Trabalho Prático 2

Aprendizado de Máquina

Brisa do Nascimento Barbosa
Universidade Federal de Minas Gerais (UFMG)

1. Introdução

No presente trabalho, focamos na classificação de jogadores da NBA com base em 19 atributos distintos, com o objetivo de determinar se suas carreiras se estenderam por um período mínimo de 5 anos na liga. Para isso, foram implementados dois algoritmos de aprendizado de máquina: o KNN, que pertence à categoria de aprendizado supervisionado, e o KMeans, que se enquadra no âmbito do aprendizado não supervisionado. Os resultados obtidos por meio dessas abordagens foram analisados e comparados, estendendo-se a uma comparação também com os resultados fornecidos pelos algoritmos da biblioteca scikit-learn.

Parte 1: Aprendizado Supervisionado

Inicialmente, implementamos um algoritmo de aprendizado supervisionado, o KNN. O aprendizado supervisionado é uma abordagem no qual o algoritmo é treinado em um conjunto de dados rotulados, ou seja, um conjunto de dados que contém as entradas e as saídas desejadas correspondentes. O objetivo é aprender uma função que mapeia as entradas para as saídas de maneira a fazer previsões em novos dados não rotulados.

1. K-Nearest Neighbors

O KNN, ou k-vizinhos mais próximos, é um algoritmo usado para classificação e regressão, fundamentado na abordagem de aprendizado preguiçoso (lazy). Este método pertence à categoria de algoritmos baseados em instâncias, caracterizando-se pelo fato de não buscar construir um modelo interno. Em vez disso, o KNN adota uma estratégia de memorização das instâncias presentes no conjunto de treinamento, ou seja, quando é necessário realizar uma classificação para uma nova instância, o algoritmo identifica os k-vizinhos mais próximos dessa instância.

A ideia principal é atribuir uma classe a uma amostra de teste com base nas classes das amostras vizinhas mais próximas no espaço de características. O parâmetro 'k' representa o número de vizinhos mais próximos a serem considerados. O algoritmo calcula as distâncias entre a amostra de teste e todas as amostras no conjunto de treinamento. Em seguida, identifica os 'k' vizinhos mais próximos com base nessas distâncias. Por fim, determina a classe mais comum entre esses vizinhos para atribuir como a classe prevista para a amostra de teste.

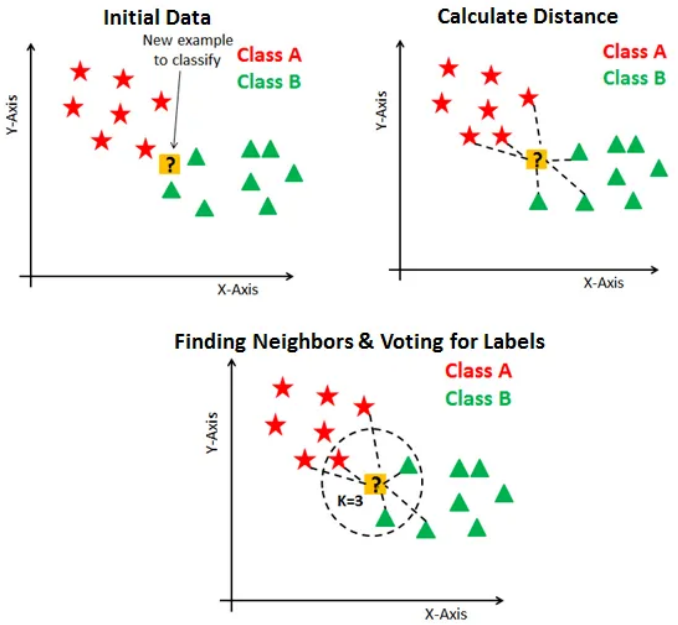


Fig. 1. Diagrama do funcionamento do KNN

No código desenvolvido, foi incorporada uma função scatter com o propósito de visualizar a classificação de forma tridimensional, utilizando cores emblemáticas da NBA, o vermelho e o azul. Vale ressaltar que a dimensão original do conjunto de dados consiste em 19 atributos, no entanto, optou-se por reduzir essas dimensões para apenas três, a fim de viabilizar a interpretação visual. Esta redução implica, inevitavelmente, na perda de algumas informações, mas ainda assim proporciona a capacidade de observar o padrão.

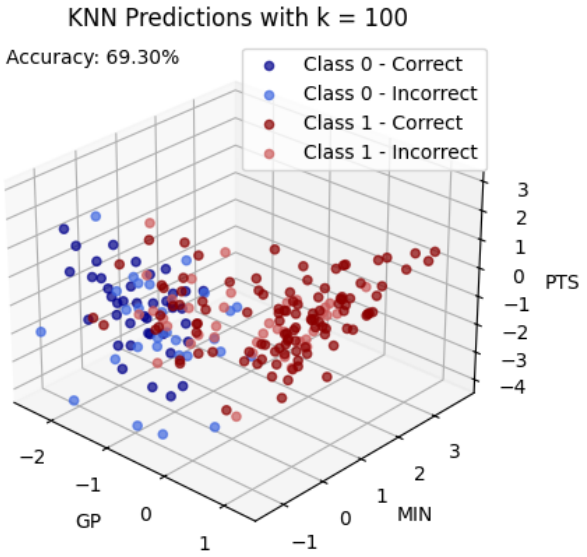


Fig. 2. Gráfico de dispersão do KNN com k=100 com a distribuição das predições corretas e incorretas

1.1. Parâmetro k

A escolha apropriada do parâmetro k no algoritmo KNN é essencial para garantir um desempenho otimizado do modelo. Em geral, o tamanho do conjunto de dados pode influenciar a escolha de k. Em conjuntos de dados menores, um valor de k muito pequeno, como o caso de k=2, pode resultar em um modelo sensível a outliers ou variações aleatórias nos dados. Por outro lado, em conjuntos de dados muito grandes, um valor de k também muito grande pode levar a uma perda de detalhes e sensibilidade nos padrões locais. No caso, como estamos trabalhando com um conjunto pequeno de dados, é necessário considerar o equilíbrio entre generalização e sensibilidade local para garantir uma boa identificação do padrão para a classificação

Ainda, como estamos lidando com um conjunto de classes binárias (jogar ou não na liga por pelo menos 5 anos), é relevante considerar a distribuição das classes, já que se as classes estão desproporcionalmente representadas no conjunto de dados, isso pode afetar a capacidade do modelo de generalizar efetivamente.

Tendo isso em vista, criamos o seguinte gráfico avaliando 4 valores de k diferentes ao longo de 30 iterações do algoritmo. Claramente, o pior desempenho é com k=2, enquanto k=100 gera as melhores acurácias (chegando próximo a 70% de acurácia e com mínimo em 62% — que é melhor que k=2 alcançou). É perceptível que os dados utilizados influenciam diretamente no resultado, independente de k. Por exemplo, notou-se que em torno da iteração 20, todos os valores de k apresentaram as melhores acurácias globais. Isso pode ser atribuído ao fato de que, a cada iteração, uma fração randomizada do conjunto de dados é selecionada para treinamento, o que pode implicar na presença de amostras mais representativos do conjunto total em determinados casos.

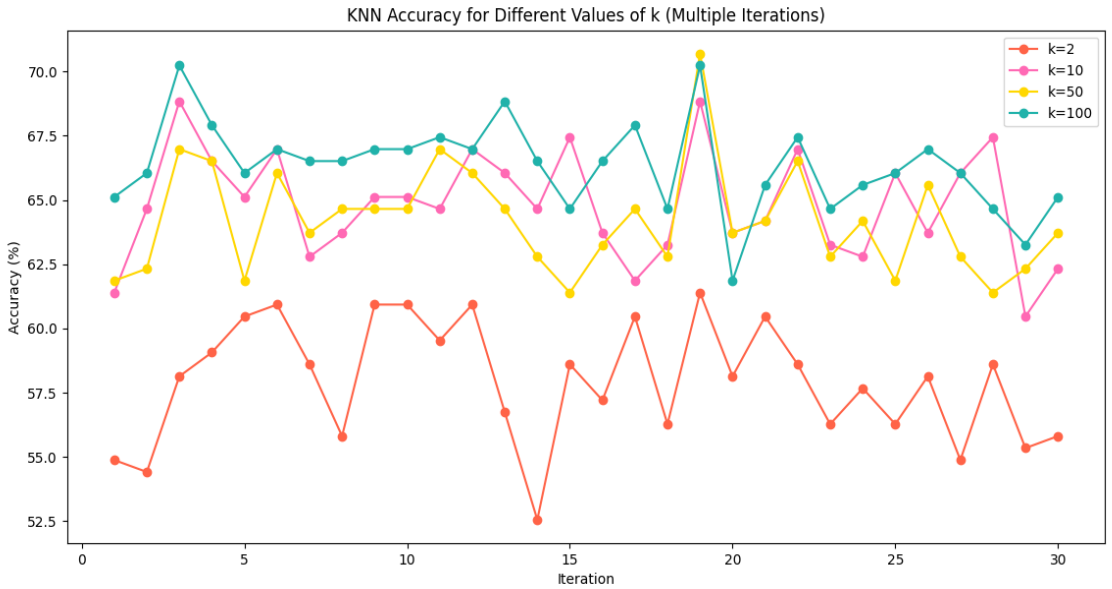


Fig. 3. Gráfico da acurácia do KNN com os k selecionados ao longo de 30 iterações

Uma possível otimização consistiria em selecionar os valores de k com base em uma abordagem mais evolutiva, assemelhando-se a algoritmos genéticos. Nesse sentido, a escolha dos novos valores de k poderia ser guiada por uma memória do desempenho anterior, permitindo a adaptação contínua dos parâmetros com o objetivo de aprimorar a classificação.

1.2. Análise das métricas

A matriz de confusão é uma tabela que mostra o desempenho de um modelo de classificação, comparando suas previsões com os valores reais do conjunto de dados. Ela inclui quatro elementos principais: Verdadeiros Positivos (TP), Falsos Positivos (FP), Verdadeiros Negativos (TN) e Falsos Negativos (FN).

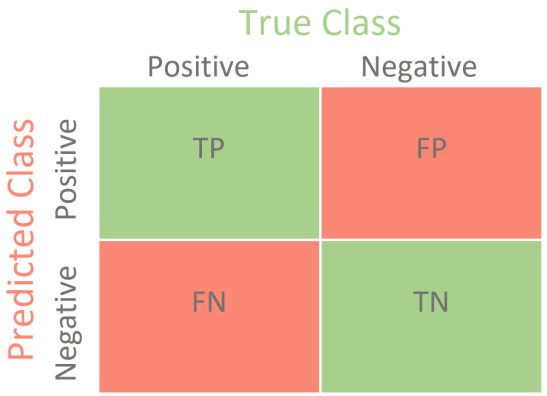


Fig. 4. Esquematização da Matriz de Confusão

Para cara k, geramos uma matriz de confusão associada.

k	2	10	50	100
Confusion Matrix	$\begin{bmatrix} 57.3 & 23.4 \\ 68 & 66.3 \end{bmatrix}$	$\begin{bmatrix} 46.6 & 34.4 \\ 41.2 & 92.8 \end{bmatrix}$	$\begin{bmatrix} 42.1 & 38.5 \\ 34 & 97.7 \end{bmatrix}$	$\begin{bmatrix} 40.2 & 36.8 \\ 34 & 104 \end{bmatrix}$

Agora, observe a seguinte tabela, que relaciona k às métricas de acurácia, precisão, recall e F1. Para chegar nesses resultados, calculamos uma média entre 20 execuções para cada k, usando as mesmas sementes de aleatoriedade.

K	Accuracy	Precision	Recall	F1
2	57.49%	57.49%	59.88%	60.04%
10	64.84%	63.41%	63.03%	63.22%
50	64.98%	62.41%	62.52%	62.47%
100	67.07%	63.83%	64.18%	64.00%

1.2.1. Acurácia

$$Acurácia = \frac{Verdadeiros\ Positivo + Verdadeiros\ Negativos}{Total\ de\ Predições}$$

A acurácia mede a proporção de predições corretas em relação ao total de predições. Observando os resultados, percebemos um aumento gradual na acurácia à medida que k cresce. Isso sugere que, em geral, o modelo está fazendo previsões mais precisas à medida que considera mais vizinhos próximos. No entanto, é importante notar que, após um certo ponto (especificamente, entre k = 50 e k = 100), o ganho na acurácia parece estagnar ou aumentar apenas bem marginalmente.

De fato, observamos essa estagnação ao realizar 20 execuções para k=150 e k=250,

K	Accuracy	Precision	Recall	F1
150	66.51%	64.00%	64.18%	64.09%
250	64.42%	60.85%	61.34%	61.09%

No entanto, vale ressaltar que essa relação não é linear, uma vez que valores excessivamente altos de k podem levar ao overfitting e resultar na perda de sensibilidade aos detalhes locais. Além de que pode acentuar a complexidade computacional.

1.2.2. Precisão

$$Precisão = \frac{Verdadeiros\ Positivos}{Verdadeiros\ Positivos + Falsos\ Positivos}$$

A precisão é a proporção de verdadeiros positivos em relação ao total de predições positivas. Notamos que, com k = 2, a precisão é a mais baixa, indicando uma tendência a classificar erroneamente algumas instâncias como positivas. Com k = 10, observamos uma melhoria na precisão, sugerindo uma redução nos falsos positivos. No entanto, entre k = 50 e k=100, a adição de mais vizinhos pode não ter um impacto significativo.

1.2.3. Revocação

$$Revocação = \frac{Verdadeiros\ Positivos}{Verdadeiros\ Positivos + Falsos\ Negativos}$$

A precisão é a proporção de verdadeiros positivos em relação ao total de predições positivas. Notamos uma tendência semelhante à precisão, onde o aumento de k leva a uma melhoria na revocação até um ponto, mas depois estagnou ou aumenta apenas marginalmente. Isso sugere que, para alguns valores específicos de k, o modelo consegue capturar mais instâncias positivas, mas além de certo limite, não há ganhos significativos.

1.2.4. F1-Score

$$F1 = 2 * \frac{Precisão * Revocação}{Precisão + Revocação}$$

O F1-Score é uma métrica que harmoniza a precisão e a revocação, fornecendo uma visão equilibrada do desempenho do modelo. F1 está acima de 60% para cada k indica um desempenho moderado do modelo, ou seja, equilíbrio entre a capacidade do modelo de fazer previsões corretas (precisão) e capturar corretamente todas as instâncias positivas (revocação).

1.3. Comparação com Scikit-learn

A execução foi notavelmente mais rápida ao utilizar a implementação do scikit-learn. Isso faz sentido porque essa biblioteca é otimizada e escrita em linguagem C, o que proporciona um desempenho superior quando comparada a uma implementação manual em Python. Além da eficiência, também foi percebida a facilidade de uso do scikit-learn, que tem funções simples de usar. Assim, embora implementar manualmente fornece maior flexibilidade e compreensão aprofundada do processo, a abordagem da biblioteca oferece uma solução mais rápida e intuitiva.

Por fim, apesar da diferença de desempenho, os resultados obtidos foram semelhantes. A análise demonstrou que para k=2, a performance continua sendo a pior, enquanto k=100 permanece como a opção mais eficaz, com não muita diferença em relação a k=50. Ainda, as acurácias máximas e mínimas foram próximas em ambas as implementações, apesar do scikit ter uma média melhor de resultados.

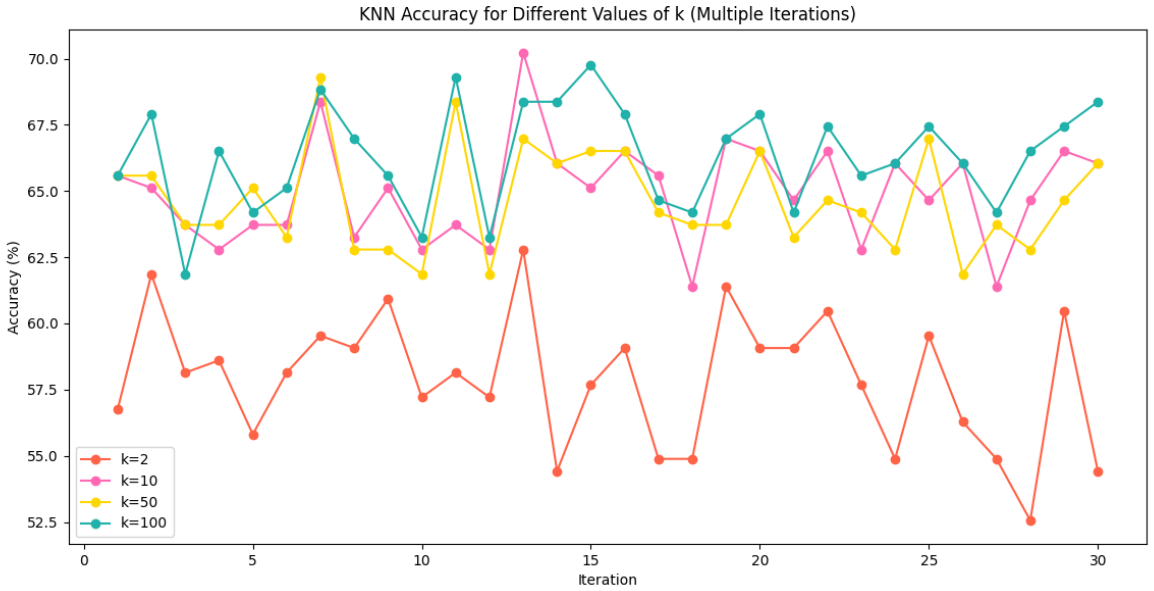


Fig. 5. Gráfico da acurácia do KNN com scikit-learn com os k selecionados ao longo de 30 iterações

Parte 2: Aprendizado Não-Supervisionado

O aprendizado não supervisionado é uma abordagem de aprendizado de máquina em que os algoritmos são treinados em conjuntos de dados não rotulados, ou seja, conjuntos de dados em que as saídas desejadas não são fornecidas. Ao contrário do aprendizado supervisionado, o algoritmo deve descobrir padrões, estruturas ou relações nos dados por conta própria. Neste trabalho, aplicamos K-Means para realizar a tarefa de clusterização dos jogadores da NBA, usando apenas os atributos das estatísticas de jogo, não incluindo a coluna target.

2. K-Means

O algoritmo KMeans é uma técnica de aprendizado não supervisionado utilizada para realizar o agrupamento de dados, também conhecido como clustering. Assim, o KMeans não requer rótulos para os dados de treinamento, pois consiste em agrupar dados de maneira natural, com base em suas características intrínsecas.

O funcionamento do KMeans é conduzido por um processo iterativo que busca dividir os dados em 'k' clusters, neste caso, k será 2 e 3. Inicialmente, o algoritmo inicia atribuindo aleatoriamente 'k' centróides, que são pontos no espaço de características, o algoritmo passa por uma fase de atualização dos centróides. Os novos centróides são calculados como a média dos pontos atribuídos a cada cluster. Esse processo é iterativamente repetido, com os pontos sendo re-atribuídos aos clusters e os centróides sendo recalculados. O objetivo é minimizar a variância intra-cluster, ou seja, reduzir a soma das distâncias quadráticas dos pontos para o centróide de seu cluster. A iteração continua até que os centróides não se movam significativamente entre as iterações, no caso, usamos 1500 iterações, indicando a convergência do algoritmo.

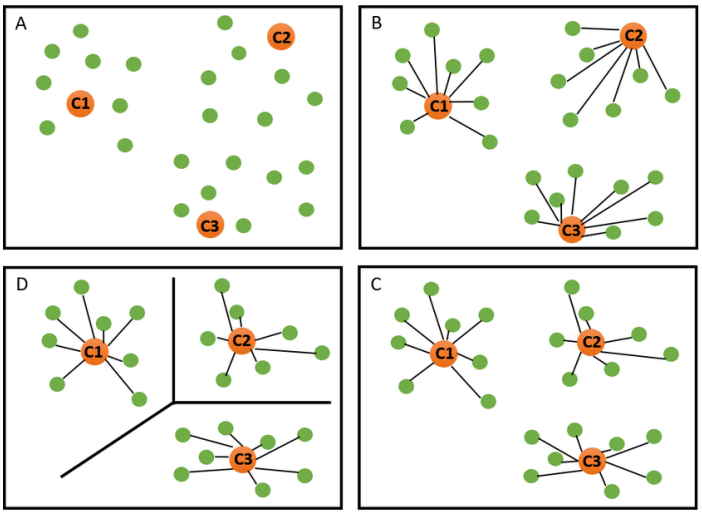


Fig. 6. Diagrama do funcionamento do KMeans

2.1. Parâmetro k

A representação tridimensional da classificação é apresentada a seguir por meio de um gráfico de dispersão (scatter plot) do KMeans para k=2 e k=3, conforme implementado no código. Apesar da inevitável perda de informação ao reduzir a dimensionalidade de 19 para 3, como observado anteriormente no contexto do KNN, ainda é possível discernir claramente os padrões de agrupamentos.

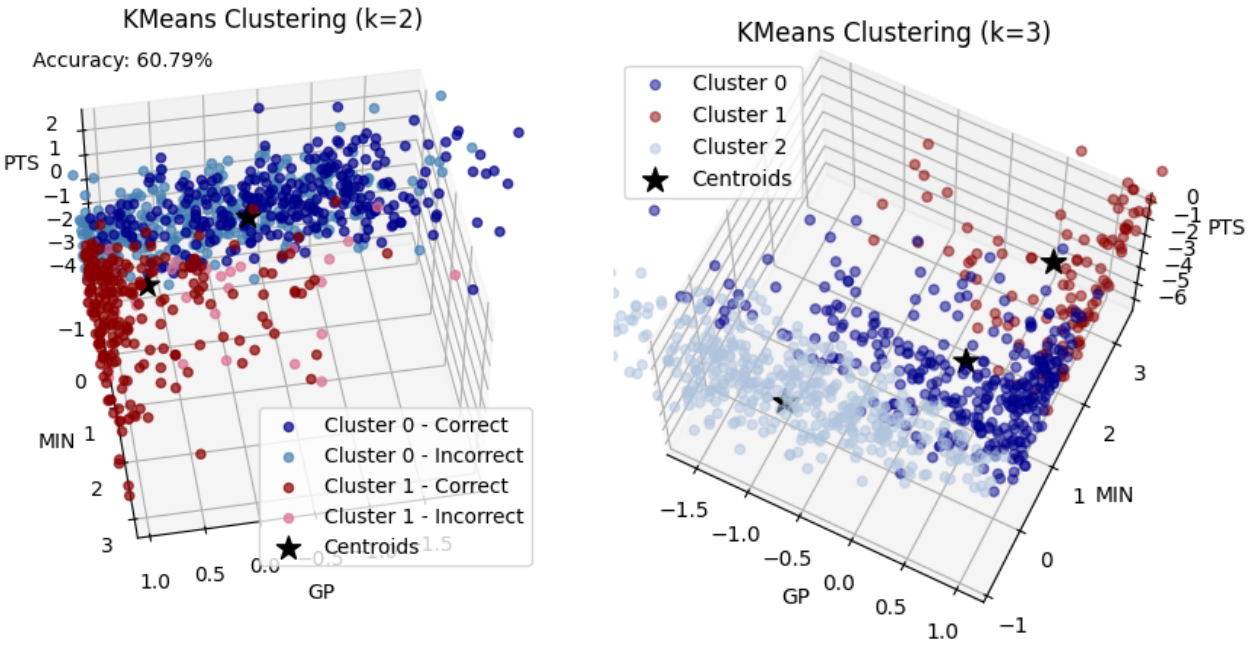
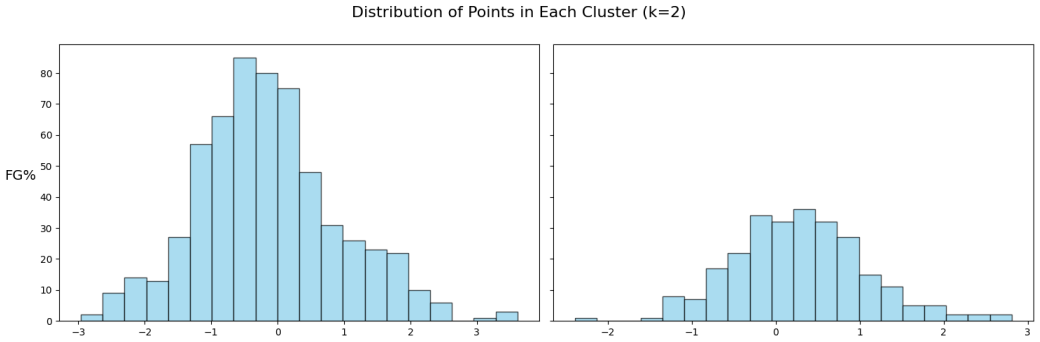


Fig. 7 e 8. Gráfico de dispersão do KMeans com k=2 e k=3, no primeiro com distribuição das predições corretas e incorretas

2.1.1 k=2

Ao executar o KMeans para dois centróides, esperamos um resultado mais consistente do que para três, tendo em vista nossa hipótese de caracterização ser baseada em se um jogador permaneceu na liga por pelo menos 5 anos ou não.

Considere a representação da distribuição dos clusters. Nos seguintes histogramas, a escala corresponde à contagem de observações em cada intervalo (bin), enquanto os atributos normalizados estão plotados no eixo vertical. Estes histogramas foram obtidos utilizando um experimento que teve 60,79% de acurácia. Cada barra nos histogramas reflete, portanto, a densidade de observações em diferentes faixas de valores normalizados, proporcionando uma visão da distribuição dos dados dentro dos dois clusters identificados. Destacamos aqui os atributos com gráficos mais relevantes.



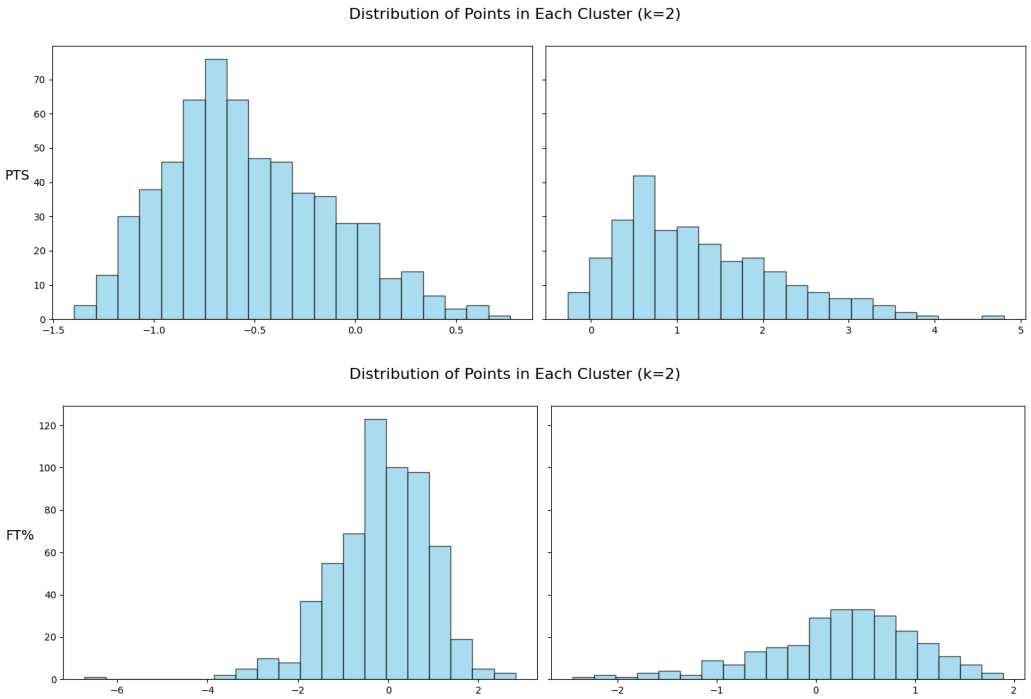


Fig. 9. Histogramas de distribuição nos clusters com k = 2 para os parâmetros 'FG%', 'PTS', 'FT%'

Assim, é perceptível a influência de certos fatores, como pontos marcados por jogo, percentual de conversão de lances livres e percentual de conversão de arremessos, na classificação. De fato, esses atributos, dentre os outros não apresentados aqui, caracterizam um jogador de alto desempenho que permanece na NBA por pelo menos 5 anos.

Também foram calculadas as acurácias ao longo de 200 iterações. Os resultados variaram entre um mínimo de 36.87% e um máximo de 63.13%, com uma média global de acurácias atingindo 50.14%. Assim, constata-se que o KMeans apresentou um desempenho inferior em relação ao observado no algoritmo KNN, que apresentou uma média global acima de 60% para k=100.

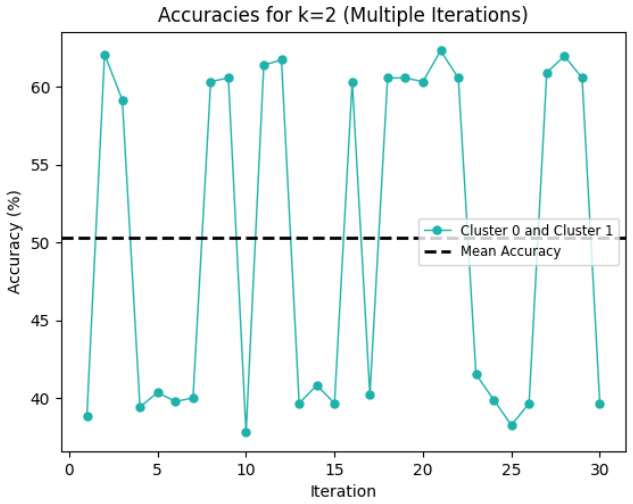


Fig. 8. Gráfico comparativo entre acurácias para clusters com k=2

2.1.1 k=3

Desenvolvemos histogramas de cada parâmetro para analisar a distribuição dos jogadores com base em seus atributos dentro de cada cluster. Destacamos a seguir as distribuições mais notáveis.

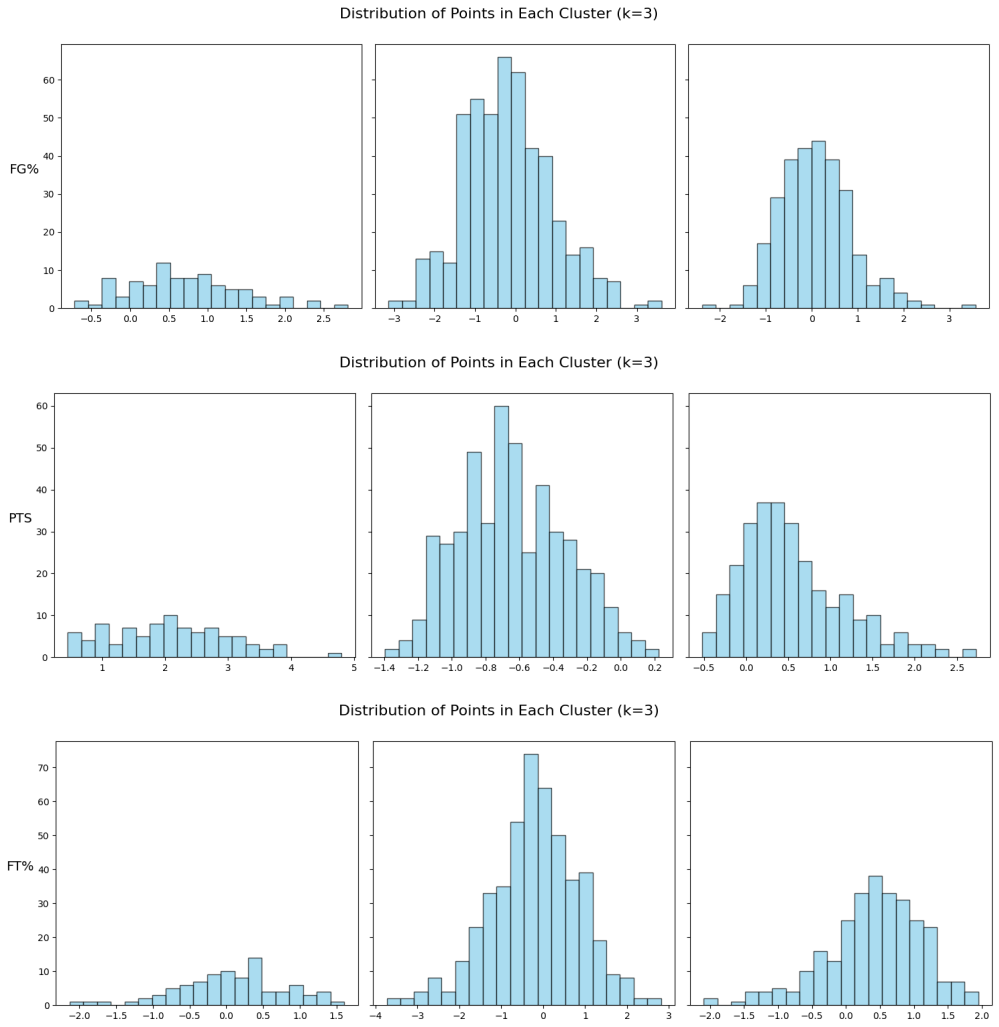


Fig. 9. Histogramas de distribuição nos clusters com k = 3 para os parâmetros 'FG%', 'PTS', 'FT%'

Com isso, inferimos que o primeiro cluster dos histogramas exibe menor representatividade populacional, ao passo que o segundo e o terceiro clusters apresentam distribuições mais substanciais, embora em volumes distintos.

Numa abordagem inicial, consideramos comparar a acurácia dos clusters em relação ao parâmetro target por meio do merge entre dois clusters, de forma que pudessem representar uma única classe '0' e o terceiro a classe '1' e pudéssemos comparar qual o melhor merge de clusters. Entretanto, essa estratégia revelou-se inadequada, pois reduziria o número de centróides para k=2, o que não pareceu relevante, dado que a configuração original emprega k=3 e já avaliamos anteriormente com k=2. Diante disso, optamos por realizar a comparação entre dois dos três clusters de maneira independente.

Os resultados, como evidenciado no gráfico a seguir, corroboram a presença de um cluster menos populoso que tende a ser classificado de forma menos precisa. O gráfico foi obtido ao executar 30 iterações para proporcionar uma visualização mais nítida. Nele, observamos que os resultados associados ao Cluster 2 são consistentemente menos satisfatórios, chegando a 3% de acurácia, o que indica uma possível sub-representação ou caracterização inadequada desse grupo. Em contraste, a classificação usando o Cluster 0 e o Cluster 1 exibiu os melhores resultados, atingindo uma acurácia máxima superior a 60%.

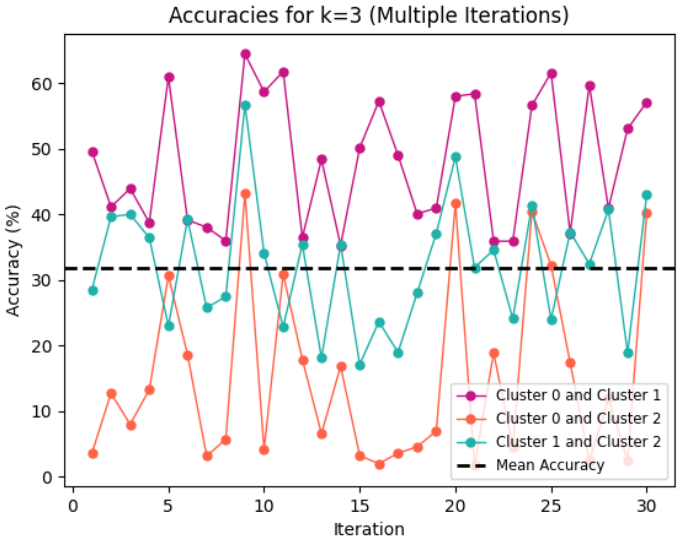


Fig. 10. Gráfico comparativo entre acurácias para pares clusters com k=3

2.2. Comparação com Scikit-learn

De forma análoga ao que experimentamos com o KNN, a utilização da biblioteca Scikit-learn para a implementação do algoritmo KMeans revelou-se notavelmente mais eficiente e intuitiva. Apresentamos dois gráficos, sendo o primeiro da aplicação do KMeans por meio do Scikit e o segundo referente à implementação manual desse algoritmo.

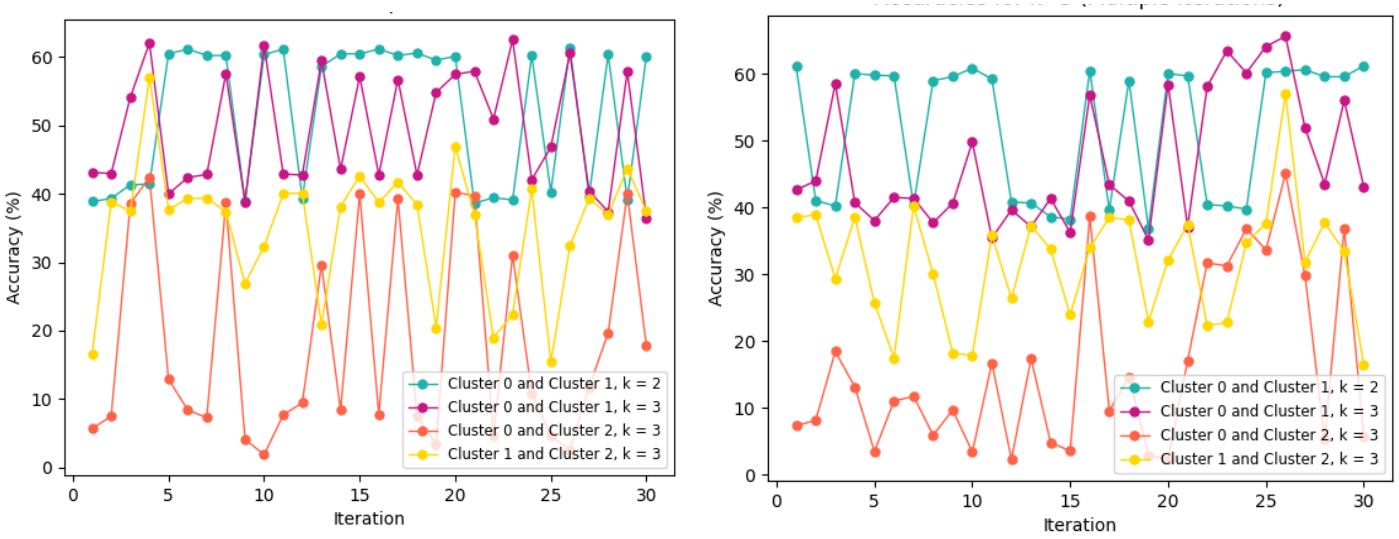


Fig. 11 e 12. Na esquerda, usando o Scikit, e na direita, o implementado. Relação entre acurácia para 30 iterações com k=2 e k=3

Ao analisar a aplicação do algoritmo com k=2, observamos semelhanças nos máximos e mínimos nos dois gráficos, indicando que o desempenho do Scikit-learn com k=2 não apresenta tanta diferença em relação ao código implementado.

Já com k=3, notamos que há uma combinação específica de clusters que assemelha-se consideravelmente àquela obtida com k=2, sugerindo que o conjunto de dados *NBA Rookie Stats* pode ser mais efetivamente representado com k=2. Esta constatação é corroborada pelo gráfico associado ao Scikit-learn, onde as curvas para k=2 e k=3 para o par de clusters (0,1) estão bastante próximas.

Além disso, é importante destacar que as estratégias de inicialização dos centróides são distintas entre as implementações, o que pode ter impacto significativo, resultando, de fato, em uma melhoria leve. No meu código, adotei uma abordagem específica para a inicialização dos centróides ao configurar o parâmetro "init" no algoritmo KMeans do Scikit-learn, atribuindo a ele o valor de "k". Essa escolha proporciona uma inicialização diferenciada, visando potencializar o desempenho do algoritmo.

No Scikit-learn, o parâmetro "n_init" no algoritmo de clustering KMeans determina quantas vezes o algoritmo será executado com diferentes sementes de centróides. Os resultados finais são selecionados entre as melhores execuções, medida em termos de inércia, que representa a soma das distâncias quadradas de cada ponto ao seu centro atribuído. Optei por definir o valor de "n_init" como sendo "k*5", resultando em 10 e 15 execuções. Essa abordagem visa aumentar a probabilidade de encontrar uma solução globalmente melhor, uma vez que o KMeans inicializa os centróides aleatoriamente a cada execução.

Portanto, ambas as implementações do algoritmo KMeans demonstraram desempenho mais satisfatório quando utilizadas com dois clusters, porém, destaca-se que a implementação do Scikit-learn demonstrou uma ligeira superioridade, dado às estratégias internas da biblioteca e otimizações.

Referências

1. RUSSELL, Stuart; NORVIG, Peter. Artificial Intelligence: A Modern Approach. 3. ed. Pearson, 2020.

2. [K-Nearest Neighbours(K-NN) Algorithm from scratch with a hands-on example in R]. Insight IMI: https://insightimi.wordpress.com/2020/03/01/k-nearest-neighboursk-nn-algorithm-from-scratch-with-a-hands-on-example-in-r/

3. [Getting Started with K-Means Clustering in Python]. Domino Data Lab. https://domino.ai/blog/getting-started-with-k-means-clustering-in-python

4. [Confusion Matrix for Your Multi-Class Machine Learning Model]. Towards Data Science. https://towardsdatascience.com/confusion-matrix-for-your-multi-class-machine-learning-model-ff9aa3bf7826

5. [The K-Means clustering process: Three centroids are randomly chosen]. ResearchGate. https://www.researchgate.net/figure/The-K-Means-clustering-process-Three-centroids-are-randomly-chosen-a-Objects-are_fig1_350301258

6. [All About KNN Algorithm]. Medium. https://rekhavsrh.medium.com/all-about-knn-algorithm-6b35a18c2b15

7. [Create Your Own K-Means Clustering Algorithm in Python]. Towards Data Science. https://towardsdatascience.com/create-your-own-k-means-clustering-algorithm-in-python-d7d4c9077670

8. [The k-nearest neighbors algorithm, also known as KNN or k-NN]. IBM. https://www.ibm.com/topics/knn#:~:text=The%20k%2Dnearest%20neighbors%20algorithm%2C%20also%20known%20as%20KNN%20or,of%20an%20individual%20data%20point.