Universidad Juárez Autónoma de Tabasco

División Académica de Ciencias Básicas

Clasificación binaria lineal utilizando Python

TESIS PARA OBTENER EL TÍTULO DE:

Licenciado en Matemáticas

PRESENTA:

Luis Felipe López Guzmán

BAJO LA DIRECCIÓN DE:
Dra. Addy Margarita Bolívar Cimé

EN CODIRECCIÓN DE:
Dr. Edilberto Nájera Rangel

CUNDUACÁN, TABASCO A 2025

Declaración de Autoría y Originalidad

Declaración de Autoría y Originalidad

En la Ciudad de Cunduacán, el día 13 del mes 03 del año 2025, el que suscribe Luis Felipe López Guzmán alumna(o) del Programa de Lic. en Matemáticas con número de matrícula 181A11003, adscrito a la División Académica de Ciencias Básicas, de la Universidad Juárez Autónoma de Tabasco, como autor(a) (es) de la Tesis presentada para la obtención del (título, diploma o grado según sea el caso) Lic. en Matemáticas y titulada Clasificación binaria lineal utilizando Python dirigida por la Dra. Addy Margarita Bolívar Cimé y codirigida por el Dr. Edilberto Nájera Rangel.

DECLARO QUE: La Tesis es una obra original que no infringe los derechos de propiedad intelectual ni los derechos de propiedad industrial u otros, de acuerdo con el ordenamiento jurídico vigente, en particular, la LEY FEDERAL DEL DERECHO DE AUTOR (Decreto por el que se reforman y adicionan diversas disposiciones de la Ley Federal del Derecho de Autor del 01 de Julio de 2020 regularizando y aclarando y armonizando las disposiciones legales vigentes sobre la materia), en particular, las disposiciones referidas al derecho de cita. Del mismo modo, asumo frente a la Universidad cualquier responsabilidad que pudiera derivarse de la autoría o falta de originalidad o contenido de la Tesis presentada de conformidad con el ordenamiento jurídico vigente.

Villahermosa, Tabasco a 13 de 03 20 25.

Luis Felipe López Guzmán
Nombre y Firma

Autorización de impresión

UJAT
UNIVERSIDAD JUÁREZ AUTÓNOMA DE TABASCO
"ESTUDIO EN LA DUDA. ACCIÓN EN LA FE"

División Académica de Ciencias Básicas

2025 AÑO DE LA Mujer Indígena

DIRECCIÓN

Cunduacán, Tabasco; a 05 de marzo de 2025.

C. LUIS FELIPE LÓPEZ GUZMÁN
PAS. DE LA LICENCIATURA EN MATEMÁTICAS
PRESENTE

Por medio del presente, me dirijo a usted para hacer de su conocimiento que proceda a la impresión del trabajo titulado **"CLASIFICACIÓN BINARIA LINEAL UTILIZANDO PYTHON"**, dirigido por la Dra. Addy Margarita Bolívar Cimé con la codirección del Dr. Edilberto Nájera Rangel, bajo la modalidad de titulación por **TESIS**. La comisión de revisión conformada por el Dr. Fidel Ulín Montejo, Dr. Aroldo Pérez Pérez, Dra. Addy Margarita Bolívar Cimé, Dr. Jair Remigio Juárez y Dr. Miguel Ángel de la Rosa Castillo, liberó el documento en virtud de que reúne los requisitos para el **EXAMEN PROFESIONAL** correspondiente.

Sin otro particular, reciba usted un cordial saludo.

A T E N T A M E N T E

DIVISIÓN ACADÉMICA DE CIENCIAS BÁSICAS

DRA. HERMICENDA PÉREZ VIDAL
DIRECTORA

C.c.p. Archivo.

DIR´DRA.HPV/kfvg

Km.1 Carretera Cunduacán-Jalpa de Méndez, A.P. 24, C.P. 86690, Cunduacán, Tab., México.
Tel/Fax: (993) 3581500 Ext. 6702,6701 E-Mail: direccion.dacb@ujat.mx

www.ujat.mx

Carta de cesión de derecho

Carta de Cesión de Derechos

Villahermosa, Tabasco a 2025.

Por medio de la presente manifestamos haber colaborado como AUTOR(A) y/o AUTORES(RAS) en la producción, creación y/o realización de la obra denominada Clasificación binaria lineal utilizando Python.

Con fundamento en el artículo 83 de la Ley Federal del Derecho de Autor y toda vez que, la creación y/o realización de la obra antes mencionada se realizó bajo la comisión de la Universidad Juárez Autónoma de Tabasco; entendemos y aceptamos el alcance del artículo en mención, de que tenemos el derecho al reconocimiento como autores de la obra, y la Universidad Juárez Autónoma de Tabasco mantendrá en un 100% la titularidad de los derechos patrimoniales por un período de 20 años sobre la obra en la que colaboramos, por lo anterior, cedemos el derecho patrimonial exclusivo en favor de la Universidad.

COLABORADORES

Luis Felipe López Guzmán

ALUMNO(A) O EGRESADA(O),

DIRECTOR(A) Y CODIRECTOR(A)

TESTIGOS

Dra. Addy M. Bolívar Cimé Dr. Edilberto Nájera Rangel

Sistema antiplagio

LICENCIATURA - CLASIFICACIÓN BINARIA LINEAL UTILIZANDO PYTHON -LUIS FELIPE LÓPEZ GUZMÁN

INFORME DE ORIGINALIDAD

19%
ÍNDICE DE SIMILITUD

FUENTES PRIMARIAS

1	pdfcookie.com Internet	697 palabras — 3%
2	1library.co Internet	349 palabras — 2%
3	hdl.handle.net Internet	250 palabras — 1%
4	materias.unq.edu.ar Internet	249 palabras — 1%
5	congresos.ujat.mx Internet	205 palabras — 1%
6	juandomingofarnos.wordpress.com Internet	161 palabras — 1%
7	accesoabierto.uh.cu Internet	139 palabras — 1%
8	mail.python.org Internet	138 palabras — 1%
9	aprenderly.com Internet	136 palabras — 1%

Agradecimientos

Quiero agradecer primero que nada a Dios, que me prestó vida y salud para poder terminar esta tesis. Ahora quiero agradecer a mis padres Juvencio López Díaz y Yolanda Guzmán Díaz por brindarme apoyo tanto económicamente como mentalmente ellos fueron una fortaleza que me impulsaba cada día terminar esta tesis, mis más sinceros agradecimientos a mis asesores la Dra. Addy Margarita Bolívar Cimé y el Dr. Edilberto Nájera Rangel, que cada semana soportaban mi falta de conocimiento, la paciencia que ellos tuvieron a lo largo de la elaboración, sin duda es de agradecer, también por compartirme sus conocimientos que valen mucho.

A los miembros del comité evaluador, la Dra. Addy Margarita Bolívar Cimé, el Dr. Aroldo Peréz Peréz, Dr. Miguel Angel de la Rosa Castillo, Dr. Jair Remigio Juárez y el Dr. Fidel Uín Montero, por sus valiosas observaciones y sugerencias que enriquecieron significativamente esta investigación. Quiero agradecer a mi hermano Carlos Mario López Guzmán que me dio apoyo moral cuando un resultado se complicaba y su apoyo económico. Agradezco enormemente a la Universidad Juárez Autónoma de Tabasco por brindarme el apoyo académico. Por último quiero agradecer a mis amigos que de alguna forma me apoyaron con sus conocimientos y sugerencias para mejorar el contenido de esta tesis.

Índice general

Índice de tablas

Índice de figuras

Resumen

Esta tesis proporciona una introducción a los conceptos de clasificación, DLF y SVM, y demuestra su aplicación en problemas de clasificación del mundo real. La comparación de los métodos ofrece información valiosa sobre su idoneidad para diferentes tipos de datos y tareas de clasificación.

Palabras claves: Clasificación binaria, Aprendizaje automático, Reconocimiento de patrones, Discriminante Lineal de Fisher (DLF), Support Vector Machine (SVM).

Abstract

This thesis provides an introduction to the concepts of binary classification, DLF and SVM, and shows their application in real-world classification problems. The comparison of the methods offers valuable insights into their suitability for different data types and classification tasks.

Keywords: Binary classification, Machine learning, Pattern recognition, Fisher Linear Discriminant (DLF), Support Vector Machines (SVM).

Introducción

Esta tesis se enfoca en el problema de la clasificación, una tarea fundamental en el aprendizaje automático y el reconocimiento de patrones. La clasificación implica asignar objetos o instancias a categorías o clases predefinidas basándose en sus características o atributos. El documento explora dos enfoques principales para la clasificación: el Discriminante Lineal de Fisher (DLF) y Support Vector Machine (SVM).

El DLF es un método clásico que busca encontrar combinaciones lineales de características que maximicen la separación entre clases mientras minimizan la varianza dentro de cada clase. Asume que los datos de cada clase comparten la misma matriz de covarianza. Por otro lado, SVM se basa en encontrar un hiperplano que maximice el margen entre las clases, incluso si los datos no son linealmente separables.

En esta tesis se aplican estos métodos a tres conjuntos de datos del mundo real: el conjunto de datos de diagnóstico de cáncer de mama de Wisconsin, un conjunto de datos de correos electrónicos no deseados y un conjunto de datos de señales de sonar. El primero implica clasificar tumores como benignos o malignos basándose en características celulares extraídas de imágenes. El segundo conjunto de datos se utiliza para clasificar correos electrónicos como spam o no spam en función de la frecuencia de palabras y caracteres específicos. El tercer conjunto de datos es utilizado para clasificar minas y rocas mediante señales de un sonar activo en varios ángulos de visión y diferentes condiciones.

Se compara el desempeño de DLF y el de SVM en estos conjuntos de datos utilizando técnicas de validación cruzada para evaluar la precisión de la clasificación. Los resultados experimentales demuestran la eficacia de ambos métodos, destacando las ventajas y desventajas de cada uno en diferentes escenarios.

Marco teórico

En un problema de clasificación binaria tenemos un conjunto de datos de entrenamiento que consiste de N observaciones $(\boldsymbol{x}_i, y_i)$, con $i = 1, 2, \ldots, N$. Aquí $\boldsymbol{x}_i \in \mathbb{R}^d$ representa un vector, mientras que $y_i \in \{0, 1\}$ denota su correspondiente índice de clase o categoría. Las clases de los datos se pueden referir a dos especies de plantas, dos tipos de tumores, la presencia o ausencia de alguna enfermedad, tipos de correos electrónicos como correo basura y no basura, entre otros, ver [13]. Asumimos que los vectores $(\boldsymbol{x}_i, y_i)$ son vectores aleatorios independientes, distribuidos de acuerdo a alguna función de distribución desconocida $P(\boldsymbol{x}, y)$. El objetivo es utilizar los datos de entrenamiento para construir una regla de clasificación $\phi(\boldsymbol{x}) : \mathbb{R}^d \longrightarrow \{0, 1\}$, la cual pueda ser usada para predecir el índice de la clase para un nuevo dato $\boldsymbol{x}$.

Existen diversas reglas de clasificación binaria. Un tipo especial son las *reglas de clasificación binaria lineales*. Una regla de clasificación binaria lineal, con vector de pesos $\boldsymbol{b} \in \mathbb{R}^d$ y sesgo o intercepto $b \in \mathbb{R}$, está dada por

$$\phi(\boldsymbol{x}) = \begin{cases} 1, & \text{si } \boldsymbol{b}^\top \boldsymbol{x} + b > 0, \\ 0, & \text{en otro caso,} \end{cases}$$

para $\boldsymbol{x} \in \mathbb{R}^d$, donde $\boldsymbol{b}^\top$ denota el transpuesto de $\boldsymbol{b}$ y $\boldsymbol{b}$ es un vector columna. Esta regla clasifica un nuevo dato $\boldsymbol{x}$ en la clase 1 si el vector $\boldsymbol{x}$ se encuentra por arriba del hiperplano $\boldsymbol{b}^\top \boldsymbol{x} + b = 0$, y lo clasifica en la clase 0 en otro caso. Notar que en el caso de datos multivariados con distribución continua, la probabilidad de que un dato $\boldsymbol{x}$ pertenezca a este hiperplano es cero. En el caso de los datos que tienen distribución discreta, la asignación a una de las dos clases, de un nuevo dato que pertenezca al hiperplano, se puede hacer aleatoriamente.

Dos métodos populares de construcción de reglas de clasificación binaria lineales son el *Discriminante Lineal de Fisher* y *Support Vector Machine*, ver [7], [13] y [12].

Justificación

Los métodos de clasificación binaria lineal son una herramienta muy útil en el análisis de datos multivariados. En el área de *aprendizaje automático (machine learning)*, la cual tiene actualmente mucho auge, la clasificación es una técnica de *aprendizaje supervisado*. Los métodos de clasificación binaria son aplicados en el análisis de datos multivariados que surgen en diversos campos, tales como genómica, medicina, riesgo, reconocimiento de imágenes, tecnología de la información, entre otros, ver [13].

Las reglas de clasificación binarias lineales han sido de interés en la Estadística desde que Ronald Aymer Fisher las introdujo en 1936, siendo el Discriminante Lineal de Fisher (DLF) una de las más populares y utilizadas, ver [13]. Otra metodología de clasificación binaria lineal que ha surgido en la era moderna es Support Vector Machine (SVM), propuesta en 1992 por Boser, Guyon y Vapnik en [3], la cual tiene un desempeño tan bueno como el DLF, e incluso en algunos casos es superior. Por otro lado, SVM tiene la ventaja de que puede ser utilizado con datos de dimensión alta, es decir, datos multivariados con dimensión mayor o igual al tamaño de la muestra, mientras que el DLF únicamente puede utilizarse en el caso clásico, donde la dimensión es menor al tamaño de la muestra.

El lenguaje de programación Python se encuentra entre los más utilizados en el área de aprendizaje automático, por lo que es importante explicar la forma en que se pueden implementar algunos métodos de clasificación binaria lineal en este lenguaje, así como ejemplificar su uso en la clasificación de datos multivariados reales.

Pregunta de investigación

¿Cuál es la efectividad comparativa de los métodos Support Vector Machine (SVM) y el Discriminante Lineal de Fisher (DLF) en la clasificación binaria de conjuntos de datos multivariados?

Supuesto

El método Support Vector Machine (SVM) lineal ofrece una mayor precisión que el Discriminante Lineal de Fisher (DLF) en la clasificación binaria de conjuntos de datos multivariados, especialmente en escenarios con mayor variabilidad en las características, debido a su capacidad para maximizar el margen entre clases.

Objetivo general

El objetivo de esta tesis es presentar la teoría básica de algunos métodos de clasificación binaria lineal, ilustrar su implementación en el lenguaje de programación Python y mostrar algunos ejemplos de aplicación con datos encontrados en la literatura.

Objetivos específicos

I. Presentar algunos métodos de clasificación binaria lineal.

II. Explicar algunas rutinas en el lenguaje Python para la implementación de los métodos de clasificación binaria lineal.

III. Mostrar algunos ejemplos de clasificación binaria de datos reales encontrados en la literatura, así como el análisis de la eficiencia de los métodos de clasificación.

Parte I

Metodología

Capítulo 1

Preliminares

En este capítulo se presenta una breve introducción a Python y las librerías que se utilizan en el desarrollo de aplicaciones. También se proporcionan definiciones y resultados básicos de la teoría de Probabilidad que serán empleados en esta tesis, los cuales en su mayoría fueron tomados de [5] y [11].

1.1. Python

En esta sección daremos una breve introducción al lenguaje de programación *Python*, uno de los lenguajes más utilizados en el aprendizaje automático, ver [14] y [21]. Describiremos también *Scikit-learn (sklearn)* de Python, que es muy útil para llevar a cabo tareas de clasificación, y que será utilizada en esta tesis. Cabe mencionar que Python puede ser descargado de `https://www.python.org/downloads/`.

1.1.1. Uso de Python

Python es un lenguaje de programación muy utilizado para la ciencia de datos y, por lo tanto, cuenta con una gran cantidad de bibliotecas adicionales de código abierto. Aunque el rendimiento de los lenguajes interpretados, como Python, para tareas de cálculo intensivo es inferior al de los lenguajes de programación de bajo nivel, se han desarrollado bibliotecas de extensión para realizar operaciones rápidas y vectorizadas en matrices multidimensionales. Para las tareas de programación de aprendizaje automático, trabajamos principalmente con la biblioteca Scikit-learn, que es actualmente una de las bibliotecas de aprendizaje automático de código abierto más populares y accesibles.

Python está disponible para los tres principales sistemas operativos: Windows, macOS y Linux. El instalador, así como la documentación, se pueden descargar desde el sitio web oficial de Python. Una vez instalado Python, podemos ejecutar *pip* desde la terminal para instalar paquetes adicionales; por ejemplo, para instalar la librería de NumPy usamos,

```
pip install numpy
```

de la misma forma para las demás bibliotecas. En este trabajo, además de Scikit-learn, utilizaremos los paquetes NumPy, Pandas y Matplotlib.

1.1.2. Scikit-learn

Scikit-learn (sklearn) es una biblioteca de código abierto para el aprendizaje automático en Python. Proporciona una amplia gama de algoritmos para clasificación, regresión, agrupamiento y reducción de dimensionalidad, junto con herramientas para preprocesamiento de datos, selección de modelos y evaluación. Scikit-learn está construido sobre la biblioteca científica NumPy de Python.

1.1.3. NumPy

NumPy es uno de los paquetes fundamentales para la computación científica en Python. Contiene funcionalidades para arreglos multidimensionales, funciones matemáticas de alto nivel, como generar números aleatorios. En Scikit-learn, el arreglo de NumPy es la estructura de datos fundamental. Scikit-learn recibe los datos en forma de arreglos de NumPy. Cualquier dato que se utilice deberá ser convertido a un arreglo de NumPy. La funcionalidad central de NumPy es la clase *ndarray*, un arreglo multidimensional (n-dimensional). Todos los elementos del arreglo deben ser del mismo tipo. Un ejemplo de arreglo de NumPy es el siguiente:

```
import numpy as np
x = np.array([[1, 2, 3], [4, 5, 6]])
print("x:\n{}".format(x))

#Salida[2]:
x:
[[1 2 3]
[4 5 6]]
```

1.1.4. Pandas

Pandas es una biblioteca de Python para el manejo y análisis de datos. Está construida alrededor de una estructura de datos llamada DataFrame que está modelada a partir del DataFrame del software R. En pocas palabras, un DataFrame de pandas es una tabla, similar a una hoja de cálculo de Excel. Pandas proporciona una gran variedad de métodos para modificar y operar en esta tabla; en particular, permite consultas tipo SQL y uniones de tablas. A diferencia de NumPy, que requiere que todas las entradas en un array sean del mismo tipo, Pandas permite que cada columna tenga un tipo separado (por ejemplo: enteros, fechas, números de punto flotante y cadenas de texto). Otra herramienta valiosa que proporciona Pandas es su

capacidad para ingerir datos de una gran variedad de formatos de archivo y bases de datos, como SQL, archivos de Excel y archivos de valores separados por comas (CSV). Aquí se pone un pequeño ejemplo de cómo crear un DataFrame usando un diccionario:

```
import pandas as pd
# crear un conjunto de datos simple de personas
data = {'Name': ["John", "Anna", "Peter", "Linda"],
        'Location' : ["New York", "Paris", "Berlin", "London"],
        'Age' : [24, 13, 53, 33]
}
data_pandas = pd.DataFrame(data)
display(data_pandas)
```

Esto produce el siguiente resultado:

	Name	Location	Age
0	John	New York	24
1	Anna	Paris	13
2	Peter	Berlin	53
3	Linda	London	33

Figura 1.1: Ejemplo de uso de la librería Pandas.

1.1.5. Matplotlib

Matplotlib es la principal biblioteca de gráficos científicos en Python. Proporciona funciones para crear visualizaciones de calidad de publicación, como gráficos de líneas, histogramas, gráficos de dispersión, entre otros. Usaremos Matplotlib para todas nuestras visualizaciones. Por ejemplo, el siguiente código produce la gráfica de la función $f(x) = \text{sen}(x)$ en el intervalo $[-10, 10]$.

```
import matplotlib.pyplot as plt
import numpy as np
# Genera una secuencia de números del -10 al 10 con 100 pasos intermedios
x = np.linspace(-10, 10, 100)
# Crea una segunda matriz usando seno
```

```
y = np.sin(x)
# Grafica la función
plt.plot(x, y, marker="x")
```

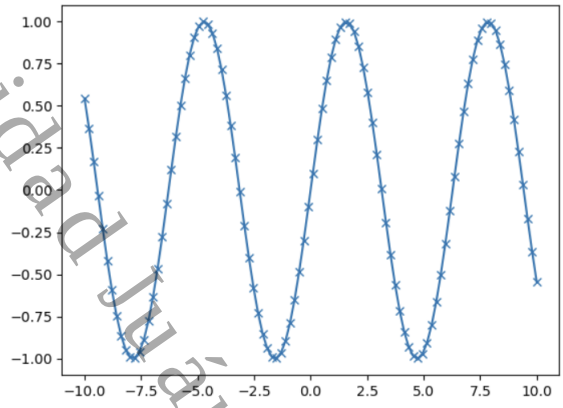

Figura 1.2: Gráfica de la función $f(x) = \text{sen}(x)$ utilizando la librería de Matplotlib y NumPy.

1.2. Teoría de conjuntos

Aunque el término conjunto no está definido en Matemáticas, por lo que es necesaria una base axiomática que regule su uso, aquí se asume la idea intuitiva de considerar un conjunto como una colección de elementos, y se acepta la existencia y unicidad del *conjunto vacío*, el cual se denota por $\emptyset$, que carece de elementos.
Dado un conjunto A, para decir que un elemento x *pertenece al conjunto* A, o que x *es un elemento de* A, se escribe $x \in A$. Similarmente, $x \notin A$ significa que x *no pertenece al conjunto* A, o que x *no es un elemento de* A, ver [11] y [17].

Definición 1.2.1. Sean A y B conjuntos.

1. El conjunto A es un *subconjunto* del conjunto B, denotado por $A \subset B$, si todo elemento de A es también un elemento de B.

2. El conjunto A es un *subconjunto propio* del conjunto B, denotado por $A \subsetneq B$, si $A \subset B$ y existe $x \in B$ tal que $x \notin A$.

Definición 1.2.2. Dos conjuntos A y B son *iguales*, denotado por $A = B$, si contienen exactamente los mismos elementos, es decir,

$$A = B \text{ si y sólo si } A \subset B \text{ y } B \subset A. \tag{1.1}$$

Con las definiciones anteriores podemos definir las operaciones usuales de subconjuntos A y B de un conjunto *universo* S.

1. La *unión* de A y B, denotada por $A \cup B$, es el conjunto de elementos de S que pertenecen a A o a B:

$$A \cup B = \{x \in S : x \in A \text{ o } x \in B\}.$$

2. La *intersección* de A y B, denotada por $A \cap B$, es el conjunto de todos los elementos de S que están en A y en B:

$$A \cap B = \{x \in S : x \in A \text{ y } x \in B\}.$$

3. El *complemento* de A, el cual se denota por A^c, es el conjunto de todos los elementos de S que no están en A:

$$A^c = \{x \in S : x \notin A\}.$$

Definición 1.2.3. El conjunto sin elementos es llamado el *conjunto vacío* y se denota por $\emptyset$.

Las demostraciones de los siguientes resultados son inmediatas a partir de las definiciones ya dadas.

Teorema 1.2.4. *Para cualesquiera tres conjuntos A, B y C, se cumplen las siguientes propiedades:*

a) Conmutatividad	$A \cup B = B \cup A$,
	$A \cap B = B \cap A$;
b) Asociatividad	$A \cup (B \cup C) = (A \cup B) \cup C$;
	$A \cap (B \cap C) = (A \cap B) \cap C$;
c) Leyes distributivas	$A \cap (B \cup C) = (A \cap B) \cup (A \cap C)$;
	$A \cup (B \cap C) = (A \cup B) \cap (A \cup C)$;
b) Leyes de De Morgan	$(A \cup B)^c = A^c \cap B^c$,
	$(A \cap B)^c = A^c \cup B^c$.

Las operaciones de unión e intersección pueden ser extendidas a colecciones infinitas de conjuntos. Si $A_1, A_2, A_3, \ldots$ es una colección de subconjuntos de S, entonces

$$\bigcup_{i=1}^{\infty} A_i = \{x \in S : x \in A_i, \text{ para algún } i\},$$

$$\bigcap_{i=1}^{\infty} A_i = \{x \in S : x \in A_i, \text{ para todo } i\}.$$

También es posible definir uniones e intersecciones sobre colecciones arbitrarias de conjuntos. Si Γ es un conjunto de índices, entonces

$$\bigcup_{a\in\Gamma} A_a = \{x \in S : x \in A_a, \text{ para algún } a\},$$

$$\bigcap_{a\in\Gamma} A_a = \{x \in S : x \in A_a, \text{ para todo } a\}.$$

1.3. Teoría de la probabilidad

Uno de los objetivos de un estadístico es extraer conclusiones sobre una población de objetos mediante la realización de un experimento. Para eso, primero hay que identificar los posibles resultados de un espacio muestral.

Definición 1.3.1. El conjunto S de todos los resultados posibles de un experimento se denomina *espacio muestral* del experimento.

Definición 1.3.2. Un *evento* es un subconjunto del espacio muestral S, es decir, un subconjunto de posibles resultados del experimento.

Para simplificar la notación usaremos letras mayúsculas para referirnos a eventos. Sea A un evento de S. Decimos que el evento A *ocurre* si el resultado del experimento está en el conjunto A.

Definición 1.3.3. Dos eventos A y B son *ajenos* (o *mutuamente excluyentes*) si $A \cap B = \emptyset$. Los eventos $A_1, A_2, \ldots$ son *ajenos por pares* (o *mutuamente excluyentes*) si $A_i \cap B_j = \emptyset$ para cualesquiera $i \neq j$.

Los siguientes ejemplos son tomados de [19].

Ejemplo 1.3.4. Si un experimento consiste en lanzar un dado y observar el número que aparece en la cara superior, entonces claramente el espacio muestral es el conjunto $S = \{1, 2, 3, 4, 5, 6\}$. Como ejemplo de un evento para este experimento podemos definir el conjunto $A = \{2, 4, 6\}$, que corresponde al suceso de obtener como resultado un número par. Si al lanzar el dado una vez se obtiene el número 4, decimos entonces que se observó la ocurrencia del evento A, y si se obtiene, por ejemplo, el resultado 1, decimos que no se observó la ocurrencia del evento A.

Ejemplo 1.3.5. Considere el experimento aleatorio de participar en un juego de lotería. Suponga que hay un millón de números en esta lotería y un jugador participa con un boleto. ¿Cuál es un posible espacio muestral para este experimento si únicamente uno de los números es el ganador? Naturalmente, al jugador le interesa conocer su suerte en este juego y puede proponer como espacio muestral el conjunto S={"ganar", "perder"}. Sin embargo puede también tomarse como espacio muestral el conjunto que contiene a todos los números participantes, es decir, $S = \{1, 2, \ldots, 10^6\}$. Este ejemplo sencillo muestra que el espacio muestral de un experimento aleatorio no es único y depende del interés del observador.

1.3.1. Fundamento axiomático

A cada evento $A \subset S$ le queremos asociar un número entre 0 y 1, el cual será llamado la *probabilidad* de A, y se denotará por $P(A)$.

Definición 1.3.6. Una colección de subconjuntos $\mathcal{B}$ de S es llamada *sigma álgebra* (σ-álgebra), si satisface las siguientes tres propiedades:

1. $\emptyset \in \mathcal{B}$.
2. Si $A \in \mathcal{B}$, entonces $A^c \in \mathcal{B}$.
3. Si $A_1, A_2, \cdots \in \mathcal{B}$, entonces $\bigcup_{i=1}^{\infty} A_i \in \mathcal{B}$.

Definición 1.3.7. Si S es un espacio muestral y $\mathcal{B}$ es una σ-álgebra en S, entonces la pareja $(S, \mathcal{B})$ se llama *espacio medible*.

Ejemplo 1.3.8. Si S es un espacio muestral de un experimento aleatorio, no es difícil comprobar que las siguientes colecciones de subconjuntos de S son σ-álgebras:

1. $\mathcal{B} = \{\emptyset, S\}$.
2. $\mathcal{B} = \{A, A^c, \emptyset, S\}$, con $A \subseteq S$.
3. $\mathcal{B} = \mathcal{P}(S)$, donde $\mathcal{P}(S)$ es el *conjunto potencia* de S, es decir, el conjunto de todos los subconjuntos de S.

Ejemplo 1.3.9. La *σ-álgebra de Borel* de $\mathbb{R}$, denotada por $\mathcal{B}(\mathbb{R})$, se define como la mínima σ-álgebra de subconjuntos de $\mathbb{R}$ que contiene a todos los intervalos de la forma $(-\infty, x]$, en el sentido de que si $\mathcal{G}$ es cualquier σ-álgebra que contiene a estos intervalos, entonces $\mathcal{B}(\mathbb{R}) \subset \mathcal{G}$. Esto se escribe de la manera siguiente:

$$\mathcal{B}(\mathbb{R}) := \sigma\{(-\infty, x] : x \in \mathbb{R}\}.$$

A los elementos de $\mathcal{B}(\mathbb{R})$ se les llama *conjuntos de Borel, conjuntos Borel medibles o simplemente borelianos* de $\mathbb{R}$.

Nota 1.3.10. De aquí en adelante llamaremos eventos solo a los elementos de la σ-álgebra de interés del espacio muestral.

Definición 1.3.11. Dado un espacio muestral S y una σ-álgebra $\mathcal{B}$ asociada, una *medida de probabilidad* es una función P con dominio $\mathcal{B}$ que satisface:

1. $P(A) \geq 0$ para todo $A \in \mathcal{B}$.
2. $P(S) = 1$.

3. Si $A_1, A_2, \cdots \in \mathcal{B}$ son ajenos por pares, entonces

$$P\left(\bigcup_{i=1}^{\infty} A_i\right) = \sum_{i=1}^{\infty} P(A_i).$$

Las propiedades de la Definición 1.3.11 se llaman los *Axiomas de Probabilidad o Axiomas de Kolmogorov*.

Definición 1.3.12. La terna $(S, \mathcal{B}, P)$ es llamada *espacio de probabilidad*.

1.3.2. Cálculo de probabilidades

La demostración del siguiente teorema se puede consultar en [5].

Teorema 1.3.13. *Si P es una medida de probabilidad y A, B son conjuntos cualesquiera en $\mathcal{B}$, entonces*

1. $P(\emptyset) = 0$, *donde $\emptyset$ es el conjunto vacío;*
2. $P(A) \leq 1$;
3. $P(A^c) = 1 - P(A)$;
4. $P(B \cap A^c) = P(B) - P(A \cap B)$;
5. $P(A \cup B) = P(A) + P(B) - P(A \cap B)$;
6. *Si $A \subset B$, entonces $P(A) \leq P(B)$.*

Definición 1.3.14. Para un entero positivo n, el *factorial* de n, denotado por $n!$, es el producto de todos los enteros positivos menores o iguales a n,

$$n! = n \times (n-1) \times (n-2) \times \cdots \times 3 \times 2 \times 1.$$

Definición 1.3.15. Para enteros no negativos n y r, donde $n \geq r$, las *combinaciones* de n en r, denotadas por $\binom{n}{r}$, están dadas por

$$\binom{n}{r} = \frac{n!}{r!(n-r)!}.$$

Las combinaciones también se conocen como *coeficientes binomiales*. El siguiente ejemplo se tomó de [23].

Ejemplo 1.3.16. Considere el problema de seleccionar dos solicitantes para un trabajo, de un grupo de cinco solicitantes, e imagine que éstos varían en grado de competencia, 1 siendo el mejor, 2 el segundo mejor y así sucesivamente para 3, 4 y 5. Estas clasificaciones son por supuesto desconocidas para el empleador. Entonces el número de formas de seleccionar dos solicitantes de entre cinco, y por tanto el número total de puntos muestrales en S, es

$$\binom{5}{2} = \frac{5!}{2!3!} = 10.$$

1.4. Probabilidad condicional e independencia

Las probabilidades que se han visto hasta ahora son probabilidades incondicionales. Se definió un espacio muestral y se calcularon las probabilidades con respecto al espacio muestral. Sin embargo, a veces queremos actualizar el espacio muestral basados en nueva información de modo tal que actualicemos el cálculo de probabilidades, o calculemos *probabilidades condicionales*.

Definición 1.4.1. Si $A, B \subset S$ son dos eventos y $P(B) > 0$, entonces la *probabilidad condicional* de A dado B, denotada por $P(A|B)$, es

$$P(A|B) = \frac{P(A \cap B)}{P(B)}.$$

Observemos que en el cálculo de la probabilidad condicionada a B, este último se convierte en el nuevo espacio muestral: $P(B|B) = 1$. Nuestro espacio muestral original S ha sido actualizado a B. En particular, si A y B son ajenos, entonces $P(A \cap B) = 0$, de lo que se sigue que $P(A|B) = P(B|A) = 0$.

Los siguientes resultados se pueden consultar en [5] y [19].

Definición 1.4.2. Si $A_1, A_2, \ldots$ son ajenos por pares y $\bigcup_{i=1}^{\infty} A_i = S$, entonces la colección $A_1, A_2, \ldots$, es una partición de S.

Teorema 1.4.3. *(Regla de Bayes) Sea $A_1, A_2, \ldots$, una partición de un espacio muestral, y sea B cualquier evento. Entonces para cada $i = 1, 2, \ldots$,*

$$P(A_i|B) = \frac{P(B|A_i)P(A_i)}{\sum_{j=1}^{\infty} P(B|A_j)P(A_j)}.$$

Definición 1.4.4. Dos eventos A y B son estadísticamente independientes si

$$P(A \cap B) = P(A)P(B).$$

Si los eventos A y B son estadísticamente independientes, entonces se dirá simplemente que son *independientes*.

Teorema 1.4.5. *Si A y B son eventos independientes, entonces los siguientes pares de conjuntos también son independientes:*

1. *A y B^c,*

2. A^c *y* B,

3. A^c *y* B^c.

Definición 1.4.6. Se dice que los eventos $A_1, \ldots, A_n$ son *mutuamente independientes* si para cualquier subcolección $A_{i_1}, \ldots, A_{i_k}$ se tiene

$$P\left(\bigcap_{j=1}^{k} A_{i_j}\right) = \prod_{j=1}^{k} P(A_{i_j}).$$

1.5. Variables aleatorias

Definición 1.5.1. Una variable aleatoria X en un espacio de probabilidad $(S, \mathcal{B}, P)$, es una función X definida de S en $\mathbb{R}$ tal que

$$X^{-1}(B) \in \mathcal{B} \text{ para todo } B \in \mathcal{B}(\mathbb{R}),$$

donde $\mathcal{B}(\mathbb{R})$ es la σ-álgebra de Borel de los reales $\mathbb{R}$.

Denotaremos por $(X \in B)$ al conjunto $X^{-1}(B)$.

Definición 1.5.2. Si X es una variable aleatoria en $(S, \mathcal{B}, P)$, la medida de probabilidad P_X inducida por X en $\mathcal{B}(\mathbb{R})$ está dada por

$$P_X(B) = P(X \in B), \quad \text{para todo } B \in \mathcal{B}(\mathbb{R}).$$

Observación 1.5.3. Se comprueba que en efecto P_X es una medida de probabilidad en $\mathcal{B}(\mathbb{R})$.

Los siguientes ejemplos son tomados de [19].

Ejemplo 1.5.4. Consideremos el experimento que consiste en lanzar una moneda justa tres veces. Defínase la variable aleatoria X como el número de soles obtenidos en los tres lanzamientos. Una enumeración completa de los valores de X para cada punto del espacio muestral es

s	SSS	SSC	SCS	CSS	CCS	CSC	SCC	CCC
$X(s)$	3	2	2	2	1	1	1	0

El rango de la variable aleatoria X es $\mathcal{X} = \{0, 1, 2, 3\}$. Supongamos que todos los puntos del espacio muestral tienen probabilidad $\frac{1}{8}$ de ocurrir, por lo que la función de probabilidad inducida sobre $\mathcal{X}$ está dada como

x	0	1	2	3
$P(X = x)$	$\frac{1}{8}$	$\frac{3}{8}$	$\frac{3}{8}$	$\frac{1}{8}$

Ejemplo 1.5.5. Sea $c \in \mathbb{R}$ una constante y sea $(S, \mathcal{B}, P)$ un espacio de probabilidad. Para cualquier experimento aleatorio con espacio muestral S se puede definir la función constante $X(s) = c$. Así, cualquier resultado del experimento aleatorio produce, a través de la función X, el número c. Véase la figura 1.3. Decimos entonces que X es la variable aleatoria constante c y se puede verificar que para cualquier conjunto $A \in \mathcal{B}(\mathbb{R})$,

$$P(X \in A) = \begin{cases} 1 \text{ si } c \in A, \\ 0 \text{ si } c \notin A. \end{cases}$$

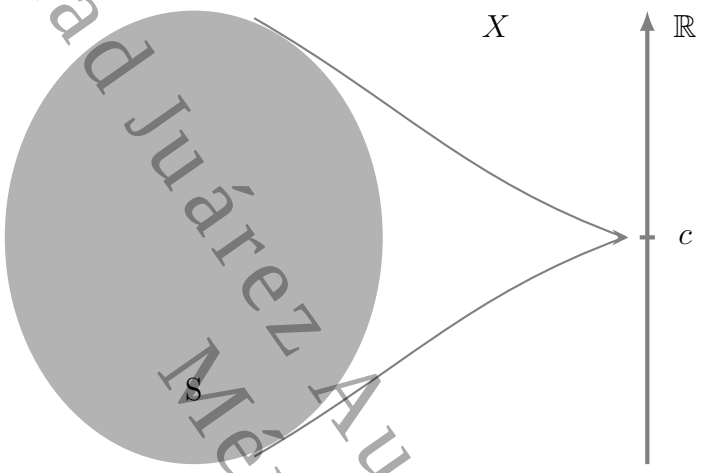

Figura 1.3: Variable aleatoria constante.

1.6. Función de distribución

Definición 1.6.1. La *función de distribución acumulada (fda)* de una variable aleatoria X, denotada por $F_X(x)$, está dada como

$$F_X(x) = P(X \leq x), \quad \text{para todo } x.$$

Los siguientes resultados pueden consultarse en [5].

Teorema 1.6.2. *La función $F(x)$ es una función de distribución acumulada si y sólo si cumple las siguientes condiciones:*

1. $\lim_{x \to -\infty} F(x) = 0$ *y* $\lim_{x \to \infty} F(x) = 1$.
2. $F(x)$ *es una función no decreciente.*
3. $F(x)$ *es continua por la derecha, es decir, para cada número* x_0,

$$\lim_{x \to x_0^+} F(x) = F(x_0).$$

Definición 1.6.3. Una variable aleatoria X es continua si $F_X(x)$ es una función continua. Es discreta si $F_X(x)$ es una función escalonada.

Definición 1.6.4. Las variables aleatorias X e Y están idénticamente distribuidas si para cada conjunto $A \in \mathcal{B}$, $P(X \in A) = P(Y \in A)$.

Teorema 1.6.5. *Las siguientes afirmaciones son equivalentes:*

1. *Las variables aleatorias X y Y están idénticamente distribuidas.*
2. *$F_X(x) = F_Y(x)$ para cada x.*

1.7. Funciones masa y de densidad

Definición 1.7.1. La *función masa de probabilidad (fmd)* de una variable aleatoria discreta X está dada por

$$f_X(x) = P(X = x).$$

Definición 1.7.2. Una *función de densidad de probabilidad (fdp)*, $f_X(x)$, de una variable aleatoria continua X es una función no negativa que satisface

$$F_X(x) = \int_{-\infty}^{x} f_X(t)dt.$$

Si existe una fdp para una variable aleatoria X, se dice que X es *absolutamente continua*.

1.8. Vectores aleatorios

Hemos dado la definición de variable aleatoria, pero solo para el caso univariado. En esta sección se definirán los vectores aleatorios n-dimensionales. Los resultados de esta sección son tomados de [2], [5] y [19].

Definición 1.8.1. Sea $\mathcal{B}_j$ una σ-álgebra de subconjuntos de S_j, $j = 1, 2, \ldots, n$, y sea $S = S_1 \times S_2 \times \cdots \times S_n$. Un rectángulo medible en S es un conjunto $A = A_1 \times A_2 \times \cdots \times A_n$, donde $A_j \in \mathcal{B}_j$ para cada $j = 1, 2, \ldots, n$. La σ- álgebra más pequeña que contiene a todos los rectángulos medibles, es la llamada *σ-álgebra producto*, denotada por $\mathcal{B}_1 \otimes \mathcal{B}_2 \otimes \cdots \otimes \mathcal{B}_n$. Si $\mathcal{B} = \mathcal{B}_j$, $j = 1, \ldots, n$, la σ-álgebra producto se denota como $\mathcal{B}^n$.

Observación 1.8.2. Si en la Definición 1.8.1, $S_j = \mathbb{R}$ y $\mathcal{B}_j = \mathcal{B}(\mathbb{R})$, $j = 1, \ldots, n$, entonces la σ-álgebra producto se llama la σ-álgebra de Borel de $\mathbb{R}^n$, y se denota como $\mathcal{B}(\mathbb{R}^n)$.

Definición 1.8.3. Sean $(S_1, \mathcal{B}_1)$ y $(S_2, \mathcal{B}_2)$ espacios medibles. Se dice que una función $f : S_1 \to S_2$ es medible con respecto a $\mathcal{B}_1$ y $\mathcal{B}_2$, si para todo $B \in \mathcal{B}_2$, $f^{-1}(B) \in \mathcal{B}_1$. Si $S_2 = \mathbb{R}^n$ y $\mathcal{B}_2 = \mathcal{B}(\mathbb{R}^n)$, entonces se dice que f es *Borel medible*.

Definición 1.8.4. Un *vector aleatorio n-dimensional* sobre un espacio de probabilidad $(S, \mathcal{B}, P)$ es una función Borel medible de S a $\mathbb{R}^n$.

Si $X = (X_1, X_2, ..., X_n)$ es un vector aleatorio, entonces cada X_i es una variable aleatoria, $i = 1, ..., n$. Decimos que un vector aleatorio es discreto o continuo, si todas las variables aleatorias que lo conforman son discretas o continuas, respectivamente.

Definición 1.8.5. Sea (X, Y) un vector aleatorio discreto bivariado. Entonces la función $f : \mathbb{R}^2 \to \mathbb{R}$ definida por $f(x, y) = P(X = x, Y = y)$ es llamada *función masa de probabilidad conjunta* o (*fmp conjunta*) de (X, Y).

Teorema 1.8.6. *Sea (X, Y) un vector aleatorio discreto bivariado con fmp conjunta $f_{X,Y}(x, y)$. Entonces las funciones masa de probabilidad marginales de X y Y, $f_X(x) = P_X(x)$ y $f_Y(y) = P_Y(y)$, están dadas por*

$$f_X(x) = \sum_{y \in \mathbb{R}} f_{X,Y}(x, y) \text{ y } f_Y(y) = \sum_{x \in \mathbb{R}} f_{X,Y}(x, y).$$

Definición 1.8.7. Una función $f : \mathbb{R}^2 \to \mathbb{R}$, es llamada *función de densidad de probabilidad conjunta* o (*fdp conjunta*) de un vector aleatorio continuo bivariado (X, Y), si para cada $A \in \mathcal{B}(\mathbb{R}^2)$,

$$P((X, Y) \in A) = \int \int_A f(x, y) dx dy.$$

1.9. Distribución condicional e independencia

Definición 1.9.1. Sea (X, Y) un vector aleatorio bivariado discreto con función masa de probabilidad conjunta $f(x, y)$ y funciones masa de probabilidad marginales $f_X(x)$ y $f_Y(y)$. Para cualquier x tal que $P(X = x) = f_X(x) > 0$, la *función masa de probabilidad condicional* de Y dado $X = x$, es la función de y denotada por $f(y|x)$ y definida como

$$f(y|x) = P(Y = y|X = x) = \frac{f(x, y)}{f_X(x)}.$$

De manera análoga se define la fmp condicional de X dado $Y = y$, como

$$f(x|y) = P(X = y|Y = y) = \frac{f(x, y)}{f_Y(y)}.$$

Definición 1.9.2. Sea (X, Y) un vector aleatorio bivariado continuo con función de densidad de probabilidad conjunta $f(x, y)$, y funciones de densidad de probabilidad marginales $f_X(x)$ y $f_Y(y)$. Para cualquier x tal que $f_X(x) > 0$, la *función de densidad de probabilidad condicional* de Y dado $X = x$, es la función de y denotada por $f(y|x)$ y definida por

$$f(y|x) = \frac{f(x, y)}{f_X(x)}.$$

De forma similar la fdp condicional de X dado $Y = y$, es

$$f(x|y) = \frac{f(x, y)}{f_Y(y)}.$$

Se puede verificar que las funciones dadas en las Definiciones 1.9.1 y 1.9.2, en efecto, son fmp y fdp para una variable aleatoria discreta o continua, respectivamente.

Definición 1.9.3. Sea (X, Y) un vector aleatorio bivariado con fdp o fmp conjunta $f(x, y)$, y funciones de densidad de probabilidad o funciones masa de probabilidad marginales $f_X(x)$ y $f_Y(y)$. Entonces X y Y son llamadas *variables aleatorias independientes*, si para cualesquiera $x \in \mathbb{R}$ y $y \in \mathbb{R}$,

$$f(x, y) = f_X(x) f_Y(y).$$

Lema 1.9.4. *Sea (X, Y) un vector aleatorio bivariado con fdp o fmp conjunta $f(x, y)$. Entonces X y Y son variables aleatorias independientes si y sólo si existen funciones $g(x)$ y $h(y)$ tales que para cualesquiera $x \in \mathbb{R}$ y $y \in \mathbb{R}$,*

$$f(x, y) = g(x)h(y).$$

Teorema 1.9.5. *Sean X y Y variables aleatorias independientes. Entonces para cualesquiera $A, B \in \mathcal{B}(\mathbb{R})$, $P(X \in A, Y \in B) = P(X \in A)P(Y \in B)$; es decir, los eventos $(X \in A)$ y $(Y \in B)$ son eventos independientes.*

Observación 1.9.6. Para el caso de un vector aleatorio de dimensión n las definiciones anteriores se generalizan de manera natural, y se obtienen resultados análogos.

1.10. Distribución gaussiana bivariada

La distribución bivariada de Gauss es una generalización de la distribución normal unidimensional; la gráfica de su función de densidad de probabilidad se caracteriza por su parecido con la forma de una campana.

Definición 1.10.1. Se dice que una variable aleatoria X univariada de valor real tiene distribución gaussiana (o normal) con media μ y varianza σ^2, denotada por $X \sim \mathcal{N}(\mu, \sigma^2)$, si su función de densidad es

$$f_X(x) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{1}{2\sigma^2}(x - \mu)^2\right) \quad \forall x \in \mathbb{R},$$

donde $-\infty < \mu < \infty$ y $\sigma > 0$.

Definición 1.10.2. Se dice que el vector aleatorio n-dimensional $\boldsymbol{X}$ tiene distribución gaussiana (o normal) multivariada con media $\boldsymbol{\mu} = (\mu_1, \mu_2, \ldots, \mu_n)^\top$ y matriz de covarianza $\boldsymbol{\Sigma}$ definida positiva y simétrica de tamaño $n \times n$, si su función de densidad es

$$f_{\boldsymbol{X}}(x_1, \ldots, x_n) = \frac{1}{(2\pi)^{n/2}|\boldsymbol{\Sigma}|^{1/2}} \exp\left(-\frac{1}{2}(\boldsymbol{x}-\boldsymbol{\mu})^\top \boldsymbol{\Sigma}^{-1}(\boldsymbol{x}-\boldsymbol{\mu})\right),$$

donde $|\boldsymbol{\Sigma}|$ es el determinante de la matriz $\boldsymbol{\Sigma}$ y $\boldsymbol{x} = (x_1, x_2, \ldots, x_n)^\top$.

La distribución normal multivariada n-dimensional con media $\boldsymbol{\mu}$ y matriz de covarianza $\boldsymbol{\Sigma}$ se denota por $\mathcal{N}_n(\boldsymbol{\mu}, \boldsymbol{\Sigma})$.

1.10.1. Casos especiales

Si $\boldsymbol{\Sigma} = \sigma^2 \boldsymbol{I}_n$, entonces la función de densidad gaussiana multivariada es

$$f_{\boldsymbol{X}}(\boldsymbol{x}) = (2\pi)^{-n/2}\sigma^{-n} \exp\left(-\frac{1}{2\sigma^2}(\boldsymbol{x}-\boldsymbol{\mu})^\top(\boldsymbol{x}-\boldsymbol{\mu})\right),$$

y es la llamada *densidad esférica de Gauss*, ya que la ecuación $(\boldsymbol{x}-\boldsymbol{\mu})^\top(\boldsymbol{x}-\boldsymbol{\mu}) = a^2$ es una esfera n-dimensional centrada en $\boldsymbol{\mu}$. En general, la ecuación $(\boldsymbol{x}-\boldsymbol{\mu})^\top\Sigma^{-1}(\boldsymbol{x}-\boldsymbol{\mu}) = a^2$ es un elipsoide centrado en $\boldsymbol{\mu}$, donde $\boldsymbol{\Sigma}$ determina la orientación y forma del elipsoide.

Cuando $n = 2$, la densidad gaussiana multivariada puede escribirse de forma explícita. Supongamos que

$$\boldsymbol{X} = (X_1, X_2)^\top \sim \mathcal{N}_2(\boldsymbol{\mu}, \boldsymbol{\Sigma}),$$

donde

$$\boldsymbol{\mu} = (\mu_1, \mu_2)^\top,\ \boldsymbol{\Sigma} = \begin{pmatrix} \sigma_{11} & \sigma_{12} \\ \sigma_{21} & \sigma_{22} \end{pmatrix} = \begin{pmatrix} \sigma_1^2 & \rho\sigma_1\sigma_2 \\ \rho\sigma_1\sigma_2 & \sigma_2^2 \end{pmatrix},$$

σ_1^2 es la varianza de X_1, σ_2^2 la varianza de X_2,

$$\rho = \frac{\operatorname{cov}(X_1, X_2)}{\sqrt{\operatorname{var}(X_1)\cdot \operatorname{var}(X_2)}} = \frac{\sigma_{12}}{\sigma_1\sigma_2}$$

es la correlación entre X_1 y X_2. De esto se sigue que

$$|\boldsymbol{\Sigma}| = (1-\rho^2)\sigma_1^2\sigma_2^2$$

y

$$\boldsymbol{\Sigma}^{-1} = \frac{1}{1-\rho^2}\begin{pmatrix} \dfrac{1}{\sigma_1^2} & \dfrac{-\rho}{\sigma_1\sigma_2} \\ \dfrac{-\rho}{\sigma_1\sigma_2} & \dfrac{1}{\sigma_2^2} \end{pmatrix}.$$

Así, la función de densidad gaussiana bivariada de $\boldsymbol{X}$ es

$$f_{\boldsymbol{X}}(\boldsymbol{x}) = \frac{1}{2\pi\sigma_1\sigma_2\sqrt{1-\rho^2}} e^{-\frac{1}{2}Q},$$

donde

$$\begin{aligned}
Q &= (\boldsymbol{x}-\boldsymbol{\mu})^\top \Sigma^{-1} (\boldsymbol{x}-\boldsymbol{\mu}) \\
&= (x_1-\mu_1, x_2-\mu_2) \frac{1}{1-\rho^2} \begin{pmatrix} \frac{1}{\sigma_1^2} & \frac{-\rho}{\sigma_1\sigma_2} \\ \frac{-\rho}{\sigma_1\sigma_2} & \frac{1}{\sigma_2^2} \end{pmatrix} \begin{pmatrix} x_1-\mu_1 \\ x_2-\mu_2 \end{pmatrix} \\
&= \frac{1}{1-\rho^2} \begin{pmatrix} \frac{x_1-\mu_1}{\sigma_1^2} - \frac{\rho(x_2-\mu_2)}{\sigma_1\sigma_2} \\ -\frac{\rho(x_1-\mu_1)}{\sigma_1\sigma_2} + \frac{x_2-\mu_2}{\sigma_2^2} \end{pmatrix} \begin{pmatrix} x_1-\mu_1 \\ x_2-\mu_2 \end{pmatrix} \\
&= \frac{1}{1-\rho^2} \left[\left(\frac{x_1-\mu_1}{\sigma_1^2} - \frac{\rho(x_2-\mu_2)}{\sigma_1\sigma_2} \right)(x_1-\mu_1) - \left(\frac{\rho(x_1-\mu_1)}{\sigma_1\sigma_2} + \frac{x_2-\mu_2}{\sigma_2^2} \right)(x_2-\mu_2) \right] \\
&= \frac{1}{1-\rho^2} \left[\left(\frac{x_1-\mu_1}{\sigma_1} \right)^2 - 2\rho \left(\frac{(x_1-\mu_1)(x_2-\mu_2)}{\sigma_1\sigma_2} \right) + \left(\frac{x_2-\mu_2}{\sigma_2} \right)^2 \right].
\end{aligned}$$

Capítulo 2

Métodos de clasificación

En este capítulo definiremos lo que es un clasificador y una clasificación. Se estudiarán los métodos de clasificación Discriminante Lineal de Fisher y Support Vector Machine. Los resultados y conceptos en este capítulo fueron tomados de [13].

Definición 2.0.1. Un conjunto de aprendizaje (o entrenamiento),

$$\mathcal{L} = \{(\boldsymbol{x}_i, y_i) : i = 1, \ldots, n\},$$

donde $\boldsymbol{x}_i \in \mathbb{R}^d$ y y_i es una etiqueta que toma los valores $1, 2, \ldots, K$, es un conjunto de datos que sirven para entrenar algún modelo.

Definición 2.0.2. Cualquier función $f : \mathbb{R}^d \to \{1, 2, \ldots, K\}$ es un *clasificador*, *regla de clasificación* o *función de decisión*.

Sea $\mathcal{L}$ un conjunto de aprendizaje, y supongamos que se sabe que cada observación procede de una de K clases predefinidas que tienen características similares. Las clases pueden ser, por ejemplo, especies de plantas, presencia o ausencia de una condición médica específica, diferentes tipos de tumores, opiniones sobre la censura en internet, o si un mensaje de correo electrónico es spam o no. Dado que las clases son conocidas, para distinguirlas se asocia una *etiqueta de clase* única (o valor de salida) a cada clase. Las observaciones se describen como *observaciones etiquetadas*. En cada una de estas situaciones hay dos objetivos principales:

Discriminación: Utilizar la información de un conjunto de aprendizaje de observaciones etiquetadas para construir un clasificador, que separará las clases predefinidas tanto como sea posible.

Clasificación: Dado un conjunto de medidas de una nueva observación no etiquetada, utilizar el clasificador para predecir la clase de esa observación.

En esta sección nos ocuparemos del primer objetivo, la discriminación. En la literatura sobre aprendizaje automático, la discriminación y la clasificación se describen como técnicas de aprendizaje supervisado; juntas, también se denominan tareas de *predicción de clases*. Los objetivos mencionados dependen de la información proporcionada por las variables de entrada.

Cuando se tienen dos clase se construye un solo clasificador binario con los datos de entrenamiento. Cuando se tienen más de dos clases, se pueden combinar varios clasificadores binarios para construir un clasificador multiclase, que al proporcionarle un nuevo dato de alguna de las clases prediga la clase a la cual pertenece.

2.1. Clases y características

Supongamos que una población P está dividida en K clases, grupos o subpoblaciones no ordenados, que se denotan por $\Pi_1, \Pi_2, \ldots, \Pi_K$. Además, cada elemento de P se clasifica en una (y sólo una) de esas clases. Las mediciones de una muestra de elementos se utilizarán para ayudar a asignar futuros elementos no clasificados a una de las clases consideradas. El vector aleatorio d-dimensional $\boldsymbol{X}$, dado por

$$\boldsymbol{X} = (x_1, x_2, \ldots, x_d)^\top \tag{2.1}$$

representa las d mediciones de un elemento. Las variables $x_1, x_2, \ldots, x_d$ se eligen por su capacidad para distinguir entre las K clases. Las variables de (2.1) se denominan variables discriminantes o características, y el vector $\boldsymbol{X}$ es el vector de características.

En esta tesis nos concentraremos únicamente en el caso binario, es decir, cuando se tienen solo dos clases ($K = 2$).

2.2. Clasificación binaria

Consideremos el problema de clasificación binaria en el que deseamos discriminar entre dos clases Π_1 y Π_2, como por ejemplo tumores "malignos" y "benignos" en un estudio de cáncer.

2.2.1. Clasificador de la regla de Bayes

Sean

$$P(\boldsymbol{X} \in \Pi_i) = \pi_i, \quad i = 1, 2, \tag{2.2}$$

las probabilidades a priori de que una observación $\boldsymbol{X} = \boldsymbol{x}$ seleccionada aleatoriamente pertenezca a Π_1 o Π_2. Supongamos también que la densidad de probabilidad multivariada condicional de $\boldsymbol{X}$ para la clase i-ésima es

$$P(\boldsymbol{X} = \boldsymbol{x} | \boldsymbol{X} \in \Pi_i) = f_i(\boldsymbol{x}), \quad i = 1, 2. \tag{2.3}$$

Observamos que no hay ningún requisito de que las f_i sean continuas; podrían ser discretas o ser distribuciones mixtas. Así, de (2.2) y (2.3), por la regla de Bayes (Teorema 1.4.3) se obtiene la probabilidad posterior de que la $\boldsymbol{x}$ observada pertenezca

a Π_i, $i = 1, 2$,

$$\begin{aligned} P(\Pi_i|\boldsymbol{x}) &= P(X \in \Pi_i|\boldsymbol{X} = \boldsymbol{x}) \\ &= \frac{P(\boldsymbol{X} = \boldsymbol{x}|\boldsymbol{X} \in \Pi_i)P(\boldsymbol{X} \in \Pi_i)}{P(\boldsymbol{X} = \boldsymbol{x}|\boldsymbol{X} \in \Pi_1)P(\boldsymbol{X} \in \Pi_1) + P(\boldsymbol{X} = \boldsymbol{x}|\boldsymbol{X} \in \Pi_2)P(\boldsymbol{X} \in \Pi_2)} \quad (2.4) \\ &= \frac{f_i(\boldsymbol{x})\pi_i}{f_1(\boldsymbol{x})\pi_1 + f_2(\boldsymbol{x})\pi_2}. \end{aligned}$$

Para una $\boldsymbol{x}$ dada, una estrategia de clasificación razonable es asignar $\boldsymbol{x}$ a la clase con la probabilidad posterior más alta. Esta estrategia se denomina *clasificador de la regla de Bayes*. En otras palabras, asignamos $\boldsymbol{x}$ a Π_1 si

$$\frac{P(\Pi_1|\boldsymbol{x})}{P(\Pi_2|\boldsymbol{x})} > 1, \quad (2.5)$$

y asignamos $\boldsymbol{x}$ a Π_2 en caso contrario. El cociente $P(\Pi_1|\boldsymbol{x})/P(\Pi_2|\boldsymbol{x})$ se denomina "*odds-ratio*" de que Π_1 en lugar de Π_2 sea la clase correcta dada la información en $\boldsymbol{x}$. Sustituyendo (2.4) en (2.5), el clasificador de la regla de Bayes asigna $\boldsymbol{x}$ a Π_1 si

$$\begin{aligned} \frac{P(\Pi_1|\boldsymbol{x})}{P(\Pi_2|\boldsymbol{x})} &= \frac{\frac{f_1(\boldsymbol{x})\pi_1}{f_1(\boldsymbol{x})\pi_1 + f_2(\boldsymbol{x})\pi_2}}{\frac{f_2(\boldsymbol{x})\pi_2}{f_1(\boldsymbol{x})\pi_1 + f_2(\boldsymbol{x})\pi_2}} > 1 \\ &= \frac{f_1(\boldsymbol{x})\pi_1}{f_2(\boldsymbol{x})\pi_2} > 1, \end{aligned}$$

es decir, si

$$\frac{f_1(\boldsymbol{x})}{f_2(\boldsymbol{x})} > \frac{\pi_2}{\pi_1}, \quad (2.6)$$

y asigna $\boldsymbol{x}$ a Π_2 en otro caso. En la frontera $\{\boldsymbol{x} \in \mathbb{R}^d : f_1(\boldsymbol{x})/f_2(\boldsymbol{x}) = \pi_2/\pi_1\}$, elegimos al azar entre asignar $\boldsymbol{x}$ a Π_1 o Π_2.

2.3. Discriminante lineal de Fisher

2.3.1. Análisis discriminante lineal gaussiano

Para hacer más específico el clasificador de la regla de Bayes, asumamos que ambas probabilidades son probabilidades gaussianas multivariadas con media $\boldsymbol{\mu}$ y matriz de covarianza $\boldsymbol{\Sigma}$, es decir, tomamos f_1 como una densidad $\mathcal{N}_d(\boldsymbol{\mu}_1, \boldsymbol{\Sigma}_1)$ y f_2 como una densidad $\mathcal{N}_d(\boldsymbol{\mu}_2, \boldsymbol{\Sigma}_2)$; además hacemos la suposición de homogeneidad de que $\boldsymbol{\Sigma}_1 = \boldsymbol{\Sigma}_2 = \boldsymbol{\Sigma}$. Entonces la razón de las densidades está dada por

$$\frac{f_1(\boldsymbol{x})}{f_2(\boldsymbol{x})} = \frac{\exp\left\{-\frac{1}{2}(\boldsymbol{x} - \boldsymbol{\mu}_1)^\top \boldsymbol{\Sigma}^{-1}(\boldsymbol{x} - \boldsymbol{\mu}_1)\right\}}{\exp\left\{-\frac{1}{2}(\boldsymbol{x} - \boldsymbol{\mu}_2)^\top \boldsymbol{\Sigma}^{-1}(\boldsymbol{x} - \boldsymbol{\mu}_2)\right\}},$$

donde los factores de normalización $(2\pi)^{-d/2}|\boldsymbol{\Sigma}|^{-1/2}$ tanto en el numerador como en el denominador se cancelan debido a la igualdad de las matrices de covarianza de ambas clases. Tomando logaritmos debido a que tenemos una función positiva, tenemos que

$$\begin{aligned}
\log\left(\frac{f_1(\boldsymbol{x})}{f_2(\boldsymbol{x})}\right) &= \log\left(\frac{\exp\left\{-\frac{1}{2}(\boldsymbol{x}-\boldsymbol{\mu}_1)^\top\boldsymbol{\Sigma}^{-1}(\boldsymbol{x}-\boldsymbol{\mu}_1)\right\}}{\exp\left\{-\frac{1}{2}(\boldsymbol{x}-\boldsymbol{\mu}_2)^\top\boldsymbol{\Sigma}^{-1}(\boldsymbol{x}-\boldsymbol{\mu}_2)\right\}}\right) \\
&= \log\left(\exp\left\{-\frac{1}{2}(\boldsymbol{x}-\boldsymbol{\mu}_1)^\top\boldsymbol{\Sigma}^{-1}(\boldsymbol{x}-\boldsymbol{\mu}_1)\right\}\right) \\
&\quad -\log\left(\exp\left\{-\frac{1}{2}(\boldsymbol{x}-\boldsymbol{\mu}_2)^\top\boldsymbol{\Sigma}^{-1}(\boldsymbol{x}-\boldsymbol{\mu}_2)\right\}\right) \\
&= -\frac{1}{2}(\boldsymbol{x}-\boldsymbol{\mu}_1)^\top\boldsymbol{\Sigma}^{-1}(\boldsymbol{x}-\boldsymbol{\mu}_1)+\frac{1}{2}(\boldsymbol{x}-\boldsymbol{\mu}_2)^\top\boldsymbol{\Sigma}^{-1}(\boldsymbol{x}-\boldsymbol{\mu}_2) \\
&= -\frac{1}{2}\left[(\boldsymbol{x}-\boldsymbol{\mu}_1)^\top\boldsymbol{\Sigma}^{-1}(\boldsymbol{x}-\boldsymbol{\mu}_1)-(\boldsymbol{x}-\boldsymbol{\mu}_2)^\top\boldsymbol{\Sigma}^{-1}(\boldsymbol{x}-\boldsymbol{\mu}_2)\right].
\end{aligned}$$

Notemos que la expresión anterior es igual a

$$\begin{aligned}
& -\frac{1}{2}\left[(\boldsymbol{x}^\top-\boldsymbol{\mu}_1^\top)(\boldsymbol{\Sigma}^{-1}\boldsymbol{x}-\boldsymbol{\Sigma}^{-1}\boldsymbol{\mu}_1)-(\boldsymbol{x}^\top-\boldsymbol{\mu}_2^\top)(\boldsymbol{\Sigma}^{-1}\boldsymbol{x}-\boldsymbol{\Sigma}^{-1}\boldsymbol{\mu}_2)\right] \\
= & -\frac{1}{2}\left[\boldsymbol{x}^\top\boldsymbol{\Sigma}^{-1}\boldsymbol{x}-\boldsymbol{\mu}_1^\top\boldsymbol{\Sigma}^{-1}\boldsymbol{x}-\boldsymbol{x}^\top\boldsymbol{\Sigma}^{-1}\boldsymbol{\mu}_1+\boldsymbol{\mu}_1^\top\boldsymbol{\Sigma}^{-1}\boldsymbol{\mu}_1-\left(\boldsymbol{x}^\top\boldsymbol{\Sigma}^{-1}\boldsymbol{x}\right.\right. \\
& \left.\left.-\boldsymbol{\mu}_2^\top\boldsymbol{\Sigma}^{-1}\boldsymbol{x}-\boldsymbol{x}^\top\boldsymbol{\Sigma}^{-1}\boldsymbol{\mu}_2+\boldsymbol{\mu}_2^\top\ \boldsymbol{\Sigma}^{-1}\boldsymbol{\mu}_2\right)\right] \\
= & -\frac{1}{2}\left[\boldsymbol{x}^\top\boldsymbol{\Sigma}^{-1}\boldsymbol{x}-\boldsymbol{\mu}_1^\top\boldsymbol{\Sigma}^{-1}\boldsymbol{x}-\boldsymbol{x}^\top\boldsymbol{\Sigma}^{-1}\boldsymbol{\mu}_1+\boldsymbol{\mu}_1^\top\boldsymbol{\Sigma}^{-1}\boldsymbol{\mu}_1-\boldsymbol{x}^\top\boldsymbol{\Sigma}^{-1}\boldsymbol{x}\right. \\
& \left.+\boldsymbol{\mu}_2^\top\boldsymbol{\Sigma}^{-1}\boldsymbol{x}+\boldsymbol{x}^\top\boldsymbol{\Sigma}^{-1}\boldsymbol{\mu}_2-\boldsymbol{\mu}_2^\top\ \boldsymbol{\Sigma}^{-1}\boldsymbol{\mu}_2\right] \\
= & -\frac{1}{2}\left[-2\boldsymbol{x}^\top\boldsymbol{\Sigma}^{-1}\boldsymbol{\mu}_1+2\boldsymbol{x}^\top\boldsymbol{\Sigma}^{-1}\boldsymbol{\mu}_2+\boldsymbol{\mu}_1^\top\boldsymbol{\Sigma}^{-1}\boldsymbol{\mu}_1-\boldsymbol{\mu}_2^\top\boldsymbol{\Sigma}^{-1}\boldsymbol{\mu}_2\right] \\
= & -\frac{1}{2}\left[-2\left(\boldsymbol{\mu}_1-\boldsymbol{\mu}_2\right)^\top\boldsymbol{\Sigma}^{-1}\boldsymbol{x}+\left(\boldsymbol{\mu}_1-\boldsymbol{\mu}_2\right)^\top\boldsymbol{\Sigma}^{-1}(\boldsymbol{\mu}_1+\boldsymbol{\mu}_2)\right] \\
= & \left(\boldsymbol{\mu}_1-\boldsymbol{\mu}_2\right)^\top\boldsymbol{\Sigma}^{-1}\left(\boldsymbol{x}-\frac{(\boldsymbol{\mu}_1+\boldsymbol{\mu}_2)}{2}\right),
\end{aligned}$$

la penúltima igualdad se tiene dado que

$$\begin{aligned}(\boldsymbol{\mu}_1 - \boldsymbol{\mu}_2)^\top \boldsymbol{\Sigma}^{-1}(\boldsymbol{\mu}_1 + \boldsymbol{\mu}_2) &= (\boldsymbol{\mu}_1^\top - \boldsymbol{\mu}_2^\top)\left(\boldsymbol{\Sigma}^{-1}\boldsymbol{\mu}_1 + \boldsymbol{\Sigma}^{-1}\boldsymbol{\mu}_2\right) \\ &= \boldsymbol{\mu}_1^\top \boldsymbol{\Sigma}^{-1}\boldsymbol{\mu}_1 - \boldsymbol{\mu}_2^\top \boldsymbol{\Sigma}^{-1}\boldsymbol{\mu}_2.\end{aligned}$$

Finalmente tenemos

$$\log\left(\frac{f_1(\boldsymbol{x})}{f_2(\boldsymbol{x})}\right) = (\boldsymbol{\mu}_1 - \boldsymbol{\mu}_2)^\top \boldsymbol{\Sigma}^{-1}(\boldsymbol{x} - \overline{\boldsymbol{\mu}}), \tag{2.7}$$

donde $\overline{\boldsymbol{\mu}} = \dfrac{(\boldsymbol{\mu}_1 + \boldsymbol{\mu}_2)}{2}$. De esta forma se puede observar que

$$\begin{aligned}L(\boldsymbol{x}) &= \log\left\{\frac{f_1(\boldsymbol{x})\pi_1}{f_2(\boldsymbol{x})\pi_2}\right\} \\ &= \log\left(\frac{f_1(\boldsymbol{x})}{f_2(\boldsymbol{x})}\right) + \log\left(\frac{\pi_1}{\pi_2}\right) \\ &= (\boldsymbol{\mu}_1 - \boldsymbol{\mu}_2)^\top \boldsymbol{\Sigma}^{-1}(\boldsymbol{x} - \overline{\boldsymbol{\mu}}) + \log\left(\frac{\pi_1}{\pi_2}\right) \\ &= (\boldsymbol{\mu}_1 - \boldsymbol{\mu}_2)^\top \boldsymbol{\Sigma}^{-1}\boldsymbol{x} - (\boldsymbol{\mu}_1 - \boldsymbol{\mu}_2)^\top \boldsymbol{\Sigma}^{-1}\overline{\boldsymbol{\mu}} + \log\left(\frac{\pi_1}{\pi_2}\right) \\ &= \boldsymbol{b}^\top \boldsymbol{x} + b_0,\end{aligned}$$

lo que nos dice que $L(\boldsymbol{x})$ es una función lineal de $\boldsymbol{x}$, donde

$$\boldsymbol{b} = \boldsymbol{\Sigma}^{-1}(\boldsymbol{\mu}_1 - \boldsymbol{\mu}_2),$$

y

$$b_0 = -(\boldsymbol{\mu}_1 - \boldsymbol{\mu}_2)^\top \boldsymbol{\Sigma}^{-1}\overline{\boldsymbol{\mu}} + \log\left(\frac{\pi_1}{\pi_2}\right).$$

Entonces, asignamos $\boldsymbol{x}$ a Π_1 si el logaritmo del cociente de las dos probabilidades posteriores es mayor que cero, es decir, si

$$L(\boldsymbol{x}) > 0,$$

y se asigna $\boldsymbol{x}$ a Π_2 en otro caso. Cuando $L(\boldsymbol{x}) = 0$ con $\boldsymbol{x} \in \mathbb{R}^d$, la ecuación resultante es lineal en $\boldsymbol{x}$ y, por tanto, define un hiperplano que divide las dos clases. A esta regla de clasificación se le conoce como *discriminante lineal gaussiano o discriminante lineal de Fisher* (DLF) y

$$U = \boldsymbol{b}^\top \boldsymbol{x},$$

se denomina *función discriminante lineal de Fisher*.

2.3.2. Probabilidad total de error de clasificación

Hemos visto que el DLF divide al espacio $\mathbb{R}^d$ en regiones de clasificación disjuntas R_1 y R_2. Si $\boldsymbol{x}$ cae en la región R_1, se asigna a Π_1, mientras que si $\boldsymbol{x}$ cae en la región R_2, se asigna a Π_2. Notar que al considerar funciones de densidad gaussianas para los datos de las clases, la probabilidad de que un dato de cualquiera de las clases esté sobre el hiperplano es cero. Calculamos la probabilidad de error de clasificación de un nuevo dato $\boldsymbol{x}$ perteneciente a alguna de las dos clases.

El error de clasificación ocurre si $\boldsymbol{x}$ se asigna a Π_2, cuando en realidad $\boldsymbol{x}$ pertenece a Π_1, o si $\boldsymbol{x}$ se asigna a Π_1, pero $\boldsymbol{x}$ pertenece a Π_2. Definamos

$$\Delta^2 = (\boldsymbol{\mu}_1 - \boldsymbol{\mu}_2)^\top \boldsymbol{\Sigma}^{-1} (\boldsymbol{\mu}_1 - \boldsymbol{\mu}_2) \tag{2.8}$$

como la *distancia de Mahalanobis al cuadrado* entre Π_1 y Π_2. Así, si $U = \boldsymbol{b}^\top \boldsymbol{X}$, dado que la distribución normal tiene media $\boldsymbol{\mu}_i$ y matriz de covarianza $\boldsymbol{\Sigma}_i$, tenemos

$$E(U|\boldsymbol{X} \in \Pi_i) = \boldsymbol{b}^\top E(\boldsymbol{X}) = \boldsymbol{b}^\top \boldsymbol{\mu}_i, \quad i = 1, 2,$$

y como $\boldsymbol{b}^\top = (\boldsymbol{\mu}_1 - \boldsymbol{\mu}_2)^\top \boldsymbol{\Sigma}^{-1}$, entonces

$$\begin{aligned} \mathrm{Var}(U|\boldsymbol{X} \in \Pi_i) &= \boldsymbol{b}^\top \mathrm{Cov}_{\Pi_i}(\boldsymbol{X}) \boldsymbol{b} \\ &= \boldsymbol{b}^\top \boldsymbol{\Sigma} \boldsymbol{b} \\ &= (\boldsymbol{\mu}_1 - \boldsymbol{\mu}_2)^\top \boldsymbol{\Sigma}^{-1} \boldsymbol{\Sigma} \boldsymbol{\Sigma}^{-1} (\boldsymbol{\mu}_1 - \boldsymbol{\mu}_2) \\ &= (\boldsymbol{\mu}_1 - \boldsymbol{\mu}_2)^\top \boldsymbol{\Sigma}^{-1} (\boldsymbol{\mu}_1 - \boldsymbol{\mu}_2) \\ &= \Delta^2, \end{aligned}$$

para $i = 1, 2$. Si A es el evento de clasificar incorrectamente a $\boldsymbol{X}$, por la regla de Bayes, la *probabilidad total de error de clasificación* es

$$\begin{aligned} P(A) &= P[(\boldsymbol{X} \in \Pi_1, \boldsymbol{X} \in R_2) \cup (\boldsymbol{X} \in \Pi_2, \boldsymbol{X} \in R_1)] \\ &= P(\boldsymbol{X} \in \Pi_1, \boldsymbol{X} \in R_2) + P(\boldsymbol{X} \in \Pi_2, \boldsymbol{X} \in R_1) \\ &= P[(\boldsymbol{X} \in \Pi_1) \cap (\boldsymbol{X} \in R_2)] + P[(\boldsymbol{X} \in \Pi_2) \cap (\boldsymbol{X} \in R_1)] \\ &= P(\boldsymbol{X} \in R_2 | \boldsymbol{X} \in \Pi_1) P(\boldsymbol{X} \in \Pi_1) + P(\boldsymbol{X} \in R_1 | \boldsymbol{X} \in \Pi_2) P(\boldsymbol{X} \in \Pi_2), \end{aligned}$$

por lo tanto

$$P(A) = P(\boldsymbol{X} \in R_2 | \boldsymbol{X} \in \Pi_1)\pi_1 + P(\boldsymbol{X} \in R_1 | \boldsymbol{X} \in \Pi_2)\pi_2. \tag{2.9}$$

Ya que $L(\boldsymbol{X}) = U + b_0$, entonces se puede estandarizar $U = \boldsymbol{b}^\top \boldsymbol{X}$ de la siguiente manera

$$\begin{aligned} Z &= \frac{U - \boldsymbol{b}^\top \boldsymbol{\mu}_i}{\sqrt{\boldsymbol{b}^\top \boldsymbol{\Sigma} \boldsymbol{b}}} \\ &= \frac{U - \boldsymbol{b}^\top \boldsymbol{\mu}_i}{\Delta} \sim \mathcal{N}(0, 1) \end{aligned}.$$

Por lo tanto,

$$\begin{aligned}
P(\boldsymbol{X} \in R_2|\boldsymbol{X} \in \Pi_1) &= P(L(\boldsymbol{X}) < 0|\boldsymbol{X} \in \Pi_1) \\
&= P(U + b_0 < 0|\boldsymbol{X} \in \Pi_1) \\
&= P(U < -b_0|\boldsymbol{X} \in \Pi_1) \\
&= P\left(\frac{U - \boldsymbol{b}^\top \boldsymbol{\mu}_i}{\sqrt{\boldsymbol{b}^\top \boldsymbol{\Sigma} \boldsymbol{b}}} < \frac{-b_0 - \boldsymbol{b}^\top \boldsymbol{\mu}_i}{\sqrt{\boldsymbol{b}^\top \boldsymbol{\Sigma} \boldsymbol{b}}} \,|\boldsymbol{X} \in \Pi_1\right) \\
&= P\left(Z < \frac{-b_0 - \boldsymbol{b}^\top \boldsymbol{\mu}_1}{\Delta}\right) \\
&= P\Bigg(Z < -\frac{1}{\Delta}\left[-\frac{1}{2}\left(\boldsymbol{\mu}_1^\top \boldsymbol{\Sigma}^{-1}\boldsymbol{\mu}_1 - \boldsymbol{\mu}_2^\top \boldsymbol{\Sigma}^{-1}\boldsymbol{\mu}_2\right) + \log\left(\frac{\pi_1}{\pi_2}\right) + \right. \\
&\qquad \left.(\boldsymbol{\mu}_1 - \boldsymbol{\mu}_2)^\top \boldsymbol{\Sigma}^{-1}\boldsymbol{\mu}_1\right]\Bigg).
\end{aligned}$$

La expresión anterior es igual a

$$\begin{aligned}
&P\left(Z < -\frac{1}{2\Delta}\Big[-\boldsymbol{\mu}_1^\top \boldsymbol{\Sigma}^{-1}\boldsymbol{\mu}_1 + \boldsymbol{\mu}_2^\top \boldsymbol{\Sigma}^{-1}\boldsymbol{\mu}_2 + 2\log\left(\frac{\pi_1}{\pi_2}\right) + \right. \\
&\left. 2\left(\boldsymbol{\mu}_1^\top \boldsymbol{\Sigma}^{-1}\boldsymbol{\mu}_1 - \boldsymbol{\mu}_2^\top \boldsymbol{\Sigma}^{-1}\boldsymbol{\mu}_1\right)\Big]\right) \\
= \; &P\left(Z < -\frac{1}{2\Delta}\left[\boldsymbol{\mu}_1^\top \boldsymbol{\Sigma}^{-1}\boldsymbol{\mu}_1 + \boldsymbol{\mu}_2^\top \boldsymbol{\Sigma}^{-1}\boldsymbol{\mu}_2 - 2\boldsymbol{\mu}_2^\top \boldsymbol{\Sigma}^{-1}\boldsymbol{\mu}_1 \right.\right. \\
&\left.\left. +2\log\left(\frac{\pi_1}{\pi_2}\right)\right]\right) \\
= \; &P\left(Z < -\frac{1}{2\Delta}\left[\Delta^2 + 2\log\left(\frac{\pi_1}{\pi_2}\right)\right]\right) \\
= \; &P\left(Z < -\frac{\Delta}{2} - \frac{\log\left(\frac{\pi_1}{\pi_2}\right)}{\Delta}\right).
\end{aligned}$$

Así, $P(\boldsymbol{X} \in R_2|\boldsymbol{X} \in \Pi_1) = \Phi\left(-\frac{\Delta}{2} - \frac{1}{\Delta}\log\left(\frac{\pi_1}{\pi_2}\right)\right)$.

Por otro lado,

$$\begin{aligned}
P(\boldsymbol{X} \in R_1|\boldsymbol{X} \in \Pi_2) &= P(L(\boldsymbol{X}) > 0|\boldsymbol{X} \in \Pi_2) \\
&= P(U + b_0 > 0|\boldsymbol{X} \in \Pi_2) \\
&= P(U > -b_0|\boldsymbol{X} \in \Pi_2),
\end{aligned}$$

lo cual es igual a

$$\begin{aligned} & P\left(\frac{U-\boldsymbol{b}^\top\boldsymbol{\mu}_i}{\sqrt{\boldsymbol{b}^\top\boldsymbol{\Sigma}\boldsymbol{b}}} > \frac{-b_0-\boldsymbol{b}^\top\boldsymbol{\mu}_i}{\sqrt{\boldsymbol{b}^\top\boldsymbol{\Sigma}\boldsymbol{b}}} \,|\boldsymbol{X}\in\Pi_2\right) \\ = \; & P\left(Z > \frac{-b_0-\boldsymbol{b}^\top\boldsymbol{\mu}_2}{\Delta}\right) \\ = \; & P\left(Z > -\frac{1}{\Delta}\left[-\frac{1}{2}\left(\boldsymbol{\mu}_1^\top\boldsymbol{\Sigma}^{-1}\boldsymbol{\mu}_1-\boldsymbol{\mu}_2^\top\boldsymbol{\Sigma}^{-1}\boldsymbol{\mu}_2\right)+\log\left(\frac{\pi_1}{\pi_2}\right)+\right.\right. \\ & \left.\left.(\boldsymbol{\mu}_1-\boldsymbol{\mu}_2)^\top\boldsymbol{\Sigma}^{-1}\boldsymbol{\mu}_2\right]\right). \end{aligned}$$

La expresión anterior es igual a

$$\begin{aligned} & P\left(Z > -\frac{1}{2\Delta}\left[-\boldsymbol{\mu}_1^\top\boldsymbol{\Sigma}^{-1}\boldsymbol{\mu}_1+\boldsymbol{\mu}_2^\top\boldsymbol{\Sigma}^{-1}\boldsymbol{\mu}_2+2\log\left(\frac{\pi_1}{\pi_2}\right)+\right.\right. \\ & \left.\left.2\left(\boldsymbol{\mu}_1^\top\boldsymbol{\Sigma}^{-1}\boldsymbol{\mu}_2-\boldsymbol{\mu}_2^\top\boldsymbol{\Sigma}^{-1}\boldsymbol{\mu}_2\right)\right]\right) \\ = \; & P\left(Z > -\frac{1}{2\Delta}\left[-\boldsymbol{\mu}_1^\top\boldsymbol{\Sigma}^{-1}\boldsymbol{\mu}_1-\boldsymbol{\mu}_2^\top\boldsymbol{\Sigma}^{-1}\boldsymbol{\mu}_2+2\boldsymbol{\mu}_1^\top\boldsymbol{\Sigma}^{-1}\boldsymbol{\mu}_2+2\log\left(\frac{\pi_1}{\pi_2}\right)\right]\right) \\ = \; & P\left(Z > -\frac{1}{2\Delta}\left[-\Delta^2+2\log\left(\frac{\pi_1}{\pi_2}\right)\right]\right) \\ = \; & P\left(Z > \frac{\Delta}{2}-\frac{\log\left(\frac{\pi_1}{\pi_2}\right)}{\Delta}\right) \\ = \; & \Phi\left(-\frac{\Delta}{2}+\frac{1}{\Delta}\log\left(\frac{\pi_1}{\pi_2}\right)\right). \end{aligned}$$

Por lo tanto,

$$P(A)=\Phi\left(-\frac{\Delta}{2}-\frac{1}{\Delta}\log\left(\frac{\pi_1}{\pi_2}\right)\right)\pi_1+\Phi\left(-\frac{\Delta}{2}+\frac{1}{\Delta}\log\left(\frac{\pi_1}{\pi_2}\right)\right)\pi_2.$$

Notemos que si $\pi_1=\pi_2$, entonces la función resultante es $P(A)=\Phi\left(-\frac{\Delta}{2}\right)$. Además, $P(A)$ como función de Δ es una curva decreciente, tiene el valor $1/2$ cuando $\Delta=0$ (es decir, las dos poblaciones tienen la misma distribución) y tiende a cero a medida que Δ aumenta. En otras palabras, cuanto mayor sea la distancia entre las dos medias poblacionales, menor será la probabilidad de clasificar erróneamente a $\boldsymbol{x}$.

Ejemplo 2.3.1. Supongamos que tenemos los siguientes parámetros $\boldsymbol{\mu}_1 = (c, 0)^\top$, $\boldsymbol{\mu}_2 = (-c, 0)^\top$, $c > 0$ y $\boldsymbol{\Sigma} = \boldsymbol{\Sigma}_1 = \boldsymbol{\Sigma}_2 = \boldsymbol{I}$; además, haciendo la suposición de que las probabilidades a priori coinciden, es decir, $\pi_1 = \pi_2$, entonces es fácil calcular $\boldsymbol{b}^\top$ y b_0 de la siguiente manera:

$$\begin{aligned} \boldsymbol{b}^\top &= (\boldsymbol{\mu}_1 - \boldsymbol{\mu}_2)^\top \boldsymbol{\Sigma}^{-1} \\ &= (c, 0) - (-c, 0)^\top \boldsymbol{I} \\ &= (2c, 0)^\top \boldsymbol{I} \\ &= 2(c, 0)^\top \boldsymbol{I} \\ &= 2\boldsymbol{\mu}_1^\top \boldsymbol{I} \\ &= 2\boldsymbol{\mu}_1^\top. \end{aligned}$$

Por otro lado,

$$b_0 = -(\boldsymbol{\mu}_1 - \boldsymbol{\mu}_2)^\top \boldsymbol{\Sigma}^{-1} \overline{\boldsymbol{\mu}} + \log\left(\frac{\pi_1}{\pi_2}\right),$$

pero $\bar{\boldsymbol{\mu}} = \dfrac{\boldsymbol{\mu}_1 + \boldsymbol{\mu}_2}{2} = \dfrac{\boldsymbol{0}}{2} = \boldsymbol{0}$ y $\log\left(\dfrac{\pi_1}{\pi_2}\right) = \log(1) = 0$, por lo que resulta

$$\begin{aligned} b_0 &= -(\boldsymbol{\mu}_1 - \boldsymbol{\mu}_2)^\top \boldsymbol{\Sigma}^{-1} \overline{\boldsymbol{\mu}} + \log\left(\frac{\pi_1}{\pi_2}\right) \\ &= -(\boldsymbol{\mu}_1 - \boldsymbol{\mu}_2)^\top \boldsymbol{\Sigma}^{-1} \boldsymbol{0} + 0 \\ &= 0. \end{aligned}$$

Así, $L(\boldsymbol{x}) = 2\boldsymbol{\mu}_1^\top \boldsymbol{x}$. Por lo tanto, el discriminante lineal de Fisher es

$$\begin{aligned} \Psi(\boldsymbol{x}) &= \begin{cases} 1, & \text{si } \ L(\boldsymbol{x}) > 0, \\ 2, & \text{si } \ L(\boldsymbol{x}) \leq 0, \end{cases} \\ &= \begin{cases} 1, & \text{si } \ 2\boldsymbol{\mu}_1^\top \boldsymbol{x} > 0, \\ 2, & \text{si } \ 2\boldsymbol{\mu}_1^\top \boldsymbol{x} \leq 0, \end{cases} \end{aligned}$$

donde $2\boldsymbol{\mu}_1^\top \boldsymbol{x} > 0$ si y sólo si $\boldsymbol{\mu}_1^\top \boldsymbol{x} > 0$, si y sólo si $\begin{pmatrix} c & 0 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} > 0$, si y sólo si $cx_1 > 0$, si y sólo si $x_1 > 0$. Así,

$$\Psi(\boldsymbol{x}) = \begin{cases} 1, & \text{si } \ x_1 > 0, \\ 2, & \text{si } \ x_1 \leq 0. \end{cases}$$

En la figura 2.1 se muestran las curvas de nivel de las funciones de densidad de dos clases distribuidas de forma normal con medias $\boldsymbol{\mu}_1 = (1.25, 0)^\top$, $\boldsymbol{\mu}_2 = (-1.25, 0)^\top$ y matriz de covarianza común $\boldsymbol{\Sigma}_1 = \boldsymbol{\Sigma}_2 = \boldsymbol{I}$. La recta del Discriminante Lineal de Fisher es la recta punteada, el eje vertical.

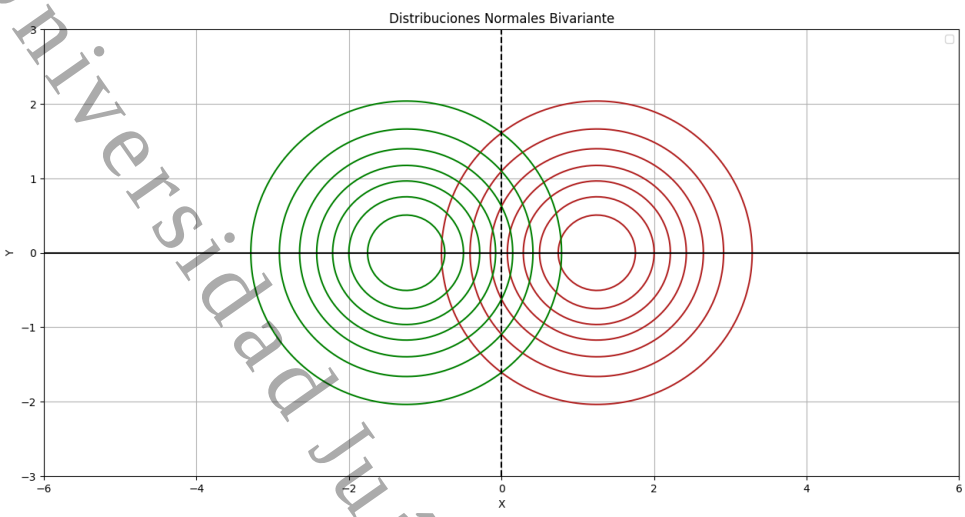

Figura 2.1: Curvas de nivel de dos clases normales bivariadas, con medias $\mu_1 = (1.25, 0)^\top$ y $\mu_2 = (-1.25, 0)^\top$, y matriz de covarianza común $\Sigma = I_2$. La recta del Discriminante Lineal de Fisher es la línea punteada (eje vertical).

2.3.3. Estimaciones por muestreo

Cuando se cuenta con un conjunto de observaciones, la estimación de las probabilidades a priori de las clases puede calcularse fácilmente. La probabilidad de que una observación cualquiera pertenezca a la clase i, es igual al número de observaciones de esa clase entre el número total de observaciones

$$\widehat{\pi}_i = \frac{n_i}{n}, \quad i = 1, 2. \tag{2.10}$$

En la práctica, a pesar de tener una certeza considerable de que los datos se distribuyen de forma normal dentro de cada clase y tienen covarianza común, los valores $\boldsymbol{\mu}_i$ y $\boldsymbol{\Sigma}$ se desconocen, por lo que tienen que ser estimados a partir de las observaciones. Las estimaciones empleadas para el DLF son:

$$\widehat{\boldsymbol{\mu}}_i = \overline{\boldsymbol{X}}_i = \frac{1}{n_i} \sum_{j=1}^{n_i} \boldsymbol{X_{ij}} \quad i = 1, 2, \tag{2.11}$$

y

$$\widehat{\boldsymbol{\Sigma}} = \frac{1}{n-2} \boldsymbol{S}, \tag{2.12}$$

donde

$$\boldsymbol{S} = \boldsymbol{S}_1 + \boldsymbol{S}_2,$$

con

$$\boldsymbol{S}_i = \sum_{j=1}^{n_i} \left(\boldsymbol{X}_{ij} - \overline{\boldsymbol{X}}_i\right) \left(\boldsymbol{X}_{ij} - \overline{\boldsymbol{X}}_i\right)^\top \quad i = 1, 2,$$

donde $n = n_1 + n_2$, las cuales corresponden a estimadores insesgados de las medias poblaciones $\boldsymbol{\mu}_i$, $i = 1, 2$, y la matriz de covarianza común $\boldsymbol{\Sigma}$. Por lo tanto, sustituyendo las estimaciones obtenemos como estimación de $L(\boldsymbol{x})$ a

$$\widehat{L}(\boldsymbol{x}) = \widehat{b}_0 + \widehat{\boldsymbol{b}}^\top \boldsymbol{x}, \tag{2.13}$$

donde

$$\widehat{\boldsymbol{b}} = \widehat{\boldsymbol{\Sigma}}^{-1}(\overline{\boldsymbol{X}}_1 - \overline{\boldsymbol{X}}_2) \tag{2.14}$$

y

$$\begin{aligned} \widehat{b}_0 &= -\frac{1}{2}\left(\widehat{\boldsymbol{\mu}}_1^\top \widehat{\boldsymbol{\Sigma}}^{-1}\widehat{\boldsymbol{\mu}}_1 - \widehat{\boldsymbol{\mu}}_2^\top \widehat{\boldsymbol{\Sigma}}^{-1}\widehat{\boldsymbol{\mu}}_2\right) + \log\left(\frac{\widehat{\pi}_1}{\widehat{\pi}_2}\right) \\ &= -\frac{1}{2}\left(\overline{\boldsymbol{X}}_1^\top \widehat{\boldsymbol{\Sigma}}^{-1}\overline{\boldsymbol{X}}_1 - \overline{\boldsymbol{X}}_2^\top \widehat{\boldsymbol{\Sigma}}^{-1}\overline{\boldsymbol{X}}_2\right) + \log\left(\frac{\frac{n_1}{n}}{\frac{n_2}{n}}\right). \end{aligned}$$

Por lo tanto,

$$\begin{aligned} \widehat{b}_0 &= -\frac{1}{2}\left(\overline{\boldsymbol{X}}_1^\top \widehat{\boldsymbol{\Sigma}}^{-1}\overline{\boldsymbol{X}}_1 - \overline{\boldsymbol{X}}_2^\top \widehat{\boldsymbol{\Sigma}}^{-1}\overline{\boldsymbol{X}}_2\right) + \log\left(\frac{n_1}{n_2}\right) \\ &= -\widehat{\boldsymbol{b}}^\top\left(\frac{\overline{\boldsymbol{X}}_1 + \overline{\boldsymbol{X}}_2}{2}\right) + \log\left(\frac{n_1}{n_2}\right). \end{aligned} \tag{2.15}$$

La regla de clasificación asigna $\boldsymbol{x}$ a Π_1 si $\widehat{L}(\boldsymbol{x}) > 0$, y asigna $\boldsymbol{x}$ a Π_2 en otro caso. Llamamos a esta regla el *Discriminante Lineal de Fisher (DLF) muestral*. La capacidad del DLF muestral para clasificar correctamente las observaciones depende de qué tan buenas sean las estimaciones de π_i y $f_i(\boldsymbol{x})$. Cuanto más cercanas al valor real, más se aproximará el clasificador DLF muestral al clasificador de la regla de Bayes.

Ejemplo 2.3.2. Utilizando el lenguaje de programación Python, creamos un conjunto de datos de dos clases. El código de este ejemplo puede ser consultado en el Apéndice B. En este caso, generamos muestras de tamaño 10 de 2 clases de datos normales bivariados. Los datos de la clase 1 fueron generados aleatoriamente siguiendo una distribución normal bivariada con media $\boldsymbol{\mu}_1 = (1.25, 0)^\top$ y matriz de covarianza igual a la identidad. Los datos de la clase 2 fueron generados aleatoriamente de la normal bivariada con media $\boldsymbol{\mu}_2 = (-1.25, 0)^\top$ y matriz de covarianza igual a la identidad. Los datos de las dos clases se presentan en la tabla 2.1. El vector de coeficientes estimados y el intercepto del Discriminante Lineal de Fisher, fueron: $\hat{\boldsymbol{b}} = [-3.01700433, -0.30399091]$ y $\hat{b}_0 = -0.61069704$.

La figura 2.2, muestra los datos generados, donde los puntos azules son los de la clase 1 y los puntos rojos corresponden a la clase 2. Se traza una línea de separación cuya ecuación es $y = -9.83x - 2.02$, representada como una línea punteada negra, que indica la frontera de decisión aprendida por el Discriminante Lineal de Fisher.

Clase	Característica 1	Característica 2
1	1.746714	-0.138264
	1.897689	1.523030
	1.015847	-0.234137
	2.829213	0.767435
	0.780526	0.542560
	0.786582	-0.465730
	1.491962	-1.913280
	-0.474918	-0.562288
	0.237169	0.314247
	0.341976	-1.412304
2	0.215649	-0.225776
	-1.182472	-1.424748
	-1.794383	0.110923
	-2.400994	0.375698
	-1.850639	-0.291694
	-1.851707	1.852278
	-1.263497	-1.057711
	-0.427455	-1.220844
	-1.041136	-1.959670
	-2.578186	0.196861

Tabla 2.1: Datos generados a partir de dos distribuciones normales bivariadas.

Esta línea separa las dos clases en función de las características de los datos. Los ejes x e y representan las características 1 y 2, respectivamente. La figura proporciona una representación visual de cómo el modelo del DLF divide el espacio de características para clasificar las dos clases generadas. Vemos que el DLF muestral es cercano al DLF poblacional, dado en el Ejemplo 2.3.1.

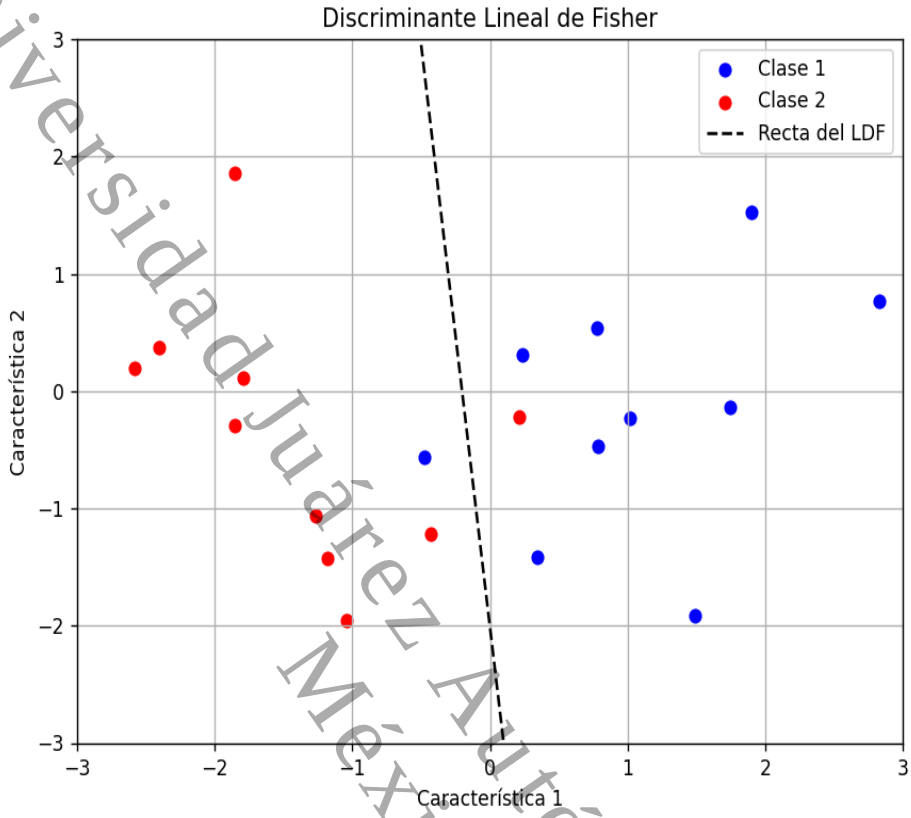

Figura 2.2: Ejemplo de clasificación con el Discriminate Lineal de Fisher.

El Discriminante Lineal de Fisher fue propuesto por Fisher en 1936 (ver [7]) al considerar datos de dos poblaciones independientes con la misma matriz de covarianza, pero no necesariamente normales. La idea fue encontrar la dirección $\boldsymbol{a}$ que separa mejor a $\boldsymbol{a}^\top \overline{X}_1$ de $\boldsymbol{a}^\top \overline{X}_2$ (las proyecciones de las medias de las clases en la dirección de $\boldsymbol{a}$), relativo a la dispersión muestral, es decir, encontrar la dirección $\boldsymbol{a}$ que maximiza

$$J(\boldsymbol{a}) = \frac{(\boldsymbol{a}^\top \overline{X}_1 - \boldsymbol{a}^\top \overline{X}_2)^2}{\boldsymbol{a}^\top S_1 \boldsymbol{a} + \boldsymbol{a}^\top S_2 \boldsymbol{a}} = \frac{[\boldsymbol{a}^\top (\overline{X}_1 - \overline{X}_2)]^2}{\boldsymbol{a}^\top (S_1 + S_2)\boldsymbol{a}}.$$

Esta dirección es precisamente $\boldsymbol{a} = (S_1 + S_2)^{-1}(\overline{X}_1 - \overline{X}_2)$. Por lo que la regla de clasificación es

$$g(\boldsymbol{x}) = \begin{cases} 1, & \text{si } \boldsymbol{a}^\top \boldsymbol{x} + a_0 > 0, \\ 2, & \text{en otro caso,} \end{cases}$$

para alguna constante a_0. Eligiendo a_0 de forma adecuada, esta regla de clasificación coincide con el DLF muestral del caso gaussiano.

2.4. Support Vector Machine

Support Vector Machine (SVM por sus siglas en inglés) ha surgido como una metodología poderosa en el aprendizaje automático, rivalizando e incluso superando en rendimiento a métodos tradicionales como el Discriminante Lineal de Fisher. SVM ha generado un gran interés entre investigadores teóricos y científicos aplicados, siendo utilizado exitosamente en una variedad de problemas de clasificación, desde reconocimiento de dígitos hasta categorización de textos y clasificación de cáncer. Tiene las ventajas de que no es afectado por mínimos locales, eficiencia en el manejo de grandes conjuntos de datos y la capacidad de manejar relaciones no lineales mediante funciones kernel. Aunque inicialmente fue diseñado para clasificación binaria, se ha extendido para abordar problemas de clasificación multiclase y regresión. Sin embargo, se reconoce que puede haber aplicaciones donde métodos de clasificación hechos a medida superen el rendimiento de SVM; ver [13].

2.4.1. Support Vector Machine lineal

Consideremos al conjunto de datos $\mathcal{L}$ de la Definición 2.0.1. Supongamos que las etiquetas de las clases son $y_i \in \{-1, +1\}$, para $i = 1, 2, \ldots, n$. Una regla de clasificación f asigna cada nuevo punto $\boldsymbol{x}$ de un conjunto de prueba T en una de las dos clases Π_+ o Π_-, dependiendo de si $C(\boldsymbol{x}) = \text{sign}(f(\boldsymbol{x}))$ es $+1$ (si $f(\boldsymbol{x}) \geq 0$) o -1 (si $f(\boldsymbol{x}) < 0$), respectivamente. De este modo f asigna todos los puntos positivos en T (es decir, aquellos con $y = +1$) a Π_+ y todos los puntos negativos en T (con $y = -1$) a Π_-.

2.4.2. El caso linealmente separable

Primero, consideremos la situación más simple: supongamos que los datos con etiquetas $y_i = +1$ y $y_i = -1$ del conjunto de aprendizaje $\mathcal{L}$ pueden ser separados por un hiperplano

$$H : \beta_0 + \boldsymbol{x}^\top \boldsymbol{\beta} = \mathbf{0}, \tag{2.16}$$

donde $\boldsymbol{\beta}$ es el vector de pesos con norma euclidiana $\|\boldsymbol{\beta}\|$ y β_0 es el sesgo o intercepto. Si este hiperplano puede separar sin error el conjunto de aprendizaje en las dos clases dadas, se le llama un *hiperplano separante*. Existe un número infinito de tales hiperplanos separantes.

Para un hiperplano separante, sea d_- la distancia del hiperplano al punto más cercano de los datos negativos, y sea d_+ la distancia del hiperplano al punto más cercano de los datos positivos. Se define el *margen* del hiperplano, como la suma de estas dos distancias, $d = d_- + d_+$. Si el margen del hiperplano es maximizado, entonces el hiperplano se considera un *hiperplano separante óptimo*, también conocido como *clasificador de margen máximo*.

Si los datos de aprendizaje de las dos clases son linealmente separables, existe un conjunto de parámetros β_0 y $\boldsymbol{\beta}$ que satisfacen las siguientes condiciones:

$$\beta_0 + \boldsymbol{x}_i^\top \boldsymbol{\beta} \geq +1, \quad \text{si} \quad y_i = +1, \tag{2.17}$$

$$\beta_0 + \boldsymbol{x}_i^\top \boldsymbol{\beta} \leq -1, \quad \text{si} \quad y_i = -1. \tag{2.18}$$

Si hay vectores de datos en $\mathcal{L}$ para los cuales se cumple la igualdad en (2.17), entonces esos vectores de datos están en el hiperplano $H_{+1} : (\beta_0 - 1) + \boldsymbol{x}^\top \boldsymbol{\beta} = 0$. Similarmente, si hay vectores en $\mathcal{L}$ que cumplen la igualdad en (2.18), entonces los datos están en el hiperplano $H_{-1} : (\beta_0 + 1) + \boldsymbol{x}^\top \boldsymbol{\beta} = 0$. Estos puntos de datos que yacen en los hiperplanos H_{+1} y H_{-1} se llaman *vectores soporte*.

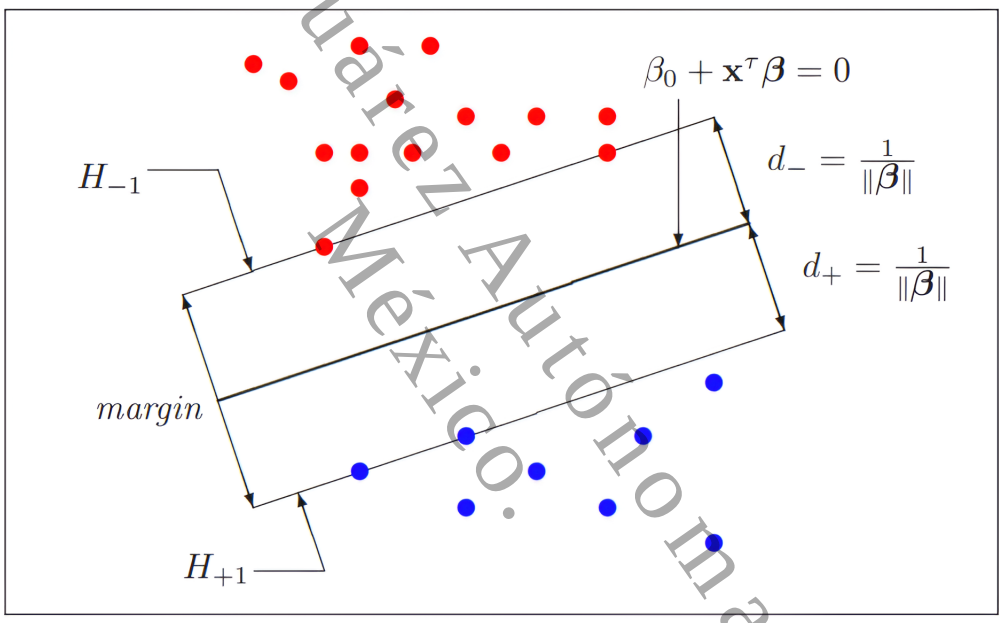

Figura 2.3: Clasificación por Support Vector Machine, caso linealmente separable. La figura es tomada de [13].

A menudo, estos vectores soporte constituyen solo un pequeño porcentaje del total de puntos de la muestra. Si $\boldsymbol{x}_{-1}$ se encuentra en el hiperplano H_{-1}, y si $\boldsymbol{x}_{+1}$ se encuentra en el hiperplano H_{+1}, entonces,

$$\beta_0 + \boldsymbol{x}_{-1}^\top \boldsymbol{\beta} = -1, \quad \beta_0 + \boldsymbol{x}_{+1}^\top \boldsymbol{\beta} = +1.$$

La diferencia de las dos ecuaciones es $\boldsymbol{x}_{+1}^\top \boldsymbol{\beta} - \boldsymbol{x}_{-1}^\top \boldsymbol{\beta} = 2$, y al sumarlas se obtiene $\beta_0 = -\frac{1}{2}\left\{\boldsymbol{x}_{+1}^\top \boldsymbol{\beta} + \boldsymbol{x}_{-1}^\top \boldsymbol{\beta}\right\}$. Las distancias perpendiculares del hiperplano $\beta_0 + \boldsymbol{x}^\top \boldsymbol{\beta} =$

0 a los puntos $\boldsymbol{x}_{+1}$ y $\boldsymbol{x}_{-1}$ son

$$d_- = \frac{|\beta_0 + \boldsymbol{x}_{-1}^\top \boldsymbol{\beta}|}{\|\boldsymbol{\beta}\|} = \frac{1}{\|\boldsymbol{\beta}\|}, \quad d_+ = \frac{|\beta_0 + \boldsymbol{x}_{+1}^\top \boldsymbol{\beta}|}{\|\boldsymbol{\beta}\|} = \frac{1}{\|\boldsymbol{\beta}\|}, \tag{2.19}$$

por lo que el margen de separación es $2/\|\boldsymbol{\beta}\|$. En la figura 2.3 los puntos rojos corresponden a puntos de datos con $y_i = -1$ y los puntos azules corresponden a puntos de datos con $y_i = +1$. El hiperplano separante es la recta $\beta_0 + \boldsymbol{\beta}^\top \boldsymbol{x} = 0$ y los vectores soporte son los puntos situados en los hiperplanos H_{-1} y H_{+1}.

De (2.17) y (2.18) podemos obtener una única desigualdad,

$$y_i(\beta_0 + \boldsymbol{x}_i^\top \boldsymbol{\beta}) \geq +1, \quad i = 1, 2, \ldots, n, \tag{2.20}$$

donde $y_i(\beta_0 + \boldsymbol{x}_i^\top \boldsymbol{\beta})$ es llamado el *margen de* $(\boldsymbol{x_i}, y_i)$ *con respecto al hiperplano* (2.16), $i = 1, 2, \ldots, n$. El problema radica en encontrar el hiperplano separante óptimo, es decir, el hiperplano que maximiza el margen $2/\|\boldsymbol{\beta}\|$ sujeto a las condiciones dadas por (2.20). Equivalentemente deseamos encontrar β_0 y $\boldsymbol{\beta}$ que resuelvan el problema

$$\text{minimizar } \frac{1}{2}\|\boldsymbol{\beta}\|^2, \tag{2.21}$$

$$\text{sujeto a } \; y_i(\beta_0 + \boldsymbol{x}_i^\top \boldsymbol{\beta}) \geq 1, \quad i = 1, 2, \ldots, n. \tag{2.22}$$

Este es un problema de optimización convexa: minimizar una función cuadrática sujeta a restricciones de desigualdad lineales. La convexidad garantiza que hay un mínimo global sin mínimos locales. El hiperplano de separación óptimo resultante se denomina *solución de margen maximal* o *duro*. Para resolver este problema, se utiliza el enfoque de los multiplicadores de Lagrange. Multiplicamos las restricciones por multiplicadores de Lagrange positivos y restamos cada producto a la función objetivo (2.21) para formar el *funcional primal*

$$F_P(\beta_0, \boldsymbol{\beta}, \boldsymbol{\alpha}) = \frac{1}{2}\|\boldsymbol{\beta}\|^2 - \sum_{i=1}^{n} \alpha_i \{y_i(\beta_0 + \boldsymbol{x}_i^\top \boldsymbol{\beta}) - 1\}, \tag{2.23}$$

donde

$$\boldsymbol{\alpha} = (\alpha_1, \ldots, \alpha_n)^\top \geq \mathbf{0}$$

es el n-vector de coeficientes lagrangianos (no negativos). Queremos minimizar F_P con respecto a las variables primales β_0 y $\boldsymbol{\beta}$ y después maximizar la F_P mínima obtenida con respecto a las *variables duales* $\boldsymbol{\alpha}$.

Las condiciones de Karush–Kuhn–Tucker (KKT) son un conjunto de condiciones necesarias y suficientes para encontrar una solución en un problema de optimización con restricciones. Para nuestro problema primal, $\beta_0, \boldsymbol{\beta}$ y $\boldsymbol{\alpha}$ deben satisfacer:

$$\frac{\partial F_P(\beta_0, \boldsymbol{\beta}, \boldsymbol{\alpha})}{\partial \beta_0} = -\sum_{i=1}^{n} \alpha_i y_i = 0 \tag{2.24}$$

$$\frac{\partial F_P(\beta_0, \boldsymbol{\beta}, \boldsymbol{\alpha})}{\partial \boldsymbol{\beta}} = \boldsymbol{\beta} - \sum_{i=1}^{n} \alpha_i y_i \boldsymbol{x}_i = 0 \tag{2.25}$$

$$y_i(\beta_0 + \boldsymbol{x}_i^\top \boldsymbol{\beta}) - 1 \geq 0 \tag{2.26}$$

$$\alpha_i \geq 0 \tag{2.27}$$

$$\alpha_i\{y_i(\beta_0 + \boldsymbol{x}_i^\top \boldsymbol{\beta}) - 1\} = 0, \tag{2.28}$$

para $i = 1, 2, \ldots, n$. Sean β_0^* y $\boldsymbol{\beta}^*$ el valor y el vector que minimizan F_P. De (2.24) y (2.25) se tiene que

$$\sum_{i=1}^{n} \alpha_i y_i = 0 \tag{2.29}$$

y

$$\boldsymbol{\beta^*} = \sum_{i=1}^{n} \alpha_i y_i \boldsymbol{x}_i. \tag{2.30}$$

Sustituyendo (2.30) en (2.23), se tiene

$$\begin{aligned} F_D(\boldsymbol{\alpha}) &= \frac{1}{2}\|\boldsymbol{\beta^*}\|^2 - \sum_{i=1}^{n} \alpha_i\{y_i(\beta_0^* + \boldsymbol{x}_i^\top \boldsymbol{\beta^*}) - 1\} \\ &= \frac{1}{2}\|\boldsymbol{\beta^*}\|^2 - \sum_{i=1}^{n} \alpha_i y_i \beta_0^* - \sum_{i=1}^{n} \alpha_i y_i \boldsymbol{x}_i^\top \boldsymbol{\beta^*} + \sum_{i=1}^{n} \alpha_i \\ &= \frac{1}{2}(\boldsymbol{\beta^*})^\top \boldsymbol{\beta^*} - \beta_0^* \sum_{i=1}^{n} \alpha_i y_i - \sum_{i=1}^{n} \alpha_i y_i \boldsymbol{x}_i^\top \boldsymbol{\beta^*} + \sum_{i=1}^{n} \alpha_i, \end{aligned}$$

pero

$$\begin{aligned} \sum_{i=1}^{n} \alpha_i y_i \boldsymbol{x}_i^\top \boldsymbol{\beta^*} &= (\boldsymbol{\beta^*})^\top \boldsymbol{\beta^*} \\ &= \left(\sum_{i=1}^{n} \alpha_i y_i \boldsymbol{x}_i^\top\right)\left(\sum_{j=1}^{n} \alpha_j y_j \boldsymbol{x}_j\right), \end{aligned}$$

luego, usando (2.29)

$$\begin{aligned} F_D(\boldsymbol{\alpha}) &= \frac{1}{2}\left(\sum_{i=1}^{n}\alpha_i y_i \boldsymbol{x}_i^\top\right)\left(\sum_{j=1}^{n}\alpha_j y_j \boldsymbol{x}_j\right) - \beta_0^*\sum_{i=1}^{n}\alpha_i y_i \\ &\quad - \left(\sum_{i=1}^{n}\alpha_i y_i \boldsymbol{x}_i^\top\right)\left(\sum_{j=1}^{n}\alpha_j y_j \boldsymbol{x}_j\right) + \sum_{i=1}^{n}\alpha_i \\ &= \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n}\alpha_i\alpha_j y_i y_j(\boldsymbol{x}_i^\top \boldsymbol{x}_j) - \sum_{i=1}^{n}\sum_{j=1}^{n}\alpha_i\alpha_j y_i y_j(\boldsymbol{x}_i^\top \boldsymbol{x}_j) + \sum_{i=1}^{n}\alpha_i, \end{aligned}$$

por lo tanto,

$$F_D(\boldsymbol{\alpha}) = \sum_{i=1}^{n}\alpha_i - \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n}\alpha_i\alpha_j y_i y_j(\boldsymbol{x}_i^\top \boldsymbol{x}_j). \tag{2.31}$$

La función obtenida se denomina el *funcional dual* del problema de optimización. Posteriormente se buscan los multiplicadores de Lagrange $\boldsymbol{\alpha}$ que maximizan el funcional dual (2.31) sujeto a las restricciones (2.27) y (2.29). El problema de maximización restringido puede escribirse en notación matricial como sigue. Hallar $\boldsymbol{\alpha}$ para

$$\text{maximizar} \quad F_D(\boldsymbol{\alpha}) = \mathbf{1}_n^\top \boldsymbol{\alpha} - \frac{1}{2}\boldsymbol{\alpha}^\top \boldsymbol{H}\boldsymbol{\alpha}, \tag{2.32}$$

$$\text{sujeto a} \quad \boldsymbol{\alpha} \geq \mathbf{0}, \ \boldsymbol{\alpha}^\top \boldsymbol{y} = 0, \tag{2.33}$$

donde $\boldsymbol{y} = (y_1, \ldots, y_n)^\top$ y $\boldsymbol{H} = (H_{ij})$ es una matriz cuadrada $n \times n$ con $H_{ij} = y_i y_j(\boldsymbol{x}_i^\top \boldsymbol{x}_j)$. Si $\widehat{\boldsymbol{\alpha}}$ resuelve este problema de optimización, el vector de ponderación óptimo $\widehat{\boldsymbol{\beta}}$ se puede calcular como

$$\widehat{\boldsymbol{\beta}} = \sum_{i=1}^{n}\widehat{\alpha}_i y_i \boldsymbol{x}_i. \tag{2.34}$$

Si $\widehat{\alpha}_i > 0$, entonces de (2.28) se tiene que $y_i(\beta_0^* + \boldsymbol{x}_i^\top \boldsymbol{\beta}^*) = 1$, y entonces $\boldsymbol{x}_i$ es un vector soporte. Para todas las observaciones que no son vectores soporte $\widehat{\alpha}_i = 0$.

Sea $sv \subset \{1, 2, \ldots, n\}$ el subconjunto de índices que identifican los vectores soporte (y también los multiplicadores de Lagrange no nulos). Entonces el vector $\boldsymbol{\beta}$ óptimo se puede expresar como una suma solo sobre los vectores soporte, es decir,

$$\widehat{\boldsymbol{\beta}} = \sum_{i \in sv}\widehat{\alpha}_i y_i \boldsymbol{x}_i. \tag{2.35}$$

Esto es, $\widehat{\boldsymbol{\beta}}$ es una función lineal sólo de los vectores soporte $\{\boldsymbol{x}_i : i \in sv\}$. En la mayoría de las aplicaciones, el número de vectores soporte será pequeño en relación

con el tamaño del conjunto de datos $\mathcal{L}$, lo que produce una solución *sparce*. En este caso, los vectores soporte contienen toda la información necesaria para determinar el hiperplano óptimo. Los problemas de optimización primal y dual producen la misma solución, aunque el problema dual es más sencillo de calcular y es más fácil de generalizar a clasificadores no lineales.

El sesgo óptimo $\widehat{\beta}_0$ puede calcularse resolviendo (2.28) para cada vector soporte y luego promediar los resultados. En otras palabras, el sesgo del hiperplano óptimo viene dado por

$$\widehat{\beta}_0 = \frac{1}{|sv|} \sum_{i \in sv} \left(\frac{1 - y_i \boldsymbol{x}_i^\top \widehat{\boldsymbol{\beta}}}{y_i} \right), \tag{2.36}$$

donde $|sv|$ es el número de vectores soporte en $\mathcal{L}$.

Entonces la función lineal que determina el hiperplano puede ser escrita como

$$\begin{aligned} \widehat{f}(\boldsymbol{x}) &= \widehat{\beta}_0 + \boldsymbol{x}^\top \widehat{\boldsymbol{\beta}} \\ &= \widehat{\beta}_0 + \sum_{i \in sv} \widehat{\alpha}_i y_i (\boldsymbol{x}^\top \boldsymbol{x}_i). \end{aligned} \tag{2.37}$$

Solo los vectores soporte son relevantes para calcular el hiperplano de separación óptimo. La regla de clasificación correspondiente al hiperplano de separación óptimo, que denominaremos *Support Vector Machine (SVM) de margen duro*, viene dada por

$$C(\boldsymbol{x}) = \text{sign}\{\widehat{f}(\boldsymbol{x})\}. \tag{2.38}$$

Si $j \in sv$, entonces de (2.37) se obtiene

$$y_j \widehat{f}(\boldsymbol{x}_j) = y_j(\widehat{\beta}_0 + \boldsymbol{x}_j^\top \widehat{\boldsymbol{\beta}}) = y_j \widehat{\beta}_0 + \sum_{i \in sv} \widehat{\alpha}_i y_i y_j (\boldsymbol{x}_j^\top \boldsymbol{x}_i) = 1. \tag{2.39}$$

De (2.29) y (2.39) se tiene que el cuadrado de la norma del vector de pesos $\widehat{\boldsymbol{\beta}}$ del hiperplano óptimo es:

$$\begin{aligned} \|\widehat{\boldsymbol{\beta}}\|^2 &= \sum_{i \in sv} \sum_{j \in sv} \widehat{\alpha}_i \widehat{\alpha}_j y_i y_j (\boldsymbol{x}_i^\top \boldsymbol{x}_j) \\ &= \sum_{j \in sv} \widehat{\alpha}_j y_j \sum_{i \in sv} \widehat{\alpha}_i y_i (\boldsymbol{x}_i^\top \boldsymbol{x}_j) \\ &= \sum_{j \in sv} \widehat{\alpha}_j (1 - y_j \widehat{\beta}_0) \\ &= \sum_{j \in sv} \widehat{\alpha}_j. \end{aligned} \tag{2.40}$$

También se tiene que el hiperplano óptimo tiene como margen máximo $2/||\widehat{\boldsymbol{\beta}}||$, donde

$$\frac{1}{\|\widehat{\boldsymbol{\beta}}\|} = \left(\sum_{j \in sv} \widehat{\alpha}_j \right)^{-1/2}. \tag{2.41}$$

2.4.3. El caso linealmente no separable

El problema planteado en la sección anterior es poco común en la práctica, porque los problemas reales se caracterizan normalmente porque los datos extraídos de dos clases no son linealmente separables. El caso *linealmente no separable* ocurre si las dos clases son separables, pero no de manera lineal, o si no existe una separación clara entre las dos clases, ya sea de manera lineal o no lineal. Una razón para la superposición entre clases es el alto nivel de ruido (es decir, grandes variaciones) en una o ambas clases. Como resultado, se violarán una o más de las restricciones para los datos vistos en la sección anterior.

La idea para abordar este nuevo problema, es introducir en la condición (2.20), que define al hiperplano de separación, un conjunto de variables reales no negativas, denominadas *variables de holgura*, ξ_i, para cada observación $(\boldsymbol{x}_i, y_i)$ $i = 1, \ldots, n$, en $\mathcal{L}$.

Sea

$$\boldsymbol{\xi} = (\xi_1, \ldots, \xi_n)^\top \geq \mathbf{0}. \tag{2.42}$$

La condición (2.20) se transforma en $y_i(\beta_0 + \boldsymbol{x}^\top\boldsymbol{\beta}) + \xi_i \geq 1$ para $i = 1, \ldots, n$. El hiperplano

$$H: \ \beta_0 + \boldsymbol{x}^\top\boldsymbol{\beta} = 0$$

tiene la función de dividir lo mejor posible a las clases. De este modo, las variables de holgura de valor cero, corresponden a datos que satisfacen $y_i(\beta_0 + \boldsymbol{x}^\top\boldsymbol{\beta}) \geq 1$; y las mayores a cero, corresponden a datos que no satisfacen esta última desigualdad. Las variables de holgura mayores que cero y menores o iguales a 1 corresponden a datos no separables pero bien clasificados; y las mayores que 1 a datos no separables y mal clasificados, ver [4]. Nuevamente, como en el caso linealmente separable, consideramos a los hiperplanos $H_{+1} : (\beta_0 - 1) + \boldsymbol{x}^\top\boldsymbol{\beta} = 0$ y $H_{-1} : (\beta_0 + 1) + \boldsymbol{x}^\top\boldsymbol{\beta} = 0$. Los vectores de datos que yacen en los hiperplanos H_{+1} y H_{-1} son los *vectores soporte*.

En la figura 2.4 se muestra el caso linealmente no separable: las variables de holgura ξ_1, ξ_2 y ξ_5, cuyos valores son mayores que 1, corresponden a datos mal clasificados; las variables de holgura ξ_3 y ξ_4, cuyos valores son mayores que cero pero menores o iguales que 1, corresponden a datos bien clasificados. Las otras variables simplemente corresponden al caso separable.

El problema de optimización de *margen suave* consiste en encontrar $\beta_0, \boldsymbol{\beta}$ y $\boldsymbol{\xi}$ para

$$\text{minimizar} \ \ \frac{1}{2}\|\boldsymbol{\beta}\|^2 + C\sum_{i=1}^{n}\xi_i, \tag{2.43}$$

$$\text{sujeto a} \ \ \xi_i \geq, 0 \ \ y_i(\beta_0 + \boldsymbol{x}_i^\top\boldsymbol{\beta}) \geq 1 - \xi_i, \ \ i = 1, \ldots, n, \tag{2.44}$$

donde $C > 0$ se denomina *parámetro de regularización* cuyo valor es elegido por el usuario, y permite controlar el grado de sobreajuste del clasificador final y la

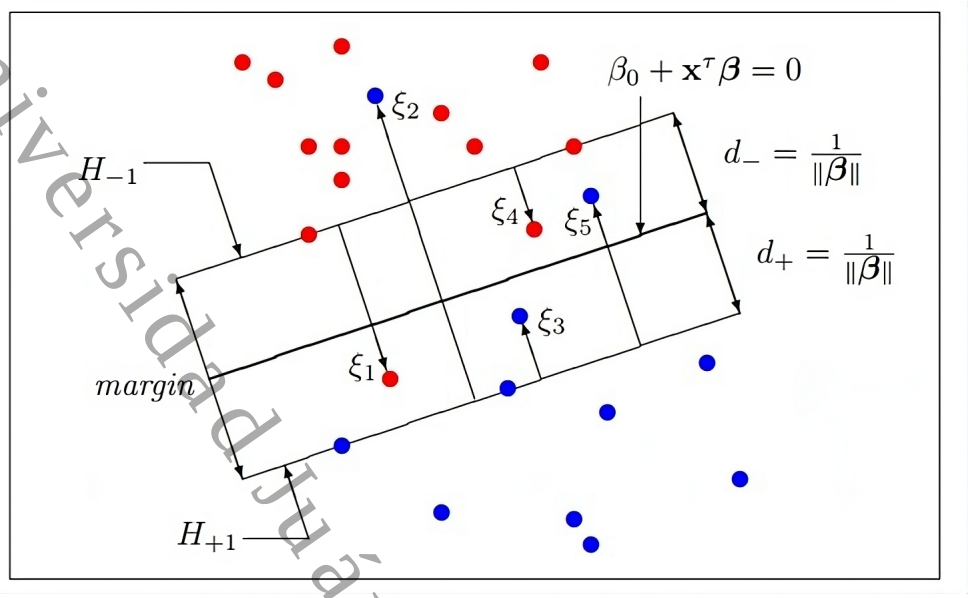

Figura 2.4: Clasificación por Support Vector Machine, caso linealmente no separable. La figura fue tomada de [13].

proporción de datos mal clasificados. Este valor permite controlar el tamaño de las variables holgura. Si el valor de C es grande, entonces permitirá que las ξ_i tomen valores pequeños, lo que garantiza un margen óptimo estrecho, es decir, un modelo que podría estar muy sobreajustado a los datos de entrenamiento. Por otro lado, si C toma valores pequeños, entonces las ξ_i podrían tomar valores grandes, lo que resultaría en un margen óptimo más ancho y un modelo que admite más datos entre los hiperplanos H_{-1} y H_{+1}, e incluso mal clasificados.

El funcional primal $F_P = F_P(\beta_0, \boldsymbol{\beta}, \boldsymbol{\xi}, \boldsymbol{\alpha}, \boldsymbol{\eta})$ es

$$F_P = \frac{1}{2}\|\boldsymbol{\beta}\|^2 + C\sum_{i=1}^{n}\xi_i - \sum_{i=1}^{n}\alpha_i(y_i(\beta_0 + \boldsymbol{x}_i^\top\boldsymbol{\beta}) - (1-\xi_i)) - \sum_{i=1}^{n}\xi_i\eta_i, \tag{2.45}$$

con $\boldsymbol{\alpha} = (\alpha_1, \ldots, \alpha_n)^\top \geq \mathbf{0}$ y $\boldsymbol{\eta} = (\eta_1, \ldots, \eta_n)^\top \geq \mathbf{0}$. Análogamente al caso separable, para obtener el funcional dual, derivamos F_P con respecto a $\beta_0, \boldsymbol{\beta}$ y $\boldsymbol{\xi}$ e igualamos a cero:

$$\frac{\partial F_P}{\partial \beta_0} = -\sum_{i=1}^{n}\alpha_i y_i = 0, \tag{2.46}$$

$$\frac{\partial F_P}{\partial \boldsymbol{\beta}} = \boldsymbol{\beta} - \sum_{i=1}^{n}\alpha_i y_i \boldsymbol{x}_i = 0, \tag{2.47}$$

$$\frac{\partial F_P}{\partial \xi_i} = C - \alpha_i - \eta_i = 0, \tag{2.48}$$

$i = 1, \ldots, n$. De aquí se tiene que los parámetros β_0^*, $\boldsymbol{\beta}^*$ y $\boldsymbol{\xi}^*$ que minimizan F_P satisfacen

$$\sum_{i=1}^{n} \alpha_i y_i = 0, \quad \boldsymbol{\beta}^* = \sum_{i=1}^{n} \alpha_i y_i \boldsymbol{x}_i, \quad \alpha_i = C - \eta_i.$$

Sustituyendo estas igualdades en (2.45) resulta el funcional dual

$$F_D(\boldsymbol{\alpha}) = \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j y_i y_j (\boldsymbol{x}_i^\top \boldsymbol{x}_j). \tag{2.49}$$

De las condiciones $C - \alpha_i - \eta_i = 0$ y $\eta_i \geq 0$ se tiene que $0 \leq \alpha_i \leq C$. Las condiciones de Karush–Kuhn–Tucker son (2.46), (2.47) y (2.48) y

$$y_i(\beta_0 + \boldsymbol{x}_i^\top \boldsymbol{\beta}) - (1 - \xi_i) \geq 0, \tag{2.50}$$
$$\xi_i \geq 0, \tag{2.51}$$
$$\alpha_i \geq 0, \tag{2.52}$$
$$\eta_i \geq 0, \tag{2.53}$$
$$\alpha_i(y_i(\beta_0 + \boldsymbol{x}_i^\top \boldsymbol{\beta}) - (1 - \xi_i)) = 0, \tag{2.54}$$
$$\xi_i(\alpha_i - C) = 0, \tag{2.55}$$

con $i = 1, \ldots, n$. De (2.55), una variable de holgura, ξ_i, sólo puede ser distinta de cero si $\alpha_i = C$. Notar también que si $0 < \alpha_i < C$, por (2.54) y (2.55), se tiene que $\boldsymbol{x}_i$ satisface la ecuación $y_i(\beta_0 + \boldsymbol{x}_i^\top \boldsymbol{\beta}) = 1$, es decir, es un vector soporte. Por otro lado, si $\alpha_i = 0$, entonces por (2.55) tenemos que $\xi_i = 0$.
Se puede escribir el problema de maximización dual en notación matricial como se presenta a continuación. Encontrar $\boldsymbol{\alpha}$ para

$$\text{maximizar} \quad F_D(\boldsymbol{\alpha}) = \mathbf{1}_n^\top \boldsymbol{\alpha} - \frac{1}{2} \boldsymbol{\alpha}^\top \boldsymbol{H} \boldsymbol{\alpha} \tag{2.56}$$

$$\text{sujeto a} \quad \boldsymbol{\alpha}^\top \boldsymbol{y} = \mathbf{0}, \quad \mathbf{0} \leq \boldsymbol{\alpha} \leq C\mathbf{1}_n. \tag{2.57}$$

La única diferencia entre este problema de optimización y el del caso linealmente separable, es que aquí los coeficientes lagrangianos α_i, $i = 1, \ldots, n$, están acotados superiormente por C. Por lo tanto, si $\widehat{\boldsymbol{\alpha}}$ resuelve este problema de optimización, entonces

$$\widehat{\boldsymbol{\beta}} = \sum_{i=1}^{n} \widehat{\alpha}_i y_i \boldsymbol{x}_i \tag{2.58}$$

es el vector de ponderación óptimo. De la condición (2.54) se tiene que un dato $\boldsymbol{x}_i$ con $0 < \alpha_i \leq C$ satisface $y_i(\beta_0 + \boldsymbol{x}_i^\top \boldsymbol{\beta}) - (1 - \xi_i) = 0$, por lo que si $\xi_i = 0$, entonces $\boldsymbol{x}_i$ es un dato sobre el hiperplano H_{-1} o H_{+1}, y por lo tanto es un vector soporte;

y si $\xi_i > 0$, como se vio anteriormente, se tiene que $\alpha_i = C$. En otras palabras, el vector de pesos óptimo $\widehat{\boldsymbol{\beta}}$ está determinado únicamente por los vectores soporte y los datos que tienen variable de holgura positiva, ya que estos son los datos para los cuales $0 < \widehat{\alpha}_i \leq C$.
Las condiciones complementarias de Karush-Kuhn-Tucker, (2.54) y (2.55), pueden utilizarse para encontrar el valor óptimo del sesgo, $\widehat{\beta}_0$, análogamente a como se hizo en el caso linealmente separable, pero considerando solo a los datos con $0 < \alpha_i < C$. Es decir, el sesgo óptimo del hiperplano puede calcularse como

$$\widehat{\beta}_0 = \frac{1}{|sv_0|} \sum_{i \in sv_0} \left(\frac{1 - y_i \boldsymbol{x}_i^\top \widehat{\boldsymbol{\beta}}}{y_i} \right), \tag{2.59}$$

donde $sv_0 = \{i : 0 < \alpha_i < C\}$, y $|sv_0|$ es la cardinalidad de sv_0.

Si $\widehat{f}(\boldsymbol{x}) = \widehat{\beta}_0 + \boldsymbol{x}^\top \widehat{\boldsymbol{\beta}}$, definimos la regla de clasificación

$$C(\boldsymbol{x}) = \text{sign}\{\widehat{f}(\boldsymbol{x})\},$$

a la cual llamamos clasificador *Suppor Vector Machine (SVM) de margen suave*.

Los códigos en Python de los siguientes ejemplos pueden consultarse en el Apéndice B.

Ejemplo 2.4.1. La figura 2.5 ilustra cómo el método SVM clasifica un conjunto de datos bidimensionales linealmente separables generados aleatoriamente. Los datos de la clase 1, representados por cuadros, fueron generados de una distribución normal con media $(5.4, 0)^\top$ y matriz de covarianza $\begin{pmatrix} 7 & 2 \\ 2 & 1 \end{pmatrix}$; los datos de la clase 2 fueron generados de una distribución normal con media $(-7, 2)^\top$ y la misma matriz de covarianza de la clase 1. Ambas clases tienen 10 elementos, los cuales se presentan en la tabla 2.2. El vector de pesos óptimo es $\widehat{\boldsymbol{\beta}} = [-0.24844317, 0.21395735]$ y el sesgo óptimo es $\widehat{\beta}_0 = -0.5574$. En la figura 2.5 se puede observar una línea sólida negra que representa la recta de separación cuya ecuación es $y = 1.16x + 2.61$. Esta línea maximiza el margen entre las dos clases. En ambos lados de la recta, hay rectas punteadas negras que delimitan el margen. Los vectores soporte se encuentran encerrados en círculos.

Ejemplo 2.4.2. Tomando los datos del Ejemplo 2.3.2, la figura 2.6 muestra cómo el método SVM clasifica dos conjuntos de datos bidimencionales linealmente no separables. Los datos de la clase 1 son representados por cuadros y los de la clases 2 por triángulos. Ambas clases tienen 10 elementos. La recta de separación tiene por ecuación $y = -4.7x - 2.71$ y su vector de peso óptimo $\widehat{\boldsymbol{\beta}} = [-1.13475158, -0.24129287]$ y sesgo $\widehat{\beta}_0 = -0.6545$. Los vectores soporte son aquellos que tienen relleno, se tiene un vector soporte en la clase 1 y dos en la clase 2. El número de datos de entrenamiento mal clasificados es 2 (señalados con cuadrados en la figura), por lo que la proporción de error de clasificación de los datos es $2/20 = 0.10$.

Clase	X	Y
1	4.11409	-0.480071
	3.41325	0.397884
	6.06066	0.0463882
	1.09202	-0.800754
	6.54042	0.701325
	6.70795	0.0903986
	5.10956	-1.34345
	10.0552	1.04051
	8.01616	0.998321
	8.05375	-0.123273
2	-10.8275	-1.10702
	-6.91894	-0.710387
	-5.5833	0.701731
	-4.03035	1.34567
	-5.36154	0.504675
	-5.74892	1.79428
	-6.77187	-0.425007
	-8.9489	-1.19121
	-7.19463	-1.14488
	-3.5301	1.37978

Tabla 2.2: Datos generados aleatoriamente a partir de dos distribuciones normales bivariadas.

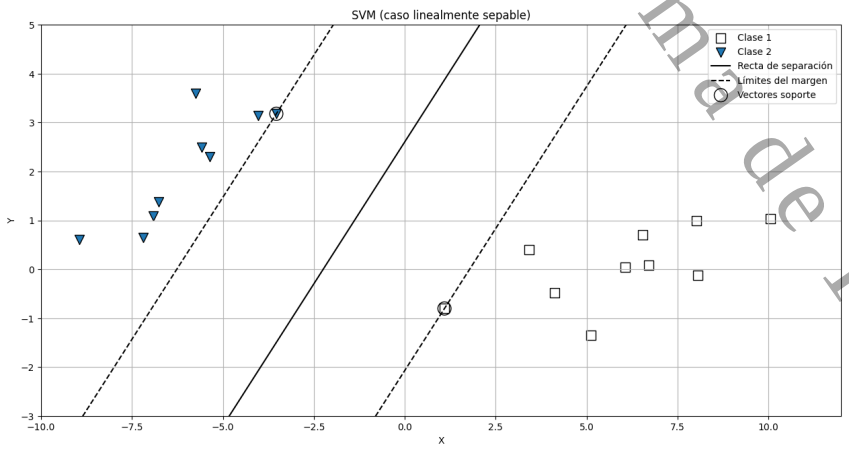

Figura 2.5: Método SVM en el caso linealmente separable para los datos generados.

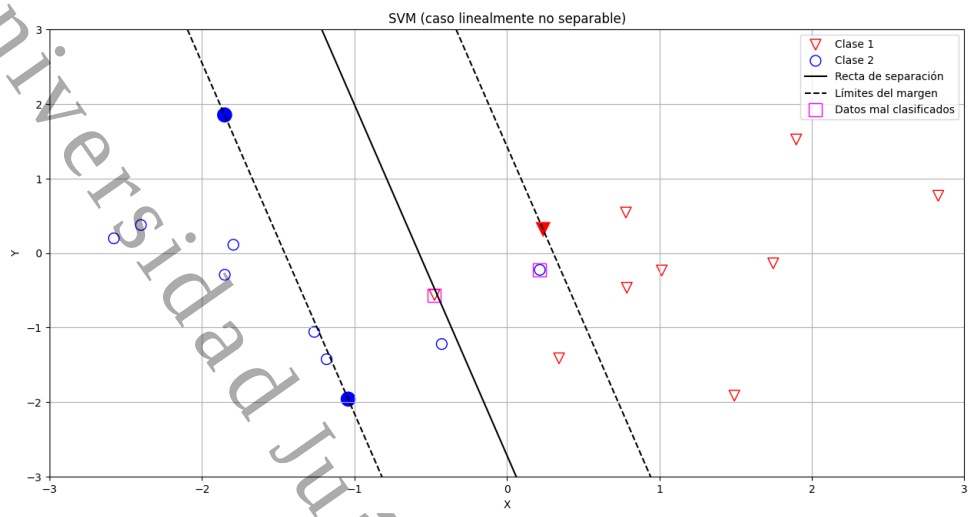

Figura 2.6: SVM en el caso linealmente no separable para los datos de la tabla 2.1.

Parte II

Resultados

Capítulo 3

Aplicaciones

En este capítulo mostraremos algunos ejemplos de aplicación del Discriminante Lineal de Fisher y Support Vector Machine. Utilizando datos reales, se compararán ambos métodos mediante matrices de confusión y proporciones de error de clasificación. Los conjuntos de datos considerados fueron estudiados en [13] y se pueden obtener en `https://archive.ics.uci.edu/datasets`. Proporcionaremos algunos análisis adicionales de estos datos a lo hecho en [13]. Los códigos en Python de lo presentado en este capítulo puede ser consultado en el Apéndice B.

3.1. Análisis de datos de cáncer de mama

Los métodos para diagnosticar el cáncer son la mamografía, el aspirado con la aguja fina (FNA) y la biopsia quirúrgica, siendo la biopsia la más precisa pero también la más costosa. En la Universidad de Wisconsin-Madison se ha desarrollado un algoritmo de obtención de imágenes con el objetivo de desarrollar un procedimiento que diagnostique las FNA con gran precisión. Se utiliza una aguja de pequeño calibre para extraer una muestra de líquido (es decir, una FNA) de un bulto o masa mamaria de una paciente (detectada mediante autoexploración y/o mamografía); la FNA se coloca en un portaobjetos de cristal y se tiñe para resaltar los núcleos de las células constituyentes; una cámara de video montada en un microscopio transfiere una imagen de la FNA a una estación de trabajo y se determinan los límites exactos de los núcleos. Se calculan diez variables del núcleo de cada célula a partir de muestras de fluido; ver tabla 3.1.

Las variables se construyen de modo que los valores mayores indiquen normalmente una mayor probabilidad de malignidad. Para cada imagen compuesta de 10 a 40 núcleos, se calculan el valor medio (mv), el valor extremo (es decir, el mayor o peor valor, el mayor tamaño, la forma más irregular) (ev) y la desviación estándar (sd) de cada una de estas características celulares, lo que da como resultado un total de 30 variables reales. Para el estudio se utiliza el logaritmo natural de las 30 variables.

El conjunto de datos consta de 569 casos (imágenes), de los cuales 212 fueron

1.	Radio	Radio de un núcleo individual.
2.	Textura	Varianza de los niveles de gris dentro del límite del núcleo.
3.	Perímetro	Distancia alrededor del perímetro del núcleo.
4.	Área	Área del núcleo.
5.	Suavidad	Suavidad del contorno de un núcleo medida por la variación local del segmento radial.
6.	Compacidad	Medida de la compacidad del núcleo de una célula mediante la fórmula $(\text{perímetro})^2/\text{área}$.
7.	Concavidad	Gravedad de las concavidades o hendiduras en un núcleo celular utilizando una medida de tamaño que enfatiza las pequeñas hendiduras.
8.	Puntos cóncavos	Número de puntos cóncavos o hendiduras en el núcleo de una célula.
9.	Simetría	Simetría del núcleo de una célula.
10.	Dimensión fractal	Dimensión fractal (del límite) de una célula.

Tabla 3.1: Diez variables para el estudio del cáncer de mama de Wisconsin.

diagnosticados como malignos (confirmados por biopsia) y 357 como benignos (confirmados por biopsia o por exámenes médicos periódicos posteriores). Nos interesa separar los tumores malignos de los benignos (sin realizar una intervención quirúrgica) utilizando los datos del estudio.

Para clasificar entre los tumores benignos y malignos, se utilizarán el Discrimiante Lineal de Fisher y el método Support Vector Machine, utilizando las 30 variables de los datos. Los parámetros estimados de los clasificadores DLF y SVM se presentan en las tablas 3.2 y 3.3, respectivamente.

Para comparar los métodos, se lleva a cabo para cada uno el procedimiento de validación cruzada leave-one-out (CV/n), que elimina una observación del conjunto de datos, entrena la regla de clasificación con el método en cuestión a partir de las $n-1$ observaciones restantes y, a continuación, clasifica la observación omitida; el procedimiento se repite 569 veces (una por cada observación del conjunto de datos). Las tablas de confusión para clasificar las 569 observaciones con el DLF y SVM se presentan en la tabla 3.4 y la tabla 3.5, respectivamente.

En la tabla 3.4, vemos que de los 212 tumores malignos, 192 se clasifican correctamente y 20 no; y de los 357 tumores benignos, 353 se clasifican correctamente y 4 no. La tasa de clasificación errónea del DLF en este ejemplo es, por tanto, estimada por $24/569 = 0.042$, es decir, 4.2 %. En la tabla 3.5, vemos que de los 212 tumores malignos 203 son clasificados correctamente y 9 no; mientras que de los 357 tumores benignos 354 fueron clasificados correctamente y 3 no. La tasa de clasificación errónea del SVM en este ejemplo es estimada por $12/569 = 0.021$, o bien 2.1 %. Esto nos dice que ambos métodos son buenos para clasificar, sin embargo el SVM es el que tiene el mejor resultado. En la tabla 3.6 comparamos los métodos con sus respectivos errores de clasificación.

Variable	Coef	Variable	Coef	Variable	Coef
Radio.mv	-116.674	Radio.sd	-10.088	Radio.ev	23.807
Textura.mv	-1.193	Textura.sd	-2.308	Textura.ev	8.865
p.mv	135.268	*p*.sd	1.019	*p*.ev	-12.197
a.mv	-9.121	*a*.sd	13.153	*a*.ev	-7.161
Suav.mv	0.980	Suav.sd	0.522	Suav.ev	6.020
Comp.mv	-8.245	Comp.sd	-1.613	Comp.ev	2.077
Conc.mv	5.284	Conc.sd	0.231	Conc.ev	-4.528
P. cónc.mv	1.988	P. cónc.sd	3.641	P. cónc.ev	-3.641
Sim.mv	-4.730	Sim.sd	-2.043	Sim.ev	11.231
D. Fractal.mv	-13.860	D. Fractal.sd	-1.972	D. Fractal.ev	15.956
$\widehat{b}_0$	-282.859				

Tabla 3.2: Coeficientes estimados de la función discriminante lineal de Fisher para los datos de diagnóstico de cáncer de mama de Wisconsin. Todas las variables son logaritmos de las variables originales.

En las figuras 3.1 y 3.2 se muestran, respectivamente, los histogramas de los valores de la función de respuesta $L(\boldsymbol{x}) = \boldsymbol{b}^\top \boldsymbol{x} + b_0$ del DLF, para los tumores benignos y los tumores malignos. Las observaciones con un valor de la función de respuesta mayor que 0 son clasificadas como malignas (M), y las observaciones con un valor menor o igual que 0 son clasificadas como benignas (B).

La figura 3.3 muestra los dos histogramas anteriores juntos. Si la mayoría de los valores de la función de respuesta para los casos malignos están a la derecha del umbral de decisión (0), significa que el modelo está clasificando aceptablemente estos casos como malignos. Si la mayoría de los valores de la función de respuesta para los casos benignos están a la izquierda del umbral de decisión, significa que el modelo está clasificando aceptablemente estos casos como benignos. Se observa que como la mayoría de los datos son correctamente clasificados, el método DLF tiene un buen desempeño.

Los histogramas anteriores ofrecen una visualización clara de cómo el Discriminante Lineal de Fisher separa las dos clases de observaciones (malignas y benignas), y ayuda a evaluar la efectividad del modelo DLF.

En las figuras 3.4 y 3.5 se muestran, respectivamente, los histogramas de los valores obtenidos de la función de respuesta del SVM, para los tumores benignos y los tumores malignos del cáncer de mama de Wisconsin. Las observaciones con un valor de la función de respuesta mayor que 0 son clasificadas como malignas (M) y las observaciones con un valor menor o igual que 0 son clasificadas como benignas (B).

La figura 3.6 muestra los dos histogramas anteriores juntos. Observamos en la figura que la mayoría de los datos de tumores malignos están a la derecha del umbral de decisión (0) y la mayoría de los datos de tumores benignos están a la izquierda

Variable	Coef	Variable	Coef	Variable	Coef
Radio.mv	0.150	Radio.sd	0.426	Radio.ev	0.435
Textura.mv	1.689	Textura.sd	-0.486	Textura.ev	2.212
p.mv	0.139	*p*.sd	0.254	*p*.ev	0.427
a.mv	0.351	*a*.sd	1.136	*a*.ev	0.966
Suav.mv	0.265	Suav.sd	0.585	Suav.ev	0.915
Comp.mv	-0.910	Comp.sd	-0.895	Comp.ev	0.187
Conc.mv	0.546	Conc.sd	0.057	Conc.ev	0.886
P. cónc.mv	1.165	P. cónc.sd	-0.115	P. cónc.ev	0.527
Sim.mv	0.126	Sim.sd	-0.511	Sim.ev	1.044
D. Fractal.mv	-0.184	D. Fractal.sd	-0.285	D. Fractal.ev	0.661
$\widehat{\beta}_0$	-22.712				

Tabla 3.3: Coeficientes estimados por el método SVM para los datos de diagnóstico de cáncer de mama de Wisconsin.

del umbral, lo que indica un buen desempeño del método de clasificación SVM.

	Benigno predicho	Maligno predicho	Total
Benigno verdadero	353	4	357
Maligno verdadero	20	192	212
Total	373	196	569

Tabla 3.4: Tabla de confusión para los datos de diagnóstico del cáncer de mama de Wisconsin utilizando el método DLF.

	Benigno predicho	Maligno predicho	Total
Benigno predicho	354	3	357
Maligno predicho	9	203	212
Total	363	206	569

Tabla 3.5: Tabla de confusión para los datos de diagnóstico del cáncer de mama de Wisconsin utilizando el método SVM.

Clasifcador	DLF	SVM
Tasa de error de clasificación	0.042	0.021

Tabla 3.6: Comparación de las tasas de error de clasificación para los datos de diagnóstico del cáncer de mama de Wisconsin utilizando los métodos DLF y SVM.

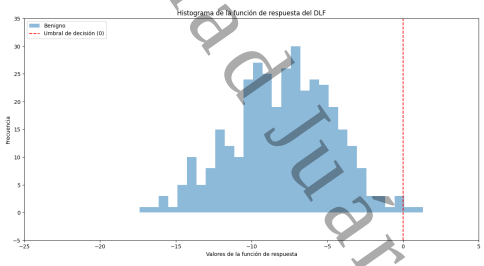

Figura 3.1: Clasificación de datos benignos de cáncer de mama por el método DLF.

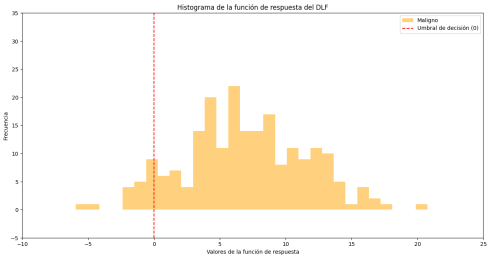

Figura 3.2: Clasificación de datos malignos de cáncer de mama por el método DLF.

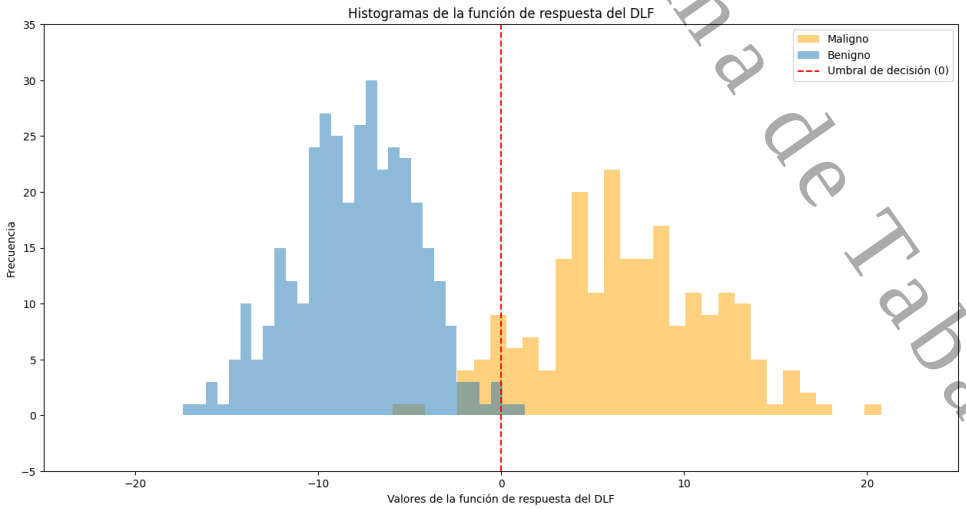

Figura 3.3: Clasificación de datos de cáncer de mama por el método DLF.

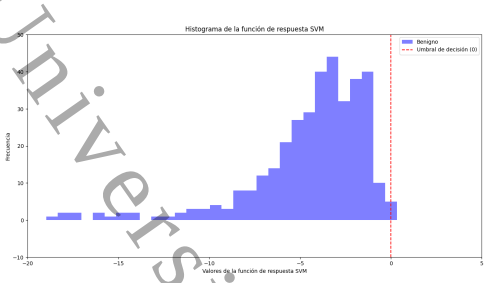

Figura 3.4: Clasificación de datos benignos de cáncer de mama por el método SVM.

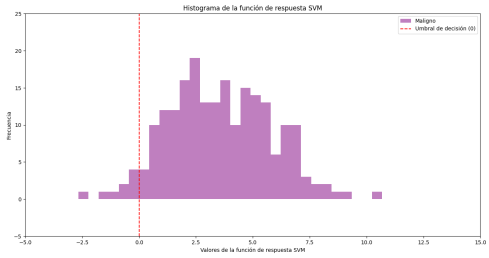

Figura 3.5: Clasificación de datos malignos de cáncer de mama por el método SVM.

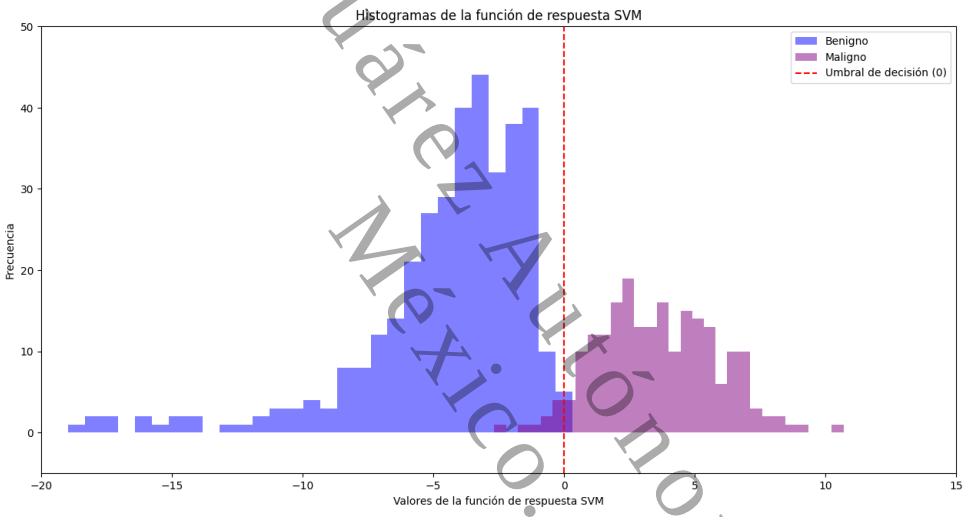

Figura 3.6: Clasificación de datos de cáncer de mama por el método SVM.

3.2. Análisis de correo electrónico no deseado

Este conjunto de datos se deriva de una colección de correos electrónicos spam (correo electrónico comercial no solicitado, que provenía de un administrador de correo y correos electrónicos personales que habían sido catalogados como spam) y correos electrónicos no spam (que provenían de correos electrónicos de trabajo y personales de confianza). Hay 57 variables (atributos) y cada mensaje está etiquetado a una de las dos clases: no spam o spam. La mayoría de las variables indican si una palabra o caracter en particular aparecía con frecuencia en el correo electrónico: 48 variables tienen la forma "word freq WORD", que indica el porcentaje de palabras en el correo electrónico que coinciden con WORD; 6 variables tienen la forma "word freq CHAR", que indica el porcentaje de caracteres en el correo electrónico que

coinciden con CHAR; y 3 variables de "longitud de secuencia", que miden la longitud promedio, la longitud de la más larga y la suma de las longitudes de las secuencias ininterrumpidas de letras mayúsculas consecutivas. Hay 1813 observaciones de spam (39.4 %) y 2788 observaciones de no spam en el conjunto de datos.

Debido a que la cantidad de datos es muy grande, para comparar los métodos, se llevó a cabo para cada uno el procedimiento de validación cruzada k-fold, en el cual se divide aleatoriamente todo el conjunto de datos en k grupos de aproximadamente el mismo tamaño; se omite uno de los grupos y se ajusta el modelo utilizando los datos combinados de los otros $k-1$ grupos (que forma el conjunto de aprendizaje); se utiliza el grupo omitido como conjunto de prueba, se predicen sus valores de salida utilizando el modelo ajustado y se calcula el error de predicción para el grupo omitido; se repite este procedimiento k veces, omitiendo cada vez un grupo diferente. En este caso se usó $k = 10$ para ambos métodos. Los resultados se muestran en las tablas 3.7 y 3.8. La tabla 3.9 muestra la comparación de los métodos, en la cual se ve que SVM es más preciso.

En las figuras 3.7 y 3.8 se muestran, respectivamente, los histogramas de los valores de la función de respuesta $L(\boldsymbol{x})$ del DLF, para los conjuntos de datos de spam y de no spam. En la figura 3.9 se muestran en una sola gráfica los dos histogramas. Por otro lado, en las figuras 3.10, 3.11 y 3.12 se muestran los histogramas análogos para los valores obtenidos de la función de respuesta $L(\boldsymbol{x})$ del método SVM, para los mismos conjuntos de datos. En los histogramas se observa que la mayoría de datos de no spams se encuentran del lado izquierdo del umbral igual a cero, y que la mayoría de los datos de spams se encuentran del lado derecho del umbral, por lo que ambos métodos clasifican de forma aceptable los datos. Sin embargo, de las tablas de confusión 3.7, 3.8 y la tabla 3.9 vemos que el método SVM hace una mejor clasificación de los datos, porque su tasa de error es casi la mitad que la tasa de error obtenida con el método DLF.

	No-spam predicho	Spam predicho	Total
No-spam verdadero	2657	131	2788
Spam verdadero	394	1419	1813
Total	3051	1550	4601

Tabla 3.7: Tabla de confusión para los datos de emails spam y no spam utilizando el método DLF.

	No-spam predicho	Spam predicho	Total
No-spam verdadero	2657	131	2788
Spam verdadero	186	1627	1813
Total	2843	1758	4601

Tabla 3.8: Tabla de confusión para los datos de emails spam y no spam utilizando el método SVM.

Clasificador	DLF	SVM
Tasa de error de clasificación	0.1141	0.0689

Tabla 3.9: Comparación de las tasas de error de clasificación para los emails spam y no spam utilizando los métodos DLF y SVM.

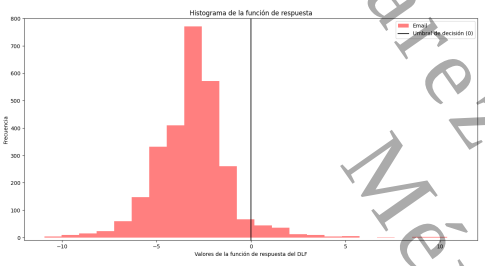

Figura 3.7: Clasificación de emails no spam por el método DLF.

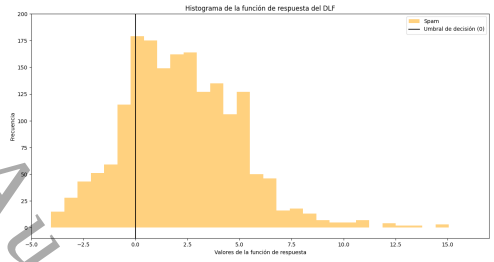

Figura 3.8: Clasificación de emails spam por el método DLF.

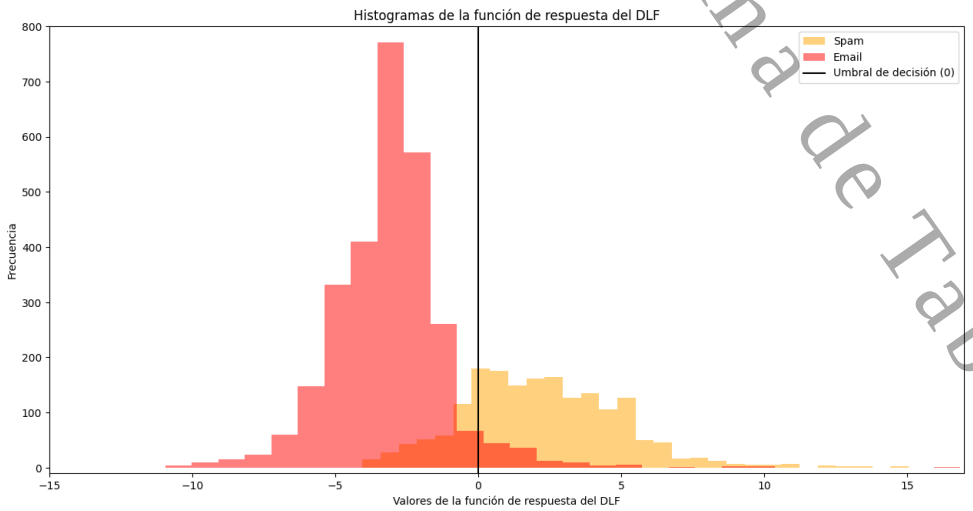

Figura 3.9: Clasificación de emails no spam y spam por el método DLF.

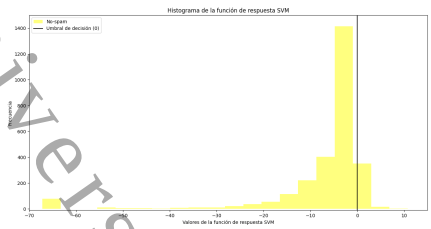

Figura 3.10: Clasificación de emails no spam por el método SVM.

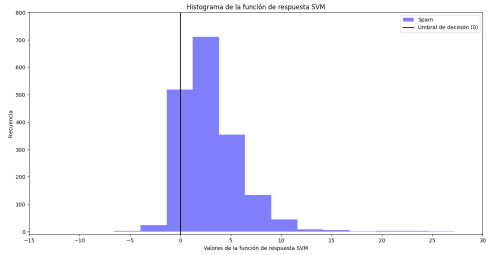

Figura 3.11: Clasificación de emails spam por el método SVM.

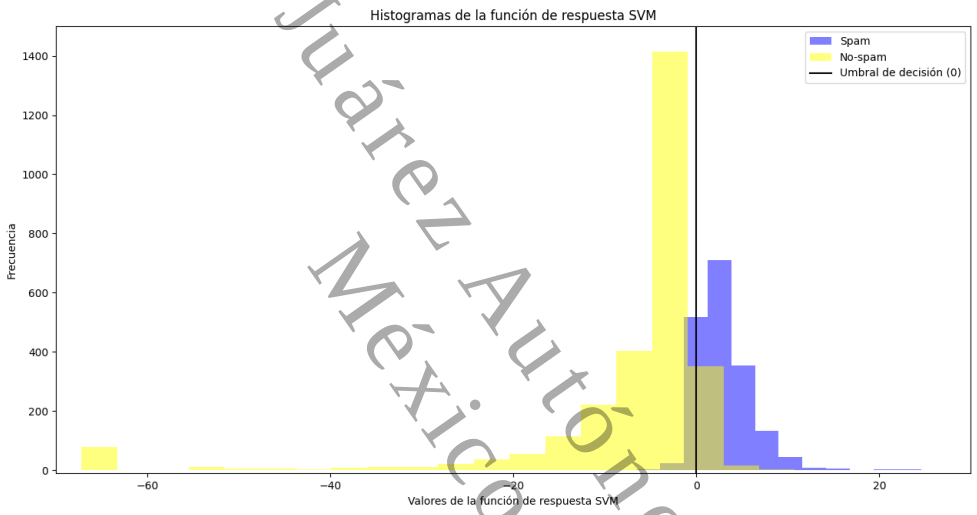

Figura 3.12: Clasificación de emails no spam y spam por el método SVM.

3.3. Análisis de datos de señales de sonar

Para finalizar, se analizará el conjunto de datos de señales de sonar. El sonar (Sound Navigation and Ranging) es una tecnología que utiliza ondas de sonido para detectar y localizar objetos en el agua, ver *What is sonar?* [National Ocean Service], 2024. Hay dos tipos principales de sonar:

- *Sonar activo*: Este sistema emite pulsos de sonidos y luego detecta los ecos que rebotan en los objetos sumergidos. Al medir el tiempo que tarda el eco en regresar, el sonar puede calcular la distancia al objeto.

- *Sonar pasivo*: Este tipo de sonar no emite ningún sonido, sino que simplemente escucha los sonidos producidos por otras fuentes, como barcos, submarinos o

animales marinos. Es útil para detectar la presencia de objetos sin revelar la propia ubicación.

El sonar se utiliza principalmente en navegación marítima, investigaciones submarinas, pesca y operaciones militares, como la detección de submarinos o minas. También es empleado por científicos para estudiar la vida marina y mapear el fondo del océano, ver [22].

El conjunto de datos de sonar que se analizará consiste en mediciones tomadas por un sonar activo. Cada registro en el conjunto de datos contiene información sobre las respuestas de los ecos de sonar reflejados por un objeto bajo el agua. Estas respuestas, o firmas acústicas, se procesan en varios ángulos de visión y bajo diversas condiciones para determinar si el objeto es una mina o una roca. Hay 111 observaciones obtenidas al rebotar el sonar en un cilindro de metal y 97 obtenidas de la roca. La señal de sonar transmitida es un chirrido de frecuencia modulada, que aumenta en frecuencia. El conjunto de datos contiene señales obtenidas desde una variedad de ángulos de visión, abarcando 90 grados para el cilindro y 180 grados para la roca. Cada observación es un conjunto de 60 números en el rango 0-1, donde cada número representa la energía dentro de una banda de frecuencia particular, integrada durante un cierto periodo de tiempo.

Para este ejemplo se aplican los dos métodos de clasificación DLF y SVM, así como validación cruzada con leave-one-out (CV/n). Los resultados obtenidos se muestran en las tablas 3.10 y 3.11; en la tabla 3.12 se muestra la comparación de resultados de ambos métodos, donde se observa una vez más que el método SVM tiene la menor tasa de error de clasificación.

Las figuras 3.13 y 3.14 muestran los histogramas de los valores de la función de respuesta $L(\boldsymbol{x})$ obtenida con el DLF para los datos de minas y rocas, respectivamente, y la figura 3.15 muestra los dos histogramas anteriores juntos. Se observa que el método de clasificación DLF es aceptable ya que los datos en su mayoría están bien clasificados. Las figuras 3.16 y 3.17 muestran los valores de la función $L(\boldsymbol{x})$ obtenida con el método SVM para los datos de minas y rocas, respectivamente, y la figura 3.18 muestra los dos histogramas juntos, donde observamos que el método SVM clasifica la mayor parte de los datos correctamente, siendo también aceptable para la clasificación de estos datos.

	Mina predicha	Roca predicha	Total
Mina verdadera	87	24	111
Roca verdadera	27	70	97
Total	114	94	208

Tabla 3.10: Tabla de confusión para el conjunto de datos de sonar utilizando el método DLF.

En los resultados obtenidos se observa que ambos métodos producen buenos resultados para la clasificación de este conjunto de datos, sin embargo la proporción

	Mina predicha	Roca predicha	Total
Mina verdadera	96	15	111
Roca verdadera	30	67	97
Total	126	82	208

Tabla 3.11: Tabla de confusión para el conjunto de datos de sonar utilizando el método SVM.

Clasificador	DLF	SVM
Tasa de error de clasificación	0.245	0.216

Tabla 3.12: Comparación de las tasas de error de clasificación para los datos de sonar utilizando los métodos DLF y SVM.

de error de clasificación no es muy baja.

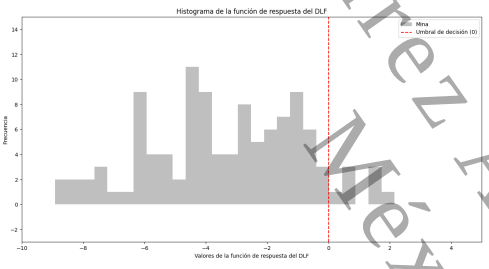

Figura 3.13: Clasificación de datos mina por el método DLF.

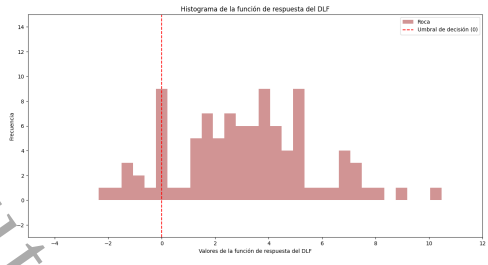

Figura 3.14: Clasificación de datos roca por el método DLF.

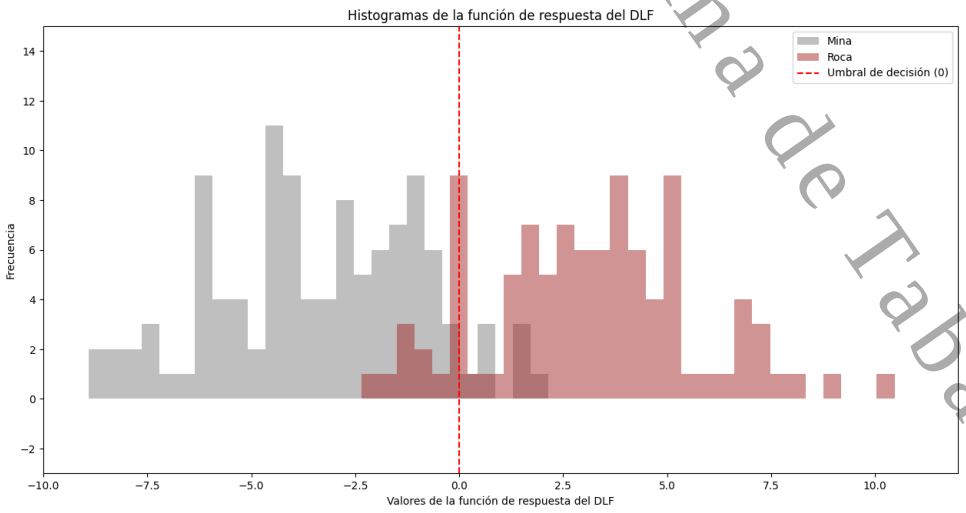

Figura 3.15: Clasificación de datos del sonar por el método DLF.

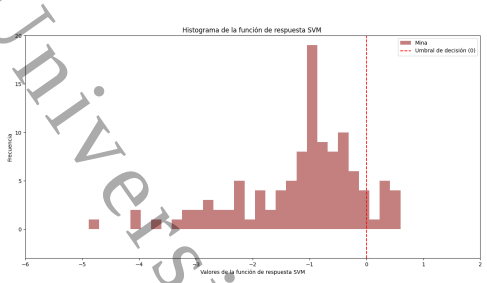

Figura 3.16: Clasificación de datos mina por el método SVM.

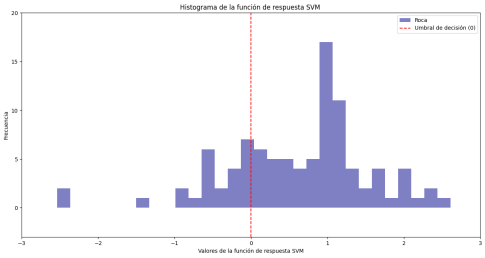

Figura 3.17: Clasificación de datos roca por el método SVM.

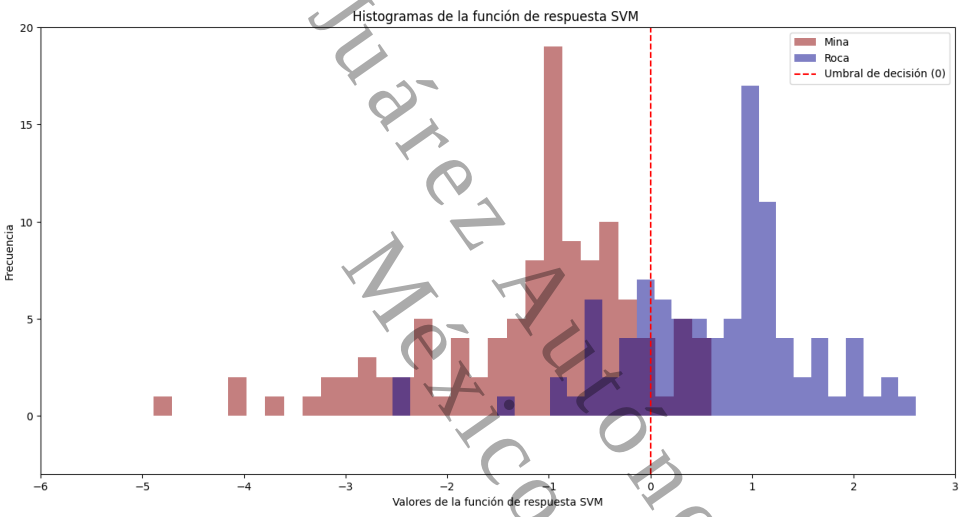

Figura 3.18: Clasificación de datos del sonar por el método SVM.

Conclusión

En esta tesis se estudiaron métodos de clasificación binaria lineales, así como su aplicación a datos reales. A lo largo del estudio, nos concentramos en dos metodologías ampliamente utilizadas: el Discriminante Lineal de Fisher (DLF) y Support Vector Machine (SVM). Estos métodos han demostrado ser herramientas poderosas en la tarea de clasificación, lo que permite asignar nuevos datos a categorías predefinidas basándose en las características extraídas de los datos.

El análisis y los experimentos realizados en este trabajo proporcionaron una comparación más amplia entre ambos métodos. El DLF, basado en la teoría estadística clásica, busca la máxima separación entre clases mediante la combinación lineal óptima de características. A su vez, SVM se enfoca en maximizar el margen entre las clases, lo que lo hace particularmente efectivo en problemas donde los datos no son linealmente separables.

Uno de los principales desafíos que enfrentan ambos métodos es la capacidad para manejar datos en diferentes dominios y formatos. Los tres conjuntos de datos utilizados, los datos de cáncer de mama de Wisconsin, los datos de correos electrónicos no deseados y los datos de señales de sonar, representan escenarios reales y diversos donde los modelos de clasificación desempeñan un papel crucial en la toma de decisiones. En estos casos, se utilizó validación cruzada para medir la precisión de los modelos, lo que permitió observar que el rendimiento de SVM fue superior en los tres conjuntos de datos estudiados. Por otro lado, el DLF mostró ventajas en cuanto a un menor costo computacional y una implementación más simple, algunas veces produciendo resultados cercanos a los de SVM.

El uso de Python como lenguaje de programación principal, junto con bibliotecas como NumPy, Scikit-learn y Pandas, facilitó el desarrollo e implementación de los modelos. Estas herramientas permitieron realizar cálculos eficientes, generar matrices de confusión y visualizar los resultados de forma clara y precisa, lo que evidencia la versatilidad y potencia de Python en el campo del análisis de datos y el aprendizaje automático.

Finalmente, se puede concluir que tanto el DLF como SVM tienen sus fortalezas y limitaciones dependiendo del tipo de datos con los que se esté trabajando. Esta tesis ha contribuido a una mejor comprensión de cómo seleccionar el método adecuado para cada escenario, ofreciendo una visión clara de cómo estas metodologías pueden utilizarse en aplicaciones de clasificación binaria en el mundo real. Además, la im-

plementación en Python proporciona una base sólida para futuras investigaciones y desarrollos en este campo.

Apéndice A

Teoría de matrices

A.1. Matriz definida-positiva

Los siguientes resultados fueron tomados de [10].

Definición A.1.1. Sea $A = (a_{ij})$ una matriz de tamaño $m \times n$ con elementos complejos. Entonces la *transpuesta conjugada* de A, denotada por A^*, es la matriz de tamaño $n \times m$ que tiene por entrada ij-ésima a $a^*_{ij} = \overline{a}_{ji}$, donde $\bar{a}$ es el conjugado de a.

Definición A.1.2. La matriz compleja A de tamaño $n \times n$ se llama *hermitiana* si $A^* = A$.

Definición A.1.3. Sea A una matriz hermitiana de tamaño $n \times n$. La matriz A cumple con ser *definida positiva* si satisface alguna de las siguientes condiciones equivalentes:

1. Para todo vector no nulo $\boldsymbol{z} \in \mathbb{C}^n$, tenemos que
$$\boldsymbol{z}^* A \boldsymbol{z} > 0.$$
2. Todos los autovalores λ_i de A son positivos.
3. La función
$$\langle \boldsymbol{x}, \boldsymbol{y} \rangle = \boldsymbol{y}^* A \boldsymbol{x},$$
define un producto interno en $\mathbb{C}^n$.

Algunas propiedades de las matrices definidas positivas son:

1. Toda matriz definida positiva es invertible y su inversa es definida positiva.
2. Si A es una matriz definida positiva y $r > 0$ es un número real, entonces rA es definida positiva.
3. Si A y B son matrices definidas positivas, entonces la suma $A + B$ es definida positiva.
4. Si A y B son matrices definidas positivas y se cumple que $AB = BA$, entonces AB es una matriz definida positiva.

Apéndice B

Códigos

A continuación se proporcionan todos los códigos en Python elaborados para esta tesis.

B.1. Códigos del Capítulo 2

Del Ejemplo 2.3.2.

```
import numpy as np
import matplotlib.pyplot as plt
from sklearn.discriminant_analysis import
↪  LinearDiscriminantAnalysis


# Definir las medias y la matriz de covarianza
media_1 = np.array([1.25, 0])
media_2 = np.array([-1.25, 0])
covarianza = np.identity(2)
# Matriz de covarianza igual a la identidad

# Generar datos para cada clase
np.random.seed(42)
datos_clase_1 = np.random.multivariate_normal(media_1,
↪  covarianza, 10)
datos_clase_2 = np.random.multivariate_normal(media_2,
↪  covarianza, 10)

# Concatenar datos y etiquetas
X = np.concatenate((datos_clase_1, datos_clase_2))
y = np.concatenate((np.zeros(10), np.ones(10)))  # Clase
↪  1: 0, Clase 2: 1
```

```
# Inicializar y ajustar el modelo de discriminante lineal
↪  de Fisher
lda = LinearDiscriminantAnalysis()
lda.fit(X, y)

# Visualizar los datos y la línea de separación
plt.figure(figsize=(8, 6))

# Plot de los datos de la clase 1
plt.scatter(datos_clase_1[:, 0], datos_clase_1[:, 1],
↪  color='blue', label='Clase 1')

# Plot de los datos de la clase 2
plt.scatter(datos_clase_2[:, 0], datos_clase_2[:, 1],
↪  color='red', label='Clase 2')

# Plot de la línea de separación
coef = lda.coef_[0]
intercept = lda.intercept_
x_values = np.linspace(-3, 3, 100)
y_values = -(coef[0] / coef[1]) * x_values - (intercept /
↪  coef[1])
plt.plot(x_values, y_values, color='black', linestyle='--',
↪  label='Recta del DLF')


plt.title('Discriminante Lineal de Fisher')
plt.xlabel('X')
plt.ylabel('Y')
plt.legend()
plt.grid(True)

plt.xlim(-3, 3)
plt.ylim(-3, 3)

plt.show()
```

Del Ejemplo 2.4.1.

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

# Definir las medias y la matriz de covarianza
media_1 = np.array([5.4, 0])
media_2 = np.array([-7, 2])
```

```
covarianza = np.array([[7, 2], [2, 1]])

# Generar datos para cada clase
np.random.seed(42)
datos_clase_1 = np.random.multivariate_normal(media_1,
↪  covarianza, 10)
datos_clase_2 = np.random.multivariate_normal(media_2,
↪  covarianza, 10)

# Concatenar datos y etiquetas
X = np.concatenate((datos_clase_1, datos_clase_2))
y = np.concatenate((np.zeros(10), np.ones(10)))  # Clase
↪  1: 0, Clase 2: 1

# Inicializar y ajustar el modelo de Support Vector
↪  Machine
svc = SVC(kernel='linear')
svc.fit(X, y)

# Obtener la pendiente e intercepto de la frontera de
↪  decisión
w = svc.coef_[0]
a = -w[0] / w[1]
b = -svc.intercept_[0] / w[1]
xx = np.linspace(-10, 10)
yy = a * xx + b

# Calcular los márgenes
yy1 = a * xx - (svc.intercept_[0]-1) / w[1]
yy2 = a * xx - (svc.intercept_[0]+1) / w[1]
#print(f"Ecuación de la recta de separación: Y = {a:.2f}
↪  * X + {b:.2f}")
# Calcular los márgenes


# Graficar los puntos, la frontera de decisión, y los
↪  márgenes
plt.figure(figsize=(8, 6))

# Plot de los datos de la clase 1
plt.scatter(X[y == 0, 0], X[y == 0, 1], marker='s',
↪  edgecolors='k', s=100, facecolors='none', label='Clase
↪  1')

# Plot de los datos de la clase 2
```

```
plt.scatter(X[y == 1, 0], X[y == 1, 1], marker='v',
↪  edgecolors='k', s=100, label='Clase 2')

# Graficar la frontera de decisión y los márgenes
plt.plot(xx, yy, 'k-', label='Recta de separación')
plt.plot(xx, yy1, 'k--', label='Límite del margen')
plt.plot(xx, yy2, 'k--')

# Resaltar los vectores de soporte
support_vector_indices = svc.support_
plt.scatter(X[support_vector_indices, 0],
↪  X[support_vector_indices, 1],
s=200, facecolors='none', edgecolors='k', label='Vectores
↪  soporte')

#Imprimir el sesgo y el intercepto
print(w)
print(svc.intercept_[0])

plt.title('SVM (caso linealmente sepable)')
plt.xlabel('X')
plt.ylabel('Y')
plt.legend()
plt.grid(True)
plt.xlim(-10, 12)
plt.ylim(-3, 5)

plt.show()
```

Del Ejemplo 2.4.2.

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

# Definir las medias y la matriz de covarianza
media_1 = np.array([1.25, 0])
media_2 = np.array([-1.25, 0])
covarianza = np.identity(2)  # Matriz de covarianza igual
↪  a la identidad

# Generar datos para cada clase
np.random.seed(42)
```

```
datos_clase_1 = np.random.multivariate_normal(media_1,
↪  covarianza, 10)
datos_clase_2 = np.random.multivariate_normal(media_2,
↪  covarianza, 10)

# Concatenar datos y etiquetas
X = np.concatenate((datos_clase_1, datos_clase_2))
y = np.concatenate((np.zeros(10), np.ones(10)))  # Clase
↪  1: 0, Clase 2: 1

# Inicializar y ajustar el modelo de Support Vector
↪  Machine
svc = SVC(kernel='linear')  # Ajuste para asegurarse de
↪  que la solución es lineal
svc.fit(X, y)

# Obtener la pendiente e intercepto de la frontera de
↪  decisión
w = svc.coef_[0]
a = -w[0] / w[1]
b = -svc.intercept_[0] / w[1]
xx = np.linspace(-3, 3, num=200)
yy = a * xx + b
#print(f"Ecuación de la recta de separación: Y = {a:.2f}
↪  * X + {b:.2f}")

# Calcular los márgenes
yy1 = a * xx - (svc.intercept_[0] - 1) / w[1]
yy2 = a * xx - (svc.intercept_[0] + 1) / w[1]

# Identificar vectores de soporte
soportes = svc.support_vectors_

# Verificar si los soportes están sobre yy1 o yy2
def vectores_sobre_el_margen(s, margen_valor):
margen_y = a * s[0] - margen_valor / w[1]
return np.isclose(s[1], margen_y, atol=0.1)  # Tolerancia
↪  para cercanía

soportes_yy1 = np.array([s for s in soportes if
↪  vectores_sobre_el_margen(s, svc.intercept_[0] - 1)])
soportes_yy2 = np.array([s for s in soportes if
↪  vectores_sobre_el_margen(s, svc.intercept_[0] + 1)])

# Graficar los puntos, la frontera de decisión, y los
↪  márgenes
```

```
plt.figure(figsize=(8, 6))

# Plot de los datos de la clase 1 (solo bordes)
plt.scatter(X[y == 0, 0], X[y == 0, 1], marker='v',
↪  edgecolors='red', s=100, facecolors='none', label='Clase
↪  1')

# Plot de los datos de la clase 2 (solo bordes)
plt.scatter(X[y == 1, 0], X[y == 1, 1], marker='o',
↪  edgecolors='blue', s=100, facecolors='none',
↪  label='Clase 2')

# Graficar la frontera de decisión y los márgenes
plt.plot(xx, yy, 'k-', label='Recta de separación')
plt.plot(xx, yy1, 'k--', label='Límite del margen')
plt.plot(xx, yy2, 'k--')

# Graficar vectores de soporte sobre los márgenes
↪  (relleno)
plt.scatter(soportes_yy1[:, 0], soportes_yy1[:, 1],
↪  marker='o', s=200, facecolors='blue', edgecolors='none',
↪  label='')
plt.scatter(soportes_yy2[:, 0], soportes_yy2[:, 1],
↪  marker='v', s=200, facecolors='red', edgecolors='none',
↪  label='')

# Predicciones y mal clasificados (solo bordes)
y_pred = svc.predict(X)
mal_clasificados = X[y != y_pred]
plt.scatter(mal_clasificados[:, 0], mal_clasificados[:,
↪  1], marker='s', s=150,
facecolors='none', edgecolors='magenta', linewidth=1,
↪  label='Datos mal clasificados')

error_proporcion = np.sum(y != y_pred) / len(y)
print(f'Proporción de error de clasificación:
↪  {error_proporcion:.2f}')

#Imprimir el sesgo y el intercepto
print(w)
print(svc.intercept_[0])

plt.title('SVM (caso linealmente no separable)')
plt.xlabel('X')
plt.ylabel('Y')
plt.legend()
```

```
plt.grid(True)
plt.xlim(-3, 3)
plt.ylim(-3, 3)

plt.show()
```

B.2. Códigos del Capítulo 3

B.2.1. Códigos de la Sección 3.1

Análisis de datos del cáncer de mama por DLF:

```
from ucimlrepo import fetch_ucirepo
from sklearn.discriminant_analysis import
↪  LinearDiscriminantAnalysis
from sklearn.model_selection import LeaveOneOut
from sklearn.metrics import accuracy_score
from sklearn.metrics import confusion_matrix
import numpy as np
import pandas as pd

# Obtener el conjunto de datos desde UCIML Repository
breast_cancer_wisconsin_diagnostic = fetch_ucirepo(id=17)

# Datos (como dataframes de pandas)
X0 = breast_cancer_wisconsin_diagnostic.data.features
y = breast_cancer_wisconsin_diagnostic.data.targets

# Se verifica si hay entradas iguales a 0 en X0
indices0=(X0==0)
np.sum(indices0)

# Se reemplazan las entradas de X0 iguales a 0 por 0.001
X1=X0.replace(0,0.001)
indices1=(X1==0)
np.sum(indices1)

# Se calcula el logaritmo de las entradas de X1
X = np.log(X1)


#y = np.ravel(y)
# Inicializar el modelo LDA (Análisis Discriminante
↪  Lineal)
lda = LinearDiscriminantAnalysis()
```

```
# DLF usando todos los datos
y = np.ravel(y)
lda.fit(X, y)

# Inicializar la validación cruzada Leave-One-Out
loo = LeaveOneOut()

# Listas para almacenar los valores verdaderos y
↪  predichos
y_true = []
y_pred = []

# Realizar la validación cruzada Leave-One-Out
for train_index, test_index in loo.split(X):
X_train, X_test = X.iloc[train_index], X.iloc[test_index]
y_train, y_test = y[train_index], y[test_index]

# Entrenar el modelo LDA
lda.fit(X_train, y_train)

# Predecir la muestra de prueba
y_pred.append(lda.predict(X_test)[0])
y_true.append(y_test[0])

# Convertir a arreglos de numpy para análisis
y_true = np.array(y_true)
y_pred = np.array(y_pred)

# Calcular la matriz de confusión
conf_matrix = confusion_matrix(y_true, y_pred)

# Convertir la matriz de confusión a un DataFrame para
↪  mejor formato
conf_matrix_df = pd.DataFrame(
conf_matrix,
index=['Verdadero benigno', 'Verdadero maligno'],
columns=['Predicho benigno', 'Predicho maligno']
)

# Agregar totales por fila y por columna
conf_matrix_df['Total fila'] = conf_matrix_df.sum(axis=1)
conf_matrix_df.loc['Total columna'] =
↪  conf_matrix_df.sum(axis=0)
```

```
# Imprimir la matriz de confusión
print(conf_matrix_df)
# Calcular la proporción de clasificación errónea
accuracy = accuracy_score(y_true, y_pred)
misclass_rate = 1 - accuracy

print("Proporción de clasificación errónea (LOOCV):",
↪  misclass_rate)

# Después del bucle LOOCV, el modelo `lda` estará
↪  entrenado con todos los datos
coefficients = lda.coef_[0]  # Obtener los coeficientes y
↪  convertirlos a una lista
#normc=np.linalg.norm(coefficients)
#print(coefficients/normc)
print(coefficients)
b0=lda.intercept_[0]
print(b0)
```

Análisis de datos del cáncer de mama utilizando SVM:

```
import numpy as np
import pandas as pd
from sklearn.model_selection import LeaveOneOut
from sklearn.svm import SVC
from sklearn.metrics import accuracy_score
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import confusion_matrix
from ucimlrepo import fetch_ucirepo

# Cargar el conjunto de datos
breast_cancer_wisconsin_diagnostic = fetch_ucirepo(id=17)

# Obtener las características y los objetivos
X0 = breast_cancer_wisconsin_diagnostic.data.features
y =
↪  breast_cancer_wisconsin_diagnostic.data.targets.values
# Convertir a arreglo NumPy

# Se verifica si hay entradas iguales a 0 en X0
indices0=(X0==0)
np.sum(indices0)

# Se reemplazan las entradas de X0 igualea a 0 por 0.001
X1=X0.replace(0,0.001)
```

```
indices1=(X1==0)
np.sum(indices1)

# Se calcula el logaritmo de las entradas de X1
X = np.log(X1)

svc = SVC(kernel='linear')
# DLF usando todos los datos
y = np.ravel(y)
svc.fit(X, y)

# Inicializar Leave-One-Out Cross-Validation
loo = LeaveOneOut()

# Listas para almacenar los valores verdaderos y
↪  predichos
y_true = []
y_pred = []

# Realizar la validación cruzada Leave-One-Out
for train_index, test_index in loo.split(X):
X_train, X_test = X.iloc[train_index], X.iloc[test_index]
y_train, y_test = y[train_index], y[test_index]


# Inicializar y ajustar el modelo SVM
svc.fit(X_train, y_train)

# Predecir la muestra de prueba
y_pred.append(svc.predict(X_test)[0])
y_true.append(y_test[0])  # Usar y_test[0] en lugar de
↪  y_test

# Calcular la matriz de confusión
conf_matrix = confusion_matrix(y_true, y_pred)

# Convertir la matriz de confusión a un DataFrame
conf_matrix_df = pd.DataFrame(
conf_matrix,
index=['Verdadero benigno', 'Verdadero maligno'],
columns=['Predicho benigno', 'Predicho maligno']
)

# Agregar totales por fila y por columna
conf_matrix_df.loc['Total fila'] =
↪  conf_matrix_df.sum(axis=0)
```

```
conf_matrix_df['Total columna'] =
↪  conf_matrix_df.sum(axis=1)

# Imprimir la matriz de confusión
print("Matriz de confusión:")
print(conf_matrix_df)
# Calcular la proporción de clasificación errónea
accuracy = accuracy_score(y_true, y_pred)
misclass_rate = 1 - accuracy

print("Proporción de clasificación errónea (LOOCV):",
↪ misclass_rate)

#El modelo `lda` estará entrenado con todos los datos
coefficients = svc.coef_[0]
normc=np.linalg.norm(coefficients)
print(coefficients/normc)
#print(coefficients)
b0=svc.intercept_[0]
print(b0)
```

Los siguientes códigos generan los histogramas de los conjuntos de datos que se trabajaron, simplemente hay que reemplazar el número *id* que indica el conjunto de datos con el que se está trabajando, además de las etiquetas. El códigos para graficar los histogramas de los valores $L(\boldsymbol{x})$ del DLF es el siguiente:

```
import numpy as np
from sklearn.discriminant_analysis import
↪  LinearDiscriminantAnalysis
from ucimlrepo import fetch_ucirepo
import matplotlib.pyplot as plt

# Obtener el conjunto de datos desde UCIML Repository
breast_cancer_wisconsin_diagnostic = fetch_ucirepo(id=17)

# Datos (como dataframes de pandas)
X0 = breast_cancer_wisconsin_diagnostic.data.features
y = breast_cancer_wisconsin_diagnostic.data.targets

# Verificar y reemplazar entradas iguales a 0 en X0
X1 = X0.replace(0, 0.001)

# Calcular el logaritmo de las entradas de X1
X = np.log(X1)

# Convertir y a numpy array
```

```
y = np.ravel(y)

# Inicializar el modelo LDA (Análisis Discriminante
↪ Lineal)
lda = LinearDiscriminantAnalysis()
# Después del bucle LOOCV, el modelo lda estará entrenado
↪ con todos los datos
lda.fit(X, y)
coefficients = lda.coef_[0]  # Obtener los coeficientes
b0 = lda.intercept_[0]

#print("Coeficientes:", coefficients)
#print("Intercepto:", b0)

# Calcular la función discriminante lineal de Fisher para
↪ cada observación
discriminant_function = np.dot(X, coefficients) + b0

# Clasificar las observaciones
predicted_class = np.where(discriminant_function > 0, 'M',
↪ 'B')

# Calcular la precisión de esta clasificación
#accuracy_fisher = accuracy_score(y, predicted_class)
#print("Precisión usando la función discriminante de
↪ Fisher:", accuracy_fisher)

# Graficar el histograma de los valores de la función
↪ discriminante
plt.figure(figsize=(10, 6))
plt.hist(discriminant_function[y == 'M'], color='orange',
↪ bins=30, alpha=0.5, label='Maligno')
plt.hist(discriminant_function[y == 'B'], bins=30,
↪ alpha=0.5, label='Benigno')
plt.axvline(0, color='red', linestyle='--', label='Umbral de
↪ decisión (0)')
plt.xlabel('Valores de la función de respuesta DLF')
plt.ylabel('Frecuencia')
plt.legend()
plt.xlim(-30, 30)
plt.ylim(-10, 35)
plt.title('Histogramas de la función de respuesta')
plt.show()
```

Para los histogramas de los valores de la función de respuesta del SVM simplemente agregamos lo siguiente:

```
# Inicializar el modelo SVM (Support Vector Machine)
svm_model = SVC(kernel='linear')  # Usando un kernel
↪  lineal

# Después del bucle LOOCV, el modelo svm_model estará
↪  entrenado con todos los datos
svm_model.fit(X, y)

# Obtener los coeficientes y el intercepto del hiperplano
↪  de decisión
coefficients = svm_model.coef_[0]
b0 = svm_model.intercept_[0]

#print("Coeficientes:", coefficients)
#print("Intercepto:", b0)

# Calcular la función de decisión para cada observación
decision_function = np.dot(X, coefficients) + b0

# Clasificar las observaciones
predicted_class = svm_model.predict(X)

# Calcular la precisión del modelo SVM
#accuracy_svm = accuracy_score(y, predicted_class)
#print("Precisión del modelo SVM:", accuracy_svm)

# Graficar el histograma de los valores de la función de
↪  decisión
plt.figure(figsize=(10, 6))
plt.hist(decision_function[y == 'B'], color='blue',
↪  bins=30, alpha=0.5, label='Benigno')
plt.hist(decision_function[y == 'M'], color='purple',
↪  bins=30, alpha=0.5, label= 'Maligno')
plt.axvline(0, color='red', linestyle='--', label='Umbral de
↪  decisión (0)')
plt.xlabel('Valores de la función de respuesta SVM')
plt.ylabel('Frecuencia')
plt.legend()
plt.xlim(-20, 20)
plt.ylim(-10, 50)
plt.title('Histogramas de la función de respuesta SVM')
plt.show()
```

B.2.2. Códigos de la Sección 3.2

Análisis de datos de correos electrónicos por DLF:

```
from ucimlrepo import fetch_ucirepo
from sklearn.discriminant_analysis import
↪  LinearDiscriminantAnalysis
from sklearn.model_selection import KFold
from sklearn.metrics import accuracy_score
from sklearn.metrics import confusion_matrix
import numpy as np
import pandas as pd

# Fetch dataset
spambase = fetch_ucirepo(id=94)

# Data (as pandas dataframes)
X = spambase.data.features
y = spambase.data.targets

# Inicializar el modelo LDA (Análisis Discriminante
↪  Lineal)
lda = LinearDiscriminantAnalysis()

# DLF usando todos los datos
y = np.ravel(y)

# Inicializar la validación cruzada K-Fold
kf = KFold(n_splits=10, random_state=0, shuffle=True)

# Listas para almacenar los valores verdaderos y
↪  predichos
y_true = []
y_pred = []

# Realizar la validación cruzada K-Fold
for train_index, test_index in kf.split(X):
X_train, X_test = X.iloc[train_index], X.iloc[test_index]
y_train, y_test = y[train_index], y[test_index]

# Entrenar el modelo LDA
lda.fit(X_train, y_train)

# Predecir las muestras de prueba
y_pred_fold = lda.predict(X_test)
y_pred.extend(y_pred_fold)
y_true.extend(y_test)
```

```
# Convertir a arreglos de numpy para análisis
y_true = np.array(y_true)
y_pred = np.array(y_pred)

# Calcular la matriz de confusión
conf_matrix = confusion_matrix(y_true, y_pred)

# Convertir la matriz de confusión a un DataFrame para
↪  mejor formato
conf_matrix_df = pd.DataFrame(
conf_matrix,
index=['Verdadero no-spam', 'Verdadero spam'],
columns=['Predicho no-spam', 'Predicho spam']
)

# Agregar totales por fila y por columna
conf_matrix_df['Total fila'] = conf_matrix_df.sum(axis=1)
conf_matrix_df.loc['Total columna'] =
↪  conf_matrix_df.sum(axis=0)

# Imprimir la matriz de confusión
print(conf_matrix_df)

# Calcular la proporción de clasificación errónea
accuracy = accuracy_score(y_true, y_pred)
misclass_rate = 1 - accuracy

print("Proporción de clasificación errónea (K-Fold):",
↪  misclass_rate)
```

Análisis de datos de correos electrónicos por SVM:

```
from ucimlrepo import fetch_ucirepo
from sklearn.svm import SVC
from sklearn.model_selection import KFold
from sklearn.metrics import accuracy_score
from sklearn.metrics import confusion_matrix
import numpy as np
import pandas as pd

# Fetch dataset
spambase = fetch_ucirepo(id=94)

# Data (as pandas dataframes)
X = spambase.data.features
```

```
y = spambase.data.targets

# Inicializar el modelo SVM
svc = SVC(kernel='linear')

# DLF usando todos los datos
y = np.ravel(y)

# Inicializar la validación cruzada K-Fold
kf = KFold(n_splits=10, random_state=0, shuffle=True)

# Listas para almacenar los valores verdaderos y
↪  predichos
y_true = []
y_pred = []

# Realizar la validación cruzada K-Fold
for train_index, test_index in kf.split(X):
X_train, X_test = X.iloc[train_index], X.iloc[test_index]
y_train, y_test = y[train_index], y[test_index]

# Entrenar el modelo LDA
svc.fit(X_train, y_train)

# Predecir las muestras de prueba
y_pred_fold = svc.predict(X_test)
y_pred.extend(y_pred_fold)
y_true.extend(y_test)

# Convertir a arreglos de numpy para análisis
y_true = np.array(y_true)
y_pred = np.array(y_pred)

# Calcular la matriz de confusión
conf_matrix = confusion_matrix(y_true, y_pred)

# Convertir la matriz de confusión a un DataFrame para
↪  mejor formato
conf_matrix_df = pd.DataFrame(
conf_matrix,
index=['Verdadero no-spam', 'Verdadero spam'],
columns=['Predicho no-spam', 'Predicho spam']
)

# Agregar totales por fila y por columna
conf_matrix_df['Total fila'] = conf_matrix_df.sum(axis=1)
```

```
conf_matrix_df.loc['Total columna'] =
↪  conf_matrix_df.sum(axis=0)

# Imprimir la matriz de confusión
print(conf_matrix_df)

# Calcular la proporción de clasificación errónea
accuracy = accuracy_score(y_true, y_pred)
misclass_rate = 1 - accuracy

print("Proporción de clasificación errónea (K-Fold):",
↪ misclass_rate)
```

B.2.3. Códigos de la Sección 3.3

Análisis de datos de Sonar por DLF:

```
from ucimlrepo import fetch_ucirepo
from sklearn.discriminant_analysis import
↪  LinearDiscriminantAnalysis
from sklearn.model_selection import LeaveOneOut
from sklearn.metrics import accuracy_score
from sklearn.metrics import confusion_matrix
import numpy as np
import pandas as pd

# fetch dataset
connectionist_bench_sonar_mines_vs_rocks =
↪  fetch_ucirepo(id=151)

# data (as pandas dataframes)
X = connectio-
↪  nist_bench_sonar_mines_vs_rocks.data.features
y = connectionist_bench_sonar_mines_vs_rocks.data.targets

# Inicializar el modelo LDA (Análisis Discriminante
↪  Lineal)
lda = LinearDiscriminantAnalysis()


# DLF usando todos los datos
y = np.ravel(y)
lda.fit(X, y)

# Inicializar la validación cruzada Leave-One-Out
loo = LeaveOneOut()
```

```
# Listas para almacenar los valores verdaderos y
↪  predichos
y_true = []
y_pred = []

# Realizar la validación cruzada Leave-One-Out
for train_index, test_index in loo.split(X):
X_train, X_test = X.iloc[train_index], X.iloc[test_index]
y_train, y_test = y[train_index], y[test_index]

# Entrenar el modelo LDA
lda.fit(X_train, y_train)

# Predecir la muestra de prueba
y_pred.append(lda.predict(X_test)[0])
y_true.append(y_test[0])

# Convertir a arreglos de numpy para análisis
y_true = np.array(y_true)
y_pred = np.array(y_pred)

# Calcular la matriz de confusión
conf_matrix = confusion_matrix(y_true, y_pred)

# Convertir la matriz de confusión a un DataFrame para
↪  mejor formato
conf_matrix_df = pd.DataFrame(
conf_matrix,
index=['Verdadero mina', 'Verdadero roca'],
columns=['Predicho mina', 'Predicho roca']
)

# Agregar totales por fila y por columna
conf_matrix_df['Total fila'] = conf_matrix_df.sum(axis=1)
conf_matrix_df.loc['Total columna'] =
↪  conf_matrix_df.sum(axis=0)

# Imprimir la matriz de confusión
print(conf_matrix_df)
# Calcular la proporción de clasificación errónea
accuracy = accuracy_score(y_true, y_pred)
misclass_rate = 1 - accuracy

print("Proporción de clasificación errónea (LOOCV):",
↪  misclass_rate)
```

Análisis de datos de Sonar por SVM:

```
import numpy as np
import pandas as pd
from sklearn.model_selection import LeaveOneOut
from sklearn.svm import SVC
from sklearn.metrics import accuracy_score
from sklearn.metrics import confusion_matrix
from ucimlrepo import fetch_ucirepo

# fetch dataset
connectionist_bench_sonar_mines_vs_rocks =
↪  fetch_ucirepo(id=151)

# data (as pandas dataframes)
X = connectio-
↪  nist_bench_sonar_mines_vs_rocks.data.features
y = connectionist_bench_sonar_mines_vs_rocks.data.targets

svc = SVC(kernel='linear')
# DLF usando todos los datos
y = np.ravel(y)
svc.fit(X, y)

# Inicializar Leave-One-Out Cross-Validation
loo = LeaveOneOut()

# Listas para almacenar los valores verdaderos y
↪  predichos
y_true = []
y_pred = []

# Realizar la validación cruzada Leave-One-Out
for train_index, test_index in loo.split(X):
X_train, X_test = X.iloc[train_index], X.iloc[test_index]
y_train, y_test = y[train_index], y[test_index]


# Inicializar y ajustar el modelo SVM
svc.fit(X_train, y_train)

# Predecir la muestra de prueba
y_pred.append(svc.predict(X_test)[0])
y_true.append(y_test[0])  # Usar y_test[0] en lugar de
↪  y_test
```

```
# Calcular la matriz de confusión
conf_matrix = confusion_matrix(y_true, y_pred)

# Convertir la matriz de confusión a un DataFrame para
↪  mejor formato
conf_matrix_df = pd.DataFrame(
conf_matrix,
index=['Verdadero mina', 'Verdadero roca'],
columns=['Predicho mina', 'Predicho roca']
)

# Agregar totales por fila y por columna
conf_matrix_df.loc['Total fila'] =
↪  conf_matrix_df.sum(axis=0)
conf_matrix_df['Total columna'] =
↪  conf_matrix_df.sum(axis=1)

# Imprimir la matriz de confusión
print("Matriz de confusión:")
print(conf_matrix_df)
# Calcular la proporción de clasificación errónea
accuracy = accuracy_score(y_true, y_pred)
misclass_rate = 1 - accuracy

print("Proporción de clasificación errónea (LOOCV):",
↪  misclass_rate)
```

Apéndice C

Alojamiento de la Tesis en el Repositorio Institucional

Título de Tesis:	Clasificación binaria lineal utilizando Python
Autor(a):	Luis Felipe López Guzmán
ORCID:	181A11003
Resumen de la Tesis:	Esta tesis proporciona una introducción a los conceptos de clasificación, DLF y SVM, y demuestra su aplicación en problemas de clasificación del mundo real. La comparación de los métodos ofrece información valiosa sobre su idoneidad para diferentes tipos de datos y tareas de clasificación.
Palabras claves de la Tesis:	Clasificación binaria, Aprendizaje automático, Reconocimiento de patrones, Discriminante Lineal de Fisher (DLF), Support Vector Machine (SVM).
Referencias citadas:	

Bibliografía

[1] Alexandre-Cortizo, E., Rosa-Zurera, M., Lopez-Ferreras F. (2005). *Application of Fisher linear discriminant analysis to speech/music classification*. EUROCON 2005 – The international conference on computer as a tool, Belgrade, Serbia, 1666–1669.

[2] Ash R. B., (1999). *Probability and Measure Theory*. 2nd ed., USA, Academic Press.

[3] Boser, B. E., Guyon, I. M., and Vapnik, V.N. (1992). *A training algorithm for optimal margin classifiers*. In: Proceedings of the fifth conference on computational learning theory (ed., D. Haussler), 144–152, New York: Association of Computing Machinery Press.

[4] Carmona Suárez E.J.,(2016). *Tutorial sobre Máquinas de Vectores Soporte*. `https://www.cartagena99.com`.

[5] Casella G., Berger L. R., (2001). *Statistical Inference*. 2nd ed., USA, Cengage Learning.

[6] Cortes, C., Vapnik, V. (1995). *Support-vector networks*. Machine learning, 20 (3): 273–297.

[7] Devroye L., Györfi L., Lugosi G., (1996). *A Probabilistic Theory of Pattern Recognition*. New York, Springer.

[8] Cristianini, N., Shawe-Taylor, J. (2000). *An introduction to support vector machines and other kernel-based learning methods*. Cambrigde: Cambridge University Press.

[9] Flowers R. J., Cruz S. H. D., López S. L., Pérez P. A., (2015). *Probabilidad*. 2da ed., Villahermosa, Tabasco: Universidad Juárez Autónoma de Tabasco.

[10] Friedberg S. H., Insel A. J., Spence L. E., (2002). *Linear Algebra*. 4th ed., USA, Pearson.

[11] Hernández H. F., (2003) *Teoría de conjuntos (una introducción)*. 2da ed., México, Sociedad Matemática Mexicana.

[12] Hastie, T., Tibshiran, R.i, Friedman, J. (2017). *The elements of statistical learning: data mining, inference, and prediction*. 2nd. Berlin: Springer.

[13] Izenman J. A., (2008). *Modern Multivariate Statistical Techniques: Regression, Classification, and Manifold Learning*. New York, Springer.

[14] Müller A. C., & Guido S. (2017). *Introduction to machine learning with Python: A guide for data scientists*. USA, O'Reilly Media.

[15] National Ocean Service (2024) *What is sonar?* https://oceanservice.noaa.gov/facts/sonar.html#:~:text=Active%20sonar%20transducers%20emit%20an,the%20strength%20of%20the%20signal

[16] Pedregosa, F., et al. (2011). *Scikit-learn: Machine learning in Python. Journal of machine learning research*, 12: 2825–2830.

[17] Pérez P. H., Nájera R. E., Cruz S. H. D., Flowers R. J., (2011). *Introducción Básica al Estudio del Análisis Matemático*. 1ra ed., Villahermosa, Tabasco: Universidad Juárez Autónoma de Tabasco.

[18] Raschka, S., Patterson, J., & Nolet, C. (2020). *Machine learning in Python: main developments and technology trends in data science, machine learning, and artificial intelligence*. Information, 11(4), 193.

[19] Rincón L. (2014). *Introducción a la probabilidad*. 1ra ed., México, D.F.: Universidad Nacional Autónoma de México, Facultad de Ciencias.

[20] Rossum, G. V. (20 January 2009). *The history of Python: a prief timeline of Python*. Disponible en https://python-history.blogspot.com/2009/01/brief-timeline-of-python.html.

[21] Scikit-learn. (2024). *Getting Started*. https://scikit-learn.org/stable/getting_started.html.

[22] Sejnowski, T., Gorman, R. (1988). *Connectionist Bench (Sonar, Mines vs. Rocks) [Dataset]*. UCI Machine Learning Repository. https://doi.org/10.24432/C5T01Q.

[23] Wackerly D. D., Mendenhall III W. y Scheaffer R.L., (2008). *Mathematical statistics with applications*. 7th ed., USA. Cengage Learning.

[24] Yu, W., Liu, T., Valdez, R. et al. (2010). *Application of support vector machine modeling for prediction of common diseases: the case of diabetes and pre-diabetes*. BMC medical informatics and decision making, 10, https://doi.org/10.1186/1472-6947-10-16.