**UNIVERSIDAD AUTÓNOMA DE MADRID**
**ESCUELA POLITÉCNICA SUPERIOR**



**Grado en Ingeniería Informática**

# TRABAJO FIN DE GRADO

## Predicción de relevancia y controversia de propuestas ciudadanas en la plataforma de presupuestos participativos Decide Madrid

**Autor: José Antonio Peña Carrero**
**Tutor: Iván Cantador Gutiérrez**

**junio 2021**

**José Antonio Peña Carrero**
**Predicción de relevancia y controversia de propuestas ciudadanas en la plataforma de presupuestos participativos Decide Madrid**

**José Antonio Peña Carrero**

IMPRESO EN ESPAÑA - PRINTED IN SPAIN

*A todos los que han estado apoyándome en todo este tiempo.*

*Sólo podemos ver parte del futuro,*
*pero lo suficiente para entender*
*que hay mucho por hacer.*

*Alan M. Turing*

# Agradecimientos

En primer lugar, este trabajo no lo habría podido llevar a cabo sin la paciencia, dedicación y ayuda de mi tutor, Iván, que ha mostrado por mí, especialmente por orientarme en la labor.

Por otro lado, agradezco el apoyo de los compañeros durante cada uno de los cursos de la carrera, por todas las prácticas de cada curso y por todas las horas dedicadas en los laboratorios. Simplemente, gracias.

Agradezco finalmente a mis más íntimos amigos y a mi familia por apoyarme en las decisiones que he llevado a cabo, así como aportarme la ayuda necesaria en general durante la carrera y especialmente para lograr este trabajo, a pesar de no estar siempre en las mejores condiciones. Ha sido un duro recorrido, pero no lo hubiese podido llevar a cabo sin cada uno de vosotros.

# Resumen

Gracias a auge de las Tecnologías de la Información y de la Comunicación (TIC), en los últimos años han surgido diversas plataformas y redes sociales en las que la población puede manifestar su opinión acerca de diversos temas.

Además, con la aparición de las ciudades inteligentes (*Smart cities*), una gran y creciente cifra de ciudades se han preocupado por los problemas que actualmente acarrean, por lo que han puesto a disposición de sus vecinos nuevas herramientas tecnológicas en los que ellos mismos pueden proponer cambios o solventar problemas urbanos haciendo acopio de plataformas de participación ciudadana.

En el caso de Madrid, en septiembre de 2015 apareció la plataforma Decide Madrid, una web organizada junto con el ayuntamiento de la capital para poder llevar a cabo presupuestos participativos y propuestas ciudadanas, e intentar implementar una democracia más directa y mejorar la interacción *citizen-to-government*. A raíz de la observación de los primeros cuatro años de la plataforma, se ha podido recopilar información sobre los diferentes intereses de los ciudadanos mediante las propuestas redactadas, los comentarios publicados de los usuarios y los apoyos que recibían las propuestas.

El objetivo de este trabajo consistirá en analizar las diversas propuestas generadas durante los primeros cuatro años de la plataforma. Especialmente, el interés reside en aquellas propuestas que tienen relevancia o generan controversia en la población. Se realizará una previa estimación de dichas características, para posteriormente obtener los atributos que mejor identifiquen las propuestas como relevantes o controvertidas.

# Palabras clave

propuestas ciudadanas, participación ciudadana, aprendizaje automático, clasificación, e-gobierno, procesamiento del lenguaje natural, presupuestos participativos, Decide Madrid

# Abstract

Thanks to the rise of Information and Communication Technologies (IT), new online platforms and social networks have appeared in the last years, which let people express their opinion about several topics.

In addition, with the firsts formed Smart Cities, a large and growing amount of cities had been worried about the problems that suffer. That is the reason why some governments brought new tools to suggest changes or solve urban problems using citizen participation platforms.

In the case of the city of Madrid, in September 2015 appeared Decide Madrid, an online service organised by the Madrid city council to implement participatory budgets and citizen proposals, and also to try to implement more direct democracy, improving citizen-to-government interactions. Based on the observation of the first four years of the platform, information about different citizen interests have been collected through the written proposals, the published comments by users and the support given to the proposals.

This research work aims to analyse the diverse proposals generated during the first four platform years. Especially, the interest resides in those proposals which are relevant or generate controversy in the population. A preliminary estimation of these characteristics will be made to later obtain the attributes that best identify the proposals as relevant or controversial.

# Keywords

citizen proposals, citizen participation, machine learning, classification, e-government, natural languaje processing, participatory budgeting, Decide Madrid

# Índice

# Listas

## Lista de códigos



## Lista de ecuaciones



## Lista de figuras

# 1

# Introducción

Para introducir al lector sobre el interés en el estudio llevado a cabo en este proyecto, a continuación se expondrá una breve situación del contexto del problema abordado, seguida de una breve motivación para mostrar el interés de este trabajo.

Posteriormente, se expondrán los objetivos que se esperan lograr. Al final del presente documento se debatirá si se han cumplido los objetivos planteados, y en caso contrario las soluciones que se proponen para mejorar.

Para acabar este capítulo introductorio, se indicará una guía resumida del contenido desarrollado en esta memoria.

En el siguiente enlace se accederá al repositorio git en el que se encuentra el código utitlizado para realizar la práctica: [Enlace al repositorio git de la EPS]. Otros datos de interés se encontrarán en el siguiente enlace a Google Drive: [Enlace a los datos]

## 1.1. Contexto de Decide Madrid

La plataforma Decide Madrid [1] surgió en el año 2015 para fomentar la participación ciudadana utilizando las Tecnologías de la Información y de la Comunicación (TIC) para mejorar las relaciones entre los ciudadanos y el gobierno Citizen To Government (C2G) , donde los ciudadanos de Madrid pueden abrir debates, proponer mejoras en la ciudad, manifestar problemas, e incluso apoyar algunos proyectos a través de los presupuestos participativos que propone anualmente el ayuntamiento.

Por ejemplo, una de las decisiones que se pudo llevar a cabo en la plataforma fue la votación sobre la remodelación de Plaza España [2] en la cual, aunque se podía votar por otros medios, los vecinos mayoritariamente votaron mediante esta plataforma.

Respecto a los presupuestos participativos, los ciudadanos pueden presentar sus proyectos (tanto el ámbito de toda la ciudad como para los distintos distritos), los cuales se revisarán en función de su

[1] Decide Madrid, web: https://decide.madrid.es/

[2] Remodelación de Plaza España: resultados, web: https://decide.madrid.es/proceso/plaza-espana-resultados

viabilidad. Seguidamente, los ciudadanos podrán apoyar los proyectos en la fase de apoyo, en la que internamente se evaluarán de nuevo. Una vez evaluados, se publicarán los precios de los diferentes presupuestos, y finalmente la ciudadanía podrá votar aquellos que más les interesen. [3] Como ejemplo, uno de los proyectos aprobados en los presupuestos participativos de 2019 consistió en repoblar la flora y fauna en las grandes zonas verdes de la ciudad. [4].


**Figura 1.1:** Página de inicio del portal Decide Madrid (fuente: Decide Madrid)

En este trabajo se han analizado, en particular, las propuestas ciudadanas (en adelante se referenciarán simplemente como propuestas). Las propuestas, en general, presentan las siguientes características:

- Título: Es una breve descripción informativa de la propuesta.
- Resumen: Descripción algo más detallada de la temática de la propuesta. No puede superar los 200 caracteres.
- Texto: Es un campo opcional. Aporta una descripción completa del problema a manifestar, y permite utilizar texto enriquecido.
- Ámbito: Lista en la que aparecen todos los distritos de la ciudad de Madrid, además del valor 'Toda la ciudad'.
- Asignación de *tags* o temáticas de la propuesta. Se dan ciertos valores preseleccionados, pero se pueden incluir otros nuevos.
- Enlace a un vídeo externo (opcional).
- Imagen y documentación adicional.
- Ubicación de la propuesta en el mapa.
- Nombre y apellidos del creador de la propuesta.

[3] Decide Madrid - Presupuestos participativos, web: https://decide.madrid.es/presupuestos

[4] Decide Madrid - Presupuestos participativos, proyecto 14474, web: https://decide.madrid.es/presupuestos/presupuestos-participativos-2019/proyecto/14474

Las propuestas, una vez se hayan creado y publicado, pueden contener comentarios de otros usuarios, e incluso respuestas a otros comentarios por parte del usuario que haya creado la propuesta. Además de los comentarios, uno de los aspectos más interesantes son los votos que las propuestas publicadas pueden recibir, y se resume como la cantidad de apoyo que necesita una propuesta para que se lleve a cabo. Para lograr este objetivo, las propuestas han de ser votadas por un mínimo de el 1 % de la población de Madrid (dicha cantidad es de 27.662 votos en el momento en el que se ha realizado este trabajo [5]).

Las propuestas que lleguen a alcanzar ese objetivo, podrán pasar a la fase de votación. En esta fase se podrán votar las propuestas que superen el umbral del 1 % de los votos en la plataforma, así como otras propuestas que el ayuntamiento considere relevantes. Una vez acabada la fase de votación, se podrá llevar a cabo su aplicación por parte del ayuntamiento siempre y cuando cuenten con la mayoría de los apoyos. Por ejemplo, el año 2017, dos propuestas se aprobaron en esta fase con más de un 90 % de los votos a favor [6]. Dichas propuestas son Madrid 100% sostenible y Billete único para el transporte público.


**Figura 1.2:** Vista de la información de una propuesta (fuente: Decide Madrid)

[5] Decide Madrid: Información del portal (propuestas), web: https://decide.madrid.es/mas-informacion#proposals

[6] Decide Madrid: Primeras votaciones, web: https://decide.madrid.es/primera-votacion-ciudadana-resultados

## 1.2. Motivación

Una vez introducida la plataforma Decide Madrid, con la información que se obtiene de las propuestas, se pretende investigar en aquellas que tengan un mayor número de apoyo, y de ahí obtener información adicional que permita saber si otras propuestas de similares características pueden tener un gran número de apoyos. De igual forma se pueden obtener valores que indiquen qué características muestran un alto grado de debate entre las propuestas, es decir, que hayan opiniones opuestas por parte de los usuarios.

Por tanto, las métricas que se van a definir serán relevancia y controversia en base a ciertos valores de las propuestas, que se detallarán más adelante. El cálculo de estas métricas y su posterior uso para comparar las propuestas es uno de los objetivos principales de este Trabajo Final de Grado (TFG) .

## 1.3. Objetivos

Si bien es cierto que la implantación de una democracia participativa motiva a los ciudadanos y ciudadanas a crear un espacio de diálogo en el que debatir aspectos a mejorar sobre los lugares y actividades que comparten en la ciudad, también se generan muchas y muy diversas opiniones. Los usuarios también crean bastantes propuestas de diversa índole, donde no siempre se ve su aplicación tan clara o necesaria por la mayoría de los vecinos y vecinas.

Uno de los objetivos de este trabajo es encontrar **en qué puntos están de acuerdo los ciudadanos y ciudadanas de Madrid**. El problema principal de este trabajo consistirá en evaluar la relevancia y controversia de las propuestas llevadas a cabo desde septiembre de 2015 hasta agosto de 2019. El problema se abarcará con modelos de clasificación. Por lo tanto, se estudiarán qué indicadores o características aportan una mayor estimación de relevancia o controversia para cada propuesta, para luego analizar atributos de diversos conjuntos de datos y poder clasificar cada propuesta de la mejor forma posible. Este conjunto de atributos se basa en otros valores (distritos, categorías o *tags* asignados a cada propuesta, además de algunos atributos obtenidos relativos al texto del *summary* y del contenido de las mismas).

Una vez obtenidos los resultados de las clasificaciones, se debatirá si se ha obtenido una buena aproximación del problema con el material y las técnicas empleadas en este TFG , así como plantear posibles soluciones alternativas.

## 1.4. Organización de la memoria

El presente documento se ha dividido en varias secciones:

Primeramente, se ha evaluado el **estado del arte** en el contexto de las *Smart Cities*, los presupuestos participativos y las democracias electrónicas [1], además de aproximar una clasificación similar a otro problema de clasificación de temáticas ciudadanas [2].

Seguidamente, se detalla el **diseño** donde se plantean las decisiones que se han llevado a cabo para elaborar este TFG (esquema y flujo general, datos con los que se ha trabajado, selección de clases y de características con las que clasificar, decisión de los problemas de clasificación a realizar y forma de evaluar los resultados).

A continuación se detalla el **desarrollo** del proyecto, explicando las diversas herramientas empleadas y lenguajes utilizados, así como librerías utilizadas y modelos de clasificación usados.

Finalmente, se comentan las **pruebas** realizadas para mostrar los **resultados** alcanzados, lo que da lugar a un posterior debate sobre las **conclusiones** obtenidas, seguido del **trabajo futuro** que se puede realizar.

# 2

# Estado del Arte

En este apartado se expondrá la actualidad del contexto que envuelve la participación ciudadana en las *Smart Cities* y en los presupuestos participativos. Además, se comentará las discusiones y los problemas generados con este método democrático de propuestas ciudadanas en otras plataformas o redes sociales.

## 2.1. Panorama actual de las Smart Cities

El concepto de *Smart City* se resume en obtener datos e información relevante a un área, principalmente urbana, mediante el uso de las TIC . La recolección de estos datos suele ser con ayuda de diferentes tipos de sensores (por ejemplo, cámaras de vídeo para detectar tráfico en las calles y avenidas principales) o redes sociales. En este último caso, la obtención de la información en base a las publicaciones suele ser más compleja que mediante los propios sensores, ya que los ciudadanos pueden manifestar problemas muy específicos u otros que no existieran previamente, a través de estos medios. [3]

Los datos que se suelen tratar son de diversa índole, pero relacionados con la convivencia en entornos urbanos (medio ambiente, movilidad, economía, etc.). En el caso a estudiar en este trabajo, interesa la temática política (*smart governance*).

Una de las primeras ciudades europeas en implementar este modelo es **Amsterdam**, mediante el portal Amsterdam Smart City [1]. La ciudad ha implementado en el año 2009 este modelo inteligente de ciudad, posicionándose como una de las más exitosas *Smart Cities* de Europa. [4] En la plataforma se llevan a cabo debates en colaboración con instituciones todo tipo, actuando como foros de discusión sobre diversos temas de la ciudad. Además, las ideas más interesantes pueden llegar a convertirse en proyectos, los cuales se pueden aplicar a pequeña escala, y se podrán reproducir en otras áreas si posteriormente resultan favorables. Las ideas que se agrupan en este concepto son **movilidad**, **energía**, **economía circular** o **educación**, entre otras. [5]

[1] Amsterdam Smart City, web: https://amsterdamsmartcity.com/

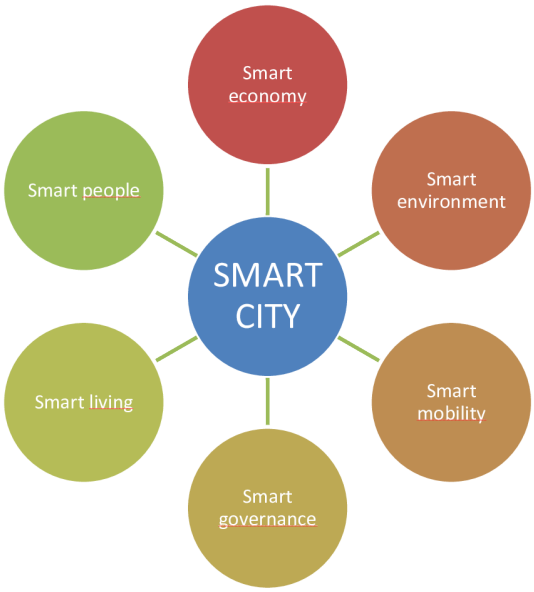



**Figura 2.1:** Temas principales de las Smart Cities. Fuente: https://www.researchgate.net/figure/Dimensions-of-smart-cities_fig1_304570582

Por otro lado, se encuentra la ciudad de **Barcelona**, en la que además de remodelar parte de los espacios industriales para albergar empresas tecnológicas (como es el caso del distrito 22@), ha sido pionera en proyectos como *MareNostrum* ( Barcelona Supercomputing Center Centro Nacional de Supercomputación (BSCCNS) ) [2], además de posicionándose como una de las mejores ciudades para emprender de Europa. [6]

La ciudad hace uso de la web de participación ciudadana [3] en colaboración con el ayuntamiento para realizar votaciones de proyectos de sostenibilidad, movilidad y urbanismo, así como llevar a cabo presupuestos participativos, entre otros. De esta forma, se ha posicionado como una de las *Smart Cities* más innovadora del continente europeo.

## 2.2. Plataformas de presupuestos participativos

Los presupuestos participativos permiten a los ciudadanos apoyar proyectos urbanos de forma más activa y directa, mejorando la toma de decisiones y las relaciones C2G . [7]

Los primeros presupuestos participativos tuvieron lugar en la ciudad brasileña de Porto Alegre [8]. Esta decisión fue tomado por el partido de carácter progresista de la ciudad, con la finalidad de reducir las desigualdades de la misma, en donde se decidió asignar entre el 15 y el 25 % del presupuesto total a proyectos de asignación participativa. Una de las decisiones principales fue dividir el municipio en varios distritos, delegando así parte de la toma de decisiones del ayuntamiento.

[2] Barcelona Supercomputing Center, web:https://www.bsc.es/marenostrum/marenostrum/

[3] Decidim Barcelona, web: https://www.decidim.barcelona/processes/PressupostosParticipatius

Desde entonces, ciudades como París [4] o Nueva York (donde varios distritos tienen su propia web de presupuestos participativos [5]) han seguido este modelo de participación ciudadana y lo decidieron aplicar a sus ciudades, ayudándose de las nuevas tecnologías para facilitar el proceso.

Como se ha comentado anteriormente, en las *Smart Cities* el proceso de mejora de las ciudades tiene que ver más con la gestión eficiente de diversas características de la ciudad. Los presupuestos participativos, por lo tanto, mejoran la gestión de la toma de decisiones, y pueden llevarse a cabo fácilmente con ayuda de las TIC , donde influyen en el área de *smart governance*.

El proceso de decisiones en este tipo de participación ciudadana emplea una estrategia ascendente *bottom-up*, ya que son los ciudadanos los que apoyan cambios estructurales que ellos mismos pueden manifestar, que más tarde el ayuntamiento o las delegaciones correspondientes podrán realizar.

## 2.3. Clasificación de temáticas ciudadanas mediante tweets

Con el uso de redes sociales como Twitter [6], los usuarios pueden manifestar su opinión y hacérsela llegar a otros usuarios. Una aproximación al trabajo desarrollado en este TFG es la *Clasificación de tweets en el contexto C2G* [2], en donde se ha aplicado un problema de Machine Learning - Aprendizaje Automático (ML) para clasificar un conjunto de tweets en los que se menciona a la cuenta del ayuntamiento de Madrid (@Madrid) con ciertos atributos obtenidos mediante Natural Processing Language - Procesamiento del Lenguaje Natural (NPL) . Cada tweet se clasificó manualmente en categorías como quejas, noticias, peticiones o propuestas, entre otras (un total de 10 problemas binarios de clasificación).

En este caso, el uso de NPL ha permitido obtener atributos basados exclusivamente en el texto de los tweets. Los atributos obtenidos se divididen en dos características lingüísticas: léxicas (número de caracteres, palabras, signos empleados, etc.) y gramaticales (número de nombres, nombres propios, verbos, adjetivos, pronombres, etc.). Los atributos gramaticales se han obtenido gracias al software de Stanford CoreNPL [9], en donde se ha procesado el lenguaje en español.

Una vez obtenidos los distintos valores para cada atributo del problema planteado, se han utilizado diversos modelos ML de clasificación. Con los resultados obtenidos, se concluye que un conjunto reducido de usuarios mencionados en los tweets discriminan muy bien ciertas categorías de tweets, así como ciertas palabras, especialmente en tweets categorizados como quejas.

[4] Paris - Budget participatif, web: https://budgetparticipatif.paris.fr/bp/
[5] New York City District 8, web: https://pbstanford.org/nyc8/knapsack
[6] Twitter, web: https://www.twitter.com

# 3

# Diseño

En este apartado se comentarán las decisiones que se han llevado a cabo para realizar el TFG .

## 3.1. Esquema general y flujo de trabajo

A continuación se presentará el flujo del trabajo realizado, detallando en alto nivel los pasos realizados:

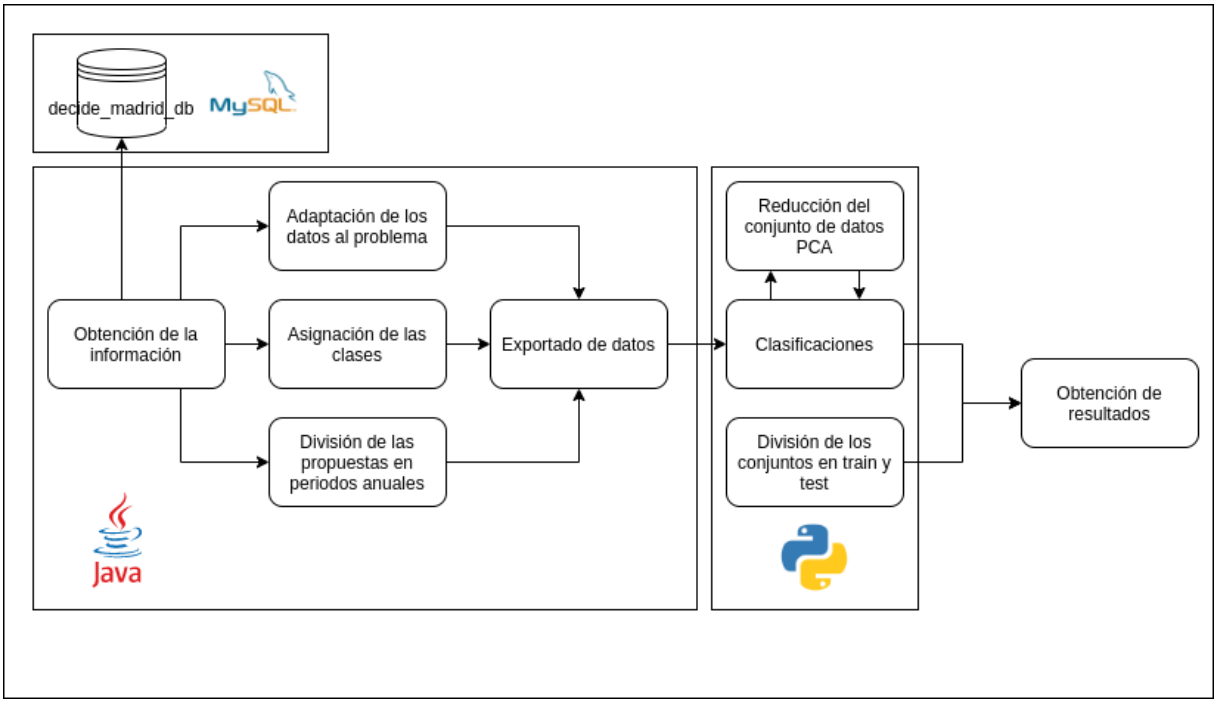



**Figura 3.1:** Esquema de alto nivel en el que se muestra el proceso de la obtención de los resultados.
Fuente de las imágenes: MySQL logo, Java logo, Python logo

**Obtención de los datos**: La primera parte de la investigación ha sido la obtención de los datos. En ese apartado se indicarán las fuentes y el contenido de la información que se ha considerado relevante para predecir la relevancia y controversia de las propuestas del portal Decide Madrid.

**Adaptación de los datos al problema**: Tras obtener la información de la base de datos, se ha modificado la información para adaptarla al problema, pasando ciertos valores de atributos categóricos a numéricos binarios.

**Asignación de clases**: Para poder poner clasificar las propuestas en relevantes o controvertidas, previamente de les ha de asignar una clase de relevancia y controversia a cada una de las propuestas a analizar. Se han definido unas métricas con las que poder evaluar las propuestas en estas dos clases, y posteriormente se ha introducido una función que evalúa dichas características. Las clasificaciones serán binarias (clases con valor 0 o 1) y multiclase (en cuatro rangos de valores).

**División de las propuestas en periodos anuales**: Para no sobreajustar los datos a la hora de clasificar, se ha decidido dividir el conjunto de las propuestas en cuatro periodos anuales de forma que entrenarán propuestas lo más diversas posibles para obtener así una clasificación más imparcial.

**Exportado de datos**: Debido a que se utilizan diferentes lenguajes para realizar diferentes tareas, se han de formatear la información para que sea perfectamente legible y así llevar a cabo las clasificaciones. Se ha optado por utilizar los formatos ARFF [1] y CSV, ya que son perfectamente compatibles con los datos a tratar y hay librerías que facilitan su lectura.

**Reducción del conjunto de datos**: Mediante el análisis de Principal Component Analysis - Análisis de componentes principales (PCA) se han obtenido los atributos más relevantes de cada conjunto de atributos, con el fin de reducir la dimensionalidad del problema a las características fundamentales.

**División de los conjuntos de entrenamiento y test**: Se han seguido un total de cuatro estrategias que implican validar las clasificaciones con propuestas tanto del mismo set como de distintos sets.

**Clasificación**: Se han utilizado varios modelos de clasificación con cada problema (relevancia y controversia, clasificación binaria y multiclase).

**Obtención de resultados**: Se han clasificado los datos de entrenamiento y se han validado con los valores de las clases del conjunto test. Una vez obtenida la matriz de confusión con los aciertos de cada valor de clase, se ha obtenido el número de aciertos total y de cada clase, para así obtener una mayor precisión y poder decidir qué valor de la clase resulta más fácil predecir, además de usar métricas como la media aritmética y la media geométrica para interpretar las clasificaciones realizadas.

**Discusión sobre los resultados obtenidos**: Una vez se organicen los datos, se discutirá sobre qué modelos han clasificado mejor, por qué los atributos obtenidos con PCA son más relevantes, o cómo poder mejorar las clasificaciones.

[1] Attribute-Relation File Format (ARFF), descripción: https://www.cs.waikato.ac.nz/ml/weka/arff.html

## 3.2. Adquisición de datos

### 3.2.1. Obtención de los datos

La información acerca del contenido de las propuestas de Decide Madrid se ha obtenido, originalmente, del servicio de Datos Abiertos que aporta el ayuntamiento de Madrid [2]. Se encuentran multitud de datos e información relevante obtenida por la ciudad (datos de tráfico, censos de población, etc.). Los datos de las propuestas se encuentran actualizados diariamente, con multitud de campos disponibles a rellenar para cada propuesta [3]. Para este proyecto se han obtenido algunos campos de este conjunto de datos, para las propuestas entre el 1 de Septiembre de 2015 hasta el mismo día del año 2019, lo cual equivale a un total de 21.737 propuestas.

Ya que la distribución de las propuestas durante los cuatro años no es de la misma forma en todos los periodos, se ha decidido dividir las propuestas en cuatro periodos anuales (desde septiembre hasta agosto de cada año) para separar las propuestas de cara a la clasificación y obtener resultados más realistas. En la figura 3.2 se muestra una gráfica en la que se comparan las propuestas publicadas durante los cuatro años de estudio, indicando el número de propuestas publicadas por mes.

Como se puede observar, durante el primer periodo anual de la plataforma se publicaron la mayor parte de las propuestas (algo más de 10.000 propuestas). Además, mientras que en los meses de otoño e invierno se publican gran parte de las propuestas, en los meses de verano desciende esta cifra, coincidiendo generalmente con el periodo vacacional.

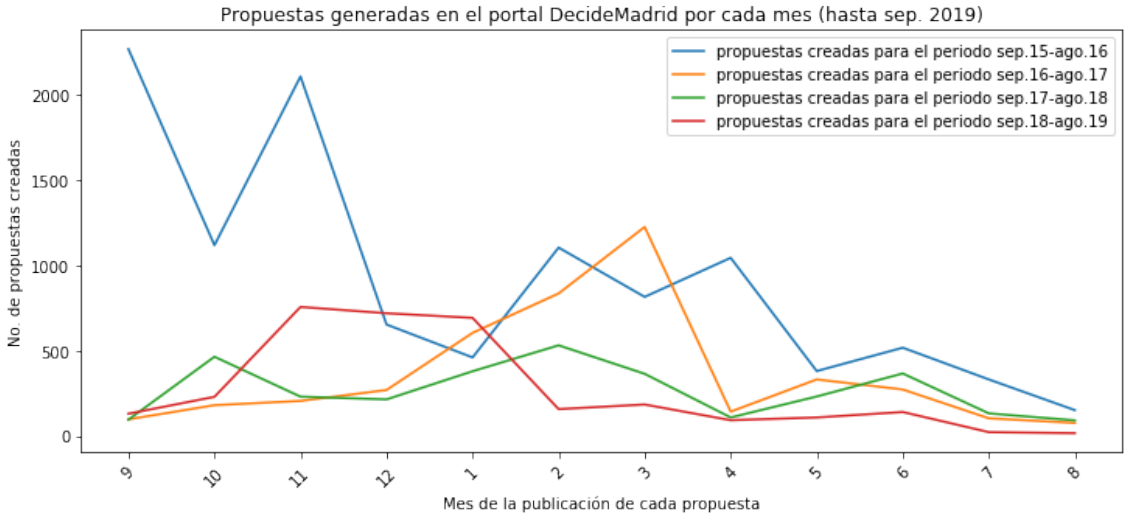



**Figura 3.2:** Distribución anual de las propuestas en Decide Madrid para los años 2015-2019

[2] Portal de datos abiertos - Madrid, web: https://datos.madrid.es/

[3] Propuestas ciudadanas - Portal Decide Madrid, enlace de descarga de datos de las propuestas: https://datos.madrid.es/egob/catalogo/300114-10434614-madrid-decide.csv

Posteriormente, debido a la baja precisión de información obtenida de los datos crudos de la plataforma (i.e. falta de información o precisión por parte de los usuarios que crearon las propuestas, como el distrito o la categoría de la misma), se realizó una labor de **procesado de los datos de las propuestas**, con el fin de crear una base de datos más acorde y veraz con su contenido. Dicho trabajo queda recogido en [10]. En dicho caso, se realizó una búsqueda de los nombres de calles, avenidas, paseos y demás lugares que aparecen en el contenido de las propuestas para así asignar una ubicación mucho más precisa del distrito al que pertenecen. La base de datos resultante [4] contiene la información relevante que se ha utilizado a lo largo del trabajo. Las tablas más relevantes de la base de datos resultante se resumen en las siguientes:

***proposals*** Contiene información de las propuestas. Para cada propuesta contiene principalmente su id, el título, el código de la propuesta, el enlace a la misma, el texto de resumen (*summary*), el texto del contenido de la propuesta, número de apoyos y número de comentarios, entre otros.

***proposal_categories*** Contiene para cada propuesta las categorías asignadas y su ponderación (las categorías vienen detalladas en la tabla *categories*).

***proposal_districts*** Contiene para cada propuesta los distritos que le corresponden y su ponderación (los distritos vienen detalladas en la tabla *districts*, aunque básicamente incluyen los 21 distritos de Madrid mas un valor que implica toda la ciudad).

***proposal_topics*** Esta tabla tiene relación con *proposal_categories*, ya que los *topics* en este caso son valores que detallan más el contenido de una categoría. Los detalles de cada uno de los 325 *topics* se detallan en la tabla homónima, en donde hacen referencia a la categoría con la que se relacionan (*topics*).

***proposal_comments*** Detalle de los comentarios de todas las propuestas, haciendo referencia a su id de propuesta, id del comentario de respuesta (si este valor es -1 significa que es un comentario raíz de la propuesta), y los votos positivos y negativos que contenga. Aunque en este trabajo no se han tenido en cuenta el contenido de los comentarios de las propuestas para analizar su relevancia o controversia de forma directa, sí que ha servido para calcular una métrica de controversia [10] que se explicará más adelante.

Tras obtener de esta forma parte de la información de las propuestas, se decidió realizar, con ayuda de Stanford CoreNPL [9], un análisis del texto de las propuestas. De forma similar al problema del análisis de temáticas ciudadanas mediante tweets [2], ha sido preferible analizar ciertos términos del texto en vez de aplicar el algoritmo *tf-idf*, ya que muchas de las propuestas contienen **poco texto**, e incluso la longitud del resumen es relativamente corta en bastantes casos (en gran parte de las propuestas el resumen no supera los 200 caracteres por delimitaciones de la plataforma). En la sección 4.2 se detallan mejor los atributos obtenidos.

[4] Citizen participation in electronic participatory budgetin, web:http://ir.ii.uam.es/egov/

### 3.2.2. Transformación de los datos

Ya que los valores categóricos del conjunto de datos son *strings* (distritos, categorías y *topics*), se han realizado las modificaciones necesarias para transformarlos **en atributos categóricos binarios**:

- 22 atributos para los distritos: Se utilizan indicadores para cada uno de los 21 distritos, además del indicador de todal la ciudad (etiquetado con el nombre Ciudad).
- 30 atributos de categorías: Cada atributo corresponde con su correspondiente categoría.
- 325 atributos de *topics*: Indican si la propuesta ha sido etiquetada con uno de los valores de *topics* descritos (que, a su vez, varios valores de *topic* se asignan a una categoría en concreto, e incluso comparten nombre).

De esta forma se marcarán como 1 los atributos de las propuestas en los que en la base de datos se indique que a dicha propuesta le corresponde ese valor y el resto de valores se mantendrán en 0 (de forma similar a la codificación *One Hot* [5] o como se codifican las estaciones a las que pertenece un mes en el ejemplo incluido en [11]). En el caso de los atributos del texto de las propuestas, los valores se mantendrán **como numéricos**.

## 3.3. Detalle de los problemas de clasificación

En la sección de introducción de este documento se han introducido los problemas de clasificación que se iban a abordar en este trabajo. Para las clasificaciones de dichas características, se han tenido en cuenta varios factores:

- Las propuestas más relevantes son aquellas que generalmente tienen muchos apoyos. También son relevantes las propuestas que en un periodo relativamente corto de tiempo pueden adquirir un número considerable de apoyos (no muy elevado, pero sí interesante de observar). Los datos de la plataforma, por contra, no dan un detalle significativo de cómo estimar este valor temporal puesto que solamente se data la fecha de creación como atributo temporal, por lo que la evaluación de esta métrica se ha basado en el **número de apoyos** y, en menor medida, el **número de comentarios**.
- La controversia de las propuestas se medirá con un valor que tenga en cuenta el debate que se genere en las mismas. Por lo tanto, un factor a tener en cuenta serán los **comentarios de las propuestas**, ya que es el espacio en el que los usuarios de la plataforma dan su opinión, aportan distintos puntos de vista o manifiestan su descontento. El valor de controversia dependerá en gran factor de las propuestas que tengan mucha actividad en los comentarios, pero también se puede tener en cuenta que las propuestas controvertidas han de tener suficiente apoyo para que el debate que se genere sea de interés general, y que no una discusión de una propuesta poco deseable en la que solo aporten su opinión pocos usuarios. Por lo tanto el **número de apoyos** también se tendrá en cuenta.

En la siguiente sección se muestran las estimaciones que se han asignado a cada propuesta para las clases relevancia y controversia.

[5] What is One Hot Encoding? Why and When Do You Have to Use it?, web: https://hackernoon.com/what-is-one-hot-encoding-why-and-when-do-you-have-to-use-it-e3c6186d008f

## 3.4. Selección de clases y características

Ya que las propuestas no tenían previamente asignado un valor o indicador de relevancia y clasificación explícito, en esta sección se detallan los valores que se han tenido en cuenta para asignar los valores de las clases.

Para empezar, y como ya se ha comentado, se han asignado cuatro tipos de clases:

- Clasificación de relevancia en una propuesta.
  - Clasificación binaria (0 - no relevante, 1 - relevante).
  - Multiclasificación con 4 valores (0 - no relevante, 1 - poco relevante, 2 - relevante, 3 - muy relevante).
- Clasificación de controversia en una propuesta.
  - Clasificación binaria (0 - no conrovertida, 1 - controvertida).
  - Multiclasificación con 4 valores (0 - no controvertida, 1 - poco controvertida, 2 - controvertida, 3 - muy controvertida).

Los valores de relevancia y controversia asignados a cada propuesta se han obtenido mediante las ecuaciones 3.1.

$$\begin{aligned} relevance_{p_i} &= 0{,}6\frac{num\_votes_{p_i}}{max_{p_{i'}}(num\_votes_{p_{i'}})} + 0{,}4\frac{num\_comments_{p_i}}{max_{p_{i'}}(num\_comments_{p_{i'}})} \\ controversy_{p_i} &= 0{,}1\frac{num\_votes_{p_i}}{max_{p_{i'}}(num\_votes_{p_{i'}})} + 0{,}1\frac{num\_comments_{p_i}}{max_{p_{i'}}(num\_comments_{p_{i'}})} \\ &\quad +0{,}8\frac{aggregation_{p_i}}{max_{p_{i'}}(aggregation_{p_{i'}})} \end{aligned} \tag{3.1}$$

Para las ecuaciones 3.1, $p_i$ indica la propuesta i-ésima, $num\_votes_{p_i}$ y $num\_comments_{p_i}$ indican, respectivamente, el número de votos y de comentarios de la propuesta i-ésima.

El valor $aggregation_{p_i}$ es una estimación de la controversia que se obtiene de hacer la media aritmética de tres valores relacionados con los comentarios de las propuestas, como aparece en [10].

Con más detalle, se comentan los valores de las tres métricas comentadas en las ecuaciones 3.2.

$$\begin{aligned} controversy_1(p) &= \sum_{c \in comments(p)} lenght(c) \\ controversy_2(p) &= 1 + min(pos(p), neg(p))\frac{min(pos(p), neg(p))}{max(pos(p), neg(p))} \\ controversy_3(p) &= \sum_{n=1}^{depth(p)} H(width(p,n) \geq n) + \frac{1}{1 + \|comments(p)\|} \end{aligned} \tag{3.2}$$

En la primera ecuación, se mide la longitud de los comentarios de una propuesta, de forma que será mayor cuantos más comentarios haya y mayor longitud tengan. Por otro lado, la segunda ecuación mi-

de el balance que hay entre los votos positivos y negativos de todos los comentarios de una propuesta, permitiendo dar una idea de cómo de polarizadas son las opiniones de los usuarios en base a los comentarios. La función obtiene la suma total de votos positivos ($pos(p) = \sum_{c \in comments(p)} positiveVotes_c$) y de votos negativos ($neg(p) = \sum_{c \in comments(p)} negativeVotes_c$), donde luego se comparan mediante las funciones $min()$ y $max()$. Por lo tanto, cuanto mayor número de votos obtengan los comentarios y más balanceados estén (i.e. similar número de votos positivos y negativos), se obtendrá un mayor valor de controversia. Por último, la tercera ecuación mide la forma en la que se desarrolla el árbol de comentarios de las propuestas, utilizando las función *Heaviside* o escalón unitario ($H(x) = 0$ si $x < 0$ y $H(x) = 1$ si $x \geq 0$), la función $depth(p)$, la cual mide la profundidad de los comentarios de una propuesta $p$, y la función $width(p, n)$, que mide la cantidad de comentarios que tiene una propuesta $p$ al nivel de profundidad $n$ del árbol de comentarios generado; por lo que esta métrica será mayor cuantas más 'contestaciones' tengan los comentarios de una propuesta, o, dicho de otro modo, por cuantos más debates se abran por cada comentario.

En la imagen 3.3 se muestran tres pares de ejemplos de árboles de comentarios que ayudan a entender las tres métricas expuestas.

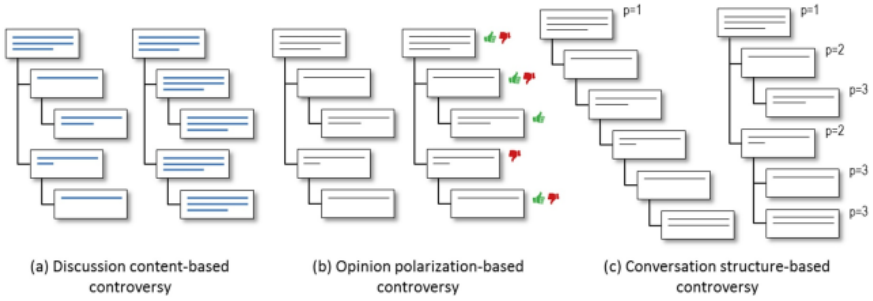



**Figura 3.3:** Muestra de tres pares de ejemplos en donde el conjunto de comentarios de la izquierda tendría menor controversia que el conjunto de la derecha, según la métrica correspondiente.

Por tanto, el valor $aggregation$ será la media ponderada de los tres valores, y su valor estará acotado entre 0 y 1, ya que cada estimación parcial de la controversia se dividirá entre el máximo valor obtenido para todas las propuestas (ver ecuación 3.3).

$$controversy(p) = \frac{1}{3} \sum_{i=1}^{3} \frac{controversy_i(p)}{max_{p'} controversy_i(p')} \in [0, 1] \tag{3.3}$$

Una vez entendidas todas las métricas que permiten el cálculo de las clases relevancia y controversia, se asigna el valor del mismo. En la clasificación binaria, la estrategia utilizada para clasificar con valor 0 o 1 dependerá del valor resultante de la función aplicada para medir la clase, es decir, si supera o no el valor umbral (la **mediana** en este caso) de los valores totales de todas las propuestas. Para las clasificaciones multiclase, el proceso será similar, en donde los valores umbrales serán los **cuartiles** $Q_1$**,** $Q_2$ **y** $Q_3$.

Cabe añadir que, al ser el conjunto de atributos bastante grande (22 + 30 + 325 + 9 atributos), se ha optar por realizar un análisis de PCA para reducir la dimensionalidad del problema, obteniendo atributos más relevantes para predecir las clases relevancia y controversia mientras no se pierde en precisión al clasificar. En el capítulo de pruebas se detallarán los valores obtenidos.

## 3.5. Modelado de los problemas de clasificación

Para este trabajo se ha empleado la **clasificación**, ya que al tener demasiados atributos categóricos sería mejor alternativa que realizar una regresión. Además, siete de los nueve atributos de texto se distribuyen mediante la ley de Zipf [6], lo cual es bastante habitual al trabajar con textos (usos de determinadas palabras, tamaños de los documentos, etc.); mientras que prácticamente ningún atributo empleado presenta una distribución normal o distribución gaussiana. En el apéndice A se pueden ver con mejor detalle las distribuciones del número de votos y número de comentarios de las propuestas. Las distribuciones de los atributos textuales se muestran en el apéndice B, donde se detallan varias gráficas que permiten visualizar fácilmente su distribución. Comentando brevemente la distribución de los atributos de texto, la distribución de las propuestas por longitud del resumen se comporta de una forma particular, en donde hay muchas propuestas que se acumulan en torno al valor 200 (límite de texto de las propuestas), pero también hay valores posteriores (que siguen una continuación de una posible ley de Zipf). Esto puede deberse a que, hasta una fecha dada, no había límite de caracteres en el resumen, y posteriormente añadieron esta restricción.

Además, la decisión de llevar a cabo varios tipos de clasificación (binaria y multiclase) han permitido obtener varios beneficios en cada uno de los casos. Por ejemplo, el más claro beneficio en las clasificaciones binarias es la **facilidad de clasificar cada propuesta con solamente dos valores posibles** (sobre todo teniendo en cuenta que la distribución, al ser mediante la ley de Zipf, muchos valores estarán más próximos a clasificarse con valor 0). Por otro lado, las clasificaciones multiclase permiten clasificar relativamente bien las propuestas que son o muy relevantes o controvertidas (con el máximo valor posible de clasificación) o las que precisamente no lo son (con el mínimo valor posible de clase, respectivamente). Además, con los modelos Linear Discriminant Analysis - Análisis Discriminante Lineal (LDA) o Quadratic Discriminant Analysis - Análisis Discriminante Cuadrático (QDA) se llegan a mostrar las asignaciones de las clases en los problemas multiclase (ya que en los problemas binarios simplemente se muestran reescalados en una recta). En la sección de resultados D se muestran las gráficas de los datos tras realizar LDA con las 2 mejores combinaciones lineales de atributos para la clasificación multiclase.

[6] Zipf's law, web: https://mathworld.wolfram.com/ZipfsLaw.html

## 3.6. Metodología de evaluación

Para esta sección se comentarán los distintos métodos de evaluación que se han llevado a cabo para clasificar la información.

### 3.6.1. División de los conjuntos de clasificación: estrategias de particionado para los conjuntos de entrenamiento y test

A la hora de clasificar los conjuntos de datos en entrenamiento y test, se han seguido cuatro estrategias:

1.– Método 1: Se dividirá en conjunto de entrenamiento y de test para cada uno de los cuatro sets anuales, y se aplicarán 5 divisiones para cada uno de los sets anuales (20 estrategias en total):

- Selección de los tres primeros meses del set anual como parte de entrenamiento y los nueve últimos como test.
- Selección de los cuatro primeros meses del set anual como parte de entrenamiento y los ocho últimos como test.
- Selección de los seis primeros meses del set anual como parte de entrenamiento y los seis últimos como test.
- Selección de los ocho primero meses del set anual como parte de entrenamiento y los cuatro últimos como test.
- Selección de los nueve primeros meses del set anual como parte de entrenamiento y los tres últimos como test.

2.– Método 2: Para todas las propuestas de los cuatro años, se seleccionará un set anual como conjunto de test, y el resto de sets anuales actuarán como datos de entrenamiento. Se obtendrá un total de cuatro estrategias.

3.– Método 3: Para cada set anual, se seleccionarán aleatoriamente 10 estrategias de entrenamiento y test, obteniendo así 40 estrategias.

4.– Método 4: Se seleccionarán 10 estrategias, en donde los datos son escogidos aleatoriamente de todo el conjunto de propuestas.

En la imagen 3.4 se muestra una aproximación de las cuatro estrategias. En el apéndice E se muestra el código en donde se aprecia con mayor detalle la estrategia de particionado en Python.

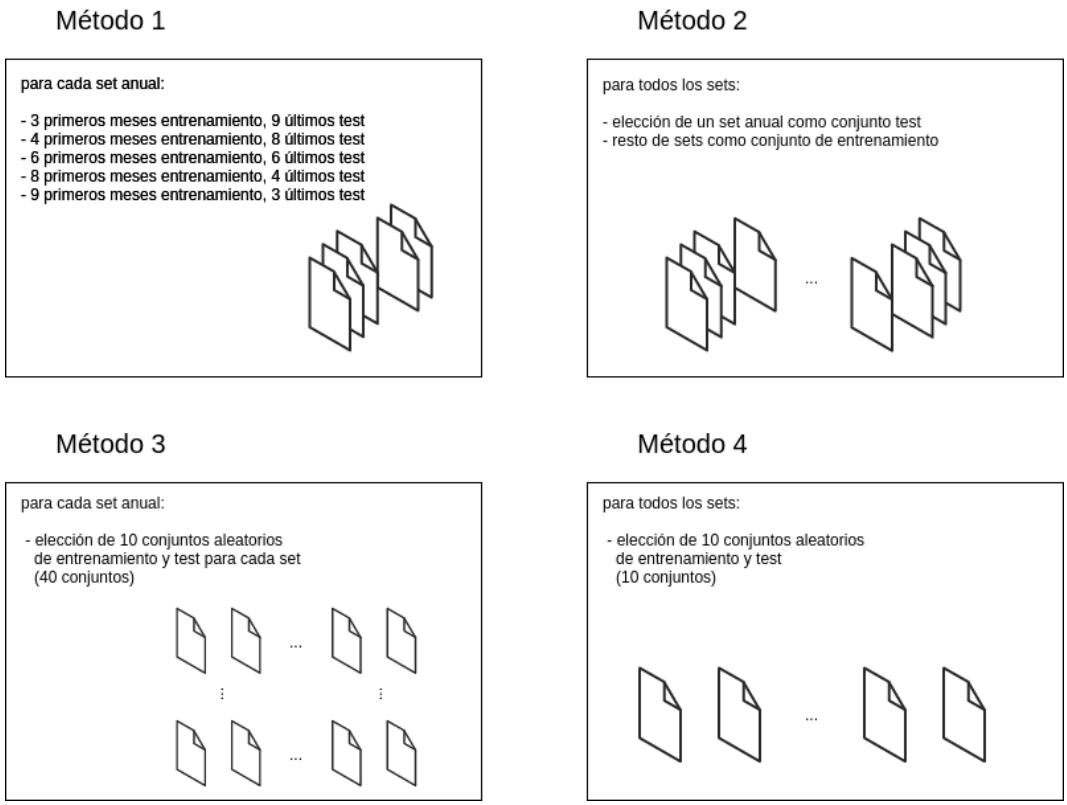



**Figura 3.4:** Aproximación de las cuatro estrategias de clasificación diseñadas

### 3.6.2. Reducción de la dimensionalidad de los atributos con PCA

Como ya se ha comentado anteriormente, este problema contenía múltiples atributos. Gracias a PCA se han detectado los atributos más relevantes, es decir, los que más representan cada clase aportando mayor varianza. En cada una de ellas se han detectado las dos combinaciones lineales de atributos. Se han realizado fundamentalmente dos pasadas del algoritmo:

- Primera pasada: Obtención de los atributos más relevantes de cada conjunto de atributos por separado, obteniendo así un listado reducido de atributos (en total, de cuatro a seis atributos más relevantes por cada conjunto, ya que algunos de ellos se repetían en las diferentes tres componentes principales que se querían analizar).
- Segunda pasada: Obtención de los cuatro atributos más relevantes de los conjuntos reducidos unidos.

Al probar la dispersión de los datos con el modelo PCA se han ido guardando en una carpeta denominada 'valores_PCA', en la que se han guardado en formato CSV las componentes principales del algoritmo PCA para cada pasada del modelo. En la sección 5.1.1 se detallan los atributos tenidos en cuenta para la reducción de la dimensionalidad.

Cabe decir que, en las primeras pasadas del método PCA , se obtuvieron resultados muy similares para algunos valores de los atributos *categories* y *topics*, por lo que se decidió no incluir en futuras clasificaciones los atributos de *topics* puesto que no añadían valor exclusivo a las propuestas, sino que simplemente puntuaban prácticamente lo mismo que algunos valores de *categories* homónimos.

### 3.6.3. Evaluación de las clasificaciones: métricas de precisión

Para evaluar los resultados obtenidos, se han implementado varias métricas de acierto. Todas ellas parten de la obtención de la matriz de confusión, en donde se comparan los valores de las clases del conjunto original con los valores obtenidos en la clasificación.

- Precisión: Es el número de aciertos para todos los valores de clase. Se obtiene de sumar los aciertos totales dividido entre el total de valores clasificados. Sin embargo, esta métrica no resulta lo suficientemente precisa para el problema abordado.
- Precisión para cada clase (acc): Es el resultado de dividir los aciertos en la clasificación para una clase (valor que está en la diagonal descendente de la matriz) entre los valores a clasificar como dicha clase (suma de los valores de la fila en la que se encuentra el valor a clasificar). Se pueden comparar las métricas de precisión de cada clase para saber qué clase resulta más fácil predecir.
- Media aritmética de las precisiones de cada clase: Obtiene una estimación media, aunque no es muy ajustada si una clase no se ha clasificado tan bien como el resto.
- Media geométrica (g) de las precisiones de cada clase: A diferencia de la media aritmética, esta métrica es más sensible a las desigualdades de clasificación de las distintas clases, obteniendo así una métrica más verosímil para representar el porcentaje de acierto de clasificación. Esta métrica es la que se usa en [2] para obtener el porcentaje de acierto de la clasificación.

En la sección de pruebas, se expondrán los valores obtenidos.

**Código 3.1:** Parte del código donde se muestran las métricas que se han empleado para medir la precisión de aciertos en la clasificación.

```
282     accs = {}
283     class_names = []
284
285     if pdt.NON_BINARY in class_problem:
286         num_class = 4
287     else:
288         num_class = 2
289
290     accs["average"] = []
291     accs["geometric_mean"] = []
292     for i in range(0, num_class):
293         accs[i] = []
294         class_names.append(i)
295
296     for X_train, X_test, y_train, y_test in zip(dict_values['X_train'], dict_values['X_test'],
297                                                 dict_values['y_train'], dict_values['y_test']):
298         clsf.fit(X_train, y_train)
299         pred_vals = clsf.predict(X_test)
300
301         # Asigna el valor medio de aciertos, sin tener en cuenta los valores de cada clase
302         accs["average"].append(round(accuracy_score(y_test, pred_vals), 4))
303
304         # Obtiene los valores de acierto de cada clase mediante la matriz de confusión
305         # Escala los datos de la matriz a valores entre 0 y 1 (evaluación de aciertos)
306         # respecto de los valores clasificados para cada clase.
307         matrixc = confusion_matrix(y_test, pred_vals, labels=class_names)
308         matrixc_accs = matrixc.astype('float') / matrixc.sum(axis=1)[:, np.newaxis]
309
310         # De la matriz escalada, se obtienen los aciertos de clase, que corresponden con
311         # los valores de la diagonal de la matriz
312         matrixc_diagonal = matrixc_accs.diagonal()
313
314         # Asigna los valores individuales de acierto para cada clase en el diccionario de aciertos
315         for i, class_acc_value in zip(class_names, matrixc_diagonal):
316             accs[i].append(round(class_acc_value, 4))
317
318         # Calcular la media geométrica
319         accs["geometric_mean"].append(round(gmean(list(matrixc_diagonal)), 4))
```

# 4

# Desarrollo

En esta sección se indican los detalles de implementación del trabajo, ya que en la parte de diseño (ver capítulo 3) se ha comentado la implementación a alto nivel.

A continuación se comentarán las tecnologías usadas para llevar a cabo el TFG : base de datos, lenguajes de programación y uso de Integrated Development Environment - Entrono de Desarrollo Integrado (IDE) , librerías y otras herramientas utilizadas y se detallarán los modelos de clasificación que se han utilizado.

## 4.1. Base de datos empleada

La base de datos que contiene los datos de Decide Madrid que, como ya se ha comentado, vienen dados por [10], se ha almacenado en un servidor de base datos MySQL [1] mediante el *script* provisto para cargar las tablas y su información.

La base de datos cuenta con unas 19 tablas, donde las más relevantes son las que contienen información de las propuestas (i.e. *proposals*,*proposal_comments*, *proposal_districts*, *proposal_categories*) y otro contenido como *geo_locations*, *geo_districts geo_neighbourhoods* que han permitido ubicar el distrito y el barrio de gran parte de las propuestas, o *cat_categories* y *cat_topics* que recogen las temáticas principales asignadas a las propuestas.

Para visualizar y hacer algunas consultas, se ha usado el software cliente DBeaver [2] para usarlo en el sistema operativo Ubuntu basado en Linux.

[1] MySQL - Oracle, web: https://www.mysql.com/

[2] DBeaver Community, web: https://dbeaver.io/about/

## 4.2. Lenguajes de programación, librerías y otras herramientas utilizadas

### 4.2.1. Desarrollo en Java

La primera parte ha sido desarrollada con **Java** en su versión 11, empleando el IDE Eclipse [3]. Gracias a la herramienta **mysql-connector-java.jar** [4] con la que se ha conectado al servidor de MySQL para poder ejecutar consultas y recibir datos de las tablas de la base de datos. Al pasar la información de los conjuntos de atributos de *categories*, *districts* y *topics* a través de consultas, se ha adaptado la categorización de los atributos textuales recibidos desde la base de datos a como se indica en [11]. Para el caso de los atributos textuales, se ha utilizado la herramienta de Stanford CoreNPL [9]. Los atributos relativos del texto de las propuestas que se han analizado han sido los siguientes:

- Longitud del resumen (*summary*).
- Longitud del contenido.
- Número de nombres propios que aparecen en el resumen y en el contenido.
- Número de palabras en mayúsculas que aparecen en el resumen y el contenido.
- Número de adjetivos que aparecen en el contenido.
- Número de palabras positivas que aparecen en el resumen y en el contenido.
- Número de palabras negativas que aparecen en el resumen y en el contenido.
- Número de determinantes posesivos y pronombres personales y posesivos que aparecen en el resumen y en el contenido.
- Número de enlaces web a los que se hace referencia tanto en el resumen como en el contenido.

Para analizar las palabras positivas y negativas que tiene cada propuesta, se ha utilizado el conjunto de palabras de Sinai - iSOL [12]. En el apéndice B se detalla gráficamente la distribución de los distintos atributos relacionados con el texto de las propuestas.

Ya obtenidos los datos, se han asignado los valores de las clases tal y como viene detallado en 3.4. Se ha tomado la estrategia de dividir los valores de las clases binarias por la **mediana** y el valor de los problemas multiclase por **cuartiles**, haciendo de esta forma que aproximadamente todos los valores asignados sean equiparables en número de propuestas.

Una vez obtenidos los datos, se han creado clases que permiten almacenar dicha información, así como otras clases que permiten adaptar esta información y exportarla a ficheros externos en formatos ARFF y CSV para que sean legibles desde otros medios.

[3] Eclipse Foundation, web: https://www.eclipse.org/

[4] MySQL Java connector, web: https://dev.mysql.com/downloads/connector/j/

### 4.2.2. Desarrollo en Python

Por otro lado, se ha utilizado Python en la versión 3.8 debido a la facilidad de librerías que permiten realizar clasificaciones a través de Scikit-Learn [13], así como otras librerías útiles para representar matrices y arrays con NumPy [5] o tablas de datos con Pandas [6], lo cual ha facilitado gran parte del trabajo. Se ha utilizado el completo IDE PyCharm Edu [7] junto con Jupyter Notebook de Anaconda [8] para mostrar gráficas a través de Matplotlib [9].

Con la librería Pandas se han leído los datos del fichero exportado desde Java, y se ha procedido a realizar varios análisis como PCA para obtener las componentes principales junto con gráficas que ayudan a visualizar la división de las clases, las estrategias de división del conjunto de atributos en entrenamiento y test, la implementación de los modelos de clasificación (se explican en el siguiente apartado) y la obtención de los resultados tras evaluar los valores de precisión (ver en sección 5.2).

## 4.3. Clasificadores empleados

Los modelos de clasificación utilizados se han implementado gracias a la librería Scikit-Learn [13]. A continuación se detallan los modelos escogidos para la clasificación:

- Naive Bayes: Modelo sencillo que realiza una clasificación asumiendo que los atributos del modelo son independientes entre sí, obteniendo un resultado razonable, a la vez que se reduce considerablemente el número de cálculos. Para este problema se han empleado las implementaciones de Bernoulli y Multinomial, debido a la cantidad de atributos categóricos del problema.
- Árboles de decisión y Random Forest: Eligen los valores que mayor dividen el conjunto de datos, si es posible. Para evitar sobreajustes, se ha limitado la profundidad de los árboles y, en el caso de Random Forest, se ha limitado el número máximo de árboles empleado.
- Regresión logística: Especialmente para clasificaciones binarias, emplea una función sigmoidal, con lo que se obtiene una probabilidad estimada de cada valor para ambas clases.
- Vecinos próximos: Elige si el valor a clasificar se asemeja más a una clase u otra por la proximidad de los K valores próximos.
- Máquinas Vector Soporte: realiza una aproximación de la frontera de los datos, y reajusta los valores de la muestra para que aparezcan las clases lo mejor divididas posibles.
- LDA y QDA : Emplean autovalores para realizar una combinación lineal de los atributos que mejor clasifiquen los datos, y a partir de ahí realiza una frontera lineal en caso de LDA o cuadrático en QDA . Este modelo intenta imitar el teorema de Bayes, ya que busca a mayor dependencia de los atributos para obtener la mejor clasificación posible.

[5] NumPy, web: https://numpy.org/
[6] Pandas, web: https://pandas.pydata.org/
[7] PyCharm Edu, web: https://www.jetbrains.com/pycharm-edu/
[8] Anaconda, web: https://www.anaconda.com/
[9] Matplotlib - Visualization with Python, web: https://matplotlib.org/stable/index.html

# 5

# Pruebas y Resultados

Para realizar clasificaciones y obtener resultados que indiquen cómo de acertada ha sido la clasificación, se han ejecutado varios *notebooks* en Python, en donde se han obtenido tanto resultados como gráficas.

En esta sección se detallarán los pasos seguidos para realizar las clasificaciones y, finalmente, se comentarán qué modelos de clasificación han dado mejores o peores resultados.

## 5.1. Ejecución de los modelos de clasificación

Para probar las clasificaciones y obtener los atributos más relevantes, se han ejecutado dos *notebooks* en los que se muestran gráficas de cómo se ha realizado algunas clasificaciones y el formato en el que los datos se han guardado (por ejemplo, cada uno de los atributos con mayor relevancia en PCA para cada pasada).

### 5.1.1. Análisis de PCA. Muestra de resultados y gráficas de PCA y LDA.

Antes de probar las clasificaciones, se ha decidido realizar una reducción de los atributos. De esta forma, se ha realizado un algoritmo en el que se ha pasado el modelo PCA con dos componentes principales en el conjunto completo de atributos, obteniendo una ponderación de la relevancia que tienen los atributos para formar cada una de las dos componentes con mayor cantidad de información del conjunto. Con dichos atributos, se ha reducido el espacio de atributos a solamente los que se han indicado como relevantes (ver 3.6.2). Tras una segunda ejecución de PCA , se obtienen los cuatro atributos que mayor varianza aportan a todo el espacio de atributos.

En 5.1 se muestra parte del código que permite calcular y mostrar los valores de PCA .

Respecto a los atributos más relevantes obtenidos (se recuerda que no se han incluido en este apartado los atributos relacionados con los *topics* puesto que en 3.6.2 se decidió que no aportaban más valor que el que ya aportaban los atributos *categories*), se han obtenido los siguientes resultados:

**Código 5.1:** Función de muestra para calcular los valores de PCA, y posteriormente se muestra la gráfica asignando a cada clase un color y forma. Las iteraciones para los tipos y valores de clasificación, así como los atributos utilizados no se muestran en este ejemplo. En las gráficas solamente se representarán los valores de PC1 y PC2 puesto que se muestran en dos dimensiones.

```
120                 # ----- PCA -----
121
122                 pca = PCA(n_components=len(pdt.list_pca))
123                 X_pca = pca.fit_transform(X, y)
124
125                 print(f" Valores PCA: {pca.explained_variance_ratio_}")
126
127                 dict_pca_info['values'][data_type] = pca.explained_variance_ratio_
128                 dict_pca_info[data_type] = pd.DataFrame(pca.components_, columns=X_columns,
                        index=pdt.list_pca)
129                 dict_pca_info[data_type] = dict_pca_info[data_type].abs()
130
131                 dict_pca_plot[data_type] = pd.DataFrame(X_pca, columns=pdt.list_pca)
132
133                 plt.title(f"PCA: resultados de los datos {data_plt} para predecir {class_name}
                        ({class_type})")
134                 plt.xlabel('PC1')
135                 plt.ylabel('PC2')
136
137                 for class_t, color, legend, a, m in zip(targets, colors, legends, alphas, markers):
138                     indices_kept = dict_data_class[data_type][index_dict_data] == class_t
139                     plt.scatter(dict_pca_plot[data_type].loc[indices_kept, 'PC-1'],
140                                 dict_pca_plot[data_type].loc[indices_kept, 'PC-2'],
141                                 label=legend, c=color, alpha=a, marker=m)
142                 plt.legend(bbox_to_anchor=(1, 1), loc='upper left')
143                 plt.show()
```

**Primera pasada**: Se han almacenado los seis atributos de cada conjunto de datos que mayor varianza aportan para formar las dos componentes principales de la primera ejecución de PCA .

- Atributos del conjunto *districts* para la Componente Principal 1: dis-00 (Ciudad), dis-08 (Fuencarral - El Pardo), dis-13 (Puente de Vallecas), dis-16 (Hortaleza), dis-02 (Arganzuela), dis-11 (Carabanchel)
- Atributos del conjunto *districts* para la Componente Principal 2: dis-13 (Puente de Vallecas), dis-18 (Villa de Vallecas), dis-00 (Ciudad), dis-05 (Chamartín), dis-16 (Hortaleza), dis-11 (Carabanchel)
- Atributos del conjunto *categories* para la Componente Principal 1: cat-17 (Medio Ambiente), cat-29 (Urbanismo), cat-25 (Sostenibilidad), cat-18 (Movilidad), cat-11 (Educación), cat-10 (Economía)
- Atributos del conjunto *categories* para la Componente Principal 2: cat-18 (Movilidad), cat-29 (Urbanismo), cat-09 (Derechos Sociales), cat-10 (Economía), cat-13 (Equidad e Integración), cat-17 (Medio Ambiente)
- Atributos del conjunto *text analysis* para la Componente Principal 1: txa-02 (Longitud del contenido), txa-03 (Número de nombres propios), txa-07 (Número de palabras negativas), txa-08 (Número de determinantes y pronombres personales y posesivos), txa-05 (Número de adjetivos), txa-06 (Número de palabras positivas)
- Atributos del conjunto *text analysis* para la Componente Principal 2: txa-07 (Número de palabras negativas), , txa-06 (Número de palabras positivas), txa-08 (Número de determinantes y pronombres personales y posesivos), txa-02 (Longitud del contenido), txa-05 (Número de adjetivos), txa-03 (Número de nombres propios)

**Segunda pasada**: Se han almacenado los cuatro atributos que mayor varianza aportan de los atributos obtenidos en la primera pasada para formar las dos componentes principales de la primera ejecución de PCA .

- Atributos obtenidos para la Componente Principal 1: cat-17 (Medio Ambiente), cat-29 (Urbanismo), dis-00 (Ciudad), cat-25 (Sostenibilidad)
- Atributos obtenidos para la Componente Principal 2: dis-00 (Ciudad), cat-17 (Medio Ambiente), cat-25 (Sostenibilidad), cat-29 (Urbanismo)

Como el objetivo de esta sección ha sido reducir el espacio de atributos lo máximo posible, se ha concluido que los cuatro atributos que mayor varianza aportan del conjunto de datos son el distrito que representa a **toda la ciudad** junto a las categorías **Medio Ambiente**, seguida de **Urbanismo** y **Sostenibilidad**, los cuales se detectaron como relevantes desde el análisis de la primera componente principal de cada uno de sus conjuntos respectivamente.

Para cada pasada del modelo PCA se han obtenido los mayores porcentajes de varianza, uno por cada componente principal analizada. Tras realizar varias pasadas, se ha logrado obtener unos valores de aproximadamente 37 % para la primera componente y un 30 % para la segunda.

Cabe decir que los valores obtenidos de clasificar con PCA para solamente los atributos de texto es bastante bueno (un 70 % de relevancia en la primera componente), pero los atributos de texto dejan de ser relevantes en cuanto se evalúa el modelo con todos los atributos. Esto se debe a que no están distribuidos ni representados de la misma manera, ya que los atributos de análisis del texto son numéricos y el resto categóricos binarios. Al escalar los atributos textuales a valores entre 0 y 1, como es *MinMaxScaler*, el problema que ocurre es que por la distribución de ley de Zipf de los valores (muchos atributos con valores bajos y muy pocos con valores elevados), hace que con este escalado

**los valores sean muy cercanos a 0 y prácticamente ninguno de los valores tengan valores cercanos a 1**, y al compararlos con el resto de atributos categóricos pierdan relevancia. Este problema se puede solventar, por ejemplo, transformando los atributos numéricos a categóricos binarios con una estrategia de división del valor de la mediana.

De forma similar a PCA , se han analizado las componentes más relevantes para clasificar las propuestas con el modelo LDA .

En el apéndice C se muestran las gráficas resultantes de aplicar el modelo PCA al conjunto de datos. Como se observa, los datos parecen relativamente concentrados; de hecho, aunque se agrupan en varios puntos (principalmente por la relevancia de los atributos categóricos), permanecen los datos de ambas clases bastante unidos. Aún así este modelo intenta dividir los atributos en función de la varianza del conjunto, y no precisamente clasificar o separar por los distintos valores de las clases.

De forma simultánea, se han ejecutado el modelo de clasificación LDA junto con el de PCA . En este caso, sí se puede apreciar más la reducción de la dimensionalidad de los atributos, sobre todo en las clasificaciones con más de dos valores. El modelo, en este caso, intenta dividir el conjunto lo mejor posible en base a los valores de clasificación, aunque como se muestra en el apéndice D no resulta tan evidente visualizar las fronteras entre las propias clases.

## 5.2. Formato de salida de los ficheros

Ya que se pretenden obtener unos resultados organizados de las clasificaciones, se ha diseñado una función que al recibir cada una de las estrategias de división de entrenamiento y test y su posterior obtención de métricas de precisión (véase de nuevo la sección 3.6), crea un árbol de ficheros.

Primero, se crea una carpeta *values*, si no esta ya creada, en la que se encuentran tantas carpetas como **conjuntos de atributos** se deseen emplear para clasificar cada problema.

Posteriormente, se crearán, tres carpetas que corresponden con el **conjunto completo** de todos los atributos, el **primer conjunto reducido** de atributos y el **segundo conjunto reducido** con los **cuatro atributos con mayor relevancia**. Para cada una de las carpetas relacionadas con los atributos se incluirá una carpeta por cada modelo de clasificación (los modelos se detallan en 4.3).

Finalmente, para cada subdirectorio se almacenarán los ficheros ordenados por su metodología usada, conteniendo los porcentajes de acierto por cada valor de clasificación y las métricas de precisión media aritmética y media geométrica para cada problema. Al final de cada fichero se aportan, de nuevo, las medias aritmética y geométrica de todos los problemas para dicha metodología.

En el apéndice E muestran parte de la implementación llevada a cabo.

## 5.3. Clasificaciones

En esta sección se comentará la precisión obtenida para cada modelo de clasificación.

Aún teniendo en cuenta las distintas métricas empleadas (como ya se ha comentado anteriormente en el apartado 3.6.3), se ha utilizado, al igual que en [2], la métrica correspondiente a la **media geométrica** debido a que la media aritmética no es tan sensible si alguna de las clasificaciones es mucho peor que otras, mientras que la media geométrica sí aporta mayor sensibilidad puesto que representa una idea de la clasificación general para todos los valores. Además, se han obtenido las **métricas de precisión para cada valor binario mejor ($acc^+$) y peor ($acc^-$)**, es decir, en las clasificaciones binarias, se ha obtenido el porcentaje de aciertos de los dos tipos de valores con mayor acierto de clasificación (generalmente el valor 0 de la clase) y el valor restante con peor valor de clasificación (generalmente el valor 1) También se ha obtenido el mejor valor ($acc^+$) para las clasificaciones con valores multiclase, quedando siempre el valor 0 como el mejor clasificado de todos.

En las tablas 5.1 y 5.2 se muestran los porcentajes de acierto mediante la media geométrica.

| RELEVANCIA | MÉTODO | Naive Bayes Bernoulli | Naive Bayes Multinomial | Árbol de Decision | Random Forest | Máquina Vector Soporte | LDA | QDA | Regresión Logística | Vecinos Próximos |
|---|---|---|---|---|---|---|---|---|---|---|
| bin - todos | METODO 1 | 54,76% | 54,76% | 48,78% | 36,23% | 52,04% | 54,10% | 40,61% | 53,37% | 50,27% |
| bin - attr, set1 * | METODO 1 | 53,67% | 46,16% | 48,89% | 42,22% | 49,32% | 51,25% | 50,95% | 50,58% | 51,63% |
| bin - todos | METODO 2 | 56,03% | 56,03% | 55,03% | 54,77% | 59,09% | 57,68% | 51,83% | 57,67% | 53,51% |
| bin - attr, set1 * | METODO 2 | 55,73% | 52,98% | 53,95% | 55,64% | 56,48% | 56,33% | 49,90% | 56,37% | 52,89% |
| bin - todos | METODO 3 | 58,55% | 58,55% | 54,90% | 55,81% | 59,70% | 59,30% | 41,12% | 59,64% | 53,72% |
| bin - attr, set1 * | METODO 3 | 57,47% | 54,78% | 54,68% | 56,62% | 57,31% | 57,68% | 50,77% | 57,53% | 54,37% |
| bin - todos | METODO 4 | 57,91% | 57,91% | 55,94% | 56,28% | 59,46% | 58,45% | 52,93% | 59,31% | 54,85% |
| bin - attr, set1 * | METODO 4 | 56,03% | 54,50% | 54,64% | 55,63% | 57,44% | 57,70% | 50,09% | 57,68% | 53,55% |
| multi - todos | METODO 1 | | | | | | 28,07% | 11,44% | | 25,87% |
| multi - attr, set1 * | METODO 1 | | | | | | 25,13% | 22,00% | | 27,25% |
| multi - todos | METODO 2 | | | | | | 29,00% | 23,76% | | 27,08% |
| multi - attr, set1 * | METODO 2 | | | | | | 26,07% | 23,18% | | 26,29% |
| multi - todos | METODO 3 | | | | | | 30,41% | 10,49% | | 28,04% |
| multi - attr, set1 * | METODO 3 | | | | | | 29,54% | 23,06% | | 28,54% |
| multi - todos | METODO 4 | | | | | | 29,36% | 24,68% | | 28,23% |
| multi - attr, set1 * | METODO 4 | | | | | | 26,55% | 23,93% | | 27,95% |

**Figura 5.1:** Tabla que muestra los resultados de las evaluaciones para la clase relevancia. Los porcentajes corresponden a los valores obtenidos con la media geométrica.

Como muestran los resultados, en general, el acierto en la clasificación ha sido **relativamente bajo**, ya que apenas supera el 60 % de aciertos en en ninguna de las métricas. Además, debido al más escaso acierto en las clasificaciones multiclase, solamente se han probado con tres modelos ( LDA , QDA y vecinos próximos) ya que este problema tardaba demasiado en clasificar y los resultados obtenidos no iban a diferir demasiado de los obtenidos en la clasificación binaria.

En las tablas 5.3 y 5.4 se muestran los valores medios de las clasificaciones binarias para todos los modelos de clasificación. Los porcentajes de $acc^+$ corresponden, pues, con los valores de la clase 0 (no relevante o no controvertido) tanto para las clasificaciones binarias como para las multiclase, mientras que los porcentajes de $acc^-$ corresponden con el valor 1 (relevante o controvertido) de las clasificaciones binarias.

| CONTROVERSIA | MÉTODO | Naive Bayes Bernoulli | Naive Bayes Multinomial | Árbol de Decision | Random Forest | Máquina Vector Soporte | LDA | QDA | Regresión Logística | Vecinos Próximos |
|---|---|---|---|---|---|---|---|---|---|---|
| bin - todos | METODO 1 | 54,73% | 50,55% | 48,77% | 37,14% | 51,63% | 54,12% | 40,75% | 53,28% | 49,77% |
| bin - attr, set1 * | METODO 1 | 53,98% | 46,73% | 49,14% | 43,24% | 49,86% | 51,62% | 51,15% | 50,93% | 51,48% |
| bin - todos | METODO 2 | 54,70% | 53,11% | 52,29% | 53,65% | 55,96% | 55,11% | 49,14% | 54,94% | 50,84% |
| bin - attr, set1 * | METODO 2 | 54,43% | 52,86% | 52,08% | 53,67% | 53,88% | 54,14% | 49,93% | 54,30% | 51,04% |
| bin - todos | METODO 3 | 58,64% | 56,33% | 54,60% | 55,52% | 59,93% | 59,20% | 41,59% | 59,24% | 53,66% |
| bin - attr, set1 * | METODO 3 | 57,66% | 54,79% | 54,47% | 56,50% | 57,13% | 57,49% | 50,10% | 57,52% | 54,59% |
| bin - todos | METODO 4 | 55,98% | 54,37% | 53,21% | 52,58% | 56,61% | 56,34% | 50,35% | 56,71% | 51,72% |
| bin - attr, set1 * | METODO 4 | 54,62% | 53,55% | 53,78% | 52,92% | 54,58% | 55,90% | 50,23% | 55,78% | 52,01% |
| multi - todos | METODO 1 | | | | | | 28,16% | 13,08% | | 26,42% |
| multi - attr, set1 * | METODO 1 | | | | | | 25,76% | 21,59% | | 27,18% |
| multi - todos | METODO 2 | | | | | | 27,15% | 23,65% | | 25,96% |
| multi - attr, set1 * | METODO 2 | | | | | | 23,95% | 22,56% | | 26,04% |
| multi - todos | METODO 3 | | | | | | 30,85% | 9,82% | | 28,03% |
| multi - attr, set1 * | METODO 3 | | | | | | 29,82% | 23,03% | | 28,77% |
| multi - todos | METODO 4 | | | | | | 28,35% | 24,64% | | 27,21% |
| multi - attr, set1 * | METODO 4 | | | | | | 24,18% | 23,67% | | 27,16% |

**Figura 5.2:** Tabla que muestra los resultados de las evaluaciones para la clase controversia. Los porcentajes corresponden a los valores obtenidos con la media geométrica.

| RELEVANCIA | valores binarios acc+ | valores binarios acc- | valores multiclase media acc | valores multiclase acc+ (valor 0) |
|---|---|---|---|---|
| Naive Bayes Bernoulli | 67,00% | 46,00% | | |
| Naive Bayes Multinomial | 69,00% | 46,00% | | |
| Árbol de Decision | 65,00% | 52,00% | | |
| Random Forest | 65,00% | 48,00% | | |
| Máquina Vector Soporte | 64,00% | 57,00% | | |
| Regresión Logística | 67,00% | 53,00% | | |
| LDA | 67,00% | 53,00% | 34,00% | 57,00% |
| QDA | 75,00% | 38,00% | 32,00% | 63,00% |
| Vecinos Próximos | 73,00% | 41,00% | 32,00% | 51,00% |

**Figura 5.3:** Porcentajes de aciertos aproximados para la clase relevancia. Los porcentajes de $acc^+$ (para ambos tipos de clasificaciones) corresponden con el valor 0 y los de $acc^-$ (solo para las clasificaciones binarias) al valor 1.

| CONTROVERSIA | valores binarios acc+ | valores binarios acc- | valores multiclase media acc | valores multiclase acc+ (valor 0) |
|---|---|---|---|---|
| Naive Bayes Bernoulli | 66,00% | 51,00% | | |
| Naive Bayes Multinomial | 69,00% | 44,00% | | |
| Árbol de Decision | 63,00% | 52,00% | | |
| Random Forest | 64,00% | 45,00% | | |
| Máquina Vector Soporte | 66,00% | 55,00% | | |
| Regresión Logística | 67,00% | 52,00% | | |
| LDA | 68,00% | 52,00% | 34,00% | 56,00% |
| QDA | 76,00% | 39,00% | 32,00% | 64,00% |
| Vecinos Próximos | 74,00% | 40,00% | 31,00% | 50,00% |

**Figura 5.4:** Porcentajes de aciertos aproximados para la clase controversia. Los porcentajes de $acc^+$ (para ambos tipos de clasificaciones) corresponden con el valor 0 y los de $acc^-$ (solo para las clasificaciones binarias) al valor 1.

Observando los resultados, los modelos que mejor clasifican la clase con mayor acierto son los clasificadores **QDA** y **KNN**, debido a que estos algoritmos clasifican por varianza o semejanza entre los valores de las propuestas. Por otro lado, para el valor 0 es el modelo de **Máquina de Vector-Soporte** con el que se obtiene el mejor porcentaje de aciertos. En general todos los clasificadores obtienen buenos resultados al clasificar la clase 0, donde en algunos casos supera el 70 % de aciertos para las clasificaciones binarias, como se muestra en las tablas 5.3 y 5.4. Esto puede ser debido a la distribución de los datos, que al ser ley de Zipf hay muchas propuestas que apenas contienen votos, comentarios y/o valor *aggregation* (véase apéndice A), por lo que los modelos de clasificación pueden aprender que es más fácil clasificar con valor 0 una propuesta.

Por otro lado, para las clasificaciones con más de 2 valores, los clasificadores generalmente dan buenos resultados para clasificar con el valor 0 (al igual que con la clasificación binaria) y el máximo valor. La argumentación de este resultado puede deberse a que, además de lo comentado para la clasificación con valor 0 en el anterior párrafo, los valores máximos de las distribuciones son bastante característicos, es decir, las propuestas que suelen tener muchos votos también suelen tener muchos comentarios, e incluso un alto valor de *aggregation* (ya que según las métricas de controversy/aggregation explicadas en 3.3 la controversia medida de esta manera depende, en parte, del número de comentarios). No obstante, en las clasificaciones multiclase, los clasificadores parecen que **no entienden muy bien cómo clasificar los valores intermedios**, ya que por la distribución de las características de las propuestas, es muy probable que, por ejemplo, propuestas clasificadas en multiclase con valores 0 y 1 tengan valores comunes, y de la misma forma al comparar los valores 1 y 2, por lo que los clasificadores pueden aumentar la tasa de fallos al comparar los valores en estos casos. Por este razonamiento, además de la obtención de los bajos porcentajes de acierto que aportan las clasificaciones con más de dos valores (apenas un 30 % de aciertos), hace que se estudie con mayor detenimiento los resultados de las **clasificaciones binarias**.

# 6

# Conclusiones y trabajo futuro

Para concluir el trabajo, se comentarán los resultados obtenidos de forma que se hayan aproximado o no a los objetivos deseados.

También se hará un análisis más detallado de los atributos que han resultado ser más relevantes a la hora de realizar clasificaciones, además de buscar otros atributos que se podrían tener en cuenta para detectar propuestas más relevantes y controvertidas.

Finalmente se expondrá el trabajo futuro a realizar para poder mejorar los resultados.

## 6.1. Conclusión respecto de los objetivos iniciales

En la sección inicial se ha hecho una breve introducción del problema que conlleva un portal de debate ciudadano y de presupuestos participativos como es Decide Madrid. A lo largo del trabajo se ha detallado cómo se han medido la relevancia y la controversia de las propuestas generadas entre los años 2015 y 2019 en función de tres de sus métricas.

Si bien es cierto que en el anterior apartado se llegó a la conclusión de que los clasificadores no resultaban muy precisos a la hora de clasificar las propuestas entre los diferentes valores de las clases, se han obtenido resultados acordes a lo esperado, aunque se podrían mejorar los resultados.

A la hora de clasificar con valores binarios las propuestas, se ha obtenido en general un mayor acierto de las clasificaciones con el valor más bajo (i.e. el valor 0). El problema de clasificación se ha agravado debido a la distribución de los datos (ver de nuevo los valores medidos en distribución de ley de Zipf en el apéndice A), de forma que sea fácil determinar las propuestas no controvertidas o no relevantes, pero complicando el entrenamiento por las **escasas propuestas con valores elevados**. Por lo tanto, si el problema fuera la estimación de las clases, se podrían buscar otros indicadores con distribuciones más homogéneas. Sin embargo, al ser en parte un problema de búsqueda en textos, es probable que otros atributos que se puedan obtener también se distribuyan mediante ley de Zipf (por ejemplo, en el apéndice B hay varios atributos relacionados con los términos del texto de las propuestas que también siguen esta distribución).

## 6.2. Análisis de las características más relevantes

Aunque no se han obtenido buenas clasificaciones, sí se han obtenido los atributos más relevantes de los conjuntos de atributos. Se ha visto que en las tablas de resultados (5.1 y 5.2) los conjuntos de atributos reducidos no han mejorado los resultados, pero tampoco han perdido prácticamente la **precisión al clasificar**, de forma que lo que ha mejorado más significativamente han sido la reducción de cálculos en las clasificaciones.

En el apéndice F se muestra con detalle los valores asignados a cada distrito y cada categoría del conjunto de las propuestas.

Echando un vistazo a los valores más relevantes (los distritos **Ciudad**, **Fuencarral - El Pardo** y **Arganzuela**, y las categorías **Medio Ambiente**, **Urbanismo** y **Movilidad**) coinciden con otros valores que se han obtenido al analizar las componentes principales con PCA (véase 5.1), lo cual es un buen índice. Teniendo en cuenta este análisis, se puede ahondar más en este tipo de problemas de la ciudadanía madrileña, ya que parecen problemas de índole común.

Por otro lado, es curioso el caso de propuestas de categorías como Religión o Vivienda en las que, a pesar de no tener muchas propuestas asignadas a dichos temas, tienen valores proporcionalmente altos de apoyos y *aggregate*.

Dos propuestas de temática religiosa o de vivienda son las que alcanzan mayor número de apoyos. Estas son:

- Aplicación del IBI a bienes inmuebles de la iglesia. [1]
- Respetar la tradición de los belenes. [2]

Otras categorías relevantes podrían ser las denominadas Not In My BackYard - No en mi patio (NIMBY) , propuestas que son necesarias, pero los ciudadanos quieren desplazarlas a otros lugares ya que en gran parte generan problemas de salud o convivencia, entre otros. El caso con mayor relevancia en la plataforma decide Madrid de este tipo es la propuesta sobre la Incineradora de Valdemingómez [3]. Para poder detectar relevancia o controversia en estas propuestas, puede ser interesante tener en cuenta el uso de pronombres y determinantes posesivos, sobre todo en los comentarios, donde las opiniones son más propias.

[1] Decide Madrid - Propuesta 965, web:https://decide.madrid.es/proposals/965
[2] Decide Madrid - Propuesta 3736, web:https://decide.madrid.es/proposals/3736
[3] Decide Madrid - Propuesta 19914, web:https://decide.madrid.es/proposals/19914

## 6.3. Atributos a tener en cuenta

Tras un análisis de las propuestas tanto en la plataforma como con los resultados obtenidos, la temática de las propuestas no reside en el lugar en el que se puede llevar a cabo, sino más bien en el **tema que la propuesta conlleva**. Esto queda demostrado cuando, al realizar el análisis de las componentes principales PCA para los atributos del conjunto de distritos, para la primera componente pondera casi íntegramente el distrito que corresponde con Ciudad (pondera más de un 95 % con solamente ese atributo), lo cual indica que las propuestas que más relevancia o controversia puedan tener en la ciudad son las que precisamente se llevan a cabo en toda ella, sin especificar un lugar en concreto (por ejemplo, los temas de limpieza, plantación de árboles, transporte público o movilidad, entre otras, no se restringen a un distrito, barrio o zona en concreto). Además, es evidente que propuestas que afectan solamente a un barrio o distrito tengan menor número de apoyos, pues los ciudadanos y ciudadanas que vivan en otras zonas no les afectarían dichas mejoras como a los vecinos y vecinas de la zona propuesta. No obstante, resultaría interesante **agrupar distritos de zonas cercanas** o áreas que **compartan similitud**, como los distritos interiores de la M-30 o los distritos del sur y este, ya que podrían compartir propuestas similares y obtener así un mayor número de apoyos.

Para medir la controversia, podría ser relevante medir el **número de usuarios que han publicado comentarios en las propuestas**, de forma que si se genera un debate, habrá mayor controversia si participa en él un mayor número de personas en vez de una mera discusión entre dos personas.

Por otro lado, la idea sería centrarse más en qué temáticas reciben la mayor cantidad de apoyo de la ciudadanía. Las temáticas más relevantes sugeridas tras realizar el análisis PCA han sido **Medio Ambiente** seguido de **Movilidad** y **Sostenibilidad**. Por lo tanto, se pueden buscar atributos o temas clave más concretos que ayuden a **seleccionar propuestas de este tema** (por ejemplo aparcamiento, Madrid Central, reducción de gases de efecto invernadero y similares).

Cabe decir que uno de los problemas detectados al clasificar las propuestas han sido los distintos tipos de valores de los conjuntos, ya que los atributos de texto, al ser numéricos perdieron relevancia contra los atributos categóricos después de escalarlos a una distribución con valores entre 0 y 1, donde se recoge y se plantea una solución a dicho problema en 5.1.1.

Estos atributos pueden clarificar las propuestas que sean relevantes o comprometidas, pero por contra, si se utilizan demasiados atributos de la misma temática, la clasificación polarizará aún más las propuestas de esta índole, ignorando el resto de problemas de otros temas o distritos que puedan surgir.

## 6.4. Problemas interesantes a tener en cuenta

Para finalizar el trabajo, se proponen alternativas que se podrían tener en cuenta para mejorar los resultados.

En general, si se utilizan atributos cuya distribución sea normal, lineal, o binomial, se podrían obtener mejores resultados ya que muchos modelos de clasificación trabajan mejor con este tipo de distribuciones.

También se podría realizar un problema de regresión o incluso obtener un *ranking* o índice con el que obtener las propuestas más relevantes o controvertidas (de forma similar que con las consultas realizadas en el apéndice F), y de este *ranking* obtener los términos más relevantes mediante el algoritmo *tf-idf* para predecir si nuevas propuestas que poseen dichos términos puedan ser más relevantes, e incluso tenerlo en cuanta a la hora de anunciarlas en la plataforma web.

También se podrían buscar atributos relacionados directamente con la población para las diferentes áreas de la ciudad (número de habitantes, renta media, ideología política, porcentajes de los distintos grupos de edades, esperanza de vida, tasa de natalidad, etc.) o relacionados con las infraestructuras (colegios, institutos, bibliotecas, hospitales, centros de salud, parques, centros comerciales, supermercados, tiendas locales y comercios, edificios de oficinas, estaciones e intercambiadores de transporte, etc.) podrían aportar información de cómo viven los vecinos y vecinas de distintas zonas, así como predecir qué necesidades tienen unos barrios que otros no tienen. Por ejemplo, en frecuente ver propuestas relacionadas con mejorar la comunicación en los nuevos barrios creados en la periferia de Madrid, donde los ciudadanos protestan ante la falta de recursos (mala comunicación en transporte público, falta de colegios, etc.) como es el caso de la propuesta número 28.839, que pide mejorar la comunicación con el barrio de El Cañaveral [4].

En definitiva, el análisis de esta plataforma refleja el comportamiento de los más de tres millones de vecinos y vecinas del municipio, los cuales pueden utilizar esta herramienta con la idea de mejorar la convivencia de sus barrios y, en general, su ciudad. Además, permite a la ciudadanía participar en debates o apoyar presupuestos participativos, fomentando la toma de decisiones de los vecinos y vecinas en la ciudad. También puede servir de debate constructivo y fomentar la creación de agrupaciones vecinales, para así llevar a cabo ciertos proyectos o mejoras que resultarían difíciles de aplicar si se manifiestan estos problemas de forma individual.

[4] Decide Madrid - propuesta 28839, web: https://decide.madrid.es/proposals/28839

# Bibliografía


[1] C. Sanford and J. Rose, "Characterizing eParticipation," *International Journal of Information Management*, vol. 27, no. 6, pp. 406–421, 2007.

[2] J. L. Lavado, I. Cantador, M. E. Cortés-Cediel, and M. Fernández, "Automatic Intent-based Classification of Citizen-to-Government Tweets," *Ongoing research paper at the 19th IFIP WG 8.5 International Conference on Electronic Government (https://dgsociety.org/egov-2021/)*, 2021.

[3] M. Deakin and H. Al Waer, eds., *Smart cities: Governing, modelling and analysing the transition*. Routledge/ Taylor and Francis, 2013.

[4] L. Mora and R. Bolici, "How to become a smart city: Learning from Amsterdam," in *Smart and Sustainable Planning for Cities and Regions*, pp. 251–266, Springer International Publishing Switzerland 2017, 2017.

[5] L. Smith, "Amsterdam Smart City: A World Leader in Smart City Development," dec 2017.

[6] T. Bakıcı, E. Almirall, and J. Wareham, "A Smart City Initiative: The Case of Barcelona," *Journal of the Knowledge Economy*, vol. 4, pp. 135–148, jun 2013.

[7] R. R. Carmona and C. Martínez, "El presupuesto participativo como herramienta de transformación social, política e institucional. un balance en el escenario argentino reciente," 2013.

[8] U. De Souza and T. Genro, "Presupuesto participativo. la experiencia de porto alegre," 1999.

[9] C. D. Manning, M. Surdeanu, J. Bauer, J. Finkel, S. J. Bethard, and D. McClosky, "The Stanford CoreNLP natural language processing toolkit," in *Association for Computational Linguistics (ACL) System Demonstrations*, pp. 55–60, 2014.

[10] I. Cantador, M. E. Cortés-Cediel, and M. Fernandez, "Exploiting open data to analyze discussion and controversy in online citizen participation," *Information Processing & Management*, vol. 57, no. 5, p. 102301, 2020.

[11] I. Cantador, D. Elliott, and J. M. Jose, "A case study of exploiting data mining techniques for an industrial recommender system," 2009.

[12] M. D. Molina-González, E. Martínez-Cámara, M.-T. Martín-Valdivia, and J. M. Perea-Ortega, "Semantic orientation for polarity classification in spanish reviews," *Expert Systems with Applications*, vol. 40, no. 18, pp. 7250–7257, 2013.

[13] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: Machine learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

# Definiciones

**comentario** En el ámbito de la plataforma Decide Madrid, es una opinión de otro usuario acerca de una propuesta. En la plataforma, permiten que otros usuarios contesten a otros comentarios, e incluso votar a favor o en contra de los mismos.

**controversia** Mide el nivel de debate que puede haber, en este caso, en las propuestas ciudadanas propuestas.

**democracia electrónica** Tipo de democracia en la que se fomenta el uso de las TIC para crear espacios de debate sobre temas políticos y la mejora de la relación entre los ciudadanos con los políticos o gobernantes ( C2G ), así como una aproximación a la democracia directa.

**presupuesto participativo** Son procesos de decisión en los que los gobiernos, ayuntamientos o autoridades competentes delegan parte de responsabilidad a la ciudadaná para puedan decidir la elección los proyectos y recursos públicos.

**propuesta ciudadana** En el ámbito de la plataforma Decide Madrid, es cada uno de los problemas o sugerencias sobre la ciudad de Madrid que pueden crear los usuarios. Posteriormente, otros usuarios podrán expresar su opinión mediante comentarios, y solamente los usuarios que residan en el Municipio de Madrid podrán votar dichas propuestas.

**relevancia** Mide el interés que tiene, en este contexto, una propuesta ciudadana en el ámbito bien de la ciudad, o bien en algún distrito o barrio en particular.

**voto** En el ámbito de la plataforma Decide Madrid, es cada uno de los apoyos directos que hace un usuario, principalmente a una propuesta. Para poder votar propuestas ciudadanas, los usuarios deben ser mayores de 16 años y residir en el municipio de Madrid. Por otro lado, los comentarios también tienen votos, tanto positivos como negativos, aunque en este caso cualquier usuario puede votar a favor o en contra de un comentario.

# Acrónimos

**BSCCNS** Barcelona Supercomputing Center  Centro Nacional de Supercomputación.

**C2G** Citizen To Government.

**IDE** Integrated Development Environment - Entrono de Desarrollo Integrado.

**LDA** Linear Discriminant Analysis - Análisis Discriminante Lineal.

**ML** Machine Learning - Aprendizaje Automático.

**NIMBY** Not In My BackYard - No en mi patio.

**NPL** Natural Processing Language - Procesamiento del Lenguaje Natural.

**PCA** Principal Component Analysis - Análisis de componentes principales.

**QDA** Quadratic Discriminant Analysis - Análisis Discriminante Cuadrático.

**TFG** Trabajo Final de Grado.

**TIC** Tecnologías de la Información y de la Comunicación.

# Apéndices

# A

# Distribución de las propuestas: Número de votos y comentarios

En el siguiente apéndice se detallan las gráficas que corresponden con la distribución de las propuestas en cuanto al número de propuestas que conienen un determinado número de votos y de comentarios.


**Figura A.1:** Distribución del número de votos obtenidos por las propuestas del primer periodo anual (septiembre 2015 - agosto 2016)

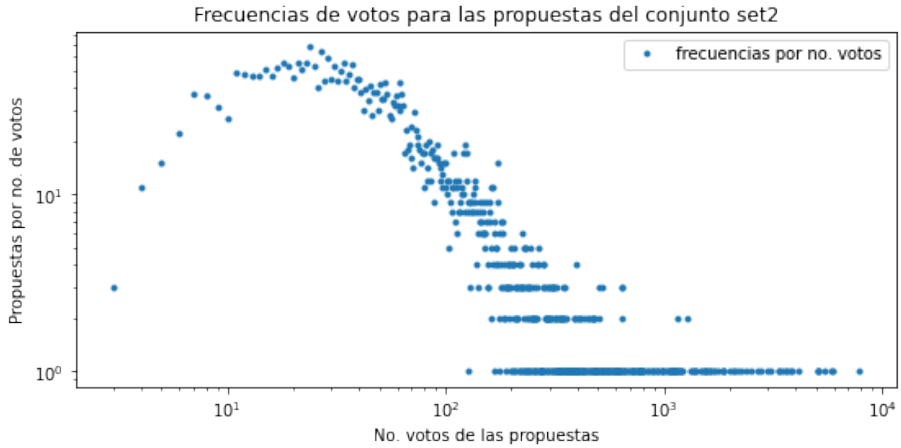



**Figura A.2:** Distribución del número de votos obtenidos por las propuestas del segundo periodo anual (septiembre 2016 - agosto 2017)

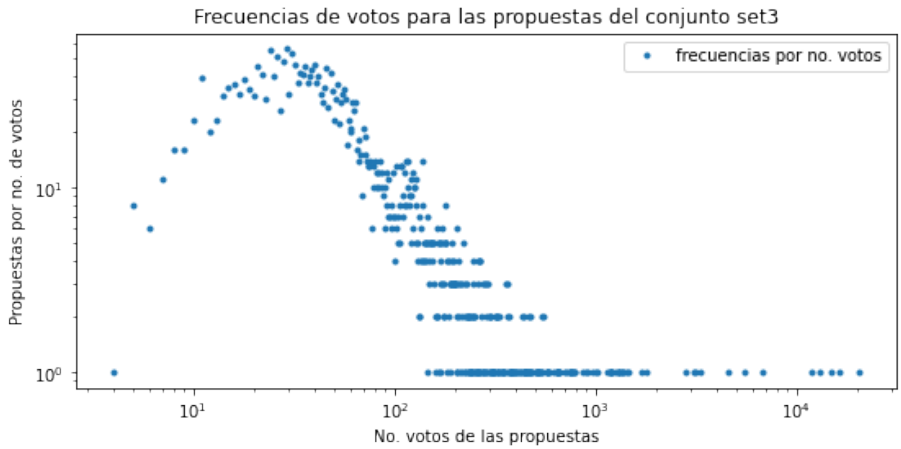



**Figura A.3:** Distribución del número de votos obtenidos por las propuestas del tercer periodo anual (septiembre 2017 - agosto 2018)

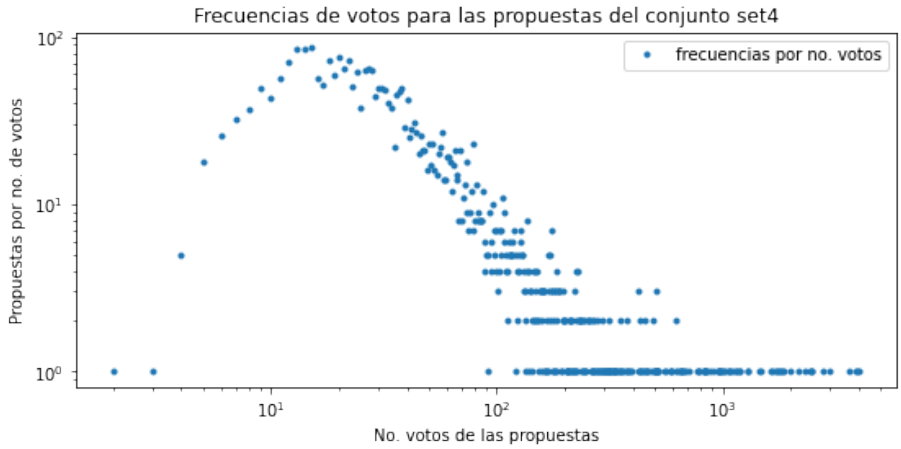



**Figura A.4:** Distribución del número de votos obtenidos por las propuestas del cuarto periodo anual (septiembre 2018 - agosto 2019)

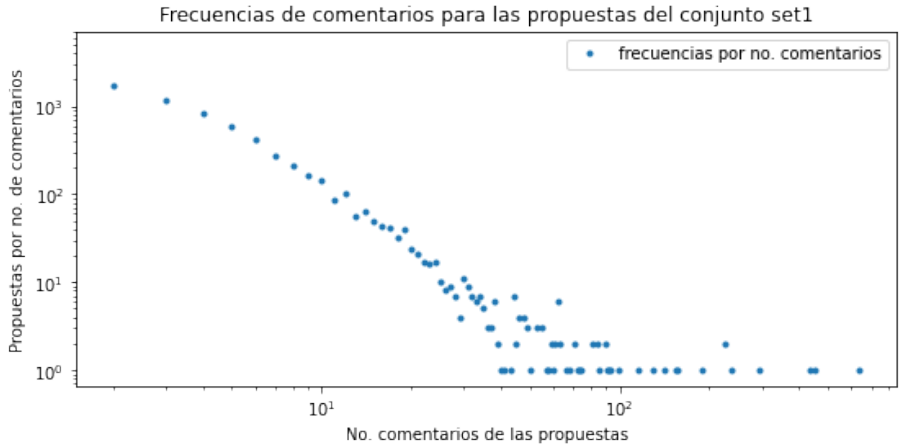



**Figura A.5:** Distribución del número de comentarios obtenidos por las propuestas del primer periodo anual (septiembre 2015 - agosto 2016)

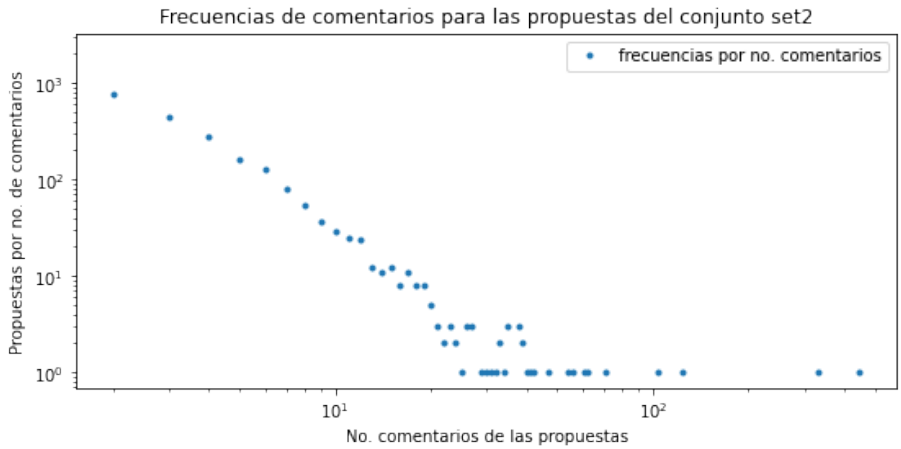



**Figura A.6:** Distribución del número de comentarios obtenidos por las propuestas del segundo periodo anual (septiembre 2016 - agosto 2017)

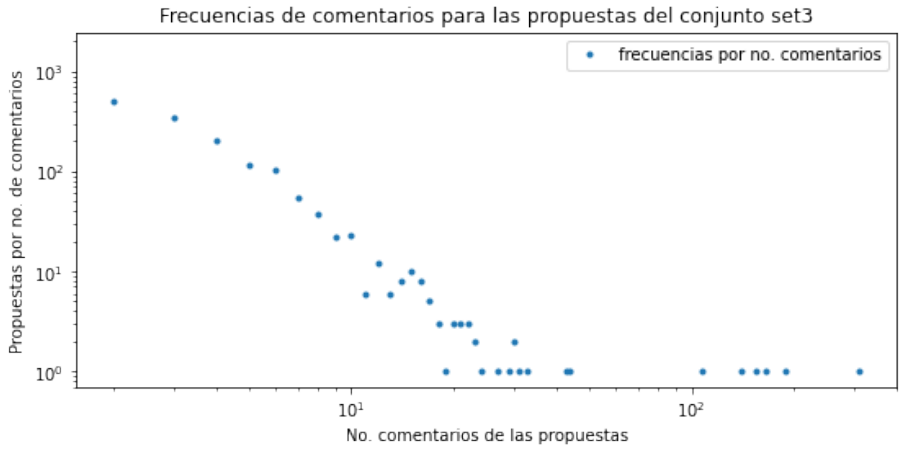



**Figura A.7:** Distribución del número de comentarios obtenidos por las propuestas del tercer periodo anual (septiembre 2017 - agosto 2018)

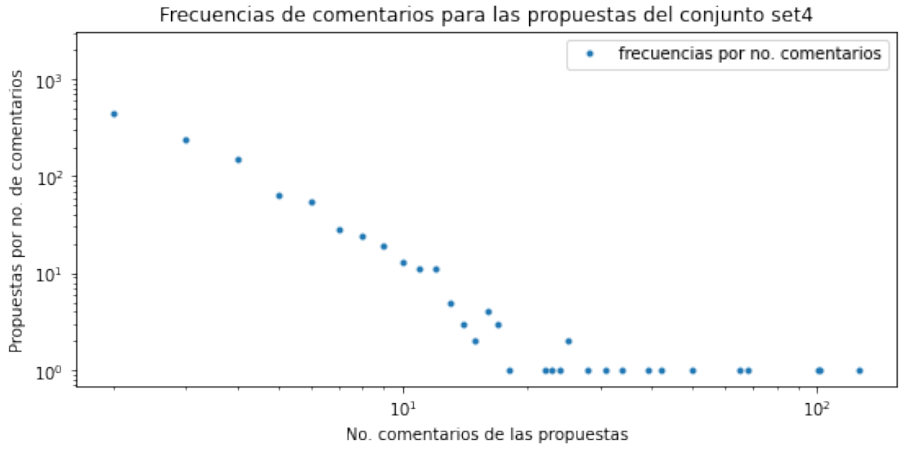



**Figura A.8:** Distribución del número de comentarios obtenidos por las propuestas del cuarto periodo anual (septiembre 2018 - agosto 2019)

# B

# Distribución de los atributos de texto

En el siguiente apéndice se detallan las gráficas que corresponden con la distribución de los nueve atributos textuales que se han analizado con ayuda del software Stanford CoreNPL [9] :

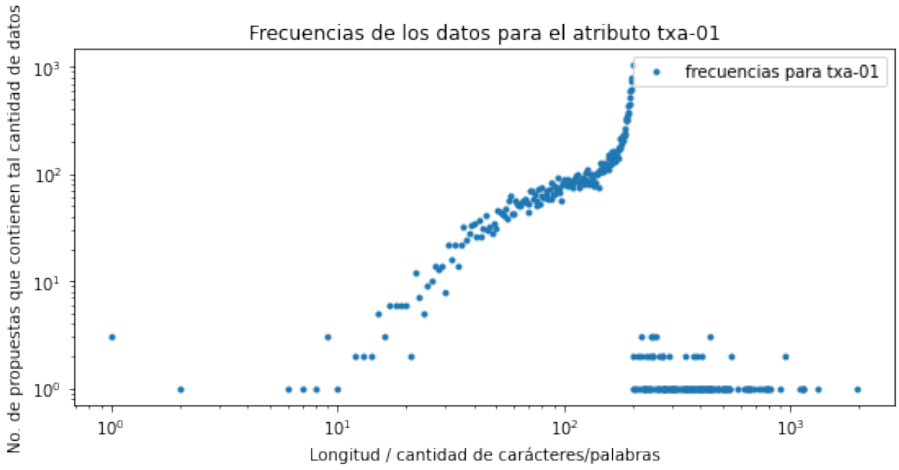



**Figura B.1:** Distribución del atributo correspondiente a la longitud del resumen de las propuestas.

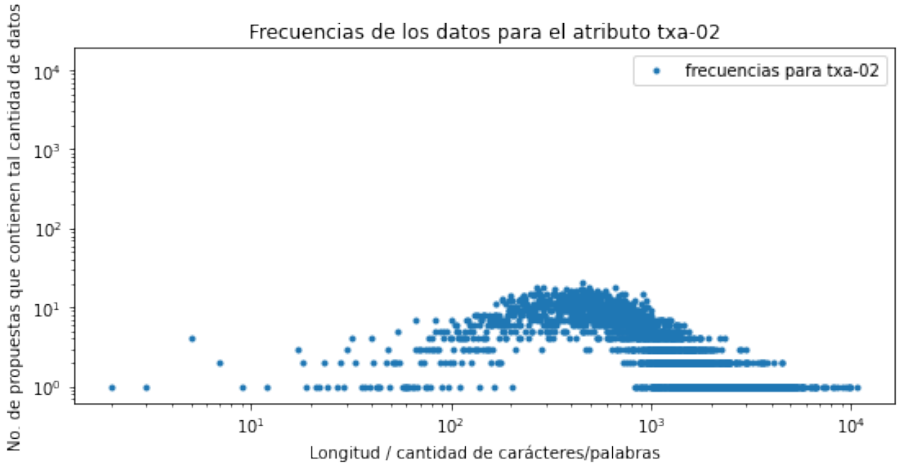



**Figura B.2:** Distribución del atributo correspondiente a la longitud del contenido de las propuestas.

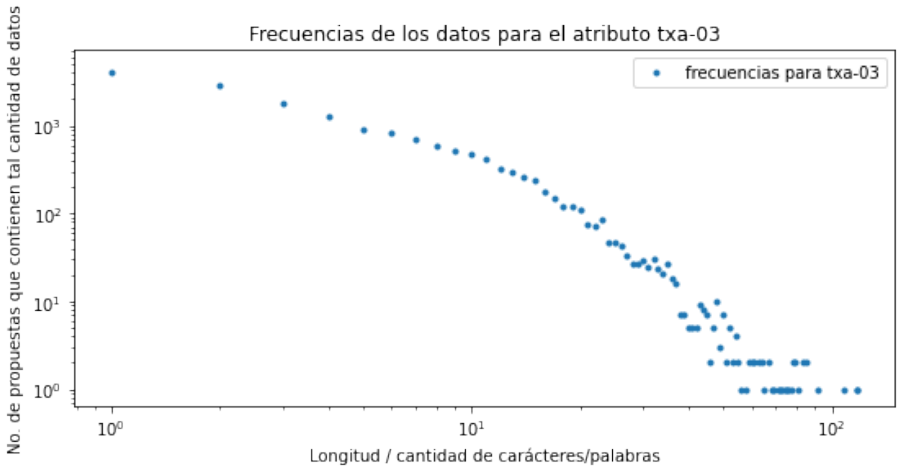



**Figura B.3:** Distribución del atributo correspondiente al número de nombres propios en las propuestas.

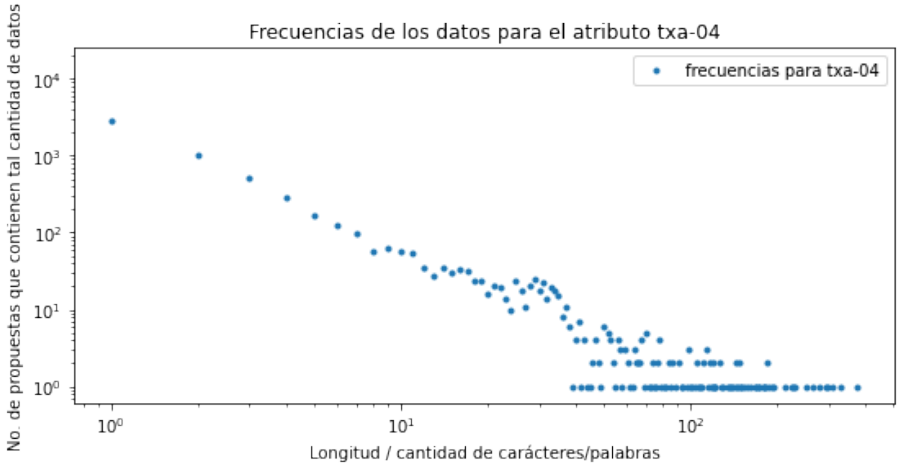



**Figura B.4:** Distribución del atributo correspondiente al número de palabras escritas en mayúsculas en las propuestas.

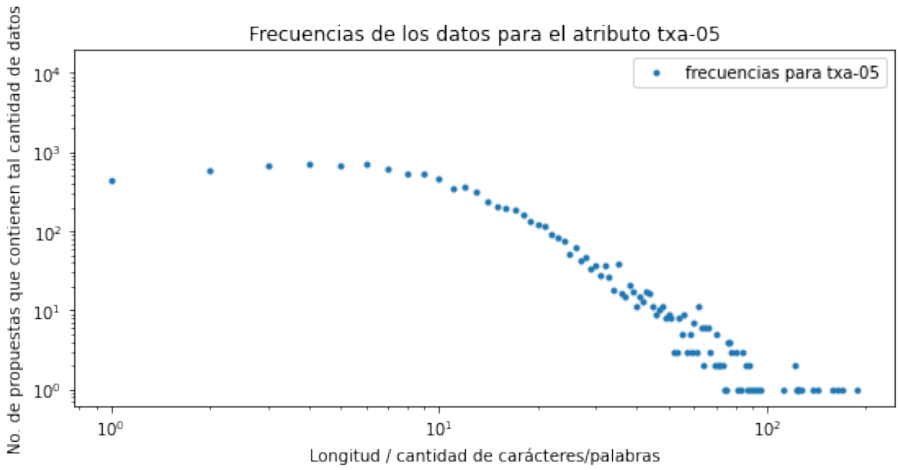



**Figura B.5:** Distribución del atributo correspondiente al número de adjetivos empleados en las propuestas.

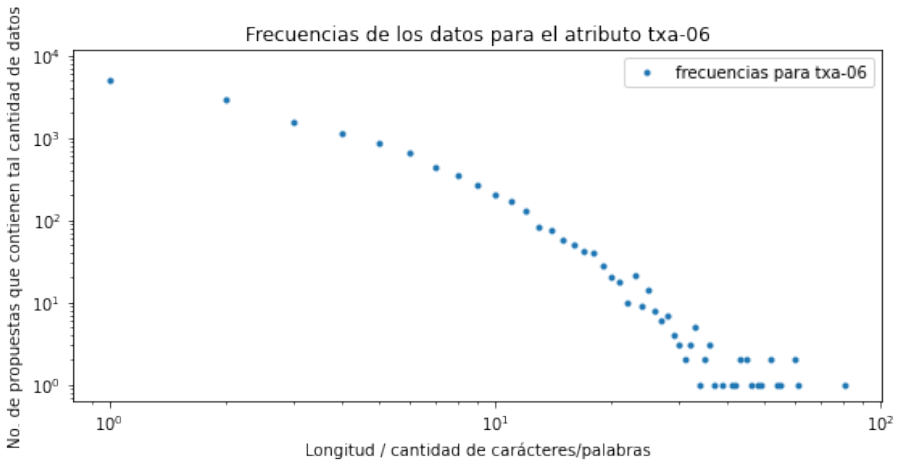



**Figura B.6:** Distribución del atributo correspondiente al número de palabras positivas que aparecen en las propuestas.

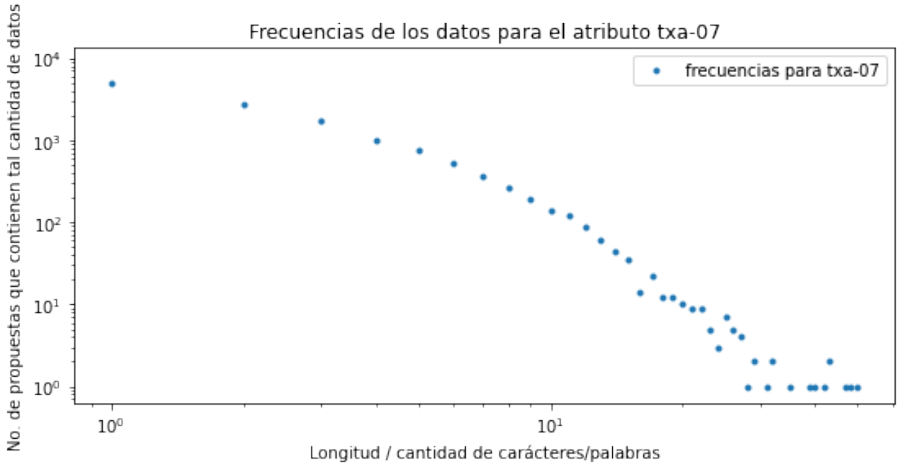



**Figura B.7:** Distribución del atributo correspondiente al número de palabras negativas que aparecen en las propuestas.

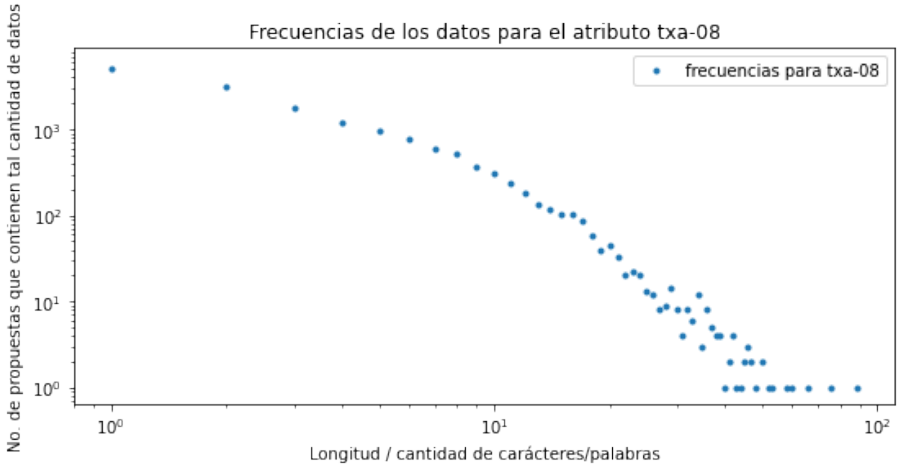



**Figura B.8:** Distribución del atributo correspondiente al número de determinantes y pronombres personales y posesivos de cada propuesta.

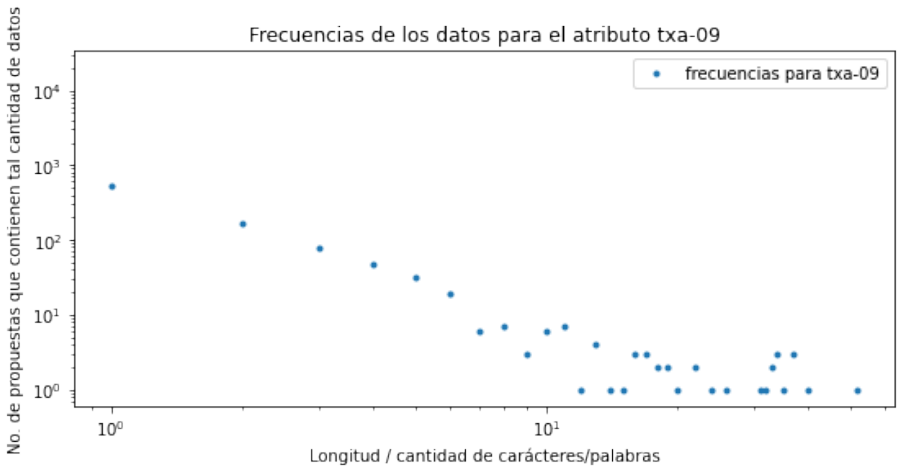
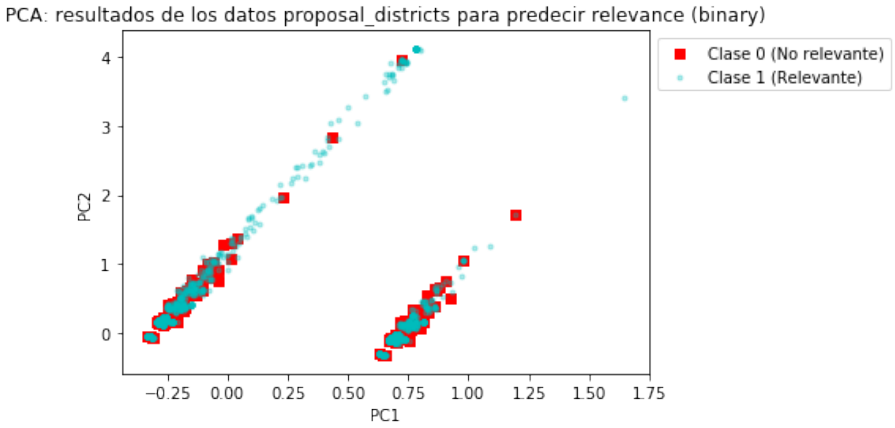



**Figura B.9:** Distribución del atributo correspondiente al número de enlaces web en las propuestas.

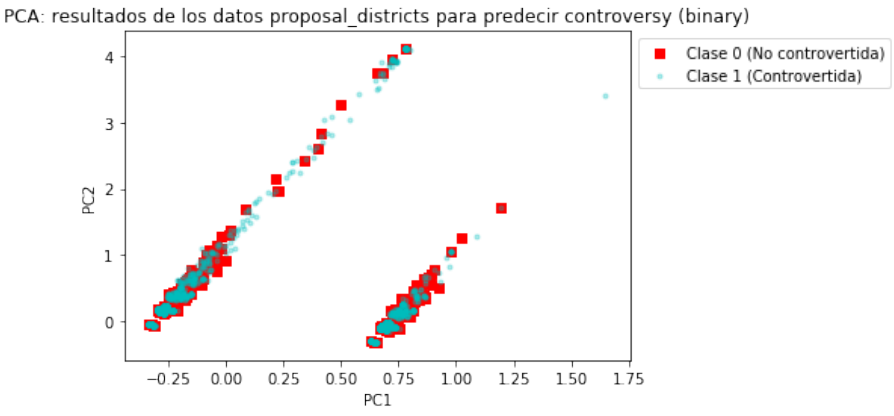
# C

# Gráficas PCA

En este apéndice se muestran las gráficas obtenidas para mostrar los valores de PCA obtenidos en las distintas iteraciones.

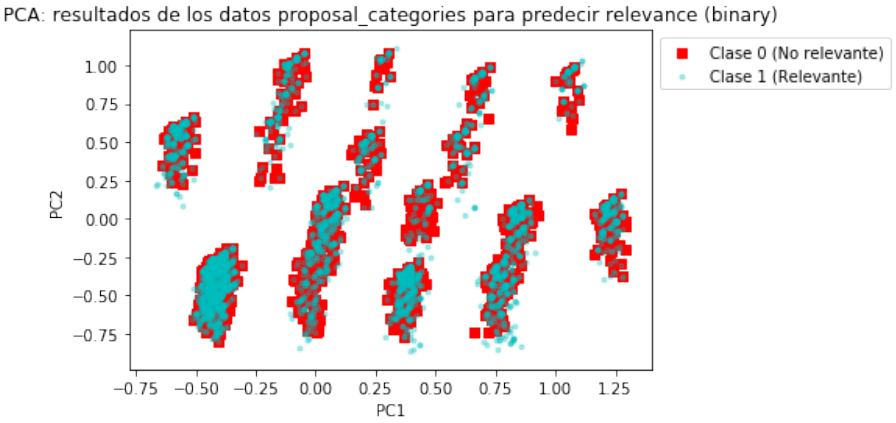



(a)

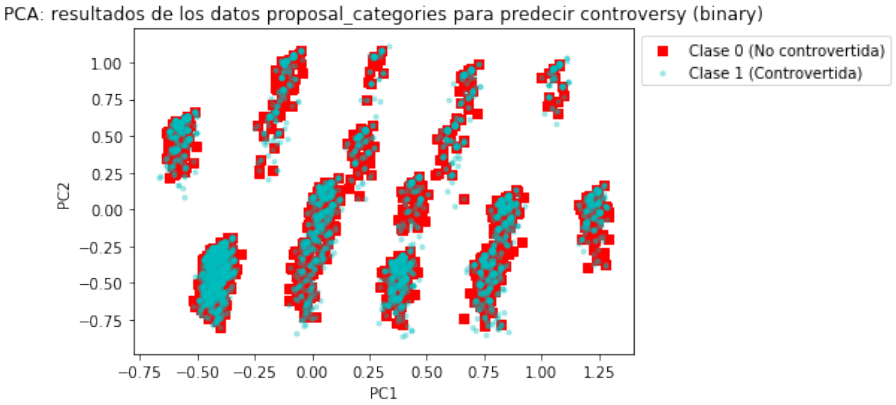



(b)

**Figura C.1:** Gráfica de relevancia y controversia de PCA para valores binarios, ejecutada con todos los atributos del conjunto districts.

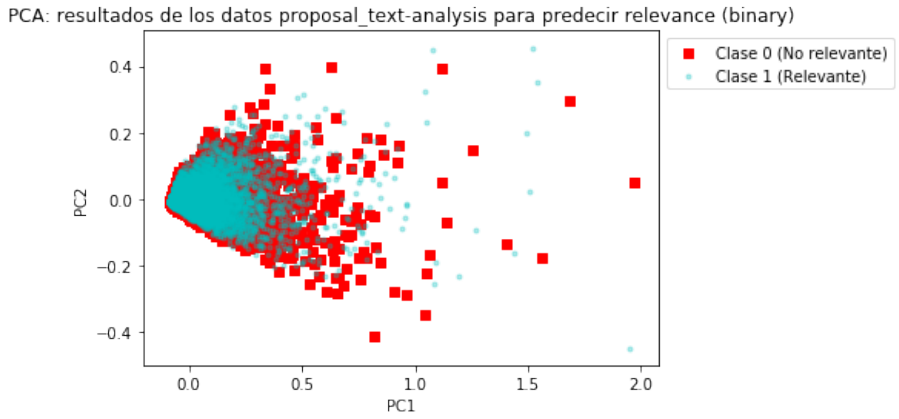



(a)

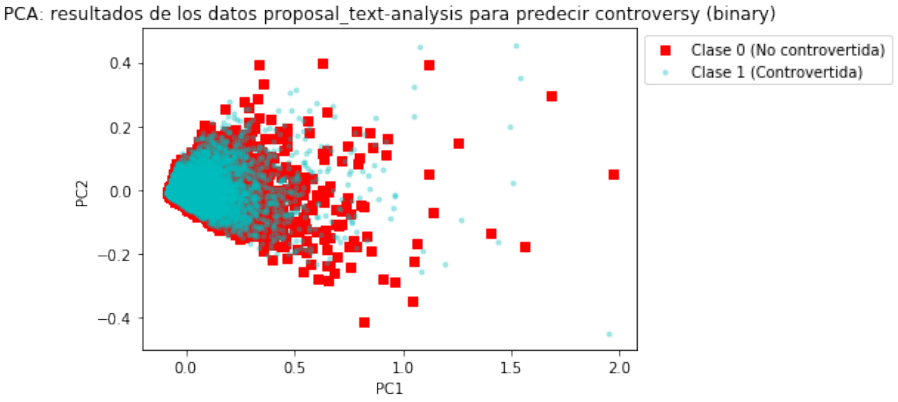



(b)

**Figura C.2:** Gráfica de relevancia y controversia de PCA para valores binarios, ejecutada con todos los atributos del conjunto categories.

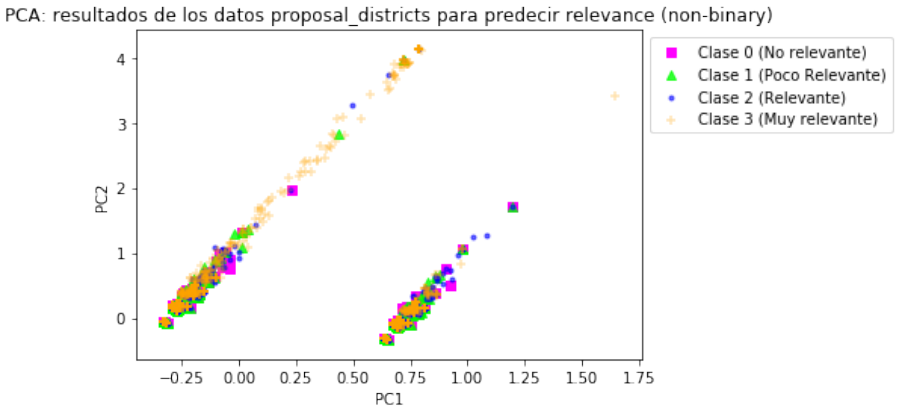



(a)

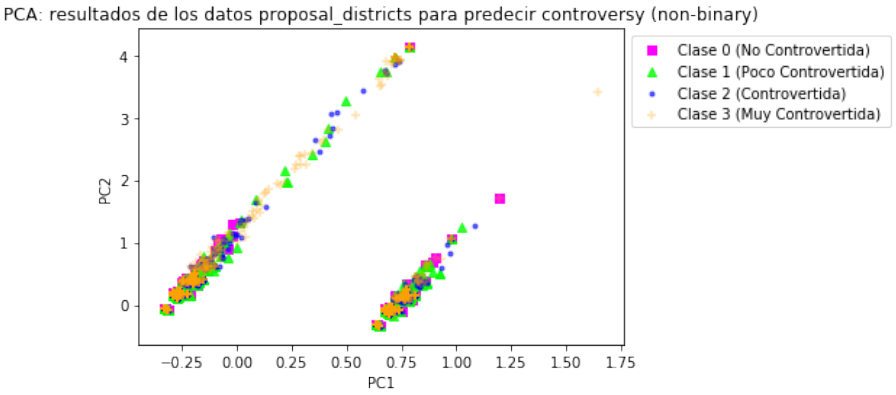



(b)

**Figura C.3:** Gráfica de relevancia y controversia de PCA para valores binarios, ejecutada con todos los atributos del conjunto text analysis.

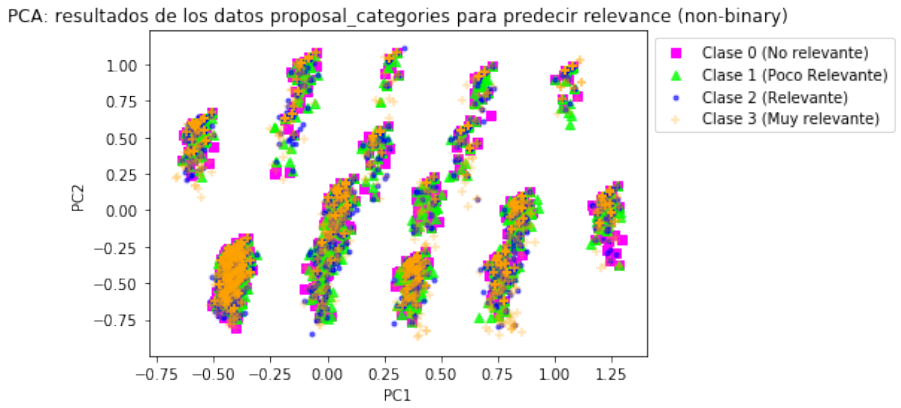



(a)

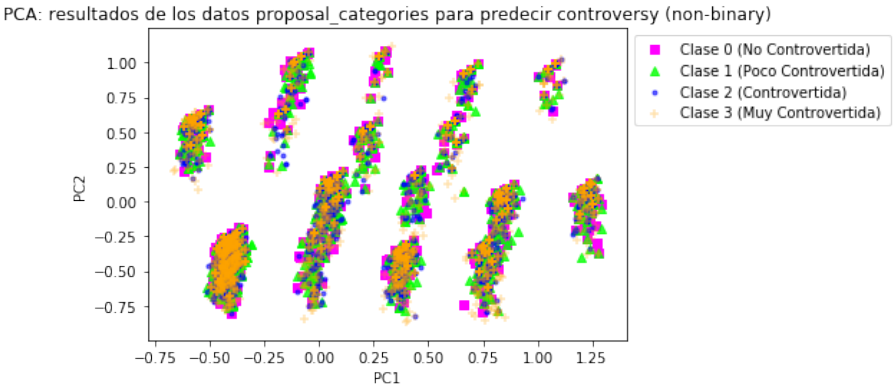



(b)

**Figura C.4:** Gráfica de relevancia y controversia de PCA para valores multiclase, ejecutada con todos los atributos del conjunto districts.

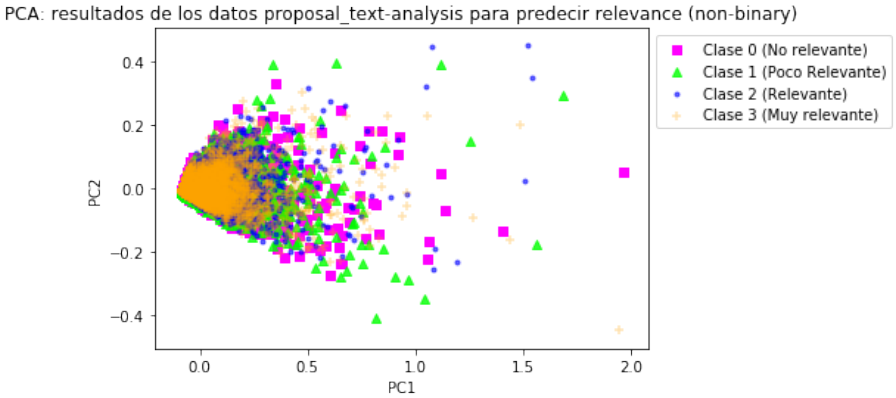



(a)

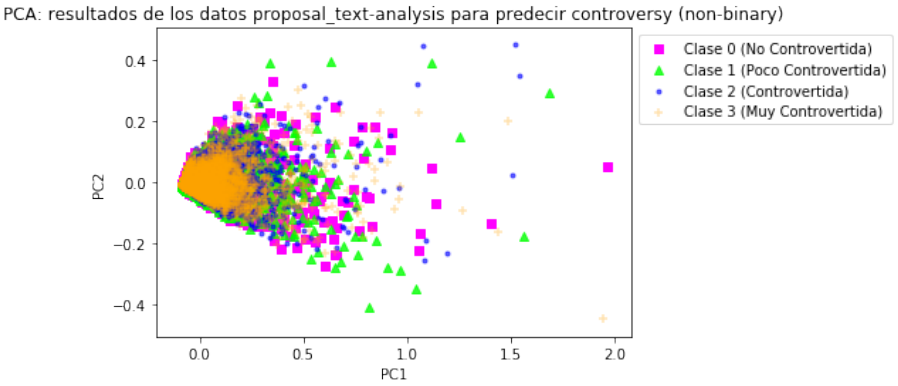



(b)

**Figura C.5:** Gráfica de relevancia y controversia de PCA para valores multiclase, ejecutada con todos los atributos del conjunto categories.

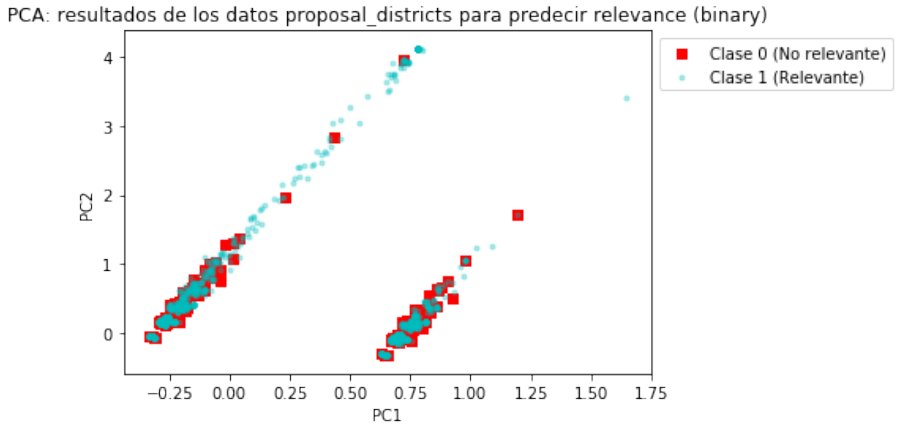



(a)

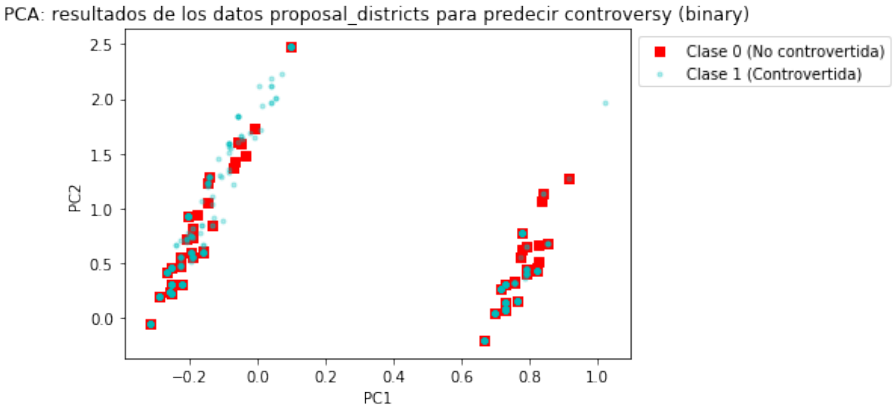



(b)

**Figura C.6:** Gráfica de relevancia y controversia de PCA para valores multiclase, ejecutada con todos los atributos del conjunto text analysis.

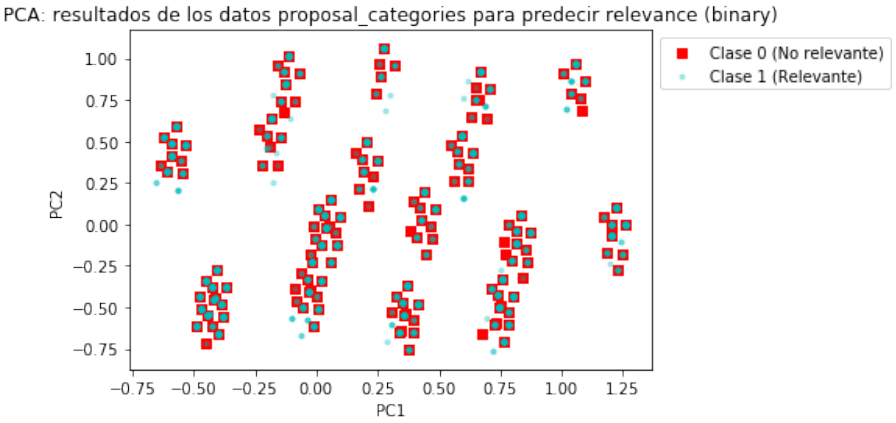



(a)

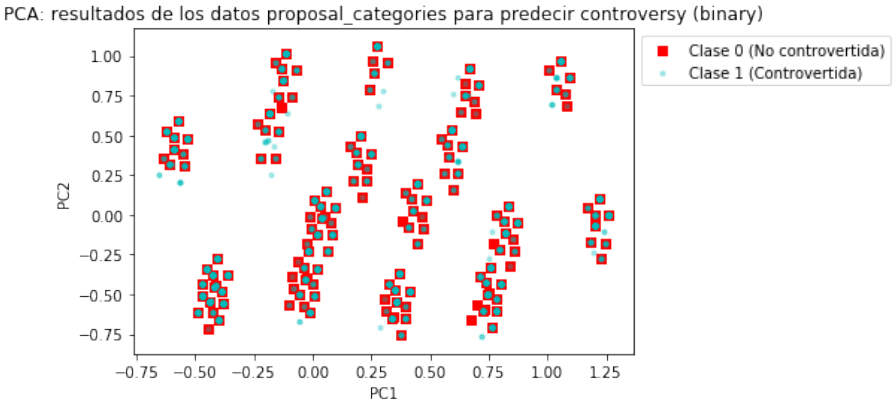



(b)

**Figura C.7:** Gráfica de relevancia y controversia de PCA para valores multiclase, ejecutada con la primera reducción del conjunto districts.

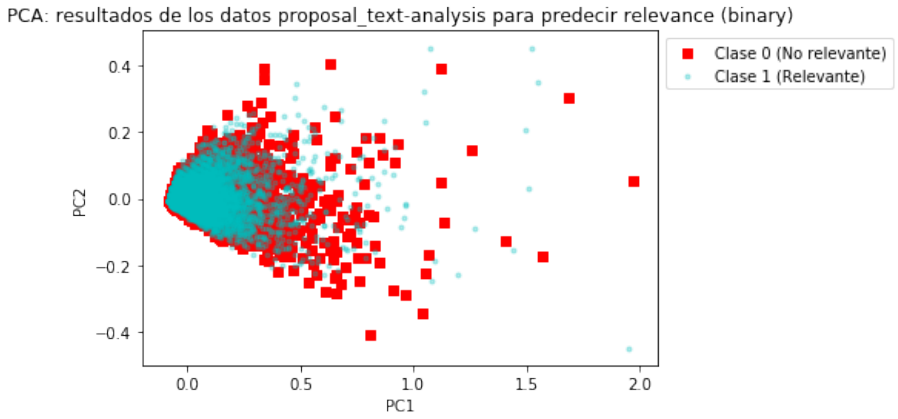



(a)

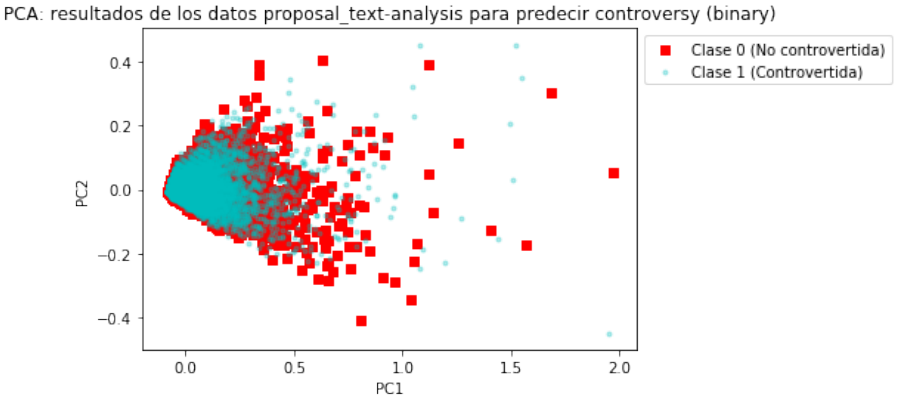



(b)

**Figura C.8:** Gráfica de relevancia y controversia de PCA para valores multiclase, ejecutada con la primera reducción del conjunto categories.

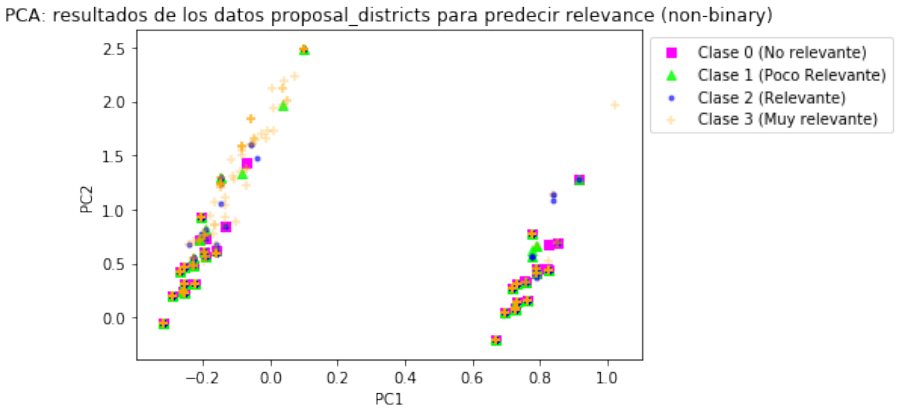



(a)

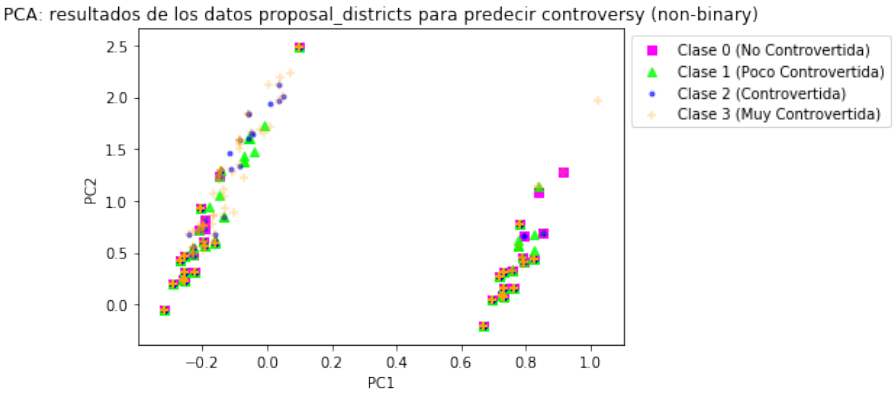



(b)

**Figura C.9:** Gráfica de relevancia y controversia de PCA para valores multiclase, ejecutada con la primera reducción del conjunto text analysis.

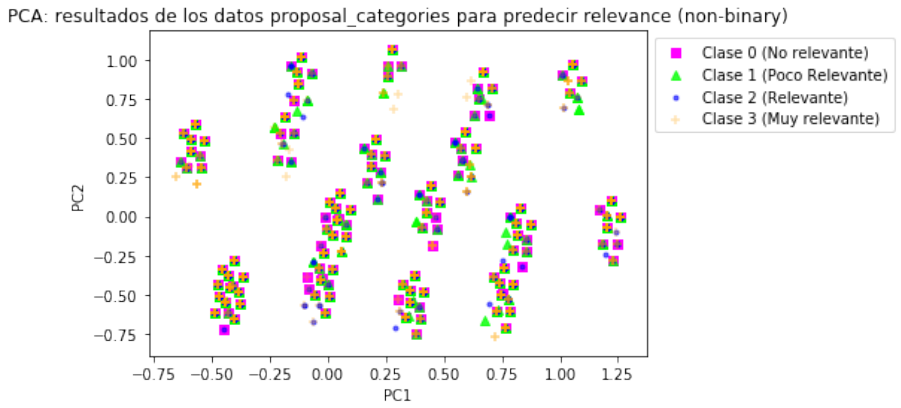



(a)

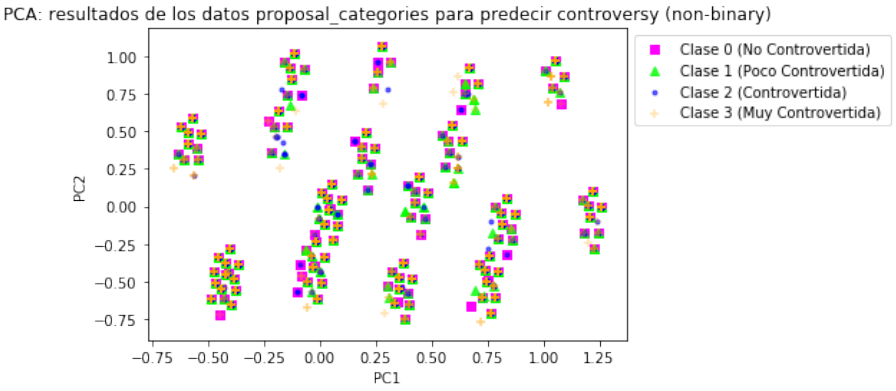



(b)

**Figura C.10:** Gráfica de relevancia y controversia de PCA para valores multiclase, ejecutada con la primera reducción del conjunto districts.

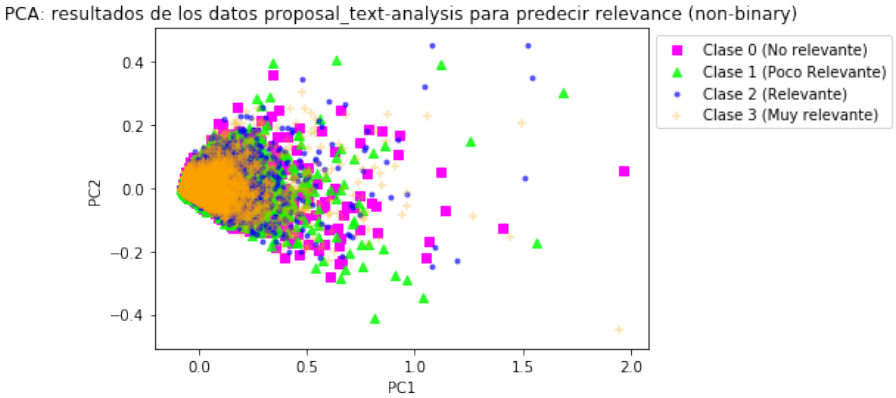



(a)

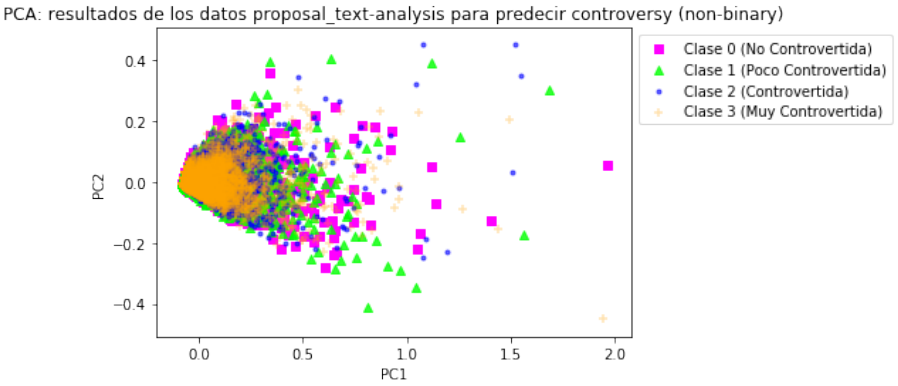



(b)

**Figura C.11:** Gráfica de relevancia y controversia de PCA para valores multiclase, ejecutada con la primera reducción del conjunto categories.

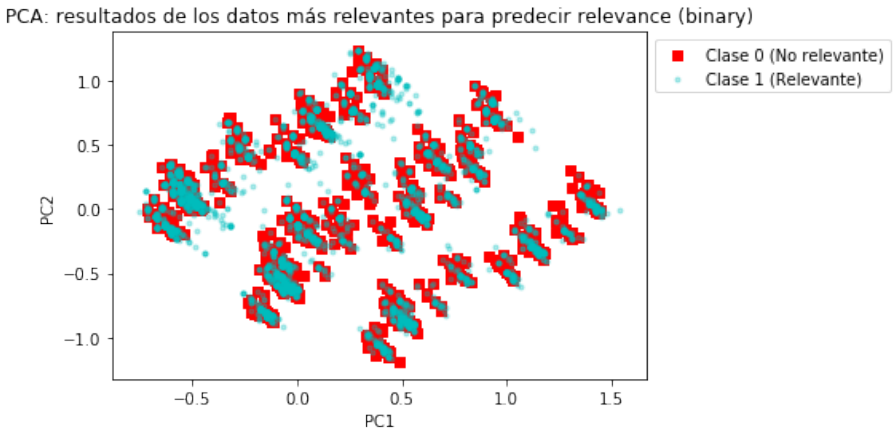



(a)

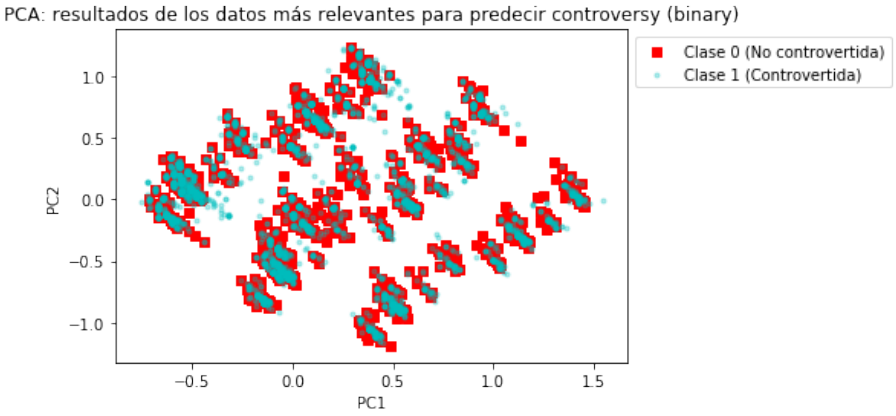



(b)

**Figura C.12:** Gráfica de relevancia y controversia de PCA para valores multiclase, ejecutada con la primera reducción del conjunto text analysis.

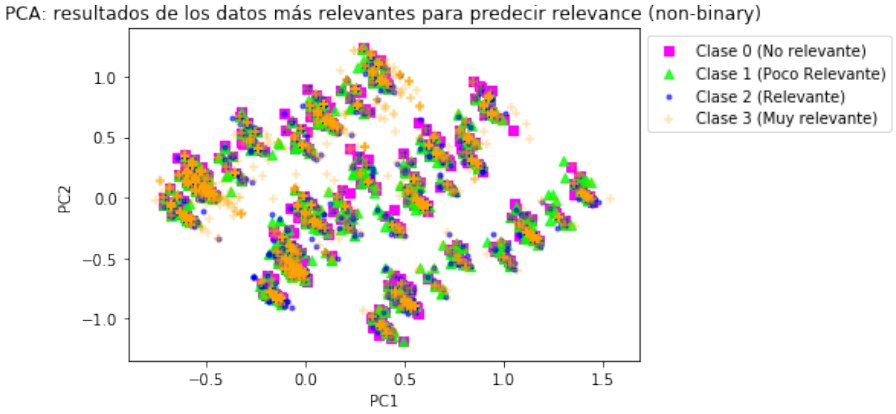



(a)

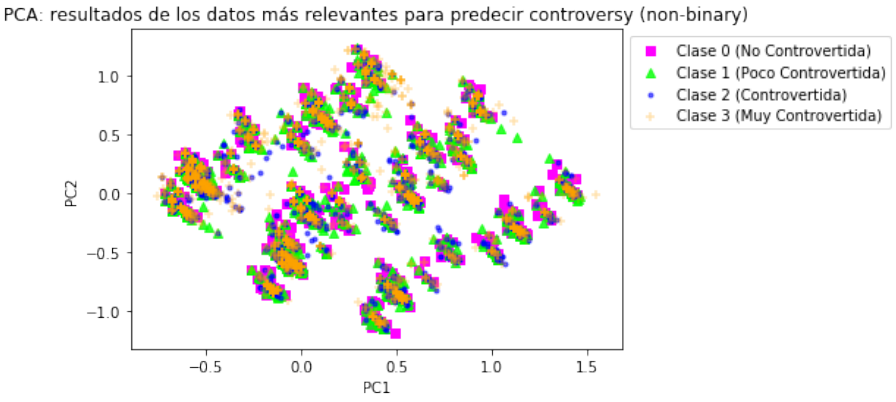



(b)

**Figura C.13:** Conjunto de gráficas de PCA para todos los conjuntos de atributos, con solamente los seis atributos más relevantes de cada conjunto (Primera pasada).

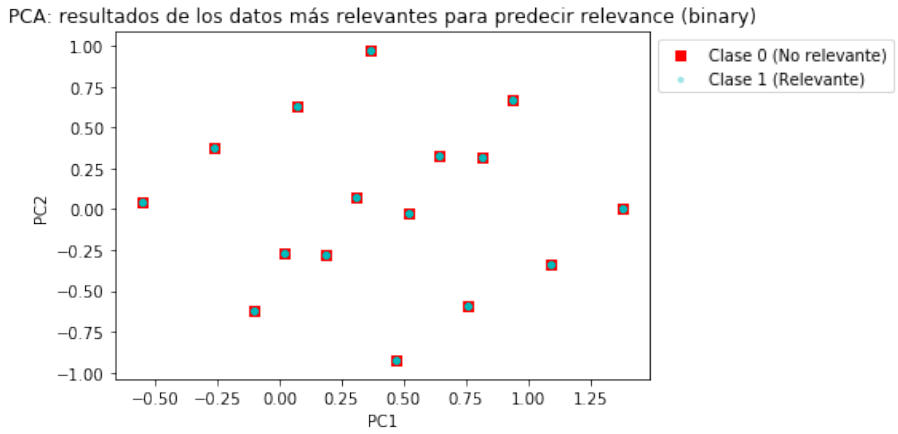



(a)

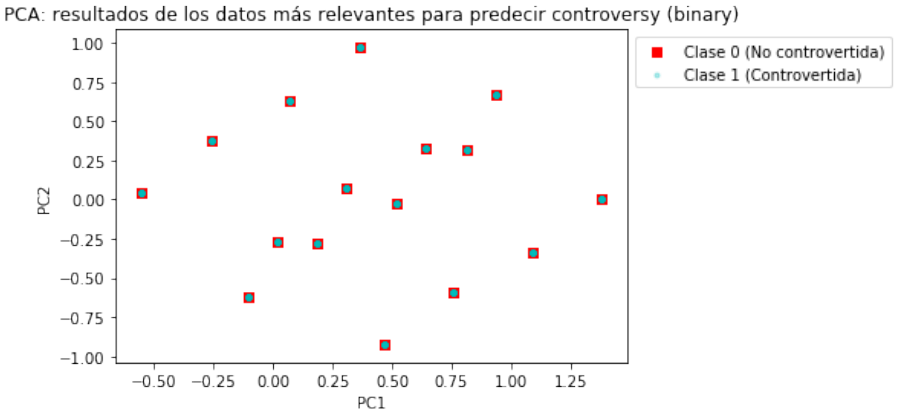



(b)

**Figura C.14:** Conjunto de gráficas de PCA para todos los conjuntos de atributos, con solamente los seis atributos más relevantes de cada conjunto (Primera pasada).

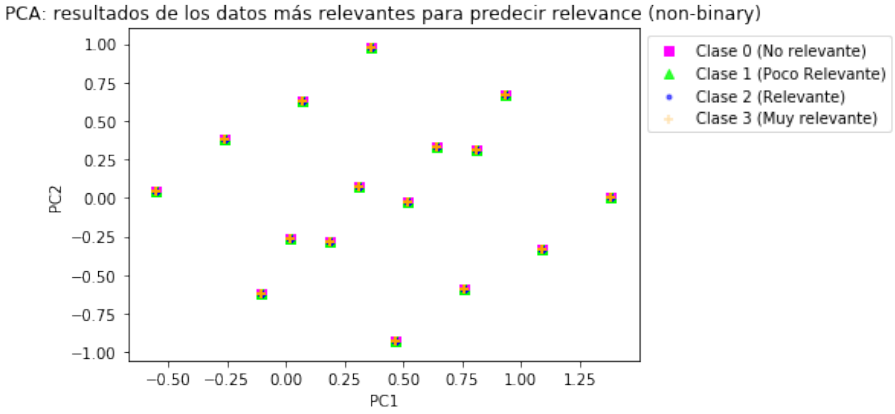



(a)

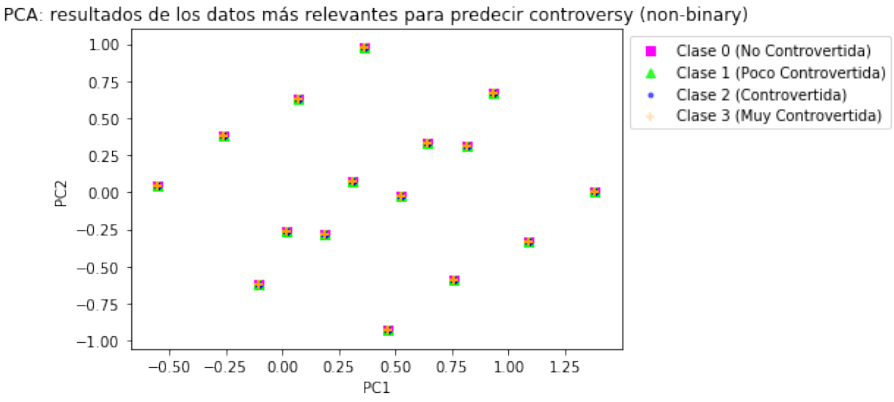



(b)

**Figura C.15:** Conjunto de gráficas de PCA para solamente los cuatro atributos más relevantes de todos (Segunda pasada).




(a)




(b)

**Figura C.16:** Conjunto de gráficas de PCA para solamente los cuatro atributos más relevantes de todos (Segunda pasada).

# D

# Gráficas LDA

En esta sección se muestran los resultados de graficar el modelo LDA para las dos componentes más relevantes.

Se adjuntan solamente las imágenes para la clasificación de relevancia, puesto que las gráficas del modelo para la clase controversia son muy parecidos, y con la clasificación de valores multiclase, puesto que para dos clases el modelo proyecta los datos en una recta (ya que reduce la dimensionalidad de clases de 2 a 1).

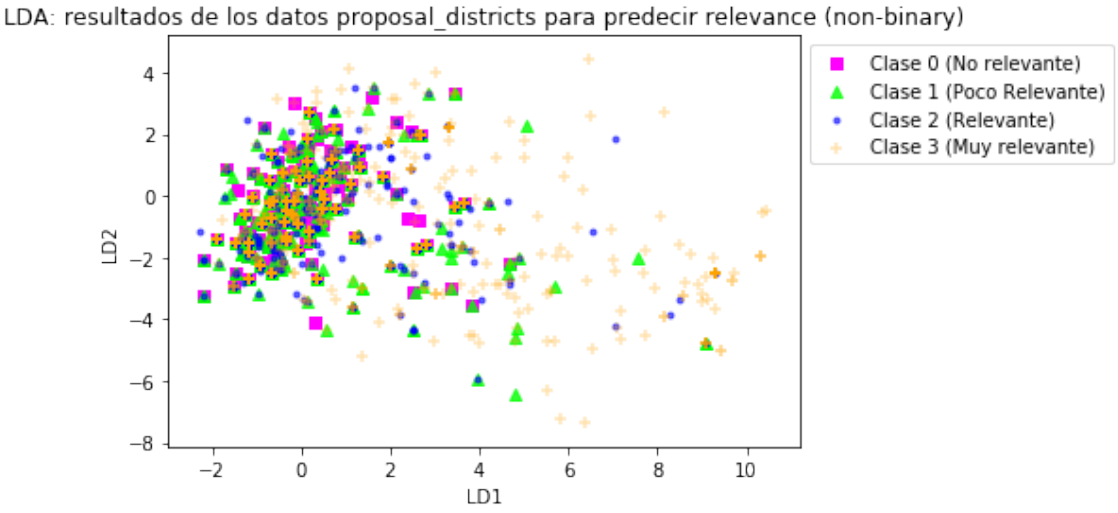



**Figura D.1:** Gráficas obtenidas con LDA para valores multiclase. Comparación de las dos componentes más relevantes. Muestra de los atributos del conjunto districts.

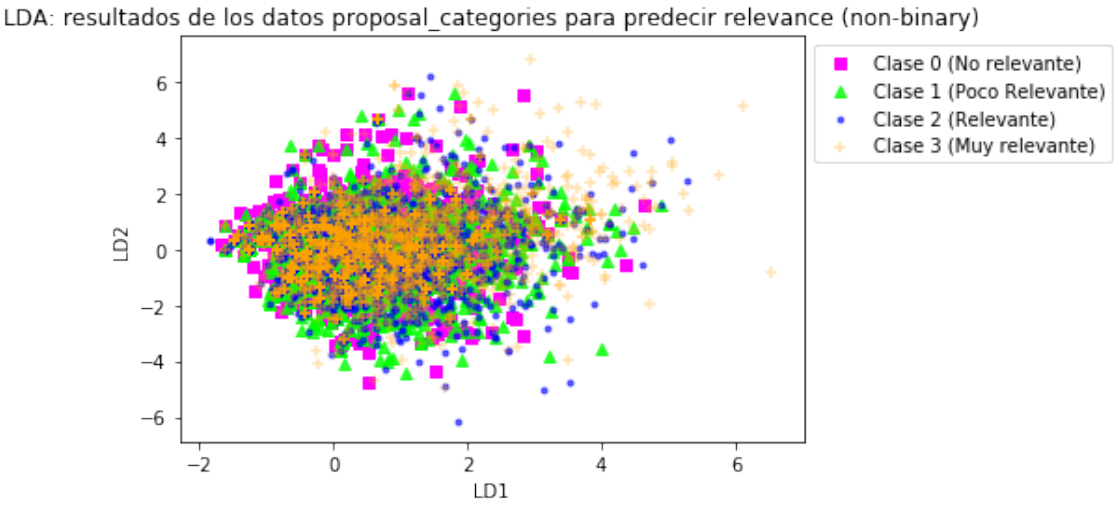



**Figura D.2:** Gráficas obtenidas con LDA para valores multiclase. Comparación de las dos componentes más relevantes. Muestra de los atributos del conjunto categories.

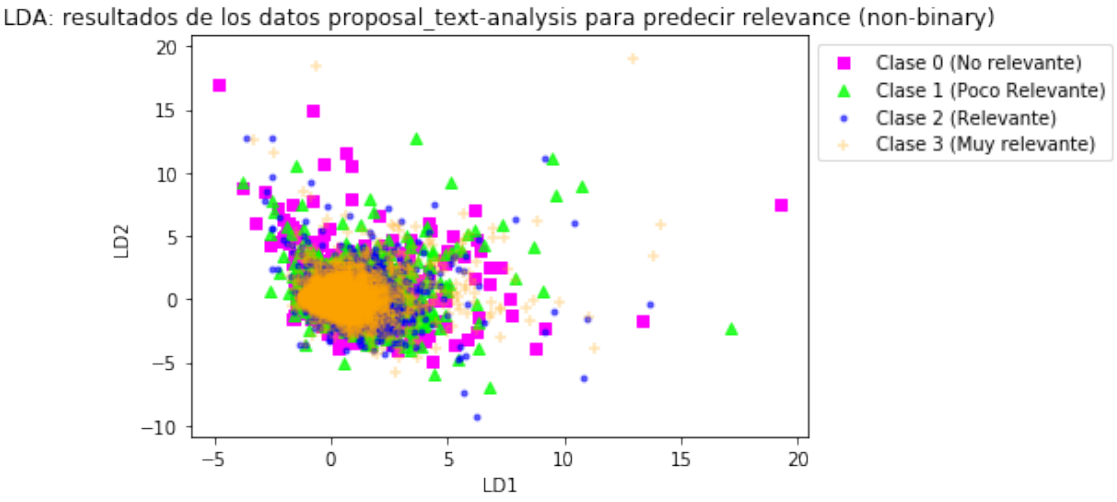



**Figura D.3:** Gráficas obtenidas con LDA para valores multiclase. Comparación de las dos componentes más relevantes.Muestra de los atributos del conjunto text analysis.

# E

# Código utilizado para dividir y clasificar los conjuntos de entrenamiento y clasificación

**Código E.1:** Estrategia 1 de particionado de datos en entrenamiento y test.

```
144     if id_crossval_method == 1:
145         """
146         Método 1: dentro de cada set
147             para cada set:
148                 train <- 3 / 4 / 6 / 8 / 9 primeros meses (comenzando desde septiembre)
149                 test <- 9 / 8 / 6 / 4 / 3 últmos meses (con agosto como último mes)
150 
151                 hacer la media para cada clasificación entre sets
152 
153             hacer a media de los 4 valores obtenidos de cada set (?)
154         """
155         list_train_months = [3, 4, 6, 8, 9]
156 
157         dict_month_dates = file_manager.load_date_proposals()
158 
159         for set_name in list_sets:
160 
161             for train_month in list_train_months:
162 
163                 # Obtención de los meses que compondrán cada conjunto train y test
164                 list_train = get_firsts_lasts_months(train_month, is_first=True)
165 
166                 train_indices = []
167                 test_indices = []
168 
169                 # Recorre todos los meses
170                 for month in range(1, 13):
171                     # Si un mes no está en el conjunto train, estrará en el conjunto test y viceversa
172                     if month in list_train:
173                         train_set_aux =
                                dict_month_dates[set_name].index[dict_month_dates[set_name] ==
                                month].tolist()
174                         for i in train_set_aux:
175                             train_indices.append(i)
176                     else:
177                         test_set_aux =
                                dict_month_dates[set_name].index[dict_month_dates[set_name] ==
                                month].tolist()
178                         for i in test_set_aux:
179                             test_indices.append(i)
180 
181                 # print(f"Train data: {len(train_indices)} vs Test data: {len(test_indices)}
                        ({train_month}/{12-train_month})")
182 
183                 # Obtención de los atributos para train y test
184                 dict_values['X_train'].append(scaled_data_sets[set_name].iloc[train_indices])
185                 dict_values['y_train'].append(scaled_data_sets_class[set_name][class_problem].iloc[train_indices])
186 
187                 dict_values['X_test'].append(scaled_data_sets[set_name].iloc[test_indices])
188                 dict_values['y_test'].append(scaled_data_sets_class[set_name][class_problem].iloc[test_indices])
```

**Código E.2:** Estrategia 2 de particionado de datos en entrenamiento y test.

```
190     elif id_crossval_method == 2:
191         """
192         Método 2: entre sets
193             para cada set (se excluye el set de todas las propuestas):
194                 train <- sets - {set}
195                 test <- {set}
196 
197             hacer la media con todos los resultados
198         """
199         for set_name_test in list_sets:
200             # conjunto de train, formado por 3 sets, quitando el que va a actuar como set de test
201             train_sets = pdt.list_sets[1:]
202             train_sets.remove(set_name_test)
203 
204             train_indices = []
205             for train_set in train_sets:
206                 start_index, end_index = get_start_end_index_set(train_set)
207                 [train_indices.append(i) for i in list(scaled_data_sets[all_sets].iloc[start_index:
                        end_index].index)]
208 
209             start_index, end_index = get_start_end_index_set(set_name_test)
210             test_indices = [i for i in list(scaled_data_sets[all_sets].iloc[start_index:end_index].index)]
211 
212             # Obtención de los atributos para train y test
213             dict_values['X_train'].append(scaled_data_sets[all_sets].iloc[train_indices])
214             dict_values['y_train'].append(scaled_data_sets_class[all_sets][class_problem].iloc[train_indices])
215 
216             dict_values['X_test'].append(scaled_data_sets[all_sets].iloc[test_indices])
217             dict_values['y_test'].append(scaled_data_sets_class[all_sets][class_problem].iloc[test_indices])
```

**Código E.3:** Estrategia 3 (aleatoria) de particionado de datos en entrenamiento y test.

```
219     elif id_crossval_method == 3:
220         """
221         Método 3: dentro de cada set aleatorio
222             para cada set:
223                 train <- aprox 80% del set
224                 test <- aprox 20% del set
225 
226                 generar varios modelos
227                 hacer la media con todos los resultados de los sets
228 
229             hacer a media de los 4 valores obtenidos de cada set (?)
230         """
231         for set_name in list_sets:
232 
233             X = scaled_data_sets[set_name]
234             y = scaled_data_sets_class[set_name][class_problem]
235 
236             # Selección aleatoria 10 veces
237             for i in range(0, 10):
238                 X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2)
239 
240                 dict_values['X_train'].append(X_train)
241                 dict_values['y_train'].append(y_train)
242 
243                 dict_values['X_test'].append(X_test)
244                 dict_values['y_test'].append(y_test)
```

**Código E.4:** Estrategia 4 (aleatoria) de particionado de datos en entrenamiento y test.

```
    elif id_crossval_method == 4:
        """
        Método 4: aleatorio
            para todo el conjunto:
                train <- {todos los atributos} - {aleatorio} (aprox 80%)
                test <- {aleatorio} (aprox 20%)

                generar varios modelos

            hacer la media con todos los resultados
        """
        X = scaled_data_sets[all_sets]
        y = scaled_data_sets_class[all_sets][class_problem]

        # Selección aleatoria 10 veces
        for i in range(0, 10):
            X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2)

            dict_values['X_train'].append(X_train)
            dict_values['y_train'].append(y_train)

            dict_values['X_test'].append(X_test)
            dict_values['y_test'].append(y_test)

    return dict_values
```

**Código E.5:** Estrategias 1 y 2 que guardan los datos de clasificación para el conjunto de entrenamiento y test.

```
364 if id_val_method == 1:
365
366     f.write(f"\n===== MODELO 1 (separación de los datos por sets anuales, con 5 iteraciones
            por meses)\n\n")
367
368     # Iteración por cada set (4 sets)
369     for i in range(0, 4):
370         f.write(f"----------------------------------------\nDatos para {pdt.list_sets[i + 1]}\n")
371         range_set = i*5
372
373         # Iteración por cada problema de división train/test (3/9, 4/8, 6/6, 8/4 y 9/3 meses)
374         f.write(f"\n3 primeros meses train / 9 últimos meses test:\n")
375         get_dict_acc_values(f, dict_acc_vals, range_set)
376
377         f.write(f"\n4 primeros meses train / 8 últimos meses test:\n")
378         get_dict_acc_values(f, dict_acc_vals, range_set + 1)
379
380         f.write(f"\n6 primeros meses train / 6 últimos meses test:\n")
381         get_dict_acc_values(f, dict_acc_vals, range_set + 2)
382
383         f.write(f"\n8 primeros meses train / 4 últimos meses test:\n")
384         get_dict_acc_values(f, dict_acc_vals, range_set + 3)
385
386         f.write(f"\n9 primeros meses train / 3 últimos meses test:\n")
387         get_dict_acc_values(f, dict_acc_vals, range_set + 4)
388
389         average_val = round(np.average(dict_acc_vals["average"][range_set + 0:range_set + 4]), 4)
390         f.write(f"\nMedia aritmética para {pdt.list_sets[i + 1]}: {average_val}\n")
391
392         geomean_val = round(np.average(dict_acc_vals["geometric_mean"][range_set + 0:range_set
                + 4]), 4)
393         f.write(f"Media geométrica para {pdt.list_sets[i + 1]}: {geomean_val}\n")
394
395 elif id_val_method == 2:
396
397     f.write(f"\n===== MODELO 2 (selección de un set anual como test y el resto de sets como
            train)\n\n")
398
399     for i in range(0, 4):
400         f.write(f"Datos para set test {pdt.list_sets[i + 1]}:\n")
401         get_dict_acc_values(f, dict_acc_vals, i)
402
403     f.write(f"----------------------------------------\n")
```

**Código E.6:** Estrategias 3 y 4 que guardan los datos de clasificación para el conjunto de entrenamiento y test de forma aleatoria.

```
405     elif id_val_method == 3:
406
407         f.write(f"\n===== MODELO 3 (selección aleatoria por sets: 10 selecciones por cada set (40
                selecciones))\n\n")
408
409         for i in range(0, 4):
410             f.write(
411                 f"----------------------------------------\nSelección de {pdt.list_sets[i + 1]} para conjunto
                        test\n")
412
413             range_sets = i *10
414             for j in range(0, 10):
415                 f.write(f"Selección {j + 1}:\n")
416                 get_dict_acc_values(f, dict_acc_vals, range_sets + j)
417
418             average_val = round(np.average(dict_acc_vals["average"][range_sets + 0:range_sets + 9]), 4)
419             f.write(f"\nMedia aritmética obtenida para {pdt.list_sets[i + 1]} como test:
                    {average_val}\n")
420
421             geomean_val = round(np.average(dict_acc_vals["geometric_mean"][range_sets +
                        0:range_sets + 9]), 4)
422             f.write(f"Media geométrica obtenida para {pdt.list_sets[i + 1]} como test:
                    {geomean_val}\n")
423
424     elif id_val_method == 4:
425
426         f.write(f"\n===== MODELO 4 (selección aleatoria de todo el conjunto de propuestas (10
                selecciones))\n\n")
427
428         for i in range(0, 10):
429             f.write(f"Selección {i + 1}:\n")
430             get_dict_acc_values(f, dict_acc_vals, i)
431
432     f.write(f"----------------------------------------\n")
433
434     average_val = round(np.average(dict_acc_vals["average"]), 4)
435     f.write(f"Media aritmética total obtenida por este método: {average_val}\n")
436
437     geomean_val = round(np.average(dict_acc_vals["geometric_mean"]), 4)
438     f.write(f"Media geométrica total obtenida por este método: {geomean_val}\n")
439
440     f.close()
```

# F

# Consulta de las métricas de relevancia y controversia

Este apéndice muestra los valores de los distritos y categorías en función de la cantidad de propuestas que tienen asignadas, el número total de apoyos y de comentarios que tienen en total, y su valor medio de *aggregation*:

**Código F.1:** Consulta en la que se muestran los distritos con el número de propuestas a las que se les ha asignado dicho distrito, junto con las tres métricas que se han utilizado para asignar las clases de relevancia y controversia a las propuestas.

```
1 SELECT district, COUNT(*) AS num_proposals, SUM(numSupports) AS supports, SUM(numComments)
      AS comments, ROUND(AVG(aggregation), 4) AS avgAggregation
2 FROM `proposal_locations` AS pl JOIN `proposals` AS p ON pl.id = p.id
3 GROUP BY district ORDER BY COUNT(*) DESC;
```

| district | num_proposals | supports | comments | avgAggregation |
|---|---|---|---|---|
| Ciudad | 15107 | 2172774 | 54018 | 0.0272 |
| Fuencarral-El Pardo | 1476 | 332355 | 4879 | 0.0242 |
| Arganzuela | 1382 | 307186 | 5831 | 0.0277 |
| Hortaleza | 1103 | 353744 | 5839 | 0.0277 |
| Moncloa-Aravaca | 956 | 194947 | 3823 | 0.0257 |
| Carabanchel | 927 | 236113 | 4676 | 0.0271 |
| Puente de Vallecas | 918 | 259643 | 4352 | 0.0278 |
| San Blas-Canillejas | 916 | 229993 | 4634 | 0.0302 |
| Latina | 893 | 186822 | 4143 | 0.0258 |
| Villa de Vallecas | 862 | 300477 | 4719 | 0.0307 |
| Ciudad Lineal | 757 | 193694 | 3916 | 0.0291 |
| Usera | 709 | 193689 | 3705 | 0.0277 |
| Barajas | 668 | 235690 | 3835 | 0.0289 |
| Salamanca | 658 | 169965 | 3778 | 0.0316 |
| Chamartín | 627 | 185912 | 4194 | 0.0348 |
| Villaverde | 618 | 193916 | 3570 | 0.0284 |
| Centro | 586 | 87377 | 1979 | 0.0256 |
| Tetuán | 537 | 171035 | 3662 | 0.0336 |
| Chamberí | 516 | 179310 | 3504 | 0.0328 |
| Moratalaz | 448 | 181131 | 3331 | 0.0323 |
| Vicálvaro | 443 | 185096 | 3116 | 0.0320 |
| Retiro | 247 | 22097 | 530 | 0.0223 |

**Figura F.1:** Tabla en la que se muestran los valores de los distritos: propuestas asignadas, suma de apoyos y comentarios, y media de valores *aggregation* para cada distrito.

**Código F.2:** Consulta en la que se muestran las categorías con el número de propuestas a las que se les ha asignado dicha categoría, junto con las tres métricas que se han utilizado para asignar las clases de relevancia y controversia a las propuestas.

```
1 SELECT category, COUNT(*) AS num_proposals, SUM(numSupports) AS supports,
      SUM(numComments) AS comments, ROUND(AVG(aggregation), 4) AS avgAggregation
2 FROM `proposal_categories` AS pc JOIN `proposals` AS p ON pc.id = p.id
3 GROUP BY category ORDER BY COUNT(*) DESC;
```

| category | num_proposals | supports | comments | avgAggregation |
|---|---|---|---|---|
| Medio ambiente | 7256 | 1220356 | 25848 | 0.0270 |
| Urbanismo | 6203 | 709466 | 15252 | 0.0230 |
| Movilidad | 5526 | 705268 | 15795 | 0.0253 |
| Sostenibilidad | 3293 | 689541 | 14023 | 0.0293 |
| Deportes | 2577 | 348187 | 8135 | 0.0245 |
| Derechos sociales | 2570 | 450933 | 10919 | 0.0291 |
| Economia | 2111 | 436102 | 9758 | 0.0298 |
| Seguridad y emergencias | 1802 | 241107 | 5608 | 0.0253 |
| Equidad e integracion | 1555 | 282040 | 7278 | 0.0305 |
| Transparencia | 998 | 222974 | 4893 | 0.0290 |
| Educacion | 992 | 127459 | 2592 | 0.0252 |
| Asociaciones | 855 | 187855 | 4518 | 0.0306 |
| Animales | 807 | 160704 | 6075 | 0.0367 |
| Familia e infancia | 606 | 81341 | 1739 | 0.0241 |
| Ocio y entretenimiento | 549 | 55719 | 1881 | 0.0271 |
| Ayuntamiento y administracion publica | 540 | 101216 | 2817 | 0.0315 |
| Salud y sanidad | 524 | 84288 | 1374 | 0.0252 |
| Delicuencia | 487 | 108470 | 2374 | 0.0316 |
| Accesibilidad | 417 | 48121 | 1087 | 0.0243 |
| Cultura | 405 | 57040 | 1287 | 0.0263 |
| Justicia | 369 | 65620 | 1897 | 0.0335 |
| Participacion ciudadana | 342 | 30012 | 1436 | 0.0256 |
| Tercera edad | 295 | 41841 | 671 | 0.0239 |
| Empleo | 212 | 33398 | 756 | 0.0301 |
| Vivienda | 205 | 58668 | 1461 | 0.0315 |
| Turismo | 166 | 30186 | 754 | 0.0294 |
| Civismo | 164 | 32798 | 1031 | 0.0337 |
| Politica | 146 | 33407 | 773 | 0.0318 |
| Jovenes | 123 | 12301 | 227 | 0.0211 |
| Religion | 86 | 59715 | 1466 | 0.0470 |

**Figura F.2:** Tabla en la que se muestran los valores de las categorías: propuestas asignadas, suma de apoyos y comentarios, y media de valores *aggregation* para cada categoría.