1 2 90
UNIVERSIDADE D
COIMBRA

09 DE MAIO DE 2025

CHASING PHISHING URLS

RECONHECIMENTO DE PADRÕES

Ana Carolina Morais MEB Nº2021222056
Fernanda Fernandes MEI Nº2021216620

ÍNDICE

01 | INTRODUÇÃO

Atualmente, a segurança na '*internet*' tornou-se uma preocupação cada vez mais significativa, especialmente com o aumento dos ataques cibernéticos direcionados à obtenção de dados pessoais e financeiros.

Um dos métodos mais frequentes adotados por *cibercriminosos* é o *phishing*, que envolve a distribuição de *hiperligações* enganosas por meio de *e-mails*, mensagens de texto e plataformas de redes sociais para ludibriar os utilizadores e roubar informações sensíveis. Para enfrentar essa questão, a identificação automática de *urls* nocivas emergiu como um foco de interesse entre os profissionais de inteligência artificial e *Machine Learning*.

Neste estudo, analisamos técnicas de reconhecimento de padrões voltadas para a classificação de *urls* como seguras ou de *phishing*, utilizando um conjunto de dados abrangente que integra informações extraídas de urls e do código-fonte das páginas da '*web*' relacionadas. O código-fonte diz respeito aos componentes contidos no *HTML* e *JavaScript* das páginas acessíveis por meio das *urls*, englobando dados como a existência de formulários, redirecionamentos, e termos duvidosos conectados a serviços financeiros ou criptomoedas.

| OBJETIVO

O principal objetivo deste trabalho é desenvolver modelos de classificação capazes de identificar urls de phishing com alta precisão.

Para tal, serão exploradas diferentes técnicas de seleção e redução de dimensionalidade, bem como métodos de *Machine Learning* para a deteção de padrões. O estudo também visa analisar o impacto de cada atributo no desempenho dos modelos e comparar os resultados obtidos com trabalhos pré-existentes na literatura, nomeadamente o seguinte trabalho [1].

| DATA

Para este projeto, o *dataset* utilizado encontra-se disponível em UCI Machine Learning Repository [2]. O primeiro passo foi obter os dados e preparar a sua utilização. O conjunto de dados inclui um total de 235.795 urls, sendo 134.850 classificadas como legítimas e 100.945 como *phishing*. As features extraídas podem ser organizadas em diferentes grupos:

- Informações sobre a estrutura da URL:
  - URLLength: Comprimento do URL
  - IsDomainIP: Indica se o domínio é um endereço IP
  - NoOfSubDomain: Número de subdomínios presentes na URL
  - CharContinuationRate: Taxa de continuação de caracteres na URL

- Características do domínio:
    - TLD: Tipo de domínio de topo
    - TLDLegitimateProb: Probabilidade de legitimidade do TLD
    - DomainLength: Comprimento do domínio
    - DomainTitleMatchScore: Correspondência entre domínio e o título da página
- Análise do código-fonte da página associada:
    - HasTitle: Indica se a página possui título
    - HasFavicon: Indica a presença de um favicon
    - HasDescription: Indica se há meta-descrição na página
    - HasExternalFormSubmit: Verifica se existem formulários que enviam dados para domínios externos
    - HasSubmitButton: Presença de botões de envio na página
- Indicadores de segurança:
    - IsHTTPS: Indica se a URL utiliza HTTPS
    - NoOfURLRedirect: Número de redirecionamentos na página
    - Bank: Presença de termos relacionados a bancos
    - Pay: Uso de palavras associadas a pagamentos
    - Crypto: Referências a criptomoedas

É de realçar que as *features* apresentadas acima são somente uma breve amostra, dado que existem 54 *features* e para facilitar iremos utilizar apenas as *features* não categóricas.

A variável alvo do *dataset* é denominada *label*, onde 0 representa urls legítimas e 1 indica urls de *phishing*. Antes de proceder com a análise dos dados, foram realizadas verificações para garantir a integridade do conjunto de dados, como a ausência de valores em falta. Para melhorar a eficiência dos modelos, serão aplicadas técnicas de redução de dimensionalidade e seleção de *features*, garantindo que somente as características mais relevantes sejam utilizadas no treino dos classificadores.

02 | PRÉ- PROCESSAMENTO

O pré-processamento dos dados é uma etapa fundamental na construção de modelos de *Machine Learning*. Inicialmente, os dados foram carregados e analisados para identificar características estatísticas e possíveis problemas, como valores nulos. A variável alvo foi renomeada de '*label*' para '*target*' para facilitar a compreensão ao longo do código.

As variáveis categóricas foram removidas, uma vez que os modelos utilizados exigem variáveis numéricas. As variáveis binárias, exceto a variável alvo, também foram excluídas para evitar redundância e reduzir a dimensionalidade do conjunto de dados. Neste contexto, muitas variáveis binárias podem representar informações pouco diferenciadas ou redundantes, não contribuindo significativamente para o desempenho do modelo. Além disso, a presença excessiva de variáveis binárias pode levar a um aumento do ruído nos dados, dificultando a aprendizagem dos padrões relevantes pelo modelo.

A normalização foi realizada através do método *Z-score (StandardScaler)*, garantindo que todas as variáveis tivessem média zero e desvio padrão unitário, melhorando a estabilidade e o desempenho dos algoritmos.

Realizámos, também, a divisão do conjunto de dados em treino (80%) e teste (20%), utilizando a função *train_test_split* com estratificação para manter a proporção das classes em ambos os subconjuntos. Os índices dos conjuntos foram resetados para evitar desalinhamentos durante o processamento, e cópias dos dados originais foram armazenadas para referência e comparações futuras, garantindo assim uma base sólida para a construção e avaliação dos modelos de classificação.

A distribuição das classes foi analisada (Fig.1), e o gráfico demonstra uma divisão relativamente equilibrada entre as categorias "Legítimo" (57,2%) e "*Phishing*" (42,8%). Esse equilíbrio minimiza problemas de viés no modelo e reduz a necessidade de técnicas avançadas de balanceamento de dados.

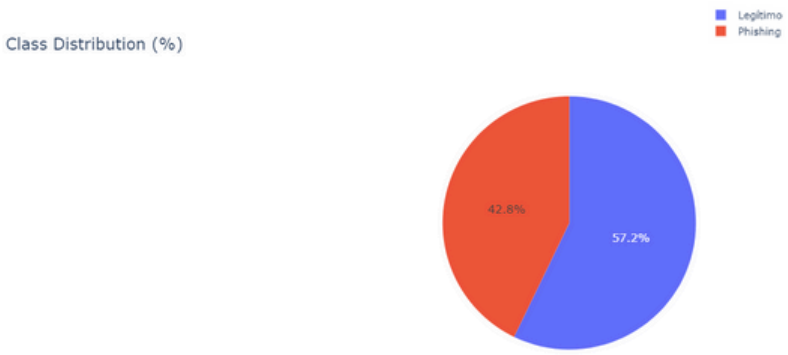

Fig. 1: Distribuição das classes (%)

02 | PRÉ-PROCESSAMENTO

Para lidar com a *multicolinearidade*, foi analisada a matriz de correlação e eliminadas as variáveis altamente correlacionadas (limite > 0.9). Essa remoção foi essencial para reduzir a redundância e evitar problemas de colinearidade, garantindo que o modelo aprenda padrões significativos sem distorções. Com esta abordagem conseguimos excluir mais 2 *features*, tendo agora 29 *features* no total sem contar com a variável alvo.

Além disso, foi gerado um *box plot* para todas as variáveis (Fig 2), permitindo visualizar a distribuição dos dados e identificar potenciais outliers. Esta análise contribui para uma melhor compreensão das relações entre as variáveis e auxilia na identificação de possíveis ajustes adicionais no pré-processamento.

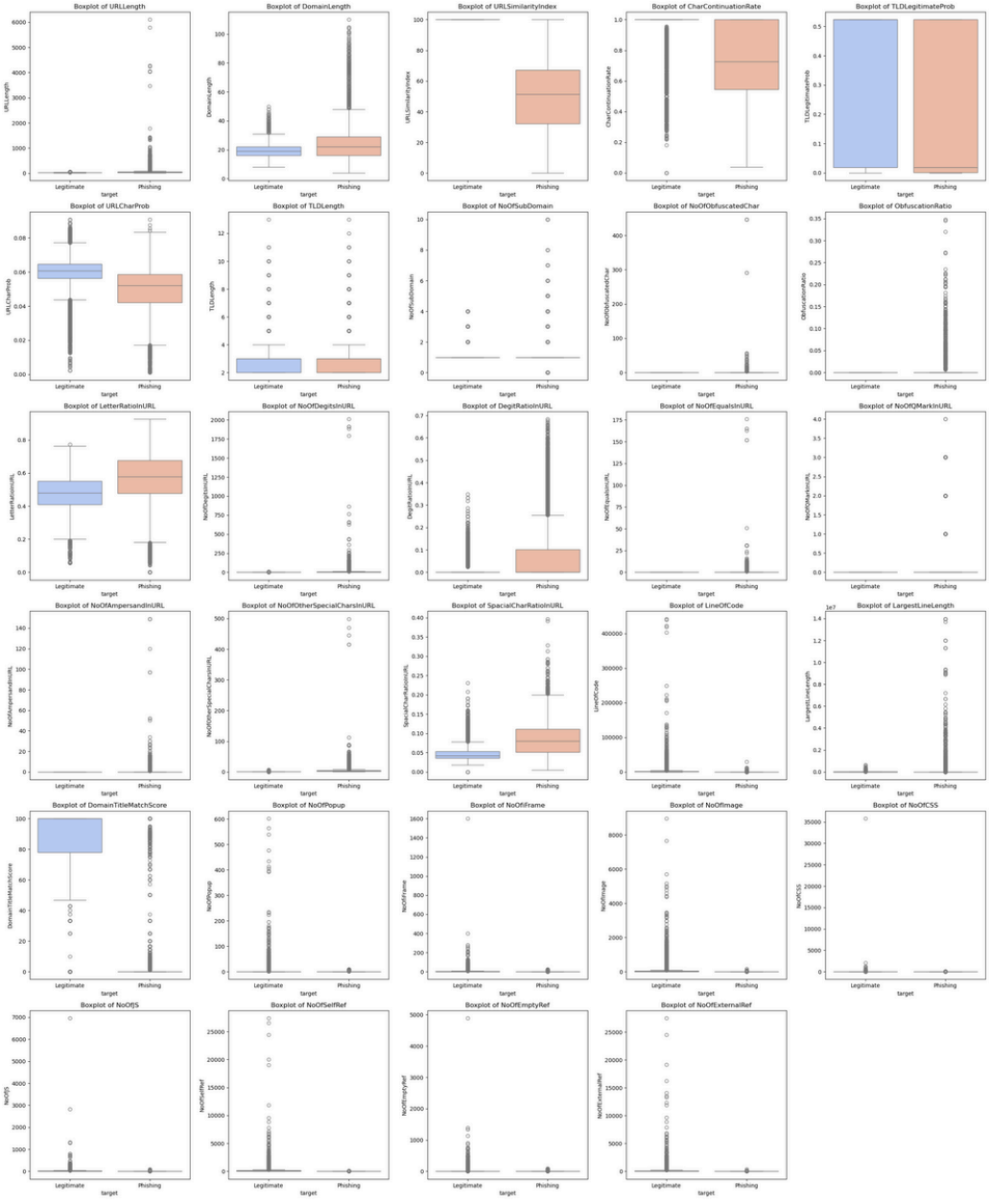

Fig. 2: Box Plot das Features

03 | SELEÇÃO DE FEATURES

Apesar de algumas *features* já terem sido eliminadas, ainda existe uma quantidade considerável de variáveis no conjunto de dados. Portanto, serão realizados testes estatísticos para uma análise detalhada da importância das características que permanecem. Serão utilizados métodos como o Teste *Kolmogorov-Smirnov* e o Teste *Kruskal-Wallis* para determinar quais variáveis continuam a ser relevantes para o modelo, assegurando que somente os atributos úteis sejam preservados.

- **Teste *Kolmogorov-Smirnov* (KS)**

O teste KS avalia a discrepância entre a função de distribuição cumulativa empírica de uma amostra de dados e a função de distribuição cumulativa teórica de uma distribuição padrão normal. Primeiro, o teste foi realizado no conjunto de dados completo para determinar se as *features* seguem uma distribuição normal. Depois, foi aplicado separadamente para cada classe (*phishing* e legítimo) para verificar se as distribuições variam entre elas. Se houver uma diferença grande entre classes, a *feature* pode ser relevante para classificação.

- **Resultados**

Os resultados indicam que todas as *features* seguem uma distribuição não normal, tanto para o conjunto de dados completo quanto para cada classe separadamente. O teste KS fornece *p-valores* que testam a hipótese de normalidade, e como todos foram inferiores ao nível de significância adotado ($\alpha = 0.05$), rejeitamos a hipótese nula, confirmando que os dados não seguem uma distribuição normal.

Os gráficos CDF vs ECDF e PDF vs EPDF auxiliam nesta análise, permitindo visualizar a correspondência entre a distribuição empírica dos dados e a distribuição normal teórica. No entanto, é importante destacar que a visualização pode sugerir uma boa correspondência, mas o teste KS é altamente sensível a pequenas diferenças, especialmente em grandes amostras, podendo detetar discrepâncias subtis que não são evidentes nos gráficos. Como na Fig. 3, em que para a *feature LetterRatioInUrl,* parece que segue uma distribuição normal, mas o teste revela que não. Isso também justifica a escolha dos testes estatísticos apropriados: como os dados não seguem uma distribuição normal, testes paramétricos como o ANOVA não são adequados. Em vez disso, é necessário recorrer a testes não paramétricos, como o *Kruskal-Wallis*, que não assumem normalidade e são mais apropriados para comparar distribuições entre grupos.

03 | SELEÇÃO DE FEATURES

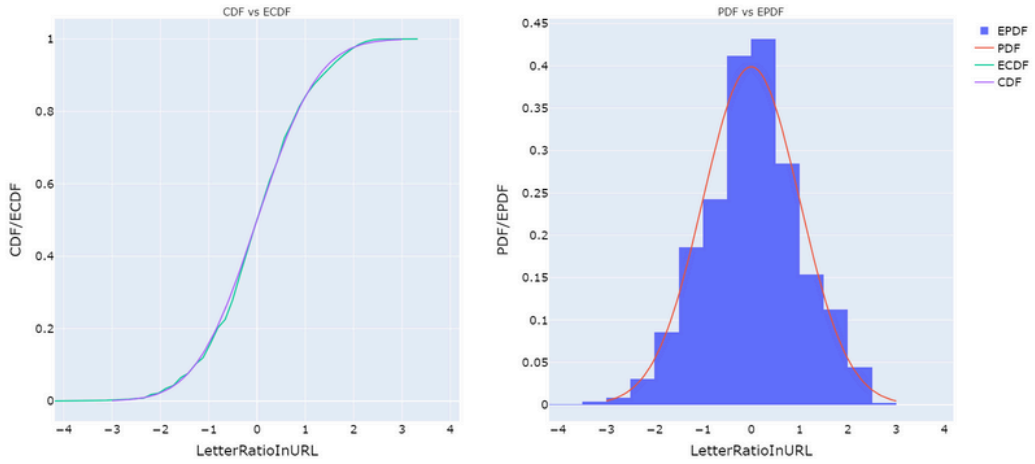

Fig. 3. Comparação entre as distribuições empíricas e teóricas da *feature LetterRatioInURL*

- **Teste *Kruskal-Wallis* (KW)**

O teste KW foi aplicado para avaliar a diferença na distribuição das *features* entre *urls legítimas* e *urls de phishing*. Este teste estatístico não paramétrico permite determinar a relevância das *features* para a classificação, atribuindo um valor de '*rank*' a cada uma delas. *Features* com valores de '*rank*' mais altos indicam uma maior diferença estatística entre as classes, sugerindo um maior poder discriminativo.

- **Resultados**

Como esperado, algumas *features* apresentam uma grande diferença estatística entre as duas classes, indicando um elevado poder discriminativo. Essas *features* obtiveram valores de '*rank*' mais altos, tornando-se candidatas fortes para a fase final da modelagem.

No gráfico de dispersão (Fig. 4), é possível observar que as *features* estão ordenadas conforme o seu valor de '*rank*'. Para facilitar a seleção, foi estabelecido um limite de 25.000 (linha vermelha), abaixo do qual as *features* foram removidas. Esta decisão baseia-se na ideia de que valores de *rank* muito baixos indicam que a *feature* tem pouca ou nenhuma influência na separação das classes. Manter tais features poderia adicionar ruído ao modelo e aumentar a sua complexidade sem benefícios significativos.

03 | SELEÇÃO DE FEATURES

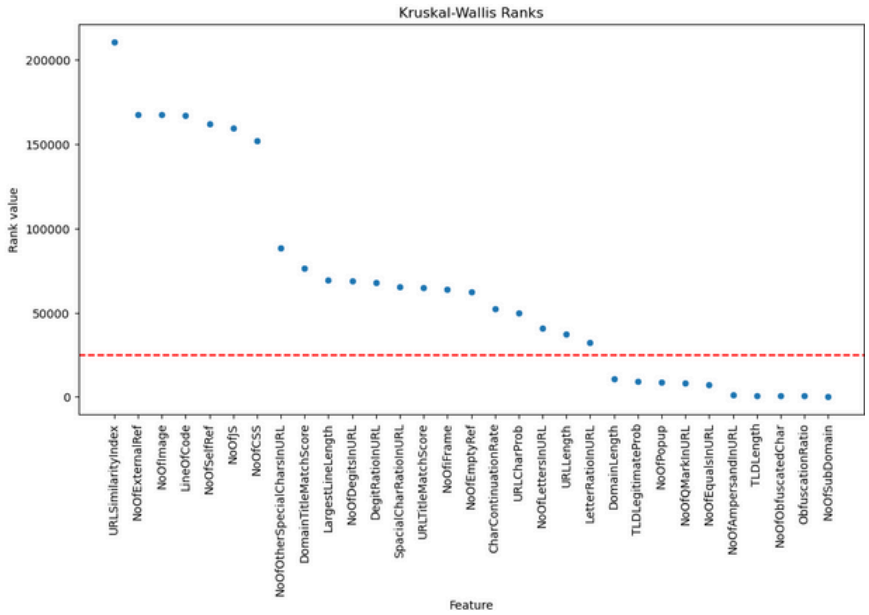

Fig. 4. Kruskal - Walls Ranks

Um aspeto importante a destacar é que, apesar de o teste KW permitir avaliar a relevância estatística de cada *feature* individualmente, ele não considera a redundância entre elas. No entanto, dado que já foram aplicadas técnicas como a análise da matriz de correlação e a exclusão de variáveis binárias para mitigar esse problema, este risco já foi amplamente reduzido. Ainda assim, é sempre recomendável validar continuamente a qualidade do conjunto de *features* escolhido, garantindo que cada uma contribui significativamente para o desempenho do modelo.

- **Curva de ROC**

A Curva ROC é uma ferramenta usada para avaliar o desempenho de modelos de classificação binária. Ela mostra a relação entre a taxa de verdadeiros positivos e a taxa de falsos positivos, permitindo perceber o quão bem o modelo consegue separar as duas classes. A área sob a curva ROC, conhecida como AUC, é um valor entre 0 e 1 que resume essa capacidade de distinção: quanto mais próximo de 1, melhor o modelo; um valor de 0.5 indica um desempenho semelhante ao de um modelo aleatório. A AUC pode também ser interpretada como a probabilidade de o modelo atribuir uma pontuação mais alta a uma amostra positiva do que a uma negativa. Esta métrica é especialmente útil para comparar modelos diferentes ou para avaliar o mesmo modelo com vários limiares de decisão.

- **Resultados**

A seleção de *features* utilizando a área sob a curva ROC (AUC-ROC) foi implementada avaliando cada característica individualmente quanto à sua capacidade discriminativa entre *urls* legítimas e de *phishing* (Fig. 5). Este método classificou *features* como URLSimilarityIndex, LineOfCode e NoOfExternalRef com os maiores valores de AUC (acima de 0.95), indicando seu elevado poder preditivo quando consideradas isoladamente.

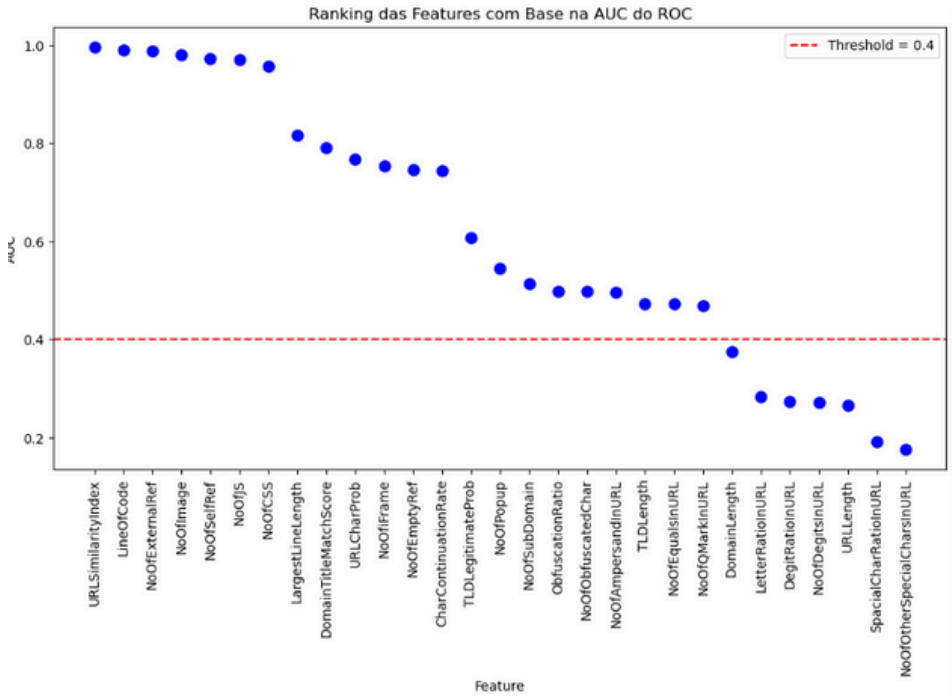

Fig. 5. Auc - Roc Rank

Ao comparar este *ranking* com os resultados obtidos pelo teste de *Kruskal-Wallis*, identificamos uma substancial sobreposição, com 16 características comuns entre as 19 *features* de maior relevância em cada método. Esta significativa concordância entre abordagens estatísticas distintas fortaleceu nossa confiança na robustez destas características, levando-nos a adotar as 19 *features* inicialmente identificadas pelo ranking de *Kruskal-Wallis* como base para nosso modelo de classificação, uma decisão respaldada pela validação cruzada proporcionada pela análise ROC.

- **Minimum Redundancy Maximum Relevance (mRMR)**

O *mRMR* é um classificador que seleciona as *features* mais relevantes para a tarefa de classificação, enquanto minimiza a redundância entre elas. Calcula a informação mútua entre cada *feature* e a variável alvo, atribuindo um valor que reflete o quão dependente cada atributo está da classe. A partir disso, o *mRMR* seleciona as k melhores *features* que apresentam a maior relevância, ao mesmo tempo que reduz a sobreposição de informação entre as variáveis selecionadas. No nosso caso, utilizamos o *mRMR* com informação mútua como critério de seleção, e a partir disso, obtemos um ranking das *features* mais importantes para a classificação. A vantagem do *mRMR* é a sua capacidade de melhorar a performance do classificador ao focar se nas variáveis mais informativas, mas a sua eficácia pode ser comprometida se a relação entre as *features* e a classe alvo for não linear ou muito complexa.

- **Resultados**

Analisando o ranking baseado no *mRMR* (Fig. 6), observamos que URLSimilarityIndex e LineOfCode se destacam novamente com os valores mais altos (acima de 0.6), seguidos por NoOfExternalRef e NoOfImage com pontuações superiores a 0.5. Nota-se uma clara separação entre as primeiras 10 *features* e as demais, com uma queda acentuada nos valores de *score* após a *feature* NoOfOtherSpecialCharsInURL. Ao comparar os três métodos de seleção de *features* (*Kruskal-Wallis*, AUC-ROC e *mRMR*), identificamos uma consistência notável nas características identificadas como mais relevantes, com aproximadamente 14 *features* em comum entre os três rankings. Esta convergência metodológica reforça nossa confiança na robustez da seleção inicial baseada no teste de *Kruskal-Wallis*, confirmando que as 19 *features* escolhidas por este método representam efetivamente os atributos mais discriminativos para a identificação de *urls* de *phishing*, com validação complementar fornecida pelos outros dois métodos independentes.

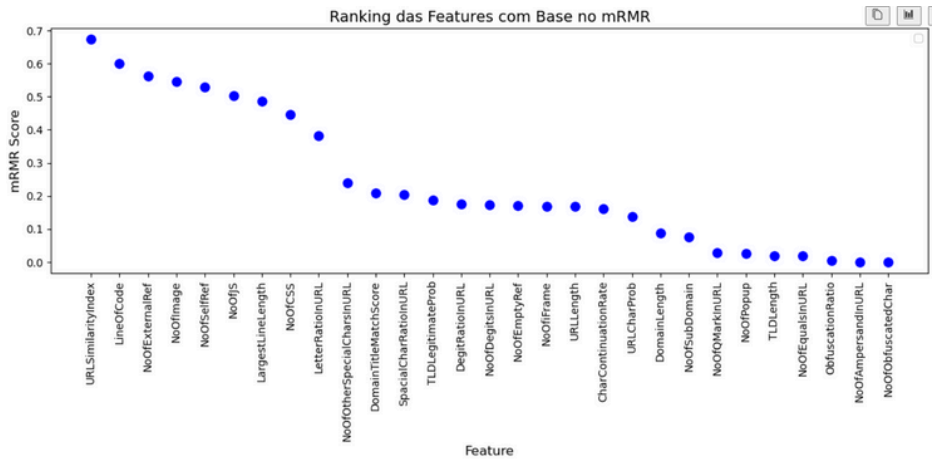

Fig. 6. Rank com base no mRMR - Informação Mútua

04 | REDUÇÃO DE DIMENSIONALIDADE

A redução de dimensionalidade é uma etapa essencial no processamento de dados, permitindo diminuir a complexidade do modelo, reduzir o ruído e melhorar a eficiência computacional. Duas abordagens comuns para este fim são a Análise de Componentes Principais (PCA) e a Análise Discriminante Linear (LDA).

- **Análise de Componentes Principais (PCA)**

O PCA é uma técnica estatística utilizada para transformar um conjunto de variáveis possivelmente correlacionadas num novo conjunto de variáveis não correlacionadas, denominadas componentes principais. Estas componentes são ordenadas para que as primeiras retêm a maioria da variabilidade dos dados.

- **Resultados**

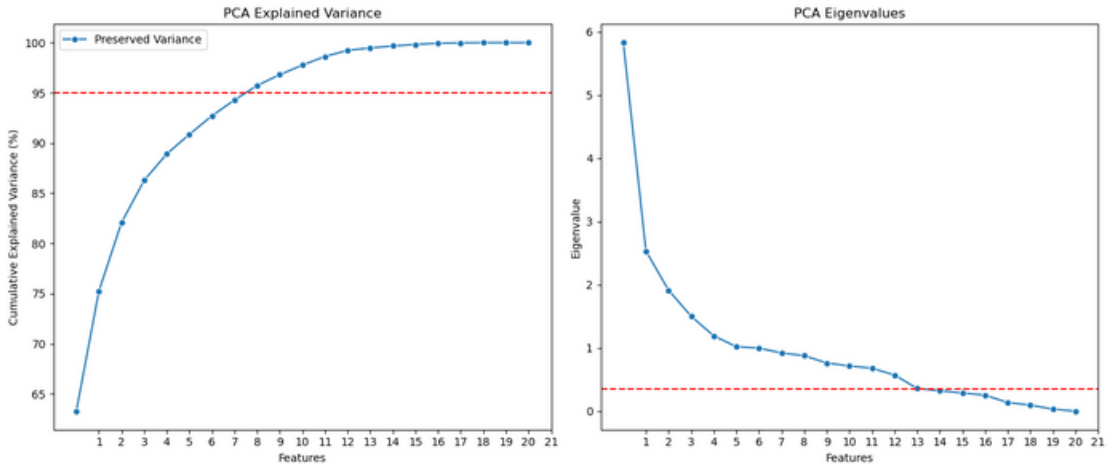

Fig. 7. A - Variância Explicada
B - Gráfio de Gráfico dos Valores Próprios

O primeiro gráfico (Fig. 7 - A) apresenta a variância explicada acumulada em função do número de componentes principais. Observa-se um crescimento rápido da variância explicada até aproximadamente a 10.ª componente, onde a curva começa a estabilizar-se. A linha vermelha representa o limiar de 95% de variância explicada, um critério comum para a seleção do número ideal de componentes. Neste caso, vemos que cerca de 10 componentes já explicam mais de 95% da variância total, tornando viável reduzir o número de dimensões sem perda significativa de informação.

O segundo gráfico exibe (Fig. 7 - B) os valores próprios (*eigenvalues*) das componentes principais. O *critério de Kaiser* sugere a retenção de componentes com *eigenvalue* superior a 1, pois estas carregam mais informação do que uma variável original típica. Além disso, aplicamos o *teste Scree*, que consiste em identificar o ponto onde o gráfico de *eigenvalues* estabiliza, indicando um decréscimo menos acentuado na variância explicada. Observamos que a estabilização ocorre por volta da 13.ª componente, tornando este um ponto adequado para seleção. Assim, optou-se por reter 13 componentes em vez de somente 10, garantindo um equilíbrio entre redução de dimensionalidade e preservação da informação.

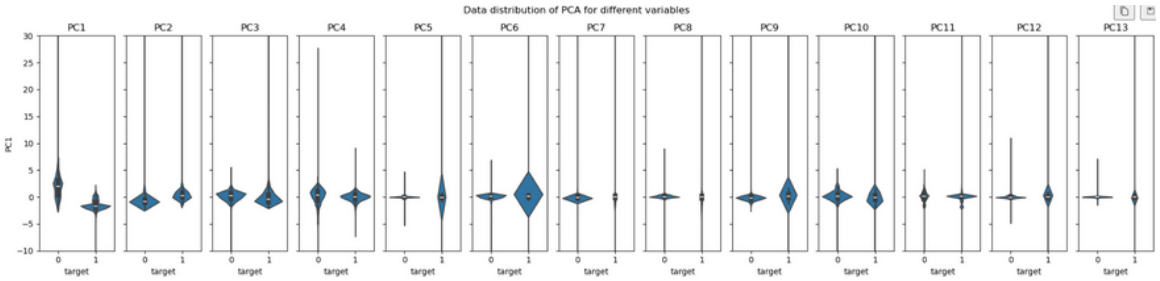

Fig. 8. - Distribuição das Componentes Principais

A Fig. 8 apresenta a distribuição das componentes principais relativamente à variável alvo. A variabilidade e separabilidade das classes diferem entre componentes. Algumas componentes, como a PC1, mostram uma clara distinção entre as classes, indicando serem mais relevantes para a modelagem preditiva. No entanto, como era de imaginar, continua a ser difícil encontrar uma solução para separar os dados linearmente. Componentes posteriores tendem a apresentar distribuições mais sobrepostas, sugerindo que a informação contida nelas não contribui significativamente para a diferenciação das classes.

- **Análise Discriminante Linear (LDA)**

A LDA é uma técnica supervisionada que projeta os dados num espaço de menor dimensão, maximizando a separabilidade entre classes. Ao contrário do PCA, que busca preservar a variância dos dados, a LDA foca na maximização da diferença entre classes.

- **Resultados**

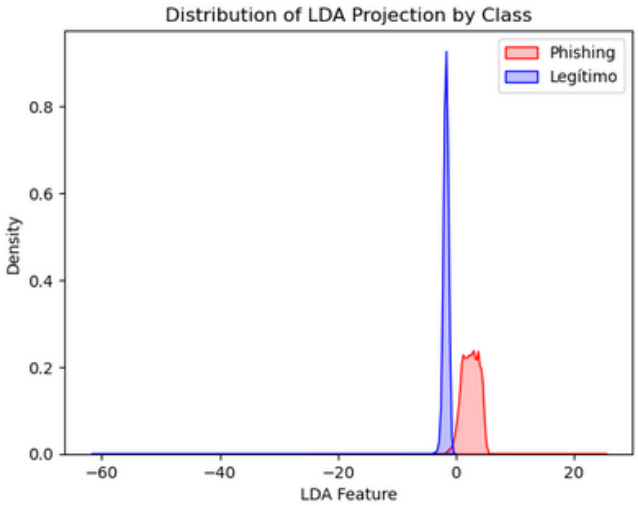

Fig. 9 - Distribuição da Projeção LDA

O gráfico (Fig. 9) resultante da projeção LDA mostra a distribuição das amostras segundo a nova variável discriminante. Nota-se que as classes "*Phishing*" e "Legítimo" apresentam distribuições distintas, indicando que a LDA conseguiu encontrar uma projeção eficaz para separá-las. A classe "Legítimo" apresenta uma distribuição mais concentrada, enquanto a classe "*Phishing*" possui maior dispersão. Isso sugere que a LDA conseguiu capturar padrões discriminantes relevantes, melhorando potencialmente o desempenho do modelo preditivo.

05 | CLASSIFICADORES

Nesta secção, aplicamos seis métodos de classificação: o Classificador de Distância Mínima (MDC) e o Fisher LDA, *Bayes Classification*, *Suport Vector Machin*e, *K- Nearest Neighbors* e *Random Forest*.

Para garantir uma avaliação mais robusta e confiável destes classificadores, utilizamos a técnica de validação cruzada estratificada (*Stratified K-Fold Cross-Validation*) durante o treinamento. Esta abordagem permite que o modelo seja treinado e validado múltiplas vezes, com diferentes subdivisões do conjunto de treino, mantendo a proporção original das classes em cada *fold*. As métricas analisadas incluem *accuracy, sensitivity e specificity*. Além disso, calculamos e representamos a matriz de confusão média +/- o desvio padrão das execuções de treino, bem como do conjunto de treino para observar o comportamento geral dos classificadores.

- **Classificador de Distância Mínima (MDC)**

O MDC é um método que atribui cada amostra à classe cujo centroide (média das amostras de treino dessa classe) estiver mais próximo segundo uma determinada métrica de distância. No nosso caso, consideramos tanto a distância euclidiana como a distância de Mahalanobis. A vantagem do MDC é a sua simplicidade computacional, mas pode não ser tão eficaz quando as classes não são linearmente separáveis.

- **Fisher LDA**

Este classificador não tem nada de novo para além do que já falamos até agora, simplesmente usamos a LDA para reduzir as variáveis e aplicamos o MDC.

- **Resultados**

Para avaliar os impactos da redução de dimensionalidade, consideramos quatro abordagens distintas:

1. Dados S/ Redução de Dimensionalidade (dados de treino originais) + MDC
2. Dados S/ Redução de Dimensionalidade (dados de treino originais) + Fisher LDA
3. Dados S/ Redução de Dimensionalidade (Mas com KW) + MDC
4. Dados S/ Redução de Dimensionalidade (Mas com KW) + Fisher LDA
5. MDC + PCA
6. PCA + FisherLDA

Esta abordagem permite verificar se a redução de dimensionalidade melhora ou prejudica a capacidade discriminativa dos classificadores e avaliar a eficácia do Fisher LDA quando aplicado diretamente aos dados originais.

Cenário 1: Dados S/ Redução de Dimensionalidade + MDC

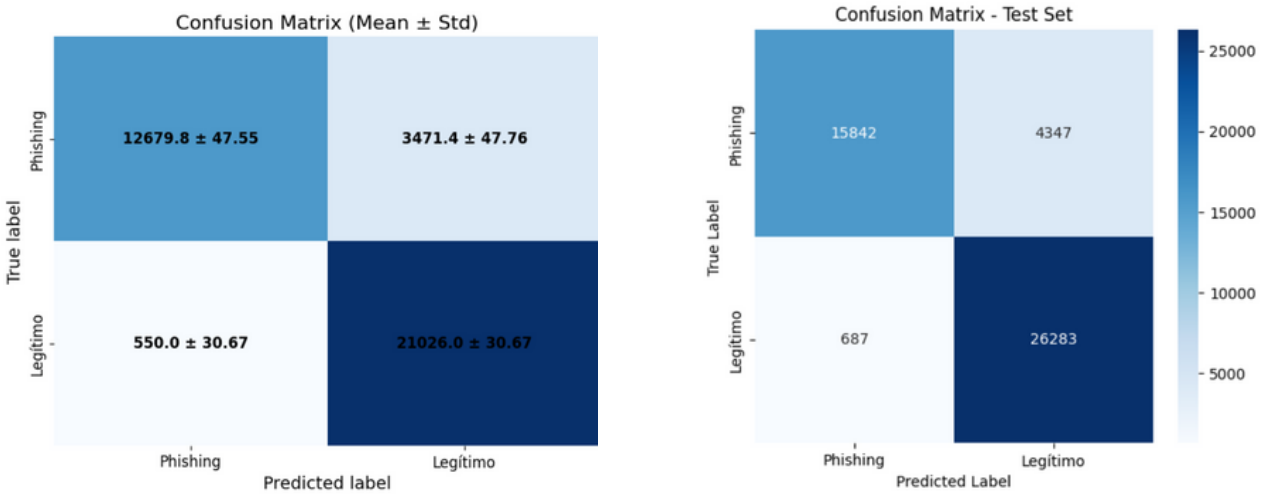

Fig. 10 - Matriz de Confusão A - Euclidiana nos dados de treino
B - Euclidiana nos dados de teste

Os resultados obtidos evidenciam que o MDC com a métrica Euclidiana apresenta limitações significativas, sobretudo na identificação de *urls* de *phishing*. Durante o treinamento, observa se um número elevado de falsos negativos, indicando que o modelo tem dificuldade em distinguir adequadamente a classe *Phishing*. No entanto, essa limitação torna-se ainda mais evidente no conjunto de teste, onde o número de erros aumenta substancialmente, sugerindo uma fraca capacidade de generalização.

Apesar da taxa de falsos positivos se manter baixa em ambos os conjuntos, a incapacidade do modelo em captar a variabilidade da classe *Phishing* reflete-se numa sensibilidade bastante reduzida. Esta discrepância entre treino e teste evidencia que o classificador não consegue manter o desempenho fora dos dados vistos, comprometendo a sua robustez. Assim, conclui-se que, sem técnicas adicionais de pré-processamento ou métricas mais eficazes, o MDC com distância Euclidiana revela-se inadequado para a tarefa de deteção de *phishing*.

Cenário 2: Dados S/ Redução de Dimensionalidade + Fisher LDA

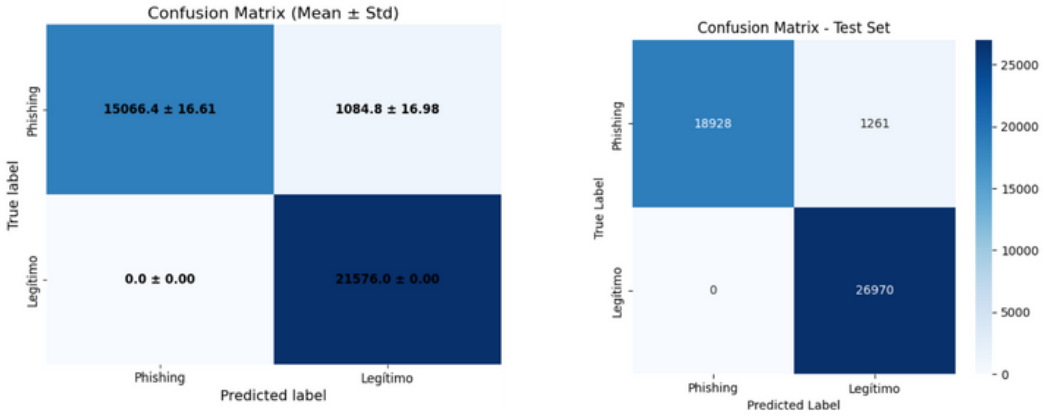

Fig. 11 - Matriz de Confusão A - Euclidiana nos dados de treino
B - Euclidiana nos dados de teste

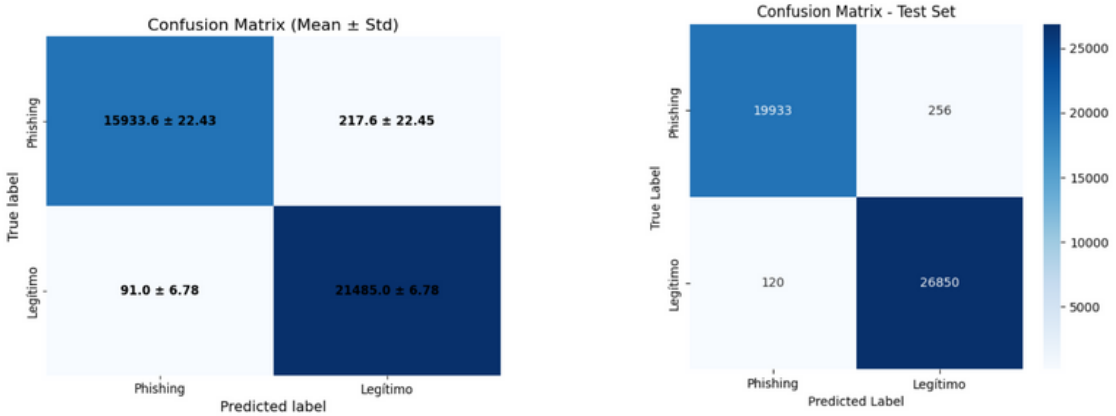

Fig. 12 - Matriz de Confusão A - Mahalanobis nos dados de treino
B - Mahalanobis nos dados de teste

No Cenário 2, aplicamos o Fisher LDA diretamente aos dados originais, utilizando duas métricas de distância: Euclidiana e *Mahalanobis*. As matrizes de confusão mostram que, embora a métrica Euclidiana tenha tido bom desempenho no treino, apresentou vários falsos negativos no teste, com amostras de *phishing* sendo classificadas como legítimas — um risco em cenários de segurança.
Em contraste, ao utilizar a distância de *Mahalanobis* (Fig. 12), o desempenho foi superior em ambos os conjuntos, com uma redução clara nos erros de classificação, especialmente nos falsos negativos. Isso mostra que a métrica de *Mahalanobis* foi mais eficaz na separação entre as classes após a projeção LDA, tornando a detecção de *phishing* mais confiável.

O gráfico de distribuição da projeção LDA (Fig. 13) mostra que as classes foram bem separadas, com a classe Legítimo apresentando uma distribuição altamente concentrada, enquanto a classe *Phishing* possui uma maior dispersão. Isso indica que o Fisher LDA conseguiu encontrar um espaço de projeção que maximiza a separação entre as classes, tornando o problema mais tratável para a classificação. No entanto, apesar da clara separação entre as distribuições, ainda é possível notar uma sobreposição residual entre as classes, o que pode explicar a existência de alguns erros de classificação.

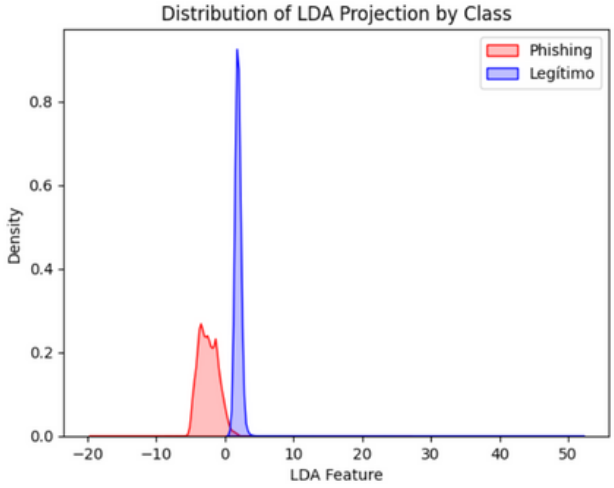

Fig. 13 - Gráfico de Dispersão da Projeção LDA

Em suma, a aplicação do Fisher LDA aos dados originais permitiu representar as classes de forma mais eficiente, facilitando a separação entre *phishing* e legítimo. Observou-se que o uso da distância de *Mahalanobis* proporcionou um desempenho superior ao da distância Euclidiana, especialmente na redução de falsos negativos. Isso reforça a importância da redução de dimensionalidade supervisionada, aliada à escolha adequada da métrica de distância, para a melhoria do desempenho do classificador.

Cenário 3: Dados S/ Redução de Dimensionalidade (com KW) + MDC

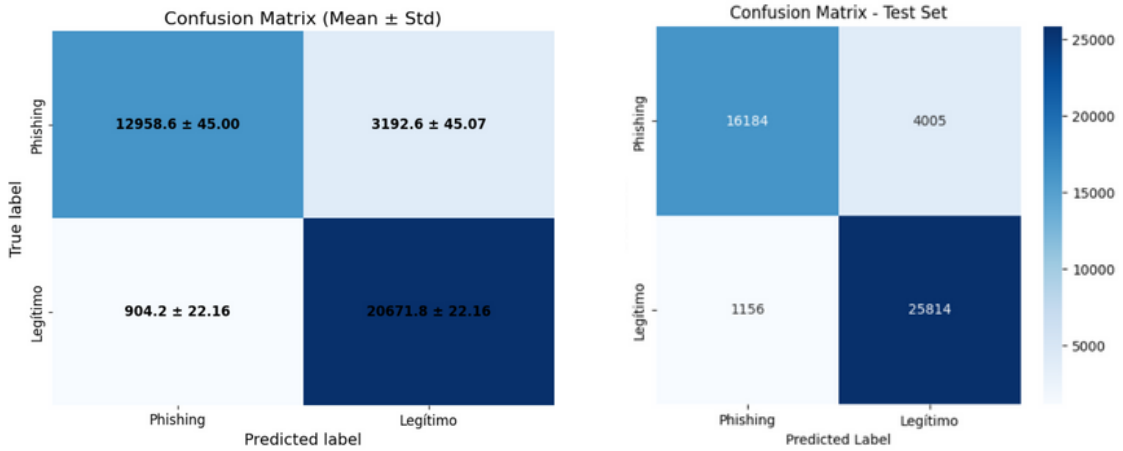

Fig. 14 - Matriz de Confusão A - Euclidiana nos dados de treino
B - Euclidiana nos dados de teste

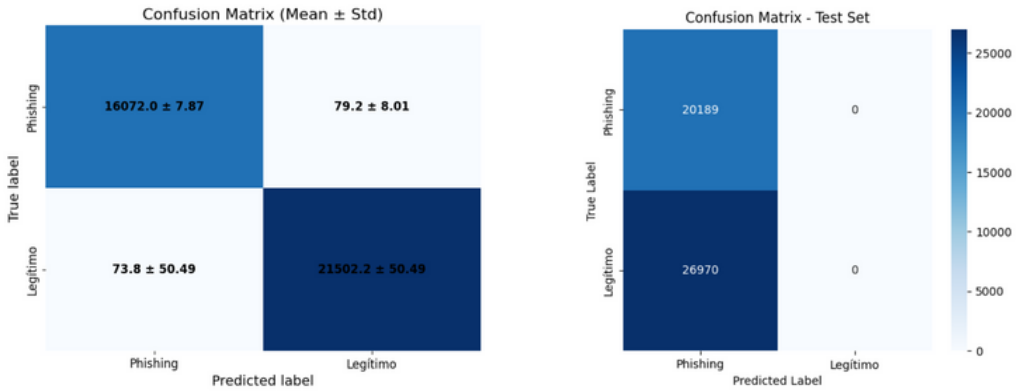

Fig. 15 - Matriz de Confusão A - Mahalanobis nos dados de treino
B - Mahalanobis nos dados de teste

Neste cenário, foi aplicada uma redução de dimensionalidade não supervisionada com o método *Kruskal-Wallis* (KW), seguida de classificação com MDC utilizando as métricas Euclidiana e *Mahalanobis*. Com a métrica Euclidiana, o desempenho foi razoável no treino, mas nos dados de teste observou-se um número elevado de falsos negativos, com várias amostras de *phishing* classificadas como legítimas, o que compromete a detecção de ataques. Já com a métrica de *Mahalanobis*, houve uma redução significativa dos erros, tanto no treino quanto no teste, destacando-se como uma abordagem mais eficaz para separar as classes mesmo após a redução de dimensionalidade. Estes resultados reforçam a importância da escolha da métrica de distância, mesmo em cenários sem redução supervisionada.

Cenário 4: Dados S/ Redução de Dimensionalidade (com KW) + FisherLDA

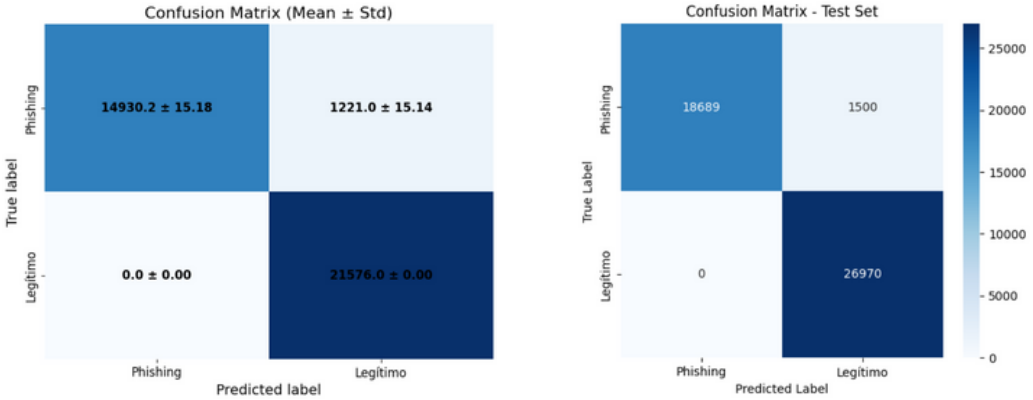

Fig. 16 - Matriz de Confusão A - Euclidiana nos dados de treino
B - Euclidiana nos dados de teste

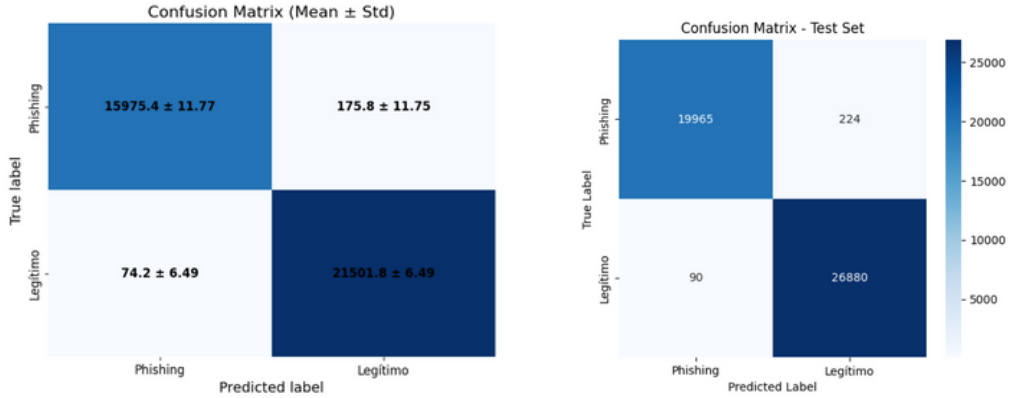

Fig. 17 - Matriz de Confusão A - Mahalanobis nos dados de treino
B - Mahalanobis nos dados de teste

- **Cenário 5: PCA + MDC**

Neste cenário foi realizada a avaliação de um classificador de distância mínima, MDC, utilizando dados transformados por PCA, o objetivo foi comparar o desempenho do modelo com duas métricas de distância: euclidiana e de *mahalanobis*.

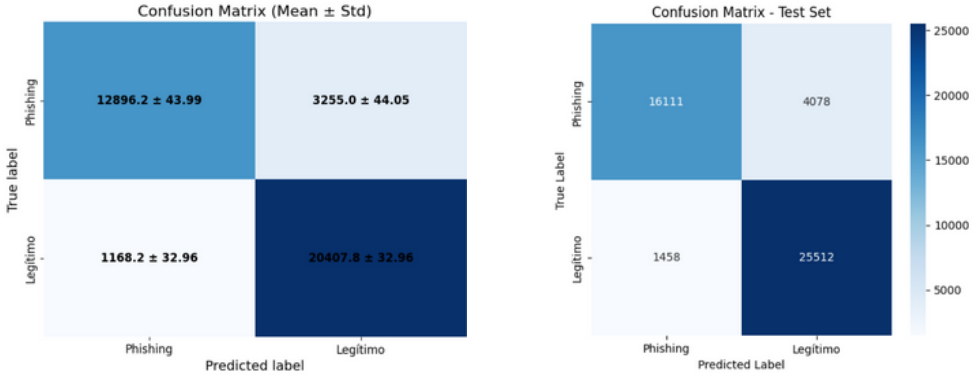

Fig. 18 - Matriz de Confusão A - Euclidiana nos dados de treino
B - Euclidiana nos dados de teste

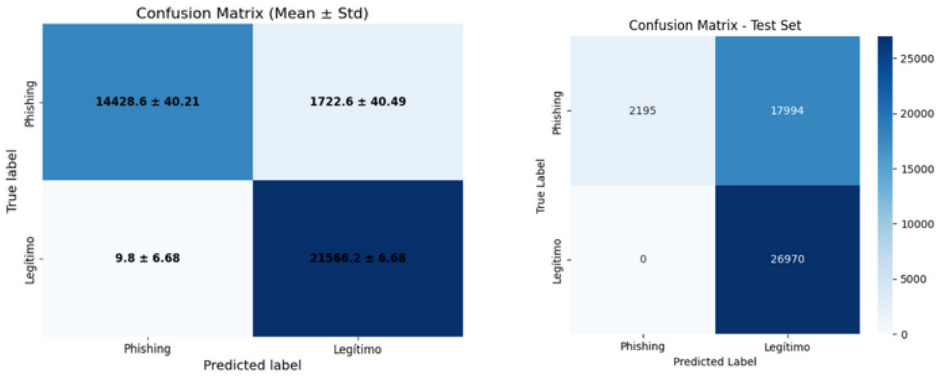

Fig. 19 - Matriz de Confusão A - Mahalanobis nos dados de treino
B - Mahalanobis nos dados de teste

Observamos assim que a métrica *mahalanobis* apresentou um desempenho superior na separação das classes quando comparada com a métrica euclidiana, isto fica evidente na matriz de confusão (Fig. 18), onde o número de erros de classificação é significativamente reduzido quando utilizada a métrica *mahalanobis*. Por outro lado, a métrica euclidiana produziu uma quantidade considerável de falsos negativos, onde *urls* de *phishing* foram classificados como legítimos, o que compromete seriamente a eficácia de detecção, deixando o sistema mais vulnerável a ameaças.

Com a métrica de *mahalanobis*, a taxa de erro foi minimizada proporcionando uma melhor separação entre classes e garantindo uma maior segurança, como evidenciado pelas matrizes de confusão que mostram uma classificação quase perfeita nos dados de teste. Isso demonstrou que considerar a variabilidade e correlação das variáveis, como faz a distância de *mahalanobis*, é essencial para melhorar o desempenho do classificador em problemas de detecção de *phishing*.

- **Cenário 6: PCA + Fisher LDA**

Neste ultimo cenário aplicamos o *Fisher* LDA sobre os dados transformados pelo PCA, seguido da classificação utilizando o MDC com duas métricas diferentes: euclidiana e *mahalanobis*.

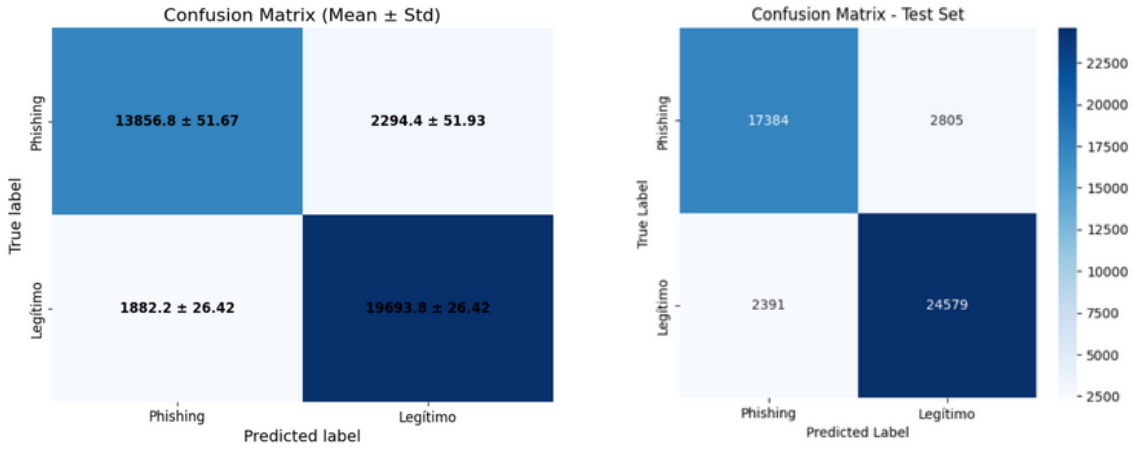

Fig. 20 - Matriz de Confusão A - Euclidiana nos dados de treino
B - Euclidiana nos dados de teste

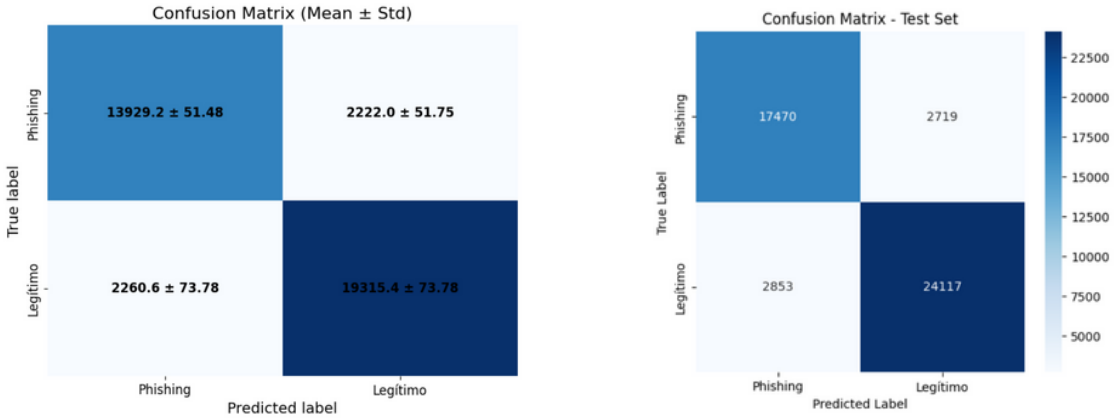

Fig. 21 - Matriz de Confusão A - Mahalanobis nos dados de treino
B - Mahalanobis nos dados de teste

Ao analisar as matrizes de confusão (Fig. 20 e 21), observamos que a utilização da métrica de *Mahalanobis* conduziu a uma redução significativa nos erros de classificação, especialmente no número de falsos positivos, ou seja, *urls* legítimas incorretamente classificadas como *phishing*. A métrica Euclidiana, por outro lado, apresentou um número mais elevado de falsos negativos, com várias *urls* de *phishing* classificadas como legítimas, o que representa um risco para a deteção eficaz de ataques.

Estes resultados foram confirmados pelas métricas de desempenho: a *accuracy* obtida com a métrica de *Mahalanobis* foi ligeiramente superior à da Euclidiana. Apesar disso, a sensibilidade da métrica Euclidiana foi marginalmente superior, indicando uma maior capacidade de identificar *urls* de *phishing*. No entanto, a especificidade da métrica de *Mahalanobis* foi substancialmente mais alta, demonstrando uma maior eficácia na identificação correta de *urls* legítimos.

Assim, embora a métrica Euclidiana apresente um bom desempenho na deteção de *phishing* devido à sua elevada sensibilidade, a métrica de *Mahalanobis* revelou-se mais equilibrada, alcançando uma melhor harmonia entre sensibilidade e especificidade. Este equilíbrio traduziu-se numa menor taxa de falsos positivos e numa classificação mais fiável das *urls* legítimas, tornando a métrica de *Mahalanobis* mais adequada para cenários onde é crucial minimizar alarmes falsos sem comprometer a segurança.

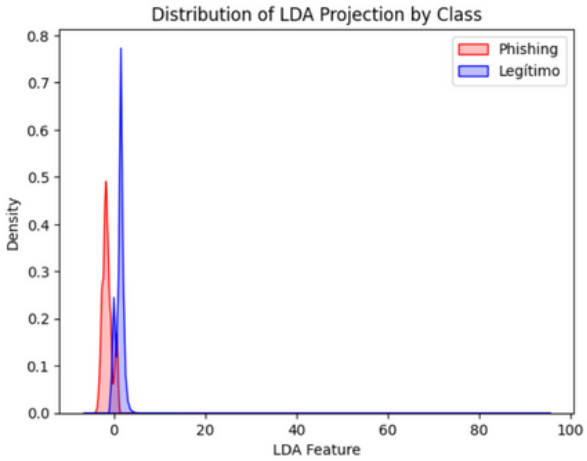

Fig. 22 - Gráfico de Dispersão da Projeção LDA

A Figura 22 apresenta a distribuição da projeção LDA para as classes *Phishing* e Legítimo. Observa-se uma separação razoável entre as distribuições das duas classes, embora exista alguma sobreposição. Esta sobreposição justifica a ocorrência de erros de classificação, especialmente no caso da métrica *Euclidiana*, que não considera a variância entre as classes. A métrica de *Mahalanobis*, ao incorporar a covariância dos dados, consegue compensar esta sobreposição, resultando numa melhor distinção entre as classes. Assim, este gráfico reforça a conclusão anterior de que a métrica de *Mahalanobis* proporciona uma separação mais eficaz no espaço projetado pela LDA, contribuindo para uma classificação mais precisa.

Comparação entre cenários

Ao comparar os seis cenários testados, observa-se que a combinação das técnicas de redução de dimensionalidade com métricas de distância adequadas tem um impacto significativo na eficácia da classificação de *urls* de *phishing*. De forma geral, os cenários que incorporam redução de dimensionalidade supervisionada (como *Fisher* LDA) ou combinações de KW + LDA, aliados à distância de *Mahalanobis*, destacaram-se pelo melhor desempenho, com menor número de erros de classificação e maior robustez entre treino e teste.

O cenário 1 apresentou os piores resultados. Verificou-se uma elevada taxa de falsos negativos, sobretudo no conjunto de teste, revelando uma baixa capacidade de generalização e compromete seriamente a segurança do sistema ao não identificar adequadamente *urls* maliciosas.

Já os cenários 2, 4 e 6, que envolvem o uso de Fisher LDA (com ou sem KW, ou PCA), revelaram-se os mais eficazes, especialmente quando combinados com a distância de *Mahalanobis*. No cenário 4 (KW + Fisher LDA), por exemplo, observou-se um desempenho excelente no teste com a métrica *Euclidiana* (zero falsos negativos), embora a *Mahalanobis* tenha mantido maior consistência e equilíbrio entre treino e teste.

O cenário 3, utilizando KW seguido de MDC, mostrou que mesmo sem uma projeção supervisionada, a distância de *Mahalanobis* continua a oferecer vantagens significativas face à *Euclidiana*, reforçando a sua eficácia na distinção entre classes, mesmo em cenários menos estruturados.

Os cenários com PCA (5 e 6) também evidenciaram essa tendência: *Mahalanobis* contribuiu para uma separação mais clara das classes e redução de falsos positivos e negativos, enquanto a *Euclidiana* apresentou limitações, especialmente na generalização para o conjunto de teste.

Em suma, os melhores desempenhos foram registados quando *Fisher* LDA foi aplicado após uma redução preliminar (KW ou PCA) e com o uso da distância de *Mahalanobis*, enquanto o pior desempenho surgiu no cenário em que nenhuma técnica de redução foi utilizada e optou-se pela distância *Euclidiana*, evidenciando a importância de um pipeline bem estruturado para garantir um sistema de deteção de *phishing* confiável e eficaz. Para visualizar as figuras com a comparação detalhada dos resultados, pode-se recorrer ao código em notebook em anexo.

- ***Bayes Classification***

O classificador *Naive Bayes* é um modelo estatístico baseado no teorema de *Bayes*, utilizado para resolver problemas de classificação. No nosso caso, foi usada a versão Gaussiana, que assume que as variáveis seguem distribuições contínuas, mesmo que os dados não apresentem exatamente uma distribuição normal. Durante a fase de previsão, o modelo utiliza essas distribuições gaussianas para calcular a probabilidade de uma nova instância pertencer a cada classe, escolhendo a classe com menor risco ou custo esperado. A principal vantagem deste classificador está na sua simplicidade e rapidez, embora o desempenho possa ser afetado quando existe forte correlação entre as variáveis.

- **Resultados**

Para avaliar os impactos da redução de dimensionalidade, consideramos quatro abordagens distintas:

1. Dados S/ redução de dimensionalidade (dados de treino originais) + *Bayes Classification*
2. Dados S/ redução de dimensionalidade (apenas com o KW) + *Bayes Classification*
3. LDA + *Bayes Classification*
4. PCA + *Bayes Classification*

- **Cenário 1: Dados S/ redução de dimensionalidade (dados de treino originais) + Bayes Classification**

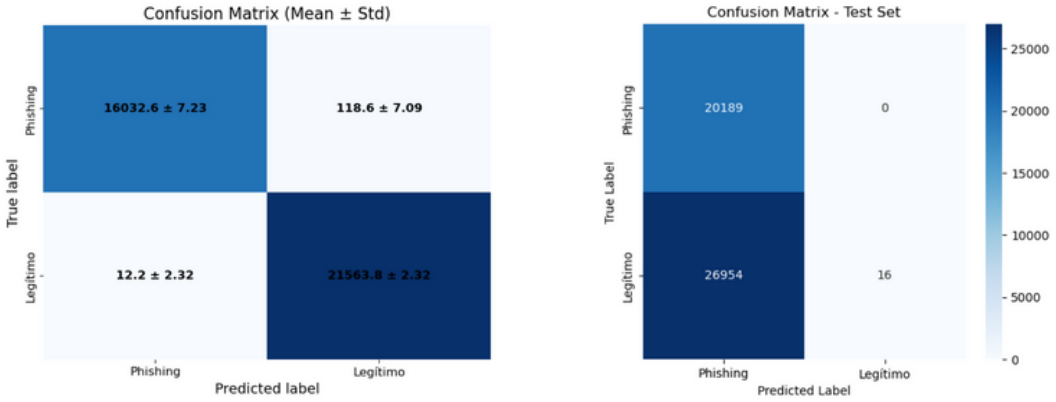

Fig. 23 - Matriz de Confusão A - Nos dados de treino
B - Nos dados de teste

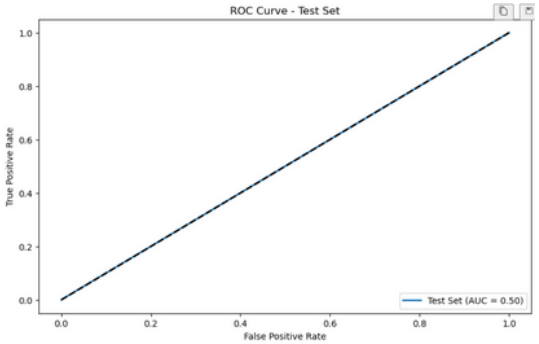

Fig. 24 - Curva de ROC

Os resultados indicam claramente que o classificador Naive Bayes aplicado aos dados originais é inadequado para esta tarefa de detecção de *phishing*. O AUC de 0.50 e a alta taxa de falsos positivos demonstram que o modelo falha em estabelecer uma fronteira de decisão eficaz. Esta falha provavelmente resulta da violação da premissa de independência do *Naive Bayes*, agravada pela ausência de técnicas apropriadas de redução de dimensionalidade. Para melhorar o desempenho, seria essencial implementar métodos como PCA ou seleção de características antes da classificação, ou considerar algoritmos alternativos que lidem melhor com dados correlacionados e de alta dimensionalidade.

- **Cenário 2: Dados S/ redução de dimensionalidade (apenas com o KW) + Bayes Classification**

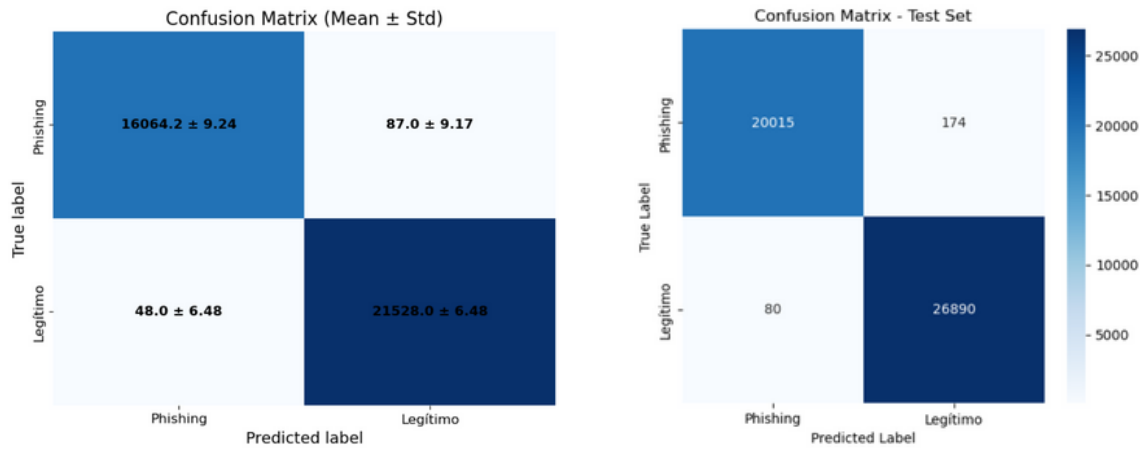

Fig. 25 - Matriz de Confusão A - Nos dados de treino
B - Nos dados de teste

Os resultados mostram uma melhoria drástica no desempenho do classificador *Naive Bayes* quando aplicada a seleção de características via teste *Kruskal-Wallis*. A curva ROC apresenta um AUC perfeito de 1.00, indicando capacidade máxima de discriminação entre classes. As matrizes de confusão confirmam esta excelência: no conjunto de teste, o modelo identificou corretamente 20.015 *urls* de *phishing* (99,1% dos *phishing*) e 26.890 *urls* legítimos (99,7% dos legítimos), com pouquíssimos erros (174 falsos negativos e 80 falsos positivos). Esta performance superior demonstra como a seleção adequada de características solucionou os problemas do cenário anterior, permitindo ao classificador Bayes operar eficientemente ao trabalhar apenas com as variáveis mais discriminativas e menos correlacionadas. O teste *Kruskal-Wallis*, ao selecionar *features* estatisticamente significativas para diferenciar as classes, contornou as limitações da premissa de independência do *Naive Bayes*, resultando em um modelo altamente eficaz para detecção de *phishing*.

- **Cenário 3: LDA + *Bayes Classification***

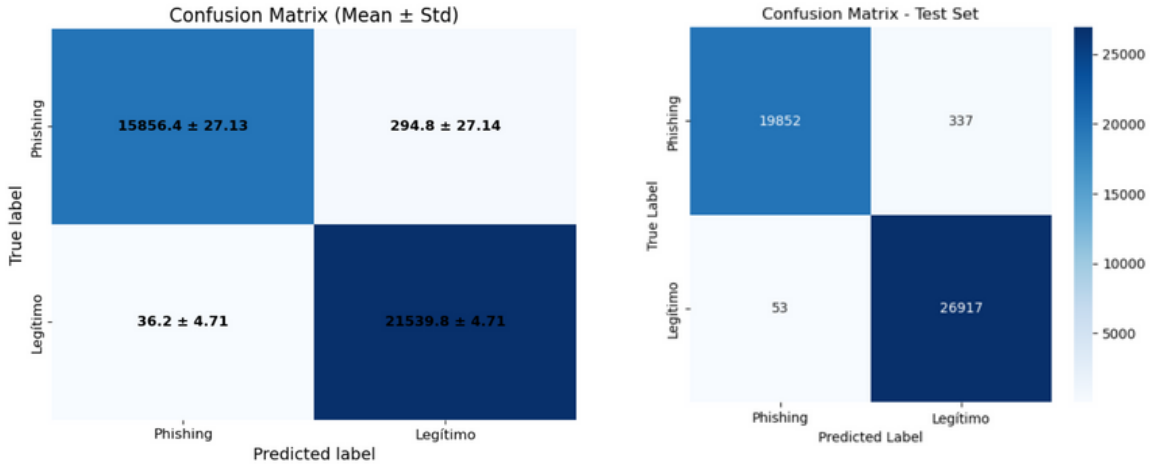

Fig. 26 - Matriz de Confusão A - Nos dados de treino
B - Nos dados de teste

O classificador *Naive Bayes* combinado com a LDA apresenta resultados muito bons, embora ligeiramente inferiores ao cenário anterior. As matrizes de confusão mostram que o modelo identificou corretamente 19.852 *urls* de *phishing* (98,3% dos *phishing*) e 26.917 *urls* legítimos (99,8% dos legítimos). Observa-se um pequeno aumento nos falsos negativos (337 *phishing* classificados como legítimos) comparado ao cenário com *Kruskal-Wallis*, mas mantém um número baixo de falsos positivos (53). Esta performance demonstra que a LDA foi eficaz na transformação do espaço de características, criando combinações lineares que maximizam a separação entre classes e minimizam a variância dentro das classes, fornecendo ao classificador *Bayes* um conjunto de dados mais adequado às suas premissas. A redução de dimensionalidade via LDA conseguiu capturar a estrutura discriminativa dos dados, resultando em um modelo confiável para detecção de *phishing*.

- **Cenário 4: PCA + *Bayes Classification***

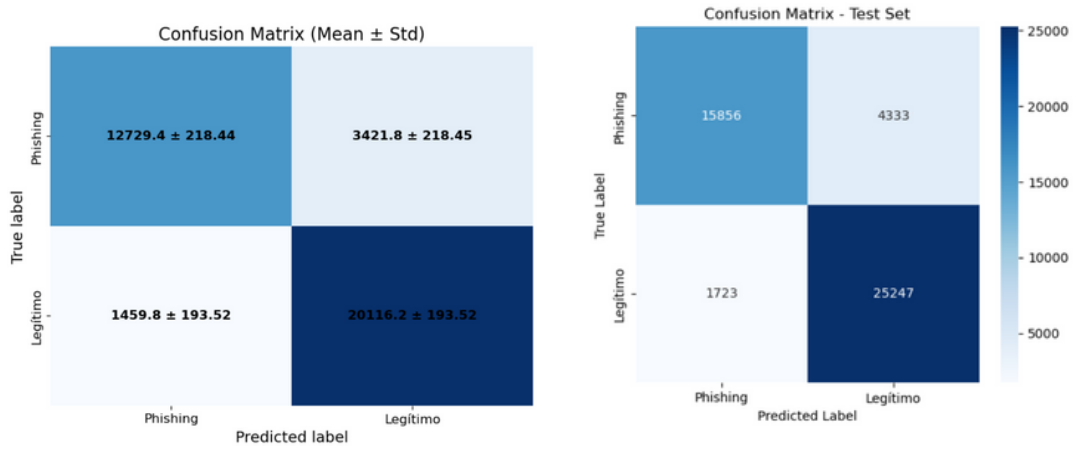

Fig. 27 - Matriz de Confusão A - Nos dados de treino
B - Nos dados de teste

A combinação de PCA com o classificador *Naive Bayes* apresenta um desempenho significativamente inferior aos cenários com LDA e *Kruskal-Wallis*. As matrizes de confusão revelam problemas importantes: o modelo conseguiu identificar apenas 15.856 *urls* de *phishing* (78,5% dos *phishing*), deixando 4.333 falsos negativos, e classificou incorretamente 1.723 *urls* legítimos como *phishing* (6,4% de falsos positivos). O alto desvio padrão nas médias (±218,44 para verdadeiros positivos) também indica instabilidade no modelo. Esta queda de performance ocorre porque, diferente do LDA e Kruskal-Wallis que são técnicas supervisionadas focadas na discriminação entre classes, o PCA é um método não-supervisionado que prioriza a variância total dos dados sem considerar as informações de classe. Consequentemente, os componentes principais selecionados podem não ser os mais relevantes para a tarefa de classificação, resultando em uma representação que, embora eficiente para compressão de dados, não otimiza a separabilidade entre *urls* legítimos e *phishing*.

- **Comparação entre os cenários em termos de métricas: *Accuracy*, *Sensitivity* e *Specificity***

A comparação dos quatro cenários com o classificador *Naive Bayes* mostra que o desempenho varia significativamente consoante o pré-processamento aplicado. As abordagens com *Kruskal-Wallis* (KW + *Bayes*) e LDA (LDA + *Bayes*) apresentaram os melhores resultados, com *accuracy*, sensibilidade e especificidade próximas de 100%. O cenário com PCA teve desempenho aceitável, mas inferior, com *accuracy* de 87% e menor especificidade (79%). Já o modelo com dados apenas normalizados (Original *Bayes*) teve baixa *accuracy* (43%) e comportamento enviesado, classificando quase tudo como *phishing*. As curvas ROC reforçam estas conclusões, destacando a eficácia das abordagens supervisionadas. Conclui-se que o Naive Bayes, embora simples, pode alcançar excelente desempenho quando combinado com pré-processamento adequado, especialmente técnicas que consideram a informação das classes. Para visualizar as figuras com a comparação detalhada dos resultados, pode-se recorrer ao código em notebook em anexo.

- **Classificador KNN**

O *k-Nearest Neighbors* procura os *k* vizinhos mais próximos e classifica uma nova amostra de acordo com a classe de maior prevalência. Durante a construção deste classificador o desafio é encontrar o melhor *k* que maximiza a performance do algoritmo. Apesar de o algoritmo ser simples de perceber teoricamente e até de implementar, pode tornar-se um algoritmo exaustivo para grandes volumes de dados, devido à procura dos pontos mais próximos.

- **Resultados**

Para avaliar os impactos da redução de dimensionalidade, consideramos quatro abordagens distintas:

1. KNN + Dados de treino originais
2. KNN + Dados após KW
3. KNN + LDA
4. KNN + PCA

- **Cenário 1: KNN + Dados de treino originais**

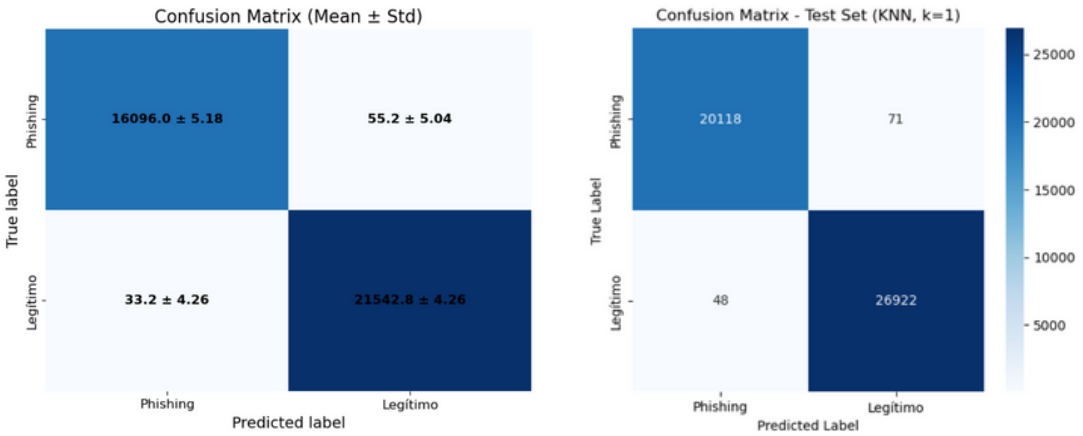

Fig. 28 - Matriz de Confusão A - Nos dados de treino
B - Nos dados de teste

O modelo KNN com k=1 aplicado ao conjunto de dados original demonstra desempenho excepcional na classificação de *urls* como *phishing* ou legítimas. A precisão média de 99,77% é confirmada pela matriz de confusão que mostra alta concentração de classificações corretas: 20.118 *urls phishing* e 26.922 legítimos foram corretamente identificados, com apenas 71 falsos negativos e 48 falsos positivos. A ROC com AUC=1.00 evidencia a capacidade quase perfeita do modelo em distinguir entre as classes, alcançando sensibilidade de 99,82% (identificação de *phishing*) e especificidade de 99,65% (identificação de *urls* legítimas). Estes resultados indicam que as características originais contêm informações altamente discriminativas que o algoritmo KNN consegue utilizar eficazmente mesmo sem redução dimensional.

- **Cenário 2: KNN + Dados após KW**

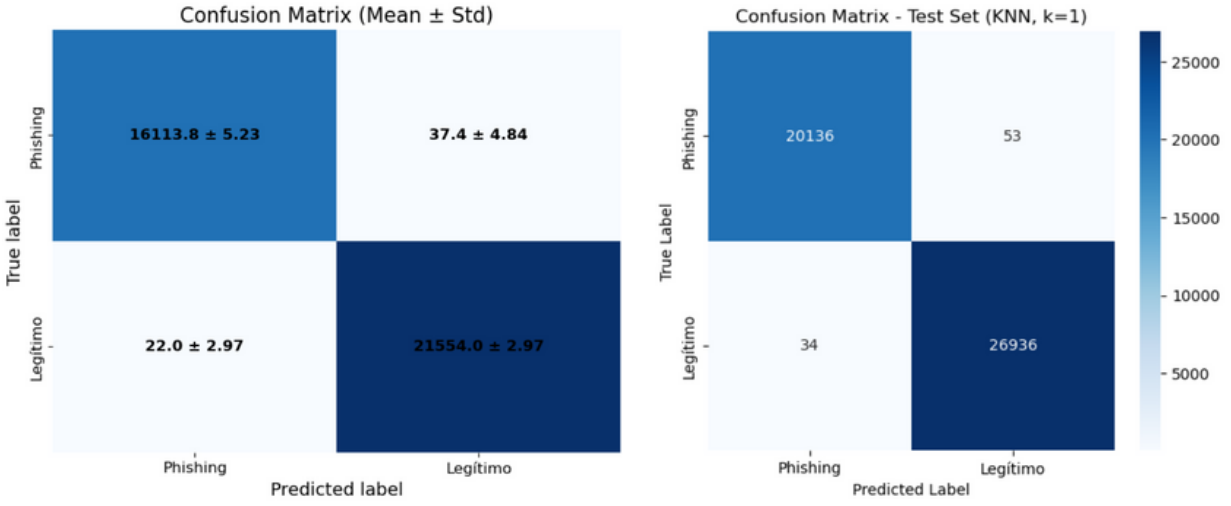

Fig. 29 - Matriz de Confusão A - Nos dados de treino
B - Nos dados de teste

O modelo KNN com k=1 aplicado aos dados após seleção de características Kruskal-Wallis apresenta desempenho superior ao cenário anterior, com precisão média de 99,84%. A matriz de confusão mostra resultados ainda mais robustos: 20.136 urls phishing e 26.936 legítimos foram classificados corretamente, reduzindo os falsos negativos para 53 e os falsos positivos para apenas 34. Isto se traduz em sensibilidade de 99,87% e especificidade de 99,74%, indicando que a seleção de características relevantes via KW potencializou a capacidade discriminativa do modelo. Este cenário demonstra que a eliminação de características irrelevantes não apenas manteve o desempenho excepcional do KNN, mas também o aprimorou ligeiramente, possivelmente ao reduzir o ruído nos dados e focar apenas nas variáveis com maior poder discriminativo entre *urls phising* e legítimas.

- **Cenário 3: KNN + LDA**

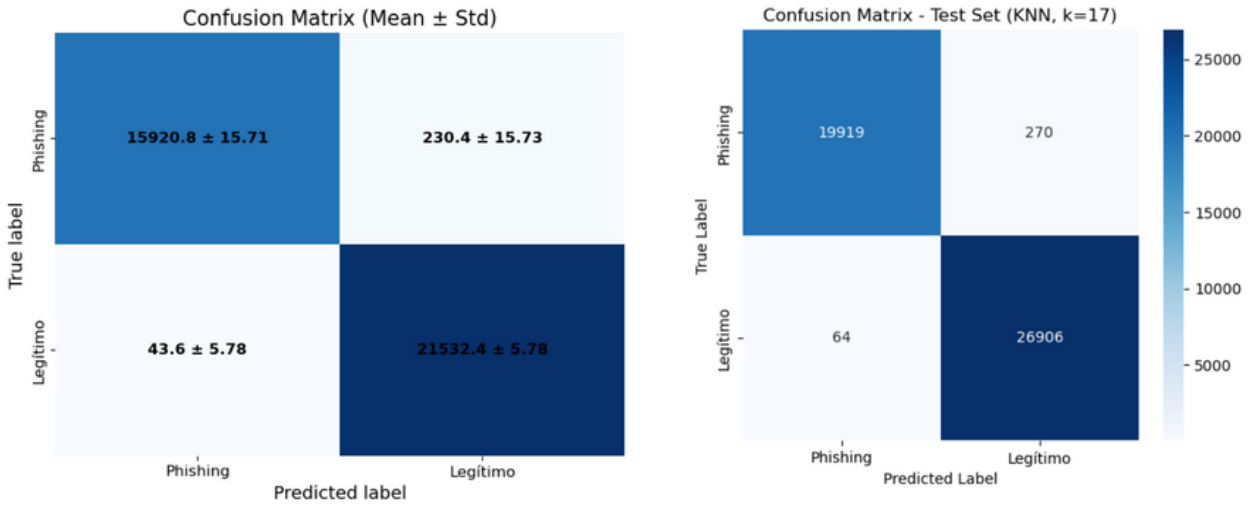

Fig. 30 - Matriz de Confusão A - Nos dados de treino
B - Nos dados de teste

O modelo KNN com LDA (k=17) mantém alta eficácia com precisão de 99,27%. A matriz de confusão mostra 19.919 acertos de *phishing* e 26.906 de legítimos, com 270 falsos negativos e 64 falsos positivos. A sensibilidade (99,76%) permanece excelente, com leve queda na especificidade (98,66%). Apesar do desempenho ligeiramente inferior aos cenários anteriores, a transformação LDA oferece vantagem na redução dimensional, simplificando o modelo enquanto preserva capacidade discriminativa robusta para identificação de *urls* maliciosas.

- **Cenário 4: KNN + PCA**

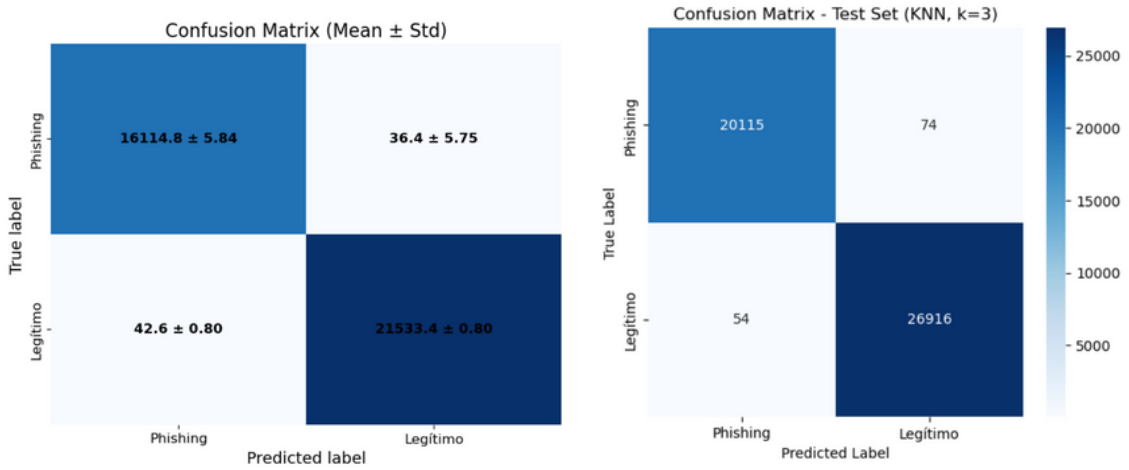

Fig. 31 - Matriz de Confusão A - Nos dados de treino
B - Nos dados de teste

O modelo KNN com redução dimensional via PCA (k=3) demonstra excelente desempenho com precisão de 99,79%. A matriz de confusão revela 20.115 *urls phishing* e 26.916 legítimos corretamente classificados, com apenas 74 falsos negativos e 54 falsos positivos. A sensibilidade de 99,80% e especificidade de 99,63% confirmam a robustez do modelo. A redução dimensional por PCA preserva eficientemente as características discriminativas dos dados originais, permitindo um valor de k menor (3) em comparação ao LDA, o que sugere melhor manutenção da estrutura informativa relevante para classificação. Este cenário oferece equilíbrio ideal entre simplificação do modelo e manutenção da alta capacidade preditiva.

- **Comparação entre os cenários em termos de métricas: *Accuracy*, *Sensitivity* e *Specificity***

A comparação dos quatro cenários KNN revela excelente desempenho em todos os casos. O modelo com seleção de características *Kruskal-Wallis* apresenta os melhores resultados: precisão de 99,84%, sensibilidade de 99,87%, especificidade de 99,74% e AUC de 0,998332. PCA+KNN ocupa o segundo lugar, seguido pelo KNN com dados originais e LDA+KNN. As curvas ROC quase perfeitas (AUC próximo a 1,0) confirmam a robustez do KNN para detecção de phishing, independente do pré-processamento. A abordagem KW oferece o melhor equilíbrio entre redução dimensional e poder discriminativo, sendo ideal para sistemas reais de detecção de phishing que necessitam de alta eficácia e eficiência computacional. Para visualizar as figuras com a comparação detalhada dos resultados, pode-se recorrer ao código em notebook em anexo.

- **Classificador SVM**

O classificador SVM (*Support Vector Machine*) é um modelo que procura um hiperplano para separar as classes de dados da melhor forma possível, maximizando a distância entre elas. A principal vantagem do SVM é a sua eficácia em problemas com muitas caraterísticas, especialmente quando as classes não são linearmente separáveis, usando *kernels* como o RBF (*Radial Basis Function*). Embora seja um modelo poderoso, o SVM pode ser lento para grandes volumes de dados.

- **Resultados**

Para avaliar os impactos da redução de dimensionalidade, consideramos quatro abordagens distintas:

1. SVM + Dados Originais
2. SVM + Dados após KW
3. SVM + LDA
4. SVM + PCA

Cenário 1: SVM + Dados Originais

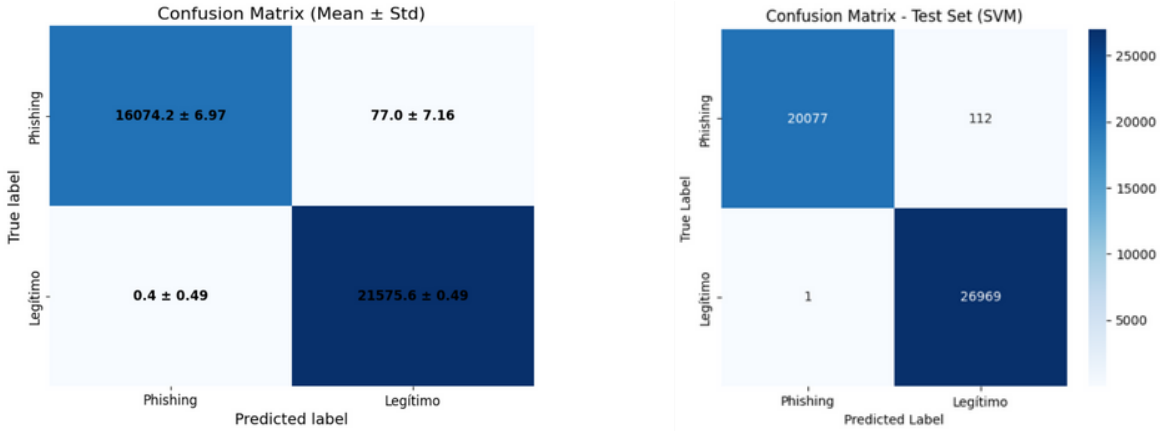

Fig. 32 - Matriz de Confusão A-No conjunto de Treino
B-No Conjunto de Teste

Na Figura 32 são apresentadas as matrizes de confusão que ilustram o desempenho do classificador SVM, com os resultados médios da validação cruzada à esquerda e os do conjunto de teste à direita. Durante a validação cruzada, o modelo demonstrou elevada consistência, com poucos falsos negativos (77 ± 7.16) e praticamente nenhum falso positivo, indicando ótima separação entre *urls* legítimos e de *phishing*. No teste, o desempenho manteve-se elevado, com 112 falsos negativos e apenas 1 falso positivo, refletindo uma excelente sensibilidade e especificidade. Apesar do baixo número de erros, é importante destacar que falsos negativos ainda representam um risco, embora os resultados confirmem a fiabilidade do modelo neste cenário.

Cenário 2: SVM + Dados após KW

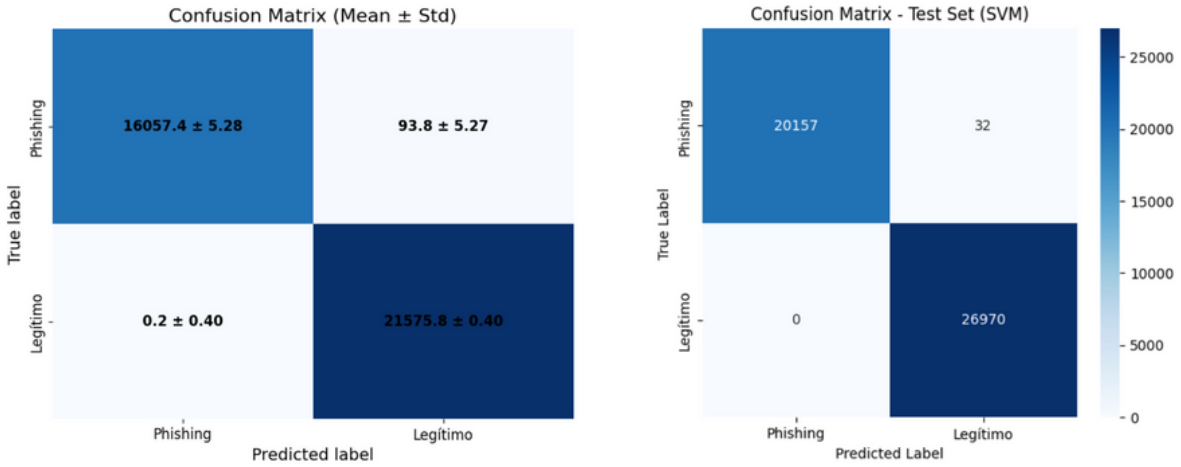

Fig. 33 - Matriz de Confusão A-No conjunto de Treino
B-No Conjunto de Teste

No Cenário 2, após a aplicação do teste de *Kruskal-Wallis* (KW) para seleção de atributos, avaliou-se o desempenho do classificador SVM, cujos resultados estão ilustrados na Figura 33. Os resultados demonstram uma elevada precisão do modelo, tanto durante o treino como na fase de teste. Na validação cruzada, registaram-se em média apenas 93.8 ± 5.27 classificações incorretas de *phishing* como legítimo e 0.2 ± 0.40 casos de legítimos como *phishing*. Já no conjunto de teste, o número de erros manteve-se muito reduzido, com 32 falsos negativos e nenhum falso positivo, evidenciando uma generalização eficaz indicando assim que a utilização do KW permitiu melhorar a separação entre as classes, contribuindo para um desempenho robusto e fiável do SVM na deteção de *urls* de *phishing*.

Cenário 3: SVM + LDA

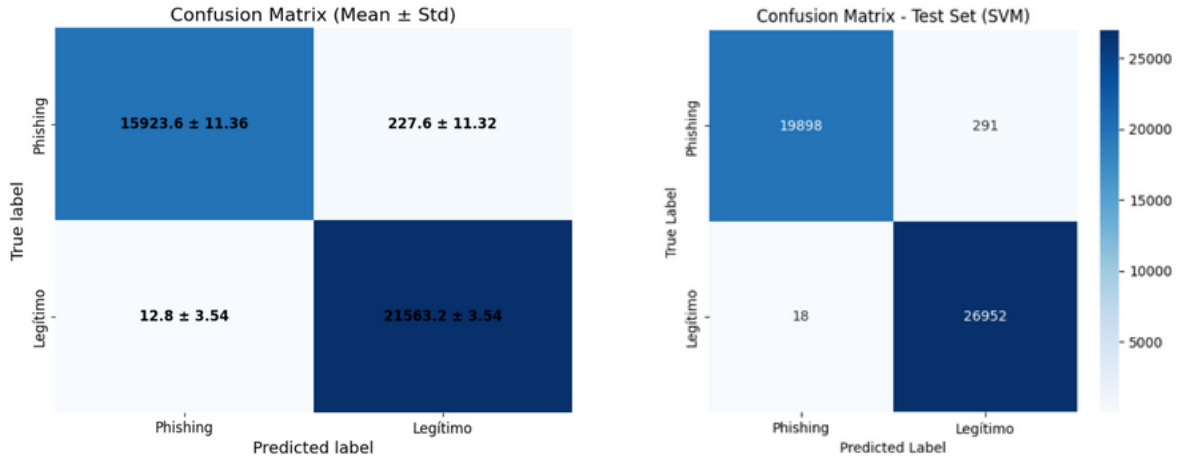

Fig. 34 - Matriz de Confusão A-No conjunto de Treino
B-No Conjunto de Teste

Com base nos resultados obtidos com a aplicação do LDA antes do classificador SVM, a Figura 34 apresenta o desempenho do modelo. Durante a validação cruzada, verifica-se um desempenho consistente, com uma média de 227.6 ± 11.32 falsos negativos (amostras de *phishing* classificadas como legítimas) e 12.8 ± 3.54 falsos positivos. No teste, embora o modelo continue eficaz, verifica-se um aumento de erros, com 291 amostras de *phishing* incorretamente classificadas como legítimas e 18 amostras legítimas como *phishing*.
Estes resultados sugerem que a aplicação do LDA contribuiu para uma melhor separação entre as classes, mas a maior variabilidade da classe *phishing* pode justificar os erros de classificação observados, sobretudo no conjunto de teste. Ainda assim, o modelo mantém um desempenho global elevado.

Cenário 4: SVM + PCA

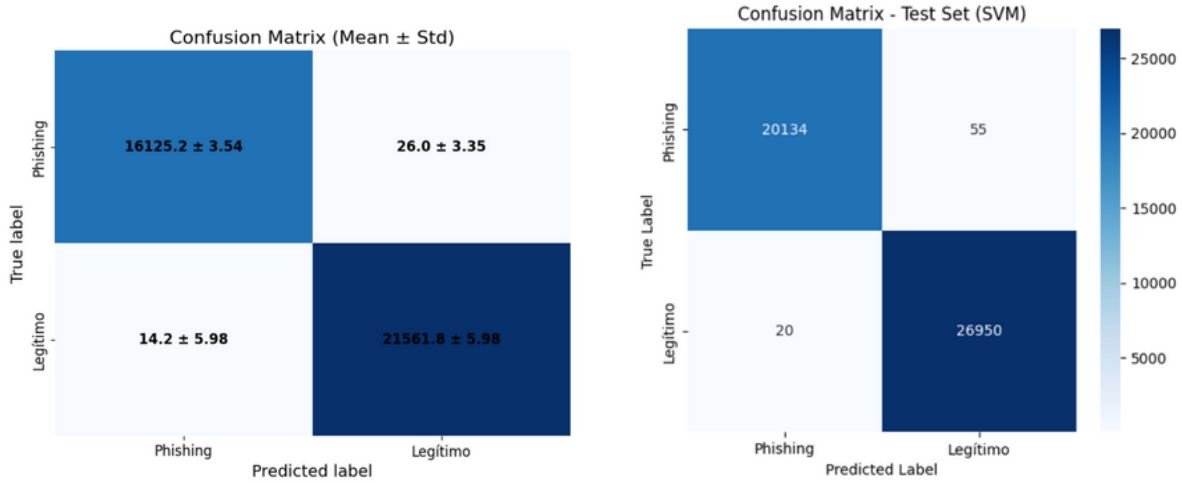

Fig. 35 - Matriz de Confusão A-No conjunto de Treino
B-No Conjunto de Teste

No Cenário 4, aplicou-se o método de redução de dimensionalidade PCA (Análise de Componentes Principais) antes da utilização do classificador SVM. Conforme ilustrado na Figura 35, os resultados obtidos evidenciam um desempenho altamente eficaz do modelo, tanto na fase de treino como na de teste. Durante a validação cruzada, observaram-se em média apenas 26.0 ± 3.35 classificações incorretas de *phishing* como legítimo e 14.2 ± 5.98 de legítimos como *phishing*, refletindo uma taxa de erro extremamente baixa. No conjunto de teste, a performance manteve-se elevada, com apenas 55 falsos negativos e 20 falsos positivos, o que indica uma boa capacidade de generalização do modelo. Assim, a utilização do PCA demonstrou ser eficaz na manutenção da separabilidade entre as classes, permitindo ao SVM apresentar um desempenho robusto na tarefa de deteção de *urls* de *phishing*.

Comparação entre cenários:

Comparando os 4 cenários, observa-se que a combinação do SVM com diferentes técnicas de pré-processamento impacta diretamente o desempenho do modelo. A aplicação do *Kruskal-Wallis* (Cenário 2) destacou-se por proporcionar os melhores resultados, com elevada precisão e quase nenhum erro, revelando uma separação muito eficaz entre *urls* legítimos e de *phishing*. A seguir, o uso do PCA (Cenário 4) também se mostrou vantajoso, mantendo um desempenho sólido mesmo após a redução da dimensionalidade dos dados.
O modelo com dados originais (Cenário 1) demonstrou resultados consistentes, confirmando que o SVM já apresenta boa performance mesmo sem transformação dos atributos. Por outro lado, o cenário com LDA (Cenário 3) obteve os resultados menos favoráveis, com maior número de erros, especialmente na identificação de *phishing*, o que limita a sua eficácia. Desta forma, conclui-se que a integração do SVM com a seleção de atributos via *Kruskal-Wallis* foi a abordagem mais eficaz na deteção de *urls* de *phishing*. Para visualizar as figuras com a comparação detalhada dos resultados, pode-se recorrer ao código em notebook em anexo.

- **Classificador *Random Forest***

O classificador *Random Forest* é um modelo que cria várias árvores de decisão usando subconjuntos aleatórios de características. A ideia principal é que cada árvore se especialize em um conjunto diferente de características, melhorando a diversidade do modelo. Quando uma nova amostra precisa ser classificada, ela é passada por todas as árvores, e o resultado final é determinado pela votação da maioria. Este modelo é eficaz porque combina a robustez de várias árvores, reduzindo o risco de *overfitting* melhorando a precisão em comparação com uma única árvore de decisão.

- **Resultados**

Para avaliar os impactos da redução de dimensionalidade, consideramos quatro abordagens distintas:

1. *Random Forest* + Dados de Originais
2. *Random Forest* + Dados após KW
3. *Random Forest* + LDA
4. *Random Forest* + PCA

Cenário 1: *Random Forest* + Dados Originais

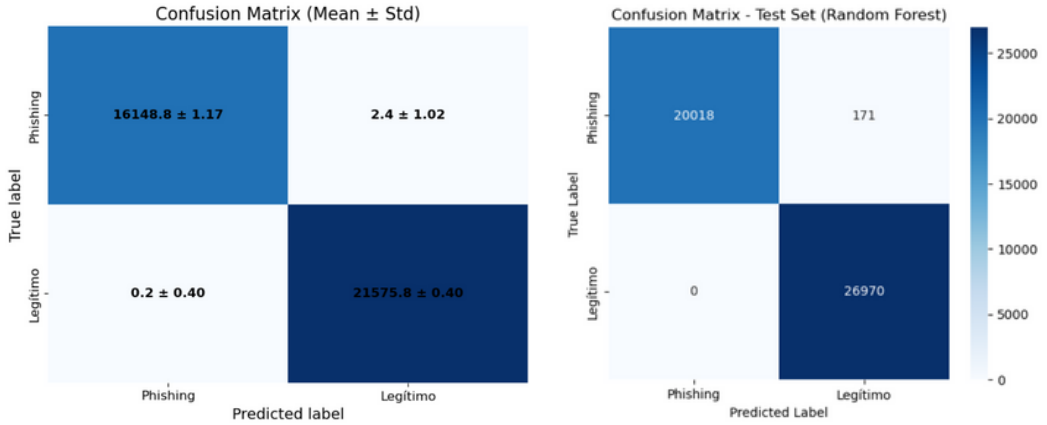

Fig. 36 - Matriz de Confusão A - Nos dados de treino
B - Nos dados de teste

O classificador *Random Forest* com dados originais demonstrou desempenho excepcional, identificando corretamente 99,99% das *urls* durante validação cruzada (16148,8 VPs, 21575,8 VNs) e 99,64% no conjunto de teste (20018 VPs, 26970 VNs). A taxa de falsos negativos permaneceu extremamente baixa (0,015% na validação e 0,85% no teste), enquanto os falsos positivos foram praticamente inexistentes (0,001% na validação e 0% no teste). Estes resultados indicam que o *Random Forest* consegue extrair eficientemente padrões relevantes dos dados brutos, oferecendo uma solução robusta e confiável para detecção de *phishing* sem necessidade de redução dimensional prévia.

- **Cenário 2: *Random Forest* + Dados após KW**

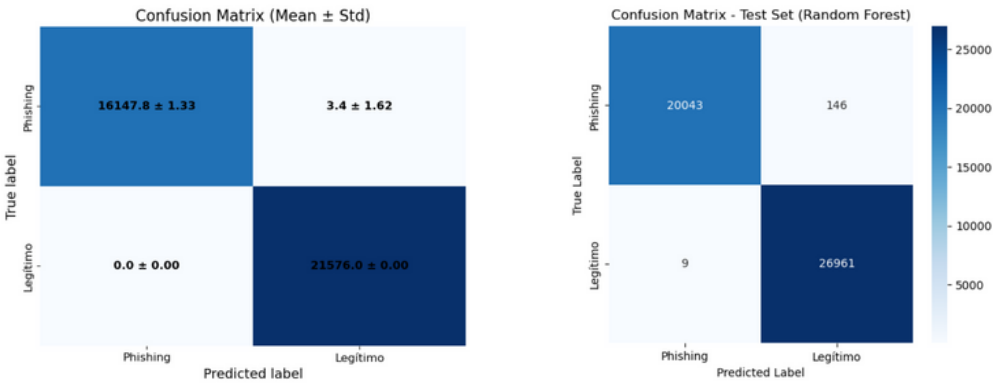

Fig. 37 - Matriz de Confusão A - Nos dados de treino
B - Nos dados de teste

O classificador *Random Forest* com dados selecionados pelo teste *Kruskal-Wallis* apresentou desempenho notável, identificando corretamente 99,98% das urls durante validação cruzada (16147,8 VPs, 21576,0 VNs) e 99,67% no conjunto de teste (20043 VPs, 26961 VNs). A redução dimensional através da seleção de características não comprometeu a eficácia, mantendo taxas mínimas de falsos negativos (0,02% na validação e 0,72% no teste) e falsos positivos (0% na validação e 0,03% no teste). Estes resultados demonstram que a seleção de características relevantes via KW mantém o poder discriminativo do modelo, proporcionando classificação eficiente com menor complexidade computacional.

- **Cenário 3: *Random Forest* + LDA**

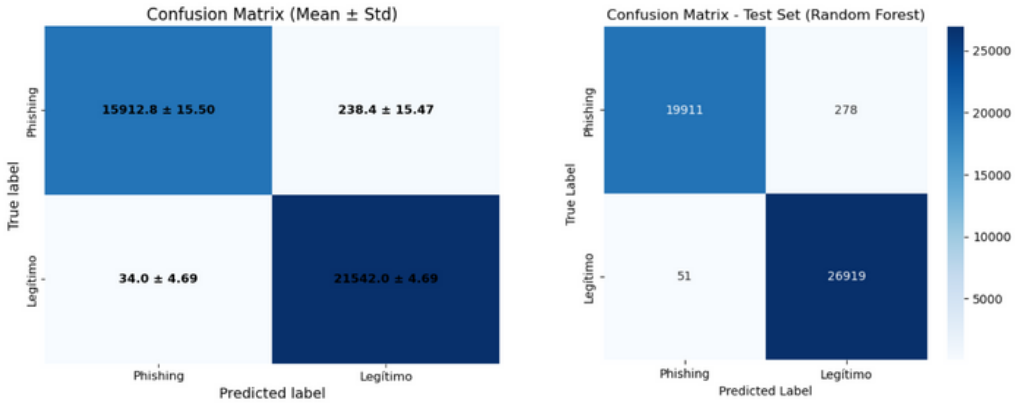

Fig. 38 - Matriz de Confusão A - Nos dados de treino
B - Nos dados de teste

O classificador *Random Forest* com redução dimensional via LDA apresentou desempenho ligeiramente inferior aos cenários anteriores, com *accuracy* de 99,27% na validação cruzada (15912,8 VPs, 21542,0 VNs) e 99,30% no conjunto de teste (19911 VPs, 26919 VNs). Observa-se um aumento notável nos erros, com taxas de falsos negativos (1,48% na validação e 1,38% no teste) e falsos positivos (0,16% na validação e 0,19% no teste) superiores aos cenários anteriores. A redução para componentes discriminantes mostrou maior variabilidade nos resultados (desvio padrão ±15,50 nos VPs), indicando que, embora o LDA preserve informações relevantes para separação das classes, há perda de capacidade discriminativa quando comparado ao uso de dados originais ou selecionados via KW.

- **Cenário 3: Random Forest + PCA**

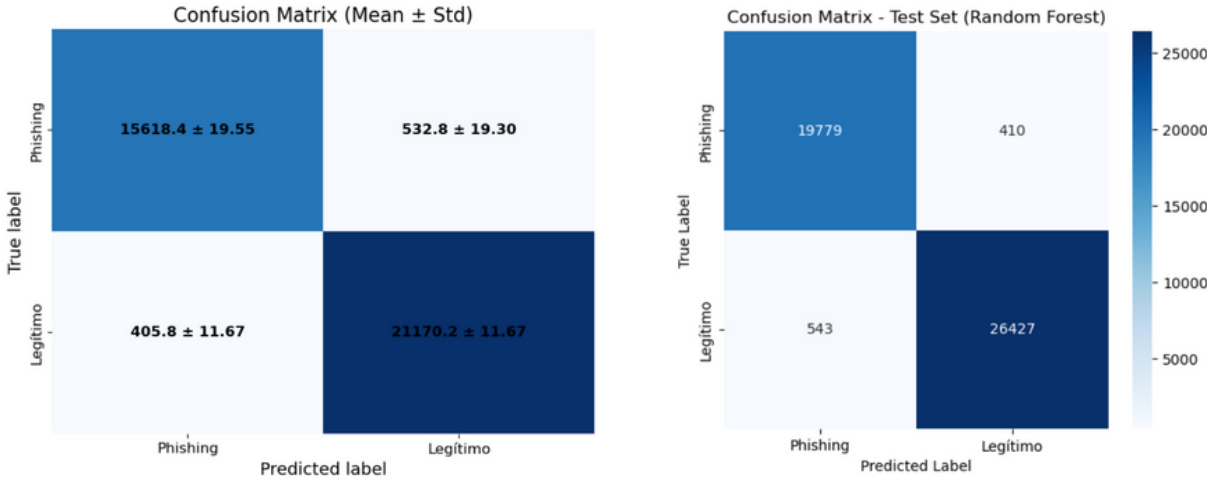

Fig. 39 - Matriz de Confusão A - Nos dados de treino
B - Nos dados de teste

O classificador *Random Forest* com redução dimensional via PCA apresentou o desempenho mais fraco entre os quatro cenários, com *accuracy* de 97,51% na validação cruzada (15618,4 VPs, 21170,2 VNs) e 97,98% no conjunto de teste (19779 VPs, 26427 VNs). As taxas de falsos negativos (3,30% na validação e 2,03% no teste) e falsos positivos (1,88% na validação e 2,01% no teste) foram significativamente maiores, com alta variabilidade nos resultados (desvio padrão ±19,55 nos VPs). Embora o PCA preserve a variância máxima dos dados, claramente perde informações discriminativas essenciais para separar *urls* legítimas de maliciosas, resultando em desempenho inferior quando comparado aos outros métodos de pré-processamento.

- **Comparação entre cenários:**

A análise dos quatro cenários de implementação do *Random Forest* revelou que os dados originais e a seleção via *Kruskal-Wallis* proporcionam desempenho superior (*accuracy* >99,98%, AUC >0,999997), com equivalência prática entre ambos. As técnicas de redução dimensional mostraram degradação progressiva, com LDA (*accuracy* 99,27%, AUC 0,997707) superando o PCA (*accuracy* 97,51%, AUC 0,997292). A abordagem com KW emerge como solução ideal para detecção de *phishing*, combinando excelente capacidade discriminativa com menor complexidade computacional, essencial para aplicações de segurança em tempo real. Para visualizar as figuras com a comparação detalhada dos resultados, pode-se recorrer ao código em notebook em anexo.

6 | CONCLUSÃO

Após uma análise abrangente dos resultados dos diferentes classificadores com as diversas técnicas de redução e seleção de *features*, podemos retirar algumas conclusões significativas.

Primeiramente, observamos que a escolha da técnica de redução e de seleção de *features* adequada desempenha um papel crucial no desempenho dos classificadores. Tanto a PCA quanto o KW (*Kruskal-Wallis*) demonstraram grande utilidade, oferecendo diferentes perspetivas sobre os dados e impactando significativamente os resultados.

A análise dos diferentes classificadores revela distintas capacidades na tarefa de deteção de *urls* de *phishing*, permitindo identificar aqueles que, de forma geral, apresentaram melhor desempenho neste problema:

1. ***Random Forest*** – Destacou-se como o classificador mais eficaz, apresentando uma precisão superior a 99% em todos os cenários, com taxas mínimas de falsos negativos e falsos positivos. Demonstrou alta robustez e excelente generalização, sendo o modelo mais fiável para deteção de *urls* de *phishing*.
2. ***K-Nearest Neighbors*** **(KNN)** – Muito próximo do *Random Forest*, especialmente após a seleção de features com KW, atingindo uma precisão média de 99,84%, com sensibilidade e especificidade elevadas.
3. ***Support Vector Machine*** **(SVM)** – Apresentou excelente desempenho com dados originais e com KW, mas teve decréscimos notáveis com LDA (291 falsos negativos no teste). Com o PCA, o desempenho foi novamente elevado. Assim, mostrou-se um classificador promissor, mas sensível à técnica de pré-processamento usada.
4. ***Naive Bayes*** – Com os dados originais teve fraco desempenho, mas com KW atingiu AUC aproximado de 1.00, mostrando que, com a seleção adequada de *features*, pode ser altamente eficaz.
5. ***Minimum Distance Classifier*** **(MDC)** – Teve o pior desempenho com a métrica Euclidiana, falhando na deteção de *phishing*. Com a métrica de *Mahalanobis* houve alguma melhoria, mas ainda ficou atrás dos restantes classificadores.

É importante notar que a escolha do melhor classificador depende dos objetivos específicos do problema em questão. Quando a prioridade é minimizar falsos negativos (conforme típico na deteção de *phishing*), classificadores como o *Random Forest* ou o KNN, especialmente com KW, são os mais adequados.

Em resumo, a seleção do classificador ideal deve ser baseada numa análise cuidadosa das necessidades do sistema e na compreensão das interações entre técnicas de pré-processamento e algoritmos de aprendizagem. A escolha correta pode levar a modelos altamente eficazes e confiáveis para a deteção de *urls* de *phishing*.

- **Comparação com o artigo**

A comparação entre o nosso trabalho e o artigo "*PhiUSIIL*" revela abordagens diferentes, mas ambas eficazes na deteção de *urls* de *phishing*. O artigo aposta em modelos de ensemble de última geração, como *XGBoost*, *LightGBM* e *CatBoost*, enquanto o nosso estudo priorizou algoritmos clássicos combinados com técnicas estatísticas de pré-processamento como *Kruskal-Wallis* (KW) e PCA.

No que diz respeito aos resultados, o *PhiUSIIL* alcança valores excecionalmente altos, com o *XGBoost* a atingir uma precisão de 99,993% e um *F1-score* de 99,994%, destacando-se como o modelo mais eficaz no estudo. Já no nosso trabalho, os melhores resultados foram obtidos com o *K-Nearest Neighbors* (KNN) após seleção de features com KW, com uma precisão de 99,84%, superando ligeiramente o modelo pré-treinado do *PhiUSIIL*, que registou 99,79%. O *Random Forest* no nosso cenário também mostrou uma performance robusta com 99,67% de precisão no teste, muito próxima da reportada no artigo (99,98%), sempre com precisões a cima de 99% em todos os outros cenários.

Além disso, enquanto o *PhiUSIIL* utilizou apenas classificadores avançados baseados em boosting e voting, nós explorámos também algoritmos mais simples como *Naive Bayes* e *Minimum Distance Classifier*, mostrando que, com um bom pré-processamento e seleção de atributos, é possível alcançar resultados bastante competitivos com modelos menos complexos.

Em síntese, o nosso estudo demonstra que técnicas clássicas, bem aplicadas, podem rivalizar com modelos de ensemble mais sofisticados, especialmente quando aliadas a uma engenharia de *features* eficaz. Essa constatação reforça que, em muitos casos, a qualidade dos dados e o processo de preparação têm tanto peso quanto a escolha do algoritmo.

07 | REFERÊNCIAS

[1] Prasad, A., & Chandra, S. (2023). PhiUSIIL: A diverse security profile empowered phishing URL detec tion framework based on similarity index and incremental learning. Computers & Security, 103545. doi: https://doi.org/10.1016/j.cose.2023.103545

[2] https://archive.ics.uci.edu/dataset/967/phiusiil+phishing+url+dataset