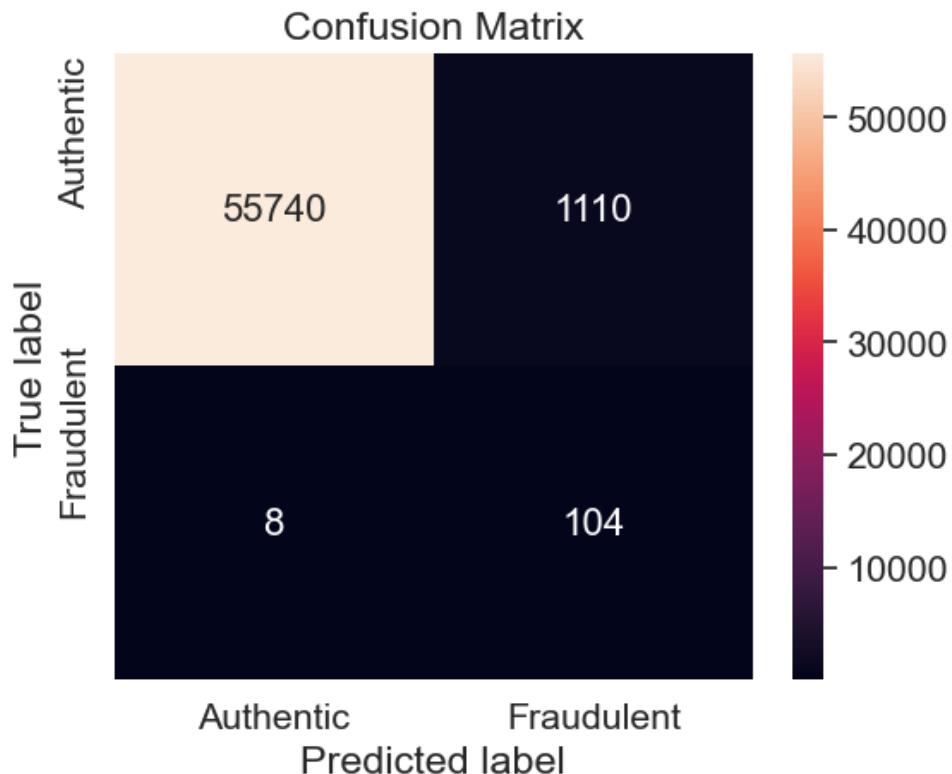
# Resumo das métricas de avaliação

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		0.278828	0.156863	0.312876	0.928571	0.085667	0.282043	0.980475	0.95417	0.10467	0

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\Diônes\AppData\Local\Temp\ipykernel_18520\4039655680.py:8: FutureWarning:

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

C:\Users\Diônes\AppData\Local\Temp\ipykernel_18520\4039655680.py:8: FutureWarning:

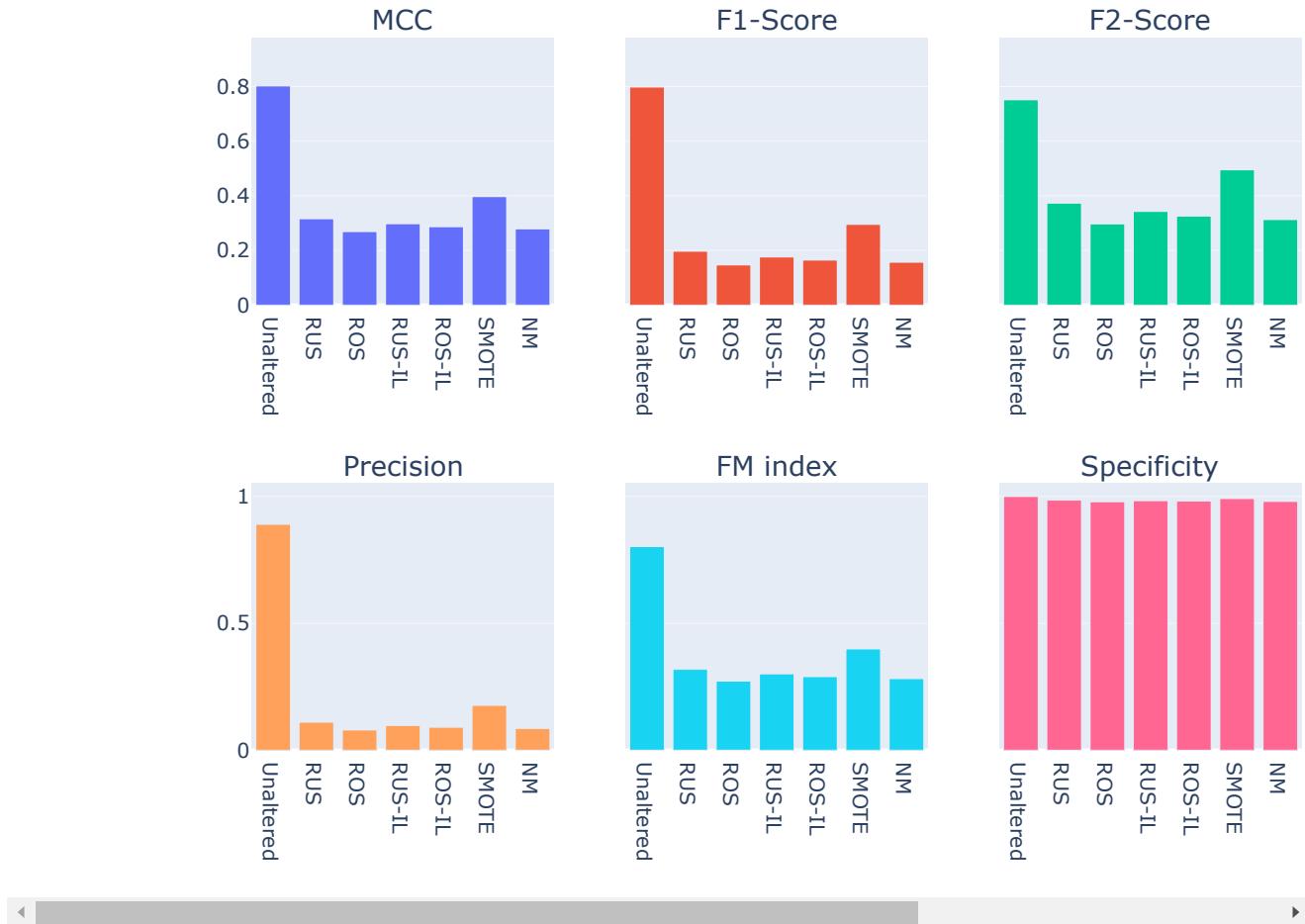
Passing 'suffixes' which cause duplicate columns {'Performance score_x'} in the result is deprecated and will raise a MergeError in a future version.

Out[]:

	Unaltered	RUS	ROS	RUS-IL	ROS-IL	SMOTE	NM
Metric							
MCC	0.801991	0.315861	0.268888	0.297584	0.286672	0.396981	0.278828
F1-Score	0.798030	0.197129	0.146996	0.176122	0.164818	0.295322	0.156863
F2-Score	0.751391	0.372918	0.296973	0.342782	0.325407	0.495098	0.312876
Recall	0.723214	0.919643	0.928571	0.928571	0.928571	0.901786	0.928571
Precision	0.890110	0.110397	0.079816	0.097287	0.090435	0.176573	0.085667
FM index	0.802334	0.318631	0.272240	0.300563	0.289785	0.399038	0.282043
Specificity	0.999824	0.985400	0.978909	0.983026	0.981601	0.991715	0.980475
G-mean	0.850345	0.951954	0.953408	0.955411	0.954718	0.945682	0.954170
F0.5-Score	0.850840	0.133975	0.097671	0.118505	0.110357	0.210417	0.104670
Accuracy	0.999280	0.985271	0.978810	0.982918	0.981496	0.991538	0.980373

```
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

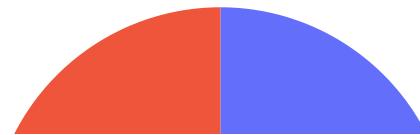
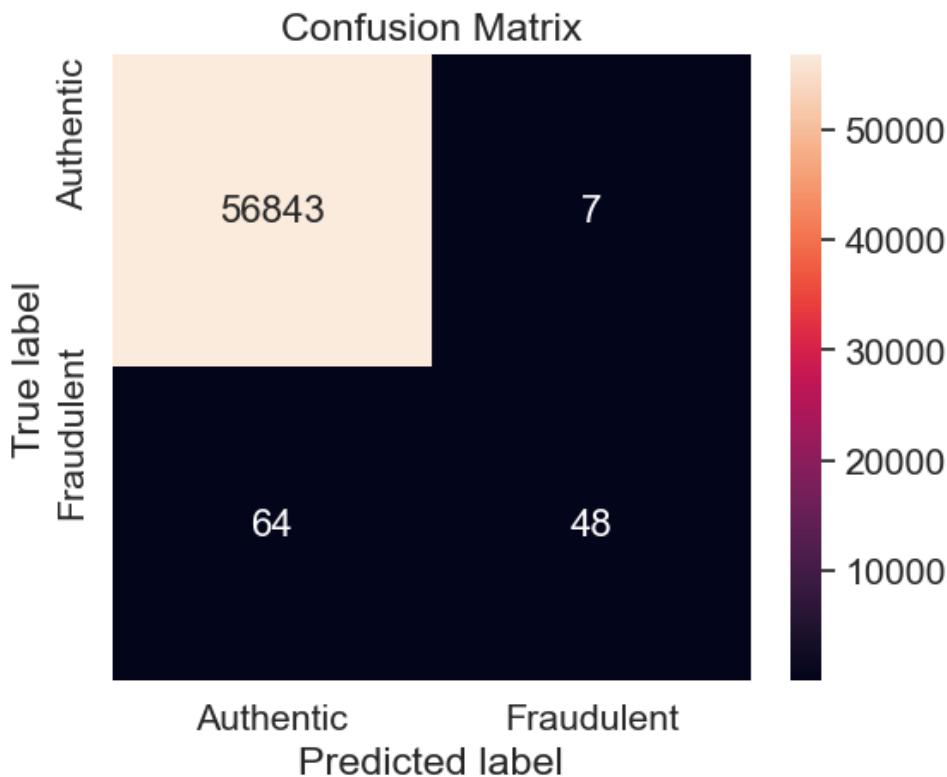
summary_ridge_unaltered = summary.copy()
summary_ridge_unaltered.set_index('Metric')

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', average_precision]
summary_ridge_unaltered_extended.set_index('Metric')

summary_ridge_unaltered_index = summary_ridge_unaltered_extended.T
summary_ridge_unaltered_index.columns = summary_ridge_unaltered_index.iloc[0]
```

```
summary_ridge_unaltered_index.drop(summary_ridge_unaltered_index.index[0], inplace = True)
summary_ridge_unaltered_index
```



```
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	0.611095	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)

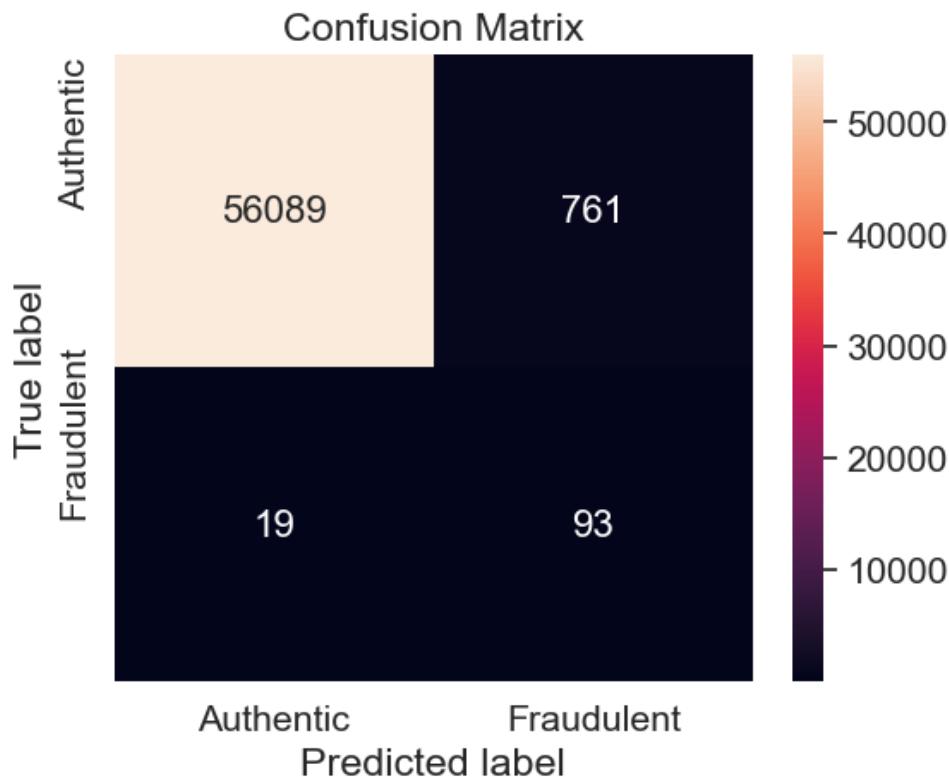
# Resumo das métricas de avaliação

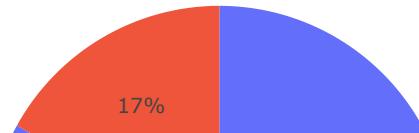
summary_ridge_under = summary.copy()
summary_ridge_under.set_index('Metric')

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_precision]
summary_ridge_under_extended.set_index('Metric')

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





```
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	0.29781	0.192547	0.357143	0.830357	0.108899	0.300708	0.986614	0.90512	0.131803	0

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)

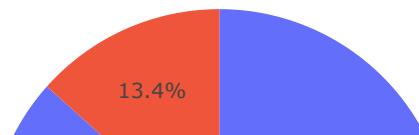
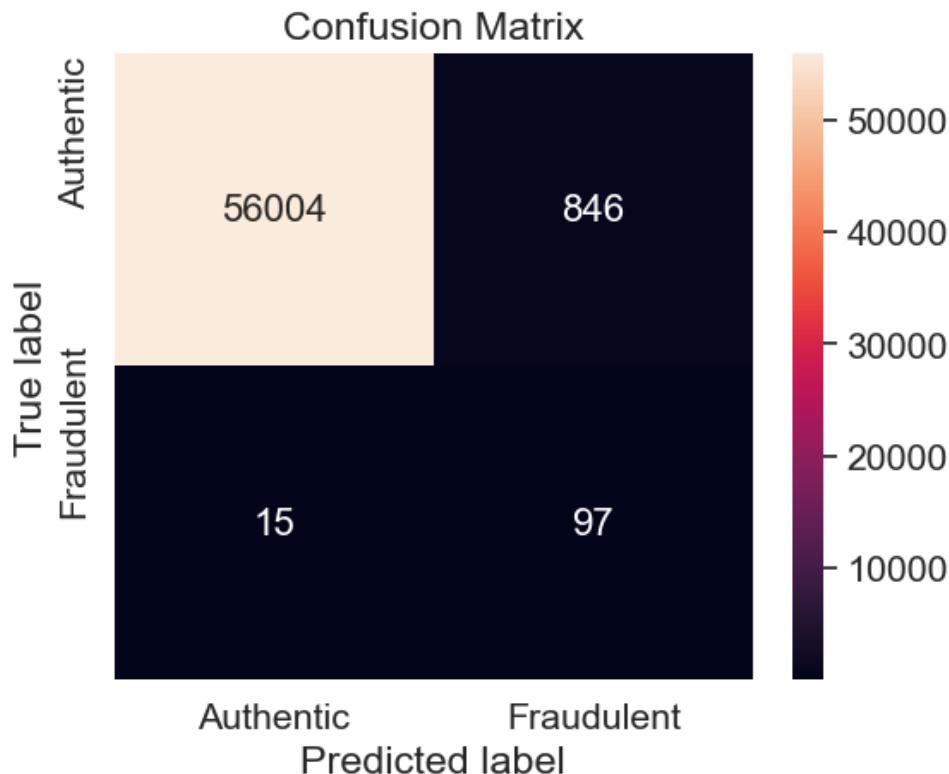
# Resumo das métricas de avaliação

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_precision]
summary_ridge_over_extended.set_index('Metric')

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



```
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		0.295513	0.183886	0.34867	0.866071	0.102863	0.298474	0.985119	0.923679	0.124871

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

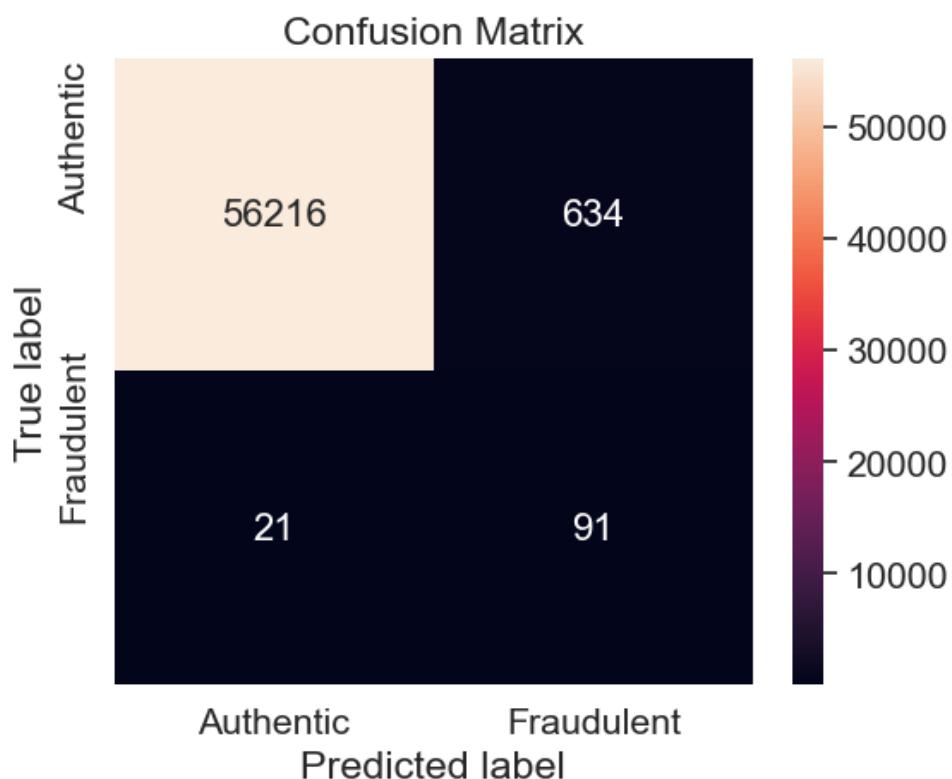
# Resumo das métricas de avaliação

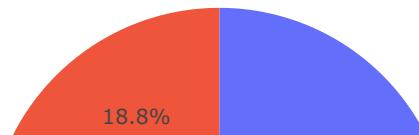
summary_ridge_under_imblearn = summary.copy()
summary_ridge_under_imblearn.set_index('Metric')

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 = True
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_

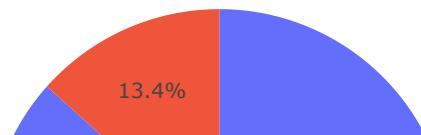
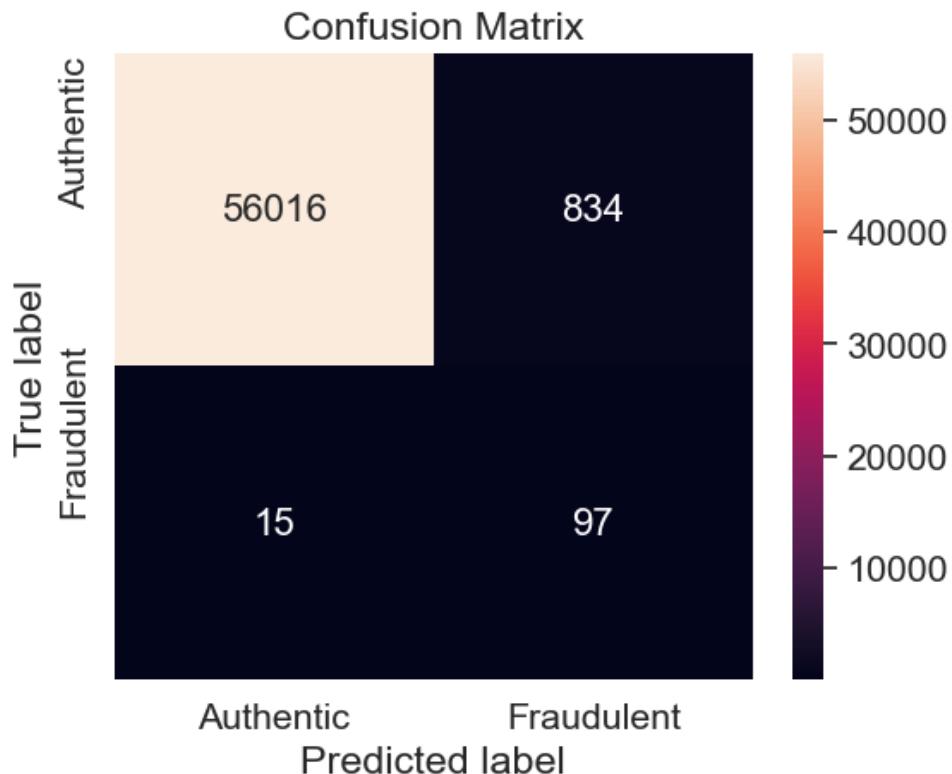
# Resumo das métricas de avaliação

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 = True)
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

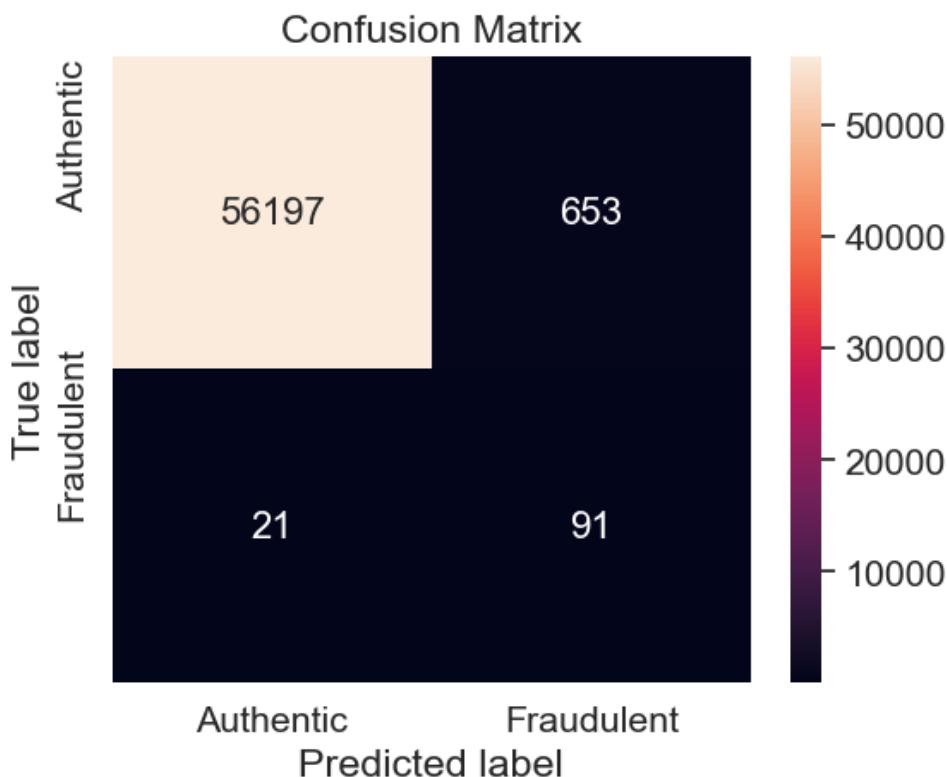
# Resumo das métricas de avaliação

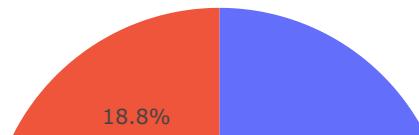
summary_ridge_over_smote = summary.copy()
summary_ridge_over_smote.set_index('Metric')

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	0.312529	0.212617	0.381711	0.8125	0.122312	0.315243	0.988514	0.896196	0.147345	0

Under-sampling via NearMiss

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

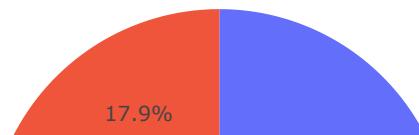
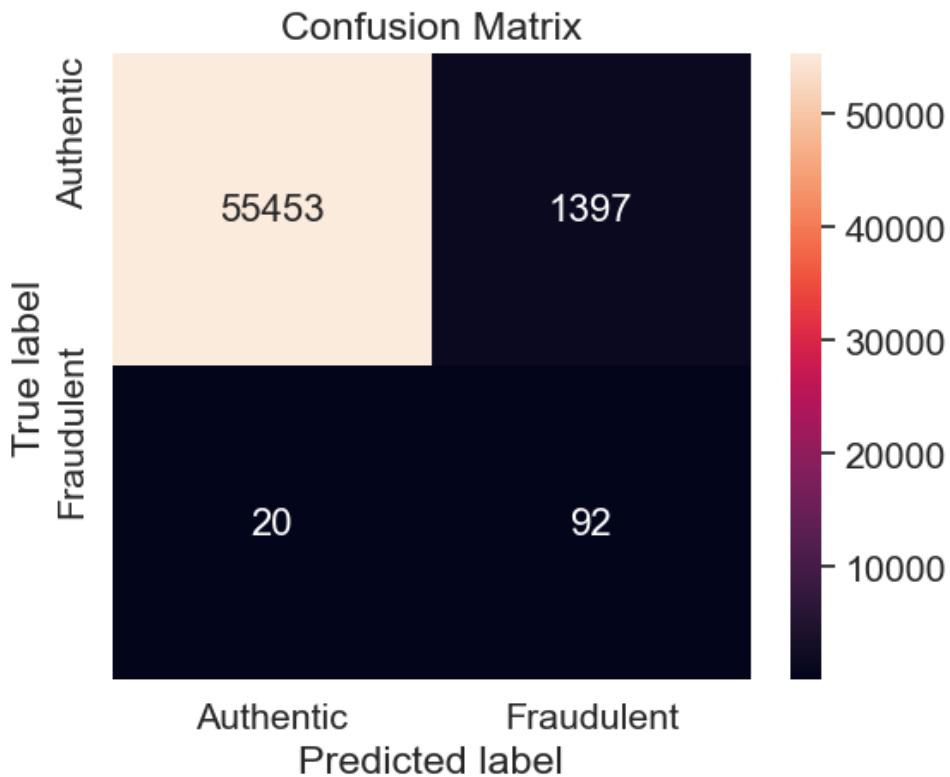
classification(ridge, X_train_under_nm_scaled_minmax, y_train_under_nm, X_test_scaled_minmax, y_
# Resumo das métricas de avaliação

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_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\Diônes\AppData\Local\Temp\ipykernel_18520\3761459512.py:8: FutureWarning:

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

C:\Users\Diônes\AppData\Local\Temp\ipykernel_18520\3761459512.py:8: FutureWarning:

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

Out[]:

Metric	Unaltered	RUS	ROS	RUS-IL	ROS-IL	SMOTE	NM
MCC	0.611095	0.297810	0.295513	0.316676	0.297454	0.312529	0.221241
F1-Score	0.574850	0.192547	0.183886	0.217443	0.186002	0.212617	0.114928
F2-Score	0.477137	0.357143	0.348670	0.387894	0.351704	0.381711	0.237481
Recall	0.428571	0.830357	0.866071	0.812500	0.866071	0.812500	0.821429
Precision	0.872727	0.108899	0.102863	0.125517	0.104189	0.122312	0.061786
FM index	0.611577	0.300708	0.298474	0.319347	0.300392	0.315243	0.225285
Specificity	0.999877	0.986614	0.985119	0.988848	0.985330	0.988514	0.975427
G-mean	0.654613	0.905120	0.923679	0.896348	0.923778	0.896196	0.895122
F0.5-Score	0.722892	0.131803	0.124871	0.151062	0.126434	0.147345	0.075808
Accuracy	0.998754	0.986307	0.984885	0.988501	0.985095	0.988168	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
"""

models = ['LogReg (Unaltered)', 'LogReg (RUS)', 'LogReg (ROS)', 'LogReg (RUS-IL)', 'LogReg (ROS-IL)',
          'KNN (Unaltered)', 'KNN (RUS)', 'KNN (ROS)', 'KNN (RUS-IL)', 'KNN (ROS-IL)', 'KNN (SMOTE)',
          'Dec Tree (Unaltered)', 'Dec Tree (RUS)', 'Dec Tree (ROS)', 'Dec Tree (RUS-IL)', 'Dec Tree',
          'SVM (Unaltered)', 'SVM (RUS)', 'SVM (ROS)', 'SVM (RUS-IL)', 'SVM (ROS-IL)', 'SVM (SMOTE)',
          'Naive Bayes (Unaltered)', 'Naive Bayes (RUS)', 'Naive Bayes (ROS)', 'Naive Bayes (RUS-IL',
          'RForest (Unaltered)', 'RForest (RUS)', 'RForest (ROS)', 'RForest (RUS-IL)', 'RForest (ROS',
          'LDA (Unaltered)', 'LDA (RUS)', 'LDA (ROS)', 'LDA (RUS-IL)', 'LDA (ROS-IL)', 'LDA (SMOTE)',
          'SGD (Unaltered)', 'SGD (RUS)', 'SGD (ROS)', 'SGD (RUS-IL)', 'SGD (ROS-IL)', 'SGD (SMOTE)',
          'Ridge (Unaltered)', 'Ridge (RUS)', 'Ridge (ROS)', 'Ridge (RUS-IL)', 'Ridge (ROS-IL)', 'Rid
```

```
metrics = ['F2-Score', 'MCC', 'Recall', 'F1-Score', 'Precision', 'FM index', 'Specificity', 'G-mean']

summary_list = [summary_logreg_unaltered_index, summary_logreg_under_index, summary_logreg_over_index,
                summary_logreg_over_imblearn_index, summary_logreg_over_smote_index, summary_logreg_over_nm_index,
                summary_knn_unaltered_index, summary_knn_under_index, summary_knn_over_index, summary_knn_over_nm_index,
                summary_knn_over_imblearn_index, summary_knn_over_smote_index, summary_knn_under_nm_index,
                summary_dt_unaltered_index, summary_dt_under_index, summary_dt_over_index, summary_dt_over_nm_index,
                summary_dt_over_imblearn_index, summary_dt_over_smote_index, summary_dt_over_nm_index,
                summary_svm_linear_unaltered_index, summary_svm_linear_over_index, summary_svm_linear_over_smote_index,
                summary_nb_unaltered_index, summary_nb_under_index, summary_nb_over_index, summary_nb_over_nm_index,
                summary_nb_over_imblearn_index, summary_nb_over_smote_index, summary_nb_over_nm_index,
                summary_rf_unaltered_index, summary_rf_under_index, summary_rf_over_index, summary_rf_over_nm_index,
                summary_rf_over_imblearn_index, summary_rf_over_smote_index, summary_rf_over_nm_index,
                summary_lda_unaltered_index, summary_lda_under_index, summary_lda_over_index, summary_lda_over_nm_index,
                summary_lda_over_imblearn_index, summary_lda_over_smote_index, summary_lda_over_nm_index,
                summary_sgd_unaltered_index, summary_sgd_under_index, summary_sgd_over_index, summary_sgd_over_nm_index,
                summary_sgd_over_imblearn_index, summary_sgd_over_smote_index, summary_sgd_over_nm_index,
                summary_ridge_unaltered_index, summary_ridge_under_index, summary_ridge_over_index,
                summary_ridge_over_imblearn_index, summary_ridge_over_smote_index, summary_ridge_over_nm_index]

pd.set_option('display.max_rows', 70)
model_comparation = pd.concat(summary_list, ignore_index=True)
model_comparation = model_comparation[model_comparation['MCC'] != 'MCC']
model_comparation = model_comparation[['models', 'F2-Score', 'MCC', 'Recall', 'F1-Score', 'Precision']]
model_comparation_descending_F2 = model_comparation.sort_values(by = ['F2-Score'], ascending = False)
model_comparation_descending_F2_idx = model_comparation_descending_F2.set_index('models')
model_comparation_descending_F2_idx
```

Out[]:

Metric	F2-Score	MCC	Recall	F1-Score	Precision	FM index	Specificity	G-mean	Accuracy
models									
RForest (SMOTE)	0.880783	0.875894	0.883929	0.876106	0.868421	0.876141	0.999736	0.940051	0.999508
RForest (ROS)	0.880218	0.902936	0.866071	0.902326	0.941748	0.903117	0.999894	0.93058	0.999631
RForest (ROS-IL)	0.872727	0.897988	0.857143	0.897196	0.941176	0.898177	0.999894	0.925771	0.999614
RForest (Unaltered)	0.857401	0.871645	0.848214	0.87156	0.896226	0.87189	0.999807	0.920896	0.999508
SVM (Unaltered)	0.857143	0.856861	0.857143	0.857143	0.857143	0.857143	0.999719	0.92569	0.999438
LDA (Unaltered)	0.849732	0.851736	0.848214	0.852018	0.855856	0.852027	0.999719	0.920856	0.999421
Dec Tree (ROS-IL)	0.827338	0.836186	0.821429	0.836364	0.851852	0.836502	0.999719	0.906199	0.999368
KNN (Unaltered)	0.804067	0.852191	0.776786	0.84878	0.935484	0.85245	0.999894	0.881308	0.999456
Dec Tree (ROS)	0.801802	0.812638	0.794643	0.812785	0.831776	0.812997	0.999683	0.891286	0.99928
Dec Tree (Unaltered)	0.774648	0.758582	0.785714	0.758621	0.733333	0.759072	0.999437	0.886156	0.999017
KNN (RUS-IL)	0.760369	0.655732	0.883929	0.628571	0.487685	0.656566	0.998171	0.939314	0.997946
SGD (Unaltered)	0.751391	0.801991	0.723214	0.79803	0.89011	0.802334	0.999824	0.850345	0.99928
KNN (RUS)	0.743551	0.6366	0.875	0.606811	0.464455	0.637494	0.998012	0.934484	0.99777
Dec Tree (SMOTE)	0.731895	0.632268	0.848214	0.607029	0.472637	0.633165	0.998135	0.920126	0.997841
KNN (ROS-IL)	0.699446	0.575425	0.901786	0.523316	0.368613	0.57655	0.996957	0.948178	0.99677
KNN (ROS)	0.699446	0.575425	0.901786	0.523316	0.368613	0.57655	0.996957	0.948178	0.99677
LogReg (Unaltered)	0.673624	0.75442	0.633929	0.743455	0.898734	0.754807	0.999859	0.79614	0.99914
KNN (NM)	0.671233	0.550216	0.875	0.497462	0.347518	0.551433	0.996763	0.933899	0.996524
KNN (SMOTE)	0.610641	0.488713	0.901786	0.411405	0.266491	0.490222	0.99511	0.947299	0.994926
SVM (SMOTE)	0.524152	0.418728	0.910714	0.320251	0.194286	0.420641	0.992559	0.950757	0.992398
SGD (SMOTE)	0.495098	0.396981	0.901786	0.295322	0.176573	0.399038	0.991715	0.945682	0.991538
Ridge (Unaltered)	0.477137	0.611095	0.428571	0.57485	0.872727	0.611577	0.999877	0.654613	0.998754
Naive Bayes (SMOTE)	0.46476	0.370966	0.8125	0.283048	0.171375	0.373151	0.99226	0.897893	0.991907
SVM (RUS-IL)	0.452946	0.368501	0.919643	0.257179	0.149492	0.370782	0.989692	0.954025	0.989554

Metric	F2-Score	MCC	Recall	F1-Score	Precision	FM index	Specificity	G-mean	Accuracy
models									
Naive Bayes (Unaltered)	0.431894	0.343272	0.696429	0.275132	0.171429	0.345525	0.993369	0.831751	0.992785
Naive Bayes (ROS-IL)	0.431115	0.347016	0.821429	0.25171	0.148627	0.349409	0.99073	0.902116	0.990397
Naive Bayes (ROS)	0.428705	0.34532	0.821429	0.249661	0.1472	0.347727	0.990624	0.902068	0.990292
Naive Bayes (RUS-IL)	0.416667	0.336357	0.8125	0.240741	0.141304	0.338836	0.990273	0.896993	0.989923
SVM (NM)	0.393731	0.329246	0.919643	0.211934	0.119767	0.331878	0.986684	0.952574	0.986552
Ridge (RUS-IL)	0.387894	0.316676	0.8125	0.217443	0.125517	0.319347	0.988848	0.896348	0.988501
Ridge (SMOTE)	0.381711	0.312529	0.8125	0.212617	0.122312	0.315243	0.988514	0.896196	0.988168
LDA (SMOTE)	0.380435	0.311676	0.8125	0.211628	0.121658	0.314399	0.988443	0.896164	0.988097
SVM (RUS)	0.372918	0.315861	0.919643	0.197129	0.110397	0.318631	0.9854	0.951954	0.985271
SGD (RUS)	0.372918	0.315861	0.919643	0.197129	0.110397	0.318631	0.9854	0.951954	0.985271
Naive Bayes (RUS)	0.361635	0.299979	0.821429	0.196581	0.11165	0.302841	0.987124	0.900473	0.986798
Ridge (RUS)	0.357143	0.29781	0.830357	0.192547	0.108899	0.300708	0.986614	0.90512	0.986307
Ridge (ROS-IL)	0.351704	0.297454	0.866071	0.186002	0.104189	0.300392	0.98533	0.923778	0.985095
LDA (ROS-IL)	0.350941	0.296965	0.866071	0.185468	0.103854	0.299909	0.985277	0.923753	0.985043
Ridge (ROS)	0.34867	0.295513	0.866071	0.183886	0.102863	0.298474	0.985119	0.923679	0.984885
LDA (ROS)	0.347421	0.294716	0.866071	0.183019	0.102321	0.297686	0.985031	0.923638	0.984797
SGD (RUS-IL)	0.342782	0.297584	0.928571	0.176122	0.097287	0.300563	0.983026	0.955411	0.982918
SGD (ROS-IL)	0.325407	0.286672	0.928571	0.164818	0.090435	0.289785	0.981601	0.954718	0.981496
LDA (RUS-IL)	0.315789	0.273827	0.857143	0.162162	0.089552	0.277054	0.982832	0.917838	0.982585
SGD (NM)	0.312876	0.278828	0.928571	0.156863	0.085667	0.282043	0.980475	0.95417	0.980373
SGD (ROS)	0.296973	0.268888	0.928571	0.146996	0.079816	0.27224	0.978909	0.953408	0.97881
LogReg (SMOTE)	0.295139	0.26601	0.910714	0.146552	0.079687	0.269393	0.979279	0.944375	0.979144
LDA (RUS)	0.277143	0.250432	0.866071	0.137199	0.074501	0.254014	0.978804	0.920714	0.978582
SVM (ROS-IL)	0.271195	0.251899	0.919643	0.131798	0.070986	0.255502	0.976288	0.947542	0.976177

Metric	F2-Score	MCC	Recall	F1-Score	Precision	FM index	Specificity	G-mean	Accuracy
models									
SVM (ROS)	0.27091	0.25172	0.919643	0.131629	0.070888	0.255326	0.976253	0.947525	0.976142
Ridge (NM)	0.237481	0.221241	0.821429	0.114928	0.061786	0.225285	0.975427	0.895122	0.975124
RForest (RUS)	0.215768	0.217619	0.928571	0.100289	0.053007	0.221858	0.967318	0.947746	0.967241
RForest (RUS-IL)	0.209087	0.213297	0.928571	0.096699	0.051005	0.217628	0.965963	0.947083	0.96589
LogReg (ROS-IL)	0.157911	0.17903	0.928571	0.070342	0.036555	0.18424	0.951785	0.940107	0.95174
LogReg (RUS)	0.157911	0.17903	0.928571	0.070342	0.036555	0.18424	0.951785	0.940107	0.95174
LogReg (ROS)	0.157576	0.178796	0.928571	0.070175	0.036466	0.184013	0.951662	0.940046	0.951617
LogReg (RUS-IL)	0.15691	0.178332	0.928571	0.069846	0.036288	0.183563	0.951416	0.939924	0.951371
LDA (NM)	0.133333	0.155659	0.857143	0.058824	0.030457	0.161573	0.946245	0.900592	0.946069
Dec Tree (RUS-IL)	0.083822	0.121271	0.919643	0.035468	0.018083	0.128956	0.901618	0.910586	0.901654
Dec Tree (RUS)	0.081765	0.121223	0.946429	0.034494	0.017567	0.128942	0.895726	0.920728	0.895825
LogReg (NM)	0.040893	0.078367	0.955357	0.016788	0.008469	0.089947	0.779631	0.863033	0.779976
RForest (NM)	0.031371	0.064833	0.955357	0.012801	0.006443	0.078459	0.70978	0.823464	0.710263
Naive Bayes (NM)	0.029813	0.057099	0.839286	0.012185	0.006137	0.071768	0.732225	0.78393	0.732436
Dec Tree (NM)	0.015122	0.032723	0.991071	0.006105	0.003062	0.055086	0.364257	0.600837	0.365489

In []: `model_comparation_descending_F2_idx.head(10)`

Out[]:	Metric	F2-Score	MCC	Recall	F1-Score	Precision	FM index	Specificity	G-mean	Accuracy
models										
	RForest (SMOTE)	0.880783	0.875894	0.883929	0.876106	0.868421	0.876141	0.999736	0.940051	0.999508
	RForest (ROS)	0.880218	0.902936	0.866071	0.902326	0.941748	0.903117	0.999894	0.93058	0.999631
	RForest (ROS-IL)	0.872727	0.897988	0.857143	0.897196	0.941176	0.898177	0.999894	0.925771	0.999614
	RForest (Unaltered)	0.857401	0.871645	0.848214	0.87156	0.896226	0.87189	0.999807	0.920896	0.999508
	SVM (Unaltered)	0.857143	0.856861	0.857143	0.857143	0.857143	0.857143	0.999719	0.92569	0.999438
	LDA (Unaltered)	0.849732	0.851736	0.848214	0.852018	0.855856	0.852027	0.999719	0.920856	0.999421
	Dec Tree (ROS-IL)	0.827338	0.836186	0.821429	0.836364	0.851852	0.836502	0.999719	0.906199	0.999368
	KNN (Unaltered)	0.804067	0.852191	0.776786	0.84878	0.935484	0.85245	0.999894	0.881308	0.999456
	Dec Tree (ROS)	0.801802	0.812638	0.794643	0.812785	0.831776	0.812997	0.999683	0.891286	0.99928
	Dec Tree (Unaltered)	0.774648	0.758582	0.785714	0.758621	0.733333	0.759072	0.999437	0.886156	0.999017

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.

Temos que Destacar que os modelos de **Over Sampling** tiveram melhores resultados que os de under sampling. ** Outro Ponto em questão** é a facilidade de cometer erros a escolha do modelo, visto que modelos que não efetuam o balanceamento também inicialmente dão bons resultados, pois olham para o resultado como um todo. mas deve-se olhar o que estamos buscando, queremos as fraudes e não saber as não fraudes, logo esse treinamento sem alteração pode ser muito enviesado para ser aplicado no dia a dia.