Ingeniería en Sistemas de Información

Proyecto Final de Carrera

"Desarrollo de prototipo de software basado en aprendizaje profundo para la clasificación de textos legales en las diferentes ramas del derecho"

Año 2022

Alumno

Gerarduzzi, Juan Ignacio - LU 24001 - email jigerarduzzi@gmail.com

Directoras

Gutierrez, Milagros y Ale, Mariel Alejandra

Asesor temático

Pacchiotti, Mauro

Agradecimientos

Quisiera aprovechar estas líneas para dar gracias a todas las personas que me acompañaron a lo largo de todos estos años en la Universidad Tecnológica Nacional de Santa Fe.

En primer lugar quería agradecer a toda mi familia por el apoyo recibido, desde mis padres y hermana hasta mis tíos y abuelos. A mis padres, que siempre me apoyaron y fueron mi ejemplo a seguir en todos los aspectos de mi vida. A mi hermana, que por supuesto es la mejor del mundo, al menos para mi. Mis abuelas, que siempre me brindaron aliento en los momentos difíciles y compañía.

Quiero mostrar mi más sincero agradecimiento a mis amigos y compañeros de clase, que han hecho que este duro proceso como lo es la carrera de ingeniería sea más amena. Me formaron como ingeniero e hicieron que madurara a lo largo de estos años. Tampoco quiero dejar de lado a Florencia que estuvo presente en los momentos buenos y malos.

Quería hacer una mención especial a mi padre, que hoy no se encuentra entre nosotros, pero estoy seguro que se sentirá muy orgulloso de mí y de que, al igual que él, haya logrado el objetivo que nos propusimos cuando decidimos estudiar Ingeniería en Sistemas de Información. Nada de esto hubiese sido posible sin su ejemplo de dedicación y ganas de salir adelante. Esto es por y para él.

Por todo esto quería darles las gracias, Juan Ignacio.

1. Introducción

El aprendizaje automático es una rama de la inteligencia artificial que desde hace varios años se encuentra en auge en el ámbito de las TICs. El mismo consiste en crear sistemas capaces de generar un comportamiento específico o esperado a partir de un conjunto de datos de entrada.

La aplicación de algoritmos de aprendizaje automático en el dominio legal es un fenómeno que puede observarse durante las últimas décadas (Bansal y col., 2019). Con la mayor disponibilidad de texto legal en formato digital, el enfoque en el desarrollo de modelos y aplicaciones inteligentes ha recibido especial atención por parte de la comunidad de investigadores. Se han aplicado una amplia variedad de enfoques, incluidos el resumen, el razonamiento, la clasificación, la traducción, el análisis de texto y otros, a una variedad de problemas de dominio legal. El uso de soporte inteligente basado en computadora tiene muchos beneficios para la comunidad profesional legal. Estos beneficios incluyen la reducción de la laboriosa tarea humana involucrada en la búsqueda y recuperación de material relevante, reduciendo los costos legales a través de la automatización; resolver problemas sin la participación de los tribunales o con menos tiempo y esfuerzo; negociación de la ley para los profesionales del derecho y también para los usuarios comunes; y tomar decisiones basadas en sistemas de predicción que puedan considerarse más precisos.

Cómo se define en Perezzini y col. (2020) los documentos legales, tales como normativas y fallos judiciales, son textos escritos por el ser humano para la propia emisión en distintos ámbitos de un gobierno. La práctica de la Ley necesariamente involucra el análisis y la interpretación del lenguaje natural en el que se presentan dichos tipos de documentos. Con el objeto de dar soporte al análisis de documentos legales, surge la denominada ingeniería legal (IL). Como se detalla en Shimazu y Le Nguyen (2014), la IL aplica nociones de las ciencias de la información, ingeniería de software e inteligencia artificial para asistir a profesionales de la Ley y a individuos en general, en tareas de toma de decisión que involucren a la legislación. Dado que la interpretación del lenguaje natural no escapa del análisis de textos legales, no es sorprendente que muchas herramientas de la inteligencia artificial y del procesamiento del lenguaje natural (NLP), como los clasificadores de texto, sean fundamentales para la mayoría de las aplicaciones existentes de la IL.

Como se mencionó anteriormente, el interés en dicha área ha aumentado considerablemente en el ámbito científico. A modo de ejemplo, en Lippi y col. (2019) se propone un sistema de aprendizaje automático con la intención de detectar cláusulas injustas dentro de largos contratos electrónicos, en Yamada y col. (2019) se han estudiado métodos de resumen de documentos legales, en Neil y col. (2019)

se presenta un enfoque para modelar argumentos legales para la defensa o por la fiscalía durante un juicio y en Cardellino y col. (2018) se mejora el acceso a una base de datos de legislación utilizando detección automática de entidades. Más recientemente, De Araujo y col. (2020) presentan los resultados de aplicar los modelos de bolsa de palabras, redes neuronales convolucionales y algoritmos de boosting a un conjunto de datos denominado VICTOR compuesto por más de 4.6 millones de páginas de documentos digitalizados de la Corte Suprema Brasileña.

En el ámbito local, el boletín oficial de la Nación Argentina (BORA) es el medio de comunicación escrito que el Estado Argentino utiliza para publicar sus normas jurídicas y otros actos de naturaleza pública. Este abarca los tres poderes, legislativo, ejecutivo y judicial, además de los Órganos de Control y el Banco Central. A su vez puede incluir comunicados y avisos, edictos particulares y oficios judiciales. El BORA es una publicación de difusión diaria que todos los ciudadanos pueden consultar. Sin embargo, la gran cantidad de información que contiene el mismo hace difícil que el usuario encuentre rápidamente lo que necesita, lo cual vuelve al proceso ineficiente.

Este Proyecto Final de Carrera aborda el problema de clasificar texto jurídico en forma automática. Para ello, se utilizarán técnicas de minería de texto y sistemas inteligentes entrenados que adquieran la capacidad de clasificar de acuerdo a las ramas del derecho. De esta forma, teniendo la información clasificada, le permite a los diferentes usuarios encontrar la información con mayor rapidez y certeza. Concretamente, este proyecto se enfocará en clasificar en las diferentes ramas del derecho las leyes sancionadas por el Congreso de la Nación, dejando de lado las demás normativas.

El objetivo principal al desarrollar este proyecto es investigar sobre la clasificación mediante etiquetas de textos. Para un ser humano puede resultar fácil leer un texto y definir una serie de palabras que definan su contenido, sin embargo existe todo un campo de estudio para que dicha generación se realice de forma automática mediante aprendizaje.

Si bien este proyecto se encuentra orientado concretamente al ámbito del Derecho, se cree que puede servir como punto de partida de cualquier técnica que se base en este tipo de clasificación, independientemente del ámbito.

Aunque existen numerosas herramientas de clasificación propuestas cuyo nivel de exactitud es elevado, todavía hay muchos ajustes a realizar en las mismas. Una de las principales limitaciones que presentan estas herramientas es que las mismas fueron diseñadas para ser aplicadas al idioma Inglés, por lo cual no funcionan de manera eficiente y efectiva en el idioma Español. Uno de los

principales enfoques a analizar en este proyecto es el estudio de posibles mejoras en las herramientas de clasificación existentes, permitiendo optimizar y adaptar las mismas para que puedan ser aplicables al idioma español. A su vez, es necesaria una investigación acerca de cómo la cantidad de datos disponibles afecta el resultado.

1.1 Objetivos

1.1.1 Objetivos generales

Desarrollar un prototipo basado en técnicas de procesamiento de lenguaje natural para clasificar las leyes presentes en el Boletín Oficial de la Nación Argentina (BORA) en las diferentes ramas del derecho.

1.1.2 Objetivos específicos

- Analizar los algoritmos de Deep Learning que se utilizan con mayor frecuencia en la clasificación de textos para seleccionar aquellos más adecuados para la tarea a realizar.
- Definir criterios de evaluación de los modelos de clasificación.
- Obtener, preprocesar y etiquetar el conjunto de datos de entrenamiento para los modelos.
- Entrenar y recopilar resultados de los distintos algoritmos para su evaluación.
- Desarrollar una interfase prototipo para el uso del modelo obtenido.

1.2 Temporalidad

El presente proyecto final de carrera se desarrolló entre los meses de noviembre del año 2021 y septiembre del año 2022.

1.3 Abreviaturas

Conforme se avance con la lectura del presente informe el lector se encontrará con diversas abreviaturas, a saber:

- ML: Machine Learning (aprendizaje automático).
- IA: Inteligencia Artificial.
- PFC: Proyecto Final de Carrera.
- NLP: Procesamiento de Lenguaje Natural.
- BORA: Boletín Oficial de la República Argentina.
- API: Application Programming Interface (interfaz de programación de aplicaciones).

2. Marco teórico

En esta sección se desarrolla el marco teórico que sienta las bases sobre las cuales se llevó a cabo la investigación y el posterior desarrollo del prototipo ingenieril.

2.1 Inteligencia artificial

La Inteligencia Artificial (IA) es la rama de la Ciencias de la Computación que trata la creación de máquinas inteligentes, cualquiera sea el concepto de inteligencia (Neil Gershenfeld – MIT).

La IA, combina algoritmos con el propósito de crear máquinas que presenten las mismas capacidades (o al menos similares) que el ser humano. Permite que los sistemas tecnológicos perciban su entorno, se relacionen con él, resuelvan problemas y actúen con un fin específico. Para esto, la máquina recibe datos (ya preparados o recopilados a través de sus propios sensores, por ejemplo, una cámara), los procesa y responde a ellos.

La IA puede ser dividida en dos importantes ejes:

En el **eje de la racionalidad** evalúa qué premisas deben guiar a una herramienta computacional para ser considerada inteligente, su búsqueda por la decisión más racional, o su cercanía a la inteligencia humana, de tinte menos objetivo:

- Sistemas que actúan racionalmente: "*Es el estudio del diseño de los agentes inteligentes*" (Poole, 1998) [2].

- Sistemas que piensan racionalmente: "*El estudio de las computadoras que permiten percibir, razonar y actuar*" (Winston, 1992) [2] .

En el **eje humano** se evalúa qué tan internalizadas están esas premisas en la estructura de una herramienta considerada inteligente. En otras palabras, ¿debe ser racional o humana, o simplemente debe parecer racional o humana. Es decir, podemos definir a la IA como:

- Sistemas que piensan como humanos: "*La automatización de actividades que se asocian con el pensamiento humano, actividades como la toma de decisiones, resolver problemas, aprender…*" (Bellman, 1978) [2].

- Sistemas que piensan como humanos: *"La automatización de actividades que se asocian con el pensamiento humano, actividades como la toma de decisiones, resolver problemas, aprender…*" (Bellman, 1978) [2].

En resumen a lo presentado anteriormente, el enfoque humano no busca ser una ciencia empírica, mientras que por otro lado, el eje de la racionalidad implica una combinación de ingeniería y matemática.

2.2 Métodos de aprendizaje

En aprendizaje automático el objetivo principal es desarrollar técnicas de Inteligencia Artificial que permitan a ordenadores aprender de sí mismos y de datos suministrados. Para llegar a este fin, se crean agentes inteligentes que puedan generalizar ciertas respuestas a partir de información provista que es suministrada a modo de ejemplos. Es decir, un agente inteligente aprende a través de datos que percibe de su entorno de modo de mejorar la performance en la tarea a la cual está asignado.

El entrenamiento al cual se somete el agente puede tener diferentes características, a continuación se presentan las más populares.

- **Aprendizaje inductivo:** permite resolver un problema mediante el empleo de problemas resueltos en el pasado similares al planteado (*Utilización Del Aprendizaje Inductivo En La Toma De Decisiones. Aplicación En Un Problema De Secuenciación*, n.d.).

- **Aprendizaje deductivo:** El uso del método deductivo es la obtención de un conocimiento nuevo a partir de los conocimientos generales que se dan por válidos. A partir de una generalidad "deduce" un hecho particular o una generalidad más restrictiva que la original. (*CVC. Diccionario De Términos Clave De ELE. Aprendizaje Deductivo.*, n.d.)
  El agente debe a su vez probar las reglas que propone, independientemente del método que utilizó para obtenerlas. Esto se logra a través de la retroalimentación que el agente percibe de su entorno. Dicha retroalimentación podrá ser:

  ➔ **Supervisada:** el aprendizaje supervisado es una técnica para deducir una función a partir de datos de entrenamiento. Es decir, el objetivo es que los ordenadores aprendan de forma automática mediante ejemplos previos.
  El agente es entrenado a partir de un conjunto de pares de entrada/salida ya existentes y ponderados. Al momento de tomar una decisión, el agente lo hace teniendo en cuenta una entrada, y

evaluando el resultado que obtuvo respecto a una salida esperada (o conocida). En otras palabras, para lograr entrenar un algoritmo de aprendizaje supervisado, los datos de entrenamiento consistirán en un par formado por el dato de entrada y su salida esperada.
Durante el entrenamiento, el algoritmo busca patrones en los datos que se correlacionen con los resultados deseados. Una vez finalizado el entrenamiento, el algoritmo de aprendizaje supervisado toma nuevas entradas, y determina los resultados en que se clasificaron las nuevas entradas según los datos de entrenamiento anteriores.
El objetivo de un modelo de aprendizaje supervisado es lograr predecir un resultado (o etiqueta) correcto para los nuevos datos de entrada que ingresan al sistema.

- ➔ **No supervisada:** El algoritmo de aprendizaje está provisto de datos no etiquetados, y es el propio agente el encargado de identificar patrones de entrada con el propósito de agruparlos.

- ➔ **Semi-supervisado:** es una clase de técnica de aprendizaje automático que utiliza datos de entrenamiento tanto etiquetados como no etiquetados. Normalmente una pequeña cantidad de datos etiquetados junto a una gran cantidad de datos no etiquetados.

- ➔ **Por refuerzo**: El agente está en contacto con un entorno dinámico que le proporciona comentarios en términos de recompensas y castigos.

El agente desarrollado en el presente PFC se trata de un prototipo de inteligencia artificial con aprendizaje **deductivo** y **supervisado**.

2.3 Procesamiento de Lenguaje Natural (NLP)

Es una forma de IA que estudia las interacciones entre las computadoras y el lenguaje humano (Alexander Gelbukh, 2010). Se ocupa de la formulación e investigación de mecanismos para la comunicación entre personas y máquinas por medio del lenguaje natural, es decir, de las lenguas del mundo. El NLP da a las máquinas la capacidad no solo de leer, sino de entender e interpretar el lenguaje humano. Es decir, las máquinas pueden dar sentido al texto escrito o hablado y realizar tareas como el reconocimiento del habla, el análisis de sentimientos y el resumen automático de textos.

A continuación, se presentan algunos de los componentes del procesamiento del lenguaje natural. No todos los análisis que se describen se aplican en cualquier tarea de NLP, sino que depende del objetivo de la aplicación.

- **Análisis morfológico o léxico:** consiste en el análisis interno de las palabras que forman oraciones para extraer lemas, rasgos flexivos, unidades léxicas compuestas. Es esencial para la información básica: categoría sintáctica y significado léxico.

- **Análisis semántico:** proporciona la interpretación de las oraciones, una vez eliminadas las ambigüedades morfosintácticas.

- **Análisis sintáctico:** consiste en el análisis de la estructura de las oraciones de acuerdo con el modelo gramatical empleado (lógico o estadístico).

- **Análisis pragmático:** incorpora el análisis del contexto de uso a la interpretación final. Aquí se incluye el tratamiento del lenguaje figurado (metáfora e ironía) como el conocimiento del mundo específico necesario para entender un texto especializado.

Uno de los principales desafíos en esta área son los diferentes grados de complejidad en un lenguaje o frase del mismo. Por dar un ejemplo, los datos generados en una conversación cotidiana son **datos no estructurados**, difíciles de manipular y clasificar.

Inevitablemente los lenguajes son extremadamente complejos y generan ambigüedad, ya sea por el uso de sinónimos, sarcasmos o frases que pueden ser interpretadas de diferentes maneras dependiendo del contexto en el que es usada.
A modo de contrarrestar estas complejidades se han desarrollado algoritmos para el NLP, los cuales se pueden resumir en:

- Text classification (Clasificación de textos).
- Eliminación de stop words
- Modelado de temas.
- Bolsa de palabras
- Tokenización
- Named Entity Recognition (NER).
- Stemming
- Lematización

En la sección “5.1.6 Preprocesamiento de datos” el lector podrá encontrar un desarrollo detallado de los algoritmos utilizados en este PFC.

A continuación se presenta el algoritmo de clasificación de textos, cuyas características son de interés para el presente PFC.

2.4 Clasificación de textos

La clasificación de texto es el proceso de clasificar el texto en un grupo de palabras. Mediante el uso de NLP, la clasificación de texto puede analizar automáticamente el texto y luego asignar un conjunto de etiquetas o categorías predefinidas en función de su contexto (Mukesh Zaver, 2011).

2.4.1 Clasificación de texto para el conjunto de datos de NLP

El NLP de clasificación de texto ayuda a las máquinas a clasificar las palabras claves importantes y percibirlas a través del procesamiento del lenguaje natural. En NLP, las palabras y oraciones cruciales deben clasificarse o etiquetarse con metadatos agregados para que el algoritmo de aprendizaje automático pueda más tarde, replicar o al menos dar una aproximación al comportamiento de los seres humanos.

En pocas palabras, en la clasificación de textos mediante Aprendizaje Automático se realizan clasificaciones en base a observaciones pasadas. Para lograr este objetivo, es necesario entrenar el modelo con ejemplos pre-etiquetados de entrenamiento.

2.5 Optimización de hiperparámetros

Antes que nada, se definirá qué son los hiperparámetros del algoritmo. Básicamente, éstos definen el modo de funcionamiento de la etapa de aprendizaje con el objetivo de modificar el modo de ejecución del algoritmo de entrenamiento. Por ejemplo, la tasa de aprendizaje o el tamaño de lote.

La optimización o ajuste de hiperparámetros es el problema de elegir un conjunto de estos, que sean óptimos para un algoritmo de aprendizaje determinado con el objetivo de generar el modelo que genere los mejores resultados posibles. Para lograr este objetivo, la optimización se realiza normalmente mediante el uso de un proceso de búsqueda cuyo objetivo consiste en encontrar la mejor selección de valores para un conjunto finito de hiperparámetros.

En este PFC, se utiliza una herramienta provista por *Sklean* (4.2.9 Sklearn) llamada *GridSearchCV*, la cual provee los medios para encontrar la combinación óptima de hiperparámetros. *GridSearchCV* permite evaluar y seleccionar de forma sistemática los parámetros de un modelo. Para esto se indica el modelo y los parámetros a probar, de esta manera es posible evaluar el rendimiento del modelo en función de los parámetros mediante validación cruzada.

3. Metodología

En este PFC se utilizó la metodología SEMMA (Sample, Explore, Modify, Model, Asset) la cual es aplicable a proyectos de investigación (School of Computing, Asia Pacific University of Innovation and Technology, 2019). Se optó por realizar ciertas modificaciones a dicha metodología, diseñando un modelo más lineal y agregando etapas y subetapas según se consideró conveniente para la ejecución del proyecto en cuestión.

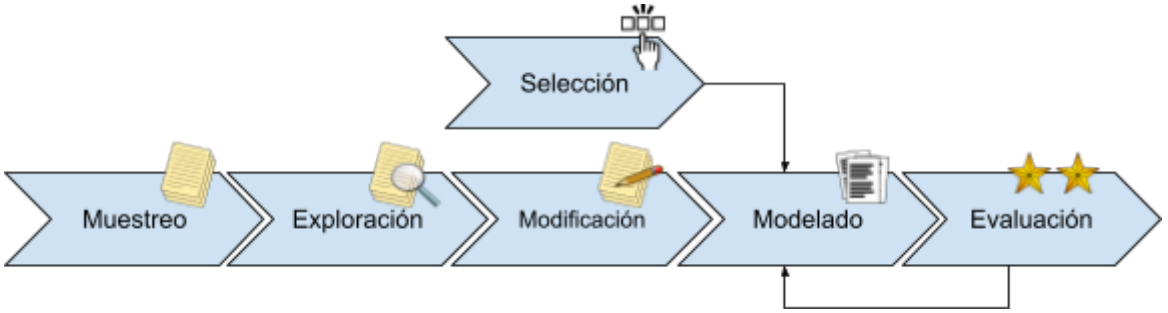

Figura 1 - Metodología utilizada

3.1 Sample (Muestreo)

En esta actividad lo que se debe lograr es la obtención de una muestra de datos que sea representativa de las características comunes que tiene la población a estudiar. Esto sirve para no tener que generar una muestra desde cero, lo cuál traería aparejado una mayor cantidad de tiempo de trabajo, aumentando de esta forma el coste de la realización del proyecto.

Es probable que no se tengan todos los datos deseados al alcance, o que no resulte fácil encontrar una muestra. Con el objetivo de suplir esto es necesario invertir esfuerzo en recabar una cantidad suficiente de leyes sancionadas y publicadas en el BORA para un posterior etiquetado de documentos.

3.2 Explore (Exploración)

Una vez obtenida (o creada) una muestra de datos, es necesario el análisis y la comprensión de la misma. Con el objetivo de lograr esto se procede al estudio de las relaciones que existen entre las diferentes variables para de esta forma poder detectar, y posteriormente eliminar, anomalías o deficiencias.

3.3 Modify (Modificación)

Esta etapa consiste en una selección y transformación de los datos de la muestra para, de esta forma, conseguir que los mismos se ajusten al enfoque de modelado que se posee.

A su vez si queremos usar librerías de aprendizaje automático que ya están disponibles, es necesario etiquetar y además darle un formato correcto a los datos que se poseen. Esto significa que, una vez obtenidos los datos de muestra, es necesario etiquetar los mismos para un posterior entrenamiento del algoritmo. En esta etapa se debe contar con un conjunto de datos y observaciones para los que ya se tiene la respuesta de destino. Las muestras para las que ya se conoce la respuesta de destino son los denominados datos etiquetados. A priori, se pueden identificar los siguientes pasos a seguir:

a. **Filtro de datos relevantes:** se eliminan todas las partes del BORA que no cumplan un propósito para el estudio.

b. **Dar formato adecuado a los datos**: si se quiere usar librerías de aprendizaje automático que ya están disponibles, primero es necesario darle formato a los datos.

c. **Calcular características relevantes:** es decir, la respuesta que desea predecir. Los algoritmos de aprendizaje automático funcionan mucho mejor si se les ofrecen características relevantes en vez de los datos puros. La fase de calcular características relevantes requiere un esfuerzo mayor.

d. **Etiquetado**: etiquetado de los textos del conjunto de datos.

3.4 Selection (Selección)

Esta actividad consiste en seleccionar técnicas y herramientas que sean capaces de realizar el procesamiento automático de textos mediante algoritmos de aprendizaje automático. Dicha actividad es independiente a las anteriores y sirve para definir el modelado a implementar. Se puede dividir en dos etapas importantes:

e. **Relevamiento:** relevamiento de información sobre algoritmos, técnicas y herramientas a utilizar.

f. **Análisis**: análisis, evaluación y selección de las técnicas y herramientas a utilizar en el proyecto.

3.5 Model (Modelado)

Una vez que se optimizan las muestras y se seleccionan las técnicas y herramientas a utilizar en el enfoque propuesto se debe proceder con la construcción del modelo. Resulta conveniente dividir esta actividad en las siguientes etapas:

g. **Diseño:** diseño de la arquitectura del modelo.

h. **Determinar tipos de entrenamientos:** selección de estrategias de entrenamiento a utilizar y configuración de parámetros de los modelos neuronales.

i. **División del conjunto de datos:** seleccionar cuáles datos se destinarán a entrenamiento y cuales a pruebas y validaciones.

j. **Implementación:** implementación de los modelos de redes neuronales.

k. **Entrenamiento:** entrenamiento de los modelos con el dataset definido y selección de modelo más adecuado.

3.6 Asses (Evaluación)

El proceso de evaluación resulta importante para entender qué tan bien funciona el modelo clasificador propuesto y si se debe mejorar el mismo.

l. **Evaluación**: se deben evaluar los resultados para determinar qué resulta mejor, si se debe utilizar un modelo más completo o más simple, concluir si es necesario más datos o que estos sean de mejor calidad, desarrollar una mejor comprensión del problema y entender mejor cuales son los siguientes pasos a realizar. Esta actividad se puede dividir en las siguientes etapas:

   i. Definición de métricas para evaluación.

   ii. Definición de casos de prueba.

   iii. Evaluación de la efectividad del modelo neuronal.

   iv. Formulación de conclusiones con respecto al modelo evaluado.

Además, con el objetivo de obtener una mejor visión de las posibles soluciones y sus alternativas, se adapta la actividad de Asses (Evaluación) y se añade una sub-etapa en la cual se construye un modelo prototipo.

m. **Prototipado**: desarrollar una herramienta de software prototipo que utilice el modelo entrenado para la clasificación de textos legales. La tarea se divide en las siguientes etapas:

   i. Análisis de requerimientos y restricciones.

   ii. Diseño de la herramienta.

   iii. Implementación de la herramienta.

4. Tecnologías Utilizadas

En esta sección se presentan las diferentes tecnologías utilizadas en el desarrollo del PFC. En primer lugar, se detalla el lenguaje de programación empleado. Luego, se hace un análisis de las diferentes herramientas esenciales para el desarrollo, haciendo una justificación de la elección de cada una de ellas. Más tarde, se listan las librerías y herramientas empleadas. Por último, se listan los entornos de desarrollo en los cuales se escribió el código fuente en cuestión.

4.1 Lenguajes y Frameworks

4.1.1 Python

Python es un lenguaje sencillo de leer y escribir debido a su alta similitud con el lenguaje humano. Además, se trata de un lenguaje multiplataforma de código abierto y, por lo tanto, gratuito, lo que permite desarrollar software sin límites. Cabe destacar también que es un lenguaje multiparadigma, ya que soporta la programación orientada a objetos, programación imperativa y también funcional.

Python es uno de los lenguajes de programación más utilizados para proyectos relacionados con el aprendizaje automático. Esto se debe, entre otras cosas, a que es posible encontrar una gran cantidad de librerías y frameworks que resultan de utilidad para proyectos vinculados al aprendizaje automático.

La solución está implementada en su totalidad en este lenguaje debido a la gran versatilidad, compatibilidad, rapidez y popularidad del mismo, particularmente en áreas de investigación y tratamiento de grandes volúmenes de datos.

Cabe destacar también que Python cuenta con una documentación detallada que facilita su aprendizaje y foros para realizar consultas. Todo esto sumado a la sencillez a la hora de leer códigos escritos en Python, lo convierten en la mejor opción a aplicar en el PFC.

4.2 Bibliotecas y herramientas de soporte

4.2.1 AppJar

AppJar es una biblioteca Python de código abierto multiplataforma para el desarrollo de GUIs. Puede ejecutarse en Linux, OS X y Windows. Brinda una gran cantidad de opciones de personalización que permiten crear interfaces sencillas en unos pocos minutos que se adaptan a casi cualquier desarrollo de software.

4.2.2 Git

Git es un software de control de versiones, pensando en la eficiencia, la confiabilidad y compatibilidad del mantenimiento de versiones de aplicaciones cuando estas tienen un gran número de archivos de código fuente.

Para utilizar Git primero se crea un repositorio remoto central donde cada desarrollador puede crear una copia local en un entorno local para trabajar en los cambios asignados. Luego, los cambios son subidos por cada desarrollador y combinados en un producto final.

Se decidió hacer uso de los servicios de Git ya que los mismos habían sido utilizados previamente por el autor del proyecto, ya sea en el ámbito facultativo o laboral.

4.2.3 Google Drive

Google Drive permite almacenar, crear, modificar, compartir y acceder a documentos, archivos y carpetas de todo tipo en un único lugar (nube). Cuenta con un almacenamiento gratuito, también ofrece la posibilidad de revisiones.

La utilización de Google Drive fue imprescindible ya que permitió trabajar de forma simultánea y realizar cambios en tiempo real en los diferentes informes del PFC, los cuales podían ser observador por las directoras.

4.2.4 Google Meet

Es un servicio libre de videollamada desarrollado por Google que funciona en cualquier dispositivo. Sus servicios

fueron muy importantes para llevar adelante las reuniones periódicas de revisión y avance.

4.2.5 Draw.io

Draw.io es una herramienta de diagramación que permite generar diagramas de flujo, de proceso, entre otras muchas funciones. Es una herramienta gratuita con la que se pueden dibujar mapas conceptuales, esquemas o diferentes representaciones gráficas, como diagrama de jerarquía o conjuntos.

4.2.6 Spacy

Spacy es una librería de Python que permite construir aplicaciones de procesamiento de lenguaje natural (PNL). Además, permite crear modelos nuevos o reentrenar los modelos que proporciona con datos propios para entrenar los modelos en campos específicos. Los modelos de spaCy de procesamiento de lenguaje natural permiten analizar un texto y extraer información tanto del texto como de las predicciones del modelo sobre su significado por el contexto. spaCy es una biblioteca que está diseñada para ayudar a la creación de aplicaciones

4.2.7 Google Collaboratory

Permite a cualquier usuario escribir y ejecutar código arbitrario de Python en el navegador. Es especialmente adecuado para tareas de aprendizaje automático y análisis de datos. Se basa en el concepto de Notebook y se almacena en Google Drive. Estas Notebooks se componen principalmente de dos tipos de celdas, una celda que puede contener texto, imágenes, etc. y un segundo tipo de celda la cual contiene el código a ejecutar.

Como es ejecutada en servidores de Google, se requiere de conexión a internet para tener acceso a dicha herramienta. Cabe destacar que en este entorno se pueden realizar cargas de datos, no solo desde una fuente local sino también desde la nube. Esto permite garantizar la consistencia de los recursos durante el desarrollo ya que cualquier actualización que se realice se verá reflejada en la plataforma.

En el contexto del proyecto, la principal ventaja que ofrece esta herramienta es que libera a nuestra máquina de tener que llevar a cabo un trabajo demasiado

costoso tanto en tiempo como en potencia. Para probar técnicas o métodos, primero se desarrolló el código en esta plataforma a modo de prueba y luego se realizaron corridas y se evaluaron resultados. Por último, una vez que se tuvo una versión que cumpla con las características deseadas, se ultimaron los detalles del código de manera local.

4.2.8 Tika Parser

Tika Parser es una interfaz que brinda la posibilidad de extraer contenido y metadatos de cualquier tipo de documento. Tika se vincula con Python mediante servicios REST, lo que permite llamar a tika de forma nativa en el lenguaje python. Para este proyecto, fue sumamente necesaria la utilización de esta interfaz ya que permite en unos pocos segundos extraer y manipular el texto de los archivos en formato PDF que contienen las diferentes leyes

4.2.9 Sklearn

Sklearn (o Scikit-learn) es una biblioteca útil y sólida para el aprendizaje automático en Python. Proporciona una selección de herramientas eficientes para el aprendizaje automático y el modelado estadístico, incluida la clasificación, la regresión, el agrupamiento y la reducción de la dimensionalidad a través de una interfaz en Python.

4.2.10 Pandas

Pandas es un paquete de Python de código abierto que se usa más ampliamente para tareas de análisis de datos y aprendizaje automático. En particular, ofrece estructuras de datos y operaciones para manipular tablas numéricas y series temporales. Para este PFC fue de suma importancia al momento de almacenar y manipular los datasets.

4.2.11 Matplotlib

Matplotlib es una biblioteca para la generación de gráficos a partir de datos contenidos en listas o arrays en el lenguaje de programación Python

4.2.12 NLTK

NLTK es un conjunto de bibliotecas y programas para el procesamiento del lenguaje natural (NLP) simbólico y estadísticos para el lenguaje de programación Python. NLTK incluye demostraciones gráficas y datos de muestra.

Está destinado a apoyar la investigación y la enseñanza en procesamiento de lenguaje natural (NLP) o áreas muy relacionadas, que incluyen la lingüística empírica, las ciencias cognitivas, la inteligencia artificial, la recuperación de información, y el aprendizaje de la máquina.

4.2.13 NumPy

Numpy es una biblioteca de funciones matemáticas provista por Python. Esta biblioteca brinda soporte para crear vectores y matrices grandes multidimensionales, junto con una gran colección de funciones matemáticas de alto nivel para operar con ellas.

Python debe su éxito en el campo investigativo y tratamiento de datos multidimensionales en gran parte a esta librería.

4.3 Entorno de desarrollo

4.3.1 Visual Studio Code - VSC

Es un editor de código fuente con una gran cantidad de funciones, tales como soporte para la depuración e integración con Git, entre otras. La opción a elegir fue Visual Studio Code ya que, nuevamente, el autor del PFC ya tenía conocimientos en este software.

5 Desarrollo

5.1 Conjunto de datos

La recolección y procesamiento del conjunto de datos fue, sin lugar a dudas, la etapa más demandante en cuanto a tiempo y esfuerzo durante el desarrollo del PFC. El siguiente gráfico (Figura 2 - Flujo de creación de dataset) representa el flujo que se siguió para generar el set de datos que posteriormente fue utilizado para entrenar el modelo de aprendizaje automático .

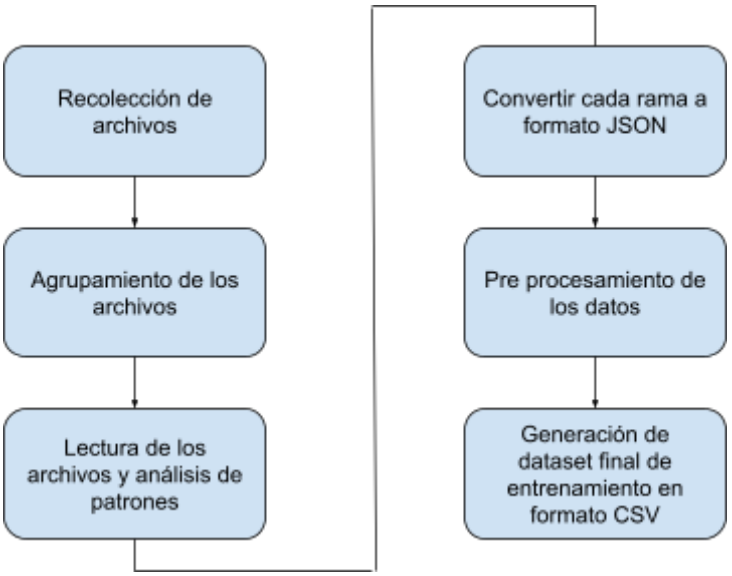

Figura 2 - Flujo de creación de dataset.

5.1.1 Recolección de archivos

Para la recolección del conjunto de datos aplicables al proyecto se realizaron búsquedas web. Principalmente se consultaron webs pertenecientes al gobierno de la república argentina.

La mayor fuente de datos se obtuvo consultando el digesto de la normativa Argentina (*Digesto Jurídico Argentino*, 2022.). Dicho digesto posee la clasificación de las leyes argentinas publicadas hasta 16 de junio de 2014. Estas leyes se encuentran en formato PDF y la clasificación de las mismas es más amplia que la de los objetivos definidos, esto es desarrollado con mayor detalle en la siguiente sección.

5.1.2 Definición de entidades

Existen diferentes maneras de dividir las ramas del derecho argentino (*Derecho Argentino*, 2022), algunas están más descompuestas que otras. Para los fines de este PFC se tomó el tercer nivel del árbol que se presenta en la siguiente imagen (Figura 3 - Definición de entidades). Quedando compuestas así las nueve ramas de estudio.

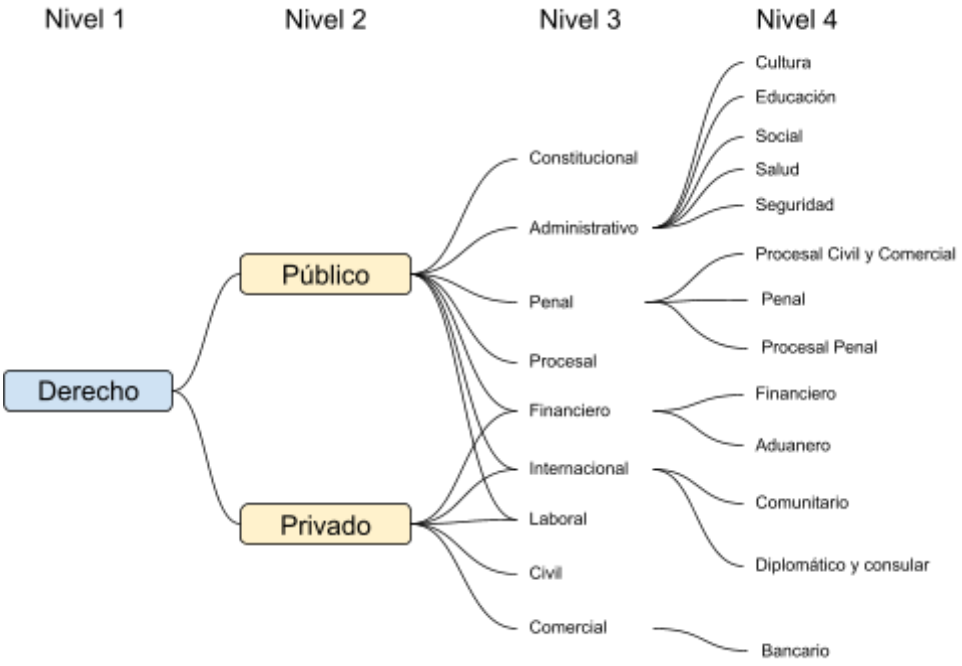

Figura 3 - Definición de entidades.

De la imagen anterior (Figura 3 - Definición de entidades), se puede observar que las ramas Laboral, Internacional y Financiero pueden pertenecer tanto al ámbito Público como al Privado.

A fines de simplificar los estudios, además de no estar dentro de los alcances del PFC, no se evalúa si alguna de las leyes pertenecen al ámbito Público o Privado, sólo se evalúa a qué rama del tercer nivel están relacionadas. Es decir, cualquiera sea la ley a analizar, esta se debe clasificar en alguna de las categorías que se muestran en el nivel 3 de la Figura 3.

En la siguiente tabla (Tabla 1 - Ejemplos de entidades) se presentan las ramas (o entidades) de clasificación definidas, acompañada de ejemplos de fragmentos de leyes de cada una de ellas.

Rama o entidad	Ejemplo

Constitucional	De los quince (15) cargos de representantes que forman parte del Ministerio, ante la Justicia del Trabajo de la Capital de la República, se transfieren al ministerio público ante los juzgados federales de la misma jurisdicción, creando con ellos otros procedimientos fiscales.
Administrativo	Realizar tareas de investigación conducentes a la formación de grupos, que posean disciplinas y técnicas necesarias para el acceso a la tecnología espacial y sus aplicaciones.
Penal	Todo el que ejerciendo autoridad civil o militar hiciese azotar algún individuo de cualquier clase o condición que fuere, será declarado inhábil para ejercer ningún empleo nacional durante diez (10) años, sin perjuicio de las acciones a que diere lugar la gravedad del hecho.
Procesal	En cuanto el estado de la causa lo permita, el dinero, títulos y valores secuestrados se depositarán, como pertenecientes a aquélla, en el Banco de la Nación Argentina, sin perjuicio de disponerse, en cualquier estado de la causa, la entrega o transferencia de dichos bienes si procediere. En el caso de las causas que tengan cuentas abiertas en el Banco de la Ciudad de Buenos Aires a la fecha de entrada en vigencia de la presente ley; el dinero, títulos y valores secuestrados correspondientes a esas causas se depositarán en dicha entidad y se mantendrán unificados hasta la extinción de las causas que le dieron origen.
Financiero	Cuando se produjera un incremento en más de las alícuotas correspondientes a los derechos de exportación de productos agropecuarios alcanzados por las disposiciones de la Ley 21453, en el período comprendido entre el Registro de la Declaración Jurada de Venta al Exterior (DJVE) y el de la oficialización de la correspondiente Destinación de Exportación, los exportadores deberán acreditar de modo fehaciente la tenencia o en su caso la adquisición de tales productos con anterioridad al aludido incremento
Internacional	Al adherir a los citados instrumentos, deberá formularse la siguiente declaración: "Conforme con los artículos 96 y 12 de la "Convención de las Naciones Unidas sobre los Contratos de Compraventa Internacional de Mercaderías", cualquier disposición del artículo 11, del artículo 29 o de la Parte II de la misma que permita que la celebración, la modificación o la extinción por mutuo acuerdo del contrato de compraventa, o la oferta, la aceptación o cualquier otra manifestación de intención, se hagan por un procedimiento que no sea por escrito, no se aplicará en el caso de que cualquiera de las partes tenga su

	establecimiento en la República Argentina".
Laboral	Se entenderá por Sociedad Laboral a aquella sociedad de cualquier tipo, en la que la mayoría del capital social sea de propiedad de los trabajadores que presten en ella servicios retribuidos en forma personal y directa y cuya relación laboral se establezca por tiempo indeterminado. Podrán obtener la calificación de Sociedad Laboral aquellas personas jurídicas en las que el número de horas trabajadas por los trabajadores contratados por tiempo determinado o indeterminado que no revistan la calidad de socios, no superen el quince por ciento (15 %) del total de las horas año trabajadas por los socios.
Civil	A partir de su entrada en vigencia, las leyes se aplicarán aún a las consecuencias de las relaciones y situaciones jurídicas existentes. No tienen efecto retroactivo, sean o no de orden público, salvo disposición en contrario. La retroactividad establecida por la Ley en ningún caso podrá afectar derechos amparados por garantías constitucionales.
Comercial	Asumir obligaciones de pago con destino fiscal, derivados de contratos de mutuo amparados por la emisión en garantía o entrega en caución de los títulos. El Banco Central de la República Argentina imputará los pagos que efectúe como consecuencia de lo dispuesto en este artículo a la "Cuenta Resultado de Operaciones de Cambio", integrante del balance de la mencionada Institución

Tabla 1 - Ejemplos de entidades

5.1.3 Agrupamiento de datos

Como se menciona en la sección 5.1.1, la clasificación de las leyes en las diferentes ramas del derecho brindada por el digesto de la normativa Argentina es más amplia (nivel 4 de Figura 3 - Definición de entidades) que las diferentes clasificaciones definidas en el alcance del PFC.

Por ejemplo, la rama de estudio Administrativo se encuentra dividida en Administrativo/Cultura, Administrativo/Educación, Administrativo/Salud, Administrativo/Seguridad y Administrativo/Social. Existen casos en los que la clasificación final no se deduce de manera directa (por ejemplo, Derecho aduanero), por lo que se debió consultar a profesionales del área de abogacía para ubicar la clasificación en la rama correspondiente, dando como resultado la división de leyes en las nueve ramas objetivo.

En la siguiente tabla (Figura 2 - Agrupamiento de datos) se detallan en la columna de la izquierda las clasificaciones originales de digesto y en la columna de la derecha la clasificación aplicada al proyecto.

Clasificación de digesto	Clasificación aplicada al proyecto
Administrativo/Cultura	Administrativo
Administrativo	
Administrativo/Educación	
Administrativo/Salud	
Administrativo/Seguridad	
Administrativo/Social	
Aduanero	Financiero
Impositivo	
Bancario	Comercial
Comercial	
Civil	Civil
Comunitario	Internacional
Diplomático y consular	
Internacional Público	
Internacional Privado	
Constitucional	Constitucional
Laboral	Laboral
Penal	Penal
Procesal Civil y Comercial	
Procesal Penal	

Figura 2 - Agrupamiento de datos.

5.1.4 Lectura de los archivos y análisis de patrones

Una vez agrupados los diferentes archivos PDF que contienen los textos de ley, el siguiente paso fue tomar muestras aleatorias a fin de analizarlos y encontrar patrones en la estructura de los mismos. En esta etapa se identificaron encabezados, textos irrelevantes y referencias que no aportan al entrenamiento del modelo.

A continuación se listan algunos ejemplos.

- Encabezados: los archivos comienzan con un encabezado similar al presentado en la siguiente imagen.

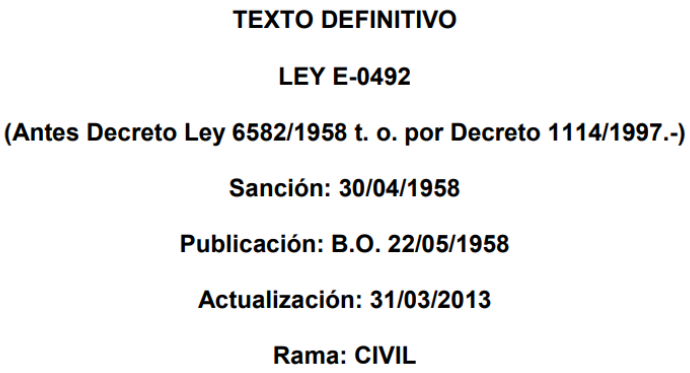

Figura 4 - Encabezados de documentos.

- Textos irrelevantes: todos los archivos cuentan con textos que no aportan información de utilidad para el modelo. Algunos ejemplos son: *Título X, Artículo X, Art. X, DNU X.*

- Referencias: en la gran mayoría de archivos, al final de los mismos, existen una serie de referencias que no aportan información útil. En la siguiente imagen se puede ver un ejemplo.

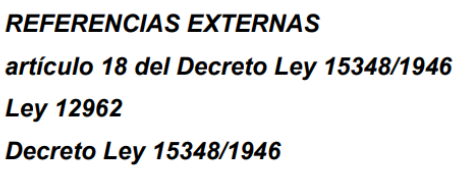

Figura 5 - Referencias de documentos.

Los patrones antes mencionados fueron eliminados en etapas posteriores.

5.1.5 Convertir cada rama a formato JSON

A fin de agrupar los textos de cada ley en un único lugar, se decidió generar un archivo JSON. Los archivos JSON además facilitan la manipulación del texto si lo comparamos con el formato original de los datos (formato PDF).

Para realizar esta conversión se hizo uso de la herramienta Tika (Sección 4.2.8 - Tika Parser), la cual en unos pocos segundos extrae el texto almacenado en un

PDF. El texto resultante de cada ley es almacenado en un archivo JSON con la siguiente estructura:

> [*{"clasificacion": "etiqueta 1", "contenido": "Esto es un texto de ejemplo"}, {"clasificacion": "etiqueta 2", "contenido": "Esto es otro texto de ejemplo"}*]

Donde:

- Cada elemento representa una ley.
- Cada *clasificacion* dentro de un elemento representa a la etiqueta asignada a la ley.
- Cada *contenido* dentro de un elemento representa al texto de cada ley.

5.1.6 Preprocesamiento de datos

En primer lugar, es importante comprender por qué se necesita reducir el número de palabras en el texto que será usado para el aprendizaje de las máquinas.

Para los seres humanos, extraer información relevante de un texto es una tarea trivial, pero es un verdadero reto para una computadora. En muchas ocasiones, no interesa conocer todos los significados de un texto sino que solamente algunos necesarios para realizar una tarea. Una buena práctica para obtener la información relevante de un texto consiste en eliminar los elementos que no aportan significado.

Para alcanzar este objetivo, se implementan algunas prácticas conocidas en NLP para obtener un conjunto de datos de calidad, las mismas se presentan a continuación.

5.1.6.1 Tokenización

El primer paso es delimitar las palabras del texto, y convertir esas palabras en elementos de una lista (*Tokenization*, 2009). Este procedimiento es conocido como tokenización. Aquí es donde entra en juego la librería spaCy (Sección 4.2.6). A continuación se presenta un ejemplo de una cadena de texto antes de aplicar la tokenización, como así también la lista resultante luego de hacerlo.

"Esto es un ejemplo" → ["Esto", "es", "un", "ejemplo"]

5.1.6.2 Limpieza de texto

La limpieza del texto consiste en eliminar palabras muy comunes y poco informativas desde el punto de vista léxico, tales como conjunciones (y, o, ni, que), preposiciones (a, en, para, por, entre otras) y verbos muy comunes (ser, ir, entre otros). Estas palabras poco representativas son conocidas como *stopwords* (*Dropping Common Terms: Stop Words*, 2009).

"Esto es un ejemplo para mostrar stopwords" → ["Esto", "ejemplo", "mostrar", "stopwords"]

Por otro lado, en esta instancia también se eliminaron los patrones mencionados en la etapa 5.1.4 Lectura de los archivos y análisis de patrones.

5.1.6.3 Normalización

El siguiente paso en el flujo de trabajo consiste en normalizar el texto. Esta etapa incluye la uniformización en el uso de mayúsculas, acentos, signos de puntuación, eliminar saltos de línea y espacios duplicados.

5.1.6.4 Lematización

En el idioma español, existen varias palabras diferentes en representación de una misma palabra. Por ejemplo, canto, cantas, canta, cantamos y cantan son distintas formas (conjugaciones) de un mismo verbo (cantar). Y que niña, niño y niños, son distintas formas del vocablo niño.

Con el fin de obviar las diferencias mencionadas, estas palabras se unifican en un mismo término. Esto es lo que se conoce como lematización (Khan, 2022), lo cual básicamente consiste en relacionar una palabra flexionada o derivada con su forma canónica o lema.

5.1.6.5 Stemming

Este proceso, aunque es similar al anterior, tiene sus pequeñas diferencias. Stemming es el procedimiento de convertir palabras en raíces. Estas raíces son la parte invariable de palabras relacionadas sobre todo por su forma. De cierta manera se parece a la lematización, pero los resultados (las raíces) no tienen por qué ser palabras de un idioma. Por ejemplo, el algoritmo de stemming puede decidir que la raíz de amamos no es am- (la raíz que) sino amam.

Normalmente, las técnicas de Lematización y Stemming no suelen aplicarse en conjunto durante el preprocesamiento de texto (*Stemming and Lemmatization*, 2009). En este PFC, se aplicaron ambas técnicas y se seleccionó la que arrojó mejores resultados de entrenamiento, la cual fue Stemming.

5.1.7 Generación de dataset final de entrenamiento en formato CSV

El último paso en el tratamiento de datos es almacenar los resultados de preprocesados con su respectiva etiqueta. Para facilitar la visualización y análisis resulta conveniente almacenarlos en un archivo de formato CSV. A continuación (Figura 3 - Ejemplo de CSV resultante) podemos ver un pequeño ejemplo del mismo

Labels	Text
Constitucional	['presupuest', 'pod', 'judicial', 'nacion', 'recurs', 'afectacion', 'especif', 'pod', 'judicial', 'nacion', 'establec', 'articul', 'deg']
Procesal	['mediacion', 'conciliacion', 'objet', 'establec', 'caract', 'obligatori', 'mediacion']
Civil	['forestal', 'constitu', 'favor', 'tercer', 'titular', 'domini', 'condomini', 'inmuebl', 'suscept']
Laboral	['mont', 'extension', 'increment', 'part', 'marz', 'mont', 'minim', 'maxim', 'prestacion']

Figura 3 - Ejemplo de CSV resultante.

5.2 Investigación y análisis de herramientas disponibles

Esta sección está destinada a realizar un análisis del proceso de selección y análisis de los diferentes modelos y herramientas aplicables a la clasificación de texto.

5.2.1 Bibliotecas

El objetivo de esta sección es justificar la elección de una librería que cuente con una serie de características esperadas y que permita implementar PLN.

5.2.1.1 características deseables

A continuación se listan las diferentes características que se esperan del modelo.

- Soporte en español: es imprescindible que las bibliotecas sean capaces de interpretar el idioma español ya que es el idioma en el que el contenido de las leyes está redactado.

- Compatibilidad con Python: el código del PFC está escrito únicamente en el lenguaje de programación Python, por lo que es necesario que la biblioteca tenga compatibilidad con dicho lenguaje.

- Forma de ejecución: la librería debe ser capaz de ejecutarse ya sea de manera local o de manera remota en Google Collaborative (Sección 4.2.7).

- Libre código abierto: programas libres y de código abierto que estén licenciados de tal manera que los usuarios pueden estudiar, modificar y mejorar su diseño.

- Comunidad: es necesario contar con librerías de comunidad activas. Esto trae beneficios a la hora de consultar foros y despejar dudas.

5.2.1.2 Herramientas analizadas

Partiendo del gran abanico de opciones que hay disponibles para implementar NLP, se hizo una primera selección de las siguientes herramientas:

- NLTK: kit de herramientas de lenguaje natural, o más comúnmente NLTK, es un conjunto de bibliotecas y programas para el procesamiento del lenguaje natural simbólico y estadísticos (*NLTK - Natural Language Toolkit*, 2022). NLTK incluye demostraciones gráficas y datos de muestra.

- spaCy: spaCy es una librería de software para procesamiento de lenguajes naturales (*SpaCy · Industrial-Strength Natural Language Processing in Python*, 2022). Está diseñada para facilitar tareas de procesamiento avanzado del lenguaje natural. Además, posee modelos ya entrenados disponibles para diferentes idiomas y aplicaciones.

- ApacheOpenNLP: juego de herramientas basado en aprendizaje automático para el procesamiento de texto en lenguaje natural (*Apache OpenNLP*, 2022). Básicamente, es una librería basada en ML para NLP, desarrollada por Apache Software Foundation.

- MonkeyLearn: Es una herramienta basada en IA, capaz de procesar grandes cantidades de texto (*MonkeyLearn - Text Analytics*, 2022)

5.2.1.3 Comparativas y elección final

En la siguiente figura (Figura 4 - resumen de herramientas consideradas) podemos observar las diferentes herramientas evaluadas.

	Soporte en español	Compatibilidad python	Forma de ejecución	Libre código abierto	Comunidad
NLTK	Si	Sí	Local o nube	Sí	Activa

spaCy	Si	Sí	Local o nube	Sí	Activa
ApacheOpenNLP	Si	No	Local	Sí	Activa
MonkeyLearn	Si	Sí	Remoto	No	Activa

Figura 4 - resumen de herramientas consideradas.

Como podemos observar, NLTK y spaCy son las herramientas que mejor se adaptan a los objetivos del proyecto. A continuación, una breve fundamentación de dicha elección.

En primer lugar, este PFC está destinado a etiquetar las diferentes leyes del BORA. Dichas leyes están redactadas en español, por lo que es imprescindible que las herramientas cuenten con soporte en este idioma. Para este caso, todas las opciones cuentan con esta funcionalidad.

Por otro lado, como se menciona en la sección “4.3 Entorno de desarrollo”, el lenguaje seleccionado para llevar a cabo este PFC es Python. De la Figura 4, podemos observar que de la lista preliminar la única herramienta que no tiene compatibilidad con Python es ApacheOpenNLP, la cual está disponible para Java Por este motivo, ApacheOpenNLP queda descartada de la lista de herramientas.

En cuanto a la forma de ejecución, se espera que el código pueda ser ejecutado tanto de manera local como en el entorno de Google Collaborative (Sección 4.2.7). Lo cual deja afuera de la lista a MonkeyLearn.

Por las anteriores razones, las dos herramientas utilizadas en el proyecto son NLTK y spaCy. Durante el desarrollo, se busca aprovechar las mejores características de ambas. Por ejemplo, las *Stopwords* del texto de las leyes son eliminadas mediante NLTK, ya que esta herramienta posee un diccionario más preciso y extenso comparado con spaCy. Por otro lado, para *tokenizar* el texto presente en los documentos de las leyes se elige spaCy por encima de NLTK ya que su tiempo de respuesta es menor y su desempeño es mayor.

Se concluye entonces que ambas herramientas pueden ser utilizadas en la limpieza y preprocesamiento de texto ya que se complementan sin interferir una a otra.

5.2.2 Modelos

En esta sección se presentan los diferentes modelos de clasificación de texto evaluados. Primero se hace mención a los diferentes algoritmos considerados para llevar a cabo el proyecto final de carrera, acompañado de una breve justificación en

la elección final de los algoritmos a evaluar. Por último, presenta un análisis de cada uno de los modelos finalmente seleccionados.

5.2.2.1 Modelos considerados

La sección 5.2.2.1 presenta los diferentes algoritmos considerados previo a la construcción del prototipo de clasificación de textos.

5.2.2.1.1 Máquinas de soporte vectorial (SVM)

Estos métodos están propiamente relacionados con problemas de clasificación y regresión. Dado un conjunto de ejemplos de entrenamiento, podemos etiquetar las clases y entrenar una SVM para construir un modelo que prediga la clase de una nueva muestra (*Máquinas De Vectores De Soporte*, 2022).

Una SVM se puede representar como puntos en un espacio donde el objetivo es crear un plano, llamado hiperplano, que se encarga de dividir estos puntos para crear particiones homogéneas. Cuando las nuevas muestras se ponen en correspondencia con dicho modelo, en función de su proximidad pueden ser clasificadas a una u otra clase.

El objetivo detrás de este algoritmo es intentar que el hiperplano que separa las clases tenga el mayor margen entre éste y los puntos más cercanos de cada clase. En la siguiente figura nos encontraremos con una representación de diferentes hiperplanos creados en un espacio. Podremos comprobar como H1 y H2, aunque separan las clases, no las separa de manera óptima, como lo hace el H3.

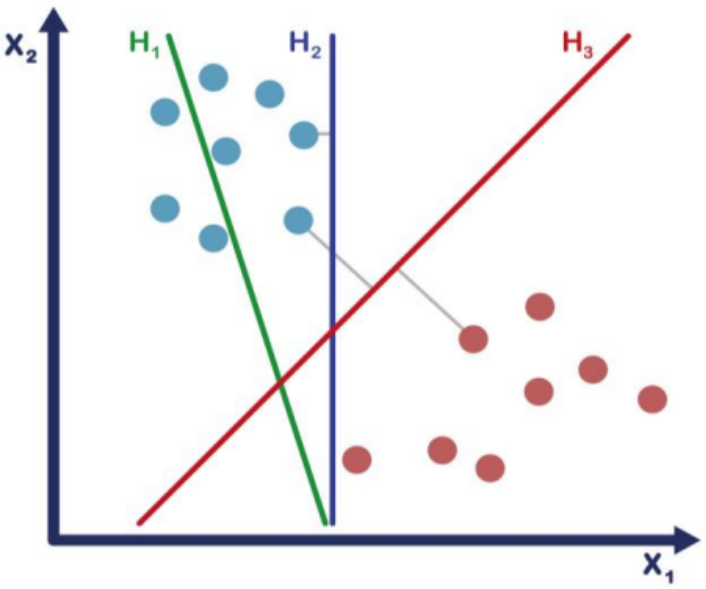

Figura 6 - Máquinas de soporte Vectorial.

Este algoritmo es muy útil para modelos de datos complejos, pero su principal desventaja es que es un modelo lento de entrenar, ya que demanda un elevado poder de cómputo al momento de encontrar un hiperplano para separar los datos.

Existen situaciones donde es posible que el hiperplano no sea capaz de separar los puntos de las clases de manera precisa. En estos casos, el algoritmo asigna un parámetro C cuya función es compensar estos errores que se presentan en el momento del entrenamiento.

5.2.2.1.3 Regresión logística

Este algoritmo es utilizado para conocer la probabilidad de que una variable se encuentre en una categoría. La variable es tratada como una variable binaria que tiene datos codificados como 1 o 0 (*Regresión Logística*, 2022).

Originalmente, este algoritmo sirve para clasificar solamente dos clases, pero la librería *sklearn* utilizada en este PFC permite extender la utilización de Regresión Logística a múltiples clases mediante el uso de *one* vs *rest*, la cual consiste en entrenar un clasificador por cada clase. De esta manera, el modelo comprueba si un dato como entrada pertenece o no a una clase, ya que el resultado será positivo (1) o negativo (0).

Podemos representar el funcionamiento de la regresión logística en la siguiente figura:

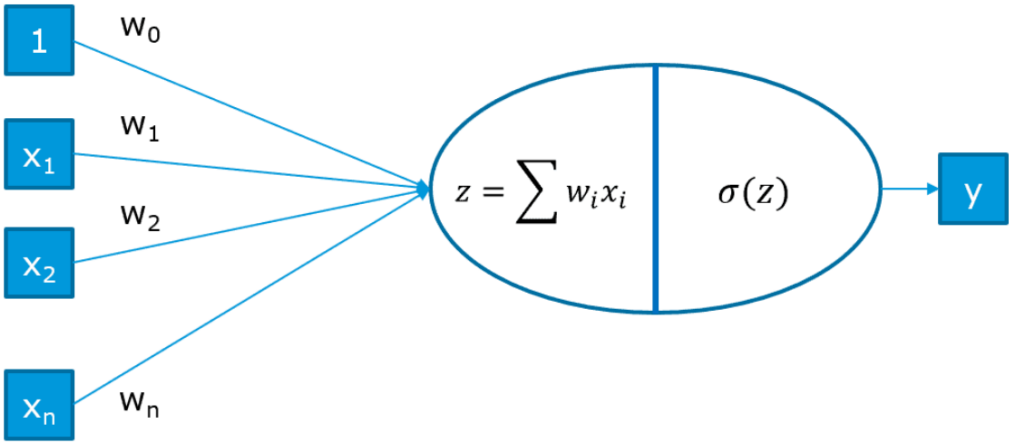

Figura 7 - Regresión logística.

Los diferentes valores de x corresponden a distintos atributos del problema a resolver. Por ejemplo, si queremos saber determinar si un texto pertenece o no al ámbito Civil se podría evaluar si dicho texto se relaciona con materias como por ejemplo la adopción, el nombre de las personas o los bienes que las personas pueden poseer. La predicción sería la probabilidad de que el texto pertenezca o no al ámbito Civil.

En términos matemáticos, se puede formular de la siguiente manera:

$$y = \sigma(z) = \sigma(WX) = \sigma(\sum(w_i x_i)) = \sigma(\sum(w_0 x_0 + w_1 x_1 + ... + w_n x_n))$$

Como se puede observar, la regresión logística tiene dos partes:

- Una combinación lineal (a la izquierda de la neurona)
- Aplicación de la función logística (a la derecha de la neurona)

Todas las entradas son combinadas con una línea con los coeficientes w. Y luego se aplica la función logística (también llamada sigmoide) al resultado. Matemáticamente, la función logística o sigmoide, se puede expresar:

$$\sigma(z) = \frac{1}{1 + e^{-z}}$$

Y cuya representación gráfica es:

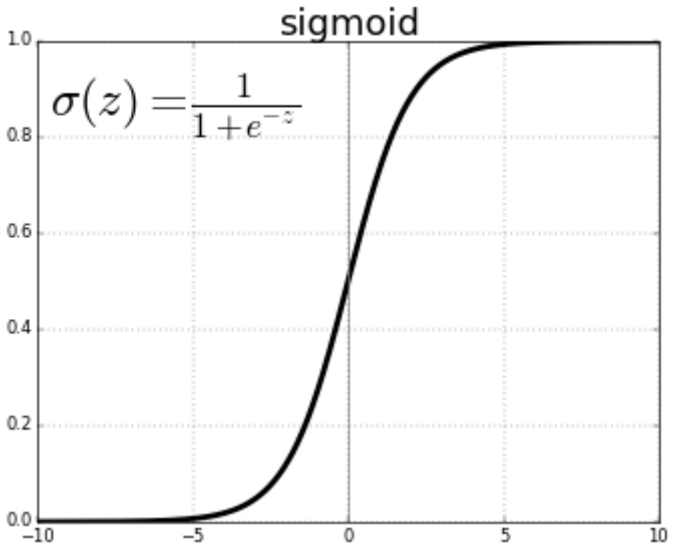

Figura 8 - Función sigmoide.

Podemos observar que las carácteristicas de la función logística son:

- Su valor mínimo es 0 y el máximo es 1.
- Podemos interpretar sus resultados como probabilidades (por ejemplo, una probabilidad 0.85 de que el texto pertenezca a Civil)

- Para problemas de clasificación binaria, como en nuestro caso, podemos suponer que los valores menores de 0.5 corresponden a la clase 0 y los superiores a 0.5 a la clase 1.

Cabe mencionar también que la regresión logística es una técnica muy empleada debido a su eficacia y simplicidad. Es decir, esta estrategia es fácil de implementar, interpretar y muy eficiente de entrenar. Otra característica importante a destacar es que no es necesario disponer de grandes recursos computacionales, tanto en entrenamiento como en ejecución.

5.2.2.1.4 Árboles de Decisión o Clasificación

Esta técnica funciona utilizando diferentes modelos que al final son combinados para poder obtener un modelo de todo el conjunto. Árboles de Decisión destacan por su sencillez (*Árbol De Decisión*, 2022).

Un árbol de decisión es un mapa de los posibles resultados de una serie de decisiones relacionadas. Por lo general, comienza con un único nodo y luego se ramifica en resultados posibles. Cada uno de esos resultados crea nodos adicionales, que se ramifican en otras posibilidades. Esto le da una forma similar a la de un árbol. Posteriormente, las predicciones son combinadas con medias de regresión.

La debilidad de este algoritmo es que no es fácilmente interpretable, lo cual torna difícil la tarea de ajustar parámetros. Además, son inestables ya que un pequeño cambio en los datos de entrada puede suponer un árbol de decisión completamente diferente.

5.2.2.1.5 K-Nearest Neighbors (KNN)

Es una de las técnicas más simples de clasificación disponibles. Este algoritmo utiliza la proximidad para hacer clasificaciones o predicciones sobre la agrupación de un punto de datos individual (*K Vecinos Más Próximos*, 2022).

Básicamente, el algoritmo calcula la distancia entre el ítem a clasificar y el resto de ítems del dataset de entrenamiento. Luego, se seleccionan los “K” elementos más cercanos y luego se realiza una “votación por mayoría”

Para realizar una clasificación, se asigna una etiqueta de clase sobre la base de un voto mayoritario, es decir, se utiliza la etiqueta que se representa con más frecuencia alrededor de un punto de datos determinado.

Por ejemplo, observemos la siguiente figura y consideremos que se escogen los K puntos más cercanos, para es caso K=5. Por lo que nuestro punto verde será azul, ya que de los 5 puntos más cercanos 3 son azules y 2 son de color rojo. Es

decir, por mayoría ganan los puntos azules y la nueva entrada será clasificada como azul.

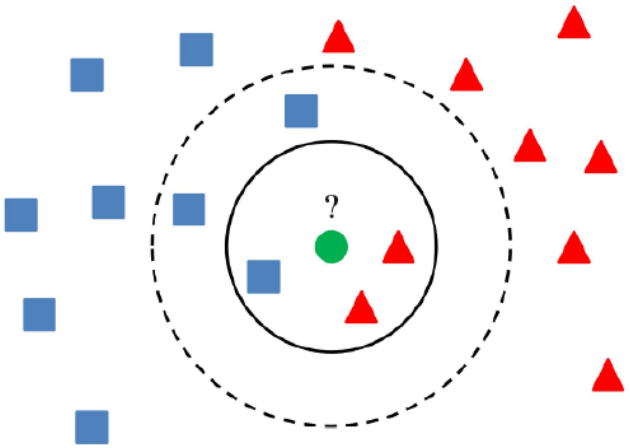

Figura 9 - KNN.

Como pros tiene sobre todo que es sencillo de aprender e implementar. Por otro lado, su contra es que no funciona bien con conjuntos de datos grandes, ya que calcular las distancias entre cada instancia de datos es muy costoso. Es decir, KNN tiende a funcionar mejor en datasets pequeños.

5.2.2.1.6 Elección final

Debido a la inestabilidad ante los cambios, el único método a descartar de la lista de modelos a probar es Árboles de Decisión o Clasificación (sección 5.2.1.4).

Por otro lado el modelo Máquinas de soporte vectorial (sección 5.2.1.1), dejando de lado el poder de cómputo requerido para entrenarlo, resulta una opción interesante. Es por esto que se llevaron a cabo entrenamientos y pruebas del modelo antes de descartar el mismo.

Por el lado de la Regresión Logística, teniendo en cuenta la experiencia del autor del proyecto con IA y los recursos disponibles, parece una buena opción dada su eficacia, simplicidad y baja demanda de recursos. Es por esto, que las primeras pruebas y evaluaciones de resultados fueron llevadas a cabo con Regresión Logística. Una vez más afianzado en las salidas generadas a partir de los diferentes tamaños de datasets, se implementan nuevos algoritmos.

Por último, como se mencionó en K-Nearest Neighbors (sección 5.2.1.6), este algoritmo tiene como ventaja que es sencillo de aprender, implementar y también tiene un buen funcionamiento en datasets pequeños. Por todo esto, se llevaron a cabo pruebas sobre este modelo.

5.2.3 Análisis de herramientas

Luego de definir los algoritmos de inteligencia artificial a utilizar, se procede con el desarrollo de códigos de prueba para evaluar los resultados de aplicar los algoritmos.

En este PFC, el primer paso fue comenzar a realizar pruebas sobre el modelo de Regresión Logística utilizando dataset limitado de 3 ramas. Esto fue con el objetivo de adquirir conocimientos sobre las herramientas, set de datos y métricas utilizadas. Durante el proceso de pruebas, se introdujeron nuevos ejemplares al set de datos hasta lograr el objetivo final de 9 ramas. Una vez conformado el conjunto de datos final, se realizaron las pruebas pertinentes a los demás modelos.

5.2.3.1 Definición de métricas

Al momento de entrenar un modelo y realizar las primeras predicciones del mismo, es necesario contar con métricas que brinden una idea sobre su desempeño. Para la tarea de clasificación, el modelo se evalúa midiendo el grado de coincidencia de una categoría predicha con la categoría real.

Para el caso de los aprendizajes supervisados puede definirse una matriz de confusión. Una matriz de confusión es una herramienta útil que permite visualizar el desempeño de un algoritmo de aprendizaje supervisado (*Guide to Confusion Matrix Terminology*, 2014). Ante una predicción del modelo, ésta puede caer en una de las cuatro categorías disponibles. Se toma como ejemplo la predicción de enfermedades en una persona. Las 4 opciones siguientes son las que conforman lo que se conoce como la matriz de confusión:

- Verdadero positivo (TP): El valor real es positivo y la prueba predijo también que era positivo. Por ejemplo, una persona está enferma y la prueba así lo demuestra.

- Verdadero negativo (VN): El valor real es negativo y la prueba predijo también que el resultado era negativo. Por ejemplo, la persona no está enferma y la prueba así lo demuestra.

- Falso negativo (FP): El valor real es positivo, y la prueba predijo que el resultado es negativo. La persona está enferma, pero la prueba dice de manera incorrecta que no lo está. Esto es lo que en estadística se conoce como error tipo II

- Falso positivo (FP): El valor real es negativo, y la prueba predijo que el resultado es positivo. La persona no está enferma, pero la prueba nos dice de manera incorrecta que sí lo está.

- Esto es lo que en estadística se conoce como error tipo I

En la siguiente imagen se puede observar una representación de una matriz de confusión:

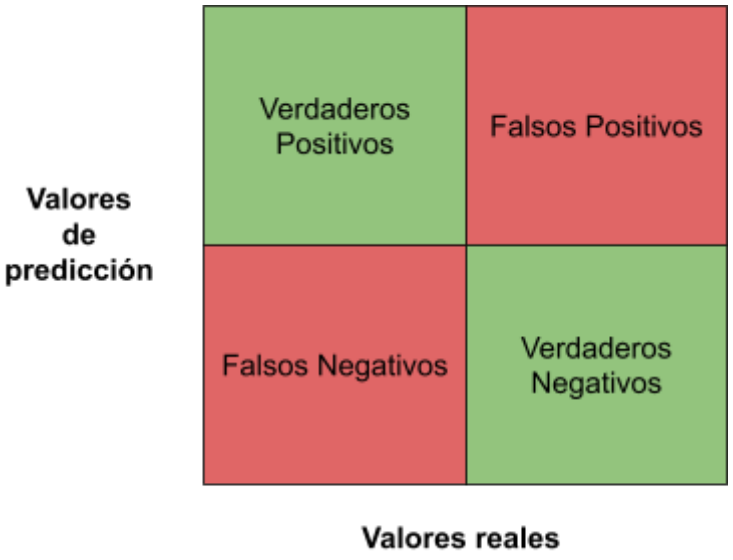

Figura 10 - Matriz de confusión.

Métricas de evaluación para la clasificación binaria

Para conocer qué tan bien se realizan las predicciones en los modelos, la librería spaCy (*Scorer - SpaCy API Documentation*, 2022) nos brinda la siguientes métricas:

- Precisión: La precisión es la proporción de predicciones correctas con un conjunto de datos de prueba. Se refiere a la dispersión del conjunto de valores obtenidos a partir de mediciones repetidas de una magnitud.
  Cuanto los resultados están más cerca de 1,00, indica que el modelo realizará mejores predicciones. Pero exactamente 1,00 indica un problema (normalmente: fuga de etiqueta/destino, sobreajuste o pruebas con datos de entrenamiento). Cuando los datos de prueba están desequilibrados (donde la mayoría de las instancias pertenece a una de las clases), el conjunto de datos es pequeño o las puntuaciones se acercan a 0,00 o 1,00, la precisión no captura realmente la eficacia de un clasificador y es necesario comprobar métricas adicionales.
  Para calcular la precisión usaremos la siguiente fórmula:

$$precision = \frac{TP}{TP + FP}$$

- Recall (Exhaustividad): La métrica de exhaustividad nos va a informar sobre la cantidad que el modelo de machine learning es capaz de identificar. Recall es conocida como el ratio de verdaderos positivos, es utilizada para saber cuántos valores positivos son correctamente clasificados. Básicamente, es la fracción de instancias relevantes que han sido recuperadas.

$$recall = \frac{TP}{TP + FN}$$

- **F1:** Esta métrica es muy empleada porque nos resume la precisión y sensibilidad en una sola métrica. Esta es una métrica muy utilizada en problemas en los que el conjunto de datos a analizar está desbalanceado. Básicamente, nos permite combinar las medidas de precisión y recall en un sólo valor lo cual es práctico porque hace más fácil el poder comparar el rendimiento combinado de la precisión y la exhaustividad entre varias soluciones.

$$F1 = 2 \cdot \frac{precision \cdot recall}{precision + recall}$$

- Mean Accuracy: La precisión media está relacionada con la precisión media lograda en N pliegues de entrenamiento diferentes.

- Std. Deviation: La desviación estándar es un índice numérico de la dispersión de un conjunto de datos (o población). Mientras mayor es la desviación estándar, mayor es la dispersión de la población. La desviación estándar es un promedio de las desviaciones individuales de cada observación con respecto a la media de una distribución.

5.2.3.2 Entrenamiento

El conjunto de datos es dividido de manera que se pueda probar el desempeño de los modelos con datos diferentes a los utilizados para entrenar el modelo. De esta manera se obtiene una objetividad mayor, la cual permite determinar si el modelo ha aprendido bien a generalizar el conocimiento que se intenta transmitir o, si en cambio, se ha centrado en peculiaridades que no son relevantes en las frases que se utilizan para entrenarlo.

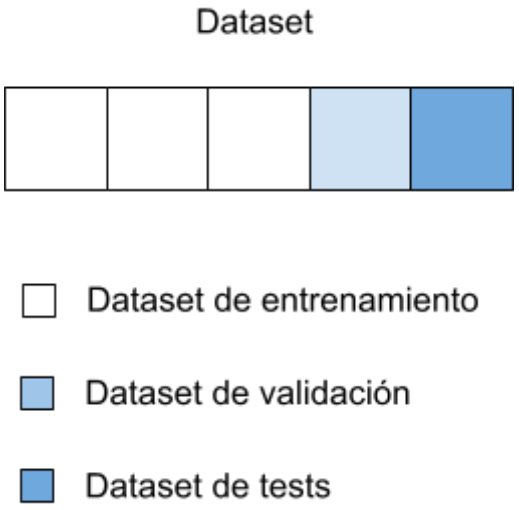

Figura 11 - Representación de dataset.

La división se realiza en: un 90% de observaciones (textos) para entrenar y un 10% para probarlo. Además se especifica una semilla (random state) de manera que la distribución 9:1 escogida en las muestras se mantenga siempre idéntica, lo cual permite saber si el modelo a mejorado en el ajuste de parámetros o si realmente ha sido una cuestión de azar al haber dado con una combinación de subconjuntos que se complementaban especialmente bien en los roles de entrenamiento y prueba.

Con el fin de asegurar que la asignación de los textos presentes en cada una de las frases a los diferentes 9 grupos sea proporcional a su predominancia en el conjunto de datos inicial, se utiliza una técnica conocida como K Fold. De esta manera no se condiciona el desempeño del algoritmo en ninguno de sus K entrenamientos por una posible distribución desbalanceada.

Figura 12 - Representación K Fold.

TFIDF

Sí bien se han aplicado diferentes técnicas para limpiar el dataset, todavía quedaban presentes palabras que se repetían en prácticamente todos los documentos. Por ejemplo, la palabra “asignación” es común en casi todas las circunstancias. Pero, sin embargo, la presencia de palabras como “comercialización”, pueden ser determinantes en la predicción.

Para resolver este problema se utiliza la técnica TF-IDF (*Term Frequency — Inverse Document Frequency*, 2019), en la que el peso que tiene cada palabra (*Wi,j*) es directamente proporcional a las veces que aparece en las frases de entrenamiento de la clase actual (*tfi,j*) e inversamente proporcional a las veces que aparece en las frases de entrenamiento de todas las clases (*dfi,j*).

$$w_{i,j} = tf_{i,j} \times \log\left(\frac{N}{df_i}\right)$$

tfi,j = Número de ocurrencias de i en j.
dfi,j = Número de documentos contenidos en i.
N = total de número de documentos.

5.2.3.3 Pruebas y análisis de Regresión Logística

A modo de realizar las primeras pruebas y tomar conocimiento sobre las herramientas de inteligencia artificial, se comenzó con un conjunto de datos limitado, contando con solo 3 categorías o ramas clasificadas.

Con 3 ramas

Las primeras pruebas de entrenamientos Regresión Logística se realizaron evaluando solamente 3 ramas. Estas son comercial, laboral y civil, a continuación (Figura 13) se presenta el set de datos correspondiente.

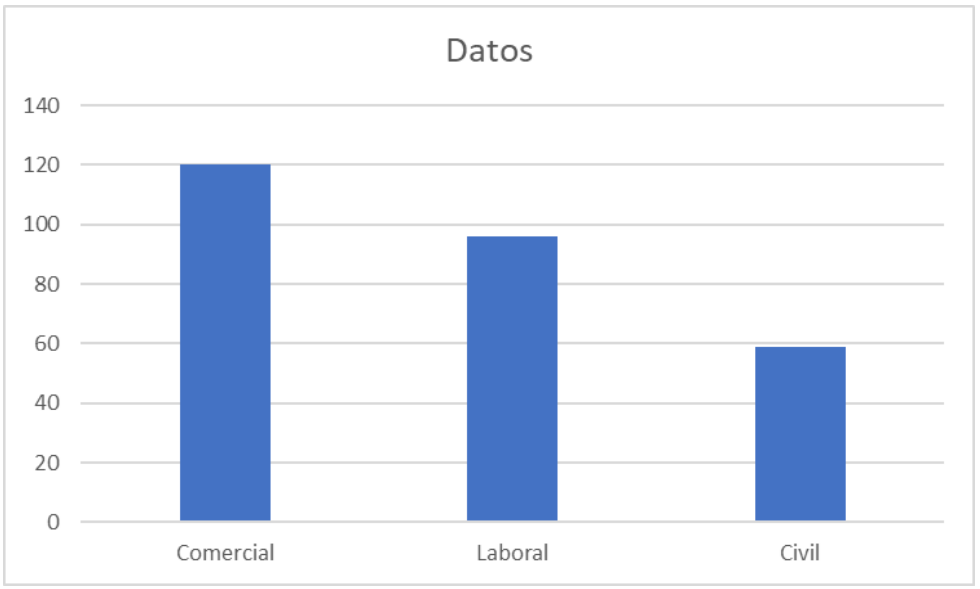

Figura 13 - Balance del conjunto de datos con 3 ramas.

Lo cual arrojó buenos resultados para ser una primera corrida, se obtuvo una precisión media mayor al 90%. A continuación (Figura 14), se presenta un resumen de los resultados.

```
Mean accuracy (Precisión media): 92,8% Std. deviation: +-0.161

              precision    recall  f1-score   support

       civil       1.00      0.71      0.83       7
   comercial       0.92      1.00      0.96      12
     laboral       0.89      1.00      0.94       8

   accuracy                           0.98      27
   macro avg       0.67      0.78      0.87      27
weighted avg       0.93      0.91      0.91      27

2 errores de 27 muestras de validación
```

Figura 14 - Resultados con 3 ramas.

Con 4 ramas:

Una vez que se agregó la rama de Financiero (Figura 15), los resultados (Figura 16) no variaron sustancialmente.

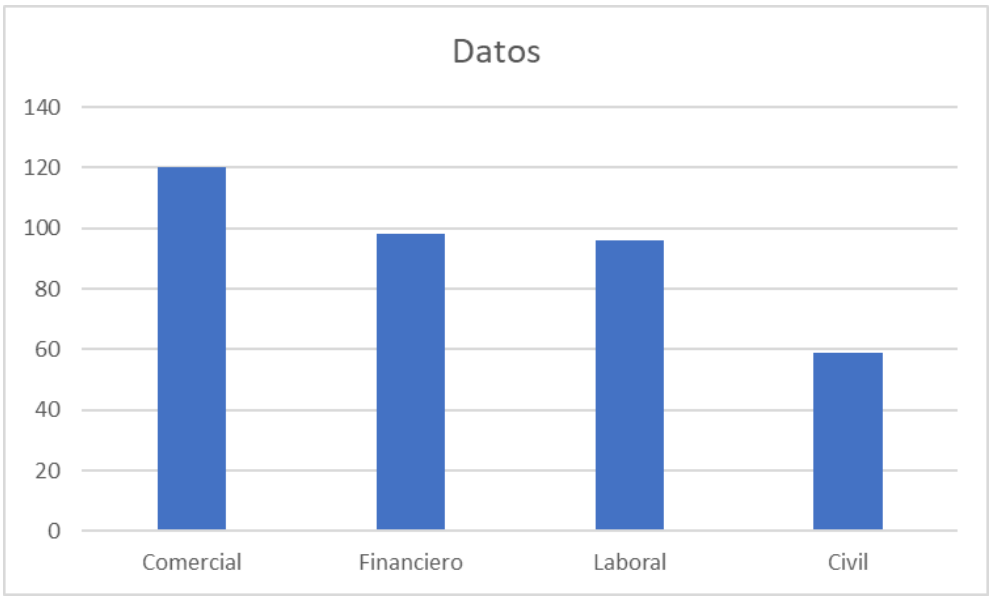

Figura 15 - Balance del conjunto de datos con 4 ramas.

```
Mean accuracy (Precisión media): 90,01% Std. deviation: +-0.183

              precision    recall  f1-score   support

       civil       1.00      0.83      0.91         6
   comercial       0.98      1.00      0.95         9
  financiero       1.00      1.00      1.00        16
     laboral       1.00      1.00      1.00         7

    accuracy                           0.97        38
   macro avg       0.97      0.96      0.96        38
weighted avg       0.98      0.97      0.97        38

1 error de 38 muestras de validación
```

Figura 16 - Resultados con 4 ramas.

Con 5 ramas:

Si bien el porcentaje de aciertos bajó, el resultado sigue siendo bueno ya que está cercano al 90%. La mayoría de los errores pertenecen a casos en los que el modelo predijo que la entrada proporcionada se trataba la rama Administrativo, cuando en realidad la entrada correspondía a una rama diferente, esto pudo deberse al marcado desbalance del conjunto de datos. En la Figura 18 y Figura 19 se presenta un resumen de los resultados obtenidos.

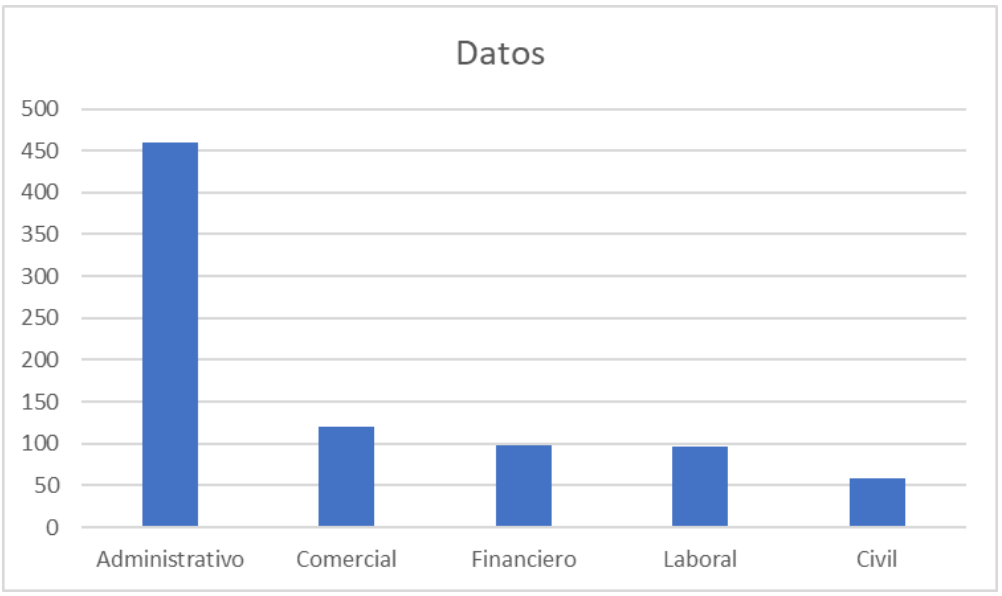

Figura 17 - Balance del conjunto de datos con 5 ramas.

```
Mean accuracy (Precisión media): 87,7% Std. deviation: +-0.143

                precision    recall  f1-score   support

administrativo       0.88      1.00      0.93       50
         civil       0.50      0.50      0.50        2
     comercial       1.00      0.70      0.82       10
    financiero       1.00      0.87      0.93       15
       laboral       1.00      0.75      0.96        8

   accuracy                             0.91       85
   macro avg       0.88      0.76      0.81       85
weighted avg       0.92      0.91      0.90       85

8 errores de 85 muestras de validación
```

Figura 18 - Resultados con 5 ramas.

```
Error 0 -> Se predijo administrativo, valor real civil
Error 1 -> Se predijo administrativo, valor real comercial
Error 2 -> Se predijo administrativo, valor real comercial
Error 3 -> Se predijo financiero, valor real civil
Error 4 -> Se predijo financiero, valor real financiero
Error 5 -> Se predijo administrativo, valor real financiero
Error 6 -> Se predijo administrativo, valor real financiero
Error 7 -> Se predijo administrativo, valor real comercial
```

Figura 19 - Errores con 5 ramas.

Con 7 ramas:

Luego de agregar las ramas de Procesal y Penal (Figura 19), las cuales cuentan con un dataset pequeño en relación a las demás ramas analizadas, el modelo mantuvo un buen porcentaje de aciertos.

Nuevamente, la mayor parte de errores estuvo relacionada a la rama Administrativo. En la Figura 20 se presenta un resumen de los resultados obtenidos.

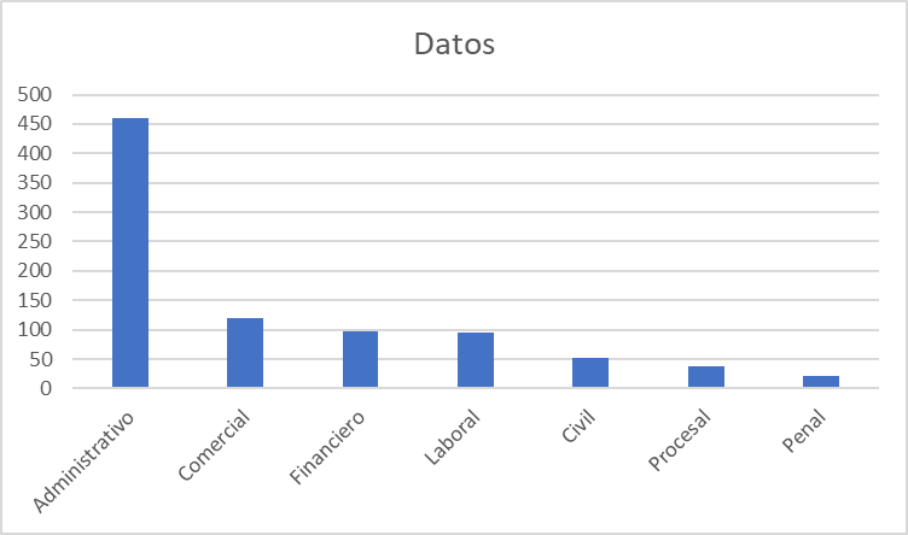

Figura 19 - Balance del conjunto de datos con 7 ramas.

Mean accuracy (Precisión media): 84,1% Std. deviation: +-0.105

	precision	recall	f1-score	support
administrativo	0.90	1.00	0.95	46
civil	0.50	0.50	0.50	2
comercial	0.86	0.92	0.89	13
financiero	1.00	0.76	0.87	17
laboral	1.00	1.00	1.00	5
penal	1.00	0.67	0.80	3
procesal	0.50	0.33	0.40	3
accuracy			0.91	85
macro avg	0.88	0.76	0.81	85
weighted avg	0.92	0.91	0.90	85

9 errores de 89 muestras de validación

Figura 20 - Balance del conjunto de datos con 7 ramas.

```
Error 0 -> Se predijo financiero, valor real laboral
Error 1 -> Se predijo financiero, valor real laboral
Error 2 -> Se predijo administrativo, valor real comercial
Error 3 -> Se predijo administrativo, valor real civil
Error 4 -> Se predijo administrativo, valor real procesal
Error 5 -> Se predijo administrativo, valor real comercial
Error 6 -> Se predijo administrativo, valor real procesal
Error 7 -> Se predijo comercial, valor real comercial
Error 8 -> Se predijo administrativo, valor real procesal
```

Figura 21 - Errores con 7 ramas.

Con 8 ramas:

Luego de agregar la rama Constitucional (Figura 22), la cual cuenta con un dataset amplio, los buenos resultados (Figura 23) se mantuvieron.

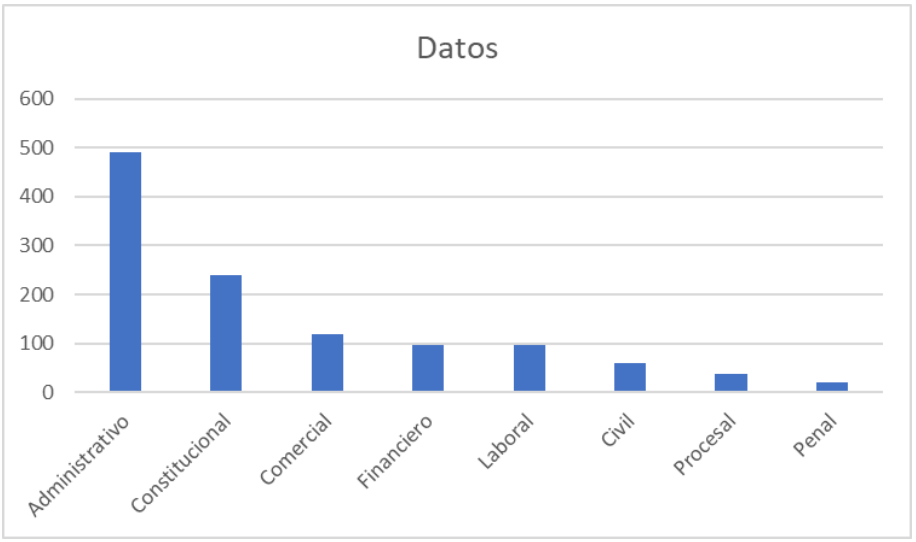

Figura 22 - Balance del conjunto de datos con 8 ramas.

Mean accuracy (Precisión media): 85,8% Std. deviation: +-0.099

	precision	recall	f1-score	support
administrativo	0.90	1.00	0.95	45
civil	1.00	1.00	1.00	5
comercial	1.00	0.90	0.95	10
constitucional	0.97	0.94	0.95	32
financiero	1.00	1.00	1.00	9
laboral	1.00	0.62	0.77	8
penal	1.00	0.67	0.80	3
procesal	0.67	1.00	0.80	2
accuracy			0.94	114
macro avg	0.94	0.89	0.90	114
weighted avg	0.95	0.94	0.94	114

8 errores de 89 muestras de validación

Figura 23 - Resultados con 8 ramas.

Reducción de dataset:
Antes de integrar la última rama a evaluar se redujo el dataset de Administrativo (Figura 24). Esto se hizo con el objetivo de descartar la posibilidad de que los errores presentes en las predicciones fueran causados por un sobreentrenamiento de dicha rama.

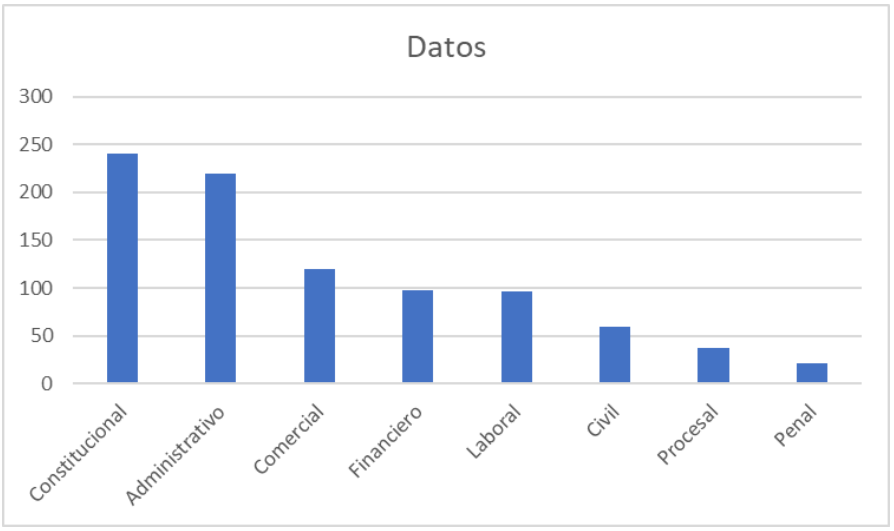

Figura 24 - Balance del conjunto de datos reducido de 8 ramas.

En la siguiente imagen (Figura 25) se puede observar que, si bien hay una leve mejoría, el resultado no tiene una gran variación, esto puede deberse a que el muestreo (véase la sección 5.2.2.1) estratificado implementado en el algoritmo ayuda a que la asignación de las frases en cada uno de nuestros K grupos sea proporcional a su predominancia en el conjunto de datos inicial, de manera que la

distribución se mantenga lo más natural posible. De esta manera, no se condiciona el desempeño del algoritmo en ninguno de sus K entrenamientos debido a una posible distribución desbalanceada

```
Mean accuracy (Precisión media): 88,2% Std. deviation: +-0.085

                precision    recall  f1-score   support

administrativo       0.86      0.96      0.91        25
         civil       0.00      0.00      0.00         1
     comercial       0.80      0.73      0.76        11
constitucional       0.96      0.89      0.93        28
    financiero       0.83      0.83      0.83        12
       laboral       0.89      1.00      0.94         8
         penal       1.00      0.50      0.67         2
      procesal       1.00      1.00      1.00         2

   accuracy                           0.88        89
   macro avg       0.79      0.74      0.75        89
weighted avg       0.88      0.88      0.87        89

7 errores de 89 muestras de validación
```

Figura 25 - Resultados con dataset reducido de 8 ramas.

Con 9 ramas:

Por último, luego de agregar la última la rama Internacional (Figura 26) el dataset final quedó conformado. Este dataset es la base de las siguientes pruebas presentes en el PFC.

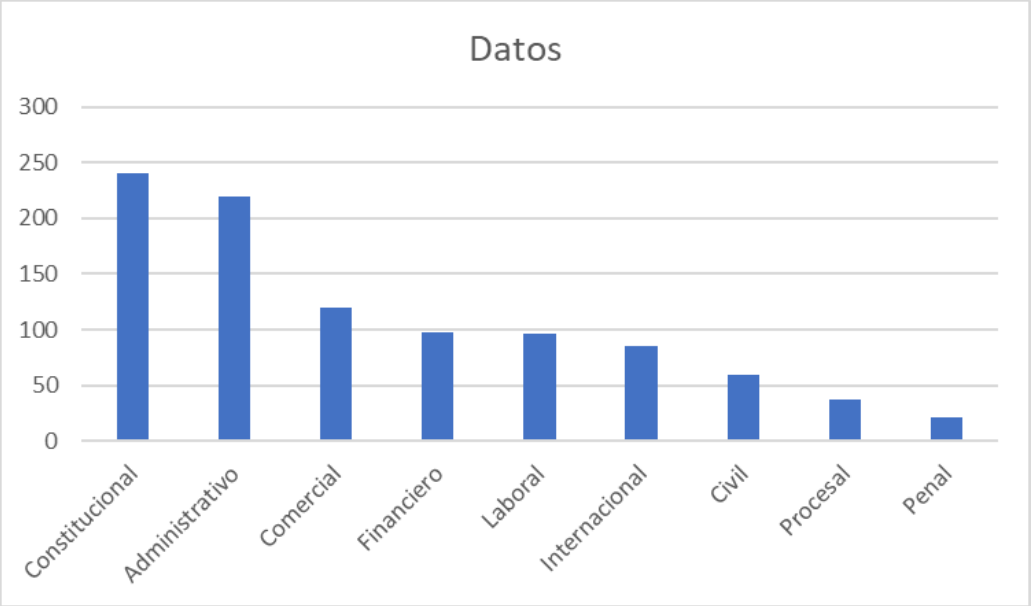

Figura 26 - Balance del conjunto de datos con 9 ramas.

En la siguiente imagen (Figura 27) se puede observar que se logra una media de precisión del 88%, lo cual es un buen resultado.

```
Mean accuracy (Precisión media): 88,1% Std. deviation: +-0.072

                precision    recall  f1-score   support

administrativo       0.84      0.96      0.90        28
         civil       0.25      0.25      0.25         4
     comercial       0.67      0.77      0.71        13
constitucional       0.93      0.93      0.93        28
    financiero       1.00      0.83      0.91         6
 internacional       1.00      0.86      0.92         7
       laboral       0.83      0.83      0.83         6
         penal       0.00      0.00      0.00         1
      procesal       1.00      0.25      0.40         4

   accuracy                           0.84        97
   macro avg       0.72      0.63      0.65        97
weighted avg       0.84      0.84      0.82        97

12 errores de 101 muestras de validación
```

Figura 27 - Resultados con dataset reducido de 8 ramas.

Balanceando el conjunto de datos

Como se puede observar en la Figura 27, existe una desproporción elevada en la cantidad de ejemplos suministrados en las clases mayoritarias y minoritarias. Por ejemplo, la rama Constitucional posee más de siete veces la cantidad de ejemplos que la rama Penal. Es probable que la técnica K Fold (vease sección “5.2.3.2 Entrenamiento”) aplicada al set de datos no sea suficiente para suplir tal diferencia.

Por ejemplo, si se cuenta con un conjunto de datos desequilibrado que contiene el 1% de una clase minoritaria y el 99% de la clase mayoritaria, un algoritmo puede predecir todos los casos como pertenecientes a la clase mayoritaria. La puntuación de precisión de este algoritmo arrojará una precisión del 99%, lo que parece impresionante, pero ¿es realmente así? La clase minoritaria es totalmente ignorada en este caso y esto puede resultar costoso en algunos problemas de clasificación.

Con el fin de solucionar el desbalance en el conjunto de datos y los posibles problemas que esto acarrea se implementaron las estrategias presentadas a continuación.

- Inserción de nuevos ejemplos de las ramas Penal y Procesal

  El primer paso para revertir la situación es recopilar una mayor cantidad de datos de las clases minoritarias. Es decir se debe verificar si es posible reunir más datos para el problema, ya que un conjunto de datos más grande podría exponer una perspectiva diferente y quizás más equilibrada de las clases.
  Luego de un gran esfuerzo, se logró reunir 19 nuevos ejemplos de la rama Penal y 10 ejemplos de Procesal. El set de datos resultante queda representado en la siguiente imagen (Figura 28).

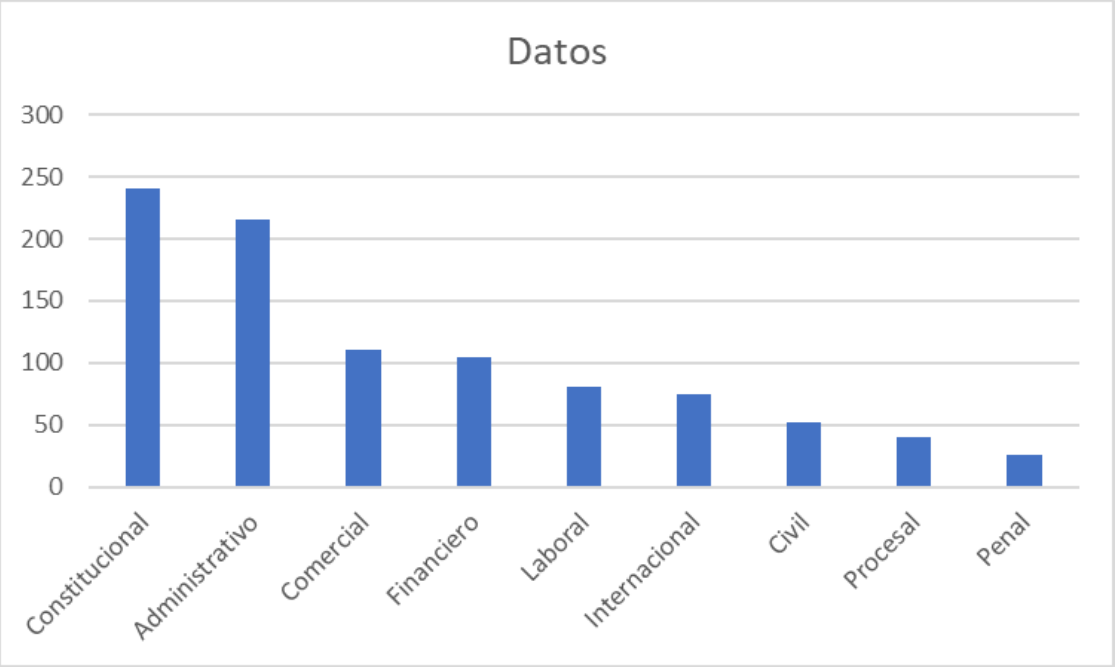

Figura 28 - Balance del conjunto de datos luego de insertar datos.

```
Mean accuracy (Precisión media): 87,7% Std. deviation: +-0.13

                precision    recall  f1-score   support

administrativo       0.91      0.91      0.91        22
         civil       0.40      0.50      0.44         4
     comercial       0.69      0.82      0.75        11
constitucional       0.89      0.89      0.89        28
    financiero       1.00      0.87      0.93        15
 internacional       1.00      1.00      1.00         5
       laboral       0.83      0.83      0.83         7
         penal       0.00      0.00      0.00         2
      procesal       1.00      0.25      0.40         6

   accuracy                           0.84       100
   macro avg       0.72      0.63      0.65       100
weighted avg       0.84      0.84      0.82       100

16 errores de 100 muestras de validación
```

Figura 29 - Resultados luego de insertar datos.

```
Error 0 -> Se predijo comercial, valor real constitucional
Error 1 -> Se predijo constitucional, valor real comercial
Error 2 -> Se predijo procesal, valor real constitucional
Error 3 -> Se predijo financiero, valor real procesal
Error 4 -> Se predijo internacional, valor real procesal
Error 5 -> Se predijo financiero, valor real procesal
Error 6 -> Se predijo laboral, valor real procesal
Error 7 -> Se predijo administrativo, valor real comercial
Error 8 -> Se predijo financiero, valor real procesal
Error 9 -> Se predijo penal, valor real procesal
Error 10 -> Se predijo administrativo, valor real procesal
Error 11 -> Se predijo administrativo, valor real procesal
Error 12 -> Se predijo financiero, valor real procesal
Error 13 -> Se predijo civil, valor real laboral
Error 14 -> Se predijo laboral, valor real constitucional
Error 15 -> Se predijo constitucional, valor real procesal
```

Figura 30 - Errores luego de insertar datos.

Como se puede observar en la Figura 29, los “buenos” resultados se mantienen. Pero es posible que todavía existan resultados sesgados debido a la desproporción todavía existente. Pero, un buen indicio de que se ha balanceado el conjunto de datos es que en los errores obtenidos (Figura 30) la diferencia entre las predicciones y los valores reales ya no predomina una sola clase como en los casos anteriores, donde predominaba Administrativo.

- <u>Eliminación de datos de Administrativo y Constitucional:</u>

La diferencia aún seguía siendo elevada entre las clases mayoritarias y minoritarias, por ejemplo Constitucional posee casi cinco veces más ejemplos que Civil. Es por esto que se decidió reducir significativamente las muestras de las ramas Administrativo y Constitucional, quedando representado el dataset en la siguiente imagen (Figura 31).

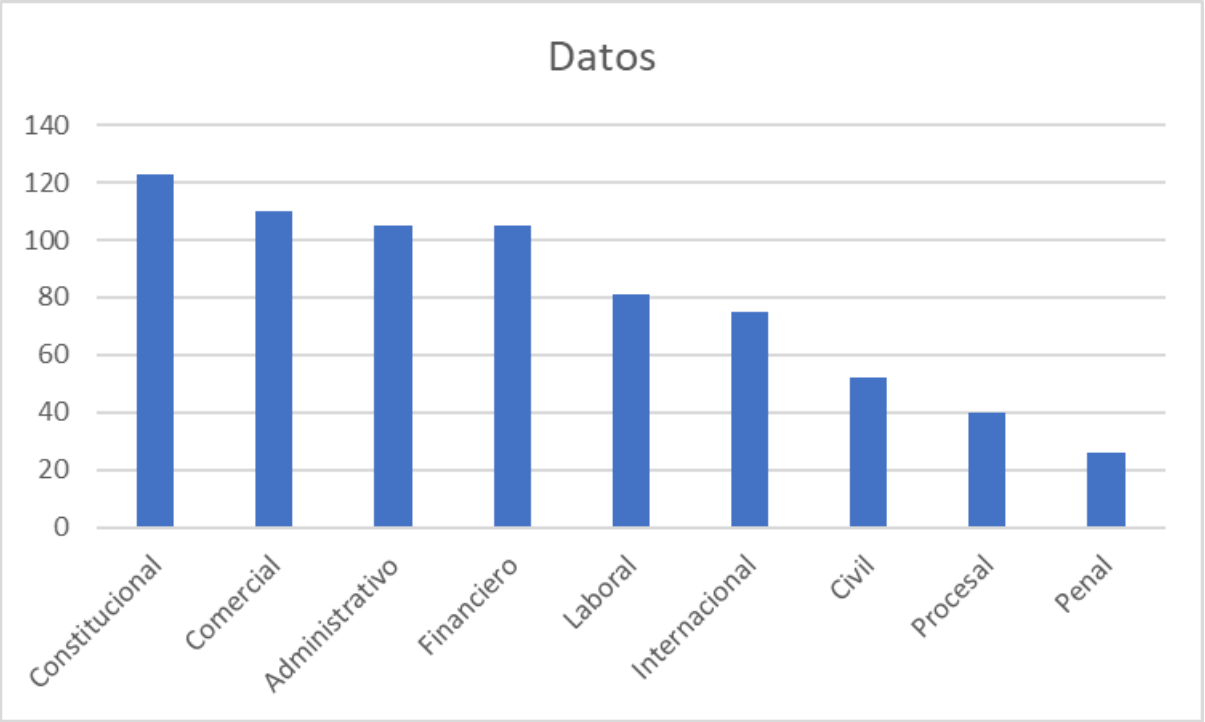

Figura 31 - Balance del conjunto de datos luego de quitar datos.

```
Mean accuracy (Precisión media): 85,9% Std. deviation: +-0.139

                precision    recall  f1-score   support

administrativo       0.80      0.80      0.80         5
         civil       0.67      0.67      0.67         6
     comercial       0.69      0.90      0.78        10
constitucional       0.91      0.83      0.87        12
    financiero       0.88      0.78      0.82         9
 internacional       0.82      0.90      0.86        10
       laboral       0.80      1.00      0.89        12
         penal       1.00      0.60      0.75        25
      procesal       0.80      0.55      0.62         6

   accuracy                           0.81        77
   macro avg       0.82      0.78      0.78        77
weighted avg       0.82      0.81      0.80        77

15 errores de 77 muestras de validación
```

Figura 32 - Errores luego de insertar datos.

Si bien estos resultados (Figura 32) arrojan una Mean Accuracy del 85% (2% menor que en el caso anterior), los mismos siguen siendo "buenos". Pero si se observa la conformación del conjunto de datos (Figura 31), se puede notar que la desproporción aún es elevada por lo que es muy apresurado decir que ya se han descartado los posibles problemas que un dataset desbalanceado podrían estar introduciendo.

- Inserción de más datos de Penal y Procesal

Luego de mucho esfuerzo de investigación, se obtuvieron e insertaron al set de datos nuevos ejemplos de Penal y Procesal. Ahora, la cantidad de ejemplos en la clase mayoritaria no supera en un 60% a la clase minoritaria. En la siguiente imagen (Figura 33) se puede observar una representación del dataset.

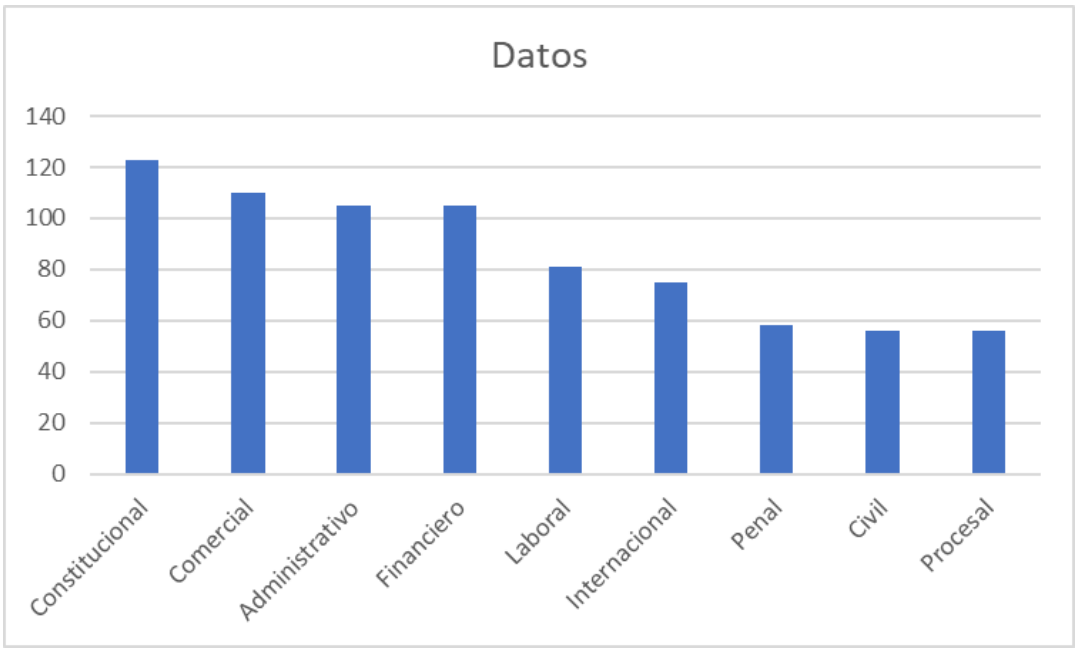

Figura 33 - Balance del conjunto de datos luego de insertar datos.

Mean accuracy (Precisión media): 87,0% Std. deviation: +-0.168

	precision	recall	f1-score	support
administrativo	0.73	0.67	0.70	12
civil	0.60	0.75	0.67	4
comercial	0.92	0.73	0.81	15
constitucional	0.78	0.88	0.82	8
financiero	0.87	1.00	0.93	13
internacional	1.00	0.91	0.95	11
laboral	0.90	1.00	0.95	9
penal	0.50	0.67	0.57	3
procesal	0.75	0.60	0.67	5
accuracy			0.82	80
macro avg	0.78	0.80	0.79	80
weighted avg	0.83	0.82	0.82	80

Figura 34 - Resultados luego de insertar datos.

```
14 mistakes from 80 validation samples:
Error 0 -> financiero
Error 1 -> comercial
Error 2 -> laboral
Error 3 -> comercial
Error 4 -> internacional
Error 5 -> financiero
Error 6 -> financiero
Error 7 -> financiero
Error 8 -> financiero
Error 9 -> penal
Error 10 -> penal
Error 11 -> comercial
Error 12 -> comercial
Error 13 -> financiero
```

Figura 34 - Errores luego de insertar datos.

Como se puede observar, los resultados se mantienen aún habiendo balanceado el set de datos. Con esto, finalmente, se puede descartar la teoría de que un Mean Accuracy sesgado debido al alto desbalance antes presente.

Optimización de hiperparámetros

Luego de balancear el set de datos, se procede a hacer uso de GridSearchCV de Sklearn (Sección 2.5), para dar con la mejor combinación de parámetros posibles con el fin de mejorar el Accuracy.

Los parámetros evaluados son C, solver y max_iter. A continuación (Figura 35) se observan los resultados arrojados por dicha optimización.

```
{C=25, solver="saga", max_iter=2000}
Mean accuracy (Precisión media): 87,41%
                precision    recall  f1-score   support

administrativo       0.97      0.97      0.97        31
         civil       0.71      0.75      0.73        16
     comercial       0.95      0.89      0.92        46
constitucional       0.99      0.97      0.99        35
    financiero       0.97      0.97      0.97        37
 internacional       0.85      0.92      0.88        25
       laboral       0.82      0.90      0.86        20
         penal       0.80      0.89      0.84         9
      procesal       0.75      0.60      0.67        10

   accuracy                              0.97       229
   macro avg       0.67      0.78      0.97       229
weighted avg       0.93      0.91      0.92       229
```

Figura 35 - Resultados de optimización de parámetros.

La optimización de hiperparámetros indica que con el conjunto de datos se puede alcanzar una Accuracy del 87,41% y un total de 24 errores. Esto se alcanza con los parámetros C=25, solver='saga', max_iter=2000.

Se genera entonces un mapa de calor (Figura 36) en forma de matriz para observar los resultados de entrenamiento en mayor detalle. El objetivo es mostrar las discrepancias entre las etiquetas previstas y las reales. En el eje vertical se observan los valores reales de los datos de test, mientras que en el eje horizontal se observan las predicciones realizadas por el modelo.

La gran mayoría de las predicciones terminan en la diagonal principal, donde la etiqueta predicha es igual a la etiqueta real (el cual es el objetivo). Sin embargo, hay una serie de clasificaciones erróneas (las que se encuentran fuera de la diagonal principal).

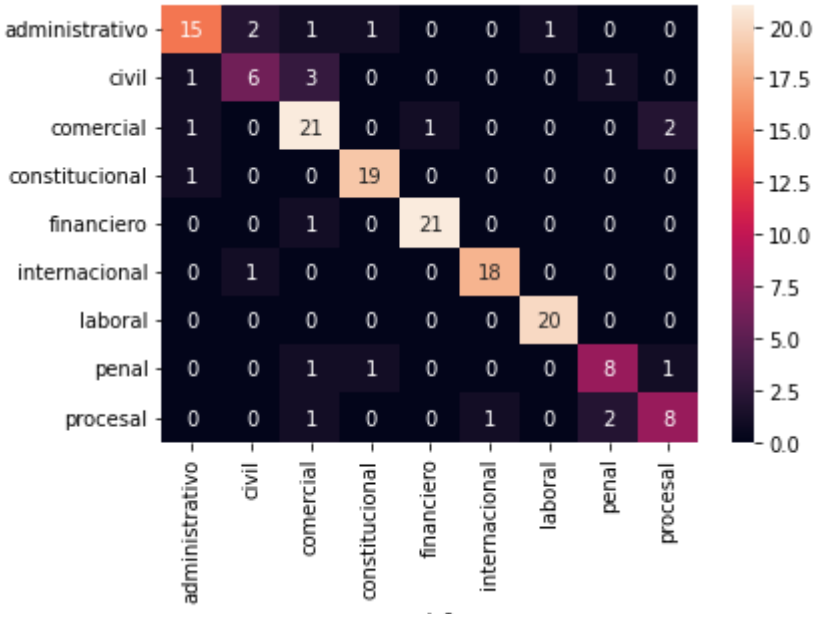

Figura 36 - Mapa de calor.

Siguientes pasos

Como se menciona al comienzo de esta sección, las primeras pruebas se realizaron con regresión logística y, una vez afianzado con el dataset y métricas, se continuó con las pruebas sobre los demás modelos. En las siguientes secciones se presentan las implementaciones y resultados de aplicar los restantes modelos de clasificación.

5.2.3.3 Pruebas y análisis de KNN

El siguiente paso fue utilizar un modelo de KNN. Cuyo hiparámetro *n_neighbors* fue seteado en 1.

Mean accuracy (Precisión media): 88,3%

	precision	recall	f1-score	support
administrativo	0.97	0.97	0.97	31
civil	0.71	0.75	0.73	16
comercial	0.95	0.89	0.92	46
constitucional	0.99	0.97	0.99	35
financiero	0.97	0.97	0.97	37
internacional	0.85	0.92	0.88	25
laboral	0.82	0.90	0.86	20
penal	0.80	0.89	0.84	9
procesal	0.75	0.60	0.67	10
accuracy				229
macro avg	0.67	0.78	0.87	229
weighted avg	0.93	0.91	0.92	229

Figura 37 - Resultados de KNN.

Los primeros resultados son buenos. Las muestras arrojan un Accuracy del 88%. Se procede a optimizar los hiperparámetros para observar si es posible lograr una mejoría en el modelo.

Optimización de hiperparámetros

Primero se crea una instancia de clasificador KNN y luego se prepara un rango de valores del hiperparámetro *n_neighbors* del 1 al 31 que es utilizado por GridSearchCV (Sección 2.5) para encontrar el mejor valor de K posible.

Luego de realizar la optimización, las salidas arrojaron que el mejor valor posible para el hiperparámetro es *n_neighbors*=3. Los resultados se presentan a continuación.

Mean accuracy (Precisión media): 91,7%

	precision	recall	f1-score	support
administrativo	0.97	0.94	0.95	32
civil	0.78	0.88	0.82	16
comercial	0.95	0.82	0.88	44
constitucional	0.98	1.00	0.99	44
financiero	0.82	0.90	0.86	20
internacional	0.88	1.00	0.94	23
laboral	0.93	0.93	0.93	29
penal	0.91	0.91	0.91	11
procesal	0.89	0.80	0.84	10
accuracy			0.92	229
macro avg	0.90	0.91	0.90	229
weighted avg	0.92	0.92	0.92	229

Figura 36 - Resultados de optimización de parámetros en KNN.

Las salidas (Figura 36) indican que es posible alcanzar un 91,7% de Accuracy y un total de 19 errores para el dataset con esta configuración de parámetro.

Si se genera un mapa de calor (Figura 38) se observa que, nuevamente, la mayor parte de predicciones caen dentro de la matriz principal. Pero en este caso, la cantidad de predicciones erróneas (es decir que no se encuentren en la matriz principal), es menor si se comparan con el mapa de calor obtenido con Regresión Logística.

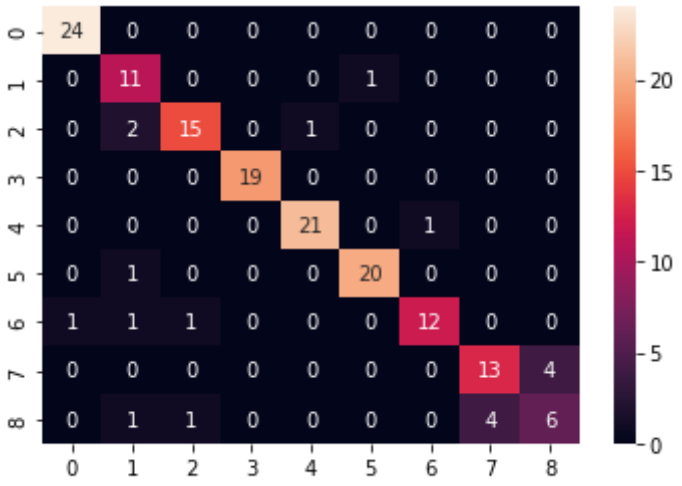

Figura 38 - Mapa de calor.

Un dato importante a tener en cuenta es que este algoritmo se ejecuta en un tiempo mucho menor al de regresión logística (solo le toma entre 1 y 2 segundos). Además:

- KNN es simple y fácil de implementar
- La fase de entrenamiento es más rápida debido al aprendizaje perezoso (lazy learning).
- Adecuado para problemas de varias clases.
- Funciona mejor para datos que cambian continuamente debido al aprendizaje basado en instancias.

Por otro lado, entre las desventajas de este algoritmo podemos encontrar:

- La fase de prueba es más lenta y costosa en términos de tiempo y memoria.
- Muy sensible a los valores atípicos.
- Curse of dimensionality, es decir, si el número de dimensiones (características de clase) aumenta, el número requerido de datos debe aumentar exponencialmente. En este caso, esta desventaja del modelo no afecta a los objetivos de este PFC ya que solo existe una característica a evaluar por cada clase, que es el texto de cada ley.

5.2.3.3 Pruebas y análisis de SVM

El siguiente paso es llevar a cabo la implementación del algoritmo SVM, cuyos hiperparámetros inicialmente son seteados de la forma C=1.0 y gamma='auto'. En la siguiente imagen se visualizan los resultados del entrenamiento del modelo.

```
Mean accuracy (Precisión media): 79,03%
                precision    recall  f1-score   support

administrativo       0.72      0.82      0.77        34
         civil       0.57      0.50      0.53        16
     comercial       0.67      0.80      0.73        35
constitucional       0.92      0.67      0.77        36
    financiero       0.81      0.97      0.89        36
 internacional       0.86      0.86      0.86        22
       laboral       0.93      0.89      0.91        28
         penal       1.00      0.60      0.75        10
      procesal       0.80      0.67      0.73        12

   accuracy                           0.79       229
   macro avg       0.82      0.75      0.77       229
weighted avg       0.80      0.79      0.79       229

48 errores de 229 muestras de validación
```

Figura 39 -Resultados de SVM.

Como se observa en la Figura 39, de los modelos evaluados hasta el momento, SVM es el que arrojó resultados más bajos, ya que el Mean Accuracy es de un 79,03%. Antes de descartar este algoritmo, se realizó una optimización de hiperparámetros para observar si era posible mejorar el modelo.

Optimización de hiperparámetros

Primero se creó una instancia de clasificador SVM y se prepararon los siguientes valores del hiperparámetro C = [0.001, 0.01, 0.1, 1, 10] y gamma = [0.001, 0.01, 0.1, 1]. Estos valores son utilizados por GridSearchCV (Sección 2.5) para encontrar los mejores hipermarámetros posibles para el modelo. Los resultados de optimización se observan en la siguiente imagen.

```
{C=10, gamma=0.1}

Mean accuracy (Precisión media) para nuestro dataset con optimización de parámetros: 79,03%
```

Figura 40 - Optimización de parámetros.

Como se puede observar en la Figura 39, los mejores valores posibles del rango seleccionado son C = 10 y gamma = 0.1, con estos valores se obtiene un Accuracy del 80,24% y un total de 33 errores. Si bien se obtiene una mejoría de aproximadamente un 1%, este modelo sigue teniendo un Accuracy considerablemente menor a los obtenidos con los modelos previamente analizados, con lo cual queda descartado de la lista de opciones.

En este caso si se genera un mapa de calor (Figura 41) para evaluar los resultados en mayor detalle, se puede observar que las predicciones fuera de la matriz principal, es decir erróneas, es mayor que en los mapas de calor de Regresión Lineal y KNN.

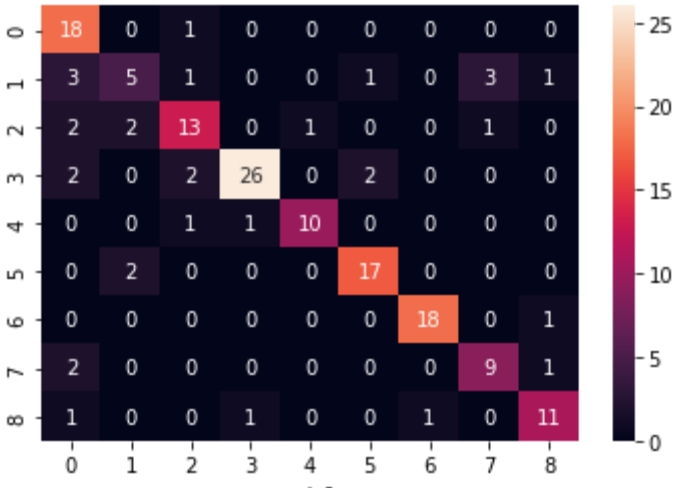

Figura 41 - Mapa de calor.

5.2.3.3 Elección final del modelo y configuración de parámetros.

A continuación se presenta un resumen de los diferentes resultados obtenidos luego de entrenar cada algoritmo.

Algoritmo	Mean Accuracy	Complejidad de algoritmo	Requerimientos computacionales	Tiempo de entrenamiento
Regresión Lineal	87,41%	Baja	Medio	90 segundos
KNN	91,7%	Baja	Bajo para dataset pequeños	2 segundos
SVM	79,03%	Baja	Medio	10 segundos

Tabla 5 - Configuraciones de parámetros.

Como se menciona en las secciones 5.2.3.2, el algoritmos SVM queda descartados de la lista de opciones a considerar.

Se debe decidir entonces si aplicar la Regresión Lineal o bien KNN. Para la elección final se evalúan los costos-beneficios de cada algoritmo, es decir, se pone en balanza el Mean Accuracy de cada uno y también el costo ligado a comprender y ejecutar los mismos.

Como se puede observar en la Tabla 5, tanto para KNN como para Regresión Lineal, los costos de ambos algoritmos son similares, excepto por el hecho que los requerimientos para Regresión Lineal es un tanto mayor. Es por esto que la elección final esta basada en el Mean Accuracy y, analizando el resumen de resultados, KNN es quien arroja mejores valores. Además, si se considera también el tiempo de entrenamiento requerido para cada uno de ellos, KNN es el ganador por una gran

diferencia. Se puede concluir entonces que el algoritmo que mejor se adapta al set de datos es KNN.

A continuación se mencionan los valores seleccionados para cada uno de los posibles parámetros, acompañado de una justificación para dicha elección.

- *test_size = 0.1*
  Este parámetro representa el porcentaje del dataset que será destinado a testear el modelo. Un valor seteado en 0.1 representa al 10% del total de los datos, es decir la distribución es 9:1.
  Luego de varias pruebas para este parámetro, se observó que variando el valor del mismo, los resultados arrojaron un menor valor de Mean Accuracy.

- *random_state = 12*
  Este parámetro asegura que la distribución 9:1 escogida en las muestras se mantenga siempre idéntica, lo cual nos permite saber si el modelo mejora al ajustar sus parámetros, o si realmente es una cuestión de azar.
  Luego de varias pruebas para este parámetro, se observó que variando el valor del mismo, los resultados arrojan un menor valor de Mean Accuracy.

- *max_iter = 2000*
  Para este caso, se eligió un número alto (inicialmente 4000). Se observó que, más allá de un número determinado (2000) el valor Mean Accuracy disminuye. Por lo cual se puede concluir que el modelo es sobreentrenado, perdiendo así efectividad.

- *n_neighbors* = 3
  Este parámetro fue seteado con este valor luego de realizar la optimización de parámetros.

5.3 Diseño y desarrollo del prototipo de software

En esta sección se desarrolla el proceso de definición de la arquitectura, componentes y otras características del prototipo de sistema desarrollado.

5.3.1 Requerimientos del prototipo de software para la clasificación de textos legales en las diferentes ramas del derecho

El primer paso en el diseño es definir los requerimientos del sistema. En este caso, el objetivo es prototipar una herramienta de software que permita entrenar un modelo KNN ingresando datos de entrenamiento para luego realizar predicciones sobre leyes presentes en los Boletines Oficiales de la República Argentina (BORAs) los cuales se encuentran en formato PDF.

Una vez establecido el objetivo general, se pueden desprender los siguientes requerimientos.

- Entrenar modelo y visualizar los resultados de entrenamiento.
- El sistema debe permitir extraer los textos de los archivos PDF y almacenarlos para una posterior clasificación.
- Clasificar textos extraídos previamente.
- Crear nuevos datasets.

5.3.2 Diagrama de clases de la herramienta

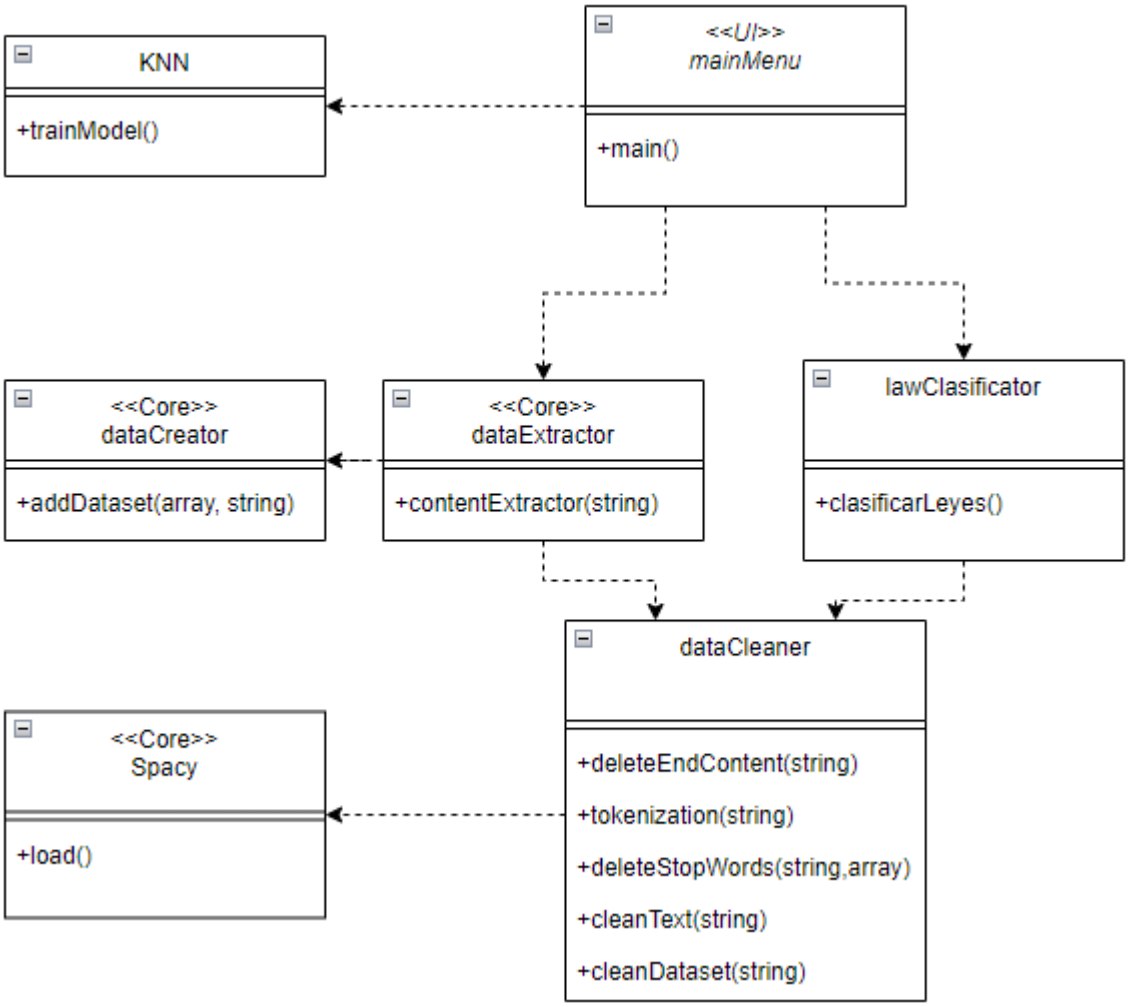

Figura 41 - Diagrama de clases.

5.3.3 Código fuente

El código fuente de la herramienta final se encuentra disponible en el siguiente repositorio online.

https://github.com/jigerarduzzi/proyecto

Los pasos de instalación del prototipo se encuentran detallados en el archivo README del repositorio.

6. Conclusiones

Son varias las conclusiones que se pueden hacer sobre este PFC, se puede comenzar destacando la satisfacción de sentir que los conocimientos adquiridos a los largo de estos años en la universidad no fueron en vano y que han servido como una herramienta para obtener resultados, solucionar problemas emergentes, rediseñar procedimientos y todo lo que he necesitado a lo largo de este proceso para lograr cumplir con la entrega de esta tesis.

Como era de esperar, concluir este PFC resultó todo un desafío dado que implicó adentrarse en nuevas tecnologías y conceptos en los cuales el autor no estaba familiarizado.

Realizar este proyecto sin lugar a dudas permitió al autor adquirir nuevos conocimientos técnicos y habilidades que asisten en la resolución de problemas. Es muy interesante ya que éstos pueden ser aplicados en un futuro, tanto en mi vida profesional como cotidiana.

En cuanto a los plazos establecidos en el Proyecto Final de Carrera se concluye que no se cumplió con los mismos. Esto fue derivado a que inicialmente la planificación contemplaba a dos integrantes activos en el proceso pero finalmente solo fue realizado por uno de ellos. Sumado a esto hubo diferentes inconvenientes personales que no se contemplaron en los riegos iniciales que también introdujeron ciertas demoras.

En cuanto a los objetivos planteados se puede concluir que se han cumplido de manera satisfactoria, ya que se analizaron los diferentes algoritmos de Aprendizaje Automático utilizados con mayor frecuencia en la clasificación de textos. Se pudo estudiar que existe una gran variedad de técnicas de aprendizaje profundo que si se aplican y ajustan de manera correcta pueden tener resultados satisfactorios. En el contexto de este PFC se evaluaron diferentes técnicas y se aplicaron las más adecuadas teniendo en cuenta las características del conjunto de datos .

En lo que al dataset refiere, si bien se obtuvo, preprocesó y etiquetó el dataset de manera satisfactoria, esta sin dudas fue la etapa más demandante de todo el proceso. Esto es así porque en esta instancia el autor se encontró con documentos que contenían textos que estaban fuera de su conocimiento, lo cual le imposibilitaba etiquetar de manera correcta los datos de entrenamiento. Es por esto que se recurrió a profesionales del área de abogacía para que lo orientaron en el proceso.

También se definieron los distintos criterios de evaluación de los modelos de clasificación, lo cual ayudó a discernir de los diferentes resultados y modelos. Esto también fue un gran apoyo al momento de identificar situaciones en las cuales se obtenían resultados que a priori parecían buenos pero en realidad estaban sesgados debido a un desbalance en los datos.

En cuanto a la etapa entrenamiento, recopilación y análisis de diferentes algoritmos, permitió interactuar y experimentar con los distintos modelos. Resultó sumamente interesante realizar pruebas ajustando los distintos hiperparámetros y realizar distintos entrenamientos porque esto me permitió lograr diferentes resultados. También dejó como enseñanza el no sobredimensionar ciertas estructuras ya que deriven en un excesivo poder de cómputo. Aquí es donde fue de mucha utilidad uso de GridSearchCV de Sklearn para definir un correcto ajuste de hiperparámetros.

Por último luego de analizar los diferentes modelos entrenados se desarrolló, aunque limitada, una interfase de prototipo la cual permite visualizar los resultados del modelo obtenido.

7 Trabajos futuros

En el transcurso del desarrollo de este Proyecto Final de Carrera, surgieron diferentes posibilidades e ideas para trabajos futuros. El principal trabajo futuro comprende un análisis a fondo de la herramienta Transformer de BERT, la cual parece un modelo prometedor y con mucho potencial a explotar pero que debido a su complejidad y a los excesos de tiempos de implementación, fue descartada de la lista de algoritmos a probar.

Por otra parte, las siguientes versiones deben comprender la implementación de interfaces de usuarios que agreguen un valor agregado a los mismos, mejorando la experiencia de uso, ya sea para el entrenamiento o visualización de datos y resultados.

También se prevé expandir el conjunto inicial de leyes presentes en el dataset de entrenamiento, esto sin dudas derivaría en un aumento en la efectividad de la herramienta.

Por último, dentro del aspecto arquitectónico una posibilidad es convertir la implementación actual, la cual se ejecuta de manera local, a una de tipo cliente-servidor. En esta se montaría el procesamiento de textos en un servidor expuesto en la nube a través de una API (Interfaz de programación de aplicaciones).

Esto posibilitará acceder a la herramienta vía web, lo cual abre un nuevo conjunto de posibilidades sobre la utilización de la misma.

Anexo A - Instrucciones de instalación y ejecución de prototipos.

Referencias bibliográficas

[1] Alexander Gelbukh. (2010, Junio 10). *Procesamiento del lenguaje natural y sus aplicaciones*.

[2] *Apache OpenNLP*. (22). Apache OpenNLP. Retrieved August 22, 2022, from https://opennlp.apache.org/

[3] *Árbol de decisión*. (2022). Wikipedia. Retrieved August 22, 2022, from https://es.wikipedia.org/wiki/%C3%81rbol_de_decisi%C3%B3n

[4] *Attention is All You Need*. (2017). NIPS papers. Retrieved August 22, 2022, from https://proceedings.neurips.cc/paper/2017/file/3f5ee243547dee91fbd053c1c4a845aa-Paper.pdf

[5] Bansal, N., Sharma, A., & Singh, R. K. (2019, May). A review on the application of deep learning in the legal domain. In *IFIP International Conference on Artificial Intelligence Applications and Innovations* (pp. 374-381). Springer, Cham.

[6] Bharath. (2022). *Transformers For Text Classification*. Paperspace Blog. Retrieved August 22, 2022, from https://blog.paperspace.com/transformers-text-classification/

[7] Cardellino, F., Cardellino, C., Haag, K., Soto, A., Teruel, M., Alonso i Alemany, L., & Villata, S. (2018). Mejora del acceso a Infoleg mediante técnicas de procesamiento automático del lenguaje. In *XVIII Simposio Argentino de Informática y Derecho (SID)-JAIIO 47 (CABA, 2018)*.

[8] Charniak, E. (2019). Introduction to deep learning. Cambridge, Massachusetts ; London,
CVC. Diccionario de términos clave de ELE. Aprendizaje deductivo. (n.d.). Centro Virtual

[9] Cervantes. Retrieved August 22, 2022, from https://cvc.cervantes.es/ensenanza/biblioteca_ele/diccio_ele/diccionario/aprendizajedeductivo.htm

[10] De Araujo, P. H. L., de Campos, T. E., Braz, F. A., & da Silva, N. C. (2020, May). VICTOR: a dataset for Brazilian legal documents classification. In *Proceedings of The 12th Language Resources and Evaluation Conference* (pp. 1449-1458).

[11] *Derecho argentino*. (n.d.). Wikipedia. Retrieved August 22, 2022, from https://es.wikipedia.org/wiki/Derecho_argentino

[12] *Digesto Jurídico Argentino*. (n.d.). SAIJ. Retrieved August 22, 2022, from http://www.saij.gob.ar/digesto-juridico-argentino

[13] *Dropping common terms: stop words*. (2009). Stanford NLP Group. Retrieved August 22, 2022, from https://nlp.stanford.edu/IR-book/html/htmledition/dropping-common-terms-stop-words-1.htm

[14] Google Drive. (n.d.). Google Drive. https://drive.google.com/

[15] *Guide to confusion matrix terminology*. (2014, March 25). Data School. Retrieved August 22, 2022, from https://www.dataschool.io/simple-guide-to-confusion-matrix-terminology/

[16] Hacia una metodología de gestión de conocimiento basada en la minería de datos - Claudia L. Hernández, María Ximena Dueñas.

[17] Khan, U. I. (2022, Enero 3). *Lemmatization In Natural Language Processing — NLP*. Towards Dev. Retrieved August 22, 2022, from https://towardsdev.com/lemmatization-in-natural-language-processing-nlp-5d2434527766

[18] *K vecinos más próximos*. (2022). Wikipedia. Retrieved August 22, 2022, from https://es.wikipedia.org/wiki/K_vecinos_m%C3%A1s_pr%C3%B3ximos

[18] Lippi, M., Pałka, P., Contissa, G., Lagioia, F., Micklitz, H. W., Sartor, G., & Torroni, P. (2019). CLAUDETTE: an automated detector of potentially unfair clauses in online terms of service. *Artificial Intelligence and Law*, *27*(2), 117-139.

[19] *Máquinas de vectores de soporte*. (2022). Wikipedia. Retrieved August 22, 2022, from https://es.wikipedia.org/wiki/M%C3%A1quinas_de_vectores_de_soporte

[20] Minsky, M. (1968). Pionero de la Inteligencia Artificial

[21] *MonkeyLearn - Text Analytics*. (2022). MonkeyLearn - Text Analytics. Retrieved August 22, 2022, from https://monkeylearn.com/

[22] Mukesh Zaver. (2011, Agosto 2). *Automatic Text Classification*.

[23] Natural Language Toolkit — NLTK 3.5 documentation. (n.d.). NLTK. https://www.nltk.org

[24] Neil, M., Fenton, N., Lagnado, D., & Gill, R. D. (2019). Modeling competing legal arguments using Bayesian model comparison and averaging. *Artificial intelligence and law*, *27*(4), 403-430.

[25] *NLTK - Natural Language Toolkit*. (22, Marzo). NLTK :: Natural Language Toolkit. Retrieved August 22, 2022, from https://www.nltk.org/

[26] Perezzini, L., Casali, A., & Deco, C. (2020). Sistema de soporte para la recuperación de normativas en la ingeniería legal. In *XX Simposio Argentino de Informática y Derecho (SID 2020)-JAIIO 49 (Modalidad virtual)*.

[27] Portal de Datos Justicia Argentina - Tesauro SAIJ de Derecho Argentino
Procesos de Ingeniería de Software - ECC Posibles Riesgos del SW - INGENIERÍA DEL SOFTWARE , Ian Sommerville

[28] Project Jupyter. (n.d.). Home. https://jupyter.org/

[29] *Regresión logística*. (2022). Wikipedia. Retrieved August 22, 2022, from https://es.wikipedia.org/wiki/Regresi%C3%B3n_log%C3%ADstica

[30] School of Computing, Asia Pacific University of Innovation and Technology. (2019, Diciembre). *Loan Default Prediction Model Using Sample, Explore, Modify, Model, and Assess (SEMMA)*. Retrieved August 22, 2022, from https://www.researchgate.net/profile/Uzair-Aslam-4/publication/335966813_Loan_Default_Prediction_Model_Using_Sample_Explore_Modify_Model_and_Assess_SEMMA/links/5d8661a7458515cbd1af3999/Loan-Default-Prediction-Model-Using-Sample-Explore-Modify-Model-and-Ass

[31] *Scorer - spaCy API Documentation*. (2022). spaCy. Retrieved August 22, 2022, from https://spacy.io/api/scorer

[32] Shimazu, A., & Le Nguyen, M. (2014). Legal Engineering and Its Natural Language Processing. In *Knowledge and Systems Engineering* (pp. 7-7). Springer, Cham.

[33] *spaCy · Industrial-strength Natural Language Processing in Python*. (2022). spaCy · Industrial-strength Natural Language Processing in Python. Retrieved August 22, 2022, from https://spacy.io/

[34] SpaCy · Industrial-strength Natural Language Processing in Python. (n.d.). SpaCy. https://spacy.io/

[35] *Stemming and lemmatization*. (2009). Stanford NLP Group. Retrieved August 22, 2022, from https://nlp.stanford.edu/IR-book/html/htmledition/stemming-and-lemmatization-1.html

[36] *Term Frecuency — Inverse Document Frecuency*. (2019, February 13). USEO. Retrieved August 22, 2022, from https://useo.es/tf-idf-relevancia/

[37] *Tokenization*. (n.d.). Stanford NLP Group. Retrieved August 22, 2022, from https://nlp.stanford.edu/IR-book/html/htmledition/tokenization-1.html

[38] Training spaCy's Statistical Models · spaCy Usage Documentation. (n.d.-c). Training SpaCy's Statistical Models. https://spacy.io/usage/training

[39] *Utilización del aprendizaje inductivo en la toma de decisiones. Aplicación en un problema de secuenciación*. (n.d.). Dialnet. Retrieved August 22, 2022, from https://dialnet.unirioja.es/servlet/articulo?codigo=1096657

[40] Wikipedia. (2020, October 26). Procesamiento de lenguajes naturales. Wikipedia, La Enciclopedia Libre. https://es.wikipedia.org/wiki/Procesamiento_de_lenguajes_naturales

[41] Yamada, H., Teufel, S., & Tokunaga, T. (2019). Building a corpus of legal argumentation in Japanese judgment documents: towards structure-based summarisation. *Artificial Intelligence and Law*, *27*(2), 141-170.