REGRESSÃO LOGÍSTICA

Vinícius Loeschner - 39725
João Pedro – 39005
Gabriel Neves - 39771

CRONOGRAMA

1. Introdução a Regressão Logística
2. Data Base
3. Código
4. Gráficos
5. Referências
6. Perguntas

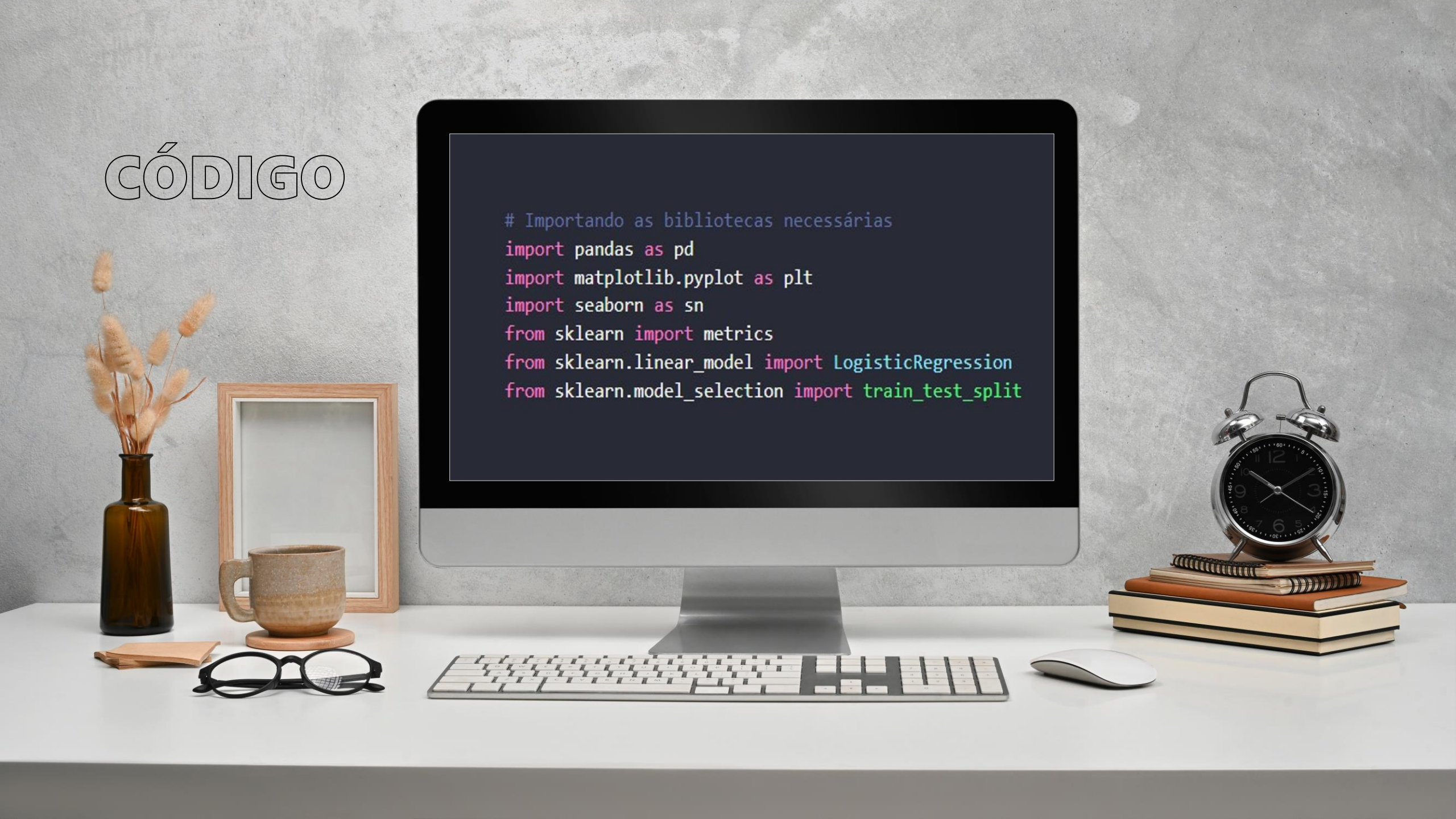
1 REGRESSÃO LOGÍSTICA


2
Data Base


CREDIT DATA
Status
Seniority
Home
Time
Age
Marital
Records
Job
Expenses
Income
Assets
Debt
Amount
Price


3
CÓDIGO

CÓDIGO


# Importando as bibliotecas necessárias
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sn
from sklearn import metrics
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import train_test_split

```
# Importando as bibliotecas necessárias
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sn
from sklearn import metrics
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import train_test_split
```

```
# Lendo o conjunto de dados
df = pd.read_csv('G:/My Drive/College/Assignments/Statistics/credit_data.csv')

# Removendo as linhas com valores faltantes
df = df.dropna()
```

```python
# Convertendo a coluna 'Status' para binário
df['Status'] = df['Status'].apply(lambda x: 1 if x == 'good' else 0)

# Definindo as colunas numéricas e colunas categóricas
colunas_numericas = ['Seniority', 'Time', 'Age', 'Expenses', 'Income', 'Assets', 'Debt', 'Amount', 'Price']
colunas_categoricas = ['Home', 'Marital', 'Records', 'Job']
```

```python
# Convertendo a coluna 'Status' para binário
df['Status'] = df['Status'].apply(lambda x: 1 if x == 'good' else 0)

# Definindo as colunas numéricas e colunas categóricas
colunas_numericas = ['Seniority', 'Time', 'Age', 'Expenses', 'Income', 'Assets', 'Debt', 'Amount', 'Price']
colunas_categoricas = ['Home', 'Marital', 'Records', 'Job']

# Função para criar variáveis dummy para colunas categóricas
def dummyfy(nome_coluna):
    global df, colunas_numericas
    novas_colunas = df[nome_coluna].unique()
    for coluna in novas_colunas:
        colunas_numericas.append(f'{nome_coluna}_{coluna}')
    dummies = pd.get_dummies(df[nome_coluna], prefix = nome_coluna)
    df = pd.concat([df, dummies], axis=1)
    df = df.drop(nome_coluna, axis=1)
```

```python
# Convertendo a coluna 'Status' para binário
df['Status'] = df['Status'].apply(lambda x: 1 if x == 'good' else 0)

# Definindo as colunas numéricas e colunas categóricas
colunas_numericas = ['Seniority', 'Time', 'Age', 'Expenses', 'Income', 'Assets', 'Debt', 'Amount', 'Price']
colunas_categoricas = ['Home', 'Marital', 'Records', 'Job']

# Função para criar variáveis dummy para colunas categóricas
def dummyfy(nome_coluna):
    global df, colunas_numericas
    novas_colunas = df[nome_coluna].unique()
    for coluna in novas_colunas:
        colunas_numericas.append(f'{nome_coluna}_{coluna}')
    dummies = pd.get_dummies(df[nome_coluna], prefix = nome_coluna)
    df = pd.concat([df, dummies], axis=1)
    df = df.drop(nome_coluna, axis=1)

# Criando variáveis dummy para todas as colunas categóricas
for nome_coluna in colunas_categoricas:
    dummyfy(nome_coluna)
```

ACTION

```
# Definindo as variáveis X e Y
X = df[colunas_numericas]
Y = df['Status']
```

```python
# Definindo uma função para realizar a regressão logística e retornar a precisão
def regressao_logistica(tamanho_teste = 0.2, seed = 1):
    # Definindo as variáveis de treino e teste
    X_treino, X_teste, y_treino, y_teste = train_test_split(X, Y, test_size = tamanho_teste, random_state = seed)

    # Definindo o modelo de regressão logística e ajustando-o aos dados de treino
    regressao_logistica = LogisticRegression(max_iter = 10000)
    regressao_logistica.fit(X_treino, y_treino)
```

```python
# Definindo uma função para realizar a regressão logística e retornar a precisão
def regressao_logistica(tamanho_teste = 0.2, seed = 1):
    # Definindo as variáveis de treino e teste
    X_treino, X_teste, y_treino, y_teste = train_test_split(X, Y, test_size = tamanho_teste, random_state = seed)

    # Definindo o modelo de regressão logística e ajustando-o aos dados de treino
    regressao_logistica = LogisticRegression(max_iter = 10000)
    regressao_logistica.fit(X_treino, y_treino)

    # Predizendo a variável alvo para os dados de teste
    y_predito = regressao_logistica.predict(X_teste)

    # Criando uma matriz de confusão e exibindo-a como um mapa de calor
    matriz_confusao = pd.crosstab(y_teste, y_predito, rownames=['Real'], colnames=['Previsto'])
    sn.heatmap(matriz_confusao, annot=True, cmap='coolwarm')
    plt.show()

    # Calculando a pontuação de precisão e imprimindo-a junto com o tamanho do teste e os valores de seed
    precisao = metrics.accuracy_score(y_teste, y_predito)
    print(f'Tamanho do teste: {tamanho_teste} | seed: {seed} | Precisão: {precisao}')
```

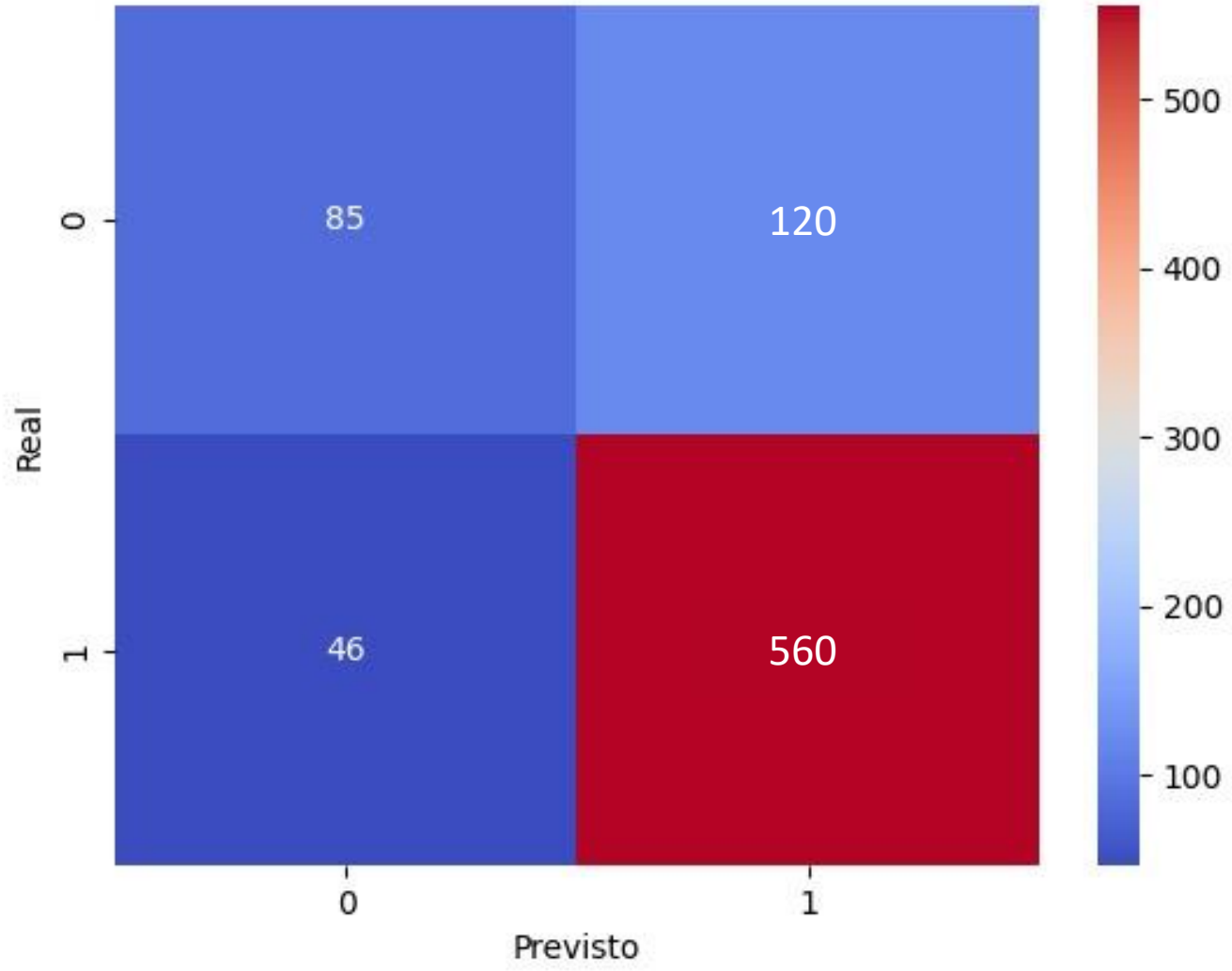
```python
# Calculando a curva ROC e a área sob a curva (AUC)
fpr, tpr, thresholds = metrics.roc_curve(y_teste, y_predito)
# fpr = false positive rate (taxa de falsos positivos)
# tpr = true positive rate (taxa de verdadeiros positivos)
```

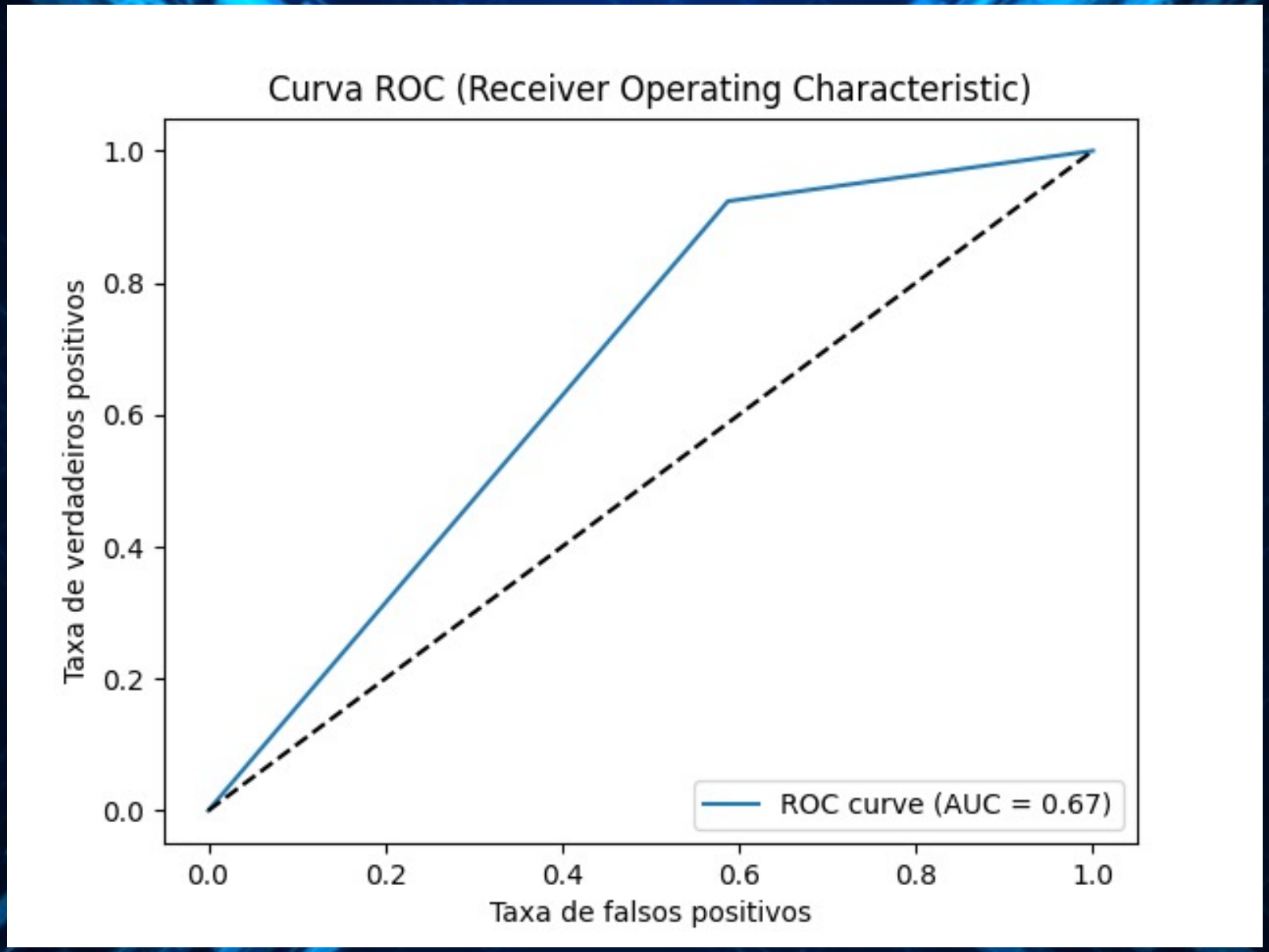
```python
# Calculando a curva ROC e a área sob a curva (AUC)
fpr, tpr, thresholds = metrics.roc_curve(y_teste, y_predito)
# fpr = false positive rate (taxa de falsos positivos)
# tpr = true positive rate (taxa de verdadeiros positivos)

auc = metrics.roc_auc_score(y_teste, y_predito)
# auc = area under the curve (área sob a curva)
plt.plot(fpr, tpr, label='ROC curve (AUC = %0.2f)' % auc)
plt.plot([0, 1], [0, 1], 'k--')  # Linha diagonal (classificador aleatório)
plt.xlabel('Taxa de falsos positivos')
plt.ylabel('Taxa de verdadeiros positivos')
plt.title('Curva ROC (Receiver Operating Characteristic)')
plt.legend(loc='lower right')
plt.show()
# Retornando o tamanho do teste, seed e pontuação de precisão
return tamanho_teste, seed, precisao
```

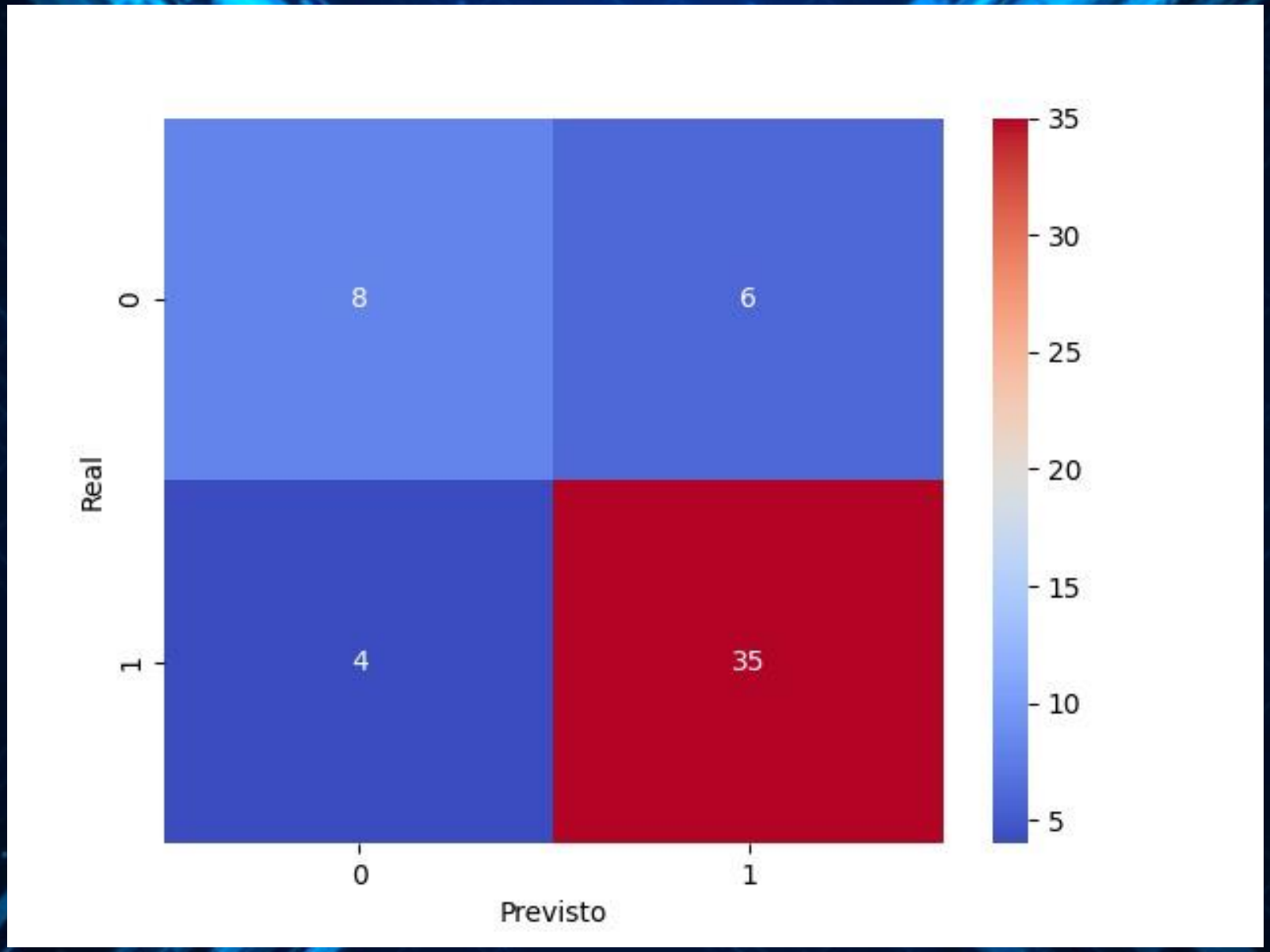
4

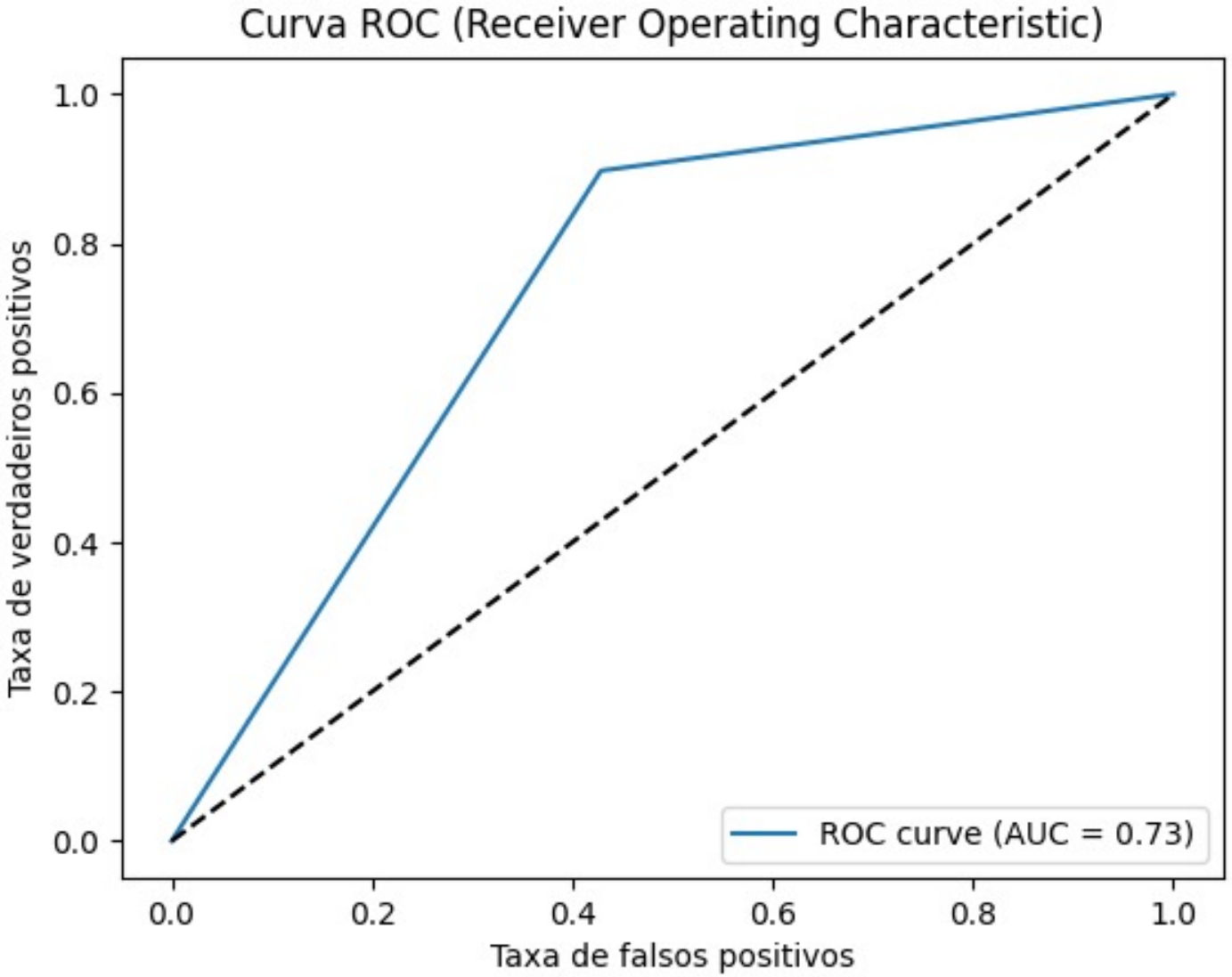
Gráficos

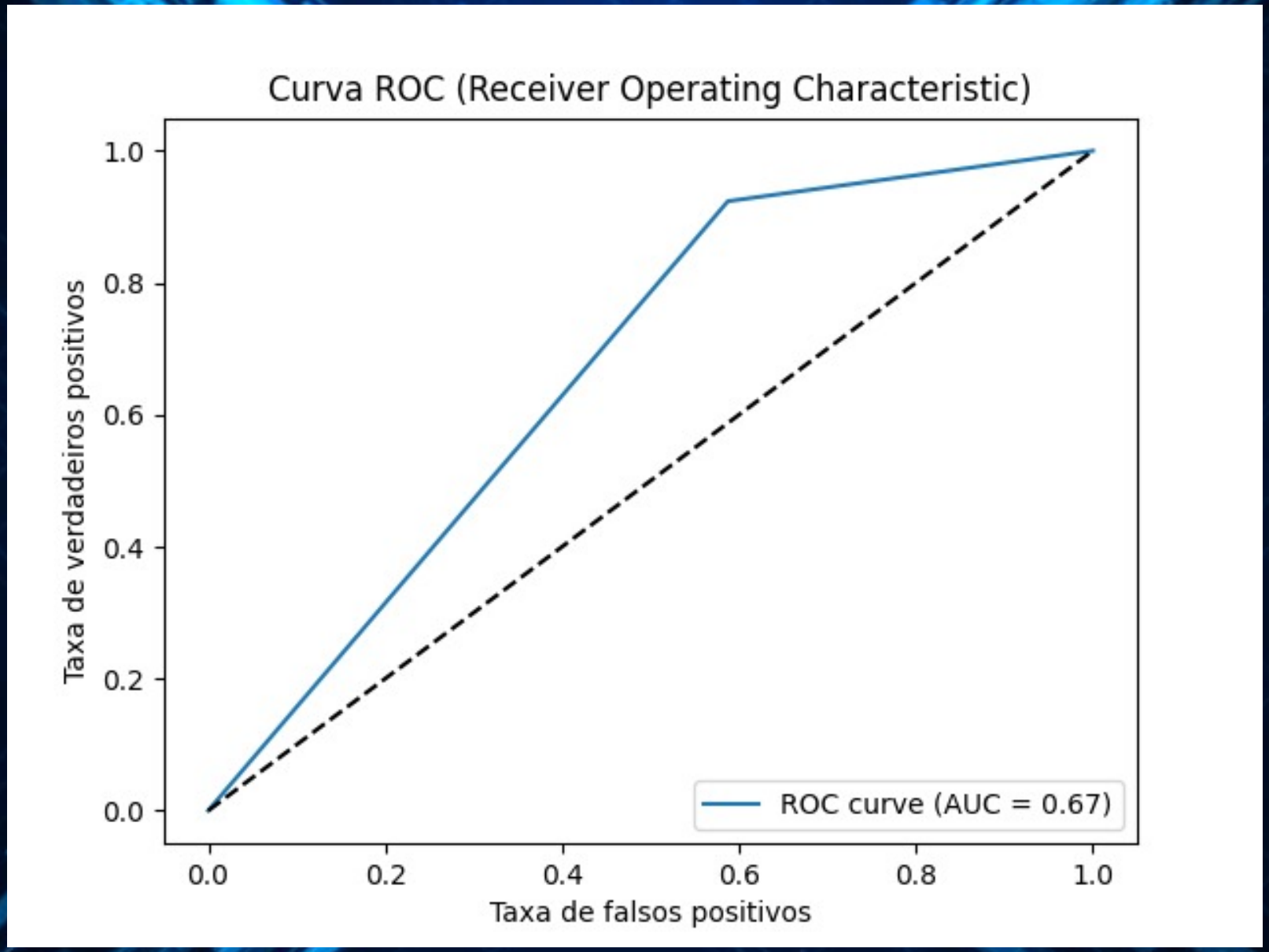
811

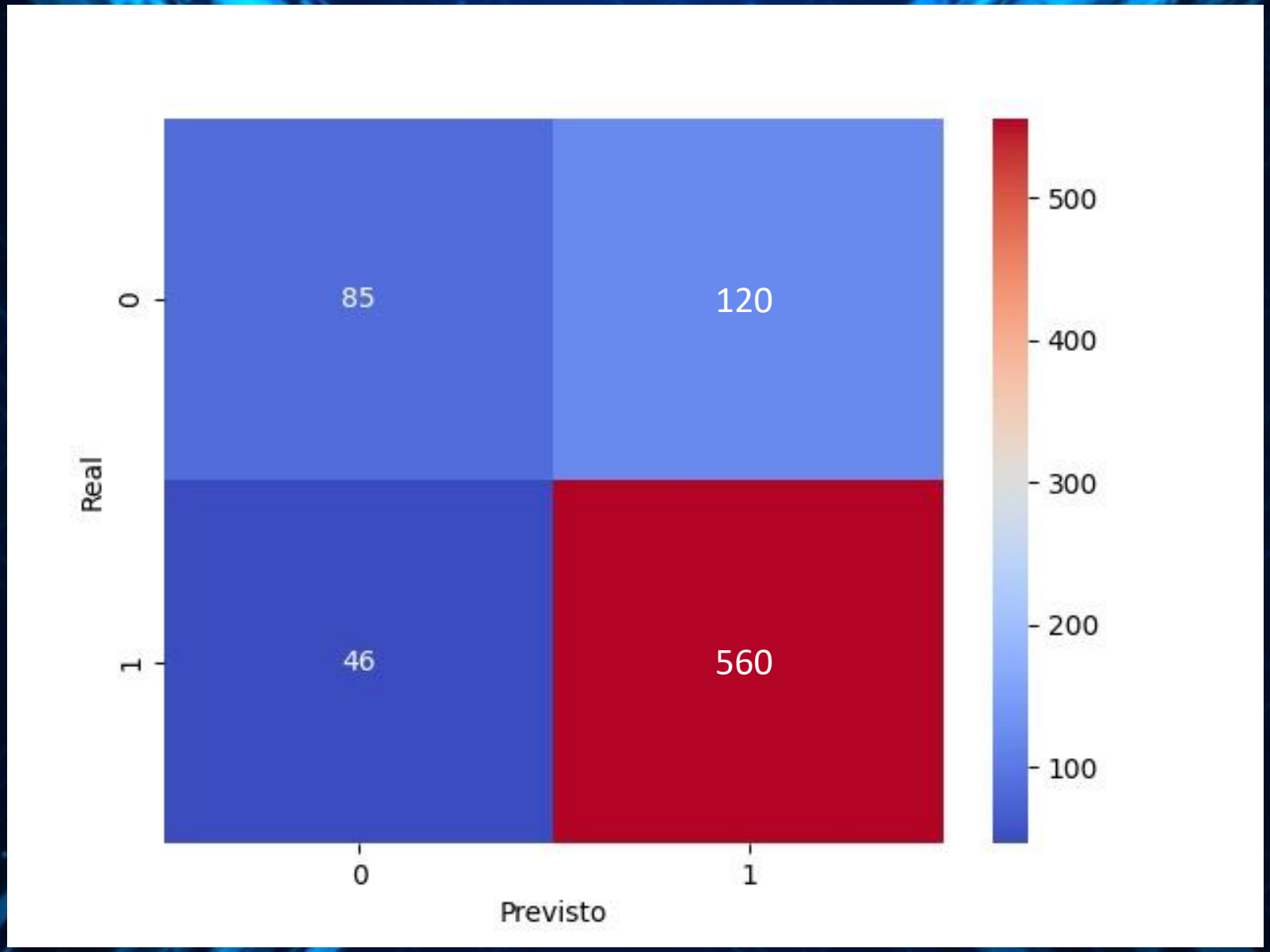



Curva ROC (Receiver Operating Characteristic)
Taxa de verdadeiros positivos
Taxa de falsos positivos
0.0
0.2
0.4
0.6
0.8
1.0
ROC curve (AUC = 0.67)


8
6
4
35
Real
0
1
0
1
Previsto
35
30
25
20
15
10
5


Curva ROC (Receiver Operating Characteristic)
Taxa de verdadeiros positivos
Taxa de falsos positivos
0.0
0.2
0.4
0.6
0.8
1.0
ROC curve (AUC = 0.73)


Curva ROC (Receiver Operating Characteristic)
Taxa de verdadeiros positivos
Taxa de falsos positivos
0.0
0.2
0.4
0.6
0.8
1.0
ROC curve (AUC = 0.67)


85
120
46
560
0
1
Real
0
1
Previsto
500
400
300
200
100

5

Referências

SKlearn

Vincentarelbundock

6