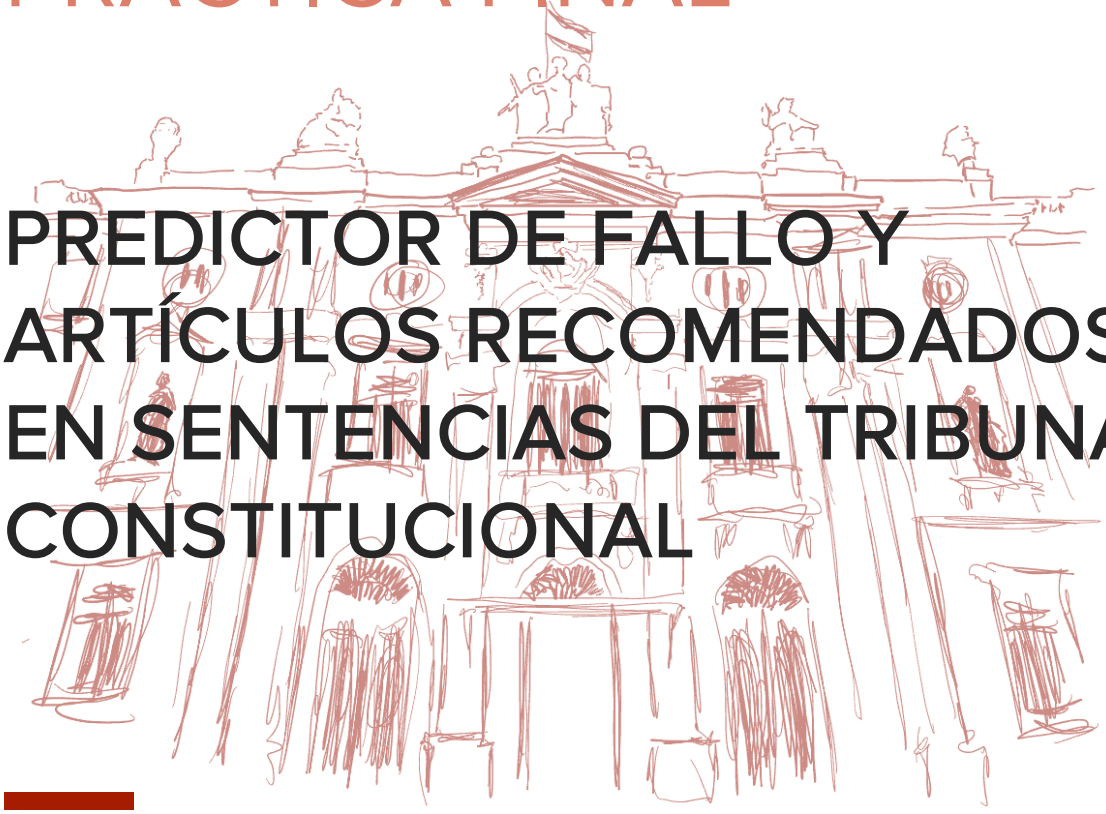
# PRÁCTICA FINAL



# PREDICTOR DE FALLO Y ARTÍCULOS RECOMENDADOS EN SENTENCIAS DEL TRIBUNAL CONSTITUCIONAL

17 DEC, 2023

(G. 06)

| | |
|---|---|
| Raúl Carrero Martín | 100451286 |
| Natalia Jarillo Collado | 100454409 |
| Isaac Buiza González | 100451176 |

Marina Buitrago Pérez 100428967

# Índice

# Índice de acrónimos

| | |
|---|---|
| **art.** | Artículo |
| **CGPJ** | Consejo General del Poder Judicial |
| **CE** | Constitución Española |
| **CC** | Código Civil |
| **EAC** | Estatuto de Autonomía de Cataluña |
| **LBRL** | Ley Reguladora de las Bases del Régimen Local |
| **LEC** | Ley de Enjuiciamiento Civil |
| **LET** | Ley del Estatuto de los Trabajadores |
| **LGSS** | Ley General de la Seguridad Social |
| **LGT** | Ley General Tributaria |
| **LJCA** | Ley reguladora de la Jurisdicción Contencioso-administrativa |
| **LOE** | Ley Orgánica de Educación |
| **LOFCA** | Ley Orgánica de Financiación de las comunidades autónomas |
| **LOREG** | Ley Orgánica del Régimen Electoral General |
| **LOPJ** | Ley Orgánica del Poder Judicial |
| **LOTC** | Ley Orgánica del Tribunal Constitucional |
| **RCD** | Reglamento del Congreso de los Diputados |
| **RPC** | Reglamento del Parlamento de Cataluña |
| **TC** | Tribunal Constitucional |
| **TS** | Tribunal Supremo |

# Definiciones

**Antecedentes de hecho:** descripción detallada y objetiva de los hechos relevantes que han dado lugar a un proceso judicial.

**Fallo:** decisión o resolución emitida por un tribunal en un caso específico.

**Fundamentos de derecho:** razones jurídicas, legales y doctrinales que sustentan la decisión del tribunal.

**Jurisprudencia:** Conjunto de las sentencias de los tribunales, y doctrina que contienen.

**Pleno del Tribunal Constitucional:** órgano colegiado principal de esta institución, integrado por todos los magistrados del tribunal.

**Precepto:** elemento de una ley o norma pudiendo constituir un artículo o parte de éste en donde se establece un mandato, una prohibición, un derecho, un principio o una autorización.

# Introducción y objetivos

## 1. Contexto

Uno de los grandes lastres en la sociedad española del siglo XXI es la lentitud de algunas de sus instituciones, especialmente la Justicia. Esta lentitud está afectando a la percepción de la ciudadanía en lo que respecta al Estado de Derecho. Un Estado de Derecho es, como señala el Consejo Europeo (s.f.), aquel que «[g]arantiza la protección y el respeto de los derechos políticos y civiles básicos, así como las libertades civiles»[1]. Ahora bien, ¿podemos afirmar que en un Estado en donde se está privando a una persona a disfrutar de sus derechos más básicos durante la litispendencia se está garantizando sus derechos? La sencilla respuesta es que no. Así lo afirma el Tribunal Constitucional ante el

[1] Consejo Europeo (s.f.). *Estado de derecho.* Recuperado de: https://www.consilium.europa.eu/es/policies/rule-of-law/#:~:text=El%20Estado%20de%20Derecho%20exige,como%20de%20las%20libertades%20civiles. [fecha de última consulta: 31/10/2023]

el Auto del Juzgado de lo Social de Sevilla nº 780/21 en referencia al derecho fundamental recogido en el art. 24.2 CE – derecho a un proceso sin dilaciones indebidas.

Las duraciones medias de los procesos son preocupantes. Como se indica desde el Consejo General del Poder Judicial (en adelante, CGPJ) en el ámbito civil el tiempo de espera para resolver un asunto alcanza los 38.9 meses (8,7 en primera instancia, 5,6 en apelación y 24,6 ante el TS). En lo que respecta a la jurisdicción Contencioso-Administrativa la media versa en 39,9 meses (12,3 en primera instancia, 14 en apelación y 18,6 ante el TS). Las cifras de la jurisdicción social son muy similares, tan solo siendo mejoradas por el orden penal con una media en 14,3 meses[2]. En lo que respecta al TC, no se mejoran las estadísticas. Este tribunal ha llegado a tardar años en decidir si admitir un recurso y en algunos casos, años en dictar fallo (Hernandez, 2010)[3].

Luego, esta lentitud está degradando la percepción ciudadana del Estado de Derecho y la Justicia, situación crítica puesto que cada vez son más las personas que desconfían de la justicia y optan por actuar al margen de ésta (Cortizo, 2019)[4].

## 2. Descripción del problema

Como se ha detallado previamente, el desafío que enfrentamos es la erosión de la percepción de la justicia por parte de la ciudadanía, consecuencia de las prolongadas

[2] Consejo General del Poder Judicial España (2022). *Estimación de los tiempos medios de duración de los procedimientos judiciales.* Recuperado de: https://www.poderjudicial.es/cgpj/es/Temas/Transparencia/Estimacion-de-los-tiempos-medios-de-duracion-de-los-procedimientos-judiciales/ [Fecha de última consulta: 31/10/2022]

[3] HERNÁNDEZ RAMOS, M. (2010). ¿Admisión discrecional de los recursos de amparo por el Tribunal Constitucional?: balance de cuatro años de aplicación del nuevo trámite de admisión. *Revista de las Cortes Generales*, 273-283.

[4] CORTIZO, G. (31/07/2019). *El CIS constata la desconfianza de los ciudadanos en la Justicia.* El Diario. Recuperado de: https://www.eldiario.es/politica/cis-constata-desconfianza-ciudadanos-justicia_1_1415905.html [Fecha de última consulta: 31/10/2022]

demoras en los procedimientos legales. Si pudiéramos reducir estos plazos y establecer un proceso eficiente que garantice todas las salvaguardias necesarias, podríamos restaurar la confianza en el sistema de justicia como protector de los derechos de los ciudadanos.

A lo largo de esta práctica, trataremos  abordar este problema. Dado el tiempo limitado y el alcance de este trabajo, nos centraremos en resolver las demoras en el Tribunal Constitucional (TC). Sin embargo, el objetivo último, –como se mencionará en la sección final de *Trabajos Futuros*-, es lograr la implementación de estas soluciones en todos los órdenes jurisdiccionales.

Una de las  principales causas de estas demoras reside en que existe un limitado número de jueces y magistrados que tiene que resolver el caso de la manera más justa posible. Este proceso implica una investigación exhaustiva de la jurisprudencia  y doctrina  del tribunal, lo que a su vez conlleva un considerable tiempo. No obstante, los Tribunales cuentan con grandes bases de datos y hemerotecas  que guardan sentencias de numerosas décadas.

Sin embargo, no se está haciendo un uso eficiente de estas fuentes de información, limitándose a realizar búsquedas por palabras clave. Si aprovecháramos eficazmente esta extensa fuente de conocimiento, podríamos proporcionar a los jueces y magistrados un mecanismo mucho más rápido para acceder a los artículos y jurisprudencia aplicable, reduciendo así esta fase que consume tanto tiempo. Efectivamente, este es el propósito central que buscamos lograr en este estudio, como se detallará en la sección subsiguiente.

## 3. Objetivos

Desarrollar un sistema de inteligencia artificial que dote a jueces y magistrados de manera eficaz de una estimación de fallo de sentencia, con el fin de mejorar la eficiencia en la resolución de casos judiciales.

Utilizar técnicas de text mining para analizar exhaustivamente los antecedentes de las sentencias judiciales, facilitando así la recopilación de información relevante.

Implementar un sistema que en función de la fuente de conocimiento generada, tengan la capacidad de identificar y proporcionar los artículos legales aplicables al caso en cuestión.

Integrar de manera coherente todos los componentes del sistema de inteligencia artificial, permitiendo una entrada sencilla de los antecedentes del caso y generando una respuesta automatizada que incluya artículos legales, doctrina aplicable y un fallo estimado.

Validar y perfeccionar el sistema a través de pruebas exhaustivas, asegurando que los resultados sean precisos y confiables para el uso en procesos judiciales.

# Estado del arte

En muchos países se ha llevado a cabo este tipo de proyectos como en Buenos Aires donde una fiscalía desarrolló una IA llamada PROMETEA la cual prepara dictámenes judiciales. Ésta ha supuesto una reducción de 90 minutos a 1 minuto (99%) para la resolución de un pliego de contrataciones, otra de 167 días a 38 días (77%) para procesos de requerimiento a juicio, y de 190 días a 42 días (78%) para amparos habitacionales con citación de terceros, entre otros (Estevez et al, 2020)[5].

En Turquía (Sert, 2022), se creó una IA en la se predecía si una solicitud ante el Tribunal Constitucional de Turquía sobre casos de libertad de expresión y moral resultaban en una "violación" o no. Para este proyecto se han usado múltiples datasets o conjunto de datos que han sido previamente procesados, para la red neuronal se ha usado un perceptrón

[5] ESTEVEZ, E. C., LINARES, S., & FILLOTTRANI, P. (2020). *PROMETEA: Transformando la administración de justicia con herramientas de inteligencia artificial*. Banco Interamericano de Desarrollo. Recuperado de: http://dx.doi.org/10.18235/0002378 [fecha de última consulta: 30/11/2023]

multicapa de redes neuronales artificiales (RNA). El resultado fue la predicción del 90% de precisión[6].

Más concretamente en España se han realizado varios proyectos de estudio sobre ámbitos más concretos en el marco legal. Uno de estos proyectos (Muñoz et. al., 2021) es la predicción de concesión de custodia o conjunta de los padres con sus hijos. En la cual se utilizaron 1884 sentencias judiciales identificando los elementos clave. Para la predicción de dichas sentencias se utilizó una red neuronal, que gracias al uso de los árboles de decisión se encontraron reglas fáciles de aplicar, lo que resultó en una predicción del modelo en más de un 85% de precisión[7].

# Descripción completa de los datos y fuentes

### Obtención de las sentencias

Las Sentencias fueron obtenidas del buscador de jurisprudencia de acceso público del Tribunal Constitucional. Para ello, mediante el uso de web scraping, se obtuvieron todas las sentencias comprendidas entre febrero de 2007 hasta la Sentencia más reciente de 2023, resultando en un total de 3332 sentencias.

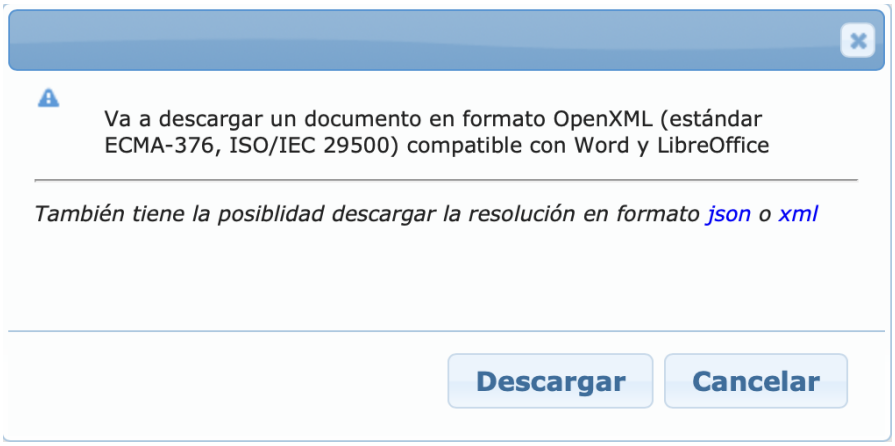
Para la obtención de los archivos de Sentencias se ha desarrollado un script de python mediante el cual se usa la librería *urllib* que permite hacer web scraping. Dado que la página web de origen permite la obtención de archivos conteniendo diferentes tipos de resoluciones se ha hecho un filtro durante la descarga de cada sentencia, en el que el atributo *"*TIPO_RESOLUCION*"* tiene que tener el valor "SENTENCIA" de lo contrario el

[6] SERT, M.F; YILDIRIM, E; HAŞLAK, İ (2022). Using artificial intelligence to predict decisions of the turkish constitutional court. Social Science Computer Review, vol. 40, no 6, p. 1416-1435.
[7] MUÑOZ SORO, J. F., & SERRANO-CINCA, C. (2021). *A model for predicting court decisions on child custody*. *PloS one*, *16*(10), e0258993.

archivo en cuestión será descartado y no se usará como datos para el posterior entrenamiento del modelo[6].

Cabe destacar que el proceso de web scraping ha sido facilitado gracias a que las sentencias del Tribunal están disponibles en formato JSON:

Figura 1: Opciones de descarga de las Sentencias del Tribunal Constitucional




Fuente: Buscador de Jurisprudencia del tribunal Constitucional[8]

### Preprocesamiento inicial de las sentencias

Se requiere evaluar los atributos necesarios de las Sentencias para determinar la información relevante y necesaria a la hora de usar los archivos como datos de entrada al modelo. Una vez analizada la información provista en una sentencia los atributos que estimamos relevantes son:

- Año de resolución
- Síntesis descriptiva: resumen de los antecedentes o hechos acontecidos que deben evaluarse para la resolución de la sentencia
- Síntesis analítica: resumen sobre qué fundamentos jurídicos versa el caso
- Antecedentes: resúmen de los hechos y decisiones de los tribunales previos. Fundamentos legales sobre los que las partes construyen su argumentación legal

[8] Tribunal Constitucional de España. Buscador de Jurisprudencia. Recuperado de: https://hj.tribunalconstitucional.es/HJ/es/Resolucion/Show/29821 [fecha de última consulta 12/12/2023]

- Fundamentos: argumentos doctrinales y jurídicos que la Sala emplea para justificar su decisión
- Fallo_txr: el texto de la decisión completa adoptada por el tribunal
- Art_fundamentos: Este atributo contiene un listado de los artículos mencionados en el atributo Fundamentos.
- Decisión del Tribunal : Contiene una palabra clave “Estima” o “Desestima” –o una expresión análoga–, ésta determina la resolución final del caso.

En el formato original de las sentencias el atributo “Antecedentes” contiene un listado de diccionarios con información como el ID del antecedente, el número del antecedente y el texto asignado a ese antecedente. Por lo que durante el procesamiento de este atributo se han descartado campos como *“ID”* y “*NUMERO”,* mientras que el campo *“TEXTO”* se ha ido concatenando con los siguientes campos de texto de los antecedentes, de tal forma que el resultado final es un campo de texto. Al igual que en el campo “Antecedentes” en el atributo “Fundamentos” y “Fallo” se ha ejecutado el mismo procedimiento ya mencionado anteriormente.

Por otro lado, para la extracción de los artículos mencionados en los fundamentos, en un primer momento se empleó el uso de *expresiones regulares* las cuales registran todas las menciones a un artículo de la Constitución. Estas expresiones regulares son aplicadas al atributo de texto ya procesado de “Fundamentos”, y el formato final será una lista con los números de los artículos utilizados.

Más adelante y tras una revisión de los artículos detectados se decidió ampliar las expresiones regulares de una sola a un grupo de estas. Este cambio se realizó con el fin de aumentar la cantidad de artículos detectados y de esta forma conseguir una mejor cantidad de datos para los futuros modelos.

Por último, se tiene que determinar si la sentencia ha sido estimada o desestimada, sin embargo, durante la evaluación de las sentencias se ha encontrado que a veces no viene explícitamente escrito dichas palabras. Para ello se realizó un estudio tomando una muestra del conjunto de nuestros datos y encontramos frases o conjuntos de palabras que determinan el fallo de dicha sentencia. De este modo logramos clasificar casi todos nuestros datos en estima o desestima:

| Estimación | Desestimación |
|---|---|
| - Estimar<br>- Otorgar el amparo<br>- Otorgar los amparos<br>- Otorgar el recurso de amparo<br>- Otorgar los recursos de amparo<br>- Admitir el recurso de amparo<br>- Admitir los recursos de amparo | - Desestimar<br>- Denegar el amparo<br>- Declarar inadmisible el recurso de amparo<br>- Inadmitir la cuestión de inconstitucionalidad<br>- Denegar los amparos<br>- Inadmitir el amparo<br>- Inadmitir el recurso de amparo<br>- Inadmitir los amparos<br>- Inadmitir los recursos de amparo |

Otro de los problemas encontrados, es la aparición de estas expresiones más de una vez o la aparición de frases que determinan la estimación y desestimación en la emisión de un fallo. Esto complica el establecimiento de la resolución de una sentencia, para ello, se han clasificado los documentos en estimados, desestimados, múltiples y sin estimar. Tras la aplicación de dos conjuntos de *regex* uno conteniendo expresiones para la estimación y el otro expresiones de desestimación, se recupera el número de expresiones encontradas por cada conjunto, en caso de haber encontrado solo una expresión tanto de estimación o de desestimación esta se clasificará con dicha resolución, si no se ha encontrado ninguna expresión esta se clasifica como sin estimar y por tanto se guardará para una revisión manual y por último si se han encontrado múltiples expresiones se descarta el archivo ya que no es posible determinar su estado. Ahora bien, los datos pertenecientes a las categorías no data y multiple data son una población muy pequeña:

| Categoría | Número de sentencias |
|---|---|
| Estima | 1552 |
| Desestima | 1217 |

| | |
|---|---|
| No data | 268 |
| Multiple data | 295 |

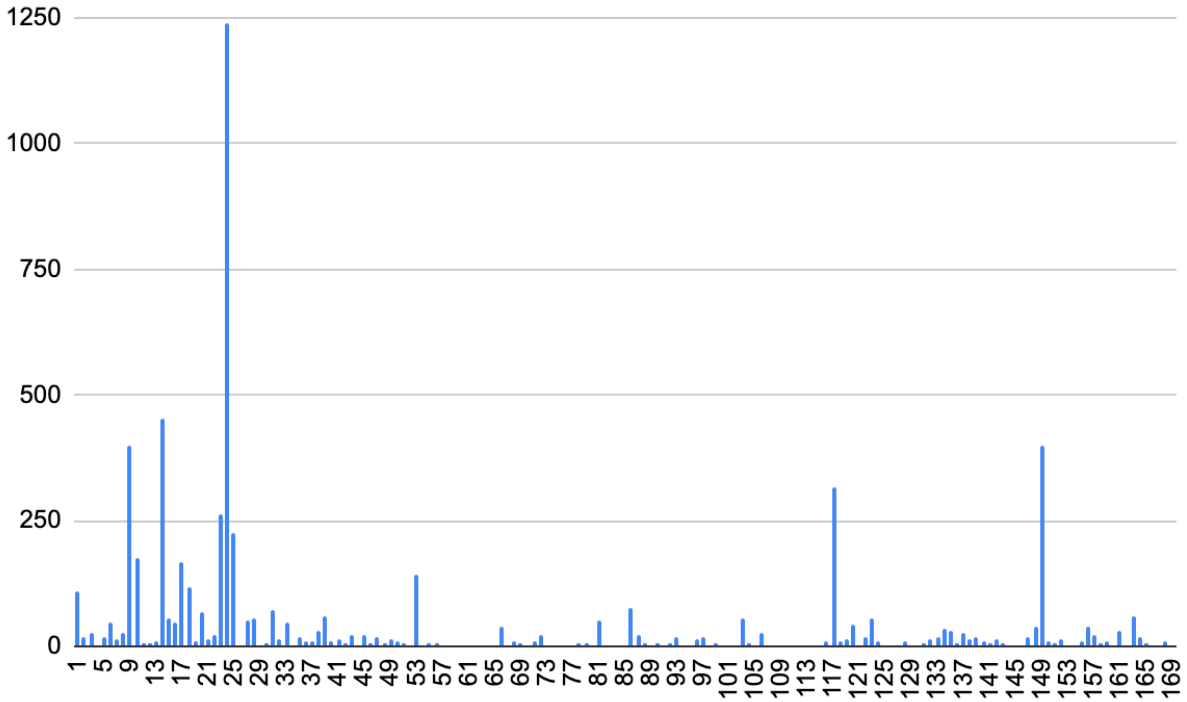
# Descripción de la solución adoptada

### Análisis Exploratorio de Datos

A partir de los datos anteriores, construimos un DataFrame haciendo uso de la librería pandas de Python, y procedimos a realizar el EDA. Para la realización de gráficas empleamos la librería matplotlib.

#### *Artículos más recurrentes utilizados en fundamentos*

Primero, nos interesaba conocer cuáles eran los artículos más recurrentes (fundamento legal sobre el que versa el contenido de la sentencia del TC) en nuestro Dataset por lo que graficamos el número de ocurrencias de cada precepto legal. De este modo, obtuvimos que los casos más recurrentes eran los que versaban sobre la tutela judicial efectiva (art. 24.1 CE), sobre el derecho a la defensa (art. 24.2 CE), sobre la igualdad formal (art. 14 CE) y el principio de legalidad, publicidad y retroactividad de las normas del art. 9.3 CE.



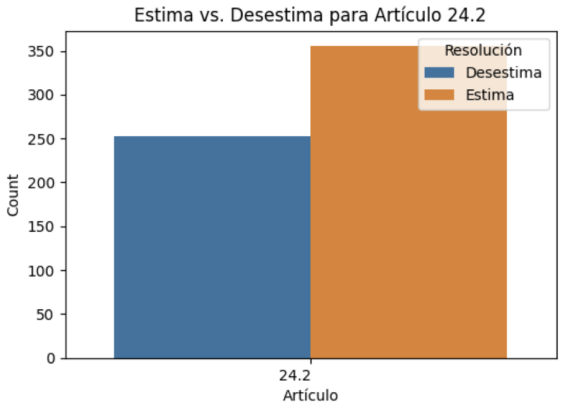

Fuente: elaboración propia

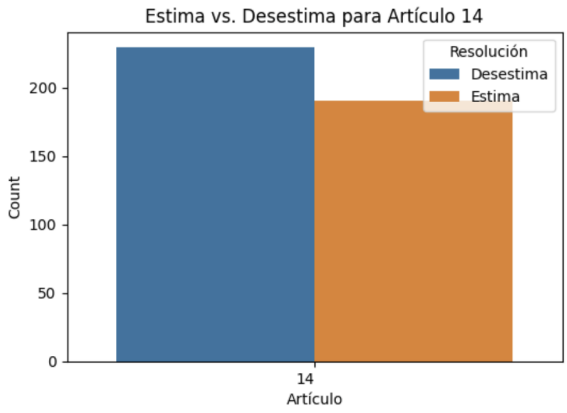
La tabla anterior muestra la mención de cada artículo en los fundamentos de las sentencias. Tal y como se puede observar, la distribución no es equitativa. Esto supone que nos encontremos ante un problema desbalanceado y, por tanto, la dificultad para construir un modelo de clasificación que nos recomiende artículos de fundamentos en base a sentencias aumenta.

### Relación Estimación/Desestimación con artículos

También estudiamos en qué número se estimaban o desestimaban las sentencias en función de cierto artículo concluyendo que para algunos artículos la estimación es mucho más alta que la estimación.

Figura 2: Ejemplo de estimación v. desestimación en función del artículo




Fuente: elaboración propia

### Leyes mayormente citadas

Ahora bien, una sentencia del Tribunal Constitucional no sólo contiene los artículos de la Constitución sino que también se mencionan otras leyes aplicadas al caso. A modo de ejemplo en un supuesto en el que se prohíbe la entrada a una persona a un local que ejercer su derecho de admisión, pueden entrar en juego el art. 24 de la Ley 17/1997, de 4 de julio, *de Espectáculos Públicos y Actividades Recreativas* y el art. 14 CE. Por consiguiente, creamos otra columna en nuestro Data Frame en donde incluímos los artículos mencionados de otras leyes que no fueran de origen constitucional. Para ello empleamos una expresión regular que recogiera el artículo y el acrónimo de la ley o el artículo el precepto y el acrónimo:

```
r'(art\.\s\d+(\.\d+)?\s[A-Z]+|artículo\s\d+(\.\d+)?\s[A-
Z]+|art\s\d+(\.\d+)?\s[A-Z]+|arts\.\s\d+(\.\d+)?\s[A-
```

```
Z]+|arts\s\d+(\.\d+)?\s[A-Z]+|artículos\s\d+(\.\d+)?\s[A-
Z]+|articulos\s\d+(\.\d+)?\s[A-Z]+|articulo\s\d+(\.\d+)?\s[A-Z]+)'
```

A continuación, clasificamos por orden de aparición las leyes mayormente mencionadas, siendo las siguientes:

| Leyes mayormente citadas | | |
|---|---|---|
| **CE** | Constitución española | https://www.boe.es/buscar/act.php?id=BOE-A-1978-31229 |
| **LEC** | Ley de Enjuiciamiento Civil | https://www.boe.es/buscar/act.php?id=BOE-A-2000-323 |
| **LOTC** | Ley Orgánica del Tribunal Constitucional | https://www.boe.es/buscar/act.php?id=BOE-A-1979-23709 |
| **LOPJ** | Ley Orgánica del Poder Judicial | https://www.boe.es/buscar/act.php?id=BOE-A-1985-12666 |
| **EAC** | Estatuto de Autonomía de Cataluña | https://www.boe.es/buscar/act.php?id=BOE-A-2006-13087 |
| **LOREG** | Ley Orgánica del Régimen Electoral General | https://www.boe.es/buscar/doc.php?id=BOE-A-1985-11672 |
| **CEDH** | Carta de los derechos fundamentales de la Unión Europea | https://www.boe.es/doue/2010/083/Z00389-00403.pdf |
| **LJCA** | Ley reguladora de la Jurisdicción Contencioso-administrativa | https://www.boe.es/buscar/act.php?id=BOE-A-1998-16718 |
| **LBRL** | Ley Reguladora de las Bases del Régimen Local | https://www.boe.es/buscar/act.php?id=BOE-A-1985-5392 |
| **LOFCA** | Ley Orgánica de Financiación de las comunidades autónomas | https://www.boe.es/buscar/act.php?id=BOE-A-1980-21166 |
| **LOE** | Ley Orgánica de Educación | https://www.boe.es/buscar/doc.php?id=BOE-A-2006-7899 |
| **LET** | Ley del Estatuto de los Trabajadores | https://www.boe.es/buscar/act.php?id=BOE-A-2015-11430 |
| **LGSS** | Ley General de la Seguridad Social | https://www.boe.es/buscar/act.php?id=BOE-A-2015-11724 |
| **RPC** | Reglamento del Parlamento de Cataluña | https://www.boe.es/buscar/doc.php?id=BOE-A-2006-2453 |
| **LGT** | Ley General Tributaria | https://www.boe.es/buscar/act.php?id=BOE-A-2003-23186 |
| **RCD** | Reglamento del Congreso de los Diputados | https://www.boe.es/buscar/act.php?id=BOE-A-1982-5196 |

| | | |
|---|---|---|
| CC | Código Civil | https://www.boe.es/buscar/act.php?id=BOE-A-1982-5196 |

Dado que estimamos que estas leyes podrían ser de utilidad para proporcionar más información y contexto al modelo realizamos web scraping mediante la librería de python Beautiful Soup almacenando todos los artículos junto con su descripción legal de las siguientes leyes mencionadas en la tabla anterior: CE, LEC, LOTC, LOPJ, EAC, LJCA, LBRL, LOFCA, LET, LGSS, LGT, RCD, CC.

A este conjunto de datos lo denominaremos ***corpus*** a partir de ahora.

### Limpieza y preprocesado de datos

**Representación de artículos CE mencionados**.

Los artículos se mencionaban con un formato no estandarizado: art. , art, articulo, articulos, artículo o artículos. Esto suponía que la mención de un mismo artículo pudiese considerarse como diferente a nivel de token. Por ello en el caso de encontrar un artículo CE con alguna de las estructuras previamente mencionadas, en vez de dividir los tokens como *" art. / art / articulo / articulos / artículo / artículos" "XX.YY" "CE / Constitución Española"* se tomará todo ello como un solo token unificándolo como "artceXX".

Ello se debe a que la palabra art. (o cualquiera de sus otras formas) y el número separado del artículo sin su respectiva ley carece de contenido semántico.

Por ejemplo:

*"art. 162.Y CE"* será reemplazado por *"artce162"* y será considerado en nuestro vocabulario:

features_name.txt

```
8395 artce1
8396 artce10
8397 artce101
8398 artce102
8399 artce103
8400 artce104
8401 artce105
8402 artce106
8403 artce108
8404 artce109
8405 artce11
8406 artce110
8407 artce111
8408 artce113
8409 artce116
8410 artce117
8411 artce118
8412 artce119
8413 artce12
8414 artce120
8415 artce121
8416 artce122
8417 artce123
8418 artce124
8419 artce125
8420 artce127
8421 artce128
8422 artce13
8423 artce130
8424 artce131
8425 artce132
8426 artce133
8427 artce134
8428 artce135
8429 artce136
8430 artce137
8431 artce138
8432 artce139
```

Para el procesamiento de datos hacemos uso principalmente de la librería Natural Language Toolkit (nltk) la cual será necesaria para realizar el procesamiento del lenguaje. De esta librería nos descargamos los conjuntos de datos “stopwords” para eliminación de palabras irrelevantes, “punkt” para la tokenización y “wordnet” para la lematización.

El procesamiento del lenguaje natural se realizará en un pipeline sobre los antecedentes de la sentencia.

Primeramente se realizará la tokenización de cada palabra de los antecedentes convirtiendo el texto en minúsculas, eliminando las stopwords, la puntuación, las urls, las tildes y los saltos de línea.

# 1. Modelo predictor del fallo

| | |
|---|---|
| Tipo de problema | Binary Classification |
| Input | Antecedentes |
| Output Esperado | Estima (1) / Desestima(0).<br>Columna decisión/resolución |

### 1.1 Datos seleccionados

En total contamos con 2769 sentencias, 1552 en las que se estima y 1217 en las que se desestima la resolución.

Para la construcción del modelo emplearemos únicamente la columna antecedentes y la columna decisión o resolución.

Consideramos la granularidad de documento a nivel de antecedente (sentencia).

### 1.2 División en conjunto de entrenamiento y test

Tras realizar la división contamos con 2076 sentencias para entrenar y con 693 para evaluar los resultados.

### 1.3 Representación y modelos realizados

#### 1.3.1 TF-IDF (sklearn)

Para la construcción de la matriz TF-IDF proporcionamos como corpus a `TfidfVectorizer()` el conjunto de antecedentes de sentencias.

`TfidfVectorizer()` construye un vocabulario con todos los tokens (palabras) que aparecen en los antecedentes ya preprocesados.

Posteriomente asocia un índice a cada una de las palabras.

Luego construye una matriz en la que cada fila corresponde a un antecedente (i) y cada columna a una palabra (j) y asocia en la posición [i][j] el número de veces que ese término aparece en el documento por frecuencia inversa del documento. Esta se calcula como el logaritmo de el número de documentos en el conjunto de datos entre los documentos que contienen la palabra.

Esto permite dar una mayor importancia a aquellas palabras que aparecen un mayor número de veces en el antecedente en concreto y menos veces en el resto de documentos.

### Modelos

Una vez construida la matriz tf-idf procedemos a realizar validación cruzada (conjunto de entrenamiento) en la que se pretende obtener la accuracy de los 5 modelos seleccionados para poder compararlos entre sí y seleccionar uno de ellos. Después de seleccionar uno de ellos se hará ajuste de hiperparámetros.

Los modelos a probar fueron los siguientes:

| Modelo |
| --- |
| Naive Bayes |
| Random Forest |
| Linear Support Vector Machine |
| LogisticRegression |

| Red Neuronal |
|---|

### 1.3.2 Ocurrencia de términos (sklearn)

Para la construcción de la matriz TF-IDF proporcionamos como corpus a `CountVectorizer()` de sklearn el conjunto de antecedentes de sentencias.

`CountVectorizer()` construye un vocabulario con todos los tokens (palabras) que aparecen en los antecedentes ya preprocesados.

Posteriomente asocia un índice a cada una de las palabras.

Luego construye una matriz en la que cada fila corresponde a un antecedente (i) y cada columna a una palabra (j) y asocia en la posición [i][j] el número de veces que ese término aparece en el documento.

#### Modelos

Se siguió el mismo método que en TF-IDF de construcción de modelos: validación cruzada con modelos por parámetros por defecto para luego seleccionar un modelo y ajustar sus hiperparámetros.

Los modelos a probar fueron los siguientes:

| Modelo |
|---|
| Naive Bayes |
| Random Forest |
| Linear Support Vector Machine |
| LogisticRegression |

### 1.3.3 Embedding

El propósito de los word embeddings es el de construir una representación de las palabras en la que las palabras con significados parecidos también cuenten con representaciones vectoriales similares. Para ello podemos utilizar word embeddings pre-entrenados o crear nuestro "propio embedding" a partir del conjunto de entrenamiento.

Debido a que contamos con un vocabulario específico del campo legal y que, el texto está en español, se opta por entrenar construir nuestro propio embedding.

La construcción de embeddings y modelos se ha realizado con Tensorflow y Keras.[9]

Para su construcción el primer paso consiste en obtener una representación numérica de cada palabra. Para ello utilizamos `TextVectorization()` de Tensorflow y le proporcionamos el conjunto de entrenamiento de sentencias para que cree el vocabulario a partir de las palabras (tokens) y asocie un entero a cada token.

#### Construcción del embedding durante el entrenamiento

En este caso la construcción del embedding se realiza al mismo tiempo que se entrena el modelo para el objetivo de clasificación de sentencias a partir de las sentencias transformadas por `TextVectorization()`.

Para ello se selecciona el tamaño del vocabulario, que se corresponde con el número de palabras distintas de las sentencias, y el tamaño del espacio vectorial del embedding.

[9] Tensor Flow (2023) *Incrustaciones de palabras. Recuperado de:* https://www.tensorflow.org/text/guide/word_embeddings?hl=es-419

Asimismo, se prueba con diferentes arquitecturas de redes neuronales. Empleamos un modelo `Sequential()` de Tensorflow que nos permite ir añadiendo capas a la red neural de manera secuencial.

### Modelo 1 - Dense Neural Network

Tras probar con diferentes configuraciones de tamaño de vocabulario y embedding nos decantamos por las siguientes:

| | |
|---|---|
| Tamaño de Vocabulario | 20000 |
| Tamaño de Embedding | 40 |

Contamos con una primera capa en la se construye una representación vectorial para las sentencias (embedding).

Luego se emplea una capa de Pooling para reducir la dimensionalidad y por una capa de neuronas conectadas para finalmente obtener la clase en la capa de salida.

Al tratarse de un problema de clasificación binaria la capa de salida únicamente cuenta con una neurona.

```
Model: "sequential_4"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 embedding (Embedding)       (None, None, 40)          800000

 global_average_pooling1d_4  (None, 40)                0
  (GlobalAveragePooling1D)

 dense_8 (Dense)             (None, 16)                656

 dense_9 (Dense)             (None, 1)                 17

=================================================================
Total params: 800673 (3.05 MB)
Trainable params: 800673 (3.05 MB)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
```

### Modelo 2 - CNN Neural Network

En la literatura encontramos casos en los que se ha empleado este tipo de arquitecturas para clasificación de textos[10]. Por ello probamos con un modelo sencillo:

| | |
|---|---|
| Tamaño de Vocabulario | 50000 |
| Tamaño de Embedding | 128 |

En este caso aumentamos el tamaño del embedding y del vocabulario. Además añadimos una capa convolucional con 128 filtros, cada uno de los cuales con un tamaño de 5.

[10] https://arxiv.org/pdf/1809.02479.pdf

Posteriormente reducimos la dimensionalidad con una capa de pooling y lo conectamos a capas de neuronas conectadas.

```
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 embedding_5 (Embedding)     (None, 5000, 128)         5120000

 conv1d_4 (Conv1D)           (None, 4996, 128)         82048

 global_max_pooling1d_4 (Gl  (None, 128)               0
 obalMaxPooling1D)

 dense_20 (Dense)            (None, 10)                1290

 dense_21 (Dense)            (None, 1)                 11

=================================================================
Total params: 5203349 (19.85 MB)
Trainable params: 5203349 (19.85 MB)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
```

### Modelo 3 - Bi-LSTM Neural Network

Por último, optamos por emplear Bi-Directional long short term memory neural networks. Esto permite a la red neuronal “entender mejor el contexto”.

También optamos por un tamaño de vocabulario de 2000 mientras que reducimos la dimensionalidad del embedding a 64.

| | |
|---|---|
| Tamaño de Vocabulario | 20000 |
| Tamaño de Embedding | 64 |

| Layer (type) | Output Shape | Param # |
|---|---|---|
| embedding_2 (Embedding) | (None, None, 64) | 1280000 |
| bidirectional (Bidirectional) | (None, 128) | 66048 |
| dense_14 (Dense) | (None, 64) | 8256 |
| dense_15 (Dense) | (None, 1) | 65 |

Total params: 1354369 (5.17 MB)
Trainable params: 1354369 (5.17 MB)
Non-trainable params: 0 (0.00 Byte)

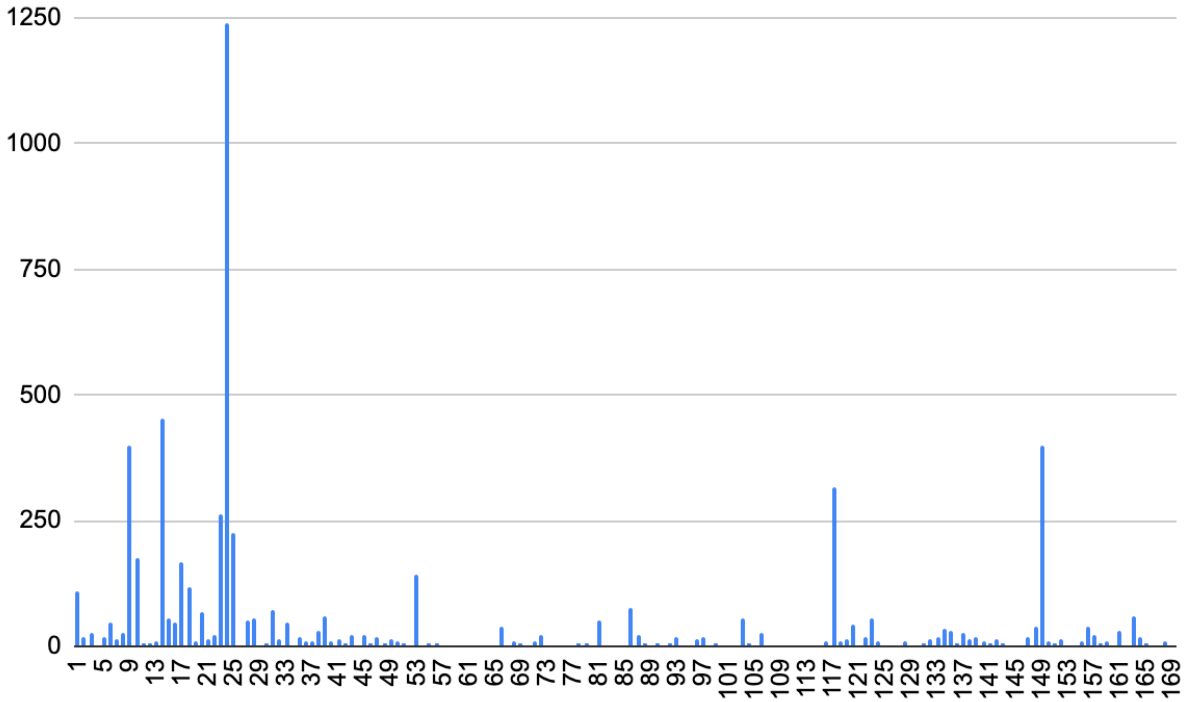
## 2. Modelo recomendador de artículos

Debido al alcance de esta práctica, la sugerencia de artículos realizada por el modelo se limita únicamente a artículos de la Constitución Española.

Cabe mencionar que esta está compuesta por 169 artículos, por lo que pretendemos que el modelo nos proporcione cuáles de estos artículos serían aplicables a los antecedentes.

### 1.1 Datos seleccionados

En  total contamos con 2769 sentencias. La ocurrencia de cada artículo en la columna de artículos fundamentos es la siguiente:




## 1.2 Representación y modelos realizados

### 1.2.1 Embedding - Bi- LSTM

| | |
|---|---|
| Tipo de problema | Multioutput-multilabel Classification |
| Input | Antecedentes |
| Output Esperado | Artículos de la constitución española aplicables.<br><br>Columna artículo fundamentos. |

En primer lugar, procedemos a manipular la columna de artículo de fundamentos. Aplicamos one-hot encoding, añadiendo una columna para cada artículo y un 1 para aquellos artículos que se mencionan en los fundamentos del antecedente de acuerdo a la columna artículos fundamentos.

Posteriormente procedemos a pasar los antecedentes y las columnas de artículos en one-hot encoding al modelo.

El modelo se trata de una red neuronal con la siguiente estructura:

```
model = Sequential()
model.add(Embedding(max_features, 256, input_length=max_sentence_length))
model.add(Bidirectional(LSTM(64)))
model.add(Dense(169, activation='softmax'))
```

Construimos un embedding con un tamaño de vocabulario de 20000 palabras (las 20000 más frecuentes) y un tamaño vectorial de 256. En este caso la última capa (salida) cuenta con 169 ya que se trata de el número de clases con las que contamos "artículos".

### 1.2.2 Embedding - Similitud de embeddigs

En lugar de plantear la sugerencia de artículos como un problema de clasificación, procedemos a plantearlo como "similitud de texto".

Nuestro objetivo es el de recomendar artículos del BOE, por lo que tenemos la hipótesis de que el contenido de los antecedentes podría estar relacionado con los artículos a emplear en los fundamentos.

En consecuencia, si construimos una representación vectorial de cada artículo del BOE y de cada sentencia, podríamos computar la similitud entre cada artículo del

BOE y la sentencia (respectivamente) y encontrar aquellos artículos más próximos (similares).

Entre las técnicas para calcular la similitud encontramos la distancia euclídea, la distancia de Jaccard o la similitud coseno. En nuestro caso incluimos únicamente la distancia euclídea y la de similitud coseno.

Por lo tanto, los pasos a realizar fueron los siguientes:

1. Entrenamos un embedding (word2vec) sobre el conjunto de artículos de distintas leyes, sobre el (conjunto de datos "corpus" mencionando anteriormente).
2. Obtenemos la representación de cada artículo del BOE de acuerdo al embedding entrenado.
3. Obtenemos la representación vectorial de la sentencia.
4. Recomendamos los 5 artículos del BOE cuya representación y la de la sentencia presentan una mayor similitud, mediante la similitud coseno.

# Métodos de evaluación y resultados obtenidos

## 1. Modelo predictor del fallo

Para la evaluación de los modelos escogimos como métrica principal la **accuracy**. Esta es calculada como $\frac{True\ Negatives + True\ Positives}{True\ Positive + False\ Positive + True\ Negative + False\ Negative}$. Esta nos permite ver el porcentaje de predicciones realizadas correctamente.

Además, también tuvimos en cuenta: **precision, recall, F1-Score** y se empleó una matriz de confusión.

### 1.1 TF-IDF

*Comparación Inicial de Modelos*

| Modelo | Accuracy |
|---|---|
| Naive Bayes | 0.712 |
| LogisticRegression | 0.767 |
| Random Forest | 0.768 |
| Linear SVM | 0.784 |
| SDG | 0.776 |
| Red Neuronal | 0.584 |

#### Ajuste de hiperparámetros

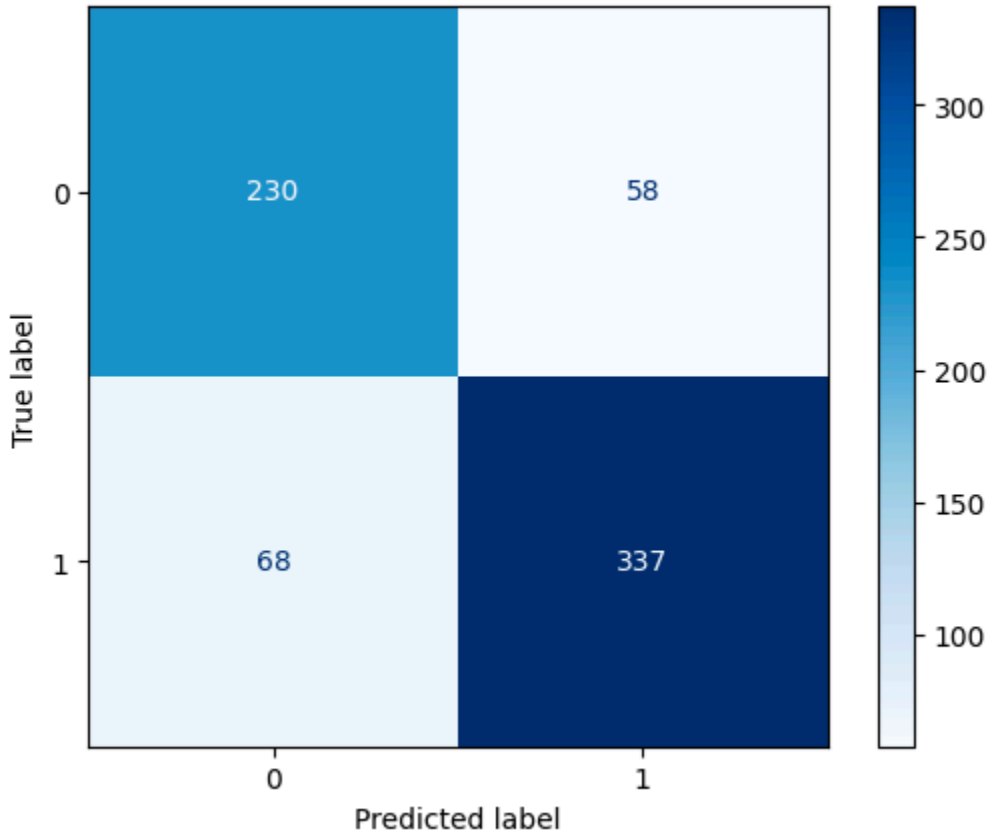
Tras apreciar que Linear SVM es el modelo que mejor valor de accuracy presenta decidimos realizar ajuste de hiperparámetros para el valor C del modelo que representa la regularización y penalty.

Para ello empleamos GridSearch sobre los parámetros anteriormente mencionados.

Tras obtener la configuración del modelo que mejoraba la accuracy se procedió a obtener diferentes métricas así como la matriz de confusión.

| Media | Valor |
|---|---|
| Accuracy | 0.818 |

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| 0 | 0.77 | 0.80 | 0.78 | 288 |
| 1 | 0.85 | 0.83 | 0.84 | 405 |
| accuracy | | | 0.82 | 693 |
| macro avg | 0.81 | 0.82 | 0.81 | 693 |
| weighted avg | 0.82 | 0.82 | 0.82 | 693 |




1.2 Count

*Comparación Inicial de Modelos*

| Modelo | Accuracy |
|---|---|

| | |
|---|---|
| Naive Bayes | 0.720 |
| LogisticRegression | 0.768 |
| Random Forest | 0.704 |
| Linear SVM | 0.762 |
| SDG | 0.753 |

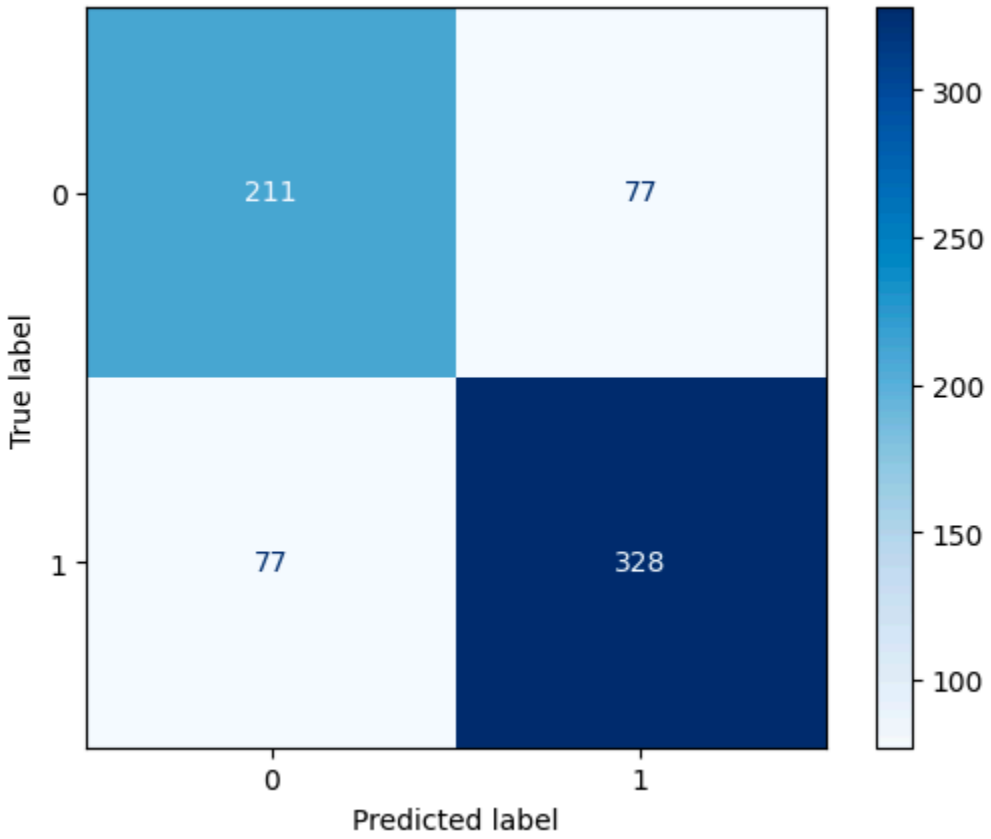
### Ajuste de hiperparámetros

Tras apreciar que LogisticRegression es el modelo que mejor valor de accuracy presenta decidimos realizar ajuste de hiperparámetros para el valor C. Para ello empleamos GridSearch sobre los parámetros anteriormente mencionados.

Tras obtener la configuración del modelo que mejoraba la accuracy se procedió a obtener diferentes métricas así como la matriz de confusión.

| Media | Valor |
|---|---|
| Accuracy | 0.777 |

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| 0 | 0.73 | 0.73 | 0.73 | 288 |
| 1 | 0.81 | 0.81 | 0.81 | 405 |
| accuracy | | | 0.78 | 693 |
| macro avg | 0.77 | 0.77 | 0.77 | 693 |
| weighted avg | 0.78 | 0.78 | 0.78 | 693 |




### 1.3 Embedding

| Modelo | Accuracy |
|---|---|
| Dense | 0.805 |
| Convolutional | 0.778 |
| Bi-LSTM | 0.415 |

## 2. Modelo recomendador de artículos

Para la evaluación de los modelos escogimos como métrica principal la **accuracy**.

Sin embargo, en el caso del "modelo" de similitud de embeddings creemos que es importante tenerla en cuenta pero no debe limitarse a esta. Resulta interesante observar el contenido de la sentencia y la del artículo recomendado.

| Modelo | Accuracy |
|---|---|
| **Clasificación - Embedding +Bi-LSTM** | 0.238 |

Para la recomendación de artículos de acuerdo a la similitud procedimos a obtener la sugerencia de artículos para varias sentencias y analizar los resultados obtenidos. A continuación se muestra un ejemplo:

| Antecedentes |
|---|
| Mediante escrito registrado en este Tribunal el 26 de enero de 2018, el abogado del Estado, en representación del Gobierno, al amparo de los artce161  y 76 y 77 de la Ley Orgánica del Tribunal Constitucional (LOTC), impugna la resolución del Presidente del Parlamento de Cataluña, de 22 de enero de 2018, por la que se propone la investidura de don Carles Puigdemont i Casamajó como candidato a Presidente del Gobierno de la Generalitat de Cataluña, publicada en el "Boletín Oficial del Parlamento de Cataluña", núm. 3, de 23 de enero de 2018, así como la resolución del Presidente del Parlamento de Cataluña, de 25 de enero de 2018, por la que se convoca sesión plenaria el 30 de enero de 2018, a las 15:00 horas, en la parte que se refiere a la inclusión en el orden del día del debate del programa y votación de investidura del diputado don Carles Puigdemont i Casamajó, publicada en el "Boletín Oficial del Parlamento de Cataluña" núm. 5, de 26 de enero de 2018. |

| |
|---|
| El escrito hace expresa invocación del artce161 y del segundo inciso del artículo 77 LOTC, a los efectos de que se acuerde la suspensión de la disposición recurrida. La impugnación se fundamenta en los siguientes motivos:<br><br>a) Se considera que se cumplen los requisitos de jurisdicción y competencia [ artce161 y 2.1 f) LOTC], legitimación (arts artce161 6 LOTC), postulación (art. 82.1 LOTC), plazo (art. 76 LOTC) y forma del escrito (art. 85.1 LOTC). Se afirma también que el acto impugnado es una resolución de una Comunidad Autónoma sin fuerza de ley, por lo que puede ser impugnada a través de este proceso constitucional artce161 . Se indica que el acto impugnado es la convocatoria de la sesión plenaria por el Presidente del Parlamento "en cuanto que incluye el debate y la votación de la investidura del concreto candidato propuesto, Sr. Puigdemont i Casamajó".<br>…<br>(continua) |
| Antecedentes recomendados: 99, 95, 163, 74 |
| Antecedentes de la columna artículos fundamentos: ['99.2', '161.2', '1.1', '152.1', '79.3', '23.2', '23.1', '110', '23', '79'] |

# Análisis de los resultados y conclusiones

## Modelo predictor del fallo

A continuación, encontramos una tabla que resume el rendimiento alcanzado por cada uno de los modelos propuestos anteriormente. Estas métricas se basan en la precisión con la que cada modelo clasifica las sentencias en sus respectivas categorías. Para cada uno de los modelos se han elegido los hiperparámetros para los que estos reflejan el mayor rendimiento.

| Representación | Modelo | Accuracy |
|---|---|---|

| | | |
|---|---|---|
| **TF-IDF** | LinearSVC | 0.818 |
| **Count** | LogisticRegression | 0.777 |
| **Embedding** | CNN | 0.805 |

**Modelo seleccionado:**

Basándonos en la información expuesta en la sección anterior, podemos determinar que el modelo con mejor rendimiento es el SVC lineal. Debido a este rendimiento y a su ligereza en comparación con alternativas como la CNN, decidimos seleccionar este como modelo final.

## Modelo recomendador de artículos

En este caso, a lo largo de toda la experimentación hemos obtenido valores de precisión bastante más bajos. Teniendo en cuenta la información obtenida sobre los datos durante el EDA, eran unos resultados esperables ya que tratamos con un problema multiclase, muy desbalanceado y en el que algunas clases carecían de representación.

**Bi-LSTM**

| Modelo | Accuracy |
|---|---|
| **Bi-LSTM** | 0.238 |

De una forma más analítica podemos observar que el modelo con mejor rendimiento en este caso presenta una precisión de 0.238. Como ya habíamos comentado esto es bastante mejorable y en las siguientes secciones se propondrán diferentes enfoques, tanto relativos al modelo como a los datos, para mejorar este rendimiento.

### Similitud de embeddings

En este caso no calculamos la accuracy, si no que visualizamos las sugerencias que realizaba dado un antecedente de sentencia. Como se mostró en el apartado de resultados, la accuracy no sería muy elevada ya que como se ejemplifica no proporciona todos los artículos mencionados en fundamentos.

Esto es lógico, ya que puede ser posible que existan artículos aplicables en cuanto a contenido pero que, no se emplearon para ese antecedente en concreto.

## Conclusiones

### Modelo predictor del fallo

Consideramos que una accuracy en torno al 80% puede considerarse un resultado satisfactorio. Sin embargo, también creemos que es susceptible de mejorar con métodos de representación y modelos más avanzados.

A pesar de que la precisión todavía es susceptible de mejoras (pudiéndose introducir más datos provenientes de otros tribunales pero de grandes implicaciones constitucionales – como muchas de las decisiones del Tribunal Supremo o algunas decisiones de otros tribunales de rango inferior) con el rendimiento de este modelo se demostrado el desarrollo de un instrumento de gran utilidad para juristas. Mediante el uso de esta herramienta los profesionales en abogacía podrán comprobar qué se predice como resultado en un caso para así orientar su defensa u optar por otras estrategias (como métodos alternativos de resolución de conflictos o negociación) para evitar el litigio. Por su parte, esta herramienta es de gran utilidad para jueces y tribunales en la asistencia de su decisión.

### Modelo sugerencia de artículos

Debido a que nuestro conjunto de datos de sentencias se encuentra desbalanceado en cuanto a los artículos mencionados en los fundamentos, encontramos dificultades para la construcción del modelo.

Una alternativa  era realizar undersampling o oversampling. En el caso de undersampling no nos parecía apropiado debido a que el artículo más mencionado, tenía una ocurrencia de 1241 mientras que otros no se mencionaban en ninguna de las sentencias. Por su parte, nos planteamos realizar oversampling con otras fuentes de origen doctrinal y estimamos que podrían mejorar los resultados notablemente. Estas fuentes doctrinales consisten en una obra compuesta por dos tomos en donde se realizan comentarios sobre todos los artículos de la constitución y se exponen sentencias en referencia a dichos artículos. Dadas las limitaciones de tiempo no procedimos a hacer text mining de esta obra, pero esta información detallada nos ayudaría notablemente en el balanceo de los datos.

# Trabajos futuros

### Longitud media de una sentencia

Debido al tamaño medio de una sentencia 3000 palabras, no ha sido posible utilizar algoritmos pre-entrenados como BERT. Estos permiten añadir una capa adicional al modelo para tareas de clasificación. Sin embargo, el número máximo de tokens de entrada normalmente es bastante inferior al de una sentencia, en el caso de BERT 512.

Tras investigar en literatura académica encontramos una alternativa, la cual consiste en dividir la sentencia en “chunks” o fragmentos para ir introduciéndolas en el modelo como input.[11]

[11] https://dcsi.cs.dal.ca/wp-content/uploads/2022/07/DCSI-2022_paper_1571.pdf

### Preprocesado

Al realizar el EDA apreciamos que las menciones de artículos no se limitaban únicamente a las de la Constitución. Normalizamos la representación de los artículos de la constitución representándolos como “artceXX”.

Sin embargo, esto no se realizó para el resto de artículos no pertenecientes a la CE. Se plantea como futura mejora ya que esto permitiría realizar una distinción a nivel de token entre artículos que tienen la misma numeración pero que pertenecen a leyes diferentes. Por ejemplo, artículo 14 del estatuto de trabajadores sería diferenciado del artículo 14 del código civil.

### Representación

Realizamos web scraping de otras leyes como el estatuto de trabajadores, el código civil, etc. (mencionados anteriormente).

Nuestro objetivo era utilizarlos para construir la representación de palabras pero debido a limitaciones de tiempo y capacidad de cómputo esto no fue posible.

### Longitud media de una sentencia

Debido al tamaño medio de una sentencia 3000 palabras, no ha sido posible utilizar algoritmos pre-entrenados como BERT. Estos permiten añadir una capa adicional al modelo para tareas de clasificación. Sin embargo, el número máximo de tokens de entrada normalmente es bastante inferior al de una sentencia, en el caso de BERT 512.

Tras investigar en literatura académica encontramos una alternativa, la cual consiste en dividir la sentencia en “chunks” o fragmentos para ir introduciéndolas en el modelo como input.

### Parámetros de redes neuronales

Se han realizado pruebas con diferentes arquitecturas de redes neuronales, sin embargo las pruebas en cuanto a parámetros de las mismas han sido limitadas. Por ello, se plantea realizar ajuste de parámetros en trabajos futuros.

### Ámbito

Solo hemos podido abarcar el ámbito del Tribunal Constitucional por ser un órgano con unas sentencias más estandarizadas al versar exclusivamente sobre el contenido de la Constitución y al que solo se recurre por medio del recurso de amparo o de inconstitucionalidad. Si bien es cierto que la reducción de los tiempos en el orden constitucional es clave, debemos de tener en cuenta que el número de casos resueltos por otras jurisdicciones es de una entidad mucho mayor. Según indica la comunicación del Poder Judicial (2023) del total de 6.682.587 asuntos, 2.806.650 pertenecieron al orden civil, 3.216.919 al orden penal, 219.265 al Contencioso-Administrativo y 367.136 a la jurisdicción Social[12]. Esto sugiere que la gran mayoría de casos pertenecen al orden civil y penal y como señalábamos en la introducción, la duración del proceso asciende a entre 22,2 y 25,8 meses y entre 6,4 y 11 meses, respectivamente.

Por consiguiente, en aras de garantizar una verdadera tutela judicial efectiva consagrada en el art. 24 CE, la reducción de dichos tiempos contribuiría significativamente a mejorar la percepción de la justicia en España. Como ya se ha justificado en el primer apartado de este trabajo, el software elaborado es una herramienta fundamental para apoyar en la redacción de sentencias a los jueces y magistrados. Luego, nuestro trabajo futuro se centrará en la escalabilidad de este software a todos los órdenes jurisdiccionales con especial atención al civil y al penal.

[12] Comunicación Poder Judicial (08/03/2023). *Los juzgados y tribunales de toda España registraron durante 2022 un total de 6.682.587 asuntos, un 6,5 por ciento más que el año anterior.* Poder Judicial de España. Recuperado de: https://www.poderjudicial.es/cgpj/es/Poder-Judicial/En-Portada/Los-juzgados-y-tribunales-de-toda-Espana-registraron-durante-2022-un-total-de-6-682-587-asuntos--un-6-5-por-ciento-mas-que-el-ano-anterior- [fecha de última consulta: 29/10/2023]

# Referencias

Comunicación Poder Judicial (08/03/2023). *Los juzgados y tribunales de toda España registraron durante 2022 un total de 6.682.587 asuntos, un 6,5 por ciento más que el año anterior.* Poder Judicial de España. Recuperado de: https://www.poderjudicial.es/cgpj/es/Poder-Judicial/En-Portada/Los-juzgados-y-tribunales-de-toda-Espana-registraron-durante-2022-un-total-de-6-682-587-asuntos--un-6-5-por-ciento-mas-que-el-ano-anterior- [fecha de última consulta: 29/10/2023].

Consejo Europeo (s.f.). *Estado de derecho.* Recuperado de: https://www.consilium.europa.eu/es/policies/rule-of-law/#:~:text=El%20Estado%20de%20Derecho%20exige,como%20de%20las%20libertades%20civiles. [fecha de última consulta: 31/10/2023].

Consejo General del Poder Judicial España (2022). *Estimación de los tiempos medios de duración de los procedimientos judiciales.* Recuperado de: https://www.poderjudicial.es/cgpj/es/Temas/Transparencia/Estimacion-de-los-tiempos-medios-de-duracion-de-los-procedimientos-judiciales/ [Fecha de última consulta: 31/10/2022].

CORTIZO, G. (31/07/2019). *El CIS constata la desconfianza de los ciudadanos en la Justicia.* El Diario. Recuperado de: https://www.eldiario.es/politica/cis-constata-desconfianza-ciudadanos-justicia_1_1415905.html [Fecha de última consulta: 31/10/2022].

ESTEVEZ, E. C., LINARES, S., & FILLOTTRANI, P. (2020). *PROMETEA: Transformando la administración de justicia con herramientas de inteligencia artificial*. Banco Interamericano de Desarrollo. Recuperado de: http://dx.doi.org/10.18235/0002378 [fecha de última consulta: 30/11/2023].

JAYASINGHE, S., RAMBUKKANAGE, L., SILVA, A., DE SILVA, N., PERERA, S., & PERERA, M. (2022, October). Learning Sentence Embeddings In The Legal Domain with Low Resource Settings. In *Proceedings of the 36th Pacific Asia Conference on Language, Information and Computation* (pp. 494-502).

MUÑOZ SORO, J. F., & SERRANO-CINCA, C. (2021). A model for predicting court decisions on child custody. *PloS one*, *16*(10), e0258993.

SERT, M.F; YILDIRIM, E; HAŞLAK, İ (2022). Using artificial intelligence to predict decisions of the turkish constitutional court. Social Science Computer Review, vol. 40, no 6, p. 1416-1435.

Tensor Flow (2023) *Incrustaciones de palabras. Recuperado de:* https://www.tensorflow.org/text/guide/word_embeddings?hl=es-419 [fecha de última consulta: 16/12/2023].

# Anexos

Colab python notebooks para el desarrollo de los modelos:
https://colab.research.google.com/drive/1zQAwC_7YEJgistG82J3QNFPpCJkzWr10?usp=sharing

Datos preprocesados y script de obtención de datos:
https://drive.google.com/drive/folders/1l1NZ96Clx-UFcWC-F9nNaMFRstI98NmK?usp=sharing

Otros recursos empleados:
https://drive.google.com/drive/folders/1AbNRkKFFFN6OecEhHAo66SpPV4aqzcWl?usp=sharing