SVM DE CLASSIFICAÇÃO

O que é SVM de Classificação?

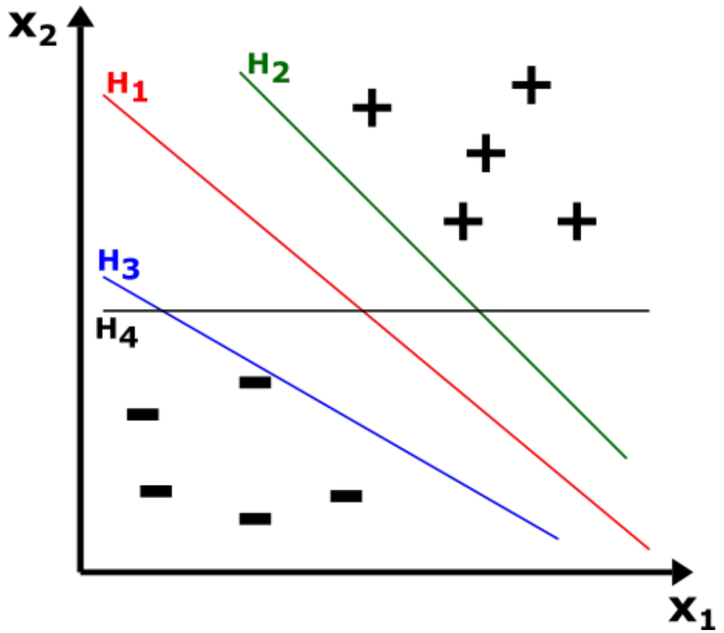
- As Máquinas de Vetores de Suporte (SVM) é um algoritmo de aprendizagem de máquina utilizado tanto para classificação quanto para regressão.

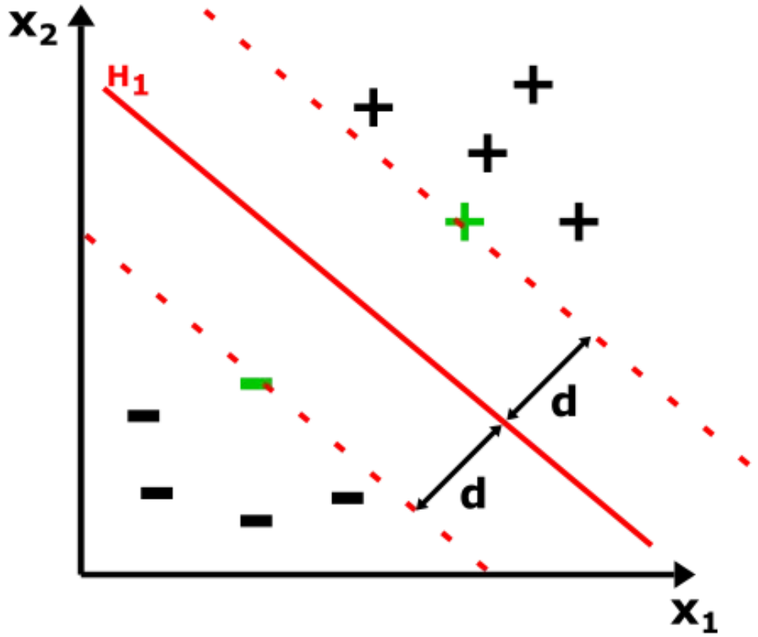
- Elas funcionam identificando um hiperplano que separa de maneira ótima as diferentes classes de dados.

- O SVM busca encontrar o melhor hiperplano para um dado cujas classes são linearmente separáveis.

- O ponto mais próximo de cada classe está a uma distância **d** do hiperplano, eles são chamados de **vetores de suporte**

- O melhor hiperplano capaz de separar as duas classes estaria localizado no **ponto médio** entre elas (maximizando a margem).

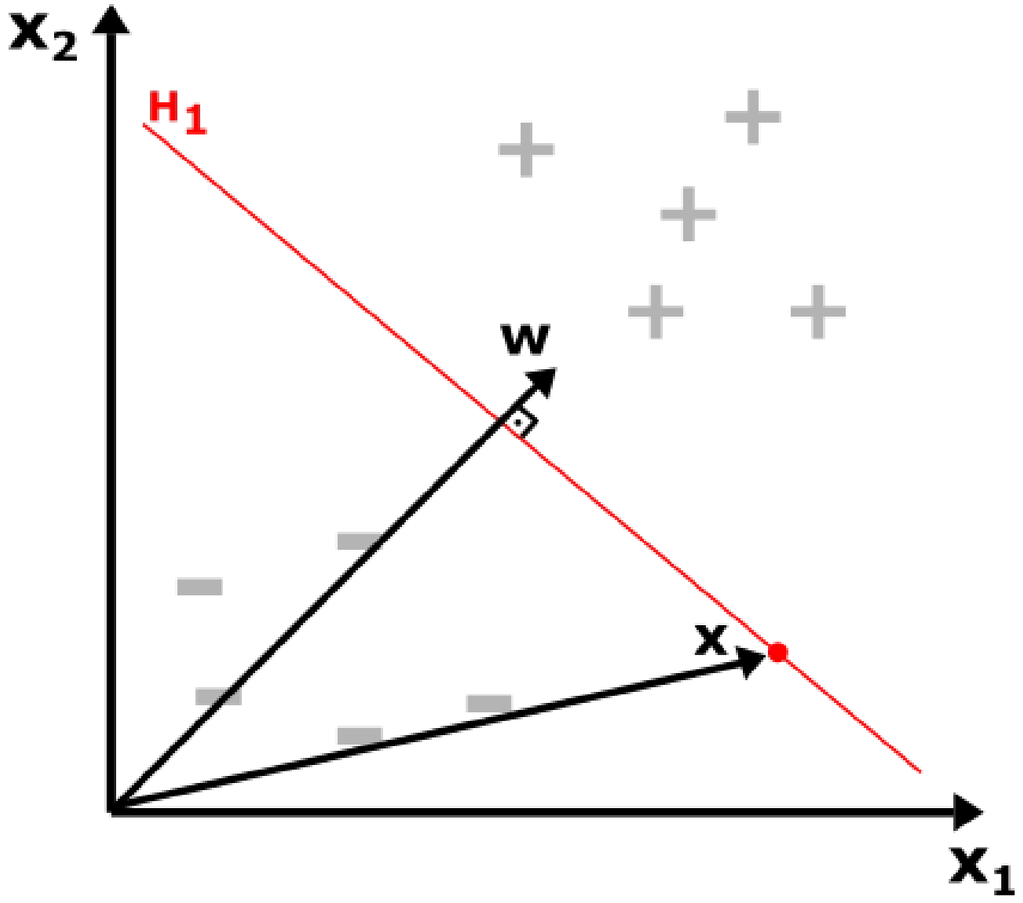




Funcionamento e Otimização

- **Calcular o Hiperplano:**

$$\overline{w}^T \cdot \overline{x} + b = 0$$

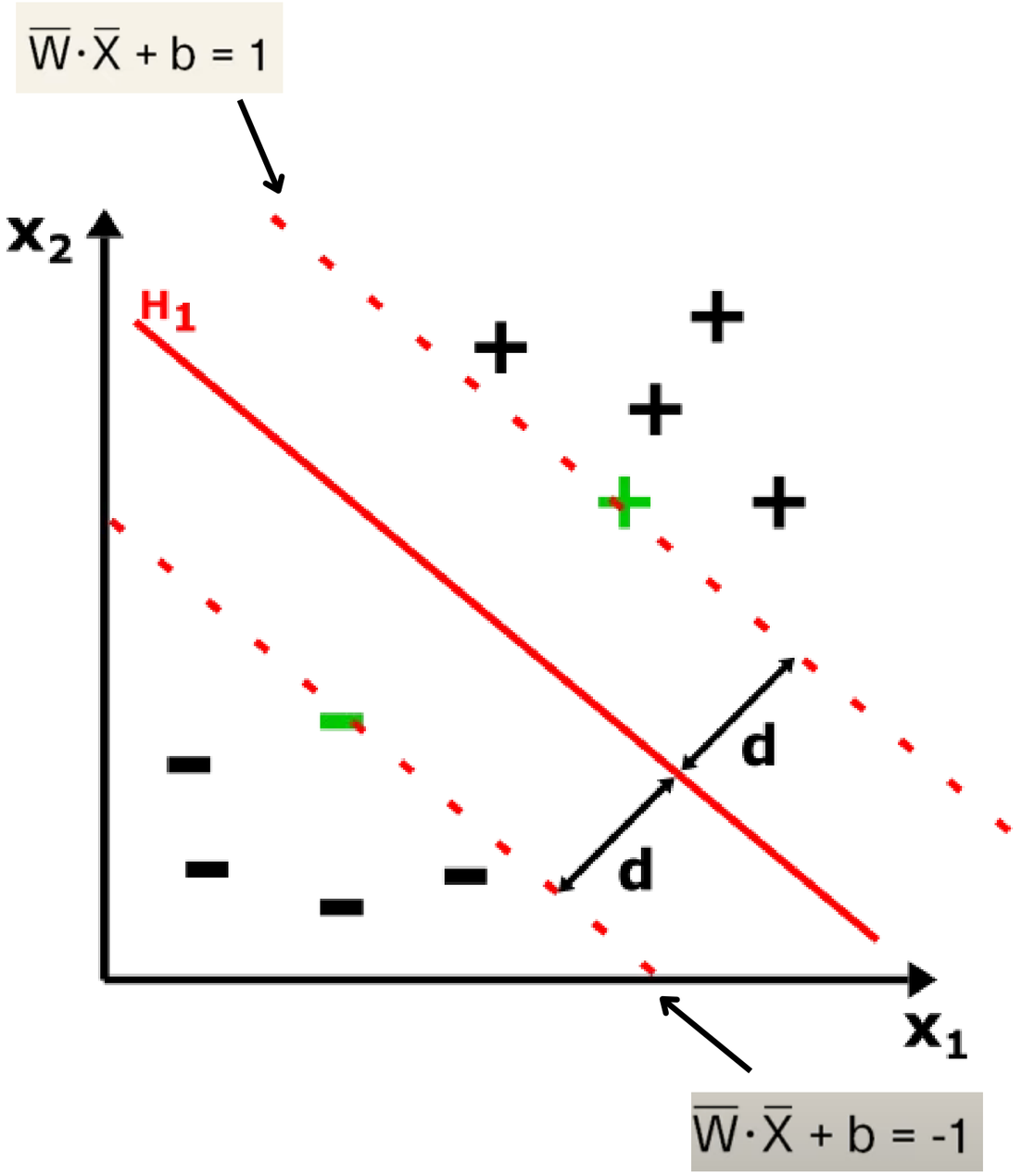


- Onde **w** é o vetor que caracteriza o hiperplano (coeficientes)
- **x** é um ponto qualquer pertencente ao hiperplano e portanto satisfaz essa equação (variáveis)
- **b** é uma constante que determina o deslocamento do hiperplano em relação a origem

- **Definir a margem / Vetores Suportes**

$$Y \begin{cases} +1 \text{ if } w^T x + b \geq 0 \\ -1 \text{ if } w^T x + b < 0 \end{cases}$$

- Se a gente for prever uma nova observação:
  - Se ela pertencer ao grupo positivo
    - Acima do hiperplano
  - Se ela pertencer ao grupo negativo
    - Abaixo do hiperplano



- **Função que maximiza a margem:**

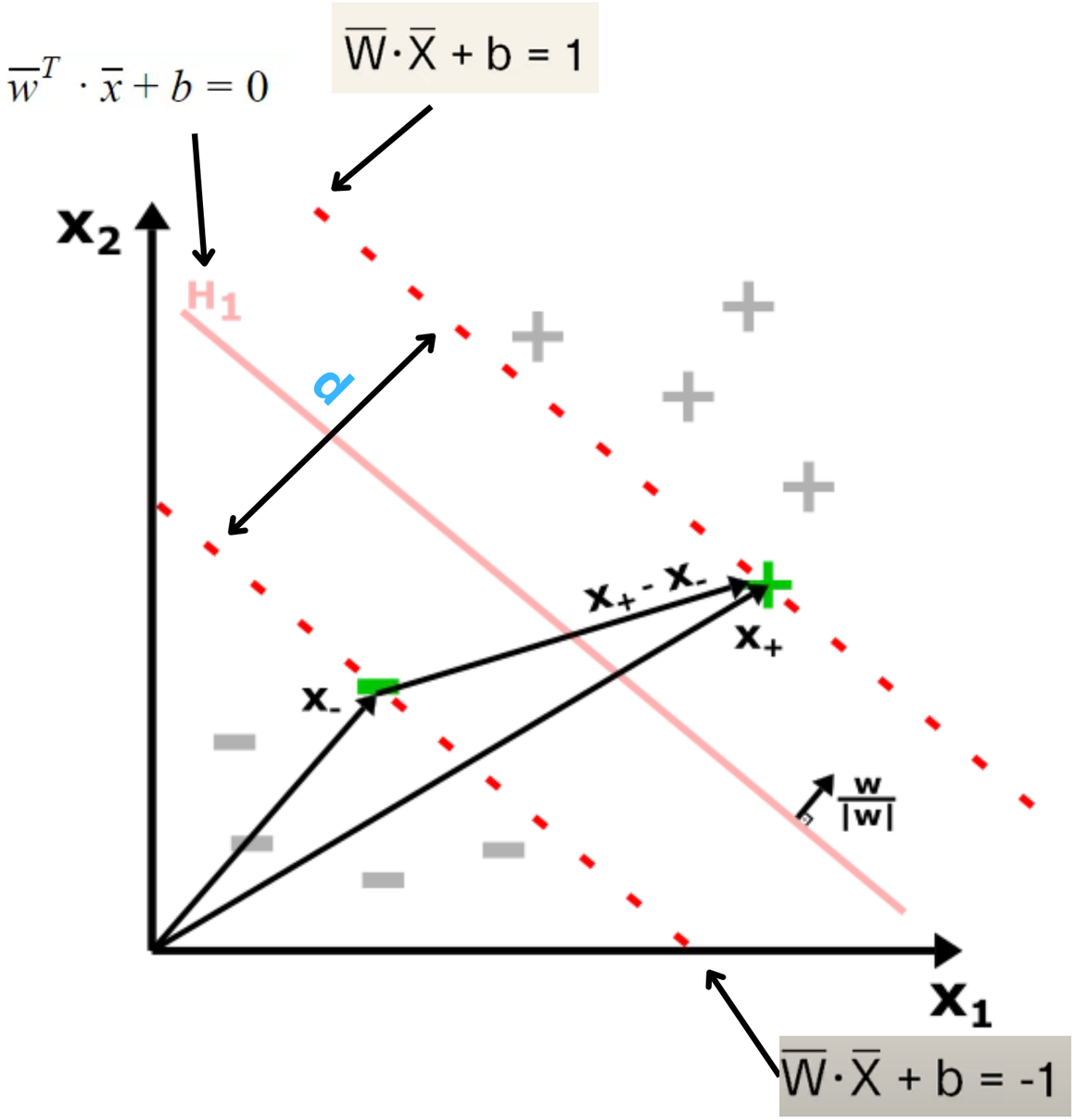
Calcular a distância entre duas linhas paralelas:

- A distância entre o hiperplano e os vetores de suporte é a mesma

$$d = \frac{|C_2 - C_1|}{\sqrt{A^2 + B^2}}$$

$$\frac{1 - b - (-1 - b)}{\|\overline{W}\|} = \max \frac{2}{\|w\|}$$

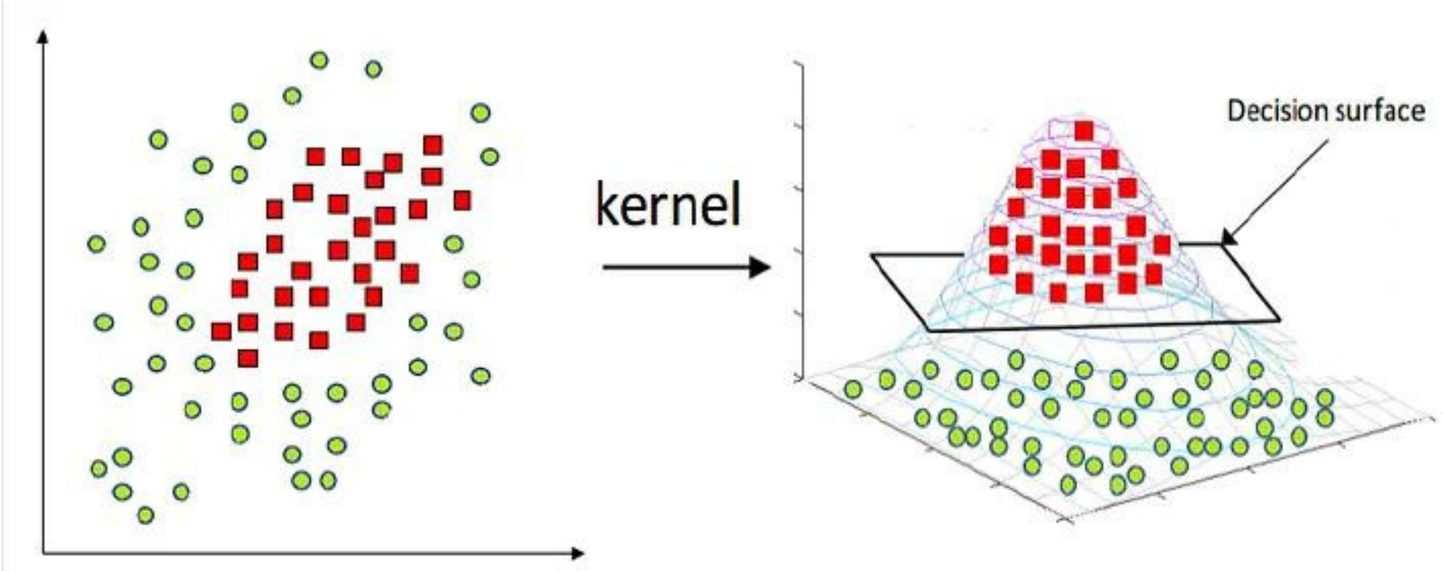
$$\min \frac{1}{2} |w|^2$$

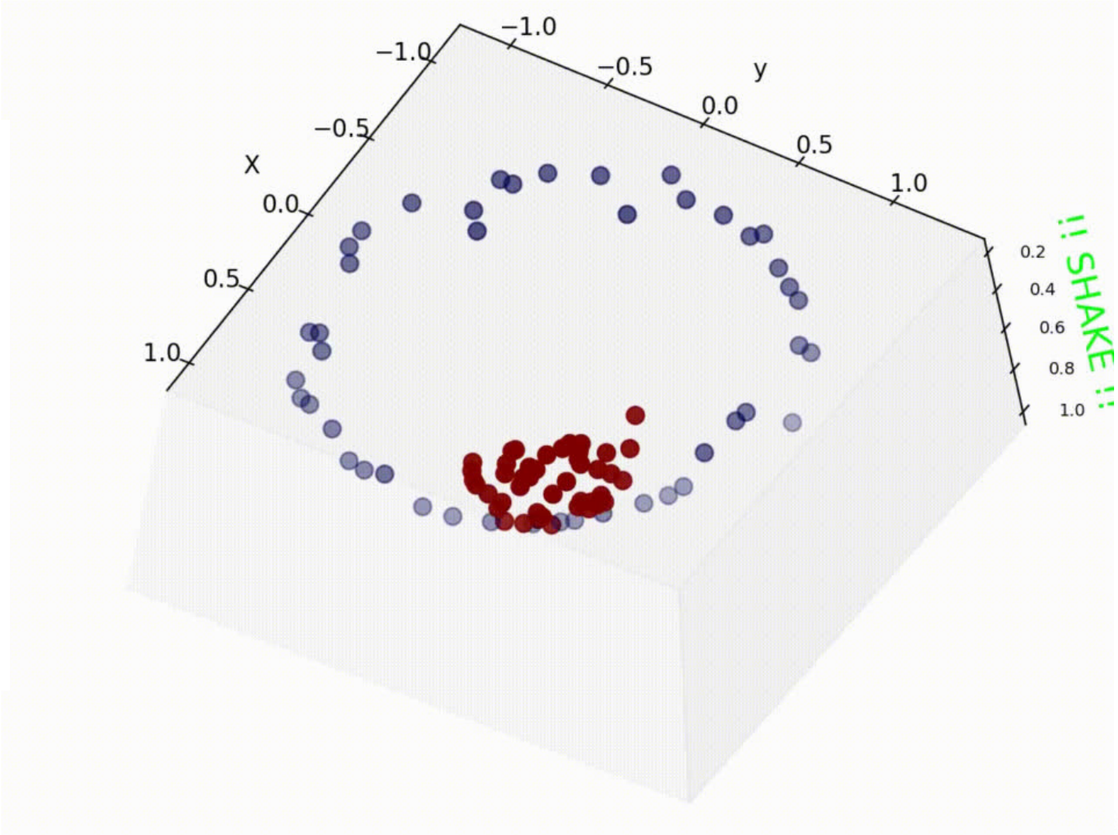


Kernel Trick

- Na maioria dos casos, não podemos separar os dados em linha reta que divida perfeitamente dos dados em classe.

- Com isso, o Kernel Trick projeta os dados em um espaço dimensional superior, onde podemos traçar o hiperplano e separar as categorias, tendo uma visão tridimensional dos dados.

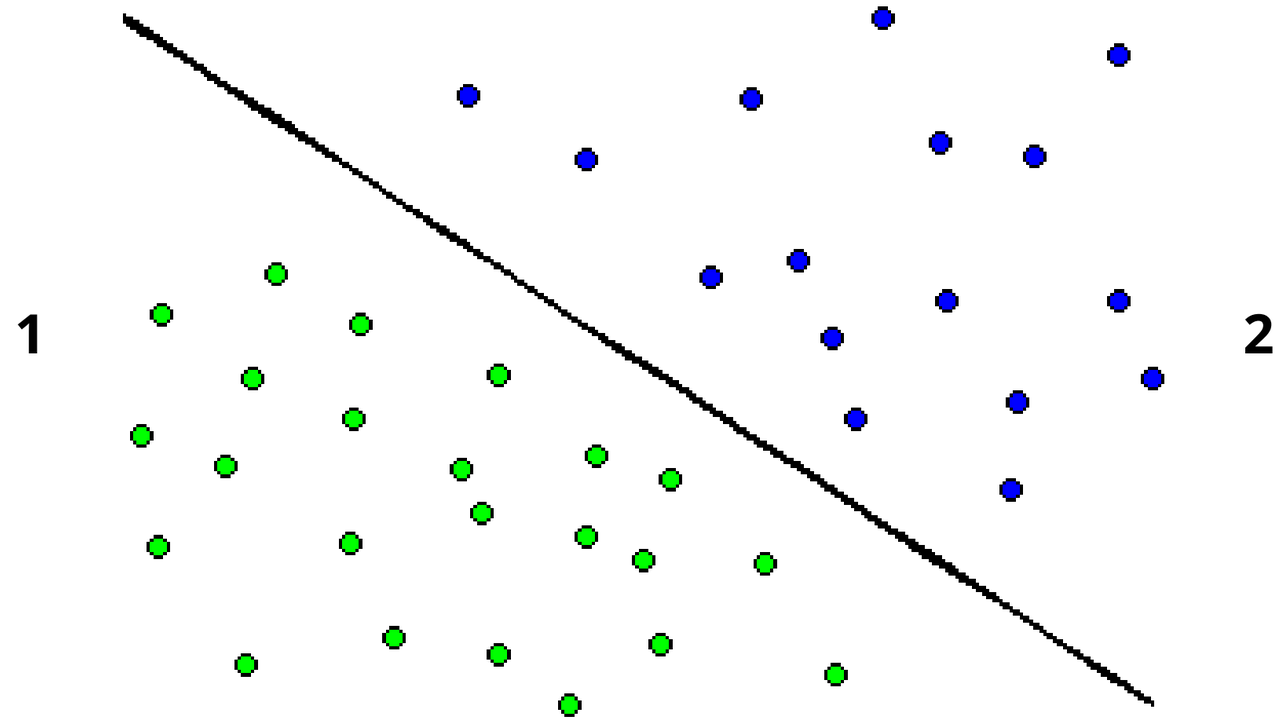


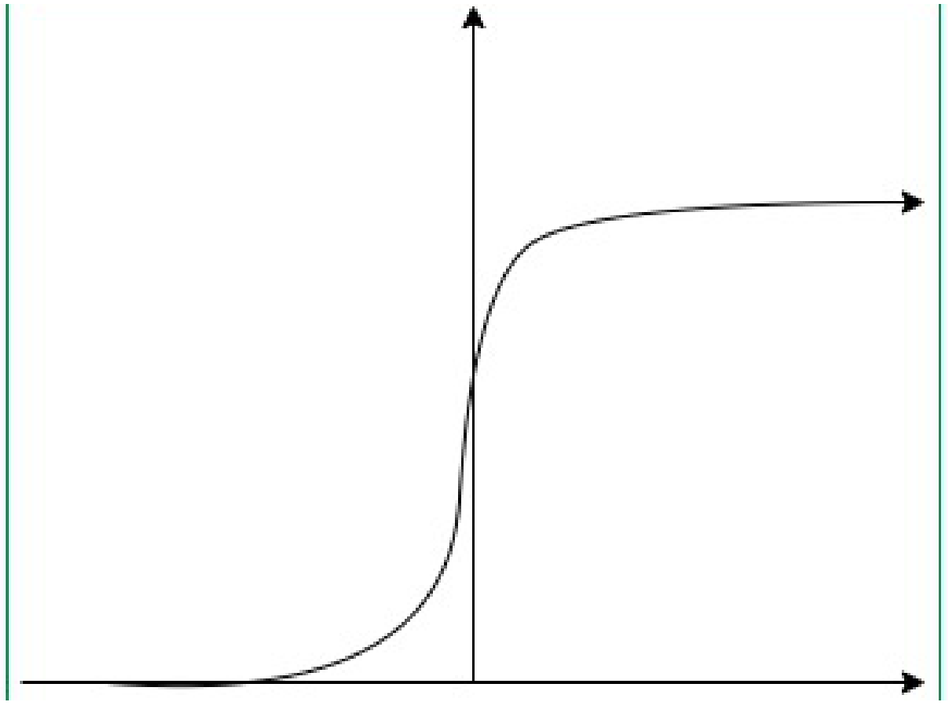


Tipos de Kernel

1.**Kernel Linear**: Usado quando os dados são linearmente separáveis.

2.**Kernel Sigmóide**: esta função é equivalente a um modelo perceptron de rede neural de duas camadas , que é usado como função de ativação para neurônios artificiais.

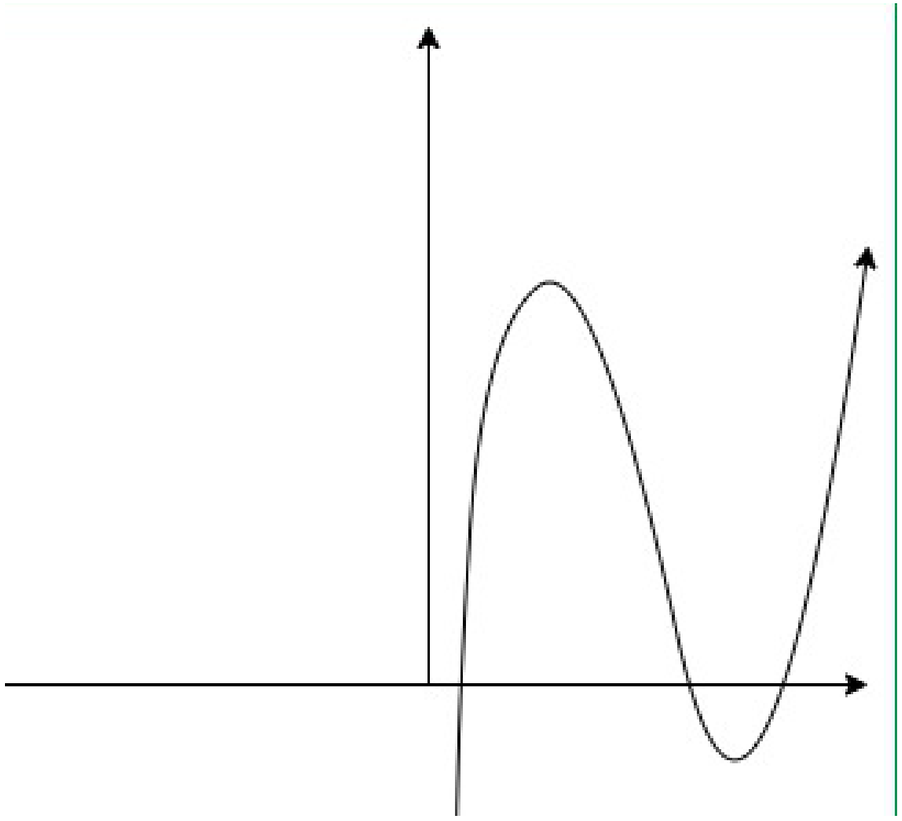




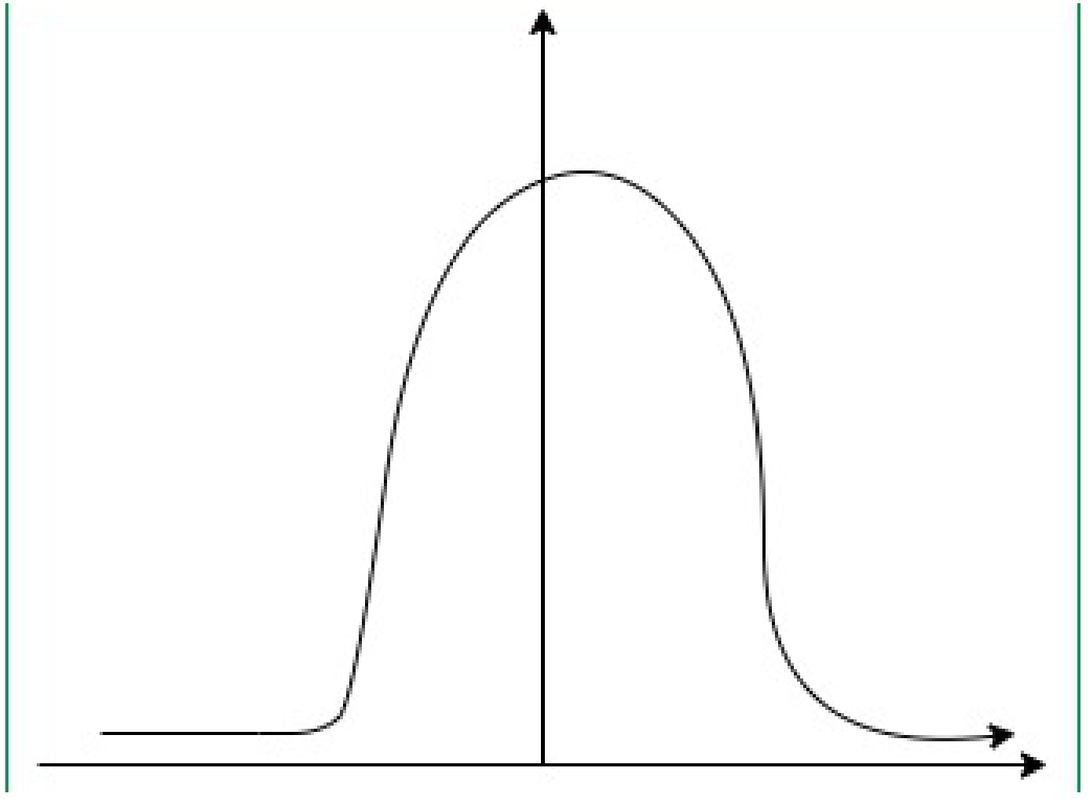
3.Kernel Polinomial: Ele representa a semelhança de vetores em treinamento conjunto de dados em um espaço de características sobre polinômios das variáveis originais usados no kernel.

4.Kernel Gaussiano(RBF - Radial Basis Function): É utilizado para realizar transformações, quando não há conhecimento prévio sobre os dados**.**

3



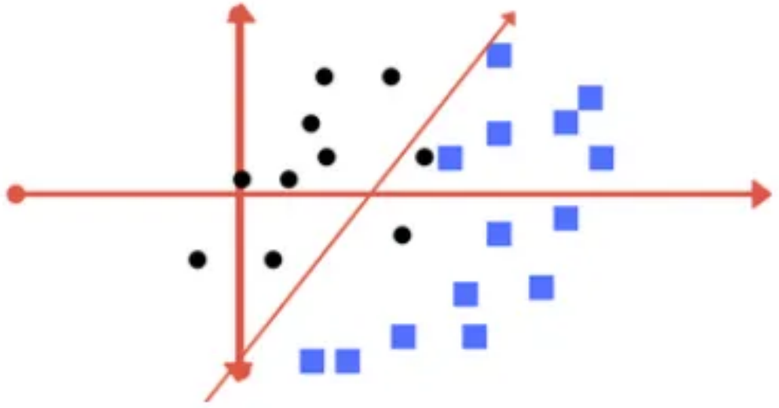
4



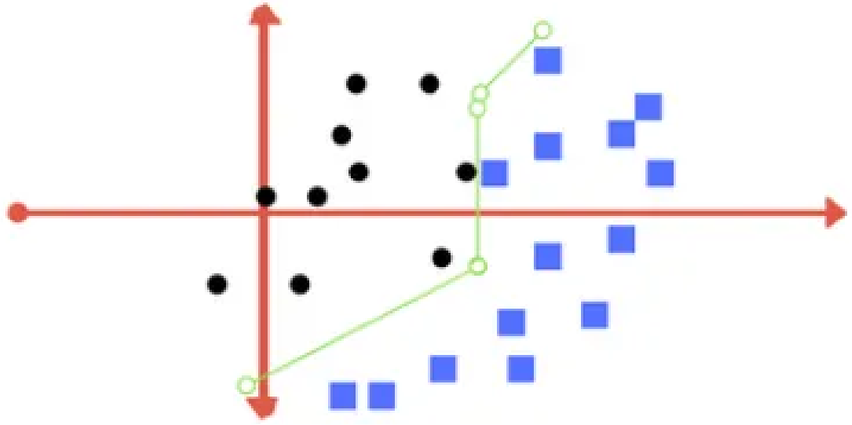
Hiperparâmetros

- Hiperparâmetros são responsáveis por adicionar versatilidade ao modelo, tolerando que algumas amostras sejam classificadas de forma incorreta (soft margin) ou tentando alcançar uma separação perfeita (hard margin).

- O ajuste dos hiperparâmetros serve para encontrar o equilíbrio certo entre a complexidade do modelo e a generalização para novos dados. Por conta disso, técnicas como validação cruzada e grid search são utilizadas para otimizar os hiperparâmetros do SVM.



Treinamento que é flexível a erros de classificação (Soft Margin)



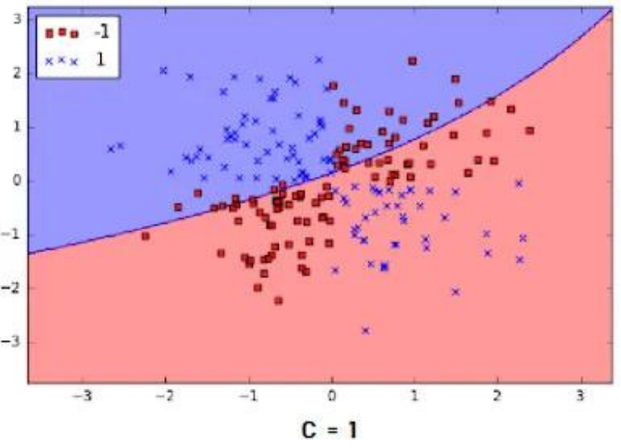
Treinamento que não tolera erros de classificação (Hard Margin)

Parâmetro C

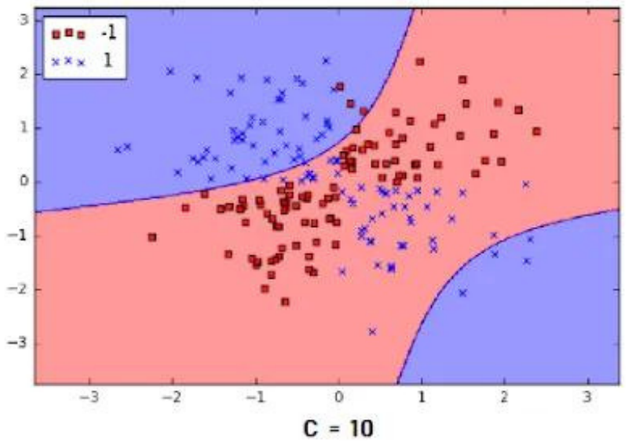
- Este parâmetro está presente tanto no SVM linear, quanto no não linear. É responsável por controlar o quão tolerante a erros de classificação será o modelo treinado.

- Um valor maior de C implica em menos regularização, permitindo que o modelo maximize a margem de decisão, mesmo que cause mais erros de classificação (Hard Margin).

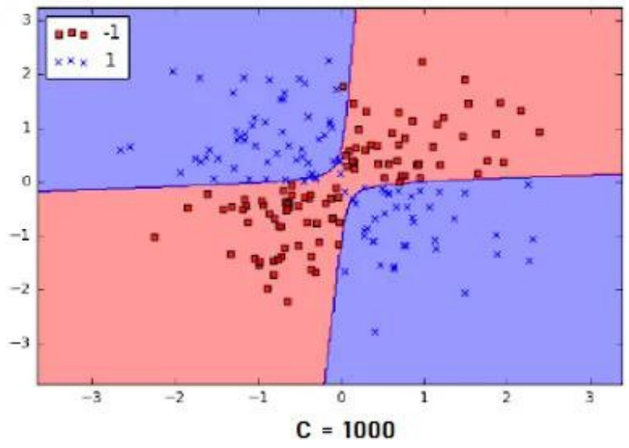
- Valores menores de C aumentam a regularização, o que pode aumentar a tolerância a erros, mas com uma margem menor (Soft Margin).



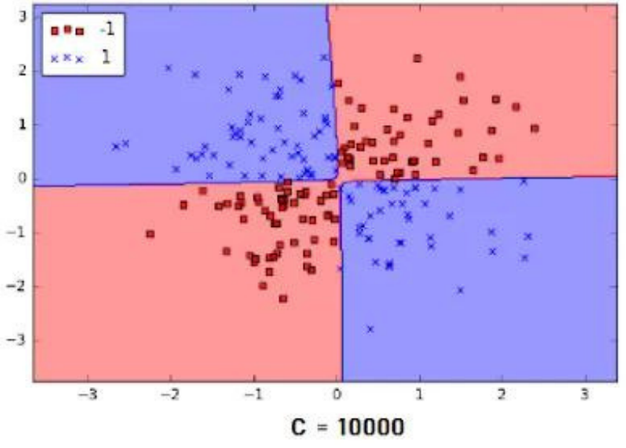
C = 1



C = 10



C = 1000



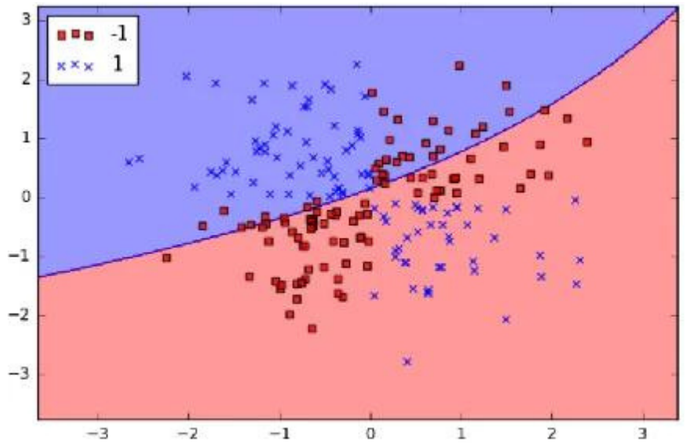
C = 10000

Parâmetro Gamma

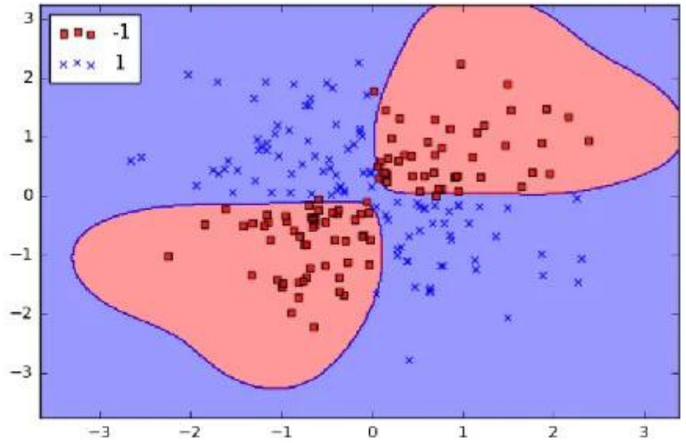
- Este parâmetro está presente apenas no SVM não linear, ele determina o quão longe a influência de um único ponto de treinamento alcança, com valores baixos significando 'longe' e valores altos significando 'perto'.

- Quando o gamma é baixo, a 'fronteira de decisão' do modelo é mais suave e mais abrangente. Isso pode levar a um modelo que generaliza melhor, pois não é muito influenciado por variações específicas nos dados de treinamento.

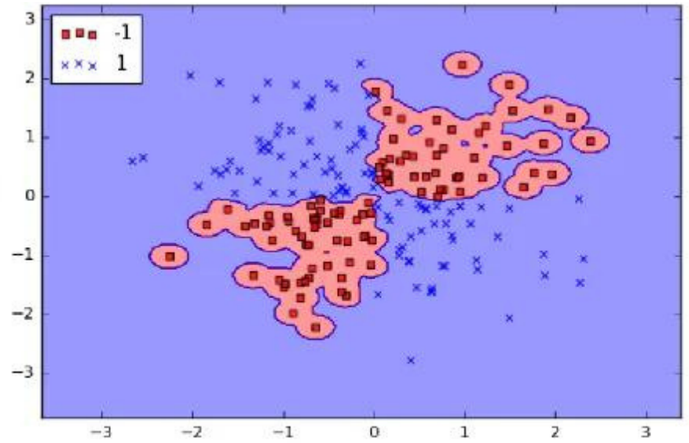
- Quando o gamma é alto, a 'fronteira de decisão' do modelo se torna mais rigorosa e detalhada. Isso faz com que o modelo se ajuste de forma mais precisa aos pontos de dados individuais, criando fronteiras que se adaptam a cada dado de treinamento.



Gamma = 0.01



Gamma = 1



Gamma = 100

BASE DE DADOS

Escolhemos um conjunto de dados contendo informações de 6 mil clientes de um banco, como idade, gênero, score de crédito, salário estimado, status de atividade, entre outros. Além disso, temos a variável alvo churn, que indica se o cliente deixou de usar os serviços do banco ou permaneceu como cliente ativo.

	customer_id	credit_score	country	gender	age	tenure	balance	products_number	credit_card	active_member	estimated_salary	churn
0	15765192	564	France	Male	26	7	84006.88	2	0	0	183490.99	0
1	15631882	688	Germany	Male	45	9	103399.87	1	0	0	129870.93	0
2	15777586	784	Spain	Female	42	2	109052.04	2	1	0	6409.55	0
3	15577107	657	Spain	Female	22	6	0.00	3	0	1	168412.07	1
4	15722731	653	France	Male	46	0	119556.10	1	1	0	78250.13	1

NORMALIZAÇÃO DA BASE

- Standart Scaler()

- One-Hot-Enconding nas variáveis categóricas

credit_score	gender	age	balance	credit_card	active_member	estimated_salary	churn	country_Germany	country_Spain
-0.884431	1	-1.231208	0.117784	0	0	1.438108	0	0	0
0.395807	1	0.555991	0.428057	0	0	0.508944	0	1	0
1.386959	0	0.273802	0.518487	1	0	-1.630477	0	0	1
0.075747	0	-1.607460	-1.226258	0	1	1.176810	1	0	1
0.034449	1	0.650055	0.686543	1	0	-0.385576	1	0	0

TREINAMENTO DO MODELO

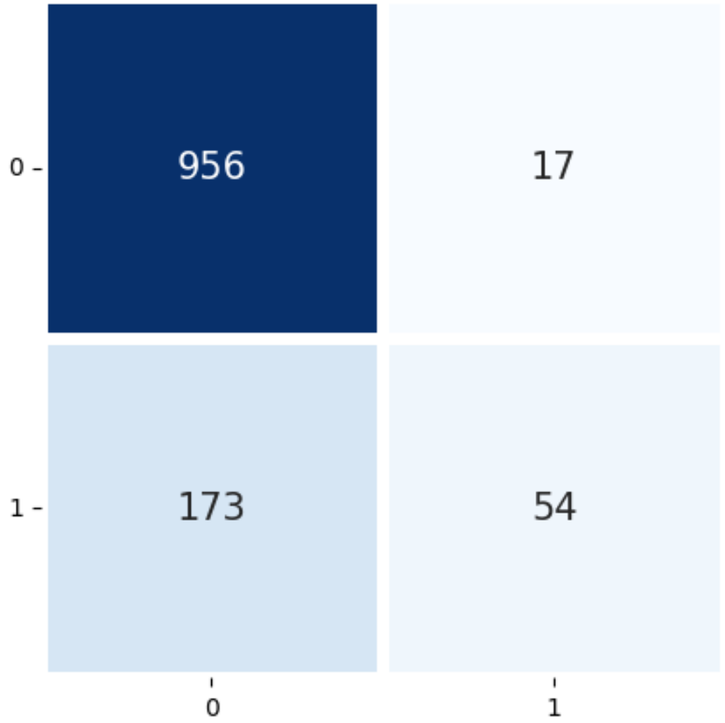
```
# Separando os dados em variáveis preditoras e variável alvo
X = df.drop('churn', axis=1)
y = df['churn']

# Dividindo os dados em conjuntos de treino e teste
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Construindo o modelo SVM
svm_model = SVC()
svm_model.fit(X_train, y_train)

param_dist = {'C': [0.1, 1, 10, 100], 'kernel': ['linear', 'rbf', 'poly']}
random_search = RandomizedSearchCV(svm_model, param_distributions=param_dist, n_iter=5, cv=5)
random_search.fit(X_train, y_train)
```

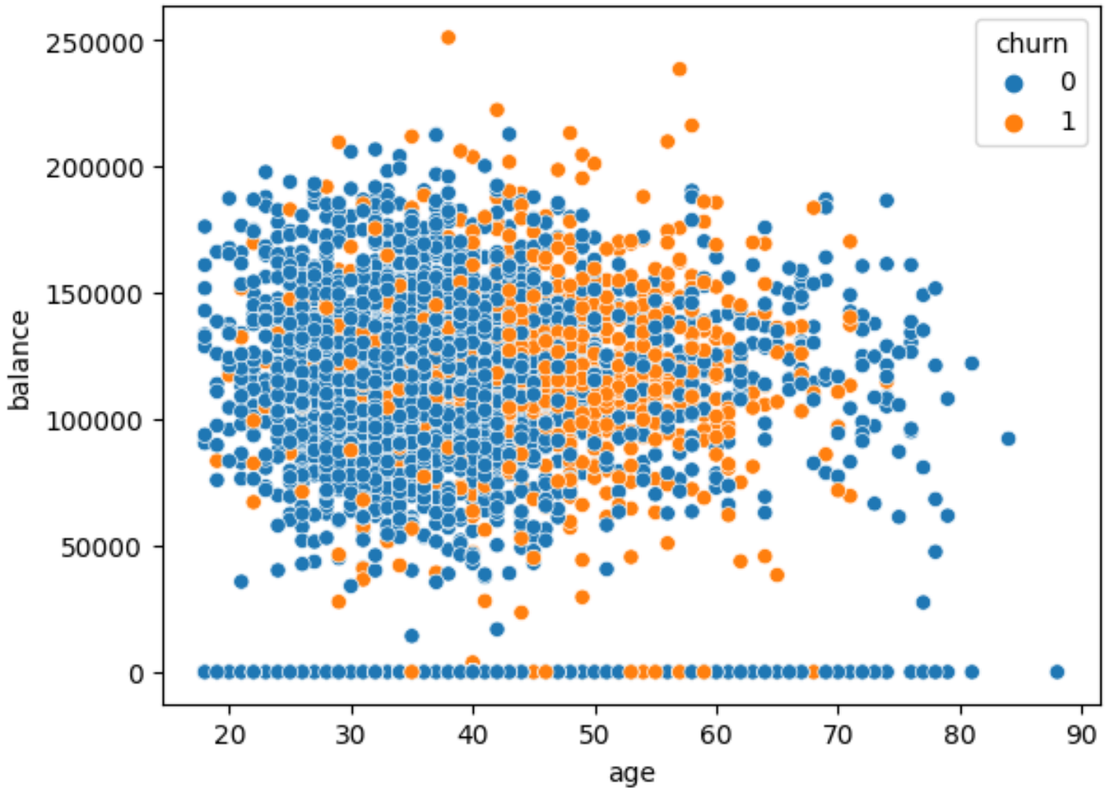
```
Melhores Parâmetros: {'kernel': 'rbf', 'C': 1}
```

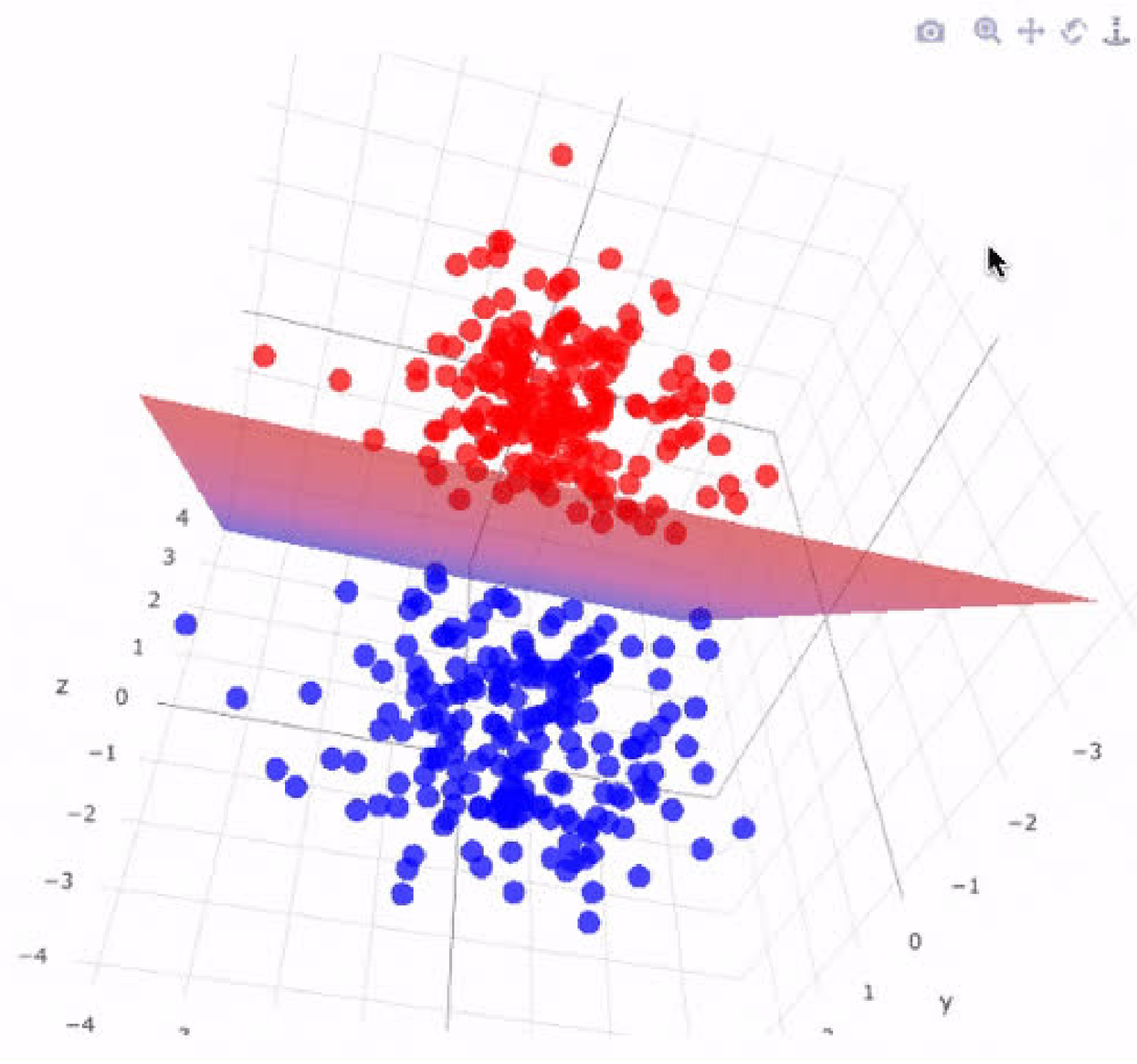
RESULTADO

Relatório de Classificação:

	precision	recall	f1-score	support
0	0.85	0.98	0.91	973
1	0.76	0.24	0.36	227
accuracy			0.84	1200
macro avg	0.80	0.61	0.64	1200
weighted avg	0.83	0.84	0.81	1200




churn
0
1
balance
age
250000
200000
150000
100000
50000
0
20
30
40
50
60
70
80
90


z
y
4
3
2
1
0
−1
−2
−3
−4