K Nearest Neighbors

*

Arthur Felipe Reis Souza
Electrical Engineering Department,
Federal University of Minas Gerais,
Belo Horizonte, Brazil
arthurfreisouza@gmail.com

Antônio de Pádua Braga and Frederico Gualberto Ferreira Coelho
Electrical Engineering Department,
Federal University of Minas Gerais,
Belo Horizonte, Brazil
apbraga@cpdee.ufmg.br, fredgfc@ufmg.br

October 13, 2024

Abstract

Esse relatório tem por objetivo demonstrar o algoritmo de Machine Learning K-Nearest Neighbors (KNN), suas vantagens, desvantagens e aplicações.

1 Introdução

O algoritmo K-Nearest Neighbors (KNN) é amplamente utilizado em Machine Learning tanto para tarefas de classificação quanto de regressão. No contexto de classificação, ele se baseia em um número K de vizinhos mais próximos e nos rótulos desses vizinhos para tomar a decisão. Ao classificar um novo ponto P, primeiro calcula-se a distância entre esse ponto e todos os outros do conjunto de dados. Em seguida, considera-se os rótulos correspondentes aos K vizinhos mais próximos, definidos pelas menores distâncias, para determinar o rótulo do novo ponto. No caso de classificação, o ponto classificado será decidido com base na moda dos valores, enquanto no caso de regressão, o algoritmo pegará a média dos K rótulos mais próximos.

*Insert applicable funding agency here. If none, delete this.

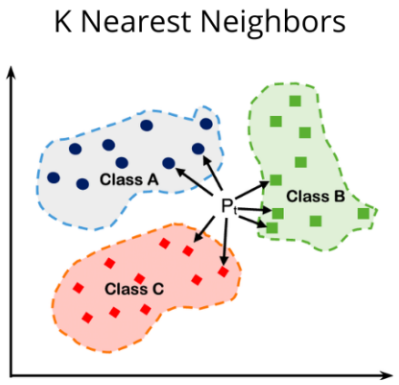

Figure 1: Algoritmo KNN.

O algoritmo K-Nearest Neighbors (KNN) é um método baseado em distância, o que o torna sensível tanto à normalização dos dados quanto à escolha da métrica de distância utilizada. Existem diversas métricas que podem ser aplicadas neste algoritmo; neste relatório, optamos pela distância euclidiana, que é expressa pela equação $\sqrt{\sum_{i=1}^{n}(x_i - y_i)^2}$.

Um fator crucial que influencia o desempenho do classificador é a escolha do número de vizinhos mais próximos, representado pelo valor de K. Valores baixos de K tendem a resultar em um desempenho insatisfatório devido ao fenômeno de overfitting, já que o algoritmo considera apenas o rótulo mais próximo. Por outro lado, valores altos de K também podem comprometer o desempenho do classificador, pois o algoritmo leva em conta todo o conjunto de dados, diluindo a relevância dos rótulos mais próximos e sendo também influenciado pelo viés dos dados, ou seja, se uma das classes tem mais amostras ela irá influenciar mais no desempenho do modelo. Portanto, é fundamental determinar um valor ideal de K. Isso pode ser alcançado por meio do algoritmo Grid Search, que explora uma grade de diferentes valores de K e avalia o desempenho do classificador utilizando o método de Cross-Validation.

2 Geração dos dados

Os dados foram gerados com base em distribuições normais, caracterizadas pela média e pelo desvio padrão. Um alto desvio padrão pode resultar na sobreposição das amostras de ambas as classes, dificultando a classificação pelo algoritmo. Em contraste, um baixo desvio padrão tende a gerar classes mais linearmente separáveis, o que favorece o desempenho do classificador. Abaixo,

apresentamos imagens que ilustram as duas situações: na primeira, não há superposição entre as classes, enquanto na segunda, um alto desvio padrão resulta em sobreposição.

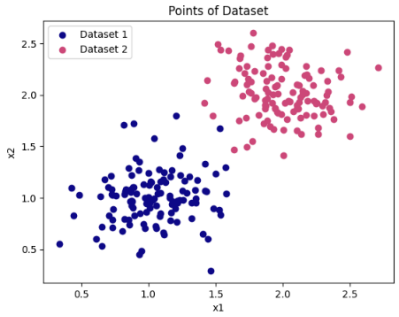

Figure 2: Dados com baixo desvio padrão $\sigma = 0{,}25$.

É notório acima que as classes são linearmente separáveis.

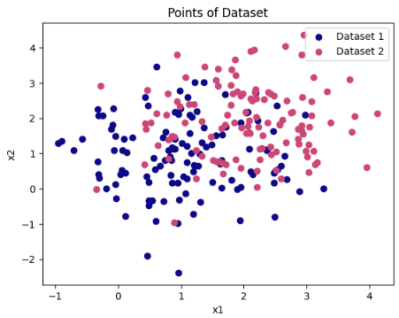

Figure 3: Dados com alto desvio padrão $\sigma = 1$.

Acima é possível observar que com o desvio padrão σ mais alto há a superposição das duas classes, influenciando no desempenho do classificador.

3 Função KNN

Esta seção tem como objetivo descrever a função do KNN tradicional, que é considerado mais simples, pois não possui parâmetros que previnem o underfitting e o overfitting. A seguir, apresentamos a implementação em Python da função do KNN, acompanhada de uma descrição detalhada, passo a passo, de seu funcionamento.

```python
def knn(matrix: np.ndarray, new_points: np.ndarray, K: int = None) -> dict:
    if K is None or K <= 0 or K/2 == 0:
        raise ValueError("The K value must be a positive integer and odd.")

    dic = {}
    count = 0

    for point in new_points:
        count += 1
        # Calculate the Euclidean distance between the point and all points in the matrix
        distances = np.linalg.norm(matrix[:, [0, 1]] - point, axis=1)

        # Find the indices of the K nearest neighbors
        nearest_indices = np.argsort(distances)[:K]

        # Sum the labels (column 2) of the nearest neighbors
        sum_labels = np.sum(matrix[nearest_indices, 2])

        # Assign label based on the sum of the nearest neighbors' labels
        dic[f"Points {count}"] = 1 if sum_labels > 0 else -1

    return dic
```

Figure 4: Implementação da função do KNN.

1. **Calcular a distância entre o ponto de teste e todos os pontos do conjunto de dados.** Para isso, pode-se utilizar a distância Euclidiana ou outra métrica de distância.

2. **Selecionar os K vizinhos mais próximos.** A partir das distâncias calculadas, escolha os K pontos que têm as menores distâncias em relação ao ponto de teste.

3. **Contar os rótulos dos K vizinhos selecionados.** Após selecionar os vizinhos, verifique os rótulos correspondentes a esses pontos.

4. **Determinar o rótulo do ponto de teste.** O rótulo do ponto de teste será o que aparecer com mais frequência entre os K vizinhos. Caso haja um empate, pode-se usar uma estratégia para decidir, mas no geral há uma restrição nos valores de K, sendo o mesmo um valor ímpar.

5. **Retornar o rótulo classificado.** Por fim, a função deve retornar o rótulo determinado para o ponto de teste.

4 Geração de novos dados

Novos dados foram gerados para realizar a inferência do modelo, abaixo estão os novos dados gerados considerando $\sigma = 0.25$ e $\sigma = 1$.

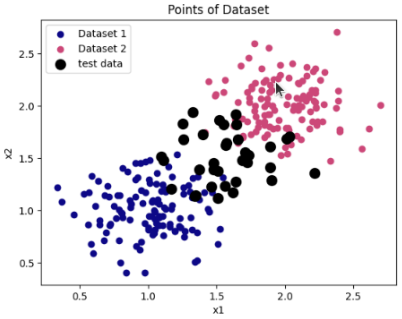

Figure 5: Gráfico com novos dados para a classificação considerando $\sigma = 0.25$.

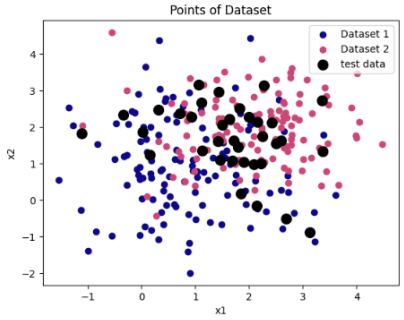

Figure 6: Gráfico com novos dados para a classificação considerando $\sigma = 1$.

A principal ideia é utiliar esses novos dados de testes para realizar a inferência do classificador e avaliar como o mesmo os classificará, plotando a classificação resultante e também a superfície de separação ao variar os parâmetros K.

5 Resultados

Os resultados abaixo mostram os resultados do classificador sobre os dados de teste, utilizando k = 3.

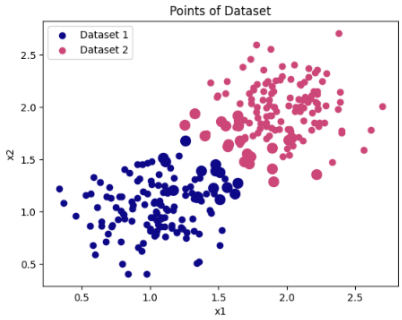

Figure 7: Resultado da classificação com $\sigma = 0.25$ e K = 3.

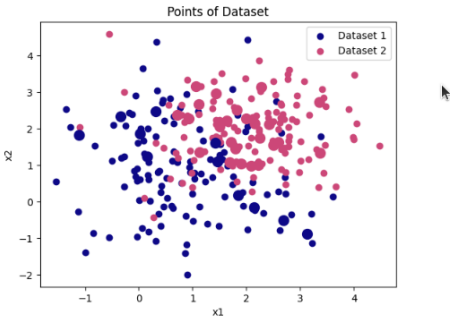

Figure 8: Resultado da classificação com $\sigma = 1$ e K = 3.

A variação do valor de K, tanto para valores baixos quanto para altos, não resultou em diferenças significativas na classificação dos dados de teste. No entanto, essa variação permite analisar como a superfície de separação se altera, possibilitando a identificação de fenômenos como overfitting e underfitting. A seguir, apresentamos gráficos que utilizam K = 1 e $\sigma = 0.25$, evidenciando um potencial overfitting do modelo. Embora o overfitting não seja claramente perceptível devido à linearidade dos dados, o modelo ainda consegue discriminar bem as classes, mesmo com valores baixos de K. Em contraste, ao adotar um valor maior de K, como K = 3, a superfície de separação torna-se mais estável, contribuindo para a mitigação do overfitting.

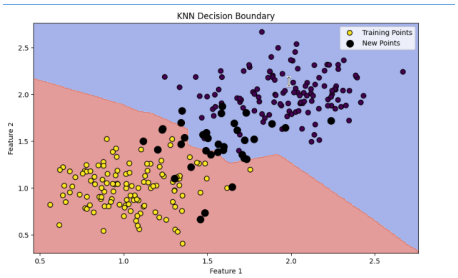

Figure 9: Superfície de separação com $\sigma = 0.25$ e K $= 1$.

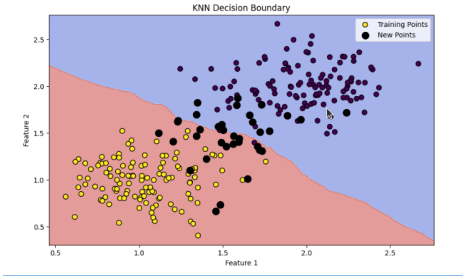

Figure 10: Superfície de separação com $\sigma = 0.25$ e K $= 3$.

No fito de observar como o overfitting do modelo, aumenta-se o desvio padrão σ para 1, onde haverá uma maior superposição entre os dados, tornando o mesmo mais visível. Abaixo estarão as superfícies de separação para valores de K $= 1$, K $= 3$, K $= 7$ e K $= 51$.

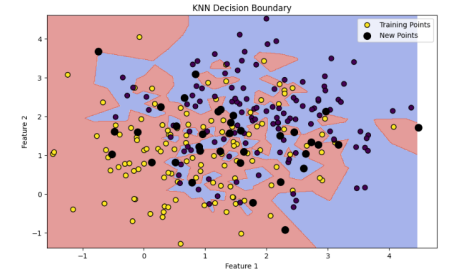

Figure 11: Superfície de separação com $\sigma = 1$ e K $= 1$.

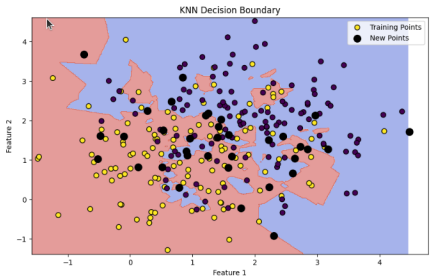

Figure 12: Superfície de separação com $\sigma = 1$ e K = 3.

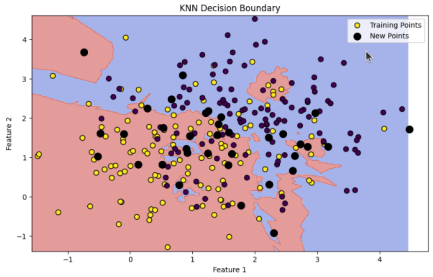

Figure 13: Superfície de separação com $\sigma = 1$ e K = 7.

Pode se notar, ao analisar os dados acima é que, aumentando o valor de K reduz o efeito do overfitting nesse conjunto de dados os quais estão superpostos com $\sigma = 1$. O efeito do overfitting agora é claramente visto com K = 1, K = 3 e também K = 7, enquanto o efeito do mesmo é reduzido consideravelmente ao se utilizar um alto de valor de K, sendo ele K = 51. Vale ressaltar também que a escolha do valor de K depende da base de dados e também da superposição do mesmo, para os dados utilizados, os quais estão superpostos, o valor de K que irá mitigar o overfitting tem um valor alto, mas para base de dados que são linearmente separáveis, o valor de K tenderá a ser menor.

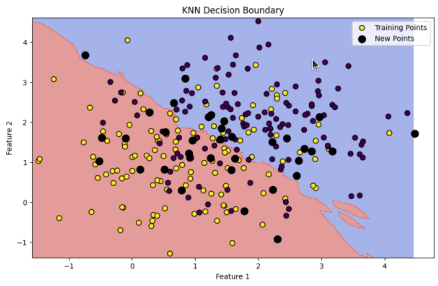

Figure 14: Superfície de separação com $\sigma = 1$ e K = 51.

6 Conclusão

Portanto, ao finalizar o exercíco é possível observar e estudar o algoritmo K Nearest Neighbors, o qual é totalmente sensível ao valor de K e também a métrica de distância utilizada. O desempenho do modelo depende totalmente de como os dados estão superpostos, e o valor de K dependerá também dessa informação. Como pode se observar na figura acima, baixos valores de K em um conjunto de dados superposto acarreta em overfitting, enquanto para um conjunto não superposto não acarreta, pois os dados são facilmente linearmente separáveis. Outro fator que influencia bastante o desempenho do modelo é se os dados estão ou não normalizados, considerando-se que é um modelo baseado em distância.