Ex 1 - KNN

Aluno: Felippe Veloso Marinho

Nº de Matrícula: 2021072260

KNN - K - Nearest Neighbors

- Classificador de distância baseado em **vizinhança**.
- Dado um ponto novo, calcula-se sua distância para todos os outros exemplos.
- O ponto é classificado segundo a **maioria dos k vizinhos mais próximos**.

1. Cálculo da distância: Para cada nova instância a ser classificada, o algoritmo calcula a distância para todas as instâncias no conjunto de treinamento.

2. Identificação dos k vizinhos mais próximos:O algoritmo seleciona os k exemplos mais próximos à nova instância, onde k é um número inteiro positivo definido pelo usuário.

3. Classificação/Regressão:

Classificação: A nova instância é classificada com base na classe mais comum entre seus k vizinhos.

Regressão: O valor da nova instância é previsto calculando a média ou mediana dos valores dos seus k vizinhos.

Implementação

A implementação do KNN consiste em:

- Para cada ponto de consulta, calcula distância a **todos** os pontos de treino.
- Pega os **k's** mais próximos.
- **Votação** pela classe majoritária (empate → menor rótulo, para ser determinístico).
- Usaremos inicialmente **distância euclidiana ao quadrado** (mesma ordem dos vizinhos, sem `sqrt`).

Criação das gaussianas bidimensionais em R

O problema em análise se utiliza da geração de dois clusteres para o uso do KNN para diferentes valores de k. Para cada experimento a superfície de separação será traçada.

```
# Geração de dois clusteres (gaussianas 2D) com 100 pontos cada,
# escaladas por s1 e s2 e transladadas para os centros (2,2) e (4,4)

### R
s 1<−0. 3
s 2<−0. 3
nc<−100
xc1<−matrix(rnorm( nc∗2 ) , ncol=2)∗s 1 +
t (matrix( c ( 2 , 2 ) ,nrow=2,ncol=nc ) )
xc2<−matrix(rnorm( nc∗2 ) , ncol=2)∗s 2 +
t (matrix( c ( 4 , 4 ) ,nrow=2,ncol=nc ) )


### Python
import numpy as np

# Parâmetros
s1 = 0.3
s2 = 0.3
nc = 100

# Gerador de números aleatórios (opcional: semente para reprodutibilidade)
rng = np.random.default_rng(42)

# Cluster 1 ~ N((2,2), s1^2 I)
xc1 = np.random.randn(nc, 2) *  s1 + np.array([2.0, 3.0])

# Cluster 2 ~ N((4,4), s2^2 I)
xc2 = np.random.randn(nc, 2) *  s1 + np.array([3.0, 3.0])
```

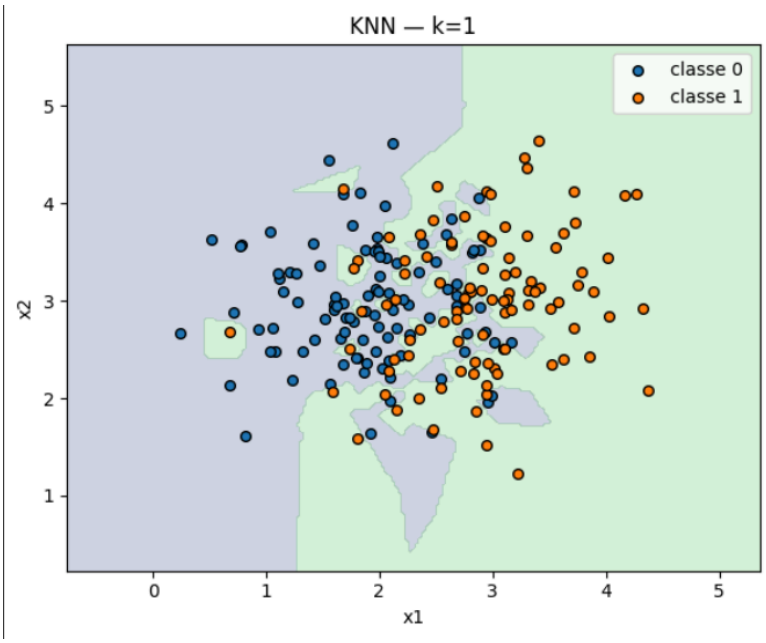
```
plt.scatter(xc1[:,0], xc1[:,1], label='Cluster 1')
plt.scatter(xc2[:,0], xc2[:,1], label='Cluster 2')
plt.legend()
plt.xlabel('x1')
plt.ylabel('x2')
plt.title('Clusteres - Gaussianas 2D')
plt.show()
```

Variação dos K’s

Os principais pontos para se notar na variação dos K’s:

- Se a superfície de separação é satisfatória;
  - Se ela separa corretamente os dados;
  - Se ela “invade” muito o outro grupo de dados (overfitting);
  - Se ela não se adapta corretamente aos conjuntos de dados (underfitting);

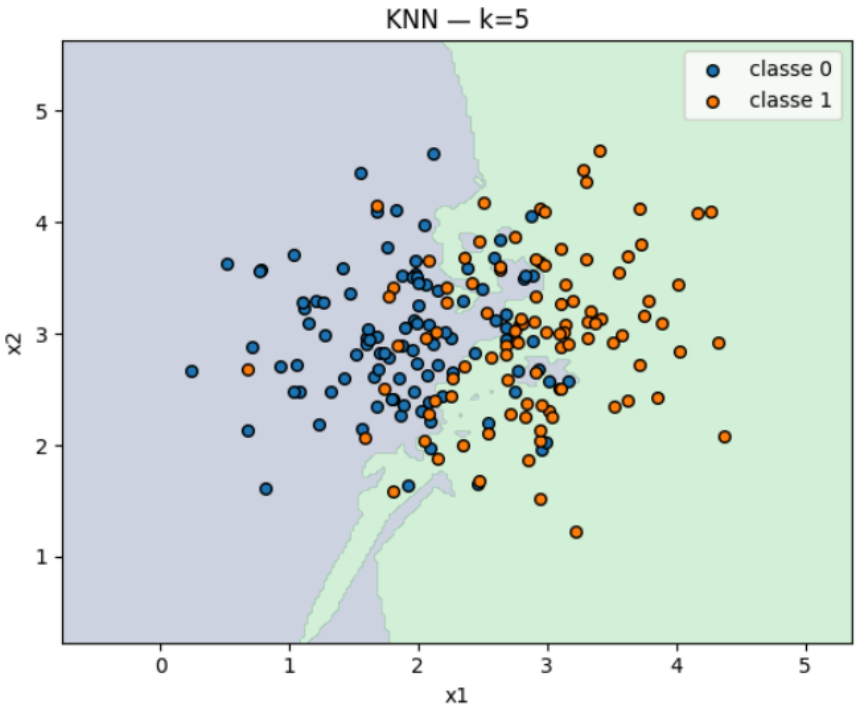
Para K = 1



O classificador apresentou uma baixa generalização visto que ao analisar o ponto laranja próximo ao ponto (1,3) ele desenha a superfície de separação mesmo com

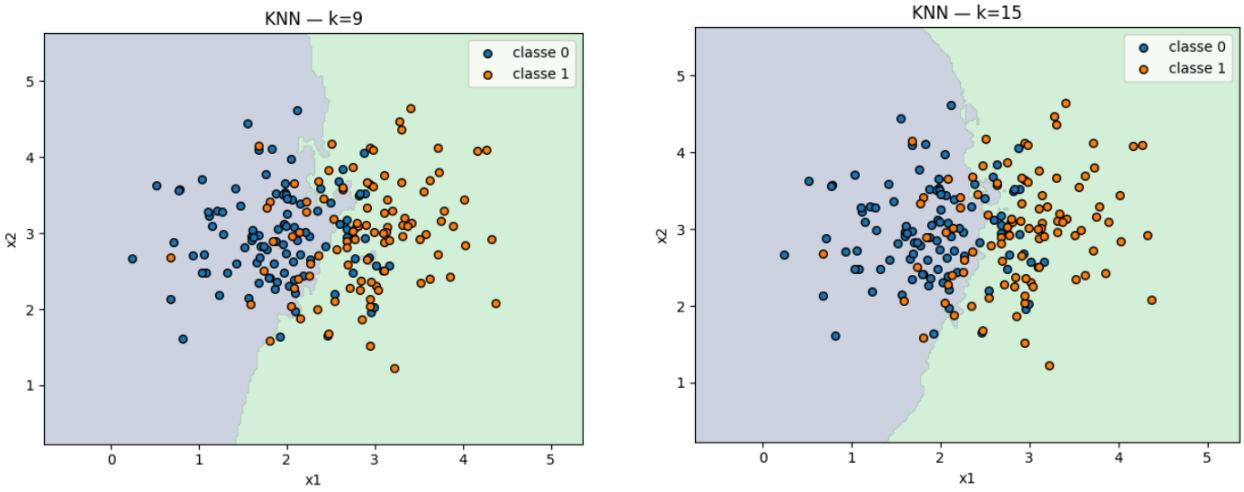
a probabilidade de ser um ponto azul ser bem mais alta.

Para K = 5



Com o K = 5, podemos ver que a separação dos dados sugere menos overfitting em relação ao resultado anterior. Sendo uma opção mais generalizável.

Para K = 9 e 15



Temos resultados bem parecidos, o que sugere que o número de K’s suficientes para uma boa classificação esteja dentro desse limiar.