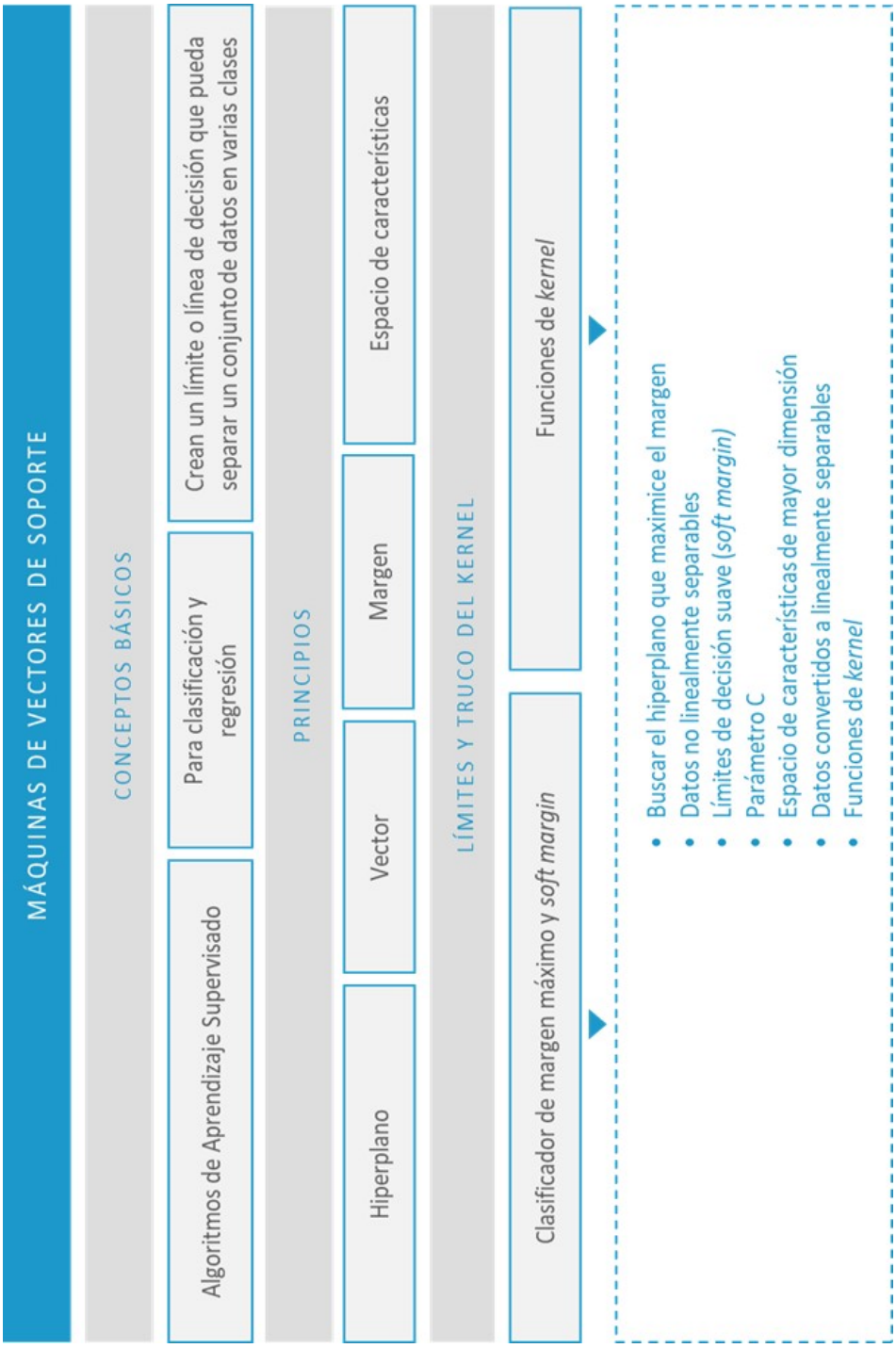
Técnicas de Aprendizaje Automático

Tema 7. Máquinas de vectores de soporte

Índice

Esquema



7.1. Introducción

Las **máquinas de vectores de soporte** o máquinas de vector soporte (del inglés *support-vector machines*, SVM) son un conjunto de algoritmos de aprendizaje supervisado desarrollados por Cortes y Vapnik, 1995. Estos métodos en su propuesta original estaban orientados a resolver una tarea de clasificación binaria. Más adelante, SVM fue extendido a los problemas de regresión.

Los SVM son uno de los modelos más estudiados y se basan en modelos de aprendizaje estadístico.

El objetivo del algoritmo SVM es crear un límite o línea de decisión que pueda separar ciertamente un conjunto de datos determinado en diferentes clases. Una vez que se establece el límite de decisión, los nuevos ejemplos se pueden clasificar en las clases apropiadas con relativa facilidad. En el algoritmo SVM, este límite de decisión se conoce como **hiperplano**. El desafío entonces es trazar este hiperplano con precisión.

En temas previos, se ha visto que la manera más sencilla de encontrar la función que empareja datos de entrada con salida es suponer que la variable objetivo (salida) se puede aproximar como una combinación lineal de las características de los datos de entrada. Para resolver esta aproximación existen los modelos lineales como la regresión lineal (Maulud y Abdulazeez, 2020), que emplea el método de mínimos cuadrados para aproximar la función en el caso de un problema de regresión, o la regresión logística para un problema de clasificación (Master's in Data Science, s.f.). Los modelos lineales muestran un gran rendimiento en aquellos problemas donde subyace una dependencia lineal entre las características de entrada y la salida, es por eso por lo que son modelos muy extendidos y estudiados. No obstante, los modelos lineales tienen un poder de predicción limitado, por eso se han realizado

extensas investigaciones sobre modelos no lineales. Sin embargo, existen varios problemas asociados a los modelos no lineales:

- **Interpretabilidad.** Los modelos no lineales tienden a ser más difíciles de interpretar en comparación con los modelos lineales debido a su complejidad inherente y la falta de una relación lineal clara entre las características y la variable de respuesta (Molnar, 2020).

- **Sobreajuste.** Uno de los principales problemas con los modelos no lineales es el sobreajuste, donde el modelo se ajusta demasiado a los datos de entrenamiento y no generaliza bien a nuevos datos (Hastie *et al.*, 2009).

- **Disponibilidad de datos.** Los modelos no lineales suelen requerir más datos de entrenamiento que los modelos lineales para obtener un rendimiento óptimo debido a su mayor complejidad y capacidad de adaptación (Bishop, 2006).

- **Coste computacional.** Son computacionalmente difíciles de resolver. El entrenamiento de modelos no lineales puede ser computacionalmente intensivo y requerir más tiempo de procesamiento en comparación con los modelos lineales, especialmente en conjuntos de datos grandes o con una alta dimensionalidad (Goodfellow *et al.*, 2016).

Estas limitaciones han hecho que los modelos lineales ganen en popularidad debido a su simplicidad tanto teórica como computacional. También se ha descubierto que, con las características apropiadas, el poder de predicción de los modelos lineales puede ser tan bueno como el de los modelos no lineales (Bishop, 2006).

SVM se utiliza ampliamente para resolver problemas de reconocimiento de patrones, clasificación de imágenes, categorización de texto y más. SVM es particularmente eficaz en espacios de alta dimensión y cuando los datos no son separables linealmente.

Ideas clave

Para comprender SVM, el presente tema está organizado de la siguiente manera: en primer lugar, vamos a ver los conceptos geométricos necesarios para entender el funcionamiento y la base que hay por detrás de las máquinas de vector de soporte. En segundo lugar, veremos los problemas existentes al separar por hiperplanos. A continuación, veremos cómo solucionar el problema de separar por hiperplanos. Finalmente, las técnicas conocidas como *kernel trick*.

Los objetivos que se pretenden conseguir son:

- Comprender el concepto básico de SVM.
- Conocer los fundamentos matemáticos detrás de SVM.
- Distinguir entre SVM lineales y no lineales.
- Comprender la función y los tipos de funciones del kernel utilizadas en SVM.

7.2. Conceptos básicos de SVM

SVM fue propuesto por el científico de origen ruso Vladimir Vapnik y su equipo en el año 1963. La gran contribución de Vapnik al campo del aprendizaje automático ha consistido precisamente en proponer uno de los primeros modelos de estimación/predicción que **no está basado en ningún modelo probabilístico**, lo cual es un gran cambio de mentalidad. Puesto que todos los modelos que se usaban hasta ese momento estaban basados en la teoría estadística y de probabilidad.

Sin embargo, SVM se puede considerar el primer modelo que utiliza un enfoque completamente distinto, que está **basado en un modelado geométrico** y que **se resuelve mediante un problema de optimización con restricciones**.

El objetivo de SVM es buscar un plano que separe las clases en un espacio de características (*feature space*). Por ***feature space*** se entiende un nuevo espacio de dimensiones diferente (por lo general, con un mayor número de dimensiones) al espacio original.

Usando el ejemplo sencillo de un clasificador binario, la idea detrás de SVM es encontrar el hiperplano que mejor separe las dos clases, maximizando el margen entre los puntos más cercanos de cada clase (llamados **vectores de soporte**). Este hiperplano será el margen de decisión usado para posteriormente clasificar al resto de los elementos.

La Figura 1 muestra una representación visual de un modelo SVM aplicado a un problema de clasificación binaria. En ella, se observan dos clases claramente separadas por un hiperplano de separación, que representa el límite de decisión del modelo SVM. Este hiperplano está posicionado de manera óptima para maximizar el margen entre las dos clases.

En la figura se muestran, además, los **vectores de soporte**, que son los puntos de datos más cercanos al hiperplano de separación. Estos vectores de soporte son cruciales para la definición del margen máximo y, por lo tanto, para la capacidad del modelo de generalizar correctamente a nuevos datos.

El margen máximo se representa como la distancia perpendicular desde el hiperplano de separación a los puntos más cercanos de ambas clases. Este margen actúa como una «zona segura» que ayuda a prevenir el sobreajuste al separar claramente las clases y proporcionar una frontera de decisión bien definida.

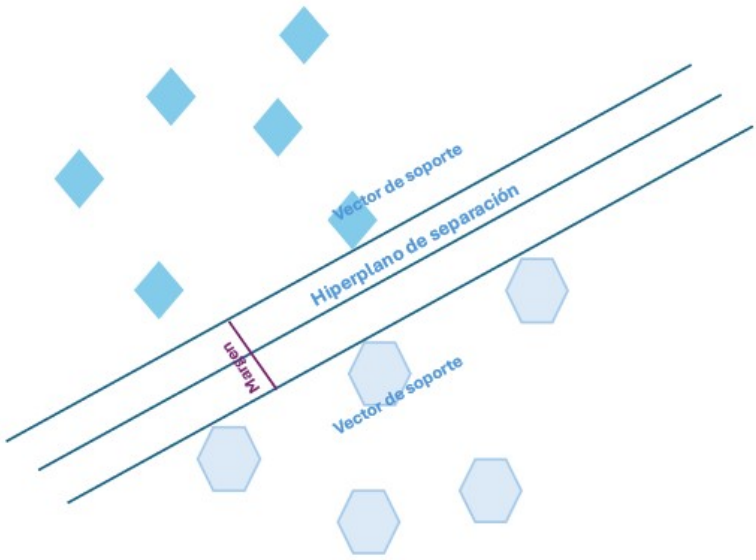


Figura 1. Representación de un modelo SVM para clasificación binaria. Fuente: elaboración propia.

En la figura se presenta el hiperplano de separación, los vectores de soporte y el margen máximo.

Debido a que, por lo general, no es posible obtener este objetivo de buscar un plano que separe las clases, en las SVM se proponen las siguientes modificaciones:

- Relajar la definición de «separar».
- Mejorar y enriquecer el *feature space* para que la separación sea posible.

Antes de entrar en detalle en el funcionamiento de las máquinas de vector de soporte, es necesario definir los conceptos de **hiperplanos** y **vectores**.

Hiperplano

En geometría, un hiperplano es una generalización n-dimensional de un plano, un subespacio con una dimensión menos (n-1) que su espacio origen. En el espacio unidimensional, un hiperplano es un punto; en el espacio bidimensional es una línea; en el espacio tridimensional es un plano ordinario, y en espacios de cuatro o más dimensiones se le llama hiperplano.

> Por ejemplo, digamos que queremos llevar a cabo un problema de clasificación y queremos saber si un producto se compra o no (una clasificación binaria) y si hay solo una característica (por ejemplo, Género) disponible como característica en el conjunto de datos. Entonces, está en un espacio unidimensional y la representación del subespacio (el límite de separación/decisión) será (n-1=0) un espacio de 0 dimensiones (si puedo llamarlo así), representado con solo un punto que muestra la separación de clases (compradas o no). Si hay dos características (Edad y Género), es un espacio bidimensional (2D), con Edad y Género en los ejes X e Y, el límite de decisión se representará como una línea simple. De manera similar, si las características son tres (Edad, Género, Ingresos), el límite de decisión será un plano bidimensional en un espacio tridimensional (n-1). Además, si tenemos puntos de datos espaciales de cuatro o más dimensiones, entonces se llama hiperplano con (dimensión n-1). Es

importante tener en cuenta que la cantidad de características para un problema de aprendizaje automático determinado se puede seleccionar mediante una técnica llamada «selección de características» (Cai, Luo, Wang y Yang, 2018), ya que no todas las características son necesariamente útiles, algunas pueden ser redundantes y crear ruido innecesario en los datos.

El hiperplano es simplemente un concepto que separa un espacio de n dimensiones en dos grupos o mitades. En términos de aprendizaje automático, es una forma de límite de decisión que algoritmos como SVM utilizan para clasificar o separar puntos de datos. Tiene dos partes, el hiperplano del lado negativo y el hiperplano de la parte positiva, donde los puntos o instancias de datos pueden encontrarse en cualquiera de las partes, lo que da lugar al grupo o clase al que pertenecen.

La ecuación general del hiperplano para un espacio de n dimensiones es:

$$\beta x_0 + \beta_1 x_1 + \cdots + \beta_n x_n = 0$$

donde el vector:

$$\beta = (\beta_0, \beta_1, \ldots, \beta_n)$$

Se llama **vector normal**, y β es un vector unitario en el cual la suma de los cuadrados es 1. Este vector apunta a una dirección ortogonal a la superficie del hiperplano.

Si proyectamos cada uno de los puntos sobre el vector normal, los puntos que caen sobre el hiperplano tienen valor 0 al proyectarse. Los puntos por encima del hiperplano tienen un valor positivo, los puntos por debajo del hiperplano, un valor negativo; y, además, estos valores son mayores cuanto más lejanos están del hiperplano. Es decir, el valor que se obtiene al proyectar los puntos sobre el vector

normal es proporcional a la distancia de los puntos al hiperplano. Esta característica geométrica hace posible el uso de las máquinas de vector de soporte para buscar los patrones necesarios.

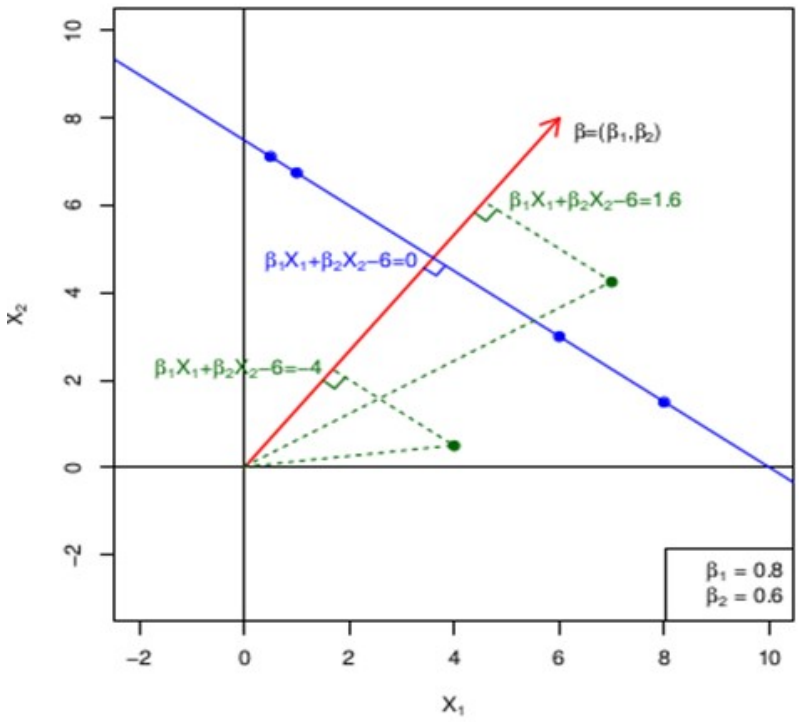


Figura 2. Representación de un hiperplano, un vector normal y la distancia de los puntos proyectados en el vector normal. Fuente: James, Witten, Hastie y Tibshirani, 2013.

La cuestión es que el **hiperplano** busca separar dos conjuntos de puntos en dos regiones distintas. Por ejemplo, en un problema de clasificación el hiperplano puede separar aquellos puntos que corresponden a personas con un tumor determinado de aquellas personas sanas.

Sin embargo, para un conjunto de puntos determinados existen múltiples hiperplanos posibles que nos separan los puntos en dos regiones.

El objetivo de las máquinas de vector de soporte es buscar un hiperplano que separe las clases en un *feature space*. Por *feature space* se entiende un nuevo espacio de

dimensiones diferente al espacio original.

Margen

El margen es la distancia entre el límite de decisión (hiperplano) y los puntos de datos más cercanos de cada clase. El objetivo de SVM es maximizar este margen minimizando los errores de clasificación. Un margen mayor indica un mayor grado de confianza en la clasificación, ya que significa que hay un espacio mayor entre el límite de decisión y los puntos de datos más cercanos de cada clase. El margen es una medida de qué tan bien separadas están las clases en el espacio de características. Las SVM están diseñadas para encontrar el hiperplano que maximiza este margen, razón por la cual a veces se las denomina **clasificadores de margen máximo**.

Vectores de soporte

Son los puntos de datos que se encuentran más cerca del límite de decisión (hiperplano) en SVM. Estos puntos de datos son importantes porque determinan la posición y orientación del hiperplano y, por lo tanto, tienen un impacto significativo en la precisión de la clasificación del SVM. De hecho, SVM recibe el nombre de estos vectores de soporte porque «soportan» o definen el límite de decisión. Los vectores de soporte se utilizan para calcular el margen, que es la distancia entre el hiperplano y los puntos de datos más cercanos de cada clase. El objetivo de las SVM es maximizar este margen y minimizar los errores de clasificación.

7.3. Separando por hiperplanos

Dada una serie de puntos en un espacio geométrico que se desea clasificar, se puede establecer que aquellos puntos que al proyectar sobre el vector normal a un hiperplano determinado sean > 0 y correspondan a una clase, y aquellos que sean < 0, a otra clase.

Es decir, de forma matemática:

Si $f(x) = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \ldots + \beta_p X_P$, entonces:

$f(x) > 0$ establece los puntos en un lado del hiperplano.

$f(x) < 0$ en el otro.

Sin embargo, lo habitual es encontrarse en la situación en la cual existen múltiples hiperplanos posibles, como lo que se muestra en la Figura 3, y por tanto es necesario decidir cuál de ellos establecer.

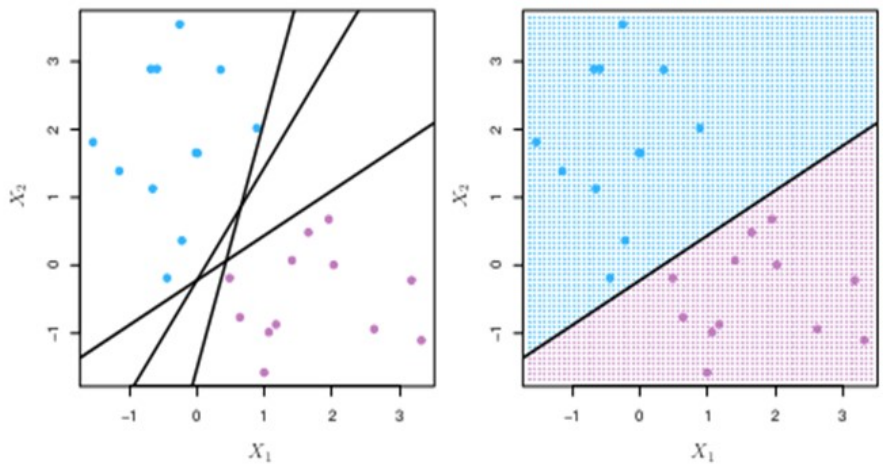


Figura 3. Ejemplo de tres hiperplanos posibles para separar dos conjuntos de clase (izquierda). En el caso de elegir un hiperplano, los puntos se clasifican en función de cada una de las regiones sombreadas de rojo y azul (derecha). Fuente: James *et al.*, 2013.

7.4. Clasificador de margen máximo

En el caso de un problema de clasificación binaria, de todos los hiperplanos posibles es necesario buscar aquel que nos proporciona la mayor diferencia entre las dos clases, lo cual se traduce en la mayor distancia entre los puntos que pertenecen a una clase y a otra. La hipótesis que hay detrás de esto es que suponemos que este hiperplano será el que tendrá una mayor distancia en el conjunto de test y en las predicciones futuras.

Esta situación se puede modelar como un problema de optimización con restricciones, donde es necesario maximizar el margen y, matemáticamente, se define:

$$maximize\, M$$

$$\beta_0,\beta_1,\ldots,\beta_p$$

$$subject \sum_{j=1}^{p} \beta_j^2 = 1$$

$$y_i\,(\beta_0 + \beta_1 x_{i1} + \ldots + \beta_p X_{ip}) \geq M$$

$$for\ \ all\ i = 1,\ldots,N$$

En la siguiente imagen se muestra de forma gráfica el concepto donde la línea continua resaltada es el *maximal-margin classifier*, y se muestran dos bandas con líneas discontinuas que contienen la distancia del hiperplano a los primeros puntos de cada una de las clases. El objetivo es maximizar la distancia de estas dos bandas.

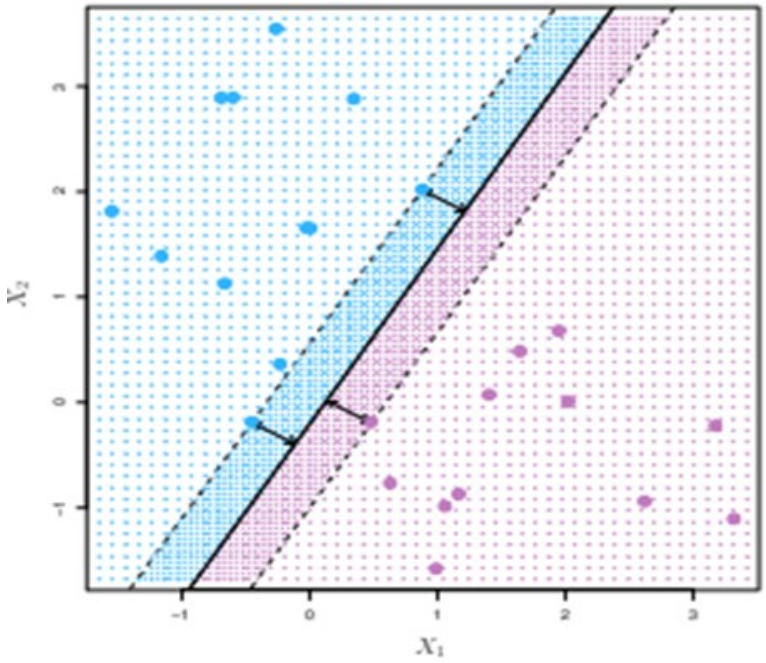


Figura 4. Ejemplo del hiperplano con una separación óptima entre dos conjuntos de puntos de clases diferentes. Fuente: James *et al.*, 2013.

El principal problema con esta separación óptima del hiperplano es que **los datos habitualmente no son linealmente separables con una recta** en el caso de un espacio de dos dimensiones, como se observa en la siguiente gráfica, donde es imposible separar los puntos de una clase de los de otra.

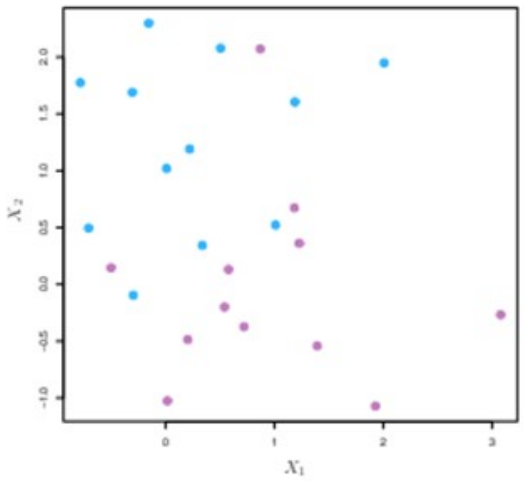


Figura 5. Ejemplo de un espacio de dos dimensiones donde los puntos no son linealmente separables.

Fuente: James *et al.*, 2013.

Esta situación produce una solución pobre para el clasificador *maximal-margin*. Por tanto, el clasificador vector de soporte maximiza un *soft margin*.

Support Vector Regression

SVR (Support Vector Regression) es una variante de SVM que se utiliza para resolver problemas de regresión en lugar de clasificación. Mientras que las SVM tradicionales se centran en encontrar un hiperplano de separación óptimo entre clases, SVR se enfoca en encontrar una función de regresión que se ajuste a los datos de manera óptima a la vez que mantiene la cantidad de error dentro de un cierto límite predefinido.

La idea principal detrás de SVR es encontrar una función que capture la relación entre las variables independientes (características) y la variable dependiente (la que se quiere predecir), de manera que minimice el error.

El proceso de entrenamiento de SVR implica encontrar un hiperplano de regresión óptimo que pase lo más cerca posible de la mayoría de los puntos de datos, dentro del margen de tolerancia especificada. Los puntos de datos que se encuentran dentro del margen de tolerancia serán los vectores de soporte.

7.5. Soft margin

Se puede decir que las instancias de datos son linealmente separables si se puede dibujar fácilmente un límite o hiperplano de separación que muestre distintivamente los diferentes grupos de clases. En la Figura 6 se puede ver un ejemplo de datos linealmente separables. Los puntos de datos linealmente separables requieren, ante todo, clasificadores lineales de aprendizaje automático.

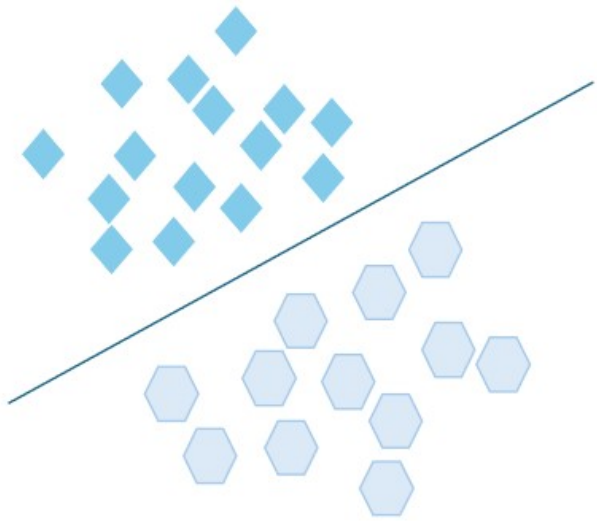


Figura 6. Datos linealmente separables. Fuente: elaboración propia.

Los datos no linealmente separables son lo opuesto a los puntos de datos linealmente separables. En la Figura 7 se puede observar que no importa cómo se intente dibujar una línea recta, algunos puntos de datos se clasificarán erróneamente de una forma u otra. SVM tiene una forma especial de clasificar este tipo de datos; utiliza funciones del kernel para representar estos puntos de datos en un espacio de dimensiones superiores y luego encuentra el hiperplano de separación óptimo.

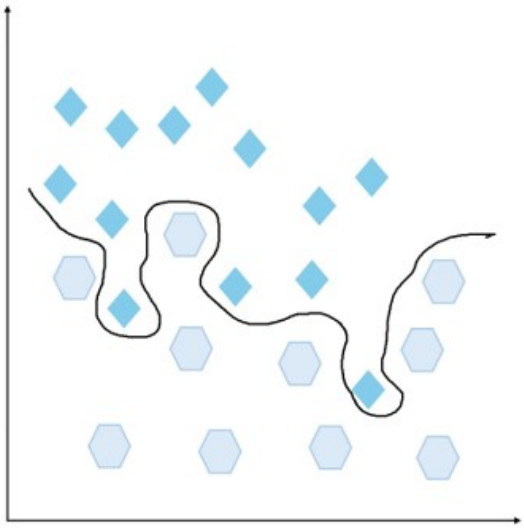


Figura 7. Datos que no son linealmente separables. Fuente: elaboración propia.

La primera solución propuesta para abordar el problema anterior en el cual los puntos no son linealmente separables es maximizar un *soft margin*. En otras palabras, el clasificador permite realizar algunos errores en la clasificación. El concepto de *soft margin* (o «margen suave») es una extensión SVM que permite cierta flexibilidad en la separación de clases. En contraste con el enfoque de *hard margin* («margen duro»), donde se requiere que todas las muestras de entrenamiento estén correctamente clasificadas y se encuentren fuera del margen, el margen suave permite que algunas muestras se encuentren dentro del margen o incluso en el lado incorrecto del hiperplano de separación.

De forma gráfica, en la siguiente figura se observa:

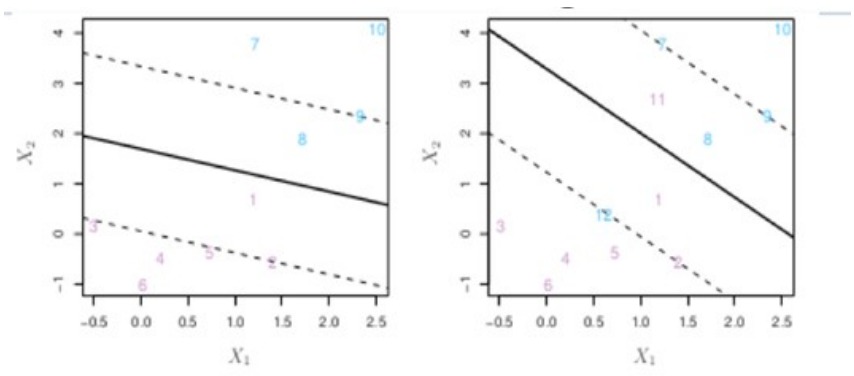


Figura 8. Ajuste de un clasificador de vector de soporte a pocos datos. Fuente: James et al., 2013.

El hiperplano se muestra con una línea sólida y los márgenes con una punteada. La imagen de la derecha muestra el efecto cuando aparecen dos nuevas observaciones (11 y 12), las cuales están en el lado incorrecto del hiperplano y de los márgenes.

Matemáticamente el problema se define de la siguiente forma:

$$maximize\, M$$

$$\beta_0,\beta_1,\ldots,\beta_p,\epsilon_1,\ldots,\epsilon_n$$

$$subject \sum_{j=1}^{p} \beta_j^2 = 1$$

$$y_i\left(\beta_0+\beta_1 x_{i1}+\ldots+\beta_p X_{ip}\right) \geq M\left(1-\epsilon_i\right)$$

$$\epsilon_i \geq 0, \sum_{i=1}^{n} \epsilon_i \leq C$$

Modificando el parámetro C de la ecuación anterior, el cual se conoce **como función de coste**, se puede hacer el margen más grande o pequeño. En concreto, un valor de C más grande hace el margen más pequeño, y a la inversa un valor C más pequeño, que hace el margen más grande.

En la siguiente figura se muestra un hiperplano y los márgenes correspondientes en función de diferente valor de C.

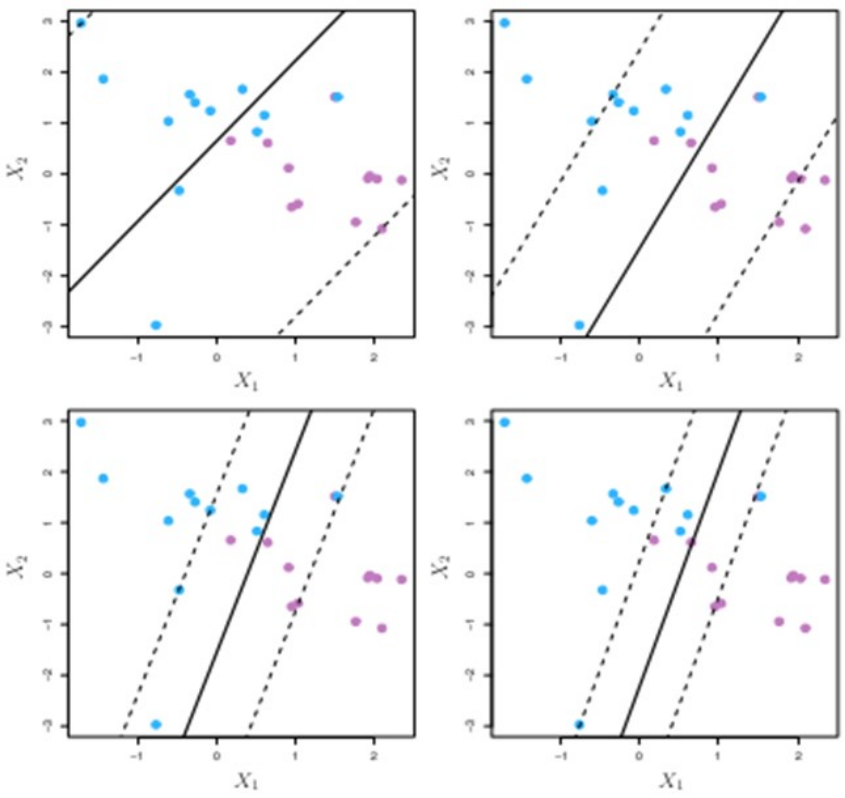


Figura 9. Diferentes fronteras de decisión en los márgenes de una *soft margin* en función de diferentes valores de C, y utilizando los mismos puntos y el mismo hiperplano. Fuente: James *et al.*, 2013.

Ten en cuenta que el margen más grande es el de la gráfica superior izquierda y el más pequeño el de la inferior derecha.

El objetivo sigue siendo maximizar el margen, pero ahora se permite un cierto grado de violación del margen para adaptarse a los datos de entrenamiento que no se pueden separar perfectamente con un hiperplano.

Problema con las fronteras lineales

Muchas veces las fronteras lineales siguen sin funcionar, independientemente del valor de coste C que se utilice. Por ejemplo, en la siguiente gráfica se muestra un problema donde la separación necesaria entre los puntos de una clase y la otra es no lineal.

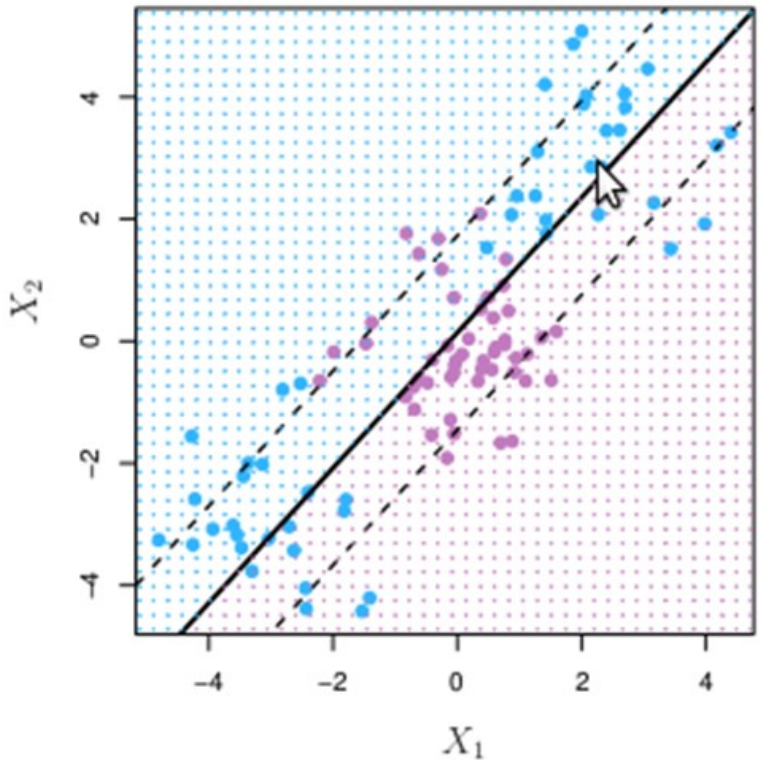


Figura 10. Ejemplo donde no es posible una separación lineal entre dos clases. Fuente: James *et al.*, 2013.

Para solucionar este problema donde no es posible realizar una separación entre dos clases, una opción es **incrementar el espacio de las variables por medio de transformaciones**; es decir, pasar de un espacio **n-dimensional** a un espacio de D dimensiones D > n. Una vez realizada esta transformación, se debería ajustar un clasificador de vector de soporte en este nuevo espacio. El objetivo es **obtener fronteras de decisión no lineales sobre el espacio original**.

Por ejemplo, si tenemos datos de entrada en dos dimensiones (X_1, X_2) es posible utilizar el siguiente espacio de 6 dimensiones: (X_1, X_2, X_1^2, X_2^2, X_1 X_2) siendo, por tanto, la **frontera de decisión:**

$$\beta_0 + \beta_1 X_1 + \beta_2 X_2 + \beta_3 X_1^2 + \beta_4 X_2^2 + B_5 X_1 X_2 = 0$$

Esta transformación conlleva fronteras de decisión no lineales en el espacio original.

Polinomios cúbicos

Utilizando una transformación en forma de polinomios cúbicos se puede pasar de 2 a 9 variables. De esta forma, el clasificador en el nuevo espacio soluciona el problema de buscar los patrones en el espacio con menor dimensiones.

En la Figura 11 se muestra el resultado de las fronteras de separación obtenidas por medio de una transformación en polinomios cúbicos, resultado del cálculo de la siguiente ecuación:

$$B_o + B_1 X_1 + B_2 X_2 + B_3 X_1^2 + B_4 X_2^2 + B_5 X_1 X_2 + B_6 X_1^2 + B_7 X_2^3 + B_8 X_1 X_2^2 + B_9 X_1^2 X_2 = 0$$

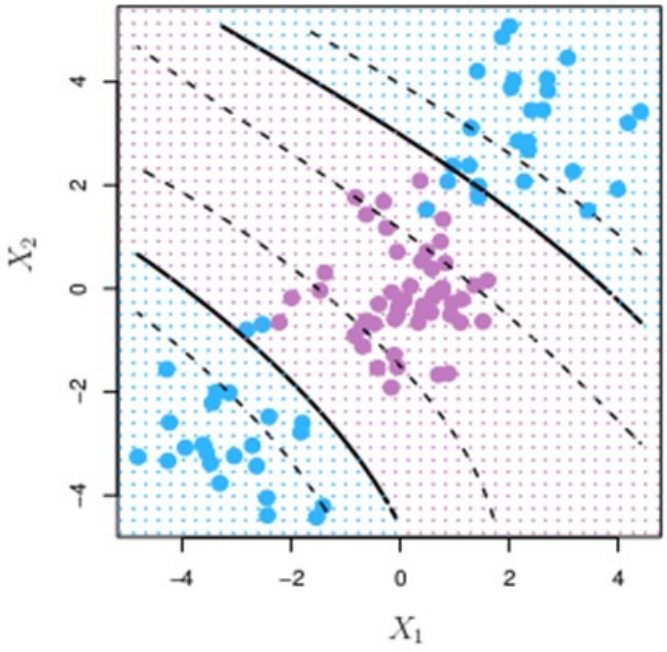


Figura 11. Ejemplo de las fronteras de decisión no lineales obtenidas por medio de una transformación obtenida con polinomios cúbicos. Fuente: James *et al.*, 2013.

7.6. El truco del kernel

Las expansiones de las variables con polinomios, especialmente aquellas con grandes dimensiones, son computacionalmente costosas. Existe una solución más elegante y controlada de introducir no linealidad que se utiliza en las máquinas vector de soporte por medio del uso de *kernels*. El *kernel* de un operador A denotado por *ker* es el conjunto de todos los vectores cuya imagen sea el vector nulo:

$$ker\, A = \{\vec{v} \in V : A\vec{v} = 0\}$$

Los *kernels* se apoyan en concepto del producto vectorial de los vectores de soporte. Se trata de funciones que reciben dos vectores como parámetros. Uno de los *kernels* más utilizados es el de base radial que asume que el *feature space* es de altas dimensiones y se define con la siguiente función matemática:

$$K\left(x_i, x_i'\right) = \exp\left(-\gamma \sum_{j-1}^{p} \left(x_{yj} - x_{i'j}\right)^2\right.$$

$$y_i \left(\beta_0 + \beta_1 x_{i1} + \ldots + \beta_p X_{ip}\right) \geq M$$

El uso de *kernels* permite obtener fronteras de decisión no lineales por medio de transformaciones matemáticas sin necesidad de tener que realizar transformaciones con polinomios. En la siguiente figura se muestran las fronteras de decisión no lineales obtenidas por medio del uso de un *kernel* de base radial.

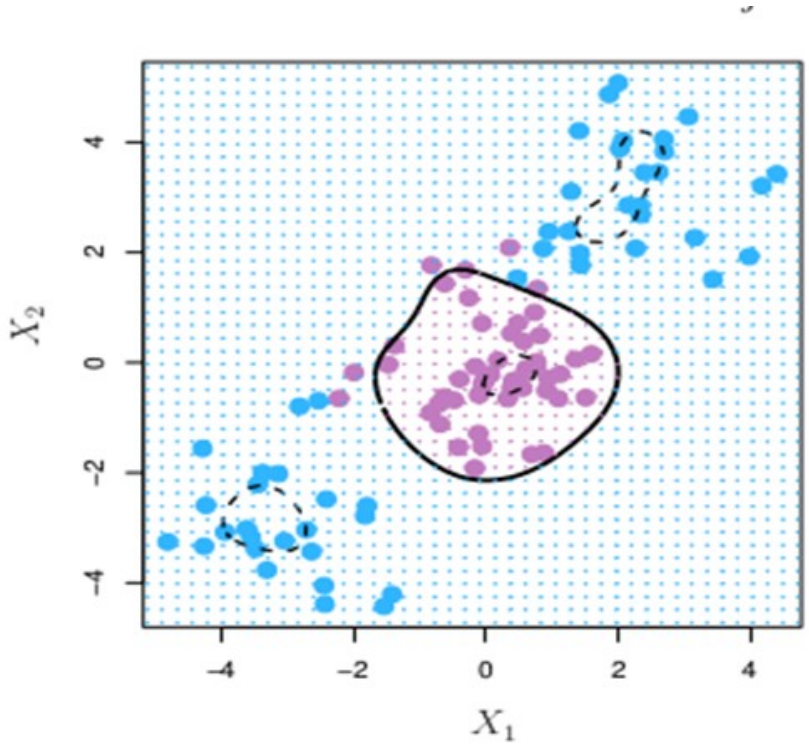


Figura 12. Ejemplo de las fronteras de decisión no lineales obtenidas por medio del uso de un kernel de base radial. Fuente: James *et al.*, 2013.

Los *kernels* o funciones de *kernel* son métodos que los clasificadores lineales como SVM utilizan para clasificar puntos de datos no linealmente separables. Esto se hace representando los puntos de datos en un espacio de mayor dimensión que su original. Por ejemplo, unos datos 1D se pueden representar como datos 2D en el espacio, unos datos 2D se pueden representar como datos 3D, etc. SVM vuelve a representar inteligentemente puntos de datos no lineales utilizando cualquiera de las funciones del *kernel*, de manera que parezca que los datos se han transformado, y luego encuentra el hiperplano de separación óptimo. Sin embargo, en realidad los datos siguen siendo los mismos, en realidad no se han transformado. Por eso se le llama *kernel trick*.

Ideas clave

El truco del *kernel* ofrece una forma de calcular relaciones entre puntos de datos utilizando funciones de *kernel* y de representar los datos de una manera más eficiente con menos cálculo. Los modelos que utilizan esta técnica se denominan **modelos kernelizados**.

Hay varias funciones que utiliza SVM para realizar esta tarea. Algunos de los más comunes son:

- **El kernel lineal:** se utiliza para datos lineales. Esto simplemente representa los puntos de datos mediante una relación lineal.

- **Función kernel polinómica:** transforma puntos de datos utilizando el producto escalar y transformando los datos a una dimensión n. n podría ser cualquier valor de 2, 3, etc., es decir, la transformación será un producto al cuadrado o superior. Por lo tanto, representa datos en un espacio de dimensiones superiores utilizando los nuevos puntos transformados.

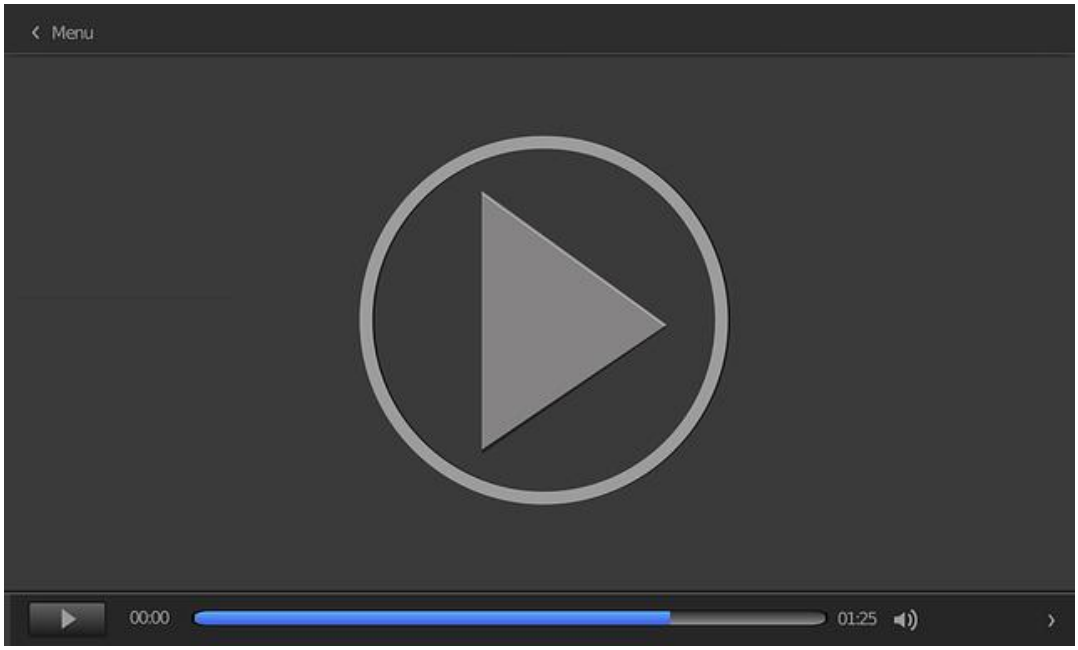
- **La función de base radial (RBF):** esta función se comporta como un modelo de vecino más cercano ponderado. Transforma los datos representándolos en infinitas dimensiones y luego utiliza el vecino más cercano ponderado (observación con mayor influencia en el nuevo punto de datos) para la clasificación. La función radial puede ser Gaussiana o Laplace. Esto depende de un hiperparámetro conocido como gamma. Este es el *kernel* más utilizado.

- **La función sigmoide:** también conocida como función tangente hiperbólica (*tanh*). La función sigmoide mide la similitud entre dos vectores de características utilizando la función *tanh* de la combinación lineal de estos dos vectores de características, más un término de sesgo. Tiene la forma de una curva S y produce valores en el rango (-1, 1). Esta función no se utiliza comúnmente en comparación con los *kernels* lineales o RBF, ya que puede ser menos efectiva en muchos problemas y puede ser más propensa al sobreajuste.

	Formula	Parameters
Linear	$K(x, x_i) = x \cdot x_i$	/
Polynomial	$K(x, x_i) = [\gamma * (x \cdot x_i) + coef]^d$	γ, *coef*, *d*
RBF	$K(x, x_i) = \exp(-\gamma * \|x - x_i\|^2)$	γ.
Sigmoid	$K(x, x_i) = \tanh(\gamma(x \cdot x_i) + coef)$	γ, *coef*

Figura 13. Tabla resumen con la fórmula y los parámetros de las funciones de *kernel* más ampliamente usadas. Fuente: Ren, Hu y Miao, 2016.

Entre el material audiovisual encontrarás el vídeo *Ejemplo en Python de SVM*, donde se muestra un ejemplo de un clasificador de especies de Iris realizado en Python con la librería scikit learn. Con este vídeo se muestra cómo realizar la implementación de un clasificador SVM, con diferentes configuraciones de parámetros.



Accede al vídeo:

https://unir.cloud.panopto.eu/Panopto/Pages/Embed.aspx?id=f374b9e4-f53d-44f4-9d1b-b12200b508ab

7.7. Aplicaciones de SVM

Las máquinas de vectores de soporte se han aplicado con éxito a diversos problemas del mundo real en diferentes dominios. A continuación, se muestran algunas aplicaciones notables de SVM:

- **Clasificación de imágenes:** las SVM se han utilizado ampliamente para el reconocimiento de objetos de imágenes, el reconocimiento de dígitos escritos a mano y el reconocimiento óptico de caracteres (OCR). Se han empleado en sistemas como el filtrado de spam basado en imágenes y en sistemas de detección de rostros utilizados para seguridad, vigilancia e identificación biométrica (Mangasarian, Street y Wolberg, 1995).

- **Clasificación de texto:** las SVM son efectivas para tareas de categorización de texto, como análisis de opiniones, detección de spam y clasificación de temas (Mullen y Collier, 2004).

- **Bioinformática:** las SVM se han aplicado en bioinformática para tareas como la predicción de la estructura de proteínas, el análisis de la expresión genética y la clasificación del ADN (Gao, Ye, Lu y Huang, 2017).

- **Detección de fraude y seguridad:** SVM se emplea para la detección de fraudes en transacciones financieras, identificación de anomalías en redes de computadoras, y clasificación de spam en correos electrónicos (Bhattacharyya, Jha, Tharakunnel, y Westland, 2011).

Las SVM también se han aplicado en otros dominios, como las geociencias, el marketing y la visión por computadora, demostrando su versatilidad y eficacia en diversos dominios problemáticos.

7.8. Cuaderno de ejercicios

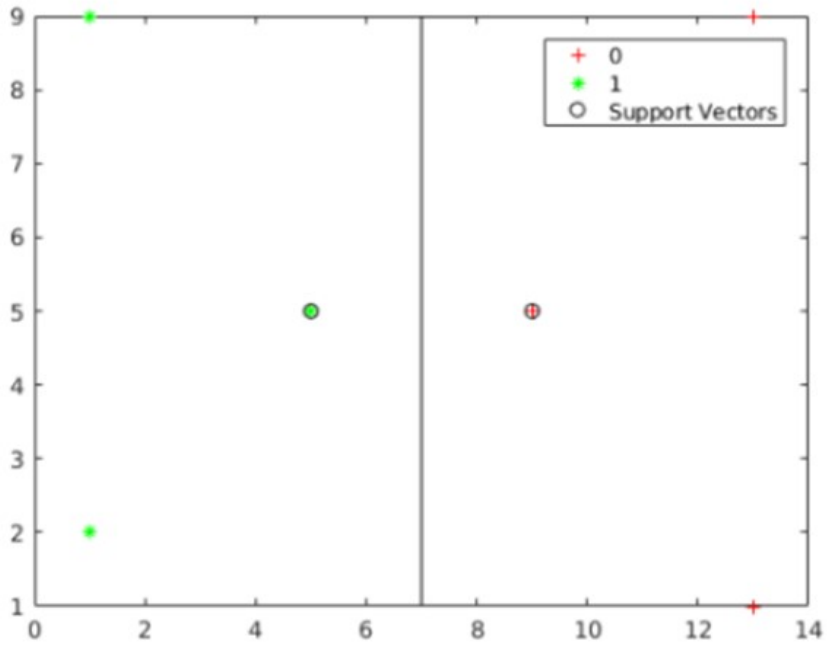
- Dado el siguiente conjunto de datos pertenecientes a dos clases:

Clase -1 [[1,9],[5,5],[1,1]]

Clase +1 [[8,5],[13,1],[13,9]]

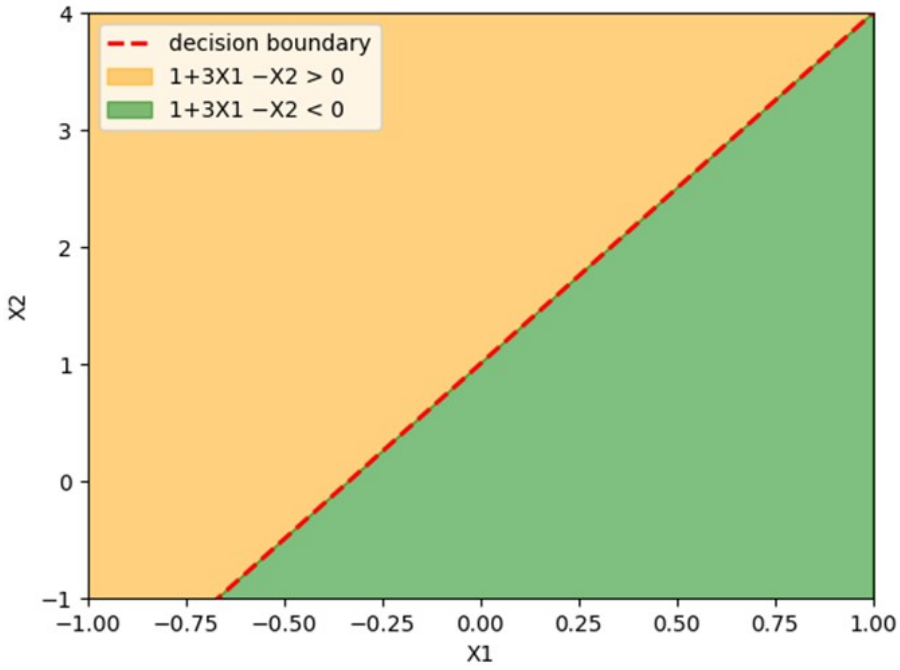
Dibuja los puntos en un plano e identifica los vectores de soporte y la frontera de decisión para un SVM lineal con un margen máximo para este conjunto de datos.

SOLUCIÓN

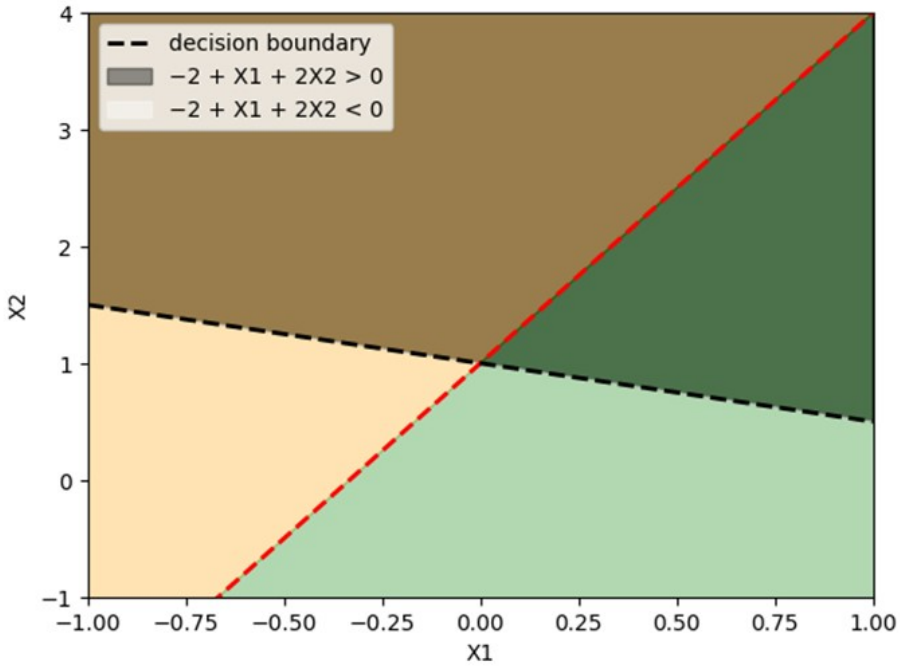


- Dibuja el hiperplano $1 + 3X_1 - X_2 = 0$. Señala el conjunto de puntos para los cuales $1 + 3X_1 - X_2 > 0$, y aquellos para los que $1 + 3X_1 - X_2 < 0$.

SOLUCIÓN



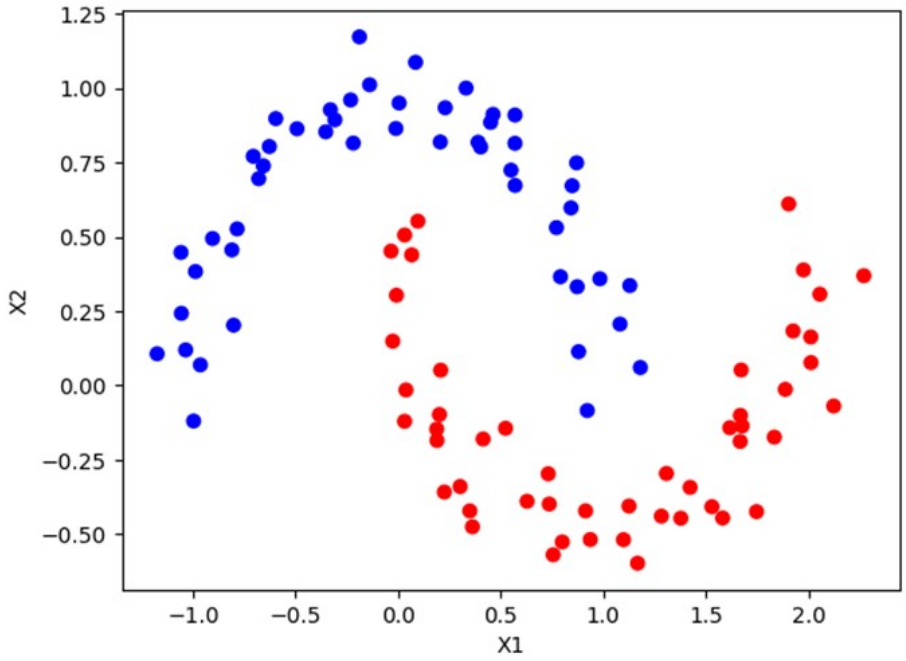
- En el mismo gráfico, dibuja el hiperplano -2 + X1+3X1-X2=0. Indica qué puntos cumplen la condición > 0 y cuáles < 0.



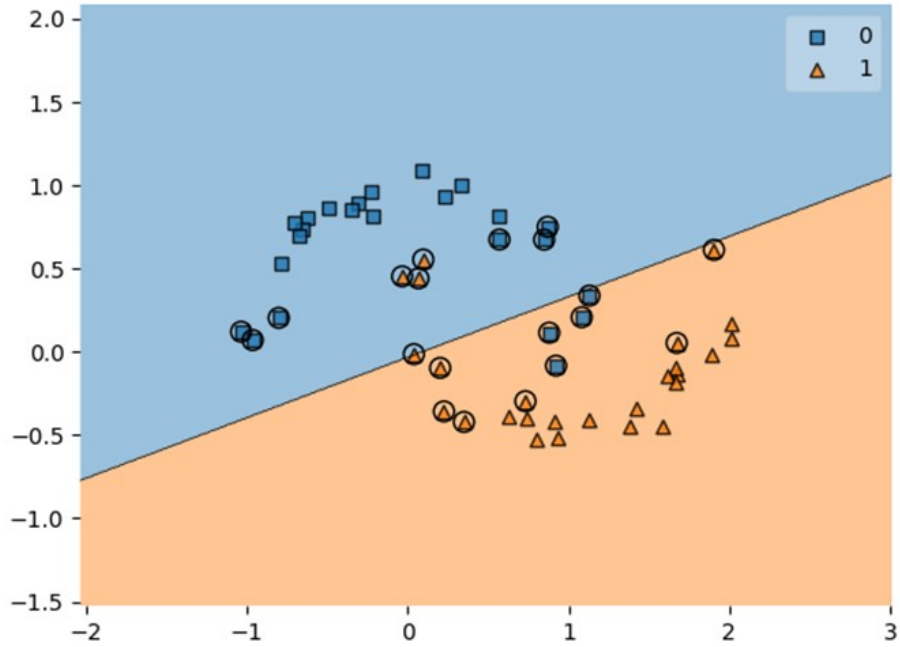
- Generar un conjunto de datos simulado de dos clases con 100 observaciones y dos características en las que exista una separación visible pero no lineal entre las dos clases. Se debe mostrar que, en esta configuración, una máquina de vectores de soporte con un núcleo polinómico (con grado mayor que 1) o un núcleo radial superará a un clasificador de vectores de soporte en los datos de entrenamiento. ¿Qué técnica funciona mejor con los datos de prueba? Se deben mostrar los gráficos que avalen la respuesta y analizar las tasas de error de prueba y entrenamiento para respaldar sus afirmaciones.

SOLUCIÓN

```
# Generate the dataset

# https://scikit-learn.org/stable/modules/generated/sklearn.datasets.make_moons.html

X, y = make_moons(n_samples=100, noise=0.1, random_state=42)



# Plot the dataset

plt.scatter(X[:, 0], X[:, 1], c=y, cmap='bwr')

plt.xlabel('X1')

plt.ylabel('X2')

plt.show()
```




```
X_train, X_test, y_train, y_test = train_test_split(X, y, train_size=0.5,
random_state=5)



# Fit a support vector classifier (kernel=linear)

svc = SVC(kernel='linear')

svc.fit(X_train, y_train)

plot_decision_regions(X_train, y_train, clf=svc,
X_highlight=svc.support_vectors_)
```

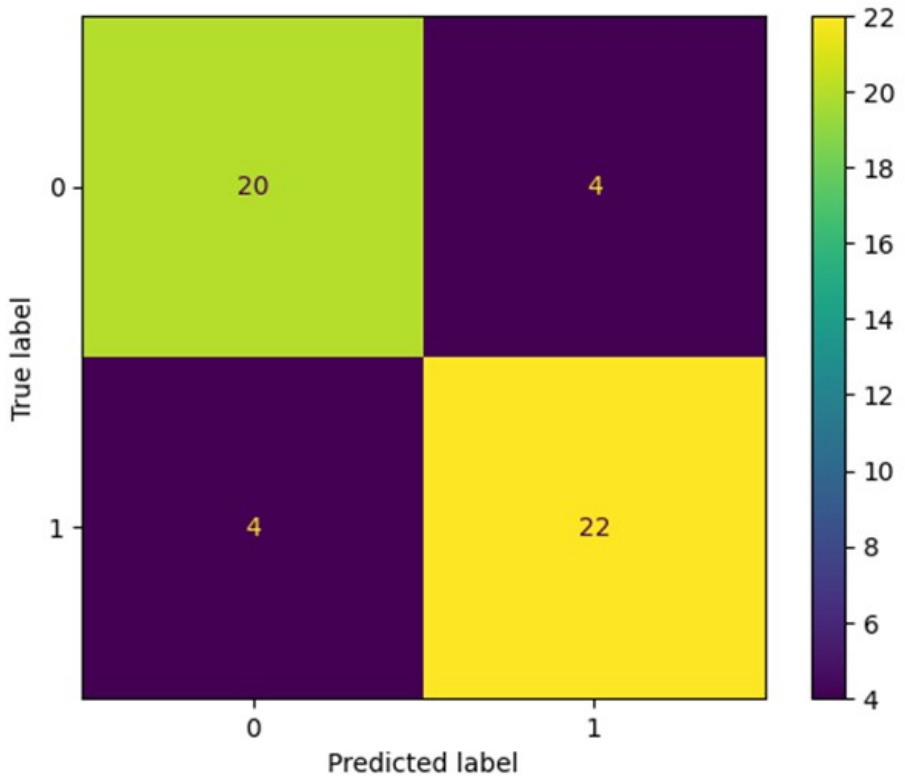



```
# Support vector classifier model summary

print('number of support vectors: ', sum(svc.n_support_))

print('indices of support vectors: ', svc.support_)
```

```
number of support vectors: 20

indices of support vectors: [14 15 16 18 26 27 37 41 44 45 3 12 17 23 31 35
40 43 47 49]
```

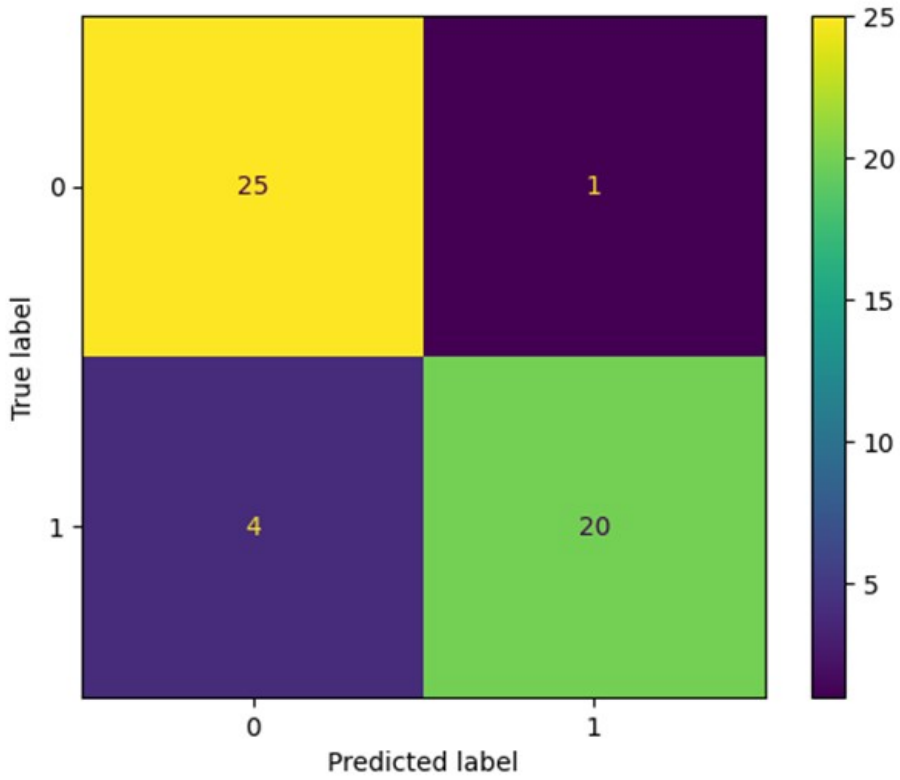
```
# training set performance

y_pred = svc.predict(X_train)

cm = confusion_matrix(y_train, y_pred, labels=svc.classes_)

disp = ConfusionMatrixDisplay(confusion_matrix=cm,
display_labels=svc.classes_)
```

```
disp.plot()
```



SVM ha clasificado erróneamente aproximadamente el 13% del conjunto de entrenamiento.

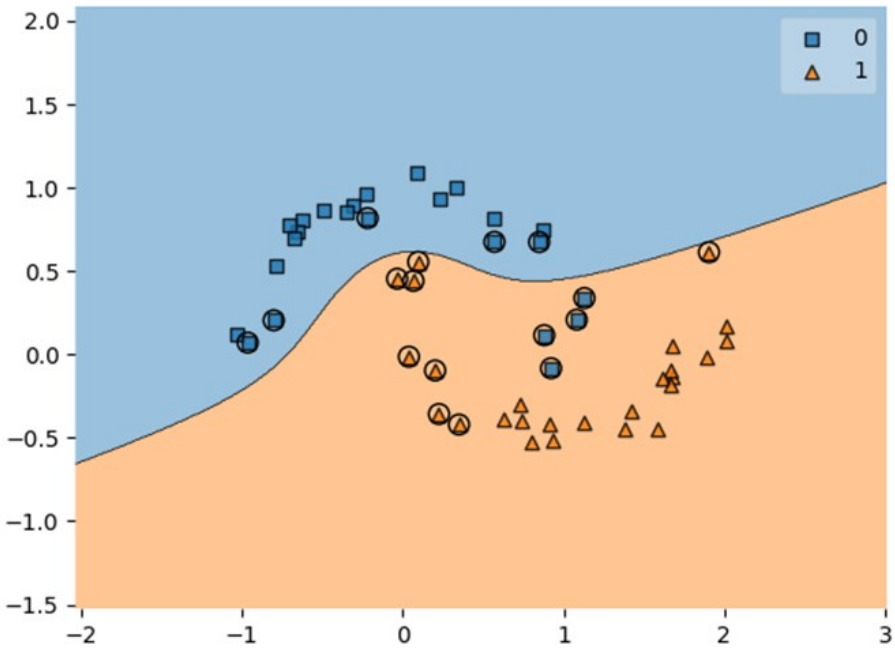
```
# Test set performance

y_pred = svc.predict(X_test)

cm = confusion_matrix(y_test, y_pred, labels=svc.classes_)

disp = ConfusionMatrixDisplay(confusion_matrix=cm,
display_labels=svc.classes_)

disp.plot()
```



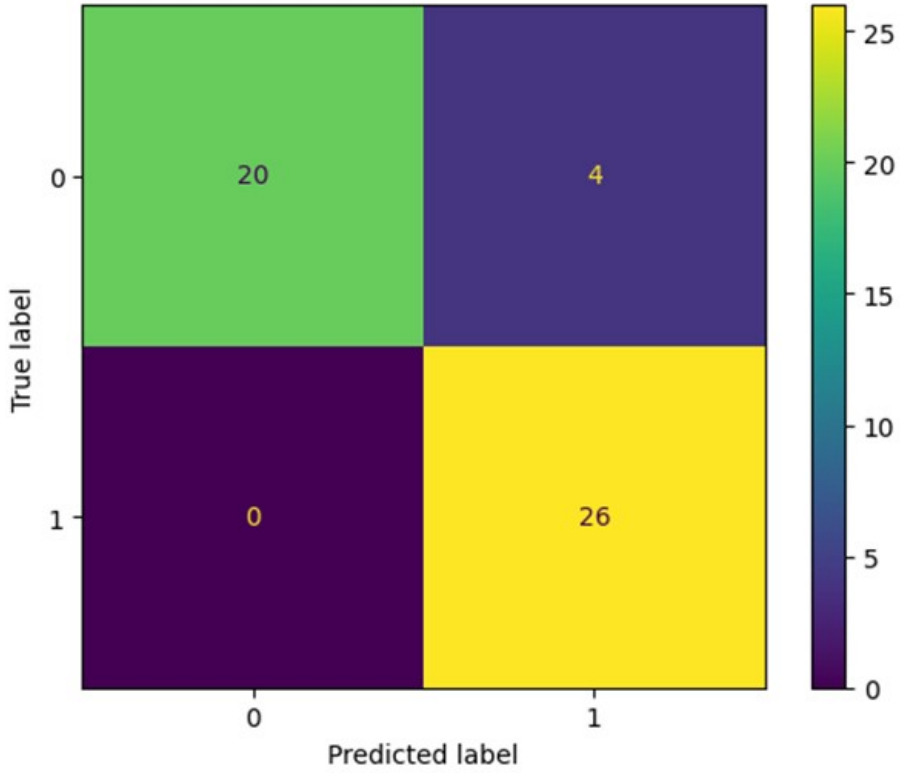
SVM ha clasificado erróneamente aproximadamente el 17% del conjunto de entrenamiento.

```
# Support Vector Machine with Polynomial Kernel

params = {'C': 10**np.linspace(-1, 3, 5),

          'degree' : [2, 3, 4, 5, 6]}

svm = SVC(kernel='poly')

tune = GridSearchCV(svm, params, cv=10)

tune.fit(X, y)

pd.DataFrame(tune.cv_results_).loc[:, ['param_degree', 'param_C',
'mean_test_score', 'std_test_score']]

tune.best_params_
```

```
{'C': 10.0, 'degree': 3}

svm = SVC(kernel='poly', C=10, degree=3)

svm.fit(X_train, y_train)

plot_decision_regions(X_train, y_train, clf=svm,
X_highlight=svm.support_vectors_)
```

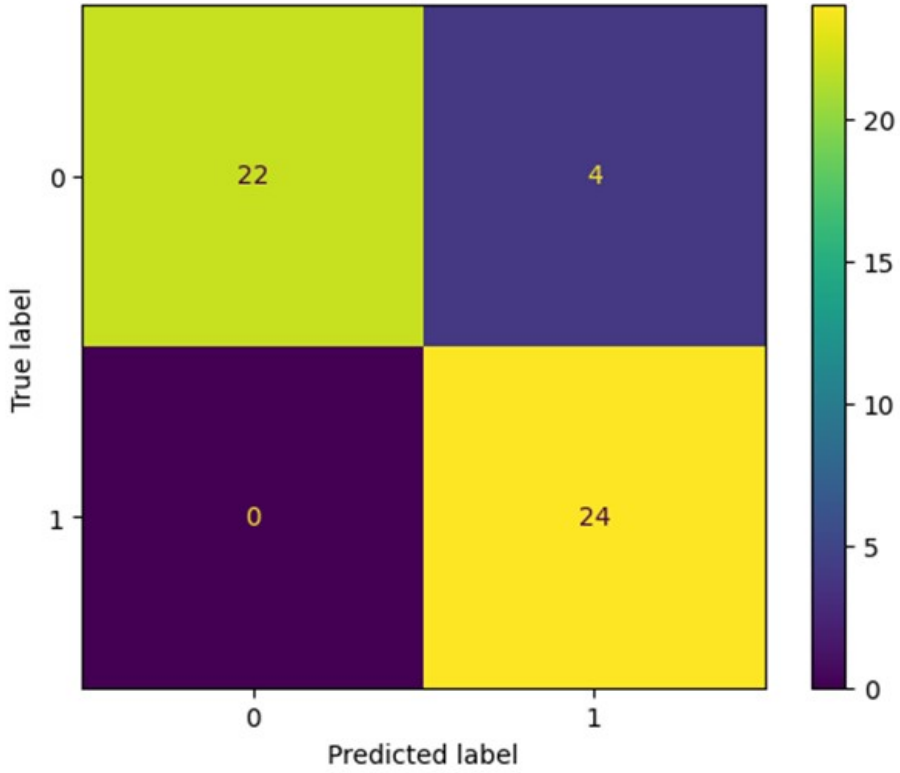



```
# SVM - training set performance (with degree=3)

y_pred = svm.predict(X_train)

cm = confusion_matrix(y_train, y_pred, labels=svm.classes_)

disp = ConfusionMatrixDisplay(confusion_matrix=cm,
display_labels=svm.classes_)

disp.plot()
```



El modelo SVM con grado 3 ha clasificado erróneamente sólo el 5% del conjunto de entrenamiento.

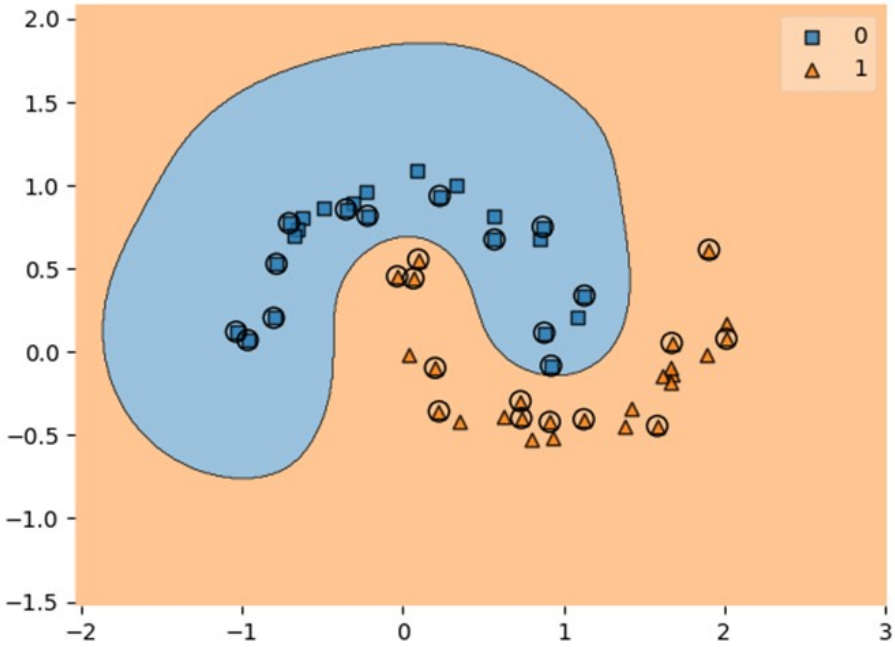
```
# SVM - test set performance (with degree=3)

y_pred = svm.predict(X_test)

cm = confusion_matrix(y_test, y_pred, labels=svm.classes_)

disp = ConfusionMatrixDisplay(confusion_matrix=cm,
display_labels=svm.classes_)

disp.plot()
```



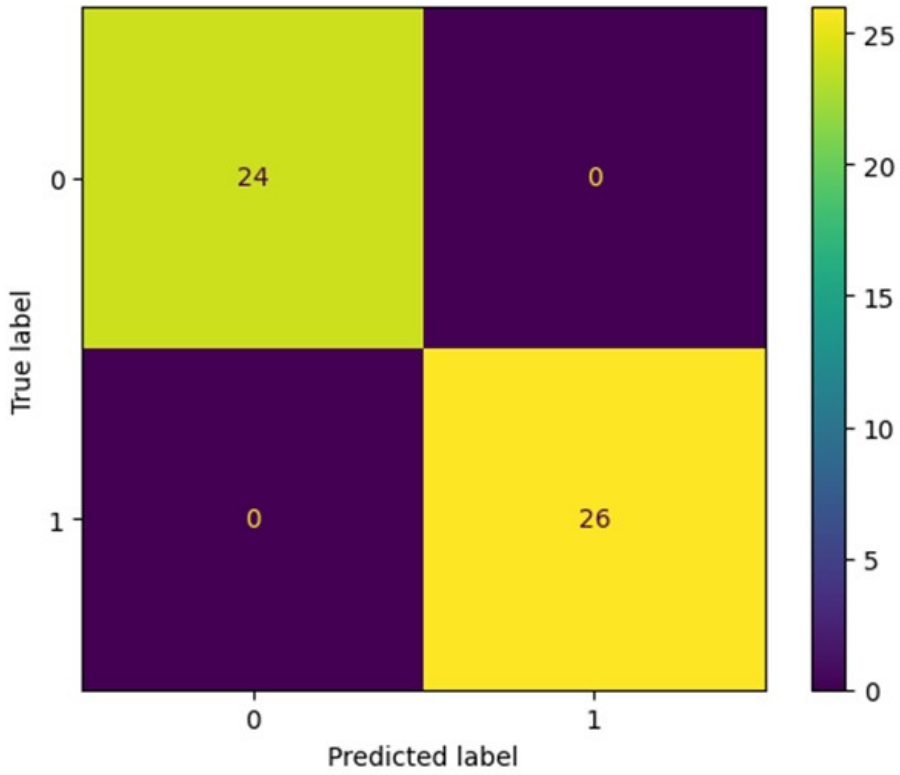
El modelo SVM con un grado de 3 ha clasificado erróneamente sólo el 13% del conjunto de test. Con base en estos resultados, podemos concluir que la SVM con un kernel no lineal (polinomio - cúbico) se desempeña mejor que la SVM con un kernel lineal tanto en el conjunto de entrenamiento como en el de test.

```
# Support Vector Machine with Radial Kernel

params = {'C': 10**np.linspace(-1, 3, 5),

          'gamma': [0.5, 1, 2, 3, 4]}

svm2 = SVC(kernel='rbf')

tune2 = GridSearchCV(svm2, params, cv=10)

tune2.fit(X, y)

pd.DataFrame(tune2.cv_results_).loc[:, ['param_gamma', 'mean_test_score',
```

```
'std_test_score']]

tune2.best_params_

svm2 = SVC(kernel='rbf', C=1, gamma=4)

svm2.fit(X_train, y_train)

plot_decision_regions(X_train, y_train, clf=svm2,
X_highlight=svm2.support_vectors_)
```

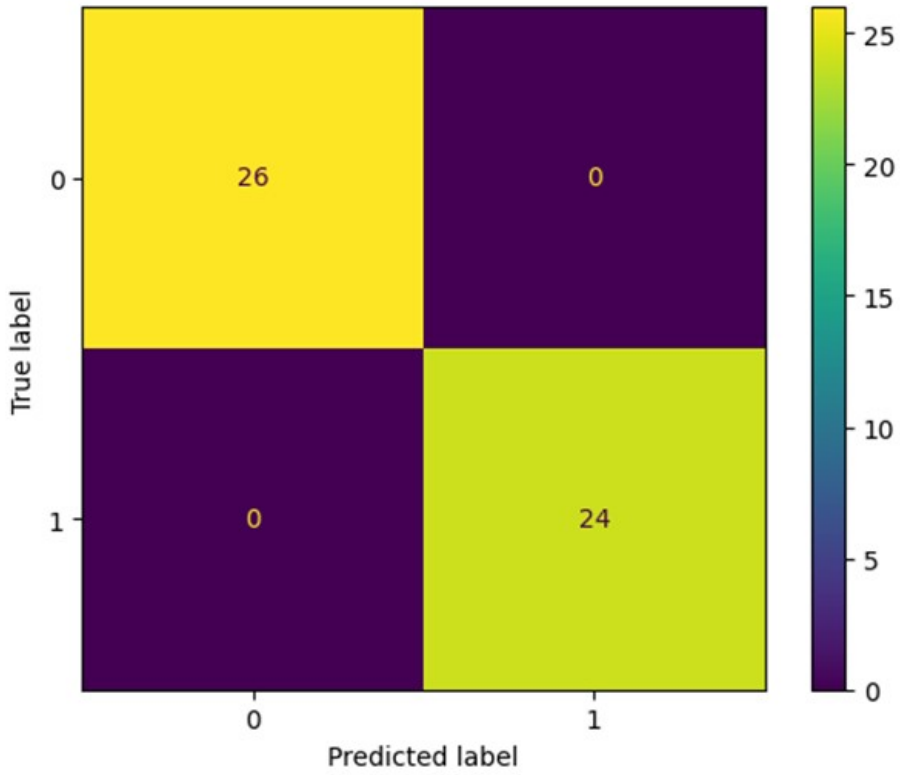



```
# SVM - training set performance (with radial kernel)

y_pred = svm2.predict(X_train)

cm = confusion_matrix(y_train, y_pred, labels=svm2.classes_)

disp = ConfusionMatrixDisplay(confusion_matrix=cm,
display_labels=svm2.classes_)

disp.plot()
```



La SVM con kernel radial ha logrado una clasificación perfecta en el conjunto de entrenamiento, sin observaciones mal clasificadas.

```
# SVM - test set performance (with radial kernel)

y_pred = svm2.predict(X_test)

cm = confusion_matrix(y_test, y_pred, labels=svm2.classes_)

disp = ConfusionMatrixDisplay(confusion_matrix=cm,
display_labels=svm2.classes_)

disp.plot()
```



La SVM con kernel radial también logró una clasificación perfecta en el conjunto de prueba, sin observaciones mal clasificadas.

Con base en los resultados obtenidos, se puede concluir que tanto el SVM con un kernel polinomial cúbico como el SVM con un kernel radial superan al modelo SVC con un kernel lineal en estos datos. Sin embargo, es importante tener en cuenta que la SVM con un kernel radial puede ser propensa a sobreajustar los datos.

- Aplique SVM al conjunto de datos del load_breast_cancer() . Intente entrenar con varios valores de C con un *kernel* lineal. ¿Puede un SVM lineal realizar una buena separación del espacio de características?

SOLUCIÓN

```
cancer = datasets.load_breast_cancer()

print("Características: ", cancer.feature_names)
```

```
#cancer.data

df = pd.DataFrame(cancer.data, columns=['mean radius', 'mean texture',
'mean perimeter', 'mean area','mean smoothness', 'mean compactness', 'mean
concavity','mean concave points', 'mean symmetry', 'mean fractal
dimension','radius error','texture error','perimeter error','area
error','smoothness error','compactness error','concavity error','concave
points error','symmetry error','fractal dimension error','worst
radius','worst texture','worst perimeter','worst area','worst
smoothness','worst compactness','worst concavity','worst concave
points','worst symmetry','worst fractal dimension'])

x = df # Todas las características

y = cancer.target # Todas las etiquetas

x_train, x_test, y_train, y_test =
sklearn.model_selection.train_test_split(x, y, test_size=0.3)



print(x_train[:5], y_train[:5])

from sklearn.preprocessing import StandardScaler

sc = StandardScaler()

x_train = sc.fit_transform(x_train)

x_test = sc.transform(x_test)



from sklearn.svm import SVC

classifierSVML = SVC(kernel = 'linear', random_state = 0)

classifierSVML.fit(x_train, y_train)

from sklearn.model_selection import KFold
```

```
from sklearn.model_selection import GridSearchCV



num_folds = 10

c_values = [0.1, 0.3, 0.5, 0.7, 0.9, 1.0, 1.3, 1.5, 1.7, 2.0]

kernel_values = ['linear', 'poly', 'rbf', 'sigmoid']

param_grid = dict(C=c_values, kernel=kernel_values)

model = SVC()

kfold = KFold(n_splits=num_folds)

grid = GridSearchCV(estimator=model, param_grid=param_grid,
scoring='accuracy', cv=kfold)

grid_result = grid.fit(x_train, y_train)

print("Mejor: %f usando %s" % (grid_result.best_score_,
grid_result.best_params_))

means = grid_result.cv_results_['mean_test_score']

stds = grid_result.cv_results_['std_test_score']

params = grid_result.cv_results_['params']

for mean, stdev, param in zip(means, stds, params):

    print("%f (%f) with: %r" % (mean, stdev, param))

classifierSVML = SVC(C=0.7, kernel = 'linear', random_state = 0)

classifierSVML.fit(x_train, y_train)

Y_pred = classifierSVML.predict(x_test)
```

```
from sklearn.metrics import confusion_matrix

cm = confusion_matrix(y_test, Y_pred)

cm
```

7.9. Referencias bibliográficas

Bhattacharyya, S., Jha, S., Tharakunnel, K. y Westland, J. C. (2011). Data mining for credit card fraud: A comparative study. *Decision Support Systems, 50*(3), 602-613.

Bishop, C. M. (2006). *Pattern recognition and machine learning*. Springer.

Cai, J., Luo, J., Wang, S. y Yang, S. (2018). Feature selection in machine learning: A new perspective. *Neurocomputing, 300*, 70-79.

Cortes, C. y Vapnik, V. (1995). Support-vector networks. *Machine learning, 20*(3), 273-297.

Gao, L., Ye, M., Lu, X. y Huang, D. (2017). Hybrid method based on information gain and support vector machine for gene selection in cancer classification. *Genomics, Proteomics and Bioinformatics, 15*(6), 389-395.

Goodfellow, I., Bengio, Y. y Courville, A. (2016). *Deep learning*. MIT press.

Hastie, T., Tibshirani R., Friedman, J. (2009). The Elements of Statistical Learning. Springer.

James, G., Witten, D., Hastie, T y Tibshirani, R. (2013). *An Introduction to Statistical Leaerning with Applications*. Springer.

Mangasarian, O. L., Street, W. N. y Wolberg, W. H. (1995). Breast cancer diagnosis and prognosis via linear programming. *Operations Research, 43*(4), 570-577.

Maulud, D. y Abdulazeez, A. M. (2020). A review on linear regression comprehensive in machine learning. *Journal of Applied Science and Technology Trends, 1* (2), 140-147.

Molnar, C. (2020). *Interpretable machine learning*. Lulu. com.

Mullen, T. y Collier, N. (2004). Sentiment analysis using support vector machines with diverse information sources. Proceedings of the 2004 conference on empirical methods in natural language processing (pp. 412-418).

Ren, Y., Hu, F. y Miao, H. (2016). *The optimization of kernel function and its parameters for SVM in well-logging*. 2016 13th International Conference on Service Systems and Service Management (ICSSSM), 1-5.

Vapnik, V. N. (1997). The Support Vector method. En W. Gerstner, A. Germond, M. Hasler, J. D. Nicoud (eds), *Artificial Neural Networks — ICANN'97. ICANN 1997. Lecture Notes in Computer Science*, vol 1327. Springer. https://doi.org/10.1007/BFb0020166

ISLR Chapter 9: Support Vector Machines

Data Science Analytics. (2018, junio 15). *StatsLearning Chapter 9 - part 1* [Vídeo]. YouTube. https://www.youtube.com/playlist?list=PL5-da3qGB5IDl6MkmovVdZwyYOhpCxo5o

Primer vídeo de la lista de reproducción con un conjunto de lecciones de los profesores de Stanford Trevor, Hastie and Rob Tibshirani, autores del libro *An Introduction to Statistical Learning with Applications in R (ISLR)*. En esta lista están todas las lecciones relativas a SVM.

Support Vector Machines in Python with scikit learn

Scikit learn. (s.f.). *1.4. Support Vector Machines*. https://scikit-learn.org/stable/modules/svm.html

Documentación oficial de la librería scikit learn sobe su implementación de SVM. Este recurso es de gran utilidad para conocer las características propias de la implementación de SVM en esta librería en concreto. Las explicaciones están acompañadas de ejemplos didácticos.

Ejemplo para predicción sobre comportamientos de clientes

Worrel, N. (2020). *Customer Predictive Analytics with Support Vector Machines (SVM) and Plotly*. Medium. https://towardsdatascience.com/predictive-analytics-on-customer-behavior-with-support-vector-machines-svm-7e68fd2be610

Artículo de blog en el que explica de manera detallada cómo hace una implementación de un clasificador de clientes para descubrir qué métodos de marketing son más eficaces para determinados segmentos de clientes. Al final se buscará predecir si un cliente responderá o no a una llamada de ventas en función de sus datos demográficos y de comportamiento pasado. En el artículo se muestra el desarrollo completo de la construcción de un clasificador incluyendo la etapa de EDA, y de evaluación. Junto con la implementación se aporta una explicación de lo que es SVM.

Ejemplo de implementación de Support Vector Regression

Rasifaghihi, N. (2023). *From Theory to Practice: Implementing Support Vector Regression for Predictions in Python*. Medium. https://medium.com/@niousha.rf/support-vector-regressor-theory-and-coding-exercise-in-python-ca6a7dfda927

Artículo de blog en el que explica de manera detallada cómo aplicar SVR a un caso práctico. En el artículo se explican las características propias de las tareas de regresión y cómo se abordan con SVR, para después mostrar su uso con la librería scikit learn.

1. ¿Cuál de las siguientes afirmaciones sobre las máquinas de vector de soporte es falsa?

A. Las SVM son eficaces para resolver problemas de clasificación binaria y también se pueden extender a problemas de regresión.

B. El objetivo principal de las SVM es encontrar un hiperplano de separación óptimo que maximice el margen entre las clases en el espacio de características.

C. Las SVM pueden manejar eficazmente conjuntos de datos de alta dimensionalidad y son robustas frente al sobreajuste cuando se utilizan técnicas como el margen suave y la optimización convexa.

D. Las SVM siempre requieren que los datos sean linealmente separables en el espacio de características.

2. ¿Cuál de las siguientes afirmaciones sobre los hiperplanos es cierta?

A. Un hiperplano solo se define para dos dimensiones.

B. Límite de decisión entre clases que separa de manera óptima las instancias de datos, en espacio de características multidimensional.

C. El hiperplano en SVM siempre pasa por el origen en el espacio de características.

D. Un hiperplano busca el sobreajuste de las instancias de datos.

3. ¿Cuál de las siguientes afirmaciones sobre los vectores en SVM es falsa?

A. Los vectores de soporte en SVM son puntos de datos que se encuentran más cerca del límite de decisión (hiperplano) entre diferentes clases.

B. La posición y la orientación del hiperplano en SVM están determinadas por los vectores de soporte.

C. Los vectores de soporte son cruciales para definir el margen, que es la distancia entre el hiperplano y los puntos de datos más cercanos de cada clase.

D. Los vectores de soporte en SVM se seleccionan aleatoriamente a partir de los datos de entrenamiento.

4. ¿Cuál de las siguientes afirmaciones es cierta?

A. Para una serie de datos de entrenamiento de un problema binario existen múltiples hiperplanos posibles.

B. Solo existe un hiperplano posible para cada uno de los conjuntos de datos.

C. Los hiperplanos hay que crearlos únicamente para realizar predicciones.

D. El objetivo de SVM es encontrar el hiperplano óptimo que maximice el margen entre todos los puntos de datos sin importar el error de clasificación.

5. ¿Cuál de las siguientes afirmaciones sobre un clasificador de margen máximo es falsa?

A. El clasificador de margen máximo, implementado en SVM, tiene como objetivo encontrar un hiperplano que maximice el margen entre clases.

B. El clasificador de margen máximo selecciona el hiperplano que mejor separa los datos de entrenamiento y maximiza la distancia (margen) a los puntos de datos más cercanos de cada clase.

C. El clasificador de margen máximo está diseñado para mejorar el rendimiento de la generalización maximizando la separación entre clases, reduciendo el riesgo de sobreajuste.

D. El clasificador de margen máximo siempre requiere que las clases sean perfectamente separables mediante un hiperplano.

6. Indica cuál de las siguientes afirmaciones sobre *soft margin* es falsa.

A. SVM de *soft margin* permite cierta clasificación errónea de los puntos de datos de entrenamiento al introducir variables débiles que relajan la rigurosidad del margen.

B. SVM de *soft margin* se utiliza cuando los datos de entrenamiento no son perfectamente separables por un hiperplano, lo que permite un límite más flexible que puede ser adecuado con datos ruidosos.

C. El parámetro de regularización C en SVM *soft margin* controla el equilibrio entre maximizar el margen y minimizar el error de clasificación en los datos de entrenamiento.

D. SVM de *soft margin* siempre funciona mejor que SVM de *hard margin* en conjuntos de datos linealmente separables.

7. ¿Cuál de las siguientes afirmaciones sobre el parámetro de regularización C en SVM es falsa?

A. El parámetro de regularización C en SVM controla el equilibrio entre maximizar el margen y minimizar el error de clasificación en los datos de entrenamiento.

B. Un valor más pequeño de C conduce a un margen más amplio, pero puede dar lugar a más errores de entrenamiento permitidos por el modelo.

C. Un valor mayor de C da como resultado un margen más estrecho, pero reduce la cantidad de errores de entrenamiento tolerados por el modelo, lo que podría provocar un sobreajuste de los datos de entrenamiento.

D. Aumentar el valor de C en SVM siempre mejora el rendimiento del modelo en datos no vistos.

8. La expansión en forma de polinomios:

A. Permite capturar relaciones no lineales entre las características originales al mapear los datos a un espacio de características de mayor dimensión.

B. La expansión en forma de polinomios en SVM solo se puede realizar utilizando un *kernel* lineal.

C. La expansión en forma de polinomios en SVM siempre mejora el rendimiento del modelo en comparación con un enfoque lineal.

D. La expansión en forma de polinomios en SVM transforma automáticamente el espacio de características original a un espacio de mayor dimensión sin introducir complejidad adicional.

9. ¿Cuál de las siguientes afirmaciones es una ventaja de los *kernels*?

A. Permiten utilizar un menor volumen de información.

B. El coste computacional es menor al utilizar *kernels*.

C. Los *kernels* permiten obtener fronteras de decisión no lineales.

D. Los *kernels* reducen la dimensionalidad de los datos al mapearlos a un espacio de características mayor.

10. ¿Cuál de las siguientes afirmaciones sobre la función sigmoide es falsa?

A. La función de *kernel* sigmoide en SVM se define como $k\left(x,x^{'}\right)=\tanh\left(\alpha x^{T}x^{1}+c\right)$ donde α y c son parámetros ajustables.

B. La función de *kernel* sigmoide puede ser útil para modelar relaciones no lineales en SVM y puede capturar patrones complejos en los datos.

C. La función de *kernel* sigmoide transforma los datos a un espacio de características de mayor dimensión utilizando una función sigmoide para facilitar la separación lineal de clases.

D. La función de *kernel* sigmoide en SVM siempre produce un mejor rendimiento que otros *kernels*, como el *kernel* lineal o el *kernel* polinomial.