

Teste dos 2 melhores modelos nos 8 materiais com maior número de registros

Define funções a serem utilizadas

```
In [1]: import numpy as np
import pandas as pd
import os
import itertools
import time
from sklearn.base import clone
from sklearn.model_selection import LeaveOneOut
from sklearn.metrics import accuracy_score, recall_score, f1_score
from sklearn.model_selection import StratifiedKFold
from sklearn.metrics import confusion_matrix
from sklearn.metrics import confusion_matrix, ConfusionMatrixDisplay
from sklearn.metrics import f1_score, accuracy_score, precision_score, recall_score
import matplotlib.pyplot as plt

# Importa bibliotecas do PyOD com os algoritmos de detecção de anomalias
from pyod.models.copod import COPOD

def carrega_material(path, material):
    arquivo = str(material) + ".csv"
    csv_path = os.path.join(path, arquivo)
    df = pd.read_csv(csv_path)
    ## Retira os campos que não serão utilizados no treinamento
    df = df.drop(['licitacao_contrato'], axis=1)
    df = df.drop(['id'], axis=1)
    df = df.drop(['data'], axis=1)
    df = df.drop(['catmat_id'], axis=1)
    df = df.drop(['unidade'], axis=1)
    df = df.drop(['valor_total'], axis=1)
    df = df.drop(['municipio_uasg'], axis=1)
    df = df.drop(['municipio_fornecedor'], axis=1)
    return df

def avalia_algoritmo(clf_name, Y, y_train_pred):
    # Avalia e imprime os resultados
    tn, fp, fn, tp = confusion_matrix(Y, y_train_pred).ravel()
    ac = (" {:.2f} ".format(round(accuracy_score(Y, y_train_pred)*100, 2))).replace(".", "")
    print('Acurácia=' + ac)
    pc = (" {:.2f} ".format(round(precision_score(Y, y_train_pred)*100, 2))).replace(".", "")
    print('Precisão=' + pc)
    rc = (" {:.2f} ".format(round(recall_score(Y, y_train_pred)*100, 2))).replace(".", "")
    print('Recall=' + rc)
    f1 = (" {:.2f} ".format(round(f1_score(Y, y_train_pred)*100, 2))).replace(".", "")
    print('F1=' + f1)
    print('Falso Negativo=' + str(fn))
    print('Falso Positivo=' + str(fp))
```

#####

```
# Função gera_matriz_de_confusao
# Objetivo: Imprimir a matriz de confusão de uma avaliação feita de um algoritmo
# Parâmetros: Y - Label real esperado
#             y_avalidado - Valor determinado pelo algoritmo
#             algoritmo - Nome do algoritmo avaliado
# Retorno: não tem (a matriz é mostrada na saída)
#####
def gera_matriz_de_confusao(Y,y_avalidado,algoritmo):
    cm = confusion_matrix(Y,y_avalidado)
    disp = ConfusionMatrixDisplay(confusion_matrix=cm, display_labels=['Normal',
    disp.plot(cmap=plt.cm.Blues)
    disp.ax_.set(title='Matriz de Confusão do '+str(algoritmo),xlabel='Valores p
    plt.show()
```

Recupera os itens que servirão de dados das avaliações

```
In [2]: from sqlalchemy import create_engine
import pymysql
from sqlalchemy.dialects.mysql import insert
import pandas as pd
sqlEngine = create_engine('mysql+pymysql://siasg:siasg@192.168.2.135/siasg
dbConnection = sqlEngine.connect()
itens = pd.read_sql("SELECT * from siasg.itens", dbConnection);

# Recupera os oito materiais com maior número de registros
maioresitens = pd.read_sql("SELECT catmat_id, count(*) from siasg.itens where va
```

Avalia os modelos

```
In [ ]: # Define a tabela que irá gravar as avaliações
tabelafinal = pd.DataFrame(columns=['material','melhorContaminação','melhorrecal
print(tabelafinal)

for index, row in maioresitens.iterrows():
    catmat = row['catmat_id']
    df=carrega_material(".",catmat)
    print(catmat)

    # Separação do Label dos dados
    x = df.iloc[:, :-1]
    y = df.iloc[:, -1]

    # Contaminação
    contaminacao=0.12
    model = COPOD(contamination=contaminacao)

    # Treinar o modelo
    model.fit(x)

    # Fazer as previsões
    y_pred = model.predict(x)
```

```

model=COPOD(contamination=contaminacao)
model.fit(x)
y_pred = model.predict(x)
tn, fp, fn, tp = confusion_matrix(y,y_pred).ravel()
recall=recall_score(y, y_pred)
acuracia=accuracy_score(y, y_pred)

avalia_algoritmo(str(catmat),y,y_pred)
gera_matriz_de_confusao(y,y_pred,catmat)

```

```

In [4]: def testa_material(catmat):
        modelo = COPOD(contamination=0.12)

        df=carrega_material(".",catmat)
        print(catmat)

        # Separação do Label dos dados
        X = df.iloc[:, :-1]
        y = df.iloc[:, -1]

        loo = LeaveOneOut()

        y_verdadeiro, y_predito = [], []

        for treino_index, teste_index in loo.split(X):
            X_treino, X_teste = X.iloc[treino_index], X.iloc[teste_index]
            y_treino, y_teste = y.iloc[treino_index], y.iloc[teste_index]

            modelo_clonado = clone(modelo)
            modelo_clonado.fit(X_treino)
            y_pred = modelo_clonado.predict(X_teste)

            y_verdadeiro.append(y_teste)
            y_predito.append(y_pred)
        matriz_confusao = confusion_matrix(y_verdadeiro, y_predito)
        print(matriz_confusao)
        avalia_algoritmo(str(catmat),y_verdadeiro,y_predito)
        gera_matriz_de_confusao(y_verdadeiro,y_predito,catmat)

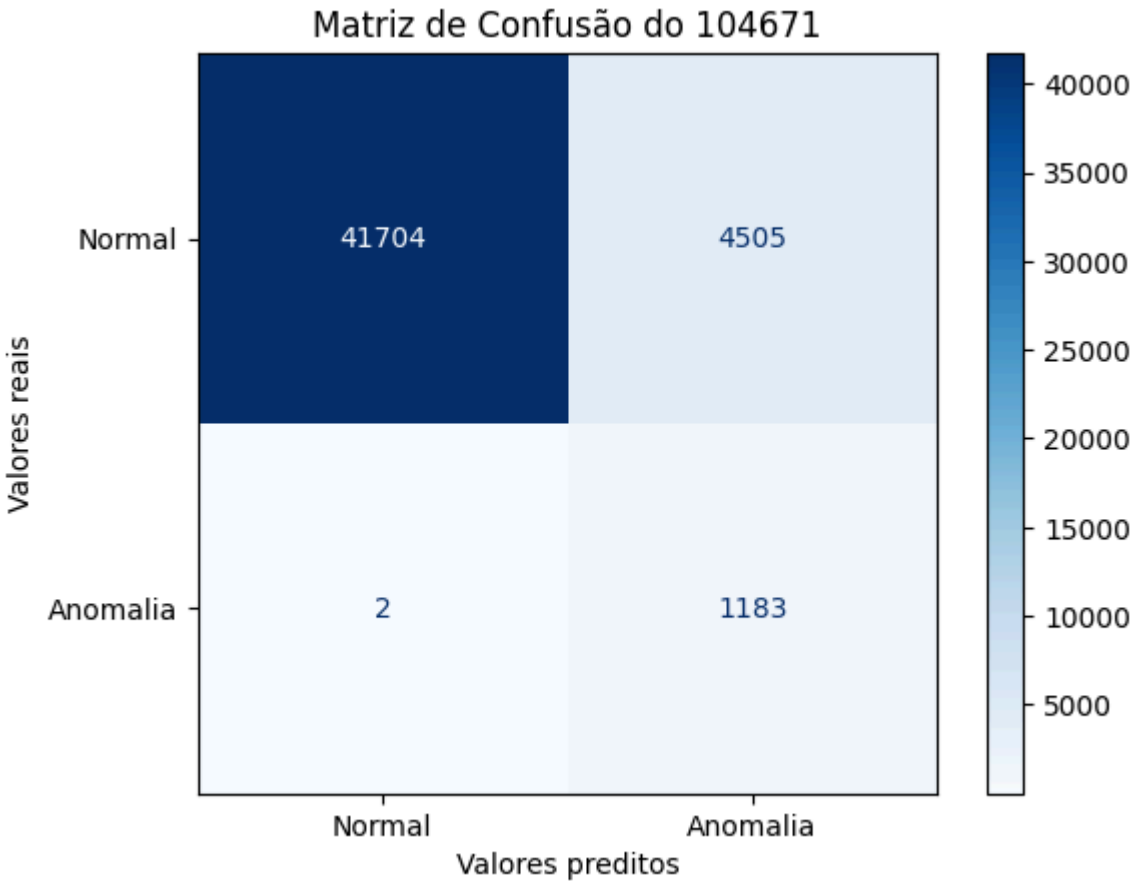
        for index, row in maioresitens.iterrows():
            catmat = row['catmat_id']
            testa_material(catmat)

```

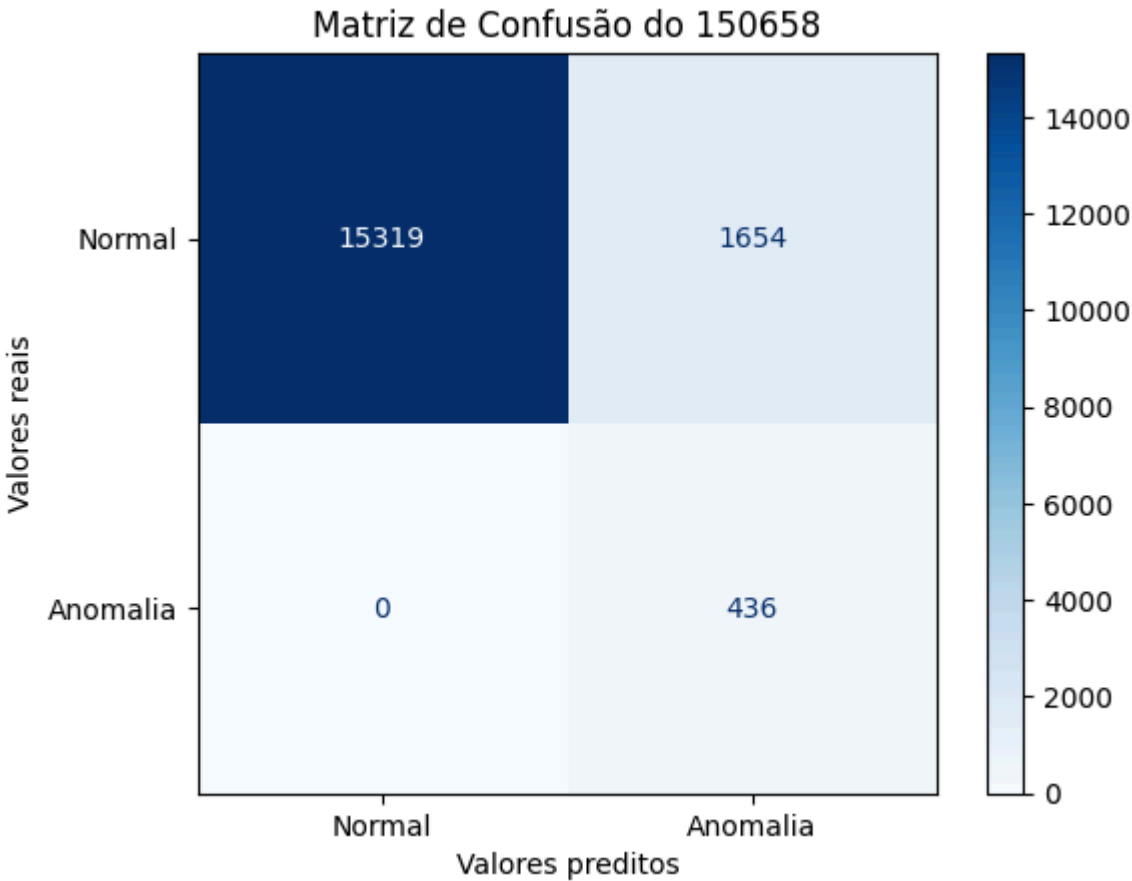
```

104671
[[41704  4505]
 [    2 1183]]
Acurácia=90,49
Precisão=20,80
Recall=99,83
F1=34,42
Falso Negativo=2
Falso Positivo=4505

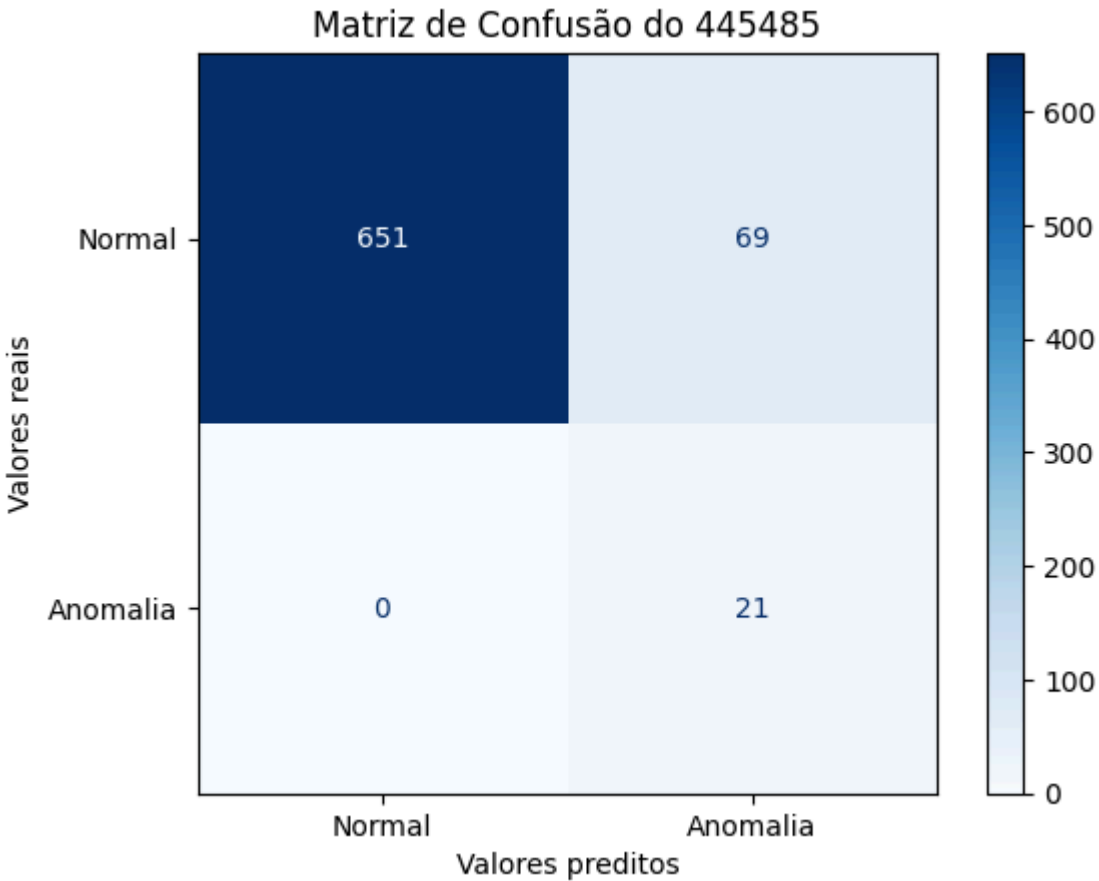
```



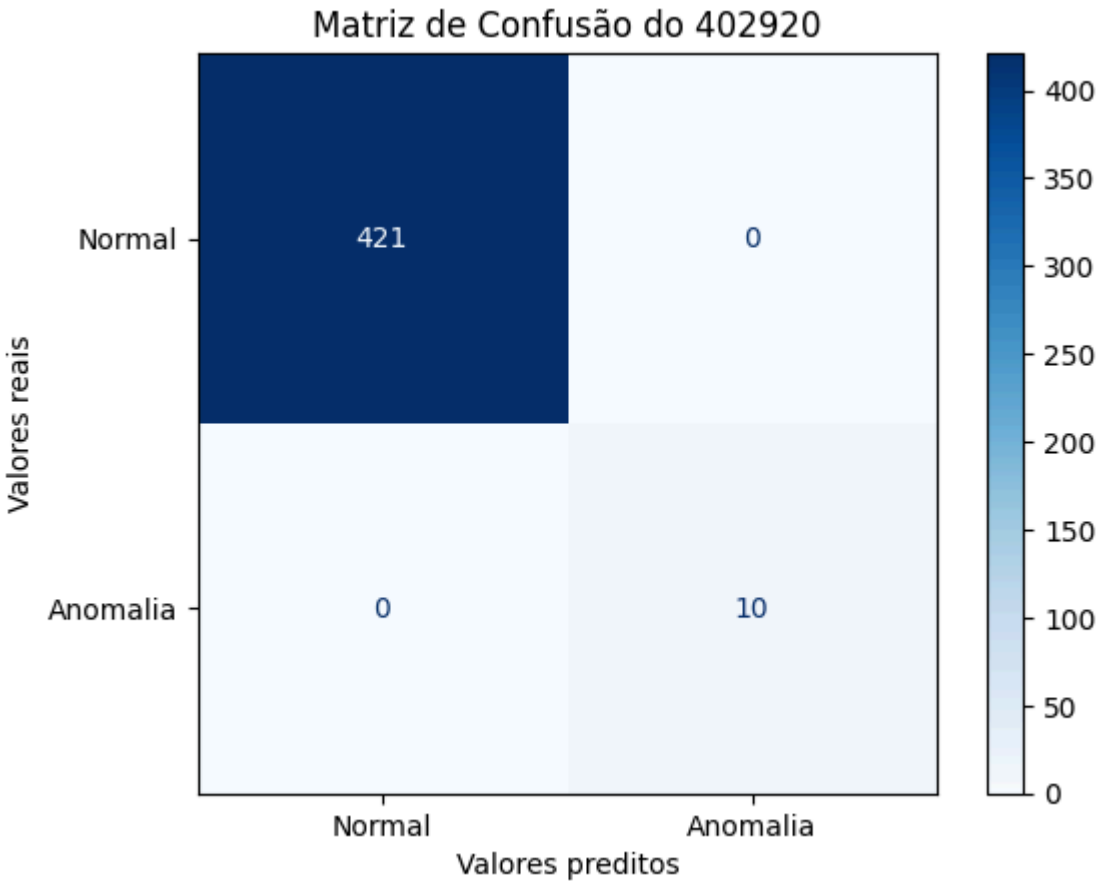
```
150658
[[15319 1654]
 [    0  436]]
Acurácia=90,50
Precisão=20,86
Recall=100,00
F1=34,52
Falso Negativo=0
Falso Positivo=1654
```



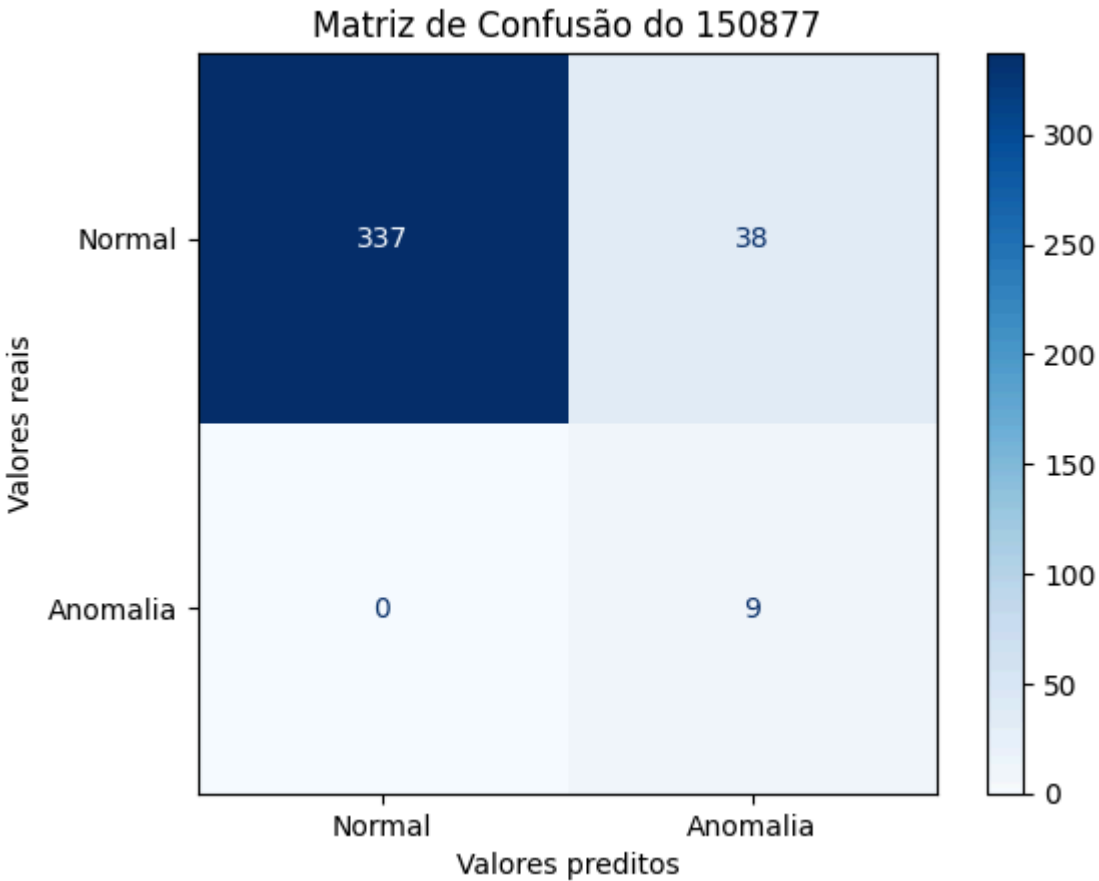
445485
[[651 69]
[0 21]]
Acurácia=90,69
Precisão=23,33
Recall=100,00
F1=37,84
Falso Negativo=0
Falso Positivo=69



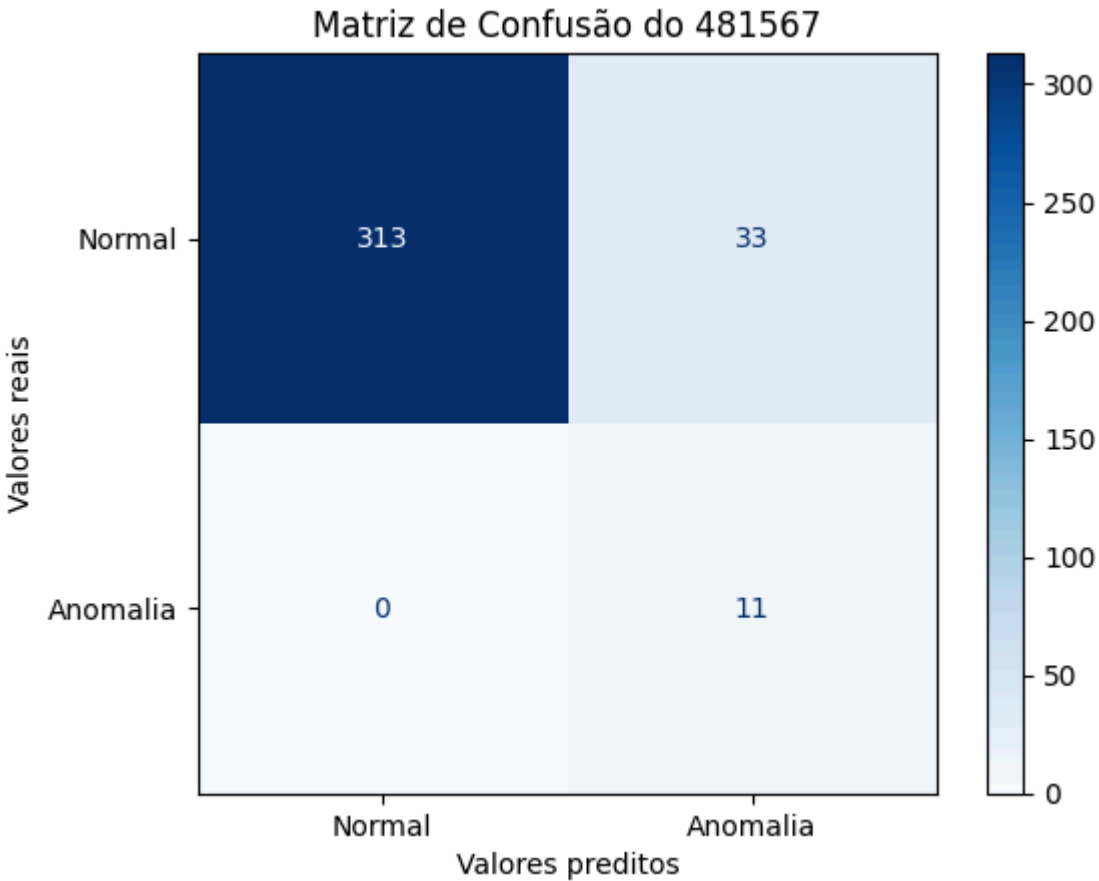
```
402920
[[421  0]
 [ 0 10]]
Acurácia=100,00
Precisão=100,00
Recall=100,00
F1=100,00
Falso Negativo=0
Falso Positivo=0
```



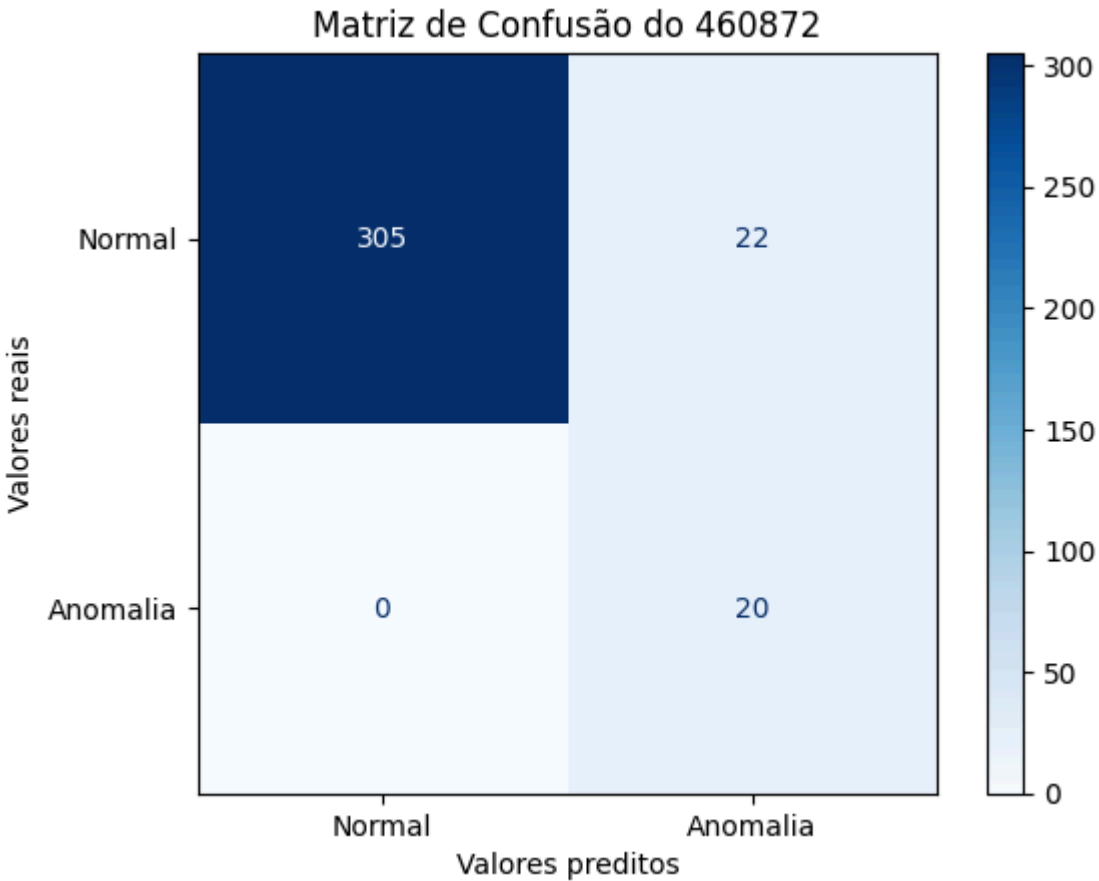
```
150877
[[337 38]
 [ 0  9]]
Acurácia=90,10
Precisão=19,15
Recall=100,00
F1=32,14
Falso Negativo=0
Falso Positivo=38
```



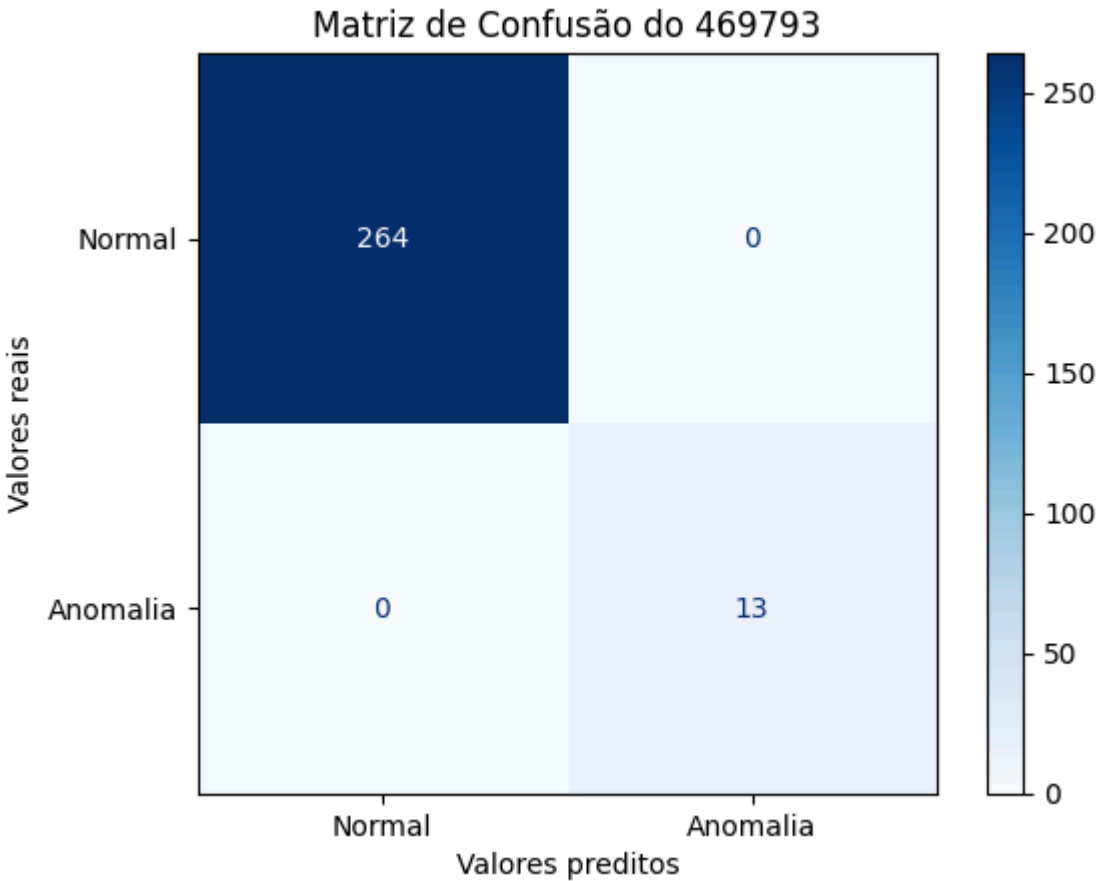
```
481567
[[313  33]
 [  0  11]]
Acurácia=90,76
Precisão=25,00
Recall=100,00
F1=40,00
Falso Negativo=0
Falso Positivo=33
```

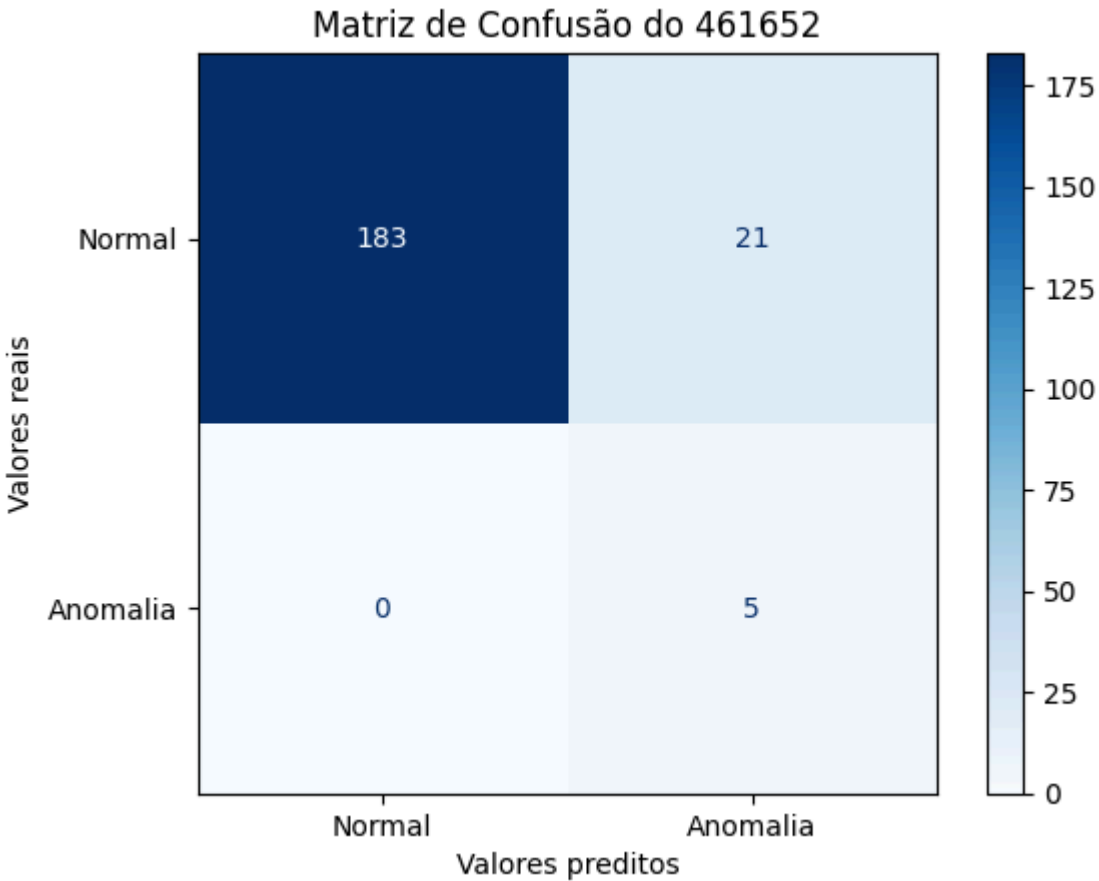
460872
[[305 22]
[0 20]]
Acurácia=93,66
Precisão=47,62
Recall=100,00
F1=64,52
Falso Negativo=0
Falso Positivo=22



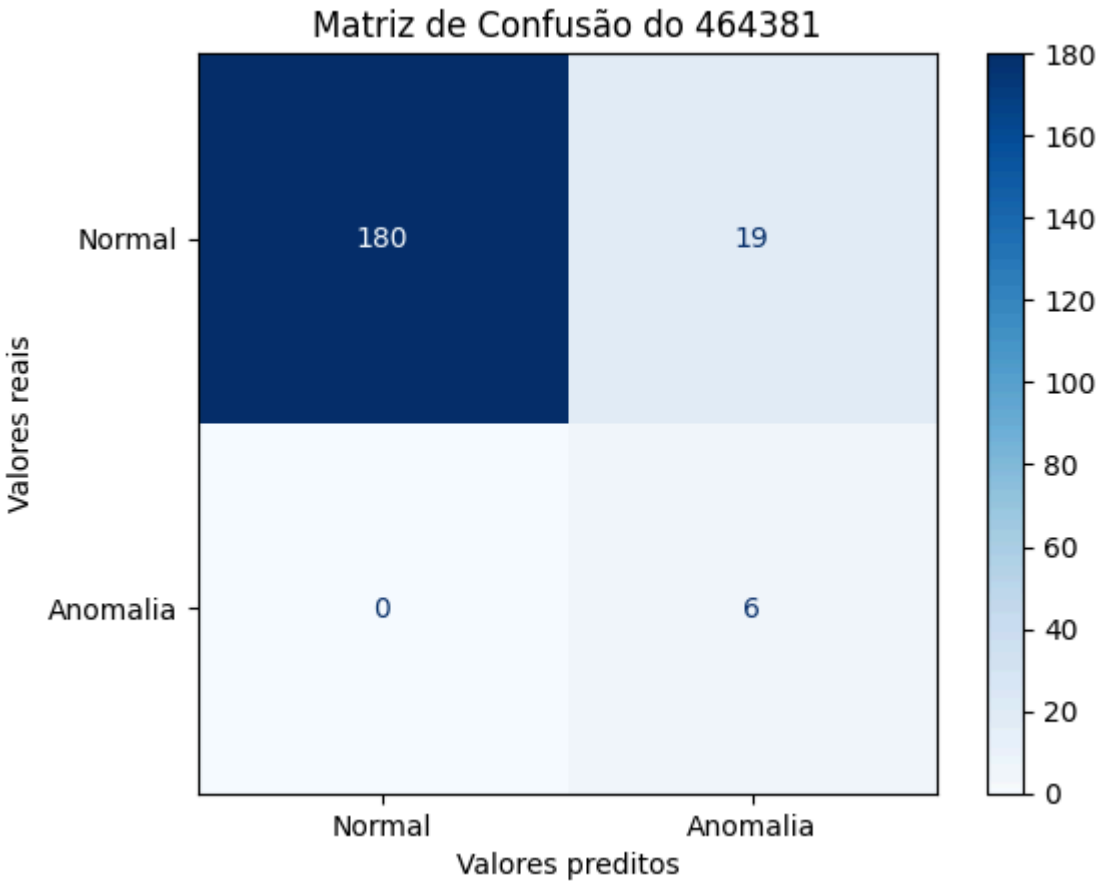
469793
[[264 0]
[0 13]]
Acurácia=100,00
Precisão=100,00
Recall=100,00
F1=100,00
Falso Negativo=0
Falso Positivo=0



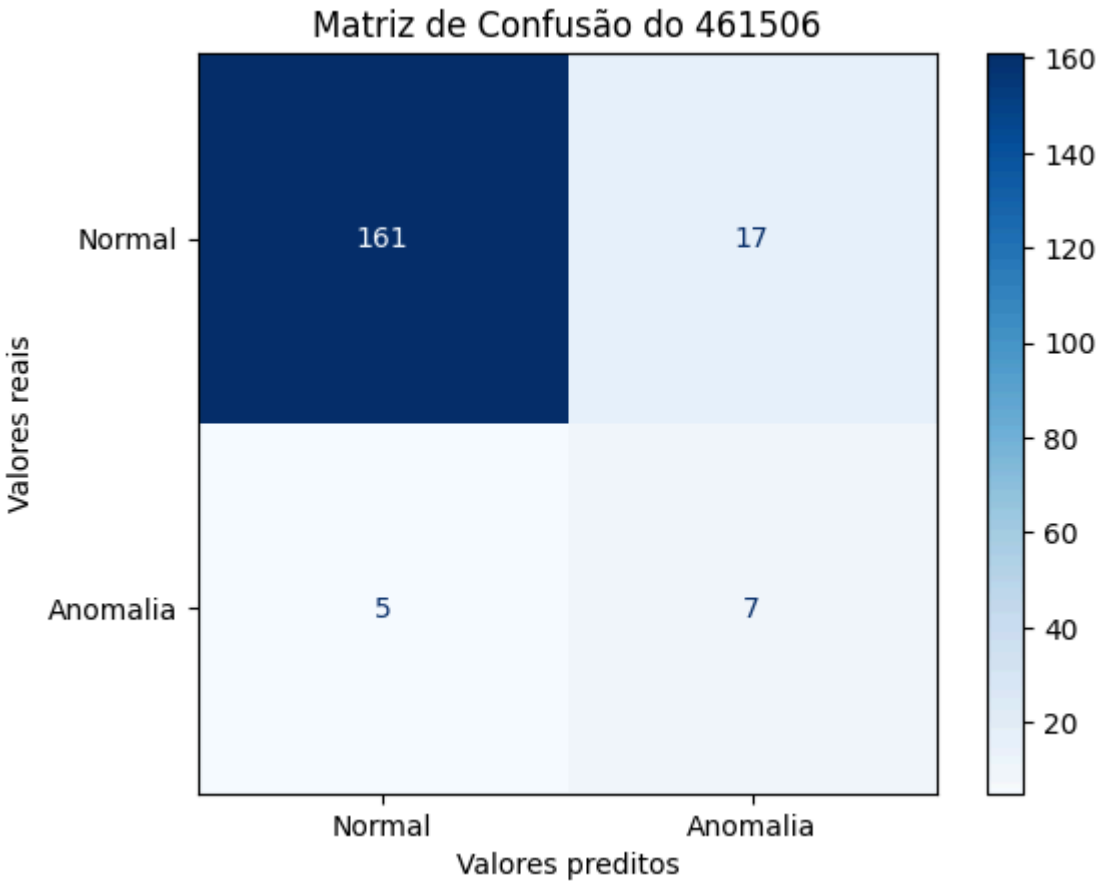
```
461652
[[183  21]
 [  0   5]]
Acurácia=89,95
Precisão=19,23
Recall=100,00
F1=32,26
Falso Negativo=0
Falso Positivo=21
```



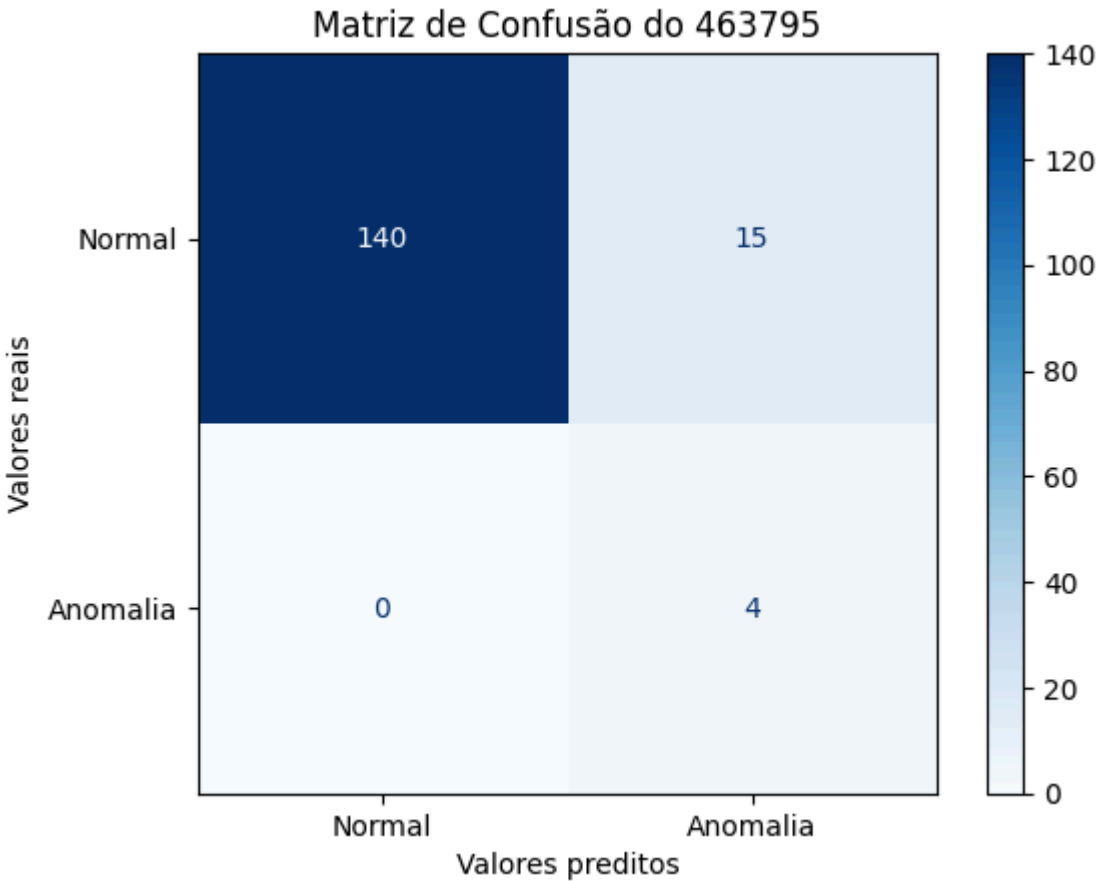
464381
[[180 19]
[0 6]]
Acurácia=90,73
Precisão=24,00
Recall=100,00
F1=38,71
Falso Negativo=0
Falso Positivo=19



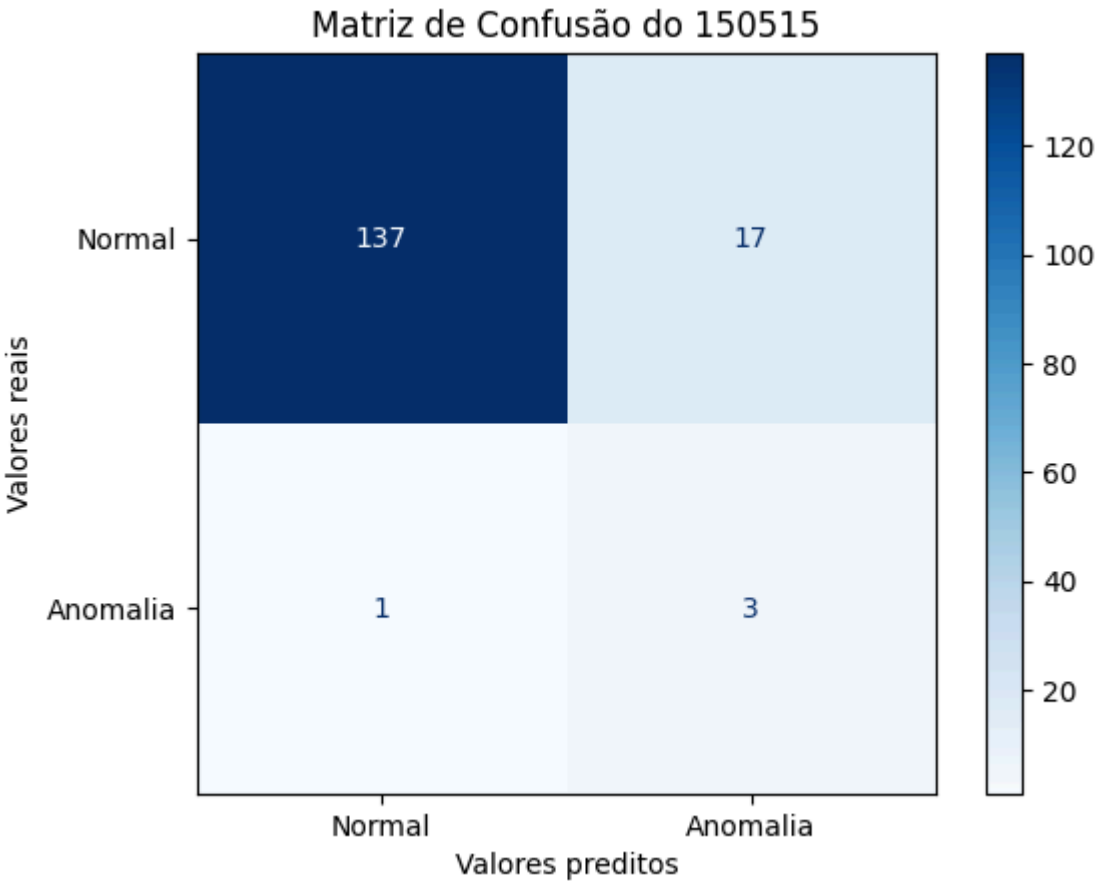
461506
[[161 17]
[5 7]]
Acurácia=88,42
Precisão=29,17
Recall=58,33
F1=38,89
Falso Negativo=5
Falso Positivo=17



463795
[[140 15]
[0 4]]
Acurácia=90,57
Precisão=21,05
Recall=100,00
F1=34,78
Falso Negativo=0
Falso Positivo=15



```
150515
[[137 17]
 [ 1  3]]
Acurácia=88,61
Precisão=15,00
Recall=75,00
F1=25,00
Falso Negativo=1
Falso Positivo=17
```



```
In [ ]:
```