Universidad Politécnica
de Madrid

**Escuela Técnica Superior de
Ingenieros Informáticos**

Máster en Ciencia de Datos

Trabajo Fin de Máster

Asignación automática del código CPV a la contratación pública

Autor: Erick Abdiel Cedeño Guerra

Madrid, Febrero, 2024

Este Trabajo Fin de Máster se ha depositado en la ETSI Informáticos de la Universidad Politécnica de Madrid.

Trabajo Fin de Máster
Máster en ***Ciencia de Datos***

Título: **Asignación automática del código CPV a la contratación pública**
Febrero, 2024

Autor: **Erick Abdiel Cedeño Guerra**

Tutor

Óscar Corcho García

ETSI Informáticos
Departamento de Inteligencia Artificial
Universidad Politécnica de Madrid

Co-Tutor

María Navas Loro

ETSI Informáticos
Departamento de Inteligencia Artificial
Universidad Politécnica de Madrid

Resumen

La contratación pública es una importante fuente de negocio. Cada año, más de 250 000 autoridades públicas de toda la Unión Europea (UE) destinan alrededor del 18% del producto interno bruto (PIB) de la UE a la contratación de servicios, obras o suministros [1].
Los contratos públicos se adjudican mediante licitaciones –procedimiento conocido como contratación pública. Estos contratos abarcan una amplia gama de ámbitos, como estudios, asistencia técnica y formación, consultoría, organización de conferencias, equipos informáticos entre otros. Su finalidad es adquirir servicios, bienes u obras para garantizar el buen funcionamiento de las instituciones y programas de la UE [1].
Este proyecto aplica técnicas avanzadas de procesamiento de lenguaje natural, así como de aprendizaje automático para predecir el tercer dígito de los códigos *Common Procurement Vocabulary* (CPV), traducido al español como *Vocabulario Común de Contratos Públicos* (VCCP), utilizados en los contratos de licitación de la Unión Europea. El esquema CPV, con alrededor de 10 000 códigos distintos, provee un método para identificar licitaciones y adjudicaciones en cualquier país de la Unión Europea. Sin embargo, su asignación manual resulta una tarea compleja. Esta investigación se centra en la predicción del tercer dígito de los códigos CPV valiéndose únicamente de la descripción del contrato. El mecanismo de predicción empleado es el modelo Random forest, que destacó por su superior precisión y eficiencia de tiempo sobre los demás modelos de aprendizaje automático evaluados. Este trabajo demuestra la viabilidad de estas técnicas para predecir las categorías de contratos de licitación basándose en sus descripciones, de lo que deriva una potencial mejora en la eficiencia de los procesos de licitación y detección temprana de irregularidades.

Abstract

Public procurement is an important source of business. Each year, more than 250,000 public authorities across the EU spend around 18% of European Union (EU) Gross Domestic Product (GDP) on the procurement of services, works or supplies [1].
Public contracts are awarded through calls for tenders (a procedure known as public procurement). They cover a wide range of areas, such as studies, technical assistance and training, consultancy, organization of conferences, IT equipment and many others. The purpose of the contracts is to procure services, goods or works to ensure the proper functioning of EU institutions or programs [1].
This project applies advanced natural language processing and machine learning techniques to predict the third digit of *Common Procurement Vocabulary* (CPV) codes used in EU tender contracts. The CPV scheme, with around 10,000 different codes, provides a method for identifying tenders and awards in any European Union country. However, its manual assignment is a challenging task. This study focuses on predicting the third digit of CPV codes using only the contract description. Several machine learning models were tested, and after detailed evaluation, the Random forest model was selected for its superior accuracy and time efficiency. This work demonstrates the feasibility of these techniques for predicting bidding contract categories based on their descriptions, potentially improving the efficiency of bidding processes and the early detection of irregularities.

Tabla de contenidos

1 Introducción 1
1.1 Objetivos 2
1.1.1 Objetivos generales 2
1.1.2 Objetivos específicos 2
1.2 Estructura del documento 4
2 Estado del arte 5
2.1 Contrataciones públicas 5
2.2 Sistema de categorización de productos 5
2.2.1 Esquema CPV 5
2.3 Proyectos relacionados con el presente trabajo 7
2.3.1 Las licitaciones públicas: análisis de datos y sistemas predictivos utilizando métodos de aprendizaje automático 7
2.3.2 Clasificador de Códigos de Licitaciones Públicas: Caso de Estudio España y México 7
2.3.3 Multi-label Text Classification for Public Procurement in Spanish 8
2.4 Procesamiento de lenguaje natural 8
2.4.1 Lowercasing 8
2.4.2 Tokenización 8
2.4.3 Eliminación de ruido 9
2.4.4 La eliminación de palabras vacías 9
2.4.5 Lematización 9
2.4.6 Técnicas de vectorización 9
2.4.6.1 Bag of words 9
2.4.6.2 TF-IDF (Term Frequency-Inverse Document Frequency) 10
2.4.7 Algoritmos de clasificación 10
2.4.7.1 Random forest 10
2.4.7.2 Decision tree 10
2.4.7.3 MutinominalNB 11
2.4.7.4 Gradient boosting 11
2.4.7.5 Support vector machine 11
3 Desarrollo 13
3.1 Recolección de los datos 13
3.2 Exploración de los datos 14
3.2.1 Exploración de CPV 15
3.3 Preprocesamiento de los datos 18
3.4 Modelado y Evaluación 20
3.4.1 Modelado y evaluación de códigos CPV 23
3.4.1.1 Random forest 24
3.4.1.2 Decision tree 27
3.4.1.3 MultinominalNB 32
3.4.1.4 Gradient boosting 35
3.4.1.5 Support vector machine 39
3.4.2 Comparación de los modelos 43

4 Resultados y conclusiones **46**
5 Bibliografía **47**
6 Anexos **49**
Prueba con todos los 2 primeros dígitos 49
Código para quedarnos sólo las filas y columnas cuyo código empieza por dos dígitos específico 49
Función para procesar las descripciones 49
Entrenamiento del modelo 50

1 Introducción

La contratación pública permite distintos tipos de contratación. El uso de una estructura codificada fomenta que las licitaciones se encuentren a disposición de las autoridades públicas.
El CPV se compone de un código numérico de 8 dígitos, ampliable con un guión y un dígito adicional. Este sistema permite una clasificación jerárquica de más de 10 000 referencias mediante una codificación numérica que hace referencia a los sujetos contratantes. La finalidad de este sistema es establecer un lenguaje común que permita clasificar de forma normalizada los objetos de los contratos realizados en el marco de la contratación pública [2].
La longitud habitual de los CPV es de 5 dígitos. Hay tipos de contratos y materias en los que es más habitual el uso de CPV muy específicos, bien sea por la regulación o por la especialización de las unidades contratantes (como en el caso de los contratos de obras).
Como se trata de una estructuración jerárquica, la especificidad del código aumenta a medida que el contrato es más detallado (A. Ramírez Sixtos, 2022) [7]. Por ejemplo, un código que contenga 2 o 3 dígitos seguido por varios ceros es más genérico, mientras que uno que agote los 9 dígitos va a ser más específico. Generalmente, se utilizan códigos de 4 o 5 dígitos. Al tratarse de una estructuración jerárquica, la especificidad del código aumenta conforme se detalla el objeto del contrato. En la sección Esquema CPV se explica más a fondo.
El CPV permite que una administración pública indique qué es lo que necesite más allá de sus consideraciones lingüísticas y operativas y que los agentes económicos lo puedan localizar más allá de estas. El uso de los CPV es obligatorio en la definición del objeto del contrato, tal y como indica el artículo 2 de la Ley de Contratos del Sector Público (LCSP) [2]. Un CPV tiene que permitir la identificación del mayor número de operadores económicos interesados en el objeto del contrato, evitando al mismo tiempo la identificación de los que no correspondan a los requisitos del contrato. De ahí la importancia de elegir el CPV con la mayor precisión posible. Por ejemplo, si un ente gubernamental desea licitar por una tonelada de óxido de aluminio, la elección del código CPV puede influir significativamente en la eficacia de este proceso. Si utilizan el código CPV 14700000, que corresponde a la categoría genérica de "metales básicos", la convocatoria de licitación podría llegar tanto a proveedores de óxido de aluminio como a proveedores de otros metales. Esto podría generar cierto ruido en el proceso de licitación, ya que puede atraer ofertas que no son precisamente las deseadas. Sin embargo, si se opta por utilizar un código CPV más específico, como el 14721100, que corresponde exactamente al óxido de aluminio, entonces la convocatoria de licitación llegaría exclusivamente a proveedores de ese material específico. De este modo, se optimiza el proceso de licitación, enfocándose únicamente en los proveedores más relevantes para la necesidad en cuestión.
Es crucial destacar que la utilización incorrecta de un código CPV puede generar complicaciones no solo de carácter operativo, sino potencialmente también legales. Tal equivocación puede obstaculizar la participación de empresas idóneas en una licitación, permitiendo al mismo tiempo que empresas no aptas puedan presentar sus propuestas.
Un estudio realizado por Gobierto [2] (plataforma de gobierno que ofrece herramientas de transparencia, participación y rendición de cuentas para administraciones públicas) acerca del uso de CPV, halló que los contratos con

CPV más específicos presentan una concurrencia más elevada que los genéricos.
Cada proceso de contratación pública debe estar clasificado con al menos un CPV. Debido al plurilingüismo de la Unión Europea (UE), así como de gran parte de los estados que la forman, localizar posibles oportunidades de contratación resulta difícil, sin embargo, este lenguaje numérico supera esta limitación y ayuda a mejorar la organización de los contratos.
Es importante señalar que la codificación CPV presenta un problema como la existencia de más de 9 000 códigos distintos Esta gran cantidad puede complicar la tarea de aquellos responsables de asignar estos códigos a un proceso específico, incrementando el margen de errores humanos. Las oficinas pueden seguir directrices variadas al asignar estos códigos, lo que podría conducir a niveles de granularidad dispares, dependiendo del anotador. Esta falta de consistencia en la asignación de códigos CPV puede dar lugar a inconsistencias y errores.

1.1 Objetivos

1.1.1 Objetivo general

La automatización y mejora del proceso de clasificación de productos y servicios en el ámbito de la contratación pública mediante el uso de código CPV. El clasificador propuesto debe ser capaz de analizar descripciones de productos y servicios escritos en lenguaje natural y asignar automáticamente el código CPV correspondiente, de manera que la carga de trabajo manual de los clasificadores humanos se reduzca. Este enfoque busca predecir el tercer dígito del código CPV, basándose en un trabajo previo llevado a cabo por el grupo de investigación OEG de la Universidad Politécnica de Madrid (Navas-Loro et al., 2022) [10]. Con ello, se pretende establecer una metodología más rigurosa y efectiva en la asignación de estos códigos.
Asimismo, la mejora de la eficiencia y la precisión de la clasificación de productos y servicios fomenta la transparencia y la equidad en los procesos de contratación pública, lo que puede generar una mayor eficiencia y eficacia en los procesos de contratación, así como un ahorro significativo de tiempo y capital para la UE.

1.1.2 Objetivos específicos

A fin de conducir adecuadamente la realización del proyecto es necesario establecer una serie de objetivos de carácter específico que reduzcan la complejidad del trabajo y son los siguientes:

- Mejora de la precisión y la velocidad en la asignación de códigos CPV: el clasificador debe ser capaz de identificar automáticamente el código CPV correspondiente a la descripción del producto o servicio, reduciendo así la necesidad de una clasificación manual y mejorando la velocidad y la precisión de la asignación.

- Reducción de los errores de clasificación: el clasificador debe ser capaz de detectar y corregir errores en la descripción del producto o servicio, y asignar el código CPV correcto en caso de ambigüedad o falta de

información.

- Clasificación coherente y unificada: el clasificador debe ser capaz de proporcionar una clasificación coherente y unificada para todos los productos y servicios en la UE, lo que puede mejorar la transparencia y la equidad en los procesos de contratación pública.

- Actualización regular para incluir nuevas categorías de productos y servicios: el clasificador debe ser capaz de actualizarse regularmente con el objetivo de incluir nuevas categorías de productos y servicios, lo que puede mejorar la capacidad del clasificador para satisfacer las necesidades cambiantes de la contratación pública en la UE.

1.2 Estructura del documento

El documento se compone de cuatro partes:

- **Estado del arte**
  En esta sección, se revisa la literatura existente y las investigaciones realizadas en el campo de la clasificación de códigos CPV. Se describen los métodos y enfoques existentes para abordar el problema, incluidos los algoritmos de clasificación, las técnicas de procesamiento del lenguaje natural y las herramientas específicas para el análisis de texto en español. Además, se discuten las ventajas y desventajas de estos enfoques y cómo se relacionan con el proyecto en cuestión.

- **Desarrollo**
  En la sección de desarrollo, se detallan el proceso de diseño y de implementación del proyecto. Asimismo, se exponen las decisiones tomadas en cuanto a la selección de los clasificadores, la preparación y el preprocesamiento de los datos, y la creación de un pipeline de clasificación. También se explican las técnicas de validación y optimización del modelo, como la división de los datos en conjuntos de entrenamiento y prueba y la búsqueda de los mejores hiperparámetros.

- **Resultado**
  Este apartado presenta los resultados del proyecto, entre los que se cuentan las métricas de evaluación del rendimiento del modelo, como la precisión, la exhaustividad, el F1-score y la matriz de confusión. Por un lado, los resultados obtenidos se comparan por medio de diferentes clasificadores, por otro, se analizan las posibles causas que originan el distinto rendimiento. Además, se proporcionan ejemplos de predicciones realizadas por el modelo, junto con discusiones sobre casos de éxito y posibles mejoras.

- **Conclusiones**
  En la sección de conclusiones, se resume el trabajo realizado y se destacan los principales hallazgos del proyecto. Se discuten las implicaciones de los resultados obtenidos y se identifican las limitaciones del enfoque actual y las áreas de mejora. Por último, se sugieren posibles direcciones futuras para la investigación en el campo de la clasificación de códigos CPV, incluidas las extensiones del trabajo actual y la exploración de nuevas técnicas y enfoques.

2 Estado del arte

2.1 Contrataciones públicas

En la UE, la contratación pública se rige por un conjunto de directivas y regulaciones que buscan garantizar la transparencia, la competencia leal, la igualdad de trato y la eficiencia en el proceso de adjudicación de contratos. Estas directivas incluyen la Directiva 2014/24/UE sobre contratación pública y la Directiva 2014/25/UE sobre contratación por entidades que operan en los sectores del agua, la energía, los transportes y los servicios postales [3].
En el ámbito académico y de investigación, se han llevado a cabo estudios sobre diversos aspectos de la contratación pública, licitaciones y contratos en la UE. Estos estudios abordan temas como la eficiencia y la transparencia en la adjudicación de contratos, el análisis de los patrones de contratación y la identificación de riesgos de corrupción, así como la aplicación de tecnologías de la información y comunicación en la contratación pública (A. Ramírez Sixtos, 2022) [7].
En particular, la clasificación automática de códigos CPV ha sido un tema de interés en la investigación, puesto que facilita la gestión de las licitaciones y permite a las entidades públicas y los proveedores identificar rápidamente las oportunidades de contratación relevantes. Se han utilizado enfoques de procesamiento del lenguaje natural y aprendizaje automático para mejorar la precisión en la clasificación de códigos CPV, incluidos algoritmos de clasificación tradicionales y modelos pre entrenados de lenguaje, como BERT.
A pesar de los avances en la investigación sobre contratación pública en la UE, aún existen brechas en áreas como la clasificación de códigos CPV en idiomas distintos al inglés y la adaptación de las técnicas de clasificación a los desafíos específicos de la contratación pública. Este estado del arte busca proporcionar una base sólida para abordar estos desafíos en proyectos futuros y contribuir al desarrollo de soluciones efectivas y eficientes en el ámbito de la contratación pública en la UE.

2.2 Sistema de categorización de productos

Un sistema de categorización de productos es un esquema estructurado que permite organizar y clasificar productos y servicios en diferentes categorías y subcategorías según sus características, funcionalidades o aplicaciones. Este tipo de sistema facilita la búsqueda, comparación y análisis de productos y servicios en diversas industrias y contextos, como la contratación pública, el comercio electrónico y la gestión de inventarios. Los sistemas de categorización de productos pueden variar en términos de estructura y complejidad, pero generalmente siguen un enfoque jerárquico, donde las categorías principales se dividen en subcategorías más específicas.

2.2.1 Esquema CPV

El CPV se organiza de forma jerárquica, que se divide en grupos, clases y categorías. Cada nivel de la jerarquía se representa mediante un conjunto de dígitos numéricos. Por ejemplo, un código CPV completo consta de 8 dígitos y un dígito de control (A. Ramírez Sixtos, 2022) [7]. El dígito de control se utiliza

para verificar la exactitud del código CPV. El sistema CPV se actualiza regularmente para reflejar las nuevas necesidades y tendencias del mercado, y es obligatorio su uso en todos los procedimientos de contratación pública en la UE. Además, también puede utilizarse en otros contextos, como la gestión de inventarios y la clasificación de productos y servicios en general.
El formato de estos códigos CPV (A. Ramírez Sixtos, 2022) [7] siguen una estructura de árbol de cinco niveles compuesta por los siguientes dígitos:

- **Divisiones:** Se identifican mediante los dos primeros dígitos del código (XX000000-Y).
- **Grupos:** Se identifican mediante los tres primeros dígitos del código (XXX00000-Y).
- **Clases:** Se identifican mediante los cuatro primeros dígitos del código (XXXX0000-Y).
- **Categorías:** Se identifican mediante los cinco primeros dígitos del código (XXXXX000-Y).

Cada uno de los tres últimos dígitos proporciona un mayor nivel de detalle dentro de cada categoría. Además, un noveno dígito se utiliza para verificar la exactitud de los dígitos anteriores. El vocabulario suplementario puede emplearse para proporcionar una descripción más detallada del objeto del contrato. Consiste en un código alfanumérico asociado a un enunciado que permite especificar con mayor precisión la naturaleza o el propósito particular del bien a adquirir [8].
Un ejemplo de un código CPV podría ser "34144210-3" y se desglosa de la siguiente forma:

Nivel	Código	Descripción
División	34 (XXXX0000-Y)	Vehículos a motor y sus partes.
Grupo	341 (XXX00000-Y)	Vehículos automotores.
Clase	3414 (XXXX0000-Y)	Vehículos especiales.
Categoría	34144 (XXXXX000-Y)	Vehículos de transporte de emergencia.
Subcategoría	341442 (XXXXXX00-Y)	Vehículos de bomberos.
Ítem específico	34144210 (XXXXXXX0-Y)	Vehículos de bomberos con equipo especializado.

Tabla 1. Ejemplo de código CPV

El código completo, 34144210-3, representa vehículos de bomberos con equipo especializado dentro de la categoría de vehículos de transporte de emergencia. Estos vehículos pueden incluir características adicionales o equipos especializados para hacer frente a situaciones de emergencia específicas o para mejorar su capacidad de respuesta en diferentes escenarios.

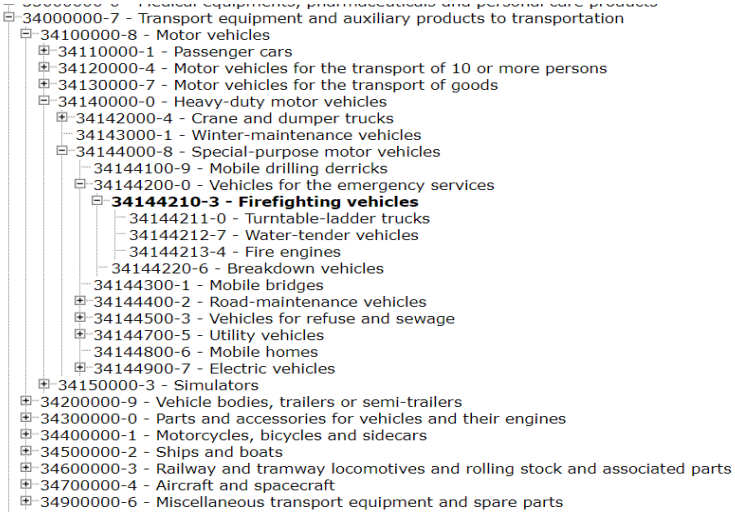

Figura 1. Estructura Jerárquica del código 34144210.
http://www.cpv.enem.pl/en/34144210-3

2.3 Proyectos relacionados con el presente trabajo

A continuación, se describen trabajos relevantes relacionados con la clasificación de CPV.

2.3.1 Las licitaciones públicas: análisis de datos y sistemas predictivos utilizando métodos de aprendizaje automático

Esta tesis (García Rodríguez, 2022) [9] aplica la ciencia de datos a la contratación pública, combinando conocimientos matemáticos-estadísticos, habilidades de programación y conocimientos del objeto de estudio. Se analizan datos de licitaciones en España y otros países, abordando problemas en este campo mediante algoritmos de aprendizaje automático, parte de la inteligencia artificial. La investigación es innovadora en el ámbito académico, gubernamental y privado. El estudio repasa la contratación, la ciencia de datos, sus evoluciones históricas y los desafíos actuales. Se exploran organismos públicos españoles relacionados con la contratación, la transparencia, los datos abiertos y las iniciativas de la Comisión Europea (la tesis consta de cuatro artículos que abordan la regulación, análisis de datos, estimación del importe de adjudicación, recomendación de licitadores y detección de colusión en licitaciones mediante el uso de aprendizaje automático). Los artículos ofrecen ejemplos prácticos para ayudar a diferentes actores involucrados en la contratación y, en última instancia, a la sociedad.

2.3.2 Clasificador de Códigos de Licitaciones Públicas: Caso de Estudio España y México

El objetivo de esta tesis (A. Ramírez Sixtos, 2022) [7] es desarrollar un modelo de clasificación multiclase para los esquemas de clasificación de licitaciones públicas en España y México. En España, se utiliza el estándar CPV (Common Procurement Vocabulary), mientras que en México se emplea el Clasificador

Único de las Contrataciones Públicas (CUCoP). Ambos estándares tienen como objetivo la unificación de las referencias utilizadas por los órganos de contratación para describir el objeto del contrato en una licitación.

2.3.3 Multi-label Text Classification for Public Procurement in Spanish

Este artículo (Navas-Loro et al., 2022) [10] aborda la creación de modelos de clasificación multietiqueta con el objetivo de asignar objetos de contrato a 45 categorías principales del estándar CPV. Los modelos se basan en descripciones textuales en español de procedimientos de contratación del año 2019, publicadas en el portal de datos abiertos del Ministerio de Hacienda de España. Los investigadores destacaron que este enfoque puede adaptarse a otros idiomas sin mayores dificultades. Se utilizaron varios algoritmos de clasificación conocidos, como Naïve bayes, KNN, Decision tree, Random forest y SVM. De todos ellos, al modelo SVM obtuvo los mejores resultados con un F1-score de 0.69, superando el modelo de Kaan Görgün, que logró un F1-score de 0.64. Además, se experimentó con modelos basados en MarIA, que es un modelo RoBERTa transformed-based (A. Gutiérrez-Fandiño et al., 2021) [11] entrenado con un corpus de la Biblioteca Nacional de España. El mejor de estos modelos alcanzó un F1-score de 0.80, superando a los otros modelos evaluados.

2.4 Procesamiento de lenguaje natural

En esta sección se exponen algunas técnicas utilizadas para la limpieza de datos basados en textos, útiles para obviar caracteres y palabras innecesarias durante la aplicación de los modelos de clasificación.

2.4.1 Lowercasing

Lowercasing remite a una técnica básica de preprocesamiento de texto en el *procesamiento del lenguaje natural* (PLN) que consiste en convertir todas las letras del texto a minúsculas. Esta técnica resulta fundamental para homogeneizar y estandarizar el texto, lo que ayuda a mejorar la precisión y la eficiencia de los algoritmos de PLN [13]. *Lowercasing* es especialmente útil en el análisis de contratos públicos, pues permite identificar y comparar palabras clave y términos de manera efectiva, independientemente de las diferencias en el uso de mayúsculas y minúsculas en el texto original.

2.4.2 Tokenización

La tokenización es una técnica esencial de preprocesamiento de texto en el PLN que implica dividir el texto en unidades más pequeñas llamadas tokens, correspondientes a palabras, frases o símbolos ⁅13]. La tokenización permite analizar y procesar el texto de manera más eficiente y resulta fundamental para tareas de PLN como la clasificación de documentos, el análisis de sentimiento y la extracción de información. En el contexto de los contratos públicos, la tokenización se puede utilizar para extraer y analizar información

relevante, facilitar la búsqueda de palabras clave y patrones, así como mejorar la precisión y eficiencia de los algoritmos de PLN.

2.4.3 Eliminación de ruido

La eliminación de ruido o *noise removal* es una técnica de preprocesamiento de texto en PLN que consiste en eliminar palabras, caracteres o símbolos irrelevantes o innecesarios del texto [13]. La eliminación de ruido mejora la calidad del texto y facilita el análisis y la interpretación del contenido de los documentos. En el contexto de los contratos públicos, la eliminación de ruido ayuda a identificar y extraer información relevante y significativa con mayor precisión.

2.4.4 La eliminación de palabras vacías

La eliminación de palabras vacías o *stop-word removal* es una técnica de preprocesamiento de texto en PLN que consiste en eliminar palabras comunes que no aportan información significativa al análisis [13]. En el contexto de contratos públicos, la eliminación de palabras vacías mejora la eficiencia del análisis y facilita la extracción de información relevante.

2.4.5 Lematización

La lematización es una técnica de preprocesamiento de texto en PLN que consiste en reducir las palabras a su forma base o raíz, mejorando así la comparabilidad y el análisis de los datos de texto [13]. En el contexto de contratos públicos, la lematización puede facilitar la identificación y extracción de información relevante al unificar palabras que tienen el mismo significado, pero diferentes formas gramaticales.

2.4.6 Técnicas de vectorización

Las técnicas de vectorización en PLN son esenciales para transformar el texto en representaciones numéricas que puedan ser utilizadas por algoritmos de aprendizaje automático y minería de datos [13]. La aplicación de estas técnicas en los contratos públicos permite analizar y extraer información relevante de los documentos, facilitando la clasificación, agrupación, comparación y detección de patrones.
La aplicación de técnicas de vectorización en los contratos públicos es crucial para abordar desafíos en la clasificación de contratos, detección de fraude y colusión, análisis de tendencias y patrones, y la generación de información útil para la toma de decisiones en el ámbito de la contratación pública.

2.4.6.1 Bag of words

La aplicación de esta técnica en los contratos públicos permite convertir el texto de los documentos en representaciones numéricas en forma de vectores de recuento de palabras. Este enfoque, también conocido como bolsa de

palabras (*bag of words*) [13], se basa en contar la frecuencia de cada palabra en el documento, sin tener en cuenta el orden o la estructura gramatical del texto. El uso de esta técnica en el análisis de contratos públicos puede tener múltiples aplicaciones, como la clasificación automática de contratos según su contenido, la identificación de palabras clave y temas comunes, y la detección de patrones y anomalías en el lenguaje utilizado en los documentos.

2.4.6.2 TF-IDF (Term Frequency-Inverse Document Frequency)

Otra técnica más avanzada es el modelo de frecuencia de términos: *Term Frequency-Inverse Document Frequency* (TF-IDF) (traducido al español como relevancia en la búsqueda de información) [14], que pondera las palabras en función de su importancia tanto en el documento como en el corpus completo. Esto permite que palabras menos comunes, pero más relevantes tengan un mayor peso en la representación vectorial, lo que mejora la precisión en tareas de clasificación y búsqueda de información. A continuación, se muestra una parte del código donde se utiliza la TF-IDF:

```
# Crear un vectorizador TF-IDF para transformar la descripción

vectorizer = TfidfVectorizer(max_features=1000)

# Entrenar el vectorizador en los datos de entrenamiento

X_train_tfidf = vectorizer.fit_transform(X_train['description'])

# Aplicar el vectorizador a los datos de prueba

X_test_tfidf = vectorizer.transform(X_test['description'])
```

2.4.7 Algoritmos de clasificación

A continuación, se describen los principales algoritmos de clasificación, así como algunas de sus implementaciones.

2.4.7.1 Random forest

Random forest (A. Géron, 2023) [21] es un algoritmo de aprendizaje supervisado basado en árboles de decisión que puede ser utilizado para la clasificación y regresión. En el caso de predecir los códigos CPV en contratos públicos, se aplica el Random forest para clasificar los contratos según sus categorías, basándose en características extraídas del texto de los contratos. El modelo Random forest tiene un enfoque sólido y versátil para la clasificación en problemas de PLN, como predecir códigos CPV.

2.4.7.2 Decision tree

Decision tree (Árbol de decisión) (A. Géron, 2023) [21] es un algoritmo de aprendizaje supervisado que puede utilizarse para la clasificación y regresión. Para predecir los códigos CPV en contratos públicos, se utiliza el árbol de

Decisión con el objetivo de clasificar los contratos en función de sus respectivas categorías, tomando como base las características derivadas del contenido textual de los contratos.
El modelo Decision tree es un enfoque interpretable y fácil de visualizar para la clasificación en problemas de PLN, como predecir códigos CPV. Sin embargo, puede ser propenso al sobreajuste, especialmente cuando el árbol se vuelve demasiado complejo. Por lo tanto, es importante ajustar los parámetros y experimentar con diferentes técnicas de preprocesamiento y vectorización para mejorar el rendimiento del modelo.

2.4.7.3 MutinominalNB

El clasificador multinomial Naive Bayes (A. Géron, 2023) [21] es un algoritmo de aprendizaje automático supervisado basado en el teorema de Bayes, que asume la independencia condicional entre las características. Este algoritmo es especialmente adecuado para la clasificación de texto y, en este caso, se utiliza para predecir los códigos CPV en contratos públicos.
Este clasificador es una opción simple y eficiente para problemas de clasificación de texto, como la predicción de códigos CPV. Sin embargo, dado que asume la independencia entre las características, puede no funcionar tan bien cuando las palabras están fuertemente relacionadas entre sí (J. Grus, 2015) [16]. Para mejorar el rendimiento del modelo, es fundamental probar diferentes técnicas de preprocesamiento y vectorización, así como ajustar los parámetros del algoritmo.

2.4.7.4 Gradient boosting

El clasificador Gradient boosting (J. D. Kelleher et al., 2015) [15] es un algoritmo de aprendizaje automático supervisado basado en el ensamblaje de árboles de decisión. Este algoritmo mejora el rendimiento mediante la combinación de varios árboles de decisión débiles en un modelo más fuerte y preciso.
Este clasificador es una opción potente y flexible para problemas de clasificación de texto, como la predicción de códigos CPV. Sin embargo, puede ser más complejo y lento para entrenar en comparación con otros algoritmos, como Naive bayes o Decision tree. Para mejorar el rendimiento del modelo, es crucial ajustar los parámetros del algoritmo, como la profundidad máxima de los árboles, la tasa de aprendizaje y el número de estimadores, así como probar diferentes técnicas de preprocesamiento y vectorización.

2.4.7.5 Support vector machine

Las máquinas de vectores de soporte (en inglés como *support vector machine* o SVM) (A. Géron, 2023) [21] son un tipo de algoritmo de aprendizaje supervisado que se utiliza para tareas de clasificación y regresión. El objetivo de una SVM es encontrar el hiperplano que mejor divide el espacio de características de tal manera que los elementos de diferentes clases se encuentren en lados opuestos del hiperplano tanto como sea posible.
Cuando se aplica a los códigos de vocabulario común de producto (CPV), una SVM podría utilizarse para clasificar las descripciones de las licitaciones en diferentes categorías, según los códigos CPV. En este caso, el espacio de

características podría ser un espacio de alta dimensión generada por algún método de vectorización del texto, como TF-IDF o word2vec.

3 Desarrollo

En este capítulo se explora en profundidad el papel crucial que desempeñan los códigos CPV en el análisis y la clasificación de licitaciones públicas.

El desarrollo en este contexto implica la creación e implementación de algoritmos de PLN y de aprendizaje automático para analizar y clasificar licitaciones basándose en los códigos CPV. Además, se examinará cómo se pueden desarrollar modelos predictivos que permitan anticipar tendencias y comportamientos futuros en las licitaciones públicas.

Este capítulo también tiene como objetivo ofrecer una perspectiva completa de la utilización de los códigos CPV en nuestro estudio, proporcionando al lector la información necesaria para comprender cómo estos códigos pueden optimizar los modelos de análisis y predicción en el contexto de las licitaciones públicas. Mediante una combinación equilibrada de teoría y aplicación práctica, aspiramos a esclarecer la complejidad inherente al uso de los códigos CPV, así como demostrar cómo se pueden gestionar de manera eficaz.

3.1 Recolección de los datos

La adquisición de los datos para este proyecto de investigación se llevó a cabo a través de la web de la Hacienda española, que es una fuente invaluable de información sobre las licitaciones públicas en España[1]. Esta web ofrece una gran cantidad de datos sobre las licitaciones, incluyendo información detallada sobre cada contrato, los códigos CPV asociados, los licitadores y las entidades adjudicatarias, entre otros. Este conjunto de datos se publica mensual y anualmente.

El portal de la Hacienda española ha sido diseñado con el objeto de ser transparente y accesible, permitiendo a los usuarios descargar conjuntos de datos completos para su análisis. Para este estudio, se seleccionaron y descargaron varios conjuntos de datos que se ajustaban a los requisitos de la presente investigación. Estos conjuntos de datos contienen una gran cantidad de información, incluyendo detalles sobre los contratos, las entidades adjudicatarias y los licitadores, así como los códigos CPV utilizados para clasificar los contratos. Estos conjuntos de datos son publicados en ficheros comprimidos que pueden ser descargados como se muestra en la figura 2.

[1] https://www.hacienda.gob.es/es-ES/GobiernoAbierto/Datos%20Abiertos/Paginas/LicitacionesContratante.aspx

Figura 2. Ficheros de licitaciones públicas

Fuente:https://www.hacienda.gob.es/es-ES/GobiernoAbierto/Datos%20Abiertos/Paginas/LicitacionesContratante.aspx

3.2 Exploración de los datos

En esta sección del proyecto, se realiza el análisis exploratorio de los datos relativos a los códigos CPV de tres dígitos.

El análisis exploratorio de datos es una fase crítica en cualquier proyecto de investigación, ya que permite comprender las características y patrones subyacentes de los datos. Este trabajo se centra en los códigos CPV de tres dígitos, que representan la categorización de alto nivel de productos, servicios y obras en el sistema de contratación pública de la UE.

A lo largo de esta sección, se emplean diversas técnicas de análisis de datos para desentrañar patrones, detectar anomalías y comprender profundamente la distribución de los códigos CPV de tres dígitos presentes en nuestro conjunto de datos. Es relevante mencionar que este dataset exhibe una desproporción entre las clases, lo que provoca tanto retos como oportunidades a la hora de estudiarlo. El objetivo es generar una comprensión sólida y detallada de estos códigos, estableciendo una base robusta que facilite la efectividad de las etapas sucesivas de la presente investigación.

3.2.1 Exploración de CPV

La investigación se ha desarrollado en el contexto del proyecto europeo NextProcurement[2]. Los datos utilizados en el presente proyecto se han obtenido por medio de dicho proyecto o a través de fuentes públicas de datos, como la web de la Hacienda española.

El conjunto de datos empleado en este estudio comprende 385 660 registros distribuidos en 318 categorías distintas. Cada registro está compuesto por una descripción detallada de la licitación y el código CPV correspondiente. A lo largo de las siguientes secciones, se desglosará el proceso sistemático adoptado para predecir con precisión el tercer dígito del código CPV. Cabe mencionar que la integridad del conjunto de datos es óptima, ya que no contiene valores nulos. Esta característica simplifica significativamente el proceso de exploración de datos y el preprocesamiento posterior, permitiendo una transición fluida hacia las etapas de modelado y evaluación.

La figura 3 proporciona una representación gráfica e intuitiva de la distribución de las observaciones según la frecuencia de aparición de cada código CPV.

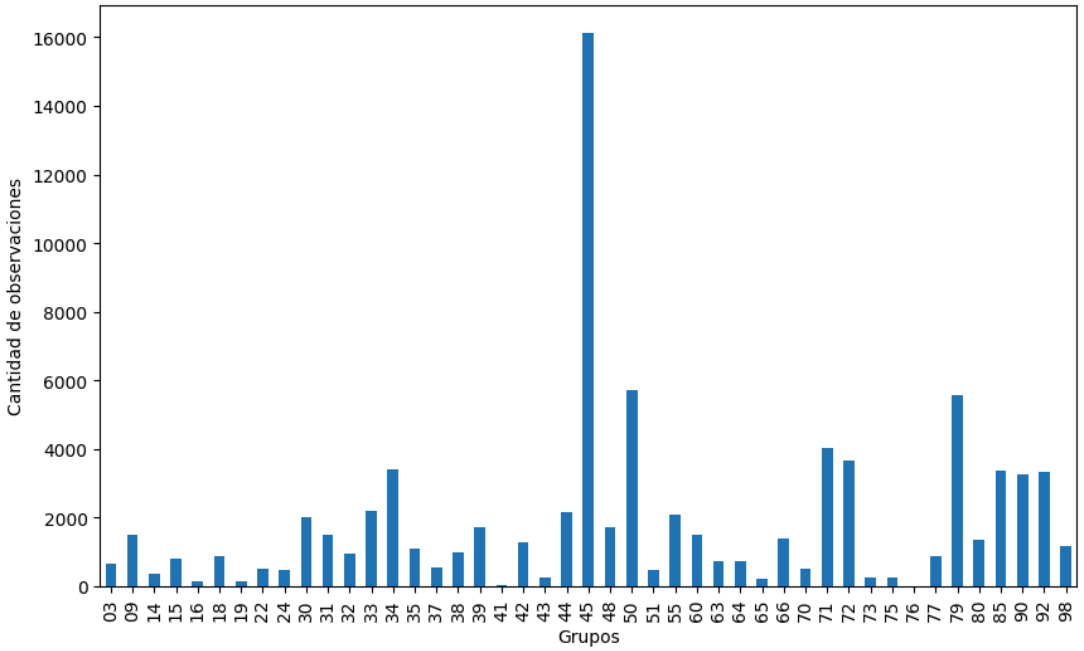

Figura 3. Distribución de observaciones por CPV

Seguidamente, se muestra la distribución de 3 de los 45 grupos junto a sus respectivas categorías, debido a la imposibilidad de realizar un análisis del total. Específicamente se ha elegido para realizar el análisis el grupo 60 (por ser el menos representativo) y el grupo 45 (por ser el más representativo).

La gráfica subsiguiente ilustra la distribución de las predicciones para el tercer dígito en el conjunto de datos asociado a los dos primeros dígitos del código CPV '60'. Se puede apreciar que la mayoría de las observaciones se clasifican dentro de la categoría 601. Por contraste, la categoría 602

[2] http://nextprocurement-project.com/

representa la proporción más baja de las observaciones, evidenciando la menor frecuencia de esta clase en los datos analizados.

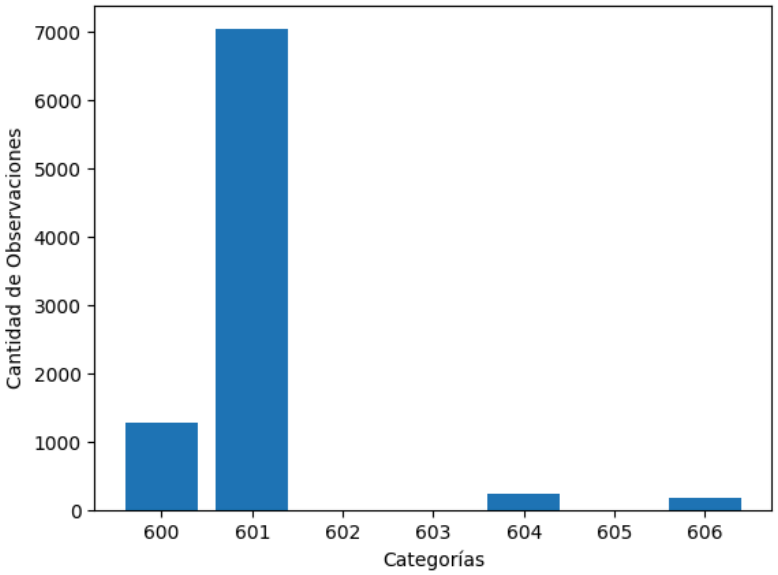

Figura 4. Distribución de la clase 60

La siguiente gráfica ilustra la distribución de las predicciones para el tercer dígito del código CPV, de aquellos empezados por 45. Este grupo se caracteriza por ser una de las clases más equilibradas, esto se puede atribuir a la variedad de contratos que pertenecen a la clase 45. Como se puede apreciar, las clases 450, 451, 453 y 454 se encuentran en un rango de observaciones que no supera las 2000 unidades. Sin embargo, la clase 452 destaca por contener un número significativamente mayor de observaciones.

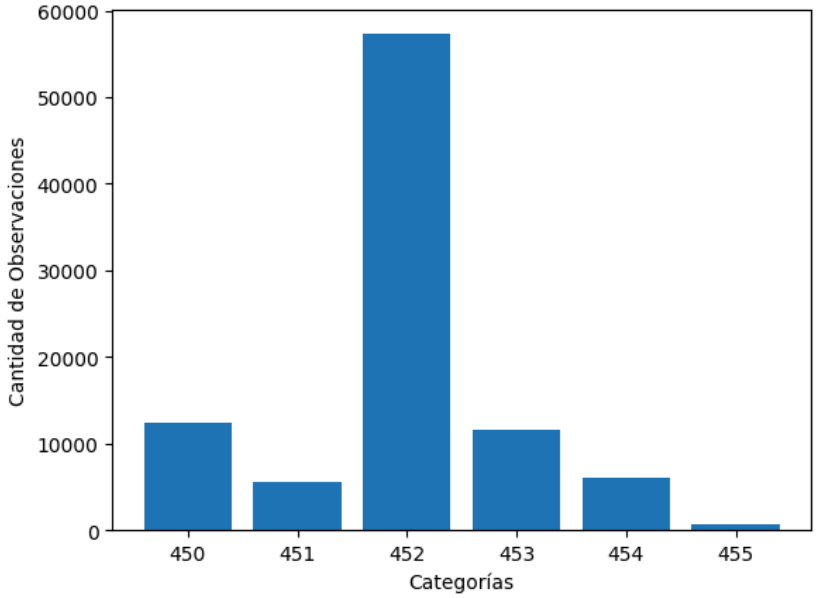

Figura 5. Distribución de la clase 45

El diagrama siguiente se centra en la distribución de las predicciones del tercer dígito para los códigos CPV que inician con 42. Este subconjunto destaca por tener la menor cantidad de observaciones en la clase 428 dentro de todo el conjunto de datos. Al examinar la distribución de los códigos de tres cifras dentro de esta agrupación, es evidente que la clase 429 recoge la mayoría de las observaciones que corresponden a esos dos dígitos particulares.

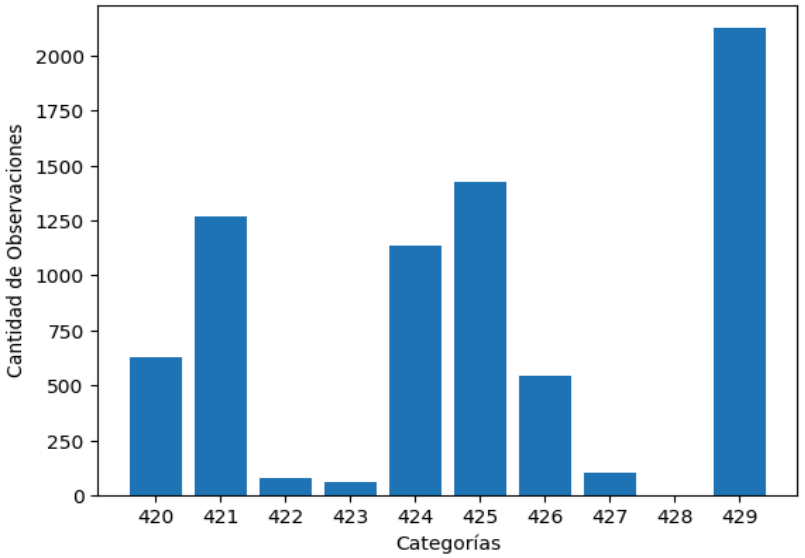

Figura 6. Distribución de la clase 42

La gráfica subsecuente destaca la distribución de predicciones para el tercer dígito del código CPV, específicamente para los códigos que empiezan con 14. En este conjunto, la clase 142 emerge con un número particularmente elevado de observaciones, lo que la distingue como la subclase más predominante dentro de la clase principal 14. Esta tendencia indica un mayor grado de especificidad en las licitaciones públicas que caen dentro de este grupo particular de códigos CPV.

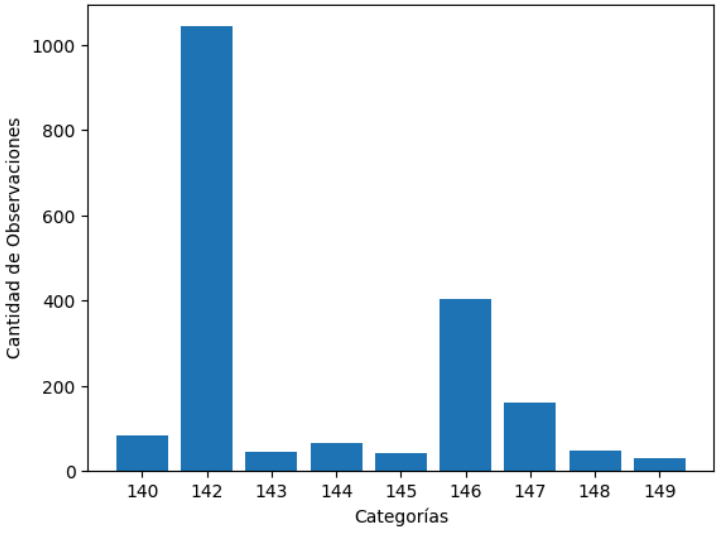

Figura 7. Distribución de la clase 14

La siguiente gráfica muestra la distribución de las predicciones para el tercer dígito del código CPV de aquellos que iniciados con 80. En este segmento, es notable la prevalencia de la subclase 805, la cual se distingue por poseer un número considerablemente superior de observaciones en comparación con las demás subclases. Este hecho sugiere una marcada prevalencia de licitaciones públicas que se ajustan a esta subcategoría específica dentro de la clase principal 80 en los códigos CPV.

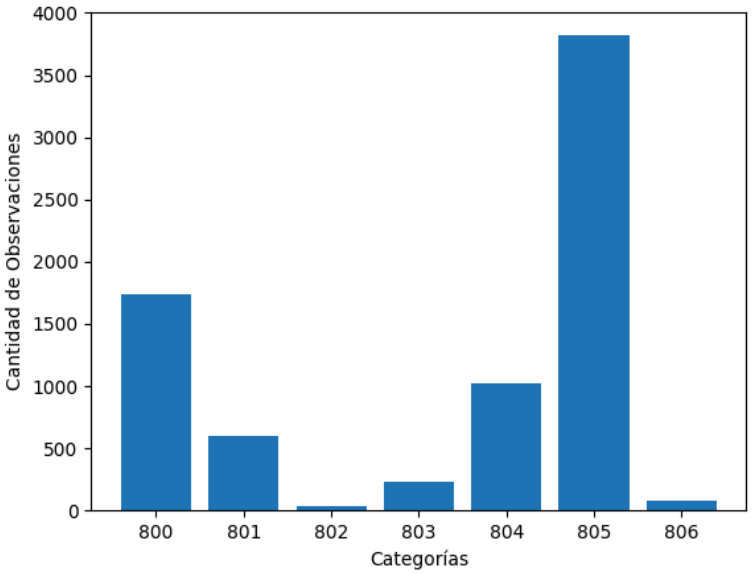

Figura 8. Distribución de la clase 80

3.3 Preprocesamiento de los datos

Una vez se ha realizado la exploración de los datos, se procede con la etapa de preprocesamiento y limpieza de los datos. Hay que recalcar que este preprocesamiento de datos solo se ajusta al campo de la descripción, porque contiene la información del bien o servicio, que es el input de nuestro clasificador. Este proceso implica preparar y limpiar los datos de texto para su posterior análisis y modelado. El objetivo principal es transformar los datos en un formato que pueda ser fácilmente interpretado por los algoritmos de PLN, maximizando la eficacia y precisión de los modelos resultantes. SpaCy es la librería escogida para aplicar las técnicas *lowercasing*, tokenización, eliminación del ruido, eliminación de palabras vacías y lematización.

A continuación, se muestra un ejemplo de preprocesamiento de los datos con una descripción que tiene los resultados de cada una de las técnicas aplicadas.

`Descripción: "Prestación del servicio de Mantenimiento del equipo contra incendios de la Delegación del Gobierno."`

La aplicación de la técnica de *lowercasing* devuelve el siguiente resultado:

`Lowercasing: "prestación del servicio de mantenimiento del equipo contra incendios de la delegación del gobierno."`

En segundo lugar, se procede a la tokenización, a cuya ejecución siguen las técnicas de eliminación del ruido y lematización.

```
Tokenization, noise removal and lemmatization: ['prestación', 'del', 'servicio', 'de', 'mantenimiento', 'del', 'equipo', 'contra', 'incendio', 'de', 'el', 'delegación', 'del', 'gobierno']
```

Por último, a partir de la técnica de eliminación de palabras vacías, se obtiene el siguiente resultado:

```
Stop words removal: ['prestación', 'servicio', 'mantenimiento', 'equipo', 'incendio', 'delegación', 'gobierno']
```

Hay que tener en cuenta que la lematización en spaCy depende del contexto de las palabras, por lo que normalmente se realiza en el documento completo antes de realizar operaciones eliminación del ruido y eliminación de palabras vacías.

Para aplicar todas estas técnicas de procesamiento al conjunto descripciones, se ha creado la siguiente función a fin de poder agilizar el proceso de procesamiento de textos:

```
def preprocess_text(text):

    # Lowercasing, procesar el texto y crear un objeto Doc

    doc = nlp(text.lower())


    # Noise removal and lemmatization

    words = [token.lemma_ for token in doc if not token.is_punct]


    # Stop word removal

        words = [word for word in words if word not in nlp.Defaults.stop_words]


    # Rejoin words

    text = ' '.join(words)


    return text
```

El preprocesamiento de los datos es una etapa fundamental y crítica en cualquier proyecto de PLN. A través de técnicas como las descritas, se acomete la preparación y transformación de los datos en un formato que puede ser fácilmente interpretado y procesado por los algoritmos de PLN. Estos pasos son esenciales para reducir el ruido y la redundancia, así como para dirigir el análisis hacía las características más significativas y relevantes del texto. Tras

el preprocesamiento de los datos, se pasa a la fase de modelado y análisis, donde se aplican diversas técnicas de PLN para extraer ideas útiles y generar predicciones precisas a partir del conjunto de datos.

La figura 9 proporciona una representación del proceso completo de procesamiento de los datos.

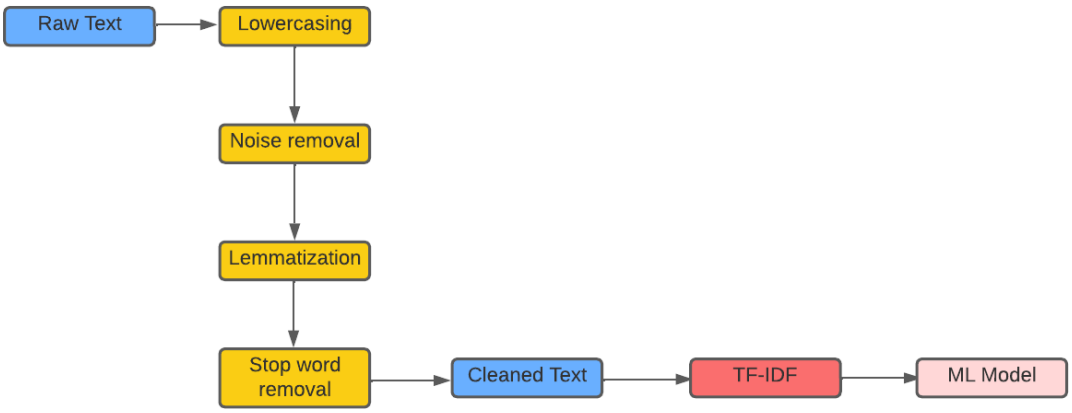

Figura 9. Preprocesamiento de los datos

3.4 Modelado y Evaluación

Después de la recolección y preprocesamiento de los datos, la investigación se embarca en la aplicación de diversas técnicas de modelado dirigidas a extraer conocimientos significativos y realizar predicciones precisas. Este proceso de modelado es fundamental ya que se convierte en el vehículo para interpretar y comprender la información oculta en los datos, proporcionando una plataforma para extraer conocimientos significativos

En cuanto a las librerías utilizadas, en el contexto de Python se recurrió a sklearn o scikit-learn (A. Géron, 2023) [21], una librería esencial para la implementación de algoritmos de aprendizaje automático y análisis de datos, que se complementa con pandas y numpy (A. Géron, 2023) [21], que permiten manipular y operar con datos tabulares y matrices de gran tamaño, respectivamente, siendo fundamentales para el preprocesamiento y manipulación de datos. Para tareas específicas de PLN, spaCy (A. Géron, 2023) [21] y nltk (S. Jugran et al., 2021) [22] proporcionan herramientas robustas y flexibles que facilitan la tokenización, la identificación de entidades, el análisis de dependencias, entre otros. Cuando es necesario guardar y cargar modelos de aprendizaje automático, joblib (A. Géron, 2023) [21] brinda eficiencia permitiendo reutilizar modelos ya entrenados. Por último, para la visualización de datos, seaborn y matplotlib (A. Géron, 2023) [21] se emplean para la generación de gráficos estadísticos atractivos y personalizables, tanto en 2D como en 3D.

El modelado es el proceso de construcción de un modelo matemático que representa los patrones y estructuras subyacentes en nuestros datos. Estos modelos permiten llevar a cabo predicciones sobre nuevas observaciones y entender mejor las relaciones y las interacciones dentro de nuestros datos. El valor de los modelos radica en su capacidad para destilar la complejidad de los

datos en un formato más manejable y comprensible. Sin embargo, es importante recordar que cualquier modelo es una simplificación de la realidad y está sujeto a ciertos límites y supuestos.

En esta sección, se explora una variedad de técnicas de modelado, seleccionando las que mejor se ajusten a nuestros datos y al problema que se trata de resolver. Asimismo, se describe en detalle cómo cada modelo se entrena y se ajusta, y se discuten las ventajas y desventajas de cada enfoque. La selección de modelos es producto de una combinación de la literatura existente, la naturaleza de los datos y los objetivos específicos de investigación.

Además de construir modelos, también se explorarán técnicas para mejorar su rendimiento, como la optimización de hiperparámetros y la regularización. Estos enfoques permiten afinar nuestros modelos para maximizar su precisión y minimizar el sobreajuste. Una vez que los modelos están construidos y optimizados, se procede a la etapa de evaluación. Aquí, se medirá la eficacia de nuestros modelos utilizando una variedad de métricas, como la precisión, la exhaustividad y la puntuación F1. La evaluación permite entender no sólo qué tan bien los modelos se desempeñan al ejecutarlos con los datos de entrenamiento, sino, además, cómo de bien se espera que generalicen a nuevos datos.

El proceso de transformación de las descripciones textuales de cada licitación en valores numéricos, aptos para el procesamiento computacional, es una etapa crucial en el estudio. Este procedimiento, también conocido como vectorización, permite convertir el texto de la descripción de cada contrato en un vector numérico que luego puede ser interpretado por los algoritmos de clasificación. La vectorización es esencial en el campo de la minería de texto y el procesamiento de lenguaje natural, ya que permite que las computadoras, que normalmente sólo comprenden números, puedan interpretar, analizar y manipular información textual. Este proceso no solo permite que las máquinas interpreten el texto, sino que también preserva la semántica y la relación entre las palabras, aspectos críticos para la comprensión del contenido de las licitaciones. Para alcanzar este objetivo, se ha usado la técnica de TF-IDF que asigna pesos a las palabras en un documento, basándose en su frecuencia en el documento y su rareza en un conjunto de documentos. Ayuda a resaltar las palabras más relevantes en un texto específico dentro de un corpus.

Una vez que se ha aplicado la metodología de preprocesamiento descrita anteriormente al conjunto de datos, obtenido de los contratos, se lleva a cabo la implementación de una serie de algoritmos de aprendizaje supervisado. A fin de determinar cuáles de estos algoritmos son los más apropiados para la investigación, se realizan una serie de pruebas rigurosas que buscan evaluar su desempeño, precisión y eficiencia en términos de tiempo y recursos computacionales. A continuación, este proceso de prueba y evaluación, se seleccionan aquellos algoritmos que consistentemente presentan los mejores resultados y se adaptan mejor a las características y requisitos del conjunto de datos.

Luego de las pruebas realizadas los algoritmos que mejores resultados dieron fueron: Random forest y Support Vector Machine. Random forest y Support Vector Machine tienden a ofrecer mejores resultados que los otros algoritmos comentados debido a ciertas características inherentes a estos métodos. Tras evaluar los resultados y los tiempos de entrenamiento, se decidió seleccionar

Random forest (1 minuto 16 segundos para la clase 60 y 25 minutos para la clase 45) como el algoritmo principal para predecir el dígito de los códigos CPV. Esta elección se fundamentó en que Random forest no solo proporcionaba resultados superiores, sino que también lo hacía en un tiempo de entrenamiento más corto y con un consumo más eficiente de los recursos computacionales.

Además, para cada algoritmo se utiliza la técnica RandomizedSearchCV para llevar a cabo el ajuste de hiper parámetros y descubrir los valores de los parámetros que optimizan su rendimiento. Esta técnica hace uso del método de validación cruzada para ofrecer un promedio equitativo del rendimiento del modelo para cada combinación aleatoria de parámetros. La elección de utilizar RandomizedSearchCV para la optimización de hiperparámetros se justifica principalmente por su eficiencia tanto en tiempo como en consumo de recursos. A diferencia de GridSearchCV, que realiza una búsqueda exhaustiva y puede requerir un tiempo considerable y un alto consumo de recursos, especialmente cuando se aplica a un par de códigos CPV de 2 dígitos, RandomizedSearchCV selecciona una muestra aleatoria de combinaciones de hiperparámetros dentro del espacio de búsqueda definido.

Esta aleatorización permite a RandomizedSearchCV explorar el espacio de hiperparámetros de manera más eficiente y rápida, proporcionando resultados competitivos con una fracción del costo de tiempo y recursos comparado con GridSearchCV. En situaciones donde la dimensión del espacio de hiperparámetros es considerable, como en nuestro caso, esta eficiencia resulta especialmente valiosa, permitiendo optimizar el rendimiento de nuestros modelos de clasificación de códigos CPV sin incurrir en excesivas demandas de tiempo y recursos computacionales.

Al tratar con un problema multiclase, los datos se consideran como una serie de problemas binarios, y para evaluar el rendimiento de cada modelo compuesto se han empleado las métricas que se detallan a continuación. Asimismo, dado que las clases no están equilibradas, como se observó en la sección de exploración de datos, estas técnicas permiten un promedio justo del rendimiento de los modelos que se componen de problemas binarios.

- **Precision:** Mide la proporción de verdaderos positivos entre todas las predicciones positivas (verdaderos positivos más falsos positivos). Es útil cuando las falsas alarmas son particularmente costosas (A. Géron, 2023) [21].
- **Recall:** Mide la proporción de verdaderos positivos entre todas las observaciones actuales positivas (verdaderos positivos más falsos negativos). Es útil cuando es importante capturar todos los casos positivos (A. Géron, 2023) [21].
- **F1-score:** Es la media armónica de Precision y Recall. Proporciona un equilibrio entre ambas métricas y es útil cuando se quiere un equilibrio entre Precision y Recall (A. Géron, 2023) [21].
- **Support:** Es el número de muestras reales que se encuentran en cada clase dentro de un conjunto de datos. Es una medida cuantitativa que proporciona información sobre el balanceo de las clases. En un informe de clasificación, por ejemplo, el support para cada clase indica cuántas instancias de esa clase específica existen en los datos de prueba que se utilizan para evaluar el modelo. Esta métrica es importante porque puede influir en otras métricas de evaluación. Por ejemplo, si una clase tiene un support muy bajo (es decir, muy pocas muestras), entonces

incluso una pequeña cantidad de errores en esa clase puede llevar a una baja precisión (A. Géron, 2023) [21].

- **Accuracy:** Es la proporción de predicciones correctas (verdaderos positivos y verdaderos negativos) sobre el total de observaciones. Es útil para tener una vista general del rendimiento del modelo, pero puede ser engañosa si las clases están muy desequilibradas (A. Géron, 2023) [21].
- **Matriz de confusión:** Es una herramienta de visualización utilizada en aprendizaje supervisado. Proporciona una descripción detallada del rendimiento del modelo de clasificación, mostrando de manera explícita los aciertos y errores del modelo (A. Géron, 2023) [21].

En vista de que la meta principal de este proyecto es crear un modelo que prediga de manera eficaz nuevas situaciones, se hace uso de la estrategia común de segmentar los conjuntos de datos originales en subconjuntos, de modo que existan datos de entrenamiento y datos de prueba para la evaluación. En este estudio, la partición de los conjuntos de datos sigue una proporción de 80 a 20, lo que significa que el 80% de las licitaciones se utilizará para el entrenamiento y el 20% restante para las pruebas. Dada la presencia de clases desequilibradas en nuestros conjuntos de datos, se lleva a cabo una división estratificada en los conjuntos de entrenamiento y prueba, manteniendo la proporción de clases en ambas porciones de los datos.

3.4.1 Modelado y evaluación de códigos CPV

En esta sección, se detalla el proceso esencial de modelado y evaluación de los códigos CPV, donde se evalúan los códigos 60 y 45. Este proceso está centrado en el desarrollo y prueba de varios modelos de aprendizaje automático, cada uno con su conjunto único de fortalezas y debilidades, y su rendimiento en la clasificación de los códigos CPV para predecir el tercer dígito, una vez tenemos ya los dos primeros dígitos predicho. Es importante señalar que, para cada uno de los 45 grupos establecidos, se lleva a cabo el entrenamiento del modelo seleccionado con el propósito específico de predecir el tercer dígito del código CPV. Este proceso repetitivo asegura que el modelo se adapte y aprenda de cada grupo de manera individual, maximizando así su capacidad de realizar predicciones precisas para cada caso.

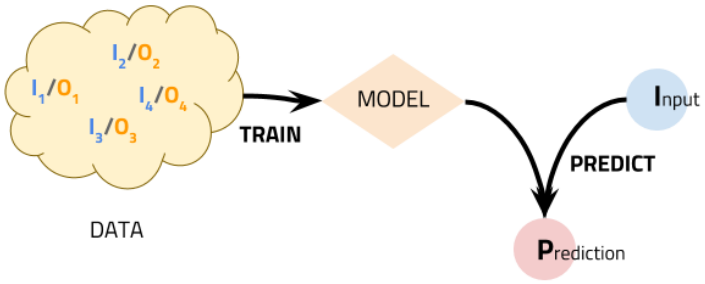

Figura 10. Proceso de predicción del tercer código cpv

Se comienza con la preparación de los datos, dividiendo el conjunto de datos en subconjuntos de entrenamiento y prueba, manteniendo una proporción de 80/20 respectivamente. Este paso es crucial ya que permite entrenar los modelos en una porción de los datos y luego probar su capacidad de generalización en datos no vistos anteriormente. Dada la naturaleza

desequilibrada de las clases en el conjunto de datos, se realiza una división estratificada para preservar la proporción de clases tanto en los conjuntos de entrenamiento como de prueba.

A continuación, se realiza la selección y entrenamiento de los modelos. Entre los modelos considerados se encuentran: la máquina de soporte vectorial, el Random forest, el árbol de decisión, el Naive bayes multinomial y el Gradient boosting. Cada uno de estos modelos es entrenado utilizando el subconjunto de datos de entrenamiento, luego son evaluados en base a su rendimiento en el subconjunto de datos de prueba.

Para afinar estos modelos y buscar el mejor rendimiento, se utiliza la técnica RandomizedSearchCV para optimizar los hiperparámetros. Este método emplea validación cruzada para obtener una evaluación justa y robusta del rendimiento de cada modelo.

Finalmente, se evalúa la eficacia de cada modelo utilizando varias métricas, incluyendo precisión, recall, F1-score y exactitud. Estas métricas nos proporcionan información detallada sobre el rendimiento de cada modelo, permitiéndonos identificar cuál de ellos es el más eficaz en la clasificación de los códigos CPV. Esta evaluación se visualiza claramente con la ayuda de la matriz de confusión, permitiendo una comprensión detallada del rendimiento de cada modelo.

3.4.1.1 Random forest

Tras completar la etapa de procesamiento de datos empleando técnicas avanzadas de PLN, se divide el conjunto de datos en dos partes: entrenamiento y prueba. Posteriormente, se aplica la metodología TF-IDF como preparación para el adiestramiento del modelo. A continuación, se proporciona una explicación detallada de todo el procedimiento:

- Se inicia el clasificador, luego se usa random_state=42 para garantizar la reproducibilidad de los resultados.
- Se crea un objeto de pipeline utilizando el clasificador. Los pipelines son una forma de estandarizar y simplificar el flujo de trabajo de aprendizaje automático, ya que encadenan múltiples pasos de procesamiento y modelado, y todos se ejecutan en secuencia. En este caso, sin embargo, el pipeline solo incluye un paso: el clasificador.
- Se definen las distribuciones de parámetros para la búsqueda aleatoria de hiperparámetros. Estos incluyen:
    - **clf__n_estimators:** Es el número de árboles que se incluirán en el bosque. Más árboles generalmente resulta en un modelo más robusto, pero también puede requerir más tiempo para entrenar y hacer predicciones.
    - **clf__max_depth:** Controla la profundidad máxima de cada árbol. Los árboles más profundos pueden capturar más detalles, pero también pueden sobreajustar.
    - **clf__min_samples_split:** Determina el número mínimo de muestras necesarias para dividir un nodo interno. Si un nodo tiene menos muestras que este número, no se dividirá.
    - **clf__min_samples_leaf:** Especifica el número mínimo de muestras que se requieren para ser un nodo hoja.
- Se inicia RandomizedSearchCV con los siguientes parámetros:

    - **n_iter=10:** Solo se realizarán 10 iteraciones de búsqueda de hiperparámetros. Cada iteración seleccionará una combinación de parámetros de param_dist al azar y entrena y evalúa el modelo utilizando esa combinación.
    - **cv=3:** Se realizará una validación cruzada de tres divisiones. Esto significa que los datos de entrenamiento se dividen en tres partes, y el modelo se entrena y evalúa tres veces, cada vez usando una parte diferente como datos de validación.
    - **scoring="accuracy":** Se evalúa el modelo en base a su precisión, qué es la proporción de predicciones correctas sobre todas las predicciones.
- Se entrena el modelo utilizando el conjunto de datos de entrenamiento (X_train_tfidf, y_train.values.ravel()).
- Finalmente, hace una predicción utilizando el modelo entrenado sobre el conjunto de datos de prueba (X_test_tfidf).

Este procedimiento se puede observar en la siguiente sección de código:

```
clf = RandomForestClassifier(random_state=42)

pipeline = Pipeline([

    ('clf', clf)

])

param_dist = {

        "clf__n_estimators": [100, 200, 300, 400, 500],

        "clf__max_depth": [10, 20, 30, 40, 50, None],

        "clf__min_samples_split": [2, 5, 10, 15],

        "clf__min_samples_leaf": [1, 2, 5, 10],

    }

random_search = RandomizedSearchCV(

        estimator=pipeline,

        param_distributions=param_dist,

        n_iter=10,

        cv=3,

        scoring="accuracy",

        random_state=42,

)

random_search.fit(X_train_tfidf, y_train.values.ravel())

y_pred = random_search.predict(X_test_tfidf)

print(classification_report(y_test, y_pred))
```

Las figuras 11 y 12 presentan las matrices de confusión para los códigos CPV de tres dígitos que inician con 60 y 45, respectivamente. En la figura 11, la escasez de observaciones para algunas subclases de la categoría 60 se refleja en la baja precisión de la clasificación para las clases 2, 3 y 5. En contraste, la figura 12 ilustra una clasificación más precisa para la categoría 45, destacando un mejor rendimiento de clasificación para las subclases 0, 2 y 3. Cabe señalar que algunas descripciones de licitaciones pueden estar asociadas con múltiples clases, lo que agrega una capa adicional de complejidad a la clasificación.

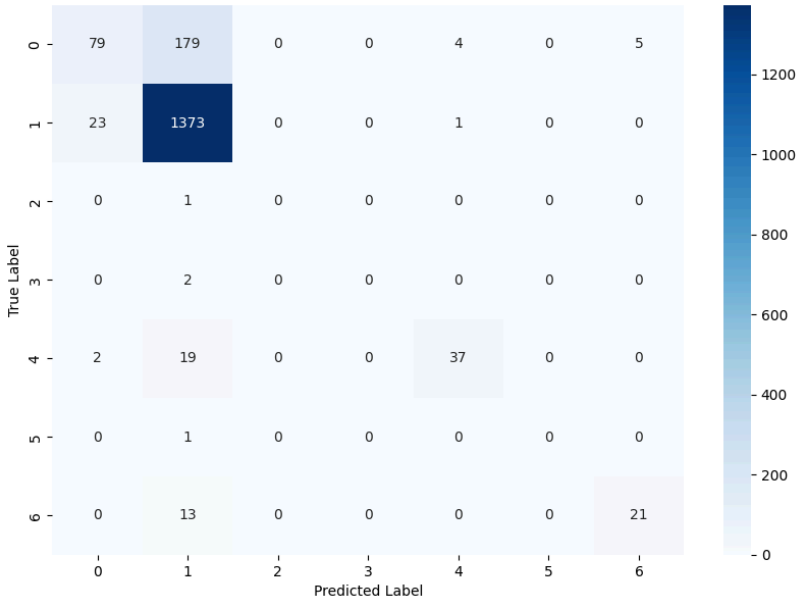

Figura 11. Matriz de confusión para los terceros dígitos de la categoría 60 - Random Forest

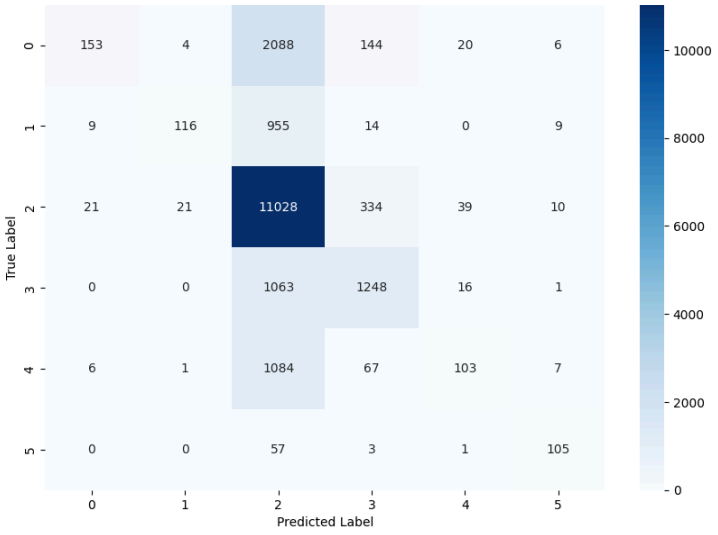

Figura 12. Matriz de confusión para los terceros dígitos de la categoría 45 - Random Forest

Tras completar todas las etapas del proceso, la tabla 2 proporciona una comparativa de los resultados obtenidos al entrenar el modelo con el código 60 y sus terceros dígitos.

Tercer dígito	Precision	Recall	F1-Score	Support
0	0.76	0.30	0.43	267
1	0.86	0.98	0.92	1397
2	0.00	0.00	0.00	1
3	0.00	0.00	0.00	2
4	0.88	0.64	0.74	58
5	0.00	0.00	0.00	1
6	0.81	0.62	0.70	34

Tabla 2. Reporte de clasificación del código 60 y sus terceros dígitos - Random forest

Como se puede apreciar, los dígitos terciarios como 2, 3 y 5 registran un 0% en las métricas de evaluación, resultado que se debe a la presencia de solo 1 o 2 observaciones de estos dígitos en el conjunto de datos. En cuanto al desempeño del modelo aplicado a este código, se alcanza una precisión (*accuracy*) del 85% y el tiempo de entrenamiento se situó en 1 minuto y 16 segundos.

Ahora en la tabla 3 muestra una comparativa de los resultados obtenidos al entrenar el modelo con el código 45 y sus terceros dígitos.

Tercer dígito	Precision	Recall	F1-Score	Support
0	0.81	0.06	0.12	2415
1	0.82	0.11	0.19	1103
2	0.68	0.96	0.80	11453
3	0.69	0.54	0.60	2328
4	0.58	0.08	0.14	1268
5	0.76	0.63	0.69	166

Tabla 3. Reporte de clasificación del código 45 y sus terceros dígitos - Random forest

Es evidente que el dígito 2 exhibe un mayor índice de recall y puntaje f1-score. Este fenómeno se puede atribuir a la mayor cantidad de observaciones de dicho dígito en el conjunto de datos. Respecto al rendimiento del modelo para este código, se alcanza una precisión del 68%, mientras que el periodo de entrenamiento se estableció en 25 minutos.

3.4.1.2 Decision tree

Se realiza el mismo enfoque de modelado, analizando la distribución de los terceros dígitos. A continuación, se ofrece una explicación en profundidad del procedimiento llevado a cabo:

- Se inicia el clasificador. Se asigna random_state=42 para garantizar que cada ejecución del modelo tenga el mismo resultado, siempre y cuando los datos de entrada sean los mismos.
- Se establece un 'pipeline' que sólo contiene el clasificador del árbol de decisiones. En este caso, el pipeline es bastante simple, ya que solo incluye un paso.
- Se definen las distribuciones de los parámetros (param_dist) para la búsqueda aleatoria de hiperparámetros. Estos incluyen:
    - **clf__criterion:** Este parámetro define la función para medir la calidad de una división. gini y entropy son dos métodos comunes para medir la impureza de un nodo.
    - **clf__max_depth:** Este parámetro controla la profundidad máxima del árbol. Un árbol más profundo puede capturar más detalles, pero también puede causar sobreajuste.
    - **clf__min_samples_split:** Este parámetro determina el número mínimo de muestras requeridas para dividir un nodo interno.
    - **clf__min_samples_leaf:** Este parámetro especifica el número mínimo de muestras requeridas para ser un nodo hoja.
- Se inicializa RandomizedSearchCV. Esta herramienta ejecuta una búsqueda aleatoria a través del espacio de hiperparámetros para encontrar la mejor combinación de parámetros. En comparación con GridSearchCV, que prueba todas las combinaciones posibles, RandomizedSearchCV es más eficiente cuando se tiene un gran número de parámetros y/o un conjunto de datos grande. Algunos de los parámetros que se utiliza son:
    - **pipeline:** Este es el estimador base que se va a usar.
    - **param_distributions:** Este es el espacio de parámetros a muestrear.
    - **n_iter:** Este es el número de iteraciones, es decir, el número de combinaciones de parámetros a probar.
    - **cv:** Este es el número de particiones que se usa para la validación cruzada.
    - **verbose:** Este parámetro controla la cantidad de mensajes que se imprimen durante la búsqueda.
    - **random_state:** Este es el generador de números aleatorios que se utiliza para la selección de parámetros.
    - **n_jobs:** Este parámetro controla el número de CPU que se usa para la búsqueda.
- Se ajusta el modelo usando el método fit(). En este paso, se ajusta el modelo a los datos de entrenamiento utilizando varias combinaciones de parámetros, y se seleccionar la combinación que ofrezca el mejor rendimiento basado en la validación cruzada.
- Una vez que el modelo ha sido entrenado y la mejor combinación de parámetros ha sido seleccionada, se usa el modelo para hacer predicciones en los datos de prueba utilizando el método predict(). Las predicciones resultantes se usan para evaluar el rendimiento del modelo.

Este procedimiento se puede observar en la siguiente sección de código:

```
clf = DecisionTreeClassifier(random_state=42)

pipeline = Pipeline([

    ('clf', clf)

])

param_dist = {

    'clf__criterion': ['gini', 'entropy'],

    'clf__max_depth': np.arange(1, 20),

    'clf__min_samples_split': np.arange(2, 20),

    'clf__min_samples_leaf': np.arange(1, 20),

}

random_search = RandomizedSearchCV(

    pipeline,

    param_distributions=param_dist,

    n_iter=100,

    cv=5,

    verbose=1,

    random_state=42,

    n_jobs=-1

)

random_search.fit(X_train_tfidf, y_train.values.ravel())

y_pred = random_search.predict(X_test_tfidf)

print(classification_report(y_test, y_pred))
```

Las figuras 13 y 14 muestran las matrices de confusión correspondientes a los códigos CPV de tres dígitos que comienzan con 60 y 45, respectivamente, generadas a partir del modelo de árbol de decisión. En la figura 13, se evidencia cómo la limitada cantidad de observaciones para algunas subclases dentro de la categoría 60 resulta en una baja precisión de clasificación para las clases 2, 3 y 5, situación que se replicó en el modelo de Random forest. Por otro lado, la figura 14 revela un panorama más alentador para la categoría 45, donde se destacan notables índices de clasificación para las subclases 0, 2 y 3.

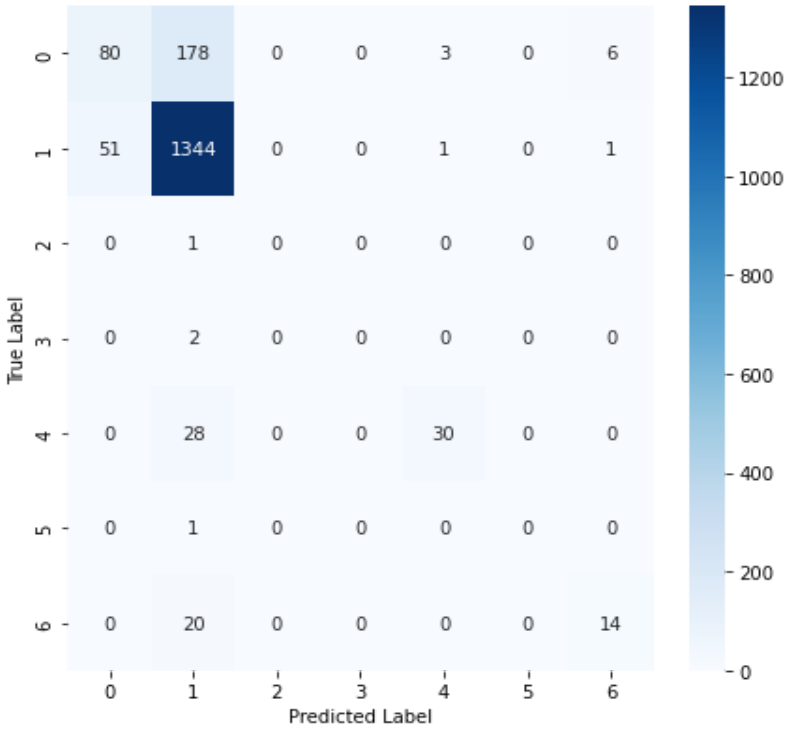

Figura 13. Matriz de confusión para los terceros dígitos de la categoría 60 - Decision tree

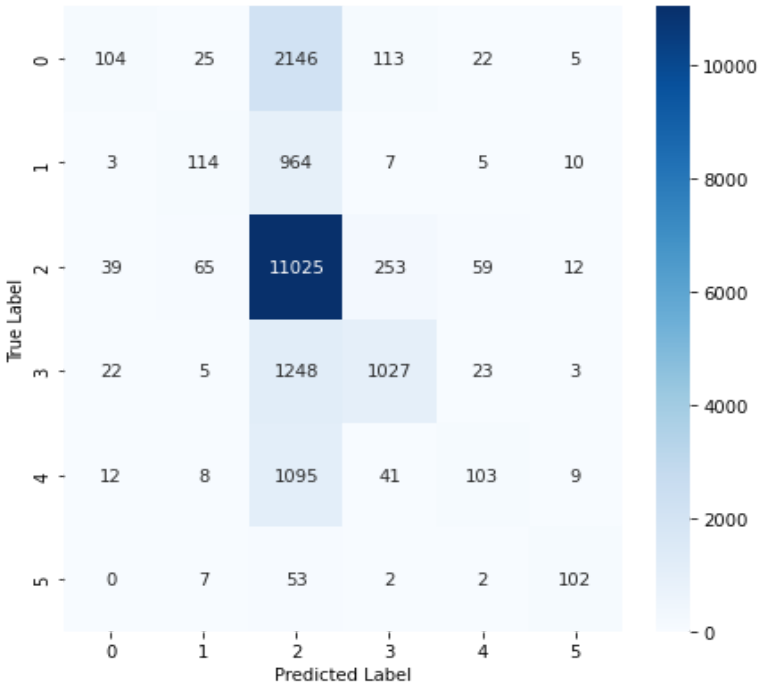

Figura 14. Matriz de confusión para los terceros dígitos de la categoría 45 - Decision tree

Una vez finalizadas todas las fases del procedimiento, la tabla 4 ofrece un contraste detallado de los rendimientos alcanzados al capacitar el modelo utilizando el código principal 60 y sus respectivas subdivisiones de tercer dígito.

Tercer dígito	Precision	Recall	F1-Score	Support
0	0.61	0.30	0.40	267
1	0.85	0.96	0.90	1397
2	0.00	0.00	0.00	1
3	0.00	0.00	0.00	2
4	0.88	0.52	0.65	58
5	0.00	0.00	0.00	1
6	0.67	0.41	0.51	34

Tabla 4. Reporte de clasificación del código 60 y sus terceros dígitos - Decision tree

Como se observa, las subdivisiones del tercer dígito tales como 2, 3 y 5 manifiestan un 0% en todas las métricas evaluativas, al igual que en el modelo previo, debido a la escasa presencia de estas categorías en el conjunto de datos, contando solo con una o dos instancias cada una. Respecto al rendimiento del modelo sobre este conjunto de códigos, se logra una precisión (*accuracy*) del **83%** y el tiempo requerido para el entrenamiento se establece en alrededor de un minuto.

Ahora en la tabla 5 proporciona una comparativa de los resultados obtenidos al entrenar el modelo con el código 45 y sus terceros dígitos.

Tercer dígito	Precision	Recall	F1-Score	Support
0	0.58	0.04	0.08	2415
1	0.51	0.10	0.17	1103
2	0.67	0.96	0.79	11453
3	0.71	0.44	0.54	2328
4	0.48	0.08	0.14	1268
5	0.72	0.61	0.66	166

Tabla 5. Reporte de clasificación del código 45 y sus terceros dígitos - Decision tree

Los resultados obtenidos a partir de este modelo exhiben ciertas similitudes con los producidos por el modelo de random forest. En términos de rendimiento para este código específico, el modelo alcanza una precisión (*accuracy*) del **66%**, con un periodo de entrenamiento que se extiende hasta los 7 minutos. Estas similitudes pueden ser atribuidas al hecho de que tanto el Decision tree como el RandomForest comparten la misma base: ambos son algoritmos de aprendizaje automático basados en árboles de decisión. En otras palabras, los dos modelos implementan la misma lógica de partición de datos y formación de reglas de decisión, aunque en grados de complejidad distintos.

3.4.1.3 MultinominalNB

En este modelo se presenta una explicación detallada de todo el proceso, aunque algunos parámetros utilizados son parecidos al clasificador de árboles de decisión previamente explicado:

- Primero se inicia el clasificador.
- A continuación, se define un diccionario con los hiperparámetros para el clasificador que se desea optimizar mediante RandomizedSearchCV. Estos hiperparámetros incluyen:
    - **'clf__alpha':** Es el parámetro de suavizado aditivo (Laplace/Lidstone). Este parámetro ayuda a manejar las características que no aparecen en los datos de entrenamiento para evitar multiplicaciones por cero en el cálculo de la probabilidad.
    - **'clf__fit_prior':** Es un parámetro booleano que decide si se deben aprender las probabilidades previas (o a priori) de las clases o no. Si se establece en falso, se utiliza una distribución uniforme a priori.
- Tras la inicialización de RandomizedSearchCV, se ejecuta un proceso similar al realizado con el algoritmo previo.
- Por último, se ajusta la búsqueda aleatoria a los datos de entrenamiento, y luego se utilizan los parámetros óptimos para predecir los datos de prueba.
- Una vez que el modelo ha sido entrenado y la mejor combinación de parámetros ha sido seleccionada, se usa el modelo para hacer predicciones en los datos de prueba utilizando el método predict().

Este procedimiento se puede observar en la siguiente sección de código:

```
clf = MultinomialNB()

pipeline = Pipeline([

    ('clf', clf)

])

param_dist = {

    'clf__alpha': np.linspace(0.5, 1.5, 6),

    'clf__fit_prior': [True, False],

}

random_search = RandomizedSearchCV(

    pipeline,

    param_distributions=param_dist,

    n_iter=100,

    cv=5,

    verbose=1,

    random_state=42,
```

```
    n_jobs=-1
)

random_search.fit(X_train_tfidf, y_train.values.ravel())

y_pred = random_search.predict(X_test_tfidf)

print(classification_report(y_test, y_pred))
```

Las figuras 15 y 16 exhiben las matrices de confusión para los códigos CPV de tres dígitos que empiezan con 60 y 45, respectivamente, obtenidas a partir del uso del modelo MultinomialNB. En la figura 15, se constata la repercusión de la escasez de observaciones para ciertas subcategorías en la categoría 60, lo cual provoca una reducida precisión de clasificación para las clases 2, 3 y 5, similar a los resultados del clasificador de árbol de decisión. Contrastantemente, la figura 16 destaca un escenario más prometedor para la categoría 45, demostrando un destacado desempeño de clasificación para las sub categorías 0, 2 y 3. Cabe destacar que, a pesar de la diferencia en los algoritmos de modelado, los resultados obtenidos presentan una variación mínima en comparación con los modelos anteriores, lo que sugiere una consistencia en la interpretación de los datos.

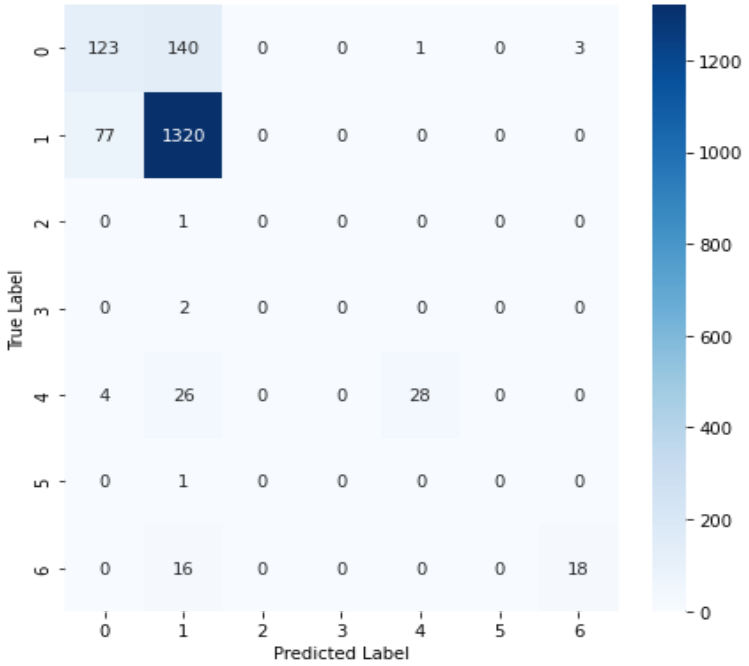

Figura 15. Matriz de confusión para los terceros dígitos de la categoría 60 - MultinominalNB

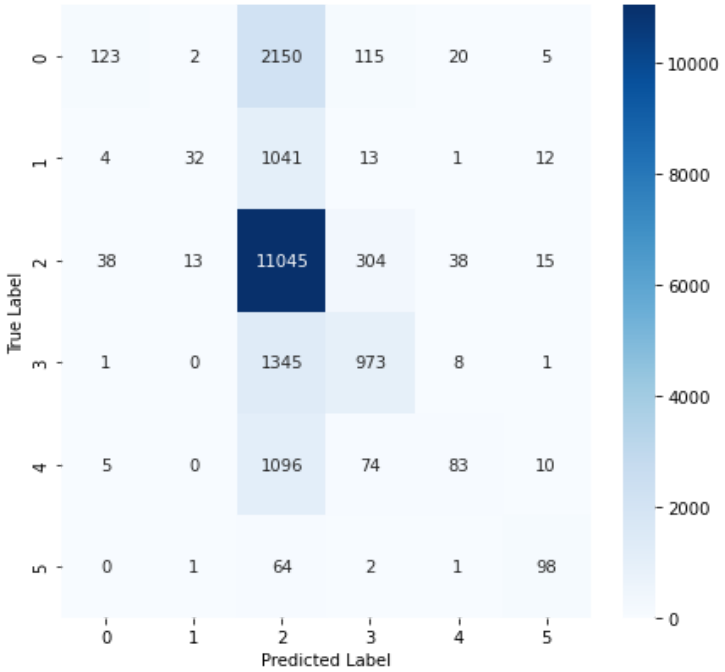

Figura 16. Matriz de confusión para los terceros dígitos de la categoría 45 - MultinominalNB

Con la conclusión de todas las etapas del método implementado, la tabla 6 proporciona una comparación exhaustiva de las métricas obtenidas al entrenar el nuevo modelo, en este caso el MultinomialNB, con el código 60 y sus correspondientes terceros dígitos.

Tercer dígito	Precision	Recall	F1-Score	Support
0	0.60	0.46	0.52	267
1	0.88	0.94	0.91	1397
2	0.00	0.00	0.00	1
3	0.00	0.00	0.00	2
4	0.97	0.48	0.64	58
5	0.00	0.00	0.00	1
6	0.86	0.53	0.65	34

Tabla 6. Reporte de clasificación del código 60 y sus terceros dígitos - MultinominalNB

En el análisis de los resultados, las categorías correspondientes a los terceros dígitos como 2, 3 y 5 presentan un porcentaje de 0% en todas las métricas de evaluación, al igual que en los modelos anteriores. Esto se atribuye a la limitada representación de estas categorías en el conjunto de datos, con solo una o dos instancias para cada una. El tercer dígito 1 es el que destaca por obtener el mejor recall y f1-score. En cuanto al desempeño general del modelo MultinomialNB aplicado a estos códigos, se consigue un nivel de precisión del

84% y el proceso de entrenamiento se completa en un tiempo aproximado de cinco segundos.

Ahora en la tabla 7 proporciona una comparativa de los resultados obtenidos al entrenar el modelo con el dígito 45 y sus terceros dígitos.

Tercer dígito	Precision	Recall	F1-Score	Support
0	0.72	0.05	0.10	2415
1	0.67	0.03	0.06	1103
2	0.66	0.96	0.78	11453
3	0.66	0.42	0.51	2328
4	0.55	0.07	0.12	1268
5	0.70	0.59	0.64	166

Tabla 7. Reporte de clasificación del código 45 y sus terceros dígitos - MultinominalNB

Los resultados arrojados por el modelo MultinomialNB presentan un notable paralelismo con aquellos obtenidos a partir de las implementaciones de los clasificadores random forest y decision tree. En este caso particular, el modelo MultinomialNB logra un *accuracy* del **65%** con un tiempo de entrenamiento de tan solo 12 segundos. Esta coincidencia en los resultados se puede atribuir a la esencia de los algoritmos de aprendizaje automático utilizados: tanto random forest como decision tree se fundamentan en la misma estructura de árboles de decisión, aunque con niveles de complejidad diferentes. Es decir, ambos modelos siguen una lógica similar de división de datos y formación de reglas de decisión. Por lo tanto, para este conjunto de datos específico, es plausible que el incremento en complejidad introducido por random forest no traduzca en una mejora significativa en las métricas de rendimiento en comparación con decision tree, lo que se refleja en desempeños comparables. De manera similar, MultinomialNB, aunque de naturaleza distinta a los modelos basados en árboles, puede proporcionar un rendimiento análogo, reflejando que, en este contexto, los tres modelos ofrecen soluciones viables y eficientes.

3.4.1.4 Gradient boosting

Continuaremos con el mismo protocolo de modelado, pero esta vez implementaremos el algoritmo gradient boosting. Los procedimientos llevados a cabo son los siguientes:

- Primero, se inicializa el clasificador como todos los procedimientos de los modelos anteriores.
- Se crea un pipeline.
- Posteriormente, se define el espacio de búsqueda para los hiperparámetros del modelo. Estos incluyen:
    - **clf__n_estimators**: [100, 200, 300, 400, 500]: Este parámetro se refiere al número de etapas de aumento que se realizan. En general, a más etapas de aumento, mejor es la capacidad del

modelo para aprender los datos, pero también puede dar lugar a un sobreajuste si el número es demasiado alto. Aquí, se ha escogido una gama de valores relativamente grande para explorar la efectividad de un número diverso de etapas de aumento.
    - **clf__learning_rate:** [0.01, 0.1, 0.2, 0.3, 0.4, 0.5]: La tasa de aprendizaje controla cuánto contribuye cada árbol a la predicción final. Una tasa de aprendizaje más baja significa que el modelo aprende más lentamente y puede requerir más árboles para obtener un buen rendimiento. Aquí, se elige una variedad de tasas de aprendizaje para investigar cuál funciona mejor.
    - **clf__max_depth:** np.arange(1, 10): Este parámetro controla la profundidad máxima de cada árbol individual. Cuanto mayor es la profundidad, más interacciones entre las características puede modelar el árbol. Sin embargo, una profundidad demasiado alta puede llevar a un sobreajuste. Se ha elegido un rango de 1 a 10 para explorar cómo influye la profundidad del árbol en el rendimiento del modelo.
    - **clf__min_samples_split:** np.arange(2, 10) y **'clf__min_samples_leaf':** np.arange(1, 10): Estos dos parámetros controlan el tamaño de las muestras requeridas para dividir un nodo interno y para ser un nodo hoja, respectivamente. Al ajustar estos valores, se puede controlar cuánto crecen los árboles y prevenir el sobreajuste. Se eligen rangos pequeños porque, en general, valores más altos de estos parámetros pueden limitar demasiado el crecimiento del árbol y, por lo tanto, reducir la capacidad del modelo para aprender los datos.
- Después de poner en marcha RandomizedSearchCV, se implementa un procedimiento que sigue las mismas pautas que las de los algoritmos anteriormente probados.
- Finalmente, se ajusta el modelo al conjunto de datos de entrenamiento y se hace una predicción en los datos de prueba.

Este procedimiento se puede observar en la siguiente sección de código:

```
clf = GradientBoostingClassifier(random_state=42)

pipeline = Pipeline([

    ('clf', clf)

])

param_dist = {

    'clf__n_estimators': [100, 200, 300, 400, 500],

    'clf__learning_rate': [0.01, 0.1, 0.2, 0.3, 0.4, 0.5],

    'clf__max_depth': np.arange(1, 10),

    'clf__min_samples_split': np.arange(2, 10),

    'clf__min_samples_leaf': np.arange(1, 10),

}

random_search = RandomizedSearchCV(
```

```
    pipeline,
    param_distributions=param_dist,
    n_iter=100,
    cv=5,
    verbose=1,
    random_state=42,
    n_jobs=-1
)
random_search.fit(X_train_tfidf, y_train.values.ravel())
y_pred = random_search.predict(X_test_tfidf)
print(classification_report(y_test, y_pred))
```

En la figura 17, se evidencia cómo la escasez de observaciones para ciertas subcategorías en el dominio 60 genera una precisión relativamente moderada en las clases 2, 3 y 5, cuando se utiliza el modelo gradient boosting. Es esencial subrayar que algunas descripciones de licitaciones pueden es Por otro lado, la figura 18 muestra un panorama más optimista para la categoría 45, con un notable rendimiento en la clasificación.

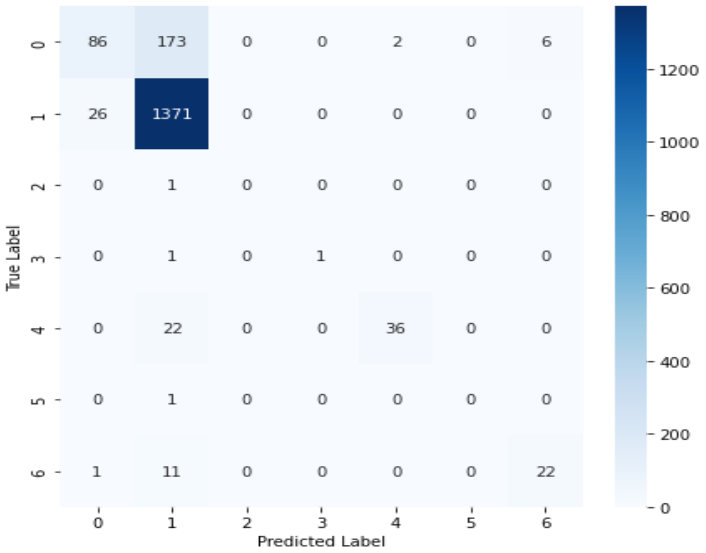

Figura 17. Matriz de confusión para los terceros dígitos de la categoría 60 – Gradient boosting

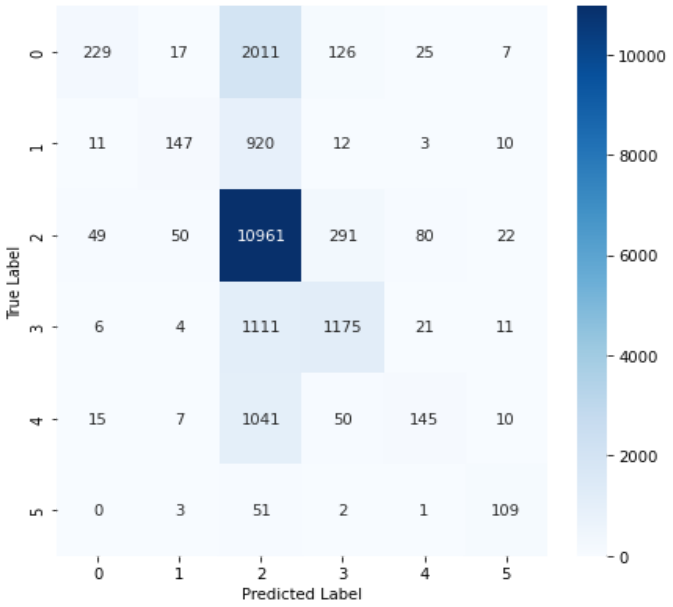

Figura 18. Matriz de confusión para los terceros dígitos de la categoría 45 – Gradient boosting

Una vez concluidas todas las fases de la estrategia empleada, la tabla 8 brinda un análisis comparativo minucioso de los resultados logrados al adiestrar el modelo.

Tercer dígito	Precision	Recall	F1-Score	Support
0	0.76	0.32	0.45	267
1	0.87	0.98	0.92	1397
2	0.00	0.00	0.00	1
3	1.00	0.50	0.67	2
4	0.95	0.62	0.75	58
5	0.00	0.00	0.00	1
6	0.79	0.65	0.71	34

Tabla 8. Reporte de clasificación del código 60 y sus terceros dígitos – Gradient boosting

En la revisión de los hallazgos, se observa que las subcategorías asociadas a los terceros dígitos, tales como 2 y 5, exhiben una tasa de 0% en todas las medidas de evaluación, en línea con lo observado en los modelos previos. Esta circunstancia surge a raíz de la escasa incidencia de estas subcategorías en el conjunto de datos, al contar únicamente con una o dos apariciones de cada una. Resulta relevante señalar que, a pesar de que la subcategoría correspondiente al tercer dígito 3 muestra la precisión más elevada, cabe destacar que solo cuenta con 2 observaciones para dicha clase. Sin embargo, es el tercer dígito 1 el que se distingue al lograr el recall y f1-score más altos. Al tratar estos códigos, se alcanza un grado de precisión (*accuracy*) del **85%**, y el lapso necesario para la capacitación del modelo se aproxima a 1 hora.

Ahora en la tabla 9 nos proporciona una comparativa de los resultados obtenidos al entrenar el modelo con el dígito 45 y sus terceros dígitos.

Tercer dígito	Precision	Recall	F1-Score	Support
0	0.74	0.09	0.17	2415
1	0.64	0.13	0.22	1103
2	0.68	0.96	0.80	11453
3	0.71	0.50	0.59	2328
4	0.53	0.11	0.19	1268
5	0.64	0.66	0.65	166

Tabla 9. Reporte de clasificación del código 45 y sus terceros dígitos - Gradient boosting

Los resultados generados por la implementación del modelo exhiben cierta similitud con los obtenidos de las implementaciones de los modelos anteriormente comentados. Sin embargo, estas semejanzas no deben enmascarar las diferencias inherentes de estos modelos y su impacto en el rendimiento. Para este conjunto de datos en particular, el modelo alcanza una exactitud o *accuracy* del 68%. Este resultado es, por cierto, impresionante, pero debe matizarse con el hecho de que este modelo requiere un tiempo de entrenamiento significativamente más largo, alcanzando las 5 horas. Al analizar las métricas de precisión, se puede notar que la mayoría de las clases superan el umbral del 60%. Sin embargo, un punto de interés es la excepción de la clase 4. Este puede ser un indicativo de que esta clase en particular puede ser más difícil de predecir o que las características presentes en los datos no son suficientes para distinguir adecuadamente esta clase.

3.4.1.5 Support vector machine

Se persiste con el enfoque de modelado constante, pero en esta ocasión se despliega el algoritmo de máquinas de vectores de soporte. Los pasos realizados en este modelo son los siguiente:

- Primero se inicializa el clasificador
- Se crea un objeto de pipeline.
- Luego se definen las distribuciones de los hiperparámetros para la búsqueda aleatoria. En este caso, se están variando los siguientes hiperparámetros:
    - **C:** Es el parámetro de regularización en soporte vector machine. Controla la compensación entre conseguir el mayor margen y minimizar las violaciones de margen (errores).
    - **gamma:** Es el coeficiente de kernel para 'rbf', 'poly' y 'sigmoid'. Puede ser 'scale', 'auto' o un número real. En caso de 'scale', se calcula como 1 / (n_características * varianza de X) y para 'auto' como 1/n_características.
    - **kernel:** Es el tipo de hiperplano utilizado para separar los datos. Puede ser 'linear', 'rbf' (Radial basis function), 'poly' (polinomial) o 'sigmoid'.

    - **degree:** Es el grado del polinomio para el kernel 'poly'. Es ignorado por todos los otros kernels.
- Se implementa el RandomizedSearchCV.
- Por último se realiza el entrenamiento del modelo y se realiza las predicciones

Este procedimiento se puede observar en la siguiente sección de código:

```
clf = SVC(random_state=42)

pipeline = Pipeline([

   ('clf', clf)

])

param_dist = {

   'clf__C': [0.1, 1, 10, 100],

   'clf__gamma': ['scale', 'auto'],

   'clf__kernel': ['linear', 'rbf', 'poly', 'sigmoid'],

     'clf__degree': [2, 3, 4]   # Ignorado por todos los kernels
excepto 'poly'

}

random_search = RandomizedSearchCV(

   pipeline,

   param_distributions=param_dist,

   n_iter=10,

   cv=3,

   verbose=1,

   random_state=42,

   n_jobs=-1

)

random_search.fit(X_train_tfidf, y_train.values.ravel())

y_pred = random_search.predict(X_test_tfidf)

print(classification_report(y_test, y_pred))
```

En la figura 19, se percibe de manera evidente cómo la limitada cantidad de observaciones correspondientes a ciertas subclases en el dominio 60 conduce a una precisión moderada en las clases 2, 3 y 5 cuando se aplica el modelo. Por su parte, la figura 20 presenta un escenario más prometedor para la categoría 45, donde se observa una clasificación efectiva y precisa. En general, los resultados obtenidos presentan paralelismos con los modelos previamente

utilizados, pero es necesario tener en cuenta estas peculiaridades al elegir el modelo más apropiado para tareas de clasificación en este contexto.

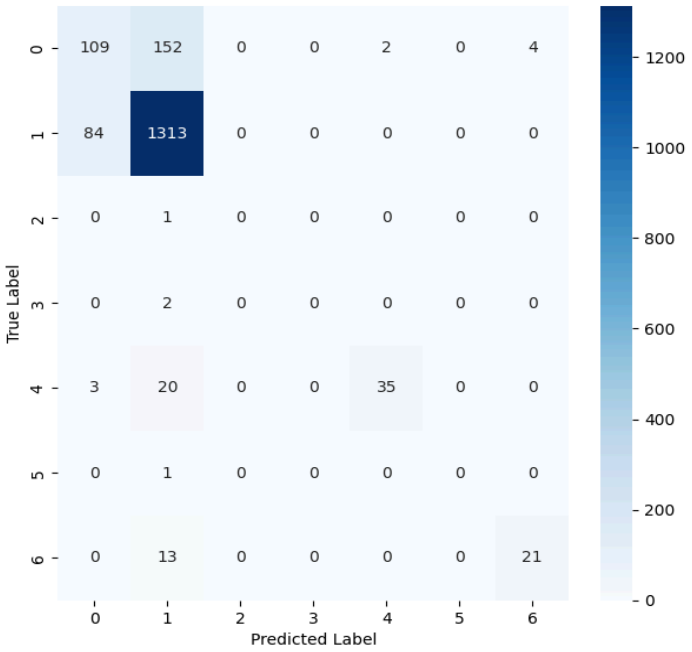

Figura 19. Matriz de confusión para los terceros dígitos de la categoría 60 - Support vector machine

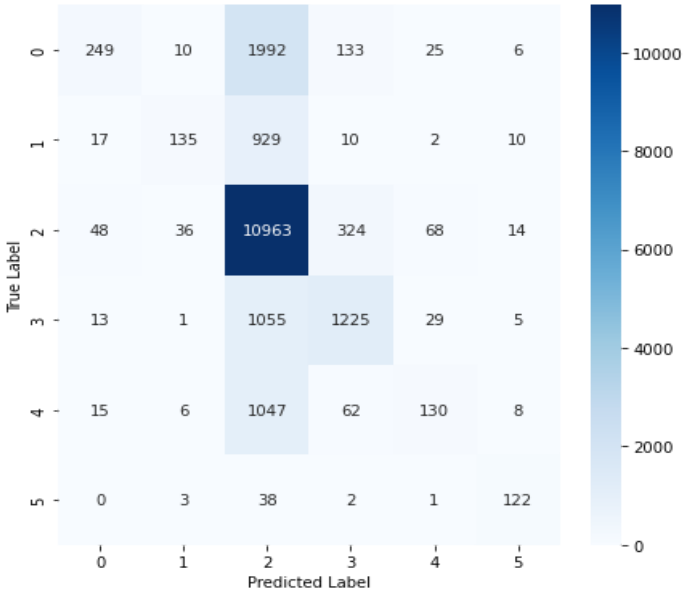

Figura 20. Matriz de confusión para los terceros dígitos de la categoría 45 - Support vector machine

Una vez finalizadas todas las etapas del procedimiento implementado, la tabla 10 proporciona una comparación detallada de las métricas logradas al entrenar modelo, con el código 60 y las respectivas categorías asociadas a su tercer dígito.

Tercer dígito	Precision	Recall	F1-Score	Support
0	0.56	0.41	0.47	267
1	0.87	0.94	0.91	1397
2	0.00	0.00	0.00	1
3	0.00	0.00	0.00	2
4	0.95	0.60	0.74	58
5	0.00	0.00	0.00	1
6	0.84	0.62	0.71	34

Tabla 10. Reporte de clasificación del código 60 y sus terceros dígitos - Support vector machine

Al analizar los resultados obtenidos a través del modelo, se muestra que ciertas subcategorías correspondientes a los terceros dígitos, tales como 2, 3 y 5, muestran un índice de 0% en todas las métricas evaluativas, una tendencia que se ha mantenido constante en los modelos anteriores. Este fenómeno puede ser atribuido a la baja representación de estas subcategorías en el conjunto de datos que estamos manejando, ya que cada una cuenta únicamente con una o dos instancias. Es interesante resaltar que, en esta ocasión, el tercer dígito 0 muestra una precisión menor en comparación con los resultados obtenidos en los otros modelos. Esto puede sugerir que el modelo, a pesar de su capacidad para manejar espacios de alta dimensión, puede tener dificultades para clasificar correctamente las instancias de esta subcategoría. En términos generales, este modelo exhibe un nivel de precisión (*accuracy*) del 83%. Este rendimiento, si bien es elevado, debe considerarse en el contexto del tiempo de entrenamiento requerido, que en este caso fue de 5 minutos. Este tiempo es relativamente corto, lo que indica que el modelo puede ser una opción eficiente y efectiva para este tipo de tareas de clasificación.

Ahora en la tabla 11 proporciona una comparativa de los resultados obtenidos al entrenar el modelo con el código 45 y sus terceros dígitos.

Tercer dígito	Precision	Recall	F1-Score	Support
0	0.73	0.10	0.18	2415
1	0.71	0.12	0.21	1103
2	0.68	0.96	0.80	11453
3	0.70	0.53	0.60	2328
4	0.51	0.10	0.17	1268
5	0.74	0.73	0.74	166

Tabla 11. Reporte de clasificación del código 45 y sus terceros dígitos - Support vector machine

El análisis de los resultados del modelo pone de manifiesto ciertas correspondencias con los datos obtenidos a través de los modelos previamente implementados. No obstante, es importante no pasar por alto las diferencias inherentes entre estos modelos, y cómo estas peculiaridades pueden impactar en el desempeño de cada uno. En este escenario específico, el modelo ha conseguido una precisión del 68%. Si bien esta cifra es notable, es necesario tener en cuenta que el modelo requiere de un tiempo de entrenamiento considerablemente extenso, llegando a sumar hasta 6 horas. Al desglosar las métricas de precisión, observamos que la mayoría de las clases superan la marca del 60%. No obstante, la clase 4 se muestra como una excepción a esta tendencia, replicando el patrón observado en el modelo anterior. Esta circunstancia puede sugerir que la clase 4 podría presentar una mayor complejidad a la hora de su predicción, o que las características disponibles en el conjunto de datos no proveen suficiente información para su adecuada distinción. En resumen, aunque el modelo demostró ser competente en general, las particularidades de algunas subcategorías y la exigencia de un tiempo de entrenamiento más prolongado invitan a considerar estos factores al elegir el modelo más adecuado para este tipo de tareas de clasificación.

3.4.2 Comparación de los modelos

En las siguientes tablas se observa la comparación de los modelos probados con su f1-score macro avg (este valor es la media no ponderada del F1-score de cada clase) esta métrica es útil cuando las clases son igualmente importantes, ya que cada una contribuye de igual manera al promedio final y f1-score weighted avg (este valor es la media ponderada del F1-score de cada clase) esta métrica es útil cuando hay un desbalance entre las clases, ya que da más importancia a las clases más frecuentes y tiempo de entrenamiento utilizado tanto para la clase 45 y 60.

Modelo	F1-Score Macro Avg	F1-Score Weighted Avg	Tiempo de Entrenamiento
Random Forest	0.40	0.83	1 min
Decision Tree	0.35	0.81	1 min
MultinominalNB	0.39	0.83	2 seg
Gradient Boosting	0.50	0.84	1 hr
Support Vector Machine	0.40	0.83	5 min

Tabla 12. Comparación de modelos para la clase 60

Modelo	F1-Score - Macro Avg	F1-Score Weighted Avg	Tiempo de Entrenamiento
Random Forest	0.42	0.60	25 min
Decision Tree	0.40	0.59	7 min
MultinominalNB	0.37	0.57	12 seg
Gradient Boosting	0.44	0.61	5 hrs
Support Vector Machine	0.45	0.62	6 hrs

Tabla 13. Comparación de modelos para la clase 45

A través de las tablas de comparación presentadas, se evalúa de forma directa y transparente el desempeño de cada uno de los modelos analizados, específicamente para las clases 45 y 60. Es notable que las métricas de f1-score obtenidas en ambos casos muestran cierto paralelismo, indicando un nivel de desempeño consistente y robusto a través de diferentes conjuntos de datos. No obstante, es importante destacar que, aunque las tasas de exactitud puedan parecer similares a primera vista, existen sutilezas y variaciones en el desempeño que se reflejan en las métricas secundarias y en el tiempo de entrenamiento de cada modelo. Estos aspectos son vitales para seleccionar el modelo más adecuado, ya que proporcionan información sobre la eficiencia y la generalización del modelo, es decir, su capacidad para producir resultados consistentes frente a nuevos datos.

En resumen, estos resultados de comparación permiten entender las fortalezas y debilidades de cada modelo y proporcionan una base sólida para la elección del modelo a implementar en el sistema de clasificación de códigos CPV. A pesar de las similitudes en la exactitud, es esencial considerar todas las dimensiones de rendimiento al tomar esta decisión.

Una vez realizado todo el proceso de entrenamiento, ya se puede realizar la predicción del tercer dígito, que se observa en el siguiente código:

```
descripcion_nueva = "Contratación de Conducción Extraordinaria de
Transporte del Correo en La Zona 4, Madrid Uses Apoyo a Centro y
Unidades (6 lotes)."

descripcion_vectorizada = vectorizer.transform([descripcion_nueva])

third_digit_pred = random_search.predict(descripcion_vectorizada)

print(f"El tercer dígito predicho es:
{twodig}{third_digit_pred[0]}")
```

A continuación, se explica esa sección del código:

- **descripcion_nueva = "Contratación de Conducción Extraordinaria de Transporte del Correo en La Zona 4, Madrid Uses Apoyo a Centro y Unidades (6 lotes).":** Esta línea está estableciendo la descripción de un nuevo contrato para predecir el tercer dígito del código CPV.

- **descripcion_vectorizada = vectorizer.transform([descripcion_nueva]):** Aquí, se utiliza el vectorizer (que debe haber sido previamente ajustado con el método fit a los datos de entrenamiento) para transformar la descripción del contrato en una forma que el modelo puede utilizar para hacer una predicción. En este caso, la descripción es transformada en un vector de características.
- **third_digit_pred = random_search.predict(descripcion_vectorizada):** Esta línea utiliza el modelo (en este caso, random_search, que parece ser un modelo de búsqueda aleatorizada de hiperparámetros previamente ajustado) para hacer una predicción basada en la descripción vectorizada del contrato. El resultado es la predicción del tercer dígito del código CPV.
- **print(f"El tercer dígito predicho es: {twodig}{third_digit_pred[0]}"):** Finalmente, imprime la predicción realizada por el modelo. En esta línea, {twodig} se refiere a los primeros dos dígitos del código CPV y {third_digit_pred[0]} es la predicción realizada por el modelo del tercer dígito.

En resumen, este fragmento de código toma una descripción de contrato, la transforma en una representación numérica que el modelo puede entender, y luego utiliza el modelo para predecir el tercer dígito del código CPV basado en esa descripción. El procedimiento ilustrado a lo largo de este documento, inicialmente enfocado en las clases 45 y 60, se amplía y se aplica de igual manera al resto de las 45 clases. Este enfoque integral se centra en predecir el tercer dígito del código CPV. Para llevar a cabo esta tarea, se elige el algoritmo Random Forest, decisión respaldada por su excepcional rendimiento, eficiencia en el uso de recursos computacionales y rapidez en términos de tiempo de entrenamiento. El proceso de entrenamiento del modelo se completó en un tiempo de 3 horas y 30 minutos. Cabe destacar que el código, que comprende desde la etapa de preparación de datos hasta la implementación y el entrenamiento del modelo para las 45 clases con miras a predecir el tercer dígito, se encuentra en la sección de anexos de este proyecto, disponible en el sexto capítulo y así mismo se encuentra disponible online en el siguiente repositorio: **https://github.com/erick4556/CPV_Classification**.

4 Resultados y conclusiones

A lo largo de este proyecto, se han explorado diversas técnicas de procesamiento del lenguaje natural y modelos de aprendizaje automático con el propósito de predecir el tercer dígito de los códigos CPV basándose en las descripciones de los contratos. Después de probar con varios modelos y evaluar su rendimiento, se ha seleccionado el algoritmo random forest como el modelo final debido a su alta precisión y eficiencia. El algoritmo random forest demostró ser una opción superior debido a su excelente equilibrio entre precisión y eficiencia. A pesar de que otros modelos mostraron resultados prometedores, la combinación de un tiempo de entrenamiento moderado y una precisión sobresaliente hizo que el random forest sobresaliera entre los demás. Se presentaron varios desafíos, principalmente la cantidad significativa de tiempo requerida para el entrenamiento de los modelos y el consumo de recursos computacionales. A pesar de estas dificultades, se ha logrado construir un modelo robusto y preciso para predecir el tercer dígito de los códigos CPV a partir de las descripciones de los contratos.

En cuanto a futuras líneas de trabajo, este proyecto puede ser la base para predecir los siguientes dígitos de los códigos CPV. Esto podría implicar el manejo de un mayor nivel de complejidad en la clasificación, debido a la gran cantidad de posibles códigos y la falta de observaciones para ciertas categorías. Adicionalmente, puede ser beneficioso explorar técnicas y modelos que puedan disminuir aún más el tiempo de entrenamiento sin sacrificar la precisión del modelo. Esta tarea probablemente implica la inversión en mejoras de hardware para manejar el entrenamiento de modelos más complejos o el uso de técnicas de optimización de software más avanzadas.

Además de lo que ya se ha mencionado, la importancia de la selección de características y la ingeniería de las mismas en el desempeño de los modelos. Como se ha visto a lo largo de este proyecto, la forma en que se representan los datos de entrada puede tener un impacto significativo en la capacidad del modelo para aprender patrones efectivos.

Por último, este trabajo ha demostrado la viabilidad de aplicar técnicas de aprendizaje automático para automatizar y mejorar la precisión de la clasificación de códigos CPV. Sin embargo, cabe subrayar que el éxito de estos modelos depende de la calidad y cantidad de los datos disponibles. Por lo tanto, los esfuerzos para recopilar y curar conjuntos de datos más grandes y representativos seguirán siendo de vital importancia para futuras investigaciones en este campo.

5 Bibliografía

[1] "Información sobre licitaciones" Cogiti. [Online]. Disponible en: https://cogiti.es/licitaciones/informacion.

[2] "Códigos CPV en la contratación pública" Gobierto. [Online]. Disponible en: https://www.gobierto.es/blog/cpv.

[3] "Contratos públicos" European Union. [Online]. Disponible en: https://european-union.europa.eu/live-work-study/public-contracts_es.

[4] S. Arrowsmith, The Law of Public and Utilities Procurement: Regulation in the EU and the UK, vol. 1, Sweet & Maxwell, 2014.

[5] J. Devlin, M. W. Chang, K. Lee, and K. Toutanova" BERT: Pre-training of deep bidirectional transformers for language understanding," 2018.

[6] K. S. Jones, S. Walker, and S. E. Robertson"A probabilistic model of information retrieval: development and comparative experiments," Information Processing & Management, 2016.

[7] A. Ramírez Sixtos, "Clasificador de códigos de licitaciones públicas: caso de estudio España y México," M.S. thesis, Dept. Inteligencia Artificial, E.T.S. de Ingenieros Informáticos, Univ. Politécnica de Madrid, Madrid, Spain, 2022. [Online]. Disponible: https://oa.upm.es/71383/

[8] "CPV - Códigos CPV," SIMAP. [Online]. Disponible en: https://simap.ted.europa.eu/es/cpv#:~:text=Divisiones%2C%20identificadas%20por%20los%20dos,del%20c%C3%B3digo%20(XXXXX000%2DY)%3B.

[9] M. J. García Rodríguez, Las licitaciones públicas: análisis de datos y sistemas predictores utilizando métodos de machine learning, Universidad de Oviedo, 2022.

[10] Navas-Loro, M., Garijo, D., & Corcho, O. (2022). Multi-label Text Classification for Public Procurement in Spanish. Procesamiento Del Lenguaje Natural, 69, 73-82. http://journal.sepln.org/sepln/ojs/ojs/index.php/pln/article/view/6429.

[11] A. Gutiérrez-Fandiño et al., "MarIA: Spanish Language Models," arXiv preprint, vol. arXiv:2107.07253, 2021.

[12] Y. Liu et al., "RoBERTa: A Robustly Optimized BERT Pretraining Approach," arXiv preprint, vol. arXiv:1907.11692, 2019.

[13] "Introduction to Natural Language Processing for Text" Towards Data Science. [Online]. Disponible en: https://towardsdatascience.com/introduction-to-natural-language-processing-for-text-df845750fb63.

[14] "TF-IDF: La relevancia en la búsqueda de información" USEO. [Online]. Disponible en: https://useo.es/tf-idf-relevancia/.

[15] J. D. Kelleher, B. Mac Namee, and A. D'Arcy, Fundamentals of Machine Learning for Predictive Data Analytics: Algorithms, Worked Examples, and Case Studies, MIT Press, 2015.

[16] J. Grus, Data Science from Scratch: First Principles with Python, O'Reilly Media, Inc., 2015.

[17] "Multinomial Naïve Bayes for Documents Classification and Natural Language Processing (NLP)" Towards Data Science. [Online]. Disponible en: https://towardsdatascience.com/multinomial-na%C3%AFve-bayes-for-documents-classification-and-natural-language-processing-nlp-e08cc848ce6.

[18] P. Fournier-Viger, J. C. Lin, R. U. Kiran, Y. S. Koh, and R. Thomas, "A Survey of Sequential Pattern Mining," pp. 54-77, 2017.

[19] X. Xue, H. Yang, and J. Zhang" Using Population-based Incremental Learning Algorithm for Matching Class Diagrams," vol. 3, no. 1, pp. 1-8, 2019.

[20] B. Myroniv, C. Wu, Y. Ren, A. B. Christian, E. Bajo, and Y. Tseng "Analyzing User Emotions via Physiology Signals," pp. 11-25, 2017.

[21] A. Géron, Aprende Machine Learning con Scikit-Learn, Keras y TensorFlow. 3ra ed. Madrid, España: Anaya Multimedia, 2023.

[22] S. JUGRAN, A. KUMAR, B. S. TYAGI and V. ANAND "Extractive Automatic Text Summarization using SpaCy in Python & NLP," 2021 International Conference on Advance Computing and Innovative Technologies in Engineering (ICACITE), Greater Noida, India, 2021, pp. 582-585, doi: 10.1109/ICACITE51222.2021.9404712.

[23] Deloitte "Study on up-take of emerging technologies in public procurement," Technical report, 2020.

6 Anexos

Prueba con todos los 2 primeros dígitos

Código para quedarnos sólo las filas y columnas cuyo código empieza por dos dígitos específico

In [ ]:

```
twodig = '38'
columns = df.columns.str.startswith(twodig) #Cogemos las columnas que empiezan por los dos primeros digitos que queremos
```

In [ ]:

```
columns[1]= True #También queremos la descripcion, asi que la primera columna tambien la ponemos a True
df_03 = df.loc[:,columns] #Nos quedamos con las columnas seleccionadas
df_03 = df_03[df_03[list(df_03.columns[1:].values)].any(axis='columns')] #Nos quedamos solo las filas que tengan algún 1
df_03.head(20)
```

Función para procesar las descripciones

```
# Cargar el modelo en español de spaCy
nlp = spacy.load('es_core_news_sm')
# Función para procesar las descripciones utilizando técnicas de NLP
def preprocess_text(text):
    # Lowercasing, procesar el texto y crear un objeto Doc
    doc = nlp(text.lower())

    # Noise removal and lemmatization
    words = [token.lemma_ for token in doc if not token.is_punct]

    # Stop word removal
    words = [word for word in words if word not in nlp.Defaults.stop_words]

    # Rejoin words
    text = ' '.join(words)

    return text

# Segunda función para procesar el texto usando el modelo en español spacy
def preprocess_text2(text):
    #Lowercasing
    doc = nlp(text.lower())
    #Tokenization, noise removal, eliminación de stop words y lemmatization
```

```
    lemmatized = [token.lemma_ for token in doc if not token.is_stop and
not token.is_punct]
    return ' '.join(lemmatized)
```

Entrenamiento del modelo

```
from collections import defaultdict
two_digits_list = ['03', '09', '14', '15', '16', '18', '19', '22', '24',
'30', '31', '32', '33', '34', '35', '37', '38', '39', '41', '42', '43',
'44', '45', '48', '50', '51', '55', '60', '63', '64', '65', '66', '70',
'71', '72', '73','75', '76', '77', '79', '80', '85', '90', '92', '98']
# Inicializar un diccionario para almacenar los resultados
results = {twodig: {} for twodig in two_digits_list}
path = "/ruta"
def train_and_evaluate_for_twodig(twodig):
    columns = df.columns.str.startswith(twodig)
    columns[1] = True #columns[1] = True
    df_twodig = df.loc[:, columns]
                                                    df_twodig                =
df_twodig[df_twodig[list(df_twodig.columns[1:].values)].any(axis='columns
')]

    X = []
    y = []

    for index, row in df_twodig.iterrows():
        descripcion = row['description']

        for col in df_twodig.columns[1:]:
            if row[col] == 1:
                two_digits = col[:2]
                third_digit = col[2]

                X.append((descripcion, two_digits))
                y.append(third_digit)

    X = pd.DataFrame(X, columns=['description', 'two_digits'])
    y = pd.DataFrame(y, columns=['third_digit'])

    X['description'] = X['description'].apply(preprocess_text)

        X_train,  X_test,  y_train,  y_test  =  train_test_split(X,  y,
test_size=0.2, random_state=42)

    vectorizer = TfidfVectorizer(max_features=1000)
    X_train_tfidf = vectorizer.fit_transform(X_train['description'])
    X_test_tfidf = vectorizer.transform(X_test['description'])
```

```
#---------Random Forest---------

  rf_model = RandomForestClassifier(random_state=42)

  pipeline = Pipeline([
      ('clf', rf_model)
  ])


  # Especifica los rangos de hiperparámetros
  param_dist = {
      "clf__n_estimators": [100, 200, 300, 400, 500],
      "clf__max_depth": [10, 20, 30, 40, 50, None],
      "clf__min_samples_split": [2, 5, 10, 15],
      "clf__min_samples_leaf": [1, 2, 5, 10],
  }

  # Inicializa RandomizedSearchCV
  random_search = RandomizedSearchCV(
      estimator=pipeline,
      param_distributions=param_dist,
      n_iter=10,
      cv=3,
      scoring="accuracy",
      random_state=42,
  )

  # Ajusta RandomizedSearchCV en los datos de entrenamiento
  random_search.fit(X_train_tfidf, y_train.values.ravel())

  # Obtén el mejor pipeline y sus hiperparámetros
  best_pipeline = random_search.best_estimator_
  best_params = random_search.best_params_

  y_pred = best_pipeline.predict(X_test_tfidf)

  accuracy = accuracy_score(y_test, y_pred)

  #Guardar el pipeline y el vectorizer
  dump(best_pipeline, f'{path}pipeline_{twodig}.joblib')
  dump(vectorizer, f'{path}vectorizer_{twodig}.joblib')

  return accuracy, pipeline, vectorizer
```

```
# Entrenar y evaluar un modelo para cada par de dígitos
for twodig in two_digits_list:
  accuracy, model, vectorizer = train_and_evaluate_for_twodig(twodig)
```

```
    results[twodig]['accuracy'] = accuracy
    results[twodig]['model'] = model
    results[twodig]['vectorizer'] = vectorizer
```

```
# Dada la descripción, predecir el mejor tercer dígito
def predict_best_third_digit(description, twodigits):
    # Cargar el modelo y el vectorizer
    model = load(f'{path}pipeline_{twodig}.joblib')
    vectorizer = load(f'{path}vectorizer_{twodig}.joblib')

    # Transformar la descripción con el vectorizer cargado
    descripcion_vectorizada = vectorizer.transform([description])

    # Predecir la probabilidad y la etiqueta
    third_digit_proba = model.predict_proba(descripcion_vectorizada)
    third_digit_pred = model.predict(descripcion_vectorizada)

    return third_digit_pred[0], np.max(third_digit_proba)
```

```
descripcion_nueva = "Quinto pedido derivado del Acuerdo Marco para el
suministro de equipamiento multimedia. (Exp. 066/20)."
twodig = '38'
best_prediction,                          best_proba                          =
predict_best_third_digit(descripcion_nueva,twodig)
print(f"La mejor predicción para la descripción dada es:")
print(f"Dos   dígitos:   {twodig},   tercer   dígito:   {best_prediction},
probabilidad: {best_proba}")
print(f"Concatenación de los 3 dígitos: {twodig}{best_prediction}")
```

```
La mejor predicción para la descripción dada es:
Dos dígitos: 38, tercer dígito: 6, probabilidad: 0.6402952822890903
Concatenación de los 3 dígitos: 386
```