UNIVERSIDAD POLITÉCNICA DE MADRID

E.T.S. DE INGENIERÍA DE SISTEMAS INFORMÁTICOS

PROYECTO FIN DE GRADO

GRADO EN INGENIERÍA DEL SOFTWARE

PREDICCIÓN DE NOTICIAS FALSAS CON MACHINE LEARNING

Autora: Inna Krasimirova Hristova

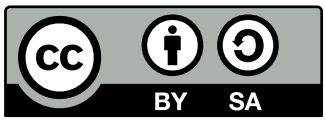
Director: Fernando Ortega Requena

Madrid, mayo 2022

Inna Krasimirova Hristova

PREDICCIÓN DE
NOTICIAS FALSAS
CON MACHINE LEARNING
Proyecto Fin de Grado, jueves 19 mayo, 2022
Director: Fernando Ortega Requena

E.T.S. de Ingeniería de Sistemas Informáticos
Campus Sur UPM, Carretera de Valencia (A-3), km. 7
28031, Madrid, España



Resumen

En los últimos años con la innovación tecnológica el concepto de ”fake news” esta cada vez mas presente en la sociedad, se propagan a la velocidad de la luz y puede llevarnos a una toma de decisiones errónea. Nos encontramos en un punto en que el propio usuario es productor y consumidor de noticias.

Aprovechando la era tecnológica en la que estamos y todos los beneficios que nos brinda, utilizaremos algoritmos de Machine Learning para la predicción de noticias falsas. El objetivo es estudiar y comparar varios algoritmos, y elegir el que mejor se adapte a nuestro caso y más predicciones correctas haga con nuestros datos.También profundizaremos en dicho algoritmo y explicaremos por que es el más eficiente para este proyecto en concreto.

Sabemos que las noticias falsas generan odio y pueden ocasionar importantes pérdidas económicas a las empresas, los países y las familias, e interfieren en el derecho de las personas a buscar y recibir información. Se trata de poder identificarlas y a detener su propagación. Este proyecto pretende que este impacto social negativo que se ha generado estos últimos años, pase a ser un impacto social positivo, que los usuarios se beneficien. Se trata de que los usuarios al poseer tanta información y conocimientos sean capaces de enriquecerse con ello y Internet es un medio de incalculable valor para investigar sobre cualquier tópico y hay que saber aprovecharlo.

Palabras clave: Machine Learning; Predicción de Noticias Falsas;

Abstract

In recent years with technological innovation the concept of "fake news" is increasingly present in society, propagate at the speed of light and can lead to wrong decision making. We are at a point where the user himself is a producer and consumer of news.

Taking advantage of the technological era we are in and all the benefits it gives us, we will use Machine Learning algorithms for the prediction of fake news. The goal is to study and compare several algorithms and choose the one that best suits our case and makes the most correct predictions with our data. We will also delve into this algorithm and explain why it is the most efficient for this project.

We know that fake news generates hatred and cause significant economic losses to businesses, countries and families and interfere with the right of individuals to seek and receive information. It's about being able to identify them and stop their spread. This project aims to make this negative social impact, that users benefit from each other. It is that users have so much information and knowledge are able to get rich themselves with it and the Internet is an invaluable means to investigate any topic and you have to know how to take advantage of it

Keywords: Machine Learning; Predict Fake News;

Agradecimientos

Hace unos 8 años cuando acabé segundo de Bachillerato tenía muy claro que no quería ir a la Universidad (no sabía nada de Matemáticas, ni de Inglés) y seguí el camino del "Grado Superior", donde me dieron uno de los mejores consejos, que fue que no perdía nada intentándolo (gracias Laura). Me di cuenta que las oportunidades y el tiempo no volvían así que pase de no querer ir a la universidad a estar en un doble grado.

Quería dar las gracias a mi familia en primer lugar y a mis amigos (Paula, Sara , Fernando, Dana) que siempre han estado apoyándome en este camino tan largo y difícil. Darme las gracias a mi misma por haber sido capaz de llegar hasta aquí y no abandonar, y por aguantarme a mi misma. Desde hace años tenía muy claro que tutor quería para mi Proyecto de Fin de Grado, así que también darle las gracias a Fernando por haberme ayudado tanto y haber estado conmigo durante este tiempo.

Por último, darle las gracias a la persona que me dio la idea de irme de Erasmus, ya que ha sido la mejor manera de acabar esta etapa de mi vida.

Índice general

Índice de figuras

Índice de cuadros

1. Introducción

Las noticias falsas y la información deliberadamente manipulada con el objetivo de engañar a los usuarios se han convertido en un fenómeno mundial. Esto no es algo nuevo, pero sí lo es la magnitud con la que pueden llegar a ser propagadas gracias a Internet y las nuevas tecnologías. Aquí es donde entran las redes sociales, que permiten a los usuarios ser productores y consumidores de contenido a la vez. Basta con que la noticia se vuelva viral para que empiece a propagarse a la velocidad de la luz, incluso en tiempo real. Se prioriza la viralidad sobre la veracidad, sólo era cuestión de tiempo. Sabemos que incluso los expertos tienen dificultades para detectar si una noticia es falsa de manera inmediata.

El 60% de los españoles cree que sabe detectar las noticias falsas, pero en realidad solo un 14% sabela diferencia. Veamos un ejemplo:

- **Noticia falsa**: "Amazon se prepara para lanzar un supermercado robotizado" → 57% de los encuestados respondió que es verdadera. Esta noticia lanzada por el New York Post basada en fuentes anónimas y desmentida por Jeff Bezos es recogida por numerosos medios de todo el mundo [1].
- **Noticia verdadera**: "Se ha descubierto una isla de plástico en el océano Pacífico" → La mitad de los encuestados pensó que era falsa.

Por si esto fuera poco, actualmente existen páginas web (NoticiasTT.com o 12Minutos.com) que ayudan a que los usuarios a crear noticias falsas, incluyendo titular, contenido y foto, de manera que tenga la misma apariencia que una verdadera. Las noticias falsas pueden tener consecuencias muy graves desde difamar a una persona, hasta influir en la opinión pública o provocar una alarma social.

Identificando al autor de la noticia, la fuente o el medio, ya no es suficiente para poder identificarlas. Es necesario buscar una solución que nos ayude a identificarlas con la misma rapidez con la que se propagan. Esto ayudará a mantener a los usuarios bien informados y seguros. Antes que nada, se deberá enseñar a los usuarios a no compartir noticias si no tienen claro si son verdaderas o no.

Con este Proyecto de Fin de Grado se pretende ayudar a los usuarios prediciendo si una noticia es falsa o no de la manera mas rápida posible, con la ayuda de técnicas de Machine Learning.

1.1 Motivación

Diariamente sufrimos bombardeos de noticias que recibimos a través de los medios digitales, pero, ¿como distinguimos si una noticia es verdadera o falsa, si las noticias falsas tienen la misma apariencia que las verdaderas?. Este Proyecto de Fin de Grado nace de titulares como :

- "Las 'fake news' aumentaron un 50% tras el confinamiento"
- "En el artículo señala 2022, que está a la vuelta de la esquina, como el año en el que el lector recibirá más noticias falsas que verdaderas " [2].
- "Las fake news crecen un 365% y amenazan la reputación corporativa"
- "Los competidores son los principales impulsores de las 'fake news' contra las empresas en las redes sociales".
- "La ciencia confirma que las 'fake news' se extienden más rápido que la verdad " [3].

Con esto nos damos cuenta de que hay un problema muy grave que afecta a la sociedad de hoy en día y puede llevar al usuario a tomar unas decisiones erróneas. Incluso nos puede pasar a nosotros mismos. Este proyecto quiere ayudar a los usuarios a que puedan consumir noticias totalmente veraces y que tengan una completa confianza en ello. Queremos buscar una solución adaptada a las nuevas tecnologías y qué mejor forma que poniendo en práctica técnicas de Machine Learning.

1.2 Objetivos

Igual que las fake news se propagan de una manera tan rápida gracias a las nuevas tecnologías, también vamos ha hacer uso de ellas para combatirlas. El objetivo de este proyecto será predecir

si una noticia es falsa o no mediante el aprendizaje supervisado, técnicas de clasificación. Cuando usamos clasificación, el resultado es una clase, entre un número limitado de clases, según el tipo de problema. En este caso si una noticia es falsa o no, sólo tiene dos clases. Y los algoritmos de Machine Learning de clasificación, tras darle una misma entrada de datos (noticias), tienen que elegir a qué clase pertenece. Gracias al Machine Learning la entrada de datos es ilimitada, aprende de comportamientos pasados y procesa, analiza y prevé de forma rápida. Es decir, dado que los modelos de aprendizaje supervisado hacen predicciones del mundo real basándose en los datos del pasado, los modelos deben ser reconstruidos periódicamente con el fin de mantener sus predicciones actualizadas.

Los objetivos específicos del proyecto son los siguientes:

- Conocer las ventajas del Machine Learning y de sus algoritmos.
- Estudiar los distintos algoritmos de clasificación supervisada de Machine Learning.
- Estudiar los algoritmos de Machine Learning que más se ajustasen a los datos que vamos a tratar.
- Comparar los resultados obtenidos de las distintos algoritmos de clasificación.
- Entender por que el algoritmo escogido es el que más nos conviene.
- Entender el gran impacto social que tienen las noticias falsas hoy en día.

1.3 Estructura de la memoria

Esta memoria presenta la siguiente estructura:

- **Introducción**: en esta sección se expone el tema del proyecto, la motivación que nos ha llevado a realizarlo y cuales son los objetivos que pretendemos alcanzar con este desarrollo.
- **Tratamiento de datos**: descripción de los datos que vamos a utilizar y que el posterior tratamiento que e hará con ellos.
- **Clasificación supervisada**: se explica en que consisten los algoritmos de clasificación que vamos a utilizar.

- **Metodología**: descripción detallada de los pasos que se han seguido para el desarrollo del sistema y las librerías que se han utilizado para su desarrollo.
- **Evaluación**: en esta sección vamos a comparar los distintos resultados obtenidos y a analizarlos detalladamente.
- **Técnica final**: se explica de manera mas detallada el algoritmo que finalmente hemos elegido.
- **Conclusiones**: se analiza si la implementación realizada cumple con los objetivos propuestos.
- **Impacto social**: analizamos como este proyecto puede impactar y llevar a la sociedad a una mejora global.
- **Futuras mejoras**: se habla de las mejoras que puede sufrir el proyecto en un futuro.

2. Tratamiento de datos

2.1 Descripción de datos

Los datos en este proyecto son clave, nos van ha permitir analizar las noticias y entrenar nuestro algoritmo. El conjunto de datos del que vamos a partir para realizar este proyecto son extraídos de la plataforma "Kaggle" [4]. Kaggle es es una comunidad en línea de científicos de datos y profesionales del aprendizaje automático, que permite encontrar y publicar conjuntos de datos.

Nustro dataset en particular esta formado por las siguientes columnas como podemos ver en la Figura 2.1:

- "**URLs**": Muestra el enlace en donde se ha obtenido la noticia.
- "**Headline**": Nos ofrece el título de la noticia.
- "**Body**": Expone el cuerpo de la noticia.
- "**Label**": Es la variable que nos indica si una noticia es falsa "0" y si es verdadera "1".

Figura 2.1: Columnas del dataset utilizado

```
URLs        http://www.bbc.com/news/world-us-canada-414191...
Headline           Four ways Bob Corker skewered Donald Trump
Body        Image copyright Getty Images\nOn Sunday mornin...
Label                                                       1
Name: 0, dtype: object
```

Mostramos en detalle el formato de texto de la noticia y es de aquí de donde vamos a partir con los datos:

Image copyright Getty Images On Sunday morning, Donald Trump went off on a Twitter tirade against a member of his own party. This, in itself, isn't exactly huge news. It's far from the first time the president has turned his rhetorical cannons on his own ranks. This time, however, his attacks were particularly biting and personal. He essentially called Tennessee Senator Bob Corker, the chair of the powerful Senate Foreign Relations Committee, a coward for not running for re-election. He said Mr Corker "begged" for the president's endorsement, which he refused to give. He wrongly claimed that Mr Corker's support of the Iranian nuclear agreement was his only political accomplishment. Unlike some of his colleagues, Mr Corker - free from having to worry about his immediate political future - didn't hold his tongue. Skip Twitter post by @SenBobCorker It's a shame the White House has become an adult day care center. Someone obviously missed their shift this morning. — Senator Bob Corker (@SenBobCorker) October 8, 2017 Report That wasn't the end of it, though. He then spoke with the New York Times and really let the president have it. Here are four choice quotes from the Tennessee senator's interview with the Times and why they are particularly damning. "I don't know why the president tweets out things that are not true. You know he does it, everyone knows he does it, but he does." You can't really sugarcoat this one. Mr Corker is flat-out saying the president is a liar - and everyone knows it. The senator, in particular, is challenging Mr Trump's insistence that he unsuccessfully pleaded for his endorsement, but the accusation is much broader. Mr Corker and the president used to be something akin to allies. The Tennessean was on Mr Trump's short list for vice-president and secretary of state. Image copyright Getty Images Image caption Bob Corker at Trump campaign rally in July 2016 Those days are seemingly very much over now - and it's not like Mr Corker is going anywhere anytime soon. Although he's not running for re-election, he'll be in the Senate, chairing a powerful committee, until January 2019. The president's margin for success in that chamber is razor-thin. If Democrats can continue to stand together in opposition, he can afford to lose only two votes out of 52 Republican senators. That's why healthcare reform collapsed in July - and it could be bad news for tax efforts. From here on out, Mr Corker isn't going to do the president any favours. "Look, except for a few people, the vast majority of our caucus understands what we're dealing with here." Frustration in Congress has been growing over what Republicans feel has been the president's inability to focus on advancing their agenda. Getting a sharply divided party to come together on plans to repeal Obamacare, reform taxes or boost infrastructure spending is challenging enough. Doing so when the president stirs up unrelated controversies on a seemingly daily basis makes things all the harder. One of the president's gifts has been his ability to shake off negative stories by quickly moving on to a different subject. That worked brilliantly during his presidential campaign, but it's less effective during the legislative slow grind. Image copyright Getty Images Image caption Corker at the confirmation hearing for Secretary of State Rex Tillerson For months, Republicans in Congress have been grumbling about this in the background and among themselves. Occasionally, someone like Mr McConnell will lament that the president doesn't understand how the Senate works. Mr Corker has now stated it loud and clear. And, what's more, he says almost everyone agrees with him. They've kept silent until now because they still hope to pass conservative legislation that the president can sign or fear Mr Trump's legions will back a primary challenge next year or stay home during the general election. If that calculus ever changes - if it becomes riskier to stay silent than speak out - Mr Trump will be in real trouble. "A lot of people think that there is some kind of 'good cop, bad cop' act underway, but that's just not true." Time and again, Mr Trump has appeared to undercut Secretary of State Rex Tillerson and others in his administration who are attempting to use soft diplomacy to deal with a range of international crises. The war against the Taliban in Afghanistan, Iran's compliance with the multinational nuclear agreement, the ongoing dispute between Qatar and its Persian Gulf neighbours, the unrest in Venezuela and, most recently, North Korea's continued ballistic missile tests have all been the target of the president's offhand remarks and Twitter invective. Some administration defenders have said this is all a part of Mr Trump's strategy - an updated version of the Nixon-era "madman theory", in which the president forces adversaries to give way because they fear an unpredictable US leader's actions. Mr Corker isn't buying it. There's no strategy, he says, just the possibility of chaos - which he hopes Mr Trump's senior advisers will be able to avoid. "I know for a fact that every single day at the White House, it's a situation of trying to contain him." There's now a growing collection of John Kelly face-palm photos that serve as a testament to the chief-of-staff's reported frustration at dealing with the president. Mr Trump goes off-script to praise torch-bearing white nationalists at a rally in Charlottesville, and Mr Kelly is captured closing his eyes and rubbing the arch of his nose, as if attempting to stave off a migraine. Image copyright Reuters Image caption White House Chief of Staff John Kelly looks on as US President

Donald Trump speaks at a campaign rally The president calls North Korean leaders "criminals" in a speech to the United Nations, and Mr Kelly straight-up buries his face in his hands. The White House communications team is often left scrambling to try to explain or reframe an indelicate presidential "joke" or remark that directly contradicts what was until then the official administration line. Even though Mr Kelly has brought some discipline to the West Wing staff, the president still marches to the beat of his own drum - and continues to have unfettered access to his phone's Twitter app. Bob Corker is only the latest person - politician, journalist, sports star or celebrity - to feel the mercurial president's uncontainable ire.

Estamos en la era del Big Data, es decir, tratamos con cantidades inmensas de información. Queremos extraer conocimiento de calidad de estos datos, y para ello los datos con los que tratamos tienen que tener una buena calidad por que los datos se pueden ver afectados por distintos factores como puede ser el ruido, valores inconsistentes o valores perdidos.

2.2 Preprocesamiento de texto

La calidad de datos impulsa la innovación, el desarrollo y la transformación. Partiendo de un conjunto de datos sin tratamiento, vamos a depurarlos, para manejarlos de manera más fácil y eficiente. Tenemos que mencionar el Procesamiento de Lenguaje Natural o NLP que se ocupa de las interacciones entre los ordenadores y los lenguajes humanos (naturales). Se trata de programar a los ordenadores para procesar y analizar grandes cantidades de datos del lenguaje natural. Utilizaremos varias técnicas para procesar nuestras noticias:

- **Tokenización**: La tokenización consiste en separar las palabras del texto en entidades llamadas tokens o unidad de procesamiento semántico, con los que trabajaremos. Deberemos estudiar si utilizaremos los signos de puntuación como token, si daremos importancia o no a las mayúsculas y si unificamos palabras similares en un mismo token.
- **Normalización de texto**: La normalización de carácteres de mayúscula a minúscula es necesaria, ya que la misma palabra en mayúscula y en minúscula se consideran dos palabras diferentes.
- **Eliminación de texto irrelevante**:
    1. **Eliminación de columnas**: se eliminaran columnas que no aporten ningún tipo de información y así evitamos cargar datos que no vamos a manipular.
    2. **Eliminación de filas**: se eliminan las filas que contienen algún dato vacío, ya que no aportan información útil.
    3. **Eliminación de caracteres**: también hay caracteres que no nos aportan ningún tipo de información:

  - **Eliminación de números**: se eliminarán todos los números.
  - **Eliminación de caracteres especiales**: eliminación caracteres no alfanuméricos.
  - **Eliminación de signos de puntuación**: eliminación de los signos de puntuación.
  - **Eliminación de palabras redundantes**: en esta parte se van a eliminar las ”palabras vacías”, que son las palabras mas comunes en un idioma como son los pronombres, artículos o adverbios entre otros. Para definirlas usamos el concepto de ”stop words”. Esto nos lleva a eliminarlas del texto antes de entrenar los modelos de aprendizaje aprendizaje automático, ya que las palabras vacías ocurren en abundancia, por lo que brindan poca o ninguna información única que se pueda usar para la clasificación. Además el tamaño del conjunto de datos disminuye y el tiempo para entrenar el modelo también disminuye sin un gran impacto en la precisión del modelo y mejora el rendimiento, ya que solo quedan los tokens importantes importantes. Por lo tanto, la precisión de la clasificación puede mejorarse.

- **Lematización** : muchas veces no nos interesa conocer todos los significados de un texto, sino solamente algunos pertinentes para realizar una tarea.Una buena técnica para obtener la información relevante de un texto consiste en eliminar los elementos que puedan ser irrelevantes, y resaltar más lo que los textos tienen en común que sus diferencias. La lematización es el proceso mediante el cual las palabras de un texto que pertenecen a un mismo paradigma flexivo o derivativo son llevadas a una forma normal que representa a toda la clase.

- **Stemming**: es un método para reducir las palabras a su raíz o steam. Quedarnos con la raíz de cada palabra, nos ayuda a quedarnos con el significado global. Habíendo visto lematización y stamming, a primera vista no vemos la diferencia entre ambos, pero sí que existe. La lematización hace un examen morfológico de las palabras, además aporta contexto a las palabras y une las palabras que tienen un significado similar. En cambio, stemming simplemente corta la raíz de la palabra sin analizarla.

En este proyecto los datos de entrada que vamos a utilizar para el algoritmo de Machine Learning son textos. Para poder trabajar es necesario transformar el texto en un vector numérico.

- **Bag of Words**: es un método que se utiliza en el procesado del lenguaje para representar textos ignorando el orden de las palabras. En este modelo, cada noticia parece una bolsa que contiene palabras y se usa generalmente para representar características. La longitud

del vector es el tamaño de la bolsa. Cada bit en el vector indica el número de veces que la palabra aparece en el texto. Es fácil de usar y tiene eficiencia computacional buena.
Como ya las tenemos convertidas en tokens, ahora tenemos que contar la frecuencia con la que aparecen los tokens en el texto. Para crear el modelo de bolsa de palabras, necesitamos crear una matriz donde las columnas corresponden a las palabras más frecuentes en nuestro diccionario y donde las filas corresponden a nuestro texto. En la Figura 4.8 se muestra un ejemplo de una ”bag of words”.

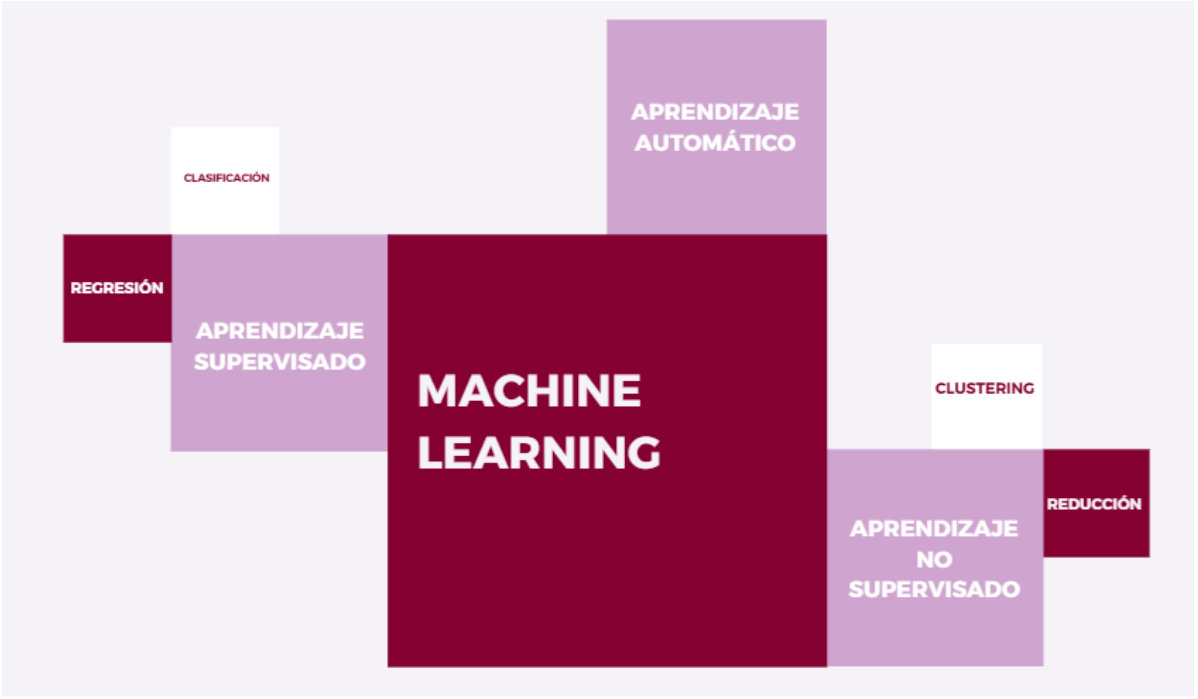
La preparación de los datos nos lleva a generar un conjunto de datos de calidad mas pequeño que el original, lo cual puede mejorar la eficiencia del proceso, incrementamos la productividad y minimizamos los errores.

3. Machine Learning

El aprendizaje automático o Machine Learning es un proceso en el que su sistema aprende de los sucesos, experimenta y sigue mejorando sus habilidades y capacidad de toma de decisiones. Se trata de transformar los datos en información y con estos tomar decisiones. Además esto implica que los sistemas se mejoran de forma autónoma con el tiempo, sin intervención humana.

Este tipo de aprendizaje es más sencillo de realizar ya que partimos de un conocimiento previo, que son los datos con los que realizamos el entrenamiento. A medida que aumentan los datos de entrenamiento para representar el mundo de una forma más realista, el algoritmo calcula resultados más precisos. Esto nos lleva a tener ventajas como la entrada de un número de datos ilimitados y el rápido proceso y análisis de estos datos. Al tener el control total sobre el material de entrenamiento, solo se necesitan las entradas y el tiempo suficientes para configurar a los algoritmos correctamente. Lo más importante es compilar la mayor variedad de datos posible. Puesto que se deben etiquetar todos los elementos, es necesario un esfuerzo considerable por parte de los desarrolladores y científicos.

Figura 3.1: Clasificación Machine Learning



En la Figura 3.1 podemos observar, que en Machine Learning existen varios tipos de aprendizaje, el proyecto se desarrollará con aprendizaje supervisado, el cual necesita un conjunto de datos previamente etiquetados, nosotros le indicamos al modelo que es lo que queremos que aprenda. dentro del aprendizaje supervisado nos centramos en clasificación, el resultado es una clase entre un número limitado de clases. Estos algoritmos son útiles cuando la respuesta al problema cae dentro de un conjunto finito de resultados posibles, y el resultado que queremos obtener es una de esas etiquetas.[5]

En el caso de que el modelo entrenado es para predecir cualquiera de las dos clases objetivos, verdadero o falso, por ejemplo, se le conoce como clasificación binaria. En este caso estamos tratando de predecir si una noticia es falsa o no, por lo tanto nos basaremos en una clasificación binaria.

Podemos destacar seis pasos básicos para realizar la tarea de aprendizaje automático [6]:

1. "**Recolección de datos**": obtención de los datos necesarios para alimentar nuestra máquina. Esto se explica más detalladamente en Capítulo 2.1.

2. "**Preparación de datos**": mejorar la calidad de nuestros datos. Lo veremos en el Capítulo 2.2

3. "**Elección del modelo**": de acuerdo a nuestro objetivo se eligen varios algoritmos. Explicado en el Capítulo 3.1
4. "**Entrenamiento del modelo**": Con los datos procesados, se entrena el modelo. Perteneciente al Capítulo 3.1
5. "**Evaluación del modelo**": se estudian los resultados obtenidos al entrenar el modelo. Perteneciente al Capítulo 5
6. "**Configuración del parámetro**": en caso de ser necesario, se pueden configurar los distintos parámetros para obtener un mejor resultado. Detallado en el Capítulo 4.

3.1 Algoritmos de clasificación

En este apartado veremos los distintos algoritmos de clasificación que podemos utilizar, y las ventajas y desventajas, que tiene cada uno de ellos. Esto no quiere decir que un algoritmo sea mejor que otro, simplemente cada algoritmo se adapta mejor o peor a nuestro caso en particular.

Regresión Logística

La regresión logística es la técnica predictiva más simple y resulta útil para los casos en los que se desea predecir la presencia o ausencia de una característica. Se trata de una red neuronal de una sola neurona.

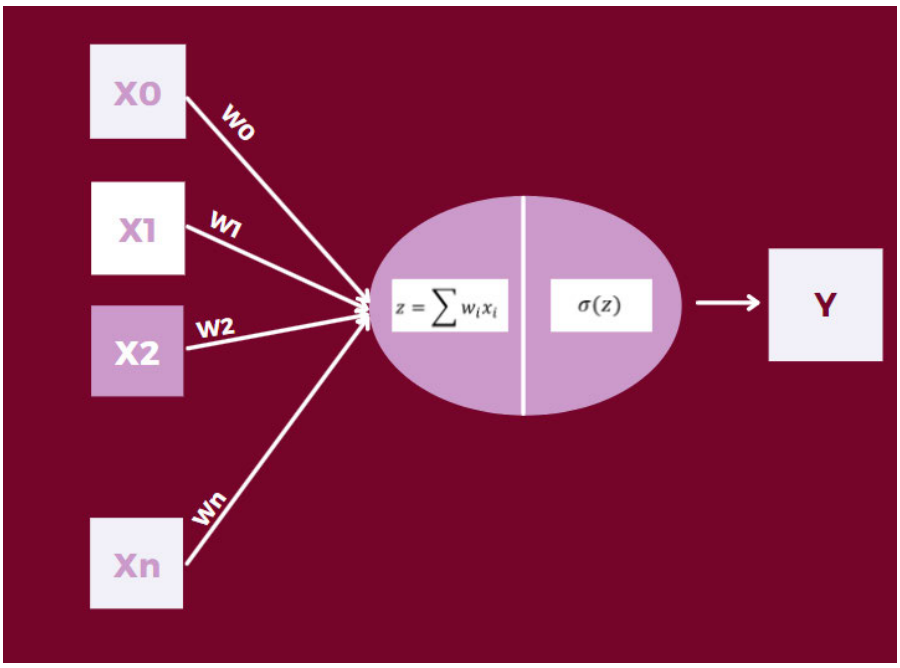
La función que relaciona la variable dependiente con las independientes, es una curva en forma de S que puede tomar cualquier valor entre 0 y 1, pero nunca valores fuera de estos límites. La respuesta ha de ser binaria y las observaciones independientes.

La ecuación que define la función es la que se muestra a continuación:

$$f(x) = \frac{1}{1 + e^{-x}} \tag{3.1}$$

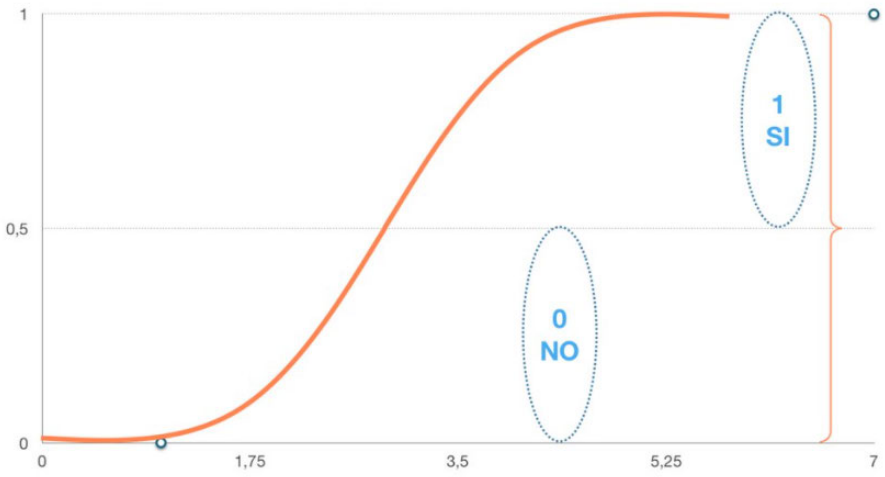
En la función X representa los distintos atributos de nuestro problema, en nuestro caso, puede ser la frecuencia de una palabra y la predicción será de que la noticia sea falsa o no. A la izquierda de la neurona tenemos una combinación lineal con los coeficientes W y a la derecha aplicamos la función logística. [7]

Figura 3.2: Representación de Regresión Logística



En la Figura 3.3 vemos representada la función. Si la salida de la función es mayor que 0.5 el resultado se clasifica como 1 y si el resultado obtenido es menor que 0.5 lo clasificamos como 0.

Figura 3.3: Función Regresión Logística



Utilizando este algoritmo tenemos las siguientes ventajas:

- Técnica eficiente y simple.

- Es muy resistente al ruido y puede evitar el sobre ajuste.
- Rápido para entrenar.
- Fácil de entrenar sobre un gran número de datos.

También contamos con algunas desventajas:

- No resuelve directamente problemas no lineales, porque la expresión que toma la decisión es lineal.
- No maneja numero grande de características.
- Los valores atípicos pueden ser un problema.
- Existen otros modelos mas potentes.

Naïve Bayes

Naïve Bayes es una técnica de clasificación y predicción supervisada que construye modelos probabilísticos basada en el Teorema de Bayes, que expresa la probabilidad de que ocurra un evento, sabiendo que también ocurre otro evento. Se asume que las variables predictorias son independientes entre sí.

La probabilidad de que ocurra un evento A, sabiendo que ha ocurrido el evento B. Es el conocimiento de que algo ya ha sucedido. Utilizando la probabilidad previa, podemos calcular la probabilidad posterior, que es la probabilidad de que ocurra el evento B dado que ha ocurrido A . El clasificador Naïve Bayes utiliza la variable de entrada para elegir la clase con la probabilidad posterior más alta. La ecuación que define la función es:

$$P(A|B) = \frac{P(B|A)P(A)}{P(A)} \tag{3.2}$$

$$P(Xi|Y) = Xi \cdot P(Yy) + (1 - Xi)(1 - P(i|Y)) \tag{3.3}$$

donde X debe ser binario.

Aplicando este algoritmo tenemos las siguientes ventajas[8]:

- Extremadamente rápido y puede hacer predicciones en tiempo real.
- Asume independencia de los datos.
- Cada característica se trata de forma independiente.
- Maneja bien características irrelevantes

Nos encontramos con algunas desventajas:

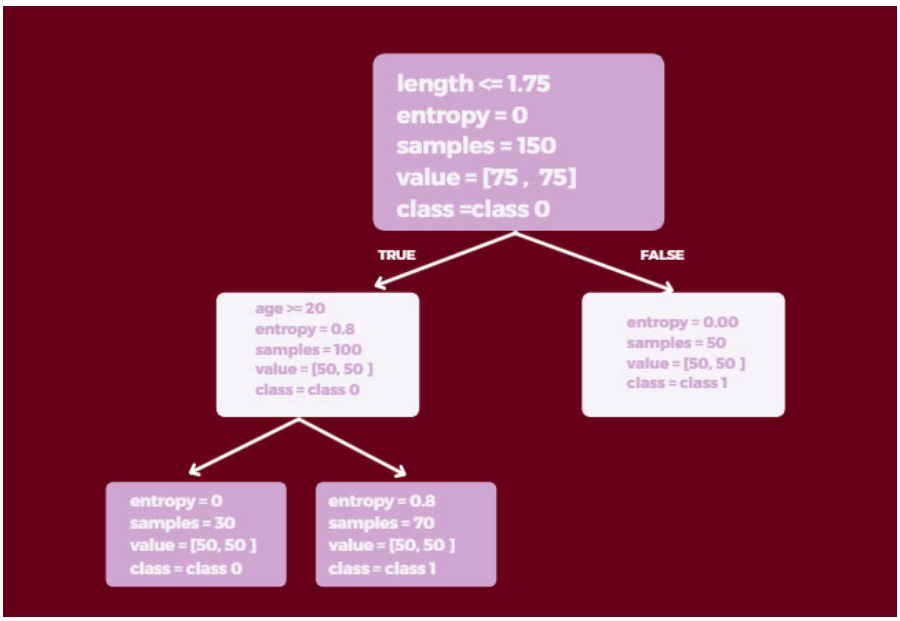
- Los predictores se consideran independientes entre sí y esto puede llevarnos a que no se ajuste correctamente a los datos.
- Al ser un clasificador ingenuo (que muestra una falta de experiencia), a veces hace una suposición basada en la forma de los datos

Naïve Bayes tiene varias implementaciones entre las cuales se encuentra "multinomial". Esta implementación es la más adecuada para las clasificaciones discretas como es contar las palabras de un texto. Por lo tanto, este algoritmo destaca sobe los otros algoritmos en la capacidad de manejar de un gran número de características.

Árbol de decisión

Los árboles de decisión son una técnica que emplea algoritmos en forma de árbol para enseñar a las máquinas a tomar decisiones y a resolver problemas de regresión o de clasificación. Obtenemos modelos predictivos precisos y fiables. Su estructura es similar a la de un diagrama de flujo donde un nodo interno representa una característica (o atributo) en función del valor del atributo, la rama representa una regla de decisión y cada nodo hoja representa el resultado.
Una vez entrenado el algoritmo, cuando se encuentra ante una disyuntiva, utiliza el conjunto de datos con el que ha sido entrenado para plantear correlaciones entre los datos de su entrenamiento y los que tiene ahora y decantarse por la opción correcta [9]. Vemos un ejemplo en la Figura 3.4:

Figura 3.4: Ejemplo árbol de clasificación



El árbol de decisión anterior esta formado por distintos elementos:

- **Nodos**: se da cuando se plantea una disyuntiva con sus diferentes opciones. Además cada nodo contiene la siguiente información:
  - **Condition**: si es un nodo donde se toma una decisión, nos muestra dicha condición.
  - **Entropy**: La entropía es la cantidad de aleatoriedad en los datos y se debe calcular para cada nodo. Nos conviene buscar árboles que tengan la entropía más pequeña en sus nodos. La entropía se utiliza para calcular la homogeneidad de las muestras en ese nodo. Muestra completamente homogénea la entropía es cero y si están divididas equitativamente tiene una entropía de uno.
  - **Samples**: número de muestras que satisfacen las condiciones necesarias para llegar a dicho nodo.
  - **Value**: número de muestras de cada clase llegan a este nodo.
  - **Class**: clase que se asigna a las muestras que llegan a ese nodo.
- **Flechas**: unen los nodos entre sí. Unidireccionales.
- **Vectores**: cada vector representa la opción por la que se opta entre todas las posibilidades que plantea el nodo.

- **Etiquetas**: une nodos y flechas y denominan las acciones que se llevan a cabo.

Estudiando este algoritmo de clasificación encontramos varias ventajas:

- Resistente a valores atípicos y faltantes.
- El tipo de datos no es una restricción.
- Selecciona las variables más importantes y en su creación no siempre se hace uso de todos los predictores.

Desventajas del árbol de decisión:

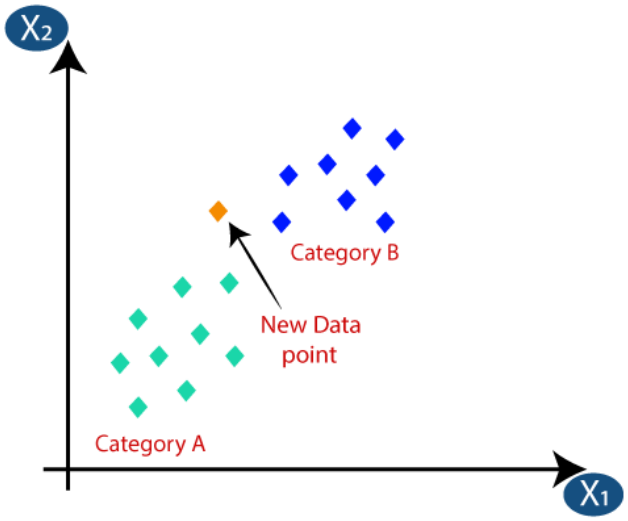
- Tienden al sobre ajuste de los datos, es decir, el modelo al predecir nuevos casos no estima con el mismo índice de acierto.
- Un pequeño cambio en los datos puede modificar la estructura del árbol y puede producir gran inestabilidad.
- Se puede dar el caso de árboles sesgados, es decir, si una de las clases domina sobre las demás.

K-Nearest-Neighbor

Es un sistema de clasificación basado en la comparación de instancias nuevas con instancias presentes en los datos de entrenamiento. Los grupos se realizan a partir del propio juego de datos de entrenamiento, tomando como referencia el parámetro k. Este algoritmo asume que todas las instancias corresponden a puntos que se encuentran en un espacio de dimensión n. El vecino más cercano de una instancia es definido en términos de la distancia Euclidiana estándar. KNN no genera un modelo fruto del aprendizaje con datos de entrenamiento, sino que el aprendizaje sucede en el mismo momento en el que se prueban los datos de test. [10][11]

En general, este modelo consta principalmente de cinco pasos:

Figura 3.5: Representación KNN

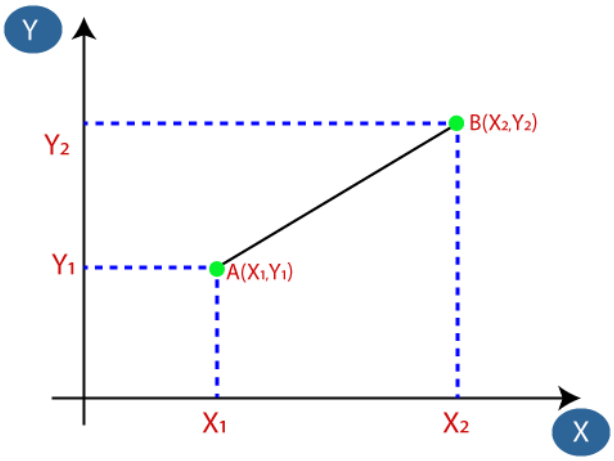


1. Seleccionar el número k de vecinos.

2. Calculamos distancia euclídea de k vecinos. Para ello aplicamos la siguiente fórmula:

$$d(Xi, Xj) = \sqrt{\sum_{n=1}^{p} (Xri - Xrj)^2} \tag{3.4}$$

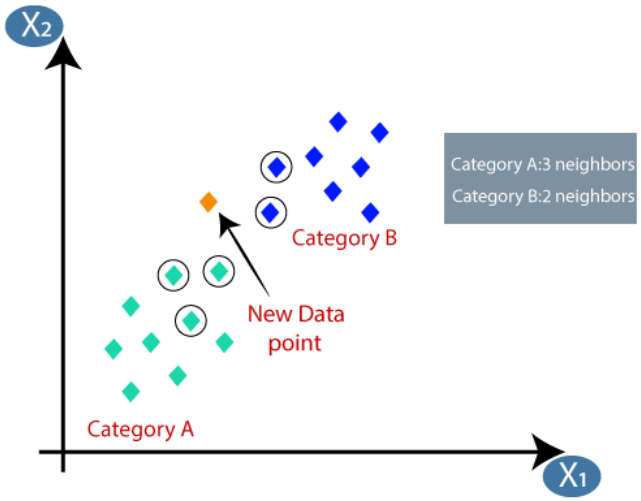
Figura 3.6: Distancia Euclídea



3. Se toman los k vecinos mas cercanos según la distancia euclídea calculada.

4. Entre esos k vecinos, contamos el número de datos que hay en cada categoría.

Figura 3.7: K Vecinos más cercanos



5. Nuestro dato lo asignamos a la categoría que tiene más vecinos o datos.

Algunas ventajas del algoritmo KNN son:

- Insensible a valores atípicos.
- Alta precisión.
- Sin suposiciones de entrada de datos.

Este algoritmo también presenta algunas desventajas:

- Computación costosa.
- Baja eficiencia, se vuelve más lento según van aumentando el número de datos, ya que su objetivo no es obtener un modelo optimizado, sino que cada instancia de prueba es comparada con todo el juego de datos de entrenamiento

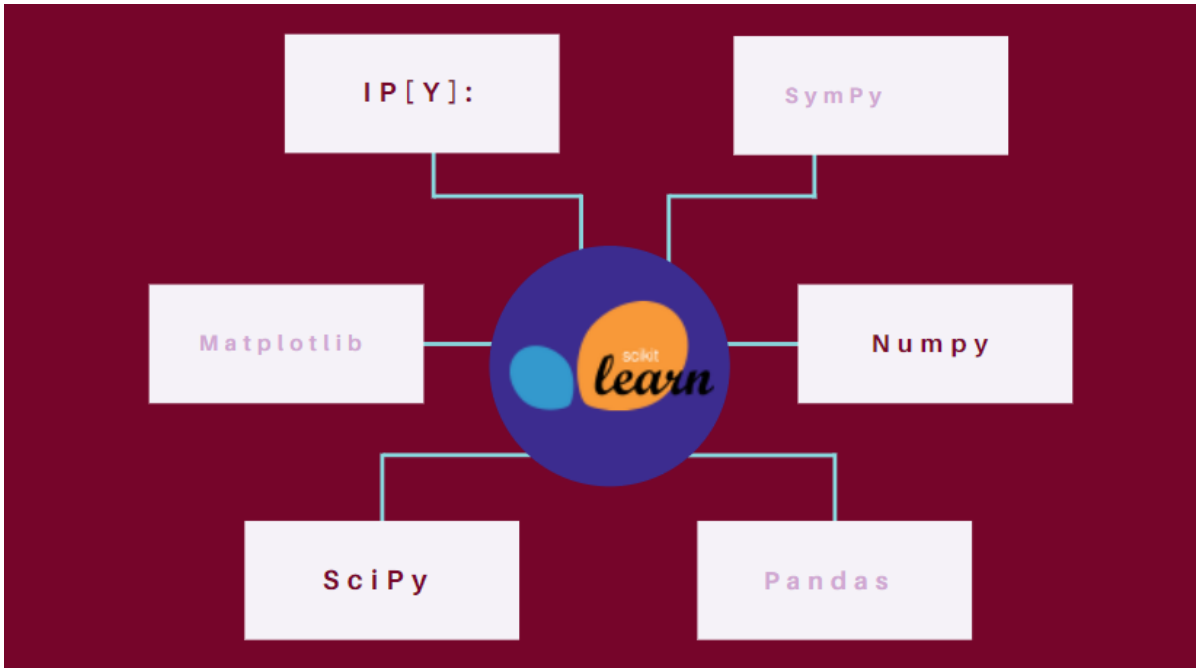
4. Metodología

En el apartado de metodología se va ha explicar todos los pasos que se han seguido para el desarrollo de este sistema y como vamos a lograr los objetivos propuestos. Todo esto realizado con nuestros datos. Para ello vamos a utilizar la plataforma "Colaboratory" [12] que nos permite ejecutar scripts de Python a través de los servidores de Google y es perfecto para implementar algoritmos de aprendizaje máquina. Nos permite utilizar diversas librerías que mas adelante se mencionan y se explican.

4.1 Procesamiento de datos

Además de ir mostrando los pasos que hemos seguido para procesar los datos, también vamos a ir viendo que librerías hemos utilizado y que beneficios nos aportan. Principalmente tenemos que destacar la librería `Scikit-Learn` [13], es la librería más completa y utilizada para Machine Learning en Python, es de código abierto y reutilizable. Esta esta compuesta por otras librerías que podemos observar en la Figura 4.1 y según vayamos utilizando algunas de ellas las explicamos más detalladamente.

Figura 4.1: Librería Scikit-Learn



1. Cargamos nuestros datos, y comprobamos que los datos se pueden ver correctamente. Ya que es importante saber que partimos de unos datos correctos. Contamos con 4009 noticias clasificadas en verdaderas y falsas, entre las cuales 2120 son falsas y 1868 son verdaderas (se tienen en cuenta las que están etiquetadas). Todos estos datos no tienen tratamiento previo para este proyecto.
   Para ello vamos a utilizar la librería `Pandas`, una herramienta de manipulación de datos de alto nivel que permite almacenar y manipular datos tabulados en filas y columnas. Algunas de sus grandes ventajas es que permite leer y escribir fácilmente ficheros en formato CSV (el que nosotros manejamos), ofrece métodos para reordenar, dividir y combinar conjuntos de datos, y maneja DataFrame (estructuras de datos de dos dimensiones).

Figura 4.2: Datos sin preprocesamiento

	URLs	Headline	Body	Label
0	http://www.bbc.com/news/world-us-canada-414191...	Four ways Bob Corker skewered Donald Trump	Image copyright Getty Images\nOn Sunday mornin...	1
1	https://www.reuters.com/article/us-filmfestiva...	Linklater's war veteran comedy speaks to moder...	LONDON (Reuters) - "Last Flag Flying", a comed...	1
2	https://www.nytimes.com/2017/10/09/us/politics...	Trump's Fight With Corker Jeopardizes His Legi...	The feud broke into public view last week when...	1
3	https://www.reuters.com/article/us-mexico-oil-...	Egypt's Cheiron wins tie-up with Pemex for Mex...	MEXICO CITY (Reuters) - Egypt's Cheiron Holdin...	1
4	http://www.cnn.com/videos/cnnmoney/2017/10/08/...	Jason Aldean opens 'SNL' with Vegas tribute	Country singer Jason Aldean, who was performin...	1
...	...	...	...	...
4004	http://beforeitsnews.com/sports/2017/09/trends...	Trends to Watch	Trends to Watch\n% of readers think this story...	0
4005	http://beforeitsnews.com/u-s-politics/2017/10/...	Trump Jr. Is Soon To Give A 30-Minute Speech F...	Trump Jr. Is Soon To Give A 30-Minute Speech F...	0
4006	https://www.activistpost.com/2017/09/ron-paul-...	Ron Paul on Trump, Anarchism & the AltRight	NaN	0
4007	https://www.reuters.com/article/us-china-pharm...	China to accept overseas trial data in bid to ...	SHANGHAI (Reuters) - China said it plans to ac...	1
4008	http://beforeitsnews.com/u-s-politics/2017/10/...	Vice President Mike Pence Leaves NFL Game Beca...	Vice President Mike Pence Leaves NFL Game Beca...	0

2. Eliminamos las columnas que no vamos a utilizar, es decir, "URL's" y "Headline". Aquí también vamos a eliminar las filas que tienen algún dato vacío, es decir, si una noticia no sabemos si es falsa o no, o si solo tenemos una etiqueta, no nos aporta ninguna información. Ahora contamos con 2120 noticias falsas y 1868 noticias verdades. Observamos que seguimos teniendo el mismo numero de noticias verdaderas y falsas, pero ahora contamos con menos datos, 3988 exactamente, esto se debe a que se han eliminado las noticias que no estaban etiquetadas.
   De nuevo hacemos uso de la librería `Pandas` y sus métodos para eliminar filas y columnas. [14]

3. Tenemos el DataFrame preparado, ahora vamos a trabajar con el procesamiento del texto de las noticias. Para poder mostrar de manera visual los pasos, se mostrarán los cambios de la primera noticia, pero los cambios se aplican a todas las noticias. La primera noticia cuenta con 1048 palabras. Podemos ver la noticia completa en la siguiente referencia 2.1.

   - Convertimos todas las mayúsculas a minúsculas, ya que no queremos distinguir entre mayúsculas y minúsculas.
     Como seguimos tratando datos, usamos la librería `Pandas`, y dentro de ella la función `.lower()`.
   - Empezamos a trabajar con las palabras una a una, y vamos a tokenizar las noticias, es decir, convertir cada palabra en un token. Vamos contando el número de tokens o palabras con las que vamos trabajando, en esta etapa contamos con 888 tokens, esta reducción de tokens se debe a que no distinguimos entre mayúsculas y mayúsculas.
     Para tokenizar nuestras noticias utilizamos de la librería NLTK es un conjunto de librerías y programas para Python que nos permiten llevar a cabo tareas relacionadas con NLP. Estas tareas ya están desarrolladas y las podremos usar directamente. Aquí

usaremos `word_tokenize` y le iremos pasando una a una las noticias para que las convierta en tokens.

- Eliminamos las "stopwords", eso sí, las descargamos en inglés ya que todas nuestras noticias son en inglés y las eliminamos.Veamos un ejemplo de "stopwords" para entenderlo mejor:

Figura 4.3: Stopwords en Inglés

```
['i', 'me', 'my', 'myself', 'we', 'our', 'ours', 'ourselves', 'you', "you're", "you've", "you'll",
```

Antes de eliminarlas, vamos a ver cuales son las palabras mas frecuentes que tenemos y de las 10 palabras mas frecuentes, 9 son stopwords :

Figura 4.4: Densidad de palabras

Densidad de palabras	
'the'	44 (5%)
's	25 (3%)
'to'	20 (2%)
'of'	19 (2%)
'mr'	17 (2%)
'a'	16 (2%)
'and'	16 (2%)
'in'	15 (2%)
'president'	15 (2%)
'he'	15 (2%)

Concretamente vamos a importar `stopwords` y convertirlos en tokens, perteneciente a la librería NLTK , esto lo que hará será descargarnos las palabras y después las eliminaremos de nuestros tokens.

- Stemming (reducir las palabras a su raíz) y Lematización.
  Al eliminar las stopwords y aplicar stemming y lematización a los tokens, ahora contamos con 323 tokens.
  Volvemos a hacer uso de NLTK, para Stemming importamos `PorterStemmer` y su correspondiente función `.stem(palabra)`. En cuanto a la Lematización importamos

`WordNetLemmatizer` y su correspondiente función `.lematize(palabra)`. Ambas funciones las vamos aplicando a todas las palabras de una noticia.

- Eliminación de números y signos de puntuación.
  Se usa la librería `Pandas` y la función `str.isalpha()`, con esto lo que conseguimos es eliminar todas los tokens que no sean de carácter alfabético.

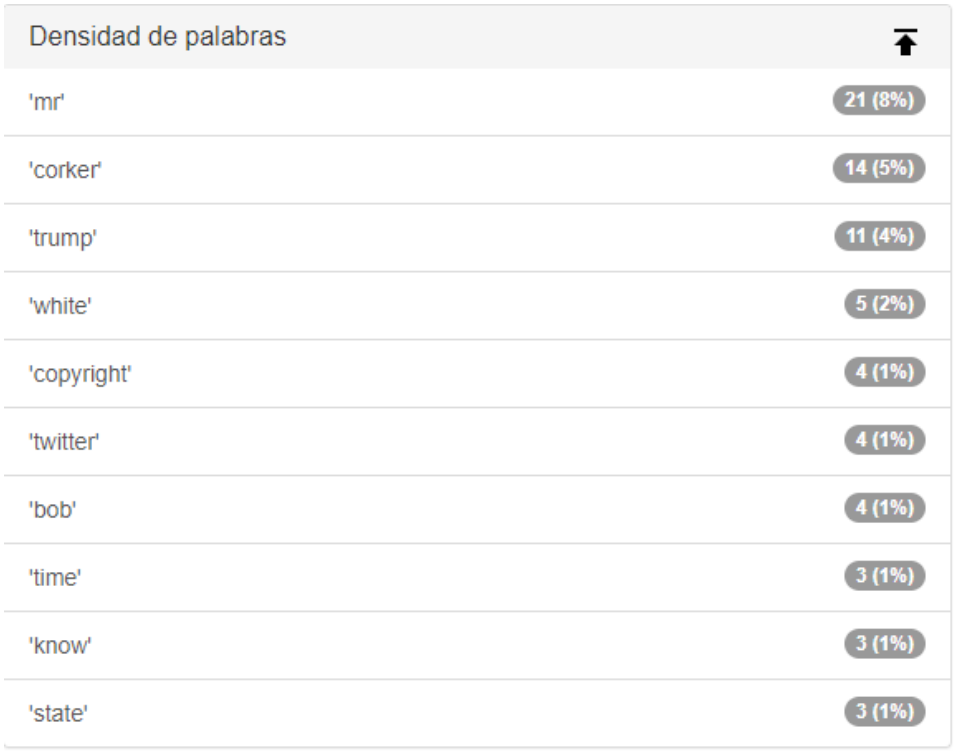
En la Figura 4.5 se muestran los tokens de nuestra noticia, ya procesados y listos para ser utilizados con los algoritmos.Ahora ya contamos con 276 palabras.

Figura 4.5: Texto final

['copyright', 'sunday', 'donald', 'trump', 'went', 'twitter', 'member', 'huge', 'news', 'far', 'first', 'time', 'time', 'bob', 'corker', 'chair', 'foreign', 'coward', 'said', 'mr', 'corker', 'give', 'mr', 'corker', 'support', 'iranian', 'nuclear', 'agreement', 'mr', 'corker', 'free', 'hold', 'skip', 'twitter', 'post', 'shame', 'white', 'adult', 'day', 'care', 'center', 'shift', 'bob', 'corker', '8', '2017', 'report', 'end', 'though', 'spoke', 'new', 'york', 'let', 'four', 'interview', 'know', 'true', 'know', 'ca', 'sugarcoat', 'one', 'mr', 'corker', 'liar', 'particular', 'mr', 'trump', 'much', 'broader', 'mr', 'corker', 'akin', 'tennessean', 'mr', 'trump', 'short', 'list', 'state', 'copyright', 'caption', 'bob', 'corker', 'trump', 'campaign', '2016', 'much', 'like', 'mr', 'corker', 'soon', 'although', 'margin', 'success', 'chamber', 'stand', 'afford', 'lose', 'two', '52', 'republican', 'reform', 'could', 'bad', 'news', 'tax', 'mr', 'corker', 'look', 'except', 'vast', 'congress', 'feel', 'agenda', 'come', 'repeal', 'reform', 'boost', 'enough', 'harder', 'one', 'shake', 'subject', 'campaign', 'slow', 'grind', 'copyright', 'caption', 'corker', 'state', 'rex', 'tillerson', 'congress', 'background', 'among', 'like', 'mr', 'lament', 'understand', 'mr', 'corker', 'loud', 'clear', 'almost', 'kept', 'silent', 'still', 'hope', 'sign', 'fear', 'mr', 'trump', 'back', 'next', 'year', 'stay', 'home', 'ever', 'riskier', 'stay', 'silent', 'speak', 'mr', 'trump', 'real', 'lot', 'think', 'kind', 'cop', 'bad', 'cop', 'act', 'underway', 'true', 'time', 'mr', 'trump', 'undercut', 'state', 'rex', 'tillerson', 'use', 'soft', 'deal', 'war', 'taliban', 'afghanistan', 'iran', 'nuclear', 'agreement', 'qatar', 'persian', 'gulf', 'unrest', 'venezuela', 'north', 'korea', 'target', 'offhand', 'twitter', 'said', 'part', 'mr', 'trump', 'version', 'madman', 'give', 'way', 'fear', 'leader', 'mr', 'corker', 'mr', 'trump', 'senior', 'avoid', 'know', 'fact', 'day', 'white', 'contain', 'john', 'testament', 'mr', 'trump', 'white', 'mr', 'arch', 'nose', 'stave', 'copyright', 'caption', 'white', 'chief', 'staff', 'john', 'donald', 'trump', 'campaign', 'north', 'korean', 'speech', 'mr', 'face', 'white', 'team', 'often', 'left', 'explain', 'joke', 'remark', 'line', 'even', 'though', 'mr', 'brought', 'west', 'wing', 'staff', 'still', 'beat', 'drum', 'access', 'phone', 'twitter', 'app', 'bob', 'corker', 'latest', 'person', 'politician', 'journalist', 'star', 'feel', 'ire']

Volvemos a comprobar cuales son las palabras más frecuentes en las 276 palabras que nos quedan y como podemos observar en la Figura 4.6 todas nos aportan información. Por lo tanto se ha realizado un procesamiento de datos correcto y hemos obtenido los resultados que queríamos.

Figura 4.6: Densidad de palabras final



4.2 Algoritmos de clasificación

Anteriormente hemos hablado de los distintos tipos de algoritmos de clasificación que podemos utilizar para trabajar con nuestros datos sobre noticias falsas. Vamos a tratar con clasificación binaria, ya que tratamos de predecir si una noticia es falsa o no, solo tenemos dos posibles clases para clasificar una noticia.

En este apartado iremos aplicando los distintos algoritmos sobre nuestros datos y estudiando la información que vamos obteniendo de cada uno de ellos.

Para trabajar con las noticias, es necesario convertir el texto en un vector, es decir, generar el vector de frecuencia de palabras.Para ello utilizamos la librería `sklearn` con su correspondiente librería `CountVectorizer`, que proporciona una manera simple de tokenizar una colección textos y construir un vocabulario de palabras conocidas[15]. Se crea una instancia de la clase `CountVectorizer`. Se llama a la función `fit_transform()` para aprender un vocabulario de un documento o más y codificar cada uno como un vector.

Figura 4.7: Código para BOW

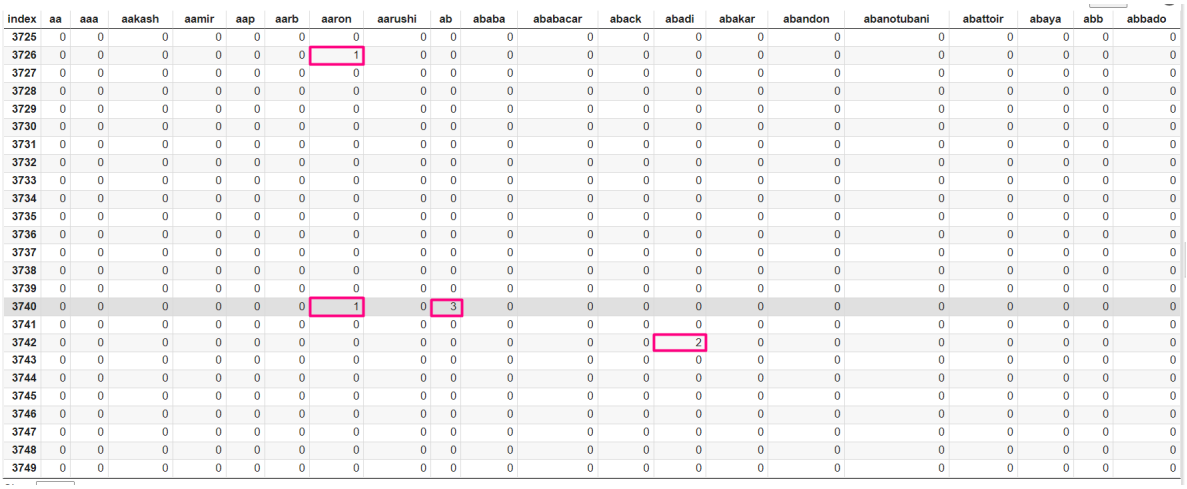
```
import numpy as np
import pandas as pd
import os
from sklearn.feature_extraction.text import CountVectorizer

data =datos['Body'].astype(str)
datosTRATADOS = countvec.fit_transform(data)
bow = pd.DataFrame(datosTRATADOS.toarray(), columns = countvec.get_feature_names())
```

```
/usr/local/lib/python3.7/dist-packages/sklearn/utils/deprecation.py:87: FutureWarning: Function
  warnings.warn(msg, category=FutureWarning)
```

Se devuelve un vector codificado con una longitud de todo el vocabulario y un número entero para el número de veces que cada palabra apareció en el documento.

Figura 4.8: Nuestra BOW de noticias



index	aa	aaa	aakash	aamir	aap	aarb	aaron	aarushi	ab	ababa	ababacar	aback	abadi	abakar	abandon	abanotubani	abattoir	abaya	abb	abbado
3725	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
3726	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0
3727	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
3728	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
3729	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
3730	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
3731	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
3732	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
3733	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
3734	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
3735	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
3736	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
3737	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
3738	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
3739	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
3740	0	0	0	0	0	0	1	0	3	0	0	0	0	0	0	0	0	0	0	0
3741	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
3742	0	0	0	0	0	0	0	0	0	0	0	0	2	0	0	0	0	0	0	0
3743	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
3744	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
3745	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
3746	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
3747	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
3748	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
3749	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

A continuación seguimos con la parte de entrenamiento, en esta etapa, se necesita disponer de suficiente cantidad y calidad de los datos para que el algoritmo sea capaz de extraer los patrones necesarios porque la idea es transferir el conocimiento de los datos al algoritmo.Después tenemos que dividir los datos en conjuntos de entrenamiento y prueba, y usaremos la librería `sklearn.model_selection` y la función `train_test_split()` que se encarga de dividir matrices de datos en dos, el conjunto de datos se divide en una proporción de 75:25. Significa que el 75% de los datos se utilizarán para el entrenamiento de modelos y el 25% para las pruebas de modelos. [16]

Figura 4.9: Train_test_split

```
from sklearn.model_selection import train_test_split

X = datosTRATADOS.toarray()
y = np.array(datos.iloc[:, 1])

X_train, X_test, y_train, y_test = train_test_split(X, y)
```

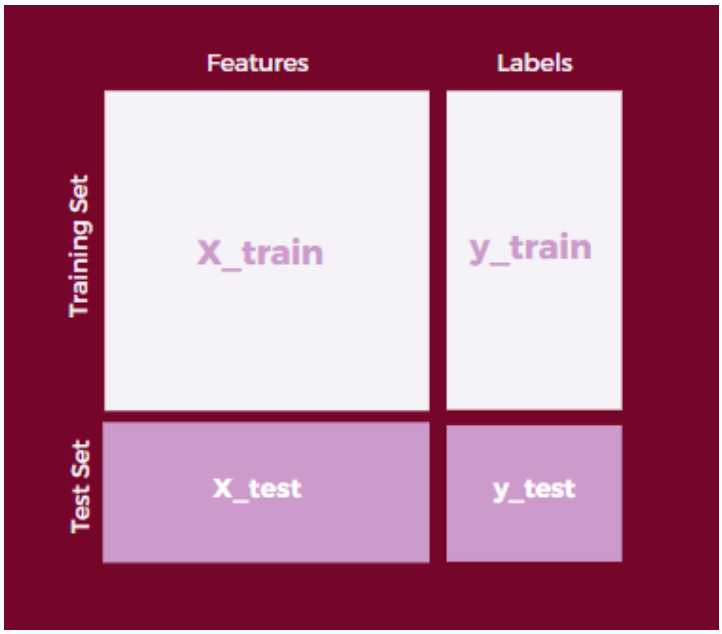
Las variables utilizadas son:

- **X_train**: registros para entrenar.
- **y_train**: son las "etiquetas" de los resultados esperados de X_train.
- **X_test**: registros para test.

- **y_test**: son las "etiquetas" de los resultados de X_test.

En la Figura 4.10 se representa de manera gráfica la división de datos anterior.

Figura 4.10: Representación gráfica de la división de datos



Construyamos los modelos de predicción de Machine Learning. Se predice si una noticia es falsa o no:

Regresión Logística

Construimos el modelo de predicción de las noticias falsas. Se trata de utilizar el clasificador de regresión logística. Importamos el módulo de Regresión Logística utilizando la función de `LogisticRegression()`, en la cuál tenemos el parámetro `max_iter` para indicar el número máximo de iteraciones que hará el algoritmo. El modelo se ajusta en el conjunto de X_train Y_train y se realiza la predicción usando la función `predict()`, todas estas funciones importadas de la librería `LogisticRegression`. Se ajusta el modelo en el conjunto de entrenamiento utilizando `fit()` y se realiza la predicción en el conjunto de prueba usando `predict()` [17].

Figura 4.11: Código de Regresión Logística

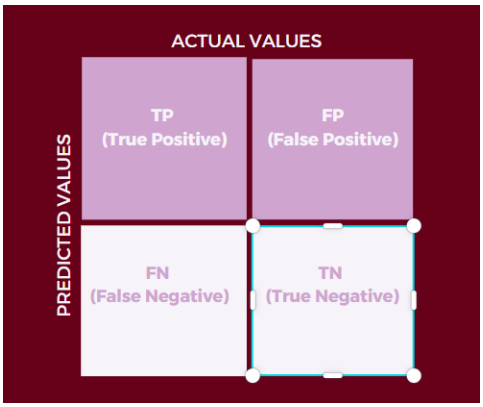
```
from sklearn.linear_model import LogisticRegression

logistic_regression = LogisticRegression( solver='lbfgs', max_iter=4000)

logistic_regression.fit(X_train, y_train)
y_predict = logistic_regression.predict(X_test)
```

Creamos la matriz de confusión, que es una tabla que describe el rendimiento de un modelo supervisado. Lo importante es el número de predicciones correctas e incorrectas. La matriz de confusión es de 2 x 2 por que trabajamos con una clasificación binaria. En la Figura 4.12 representamos de manera esquemática como es esta matriz de confusión. Los valores diagonales representan predicciones reales (TP y TN) y los dos elementos no diagonales (FP y FN) son predicciones incorrectas:

Figura 4.12: Matriz de Confusión



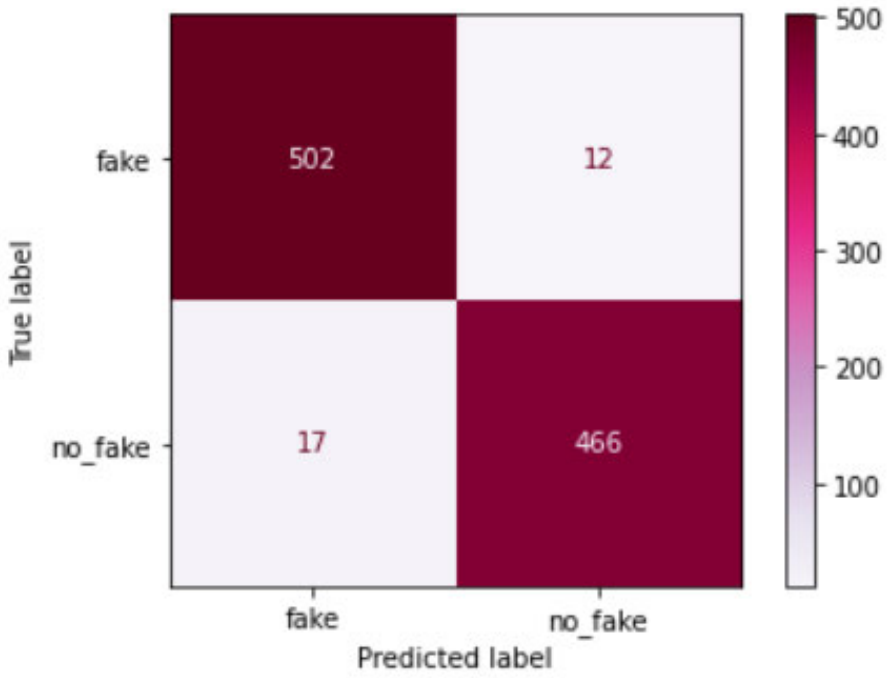
Significado de las variables mostradas en la Figura 4.12 es:

- **True Positive o Verdadero Positivo**: se predijo positivo y el valor real es positivo.

- **True Negative o Verdadero Negativo**: se predijo negativo y el valor real es negativo.
- **False Positive o Falso Positivo**: se predijo positivo y el valor real es negativo. En estadística se conoce como error tipo I.
- **False Negative o Falso Negativo**: se predijo negativo y el valor real es positivo. En estadística se conoce como error tipo II.

Creamos la matriz de confusión con nuestros datos (Figura 5.1) con la librería `sklearn.matrix` y la función `confusion_matrix()`. Se observa que 502 y 446 predicciones son reales, y 17 y 12 predicciones son incorrectas :

Figura 4.13: Matriz de Confusión



Además de la matriz de confusión también contamos con diferentes medidas numéricas con las cuales podremos evaluar el modelo:

- **Accuracy**: métrica que evalúa la fracción de predicciones que el modelo acertó [18]. Se usa la librería `sklearn.metrics` y la función `classification_report` :

$$Accuracy = \frac{Nmero\ de\ predicciones\ correctas}{Nmero\ total\ de\ predicciones} \tag{4.1}$$

Pero como estamos en clasificación binaria, también hay otra manera de calcularla usando términos positivos y negativos, es decir, calculando la cantidad de predicciones positivas que fueron correctas.

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} = \frac{957}{997} = 0,96 \tag{4.2}$$

- **precision**: nos muestra la proporción de identificaciones positivas que fueron realmente correctas, es decir, el porcentaje de noticias verdaderas clasificadas correctamente.

$$Precision = \frac{TP}{TP + FP} = \frac{511}{511 + 15} = \frac{511}{526} = 0,97 \tag{4.3}$$

- **Recall**: muestra la proporción de positivos reales que se identificaron correctamente, es decir, mide el porcentaje de noticias falsas que se clasificaron correctamente.

$$Recall = \frac{TP}{TP + FN} = \frac{511}{511 + 25} = \frac{511}{536} = 0,95 \tag{4.4}$$

  Cabe destacar que cuando la variable de precisión aumenta , la variable Recall se reduce y viceversa, es decir, si el numero de falsos positivos disminuye, el de falsos negativos aumenta.

- **Support**: es el numero de ocurrencias de una clase dada en su subconjunto de datos. Tenemos 531 datos en una clase y 466 en la otra clase, por lo tanto, el conjunto esta bien equilibrado ya que no hay mucha diferencia.

- **F1-score**: media armónica entre precision y recall.

$$F1 = 2 * \frac{precision * recall}{precision + recall} \tag{4.5}$$

- **Macro avg**: calcula un promedio de las precisiones de todas las clases.

- **Weighted avg**: también se calcula en función de la Precisión por clase, pero tiene en cuenta el número de muestras de cada clase en los datos

Resumen de los datos obtenidos de las variables anteriores:

Figura 4.14: Resumen Regresión Logística

```
[60] from sklearn.metrics import classification_report
     target_names = ['fake', 'no_fake']
     print(classification_report(y_predict, y_test, target_names=target_names))
```

```
                precision    recall  f1-score   support

          fake       0.98      0.97      0.97       519
       no_fake       0.96      0.97      0.97       478

      accuracy                           0.97       997
     macro avg       0.97      0.97      0.97       997
  weighted avg       0.97      0.97      0.97       997
```

También se ha hecho un resumen con los datos obtenidos aplicando regresión logística sin el preprocesamiento de los datos y con su correspondiente matriz de confusión:

Figura 4.15: Resumen Regresión Logística sin preprocesamiento de datos

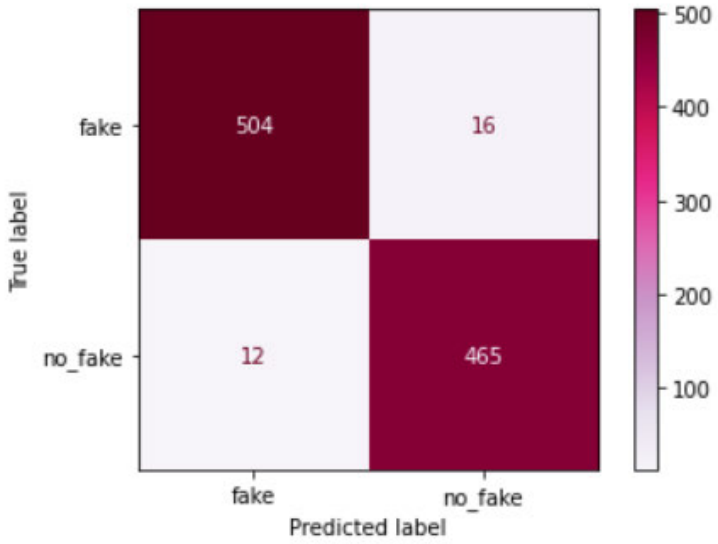
```
[8] from sklearn.metrics import classification_report
    target_names = ['fake', 'no_fake']
    print(classification_report(y_predict, y_test, target_names=target_names))
```

```
                precision    recall  f1-score   support

          fake       0.97      0.98      0.97       516
       no_fake       0.97      0.97      0.97       481

      accuracy                           0.97       997
     macro avg       0.97      0.97      0.97       997
  weighted avg       0.97      0.97      0.97       997
```

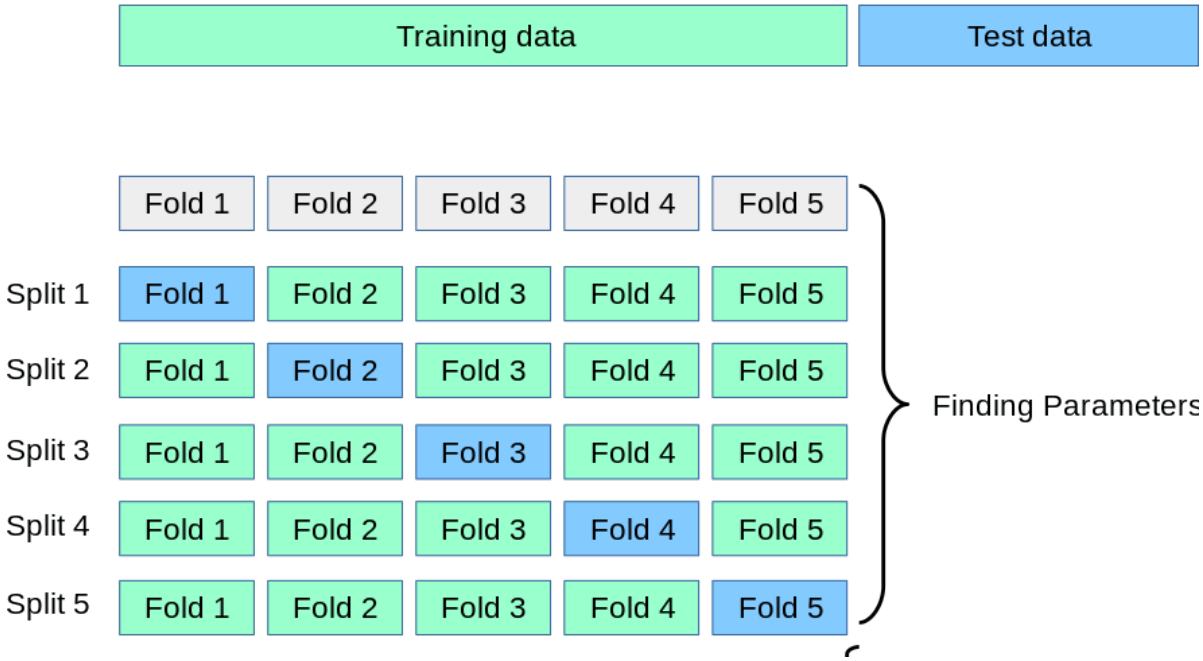
Figura 4.16: Matriz Regresión Logística sin preprocesamiento de datos



Cross Validation:

En este apartado realizaremos la técnica de validación cruzada para poder garantizar la independencia de la partición de los datos de entrenamiento y prueba. Consiste en repetir esa división y calcular la media aritmética de las medidas de evaluación sobre diferentes particiones. Se espera que este resultado medio sea una estimación mas precisa y así reducir el error de estimación. Se utiliza en entornos donde el objetivo principal es la predicción. En la Figura 4.17 se muestra el procedimiento que se realizará a continuación para obtener los resultados. Además de ello podemos elegir los distintos parámetros que queremos probar para así obtener la combinación que mas beneficie nuestro conjunto de datos.

Figura 4.17: Funcionamiento Cross Validation



Para ello vamos a utilizar librerías y funciones de sklearn como `GridSearchCV()` en la que configuramos los siguientes parámetros [19]:

- **estimador**: el modelo que se va a evaluar.
- **param_grid**: se indican los parámetros que queremos evaluar.
- **cv**: el numero de conjuntos en los que vamos a dividir nuetro juego de datos para la evaluación cruzada.

Antes que nada, tenemos que configurar los parámetros con los que queremos probar y el propio algoritmo nos devolverá la combinación de parámetros más óptima. Como tratamos con Regresión Logística el parámetro y "max_iter":

Figura 4.18: Entrenando modelo con Regresión Logística

```
[25] from sklearn.model_selection import GridSearchCV

     parameters = {'max_iter' : [100,500,1000,2000,3000,4000]}

     grid_search_dt = GridSearchCV(estimator = logistic_regression,
     param_grid = parameters,
     scoring = 'accuracy',
     cv = 5,
     verbose = 1)

     grid_search_dt.fit(X_train, y_train)
```

```
Fitting 5 folds for each of 6 candidates, totalling 30 fits
GridSearchCV(cv=5, estimator=LogisticRegression(max_iter=4000),
             param_grid={'max_iter': [100, 500, 1000, 2000, 3000, 4000]},
             scoring='accuracy', verbose=1)
```

Obtenemos un Acurracy bastante alto de un 96,79%:

Figura 4.19: Entrenando modelo con Regresión Logística

```
print(grid_search_dt.score(X_test, y_test))
```

```
0.970912738214644
```

Naive Bayes

Construimos el modelo de predicción de noticias falsas con el algoritmo de Naïve Bayes, concretamente el modelo Multinomial. Para ello comenzamos entrenando el modelo con la librería `MultinomialNB` y su correspondiente función `MultinomialNB()` y `fit()` para entrenar el modelo, aquí como necesitamos representar las muestras como vectores con valores binarios, podemos destacar la variable binarize que se encarga de binarizar nuestros datos. En nuestro caso tenemos nuestra BOW, con valores binarios, por lo tanto indicamos binarize=none, para que no nos binarize nuestros datos.

Figura 4.20: Entrenando modelo con MultinomialNB de Naive Bayes

```
from sklearn.naive_bayes import MultinomialNB
naive_bayes = MultinomialNB( alpha=1.0, fit_prior=True, class_prior=None)
naive_bayes.fit(X_train, y_train)
```

Con la función `predict()` haremos la predicción. Este algoritmo es bastante simple ya que no requiere de mas configuración. En la Figura 4.21 podemos ver los resultados obtenidos y vemos que tenemos 0.95 en accuracy, es decir, el modelo acierta solo un 95% de las veces.

Figura 4.21: Resultados de MultinomialNB de Naive Bayes (datos procesados)

```
from sklearn.metrics import classification_report
target_names = ['fake', 'no_fake']
print(classification_report(y_predict, y_test, target_names=target_names))
```

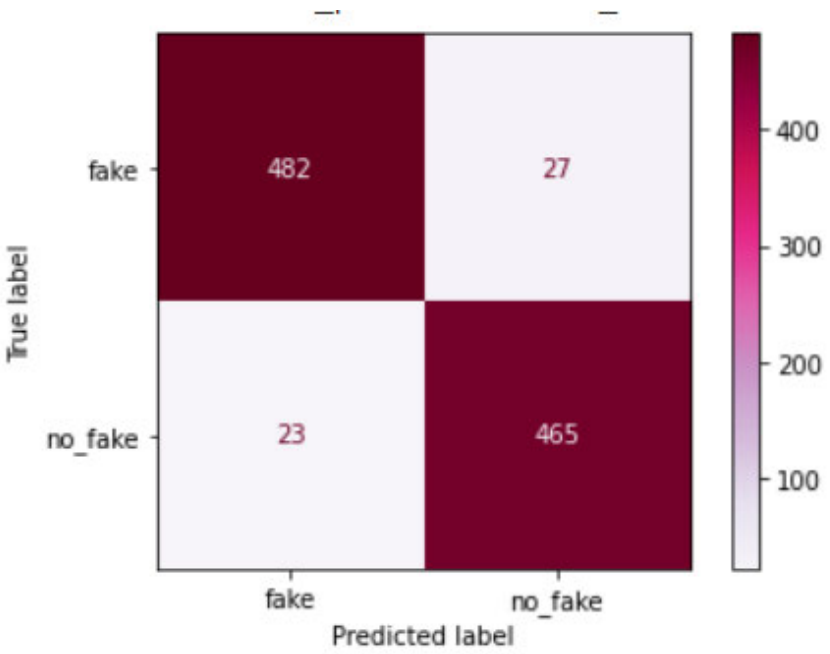
```
              precision    recall  f1-score   support

        fake       0.95      0.95      0.95       505
     no_fake       0.95      0.95      0.95       492

    accuracy                           0.95       997
   macro avg       0.95      0.95      0.95       997
weighted avg       0.95      0.95      0.95       997
```

También mostramos la matriz de confusión para poder entender mejor el resultado anterior obtenido y vemos que tenemos un numero elevado de verdaderos positivos (482) y verdaderos negativos (465) frente a los falsos positivos (27) y falsos negativos (23).

Figura 4.22: Matriz de confusión de MultinomialNB de Naive Bayes (datos procesados)



A continuación hacemos lo mismo pero con los datos sin procesar. Aquí también obtenemos un valor similar de accuracy. Un 97% de las predicciones serán correctas.

Figura 4.23: Resultados de MultinomialNB de Naive Bayes (datos sin procesar)

```
from sklearn.metrics import classification_report
target_names = ['fake', 'no_fake']
print(classification_report(y_predict, y_test, target_names=target_names))
```

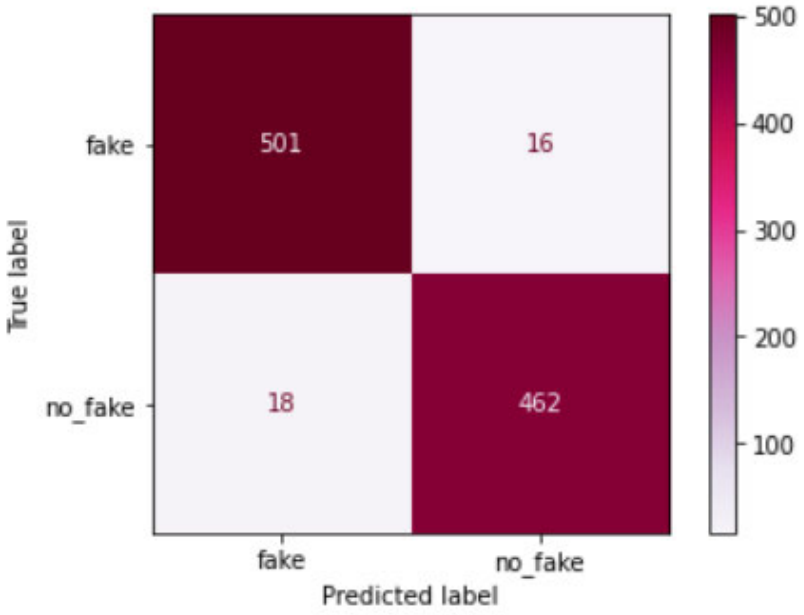
```
              precision    recall  f1-score   support

        fake       0.97      0.97      0.97       519
     no_fake       0.96      0.97      0.96       478

    accuracy                           0.97       997
   macro avg       0.97      0.97      0.97       997
weighted avg       0.97      0.97      0.97       997
```

En esta matriz de confusión solo contamos con 34 predicciones incorrectas:

Figura 4.24: Matriz de confusión de MultinomialNB de Naive Bayes (datos sin procesar)



Cross Validation:

En primer lugar lo que haremos es configurar los parámetros, aunque en este caso solo podemos hacer pruebas con el parametro "alpha", por que tenemos los datos ya binarizados. A continuación elegimos una cross validation con 5 iteraciones.

Figura 4.25: Grid Search Naive Bayes (datos procesados)

```
from sklearn.model_selection import GridSearchCV

parameters = {
'alpha' : [0,0.5,1],
'fit_prior' : [bool,True],
'class_prior' : [1,5,7,10, None]}

grid_search_dt = GridSearchCV(estimator = naive_bayes,
param_grid = parameters,
scoring = 'accuracy',
cv = 5,
verbose = 1)

grid_search_dt.fit(X_train, y_train)
```

Obtenemos un 94,38%, es decir de cada cien noticias predicadas, solo unas 94 noticias serán pre-

dichas de manera correcta. Un resultado que se acerca bastante a lo que nosotros buscamos.

Figura 4.26: resultados Grid Search Naive Bayes (datos procesados)

```
print(grid_search_dt.score(X_test, y_test))
```

```
0.9438314944834504
```

```
print(grid_search_dt.best_estimator_)
```

```
MultinomialNB(alpha=1, fit_prior=<class 'bool'>)
```

Árbol de decisión

Construimos el modelo de predicción de noticias falsas con el algoritmo de Tree Decision y para ello utilizamos la librería y función `DecisionTreeClassifier`, en la cual podemos ajustar los siguientes parámetros:

- **Criterion**: mide la calidad de una división. Los criterios admitidos son ”gini” para la impureza de Gini y ”entropy” para la ganancia de información.
- **Splitter**: indica la estrategia utilizada para indicar la división en cada nodo. Se puede utilizar ”best” para elegir la mejor división o ”random” para elegir la mejor división aleatoria.
- **Max_depth**: indica la profundidad máxima del árbol. Si indicamos ninguno, los nodos se expanden hasta que todas las hojas sean puras.
- **Random_state**: controla la aleatoriedad del estimador. Las características siempre se permutan aleatoriamente en cada división, incluso si splitter es ”best”. El algoritmo selecciona al azar en cada división antes de encontrar la mejor división entre ellas. Pero la mejor división encontrada puede variar entre diferentes ejecuciones.

Figura 4.27: Código de Predicción con Árbol de decisión

```
from sklearn.tree import DecisionTreeClassifier
clf = DecisionTreeClassifier(criterion='entropy', splitter='random', max_depth=3, random_state=43)
clf = clf.fit(X_train,y_train)
```

Una vez entrenado nuestro modelo, vamos a mostrar el reporte con los datos que hemos obtenido. En la Figura 4.28 se muestra el reporte y en la Figura 4.29 la matriz de confusión obtenida, ambos con los datos procesados:

Figura 4.28: Classification_report

```
from sklearn.metrics import classification_report
target_names = ['fake', 'no_fake']
print(classification_report(y_predict, y_test, target_names=target_names))
```

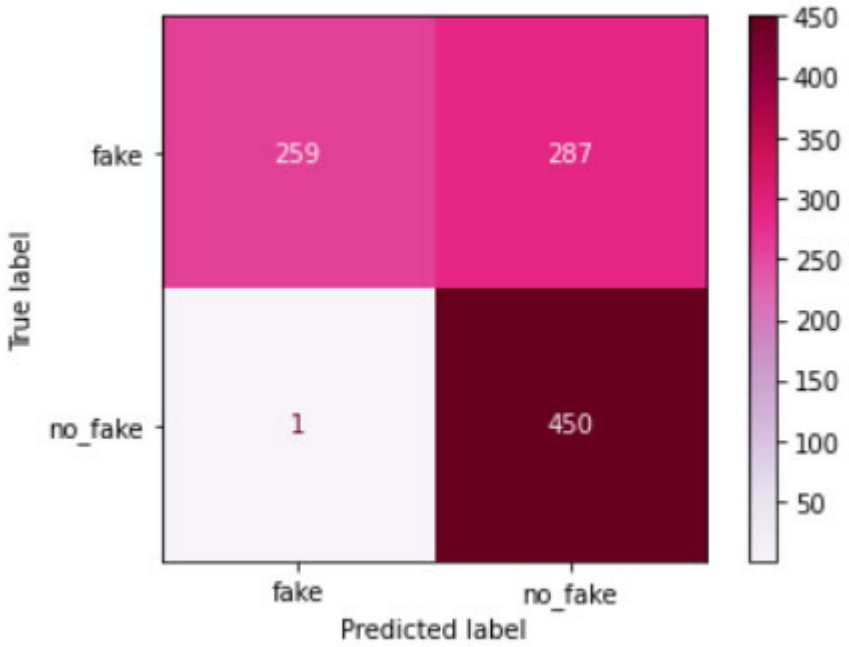
```
              precision    recall  f1-score   support

        fake       0.47      1.00      0.64       260
     no_fake       1.00      0.61      0.76       737

    accuracy                           0.71       997
   macro avg       0.74      0.80      0.70       997
weighted avg       0.86      0.71      0.73       997
```

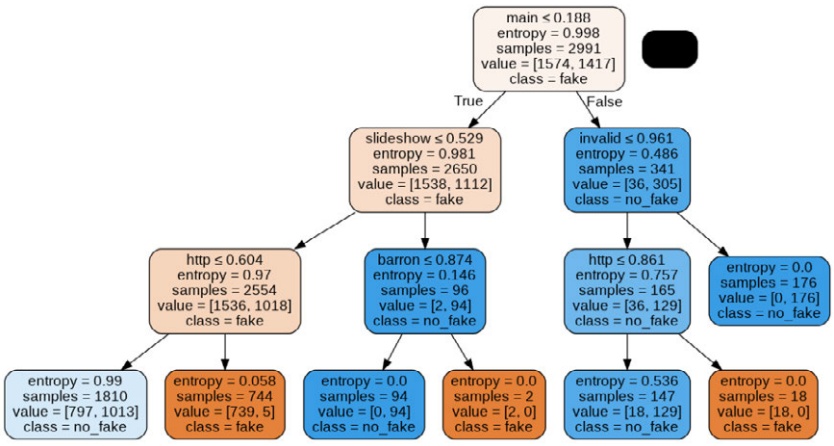
En esta matriz de confusión podemos destacar que los verdaderos negativos son mucho mas que los falsos negativos y que por lo tanto obtenemos demasiadas predicciones incorrectas. El modelo predijo que 260 noticias pertenecen a la clase ”fake” y 737 pertenecen a la clase ”no_fake” y la realidad es que 451 pertenecen a la clase ”fake” y 546 pertenecen a la clase ”no_fake”. Lo ideal seria disminuir el numero de falsos positivos.

Figura 4.29: Matriz de confusión



En la Figura 4.30 mostramos el árbol de decisión que hemos obtenido. El objetivo es que los nodos queden lo mas puros posibles, es decir, que cada nodo quede lo mas concentrado posible a una sola clase. Para la optimización del árbol de decisión es necesario realizar poda. y para ello se puede utilizar la profundidad máxima del árbol. En este caso hemos elegido una profundidad de tres. En cuanto a la selección de atributos, hay varías medidas, entre ellas la entropía que nos ayuda a la ganancia de información, nos indica cuanta información nos proporciona una característica o atributo. Para generar este árbol se han utilizado varias librerías y funciones como `plt` , `metrics`, `export_graphviz`, `StringIO`, `Image`.

Figura 4.30: Árbol de decisión de profundidad 3 (datos procesados)



Para entender el resultado obtenido de la Figura 4.30, vamos a estudiar mas en profundidad el árbol y a entender toda la información que nos proporciona:

- En cada nodo nos encontramos en primer lugar con un nodo raíz el cual realizará la primera subdivisión por la palabra 'main' y en caso de que el valor sea menor o igual a 0.188 la salida irá a "true" y en caso contrario a "false". También nos indica que este primer nodo divide en muestras de 2650 en 'true' y las 341 restantes en 'false'. A medida que seguimos estudiando el árbol de decisiones, veremos que los valores de entropía se acerca mas a 1 cuando tiene mas muestras pertenecientes a la clase "fake" y cuando se acerca mas a 0, tiene mas muestras pertenecientes a la clase "no_fake". A lo largo de los niveles vemos divisiones por las importancia de las distintas palabras, sucede que en algunas hojas finalizan antes de llegar al último nivel (en este caso 3), esto sucede por que alcanzan un nivel de entropía 0 o por que la cantidad de muestras pertenecientes a la otra clase es mínima. Concluimos que el algoritmo para saber si una noticia es falsa o no, empieza desde la raíz del y según el valor de las palabras va siguiendo el árbol hasta llegar a un nodo final que nos clasifica la noticia. En este caso las palabras que tienen una mayor importancia son "main" , "slideshow" ,"valid" , "http" y "barron", por lo tanto son las que mas influyen en la clasificación.

Creamos una tabla en la cual vamos a ir mostrando el valor de Accuracy que vamos obteniendo con árboles de distintas profundidades. Según aumenta la profundidad del árbol puede pasar que sobre entrenemos el árbol. La profundidad máxima teórica que puede alcanzar un árbol de decisión es uno menos que el número de muestras de entrenamiento, pero no llega a ese punto por que se producirá sobreajuste. En el caso de "default", el árbol se expande hasta que todas las hojas tengan solo etiquetas.

Nivel	Accuracy
max_depth = 3	0.72
max_depth = 5	0.77
max_depth = 7	0.84
max_depth = 9	0.90
deafult	0.95

Cuadro 4.1: Tabla de pruebas Árbol de decisiones (datos procesados)

A continuación repetimos el reporte de los datos obtenidos y la matriz de confusión pero ahora están hechos con los datos sin procesar. Aquí observamos que hay muchas menos predicciones incorrectas comparado con las predicciones que se hicieron con los datos procesados. Pero a la vez acurracy o el porcentaje de predicciones correctas es muy similar.

Figura 4.31: Classification_report

```
from sklearn.metrics import classification_report
target_names = ['fake', 'no_fake']
print(classification_report(y_predict, y_test, target_names=target_names))
```

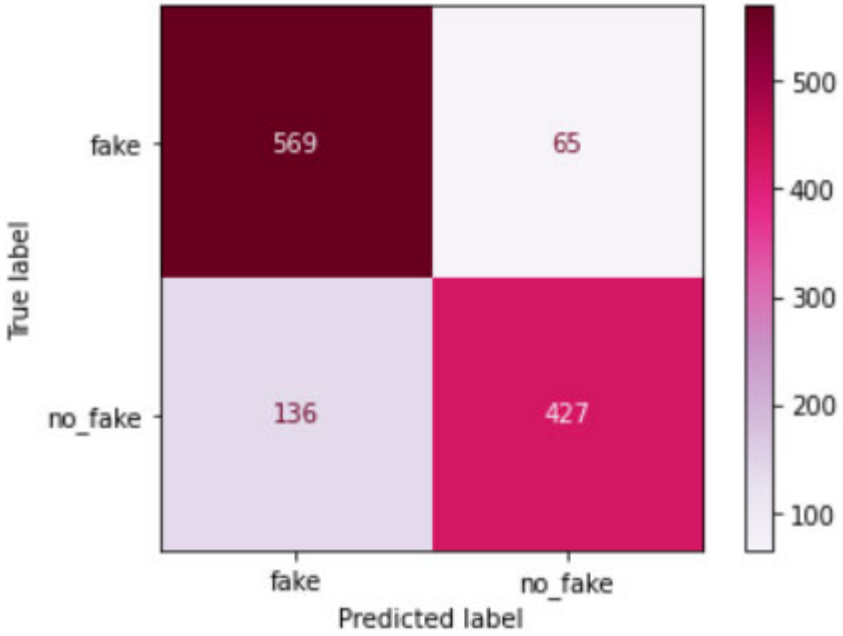
```
              precision    recall  f1-score   support

        fake       0.90      0.81      0.85       705
     no_fake       0.76      0.87      0.81       492

    accuracy                           0.83      1197
   macro avg       0.83      0.84      0.83      1197
weighted avg       0.84      0.83      0.83      1197
```

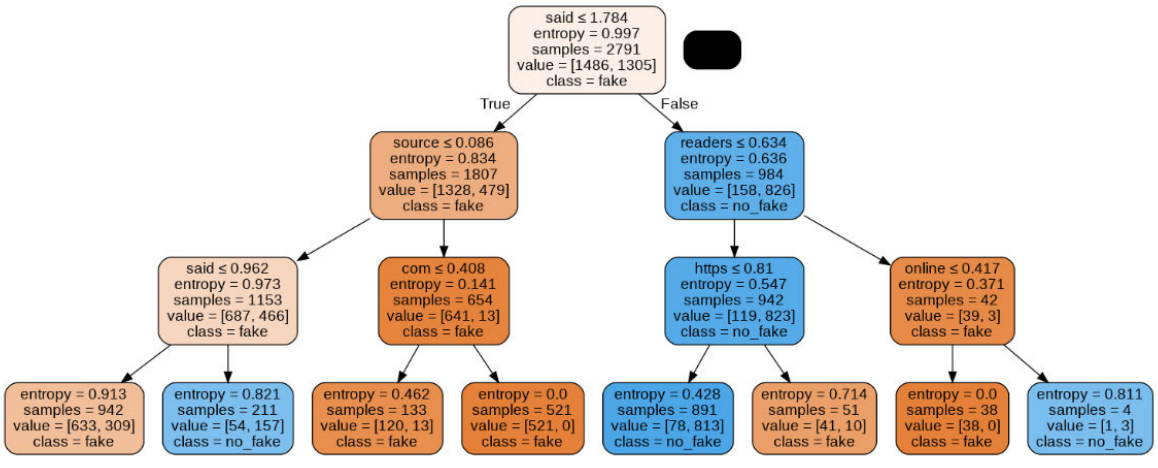
Según la matriz de confusión el modelo predijo que 705 noticias pertenecen a la clase fake y 492 pertenecen a la clase no_fake y la realidad es que 634 pertenecen a la fake y 536 pertenecen a la no_fake.

Figura 4.32: Matriz de confusión



Cada nodo interno tiene una regla de decisión que divide el árbol según el valor de algún atributo/característica del conjunto de datos. Tanto en este árbol, como en el anterior no podemos garantizar que el árbol generado sea el más óptimo.

Figura 4.33: Árbol de decisión de profundidad 3 (datos sin procesar)



Para entender el resultado obtenido de la Figura 4.33, vamos a estudiar mas en profundidad el árbol y a entender toda la información que nos proporciona:

- Aquí en este árbol el nodo raíz contiene la palabra 'said' y en caso de que el valor sea menor o igual a 1.784 la salida irá a "true" y en caso contrario a "false". También nos indica que este primer nodo divide en muestras de 1807 en 'true' y las 984 restantes en 'false'. También los valores de entropía se acerca mas a 1 cuando tiene mas muestras pertenecientes a la clase "no_fake" y cuando se acerca mas a 0, tiene mas muestras pertenecientes a la clase "fake". A lo largo de los niveles vemos divisiones por los valores de las distintas palabras, en este caso todas finalizan en el nivel 3, ninguna finaliza antes, esto sucede por que no alcanzan un nivel de entropía 0 antes de llegar al nivel tres. Concluimos que el algoritmo para saber si una noticia es falsa o no, empieza desde la raíz del y según el valor de una palabra va siguiendo el árbol hasta llegar a un nodo final que nos clasifica la noticia. En este caso, las palabras que mas influyen en la clasificación por su importancia son "said", "sorce" , "readers" , "com" y "https".

Volvemos a repetir el estudio del árbol con distintos niveles, y el que mejor Accuracy tiene es el nivel por defecto. Pero tenemos que tener en cuenta que cuantos más niveles tengamos, necesitamos un mayor procesamiento del algoritmo ya que el árbol se expande hasta el final, por lo tanto, no siempre es lo más conveniente.

Nivel	Accuracy
max_depth = 3	0.72
max_depth = 5	0.77
max_depth = 7	0.84
max_depth = 9	0.90
deafult	0.95

Cuadro 4.2: Tabla de pruebas Árbol de decisiones (datos sin procesar)

Para acabar con el árbol de decisión tenemos que destacar la importancia de los atributos, para ello podemos ver los atributos más importantes en el propio árbol, o también usando la función `feature_importances_` como en la Figura 4.34 y los resultados (Figura 4.35) que obtenemos es la importancia de cada atributo.

Figura 4.34: Código de fearute_importances

```
importancia_predictores = pd.DataFrame(
     {'predictor': countvec.get_feature_names(),
     'importancia': clf.feature_importances_}  )

importancia_predictores.sort_values('importancia', ascending=False)
```

Como podemos observar son los mismos atributos que los del árbol de la Figura 4.30:

Figura 4.35: Atributos más importantes del árbol

	predictor	importancia
7340	http	0.643058
9614	main	0.204186
14844	slideshow	0.101416
7702	invalid	0.038252
1266	barron	0.013088
...	...	...
6080	gemma	0.000000
6081	gen	0.000000
6082	gender	0.000000
6083	gene	0.000000
18221	"he	0.000000

Cross Validation:

Queremos obtener el mejor resultado posible, por lo tanto con cross validation vamos a ver que parámetros son los que más nos convienen:

Figura 4.36: Código GridSearchCV

```
from sklearn.model_selection import GridSearchCV

parameters = {'splitter' : ['best', 'random'],
'criterion' : ['gini', 'entropy'],
'max_depth': [2, 3, 5, 10],
'random_state' : [0,1,2,3] }

grid_search_dt = GridSearchCV(estimator = clf,
param_grid = parameters,
scoring = 'accuracy',
cv = 5,
verbose = 1)

grid_search_dt.fit(X_train, y_train)
```

En este caso, podemos ver que el modelo logró una precisión de clasificación estimada de alrede-

dor del 93,3%, que es inferior al resultado de una sola ejecución informado anteriormente del 97% . Esto puede sugerir que el resultado de una sola ejecución puede ser optimista y que el resultado de cinco repeticiones podría ser una mejor estimación del verdadero rendimiento medio del modelo. En la Figura 4.37 vemos con que parámetros se ha obtenido el anterior resultado. Con esto conjunto de parámetros es con el que mejor rendimiento le sacamos al algoritmo.

Figura 4.37: resultados obtenidos

```
print(grid_search_dt.score(X_test, y_test))
```

```
0.9338014042126379
```

```
print(grid_search_dt.best_estimator_)
```

```
DecisionTreeClassifier(criterion='entropy', max_depth=10, random_state=0)
```

K-Nearest-Neighbor

Construido el modelo de predicción de noticias falsas, hay que entrenar el modelo con el algoritmo KNN con nuestro juego de datos. Para ellos vamos a utilizar la librería `KNeighborsClassifier`.

Figura 4.38: KNN entrenamiento

```
from sklearn.neighbors import KNeighborsClassifier

knn = KNeighborsClassifier(n_neighbors=5)

knn.fit(X_train,y_train)

y_predict= knn.predict(X_test)
```

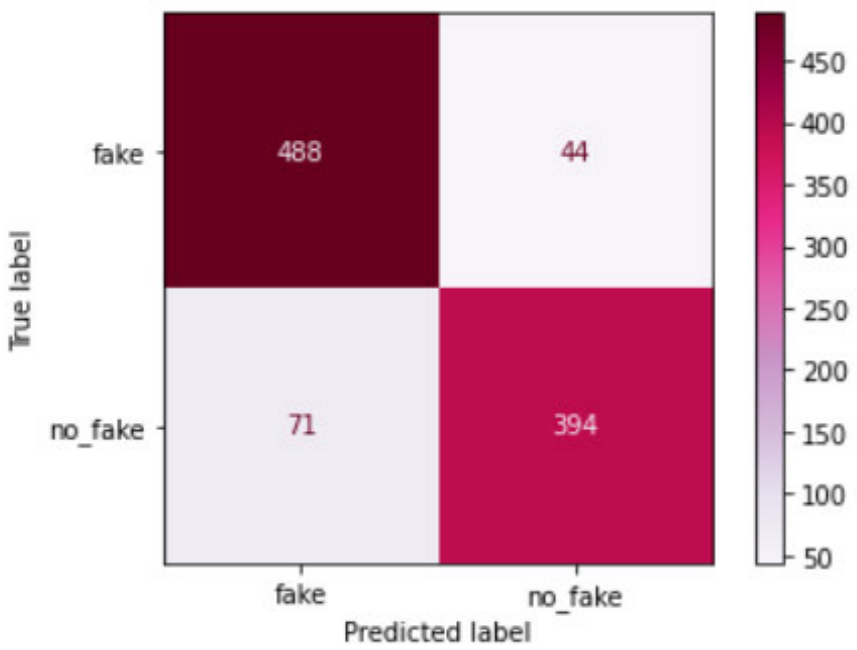
Primero hacemos el estudio con los datos procesados y obtenemos un accuracy bastante alto de 0.88, es decir, un 88% de las predicciones serán ciertas.

Figura 4.39: Estudio obtenido (datos procesados)

	precision	recall	f1-score	support
fake	0.92	0.87	0.89	559
no_fake	0.85	0.90	0.87	438
accuracy			0.88	997
macro avg	0.88	0.89	0.88	997
weighted avg	0.89	0.88	0.88	997

También obtenemos la matriz de decisión para entender mejor el resultado anterior y vemos que hay un número alto de predicciones correctas, aunque no es lo ideal ya que seguimos teniendo un número alto predicciones incorrectas.

Figura 4.40: Estudio obtenido (datos procesados)



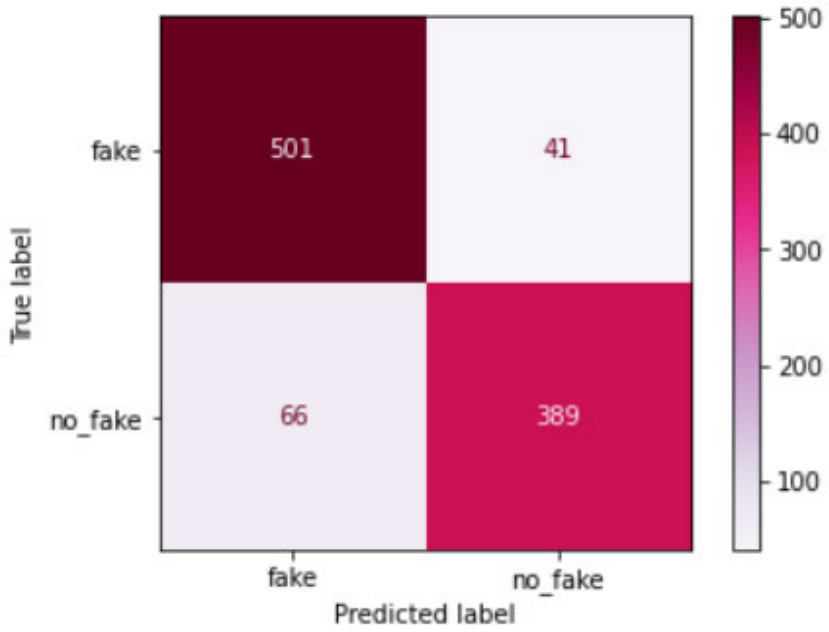
Aquí tenemos que ajustar el parámetro de n_neighbors y según van aumentando el número de vecinos, el valor de accuracy va disminuyendo . N_neighbors es el número de puntos más cercanos al nuevo que queremos que voten, los "N" más cercanos. Cuanto más puntos incluyamos, más veces hay que calcular las distancias y más votaciones habrá que hacer, por lo que el algoritmo será más lento y más costoso.

Después de hacer el análisis con los datos procesados, ahora lo haremos con los datos sin procesar. Obtenemos un mejor resultado que con los datos procesados (Figura 4.41, Figura 4.42), aún así seguimos obteniendo resultados muy similares, pero no lo suficientemente buenos.

Figura 4.41: Estudio obtenido (datos no procesados)

	precision	recall	f1-score	support
fake	0.89	0.91	0.90	533
no_fake	0.89	0.87	0.88	464
accuracy			0.89	997
macro avg	0.89	0.89	0.89	997
weighted avg	0.89	0.89	0.89	997

Figura 4.42: Estudio obtenido (datos procesados)



Los resultados anteriores obtenidos al ser tan similares, y también dependen de nuestro juego de datos y número de datos, elegiríamos el que menos coste de procesamiento tenga. La investigación ha demostrado que no existe un número óptimo de vecinos que se adapte a todos los conjuntos de datos, cada conjunto tiene sus propios requisitos. Un valor muy bajo como n = 1 ó n = 2, puede provocar efectos atípicos en el modelo y los valores demasiado grandes, aunque son buenos pueden encontrar algunas dificultades.[20][21]

Cross Validation:

De nuevo volvemos a aplicar la técnica de validación cruzada para poder tener una mejor garantía de que nuestro proyecto tenga una mayor certeza.

Figura 4.43: Funcionamiento Cross Validation

```
from sklearn.model_selection import GridSearchCV

parameters = {'n_neighbors' : [2,3,5,6,7,9,10,15,20,30]}

grid_search_dt = GridSearchCV(estimator = knn,
param_grid = parameters,
scoring = 'accuracy',
cv = 5,
verbose = 1)

grid_search_dt.fit(X_train, y_train)
```

El promedio que obtenemos es de un 89,76%. El parámetro n_neighbors que más se ajusta a nuestro caso es 3. El valor obtenido es uno de los más altos obtenidos.

Figura 4.44: Funcionamiento Cross Validation

```
] print(grid_search_dt.score(X_test, y_test))

0.8976930792377131

print(grid_search_dt.best_estimator_)

KNeighborsClassifier(n_neighbors=3)
```

5. Evaluación

5.1 Resultados obtenidos

En este apartado vamos a comparar los resultados obtenidos de los distintos algoritmos que hemos aplicado. Mostramos los resultados obtenidos con grid search tanto con datos procesados, como con datos sin procesar.

ALGORITMO	ACCURACY (DATOS CON)	ACCURACY (DATOS SIN)
Regresión Logística	0.9672	0.9709
Naive Bayes	0,9438	0.9689
Árbol de Decisiones	0.9338	0.9423
KNN	0.8976	0.8896

Cuadro 5.1: Tabla de resultados

En primer lugar observamos que la diferencia entre proporcionar una entrada de datos procesados o no procesados es muy pequeña. Esto se puede deber a que el procesamiento de datos realizado no es lo suficientemente adecuado, por lo tanto habría que realizar un estudio mas profundo sobre el procesamiento de los datos. Esto nos llevaría a tener un mejor resultado con los datos procesados.

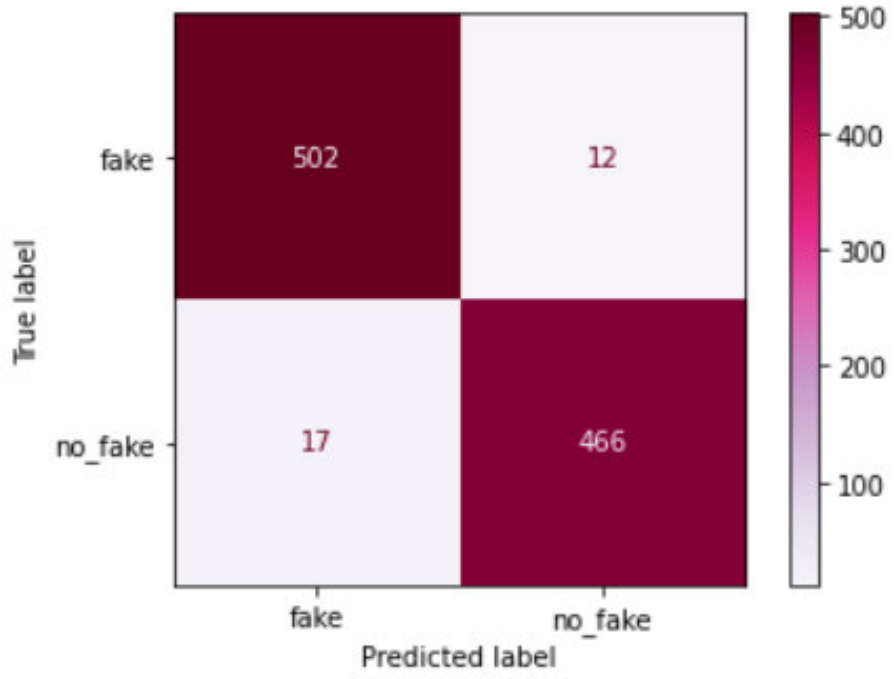
Analizando todos los resultados obtenidos en la Figura 5.1 se llega a la conclusión de que el algoritmo que mejor resultado nos ha dado ha sido Regresión Logística con los datos procesados, que ha sido de 0,9709. Además según lo que hemos observado durante el estudio de este proyecto, es que, es un método bastante adecuado para la predicción binaria, ya que el resultado o variable objetivo es de naturaleza dicotómica, es decir, que solo hay dos clases posibles. Proporciona una salida discreta.

5.2 Análisis de la mejor solución

Después de un largo estudio de distintos algoritmos a los que proporcionamos una misma entrada de datos, el algoritmo que mejor se ajusta a nuestra necesidad y mejores resultados hemos obtenido es Regresión Logística con datos procesados. Obtenemos unas predicciones bastante precisas. Es un método muy adecuado para la clasificación binaria, es un método simple que clasifica clases y es fácil de interpretar. Estima la relación entre una variable dependiente y las variables independientes. Con cuantos más datos vayamos entrenando nuestro algoritmo, mejores resultados obtendremos. La salida que nos proporciona un modelo logístico es una probabilidad, para conseguir la clasificación, establecemos un límite (threshold) a partir del cual se considera que la variable pertenece a uno de los niveles, al grupo 1 si la probabilidad estimada es mayor de 0.5 y al grupo 0 de lo contrario.

El modelo de Regresión Logística nos ha dado los siguientes resultados, empezamos por el informe de clasificación que nos ayuda a medir la calidad de las predicciones de nuestro algoritmo. Todos estos datos se calculan con los verdaderos positivos, falsos positivos, verdaderos negativos y falsos negativos. Nuestro modelo solo ha realizado 29 predicciones incorrectas de las 997 predicciones que ha realizado, es decir, ha realizado un 2,9% de predicciones incorrectas.

Figura 5.1: Matriz de Confusión



Como el propio reporte nos mostraba hay un 97% de Accuracy, que nos indica que cada 100 predicciones unas 3 son incorrectas, que coincide con los resultados obtenidos en la matriz de confusión. Este modelo tiene una ”Precision” de 97% que es la capacidad del clasificador de no etiquetar una instancia positiva siendo negativa, mide la calidad del modelo. Es decir, que un 97% de las prediccones serán correctas, y que el modelo se equivocará un 3%. El numero de positivos que se identifican de manera correcta, es decir, el porcentaje de noticias que se clasificaron correctamente, que nos lo muestra la variable Recall, que en este caso es 95%. Esto quiere decir, que el modelo es capaz de identificar un 95% de ”fake_news”.

Con F1Score 96%, comparamos el rendimiento del modelo con ’precision’ y ’recall’, esto nos lleva a que nos importa de igual forma la ’precision’ y ’recall’ en este caso.

Figura 5.2: Resumen Regresión Logística

```
Accuracy: 0.964895
Precision: 0.969697
Recall: 0.949772
F1 score: 0.959631
```

Otro valor importante que tenemos que mencionar es el coeficiente, y en este caso tratamos de encontrar los coeficientes más óptimos para todos los atributos. Esto nos esta indicando que el efecto que tiene cada palabra en nuestra predicción, es decir, la importancia de dicha característica a la hora de decidir a que clase pertenece cada noticia. En caso de que el coeficiente sea un valor positivo, nos indica que corresponde a la clase 1 o en este caso particular ”fake” y si el valor es negativo a la clase 0, nos indica que corresponder a la clase ”no_fake”. Cuanto mayor sea el valor, más afectará a dicha clase.

Aquí podemos observar los coeficientes más bajos, o las palabras que más influyen en la clase ”fake”. En primer lugar tendríamos la palabra *http* que realmente no tiene mucho sentido en que influyan en que una noticia sea falsa, pero tampoco en que la noticia sea verdadera, lo mismo pasa con *even, add, red*. Pero palabras como *diy, potato o pot*, por ejemplo, la palabra *patata* nos podría llevar más a pensar que la noticia podría ser falsa. También hay palabras como *news, campaign, fact* , como por ejemplo, *fact* que nos llevaría a pensar que una noticia pertenece a la clase ”no_fake”.

Figura 5.3: Coeficientes menores

	Coeficiente
http	-2.493665
news	-1.702015
danger	-1.184093
red	-1.170936
campaign	-1.161036
fact	-1.080379
potato	-1.076449
diy	-1.072210
publish	-1.066218
video	-1.048754
fentanyl	-1.027612
even	-1.005210
pot	-1.004760
add	-0.963419
img	-0.932416

Y aquí tenemos los coeficientes más altos, o las palabras que mas han influido en la clase "no_fake". Tenemos palabras como *copyright, main, cnn, bbc, press, photo* que coherentemente nos llevaría a clasificar a una noticia como verdadera. Pero también hay palabras como *Tom, Wednesday* que no nos dan ningún tipo de información de sí una noticia podría ser falsa o no.

Figura 5.4: Coeficientes mayores

	Coeficiente
tom	0.530820
wednesday	0.535638
film	0.537196
korean	0.548133
exhibit	0.548671
copyright	0.636451
caption	0.719554
photo	0.733673
main	0.833783
said	0.852374
press	0.988085
find	1.033616
bbc	1.058536
chat	1.140902
cnn	1.310058

Todo esto nos llevaría a la conclusión de que posiblemente haría falta un mayor pre procesamiento de los datos y que palabras como *http o Tom* deberían de ser descartadas ya que no nos aportan nada, lo único que hacen es aturdir la clasificación de las noticias.

6. Impacto Social

El tema principal que estamos tratando es la desinformación a la que se exponen los usuarios consumidores de noticias, las consecuencias que sobre todo han sido provocadas por el surgimiento de las nuevas herramientas digitales. Llegamos a un punto en el que al ser divulgadas de manera tan rápida y fácil, dificulta los filtros y controles de veracidad, afectando directamente a la credibilidad de los usuarios de manera constante. Sí con el surgimiento de las nuevas tecnologías hubo un antes y un después en las ”fake news”, también lo ha habido durante la pandemia del COVID-19 teniendo unas consecuencias bastante graves afectando a la salud pública global. Hoy en día los usuarios son bastante exigentes y tratan de satisfacer su necesidad de estar permanentemente informados, pero también tratan de asegurarse de que esto les lleva a tomar decisiones correctas. Aquí es donde con este proyecto que trata de predecir si una noticia es falsa o no y así evitar la información de baja calidad. Con esto se trata de proteger la salud pública, la difusión de rumores y teorías conspiratorias, manipulación de mercados financieros y difamación. Se evita que las noticias falsas tengan mas impacto que las noticias reales, sobretodo en las Redes Sociales, que hablamos de noticias rápidas. Evitar llegar a esa crisis de confianza de la información evitamos la divulgación de parodias, contenido engañoso, contexto falso, contenido manipulado y contenido falso fabricado.

7. Conclusiones

Llegados a este punto del proyecto, miramos hacía atrás y vemos que hemos aprendido sobre Machine Learning, sus algoritmos, como funciona cada uno de ellos, como se ha implementado todo y como se adapta cada algoritmo a nuestro juego de datos. Además hemos comparado los distintos resultados obtenidos y nos hemos centrado en el algoritmo que más nos conviene. Por lo tanto, podemos decir que hemos abarcado los objetivos principales de este proyecto. Las nuevas tecnologías no se quedan atrás y nos han mostrado lo rápido que funcionan y lo eficientes que son.

También hemos tratado de explicar y entender la importancia de las noticias falsas y como ha impactado en la sociedad y las consecuencias tan graves que pueden tener.

Finalmente concluimos que este proyecto trata de mejorar la sociedad en la que vivimos, ayudando a crear fuentes de información más fiables y seguras, y a ser unos productores y consumidores de noticias verdaderas.

8. Futuras mejoras

- **Mejorar el pre procesamiento de los datos** : eliminación del ruido y de la inconsistencias de algunos datos para tener una mejor precisión a la hora de clasificar, ya que durante el estudio de este proyecto se ha llegado a la conclusión de que es necesario.
- **Estudiar otros modelos de Machine Learning** : aplicación de algoritmos como Random Forest y Support Vector Machine entre otros y se compararán con los demás que se han estudiado en este proyecto.
- **Estudio con Ensemble Learning** : que se trata de usar múltiples modelos de aprendizaje automáticos para mejorar la confiabilidad y precisión de los mensajes, es decir, entonaremos múltiples modelos y combinaremos sus resultados para crear un modelo óptimo.
- **Estudiar modelos de Deep Learning** : se estudiaría la clasificación de texto con Redes Neuronales entre otros algoritmos y se compararían entre ellos y con los algoritmos de Machine Learning para ver con cual obtendremos el mejor resultado.
- **Entrenar los algoritmos con un número mayor de datos** : consistiría en en volver a probar los mismos algoritmos y otros nuevos con un dataset mucho más grande y así los algoritmos tengan un mayor entrenamiento y una predicción mucho mas precisa.
- **Poner en práctica el modelo en la realidad** : siendo este un proyecto que aporta a la sociedad, podría ser desarrollado para que los usuarios lo usen, ya sea en modo aplicación o página web, de esta manera, el propio usuario podría comprobar si la noticia que esta leyendo sería verdadera o falsa.

Bibliografía

[1] (). «Noticia Ámazon», dirección: https://www.lavanguardia.com/tecnologia/20170208/414158728901/supermercado-amazon-robots-automatizado-fake-news-noticias-falsas-periodismo-tecnologico-rumores.html.

[2] (). «Noticia Ámazon», dirección: https://www.ucm.es/otri/noticias-las-fake-news-siempre-han-existido-pero-hoy-en-dia-se-han-visto-catapultadas-por-las-redes-sociales.

[3] (). «Noticia Ámazon», dirección: https://www.muyinteresante.es/tecnologia/articulo/la-ciencia-confirma-que-las-fake-news-se-extienden-mas-rapido-que-la-verdad-581520594406.

[4] Kaggle. (). «Fake News detection», dirección: https://www.kaggle.com/jruvika/fake-news-detection.

[5] (). «Diferencia entre algoritmos de clasificación y regresión», dirección: https://aprendeia.com/diferencia-entre-algoritmos-de-clasificacion-y-regresion/.

[6] (). «7 pasos del Machine Learning para construir tu máquina», dirección: https://www.aprendemachinelearning.com/7-pasos-machine-learning-construir-maquina/.

[7] (). «Regresión Logística», dirección: https://www.iartificial.net/regresion-logistica-para-clasificacion/.

[8] (). «Bernoulli Naive Bayes», dirección: https://iq.opengenus.org/bernoulli-naive-bayes/.

[9] (). «Qué son los árboles de decisión y cómo se usan en Inteligencia Artificial», dirección: https://agenciab12.com/noticia/que-son-arboles-de-decision-inteligencia-artificial.

[10] MERKLE. (). «El algoritmo K-NN y su importancia en el modelado de datos», dirección: https://www.merkleinc.com/es/es/blog/algoritmo-knn-modelado-datos.

[11] (). «K-Nearest Neighbor(KNN) Algorithm for Machine Learning», dirección: https://www.javatpoint.com/k-nearest-neighbor-algorithm-for-machine-learning.

[12] (). «¿Qué es Colaboratory?», dirección: https://colab.research.google.com/?hl=es.

[13] (). «scikit-learn Machine Learning in Python», dirección: https://scikit-learn.org/stable/.

[14] (). «NLP», dirección: https://www.aprendemachinelearning.com/ejercicio-nlp-cuentos-de-hernan-casciari-python-espanol/.

[15] (). «sklearn.feature$_e xtraction.text.CountVectorizer$», dirección: https://scikit-learn.org/stable/modules/generated/sklearn.feature_extraction.text.CountVectorizer.html.

[16] (). «Splitting Datasets With the Sklearn train$_t est_s plitFunctionr$», dirección: https://www.bitdegree.org/learn/train-test-split#what-is-train-test-split.

[17] (). «Comprender la regresión logística en Python», dirección: https://www.datacamp.com/community/tutorials/understanding-logistic-regression-python.

[18] (). «Clasificación», dirección: https://developers.google.com/machine-learning/crash-course/classification/precision-and-recall.

[19] (). «Cross Validation», dirección: https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.RepeatedKFold.html.

[20] (). «Algoritmo K-vecino más cercano (KNN) para aprendizaje automático», dirección: https://www.javatpoint.com/k-nearest-neighbor-algorithm-for-machine-learning.

[21] (). «Clasificación con k-nearest neighbors», dirección: https://www.ellaberintodefalken.com/2019/02/clasificacion-con-k-nearest-neighbors.html.

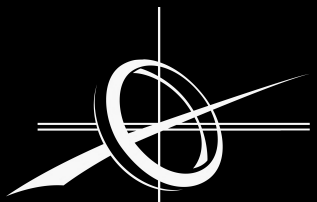
Glosario

BOW Bag of Words. ii, 26, 27, 36

CSV Las siglas CSV vienen del inglés ”Comma Separated Values” y significan valores separados por comas. 21

NLP Natural Language Processing o Procesamiento de Lenguaje Natural. 7, 22

NLTK Librería Python para Natural Language Toolkit. 22, 23


Universidad
Politécnica
de Madrid
ETSI SISTEMAS
INFORMÁTICOS