INSTITUTO FEDERAL
Ceará
Campus Fortaleza

Programa de Pós-graduação em Engenharia deTelecomunicações
PPGET - IFCE

Relatório - Reconhecimento de Padrões 2024.1

Trabalho 1 - KNN e DMC

03/04/2024

- Professor: Ajalmar Rêgo da Rocha Neto
- Aluna: Gabriela de Carvalho Barros Bezerra
- Código: Repositório do Github

1. Introdução

Este trabalho consistiu na implementação dos modelos KNN e DMC utilizando python, e aplicação destes nos cojuntos Iris, Coluna 2D, Coluna 3D, e Artificial I.

Foram feitos 8 experimentos principais utilizando dois modelos (knn e dmc) para cada *conjunto de dados*, e 74 experimentos adicionais para cada par de características em cada conjunto de dados, e cada modelo. Cada experimento principal consistiu em 20 realizações de uma simulação computacional que se deu em 6 fases:

1. Leitura do *conjunto de dados*.
2. Separação do *conjunto de dados* em *conjunto de treinamento* e *conjunto de teste*, com possível visualização dos subconjuntos obtidos.
3. Treinamento do *modelo* com o *conjunto de treinamento*.
4. Predição do *conjunto de teste* com o *modelo*.
5. Cáculo das *métricas*.

Como resultado, foram obtidas visualizações dos conjuntos de treinamento e teste para cada *conjunto de dados*, visualizações das *superfícies de decisão* para cada par de características de todos os conjuntos de dados (somente um par de cada exemplo mostrado no relatório), a métricas de *acurácia* e *desvio padrão* para cada modelo em cada *conjunto de dados*, e as *matrizes de confusão* da pior realização de cada modelo em cada *conjunto de dados*.

2. Metodologia

Durante a implementação do trabalho, foram utilizadas as bibliotecas:

- *pandas* para leitura e separação dos *conjuntos de dados* (fases 1 e 2) e para construção da matriz de confusão (fase 5).
- *numpy* para realização de cálculos e manipulações vetorais (fases 3, 4, e 5).
- *matplotlib* para visualização dos conjuntos de treinamento e teste (fase 2) e exibição das superfícies de decisão no experimentos adicionais.

2.1. Implementação dos Modelos

Na implementação do trabalho, cada modelo foi abstraído como uma classe com métodos *fit* para treinamento no *conjunto de treinamento* e *predict* para predição do *conjunto de teste*. O modelo KNN foi implementado com o hiperparâmetro K, estimado através do cálculo da raiz quadrada da quantidade de pontos no conjunto de dados utilizado no treinamento. O modelo DMC não foi implementado com hiperparâmetros. Para ambos modelos, foi utilizada a distância euclidiana.

2.2. Experimentos principais

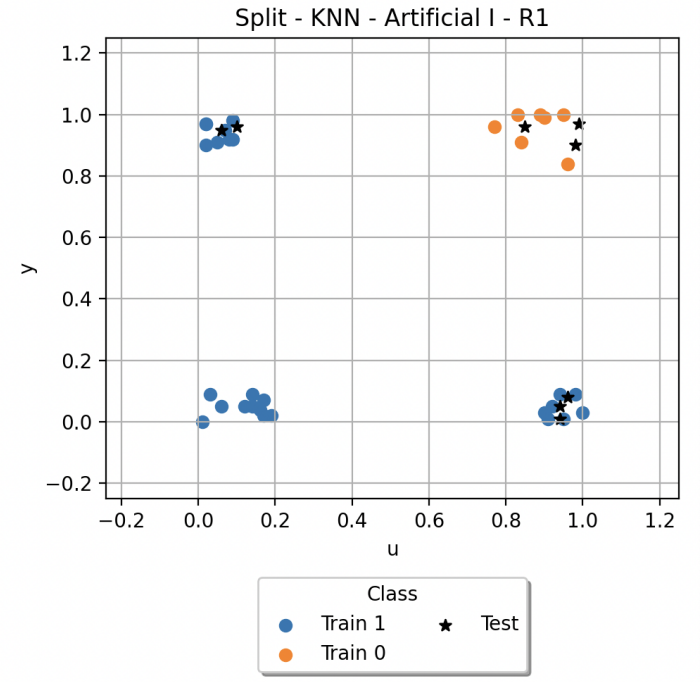
2.2.1. Leitura dos conjuntos de dados

A leitura do *conjunto de dados* foi feita utilizando a função *read_csv* da biblioteca *pandas*. Durante as visualizações, foi identificado e removido um *outlier* nos conjuntos de dados Coluna 2D e Coluna 3D.

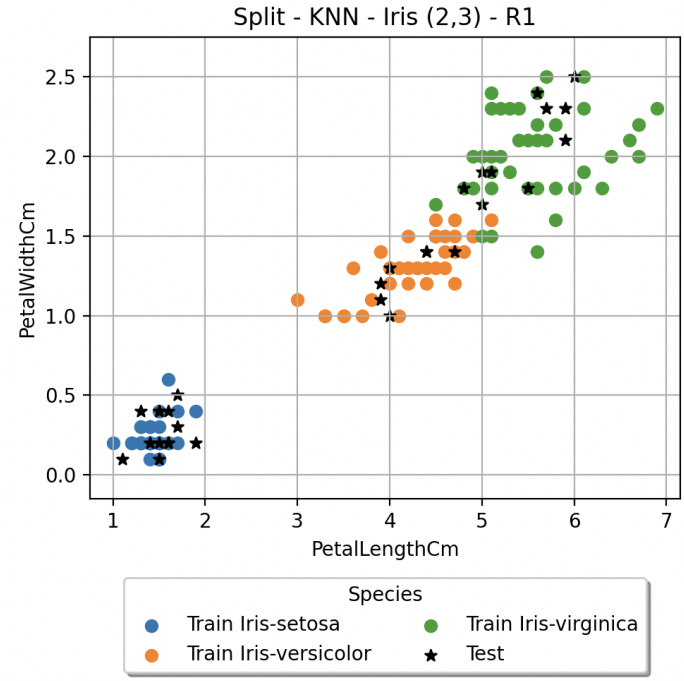
2.2.2 Separação do conjunto de dados em treinamento e teste

Para separação do *conjunto de dados*, foi utilizado um método conhecido como *Holdout*, que embaralha os pontos de maneira aleatória e divide o conjunto de dados proporcionalmente em conjuntos de treinamento e teste. O embaralhamento foi implementado utilizando a função *sample* da biblioteca *pandas*. Foi parametrizado uma porcentagem para a divisão do conjunto de dados em os conjuntos de treinamento e teste. Por padrão, foi utilizado 80% para *conjunto de treinamento* e 20% para *conjunto de teste*.

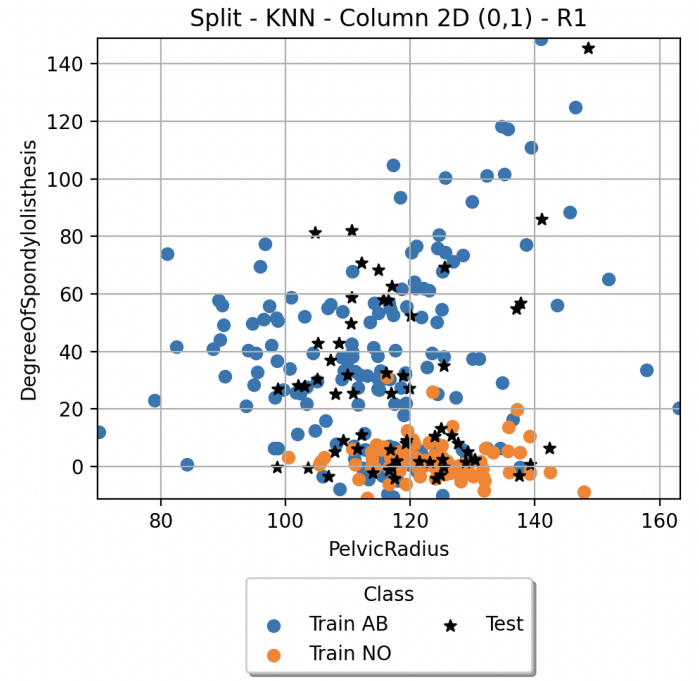
Visualização gráfica após dos conjuntos de treinamento e teste após a separação do *conjunto de dados* Artificial I em uma realização.



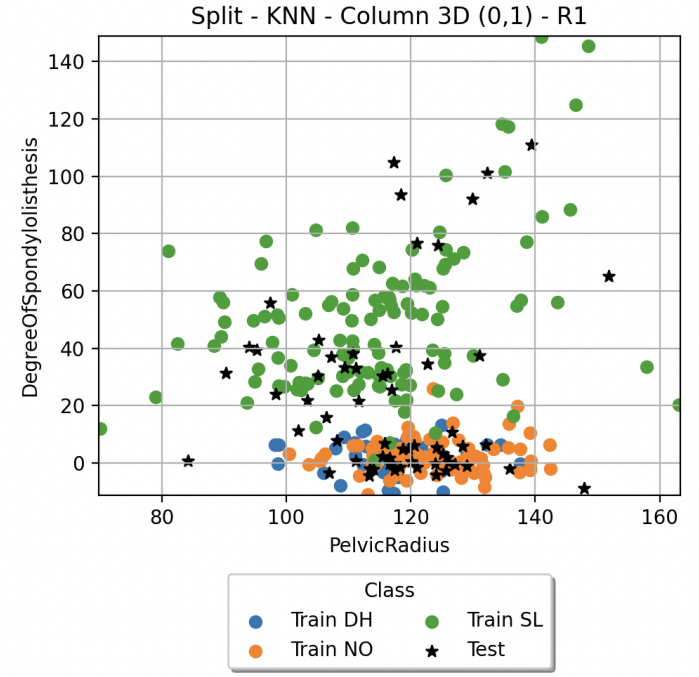
Visualização gráfica após dos conjuntos de treinamento e teste após a separação do *conjunto de dados* Iris em uma realização.



Visualização gráfica após dos conjuntos de treinamento e teste após a separação do *conjunto de dados* Coluna 2D em uma realização.



Visualização gráfica após dos conjuntos de treinamento e teste após a separação do *conjunto de dados* Coluna 3D em uma realização.



2.2.3. Treinamento do modelo

KNN

A implementação da função *fit* do modelo KNN consiste em simplesmente armazenar os pontos do *conjunto de treinamento* em memória.

DMC

Durante a implementação da função *fit*, o modelo DMC calcula os valores médios de cada característica para cada classe no conjunto de treinamento. Esses valores médios são denomidados *centróides*, e são armazenados para posterior uso na etapa de predição.

2.2.4. Predição do modelo

KNN

A implementação da função *predict* consiste em, para cada ponto do *conjunto de teste*:

1. Calcular a *distância euclidiana* entre o ponto de teste e todos os pontos de treinamento.
2. Ordenar as distâncias encontradas e identificar os k-vizinhos (pontos de treinamento) mais próximos.
3. Atribuir a classe mais frequente entre os k-vizinhos mais próximos ao ponto de teste.

Esse processo é repetido para todos os pontos do *conjunto de teste*, e as classes que resultam da predição são retornadas como saída.

DMC

A implementação da função *predict* consiste em, para cada ponto do conjunto de teste:

1. Calcular a *distância euclidiana* entre o ponto de teste e os *centróides* de cada classe previamente calculados durante o treinamento.
2. Identificar o *centróide* mais próximo.
3. Atribuir a classe do *centróide* mais próximo ao ponto de teste.

Esse processo é repetido para todos os pontos do *conjunto de teste*, e as classes que resultam da predição são retornadas como saída.

2.2.5. Cálculo das métricas

As métricas foram calculadas através da construção da *matriz de confusão* seguida do cômputo da *taxa de acerto* e *desvio padrão* a partir das predições realizadas.

Ao final de cada experimento foi exibida a *matriz de confusão* para a realização que obteve as piores métricas (acurácia geral mais baixa), no intuíto de entender como o modelo está se comportando no pior caso.

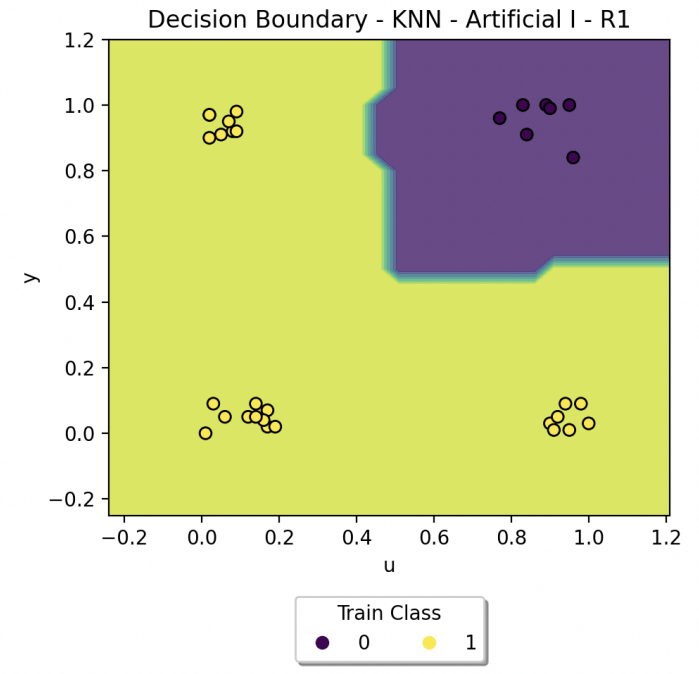
2.3. Experimentos adicionais

Também foram realizados experimentos adicionais para cada par de características em cada conjunto de dados, e cada modelo, totalizando 74 experimentos adicionais. No melhor caso de cada experimento adicional, foi exibida a *superfície de decisão* no intuído de entender qual par de características separa melhor cada conjunto de dados.

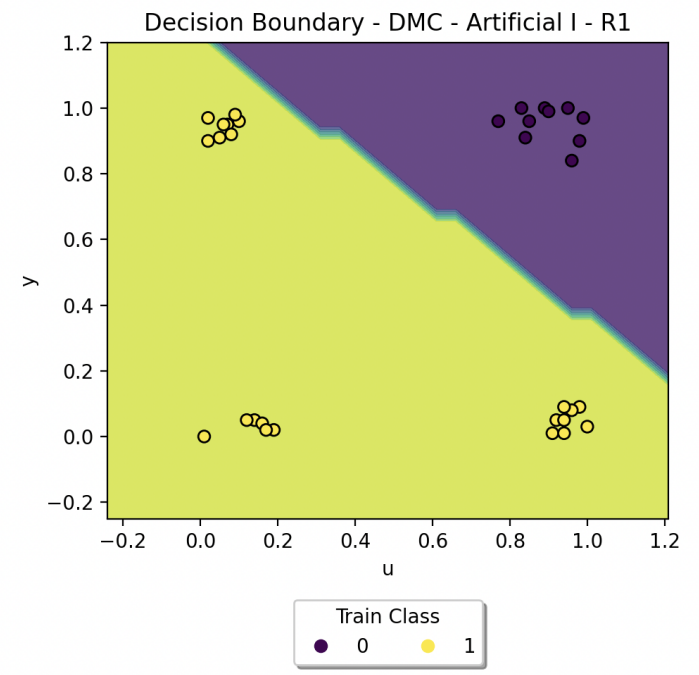
3. Resultados e Discussões

3.1 Conjunto de dados Artificial I

O conjunto de dados Artificial I é linearmente separável, como podemos ver pela superfície de decisão com as únicas duas características do *conjuntos de dados* Artificial I para o KNN.



O conjunto de dados Artificial I é linearmente separável, como podemos ver pela superfície de decisão com as únicas duas características do *conjuntos de dados* Artificial I para o DMC.



Métricas a partir das taxas de acerto das 20 realizações

Acurácia

Classe	KNN	DMC
All	1.00	1.00
0.0	1.00	1.00
1.0	1.00	1.00

Desvio Padrão

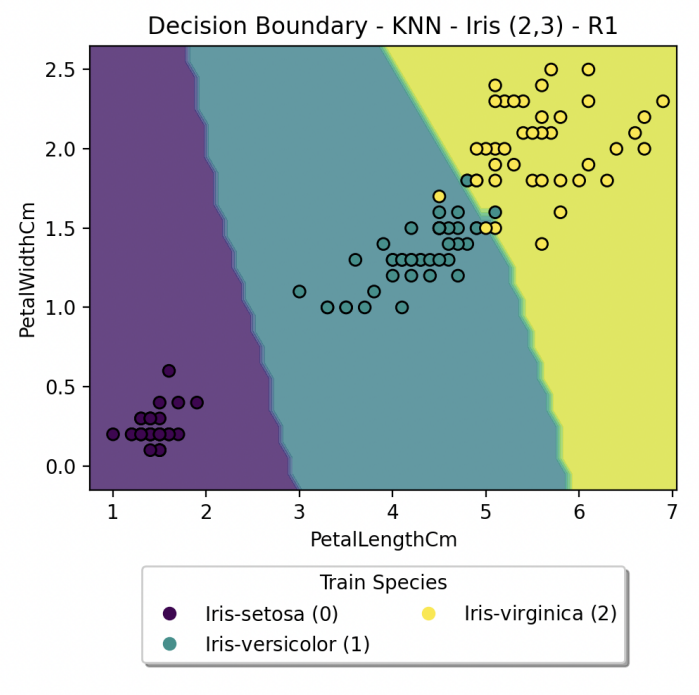
Classe	KNN	DMC
All	0.00	0.00
0.0	0.00	0.00
1.0	0.00	0.00

Matriz de Confusão de ambos DMC e KNN

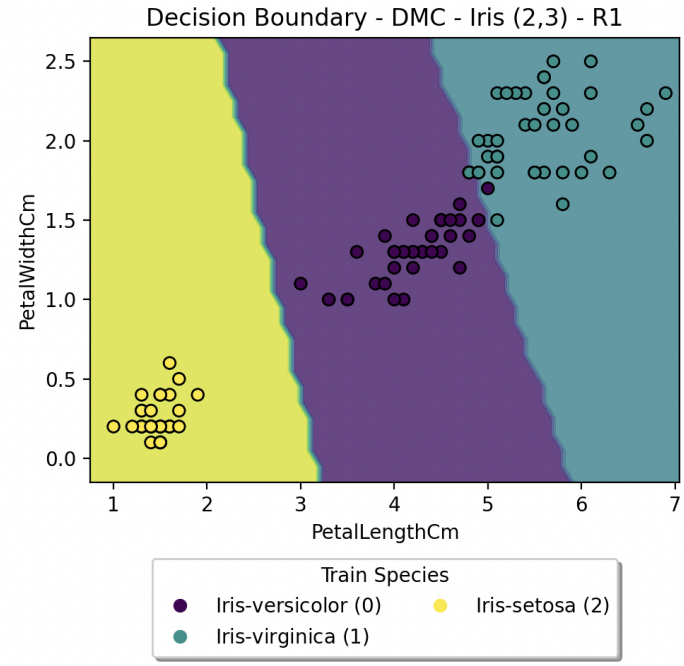
```
Predicted  0.0  1.0
True
0.0          4    0
1.0          0    4
```

3.2. Iris

O conjunto de dados Iris parece ser linearmente separável, como podemos ver pela superfície de decisão com as duas características que melhor separam as classes do *conjuntos de dados* Iris para o KNN.



O conjunto de dados Iris parece ser linearmente separável, como podemos ver pela superfície de decisão com as duas características que melhor separam as classes do *conjuntos de dados* Iris para o DMC.



Métricas a partir das taxas de acerto das 20 realizações

Acurácia

Classe	KNN	DMC
All	0.96	0.93
Iris-setosa	1.00	1.00
Iris-versicolor	0.94	0.89
Iris-virginica	0.93	0.92

Desvio Padrão

Classe	KNN	DMC
All	0.03	0.03

Classe	KNN	DMC
Iris-setosa	0.00	0.00
Iris-versicolor	0.07	0.09
Iris-virginica	0.07	0.08

Matriz de Confusão KNN (realização com pior acurácia)

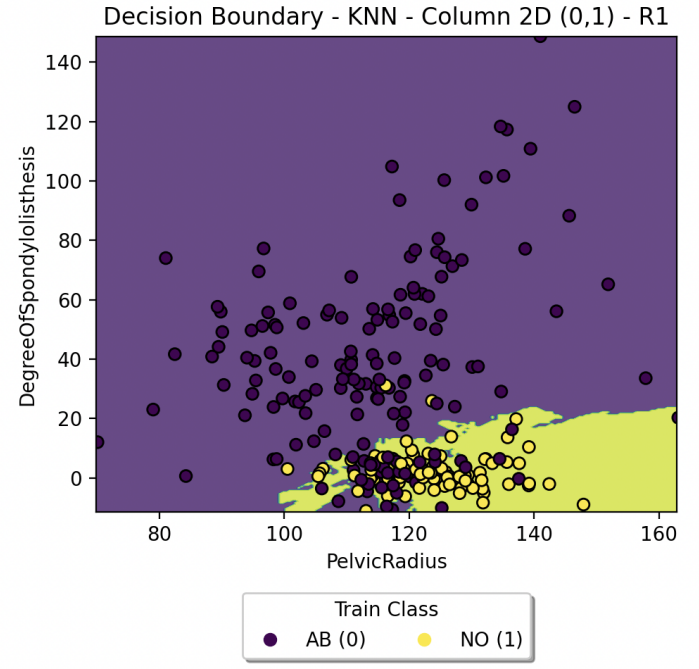
```
Predicted        Iris-setosa  Iris-versicolor  Iris-virginica
True
Iris-setosa                6                0               0
Iris-versicolor            0                9               1
Iris-virginica             0                1              13
```

Matriz de Confusão DMC (realização com pior acurácia)

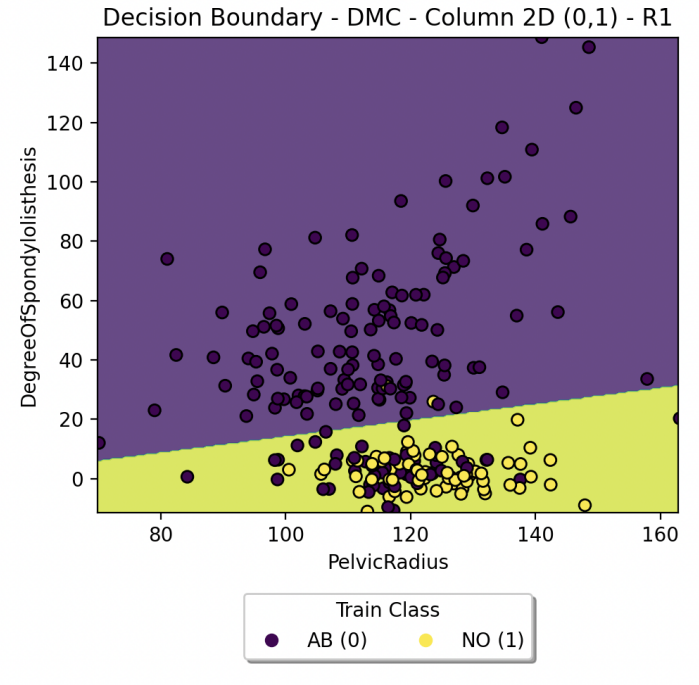
```
Predicted        Iris-setosa  Iris-versicolor  Iris-virginica
True
Iris-setosa               11                0               0
Iris-versicolor            0                8               1
Iris-virginica             0                2               8
```

3.3. Coluna 2D

Superfície de decisão com as duas características que melhor separam as classes do *conjuntos de dados* Coluna 2D para o KNN.



Superfície de decisão com as duas características que melhor separam as classes do *conjuntos de dados* Coluna 2D para o DMC.



Métricas a partir das taxas de acerto das 20 realizações

Acurácia

Classe	KNN
All	0.85
AB	0.90
NO	0.77

Desvio Padrão

Classe	KNN	DMC
All	0.03	0.05
AB	0.05	0.04
NO	0.11	0.08

Matriz de Confusão KNN (realização com pior acurácia)

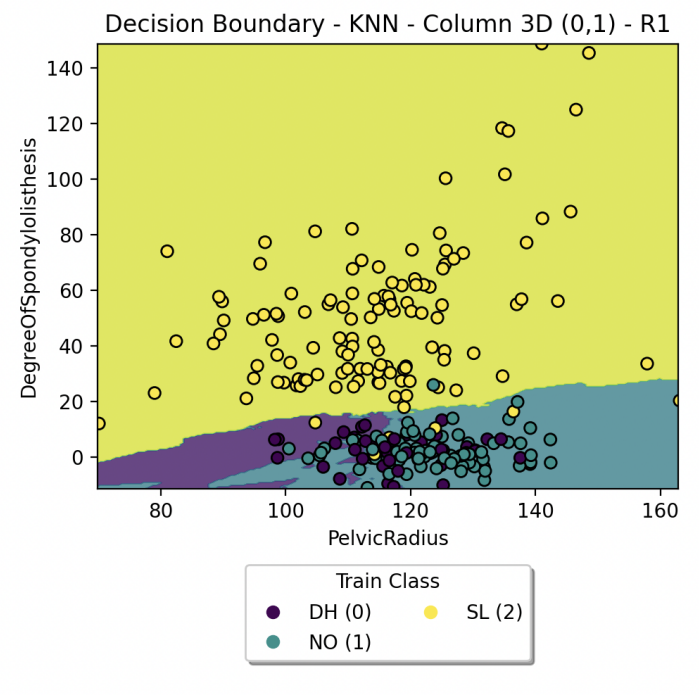
```
Predicted  AB  NO
True
AB         39   4
NO          4  15
```

Matriz de Confusão DMC (realização com pior acurácia)

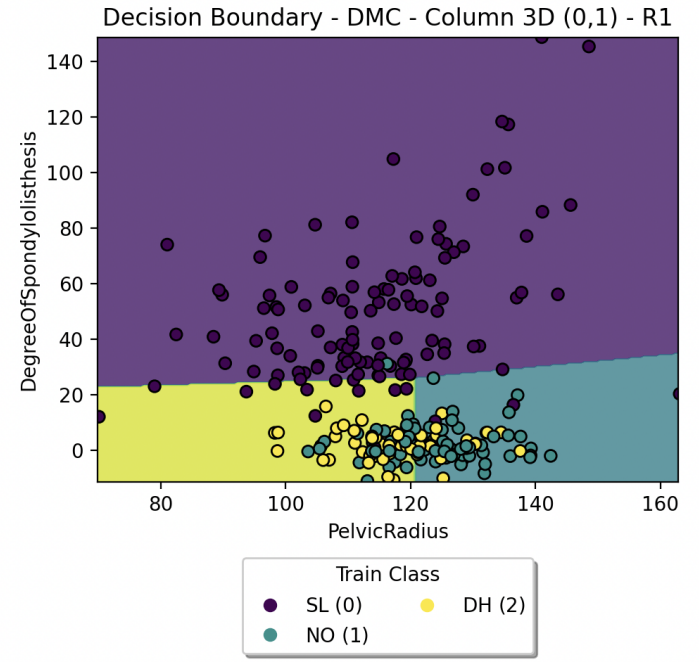
```
Predicted  AB  NO
True
AB         29   2
NO         11  20
```

3.4. Coluna 3D

O conjunto de dados Coluna 3D não é um problema linearmente separável, portanto não foi possível encontrar uma superfície de decisão com as duas características que separam bem as classes do *conjuntos de dados* Coluna 3D para o KNN.



O conjunto de dados Coluna 3D não é um problema linearmente separável, portanto não foi possível encontrar uma superfície de decisão com as duas características que separam bem as classes do *conjuntos de dados* Coluna 3D para o DMC.



Métricas a partir das taxas de acerto das 20 realizações

Acurácia

Classe	KNN	DMC
All	0.82	0.76
DH	0.76	0.55
NO	0.68	0.69
SL	0.97	0.97

Desvio Padrão

Classe	KNN	DMC
All	0.04	0.05

Classe	KNN	DMC
DH	0.15	0.10
NO	0.09	0.06
SL	0.03	0.03

Matriz de Confusão KNN (realização com pior acurácia)

```
Predicted  DH  NO  SL
True
DH         11   1   1
NO          6  18   2
SL          0   1  22
```

Matriz de Confusão DMC (realização com pior acurácia)

```
Predicted  DH  NO  SL
True
DH          9   7   1
NO          1  16   3
SL          0   2  23
```

4. Conclusão

Em geral, as superfícies de decisão do KNN parecem ter mais "resolução", mais detalhes que as superfícies de decisão do DMC, que tendem a generalizar uma "reta". Isso faz sentido pois o DMC baseia-se em um centróide, ou um ponto, para cada classe no momento da predição, enquando o KNN se baseia em cada ponto do conjunto de treinamentos. Pode-se concluir que o DMC se adequa mais a problemas onde as classes são claramente linearmente separáveis, enquanto o KNN (apesar de ser mais custoso computacionalmente) tanto se adequa a problemas linermanete separáveis quanto problemas não-lineares.