Escuela Superior
de Ingeniería y Tecnología
Universidad de La Laguna

Trabajo de Fin de Grado

Grado en Ingeniería Informática

Categorización de información parlamentaria usando aprendizaje automático.

Categorization of parliamentary information using machine learning.

Raúl Martín Rigor

Dña. **María Elena Sánchez Nielsen** con DNI 42848599J profesora Titular de Universidad adscrita al Departamento de Ingeniería Informática y de Sistemas de la Universidad de La Laguna, como tutora.

C E R T I F I C A

Que la presente memoria titulada:

"Categorización de información parlamentaria usando aprendizaje automático"

ha sido realizada bajo su dirección por D. **Raúl Martín Rigor**,

con DNI 43841375-V.

Y para que así conste, en cumplimiento de la legislación vigente y a los efectos oportunos firman la presente en La Laguna a 7 de Julio de 2022

La Laguna, 7 de Julio de 2022

Agradecimientos

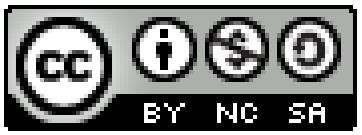
Quiero agradecer el apoyo de toda mi familia y seres queridos a la hora de realizar tanto este trabajo, como el grado completo y muchos otros proyectos que he desarrollado a lo largo de mi vida. Llegar a este punto no hubiera sido posible estando solo, muchas gracias a las personas con las que comparto mi vida y comparten su vida conmigo.

En el ámbito académico, he gozado siempre de buenos/as profesores/as tanto en la carrera como en mis estudios previos. Quería agradecer a todo el personal docente que hace su trabajo correctamente y despierta un interés en el alumnado.

A lo largo de la carrera, me he sentido muy cómodo con mi entorno y he disfrutado de la compañía de muy buenos compañeros. Quería agradecer en especial a un grupo de compañeros que se han convertido en verdaderos amigos: Acoidán Mesa Hernández, Adrián Hernández Suárez, Borja Guanche Sicilia, Sergio Pitti de Armas, Diego Rodriguez Pérez y Alberto Rios de la Rosa. Me han demostrado que podemos ser un equipo de trabajo altamente capaz. Conocemos nuestras virtudes y defectos. Y han resultado ser personas con las que puedo contar si me surge cualquier problema. Sin ellos, muy probablemente no hubiera sido capaz de seguir en la carrera y por ello les doy las gracias.

Finalmente quería agradecerme a mí mismo por ser la persona que más cree y confía en mí. Valoro el éxito y progreso que estoy logrando y me doy las gracias por mantenerme siempre constante en el camino correcto.

Licencia



Resumen

El objetivo principal de este trabajo es la categorización de iniciativas parlamentarias según su ámbito mediante métodos, técnicas y algoritmos de “machine-learning” o aprendizaje automático.

A través del desarrollo, entrenamiento y testeo de un modelo de aprendizaje, se logrará la estructuración y clasificación de la información parlamentaria disponible. Se obtendrá una herramienta capaz de categorizar con gran fiabilidad iniciativas actuales y futuras. En este trabajo también quedarán recogidos todos los estudios de los resultados obtenidos por este modelo y todas sus métricas asociadas con el fin de respaldar su efectividad.

Palabras clave: machine-learning, categorización, entrenamiento, testeo, modelo

Abstract

The main objective of this work is the categorization of parliamentary initiatives according to their scope by means of machine-learning methods, techniques and algorithms.

Through the development, training and testing of a learning model, the structuring and classification of the parliamentary information available will be achieved. We will obtain a tool capable of reliably categorizing current and future initiatives. This work will also include all the studies of the results obtained by this model and all its associated metrics in order to support its effectiveness.

Keywords: machine-learning, categorization, training, testing, model

Índice general

Índice de figuras

Índice de tablas

Capítulo 1 Introducción

1.1 Antecedentes

En la actualidad, el aumento exponencial de volúmenes de información ofrece nuevos desafíos para las organizaciones. En el caso de los Parlamentos Autonómicos, dado el volumen masivo de iniciativas parlamentarias, presentan diferentes desafíos, tales como poder estructurar y categorizar la información con el fin de poder realizar búsquedas sobre dicha información y llevar a cabo un análisis posterior. Las administraciones públicas se enfrentan continuamente a problemas como este, dónde se tiene que manejar una gran cantidad de información sin una organización determinada de los mismos. El tratamiento de datos es una problemática creciente que demanda una solución efectiva cuanto antes.

Actualmente, se están poniendo en práctica muchas tecnologías diferentes que pretenden automatizar estos procesos facilitando en gran medida la gestión de la información en administraciones públicas *[1]*. El machine learning ha arrojado luz sobre el problema, siendo una de las herramientas principales escogidas para formar parte de la solución. Se trata de una disciplina concreta de todas las existentes en la Inteligencia Artificial, encargada de fabricar modelos predictivos con determinada fiabilidad en base a un entrenamiento exhaustivo en reconocimiento de patrones sobre una gran cantidad de datos. Hoy en día ya disponemos de numerosos ejemplos donde se aplican soluciones basadas en aprendizaje automático para resolver problemas con resultados favorables a tener en cuenta:

En 2016, se realizó un estudio sobre Algoritmos de clustering y aprendizaje automático aplicados a Twitter por parte del alumno de la Universidad Politécnica de Cataluña, Eric-Joel Blanco-Hermida Sanz [2]. En este estudio, se pretende hacer uso de algoritmos de clustering para clasificar tweets sobre opiniones de la empresa con la que colaboraba, y un posterior análisis del sentimiento para identificar si la opinión dada era negativa o positiva.

1.2 Objetivos

Se pretende hacer uso de la tecnología más acertada posible para categorizar de forma efectiva todas las iniciativas actuales y futuras del Parlamento de Canarias. Para este propósito, se estudiará un conjunto de posibles soluciones y se realizarán diversas pruebas para considerar la viabilidad de la aplicación de cada una. Una vez elegida la opción definitiva, se pretende llevar a cabo el diseño, desarrollo e implementación de un sistema que cumpla con dichas expectativas. También se realizará un registro detallado de las fases del desarrollo, decisiones tomadas en el proceso y problemas planteados en la ejecución. Se desea recoger también las pruebas y estudios posteriores que confirmen la efectividad de la herramienta.

Capítulo 2 Estado del arte

Una de las tantas aplicaciones de la Inteligencia Artificial, es la clasificación de datos. Bajo esta premisa, existen distintas técnicas y métodos para realizar el mismo propósito. Dependiendo de la naturaleza del problema, convendrá realizar un proceso u otro.

En el caso concreto del machine learning, el aprendizaje del algoritmo viene determinado por el conocimiento de los datos a tratar. En función de si se conocen características comunes que relacionen los datos o no, se establecerá el criterio para elegir un tipo de algoritmo u otro.

2.1 Aprendizaje no supervisado

En el aprendizaje no supervisado, el modelo no recibe indicaciones sobre el grupo al que pertenecen los datos de la observación. El algoritmo se encargará por sí mismo de identificar patrones y comportamientos similares entre los datos que aparentemente no tienen relación con el fin de agruparlos en función de sus características. Este tipo de agrupaciones son denominadas *clústers*.

KMeans

Se trata de uno de los algoritmos más utilizados para realizar agrupaciones (clustering).

El funcionamiento del algoritmo de KMeans es bastante simple. Es necesario averiguar la cantidad óptima de clusters a realizar. Una vez obtenido este número, se le indica al algoritmo y este realizará iteraciones continuamente para determinar los puntos más diferenciados sobre los cuales se agruparán el resto de datos similares (a estos puntos se les denomina centroides). De esta manera, las observaciones quedarán divididas para cada cluster logrando una clasificación de manera automática.

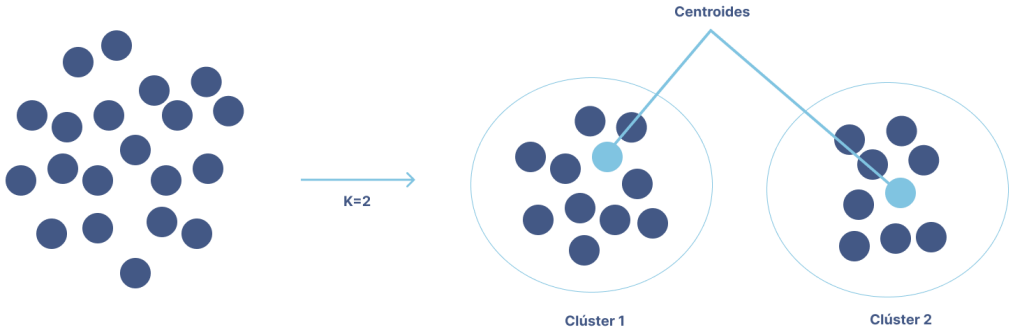


Figura 1: clustering

En este campo tomamos como referencia los mencionados estudios sobre clustering de Eric-Joel Blanco-Hermida Sanz [2].

2.2 Aprendizaje supervisado

En el aprendizaje supervisado, el modelo conoce de antemano las características por las que deben ser agrupados los datos. La observación viene precedida de un etiquetado correcto del dataset fruto del análisis, preparación y dotación de sentido de los mismos. La máquina es entrenada con estos datos para que aprenda la clasificación que corresponde a cada elemento según las características indicadas.

A la hora de poner en funcionamiento el algoritmo, se hablará de dos posibles casos: si lo que se desea es predecir un valor, se tratará de una regresión; en el caso contrario, se hablará de clasificación.

Regresión lineal

La regresión lineal se trata de una operación que trata de buscar una relación entre la variable explicada y la variable explicativa. Esta relación vendrá representada por una recta que tratará de acercarse lo máximo posible a la nube de puntos que representan los datos observados.

De esta manera, la regresión lineal puede ser utilizada para establecer una clasificación binaria en la que se determina si un dato es válido o no en función de su longitud a la recta.

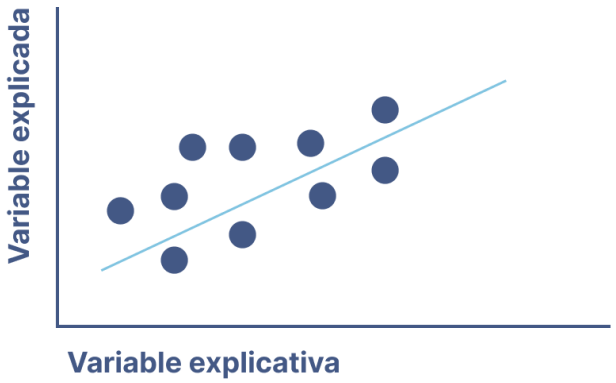


Figura 2: regresión lineal

Para entender sobre este método, se ha hecho uso una guía disponible en documentación de la Universidad Carlos III de Madrid[3].

Regresión logística

Este tipo de algoritmos son de gran utilidad cuando se trata de clasificar un conjunto de datos. Al igual que la regresión lineal, es un algoritmo que se utiliza para predecir el valor de una variable dependiente en función de las variables independientes. A diferencia de la regresión lineal, este arroja un resultado probabilístico por lo que resulta extremadamente útil para clasificar elementos que pueden pertenecer a distintas categorías en función de su relación con ellas. No se acoge tanto a una solución binaria que dictamine un caso concreto e invariable de “verdadero o falso” y, por lo tanto, reduce las probabilidades de equivocarse en un caso de clasificación multiclase.

La editorial de medicina Elsevier, publicó en 2020 un artículo sobre la “Utilidad del

aprendizaje automático en el desarrollo de una puntuación de predicción de mortalidad temprana en la gripe española [4]". Se utilizó un modelo de regresión logística entrenado con los datos de 3959 pacientes con una edad media de 55 años. El modelo presentó una precisión del 83%, demostrando así las capacidades del machine learning de predecir resultados con bastante fiabilidad que ayudan a la comunidad científica y, en este caso concreto, a complementar las técnicas desarrolladas en el ámbito de la salud.

También ha sido de gran utilidad la información encontrada respecto a este campo en la documentación de la Universidad Carlos III de Madrid [5].

2.3 Redes Neuronales

Las redes neuronales [6] son un tipo de aprendizaje automático basado en replicar la forma de aprender del cerebro humano. La carga de trabajo se distribuye en numerosas unidades de procesamiento independientes que se conectan entre sí formando una red que se retroalimenta.

Por lo general, este tipo de redes se estructura en capas. Se puede distinguir en ellas una capa de entrada (por la que el modelo recoge las variables), la capa o capas ocultas y la capa de salida (que representa los resultados del procesado).

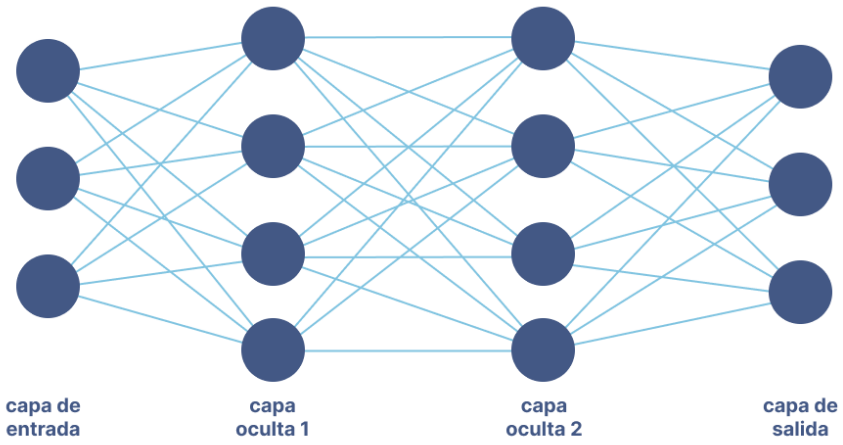


Figura 3: redes neuronales

Es interesante mencionar un estudio realizado sobre análisis del sentimiento para ayudar en emergencias, desastres naturales y COVID-19 mediante redes sociales. En el informe "International Journal of Disaster Risk Reduction, Twitter for disaster relief through sentiment analysis for COVID-19 and natural hazard crises [7]" se expone la idea de que, hoy en día se transmite la información por redes sociales de manera mucho más rápida que por informativos o noticias. Por lo tanto, en él se estudia la posibilidad de clasificar tweets para recopilar esta valiosa información. Uno de los métodos que se investigó para esta labor, fué el uso de redes neuronales ya que funcionan muy bien para datasets en los que la información no está organizada y es difícil de clasificar.

Capítulo 3 Metodología, tecnologías y estructura

En este capítulo se describen las tecnologías necesarias para llevar a cabo el desarrollo del proyecto, así como su estructura y definición de directorios requeridos.

3.1 Metodología

Se optó por utilizar una técnica de aprendizaje supervisado, ya que, para clasificaciones de texto posterior a un correcto procesamiento del lenguaje natural, presentaba muy buenos resultados. Sin embargo, la regresión lineal presentaba ciertas limitaciones debido a su naturaleza que hicieron cambiar la decisión de modelado. Tras las pruebas realizadas, se concluyó el método óptimo para el caso de las iniciativas parlamentarias. En concreto la decisión final fué la de optar por un modelo de regresión logística (el cual ha sido descrito anteriormente y será desarrollado en el Capítulo de Modelado).

Esta metodología es posible implementarla mediante la siguiente arquitectura.

Arquitectura de la solución

Para solucionar el problema se estableció un conjunto de fases que serán descritas a detalle posteriormente. Estas fases definen los pasos a seguir (y el orden de ejecución) a la hora de desarrollar un modelo de aprendizaje automático para procesamiento del lenguaje natural. Este proceso se divide en cuatro fases principales: Tratamiento de los datos, Preprocesado, Modelado y Testeo.

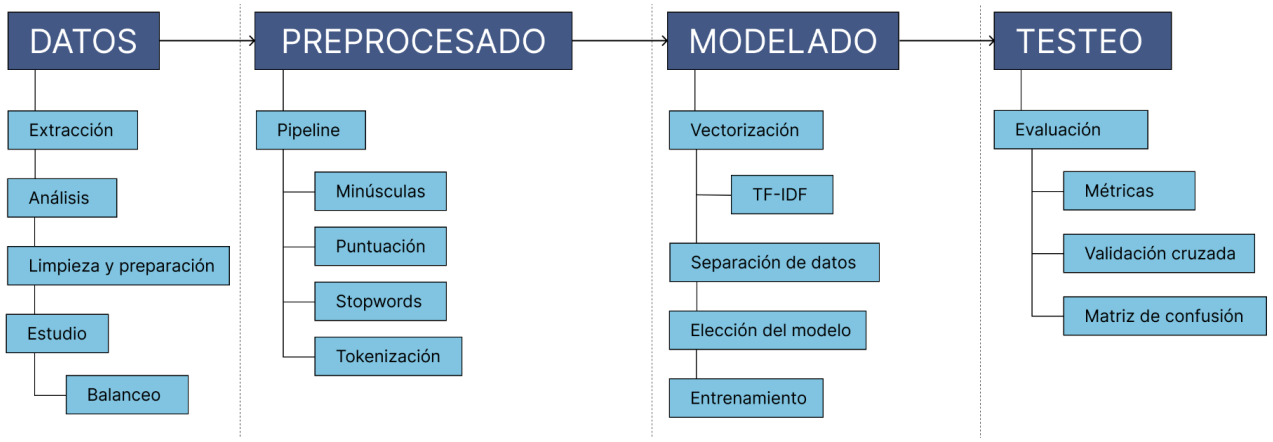


Figura 4: Arquitectura de la solución

Estas fases serán descritas a partir del Capítulo 4, entrando en detalle para cada uno de los pasos seguidos.

3.2 Tecnologías

Para llevar a cabo la metodología planteada, fue necesario el uso de ciertas tecnologías para cada propósito.

Visual Studio Code

Visual Studio Code se trata de un IDE (Entorno de Desarrollo Integrado), una aplicación que facilita el desarrollo de código. Esta herramienta permitió desarrollar el sistema bajo el lenguaje de programación Python. Los criterios bajo los que se seleccionó esta herramienta fueron: su sencillo uso, la comodidad que aporta para estructurar el código, la experiencia habiéndolo utilizado y su acceso a numerosas extensiones que sirvieron de gran ayuda.

Google Colab

Esta herramienta permite la creación de notebooks de Python que permiten explicar detalladamente el desarrollo acompañado de su ejecución en la nube. Resultó ser muy útil para documentar correctamente las pruebas previas realizadas.

Microsoft Excel

Se utilizó la herramienta de Excel para manipular el dataset (formato CSV) añadiendo, eliminando y modificando las diferentes columnas y contenido. Permite además, alterar la estructura del archivo, cambiando simultáneamente propiedades como el delimitador o la extensión (opción muy útil para el procesado posterior en el código).

3.3 Estructura

Los ficheros del repositorio siguen una determinada estructura que permite la correcta organización del proyecto y agilizan el flujo de trabajo y su ejecución.

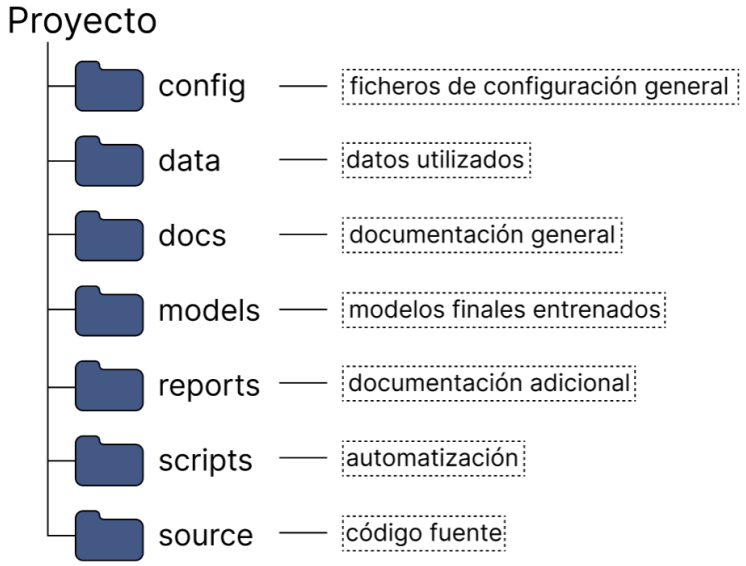


Figura 5: estructura de ficheros

La explicación detallada de la utilidad de cada carpeta y los ficheros que contienen, viene dada en el Apéndice C: Estructura del repositorio.

Capítulo 4 Tratamiento de los datos

El tratamiento de los datos es la primera fase del pipeline del proyecto. En esta etapa, se analizan los datos en crudo y se manipulan para convertirlos en un dataset procesable.

4.1 Naturaleza de los datos

Árbol de iniciativas

Los datos de los que se dispone son las propias iniciativas del Parlamento de Canarias. Estos datos son accesibles para el público general a través de la web https://www.parcan.es/. En ella se encuentran todo tipo de datos del Parlamento, entre ellos, las iniciativas. Disponemos de un árbol desplegable con todas las iniciativas para cada legislatura (https://www.parcan.es/iniciativas/arbol.py).

Figura 6: árbol de iniciativas

Este árbol ayuda en la visualización de los datos. Sin embargo, extraer los datos directamente de él, resulta una tarea inviable debido a la cantidad de datos requeridos para entrenar correctamente a un modelo.

Alternativa de extracción

La extracción de los datos correspondientes a las diferentes iniciativas parlamentarias del Parlamento de Canarias, puede ser simplificada. Dichos datos pueden obtenerse directamente a partir del conjunto de datos abiertos del Parlamento [8], en lugar del árbol desplegable de iniciativas. Concretamente, las Iniciativas se encuentran agrupadas según el año y número de legislatura. Cada una de ellas posee su identificador único.

Figura 7: formatos de los datos

Los grupos de iniciativas completos se encuentran disponibles tanto en formato JSON como en CSV.


```
[
  {
    "id_iniciativa": "10L/CSUE-0198",
    "legislatura": 10,
    "extracto": "Propuesta por la que se modifica el reglamento (ue) n.º 600/2014 en lo que se refiere a la mejora de la transparenc
establecimiento de un sistema de información consolidada, la optimización de las obligaciones de negociación y la prohibición de rec
pertinente a efectos del eee) com(2021) 727 final.",
    "f_creacion": "2022-01-26",
    "situacion": "En tramitación",
    "procedimiento": "Ordinario",
    "tipo_finalizacion": "",
    "tipo_descripcion": "CONSULTAS/INICIATIVAS LEGISLATIVAS EUROPEAS"
  },
```


Figura 8: datos en JSON

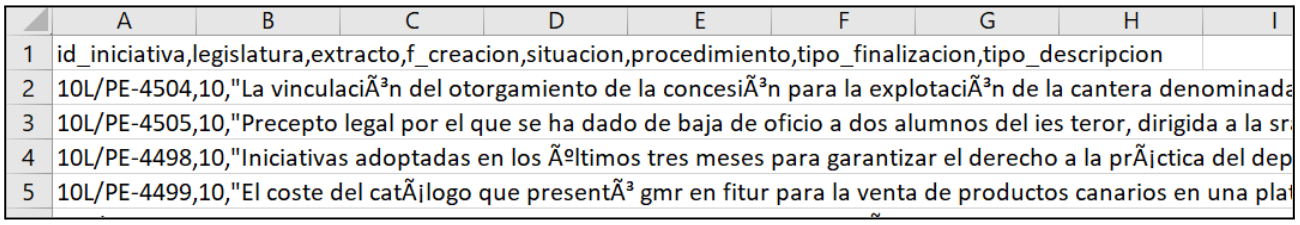
El dataset concreto que fué escogido en un principio para realizar el entrenamiento del modelo, fué el de Iniciativas PE del año 2022 - X Legislatura [9]. Este dataset estaba compuesto por 604 iniciativas que fueron usadas en formato csv.

4.2 Limpieza y preparación de los datos

Para el propósito del proyecto, la mayoría de campos del dataset no son utilizados. Además, el dataset precisa de una clasificación adicional de la información que contiene.

Limpieza

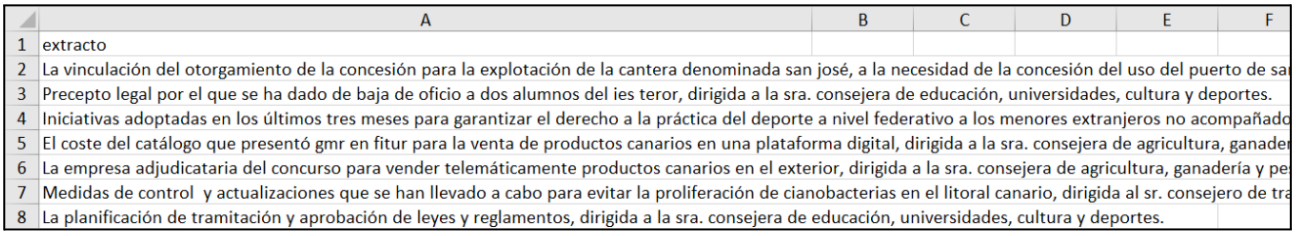
Los datos fueron manipulados en Excel para eliminar los campos innecesarios. Fueron eliminados los campos de “id_iniciativa”, “legislatura”, “f_creacion”, “situacion”, “procedimiento”, “tipo_finalizacion”, “tipo_descripcion”. Además, el delimitado del documento fué sustituido por “;” para su correcta lectura por el procesador de csv (conservando tildes, puntuación…)



	A	B	C	D	E	F	G	H	I
1	id_iniciativa,legislatura,extracto,f_creacion,situacion,procedimiento,tipo_finalizacion,tipo_descripcion								
2	10L/PE-4504,10,"La vinculaciÃ³n del otorgamiento de la concesiÃ³n para la explotaciÃ³n de la cantera denominada								
3	10L/PE-4505,10,"Precepto legal por el que se ha dado de baja de oficio a dos alumnos del ies teror, dirigida a la sr								
4	10L/PE-4498,10,"Iniciativas adoptadas en los Ãºltimos tres meses para garantizar el derecho a la prÃ¡ctica del dep								
5	10L/PE-4499,10,"El coste del catÃ¡logo que presentÃ³ gmr en fitur para la venta de productos canarios en una plat								

Figura 9: campos de Excel

El resultado final de este paso es un fichero csv con todos los extractos. Este es el campo que se utilizó de las iniciativas para entrenar al modelo.



	A	B	C	D	E	F
1	extracto					
2	La vinculación del otorgamiento de la concesión para la explotación de la cantera denominada san josé, a la necesidad de la concesión del uso del puerto de sa					
3	Precepto legal por el que se ha dado de baja de oficio a dos alumnos del ies teror, dirigida a la sra. consejera de educación, universidades, cultura y deportes.					
4	Iniciativas adoptadas en los últimos tres meses para garantizar el derecho a la práctica del deporte a nivel federativo a los menores extranjeros no acompañado					
5	El coste del catálogo que presentó gmr en fitur para la venta de productos canarios en una plataforma digital, dirigida a la sra. consejera de agricultura, ganade					
6	La empresa adjudicataria del concurso para vender telemáticamente productos canarios en el exterior, dirigida a la sra. consejera de agricultura, ganadería y pe					
7	Medidas de control y actualizaciones que se han llevado a cabo para evitar la proliferación de cianobacterias en el litoral canario, dirigida al sr. consejero de tra					
8	La planificación de tramitación y aprobación de leyes y reglamentos, dirigida a la sra. consejera de educación, universidades, cultura y deportes.					

Figura 10:Excel limpio

Clasificación

Sin embargo, el extracto no es el único campo que se necesita para el proyecto. Al tratarse de un modelo de aprendizaje supervisado, es necesario que el dataset tenga de antemano la información clasificada para que el algoritmo pueda aprender lo que debe hacer correctamente. Para establecer las clases que se van a utilizar en el dataset, se eligió en un principio la utilización de los principales campos temáticos del tesauro EuroVoc [10] de la Unión Europea. En estos campos se encuentran englobados todos los conceptos de los que puede tratar una iniciativa parlamentaria.

Se volvió a hacer uso de la herramienta de Excel. Cada extracto fué categorizado manualmente para cada una de las iniciativas a clasificar. (asignando de esta manera la categoría correspondiente a cada uno de los 604 extractos que se tenían disponibles).

4.3 Estudio y análisis de los datos limpios

En esta primera instancia quedaron las iniciativas categorizadas, sin embargo, no todos los 21 campos del léxico EuroVoc. Las categorías que fueron utilizadas son:

- 'TRANSPORTES'
- 'AGRICULTURA, SILVICULTURA Y PESCA'
- 'ASUNTOS SOCIALES'
- 'EDUCACIÓN Y COMUNICACIÓN'
- 'INTERCAMBIOS ECONÓMICOS Y COMERCIALES'
- 'ASUNTOS FINANCIEROS'
- 'MEDIO AMBIENTE'
- 'ECONOMÍA'
- 'INDUSTRIA'
- 'TRABAJO Y EMPLEO'
- 'VIDA POLÍTICA'
- 'PRODUCCIÓN, TECNOLOGÍA E INVESTIGACIÓN'
- 'DERECHO'
- 'ENERGÍA'

Balanceo

Se estudió la cantidad de iniciativas que había para cada categoría establecida. Para ello se realizó un diagrama de barras del dataset.

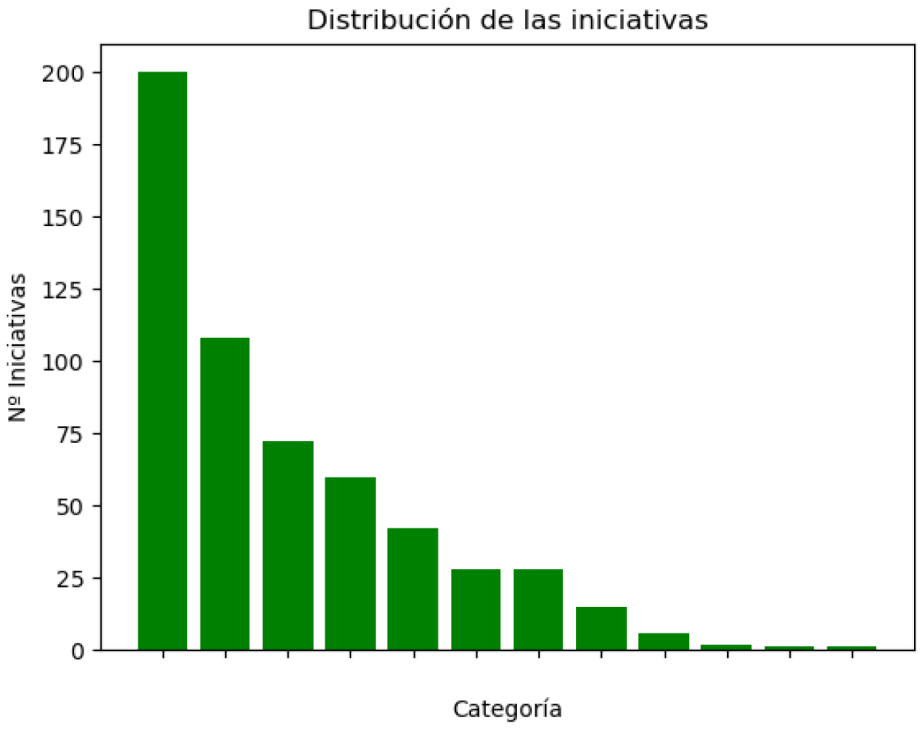


Figura 11: datos desbalanceados

Observando los datos, se puede detectar a simple vista un gran desbalance de los mismos. Esto no significa necesariamente que el resultado final empeore, sin embargo, un mal “recall” (reconocimiento de las clases) suele ir íntimamente relacionado con un muestreo muy pobre de la misma en relación con el resto de clases del dataset. Para evitar este problema a futuro, se estudiaron las diferentes maneras de equiparar las

distintas clases para reducir este efecto y optimizar el recall.

Una de las posibles estrategias para solucionar este problema es el denominado *subsampling*. Se basa en reducir las muestras de la clase o clases mayoritarias para así contar con un número parecido de muestras en cada una, esta reducción generalmente se aplica usando un algoritmo como “k-nearest neighbor” que se encargue de reducir de manera discriminada eligiendo los elementos más diferenciativos. Este método funciona bastante bien y es capaz de mejorar el recall considerablemente; sin embargo, al reducir tan drásticamente la cantidad de de muestras como podría ser el caso de las iniciativas, la cantidad de falsos positivos se incrementaría mucho, reduciendo así la efectividad general de la aplicación.

El *oversampling* resultó ser una estrategia mucho más adecuada para el problema. Esta técnica se basa en aumentar las muestras de las categorías minoritarias de manera sintética. Realizando esto, se puede aumentar el recall y conservar el dataset al completo sin afectar a la precisión. En concreto, la técnica de oversampling utilizada se conoce como SMOTE (Synthetic Minority Oversampling Technique) [11]. Con esta técnica se consiguen sintetizar nuevos ejemplos para la muestra en lugar de técnicas más simples como podría ser duplicar las existentes.

Para este proyecto se combinó SMOTE con el uso de “Tomek links”

```
if cfg.pipeline.balanceo == True:
  print("\n> Empezando el balanceo del dataset...")
  resample = SMOTETomek(tomek=TomekLinks(sampling_strategy='majority'))
  X, y = resample.fit_resample(X, y)
```

Para poder realizar SMOTETomek, es necesario un mínimo de datos para cada categoría ya que el oversampling se realiza mediante el método de k-nearest neighbours. Para lograr este ligero balance, se introdujeron manualmente iniciativas de la legislatura X de los años 2021 y 2020, únicamente de las categorías más desbalanceadas hasta llegar a ese mínimo. Se añadieron elementos a las categorías: “ENERGÍA”, “PRODUCCIÓN, TECNOLOGÍA E INVESTIGACIÓN”, “DERECHO” y “INTERCAMBIOS ECONÓMICOS Y COMERCIALES”. Tras este ligero balanceo, quedan 649 elementos en el dataset distribuidos como se puede comprobar en la Figura 12:

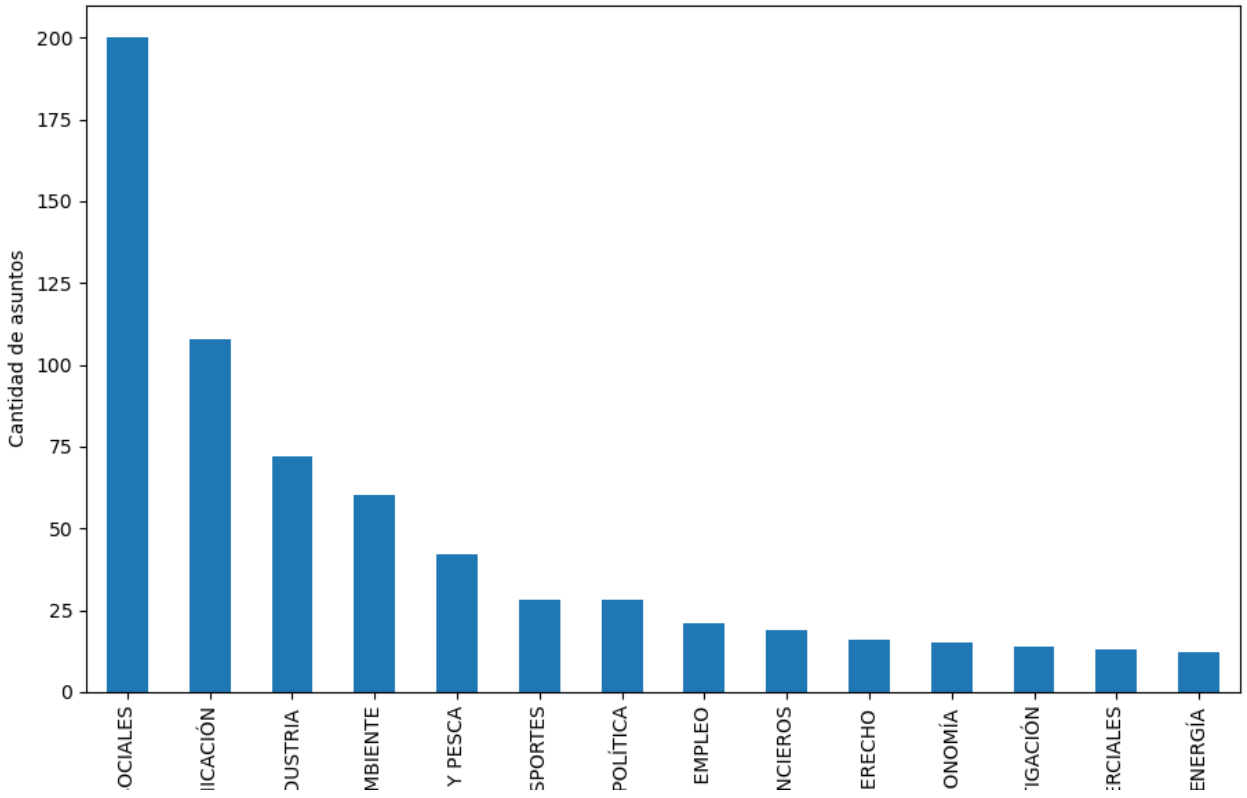


Figura 12: datos complementados

Se aplicó el balanceo a los datos mediante SMOTETomek, quedando, de esta manera, todos los datos nivelados a 200 elementos.

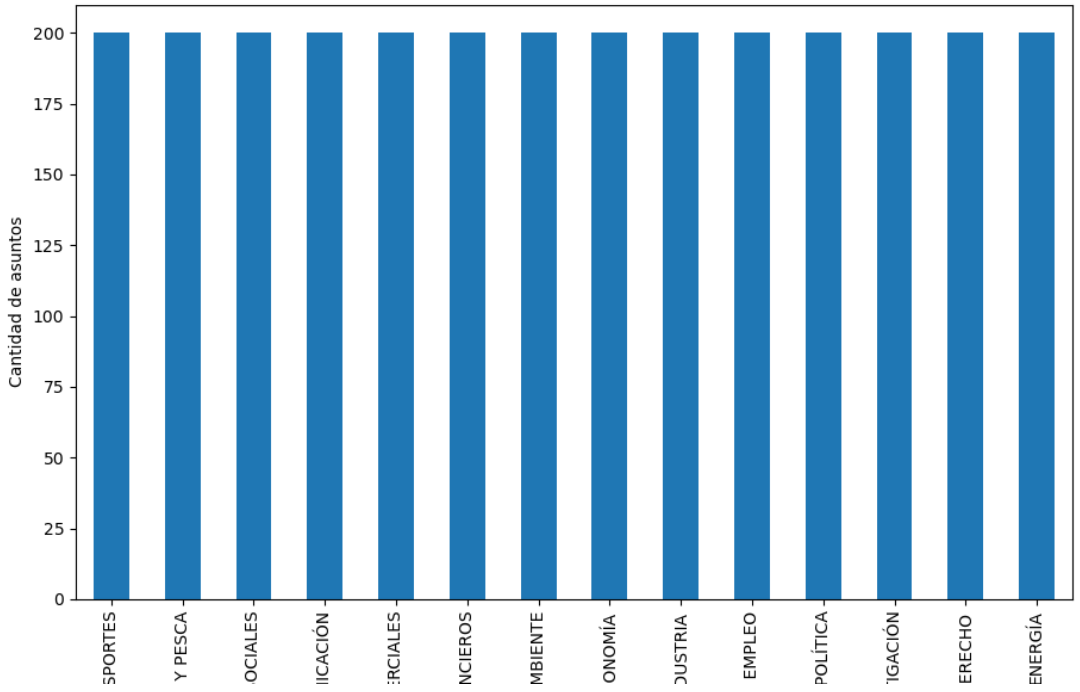


Figura 13: datos balanceados

Tras realizar el balanceo, el dataset resultante acabó con 2800 datos entre los elementos escogidos y las muestras sintéticas generadas.

Capítulo 5 Preprocesado

Para poder utilizar el dataset correctamente es necesario realizar un preprocesado de los datos. Esta fase se subdivide en distintos procesos a los que se somete el dataset para descartar o formatear los elementos que suponen información no deseada ni útil para el entrenamiento del modelo.

5.1 Minúsculas

Existe una función implementada para transformar palabras de caracteres en mayúsculas a minúsculas. Con la función que implementamos, es posible recorrer todo el texto del dataset y aplicar esta transformación a cada elemento.

```
def minusculas(texto):

  texto_minusculas = "".join(palabra.lower() for palabra in texto)

  return texto_minusculas
```

Una vez convertido todo el texto a minúsculas, se logra un dataset mucho más uniforme que nos facilitará el trabajo, ya que será más fácil para el algoritmo entender que una palabra significa lo mismo esté escrita en mayúscula o en minúscula.

5.2 Puntuación

En esta fase, eliminamos todos los elementos de puntuación, ya que son caracteres que no son utilizados por el modelo y generan ruido a la hora de entrenarlo. La librería “string” contiene un elemento llamado “punctuation” que almacena todos los caracteres de puntuación:

```
!"#$%&'()*+, -./:;<=>?@[\]^_`{|}~
```

A esta cadena también le sumamos los caracteres de interrogación usados en español (que también es deseable descartarlos). Una vez obtenida la cadena de comprobación, se recorre el texto entero comprobando, para cada caso, si el elemento se encuentra o no en la cadena. En caso de no encontrarse en ella sabremos que no es un elemento de puntuación y se contará como texto limpio.

```
import string


def puntuacion(texto):
  punctuation_words = string.punctuation
  punctuation_words += "¿¡"
   texto_limpio = "".join(palabra for palabra in texto if palabra not in
punctuation_words)
  return texto_limpio
```

5.3 Stopwords

En el preprocesado de texto, hay ciertas palabras que siempre se repiten con demasiada frecuencia y no aportan información relevante para el modelo. Estas palabras no tienen un significado que pueda ser de utilidad ni determinante para establecer una solución. Entre este tipo de palabras se encuentran: artículos, pronombres, preposiciones, determinantes… Ha sido confeccionada una lista de las “stopwords” más frecuentes y, de esta manera, quedarán ignoradas para mejorar el procesado.

```
import pandas as pd


datos_csv = pd.read_csv('../../data/external/stopwords.csv')
stopwords = datos_csv['stopwords'].values.tolist()


def limpiar_stopwords(texto):
  texto = texto.split()
  texto_limpio = [palabra for palabra in texto if palabra not in stopwords]
  return texto_limpio
```

Recorriendo el texto, se comprobará palabra por palabra que no se encuentre en la lista predefinida de stopwords. Si la palabra efectivamente no coincide con ninguna de las de la lista, se considerará como texto limpio.

5.4 Tokenización

Tokenizar es el último paso del preprocesado. En la tokenización se divide el texto en los elementos que lo conforman. Cada elemento supone la palabra mínima que conserva un significado propio único.

Para realizar la tokenización nos apoyamos del método “word_tokenize” de la librería “NLTK”. Este método se encarga de subdividir la cadena que recibe y devuelve un

array con todas las palabras resultantes.

```
from nltk.tokenize import word_tokenize


def tokenizar(texto):
  texto = " ".join(texto)
  tokenizado = word_tokenize(texto)
  return tokenizado


if __name__ == '__main__':
  tokenizar()
```

5.5 Pipeline de preprocesado

Todas las operaciones descritas anteriormente quedan englobadas en un “pipeline” de preprocesado que establece el orden correcto y la manera de aplicarlas. Este pipeline hace uso de las variables globales de configuración y, en función de su valor, aplica o no las operaciones correspondientes. En él se llama a todas las funciones para poder llevar a cabo el proceso completo.

```
from data.minusculas import minusculas
from data.puntuacion import puntuacion
from data.stopword import limpiar_stopwords
from data.tokenizacion import tokenizar


def preprocesado(text, lw, pc, st, tk):
  print('> Empezando el preprocesado...')
  print('-- text raw: ', text)
  if lw == 1:
    text = minusculas(text)
    print('> Minúsculas: Hecho')
  if pc == 1:
    text = puntuacion(text)
    print('> Puntuación: Hecho')
  if st == 1:
```

```
    text = limpiar_stopwords(text)
    print('> Stopwords: Hecho')
  if tk == 1:
    text = tokenizar(text)
    print('> Tokenizado: Hecho')
  print('Preprocesado completado! <\n')
  return text


if __name__ == '__main__':
    preprocesado()
```

A continuación se muestra un ejemplo de la transformación que sufre el texto a medida que pasa por cada proceso:

Texto en crudo:

```
Inversión para la potenciación del comercio en los mercados y mercadillos en 2019, dirigida a la sra. consejera de turismo, industria y comercio.
```

Minúsculas:

```
inversión para la potenciación del comercio en los mercados y mercadillos en 2019, dirigida a la sra. consejera de turismo, industria y comercio.
```

Puntuación:

```
inversión para la potenciación del comercio en los mercados y mercadillos en 2019 dirigida a la sra consejera de turismo industria y comercio
```

Stopwords:

```
inversión potenciación comercio mercados mercadillos 2019 dirigida consejera turismo industria comercio
```

Tokenización:

```
['inversión', 'potenciación', 'comercio', 'mercados', 'mercadillos', '2019', 'dirigida', 'consejera', 'turismo', 'industria', 'comercio']
```

Una vez sometido a este preprocesado, el texto de entrada está apto para entrenar y testear al modelo.

Capítulo 6 Modelado

Una vez se dispone de los datos preprocesados, está todo preparado para entrenar al modelo de aprendizaje automático. Para ello, se seguirán una serie de pasos ordenados jerárquicamente.

6.1 Vectorización

Para poder trabajar con los datos de entrada, el algoritmo no puede usar el texto natural para realizar los cálculos necesarios. Todas las palabras que hemos procesado deben "vectorizarse", es decir, transformarse en valores numéricos.

Para entender el concepto de vectorización, tenemos que observar nuestro conjunto de información recogida mediante los procesos anteriores como un modelo de "bolsa de palabras" o "BOW" (por sus siglas en inglés, bag-of-words). Este modelo contendría todas las palabras sin repeticiones representando así, todo el conocimiento único que ha logrado recoger la máquina. Las palabras o frases que vectorizamos, se codifican en base a las apariciones de los elementos únicos de nuestra bag-of-words en ellas.

Suponemos un ejemplo en el que tenemos que procesar 3 frases distintas:

1. *A mi me gustan las manzanas pero no las peras.*
2. *Hay dos camiones rojos aparcados.*
3. *Mis frutas favoritas son las manzanas, me gustan mucho.*

La información de la que disponemos pasaría por un estricto preprocesado del que se extraería información útil para construir este bag-of-words:

["gustan", "manzanas", "peras", "camiones", "rojos", "aparcados", "frutas", "mucho"]

De esta manera se podrían vectorizar las frases de entrada:

bag-of-words

frases	gustan	manzanas	peras	camiones	rojos	aparcados	frutas	favoritas	mucho
1	1	1	1	0	0	0	0	0	0
2	0	0	0	1	1	1	0	0	0
3	1	1	0	0	0	0	0	1	1

Tabla 1: vectorización

Podemos comprobar que los vectores se parecen en mayor o menor medida en

función de la relación que guardan. De esta manera, las frases 1 y 3 guardan cierta relación y queda reflejado en su vectorización:

1. ***[1,1,1,0,0,0,0,0,0]***
2. ***[0,0,0,1,1,1,0,0,0]***
3. ***[1,1,0,0,0,0,0,1,1]***

TF-IDF

Tras la vectorización, la información recogida es bastante completa. Se puede codificar el conjunto de datos en función de una gran cantidad de términos con sus significados específicos. Sin embargo, podemos obtener nueva información interesante y valiosa a partir del registro de apariciones de estos términos. Se considera que un término es relevante en función del número de veces que aparece.

El TF*IDF es una métrica que mide la "frecuencia del término" por la "frecuencia inversa del documento", es decir, tiene en cuenta el número total de la colección de documentos que se analizan y mide con qué frecuencia aparece un término en función de este número.

Suponemos un ejemplo en el que tenemos una gran cantidad de datos hablando sobre un tópico concreto:

Imaginemos que tenemos un dataset de recetas de cocina. En la gran mayoría de datos recogidos aparecerán términos como "preparación" o "elaboración". Estos términos aparecerán con una frecuencia altamente superior al resto, sin embargo, no son términos realmente diferenciales y no van a aportar información nueva que ayude al algoritmo a mejorar su aprendizaje. Este tipo de términos no son útiles para establecer relaciones entre nuestros datos, ya que prácticamente todos los comparten.

En el caso concreto de nuestro dataset, se observa esta característica para el término "dirigida". Esto es debido a que todas las iniciativas acaban especificado a que consejal, ministerio o cualquier otro organismo va dirigida la intervención y, para todas ellas, se utiliza la palabra "dirigida". Gracias al uso de la vectorización por TF-IDF, este término no ocasionará problemas para nuestro modelo mejorando considerablemente su rendimiento.

Para aplicar esta vectorización en Python, haremos uso de la función de la librería sklearn "*TfidfVectorizer* [12]" y la aplicaremos a nuestro conjunto de datos:

```
bow_converter = TfidfVectorizer()
X = bow_converter.fit_transform(X)
```

6.2 Separación de datos

La máquina necesita un conjunto de datos para entrenarse. En función del conocimiento inferido de este entrenamiento, se probará su funcionamiento con otro conjunto de datos para comprobar la proporción de aciertos y errores.

Esta división de los datos se realiza sobre el dataset del que disponemos. La proporción puede variar, para este modelo se probó dedicando un 80% de la muestra a

entrenamiento y el 20% restante a testeo

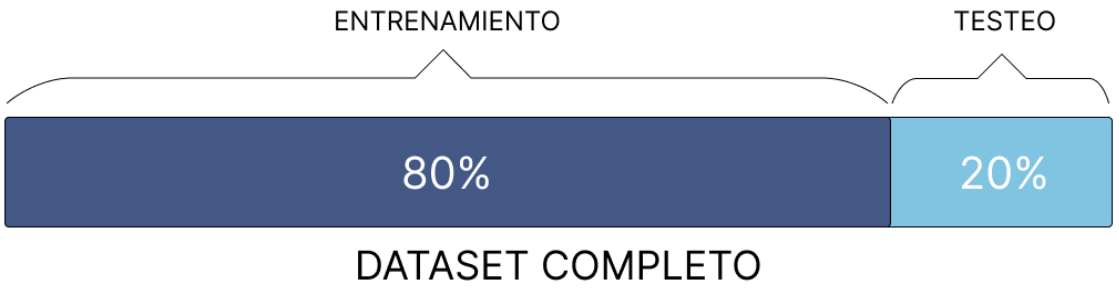


Figura 14: separación de datos

Para este propósito, nos apoyaremos de un método ya implementado de la librería de sklearn que dividirá los datos de manera aleatoria para mantener cierta imparcialidad.

```
X_train,X_test, Y_train,Y_test = train_test_split(X,y,test_size=0.2,
random_state=25)
```

Al indicarle un 0.2 al argumento “test_size”, establecemos que efectivamente se divida un 20% para testeo.

Al dividir los datos en X e Y tanto para testeo como para entrenamiento, lograremos tener los asuntos separados de sus categorías para, posteriormente, evaluar si el algoritmo ha categorizado correctamente los asuntos con los que ha sido probado.

6.3 Elección del algoritmo

Tras la realización de diversas pruebas (como se documentó en capítulos anteriores), se formuló una decisión concreta sobre qué tipo de algoritmo sería el óptimo para el problema en cuestión.

Para resolver este problema se utilizó una implementación de regresión logística. El método de regresión logística fue descrito en el Capítulo 2 de Estado del Arte. En base a esta descripción se justifica su elección, ya que es, de los métodos estudiados, el que más se adapta a nuestro tipo de problema.

6.4 Entrenamiento

Durante el entrenamiento, se crea el modelo y este recibe los datos seleccionados para este propósito. Mediante el procesamiento de estos datos, el algoritmo aprende cómo debe clasificar los datos de entrada y establece sus propias relaciones entre ellos.

Para crear el modelo, haremos uso del método “*LogisticRegression* [13]” de sklearn. Tras diversas pruebas, se escogieron los valores de los argumentos “penalización”, “C” y “max_iter” que mejores resultados aportaban para nuestro caso.

```
model_precision = LogisticRegression(C=1000, penalty='l2', max_iter=10000,
multi_class="ovr").fit(X_train, Y_train)
```

Gracias al conjunto de operaciones aplicadas en el preprocesado y el tratamiento de los datos, el entrenamiento resultó un procedimiento muy sencillo de aplicar y no supuso mayores complicaciones en el desarrollo.

Capítulo 7 Testeo

Una vez entrenado el modelo, el siguiente paso es comprobar si los resultados obtenidos de este entrenamiento se corresponden con la realidad. Mediante una serie de medidas y técnicas, se puede corroborar la fiabilidad del modelo y comprobar la manera en la que se comporta frente a los datos.

7.1 Evaluación

Disponemos de distintas formas de evaluar los resultados.

Métricas

Hasta este punto del problema, solo habíamos utilizado la parte de los datos correspondiente a entrenamiento. Una vez finalizada esta etapa, llega el momento de utilizar la parte de los datos que teníamos reservada para testeo (corresponde a un 20% del total del dataset).

El método de regresión logística que hemos utilizado, nos da la posibilidad de establecer predicciones de categorización en base al entrenamiento realizado. Estas predicciones son, en esencia, el criterio que vamos a utilizar para determinar la efectividad del algoritmo.

La manera en la que funciona el método "*predict*" de la librería sklearn se basa en tratar de hallar las categorías correspondientes para el conjunto de datos que le habíamos indicado que eran para testeo. Estas predicciones se compararán con las categorías reales que tenían los datos originalmente, y se podrán obtener datos de con qué probabilidad es capaz de acertar el algoritmo en su clasificación.

```
model_precision = LogisticRegression(C=1000, penalty='l2', max_iter=10000,
multi_class="ovr").fit(X_train, Y_train)

pred_tests = model_precision.predict(X_test)

print("Modelo de precisión completado! <\n")
```

Para evaluar correctamente un modelo, hemos decidido obtener los resultados de 4 métricas principales. Una vez más, haremos uso de métodos predefinidos de la librería sklearn (en este caso, del apartado de "*metrics* [14]"):

- **Precisión**: La precisión mide la habilidad del modelo de no identificar como positivo un elemento que es negativo. Se calcula por el número de "verdaderos positivos" de entre todos los positivos asignados (ratio de asignaciones correctas):

$$\frac{tp}{(tp + fp)}$$

"tp" y "fp" representan el número de "verdaderos positivos" y "falsos positivos"

respectivamente.

- **Recall**: El recall se utiliza para determinar la facilidad del modelo para reconocer las clases. Se define como la habilidad del modelo de no clasificar como “positivo” un elemento “negativo”.

$$\frac{tp}{(tp + fn)}$$

“tp” y “fp” representan el número de “verdaderos positivos” y “falsos positivos” respectivamente.

- **F1**: La medida F1 se puede definir como la media armónica de precisión y recall. La contribución por separado de la precisión y el recall a la métrica es igual.

$$F1 = 2 * \frac{(precisión * recall)}{(precisión + recall)}$$

- **Accuracy**: Con esta medida podemos identificar los casos en los que las predicciones encajan exactamente con las categorías establecidas en los datos reservados para testeo.

```
print("-- Testeo de la precisión:", precision_score(Y_test, pred_tests,
average='weighted'))

print("-- Testeo del recall:", recall_score(Y_test, pred_tests,
average='weighted'))

print("-- Testeo del f1:", f1_score(Y_test, pred_tests, average='weighted'))

print("-- Testeo del accuracy:", accuracy_score(Y_test, pred_tests))
```

Nuestro modelo ha obtenido unos resultados muy favorables (los valores están comprendidos entre 0 y 1, siendo 1 el máximo):

- Precisión: **0.9815**
- Recall: **0.9804**
- F1: **0.9819**
- Accuracy: **0.9804**

Podemos generalizar, a partir de los resultados, que el algoritmo acierta la clasificación en un 98% de los casos.

La captura completa del reporte se encuentra en el Apéndice D: Resultados de la evaluación.

Validación cruzada

A la hora de obtener las métricas y evaluar el rendimiento del algoritmo, es posible que el resultado obtenido no refleje la realidad de manera fidedigna por diferentes

motivos.

A la hora de establecer la separación de los datos de manera aleatoria, cabe la posibilidad de que, justo el dataset reservado en unas proporciones concretas, de un resultado más o menos óptimo y que no sea el resultado más representativo del trabajo realizado por el modelo.

La validación cruzada [15] es una técnica que se basa en realizar un número de iteraciones del mismo entrenamiento escogiendo y separando en cada una, una parte distinta del dataset. De esta manera, será muy sencillo localizar lo que se conoce como un sobreajuste [16], o una anomalía en los datos.

Los resultados de la validación cruzada se observan mediante la comparación de las métricas obtenidas en cada una de las iteraciones. Si el conjunto de datos está equilibrado y el modelo funciona correctamente, los valores de las distintas métricas deberían variar muy poco entre sí. Un conjunto de mediciones homogéneas indica, presumiblemente, una gran robustez del algoritmo y ejemplifica un comportamiento adecuado frente a distintos tipos de datos de prueba.

En nuestro caso concreto, se ha empleado esta validación cruzada estratificada en una serie de iteraciones, dando como resultado unas métricas muy parecidas. Se ha elaborado un reporte final que calcula los valores máximos, mínimos y totales de precisión en los que identificamos valores muy correctos:

- Precisión máxima: **0.9**
- Precisión mínima: **0.8393**
- Precisión general: **0.87**

Tras haber realizado todos estos tests, podemos comprobar que la división establecida inicialmente es bastante optimista. El 87% de precisión general supone un resultado más representativo.

El reporte completo se encuentra disponible en el Apéndice D: Resultados de la evaluación.

Matriz de confusión

Hasta ahora, hemos realizado una evaluación de los resultados a nivel general. Con la evaluación de métricas logramos ver el porcentaje de acierto y con qué fiabilidad es capaz de establecer una categorización nuestro modelo. Sin embargo, con el fin de mejorar el modelo, es de gran interés conocer la manera en la que falla el modelo, los casos en los que nuestro algoritmo no es capaz de asignar la categoría correcta correspondiente.

La matriz de confusión [17] es una herramienta que nos permite visualizar de qué manera ha asignado las categorías nuestro algoritmo. Las filas representan las categorías disponibles para asignar y las columnas representan la cantidad de asignaciones que se han realizado.

Para simplificar el concepto, ejemplificaremos una matriz de confusión en la que el caso de estudio es una clasificación binaria en lugar de una multiclase:

"*Imaginemos el caso de un algoritmo que debe clasificar ciertas sentencias en las categorías "verdadero" o "falso". En este caso, constaríamos de 100 sentencias donde la*

mitad son falsas y la otra mitad son verdaderas. De esta manera, si el algoritmo clasifica todas las sentencias correctamente, debería de ser capaz de colocar las 50 sentencias de cada categoría en la que corresponde."

A la hora de mostrar la matriz de confusión, se pueden dar dos casos: En el caso favorable, el algoritmo logra reconocer perfectamente ambas clases y no se dan casos de falsos positivos ni negativos. Sin embargo, puede darse el caso de que el modelo identifique un elemento como perteneciente a otra de las clases en mayor o menor medida.

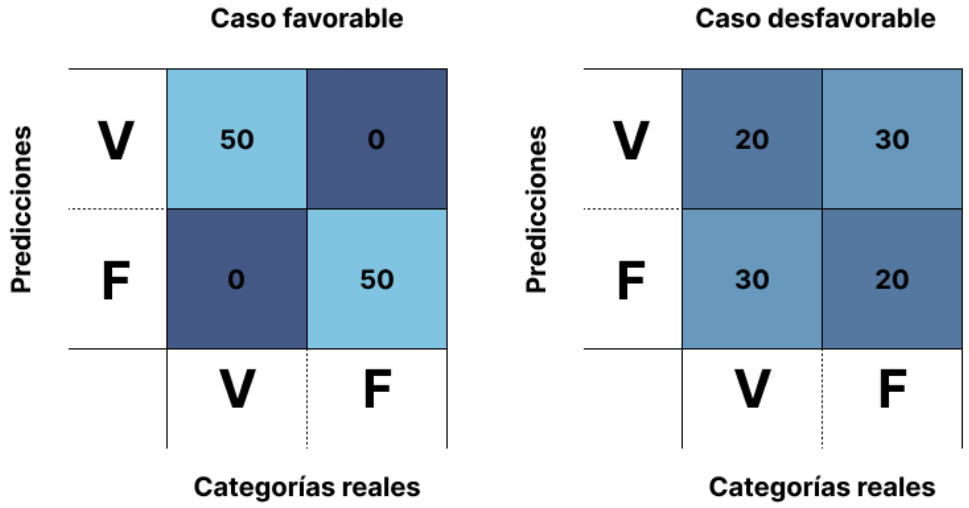


Figura 15: matriz de confusión binaria

En el caso desfavorable del ejemplo se puede comprobar visualmente y de manera clara, que las categorías no se están reconociendo de manera correcta. Viendo esta matriz de confusión, se puede determinar que el modelo no funciona correctamente.

En casos como el nuestro, donde se manejan múltiples clases, la matriz de confusión resulta de mayor utilidad todavía. Al observar la manera en la que el modelo confunde las clases, se pueden detectar problemas con ciertas clases en concreto (por ejemplo clases parecidas que se pudieran unificar, o establecer límites más restrictivos y determinantes) y trabajar directamente sobre ellas para mejorar notablemente el rendimiento.

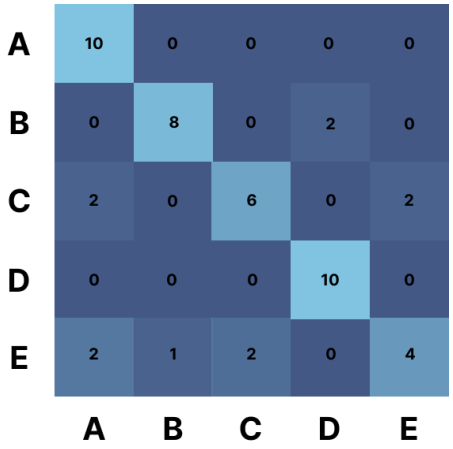


Figura 16: matriz de confusión multiclase

Observando la matriz de la figura 16, comprobamos que el modelo que representa funciona bastante bien a la hora de reconocer las clases. Sin embargo detectamos un fallo recurrente a la hora de tratar de clasificar elementos de la categoría “E”. La matriz nos aporta la información, además, de que la confusión de esta categoría se puede deber a su relación con las categorías “A”, “B” y “C”, ya que el modelo suele clasificar sus elementos en ellas.

En muchos casos, los problemas a resolver poseen un gran número de categorías. Para este tipo de problemas, la matriz de confusión se realiza con un número reducido de categorías escogidas de manera aleatoria pretendiendo que sea lo más representativo posible. En nuestro caso, no disponemos de tantas categorías y, por lo tanto, se pueden registrar todas en la matriz sin mayor complicación.

Tras sacar nuestra propia matriz de confusión, no se identifican clases lo suficientemente mal reconocidas como para tener en cuenta su error. La matriz entera se encuentra en el Apéndice D: Resultados de la evaluación.

Capítulo 8 Conclusiones y líneas futuras

Conclusiones

En el ámbito de las administraciones públicas, se recogen grandes cantidades de datos de muchas índoles distintas. Estos datos son originarios de distintas fuentes y recogidos de manera diferente en cada caso. Es de vital importancia conocer métodos, técnicas y herramientas que ayuden a entender estos recursos en conjunto, procesarlos según corresponda y utilizarlos para beneficio propio. El aprendizaje automático juega un papel fundamental en la transformación de datos en conocimiento e información útil, así como la elaboración de modelos considerablemente capaces.

Este proyecto aporta una posible solución a uno de estos problemas. En este caso, se pretende resolver un problema de procesamiento de lenguaje natural y elaboración de un sistema de categorización. Se ha desarrollado una aplicación propuesta como solución que se ayuda de un tratamiento y procesamiento de datos específico para implementar un algoritmo de clasificación, en concreto mediante regresión logística, capaz de categorizar de manera efectiva un gran número de iniciativas parlamentarias.

El modelo final es capaz de clasificar los elementos con un 98% de acierto, dejando un márgen de error bastante reducido. El modelo fue testeado mediante numerosas pruebas para corroborar su fiabilidad.

El trabajo realizado, supondrá una gran ayuda y referencia para otros problemas de clasificación de datos.

Líneas futuras

Se pretende continuar colaborando con el Parlamento de Canarias para mejorar esta idea, seguir su desarrollo y aportar avances y soluciones de manera conjunta para el ámbito de la categorización y el aprendizaje automático en general. A corto plazo, se pretende publicar estos resultados para construir una base sobre la que actuar y unir fuerzas.

Se planea continuar el testeo del modelo. Mediante la introducción de nuevos datasets de iniciativas, se lograría previsiblemente mejorar el modelo de manera sencilla. Se considera que el modelo es lo suficientemente capaz como para ser utilizado en la categorización de nuevos conjuntos de iniciativas. Se propone categorizar nuevos bloques de datos y, tras una ligera revisión manual, alimentar de nuevo al modelo con estos datos correctos. De esta manera, se amplía el conocimiento del que dispone el sistema y se prevé, con ello, una mejora considerable de su fiabilidad.

Se está estudiando el uso y la aplicación a corto plazo de una librería de preprocesado altamente eficiente (actualmente en desarrollo) que engloba e implementa las operaciones más frecuentes para el procesamiento de lenguaje natural en español. Mediante la librería Preln [19], se presupone una eficaz refactorización del código fuente de la aplicación y una mejora de los tiempos de ejecución al aligerar la fase de preprocesado.

Capítulo 9 Summary and Conclusions

Conclusiones

In the public administrations' ambit, large amounts of data of many different kinds are collected. These data originate from different sources and are collected differently in each case. It is truly important to know methods, techniques and tools that help understanding these resources as a collection, process them accordingly and use them to our advantage. Machine learning plays a fundamental role in transforming data into useful knowledge and information, as well as building considerably capable models.

This project provides a possible solution to one of these problems. In this case, it is intended to solve a problem of natural language processing and elaboration of a categorization system. A proposed application has been developed as a solution that uses a specific data treatment and processing to implement a classification algorithm, specifically by means of logistic regression, capable of effectively categorizing a large number of parliamentary initiatives.

The final model is able to classify the elements with 98% accuracy, leaving a fairly small margin of error. The model was tested through numerous tests to corroborate its reliability.

The work done will be a great help and reference for other data classification problems.

Líneas futuras

It is intended to continue collaborating with the Parliament of the Canary Islands to improve this idea, continue its development and jointly provide advances and solutions for the ambit of categorization and machine learning in general. In the short term, we intend to publish these results to build a base on which to act and join forces.

Further testing of the model is planned. By introducing new datasets of initiatives, the model is expected to be improved in a straightforward manner. The model is considered sufficiently capable to be used for categorizing new sets of initiatives. It is proposed to categorize new blocks of data and, after a slight manual revision, to feed this correct data back into the model. In this way, the knowledge available to the system is expanded and, with it, a considerable improvement in its reliability is expected.

Is being studied the use and application in the short term of a highly efficient preprocessing library (currently under development) that groups and implements the most frequent operations for natural language processing in Spanish. By means of the Preln library [19], we assume an efficient refactoring of the application source code and an improvement of the execution times by lightening the preprocessing phase.

Capítulo 10 Presupuesto

En este apartado se refleja una propuesta de presupuesto para el trabajo realizado a lo largo del proyecto. Para este proyecto no fue necesario ningún material en específico por lo que sólo se contemplan las horas de trabajo en el modelo.

10.1 Tabla de Costos

Tarea	**Tiempo estimado (horas)**	**Costo €/h**
Recopilación de información y referencias	30	5
Revisión del estado del arte	10	10
Planificación de las fases	15	10
Desarrollo del sistema	40	25
Testeo del sistema	10	25
Total	**105**	**1650€**

Tabla 2: Presupuesto

Capítulo 11 Apéndice A: Pruebas con Kmeans

En esté apéndice quedan registradas las pruebas realizadas con el método KMeans.

Para estudiar la viabilidad de aplicar este método, se realizaron varias pruebas con un dataset distinto (disponibles en un notebook de python para consulta).

Se determinó la cantidad óptima de clusters a utilizar mediante un codo de Jambú. Se basa en realizar una estimación orientativa para mejorar los resultados previsiblemente antes de realizar la clasificación.

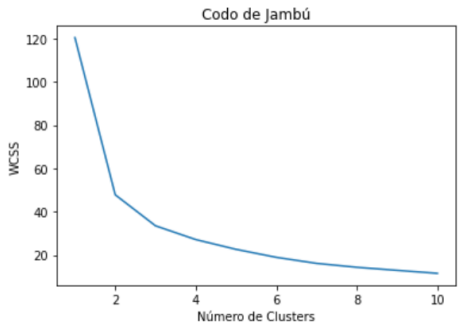
Codo de Jambú

Crearemos una función para determinar la cantidad de clústers óptimos a crear

```
[ ]  wcss = [] #creo una lista para almacenar los valores de wcss

     for i in range(1,11): #el rango máximo de clústeres +1 (10+1)
       kmeans=KMeans(n_clusters=i,max_iter=300) #función kmeans (los clústeres van cambiando)
       kmeans.fit(corredores_norm) #aplicamos a nuestra tabla
       wcss.append(kmeans.inertia_) #se van añadiendo los valores de wcss a la lista
```

Visualizamos el codo de Jambú

```
[ ]  plt.plot(range(1,11),wcss)
     plt.title("Codo de Jambú")
     plt.xlabel('Número de Clusters')
     plt.ylabel('WCSS')
     plt.show()
```



El valor óptimo de clústeres será cuando el valor de wcss deje de disminuir drásticamente (en nuestro caso son 3)

```
[ ]  clustering = KMeans(n_clusters=3,max_iter=300) #Aplicamos Kmeans con el número óptimo de clústeres
     clustering.fit(corredores_norm)

     KMeans(n_clusters=3)
```

```
[ ]
```

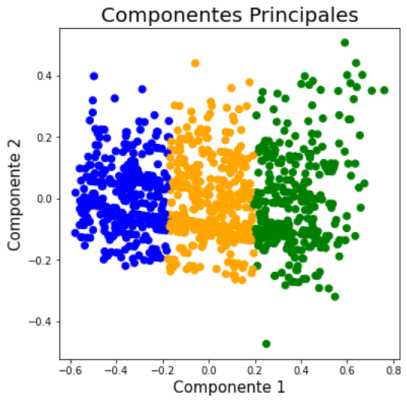
Con el fin de simplificar la visualización y el estudio, se realizó un PCA [18] (Análisis de Componentes Principales) a partir de las variables obtenidas de la realización del clustering. En la gráfica final se puede comprobar que se realizó la clasificación

correctamente asignando un cluster bien diferenciado correspondiente a cada dato.

Visualización de los clusters

```
[ ] from sklearn.decomposition import PCA

    pca = PCA(n_components=2)
    pca_corredores = pca.fit_transform(corredores_norm)
    pca_corredores_df = pd.DataFrame(data=pca_corredores,columns=['Componente_1','Componente_2'])
    pca_nombres_corredores = pd.concat([pca_corredores_df,corredores[['KMeans_Clusters']]],axis=1)

    pca_nombres_corredores
```

```
[ ] fig = plt.figure(figsize = (6,6))

    ax = fig.add_subplot(1,1,1)
    ax.set_xlabel('Componente 1', fontsize = 15)
    ax.set_ylabel('Componente 2', fontsize = 15)
    ax.set_title('Componentes Principales', fontsize = 20)

    color_theme = np.array(["blue","green","orange"])
    ax.scatter(x= pca_nombres_corredores.Componente_1, y= pca_nombres_corredores.Componente_2, c=color_theme[pca_nombres_corredores.KMeans_Clusters], s = 50)

    plt.show()
```



Guardar archivo final

```
[ ] corredores.to_csv('final_con_cluster.csv')
```

Enlace al repositorio

Las pruebas realizadas quedan registradas en el repositorio accesible mediante el siguiente enlace:

- **Pruebas KMeans:**

  https://drive.google.com/drive/folders/1RjcmDDpVL-2W_q6H1ePQo1etNQBAtNM7?usp=sharing

Apéndice B: Pruebas con Regresión Lineal

Se realizaron pruebas con el método de regresión lineal. Se pone a disposición de consulta un notebook de python donde se hizo uso de librerías como *sklearn* y *pydataset* para entender el funcionamiento del método de regresión lineal puesto en práctica con un dataset de prueba en un entorno controlado.

Enlace al repositorio

Las pruebas realizadas quedan registradas en el repositorio accesible mediante el siguiente enlace:

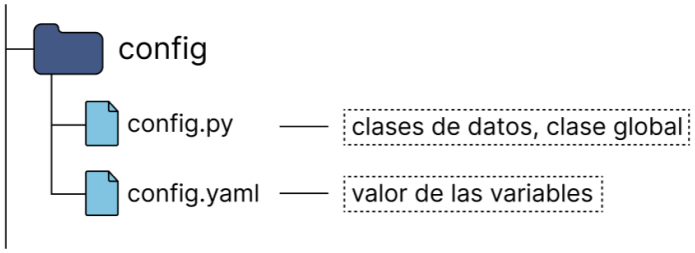
- **Pruebas Regresión Lineal:**

  https://drive.google.com/drive/folders/1lbBgZ2udgo2KFpsglFIH8z-z2KKtvpOT?usp=sharing

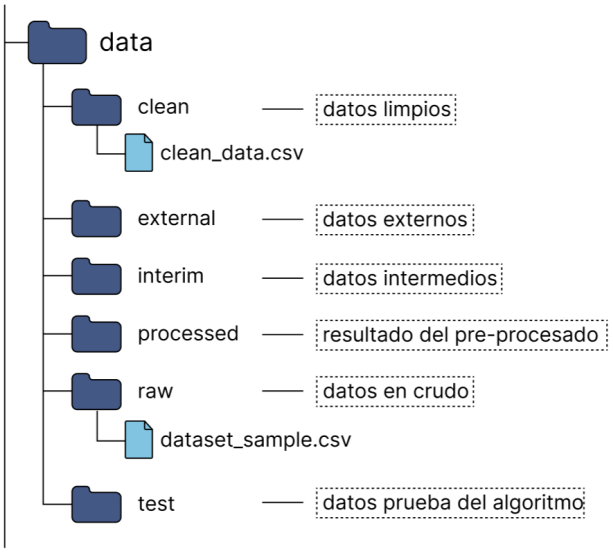
Capítulo 12 Apéndice C: Estructura del repositorio

Explicación de cada directorio

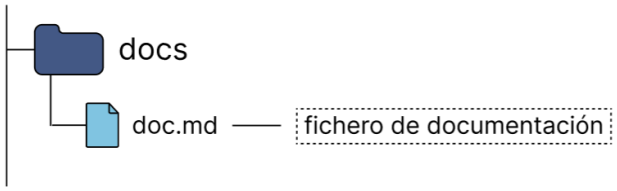
La carpeta de config contiene 2 ficheros de configuración de las variables globales del programa. Hay un fichero con extensión “py” que contiene las clases de datos y especifica el tipo de dato de cada variable (al final de este fichero se encuentra una clase global que engloba al resto de clases). También contiene un fichero “.yaml” que almacena los valores de las variables.



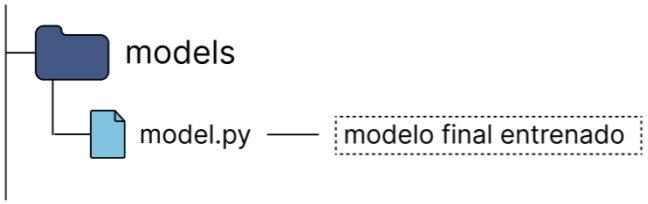
La carpeta data contiene los datos que utiliza el programa. La carpeta raw contiene los datos en crudo extraídos de la fuente original. La carpeta processed guarda los datos resultantes del preprocesado. La carpeta interim contiene los datos de procesos intermedios. La carpeta external almacena los datos externos al programa necesarios para el funcionamiento del código. La carpeta test se usa para almacenar los datos resultantes del testeo del algoritmo. Y la carpeta clean guarda los datos ya procesados (el dataset a utilizar).



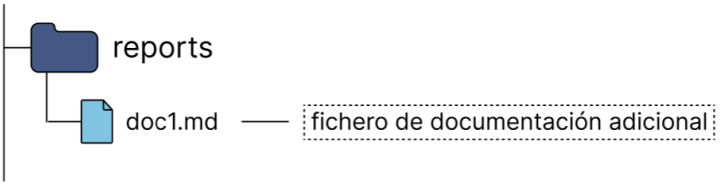
La carpeta docs guarda la documentación principal del programa en un fichero tipo “markdown”.



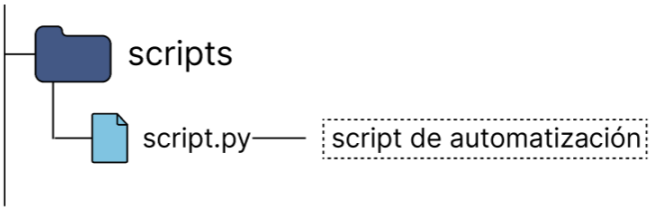
La carpeta models contiene el modelo final entrenado. Al almacenarlo de esta manera se evita entrenarlo para cada ejecución.



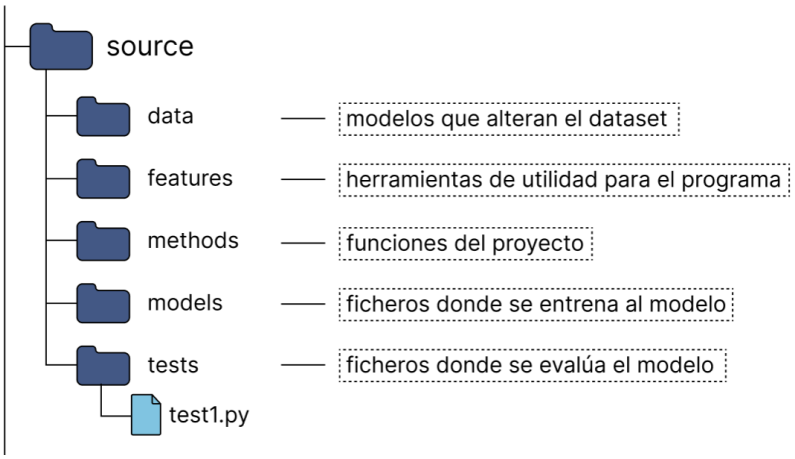
En la carpeta reports se almacenan los informes que representan la documentación adicional del programa.



La carpeta scripts contiene archivos ejecutables que ayudan a la automatización de tareas que realiza el programa.



Finalmente, la carpeta source contiene el código fuente del proyecto. En esta carpeta se encuentran todos los archivos necesarios para que funcione. Entre sus directorios está la carpeta data que contiene los ficheros donde se especifican los modelos que afectan al dataset. Features contiene herramientas que realizan acciones de utilidad para el programa. Methods contiene todas las funciones principales del programa. Models contiene los ficheros donde se entrena al modelo. Y test almacena los ficheros donde se evalúa este modelo entrenado.



Enlace al repositorio

El repositorio de GitHub que contiene todos los ficheros necesarios para el funcionamiento del programa se encuentra disponible mediante el siguiente link:

- **Repositorio GitHub:**

  https://github.com/raul-martin-dev/categorizacion-iniciativas-parlamentarias-parcan.git

Capítulo 13 Apéndice D: Resultados de la evaluación

13.1 Resultados: Métricas

```
-- Método de regresión logística: entrenamiento
> Comenzando el modelo de precisión...


-- Testeo de la precisión: 0.9814719575100543
-- Testeo del recall: 0.9803571428571428
-- Testeo del f1: 0.9801898648139682
-- Testeo del accuracy: 0.9803571428571428
```

13.2 Resultados: Validación cruzada

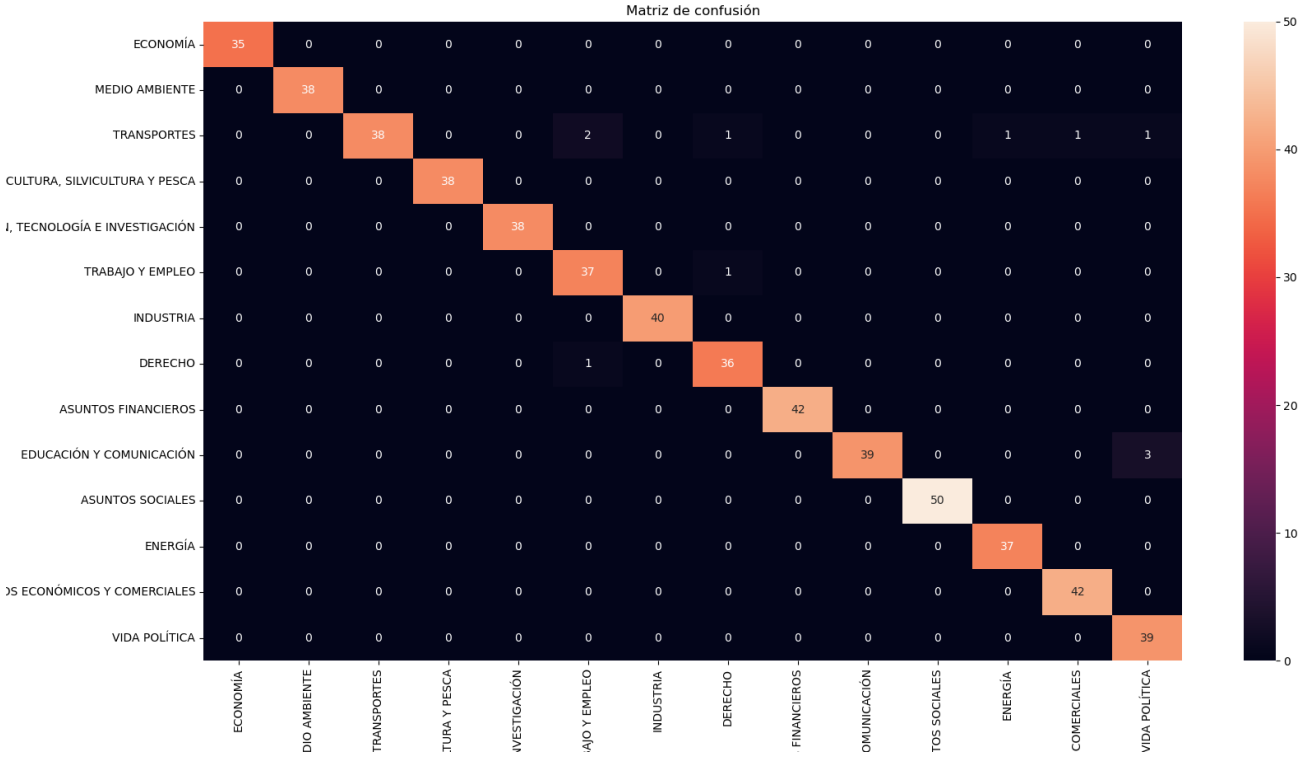
```
> Comenzando modelo de precisión 1...
> Testeando modelo...

-- Test 1:

-- Test precision: 0.9003752587991718
-- Test recall: 0.9
-- Test f1: 0.8981719068398472
-- Test accuracy: 0.9


Modelo de precisión 1 e informe completado! <

> Comenzando modelo de precisión 2...
> Testeando modelo...

-- Test 2:

-- Test precision: 0.8634944727492273
-- Test recall: 0.8571428571428571
-- Test f1: 0.8573210053037669
-- Test accuracy: 0.8571428571428571


Modelo de precisión 2 e informe completado! <

> Comenzando modelo de precisión 3...
> Testeando modelo...

-- Test 3:

-- Test precision: 0.8661481817607425
-- Test recall: 0.8678571428571429
-- Test f1: 0.8654780137083457
-- Test accuracy: 0.8678571428571429


Modelo de precisión 3 e informe completado! <
```

```
> Comenzando modelo de precisión 4...
> Testeando modelo...

-- Test 4:

-- Test precision: 0.8722750781194524
-- Test recall: 0.8642857142857143
-- Test f1: 0.8644136796252074
-- Test accuracy: 0.8642857142857143


Modelo de precisión 4 e informe completado! <

> Comenzando modelo de precisión 5...
> Testeando modelo...

-- Test 5:

-- Test precision: 0.8751936423969768
-- Test recall: 0.8785714285714286
-- Test f1: 0.8737961628029609
-- Test accuracy: 0.8785714285714286


Modelo de precisión 5 e informe completado! <

> Comenzando modelo de precisión 6...
> Testeando modelo...

-- Test 6:

-- Test precision: 0.8408130999435348
-- Test recall: 0.8428571428571429
-- Test f1: 0.8385899090852225
-- Test accuracy: 0.8428571428571429


Modelo de precisión 6 e informe completado! <

> Comenzando modelo de precisión 7...
> Testeando modelo...

-- Test 7:

-- Test precision: 0.8904867744199129
-- Test recall: 0.8857142857142857
-- Test f1: 0.8864697922343808
-- Test accuracy: 0.8857142857142857


Modelo de precisión 7 e informe completado! <
```

```
> Comenzando modelo de precisión 8...
> Testeando modelo...

-- Test 8:

-- Test precision: 0.8359318618231564
-- Test recall: 0.8392857142857143
-- Test f1: 0.8350909710787354
-- Test accuracy: 0.8392857142857143


Modelo de precisión 8 e informe completado! <

> Comenzando modelo de precisión 9...
> Testeando modelo...

-- Test 9:

-- Test precision: 0.8813220982172772
-- Test recall: 0.8821428571428571
-- Test f1: 0.8806710767007851
-- Test accuracy: 0.8821428571428571


Modelo de precisión 9 e informe completado! <

> Comenzando modelo de precisión 10...
> Testeando modelo...

-- Test 10:

-- Test precision: 0.8813159043521716
-- Test recall: 0.8821428571428571
-- Test f1: 0.8785945093803154
-- Test accuracy: 0.8821428571428571


Modelo de precisión 10 e informe completado! <
```

```
> Final report:

-- Maximum Accuracy 0.9
-- Minimum Accuracy: 0.8392857142857143
-- Overall Accuracy: 0.87
```

13.3 Resultados: Matriz de confusión



Bibliografía

1. *Tecnologías para hacer una administración pública más eficiente: https://www.directivosyempresas.com/internet/tecnologia/tecnologias-para-hacer-una-administracion-publica-mas-eficiente/*
2. *Algoritmos de clustering y aprendizaje automático aplicados a Twitter, por Eric-Joel Blanco-Hermida Sanz: https://upcommons.upc.edu/bitstream/handle/2117/82434/113257.pdf*
3. *Documentación de la Universidad Carlos III de Madrid, Regresión Lineal: http://halweb.uc3m.es/esp/Personal/personas/jmmarin/esp/GuiaSPSS/18reglin.pdf*
4. *Spanish Influenza Score (SIS): Usefulness of machine learning in the development of an early mortality prediction score in severe influenza: https://pubmed.ncbi.nlm.nih.gov/32798052/*
5. *Documentación de la Universidad Carlos III de Madrid, Regresión Logística: http://halweb.uc3m.es/esp/Personal/personas/jmmarin/esp/GuiaSPSS/18reglin.pdf*
6. *Redes Neuronales: https://www.ibm.com/docs/es/spss-modeler/SaaS?topic=networks-neural-model*
7. *International Journal of Disaster Risk Reduction, Twitter for disaster relief through sentiment analysis for COVID-19 and natural hazard crises: https://www.sciencedirect.com/science/article/pii/S2212420921000674*
8. *E. Sánchez-Nielsen, A. Morales, O. Mendo and F. Chávez-Gutiérrez, "SuDaMa: Sustainable Open Government Data Management Framework for Long-Term Publishing and Consumption," in IEEE Access, vol. 9, pp. 151841-151863, 2021, doi: 10.1109/ACCESS.2021.3127472.*
9. *Dataset crudo inicial: https://datos.parcan.es/dataset/iniciativas_tipope_anio2022_xlegislatura*
10. *Tesauro EuroVoc: https://eur-lex.europa.eu/browse/eurovoc.html?locale=es*
11. *SMOTE: https://machinelearningmastery.com/smote-oversampling-for-imbalanced-classification*
12. *TfidfVectorizer doc (sklearn): https://scikit-learn.org/stable/modules/generated/sklearn.feature_extraction.text.TfidfVectorizer.html*
13. *LogisticRegression doc (sklearn): https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LogisticRegression.html*
14. *Sklearn.metrics: https://scikit-learn.org/stable/modules/classes.html#module-sklearn.metrics*
15. *Validación cruzada (AWS): https://docs.aws.amazon.com/es_es/machine-learning/latest/dg/cross-validation.html*

16. *Sobreajuste: https://docs.microsoft.com/es-es/azure/machine-learning/concept-manage-ml-pitfalls*

17. *Matriz de confusión: https://www.juanbarrios.com/la-matriz-de-confusion-y-sus-metricas/*

18. *PCA: https://www.cienciadedatos.net/documentos/35_principal_component_analysis#:~:text=Principal%20Component%20Analysis%20(PCA)%20es,vez%20que%20conserva%20su%20informaci%C3%B3n.*

19. *Preln, a package for preprocessing text in spanish (Adrián Hernández Suárez y Raúl Martín Rigor): https://github.com/Adri-Hdez/Preln*