Facultad de Ciencias

Clasificación automática de contratos del sector público

(Automatic classification of public procurement contracts)

Trabajo de Fin de Máster
para acceder al

MÁSTER EN CIENCIA DE DATOS

Autora: Julia Ruiz Salmón

Directora: Cristina Tirnauca

Septiembre - 2019

Resumen

Este proyecto tiene como objetivo el desarrollo de un producto *software* que permita realizar una clasificación automática, con la máxima precisión posible, del mayor número de contratos públicos del Gobierno de Cantabria, en relación a unas dimensiones predefinidas: *Naturaleza del gasto* y *Objeto de administración*.

La fuente de datos contiene una serie de textos descriptivos de los contratos de los años 2014-2015 y la clase correspondiente a la que pertenece el contrato según la dimensión. Los textos constan de diferente vocabulario que describe las solicitudes de contrato. Gran parte de este vocabulario, e incluso simbología, carece de significado a la hora de clasificar los contratos en las diferentes clases.

Por lo tanto, el propósito de este proyecto es aplicar diversas técnicas de preprocesado de los datos de entrada para eliminar el vocabulario irrelevante de los textos y conseguir que contengan únicamente el vocabulario necesario y relevante con respecto a la categoría de clasificación.

El tamaño variable y la estructura sin formato de los textos no son procesables por los algoritmos de clasificación. Así, el siguiente paso es convertir los textos en vectores de características numéricas a través del cálculo del *term frequency-inverse document frequency (tf-idf)*.

Por último, se aplican diferentes algoritmos de clasificación pertenecientes al aprendizaje supervisado: regresión logística, máquinas de vector soporte (*SVM*), árboles de decision, *Random Forest* y el clasificador *Naive Bayes*.

Los resultados se analizan a través de tres métricas diferentes (*accuracy, matriz de confusión*, *precision* y *recall*), que muestran diversas conclusiones para cada uno de los clasificadores empleados.

Palabras clave: preprocesado, contrato público, *tf-idf*, *stemming*, algoritmo, clasificación.

Abstract

The aim of this project is to develop a software product that is able to classify automatically, with the highest possible accuracy, the largest number of public procurement contracts from the Government of Cantabria, in relation to predefined dimensions: *Naturaleza del gasto* y *Objeto de administración*.

Available data contains a text collection describing *public procurement contracts* from 2014 and 2015, and the corresponding category to which the contract belongs according to the dimension. Texts consist of different vocabulary that describes the contract. Most of words and symbology do not provide relevant information to classify the contracts into different categories.

Therefore, the purpose of this project is to apply several data preprocesssing techniques to take out irrelevant vocabulary from the texts and to exclusively obtain the necessary and important words regarding the category of classification.

The variable size and unformatted structure of the texts are not actionable by the algorithms. Thus, the next step is to convert the texts into numerical characteristics vectors through the *term frequency-inverse document frequency (tf-idf)*.

Finally, some supervised learning algorithms are applied: logistic regression, support vector machines (*SVM*), decision trees, Random Forest and Naive Bayes classifier.

Results are analysed using three metrics (*accuracy, confusion matrix*, *precision* and *recall*), showing relevant conclusions for each of the classifiers employed.

Keywords: data preprocessing, public procurement contract, tf-idf, stemming, algorithm, classification.

Índice general

1 Introducción

En la actualidad, uno de los ejes fundamentales del día a día del ser humano es el conocimiento de las cosas y de la información. Sin embargo, la cantidad de datos y de textos existentes crece cada vez más rápido. Por esta razón, y por la gran demanda en el estudio de este tipo de datos, surgió la minería de textos como interdisciplina. Esta ciencia es el proceso que se encarga de la extracción y recopilación de patrones en los datos. Estos patrones representan la información de los textos y extraen el conocimiento útil del contenido de los mismos. La minería de textos se puede aplicar a diferentes ámbitos como, por ejemplo, extracción de la información, clasificación de textos, resumen de datos, relación de conceptos o agrupación de datos (*clustering*) entre otros [1].

El siguiente proyecto se centra en un caso de estudio de la clasificación de textos. Se trata de realizar una clasificación de registros de contratos públicos disponibles en el *Sistema de Tramitación de Solicitudes de remisión de Información relativa a actuaciones de Gasto del Gobierno de Cantabria*.

1.1. Objetivo

El objetivo es desarrollar un producto software que permita la clasificación automática, con la máxima precisión posible, del mayor número de registros de contrato, en base a unas dimensiones predefinidas.

1.2. Metodología

La metodología que se sigue consta de la aplicación de métodos de preprocesado y manipulación de los textos de entrada, y en aplicar técnicas de aprendizaje automático en el conjunto de datos disponible. Para ello, se ha utilizado el lenguaje de programación Python a través del editor de código fuente *Visual Studio Code*. A continuación, se detallan las principales librerías de Python utilizadas:

- Librería *nltk*: para preprocesar los datos de entrada.
- Librerías *pandas* y *numpy*: para manipular los datos de entrada.

- Librería *matplotlib* y *manifold*: para dibujar gráficas de los datos.
- Librería *sklearn*: para la vectorización de los datos de entrada y para la aplicación de los algoritmos que clasifican los datos.

El desarrollo del software se puede encontrar en https://gitlab.com/rsj9999/text-classifier.git.

1.3. Estructura del proyecto

El proyecto consta de diferentes partes que se van a explicar a continuación para conseguir los objetivos propuestos:

- Normalización de los textos de entrada.
- Conversión de los textos en vectores numéricos.
- Visualización de los datos.
- Balanceo de datos para equilibrar el número de textos pertenecientes a una u otra categoría.
- Aplicación de los algoritmos de aprendizaje automático para la clasificación.
- Interpretación de los resultados.

A continuación, se exponen de manera detallada los datos de partida. Para comenzar un proyecto que depende del análisis de unos datos es muy importante conocer la estructura y naturaleza de los mismos.

2
Preprocesado de los datos

2.1. Naturaleza de los datos

Los datos para este proyecto tienen su origen en el sector público del Gobierno de Cantabria. Estos datos están anonimizados y contienen textos descriptivos de los tipos de contratos de obra realizados en los años 2014 y 2015 en diferentes lugares de la comunidad de Cantabria. Los diversos motivos para la elaboración de los contratos y la naturaleza de los mismos han propiciado la aplicación de una categorización en diferentes ámbitos, determinando si la obra es nueva o de mantenimiento, qué o quién se beneficia o el lugar donde se ha realizado la obra, entre otros.

Por esta razón, los datos de los que se parte se dividen en cuatro ámbitos o dimensiones diferentes. A continuación se expone una breve explicación de las diferentes dimensiones:

- **Naturaleza del gasto**: indica si la solicitud de obra es para mantenimiento o conservación, o si es una nueva inversión o mejora del objeto.
- **Objeto de administración**: indica qué o cuál es el objeto al que se le va a aplicar el contrato de obra.
- **Cliente beneficiario**: indica qué empresa o institución es la beneficiaria del contrato de obra.
- **Ámbito geográfico**: indica en qué lugar se ha realizado la obra.

Este proyecto se va a centrar en la clasificación de los contratos de obra únicamente en las dos primeras dimensiones, *Naturaleza del gasto* y *Objeto de administración*. En el caso de la tercera dimensión, solo se dispone de datos anonimizados. La cuarta dimensión no requiere aplicar algoritmos de Minería de Datos.

Por otro lado, la categorización de los contratos se subdivide en niveles dentro de cada dimensión. En el caso de la dimensión *Naturaleza del gasto*, solamente se tienen datos de un nivel en el que se establece si la obra es nueva o de mantenimiento. La Tabla 2.1 muestra un ejemplo de los datos de partida para la dimensión *Naturaleza del gasto* y su nivel correspondiente, así como las categorías en las que se clasifica cada registro de contrato.

Identificador	Categoría	Texto del contrato
2014-04-08-000259	1002000000000-INVESIÓN NUEVA/ CONSTRUCCIÓN/ MEJORAS	**Mejora de plataforma y refuerzo de firme CA-429, Acceso a Estrada de Hoz, p.k. 0,000 al p.k. 1,600. Tramo: Anero - Estrada de Hoz.**
2014-04-08-000276	1003000000000-MANTENIMIENTO/ CONSERVACIÓN/ INVERSIÓN DE REPOSICIÓN	**PROYECTO DE REPARACION PUNTUAL DE LA CAPA DE RODADURA DE LA CARRETERA AUTONOMICA CA-850 DE TRECEÑO A PUENTE EL ARRUDO, PK 0+000 AL PK 17+200"**

Tabla 2.1: Textos de entrada para las categorías correspondientes al *nivel 1.1* de la dimensión *Naturaleza del gasto* y su identificador.

En el *nivel 2.1* de la dimensión *Objeto de administración* se muestran categorías más generales del objeto (veáse Tabla 2.2). De este modo, una categorización posible puede ser que el objeto sea una obra civil o infraestructura, una edificación o instalación o que se trate de terrenos o bienes naturales. Además, estas categorías se subdividen en otras dos según sean contratos llevados a cabo por el gobierno de la comunidad (C.A. Cantabria) o por los ayuntamientos (Otras administraciones).

Categorías para el nivel 2.1
2003-CENTROS/EDIFICACIONES/INSTALACIONES - C.A. CANTABRIA
2103-CENTROS/EDIFICACIONES/INSTALACIONES - OTRAS ADMINISTRACIONES
2002-OBRA CIVIL/ INFRAESTRUCTURAS - C.A. CANTABRIA
2102-OBRA CIVIL/ INFRAESTRUCTURAS - OTRAS ADMINISTRACIONES
2001-TERRENOS Y BIENES NATURALES - C.A. CANTABRIA
2101-TERRENOS Y BIENES NATURALES - OTRAS ADMINISTRACIONES

Tabla 2.2: Categorías de clasificación correspondientes al *nivel 2.1*.

El *nivel 2.2* se refiere a una definición más particular del objeto con respecto al *nivel 2.1*. En este nivel, la solicitud del contrato puede formar parte de una infraestructura vial, portuaria, medio ambiental o de transporte, de un demandial o de un abastecimiento, entre otros. La clasificación para este nivel se realizará junto con las categorías del *nivel 2.1* ya que depende del primero. Por tanto, se obtiene un mayor número de clases.

Por último, el *nivel 2.3* define el objeto en cuestión. De esta manera, las solicitudes de contrato pueden estar destinadas a una obra de carreteras, a edificios judiciales o a centros deportivos, entre otros. En este último nivel de clasificación no se va a centrar el proyecto debido a la falta de categorización de los datos por parte de la fuente de datos.

A continuación, se muestran las Tablas 2.3 y 2.4 donde se recogen algunos ejemplos de datos de origen para los *niveles 2.1* y *2.2*, y las categorías que resultan de unir ambos niveles.

Identificador	Categoría	Texto del contrato	Nivel
2014-06-00-001471	2003000000000-CENTROS/ EDIFICACIONES/INSTALACIONES - C.A. CANTABRIA	**Reubicación de la centralita de alarma de incendio del edificio Palacio Machoo, sede de la Consejería de Economía, Hacienda y Empleo**	2.1
2015-07-03-001302	2101000000000-TERRENOS Y BIENES NATURALES - OTRAS ADMINISTRACIONES	CONTRATO MENOR DE OBRA RECUPERACIÓN AMBIENTAL Y PAISAJÍSTICA DE ANTIGUO VERTEDERO DE INERTES (T.M. SANTIURDE DE REINOSA)	2.1
2014-09-00-000904	2003002000000-DEMANIALES	REFORMA DE LOS ASEOS DE LA 7ª PLANTA DE LA CONSEJERÍA DE EDUCACIÓN, CULTURA Y DEPORTE EN EL EDIFICIO MINISTERIOS C/ VARGAS 53 DE SANTANDER	2.2
2015-03-04-004530	2002011000000-SANEAMIENTOS	MEDICIÓN GENERAL Y CERTIFICACIÓN FINAL DE LA OBRA DE SANEAMIENTO EN EL BARRIO EL PUENTE (ORUÑA)	2.2

Tabla 2.3: Textos de entrada para las categorías correspondientes a los *niveles 2.1* y *2.2* de la dimensión *Objeto de administración* y su identificador.

Categorías para los niveles 2.1-2.2	
1-CENTROS/EDIFICACIONES/ INSTALACIONES - C.A. CANTABRIA-DEMANIALES	**2**-CENTROS/EDIFICACIONES/ INSTALACIONES-OTRAS ADMINISTRACIONES-DEMANIALES
3-OBRA CIVIL/ INFRAESTRUCTURAS - C.A. CANTABRIA- INFRAESTRUCTURAS PORTUARIAS	**4**-OBRA CIVIL/ INFRAESTRUCTURAS - C.A. CANTABRIA- ABASTECIMIENTOS
5-OBRA CIVIL/ INFRAESTRUCTURAS - C.A. CANTABRIA- INFRAESTRUCTURA VIAL	**6**-OBRA CIVIL/ INFRAESTRUCTURAS - C.A. CANTABRIA- INFRAESTRUCTURAS DE TRANSPORTE
7-OBRA CIVIL/ INFRAESTRUCTURAS - OTRAS ADMINISTRACIONES- ABASTECIMIENTOS	**8**-OBRA CIVIL/INFRAESTRUCTURAS-OTRAS ADMINISTRACIONES- INFRAESTRUCTURA VIAL
9-OBRA CIVIL/INFRAESTRUCTURAS - OTRAS ADMINISTRACIONES-PAVIMENTACIÓN DE CAMINOS Y CALLES	**10**-OBRA CIVIL/INFRAESTRUCTURAS-OTRAS ADMINISTRACIONES-SANEAMIENTOS
11-TERRENOS Y BIENES NATURALES - OTRAS ADMINISTRACIONES	

Tabla 2.4: Categorías de clasificación correspondientes al *nivel 2.1* y al *nivel 2.2*.

Una vez se conocen la estructura y naturaleza de los datos, su tratamiento es otra de las características más importantes en el análisis de los datos. Por esta razón, el preprocesado de los textos descriptivos del contrato es una de las partes en las que se va a centrar este proyecto antes de aplicar las técnicas algorítmicas.

2.2. Técnicas de preprocesado de los datos

En la sección anterior se ha podido observar que los textos descriptivos de las solicitudes de contrato contienen mucho vocabulario y simbología prescindible para el objetivo de este

trabajo. La clasificación de textos, en base a los diferentes niveles y dimensiones de los datos de entrada, se centra en observar el vocabulario del que consta el texto y en encontrar las relaciones existentes (o no) entre las palabras y los nombres de las categorías para extraer información útil del vocabulario. De esta manera, según el vocabulario que contiene el texto, el algoritmo tiene como objetivo ser capaz de clasificar el contrato en una u otra categoría.

En este proyecto se han aplicado técnicas de preprocesado de datos que se pueden clasificar, por un lado, en técnicas comunes para cualquier tipo de texto orientado a un trabajo de clasificación de textos y, por otro lado, en técnicas de preprocesado más específicas dependiendo del vocabulario y el ámbito al que los textos pertenecen.

Existen varias técnicas generales o comunes que tienen como objetivo la normalización de los textos para una mejor comprensión o tratamiento de los mismos. Algunos de los procesos de normalización que se aplican son el *stemming* y la *expansión asimétrica*, que se detallan más adelante.

Como se ha explicado en la introducción de este proyecto, este tipo de técnicas se han llevado a cabo a través de librerías contenidas en el lenguaje de programación Python.

2.2.1. Normalización de textos

Texto a minúscula

El primer paso del preprocesado de datos es convertir los textos a minúscula con el objetivo de facilitar la manipulación de los textos y de homogeneizar todo el conjunto de datos.

Expresiones regulares

Las expresiones regulares [2] son secuencias de caracteres que buscan patrones comunes y permiten efectuar operaciones de sustitución en textos, como en el caso del conjunto de datos de este trabajo.

En particular, estos comandos de programación han servido para eliminar palabras sin significado propio por sí mismas y caracteres alfanuméricos (monosílabos, símbolos como ‘º’, ‘?’, ...), diversos espacios en blanco innecesarios, los acentos de las palabras y sustituir algunas expresiones por otras de significado similar con el objetivo de homogeneizar el vocabulario existente en los textos. Es lo que se denomina *expansión asimétrica*, que consiste en, por ejemplo, reemplazar la abreviatura ‘ayto“*nombre*“’ por ‘+ayuntamiento’ en todos los textos que aparecen.

Ejemplo de sustitución de ‘ayto’ por ‘+ayuntamiento’	
Sin sustituir	“tratamiento de eliminación de las inusualmente altas concentraciones de hierro contenidas en el agua bruta de la etap de agüera **ayto guriezo**”
Con sustitución	“tratamiento de eliminación de las inusualmente altas concentraciones de hierro contenidas en el agua bruta de la etap de agüera **+ayuntamiento**”

Por otro lado, también se han utilizado estas expresiones para reemplazar diferente vocabulario contenido en el mismo campo semántico (nombres de colegios, de institutos, nombres de municipios, etc) a través de la creación de diccionarios (objetos propios del lenguaje de pro-

gramación Python) cuya explicación se detalla más adelante.

Stopwords y preposiciones

El término *stopwords* se refiere a aquellas palabras que son más comunes pero que no contienen importante significado en los textos y que, por lo general, se eliminan. Algunos ejemplos de este tipo de palabras en el lenguaje español son los artículos o determinantes (la, el, lo, los, su, mi ...) o las conjunciones (y, o, u). Sin embargo, existe otro tipo de palabras pertenecientes a este grupo como por ejemplo: 'que', 'muy', 'cuando' o 'también'.

Para la eliminación de estas palabras se ha utilizado la librería *nltk* de *Python* que contiene por defecto todas estas palabras según el idioma que se le especifique al método [3].

Por otro lado, existen palabras en español de tipo preposición (a, ante, cabe, con,...) que también se han eliminado. Estas palabras se han guardado en un fichero *'.csv'* y se han borrado a través de la lectura del mismo y de la aplicación de un comando Python que ignora estas palabras.

Para la realización de este preprocesado cabe destacar que, previamente a la eliminación de las palabras, se ha dividido el texto en *tokens* (denominado así en clasificación de textos), o, lo que es lo mismo, en las palabras del texto por separado y contenidas en una lista-objeto de Python. En el siguiente ejemplo, se muestra el resultado para uno de los textos del proyecto.

Ejemplo de eliminación de *stopwords*	
Sin eliminar	"contrato **de** obra denominado mejora **del** valor socioeconómico **y** recuperación **de** espacios existentes **en el** entorno **de** río besaya "
Tokens	['contrato', 'obra', 'denominado', 'mejora', 'valor', 'socioeconómico', 'recuperación', 'espacios', 'existentes', 'entorno', 'río', 'besaya']

Stemming vs *Lemmatización*

Stemming es el proceso que aplica la reducción de palabras a su base o raíz [3, 4]. Este tipo de preprocesado de textos hace que el vocabulario se reduzca, agrupando las palabras cuya raíz es la misma.

No obstante, existen ciertos inconvenientes a la hora de utilizar este tipo de preprocesado. Por un lado, cuando se reduce una palabra a su base o raíz se pierde el significado de la misma, es decir, la palabra se convierte en una palabra no válida en el lenguaje. Por otro lado, el vocabulario para este caso de estudio es muy reducido y de manera peligrosa se puede llegar a prescindir de textos enteros reduciendo demasiado el vocabulario.

Existen diferentes métodos de la librería *nltk* para hacer *stemming*. En cambio, en el lenguaje español solamente existe un método denominado *SnowballStemmer*.

Lemmatización es un proceso muy similar a *stemming* ya que reduce las palabras a una base común, con la diferencia de que simplemente no corta las terminaciones. La *lemmatización* busca en las palabras las bases del conocimiento léxico para obtener palabras cuya forma básica es correcta en el idioma correspondiente [3, 4].

En este proyecto, este tipo de preprocesado sería el adecuado para los datos que se tienen. Sin embargo, para el lenguaje español no se ha determinado ningún método conocido o válido.

En esta parte del trabajo, en el que se preparan los datos de entrada, no se ha aplicado este preprocesado. En cambio, aplicando otras técnicas de preprocesado, posteriormente se decide

(en la vectorización de los textos con *tf-idf*) si aplicar *stemming* mejora o no los resultados de clasificación (veáse la Subsección 4.1.1).

2.2.2. Otras técnicas determinadas

Creación de diccionarios de palabras

De manera más específica, se han aplicado métodos de preprocesado de los textos atendiendo al tipo de vocabulario y al ámbito de los mismos.

En este caso, las librerías *pandas* y *numpy* y el propio lenguaje de *Python* permiten la creación de ficheros donde se establece una sustitución de las palabras o expresiones originales con otras pertenecientes al mismo campo semántico. Esta sustitución se lleva a cabo a través de expresiones regulares como se ha mencionado en el punto **Expresiones regulares**.

Las sustituciones de estas palabras se han aplicado a entidades como nombres de colegios e institutos, lugares, coordenadas de carretera, municipios o localidades que aparecen en el conjunto de datos. Para este mapeo, las expresiones utilizadas como reemplazo han sido *+colegio*, *+ies*, *+carretera* y *+localidad* (ej: *colegio pintor agustín riancho*, *ies josé zapatero domínguez*, *castro urdiales*).

Es conveniente resaltar que para aplicar este tipo de preprocesado se necesita llevar a cabo un pre-análisis de los datos de partida, observando las expresiones o los conjuntos de palabras que aparecen con mayor frecuencia en los diferentes textos para ver en qué categoría han sido clasificados. Este análisis previo de los datos permite conocer también, a simple vista, patrones de vocabulario que aparecen siempre cuando el texto se clasifica en una u otra categoría, y de este modo, en un futuro, llegar a clasificar correctamente nuevos textos de ámbito similar.

Corrección de palabras

En algunos casos se han detectado palabras mal escritas y esto conlleva un aumento del vocabulario innecesario. Por esta razón, también se ha creado un diccionario en el que se aplica la corrección de ciertas palabras (por ejemplo, *adeucación* por *adecuación*).

No obstante, una vez se hayan aplicado estas técnicas de preprocesado, la precisión en los resultados no ha variado apenas. De esta manera, hasta que no se ha eliminado el vocabulario con menor relevancia para la clasificación, no ha mejorado la precisión. Para encontrar este vocabulario y la métrica que mide la relevancia de estas palabras, se ha aplicado lo que se conoce como *Feature Importance* y que se detalla a continuación.

2.2.3. *Feature importance*

La importancia de cada característica en problemas de clasificación de textos se puede obtener de forma sencilla, y resulta muy útil para preprocesar los textos.

Esta propiedad atribuye una puntuación para cada característica de los datos (cada palabra de los textos), y cuanto mayor sea, más relevante es la característica hacia la clase o categoría de salida. Para obtener la importancia de cada característica a la hora de clasificar los textos se utiliza una propiedad contenida en el modelo [5].

En las siguientes tablas se muestran los resultados de aplicar este método. Las Tablas 2.5 y 2.6 recogen las 10 características con máxima puntuación en base a las dimensiones *Naturaleza del gasto* y *Objeto de administración (nivel 2.1)*, respectivamente.

feature	**score**	**feature**	**score**
+ayuntamiento	0.062190	menor	0.020104
abastecimiento	0.032659	saneamiento	0.019508
contrato	0.029665	red	0.016538
construccion	0.025866	tramo	0.016146
+localidad	0.023815	agua	0.013557

Tabla 2.5: Diez primeras *features* con la mayor puntuación hacia la categoría de salida con respecto a la dimensión *Naturaleza del gasto*.

feature	**score**	**feature**	**score**
+colegio	0.067849	reparacion	0.020934
+numero_km	0.063616	puerto	0.017488
+carretera	0.061810	+localidad	0.017294
+instituto	0.041292	red	0.015579
+ayuntamiento	0.022511	saneamiento	0.014165

Tabla 2.6: Diez primeras *features* con la mayor puntuación hacia la categoría de salida con respecto a la dimensión *Objeto de administración* y el *nivel 2.1*.

Del mismo modo, existen 525 *features* (de 1318) que tienen puntuación 0.0 correspondientes a la dimensión *Naturaleza del gasto* y 528 *features* para la dimensión *Objeto de administración* y el *nivel 2.1*. Algunos de estos ejemplos de palabras son: *acometidas, acondicionado, adquisición* o *adjudicación* en el caso de la dimensión *Naturaleza del gasto* y, del mismo modo, las palabras *acera, actos, adaptaciones* o *acereda* para la dimensión *Objeto de administración* y el *nivel 2.1*.

Por otro lado, se pueden visualizar gráficamente estos resultados a través de diagramas de barras que muestren la puntuación obtenida (Figuras 2.1 y 2.2).

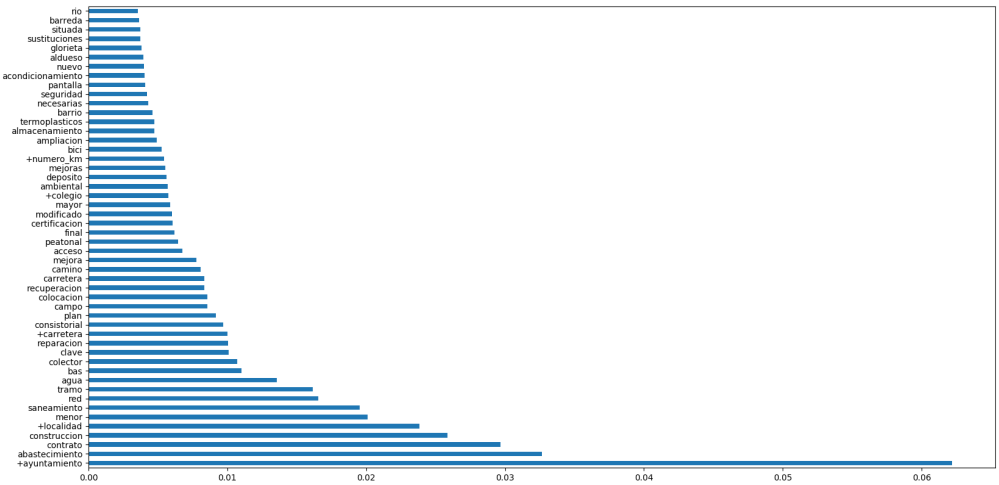

Figura 2.1: Diagrama de barras de las cincuenta características con mayor puntuación con respecto a la categoría de salida para la dimensión *Naturaleza del gasto*.

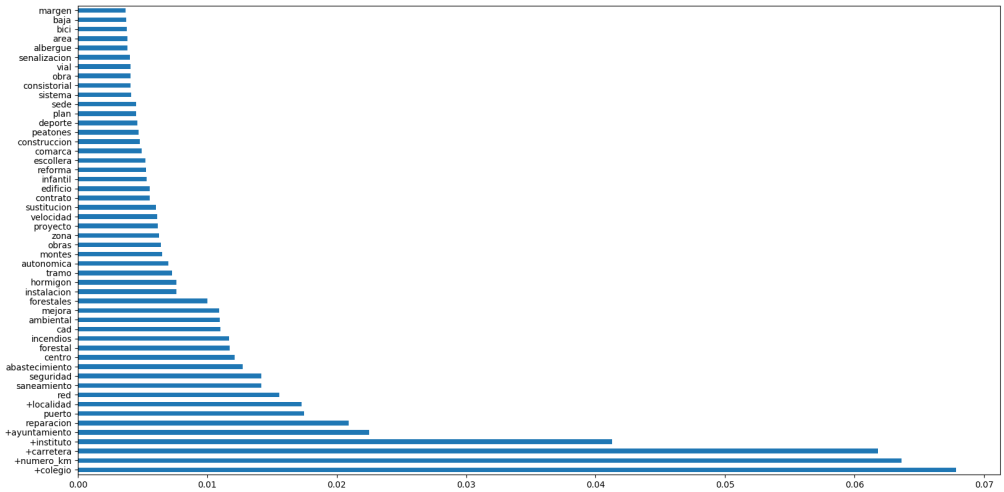

Figura 2.2: Diagrama de barras de las cincuenta características con mayor puntuación con respecto a la categoría de salida para la dimensión *Objeto de administración* y el *nivel 2.1*.

Los clasificadores de árboles, sin entrar en profundidad, también permiten conocer qué vocabulario es más relevante a la hora de clasificar en una u otra clase a través de la computación de diagramas de árboles [6]. A continuación, se muestra la elaboración de un diagrama para la dimensión *Naturaleza del gasto* (veáse la Figura 2.3).

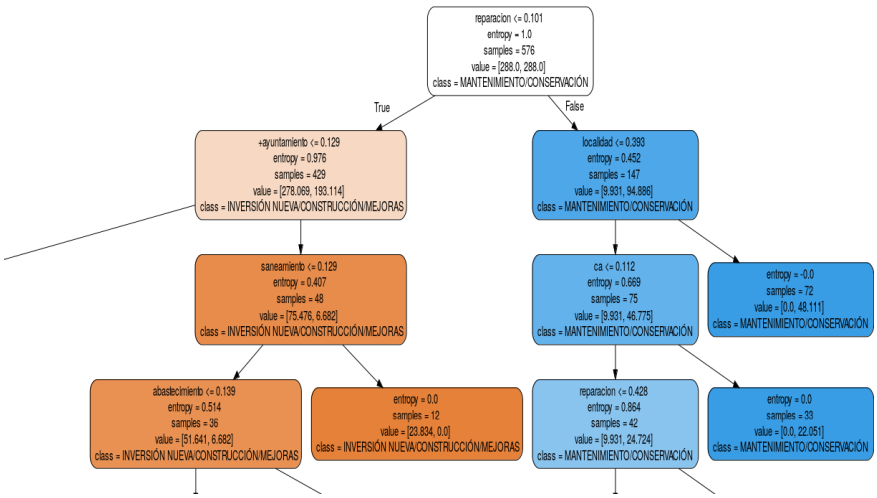

Figura 2.3: Parte del diagrama de árbol que muestra las palabras más relevantes para clasificar en una u otra categoría en base a la dimensión *Naturaleza del gasto* [7].

En la Figura 2.3 se pueden observar los diferentes términos que entran en juego para consecuentemente clasificar en una u otra categoría. A través de la medida de *entropía* (veáse la Sección 3.3) y de los pesos de cada característica se puede determinar bajo qué condiciones el texto de entrada se clasifica en una categoría o en otra.

Por ejemplo, en el nodo raíz se encuentra el término *reparacion*. Según el valor del peso que tenga el término en el texto que se analiza, se salta al siguiente nivel del árbol siguiendo una de las dos categorías: si el peso de *reparacion* es <=0.101, el texto momentáneamente es clasificado en la clase *INVERSIÓN*, mientras que si ese valor es mayor, el texto es clasificado en la clase *MANTENIMIENTO* en el siguiente nivel de profundidad. La entropia la calcula el algoritmo para decidir cuál será, en cada paso, el atributo elegido para bifurcarse. Progresivamente, se analizan todos los términos que aparecen, su correspondiente peso en el texto y el valor de la entropía para observar si se gana información (si la entropía se acerca a 0) o se pierde (si la

entropía se acerca al valor máximo posible: veáse expresión de la Sección 3.3), a medida que se avanza en el árbol hasta llegar a los nodos finales que determinan las categorías de clasificación.

Del mismo modo, se pueden observar las palabras (Figura 2.4) que aparecen con mayor frecuencia en los documentos antes de aplicar el preprocesado de los datos. Nótese que algunos términos, aunque aparezcan muchas veces, no aporten información o no son relevantes.

Figura 2.4: Nube de palabras que aparecen con mayor frecuencia en la colección de documentos creado con la librería *wordcloud* [8].

En el presente proyecto se ha aplicado otro ejemplo de preprocesado de textos teniendo en cuenta el método expuesto anteriormente. El preprocesado ha consistido en eliminar del conjunto de datos el vocabulario cuya puntuación era 0.0. Sin embargo, únicamente se han eliminado 100 palabras con puntuación 0.0 debido a que el borrado de todas las palabras con dicha puntuación significaría la desaparición de algunos textos de entrada dado que son textos descriptivos muy cortos.

En la siguiente sección se refleja aun más la importancia que puede tener el vocabulario en los diferentes textos en un problema de clasificación. Para ello, se va a hacer uso de la frecuencia de los términos o palabras a través del cálculo de la medida *tf-idf*.

2.3. Vectorización de los datos

En un problema de clasificación de textos, los algoritmos de aprendizaje que se emplean no pueden procesar este tipo de datos en su estructura original. Por un lado, los datos textuales no contienen ningún formato y su longitud es variable en cada caso y, por otro lado, los clasificadores únicamente permiten vectores de características numéricas con un tamaño fijo. Por esta razón, los textos se transforman en vectores numéricos y se hace a partir de una medida denominada *frecuencia de término-frecuencia de documento inversa (term frequency-inverse document frequency: tf-idf)* [9].

2.3.1. *Term frequency-inverse document frequency: tf-idf*

La *frecuencia de término-frecuencia de documento inversa* es una medida estadística numérica que se utiliza para mostrar la importancia que tiene una palabra en un documento dentro de un conjunto (en este ámbito se denomina *corpus*). El valor del *tf-idf* refleja esta importancia a través de un factor de ponderación que aumenta proporcionalmente las veces que aparece una palabra en el documento mientras que, por otro lado, se compensa con el número de documentos del *corpus* que contienen la palabra. De este modo, se atribuyen unos pesos que sirven

para ajustar el hecho de que algunas palabras aparecen con mayor frecuencia que otras y, sin embargo, no sean tan relevantes.

El *tf-idf* se calcula a través del producto de dos medidas estadísticas, *frecuencia de término* y *frecuencia de documento inversa* [10].

- *Frecuencia de término (tf)*: indica el número de veces que una palabra aparece en cada documento. Sin embargo, existen diferentes variaciones en su cálculo ya que el tamaño de los textos varía mucho. A continuación, se muestra una tabla con los diferentes cálculos de peso que se pueden obtener para esta medida:

Diferentes cálculos del peso frecuencia de término(*tf*)	
Esquema de ponderación	**Peso *tf***
Frecuencia booleana	$f_{t,d} = 1$ si $t \in d$, $f_{t,d} = 0$ si $t \notin d$
Recuento bruto	$f_{t,d}$
Frecuencia del término	$\frac{f_{t,d}}{\sum_{t' \in d} f_{t',d}}$
Normalización logarítmica	$1 + log(f_{t,d})$
Doble normalización 0,5	$\frac{0{,}5+0{,}5 f_{t,d}}{max_{t' \in d} f_{t',d}}$
Doble normalización k	$\frac{k+(1-k) f_{t,d}}{max_{t' \in d} f_{t',d}}$

Tabla 2.7: Variantes del cálculo del peso frecuencia de término (tf) [11].

En general, los dos métodos que más se emplean para este tipo de problemas son el recuento bruto y la normalización logarítmica. En el primer caso, se tiene el número de veces $f_{t,d}$ que aparece el término t en el documento d y se le asigna ese valor como peso. En el segundo caso, se establece el peso de cada término en función del número de veces que aparece en el documento, siendo este valor proporcional y escalado para equilibrar todos los pesos (tanto de las palabras que no aparecen como de las que sí aparecen): si la palabra aparece, se le asigna peso igual a $1 + log(f_{t,d})$ en el documento, si no aparece, el peso asignado es 0.

- *Frecuencia de documento inversa (idf)*: indica la cantidad de información que proporciona la palabra, es decir, si la palabra es muy común o si aparece muy pocas veces en los documentos de la colección. La manera habitual de medir este valor es a través de la fracción inversa en escala logarítmica del número de textos que contienen la palabra. Se calcula aplicando el logaritmo al valor de dividir la cantidad total de documentos por el número de documentos que contienen el término. Sin embargo, existen otras variantes de cálculo que se exponen a continuación:

Diferentes cálculos del peso: frecuencia de documento inversa (*idf*)	
Esquema de ponderación	**Peso *idf*** ($n_t = \mid \{d \in D : t \in d\} \mid$)
Unitaria	1
Frecuencia de documento inversa	$log(\frac{N}{n_t})$
Frecuencia de documento inversa suave	$log(\frac{N}{1+n_t})$
Frecuencia de documento inversa máxima	$log(\frac{max_{t' \in d} n'_t}{1+n_t})$
Frecuencia de documento inversa probabilística	$log(\frac{N-n_t}{n_t})$

Tabla 2.8: Variantes del cálculo del peso frecuencia de documento inversa (idf) [11].

En la práctica, el cálculo del *idf* utiliza la frecuencia de documento inversa y se puede decidir si se calcula o no. En este caso, se ha probado a realizar el cálculo de las dos formas. De este modo, el resultado global del *tf-idf* seguiría dos esquemas diferentes que se exponen en el siguiente punto.

- *Frecuencia de término - frecuencia de documento inversa (tf-idf)*: es el valor que resulta de hacer el producto de las medidas *tf* e *idf*.

Un peso alto de *tf-idf* significa una frecuencia de término alta (en el texto dado) y una frecuencia de documento inversa baja del término en el corpus. De este modo, los pesos tienden a filtrar las palabras más comunes.

Los esquemas que se siguen en la práctica son los siguientes:

Esquemas de *tf-idf*
$1 + log(f_{t,d})$
$(1 + log(f_{t,d}))log(\frac{N}{n_t})$

Tabla 2.9: Esquemas de ponderación de *tf-idf*.

Por otro lado, para el cálculo de esta medida se ha utilizado el método *TfidfVectorizer()* de la librería *sklearn* de Python. Este método es parametrizable y se pueden aplicar una o varias de las opciones anteriormente descritas con diferentes parámetros. A continuación, se detallan los parámetros que se utilizan en la práctica:

- **tokenizer**: aplica un método de tokenización de los datos al que se le puede pasar el preprocesado de *stemming* si se desea. En este paso es donde se decide aplicar *stemming* o no y observar si los resultados mejoran o empeoran en cada caso.

- **ngram_range**: divide los textos en n-gramas. Un *n-grama* es una subsecuencia de n palabras a partir de una secuencia de palabras anteriores. En este caso, se establecen los *tokens* de una palabra denominado *unigrama* ($n = 1$) y las expresiones de dos palabras denominado *bigrama* ($n = 2$) del siguiente modo: **(1,2)**. El límite inferior establece los unigramas y el límite superior los bigramas.

Para el cálculo de un modelo *n-grama* se utilizan las probabilidades condicionadas de las palabras. Concretamente, un *bigrama* se calcula con la aproximación de la probabilidad de una palabra dadas todas las anteriores, utilizando solamente la probabilidad condicional del término anterior. Más formalmente, cuando se utiliza el modelo *bigrama* para predecir estadísticamente la siguiente palabra, se realiza con la siguiente aproximación:

$$P(w_k|w_1^{k-1}) \approx P(w_k|w_{k-1}) \tag{2.1}$$

donde w_k es el término cuya probabilidad se quiere calcular, w_1^{k-1} es el conjunto de todas las palabras anteriores de una secuencia sucedida hasta el momento, y w_{k-1} es la palabra previa a w_k.

La suposición anterior de que la probabilidad de una palabra depende solamente de la palabra anterior se denomina presunción de Markov y consiste en predecir la probabilidad de alguna palabra futura sin tener que observar palabras muy lejos en el pasado. Generalizando, la fórmula de los *N-gramas* es [12]:

$$P(w_k|w_1^{k-1}) \approx P(w_k|w_{k-N+1}^{k-1}) \tag{2.2}$$

- **sublinear_tf**: establece el esquema del cálculo *tf*. Si es *True*, utiliza la normalización logarítmica $1 + log(f_{t,d})$. Si es *False* (por defecto), utiliza la frecuencia del término. En la práctica, se establece **True**.

- **use_idf**: habilita la ponderación de *idf*. Si no se aplica (*use_idf=False*), se utiliza el primer esquema de la Tabla 2.8. En caso contrario, *use_idf=True*, se utiliza el segundo esquema de la Tabla 2.8. Por defecto, se habilita el cálculo.

- **smooth_idf**: suaviza las ponderaciones o pesos de *idf*. Por defecto, se establece *True*, es decir, se habilita el suavizado en las ponderaciones cuando *use_idf=True*, también por defecto [13]. En la práctica, se establece a **False**.

Una vez se preprocesa y se crea el conjunto de datos como se indica en la Sección 2.2, la aplicación de estos parámetros es determinante en los resultados finales puesto que este método realiza un tratamiento de las palabras y su objetivo, como se ha mencionado anteriormente, es reflejar la importancia de una palabra en un texto. Finalmente, el algoritmo muestra una precisión de clasificación en base a esta importancia y a la relación de las palabras con la categoría correspondiente.

2.3.2. Visualización de los datos: T-SNE

Considerando la aplicación de *tf-idf*, el conjunto de datos ya no tiene estructura de texto y forman un conjunto de datos numéricos, en concreto, una matriz de vectores de puntos. A través de la librería *manifold* que contiene el método *TSNE*, se puede visualizar la nube de puntos que representa, a simple vista, si los documentos pueden ser fácilmente categorizados o no, es decir, si *los puntos son linealmente separables* o no.

En el caso de la primera dimensión, *Naturaleza del gasto*, se muestran las siguientes imágenes.

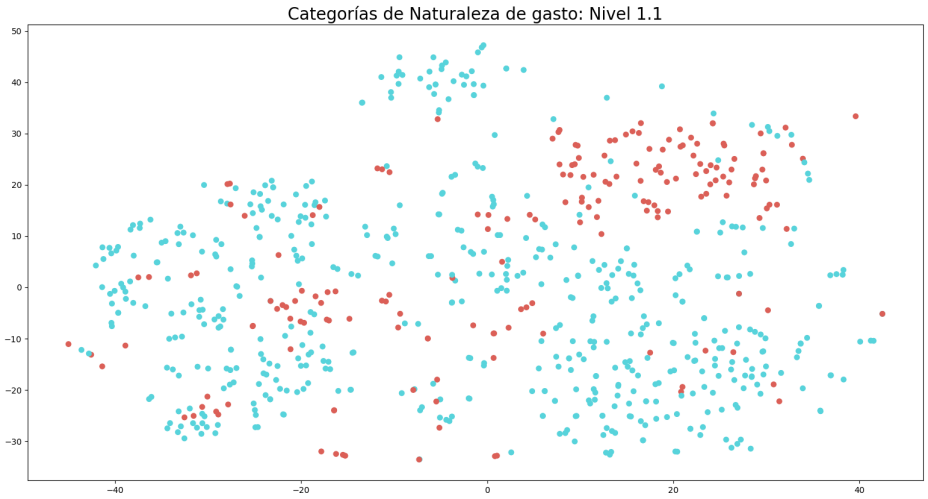

Figura 2.5: Representación en 2D de los textos a través del cálculo de *tf-idf* para la dimensión *Naturaleza del gasto* y las categorías correspondientes. Punto rojo: INVERSIÓN NUEVA/CONSTRUCCIÓN/MEJORAS, Punto azul: MANTENIMIENTO/CONSERVACIÓN/INVERSIÓN DE REPOSICIÓN.

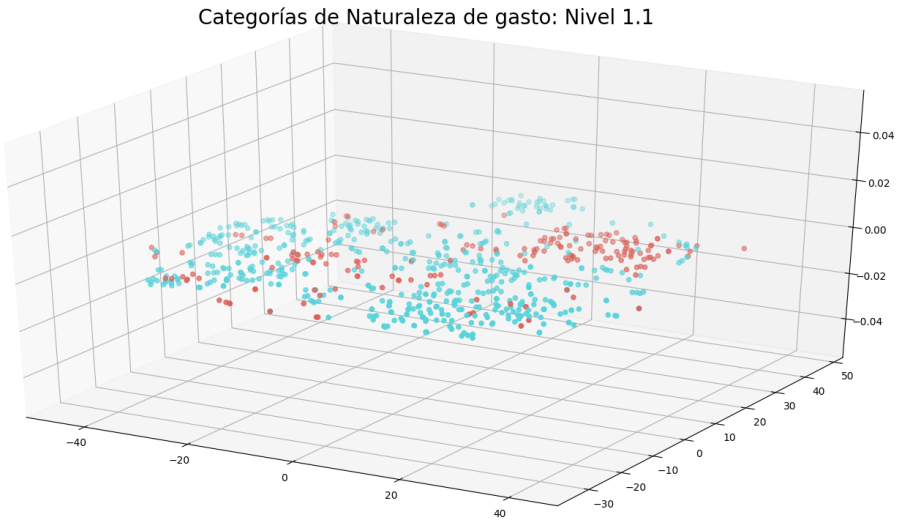

Figura 2.6: Representación en 3D de los textos a través del cálculo de *tf-idf* para la dimensión *Naturaleza del gasto* y las categorías correspondientes.

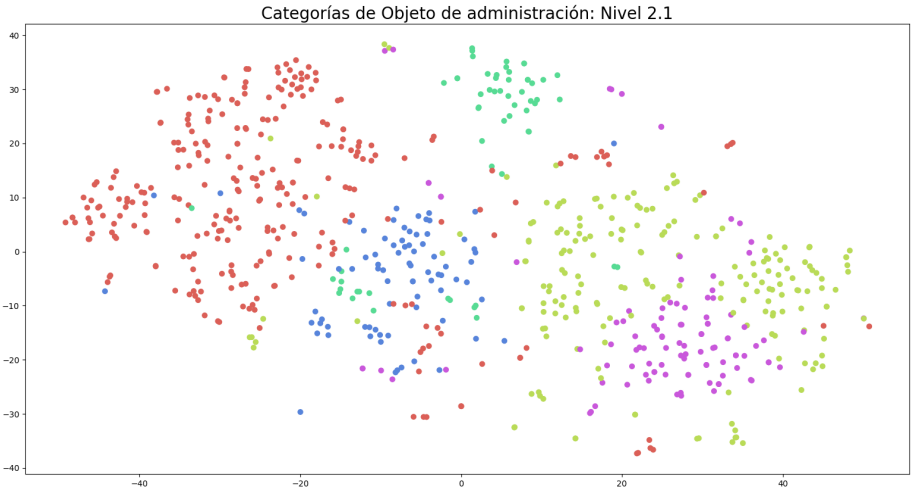

Figura 2.7: Representación en 2D de los textos a través del cálculo de *tf-idf* para la dimensión *Objeto de administración*, el *nivel 2.1* y las categorías correspondientes.

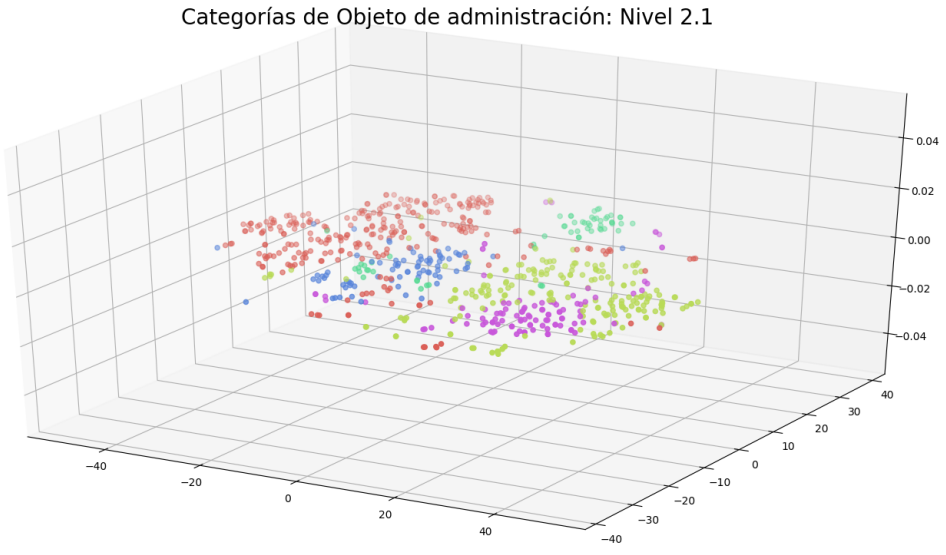

Figura 2.8: Representación en 3D de los textos a través del cálculo de *tf-idf* para la dimensión *Objeto de administración*, el *nivel 2.1* y las categorías correspondientes.

Se puede observar en las Figuras 2.5 y 2.6 que, a simple vista, los textos no son linealmente separables.

Además, también se puede observar un desequilibrio de tamaño en los datos de una y otra categoría. Este desbalanceo en los datos se trata en el siguiente apartado.

Por el contrario, si se observan las Figuras 2.7 y 2.8 para la dimensión *Objeto de administración* y el *nivel 2.1* el resultado es diferente. En esta ocasión, los datos sí parecen linealmente separables y se espera que la precisión en los resultados sea mejor que para la primera dimensión.

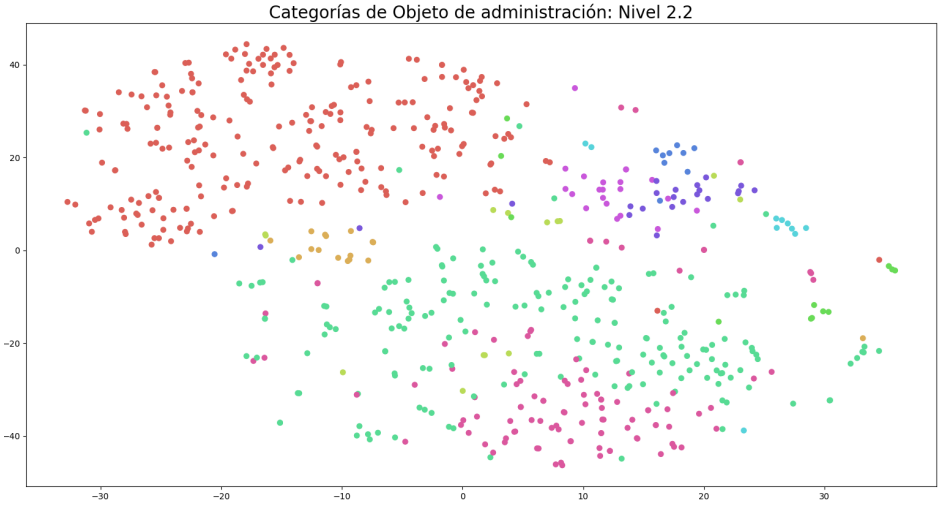

Figura 2.9: Representación en 2D de los textos a través del cálculo de *tf-idf* para la dimensión *Objeto de administración*, teniendo en cuenta ambos niveles, 2.1 y 2.2, y las categorías correspondientes.

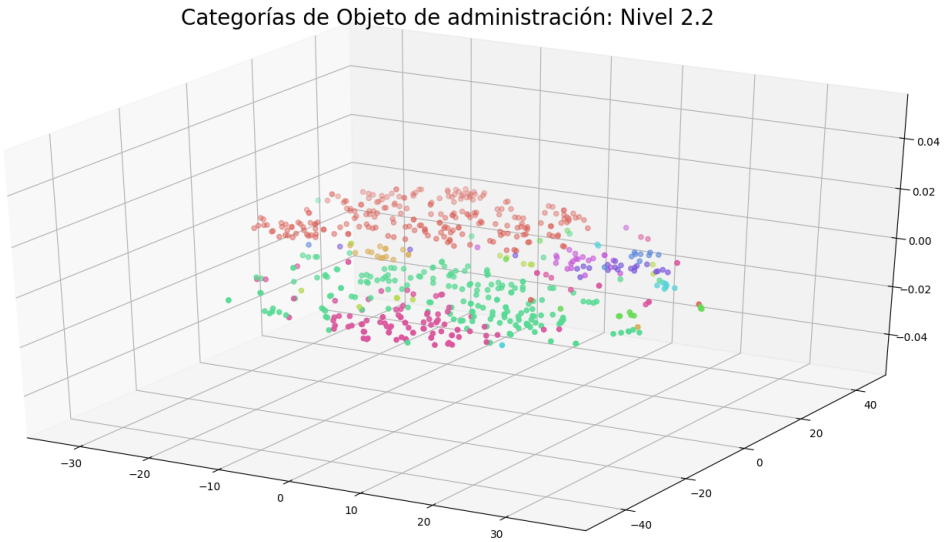

Figura 2.10: Representación en 3D de los textos a través del cálculo de *tf-idf* para la dimensión *Objeto de administración*, teniendo en cuenta ambos niveles 2.1 y 2.2 y las categorías correspondientes.

Para la dimensión *Objeto de administración*, teniendo en cuenta un mayor número de categorías correspondientes a la unión del *nivel 2.1* y *nivel 2.2*, en las Figuras 2.9 y 2.10 se observa que los textos pertenecientes a las clases mayoritarias están claramente definidos. Sin embargo,

los textos de las clases minoritarias tienen sus puntos un poco más dispersos. Con esta visualización, cabe esperar que la clasificación de los textos de las clases mayoritarias proporcione una mayor precisión y mejor resultado que aquellos que pertenecen a categorías minoritarias.

2.3.3. Balanceo de datos

En el apartado anterior se ha podido observar un desequilibrio de categorías en el conjunto de datos, es decir, existen algunas categorías que contienen muy pocas muestras para entrenar. Esta situación es la que provoca un desbalanceo en los datos y que afecta a los algoritmos en el proceso de generalización de la información. De esta manera, las clases minoritarias son las perjudicadas en los resultados finales puesto que cuando se tiene un conjunto de datos desequilibrado, se suele obtener una precisión alta en la clase mayoritaria y un *recall* bajo (una de las medidas que se utiliza para la obtención de los resultados finales) en la clase minoritaria.

A continuación, se muestra un ejemplo de clasificación de datos desequilibrados para la dimensión *Objeto de administración* según los *niveles 2.1* y *2.2*. En este ejemplo, se utiliza el clasificador de *Regresión Logística* (*LogisticRegression()* de *sklearn*) y los resultados muestran la matriz de confusión, los valores de *precision* y *recall* y el *accuracy* alcanzado.

id	**categoría**	**nº textos en la categoría**
1	CENTROS/EDIFICACIONES/INSTALACIONES - C.A. CANTABRIA-DEMANIALES	192
2	CENTROS/EDIFICACIONES/INSTALACIONES - OTRAS ADMINISTRACIONES-DEMANIALES	105
3	OBRA CIVIL/ INFRAESTRUCTURAS - C.A. CANTABRIA- INFRAESTRUCTURAS PORTUARIAS	16
4	OBRA CIVIL/ INFRAESTRUCTURAS - C.A. CANTABRIA-ABASTECIMIENTOS	14
5	OBRA CIVIL/ INFRAESTRUCTURAS - C.A. CANTABRIA-INFRAESTRUCTURA VIAL	232

- **Matriz de confusión**:

```
id
1       [[192   0   0   0   0   0   0   0   0   0   0]
2        [ 21  84   0   0   0   0   0   0   0   0   0]
3        [  7   0   4   0   5   0   0   0   0   0   0]
4        [ 12   0   0   0   2   0   0   0   0   0   0]
5        [  4   0   0   0 228   0   0   0   0   0   0]]
```

- ***Accuracy***: 0.8333333333333334 (de las once categorías aunque solo se muestren cinco en el ejemplo)

- ***precision y recall***:

id	*precision*	*recall*
1	0.674	1.000
2	1.000	0.800
3	1.000	0.250
4	0.000	0.000
5	0.927	0.983

En este ejemplo se puede observar que para las categorías minoritarias, 3 y 4, el valor de *recall* es muy bajo (0.25 y 0) y que para el resto de categorías el valor de *precision* es relativamente alto (0.674, 1 y 0.927).

Además, se puede observar en la matriz de confusión que para la categoría 4, una de las categorías minoritarias, no hay ningún acierto y que todos los textos pertenecientes a dicha categoría han sido mal clasificados.

Existen diferentes formas de solventar este desequilibrio en los datos. En este proyecto, para resolver este desbalanceo en los datos se ha optado por ajustar uno de los parámetros de los algoritmos que se utilizan (aquellos que lo permiten): *class_weight='balanced'*. Este parámetro intenta equilibrar a la clase minoritaria, penalizando a la clase mayoritaria en el entrenamiento del conjunto de datos [14].

Por otro lado, en los datos existían diferentes valores atípicos (*outliers*) para clasificar en una u otra dimensión. Estos valores han sido eliminados antes de crear el conjunto de datos preprocesado. Algunos de estos valores atípicos han sido:

- Texto mal clasificado según la dimensión: en la dimensión *Naturaleza del gasto*, existía un texto clasificado según el *nivel 2.1* de la dimensión *Objeto de administración*.
- Textos no clasificados en ninguna categoría: en la dimensión *Objeto de administración* existían dos textos que no estaban clasificados según el objeto del contrato.
- Categorías con muy pocas muestras (uno, dos o seis textos por categoría): en la dimensión *Objeto de administración*, para la clasificación según ambos niveles, existían categorías con muy pocas muestras para entrenar. En este caso, se ha optado por eliminar estos textos. Sin embargo, seguían existiendo categorías minoritarias y también se ha aplicado el balanceo comentado anteriormente.

2.4. División de los datos

En un problema de clasificación, el objetivo no es obtener unos resultados del conjunto de datos con el que se entrene, sino que dichos resultados se validen y que el problema pueda ser generalizado a otro conjunto de datos. Por esta razón, es importante la división de los datos de partida: qué y cuánto del conjunto de datos se utiliza para entrenar el modelo y qué y cuánto se utiliza para realizar las comprobaciones y observar si el modelo generaliza la información o no. Existen diversas técnicas de particionado de datos en este tipo de problemas. Este capítulo se va a centrar en detallar dos técnicas de división: particionado en dos conjuntos, entrenamiento y prueba, según una proporción dada y la técnica de validación cruzada, concretamente el método *Leave One Out*.

2.4.1. Entrenamiento/Prueba

Generalmente, el conjunto de datos se divide en datos de entrenamiento y datos de validación o prueba. El primer conjunto de datos se utiliza para obtener un modelo y comprobar si dicho

modelo se ajusta sobre estos datos y, posteriormente, poder generalizarse y ajustarse a nuevos datos. Por el contrario, el segundo conjunto sirve para probar el modelo y observar si los resultados son buenos o malos.

Para este proyecto, se ha realizado una partición de manera aleatoria del 80 % del conjunto total para los datos de entrenamiento y el 20 % restante para el conjunto de prueba. Por otro lado, esta técnica se ha utilizado para clasificar los textos de la dimensión *Naturaleza del gasto*. En esta dimensión, existía un desequilibrio de los datos de ambas categorías. Por esta razón, para realizar la división en datos de entrenamiento y datos de prueba, la proporción de textos de una categoría y otra ha sido la misma para ambos conjuntos. Por ejemplo, si 181 textos pertenecen a la categoría minoritaria y 539 textos pertenecen a la categoría mayoritaria, el 80 % de los 181 textos de la primera categoría y el 80 % de los 539 textos de la segunda categoría formarían parte del conjunto de entrenamiento. De esta forma, se evita que el conjunto de entrenamiento (o el de validación) contengan únicamente textos de la clase mayoritaria (y de la clase minoritaria en el caso del conjunto de validación).

Sin embargo, con esta técnica una vez se realizaba el entrenamiento y la validación de los resultados, se observaba que los resultados mejoraban o empeoraban sorprendentemente dependiendo de la partición aleatoria del principio que se realiza. Los valores del *accuracy* variaban entre 0.85 y 0.93 cuando se probaba con diferentes particiones aleatorias.

2.4.2. Validación cruzada: Leave One Out (LOO)

La técnica de validación cruzada se utiliza para solventar algunos problemas como el de la generalización del modelo. Esta técnica es muy similar a la partición en entrenamiento y validación pero se aplica a un mayor número de subconjuntos.

La validación cruzada consiste en dividir el conjunto de datos en k subconjuntos, entrenar en *k-1* subconjuntos y validar en el restante. Esta operación se hace por cada uno de los subconjuntos posibles y después se realiza la media de los resultados obtenidos.

Existen diferentes métodos de validación cruzada. Sin embargo, este proyecto se centra en utilizar el método *Leave One Out (LOO)*. En este método el número de subconjuntos es igual al número de datos en el conjunto. Por lo tanto, se entrena con todos los datos menos uno donde se valida cada vez y después se realiza el promedio para obtener el error del modelo. Desde el punto de vista computacional, el método *LOO* es muy costoso ya que depende del número de muestras del conjunto de datos. Por esta razón, esta técnica debe usarse en conjuntos de datos pequeños como es el caso que aborda este proyecto (721 textos) [15].

Esta técnica se ha utilizado en los resultados finales para los diferentes problemas de clasificación que se muestran en este proyecto: categorización respecto de la dimensión *Naturaleza del gasto*, respecto al *nivel 2.1* de la dimensión *Objeto de administración* y, por último, respecto a los *niveles 2.1* y *2.2* conjuntamente de la dimensión *Objeto de administración*.

3

Algoritmos de aprendizaje automático

En este capítulo se exponen los diferentes algoritmos que se han utilizado para realizar la clasificación de los contratos del sector público. Estos algoritmos forman parte del aprendizaje supervisado que trabaja con datos previamente etiquetados o categorizados como es el caso del problema de clasificación que se expone en este proyecto. A continuación, se detalla cada uno de los algoritmos que se utilizan.

3.1. Regresión Logística: *LogisticRegression()*

El clasificador de Regresión Logística lineal es un modelo de clasificación supervisada basado en el modelo lineal. Mientras que la regresión lineal se utiliza para predecir una variable cuyos valores son continuos, este clasificador de regresión logística lineal se encarga de predecir una variable de tipo categórico, es decir, que toma valores discretos de clases o categorías con respecto a otras variables independientes denominadas *predictores*. Por otro lado, la predicción de salida del modelo de regresión logística lineal viene dado por las siguientes fórmulas:

$$Y = X\beta^T + b \tag{3.1}$$

$$Z = logistic(Y) \tag{3.2}$$

donde Z es la variable de salida para todos los elementos del conjunto de datos, X es una matriz cuyas dimensiones son el número de elementos del conjunto de datos por el número de características para cada elemento, β^T es un vector columna que contiene coeficientes con dimensión el mismo número de características (recordar que la fórmula de la regresión lineal es 3.1), b es el término *bias* y *logistic* representa la función logística de predicción que depende del tipo de problema de clasificación.

En la regresión logística existen dos tipos de problemas de clasificación: binario y multinomial. El clasificador de regresión logística binaria se utiliza en problemas de clasificación binaria donde la variable categórica que se predice consta de dos únicas categorías que se identifican con los valores 0 y 1 (por ejemplo el animal es gato: 0, perro: 1). Por otro lado, el clasificador de regresión logística multinomial se utiliza en clasificación múltiple. En la clasificación múltiple, la variable categórica establece más de dos clases de predicción (por ejemplo, la flor es un/a rosa: 0, margarita: 1, girasol: 2, etc.).

Los modelos de regresión logística se clasifican según el número de categorías que se pueden predecir y se utiliza una función de predicción diferente según el tipo de clasificación.

Para problemas de clasificación y regresión logística binaria, la función que se utiliza es la *sigmoidal*. La *función sigmoide* toma valores entre 0 y 1 y la curva que produce tiene forma de S. La fórmula correspondiente para esta función es $\frac{1}{1+e^{-z}}$ y su representación gráfica, para un ejemplo en concreto, se puede observar en la Figura 3.1.

La idea en este tipo de problemas de regresión logística binaria, donde las variables de entrada toman valores continuos entre 0 y 1 pero la variable de salida toma valores discretos (0 o 1), es establecer un valor denominado *threshold*. De esta manera, los valores que se encuentran por encima de este punto de corte pertenecen a la clase del 1, mientras que valores por debajo de dicho punto de corte se clasifican en la clase del 0.

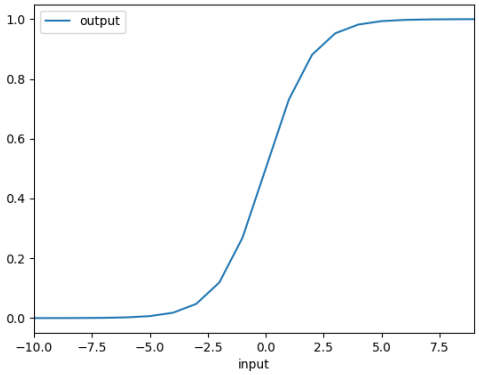

Figura 3.1: La salida de una función logística para una entrada continua.

Por otro lado, en problemas de clasificación multinomial la función que se utiliza se denomina *Softmax*. La función *Softmax* calcula las probabilidades de cada categoría sobre todas las categorías posibles. Estas probabilidades son las que determinan la predicción de la variable categórica según los datos de entrada y su fórmula es la siguiente:

$$P(y_i = k) = \frac{e^{x_i \beta_k^T}}{\sum_{j=1}^{K} e^{x_i \beta_j^T}} \tag{3.3}$$

donde k es una determinada clase y K indica el número total de clases [16].

En los tres problemas de clasificación que se muestran en este proyecto, se van a utilizar ambos modelos de regresión logística. En el caso de la dimensión *Naturaleza del gasto*, la variable categórica consta de dos únicas clases posibles. La clasificación binaria es la que se utiliza en este problema. Por otro lado, en el caso de los dos problemas de clasificación que conciernen a la dimensión *Objeto de administración*, la categorización que se aplica es la clasificación múltiple debido a que el número de categorías que se predicen es mayor que dos (seis en el primer caso y once en el segundo).

En la práctica, la forma de especificar si la clasificación es binaria o múltiple reside en un parámetro del algoritmo *LogisticRegression()* de *sklearn: multi_class = 'auto'*. De esta manera, auto distingue cuando el problema es de regresión logística binaria o multinomial.

3.2. Máquinas de soporte vectorial (*SVM*): *SGDClassifier()*

La idea general de un modelo *SVM* es predecir las categorías a las que pertenecen los datos a través de una función lineal (3.1) conocida como *hiperplano*. De este modo, los hiperplanos

$X\beta^T + b = +1/-1$, actúan como separadores de clases: cuando $X\beta^T + b \geq +1$ se clasifica en una clase y, cuando $X\beta^T + b < -1$, en otra.

Más formalmente, un clasificador *SVM* es un algoritmo de aprendizaje supervisado que se encarga de construir un hiperplano o conjunto de hiperplanos, en un espacio de características, que separen claramente las categorías de puntos. Geometricamente, se puede entender esta separación como la distancia del hiperplano al punto x, siendo β un vector ortogonal al plano. De este modo, si dos clases están *linealmente separadas*, la anchura entre las dos clases, que se representa mediante el valor $\frac{1}{||\beta||}$, es lo más grande posible y, por tanto, el objetivo es encontrar el valor óptimo de β minimizando la norma de β ($||\beta|| = \sqrt{\beta_1^2 + \beta_2^2 + .. + \beta_m^2}$). Del mismo modo, el hiperplano óptimo únicamente depende de los puntos que están sobre los hiperplanos soporte ($X\beta^T + b = +1/-1$). A estos puntos se les denomina *vectores soporte*.

Por otro lado, en el momento de buscar el óptimo de β también se intenta minimizar el error que se produce cuando se asignan incorrectamente los datos en una clase. Este error se basa en una función de pérdida, y su estructura es la siguiente:

$$L(\beta, b, \alpha) = \frac{1}{2}||\beta||^2 - \sum_{i=1}^{m} \alpha_i (y_i(x_i\beta^T + b) - 1) \tag{3.4}$$

donde y es la categoría correcta de x. Si x se clasifica correctamente, entonces $y_i(x_i\beta^T + b) \geq 1$ y en la función L se restan valores, de tal modo que se minimiza el error. Por el contrario, si $y_i(x_i\beta^T + b) < 1$, se suman valores a L y el error aumenta [16].

Una de las funciones más importantes del modelo *SVM* es que, si los datos no son *linealmente separables* en el espacio de características de entrada, se recurre a espacios de dimensión mayor donde los puntos sí son *linealmente separables* y se requiere de una función conocida como *kernel trick*.

Para este proyecto, el espacio de características de partida de los datos tiene aproximadamente 1900 dimensiones (número de características/palabras del corpus sin aplicar preprocesado) y un corpus de únicamente 721 textos. Por lo tanto, en este caso, aplicar un modelo *SVM* con *kernel* no aumentaría la dimensión del espacio de entrada a una dimensión superior, sino todo lo contrario. Sin embargo, *SVM* es un clasificador que se utiliza mucho en el ámbito de la categorización de textos sin *kernel*. Únicamente se utiliza la vectorización de los textos para posteriormente entrenar los datos. Por tanto, se ha empleado en este trabajo, obteniendo unos resultados sorprendentemente buenos (ver Capítulo 4).

En el apartado 2.3.2 se podía observar si existía la propiedad *linealmente separable* a través de las gráficas de puntos en 2D y 3D para los diferentes problemas de clasificación planteados.

El algoritmo que se utiliza en la práctica se denomina *SGDClassifier()* de la librería *sklearn*. Para la aplicación de este clasificador se ha utilizado la función de pérdida de *SVM*: $max(0, 1 - y_i(x_i\beta + b_i))$ y el término de regularización L_2 a través de los parámetros *loss='hinge'* y *penalty=* l2 [17].

3.3. Árbol de decision y *Random Forests*

Un *árbol de decision* es un algoritmo supervisado que se utiliza en problemas de clasificación donde las variables objetivo son de tipo categórico y las características o variables predictoras son categóricas o númericas. Se basa en un conjunto de condiciones (*si...entonces*) que se representan como un *árbol* [16].

Este algoritmo es muy eficaz como herramienta de toma de decisiones, debido a que la estructura y visualización del *árbol* ofrecen unos resultados fácilmente interpretables y su implementación es sencilla en comparación con otros modelos de predicción.

Un *árbol* está compuesto por tres elementos esenciales: *nodos, ramas* y *hojas* [16]. La conexión entre estos tres elementos en un diagrama de *árbol* es sencilla. Un *árbol* empieza con un *nodo raíz*, continua por el resto de *nodos de decision* a través de una o varias ramas y finaliza en los *nodos terminales* donde se establecen las reglas de decision. Mientras que el *nodo raíz* y los *nodos de decision* representan las variables y, las ramas los valores categóricos de esas variables, el *nodo terminal* muestra la decisión final que se corresponde con la categoría en un problema de clasificación (veáse Figura 2.3).

La idea principal de este algoritmo es asociar valores similares de la variable categórica final para tomar una decisión específica. Se trata de conseguir una homogeneidad con respecto a la variable objetivo. Para ello, el algoritmo intenta identificar una variable y una clasificación que obtengan una mayor distribución homogénea en relación a la variable objetivo.

El algoritmo es el que se encarga de la elección del *nodo raíz* y de las posibles subdivisiones a través del análisis de los datos y de las categorías de salida. Del mismo modo, si no se fija un nivel de profundidad del árbol (particiones de los *nodos de decision* en diferentes ramas), el algoritmo se encarga de establecerlo y de cuándo parar la subdivisión al encontrar el diagrama óptimo para el problema que se plantea.

Para esta elección y para medir la homogeneidad, el algoritmo utiliza lo que se conoce como *entropía* y su variación con la *Ganancia de información* [16]. A continuación, se muestran las expresiones de ambas métricas:

$$Entropía(S) = \sum -p_i log_2 p_i \tag{3.5}$$

donde S es la entropía total del sistema al principio, p_i es la proporción (sobre el total de observaciones) de la *i-ésima* categoría de la variable objetivo. El rango de valores de la *entropía* es de 0 a $log_2 c$ (c es el número de categorías en la variable objetivo). Cuando las observaciones sean del todo homogéneas (existe un valor $p_i = 1$ y el resto son 0) se alcanza el valor mínimo, mientras que, en caso contrario, se alcanza el valor máximo (todos los valores p_i son iguales).

La *entropía* se reduce a medida que se introducen variables como nodos. Esta reducción es lo que se muestra como una ganancia de información y se mide, para una variable particular V, a través de la siguiente expresión:

$$GananciaInformación(S, V) = Entropía(S) - \sum_c \frac{|V_c|}{|S|} Entropía(V_c) \tag{3.6}$$

donde c representa las categorías de la variable V, V_c es el conjunto de observaciones con la categoría c en el nodo V, y $Entropia(V_c)$ representa la entropía que tienen esas observaciones. Cuanto más grande sea este valor, mayor ganancia de información y mayor poder de discriminación para la siguiente subdivisión en el *árbol*.

Como se ha mencionado anteriormente, los árboles de clasificación se utilizan como método intermedio para obtener una descripción de los datos y como pre-análisis para la toma de decisiones. En este proyecto, debido a que en el entrenamiento de los datos la aplicación de este clasificador no ha obtenido muy buenos resultados en comparación con los resultados obtenidos por otros algoritmos (se exponen algunos resultados en la Sección 4.2), el clasificador se ha aplicado como método intermedio. No obstante, el algoritmo ha servido para obtener el vocabulario más y menos relevante.

En el análisis práctico, el algoritmo que se utiliza es *DecisionTreeClassifier()* de la librería *sklearn*. El análisis consiste en establecer diferentes niveles de profundidad del árbol (de 1 a 10), a través del parámetro *max_depth*, para encontrar la profundidad óptima del árbol (*poda del árbol* a partir de *validación cruzada*) y en evaluar los resultados obtenidos. Del mismo modo, se han establecido diferentes parámetros:

- *min_samples_split*: número mínimo de muestras necesarias para dividir un nodo interno.

- *min_samples_leaf*: número mínimo de muestras necesarias para estar en un nodo final u hoja.

- *criterion*: función de medida de la calidad de una división. En este caso, la *entropy* para la Ganancia de información [18].

Por otro lado, normalmente este tipo de algoritmos presentan diversos inconvenientes. Uno de los problemas más comunes de los árboles de decision es el sobreajuste debido a que tienden a "memorizar" las soluciones, en vez de generalizar el aprendizaje a otros conjuntos de datos cuando se le asigna una profundidad suficiente. Por esta razón, surge otro algoritmo como solución a este problema: el algoritmo *Random Forest*.

La diferencia con el *árbol de decisión* reside en que el algoritmo *Random Forest* es una colección de varios árboles de decisión que trabajan conjuntamente sobre diferentes subconjuntos de la muestra de datos. Este algoritmo (que pertenece al *aprendizaje de conjuntos*: modelo que hace predicciones basadas en otros diferentes) crea árboles con muestras de los datos iniciales obtenidas de manera aleatoria y, posteriormente, realiza el promedio de las predicciones. De esta manera, *Random Forest* está formado por diversos modelos predictivos que tienen como objetivo mejorar la precisión y estabilidad del modelo con respecto a los resultados que ofrece un único *árbol de decisión*.

Una de las características de este modelo es que el tamaño del subconjunto siempre es el mismo que el conjunto de datos. Sin embargo, la composición de los subconjuntos de datos con los que se construyen los árboles se realiza a través de la extracción de los datos con reemplazo (método denominado *bagging* o *bootstrap aggregating* [16]) De esta manera, se garantiza la aleatoriedad en las submuestras obteniendo una gran variedad de árboles.

Del mismo modo que con el modelo de árbol de decisión, en la práctica se han establecido varios niveles de profundidad de los árboles (1 a 10), a partir del parámetro *max_depth*. Asimismo, el resto de parámetros que se aplican coinciden con los parámetros del modelo de árbol de decisión que se mencionan previamente. Sin embargo, a diferencia del algoritmo anterior, se establecen otros dos parámetros que indican la cantidad de árboles que se generan (en la práctica se han establecido 100 árboles) y, por defecto, el número de características que se consideran cuando se busca la mejor división (raíz cuadrada del número total de características).

- *n_estimators*: número de árboles del "bosque".

- *max_features*: cantidad de características cuando se busca la mejor subdivisión del árbol.

Para el caso de estudio que se plantea, ambos algoritmos han obtenido peores resultados que otros clasificadores. Sin embargo, el algoritmo *Random Forest* sí ha obtenido mejores resultados que la aplicación de un único *árbol de decisión*, tal y como se esperaba [19].

3.4. Clasificador bayesiano: *MultinomialNB()*

Otro de los algoritmos más utilizados en el ámbito de la clasificación de textos es el clasificador bayesiano. Este clasificador es apropiado para problemas de clasificación con características discretas donde se utiliza el recuento de palabras para su aplicación e incluso en problemas donde se aplica *tf-idf* con recuento fraccionario [20].

El *clasificador bayesiano* es una técnica estadística basada en el *Teorema de Bayes* [21]. Asimismo, una de las propiedades más importantes que tiene este algoritmo a la hora de trabajar es el conocimiento previo de las características o datos de entrada y su relación con las categorías finales de la clasificación. De este modo, el algoritmo se centra en la aplicación de estas dos

propiedades: *Teorema de Bayes* y relación de las características de entrada y las categorías de salida.

El algoritmo utiliza el cálculo de probabilidades a partir del *Teorema de Bayes* para conseguir la predicción de una categoría. La manera de obtener estas predicciones viene dada por el cálculo de probabilidad de dicha categoría, es decir, por la probabilidad que se obtiene cuando determinado objeto se clasifica en esa categoría (en el caso de un problema de clasificación como el que se plantea).

De manera formal, se trata de expresar la probabilidad condicional de la clase bajo ciertas condiciones que dependen de una característica o un conjunto de características de entrada. Para ello, se basa en el cálculo de la siguiente probabilidad:

$$P(h|D) = \frac{P(D|h)P(h)}{P(D)} \tag{3.7}$$

- *P(h)*: es la probabilidad de que la hipótesis h sea verdadera (independientemente de los datos o las características de entrada). Esto se conoce como la probabilidad *a priori*. De manera general, el cálculo de este tipo de probabilidades es sencillo y se basa en la siguiente ecuación:

$$P(A) = \frac{\text{nº de veces que ocurre A}}{\text{nº de casos posibles}} \tag{3.8}$$

  Por ejemplo, en este caso, sea h = *'el contrato pertenece a la clase 'INVERSIÓN NUEVA/...'*. Si se tienen 181 textos que pertenecen a la categoría *'INV.NUEVA/...'* y en total 720 textos (sin el outlier), esta probabilidad sería $P(h) = \frac{181}{720}$.

- *P(D)*: es la probabilidad de que los datos o características del problema ocurran (independientemente de la hipótesis) donde D representa una característica (o conjunto de ellas) cualesquiera del que se conocen las probabilidades condicionales y no hace falta calcularla para la hipótesis de máxima probabilidad (es el mismo valor para cualquiera).

- *P(D|h)*: es una probabilidad condicional. Representa la probabilidad de los textos dado que una condición h sea verdadera y la expresión general para su cálculo es la siguiente:

$$P(B|A) = \frac{P(A \cap B)}{P(A)} = \frac{\frac{\text{nº de veces que ocurren A y B a la vez}}{\text{nº de casos posibles}}}{\frac{\text{nº de veces que ocurre A}}{\text{nº de casos posibles}}} \tag{3.9}$$

  para todos los sucesos A, B siempre y cuando $P(A) > 0$ y utilizando 3.8.

  Por ejemplo, en este caso, sea D = *'la característica +colegio'* y h = *'el contrato pertenece a la clase 'INVERSIÓN NUEVA/...'*. En base a 3.10 si, por ejemplo:

  - *nº de veces que aparece '+colegio' en 181 textos que cumplen la condicion* $h = 20$.
  - *nº de palabras en esos 181 textos* $= 331$.
  - *nº veces que aparece '+colegio' en el corpus* $= 150$.
  - *nº de palabras totales* $= 1318$.

  se tendría

$$P(D|h) = \frac{\frac{20}{331}}{\frac{150}{1318}} \tag{3.10}$$

- *P(h|D)*: es la probabilidad de hipótesis h dado un conjunto de datos D. Esto se conoce como probabilidad *a posteriori*.

  Del mismo modo que en 3.10, se calcularía esta última probabilidad.

La decisión final reside en ver qué categoría proporciona la probabilidad más alta, puesto que el dato de entrada pertenecerá a la clase con la probabilidad mayor.

Sin embargo, existe algún inconveniente en el momento de aplicar el *clasificador bayesiano*. En algunas ocasiones, la probabilidad *a posteriori* resultante es cero debido a que los datos o las características no se encuentran en el conjunto de datos de entrenamiento. Para resolver este tipo de problemas, se aplica lo que se conoce como *suavizado de Laplace* que consiste en agregar un elemento de conteo más para que la probabilidad resultante nunca sea cero y al menos el dato aparezca una vez.

Para el análisis práctico, el algoritmo que se utiliza es *MultinomialNB()* de *sklearn*. Existen diferentes clasificadores bayesianos según el problema de clasificación. En este caso, el problema que se plantea es una categorización en múltiples clases y es la razón por la que se utiliza un *clasificador bayesiano multinomial*. El parámetro *alpha* que se utiliza en este algoritmo sirve para establecer el *suavizado de Laplace* anterior.

En el caso práctico, se han establecido diferentes valores para el parámetro *alpha* ([0.05, 0.0001, 0.001, 0.1, 1]) y se ha utilizado el método de *validación cruzada* para obtener el valor óptimo en cada problema de clasificación concreto. En la Tabla 3.1 se muestran los valores óptimos de *alpha* obtenidos:

Naturaleza del gasto	**Objeto de administración (2.1)**	**Objeto de administración (2.1 y 2.2)**
0.0001	0.1	0.1 y 0.0001 *

Tabla 3.1: *: Valores obtenidos acorde a los diferentes parámetros establecidos en el cálculo del *tf-idf*: aplicación de stemming o no, de *n-grams* o no, cálculo de *idf* o no...

Finalmente, en el capítulo siguiente se muestran los resultados obtenidos para el caso de estudio que se plantea en este proyecto.

4

Caso de estudio: clasificación de contratos del sector público

En este capítulo se van a mostrar los diferentes resultados obtenidos en un problema de clasificación. Como se ha mencionado en capítulos anteriores, el estudio ha consistido en clasificar un conjunto pequeño de textos descriptivos de solicitudes de contratos del sector público del Gobierno de Cantabria, en diferentes categorías según ciertas dimensiones y niveles predefinidos.

Para llegar a un resultado final, se han seguido diversos procedimientos como la limpieza y preprocesado de los textos de partida, la extracción de las palabras con mayor y menor relevancia a partir de la relación con las categorías finales y, por último, la conversión de estos textos en *tokens* de palabras (características) y su consecuente vectorización en una matriz numérica procesable por los clasificadores (*tf-idf*). De estos procedimientos se han obtenido resultados interesantes y que en secciones posteriores se muestran.

Por otro lado, se han establecido principalmente tres métricas para evaluar y entender los resultados finales de la clasificación: la *matriz de confusión*, el *accuracy* y los valores *precision* y *recall*. A continuación, se definen estas métricas con más detalle:

- ***matriz de confusión***: resumen o tabla que describe los resultados de predicción obtenidos en un problema de clasificación. También describe el rendimiento del modelo y muestra el número de predicciones correctas e incorrectas, es decir, el número de datos bien y mal clasificados [22]. Esta métrica se basa en cuatro valores que se describen a continuación para un ejemplo de clasificación binaria (se extiende a más categorías):

  - Positivo (P): el dato observado es positivo, es decir, el dato observado es "algo" o pertenece a "algo".
  - Negativo (N): el dato observado es negativo, es decir, el dato observado NO es "algo" o NO pertenece a "algo".

    A partir de estas dos definiciones se describen los siguientes términos de la matriz de confusión.
  - Verdadero positivo (VP): el dato observado es positivo y se predice positivo.
  - Falso negativo (FN): el dato observado es positivo y se predice negativo.
  - Verdadero negativo (VN): el dato observado es negativo y se predice negativo.

- Falso positivo (FP): el dato observado es negativo y se predice positivo.

La estructura de la matriz es la siguiente:

	Clase 1 (predicción)	Clase 2 (predicción)
Clase 1 (observación)	VP	FN
Clase 2 (observación)	FP	VN

- ***accuracy***: indica la tasa de clasificación o de precisión en los resultados y también depende de los cuatro términos definidos anteriormente: VP, FN, FP y VN [22]. La fórmula que utiliza el algoritmo internamente para su cálculo es la siguiente:

$$accuracy = \frac{VP + VN}{VP + FN + FP + VN} \tag{4.1}$$

- ***precision***: indica el valor de dividir el número total de datos positivos correctamente clasificados entre el número de datos positivos predichos [22]. Un valor alto de precision indica que el dato que se ha clasificado como positivo es positivo. La fórmula, por tanto, es la siguiente:

$$precision = \frac{VP}{VP + FP} \tag{4.2}$$

- ***recall***: indica el valor de dividir el número total de datos positivos correctamente clasificados entre el número total de los datos positivos, bien y mal clasificados [22]. Por tanto, un recall alto muestra que la clase es correctamente reconocida (esto es porque el valor de los datos positivos mal clasificados (FN) es pequeño). La fórmula es la siguiente:

$$recall = \frac{VP}{VP + FN} \tag{4.3}$$

Estas dos últimas métricas se pueden analizar de manera conjunta a la hora de explicar los resultados finales.

- **recall alto y precision baja**: los datos positivos se reconocen correctamente pero, sin embargo, existen muchos falsos positivos, es decir, datos que no son positivos se predicen como positivos. Por ejemplo, se tienen manzanas y peras; las manzanas son correctamente clasificadas como manzanas. Sin embargo, hay peras que también son clasificadas como manzanas y no lo son.

- **recall bajo y alta precision**: los datos que se predicen como positivos son realmente positivos pero existen otros muchos datos positivos que no son bien clasificados. En el ejemplo de las manzanas y las peras; se predicen unas pocas manzanas como manzanas correctamente. Sin embargo, hay muchas otras manzanas que se han clasificado como peras y no lo son.

Como conclusión, lo que se busca en un problema de clasificación es obtener un alto recall y una alta precision (es decir, que los falsos negativos y positivos sean pocos).

4.1. Algunos resultados del preprocesado de los datos

En esta sección se muestra una tabla comparativa entre algunos de los datos de entrada sin aplicar ningún tipo de preprocesado y los datos finales una vez se aplican todos los procedimientos de preprocesado mencionados en la Sección 2.2.

Solicitudes de los contratos	
Sin preprocesado	Con preprocesado
RENOVACION **DE LA** SEÑALIZACION VERTICAL **EN LA CA-181**, **DE PESUES A PUENTENANSA DE LA** RED AUTONÓMICA DE CARRETERAS**.** AREA DE CONSERVACION **DE CABEZON DE LA SAL**	renovacion senalizacion vertical **+carretera +localidad +localidad** red autonomica carreteras area conservacion **+localidad**
MODIFICACION **DE** TERMINALES **DE** BARRERA **DE** SEGURIDAD **EN** DIVERSAS CARRETERAS **DE LA** ZONA OCCIDENTAL **DE** CONSERVACION, ADSCRITAS **A LA** RED **DE** CARRETERAS **DEL** GOBIERNO **DE** CANTABRIA**.** AREA CONSERVACION, **CABEZÓN DE LA SAL**	modificacion terminales barrera seguridad diversas carreteras zona occidental conservacion adscritas red carreteras gobierno cantabria area conservacion **+localidad**
CONTRATO MAYOR **DE** OBRA **PARA LA** REGENERACIÓN MEDIOAMBIENTAL **EN EL** ENTORNO **DEL** CEMENTERIO **DE CABEZÓN DE LA SAL (T.M. DE CABEZÓN DE LA SAL)**	contrato mayor obra regeneracion medioambiental entorno cementerio **+localidad +localidad**

Tabla 4.1: Ejemplo de los datos sin y con preprocesado.

En la Tabla 4.1 se pueden observar en negrita los *stopwords* que se han eliminado y las expresiones o palabras que se han mapeado por **+carretera** y **+localidad**. De este modo, el resultado final es un texto con menor número de palabras y con menos vocabulario irrelevante. El resultado que muestra la Tabla 4.1 es un preprocesado común a los tres tipos de problemas de clasificación que se abordan en este proyecto. La diferencia de los textos de entrada para cada uno de las tareas de clasificación reside en la eliminación del vocabulario menos relevante, ya que éste depende de las clases de salida, diferentes en cada problema que se plantea.

Por otro lado, al aplicar el preprocesado de datos y la eliminación de los outliers (veáse la Sección 2.3.3) en cada problema de clasificación, el número de textos de entrada y la cantidad de palabras difiere en cada caso.

	nº textos	**nº palabras sin preprocesar**	**nº palabras preprocesado**
Naturaleza del gasto	720	1934	1318
Objeto de administración: 2.1	719	1930	1316
Objeto de administración: 2.1 y 2.2	696	1890	1283

Tabla 4.2: Número de textos, palabras sin preprocesar los textos y palabras aplicando el preprocesado de textos.

4.1.1. Algunos resultados de la aplicación de *tf-idf*

Para la aplicación de *tf-idf* se han tenido en cuenta dos aspectos. En primer lugar, *tf-idf* se utiliza para establecer el vocabulario con mayor y menor relevancia a través de una puntuación

(*score*) y se aplica en el momento del preprocesado de los textos (veáse la Sección 2.2.3). En segundo lugar, una vez preprocesados los textos de entrada, *tf-idf* se aplica para obtener el peso de cada palabra implicada en los textos. Para ello, se aplican diferentes técnicas de vectorización de las palabras y del cálculo de *tf* e *idf* (*stemming, n-grams*, etc...(Sección 2.3.1)) para, posteriormente, entrenar el modelo de clasificación.

- ***n-grams & stemming***

En los resultados finales influye el tratamiento que se aplica a las palabras o características que componen los textos antes de emplear el clasificador. Para el conjunto de datos preprocesado, el cálculo de *tf-idf* muestra una matriz de pesos que dependen de la estructura de cada palabra y de las fórmulas que se utilicen para el cálculo del *tf* e *idf*.

En el primer caso, la estructura de la palabra, se aplica *n-grams* y/o *stemming*. En el segundo caso, las fórmulas de cálculo, se asigna el estado en los parámetros del método (*sublinear_tf = True*, *smooth_idf=False* y *use_idf=False o True*). En la Sección 2.3.1 se explicaba con mayor detalle el significado de cada parámetro.

A continuación, se expone un ejemplo cualquiera de los diferentes resultados de la estructura de las palabras y del valor de los pesos cuando se aplican los parámetros anteriores para la dimensión *Naturaleza del gasto*:

***n-grams* (1,2)**	**Vocabulario**
4136 palabras o expresiones	['renovacion instalacion', 'renovacion ramal', 'renovacion red', 'renovacion senalizacion', 'renovacion tramo', 'renovacion tuberia', 'sistema incendios', 'sistema intrusos', 'sistema puerta', 'sistema purificacion', 'reordenacion', 'reordenacion espacios', 'reparacion', 'reparacion +colegio', 'reparacion +instituto', 'reparacion accesos', retirada tocones', 'retortillo', 'retortillo +localidad', 'reubicacion', 'reubicacion centralita', 'reubicacion control', 'revestida',...
stemming	**Vocabulario**
1022 palabras o expresiones	['iglesi', 'iluminacion', 'impermeabilizacion', 'impulsion', 'incendi', 'inclu', 'incorporacion', 'industri', 'industrial', 'inert', 'infanci', 'inferior', 'inform', 'informat', 'infraestructur', 'inmobiliari', 'innovacion', 'instal', 'integral', 'intercambi', 'intercept', 'interior', 'intrus', 'inusual', 'inversion', 'invert', 'inyeccion', 'isla', 'iva', 'izquierd', 'judicial', 'junt', 'juvenil', 'juzg', 'lech', 'licitacion', 'limpiez, 'ahed', 'alumn', 'ambos', 'amor', 'anclaj',...
n-grams/ stemming	**Vocabulario**
3811 palabras o expresiones	['centr', 'ubiarc', 'ubiarc +ayunt', 'ubic', 'ubic +local', 'ubic isla', 'uci', 'uci +local', 'udi', 'uned', 'uned +local', 'unid', 'unid socializacion', 'union', 'union escal', 'unqu', 'unqu +local', 'unqu fas', 'unqu ramal', 'urban', 'urban tram', 'urbanizacion', 'urbanizacion exterior', 'urbanizacion gloriet', 'urbanizacion reordenacion', 'urogall', 'uso', 'uso almac', 'uso domest', 'util', 'util public', 'uznay', 'uznay acces', 'valdear', 'valdepr', 'alumn', 'ambos', 'amor', 'anclaj',...

Tabla 4.3: Vocabulario cuando se aplica *tf-idf* con diferentes parámetros.

(nº texto, característica)	peso	(nº texto, característica)	peso
(0, 2096)	0.19892504533076744	(0, 2624)	0.19892504533076744
(0, 2615)	0.19892504533076744	(0, 2898)	0.19892504533076744
(0, 2894)	0.19892504533076744	(0, 1538)	0.19892504533076744
(0, 1537)	0.19892504533076744	(0, 5)	0.19892504533076744
(0, 1)	0.19892504533076744	(0, 162)	0.19892504533076744
(0, 159)	0.19892504533076744	(0, 55)	0.19892504533076744
(0, 51)	0.4174665446550094	(0, 123)	0.19892504533076744
(0, 120)	0.3368093796445482	(0, 139)	0.19892504533076744
(0, 3523)	0.19892504533076744	(0, 3524)	0.19892504533076744
(0, 2115)	0.19892504533076744	(0, 54)	0.19892504533076744

Tabla 4.4: Pesos obtenidos de cada *token* en el primer texto aplicando *stemming*, *n-grams:(1,2)*, con normalización logarítmica para el cálculo de *tf* (*sublinear_tf = True*), sin suavidad en los pesos y sin calcular *idf* (*use_tf=False)*.

En la Tabla 4.4 se observan dos pesos que destacan del resto. Ambos pesos se corresponden con los unigramas (habiendo aplicado *stemming*) *+local* (0.41) y *+numero_km* (0.33). En cambio, el resto de unigramas y/o bigramas de ese texto tienen el mismo peso (0.19).

Para diferentes parámetros, los pesos de la Tabla 4.4 varían debido a que el conjunto de palabras y expresiones difiere según se aplique o no *stemming* y *n-grams*. Del mismo modo, los pesos cambian según la fórmula que se utilice para calcular *tf* e *idf*, es decir, según si el estado de los parámetros anteriores se encuentran en *True* o *False*.

4.2. Análisis de los resultados: clasificación de contratos por *Naturaleza del gasto*

En esta sección se presentan los resultados del problema de clasificación en el que intervienen las categorías de la dimensión *Naturaleza del gasto*. Para presentar los resultados, se han utilizado las métricas comentadas al principio de este capítulo.

En primer lugar, se va a mostrar una tabla en la que se comparan los resultados que se obtienen cuando no se procesan los textos y cuando se procesan, utilizando los parámetros por defecto del método *TfidfVectorizer()*.

- Parámetros por defecto de *TfidfVectorizer()* :
  - ngram_range = (1,1) (solamente palabras únicas).
  - use_idf = True (habilita la ponderación inversa de la frecuencia de documentos: idf).
  - smooth_idf = True (suaviza: añade uno a las frecuencias del documento para los pesos idf para que al menos el término aparezca una vez en cada documento).
  - sublinear_tf = False (aplica el 'recuento bruto' (Tabla 2.7) en lugar de la normalización logarítmica).

- *Accuracy*:

	Textos sin procesar	**Textos procesados**
LogisticRegression()	0.882	0.882
SGDClassifier()	0.879	0.884
MultinomialNB(alpha=0.1)	0.858	0.882
DecisionTreeClassifier() *(depth=5)*	0,833	0.838
RandomForestClassifier() *(depth=5)*	0,851	0.844

Tabla 4.5: *Accuracy* que se obtiene de aplicar los cinco algoritmos propuestos para los textos sin preprocesar y los textos con preprocesado.

- *Matriz de confusión*:

	LogisticRegression()	INV.NUEVA/MEJORA (predicción)	MANTENIMIENTO (predicción)	Total
Textos sin procesar	INV.NUEVA/MEJORA (obs)	138	43	181
	MANTENIMIENTO (obs)	42	497	539
	LogisticRegression()	INV.NUEVA/MEJORA (predicción)	MANTENIMIENTO (predicción)	Total
Textos procesa-dos	INV.NUEVA/MEJORA (obs)	140	41	181
	MANTENIMIENTO (obs)	44	495	539

Tabla 4.6: *Matriz de confusión* que se obtiene de aplicar *LogisticRegression()* para los textos sin preprocesar y preprocesados.

	SGDClassifier()	INV.NUEVA/MEJORA (predicción)	MANTENIMIENTO (predicción)	Total
Textos sin procesar	INV.NUEVA/MEJORA (observación)	136	45	181
	MANTENIMIENTO (observación)	42	497	539
	SGDClassifier()	INV.NUEVA/MEJORA (predicción)	MANTENIMIENTO (predicción)	Total
Textos preproce-sados	INV.NUEVA/MEJORA (observación)	131	50	181
	MANTENIMIENTO (observación)	33	506	539

Tabla 4.7: *Matriz de confusión* que se obtiene de aplicar *SGDClassifier()* para los textos sin preprocesar y preprocesados.

En general, se puede observar en la Tabla 4.5 que la aplicación de procedimientos para preprocesar los datos mejoran un poco los resultados. Sin embargo, esta mejora no es muy significante salvo en el *clasificador Naive Bayes* cuya mejoría está en 0.3 décimas. Cabe destacar que la precisión de los resultados ya es bastante buena teniendo en cuenta que, sin preprocesar, el *accuracy* es de **0.88** en el mayor de los casos.

En la Tabla 4.6, se puede apreciar que la única diferencia reside en el número de textos bien clasificados para ambos casos. Mientras que sin preprocesar se clasifican bien 497 textos en la clase mayoritaria y 138 en la clase minoritaria; con preprocesado se clasifican correctamente 495 y 140 textos en las clases mayoritaria y minoritaria, respectivamente. La diferencia reside en dos textos. Por tanto, cabe esperar que el *accuracy* sea el mismo (veáse fórmula 4.1).

En el caso del algoritmo *SGDClassifier()*, la mejoría del *accuracy* tiene que ver con el aumento de los textos correctamente clasificados para la clase mayoritaria, en relación con el número de textos correctamente clasificados para la clase minoritaria, que disminuye en menor proporción.

Por el contrario, como se observa en la Tabla 4.8, la clasificación que se obtiene al aplicar el clasificador *Naive Bayes* es relativamente mejor cuando los textos son preprocesados. El número de textos que son clasificados correctamente aumenta para ambas categorías.

	MultinomialNB()	INV.NUEVA/MEJORA (predicción)	MANTENIMIENTO (predicción)	Total
Textos sin procesar	INV.NUEVA/MEJORA (observación)	113	68	181
	MANTENIMIENTO (observación)	34	505	539
	MultinomialNB()	INV.NUEVA/MEJORA (predicción)	MANTENIMIENTO (predicción)	Total
Textos preprocesados	INV.NUEVA/MEJORA (observación)	126	55	181
	MANTENIMIENTO (observación)	30	509	539

Tabla 4.8: *Matriz de confusión* que se obtiene de aplicar *MultinomialNB()* para los textos sin preprocesar y preprocesados.

	DecisionTreeClass...	INV.NUEVA/MEJORA (predicción)	MANTENIMIENTO (predicción)	Total
Textos sin procesar	INV.NUEVA/MEJORA (observación)	83	98	181
	MANTENIMIENTO (observación)	34	517	539
	DecisionTreeClass...	INV.NUEVA/MEJORA (predicción)	MANTENIMIENTO (predicción)	Total
Textos preprocesados	INV.NUEVA/MEJORA (observación)	105	76	181
	MANTENIMIENTO (observación)	41	498	539

Tabla 4.9: *Matriz de confusión* que se obtiene de aplicar *DecisionTreeClassifier()* para los textos sin preprocesar y preprocesados.

	RandomForest...	INV.NUEVA/MEJORA (predicción)	MANTENIMIENTO (predicción)	Total
Textos sin procesar	INV.NUEVA/MEJORA (observación)	124	57	181
	MANTENIMIENTO (observación)	50	517	489
	RandomForest...	INV.NUEVA/MEJORA (predicción)	MANTENIMIENTO (predicción)	Total
Textos preprocesados	INV.NUEVA/MEJORA (observación)	126	55	181
	MANTENIMIENTO (observación)	57	482	539

Tabla 4.10: *Matriz de confusión* que se obtiene de aplicar *RandomForestClassifier()* para los textos sin preprocesar y preprocesados.

Los clasificadores de árboles no ofrecen unos resultados sorprendentemente buenos en comparación con el resto de algoritmos. Sin embargo, se puede observar mejores resultados de clasificación y de *accuracy* cuando se aplica un mayor número de árboles de decisión (*Random Forest*) que un único árbol (*Decision Tree*), como se menciona en la Sección 3.3.

Por otro lado, en lo que resta, el proyecto se va a centrar en mostrar los resultados que se obtienen para los tres primeros algoritmos ya que son mejores y se acercan al objetivo del trabajo.

- *precisión & recall*:

LogisticRegression()	**Textos sin procesar**		**Textos preprocesados**	
	precision	*recall*	*precision*	*recall*
INV.NUEVA/MEJORA	0.767	0.762	0.767	0.762
MANTENIMIENTO	0.920	0.922	0.920	0.922
SGDClassifier()	**Textos sin procesar**		**Textos preprocesados**	
	precision	*recall*	*precision*	*recall*
INV.NUEVA/MEJORA	0.764	0.751	0.799	0.724
MANTENIMIENTO	0.917	0.922	0.910	0.939
MultinomialNB()	**Textos sin procesar**		**Textos preprocesados**	
	precision	*recall*	*precision*	*recall*
INV.NUEVA/MEJORA	0.769	0.624	0.808	0.696
MANTENIMIENTO	0.881	0.937	0.902	0.944

Tabla 4.11: Valores de *precision* y *recall* que se obtienen de aplicar *LogisticRegression()*, *SGDClassifier()* y *MultinomialNB()* para los textos sin preprocesar y preprocesados.

En la Tabla 4.11, se observa que para el algoritmo *LogisticRegression()* los valores *precision* y *recall* son relativamente altos para ambas categorías. Sin embargo, para la clase mayoritaria estos valores son aun más altos que para la clase minoritaria. Estos valores indican que los textos

de la categoría mayoritaria se han clasificado muy bien y apenas existe error en la clasificación (de 539 textos, aproximadamente se han clasificado bien 500). Por otro lado, los valores de *precision* y *recall* son un poco más bajos para la categoría minoritaria (debido al menor número de textos que contiene), pero la clasificación es relativamente buena también (de 181 textos se clasifican aproximadamente 140 correctamente). En el caso del algoritmo *SGDClassifier()* ocurre algo similar que para el algoritmo *LogisticRegression()*.

Por otro lado, para el clasificador *Naive Bayes* existen diferencias en los valores *precision* y *recall*. En el caso de los textos sin procesar, se obtienen un ***recall alto y precision baja*** (veáse el Capítulo 4) para la clase mayoritaria (aunque ambos valores son altos, *recall*>*precision*). Finalmente, el resultado muestra una clasificación correcta de una gran parte de los textos pertenecientes a la clase mayoritaria. Por el contrario, la predicción de un gran número de textos pertenecientes a la clase minoritaria ha resultado errónea (aumento del valor de falsos negativos (FN)). Este resultado también puede deducirse observando los valores de *precision* y *recall* de la clase minoritaria y con el argumento que se expone en ***recall bajo y precision alta*** (veáse Capítulo 4).

Sin embargo, cuando se procesan los textos, los resultados cambian y mejoran para este clasificador. Los valores *precision* y *recall* se asemejan a los que se obtienen con el clasificador *LogisticRegression()*, con la única diferencia de que los resultados son mejores con el algoritmo de regresión logística debido a que la clasificación resultante es mucho más equilibrada.

Para este último argumento se puede concluir que el clasificador *LogisticRegression()* es el que mejor resultados proporciona en conjunto. Sin embargo, para este clasificador no difieren apenas los resultados al aplicar preprocesado de textos. Por el contrario, los clasificadores *SGDClassifier()* y *MultinomialNB()* proporcionan una ligera mejoría al aplicar los procedimientos de preprocesado.

A continuación, una vez se analizan los resultados globales que se obtienen sin preprocesar los textos y, posteriormente, con preprocesado de los mismos; el objetivo del presente proyecto ha consistido en intentar mejorar los resultados aplicando las técnicas vistas en la Subsección 4.1.1. Para la obtención de estos resultados se ha optado por utilizar la normalización logarítmica en el cálculo del valor *tf* y no aplicar suavizado a los pesos de *idf* ya que, por el contrario, los resultados no mejoraban.

1. *n-grams, stemming, sublinear_tf = True, smooth_idf = False* y no cálculo de *idf* (*use_idf=False*): únicamente *tf*

	acc	clase	*precision*	*recall*
LogisticRegression()	0.869*	INV.NUEVA/ MEJORA	0.728*	0.768*
		MANTENIMIENTO	0.921*	0.904*
SGDClassifier()	0.891*/ 0.902	INV.NUEVA/ MEJORA	0.832*/ 0.845	0.713*/0.751
		MANTENIMIENTO	0.908*/ 0.919	0.952*/0.954
NB(alpha=0.1)	0.894	INV.NUEVA/ MEJORA	0.797	0.779
		MANTENIMIENTO	0.926	0.933

Tabla 4.12: Resultados de *accuracy, precision* y *recall* para los tres clasificadores. Los valores con * son los que resultan de balancear los datos.

LogisticRegression()	INV.NUEVA/MEJORA (predicción)	MANTENIMIENTO (predicción)	Total
INV.NUEVA/MEJORA (observación)	139*	42*	181
MANTENIMIENTO (observación)	52*	487*	539
SGDClassifier()	INV.NUEVA/MEJORA (predicción)	MANTENIMIENTO (predicción)	Total
INV.NUEVA/MEJORA (observación)	129*/136	52*/45	181
MANTENIMIENTO (observación)	26*/25	513*/514	539
MultinomialNB()	INV.NUEVA/MEJORA (predicción)	MANTENIMIENTO (predicción)	Total
INV.NUEVA/MEJORA (observación)	141	40	181
MANTENIMIENTO (observación)	36	503	539

Tabla 4.13: *Matriz de confusión* que se obtiene de aplicar los tres clasificadores. Los valores con * son los que resultan de balancear los datos.

En primer lugar, se observa en la Tabla 4.12 que la aplicación de estos parámetros ha hecho que la precisión disminuya con el algoritmo *LogisticRegression()*: **0.869**. Por el contrario, *SGDClassifier()* y *MultinomialNB()* ofrecen una mejoría considerable en los valores de *accuracy*: en torno al **0.89**. Cabe destacar que, para el clasificador *SGDClassifier()*, los resultados obtenidos son aun mejores cuando los datos no son balanceados (**0.90**) y los valores de la matriz de confusión muestran una mejor clasificación en conjunto (aumentan los textos bien clasificados para la clase minoritaria, Tabla 4.13).

En segundo lugar, en la Tabla 4.13 se muestra una mejoría en la matriz de confusión debido a que se clasifican correctamente un mayor número de textos en ambas categorías y los algoritmos *SGDClassifier()* y *MultinomialNB()*.

2. *n-grams, sublinear_tf = True, smooth_idf = False* y no cálculo de *idf* (*use_idf=False*): únicamente *tf*

	acc	clase	*precision*	*recall*
LogisticRegression()	0.875*	INV.NUEVA/ MEJORA	0.743*	0.768*
		MANTENIMIENTO	0.921*	0.952*
SGDClassifier()	0.893*/ 0.908	INV.NUEVA/ MEJORA	0.833*/ 0.866	0.718*/0.751
		MANTENIMIENTO	0.910*/ 0.920	0.952*/0.961
NB(alpha=0.1)	0.897	INV.NUEVA/ MEJORA	0.796	0.796
		MANTENIMIENTO	0.931	0.931

Tabla 4.14: Resultados de *accuracy, precision* y *recall* para los tres clasificadores. Los valores con * son los que resultan de balancear los datos.

LogisticRegression()	INV.NUEVA/MEJORA (predicción)	MANTENIMIENTO (predicción)	Total
INV.NUEVA/MEJORA (observación)	139*	42*	181
MANTENIMIENTO (observación)	48*	491*	539
SGDClassifier()	INV.NUEVA/MEJORA (predicción)	MANTENIMIENTO (predicción)	Total
INV.NUEVA/MEJORA (observación)	130*/136	51*/45	181
MANTENIMIENTO (observación)	26*/21	513*/518	539
MultinomialNB()	INV.NUEVA/MEJORA (predicción)	MANTENIMIENTO (predicción)	Total
INV.NUEVA/MEJORA (observación)	144	37	181
MANTENIMIENTO (observación)	37	502	539

Tabla 4.15: *Matriz de confusión* que se obtiene de aplicar los tres clasificadores. Los valores con * son los que resultan de balancear los datos.

En esta ocasión, la diferencia con el apartado anterior reside en que no se aplica *stemming*. La elección de estos parámetros implica una mejora en los resultados, aumentando el *accuracy* para los tres clasificadores y los textos bien clasificados para ambas categorías.

3. *n-grams, sublinear_tf = True, smooth_idf = False* y cálculo de *idf* (*use_idf=True*)

En las Tablas 4.16 y 4.17, los resultados que se obtienen son favorables para el algoritmo *LogisticRegression()*: aumenta el *accuracy* (0.894) respecto del punto 2. (0.875) cuando se calcula *idf*. Sin embargo, no ocurre lo mismo para los otros dos clasificadores. Ambos obtienen un valor del *accuracy* más bajo cuando se calcula el *idf* (SGD:0.88/0.89 y NB:0.87) con respecto a los parámetros del punto 2. (SGD:0.89/0.90 y NB:0.89).

Por otro lado, los valores *precision* y *recall* aumentan en el caso del clasificador de regresión logística obteniéndose una mejor clasificación de los textos para las categorías conjuntamente.

Por el contrario, para los clasificadores *SGDClassifier()* y *MultinomialNB()* se obtiene una peor clasificación de los textos pertenecientes a la clase minoritaria y mayoritaria, respectivamente.

	acc	clase	*precision*	*recall*
LogisticRegression()	0.894*	INV.NUEVA/ MEJORA	0.818*	0.746*
		MANTENIMIENTO	0.917*	0.944*
SGDClassifier()	0.888*/ 0.891	INV.NUEVA/ MEJORA	0.853*/0.876	0.674*/0.663
		MANTENIMIENTO	0.898*/0.895	0.961*/0.968
NB(alpha=0.0001)	0.877	INV.NUEVA/ MEJORA	0.738	0.796
		MANTENIMIENTO	0.930	0.905

Tabla 4.16: Resultados de *accuracy, precision* y *recall* para los tres clasificadores. Los valores con * son los que resultan de balancear los datos.

LogisticRegression()	INV.NUEVA/MEJORA (predicción)	MANTENIMIENTO (predicción)	Total
INV.NUEVA/MEJORA (observación)	135*	46*	181
MANTENIMIENTO (observación)	30*	509*	539
SGDClassifier()	INV.NUEVA/MEJORA (predicción)	MANTENIMIENTO (predicción)	Total
INV.NUEVA/MEJORA (observación)	122*/131	59*/50	181
MANTENIMIENTO (observación)	21*/26	518*/513	539
MultinomialNB()	INV.NUEVA/MEJORA (predicción)	MANTENIMIENTO (predicción)	Total
INV.NUEVA/MEJORA (observación)	144	37	181
MANTENIMIENTO (observación)	51	488	539

Tabla 4.17: *Matriz de confusión* que se obtiene de aplicar los tres clasificadores. Los valores con * son los que resultan de balancear los datos.

En consecuencia, aplicar los diferentes parámetros en el cálculo del *tf-idf* benefician, según qué parámetros se utilicen, a uno u otro clasificador en los resultados finales.

Sin embargo, sí se puede concluir que la aplicación del proceso de *stemming* no mejora en absoluto los resultados en este problema de clasificación.

Por otro lado, el balanceo de datos para categorías desequilibradas no siempre aporta resultados mejores, como es el caso del clasificador *SGDClassifier()*, que obtiene una mejor clasificación de los textos cuando los datos están desbalanceados.

4.3. Análisis de los resultados: clasificación de contratos por *Objeto de administración: nivel 2.1*

En esta sección se presentan los resultados del problema de clasificación en el que intervienen las categorías de la dimensión *Objeto de administración* para el *nivel 2.1*.

En primer lugar, se va a mostrar una tabla en la que se comparan los resultados que se obtienen cuando no se procesan los textos y cuando se procesan, utilizando los parámetros por defecto del método *TfidfVectorizer()* (se detallan en la Sección 4.2).

- *Accuracy*:

	Textos sin procesar	**Textos procesados**
LogisticRegression()	0.915	0.917
SGDClassifier()	0.923	0.930
MultinomialNB(alpha=0.1)	0.899	0.912

Tabla 4.18: *Accuracy* que se obtiene de aplicar los tres algoritmos para los textos sin preprocesar y los textos con preprocesado.

En general, se puede observar en la Tabla 4.18 que la aplicación de procedimientos para preprocesar los datos mejoran los resultados para los tres algoritmos. Además, los valores del *accuracy* son relativamente buenos sin preprocesar los datos ya que en su mayoría supera el 90 % de precisión. Por otro lado, destaca la precisión que ofrece el clasificador *SGDClassifier()* ya que de los tres algoritmos es el que mayor *accuracy* proporciona: 93 % de precisión.

Para exponer la matriz de confusión se utilizan los identificadores de las categorías que se encuentran en la Tabla 2.2.

- *Matriz de confusión*:

Textos sin procesar						
L.Regression()	2002 (pred)	2003 (pred)	2101 (pred)	2102 (pred)	2103 (pred)	Total
2002 (obs)	259	8	1	11	0	279
2003 (obs)	6	185	0	1	2	194
2101 (obs)	1	4	53	2	0	60
2102 (obs)	4	2	2	71	1	80
2103 (obs)	2	11	0	3	89	105
Textos preprocesados						
L.Regression()	2002 (pred)	2003 (pred)	2101 (pred)	2102 (pred)	2103 (pred)	Total
2002 (obs)	259	9	0	11	0	279
2003 (obs)	0	192	0	1	1	194
2101 (obs)	1	3	54	2	0	60
2102 (obs)	3	4	2	70	1	80
2103 (obs)	0	15	0	6	84	105

Tabla 4.19: *Matriz de confusión* que se obtiene de aplicar *LogisticRegression()* para los textos sin preprocesar y con preprocesado.

Textos sin procesar						
SGD...()	2002 (pred)	2003 (pred)	2101 (pred)	2102 (pred)	2103 (pred)	Total
2002 (obs)	264	6	1	8	0	279
2003 (obs)	4	189	0	0	1	194
2101 (obs)	2	3	52	2	1	60
2102 (obs)	11	2	1	64	2	80
2103 (obs)	1	9	0	1	94	105
Textos preprocesados						
SGD...()	2002 (pred)	2003 (pred)	2101 (pred)	2102 (pred)	2103 (pred)	Total
2002 (obs)	265	5	1	8	0	279
2003 (obs)	2	189	0	2	1	194
2101 (obs)	1	1	56	2	0	60
2102 (obs)	9	2	2	66	1	80
2103 (obs)	2	9	0	2	92	105

Tabla 4.20: *Matriz de confusión* que se obtiene de aplicar *LogisticRegression()* para los textos sin preprocesar y con preprocesado.

En la Tabla 4.19, se puede apreciar una buena clasificación de los textos en gran parte de las categorías. Sin embargo, existen dos pares de categorías, correspondientes a 'OBRAS CIVIL' y 'CENTROS', que contienen textos mal clasificados debido a que la única diferencia entre estas categorías es la administración que lleva a cabo las solicitudes de los contratos. En el caso de los textos sin preprocesar, 11 solicitudes de contratos de obra civil, elaboradas por el Gobierno de Cantabria (*C.A CANTABRIA*), se han clasificado en la clase de las solicitudes para este mismo objeto que han sido elaboradas por ayuntamientos (*OTRAS ADMINISTRACIONES*). Del mismo modo, 11 solicitudes de contratos de centros y edificaciones, realizadas por los ayuntamientos, se han clasificado en la categoría de las solicitudes realizadas por el Gobierno de Cantabria para este mismo objeto.

De manera similar ocurre con los resultados que se obtienen para estas dos clases cuando los textos se preprocesan.

En el caso del algoritmo *SGDClassifier()*, se puede observar en la Tabla 4.20 una mejoría en la clasificación de los textos que pertenecen a las clases mayoritarias con respecto al clasificador *LogisticRegression()*. Por otro lado, la predicción para los textos de las categorías minoritarias empeora, sin embargo, en menor proporción que el aumento de textos correctamente clasificados para las categorías mayoritarias. Esto hace que el *accuracy* para este clasificador sea más alto que el que se obtiene con *LogisticRegression()*.

Textos sin procesar						
NB()	2002 (pred)	2003 (pred)	2101 (pred)	2102 (pred)	2103 (pred)	Total
2002 (obs)	261	5	1	11	1	279
2003 (obs)	2	178	1	2	11	194
2101 (obs)	1	1	53	4	1	60
2102 (obs)	8	1	2	65	4	80
2103 (obs)	0	13	0	3	89	105
Textos preprocesados						
NB()	2002 (pred)	2003 (pred)	2101 (pred)	2102 (pred)	2103 (pred)	Total
2002 (obs)	262	4	1	11	1	279
2003 (obs)	0	179	0	4	11	194
2101 (obs)	1	1	55	3	0	60
2102 (obs)	6	2	3	67	2	80
2103 (obs)	0	11	0	2	92	105

Tabla 4.21: *Matriz de confusión* que se obtiene de aplicar *MultinomialNB()* para los textos sin preprocesar y con preprocesado.

En general, para el clasificador *Naive Bayes* los resultados que se obtienen son muy similares a los del clasificador *LogisticRegression()*. Las categorías correspondientes a los objetos de administración 'OBRA CIVIL' y 'CENTROS', clasifican algunos textos indistintamente en una u otra categoría según el contrato se realice por el Gobierno de Cantabria o por un ayuntamiento.

- *precisión & recall*:

L.Regression()	**Textos sin procesar**		**Textos preprocesados**	
	precision	*recall*	*precision*	*recall*
2002 (obs)	0.952	0.928	0.985	0.928
2003 (obs)	0.881	0.954	0.861	0.990
2101 (obs)	0.946	0.883	0.964	0.900
2102 (obs)	0.807	0.887	0.778	0.875
2103 (obs)	0.967	0.848	0.977	0.800

Tabla 4.22: Valores de *precision* y *recall* que se obtienen de aplicar *LogisticRegression()* para los textos sin preprocesar y preprocesados.

SGDClass...()	**Textos sin procesar**		**Textos preprocesados**	
	precision	*recall*	*precision*	*recall*
2002 (obs)	0.936	0.946	0.950	0.950
2003 (obs)	0.904	0.974	0.917	0.974
2101 (obs)	0.963	0.867	0.949	0.933
2102 (obs)	0.853	0.800	0.825	0.825
2103 (obs)	0.959	0.895	0.979	0.876

Tabla 4.23: Valores de *precision* y *recall* que se obtienen de aplicar *SGDClassifier()* para los textos sin preprocesar y preprocesados.

Multi...NB()	**Textos sin procesar**		**Textos preprocesados**	
	precision	*recall*	*precision*	*recall*
2002 (obs)	0.960	0.935	0.974	0.939
2003 (obs)	0.899	0.918	0.909	0.923
2101 (obs)	0.930	0.883	0.932	0.917
2102 (obs)	0.765	0.812	0.770	0.838
2103 (obs)	0.840	0.848	0.868	0.876

Tabla 4.24: Valores de *precision* y *recall* que se obtienen de aplicar *MultinomialNB()* para los textos sin preprocesar y preprocesados.

Los valores *precision* y *recall* son relativamente altos para los tres clasificadores. Estos valores muestran una buena clasificación de los textos en el que los valores falsos positivos (FP) y falsos negativos (FN) son relativamente bajos para todas las clases.

De la misma manera que se hizo para el problema de clasificación de la dimensión *Naturaleza del gasto*, se ha intentado mejorar los resultados con la aplicación de las técnicas vistas en la Subsección 4.1.1 para el cálculo de *tf-idf*. Al aplicar diversos parámetros en el cálculo de *tf-idf*, los resultados no se han conseguido mejorar respecto de los anteriores (parámetros por defecto vistos al principio de la Sección 4.2).

A continuación, se exponen algunos resultados en los que no se aplica *stemming*, ni se cálcula *idf* ya que para el clasificador *MultinomialNB()* los resultados han sido un poco mejores y en el resto apenas varían.

n-grams, sublinear_tf = True, smooth_idf = False* y no cálculo de *idf* (*use_idf=False*): únicamente *tf

	acc	clase	*precision*	*recall*
LogisticRegression()	0.901	2002	0.988	0.903
		2003	0.814	0.990
		2101	0.917	0.917
		2102	0.793	0.863
		2103	0.988	0.752
SGDClassifier()	0.926	2002	0.967	0.943
		2003	0.895	0.969
		2101	0.917	0.917
		2102	0.840	0.850
		2103	0.958	0.867
MultinomialNB(alpha=0.1)	0.928	2002	0.992	0.925
		2003	0.925	0.954
		2101	0.917	0.917
		2102	0.809	0.900
		2103	0.890	0.924

Tabla 4.25: Resultados de *accuracy, precision* y *recall* para los tres clasificadores.

L.Regression()	2002 (pred)	2003 (pred)	2101 (pred)	2102 (pred)	2103 (pred)	Total
2002 (obs)	252	15	3	9	0	279
2003 (obs)	1	192	0	1	0	194
2101 (obs)	0	4	55	1	0	60
2102 (obs)	2	6	2	69	1	80
2103 (obs)	0	19	0	7	79	105
SGD...()	2002 (pred)	2003 (pred)	2101 (pred)	2102 (pred)	2103 (pred)	Total
2002 (obs)	263	5	3	8	0	279
2003 (obs)	3	188	0	1	2	194
2101 (obs)	1	2	55	2	0	60
2102 (obs)	4	4	2	68	2	80
2103 (obs)	1	11	0	2	91	105
NB()	2002 (pred)	2003 (pred)	2101 (pred)	2102 (pred)	2103 (pred)	Total
2002 (obs)	258	5	2	11	3	279
2003 (obs)	0	185	1	2	6	194
2101 (obs)	0	1	55	2	2	60
2102 (obs)	2	3	2	72	1	80
2103 (obs)	0	6	0	2	97	105

Tabla 4.26: *Matriz de confusión* que se obtiene de aplicar los tres clasificadores.

En consecuencia con los resultados que se han obtenido para este problema de clasificación, se puede concluir que el preprocesado de los textos que se realiza mejora los resultados de

clasificación. Además, los tres algoritmos ofrecen unos resultados muy buenos. Sin embargo, el clasificador *SGDClassifier()* obtiene mayor precisión en los resultados.

Por otro lado, cabe destacar que para las categorías 'OBRA CIVIL' y 'CENTROS' existen algunos valores falsos positivos y negativos que recalcar. Esto se debe a la incorrecta clasificación que hace de alguna solicitud de contrato cuando proviene del Gobierno de Cantabria o de un ayuntamiento.

Sin embargo, la importancia de la categorización del contrato para esta dimensión recae sobre el objeto de la administración (ej: obra civil, edificio, terreno, etc...) y no de quién elabora la solicitud (Gobierno de Cantabria u otras entidades).

Este último argumento ha permitido la realización de este mismo problema de clasificación pero eliminando de las etiquetas de las clases la referencia a la entidad o administración del contrato (*C.A CANTABRIA, OTRAS ADMINISTRACIONES*). De este modo e indistintamente, se han reestablecido y disminuido las categorías, teniendo en cuenta únicamente el objeto de administración y, el número de contratos pertenecientes a cada una. Finalmente, cabe esperar que los resultados mejoren y los falsos positivos y negativos disminuyan.

A continuación, se muestran los resultados que se obtienen para el clasificador que ofrecía una mayor precisión en los resultados anteriores: *SGDClassifier()*.

	acc	clase	2001 (OBRA)	2003 (CENTRO)	2101 (TERRENO)
SGDClassifier	0.967	2001 (OBRA)	347	10	2
		2003 (CENTRO)	6	292	1
		2101 (TERRENO)	4	0	56

Tabla 4.27: Resultados de *accuracy* y *matriz de confusión* para *SGDClassifier()* y los parámetros por defecto para *tf-idf*.

4.4. Análisis de los resultados: clasificación de contratos por *Objeto administración: nivel 2.1 y nivel 2.2*

En esta sección se presentan los resultados del problema de clasificación en el que intervienen las categorías de la dimensión *Objeto de administración* para los *niveles 2.1* y *2.2*.

En primer lugar, se va a mostrar una tabla en la que se comparan los resultados que se obtienen de los tres algoritmos, cuando no se procesan los textos y cuando se procesan. Se utilizan los parámetros por defecto del método *TfidfVectorizer()* (se detallan en la Sección 4.2).

- *Accuracy*:

	Textos sin procesar	**Textos procesados**
LogisticRegression()	0.908	0.913
SGDClassifier()	0.928	0.935
MultinomialNB(alpha=0.1)	0.866	0.890

Tabla 4.28: *Accuracy* que se obtiene de aplicar los tres algoritmos para los textos sin preprocesar y los textos con preprocesado.

Los valores de *accuracy* son muy altos para los tres algoritmos y, además, aumentan con el preprocesado de los datos. Destaca sobre el resto, el algoritmo *SGDClassifier()*, cuya precisión es del 93 %.

A continuación, únicamente se mostrarán el *accuracy* y *matriz de confusión* de los mejores resultados al realizar todo el proceso comentado en las anteriores Secciones 4.2 y 4.3. Es decir, los resultados que se obtienen de aplicar el clasificador *SGDClassifier()*, el preprocesado de los textos de entrada y los valores por defecto del *tf-idf*.

- *Accuracy*: **0.935**
- *Matriz de confusión*:

En la Tabla 2.4 se encuentran los identificadores correspondientes a cada una de las categorías y se utilizan para exponer los resultados de la matriz de confusión.

Textos preprocesado												
SGD()	1 (pred)	2 (pred)	3 (pred)	4 (pred)	5 (pred)	6 (pred)	7 (pred)	8 (pred)	9 (pred)	10 (pred)	11 (pred)	Total
1 (obs)	**189**	2	0	0	0	1	0	0	0	0	0	192
2 (obs)	11	**93**	0	0	0	0	0	0	0	1	0	105
3 (obs)	0	0	**15**	0	0	0	0	0	0	0	1	16
4 (obs)	0	0	0	**12**	0	0	2	0	0	0	0	14
5 (obs)	0	0	1	0	**229**	0	0	2	0	1	1	232
6 (obs)	2	0	0	0	2	**7**	0	0	0	0	0	11
7 (obs)	1	0	0	2	0	0	**17**	0	0	2	0	22
8 (obs)	0	0	0	0	3	0	0	**7**	0	0	1	11
9 (obs)	0	0	0	0	1	0	1	0	**7**	1	0	10
10 (obs)	1	0	0	0	1	0	1	0	2	**18**	0	23
11 (obs)	3	0	0	0	0	0	0	0	0	0	**57**	60

Tabla 4.29: *Matriz de confusión* que se obtiene de aplicar *SGDClassifier()* para los textos con preprocesado.

Como se muestra en la Tabla 4.29, la clasificación de los textos es muy buena en su mayoría. Sin embargo, para las categorías correspondientes al objeto 'CENTROS', se puede apreciar que de 105 contratos que han sido elaborados por ayuntamientos (*OTRAS ADMINISTRACIONES*), 11 de ellos han sido clasificados incorrectamente como contratos realizados por el Gobierno de Cantabria (*C.A CANTABRIA*). De manera similar ocurría en el problema de clasificación de la Sección 4.3.

Por otro lado, al igual que se hizo para los casos de estudio anteriores, se ha intentado mejorar los resultados con la aplicación de las técnicas vistas en la Subsección 4.1.1. Sin embargo, los valores que se obtienen no son mejores. Las diferentes variaciones de parámetros para el cálculo de *tf-idf* han conseguido alcanzar un *accuracy* en torno al 92 % en cualquiera de los tres casos planteados y, una clasificación que empeora aun más en las categorías referidas al objeto de administración 'CENTROS'.

En vista de los resultados, al igual que en el anterior caso de estudio, se puede esperar que la importancia de la categorizacion del contrato sea en torno al objeto de administración. Por esta razón, se ha llevado a cabo el mismo problema de clasificación pero sin tener en cuenta la entidad de la que proviene el contrato (*C.A CANTABRIA, OTRAS ADMINISTRACIONES*). Por tanto, el número de categorías disminuye y aumenta la cantidad de textos en cada categoría restante. De este modo, cabe esperar que los resultados mejoren respecto de los anteriores.

SGD		CENTRO-DEMANDIAL	INF.-PORTUARIA	INF.-ABASTEC.	INF.-VIAL	INF.-TRANS.	INF.-CAMINO	INF.-SANEAM.	TERRENO
	CENTRO-DEMANDIAL	**295**	0	0	0	1	1	0	0
acc	INF.-PORTUARIA	0	**15**	0	0	0	0	0	1
	INF.-ABASTEC.	1	0	**34**	0	0	0	1	0
	INF.- VIAL	1	1	0	**238**	0	1	1	1
	INF.-TRANS.	2	0	0	1	**8**	0	0	0
0.966	INF.-CAMINO	0	0	0	1	0	**8**	1	0
	INF.-SANEAM.	0	0	0	1	0	2	**19**	1
	TERRENO	3	0	1	0	0	0	0	**56**

Tabla 4.30: Resultados de *accuracy* y *matriz de confusión* para *SGDClassifier()* y los parámetros por defecto para *tf-idf*. *INF = INFRAESTRUCTURA*.

5 Conclusiones

Abordar este tipo de tareas de clasificación no es un trabajo sencillo. Los problemas que se plantean y su dependencia con la estructura de los datos disponibles conducen a la realización de un profundo análisis de las técnicas de *Procesamiento de Lenguaje Natural* y de *Minería de Datos* (*Minería de Textos*) y, consecuentemente, a la realización de este proyecto.

Destacar que en los tres casos de estudio se obtenían clasificaciones muy buenas sin necesidad de aplicar ningún método de preprocesado previo y, mejorar estos resultados era una tarea díficil.

La pequeña colección de textos y la escasez de vocabulario en los mismos, han hecho que este trabajo se centrase en escoger cuidadosamente las técnicas de preprocesado, a fin de evitar reducir aun más el conjunto de datos.

A fin de cumplir el objetivo del proyecto, los resultados han mejorado ligeramente con el preprocesado de los datos en los tres problemas de clasificación y utilizando los tres algoritmos principales: *LogisticRegression(), SGDClassifier()* y *MultinomialNB()*. Sin embargo, uno de los clasificadores ha mostrado, en conjunto, mejores resultados que el resto: *SGDClassifier()*.

En vista de las recomendaciones con respecto a balancear los datos, se ha demostrado que los resultados no son siempre lo que se espera. Cada conjunto de datos requiere de un tratamiento especial y un preprocesado de datos particular. En este caso, el balanceo de los datos no ha conseguido mejorar los resultados en el caso del algoritmo *SVM* (siendo mejores sin balancear) y la aplicación del mismo algoritmo ha concluido mejores resultados que el resto de algoritmos.

Por otro lado, el estudio no finaliza en este proyecto. A pesar de los buenos resultados que se obtienen para el conjunto reducido de datos, no se dispone de un conjunto nuevo para validar estos resultados (una de las razones por las que se ha utilizado *validación cruzada: Leave One Out* en el procesado de división, entrenamiento de los datos y validación del modelo). Con la disposición de un nuevo conjunto de datos (*p.e: los contratos de 2016 y 2017*), las conclusiones se describirían con mayor certeza.

Por último, se podrían probar otros algoritmos y métodos para la clasificación de textos (*KNN*, *redes neuronales*,...), o algún preprocesado de textos diferente (por ejemplo, utilizar *word embeddings*) y analizar de nuevo los resultados. Sin embargo, para el objetivo de este trabajo y la colección de textos disponible, se han explicado argumentadamente estas técnicas de preprocesado y estos algoritmos.

Bibliografía

[1] Marti A. Hearst. **Untangling Text Data Mining**. *School of Information Management & Systems University of California, Berkeley, 102 South Hall, Berkeley, CA 94720-4600*, pages 3–7, 1999.

[2] J.E. Hopcroft, R. Motwani, and J.D. Ullman. ***Automata Theory, Languages, and Computation***. Addison-Wesley. ISBN 0-321-45536-3, 3 edition, 2006.

[3] Olga Davydova. **Text Preprocessing in Python: Steps, Tools, and Examples**. Data Monsters https://es.wikipedia.org/wiki/Expresi%C3%B3n_regular.

[4] 2018 Hafsa Jabeen, October 23rd. **Stemming and Lemmatization in Python**. DataCamp https://www.datacamp.com/community/tutorials/stemming-lemmatization-python.

[5] 2018 Raheel Shaikh, Oct 28. **Feature Selection Techniques in Machine Learning with Python**. Towards Data Science https://towardsdatascience.com/feature-selection-techniques-in-machine-learning-with-python-f24e7da3f36e.

[6] 2018 Daniel Rodríguez, Noviembre 9. **Visualización de árboles de decisión en Python con PyDotPlus**. https://www.analyticslane.com/2018/11/09/visualizacion-de-arboles-de-decision-en-python-con-pydotplus/.

[7] **sklearn.ensemble.ExtraTreesClassifier**. 2007 - 2019, scikit-learn developers (BSD License) https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.ExtraTreesClassifier.html.

[8] 2018 Javaid Nabi, Sep 13. **Machine Learning — Text Processing**. Towards Data Science https://towardsdatascience.com/machine-learning-text-processing-1d5a2d638958.

[9] 2018 Susan Li, Feb 19. **Multi-Class Text Classification with Scikit-Learn**. Towards Data Science https://towardsdatascience.com/multi-class-text-classification-with-scikit-learn-12f1e60e0a9f.

[10] Wei Di, Anurag Bhardwaj, and Jianing Wei. ***Deep Learning Essentials: Weighting the terms tf-idf***. Packt Publishing Ltd. ISBN 978-1-78588-036-0, 35 Livery Street, Birmingham, B3 2PB, UK., 2006.

[11] Definición de tfidf. https://en.wikipedia.org/wiki/Tf%E2%80%93idf.

[12] Daniel Jurafsky and James H.Martin. ***Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition***. Draft of September 23, 2018, 3 edition, 2006.

[13] **sklearn.feature_extraction.text.TfidfVectorizer**. 2007 - 2019, scikit-learn developers (BSD License) https://scikit-learn.org/stable/modules/generated/sklearn.feature_extraction.text.TfidfVectorizer.html.

[14] **Clasificación con datos desbalanceados**. Mayo 16, https://www.aprendemachinelearning.com/clasificacion-con-datos-desbalanceados/.

[15] 2017 Adi Bronshtein, May 17. **Train/Test Split and Cross Validation in Python**. Towards Data Science https://towardsdatascience.com/train-test-split-and-cross-validation-in-python-80b61beca4b6.

[16] Ashish Kumar and Joseph Babcock. ***Python: Advanced Predictive Analytics; Chapter 5, "Logistic Regression"***. Packt Publishing Ltd. ISBN 978-1-78899-236-7, 35 Livery Street, Birmingham, B3 2PB, UK., 2 edition, 2017.

[17] **sklearn.linear_model.SGDClassifier**. 2007 - 2019, scikit-learn developers (BSD License) https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.SGDClassifier.html.

[18] **1.10. Decision Trees**. 2007 - 2019, scikit-learn developers (BSD License) https://scikit-learn.org/stable/modules/tree.html.

[19] **sklearn.ensemble.RandomForestClassifier**. 2007 - 2019, scikit-learn developers (BSD License) https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html.

[20] **sklearn.naive_bayes.MultinomialNB**. 2007 - 2019, scikit-learn developers (BSD License) https://scikit-learn.org/stable/modules/generated/sklearn.naive_bayes.MultinomialNB.html.

[21] **Teorema de Bayes**. https://es.wikipedia.org/wiki/Teorema_de_Bayes.

[22] **Confusion Matrix in Machine Learning**. GeeksforGeek: A computer science portal for geeks https://www.geeksforgeeks.org/confusion-matrix-machine-learning/.