Support Vector Machine

Diego Addan

Unibrasil 2019

Algoritmos de Aprendizagem de Máquinas (ML)

Árvore de decisão

Floresta aleatória

KNN (K-vizinhos mais próximos)

SVM (Support Vector Machine)

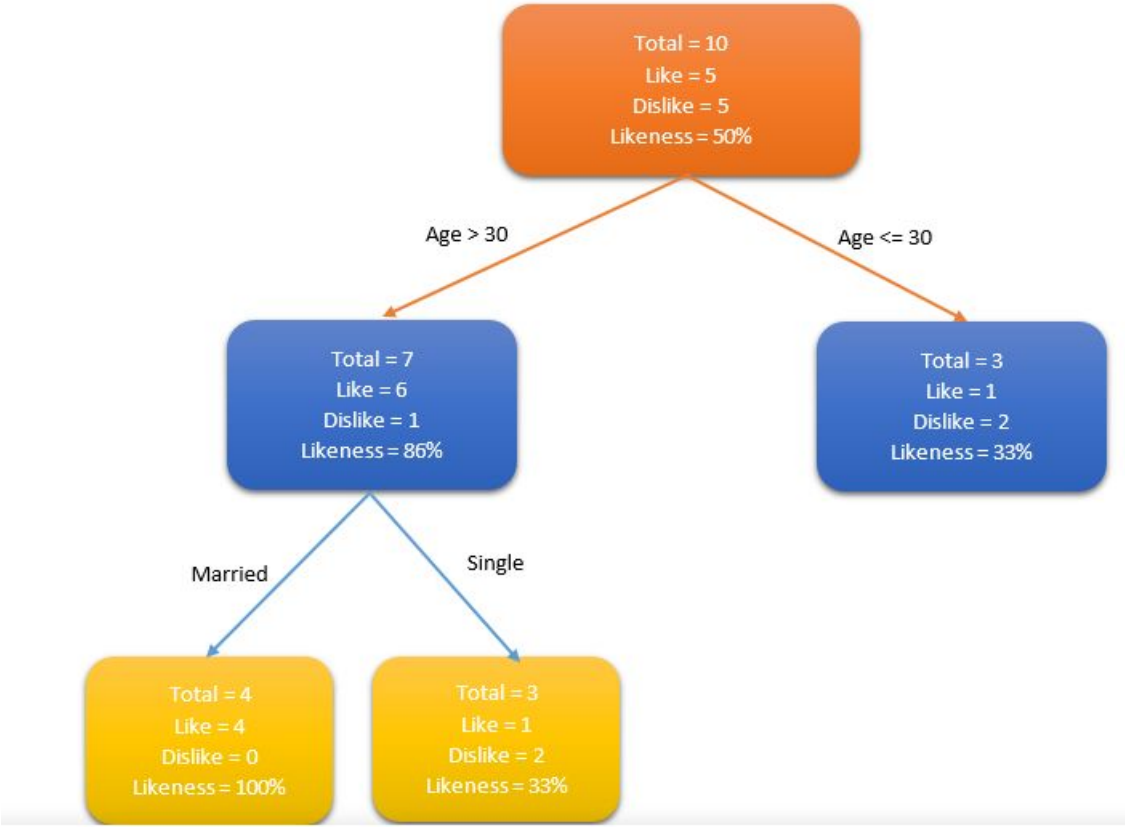
K-Means

Algoritmos de redução dimensional

Algoritmos de aumento de gradiente

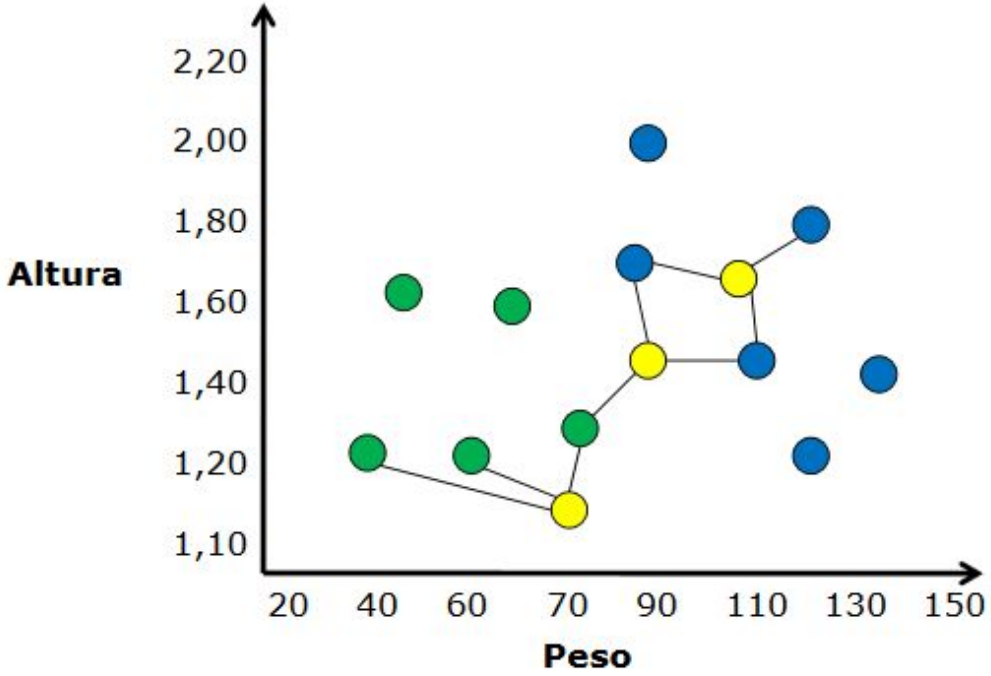
Algoritmos de Aprendizagem de Máquinas (ML)

Árvore de decisão



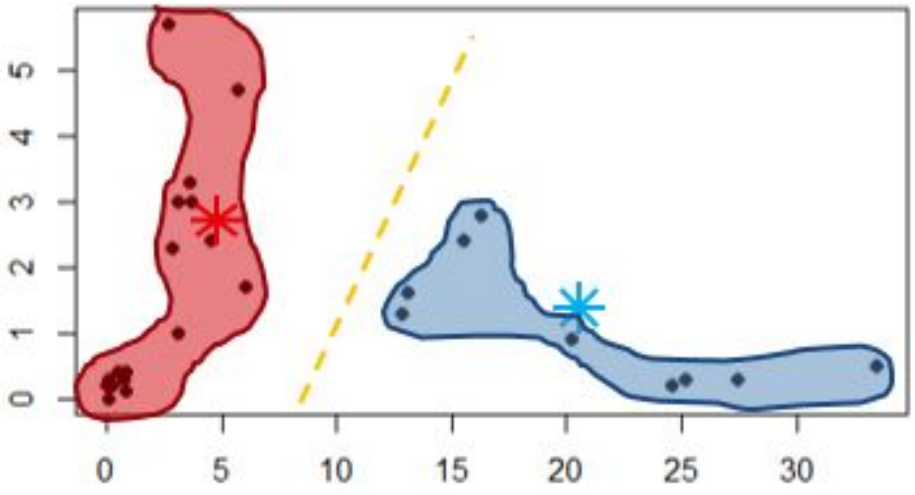
Algoritmos de Aprendizagem de Máquinas (ML)

KNN: K-vizinhos mais próximos



Algoritmos de Aprendizagem de Máquinas (ML)

Algoritmo K-Means



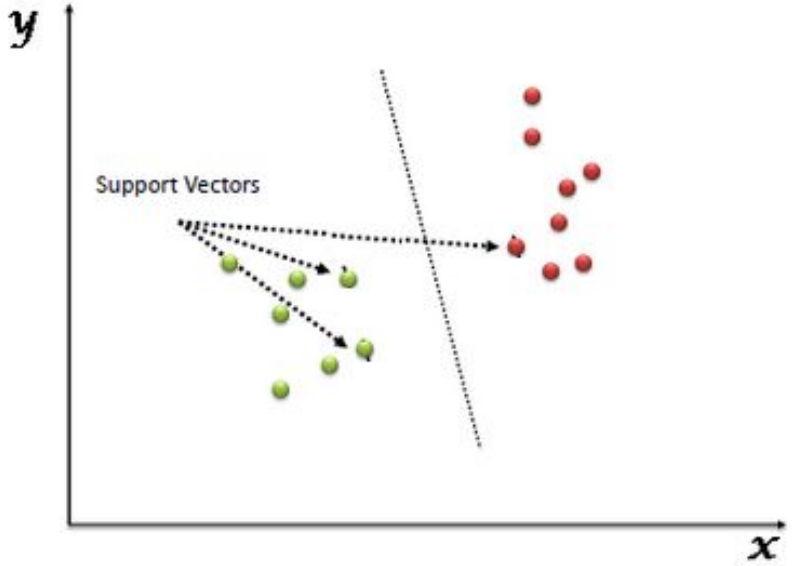
Support Vector Machine

Algoritmos de aprendizado de máquina, supervisionados e de classificação funcionam em conjuntos de dados menores, mas podem ser muito mais fortes e poderosos na criação de modelos.

“Support Vector Machine” (SVM) é um algoritmo de aprendizado de máquina supervisionado que pode ser usado para desafios de classificação ou regressão. Seu foco maior é no treinamento e classificação de um dataset.

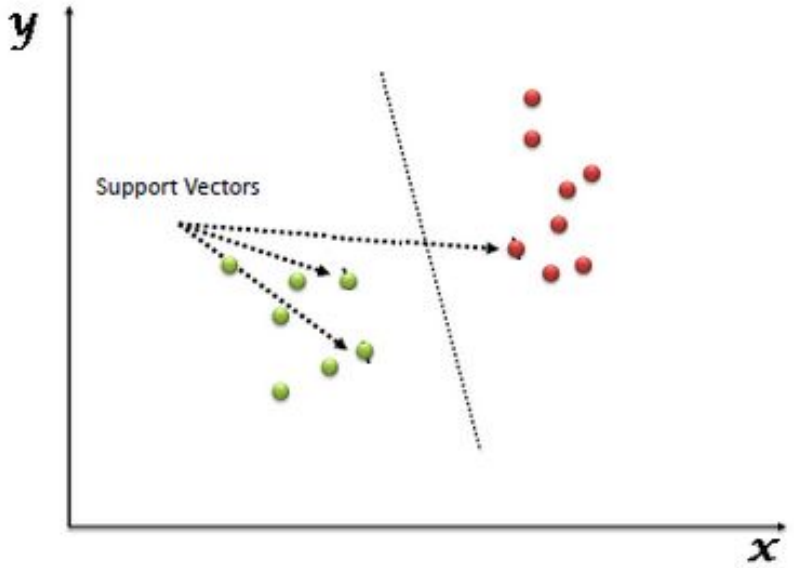
Support Vector Machine

Nesse algoritmo, plotamos cada item de dados como um ponto no espaço n-dimensional (onde n é o número de recursos que você tem), com o valor de cada recurso sendo o valor de uma determinada coordenada. Então, nós executamos a classificação encontrando o hiperplano que melhor diferencia as duas classes.



Support Vector Machine

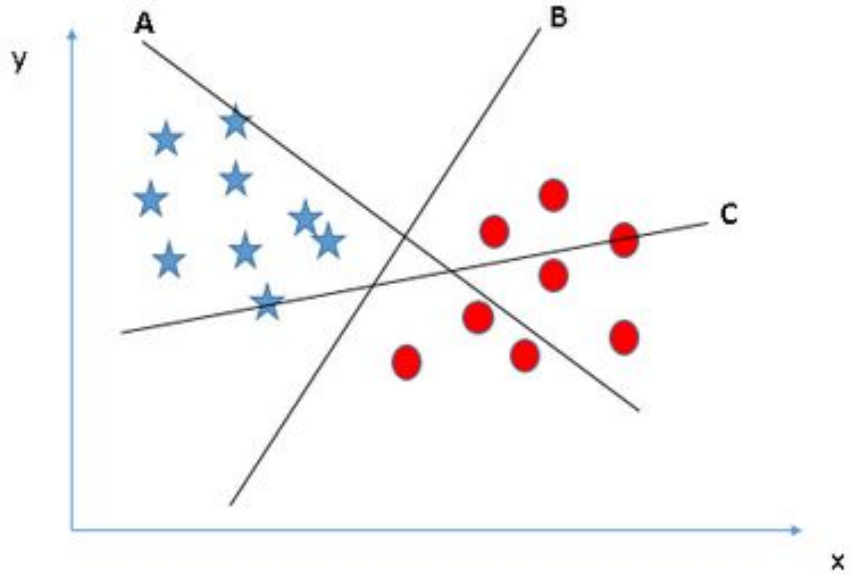
Os “Vetores de suporte” são simplesmente as coordenadas da observação individual. Support Vector Machine é uma fronteira que melhor segrega as duas classes (hiperplano / linha).



Support Vector Machine

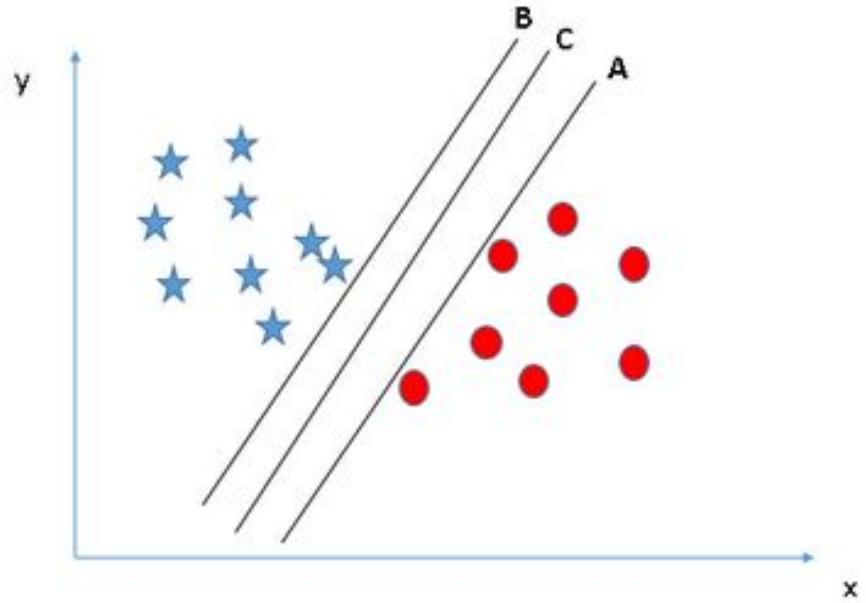
Em SVM a questão mais importante é “como podemos identificar o hiperplano?”.

Desenvolvendo a hipótese: Aqui, temos três hiperplanos (A, B e C). Agora, identifique o hiperplano certo para classificar estrela e círculo.



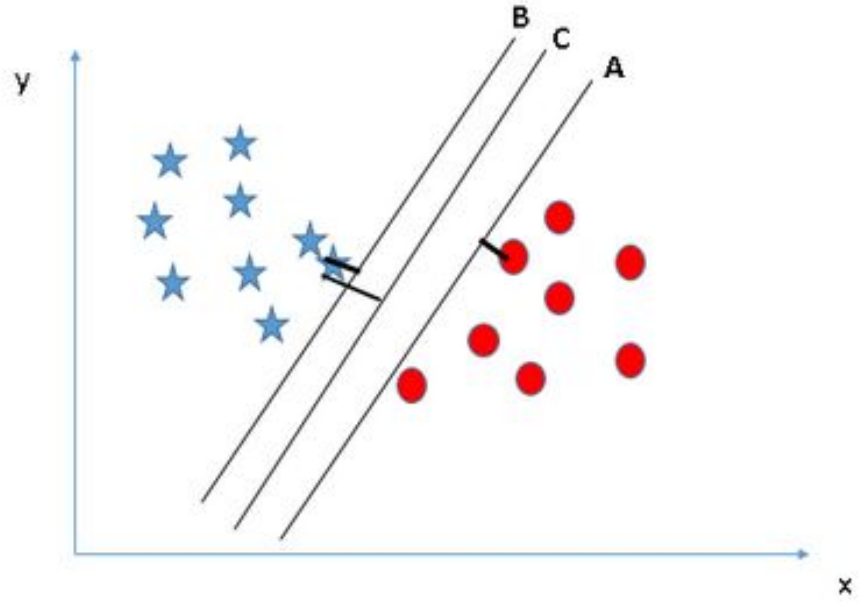
Support Vector Machine

Aqui, temos três hiperplanos (A, B e C) e todos estão segregando bem as classes. Agora, como podemos identificar o hiperplano certo?



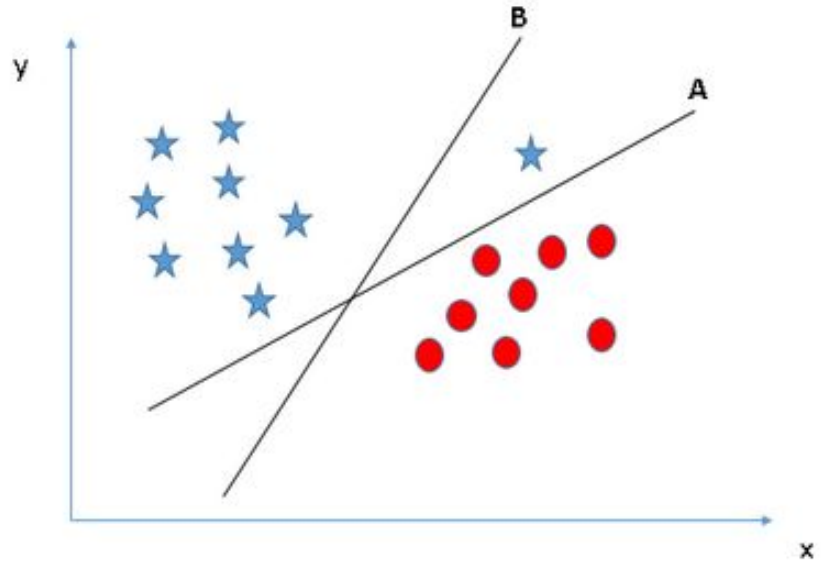
Support Vector Machine

Aqui, maximizar as distâncias entre o ponto de dados mais próximo (de qualquer classe) e o hiperplano nos ajudará a decidir o hiperplano correto. Essa distância é chamada de Margem.



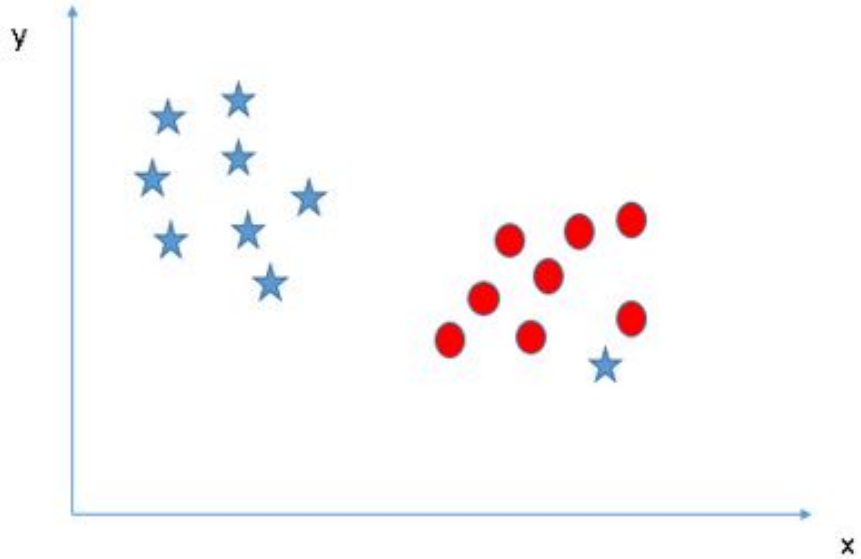
Support Vector Machine

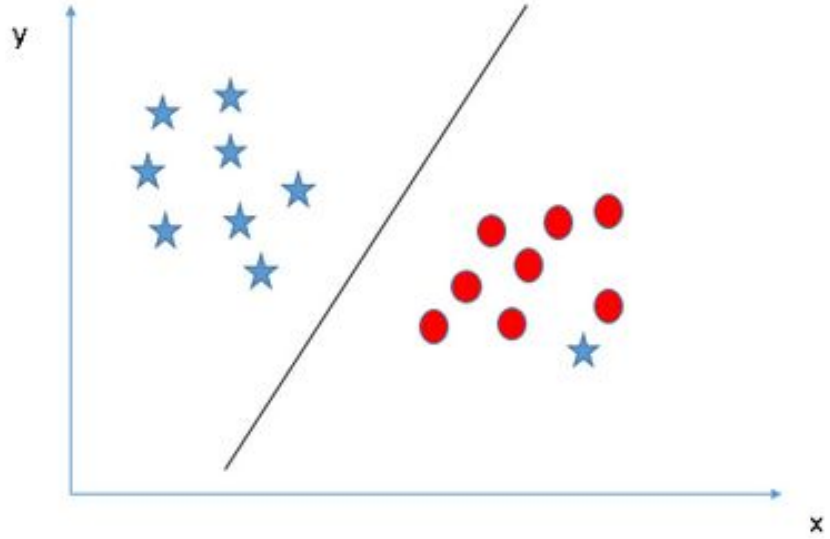
Neste caso, o melhor hiperplano é o B (já que ele tem uma margem maior em comparação a A)? O SVM seleciona o hiperplano que classifica as classes com precisão antes de maximizar a margem. Aqui, o hiperplano B tem um erro de classificação e A classificou tudo corretamente. Portanto, o melhor hiperplano é A.



Support Vector Machine

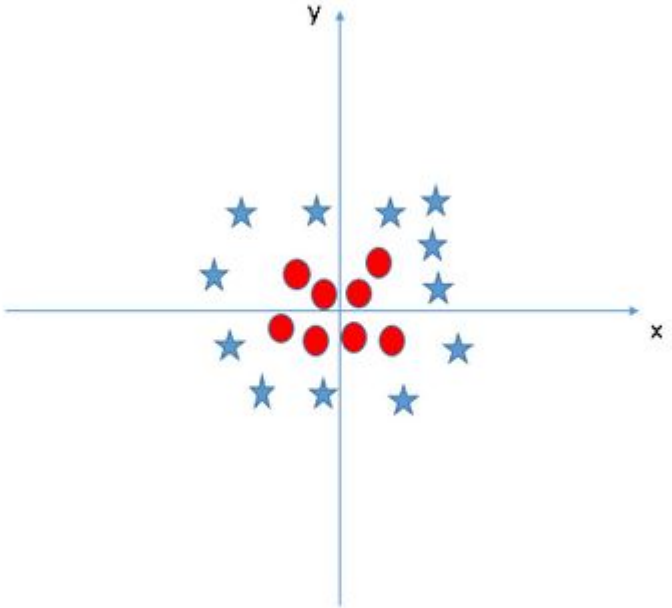
Existem caso onde não é possivel separar as duas classes usando uma linha reta, pois uma das classes está no território de outra (**outlier**). O SVM tem um recurso para ignorar valores discrepantes e encontrar o hiperplano que tem margem máxima. Portanto, podemos dizer que SVM é robusto para outliers.





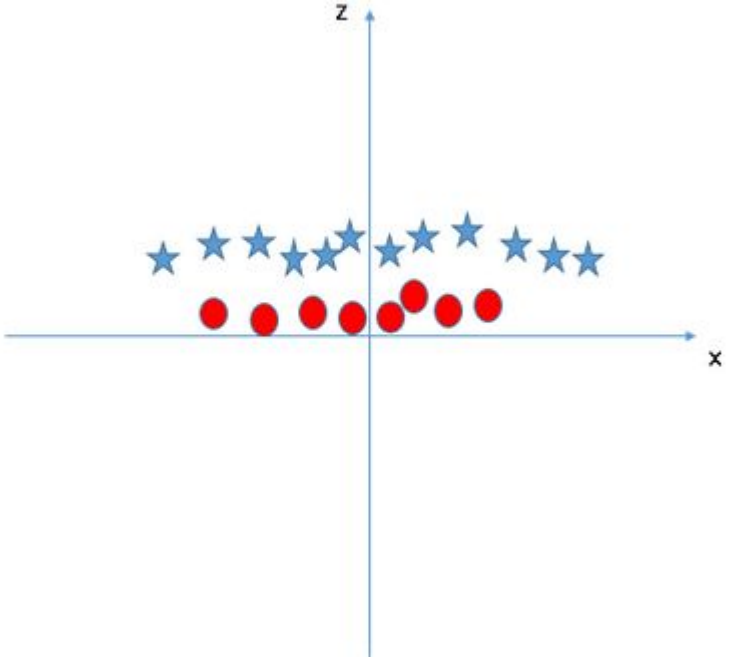
Support Vector Machine

No cenário abaixo, não podemos ter um hiperplano linear entre as duas classes, então, como o SVM classifica essas duas classes? Até agora, só olhamos para o hiperplano linear.



Support Vector Machine

O SVM resolve esse problema introduzindo um recurso adicional. Aqui, vamos adicionar um novo recurso (regra para o indutor) z = x ^ 2 + y ^ 2. Agora, vamos plotar os pontos de dados nos eixos x e z:



Support Vector Machine

Na plotagem anterior, os pontos a serem considerados são:

- Todos os valores para z seriam positivos sempre porque z é a soma quadrática de x e y
- Na imagem original, os círculos vermelhos aparecem próximos da origem dos eixos x e y, levando a um valor menor de z e a classe estrela relativamente longe do resultado da origem para um valor maior de z.

No SVM, é fácil ter um hiperplano linear entre essas duas classes. Mas, outra pergunta que surge é, devemos adicionar esse recurso manualmente para ter um hiperplano. Não, o SVM tem uma técnica chamada **truque do kernel**.

Support Vector Machine

Estas são funções que ocupam um espaço de entrada dimensional baixo e o transformam em um espaço dimensional superior, isto é, ele converte um problema não separável em um problema separável.

Essas funções são chamadas de **núcleos**. É principalmente útil no problema de separação não linear. Simplificando, ele faz algumas transformações de dados extremamente complexas e, em seguida, descobre o processo para separar os dados com base nos rótulos ou nas saídas que você definiu.

O SVM não fornece estimativas de probabilidade diretamente, elas são calculadas usando uma validação cruzada onerosa de cinco etapas (método SVC relacionado da biblioteca scikit-learn do Python).

Support Vector Machine

- **Prós:**
  - Funciona muito bem com margem de separação clara.
  - É eficaz nos casos em que o número de dimensões é maior que o número de amostras.
  - Ele usa um subconjunto de pontos de treinamento na função de decisão (chamados de vetores de suporte), portanto, também é eficiente em termos de memória.

  Contras:

  - Não tem um bom desempenho quando temos um grande conjunto de dados porque o tempo de treinamento necessário é grande.
  - Ele também não funciona muito bem quando o conjunto de dados tem mais ruído, ou seja, as classes de destino estão sobrepostas.

SVM - Exemplo

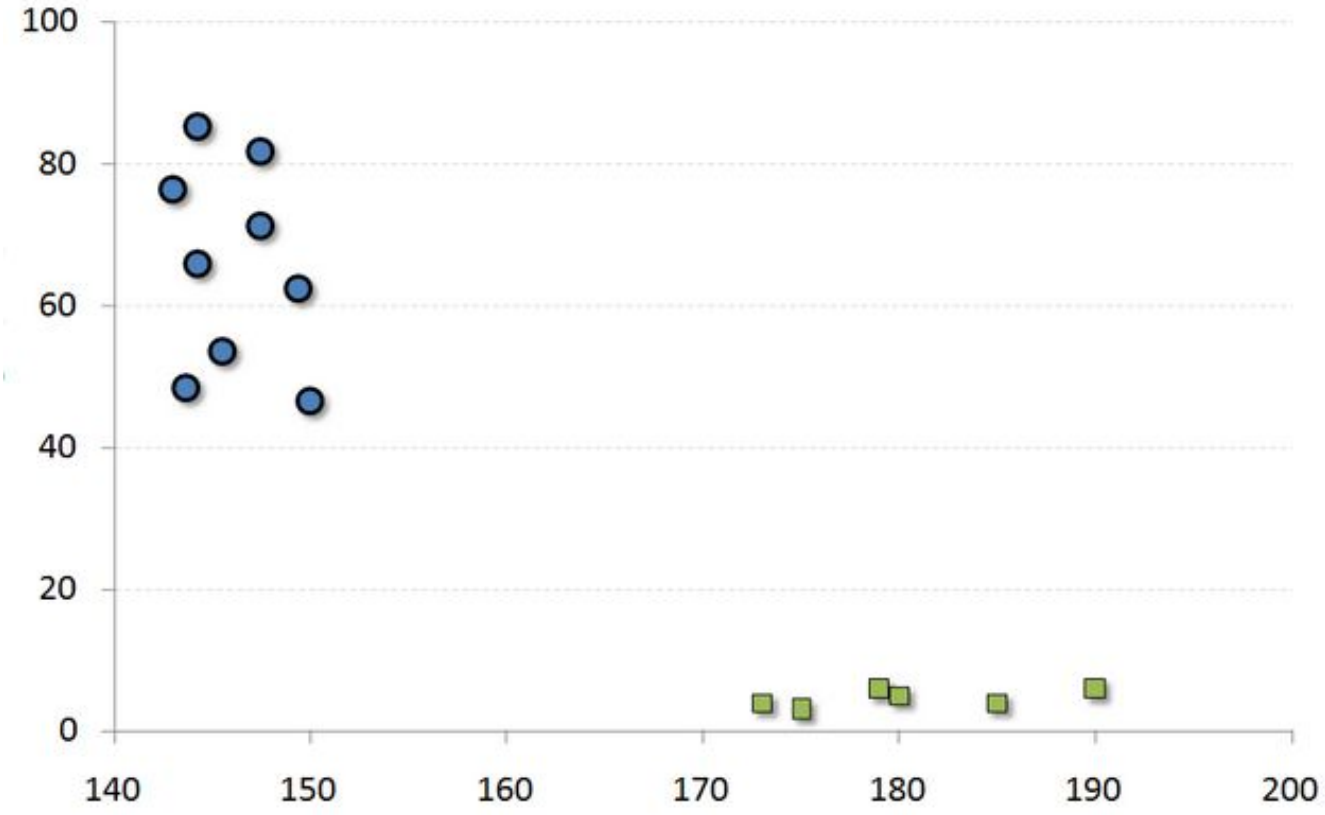
Considere o exemplo: Temos uma população composta por **50%-50% machos e fêmeas**.

Usando uma amostra dessa população, você quer criar um conjunto de regras que defina a classe de gênero para o resto da população.

Usando este algoritmo, pretendemos construir um método que possa identificar se um exemplo é um macho ou uma fêmea.

Este é um problema de **amostra da análise de classificação**. Usando algum conjunto de **regras**, tentaremos **classificar** a população em dois segmentos possíveis. Para simplificar, suponha que os dois fatores de diferenciação identificados sejam: comprimento do indivíduo e Comprimento da garra.

SVM - Exemplo



SVM - Exemplo

Os círculos azuis na imagem representam as fêmeas e os quadrados verdes representam os machos. Alguns insights esperados do gráfico são:

1. Os machos da nossa população têm um comprimento médio maior.

2. As fêmeas da nossa população têm garras mais longas.

Se fôssemos ver um indivíduo com 180cm de comprimento e comprimento da garra de 4cm, nosso melhor palpite seria classificar esse indivíduo como um macho. É assim que fazemos uma análise de classificação.

SVM - Exemplo

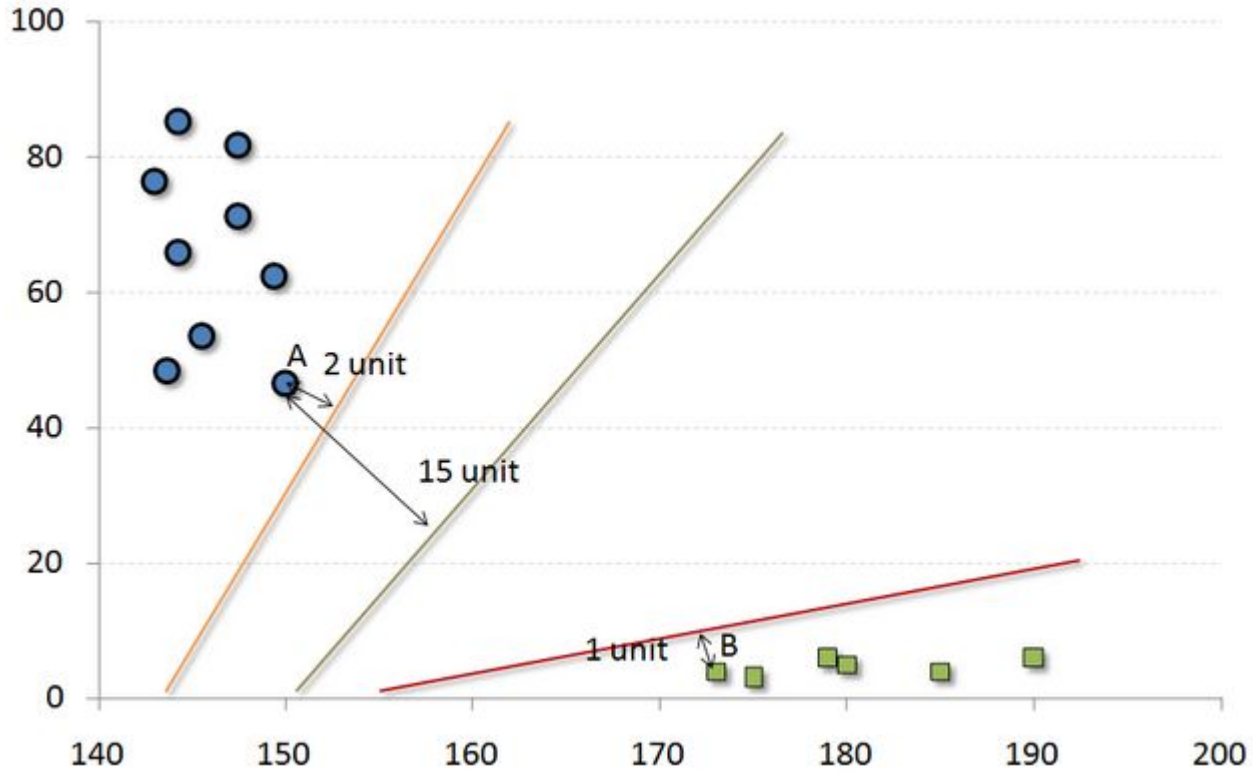
Vetores de Suporte são simplesmente as coordenadas da observação individual.

Por exemplo, (45, 150) é um vetor de suporte que corresponde a uma fêmea. Support Vector Machine é uma fronteira que melhor segrega as classes.

Neste caso, as duas classes estão bem separadas umas das outras, por isso é mais fácil encontrar um SVM.

Existem muitas fronteiras possíveis que podem classificar o problema em questão. A seguir estão três fronteiras possíveis.

SVM - Exemplo


100
80
60
40
20
0
140
150
160
170
180
190
200
A
2 unit
15 unit
1 unit
B

SVM - Exemplo

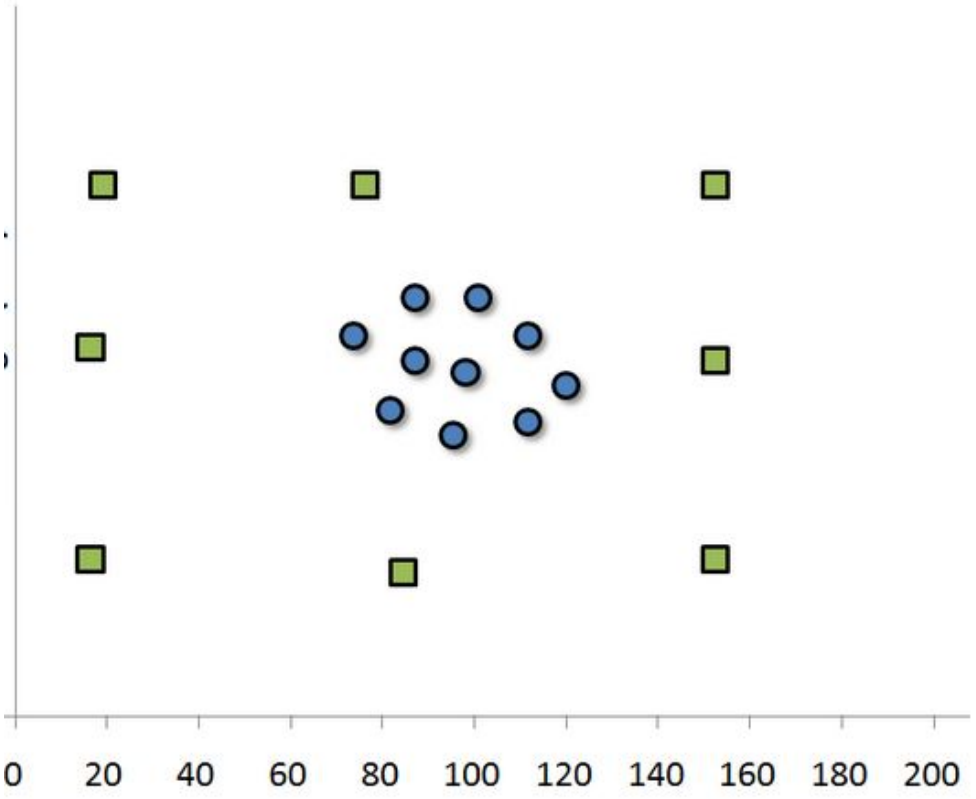
Como decidimos qual é a melhor fronteira?

A maneira mais fácil de interpretar a função objetivo em um SVM é encontrar a **distância mínima** da fronteira do vetor de suporte mais próximo (isso pode pertencer a qualquer classe).

Por exemplo, a fronteira laranja está mais próxima dos círculos azuis. E o círculo azul mais próximo está a 2 unidades da fronteira. Uma vez que tenhamos essas distâncias para todas as fronteiras, simplesmente escolhemos a fronteira com a distância máxima (do vetor de suporte mais próximo). Das três fronteiras mostradas, vemos que a fronteira negra está mais distante do vetor de suporte mais próximo (ou seja, 15 unidades).

SVM - Exemplo

E se não encontrarmos uma fronteira limpa que segregue as classes?

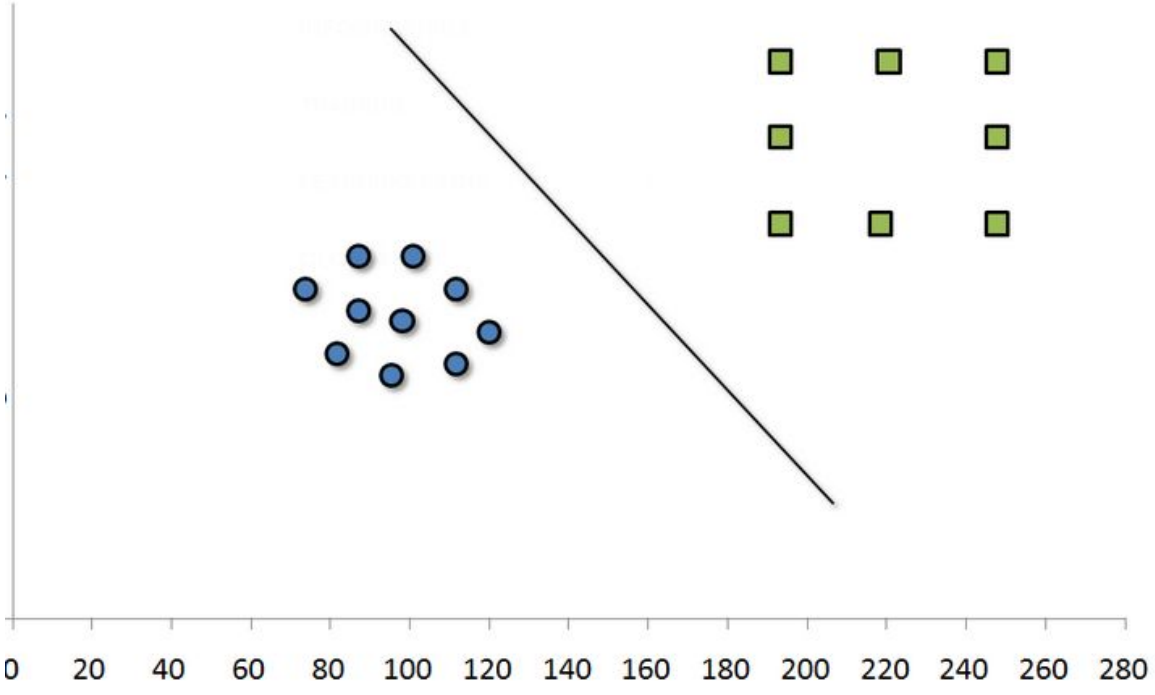


SVM - Exemplo

Nesses casos, não vemos uma fronteira de linha reta diretamente no plano atual, que pode servir como SVM.

Nesses casos, precisamos mapear esses vetores para um plano de dimensão mais alto, de modo que eles sejam segregados uns dos outros.

Cada quadrado verde na distribuição original é mapeado em uma escala transformada.

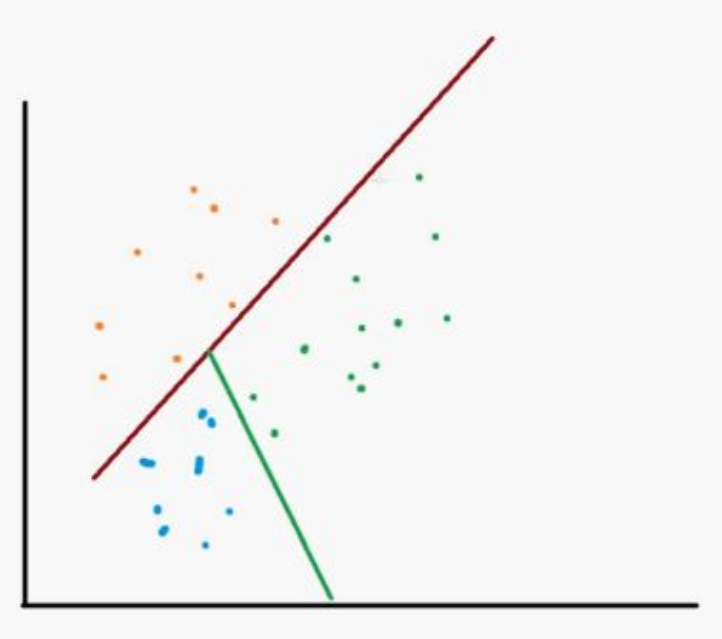


SVM - Implementação

Assim como as aulas anteriores, a biblioteca para python sklearn possui datasets compatíveis com experimentos de indutores para classificação e rotulação.

```
import matplotlib.pyplot as plt

from sklearn import datasets
from sklearn import svm
```

KNN, K-Means e SVM em python:



SVM - Exemplo

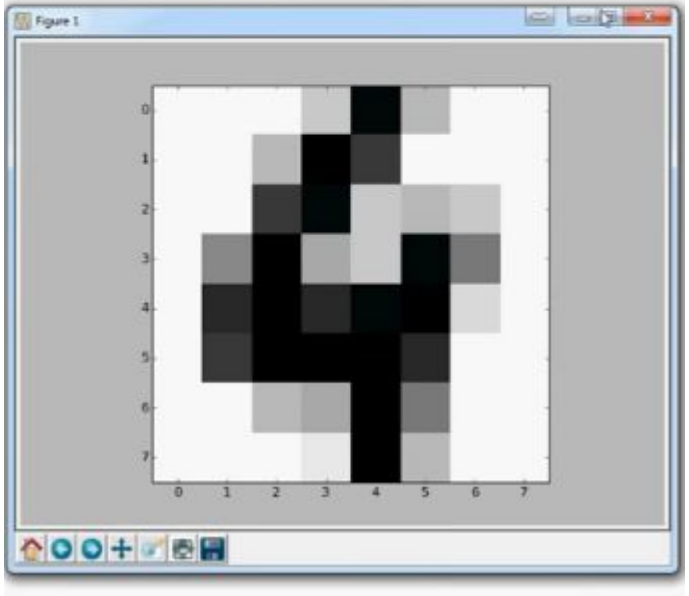
Definir bem o problema e tratar os dados de treinamento.

Os dados devem ser numéricos (é a única maneira de relacionar e normalizar os indutores)

```
digits = datasets.load_digits()

clf = svm.SVC(gamma=0.001, C=100)

print(len(digits.data))

x,y = digits.data[:-1], digits.target[:-1]
clf.fit(x,y)

print('Prediction:',clf.predict(digits.data[-1]))

plt.imshow(digits.images[-1], cmap=plt.cm.gray_r, interpolation="nearest")
plt.show()
```

SVM - Exemplo

```
Python 3.4.1 (v3.4.1:c0e311e010fc, May 1
D64)] on win32
Type "copyright", "credits" or "license(
>>> ================================ RES
>>>
1797
Prediction: [4]
```

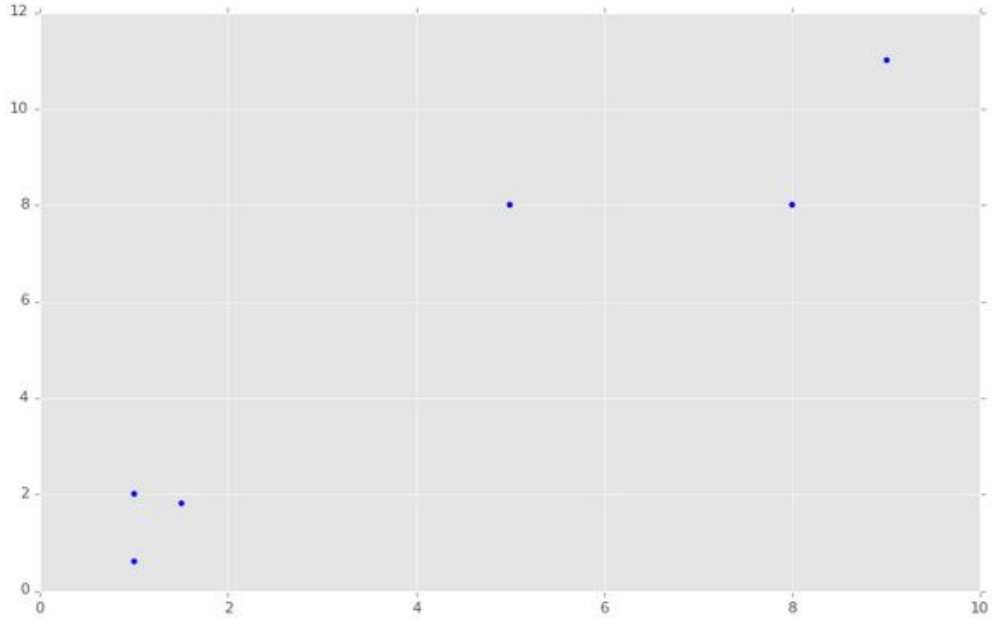


SVM - Exemplo

Outro exemplo (ainda em python):

```
import numpy as np
import matplotlib.pyplot as plt
from matplotlib import style
from sklearn import svm
```

```
x = [1, 5, 1.5, 8, 1, 9]
y = [2, 8, 1.8, 8, 0.6, 11]
plt.scatter(x,y)
plt.show()
```

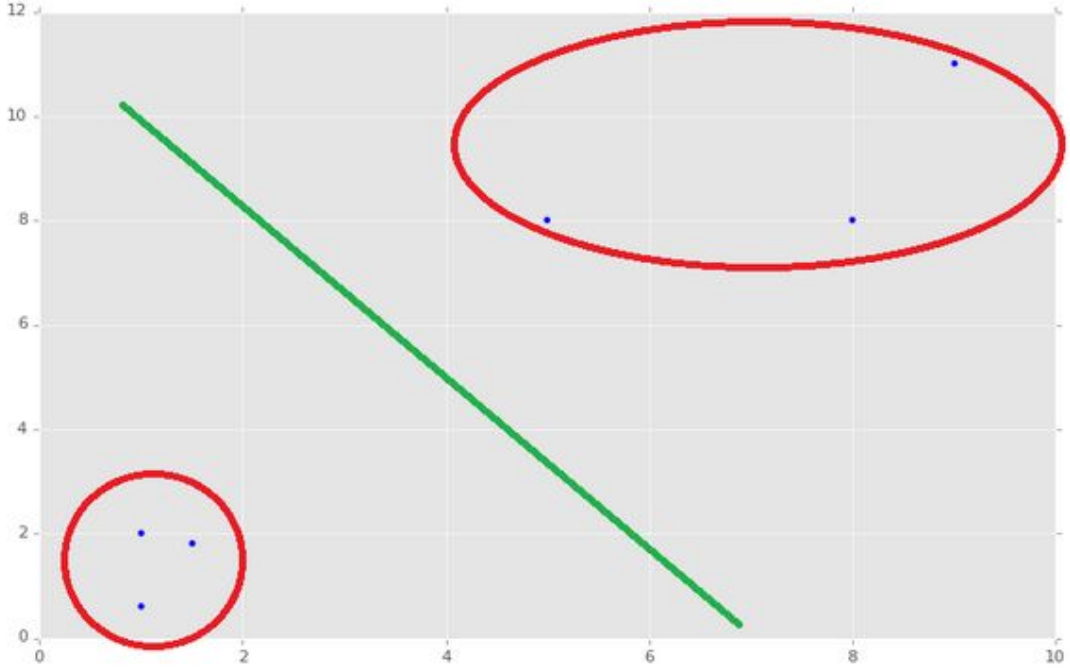


SVM - Exemplo

Outro exemplo (ainda em python):

```
import numpy as np
import matplotlib.pyplot as plt
from matplotlib import style
from sklearn import svm
```

```
x = [1, 5, 1.5, 8, 1, 9]
y = [2, 8, 1.8, 8, 0.6, 11]
plt.scatter(x,y)
plt.show()
```



SVM - Exemplo

Outro exemplo (ainda em python):

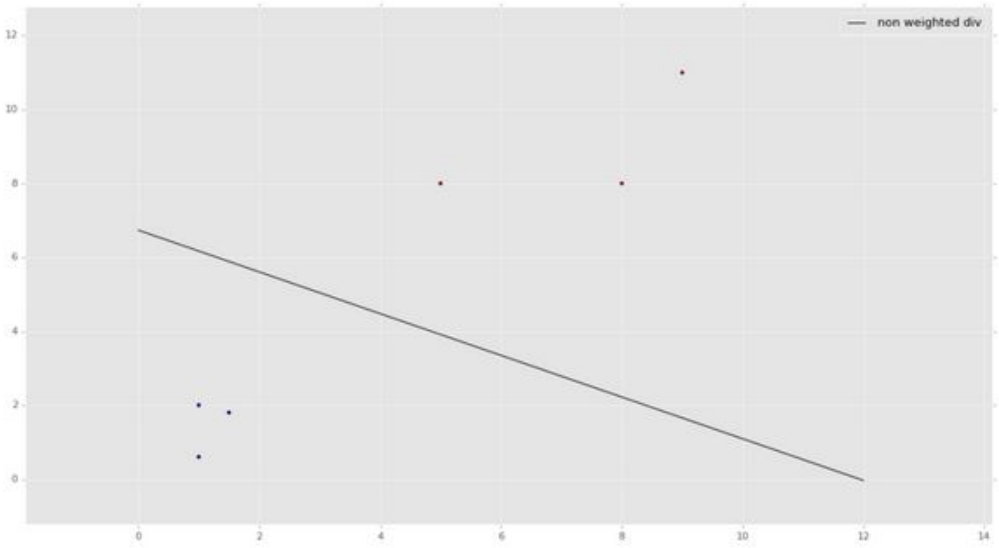
```
#data x e y
X = np.array([[1,2],[5,8],[1.5,1.8],[8,8],[1,0.6],[9,11]])
y = [0,1,0,1,0,1] #meus labels, usaremos 0 e 1

clf = svm.SVC(kernel='linear', C = 1.0)

clf.fit(X,y)

print(clf.predict([0.58,0.76]))

>>> 0

print(clf.predict([10.58,10.76]))

>>>1
```

SVM - Exemplo

Outro exemplo (ainda em python):

Para visualizar os dados:

```
w = clf.coef_[0]
print(w)
a = -w[0] / w[1]
xx = np.linspace(0,12)
yy = a * xx - clf.intercept_[0] / w[1]
h0 = plt.plot(xx, yy, 'k-',
              label="non weighted div")
plt.scatter(X[:, 0], X[:, 1], c = y)
plt.legend()
plt.show()
```



Trabalho de implementação

Crie um dataset com exemplos e atributos numéricos.

Com base no dataset, desenvolva um classificador e uma ou mais hipóteses de classificação utilizando o algoritmo SVM .

Apresente uma base de teste e aplique seu sistema a fim de classificar os novos exemplos.

Regras:

- Pode ser implementado em até 3 pessoas.
- Deve ser entregue um TR (relatório técnico) e a implementação simples.
- Não utilize métodos prontos.
- Entrega: **06/06.**