Universidad Politécnica
de Madrid

**Escuela Técnica Superior de
Ingenieros Informáticos**

GRADO en INGENIERÍA INFORMÁTICA

Trabajo Fin de GRADO

Clasificación Inferida mediante Datos Departamento del Director del TFG

Autor: MIGUEL MUÑOZ RAMOS
Tutor(a): JUAN ANTONIO FERNÁNDEZ DEL POZO SALAMANCA

Madrid, JUNIO 2025

Este Trabajo Fin de GRADO se ha depositado en la ETSI Informáticos de la Universidad Politécnica de Madrid para su defensa.

Trabajo Fin de GRADO
GRADO en INGENIERÍA INFORMÁTICA

Título: Clasificación Inferida mediante Datos Departamento del Director del TFG

JUNIO 2025

Autor: MIGUEL MUÑOZ RAMOS
Tutor: JUAN ANTONIO FERNÁNDEZ DEL POZO SALAMANCA
DEPARTAMENTO DE INTELIGENCIA ARTIFICIAL
Escuela Técnica Superior de Ingenieros Informáticos
Universidad Politécnica de Madrid

Índice general

Índice de Figuras

Índice de Tablas

Índice de Listings

Resumen

Se realiza un análisis estadístico, exploración de los datos e información disponible en el **Archivo Digital de la UPM**, y un modelo de clasificación centrado en la variable Departamento del director de los trabajos de fin de grado. Se realizará una descripción de la estructura del **AD-UPM**, las variables que modelizan el título y el resumen. Se tomará una muestra de un periodo de al menos dos años y diversas Escuelas, Departamentos, etc. Realizaremos un análisis exploratorio para generar hipótesis y proponer algunas técnicas de clasificación de los trabajos de fin de grado.

Se trata de desarrollar una metodología para extraer conocimiento del **AD-UPM** y diseñar una propuesta de análisis de los datos: Adquisición de datos, modelización, inferencia y síntesis de conclusiones. Documentar las variables y establecer los objetivos para la propuesta de análisis. Planificar el proceso de preparación de los datos, construcción y evaluación del modelo.

Se debe definir las estructuras de datos para representar la información de interés que podemos extraer del **AD-UPM** y de los trabajos de fin de grado en particular. Completamos una muestra significativa con el fin de aprender un modelo probabilístico. Mostramos una descriptiva de los datos, generando una estructura de grupos de documentos para facilitar el aprendizaje de un clasificador. Finalmente estudiamos el rendimiento (sensibilidad, especificidad,....) del modelo de clasificación.

Palabras Clave: Minería de datos, Minería de textos, Clustering, Análisis estadístico

Abstract

A statistical analysis is carried out, exploring the data and information available in the **Digital Archive of the UPM**, and a classification model focused on the variable Department of the director of the final degree theses. A description will be made of the structure of the **AD-UPM**, the variables that model the title and the abstract. A sample will be taken from a period of at least two years and various Schools, Departments, etc. We will carry out an exploratory analysis to generate hypotheses and propose some classification techniques for final degree theses.

The aim is to develop a methodology to extract knowledge from the **AD-UPM** and to design a proposal for data analysis: Data acquisition, modelling, inference and synthesis of conclusions. Document the variables and set the objectives for the analysis proposal. Plan the process of data preparation, model building and evaluation.

Data structures must be defined to represent the information of interest that we can extract from the **AD-UPM** and from the final degree projects in particular. We complete a significant sample in order to learn a probabilistic model. We show a descriptive summary of the data, generate a structure of groups of documents to facilitate the learning of a classifier. Finally, we study the performance (sensitivity, specificity,....) of the classification model.

Keywords: Data mining, Text mining, Clustering, Statistical analysis

Capítulo 1

Introducción

En este capítulo veremos la motivación del proyecto, el objetivo y el alcance del mismo y por último la estructura de la memoria con un breve resumen de los distintos capítulos.

1.1. Motivación del proyecto

En la era digital, la cantidad de información almacenada en repositorios académicos ha crecido exponencialmente, lo que plantea un desafío en la organización, recuperación y análisis de estos datos. En particular, el **Archivo Digital de la UPM (AD-UPM)** contiene una gran colección de Trabajos de Fin de Grado (TFG), entre otros muchos documentos, en los cuales vamos a centrar nuestro estudio.

Este Trabajo de Fin de Grado surge con el objetivo de explorar, analizar y estructurar la información contenida en el **AD-UPM** mediante un enfoque basado en ciencia de datos y técnicas de clasificación. A través de un estudio detallado de las variables que modelizan los títulos y resúmenes de los TFG, se busca identificar patrones, generar hipótesis y diseñar una metodología que permita una clasificación eficiente de los trabajos en función del Departamento al que pertenecen.

La importancia de este trabajo radica en la necesidad de optimizar el acceso y la organización de los TFGs en el **AD-UPM**, facilitando la búsqueda de información relevante tanto para estudiantes como para investigadores. Además, la implementación de análisis exploratorio de datos (EDA) y modelos de clasificación permitirá evaluar la adecuación de distintas metodologías dentro del campo de la minería de datos y la inteligencia artificial.

1.2. Objetivo y alcance

El objetivo principal de este Trabajo de Fin de Grado es desarrollar una metodología que permita extraer, analizar y clasificar los TFG's almacenados en el **Archivo Digital de la UPM (AD-UPM)**. Para ello, se propone aplicar técnicas de ciencia de datos y clasificación con el fin de estructurar la información disponible y mejorar su organización. Se busca analizar la estructura del **AD-UPM**, identificando las principales variables que modelizan los títulos y resúmenes de los TFG's, con el propósito de comprender su distribución y características fundamentales.

A lo largo del estudio, se llevará a cabo un análisis exploratorio de los datos para detectar patrones relevantes y generar hipótesis que faciliten su clasificación. Con esta base, se diseñará e implementará un modelo que asigne automáticamente cada trabajo a su correspondiente Departamento, evaluando distintas estrategias de modelización y selección de características. Además, se examinará el rendimiento del modelo mediante métricas como la sensibilidad y la especificidad, entre otras, con el fin de validar su precisión y eficacia.

El alcance de este trabajo se limita al análisis de los TFG's disponibles en el **AD-UPM**, tomando como referencia una muestra de documentos publicados en un período de al menos dos años y pertenecientes a distintas Escuelas y Departamentos. En el desarrollo del proyecto se contemplará la adquisición y preprocesamiento de los datos, su exploración mediante herramientas estadísticas y la implementación de algoritmos de clasificación para optimizar su organización.

Los resultados obtenidos pueden servir como base para futuras optimizaciones en la gestión de los TFG's dentro del **AD-UPM**, facilitando su acceso y organización de manera más eficiente.

1.3. Estructura del documento

La memoria se estructura en varios capítulos que abordan de manera detallada los distintos aspectos del proyecto. La organización es la siguiente:

Capítulo 2, Fundamentos, Estado del Arte y alineamiento con la ODS: En este capítulo se abordan los principios teóricos y estudios previos relacionados con el análisis de datos y la clasificación automática. Se revisan los conceptos clave de la minería de datos, el aprendizaje automático y las metodologías de análisis exploratorio. Además, se analiza el alineamiento del proyecto con los Objetivos de Desarrollo Sostenible (ODS).

Capítulo 3, Modelo de datos: Este capítulo detalla la estructura de los datos utilizados en el estudio. Se estudia el modelo de datos del **Archivo Digital de la UPM (AD-UPM)**, el esquema de organización de la información y las variables seleccionadas para el análisis. También se abordan las estrategias de preprocesamiento y limpieza de datos.

Capítulo 4, Extracción y análisis: En este capítulo se describe el proceso de adquisición y exploración de datos. Se presentan las técnicas utilizadas para la extracción de la muestra y el análisis exploratorio de datos (EDA), incluyendo visualizaciones y estadísticas descriptivas que permiten identificar patrones y tendencias relevantes.

Capítulo 5, Algoritmos y pruebas: Este apartado se centra en la implementación del modelo de clasificación. Se preparan los datos mediante técnicas de minería de textos para transformar los documentos en información estructurada. Se explican los algoritmos considerados, las métricas utilizadas para evaluar su desempeño y las pruebas realizadas para ajustar los parámetros y optimizar los resultados.

Capítulo 6, Análisis de resultados: Se presentan los resultados obtenidos tras la aplicación del modelo de clasificación. Se analizan métricas de rendimiento como la precisión, sensibilidad y especificidad del modelo, comparando los resultados con diferentes configuraciones y enfoques metodológicos.

Capítulo 7, Conclusiones: En este capítulo se realiza una evaluación detallada del cumplimiento de los objetivos del proyecto, se presentan propuestas para futuras investigaciones, una reflexión personal sobre la experiencia de realización del TFG, y se analiza el impacto del trabajo en relación con los Objetivos de Desarrollo Sostenible (ODS).

Capítulo 2

Fundamentos y Estado del Arte

En este capítulo veremos los fundamentos teóricos más importantes relacionados con este trabajo.

2.1. Minería de datos y clustering

En este trabajo se desarrollará un modelo de clasificación de datos basado en técnicas estadísticas y de minería de datos. A continuación, se presentan los fundamentos teóricos que sustentan este estudio.

2.1.1. Minería de datos

La minería de datos [1] es un proceso en el que se analizan grandes volúmenes de datos con el fin de hallar patrones que expliquen su comportamiento en un contexto determinado, según la definición de la Real Academia Española (RAE). Este proceso forma parte del campo más amplio de la ciencia de datos y se basa en la aplicación de técnicas estadísticas, matemáticas y de inteligencia artificial para extraer información útil y conocimiento a partir de datos en bruto.

Para lograr estos objetivos, la minería de datos emplea diversas técnicas como la clasificación, que asigna etiquetas a los datos en función de características predefinidas; la agrupación (clustering), que organiza los datos en grupos con características similares sin necesidad de etiquetas previas; la extracción de reglas de asociación, que busca relaciones entre variables dentro de un conjunto de datos; y la detección de anomalías, que identifica valores atípicos o patrones inusuales que podrían indicar errores, fraudes o eventos poco comunes.

En la actualidad, la minería de datos es una disciplina fundamental en múltiples áreas, incluyendo el comercio electrónico, la salud, la seguridad informática y la ingeniería.

2.1.2. Clustering

El clustering [1] es una técnica de minería de datos para hacer análisis exploratorio que consiste en agrupar elementos dentro de un conjunto de datos según su similitud, sin necesidad de contar con etiquetas predefinidas. Este proceso permite

identificar estructuras ocultas en los datos y descubrir patrones subyacentes, lo que lo convierte en una herramienta fundamental en la exploración y análisis de grandes volúmenes de información.

Desde un punto de vista técnico, el clustering se basa en medidas de similitud o distancia, como la distancia euclídea, para organizar los datos en grupos o clústeres. Entre los algoritmos más utilizados para este propósito destacan **k-means** (clustering particional), que forma grupos minimizando la variabilidad interna de cada clúster; **DBSCAN** (clustering de densidad), que detecta agrupaciones de distinta densidad y es robusto ante valores atípicos; y jerárquico, que construye una estructura en forma de árbol para visualizar las relaciones entre los datos.

La validación del clustering se basa en usar varios algoritmos, conocimiento experto del dominio para interpretar los grupos y métricas de rendimiento.

Por todo esto, el clustering es una técnica clave dentro de la ciencia de datos y la inteligencia artificial.

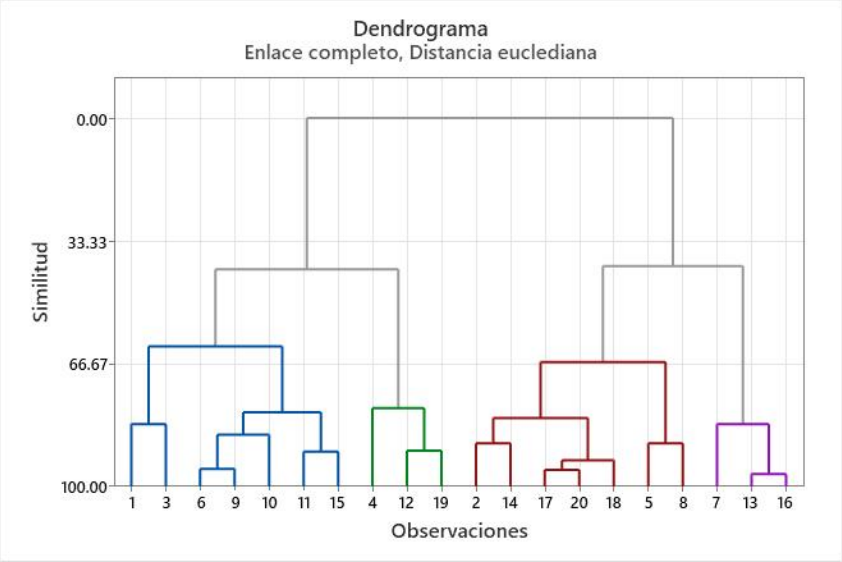

Figura 2.1: Dendrograma con método de enlace completo y distancia euclídea.

2.2. Técnicas de clasificación de datos

La clasificación es una de las tareas clave en la minería de datos y se utiliza ampliamente en el análisis de textos. Algunas de las técnicas de clasificación más importantes que se abordan en esta memoria son las siguientes [1]:

2.2.1. Árboles de decisión

Los Árboles de Decisión [1] representan una estrategia popular utilizada en clasificación, ya que organizan la información en una estructura jerárquica en la que cada nodo corresponde a una decisión basada en un atributo específico, y las hojas representan las categorías finales. Una de sus grandes ventajas es su facilidad de interpretación y visualización, lo que los convierte en una opción atractiva para la toma de decisiones en entornos donde se requiere comprender el proceso de clasificación.

Además, no requieren normalización de datos y pueden manejar tanto variables numéricas como categóricas. Sin embargo, si no se implementa un mecanismo adecuado de poda, los árboles de decisión pueden ser propensos al sobreajuste, lo que disminuye su capacidad de generalización a nuevos datos.

También, son sensibles a pequeños cambios en el conjunto de entrenamiento, lo que puede afectar su estabilidad y generar estructuras diferentes en cada ejecución. Su eficiencia también puede verse comprometida cuando se trabaja con grandes conjuntos de datos y múltiples atributos.

Para construir un árbol de decisión, se utilizan métricas como la Ganancia de información basada en la entropía:

$$H(S) = -\sum_{i=1}^{n} p_i \log_2 p_i$$

Donde S es el conjunto de datos sobre el que se calcula la entropía, p_i es la probabilidad de que un elemento de S pertenezca a la i-ésima clase y H(S) representa la entropía o medida de impureza del conjunto S.

Y la Ganancia de información se define como:

$$\text{Ganancia}(S, A) = H(S) - \sum_{v \in A} \frac{|S_v|}{|S|} H(S_v)$$

Donde H(S) es la entropía del conjunto de datos y H(S_v) es la entropía después de dividir según el atributo A. La ganancia se maximiza con el mejor árbol de decisión.

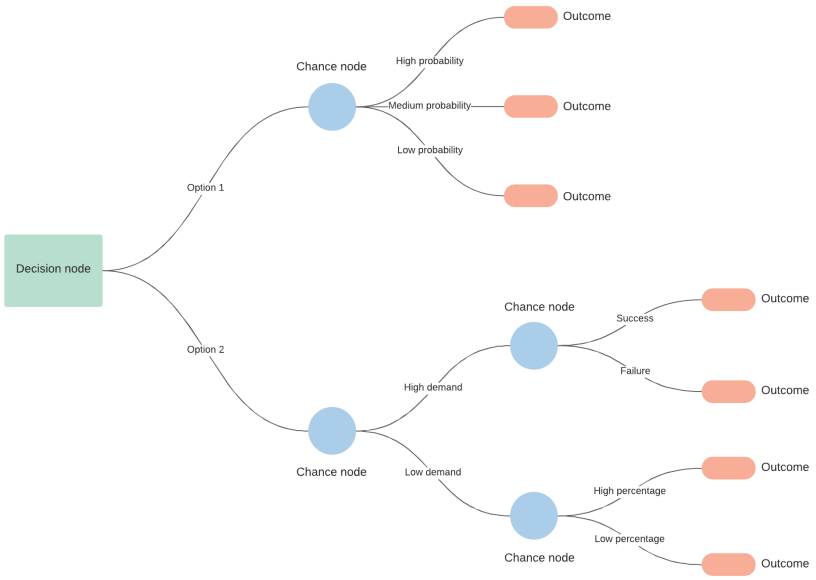

Figura 2.2: Esquema básico de un árbol de decisión

2.2.2. Random Forest

El algoritmo de *Random Forest* [1] es un método de ensamblado basado en la construcción de múltiples árboles de decisión independientes, entrenados cada uno sobre

una muestra diferente del conjunto de datos y evaluando en cada nodo solo un subconjunto aleatorio de características. Para clasificar una nueva instancia, cada árbol "vota" por una clase, y la predicción final es la clase mayoritaria (votación por mayoría).

Gracias a este ensamblado, resulta especialmente eficaz en espacios de alta dimensión y frente a variables correlacionadas, y además ofrece de manera natural una medida de la importancia de cada atributo. Sin embargo, esa misma agregación de modelos hace que su interpretabilidad global sea mucho menor que la de un solo árbol, y el entrenamiento y almacenamiento de decenas o centenas de árboles implica un coste computacional muy superior.

Matemáticamente, si disponemos de T árboles que producen predicciones $h_i(x)$ para una instancia x, la salida del bosque es

$$\hat{y} = \arg\max_{c} \sum_{i=1}^{T} \mathbb{I}\{h_i(x) = c\},$$

donde $\mathbb{I}\{\cdot\}$ es la función indicadora que vale 1 si el árbol i predice la clase c.

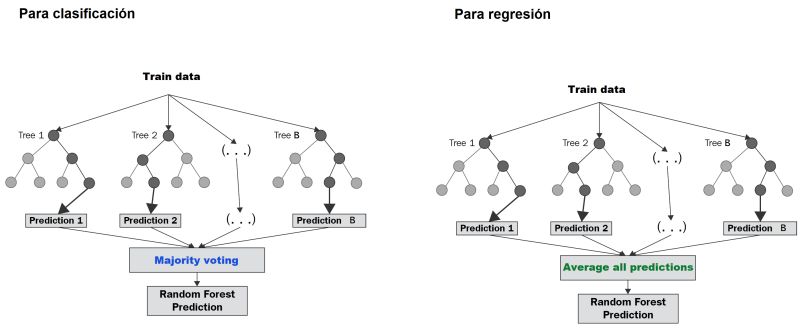

Figura 2.3: Esquema básico de random forest

La imagen 2.3 muestra cómo, tanto en clasificación como en regresión, un Random Forest combina las predicciones de múltiples árboles entrenados sobre distintas muestras de los datos. En clasificación, cada árbol "vota" por una etiqueta y se elige la más frecuente, mientras que en regresión se promedian sus salidas numéricas.

2.2.3. Máquinas de Soporte Vectorial (SVM)

Las Máquinas de Soporte Vectorial (SVM) [1] son un método de clasificación que busca encontrar un hiperplano óptimo que separe los datos en distintas clases maximizando el margen entre ellos. Este enfoque resulta especialmente eficiente en espacios de alta dimensión, habituales en minería de textos, y es adecuado para conjuntos de datos pequeños con una baja cantidad de ruido.

No obstante, su aplicación en grandes volúmenes de datos puede ser costosa en términos computacionales, ya que el proceso de optimización se vuelve más complejo. Además, el rendimiento depende en gran medida del ajuste de hiperparámetros, como el factor de regularización C, que equilibra la amplitud del margen y los errores de

clasificación— y de la elección y parametrización del kernel (lineal, polinómico, RBF, sigmoide...), que define la transformación de los datos a un espacio donde sean separables. Otro inconveniente es que su sensibilidad a datos ruidosos puede afectar su capacidad de generalización.

Matemáticamente, el problema de optimización en SVM se define como:

$$\min_{w,b} \frac{1}{2}\|w\|^2$$

Sujeto a:

$$y_i(w \cdot x_i + b) \geq 1, \quad \forall i$$

Donde w es el vector normal al hiperplano, b es el sesgo, b/|w| es el offset del origen de coordenadas según la dirección de w,x_i son los puntos de datos e y_i = 1, -1 son sus etiquetas de clase.

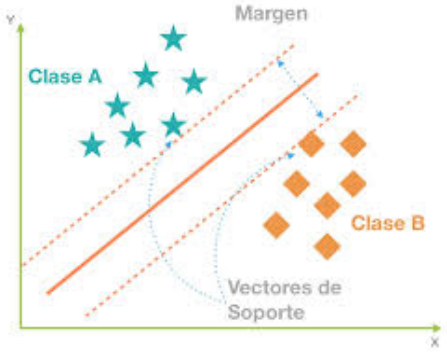

Figura 2.4: Diagrama básico MSV

El diagrama muestra un hiperplano (línea continua) que separa dos clases, con dos líneas discontinuas paralelas que marcan el margen máximo entre ellas. Los puntos que quedan justo sobre esos márgenes son los vectores de soporte, pues son los datos más cercanos al hiperplano y determinan su posición óptima. La SVM elige así la separación que maximiza esa distancia, garantizando mayor robustez entre clases.

2.2.4. k-Nearest Neighbors (k-NN)

El algoritmo k-Nearest Neighbors (k-NN) [1] se basa en la idea de clasificar una nueva instancia en función de sus k vecinos más cercanos dentro del espacio de características, empleando generalmente la distancia euclídea como medida de proximidad. Su simplicidad es una de sus principales ventajas, ya que es fácil de entender e implementar. Además, no requiere un proceso de entrenamiento explícito, ya que la clasificación se realiza directamente a partir de los datos almacenados.

Sin embargo, cuando se trabaja con grandes volúmenes de datos, el cálculo de distancias para cada nueva instancia puede volverse ineficiente y costoso en términos de tiempo de procesamiento.

Asimismo, su desempeño es altamente dependiente de la selección del valor de k, y si los datos no están bien escalados, los resultados pueden verse afectados. También es sensible a la presencia de características irrelevantes o ruido en los datos, lo que puede disminuir su precisión.

La distancia euclídea, utilizada comúnmente en k-NN, se define como:

$$d(x, y) = \sqrt{\sum_{i=1}^{n}(x_i - y_i)^2}$$

Donde x e y son dos puntos en un espacio de características n-dimensional. Este algoritmo en alta dimensión no es eficaz y con muchos puntos la complejidad es muy alta.

2.2.5. Naïve Bayes Classifier

El algoritmo Naïve Bayes [1] es una técnica de clasificación probabilística basada en el Teorema de Bayes, que asume independencia condicional entre las características de los datos. Su principal ventaja es su rapidez y eficiencia, incluso cuando se trabaja con grandes volúmenes de información, lo que lo hace particularmente útil en tareas de clasificación de textos, como el filtrado de spam y el análisis de sentimientos.

Además, no requiere grandes cantidades de datos para entrenarse y puede aprender con datos incompletos, lo que lo convierte en una opción ideal cuando se dispone de conjuntos de datos limitados. Sin embargo, una de sus principales desventajas es que la suposición de independencia entre características rara vez se cumple en la práctica, lo que puede afectar negativamente su precisión. Asimismo, es sensible a variables altamente correlacionadas, lo que puede generar errores en la clasificación.

Matemáticamente, el clasificador Naïve Bayes calcula la probabilidad de que un elemento pertenezca a una clase C_k dada una observación X=(x_1,x_2,...,x_n), usando la siguiente fórmula:

$$P(C_k|X) = \frac{P(X|C_k)P(C_k)}{P(X)}$$

Dado que se asume independencia condicional entre las características, la probabilidad conjunta dada la clase se descompone como:

$$P(X|C_k) = \prod_{i=1}^{n} P(x_i|C_k)$$

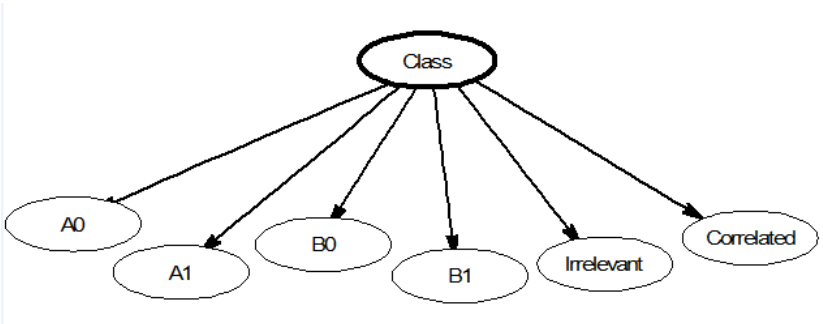

Figura 2.5: Diagrama de un clasificador Naive Bayes representado como una red Bayesiana

En este diagrama, los arcos indican las dependencias condicionales entre la variable objetivo “Class” y cada predictor. Para lograr un modelo más eficiente, es esencial

realizar una selección de variables: eliminar las variables irrelevantes o redundantes. Además, una de las fortalezas del clasificador Naïve Bayes es que puede trabajar tanto con predictores discretos como continuos.

2.2.6. Regresión Logística

La Regresión Logística [1] es un modelo estadístico utilizado principalmente para clasificación binaria. Emplea la función sigmoide para modelar la probabilidad de pertenencia a una determinada clase.

Su principal ventaja radica en su facilidad de interpretación y entrenamiento, lo que la convierte en una opción eficiente cuando se manejan grandes volúmenes de datos. Además, es menos propensa al sobreajuste en comparación con modelos más complejos, lo que la hace adecuada en escenarios donde se busca un balance entre precisión y simplicidad. No obstante, su aplicación es limitada en problemas donde los datos son linealmente separables, a menos que se utilicen transformaciones adicionales.

Asimismo, no es capaz de capturar relaciones complejas entre variables sin realizar modificaciones adicionales, lo que restringe su capacidad predictiva en casos donde la interacción entre atributos juega un papel clave en la clasificación. La regresión logística modela la probabilidad de una clase como:

$$P(Y = 1|X) = \frac{1}{1 + e^{-(w_0 + w_1 x_1 + w_2 x_2 + \cdots + w_n x_n)}}$$

Donde w_0 es el sesgo y w_i son los coeficientes de los atributos x_i. La función sigmoide utilizada para transformar la salida en una probabilidad se define como:

$$\sigma(z) = \frac{1}{1 + e^{-z}}$$

Este modelo se entrena optimizando los coeficientes mediante la maximización de la verosimilitud [1], usando algoritmos como el descenso de gradiente [1].

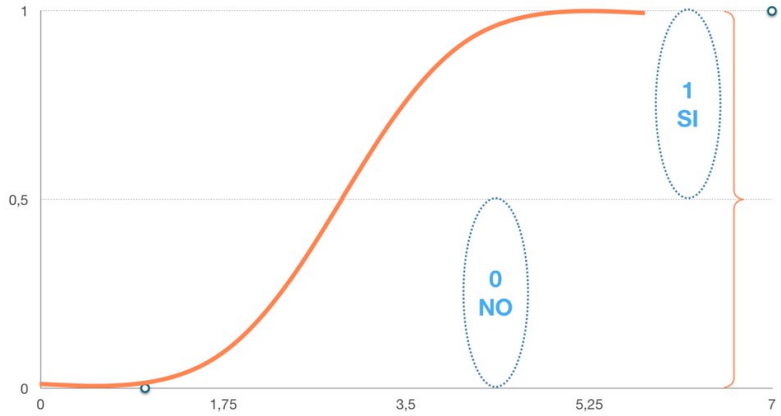

Figura 2.6: Diagrama de un clasificador de regresión logística

2.3. Tipos de documentos y su relevancia en el análisis

En el ámbito del procesamiento automático de textos con distintas clases de documentos. Cada clase de documento posee una estructura, un lenguaje y una finalidad distintos, lo que afecta directamente a las técnicas de análisis y a los modelos que se deben aplicar.

2.3.1. Documentos académicos, clínicos y periodísticos

Los documentos académicos [1] se caracterizan por su lenguaje técnico, su estructura formal y su orientación expositiva. Presentan una gran densidad de conceptos, citas y terminología específica del campo de estudio. Su análisis suele centrarse en tareas como la extracción de conceptos clave, la identificación de temas o la clasificación por disciplinas.

Los textos clínicos [1], por otro lado, incluyen historiales médicos, informes diagnósticos o registros electrónicos de salud. Estos documentos requieren un tratamiento especializado debido a la terminología médica, la codificación estandarizada (por ejemplo, CIE-10) y la necesidad de mantener la confidencialidad de la información.

Los artículos periodísticos [1], especialmente las noticias, ofrecen un lenguaje más accesible, pero también presentan sesgos, subjetividad o estructuras narrativas que deben ser tenidas en cuenta. Su análisis puede orientarse a la detección de eventos, el análisis de sentimientos o la categorización temática.

2.3.2. Código fuente, redes sociales y textos jurídicos

El código fuente [1] representa un caso especial de documento, donde el contenido tiene una sintaxis formal definida por lenguajes de programación. El análisis de estos textos se vincula a tareas como la clasificación de fragmentos de código, la detección de patrones, el análisis de dependencias o incluso la traducción automática entre lenguajes.

Las redes sociales generan textos breves, informales y frecuentemente acompañados de elementos multimedia. Aquí es común trabajar con ruido lingüístico, abreviaturas, emoticonos y lenguaje coloquial. Estas características implican preprocesamientos específicos y modelos robustos ante variabilidad lingüística.

Los documentos jurídicos [1], como sentencias, leyes o contratos, destacan por su estructura normativa, uso de fórmulas legales y ambigüedad controlada. El análisis jurídico automatizado se enfrenta a retos de desambiguación semántica, extracción de entidades y reconocimiento de estructuras legales.

2.4. Análisis documental: de técnicas tradicionales a modelos generativos

La evolución en el análisis de documentos ha pasado de técnicas basadas en reglas y modelos estadísticos simples a sistemas mucho más complejos capaces de entender

la semántica y el contexto de los textos. Esta sección aborda las principales estrategias actuales y su aplicación en tareas relevantes para el proyecto.

2.4.1. Procesamiento lingüístico y estructural

El análisis de un documento comienza con procesos básicos como la tokenización, la lematización, la eliminación de *stopwords* y la normalización del texto [2]. Estos pasos forman parte del análisis léxico, que permite representar el contenido de manera estructurada y homogénea.

A continuación, se emplean analizadores sintácticos (*parsers*) [2] que identifican la estructura gramatical de las oraciones. Esto permite extraer relaciones entre palabras y generar árboles sintácticos [1] que pueden ser útiles para modelos más avanzados. En algunos casos, también se realiza análisis semántico, mediante el cual se identifican entidades nombradas, relaciones y significados implícitos.

2.4.2. Clasificación, predicción y modelos generativos

Una vez representados y estructurados los documentos, se pueden aplicar técnicas de clasificación supervisada o no supervisada para agruparlos o etiquetarlos. Modelos como SVM, k-NN o Naïve Bayes [1] han sido ampliamente utilizados en este campo, pero en la actualidad los modelos basados en transformadores como BERT o GPT-4 [2] están marcando la diferencia.

Modelos generativos como GPT permiten no solo clasificar, sino también realizar tareas de predicción contextual, traducción automática, resumen de documentos, respuesta a preguntas y generación de texto coherente. Estos modelos han sido entrenados con enormes cantidades de datos y pueden ser adaptados a tareas específicas mediante *fine-tuning* o *few-shot learning* [2].

Capítulo 3

Modelo de datos

3.1. Origen de los datos

Para obtener los datos que utilizaremos en nuestros modelos, vamos a usar como fuente la página web del archivo digital de la UPM. Dentro de la página principal del **Archivo Digital de la UPM (AD-UPM)** hacemos una búsqueda para filtrar por TFG, ya que es el tipo de trabajo que nos interesa analizar.

El enlace al repositorio con los trabajos ya filtrados por TFG sería el siguiente: Repositorio TFGs. Vamos a obtener una muestra inicial de 500 de un total de más de 11000 trabajos a fecha de 01/06/2025.

En uno de estos TFGs tenemos varios elementos de los que podemos extraer información que pueden ser datos útiles así como el propio trabajo completo. En el siguiente apartado veremos cuáles de estos datos son los que vamos a obtener.

3.2. Variables de los datos

El siguiente enlace https://oa.upm.es/88613/ nos lleva a un ejemplo de la página de la que vamos a obtener los elementos.

En esta página podemos observar los campos que aparecen para cada uno de los trabajos. De todos ellos, los que vamos a elegir son los siguientes: Título, Autor/es, Director/es, Palabras clave, Escuela, Departamento y Resumen.

A parte de todos estos elementos, en nuestros datos iniciales también guardaremos el enlace de cada uno de los trabajos de los que hemos obtenido datos, para tener trazabilidad si ocurre algún error.

Desde el punto de vista de la tarea de clasificación, procesaremos el texto extraído del Archivo Digital como un dataset etiquetado que nos permite hacer un clasificador mediante aprendizaje automático supervisado.

3.3. Preparación de los datos

Partiendo de los datos en crudo, que son los obtenidos mediante el proceso de web scraping, explicado en el capítulo 4, y con los campos que queríamos ya obtenidos en

un archivo .txt como en la imagen 3.1.

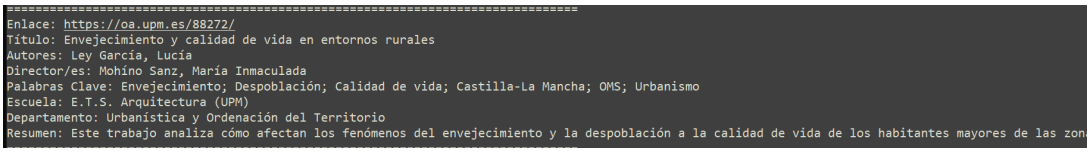

Figura 3.1: Ejemplo de una entrada de datos de un TFG del AD-UPM

Para preparar los datos crudos de entrada, se desarrolló un script en Python que realiza una serie de pasos de limpieza, extracción y preprocesamiento textual sobre los datos originales almacenados en el archivo `Datos_raw.txt`. A continuación, se detallan las fases principales del procesamiento:

1. **Lectura del archivo de datos crudos**:

   Se abre y lee el contenido completo del archivo `Datos_raw.txt`, que contiene registros de trabajos académicos separados por una línea de 80 signos de igual (=). Cada bloque de texto representa un registro distinto.

2. **Extracción estructurada de campos mediante expresiones regulares (A.5)**:

   A partir del texto crudo, se utilizan expresiones regulares [1] para identificar y extraer de cada registro información clave como: el enlace, el título, los autores, los directores, las palabras clave, la escuela, el departamento y el resumen. Los registros extraídos se almacenan en una estructura tipo tabla (`DataFrame`) utilizando la biblioteca `pandas` [3].

3. **Preprocesamiento del texto de los resúmenes (A.6)**:

   Con el fin de preparar el texto para un posterior análisis, se aplica el siguiente flujo de preprocesamiento sobre el campo Resumen:

   - Conversión de todo el texto a minúsculas.
   - Eliminación de tildes y caracteres especiales (como la letra "ñ"), normalizando el texto con `unidecode` [4].
   - Eliminación de signos de puntuación y números.
   - Tokenización [1] del texto (división en palabras individuales).
   - Eliminación de palabras vacías en español que no aportan significado relevante (como "de", "la", etc.).
   - Conservación únicamente de palabras alfabéticas de longitud superior a dos caracteres.
   - Aplicación de stemming [2] (reducción de las palabras a su raíz) utilizando el algoritmo de Snowball para español.

4. **Agrupación departamentos**: Con el objetivo de simplificar la tarea y tener un número de departamentos manejable, se usa la función (A.8) para agrupar departamentos según el campo. Después de agrupar los departamentos los trabajos quedan clasificados como aparece en A.1

5. **Generación y guardado de archivos procesados (A.7)**:

Finalmente, los datos procesados se almacenan en distintos archivos:

- `Datos_preprocesados.csv`: contiene todos los campos extraídos y el texto procesado.
- `dataset_para_modelo.csv`: incluye únicamente el resumen procesado y el departamento.
- `tokens_para_modelo.txt`: cada línea corresponde a la lista de tokens del resumen de un trabajo, separados por espacios.

6. **Notas adicionales**:

   Durante el procesamiento, se utilizan las librerías `nltk` [5] y `pandas` [3], y el programa se encarga de descargar automáticamente los recursos necesarios de `nltk` [5] (como los "stopwordsz el "tokenizer").

Al finalizar la ejecución, el sistema confirma que los datos han sido procesados y guardados correctamente mediante un mensaje en consola.

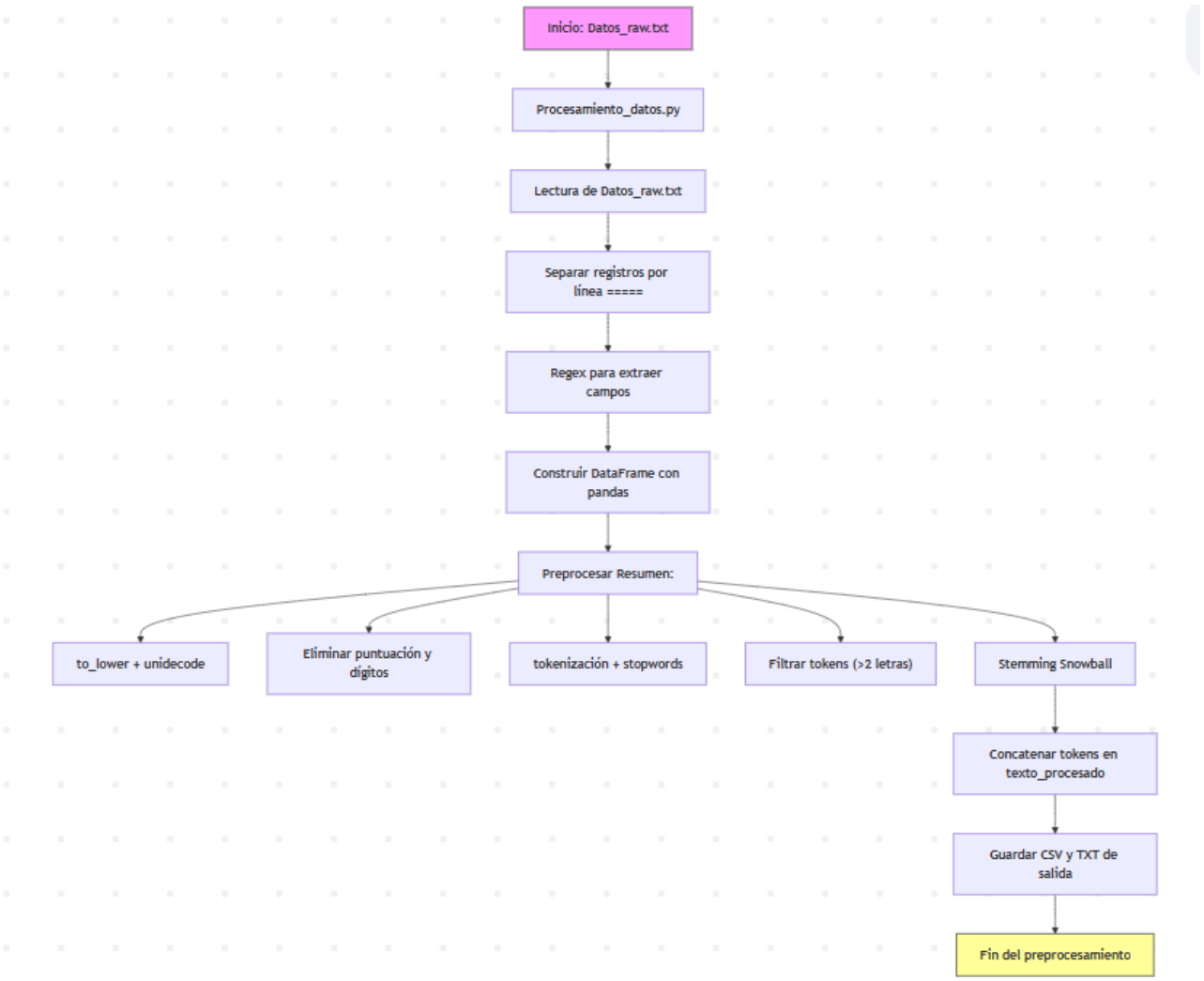

Figura 3.2: Diagrama del procesamiento de datos

Capítulo 4

Análisis y extracción de datos

En este capítulo se describe el proceso seguido para la adquisición de datos mediante técnicas de *web scraping*, así como la descripción del conjunto de resultados obtenidos. Este procedimiento es necesario para construir una muestra real de Trabajos de Fin de Grado (TFG) desde el **Archivo Digital de la UPM (AD-UPM)**, sobre el cual se aplicarán las posteriores técnicas de análisis y clasificación.

4.1. Web scraping del AD-UPM

Para recopilar los metadatos de los TFGs de forma automática se empleó un proceso en dos fases, implementado mediante dos scripts en Python.

4.1.1. Extracción de enlaces

El primer paso consiste en generar la lista de URL de los trabajos. Para ello, el script `SCRAPPER_ENLACES_TFG.py` (A.1) recorre en bloques de 25 resultados las páginas de búsqueda de TFGs del AD-UPM, realizando peticiones HTTP con `requests` [6] y parseando el HTML con `BeautifulSoup` [7]. Se filtran las etiquetas `<a>` cuyo atributo `href` cumple el patrón de identificador numérico de un TFG, se normalizan a URL absolutas y se van acumulando en memoria hasta alcanzar 500 enlaces únicos. Finalmente, se vuelcan en el fichero `enlaces.txt` para su uso posterior. Se puede ver el diagrama de flujo en la imagen 4.1

4.1.2. Descarga de metadatos

Con la lista de URL generada, el segundo script `SCRAPPER_DATOS_TFG.py` itera cada enlace, vuelve a usar `requests`[6]/`BeautifulSoup`[7] y extrae los campos de interés (título, autores, directores, palabras clave, escuela, departamento, resumen) mediante funciones auxiliares (`get_text_by_class` (A.4), `clean_text` (A.2), `extract_directors` (A.3)). Cada registro se escribe en `Datos_raw.txt`, separado por una línea de 80 signos “=” para facilitar su posterior parsing. Se incorpora un retardo de un segundo entre peticiones para respetar la carga del servidor. Se puede observar el diagrama de flujo en la imagen 4.2

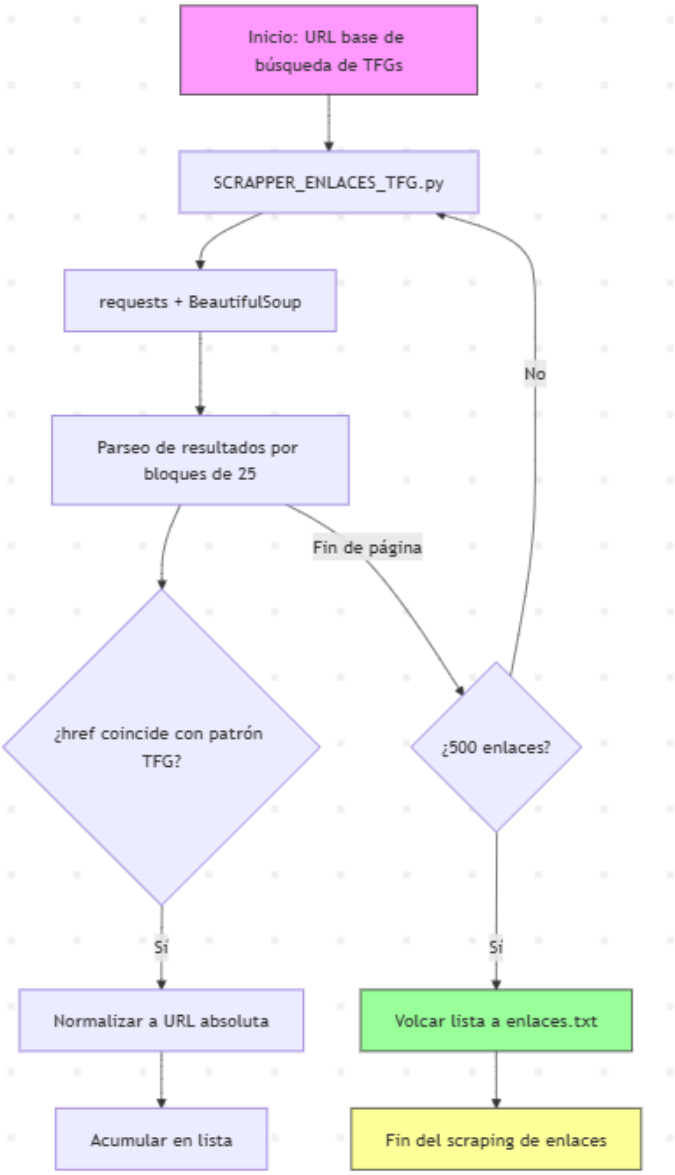

Figura 4.1: Diagrama de flujo del scrapping de los enlaces

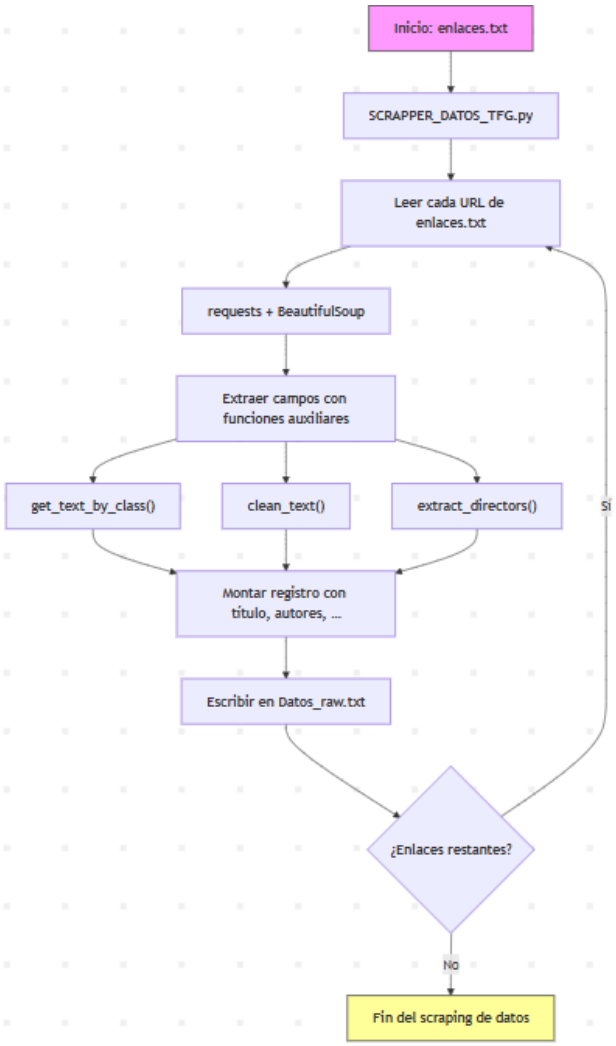

Figura 4.2: Diagrama de flujo del scrapping de los datos

Capítulo 5

Algoritmos escogidos y entrenamiento

A continuación, se detallan las técnicas de representación estructurada de textos y los algoritmos de clasificación y agrupamiento seleccionados para abordar el problema de predicción del Departamento del director de un Trabajo de Fin de Grado (TFG) a partir de sus descripciones textuales. Se describe además el flujo de entrenamiento repetido, diseñado para obtener resultados más robustos, junto con las métricas empleadas para evaluar el rendimiento medio de los modelos.

5.1. Preparación del conjunto de datos

Con el objetivo de proporcionar una base estadística más sólida y generalizable, se configuró un sistema de evaluación basado en 10 repeticiones entrenamiento/validación con datos distintos en cada iteración, se conoce como *10-fold cross-validation*s. En todas ellas se utilizó el mismo conjunto fijo de 500 TFGs, sobre el cual se aplicaron particiones aleatorias distintas en cada iteración para separar los datos de entrenamiento y validación.

En cada repetición, los textos fueron preprocesados y almacenados en el archivo `dataset_para_modelo.csv`, que contiene dos columnas: el texto procesado de cada TFG y su etiqueta asociada (el Departamento académico).

5.2. Vectorización con TF-IDF

Para representar los textos como vectores numéricos, se aplicó la técnica TF-IDF [8] (Term Frequency – Inverse Document Frequency). Este enfoque pondera la frecuencia de cada término dentro de un documento en función de su frecuencia global, lo que permite captar mejor la relevancia de las palabras.

Se estableció un máximo de 1000 características para mantener una dimensionalidad manejable.

5.3. División de datos

En cada una de las 10 iteraciones, el conjunto vectorizado se dividió aleatoriamente en un 80 % para entrenamiento y un 20 % para validación utilizando `train_test_split`. Esta aleatoriedad entre ciclos permite evaluar el comportamiento medio de los modelos en distintos escenarios de partición.

5.4. Métricas de clasificación

Para evaluar el rendimiento de los modelos supervisados utilizados (k-NN y Random Forest), se emplearon las métricas más habituales en tareas de clasificación multiclase. A continuación se describen brevemente:

- **Accuracy** [1]: proporción de predicciones correctas sobre el total de instancias. Es una métrica general útil cuando las clases están equilibradas.
- **Precision** [1]: porcentaje de verdaderos positivos sobre el total de elementos predichos como positivos. En contextos multiclase, se calcula para cada clase y se promedia. Mide la exactitud del modelo al identificar correctamente los elementos de una clase concreta.
- **Recall** [1]: porcentaje de verdaderos positivos sobre el total de elementos que realmente pertenecen a una clase. Indica la capacidad del modelo para recuperar correctamente los ejemplos de cada clase.
- **F1-score**[1]: media armónica entre precisión y recall. Es especialmente útil cuando existe un desbalance entre clases, ya que penaliza los extremos y refleja un equilibrio entre ambas métricas.

5.4.1. Formulación de métricas de clasificación

$$\text{Accuracy} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{TN} + \text{FP} + \text{FN}}$$

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}}$$

$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}}$$

$$F_1\text{-score} = 2 \cdot \frac{\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}}$$

donde:

- TP (True Positives) son los verdaderos positivos. El modelo predice que pertenece a la clase y acierta.
- TN (True Negatives) son los verdaderos negativos. El modelo predice que no pertenece a la clase y acierta.
- FP (False Positives) son los falsos positivos. El modelo predice que pertenece a la clase y falla.

- FN (False Negatives) son los falsos negativos. El modelo predice que no pertenece a la clase y falla.

Estas métricas fueron calculadas en cada una de las 10 repeticiones experimentales y promediadas para ofrecer una estimación representativa del rendimiento general de cada modelo.

5.5. Modelo basado en *k-Nearest Neighbors* (k-NN)

Se utilizó como primer clasificador el algoritmo *k-Nearest Neighbors* (k-NN), explicado en el capítulo 2, con `k=3`, que asigna a cada TFG la clase más común entre sus tres vecinos más cercanos en el espacio vectorizado. Se elige ese valor de `k` ya que, al ser un número impar, se evitan empates y proporciona un buen equilibrio para evitar ruido de entrenamiento y alto sesgo.

El modelo fue entrenado y validado en cada ciclo de manera independiente, generando métricas como *accuracy*, *precision*, *recall* y *F1-score*, que fueron promediadas al final de los 10 ciclos. También se construyó una matriz de confusión promedio a partir de las predicciones acumuladas, se muestran los resultados en el capítulo 6.

5.6. Modelo basado en *Random Forest*

Como segundo clasificador se utilizó el algoritmo *Random Forest*, explicado en el capítulo 2, configurado con 100 árboles, ya que a partir de ese volumen el error de generalización se estabiliza. Este enfoque, basado en múltiples árboles de decisión entrenados sobre subconjuntos aleatorios de datos y características, es especialmente eficaz para tareas multiclase y datos textuales.

De igual forma que con k-NN, se entrenó y evaluó el modelo durante 10 iteraciones, almacenando los resultados y calculando una media global de métricas de rendimiento y una matriz de confusión promedio, mostrándose los resultados en el capítulo 6

5.7. Clustering: *K-Means* y Clustering Jerárquico

Además de los modelos supervisados, se exploraron enfoques no supervisados mediante algoritmos de agrupamiento. Se aplicaron dos técnicas distintas sobre las representaciones TF-IDF de los TFGs: *K-Means* y *clustering jerárquico aglomerativo*.

5.7.1. Modelo *K-Means* particional

El algoritmo *K-Means* fue configurado para crear un número de clusters igual al número de departamentos, en este caso 7. En cada iteración se ejecutó el algoritmo sobre la muestra seleccionada, sin emplear las etiquetas reales durante el entrenamiento.

5.7.2. Modelo Jerárquico Aglomerativo

Se añadió también un modelo de clustering jerárquico aglomerativo como alternativa a K-Means. Esta técnica construye una jerarquía de agrupamientos en forma de dendrograma, uniendo progresivamente instancias similares hasta alcanzar el número

deseado de clusters. Se utilizó el mismo número de clusters que departamentos, para este caso 7.

5.7.3. Evaluación de los métodos de clustering

Ambos algoritmos fueron evaluados en cada una de las 10 iteraciones utilizando las siguientes métricas estándar:

- **Índice de Rand Ajustado (ARI)** [1]: cuantifica el grado de coincidencia entre los clusters generados y las etiquetas reales, ajustando por el azar.
- **Información Mutua Normalizada (NMI)** [1]: mide la información compartida entre la asignación de clusters y las etiquetas verdaderas.

5.7.4. Formulación de métricas de clustering

Sean dos particiones del mismo conjunto de n elementos: $\mathcal{U} = \{U_1, \ldots, U_r\}$ y $\mathcal{V} = \{V_1, \ldots, V_s\}$. Denotemos $n_{ij} = |U_i \cap V_j|$, $a_i = \sum_j n_{ij}$ y $b_j = \sum_i n_{ij}$.

5.7.4.1. Índice de Rand Ajustado (ARI)

$$\mathrm{ARI} = \frac{\displaystyle\sum_{i,j} \binom{n_{ij}}{2} - \frac{\sum_i \binom{a_i}{2} \sum_j \binom{b_j}{2}}{\binom{n}{2}}}{\frac{1}{2}\Big(\sum_i \binom{a_i}{2} + \sum_j \binom{b_j}{2}\Big) - \frac{\sum_i \binom{a_i}{2} \sum_j \binom{b_j}{2}}{\binom{n}{2}}}$$

Esta fórmula ajusta el índice de Rand estándar para tener valor nulo si la concordancia es la esperada por azar y máximo si las particiones coinciden perfectamente.

5.7.4.2. Información Mutua Normalizada (NMI)

La información mutua entre $\mathcal{U}$ y $\mathcal{V}$ es

$$I(\mathcal{U};\mathcal{V}) = \sum_{i=1}^{r}\sum_{j=1}^{s} \frac{n_{ij}}{n} \log\Big(\frac{n_{ij}/n}{(a_i/n)\,(b_j/n)}\Big).$$

Las entropías de cada partición son

$$H(\mathcal{U}) = -\sum_{i=1}^{r} \frac{a_i}{n} \log \frac{a_i}{n}, \qquad H(\mathcal{V}) = -\sum_{j=1}^{s} \frac{b_j}{n} \log \frac{b_j}{n}.$$

Entonces la NMI se define como

$$\mathrm{NMI} = \frac{I(\mathcal{U};\mathcal{V})}{\sqrt{H(\mathcal{U})\,H(\mathcal{V})}}$$

y toma valores entre 0 (sin información compartida) y 1 (igualdad perfecta de particiones).

Se calcularon los valores medios de ARI y NMI tras las 10 ejecuciones, lo que permite comparar la estabilidad y capacidad estructural de ambos enfoques de agrupamiento.

5.8. Resumen del procedimiento experimental

Todo el proceso experimental se desarrolló sobre una muestra fija de 500 Trabajos de Fin de Grado (TFGs). A partir de esta muestra, se realizaron 10 repeticiones del entrenamiento y evaluación de los modelos, cada una con una partición aleatoria distinta entre entrenamiento (80 %) y validación (20 %).

Esta metodología permite estimar el rendimiento de los modelos frente a diferentes divisiones del conjunto de datos, mitigando así el posible sesgo introducido por una única partición concreta.

Al finalizar las 10 iteraciones, se calcularon los promedios de las principales métricas de evaluación, así como las matrices de confusión acumuladas. Estos resultados agregados ofrecen una visión global del comportamiento medio de los modelos. El capítulo de resultados analiza en detalle estos valores y las visualizaciones asociadas.

Capítulo 6

Resultados

Seguidamente se presentan y analizan los resultados obtenidos tras el entrenamiento de los modelos de clasificación y agrupamiento. Para proporcionar una evaluación más robusta, todo el procedimiento fue repetido 10 veces con particiones aleatorias, y se calcularon los valores medios de las métricas para cada modelo.

Para interpretar los resultados se puede ver la correspondencia Departamento-ID en A.2.

6.1. Resultados del modelo k-NN

El clasificador basado en *k-Nearest Neighbors* obtuvo los siguientes valores promedio tras las 10 iteraciones:

Métrica	**Valor promedio**
Accuracy	0.596
Precision	0.558
Recall	0.360
F1-score	0.276

Cuadro 6.1: Métricas promedio del modelo k-NN (10 iteraciones)

Estos resultados reflejan una capacidad moderadamente limitada para identificar correctamente el departamento de cada TFG, especialmente en clases minoritarias. El modelo tiende a funcionar mejor en clases con mayor representación.

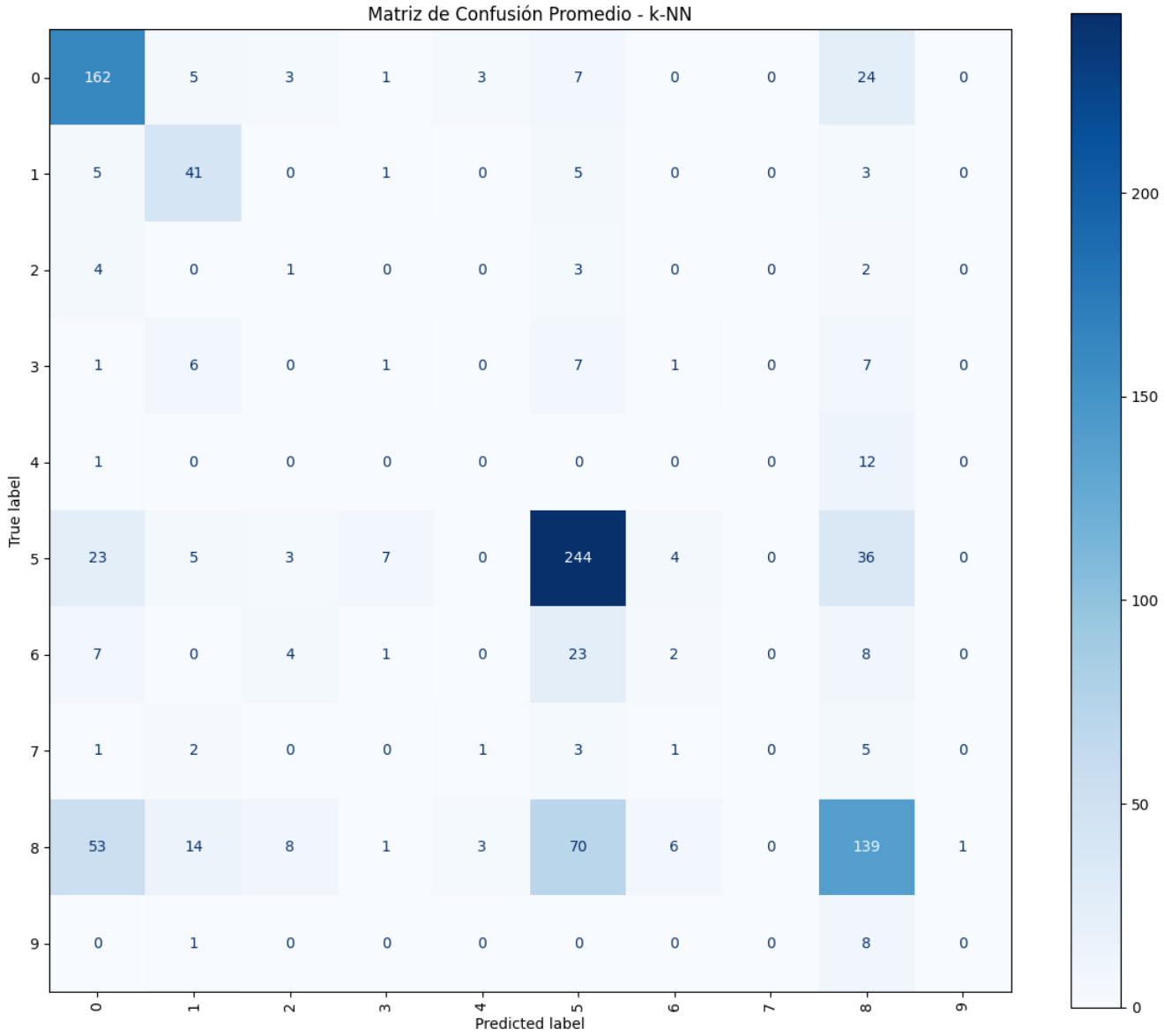

Figura 6.1: Matriz de confusión promedio del modelo k-NN (10 iteraciones)

6.2. Resultados del modelo Random Forest

El modelo *Random Forest* mostró una mejora en las métricas generales respecto a k-NN. Los resultados promedio fueron:

Cuadro 6.2: Métricas promedio del modelo Random Forest (10 iteraciones)

Métrica	**Valor promedio**
Accuracy	0.583
Precision	0.796
Recall	0.299
F1-score	0.272

La alta precisión sugiere que el modelo clasifica correctamente cuando predice una clase, pero con baja cobertura sobre todas las clases (recall). Esto es típico cuando hay clases desbalanceadas.

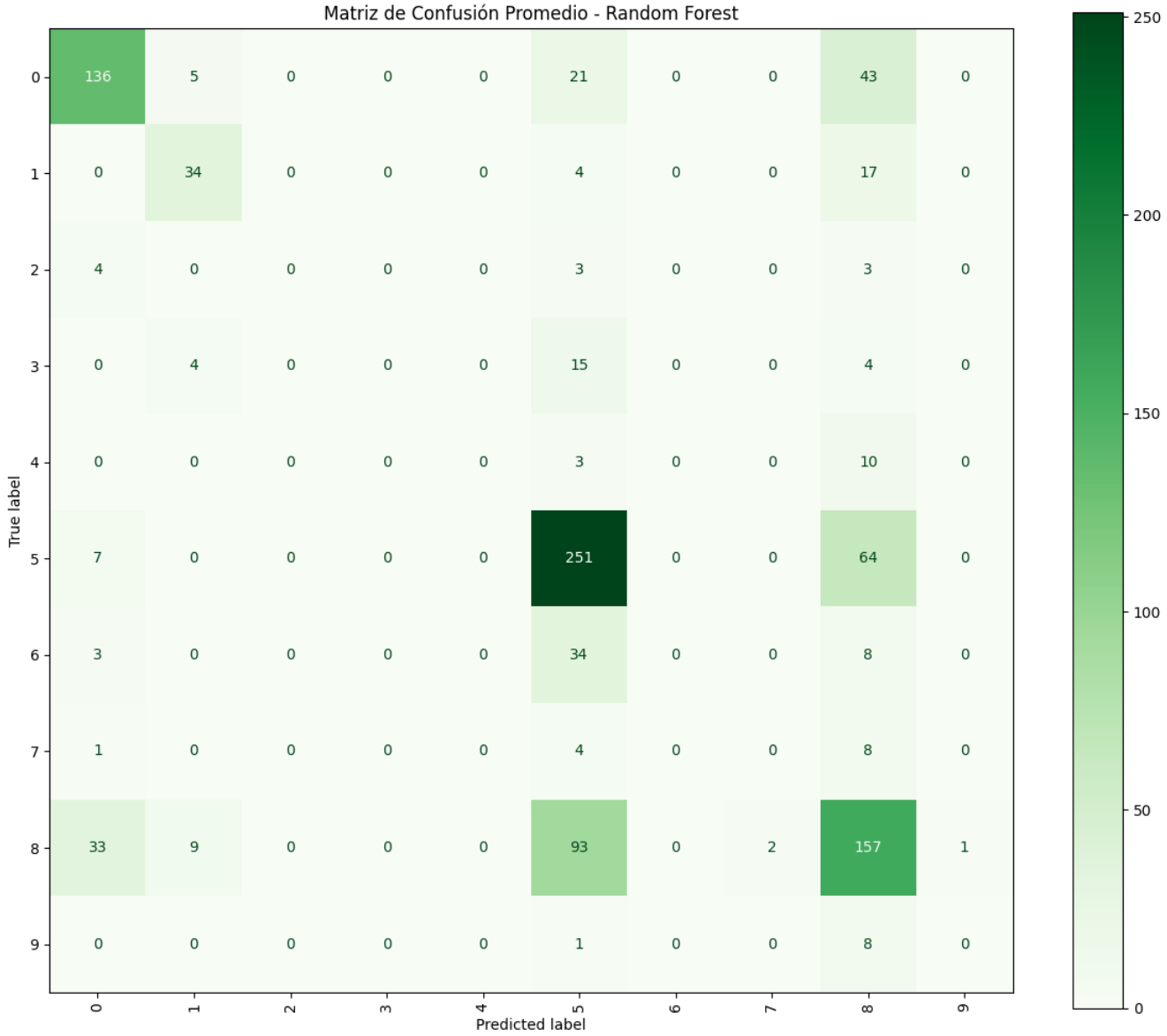

Figura 6.2: Matriz de confusión promedio del modelo Random Forest (10 iteraciones)

6.3. Comparación gráfica de clasificadores

Para facilitar la comparación entre los dos modelos supervisados, se generó el siguiente gráfico de barras con las métricas promedio:

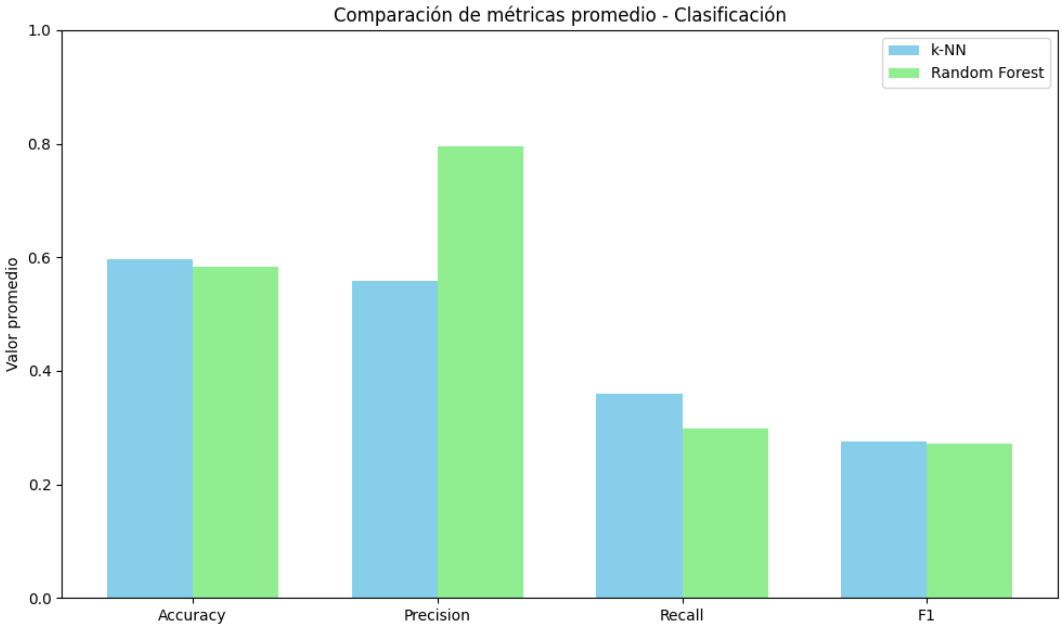

Figura 6.3: Comparación de métricas promedio entre k-NN y Random Forest

Como se observa en el gráfico, el modelo Random Forest alcanza una precisión superior a la de k-NN, lo que indica que, cuando predice una clase, lo hace con mayor fiabilidad. Sin embargo, sus valores de recall [1] y F1-score [1] son bajos, lo que refleja dificultades para identificar correctamente muchas clases reales, especialmente las menos representadas. Esta diferencia se debe al desequilibrio en la distribución de clases: Random Forest tiende a concentrar sus aciertos en las clases mayoritarias, ignorando parcialmente las minoritarias.

En cambio, el modelo k-NN muestra un rendimiento más equilibrado entre precisión y recall, aunque ambos valores son moderados. Esto se debe a que k-NN no aprende patrones explícitos, sino que basa sus decisiones en la similitud entre textos, lo cual resulta ineficiente en un contexto con muchas clases y representación dispersa.

En resumen, Random Forest detecta mejor patrones dominantes en clases frecuentes, mientras que k-NN mantiene un comportamiento más uniforme pero menos preciso en general.

6.4. Resultados de Clustering

Se evaluaron dos técnicas de agrupamiento no supervisado: *K-Means* y *clustering jerárquico aglomerativo*. Ambos modelos utilizaron el mismo número de clusters que clases reales (departamentos), y fueron evaluados mediante las métricas externas:

- **ARI (Adjusted Rand Index)**
- **NMI (Normalized Mutual Information)**

6.4.1. Resultados promedio

Los valores medios tras 10 ejecuciones fueron los siguientes:

Modelo	ARI	NMI
K-Means	0.056	0.130
Jerárquico	0.081	0.146

Cuadro 6.3: Métricas promedio de clustering tras 10 iteraciones

Ambos modelos muestran bajos valores de ARI (indican baja coincidencia con las etiquetas reales), aunque el NMI refleja cierta estructura compartida, más visible en el clustering jerárquico.

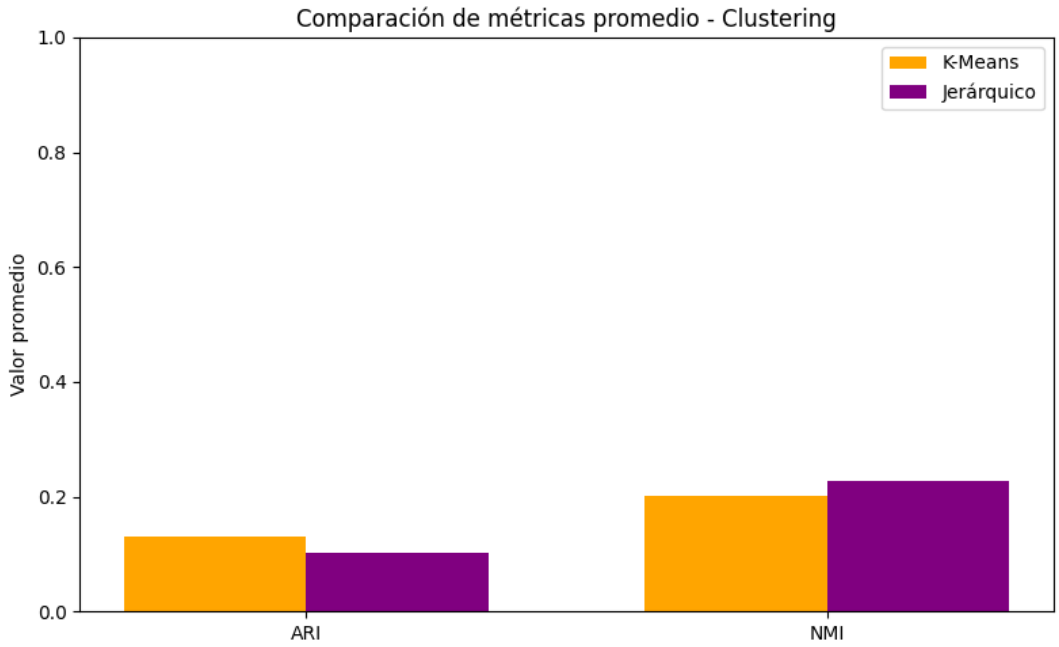

Figura 6.4: Comparación de métricas promedio entre K-Means y clustering jerárquico

Además de las métricas cuantitativas, a continuación se muestran algunos ejemplos de departamentos originales y el clúster que asignó el algoritmo jerárquico:

Departamento original	Cluster asignado
Otros	1
Otros	2
Arquitectura	2
Arquitectura	0
Otros	2

Cuadro 6.4: Ejemplos de clustering jerárquico tras 10 iteraciones

6.5. Resumen comparativo

De forma global, el rendimiento de los modelos supervisados fue limitado, especialmente por el desbalance entre las clases. El modelo Random Forest obtuvo mejores

resultados que k-NN, especialmente en precisión, aunque su F1-score sigue siendo bajo.

Por otro lado, los modelos no supervisados permiten analizar la estructura semántica de los datos desde una perspectiva diferente. Aunque sus métricas indican una coincidencia parcial con las etiquetas reales, permiten detectar agrupamientos temáticos latentes.

- **Modelos supervisados:** rendimiento numérico mejor definido, pero limitado por la distribución de clases.
- **Modelos no supervisados:** útiles para explorar la organización temática de los textos, incluso sin etiquetas.

Capítulo 7

Conclusiones

7.1. Evaluación de los objetivos

El objetivo principal del trabajo, que era desarrollar una metodología para extraer, analizar y clasificar los Trabajos de Fin de Grado (TFG) almacenados en el **Archivo Digital de la UPM (AD-UPM)** basándonos en el departamento del director del TFG, se ha completado de manera adecuada. Se ha realizado un análisis exploratorio extenso, identificando variables clave y patrones mediante técnicas de minería de datos y aprendizaje automático. La implementación de modelos supervisados, como los algoritmos k-NN y Random Forest, y no supervisados, como K-Means y clustering jerárquico, ha permitido evaluar su rendimiento, mostrando sus puntos fuertes y débiles que han sido identificados a lo largo del estudio.

7.2. Líneas futuras

Para futuras investigaciones, lo que podría mejorar nuestro trabajo sería incorporar técnicas avanzadas basadas en transformadores (BERT, GPT), lo cual podría mejorar significativamente la precisión del modelo, especialmente en escenarios con un alto número de clases. También se recomienda explorar estrategias adicionales de preprocesamiento de textos que puedan incluir características semánticas y contextuales, así como incorporar otras variables del **AD-UPM** que no fueron incluidas en este estudio.

7.3. Evaluación personal del proceso de realización del TFG

El proceso de realizar este Trabajo de Fin de Grado ha sido una experiencia llena de desafíos constantes, como trabajar con conjuntos de datos desequilibrados y buscar la forma de mejorar la precisión y eficacia de los modelos utilizados. Estos retos han implicado una constante necesidad de aprendizaje y adaptación. Además, me ha permitido aplicar de manera práctica muchos de los conocimientos adquiridos durante la carrera sobre minería de datos y aprendizaje automático, así como aprender nuevos conceptos en estos campos.

7.4. Análisis de impacto, ODS

Este proyecto está alineado con varios Objetivos de Desarrollo Sostenible (ODS) definidos por las Naciones Unidas, particularmente con el objetivo número 4 (Educación de Calidad), ya que mejora la accesibilidad y organización del conocimiento, en este caso del que se encuentra en el **Archivo Digital de la UPM (AD-UPM)**. También contribuye al objetivo número 9 (Industria, Innovación e Infraestructura), proporcionando herramientas tecnológicas que facilitan la gestión eficiente y la recuperación de información académica.

Bibliografía

[1] Charu C. Aggarwal y ChengXiang Zhai. *Mining Text Data*. Springer, 2012. ISBN: 9781461432234. URL: 10.1007/978-1-4614-3223-4.

[2] Julia Silge y David Robinson. *Text Mining with R: A Tidy Approach*. O'Reilly Media, 2017. ISBN: 9781491981658. URL: https://www.tidytextmining.com/.

[3] Wes McKinney y the pandas Development Team. *pandas: Python Data Analysis Library*. https://pandas.pydata.org/. Python library for data manipulation and analysis. 2023.

[4] The Unidecode Development Team. *Unidecode Python Package*. https://pypi.org/project/Unidecode/. Python library for transliterating Unicode text into ASCII. 2023.

[5] Steven Bird, Ewan Klein y Edward Loper. *Natural Language Toolkit (NLTK)*. https://www.nltk.org/. Python library for natural language processing. 2009.

[6] Kenneth Reitz y Contributors. *Requests: HTTP for Humans*. https://docs.python-requests.org/. Python library. 2023.

[7] Leonard Richardson y Contributors. *Beautiful Soup Documentation*. https://www.crummy.com/software/BeautifulSoup/bs4/doc/. Python library for parsing HTML and XML. 2023.

[8] Rafael Caballero, Enrique Martín Martín y Adrián Riesco. *Análisis y minería de textos con Python*. Alphaeditorial RC Libros, 2023. ISBN: 9789587789362. URL: https://www.alpha-editorial.com/Papel/9789587789362/An%C3%A1lisis%2BY%2BMiner%C3%ADa%2BDe%2BTextos%2BCon%2BPython.

Apéndice A

Primer anexo

Listados de Código

Listing A.1: Scraper de enlaces

```
import requests
from bs4 import BeautifulSoup

base_url = "https://oa.upm.es/cgi/search/simple?q=TFG&_action_search=Buscar&_action_search=Search&_order=bytitle&basic_srchtype=ALL&_satisfyall=ALL"

headers = {
    "User-Agent": "Mozilla/5.0"
}

tfg_links = []
offset = 0
max_tfgs = 500

while len(tfg_links) < max_tfgs:
    url = f"{base_url}&search_offset={offset}"
    response = requests.get(url, headers=headers)
    soup = BeautifulSoup(response.text, "html.parser")

    for link in soup.find_all("a", href=True):
        href = link["href"]
        if href.startswith("/"):
            href = "https://oa.upm.es" + href
        if href.startswith("https://oa.upm.es/") and href[19:23].isdigit() and href.endswith("/"):
            tfg_links.append(href)

    tfg_links = list(set(tfg_links))
    if len(tfg_links) >= max_tfgs:
        break

    offset += 25

with open("enlaces.txt", "w") as file:
    for link in tfg_links:
        file.write(link + "\n")

print(f"Se encontraron {len(tfg_links)} TFGs y se han guardado en 'enlaces.txt'.")
```

Listing A.2: Función de limpieza de texto

```
def clean_text(text):
    text = re.sub(r'^(T t u l o|Autor/es|Director/es|Palabras Clave Informales|
        Escuela|Departamento|Resumen):', '', text, flags=re.IGNORECASE)
    return text.strip()
```

Listing A.3: Extracción de nombres de directores

```
def extract_directors(soup, class_name):
    directors_section = soup.find(class_=class_name)
    if not directors_section:
        return "No disponible"

    # Buscar todos los elementos <li> dentro de la lista de directores
    directors = []
    for li in directors_section.find_all("li"):
        name_span = li.find("span", class_="person_name")
        if name_span:
            directors.append(name_span.get_text(strip=True))

    return "; ".join(directors) if directors else "No disponible"
```

Listing A.4: Obtención de texto por clase CSS

```
def get_text_by_class(soup, class_name):
    element = soup.find(class_=class_name)
    if element:
        text = element.get_text(strip=True)
        return clean_text(text)
    return "No disponible"
```

Listing A.5: Extracción de campos

```
with open("Datos_raw.txt", "r", encoding="utf-8") as f:
    raw_data = f.read()

records = raw_data.split("=" * 80)
records = [rec.strip() for rec in records if rec.strip() != ""]

pattern = re.compile(
    r"Enlace:\s*(?P<Enlace>.+?)\n"
    r"T t u l o:\s*(?P<T t u l o>.+?)\n"
    r"Autores:\s*(?P<Autores>.+?)\n"
    r"Director/es:\s*(?P<Director>.+?)\n"
    r"Palabras Clave:\s*(?P<Palabras_Clave>.+?)\n"
    r"Escuela:\s*(?P<Escuela>.+?)\n"
    r"Departamento:\s*(?P<Departamento>.+?)\n"
    r"Resumen:\s*(?P<Resumen>.+)",
    re.DOTALL
)

data = []
for rec in records:
    match = pattern.search(rec)
```

```
    if match:
        data.append(match.groupdict())

df = pd.DataFrame(data)
```

Listing A.6: Limpieza y tokenización

```
stop_words = set(stopwords.words('spanish'))
stemmer = SnowballStemmer("spanish")

def limpiar_texto(texto):
    texto = texto.lower()
    texto = unidecode.unidecode(texto)  # Elimina tildes y
    texto = texto.translate(str.maketrans('', '', string.punctuation + string.
        digits))
    tokens = word_tokenize(texto)
    tokens = [t for t in tokens if t not in stop_words and t.isalpha() and len(t) >
         2]
    tokens = [stemmer.stem(t) for t in tokens]
    return tokens

df["tokens"] = df["Resumen"].apply(limpiar_texto)
df["texto_procesado"] = df["tokens"].apply(lambda x: " ".join(x))
```

Listing A.7: Guardado de datos procesados

```
df.to_csv("Datos_preprocesados.csv", index=False, encoding="utf-8")
df[["texto_procesado", "Departamento"]].to_csv(
    "dataset_para_modelo.csv", index=False, encoding="utf-8"
)

with open("tokens_para_modelo.txt", "w", encoding="utf-8") as f:
    for tokens in df["tokens"]:
        f.write(" ".join(tokens) + "\n")

print("    Datos procesados y guardados.")
```

Listing A.8: Agrupación de departamentos

```
def agrupa_dept_equilibrado(dept: str) -> str:
    d = dept.lower()
    # Ingenier a: grandes ramas
    if any(k in d for k in ["inform t", "informatic", "comput"]):
        return "Inform tica"
    if any(k in d for k in ["el ctr", "electr", "telecom"]):
        return "Electr nica y Telecom."
    if any(k in d for k in ["mec n", "mecan"]):
        return "Mec nica"
    if "industrial" in d:
        return "Industrial"
    if "civil" in d:
        return "Civil"
    if "qu m" in d or "quim" in d:
        return "Qu mica"
    # Resto de subgrupos basados en frecuencia en    Otros
    if "arquitect" in d:
        return "Arquitectura"
    if "edificaci" in d:
```

```
20          return "Edificaci n"
21      if "biolog" in d:
22          return "Biolog a"
23      if "matem t" in d or "matematic" in d:
24          return "Matem ticas"
25      if "biom d" in d or "biomed" in d or "sanit" in d:
26          return "Biom dica"
27      # Todo lo dem s
28      return "Otros"
29
30  df["Departamento"] = df["Departamento"].apply(agrupa_dept_equilibrado)
```

Agrupación departamentos

Cuadro A.1: Recuento de TFGs por Departamento tras el preprocesamiento

Departamento	**Recuento**
Informática	163
Otros	150
Arquitectura	99
Biología	29
Matemáticas	18
Electrónica y Telecom.	12
Edificación	8
Mecánica	7
Industrial	5
Química	4

Correspondencia ID - Departamento

Cuadro A.2: Relación entre IDs numéricos y departamentos

ID	**Departamento**
0	Arquitectura
1	Biología
2	Edificación
3	Electrónica y Telecom.
4	Industrial
5	Informática
6	Matemáticas
7	Mecánica
8	Otros
9	Química

Este documento esta firmado por

	**Firmante**	CN=tfgm.fi.upm.es, OU=CCFI, O=ETS Ingenieros Informaticos - UPM, C=ES
	Fecha/Hora	Wed Jun 04 18:36:49 CEST 2025
	Emisor del Certificado	EMAILADDRESS=camanager@etsiinf.upm.es, CN=CA ETS Ingenieros Informaticos, O=ETS Ingenieros Informaticos - UPM, C=ES
	Numero de Serie	561
	Metodo	urn:adobe.com:Adobe.PPKLite:adbe.pkcs7.sha1 (Adobe Signature)