

Project Assignment

Chasing Phishing URLs

Reconhecimento de Padrões

Vasco Rodrigues Nr.º 2024169097

21/03/2025

Índice

1. Introdução	4
2. Implementação	5
2.1. Normalização	5
2.1.1 Min-max.....	5
2.1.2 Z-score	5
2.2. Kruskal-Wallis.....	6
2.3. PCA.....	6
2.4. Minimum Distance Classifier	7
2.4.1 Euclidean distance discriminant	7
2.4.2 Mahalanobis distance discriminant.....	8
2.5. Fisher LDA	8
2.6. Cross Validation.....	9
3. Resultados	10
3.1. Kruskal Wallis	10
3.2. Minimum Distance classifier	12
3.2.1 Euclidean Distance Discriminant	12
3.2.2 Mahalanobis Distance Discriminant	15
3.3. Fisher LDA	17
4. Discussão e Conclusão.....	19
5. Referências	20

Índice Figuras

Figura 2-1 - Demonstração de kaiser criterium e scree test.....	7
Figura 2-2 - Hiperplano de decisão do Fisher LDA	9
Figura 3-1 - Resultados Kruskal wallis.....	10
Figura 3-2 - Ranking of H-value	11
Figura 3-3 - Matriz de correlação	12
Figura 3-4 - Antes da remoção de variáveis altamente correlacionadas.....	12
Figura 3-5 - Após a remoção de variáveis altamente correlacionadas.....	12
Figura 3-6 - Resultados teste MDC-EDD sem tratamento de dados.....	13
Figura 3-7 - Resultados teste MDC-EDD com normalização z-score	13
Figura 3-8 - Resultados teste MDC-EDD com normalização e kruskal wallis.....	14
Figura 3-9 - Resultados teste MDC-EDD com normalização, kruskal wallis e PCA (kaiser)	14
Figura 3-10 - Plot para scree test.....	15
Figura 3-11 - Resultados teste MDC-EDD com normalização, kruskal wallis e PCA (scree)	15
Figura 3-12 – Resultados teste MDC-MDD com normalização z-score	16
Figura 3-13 - Resultados teste MDC-MDD com normalização z-score e kruskal wallis.....	16
Figura 3-14 - Resultados teste MDC-MDD com normalização z-score, kruskal wallis e PCA (kaiser).....	16
Figura 3-15 - Plot para scree test.....	17
Figura 3-16 - Resultados teste MDC-MDD com normalização z-score, kruskal wallis e PCA (scree).....	17
Figura 3-17 - Resultados teste Fisher LDA com normalização.....	17
Figura 3-18 - Resultados teste Fisher LDA com normalização e kruskal wallis.....	18
Figura 3-19 - Resultados de Fisher LDA com PCA.....	18

1. Introdução

Atualmente, enquanto navegamos na internet, estamos em constante perigo e sujeitos a tentativas de roubo da nossa informação pessoal sem a nossa permissão. Por esta razão, o trabalho proposto e realizado no âmbito da cadeira de Reconhecimento de Padrões apresenta uma grande importância para assegurar a nossa segurança. O objetivo do mesmo é a identificação de *URLs* maliciosos através de um modelo de classificação.

Para garantir que esta classificação seja a melhor possível, dados os dados disponíveis, serão realizadas diversas operações, como *feature selection*, *cleaning* e *normalization of data*, e *dimensionality reduction*. Além disso, serão utilizados diversos modelos, tendo os seus resultados comparados entre si, de forma a garantir que seja escolhido o modelo que melhor se comporta em dados nunca antes vistos.

Dentro das operações de *feature selection*, serão utilizados testes como o de *Kruskal-Wallis*, que permitirá verificar como as características se relacionam com a variável *target*, podendo assim escolher aquelas que são mais discriminativas e excluir as que menos. Será também verificada a matriz de correlação para identificar dados altamente correlacionados e, consequentemente, remover características redundantes.

Já nas operações de *dimensionality reduction*, será testada a utilização de *PCA (Principal Component Analysis)*, que procura representar os dados num novo eixo, de forma a obter uma melhor representação e uma mais fácil discriminação dos dados.

Os modelos que serão utilizados são: *Minimum Distance Classifier*, tanto com *Euclidean Distance Discriminant* como com *Mahalanobis Distance Discriminant*, e *Fisher LDA (Linear Discriminant Analysis)*. Ambos os modelos serão treinados e testados no mesmo *dataset*, utilizando *cross-validation*, para garantir resultados de treino/teste mais coerentes. As métricas utilizadas para avaliar estes modelos serão: *sensitivity*, *specificity*, *F1-score* e *ROC Curves*. A *accuracy* não será considerada, pois, como é referido no enunciado do próprio projeto, este é um *dataset* desequilibrado (*imbalanced dataset*), e a *accuracy*, no seu cálculo, não tem em consideração este facto.

2. Implementação

Antes da especificação da implementação dos modelos é importante mencionar que as características categóricas e binárias presentes no *dataset* foram totalmente removidas isto porque os modelos dados até agora apenas lidam com dados contínuos e são próprios para esse tipo de dados.

2.1. Normalização

O processo de normalização consiste na conversão dos valores de dados contínuos para uma escala comum, de forma a garantir que todas as características contribuam de forma equilibrada para o processo de classificação.

Neste trabalho foram considerados dois métodos de normalização: *min-max* e *z-score*.

2.1.1 Min-max

O método *min-max* transforma os dados de forma que estes se situem no intervalo 0 e 1. Para isso, utiliza o valor mínimo e máximo de cada característica. O processo é o seguinte:

1. Subtrai-se o valor mínimo da característica a cada valor dos dados.
2. Divide-se o resultado pela diferença entre o valor máximo e o mínimo da característica.

A normalização *min-max* é expressa pela fórmula:

$$x_{normalizado} = \frac{x - \min(X)}{\max(X) - \min(X)}$$

Onde x é o valor original, $\min(X)$ é o valor mínimo da característica, e $\max(X)$ é valor máximo.

Neste método *outliers* podem ter uma grande influência, pois eles afetam diretamente os valores de mínimo e máximo

2.1.2 Z-score

O método *z-score*, transforma os dados de forma que tenham uma média de 0 e um desvio padrão de 1. Para isso, utiliza a média e o desvio padrão de cada característica. O processo é o seguinte:

1. Subtrai-se a média da característica a cada valor dos dados.
2. Divide-se o resultado pelo desvio padrão da característica.

A normalização *z-score* é expressa pela fórmula:

$$x_{normalizado} = \frac{x - \mu}{\sigma}$$

Onde x é o valor original, μ é a média da característica e σ é o desvio padrão.

Este método é mais robusto à presença de *outliers*, pois a média e o desvio padrão são menos sensíveis a valores extremos do que o mínimo e o máximo. No entanto, assume que os dados seguem uma distribuição aproximadamente normal, o que nem sempre é verdade.

2.2. Kruskal-Wallis

Este teste permite verificar quais características são mais discriminativas em relação a uma dada classe, ou seja, quais *features* apresentam diferenças significativas nas suas distribuições entre as diferentes classes.

Para a sua implementação foi utilizada o módulo *kruskal* da biblioteca *scipy* para o cálculo do *H-value*, no entanto o seu cálculo é feito da seguinte forma:

1. É selecionado uma *feature*;
2. Para cada classe do target são atribuídos *ranks* por ordem crescente em relação aos valores da *feature* atual;
3. É calculada a média do *rank* para cada classe da variável target;
4. É calculada a média de todos os *ranks*;
5. É calculado o *H-value* que tem a seguinte formula

$$H = \frac{12}{n(n+1)} \sum_{i=1}^c n_i (R_i - \bar{R})^2$$

Onde n = número total de observações, c é o número de classes da variável target e n_i é o tamanho de cada grupo.

A *feature* que obtiver o *H-value* maior é aquela que melhor discrimina os dados. Isto pode então ser utilizado para *feature selection*.

Como este teste apenas verifica o poder discriminativo de uma característica é necessário descartar *features* que sejam altamente correlacionadas para isso foi feito o uso da matriz de correlação e serão removidas aquelas que apresentam valores muito altos entre si novamente, *feature selection*

2.3. PCA

O *Principal Component Analysis (PCA)* tem como objetivo principal descobrir um novo conjunto reduzido de *features* com menos redundância, com perda mínima de informação, esta preservação de informação é medida em termos de variância. Ele projeta os dados em direções onde os dados apresentam maior variação.

Para isto o PCA segue os seguintes Passos:

1. Normalização dos dados para ter média zero e desvio padrão de 1. Para garantir que a variância seja comparável entre as características
2. Cálculo da matriz de covariância.
3. Cálculo dos *eigenvectors* e *eigenvalues*, pois estes representam as direções no espaço dos dados e a quantidade de variância capturada por cada componente principal, respetivamente.
4. Ordenação dos *eigenvectors* e *eigenvalues* em ordem decrescente

5. Seleção dos componentes principais, foram implementados dois métodos:
- Kaiser criterium*: Selecionar o eigenvalue antes do plot cair para baixo de 1, *Figura 2-1*
 - Scree test*: Selecionar o eigenvalue que estabilizou a curva, *Figura 2-1*

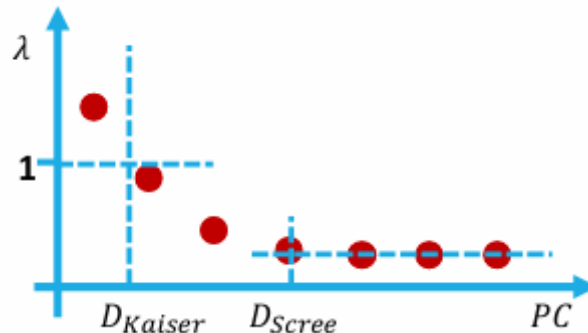


Figura 2-1 - Demonstração de kaiser criterium e scree test

Desta forma serão selecionadas um novo conjunto reduzido de *feature* que se insere no âmbito da *feature reduction/dimensionality reduction*.

2.4. Minimum Distance Classifier

Neste classificador uma amostra é classificada como a classe cuja distribuição conhecida ou estimada mais se assemelha com amostra em questão. Esta medida de semelhança é a distância no espaço de distribuição, ou seja, calcula-se a distância entre a amostra e as representações das diferentes classes (usualmente a médias dos dados delas), atribuindo-se a amostra à classe que estiver mais "próxima".

Este tipo de classificador baseia-se na ideia de que as amostras de uma mesma classe tendem a estar próximas umas das outras no espaço de características, enquanto amostras de classes diferentes tendem a estar mais afastadas. Para isto, é necessário definir métricas de distância. Entre várias existentes as mais comuns são *Euclidean Distance* e *Mahalanobis Distance*:

- *Euclidean Distance*: Mede a distância em "linha reta" entre dois pontos no espaço. E por consequência considera que as distribuições das classes são esféricas.
- *Mahalanobis Distance*: Leva em consideração a correlação entre as características e as variações nas diferentes direções do espaço. Esta é particularmente útil quando as distribuições das classes não são esféricas ou têm escalas diferentes

2.4.1 Euclidean distance discriminant

Como mencionado anteriormente este tipo de discriminante considera que as classes e as suas distribuições são esféricas pela forma como é calculada a sua distância. A distância euclidiana mede a distância "em linha reta" entre dois pontos no espaço de características, sem considerar a correlação entre as variáveis ou as diferenças de escala.

Esta abordagem pode não ser a melhor opção quando as distribuições das classes são alongadas ou têm orientações diferentes, pois ignora a estrutura de covariância dos dados.

Tendo em conta esta informação para a sua implementação é necessário:

1. É calculada a média de cada classe da variável *target* com base nos dados
2. É feita a classificação de novos pontos calculando a distância euclidiana entre as médias da classe. A distância euclidiana é dada pela fórmula:

a. $distancia = \sqrt{\sum (x_1 - x_2)^2}$

2.4.2 Mahalanobis distance discriminant

Ao contrário do discriminante anterior, o *Mahalanobis distance discriminant* leva em consideração as variações nas diferentes direções do espaço. Isto é feito incorporando a matriz de covariação no cálculo da distância, o que permite ajustar a métrica à forma e à orientação da distribuição dos dados.

Esta característica torna o *Mahalanobis distance discriminant* mais flexível e robusto, especialmente quando as classes têm distribuições elípticas ou quando as características estão correlacionadas. No entanto, o cálculo da matriz de covariação pode ser computacionalmente mais exigente.

Tendo em conta esta informação para a sua implementação é necessário:

1. É calculada a média de cada classe da variável *target* com base nos dados
2. É calculada a inversa da matriz de covariação para cada classe
3. É feita a classificação de novos pontos calculando a distância *mahalanobis* entre cada ponto e a média das classes, a distância mahalanobis é dada pela fórmula:

a. $D_M(p, \mu) = \sqrt{(p - \mu)^T C^{-1} (p - \mu)}$

Onde p é o vetor que representa o ponto no espaço, μ é o vetor que representa a média das classes da variável *target*, C é a matriz de covariação.

2.5. Fisher LDA

Enquanto técnicas como o *PCA* (*Principal Component Analysis*) se focam em encontrar uma representação dos dados num subespaço linear de dimensão reduzida que captura as maiores variâncias nos dados, o *Fisher's Linear Discriminant Analysis* (*LDA*) tem como objetivo maximizar a separabilidade entre classes.

O *PCA* não tem em conta a informação das classes, ou seja, não leva em consideração as *labels* dos dados ao determinar as direções de máxima variância. Embora isso seja útil para representação dos dados, as direções de maior variância nem sempre são as melhores para discriminar duas classes distintas.

O *LDA*, por outro lado, procura encontrar uma projeção linear dos dados num subespaço que maximize a separação entre as classes. Para isso, utiliza um critério de otimização baseado na razão de *Fisher* (*Fisher's ratio*), que mede a relação entre a variância entre classes (*inter-class*) e a variância dentro de cada classe (*intra-class*). Ao maximizar esta razão, o *LDA* garante que as projeções dos dados no novo subespaço mantenham as classes tão distantes quanto possível, enquanto minimiza a dispersão dentro de cada classe.

Por esta razão, não faz sentido aplicar métodos como a *PCA* antes de utilizar o classificador *Fisher LDA*. Isto porque a *PCA* representa os dados num novo eixo que maximiza a variância, mas ignora a informação das classes. Como resultado, esta transformação pode dificultar a tarefa do *Fisher*

LDA, que procura encontrar um eixo que maximize a separação entre as classes. Por outras palavras a projeção inicial feita pelo PCA pode destruir as diferenças que existem entre as classes, reduzindo a eficácia do Fisher LDA na discriminação dos dados.

Para a sua implementação foi utilizado o módulo *LDA* da biblioteca *scipy*, no entanto este é implementado da seguinte forma:

1. Calcular as médias das classes da variável target
2. Calcular a matriz de dispersão dentro das classes, que é dada pela seguinte formula:
 - a. $S_w = S_1 + S_2 = \sum_{k=1}^2 (x - m_k)(x - m_k)^T$
3. Calcular o vetor de projeção w , dado pela formula:
 - a. $w = S_w^{-1}(m_1 - m_2)$, m_1 e m_2 são as médias das classes
4. Projetar os dados no novo espaço através de
 - a. $y = w^T x$
5. Calcular o Hiperplano de decisão no novo espaço
 - a. $y = x = \frac{w^T m_1 + w^T m_2}{2}$
6. Classificar dados segundo este hiperplano, *Figura 2-2*

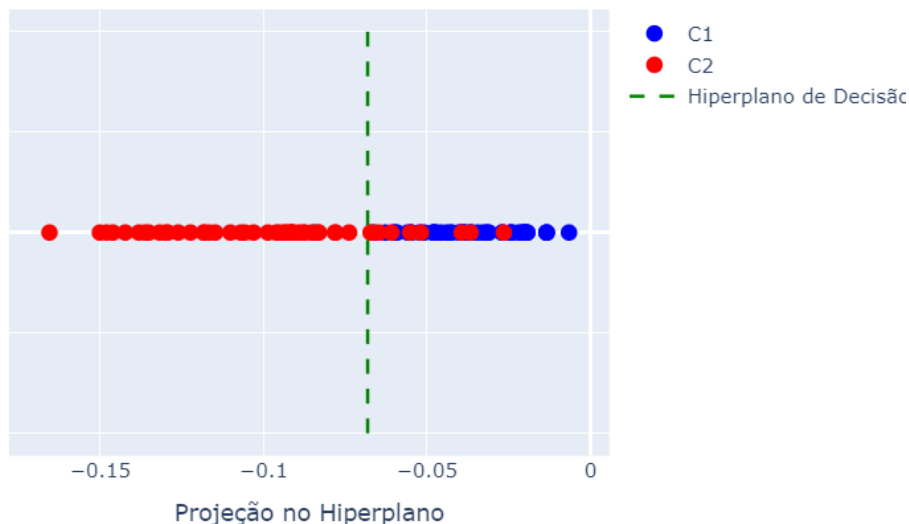


Figura 2-2 - Hiperplano de decisão do Fisher LDA

2.6. Cross Validation

Para os obter dados de avaliação dos modelos mais coerentes foi utilizado *cross-validation*, que é uma técnica que parte o *dataset* em k partições, onde 1 delas é escolhida para teste e as restantes $k-1$ são utilizadas para treino, este processo é repetido k vezes. Isto permite obter resultados médios e portanto mais confiáveis.

3. Resultados

Tendo todas as implementações feitas basta agora fazer os testes para verificar qual combinação de classificadores se melhor comporta.

Foram feitas várias combinações para comparar os modelos, que são as seguintes:

- Modelo + normalização
- Modelo + normalização + *kruskal wallis*
- Modelo + normalização + *kruskal wallis* + *PCA (kaiser/scree)*

Estas têm como objetivo tentar minimizar o erro na classificação de novos dados e os resultados estarão nos subcapítulos a baixo.

3.1. Kruskal Wallis

A aplicação do teste de *kruskal wallis*, que nos permitirá verificar quais características melhor discriminam a variável *target*.

E os resultados deste teste podem ser vistos na seguinte *Figura 3-1*.

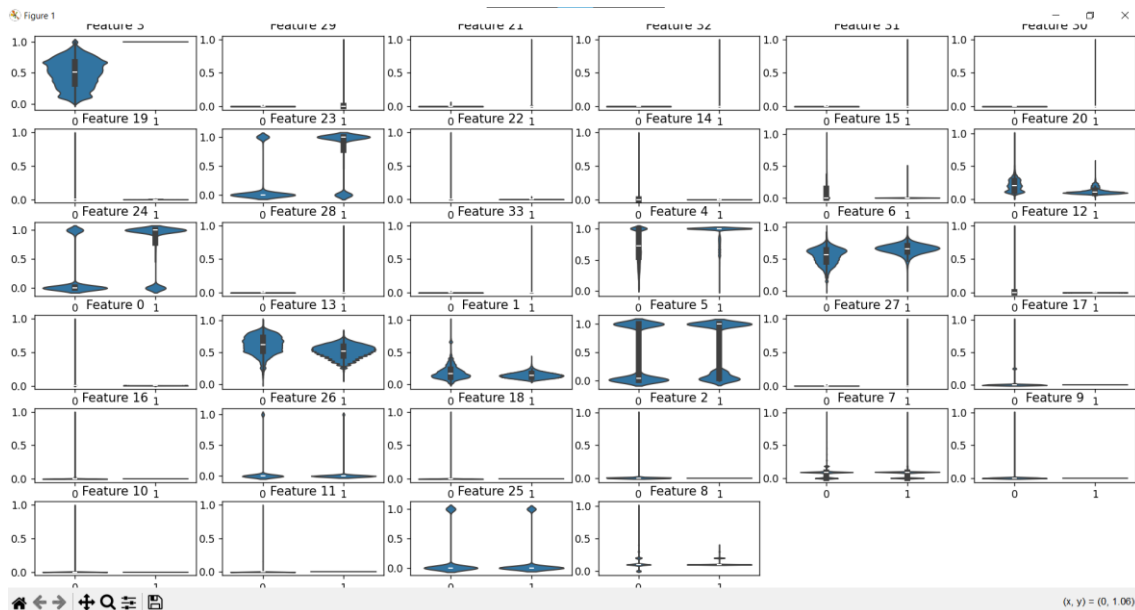


Figura 3-1 - Resultados Kruskal wallis

Como é possível ver na figura acima existem características que discriminam bem as classes do target e outras que simplesmente não fazem qualquer diferença sendo a que melhor discrimina a *feature 3*, então devem ser mantidas todas as *features* antes da *feature 13*, os resultados dos valores *h* estão na seguinte *Figura 3-2*, ao visualizar estes valores é possível verificar que a escolha de manter apenas as *features* antes da 13 é suportada pois há um grande salto no *H-value* da *feature 1* para a 13.

```
Features ranked by Kruskal-Wallis H statistic:  
Feature 3: H = 210584.3904  
Feature 29: H = 167374.1057  
Feature 21: H = 166758.9981  
Feature 32: H = 162123.5791  
Feature 31: H = 159715.8940  
Feature 30: H = 152179.2699  
Feature 19: H = 88465.3387  
Feature 23: H = 76545.3020  
Feature 22: H = 69335.6563  
Feature 14: H = 68961.0704  
Feature 15: H = 67811.9838  
Feature 20: H = 65590.2345  
Feature 24: H = 65028.7266  
Feature 28: H = 63975.0585  
Feature 33: H = 62589.4245  
Feature 4: H = 52089.1064  
Feature 6: H = 49701.3827  
Feature 12: H = 40733.6881  
Feature 0: H = 37479.0548  
Feature 13: H = 32077.0349  
Feature 1: H = 10750.9302  
Feature 5: H = 9216.9150  
Feature 27: H = 8777.7796  
Feature 17: H = 8406.9712  
Feature 16: H = 7383.0370  
Feature 26: H = 1378.5920  
Feature 18: H = 1189.3858  
Feature 2: H = 854.5976  
Feature 7: H = 698.2786  
Feature 9: H = 649.2325  
Feature 10: H = 649.2317  
Feature 11: H = 649.2316  
Feature 25: H = 508.8842  
Feature 8: H = 263.1706
```

Figura 3-2 - Ranking of H-value

Através da matriz de correlação é possível visualizar *features* que são altamente correlacionadas e portanto redundantes para os classificadores, e então removê-las por não acrescentarem nada à previsão de novos dados. Ao visualizar a *Figura 3-3* é possível verificar que existem sim características bastante correlacionadas e então irão ser removidas aquelas com um nível de correlação superior a 80%, ou seja, as *features* 12, 14, 16, 19 e 23

Figure 1

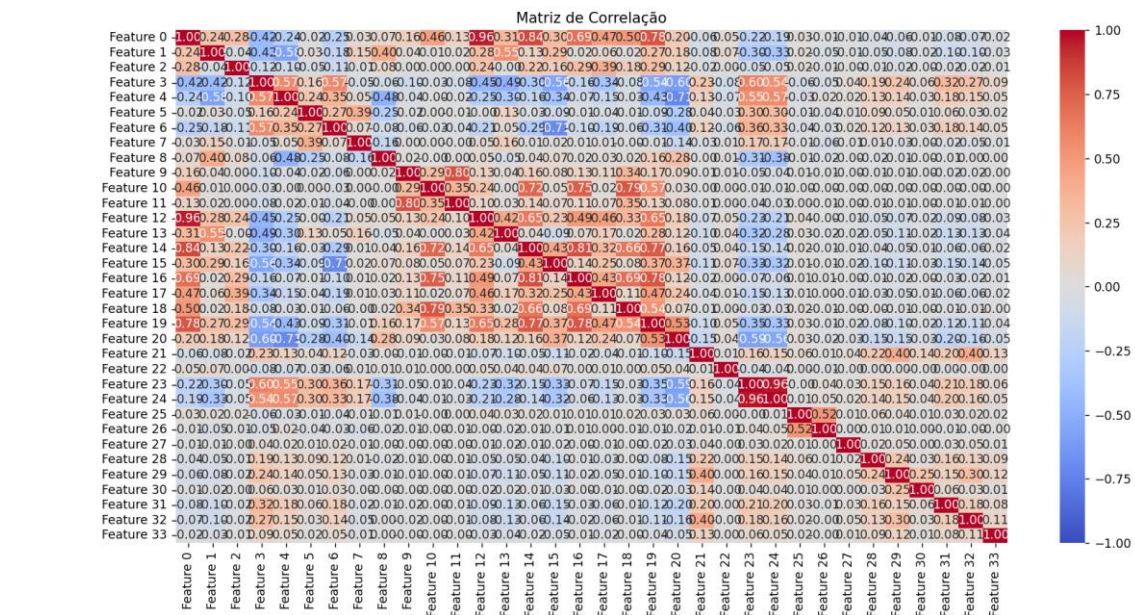


Figura 3-3 - Matriz de correlação

Fazendo agora uma pequena comparação do antes e após a seleção e remoção de features através de estes dois resultados é possível verificar que há uma leve melhoria nos resultados de classificação visível nas Figura 3-4 e Figura 3-5.

Mean accuracy (cross-validation): 0.8814
Sensitivity (Recall): 0.9502
Specificity: 0.7895
F-measure (F1-score): 0.9017

Figura 3-4 - Antes da remoção de variáveis altamente correlacionadas

Metrics
Mean accuracy (cross-validation): 0.8900
Sensitivity (Recall): 0.9730
Specificity: 0.7792
F-measure (F1-score): 0.9102

Figura 3-5 - Após a remoção de variáveis altamente correlacionadas

3.2. Minimum Distance classifier

3.2.1 Euclidean Distance Discriminant

O primeiro teste será a classificação de dados sem qualquer tipo de *feature selection* ou *reduction*, para verificar como os modelos se comportam sem o devido tratamento de dados prévio.

Novamente é de notar que todas as features categóricas e binárias foram removidas antes de quaisquer testes devido aos modelos aprendidos não considerarem este tipo de dados.

Os resultados podem então ser observados na *Figura 3-6*.

```
Mean accuracy (cross-validation): 0.5338  
Sensitivity (Recall): 0.8736  
Specificity: 0.0799  
F-measure (F1-score): 0.6819
```

Figura 3-6 - Resultados teste MDC-EDD sem tratamento de dados

Tendo em conta estes resultados é possível concluir o seguinte:

- *Accuracy*: com um resultado de 53.38% isto indica que o modelo acertou cerca de metade das suas previsões. Isto sugere que o modelo não está a ter uma boa performance
- *Sensitivity*: com um resultado de 87.36% isto indica que a maioria dos exemplos de classe positiva o modelo está a acertar
- *Specificity*: com um resultado de 7.99% isto indica que o modelo classifica como positivos a grande maioria de casos negativos
- *F-score*: Com um resultado de 68.19% isto indica que o modelo tem um desempenho moderado. Apesar de tudo é um resultado enganoso por causa dos resultados obtidos nos testes *sensitivity* e *specificity*.

O próximo teste será feito com o classificador e com uma normalização através de *z-score*, e os resultados poderão ser visualizados abaixo na *Figura 3-7*.

```
Mean accuracy (cross-validation): 0.8814  
Sensitivity (Recall): 0.9502  
Specificity: 0.7895  
F-measure (F1-score): 0.9017
```

Figura 3-7 - Resultados teste MDC-EDD com normalização z-score

Tendo em conta os resultados é possível concluir o seguinte:

- *Accuracy*: com um resultado de 88.14% isto indica que o modelo acertou grande parte das previsões que fez;
- *Sensitivity*: com um resultado de 95.02% isto indica que praticamente todos os exemplos de classe positiva o modelo está a acertar;
- *Specificity*: com um resultado de 78.95% isto indica que o modelo consegue classificar bem a classe negativa;
- *F-score*: Com um resultado de 90.17% isto indica que o modelo tem um desempenho muito bom. Apesar de tudo continua a ser um resultado um pouco *biased* pela *sensitivity*, pois esta é muito alta e a classificação de casos negativos não é tão boa quanto ela

Tendo em conta estes novos resultados a normalização através de *z-score* teve um impacto muito positivo no classificador, que conseguiu discriminar as classes de uma melhor forma.

Isto acontece, pois, a distância euclidiana é muito sensível à escala dos dados. Então se uma característica tem valores muito maiores que outras elas acabam por contribuir mais para o valor final da distância, fazendo com que a classificação dos dados seja errada, a normalização permitiu corrigir isso.

O próximo teste será com para além da normalização dos dados a aplicação do teste de *kruskal wallis*, que nos permitirá verificar quais características melhor discriminam a variável *target*.

E os resultados deste teste são visíveis na *Figura 3-8*

```
Metrics  
Mean accuracy (cross-validation): 0.8920  
Sensitivity (Recall): 0.9696  
Specificity: 0.7883  
F-measure (F1-score): 0.9113
```

Figura 3-8 - Resultados teste MDC-EDD com normalização e kruskal wallis

Com estes resultados é possível concluir o seguinte:

Os resultados foram bastante idênticos ao anterior sendo apenas ligeiramente melhores, isto significa que ao remover as variáveis altamente correlacionadas não foi perdida qualquer informação, mas também não foi ganha muito mais.

Em seguida e o último teste será a combinação de todos os métodos, ou seja, normalização, *kruskal wallis* e *PCA kaiser e scree*, e estes resultados estão presentes nas *Figura 3-9* e *Figura 3-11*, o número de componentes principais escolhidos para o *scree test* foi com base no *plot* na *Figura 3-10* e foi escolhido 11.

```
Mean accuracy (cross-validation): 0.8750  
Sensitivity (Recall): 0.9499  
Specificity: 0.7748  
F-measure (F1-score): 0.8969
```

Figura 3-9 - Resultados teste MDC-EDD com normalização, kruskal wallis e PCA (kaiser)

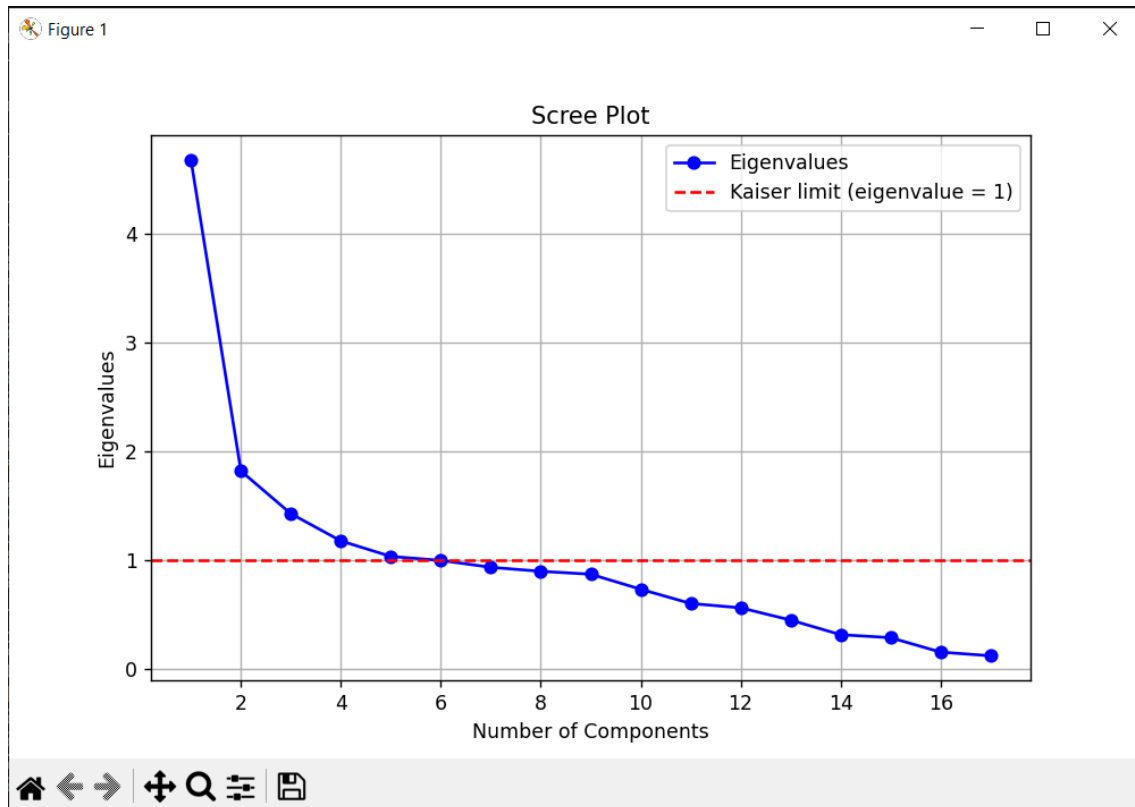


Figura 3-10 - Plot para scree test

```
Mean accuracy (cross-validation): 0.8862
Sensitivity (Recall): 0.9632
Specificity: 0.7833
F-measure (F1-score): 0.9065
```

Figura 3-11 - Resultados teste MDC-EDD com normalização, kruskal wallis e PCA (scree)

Isto pode acontecer devido à redução de dimensionalidade por parte do PCA algumas informações discriminativas podem estar a ser perdidas neste processo devido ao seu funcionamento e limitações. Por exemplo se *features* discriminativas não estiverem alinhadas com as direções de maior variância, o PCA pode reduzir a capacidade do classificador de distinguir as classes.

3.2.2 Mahalanobis Distance Discriminant

O primeiro teste será a classificação de dados sem qualquer tipo de *feature selection* ou *reduction*, para verificar como os modelos se comportam sem o devido tratamento de dados prévio.

Este teste não foi possível fazer, após uma pesquisa foi possível concluir que quando os dados não são normalizados *features* com escalas muito diferentes podem levar a instabilidade numérica e assim impossibilitando o cálculo da matriz de covariação necessária para a distância mahalanobis.

Tendo em conta o anterior será feito o teste seguinte que é o uso do classificador com os dados normalizados,

Isto é possível verificar na *Figura 3-12*.

```
Mean accuracy (cross-validation): 0.9976  
Sensitivity (Recall): 0.9968  
Specificity: 0.9988  
F-measure (F1-score): 0.9979
```

Figura 3-12 – Resultados teste MDC-MDD com normalização z-score

É possível verificar resultados extremamente altos para todas as métricas o que mostra que o classificador é extremamente discriminante das classes presentes no *dataset*.

Passando então para a aplicação de *kruskal wallis* ao modelo com os dados normalizados, os resultados desta combinação podem ser vistos na *Figura 3-13*.

```
Mean accuracy (cross-validation): 0.9967  
Sensitivity (Recall): 0.9976  
Specificity: 0.9954  
F-measure (F1-score): 0.9971
```

Figura 3-13 - Resultados teste MDC-MDD com normalização z-score e kruskal wallis

Os resultados apresentados por este modelo continuam a ser extremamente bons apesar de ter havido uma ligeira diminuição na performance do mesmo, isto pode significar que apesar de serem removidas *features* que não são tão discriminativas das classes da variável *target*, está se a perder alguma informação e por isso o modelo apresenta uma ligeira maior dificuldade a classificar novos dados.

Passando então para a última combinação em que adicionamos ao último modelo também o *PCA* com *kaiser* e *scree test*, e os resultados deste novo teste estão nas *Figura 3-14* e *Figura 3-16*, já o número de componentes principais escolhidos estão na *Figura 3-15* foi então escolhido 8.

```
Mean accuracy (cross-validation): 0.9604  
Sensitivity (Recall): 0.9965  
Specificity: 0.9123  
F-measure (F1-score): 0.9665
```

Figura 3-14 - Resultados teste MDC-MDD com normalização z-score, kruskal wallis e PCA (kaiser)

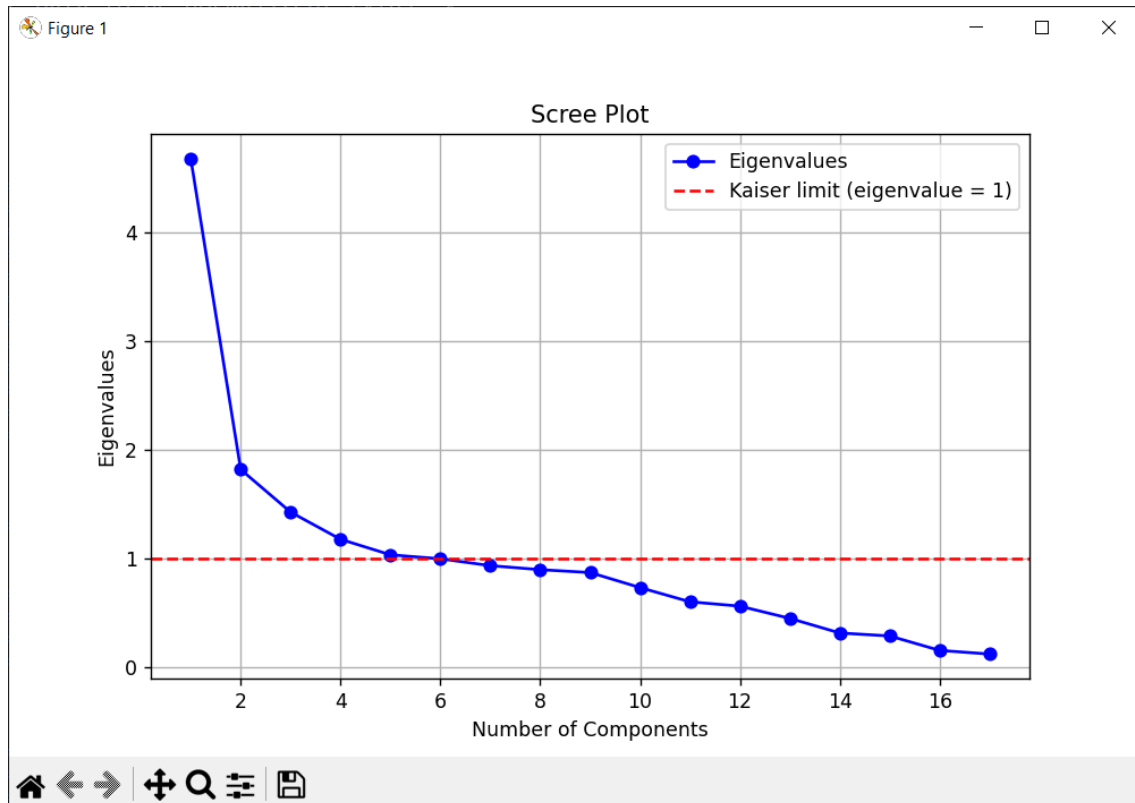


Figura 3-15 - Plot para scree test

```
Mean accuracy (cross-validation): 0.9618
Sensitivity (Recall): 0.9984
Specificity: 0.9131
F-measure (F1-score): 0.9677
```

Figura 3-16 - Resultados teste MDC-MDD com normalização z-score, *kruskal wallis* e PCA (scree)

Os resultados deste teste foram apesar de bastante satisfatórios piores ainda que o teste com apenas normalização e *kruskal wallis*, isto porque a redução de dimensionalidade por parte do *PCA* pode levar à perda de algumas informações discriminativas. Por exemplo se *features* discriminativas não estiverem alinhadas com as direções de maior variância, o *PCA* pode reduzir a capacidade do classificador de distinguir as classes.

3.3. Fisher LDA

O primeiro teste que seria realizado para este classificador será sem qualquer tipo de tratamento de dados, no entanto na biblioteca que está a ser usada isto já é feito inerentemente para o cálculo das matrizes de dispersão. Os resultados estão na *Figura 3-17* abaixo.

```
Mean accuracy (cross-validation): 0.9682
Sensitivity (Recall): 1.0000
Specificity: 0.9258
F-measure (F1-score): 0.9730
```

Figura 3-17 - Resultados teste Fisher LDA com normalização

Com a normalização os resultados apresentados pelo Fisher LDA já são excelentes. Assim como os 2 modelos anteriores, o que significa que as *features* escolhidas e a normalização permitem que o modelo discrimine bem os dados.

O próximo teste será a aplicação do teste *kruskal wallis* ao modelo anterior com a normalização dos dados, e os resultados podem ser verificados na *Figura 3-18*.

```
Mean accuracy (cross-validation): 0.9650  
Sensitivity (Recall): 1.0000  
Specificity: 0.9182  
F-measure (F1-score): 0.9703
```

Figura 3-18 - Resultados teste Fisher LDA com normalização e kruskal wallis

Os resultados apresentados por este modelo continuam a ser extremamente bons apesar de ter havido uma ligeira diminuição na performance do mesmo, isto pode significar que apesar de serem removidas *features* que não são tão discriminativas das classes da variável *target*, está se a perder alguma informação e por isso o modelo apresenta uma ligeira maior dificuldade a classificar novos dados.

O próximo teste seria a implementação do PCA ao modelo anterior, no entanto como explicado no capítulo 2.5, a aplicação do PCA seguida de Fisher LDA não funciona, pois, a projeção inicial feita pelo PCA pode destruir as diferenças que existem entre as classes, reduzindo a eficácia do Fisher LDA na discriminação dos dados. Isto pode mesmo ser visto na *Figura 3-19*, em que o PCA foi implementado antes da classificação *fisher*

```
Mean accuracy (cross-validation): 0.8684  
Sensitivity (Recall): 0.9399  
Specificity: 0.7730  
F-measure (F1-score): 0.8911
```

Figura 3-19 - Resultados de Fisher LDA com PCA

4. Discussão e Conclusão

Tendo em conta todos os resultados obtidos na secção de resultados é possível concluir que o modelo que apresenta melhores resultados médios em resultados nunca antes visto é a implementação do *Minimum Distance Classifier* com *Mahalanobis distance classifier*, obtendo um F1-score médio de 99.79%.

É possível ainda concluir que a implementação do teste *Kruskal wallis* e do *PCA* apesar de serem ferramentas poderosas nem sempre são úteis principalmente quando as *features* já tiveram uma seleção minuciosa, como no nosso caso após a remoção das *features* irrelevantes e redundantes, o conjunto de dados já estava bem estruturado e com *features* informativas. Por isso, a aplicação do *Kruskal-Wallis* e do *PCA* resultou alguma perda de informação.

Foi também possível concluir que a implementação do *PCA* antes da aplicação do classificador *Fisher LDA* não faz qualquer sentido pois o *PCA* projeta os dados num novo eixo, focando-se nas direções de maior variância. No entanto, essas direções nem sempre estão alinhadas com as que maximizam a separação entre as classes, que é o objetivo do *Fisher LDA*. Como resultado, o *PCA* pode fazer com que os dados percam informação importante para a discriminação das classes, diminuindo a eficácia do *Fisher LDA* na classificação.

Foi possível também verificar a importância da normalização dos dados e *feature selection* (remoção de *features* categóricas e binárias e *features* redundantes). Pois aplicando estes dois simples métodos foi possível verificar uma melhoria enorme na classificação de novos dados por parte dos classificadores.

5. Referências

- [1] Wacker, A G, and D A Landgrebe. 'Minimum Distance Classification in Remote Sensing', n.d.
- [2] 'Fisher Linear Discriminant - an Overview | ScienceDirect Topics'. Accessed 21 March 2025. <https://www.sciencedirect.com/topics/engineering/fisher-linear-discriminant>.
- [3] Alexander. 'Mahalanobis Distance: Simple Definition, Examples'. Statistics How To, 8 May 2024. <https://www.statisticshowto.com/mahalanobis-distance/>.
- [4] GeeksforGeeks. 'Euclidean Distance | Formula, Derivation & Solved Examples', 18:50:03+00:00. <https://www.geeksforgeeks.org/euclidean-distance/>.
- [5] Wei, Dagang. 'Demystifying Machine Learning: Normalization'. *Medium* (blog), 12 February 2025. <https://medium.com/@weidagang/demystifying-machine-learning-normalization-0cdb8b281234>.
- [6] 'T-Test, Chi-Square, ANOVA, Regression, Correlation...' Accessed 21 March 2025. <https://datatab.net/tutorial/kruskal-wallis-test>.