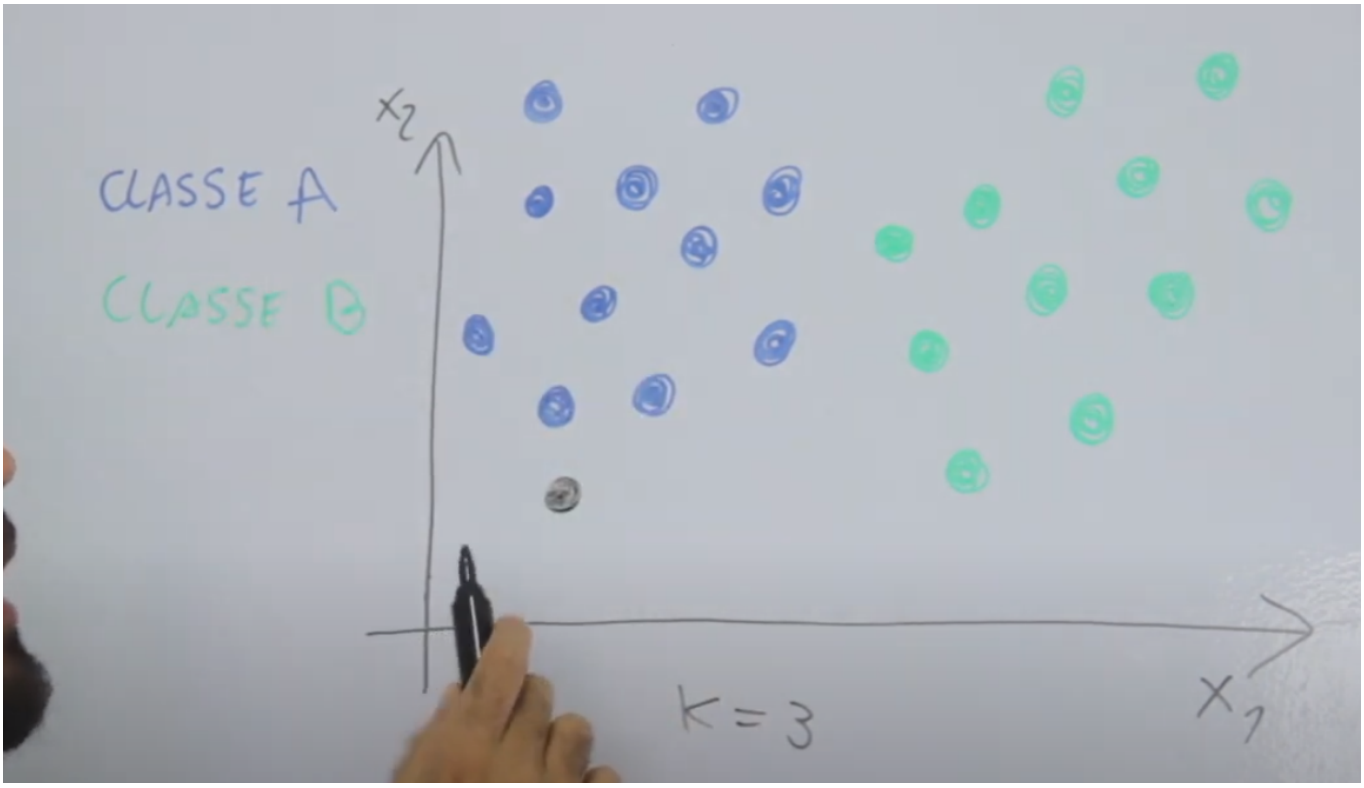
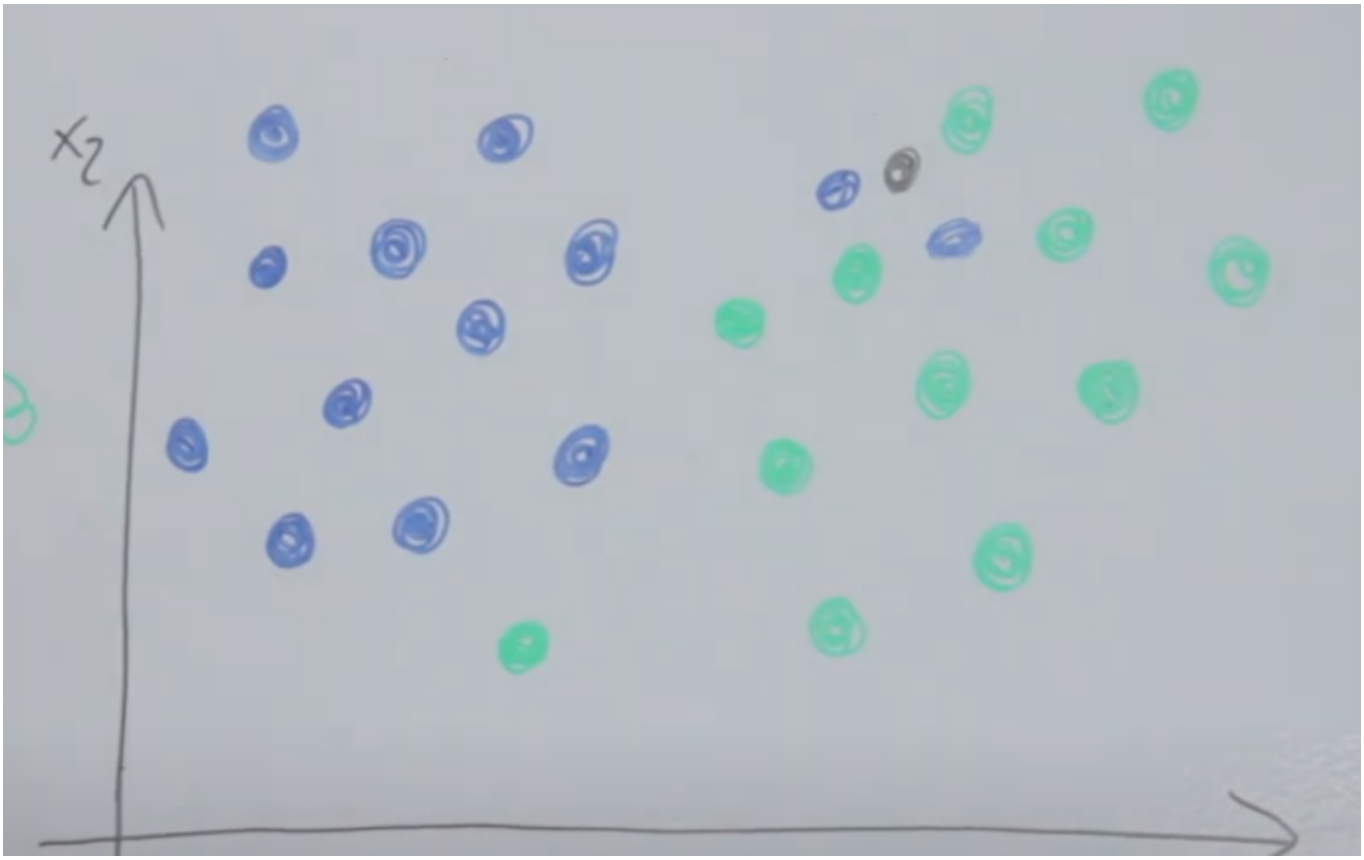
Algoritmo

O algoritmo KNN ou K-vizinhos próximos ou K-Nearest Neighbors é muito simples na realidade, como mostrado abaixo, podemos dividir nossos dados em clusters e a partir de novos dados, classificá-los nos seus devidos clusters:



Nesse exemplo, temos duas classes e um K = 3, o que significa a quantidade de vizinhos que será considerado para classificar um novo exemplo. Observando a bolinha preta, o algoritmo irá analisar a distância de todos os outros pontos do gráfico, e selecionar os 3 mais próximos. A partir disso, deve-se verificar qual classe é predominante entre os 3 mais próximos e definir então a classe do novo dado inserido.

É importante a definição de K, pois em alguns casos, um K muito pequeno pode acabar levando a uma classificação errada para a classe do novo dado, como podemos ver abaixo:



Com um K = 3, o ponto preto será classificado como azul, pois há 2 pontos azuis e um verde próximo a eles o que visualmente é errado, pois a maioria dos dados da direita são verdes, e somente um K > 3 sanaria o problema dos outliers. Por outro lado, um K muito grande pode ser um problema, caso uma das classes não tenham muitos exemplos, uma vez que esses pontos seriam atribuídos a outras com mais dados e não a classe minoritária.

Cálculo da distância:

O algoritmo pode calcular a distância entre os pontos de diversas formas, abaixo estão as implementações existentes no scikit-learn:

Metrics intended for real-valued vector spaces:

identifier	class name	args	distance function
"euclidean"	EuclideanDistance	•	`sqrt(sum((x - y)^2))`
"manhattan"	ManhattanDistance	•	`sum(|x - y|)`
"chebyshev"	ChebyshevDistance	•	`max(|x - y|)`
"minkowski"	MinkowskiDistance	p	`sum(|x - y|^p)^(1/p)`
"wminkowski"	WMinkowskiDistance	p, w	`sum(|w * (x - y)|^p)^(1/p)`
"seuclidean"	SEuclideanDistance	V	`sqrt(sum((x - y)^2 / V))`
"mahalanobis"	MahalanobisDistance	V or VI	`sqrt((x - y)' V^-1 (x - y))`