SVM y Árboles de Decisión: Una Aplicación Práctica en una Base de Datos Real

SVM and Decision Trees: A Practical Application on a Real Dataset

Trabajo Fin de Grado en Matemáticas
Universidad de Málaga

Autor: Teresa Moral Fernández

Área de conocimiento y/o departamento: Área de Estadística e Investigación Operativa.

Fecha de presentación: Junio de 2025

Tema: Optimización, Análisis de Datos e Inteligencia Artificial

Tipo: Bibliográfico

Modalidad: individual

Número de páginas: 54

DECLARACIÓN DE ORIGINALIDAD DEL TFG

D./Dña. *Teresa Moral Fernández*, con DNI ..., estudiante del Grado en *Matemáticas* de la Facultad de Ciencias de la Universidad de Málaga,
DECLARO:

Que he realizado el Trabajo Fin de Grado titulado "*SVM y Árboles de Decisión: Una Aplicación Práctica en una Base de Datos Real*" y que lo presento para su evaluación. Dicho trabajo es original y todas las fuentes bibliográficas utilizadas para su realización han sido debidamente citadas en el mismo.

De no cumplir con este compromiso, soy consciente de que, de acuerdo con la normativa reguladora de los procesos de evaluación de los aprendizajes del estudiantado de la Universidad de Málaga de 23 de julio de 2019, esto podrá conllevar la calificación de suspenso en la asignatura, sin perjuicio de las responsabilidades disciplinarias en las que pudiera incurrir en caso de plagio.

Para que así conste, firmo la presente en Málaga, el *6 de junio de 2025*

Fdo:.....

A mi familia, por ser un apoyo incondicional y mi modelo a seguir en todos los aspectos de mi vida. Por haberme enseñado el poder y la importancia de la enseñanza y el saber.

A mis abuelos, porque nada me hará más feliz que verlos leyendo esto.

A mis compañeros y amigos, que han sido mi refugio y hogar durante tantos años inolvidables, con quienes he compartido mi vida y me han ayudado a seguir adelante en cada etapa, enseñándome que la amistad siempre perdura si se cuida.

A Málaga, por haber conseguido enamorarme en todos los sentidos. Siempre te estaré agradecida por haberme brindado tantas enseñanzas y tanto amor.

A mí, por ser tan persistente y trabajadora y por poder decir, al fin, que lo he conseguido.

Índice general

SVM y Árboles de Decisión: Una Aplicación Práctica en una Base de Datos Real

Resumen

Este trabajo tiene como objetivo introducir los principios fundamentales del aprendizaje supervisado, en particular, se centrará en la clasificación binaria. Para ello, se presentan y comparan dos modelos de clasificación: uno de carácter lineal resuelto a partir de un problema de optimización, conocido como Máquinas de Vectores Soporte, y otro de naturaleza heurística y carácter no lineal, los Árboles de Decisión. Ambas metodologías se aplican a una base de datos real.

El Capítulo 1 introduce los conceptos fundamentales del aprendizaje automático, distinguiendo entre sus principales tipologías. En particular, se profundiza en el aprendizaje supervisado y se presenta el concepto de clasificación binaria, que será el enfoque adoptado a lo largo del trabajo.

En el Capítulo 2 se estudia la clasificación binaria mediante el uso de Máquinas de Vectores Soporte (en inglés, *Support Vector Machines*). Se analizan tanto el caso de datos linealmente separables, conocido como SVM de margen duro, como el caso más general en el que se permiten ciertos errores en la clasificación, abordado mediante el SVM de margen blando.

El Capítulo 3 se centra en los Árboles de Decisión como técnica alternativa de clasificación. Se describen sus elementos fundamentales, así como el proceso de construcción y entrenamiento del modelo. Asimismo, se explica en detalle el funcionamiento del algoritmo CART, utilizado para su resolución.

El Capítulo 4 presenta los experimentos computacionales realizados. Se entrenan y evalúan los modelos introducidos en los capítulos anteriores utilizando una base de datos real procedente de la estación meteorológica de la Base Naval de Rota (Cádiz). Además, se comparan ambos modelos aplicándolos a diferentes combinaciones de variables de entrada.

Por último, el Capítulo 5 recoge las conclusiones extraídas del estudio y plantea posibles líneas futuras de trabajo.

Palabras clave:

Aprendizaje supervisado, clasificación binaria, Máquinas de vectores soporte, árboles de decisión, Optimización, Heurísticos.

SVM and Decision Trees: A Practical Application on a Real Dataset

Abstract

This work aims to introduce the fundamental principles of supervised learning, with a particular focus on binary classification. For this purpose, two classification models are presented and compared: one linear model, formulated as an optimization problem and known as *Support Vector Machines* (SVM), and another heuristic and non-linear model, *Decision Trees*. Both methodologies are applied to a real dataset.

Chapter 1 introduces the core concepts of machine learning, outlining its main categories. Special emphasis is placed on supervised learning, and the notion of binary classification is developed, as it constitutes the central theme of the study.

Chapter 2 explores binary classification through the use of Support Vector Machines. Both the case of linearly separable data, known as *hard-margin SVM*, and the more general case allowing classification errors, addressed via *soft-margin SVM*, are analyzed.

Chapter 3 focuses on Decision Trees as an alternative classification technique. The key components of the model are described, along with the construction and training process. The *CART* algorithm, used to implement the model, is also explained in detail.

Chapter 4 presents the computational experiments carried out. The models introduced in previous chapters are trained and evaluated using a real dataset collected from the meteorological station at the Naval Base of Rota (Cádiz). Moreover, both models are compared by applying them to different combinations of input variables.

Finally, Chapter 5 summarizes the conclusions drawn from the study and suggests potential directions for future research.

key words:

SUPERVISED LEARNING, BINARY CLASSIFICATION, SUPPORT VECTOR MACHINES, DECISION TREES, OPTIMIZATION, HEURISTICS.

Capítulo 1

Introducción

El objetivo principal del aprendizaje automático es desarrollar modelos que aprendan ciertas características según la información dada por los datos. Podemos distinguir tres tipos de aprendizaje automático: el aprendizaje supervisado, el aprendizaje no supervisado y el aprendizaje por refuerzo. En los dos primeros, el modelo aprende a partir de datos ya proporcionados a través del llamado conjunto de entrenamiento. En cambio, el aprendizaje por refuerzo se basa en la interacción del conocido como agente con un entorno, es decir, los datos se van adquiriendo a través de las distintas iteraciones del algoritmo, [32].

En el **aprendizaje supervisado** el *conjunto de entrenamiento* está formado por tuplas de dos componentes, la primera corresponde a las variables de entrada (*inputs* o *variables explicativas*) y la segunda componente es la variable de salida (*output*), [11]. El objetivo del conjunto de entrenamiento es entrenar al modelo y obtener sus parámetros.
Por otro lado, además del conjunto de entrenamiento se dispone del llamado *conjunto test*. Está formado por variables de entrada que no se han usado anteriormente para entrenar el modelo y se utiliza para evaluar el rendimiento del mismo una vez que ha sido entrenado. Es fundamental entender que el conjunto de entrenamiento se utiliza para enseñar al modelo a reconocer patrones a partir de los datos dados, mientras que el objetivo del conjunto test es evaluar la eficiencia de dicho modelo en muestras que no se han visto antes. Esta distinción es crucial para evitar el llamado *sobreajuste* y garantizar que el modelo generalice correctamente, [23].
Para comprender bien este aprendizaje, supongamos que queremos detectar correos no deseados, [19]. Para ello, consideramos un conjunto de entrenamiento formado por varios correos electrónicos ya etiquetados previamente como "spam" o "no spam". Es decir, las variables de salida serían "spam" o "no spam". Por otro lado, las *variables explicativas* incluyen la frecuencia de palabras como "gratis" u "oferta", el número de enlaces o si hay archivos adjuntos sospechosos. Con estos datos, se construye un modelo que permite predecir si un nuevo correo (definido a través de sus variables explicativas) es "spam" o "no spam".

En el **aprendizaje no supervisado** también se asume conocido un conjunto de entrenamiento. A diferencia de lo que ocurre en el aprendizaje supervisado, este conjunto sólo está formado por las variables de entrada, es decir, no hay variable de salida. El objetivo del aprendizaje no supervisado es encontrar un modelo que permita identificar las relaciones y patrones dentro de ese conjunto de variables de entrada. Dos ejemplos clásicos son el clustering o agrupamiento y la reducción de la dimensionalidad, [14].

Veámos a continuación un ejemplo muy conocido e interesante sobre el aprendizaje no supervisado: A finales de los 90, un supermercado decidió analizar los datos de compra de sus clientes para descubrir patrones de comportamiento. Disponía de un conjunto de entrenamiento formado únicamente por las variables de entrada: qué productos compraba cada cliente, su edad, su género, etc., pero no había ninguna variable de salida que indicara, por ejemplo, un tipo de cliente o una categoría de compra. Se trata, por lo tanto, de un problema a tratar con aprendizaje no supervisado. Al buscar patrones en esos datos, descubrieron una sorprendente correlación entre la compra de pañales y cerveza, especialmente entre hombres de 25 a 35 años. El supermercado colocó ambos productos juntos, lo que impulsó las ventas de estos entre un 10 % y un 15 %, [15].

Por último, pasamos al **aprendizaje por refuerzo**. Este método simula la manera en la que los seres humanos adquirimos habilidades, basándose en un sistema de recompensas que fortalecen nuestro comportamiento, [24].
Durante el algoritmo, el llamado agente, aprende a tomar decisiones autónomas para maximizar una recompensa en un entorno. Por ejemplo, supongamos que se quiere entrenar a un perro: el perro (*agente*) aprende a realizar acciones en el parque (*entorno*) para obtener recompensas, como golosinas, y mejora su comportamiento con el tiempo. Del mismo modo, el agente toma acciones en su entorno para obtener recompensas y ajusta su comportamiento en función de lo que aprende, [30].

En este trabajo nos centraremos en el aprendizaje supervisado y, en concreto, en lo que detallaremos más adelante como clasificación binaria.

1.1. Aprendizaje supervisado

El **aprendizaje supervisado** se caracteriza por la utilización de conjuntos de datos que cuentan con etiquetas, los cuales son empleados para entrenar modelos y hacer predicciones de nuevos datos no vistos, [19].

En el aprendizaje supervisado podemos distinguir dos tareas: clasificación y regresión. La diferencia entre estas dos tareas radica en cómo es la variable respuesta.
En la **clasificación**, la variable respuesta es cualitativa. Un ejemplo podría ser el reconocimiento de distintas letras escritas a mano, [26]. Normalmente, suele codificarse con números enteros a través de un conjunto de clases $\mathcal{C}$. Si sólo hay dos clases, hablamos de clasificación binaria, y tendremos $\mathcal{C} = \{-1, 1\}$, $\mathcal{C} = \{0, 1\}$, o $\mathcal{C} = \{1, 2\}$ de acuerdo al tipo de modelo que se esté usando. Si la variable respuesta tiene tres o más etiquetas trataremos con la clasificación multiclase, que codificaremos por ejemplo, con $\mathcal{C} = \{1, 2, 3, \ldots\}$.

Por otro lado, en la **regresión** la variable respuesta es *cuantitativa*, y por tanto, es un valor que se encuentra en $\mathbb{R}$. Esta tarea se emplea, por ejemplo, para predecir el valor de la vivienda en función de variables explicativas como ubicación, tamaño, etc, [32].

1.1.1. Descripción formal

A continuación, pasaremos a dar una descripción formal de las componentes definidas con anterioridad. Asumimos dado un **conjunto de entrenamiento** formado por n muestras y denotado como:

$$\mathcal{S} = \{(\vec{x}_1, y_1), \ldots, (\vec{x}_n, y_n)\} \tag{1.1}$$

Cada $\vec{x}_i = (x_i^1, \cdots, x_i^p) \in \mathbb{R}^p$ es un vector de entrada del conjunto de entrenamiento. En el caso en el que no queramos representar la dependencia de cada individuo, escribimos de manera genérica $x = (x^1, \cdots, x^p)$. Por otro lado, y_i es la *variable de salida* asociada al individuo i, [29].

El objetivo del aprendizaje supervisado es construir, a partir del conjunto de entrenamiento, $\mathcal{S}$, un modelo de predicción, $f : \mathbb{R}^p \to \mathbb{R}$ de forma que dado un vector de entrada x no visto anteriormente, se haga una predicción de la salida denotada como $\hat{y}$, es decir, $\hat{y} = f(x)$. Dependiendo de los valores que tome esta y se trabajará de una forma u otra. Es importante entender que si y toma valores en $\mathbb{R}$ entonces $\hat{y}$ también lo hará. Este comportamiento también se produce en la clasificación donde la variable de salida es cualitativa.

A continuación, nos centraremos en la clasificación y la expondremos con detalle.

1.1.2. Clasificación

En este tipo de problemas, la variable respuesta y es de tipo **cualitativo**. Estas variables suelen tomar valores en un conjunto de clases $\mathcal{C}$ contenido en el conjunto de los números enteros. Es decir,

$$y_i \in \mathcal{C} \subset \mathbb{Z}, \quad \forall i.$$

En la *clasificación binaria*, las variables *cualitativas* suelen representarse numéricamente con $\{-1, 1\}$ o $\{0, 1\}$.

A continuación veremos un ejemplo, [1], que usaremos de ahora en adelante para ilustrar cómo se representan los elementos descritos con anterioridad.

Ejemplo 1.1. Suponemos que queremos averiguar si un coche es familiar o no de acuerdo a ciertas características. Se parte de un conjunto de coches previamente etiquetados como "familiares" o "no familiares".

Tras consultar con expertos, se concluye que las características más relevantes para diferenciar un coche familiar de uno que no lo es son el **precio** y la **potencia del motor**. Estas serán las variables que se usarán en el análisis, [1].

Comencemos representando al conjunto de entrenamiento:
Denotemos el precio (por ejemplo, en euros) como la primera característica de entrada x^1 y la potencia del motor (por ejemplo, en caballos de vapor) como la segunda característica x^2. Así, representamos cada coche utilizando dos valores numéricos, [1]:

$$\vec{x} = (x^1, x^2) \tag{1.2}$$

Y su etiqueta indica su tipo:

$$y = \begin{cases} 1 & \text{si } \vec{x} \text{ es un coche familiar} \\ 0 & \text{si } \vec{x} \text{ no es un coche familiar} \end{cases} \tag{1.3}$$

Cada coche se representa mediante un par ordenado $(\vec{x}, y)$, y el conjunto de entrenamiento contiene n de estos ejemplos:

$$\mathcal{S} = \{(\vec{x}_i, y_i)\}_{i=1}^{n} \tag{1.4}$$

Suponemos conocido este conjunto de entrenamiento formado por 13 coches. La primera columna indica el identificador del coche. Las columnas 2 y 3 indican el precio del coche en miles de euros y la potencia del motor en caballos de vapor. Finalmente, la última columna indica si el coche ha sido etiquetado como familiar (y=1) o no familiar (y=0).

Coche	$\boldsymbol{x^1}$ (Precio, €)	$\boldsymbol{x^2}$ (Potencia, CV)	$\boldsymbol{y}$
1	18,000	80	1
2	22,000	100	1
3	25,000	107	1
4	35,000	133	1
5	15,000	120	1
6	28,000	167	1
7	12,000	60	1
8	45,000	200	0
9	62,000	300	0
10	50,000	220	0
11	70,000	350	0
12	40,000	180	0
13	55,000	280	0

Tabla 1.1: Conjunto de entrenamiento $\mathcal{S}$ con variables x^1: Precio, x^2: Potencia y salida y.

Estos datos nos permitirán extraer patrones sobre qué características comparten los coches familiares y cuáles los distinguen de otros tipos de vehículos, para así hacer una buena predicción.

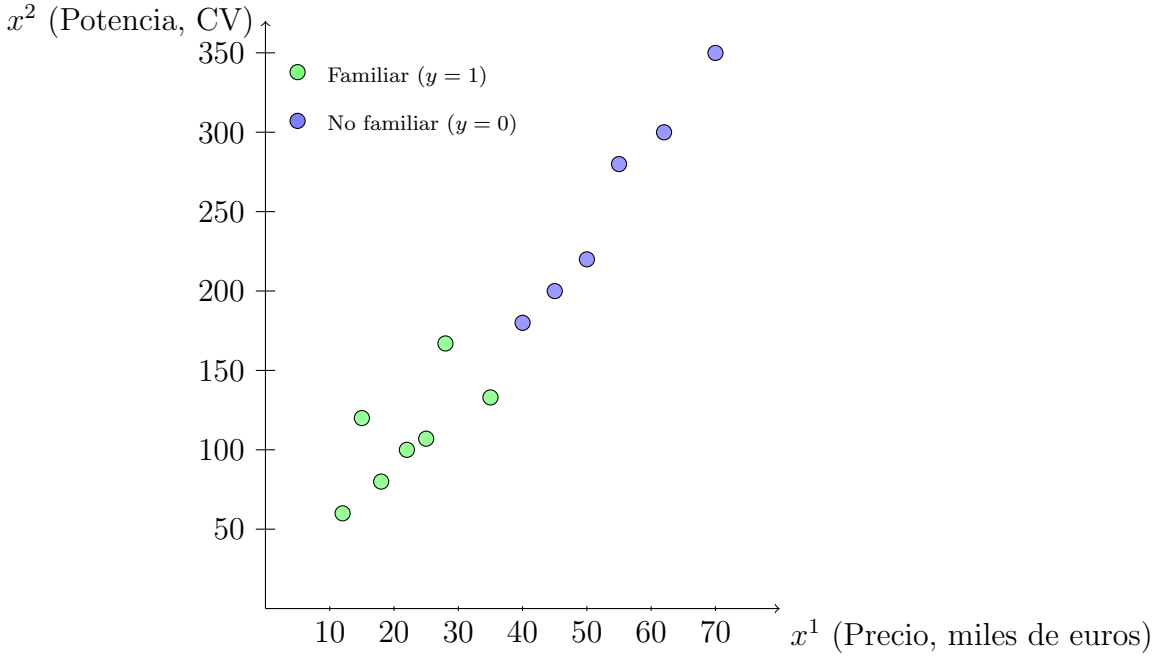


Figura 1.1: Representación del conjunto de entrenamiento

Como se observa en la Figura 1.1, los datos de entrenamiento pueden ser graficados en el espacio bidimensional (x^1, x^2), donde x^1 es el precio en miles de euros y x^2 es la potencia en caballos de vapor de cada uno de los coches de la Tabla 1.2. Los coches considerados familiares (y=1) se representan con círculos verdes, a diferencia de los no familiares (y=0) que son círculos azules.

Estamos ante un problema de *aprendizaje supervisado*, dado que contamos con datos etiquetados, y en concreto ante una *clasificación binaria* puesto que sólo se tienen dos clases: "familiar" y "no familiar". Tomando un modelo de clasificación adecuado, se tiene que un coche se considera familiar si cumple lo siguiente:

$$12\,000 \leq \text{precio} \leq 35\,000 \quad \text{y} \quad 60 \leq \text{potencia} \leq 167. \tag{1.5}$$

En otras palabras, podemos expresar la regla de predicción para un nuevo punto $\vec{x} = (x^1, x^2)$ como

$$\hat{y} = f(\vec{x}) = \begin{cases} 1, & \text{si } 12\,000 \leq x^1 \leq 35\,000 \ \wedge \ 60 \leq x^2 \leq 167, \\ 0, & \text{en caso contrario.} \end{cases} \tag{1.6}$$

Visualmente comprobamos que esta regla concuerda con la región donde aparecen los coches familiares en la Figura 1.1.

Ahora podemos usar esta descripción para hacer predicciones sobre coches nuevos. Consideremos el caso de un coche nuevo, para el cual asumimos las siguientes características:

$$\text{precio} = 28\,000€ \quad \text{y} \quad \text{potencia} = 141\,\text{CV}$$

En la Figura 1.2, se puede observar en qué región se encuentra el coche nuevo. Siguiendo el modelo de predicción de la Ecuación (1.6) tenemos que el nuevo coche cumple con los

criterios definidos para la clase de coches familiares, es decir, $\hat{y} = 1$, y por tanto se considera familiar.

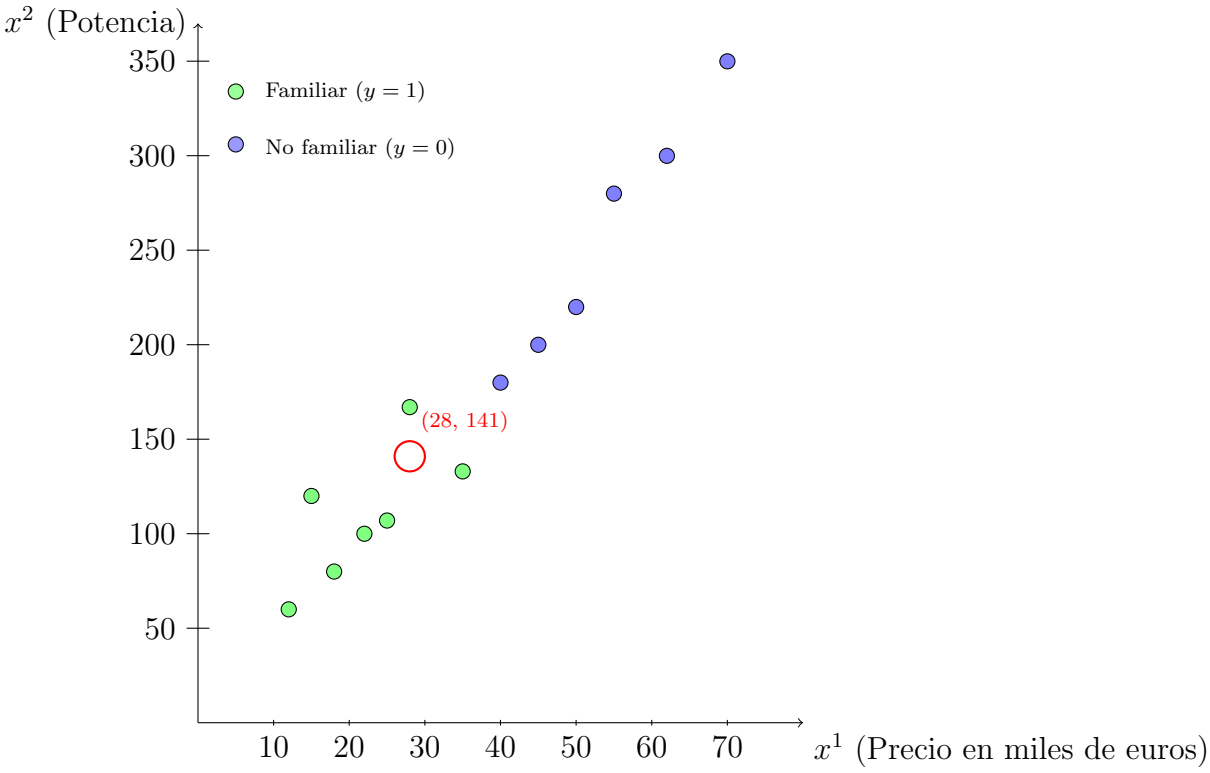


Figura 1.2: Representación del conjunto de entrenamiento con el coche nuevo

Este análisis no solo permite realizar la predicción, sino también interpretar y comunicar de forma sencilla qué características son comunes entre los coches familiares: *precio moderado y potencia contenida*, lo cual se alinea con los criterios de eficiencia, accesibilidad y utilidad que suelen asociarse con este tipo de vehículos.

Una vez visto el ejemplo y entendido el proceso de clasificación binaria, donde el objetivo es asignar a cada dato nuevo una de dos posibles categorías, es natural extender esta idea a problemas más generales en los que existen más de dos clases posibles. Cuando hay más de dos clases, se denomina *clasificación multiclase*, [19].

Para los problemas de **clasificación multiclase**, hay diferentes métodos que se aplican, podemos destacar los métodos *uno-contra-todos (abreviado como OAA del inglés One-Against-All)* y *uno-contra-uno (del inglés One-Against-One abreviado como OAO)*, [21], ambos basados en la clasificación binaria.

En la metodología OAA se construyen K clasificadores binarios, siendo $K > 2$ el número total de clases existentes. Para cada clasificador $m = 1, \ldots, K$ se crean dos grupos ficticios. En el primero, etiquetado por ejemplo con la etiqueta 1, se incluyen los datos de la clase C_m, mientras que en el segundo grupo etiquetado por ejemplo con 0 se incluyen todos los datos del resto de clases, es decir, de las clases C_j, $\forall j \neq m$. Usando esta distinción, se construye una función f_m basándose en algoritmos de clasificación binaria. Finalmente, la clase asignada de entre las K posibles será aquella que maximice las funciones f_m, $\forall m$.

Es decir,

$$\hat{y} = \arg \max_{m \in \{1,\dots,K\}} f_m(\vec{x}) \tag{1.7}$$

Por otro lado, en la metodología OAO se construyen $\frac{K(K-1)}{2}$ clasificadores binarios, uno para cada par de clases posibles donde, de nuevo, K es el número de clases. Para cada clasificador asociado a un par (r, s) tales que $r, s = 1, \dots, K$ y $r < s$, se crean dos grupos ficticios: en el primero, etiquetado por ejemplo con la etiqueta 1, se incluyen los datos de la clase C_r, mientras que en el segundo grupo, etiquetado con la etiqueta 0, se incluyen los datos de la clase C_s. De esta forma, cada clasificador se entrena exclusivamente para distinguir entre dos clases, lo cual ayuda a evitar el problema de desequilibrio de datos característico del método OAA, ya que los tamaños de los dos grupos suelen ser similares. Para hacer la clasificación final de un nuevo dato, se utilizan todas las funciones de clasificación $f_{r,s}$ asociadas a cada par de clases, (C_r, C_s), [31]. Cada clasificador emite un voto a favor de una de las dos clases que compara, el número de votos obtenidos para la clase C_m viene dado por:

$$V_m(\vec{x}) = \sum_{r=1}^{m-1} \mathbb{I}\big(f_{r,m}(\vec{x}) < 0\big) \; + \sum_{s=m+1}^{K} \mathbb{I}\big(f_{m,s}(\vec{x}) \geq 0\big),$$

donde $\mathbb{I}(\cdot)$ es una función que vale 1 si la condición es verdadera y 0 si no lo es.
En la primera suma recorremos los clasificadores que comparan (r, m) con $r < m$. Ahí el voto a favor de la clase m ocurre cuando

$$f_{r,m}(\vec{x}) < 0$$

(porque el signo negativo significa "elige la segunda clase").

En la segunda suma recorremos los clasificadores que comparan (m, s) con $s > m$. Ahí el voto a favor de m ocurre cuando

$$f_{m,s}(\vec{x}) \geq 0$$

(el signo no negativo significa "elige la primera clase").

La clase final asignada será aquella que reciba el mayor número de votos entre todos los clasificadores, es decir:

$$\hat{y} = \arg \max_{m \in \{1,\dots,K\}} V_m \tag{1.8}$$

Ejemplo numérico con $K = 3$

Clases

1. Familiares
2. Deportivos
3. Sedanes de lujo

Funciones de decisión (para $x = (x_1, x_2)$):

$$f_{1,2}(x) = x_1 - x_2,$$
$$f_{1,3}(x) = 2x_1 - x_2 - 1,$$
$$f_{2,3}(x) = -x_1 + 2x_2 + 1.$$

Evaluación para $x = (3, 1)$:

$$f_{1,2}(3,1) = 3 - 1 = 2 \geq 0 \quad \Rightarrow \text{ voto por clase 1,}$$
$$f_{1,3}(3,1) = 2 \cdot 3 - 1 - 1 = 4 \geq 0 \quad \Rightarrow \text{ voto por clase 1,}$$
$$f_{2,3}(3,1) = -3 + 2 \cdot 1 + 1 = 0 \geq 0 \quad \Rightarrow \text{ voto por clase 2.}$$

Cálculo de votos $V_p(x)$:

$$V_1(x) = \mathbb{I}(f_{1,2}(x) \geq 0) + \mathbb{I}(f_{1,3}(x) \geq 0) = 1 + 1 = 2,$$
$$V_2(x) = \mathbb{I}(f_{1,2}(x) < 0) + \mathbb{I}(f_{2,3}(x) \geq 0) = 0 + 1 = 1,$$
$$V_3(x) = \mathbb{I}(f_{1,3}(x) < 0) + \mathbb{I}(f_{2,3}(x) < 0) = 0 + 0 = 0.$$

Clasificación final

$$\hat{y} = \arg \max_{p \in \{1,2,3\}} V_p(x) = 1 \quad \text{(Familiares).}$$

En el Ejemplo 1.1, hemos resuelto un problema de clasificación para dos clases, es decir, se trata de clasificación binaria. ¿Qué ocurre si se quiere ver si un coche es deportivo, sedán de lujo o familiar? Estaríamos tratando con un problema con más de dos etiquetas. En el caso general, tenemos K clases denotadas como C_m, donde $m = 1, \ldots, K$, y una instancia de entrada pertenece a una y sólo a una de ellas, [1].

A continuación, analizamos un caso de estudio con tres clases de coches: *familiar*, *deportivo* y *sedán de lujo*, utilizando como características el **precio** y la **potencia del motor**.

Ejemplo 1.2. Partiremos de un conjunto de entrenamiento $\mathcal{S} = \{(\vec{x_i}, y_i)\}_{i=1}^{n}$ donde $y_i \in \{C_1, C_2, C_3\}$. Durante todo este ejemplo usaremos que un coche con clase C_1=1 corresponde a un coche familiar, C_2=2 a un deportivo y por último, C_3=3 a un sedán de lujo.

Disponemos del siguiente conjunto de entrenamiento formado por 18 coches. La primera columna indica el identificador del coche. Las columnas 2 y 3 indican el precio del coche en euros y la potencia del motor en caballos de vapor. Finalmente, la última columna indica si el coche ha sido etiquetado como familiar (y=1), deportivo (y=2) o sedán de lujo (y=3).

Coche	$\boldsymbol{x^1}$ (Precio, €)	$\boldsymbol{x^2}$ (Potencia, CV)	$\boldsymbol{y}$
1	18,000	80	1
2	22,000	100	1
3	25,000	107	1
4	35,000	133	1
5	15,000	120	1
6	12,000	60	1
7	28,000	167	1
8	100,000	400	2
9	107,000	495	2
10	122,000	510	2
11	145,000	600	2
12	135,000	555	2
13	150,000	625	2
14	60,000	190	3
15	65,000	200	3
16	75,000	230	3
17	80,000	225	3
18	90,000	250	3

Tabla 1.2: Conjunto de entrenamiento $\mathcal{S}$ con variables x^1: Precio, x^2: Potencia y salida y.

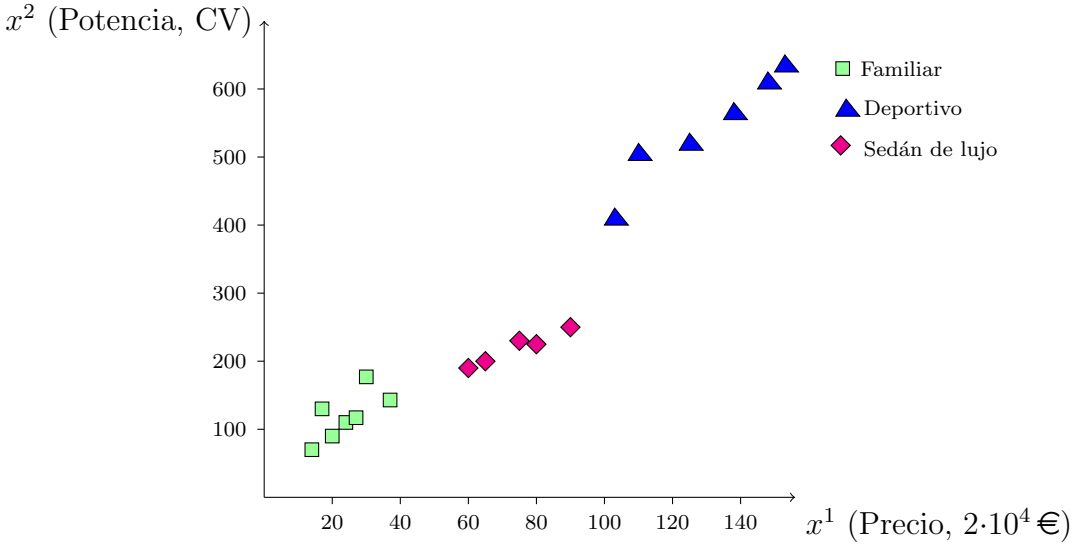


Figura 1.3: Conjunto de entrenamiento del Ejemplo 2.1.

En la Figura 1.3 se muestran con cuadrados verdes los coches familiares, con triángulos azules los coches deportivos y por último, con rombos magentas los sedanes de lujo.
Para resolver este problema usaremos el método **OAA**, definido anteriormente.

En este ejemplo, se usan 3 clasificadores binarios ($m = 1, 2, 3$). A continuación, construiremos esta función para cada uno de ellos:

Por ejemplo, en el caso $m = 1$, queremos distinguir los coches familiares de los no

familiares, por lo que tendríamos

$$y_i^{(1)} = \begin{cases} 1 & \text{si } y_i = C_1 \\ 0 & \text{si } y_i \neq C_1 \end{cases} \tag{1.9}$$

Tras entrenar un modelo de clasificación binaria con las variables explicativas de la Tabla 1.2, y la salida de (1.9), se tiene que la función de predicción asociada al clasificador de $p = 1$, $f_1(\vec{x})$ viene dada por:

$$f_1(\vec{x}) = 2.14 \; - \; 3.26 \times 10^{-5}\, x^1 \; - \; 3.02 \times 10^{-3}\, x^2$$

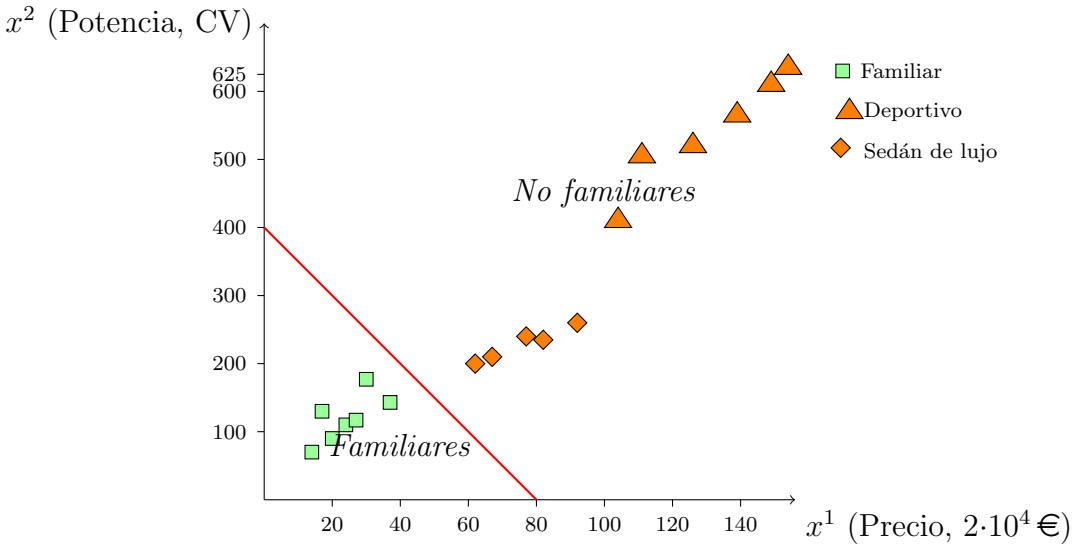


Figura 1.4: Clasificación entre coches familiares ($y = 1$) y no familiares ($y = 2, 3$) en el plano (x^1, x^2), con frontera del clasificador f_1.

En la Figura 1.4, podemos ver los datos del conjunto de entrenamiento, distinguiendo los familiares (en verde) y los no familiares (en naranja). Además, podemos observar en rojo la proyección en el plano (x^1, x^2) de la función de predicción $f_1(\vec{x})$.

De forma análoga, tenemos las etiquetas que representan los coches deportivos vs los no deportivos, así como los sedán vs no sedán vienen dadas por $y_i^{(2)}$ e $y_i^{(3)}$ respectivamente:

$$y_i^{(2)} = \begin{cases} 1 & \text{si } y_i = C_2 \\ 0 & \text{si } y_i \neq C_2 \end{cases}$$

$$y_i^{(3)} = \begin{cases} 1 & \text{si } y_i = C_3 \\ 0 & \text{si } y_i \neq C_3 \end{cases}$$

Asimismo, las funciones de predicción asociadas a los clasificadores $m = 2$ y $m = 3$ son:

$$f_2(\vec{x}) = -3.16 \; + \; 8.63 \times 10^{-6}\, x^1 \; + \; 6.54 \times 10^{-3}\, x^2$$

$$f_3(\vec{x}) = -0.96 \; + \; 2.37 \times 10^{-5}\, x^1 \; - \; 5.8 \times 10^{-3}\, x^2$$

En las figuras a continuación pueden verse los datos del conjunto $\mathcal{S}$, la Figura 1.5 diferencia los coches clasificados como deportivos (en azul) de los no deportivos (en naranja). Análogamente, en la Figura 1.6 se observan los sedanes de lujo (en magenta), en contraste con los coches no sedanes (en naranja). Además, la línea en rojo hace referencia a la proyección en el plano (x^1, x^2) de la función de predicción $f_2(\vec{x})$ y $f_3(\vec{x})$, respectivamente.

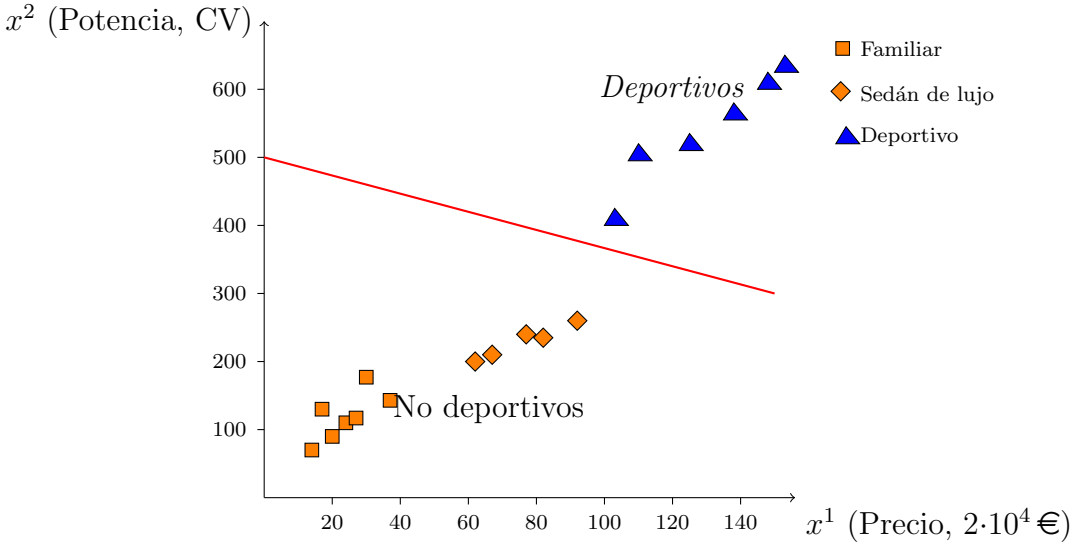


Figura 1.5: Clasificación entre coches deportivos ($y = 2$) frente a los que no lo son ($y = 1, 3$), la línea roja es la proyección en el plano (x^1, x^2) de la función de predicción f_2.

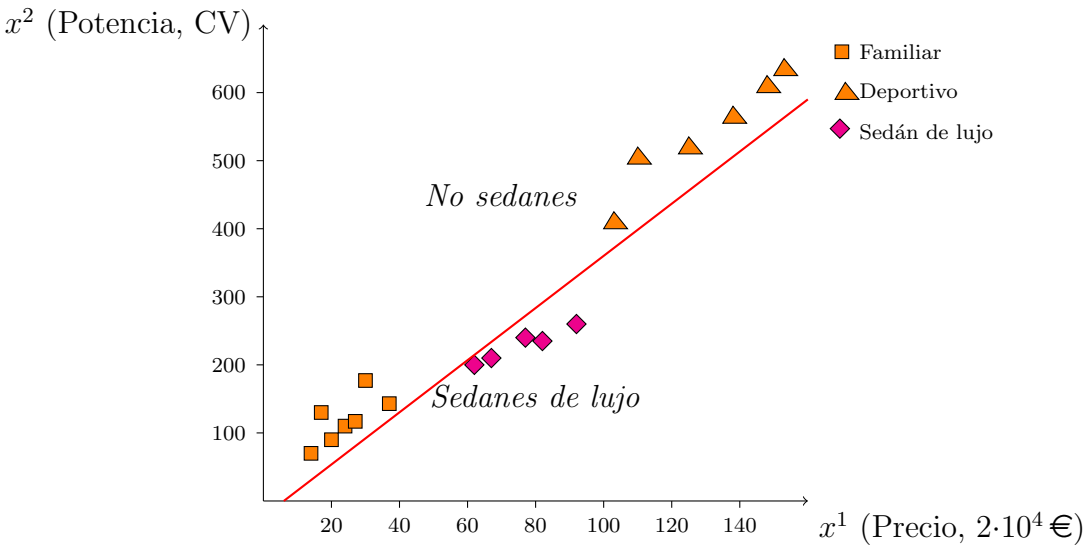


Figura 1.6: Representación de los coches sedanes de lujo ($y = 3$) frente a los no sedanes ($y = 1, 2$), la línea en rojo hace referencia a la proyección en el plano (x^1, x^2) de la función f_3.

Para finalizar este ejemplo, veremos como se hace la predicción de un nuevo punto, de acuerdo a la Ecuación (1.7). Asumimos dado un nuevo coche con un precio de 122 000 € y potencia de 400 CV. Es decir, $\vec{x} = (122\,000, 400)$. La Figura 1.7 muestra la representación del nuevo punto junto con los datos de entrenamiento.

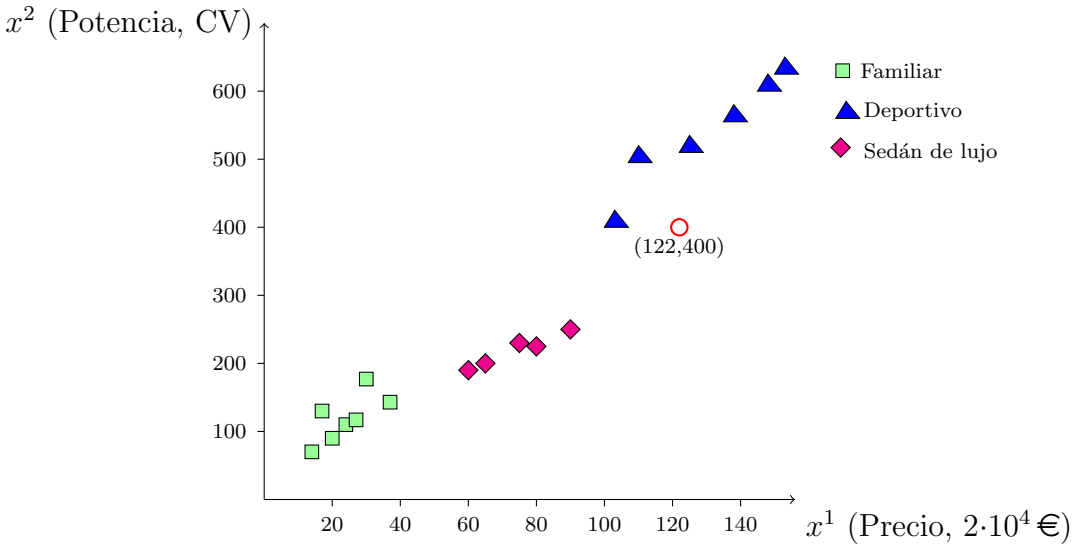


Figura 1.7: Representación del coche nuevo $(122\,000, 400)$

A continuación, evaluaremos las funciones de predicción $f_m(\vec{x})$, $m = 1, 2, 3$ para el nuevo punto:

$$\begin{aligned} f_1(122\,000, 400) &= 2.14 - 3.26 \times 10^{-5} \cdot 120000 - 3.02 \times 10^{-3} \cdot 400 &&= -2.98 \\ f_2(122\,000, 400) &= -3.16 + 8.63 \times 10^{-6} \cdot 120000 + 6.54 \times 10^{-3} \cdot 400 &&= 0.49 \\ f_3(122\,000, 400) &= -0.96 + 2.37 \times 10^{-5} \cdot 120000 - 5.8 \times 10^{-3} \cdot 400 &&= -0.43 \end{aligned}$$

Siguiendo la Ecuación (1.7), tenemos que

$$\hat{y} = \arg \max_{m=1,2,3} f_m(\vec{x}) = 2$$

Por lo tanto, el nuevo coche es clasificado como deportivo.

En conclusión, el método OAA permite una interpretación clara de las características discriminativas para cada clase. Los vehículos familiares se distinguen por presentar un precio y una potencia moderados; los deportivos destacan por su elevada potencia y su alto coste; y los sedanes combinan un precio elevado con una potencia media.

Capítulo 2

Support Vector Machines (SVM)

Las Máquinas de Vectores de Soporte (*Support Vector Machines*) son un método de aprendizaje supervisado lineal utilizado para la clasificación binaria. La idea básica es encontrar un hiperplano que separe los datos en sus dos clases, [6].
Su historia se remonta a finales de los años 70, con los primeros trabajos de Vladimir Vapnik y sus colaboradores, aunque no fue hasta 1995 cuando se publicó el artículo que consolidó formalmente esta técnica dentro del campo del aprendizaje automático ([29]).
Este método ha sido investigado y aplicado en muchos otros campos, por ejemplo, la categorización de textos, el reconocimiento de imágenes y la bioinformática, entre otros, [18].

En este capítulo abordaremos dos tipos de algoritmos, uno para ejemplos linealmente separables (*hard-margin SVM*) y otro para ejemplos cuasi-separables (*soft-margin SVM*), [28].

2.1. Hard margin SVM

Sea el conjunto de datos $\mathcal{S} = \{(\vec{x_i}, y_i)\}_{i=1}^n$, donde $\vec{x_i} \in \mathbb{R}^p$ e $y_i \in \{-1, 1\}, \forall i$. Asumimos que los datos son perfectamente separables, es decir, que se puede definir un hiperplano lineal, capaz de dividir el conjunto en dos partes sin cometer errores de clasificación, [28]. Si denotamos por $\vec{w} = (w_1, ..., w_p) \in \mathbb{R}^p$ al vector normal del hiperplano, $b_0 \in \mathbb{R}$ el término independiente de la ecuación, y $\langle \cdot, \cdot \rangle$ el producto escalar, entonces el hiperplano $D(\vec{x})$ viene dado por:

$$D(\vec{x}) = \langle \vec{w}, \vec{x} \rangle + b_0 \tag{2.1}$$

A este hiperplano se le conoce como hiperplano de separación. De este modo, todos los vectores x_i que cumplan $\langle \vec{w}, \vec{x_i} \rangle + b_0 \geq 0$ tendrán la etiqueta $y_i = 1$, con lo cual quedarán ubicados en un mismo lado del hiperplano, [10]. Como resultado, aquellos ejemplos con $\langle \vec{w}, \vec{x_i} \rangle + b_0 < 0$ tendrán la etiqueta $y_i = -1$ y se situarán en el lado opuesto del mismo. Por lo tanto, este hiperplano deberá cumplir lo siguiente para todo $\vec{x_i} \in \mathcal{S}$:

$$y_i(\langle \vec{w}, \vec{x_i} \rangle + b_0) \geq 0 \qquad i = 1, ..., n \tag{2.2}$$

En la Figura 2.1 podemos observar un ejemplo de un conjunto de datos separados en dos clases por un hiperplano para cierto vector $\vec{w}$ y un valor b_0.

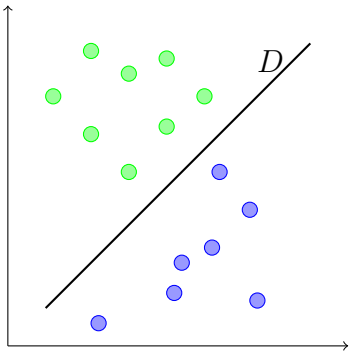


Figura 2.1: Hiperplano de separación (D) en un conjunto linealmente separable.

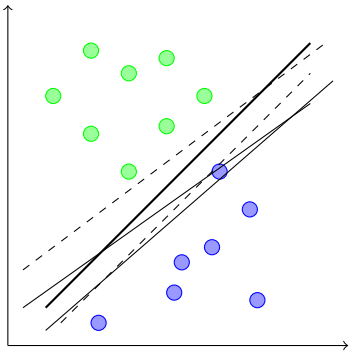


Figura 2.2: Otros hiperplanos de separación posibles.

A partir de la Figura 2.2, se observa que existen muchos clasificadores lineales (hiperplanos) que separan los datos. Sin embargo, solo uno de ellos logra la máxima separación entre los puntos más cercanos de cada clase. La razón por la que esto es importante es que, si utilizamos un hiperplano cualquiera para clasificar, podría quedar más cerca de uno de los conjuntos de datos que del otro, lo cual no es deseable porque puede favorecer más a una clase que a otra. Por ello, surge el concepto de clasificador o hiperplano de margen óptimo como una solución. Para ello debemos introducir los siguientes conceptos:

Podemos definir la distancia entre el hiperplano de separación $D(x)$ y el punto x_i como:

Definición 2.1. Sea $\vec{x_i}$ un dato con clase y_i. Sea $D(\vec{x}) \equiv \langle \vec{w}, \vec{x} \rangle + b_0 = 0$ un hiperplano de separación con $\vec{w} = (w_1, \ldots, w_p) \in \mathbb{R}^p$ el vector normal del hiperplano y $b_0 \in \mathbb{R}$ el término independiente de la ecuación. Entonces, la distancia de $\vec{x_i}$ al hiperplano viene dada por:

$$Dist(\vec{x}_i, D(\vec{x})) = \frac{y_i(\langle \vec{w}, \vec{x}_i \rangle + b_0)}{\|\vec{w}\|} \tag{2.3}$$

Una vez introducida la distancia, podemos definir el *margen* del hiperplano de separación:

Definición 2.2. Sea $D(\vec{x})$ el hiperplano de separación dado en la Ecuación (2.1). Se define el margen τ como la mínima distancia entre dicho hiperplano y el dato más cercano

de cualquiera de las dos clases. En otras palabras, el margen se define como:

$$\tau = \min_{i=1,\dots,n} \frac{y_i(\langle \vec{w}, \vec{x}_i \rangle + b_0)}{\|\vec{w}\|} \tag{2.4}$$

Definición 2.3. Sea el hiperplano $D(\vec{x})$ tal que $D(\vec{x}) = \langle \vec{w}, \vec{x} \rangle + b_0$. Se define el hiperplano óptimo como aquel que maximiza el margen. Es decir, es aquel hiperplano que tiene el margen máximo.

Utilizando la definición de margen máximo y las expresiones (2.2) y (2.3), se puede afirmar que todos los datos de entrenamiento tendrán una distancia al hiperplano de separación óptimo que es mayor o igual al margen definido. Es decir,

$$\frac{y_i(\langle \vec{w}, \vec{x}_i \rangle + b_0)}{\|\vec{w}\|} \geq \tau, \quad i = 1, \dots, n \tag{2.5}$$

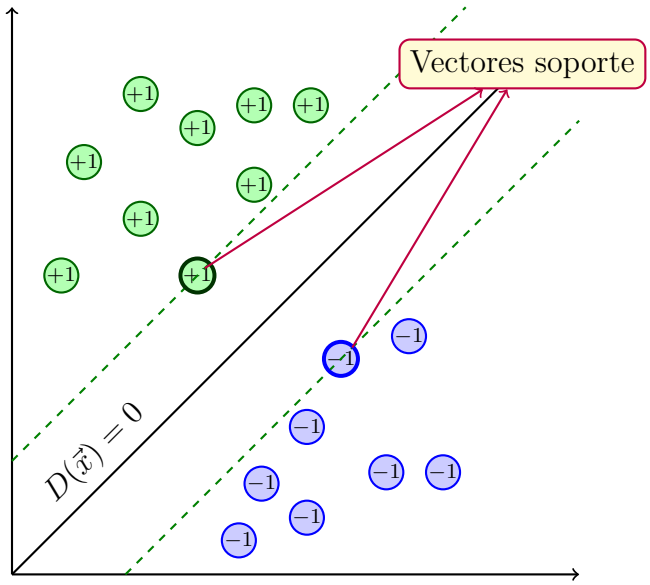


Figura 2.3: Diagrama SVM con vectores de soporte identificados.

En la Figura 2.3 se muestra el hiperplano de separación óptimo y los datos ubicados en la frontera, ambos señalados con una línea gruesa azul y verde respectivamente. Estos datos definen el margen a ambos lados del hiperplano de separación y son conocidos como *vectores soporte*. Por definición los vectores soporte satisfacen que:

$$\frac{y_i(\langle \vec{w}, \vec{x}_i \rangle + b_0)}{\|\vec{w}\|} = \tau \qquad \forall i \in V_S \tag{2.6}$$

donde V_S es el conjunto de vectores soporte.

Sin embargo, existen infinitas formas de definir un mismo hiperplano, por ejemplo, para cierto $\alpha > 0$,

$$\langle \alpha \vec{w}, \vec{x} \rangle + \alpha b_0$$

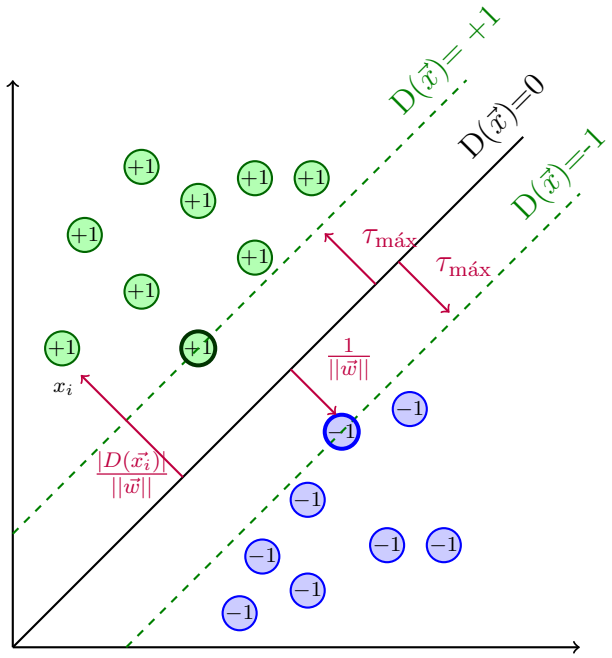


Figura 2.4: El hiperplano óptimo D maximiza el margen $\tau_{\text{máx}}$ entre las clases. Los vectores soporte definen el ancho del margen.

representa el mismo hiperplano que $\langle \vec{w}, \vec{x} \rangle + b_0$ pero tienen distinta norma. En nuestro problema, nos interesa la dirección, no la norma, por lo que normalizamos el margen máximo como sigue:

$$\tau = \frac{1}{||\vec{w}||} \tag{2.7}$$

De las Ecuaciones (2.6) y (2.7) se tiene que maximizar el margen equivale a minimizar $||\vec{w}||$. Veámoslo:

$$\text{máx} \left(\frac{y_i(\langle \vec{w}, \vec{x}_i \rangle + b_0)}{\|\vec{w}\|} \right) = \text{máx}\, \tau = \text{máx}\, \frac{1}{||\vec{w}||} = \text{mín}\, ||\vec{w}|| \tag{2.8}$$

En consecuencia, usando la Ecuación (2.5), se tiene que:

$$y_i D(\vec{x}_i) = y_i(\langle \vec{w}, \vec{x}_i \rangle + b_0) \geq 1, \quad i = 1, \ldots, n \tag{2.9}$$

En concreto, los vectores soporte cumplen que $D(\vec{x}) = +1$ si $y_i = +1$, y $D(\vec{x}) = -1$ si $y_i = -1$.

Otra característica que cumplen los vectores de soporte es la siguiente, [18]:

Proposición 2.1. Sea $\vec{x}_i$ un dato de entrenamiento, con clase y_i. Sea $D(\vec{x}) = \langle \vec{w}, \vec{x} \rangle + b_0$ un hiperplano con vector normal $\vec{w} = (w_1, ..., w_p)$ y b_0 el término independiente. Entonces $\vec{x}_i$ es un vector de soporte si satisface:

$$Dist(\vec{x}_i, D(\vec{x})) = \frac{y_i(\langle \vec{w}, \vec{x}_i \rangle + b_0)}{\|\vec{w}\|} = \frac{1}{||\vec{w}||}$$

En la Figura 2.4, los datos que aparecen marcados con una línea gruesa son vectores soporte y su distancia al hiperplano es $\frac{1}{||\vec{w}||}$, en concordancia con la Proposición (2.1). Esta distancia coincide con $\tau_{\text{máx}}$. Por otro lado, la distancia a cualquier otro dato viene dada por $\frac{|D(\vec{x_i})|}{||\vec{w}||}$.

Por lo tanto, para encontrar el hiperplano óptimo debemos encontrar el hiperplano que maximice su distancia con los vectores de soporte. Como hemos visto en la Ecuación (2.8), esto se consigue minimizando $||\vec{w}||$, sujeto a las restricciones dadas por la expresión (2.9). Formalmente:

$$\begin{aligned} \min_{\vec{w},b_0} \quad & \frac{1}{2}\|\vec{w}\|^2 \\ \text{s.a} \quad & y_i\big(\langle \vec{w}, \vec{x_i}\rangle + b_0\big) \;\geq\; 1, \quad i = 1, \ldots, n \end{aligned} \tag{2.10}$$

Este es el problema de optimización que se utiliza para resolver el SVM de margen duro. Se trata de un problema convexo, ya que minimiza una función cuadrática sujeta a restricciones lineales.

Una vez resuelto el problema (2.10) con técnicas de optimización convexa, [7], se obtienen $\vec{w}^*$ y b_0^*. Sustituyendo estos valores en la Expresión (2.1) se obtiene el hiperplano óptimo siguiente:

$$D(\vec{x}) = \langle \vec{w^*}, \vec{x}\rangle + b_0^* \tag{2.11}$$

Así, si se quiere predecir la salida de un nuevo dato no visto anteriormente, llamado, por ejemplo, $\vec{x}^{\text{test}}$, se evalúa en la expresión (2.11).
Si $D(\vec{x}^{\text{test}}) \geq 0$, se tiene que $\hat{y}^{\text{test}} = 1$, en cambio, si $D(\vec{x}^{\text{test}}) < 0$, entonces $\hat{y}^{\text{test}} = -1$, [16].

2.2. Soft Margin SVM

El problema descrito en la sección anterior tiene un interés limitado desde el punto de vista práctico, ya que los problemas reales suelen incluir ejemplos ruidosos y rara vez son perfectamente separables a través de un clasificador lineal. Para abordar este tipo de situaciones, se adopta una estrategia que relaja la exigencia de separabilidad, permitiendo cierto margen de error en la clasificación de algunos ejemplos del conjunto de entrenamiento.

Diremos que una muestra del conjunto de datos $\mathcal{S} = \{(\vec{x_i}, y_i)\}_{i=1}^n$ es no-separable de forma lineal si no cumple la condición (2.9), es decir, algunos de estos puntos no se han clasificado correctamente mediante el hiperplano de separación lineal.

La Figura 2.5 muestra el diagrama de un SVM de margen blando en que los círculos verdes y círculos azules son puntos de clase +1 y -1, respectivamente, mientras que las líneas verdes discontinuas muestran los hiperplanos $D(\vec{x}) = \pm 1$ que delimitan el margen y la línea sólida negra es el hiperplano de decisión $D(\vec{x}) = 0$. Además, se resaltan con un trazo grueso los vectores soporte. En la figura se ve claramente cómo, a diferencia del SVM de margen duro que exige que todos los puntos queden perfectamente fuera de las bandas de margen, aquí hay varios datos (como algún círculo azul situado por encima de la línea verde de $D(\vec{x}) = +1$ o algún círculo verde cruzando hacia abajo la línea de

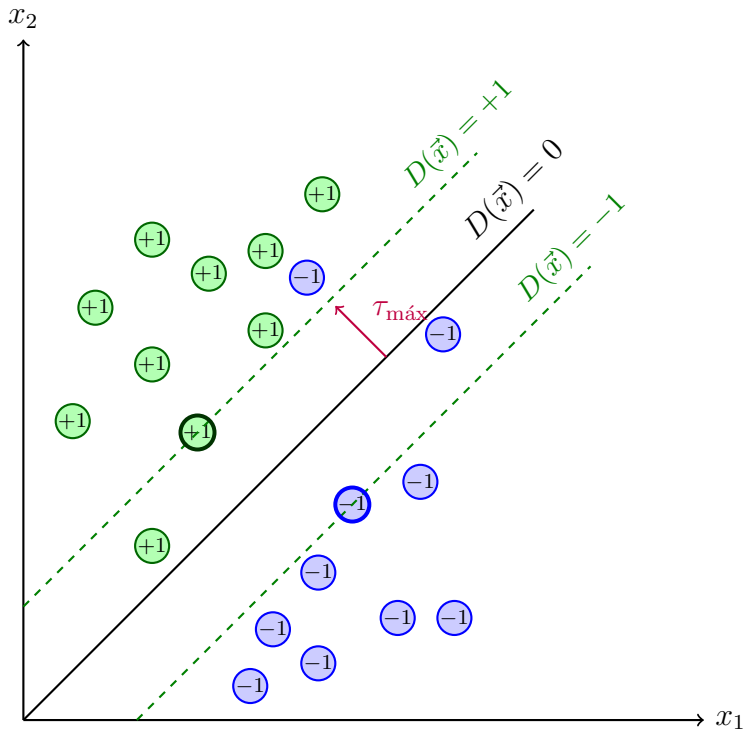


Figura 2.5: Diagrama del SVM de margen blando

$D(\vec{x}) = +1$) que invaden el margen.

Para tratar este caso, se cambian y por lo tanto modifican su predicción las restricciones del hiperplano definido anteriormente en (2.1) introduciendo un conjunto de variables de holgura, $\xi_i \geq 0$, una para cada dato $(\vec{x_i}, y_i)$, esto permitirá cuantificar el número de datos no-separables que se admite, [28]. Por lo tanto, la condición que define al hiperplano de separación en este caso es:

$$y_i D(\vec{x_i}) = y_i(\langle \vec{w}, \vec{x_i} \rangle + b_0) \geq 1 - \xi_i, \quad i = 1, \ldots, n \tag{2.12}$$

Estas variables de holgura cuantifican la distancia de cada punto al borde del margen que le corresponde según su etiqueta. En primer lugar, si $\xi_i = 0$, la restricción es exactamente $y_i(\langle \vec{w}, \vec{x_i} \rangle + b_0) \geq 1$, es decir, el punto está en el lado correcto del hiperplano con una distancia mayor o igual a $\frac{1}{||\vec{w}||}$ y, en consecuencia, está correctamente clasificado. Por otro lado, si $0 < \xi_i < 1$, se tiene que $0 < y_i(\langle \vec{w}, \vec{x_i} \rangle + b_0) < 1$, esto significa que el punto sigue en el lado correcto del hiperplano, pero se encuentra dentro del margen aunque se etiqueta correctamente. Por último, si $\xi_i \geq 1$, $y_i(\langle \vec{w}, \vec{x_i} \rangle + b_0) < 0$, entonces el punto se encuentra en el lado contrario del hiperplano y, debido a esto, estará mal clasificado.

En la Figura 2.6 se ilustra una SVM de margen blando que muestra el hiperplano de separación $D(\vec{x}) = 0$ y las dos líneas de margen discontinuas verdes, $(D(\vec{x}) = \pm 1)$. También se han seleccionado cuatro puntos particulares denotados por $\vec{x_i}, \vec{x_j}, \vec{x_k}$ y $\vec{x_l}$ para ciertos índices i, j, k, l. Se observa que $\vec{x_j}$ y $\vec{x_l}$ tienen asociada una variable de holgura $\xi_j = \xi_l = 0$, mientras que $0 < \xi_i < 1$ y $\xi_k > 1$. Así, se muestra cómo los puntos con holgura cero quedan justo sobre el margen, los de holgura entre cero y uno invaden el margen

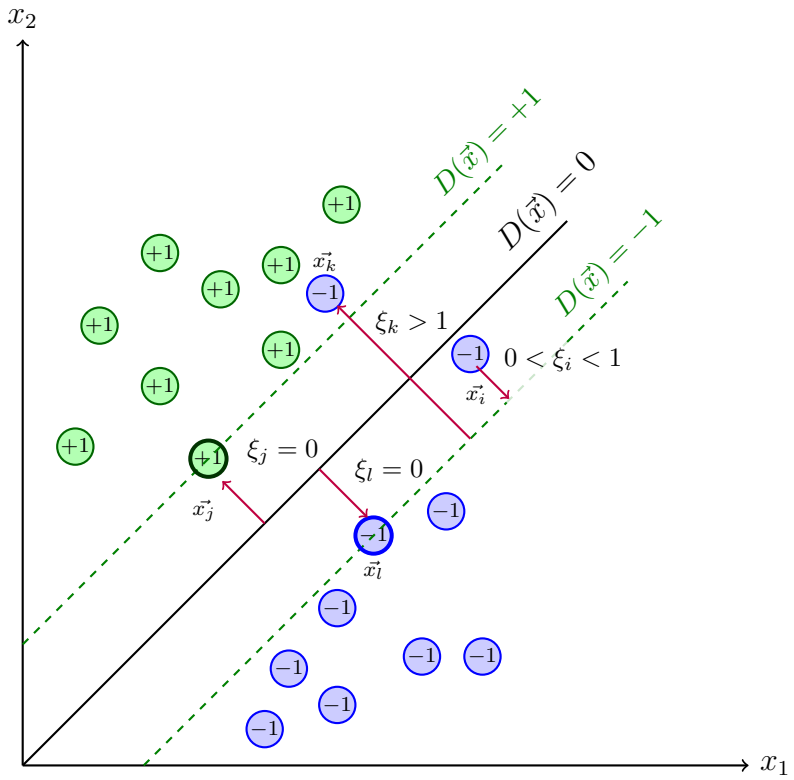


Figura 2.6: Diagrama del SVM de margen blando

sin cruzar la frontera de decisión y los que tienen un valor de ξ mayor a uno incluso acaban mal clasificados al otro lado del hiperplano. Por último, los puntos que tienen holgura $\xi = 1$ están exactamente sobre el hiperplano de decisión, por lo que están bien clasificados.

Para resolver este problema, añadimos a la función a optimizar definida en (2.10) el término $\sum_{i=1}^{n} \xi_i$ para penalizar tanto las invasiones suaves del margen como los errores completos de clasificación. Así, la nueva función objetivo será:

$$f(\vec{w}, \xi) = \frac{1}{2}||\vec{w}||^2 + \mathcal{R} \sum_{i=1}^{n} \xi_i \tag{2.13}$$

donde $\mathcal{R}$ es el parámetro de regularización que es elegido por el usuario. Este parámetro determina cuánto influye la penalización de los datos no separables en la función objetivo, es decir, ajusta el equilibrio entre la tendencia al sobreajuste del clasificador final y la tasa de instancias clasificadas de manera incorrecta, [28]. De esta manera, cuando $\mathcal{R}$ es muy grande, las variables de holgura, ξ_i tienden a 0, por lo tanto el margen óptimo se estrecha y el modelo tiene alto riesgo de sobreajuste. En cambio, valores más pequeños de $\mathcal{R}$ permiten que el margen se ensanche ya que se admiten más puntos dentro de éste que pueden estar mal clasificados.

Por lo tanto, el problema de optimización denominado SVM de margen blando (*soft-margin SVM*) es el siguiente:

$$\begin{aligned} \min_{\vec{w},b_0,\xi_i,i=1,\ldots,n} \quad & \frac{1}{2}\|\vec{w}\|^2 + \mathcal{R}\sum_{i=1}^{n}\xi_i \\ \text{s.a} \quad & y_i\big(\langle \vec{w}, \vec{x_i}\rangle + b_0\big) \;\geq\; 1-\xi_i, \quad i=1,\ldots,n \\ & \xi_i \geq 0, \quad i=1,\ldots,n \end{aligned} \tag{2.14}$$

Al igual que en la sección anterior, nos encontramos ante un problema convexo porque minimiza una función convexa (cuadrática en $\vec{w}$ y lineal en ξ) sobre un conjunto factible definido por restricciones lineales. Este problema se resuelve con optimización convexa.

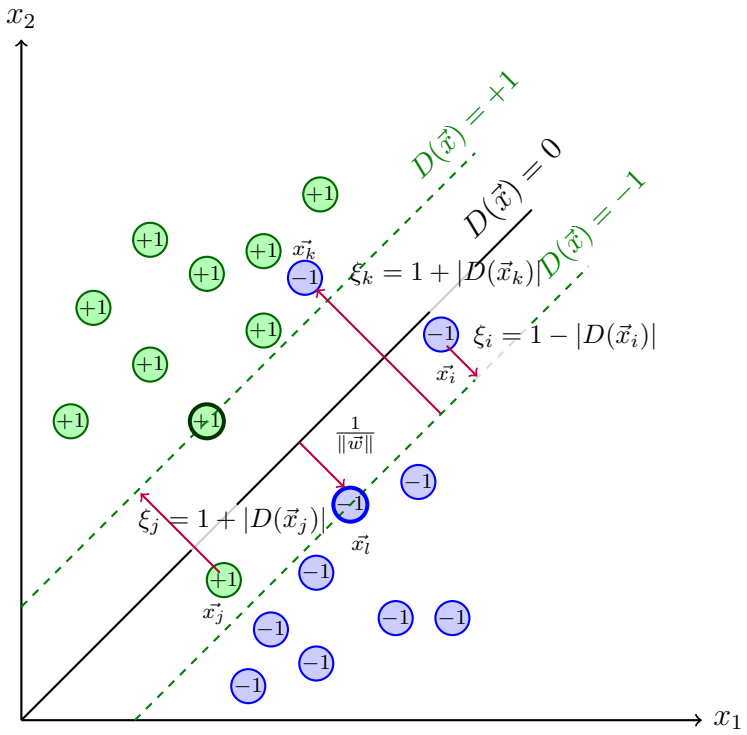


Figura 2.7: Diagrama del SVM de margen blando

En la Figura 2.7 se muestra un escenario más detallado del SVM de margen blando. Al igual que en la Figura 2.6, aparecen ejemplos con $\xi = 0$ (vectores soporte como $\vec{x}_l$), con $0 < \xi < 1$ (como $\vec{x}_i$) y con $\xi > 1$ (como $\vec{x}_j$ y $\vec{x}_k$). Sin embargo, en esta figura se destacan explícitamente las expresiones de las holguras en función del hiperplano óptimo, por ejemplo, $\xi_i = 1 - |D(\vec{x}_i)|$ cuando el punto está dentro del margen, o $\xi_k = 1 + |D(\vec{x}_k)|$ cuando está mal clasificado. Además, se representa gráficamente la distancia entre un vector soporte y el hiperplano como $\frac{1}{\|\vec{w}\|}$.

Al igual que en el problema de la Sección 2.1 la solución óptima a este problema (2.16) viene dada por $\vec{w}^*$ y b_0^*, por lo tanto, se obtiene el hiperplano óptimo:

$$D(\vec{x}) = \langle \vec{w}^*, \vec{x}\rangle + b_0^* \tag{2.15}$$

Al igual que en el problema de margen duro, la función de predicción es

$$\hat{y}(\vec{x}) = sign(D(\vec{x})) \tag{2.16}$$

En consecuencia, si queremos clasificar un dato nuevo, $\vec{x}^{test}$, lo evaluamos en el hiperplano (2.15) y usamos la regla de predicción (2.16).
Así, si $D(\vec{x}^{test}) \geq 0$, entonces $\hat{y}(\vec{x}^{test}) = 1$. Sin embargo, si $D(\vec{x}^{test}) < 0$, se tiene que $\hat{y}(\vec{x}^{test}) = -1$.

Capítulo 3

Árboles de decisión

En este capítulo describiremos un nuevo modelo de aprendizaje supervisado para resolver los problemas de clasificación binaria: los árboles de decisión.
La idea general de los árboles de decisión es dividir la región inicial de los datos en subregiones, de forma que cada subregión contenga el mayor número posible de puntos de una misma clase.

Ejemplo 3.1. A modo ilustrativo, supongamos que tenemos los datos de la Figura 3.1.

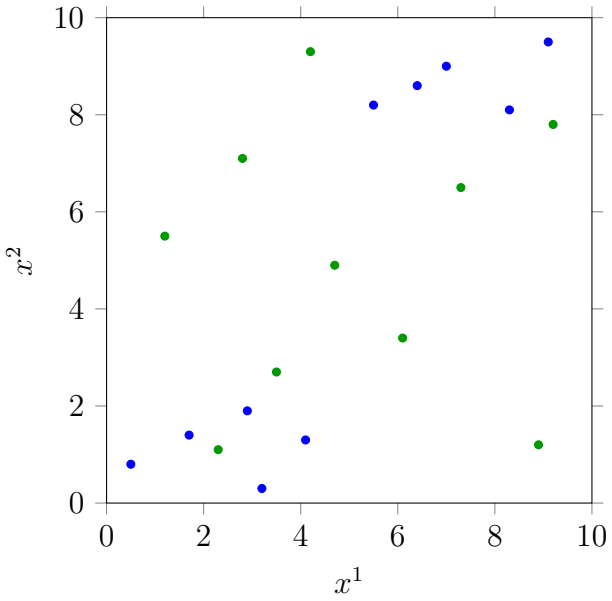


Figura 3.1: Datos de entrenamiento en el espacio de características (x^1, x^2).

Observamos que los datos tienen dos variables de entrada (x^1, x^2). Los círculos azules representan la clase -1 y los círculos verdes tienen etiqueta +1. Es decir, en el primer caso la variable de salida es $y = -1$ y en el segundo tenemos $y = +1$.
La Figura 3.2 representa la salida de un árbol de decisión que divide la región inicial en cuatro subregiones, asignada a cada una la clase 1 o -1.

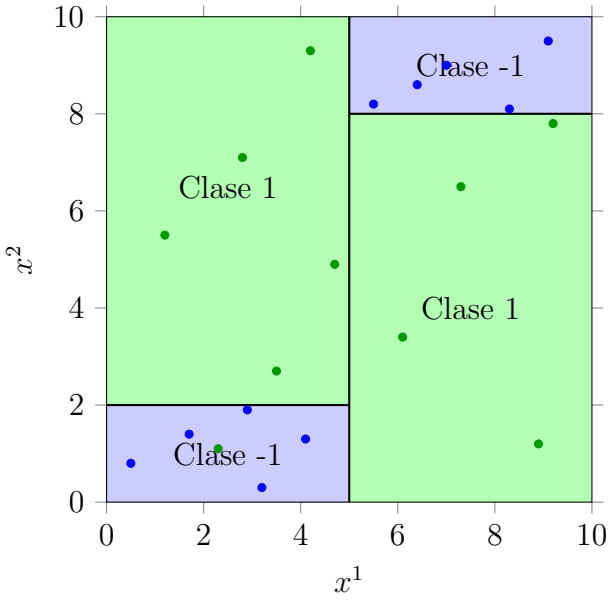


Figura 3.2: División del espacio de características en cuatro regiones según los umbrales en x^2 y x^1.

Los árboles de decisión son conocidos por su gran interpretabilidad: su representación en forma de reglas de decisión sencillas permite a cualquier usuario, incluso sin formación técnica, comprender con facilidad cómo se llega a cada predicción. Su popularidad también radica en la capacidad de cuantificar la relevancia de cada variable de entrada sobre la variable de salida, lo que facilita detectar y descartar características irrelevantes. Además, a diferencia del SVM, son capaces de trabajar directamente con variables tanto categóricas como continuas, sin necesidad de transformaciones previas, [12].
La idea original de este modelo se remonta al trabajo de Hoveland y Hunt a finales de la década de 1950, [20], y culmina en el libro pionero *Experiments in Induction*, [22], que describe experimentos extensos con varias implementaciones de modelos de aprendizaje. Otros investigadores han llegado de forma independiente a métodos iguales o similares; en particular, Breiman y Friedman sientan las bases del famoso sistema CART (del inglés *Classification and regression trees*), [9].

Estos métodos de aprendizaje se han aplicado con éxito a una amplia variedad de tareas, desde el diagnóstico de casos médicos hasta la evaluación del riesgo crediticio de solicitantes de préstamos, [3; 25]

A lo largo de este capítulo nos centraremos en la descripción tanto de los elementos como del funcionamiento de los árboles de decisión. Además, se tratará en concreto el caso de clasificación binaria, es decir, cuando la variable de salida toma solo dos posibles clases.

3.1. Representación de los árboles de decisión

Previo a abordar formalmente la construcción de un árbol de clasificación, se proporcionarán algunas nociones generales. También se presentan los elementos principales de un árbol.

Un árbol de decisión consta de tres tipos de nodos: nodos raíz, nodos internos y nodos hoja, [2]. A continuación se describen cada uno de ellos:

El **nodo raíz** es aquel que inicia el proceso de partición y representa el conjunto completo de datos ($\mathcal{S}$) antes de aplicar cualquier división, no tiene ramas entrantes y su función es englobar la totalidad del espacio de entrada, [2]. Desde él parten las primeras divisiones que dan lugar a nodos hijos. Las ramas salientes del nodo raíz alimentan a los nodos internos.

Llamamos splits o reglas de división a cada una de las ramas en las que se particiona un nodo.

Un **nodo interno o intermedio** es aquel en el que se encuentran los datos que responden a una pregunta sobre una variable de entrada (por ejemplo, $x^j \leq c$ o $x^j = b$, es decir, ¿es la variable j-ésima menor o igual que un cierto valor c?, o ¿es la variable x^j igual a cierto valor b? A continuación, se generan tantas ramas como resultados posibles tenga la pregunta.

Por último, un **nodo hoja o terminal** es aquel que no posee hijos y en el que se detiene la partición de los datos originales. Los nodos hoja son los nodos del árbol que no tienen nodos adicionales conectados. No dividen los datos más allá; simplemente proporcionan una clasificación para los ejemplos que llegan a ese nodo, es decir, se le asigna una etiqueta de clase. En la siguiente sección se verá cuál es el mecanismo para asignar cada etiqueta.

Los árboles de decisión clasifican una instancia guiándola desde el nodo raíz hasta una hoja, donde se determina su etiqueta. En cada nodo se evalúa un atributo de la instancia, y la rama que se sigue corresponde al valor que éste toma. Así, tras comprobar el atributo en el nodo actual, se desciende por la rama adecuada y se repite el mismo procedimiento en el siguiente nodo, hasta alcanzar una hoja que proporciona la clasificación final, [27].

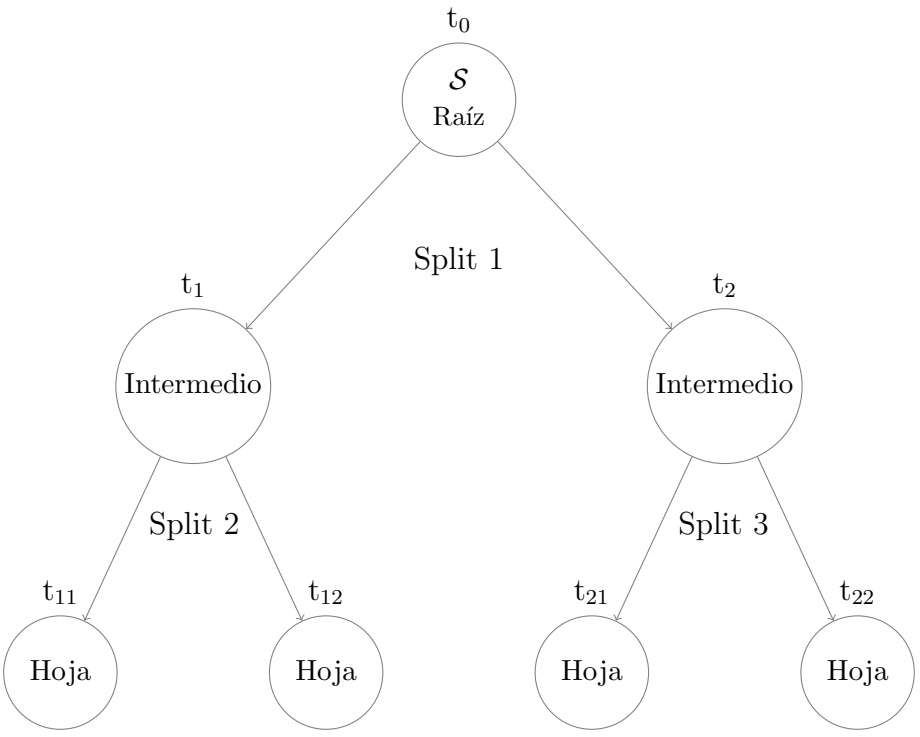


Figura 3.3: Árbol de decisión con etiquetas de división por nivel.

En la Figura 3.3 se muestra un árbol de decisión genérico: partiendo del nodo raíz ($\mathcal{S}$),

pasando por los nodos intermedios (t_1, t_2) donde ocurren nuevos splits, y finalizando en los nodos hoja (t_{11}, t_{12}, t_{21} y t_{22}) que proporcionan la clasificación final. Por construcción, las regiones que determinan los nodos hoja son disjuntas ya que en cada división, los datos se particionan de forma que cada subconjunto resultante recibe una parte distinta del dominio original; de modo que, al llegar a una hoja, aquel ejemplar ha seguido un camino único de pruebas y queda asignado a una sola región.

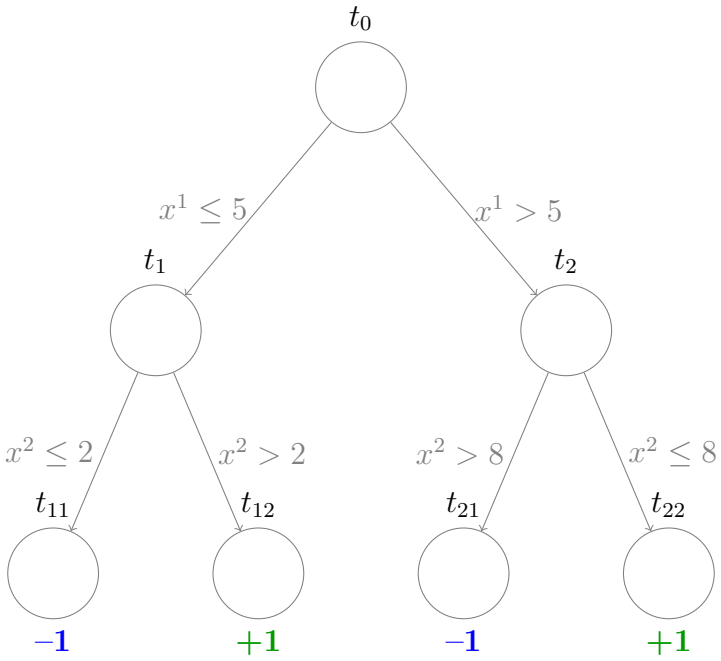


Figura 3.4: Árbol de decisión con reglas de partición sobre x^1 y x^2.

Volviendo a los datos del Ejemplo 3.1, en la Figura 3.2 podemos observar el árbol resultante tras entrenar el modelo, cuyos resultados finales además coinciden con las regiones que quedan determinadas en la Figura 3.2. En primer lugar, la Figura 3.4 muestra que el espacio de características se divide por la línea vertical $x^1 = 5$, lo que corresponde a la prueba del nodo raíz del árbol de la Figura 3.4. A continuación, en la región donde $x^1 > 5$ se aplica el umbral $x^2 = 8$ que separa los datos de clase -1 ($x^2 > 8$) de los de clase 1 ($x^2 \leq 8$), mientras que en la región donde $x^2 \leq 5$ se emplea el umbral $x^2 = 2$ para distinguir los de clase 1 ($x^2 > 2$) de clase -1 ($x^2 \leq 2$). Cada hoja del árbol coincide así exactamente con una de las cuatro regiones coloreadas en la Figura 3.2.

Supongamos ahora que queremos hacer una predicción sobre la clase de un nuevo punto no usado previamente en el entrenamiento, por ejemplo, $(x^1, x^2) = (6, 3)$. Para ello, seguimos el recorrido del árbol de decisión de la Figura 3.4: en la raíz comprobamos si $x^1 > 5$, lo cual es verdadero pues $6 > 5$, por lo que descendemos por la rama t_2. Evaluamos a continuación si $x^2 > 8$; dado que $3 \leq 8$, seguimos la rama t_{22} y llegamos a la hoja etiquetada como clase 1. Por tanto, el clasificador asigna al punto $(6, 3)$ la clase 1.

3.2. Construcción de los árboles de decisión

En esta sección describimos las cuatro componentes clave en la construcción de un árbol de decisión: la topología y el tipo de división, los criterios de parada, la regla de asignación de etiquetas y los criterios de división.

3.2.1. Topología y tipo de división

La **topología** de un árbol indica cuántas ramas pueden surgir en cada división. Habitualmente se utilizan árboles binarios, en los que cada nodo interno genera dos hijos, aunque en algunos casos puede optarse por divisiones múltiples. Para ello, separaremos los casos en los que las variables son categóricas, por ejemplo, el color de ojos, y los casos en los que sean cuantitativas, como la altura de una persona, [12].

En primer lugar, asumimos dada una variable categórica x^j con Q categorías $c_1, \ldots, c_Q$. Por ejemplo, el color de ojos de una persona que tiene por categorías c_1 = "azul", c_2="verde" y c_3="marrón". Las divisiones binarias pueden aislar una única categoría c_q para algún q:

$$\begin{cases} x^j = c_q \\ x^j \neq c_q \end{cases}$$

donde en nuestro caso, si tomamos por ejemplo, $q = 1$, es decir, c_1 = "azul", tenemos:

$$\begin{cases} x^j = \text{“azul”} \\ x^j \neq \text{“azul”} \end{cases}$$

Además, también podemos agrupar subconjuntos de categorías, $J \subset \{c_1, \ldots, c_Q\}$, y dar lugar a divisiones binarias

$$\begin{cases} x^j \in J \\ x^j \notin J \end{cases}$$

Por ejemplo, $J = \{$"verde", "marrón"$\}$:

$$\begin{cases} x^j \in \{\text{verde, marrón}\} \\ x^j \notin \{\text{verde, marrón}\} \end{cases}$$

En algunos casos, especialmente cuando la variable es categórica con muchas modalidades, puede ser conveniente realizar **divisiones múltiples**, en lugar de binarias. En este caso, cada categoría se asigna directamente a una rama hija distinta. Así, si x^j es una variable con tres categorías $\{c_1, c_2, c_3\}$, el nodo puede dividirse en tres nodos hijos, uno para cada valor. Este tipo de partición da lugar a árboles no binarios. Por ejemplo,

$$\begin{cases} x^j = c_1 \\ x^j = c_2 \\ x^j = c_3 \end{cases} \longrightarrow \begin{cases} x^j = \text{“azul”} \\ x^j = \text{“marrón”} \\ x^j = \text{“verde”} \end{cases}$$

Por otro lado, cuando se tiene una variable cuantitativa x^j, las divisiones binarias se basan en puntos de corte, a. Siguiendo el ejemplo mencionado anteriormente, si la variable

es la altura de una persona, la división binaria es de la forma siguiente:

$$\begin{cases} x^j \leq a \\ x^j > a \end{cases}$$

donde a suele situarse a mitad de distancia entre dos valores consecutivos en la muestra. Si tomamos, $a = 1.60$ m de altura, tenemos

$$\begin{cases} x^j \leq 1.60 \\ x^j > 1.60 \end{cases}$$

Para seleccionar de forma genérica los posibles umbrales de división en un atributo numérico x a partir del conjunto de entrenamiento, se procede así:

1. Sea $\{v_1, \ldots, v_q\}$ el conjunto de valores distintos de la variable de entrada x^j que aparecen en el nodo, ordenados de forma creciente ($v_1 < v_2 < \cdots < v_q$).

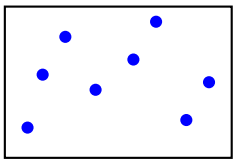
2. Como candidatos de la regla de decisión tomamos los puntos medios

$$s_k = \frac{v_k + v_{k+1}}{2}, \qquad k = 1, \ldots, q-1.$$

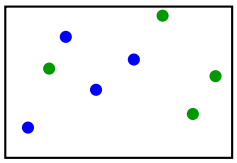
3. Opcionalmente, para reducir el tiempo computacional, sólo se conservan aquellos s_k para los que las etiquetas de clase cambian entre los ejemplos con valor v_k y los de valor v_{k+1}.

3.2.2. Criterio de división

Los algoritmos de árboles de decisión necesitan un criterio para dividir los nodos y formar el árbol. Para decidir cuál es la mejor regla de división en un nodo raíz t, nos basamos en el concepto de impureza. El objetivo es encontrar un corte de forma que los nodos hoja resultantes sean lo más "puros" posible (o su homogeneidad sea máxima), es decir, que haya el máximo número de puntos de alguna de las clases.



(a) Nodo puro



(b) Nodo impuro

Figura 3.5: Ejemplos de nodo puro vs. nodo impuro en árboles de decisión

En las Figuras 3.5a y 3.5b se ilustran dos tipos de nodos en un árbol de decisión: mientras que en el nodo puro (Figura 3.5a) todos los puntos dentro del rectángulo comparten la misma etiqueta de clase (puntos azules), en el nodo impuro (Figura 3.5b) coexisten ejemplares de ambas clases (puntos azules y verdes), mostrando visualmente cómo la pureza de un nodo disminuye cuando la mezcla de clases aumenta. En esta sección estudiaremos las medidas de impureza más conocidas.

Antes de ello, debemos definir algunos conceptos. Nótese que, dado que en este trabajo nos estamos centrando en la clasificación binaria, escribiremos las definiciones en base a dos clases, aunque éstas pueden generalizarse de una forma sencilla al caso general con K clases.

Definición 3.1. Asumimos dados n pares $(\vec{x}_i, y_i)$, donde $\vec{x}_i \in \mathbb{R}^p$ es el vector de características del dato i. Sea $n_k(t)$ el número de datos de la clase C_k con $k = 1, 2$ que han llegado al nodo t, y $n(t)$ el número total de datos en ese mismo nodo. Se define la probabilidad de la clase C_k dado un nodo t como:

$$p(C_k \,|\, t) = \frac{n_k(t)}{n(t)} \tag{3.1}$$

Obsérvese que la Ecuación (3.1) es la versión frecuentista de la probabilidad.

Volviendo a la Figura 3.5 observamos que, en el nodo puro de la Figura 3.5a, tenemos $n(t) = 8$ y todos los puntos pertenecen a la misma clase, digamos la clase 1, de modo que

$$n_1(t) = 8, \quad n_{-1}(t) = 0 \quad \Longrightarrow \quad p(1 \,|\, t) = \frac{8}{8} = 1, \;\; p(-1 \,|\, t) = \frac{0}{8} = 0.$$

En cambio, en el nodo impuro de la Figura 3.5b, con un total de $n(t) = 8$ puntos repartidos a partes iguales entre las clases 1 y -1, tenemos

$$n_1(t) = 4, \quad n_{-1}(t) = 4 \quad \Longrightarrow \quad p(1 \,|\, t) = \frac{4}{8} = 0.5, \quad p(-1 \,|\, t) = \frac{4}{8} = 0.5.$$

Definición 3.2. Sea $p = (p_1, p_2)$ el vector de proporciones correspondiente a las dos clases posibles en un nodo t, tal como se ha visto en la Definición 3.1. Es decir, $p_k = p(C_k \,|\, t)$ con $k = 1, 2$ representa la probabilidad de pertenecer a la clase C_k en dicho nodo. Una *función de impureza* ϕ debe satisfacer las siguientes propiedades, [9]:

- $\phi(p)$ es convexa en p,
- $\phi(p)$ alcanza su valor mínimo cuando existe k tal que $p_k = 1$,
- $\phi(p)$ alcanza su valor máximo cuando $p_1 = p_2 = \frac{1}{2}$.

Definición 3.3. Sea ϕ una función de impureza. Una medida de impureza $i(t)$ de un nodo t se define como

$$i(t) = \phi(p(C_1 \,|\, t), p(C_2 \,|\, t)) \tag{3.2}$$

Definición 3.4. Sea s una regla de división (*split*) que particiona un nodo t en R ramas que dan lugar a R nodos $\{t_1, \ldots, t_R\}$. Sea i una medida de impureza. Se define la ganancia de información de la regla de división s en el nodo t como la diferencia entre la impureza del nodo antes de la partición (nodo padre) y la impureza media del nodo después de la partición (nodo hijo):

$$G(s, t) = i(t) - \sum_{r=1}^{R} \frac{n(t_r)}{n(t)} i(t_r) \tag{3.3}$$

donde $n(t)$ corresponde al número de observaciones en el nodo padre y $n(t_r)$ en el nodo hijo. Obsérvese que el primer término corresponde a la impureza inicial, mientras que el segundo es la impureza media tras el split.

A continuación, se expondrán las medidas de impureza más conocidas y, posteriormente, se ilustrará un ejemplo de su cálculo, [12]:

Definición 3.5. Dado un nodo t, con dos posibles clases y donde $n(t)$ es el número total de instancias en t, y $n_k(t)$ el número de instancias de la clase C_k con $k = 1, 2$. Se define la entropía de Shannon como:

$$H(t) = -\sum_{k=1}^{2} p(C_k \,|\, t) \log_2 p(C_k \,|\, t) \tag{3.4}$$

La entropía de Shannon en un nodo t cuantifica el grado de incertidumbre o mezcla de clases en dicho nodo. Cuando el nodo es puro (todas las instancias pertenecen a una única clase), entonces existe una clase k tal que $p(C_k \,|\, t) = 1$ y $p(C_d \,|\, t) = 0, \forall d \neq k$, por lo que $H(t) = 0$. Por el contrario, si las 2 clases son equiprobables, es decir, $p(C_k \,|\, t) = \frac{1}{2}$ para todo $k = 1, 2$, la entropía alcanza su valor máximo ($H(t) = \log_2 2$). Por tanto, un nodo con mayor entropía es más "impuro" y contiene mayor mezcla de clases, mientras que un nodo con entropía baja es más "puro" y tiene menos mezcla de etiquetas.

En la práctica, se selecciona el punto de corte que maximiza la ganancia de información, es decir, aquel que produce la mayor reducción en la impureza tras la división del nodo.

Definición 3.6. Sea un nodo t, con dos clases posibles y donde $p(C_k \,|\, t)$ para $k = 1, 2$ representa la proporción de instancias de la clase C_k en dicho nodo. El *índice de Gini* cuantifica la probabilidad de clasificar erróneamente un punto escogido al azar, y se define como:

$$Gini(t) = 1 - \sum_{k=1}^{2} p(C_k \,|\, t)^2 \tag{3.5}$$

Definición 3.7. Sea un nodo t, con dos clases posibles y donde $p(C_k \,|\, t)$ para $k = 1, 2$ representa la probabilidad de que un dato pertenezca a la clase C_k en dicho nodo. La *tasa de error de clasificación* mide la proporción de datos que no pertenecen a la clase mayoritaria, y se define como:

$$Error(t) = 1 - \max_{1 \leq k \leq 2} p(C_k \,|\, t) \tag{3.6}$$

Ejemplo 3.2. A continuación, veamos con un ejemplo, cómo se calculan las tres medidas de impureza de las Definiciones 3.5, 3.6 y 3.7. Supongamos que un atributo tiene tres decisiones posibles y la etiqueta de clase es binaria. Es decir, se tiene un árbol como el de la Figura 3.6, en el que se muestra el número de elementos de cada clase, donde la primera y segunda coordenada equivalen al número de datos pertenecientes a la clase "No" y "Si", respectivamente. Por ejemplo, en el nodo t_1 no hay datos de la clase "No" y los seis elementos restantes pertenecen a la clase "Sí".

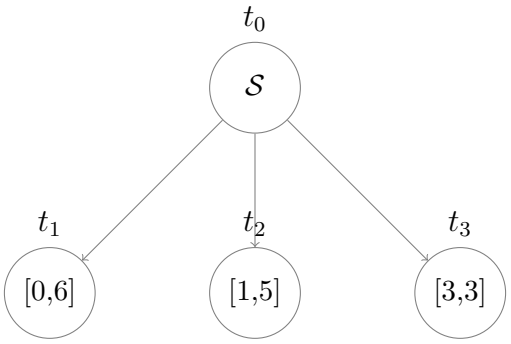


Figura 3.6: Árbol de decisión con división múltiple del conjunto $\mathcal{S}$.

Entonces, la entropía de Shannon, el índice de Gini y la tasa de error se calculan de acuerdo a las Definiciones 3.5, 3.6 y 3.7 cuyos detalles se muestran en la Tabla 3.1.

Medida de impureza	**Nodo t_1**	**Nodo t_2**	**Nodo t_3**
Entropía	$-(0/6)\log_2(0/6) - (6/6)\log_2(6/6) = 0$	$-(1/6)\log_2(1/6) - (5/6)\log_2(5/6) = 0.650$	$-(3/6)\log_2(3/6) - (3/6)\log_2(3/6) = 1$
Índice de Gini	$1-(0/6)^2-(6/6)^2 = 0$	$1-(1/6)^2-(5/6)^2 = 0.278$	$1-(3/6)^2-(3/6)^2 = 0.5$
Error de clasificación	$1-\text{máx}\{0/6, 6/6\} = 0$	$1-\text{máx}\{1/6, 5/6\} = 0.167$	$1-\text{máx}\{3/6, 3/6\} = 0.5$

Tabla 3.1: Cálculo de medidas de impureza

En primer lugar, en el nodo t_1 todos los ejemplos son "Sí", por lo que la entropía es 0, el índice de Gini es 0 y el error de clasificación es 0, reflejando pureza total sin ninguna mezcla de clases.

Por otro lado, para t_2 con un caso "No" frente a cinco "Sí", la entropía sube a aproximadamente 0.65, el índice de Gini a unos 0.278 y el error a 0.167, lo que indica cierta impureza pero una clara dominancia de la clase "Sí" que se refleja numéricamente en que las tres medidas de impureza toman valores bajos pero no nulos, lo que indica que, aunque hay algo de mezcla, la mayoría de los ejemplos (5 de 6) pertenecen a una única clase.

Por último, en el nodo t_3 al tener tres "No" y tres "Sí", la entropía alcanza 1, el índice de Gini y el error valen 0.5, señalando el máximo desorden e incertidumbre cuando las clases están exactamente equilibradas.

3.2.3. Criterios de parada

El siguiente paso es decidir cuándo declarar un nodo como terminal y evitar el problema del *sobreajuste*, que ocurre cuando el árbol aprende demasiado de la muestra de entrenamiento, ajustándose mucho a los datos de entrenamiento, perdiendo la capacidad de generalización.

Para impedir este crecimiento excesivo y garantizar un árbol con buen poder predictivo, se emplean habitualmente varios criterios de parada:

- Se alcanza una profundidad máxima, $p_{\text{máx}}$, predefinida.
- Si
$$\max_{s \in S} G(s,t) < \alpha$$
no se divide el nodo t, donde $G(s,t)$ es la ganancia de información obtenida al partir el nodo t según la partición s, S es el conjunto de posibles splits y α es un umbral preestablecido por el usuario.
- El número de observaciones en un nodo terminal es menor que un $m_{\text{mín}}$ dado.

3.2.4. Regla de asignación en nodos terminales

Una vez que un nodo deja de dividirse mediante uno de los criterios vistos en la sección anterior, se le asigna la clase mayoritaria de sus observaciones. Formalmente, si $n_k(t)$ es el número de ejemplos de la clase C_k con $k = 1, 2$ en el nodo t, la etiqueta asignada al nodo es:

$$\hat{y}(t) \;=\; \arg\max_{k=1,2} \; n_k(t) \tag{3.7}$$

En caso de empate, puede aplicarse un voto ponderado o elegir aleatoriamente una de las clases donde se ha producido el empate.

3.3. Entrenamiento de un árbol de decisión

A continuación, veremos cómo se produce el entrenamiento de un árbol de decisión de acuerdo al algoritmo CART (*Classification and Regression Trees*), [12].

3.3.1. Algoritmo CART de clasificación

El algoritmo CART es un algoritmo heurístico voraz (*greedy*) donde en cada paso local se elige la mejor decisión hasta el momento con la esperanza de llegar al óptimo global (aunque esto no siempre se pueda conseguir, precisamente por la naturaleza heurística de la metodología). En particular, el algoritmo CART se basa en particiones recursivas del espacio de características. En el caso de clasificación, su objetivo es dividir sistemáticamente el conjunto de datos en nodos cada vez más puros, mediante la selección de la variable y el umbral de corte que maximicen la reducción de impureza. Según Breiman *et al.*, [9], la construcción de un árbol de clasificación sigue estos pasos:

Algoritmo 1 Construcción arriba-abajo de un árbol de clasificación

Entrada: Conjunto de entrenamiento $\mathcal{S}$, tipo de partición, criterio de división, criterio de parada.

Salida: Árbol de clasificación (conjunto de nodos terminales P_{final}).

```
P := {S}
P_final := ∅
while P ≠ ∅ do
  for all t ∈ P do
    if el criterio de parada se satisface en t then
      P = P \ {t}
      P_final = P_final ∪ {t}
    end if
    Calcular la ganancia de información en t sobre cada variable predictora según la
    topología del árbol.
  end for
  Seleccionar la partición con ganancia de información máxima.
  Hacer un corte en el nodo t según la partición elegida, obteniendo los nodos
  {t_1, ..., t_R}.
  P = (P \ {t}) ∪ {t_1, ..., t_R}
end while
Etiquetar los nodos en P_final según la Expresión (3.7).
```

Según el Algoritmo 1, en primer lugar se define el nodo raíz con el conjunto de entrenamiento $\mathcal{S}$, el tipo de partición, el criterio de división y el de parada. En la fase de inicialización, se establece la variable P como una lista inicial de nodos no terminales, que en principio contendrá solo al nodo raíz, además de P_{final} que se inicializa con el vacío y posteriormente se irán añadiendo los nodos terminales, es decir, aquellos que cumplan el criterio de parada.

A continuación, mientras P no esté vacío, para cada nodo $t \in P$ si se cumple el criterio de parada establecido, entonces dicho nodo se elimina de P y se añade a P_{final}; en otro caso se calcula la ganancia de información para cada posible corte s y se selecciona el par (s^*, t^*) que maximiza $G(s, t)$. El nodo óptimo t^* se divide según s^*, generando hijos $\{t_1, \ldots, t_R\}$, que se añaden al conjunto P, mientras que el nodo t^* se elimina de dicho conjunto. El proceso itera hasta que P queda vacío; finalmente, cada hoja $t \in P_{\text{final}}$ recibe la etiqueta de la clase mayoritaria de acuerdo con la Ecuación (3.7).

Tras la construcción del árbol puede realizarse una poda para mejorar la generalización, en este trabajo no entraremos en más detalle sobre este procedimiento. Para más información ver el capítulo 8 del libro [9].

A continuación se presenta un ejemplo ilustrativo del funcionamiento del algoritmo CART aplicado a un problema de clasificación binaria.

Ejemplo 3.3. En muchas organizaciones, la decisión de contratar a un nuevo empleado plantea un difícil dilema que combina factores tanto cuantitativos como cualitativos. Este problema puede plantearse como una tarea de clasificación binaria. Para ello, consideramos un conjunto de características del trabajador, la experiencia en años que tiene y el

nivel de certificaciones que posee: Baja, Media o Alta. Se debe predecir si corresponde añadirlo a la plantilla o no. Para ello, usaremos el algoritmo visto en esta sección.

Por un lado, en la Tabla 3.2 se tiene el conjunto de entrenamiento formado por 11 candidatos. La primera columna indica el identificador de dichas personas. Las columnas 2 y 3 indican la experiencia de la persona en años y el nivel de certificación de estos, respectivamente. Finalmente, la última columna indica si dicha persona ha sido etiquetada con la clase C_1= "Sí" o C_2 = "No" .

Candidato	Experiencia (en años)	Nivel de Certificación	Contratado
1	1	Bajo	No
2	2	Medio	No
3	3	Medio	Sí
4	4	Bajo	Sí
5	5	Alto	Sí
6	6	Alto	No
7	7	Medio	Sí
8	8	Alto	Sí
9	9	Bajo	No
10	10	Medio	No
11	11	Alto	Sí

Tabla 3.2: Conjunto de entrenamiento

En primer lugar, se elige el criterio de división, para la variable continua "Experiencia", elegiremos una división binaria. Para la variable categórica "Nivel de certificación" será una partición múltiple. La medida de impureza que se considerará es la entropía de Shannon y el criterio de parada consistirá en que un nodo tenga menos de 5 observaciones.

Una vez establecidos estos requisitos, comenzamos con los pasos del algoritmo.

En el nodo raíz hay 6 datos con etiqueta "Sí" y 5 con etiqueta "No", por lo que la entropía de Shannon es:

$$i(t_0) \;=\; -\frac{6}{11}\log_2\!\left(\tfrac{6}{11}\right) - \frac{5}{11}\log_2\!\left(\tfrac{5}{11}\right) \;\approx\; 0.99.$$

A continuación, se calcula la ganancia de información para cada división (*split*) posible. Esos umbrales surgen directamente de los valores de la variable "Experiencia" en el conjunto de entrenamiento: primero se ordenan los datos únicos de esa variable y, a continuación, se proponen como candidatos de split los puntos medios entre cada par de valores consecutivos (por ejemplo, $(2+3)/2 = 2.5$, $(5+6)/2 = 5.5$, etc.). Para cada uno de estos umbrales se calcula la ganancia de información, o reducción de impureza, que produce al dividir el nodo en dos grupos, y se selecciona como corte óptimo aquel que maximiza dicha ganancia. De este modo, los umbrales $\{2.5, 5.5, 6.5, 8.5, 10.5\}$ de la Tabla 3.3 corresponden exactamente a los puntos medios entre los valores consecutivos de "Experiencia" presentes en los datos.

En la Tabla 3.3 se muestra para la variable continua "Experiencia" la siguiente información observadas las columnas de izquierda a derecha: cada punto de corte posible

Experiencia							
Punto de corte	**Particiones**	$\mathbf{q_i}$	$\mathbf{p}(\text{No} \mid \mathbf{t_i})$	$\mathbf{p}(\text{Sí} \mid \text{t}_\text{i})$	$\mathbf{i(t_i)}$	$\sum_j q_j\, i(t_j)$	$\mathbf{G(t_0, s)}$
2.5	Experiencia ≤ 2.5	2/11	2/2	0/2	0.00	0.75	0.24
	Experiencia > 2.5	9/11	3/9	6/9	0.918		
5.5	Experiencia ≤ 5.5	5/11	2/5	3/5	0.97	0.99	0.00
	Experiencia > 5.5	6/11	3/6	3/6	1.00		
6.5	Experiencia ≤ 6.5	6/11	3/6	3/6	1.00	0.99	0.00
	Experiencia > 6.5	5/11	2/5	3/5	0.97		
8.5	Experiencia ≤ 8.5	8/11	3/8	5/8	0.95	0.95	0.04
	Experiencia > 8.5	3/11	2/3	1/3	0.92		
10.5	Experiencia ≤ 10.5	10/11	5/10	5/10	1.00	0.91	0.08
	Experiencia > 10.5	1/11	0/1	1/1	0.00		

Tabla 3.3: Experiencia. Split 1.

para dicho nodo t_0, la división que genera, el tamaño de los subnodos generados, es decir, el número de observaciones que quedarían en cada nodo hijo si se lleva a cabo ese split y que se denota por q_i, la proporción de ejemplos con etiqueta "No" y "Sí" dentro de la partición t_i. Para estos subnodos se muestra la impureza, la impureza media tras la división y, por último, la ganancia de información.
Como vimos anteriormente, el corte que maximiza esta ganancia es el elegido para la primera partición. Con lo cual, debemos calcular los splits para la variable "Nivel de Certificación" antes de elegir uno.

Nivel de Certificación						
Particiones	$\mathbf{q_i}$	$\mathbf{p}(\text{No} \mid \mathbf{t_i})$	$\mathbf{p}(\text{Sí} \mid \text{t}_\text{i})$	$\mathbf{i(t_i)}$	$\sum_j q_j\, i(t_j)$	$\mathbf{G(t_0, s)}$
Certificación = Baja	3/11	2/3	1/3	0.918	0.91	0.08
Certificación = Media	4/11	2/4	2/4	1.000		
Certificación = Alta	4/11	1/4	3/4	0.811		

Tabla 3.4: Nivel de certificación. Split 1.

En la Tabla 3.4 se muestra q_i, las proporciones de datos con etiqueta "No" y "Sí" y la entropía $i(t_i)$ del subnodo para cada categoría. En las dos últimas columnas, aparecen la impureza media tras el split correspondiente $\sum_j q_j\, i(t_j) = 0.91$ y la ganancia de información $G(t_0, s) = 0.08$, calculadas conjuntamente para todo el split.

Por tanto, como se puede observar, el primer split será "Experiencia" ≤ 2.5 y "Experiencia" > 2.5 ya que su ganancia de información es aproximadamente 0.24 siendo la mayor de todas las calculadas en las Tablas 3.4 y 3.3.

Así, nos quedaría un árbol como el de la Figura 3.7. Del nodo t_0 gracias al corte visto anteriormente se obtienen los nodos t_1 y t_2. Por un lado, el nodo t_1 es un nodo terminal ya que se ha cumplido el criterio de parada, al no tener más de cinco observaciones y además, se ha asignado la etiqueta de clase correspondiente a la clase mayoritaria de ese nodo, que en este caso es "No". Por otro lado, el nodo t_2 consta de los 9 datos restantes.

El siguiente paso es volver a calcular la ganancia de información de cada posible corte en este último nodo y así hacer un segundo split. Hacemos el mismo procedimiento de nuevo:

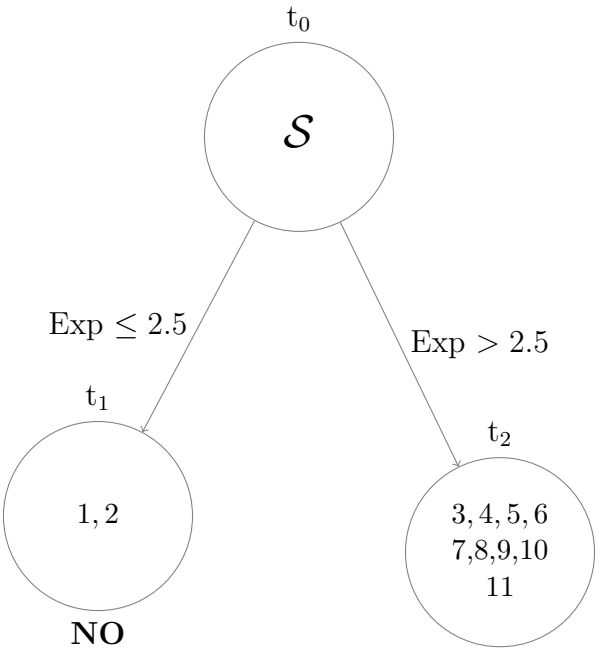


Figura 3.7: Árbol de decisión tras el split 1.

La impureza en t_2 teniendo en cuenta la submuestra en la que ha sido dividido el conjunto de entrenamiento tras el primer corte (Tabla 3.5) es la siguiente:

$$i(t_2) = -\tfrac{5}{9}\log_2(\tfrac{5}{9}) - \tfrac{4}{9}\log_2(\tfrac{4}{9}) \approx 0.918.$$

Candidato	Experiencia	Nivel de Certificación	Contratado
3	3	Medio	Sí
4	4	Bajo	Sí
5	5	Alto	Sí
6	6	Alto	No
7	7	Medio	Sí
8	8	Alto	Sí
9	9	Bajo	No
10	10	Medio	No
11	11	Alto	Sí

Tabla 3.5: Submuestra en t_2 (Experiencia > 2.5)

La Tabla 3.5 muestra las 9 instancias que quedan en el subnodo t_2 tras el primer split. Se usa como punto de partida para calcular las nuevas ganancias en el siguiente nivel.

A continuación, se muestran los datos calculados anteriormente para cada punto de corte posible en el nodo t_2. Como se puede observar en la Tabla 3.6 el mayor valor de la ganancia de información para la variable "Nivel de experiencia" es 0.251. Por otro lado, la Tabla 3.7 muestra un valor de la ganancia de información de 0.251. Debido a que ambos valores son iguales, elegimos, por ejemplo, la división múltiple de la variable "Nivel de certificación".

Experiencia. Split 2							
Punto de Corte	**Particiones**	$\mathbf{q_i}$	$\mathbf{p}(\text{No} \mid \mathbf{t_i})$	$\mathbf{p}(\text{Sí} \mid \text{t}_\text{i})$	$\mathbf{i}(\mathbf{t_i})$	$\sum_j q_j\, i(t_j)$	$\mathbf{G}(\mathbf{t_2}, \mathbf{s})$
5.5	Experiencia ≤ 5.5	3/9	0/3	3/3	0.000	0.667	0.251
	Experiencia > 5.5	6/9	3/6	3/6	1.000		
6.5	Experiencia ≤ 6.5	4/9	1/4	3/4	0.811	0.900	0.018
	Experiencia > 6.5	5/9	2/5	3/5	0.971		
8.5	Experiencia ≤ 8.5	6/9	1/6	5/6	0.650	0.739	0.179
	Experiencia > 8.5	3/9	2/3	1/3	0.918		
10.5	Experiencia ≤ 10.5	8/9	3/8	5/8	0.954	0.844	0.074
	Experiencia > 10.5	1/9	0/1	1/1	0.000		

Tabla 3.6: Nivel de Experiencia. Split 2 (nodo t_2).

Nivel de Certificación. Split 2						
Particiones	$\mathbf{q_i}$	$\mathbf{p}(\text{No} \mid \mathbf{t_i})$	$\mathbf{p}(\text{Sí} \mid \text{t}_\text{i})$	$\mathbf{i}(\mathbf{t_i})$	$\sum_j q_j\, i(t_j)$	$\mathbf{G}(\mathbf{t_2}, \mathbf{s})$
Certificación = Baja	2/9	0/2	2/2	0.00	0.667	0.251
Certificación = Media	3/9	2/3	1/3	0.918		
Certificación = Alta	4/9	2/4	2/4	1.000		

Tabla 3.7: Nivel de Certificación. Split 2

Por lo tanto, se alcanza el criterio de parada establecido, ya que en todos los nodos restantes hay menos de 5 observaciones. La Figura 3.8 muestra el último corte que es una división múltiple según el nivel de experiencia del trabajador. Por último, se han etiquetado los nodos terminales t_{21}, t_{22}, t_{23} según la clase predominante en cada uno.

En conclusión, si un trabajador tiene menos de dos años y medio de experiencia o más de dos años y medio de experiencia pero su nivel de certificaciones es bajo o medio, entonces no será contratado, mientras que en otro caso será contratado por la empresa.

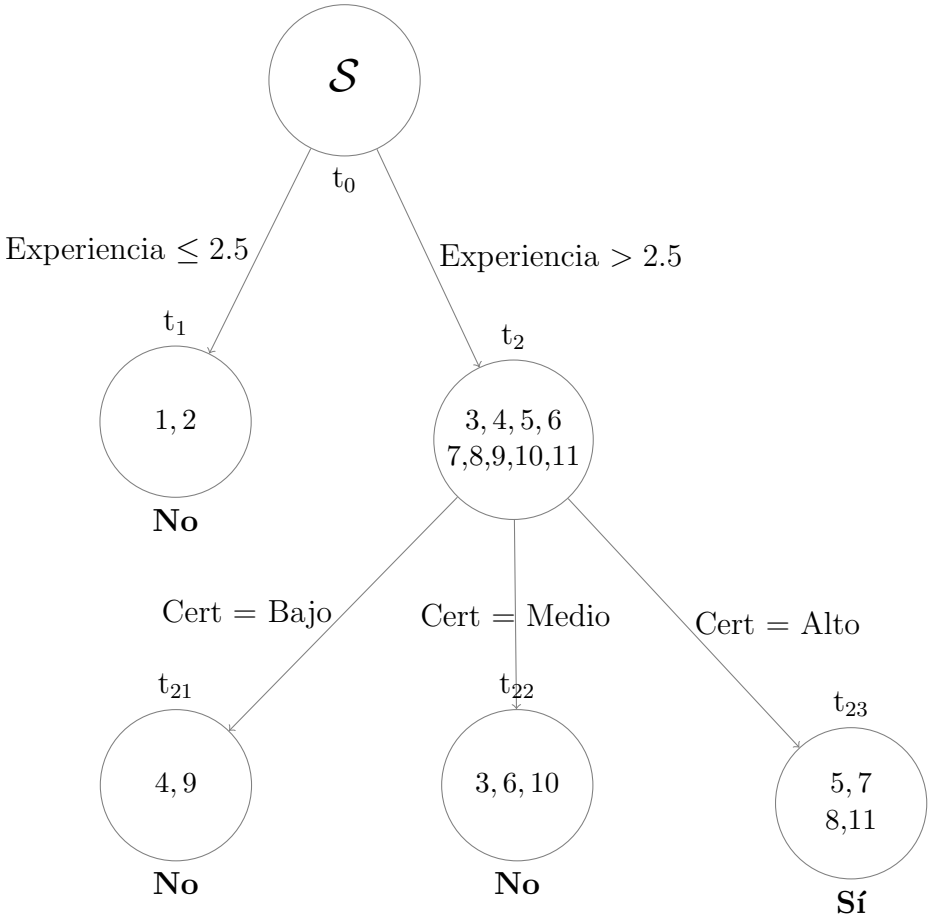


Figura 3.8: Árbol de decisión completo tras los splits en *Experiencia* y *Nivel de Certificación.*

Capítulo 4

Experimentos computacionales

4.1. Introducción

En este capítulo nos centraremos en aplicar los dos modelos de clasificación (SVM y árboles de decisión) estudiados en este trabajo a una base de datos real. En particular, partimos con $n = 708$ observaciones recogidas en el Observatorio de Rota-Base Naval y que se pueden descargar en la web AEMET OpenData. Cada observación incluye información sobre un día concreto. Para ser más precisos, en este trabajo nos centraremos en analizar cinco variables de entrada, a saber: la temperatura máxima (`tmax`), la velocidad media del viento (`wspd`), el volumen de precipitaciones (`prec`), la presión atmosférica (`pressMax`) y la dirección del viento (`dir`). La variable de salida no está definida en los datos descargados. Es por ello que se generará de forma sintética. En concreto, se ha elegido predecir si un día es apto o no para ir a la playa de Rota. La etiqueta $y = 1$ estará asignada a los días aptos, mientras que la etiqueta $y = -1$ se asociará a los días no aptos. Los detalles sobre la generación de las etiquetas se detallarán en las siguientes secciones.

Para evaluar la calidad de los distintos modelos de clasificación usaremos la métrica conocida como precisión o *accuracy*, y que se define como la proporción de datos que el modelo clasifica correctamente. Más formalmente, si tenemos un conjunto de datos $\mathcal{S} = \{(\vec{x_i}, y_i)\}_{i=1}^{n}$ y el clasificador produce las predicciones $\hat{y}_i$, entonces la precisión se define como:

$$\frac{1}{n}\sum_{i=1}^{n} \mathbf{1}\big(\hat{y}_i = y_i\big),$$

donde $\mathbf{1}(\cdot)$ es la función indicadora que vale 1 si la condición es verdadera y 0 en caso contrario.

Con el fin de evaluar la capacidad de generalización, durante todos los experimentos dividimos aleatoriamente los datos de cada conjunto de datos utilizado en un 70 % para entrenamiento (495 días) y un 30 % para prueba (213 días).
Todos los experimentos desarrollados en este trabajo se han codificado utilizando el lenguaje de programación Python. Para la implementación de las máquinas de vector soporte se ha empleado la biblioteca `scikit-learn` (`sklearn`). En particular, se ha configurado el parámetro `dual=False` dentro del modelo `LinearSVC`. Este clasificador se ha aplicado a la resolución del problema descrito en el Capítulo 2.

Para la implementación de los árboles de decisión se ha utilizado la clase `DecisionTreeClassifier` de la biblioteca `scikit-learn`. En el desarrollo de los experimentos se ha usado como criterio de impureza la entropía de Shannon, y se han ajustado

parámetros como la profundidad máxima del árbol que se especificará en cada apartado y el número mínimo de muestras por nodo para evitar el sobreajuste, en nuestro caso se ha usado $m_{\text{mín}} = 2$.

Todo el código fuente utilizado para la implementación de los modelos, así como los scripts de preprocesamiento y análisis de resultados, se encuentra disponible públicamente y puede consultarse en el siguiente repositorio de GitHub: `https://github.com/Teresamoral/TFG-2025`.

A continuación, analizaremos distintos casos de estudio.

4.2. Estudio con dos variables

En esta primera sección con el objetivo de visualizar de forma ilustrativa los datos y resultados, asumimos que las variables de entrada tienen sólo dos componentes, es decir, $x_i = (x_i^1, x_i^2), \forall i$ donde la primera componente está asociada a la temperatura máxima (`tmax`) y la segunda a la velocidad media del viento (`wspd`). En cuanto a la generación sintética de la variable respuesta, denotada por `aptoplay`, asumiremos que depende de las dos variables de entrada de la siguiente forma:

$$\texttt{aptoplay} = \begin{cases} +1, & \text{si } \texttt{tmax} \geq 25\ ^\circ\text{C} \ \wedge\ \texttt{wspd} \leq 8\ \text{km/h}, \\ -1, & \text{en otro caso.} \end{cases} \tag{4.1}$$

Es decir, diremos que un día es apto para ir a la playa si la temperatura máxima es mayor o igual a 25°C y la velocidad media del viento es menor o igual a 8km/h.

En la Tabla 4.1 se observa una pequeña muestra de cómo son los datos en base a la definición propuesta de (4.1). La primera columna corresponde a la fecha a la que corresponden los valores, la segunda y tercera columna hacen referencia a la temperatura máxima y a la velocidad media del viento de ese día, respectivamente. Por último, la última columna indica la etiqueta con la que ha sido clasificado. Por ejemplo, vemos que el 1 de febrero de 2023 no fue un día apto para ir a la playa, pero el 15 de marzo de 2023 sí que lo fue.

fecha	**tmax (°C)**	**wspd (km/h)**	**aptoplay**
2023-02-01	17.9	1.7	-1
2023-02-05	21.6	1.4	-1
2023-03-15	25.5	5.6	$+1$
2023-03-25	26.1	3.3	$+1$

Tabla 4.1: Ejemplos de observaciones

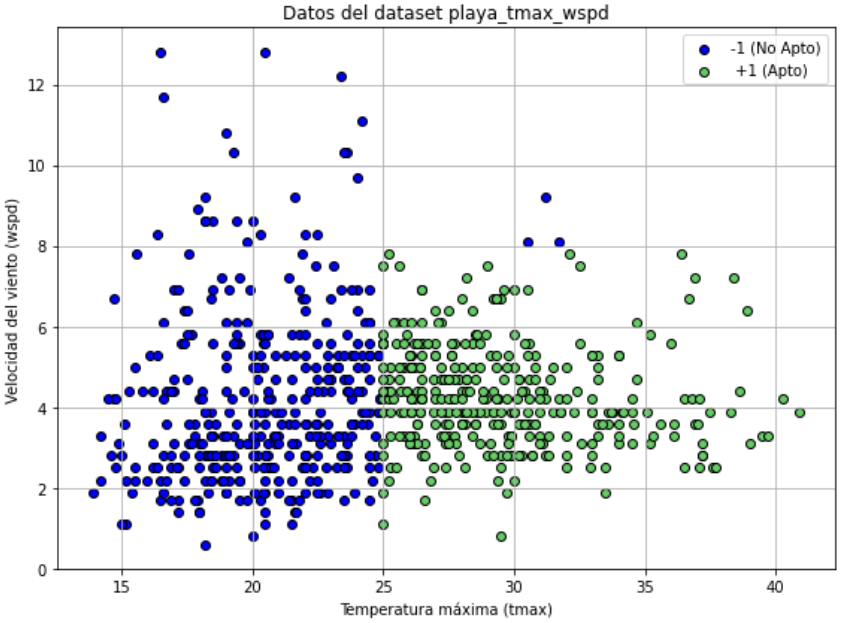


Figura 4.1: Representación de los datos del conjunto original en el plano definido por la temperatura máxima (`tmax`) y la velocidad del viento (`wspd`)

En la Figura 4.1 se muestra la distribución del conjunto de datos original en función de las dos variables predictoras: la temperatura máxima diaria (`tmax`) y la velocidad del viento (`wspd`). Cada punto de la gráfica representa un día registrado en el dataset. Los círculos en color verde corresponden a los días clasificados como aptos para la playa (+1), mientras que los círculos en color azul representan los días no aptos (-1).

A continuación, analizaremos los resultados obtenidos tras ejecutar el modelo SVM (Sección 4.2.1) o el modelo de árboles de decisión (Sección 4.2.2).

4.2.1. Resultados con SVM

Comenzamos entrenado el SVM lineal de margen blando resolviendo el problema primal (2.14) con parámetro de regularización $\mathcal{R} = 1.0$. El vector $\vec{w}$ y el término independiente b_0 del hiperplano fueron:

$$\vec{w} = \begin{pmatrix} 0.2959 \\ -0.0999 \end{pmatrix}, \qquad b_0 = -6.9126,$$

de modo que la ecuación del hiperplano óptimo es

$$0.2959\,\texttt{tmax} \;-\; 0.0999\,\texttt{wspd} \;-\; 6.9126 \;=\; 0.$$

El margen máximo (τ_{max}), calculado como $1/\|\vec{w}\|$, resultó

$$\frac{1}{\sqrt{0.2959^2 + (-0.0999)^2}} \;\approx\; 3.2019,$$

y la anchura total del margen, $2/\|\vec{w}\|$, fue aproximadamente 6.4038.

En el conjunto de entrenamiento definimos las variables de holgura ξ_i como

$$\xi_i = \max\bigl(0,\, 1 - y_i(\vec{w} \cdot x_i + b_0)\bigr),$$

observando que su suma total resultó

$$\sum_i \xi_i = 107.2127,$$

por lo que el término de penalización $\mathcal{R}\sum_i \xi_i$ vale también 107.2127. Identificamos los vectores soporte seleccionando aquellos datos con $\xi_i \approx 0$ y $|D(\vec{x_i})| \approx 1$, obteniendo:

$$\begin{aligned} \vec{x_6} &= (20.8, 2.5), \quad y = -1, \quad D(\vec{x}) = -1.008, \\ \vec{x_{90}} &= (21.0, 3.1), \quad y = -1, \quad D(\vec{x}) = -1.009, \\ \vec{x_{225}} &= (27.8, 3.1), \quad y = +1, \quad D(\vec{x}) = +1.003. \end{aligned}$$

Finalmente, las precisiones obtenidas fueron

$$\text{Accuracy}_{\text{train}} = 0.9677, \qquad \text{Accuracy}_{\text{test}} = 0.9624,$$

En conclusión, el hiperplano

$$0.2959\, x^1 \; - \; 0.0999\, x^2 \; - \; 6.9126 \; = \; 0. \tag{4.2}$$

donde x^1 =tmax y x^2=wspd es el óptimo bajo el criterio de margen blando con $\mathcal{R} = 1.0$, proporcionando un margen de anchura aproximadamente 6.4038.

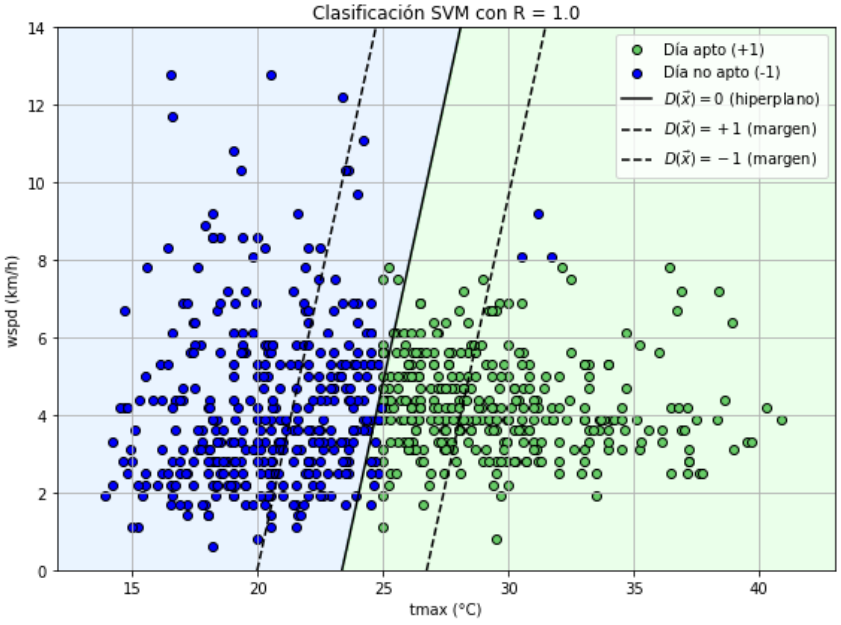


Figura 4.2: Representación del hiperplano óptimo para $\mathcal{R} = 1$.

La Figura 4.2 muestra el resultado del hiperplano óptimo de la Expresión 4.2 con parámetro de regularización $\mathcal{R} = 1.0$, sobre el conjunto de datos definido por la temperatura máxima (`tmax`) y la velocidad del viento (`wspd`). La línea negra continua corresponde al hiperplano de decisión $D(\vec{x}) = 0$, mientras que las líneas discontinuas representan los márgenes máximos de separación $D(\vec{x}) = \pm 1$. El área sombreada ilustra la región que el modelo clasifica como apta (verde) o no apta (azul).

A continuación, realizaremos un estudio de la influencia del valor de $\mathcal{R}$ en este método.

Influencia del parámetro de regularización $\mathcal{R}$

Tal y como se comentó en la sección (2.2), el parámetro de regularización $\mathcal{R}$ controla el compromiso entre el ancho del margen y la penalización por clasificar datos de forma incorrecta, [28]. En particular:

- Si $\mathcal{R} \to \infty$, las variables de holgura ξ_i tienden a cero y el SVM busca que ningún punto viole el margen, estrechando dicho margen y aumentando el riesgo de sobreajuste. En este caso, el problema de optimización resultante equivale al de margen duro visto en la Sección 2.1.
- Si $\mathcal{R} \to 0$, el término $\mathcal{R}\sum_i \xi_i$ prácticamente desaparece del objetivo, de modo que el SVM permite muchos datos mal clasificados dentro del margen a fin de maximizar exclusivamente su anchura, lo cual puede derivar en un clasificador muy laxo.

Para ilustrar esta dependencia, entrenamos tres modelos SVM con $\mathcal{R} \in \{0.1,\ 1.0, 10^6\}$ sobre el mismo conjunto de 495 datos de entrenamiento y comparamos los resultados en términos del hiperplano óptimo obtenido a través de las variables de decisión $(\vec{w}, b_0)$, la norma del vector normal $\|\vec{w}\|$, el margen máximo $\tau_{\text{máx}}$, la suma de las penalizaciones $\sum_i \xi_i$, y la precisión en el conjunto test, Acc_{test}. La Tabla 4.2 muestra los resultados obtenidos.

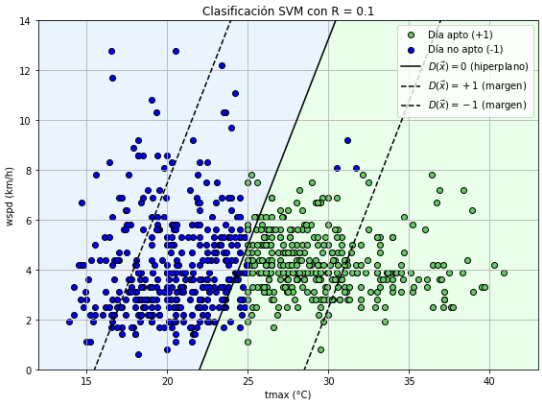
$\mathcal{R}$	$\|\tilde{\mathbf{w}}\|$	$\tau_{\text{máx}}$	$\sum_i \xi_i$	Acc_{test}
0.1	0.1793	5.5760	199.1610	0.9484
1.0	0.3123	3.2019	107.2130	0.9624
10^6	0.5082	1.9679	70.6327	0.9577

Tabla 4.2: Efecto de $\mathcal{R}$ sobre la norma de $\tilde{\mathbf{w}}$, el margen $1/\|\tilde{\mathbf{w}}\|$, la suma de holguras $\sum_i \xi_i$ y la precisión en test para el nuevo dataset.

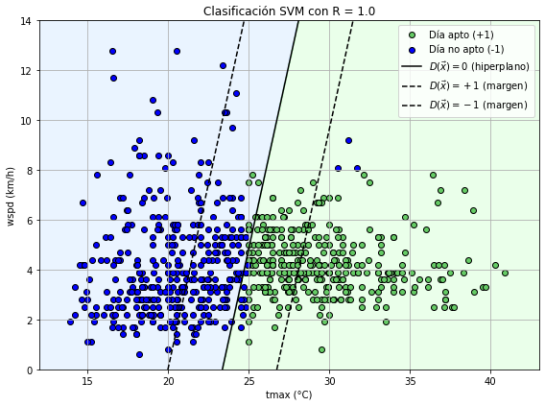
Para cada valor de $\mathcal{R}$, la ecuación del hiperplano óptimo es:

$$\begin{aligned}
\mathcal{R} = 0.1: &\quad 0.1536\,\texttt{tmax} - 0.0927\,\texttt{wspd} - 3.3789 = 0,\\
\mathcal{R} = 1.0: &\quad 0.2959\,\texttt{tmax} - 0.0999\,\texttt{wspd} - 6.9126 = 0,\\
\mathcal{R} = 10^6: &\quad 0.4847\,\texttt{tmax} - 0.1525\,\texttt{wspd} - 11.4258 = 0.
\end{aligned}$$

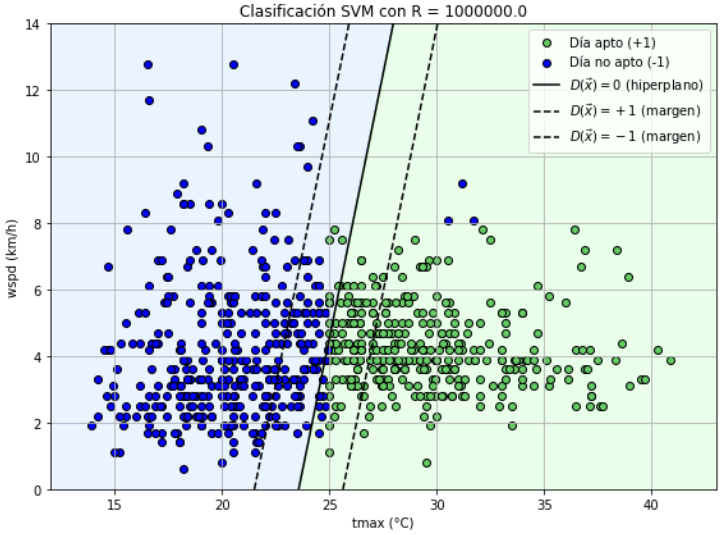
En la Figura 4.3 se muestra una representación de los hiperplanos resultantes para cada valor de $\mathcal{R}$, además del margen correspondiente a cada uno.



(a) $\mathcal{R} = 0.1$



(b) $\mathcal{R} = 1.0$



(c) $\mathcal{R} = 10^6$

Figura 4.3: Comparación de modelos SVM con distintos valores del parámetro $\mathcal{R}$.

Para el caso $\mathcal{R} = \mathbf{0.1}$, al asignar un peso muy bajo a la penalización de las holguras, el SVM busca ante todo maximizar el margen, incluso a costa de tolerar muchos errores de clasificación. En nuestro experimento, esto se traduce en un margen muy amplio como puede observarse en la Figura 4.3a y una suma de las variables de holgura muy grande, lo que provoca una *accuracy* en test de 0.9484. El clasificador se relaja y admite numerosas violaciones del margen para ensancharlo.

Por otro lado, si $\mathcal{R} = \mathbf{1.0}$ se tiene un término de penalización equilibrado, el SVM compensa de forma razonable el ancho del margen y los errores interiores (Figura 4.3b). Este valor intermedio de $\mathcal{R}$ ofrece un buen compromiso entre generalización y reducción de errores de margen.

Por último, el caso $\mathcal{R} = \mathbf{10^6}$ sería equivalente a resolver el problema SVM de margen duro (Sección 2.1), puesto que al hacer $\mathcal{R}$ tender a infinito, las variables de holgura tienden a cero. Nótese, no obstante, que tal y como se aprecia en la Tabla 4.2, la suma de las

variables de holgura no tiende a cero, sino que su valor se aproxima a 70. Esto es debido a que los datos no son linealmente separables, y por lo tanto, no se puede prescindir totalmente de la holgura, proporcionada por las ξ_i, por lo que el modelo tolera algunos errores tal y como puede observarse en la Figura 4.3c.

4.2.2. Resultados con árboles de decisión

A continuación, resolveremos el problema entrenando un árbol de decisión con los siguientes criterios: la entropía de Shannon como función de impureza y una profundidad máxima de 3.

La Figura 4.4 muestra un esquema del árbol de decisión resultante al entrenar los datos extraídos para el conjunto de entrenamiento. Cada rama muestra el *split* elegido (la característica y su umbral). En primer lugar, si la temperatura máxima del día cumple $\texttt{tmax} \leq 24.95$, se predice que el día no es apto para ir a la playa (clase -1). En caso contrario, si $\texttt{tmax} > 24.95$, se evalúa la velocidad del viento: si $\texttt{wspd} \leq 7.95$, el modelo predice un día apto (clase $+1$); por el contrario, si $\texttt{wspd} > 7.95$, vuelve a clasificarse como no apto (clase -1).

La entropía del nodo, el número de muestras que contiene y el vector de conteo de clases, este último muestra un par de coordenadas con el número de datos pertenecientes a la clase -1 y +1, respectivamente. Concretamente, el nodo raíz presenta la entropía máxima, indicando mezcla de clases, y se divide según la característica que maximiza la ganancia de información. Cada rama añade un nuevo split hasta alcanzar nodos hoja de entropía cero, donde todas las muestras pertenecen a la misma clase y se asigna directamente la etiqueta correspondiente.

El modelo alcanzó precisión perfecta tanto en entrenamiento como en prueba:

$$\text{Accuracy}_{\text{train}} = 1.0000, \qquad \text{Accuracy}_{\text{test}} = 1.0000.$$

El hecho de que el modelo de árbol de decisión alcance una precisión perfecta tanto en el conjunto de entrenamiento como en el de prueba es coherente debido a la forma en que se ha definido la variable de salida `aptoplay`. Esta variable ha sido generada directamente a partir de reglas lógicas aplicadas sobre las variables `tmax` y `wspd`. Por tanto, al tratarse de una función perfectamente separable en función de estas dos variables con cortes ortogonales, un árbol de decisión con suficiente profundidad es capaz de aprender las condiciones exactas que determinan la clase sin cometer errores.

La Figura 4.5 muestra cómo el árbol de decisión ha aprendido dos cortes ortogonales en el plano (tmax, wspd) para separar perfectamente los días aptos (círculos verdes) de los no aptos (círculos azules): primero, traza una línea vertical en $\texttt{tmax} = 24.95\,^{\circ}\text{C}$ que clasifica todo a la izquierda como "no apto"; luego, para los días con $\texttt{tmax} > 24.95\,^{\circ}\text{C}$, aplica un segundo corte horizontal en $\texttt{wspd} = 7.95\,\text{km/h}$, de modo que la subzona inferior (viento moderado) se considera "apta" y la subzona superior (viento fuerte) vuelve a etiquetarse como "no apta". A partir de estos cortes se genera una región de decisión para los días aptos (zona verde) y otra para los días no aptos (zona azul).

En conclusión, el árbol de decisión demuestra que la temperatura máxima es la variable más informativa para decidir la aptitud de un día de playa, y que solo en los casos de temperatura alta entra en juego la velocidad del viento, con un umbral de 7.95 km/h.

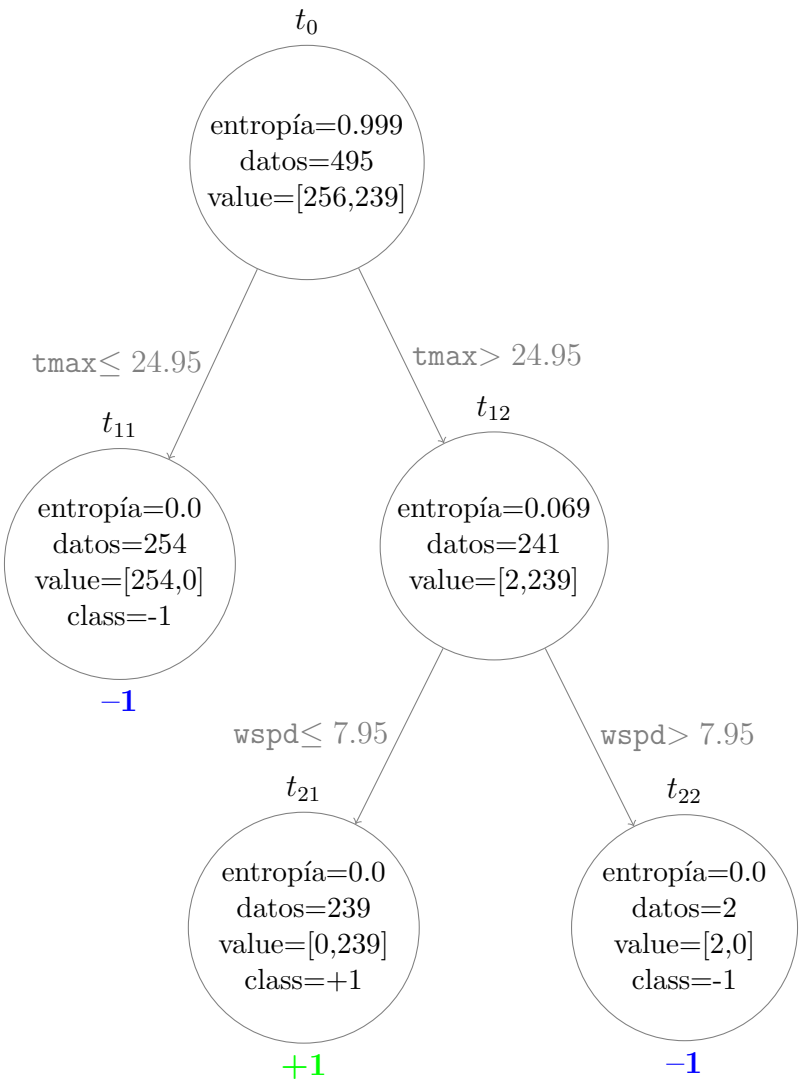


Figura 4.4: Árbol de Decisión mostrando umbrales, entropía, número de muestras, vectores de conteo y clase en cada nodo.

La sencillez de este modelo y su precisión perfecta confirman que estas dos condiciones bastan para clasificar con éxito los días aptos o no aptos.

4.2.3. Predicción de un nuevo punto con SVM y árboles de decisión

Pasamos ahora a ver con un ejemplo concreto cómo se realiza la predicción de un nuevo día. Supongamos que hoy se han registrado en el observatorio de Rota los siguientes valores meteorológicos: una temperatura máxima de 26.7 °C y una velocidad media del viento de 6.2 km/h. Queremos predecir si, dadas estas condiciones, el día puede considerarse apto para ir a la playa.

Comenzamos evaluando el modelo entrenado mediante Máquinas de Vectores Soporte (SVM) de margen blando con $\mathcal{R} = 1.0$, sustituyendo los valores observados en la Ecuación

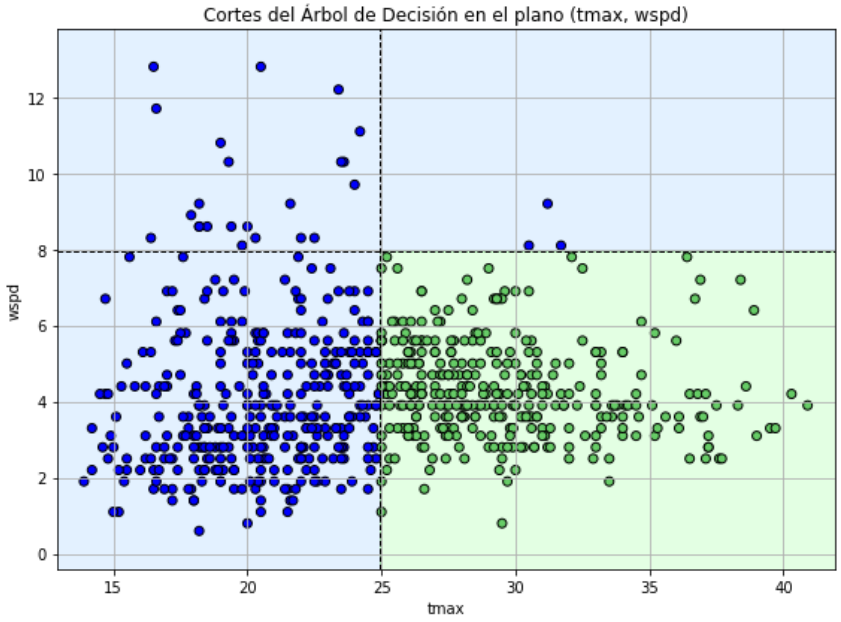


Figura 4.5: Región de decisión del árbol de decisión

del hiperplano óptimo (4.2), se obtiene:

$$D(\vec{x}) = 0.2959 \cdot 26.7 - 0.0999 \cdot 6.2 - 6.9126 \approx 0.234.$$

Como el valor de la función de decisión es positivo, el modelo predice que el día es apto (clase $+1$).

Por otro lado, si aplicamos el árbol de decisión entrenado con las mismas variables, el procedimiento de clasificación sigue la estructura del árbol mostrado en la Figura 4.5. En primer lugar, se comprueba si la temperatura supera el umbral de 24.95 °C, lo cual se cumple. Finalmente, se evalúa la velocidad del viento, que también supera el valor límite de 7.95 km/h, por lo que el árbol clasifica este día como apto para ir a la playa.

En conclusión, ambos modelos coinciden en su predicción.

4.3. Estudio con variables adicionales

En esta sección se comparan los resultados obtenidos al entrenar modelos de clasificación sobre cinco versiones distintas del conjunto de datos, que contienen respectivamente 1, 2, 3, 4 y 5 variables predictoras. En todos los casos, se utiliza la misma variable objetivo binaria que en la Sección 4.2, definida a partir de la siguiente regla:

$$\texttt{aptoplay} = \begin{cases} +1, & \text{si } \texttt{tmax} \geq 25\ ^\circ\text{C} \ \wedge\ \texttt{wspd} \leq 8\ \text{km/h}, \\ -1, & \text{en otro caso.} \end{cases}$$

La finalidad de este análisis es evaluar si la incorporación progresiva de más variables permite mejorar el rendimiento de los modelos, o si por el contrario dichas variables actúan como ruido y perjudican a la clasificación. Para ello, se entrenan tanto máquinas de vectores soporte (SVM) como árboles de decisión sobre cada conjunto, comparando la precisión obtenida en entrenamiento y test. A continuación se muestran ejemplos representativos

de cada dataset reducido, lo que permite visualizar cómo evoluciona la complejidad de los datos a medida que se amplía la dimensión.

Las siguientes tablas recogen una muestra representativa de cada uno de estos conjuntos. La Tabla 4.3 muestra el caso más simple, con únicamente la temperatura máxima. En la Tabla 4.4 se añade la velocidad del viento, lo que permite capturar completamente la lógica de la etiqueta. A partir de ahí, se incorporan variables adicionales que podrían influir indirectamente: la cantidad de sol diario en la Tabla 4.5, la presión atmosférica máxima en la Tabla 4.6 y, finalmente, la dirección del viento en la Tabla 4.7. Estos conjuntos serán utilizados para entrenar y comparar clasificadores SVM y árboles de decisión, evaluando su precisión en entrenamiento y test con el fin de valorar si las variables añadidas aportan información útil o simplemente introducen ruido.

Fecha	**tmax**	**aptoplay**
2023-02-08	16.7	-1
2023-03-19	24.2	-1
2023-04-03	20.5	-1
2023-05-31	26.4	+1
2023-06-27	36.9	+1

Tabla 4.3: Conjunto de cinco datos reales con una única variable predictora, `tmax`, y la etiqueta `aptoplay`

Fecha	**tmax**	**wspd**	**aptoplay**
2023-02-08	16.7	4.4	-1
2023-03-19	24.2	4.4	-1
2023-04-03	20.5	3.9	-1
2023-05-31	26.4	5.0	+1
2023-06-27	36.9	3.6	+1

Tabla 4.4: Valores reales del conjunto de datos para cinco fechas representativas, incluyendo las variables predictoras `tmax`, y `wspd`.

Fecha	**tmax**	**wspd**	**sol**	**aptoplay**
2023-02-08	16.7	4.4	3.8	-1
2023-03-19	24.2	4.4	10.8	-1
2023-04-03	20.5	3.9	10.8	-1
2023-05-31	26.4	5.0	13.4	+1
2023-06-27	36.9	3.6	11.9	+1

Tabla 4.5: Valores reales del conjunto de datos para cinco fechas representativas, incluyendo tres variables predictoras: `tmax`, `wspd` y `sol`; y la etiqueta `aptoplay`.

Fecha	tmax	wspd	sol	presMax	aptoplay
2023-02-08	16.7	4.4	3.8	1019.4	-1
2023-03-19	24.2	4.4	10.8	1016.7	-1
2023-04-03	20.5	3.9	1080	1017.5	-1
2023-05-31	26.4	5.0	13.4	1014.6	+1
2023-06-27	36.9	3.6	11.9	1013.4	+1

Tabla 4.6: Valores reales del conjunto de datos para cinco fechas representativas, incluyendo las variables `tmax`, `wspd`, `sol`, `presMax` y la etiqueta `aptoplay`.

Fecha	tmax	wspd	sol	presMax	dir	aptoplay
2023-02-08	16.7	4.4	3.8	1019.4	15.0	-1
2023-03-19	24.2	4.4	10.8	1016.7	1.0	-1
2023-04-03	20.5	3.9	1080	1017.5	1.0	-1
2023-05-31	26.4	5.0	13.4	1014.6	99.0	+1
2023-06-27	36.9	3.6	11.9	1013.4	28	+1

Tabla 4.7: Versión más completa del dataset con las variables `tmax`, `wspd`, `sol`, `presMax`, `dir` y la etiqueta `aptoplay`.

4.3.1. Entrenamiento con SVM

En este apartado se han entrenado modelos SVM (máquinas de vectores soporte) utilizando entre una y cinco características. En todos los casos, se ha empleado un SVM de margen blando con formulación primal y $\mathcal{R} = 1$, entrenados sobre los conjuntos de datos detallados anteriormente.

La Tabla 4.8 resume la precisión obtenida en los conjuntos de entrenamiento y test para cada configuración. Se observa que incluso utilizando una única variable, `tmax`, el modelo ya alcanza una precisión notable, y que añadir variables como `wspd` o `sol` no siempre implica una mejora significativa. No obstante, los resultados son muy competitivos en todos los casos, pero en concreto, se puede observar que la variable `tmax` es la que tiene mayor poder predictivo ya que al evaluar el modelo solamente con esta variable se obtiene la mayor precisión tanto en el conjunto de entrenamiento como en el del test.

Número de variables	Características	Train Acc.	Test Acc.
1	`tmax`	0.9939	0.9906
2	`tmax, wspd`	0.9677	0.9624
3	`tmax, wspd, sol`	0.9754	0.9569
4	`tmax, wspd, sol, presMax`	0.9774	0.9522
5	`tmax, wspd, sol, presMax, dir`	0.9753	0.9665

Tabla 4.8: Precisión en entrenamiento y prueba para cada conjunto de características.

Los hiperplanos de decisión aprendidos por cada modelo tienen la forma $D_i(\vec{x}) = 0$,

con $i = 1, \ldots, 5$, donde $\vec{x}$ representa el vector de variables predictoras e i representa el número de éstas. Las expresiones exactas para cada uno de ellos se recogen a continuación:

$$D_1(\vec{x}) = 0.28\,\texttt{tmax} - 6.90 = 0$$
$$D_2(\vec{x}) = 0.30\,\texttt{tmax} - 0.10\,\texttt{wspd} - 6.91 = 0$$
$$D_3(\vec{x}) = 0.28\,\texttt{tmax} - 0.11\,\texttt{wspd} + 0.03\,\texttt{sol} - 6.81 = 0$$
$$D_4(\vec{x}) = 0.49\,\texttt{tmax} - 0.20\,\texttt{wspd} + 0.03\,\texttt{sol} - 0.01\,\texttt{presMax} + 0.001 = 0$$
$$D_5(\vec{x}) = 0.48\,\texttt{tmax} - 0.15\,\texttt{wspd} + 0.02\,\texttt{sol} - 0.01\,\texttt{presMax} + 0.0006\,\texttt{dir} + 0.0006 = 0$$

Con el fin de visualizar la evolución del rendimiento de los modelos, se representa en la Figura 4.6 una gráfica de barras que compara la precisión en entrenamiento y prueba según el número de variables utilizadas. Esta comparación permite observar con claridad que el modelo más sencillo es el que ofrece mayor precisión, y que los modelos con más variables tienden a estabilizarse alrededor de una precisión próxima al 96–97 %.

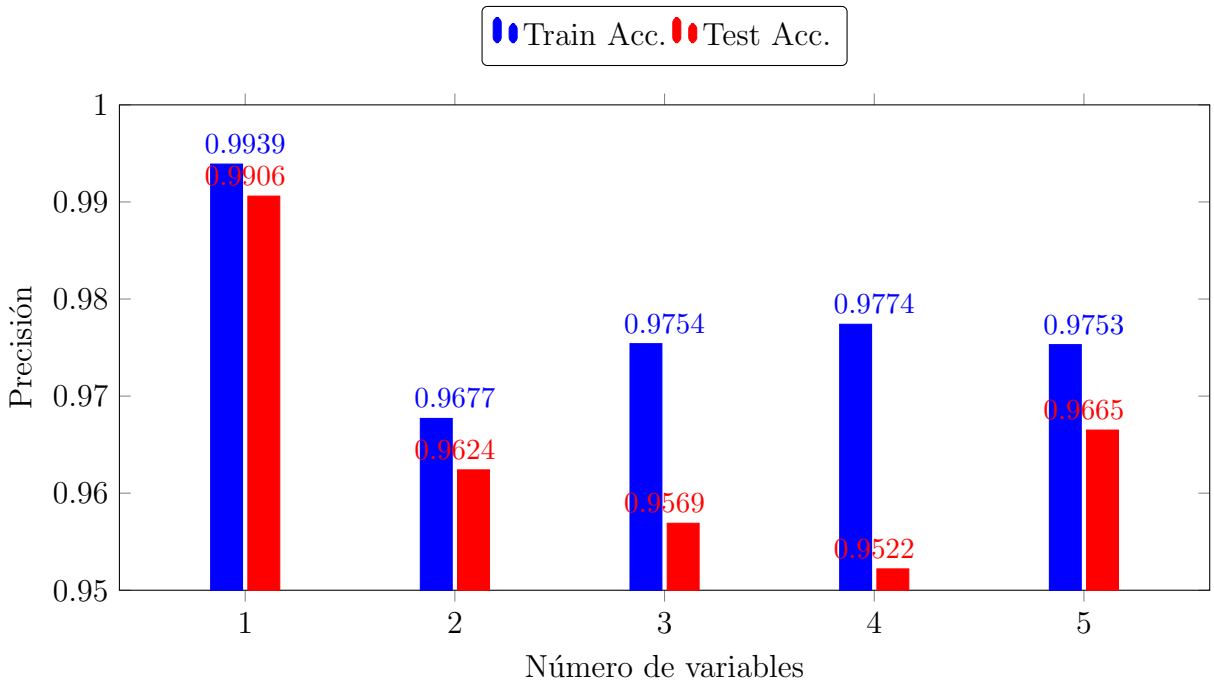


Figura 4.6: Precisión de entrenamiento y de prueba en función del número de variables utilizadas en el modelo SVM.

4.3.2. Entrenamiento con Árboles de Decisión

Además de los modelos SVM, se han entrenado clasificadores basados en árboles de decisión utilizando los mismos conjuntos de datos con entre una y cinco variables predictoras. En todos los casos se ha limitado la profundidad máxima del árbol a 4, con el objetivo de obtener modelos simples, interpretables y comparables entre sí.

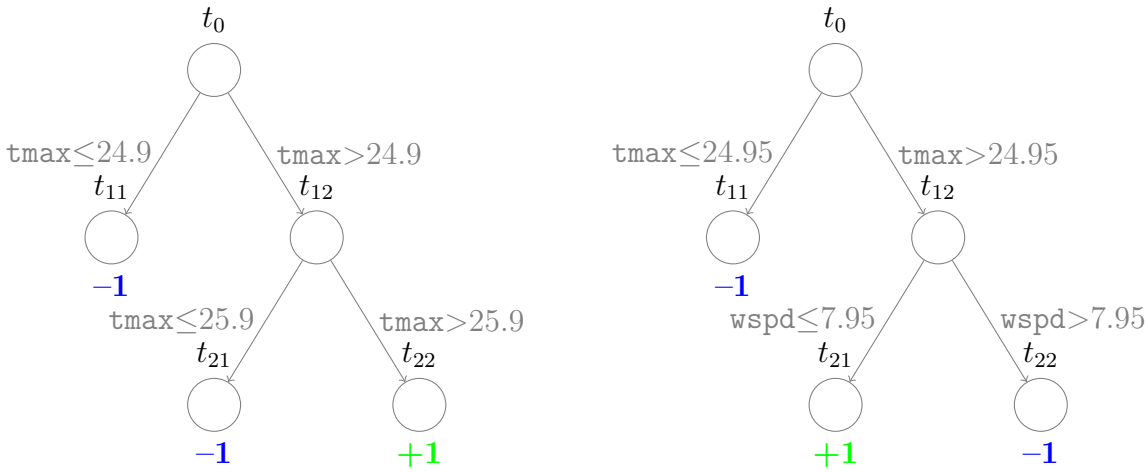
La Tabla 4.9 resume la precisión alcanzada por estos árboles de decisión en los conjuntos de entrenamiento y test. Como puede observarse, incluso utilizando una única variable

(`tmax`), el modelo ya es capaz de alcanzar una precisión superior al 99 % en ambos subconjuntos. A partir de dos variables, todos los modelos logran un ajuste perfecto, tanto en entrenamiento como en test.

Número de variables	Características	Train Acc.	Test Acc.
1	`tmax`	0.9960	0.9953
2	`tmax`, `wspd`	1.0000	1.0000
3	`tmax`, `wspd` , `sol`	1.0000	1.0000
4	`tmax`,`wspd` , `sol`, `presMax`	1.0000	1.0000
5	`tmax`, `wspd` , `sol`, `presMax`, `dir`	1.0000	1.0000

Tabla 4.9: Precisión en entrenamiento y prueba para árboles de decisión con profundidad máxima 4.

A continuación se muestra en la Figura 4.7 un esquema de los árboles obtenidos para los modelos con una y dos variables predictoras, respectivamente. El árbol de la Figura 4.7a realiza dos cortes sucesivos sobre la variable `tmax`, lo que indica que esta variable por sí sola permite segmentar con bastante precisión los datos, mientras que el árbol de la Figura 4.7b, al disponer de una segunda variable (`wspd`), logra separar perfectamente las clases con una única ramificación adicional, simplificando la estructura y mejorando la interpretabilidad del modelo.



(a) Modelo con una variable (`tmax`).

(b) Modelo con dos variables (`tmax`, `wspd`).

Figura 4.7: Estructura de los árboles de decisión para 1 y 2 variables.

Para los conjuntos de datos con 3, 4 y 5 variables se obtiene exactamente el mismo árbol que en la Figura 4.7b. Esto refleja que el modelo aprende de forma consistente, es decir, que las variables `tmax` y `wspd` son suficientes para realizar una separación perfecta de las clases.

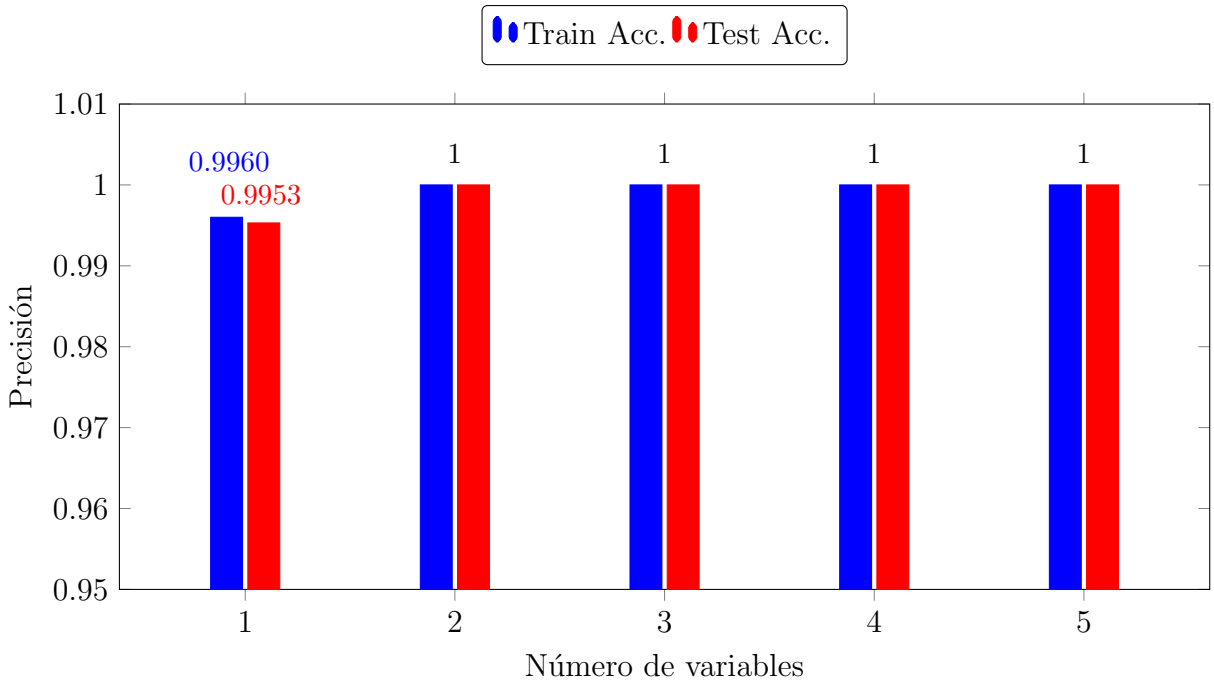


Figura 4.8: Precisión en entrenamiento y prueba en árboles de decisión con profundidad máxima 4.

La Figura 4.8 representa gráficamente la precisión alcanzada por los árboles de decisión entrenados con diferentes combinaciones de variables predictoras, desde una sola hasta cinco. En el eje horizontal se indica el número de variables empleadas, mientras que en el eje vertical se muestra la precisión (*accuracy*) obtenida, tanto en el conjunto de entrenamiento (barras azules) como en el de test (barras rojas). Se observa que, incluso utilizando únicamente `tmax`, el modelo alcanza una precisión superior al 99 %. A partir de dos variables (`tmax` y `wspd`), todos los modelos logran un ajuste perfecto (*accuracy* = 1.0), tanto en entrenamiento como en test. Este comportamiento indica que el problema es muy fácil de separar con un número reducido de variables.

Los resultados obtenidos muestran que los árboles de decisión, incluso con una profundidad muy limitada, son capaces de capturar la lógica subyacente del problema de clasificación con gran eficacia, esto se debe a que la etiqueta está formulada siguiendo una lógica similar a la de los árboles de decisión. El hecho de que los modelos no cambien su estructura al aumentar el número de variables refuerza la hipótesis de que las condiciones climáticas definidas por `tmax` y `wspd` contienen toda la información relevante para predecir la variable de salida.

4.4. Selección de variables

En esta sección se aborda el proceso de selección de variables (*feature selection*), una etapa fundamental en la construcción de modelos predictivos eficientes e interpretables. El objetivo es identificar qué subconjunto de variables contiene la información más relevante para predecir la variable de interés, eliminando aquellas que resultan redundantes o irrelevantes. Para ello, añadimos una variable nueva a la etiqueta de salida, la presión atmosférica máxima (`presMax`). Consideramos que un día es bueno si este valor supera

los 1010 hPa porque una presión atmosférica alta se asocia normalmente con condiciones meteorológicas estables: cielo despejado, baja probabilidad de precipitaciones y vientos suaves; además de cumplir las condiciones de temperatura y viento.

Por tanto, la nueva etiqueta `aptoplay` se define como

$$\texttt{aptoplay} = \begin{cases} +1, & \text{si } \texttt{tmax} \geq 25\ ^\circ\text{C},\ \texttt{wspd} \leq 8\ \text{km/h},\ \texttt{presMax} > 1010\text{hPa}, \\ -1, & \text{en otro caso.} \end{cases}$$

De esta manera, el modelo aprenderá no solo de la temperatura máxima (`tmax`) y la velocidad media del viento (`wspd`), sino también de la presión atmosférica (`presMax`) como requisito imprescindible para un día de playa.

En nuestro caso, se ha llevado a cabo una búsqueda exhaustiva conocida como "fuerza bruta" aunque existen otras técnicas más sofisticadas que pueden consultarse en [8; 17]. Utilizamos únicamente tres variables meteorológicas: `tmax`, `wspd` y `presMax`. Se han generado todas las combinaciones posibles de una, dos y tres variables, y para cada una se ha entrenado un clasificador SVM con $\mathcal{R} = 1$. Posteriormente, se ha calculado la precisión del modelo sobre los conjuntos de entrenamiento y test, permitiendo así comparar el rendimiento de cada combinación.

Variables	**Train Accuracy**	**Test Accuracy**
tmax	0.9324	0.8936
wspd	0.5498	0.5674
presMax	0.5498	0.5674
tmax, wspd	0.9270	0.9007
tmax, presMax	0.9324	0.8936
wspd, presMax	0.5498	0.5674
tmax, wspd, presMax	0.9235	0.9078

Tabla 4.10: Precisión del clasificador SVM lineal para distintas combinaciones de variables.

Como puede observarse en la Tabla 4.10, las combinaciones que incluyen únicamente `wspd` o `presMax` no resultan útiles por sí solas, ya que el modelo apenas supera una precisión del 57 % en el conjunto de test. Por el contrario, la variable `tmax`, tanto sola como en combinación con otras, logra resultados notablemente superiores. Las combinaciones `tmax + wspd` y `tmax + presMax` mejoran ligeramente la generalización del modelo, pero es la combinación completa `tmax, wspd, presMax` la que obtiene la mejor precisión en test (90.78 %).

Además del análisis con SVM, se ha realizado una evaluación del rendimiento de árboles de decisión sobre las mismas combinaciones de variables: `tmax`, `wspd` y `presMax`. Para cada subconjunto posible (de 1 a 3 variables), se ha entrenado un árbol con una profundidad máxima de 4 y se ha evaluado su precisión sobre los conjuntos de entrenamiento y test. Los resultados se resumen en la Tabla 4.11.

Variables	**Train Accuracy**	**Test Accuracy**
tmax	0.9609	0.9574
wspd	0.6459	0.5674
presMax	0.7562	0.7589
tmax, wspd	0.9680	0.9574
tmax, presMax	0.9964	1.0000
wspd, presMax	0.7705	0.7305
tmax, wspd, presMax	1.0000	1.0000

Tabla 4.11: Precisión del árbol de decisión (profundidad máxima 4) para distintas combinaciones de variables.

Como se puede observar en la Tabla 4.11, los árboles de decisión muestran un rendimiento muy competitivo desde el uso de la variable `tmax` que proporciona una buena precisión, y su combinación con `presMax` permite alcanzar una precisión perfecta. El modelo que utiliza las tres variables logra también un ajuste perfecto tanto en entrenamiento como en prueba, lo que sugiere que estas variables capturan de forma completa la lógica subyacente de la etiqueta. A diferencia del modelo SVM, los árboles parecen beneficiarse más claramente de la inclusión de `presMax`, lo que podría deberse a su capacidad para generar reglas de decisión no lineales que capturan mejor las interacciones entre variables y a la forma en la que se ha definido la etiqueta `aptoplay`.

Por lo tanto, este análisis confirma que `tmax` es la variable más informativa para la predicción de días aptos para ir a la playa de Rota y que su combinación con `wspd` y `presMax` puede aportar una ligera mejora adicional.

Capítulo 5

Conclusiones y líneas de futuro

En este trabajo se introdujeron los fundamentos del aprendizaje supervisado aplicados a la clasificación binaria, y se compararon dos enfoques metodológicos distintos: las Máquinas de Vectores Soporte (SVM), y los Árboles de Decisión. Ambos modelos se entrenaron sobre una base de datos real, con el objetivo de predecir la aptitud de los días para acudir a la playa en función de variables meteorológicas. A lo largo del estudio se observaron las ventajas y limitaciones de cada enfoque, tanto desde el punto de vista predictivo como interpretativo.

Como líneas futuras de mejora, se plantea por un lado, extender el modelo SVM al caso dual para aplicar funciones *kernel* que permitan trabajar con hiperplanos no lineales, [28]. Por otro lado, reemplazar la construcción heurística de árboles de decisión por métodos basados en problemas de optimización tal y como se propone en [4]. Una última dirección futura consistirá en el estudio de un problema de optimización que combine la resolución del modelo SVM y los árboles de decisión tal y como se ha hecho en [5; 13].

Bibliografía

[1] Ethem Alpaydin. *Introduction to machine learning*. MIT press, 2009.

[2] Jay E. Aronson. Decision trees – an overview, 2025.

[3] Bart Baesens, Tom Van Gestel, Stijn Viaene, Mila Stepanova, Johan A. K. Suykens, and Jan Vanthienen. Benchmarking state-of-the-art classification algorithms for credit scoring. *Journal of the Operational Research Society*, 54(6):627–635, 2003.

[4] Dimitris Bertsimas and Jack Dunn. Optimal classification trees. *Machine Learning*, 106:1039–1082, 2017.

[5] Víctor Blanco, Alberto Japón, and Justo Puerto. Multiclass optimal classification trees with svm-splits. *Machine Learning*, 112(12):4905–4928, 2023.

[6] Dustin Boswell. Introduction to support vector machines. *Department of Computer Science and Engineering University of California San Diego*, 11:16–17, 2002.

[7] Stephen P Boyd and Lieven Vandenberghe. *Convex optimization*. Cambridge university press, 2004.

[8] Paul S Bradley and Olvi L Mangasarian. Feature selection via concave minimization and support vector machines. In *ICML*, volume 98, pages 82–90, 1998.

[9] Leo Breiman, Jerome Friedman, Richard A Olshen, and Charles J Stone. *Classification and regression trees*. Routledge, 2017.

[10] Antonio Luis Gómez Cuevas. Selección de variables en máquinas de vectores soporte, 2024. Trabajo de Fin de Grado.

[11] Pádraig Cunningham, Matthieu Cord, and Sarah Jane Delany. *Supervised Learning*. Springer Berlin Heidelberg, Berlin, Heidelberg, 2008.

[12] María Cristina Molero del Río. Aprendizaje supervisado mediante random forest, 2017. Trabajo de Fin de Máster.

[13] Federico D'Onofrio, Giorgio Grani, Marta Monaci, and Laura Palagi. Margin optimal classification trees. *Computers & Operations Research*, 161:106441, 2024.

[14] Zoubin Ghahramani. Unsupervised learning. In *Summer school on machine learning*, pages 72–112. Springer, 2003.

[15] Erika Vilches González and Iván A Escobar Broitman. Minería de datos, 2007.

[16] Gema Valenzuela González. Supervised learning: Methods, properties and applications, 2022. Trabajo de Fin de Grado.

[17] Isabelle Guyon, Steve Gunn, Masoud Nikravesh, and Lofti A Zadeh. *Feature extraction: foundations and applications*, volume 207. Springer, 2008.

[18] Yukihiro Hamasuna, Yasunori Endo, and Sadaaki Miyamoto. Support vector machine for data with tolerance based on hard-margin and soft-margin. In *2008 IEEE International Conference on Fuzzy Systems (IEEE World Congress on Computational Intelligence)*, pages 750–755, 2008.

[19] Trevor Hastie, Robert Tibshirani, Jerome H Friedman, and Jerome H Friedman. *The elements of statistical learning: data mining, inference, and prediction*, volume 2. Springer, 2009.

[20] Carl I. Hovland and Earl B. Hunt. Computer simulation of concept attainment. *Behavioral Science*, 5(3):265–267, 1960.

[21] Guang-Bin Huang, Hongming Zhou, Xiaojian Ding, and Rui Zhang. Extreme learning machine for regression and multiclass classification. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, 42(2):513–529, 2011.

[22] Earl B Hunt, Janet Marin, and Philip J Stone. Experiments in induction. 1966.

[23] Iartificial. Diferencia entre entrenamiento y prueba en aprendizaje supervisado, 2023.

[24] H Jonathan. Connell and sridhar mahadevan. *ROBOT LEARNING*, 1993.

[25] Igor Kononenko. Machine learning for medical diagnosis: History, state of the art and perspective. *Artificial Intelligence in Medicine*, 23(1):89–109, 2001.

[26] John Paul Mueller and Luca Massaron. *Machine learning for dummies*. John Wiley & Sons, 2021.

[27] Wei Peng, Juhua Chen, and Haiping Zhou. An implementation of id3-decision tree learning algorithm. *From web. arch. usyd. edu. au/wpeng/DecisionTree2. pdf Retrieved date: May*, 13, 2009.

[28] Enrique J Carmona Suárez. Tutorial sobre máquinas de vectores soporte (svm). *Tutorial sobre Máquinas de Vectores Soporte (SVM)*, 1:1–12, 2014.

[29] Vladimir Vapnik and Vlamimir Vapnik. Statistical learning theory wiley. *New York*, 1(624):2, 1998.

[30] Vicente Ferranco González José García Cuenca Teresa Moral Fernández Víctor Aguilera Martín, Laura Castillero Ramírez. Aprendizaje por refuerzo aplicado al blackjack, 2023. Trabajo de Investigación Operativa.

[31] Qin Yang, Lin Tan, Ben-Qing Wu, Guo-Li Tian, Lu Xu, Jiang-Tao Yang, Jian-Hui Jiang, and Ru-Qin Yu. Beyond one-against-all (oaa) and one-against-one (oao): An exhaustive and parallel half-against-half (hah) strategy for multi-class classification and applications to metabolomics. *Chemometrics and Intelligent Laboratory Systems*, 204:104107, 2020.

[32] Mohd Izhan Mohd Yusoff. Machine learning: An overview. *Open Journal of Modelling and Simulation*, 12(3):89–99, 2024.