Pedro Conrado Negreiro
Matheus Henrique Imberio
Victor Hugo Brito

Projeto final
Desenvolvimento de um sistema inteligente para classificação de dados

Relatório técnico do projeto final solicitado pelo professor Juliano Foleis na disciplina de Inteligência Computacional, Bacharelado em Ciência da Computação da Universidade Tecnológica Federal do Paraná.

Universidade Tecnológica Federal do Paraná – UTFPR
Departamento Acadêmico de Computação – DACOM
Bacharelado em Ciência da Computação – BCC

Campo Mourão
Fevereiro/2025

Sumário

1 Introdução

A segurança cibernética é um campo essencial na proteção de sistemas e redes contra acessos não autorizados, ataques maliciosos e vazamento de dados. Com o avanço das ameaças digitais, a detecção de intrusões tornou-se uma necessidade crítica para empresas e organizações, exigindo soluções eficientes e automatizadas.

Neste contexto, o uso de técnicas de Inteligência Artificial (IA) tem se destacado como uma abordagem promissora para identificar padrões suspeitos e distinguir atividades legítimas de possíveis ataques. Dentre os modelos avaliados, a Máquina de Vetores de Suporte (SVM - Support Vector Machine) demonstrou melhor desempenho na classificação de dados de segurança cibernética, superando outras abordagens como Redes Neurais Artificiais (RNA), Árvores de Decisão (Decision Tree) e K-Nearest Neighbors (KNN). Enquanto a RNA apresentou resultados competitivos, exigiu maior tempo de treinamento e ajustes finos para evitar overfitting. O Decision Tree, apesar de ser rápido e interpretável, mostrou-se mais suscetível a ruídos no conjunto de dados, reduzindo sua precisão. Já o KNN teve um desempenho inferior devido ao alto custo computacional para grandes volumes de dados e à sensibilidade a ruídos.

2 Fundamentação

A classificação de dados é uma das principais tarefas em aprendizado de máquina supervisionado, onde o objetivo é prever a classe de uma amostra com base em suas características. Esse processo envolve três etapas principais: a extração de características, a seleção e treinamento de um modelo de classificação e a avaliação do desempenho do modelo.

Extração de Características

A extração de características é uma etapa fundamental para garantir que as informações mais relevantes sejam capturadas dos dados brutos. Essa etapa pode ser realizada utilizando descritores manuais, como histogramas, estatísticas baseadas em textura ou técnicas baseadas em aprendizado profundo.

Modelos de Classificação

Os modelos de classificação supervisionada são construídos com base em dados rotulados e incluem algoritmos como k-Nearest Neighbors (k-NN), Árvores de Decisão, Support Vector Machines (SVM) e Perceptron Multicamadas (MLP). Cada um desses algoritmos possui vantagens e limitações, que podem ser mitigadas através da otimização de seus hiperparâmetros.

Validação e Métricas de Avaliação

A validação cruzada em k vias é amplamente utilizada para avaliar a capacidade de generalização de um modelo, dividindo os dados em k subconjuntos e alternando entre treino e teste. As métricas de avaliação incluem:

Acurácia: Proporção de predições corretas.

Precision: Capacidade de um modelo de evitar falsos positivos.

Recall: Capacidade de identificar corretamente os exemplos de uma classe.

F1-Score: Média harmônica entre precision e recall.

3 Materiais

O desenvolvimento do sistema inteligente será realizado utilizando as seguintes ferramentas e tecnologias:

Linguagem de programação: Python, pela sua ampla biblioteca de ferramentas para aprendizado de máquina, como o scikit-learn.

Ambiente de desenvolvimento: Google Colab, devido à sua facilidade de uso e acesso a recursos computacionais em nuvem.

Conjunto de dados: Conjunto de dados de detecção de intrusão de segurança cibernética foi projetado para detectar intrusões cibernéticas com base no tráfego de rede e no comportamento do usuário.

Bibliotecas utilizadas:

scikit-learn: Para implementação dos modelos de classificação e extração de métricas de desempenho.

Pandas e NumPy: Para manipulação e análise de dados.

Matplotlib e Seaborn: Para visualização de resultados, incluindo matrizes de confusão.

A implementação seguirá as etapas descritas na fundamentação teórica, com validação cruzada em k vias (k>=5) para avaliar a robustez dos modelos. Além disso, os hiperparâmetros de cada classificador serão ajustados utilizando técnicas como grid search ou random search para maximizar o desempenho do sistema.

4 Procedimentos e Resultados

Neste trabalho, utilizamos diferentes modelos de aprendizado de máquina para realizar a detecção de intrusões em um conjunto de dados de segurança cibernética. Entre os modelos avaliados, testamos uma Rede Neural Artificial (RNA), especificamente o MLP-

Classifier, além de KNN (K-Nearest Neighbors), Árvore de Decisão (Decision Tree) e SVM (Support Vector Machine), com o objetivo de comparar seus desempenhos e identificar a abordagem mais eficaz.

Pré-processamento dos Dados:

Carregamento e Limpeza de Dados: O dataset foi carregado e as colunas foram ajustadas para remover espaços em branco no início e no final dos nomes.

Transformação das Variáveis Categóricas: Utilizamos o OneHotEncoder para transformar variáveis categóricas, como "protocol_type", "encryption_used"e "browser_type".

Divisão do Conjunto de Dados: O conjunto de dados foi dividido em três subconjuntos: treinamento (80%), teste (20%) e validação (20% dos 80% do treinamento). Essa divisão foi feita de forma estratificada para garantir que as distribuições de classes se mantivessem equilibradas. Normalização:

Escalonamento dos Dados: Utilizamos o StandardScaler para normalizar os dados, de modo que as características tenham média 0 e desvio padrão 1. O escalonamento foi aplicado separadamente nos conjuntos de treinamento, validação e teste para evitar vazamento de dados.

Treinamento e Avaliação do Modelo de Classificação:

Neste trabalho, foram treinados e avaliados quatro modelos de classificação para um conjunto de dados, com o objetivo de comparar seu desempenho. Os modelos avaliados foram: MLP (Multilayer Perceptron), KNN (K-Nearest Neighbors), Árvore de Decisão e SVM (Support Vector Machine).

MLP: Foi configurado com diferentes combinações de camadas ocultas e funções de ativação. O treinamento foi realizado por até 2000 iterações, com early stopping ativado (early_stopping=True), permitindo que o processo fosse interrompido antecipadamente caso a convergência fosse atingida antes do limite máximo de épocas. As configurações do modelo foram as seguintes: Foi testado com diferentes configurações de camadas ocultas, incluindo [(50,), (100, 50), (100, 100, 100)], e funções de ativação relu e tanh. Também experimentamos uma arquitetura mais profunda [(200, 200, 200, 200)] para avaliar possíveis melhorias no desempenho. No entanto, o ganho foi insignificante em relação ao aumento significativo no tempo de processamento, levando-nos a manter as configurações mais enxutas.

KNN: Número de vizinhos ([3, 5, 7]), métricas de distância ([Euclidiana, Manhattan]) e pesos ([Uniforme, Distância]).

Árvore de Decisão:Árvore de Decisão: Profundidade máxima ([None, 10, 20]) e mínimo de amostras para divisão ([2, 5, 10]).

SVM: Valor de C ([0.1, 1, 10]) e kernel ([Linear, RBF]).

A avaliação foi realizada utilizando validação cruzada estratificada de 5 dobras, para garantir uma distribuição equilibrada das classes. As métricas de desempenho utilizadas foram Acurácia, F1-Score, Precisão e Recall, sendo o F1-Score ponderado a principal métrica devido ao desbalanceamento de classes no conjunto de dados.

Os resultados foram analisados e comparados para determinar qual modelo apresentou o melhor desempenho.

Resultados usando diferentes modelos:

Seguem os hiperparâmetros que utilzei para melhores resultados de cada modelo utilizando validação cruzada em 5 vias:

```python
def get_model_and_params(model_type):
    if model_type == 'svm':
        model = SVC()
        param_grid = {
            'model__C': [0.1, 1, 10],
            'model__kernel': ['linear', 'rbf']
        }
    elif model_type == 'knn':
        model = KNeighborsClassifier()
        param_grid = {
            'model__n_neighbors': [3, 5, 7],
            'model__weights': ['uniform', 'distance'],
            'model__metric': ['euclidean', 'manhattan']
        }
    elif model_type == 'tree':
        model = DecisionTreeClassifier()
        param_grid = {
            'model__max_depth': [None, 10, 20],
            'model__min_samples_split': [2, 5, 10]
        }
    elif model_type == 'mlp':
        model = MLPClassifier(max_iter=2000, solver='adam', learning_rate_init=0.001, early_stopping=True)
        param_grid = {
           'model__hidden_layer_sizes': [(50,), (100, 50), (100, 100, 100)],
            'model__activation': ['relu', 'tanh']
        }
    else:
        raise ValueError("Modelo desconhecido")
    return model, param_grid
```

Figura 1 – Modelos e seus Hiperparâmetros

4.1 KNN

```
Melhores hiperparâmetros encontrados:
{'model__metric': 'manhattan', 'model__n_neighbors': 7, 'model__weights': 'distance'}
F1-score no conjunto de validação: 0.7736
F1-score no conjunto de teste: 0.7716
```

Figura 2 – Melhores hiperparâmetros

F1-score médio (cross-validation): 0.7747 ± 0.0087
Acurácia: 0.7584
Métricas de Classificação:

	precision	recall	f1-score	support
0	0.73	0.90	0.81	1055
1	0.83	0.58	0.68	853
accuracy			0.76	1908
macro avg	0.78	0.74	0.74	1908
weighted avg	0.77	0.76	0.75	1908

Figura 3 – Métricas

O F1-score médio (cross-validation) de 0.7747 ± 0.0087 representa a média dos F1-scores obtidos durante a validação cruzada, indicando a estabilidade do modelo. A acurácia de 75,84% mostra a proporção de previsões corretas, mas pode ser influenciada por um possível desbalanceamento do dataset. A precisão mede quantas das previsões positivas realmente pertencem àquela classe, sendo 73% para a classe 0 e 83% para a classe 1. O recall indica a taxa de recuperação dos exemplos reais de cada classe, com 90% para a classe 0 e 58% para a classe 1, sugerindo que o modelo identifica melhor a classe 0. O F1-score, que equilibra precisão e recall, é 0.81 para a classe 0 e 0.68 para a classe 1, mostrando que a classe 1 tem mais erros. A média macro calcula a média simples das métricas de cada classe, enquanto a média ponderada ajusta os valores conforme o número de exemplos de cada classe, refletindo melhor o desempenho geral. A baixa taxa de recall da classe 1 indica que muitos exemplos dessa categoria estão sendo mal classificados, o que pode ser um problema dependendo do contexto da aplicação.

Este parágrafo se aplica a todos os modelos, servindo como uma explicação das métricas utilizadas para uma melhor compreensão dos resultados.

Segue a matriz de confusão:

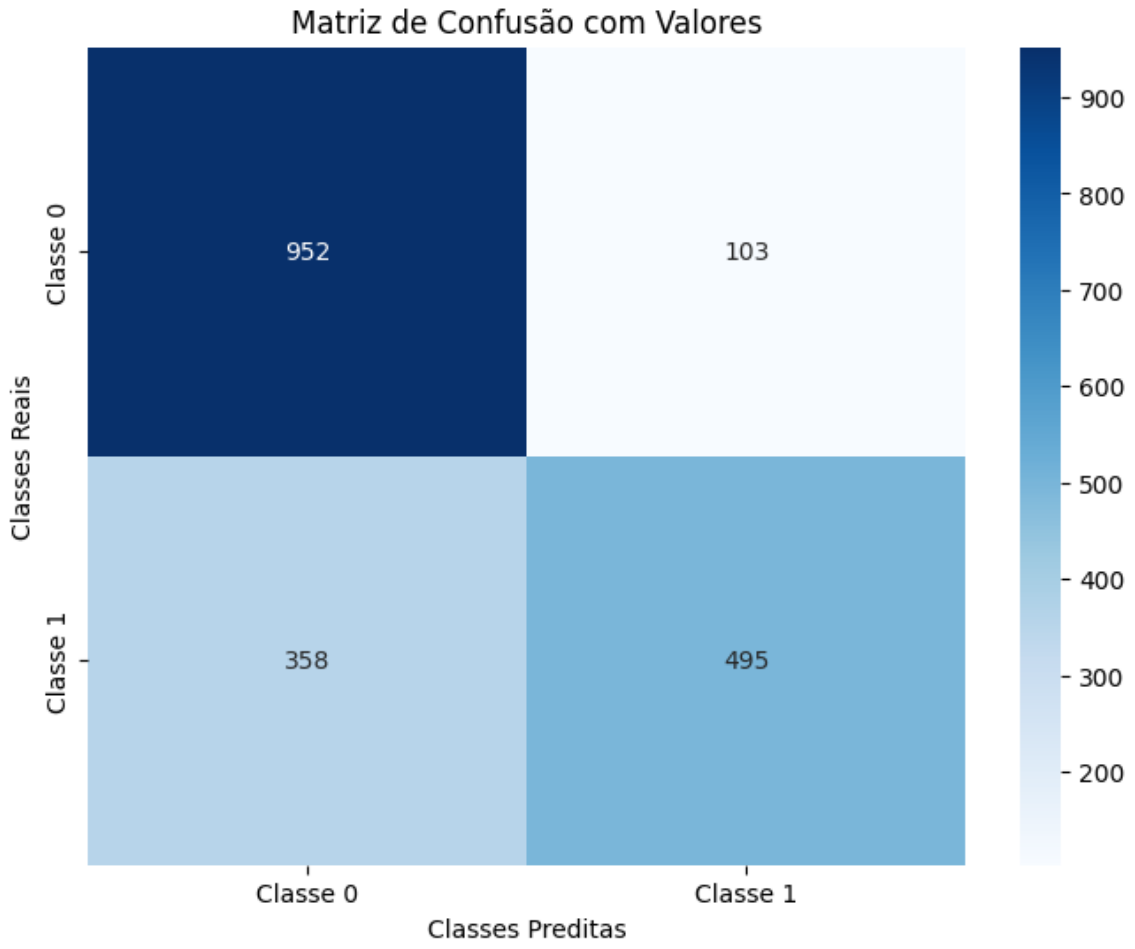

Figura 4 – Matriz de Confusão KNN

4.2 SVM

```
Melhores hiperparâmetros encontrados:
{'model__C': 10, 'model__kernel': 'rbf'}
F1-score no conjunto de validação: 0.8645
F1-score no conjunto de teste: 0.8607
```

Figura 5 – Melhores hiperparâmetros

```
F1-score médio (cross-validation): 0.8590 ± 0.0094
Acurácia: 0.8443
Métricas de Classificação:
              precision    recall  f1-score   support

           0       0.81      0.94      0.87      1055
           1       0.91      0.72      0.81       853

    accuracy                           0.84      1908
   macro avg       0.86      0.83      0.84      1908
weighted avg       0.85      0.84      0.84      1908
```

Figura 6 – Métricas

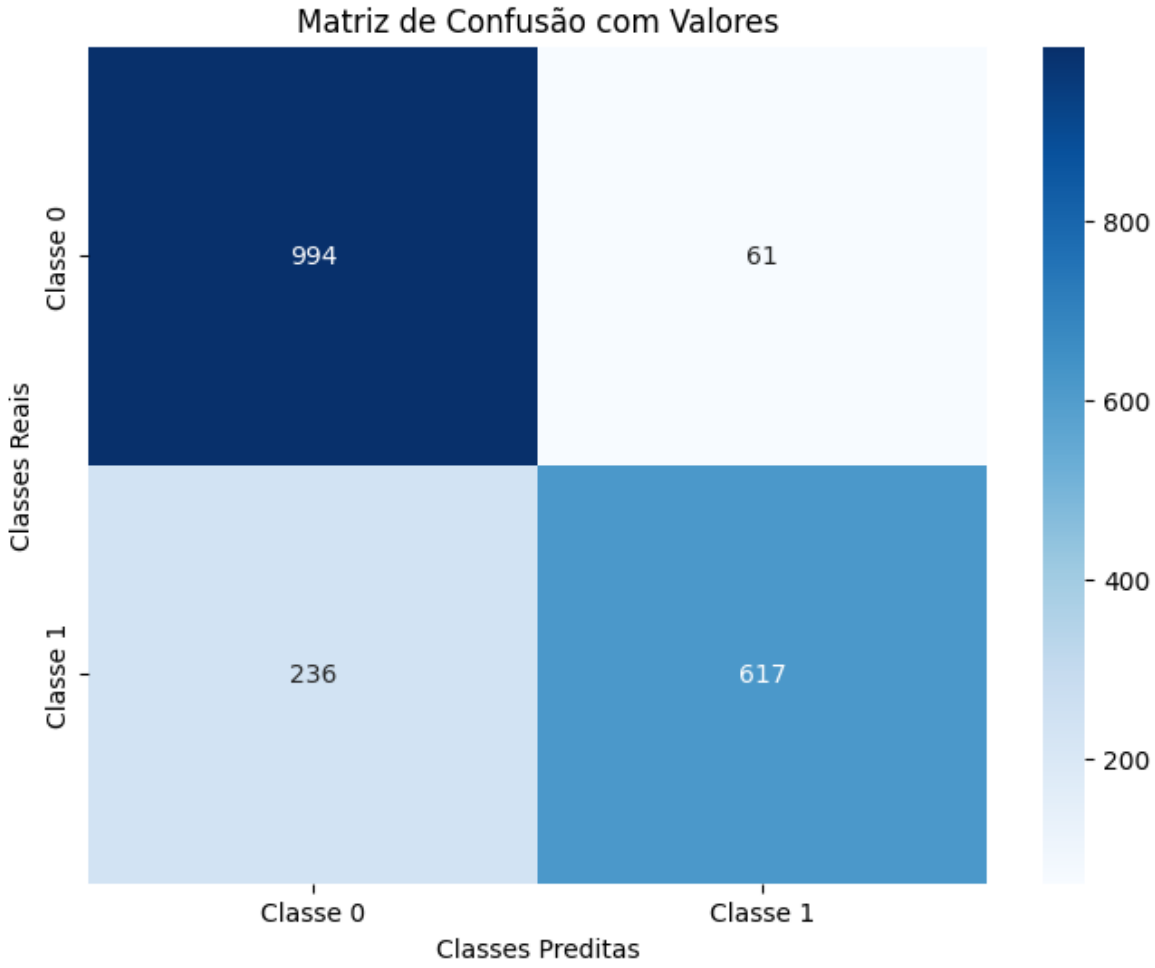

Figura 7 – Matriz de Confusão SVM

4.3 DECISION TREE

```
Melhores hiperparâmetros encontrados:
{'model__max_depth': 10, 'model__min_samples_split': 10}
F1-score no conjunto de validação: 0.8850
F1-score no conjunto de teste: 0.8818
```

Figura 8 – Melhores hiperparâmetros

```
F1-score médio (cross-validation): 0.8161 ± 0.0062
Acurácia: 0.8056
Métricas de Classificação:
              precision    recall  f1-score   support

           0       0.84      0.80      0.82      1055
           1       0.77      0.81      0.79       853

    accuracy                           0.81      1908
   macro avg       0.80      0.81      0.80      1908
weighted avg       0.81      0.81      0.81      1908
```

Figura 9 – Métricas

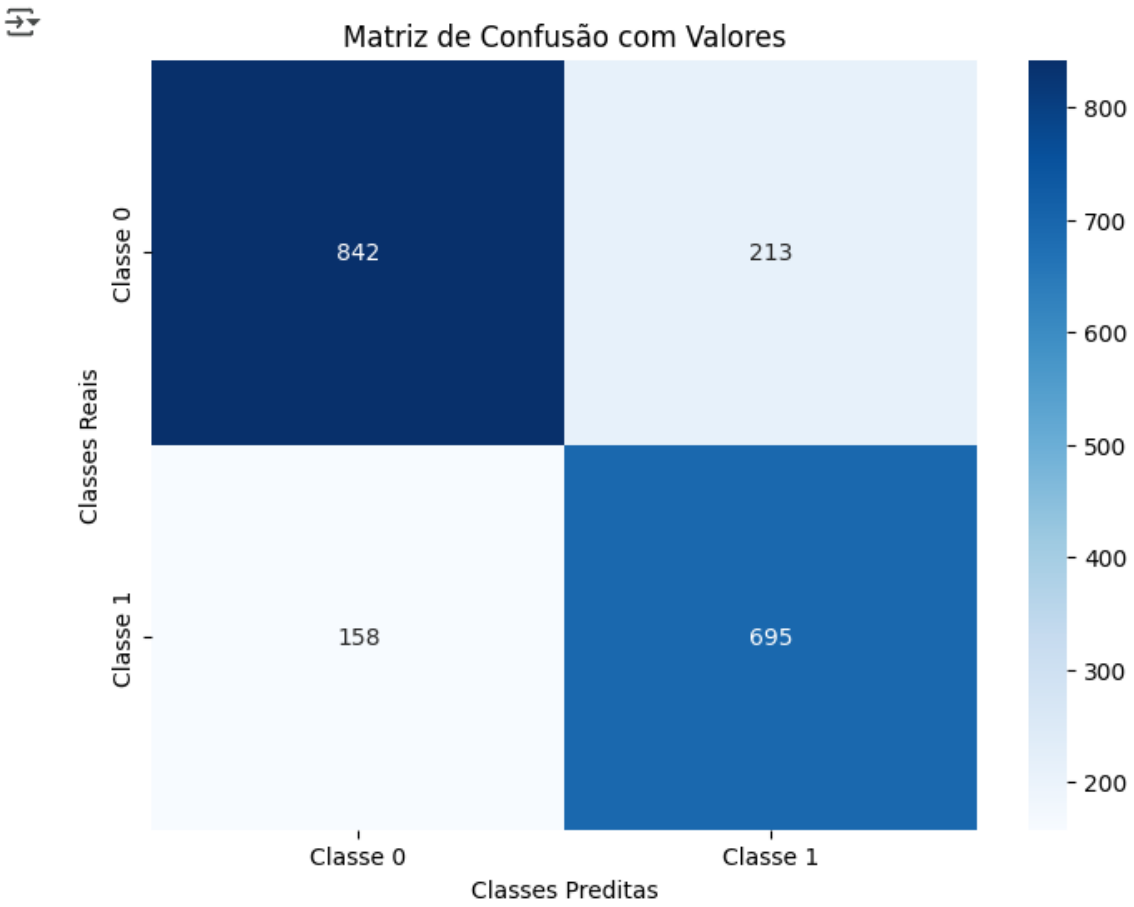

Figura 10 – Matriz de Confusão DECISION TREE

4.4 RNA

```
Melhores hiperparâmetros encontrados:
{'model__activation': 'relu', 'model__hidden_layer_sizes': (100, 50)}
F1-score no conjunto de validação: 0.8743
F1-score no conjunto de teste: 0.8747
```

Figura 11 – Melhores hiperparâmetros

```
F1-score médio (cross-validation): 0.8742 ± 0.0083
Acurácia: 0.8669
Métricas de Classificação:
              precision    recall  f1-score   support

           0       0.83      0.96      0.89      1055
           1       0.94      0.75      0.83       853

    accuracy                           0.87      1908
   macro avg       0.88      0.86      0.86      1908
weighted avg       0.88      0.87      0.86      1908
```

Figura 12 – Métricas

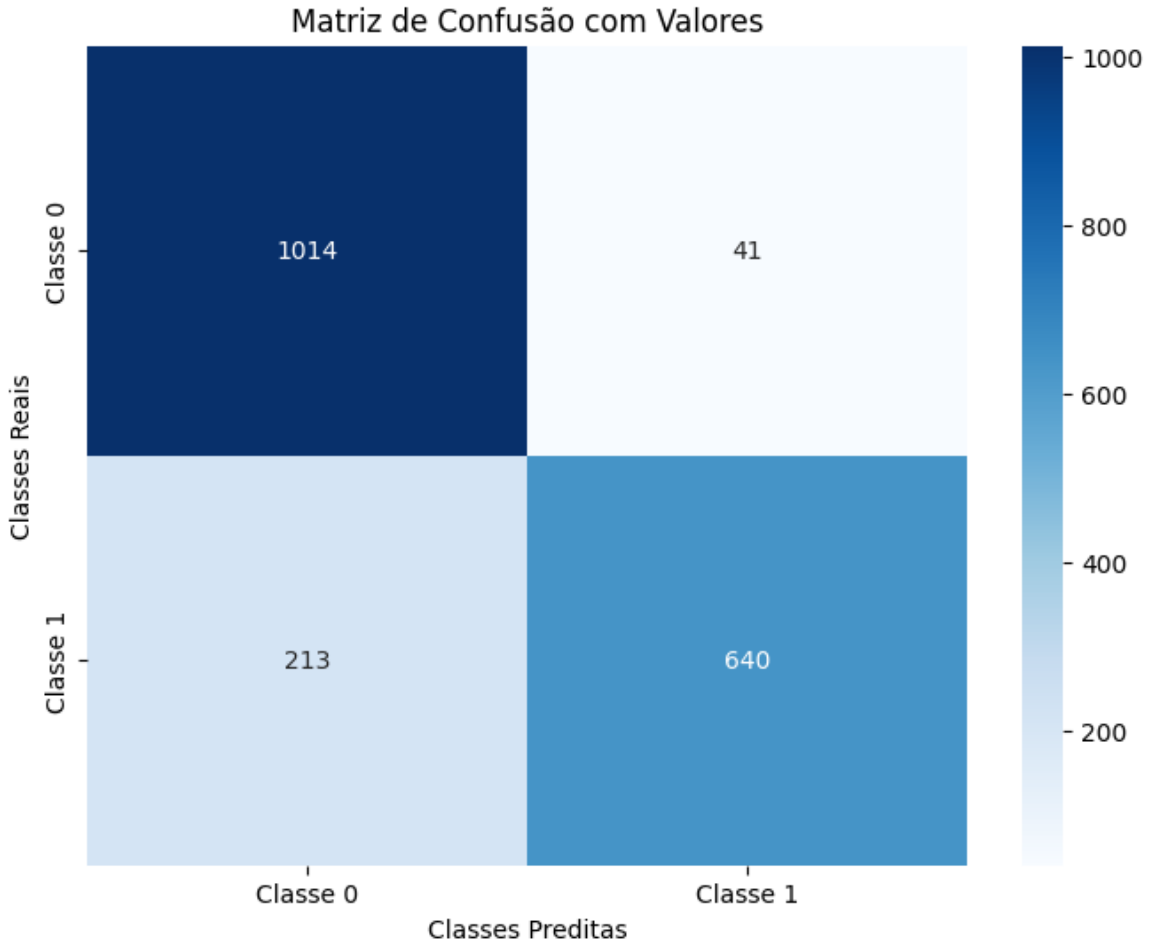

Figura 13 – Matriz de Confusão RNA

Essas métricas indicam que o modelo apresenta um bom desempenho na tarefa de detecção de intrusões, com alta capacidade de classificar corretamente as instâncias, mesmo diante de um conjunto de dados desbalanceado.

5 Discussão dos Resultados

Análise Comparativa dos Modelos:

KNN (K-Nearest Neighbors):

O modelo teve um melhor desempenho para a classe 0 (F1-score de 0.81) do que para a classe 1 (F1-score de 0.68), indicando dificuldades em classificar corretamente exemplos da classe 1. Esse comportamento pode ser reflexo de um possível desbalanceamento nos dados ou da baixa separabilidade dessa classe.

Em comparação com os demais modelos, o KNN teve o pior desempenho geral, o que pode estar relacionado à sua sensibilidade a ruídos e ao aumento da dimensionalidade dos dados.

SVM (Support Vector Machine):

O desempenho foi equilibrado entre as classes, com um F1-score de 0.87 para a classe 0 e 0.81 para a classe 1, demonstrando que a SVM conseguiu separar bem as classes e generalizar os dados.

Em comparação com o KNN, a SVM teve uma melhora significativa em todas as métricas, indicando que um classificador mais robusto pode ser mais eficaz para este conjunto de dados.

Decision Tree (Árvore de Decisão):

As métricas mostram um equilíbrio entre as classes, com um F1-score de 0.82 para a classe 0 e 0.79 para a classe 1. Isso indica que a árvore de decisão conseguiu capturar bem os padrões dos dados, mas seu desempenho ficou abaixo da SVM.

Embora a Decision Tree tenha ficado acima do KNN, ela pode sofrer com overfitting se não for bem regularizada.

RNA (Rede Neural Artificial):

O modelo RNA apresentou um desempenho sólido entre os modelos testados. Com os hiperparâmetros ótimos de ativação 'relu' e camadas ocultas de (100, 50), o modelo obteve um F1-score médio de 0.8742 ± 0.0083 e uma acurácia de 86.69%.

A principal vantagem do modelo RNA é sua capacidade de capturar padrões complexos nos dados. No entanto, o custo computacional é significativo, exigindo mais tempo e recursos para treinamento e ajuste em comparação com modelos mais simples.

6 Conclusões

Comparando os modelos:

KNN teve o pior desempenho, o que sugere que esse modelo não é adequado para este problema específico, possivelmente devido à alta dimensionalidade dos dados.

Decision Tree teve um desempenho intermediário, sendo uma alternativa interpretável, mas menos eficiente que a SVM.

SVM foi um dos melhores modelos, com um bom equilíbrio entre precisão e recall e sem a necessidade de tantos recursos computacionais.

RNA foi o modelo de melhor desempenho, com o maior F1-score e acurácia, indicando que conseguiu capturar padrões mais complexos nos dados.

Se o objetivo for maximizar a performance, a RNA deve ser a escolhida. Caso seja necessário um modelo mais interpretável e eficiente, a SVM pode ser uma alternativa viável.

7 Referências

7.1 Link do Dataset no kaggle

Dataset

https://www.kaggle.com/datasets/dnkumars/cybersecurity-intrusion-detection-dataset

7.2 Links e materiais disponibilizados pelo Professor

Extração de Características e Reconhecimento de Padrões

https://periodicos.furg.br/vetor/article/download/3363/3811

Extração e Seleção de Características para a Classificação Eficiente

https://repositorio.ufu.br/bitstream/123456789/35084/1/ExtracaoSelecaoCaracteristicas.pdf

Extração de Características em Imagens para Classificação

https://repositorio.ufc.br/bitstream/riufc/30005/1/2017$_d i s_j hnaquino.pdf$

Documentação Oficial do Scikit-learn

https://scikit-learn.org

Repositório do Scikit-learn no GitHub

https://github.com/scikit-learn/scikit-learn