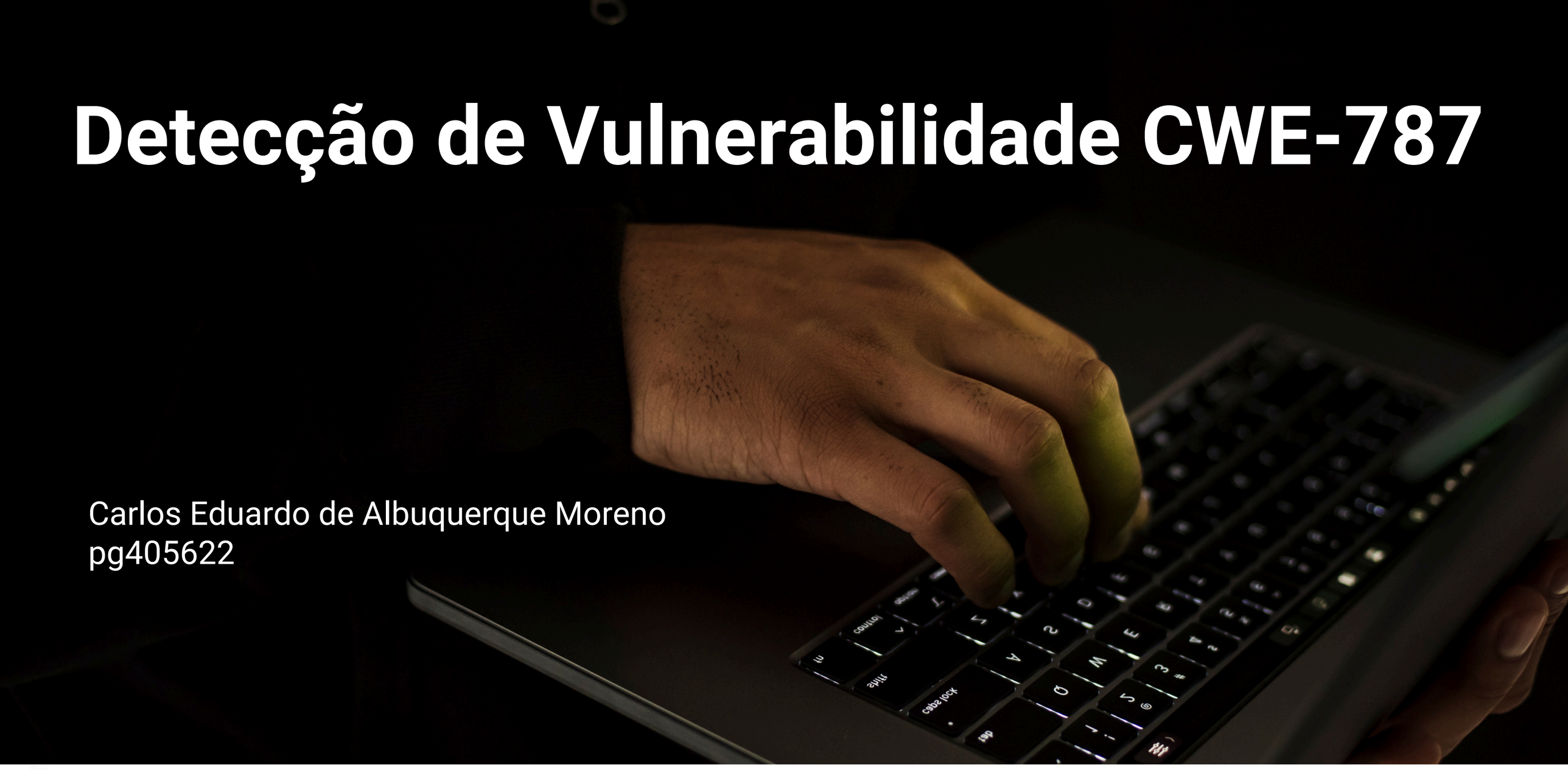
Detecção de Vulnerabilidade CWE-787



Carlos Eduardo de Albuquerque Moreno
pg405622

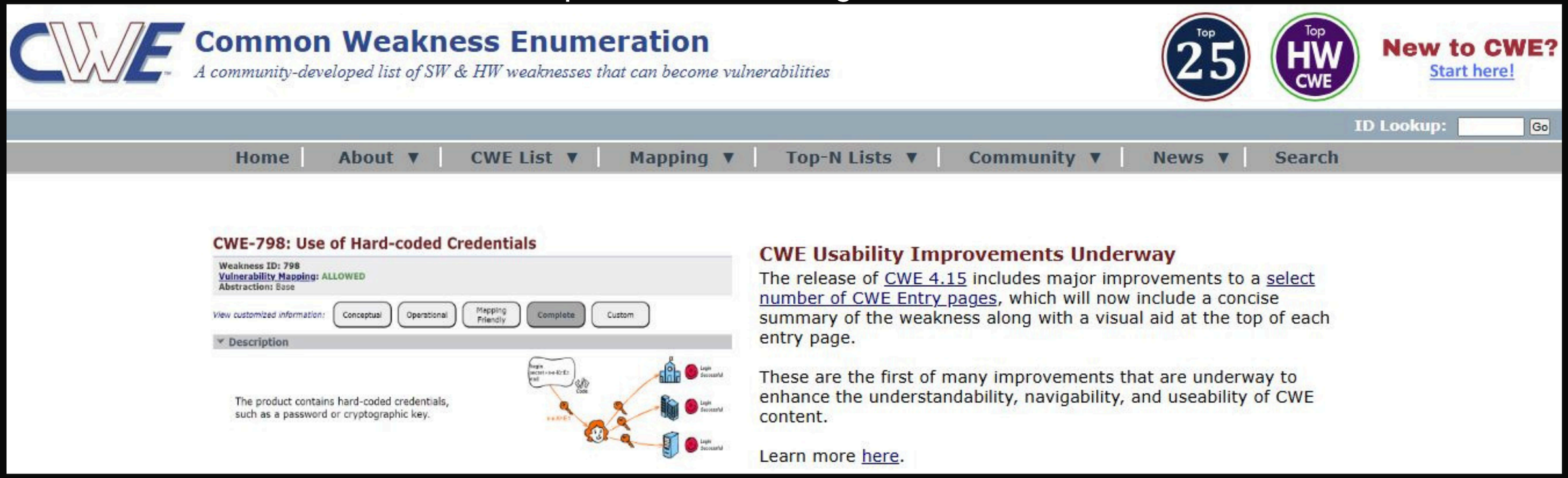
Programa de
Pós-Graduação em
Ciência da Computação

UEM Universidade
Estadual de
Maringá

CWE - COMMON WEAKNESS ENUMERATION

Catálogo amplamente reconhecido que classifica e descreve tipos de vulnerabilidades e fraquezas em software.
Ele serve como uma linguagem comum para identificar falhas, facilitar a comunicação entre desenvolvedores e melhorar a segurança do software ao orientar correções e prevenções.

https://cwe.mitre.org/index.html



ESCOLHA DO DATASET & LIMPEZA DE DADOS

?

CWE (939 Vulnerabilidades e 374 Categorias)

Dataset “2023 CWE Top 25 Most Dangerous Software Weaknesses”

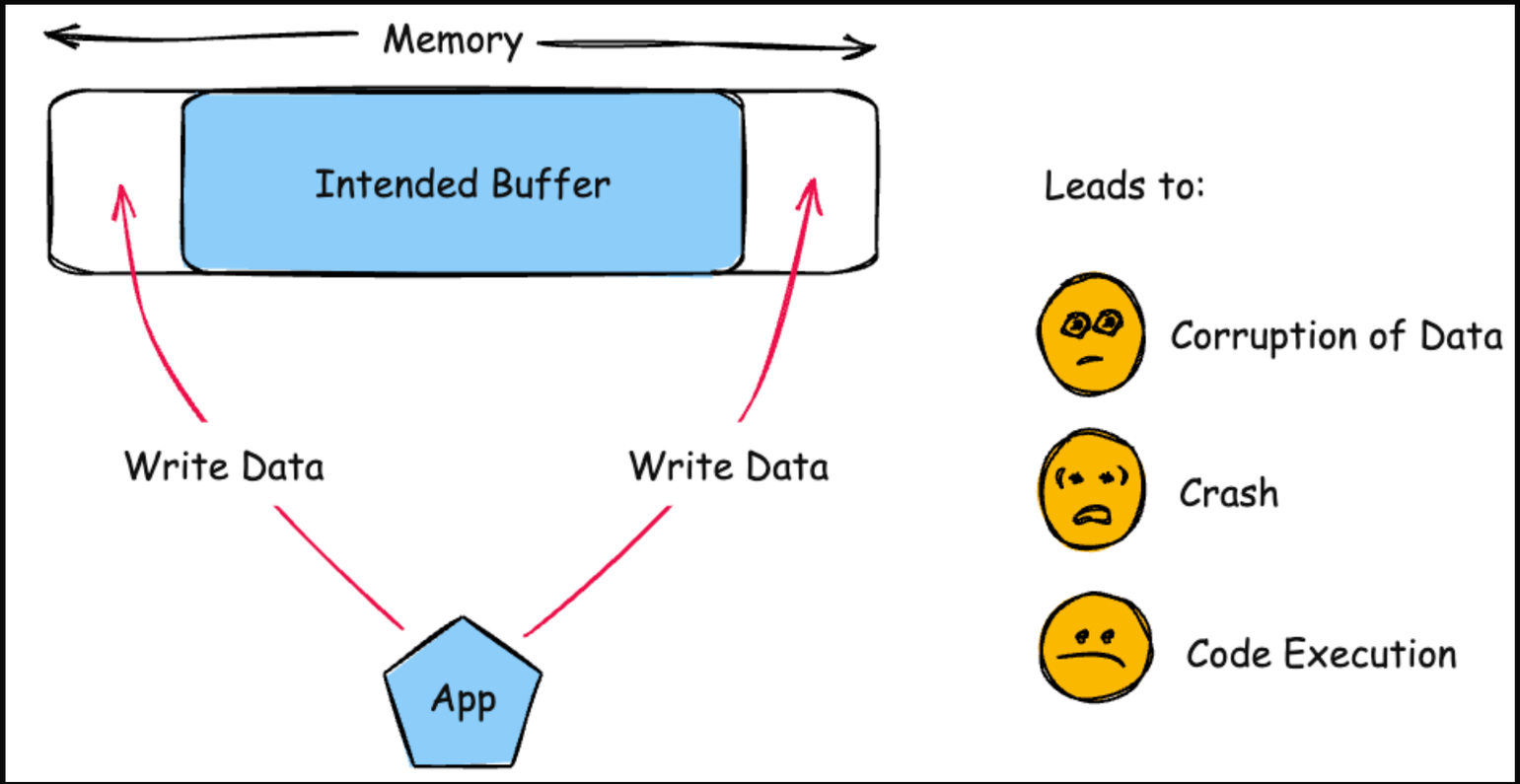
CWE-787 (Out-of-bounds Write)

CWE-787

Vulnerabilidade de segurança que corresponde a um problema conhecido como "Out-of-Bounds Write" (escrita fora dos limites).

- Ocorre quando um programa escreve dados em uma área de memória fora dos limites alocados para uma variável ou buffer.

https://cwe.mitre.org/data/definitions/787.html



- Pode ocorrer em linguagens de programação que manipulam diretamente a memória, frequentemente quando há operações de cópia de dados sem a devida verificação de limites.

Exemplos comuns:

- Acesso a memória fora de um array: Código tenta escrever além do tamanho de um array alocado pode sobrescrever áreas de memória causando comportamento inesperado.
- Buffer overflow: Quando os dados são escritos para um buffer maior do que o espaço alocado, corrompendo outras variáveis ou áreas de memória adjacentes.
- Falta de verificação de limites: Quando o código não verifica adequadamente o tamanho de dados antes de escrevê-los em um buffer ou variável.

Impacto de segurança:

- Execução de código malicioso.
- Corrompimento de dados.
- Acesso não autorizado através de falhas na segurança.

Dataset 01: Vulnerável

- Contém todos os exemplos de códigos comprometidos pela vulnerabilidade CWE-787 disponíveis através do site da comunidade CWE.
- Exemplos diversos encontrados em várias linguagens também contendo a vulnerabilidade.

Dataset 02: Seguro

- Trechos de códigos que não contem a vulnerabilidade CWE-787 e suas características.

- C - Impressão, soma de números, funções, loop for, manipulação de arrays.
- Java - Impressão, soma de números, funções, loop for, manipulação de arrays.
- JavaScript - Impressão, soma de números, funções, loop for, manipulação de arrays.
- C# - Impressão, soma de números, funções, loop for, manipulação de arrays.
- Ruby - Impressão, soma de números, funções, loop times, manipulação de arrays.
- Perl - Impressão, soma de números, funções, loop foreach, verificação de paridade.
- PHP - Impressão, soma de números, funções, loop foreach, verificação de paridade.
- Swift - Impressão, soma de números, funções, loop for-in, verificação de paridade.
- Kotlin - Impressão, soma de números, funções, loop for-in, verificação de paridade.
- Go - Impressão, soma de números, funções, loop range, verificação de paridade.

Bibliotecas utilizadas

Pandas (pd)

Utilizado para leitura, manipulação e análise de dados. No código, é usado para ler datasets CSV.

Scikit-learn (sklearn)

Biblioteca de machine learning que fornece várias ferramentas para modelagem de dados, incluindo processos de classificação, regressão e clustering.

- TfidfVectorizer: Transforma texto em uma matriz de features TF-IDF.
- train_test_split: Separa os dados em conjuntos de treino e teste.
- MultinomialNB: Implementa o classificador Naive Bayes para variáveis categóricas multinomiais.
- classification_report, confusion_matrix: Ferramentas para avaliar o desempenho de modelos de classificação.

Seaborn (sns)

Biblioteca para visualização de dados baseada em Matplotlib. Facilita a criação de gráficos estatísticos atraentes e informativos.

Matplotlib (plt)

Biblioteca de plotagem para criar gráficos em 2D.

Numpy (np)

Biblioteca para computação numérica em Python. Oferece suporte para arrays multidimensionais e várias operações matemáticas de alto desempenho.

Processo Passo a Passo

- Leitura dos Dados:

Dois arquivos CSV são carregados, um com exemplos de códigos vulneráveis e outro com exemplos de códigos seguros.

- Etiqueta dos Dados (label):

Os códigos vulneráveis são marcados com 1 e os códigos seguros com 0.

- Combinação dos Dados:

Os dados são combinados em um único conjunto.

- Transformação dos Dados:

O texto dos códigos são convertidos em números utilizando a técnica TF-IDF.

- Divisão dos Dados:
  - Treinamento: 70% dos dados para treinar o modelo.
  - Teste: 15% dos dados para avaliar como o modelo funciona com dados novos.
  - Validação: 15% dos dados para ajustes e validar o modelo durante a fase de treino.
- Validação Cruzada (K-Fold Cross-Validation):

Dividiu-se os 70% dos dados de treinamento em 5 partes (folds) para treinar e validar o modelo 5 vezes, usando cada parte uma vez para validação e as outras 4 para treinamento. A média dos resultados é calculada para obter uma avaliação mais robusta do modelo.

- Treinamento Final:

O modelo é treinado usando todos os 70% de dados de treinamento após a validação cruzada.

- Avaliação Final:

O modelo final é testado usando os 15% dos dados de teste para verificar sua performance em dados completamente novos.

Matriz de confusão

	Previsto Seguro (0)	Previsto Vulnerável (1)
Realmente Seguro (0)	16	3
Realmente Vulnerável (1)	1	1

Verdadeiros Negativos (TN): 16

- **O modelo previu corretamente 16 instâncias como seguras, e elas realmente eram seguras.**

Falsos Positivos (FP): 3

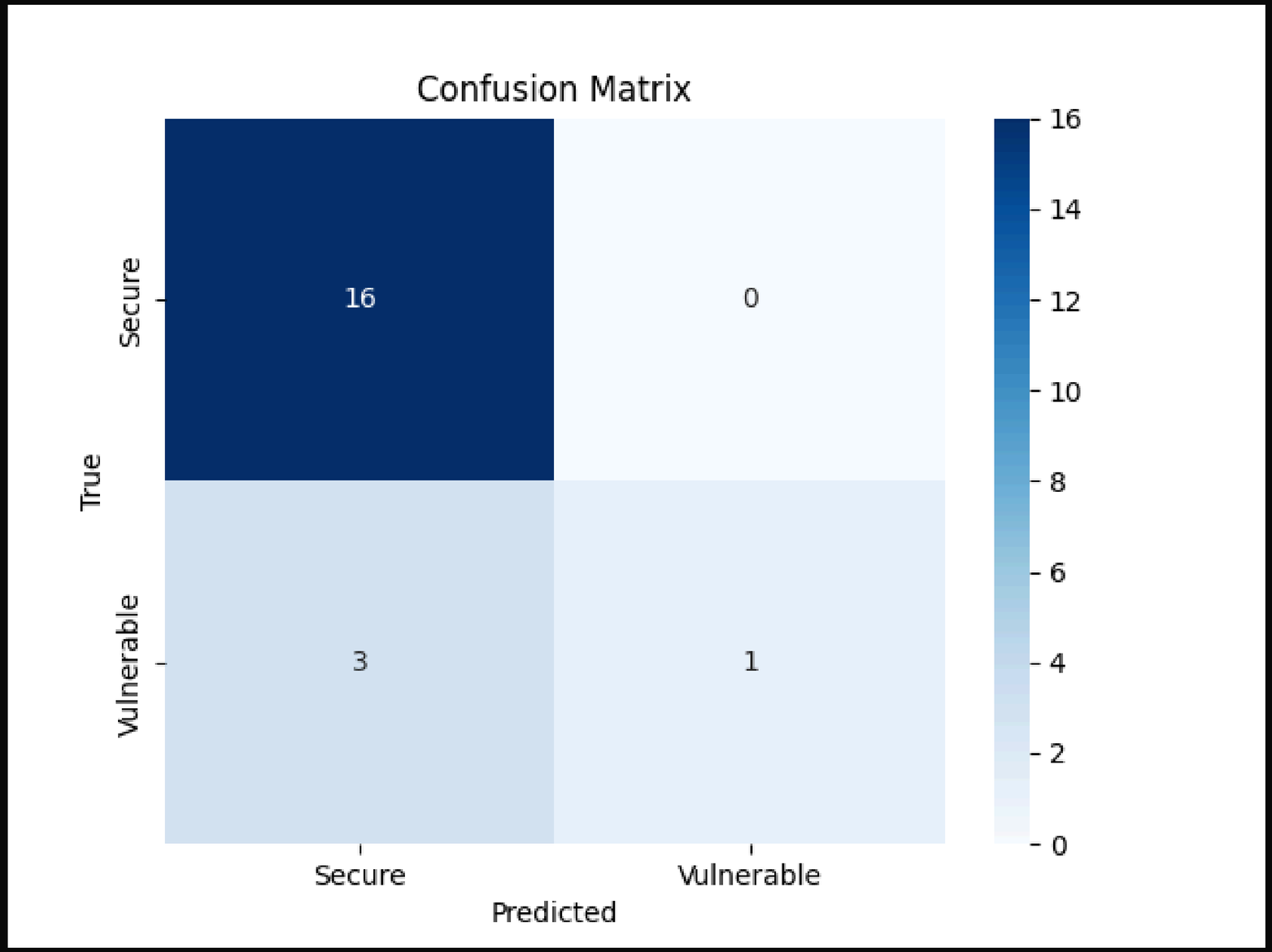
- **O modelo previu 3 instâncias como vulneráveis, mas elas eram na verdade seguras (falsos alarmes).**

Falsos Negativos (FN): 1

- **O modelo previu 1 instância como segura, mas ela era realmente vulnerável (erro não detectado).**

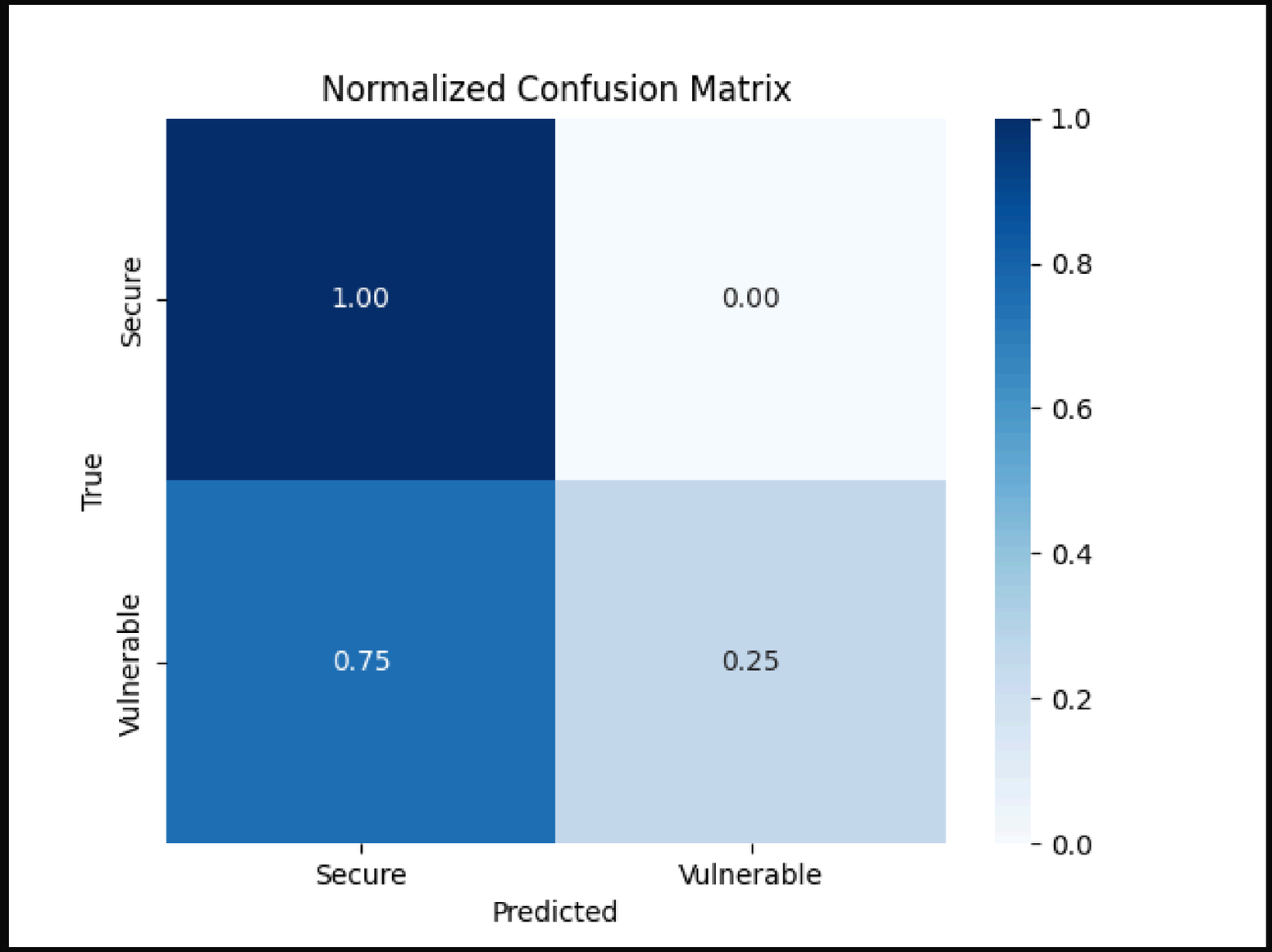
Verdadeiros Positivos (TP): 1

- **Houve 1 instância corretamente prevista como vulnerável.**



Matriz de confusão normalizada

Mostra as proporções de verdadeiros positivos e negativos facilitando a comparação entre classes.

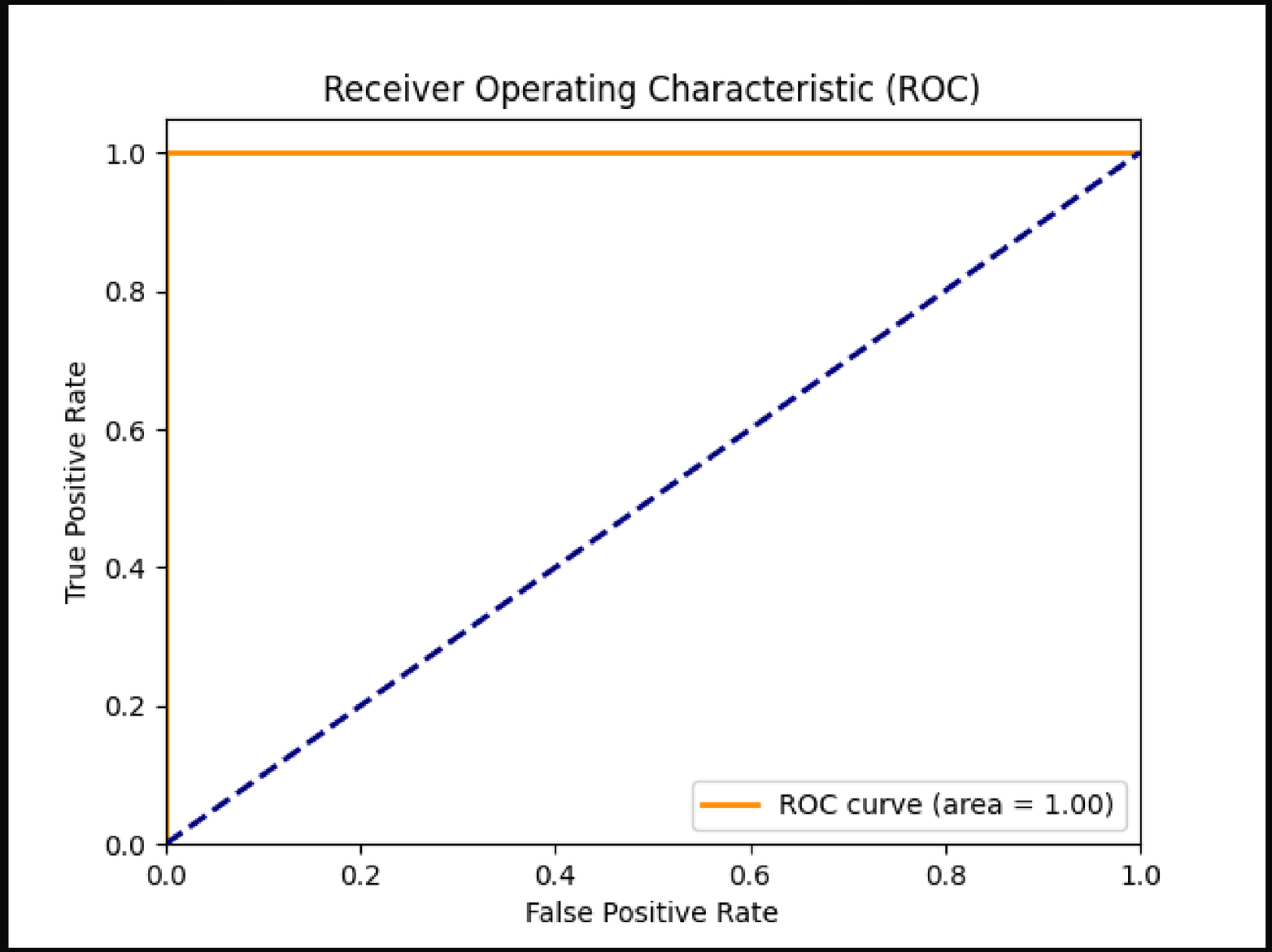


Característica Operacional do Receptor

Exibe a relação entre a taxa de falsos positivos (FPR) e a taxa de verdadeiros positivos (TPR).

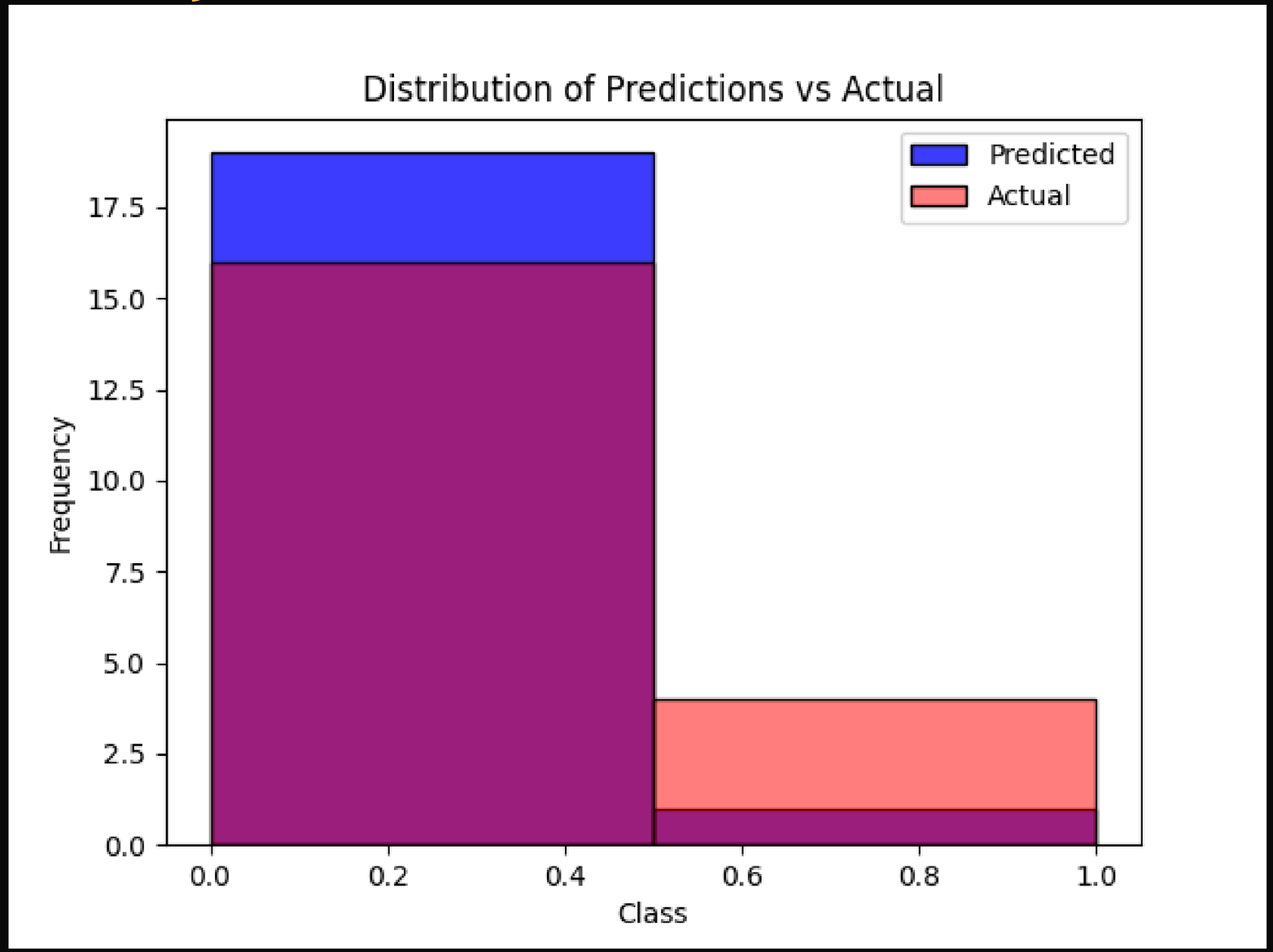
AUC é a área sob a curva ROC, indicando a performance geral do modelo.

Curva ROC ideal seria a que se aproxima do canto superior esquerdo, onde a TPR é 1 (100%) e a FPR é 0 (0%). Isso indica que o modelo está acertando todos os positivos e não está cometendo falsos positivos.



Distribuição das Previsões

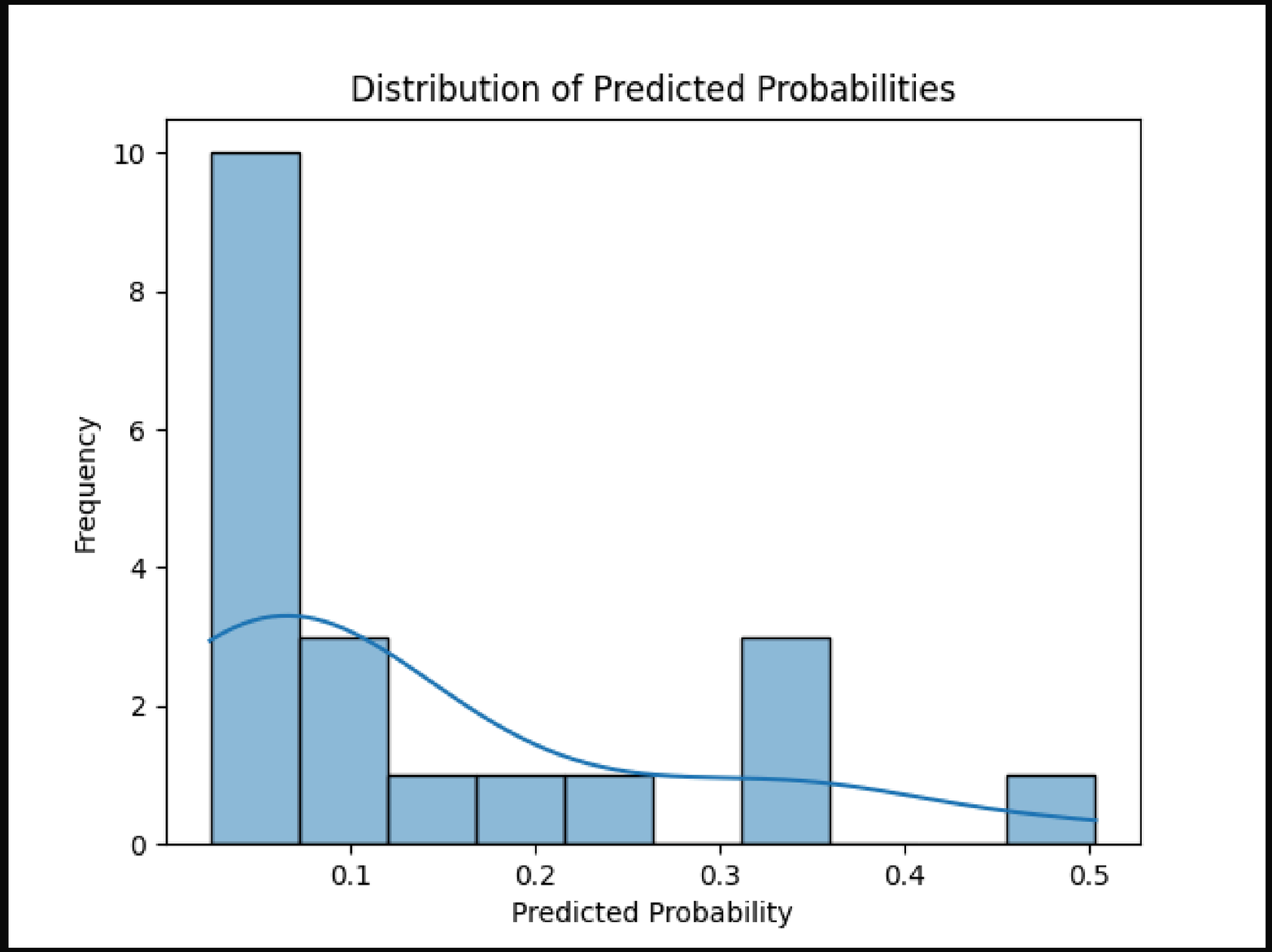
Mostra a distribuição de instâncias previstas em relação às esperadas.
Útil para visualizar quantas instâncias foram previstas corretamente/incorretamente.



Distribuição das Probabilidades Preditas

Exibe como as probabilidades preditas pelo modelo estão distribuídas.

Útil para entender a confiança do modelo nas suas previsões.



Métricas

	Precision	Recall	f1-score	support
0 - Seguro	0.84	1.00	0.91	16
1 - Vulnerável	1.00	0.25	0.40	4
Accuracy			0.85	20
Macro avg	0.92	0.62	0.66	20
Weighted avg	0.87	0.85	0.81	20

QUESTÕES DE PESQUISA?

Existe uma ou mais vulnerabilidades com 100% de eficácia de detecção pelo software? A que se deve esse fator?

Existe uma ou mais vulnerabilidades com 0% de eficácia de detecção pelo software? A que se deve esse fator?

OBRIGADO