

Máster en Ingeniería Industrial
2021-2022

Trabajo Fin de Máster

Detección y reconocimiento de imagen
con aplicación ferroviaria para la
validación de equipos industriales y
operación automática

Antonio Rodríguez Alhambra

Tutor/es

M. Isabel Herreros Cid

Miguel López Hernández

Lugar y fecha de presentación prevista

DETECCIÓN DEL PLAGIO

La Universidad utiliza el programa **Turnitin Feedback Studio** para comparar la originalidad del trabajo entregado por cada estudiante con millones de recursos electrónicos y detecta aquellas partes del texto copiadas y pegadas. Copiar o plagiar en un TFM es considerado una **Falta Grave**, y puede conllevar la expulsión definitiva de la Universidad.



[Incluir en el caso del interés en su publicación en el archivo abierto]

Esta obra se encuentra sujeta a la licencia Creative Commons **Reconocimiento - No Comercial - Sin Obra Derivada**

RESUMEN

Palabras clave:

DEDICATORIA

ÍNDICE GENERAL

1. INTRODUCCIÓN	1
2. OBJETIVOS	3
3. ESTADO DEL ARTE	4
3.1. Inteligencia Artificial	4
3.1.1. <i>Deep Learning</i>	4
3.1.2. <i>Machine learning</i>	5
3.1.3. Implementación	6
4. MARCO TEÓRICO	8
4.1. Extracción de características	8
4.1.1. Momentos de imagen	9
4.1.2. Histograma de gradientes	11
4.1.3. Patrones locales binarios	13
4.1.4. Descriptores de Fourier	15
4.2. Selección de características	17
4.2.1. ANOVA	17
4.2.2. SFS	19
4.3. Métodos de evaluación	20
4.3.1. Matriz de confusión	20
4.3.2. Curva ROC	24
4.3.3. Validación cruzada	26
4.4. Modelos de clasificación	27
4.4.1. K-vecinos más cercanos	27
4.4.2. SVM	29
5. METODOLOGÍA	38
6. RESULTADOS	41
6.1. Problema desbalanceado	41
6.2. ANOVA	43
6.2.1. Área B345	44

6.2.2. Área C1C9	45
6.3. K-vecinos más cercanos	46
6.3.1. Resultados con clasificador Knn para área B345	46
6.3.2. Resultados con clasificador Knn para área C1C9	53
6.4. SVM.	57
6.4.1. SVM con área C1c9	57
6.4.2. SVM con área B345	61
7. CRONOGRAMA DE ACTIVIDADES	65
BIBLIOGRAFÍA	67
7.1. Creación de dataset y extracción de características	

ÍNDICE DE FIGURAS

4.1	Esquema método LBP	13
4.2	Descriptores de Fourier	15
4.3	16
4.4	Distribuciones para un caso ejemplo de ANOVA	17
4.5	Sacado de wikipedia (EDITAR)	25
4.6	División de dataset en entrenamiento y validación	26
4.7	Validación cruzada de 4 particiones	27
4.8	Ejemplo clasificación Knn con datos bidimensionales (caso balanceado) .	28
4.9	Ejemplo de SVM	30
4.10	Recta óptima SVM	31
4.11	Hiperplanos adicionales SVM	32
4.12	Aquí debe ir una imagen sin referencia	35
5.1	Procesos en la fase de extracción y selección de características	38
5.2	Selección y análisis del clasificador	40
6.1	Nº de muestras para el dataset del área B345	41
6.2	Nº de muestras para el dataset del área C1C9	42
6.3	Resultados ANOVA para área B345	44
6.4	Resultados ANOVA para área C1C9	46
6.5	Curva de validación para modelo KNN uniforme, F1 macro	47
6.6	Curva de validación para modelo KNN uniforme, Recall macro	47
6.7	Curva de validación para modelo KNN uniforme, Precisión macro	48
6.8	Curva de validación para modelo KNN uniforme en rango n=[5,20], Precisión macro	49
6.9	Curva de validación para modelo KNN uniforme en rango n=[5,20], Recall macro	49
6.10	Curva de validación para modelo KNN uniforme en rango n=[5,20], F1 macro	50
6.11	Curva de validación para modelo KNN uniforme, F1 macro	53

6.12	Curva de validación para modelo KNN uniforme, Recall macro	54
6.13	Curva de validación para modelo KNN uniforme, Precisión macro	54
6.14	Curva de validación para modelo KNN uniforme, F1 macro	55
6.15	Curvas de validación para kernels de modelo SVC en área C1c9 en función de parámetro de penalización C.	57
6.16	Curva de validación para kernel RBF en función de γ en área C1c9.	58
6.17	Curvas de validación para kernel polinómico	59
6.18	Curvas de validación para kernels de modelo SVC en área B345 en función de parámetro de penalización C.	61
6.19	Curva de validación para kernel RBF en función de γ en área B345.	62
6.20	Curvas de validación para kernel polinómico	62
7.1	Procesos en la fase de extracción y selección de características	66

ÍNDICE DE TABLAS

4.1	Histograma de gradientes, tabla ejemplo	11
4.2	Matriz de confusión binaria	20
4.3	Matriz de confusión multiclas	21
4.4	Ejemplo matriz de confusión para métricas ponderadas	22
4.5	Ejemplo métricas para caso desbalanceado	23
4.6	Ejemplo métricas promedio	23
4.7	Ejemplo distancias clasificación knn	29
6.1	N° de muestras para área B345	42
6.2	N° de muestras para área C1C9	43
6.3	Resultados ANOVA para área B345	44
6.4	Resultados ANOVA para área C1C9	45
6.5	Mejores parámetros modelo Knn para área B345	50
6.6	Métricas por clase y promedios para área B345 con modelo Knn	51
6.7	Métricas promediadas Knn B345	51
6.8	Mejores parámetros modelo Knn para área C1C9	55
6.9	Métricas por clase y promedios para área C1C9 con modelo Knn	56
6.10	Métricas promediadas Knn C1C9	56
6.11	Resultados AUC para diferentes Kernel SVM para área C1C9	60
6.12	AUC macro para C1C9	60
6.13	Resultados AUC para diferentes Kernel SVM para área B345	64
6.14	AUC macro para B345	64

1. INTRODUCCIÓN

QUEDA POR ESCRIBIR:

- Resumen (lo último)
- Introducción:
 - Motivación del trabajo
 - Hablar del problema del HMI, de porqué se quiere implementar un sistema que reconozca los símbolos en tiempo real de la pantalla por áreas
 - Qué áreas son las que componen el HMI.
- Objetivos: aunque estén escritos, poner muchos más de los que hablar en las conclusiones sobre si se han cumplido o no.
- Estado del arte
 - Está hecha una introducción a la inteligencia artificial, distinguiendo entre machine learning y deep learning (quizás añadir alguna imagen).
 - Está hecho el porqué de elegir Python, pero queda poner porqué se ha usado Machine Learning y porqué Scikit-learn (EN DETALLE)
 - Hablar de la aplicación de machine learning en el ámbito ferroviario, porqué es importante y donde se aplica (pruebas de laboratorio, pruebas de ensayo, etc) (aquí necesito la ayuda de Miguel para que me informe)
- Discusión de los resultados: necesito el visto bueno de Miguel sobre la primera versión de los resultados. Si no me lo puede dar hago una versión inicial de este apartado.
- Impacto socio-económico: calcular (inventado, a ver quién sabe los números exactos) el coste del proyecto.
- Impacto social: importancia del transporte ferroviario, importancia de las pruebas de ensayo e importancia de automatizar en el sector ferroviario (necesaría una pequeña guía de Miguel sobre a dónde enfocar esto)
- Conclusiones finales (lo penúltimo antes del resumen)
- Futuras líneas del trabajo (pedir ideas a Miguel)
 - Mejoras del trabajo: principalmente hablar de obtener mejores imágenes, probar todos los posibles modelos de aprendizaje supervisado una vez se tenga un dataset fiel al que se vaya a utilizar de verdad.

- Ampliación del estudio: lo que pone encima de probar con más modelos, decir cuales podrían ir (porqué) y cuáles no (porqué no) (los árboles de decisiones que sobre ajustan que da gusto).
- Problemas sin resolver: que yo sepa que distinga el color, el medidor de velocidad y las áreas que no sean símbolos sin más.

2. OBJETIVOS

El principal objetivo de este Trabajo de Fin de Máster es encontrar un algoritmo de clasificación multiclas con un alto rendimiento de discriminación entre las clases de cada área del HMI.

Métodos de extracción de características Investigar e implementar métodos de obtención de datos, sobre las imágenes a discriminar, que obtengan información que permita al clasificador separar entre clases.

Sencillez en la ejecución Desarrollar un proceso de preparado de datos, selección y clasificación que no requiera de procesos extensivos y computacionalmente pesados para lograr el mejor resultado con la mayor simplicidad posible. Además, que no requiera de grandes conocimientos en el área de *machine learning*, de tal forma que el proceso pueda ser modificado o expandido en un futuro sin requerir formación en el campo.

Aprendizaje del campo Aún cuando la formación del autor es en ingeniería industrial, electrónica y automática, y el área de *machine learning* queda fuera de las enseñanzas de esta disciplina, un objetivo de este trabajo es una puesta en contacto con un área de investigación y desarrollo cada vez más en auge en el mundo actual, debido a su increíble potencial en casi cualquier área que requiera de clasificación o regresión de datos.

Realización del Trabajo para finalización del curso Completar satisfactoriamente el proyecto para su presentación en el año de graduación y terminación del Máster.

Nota: AÚN por terminar.....

3. ESTADO DEL ARTE

3.1. Inteligencia Artificial

Para entrar en el campo de la inteligencia artificial, ya sea para definirlo, conocer sus múltiples ramas, estudiarlo, etc., es imprescindible nombrar a uno de sus más importantes creadores, John McCarthy.

Informático y científico cognitivo, John McCarthy definió la AI en 2004 como **la ciencia y la ingeniería de hacer máquinas inteligentes, especialmente programas informáticos inteligentes**[1]. Sin embargo, muchos años antes ya se empezaba a hablar de "máquinas inteligentes" con grandes figuras como Alan Turing y su famoso "¿Puede pensar una máquina?"(del original en inglés *Can a machine think?*), donde aparece por primera vez el famoso Test de Turing, en el cual un interrogador humano trata de discriminar si a lo que se está enfrentando es otro humano o una máquina.

A *grosso modo*, el campo de la inteligencia artificial es un área de las matemáticas enfocado en la resolución y optimización de problemas.

Originalmente, los investigadores de I.A. trataban de replicar el funcionamiento del cerebro humano, pero de tal es la complejidad es la tarea, que actualmente la I.A. se centra en el desarrollo de modelos para problemas específicos que no necesitan "pensar" necesariamente como lo haría un humano.

Hoy en día se han desarrollado dos ramas de la inteligencia artificial, conocidas comúnmente en lo público por sus nombres en inglés *machine learning* y *deep learning*. Estas disciplinas son, realmente, algoritmos y métodos matemáticos de inteligencia artificial capaces de hacer predicciones sobre datos discretos (clasificación) o continuos (regresión).

Aún cuando son usadas indiscriminadamente para referirse a modelos de clasificación y regresión, el campo del aprendizaje profundo (*deep learning*) es, realmente, un campo del *machine learning*.

3.1.1. Deep Learning

El aprendizaje profundo, a diferencia de *machine learning*, no requiere de un preprocesado de datos, con preparación, selección, etc. Los datos pueden introducirse directamente en su forma primitiva en el modelo y este es el encargado de encontrar los patrones y características para discriminar.

3.1.2. Machine learning

El nombre *Machine learning* fue acuñado en primera instancia por Arthur Samuel, en referencia al juego de las damas, como "Programming computers to learn from experience should eventually eliminate the need for much of this detailed programming effort[2].

Desde que Arthur Samuel escribiera aquel artículo en 1959 hasta hoy, ha habido innumerables avances tecnológicos que han permitido una mejora del campo de la inteligencia artificial. Las mejoras de almacenamiento con dispositivos capaces de almacenar descomunales cantidades de datos, procesadores con procesamiento paralelo en nubes y centros de datos, tarjetas gráficas para la manipulación matricial, etc.

Machine learning requiere la intervención humana para la preparación de un dataset procesado, estandarizado, normalizado, etc, el cual introducir en el modelo para que este, según las características escogidas por los usuarios, encuentre un estado en el cual el modelo discrimine.

Los algoritmos de *Machine learning* pueden clasificarse en dos grandes grupos:

Aprendizaje supervisado

Del inglés *supervised learning*, se caracteriza por el uso de datos con referencias a las clases a las que pertenecen. El modelo es entrenado según unos datos de entrenamiento y sus etiquetas de correspondencia a su clase, modificando los parámetros del modelo matemático para obtener la mejor clasificación posible. Debido a que son evaluados con datos diferentes respecto con los que se entrena, existe la posibilidad de que sobreajusten su discriminación y no generalicen lo suficiente los datos de entrenamiento, resultando en una mala clasificación para cualquier otro dato que no sea de ese grupo; o puede que subajusten, donde generalizarían demasiado en los datos de entrenamiento sin recoger información relevante para la discriminación de otros datos.

Muchos son los modelos de clasificación supervisada, desde los modelos SVM de clasificación binaria basados en hiperplanos de n dimensiones, desarrollado por el equipo de Vladimir Vapnik en los laboratorios Bell de AT&T durante los primeros años de la década de 1990; como el modelo Knn desarrollado por Evelyn Fix y Joseph Hodges en 1951; el clasificador de Naive Bayes (ó Clasificador bayesiano ingenuo) basado en el teorema de Bayes; o los famosos árboles de decisión desarrollados también en los laboratorios Bell de AT&T por Tin Kam Ho y más tarde la implementación algorítmica por Leo Breiman y Adele Cutler.

Aprendizaje no supervisado

Del inglés *unsupervised learning*, a diferencia del supervisado los datos de clasificación no cuentan con un etiquetado de la clase a la que pertenecen, sino que es el modelo

el encargado de encontrar las fronteras que separan a los grupos de datos para determinar su pertenencia. Aunque no es una categoría en si de *machine learning*, suele utilizarse para agrupar todos aquellos métodos que extraen información de datos sin pre procesar. Son altamente usados para encontrar patrones .ºcultos.º grupos de datos sin necesidad de interpretación humana. Una parte de estos modelos son las famosas redes neuronales, cuyos datos son introducidos de forma primitiva y es el propio modelo el encargado de encontrar las características y diferentes grupos a los que estos datos puedan pertenecer.

Entre los muchos modelos, cabe destacar el modelo PCA (Principal Component Analysis) usado ampliamente para la reducción de características mediante la transformación de una dimensión a otra del espacio inicial de los datos.

3.1.3. Implementación

Cuando se habla de la implementación de un modelo de inteligencia artificial, múltiples son los lenguajes de programación que entran en juego: C, C++, Python, R, MATLAB, etc. Sin embargo, de entre todos estos, es Python la elección a nivel industrial. Aunque es importante considerar que las funciones y demás métodos están escritas en C o C++, estas al final son llamadas con Python, la herramienta que el usuario maneja.

Python es conocido por ser un lenguaje de alto nivel con una sintaxis que se asemeja a la forma de hablar y pensar de una persona, haciéndolo un lenguaje ideal para principiantes con barreras de entradas de muy baja dificultad. Además, ser un proyecto de código abierto permite a los desarrolladores acomodar y personalizar Python a sus necesidades y, en el mundo de la inteligencia artificial donde los problemas crecen y evolucionan, poder disponer de una poderosa herramienta adaptable es de extrema ayuda para un rápido y eficaz desarrollo. Igualmente, el código abierto implica que cualquier usuario puede "jugar con Python descubriendo y resolviendo bugs, ayudando al mantenimiento y mejora del lenguaje.

Sin embargo, la principal razón por la que Python es uno de los lenguajes de referencia en inteligencia artificial se debe a Google.

Cuando Google se fundó en 1998, cuatro años se había fundido su mayor competidor en la época, Yahoo. Por aquel entonces, ambas empresas buscaban un lenguaje de programación de código abierto y de muy alto nivel, ya que el tiempo de desarrollo era (y es) muchísimo más importante que el tiempo de procesado. Google apostó por Python y Yahoo por Perl.

El ganador, Google, decidió invertir y apostar por Python, hasta tal punto de servicios como su famoso motor de búsqueda o Youtube usen el lenguaje, así como todos sus departamentos de inteligencia artificial y de robótica[3].

Otro importante aspecto es la cantidad de librerías escritas para Python. Los dos mayores exponentes de este caso son las famosas Numpy y Scipy. Scikit-learn (una librería de machine learning) está escrita específicamente para Python (en C y C++) sobre Numpy, al

igual que PyTorch y Keras (librerías de deep learning). Python es famoso por su extrema facilidad a la hora de importar librerías e instalarlas en el entorno de trabajo.

No obstante, Python es un lenguaje dinámico interpretado, que lo hace, por definición, lento. Pero la facilidad de escribir un código prototipo de forma rápida y que funcione es ventajoso frente a otros lenguajes más rápidos pero de menor nivel (C, C++) ya que el tiempo que el desarrollador emplear en pensar e implementar la solución a un problema es mucho más importante que el tiempo de ejecución del código). Por otro lado, a la hora de desarrollar un prototipo o probar posibles soluciones, es un error enfocarse en una optimización prematura[4].

Los modelos de *machine learning* son, realmente, modelos matemáticos con reglas condicionantes, probabilísticas, lógicas, etc, que requieren de la aplicación de una optimización para obtener los mejores valores de dicho modelo (veáse, por ejemplo, la condición de optimización de un modelo SVM en 4.33).

Dado que los conocimientos necesarios para la resolución de estos problemas son muy elevados, existen librerías de programación que implementan clases y funciones, en las cuales sólo es necesario modificar parámetros, que devuelven los modelos entrenados y listos para su uso.

Algunos ejemplos, para Python, son Sklearn, Pytorch, OpenCV y Keras, entre muchos otros.

4. MARCO TEÓRICO.

Nota: Este capítulo sirve como una breve introducción a todos los métodos, conceptos y modelos considerados para el desarrollo de este trabajo.

Todo proceso de clasificación de imágenes puede dividirse en los siguientes pasos:

- Tratado inicial de las imágenes
- Extracción de características
- Tratamiento de datos
- Entrenamiento de modelo de clasificación
- Evaluación modelo de clasificación

4.1. Extracción de características

En este paso el objetivo es obtener información cuantitativa (numérica) de las muestras a partir de diversos métodos.

Por ejemplo, de una clasificación de números escritos a mano se pueden obtener características como el número de trazos, anchura del contorno, color, etc. Es decir, se obtienen una serie de propiedades informativas de la muestra para, después, entrenar el modelo de clasificación.

Para este trabajo, se han utilizado los siguientes métodos:

- Descriptores de Fourier
- Momentos de imagen (Momentos Hu)
- Descriptores locales binarios (*Local Binary Patterns*)
- Histograma de gradientes
- *Método propio

4.1.1. Momentos de imagen.

Los momentos de imagen son un promedio de las intensidades de una imagen binaria. La convención habitual define un momento M para una imagen binaria B de la siguiente forma:

$$M_{ij} = \sum_x \sum_y x^i y^j B(x, y) \quad (4.1)$$

Utilizando la ecuación 4.1, se pueden obtener características la imagen como los centroides:

$$\bar{x} = \frac{M_{10}}{M_{00}} \quad (4.2)$$

$$\bar{y} = \frac{M_{01}}{M_{00}} \quad (4.3)$$

Sin embargo, aplicando una transformación a la ecuación 4.1, se pueden obtener momentos invariantes a la traslación:

$$\mu_{i,j} = \sum_x \sum_y (x - \bar{x})^i (y - \bar{y})^j I(x, y) \quad (4.4)$$

A la ecuación 4.4 se la conoce como **momentos centrales**.

Además, aplicando otra transformación se pueden obtener momentos invariantes al escalado también:

$$\eta_{i,j} = \frac{\mu_{i,j}}{\frac{i+j}{2} + 1} \quad (4.5)$$

La fórmula 4.5 se conoce como **momento centralizado**.

Momentos de Hu

Los momentos de Hu [5] son un conjunto de siete fórmulas obtenidas a partir de los momentos centralizados que permiten obtener siete momentos invariantes tanto a traslación, rotación, escalado y volteado (el séptimo momento es el invariante a volteado, cambiando de signo cuando la imagen es reflejada).

$$\begin{aligned}
M_1 &= \eta_{20} + \eta_{02} \\
M_2 &= (\eta_{20} - \eta_{02})^2 + 4\eta_{11}^2 \\
M_3 &= (\eta_{30} - 3\eta_{12})^2 + (3\eta_{21} - \eta_{03})^2 \\
M_4 &= (\eta_{30} + \eta_{12})^2 + (\eta_{21} + \eta_{03})^2 \\
M_5 &= (\eta_{30} - 3\eta_{12})(\eta_{30} + \eta_{12})[(\eta_{30} + \eta_{12})^2 - 3(\eta_{21} + 3\eta_{03})^2] \\
&\quad + (3\eta_{21} - \eta_{03})(\eta_{21} + \eta_{03})[3(\eta_{30} + \eta_{12})^2 - (\eta_{21} + \eta_{03})^2] \\
M_6 &= (\eta_{20} - \eta_{02})[(\eta_{30} + \eta_{12})^2 - (\eta_{21} + \eta_{03})^2] \\
&\quad + 4\eta_{11}(\eta_{30} + \eta_{12})(\eta_{21} + \eta_{03}) \\
M_7 &= (3\eta_{21} - \eta_{03})(\eta_{30} + \eta_{12})[(\eta_{30} + \eta_{12})^2 - 3(\eta_{21} + 3\eta_{03})^2] \\
&\quad - (\eta_{30} - 3\eta_{12})(\eta_{21} + \eta_{03})[3(\eta_{30} + \eta_{12})^2 - (\eta_{21} + \eta_{03})^2]
\end{aligned} \tag{4.6}$$

A partir de 4.6, se pueden obtener siete características numéricas.

4.1.2. Histograma de gradientes

El histograma de gradientes [6] es una técnica de extracción de características muy utilizada en la detección de objetos.

A partir de una imagen binaria de dimensiones $n \times m$, se calculan los gradientes de intensidad, así como la magnitud y el ángulo:

$$\begin{aligned} G_x &= B(x+1, y) - B(x, y) \\ G_y &= B(x, y+1) - B(x, y) \end{aligned} \quad (4.7)$$

La ecuación 4.7 determina los gradientes de la imagen binaria B . Tanto G_x como G_y son dos imágenes binarias con la información de gradientes en ejes X e Y, respectivamente. A partir de la ecuación 4.7, se obtienen las magnitudes y ángulos:

$$Mag_{(x,y)}(\mu) = \sqrt{G_x^2 + G_y^2} \quad (4.8)$$

$$Ang_{(x,y)}(\theta) = |\tan^{-1}\left(\frac{G_y}{G_x}\right)| \quad (4.9)$$

Tanto la ecuación 4.8 como 4.9 representan imágenes binarias.

El siguiente paso consiste en dividir las imágenes de magnitud y ángulos en N cuadrículas, pudiendo ser $N = 1$ (una única cuadrícula).

Para cada cuadrícula se representa un histograma de 9 posiciones, con cada posición en el rango $[\theta, \theta + \delta\theta]$. Convencionalmente, se suele utilizar $\delta\theta = 20^\circ$. De esta forma se obtiene el siguiente histograma H_N :

Magnitud								
Ángulo	0	20	40	60	80	100	120	140

TABLA 4.1. HISTOGRAMA DE GRADIENTES, TABLA EJEMPLO

A cada intervalo angular y de magnitud, se le denomina θ_j y μ_j , respectivamente, para $j \in [0, 8]$.

De tal forma, si en la cuadrícula N_i existe un $\theta_{x,y}$ que pertenece al rango $[\theta, \theta + \delta\theta \cdot j]$, para $j \in [0, 8]$, se determina que $\mu_j = \mu_{x,y} + \mu_j$.

$$\mu_j = \sum \mu_{x,y} \iff \theta_{x,y} \in [\theta, \theta + \delta\theta \cdot j] \quad (4.10)$$

Existen casos en los que $\theta_{x,y} > 160^\circ$, entonces $\mu_{x,y}$ contribuye tanto a 0° como a 160° .

$$\begin{aligned} \frac{180 - \theta_{x,y}}{20} \cdot \mu_{x,y} &\implies [160, 180) \\ \left(1 - \frac{180 - \theta_{x,y}}{20}\right) \cdot \mu_{x,y} &\implies [0, 20) \end{aligned} \tag{4.11}$$

El último paso consiste en normalizar el histograma:

$$\mu_j = \frac{\mu_j}{\max(\mu_j)} \tag{4.12}$$

Tras la aplicación de 4.12, se tienen $9 \cdot N$ características, siendo N el número de cuadrículas.

4.1.3. Patrones locales binarios

También conocido por sus siglas del inglés **LBP** (del inglés, *Local Binary Patterns*), este método propuesto en la década de 1990 [7] permite extraer un histograma de 256 características de una muestra.

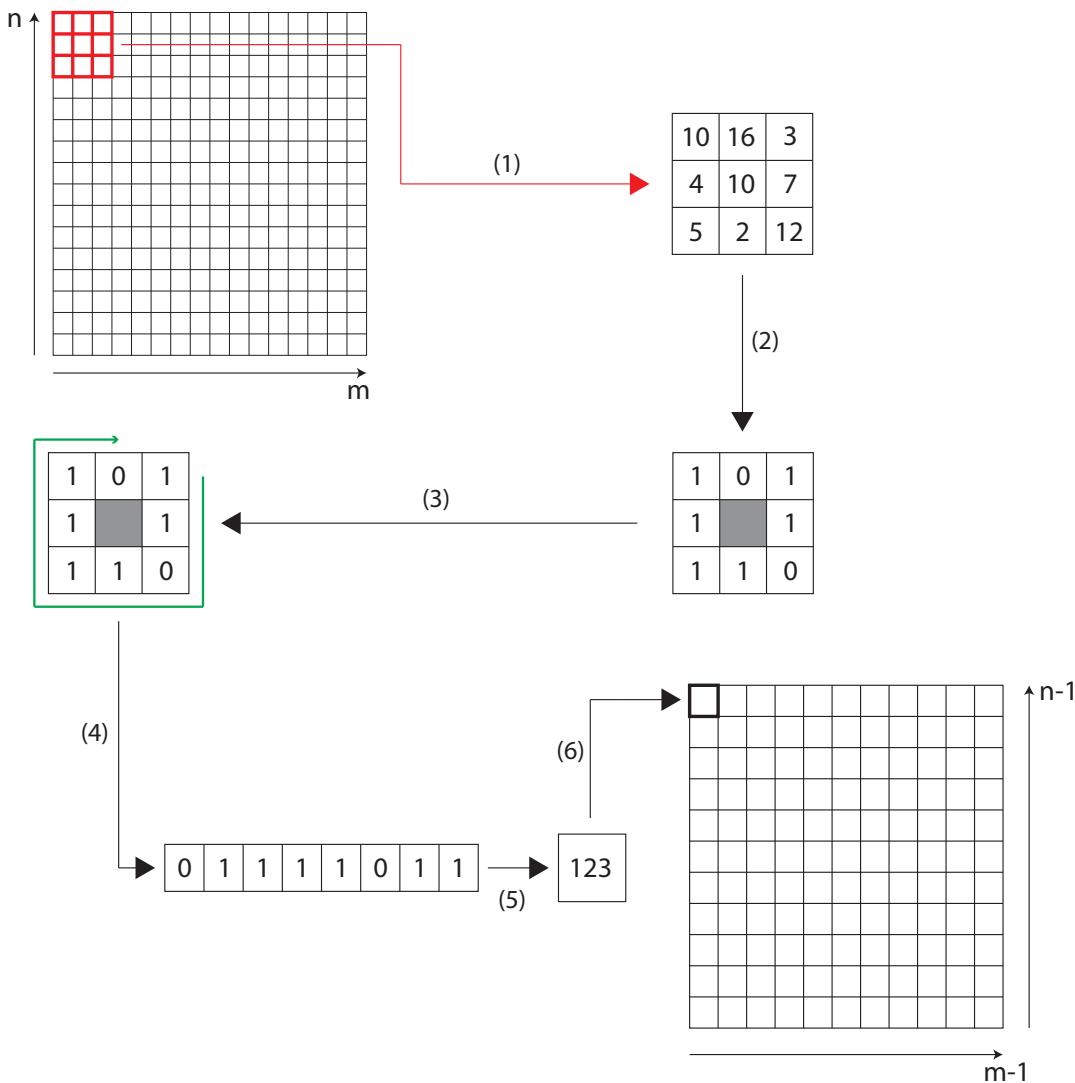


Fig. 4.1. Esquema método LBP

La figura 4.1 muestra un esquematizado resumen del proceso de extracción de características por este método.

La imagen original es tratada como una matriz de dimensiones $n(\text{filas}) \times m(\text{columnas})$. A partir de esta, se extraen submatrices de dimensiones 3×3 y se comienza el análisis de datos.

En la nueva submatriz, la posición central es comparada con las exteriores. Si, para una posición exterior, el valor central es menor o igual, se asigna a esta posición un valor 1. De ser mayor, se asigna un 0.

La nueva matriz binaria es tratada como un número binario de 8 dígitos, consistiendo el siguiente paso en obtener el equivalente en base diez. Este último valor es almacenado en una nueva matriz de dimensiones $(n - 1)x(m - 1)$.

Realizando este proceso para toda la imagen original se tiene un conjunto de datos a partir de los cuales obtener el histograma sobre el que se basa el método **LBP**.

Teóricamente, el número máximo de características extraíbles es 256 (ver 4.13), pero pueden agruparse por rangos obteniendo grupos de características. Esta última opción es dependiente de la aplicación y del problema a tratar.

$$[00000000, 11111111]_2 \rightarrow [0, 255]_{10} \quad (4.13)$$

4.1.4. Descriptores de Fourier

Los descriptores de Fourier son un método de extracción de características por el cual un objeto bidimensional, cuyos puntos tienen las coordenadas (x_k, y_k) , es mapeado a un dominio complejo de la forma (x_k, iy_k) . De tal forma, el objeto es tratado como una señal discreta compleja a la cual se aplica la transformada discreta de Fourier para encontrar sus armónicos (descriptores de Fourier).

La figura 4.2a representa una forma hexagonal rellena. Para obtener los descriptores de Fourier de este objeto, es necesario obtener el contorno exterior de la figura (véase 4.2b).

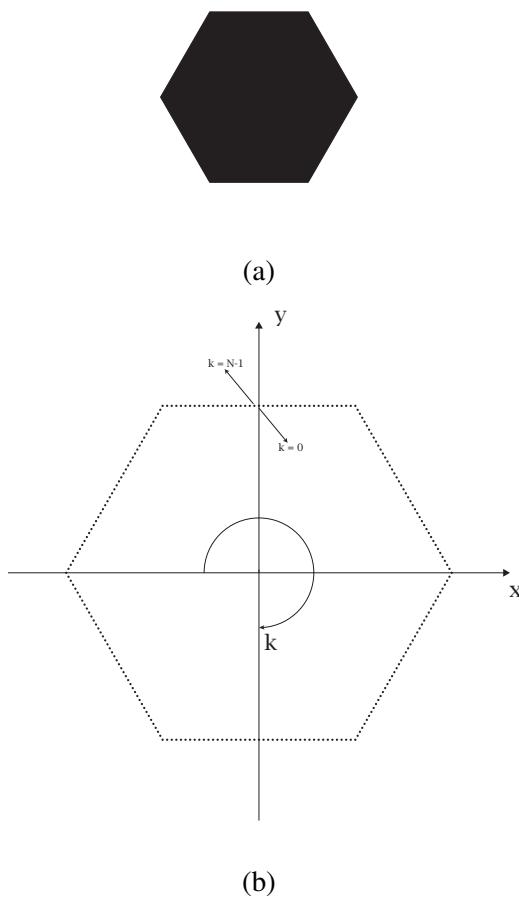


Fig. 4.2. Descriptores de Fourier

Este nuevo contorno está representado por N puntos, de tal forma que cada punto puede definirse de la forma (x_k, y_k) , con $k = 0, 1, 2, \dots, N-1$ y el contorno, a su vez, como la señal $f_k = (x_k, y_k)$.

Si f_k es transformada al dominio complejo de la forma $f_k = x_k + iy_k$, es posible aplicar la transformada discreta de Fourier a f_k para obtener los N armónicos de la señal (descriptores de Fourier):

$$F_m = \sum_{k=0}^{N-1} f_k \cdot e^{-\frac{2\pi i}{N} mk}, \quad m = 0, 1, 2, \dots, N-1 \quad (4.14)$$

La ecuación 4.14 (transformada discreta de Fourier), proporciona la serie de valores $F_0, F_1, F_2, \dots, F_{N-1}$, a partir de los cuales se obtienen los descriptores de Fourier (valores absolutos) $|F_0, F_1, F_2, \dots, F_{N-1}|$.

En función de la resolución de la imagen original y su definición, el número de puntos del contorno N variará, obteniéndose una cantidad diferente para cada caso. Por ello, el número de descriptores de Fourier obtenibles de cada imagen no será el mismo. Es por esto que, generalmente, de los N descriptores obtenidos para cada imagen, se escogen solo $n \mid 0 \leq n \leq N$.

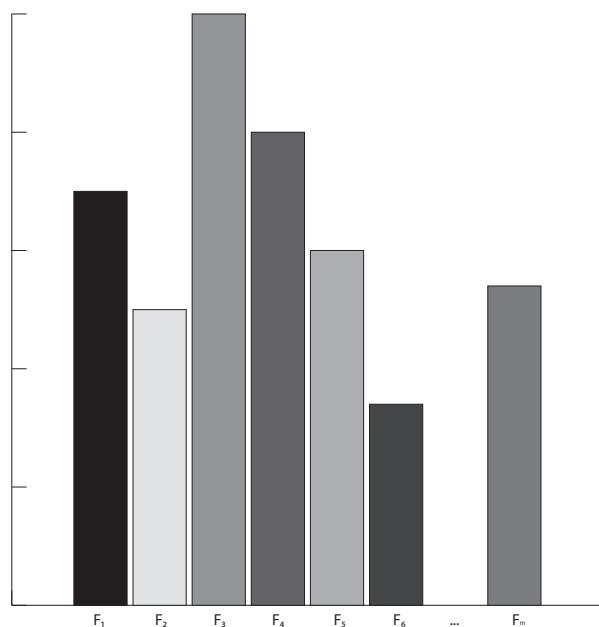


Fig. 4.3

4.2. Selección de características

Una vez se han aplicado los métodos propuestos en la sección 4.1, es necesario realizar un análisis del poder clasificatorio de las características extraídas.

Para este trabajo, se han utilizado los siguientes métodos:

- ANOVA
- SFS

4.2.1. ANOVA

Nota: ANOVA es un conjunto de métodos estadísticos, entre los cuales está el F-Test, que es el que se usa aquí, principalmente.

Nota: Casi toda la información está sacada de [8].

Conocido como Análisis de la Varianza (del inglés, *ANalysis Of VAriance*).

Dado un ejemplo como una distribución de datos de dos clases (véase figura 4.4), se disponen de dos características de clasificación: (x, y).

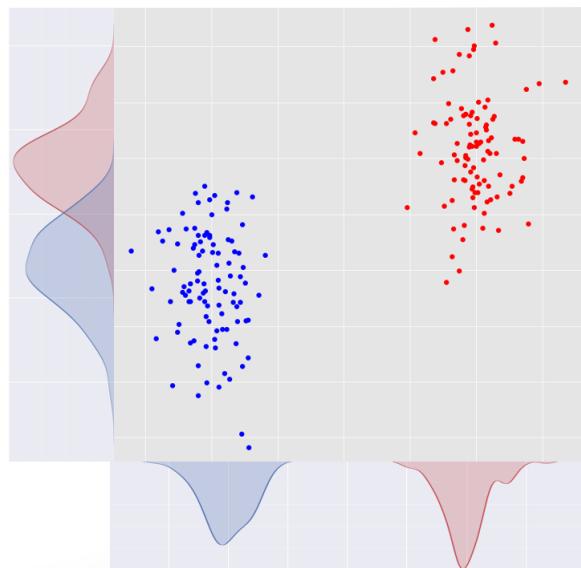


Fig. 4.4. Distribuciones para un caso ejemplo de ANOVA

La característica x es mucho mejor que la y a simple vista pues, comparando las distribuciones de ambas, para x se obtienen dos distribuciones claramente separadas por completo, mientras que para la característica y las distribuciones se solapan.

También se puede comprobar que la distribución para la característica x presenta una menor varianza (son más compactas) que con y .

Por tanto, para este ejemplo se puede definir la condición 4.15 como un parámetro de discriminación de características. Cuanto mayor sea para una característica, mejor será su poder de clasificación.

$$F = \frac{\text{Distancia entre clases}}{\text{Compacidad de clases}} \quad (4.15)$$

Matemáticamente, la fórmula 4.15 puede expresarse de la siguiente forma siguiendo el método ANOVA:

- El numerador (distancia entre clases) es definible con:

$$n_{azul}(\overline{x_{azul}} - \bar{x})^2 + n_{roja}(\overline{x_{roja}} - \bar{x})^2 \quad (4.16)$$

- El denominador (compacidad de clases), que no es sino la varianza de clases, puede expresarse como:

$$\left(\frac{1}{(n_{azul} - 1) + (n_{roja} - 1)} \right) \left(\sum_{i=1}^{n_{azul}} (x_i - \overline{x_{azul}})^2 + \sum_{i=1}^{n_{roja}} (x_i - \overline{x_{roja}})^2 \right) \quad (4.17)$$

De tal forma, la ecuación 4.15 queda como:

$$F = \frac{n_{azul}(\overline{x_{azul}} - \bar{x})^2 + n_{roja}(\overline{x_{roja}} - \bar{x})^2}{\left(\frac{1}{(n_{azul} - 1) + (n_{roja} - 1)} \right) \left(\sum_{i=1}^{n_{azul}} (x_i - \overline{x_{azul}})^2 + \sum_{i=1}^{n_{roja}} (x_i - \overline{x_{roja}})^2 \right)} \quad (4.18)$$

La ecuación 4.18 da un parámetro para una variable (en este caso x). En un problema con k características, se obtendrían k valores de F , es decir, una F para cada variable.

Utilizando las k características extraídas con los métodos propuestos en 4.1, ANOVA proporcionaría F_k valores, de entre los cuales se seleccionarían los N con mayor valor F . La ecuación 4.19 define el valor de F para un caso con $j = 1, 2, 3, \dots, N$ clases.

$$F = \frac{\sum_{j=1}^N n_j (\overline{x_j} - \bar{x})^2}{\left(\frac{1}{\sum_{j=1}^N (n_j - 1)} \right) \left(\sum_{j=1}^N \left(\sum_{i=1}^{n_j} (x_{ij} - \overline{x_j})^2 \right) \right)} \quad (4.19)$$

Nota:

Este método solo da información sobre lo bien que UNA variable discrimina, pero no dice como de bien lo harían varias juntas.

4.2.2. SFS

Los algoritmos SFS (del inglés *Sequential Feature Selector*) son un conjunto de técnicas de selección de características (algoritmos voraces) usados para reducir un espacio inicial n-dimensional de características a otro k-dimensional donde $k \leq d$.

Si se tiene el siguiente conjunto n-dimensional $Y = \{y_1, y_2, y_3, \dots, y_n\}$ de características de entrada, utilizando un estimador determinado (SVM, Knn, SDG, etc.) y una métrica de rendimiento determinada, el algoritmo SFS busca obtener un conjunto de salida $X_k = \{x_j | j = 1, 2, \dots, d; x_j \in Y\}$, con $k = (0, 1, 2, \dots, n)$ tal que X_k esté formado por las k características que maximicen la métrica.

Existen dos formas de realizar este proceso: hacia adelante (*Forward*) y hacia detrás (*Backwards*).

- Sequential Forward Selection: se comienza con un conjunto vacío $X_0 = \emptyset/k = 0$. En cada iteración se aumenta el número de características hasta encontrar la combinación que de el mejor resultado (según la métrica a resolver).
- Sequential Backward Selection: se comienza con el conjunto inicial Y . En cada iteración se disminuye el número de características hasta encontrar la combinación que de el mejor resultado (según la métrica a resolver).

Es posible no llegar al mismo resultado empleando sentidos contrarios y tampoco obtener el mismo resultado, en varias iteraciones, utilizando el mismo método.

Nota:

En *Sklearn* la métrica para su modelo SFS es una evaluación de validación cruzada con un estimador definido en la llamada al método SFS.

4.3. Métodos de evaluación

Antes de entrar en la sección de modelos de clasificación, es necesario definir las métricas propuestas utilizadas para analizar los resultados de los clasificadores.

En una primera instancia, es necesario presentar la Matriz de confusión binaria.

4.3.1. Matriz de confusión

		Valor real	
		1	0
Valor predicho	1	True Positive (TP)	False Positive (FP)
	0	False Negative (FN)	True Negative (TN)

TABLA 4.2. MATRIZ DE CONFUSIÓN BINARIA

La matriz de confusión es una herramienta muy utilizada para representar y ver de forma sencilla los resultados de una clasificación. En el caso de la tabla 4.2, se representa una matriz de confusión para un caso de clasificación binaria, sin embargo puede utilizarse para casos multiclasses.

A partir de la matriz de confusión binaria se obtienen las métricas de rendimiento de evaluación de un clasificador: exactitud, precisión, exhaustividad, f1, etc.

$$\text{Exactitud (ACC)} = \frac{TP + TN}{TP + TN + FP + FN} \quad (4.20)$$

$$\text{Exhaustividad (TPR)} = \frac{TP}{P} = \frac{TP}{TP + FN} \quad (4.21)$$

$$\text{Precision} = \frac{TP}{TP + FP} \quad (4.22)$$

$$F_{\beta} = (1 + \beta^2) \frac{2 \cdot \text{Precision} \cdot \text{Exhaustividad}}{\text{Precision} + \text{Exhaustividad}} \quad (4.23)$$

- Exactitud: (en inglés *accuracy*) (ecuación 4.20) es una medida de la proporción de predicciones correctas (tanto TP como TN) del total de casos a predecir.
- Precisión: (ecuación 4.22) medida del porcentaje de valores positivos acertados (TP) de entre todos los valores positivos predichos (TP+FP). La precisión muestra la robustez del clasificador para acertar en los resultados positivos. Sin embargo, no tiene en cuenta como de bueno es acertando las clases negativas.

- Exhaustividad: (en inglés *recall*) (ecuación 4.21) medida del porcentaje de las muestras positivas predichas correctamente como positivas (TP) entre el numero total de muestras positivas reales (TP+FN). La exhaustividad es una medida de la capacidad del clasificador para detectar muestras positivas. A mayor *recall*, mas muestras positivas (correctas e incorrectas) detecta. Sin embargo, es una métrica que no tiene para nada en cuenta la clasificación de muestras negativas reales (FP y TN).
- F-score: es una media armónica de la exhaustividad y la precisión. En función del parámetro β , se le da peso a la exhaustividad. A mayor valor de β , mas importante el *recall*. Es muy común utilizar la métrica para $\beta = 1$, es decir, *F1-score*, misma valoración tanto a la precisión como a la exhaustividad. Sin embargo, la métrica F_1 no considera los negativos verdaderos (TN) (ni 4.21, ni 4.22 usan TN en sus cálculos) y para problemas con clases desbalanceadas es erróneo usar esta métrica, prefiriendo el uso del coeficiente de Kappa[9].

Exactitud con problemas desbalanceados Si se supone un ejemplo donde se tiene un dataset de 10000 muestras de personas enfermas con gripe o coronavirus, del cual 9990 muestras son de personas con gripe, el clasificador podría asignar ciegamente a cada nueva muestra la condición de gripe y sería exacto en un 99,9 % de las veces. Lo que implica que la métrica de exactitud no es correcta para casos desbalanceados.

Matriz de confusión para problemas multiclasses

Para problemas con más de 2 clases, la matriz de confusión 4.2 se convierte en 4.3, donde se representa el número de muestras de una clase que han sido identificadas como de otra clase o de si misma. Además de ser simétrica respecto de la diagonal.

Clase	1	2	...	N
1	X			
2		X		
...			X	
N				X

TABLA 4.3. MATRIZ DE CONFUSIÓN MULTICLASE

Para este caso, las variables la matriz de confusión binaria cambian de definición. Las predicciones correctas de cada clase con si misma (marcadas con X en la tabla 4.3) corresponden a los verdaderos positivos (TP); el sumatorio de los valores de cada fila, excepto el valor X, corresponde a los falsos positivos (FP); para una clase N , el sumatorio de todas las predicciones erróneas de las demás clases que predijeron pertenecer a la clase N es el falso negativo (FN) de la N ; y para una clase N_i el sumatorio de los verdaderos

Clase	F1	Precisión	Recall
1	F_{1_1}	P_1	P_1
2	F_{1_2}	P_2	P_2
3	F_{1_3}	P_3	P_3

positivos (TP) del resto de clases es TN (verdadero negativo) para la clase N_i . (Véase Conversión de problema multiclas a binario).

Las métricas de exactitud, precisión, exhaustividad y F_1 , para este caso, son aplicables en cuanto a cada clase se refiere. Para N clases se obtienen N métricas, sin embargo, es deseable obtener un solo valor, de cada métrica, que represente a todas las clases. Para ello, es necesario hacer un promediado 'micro', 'macro' o ponderado.

Promediado 'micro', 'macro' y ponderado.

- Macro: consiste en obtener la media aritmética entre clases. Da un mismo peso a todas las clases.

$$P = \frac{P_i + P_{i+1} + P_{i+2} + \dots + P_N}{N} \quad (4.24)$$

- Micro: consiste en aplicar la fórmula de cada métrica con el sumatorio total de variables de cada clase.

$$P = \frac{\sum_{i=1}^N TP_i}{\sum_{i=1}^N (TP_i + FP_i)} \quad (4.25)$$

- Ponderado: consiste en aplicar la frecuencia de cada clase en el dataset total al sumatorio de métricas.

$$P = \sum_{i=1}^N P_i f_i; \text{ donde } f_i = \frac{N_i}{N} \quad (4.26)$$

Tómese por ejemplo un caso de clasificación desbalanceada de tres clases según la matriz de confusión 4.4 y las métricas de cada clase 4.5.

Clase	0	1	2
0	5331	1921	748
1	1014	357	129
2	344	122	34

TABLA 4.4. EJEMPLO MATRIZ DE CONFUSIÓN PARA MÉTRICAS PONDERADAS

Clase	Precisión	Recall	F1	Nº de muestras
0	0.8	0.67	0.73	8000
1	0.15	0.24	0.18	1500
2	0.04	0.07	0.05	500

TABLA 4.5. EJEMPLO MÉTRICAS PARA CASO
DESBALANCEADO

La tabla 4.6 muestra las métricas promedio:

Clase	Precisión	Recall	F1	Nº de muestras
Micro	0.5722	0.4856	0.4433	8000
Macro	0.3276	0.3241	0.3190	1500
Ponderada	0.6617	0.8322	0.611	500

TABLA 4.6. EJEMPLO MÉTRICAS PROMEDIO

Para cualquiera de las tres métricas, Macro < Micro < Ponderada. Tomando como referencia la tabla 4.5, para la precisión (por ejemplo), la clase cero tiene un valor muy elevado respecto a las otras dos, incluso se puede decir que el clasificador es nefasto con las clases dos y tres. Cuando se calcula la precisión macro, se realiza un simple media entre las precisiones, dándole el mismo peso ó importancia a cada clase, por tanto, los valores reducidos de las clases dos y tres contribuyen a bajar mucho el valor de la precisión total del problema, aún cuando para la clase cero es muy elevada. Para el caso ponderado, como a cada precisión se le multiplica por su frecuencia en el dataset, la clase cero contribuye a aumentar mucho el valor pues, tiene 8000 muestras ella sola frente a las 2000 de las clases una y dos. La precisión micro obtiene un valor intermedio ya que tiene en cuenta los falsos positivos de cada clase, aumentando la influencia de la clase más frecuente ya que sus falsos positivos serán mayores al tener más muestras.

Por tanto, en general, para casos desbalanceados, si se considera a cada clase igual de importante que las demás, es conveniente utilizar un promedio macro. Si, por el contrario, es la clase más frecuente la más importante, el promedio ponderado es el preferible. Si se desea obtener un resultado que favorezca a la muestra más frecuente sin considerarla la más importante, un promedio micro es considerable.

No siempre ocurre que la micro es mayor que la macro pues, depende de los datos. Si la clase más frecuente tiene la métrica más pequeña con las demás siendo muy elevadas, es probable que la micro resulte de menor magnitud que la macro (a diferencia del ejemplo presentado), ya que la micro favorece a la clase más frecuente, sin llegar al punto de la ponderación.

Ya que, al final, depende del caso, es imprescindible contar con algo más de información (matriz de confusión, por ejemplo) para poder interpretar correctamente los datos.

Conversión de problema multiclase a binario Es posible convertir un problema de clasificación multiclase a un conjunto de problemas binarios utilizando los métodos Clasificación Uno contra Todos o Clasificación Uno contra Uno.

4.3.2. Curva ROC

Lo habitual, además de deseable, a la hora de obtener los valores predichos de un clasificador, es hacerlo en forma de estimaciones probabilísticas en el rango [0, 1]. De tal forma que, cuando se obtenga un valor, se puede comprobar con qué nivel de confianza el clasificador asigna a qué clase la muestra a predecir.

Muchas librerías de algoritmos supervisados devuelven los resultados, por defecto, como valores enteros 0 o 1, aplicando un umbral de clasificación de 0,5. En el caso de obtener 0,5000001 para la clase positiva y 0,4999999 para la negativa, el clasificador asignaría automáticamente la muestra a la clase positiva, obviando el hecho de que realmente no existe un nivel de confianza suficiente para clasificar la muestra de tal forma.

Al obtenerse probabilidades y no valores discretos, para asignar clases, es necesario establecer un umbral de clasificación por el cual si $p(n) \geq \text{umbral} \rightarrow n = 1$. Por tanto, es conveniente encontrar un valor para el umbral de clasificación que haga que el clasificador funcione lo mejor posible.

Para cada valor de umbral se obtiene una matriz de confusión diferente, así como sus pertinentes métricas. Puede que exista un valor de umbral en el que el clasificador, para la base de datos dada, sea idóneo o puede que no existe ningún valor de umbral que haga que el clasificador discrimine correctamente. Sin embargo, es posible que, independientemente del umbral, el clasificador obtenga buenos resultados.

Para poder analizar estos casos, se recurre a la curva ROC (del inglés *Receiver operating characteristic*).

La curva ROC es un gráfico utilizado para determinar la habilidad discriminante de un clasificador binario según su umbral de clasificación es variado.

Este gráfico se basa en el espacio ROC, que es básicamente la representación de la tasa de verdaderos positivos (o exhaustividad) (4.21) frente a la tasa de falsos positivos (4.27).

Nota:

Exhaustividad: cantidad de valores reales positivos predichos como positivos.

$$\text{Tasa de falsos positivos (FPR)} = \frac{FP}{N} = \frac{FP}{FP + TN} \quad (4.27)$$

La línea de puntos roja representa un clasificador puramente aleatorio que asigna un

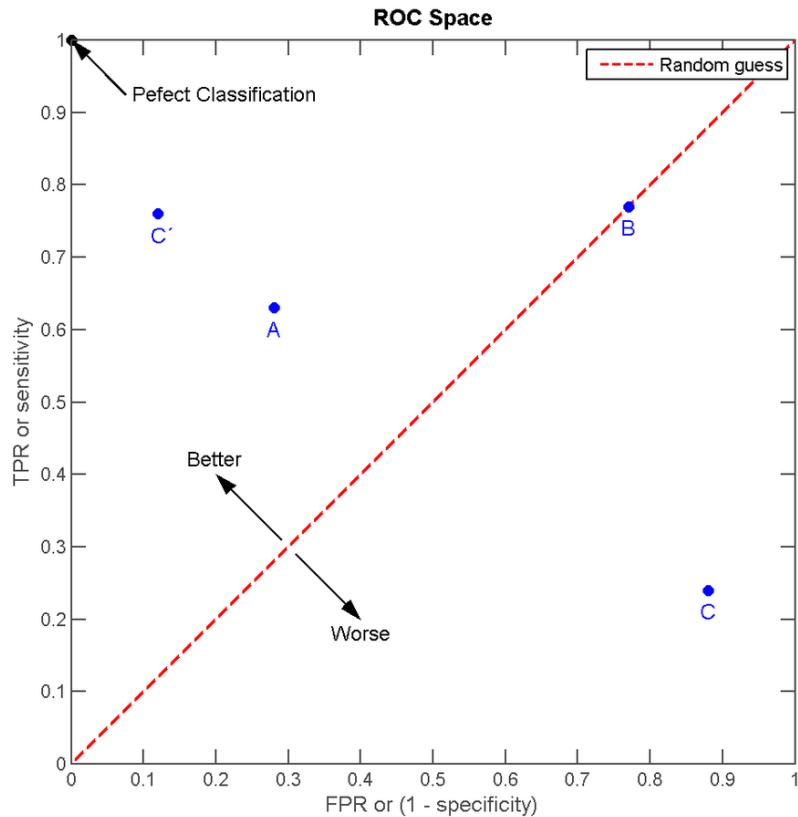


Fig. 4.5. Sacado de wikipedia (EDITAR)

0,5 de probabilidades de pertenecer a la clase positiva a cada muestra.

Para valores de umbral en el rango [0, 1], se obtienen los valores de 4.21 y 4.27, y se representa el punto correspondiente en el gráfico. De tal forma, se obtiene una curva escalonada que representa la calidad del clasificador.

El caso idóneo es aquel en el que la curva del clasificador es un escalón unidad, es decir, aquel clasificador que, para cualquier valor de umbral, se obtiene un 100 % de exhaustividad. Numéricamente, este tipo de curva corresponde a una curva con una área bajo la curva de la unidad.

El peor caso es el clasificador cuya curva sea la inversa al escalón unidad pues, se obtiene un 0 % de exhaustividad para cualquier umbral, es decir, aquella curva con una área bajo la curva nula.

Por tanto, se define el parámetro **AUC**, del inglés *Area Under the Curve*, que determina la calidad discriminante del clasificador. Cuanto mayor sea el valor de AUC, mejor el clasificador.

Sin embargo, la curva ROC no es correcta para casos desbalanceados ya que no representa correctamente los resultados, siendo necesario sustituirla por una curva de precisión-

exhaustividad [10].

Clasificación multiclas La curva ROC y sus derivadas sólo son utilizables para problemas binarios. Para casos con más de dos clases, se requiere una Clasificación Uno contra Todos, obteniendo tantas curvas ROC como clases, así como un valor AUC para cada clase, que representaría la calidad del clasificador de discriminar esa clase respecto a las demás; o Clasificación Uno contra Uno, obteniendo $\frac{N(N - 1)}{2}$ curvas ROC y valores AUC.

4.3.3. Validación cruzada

Todo proceso de clasificación necesita de dos conjuntos de datos para entrenamiento y validación de resultados.

Lo más simple es dividir el dataset de partida en dos particiones: entrenamiento y validación.

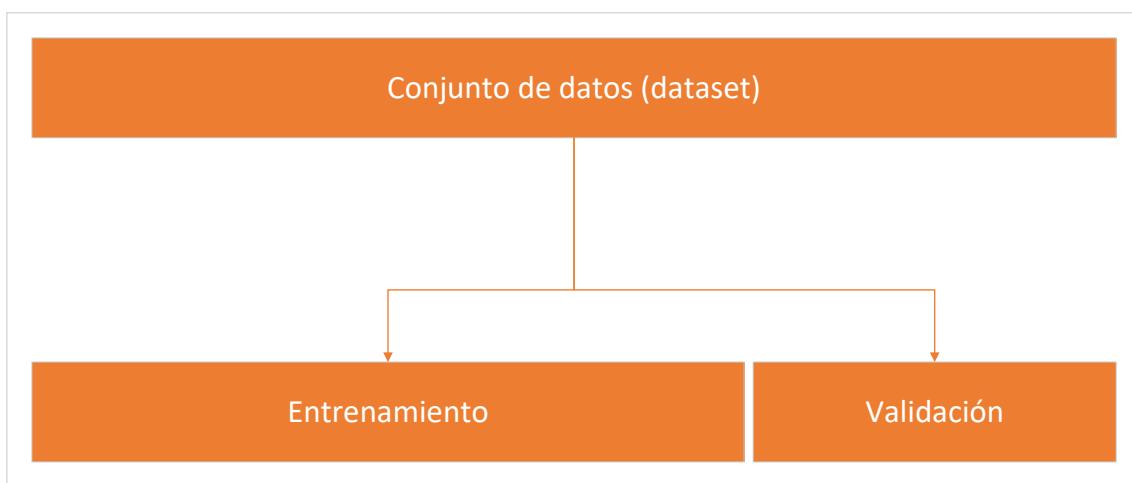


Fig. 4.6. División de dataset en entrenamiento y validación

Sin embargo, cabe la posibilidad de que al dividir el dataset, los datos de la partición de entrenamiento sean diferentes a la de validación. Es decir, puede que los datos de entrenamiento no sean suficientemente generalistas (atípicos) y al validar el modelo con los datos de validación, se obtengan resultados muy pobres. Para evitar esto se recurre a la validación cruzada.

Del inglés *Cross-Validation*, la validación cruzada es un método de análisis de resultados en modelos de clasificación que garantiza la independencia de los resultados frente a la división de datos de entrenamiento y validación.

El dataset inicial es dividido en N particiones (además, es de buenas prácticas mezclar los datos entre particiones) y cada partición es dividida en N partes. $N - 1$ partes de cada

partición son empleadas para entrenar el modelo y la parte restante para validación. Este proceso se repite para todas las particiones, obteniendo N resultados.

La figura 4.7 representa una validación cruzada de cuatro particiones.

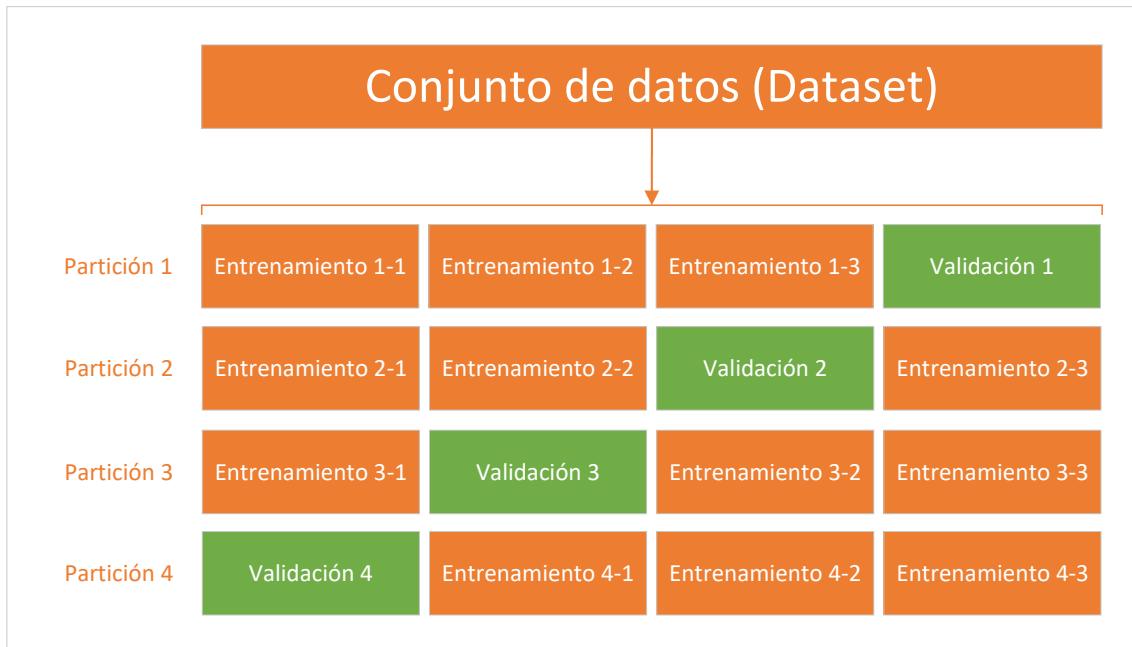


Fig. 4.7. Validación cruzada de 4 particiones

Con los N resultados se realizan análisis estadísticos (principalmente media y desviación típica) para obtener métricas independientes del punto de división de la base de datos.

4.4. Modelos de clasificación

Esta sección presenta, teóricamente, los modelos propuestos para la resolución del trabajo. Aunque existen muchos otros tipos, solo se tratarán los comentados a continuación.

Nota: Los siguientes modelos se caracterizan por ser de aprendizaje supervisado. Los datos de entrenamiento son presentados en pares, con características y etiquetas de identificación.

4.4.1. K-vecinos más cercanos

K-vecinos más cercanos, abreviado como Knn, (del inglés, *K-nearest neighbors*), es un modelo de clasificación supervisado basado en la idea de clasificar una muestra en función de los puntos más cercanos a él (vecinos más cercanos), con la muestra siendo asignada en función de la clase más repetida entre los k puntos menos distantes.

Knn soporta tanto clasificación multiclase como binaria.

Suponiendo que se tiene un conjunto de datos de la forma $(X_1, Y_1), (X_2, Y_2), \dots, (X_n, Y_n)$ siendo Y_n la etiqueta de la clase a la que pertenece la muestra X_n , con X_n siendo un conjunto d-dimensional de d características extraídas de la forma (x_1, x_2, \dots, x_d) y una muestra a clasificar M de la misma forma que X_n , el algoritmo calcula la distancia Minkowski (véase ec. 4.28) de la muestra M a todas las n muestras de X obteniéndose n distancias, de las cuales, la clase con más frecuencia entre las k distancias menores es asignada a la muestra M .

$$D(X_n, M) = \left(\sum_{i=1}^d |X_{ni} - M_i|^p \right)^{\frac{1}{p}}, \quad p \geq 1 \quad (4.28)$$

Es común que el parámetro p de 4.28 valga 1 (distancia Manhattan) o 2 (distancia Euclídea).

La figura 4.8 ilustra el procedimiento de un algoritmo knn en la clasificación de una muestra M respecto a tres clases balanceadas. A su vez, la tabla 4.7 muestra, ordenadas de menor a mayor, las distancias de la muestra a los puntos de cada clase.

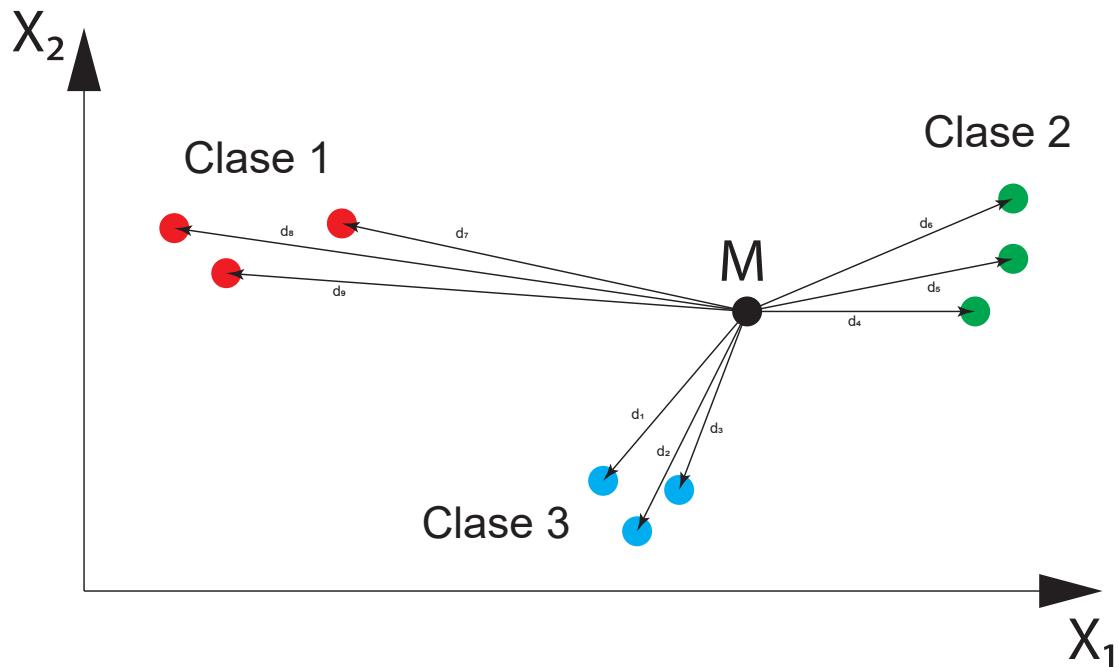


Fig. 4.8. Ejemplo clasificación Knn con datos bidimensionales (caso balanceado)

Distancia	Clase
d_1	3
d_2	3
d_4	2
d_3	3
d_5	2
d_6	2
d_7	1
d_8	1
d_9	1

TABLA 4.7. EJEMPLO DISTANCIAS CLASIFICACIÓN KNN

En función del parámetro k , la muestra M será asignada a la clase más repetida de entre las k menores distancias en la tabla 4.7. De ser $k = 4$, $M \rightarrow$ Clase 3 pues, de las 4 mínimas distancias, la clase 3 tiene la mayor frecuencia.

Centroide más cercano Existe una variación al método knn que se basa en calcular la distancia de la muestra M a los centroides de cada clase. Es decir, cada clase se representa como el centroide de todos sus puntos, a partir del cual se calcula la distancia con la muestra objetivo y se escoge la clase con la mínima distancia. La ventaja de este método es que no requiere ningún parámetro, sólo especificar el tipo de distancia a utilizar, sin embargo, no es una buena opción cuando las clases son no-convexas así como si presentan varianzas muy diferentes.

Knn con datos desbalanceados El modelo knn no sufre con datos desbalanceados, sin embargo es posible implementar un peso por muestra invirtiendo su distancia, de tal forma que, a mayor distancia de la muestra, menor influencia en el resultado.

4.4.2. SVM

Del inglés, *Support Vector Machines*, SVM es un modelo de clasificación binaria supervisada que se basa en la aplicación de hiperplanos para separar las muestras de dos clases.

Suponiendo que se tiene un conjunto de datos de la forma $(X_1, Y_1), (X_2, Y_2), \dots, (X_n, Y_n)$ siendo Y_n la etiqueta de la clase a la que pertenece la muestra X_n de la forma $(-1, 1)$, con X_n siendo un conjunto d-dimensional de d características extraídas de la forma (x_1, x_2, \dots, x_d) y una muestra a clasificar S de la misma forma que X_n , el modelo SVM busca un hiperplano que separe los puntos X_n con $Y_n = -1$ de los puntos con $Y_n = 1$.

Para ilustrar, de forma sencilla, el funcionamiento de los modelos SVM, se suponen dos clases de datos, $(1, -1)$, en un espacio de características bidimensional.

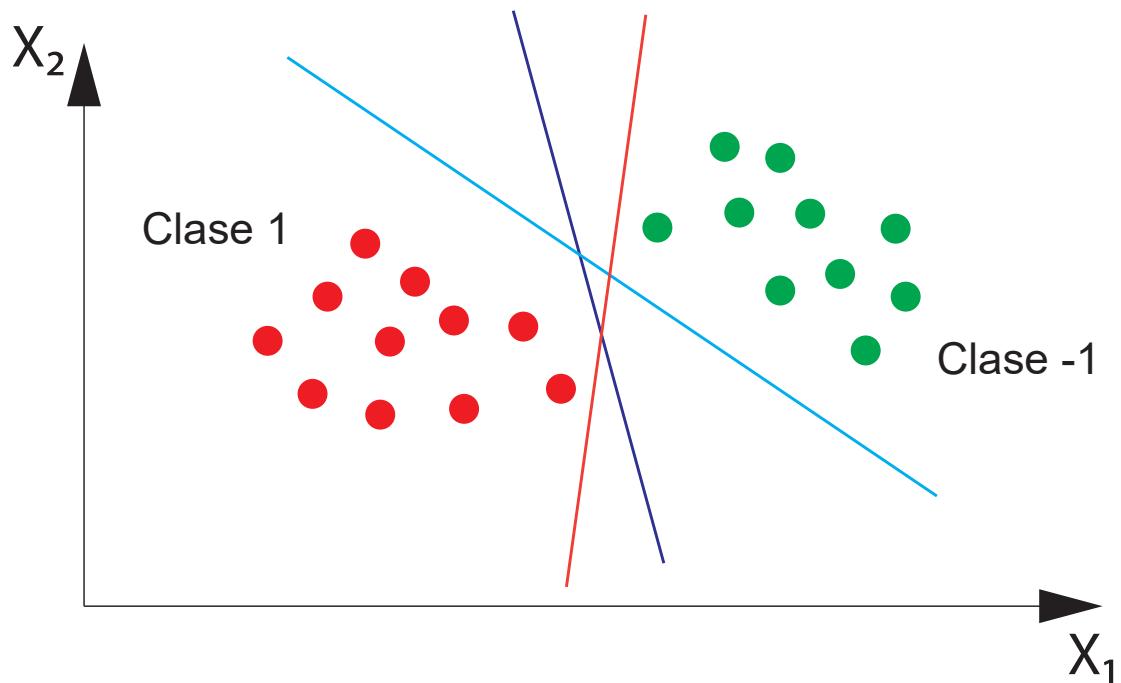


Fig. 4.9. Ejemplo de SVM

La figura 4.9 muestra posibles hiperplanos (en dos dimensiones, un hiperplano es una recta) para clasificar a las dos clases. Cualquiera de las tres rectas mostradas es correcta, sin embargo, la mejor solución posible es aquella recta que maximice la distancia (máximo margen) a ambas clases (veáse figura 4.10).

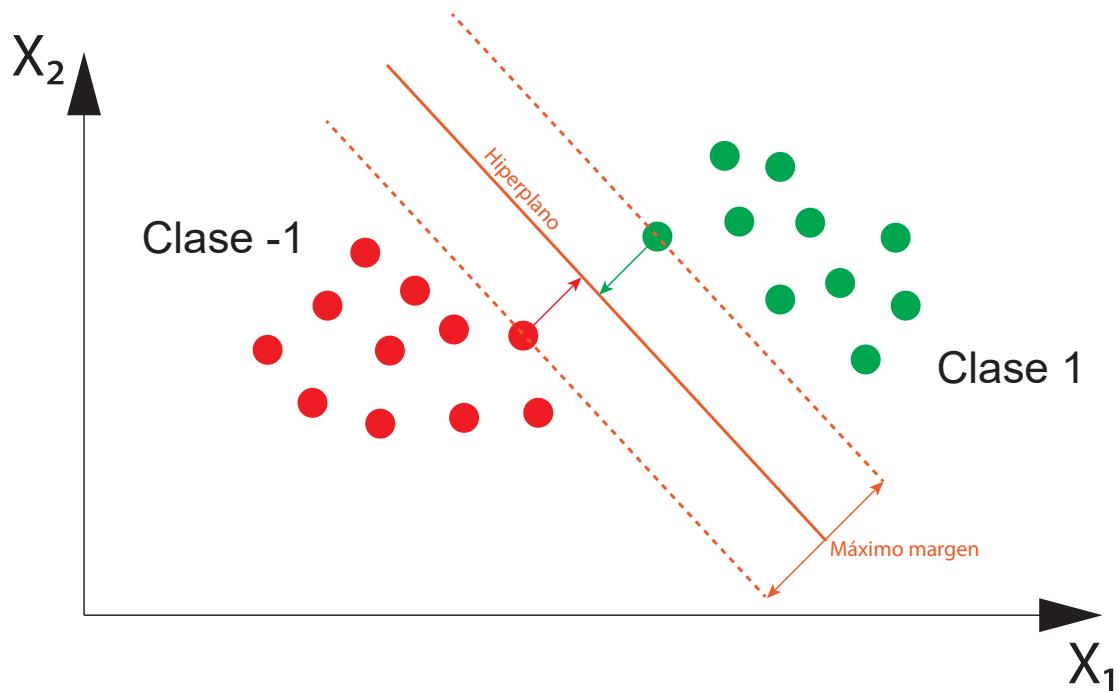


Fig. 4.10. Recta óptima SVM

Matemáticamente, un hiperplano es definible como $w^T x - b = 0$, siendo w el vector normal al hiperplano y $\frac{b}{\|w\|}$ el desplazamiento del hiperplano respecto al origen en la dirección de w .

Existen dos posibles casos según las características de los datos:

Datos linealmente separables

En este caso, es posible definir dos hiperplanos adicionales de la forma $w^T x - b = 1$ (hiperplano superior al principal) y $w^T x - b = -1$ (hiperplano inferior al principal).

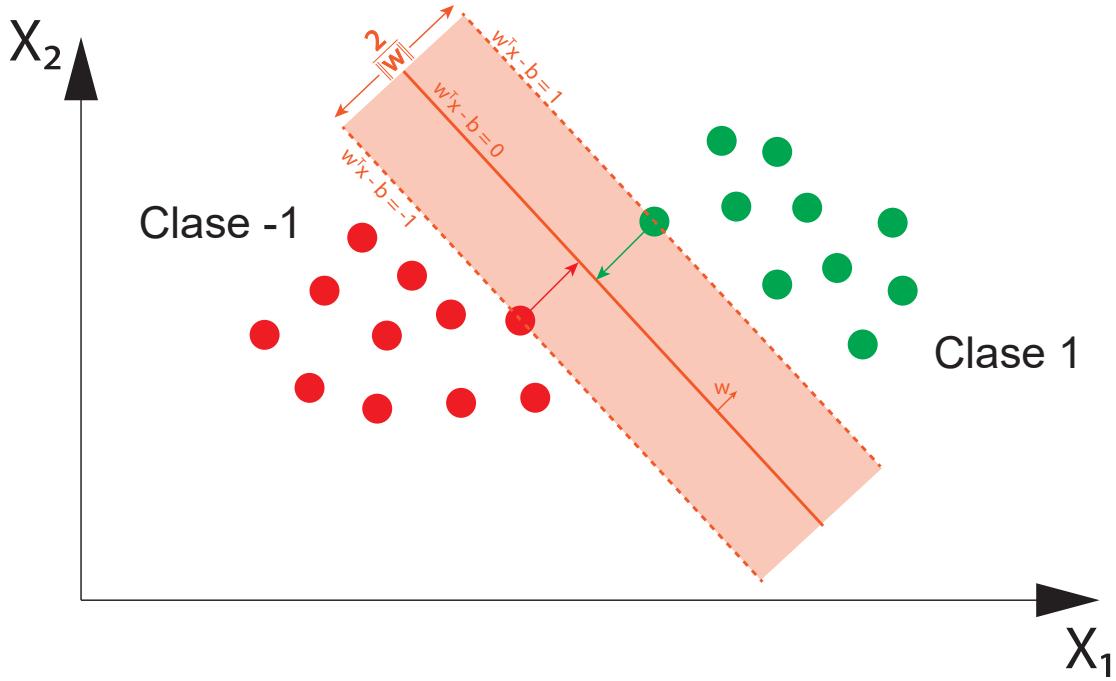


Fig. 4.11. Hiperplanos adicionales SVM

La distancia entre los dos hiperplanos adicionales es $\frac{2}{\|\mathbf{w}\|}$, por tanto, si se quiere maximizar el margen entre ambas rectas, se ha de minimizar $\|\mathbf{w}\|$. Además, dado que ningún punto de ambas clases debe estar entre los dos hiperplanos, se pueden añadir dos condiciones:

$$\mathbf{w}^T \mathbf{x}_i - b \geq 1, \text{ si } y_i = 1$$

$$\mathbf{w}^T \mathbf{x}_i - b \leq -1, \text{ si } y_i = -1$$

Reescribiendo ambas condiciones, se obtiene la expresión 4.29 y junto con la condición de minimizar $\|\mathbf{w}\|$ se define el problema de optimización 4.30:

$$y_i (\mathbf{w}^T \mathbf{x}_i - b) \geq 1, \forall 1 \leq i \leq n \quad (4.29)$$

$$\text{Minimizar } \|\mathbf{w}\| \text{ acorde a la condición } y_i (\mathbf{w}^T \mathbf{x}_i - b) \geq 1, \forall 1 \leq i \leq n \quad (4.30)$$

Datos linealmente inseparables

Cuando los datos no son linealmente separables se utiliza la función de pérdida conocida como *Hinge loss* (no tiene traducción formal al castellano) para minimizar el error.

Nota:

Una función de pérdida analiza cuánto se ha desviado la predicción del modelo frente al valor real.

La función de pérdida es definida como:

$$h(y) = \max(0, 1 - t \cdot y) \quad (4.31)$$

La variable t representa el valor real a determinar, mientras que y es la predicción del clasificador.

Comparando con la ecuación 4.29, $t = w^T x_i - b = \pm 1$.

En el caso de que el clasificador prediga correctamente la clase, $y = t$, siendo $|y| = 1$ y, por tanto, la función de pérdida $h(y) = \max(0, 0) = 0$. Cuando la predicción no sea correcta, $y = -t$, la función de pérdida $h(y) = \max(0, 1 - (-1) \cdot (1)) = \max(0, 2) = 2$. Es decir, $h(y)$ será nula cuando la predicción sea correcta.

Esto significa que la función de pérdida $h(y)$ será nula cuando x_i se encuentre en el lado correcto del hiperplano y fuera del margen.

Para los puntos que no puedan clasificarse y su error sea permitido, se introduce la variable ζ_i para x_i , que no es sino la función 4.31 evaluada para x_i . Además, se define un parámetro de regularización, C , que controla el maximizar el margen contra minimizar el error. Es decir, para elevados valores de C , el problema de optimización determinará un hiperplano con los márgenes mínimos posibles para poder abarcar todos los puntos posibles (minimizar los puntos mal clasificados); para valores reducidos de C , el optimizador buscará un hiperplano con márgenes más amplios, aunque cometa el error de clasificar erróneamente más puntos.

Por tanto, para los puntos "fuera de posición", el error se define como:

$$C \sum_{i=1}^n \zeta_i = C \sum_{i=1}^n \max(0, 1 - t \cdot y) \quad (4.32)$$

De tal forma, ahora el problema de optimización es:

$$\text{Minimizar } \frac{1}{2} \|w\|^2 + C \sum_{i=1}^n \max(0, 1 - t \cdot y) \quad (4.33)$$

acorde a la condición $y_i(w^T x_i - b) \geq 1 - \zeta_i, \forall 1 \leq i \leq n$

Parámetro ν INSERTAR CITA proponen el parámetro ν como alternativa a C , que representa un límite superior en la fracción de errores de margen y un límite inferior de la fracción de vectores de soporte. Es decir, el valor de ν determina el máximo de muestras

a ser mal clasificadas (fuera de márgenes) y el número mínimo de muestras que serán vectores de soporte. C y ν se relacionan de la forma $\nu = \frac{c_1 c_2}{C}$ siendo c_1 y c_2 constantes a calcular por el optimizador.

Problema dual

De acuerdo al Teorema de la Dualidad, es posible ver el problema de optimización definido por 4.30 (conocido como problema primitivo) desde otra perspectiva (problema dual).

Utilizando el método de los multiplicadores de Lagrange, se puede definir la ecuación 4.30 como 4.34, a partir de la cual el problema de optimización se define como 4.35. Por un lado se busca encontrar el valor de w, b que minimice el valor de 4.34 para cada α_i y escoger, de entre todos los α_i , el valor máximo.

$$\mathcal{L}(w, b, \alpha) = \frac{1}{2} \|w\|^2 - \sum_{i=1}^n \alpha_i (y_i((w^T x_i - b) - 1)) \quad (4.34)$$

$$\text{MAX}_{\alpha_i \geq 0} [\text{MIN}_{w,b} \mathcal{L}(w, b, \alpha)] \quad (4.35)$$

La resolución de 4.35 se consigue derivando parcialmente \mathcal{L} respecto de w y de b :

$$\begin{aligned} \frac{\partial \mathcal{L}}{\partial w} &= 0 \rightarrow w = \sum_{i=1}^n \alpha_i y_i x_i \\ \frac{\partial \mathcal{L}}{\partial b} &= 0 \rightarrow \sum_{i=1}^n \alpha_i y_i = 0 = \alpha_T y \end{aligned} \quad (4.36)$$

Para facilitar el cálculo, para los x_i que no sean vectores soporte (a partir de ahora SV), sus multiplicadores de Lagrange $\alpha_i = 0$.

De tal forma, si se sustituye 4.36 en 4.35 se obtiene la expresión de optimización para un clasificador SVM de la forma dual:

$$\text{MIN}_{\alpha_i \geq 0} \left[\sum_{i=1}^n \alpha_i - \frac{1}{2} \sum_{i,j} \alpha_i \alpha_j y_i y_j x_i^T x_j \right] \quad (4.37)$$

Nota: En la ecuación superior, (i, j) representan cada clase del problema.

Tanto la resolución del problema dual como la del primitivo permiten obtener las soluciones del otro (propiedad del Teorema de la Dualidad). Sin embargo, optimizar el SVM mediante 4.37 permite hacer uso de Kernels.

Aplicación de kernels en SVM

En muchas aplicaciones, los datos a clasificar pueden no parecer ser linealmente separables. Véase, por ejemplo, la figura

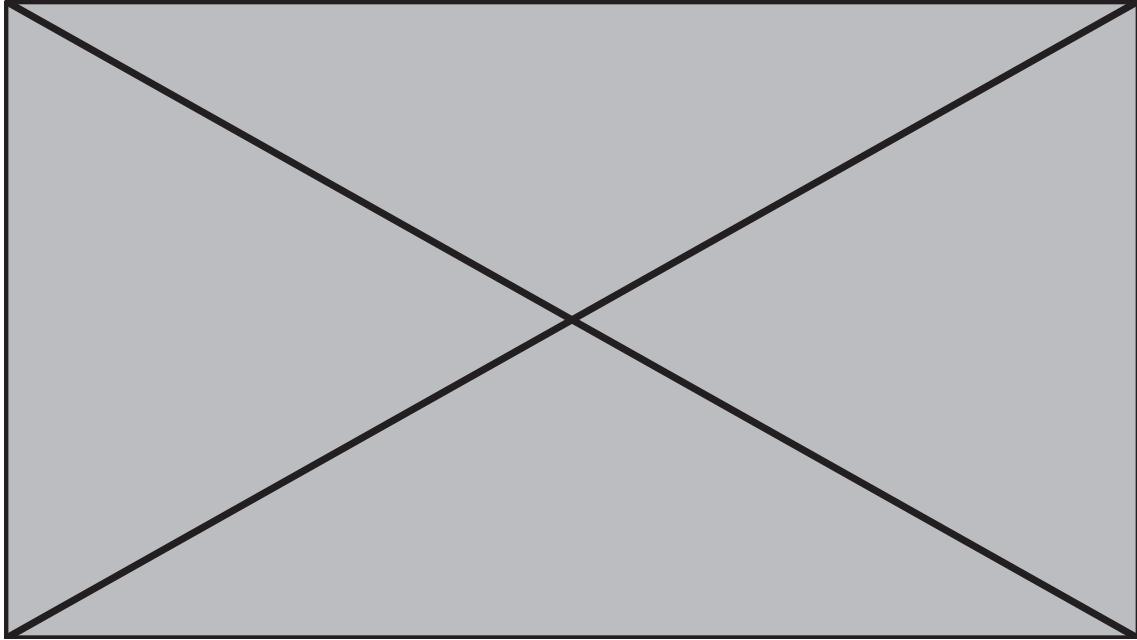


Fig. 4.12. Aquí debe ir una imagen sin referencia

Un conjunto de datos $X \in \mathbb{R}^n$ puede ser no separable con hiperplanos en dimensión n . Sin embargo, para una dimensión m , tal que $m > n$, X si es separable. De tal forma, se puede definir una función $\phi : \mathbb{R}^n \rightarrow \mathbb{R}^m$, que transforme los datos de un espacio dimensional n , donde no sean separables linealmente, a un espacio dimensional superior m donde si lo sean.

Haciendo referencia al problema dual, en la ecuación 4.37, segundo término, se presenta la multiplicación escalar de $x_i^T x_j$ para el espacio dimensional original \mathbb{R}^n . Esta expresión se entiende, a *grosso modo*, como que el clasificador SVM va a aprender de los datos en la dimensión n .

Si se da el caso de datos no lineales en n , pero si en m , el clasificador tendría que .^aprender" de los datos en la dimensión n y, por tanto, sería necesario calcular los valores de X en \mathbb{R}^m . Si m es muy elevado, esto es computacionalmente muy pesado, haciendo que sea una gran desventaja.

La aplicación de un kernel permite evitar tener que calcular la representación de X en \mathbb{R}^m , es decir, evita que el clasificador tenga que .^aprender" de los valores en la dimensión superior, haciendo que sólo haga falta conocer el resultado del producto vectorial en \mathbb{R}^m .

Un kernel $k(x_1, x_2)$ es una función que, bajo las condiciones del Teorema de Mercer, puede expresarse como una multiplicación escalar $\langle \phi(x_1), \phi(x_2) \rangle$, siendo $\phi : \mathbb{R}^n \rightarrow \mathbb{R}^m$.

Se define, por ejemplo, el kernel $k(x, x') = (x^T x')^3$, con $x, x' \in \mathbb{R}^2$, que resuelve el

producto escalar de x y x' en \mathbb{R}^4 usando sus formas en \mathbb{R}^2 , sin necesidad de calcular los valores de x, x' en \mathbb{R}^4 y realizar la multiplicación escalar en tal dimensión.

$$\begin{aligned}
k(a, b) &= (a^T b)^3 = ((a_1, a_2)^T (b_1, b_2))^3 \\
&= (a_1 b_1 + a_2 b_2)^3 \\
&= a_1^3 b_1^3 + 3a_1^2 b_1^2 a_2 b_2 + 3a_1 b_1 a_2^2 b_2^2 + a_2^3 b_2^3 \\
&= (a_1^3, \sqrt{3}a_1^2 a_2, \sqrt{3}a_1 a_2^2, a_2^3)^T (b_1^3, \sqrt{3}b_1^2 b_2, \sqrt{3}b_1 b_2^2, b_2^3) \\
&= \phi(a) \cdot \phi(b)
\end{aligned} \tag{4.38}$$

Gracias a 4.38, se ha obtenido el resultado de la multiplicación escalar $\langle \phi(x_1), \phi(x_2) \rangle$, sin necesidad de transformarlos a \mathbb{R}^4 , haciendo uso de sus formas en \mathbb{R}^2 , ahorrando el tiempo y cálculo necesario para ello.

De entre la gran cantidad de kernels ya definidos y los definibles por el usuario, se destacan los que serán usados más adelante en la implementación:

- Kernel lineal

$$k(x_1, x_2) = \langle x_1, x_2 \rangle \tag{4.39}$$

- Kernel polinómico

$$k(x_1, x_2) = (\gamma \langle x_1, x_2 \rangle + r)^d \tag{4.40}$$

- Kernel RBF

$$k(x_1, x_2) = e^{-\gamma \|x_1 - x_2\|^2} \tag{4.41}$$

- Kernel Sigmoid

$$k(x_1, x_2) = \tanh(\gamma \langle x_1, x_2 \rangle + r) \tag{4.42}$$

Clasificación multiclas con SVM

Debido al uso de hiperplanos, los modelos SVM sólo son capaces de clasificar binaria. Si se requiere su uso para la clasificación de más de dos clases, es necesario elegir entre dos estrategias:

Clasificación Uno contra Todos Abreviada como OVR (del inglés *One vs Rest*), el método de uno contra todos convierte una clasificación multiclas a binaria considerando la clase objetivo como la clase positiva (1) y las clases restantes como negativas (0). Sin embargo, requiere entrenar un clasificador por clase. Para un caso de N clases, se requiere de N clasificadores.

Clasificación Uno contra Uno Abreviada como OVO (del inglés *One vs One*), el método de uno contra uno convierte una clasificación multiclase a binaria considerando una clase como positiva (1) y entrenando un clasificador para cada clase restante, considerada como negativa (0). Este planteamiento requiere entrenar $\frac{N(N - 1)}{2}$ clasificadores, muchos más que con OVR, haciéndolo más lento y computacionalmente más pesado. Sin embargo, su gran ventaja es que, para métodos de clasificación (SVM y otros métodos) que sufren con grandes cantidades de datos, OVO solo clasifica entre dos clases, lo que implica clasificar muchas menos muestras en vez de la totalidad de ellas (OVR).

5. METODOLOGÍA

El proceso completo de clasificación puede dividirse en dos fases:

Primera fase, extracción y selección de características

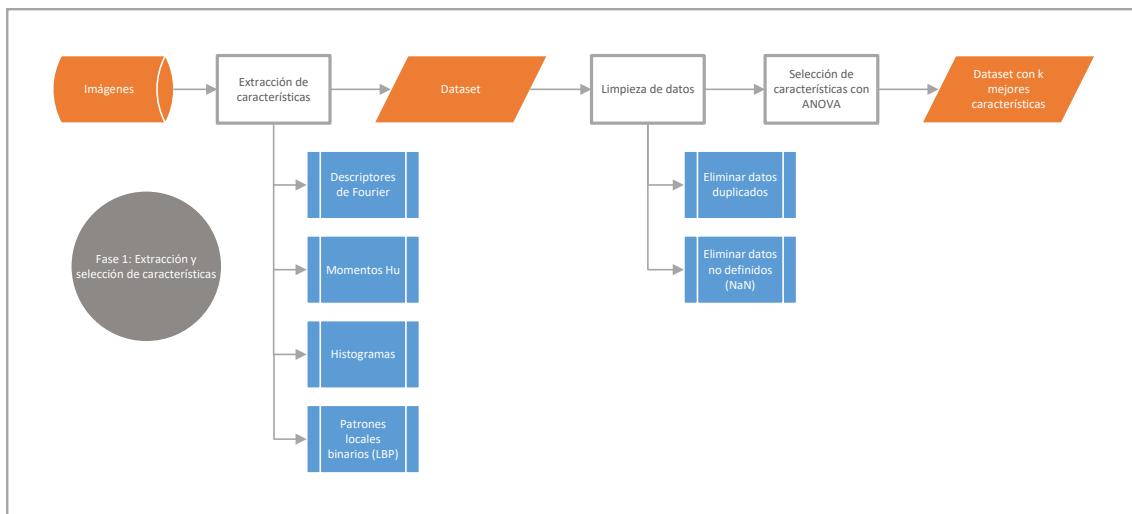


Fig. 5.1. Procesos en la fase de extracción y selección de características

Extracción de características El problema parte de un conjunto de imágenes almacenadas por clases y área a las que pertenecen. A partir de estas, se aplican los métodos de extracción de características.

En el caso de extracción de Momentos Hu se ha hecho uso de la librería OpenCV, que implementa un algoritmo para extraer siete momentos Hu de una imagen binaria. La extracción de los histogramas de gradientes se ha realizado a partir de código propio. Los patrones locales binarios han sido implementados de igual forma con código propio. Los descriptores de Fourier han sido obtenidos haciendo uso de la librería [11] para *Python*, creada por Domingo Mery.

Nota:

No sé si puedo mostrarlo, pero realmente no es nada del otro mundo el código, solo unas pocas funciones no muy eficientes XD.

De cada imagen se obtiene un vector de treinta y una posiciones (LBP ha sido descartado por no obtenerse un número uniforme de características, ya que estas dependen de la resolución de la imagen). Con un total de n instancias (ó imágenes) se crea una matriz de dimensiones $nx31$, llamada X . Dado que los algoritmos de clasificación son supervisados,

se crea, además, una matriz de etiquetas de valores $0, 1, 2, \dots, N$, siendo N el número de clases, de tamaño $nx1$, llamada Y . Uniendo ambas matrices se obtiene el dataset.

Nota:

Normalmente un dataset se guarda como un diccionario, u objeto *Bunch* de Sklearn[12], donde sus llaves son las etiquetas, datos, datos de fourier, datos de HoG y datos de momentos, de tal forma que se puede guardar en memoria y cargarlo en cualquier momento.

Limpieza de datos Con el dataset ya creado, es necesario entrar el proceso de limpieza de datos, que corresponde a dos subprocessos (para este caso):

- Eliminación de datos duplicados: existe la posibilidad de que en el directorio de almacenamiento de imágenes exista la misma imagen duplicada varias veces, obteniéndose datos redundantes en el dataset, para ello, es necesario eliminar esos datos duplicados, dejando claro está al menos una muestra original.
- Eliminación de datos no definidos: En la obtención de momentos, al aplicar el escalado logarítmico es posible que algunos valores se devuelvan como indeterminados (por la naturaleza del logaritmo), de tal forma, se aplica la fórmula $Hu = -\text{signo}(Hu)\log|Hu + 10^{-12}|$ para obtener datos numéricos y no indeterminaciones.

Selección de características Con el dataset creado y corregido, se aplica el método de los valores F de ANOVA para obtener una descripción del poder discriminatorio de cada una de las 31 características del dataset. ANOVA se implementa a partir de la clase *f_classif* de Sklearn[12], que devuelve un vector de tamaño $1x31$ con los 31 F valores de las características. Con la clase *SelectKBest*, de Sklearn[12] también, se pueden seleccionar automáticamente, sin obtener previamente los F valores, las k mejores características en función de ANOVA.

Tras las selección de características, el dataset reduce su tamaño de $nx31$ a nxk .

Segunda fase, selección y análisis del clasificador

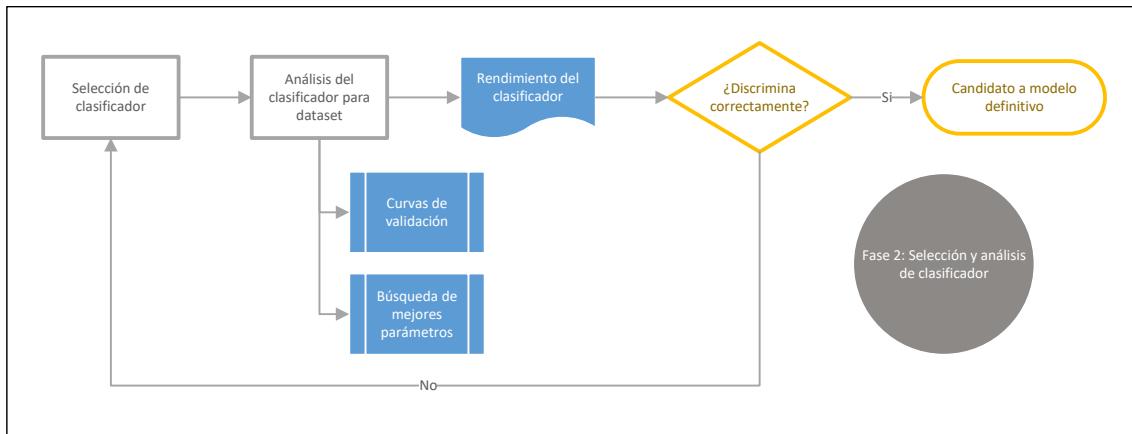


Fig. 5.2. Selección y análisis del clasificador

Con el dataset ya creado y las k mejores características seleccionadas se procede a realizar pruebas con diferentes clasificadores.

En primer lugar, es necesario comprobar si el dataset contiene un número balanceado de muestras, para así, poder determinar que métricas utilizar para analizar el rendimiento.

Con un clasificador elegido para analizar su rendimiento (todos son clases de Sklearn [13]), se analiza el rendimiento del clasificador para el dataset en función de los valores de sus parámetros (curva de validación). Con este paso se consigue extraer un rango de valores para cada parámetro en el que el clasificador sobre ajuste una mínima cantidad, o discrimine perfectamente, con el cual hacer una búsqueda de los parámetros que devuelvan el mayor rendimiento para el clasificador elegido en el dataset.

Una vez obtenidos los mejores parámetros, se obtienen las métricas promedio de rendimiento (ya que es un problema multiclas) para el clasificador con esas variables.

Si se considera que el clasificador realiza una buena discriminación, se plantea como buen candidato a modelo definitivo. De lo contrario, se escoge otro modelo de clasificador y se realiza de nuevo el mismo proceso.

6. RESULTADOS

6.1. Problema desbalanceado

Visualizando el número de muestras para tanto la clase B345 como C1C9 (figuras 6.1 y 6.2, respectivamente y el número de muestras por clase en las tablas 6.1 y 6.2) se puede afirmar que se trata de un problema de clasificación desbalanceado.

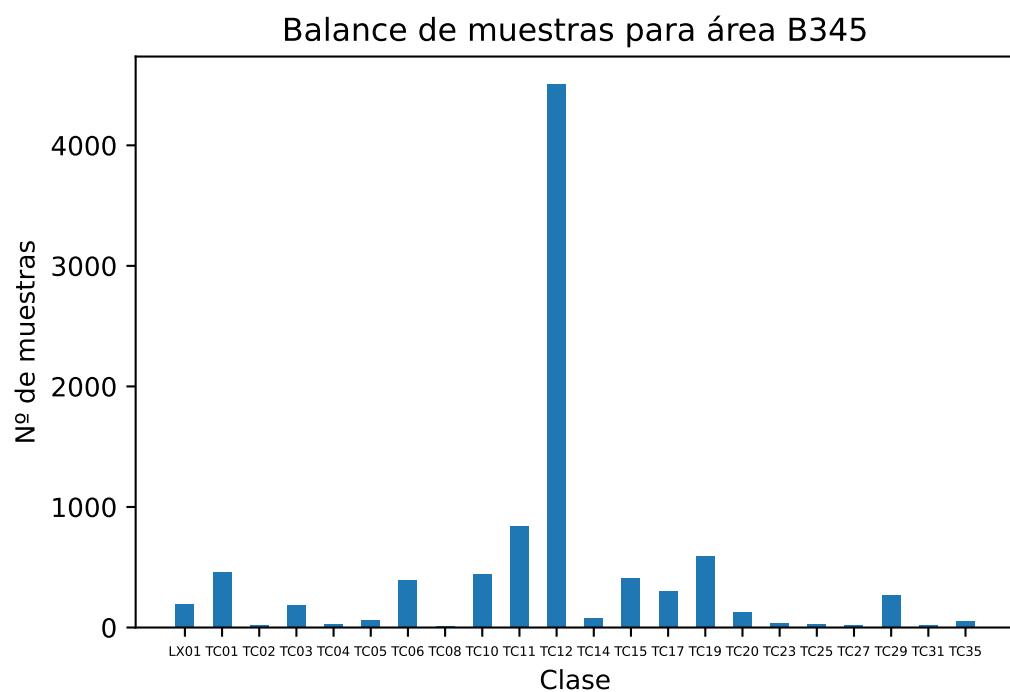


Fig. 6.1. N° de muestras para el dataset del área B345

Clase	Nº de muestras
LX01	194
TC01	457
TC02	22
TC03	185
TC04	25
TC05	64
TC06	397
TC08	15
TC10	442
TC11	838
TC12	4511
TC14	82
TC15	414
TC17	304
TC19	590
TC20	131
TC23	38
TC25	30
TC27	21
TC29	269
TC31	18
TC35	52

TABLA 6.1. N° DE MUESTRAS PARA ÁREA B345

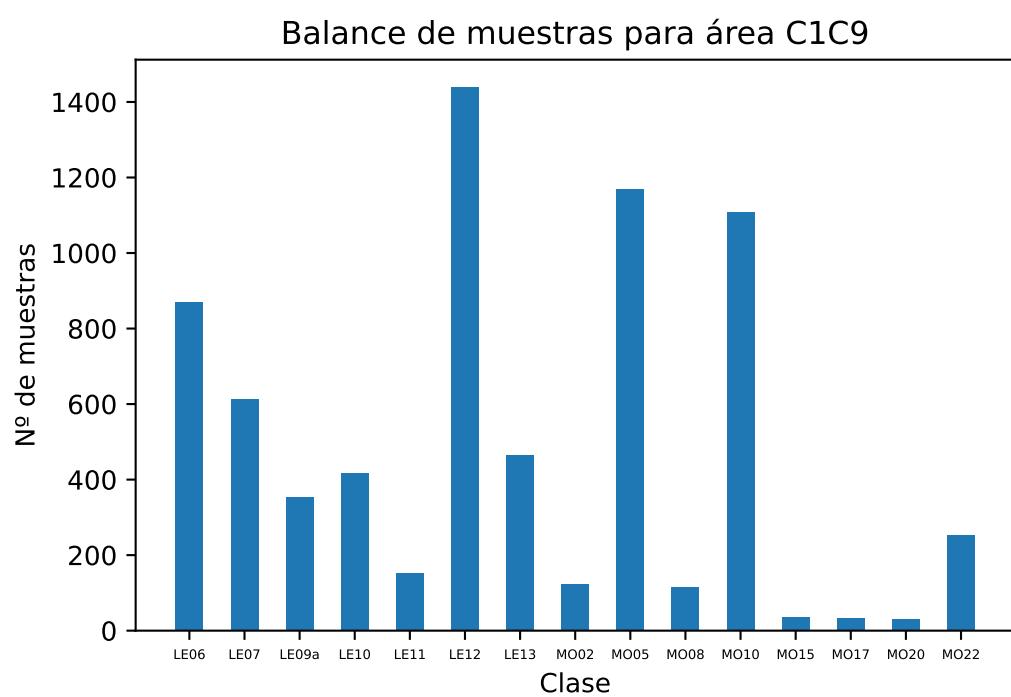


Fig. 6.2. N° de muestras para el dataset del área C1C9

Clase	Nº de muestras		
LE06	869	MO05	1168
LE07	613	MO08	116
LE09a	354	MO10	1109
LE10	418	MO15	35
LE11	153	MO17	34
LE12	1440	MO20	31
LE13	464	MO22	253
MO02	124		

TABLA 6.2. N° DE MUESTRAS PARA ÁREA C1C9

Para analizar resultados, al tratar con clases desbalanceadas, no se usa en ningún momento la exactitud y, a priori, tanto promedio micro, macro y ponderado de las métricas darán resultados que favorezcan a las clases más frecuentes.

6.2. ANOVA

El método LBP no ha sido utilizado debido a que el numero de características extraídas de una imagen depende de su resolución. Dado que el dataset contiene imágenes con resoluciones diferentes, no se puede obtener un número uniforme de características.

Para la selección de características con ANOVA, se obtienen los siguientes resultados:

6.2.1. Área B345

Característica	F-valor		
H0	23859.201619	F3	821.479145
H4	12429.265116	F11	675.605577
H6	7880.238366	F1	670.737686
M3	3895.946032	F0	661.628967
H1	3827.930644	M4	514.887416
M0	3486.251597	F9	472.654444
H5	3090.149390	M5	386.561602
M2	2880.764008	F12	369.108364
F7	2439.522794	M6	202.739105
F6	1810.805758	F8	202.675900
H2	1715.595292	F14	172.541408
F5	1267.176107	F10	144.443042
F4	1049.974086	F15	86.515230
M1	1003.760963	F13	77.369423
F2	879.120903	H3	nan
		H7	nan

TABLA 6.3. RESULTADOS ANOVA PARA ÁREA B345

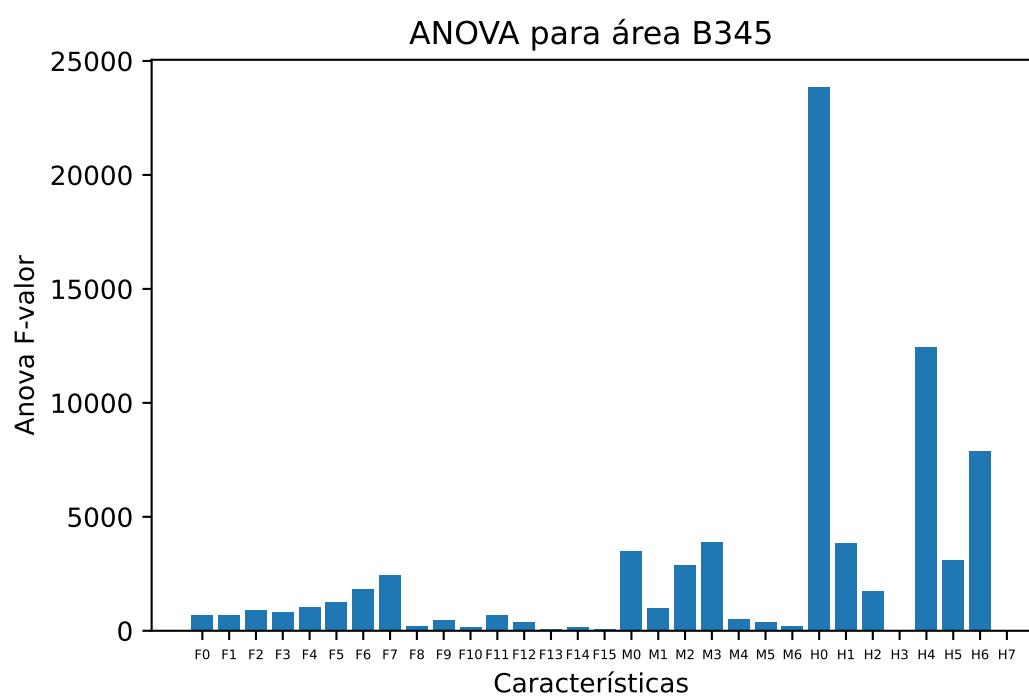


Fig. 6.3. Resultados ANOVA para área B345

De entre todas las características extraídas, por el método ANOVA se ha determinado que H0, H1, H4, H5, H6 y F7 son las seis características más discriminantes, seguidas por los momentos M0, M1, M2 y M3.

En el caso de H3 y H7, ANOVA devuelve un valor no definido pues, estas características son constantes para todas las muestras.

6.2.2. Área C1C9

Característica	F-valor		
M1	6125.638894	F6	221.004593
H4	5857.112939	F5	189.035358
H0	4554.790583	F8	164.096954
M0	3048.724574	F14	156.956539
H5	1787.841543	M4	140.206197
M2	1752.153444	F2	133.046369
H2	1427.452251	F9	122.262756
H1	1040.217828	F11	105.074395
M3	817.966512	F0	78.174618
F3	528.279034	F4	77.091309
H6	376.474004	M6	75.595968
F7	276.076666	F15	73.126625
F1	267.434392	F12	60.370951
F10	252.051888	F13	48.366221
M5	229.500016	H3	nan
		H7	nan

TABLA 6.4. RESULTADOS ANOVA PARA ÁREA C1C9

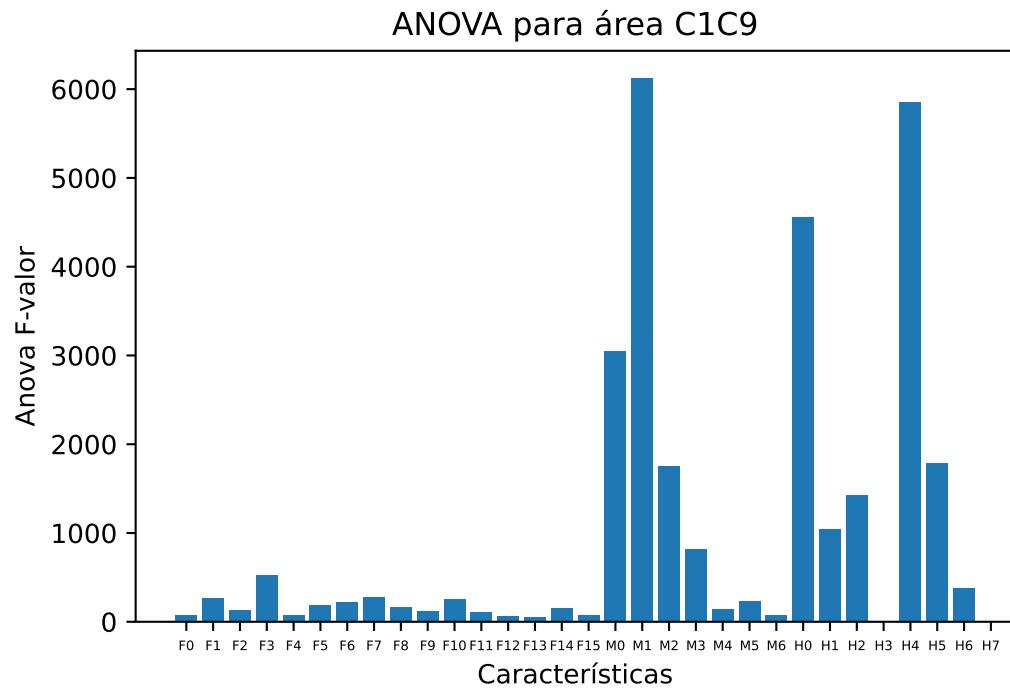


Fig. 6.4. Resultados ANOVA para área C1C9

Al igual que en el área B345, las características de Hog siguen siendo significativas, pero en este caso el momento de Hu segundo (M1) es la segunda característica más significativa.

Igual que para B345, H3 y H7 no están definidas (*NaN*) por ser constantes.

Nota: Usando la clase *SelectKBest*, proporcionada por Sklearn, se pueden escoger automáticamente las k mejores características según los F-valores de ANOVA.

6.3. K-vecinos más cercanos

Al no devolver Knns predicciones probabilísticas, no se usa la curva ROC ó Precisión-Exhaustividad para analizar sus resultados.

6.3.1. Resultados con clasificador Knns para área B345

Curvas de validación para Knns

A continuación, se presenta la curva de validación para ambos modelos de clasificador Knns (uniforme y distancia) para las métricas de recall, precisión y f_1 .

Para todos los resultados, se han seleccionado las siete mejores características según ANOVA (véase 6.3).

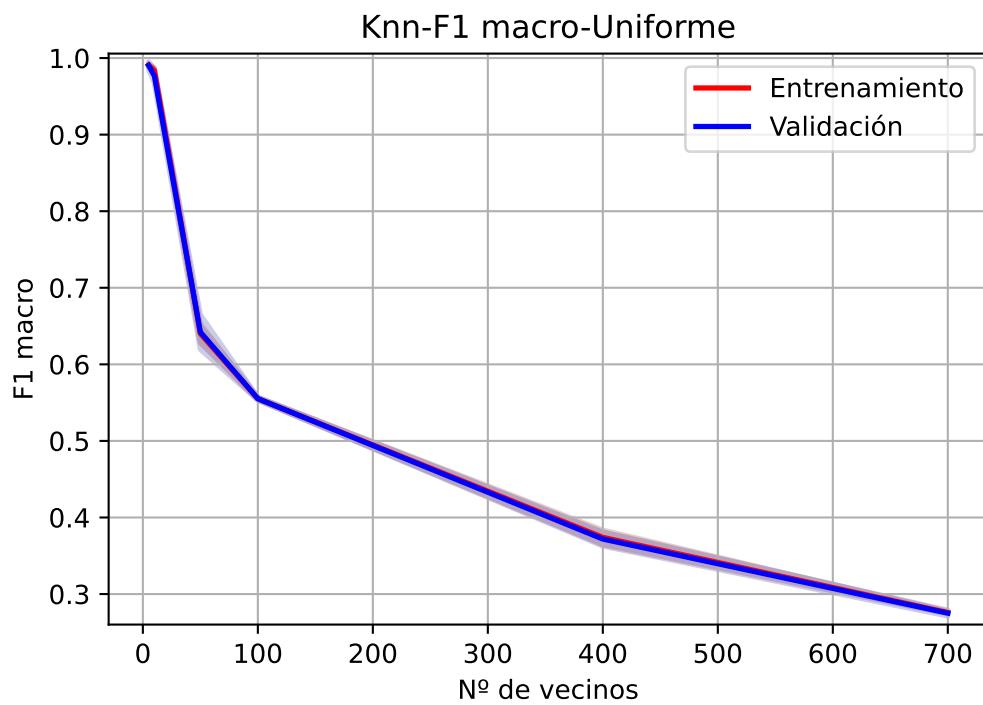


Fig. 6.5. Curva de validación para modelo KNN uniforme, F1 macro

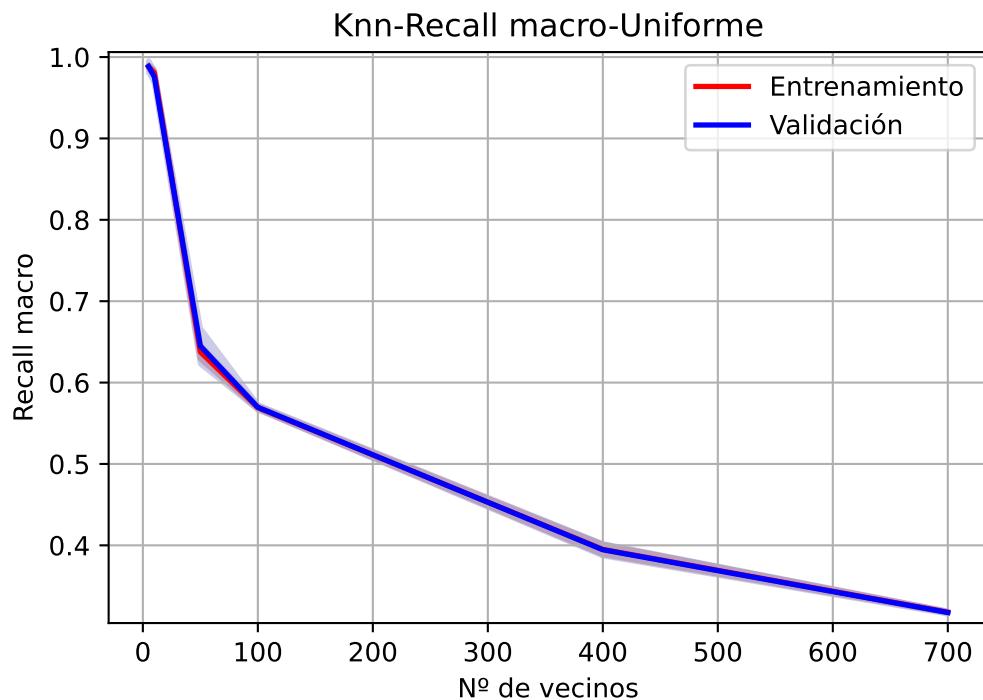


Fig. 6.6. Curva de validación para modelo KNN uniforme, Recall macro

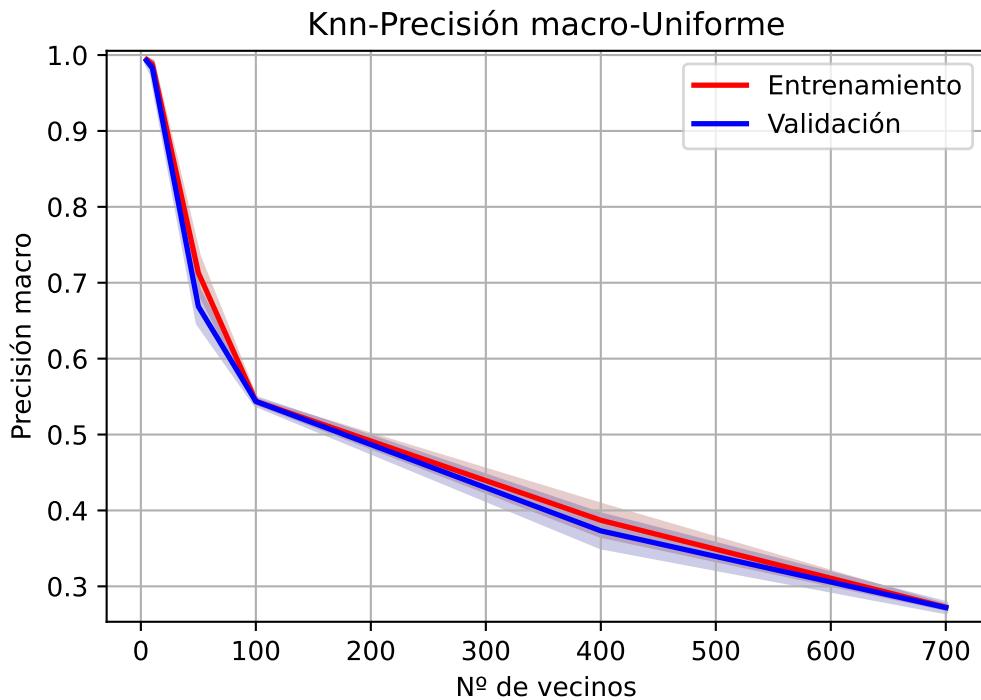


Fig. 6.7. Curva de validación para modelo KNN uniforme, Precisión macro

Las tres figuras muestran la independencia del nº de vecinos del clasificador frente a un sobre ajuste y sub ajuste del modelo. También, para siete características, a mayor valores de n peor es el resultado del modelo.

Los mejores resultados se obtienen para el rango de [5, 20], como se muestra en las siguientes figuras:

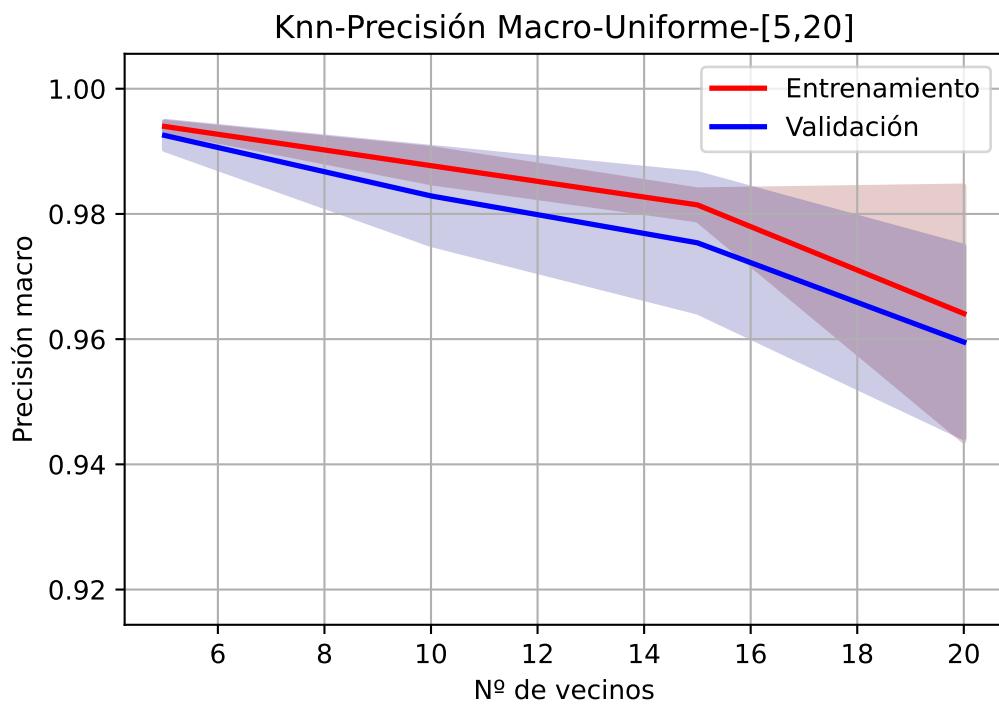


Fig. 6.8. Curva de validación para modelo KNN uniforme en rango n=[5,20], Precisión macro

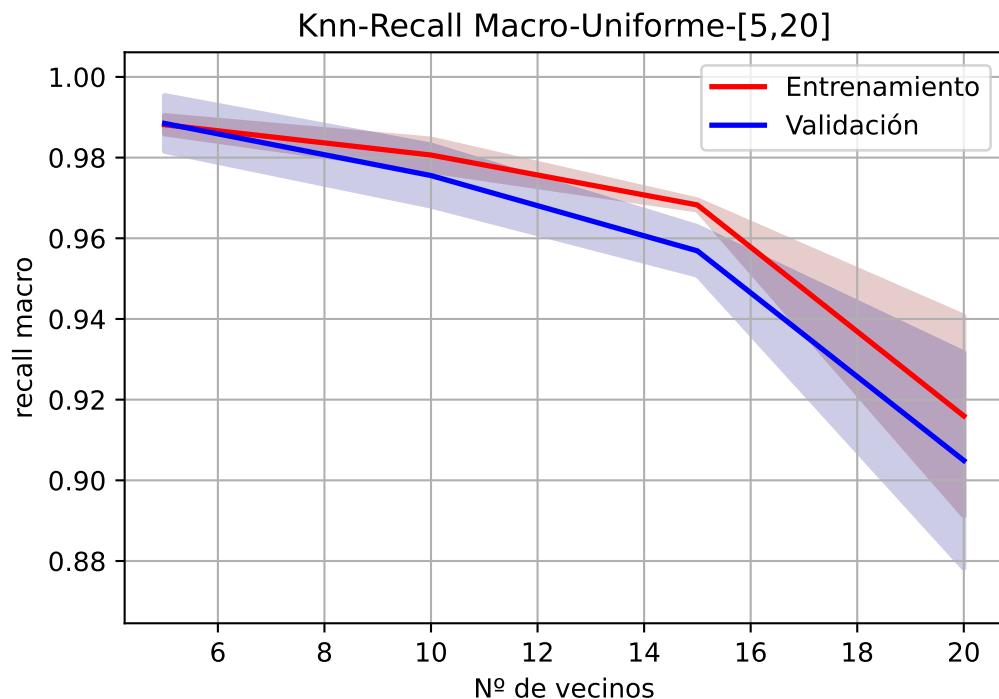


Fig. 6.9. Curva de validación para modelo KNN uniforme en rango n=[5,20], Recall macro

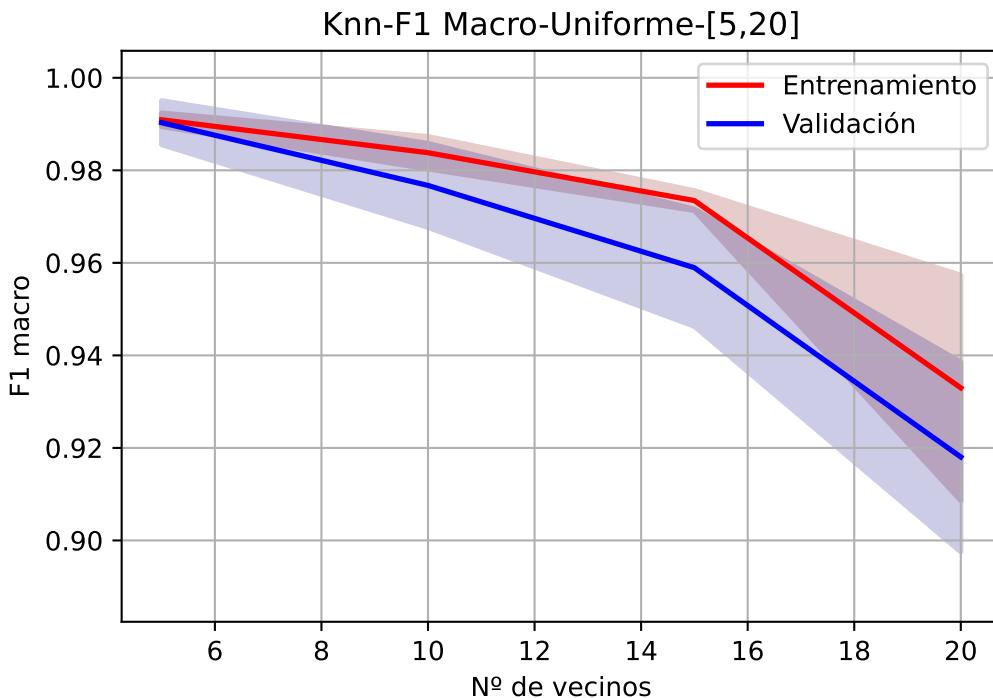


Fig. 6.10. Curva de validación para modelo KNN uniforme en rango $n=[5,20]$, F1 macro

En todas las figuras, sobre todo 6.10 y 6.9, a partir de $n = 15$, inclusive, se presenta un aumento en la desviación de las métricas, indicando una mayor variación en el rendimiento en función de la partición escogida del dataset, lo que representa una menor capacidad de generalización del modelo en la fase de entrenamiento, que no consigue plasmar los mismos buenos resultados (en comparación, porque en general son excelentes, aún con este desliz) en la fase de validación.

De las seis figuras anteriores, se puede afirmar que los mejores resultados, con las siete mejores características, se dan para $n < 15$.

Mejores parámetros B345

Realizando una búsqueda de parámetros entre [2, 31] mejores características y [5, 30] vecinos más cercanos, con una estrategia de validación cruzada y mezclado de datos, se obtiene la tabla 6.5.

Nº de características	19
Nº de vecinos	6

TABLA 6.5. MEJORES PARÁMETROS MODELO KNN PARA ÁREA
B345

Los resultados por clase y promediados para un modelo knn de 6 vecinos más cercanos

utilizando las 19 mejores características devuelve los resultados de la tabla 6.6.

Métricas por clase				
Clase	Precisión	Recall	F1	Nº de muestras
LX01	1.000000	0.994845	0.997416	194
TC01	0.995643	1.000000	0.997817	457
TC02	1.000000	1.000000	1.000000	22
TC03	1.000000	1.000000	1.000000	185
TC04	1.000000	0.960000	0.979592	25
TC05	0.984615	1.000000	0.992248	64
TC06	1.000000	1.000000	1.000000	397
TC08	1.000000	0.933333	0.965517	15
TC10	0.972540	0.961538	0.967008	442
TC11	0.979834	0.985680	0.982748	838
TC12	0.999778	1.000000	0.999889	4511
TC14	1.000000	0.987805	0.993865	82
TC15	0.997590	1.000000	0.998794	414
TC17	0.996721	1.000000	0.998358	304
TC19	0.998305	0.998305	0.998305	590
TC20	0.992424	1.000000	0.996198	131
TC23	1.000000	0.921053	0.958904	38
TC25	1.000000	1.000000	1.000000	30
TC27	1.000000	1.000000	1.000000	21
TC29	1.000000	1.000000	1.000000	269
TC31	1.000000	1.000000	1.000000	18
TC35	1.000000	1.000000	1.000000	52
Micro	0.995934	0.995934	0.995934	9099
Macro	0.952933	0.945329	0.948985	9099
Ponderada	0.995934	0.995934	0.995920	9099

TABLA 6.6. MÉTRICAS POR CLASE Y PROMEDIOS PARA ÁREA
B345 CON MODELO KNN

Promedio	Precisión	Recall	F1
Micro	99.59 %	99.59 %	99.59 %
Macro	95.29 %	94.53 %	94.90 %
Ponderada	99.59 %	99.59 %	99.59 %

TABLA 6.7. MÉTRICAS PROMEDIADAS KNN B345

La tabla 6.7 muestra que los resultados promedios para el problema general son muy elevados, obteniéndose un 94,9 % para F_1 . La diferencia entre los promediados micro y

ponderado frente al macro indican (además de un desbalanceado en los datos) que no todas las clases tienen un rendimiento cerca del 100 % como indicarían los otros promedios, la TC23 obtiene un $F_1 = 95,89\%$ y sólo presenta 38 muestras, por ejemplo. Sin embargo, los resultados son más que satisfactorios dada la poca cantidad de muestras y la gran cantidad de clases (22). Los valores de recall y precisión indican que el modelo es bueno identificando muestras positivas de cada clase y acertando en la identificación de estas a la clase pertinente, respectivamente.

Nota:

Es muy posible que se obtengan resultados diferentes por milésimas seleccionando menos características, ya que el método *GridSearchCV* va a buscar el mejor valor posible según la métrica escogida y devolverá los parámetros de esa métrica, sin considerar que el segundo mejor valor puede estar en el orden de millonésimas por debajo. Sin embargo, con los parámetros escogidos, la clasificación es la mejor posible, así que tampoco importa cuantos coja. Se puede jugar un poco y obtener la métrica para cada combinación de parámetros, pero no es el objetivo de este trabajo, solo decir que para estos parámetros el clasificador funciona que da gusto.

6.3.2. Resultados con clasificador Knn para área C1C9

Curvas de validación

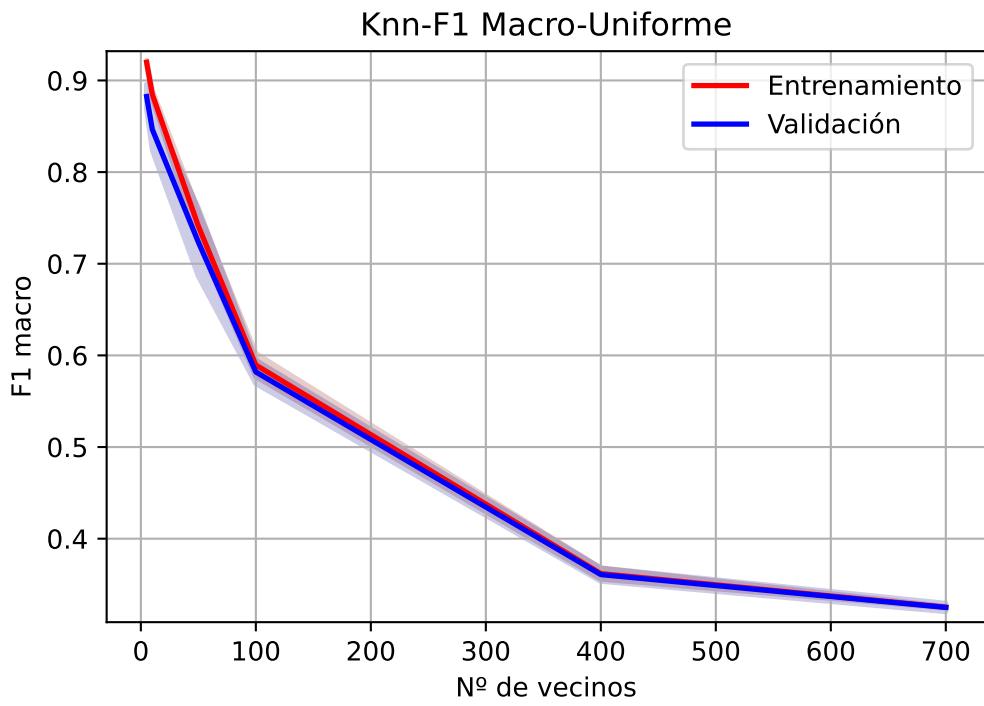


Fig. 6.11. Curva de validación para modelo KNN uniforme, F1 macro

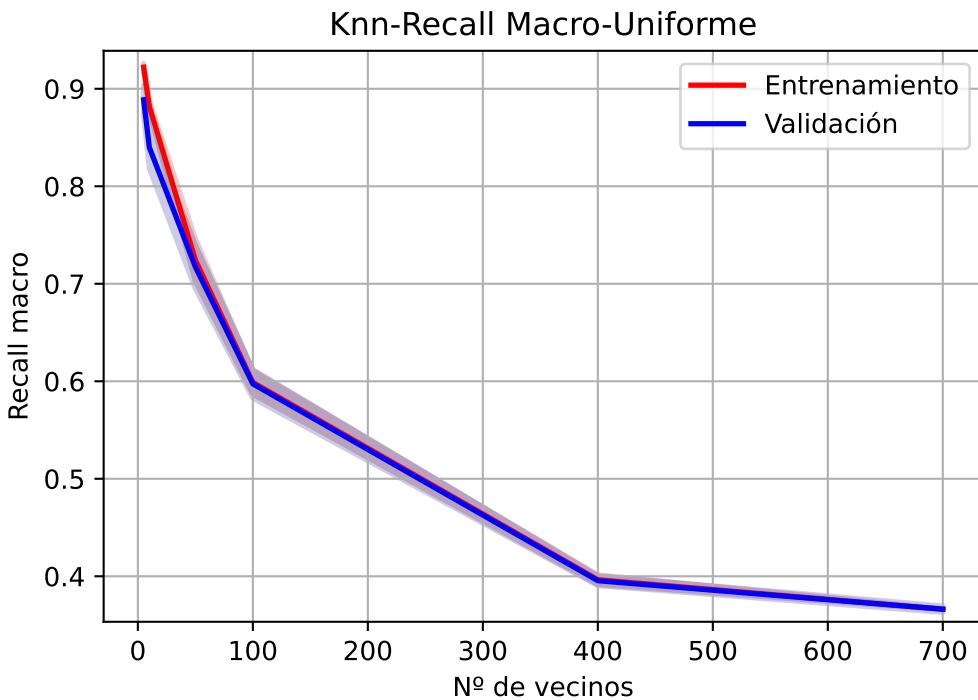


Fig. 6.12. Curva de validación para modelo KNN uniforme, Recall macro

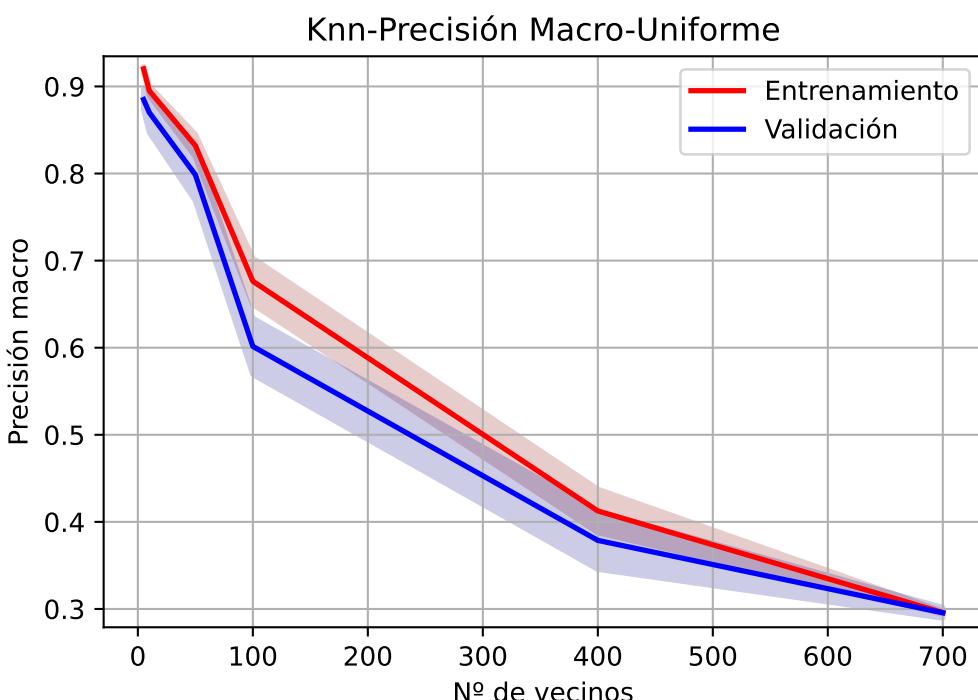


Fig. 6.13. Curva de validación para modelo KNN uniforme, Precisión macro

A diferencia que para el área B345, en este caso la curva de validación para el área C1C9 muestra un resultado sobre ajustado en precisión. Aunque el modelo es exhaustivo

encontrando muestras positivas, no consigue acertarlas correctamente. Los falsos positivos para el caso de validación son mayores que en el entrenamiento, indicando que existe una similitud entre varios símbolos que el clasificador no es capaz de discriminar correctamente. Este comportamiento se acentúa en $n = 100$, desapareciendo por completo en $n = 700$ y, aun así, los resultados son aceptables para $n < 30$.

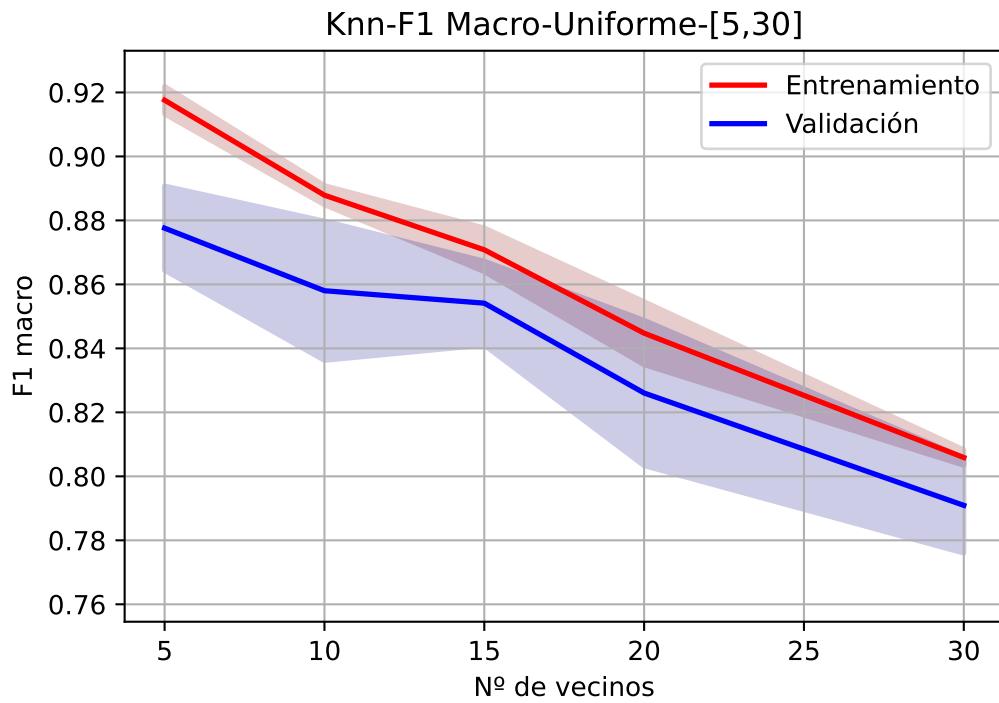


Fig. 6.14. Curva de validación para modelo KNN uniforme, F1 macro

Mejores parámetros C1C9

Realizando una búsqueda de parámetros entre [2, 31] mejores características y [5, 30] vecinos más cercanos, con una estrategia de validación cruzada y mezclado de datos, se obtiene la tabla 6.8.

Nº de características	7
Nº de vecinos	5

TABLA 6.8. MEJORES PARÁMETROS MODELO KNN PARA ÁREA
C1C9

Los resultados por clase y promediados para un modelo knn de 5 vecinos más cercanos utilizando las 7 mejores características devuelven los resultados de la tabla 6.9.

Métricas por clase				
Clase	Precisión	Recall	F1	Nº de muestras
LE06	0.794737	0.868815	0.830126	869
LE07	0.758879	0.662316	0.707317	613
LE09a	0.991525	0.991525	0.991525	354
LE10	0.917040	0.978469	0.946759	418
LE11	0.888112	0.830065	0.858108	153
LE12	0.930061	0.960417	0.944995	1440
LE13	0.852217	0.745690	0.795402	464
MO02	0.881890	0.903226	0.892430	124
MO05	0.992201	0.980308	0.986219	1168
MO08	0.935185	0.870690	0.901786	116
MO10	0.956289	0.966637	0.961435	1109
MO15	0.842105	0.914286	0.876712	35
MO17	0.969697	0.941176	0.955224	34
MO20	0.658537	0.870968	0.750000	31
MO22	0.890756	0.837945	0.863544	253
Micro	0.906559	0.906559	0.906559	7181
Macro	0.576488	0.579241	0.576591	7181
Ponderada	0.906031	0.906559	0.905327	7181

TABLA 6.9. MÉTRICAS POR CLASE Y PROMEDIOS PARA ÁREA C1C9 CON MODELO KNN

Promedio	Precisión	Recall	F1
Micro	90.66 %	90.66 %	90.66 %
Macro	57.65 %	57.92 %	57.66 %
Ponderada	90.6 %	90.66 %	90.53 %

TABLA 6.10. MÉTRICAS PROMEDIADAS KNN C1C9

Las métricas ponderadas de la tabla 6.10 revelan que, en el caso de promedio macro, aun existiendo clases con un alto valor de rendimiento, existen otras cuyos valores no son tan elevados, siendo aquellas menos frecuentes en el dataset. En general los resultados son aceptables, excepto para las clases MO20 y LE07.

Nota:

Esto se debe a que la calidad de las imágenes de la C1C9 es bastante peor que la B345, porque en muchas aparecen hasta dedos. No sé si puedo mostrar imágenes ejemplo de los símbolos, pero supongo que si no, al menos puedo hacer un inciso a porqué los resultados salen peor que para la B345.

6.4. SVM

Los modelos SVM de *Sklearn* permiten obtener predicciones probabilísticas de los datos de validación, permitiendo el uso de la curva de precisión-recall en la evaluación de su rendimiento.

6.4.1. SVM con área C1c9

Curvas de validación para diferentes Kernel

La cuadrícula de imágenes 6.15 representa las curvas de validación para los diferentes kernel aplicables a un modelo SVM, implementados en la clase SVC de *Sklearn*, definidos en las ecuaciones 4.39, 4.41, 4.40 y 4.42.

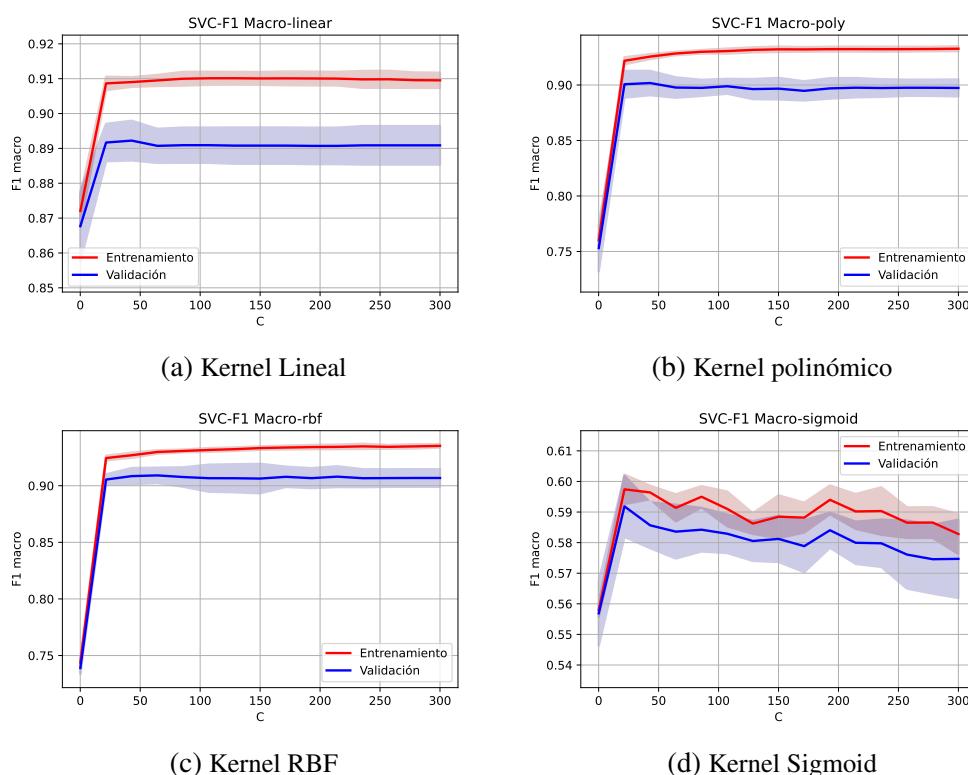


Fig. 6.15. Curvas de validación para kernels de modelo SVC en área C1c9 en función de parámetro de penalización C.

Para los cuatro kernel disponibles en la clase SVC de *Sklearn*, el modelo presenta para valores altos de C un sobreajuste de los datos de entrenamiento frente a los de validación. Para valores pequeños de C, el modelo tiene un buen comportamiento.

La figura 6.15d presenta un rendimiento inferior a 0.6, indicando que el kernel *sigmoid* no es aplicable a esta clasificación.

Kernel RBF El kernel RBF (ecuación 4.41) depende del parámetro γ en su definición. La figura 6.16 representa la curva de validación de dicho parámetro:

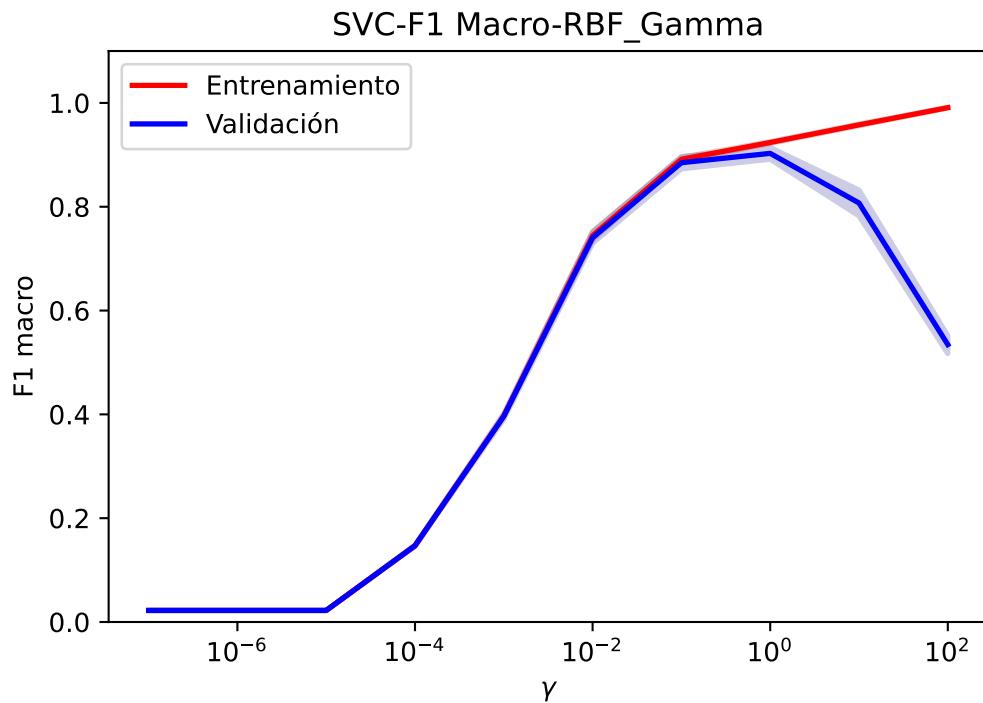
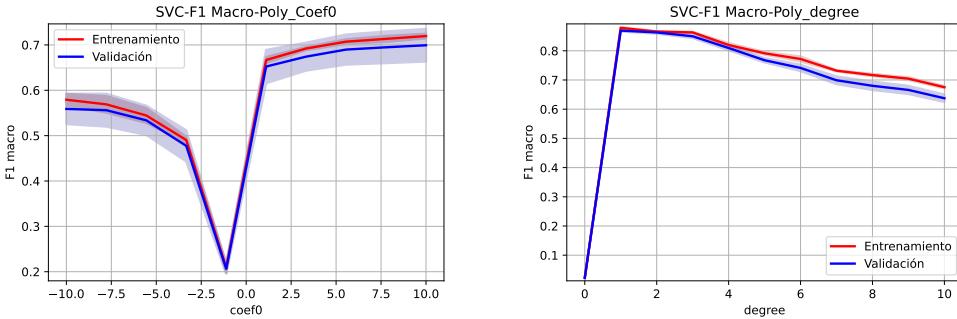


Fig. 6.16. Curva de validación para kernel RBF en función de γ en área C1c9.

A partir de $\gamma = 10^{-2} = 0,01$ el modelo empieza a sobre ajustar los datos de entrenamiento frente a validación, divergiendo completamente a partir de $\gamma = 10^{-1} = 0,1$. Para $\gamma < 10^{-2}$ el rendimiento para entrenamiento y validación muestran los mismos resultados, indicando que el modelo funciona correctamente para tal rango. Sin embargo, el rendimiento para estos valores es muy bajo.

Kernel polinómico Definido por la ecuación 4.40, el kernel polinómico es dependiente de parámetros γ , r (también llamado *coef0*) y d (grado del polinomio, en inglés *degree*).



(a) Curva de validación para kernel polinómico en función de r en área C1c9. (b) Curva de validación para kernel polinómico en función del grado en área C1c9.

Fig. 6.17. Curvas de validación para kernel polinómico

La figura 6.17a representa un correcto comportamiento del modelo para el rango $r \in [-3, 1, 75]$, sobre ajustando fuera de estos límites. Para 6.17b, el modelo funciona correctamente en $d < 3$, tendiendo a sobre ajustar a medida que se aumenta el grado del kernel.

Kernel lineal El kernel lineal (ecuación 4.39) no requiere de más parámetros excepto C , el parámetro de regularización, por lo tanto, su curva de validación ya se haya representada en 6.15a.

Mejores parámetros SVC para área C1C9

Kernel RBF Los mejores parámetros encontrados, según métrica F_1 , para $k(\text{ANOVA}) \in [2, 31]$, $\gamma \in [10^{-7}, 10^2]$ y $C \in [1, 100]$ han resultado:

Parámetro	Valor
k (ANOVA)	9
C	20
γ	1

Kernel Lineal Los mejores parámetros encontrados, según métrica F_1 , para $k(\text{ANOVA}) \in [2, 31]$ y $C \in [1, 100]$ han resultado:

Parámetro	Valor
k (ANOVA)	9
C	15

Kernel polinómico Los mejores parámetros encontrados, según métrica F_1 , para $k(\text{ANOVA}) \in [2, 31]$ y $C \in [1, 100]$ han resultado:

Parámetro	Valor
k (ANOVA)	9
C	15
r	1
Grado	3
γ	1

AUC-Ovo-C1C9-Poly		AUC-Ovo-C1C9-Lineal		AUC-Ovo-C1C9-RBF	
Clase	AUC	Clase	AUC	Clase	AUC
LE06	0.897256	LE06	0.835289	LE06	0.923290
LE07	0.801268	LE07	0.703032	LE07	0.797598
LE09a	1.000000	LE09a	1.000000	LE09a	1.000000
LE10	0.982807	LE10	0.972459	LE10	0.986225
LE11	0.898879	LE11	0.863337	LE11	0.934888
LE12	0.992003	LE12	0.924740	LE12	0.990411
LE13	0.943605	LE13	0.776634	LE13	0.943534
MO02	0.989694	MO02	0.990316	MO02	0.997832
MO05	0.999336	MO05	0.999177	MO05	0.998750
MO08	0.982773	MO08	0.955717	MO08	0.992643
MO10	0.999272	MO10	0.999730	MO10	0.999781
MO15	0.985674	MO15	0.966846	MO15	0.993808
MO17	0.988889	MO17	1.000000	MO17	0.979798
MO20	0.596893	MO20	0.667409	MO20	0.646762
MO22	0.997976	MO22	0.996742	MO22	0.997654

TABLA 6.11. RESULTADOS AUC PARA DIFERENTES KERNEL
SVM PARA ÁREA C1C9

Kernel	AUC Macro
Polinómico	0.9371
Lineal	0.9116
RBF	0.9467

TABLA 6.12. AUC MACRO PARA C1C9

6.4.2. SVM con área B345

Curvas de validación para diferentes Kernel

La cuadrícula de imágenes 6.18 representa las curvas de validación para los diferentes kernel aplicables a un modelo SVM, implementados en la clase SVC de *Sklearn*, definidos en las ecuaciones 4.39, 4.41, 4.40 y 4.42.

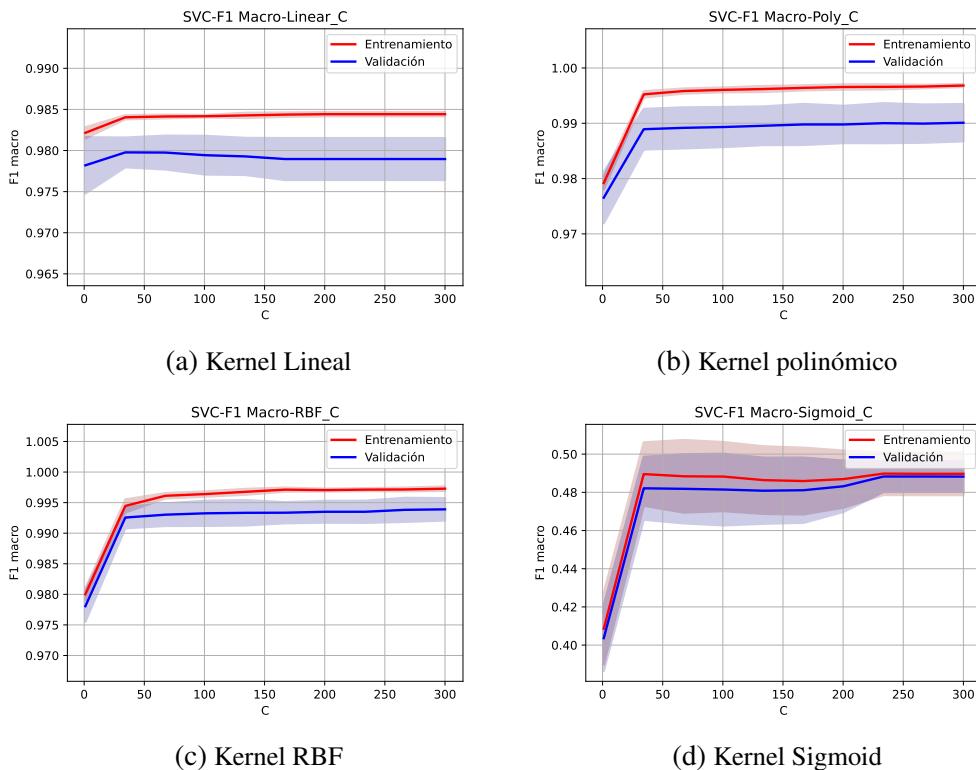


Fig. 6.18. Curvas de validación para kernels de modelo SVC en área B345 en función de parámetro de penalización C.

Para los cuatro kernel disponibles en la clase SVC de *Sklearn*, en general, el modelo presenta obreajuste de los datos de entrenamiento frente a los de validación. La figura 6.15d presenta un rendimiento inferior a 0.5, indicando que el kernel *sigmoid* no es aplicable a esta clasificación.

Kernel RBF El kernel RBF (ecuación 4.41) depende del parámetro γ en su definición. La figura 6.19 representa la curva de validación de dicho parámetro:

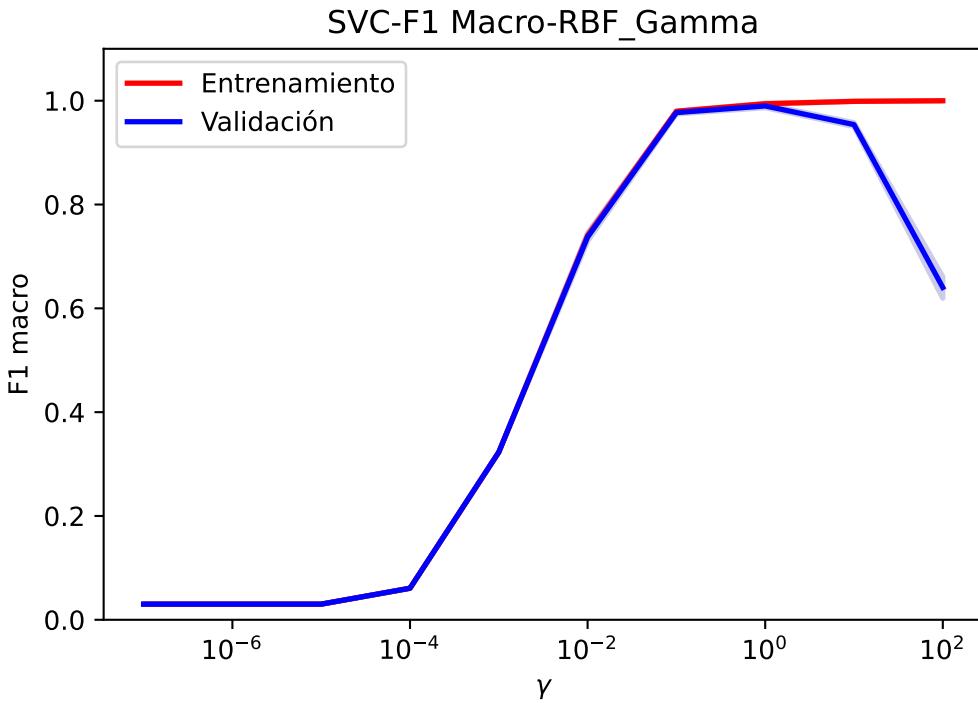
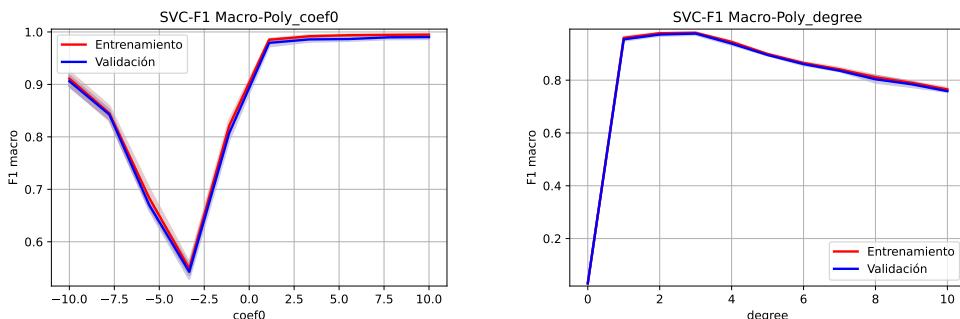


Fig. 6.19. Curva de validación para kernel RBF en función de γ en área B345.

El modelo presenta un comportamiento perfecto para $\gamma < 10^{-1}$, sobre ajustando para valores superiores.

Kernel polinómico Definido por la ecuación 4.40, el kernel polinómico es dependiente de parámetros γ , r (también llamado *coef0*) y d (grado del polinomio, en inglés *degree*).



(a) Curva de validación para kernel polinómico en función de r en área B345. (b) Curva de validación para kernel polinómico en función del grado en área B345.

Fig. 6.20. Curvas de validación para kernel polinómico

Para el caso del parámetro r , el modelo sobre ajusta en casi todo el rango, pero es de tan mínima la diferencia que se puede considerar un comportamiento casi perfecto. En el caso de d , el clasificador es impecable pues, ni sobre ajusta ni sub ajusta.

La figura 6.20a representa un correcto comportamiento del modelo para el rango $r \in [-3, 1, 75]$, sobre ajustando fuera de estos límites. Para 6.20b, el modelo funciona correctamente en $d < 3$, tendiendo a sobre ajustar a medida que se aumenta el grado del kernel.

Kernel lineal El kernel lineal (ecuación 4.39) no requiere de más parámetros excepto C , el parámetro de regularización, por lo tanto, su curva de validación ya se haya representada en 6.18a.

Mejores parámetros SVC para área B345

Kernel RBF Los mejores parámetros encontrados, según métrica F_1 , para $k(\text{ANOVA}) \in [2, 31]$, $\gamma \in [10^{-7}, 10^{-1}]$ y $C \in [1, 100]$ han resultado:

Parámetro	Valor
k (ANOVA)	12
C	20
γ	0.1

Kernel Lineal Los mejores parámetros encontrados, según métrica F_1 , para $k(\text{ANOVA}) \in [2, 31]$ y $C \in [1, 100]$ han resultado:

Parámetro	Valor
k (ANOVA)	12
C	15

Kernel polinómico Los mejores parámetros encontrados, según métrica F_1 , para $k(\text{ANOVA}) \in [2, 31]$ y $C \in [1, 100]$ han resultado:

Parámetro	Valor
k (ANOVA)	12
C	20
r	1
Grado	3
γ	1

AUC-Ovo-B345-Poly		AUC-Ovo-B345-Lineal		AUC-Ovo-B345-RBF	
Clase	AUC	Clase	AUC	Clase	AUC
LX01	1.000000	LX01	1.000000	LX01	1.000000
TC01	1.000000	TC01	1.000000	TC01	1.000000
TC02	1.000000	TC02	1.000000	TC02	1.000000
TC03	1.000000	TC03	1.000000	TC03	1.000000
TC04	1.000000	TC04	1.000000	TC04	1.000000
TC05	1.000000	TC05	1.000000	TC05	1.000000
TC06	1.000000	TC06	1.000000	TC06	1.000000
TC08	1.000000	TC08	1.000000	TC08	1.000000
TC10	0.982638	TC10	0.765972	TC10	0.971630
TC11	0.996768	TC11	0.963289	TC11	0.993925
TC12	1.000000	TC12	1.000000	TC12	1.000000
TC14	1.000000	TC14	1.000000	TC14	1.000000
TC15	1.000000	TC15	1.000000	TC15	1.000000
TC17	0.993755	TC17	0.999905	TC17	0.999905
TC19	1.000000	TC19	1.000000	TC19	1.000000
TC20	1.000000	TC20	1.000000	TC20	1.000000
TC23	1.000000	TC23	1.000000	TC23	1.000000
TC25	1.000000	TC25	1.000000	TC25	1.000000
TC27	1.000000	TC27	1.000000	TC27	1.000000
TC29	1.000000	TC29	1.000000	TC29	1.000000
TC31	1.000000	TC31	1.000000	TC31	1.000000
TC35	1.000000	TC35	1.000000	TC35	1.000000

TABLA 6.13. RESULTADOS AUC PARA DIFERENTES KERNEL
SVM PARA ÁREA B345

Kernel	AUC Macro
Polinómico	0.9988
Lineal	0.9877
RBF	0.9984

TABLA 6.14. AUC MACRO PARA B345

7. CRONOGRAMA DE ACTIVIDADES

Fase	ACTIVIDAD	Febrero				Marzo				Abril				Mayo				Junio			
		Sem. 1	Sem. 2	Sem. 3	Sem. 4	Sem. 1	Sem. 2	Sem. 3	Sem. 4	Sem. 1	Sem. 2	Sem. 3	Sem. 4	Sem. 1	Sem. 2	Sem. 3	Sem. 4	Sem. 1	Sem. 2	Sem. 3	Sem. 4
1	Estudio métodos de extracción de características																				
	Obtención de momentos Hu																				
	Estudio y programación de HoG																				
	Estudio de LBP																				
	Estudio de descriptor de Fourier																				
	Estudio métodos de selección de características																				
	Aplicación y estudio de SFS																				
	Aplicación y estudio de ANOVA F-test																				
2	Estudio modelos de clasificadores																				
	Estudio métricas de rendimiento																				
	Estudio Sklearn API																				
	Aplicación modelo Knn																				
	Estudio resultados Km																				
	Aplicación modelos SVC, NuSVC, LinearSVC																				
	Estudio resultados SVC, NuSVC, LinearSVC																				

Fig. 7.1. Procesos en la fase de extracción y selección de características

BIBLIOGRAFÍA

- [1] J. M. McCarthy, “WHAT IS ARTIFICIAL INTELLIGENCE?” *John McCarthy’s Home Page*, nov. de 2004. [En línea]. Disponible en: https://borgheze.dii.unimi.it/Teaching/AdvancedIntelligentSystems/Old/IntelligentSystems_2008_2009/Old/IntelligentSystems_2005_2006/Documents/Symbolic/04_McCarthy_whatisai.pdf.
- [2] A. L. Samuel, “Some Studies in Machine Learning Using the Game of Checkers,” *IBM J. Res. Dev.*, vol. 44, pp. 206-227, 1959.
- [3] S. Levy, *In the Plex: How Google Thinks, Works, and Shapes Our Lives*. 2011.
- [4] D. Knuth, *The Art of Computer Programming*. 1968.
- [5] M.-K. Hu, “Visual pattern recognition by moment invariants,” *IRE Transactions on Information Theory*, vol. 8, n.º 2, pp. 179-187, 1962. doi: [10.1109/TIT.1962.1057692](https://doi.org/10.1109/TIT.1962.1057692).
- [6] R. K. McConnell, “Method of and apparatus for pattern recognition,” ene. de 1986. [En línea]. Disponible en: <https://www.osti.gov/biblio/6007283>.
- [7] D.-c. He y L. Wang, “Texture Unit, Texture Spectrum, And Texture Analysis,” *IEEE Transactions on Geoscience and Remote Sensing*, vol. 28, n.º 4, pp. 509-512, 1990. doi: [10.1109/TGRS.1990.572934](https://doi.org/10.1109/TGRS.1990.572934).
- [8] J. Frost, *How F-tests work in Analysis of Variance (ANOVA)*, 2020. [En línea]. Disponible en: <https://statisticsbyjim.com/anova/f-tests-anova/>.
- [9] D. M. W. Powers, “Evaluation: from precision, recall and F-measure to ROC, informedness, markedness and correlation,” 2020. doi: [10.48550/ARXIV.2010.16061](https://arxiv.org/abs/2010.16061). [En línea]. Disponible en: <https://arxiv.org/abs/2010.16061>.
- [10] T. Saito y M. Rehmsmeier, “The Precision-Recall Plot Is More Informative than the ROC Plot When Evaluating Binary Classifiers on Imbalanced Datasets,” *PLOS ONE*, vol. 10, n.º 3, pp. 1-21, mar. de 2015. doi: [10.1371/journal.pone.0118432](https://doi.org/10.1371/journal.pone.0118432). [En línea]. Disponible en: <https://doi.org/10.1371/journal.pone.0118432>.
- [11] D. Mery, *Python3 implementation of Computer Vision and Pattern Recognition library Balu*, <https://github.com/computervision-xray-testing/pybalu>, 2020.
- [12] F. Pedregosa et al., “Scikit-learn: Machine Learning in Python,” *Journal of Machine Learning Research*, vol. 12, pp. 2825-2830, 2011.
- [13] L. Buitinck et al., “API design for machine learning software: experiences from the scikit-learn project,” en *ECML PKDD Workshop: Languages for Data Mining and Machine Learning*, 2013, pp. 108-122.

ANEXO A

7.1. Creación de dataset y extracción de características