Projeto Final

October 19, 2022

1 Bibliotecas Usadas

[3]:
```
from os.path import exists
from os import environ
environ['TF_CPP_MIN_LOG_LEVEL'] = '3'
from sklearn.neural_network import MLPClassifier
import project_metrics.project_metrics as metrics
import numpy as np
import pandas as pd
import pickle
import joblib


from keras.wrappers.scikit_learn import KerasClassifier
from sklearn.svm import SVC
from sklearn.ensemble import GradientBoostingClassifier,␣
 ↪RandomForestClassifier, VotingClassifier
from sklearn.neural_network import MLPClassifier
```

2 Separando os Dados

[4]:
```
import pandas as pd
```

3 Carregando os dados que Foram Previamente Separados

[ ]:
```
df_train = pd.read_csv("Assets/train.csv")
df_val= pd.read_csv("Assets/val.csv")
df_test = pd.read_csv("Assets/test.csv")

df_train.drop("INDEX", inplace=True, axis =1)
df_val.drop("INDEX", inplace=True, axis =1)
df_test.drop("INDEX", inplace=True, axis =1)

df_train.drop("Unnamed: 0", inplace=True, axis =1)
df_val.drop("Unnamed: 0", inplace=True, axis =1)
```

```
df_test.drop("Unnamed: 0", inplace=True, axis =1)

df_train.drop("IND_BOM_1_2", inplace=True, axis =1)
df_val.drop("IND_BOM_1_2", inplace=True, axis =1)
df_test.drop("IND_BOM_1_2", inplace=True, axis =1)

y_train = df_train["IND_BOM_1_1"].values
y_val = df_val["IND_BOM_1_1"].values
y_test = df_test["IND_BOM_1_1"].values

df_train.drop("IND_BOM_1_1", inplace=True, axis =1)
df_val.drop("IND_BOM_1_1", inplace=True, axis =1)
df_test.drop("IND_BOM_1_1", inplace=True, axis =1)

x_train = df_train.values
x_val = df_val.values
x_test = df_test.values

x_train.shape
```

4 Algoritmos de Aprendizado de Máquina

4.1 MLP

[5]:
```
from keras.models import Sequential
from keras.layers import Dense
from keras.callbacks import EarlyStopping
import keras

MODELS_PATH = "Models/MLP"
HISTORY_PATH = f"{MODELS_PATH}/history"
MODEL_PATH = f"{MODELS_PATH}/model"
```

4.1.1 Experimento1

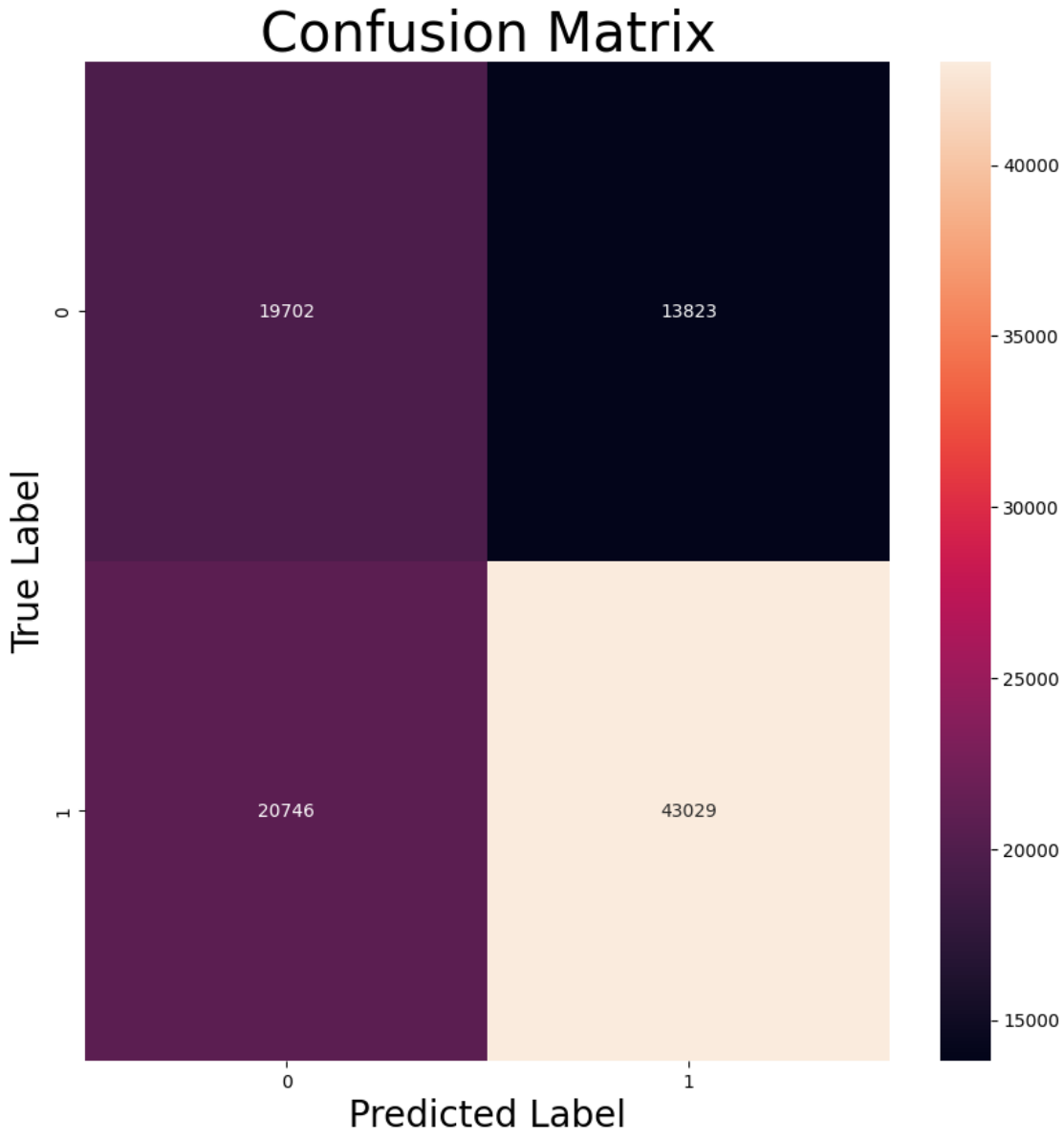
[6]:
```
input_dim = x_train.shape[1]

classifier = Sequential()

classifier.add(Dense(16, activation='relu', input_dim=input_dim))
classifier.add(Dense(1, activation='sigmoid'))
classifier.compile(optimizer='adam', loss='mean_squared_error')

history_file = HISTORY_PATH + "1.npy"
model_file = MODEL_PATH + "1"
```

```
if not exists(history_file) or  not exists(model_file):
    history = classifier.fit(x_train, y_train, batch_size=32, epochs=10_000,␣
 ↪callbacks=[EarlyStopping(patience=10,verbose=1)], validation_data=(x_val,␣
 ↪y_val))
    np.save(history_file,history.history)
    classifier.save(model_file)
else:
    print("Model was already trained")

history=np.load(history_file, allow_pickle='TRUE').item()
classifier = keras.models.load_model(model_file)
```

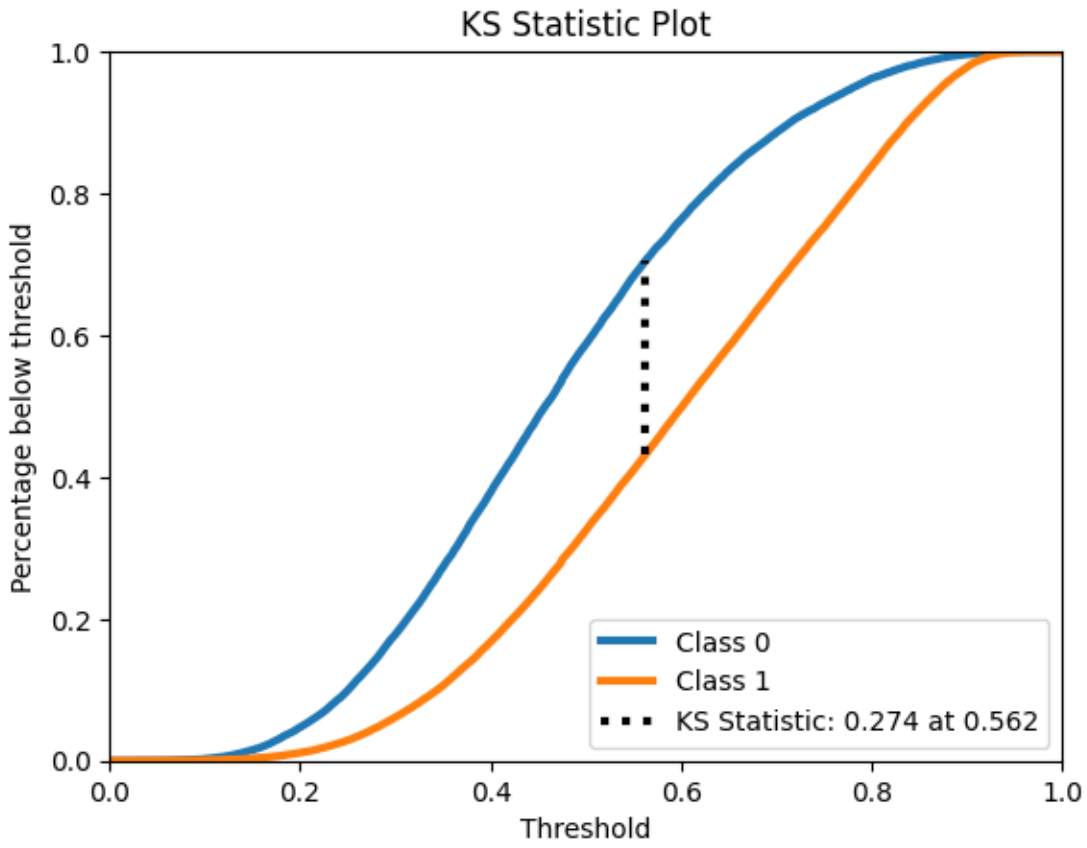
```
Model was already trained
```

[7]:
```
# Fazer predições no conjunto de teste
y_pred_scores = classifier.predict(x_test)
y_pred_class = (y_pred_scores > 0.5).astype("int32")
y_pred_scores_0 = 1 - y_pred_scores
y_pred_scores = np.concatenate([y_pred_scores_0, y_pred_scores], axis=1)

## Matriz de confusão
print('Matriz de confusão no conjunto de teste:')
metrics.plot_confusion_matrix(y_test, y_pred_class)

## Resumo dos resultados
losses = metrics.extract_final_losses(history)
print()
print("{metric:<18}{value:.4f}".format(metric="Train Loss:",␣
 ↪value=losses['train_loss']))
print("{metric:<18}{value:.4f}".format(metric="Validation Loss:",␣
 ↪value=losses['val_loss']))
print('\nPerformance no conjunto de teste:')
accuracy, recall, precision, f1, auroc, aupr = metrics.
 ↪compute_performance_metrics(y_test, y_pred_class, y_pred_scores)
metrics.print_metrics_summary(accuracy, recall, precision, f1, auroc, aupr)
```

```
3041/3041 [==============================] - 2s 692us/step
Matriz de confusão no conjunto de teste:
```




```
Train Loss:       0.2187
Validation Loss:  0.2214

Performance no conjunto de teste:
```




```
Accuracy:           0.6447
Recall:             0.6747
Precision:          0.7569
F1:                 0.7134
AUROC:              0.6887
AUPR:               0.8014
```

4.1.2 Experimento2

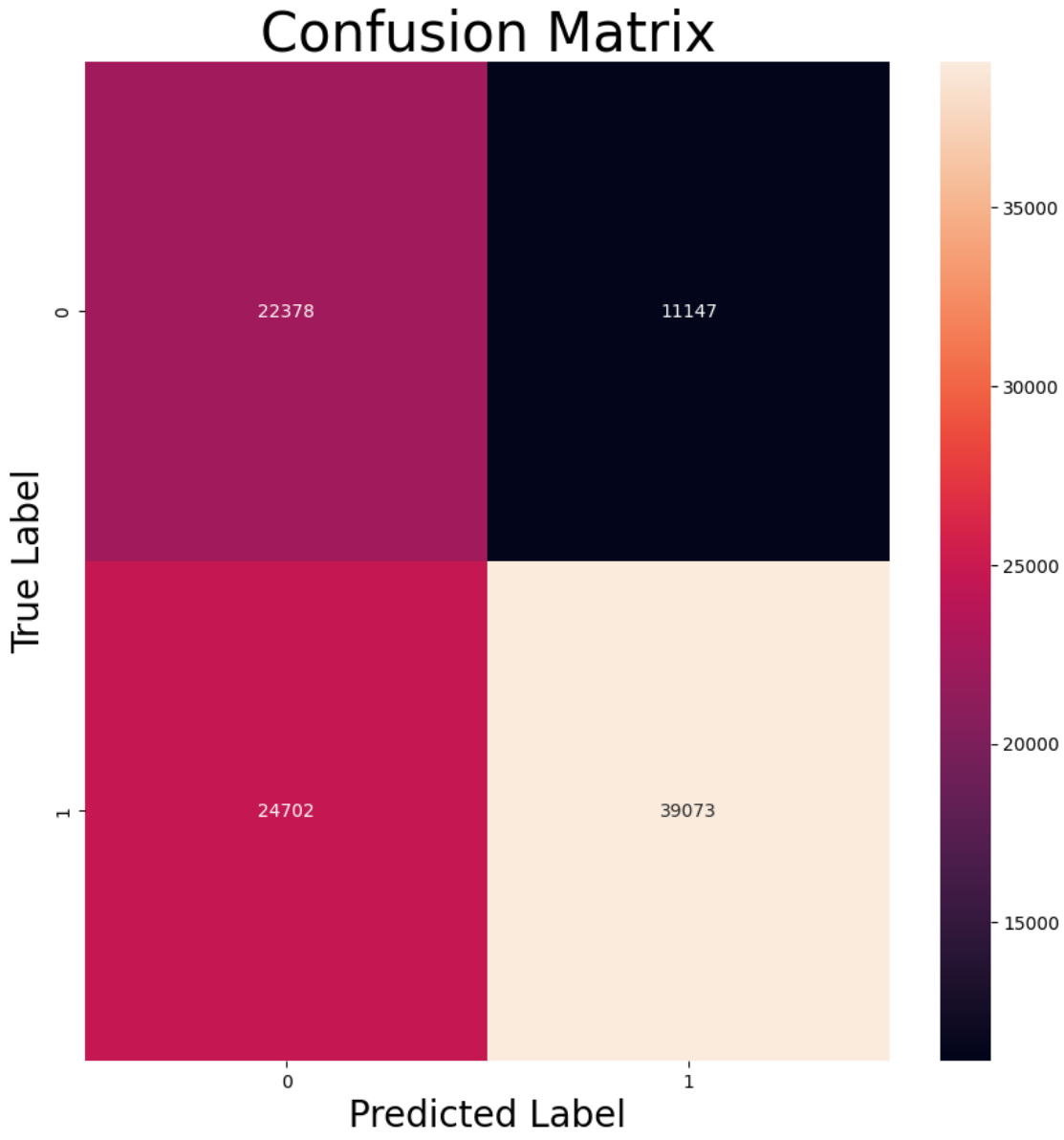
[11]:
```
# Número de features do nosso data set.
input_dim = x_train.shape[1]

# Aqui criamos o esboço da rede.
classifier = Sequential()

classifier.add(Dense(256, activation='relu', input_dim=input_dim))
classifier.add(Dense(128, activation='relu', input_dim=input_dim))
classifier.add(Dense(1, activation='sigmoid'))
classifier.compile(optimizer='adam', loss='mean_squared_error')
```

```
history_file = HISTORY_PATH + "2.npy"
model_file = MODEL_PATH + "2"

if not exists(history_file) or  not exists(model_file):
    history = classifier.fit(x_train, y_train, batch_size=32, epochs=10_000,␣
 ↪callbacks=[EarlyStopping(patience=20,verbose=1,restore_best_weights=True)],␣
 ↪validation_data=(x_val, y_val))
    np.save(history_file,history.history)
    classifier.save(model_file)
else:
    print("Model was already trained")

history=np.load(history_file, allow_pickle='TRUE').item()
classifier = keras.models.load_model(model_file)
```

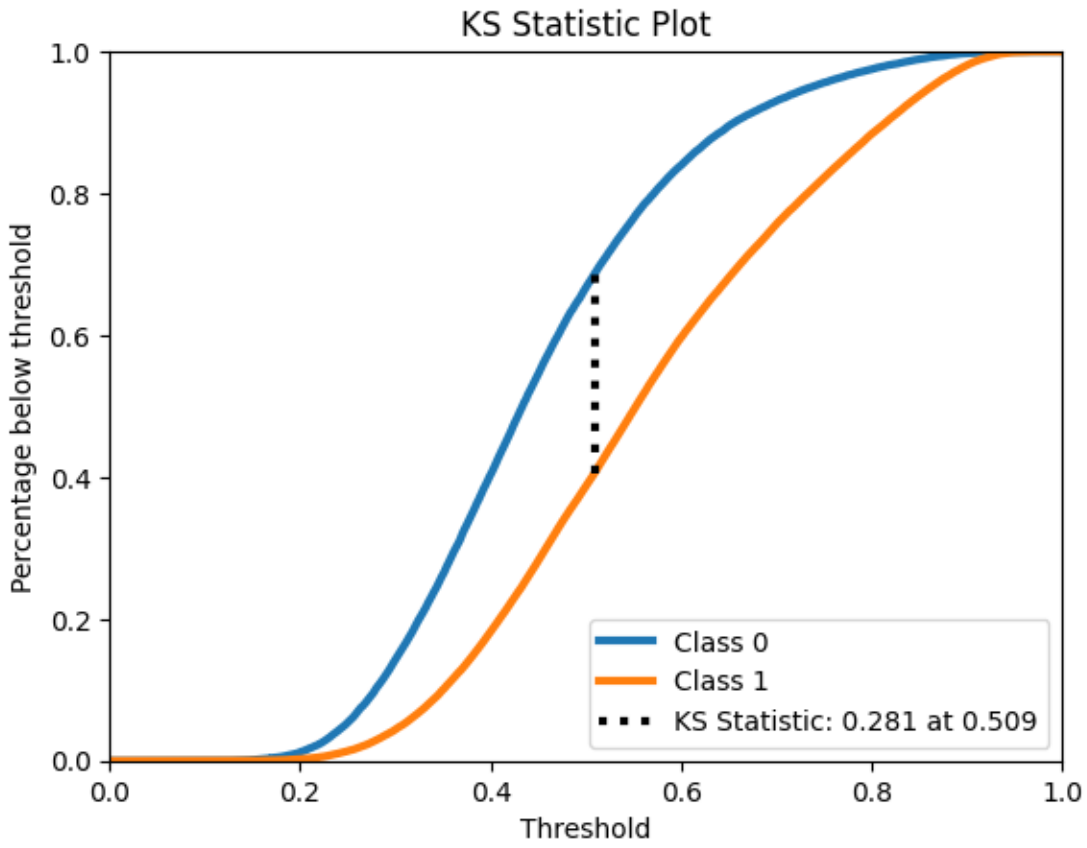
```
Model was already trained
```

[12]:
```
# Fazer predições no conjunto de teste
y_pred_scores = classifier.predict(x_test)
y_pred_class = (y_pred_scores > 0.5).astype("int32")
y_pred_scores_0 = 1 - y_pred_scores
y_pred_scores = np.concatenate([y_pred_scores_0, y_pred_scores], axis=1)

## Matriz de confusão
print('Matriz de confusão no conjunto de teste:')
metrics.plot_confusion_matrix(y_test, y_pred_class)

## Resumo dos resultados
losses = metrics.extract_final_losses(history)
print()
print("{metric:<18}{value:.4f}".format(metric="Train Loss:",␣
 ↪value=losses['train_loss']))
print("{metric:<18}{value:.4f}".format(metric="Validation Loss:",␣
 ↪value=losses['val_loss']))
print('\nPerformance no conjunto de teste:')
accuracy, recall, precision, f1, auroc, aupr = metrics.
 ↪compute_performance_metrics(y_test, y_pred_class, y_pred_scores)
metrics.print_metrics_summary(accuracy, recall, precision, f1, auroc, aupr)
```

```
 123/3041 [>…] - ETA: 2s

2022-10-18 22:52:16.651856: W
tensorflow/core/framework/cpu_allocator_impl.cc:82] Allocation of 94575600
exceeds 10% of free system memory.

3041/3041 [==============================] - 3s 879us/step
Matriz de confusão no conjunto de teste:
```




```
Train Loss:       0.2206
Validation Loss:  0.2213

Performance no conjunto de teste:
```




```
Accuracy:           0.6316
Recall:             0.6127
Precision:          0.7780
F1:                 0.6855
AUROC:              0.6921
AUPR:               0.8042
```

4.1.3 Experimento3

[13]:
```
# Número de features do nosso data set.
input_dim = x_train.shape[1]

# Aqui criamos o esboço da rede.
classifier = Sequential()

classifier.add(Dense(512, activation='sigmoid', input_dim=input_dim))
classifier.add(Dense(256, activation='sigmoid', input_dim=input_dim))
classifier.add(Dense(1, activation='sigmoid'))
```

```
optimizer=keras.optimizers.SGD( learning_rate=0.01)
classifier.compile(optimizer=optimizer, loss='mean_squared_error')

history_file = HISTORY_PATH + "3.npy"
model_file = MODEL_PATH + "3"

if not exists(history_file) or  not exists(model_file):
    history = classifier.fit(x_train, y_train, batch_size=32, epochs=1_000,␣
 ↪callbacks=[EarlyStopping(patience=20,verbose=1,restore_best_weights=True)],␣
 ↪validation_data=(x_val, y_val))
    np.save(history_file,history.history)
    classifier.save(model_file)
else:
    print("Model was already trained")

history=np.load(history_file, allow_pickle='TRUE').item()
classifier = keras.models.load_model(model_file)
```

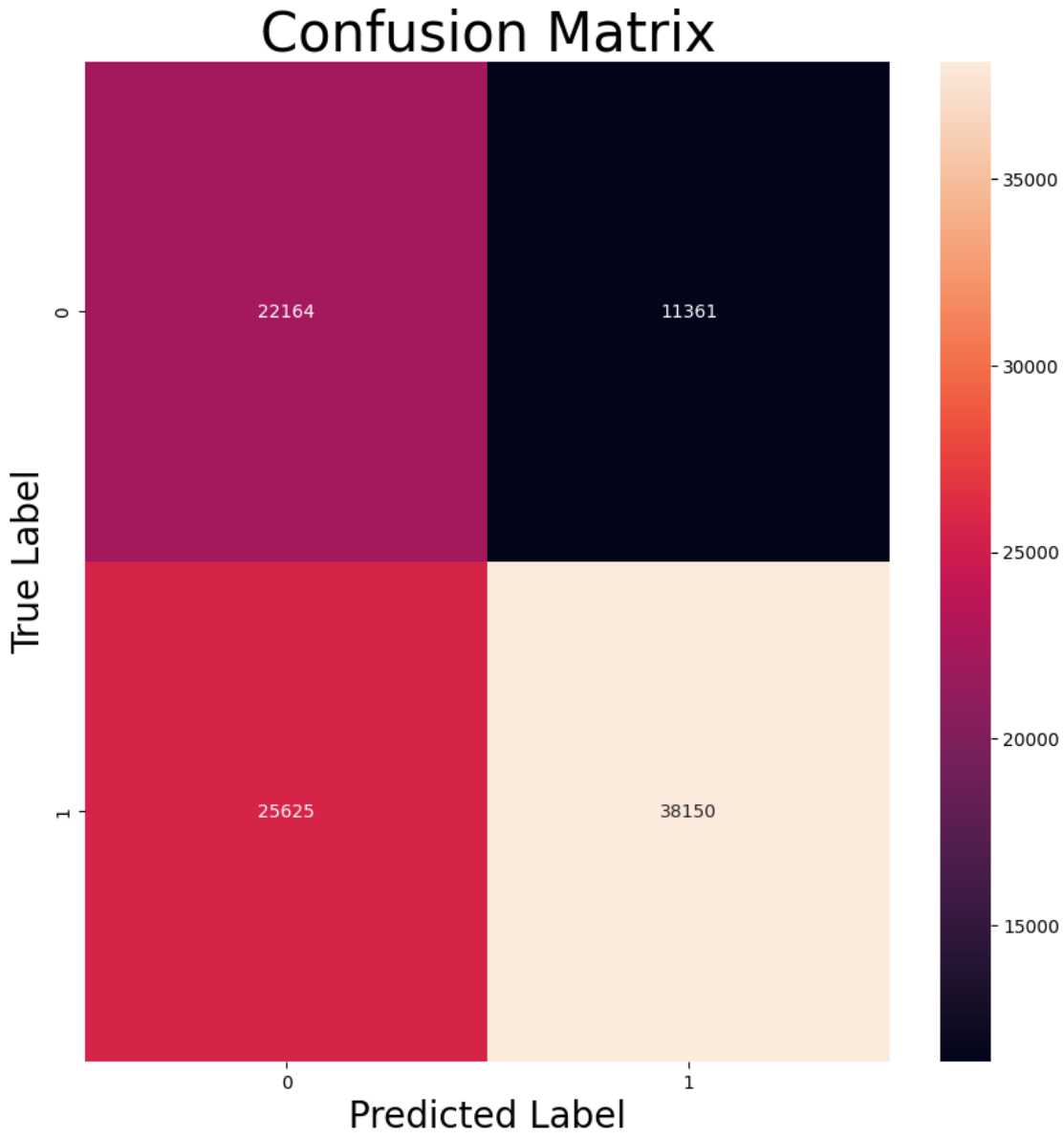
```
Model was already trained
```

[14]:
```
# Fazer predições no conjunto de teste
y_pred_scores = classifier.predict(x_test)
y_pred_class = (y_pred_scores > 0.5).astype("int32")
y_pred_scores_0 = 1 - y_pred_scores
y_pred_scores = np.concatenate([y_pred_scores_0, y_pred_scores], axis=1)

## Matriz de confusão
print('Matriz de confusão no conjunto de teste:')
metrics.plot_confusion_matrix(y_test, y_pred_class)

## Resumo dos resultados
losses = metrics.extract_final_losses(history)
print()
print("{metric:<18}{value:.4f}".format(metric="Train Loss:",␣
 ↪value=losses['train_loss']))
print("{metric:<18}{value:.4f}".format(metric="Validation Loss:",␣
 ↪value=losses['val_loss']))
print('\nPerformance no conjunto de teste:')
accuracy, recall, precision, f1, auroc, aupr = metrics.
 ↪compute_performance_metrics(y_test, y_pred_class, y_pred_scores)
metrics.print_metrics_summary(accuracy, recall, precision, f1, auroc, aupr)
```

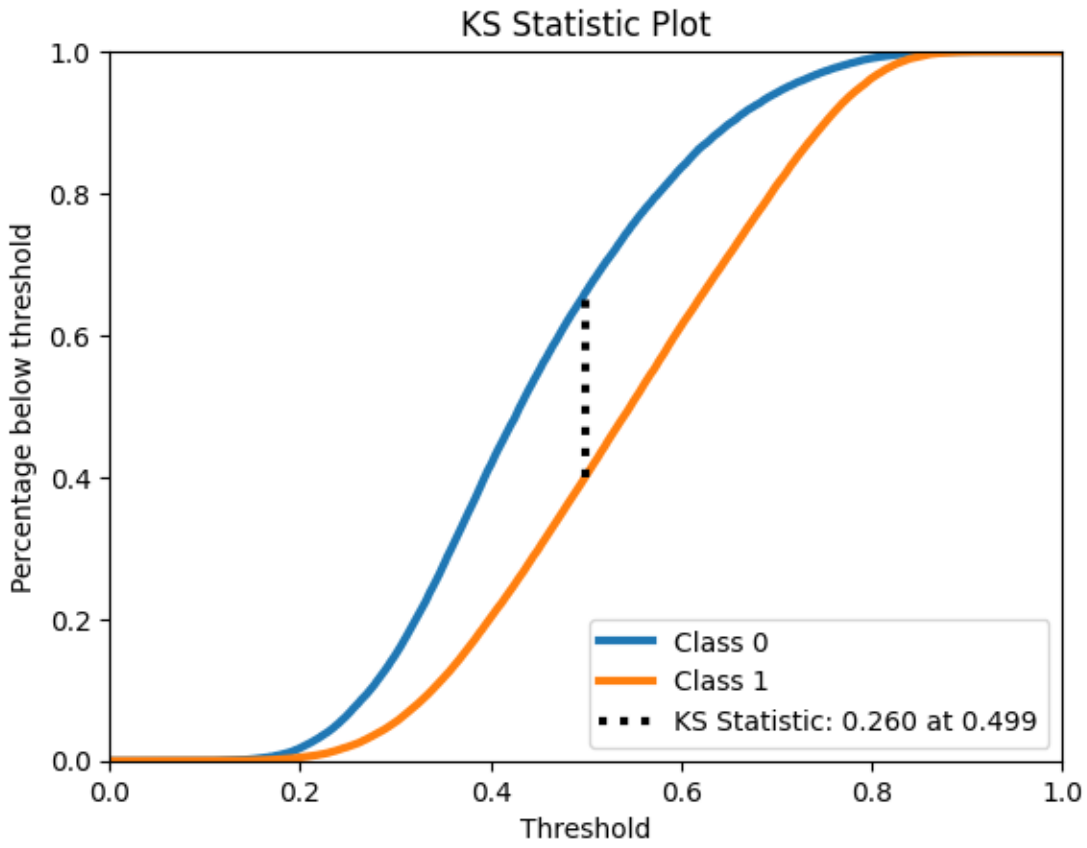
```
 107/3041 [>…] - ETA: 2s
```

```
2022-10-18 22:52:21.271167: W
tensorflow/core/framework/cpu_allocator_impl.cc:82] Allocation of 94575600
exceeds 10% of free system memory.
```

```
3041/3041 [==============================] - 3s 979us/step
Matriz de confusão no conjunto de teste:
```




```
Train Loss:       0.2258
Validation Loss:  0.2255

Performance no conjunto de teste:
```




```
Accuracy:           0.6199
Recall:             0.5982
Precision:          0.7705
F1:                 0.6735
AUROC:              0.6774
AUPR:               0.7885
```

4.1.4 Experimento4

[15]:
```
# Número de features do nosso data set.
input_dim = x_train.shape[1]

# Aqui criamos o esboço da rede.
classifier = Sequential()

classifier.add(Dense(512, activation='tanh', input_dim=input_dim))
classifier.add(Dense(256, activation='tanh', input_dim=input_dim))
classifier.add(Dense(1, activation='sigmoid'))
```

```
optimizer=keras.optimizers.SGD( learning_rate=0.01)
classifier.compile(optimizer=optimizer, loss='mean_squared_error')

history_file = HISTORY_PATH + "4.npy"
model_file = MODEL_PATH + "4"

if not exists(history_file) or  not exists(model_file):
    history = classifier.fit(x_train, y_train, batch_size=32, epochs=1_000,␣
 ↪callbacks=[EarlyStopping(patience=20,verbose=1,restore_best_weights=True)],␣
 ↪validation_data=(x_val, y_val))
    np.save(history_file,history.history)
    classifier.save(model_file)
else:
    print("Model was already trained")

history=np.load(history_file, allow_pickle='TRUE').item()
classifier = keras.models.load_model(model_file)
```

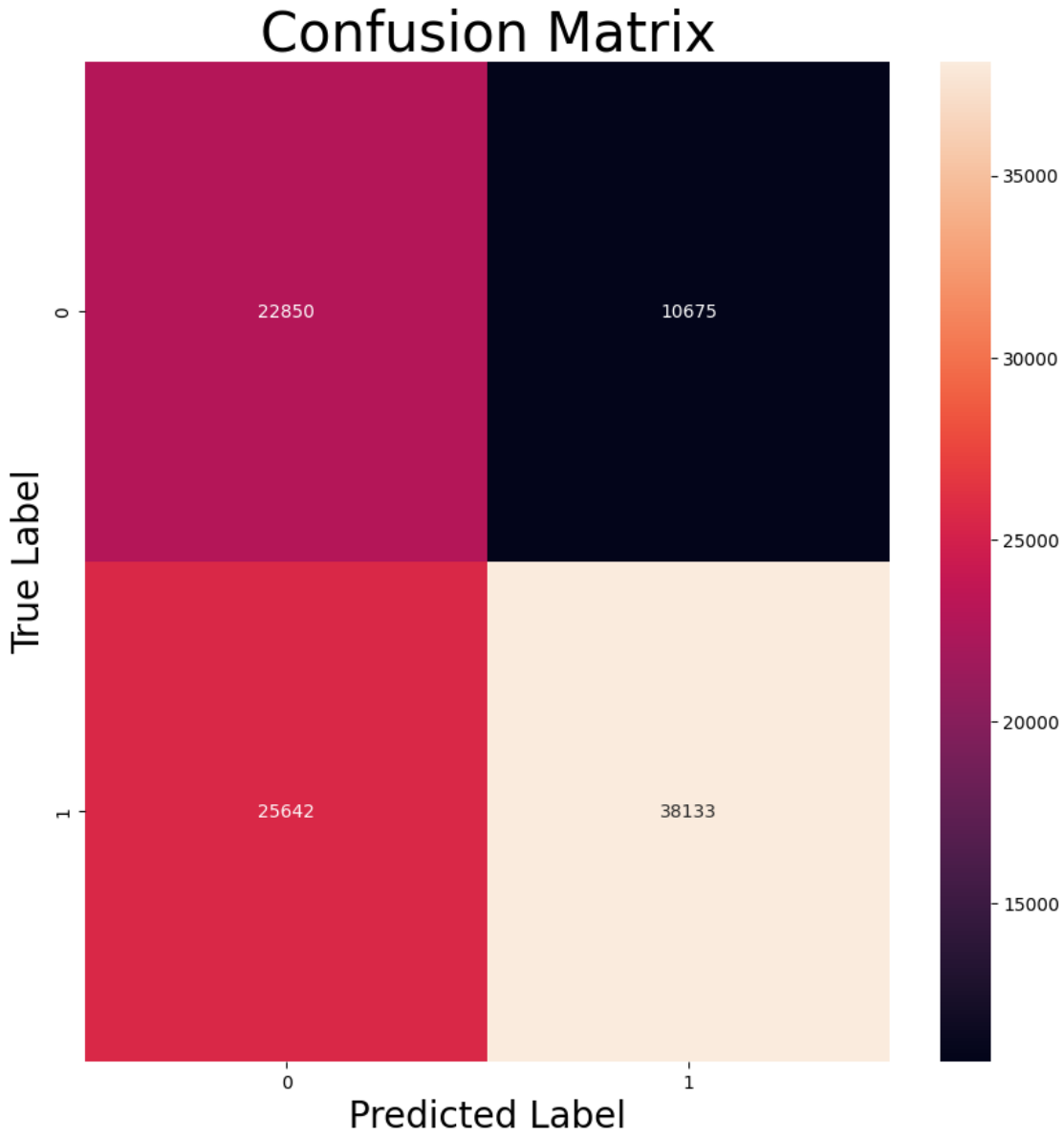
```
Model was already trained
```

[16]:
```
# Fazer predições no conjunto de teste
y_pred_scores = classifier.predict(x_test)
y_pred_class = (y_pred_scores > 0.5).astype("int32")
y_pred_scores_0 = 1 - y_pred_scores
y_pred_scores = np.concatenate([y_pred_scores_0, y_pred_scores], axis=1)

## Matriz de confusão
print('Matriz de confusão no conjunto de teste:')
metrics.plot_confusion_matrix(y_test, y_pred_class)

## Resumo dos resultados
losses = metrics.extract_final_losses(history)
print()
print("{metric:<18}{value:.4f}".format(metric="Train Loss:",␣
 ↪value=losses['train_loss']))
print("{metric:<18}{value:.4f}".format(metric="Validation Loss:",␣
 ↪value=losses['val_loss']))
print('\nPerformance no conjunto de teste:')
accuracy, recall, precision, f1, auroc, aupr = metrics.
 ↪compute_performance_metrics(y_test, y_pred_class, y_pred_scores)
metrics.print_metrics_summary(accuracy, recall, precision, f1, auroc, aupr)
```

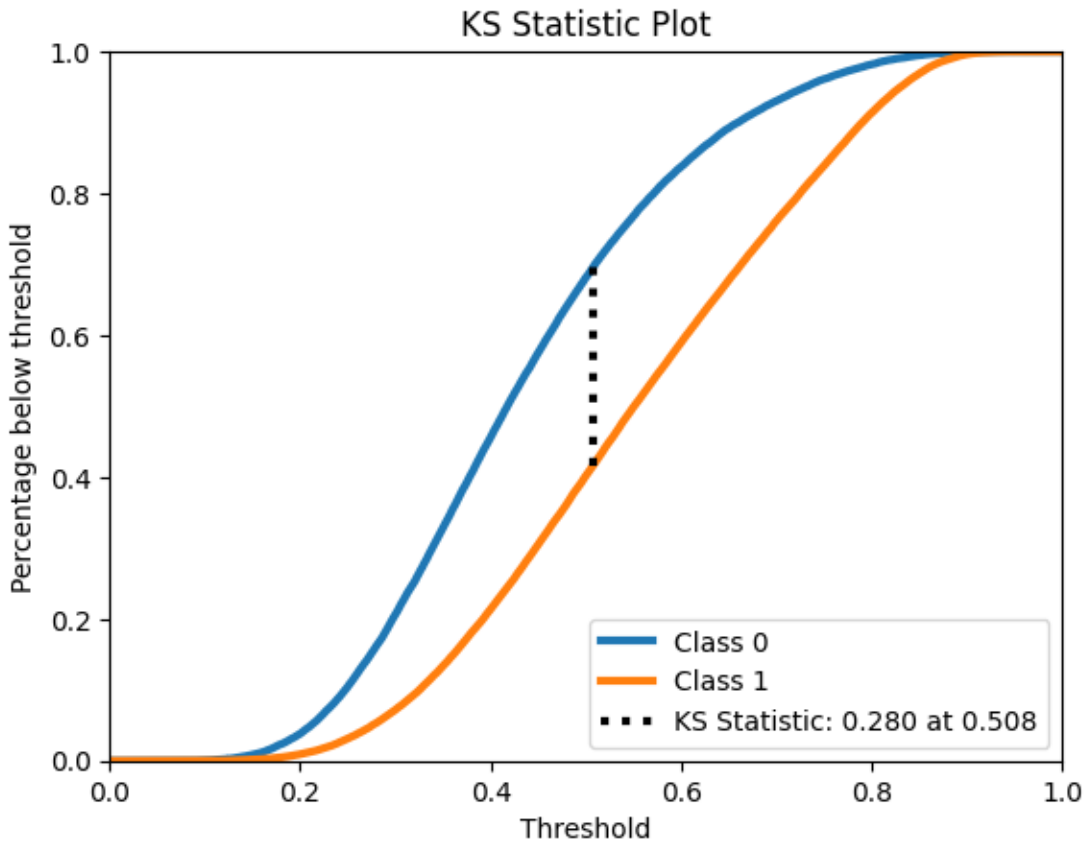
```
 104/3041 [>…] - ETA: 2s
```

```
2022-10-18 22:52:25.851786: W
tensorflow/core/framework/cpu_allocator_impl.cc:82] Allocation of 94575600
exceeds 10% of free system memory.
```

```
3041/3041 [==============================] - 3s 977us/step
Matriz de confusão no conjunto de teste:
```




```
Train Loss:       0.2174
Validation Loss:  0.2212

Performance no conjunto de teste:
```




```
Accuracy:          0.6268
Recall:            0.5979
Precision:         0.7813
F1:                0.6774
AUROC:             0.6925
AUPR:              0.8028
```

4.1.5 Experimento5

[17]:
```
# Número de features do nosso data set.
input_dim = x_train.shape[1]

# Aqui criamos o esboço da rede.
classifier = Sequential()

classifier.add(Dense(1024, activation='tanh', input_dim=input_dim))
classifier.add(Dense(1, activation='sigmoid'))
classifier.compile(optimizer='adam', loss='mean_squared_error')
```

```
history_file = HISTORY_PATH + "5.npy"
model_file = MODEL_PATH + "5"

if not exists(history_file) or  not exists(model_file):
    history = classifier.fit(x_train, y_train, batch_size=32, epochs=1_000,␣
 ↪callbacks=[EarlyStopping(patience=20,verbose=1)], validation_data=(x_val,␣
 ↪y_val))
    np.save(history_file,history.history)
    classifier.save(model_file)
else:
    print("Model was already trained")

history=np.load(history_file, allow_pickle='TRUE').item()
classifier = keras.models.load_model(model_file)
```

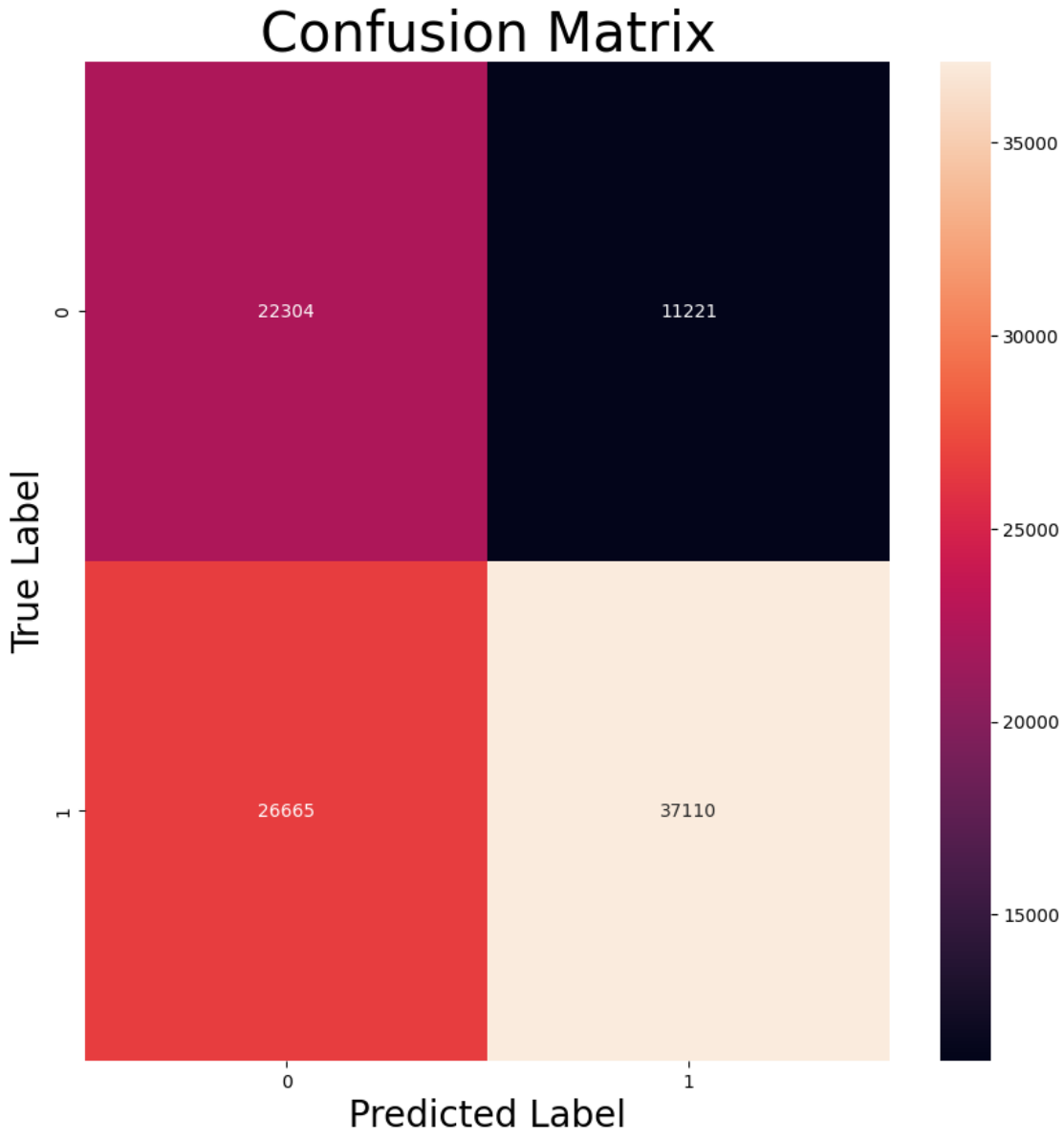
```
Model was already trained
```

[18]:
```
# Fazer predições no conjunto de teste
y_pred_scores = classifier.predict(x_test)
y_pred_class = (y_pred_scores > 0.5).astype("int32")
y_pred_scores_0 = 1 - y_pred_scores
y_pred_scores = np.concatenate([y_pred_scores_0, y_pred_scores], axis=1)

## Matriz de confusão
print('Matriz de confusão no conjunto de teste:')
metrics.plot_confusion_matrix(y_test, y_pred_class)

## Resumo dos resultados
losses = metrics.extract_final_losses(history)
print()
print("{metric:<18}{value:.4f}".format(metric="Train Loss:",␣
 ↪value=losses['train_loss']))
print("{metric:<18}{value:.4f}".format(metric="Validation Loss:",␣
 ↪value=losses['val_loss']))
print('\nPerformance no conjunto de teste:')
accuracy, recall, precision, f1, auroc, aupr = metrics.
 ↪compute_performance_metrics(y_test, y_pred_class, y_pred_scores)
metrics.print_metrics_summary(accuracy, recall, precision, f1, auroc, aupr)
```

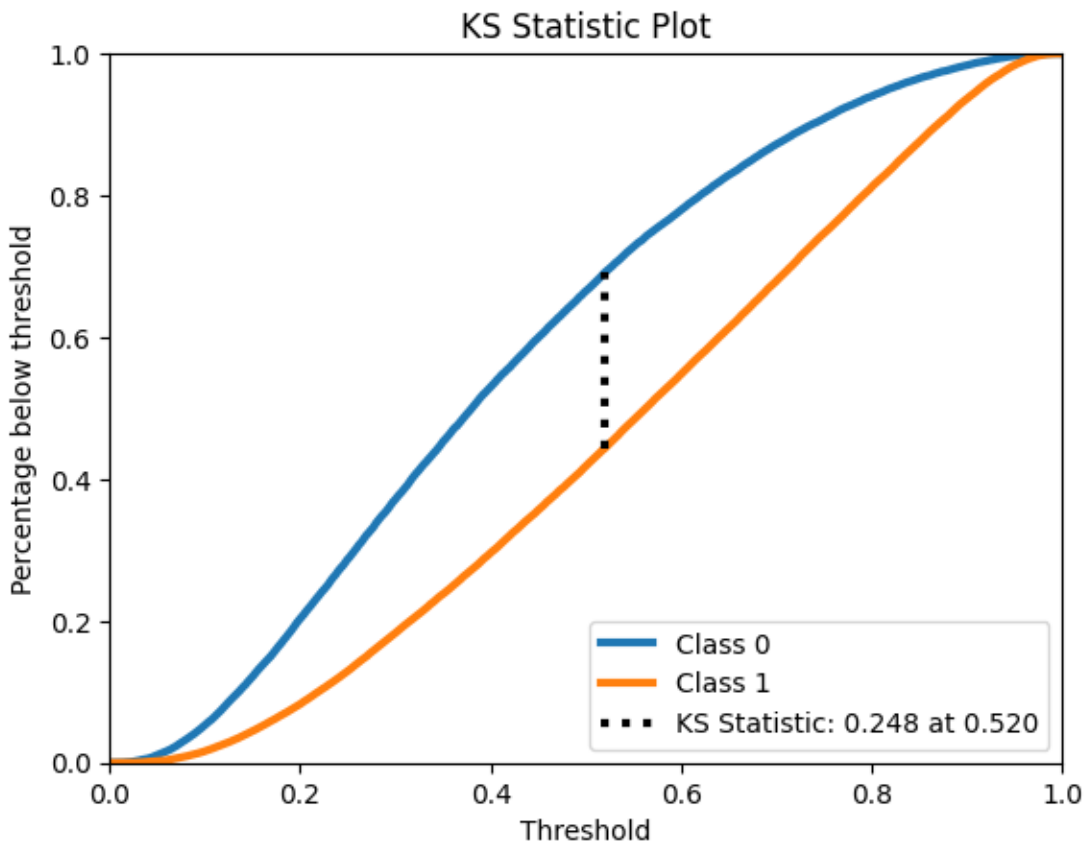
```
  94/3041 […] - ETA: 3s
```

```
2022-10-18 22:52:30.493434: W
tensorflow/core/framework/cpu_allocator_impl.cc:82] Allocation of 94575600
exceeds 10% of free system memory.
```

```
3041/3041 [==============================] - 3s 1ms/step
Matriz de confusão no conjunto de teste:
```




```
Train Loss:       0.2259
Validation Loss:  0.2234

Performance no conjunto de teste:
```




```
Accuracy:          0.6106
Recall:            0.5819
Precision:         0.7678
F1:                0.6621
AUROC:             0.6690
AUPR:              0.7847
```

4.2 MLP Ensemble

[20]:
```
MLP_ENS_PATH = "Models/MLP_ENS"
MLP_MODEL_ENS_PATH = f"{MLP_ENS_PATH}/model"
```

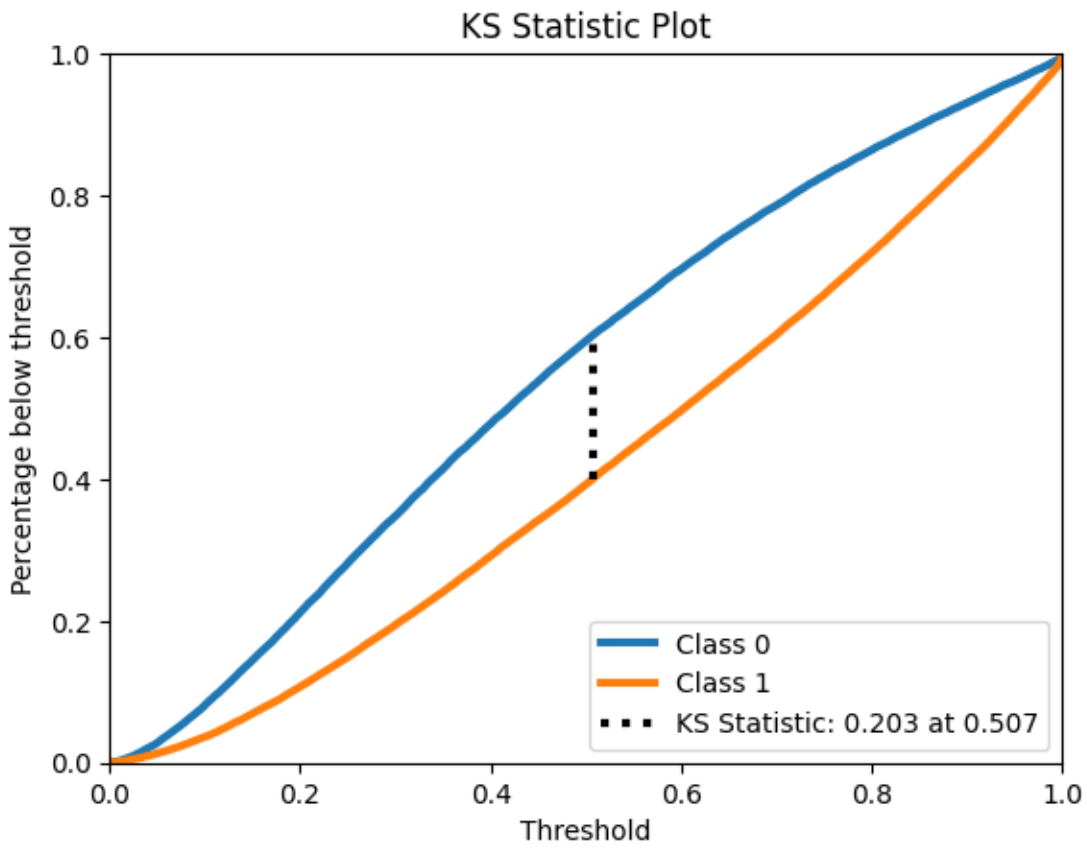
4.2.1 Experimento1

[21]:
```
mlp = MLPClassifier(verbose=True)
mlp1 = MLPClassifier(verbose=True)
mlp2 = MLPClassifier(verbose=True)
mlp3 = MLPClassifier(verbose=True)
```

```
model_file = MLP_MODEL_ENS_PATH+"1"
if not exists(model_file):
    mlp_ns = mlp.fit(x_train,y_train)
    mlp_ens = VotingClassifier([('mlp1', mlp1), ('mlp2', mlp2), ('mlp3',␣
 ↪mlp1)], voting='soft')
    mlp_ens.fit(x_train,y_train)
    with open(MLP_MODEL_ENS_PATH+"1", 'wb') as file:
        pickle.dump(mlp_ens,file)
    with open(MLP_MODEL_ENS_PATH+"ns"+"1", 'wb') as file:
        pickle.dump(mlp_ns,file)
else:
    print("Model was already trained")
    with open(MLP_MODEL_ENS_PATH+"1", 'rb') as file:
        mlp_ens = pickle.load(file)
    with open(MLP_MODEL_ENS_PATH+"ns"+"1", 'rb') as file:
        mlp_ns= pickle.load(file)
```

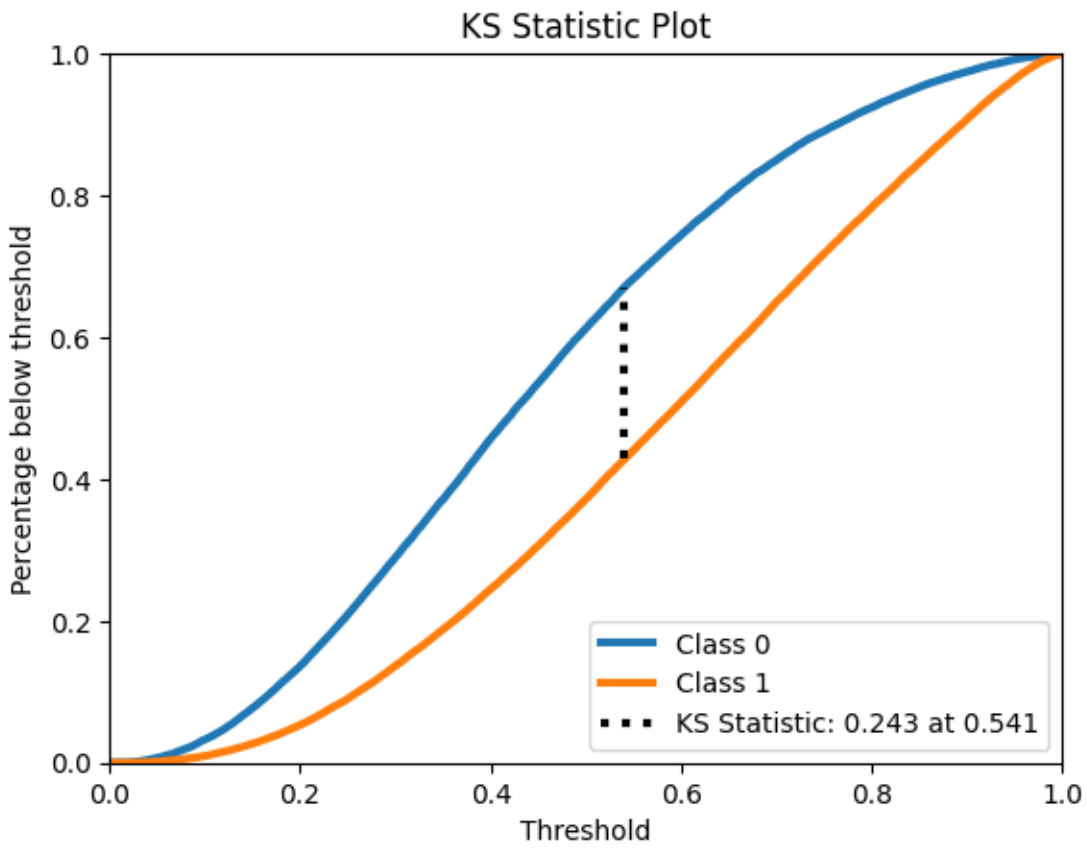
```
Model was already trained
```

[22]:
```
mlp_pred_class = mlp_ns.predict(x_test)
mlp_pred_scores = mlp_ns.predict_proba(x_test)
accuracy, recall, precision, f1, auroc, aupr = metrics.
 ↪compute_performance_metrics(y_test, mlp_pred_class, mlp_pred_scores)
print('Performance no conjunto de validação:')
metrics.print_metrics_summary(accuracy, recall, precision, f1, auroc, aupr)
```




```
Performance no conjunto de validação:

Accuracy:           0.6031
Recall:             0.6074
Precision:          0.7405
F1:                 0.6674
AUROC:              0.6362
AUPR:               0.7523
```

[23]:
```
mlp_pred_class = mlp_ens.predict(x_test)
mlp_pred_scores = mlp_ens.predict_proba(x_test)
accuracy, recall, precision, f1, auroc, aupr = metrics.
  ↪compute_performance_metrics(y_test, mlp_pred_class, mlp_pred_scores)
print('Performance no conjunto de validação:')
metrics.print_metrics_summary(accuracy, recall, precision, f1, auroc, aupr)
```




```
Performance no conjunto de validação:

Accuracy:           0.6231
Recall:             0.6285
Precision:          0.7554
F1:                 0.6861
AUROC:              0.6652
AUPR:               0.7821
```

4.2.2 Experimento2

[24]:
```
mlp1 = MLPClassifier(activation="relu",verbose=True)
mlp2 = MLPClassifier(activation="logistic", verbose=True)
mlp3 = MLPClassifier(activation="tanh", verbose=True)

model_file = MLP_MODEL_ENS_PATH+"2"
if not exists(model_file):
    mlp_ens = VotingClassifier([('mlp1', mlp1), ('mlp2', mlp2), ('mlp3',␣
 ↪mlp1)], voting='soft')
    mlp_ens.fit(x_train,y_train)
```

```
    with open(model_file, 'wb') as file:
        pickle.dump(mlp_ens,file)
else:
    print("Model was already trained")
    with open(model_file, 'rb') as file:
        mlp_ens = pickle.load(file)
```

```
Iteration 1, loss = 0.64753509
Iteration 2, loss = 0.63663109
Iteration 3, loss = 0.63208871
Iteration 4, loss = 0.62801382
Iteration 5, loss = 0.62430243
Iteration 6, loss = 0.62122509
Iteration 7, loss = 0.61808967
Iteration 8, loss = 0.61470382
Iteration 9, loss = 0.61206682
Iteration 10, loss = 0.60931509
Iteration 11, loss = 0.60675046
Iteration 12, loss = 0.60357008
Iteration 13, loss = 0.60122659
Iteration 14, loss = 0.59904267
Iteration 15, loss = 0.59629751
Iteration 16, loss = 0.59425815
Iteration 17, loss = 0.59253771
Iteration 18, loss = 0.59047747
Iteration 19, loss = 0.58861743
Iteration 20, loss = 0.58714252
Iteration 21, loss = 0.58509986
Iteration 22, loss = 0.58397579
Iteration 23, loss = 0.58205758
Iteration 24, loss = 0.58069171
Iteration 25, loss = 0.57936672
Iteration 26, loss = 0.57757033
Iteration 27, loss = 0.57645045
Iteration 28, loss = 0.57522930
Iteration 29, loss = 0.57419786
Iteration 30, loss = 0.57276861
Iteration 31, loss = 0.57193634
Iteration 32, loss = 0.57015624
Iteration 33, loss = 0.56905297
Iteration 34, loss = 0.56811091
Iteration 35, loss = 0.56690543
Iteration 36, loss = 0.56614857
Iteration 37, loss = 0.56520110
Iteration 38, loss = 0.56421993
Iteration 39, loss = 0.56327354
Iteration 40, loss = 0.56252937
```

```
Iteration 41, loss = 0.56180772
Iteration 42, loss = 0.56095066
Iteration 43, loss = 0.56049071
Iteration 44, loss = 0.55949278
Iteration 45, loss = 0.55856177
Iteration 46, loss = 0.55795933
Iteration 47, loss = 0.55734514
Iteration 48, loss = 0.55623639
Iteration 49, loss = 0.55593602
Iteration 50, loss = 0.55529719
Iteration 51, loss = 0.55434387
Iteration 52, loss = 0.55370338
Iteration 53, loss = 0.55354198
Iteration 54, loss = 0.55237061
Iteration 55, loss = 0.55178463
Iteration 56, loss = 0.55164237
Iteration 57, loss = 0.55085894
Iteration 58, loss = 0.55095789
Iteration 59, loss = 0.54985711
Iteration 60, loss = 0.54967277
Iteration 61, loss = 0.54898829
Iteration 62, loss = 0.54803315
Iteration 63, loss = 0.54822083
Iteration 64, loss = 0.54775464
Iteration 65, loss = 0.54735256
Iteration 66, loss = 0.54646948
Iteration 67, loss = 0.54602017
Iteration 68, loss = 0.54606078
Iteration 69, loss = 0.54551202
Iteration 70, loss = 0.54497839
Iteration 71, loss = 0.54451900
Iteration 72, loss = 0.54405401
Iteration 73, loss = 0.54370530
Iteration 74, loss = 0.54332152
Iteration 75, loss = 0.54318443
Iteration 76, loss = 0.54286014
Iteration 77, loss = 0.54227525
Iteration 78, loss = 0.54213666
Iteration 79, loss = 0.54136454
Iteration 80, loss = 0.54138211
Iteration 81, loss = 0.54078366
Iteration 82, loss = 0.54050826
Iteration 83, loss = 0.54037260
Iteration 84, loss = 0.54015072
Iteration 85, loss = 0.53945489
Iteration 86, loss = 0.53900001
Iteration 87, loss = 0.53876274
Iteration 88, loss = 0.53893215
```

```
Iteration 89, loss = 0.53790806
Iteration 90, loss = 0.53776196
Iteration 91, loss = 0.53814522
Iteration 92, loss = 0.53766285
Iteration 93, loss = 0.53685564
Iteration 94, loss = 0.53679662
Iteration 95, loss = 0.53628336
Iteration 96, loss = 0.53625762
Iteration 97, loss = 0.53629534
Iteration 98, loss = 0.53583006
Iteration 99, loss = 0.53548095
Iteration 100, loss = 0.53508100
Iteration 101, loss = 0.53485296
Iteration 102, loss = 0.53427050
Iteration 103, loss = 0.53445166
Iteration 104, loss = 0.53436462
Iteration 105, loss = 0.53396239
Iteration 106, loss = 0.53339743
Iteration 107, loss = 0.53376229
Iteration 108, loss = 0.53354723
Iteration 109, loss = 0.53331103
Iteration 110, loss = 0.53256008
Iteration 111, loss = 0.53242735
Iteration 112, loss = 0.53259901
Iteration 113, loss = 0.53208439
Iteration 114, loss = 0.53241070
Iteration 115, loss = 0.53145281
Iteration 116, loss = 0.53153425
Iteration 117, loss = 0.53125877
Iteration 118, loss = 0.53113045
Iteration 119, loss = 0.53098857
Iteration 120, loss = 0.53077293
Iteration 121, loss = 0.53031286
Iteration 122, loss = 0.53060349
Iteration 123, loss = 0.53018776
Iteration 124, loss = 0.52978957
Iteration 125, loss = 0.52974344
Iteration 126, loss = 0.52912534
Iteration 127, loss = 0.52924852
Iteration 128, loss = 0.52936792
Iteration 129, loss = 0.52882348
Iteration 130, loss = 0.52876517
Iteration 131, loss = 0.52863485
Iteration 132, loss = 0.52835801
Iteration 133, loss = 0.52865486
Iteration 134, loss = 0.52814913
Iteration 135, loss = 0.52810136
Iteration 136, loss = 0.52812694
```

```
Iteration 137, loss = 0.52756814
Iteration 138, loss = 0.52777091
Iteration 139, loss = 0.52770793
Iteration 140, loss = 0.52724650
Iteration 141, loss = 0.52686546
Iteration 142, loss = 0.52728378
Iteration 143, loss = 0.52694407
Iteration 144, loss = 0.52646595
Iteration 145, loss = 0.52640921
Iteration 146, loss = 0.52712103
Iteration 147, loss = 0.52633605
Iteration 148, loss = 0.52630089
Iteration 149, loss = 0.52598950
Iteration 150, loss = 0.52601645
Iteration 151, loss = 0.52591181
Iteration 152, loss = 0.52534908
Iteration 153, loss = 0.52584657
Iteration 154, loss = 0.52501647
Iteration 155, loss = 0.52497134
Iteration 156, loss = 0.52518838
Iteration 157, loss = 0.52504248
Iteration 158, loss = 0.52450625
Iteration 159, loss = 0.52467902
Iteration 160, loss = 0.52432522
Iteration 161, loss = 0.52472750
Iteration 162, loss = 0.52462001
Iteration 163, loss = 0.52371846
Iteration 164, loss = 0.52384028
Iteration 165, loss = 0.52402553
Iteration 166, loss = 0.52361967
Iteration 167, loss = 0.52363907
Iteration 168, loss = 0.52323169
Iteration 169, loss = 0.52329199
Iteration 170, loss = 0.52305873
Iteration 171, loss = 0.52279279
Iteration 172, loss = 0.52259713
Iteration 173, loss = 0.52279944
Iteration 174, loss = 0.52269834
Iteration 175, loss = 0.52261996
Iteration 176, loss = 0.52242699
Iteration 177, loss = 0.52282451
Iteration 178, loss = 0.52246146
Iteration 179, loss = 0.52195818
Iteration 180, loss = 0.52201232
Iteration 181, loss = 0.52183884
Iteration 182, loss = 0.52199692
Iteration 183, loss = 0.52163606
Iteration 184, loss = 0.52171333
```

```
Iteration 185, loss = 0.52130629
Iteration 186, loss = 0.52128210
Iteration 187, loss = 0.52122208
Iteration 188, loss = 0.52099076
Iteration 189, loss = 0.52081576
Iteration 190, loss = 0.52105157
Iteration 191, loss = 0.52108614
Iteration 192, loss = 0.52034564
Iteration 193, loss = 0.52030714
Iteration 194, loss = 0.52105157
Iteration 195, loss = 0.52029085
Iteration 196, loss = 0.52034301
Iteration 197, loss = 0.51990438
Iteration 198, loss = 0.52026679
Iteration 199, loss = 0.52050847
Iteration 200, loss = 0.51999569
```

```
/home/yesternight/Desktop/neural-networks/neural-networks/lib/python3.10/site-
packages/sklearn/neural_network/_multilayer_perceptron.py:702:
ConvergenceWarning: Stochastic Optimizer: Maximum iterations (200) reached and
the optimization hasn't converged yet.
  warnings.warn(
```

```
Iteration 1, loss = 0.65155209
Iteration 2, loss = 0.64429180
Iteration 3, loss = 0.64234995
Iteration 4, loss = 0.63921400
Iteration 5, loss = 0.63653453
Iteration 6, loss = 0.63478096
Iteration 7, loss = 0.63354119
Iteration 8, loss = 0.63239088
Iteration 9, loss = 0.63115809
Iteration 10, loss = 0.63022432
Iteration 11, loss = 0.62931609
Iteration 12, loss = 0.62827393
Iteration 13, loss = 0.62711916
Iteration 14, loss = 0.62641421
Iteration 15, loss = 0.62518403
Iteration 16, loss = 0.62423406
Iteration 17, loss = 0.62328709
Iteration 18, loss = 0.62250656
Iteration 19, loss = 0.62116275
Iteration 20, loss = 0.62013667
Iteration 21, loss = 0.61917812
Iteration 22, loss = 0.61808399
Iteration 23, loss = 0.61679944
Iteration 24, loss = 0.61579868
Iteration 25, loss = 0.61443743
Iteration 26, loss = 0.61326962
```

```
Iteration 27, loss = 0.61194908
Iteration 28, loss = 0.61077482
Iteration 29, loss = 0.60940109
Iteration 30, loss = 0.60836691
Iteration 31, loss = 0.60692532
Iteration 32, loss = 0.60544587
Iteration 33, loss = 0.60443397
Iteration 34, loss = 0.60289900
Iteration 35, loss = 0.60196312
Iteration 36, loss = 0.60060799
Iteration 37, loss = 0.59908238
Iteration 38, loss = 0.59780072
Iteration 39, loss = 0.59673256
Iteration 40, loss = 0.59547748
Iteration 41, loss = 0.59428788
Iteration 42, loss = 0.59317121
Iteration 43, loss = 0.59189399
Iteration 44, loss = 0.59089748
Iteration 45, loss = 0.58993992
Iteration 46, loss = 0.58847574
Iteration 47, loss = 0.58771106
Iteration 48, loss = 0.58685018
Iteration 49, loss = 0.58549171
Iteration 50, loss = 0.58450643
Iteration 51, loss = 0.58351642
Iteration 52, loss = 0.58266202
Iteration 53, loss = 0.58158421
Iteration 54, loss = 0.58069426
Iteration 55, loss = 0.58000563
Iteration 56, loss = 0.57894089
Iteration 57, loss = 0.57797334
Iteration 58, loss = 0.57726795
Iteration 59, loss = 0.57644477
Iteration 60, loss = 0.57560218
Iteration 61, loss = 0.57470045
Iteration 62, loss = 0.57377156
Iteration 63, loss = 0.57322805
Iteration 64, loss = 0.57255915
Iteration 65, loss = 0.57152135
Iteration 66, loss = 0.57108013
Iteration 67, loss = 0.57011587
Iteration 68, loss = 0.56925000
Iteration 69, loss = 0.56895461
Iteration 70, loss = 0.56801005
Iteration 71, loss = 0.56745066
Iteration 72, loss = 0.56677052
Iteration 73, loss = 0.56620533
Iteration 74, loss = 0.56559116
```

```
Iteration 75, loss = 0.56483819
Iteration 76, loss = 0.56407590
Iteration 77, loss = 0.56362644
Iteration 78, loss = 0.56278137
Iteration 79, loss = 0.56267571
Iteration 80, loss = 0.56168106
Iteration 81, loss = 0.56114488
Iteration 82, loss = 0.56084184
Iteration 83, loss = 0.56028527
Iteration 84, loss = 0.55947545
Iteration 85, loss = 0.55920348
Iteration 86, loss = 0.55846518
Iteration 87, loss = 0.55827437
Iteration 88, loss = 0.55772508
Iteration 89, loss = 0.55720959
Iteration 90, loss = 0.55662045
Iteration 91, loss = 0.55638824
Iteration 92, loss = 0.55592380
Iteration 93, loss = 0.55538412
Iteration 94, loss = 0.55480778
Iteration 95, loss = 0.55423494
Iteration 96, loss = 0.55406217
Iteration 97, loss = 0.55360550
Iteration 98, loss = 0.55299536
Iteration 99, loss = 0.55242301
Iteration 100, loss = 0.55253120
Iteration 101, loss = 0.55163281
Iteration 102, loss = 0.55140790
Iteration 103, loss = 0.55083727
Iteration 104, loss = 0.55043238
Iteration 105, loss = 0.55025387
Iteration 106, loss = 0.54974095
Iteration 107, loss = 0.54937667
Iteration 108, loss = 0.54921484
Iteration 109, loss = 0.54866116
Iteration 110, loss = 0.54830068
Iteration 111, loss = 0.54789529
Iteration 112, loss = 0.54764133
Iteration 113, loss = 0.54724077
Iteration 114, loss = 0.54688606
Iteration 115, loss = 0.54624982
Iteration 116, loss = 0.54612953
Iteration 117, loss = 0.54575085
Iteration 118, loss = 0.54527824
Iteration 119, loss = 0.54511544
Iteration 120, loss = 0.54478006
Iteration 121, loss = 0.54435478
Iteration 122, loss = 0.54416125
```

```
Iteration 123, loss = 0.54384473
Iteration 124, loss = 0.54330924
Iteration 125, loss = 0.54326968
Iteration 126, loss = 0.54313279
Iteration 127, loss = 0.54277885
Iteration 128, loss = 0.54240570
Iteration 129, loss = 0.54213166
Iteration 130, loss = 0.54169345
Iteration 131, loss = 0.54152044
Iteration 132, loss = 0.54119333
Iteration 133, loss = 0.54098642
Iteration 134, loss = 0.54072215
Iteration 135, loss = 0.54038972
Iteration 136, loss = 0.53993084
Iteration 137, loss = 0.53963417
Iteration 138, loss = 0.53961459
Iteration 139, loss = 0.53943236
Iteration 140, loss = 0.53916044
Iteration 141, loss = 0.53879139
Iteration 142, loss = 0.53839708
Iteration 143, loss = 0.53834944
Iteration 144, loss = 0.53811935
Iteration 145, loss = 0.53812278
Iteration 146, loss = 0.53774740
Iteration 147, loss = 0.53751400
Iteration 148, loss = 0.53722353
Iteration 149, loss = 0.53640393
Iteration 150, loss = 0.53680485
Iteration 151, loss = 0.53652507
Iteration 152, loss = 0.53617368
Iteration 153, loss = 0.53581659
Iteration 154, loss = 0.53587741
Iteration 155, loss = 0.53568055
Iteration 156, loss = 0.53540059
Iteration 157, loss = 0.53523132
Iteration 158, loss = 0.53466854
Iteration 159, loss = 0.53459456
Iteration 160, loss = 0.53470478
Iteration 161, loss = 0.53429263
Iteration 162, loss = 0.53387903
Iteration 163, loss = 0.53398596
Iteration 164, loss = 0.53358132
Iteration 165, loss = 0.53351211
Iteration 166, loss = 0.53314987
Iteration 167, loss = 0.53299064
Iteration 168, loss = 0.53284411
Iteration 169, loss = 0.53267401
Iteration 170, loss = 0.53218006
```

```
Iteration 171, loss = 0.53209646
Iteration 172, loss = 0.53211672
Iteration 173, loss = 0.53163272
Iteration 174, loss = 0.53152847
Iteration 175, loss = 0.53145416
Iteration 176, loss = 0.53106944
Iteration 177, loss = 0.53088365
Iteration 178, loss = 0.53096305
Iteration 179, loss = 0.53064605
Iteration 180, loss = 0.53049390
Iteration 181, loss = 0.53033776
Iteration 182, loss = 0.52997971
Iteration 183, loss = 0.53003647
Iteration 184, loss = 0.52960358
Iteration 185, loss = 0.52988689
Iteration 186, loss = 0.52964839
Iteration 187, loss = 0.52924540
Iteration 188, loss = 0.52929580
Iteration 189, loss = 0.52891340
Iteration 190, loss = 0.52855333
Iteration 191, loss = 0.52870718
Iteration 192, loss = 0.52875709
Iteration 193, loss = 0.52833508
Iteration 194, loss = 0.52793886
Iteration 195, loss = 0.52817781
Iteration 196, loss = 0.52764578
Iteration 197, loss = 0.52790520
Iteration 198, loss = 0.52749653
Iteration 199, loss = 0.52756997
Iteration 200, loss = 0.52698687
```

```
/home/yesternight/Desktop/neural-networks/neural-networks/lib/python3.10/site-
packages/sklearn/neural_network/_multilayer_perceptron.py:702:
ConvergenceWarning: Stochastic Optimizer: Maximum iterations (200) reached and
the optimization hasn't converged yet.
  warnings.warn(
```

```
Iteration 1, loss = 0.64702370
Iteration 2, loss = 0.63611830
Iteration 3, loss = 0.63179126
Iteration 4, loss = 0.62855593
Iteration 5, loss = 0.62499007
Iteration 6, loss = 0.62224144
Iteration 7, loss = 0.61913715
Iteration 8, loss = 0.61619285
Iteration 9, loss = 0.61319078
Iteration 10, loss = 0.61030318
Iteration 11, loss = 0.60732739
Iteration 12, loss = 0.60516074
```

```
Iteration 13, loss = 0.60230904
Iteration 14, loss = 0.60013840
Iteration 15, loss = 0.59761100
Iteration 16, loss = 0.59558348
Iteration 17, loss = 0.59340199
Iteration 18, loss = 0.59132690
Iteration 19, loss = 0.58932636
Iteration 20, loss = 0.58772309
Iteration 21, loss = 0.58606257
Iteration 22, loss = 0.58418382
Iteration 23, loss = 0.58237601
Iteration 24, loss = 0.58097747
Iteration 25, loss = 0.57977244
Iteration 26, loss = 0.57855647
Iteration 27, loss = 0.57694403
Iteration 28, loss = 0.57562487
Iteration 29, loss = 0.57402644
Iteration 30, loss = 0.57266444
Iteration 31, loss = 0.57156930
Iteration 32, loss = 0.57061796
Iteration 33, loss = 0.56953364
Iteration 34, loss = 0.56865884
Iteration 35, loss = 0.56789229
Iteration 36, loss = 0.56667854
Iteration 37, loss = 0.56526158
Iteration 38, loss = 0.56478831
Iteration 39, loss = 0.56346587
Iteration 40, loss = 0.56282305
Iteration 41, loss = 0.56257390
Iteration 42, loss = 0.56179085
Iteration 43, loss = 0.56077544
Iteration 44, loss = 0.56011531
Iteration 45, loss = 0.55899617
Iteration 46, loss = 0.55851194
Iteration 47, loss = 0.55770540
Iteration 48, loss = 0.55748426
Iteration 49, loss = 0.55627364
Iteration 50, loss = 0.55620737
Iteration 51, loss = 0.55508730
Iteration 52, loss = 0.55459251
Iteration 53, loss = 0.55422886
Iteration 54, loss = 0.55342087
Iteration 55, loss = 0.55269082
Iteration 56, loss = 0.55232995
Iteration 57, loss = 0.55217388
Iteration 58, loss = 0.55119804
Iteration 59, loss = 0.55080381
Iteration 60, loss = 0.55016746
```

```
Iteration 61, loss = 0.54984788
Iteration 62, loss = 0.54962082
Iteration 63, loss = 0.54899790
Iteration 64, loss = 0.54837469
Iteration 65, loss = 0.54785120
Iteration 66, loss = 0.54751440
Iteration 67, loss = 0.54691098
Iteration 68, loss = 0.54689386
Iteration 69, loss = 0.54603434
Iteration 70, loss = 0.54596707
Iteration 71, loss = 0.54535125
Iteration 72, loss = 0.54525036
Iteration 73, loss = 0.54495242
Iteration 74, loss = 0.54412381
Iteration 75, loss = 0.54401959
Iteration 76, loss = 0.54373442
Iteration 77, loss = 0.54300687
Iteration 78, loss = 0.54264697
Iteration 79, loss = 0.54247096
Iteration 80, loss = 0.54223415
Iteration 81, loss = 0.54175506
Iteration 82, loss = 0.54122698
Iteration 83, loss = 0.54093973
Iteration 84, loss = 0.54107935
Iteration 85, loss = 0.54049400
Iteration 86, loss = 0.54023754
Iteration 87, loss = 0.53953271
Iteration 88, loss = 0.53978784
Iteration 89, loss = 0.53909874
Iteration 90, loss = 0.53914372
Iteration 91, loss = 0.53887453
Iteration 92, loss = 0.53828846
Iteration 93, loss = 0.53788415
Iteration 94, loss = 0.53772754
Iteration 95, loss = 0.53779962
Iteration 96, loss = 0.53712480
Iteration 97, loss = 0.53701212
Iteration 98, loss = 0.53638246
Iteration 99, loss = 0.53639002
Iteration 100, loss = 0.53657872
Iteration 101, loss = 0.53607322
Iteration 102, loss = 0.53560093
Iteration 103, loss = 0.53528612
Iteration 104, loss = 0.53499038
Iteration 105, loss = 0.53493281
Iteration 106, loss = 0.53450089
Iteration 107, loss = 0.53457155
Iteration 108, loss = 0.53416242
```

```
Iteration 109, loss = 0.53423415
Iteration 110, loss = 0.53386151
Iteration 111, loss = 0.53320491
Iteration 112, loss = 0.53343837
Iteration 113, loss = 0.53293618
Iteration 114, loss = 0.53322702
Iteration 115, loss = 0.53267852
Iteration 116, loss = 0.53251596
Iteration 117, loss = 0.53182004
Iteration 118, loss = 0.53219764
Iteration 119, loss = 0.53198512
Iteration 120, loss = 0.53202482
Iteration 121, loss = 0.53173083
Iteration 122, loss = 0.53144421
Iteration 123, loss = 0.53144078
Iteration 124, loss = 0.53099757
Iteration 125, loss = 0.53090951
Iteration 126, loss = 0.53028339
Iteration 127, loss = 0.53065066
Iteration 128, loss = 0.53013022
Iteration 129, loss = 0.53002005
Iteration 130, loss = 0.52973475
Iteration 131, loss = 0.52980783
Iteration 132, loss = 0.52926796
Iteration 133, loss = 0.52934952
Iteration 134, loss = 0.52886407
Iteration 135, loss = 0.52913751
Iteration 136, loss = 0.52917415
Iteration 137, loss = 0.52881633
Iteration 138, loss = 0.52878286
Iteration 139, loss = 0.52817209
Iteration 140, loss = 0.52832828
Iteration 141, loss = 0.52811952
Iteration 142, loss = 0.52767794
Iteration 143, loss = 0.52782519
Iteration 144, loss = 0.52754033
Iteration 145, loss = 0.52723843
Iteration 146, loss = 0.52753340
Iteration 147, loss = 0.52705190
Iteration 148, loss = 0.52691383
Iteration 149, loss = 0.52714253
Iteration 150, loss = 0.52672059
Iteration 151, loss = 0.52670332
Iteration 152, loss = 0.52610854
Iteration 153, loss = 0.52632477
Iteration 154, loss = 0.52614529
Iteration 155, loss = 0.52586819
Iteration 156, loss = 0.52539256
```

```
Iteration 157, loss = 0.52547589
Iteration 158, loss = 0.52527168
Iteration 159, loss = 0.52544444
Iteration 160, loss = 0.52499152
Iteration 161, loss = 0.52515897
Iteration 162, loss = 0.52505232
Iteration 163, loss = 0.52491505
Iteration 164, loss = 0.52466506
Iteration 165, loss = 0.52445306
Iteration 166, loss = 0.52419378
Iteration 167, loss = 0.52436323
Iteration 168, loss = 0.52427023
Iteration 169, loss = 0.52406445
Iteration 170, loss = 0.52372727
Iteration 171, loss = 0.52395954
Iteration 172, loss = 0.52376345
Iteration 173, loss = 0.52321558
Iteration 174, loss = 0.52340890
Iteration 175, loss = 0.52288795
Iteration 176, loss = 0.52348597
Iteration 177, loss = 0.52331425
Iteration 178, loss = 0.52290233
Iteration 179, loss = 0.52266064
Iteration 180, loss = 0.52277530
Iteration 181, loss = 0.52285520
Iteration 182, loss = 0.52256273
Iteration 183, loss = 0.52230541
Iteration 184, loss = 0.52252289
Iteration 185, loss = 0.52203995
Iteration 186, loss = 0.52227225
Iteration 187, loss = 0.52207781
Iteration 188, loss = 0.52205981
Iteration 189, loss = 0.52213545
Iteration 190, loss = 0.52185211
Iteration 191, loss = 0.52156630
Iteration 192, loss = 0.52146529
Iteration 193, loss = 0.52123476
Iteration 194, loss = 0.52162043
Iteration 195, loss = 0.52119709
Iteration 196, loss = 0.52147756
Iteration 197, loss = 0.52090900
Iteration 198, loss = 0.52062466
Iteration 199, loss = 0.52078229
Iteration 200, loss = 0.52103379

/home/yesternight/Desktop/neural-networks/neural-networks/lib/python3.10/site-
packages/sklearn/neural_network/_multilayer_perceptron.py:702:
ConvergenceWarning: Stochastic Optimizer: Maximum iterations (200) reached and
```

```
the optimization hasn't converged yet.
  warnings.warn(
```

[25]:
```
mlp_ens.fit(x_train, y_train)
mlp_pred_class = mlp_ens.predict(x_test)
mlp_pred_scores = mlp_ens.predict_proba(x_test)
accuracy, recall, precision, f1, auroc, aupr = metrics.
 ↪compute_performance_metrics(y_test, mlp_pred_class, mlp_pred_scores)
print('Performance no conjunto de validação:')
metrics.print_metrics_summary(accuracy, recall, precision, f1, auroc, aupr)
```

```
Iteration 1, loss = 0.64770523
Iteration 2, loss = 0.63666854
Iteration 3, loss = 0.63180921
Iteration 4, loss = 0.62866338
Iteration 5, loss = 0.62533040
Iteration 6, loss = 0.62236327
Iteration 7, loss = 0.61971495
Iteration 8, loss = 0.61665227
Iteration 9, loss = 0.61397063
Iteration 10, loss = 0.61087467
Iteration 11, loss = 0.60827173
Iteration 12, loss = 0.60581651
Iteration 13, loss = 0.60353974
Iteration 14, loss = 0.60144946
Iteration 15, loss = 0.59892301
Iteration 16, loss = 0.59696898
Iteration 17, loss = 0.59451685
Iteration 18, loss = 0.59234238
Iteration 19, loss = 0.59069855
Iteration 20, loss = 0.58859560
Iteration 21, loss = 0.58699210
Iteration 22, loss = 0.58532093
Iteration 23, loss = 0.58373534
Iteration 24, loss = 0.58170461
Iteration 25, loss = 0.58076439
Iteration 26, loss = 0.57942830
Iteration 27, loss = 0.57819718
Iteration 28, loss = 0.57656389
Iteration 29, loss = 0.57534182
Iteration 30, loss = 0.57390135
Iteration 31, loss = 0.57301611
Iteration 32, loss = 0.57211315
Iteration 33, loss = 0.57046742
Iteration 34, loss = 0.56968253
Iteration 35, loss = 0.56864120
Iteration 36, loss = 0.56769580
Iteration 37, loss = 0.56670183
```

```
Iteration 38, loss = 0.56582668
Iteration 39, loss = 0.56507564
Iteration 40, loss = 0.56406572
Iteration 41, loss = 0.56332297
Iteration 42, loss = 0.56249411
Iteration 43, loss = 0.56114911
Iteration 44, loss = 0.56086134
Iteration 45, loss = 0.56056944
Iteration 46, loss = 0.55936345
Iteration 47, loss = 0.55852710
Iteration 48, loss = 0.55851019
Iteration 49, loss = 0.55749132
Iteration 50, loss = 0.55738122
Iteration 51, loss = 0.55633323
Iteration 52, loss = 0.55564354
Iteration 53, loss = 0.55524605
Iteration 54, loss = 0.55441996
Iteration 55, loss = 0.55426678
Iteration 56, loss = 0.55332774
Iteration 57, loss = 0.55255321
Iteration 58, loss = 0.55244982
Iteration 59, loss = 0.55146988
Iteration 60, loss = 0.55104185
Iteration 61, loss = 0.55096446
Iteration 62, loss = 0.55031568
Iteration 63, loss = 0.55043502
Iteration 64, loss = 0.54969954
Iteration 65, loss = 0.54892858
Iteration 66, loss = 0.54860963
Iteration 67, loss = 0.54798472
Iteration 68, loss = 0.54780431
Iteration 69, loss = 0.54738029
Iteration 70, loss = 0.54716898
Iteration 71, loss = 0.54687350
Iteration 72, loss = 0.54641705
Iteration 73, loss = 0.54568066
Iteration 74, loss = 0.54560661
Iteration 75, loss = 0.54527783
Iteration 76, loss = 0.54473610
Iteration 77, loss = 0.54478472
Iteration 78, loss = 0.54392708
Iteration 79, loss = 0.54368565
Iteration 80, loss = 0.54357411
Iteration 81, loss = 0.54314657
Iteration 82, loss = 0.54276026
Iteration 83, loss = 0.54260631
Iteration 84, loss = 0.54213039
Iteration 85, loss = 0.54194411
```

```
Iteration 86, loss = 0.54155478
Iteration 87, loss = 0.54092032
Iteration 88, loss = 0.54114961
Iteration 89, loss = 0.54057024
Iteration 90, loss = 0.54058092
Iteration 91, loss = 0.53990828
Iteration 92, loss = 0.54006170
Iteration 93, loss = 0.53960155
Iteration 94, loss = 0.53951973
Iteration 95, loss = 0.53907502
Iteration 96, loss = 0.53863086
Iteration 97, loss = 0.53843325
Iteration 98, loss = 0.53904078
Iteration 99, loss = 0.53781627
Iteration 100, loss = 0.53770548
Iteration 101, loss = 0.53732914
Iteration 102, loss = 0.53743320
Iteration 103, loss = 0.53687673
Iteration 104, loss = 0.53681852
Iteration 105, loss = 0.53646370
Iteration 106, loss = 0.53636600
Iteration 107, loss = 0.53618170
Iteration 108, loss = 0.53599862
Iteration 109, loss = 0.53585554
Iteration 110, loss = 0.53582597
Iteration 111, loss = 0.53524551
Iteration 112, loss = 0.53488718
Iteration 113, loss = 0.53482830
Iteration 114, loss = 0.53466318
Iteration 115, loss = 0.53418741
Iteration 116, loss = 0.53408840
Iteration 117, loss = 0.53396246
Iteration 118, loss = 0.53383203
Iteration 119, loss = 0.53357463
Iteration 120, loss = 0.53317769
Iteration 121, loss = 0.53306250
Iteration 122, loss = 0.53275258
Iteration 123, loss = 0.53264800
Iteration 124, loss = 0.53313076
Iteration 125, loss = 0.53250167
Iteration 126, loss = 0.53182771
Iteration 127, loss = 0.53150795
Iteration 128, loss = 0.53171667
Iteration 129, loss = 0.53164313
Iteration 130, loss = 0.53131218
Iteration 131, loss = 0.53120528
Iteration 132, loss = 0.53100459
Iteration 133, loss = 0.53050383
```

```
Iteration 134, loss = 0.53085368
Iteration 135, loss = 0.53027688
Iteration 136, loss = 0.53056990
Iteration 137, loss = 0.53009048
Iteration 138, loss = 0.53016731
Iteration 139, loss = 0.52950493
Iteration 140, loss = 0.52978870
Iteration 141, loss = 0.52953675
Iteration 142, loss = 0.52929575
Iteration 143, loss = 0.52934671
Iteration 144, loss = 0.52889577
Iteration 145, loss = 0.52873059
Iteration 146, loss = 0.52847753
Iteration 147, loss = 0.52876536
Iteration 148, loss = 0.52867539
Iteration 149, loss = 0.52790965
Iteration 150, loss = 0.52827973
Iteration 151, loss = 0.52801443
Iteration 152, loss = 0.52745572
Iteration 153, loss = 0.52769407
Iteration 154, loss = 0.52728286
Iteration 155, loss = 0.52759251
Iteration 156, loss = 0.52691817
Iteration 157, loss = 0.52741296
Iteration 158, loss = 0.52696423
Iteration 159, loss = 0.52696074
Iteration 160, loss = 0.52703475
Iteration 161, loss = 0.52626316
Iteration 162, loss = 0.52634650
Iteration 163, loss = 0.52604705
Iteration 164, loss = 0.52622755
Iteration 165, loss = 0.52611557
Iteration 166, loss = 0.52578792
Iteration 167, loss = 0.52535538
Iteration 168, loss = 0.52568626
Iteration 169, loss = 0.52546140
Iteration 170, loss = 0.52605715
Iteration 171, loss = 0.52563838
Iteration 172, loss = 0.52518911
Iteration 173, loss = 0.52510565
Iteration 174, loss = 0.52497057
Iteration 175, loss = 0.52479207
Iteration 176, loss = 0.52478137
Iteration 177, loss = 0.52474065
Iteration 178, loss = 0.52446399
Iteration 179, loss = 0.52446773
Iteration 180, loss = 0.52454304
Iteration 181, loss = 0.52407564
```

```
Iteration 182, loss = 0.52412529
Iteration 183, loss = 0.52409485
Iteration 184, loss = 0.52415942
Iteration 185, loss = 0.52364315
Iteration 186, loss = 0.52346680
Iteration 187, loss = 0.52391793
Iteration 188, loss = 0.52320373
Iteration 189, loss = 0.52338617
Iteration 190, loss = 0.52342321
Iteration 191, loss = 0.52277659
Iteration 192, loss = 0.52277373
Iteration 193, loss = 0.52279912
Iteration 194, loss = 0.52250860
Iteration 195, loss = 0.52274338
Iteration 196, loss = 0.52251961
Iteration 197, loss = 0.52240224
Iteration 198, loss = 0.52222373
Iteration 199, loss = 0.52245783
Iteration 200, loss = 0.52197124
```

```
/home/yesternight/Desktop/neural-networks/neural-networks/lib/python3.10/site-
packages/sklearn/neural_network/_multilayer_perceptron.py:702:
ConvergenceWarning: Stochastic Optimizer: Maximum iterations (200) reached and
the optimization hasn't converged yet.
  warnings.warn(
```

```
Iteration 1, loss = 0.65241031
Iteration 2, loss = 0.64426881
Iteration 3, loss = 0.64253779
Iteration 4, loss = 0.63928148
Iteration 5, loss = 0.63655545
Iteration 6, loss = 0.63464104
Iteration 7, loss = 0.63342635
Iteration 8, loss = 0.63213036
Iteration 9, loss = 0.63123336
Iteration 10, loss = 0.63002572
Iteration 11, loss = 0.62898226
Iteration 12, loss = 0.62798991
Iteration 13, loss = 0.62720104
Iteration 14, loss = 0.62597663
Iteration 15, loss = 0.62506688
Iteration 16, loss = 0.62394984
Iteration 17, loss = 0.62302970
Iteration 18, loss = 0.62185313
Iteration 19, loss = 0.62081151
Iteration 20, loss = 0.61960095
Iteration 21, loss = 0.61837719
Iteration 22, loss = 0.61715575
Iteration 23, loss = 0.61604906
```

```
Iteration 24, loss = 0.61467139
Iteration 25, loss = 0.61340065
Iteration 26, loss = 0.61193478
Iteration 27, loss = 0.61067347
Iteration 28, loss = 0.60932239
Iteration 29, loss = 0.60812910
Iteration 30, loss = 0.60630192
Iteration 31, loss = 0.60516977
Iteration 32, loss = 0.60393623
Iteration 33, loss = 0.60221387
Iteration 34, loss = 0.60093012
Iteration 35, loss = 0.59975247
Iteration 36, loss = 0.59840901
Iteration 37, loss = 0.59696681
Iteration 38, loss = 0.59583181
Iteration 39, loss = 0.59450371
Iteration 40, loss = 0.59330507
Iteration 41, loss = 0.59206057
Iteration 42, loss = 0.59093003
Iteration 43, loss = 0.59004244
Iteration 44, loss = 0.58846630
Iteration 45, loss = 0.58747807
Iteration 46, loss = 0.58630800
Iteration 47, loss = 0.58532186
Iteration 48, loss = 0.58419852
Iteration 49, loss = 0.58313208
Iteration 50, loss = 0.58205237
Iteration 51, loss = 0.58103311
Iteration 52, loss = 0.58008964
Iteration 53, loss = 0.57914443
Iteration 54, loss = 0.57844089
Iteration 55, loss = 0.57731919
Iteration 56, loss = 0.57654987
Iteration 57, loss = 0.57561127
Iteration 58, loss = 0.57459828
Iteration 59, loss = 0.57388071
Iteration 60, loss = 0.57278261
Iteration 61, loss = 0.57214440
Iteration 62, loss = 0.57147376
Iteration 63, loss = 0.57081751
Iteration 64, loss = 0.56977881
Iteration 65, loss = 0.56878535
Iteration 66, loss = 0.56862937
Iteration 67, loss = 0.56768649
Iteration 68, loss = 0.56679718
Iteration 69, loss = 0.56633171
Iteration 70, loss = 0.56546066
Iteration 71, loss = 0.56491604
```

```
Iteration 72, loss = 0.56429681
Iteration 73, loss = 0.56343522
Iteration 74, loss = 0.56284342
Iteration 75, loss = 0.56215958
Iteration 76, loss = 0.56153732
Iteration 77, loss = 0.56123101
Iteration 78, loss = 0.56032332
Iteration 79, loss = 0.55979791
Iteration 80, loss = 0.55906654
Iteration 81, loss = 0.55872628
Iteration 82, loss = 0.55831897
Iteration 83, loss = 0.55734773
Iteration 84, loss = 0.55706772
Iteration 85, loss = 0.55636086
Iteration 86, loss = 0.55573488
Iteration 87, loss = 0.55521705
Iteration 88, loss = 0.55467069
Iteration 89, loss = 0.55459787
Iteration 90, loss = 0.55400508
Iteration 91, loss = 0.55332781
Iteration 92, loss = 0.55282544
Iteration 93, loss = 0.55229029
Iteration 94, loss = 0.55200604
Iteration 95, loss = 0.55116401
Iteration 96, loss = 0.55090202
Iteration 97, loss = 0.55038993
Iteration 98, loss = 0.54973155
Iteration 99, loss = 0.54953737
Iteration 100, loss = 0.54945872
Iteration 101, loss = 0.54889323
Iteration 102, loss = 0.54803701
Iteration 103, loss = 0.54794918
Iteration 104, loss = 0.54753202
Iteration 105, loss = 0.54691535
Iteration 106, loss = 0.54665001
Iteration 107, loss = 0.54618754
Iteration 108, loss = 0.54619437
Iteration 109, loss = 0.54559935
Iteration 110, loss = 0.54523394
Iteration 111, loss = 0.54474528
Iteration 112, loss = 0.54436574
Iteration 113, loss = 0.54405184
Iteration 114, loss = 0.54378215
Iteration 115, loss = 0.54347010
Iteration 116, loss = 0.54297102
Iteration 117, loss = 0.54275902
Iteration 118, loss = 0.54261253
Iteration 119, loss = 0.54206435
```

```
Iteration 120, loss = 0.54164340
Iteration 121, loss = 0.54159164
Iteration 122, loss = 0.54114864
Iteration 123, loss = 0.54086298
Iteration 124, loss = 0.54031448
Iteration 125, loss = 0.54025226
Iteration 126, loss = 0.54003622
Iteration 127, loss = 0.53948536
Iteration 128, loss = 0.53925780
Iteration 129, loss = 0.53876216
Iteration 130, loss = 0.53870060
Iteration 131, loss = 0.53835545
Iteration 132, loss = 0.53832403
Iteration 133, loss = 0.53781475
Iteration 134, loss = 0.53767243
Iteration 135, loss = 0.53716193
Iteration 136, loss = 0.53713294
Iteration 137, loss = 0.53680800
Iteration 138, loss = 0.53646147
Iteration 139, loss = 0.53631213
Iteration 140, loss = 0.53591572
Iteration 141, loss = 0.53569865
Iteration 142, loss = 0.53573600
Iteration 143, loss = 0.53509133
Iteration 144, loss = 0.53487868
Iteration 145, loss = 0.53478533
Iteration 146, loss = 0.53448523
Iteration 147, loss = 0.53458673
Iteration 148, loss = 0.53407529
Iteration 149, loss = 0.53374765
Iteration 150, loss = 0.53369386
Iteration 151, loss = 0.53348922
Iteration 152, loss = 0.53291767
Iteration 153, loss = 0.53308259
Iteration 154, loss = 0.53247853
Iteration 155, loss = 0.53215712
Iteration 156, loss = 0.53225970
Iteration 157, loss = 0.53202129
Iteration 158, loss = 0.53187141
Iteration 159, loss = 0.53153100
Iteration 160, loss = 0.53142616
Iteration 161, loss = 0.53114845
Iteration 162, loss = 0.53082988
Iteration 163, loss = 0.53061923
Iteration 164, loss = 0.53034960
Iteration 165, loss = 0.53031128
Iteration 166, loss = 0.53015430
Iteration 167, loss = 0.52978552
```

```
Iteration 168, loss = 0.52959575
Iteration 169, loss = 0.52936514
Iteration 170, loss = 0.52935913
Iteration 171, loss = 0.52919053
Iteration 172, loss = 0.52885286
Iteration 173, loss = 0.52873863
Iteration 174, loss = 0.52852438
Iteration 175, loss = 0.52830472
Iteration 176, loss = 0.52802553
Iteration 177, loss = 0.52802453
Iteration 178, loss = 0.52753816
```

4.2.3 Experimento3

[ ]:
```
mlp1 = MLPClassifier(solver="lbfgs",verbose=True)
mlp2 = MLPClassifier(solver="sgd", verbose=True)
mlp3 = MLPClassifier(solver="adam", verbose=True)

model_file = MLP_MODEL_ENS_PATH+"3"
if not exists(model_file):
    mlp_ens = VotingClassifier([('mlp1', mlp1), ('mlp2', mlp2), ('mlp3',␣
 ↪mlp1)], voting='soft')
    mlp_ens.fit(x_train,y_train)
    with open(model_file, 'wb') as file:
        pickle.dump(mlp_ens,file)
else:
    print("Model was already trained")
    with open(model_file, 'rb') as file:
        mlp_ens = pickle.load(file)
```

[ ]:
```
mlp_ens.fit(x_train, y_train)
mlp_pred_class = mlp_ens.predict(x_test)
mlp_pred_scores = mlp_ens.predict_proba(x_test)
accuracy, recall, precision, f1, auroc, aupr = metrics.
 ↪compute_performance_metrics(y_test, mlp_pred_class, mlp_pred_scores)
print('Performance no conjunto de validação:')
metrics.print_metrics_summary(accuracy, recall, precision, f1, auroc, aupr)
```

4.2.4 Experimento4

[ ]:
```
mlp1 = MLPClassifier(hidden_layer_size=(16,), activation="relu", solver="adam",␣
 ↪verbose=True)
mlp2 = MLPClassifier(hidden_layer_size=(64,32), activation="tanh",␣
 ↪solver="sgd", verbose=True)
mlp3 = MLPClassifier(activation="logistic", solver="lbfgs", verbose=True)

model_file = MLP_MODEL_ENS_PATH+"4"
```

```
if not exists(model_file):
    mlp_ens = VotingClassifier([('mlp1', mlp1), ('mlp2', mlp2), ('mlp3',␣
 ↪mlp1)], voting='soft')
    mlp_ens.fit(x_train,y_train)
    with open(model_file, 'wb') as file:
        pickle.dump(mlp_ens,file)
else:
    print("Model was already trained")
    with open(model_file, 'rb') as file:
        mlp_ens = pickle.load(file)
```

[ ]:
```
mlp_ens.fit(x_train, y_train)
mlp_pred_class = mlp_ens.predict(x_test)
mlp_pred_scores = mlp_ens.predict_proba(x_test)
accuracy, recall, precision, f1, auroc, aupr = metrics.
 ↪compute_performance_metrics(y_test, mlp_pred_class, mlp_pred_scores)
print('Performance no conjunto de validação:')
metrics.print_metrics_summary(accuracy, recall, precision, f1, auroc, aupr)
```

4.3 Gradient Boosting

[ ]:
```
GB_PATH = "Models/GB"
GB_MODEL_PATH = f"{GB_PATH}/model"
```

4.3.1 Experimento1

[ ]:
```
gb_clf = GradientBoostingClassifier()

model_file = GB_MODEL_PATH+"1"
if not exists(model_file):
    gb_clf.fit(x_train, y_train)
    with open(model_file, 'wb') as file:
        pickle.dump(gb_clf,file)
else:
    print("Model was already trained")
    with open(model_file, 'rb') as file:
        gb_clf = pickle.load(file)

gb_pred_class = gb_clf.predict(x_test)
gb_pred_scores = gb_clf.predict_proba(x_test)
accuracy, recall, precision, f1, auroc, aupr = metrics.
 ↪compute_performance_metrics(y_test, gb_pred_class, gb_pred_scores)
metrics.print_metrics_summary(accuracy, recall, precision, f1, auroc, aupr)
```

4.3.2 Experimento2

[ ]:
```
gb_clf = GradientBoostingClassifier(n_estimators=500)

model_file = GB_MODEL_PATH+"2"
if not exists(model_file):
    gb_clf.fit(x_train, y_train)
    with open(model_file, 'wb') as file:
        pickle.dump(gb_clf,file)
else:
    print("Model was already trained")
    with open(model_file, 'rb') as file:
        gb_clf = pickle.load(file)

gb_pred_class = gb_clf.predict(x_test)
gb_pred_scores = gb_clf.predict_proba(x_test)
accuracy, recall, precision, f1, auroc, aupr = metrics.
 ↪compute_performance_metrics(y_test, gb_pred_class, gb_pred_scores)
metrics.print_metrics_summary(accuracy, recall, precision, f1, auroc, aupr)
```

4.3.3 Experimento3

[ ]:
```
gb_clf = GradientBoostingClassifier(n_estimators=50)

model_file = GB_MODEL_PATH+"3"
if not exists(model_file):
    gb_clf.fit(x_train, y_train)
    with open(model_file, 'wb') as file:
        pickle.dump(gb_clf,file)
else:
    print("Model was already trained")
    with open(model_file, 'rb') as file:
        gb_clf = pickle.load(file)

gb_pred_class = gb_clf.predict(x_test)
gb_pred_scores = gb_clf.predict_proba(x_test)
accuracy, recall, precision, f1, auroc, aupr = metrics.
 ↪compute_performance_metrics(y_test, gb_pred_class, gb_pred_scores)
metrics.print_metrics_summary(accuracy, recall, precision, f1, auroc, aupr)
```

4.3.4 Experimento4

[ ]:
```
gb_clf = GradientBoostingClassifier(n_estimators=10)

model_file = GB_MODEL_PATH+"4"
if not exists(model_file):
    gb_clf.fit(x_train, y_train)
```

```
    with open(model_file, 'wb') as file:
        pickle.dump(gb_clf,file)
else:
    print("Model was already trained")
    with open(model_file, 'rb') as file:
        gb_clf = pickle.load(file)

gb_pred_class = gb_clf.predict(x_test)
gb_pred_scores = gb_clf.predict_proba(x_test)
accuracy, recall, precision, f1, auroc, aupr = metrics.
 ↪compute_performance_metrics(y_test, gb_pred_class, gb_pred_scores)
metrics.print_metrics_summary(accuracy, recall, precision, f1, auroc, aupr)
```

4.3.5 Experimento5

[ ]:
```
gb_clf = GradientBoostingClassifier(learning_rate=0.5)

model_file = GB_MODEL_PATH+"5"
if not exists(model_file):
    gb_clf.fit(x_train, y_train)
    with open(model_file, 'wb') as file:
        pickle.dump(gb_clf,file)
else:
    print("Model was already trained")
    with open(model_file, 'rb') as file:
        gb_clf = pickle.load(file)

gb_pred_class = gb_clf.predict(x_test)
gb_pred_scores = gb_clf.predict_proba(x_test)
accuracy, recall, precision, f1, auroc, aupr = metrics.
 ↪compute_performance_metrics(y_test, gb_pred_class, gb_pred_scores)
metrics.print_metrics_summary(accuracy, recall, precision, f1, auroc, aupr)
```

4.3.6 Experimento6

[ ]:
```
gb_clf = GradientBoostingClassifier(learning_rate=0.05)

model_file = GB_MODEL_PATH+"6"
if not exists(model_file):
    gb_clf.fit(x_train, y_train)
    with open(model_file, 'wb') as file:
        pickle.dump(gb_clf,file)
else:
    print("Model was already trained")
    with open(model_file, 'rb') as file:
        gb_clf = pickle.load(file)
```

```
gb_pred_class = gb_clf.predict(x_test)
gb_pred_scores = gb_clf.predict_proba(x_test)
accuracy, recall, precision, f1, auroc, aupr = metrics.
 ↪compute_performance_metrics(y_test, gb_pred_class, gb_pred_scores)
metrics.print_metrics_summary(accuracy, recall, precision, f1, auroc, aupr)
```

4.3.7 Experimento7

[ ]:
```
gb_clf = GradientBoostingClassifier(learning_rate=0.01)

model_file = GB_MODEL_PATH+"7"
if not exists(model_file):
    gb_clf.fit(x_train, y_train)
    with open(model_file, 'wb') as file:
        pickle.dump(gb_clf,file)
else:
    print("Model was already trained")
    with open(model_file, 'rb') as file:
        gb_clf = pickle.load(file)

gb_pred_class = gb_clf.predict(x_test)
gb_pred_scores = gb_clf.predict_proba(x_test)
accuracy, recall, precision, f1, auroc, aupr = metrics.
 ↪compute_performance_metrics(y_test, gb_pred_class, gb_pred_scores)
metrics.print_metrics_summary(accuracy, recall, precision, f1, auroc, aupr)
```

4.4 Random Forest

4.5 Ensemble Misto

4.6 SVC

[ ]:
```

```