Universidad Europea

UNIVERSIDAD EUROPEA DE MADRID

ESCUELA DE ARQUITECTURA, INGENIERÍA Y DISEÑO

MÁSTER UNIVERSITARIO EN

ANÁLISIS DE DATOS MASIVOS (BIG DATA)

TRABAJO FIN DE MÁSTER

DETECCIÓN DE PATRONES MACHISTAS EN NOTICIAS DE INTERNET

NOMBRE:

Daniel Riveros García

CURSO 2021-2022

TÍTULO: **Detección de patrones machistas en noticias de internet.**

AUTOR: Daniel Riveros García

TITULACIÓN: Máster Universitario en Análisis de Datos Masivos (Big Data)

DIRECTOR DEL PROYECTO: María Cruz Gaya López

FECHA: Octubre de 2022

RESUMEN

En la actualidad, si buscamos en internet la palabra “machismo” nos podemos encontrar con multitud de artículos que tratan el machismo y la importancia que están tomando las redes sociales a la hora de educar y manipular a los más jóvenes (Anónimo, 2013) (Ramírez, 2022).

La creciente tendencia de los algoritmos de machine learning junto al fenómeno anteriormente mencionado nos plantea una nueva forma de incorporar estas herramientas para la detección de micromachismos que, a priori podrían pasar inadvertidos ante nuestros propios ojos.

Con el uso de estas herramientas junto a un correcto procesado y posterior análisis de los textos podemos conseguir un pequeño programa capaz de indicarnos de forma resumida el grado de machismo que podemos encontrar en una noticia, en nuestro caso especial, obtenidas del periódico Independent.

La correcta interpretación de los datos nos puede dar lugar a una primera aproximación de una herramienta que podría ser vital para cualquier sitio de internet, evitando sesgos e inclusión de machismo en las diferentes webs publicadas, tomando especial interés en las áreas del periodismo.

Palabras clave: Machine Learning, Machismo, Sesgos machistas, Análisis de Sentimientos, Periodismo

---

ABSTRACT

At present, if we search on internet the word “machismo” we can find lot of articles related to sexism and the importance that have the social media for education and manipulate joung people (Anónimo, 2013) (Ramírez, 2022).

The increasing tendency of machine learning algorithm along with the previous freak propose us a new form to incorporate this tools in the detection of “micromachismos” that can be undetected for humans.

Using this tools along a correct text proccesing and analysis allows to a small program able to detect the sexism grade that have a news, in our special case, news from Independent newspaper.

The correct implementation of data can brin gus a first aproximation of a tool that can be vital for any website, avoiding sexism biases on the differentes published web, specially for journalism news.

Key words: Machine Learning, Sexism, Seximes Bias, Sentiment Analysis, Journalism

AGRADECIMIENTOS

A mis padres, que me han apoyado inconcionalmente durante toda mi vida.

Índice General

Índice de Figuras

Capítulo 1
Propuesta del Proyecto

En este primer capítulo vamos a desarrollar las diferentes motivaciones que nos han llevado a la realización de este proyecto, focalizándonos en los diferentes aspectos de la sociedad actual y a algunos de los problemas que se plantean diariamente en nuestras vidas.

Esta motivación la acompañaremos de un pequeño apartado donde explicaremos los diferentes requisitos que queremos cumplir con la realización de este TFM.

1.1 Motivación

Con el paso del tiempo, el papel que ha tenido la mujer en la sociedad ha ido evolucionando hasta lo que estamos viviendo en la actualidad, encontrando cada vez menos diferencias con los hombres y consiguiendo así una cultura más justa ante los diferentes ámbitos de la propia vida cotidiana, como el trabajo o las tareas domésticas.

Esta evolución ha supuesto un gran cambio en multitud de sectores de trabajos (donde antes solo podían trabajar hombres o mujeres), en el rol dentro de una familia (antes la mujer no podía trabajar por tener que ocuparse de la casa) e incluso dentro de culturas más extremistas (por ejemplo, la árabe).

La constante lucha por los derechos y la gran evolución en multitud de sectores ha conseguido que estas diferencias con los hombres se vean reducidas en gran medida y, aunque aún queda mucho trabajo por delante, podemos afirmar que esta "revolución" está siendo un éxito (Gamba, 2008).

Figura 1. *Imagen del feminismo*

No es difícil encontrar titulares con las palabras “Machismo” (Val, 2020) o “micromachismos” (Anónimo, Oxfam Intermón, s.f.); referencias a los temas de sesgos por raza o sexo en la IA (Schiebinger, 2018) e incluso reportajes donde se estudia el machismo dentro de la política (Martín, 2021), uno de los temas que más controversia ha tenido en los últimos años, sobre todo con la incorporación de partidos de extrema derecha.

Este tema causa una fuerte **controversia** en el mundo real, siendo un gran ejemplo de cara a realizar cambios útiles para la comunidad el realizar trabajos relacionados con el mismo, pues estos son bien recibidos por los usuarios que, a fin de cuentas, buscan mantener la igualdad a la hora de la elaboración de textos como pueden ser las noticias periodísticas o los boletines oficiales.

Todo este contexto nos plantea un escenario ideal para la utilización de los recursos aprendidos en el máster, pudiendo usar el conocimiento adquirido para acercarnos a una solución que pueda mejorar en gran medida la eliminación de sesgos machistas y que podamos dar un paso más hacia una sociedad más igualitaria.

Figura 2. *Representación Igualdad entre hombres y mujeres*

1.2 Requisitos a suplir por el TFM

Habiendo realizado una introducción a la motivación principal de nuestro trabajamos, es casi evidente intuir el problema que queremos afrontar y el cómo vamos a hacerlo. Especificando un poco más sobre el tema expuesto, nuestro principal requisito es la creación de un programa que nos ofrezca información útil sobre textos de la web, pudiendo realizar un análisis a partir de ellos.

Este primer requisito podemos dividirlo en requisitos de menor índole, los cuales podemos plantear de una manera muy temprana y que nos van a servir para introducir el siguiente apartado de la memoria:

- ¿Qué tipo de textos vamos a analizar? Este primer requisito lo centraremos a un único medio digital, del cual obtendremos la información necesaria para realizar nuestro procesamiento de datos.

- ¿Cómo se realizará la categorización de textos? Para el aprendizaje usaremos varias técnicas de *Machine Learning*, apoyándonos sobre todo en el análisis de sentimientos de textos, para los cuales usaremos palabras relacionadas con el machismo para obtener el puntuaje.

- ¿Cómo ayudará el programa al usuario? La idea principal sobre este punto es la creación de un panel de control que ayude a identificar los caracteres sensibles dentro de sus textos, asociando estos a un sistema de puntuación numérico fácil de entender por los usuarios.

Estas tres premisas nos sirven como una primera línea de acción hacía el resultado final, el cual (si se consiguen implementar con éxito los tres requisitos), nos brindará de una herramienta útil capaz de analizar de forma rápida un texto cualquiera de la web.

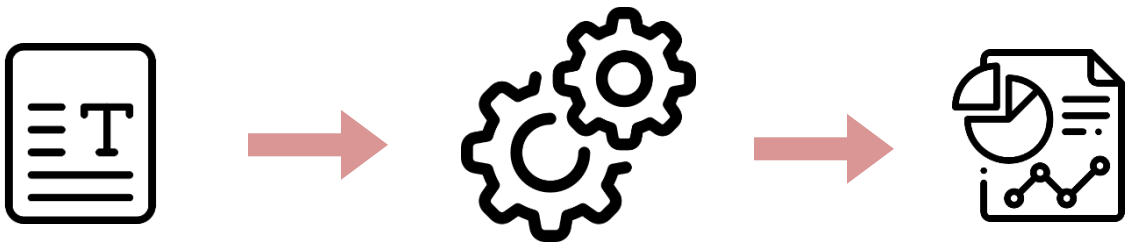


Figura 3. *Resumen en diagrama de nuestro problema*

Este proyecto podrá ser usado para la mejora de los textos actualmente publicados e incluso para un futuro, en una herramienta de obligatorio cumplimiento para la publicación de ciertos artículos, evitando sesgos e introducción involuntaria de términos machistas que puedan dar lugar a controversia.

Con esta aproximación y habiendo estudiado la motivación principal podemos dar paso a la siguiente sección de la memoria, donde indagaremos en mayor medida a los principios de nuestro proyecto.

Capítulo 2
Introducción

Durante este segundo capítulo definiremos la estructura que seguirá nuestro proyecto, ofreciendo una visión global de lo que vamos a ir exponiendo a lo largo de la memoria, consiguiendo que el lector se sitúe dentro de cada apartado.

Tras esto, haremos una pequeña introducción a las herramientas de *machine learning*, las cuales hemos visto a lo largo del máster y que serán nuestro pilar fundamental durante la realización del TFM. Junto a esto, realizaremos una pequeña introducción de los pilares secundarios que nos servirán de apoyo para la construcción completa de nuestro proyecto.

A continuación, y para finalizar, tomaremos como referencia los diferentes puntos expuestos en el apartado anterior, revisando los requisitos del capítulo 1.2 para dar lugar a los objetivos finales de nuestro proyecto, los cuales iremos resolviendo conforme vamos avanzando en la memoria, dando un carácter continuo a esta.

2.1 Estructura de la Memoria

Como primera parte de esta memoria vamos a crear una pequeña guía con las diferentes secciones a tratar a lo largo de la memoria junto a cada uno de los puntos clave que abordaremos en cada una de ellas, creando una estructura ordenada y fácilmente legible, ayudando a su interpretabilidad.

En esta sección excluiremos los apartados expuestos en la primera sección de la memoria (marcada con numeración románica), centrándonos en las secciones puramente teóricas.

I. Propuesta del proyecto:

La primera de estas ya ha sido expuesta al completo y comprende una de las partes más relevantes a este trabajo, pues, sin esta motivación, nunca hubiéramos llegado a realizar este pequeño gran proyecto el cual trata uno de los temas que más controversia han causado en los últimos años.

En esta misma hemos incluido un pequeño resumen de la primera idea naciente que se creó a partir del tema elegido para la realización del TFM, formando una pequeña lista de "requisitos" los cuales hemos conseguido tratar con la realización completa de este trabajo.

II. Introducción:

En este apartado realizaremos una introducción a las diferentes tecnologías a usar para el cumplimiento de lo expuesto en el punto 1.2, consiguiendo anclar las ideas sueltas de una idea inicial con la tecnología presente en la actualidad y la cual será necesaria para la obtención de todos los objetivos que posteriormente se introducirán en esta misma sección.

Estos objetivos vendrán acompañados del plan de trabajo junto a un pequeño resumen del estado del arte actual de problemas parecidos al nuestro, pudiendo encajar este proyecto dentro de la actualidad de los problemas de *machine learning* relacionados con el machismo.

III. Metodología de Trabajo y Experimentación:

Este apartado es el que más relevancia va a tomar a la hora de explicar los diferentes criterios tomados a la hora de realizar todo el TFM, exponiendo las diferentes decisiones que se han ido tomando a lo largo de la realización de este junto a los avances que se han ido consiguiendo con la utilización de las tecnologías previamente introducidas en apartados anteriores.

Durante esta sección indagaremos con más profundidad las diferentes tecnologías usadas para la realización del proyecto final, obteniendo un base experimental fuerte la cual puede ser usada como guía para diferentes proyectos de similares características.

Para finalizar y poner la guinda a esta tercera sección de nuestra memoria, indicaremos de forma resumida los criterios elegidos a la hora de elaborar una solución final a nuestro problema junto a la presentación de la interfaz final que dispondrá el usuario para acceder a nuestro trabajo.

IV. Conclusiones y Trabajo Futuro:

En esta sección se estudiarán con más detenimiento las limitaciones de nuestro proyecto junto con el recalco de los puntos más importantes de nuestro proyecto, indicando los puntos fuertes y débiles del mismo. Esto último irá acompañado de diferentes soluciones y posible trabajo futuro que se puede aplicar sobre esta base bien formada que hemos conseguido desarrollar con la realización de este trabajo de fin de máster.

V. Bibliografías y Referencias:

Para terminar y poner punto final al proyecto vamos a representar de forma ordenada las diferentes referencias tomadas para obtener la información relevante del proyecto junto a la bibliografía utilizada.

La puesta sucesiva de cada uno de los puntos formará en su conjunto la memoria de nuestro proyecto, elaborando un proyecto limpio y ordenado que servirá como introducción y primera aproximación al análisis de textos.

2.2 Introducción al Machine Learning

El aprendizaje automático es una de las tecnologías que más revolución ha causado a lo largo de la historia, suponiendo un antes y un después en la resolución de problemas de diferente índole, ofreciendo incluso soluciones a problemas que, hasta la aparición de este tipo de algoritmos, nunca habían podido ser resueltos.

Este concepto nos lleva a pensar en la forma en la que un ente (o en nuestro caso, computador) es capaz de aprender de forma autónoma a partir de una serie de ejemplos. A simple vista, puede parecer un simple conjunto de reglas variantes dependiendo de los objetos vistos, pero este tipo de algoritmos va mucho más allá de lo que se pueda imaginar a priori.

El fenómeno del aprendizaje está produciéndose contantemente en el mundo biológico y ha sido en todo momento la principal referencia que han tenido los científicos e ingenieros para elaborar este tipo de algoritmos. Nosotros los humanos tenemos inherido el aprendizaje a través de estímulos y respuestas de nuestro cuerpo, por ejemplo: Si acerco la mano al fuego me quemo, por lo tanto, evitaré acercarme al fuego en el futuro.

Llevando este concepto a la informática nos damos cuenta de que, para resolver cualquier tipo de problema, debíamos elaborar un algoritmo capaz de estudiar todos los datos de entrada para desarrollar una salida dependiente del problema en cuestión.

El aprendizaje automático surge cuando nos topamos con problemas que no cuentan con un algoritmo trivial que nos brinde de la solución de este, como por ejemplo los relacionados con la visión artificial, el reconocimiento del habla o la conducción autónoma. Este tipo de problemas plantean un nuevo reto a la sociedad, consiguiendo que se comiencen nuevas investigaciones que consigan resolver este tipo de problemas.

Estas investigaciones introducen el término de aprendizaje en el mundo digital, consiguiendo que sea el propio algoritmo el que busque patrones a partir de un conjunto de datos de entrada, sin la necesidad de indicar explícitamente los pasos a seguir. De este concepto nació los algoritmos de *Machine Learning,* los cuales conseguían a la perfección replicar este comportamiento humano que hemos expuesto anteriormente.

Estos algoritmos cuentan principalmente con dos etapas: una primera dedicada a la preparación de los datos de entrada y construir un modelo efectivo usando estos; y una segunda donde se utiliza este modelo generado sobre nuevos datos que nos ayudarán a elaborar una serie de resultados.

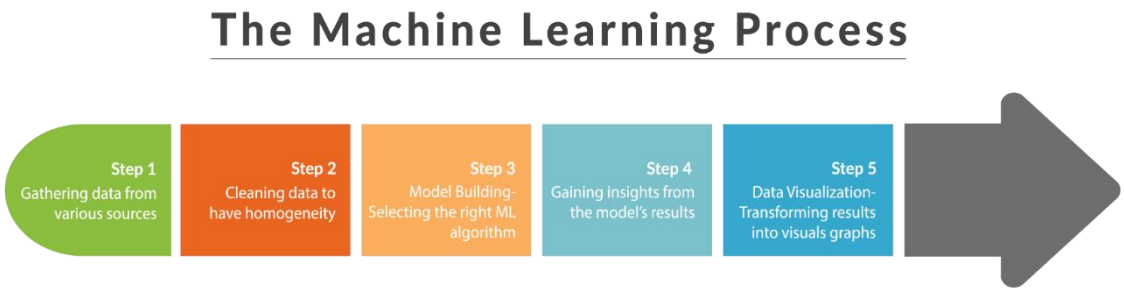


Figura 4. *Etapas del Machine Learning*

Como podemos ver en las etapas de estudio del en la Figura 4, la elaboración de gráficos y tablas de precisión es una etapa vital en el uso de estos algoritmos, siendo obligatorio realizar un correcto análisis de los resultados arrojados por estos.

Los modelos, al generarse de forma automática sobre una estructura preestablecida por el ingeniero, no pueden estudiarse en profundidad a simple vista, por ello, los datos resultantes de ellos son la viva imagen de estos, siendo la única forma de comprender el comportamiento de estos.

Se pueden diferenciar diferentes tipos de algoritmos de Machine Learning, los cuales son usados de forma independiente o de forma conjunta dependiendo del tipo de problema que estemos tratando. Estos son:

- **Aprendizaje Supervisado**: En todo momento tenemos constancia de la salida que debe de tener un conjunto de entrada determinado. El modelo que se genera intenta que los datos de entrada lleguen al resultado conocido.

- **Aprendizaje no Supervisado**: No se conoce la salida para el conjunto de entrada. Este algoritmo busca similitudes entre los datos de entrada, generando agrupaciones en función de sus características.

- **Aprendizaje por refuerzo**: En este tipo de aprendizaje no contamos con ejemplos de los cuales aprender, sino que el algoritmo a través de prueba y error consigue elaborar una solución factible a través de la función recompensa/castigo**.**

En la figura 5 podemos ver un pequeño esquema de los diferentes tipos de aprendizaje.

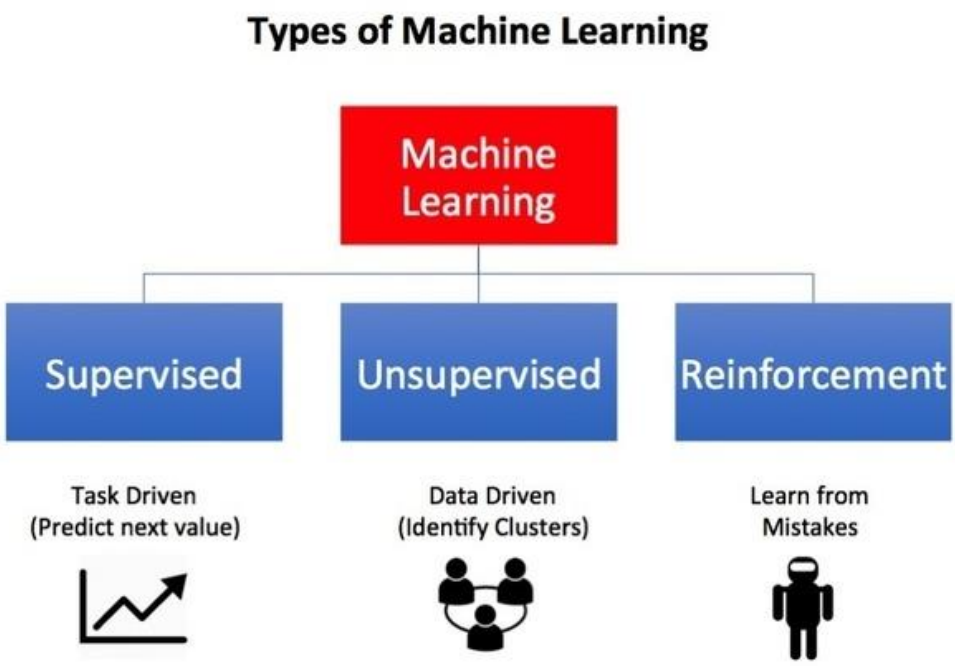


Figura 5. *Diferentes tipos de Aprendizaje Automático*

Tras esta pequeña clasificación sobre los diferentes tipos de aprendizaje que podemos encontrar según la naturaleza del problema, continuaremos con la siguiente clasificación, atendiendo esta vez al tipo de conocimiento que podemos queremos adquirir. Podemos diferenciar tres tipos:

- **Aprendizaje simbólico**: Adquiere conocimiento que permita representaciones de conceptos (reglas lógicas, valores de atributos…).

- **Aprendizaje numérico**: Adquiere conocimiento relacionados con factores numéricos no relacionados (pesos de una red neuronal…)

- **Aprendizaje mixto**: Adquiere conceptos expresados por medio de factores numéricos. Relaciones Valor-Número.

Por último y para finalizar la introducción al *Machine Learning*, echaremos un ojo a alguno de los problemas que se han tratado con estas herramientas, mencionando algunos de los algoritmos que se han usado parala resolución de estos.

- **Algoritmos de Regresión**: Estos son los encargados de clasificar o predecir valores. A lo largo del entrenamiento buscan minimizar el error. Ejemplos de algoritmos: Regresión Lineal, Regresión Logística.

- **Algoritmos basados en Instancias**: Utilizados especialmente en casos donde los datos y entrenamiento son muy importantes y son requeridos por el modelo. Ejemplo de algoritmo: K-vecinos más cercanos (kNN)

- **Algoritmos de Árbol de Decisión**: Este tipo de algoritmos se caracterizan por la gran potencia que tienen clasificando la información usando bifurcaciones por probabilidad de ocurrencia. A menudo hay que usar técnicas de post procesado (como la poda) para mejorar el modelo. Ejemplos de algoritmos: Árboles de Clasificación y Regresión (CART), Árbol condicional.

- **Algoritmos Bayesianos**: Se apoyan en el teorema de Bayes y son usados en problemas de clasificación y regresión. Ejemplos de algoritmos: *Naive Bayes*, *Bayesian Network*.

- **Algoritmos de Clustering (agrupación)**: Normalmente están orientados a problemas de aprendizaje no supervisado, donde consigue agrupar los datos en función de las características similares de los datos de entrada. Ejemplos de algoritmos: *K-means*, *Hierarchical Clustering*.

- **Algoritmos de Redes Neuronales**: Estos algoritmos supusieron una evolución importante a los problemas de *machine learning*. Están basados en las funciones biológicas de las redes neuronales humanas, consiguiendo grandes resultados en cualquier tipo de problema. A su salida, estaban muy limitadas por la potencia de cómputo de la época. Ejemplos de algoritmos: Perceptrón, *Back-Propagation*.

- **Algoritmos de Aprendizaje Profundo**: Se tratan de la evolución directa a las Redes Neuronales Artificiales, la cuales usaron los grandes avances en tecnología para conseguí conectar diferentes capas de redes neuronales, consiguiendo una gran capacidad de cómputo en paralelo. Ejemplo de algoritmos: *Convolutional Neuronal Networks* (CNN), *Long Short Term Memory Neuronal Networks* (LSTNN).

- **Algoritmos de Reducción de Dimensión**: Explotan los datos que disponemos de forma no supervisada con la intención de reducir y simplificarlos. Ejemplo de algoritmo: *Principal Component Analysis* (PCA).

- **Procesamiento del Lenguaje Natural**: Intentan comprender el lenguaje humano tal y como lo conocemos, tanto de forma escrita como de forma oral.

  Este tipo de problema será el que tratemos a lo largo del TFM, donde descubriremos de mejor manera las limitaciones y ventajas de diferentes implementaciones de este algoritmo. El análisis de sentimientos y los dispositivos procesadores de lenguaje oral (Siri, Cortana, Alexa…) son algunos de los logros más relevantes conseguidos por esta rama del *machine learning*. Este tipo de problema se resuelve usando técnicas de preprocesado de textos junto a algún algoritmo de los que hemos mencionado anteriormente.

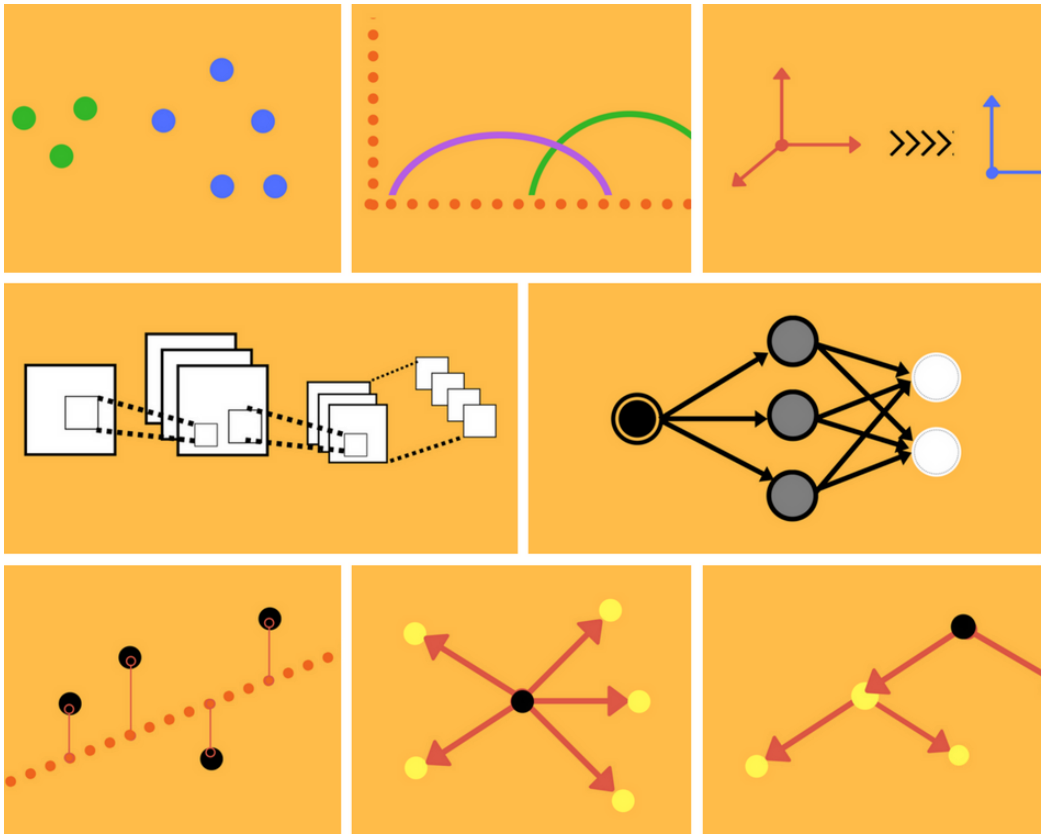


Figura 6. *Representación diferentes técnicas de Machine Learning*

2.3 Introducción General a las Herramientas Usadas

En este segundo apartado de introducción haremos una pequeña revisión a las diferentes tecnologías que más relevancia han tomado a lo largo del proyecto, formando parte fundamental del mismo.

La realización de este apartado (junto con el anterior) están orientados a introducir el tema de forma continua hasta llegar al desenlace de nuestro problema, donde podremos dar sentido a todo este conocimiento que estamos poniendo en premisa de la experimentación realizada, la cual podemos ver en capítulos posteriores de forma completa.

2.3.1 Introducción a la Obtención de Datos

Antes de realizar cualquier proyecto relacionado con el aprendizaje automático es importante conocer la forma en la que los datos van a ser obtenidos para su posterior análisis. La obtención de datos de calidad es tan importante como los pasos posteriores de procesamiento y construcción del modelo. Sin unos buenos datos de entrada sería muy difícil construir un modelo decente para resolver el problema (Gudivada, Apon, & Ding, 2017).

Esta obtención de datos se puede dividir en dos secciones sencillas de interpretar: una primera dedicada a la obtención de datos de los datos que usaremos para el entrenamiento del modelo; y una segunda que se focaliza en los datos que posteriormente serán pasados a un modelo ya entrenado para ser clasificados, los cuales será importante que tengan una fuerte correlación con los datos de entrenamiento.

Estos segundos datos son los que impondrán las condiciones a los primeros de forma que se adapten de la mejor forma posible a los datos finales. Cuanto más parecidos sean este tipo de datos (en estructura, no es parecido "físico"), mejores resultados conseguiremos a la larga, pues mejor entrenado estará nuestro modelo para actuar frente a este tipo de situaciones.

	id	dataset	text	toxicity	sexist	of_id
0	0	other	MENTION3481 i didn't even know random was an option!	0.118180	False	-1
1	1	other	Bottom two should've gone! #mkr	0.251850	False	-1
2	2	callme	MENTION3111 MENTION3424 ladyboner deserves so much more credit than dudeboner. #bonerdebate #reddetails	0.113331	False	-1
3	3	other	She shall now be known as Sourpuss #MKR #KatAndre #FailedFoodies	0.531153	False	-1
4	4	other	Tarah W threw a bunch of women under the bus so she could get Wadhwa's support for her Women in Tech book.	0.118718	False	-1
5	5	callme	I just don't trust an adult who uses coupons.	0.119590	False	13227
6	6	other	Not again #MKR	0.198415	False	-1
7	7	other	#MKR2016 returns in 2020 once the all the couples, intruders, gate crashers and second-chancers are eliminated MENTION786 #MKR #MKR2015	0.447262	False	-1
8	8	other	MENTION526 most abuse comes from gamergate accounts that are 30-90 days old.	0.254602	False	-1
9	9	other	Great to see the local National Park workers tucking into a free feed. How about you empty those loo's instead. #MKR	0.392942	False	-1
10	10	callme	All my sons have grown up with computer games but I'm not interested. I see them as a male thing. MENTION203	0.091025	True	-1

Figura 7. *Datos de entrada de nuestro propio dataset*

La buena obtención de datos se rige normalmente de un buen conjunto de datos (*dataset*) el cual esté preparado para su explotación de forma automática. Para conseguir un buen *dataset* es necesario realizar un estudio exhaustivo de nuestro problema, recopilando a lo largo del tiempo diferentes datos que nos puedan ayudar a realizar el modelo final, suponiendo una gran cantidad de horas y trabajo que se escapan de los límites de este TFM.

En esta sección nos centraremos más en la segunda parte de obtención de datos donde, como ya hemos mencionado antes, debemos usar una serie de herramientas que nos permitan obtener datos directamente de las propias webs de noticias que podamos encontrar, intentando ser lo más flexible posible con los diferentes tipos de temas que puedan entrar.

A lo largo de la historia, la herramienta a la que más uso se le ha dado ha sido la denominada como *web scraping*, la cual consiste en la obtención de información útil directamente de las estructuras de las páginas web tal y como las conocemos (Web Scraper, 2021).

Este tipo de algoritmo es tan potente y puede ser utilizado en tantos sectores que las páginas más importantes del mundo tienen un sistema de seguridad dedicado a limitar el uso de estos, creando limitaciones que hay que tener en cuenta a la hora del usar este tipo de herramientas (Milly, 2021).

Uno de los casos más importantes es el generado por el gigante de ventas Amazon, el cual cambia de forma contante la estructura interna de su web de forma que las etiquetas y estructuras sean distintas cada cierto tiempo, evitando así el uso del *web scraping* para por ejemplo realizar un seguimiento de precios. Si alguien pudiera obtener a través de herramientas como estas el catálogo completo de Amazon podría suponer la perdida de millones de dólares para la compañía.

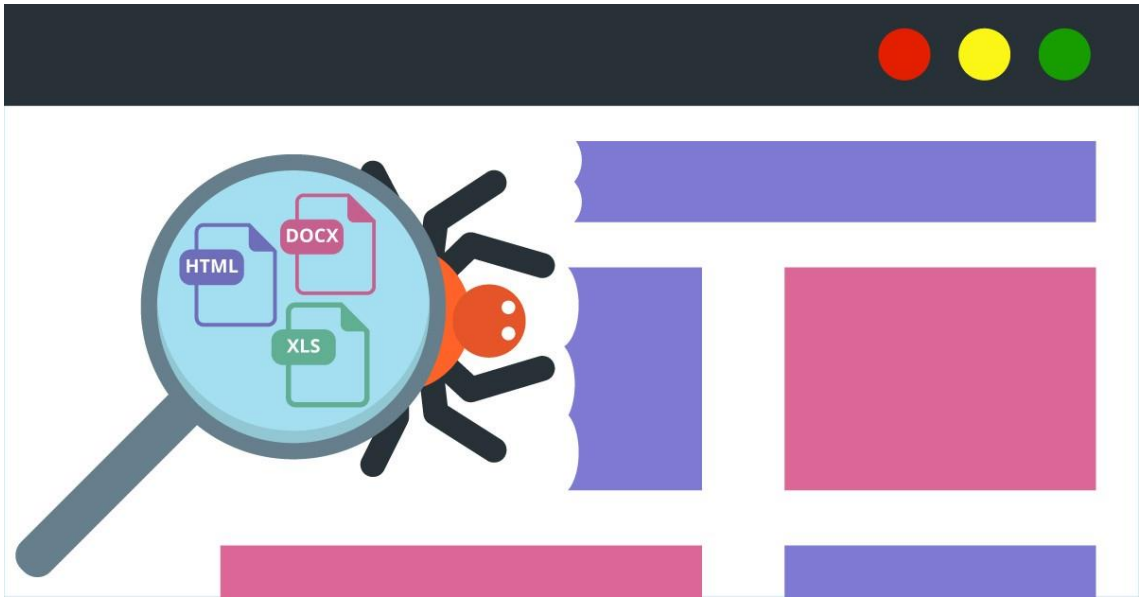


Figura 8. *Web Scraping*

En nuestro caso usaremos esta herramienta para la obtención de textos planos los cuales serán analizados y posteriormente presentados al usuario final con la información extraída de ellos, siendo una práctica ética y que no supone ningún tipo de daño ni para web de la que vamos a extraer los datos ni para los creadores de contenido de esta.

2.3.2 Introducción al Preprocesamiento de Datos

Durante la realización del máster se ha dejado constancia de la importancia de la preparación de los datos que vamos a usar en cualquiera de los ámbitos estudiados, desde la creación de una base de datos relacional hasta la creación de un modelo complejo para el aprendizaje automático.

Tanta es la importancia de este fenómeno que incluso hemos cursado una materia dedicada únicamente al tratamiento de estos datos, recalcando de forma mucho más agresiva la importancia que tiene un buen preprocesamiento y limpiado de los diferentes elementos que forman nuestro conjunto de datos, lo cual puede suponer el éxito o fracaso de la tarea a realizar a posteriori con ellos.

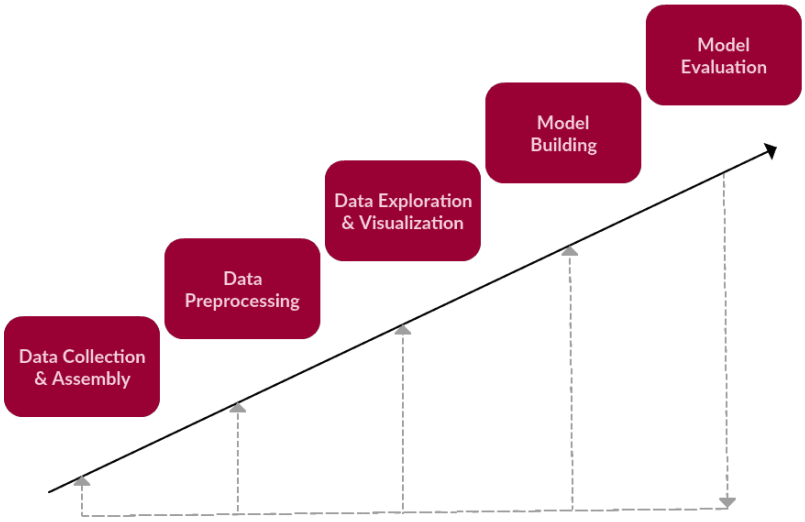


Figura 9. *Fases fundamentales de la creación de un modelo*

En la figura 9 se puede apreciar una escalera con las diferentes fases de la creación del modelo, donde el preprocesado de datos se representa en una de las fases más tempranas de esta, siendo una de las etapas que crearan una buena base sobre la que construiremos los pilares de nuestro proyecto final.

Esta lleva considerándose importante desde las primeras fases del aprendizaje automático, siendo una de las que más estudios acumula a lo largo del tiempo, como este artículo que habla de la importancia de la misma datado de 1996 (Alexander & Jetton, 1996) o este otro datado de 2007 (McCrudden & Schraw, 2007).

La evolución de las diferentes tecnologías a lo largo de los años ha supuesto un gran avance tanto los modelos de aprendizaje automático como para los algoritmos de preprocesamiento de datos, llegando a tener en la actualidad potentes herramientas, las cuales se vuelven un indispensable a la hora de construir un modelo de calidad.

Dentro de este gran mundo del preprocesamiento de datos, vamos a centrarnos en el relacionado con la clasificación de textos. Este tipo de problemas en concreto realizan un uso intensivo del preprocesado, siendo obligatorio realizar una buena limpieza del texto "sucio" (sin tratar) antes de poder aplicar alguna técnica de aprendizaje automático.

Este tipo de preprocesado se basa en el *Natural Languaje Processing* (NLP), el cual está en constante estudio y mejora, consiguiendo cada vez mejores resultados con los diferentes textos que podemos utilizar como datos de entrada. Este procesado forma parte fundamental de la inteligencia artificial tal y como la conocemos hoy día, siendo la base de los dispositivos inteligentes como Alexa (a nivel oral) o de los chats automáticos (a nivel textual).

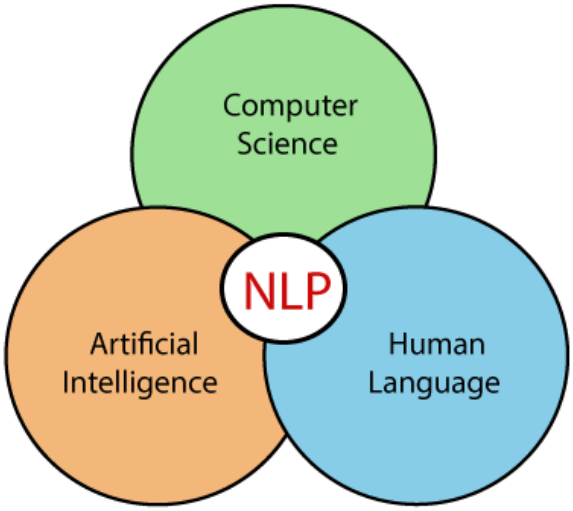


Figura 10. *Diagrama de cómo encaja NLP en el entorno de la IA*

Uno de los principales problemas que ha tenido y sigue teniendo este tipo de algoritmos proviene de la propia naturaleza de los textos, la cual, al estar en formato de caracteres, supone un impedimento crítico en la creación de modelos de aprendizaje automático, pues estos solo pueden tratar con datos de entrada en formato numérico, quedando completamente excluidos las cadenas de textos.

La transformación del texto al formato numérico tiene multitud de alternativas, teniendo cada una ciertas ventajas e inconvenientes frente a otras, pero todas aportando una interpretación del texto en un formato compatible para poder llevar a cabo nuestra fase de aprendizaje.

Una de las primeras aproximaciones que se encontraron para realizar la transformación de texto a números fue utilizar el conteo de palabras, la cual, a partir de reconocer diferentes caracteres dentro de los textos de nuestro conjunto de documentos, es capaz de elaborar un contador de estos, consiguiendo una estructura numérica que puede servir como entrada a nuestro modelo, tal y como podemos ver en la figura 11.

No es difícil encontrar multitud de artículos donde se utilicen las técnicas NLP para realizar este preprocesado (Weng, 2019); o artículos donde se utilice el lenguaje Python (Dhingra, 2022), donde podemos encontrar un gran catálogo de librerías que integran el procesamiento de datos a la perfección, donde especialmente destaca la librería Sklearn.

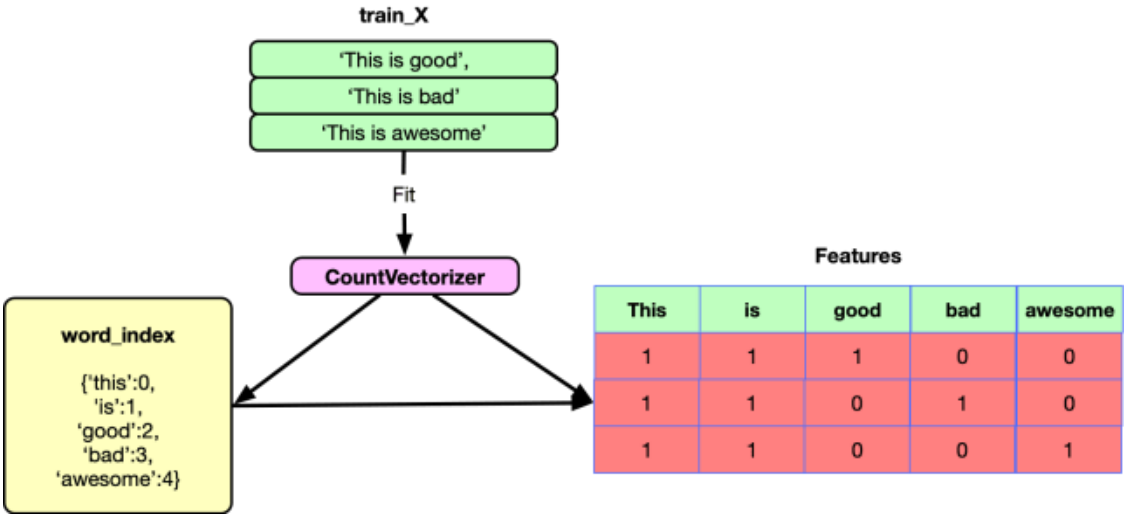


Figura 11. *Esquema de utilización de Count Vectorizer*

Este tipo de preprocesado sigue evolucionando día a día, consiguiendo cada vez mejores resultados usando menos tiempo gracias a las nuevas tendencias de NLP, como por ejemplo las últimas investigaciones con el uso de los *Transformers*. Estos prometen mejorar en gran medida el procesado de textos usando en lugar de palabras sueltas las estructuras de textos completas (frases) para realizar el entrenamiento (Kłeczek, 2020), aportando una forma de procesamiento rompedora que supone una gran ventaja frente a los actuales métodos de tratamientos de caracteres.

2.3.3 Introducción al Análisis de Sentimientos

Como apoyo al análisis que vamos a realizar de los textos obtenidos, vamos a usar una de las herramientas que más impacto ha generado en los últimos tiempos en multitud de situaciones diferentes, sobre todo en las relacionadas con la venta.

El análisis de sentimientos, como hemos mencionado anteriormente, proviene del tipo de problemas relacionados con el lenguaje natural, el cual consigue obtener información relevante a partir de una serie de caracteres: una frase, un párrafo e incluso una base de datos completa de documentos.

Esta información conseguida puede aplicarse en multitud de situaciones donde conocer la intención lingüística del autor de cierto texto puede suponer una ventaja competitiva sobre otras empresas de la misma índole, incluso llegando a predecir o evitar una pérdida de un cliente potencial, o consiguiendo conocer el estado actual de una cierta marca únicamente estudiando lo que se dice de ella en comentarios en las redes sociales.

La introducción del *machine learning* en sectores que no están relacionados directamente con la informática ha sido uno de los precursores del éxito de este, despertando la curiosidad de multitud de emprendedores que buscan mejorar sus empresas a partir del uso de estas herramientas, como podemos comprobar en el *artículo* de Kerstin Denecke, donde habla de la incorporación de estas en la rama de la medicina (Denecke & Deng, 2015).

El principio de este tipo de análisis de texto, tal y como su nombre indica, es obtener información sobre la intención y el "sentimiento" expuestos por el emisor en los diferentes textos a analizar, obteniendo así una aproximación sobre la opinión que una persona tiene sobre un cierto tema.

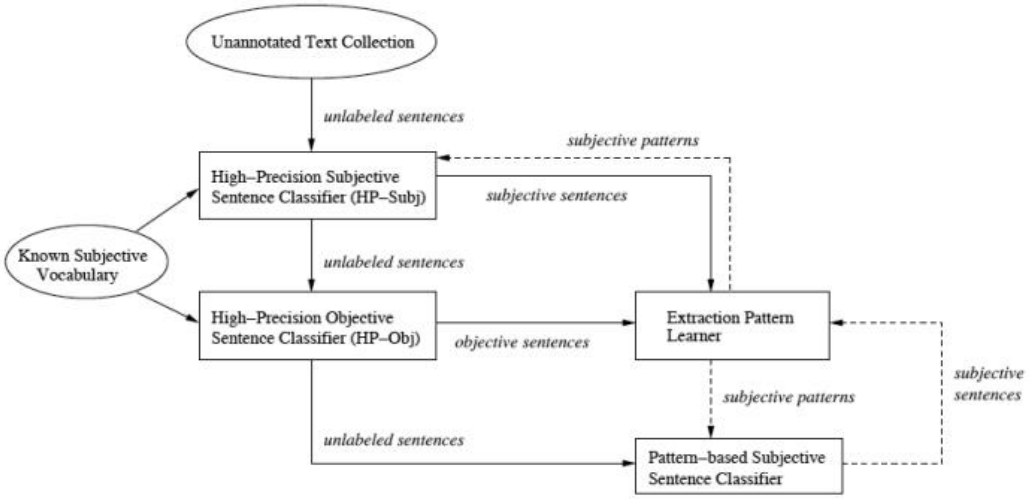


Figura 12. *Esquema de análisis de textos*

En la figura 12 extraída del *paper* de Adam Westerski (Westerski) podemos apreciar cómo se aplican diferentes tipos de aprendizaje para obtener estas características y patrones de los textos, obteniendo así la información relevante para el usuario de este tipo de modelos.

En este mismo *paper* podemos apreciar el estado del arte de este tipo de modelo, conociendo en profundidad los tipos de evaluación que se han usado a lo largo del tiempo para clasificar el vocabulario en diferentes tipos, observando como una entidad computacional ha conseguido comprender estructuras complejas de palabras, como las frases, siendo capaz de obtener la subjetividad e intención de emisor (algo incluso complicado para los seres humanos) con tan solo el uso de herramientas de análisis de texto.

Uno de los mejores ejemplos que confirman el éxito de este tipo de problemas es la página de *Hugging Face*, la cual cuenta con una de las comunidades de Inteligencia Artificial (IA) más grandes del mundo, donde podemos encontrar uno de los analizadores de sentimientos más potentes del mundo (Pascual, 2022).

Figura 13*. Análisis de Sentimientos Hugging Face x Python*

Todo esto convierte al análisis de sentimientos en una de las herramientas indispensables para nuestro problema, añadiendo una de las características más atractivas que se pueden obtener del análisis de textos. Cumplimentando en gran medida el alcance de nuestro proyecto.

2.3.4 Introducción al Estudios de Resultados

Una de las etapas que encontramos en la creación de modelos de aprendizaje automático, tal y como podemos ver en la figura 9, es el análisis de los resultados obtenidos, tanto por parte del modelo, como para la representación de los datos a los usuarios finales de nuestro proyecto.

A lo largo del máster hemos aprendido a sacar el mayor partido posible a nuestros datos con el uso de diferentes herramientas como matplotlib (14 a) o Tkinter (14 b), cada una dedicada a una sección diferente dentro de nuestra pila de trabajo. En el caso de matplotlib para realizar un estudio del modelo obtenido y en el caso de Tkinter para realizar una interfaz de usuario que facilite la representación de los datos al usuario final.

Figura 14. *Logos de MatplotLib y Tkinter*

La realización de un buen estudio de los resultados nos puede ayudar a mejorar la eficiencia y eficacia de estos fabricando, a partir de los resultados de los modelos primerizos, modificaciones de estos que cada vez nos vayan acercando un poco más a la solución final de nuestro problema.

Este tipo de seguimiento y mejora del modelo se realiza en todos los tipos de problemas de *machine learning*, sobre todo en los relacionados con el aprendizaje profundo, donde el comportamiento del modelo en su interior es opaco para el programador, únicamente teniendo los datos de salida para realizar una aproximación a las acciones que está tomando este en todo el proceso de aprendizaje.

Por otro lado, el uso de interfaz de usuario (GUI) y elementos gráficos ayudan a la interpretabilidad de nuestro programa y modelo, ofreciendo una solución mucho más atractiva a la hora de presentar un trabajo basado en la computación.

En la asignatura de Visualización de Datos se nos ha inculcado toda la cultura que hay detrás de cada una de las secciones que puede tener una representación gráfica, haciéndonos ver la importancia que tienen estos en el sector que estamos trabajando, siendo parte fundamental y obligatoria en cada una de las presentaciones que vayamos a realizar dentro de nuestro trabajo.

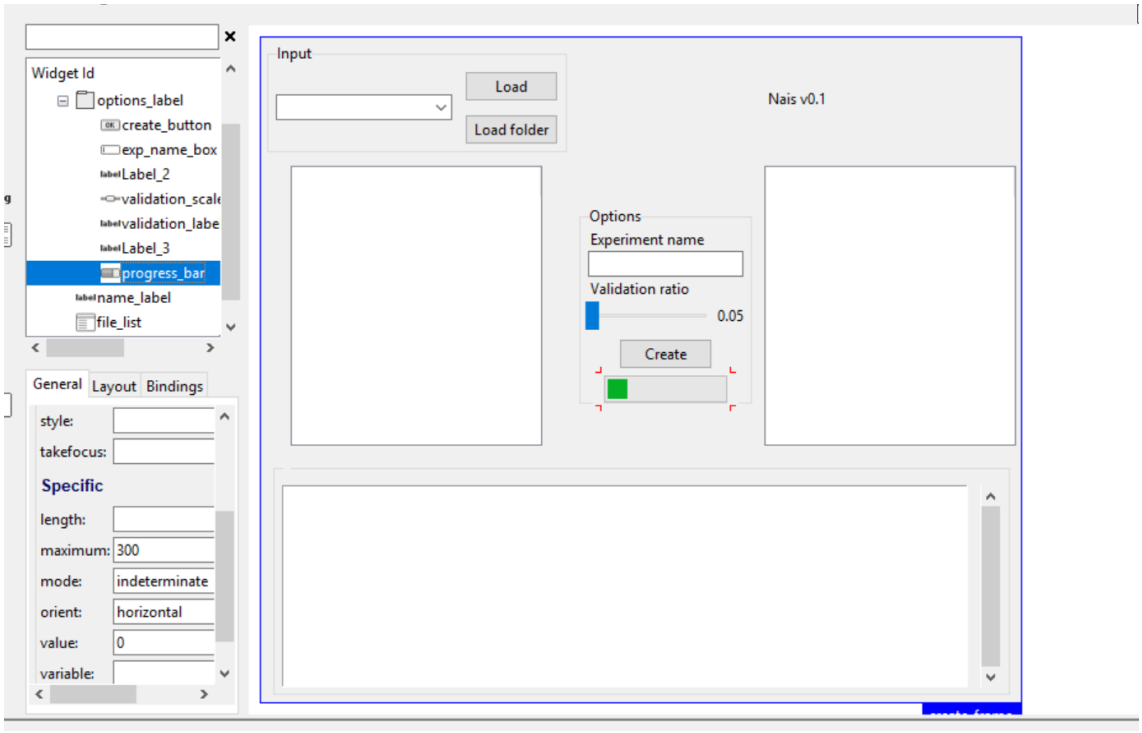


Figura 15. *Ejemplo de GUI realizado en tkinter*

Habiendo introducido las principales tecnologías a aplicar en durante la elaboración de nuestro TFM y habiendo recalcado la importancia del estudio de los datos que vamos a obtener, podemos concluir este apartado y dar siguiente al siguiente punto de la introducción, donde haremos un repaso a los objetivos que vamos a tratar a lo largo del proyecto.

2.4 Objetivos del Proyecto

Con todo el contexto visto a lo largo de la memoria no es complicado definir los diferentes objetivos que formarán los principios básicos de los trabajos de experimentación, donde iremos supliendo cada uno de ellos conforme vamos realizando las tareas asociadas.

En esta pequeña sección representaremos de forma resumida el objetivo principal de nuestro TFM, subdividiendo el mismo en otros 4 más específicos, los cuales se usarán de índice para los siguientes temas donde expondremos la experimentación aplicada.

El pilar fundamental de nuestro proyecto y que se tomará como **objetivo principal** será la realización de un programa capaz de analizar textos buscando patrones y sesgos machistas en ellos, proporcionando, en última instancia, un panel representativo con la información más relevante del estudio, pudiendo conocer en todo momento el porqué de esos resultados.

Los subobjetivos que con su realización supondrán la finalización del trabajo son los siguientes:

I. **Obtención de textos.** El primer subobjetivo será construir un script de *web scraping* capaz de obtener información relevante de los textos extraídos de la web Independent de forma automática, siendo lo más flexible posible a la hora de conseguirlos.

II. **Procesamiento de textos.** Como hemos estudiado anteriormente, esta fase es de vital importancia para conseguir buenos resultados en fases posteriores donde entran en juego las herramientas de aprendizaje automático.

Hay que tener en cuenta que este preprocesamiento debe ser aplicado a su vez a los textos analizados, teniendo que ser lo más flexible ante la variedad de caracteres que pueden entrar en nuestra aplicación.

III. **Aplicación de soluciones Machine Learning.** En esta fase realizaremos el entrenamiento y modelado de un sistema basado en aprendizaje automático a partir de un *dataset* con información sobre textos clasificados como machistas o no. Estos resultados los fusionaremos con un análisis de sentimiento para obtener un resultado lo más eficaz posible.

IV. **Representación de los datos.** Por último, realizaremos un cuadro resumen con la información útil para el usuario, exponiendo de forma clara el puntuaje obtenido por nuestra aplicación, entre los que se incluirá el porcentaje de machismo, el valor de subjetividad del texto y el sentimiento expuesto.

Como hemos podido ver, estos objetivos corresponden a la perfección con las diferentes tecnologías expuestas en las diferentes subsecciones de la introducción, conociendo, en este punto de la memoria, los conceptos básicos de cada uno de los puntos a tratar en el proyecto, facilitado así el seguimiento de este.

En la figura 16 podemos apreciar de forma resumida los diferentes objetivos del proyecto, pudiendo visualizar de una mejor forma como la unión de los diferentes subobjetivos supone la creación y obtención con éxito del objetivo principal.

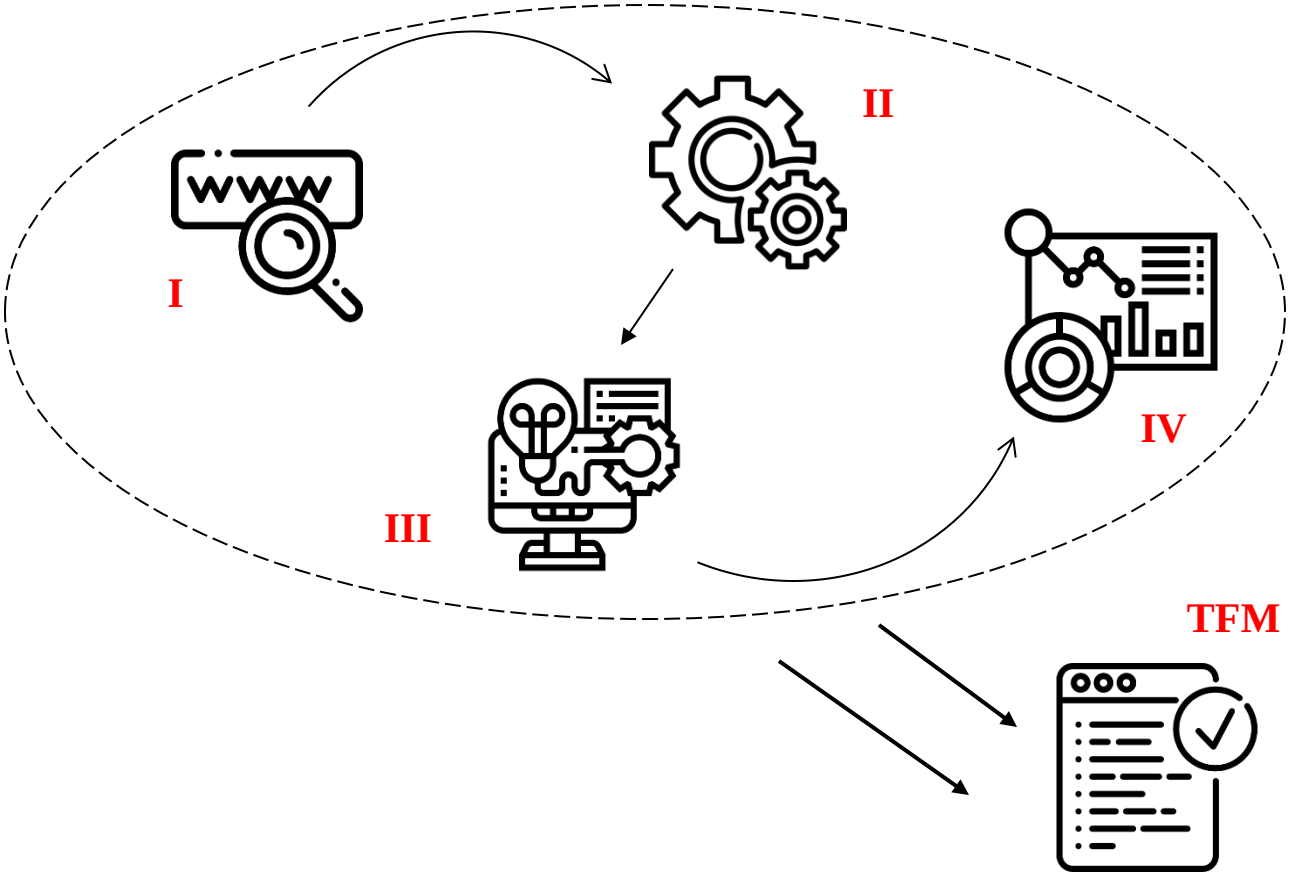


***Figura 16**. Esquema de los objetivos del TFM*

2.5 Fases del Proyecto

Con los objetivos bien definidos, podemos desarrollar el plan de trabajo que seguiremos a la hora de la realización completa de nuestro trabajo de fin de máster, consiguiendo dedicar a cada una estas etapas el tiempo y esfuerzo necesario para obtener el mejor resultado y así cumplir con cada uno de los objetivos propuestos en el apartado 2.4. .

Dada la naturaleza de nuestro problema, el primer paso que plantearemos es la investigación sobre el machismo y los diferentes proyectos que se han realizado en este ámbito, donde nos encontramos diferentes trabajos relacionados. “Call me sexist… but” es uno de los que más nos llamó más la atención, tomándolo como referencia para multitud de desarrollos (Samory, Sen, Kohne, Flöck, & Wagner, 2021).

Este *paper* nos servirá como punto de partida a la hora de plantear nuestro proyecto, conociendo una primera aproximación de los resultados a esperar obtener y del esfuerzo que supondría el estudio de cada una de las secciones que íbamos a desarrollar durante nuestro proyecto.

Realizado un estudio de viabilidad y habiendo descartado varias ideas relacionadas con el mismo, el primer paso evidente era la realización de un modelo con un *dataset* de prueba, el cual nos pudiera servir para estudiar el comportamiento de los datos con todo lo expuesto teóricamente, consiguiendo unos primeros resultados, los cuales nos servirían como análisis inicial de nuestro proyecto, pudiendo avanzar a partir de estos en la búsqueda de nuestro objetivo final.

La consecución de estos primeros resultados nos obligará a realizar una búsqueda exhaustiva de un *dataset* relacionado con el tema, consiguiendo analizar diferentes tipos de estructuras hasta finamente encontrar la idónea para nuestro problema.

Esta etapa supondrá un gran avance a la hora de establecer los límites de nuestro TFM, ya que, al saber el tipo de datos que vamos a disponer para nuestro problema, no será difícil conocer el punto más lejano a conseguir con ellos, estableciendo una meta factible a nuestro problema.

Acompañando a la búsqueda de datos, encontramos la inserción de estos en nuestra primera aproximación de modelo, donde debemos crear una primera versión de procesamiento de textos que posteriormente usaremos en las versiones finales. La correcta adaptación de este procesado es de vital importancia en vista de fases futuras, siendo una de las fases que más estudio y cuidado hemos tenido que emplear para conseguir buenos resultados.

Con un buen conjunto de datos tratados y con los resultados obtenidos por el primer modelo “de juguete” creado, podemos pasar a las fases más complejas de nuestro TFM. Estas consisten en la aplicación de diferentes técnicas vistas a lo largo del máster para conseguir optimizar de la mejor forma posible los modelos de aprendizaje automático usados.

La búsqueda continua de mejora supondrá la mayor parte de nuestro proyecto, el cual será completado junto a la creación de diferentes scripts relacionados con la obtención de los textos desde la web y de la creación de una GUI donde podamos expresar de forma clara los resultados obtenidos por nuestros mejores modelos.

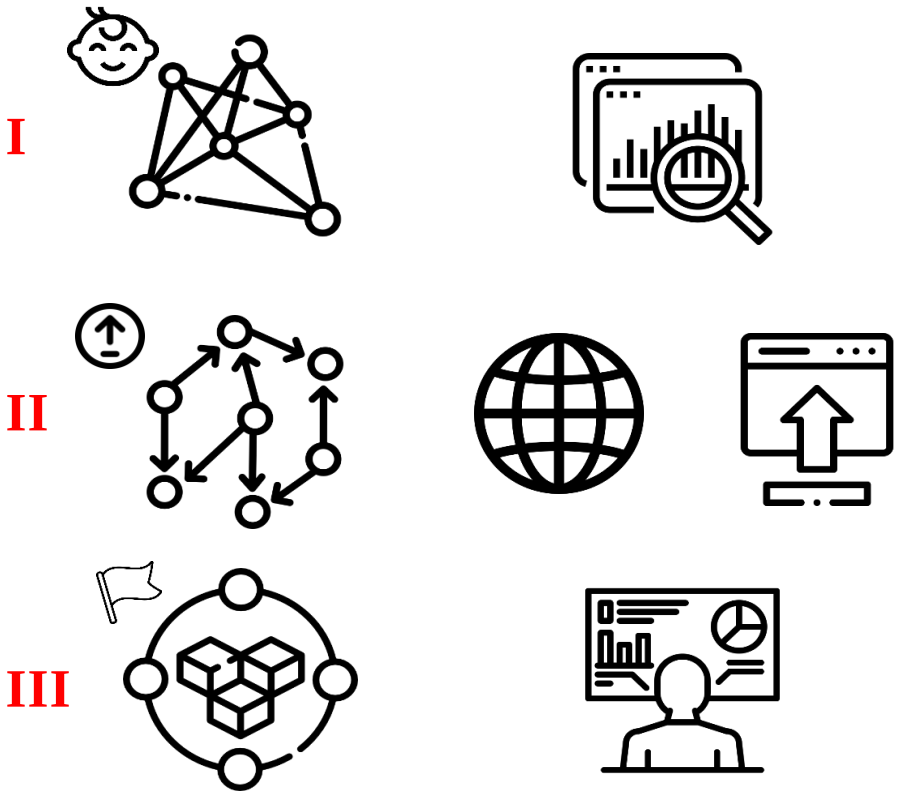


Figura 17. *Esquema del proyecto (I)*

Estos primeros conceptos completarán todo el grueso del proyecto, completando toda la parte de experimentación y consiguiendo todos los resultados necesarios para la poder comenzar la realización de los documentos explicativos de estos. Estos estarán constituidos por esta memoria y el power point correspondiente a la misma, el cual expondremos en la fecha de presentación del trabajo. Esta última fase podemos verla reflejada en la figura 18, la cual se trata de una continuación de la anteriormente expuesta (figura 17).

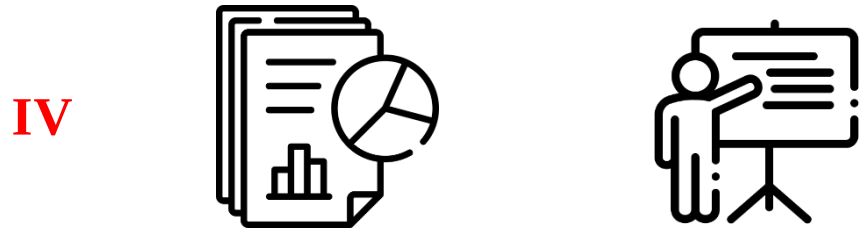


***Figura 18**. Esquema del proyecto (II)*

Con el cumplimiento y seguimiento de este plan de trabajo conseguiremos obtener todos nuestros objetivos, consiguiendo elaborar un proyecto de éxito cuyo contenido sea viable como punto de comienzo para futuros trabajos relacionados con este tema tan actual como es el machismo.

2.6 Tecnologías usadas

La consecución de los diferentes objetivos supone la realización de un proyecto el cual haga uso de multitud de herramientas de diferente índole, desde las dedicadas a la extracción de textos de la web hasta las dedicadas a la creación de interfaces gráficas de alta complejidad.

Siguiendo la estructura inherida a lo largo de la memoria con respecto a las diferentes fases del proyecto, dividiremos este subcapítulo en cuatro secciones, los cuales harán referencia a los cuatro subobjetivos de nuestro proyecto. Estas serán: Análisis de Textos de Internet (Web Scraping), Procesamiento de Datos, Modelaje de Soluciones de Aprendizaje Automático y Representación de los Datos (GUI).

2.6.1 Análisis de Textos de Internet (Web Scraping)

Esta primera fase será a su vez el primer paso que seguirá nuestro programa para conseguir el análisis final de la web, teniendo que ser lo más flexible posible ante las diferentes noticias que pueden entrar en la web, las cuales contendrán diferentes elementos.

En esta fase nos apoyaremos principalmente de la librería Beatiful Soap (figura 19), la cual es una de las más conocidas y potentes para realizar este tipo de análisis, siendo una candidata perfecta para usarla en la construcción nuestro programa.

***Figura 19**. Logo de Beautiful Soap*

2.6.2 Procesamiento de Datos

Habiendo estudiado la forma en la que vamos a obtener los datos y teniendo claro los tipos de documentos que vamos a tratar, vamos a estudiar las herramientas necesarias para conseguir obtener la información relevante de estos, siendo necesario el uso de diversas librerías.

Algo muy importante para tener en cuenta es la necesidad de usar la misma función de preprocesamiento tanto para los datos de entrenamiento como para los datos que vamos a analizar, teniendo que ser lo más flexibles posible ante los diferentes textos que pueden introducirse en nuestra aplicación.

Como ya hemos mencionado anteriormente en el capítulo 2.3.1, los datos que vamos a usar para el entrenamiento provienen de un *dataset* de *Tweets* (Figura 21) (mensajes de la aplicación Twitter), los cuales cuentan con una serie de caracteres especiales que deben ser tratados, como pueden ser los hashtags (#), las menciones o los emoticonos.

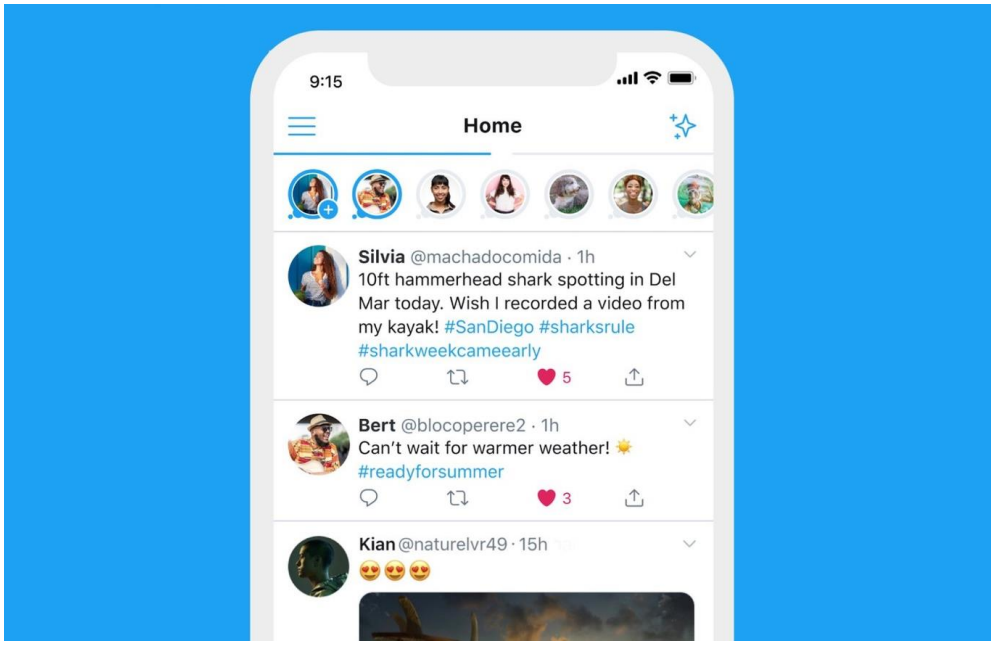


Figura 20. *Ejemplos de Tweets*

Una de las mejores librerías que dispone Python para el tratamiento de textos es la "re" (*Regular Expression Operations*), la cual nos permite operar las diferentes cadenas de caracteres usando expresiones regulares, siendo muy flexible a la hora de tener que tratar con diferentes tipos de entradas.

Las expresiones regulares se usan en multitud de situaciones donde queremos extraer o eliminar cierto tipo de patrones de nuestro *dataset*. Este tipo de expresiones son de alta complejidad y deben ser estudiadas con detenimiento para conocer el patrón exacto que estas reflejan, siendo casi indispensable el uso de páginas web que ayudan a identificar de forma clara cada uno de los elementos que forman la expresión regular.

Una de las mejores webs de este tipo que hemos encontrado es la llamada RegExr (Figura 22), donde incluso podemos probar nuestro propio texto con una expresión regular especialmente creada para el mismo. Esta misma web dispone de un pequeño apartado índice donde podemos consultar los diferentes tipos de símbolos que podemos utilizar en las expresiones.

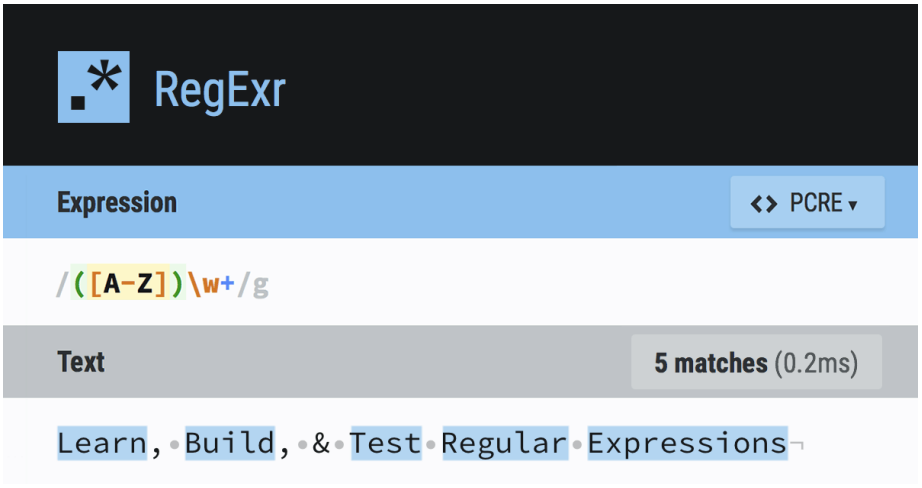


Figura 21*. RegExr*

Otra de las librerías que hemos utilizado para el tratamiento de los textos planos es "string", la cual está incluida en la instalación base de Python, al igual que re. Esta librería incluye una lista con diferentes símbolos de puntuación, pudiendo usar esta para la eliminación de estos de nuestros textos de forma sencilla.

Como último procesamiento de los textos en crudo (*raw*), se ha aplicado la llamada lematización, la cual consiste en reducir todas las palabras a su lema correspondiente, consiguiendo reducir en gran medida el número de palabras diferentes y de significado parecido que podemos encontrar en el texto (Techslang).

Figura 22. *Proceso de lematización*

En la figura 23 podemos ver un ejemplo de lematización para la palabra *study* y sus variantes, haciendo que tres palabras de diferente estructura se conviertan en la misma, algo que ayudara en las fases posteriores de entrenamiento.

Para utilizar esta herramienta hemos optado por usar la librería nltk, la cual cuenta con multitud de funciones dedicadas al preprocesamiento de textos, siendo una alternativa muy viable y de gran potencia de cómputo para ser utilizada en nuestro proyecto.

El tratamiento que hemos realizado hasta ahora solo nos ha ayudado a eliminar el ruido de las cadenas de caracteres, consiguiendo a su salida otra cadena de caracteres, siendo un formato que no podemos usar para entrenar los modelos, pues estos necesitan de una entrada numérica para funcionar de forma correcta.

Para realizar esta transformación podemos encontrar multitud de funciones que nos ayuden a traspasar nuestras cadenas de texto a formato numérico, consiguiendo así una entrada válida para el entrenamiento y predicción de nuestro modelo. La mayoría de estas se encuentran incluidas dentro de la librería scikit-learn, como pueden ser CountVectorizer o TfidfVectorizer.

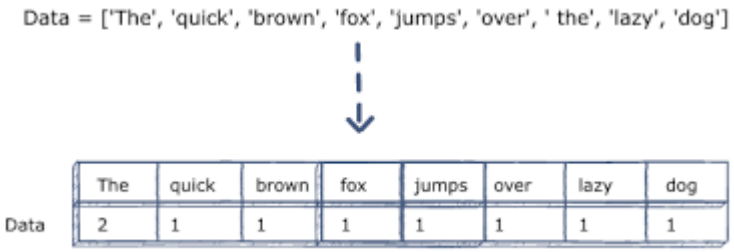


Figura 23*. Ejemplo de CountVectorizer*

En la figura 24 podemos ver un ejemplo básico del funcionamiento del primero de los algoritmos, el cual consiste en el conteo de las diferentes palabras que contiene una cadena de entrada, convirtiendo cada uno de los elementos de esta en columnas, mientras que a las filas las convierte en el número de veces que aparece este elemento en cada uno de los documentos de nuestro *dataset*.

Este algoritmo es uno de que mejor funciona en este tipo de problemas, siendo una solución factible y de fácil implementación, aunque cuenta con algunas desventajas, ya que varias entradas de un documento (elemento del *dataset*) pueden tener puntuaciones similares si cuentan con las mismas palabras, siendo difícil la clasificación entre estos.

Una de las alternativas que surgió a partir de este fue el segundo mencionado anteriormente, el cual podemos ver en funcionamiento en la figura 25. Este hace uso de las fórmulas de frecuencia de términos, como su propio nombre indica (TF-IDF – *Term Frecuency – Inverse Document Frecuency*). Este supone un paso más al algoritmo anterior, realizando una multiplicación de la frecuencia de aparición de los términos en un mismo documento con la inversa de la frecuencia que tiene ese término de aparición en otros documentos (Karabiber, 2020). Estas fórmulas son las siguientes:

$$TF = \frac{Número\ de\ veces\ que\ aparece\ un\ término\ en\ el\ documento}{Número\ total\ de\ términos\ en\ el\ documento}$$

$$IDF = \log\left(\frac{Número\ de\ documentos\ en\ el\ dataset}{Número\ de\ documentos\ en\ el\ dataset\ que\ contienen\ el\ término}\right)$$

$$TF - IDF = TF * IDF$$

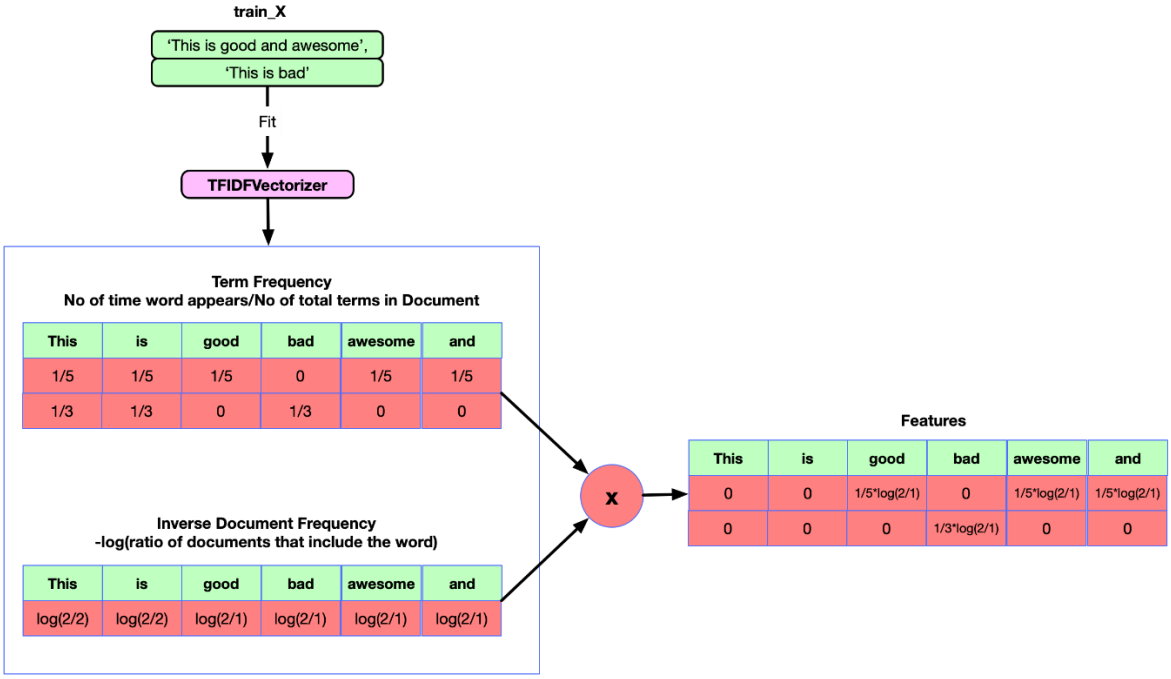


***Figura 24**. Ejemplo de TFIDFVectorizer*

Este tipo de algoritmo ayuda a controlar el número de veces que aparece un término en documentos que no son el mismo, obteniendo un valor más real para este tipo de entrenamientos, decantándonos por esta segunda opción para entrenar y predecir los textos de nuestro *dataset* de entrada.

Debida a la dificultad de realizar estas técnicas en análisis de texto, consideramos usar técnicas de alta complejidad computacional, las cuales usan el aprendizaje automático para la generación de textos similares a los de nuestro conjunto de datos de forma que podamos crear textos que pertenezcan a la misma clase y mantengan su significado pero que funcionen como un ejemplo nuevo para nuestro algoritmo (Shahul, 2022).

La mejor herramienta para realizar este tipo de procesamiento es TextAttack, la cual, usando transformadores y la librería de TensorFlow, realiza este tipo de procesamiento. En concreto, nuestro procesamiento se basa en la búsqueda de sinónimos de las frases de nuestra clase minoritaria (Figura 26), consiguiendo multiplicar el número de ejemplos disponibles y teniendo en definitiva un *dataset* balanceado.

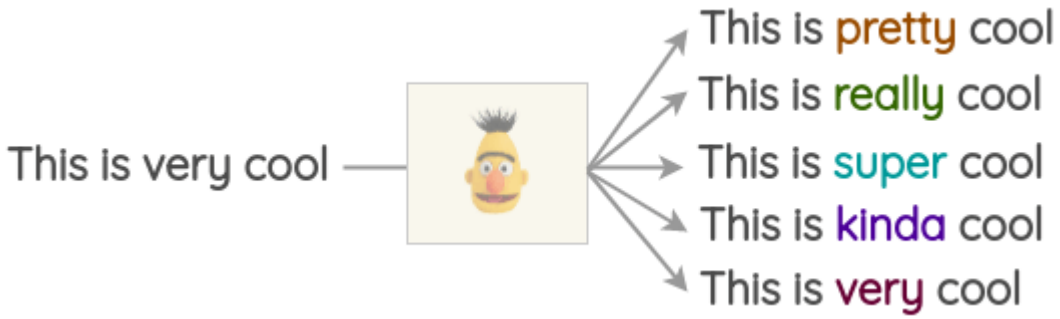


***Figura 25**. Ejemplo de uso de TextAttack*

Con las técnicas de extracción y transformación de datos solo nos haría falta definir las relacionadas con la manipulación de estos, siendo de gran ayuda a la hora de necesitar de cambiar los datos en función de cómo estén distribuidos en nuestro *dataset*.

Siguiendo el estudio de nuestro conjunto de datos podemos darnos cuenta de que una de las clases está mucho menos representada, teniendo que afrontar un grave problema de desbalanceo, el cual puede afectar muy negativamente a los resultados de nuestros modelos.

Para corregir esto, una de las mejores soluciones que podemos encontrar es aplicar técnicas de *Data Augmentation*, las cuales consisten en reducir o incrementar el número de ejemplos de las clases afectadas de modo que consigamos una distribución uniforme.

Este tipo de técnicas es comúnmente usado en problemas de aprendizaje profundo donde los datos de entrada son imágenes a las cuales se les aplica ciertas transformaciones de forma que de una misma imagen consigamos extraer otras tantas de características parecidas, pero no iguales (Shah, 2022).

2.6.3 Modelaje de Soluciones de Aprendizaje Automático

Una vez hemos terminado la etapa de preprocesamiento debemos aplicar los algoritmos de aprendizaje automático para conseguir aprender a diferenciar los textos machistas de los que no lo son. Para esto, utilizaremos el *dataset* de tweets que vienen clasificados como machistas o no dependiendo del contenido.

Este *dataset* lo hemos obtenido del *paper* "Call Me Sexist… But" (Samory, Sen, Kohne, Flöck, & Wagner, 2021), el cual ya hemos mencionado antes. En este *paper* realizan una clasificación de los textos en función de la categoría de estos, distinguiendo entre textos de "*Behavioral Expectations*", "*Stereotypes & Comparisons*", "*Endorsements of Inequality*" y "*Denying Inequality & Rejection of Feminism*", obteniendo una puntuación F1 de aproximadamente un 61%, sirviéndonos como primer desafío a superar con nuestros algoritmos.

El *dataset* que hemos obtenido de este *paper* (link) se trata de una versión reducida del original usado en el mismo, tal y como podemos ver en la figura 27. Estos datos se han creado a partir de encuestas y páginas relacionas con el tema, teniendo como resultado una fuente de datos de calidad y de interés para ser estudiados.

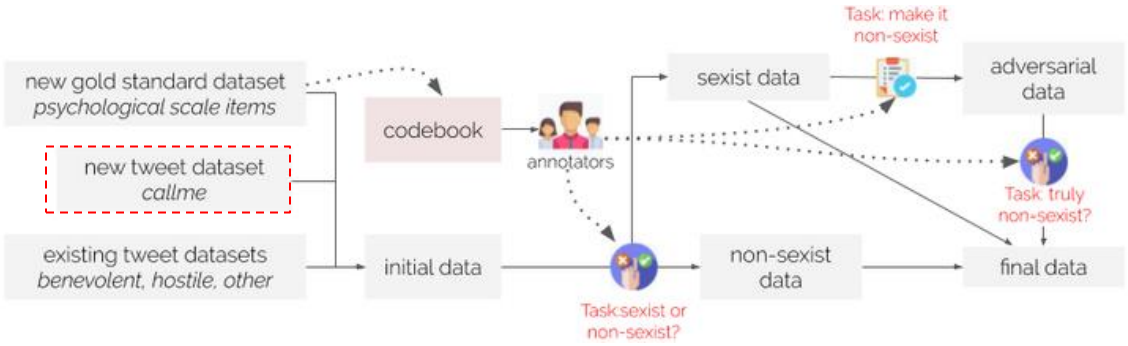


***Figura 26**. Resumen del paper Call Me Sexist... But*

Para la realización de nuestro proyecto vamos a descartar el uso de algoritmo de *Deep Learning* (DL) por su falta de transparencia, ya que parte de los resultados que queremos mostrar es el tipo de características que utiliza el algoritmo para clasificar los textos, siendo parte importante del análisis que finalmente mostraremos al usuario.

En el capítulo 2.2. realizamos una introducción de los diferentes tipos de algoritmos de aprendizaje automático que existen, de los cuales, basándonos en los *papers* estudiados que tratan problema parecidos a los nuestros, elegiremos tres para realizar los análisis necesarios. Estos algoritmos son Random Forest Classifier, Logistic Regression y GaussianNB, todos utilizados a partir de la librería de scikit-learn.

Como veremos posteriormente en el capítulo 3.2.3., los algoritmos que finalmente hemos escogido son Random Forest Classifier y Logistic Regression, descartano el modelo Gausiano debido a los resultados. Vamos a definir estos dos de forma más específica:

- **Random Forest Classifier:** Este tipo de algoritmos se basan en el principio en los árboles de decisión, realizando varios árboles de decisión cogiendo características aleatorias del conjunto de datos, obteniendo un resultado final a partir de los resultados de los subárboles de decisión (Figura 28).

  Los árboles de decisión son un tipo de algoritmo que va dividiendo el espacio de predictores según las observaciones que va recibiendo de entrada, construyendo subdivisiones cada vez más profundas conforme más ejemplos van entrando en el conjunto, siendo fácil que se produzca un sobre entrenamiento del modelo (Ferrero, 2020).

  Para evitar esto normalmente se aplican técnicas para controlar el entrenamiento excesivo, como por ejemplo la poda de los nodos más lejanos al nodo raíz (más profundos), eliminando así las hojas que no aportan nada a la precisión del modelo.

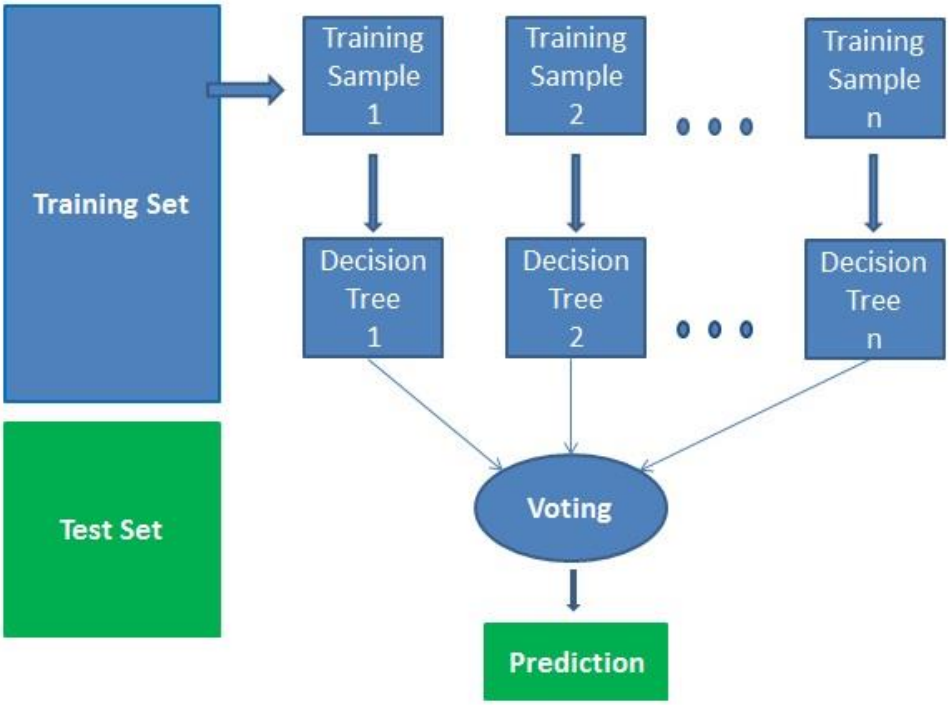


Figura 27. *Diagrama Random Forest Classifier*

- **Logistic Regression:** Los algoritmos de regresión están basados en técnicas matemáticas que busquen, a partir de los datos de entrada, un conjunto de ellas que sean capaces de acertar la salida correspondiente. Esto lo realizan asociando un peso a cada una de las características de los datos de entrada (columnas), los cuales se multiplican por el valor que tengan cada una de ellas, consiguiendo así un resultado final de salida (Edgar & Manz, 2017).

En nuestro caso en concreto, al tratarse de regresión logística, el algoritmo se apoya en la función logística (Figura 29), pudiendo usar tanto valores positivos como valores negativos para los pesos, teniendo un comportamiento diferente a otros algoritmos de regresión, como la regresión lineal, los cuales solo pueden ser usados para predecir clases lineales.

Este tipo de algoritmos se caracterizan por la simplicidad y la potencia que tienen frente a problemas con multitud de datos de entrada (como es nuestro caso), siendo una opción muy interesante para nuestro problema.

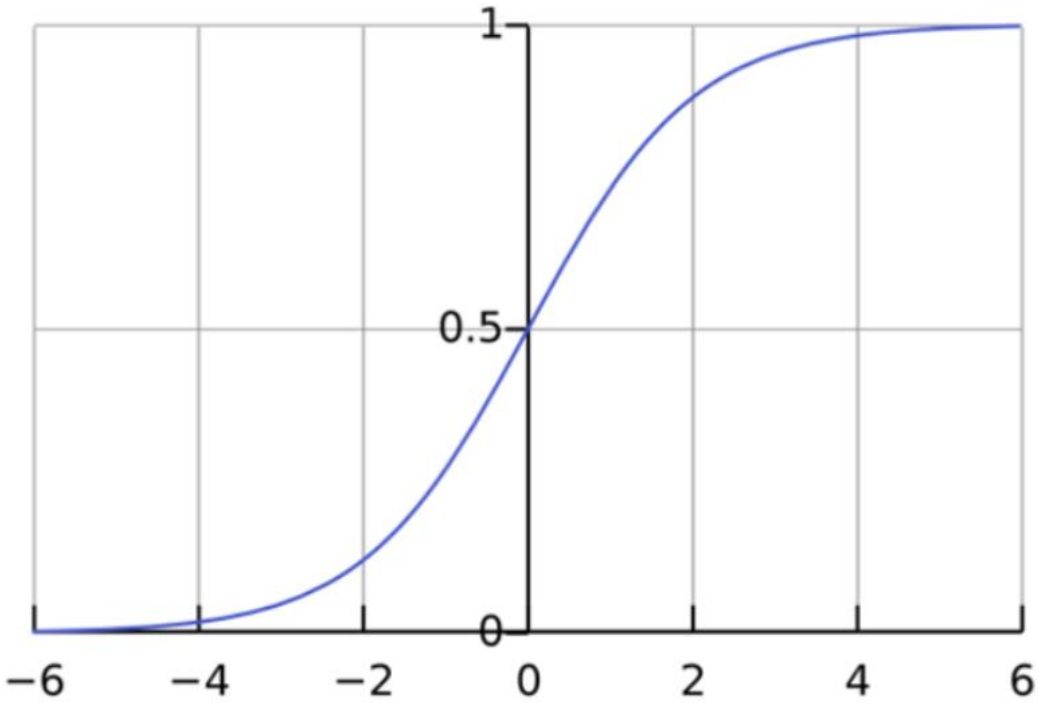


Figura 28. *Función Sigmoide*

Como extra a la hora de entrenar los modelos vamos a utilizar las librerías de scikit dedicadas tanto a la división de los datos de entrada en entrenamiento y validación (train_test_split) como a la búsqueda de hiperparámetros (GridSearchCV), siendo dos recursos de gran utilidad para conseguir elaborar los mejores modelos posibles ante nuestro conjunto de datos.

Con los modelos de aprendizaje automático dedicados al análisis de textos machistas definidos, solo nos falta conocer el tipo de herramienta a usar para realizar el análisis de sentimientos de los textos.

Este tipo de análisis se apoya casi en su totalidad al uso de la librería nltk, la cual ya hemos introducido en el apartado 2.6.2 del documento y que sirve para realizar multitud de procesamiento diferente a los textos, desde categorizar las palabras por tipo (adjetivos, nombres…) hasta la división por frases de las propias cadenas de textos.

Esta librería forma la base de otra aún más potente llamada TextBlob, que haciendo uso de las diferentes funciones de esta consigue obtener resultados muy interesantes de los textos de entrada, como por ejemplo la subjetividad y la objetividad de estos.

2.6.4 Representación de los Datos (GUI).

En el capítulo 2.3.4 realizamos una pequeña introducción a las dos librerías que principalmente nos van a servir para conocer los resultados de nuestros modelos, así como para la representación final de nuestro proyecto de cara al usuario final.

La primera de ellas es matplotlib, una de las librerías más conocidas y usadas en toda la comunidad de Python por su gran versatilidad y por la multitud de gráficas que tiene integrada en su código. Esta en concreto se apoya mucho en las figuras que puedes generar con Matlab (de ahí su nombre), por lo que es una gran ventaja conocer un poco del lenguaje de Matlab para su uso.

Con el uso de esta librería podemos representar desde las distribuciones de las clases por el conjunto de datos hasta los resultados del modelo entrenado, siendo una opción de gran interés a la hora de mejorar el modelo conforme a estos datos.

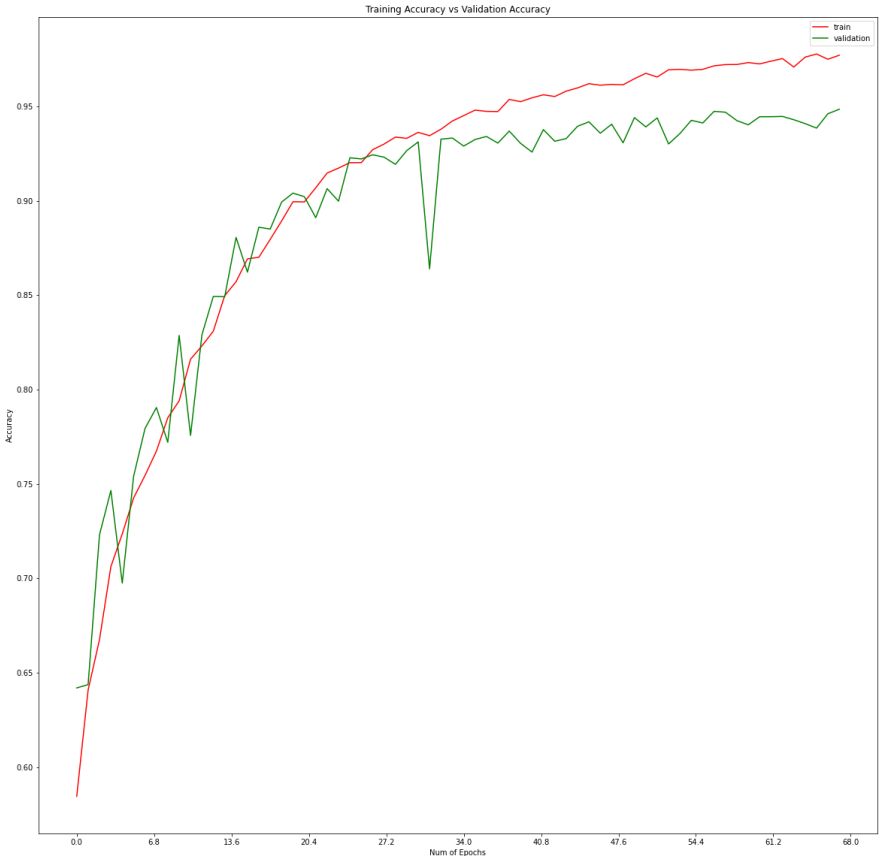


Figura 29. *Ejemplo de gráfica realizada con MatplotLib*

La segunda librería que vamos a usar tiene un cometido completamente diferente, siendo una herramienta dedicada a la creación de interfaces gráficas para los usuarios, haciendo que nuestro trabajo se vea más atractivo y por lo tanto más presentable.

Para realizar esto nos hemos apoyado en la librería tkinter, la cual viene incluida con la propia instalación de Python, por lo que no es necesario tener instalado ninguna herramienta extra para usarla.

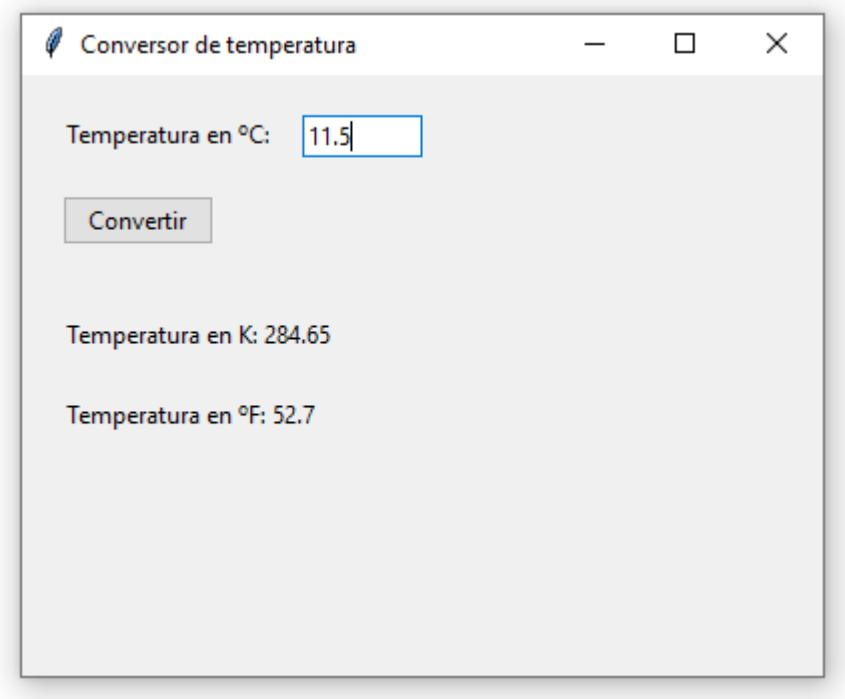


***Figura 30**. Ejemplo de interfaz en Tkinter*

Con la tecnología de nuestra última parte del proyecto definida, podemos pasar a las diferentes ventajas que nos han dado usar este tipo de tecnologías respecto a otras, terminando de justificar un poco más la elección de estas en la realización de nuestro TFM.

Capítulo 3
Planteamiento del Problema

Tras haber puesto en contexto los diferentes puntos que vamos a tratar a lo largo de este proyecto, vamos a comenzar con la justificación de las diferentes tecnologías que han sido utilizadas en la realización de este trabajo.

En esta sección definiremos con nombre y apellidos todas las herramientas que hemos ido utilizando en las diferentes fases del proyecto, las cuales han sido definidas en capítulos anteriores y que nos permitirán mantener la continuidad de este documento.

3.1 Diagrama del Proyecto

Durante la realización del trabajo hemos trabajado con diferentes herramientas dedicadas cada una a un tipo diferente de computo, usando cada una en la fase correspondiente.

Como introducción previa a la justificación de cada una de ellas, vamos a realizar un esquema que sirva de resumen a todas las tecnologías expuestas tanto en las definiciones del capítulo 2.2 y 2.3. , como en las justificaciones del capítulo 3.2. . Dividiremos las diferentes tareas en los subojetivos propuestos en el capítulo introductorio.

I

Análisis de Textos de la Web (Web Scraping):

- BeautifulSoup

Procesamiento de textos:

- TextAttack
- Pandas / Numpy
- Nltk / Re / String
- Scikit-Learn -> *Vectorizers*
- ~~Imbalacend Learn~~ -> ~~*RandomOverSampler*~~

II

III

Aplicaciones Machine Learning:

- Scikit-Learn -> ~~*Gaussian*~~ *RandomForest* *LogisticRegression* *GridSearchCV*
- Matplotlib
- TextBlob -> *Sentiment Analysis*

Visualización de datos:

- Tkinter
- Matplotlib

IV

Diagrama 1. *Resumen de las diferentes tecnologías aplicadas a cada fase*

3.2 Tecnologías Elegidas

Una vez visto el diagrama resumen de las tecnologías utilizadas en cada una de las fases, vamos realizar un repaso por las diferentes fases que dividen nuestro proyecto justificando las tecnologías utilizadas en cada una de ellas.

Usaremos como plantilla los diferentes puntos expuestos en el capítulo 2.6., donde hicimos una definición de todas las tecnología que mencionaremos a continuación.

3.2.1 Análisis de Textos de Internet (Web Scraping)

La tecnología usada en esta fase ha sido la incluida por Beautiful Soup, la cual cuenta con multitud de procesado de texto de interés para nuestro problema, siendo la mejor opción posible para el tratamiento que queremos hacer de los textos.

Una de las principales ventajas de esta librería frente a otras es la cantidad de información que podemos encontrar en la web, encontrando de forma rápida soluciones a los diferentes problemas a hacer frente durante esta fase. Disponer de una guía en la propia página oficial de la librería (Richardson, 2015), junto a una gran cantidad de tutoriales de personas ajenas a esta (Paruchuri, 2021), aplanan en gran medida la curva de aprendizaje que supone usar una nueva tecnología.

Disponer de este tipo de librería perfectamente implementada con el lenguaje Python, junto a la gran cantidad de información que la rodea, han hecho que nos decantemos ante el uso de esta solución para la extracción de textos de la web elegida (*Santa Barbara Independent*, Figura 20), la cual cuenta con una estructura perfecta para el uso del *Web Scraping*.

Analizando la estructura de la web, dimos con una etiqueta común en todas sus noticias para referenciar tanto al título como a los diferentes párrafos que forman la noticia, siendo un tratamiento casi directo la extracción de textos, consiguiendo la información relevante de forma casi instantánea, pudiendo conseguir así una solución interpretable, rápida y eficiente.

Figura 31. *Logo del periódico usado. Santa Barbara Independent.*

3.2.2 Procesamiento de Datos

Durante esta fase nos apoyaremos en el uso de librerías referidas con el preprocesado de datos de textos, donde brillan la librería NLTK y los diferentes métodos de Scikit-learn para vectorizar las cadenas de texto entrantes.

Como complemento a estas, utilizaremos librerias propias de Python como re (regular expressions) o string, ambas de gran utilidad por las numerosas funciones que podemos implementar a partir de ellas, como por ejemplo, el limpiado de caracteres de los textos.

Para la creación de dataframe y manejo de datos dentro de ellos usaremos las clásicas funciones de pandas y numpy, siendo las más eficaces y las que más se usan en todos los proyectos de este estilo.

La librería nltk cuenta con multitud de documentación referida a la misma, consiguiendo empezar su uso de manera sencilla y rápida, consiguiendo ver buenos resultados desde el comienzo (Johnson, 2022).

Esta última, usandola junto a la lematización incluida por la misma, consiguen que podamos realizar el preprocesamiento de texto al completo, obteniendo un resultado mucho más interpretable y de mucha mejor calidad para usarlo como entrada a nuestro modelo de aprendizaje automático.

Con respecto a los vectorizadores, ambos en principio parecen dar buenos resultados en los ejemplos encontrados en la web, siendo muy depedientes de problema en cuestión a analizar.

Realizando pruebas con los algoritmos de aprendizaje automático elegidos (expuestos en el capítulo 3.2.3) de estos dos transformadores, pudimos apreciar como al usar el segundo obteníamos resultados ligeramente mejores con respecto a Count Vectorizer:

	Count Vectorizer (300 features)		TFIDF Vectorizer (300 features)	
	Logistic	Random F.	Logistic	Random F.
Precisión clase 1	0.89	0.89	0.91	0.93
Precisión clase 0	0.82	0.84	0.86	0.85
Recall Clase 1	0.85	0.81	0.85	0.84
Recall Clase 0	0.92	0.92	0.91	0.94
Precisión	0.8636	0.8721	0.8827	0.8877

Tabla 1. *Comparación de Vectorizadores*

Como ya hemos mencionado anteriormente en el capítulo 2.6.2., la mejor opción para realizar un preprocesamieto de textos es el uso del aprendizaje automático, donde una de las mejores librerias existentes es textattack, la cual cuenta con una gran cantidad de aumentadores de datos de diferente índole, pudiendo encontrar uno especialmente bueno para nuestro proyecto.

A modo comparativo de esta técnica, usamos un *data augmenter* de la librería imbalanced learn llamado RandomOverSampler. Este genera ejemplos aleatorios a partir de las muestras generadas, por lo que trabaja sobre los datos ya procesados por los vectorizadores. Los resultados fueron comparados en el algoritmo de *Random Forest* (1000 *features*) usando el vectorizador TFIDF.

	RandomOverSampler	TextAttack
Precisión clase 1	0.50	0.93
Precisión clase 0	0.90	0.85
Recall Clase 1	0.38	0.84
Recall Clase 0	0.94	0.94
Precisión	0.860	0.8877

Tabla 2. *Comparación de Data Augmenters*

Podemos comprobar como usando el *augmenter* aleatorio no conseguimos solucionar los problemas de la clase positiva, pues el uso de este tipo de técnicas con datos extraídos de textos parece no dar buenos resultados, mientras que usando las técnicas de *Deep Learning* se consiguen resultados prometedores.

Estos datos comprueban la eficacia de esta librera en este tipo de problemas tan complejos, quedándose como opción final a usar en nuestro trabajo, solucionando en parte el problema de desbalanceo de nuestras clases.

3.2.3 Modelaje de Soluciones de Aprendizaje Automático

Los tres algoritmos que vamos a comparar son Random Forest, Logistic Regression y Gaussian, todos a partir de la implementación que cuenta la librería scikit-learn de ellos.

Para elegir entre estos tres realizamos una prueba sobre los datos sin aumentar, usando la misma configuración que usaban en estos *papers*, teniendo 1000 *features* para Random Forest y la configuración por defecto para los otros dos. En todas las pruebas han sido realizadas con el vectorizador TFIDF :

	Random Forest	Logistic Regression	Gaussian
Precisión clase 1	0.67	0.62	0.21
Precisión clase 0	0.92	0.91	0.97
Recall Clase 1	0.52	0.41	0.91
Recall Clase 0	0.96	0.96	0.42
Precisión	0.8944	0.8875	0.89

Tabla 3. *Comparativa Algoritmos*

Realizando un estudio de resultados acabamos optando por los dos primeros algoritmos, ambos compartiendo la baja precisión de los datos de la clase minoritaria, la cual se ha mejorado usando las técnicas de preprocesamiento de datos.

Esta elección se ha realizado también debido a la naturaleza descriptiva de estos dos algoritmos, pudiendo ver de forma clara las palabras usadas para la clasificación de lo textos de entrada, tal y como hemos mencionado antes.

En nuestro proyecto aplicaremos ambas técnicas a sendos algoritmos de la forma que, la información obtenida por cada uno de ellos dependa únicamente del tipo de solución que estemos aplicando en un momento u en otro. Esto resulta muy interesante de cara a la presentación y estudio de los textos de la web, ya que, al conocer las características que usan cada uno de los modelos, podemos aplicar el que más nos convenga según la situación.

La elección final de los hiperparámetros se hará a lo largo de la experimentación con la función GridSearchCV, la cual hemos usado a lo largo de las prácticas de las asignaturas del master y de la que más conocimiento tenemos, pudiendo conseguir mejores resultados con la misma.

Para comparar los modelos y conocer los datos de ellos usaremos matplotlib, una de la librerías gráficas más conocidas y más usadas por la comunidad de Python. Su parecido a las figuras generadas por Matlab es una de las características que nos han hecho decantarnos por ella, ya que contaba con conocimientos avanzados de Matlab.

Por último, y no por ello menos importante, para el análisis de sentimientos hemos optado por las funciones que incluye texblob. Esta permite realizar un estudio de los textos de forma rápida y sencilla, aportando una solución factible y compaginable al resto del trabajo realizado.

3.2.4 Representación de los Datos (GUI).

Para esta ultima fase usaremos la librería tkinter que, como ya hemos mencionado anteriormente, se incluye en la instalación base de Python, evitando la instalación de software extra que puede resultar engorroso.

Esta incluye funciones que nos permiten crear una GUI donde poder representar todos los datos de interés para nuestro proyecto. En nuestro caso nos interesa crear un lugar donde el usuario pueda interaccionar con nuestros modelos de aprendizaje automático, pudiendo insertar cualquier noticia de la web elegida y obteniendo resultados a partir de ella.

En ella hemos incluido el uso de las gráficas generadas anteriormente en matplotlib, consiguiendo mostrar a partir de Images (tipo de Python) los resultados obtenidos de los modelos.

Al haber tratado librerías parecidas en Java para la creación de interfaces gráficas, la curva de aprendizaje no ha sido demasiado elevada, permitiéndonos indagar un poco más profundo en las diferentes características de las misma.

Esto último unido a la cantidad de documentación con la que cuenta esta librería nos ha hecho decantarnos por ella definitivamente, consiguiendo implementar una interfaz gráfica desde el desconocimiento en poco tiempo.

Capítulo 4
Experimentación

Tras las definiciones de las diferentes tecnologías probadas, vamos a pasar al capítulo más técnico de la memoria. Como paso previo a toda la experimentación realizada, vamos a repasar tareas tales como la elección de una interfaz de trabajo, la instalación de paquetes o la búsqueda de datos de calidad, los cuales, como ya hemos mencionado anteriormente, suponen la diferencia entre el éxito y el fracaso.

Esta primera fase servirá de enlace directo al subcapítulo dedicado a la metodología, donde ejemplificaremos y mostraremos el uso de todas las tecnologías introducidas en el capítulo 3.1., desarrollando el código utilizado en la creación de nuestro trabajo.

4.1 Pasos Previos

Esta etapa es una de las que más desapercibida pasa a la hora de exponer un trabajo, dando por hecho muchas veces fases del proyecto que pueden ser perfectamente no triviales. Durante esta fase definiremos los puntos más importantes para tener en cuenta a la hora de realizar el proyecto, infraestructuralmente hablando.

Todo el proyecto ha sido creado con el lenguaje Python, uno de los lenguajes de programación que más uso y versatilidad tienen en todo el mundo. Es usado por multitud de empresas para realizar multitud de funcionalidades diferentes.

Dado al éxito que tiene, no es difícil encontrar una gran cantidad de librerías con las que realizar diferentes tipos de funciones, añadiéndole aún más valor gracias a la comunidad activa que hay detrás de él.

Estos dos factores junto al hecho de que es uno de los lenguajes predominantes en el mundo del aprendizaje automático han hecho que sea la mejor alternativa para la realización de este proyecto.

Para la utilización de este vamos a usar dos entornos diferentes, uno ubicado en la nube con el uso de Google Colaboratory (Figura 32a), y otro instalado local con el uso de Anaconda y Visual Studio Code (Figura 32b).

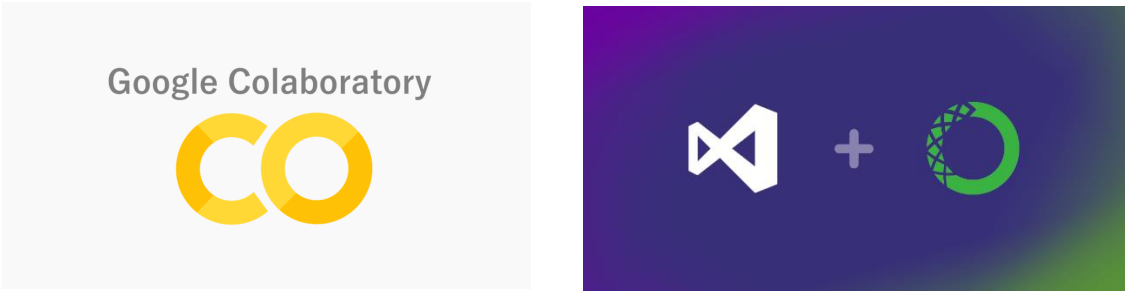


Figura 32. *Entornos utilizados*

El entorno en la nube es perfecto para realizar tareas donde es necesario una gran potencia de cómputo (como el entrenamiento de modelos), por lo que el uso de *Colaboratory* ha sido de gran ayuda tanto a la hora de realizar el *Data Augmentation*, el cual recordamos hacía uso del aprendizaje automático, y de entrenar nuestro propio modelo.

Nuestro entorno local ha sido montado para crear un programa final con la interfaz que englobe todo el trabajo realizado hasta conseguir los resultados finales. La constitución de este vendrá especificada en el Anexo I, donde estarán expuestas las librerías necesarias para la ejecución de nuestro programa.

Con nuestros dos entornos creados solo nos hace falta preparar cada uno de ellos para su uso a lo largo de la realización del trabajo, consiguiendo evitar problemas futuros que puedan dar por falta de librerías o fallos a nivel computacional por falta de recursos.

Google Colaboratory incluye en cada uno de sus cuadernos preinstaladas la mayoría de las librerías que hemos mencionado anteriormente, por lo que su uso ha sido casi instantáneo, teniendo únicamente que instalar la librería utilizada para el balanceo de las clases, la cual se instala utilizando el comando pip en el propio cuaderno.

```
!pip install textattack[tensorflow]
```

Utilizar este tipo de cuadernos para realizar tareas de alta complejidad computacional es ideal si no se dispone de un equipo de buena calidad. Google pone a su disposición entornos virtuales con tarjeta gráfica de gran potencia que agilizan este tipo de procesamiento, consiguiendo los mismos resultados en menos tiempo.

Para el uso del entorno local es necesario instalar un entorno con Python incorporado. En nuestro caso hemos utilizado el anteriormente mencionado Anaconda, ya que es uno de los que más uso he dado a lo largo de la carrera y el máster, teniendo un manejo superior frente a otras alternativas.

Una vez instalado este entorno se creó un espacio dedicado al TFM, donde se han instalado todas las librerías expuestas en el Anexo I, consiguiendo así un entorno listo para la creación y ejecución del programa.

Como entorno de programación se ha usado Microsoft Visual Studio Code, un editor de textos que contiene multitud de *Plugins* que pueden ser usados para facilitar la programación en el lenguaje que sea necesario, en nuestro caso únicamente Python.

Instalando las extensiones de Python y de Jupiter podemos empezar a programar en el entorno creado anteriormente, evitando errores por falta de compatibilidad y teniendo un entorno seguro de trabajo, pues este tiene opción de autoguardado para el caso que se produzca un error.

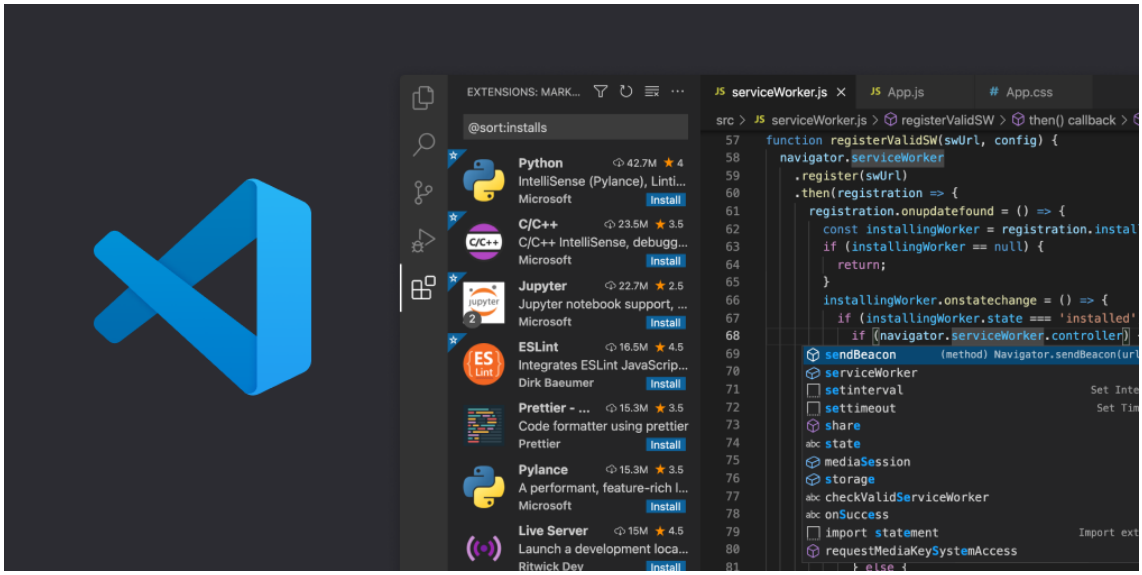


Figura 33. *Visual Studio Code*

4.2 Metodología

Tras tener creado un entorno de trabajo y tras haber estudiado las diferentes tecnologías implicadas en el proyecto, vamos a comenzar con el trabajo práctico que hemos seguido a lo largo del proceso de realización del proyecto, haciendo especial mención a los resultados obtenidos y dificultades encontradas.

Este subcapítulo a su vez lo dividiremos en las diferentes fases en las que hemos dividido la construcción del trabajo, haciendo un resumen de las diferentes etapas por las que hemos ido pasando hasta construir el programa final.

Todos los archivos mencionados a lo largo de este capítulo estarán disponibles en el GitHub del proyecto, donde podrá consultarse de manera más concreta todo el código utilizado.

4.2.1 Primera Versión del Modelo

Antes de comenzar el modelaje hay que realizar un estudio de viabilidad del trabajo para conocer el tipo de problema que estamos tratando y las limitaciones que este tiene. Estas primeras pruebas han sido realizadas en el notebook "PrimeraAproximacion.ipynb".

El primer paso evidente para realizar este estudio es construir un clasificador básico el cual extraiga las características de nuestro *dataset* y nos de unos resultados para poder analizarlos y actuar con respecto a ellos.

Esta primera fase es crítica para el trabajo, pues si lo expuesto en puntos anteriores no diera unos datos fiables o no entrara dentro de los límites establecidos deberíamos replantear el trabajo y comenzar con otra perspectiva, siempre manteniendo presentes los objetivos marcados en el proyecto.

Para realizar la construcción de este primer modelo es obligatorio realizar un procesamiento de datos al *dataset* conseguido, pues este cuenta con multitud de tweets sin filtrar y con una gran cantidad de ruido que no solo no aporta nada al modelo, si no que entorpece el entrenamiento de este (Figura 34).

```
df = pd.read_csv("Datasets\sexism_data.csv")
df.head(10)
```
✓ 0.1s

	id	dataset	text	toxicity	sexist	of_id
0	0	other	MENTION3481 i didn't even know random was an option!	0.118180	False	-1
1	1	other	Bottom two should've gone! #mkr	0.251850	False	-1
2	2	callme	MENTION3111 MENTION3424 ladyboner deserves so much more credit than dudeboner. #bonerdebate #reddetails	0.113331	False	-1
3	3	other	She shall now be known as Sourpuss #MKR #KatAndre #FailedFoodies	0.531153	False	-1
4	4	other	Tarah W threw a bunch of women under the bus so she could get Wadhwa's support for her Women in Tech book.	0.118718	False	-1
5	5	callme	I just don't trust an adult who uses coupons.	0.119590	False	13227
6	6	other	Not again #MKR	0.198415	False	-1
7	7	other	#MKR2016 returns in 2020 once the all the couples, intruders, gate crashers and second-chancers are eliminated MENTION786 #MKR #MKR2015	0.447262	False	-1
8	8	other	MENTION526 most abuse comes from gamergate accounts that are 30-90 days old.	0.254602	False	-1
9	9	other	Great to see the local National Park workers tucking into a free feed. How about you empty those loo's instead. #MKR	0.392942	False	-1

***Figura 34**. Dataset en Bruto*

Viendo el *dataset* parece evidente intuir que las únicas columnas que son de nuestro interés son el texto (entrada) y la etiqueta *sexist* (salida), pudiendo obviar el resto de las columnas para conseguir la mejor implementación posible con estas dos, pues durante la experimentación solo dispondremos de una cadena de caracteres a predecir.

Con esto en cuenta y habiendo estudiado el preprocesamiento necesario que necesita un texto de este tipo, pasamos con la construcción de una función que nos ayude con la limpieza de los textos. Como esta función es crítica optamos por realizar una aproximación final de la misma, pues tendríamos que usarla en cualquiera de las situaciones posibles del trabajo. Vamos a dividir este preprocesamiento en subapartados:

I. **Eliminación de caracteres HTML y similares**. Los tweets, al poder contener imágenes, venían con enlaces a estas en el propio campo de texto, por lo que ha sido necesario eliminarlos con el uso de expresiones regulares.

```
title_text = title_text.apply(lambda x:
re.sub(r'(https|http)?:\/\/(\w|\.|\/|\?|\=|\&|\%)*\b', '',
x, flags=re.MULTILINE))
```

Nota: *Dado que la mayoría del preprocesamiento se basa en el mismo principio y para evitar llenar la memoria de código, solo vamos a poner este fragmento de código relacionado a la limpieza de caracteres. Para ver el código completo recomendamos ver el cuaderno completo.*

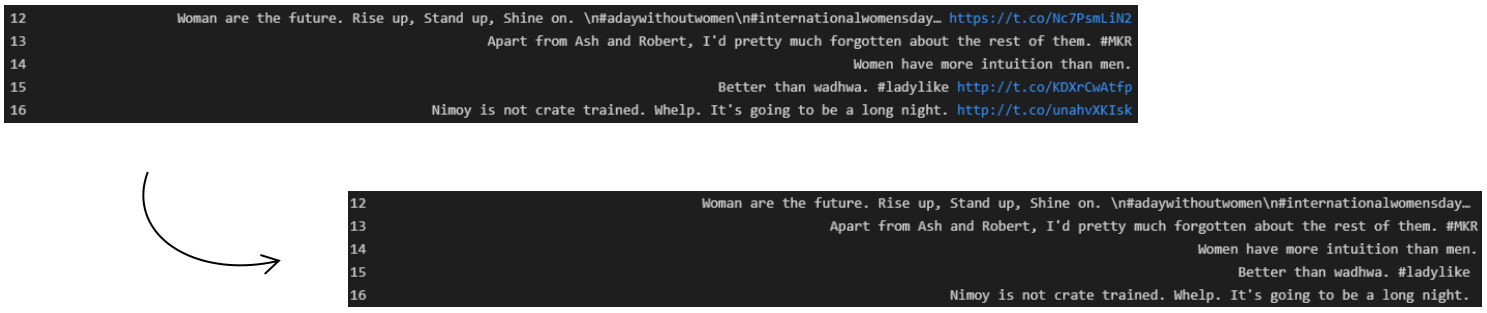


Figura 35. *Preprocesamiento. Paso 1*

II. **Eliminación de menciones.** En este caso las menciones, en lugar de venir precedidas por un "@" vienen expresadas como "MENTION…" para anonimizar los datos, los eliminamos creando la expresión regular correspondiente.

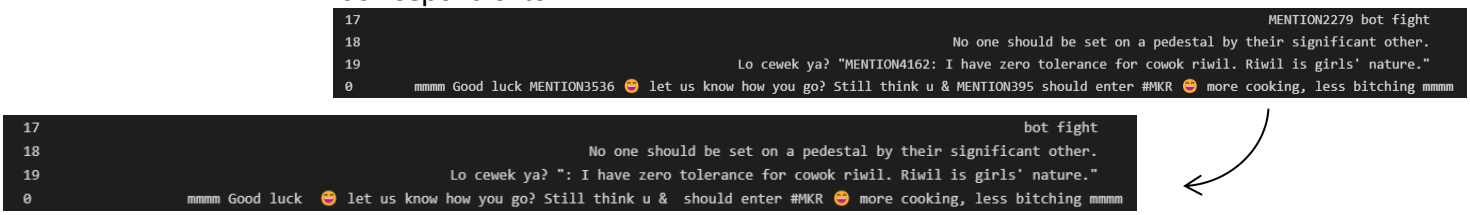


Figura 36. *Preprocesamiento. Paso 2*

III. **Eliminación de emojis.** El ruido más llamativo que podemos encontrar en el texto son los emojis, los cuales se incluyen como caracteres especiales y causan multitud de problemas en la creación de modelos.

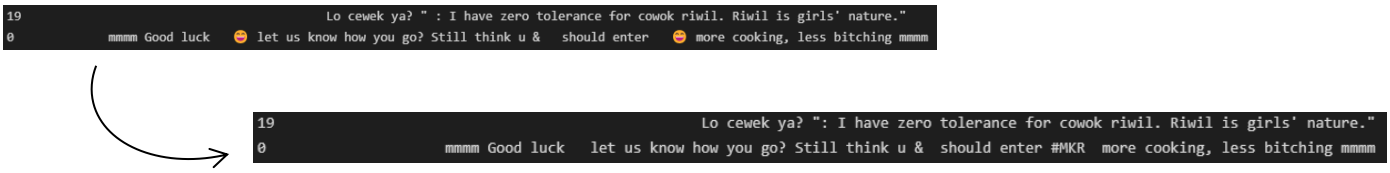


Figura 37. *Preprocesamiento. Paso 3*

IV. Eliminación de hashtags. Otro de los elementos propios de los tweets es el uso de hashtags, los cuales se diferencian por empezar por el carácter '#'.

17	MENTION2279 bot fight
18	No one should be set on a pedestal by their significant other.
19	Lo cewek ya? "MENTION4162: I have zero tolerance for cowok riwil. Riwil is girls' nature."
0	mmmm Good luck MENTION3536 😩 let us know how you go? Still think u & MENTION395 should enter #MKR 😩 more cooking, less bitching mmmm

17	bot fight
18	No one should be set on a pedestal by their significant other.
19	Lo cewek ya? ": I have zero tolerance for cowok riwil. Riwil is girls' nature."
0	mmmm Good luck 😩 let us know how you go? Still think u & should enter #MKR 😩 more cooking, less bitching mmmm

Figura 38. *Preprocesamiento. Paso 4*

V. Eliminación de símbolos de puntuación, números y espacios. Uno de los pasos más importantes en el preprocesado es la eliminación de los números y de los símbolos de puntuación, pues estos no aportan mucho al significado y pueden suponer la creación de muchas palabras extra en el modelado.

Usando tantas técnicas de reemplazo normalmente se crean espacios de más de un elemento y tabulaciones que hay que eliminar, esto lo haremos como último paso antes de realizar las transformaciones sobre los textos.

8	most abuse comes from gamergate accounts that are 30-90 days old.
9	Great to see the local National Park workers tucking into a free feed. How about you empty those loo's instead.
10	All my sons have grown up with computer games but I'm not interested. I see them as a male thing.
11	those are waaaaay more spendy. this is the drive i have. just considering using built-in or if i should go software.
12	Woman are the future. Rise up, Stand up, Shine on. \n\n…

8	most abuse comes from gamergate accounts that are days old
9	great to see the local national park workers tucking into a free feed how about you empty those loos instead
10	all my sons have grown up with computer games but im not interested i see them as a male thing
11	those are w y more spendy this is the drive i have just considering using builtin or if i should go software
12	woman are the future rise up stand up shine on

Figura 39*. Preprocesamiento. Paso 5*

Como podemos apreciar, existe una notable diferencia entre los textos sin procesar y los obtenidos tras este último paso.

VI. Lematización. Habiendo eliminado el ruido más importante de los datos podemos aplicar técnicas más avanzadas de procesamiento. La primera que vamos a usar es la lematización, la cual realizará las transformaciones explicadas en apartados anteriores.

0	i didnt even know random was an option	False
1	bottom two shouldve gone	False
2	ladyboner deserves so much more credit than dudeboner	False
3	she shall now be known as sourpuss	False
4	tarah w threw a bunch of women under the bus so she could get wadhwas suppo for her women in tech book	False

0	i didnt even know random wa an option	False
1	bottom two shouldve gone	False
2	ladyboner deserves so much more credit than dudeboner	False
3	she shall now be known a sourpuss	False
4	tarah w threw a bunch of woman under the bus so she could get wadhwas suppo for her woman in tech book	False

Figura 40. *Preprocesamiento. Paso 5*

VII. **Extracción de características.** Una vez tenemos la cadena de texto en el formato correcto debemos transformar los datos a números tal y como hemos indicado anteriormente.

Para realizar este paso vamos a usar la función TFIDFVectorizer junto al parámetro de las *stopwords* en inglés. Esto ignorará los caracteres que no aportan información relevante a los documentos.

```
X_texto = df['text']
vectorizer = CountVectorizer(max_features=1500, min_df=5,
max_df=0.7, stop_words=stopwords.words('english'))
X = vectorizer.fit_transform(X_texto).toarray()
```

Habiendo conseguido un array de entrada para nuestros modelos de aprendizaje automático, solo nos queda tratar la salida y realizar la división de los datos en un subconjunto de datos de entrenamiento y otro de validación.

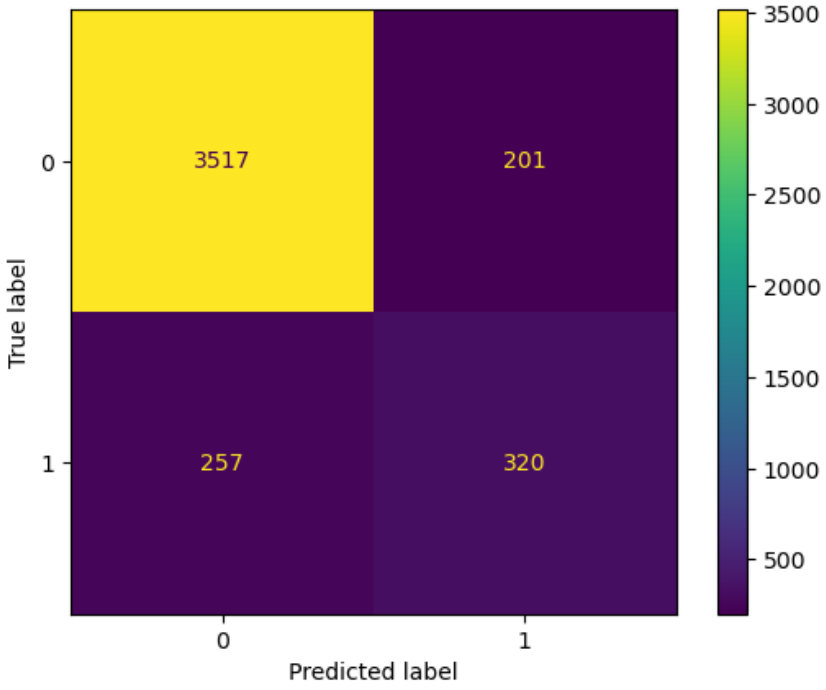
Esta división es necesaria para conseguir unos resultados más cercanos a la realidad, intentando alejar en la medida de lo posible los datos de entrenamiento de los datos evaluados.

```
# Transformación de la salida
y = df['sexist']
y = y.astype(int)
# Train Test Split
X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.3, random_state=0)
```

Con los datos listos para realizar el entrenamiento, vamos a definir los algoritmos definidos en una primera versión para conseguir una primera aproximación del problema:

- Random Forest Classifier

```
classifier = RandomForestClassifier(n_estimators=1000,
random_state=0)
classifier.fit(X_train, y_train)
pred_RF = classifier.predict(X_test)
print(confusion_matrix(y_test, pred_RF))
print(classification_report(y_test, pred_RF))
print('Accuracy Score:', accuracy_score(y_test, pred_RF))
```



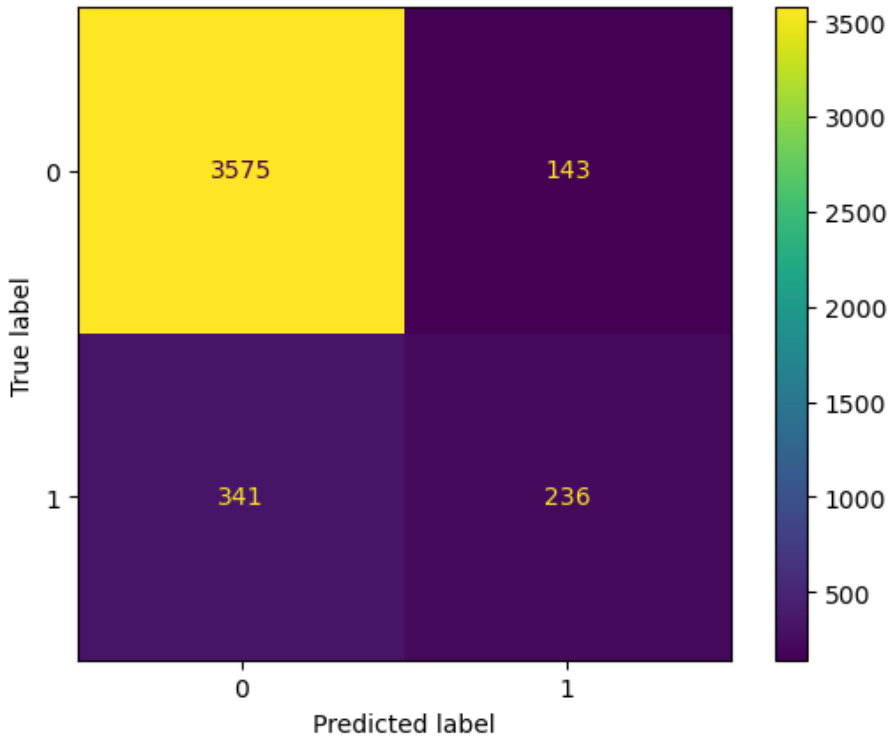
	precision	recall	f1-score	support
0	0.93	0.95	0.94	3718
1	0.61	0.55	0.58	577
accuracy			0.89	4295
macro avg	0.77	0.75	0.76	4295
weighted avg	0.89	0.89	0.89	4295

Accuracy Score: 0.8933643771827706

Figura 41. *Matriz de Confusión y Resultados Random Forest Classifier*

❖ Logistic Regression

```
logreg = LogisticRegression()
logreg.fit(X_train, y_train)
logreg.score(X_test, y_test)
from sklearn.metrics import classification_report
print(classification_report(y_test, logreg.predict(X_test)))
```



	precision	recall	f1-score	support
0	0.91	0.96	0.94	3718
1	0.62	0.41	0.49	577
accuracy			0.89	4295
macro avg	0.77	0.69	0.72	4295
weighted avg	0.87	0.89	0.88	4295

Accuracy Score: 0.8873108265424913

Figura 42. *Matriz de Confusión y Resultados Logistic Regression*

Haciendo un análisis de los resultados obtenidos podemos ver como para la clase negativa obtenemos unos valores de precisión muy cercanos al 100%, sin embargo, para la clase positiva (textos si machistas) obtenemos una puntuación bastante más baja.

Esto se puede constatar si echamos un vistazo a la matriz de confusión, viendo como la fila de la clase positiva tiene unos valores muy parecidos en ambas columnas, representando que el clasificador no sabe clasificar correctamente esta clase.

Si nos fijamos en el número de ejemplos de la clase positiva podemos ver cómo hay muchas menos muestras etiquetadas como clase machista, siendo una de las posibles causas de los resultados que estamos observando.

Figura 43. *Distribución de las clases*

En la figura 43 podemos apreciar mucho mejor este fenómeno, siendo una de las mejoras a implementar en pasos posteriores del modelaje. Aun así, los modelos de prueba han arrojado resultados prometedores, incitándonos a continuar con el proyecto por el camino elegido.

3.3.2 Mejora del Modelo Inicial

Como hemos visto en el subapartado anterior, el desbalanceo de clases es una de las principales limitaciones que tenemos en el *dataset*, siendo necesario corregirlo usando las herramientas definidas en el capítulo 3.1.2 .

Este desbalanceo vamos a corregirlo usando la función WordNetAugmenter, función que genera de forma aleatoria frases intercambiando sinónimos de algunas de las palabras que nos encontramos en el documento. Un ejemplo claro podemos verlo en la figura 44, donde la primera fase corresponde con la original y las otras 3 son generadas por el algoritmo.

```
('i didnt even know random was an option',
 ['i didnt still live random was an choice',
  'i didnt tied bed random was an alternative',
  'i didnt tied screw random was an choice'])
```

Figura 44*. WordNetAugmenter*

Este principio va a ser usado en todas las frases etiquetadas como clase positiva, realizando una inserción de las nuevas frases en un nuevo *dataframe* que usaremos como *dataset* de entrada para los posteriores ejemplos.

Todos estos pasos están expuestos en el cuaderno llamado "BalanceoClases.ipynb", donde se puede apreciar de mejor manera los diferentes pasos seguidos para conseguir elaborar el nuevo *dataset*.

Estos nuevos datos ya han sido previamente tratados hasta el paso V del preprocesamiento de datos, teniendo que aplicar únicamente la lematización y la extracción de características para entrenar los nuevos modelos en el capítulo siguiente.

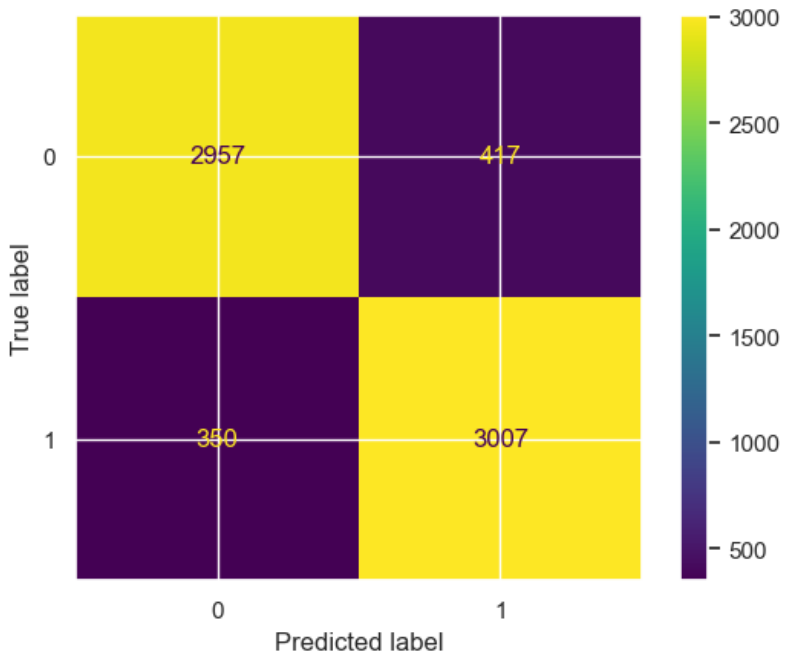
Si pintamos la misma gráfica de distribución con estos nuevos datos (Figura 45) podemos apreciar como ahora sí que contamos un *dataset* balanceado.

Figura 45. *Distribución de las clases. Nuevo Dataset*

Es importante no olvidar que estos datos han sido creados a partir de ejemplos ya existentes, siendo muy probable que la mayoría de los ejemplos tengan una estructura parecida a los ejemplos originales, siendo muy probable que los algoritmos aprendan únicamente de los ejemplos, teniendo que ser cuidadoso con el entrenamiento de próximos modelos.

Si aplicamos este nuevo *dataset* al modelo Random Forest probado en el subcapítulo anterior comprobaremos si este cambio ha supuesto una mejora significativa o no a nuestro proyecto.

```
classifier = RandomForestClassifier(n_estimators=1000,
random_state=0)
classifier.fit(X_train, y_train)
pred_RF = classifier.predict(X_test)
print(confusion_matrix(y_test, pred_RF))
print(classification_report(y_test, pred_RF))
print('Accuracy Score:', accuracy_score(y_test, pred_RF))
```



	precision	recall	f1-score	support
0	0.89	0.88	0.89	3374
1	0.88	0.90	0.89	3357
accuracy			0.89	6731
macro avg	0.89	0.89	0.89	6731
weighted avg	0.89	0.89	0.89	6731

Accuracy Score: 0.8860496211558461

Figura 46. *Matriz de Confusión y Resultados Random Forest Classifier - Balanceado*

Podemos ver como esta mejora ha supuesto subir los valores de la clase positiva, manteniendo muy altos también los encontrados en la clase negativa, consiguiendo así nivelar los resultados obteniendo anteriormente.

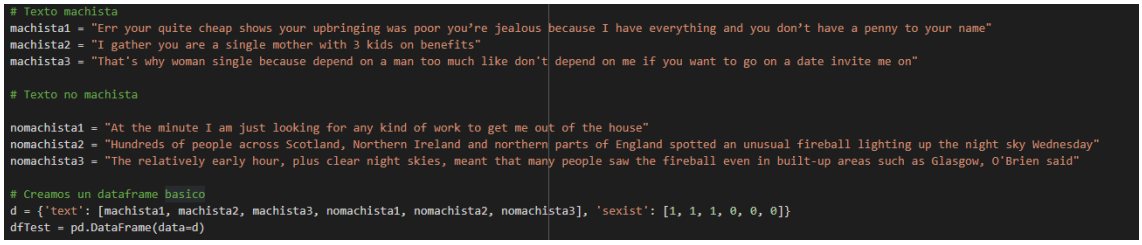
Habiendo confirmado la eficacia de esta mejora podemos continuar con la construcción de los modelos y la implementación de los demás apartados necesarios para la construcción del programa final.

3.3.3 Creación del Modelo Final

Con los datos balanceados y la estructura de modelos definidas, vamos a usar la búsqueda de hiperparámetros para encontrar la mejor combinación de valores para nuestro problema en concreto, quedándonos así con el mejor modelo obtenido. Todos los pasos aquí expuestos se pueden encontrar completos en el *notebook* "ModelosAvanzados.ipynb"

Uno de los primeros parámetros que hemos ido probando ha sido el número de características a generar por el *vectorizer*, comprobando que conforme mayor número de *features* dispongamos, más sobreaprende del modelo, siendo menos flexible ante otro tipo de textos.

Para realizar la búsqueda de hiperparámetros hemos utilizado un número de características de 400, siendo el número que mejores resultados nos ha dado con unos textos de pruebas extraídos de una web:



```
# Texto machista
machista1 = "Err your quite cheap shows your upbringing was poor you're jealous because I have everything and you don't have a penny to your name"
machista2 = "I gather you are a single mother with 3 kids on benefits"
machista3 = "That's why woman single because depend on a man too much like don't depend on me if you want to go on a date invite me on"

# Texto no machista

nomachista1 = "At the minute I am just looking for any kind of work to get me out of the house"
nomachista2 = "Hundreds of people across Scotland, Northern Ireland and northern parts of England spotted an unusual fireball lighting up the night sky Wednesday"
nomachista3 = "The relatively early hour, plus clear night skies, meant that many people saw the fireball even in built-up areas such as Glasgow, O'Brien said"

# Creamos un dataframe basico
d = {'text': [machista1, machista2, machista3, nomachista1, nomachista2, nomachista3], 'sexist': [1, 1, 1, 0, 0, 0]}
dfTest = pd.DataFrame(data=d)
```

Figura 47. *Datos de prueba externos*

Usando un modelo entrenando a partir de 400 características ha sido la única vez que ha detectado uno de los textos como machistas, siendo el que más flexibilidad ha dado con respecto a otras pruebas realizadas con más y menos características.

```
vectorizer = TfidfVectorizer(max_features=400, min_df=5, max_df=0.7
, stop_words=stopwords.words('english'))
X = vectorizer.fit_transform(df['text'])
```

Algo importante para tener en cuenta es la naturaleza de los textos. No es lo mismo enfrentarnos a textos extraídos de una red social, donde se usan muchas expresiones coloquiales, a un texto extraído de un medio de comunicación de noticias, donde el vocabulario y las frases tienen una estructura diferente. Esto supone una de la limitación más importante a la hora de obtener resultados, siendo una de las posibles mejoras que comentaremos en el apartado correspondiente.

Con el *vectorizer* fijado únicamente nos falta generar las búsquedas de rejilla con la función GridSearchCV, cada una usando los hiperparámetros correspondientes a su algoritmo, específicamente los siguientes:

❖ Random Forest Classifier

```
classifierRFC = RandomForestClassifier(random_state=42)


param_gridRFC = {
    'n_estimators': [1000, 1250, 1500],
    'max_features': ['auto', 'sqrt', 'log2'],
    'max_depth' : [6, 8, 10, 12],
    'criterion' :['gini', 'entropy']
}


CV_rfc = GridSearchCV(estimator=classifierRFC, param_grid=param_gri
dRFC, cv= 5)
CV_rfc.fit(X_test, y_test)
```

```
[ ]  CV_rfc.best_estimator_

     RandomForestClassifier(max_depth=12, max_features='log2', n_estimators=1500,
                            random_state=42)
```

Figura 48. *Mejor modelo Random Forest*

❖ Logistic Regression

```
C = np.logspace(-4, 4, 50)
penalty = ['l1', 'l2']


parametersLR = dict(C=C, penalty=penalty)


CV_lg = GridSearchCV(estimator=classifierLR, param_grid=parametersL
R, cv= 5)
CV_lg.fit(X_train, y_train)
```

```
[ ]  CV_lg.best_estimator_

     LogisticRegression(C=0.5689866029018293, max_iter=10000)
```

Figura 49. *Mejor Modelo Logistic Regression*

Teniendo los mejores modelos de cada uno de ellos podemos obtener el peso de las características que más aportan en la decisión del modelo, consiguiendo transparencia en la clasificación dada por el modelo, algo muy importante dado la naturaleza del problema y el tipo de textos que estamos clasificando.

- ❖ Random Forest Classifier Features

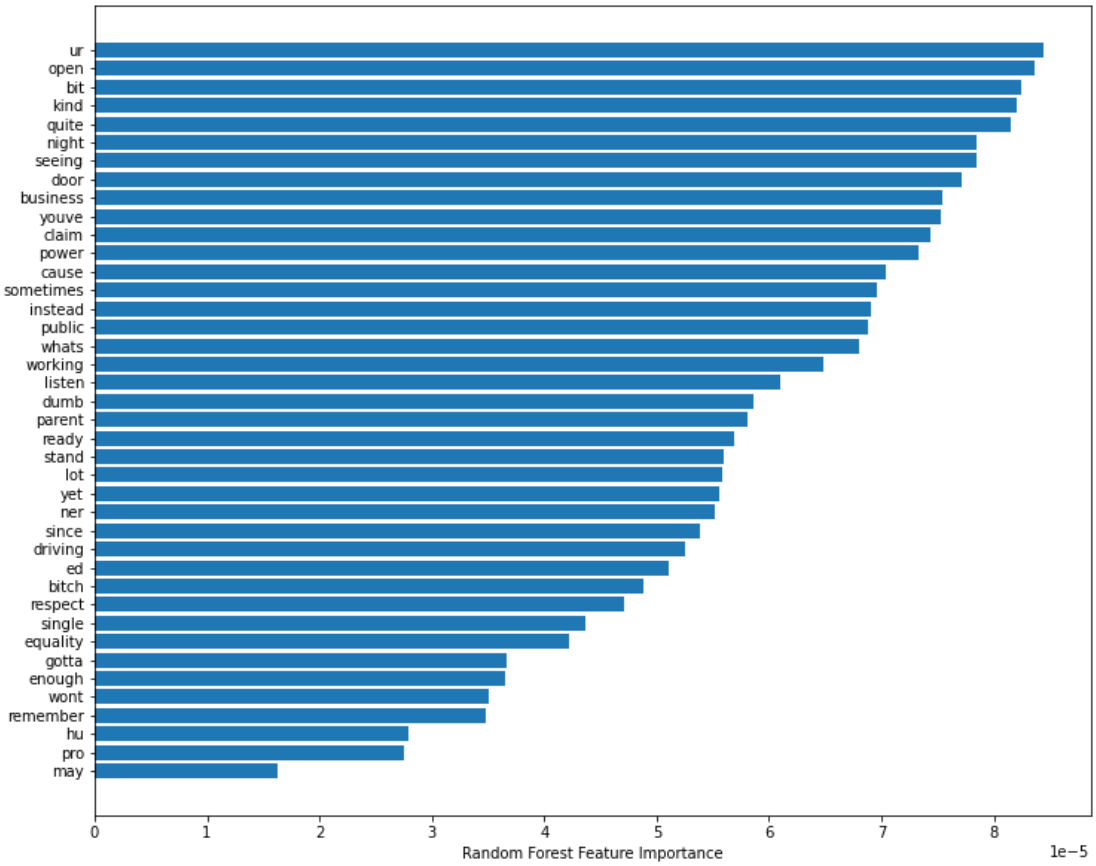


Figura 50. *Peso de las características RFC*

- Logistic Regression Features
  - Clase Positiva

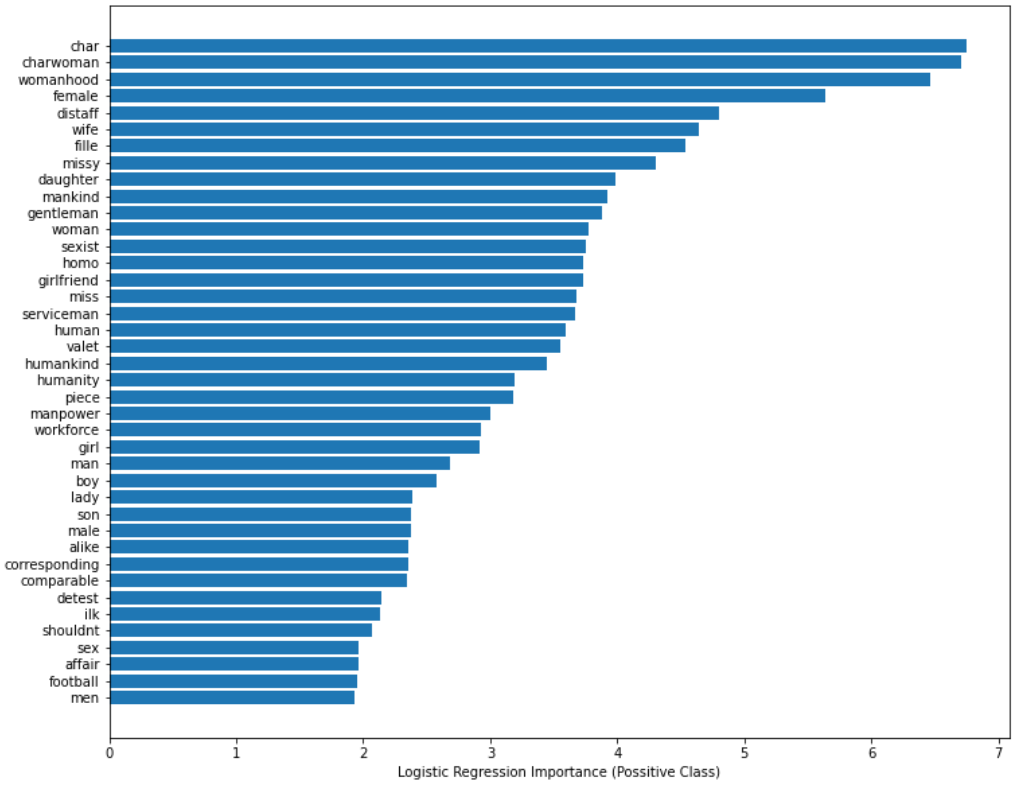


Figura 51. *Peso de las características LR. Clase Positiva*

  - Clase Negativa

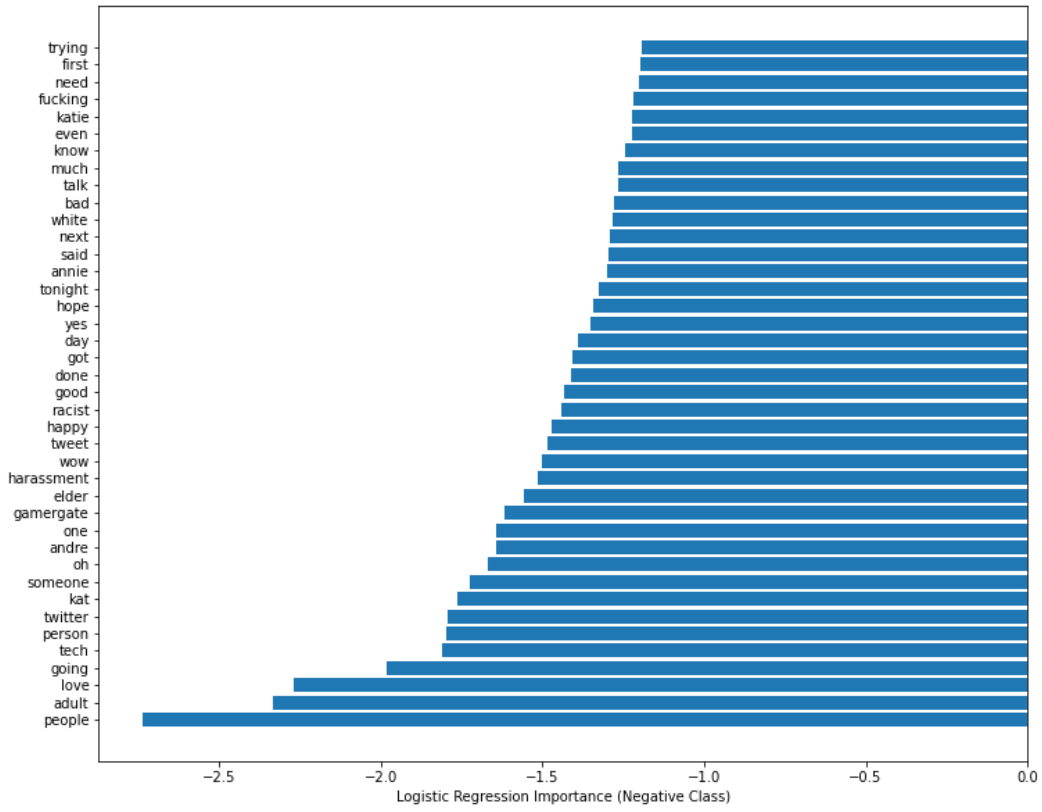


Figura 52. *Peso de las características LR. Clase Negativa*

Conociendo las *features* que busca el modelo para clasificar una clase como positiva o negativa es fácil comprender el interior del modelo, siendo muy útil para la presentación de los datos.

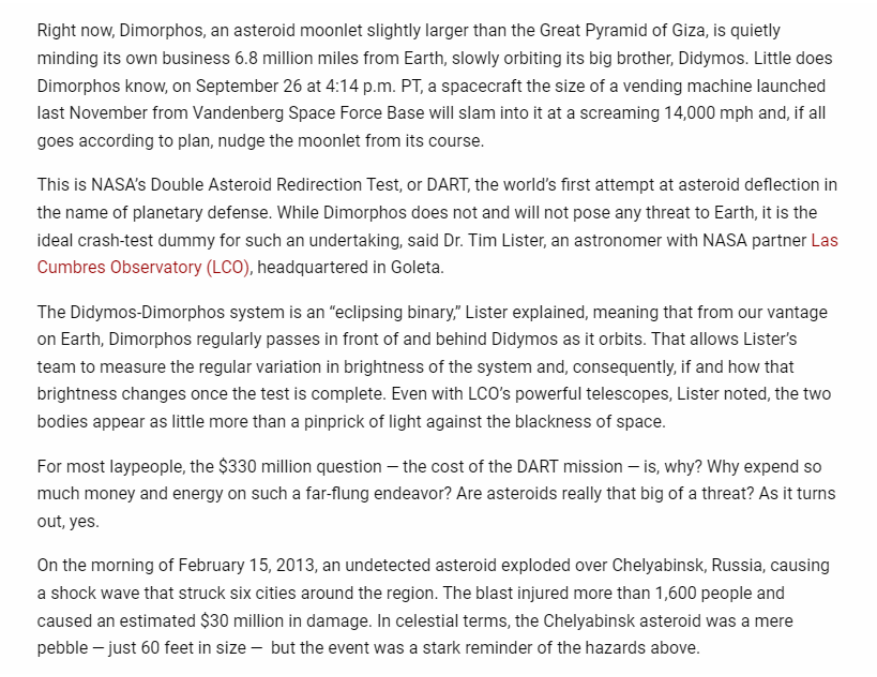
Si nos fijamos expresamente en los datos de las características de la regresión lineal podemos ver cómo tiene sentido que las palabras que más peso aportan a la clase negativa sean palabras genéricas o positivas tales como "*people*", "*adult*" o "*love*"; mientras que para la clase positiva las palabras que más aportan son connotaciones negativas como "*char*", "*charwoman*" o "*female*".

Con estos modelos podemos dar por terminada la fase de experimentación con ellos, pasando directamente a la última etapa del trabajo, la cual completará la fase de obtención de textos y de visualización de datos.

3.3.4 Web Scraping, Análisis de Sentimientos y Creación de GUI

En esta última etapa crearemos nuestro script de Web Scraping, el script para el análisis de sentimientos y el script de nuestra interfaz gráfica que dará vida a nuestro proyecto, consiguiendo una ventana de fácil interpretación para los posibles usuarios que podamos tener.

El primer paso que vamos a tratar en esta sección será la relacionada con la obtención de los textos de la web. Estos textos corresponderán a cualquier noticia que podamos obtener de la web Independent, consiguiendo transformar los párrafos que contiene la noticia (Figura 53) en un *dataframe* de pandas al que aplicar las mismas técnicas de procesado de datos que hemos aplicado a los datos de entrenamiento para predecir las clases usando los modelos preentrenados (Figura 54). Todo este proceso se puede ver al completo en el cuaderno "WebScrapping.ipynb".



Right now, Dimorphos, an asteroid moonlet slightly larger than the Great Pyramid of Giza, is quietly minding its own business 6.8 million miles from Earth, slowly orbiting its big brother, Didymos. Little does Dimorphos know, on September 26 at 4:14 p.m. PT, a spacecraft the size of a vending machine launched last November from Vandenberg Space Force Base will slam into it at a screaming 14,000 mph and, if all goes according to plan, nudge the moonlet from its course.

This is NASA's Double Asteroid Redirection Test, or DART, the world's first attempt at asteroid deflection in the name of planetary defense. While Dimorphos does not and will not pose any threat to Earth, it is the ideal crash-test dummy for such an undertaking, said Dr. Tim Lister, an astronomer with NASA partner Las Cumbres Observatory (LCO), headquartered in Goleta.

The Didymos-Dimorphos system is an "eclipsing binary," Lister explained, meaning that from our vantage on Earth, Dimorphos regularly passes in front of and behind Didymos as it orbits. That allows Lister's team to measure the regular variation in brightness of the system and, consequently, if and how that brightness changes once the test is complete. Even with LCO's powerful telescopes, Lister noted, the two bodies appear as little more than a pinprick of light against the blackness of space.

For most laypeople, the $330 million question — the cost of the DART mission — is, why? Why expend so much money and energy on such a far-flung endeavor? Are asteroids really that big of a threat? As it turns out, yes.

On the morning of February 15, 2013, an undetected asteroid exploded over Chelyabinsk, Russia, causing a shock wave that struck six cities around the region. The blast injured more than 1,600 people and caused an estimated $30 million in damage. In celestial terms, the Chelyabinsk asteroid was a mere pebble — just 60 feet in size — but the event was a stark reminder of the hazards above.

Figura 53. *Texto parcial de una noticia de la web Independent*

0	Right now, Dimorphos, an asteroid moonlet slightly larger than the Great Pyramid of Giza, is quietly minding its own business 6.8 million miles from Earth, slowly orbiting its big brother, Didymos...
1	This is NASA's Double Asteroid Redirection Test, or DART, the world's first attempt at asteroid deflection in the name of planetary defense. While Dimorphos does not and will not pose any threat t...
2	The Didymos-Dimorphos system is an "eclipsing binary," Lister explained, meaning that from our vantage on Earth, Dimorphos regularly passes in front of and behind Didymos as it orbits. That allows...
3	For most laypeople, the $330 million question — the cost of the DART mission — is, why? Why expend so much money and energy on such a far-flung endeavor? Are asteroids really that big of a threat?...
4	On the morning of February 15, 2013, an undetected asteroid exploded over Chelyabinsk, Russia, causing a shock wave that struck six cities around the region. The blast injured more than 1,600 peop...

Figura 54. *Texto transformado a dataframe*

Esto se ha conseguido gracias a la estructura interna de la web, pudiendo identificar estos párrafos dentro de toda la web con la etiqueta "pico", etiqueta que comparten todas las noticias y que facilitan en gran medida el uso de las herramientas de BeautifulSoup.

Dado que el usuario puede poner cualquier url en el buscador hemos añadido una pequeña comprobación antes de realizar la búsqueda, permitiendo únicamente urls correctas obtenidas de la web Independent, marcando como error cualquier otra petición.

Una vez obtenidos estos textos podemos aplicar las funciones de la librería TextBlob para conocer la subjetividad y polaridad de los textos. Esta librería funciona de forma muy sencilla y solo basta una llama para conseguir la información de un texto en concreto:

```
sexistTest = 'Darren Rodwell, the Labour leader of the London borou
gh of Barking and Dagenham, has been entertained 19 times by MBS Gr
oup in a luxury corporate box at West Ham's stadium in the wake of
deals paving the way for MBS and its parent company, Hackman Capita
l Partners, to create 12 sound stages on former industrial sites.'
blobS = TextBlob(sexistTest)

for sentence in blobS.sentences:
    print(sentence.sentiment)

>> Sentiment(polarity=0.13333333333333333, subjectivity=0.13333333333333333)
```

Estos datos se obtendrán a partir del texto sin tratar, pues esta librería realiza su propio procesamiento de datos en su interior, funcionando de mejor forma cuando menos tratado están los datos que se les pasa. Este ejemplo se puede visualizar mejor en el notebook "SentimentAnalysis.ipynb"

Con el análisis de sentimientos realizado solo nos queda aplicar las transformaciones a los textos para aplicar sobre ellos el modelo de aprendizaje automático que hemos construido, consiguiendo así predecir el porcentaje de machismo que encontramos en la noticia pasada. Este ejemplo lo podemos ver directamente sobre la interfaz gráfica generada en la figura X.

La forma en la que vamos a calcular los resultados finales es muy sencilla y depende directamente del número de párrafos que disponga la noticia:

I. **Obtención de textos:** Usando el web scraping obtendremos un dataframe con tantas filas como número de párrafos tenga la noticia.

II. **Análisis de sentimientos:** Aplicaremos el análisis de sentimientos anteriormente visto por cada fila, obteniendo la media por frase dentro de cada párrafo y a su vez la media por cada uno de los párrafos, calculando así el valor total de la noticia.

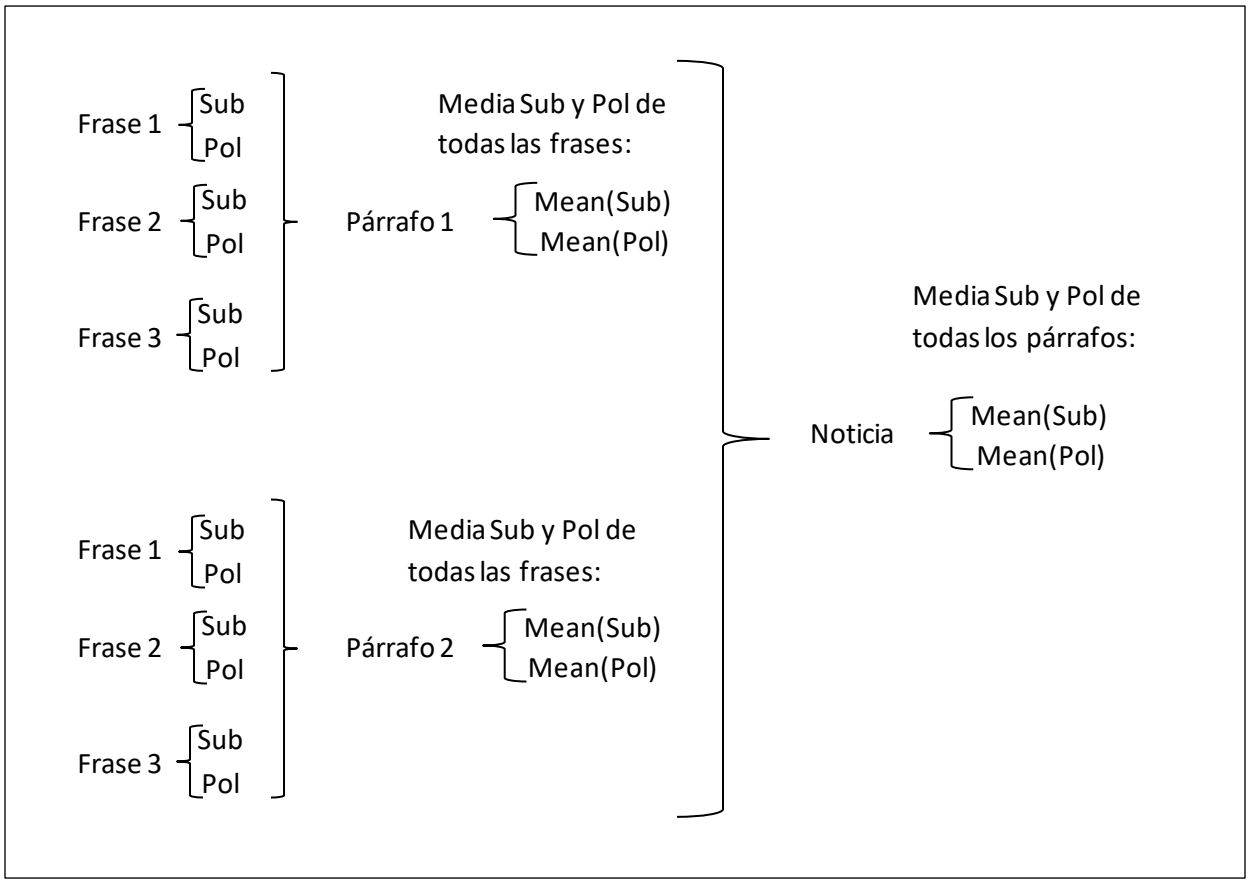


Figura 55. *Esquema de Análisis de Sentimientos*

III. **Transformación de los textos:** Con el análisis de sentimientos hecho ya podemos transformar los textos al formato numérico usando las técnicas de preprocesamiento, obteniendo así una cadena de entrada válida para la predicción.

IV. **Predicción:** Esta fase va a funcionar de forma muy parecida al análisis de sentimientos, solo que, en lugar de calcular por frases, va a predecir directamente por párrafos. Dependiendo del número de párrafos que den positivos como machistas y el número de párrafos, dará un porcentaje mayor o menor.

Por ejemplo: Si de una noticia de 5 párrafos, uno de ellos es machista, nuestro programa marcará la noticia con un 20% de machismo.

Todas estas fases forman la rutina completa de nuestro programa, al cual solo le queda tomar una estructura de interfaz de usuario. La programación completa de esta se puede encontrar en el notebook "interfaz.ipynb"

Esta interfaz está desarrollada íntegramente con la librería tkinter, por lo que todos los elementos que la componen son directamente obtenidos de la propia librería, sin la necesidad de instalar ninguna extensión.

Figura 56. *Interfaz del programa base*

Como podemos comprobar, en la interfaz podemos ver referenciadas todas las puntuaciones que hemos ido desarrollando a lo largo de la experimentación, pudiendo elegir entre los dos modelos que hemos entrenado junto a la visualización de las características.

En la figura 57 podemos ver la información de características para el algoritmo de regresión lineal de la misma forma que la hemos presentado en la memoria, permitiendo al usuario final conocer los datos de transparencia de forma sencilla.

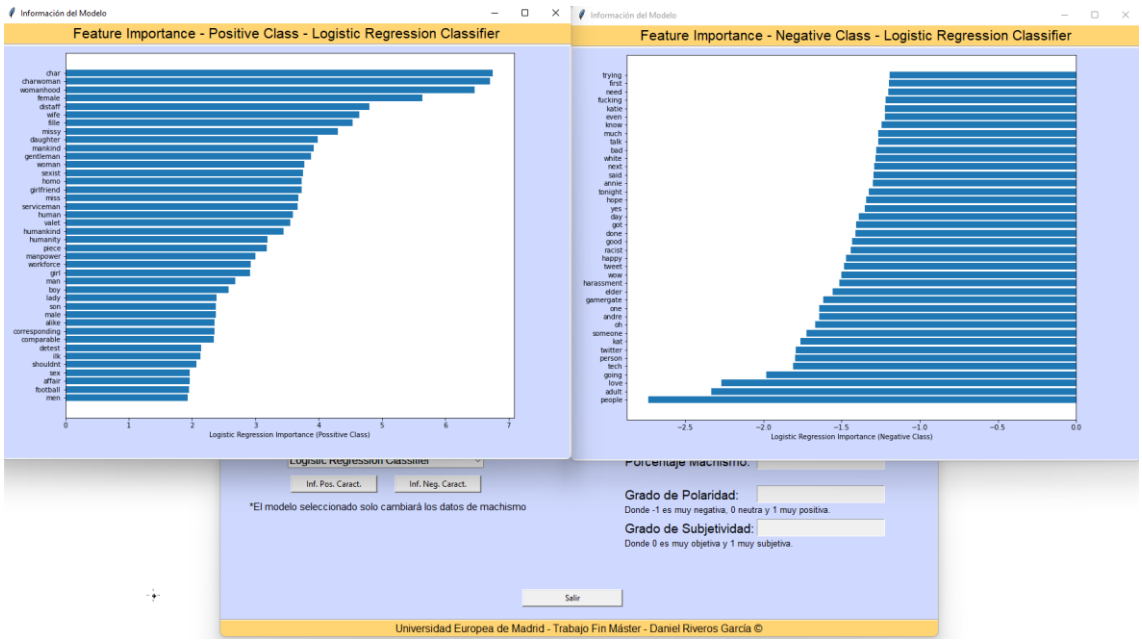


Figura 57. *Datos de Características LR. GUI*

Por último, vamos a presentar los resultados dados de la siguiente noticia, aunque lo mejor que pueden hacer es descargarse el programa y probarlo por ustedes mismo, comprobando la eficacia, versatilidad y eficiencia de nuestro proyecto.

Figura 58. *Prueba de la aplicación con una noticia cualquiera*

Con esta prueba en la propia interfaz podemos dar por concluida nuestra fase de experimentación, dando por cumplido y cerrado todos los objetivos marcados en el proyecto.

Capítulo 5
Conclusiones y Trabajo Futuro

5.1 Conclusiones

En este proyecto hemos podido hacer uso de multitud de herramientas de diferente índole, consiguiendo así aplicar todos los conocimientos vistos a lo largo del máster, pudiendo poner sentido a cada una de las asignaturas vistas a lo largo del año.

El uso de diferentes técnicas de aprendizaje automático, así como el aprendizaje sobre técnicas de *web scraping*, análisis de sentimientos y creación de interfaces gráficas, han hecho que este TFM no solo haya servido para demostrar los conocimientos adquiridos a lo largo del máster, sino que estos han sido a su vez cumplimentados con herramientas no estudiadas, sacándo aún beneficios a la realización del proyecto.

El uso de preprocesamiento diferente y la necesidad concreta de balancear las clases nos ha acercado a diferentes formas de tratamiento de textos anteriormente desconocidas, consiguiendo incrementar el conocimiento hacía el tratamiento de textos y aprendizaje automático de los mismos, uno de los objetivos personales que pretendía suplir con la realización de este trabajo.

Habiendo realizado varias pruebas de la aplicación a partir de la interfaz gráfica nos ha permitido conocer como funciona cada uno de los modelos expuestos, viendo como el modelo más preciso ha sido RandomForest, mientras que el más sensible a las palabras de mayor peso es LogisticRegression, ofrenciendo en la mayoría de los casos una puntuación machista superior al anteriormente mencionado.

Los hiperparámetros obtenidos por la búsqueda Grid nos han resultados de gran ayuda para conocer los mejores datos a aplicar en nuestro proyecto, consiguiendo la mejor opción en todo momento para nuestro trabajo.

Es cierto que la necesidad de usar preprocesamiento y la repetición de la estructura de las frases no nos ha permitido aumentar el tamaño de entrada de los vectorizadores, teniendo que acortar el número de características para evitar el sobreaprendizaje.

Si hacemos un repaso de los objetivos, y recuperando los mismos del apartado correspondiente, podemos concluir lo siguiente:

I. **Obtención de textos.** Realizado web scraping obteniendo los textos de la web correspondiente. ✓

II. **Procesamiento de textos.** Creada función para la limpieza y lematización de los textos, realizando un balaceo de clases en últma instancia. ✓

III. **Aplicación de soluciones Machine Learning.** Obtenido dos modelos de utilidad para aplicar en nuestro problema. ✓

IV. **Representación de los datos.** Creada una interfaz de usuario de fácil uso que permite trabajar con el programa de forma intuitiva. ✓

Podemos comprobar cómo hemos cumplido los objetivos propuestos desde un principio, gracias en parte a la realización de una planificación inicial al inicio del proyecto, la cual ha ayudado a cumplir los plazos y ordenar las diferentes tareas a realizar, obteniendo con el seguimiento del mismo, el trabajo realizado con éxito.

Uniendo la consecución de todos los subobjetivos conseguidos, resultamos en una buena forma de analizar diferentes noticias que podemos encontrar por la web establecida, siendo una aproximación a lo que en un futuro podría convertirse en una aplicación mucho más precisa.

5.2 Trabajo Futuro

Cómo hemos ido mencionando a lo largo del proyecto, esta aplicación puede servir como punto inicial a construir una herramienta mucho más precisa, consiguiendo implementar algunas de las limitaciones que nos hemos ido encontrando a lo largo de la realización de esta.

La principal limitación que nos hemos encontrado ha sido a la hora de obtener datos de calidad. Hay muy poca información y datos clasificados sobre el machismo en internet, siendo casi imposible encontrar un buen conjunto de datos a los que aplicar los modelos de aprendizaje automático.

Una posible mejora de esto sería realizar un estudio tal y como lo plantean los *papers* que hemos estudiado, consiguiendo un *dataset* con datos mucho más diversos y completos, ya que basarnos únicamente en textos de twitter que además han sido aumentados supone una limitación muy grande a la hora de analizar textos de diferente temática.

La mejor alternativa es estudiar un conjunto de datos parecido a los datos que vamos a predecir en última instancia, habiendo sido idel obtener el dataset etiquetado obtenido de algún tipo de texto periodístico que pueda calzar mejor con la estructura de texto que tienen las noticias de la web.

Otro factor que se ha escapado de los límites del TFM ha sido el uso del lenguaje español para analizar las noticias. Encontrar un *dataset* de textos machistas ha sido tarea imposible, siendo imposible entrenar un modelo que fuera capaz de analizar este tipo de textos.

Una de las alternativas a esto era usar APIs de traducción, las cuales son de pago y que suponen un coste fuera del alcance de este trabajo, sobre todo a la hora del uso por parte de usuarios externos, ya que prevalecemos la gratuidad del programa ante la versatilidad de idiomas.

Utilizar herramientas de pago no ha entrado dentro de los límites de nuestro TFM, pudiendo conseguir utilizar algún tipo de traductor en el caso que se requiera de más potencia o el proyecto pase a una escala mayor a la de un trabajo académico.

Por último, utilizar nuevas técnicas de aprendizaje profundo con el uso de *Transformers* o similares podría incrementar la precisión de los modelos, consiguiendo por tando mejores resultados.

Estos modelos no pueden ser estudiados de la misma forma que los usados en el trabajo, no pudiendo realizar una explicación clara sobre los modelos utilizados, siendo menos visibles para la representación de los mismos en el proyecto.

Como podemos comprobar, el potencial generado aún con las limitaciones hacen de este proyecto un primer gran paso hacía lo que podría convertirse en una herramienta indispensable en la publicación de noticias.

La idea ideal es conseguir un analizador que antes de publicar cualquier tipo de noticia pueda obtener los datos de la misma, haciendo que el editor corrija lo necesario antes de publicarla, evitando añadir sesgos de cualquier tipo.

Fin.

Capítulo 6
Bibliografía y Referencias

6.1 Bibliografía

Alexander, P. A., & Jetton, T. (1996). The role of importance and interest in the processing of text. *Educational Psychology Review*.

Anónimo. (2013). *La desigualdad de género en las redes sociales.* Vitoria-Gasteiz.

Anónimo. (s.f.). *Oxfam Intermón*. Obtenido de https://www.es.amnesty.org/en-que-estamos/blog/historia/articulo/las-frases-machistas-mas-controvertidas/

Denecke, K., & Deng, Y. (2015). *Artificial Intelligence in Medicine.* Leipzig, Germany.

Dhingra, D. (Julio de 2022). *Text Preprocessing in NLP with Python codes.* Obtenido de https://www.analyticsvidhya.com/blog/2021/06/text-preprocessing-in-nlp-with-python-codes/

Edgar, T. W., & Manz, D. O. (2017). *Research Methods for Cyber Security.* Obtenido de ScinceDirect: https://www.sciencedirect.com/topics/computer-science/logistic-regression

Ferrero, R. (Mayo de 2020). *QUÉ SON LOS ÁRBOLES DE DECISIÓN Y PARA QUÉ SIRVEN.* Obtenido de Maxima Formación: https://www.maximaformacion.es/blog-dat/que-son-los-arboles-de-decision-y-para-que-sirven/

Gamba, S. (Marzo de 2008). *Mujeres en Red*. Obtenido de https://www.mujeresenred.net/spip.php?article1397

Gudivada, V. N., Apon, A., & Ding, J. (2017). Data Quality Considerations for Big Data and.

Johnson, D. (Septiembre de 2022). *NLTK Tutorial: What is NLTK Library in Python?* Obtenido de Guru 99: https://www.guru99.com/nltk-tutorial.html

Karabiber, F. (2020). *TF-IDF — Term Frequency-Inverse Document Frequency.* Obtenido de Learn Data Scienc.

Kłeczek, D. (2020). Polbert: Attacking Polish NLP Tasks with.

Martín, J. (Mayo de 2021). *RTVE*. Obtenido de https://www.nature.com/articles/d41586-018-05707-8

McCrudden, M. T., & Schraw, G. (2007). Relevance and Goal-Focusing in Text Processing. *Educational Psychology Review*.

Milly. (Enero de 2021). *7 Web Scraping Limitations You Should Know*. Obtenido de https://www.octoparse.com/blog/web-scraping-limitations

Paruchuri, V. (Marzo de 2021). *Tutorial: Web Scraping with Python Using Beautiful Soup.* Obtenido de https://www.dataquest.io/blog/web-scraping-python-using-beautiful-soup/

Pascual, F. (Febrero de 2022). *Hugging Face.* Obtenido de https://huggingface.co/blog/sentiment-analysis-python

Ramírez, V. (Marzo de 2022). *La Sexta.* Obtenido de https://www.lasexta.com/noticias/sociedad/redes-sociales-lugar-donde-confluyen-machismo-empoderamiento-adolescentes_20220307622619db447ec100016414fd.html

Richardson, L. (2015). *Beautiful Soup*. Obtenido de https://beautiful-soup-4.readthedocs.io/en/latest/#quick-start

Samory, M., Sen, I., Kohne, J., Flöck, F., & Wagner, C. (2021). "Call me sexist, but...": Revisiting Sexism Detection Using Psychological Scales. *Computational Social Science*.

Schiebinger, J. Z. (Julio de 2018). *Nature*. Obtenido de https://www.nature.com/articles/d41586-018-05707-8

Shah, D. (Julio de 2022). *The Essential Guide to Data Augmentation in Deep Learning.* Obtenido de https://www.v7labs.com/blog/data-augmentation-guide

Shahul. (Julio de 2022). *Data Augmentation in NLP: Best Practices From a Kaggle Master.* Obtenido de Neptune: https://neptune.ai/blog/data-augmentation-nlp

Techslang. (s.f.). *What is Lemmatization?* Obtenido de https://www.techslang.com/definition/what-is-lemmatization/

Val, V. A. (Marzo de 2020). *Amnistía Internacional*. Obtenido de https://www.es.amnesty.org/en-que-estamos/blog/historia/articulo/las-frases-machistas-mas-controvertidas/

Web Scraper. (Mayo de 2021). Obtenido de https://webscraper.io/blog/brief-history-of-web-scraping#:~:text=The%20origins%20of%20very%20basic,institutes%20all%20around%20the%20world.

Weng, J. (Agosto de 2019). *Towards Data Science.* Obtenido de https://towardsdatascience.com/nlp-text-preprocessing-a-practical-guide-and-template-d80874676e79

Westerski, A. (s.f.). *Sentiment Analysis: Introduction and the State of the.* Madrid.

6.2 Referencias Generales

Getting Started with Sentiment Analysis using Python (huggingface.co)

What is Lemmatization? — Definition by Techslang

Stemming and lemmatization (stanford.edu)

Stemming and lemmatization (towardsdatascience.com)

Sklearn Random Forest Classifiers in Python Tutorial | DataCamp

Qué son los árboles de decisión y para qué sirven | Máxima Formación (maximaformacion.es)

¿Qué es la regresión logística? - Regresión logística - AWS (amazon.com)

What is Logistic regression? | IBM

Logistic Regression - an overview | ScienceDirect Topics

Logistic Regression - an overview | ScienceDirect Topics (recursospython.com)

python - Label with counter on each bar in matplotlib - Stack Overflow

Tutorial: Quickstart — TextBlob 0.16.0 documentation

Twitter Sentiment Modeling on Detecting Racist or Sexist tweets - Eric Chen's Blog (haochen23.github.io)

Massive Machine Learning Study Demonstrates Gender Stereotyping And Sexist Language In Literature (forbes.com)

exist_paper4.pdf (ceur-ws.org)

Boosting Text Classification Performance on Sexist Tweets by Text Augmentation and Text Generation Using a Combination of Knowledge Graphs (aclanthology.org)

GitHub - Harikavuppala1a/Semi-supervised-Multi-level-neural-approach

Fine-Grained Multi-label Sexism Classification Using a Semi-Supervised Multi-level Neural Approach | SpringerLink

How to scrape websites with Python and BeautifulSoup (freecodecamp.org)

ENC2045 Computational Linguistics — ENC2045 Computational Linguistics (alvinntnu.github.io)

Opinion Mining, Sentiment Analysis, Opinion Extraction (uic.edu)

Lexicon-Based Sentiment Analysis: A Tutorial | KNIME

715155.pdf (ub.edu)

Multi-label Categorization of Accounts of Sexism using a Neural Framework (aclanthology.org)

How to fine tune bert on entity recognition? - Beginners - Hugging Face Forums

regex - How to remove any URL within a string in Python - Stack Overflow

Extracting (or Removing) Mentions and Hashtags in Tweets using Python – Catris Code

removing emojis from a string in Python - Stack Overflow

RegExr: Learn, Build, & Test RegEx

NLP Tutorial for Text Classification in Python | by Vijaya Rani | Analytics Vidhya | Medium

Text Classification with Python and Scikit-Learn (stackabuse.com)

SMOTE explained for noobs – Synthetic Minority Over-sampling TEchnique line by line | Rich Data (rikunert.com)

Imbalanced dataset in text classification | Data Science and Machine Learning | Kaggle

Yet Another Twitter Sentiment Analysis Part 1 — tackling class imbalance | by Ricky Kim | Towards Data Science

Practical Text Classification With Python and Keras – Real Python

6.3 Referencias Figuras

1. https://img.freepik.com/vector-gratis/ilustracion-acuarela-bandera-feminista-puno-simbolo-femenino_23-2148868506.jpg?w=2000

2. https://educacion30.b-cdn.net/wp-content/uploads/2022/02/igualdad-en-las-aulas-2-978x652.jpg

3. Elaboración propia (Íconos obtenidos en https://www.flaticon.com/)

4. https://i.pinimg.com/originals/40/ec/d8/40ecd87c7f41c31a58ad21fb0796c5b2.png

5. https://www.cs.us.es/~fsancho/images/2019-12/machine-learning-types.jpg

6. https://i.pinimg.com/originals/e7/77/31/e77731ce84921b2678f5c31830c7241c.png

7. Elaboración propia

8. https://www.antevenio.com/wp-content/uploads/2019/03/web.jpeg

9. https://www.kdnuggets.com/wp-content/uploads/text-data-task-framework.png

10. https://static.javatpoint.com/tutorial/nlp/images/what-is-nlp.png

11. https://i1.wp.com/turbolab.in/wp-content/uploads/2021/10/countvectorizer-1.png?resize=642%2C297&ssl=1

12. http://www.adamwesterski.com/wp-content/files/docsCursos/sentimentA_doc_TLAW.pdf

13. https://huggingface.co/blog/assets/50_sentiment_python/thumbnail.png

14a. https://ecoscript.org/wp-content/uploads/2022/03/cover.png

14b. https://www.mentiona.com/wp-content/uploads/2021/10/PowerBI.jpg

15. https://i.stack.imgur.com/cQ3KA.png

16. Elaboración propia (Íconos obtenidos en flaticon.com)

17. Elaboración propia (Íconos obtenidos en flaticon.com)

18. Elaboración propia (Íconos obtenidos en flaticon.com)

19. https://sixfeetup.com/blog/an-introduction-to-beautifulsoup/@@images/27e8bf2a-5469-407e-b84d-5cf53b1b0bb6.png

20. https://upload.wikimedia.org/wikipedia/commons/d/d3/Santa_Barbara_Independent_logo.jpg

21. https://i.blogs.es/0a3d98/4af37741-965d-4021-8197-19a6fae4c182/1366_2000.jpeg

22. https://regexr.com/assets/card.png

23. https://user.oc-static.com/upload/2020/10/22/16033589274175_lemmatization%20example%2001.png

24. Link demasiado grande para poner en la memoria. Preguntar personalmente si es necesario.

25. https://mlwhiz.com/images/tfidf.png

26. https://amitness.com/images/nlp-aug-bert-augmentations.png

27. https://www.juliankohne.com/publication/callmesexistbut/featured_hu8f1124febf62d416c20e14f20240d5d7_51383_720x0_resize_lanczos_2.png

28. https://res.cloudinary.com/dyd911kmh/image/upload/f_auto,q_auto:best/v1526467744/voting_dnjweq.jpg

29. https://d1.awsstatic.com/S-curve.36de3c694cafe97ef4e391ed26a5cb0b357f6316.png

30. Elaboración propia

31. https://recursospython.com/wp-content/uploads/2021/08/conversor-de-temperatura-tkinter.png

32a. https://i0.wp.com/blog.330ohms.com/wp-content/uploads/2021/08/destacada.png?resize=696%2C377&ssl=1

32b. https://miro.medium.com/max/1200/1*HB9UFEshgtKVMErMAPtvqQ.png

33. https://code.visualstudio.com/opengraphimg/opengraph-home.png

34 - 58. Elaboración propia

Anexo I
Instalación y Uso de la Aplicación

El uso de la aplicación es muy sencillo, solo hay que descargar el notebook y usarlo localmente con Jupiter Notebook o un programa compatible con el mismo, como por ejemplo Visual Studio Code.

En este apartado explicaremos el uso con la plataforma Anaconda, la cual ha sido utilizada para la realización de la aplicación. Explicaremos todo paso a paso:

I. Descargar del repositorio de GitHub la carpeta de Images y Models junto al cuaderno llamado interfaz.ipynb. Poner en la carpeta donde queráis ejecutarlo respetando la jerarquía de carpetas.

II. Descargar Anaconda y Visual Studio Code tal y como lo indican sus páginas web.

III. Crear un entorno en conda e instalar las siguientes librerias (el mejor modo es usar la consola usando los comandos “conda install…”:
   a. nltk
   b. pillow
   c. request
   d. pandas
   e. beautifulsoup4
   f. textblob
   g. ipykernel
   h. scikit-learn

IV. Instalar en Visual Studio Code los plugins de Jupyter y Pylance.

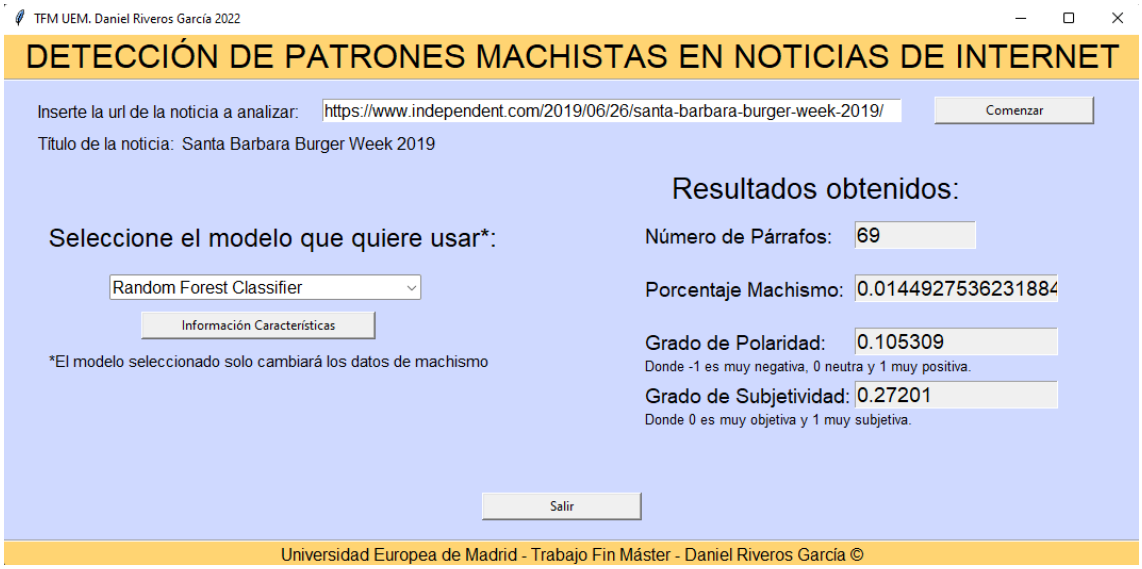
V. Abrir la carpeta en Visual Studio Code donde se haya descargado el proyecto y abrir el cuaderno interfaz.ipynb. Hay que tener cuidado con los nombres que tengan las carpetas, pues en el programa se hace uso de rutas relativas.

VI. Ejecutar el programa desde Visual Studio Code y listo.

Anexo II
Ejemplo de Ejecución

Vamos a añadir una captura de un ejemplo en concreto obtenido por nuestro programa, la url usada ha sido:

https://www.independent.com/2019/06/26/santa-barbara-burger-week-2019/



Si observamos diferentes ejemplos de las ejecuciones podemos ver como la mayoría de las noticias cuentan con un grado de subjetividad muy cercano a 0, siendo de categoría mayoritariamente objetiva, algo que cuadra con el tipo de texto que estamos tratando.

A su vez, que el grado de polaridad sea cercano a la puntuación neutral indica que el texto está bien redactado por parte del editor, consiguiendo evitar posicionarse en la noticia, siendo lo más imparcial posible,

En el porcentaje de machismo podemos comprobar como al disponer el texto de las features indicadas por el modelo, el valor de porcentaje sube, comprobando así la eficacia y funcionamiento de estos.