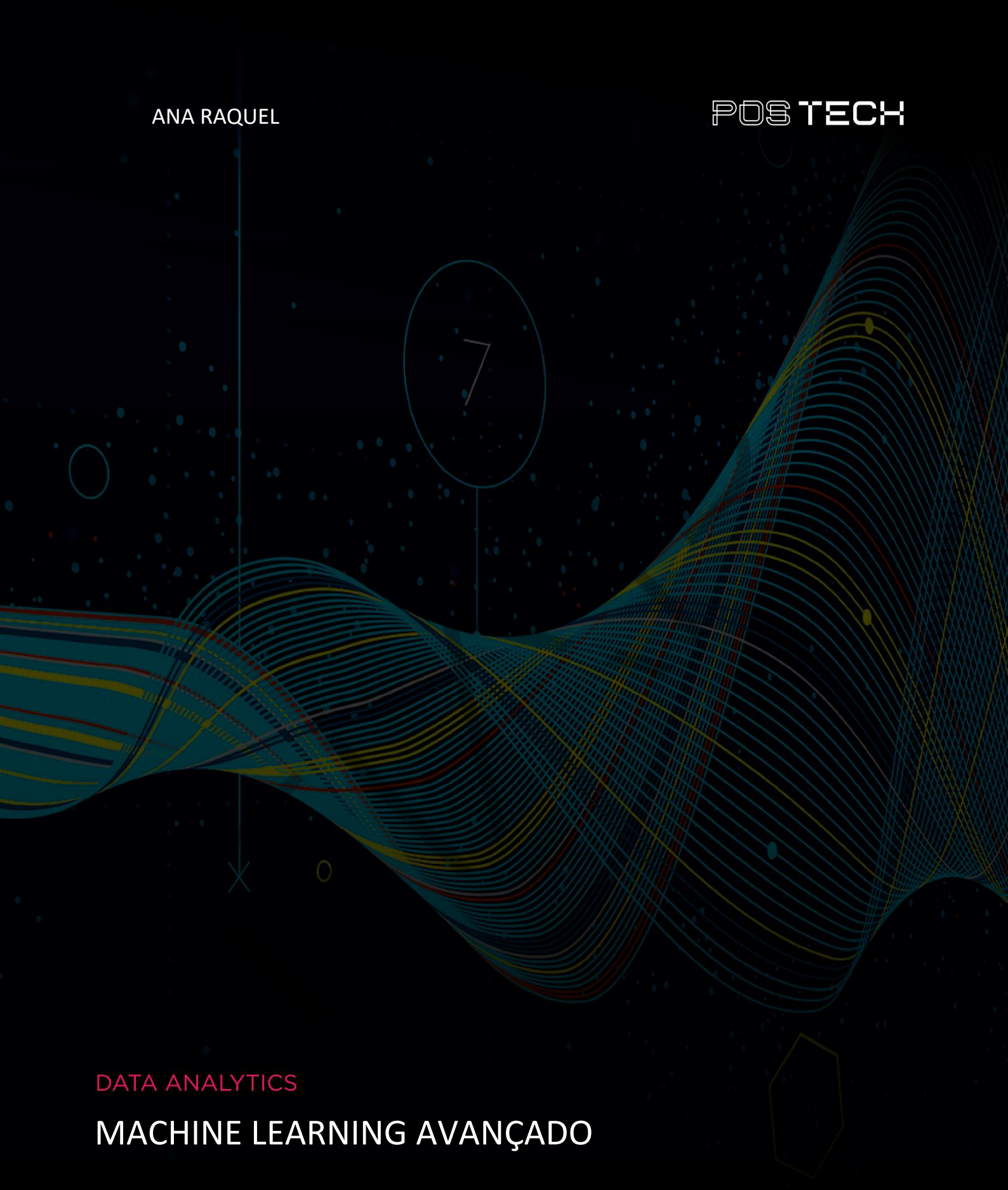
ANA RAQUEL



DATA ANALYTICS

MACHINE LEARNING AVANÇADO

AULA 02

SUMÁRIO

O QUE VEM POR AÍ?

Você aprendeu como identificar problemas supervisionados de classificação, e agora está na hora de aprender mais sobre os principais tipos de algoritmos de classificação e suas utilidades em Machine Learning.

CONHEÇA SOBRE O ASSUNTO

Nesta aula, vamos conhecer os três algoritmos poderosos em Machine Learning, que podem ser uma ótima opção de ferramenta de trabalho para resolver problemas reais.

K-Nearest Neighbors

Vamos começar pelo **KNN**! O algoritmo KNN (K-Nearest Neighbors, em português: K-Vizinhos mais próximos) é um algoritmo **supervisionado de classificação,** que basicamente classifica dados com base na **distância entre pontos (dados) mais próximos no espaço** (por isso conhecemos como o algoritmo dos vizinhos mais próximos). O KNN é um algoritmo **não paramétrico**, o que significa que ele não possui uma função matemática definida quando analisamos o conjunto de dados sob uma perspectiva total. Quando analisamos o total do conjunto dos dados, temos pontos (dados) plotados no plano cartesiano, tal que não há como definir uma função que expressa como esses dados se comportam, ou seja, não há uma lei de formação que descreve qual será a coordenada do ponto que representa o dado. Mas, por outro lado, se analisarmos apenas uma parte desse conjunto de dados, como por exemplo dois pontos, podemos utilizar um importante resultado da geometria analítica que nos permite calcular a distância entre dois pontos.

O algoritmo funciona com a seguinte lógica: caso a maioria de pontos próximos seja de uma certa classe “x”, o algoritmo entende que essa classe é predominante naquele espaço e atribui essa classe para a entrada de dado a ser prevista. Os algoritmos possuem alguns parâmetros que podem ser configurados durante a criação do modelo para ajustar a lógica do funcionamento do algoritmo. Para o KNN, utilizamos o parâmetro “k” para definir a quantidade de vizinhos mais próximos considerados durante o processo de classificação. Os resultados previstos dependem de como as características foram escalonadas, como a similaridade foi medida e qual o tamanho em que “k” foi definido.

Vamos entender como funciona na prática?

Vamos supor que eu busque classificar se um candidato ou candidata pode ou não ser contratado(a) com base em algumas características, tais como: **percentual de ensino médio, superior e especialização, experiência de trabalho e ofertas**

salariais para os profissionais colocados. O algoritmo KNN classificará o candidato com **base na proximidade de comportamento** em relação às demais pessoas da base que possuem esse mesmo comportamento similar. Se este candidato possui características semelhantes à “n”, em que n é o número de vizinhos próximos, podemos classificá-lo como contratado ou não.

Vamos imaginar que o ponto amarelo do gráfico – “KNN 1” foi posicionado para o algoritmo KNN classificar se o(a) candidato(a) pode ser contratado(a) ou não. Os pontos em vermelho são da classe “não contratado(a)” e os azuis da classe “contratado(a)”:

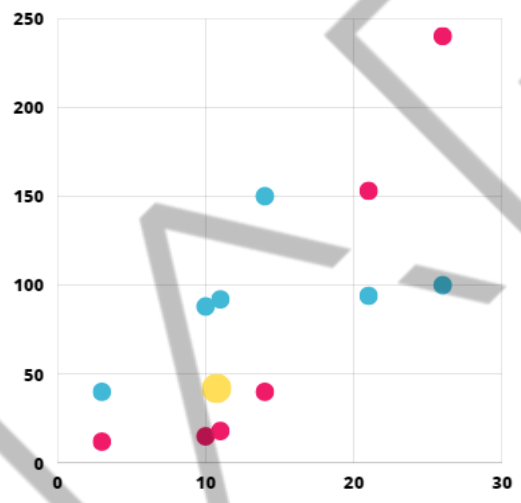


Figura 1 – KNN 1
Fonte: Elaborado pela autora (2023)

Se configuramos o algoritmo para classificar dados com base nos três vizinhos mais próximos, poderíamos entender que, com base na distância entre os pontos (dados) vizinhos, esse(a) candidato(a) pertence à classe “não contratada".

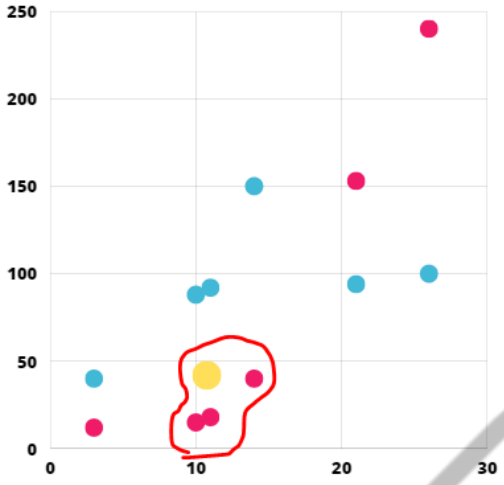


Figura 2 – KNN 2
Fonte: Elaborado pela autora (2023)

Métricas de distância

Para o algoritmo KNN realizar a classificação dos dados, podemos utilizar algumas métricas de similaridade (proximidade), que pode ser determinada utilizando uma métrica de distância. A métrica de distância mede quão longe dois registros estão um do outro. A seguir, vamos aprender algumas das métricas.

A métrica mais popular é a distância Euclidiana. A distância Euclidiana é a distância geométrica clássica que utiliza o Teorema de Pitágoras:

Distância Euclidiana: $\sqrt{((x_1 - x_2)^2 + (y_1 - y_2)^2)}$.

A distância Euclidiana oferece vantagens computacionais especiais. Isso é ainda mais importante para grandes conjuntos de dados, já que o KNN envolve a comparação em pares K x n, em que n é o número de linhas.

Outra métrica comum é a distância de **Manhattan**. A distância de Manhattan é a distância entre dois pontos, traçada em uma única direção de cada vez (por exemplo, caminhar ao redor de quarteirões).

Distância Manhattan: $|x_1 - x_2| + |y_1 - y_2|$.

Realizando uma analogia entre a diferença entre essas duas distâncias, vamos imaginar uma rota de São Paulo para Curitiba, para dois veículos: uma para um carro e outra para um avião. A Distância Euclidiana seria o segmento de uma reta na qual indicaria uma possível rota de avião (na qual não haveria preocupação com estradas, já que é um meio de transporte aéreo, e geometricamente seria a hipotenusa de um triângulo) e a Distância Manhattan seria um segmento de retas na vertical, quanto na horizontal, semelhante a uma rota de carro, obedecendo as rotas das estradas (geometricamente, seriam a soma dos catetos).

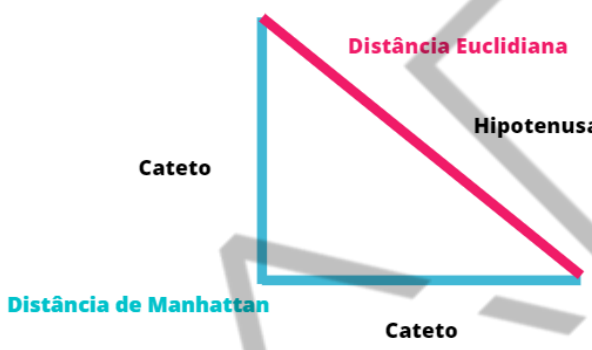


Figura 3 - Métricas de distância
Fonte: Elaborado pela autora (2023)

Outras métricas

Para saber sobre quais tipos de distâncias estão disponíveis no Sklearn, acesse a página que contém algumas métricas. Não se preocupe, pois a biblioteca do Sklearn facilita muito o trabalho dos(as) cientistas de dados em relação à configuração dos parâmetros do algoritmo. A escolha de K e a escolha da métrica de distância pode ser utilizada de forma bem simples, conforme o código exemplifica:

```
modelo_classificador = KNeighborsClassifier(n_neighbors=5)
modelo_classificador.fit(x_train, y_train)
```

Por padrão, o Sklearn deixa a métrica de distância do tipo minkowski, que também podemos conhecer como distância euclidiana. Perceba que o número de k foi definido como cinco, ou seja, a classificação será realizada com base nos cinco pontos mais próximos.

Escolhendo o valor de k

Mas como eu defino o melhor número de k? Será que existe uma fórmula? Não, mas existe uma técnica chamada otimização de hiperparâmetro! Normalmente, atribuímos um valor aleatório para k e vamos realizando a análise de erro de classificação (basicamente testamos vários valores para k), utilizamos um gráfico para auxiliar na identificação do menor erro possível. Nesse teste, a validação cruzada nos dados de treinamento pode auxiliar muito e prevenir overfiting dos dados (vamos aprender sobre overfiting ainda nessa aula, não se preocupe!). Nas próximas aulas você também irá aprender sobre validação cruzada.

```
error = []

for i in range(1, 10): #range de tentativas para k

    knn = KNeighborsClassifier(n_neighbors=i)# aqui definimos  o k

    knn.fit(x_train, y_train) #treinando para encontrar o erro

    pred_i = knn.predict(x_test_escalonado) #armazenando as previsões

   error.append(np.mean(pred_i != y_test)) #armazenando o valor do erro
médio na lista de erros
```

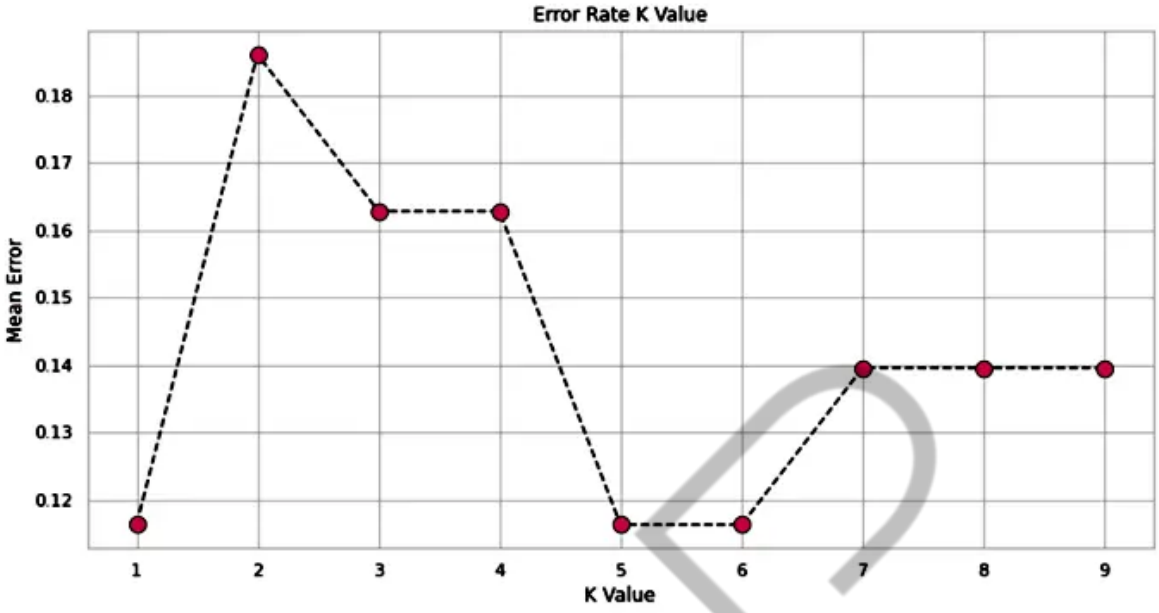


Figura 4 - Encontrando o melhor valor de K
Fonte: Elaborado pela autora (2023)

Em linhas gerais, se o k for muito baixo, podemos sobreajustá-lo incluindo o ruído nos dados. Valores de k maiores oferecem uma suavização que reduz o sobreajuste nos dados de treinamento. Por outro lado, se o k for muito alto, podemos supersuavizar os dados e perder a habilidade dos KNN de capturar a estrutura local dos dados (uma de suas muitas vantagens).

O k que melhor se equilibra entre sobreajuste e supersuavização costuma ser determinado pelas métricas de precisão e, especialmente, precisão com dados de retenção ou validação. Não existe regra geral para definir o melhor valor de k, depende muito da natureza dos dados. Para dados altamente estruturados com pouco ruído, valores menores de k funcionam melhor. Para dados ruidosos com menos estrutura, são adequados valores maiores para k. Geralmente os valores de k ficam entre 1 até 20, e é comum se escolher um número ímpar para não haver empates.

SVM (Support Vector Machine)

Antes de começarmos a conhecer sobre o algoritmo SVM, te convido a um questionamento sobre a figura 5 – "Encontrando a melhor reta". Qual é a reta que melhor separa os dados? A, B ou C?

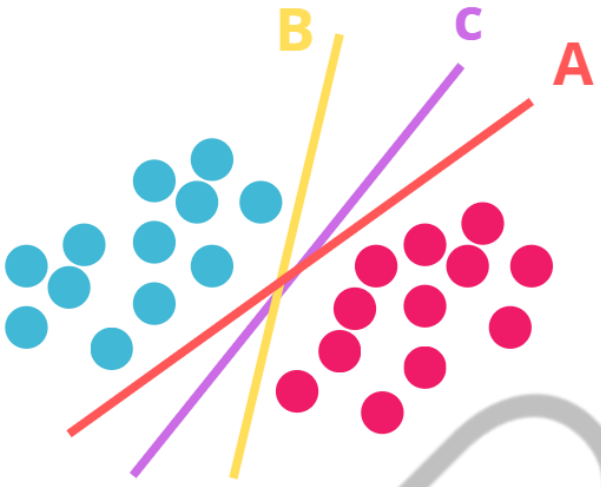


Figura 5 - Encontrando a melhor reta
Fonte: Elaborado pela autora (2023)

Se você disse que é a reta A ou B, essa não seria a resposta ideal. Podemos considerar a reta C como a melhor reta que separa os dados. Mas por quê?

Analise bem o espaçamento entre cada reta e os pontos (dados), podemos observar que a reta C possui um distanciamento maior em relação às retas A e B:

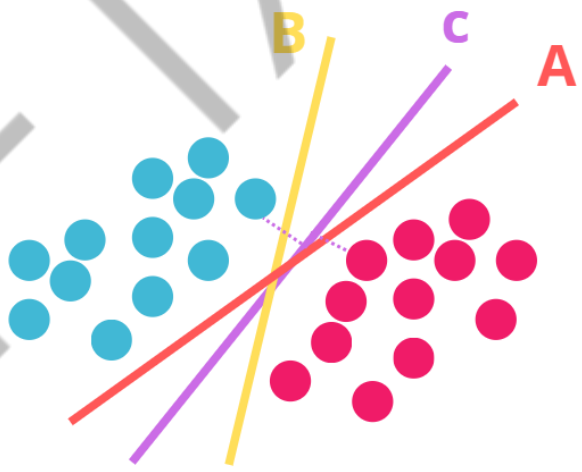


Figura 6 - Distância entre os dados e as retas
Fonte: Elaborado pela autora (2023)

Podemos identificar que a reta C consegue separar melhor os dados em classes azuis e vermelhas do que as demais retas, utilizando uma margem de classificação mais larga do que as demais. Veja que as retas A e B podem classificar muito bem uma das classes, mas podem errar em outra. Imagine se

na reta A acrescentarmos mais um ponto; poderíamos, então, ter um erro e o ponto ser considerado da classe azul por ultrapassar a reta:

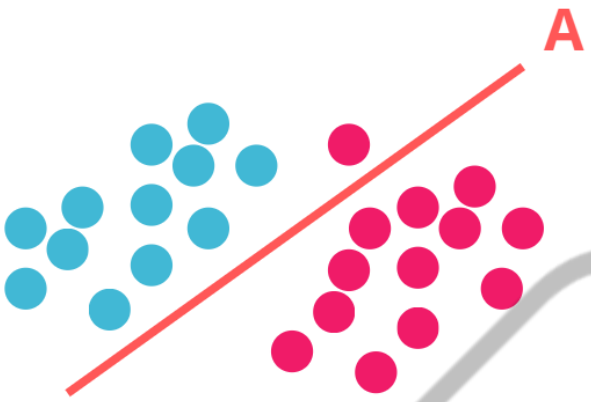


Figura 7 - Reta A
Fonte: Elaborado pela autora (2023)

Basicamente, o SVM utiliza essas margens de classificação para distinguir classes. Quanto mais larga as margens, mais dados podemos considerar dentro de uma classe.

O algoritmo SVM também é um algoritmo poderoso de classificação supervisionado, capaz de realizar classificação lineares e não lineares, de regressão e até mesmo detecção de outliers. As SVMs são particularmente adequadas para a classificação de conjuntos de dados complexos, porém de pequeno ou médio porte. O algoritmo SVM possui o objetivo de separar os dados por meio de hiperplanos, maximizando a margem de separação.

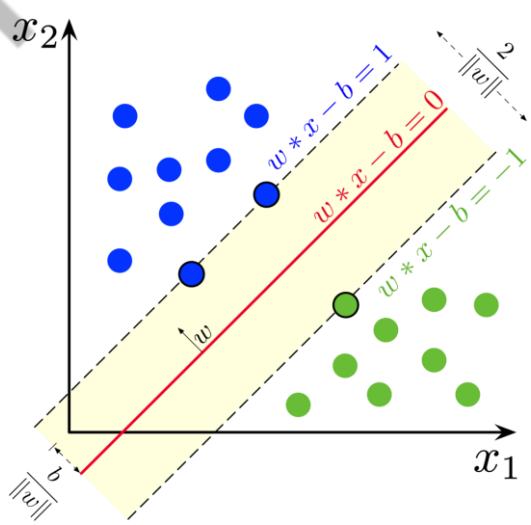


Figura 8 - Modelo SVM
Fonte: Elaborado pela autora (2023)

É importante destacar que os modelos de SVM são sensíveis às escalas dos dados, então pode ser uma boa estratégia utilizar técnicas de normalização ou padronização nos dados antes de aplicar o modelo:

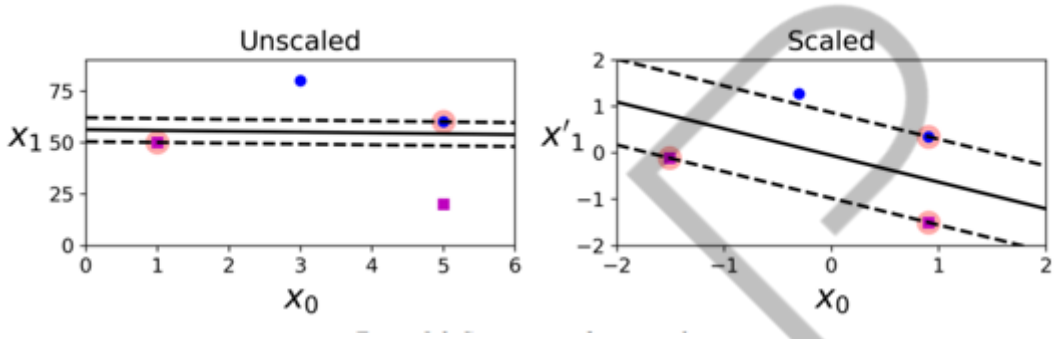


Figura 9 - Modelo SVM
Fonte: Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow (2023)

Classificação de margem suave

Na definição do tamanho da margem para separar os dados, estamos na busca do equilíbrio entre manter a via o mais larga possível e limitar as violações de margem. É preferível utilizar um modelo mais flexível a fim de evitar esses problemas.

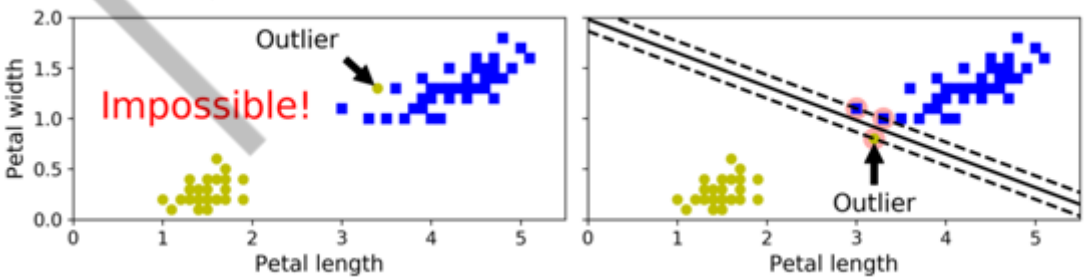


Figura 10 - Sensibilidade da margem aos outliers
Fonte: Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow (2023)

Nas classes SVM do Scikit-Learn, você consegue controlar esse equilíbrio ao utilizar o hiperparâmetro C: um valor menor de C leva a uma via mais larga

(um espaço maior para considerar os dados de uma certa classe), mas com mais violações das margens.

Observe a figura 11 – "Valor alto para o hiperparâmetro C". Ao utilizar um valor alto de C, o classificador faz menos violações na margem, mas fica com uma margem menor.

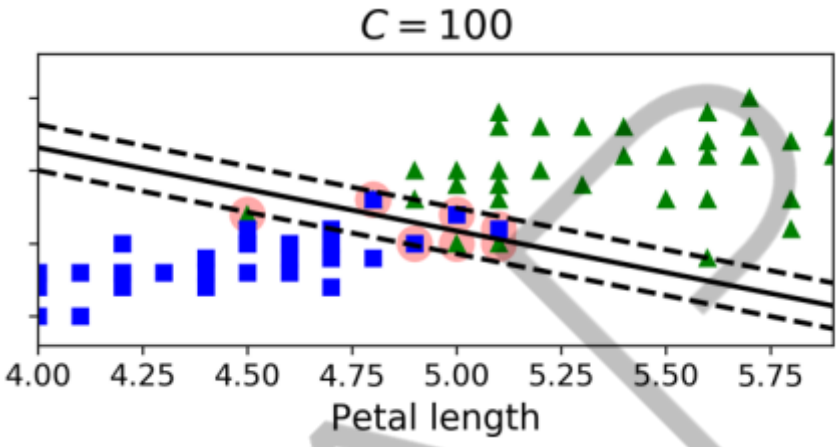


Figura 11 - Valor alto para o hiperparâmetro C
Fonte: Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow (2023)

Ao utilizar um **valor baixo para C**, a **margem fica muito maior**, mas muitas instâncias ficam na via:

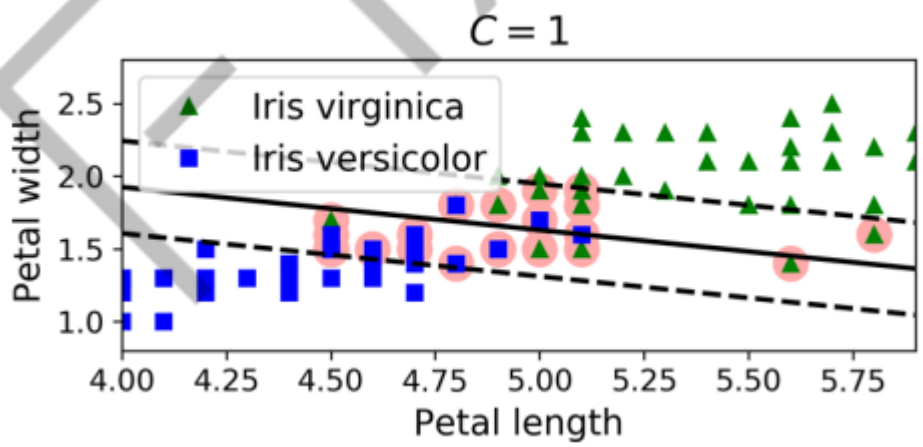


Figura 12 - Valor baixo para o hiperparâmetro C
Fonte: Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow (2023)

Caso seu modelo de SVM fique sobreajustado, você pode tentar regularizá-lo reduzindo o valor de C.

Aplicando o modelo SVM no python:

Para implementar o modelo SVM no python, podemos utilizar o Sklearn! Acesse o site do Scikit learn, onde você encontra mais detalhes sobre os hiperparâmetros do SVM.

```
from sklearn.pipeline import Pipeline
from sklearn.svm import LinearSVC


svm = Pipeline([
    ("linear_svc", LinearSVC(C=1))
])


svm.fit(x_train, y_train)
```

Para solucionar problemas não lineares

A SVM também consegue lidar com dados que não são linearmente separáveis. Mas você deve estar se questionando: o que são dados não linearmente separáveis? Bem, observe a figura 13 – "Tornando um conjunto de dados linearmente separável". Em alguns problemas não é possível de se traçar uma reta sobre os dados para separá-los. Nesse caso temos que utilizar técnicas específicas para lidar com dados que possuem muitas características (como as polinomiais) para observar os dados sobre uma dimensão diferente:

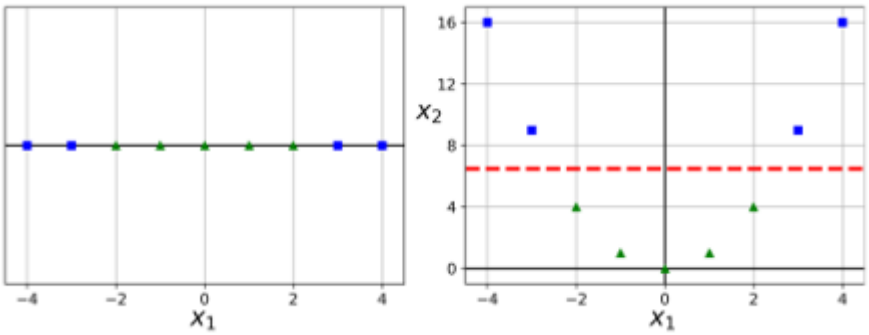


Figura 13 - Tornando um conjunto de dados linearmente separável
Fonte: Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow (2023)

Para solucionar esse caso com SVM, temos várias funções como o Kernel polinomial ou o Kernel RBF Gaussiano. Para mais informações, acesse o site da Scikit learn que contém o conteúdo sobre Kernel.

Para tornar a aplicação do SVM para separar dados não linearmente separáveis, basta configurar a opção "kernel" no pipeline do código durante a construção do SVM:

```
from sklearn.svm import SVC


poly_svm = Pipeline([
    ("svm", SVC(kernel="poly", degree=3, coef0=1, C=5))
])
svm.fit(x_train, y_train)
```

HANDS ON

Você aprendeu alguns poderosos algoritmos e técnicas para lidar com modelos supervisionados. Vamos aprender a como aplicar essas técnicas no Python utilizando um case de negócio?

Para essa aula, temos alguns notebooks disponíveis para você. Acesse aqui:

- Notebook 1
- Notebook 2

Além disso, também disponibilizamos as bases de dados, para te ajudar com os estudos e exercícios.

- Base de dados 1
- Base de dados 2

O QUE VOCÊ VIU NESTA AULA?

Modelo de classificação KNN e SVM.

Daqui em diante, é importante que você replique os conhecimentos adquiridos para fortalecer mais suas bases e conhecimentos.

IMPORTANTE: não esqueça de praticar com o desafio da disciplina, para que assim você possa aprimorar os seus conhecimentos!

Você não está só nesta jornada! Te esperamos no Discord e nas lives com os nossos especialistas, onde você poderá tirar dúvidas, compartilhar conhecimentos e estabelecer conexões!

REFERÊNCIAS

BRUCE, A. **Estatística Prática para Cientistas de Dados**. [s.l.]: O'Reilly Media, Inc., 2019.

DOCUMENTAÇÃO SCIKIT-LEARN. **Disponível em:** <https://scikit-learn.org/stable/>. Acesso em: 10 abr. 2023.

GÉRON, A. **Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow**. 2nd Edition. [s.l.]: O'Reilly Media, Inc., 2019.

PALAVRAS-CHAVE

KNN. SVM. MODELO SUPERVISIONADO DE CLASSIFICAÇÃO.