Aprendizado Supervisionado e Não-Supervisionado

Izabela Garcia Tavares Pinheiro Pérez

Novembro 2023

1 Introdução

O campo de análise de dados e aprendizado de máquina desempenha um papel crucial na extração de insights valiosos a partir de conjuntos de dados complexos. Neste trabalho, abordamos a tarefa de classificação de jogadores da NBA em duas categorias: aqueles que têm uma carreira profissional duradoura de pelo menos 5 anos e aqueles cuja carreira é mais curta.

A previsão da longevidade de uma carreira na NBA é um desafio complexo, pois envolve a análise de diversas características e desempenho de jogadores ao longo do tempo. Utilizamos métodos de aprendizado supervisionado e não supervisionado para tentar prever dados e classificar jogadores com base em características semelhantes.

Nosso objetivo é explorar como diferentes abordagens podem influenciar a precisão das previsões e como a escolha de parâmetros, o número de clusters ou vizinhos, impacta os resultados. O restante deste documento abordará detalhadamente a metodologia utilizada, a implementação dos algoritmos, os resultados obtidos e as conclusões derivadas desta análise.

2 Primeira Parte (K-Nearest Neighbors)

2.1 Pré-processamento de Dados

Antes de aplicar o algoritmo KNN, realizamos um conjunto de etapas de pré-processamento nos dados da NBA. Inicialmente, isolamos os dados de cada jogador do conjunto de teste, excluindo a característica `TARGET_5Yrs`, que será a variável a ser prevista na tarefa de classificação.

Posteriormente, normalizamos os dados para garantir escalas comparáveis entre as diferentes características. Por fim, extraímos o `vetor_target` do conjunto de teste, o qual contém as previsões ideais para a característica `TARGET_5Yrs`. Esse vetor é fundamental para a comparação com as previsões geradas pelos modelos de classificação.

2.2 KNN - Classificação de Carreiras

O algoritmo classifica um ponto de dados com base na classe predominante entre seus k vizinhos mais próximos no espaço de características. A proximidade é medida pela distância euclidiana, e a classe do ponto é determinada pela votação majoritária das classes de seus vizinhos mais próximos. A escolha adequada do parâmetro k é crucial, pois impacta diretamente na qualidade da classificação, influenciando como os dados serão agrupados em relação ao objetivo definido.

No contexto específico do nosso problema, buscamos identificar quais conjuntos de características indicam a permanência de jogadores na liga por mais de 5 anos e quais conjuntos estão associados à saída do jogador antes desse período.

Porém a dimensionalidade do vetor de características associado à cada jogador é bem alta, visto que existem 19 atributos nos dados (excluindo `TARGET_5Yrs`). Isso pode tornar a análise e o processamento dos dados mais desafiadores, especialmente ao lidar com conjuntos de dados grandes.

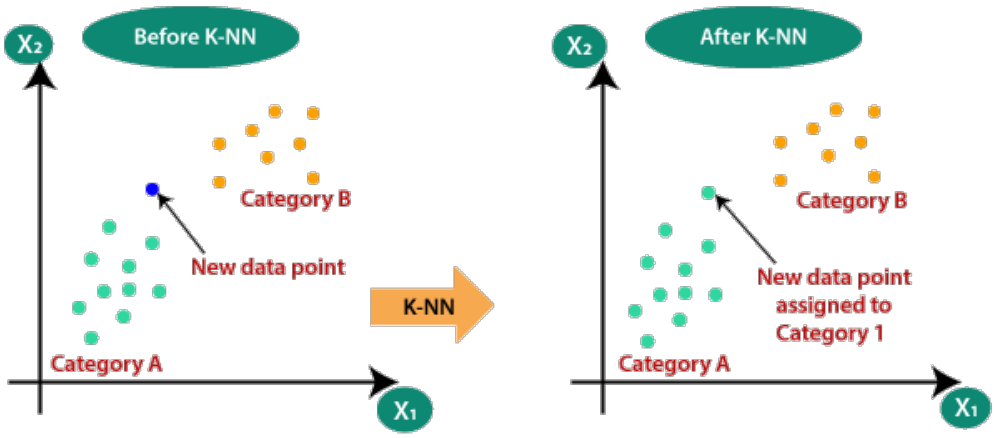

Figura 1: Funcionamento do KNN em dados de $\mathbb{R}^2$.

2.3 Escolhendo o valor de k

No processo de escolha dos valores de k para o algoritmo KNN, realizamos uma análise sistemática considerando os valores indicados (2, 10, 50) para avaliar como o desempenho variava.

Posteriormente, a fim de escolher um quarto valor proveitoso, testamos valores entre 1 e 200, para verificar qual parâmetro de k obtinha a maior acurácia. Veja a figura 2.

Figura 2: Valores de k em relação à acurácia.

Após a análise dos resultados, identificamos que o valor de k que proporcionava o melhor desempenho foi 101, onde observamos uma acurácia global máxima dentro desse intervalo. Note que os valores acima de 150 tendem a manter uma taxa mais constante, o que nos levou a considerar apenas esse intervalo de valores para k. A escolha de k = 101 proporcionou uma acurácia de 67,91%.

2.4 Avaliação de Resultados

Para avaliar o desempenho dos algoritmos, calculamos métricas como acurácia, precisão, recall e F1-score.

K	Acurácia (%)	Precisão	Recall	F1-Score
2	58.96	0.754	0.512	0.610
10	65.30	0.745	0.679	0.710
50	66.04	0.723	0.744	0.733
101	67.91	0.730	0.774	0.751

Tabela 1: Resultados do KNN para diferentes valores de k.

Adicionalmente, construímos matrizes de confusão para visualizar o desempenho em detalhes.

K	Matriz de Confusão
2	$\begin{bmatrix} 72 & 28 \\ 82 & 86 \end{bmatrix}$
10	$\begin{bmatrix} 61 & 39 \\ 54 & 114 \end{bmatrix}$
50	$\begin{bmatrix} 52 & 48 \\ 43 & 125 \end{bmatrix}$
101	$\begin{bmatrix} 52 & 48 \\ 38 & 130 \end{bmatrix}$

Tabela 2: Matrizes de Confusão para diferentes valores de k.

2.5 Análise Crítica dos Resultados

No experimento com diferentes valores de k para o algoritmo KNN, observamos o seguinte desempenho:

Para k=10, a acurácia foi de 65.30%, representando uma melhoria em comparação com k=2, embora ainda não tenha atingido um patamar considerado ótimo. Houve uma melhoria nas métricas de precisão e recall, indicando que a escolha de k foi mais robusta, resultando em um equilíbrio razoável entre precisão e recall, conforme evidenciado pelo F1-Score para ambas as classes.

Aumentando k para 50, observamos um aumento pequeno na acurácia para 66.04% em relação a k=10. As métricas de precisão e recall mantiveram níveis similares, e o F1-Score continuou indicando um equilíbrio moderado entre precisão e recall.

Ao considerar k=101, alcançamos uma acurácia de 67.91%, representando uma melhoria adicional e sugerindo benefícios ao considerar mais vizinhos na decisão de classificação. As métricas de precisão e recall foram boas para ambas as classes, e o F1-Score indicou um desempenho equilibrado entre precisão e recall.

2.6 Pontos extras - Comparação com Scikit-learn

Para comparar os resultados obtidos por meio da implementação manual do algoritmo k-NN com os resultados da biblioteca `scikit-learn`, utilizamos as funções `KNeighborsClassifier`, `accuracy_score`, `precision_score`, `recall_score`, `f1_score` e `confusion_matrix`. Ambas as abordagens apresentaram resultados idênticos no mesmo conjunto de dados. Essa concordância pode ser atribuída ao fato de que, embora a biblioteca utilize técnicas mais eficientes para calcular distâncias e validar vizinhos, a lógica subjacente à predição é a mesma. Portanto, as discrepâncias nos tempos de execução são a principal distinção, enquanto os resultados de avaliação permanecem consistentes, evidenciando a robustez e validade da implementação manual.

3 KMeans - Agrupamento de Jogadores

O algoritmo K-Means é uma técnica de agrupamento que visa particionar um conjunto de dados em subgrupos (clusters) com base em características similares. Cada ponto de dado é atribuído ao cluster cujo centroide (ponto central do cluster) é mais próximo, e os centroides são atualizados iterativamente para otimizar a posição dos clusters em relação a disposição dos dados.

No contexto do nosso trabalho, o algoritmo K-Means pode ser aplicado para identificar padrões de comportamento nos dados, agrupando jogadores com características semelhantes. Isso pode ser valioso para analisar se existem perfis distintos de jogadores (combinações de características) que permanecem na liga por mais de 5 anos ou que saem antes desse período.

No entanto, é importante destacar que o K-Means assume que os clusters são "esféricos"e de tamanhos semelhantes, o que pode não refletir a complexidade real dos dados. Além disso, a escolha do número de clusters, representado pelo parâmetro k, é um desafio, pois influencia diretamente nos resultados obtidos, visto que se os dados não forem k-partidos, a classificação é mal-executada.

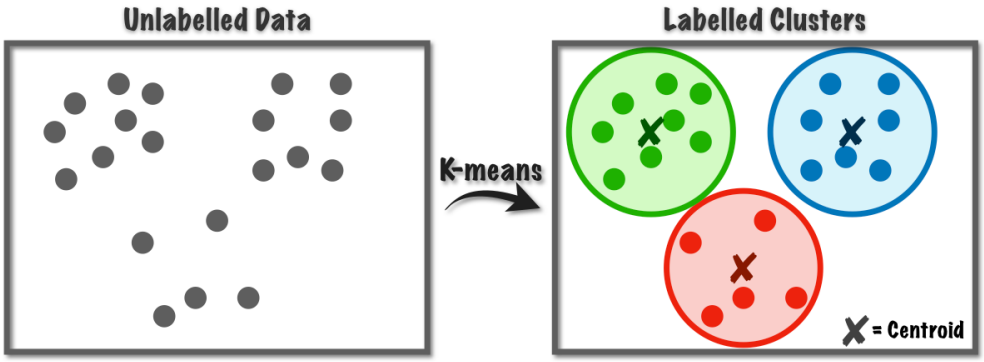

Figura 3: Funcionamento do K-means com k=3 em dados de $\mathbb{R}^2$.

3.1 Atribuindo k = 2

A atribuição de $k = 2$ no algoritmo K-Means implica na criação de dois clusters distintos, cada um representado por um centroide. Esses centroides são pontos centrais nos clusters e são ajustados iterativamente durante o processo de otimização do algoritmo.

Na nossa implementação, a não consideração da relação dos agrupamentos com os vetores `TARGET_5Ys` para classificar os clusters em 1 ou 0 torna o algoritmo sensível a essa escolha "arbitrária"da inicialização dos centroides. Isso é evidenciado ao comparar os resultados de duas execuções distintas:

Na primeira execução, os centroides foram inicializados com os seguintes resultados:

Execução 1:

- *Cluster 1:* Caracterizado por valores relativamente altos em todas as dimensões, sugerindo uma tendência de agrupar pontos com características mais elevadas.
- *Cluster 2:* Apresenta valores mais baixos, indicando uma inclinação para agrupar pontos com características mais modestas.

A acurácia para esta execução foi de 41.12%, indicando que a separação em dois clusters não reflete efetivamente a distribuição das características `TARGET_5Ys`.

Na segunda execução, os centroides foram reinicializados em posições diferentes:

Execução 2:

- *Cluster 1:* Caracterizado por valores mais baixos em todas as dimensões, indicando uma mudança significativa em relação à execução anterior.
- *Cluster 2:* Apresenta valores mais altos, sugerindo uma propensão para agrupar pontos com características mais elevadas.

A acurácia para esta execução melhorou significativamente, atingindo 58.58%. Este resultado indica uma melhor adaptação dos clusters aos dados, representando uma distribuição mais eficaz dos pontos no espaço de características.

A variação nos centroides entre as duas execuções destaca a sensibilidade do K-Means à escolha inicial. A seleção arbitrária do ponto inicial pode impactar os resultados finais, levando a soluções diferentes. Portanto, é comum realizar múltiplas execuções do algoritmo com diferentes inicializações para mitigar esse efeito e buscar uma solução mais estável.

Veja os gráficos abaixo para perceber melhor o comportamento de alternância entre as labels dos clusters e a acurácia:

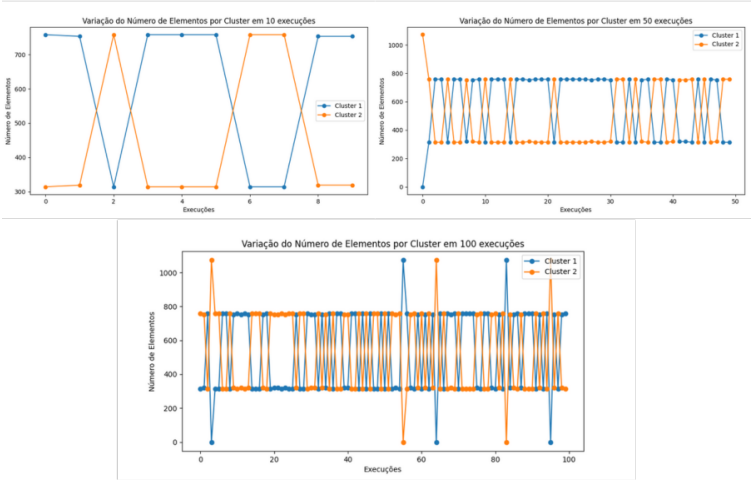

Figura 4: Distribuição dos elementos nos clusters em 10, 50 e 100 execuções.

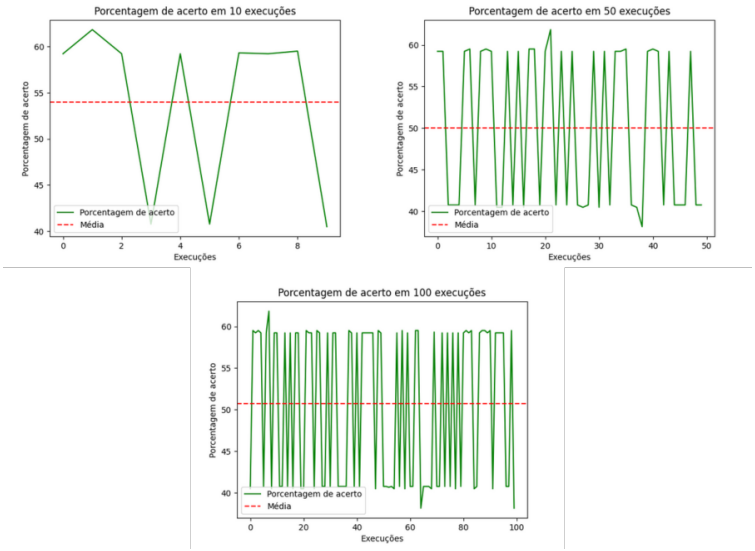

Figura 5: Acurácia em 10, 50 e 100 execuções.

3.1.1 Análise Crítica dos Resultados

Em resumo, ao aplicar o K-Means com $k = 2$, observa-se uma classificação abrangente dos dados em relação ao atributo `TARGET_5Ys`, alcançando acurácias próximas a 70%. No entanto, a instabilidade mencionada anteriormente introduz variações nas acurácias, resultando em uma média em torno de 53%, o que não é satisfatório. É importante ressaltar que a influência significativa da escolha arbitrária dos pontos iniciais dos centroides destaca a necessidade de uma abordagem consciente na classificação dos clusters para aprimorar essas métricas. Quando realizada de maneira cuidadosa, a classificação dos clusters pode significativamente melhorar o desempenho do algoritmo, transformando-o em um classificador eficaz.

3.1.2 Pontos extras - Comparação com Scikit-learn

A comparação entre os resultados da implementação manual do K-Means e a versão da biblioteca Scikit-learn revela uma notável consistência nos resultados finais, embora algumas variações sejam observadas. Ambas as implementações convergem para resultados semelhantes quando os clusters são classificados de maneira consistente, indicando a preservação da lógica subjacente do algoritmo. No entanto, algumas diferenças são evidentes, como a mudança nas posições dos centroides e a ligeira melhoria na acurácia na implementação da Scikit-learn.

Estas variações podem ser atribuídas a discrepâncias nas estratégias de inicialização dos centroides e nos métodos de otimização empregados. A Scikit-learn incorpora otimizações eficientes, incluindo uma inicialização inteligente dos centroides, que pode contribuir para uma convergência mais rápida e resultados mais estáveis. Essa estratégia de inicialização, aliada a outras técnicas de otimização internas, pode influenciar a formação dos clusters de maneira mais eficaz.

Apesar das divergências, as métricas de acurácia permanecem próximas entre as implementações, evidenciando que ambas são capazes de realizar uma classificação eficaz dos dados em dois clusters, mesmo que as estratégias internas do Scikit-learn proporcionem vantagens em termos de eficiência e estabilidade durante o processo de otimização.

Além disso, é crucial destacar que, devido à instabilidade inerente a ambos os algoritmos na tarefa de classificação dos clusters, a comparação direta entre eles pode ser limitada. Se houver discrepância na classificação, em que o algoritmo manual atribui corretamente, enquanto o da Scikit-learn não, a acurácia da implementação manual pode parecer mais elevada, mesmo que seja menos otimizada. Isso ressalta a complexidade da análise comparativa, pois a eficácia de cada abordagem pode ser influenciada por fatores como a inicialização dos centroides e a sensibilidade do algoritmo a variações nos dados de entrada. Veja as imagens abaixo para compreender essa diferença.

Figura 6: Atribuição correta em ambos os algoritmos

Figura 7: Atribuição correta somente na implementação manual

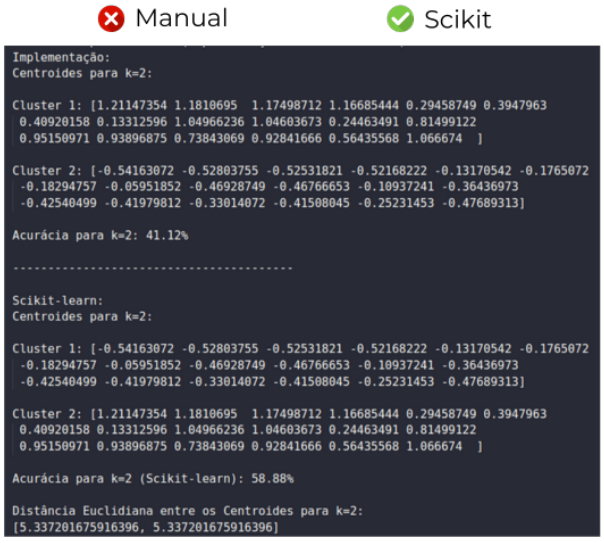

Figura 8: Atribuição correta somente no algoritmo Scikit-learn

3.2 Atribuindo k = 3

A atribuição de $k = 3$ no algoritmo K-Means implica na criação de três clusters distintos, cada um representado por um centroide. Esses centroides são pontos centrais nos clusters e também são ajustados iterativamente durante o processo de otimização do algoritmo assim como em k=2.

Nessa implementação, também não é levada em consideração a relação dos agrupamentos com os vetores `TARGET_5Yrs` para classificar os clusters em 1 ou 0 tornando o algoritmo sensível a essa escolha "arbitrária"da inicialização dos centroides.

Veja os gráficos abaixo para perceber melhor o comportamento de alternância entre as labels dos clusters e a acurácia com k = 3:

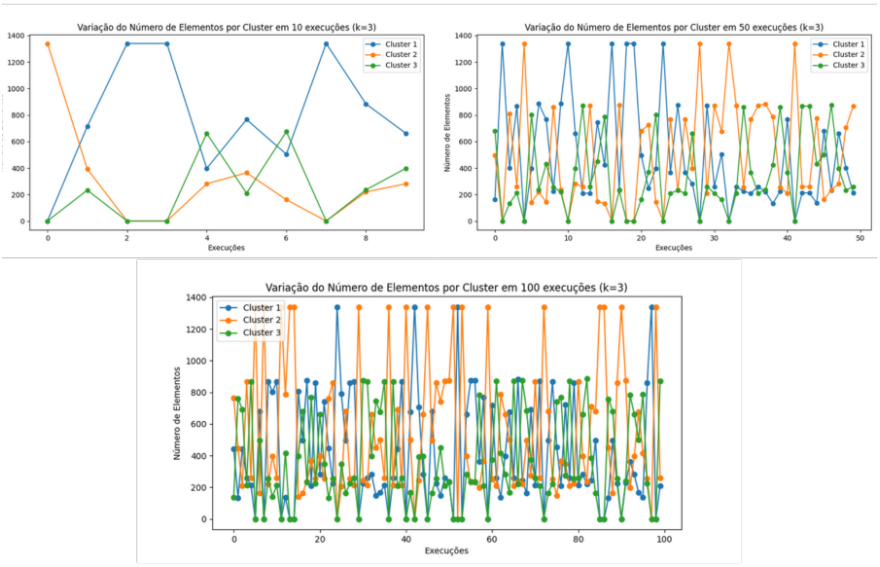

Figura 9: Distribuição dos elementos nos clusters em 10, 50 e 100 execuções.

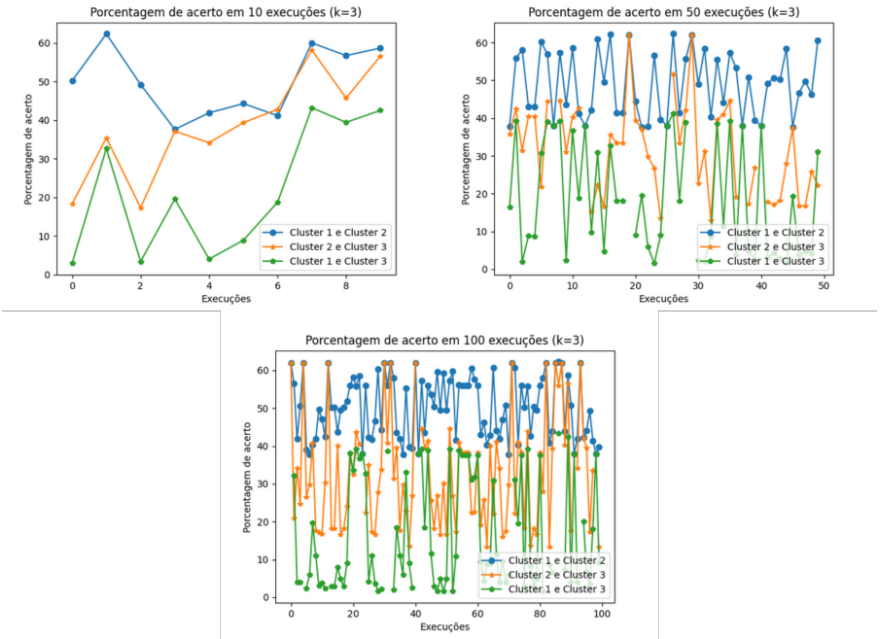

Figura 10: Acurácia em 10, 50 e 100 execuções.

3.2.1 Análise Crítica dos Resultados

Em síntese, ao empregar o algoritmo K-Means com $k = 3$, percebe-se uma classificação abrangente dos dados em relação ao atributo `TARGET_5Yrs`, englobando a maior parte dos elementos em apenas alguns pares de clusters. Isso resulta em acurácias que se aproximam de 70%, semelhante ao que foi observado com $k = 2$. Contudo, mesmo enfrentando o desafio da instabilidade mencionado anteriormente, persiste a presença de

um cluster com desempenho inferior em todas as execuções. Em outras palavras, há uma configuração ótima em que dois dos três clusters apresentam melhor desempenho, evidenciando a limitação na capacidade de partição tripla dos dados.

3.2.2 Pontos extras - Comparação com Scikit-learn

Na análise comparativa entre a implementação manual e a versão da biblioteca Scikit-learn para k=3, os resultados demonstram uma notável similaridade com a análise em k=2. Ambos os algoritmos convergem quando os clusters de maior acurácia são identificados corretamente, apresentando pequenas discrepâncias atribuíveis às otimizações internas incorporadas na implementação da Scikit-learn, como já mencionado.

Assim como na abordagem manual, a implementação da Scikit-learn evidencia a persistência de um par de clusters que apresenta melhor desempenho. Este padrão sugere que a tripartição dos dados não oferece benefícios significativos na classificação da característica `TARGET_5Yrs`. Os resultados consistentes entre as implementações reforçam a robustez do algoritmo em identificar padrões subjacentes, mesmo quando há variações nas estratégias de inicialização e otimização.

4 Conclusão

O estudo abordou a aplicação de técnicas de aprendizado supervisionado e não supervisionado na análise de dados relacionados aos jogadores da NBA, com foco na previsão da longevidade de suas carreiras. A implementação e análise crítica dos algoritmos K-Nearest Neighbors (KNN) e K-Means foram conduzidas, destacando nuances, desafios e aspectos relevantes para cada abordagem.

No contexto do KNN, a escolha adequada do parâmetro k foi crucial, e uma análise sistemática revelou que $k = 101$ proporcionou o melhor desempenho, alcançando uma acurácia de 67.91%. A avaliação detalhada por meio de métricas como precisão, recall e F1-Score permitiu uma compreensão mais profunda da capacidade preditiva do modelo.

Quanto ao K-Means, a sensibilidade à escolha arbitrária dos pontos iniciais dos centroides foi evidenciada, impactando significativamente os resultados. O experimento com $k = 2$ e $k = 3$ revelou instabilidade nos agrupamentos, destacando a importância da inicialização cuidadosa para mitigar esse efeito. Além disso, foi possível perceber que a tripartição dos dados não beneficia a classificação, indicando que as características são, de fato, bipartidas.

A comparação com a implementação da biblioteca Scikit-learn ressaltou a consistência dos resultados, obtendo valores idênticos no KNN, embora algumas variações tenham sido observadas nos resultados do K-means devido às estratégias internas de otimização da biblioteca.

Em última análise, ambos os algoritmos apresentaram desafios e pontos fortes. O KNN demonstrou capacidade de classificação robusta com uma escolha cuidadosa de k, enquanto o K-Means revelou sensibilidade à inicialização dos centroides e classificação dos clusters.

Este estudo destaca a complexidade e a importância da escolha de parâmetros e estratégias na aplicação de algoritmos de aprendizado de máquina. Além disso, ressalta a necessidade de uma compreensão profunda dos dados e da natureza do problema para tomar decisões informadas durante o processo de modelagem. Esses insights são essenciais para extrair conclusões confiáveis e relevantes a partir de conjuntos de dados complexos e variados como o dataset utilizado.

Referências

[1] NBA Rookie Longevity Project, `https://www.kaggle.com/code/mamadoudiallo/nba-rookie-longevity-project/`

[2] K-Nearest Neighbors Implementation (GitHub), `https://github.com/div3125/k-nearest-neighbors/blob/master/knn/main.py`

[3] K-Means Clustering (Scikit-learn Documentation), `https://scikit-learn.org/stable/modules/generated/sklearn.cluster.KMeans.html`

[4] K-Means: O que é, como funciona, aplicações e exemplo em Python, https://medium.com/programadores-ajudando-programadores/k-means-o-que-%C3%A9-como-funciona-aplica%C3%A7%C3%B5es-e-exemplo-em-python-6021df6e2572

[5] K-Nearest Neighbors Classifier (Scikit-learn Documentation), https://scikit-learn.org/stable/modules/generated/sklearn.neighbors.KNeighborsClassifier.html

[6] KNN (K-Nearest Neighbors): O que é, como funciona e exemplo em Python, https://medium.com/brasil-ai/knn-k-nearest-neighbors-1-e140c82e9c4e