

Support Vector Machines

Claudia Chávez

Support Vector Machine

- Partió en los años 1960's y luego fue mejorado durante los años 1980's
 - Vladimir Vapnik y su equipo en laboratorios de AT&T
- Es uno de los algoritmos más populares y antiguos utilizados
- Las Máquinas de Soporte Vectorial (Support Vector Machines SVMs) son algoritmos de aprendizaje supervisados que desarrollan métodos relacionados con los problemas de clasificación y regresión.

Concepto de hiperplano

- En un espacio de dos dimensiones un hiperplano es lo mismo que una ecuación de la recta definida por:

$$\beta_0 + \beta_1 X_1 + \beta_2 X_2 = 0$$

- $X = X_1, X_2$ corresponde a un punto de dicha línea recta
- Si consideramos un espacio p dimensional, la extensión de la ecuación anterior estaría definida por:

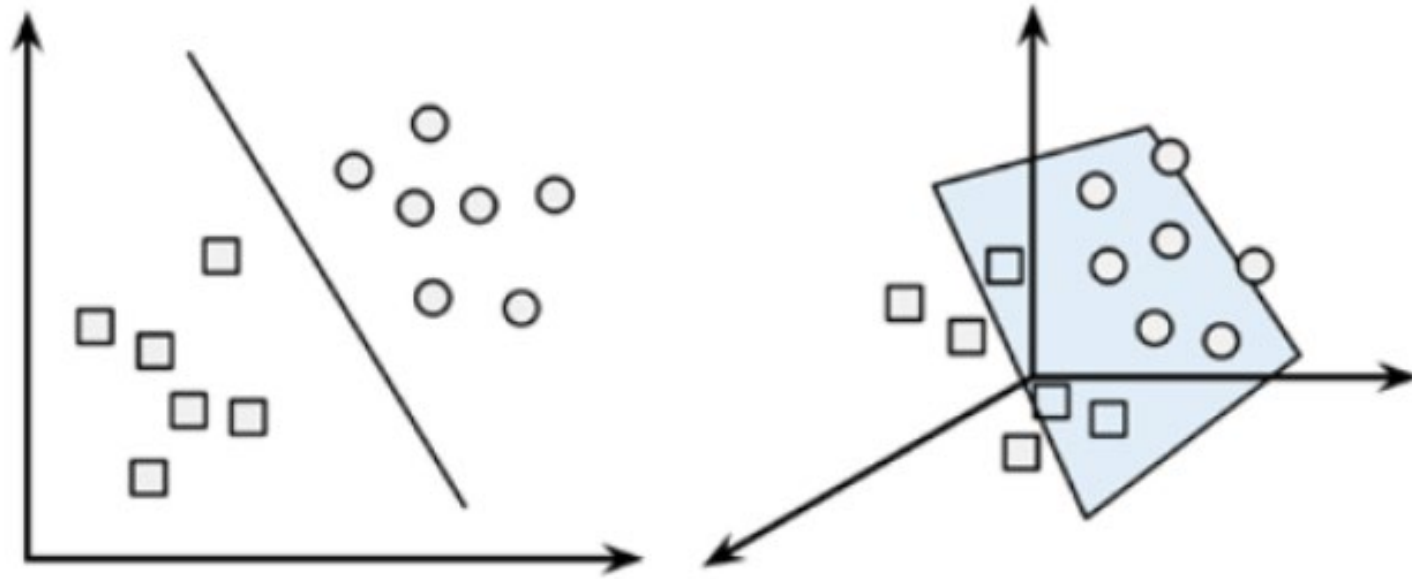
$$\beta_0 + \beta_1 X_1 + \beta_2 X_2 + \dots + \beta_p X_p = 0$$

- $X = X_1, X_2, \dots, X_p$ corresponde a un punto esta vez del hiperplano

Support Vector Machine

- Es una representación de puntos en el espacio, mapeados para que cada categoría este dividida con el espacio **más ancho posible**.
- Ese ancho define la separación óptima de dividir los datos de forma de clasificar de la mejor manera posible.
 - Este tipo de algoritmos buscan el hiperplano que tenga la máxima distancia (margen) con los puntos que estén más cerca de él mismo.
 - Clasificadores de **margen máximo**.
 - Los puntos del vector que son etiquetados con una clase estarán a un lado del hiperplano y los casos que se encuentren en la otra clase estarán al otro lado.

Separación optima lineal



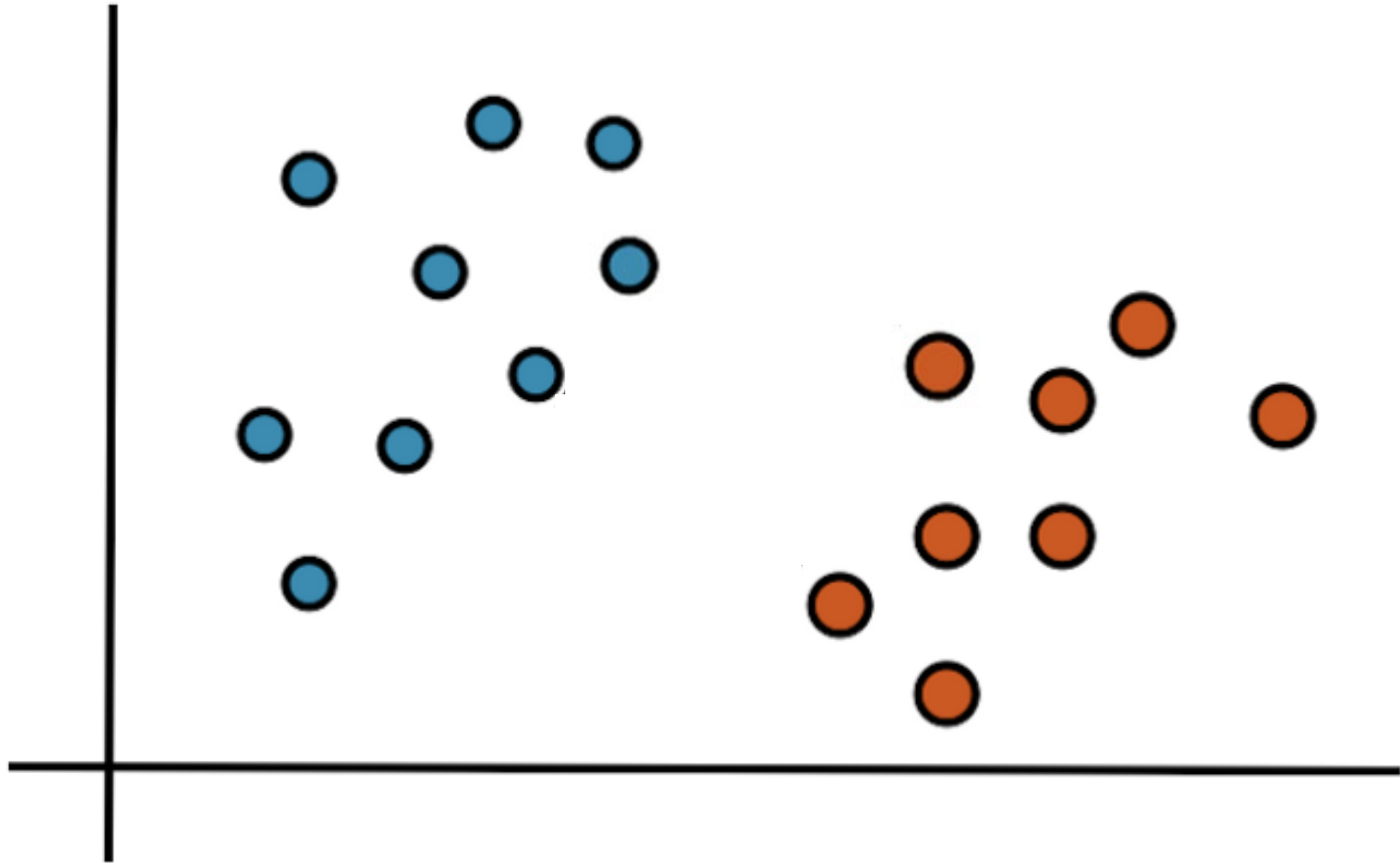


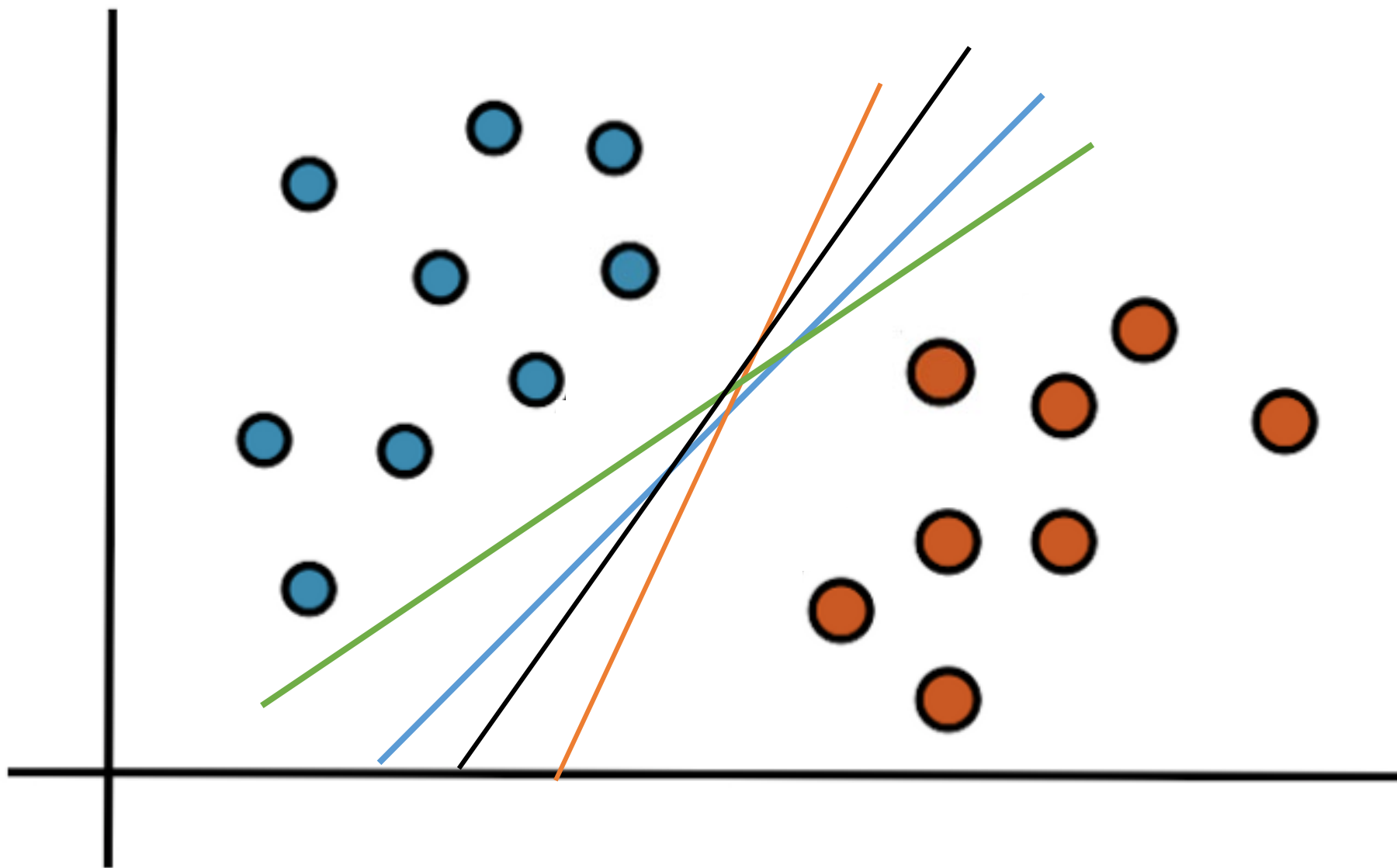
Support Vector Machine


- Parten de un conjunto de datos de entrenamiento
- Se etiquetan los casos en distintas clases y se representan los puntos en el espacio.
- Se espera obtener un espacio lo más amplio posible, para que cuando los datos de test que se evalúen en algoritmo sean clasificadas correctamente en función de su proximidad.

Dos dimensiones

- El objetivo del algoritmo SVM es encontrar dicho hiperplano
- El algoritmo solo puede encontrar este hiperplano en problemas que permiten separación lineal; en la mayoría de los problemas prácticos, el algoritmo maximiza el margen flexible permitiendo un pequeño número de clasificaciones erróneas.
- Ejemplo de datos linealmente separables

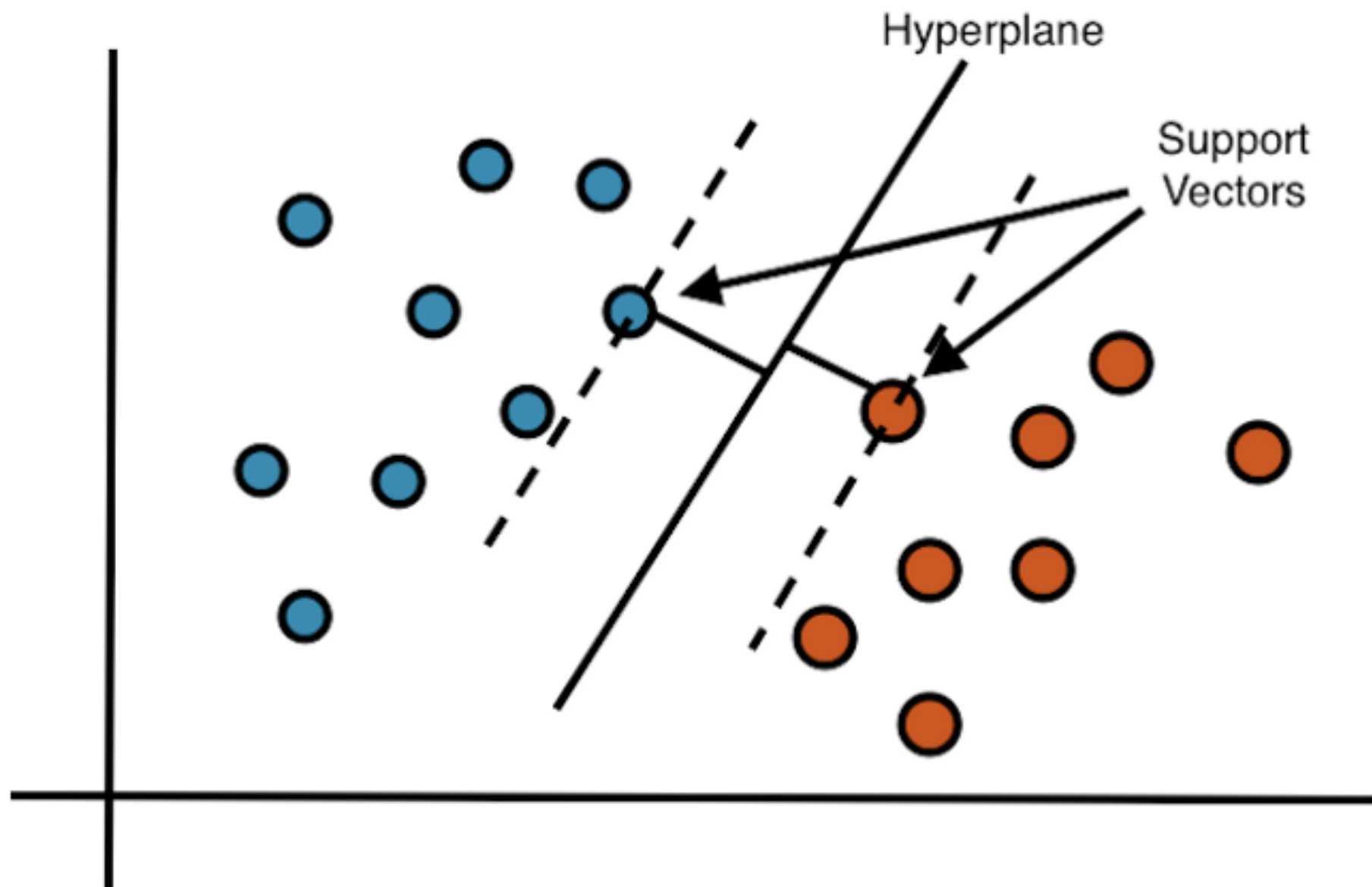






Tengo muchas
opciones
posibles...
¿entonces?

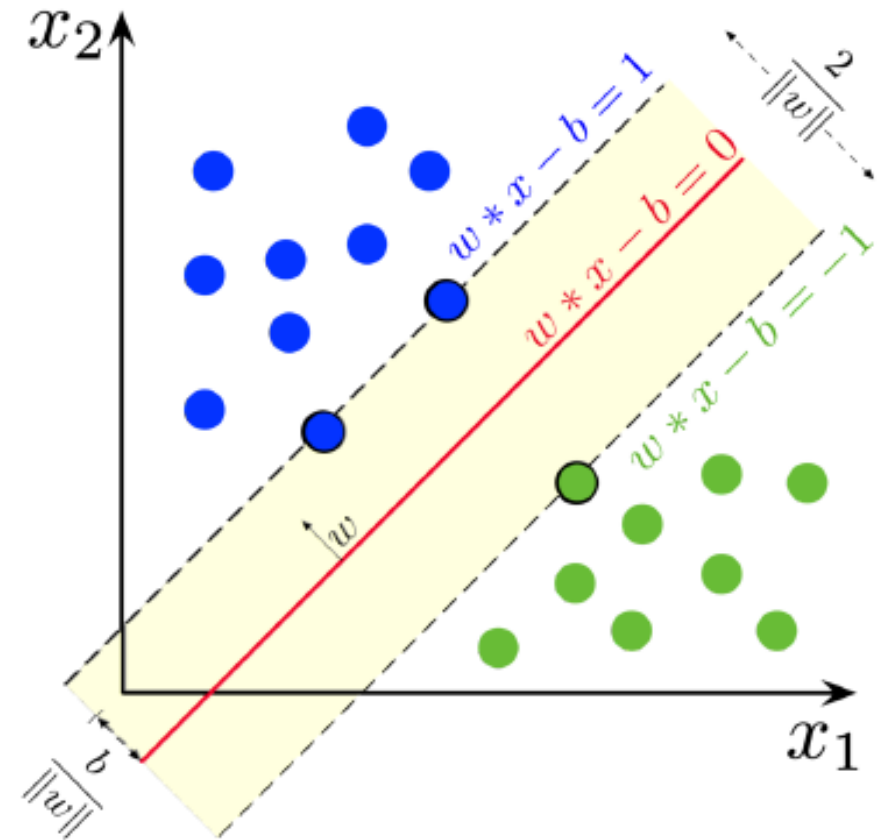
- Lo resuelvo con un hiperplano de máximo margen (MMH)
- Para esto utilizo vectores de soporte.
 - Se basa en observaciones de las clases que se encuentran mas cercanas
 - Aquellas que maximizar el margen del hiperplano
 - Y permiten dividir a los datos de entrenamiento



Esto implica que...

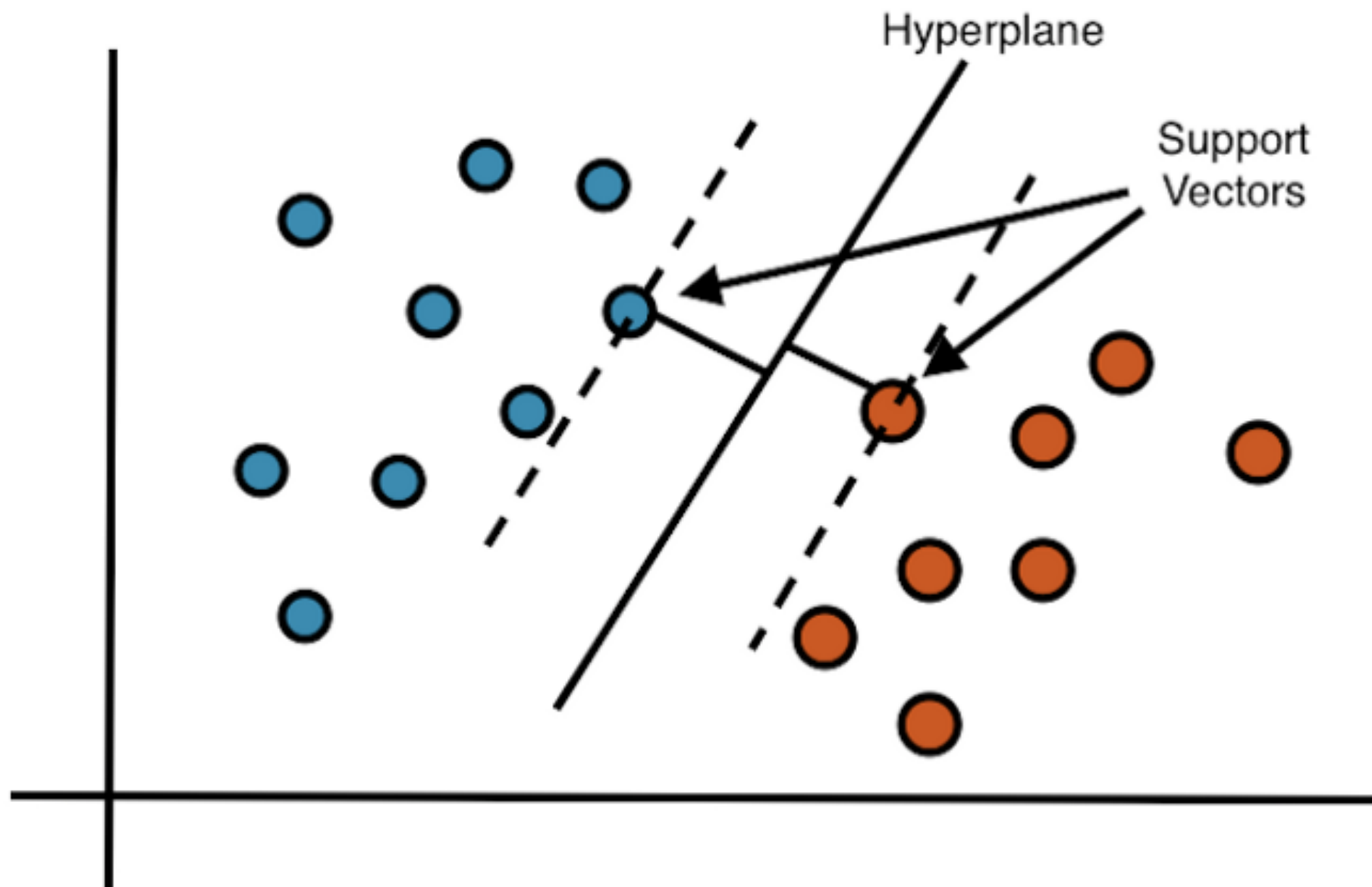
Se deben calcular las distancias perpendiculares de cada observación a un hiperplano dado, donde la distancia más pequeña se corresponde con la distancia mínima de las observaciones al hiperplano, espacio conocido como **margen**.

El hiperplano óptimo de separación el que tiene la mayor distancia mínima de las observaciones al hiperplano, es decir el mayor margen.



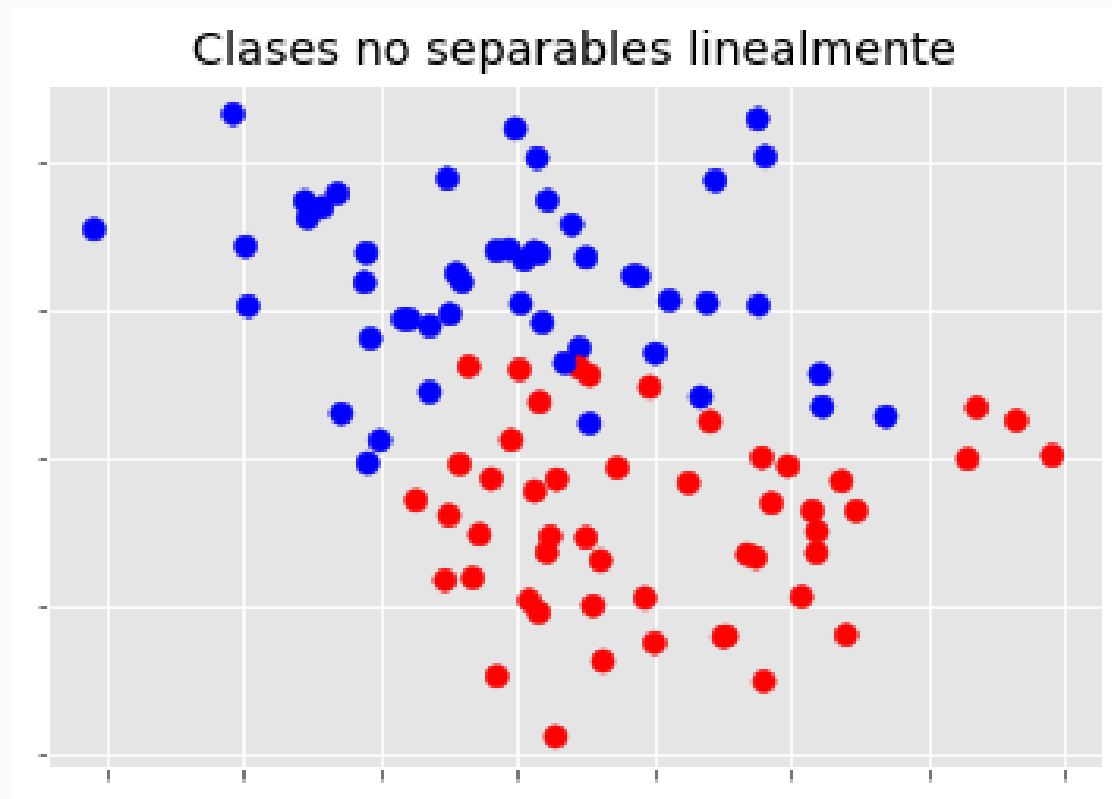
Regularización

- Ya hemos visto que los datos tienen ruido, esto es que no estén clasificados perfectamente.
 - Generalización
 - Overfitting
- Recordemos que queremos hacer un balance entre ambos conceptos



Casos no separables linealmente

- Generalmente los datos no se pueden separar linealmente de forma perfecta
 - No existe un hiperplano de separación y no puede obtenerse un *maximal margin hyperplane*.
- Para solucionar esto se extiende el concepto de *maximal margin hyperplane* para obtener un hiperplano que "casi" separe las clases, pero permitiendo que se cometan unos pocos errores.
 - *Support Vector Classifier* o *Soft Margin*.



Soft Margin classifier

- En estos casos, puede ser útil considerar el clasificador denominado **soft margin classifier** o *support vector classifier*, que, aún basado en un hiperplano, no separe perfectamente las dos clases, con el interés de obtener:
 - Mayor robustez a observaciones individuales
 - Mejor clasificación de la mayoría de las observaciones de entrenamiento y test.

Optimización (parámetro de regularización o tuning)

- El proceso incluye un hiperparámetro C , que es el que define cuanto me puedo equivocar al seleccionar el hiperplano.
- C controla el número y severidad de las violaciones del margen (y del hiperplano) que se toleran en el proceso de ajuste.
 - Controla el bias y la varianza del modelo

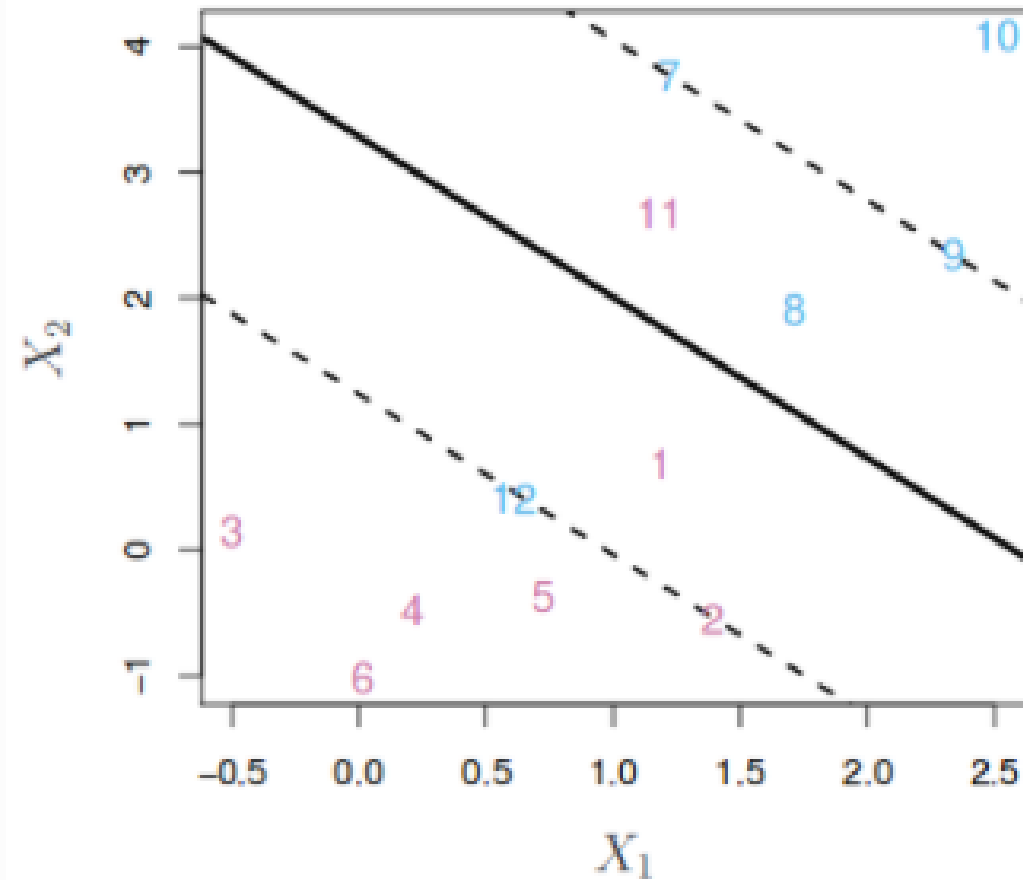
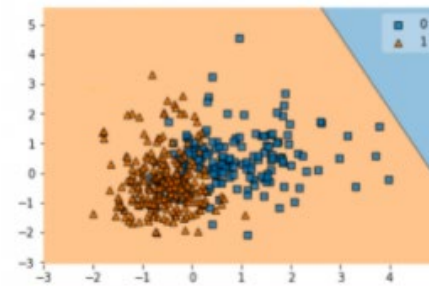


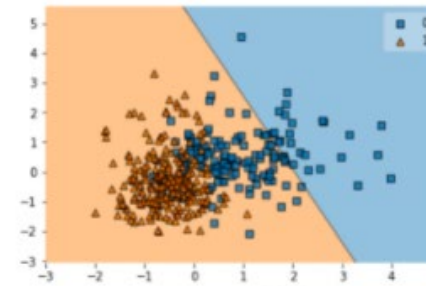
Imagen clasificador vector soporte obtenida del libro ISLR

Optimización

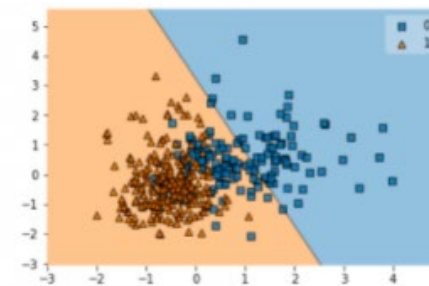
- C mayor que 0, no más de C observaciones se encuentran clasificadas de forma errónea.
 - Esto equivale a un modelo bastante bien ajustado a los datos, el cual puede tener poco bias pero mucha varianza
- Valores de C mayores implica mayor tolerancia de “error”.
 - Esto equivale a un modelo más flexible y con mayor *bias* pero menor varianza
- Valores de $C=0$ es equivalente al maximal margin classifier, pues no están permitidas violaciones sobre el margen.
 - 100% de las observaciones clasificadas correctamente



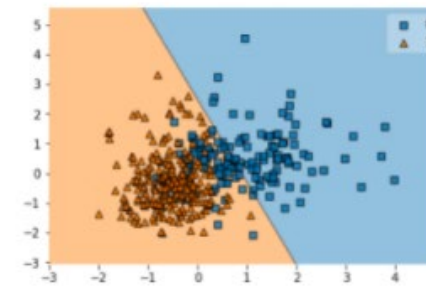
C = 0.001
Accuracy: 62.6%



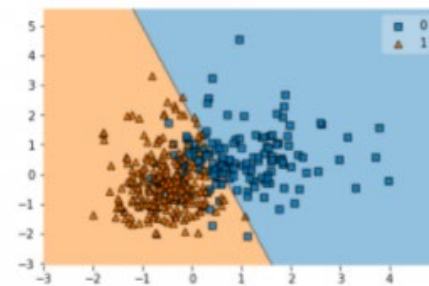
C = 0.002
Accuracy: 71.3%



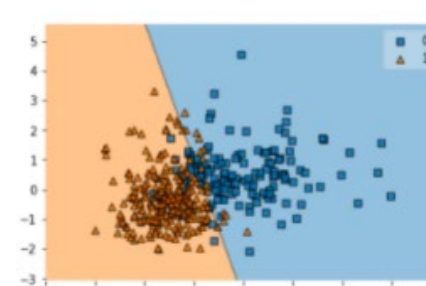
C = 0.003
Accuracy: 81.3%



C = 0.005
Accuracy: 81.7%



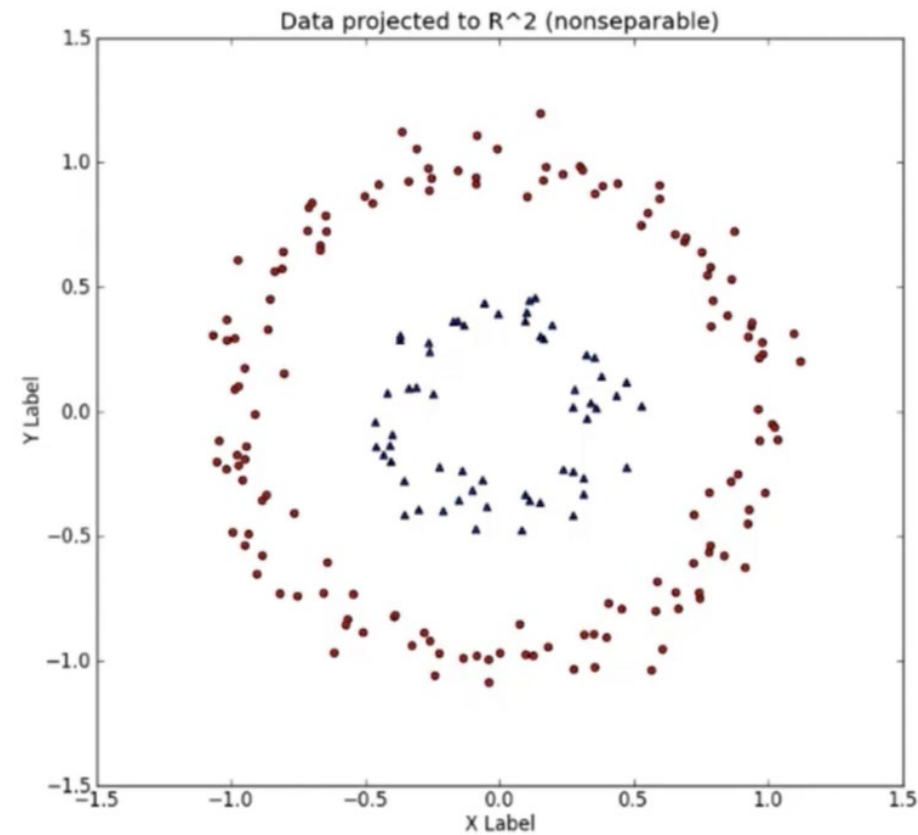
C = 0.01
Accuracy: 89.5%



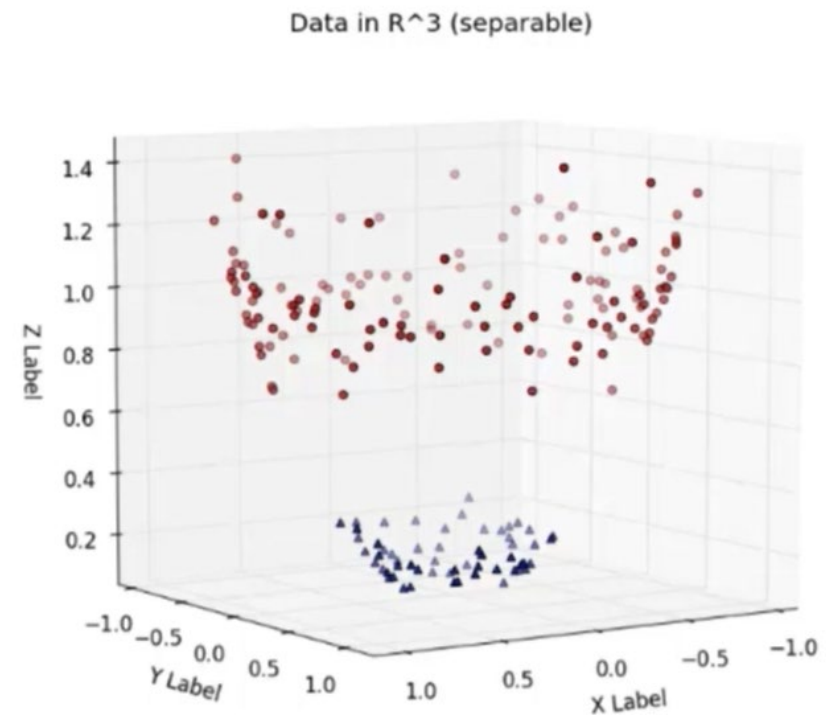
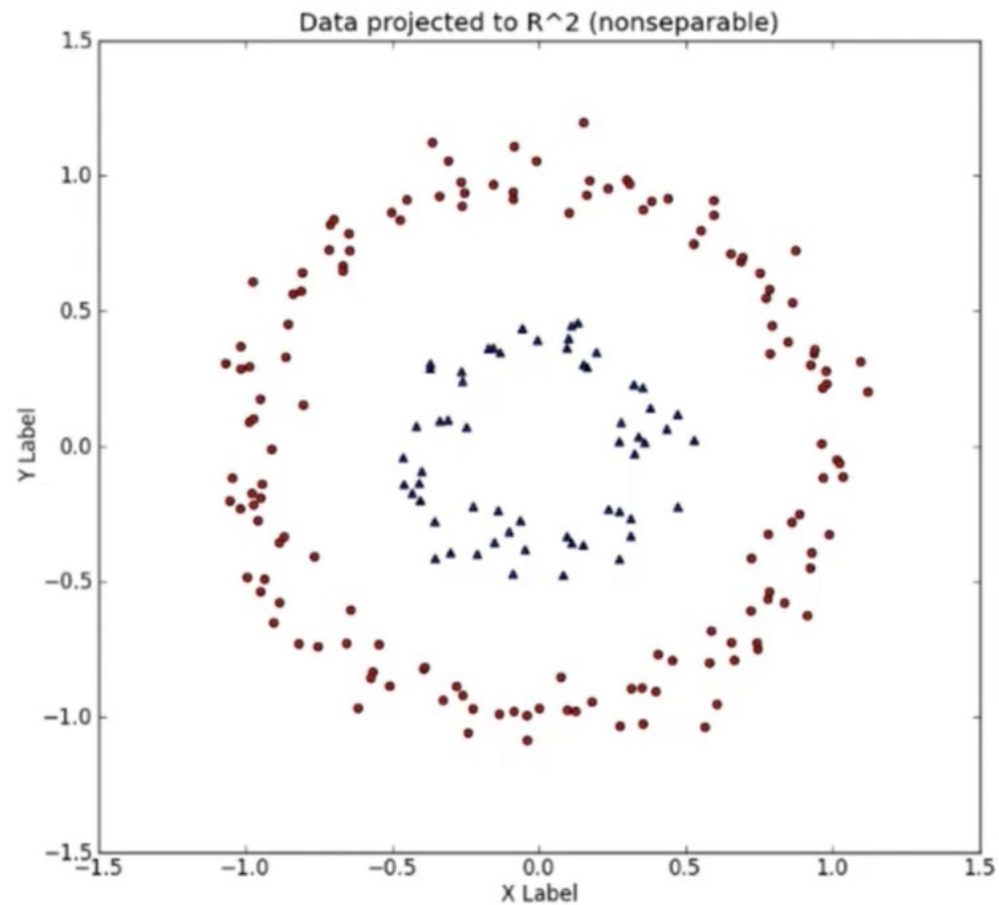
C = 10.0
Accuracy: 90.1%

Fig 3 Decision boundaries for different C Values for Linear Kernel

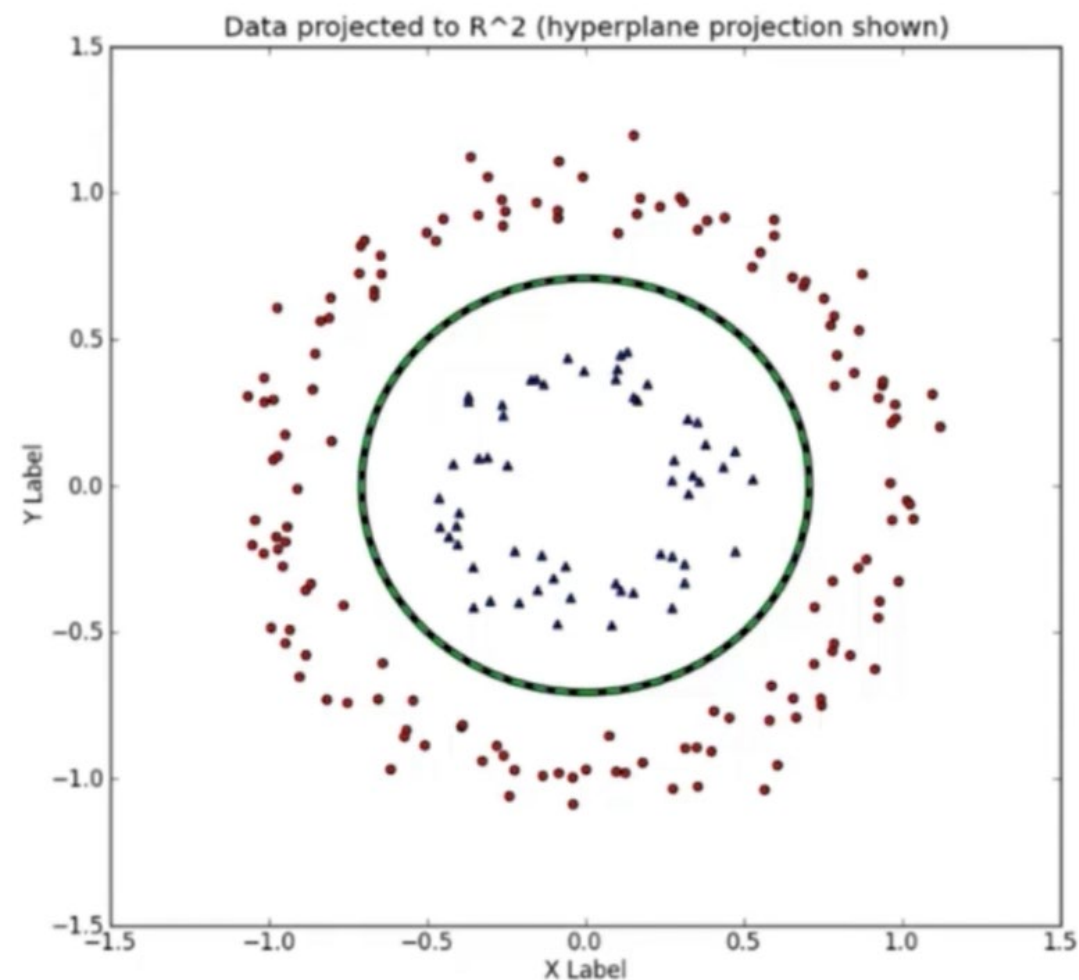
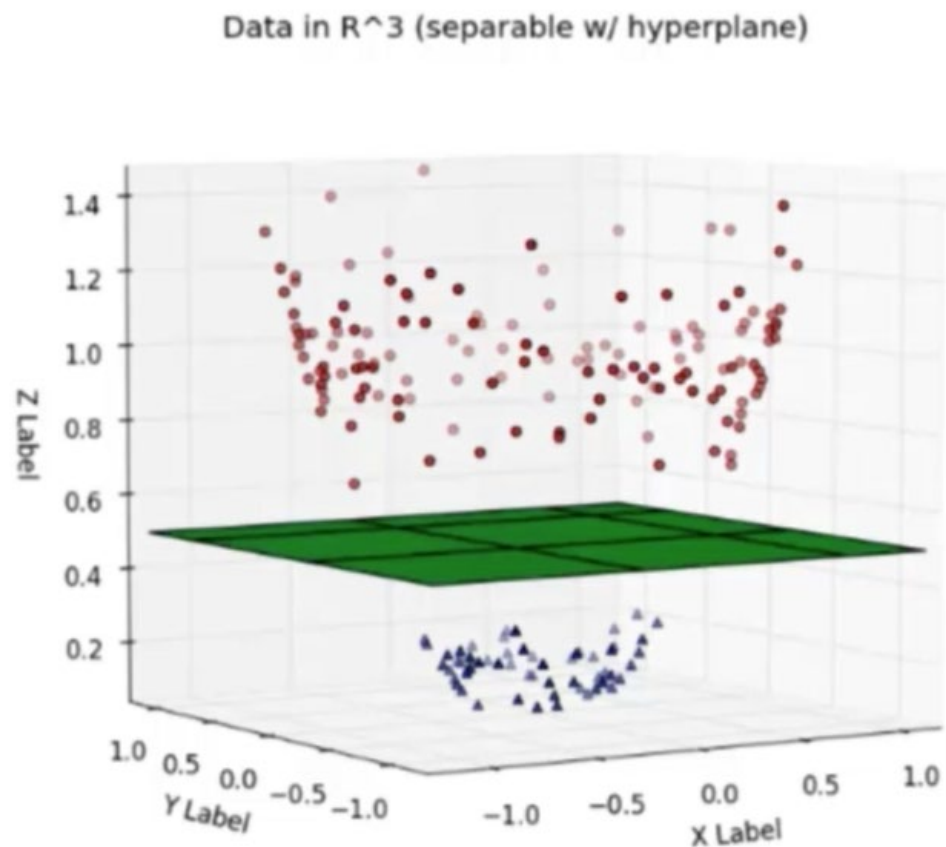
Ahora, ¿Cómo resuelvo este caso? No se puede separar linealmente



El truco del kernel



El truco del kernel

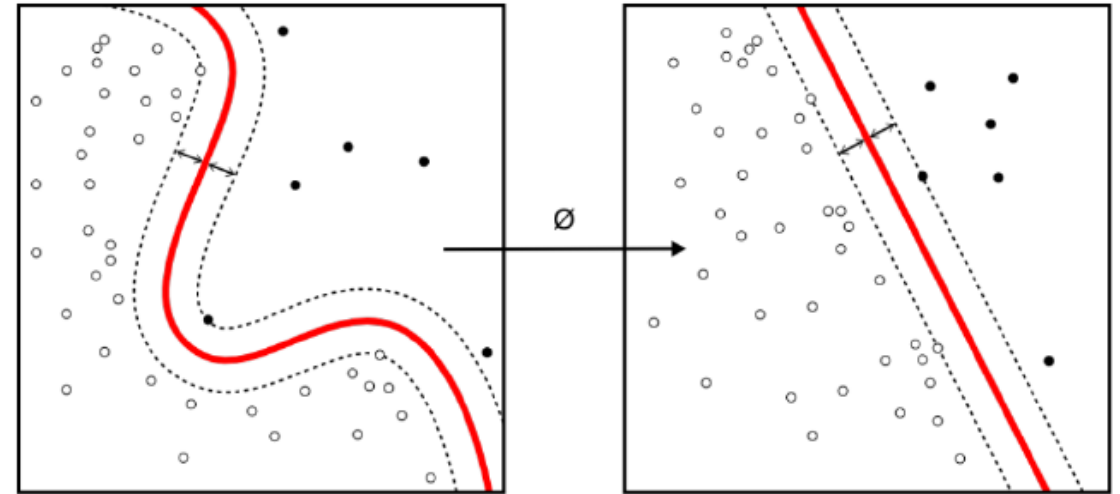


Función kernel

- Asignan los datos en un espacio dimensional diferente, generalmente superior, para facilitar la clasificación.
 - Paso de 2 a 3 dimensiones
- Luego de la transformación, se simplifica los límites de decisión más complejos no lineales, dejándolos lineales en espacio definido.
- Debido a que SVM no utiliza todos los datos para la estimación hace que este proceso no se tiene que realizar (la transformación) en todos los datos.
 - Mejora la carga computacional
 - Esto se conoce como truco de kernel

Kernel en SVM

- Los algoritmo de svm se denominan métodos kernel



Funciones existentes (kernel)

- La librería **Scikit Learn** contiene implementaciones en **Python** de los principales algoritmos de *SVM*.

Tipo de SVM	Kernel de Mercer	Descripción
Función de base radial (RBF) o gaussiana	$K(x_1, x_2) = \exp\left(-\frac{\ x_1 - x_2\ ^2}{2\sigma^2}\right)$	Aprendizaje de una clase. σ representa la anchura del kernel.
Lineal	$K(x_1, x_2) = x_1^T x_2$	Aprendizaje de dos clases.
Polinómica	$K(x_1, x_2) = (x_1^T x_2 + 1)^\rho$	ρ representa el orden del polinomio.
Sigmoide	$K(x_1, x_2) = \tanh(\beta_0 x_1^T x_2 + \beta_1)$	Representa un kernel de Mercer solo para determinados valores β_0 y β_1 .

Kernel lineal

- Si se emplea un Kernel lineal, el clasificador *Support Vector Machine* obtenido es equivalente al *Support Vector Classifier*.
- Sirve para el aprendizaje en dos clases
- Se representa como:

$$K(\mathbf{x}, \mathbf{x}') = \mathbf{x} \cdot \mathbf{x}'$$

Kernel Polinómico

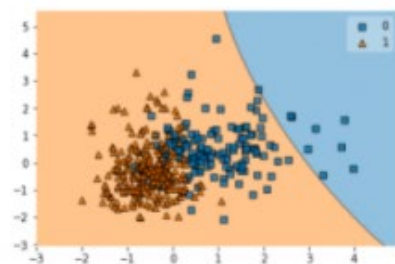
$$K(\mathbf{x}, \mathbf{x}') = (\mathbf{x} \cdot \mathbf{x}' + c)^d$$

- Cuando $d=1$ y $c=0$, corresponde a un kernel lineal.
- Si $d>1$, Aumenta la linealidad a medida que d crece.
- No suele ser recomendable emplear valores de d mayores 5 por problemas de overfitting.

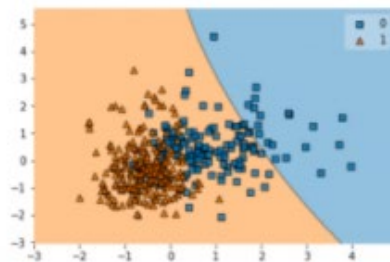
Kernel de base radial o RBF

$$K(\mathbf{x}, \mathbf{x}') = \exp(-\gamma \|\mathbf{x} - \mathbf{x}'\|^2)$$

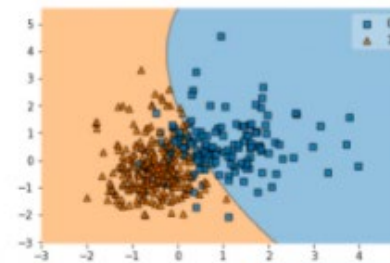
- γ controla el comportamiento del kernel
- Si su valor es pequeño corresponde a un modelo lineal.
- A medida que aumenta su valor, también lo hace la flexibilidad del modelo.



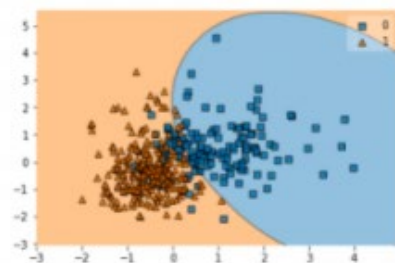
Gamma = 0.008
Accuracy: 63.7%



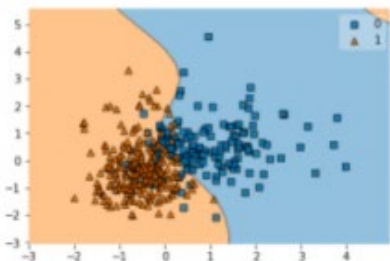
Gamma = 0.01
Accuracy: 68.4%



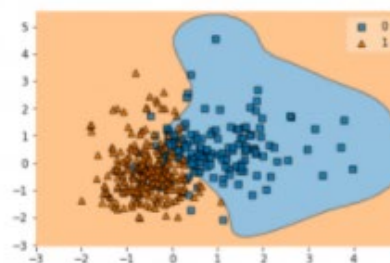
Gamma = 0.05
Accuracy: 88.9%



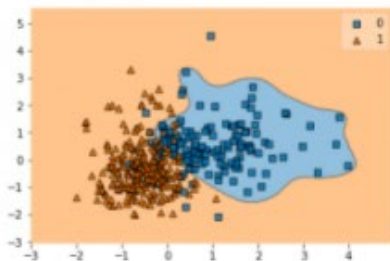
Gamma = 0.1
Accuracy: 90.1%



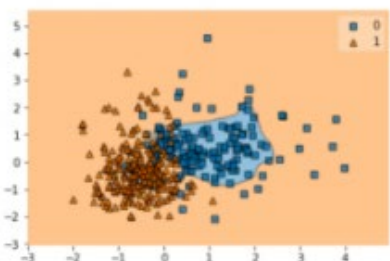
Gamma = 0.5
Accuracy: 91.6%



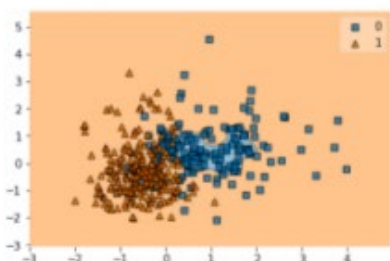
Gamma = 1.0
Accuracy: 90.1%



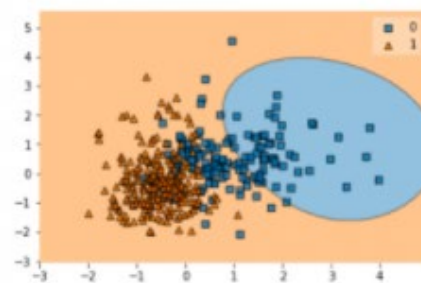
Gamma = 3.0
Accuracy: 88.9%



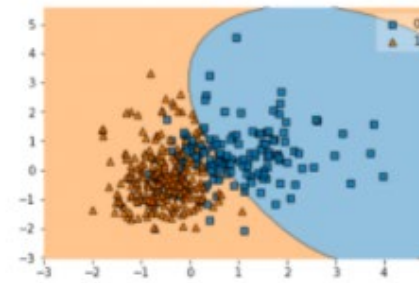
Gamma = 7.0
Accuracy: 84.8%



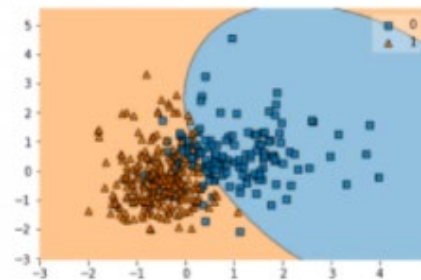
Gamma = 11.0
Accuracy: 74.9%



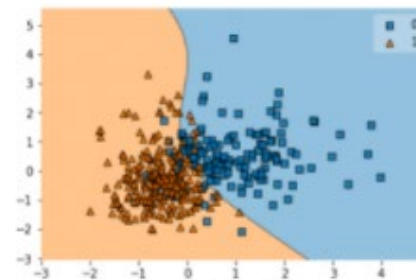
C = 0.02
Accuracy: 81.3%



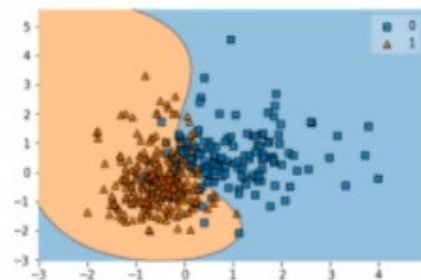
C = 0.03
Accuracy: 88.3%



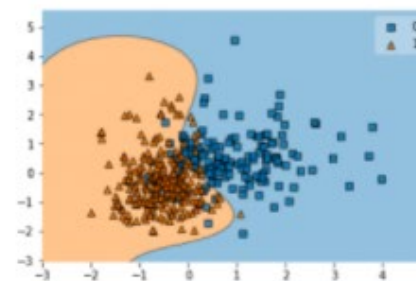
C = 0.08
Accuracy: 90.6%



C = 1.0
Accuracy: 90.6%



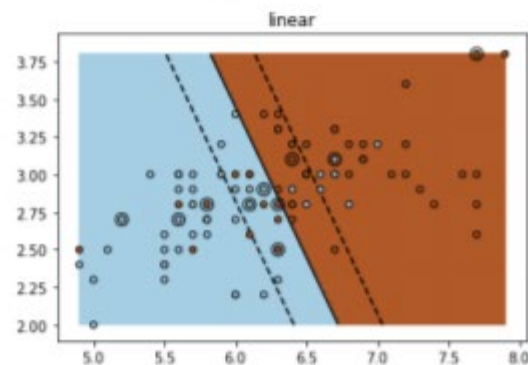
C = 10.0
Accuracy: 90.1%



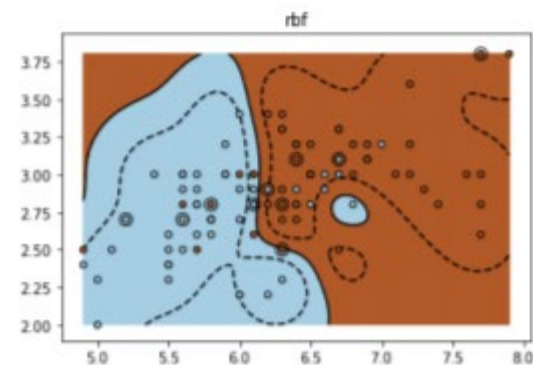
C = 100.0
Accuracy: 90.1%

Comparación kernels

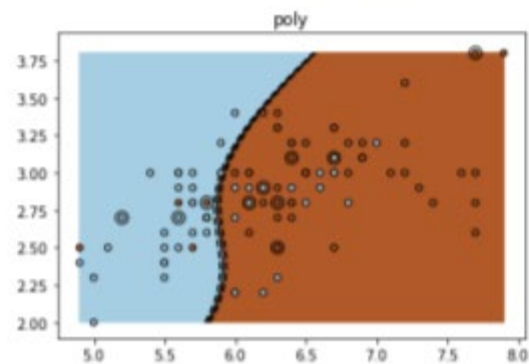
Lineal $C=1$



Gaussiano $C=1, \gamma=10$



Polinómico $C=1, orden=10$



Kernel sigmoide

- Similar a la función sigmoide de regresión logística
- Se utiliza principalmente en redes neuronales.
- Esta función del núcleo es similar a un modelo de **perceptrón** de dos capas de la red neuronal, que funciona como una función de activación para las neuronas.

$$K(X, Y) = \tanh(\gamma \cdot X^T Y + r)$$

SVM Multiclase

- Generalmente se divide en una serie de clasificaciones binarias
- Hay dos filosofías básicas para resolver el problema de querer clasificar los datos en más de dos categorías:
 - Cada categoría es dividida en otras y todas son combinadas.
 - Se construyen $k(k-1) / 2$ modelos donde k es el número de categorías.

One versus One

- Para $K > 2$ clases y que se quiere aplicar el método de clasificación basado en *SVMs*.
- Consiste en generar un total de $K(K-1)/2$ *SVMs*
 - *Se comparan todos los pares de clases.*
- Para generar una predicción, se emplean cada uno de los $K(K-1)/2$ clasificadores, registrando el número de veces que la observación es asignada a cada una de las clases.
- Finalmente, se considera que la observación pertenece a la clase a la que ha sido asignada con más frecuencia.
- La principal desventaja de esta estrategia es que el número de modelos necesarios se dispara a medida que aumenta el número de clases, por lo que no es aplicable en todos los escenarios.

One-versus-all

- Esta estrategia consiste en ajustar K SVMs distintos, cada uno comparando una de las K clases frente a las restantes $K-1$ clases.
- Como resultado, se obtiene un hiperplano de clasificación para cada clase.
- Para la predicción se usan cada uno de los K clasificadores y se asigna la observación a la clase para la que la predicción resulte positiva.
- Esta aproximación, aunque sencilla, puede causar inconsistencias, ya que puede ocurrir que más de un clasificador resulte positivo, asignando así una misma observación a diferentes clases.
- Otro inconveniente adicional es que cada clasificador se entrena de forma no balanceada.
 - Por ejemplo, si el set de datos contiene 100 clases con 10 observaciones por clase, cada clasificador se ajusta con 10 observaciones positivas y 990 negativas.

DAGSVM (Directed Acyclic Graph SVM)

- Es una mejora del método *one-versus-one*.
- La estrategia seguida es la misma, pero consiguen reducir su tiempo de ejecución eliminando comparaciones innecesarias gracias al empleo de una *directed acyclic graph (DAG)*.
- Set de datos con cuatro clases (A, B, C, D) y 6 clasificadores entrenados con cada posible par de clases (A-B, A-C, A-D, B-C, B-D, C-D).
 - Se inician las comparaciones con el clasificador (A-D) y se obtiene como resultado que la observación pertenece a la clase A (no D)
 - Se excluyen todas las comparaciones con D.
 - En la siguiente comparación se emplea el clasificador (A-C) y se predice que es A (no C).
 - Finalmente solo queda emplear el clasificador (A-B) y asignar la observación al resultado devuelto.
- En lugar de emplear los 6 clasificadores, solo ha sido necesario emplear 3. *DAGSVM* tiene las mismas ventajas que el método *one-versus-one* pero mejorando mucho el rendimiento.

Ventajas SVM

- Efectivo para datos multi-dimensionales
- Es eficaz cuando el numero de casos es menor a la cantidad de variables
- Solo utiliza algunos puntos de la data de entrenamiento, esto lo hace mas eficiente
- Las distintas funciones de kernel lo hacen más flexible
- SVM (máquina de vectores de soporte) es una técnica de clasificación y regresión que aprovecha al máximo la precisión de las predicciones de un modelo sin ajustar excesivamente los datos de entrenamiento.

Desventajas SVM

- Cuando el numero de feature es mucho mayor a la muestra puede producir overffiting, por lo que la función kernel y la regularización es muy importate.
- SVM no proporciona una probabilidad directa, sino que una aproximada con cross validation con cinco iteraciones.
- Tiene un costo computacional alto en multiclase por todas las divisiones que debe hacer para obtener resultados.