MÁSTER BIG DATA Y BUSINESS ANALYTICS
TRABAJO FIN DE MÁSTER

FAKE NEWS, CÓMO DETECTAR NOTICIAS FALSAS CON MODELOS DE MACHINE LEARNING

Alumna: Esther Fernández Barchín

Tutores: Carlos Ortega y Santiago Mota

Fecha de entrega: 2 de marzo

Índice

1. Introducción

Vivimos en un contexto tecnológico sin precedentes, conocido como la era de la información. Cada vez es mucho más sencillo conseguir un gran volumen de información de forma inmediata ya sea por cualquier medio de comunicación o redes sociales. Ante cualquier novedad o suceso, la sociedad está a la distancia de un clic para resolver sus dudas. Por ejemplo, durante el COVID en 2019 ante la situación de incertidumbre, era habitual visitar las redes sociales y ver las noticias para obtener información al respecto.

El problema de esto es que no se puede verificar si la información que estamos leyendo es de verdad o no, ya que se corre el riesgo de estar frente a un bulo o fake news. Además, con el creciente uso de la tecnología es mucho más fácil adulterar imágenes, videos, … Y las fake news pueden inundar masivamente las redes ocasionando desinformación y confusión a las personas.

Ante el crecimiento exponencial de las fake news y su rápida expansión exigen la automatización de procesos que validen las noticias. Todo el volumen implica una infraestructura de Big Data para procesar y analizar las noticias y de Machine Learning para diferenciar una fake new de una noticia normal.

Las noticias suelen tener formato de texto, el cual forma parte de los datos llamados “no estructurados”, el cual no puede ser tratado correctamente por softwares o herramientas de análisis de datos tradicionales. Para tratarlo hay que recurrir al text mining o análisis de textos, el cual consiste en transformar un texto no estructurado en datos estructurados para analizarlos posteriormente. Esta práctica se basa en la tecnología de Natural Language Processing (procesamiento natural del lenguaje) que permite que la tecnología comprenda y trate el lenguaje humano de manera automática, o que permite clasificar los textos ya sea por sentimiento, tema o intención.

1.1. Propósito

El propósito de este trabajo es estudiar la clasificación entre una noticia verdadera y una fake news de un dataset. Se desarrollará y argumentará un estudio completo sobre cómo se diferencia cada una de las categorías utilizando el lenguaje de programación de Python

1.2. Ámbito

El problema de las fake news es una cuestión interdisciplinar que comparte actualmente lingüística, Recuperación de Información o *Information Retrieval* e Inteligencia Artificial. Siendo estos dos últimos desde los que se va a abordar el trabajo. Para ello nos centraremos en el Aprendizaje Automático de la Inteligencia Artificial. Ahora definiremos los conceptos de forma básica:

- El procesamiento del lenguaje natural (NLP) es un rango teóricamente motivado de técnicas computacionales para analizar y representar textos que ocurren naturalmente en uno o más niveles de análisis lingüístico con el fin de lograr el procesamiento del lenguaje humano en una variedad de tareas o aplicaciones.
- La Recuperación de Información es la conversión de grandes volúmenes de texto en estructuras simplificadas y comprensibles para su uso posterior.
- El Aprendizaje Automático es el estudio de algoritmos de computación que mejoran automáticamente por medio de la experiencia.

1.3. Objetivo

Para clasificar correctamente las fake news, haremos los siguientes pasos:

- Realizar un **preprocesamiento** correcto de los **datos** para que puedan ser utilizados como entrada en un modelo de aprendizaje automático.
- Proponer diversos **modelos** entrenados de clasificación que presenten buenos resultados.
- Validar los modelos y compararlos entre sí.
- Definir un criterio justificado para elegir el mejor modelo.

2. Datos

2.1. Fuente

El conjunto de datos proviene de la página https://www.uvic.ca/ecs/ece/isot/datasets/fake-news/index.php. Está dividido en dos conjuntos de datos: real y fake.El dataset de real tiene 21.417 documentos y el dataset de fake tiene 23.481documentos.

Los datos de real se recopilaron a partir de fuentes verídicas a partir de noticias seleccionadas de la página Reuter. En cambio, las fake news procedían de sitios web poco fiables marcados por Politifact (una organización de comprobación de hechos de EE.UU.) y Wikipedia. El conjunto de datos contiene distintos tipos de artículos sobre diferentes temas. Sin embargo, la mayoría de los artículos se centran en noticias políticas y mundiales.

2.2. Obtención del corpus

El primer paso de todos es importarnos las librerías y los dos dataset para formar el corpus: real y fake(se puede ver en el anexo adjunto). El primer paso es identificar si es fake-news o si es real, por lo que en cada set añadimos una columna poniendo 0 si es real y 1 si es fake-news.

El segundo paso en el dataframe de real, la columna contiene información de la publicación de la noticia, por lo que con una función eliminaremos esta información sin afectar al texto.

	title	text	subject	date	label
0	As U.S. budget fight looms, Republicans flip t...	WASHINGTON (Reuters) - The head of a conservat...	politicsNews	December 31, 2017	0
1	U.S. military to accept transgender recruits o...	WASHINGTON (Reuters) - Transgender people will...	politicsNews	December 29, 2017	0

Después unimos los dos dataframe en uno solo y juntamos las columnas title y text. Dejamos solamente las columnas label y title_text. Esto sería el corpus.

	label	title_text
0	0	As U.S. budget fight looms, Republicans flip t...
1	0	U.S. military to accept transgender recruits o...
2	0	Senior U.S. Republican senator: 'Let Mr. Muell...
3	0	FBI Russia probe helped by Australian diplomat...
4	0	Trump wants Postal Service to charge 'much mor...
...	...	...
23476	1	McPain: John McCain Furious That Iran Treated ...
23477	1	JUSTICE? Yahoo Settles E-mail Privacy Class-ac...
23478	1	Sunnistan: US and Allied 'Safe Zone' Plan to T...
23479	1	How to Blow $700 Million: Al Jazeera America F...
23480	1	10 U.S. Navy Sailors Held by Iranian Military ...

44898 rows × 2 columns

2.3. Estudio del dataframe

La técnica del análisis exploratorio tiene varios pasos, documentado uno a uno.

Número de documentos y columnas

Tenemos un total de 44898 documentos y 2 columnas después del tratamiento del apartado anterior.

Número de documentos duplicados

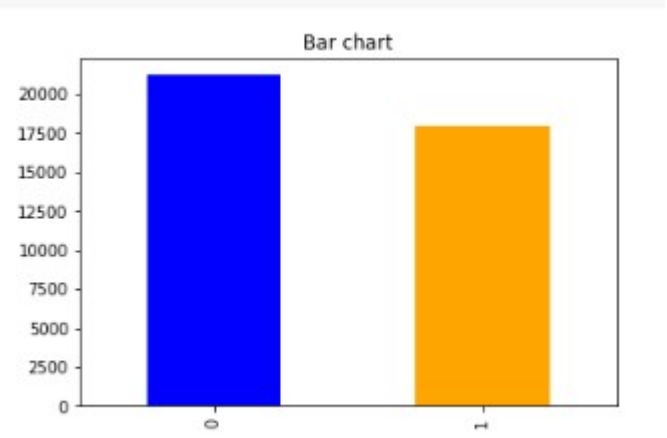
Comprobamos la existencia de duplicados en las filas, en caso afirmativo, los eliminamos. Además, se comprueba que no queden nulls en las columnas.

Número de documentos por cada clase

Contamos el número de documentos de cada clase. De esta manera, podemos comprobar si los datos están balanceados.

```
[ ] df['label'].value_counts()

    0    21196
    1    17908
    Name: label, dtype: int64
```

En este caso, hay mayor número de documentos categorizados como 0, es decir, como noticias reales, que como 1, como fake news.



Podemos observar en el gráfico que el dataset está relativamente desbalanceado pero las fake news tienen menos documentos que las noticias reales.

Distribución de la longitud de los tweets en caracteres

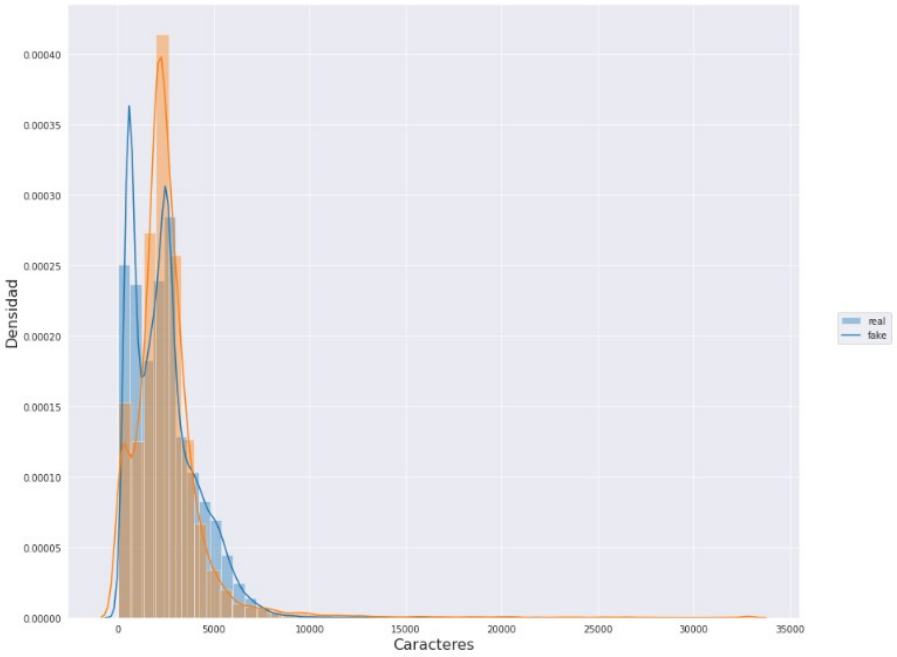
En este subapartado se ha realizado un cálculo típico, que es la longitud de cada uno de los textos, creando una nueva columna para almacenar la longitud en caracteres y en tokens de los documentos del corpus.

```
[ ] df['length'] = df['title_text'].apply(lambda x: len(x))
```

```
df.head(2)
```

	label	title_text	length
0	0	As U.S. budget fight looms, Republicans flip t...	4702
1	0	U.S. military to accept transgender recruits o...	4120

Gracias a este gráfico podemos ver que la distribución de las dos categorías es bastante similar, pero las noticias reales se concentran entre 0-5.000 caracteres, y la distribución de noticias falsas parecen tener outliers.



Cálculo del número de palabras, media de caracteres y palabras distintas

Podemos ver que existe una gran concentración de documentos de corta extensión entre 1 y 5659 palabras. Una noticia de dos palabras no es que sea muy creíble., tal y como apunta el mínimo. Por eso mismo quitaremos todas aquellas palabras que caigan en el percentil 10% (es decir un total de 92 palabras por documento). De esta forma también nos quitamos columnas que tienen un enlace como texto.

	Número de palabras por documento	Media de caracteres por palabra	Número de palabras distintas
Conteo	39.104	39.104	39.104
Media	418	5,97	260
Desviación	317	1,24	137
Mínimo	2	1,75	53
Percentil 25%	224	5,73	181
Percentil 50%	386	5,94	242
Percentil 75%	535	5,94	314
Máximo	5659	149,5	2315

Con estos cambios el dataframe se quedaría con 32.144 documentos y 6 columnas.

3.Preprocesamiento y normalización de los datos

3.1. Preprocesamiento

Al acabar de limpiar los datos, eliminaremos las columnas de conteo de palabras, características y documentos, dejando únicamente las 3 columnas: *label*,*title_text* y *length*.

	label	title_text	length	nwords	avg_char_word	n_distint_word
0	0	As U.S. budget fight looms, Republicans flip t...	4702	761	6.178712	424
1	0	U.S. military to accept transgender recruits o...	4120	632	6.518987	338

	label	title_text	length
0	0	As U.S. budget fight looms, Republicans flip t...	4702
1	0	U.S. military to accept transgender recruits o...	4120

El preprocesamiento en Aprendizaje Automático suele estar enfocado a tareas de escalado, normalización, imputación entre otras muchas transformaciones. En este caso hemos aplicado un número más limitado de operaciones.

Eliminación de espacios extra

Este paso es el primero paso es necesario para eliminar los espacios que no son necesarios entre palabras, de esta forma evitamos en la tokenizacion los espacios vacíos.

Cambiar a minúsculas

Para que los procesos siguientes identifiquen correctamente las palabras, hay que pasarlas a minúsculas.

Eliminación de emojis, html, url y paréntesis

Esta práctica es interesante, ya que los emojis pueden medir el análisis de sentimiento de lo que las personas escriben en los tweets. En este caso solo nos hemos dedicado a eliminarlos para profundizar más en el significado de las palabras. Además, eliminamos también html, urls y paréntesis para que dichos caracteres no afecten al análisis posterior.

Tokenizador

El texto bruto está compuesto por una secuencia de caracteres, es habitual que antes del análisis de los textos, estos sean divididos en fragmentos más pequeños conocidos como tokens. En el caso de datos de redes sociales, un token puede ser tanto una palabra, un símbolo de puntuación, un número o un emoticono.

Stopwords

Muchas de las librerías de procesamiento del lenguaje natural ofrecen conjuntos de palabras por idioma llamados *StopWords*. Estos grupos recogen las palabras más frecuentes que sirven como

nexos entre ideas. Por lo general, se sigue el criterio de que se quitaran de una frase que no se debería de ver mermada la información que se refleja. Dependiendo del conjunto de datos y problema, la eliminación de las *StopWords* puede mejorar significativamente los resultados de la predicción. En el caso de este datase, si he visto conveniente eliminar las *StopWords* ya que para determinar si una noticia es fake, no veo necesario el uso de las *StopWords*.

Modelos N-gram

Los n-gramas son secuencias de n tokens consecutivos provenientes de un texto. La combinación de n-gramas puede proporcionar información sobre la temática de un texto. La combinación de n-gramas puede proporcionar información sobre la temática de un texto. Generalmente se generan unigramas, que son iguales que los tokens del texto. Los bigramas, que son combinaciones pareadas de tokens y los trigramas que son triadas de tokens.

Stemming

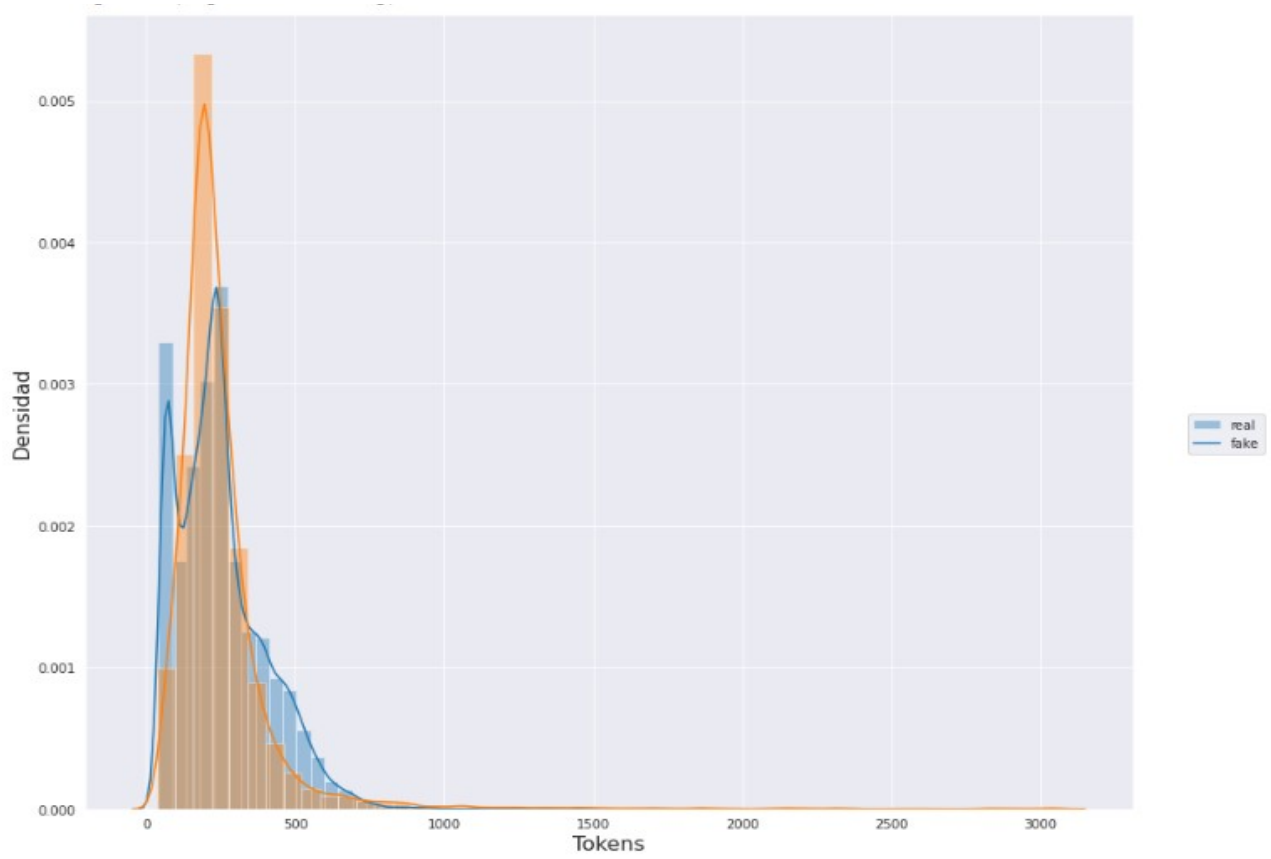
El proceso de Stemming es un método para reducir una palabra a su raíz o stem. Por ejemplo, de lcampo, campesino cogería la raíz de camp. Es el proceso que hemos utilizado para realizar todo el proceso previo al análisis.

Lemmatizacion

EL proceso de lemmatization es un proceso lingüístico de hallar el lema. El lema sería la representante de una palabra. Por ejemplo, luminoso,lucidez a luz.

3.2. Resultados del preprocesamiento

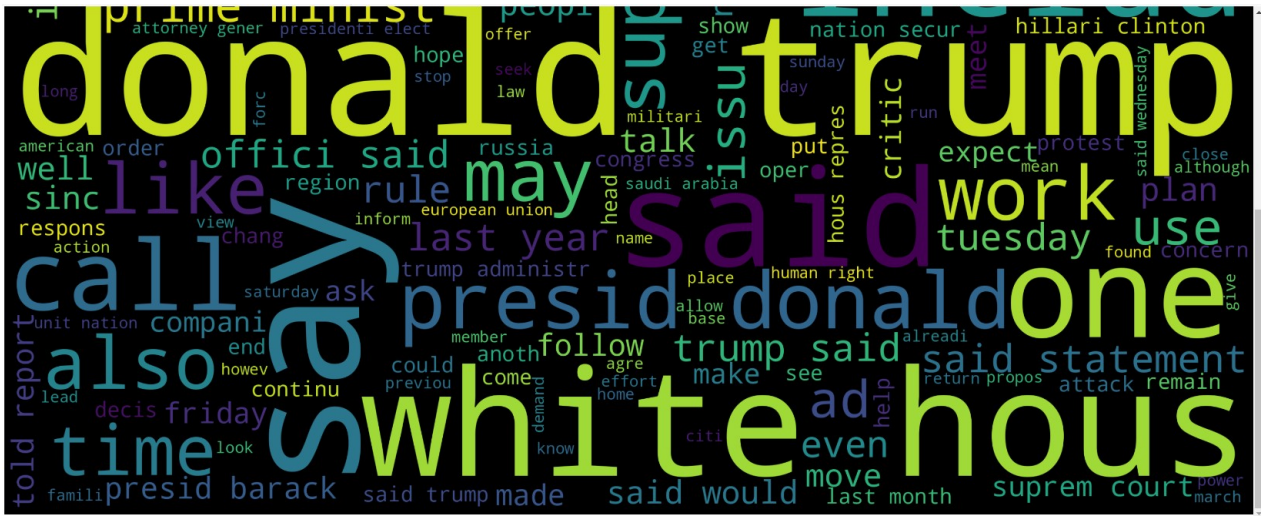
Después de la realización de todas las fases del preprocesado, repetiremos el proceso de limpieza de datos para ver la densidad y el número de tokens presentes en cada una de las clases.



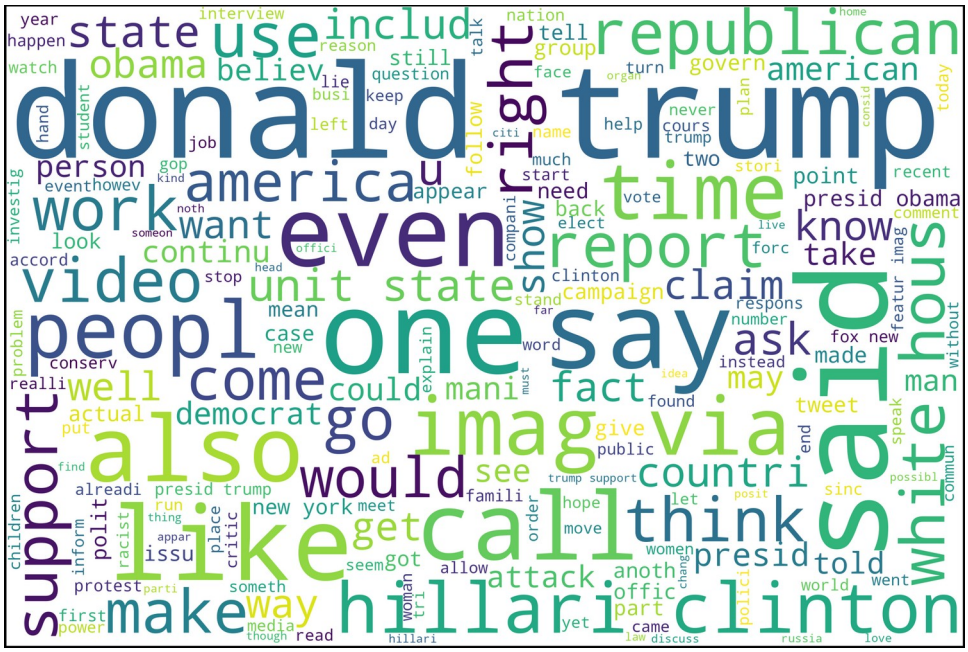
En el gráfico podemos comprobar que las noticias reales agrupan una cantidad de tokens menor a las de las noticias falsas. Las noticias falsas además tienen mayor densidad en los tokens que las noticias verdades. Esto quiere decir que las palabras que hay más tokens en los documentos que están clasificados como fake news.

Además, podemos ver qué palabras son más frecuentes para cada tipo de noticia y de forma muy visual.

Para las noticias falsas tenemos las siguientes palabras:



En cambio, para las noticias reales tenemos las siguientes:



3.3. Vectorización

En este punto del estudio del Text Mining, si queremos entrenar modelos de inteligencia artificial es necesario transformar las cadenas de caracteres de los documentos a un formato numérico que puedan ser procesados por modelos estadísticos o por algoritmos de aprendizaje automático.

La idea se basa en representar los documentos como vectores en un espacio vectorial n-dimensional. Existen diferentes maneras de transformar los documentos, pero en este caso solamente explicaré el TF-IDF.

Esta estrategia se basa en construir una matriz de términos y documentos, que representan el contenido de los textos en vectores de manera que puedan ser utilizados como fuente de datos de entrada para tareas de clasificación. Además, se aplica un coeficiente de penalización a palabras comunes en muchos documentos.

Este es el TF-IDF que he construido:

```
[ ] vectorizador = TfidfVectorizer(max_features=30,ngram_range= (1,3) ,min_df = 0.01,lowercase = False)

    vector_data = vectorizador.fit_transform(df['clean_text'])
```

A continuación, la explicación de los parámetros utilizados:

- *max_features*: Este parámetro especifica el número máximo de características o palabras que se extraen de los datos de entrada. En este caso, se han especificado 30 características para extraer del conjunto de datos.
- *ngram_range*: Este parámetro especifica el rango de longitud de las secuencias de palabras que se utilizarán como características. En este caso, se utilizarán secuencias de palabras de

longitud 1, 2 y 3 (también conocidas como unigramas, bigramas y trigramas) como características.

- min_df: Este parámetro especifica la frecuencia mínima de documento (df) que se requiere para que una palabra se incluya como característica. En este caso, cualquier palabra que aparezca en menos del 1% de los documentos no se incluirá como característica.
- lowercase=False: Este parámetro especifica si se deben convertir todas las palabras a minúsculas antes de procesarlas. En este caso, se ha establecido en False, lo que significa que las palabras no se convertirán a minúsculas y se mantendrán en su forma original, ya que hemos hecho el proceso anterior el paso de convertir a minúsculas.

4. Entrenamiento y evaluación de modelos

El primer paso es definir las variables predictoras que son las características extraídas anteriormente a través de TF-IDF. de todos es separar los datos de entrenamiento con los datos de test , lo que nos quedaría de la siguiente forma:

```
(8036, 30)
```

Para decidir qué modelo de machine learning se ha generado una función para medir la calidad de varios modelos. En un primer momento medimos solamente el accuracy.

LogisticRegression	LinearSVC	MultinomialNB	MLPClassifier	GausianNB
83,6	83,6	76,66	86,32	76,63

De entre estos modelos he seleccionado los que tienen un accuracy más alto.

4.1. Modelo Regresión Logística

El modelo de Regresión Logística es un algoritmo de clasificación que se utiliza para para predecir la probabilidad de una variable dependiente categórica. Esta clasificación es de carácter dicotómica, la variable dependiente tiene los datos codificados como 0 y 1, en este caso 0 es noticia verdadera y 1 noticia falsa.

La regresión logística se utiliza en la clasificación de texto, como la clasificación de spam de correo electrónico, la identificación de sentimientos y la clasificación de temas en los documentos.

Aunque en el análisis anterior se ha sacado bastante puntuación de accuracy es posible ajustar el modelo a través de GridSearch.Para maximizar los resultados, he hecho un Grid Search en el Pipeline para sacar directamente el modelo ganador. Para ello he utilizado los siguientes parámetros para Regresión Logística:

- *random_state*: Especifica la semilla aleatoria para garantizar la reproducibilidad del modelo. En este caso esta especificada al principio del código.
- *class_weight:* Especifica el peso de clase para el modelo. Para que el modelo trate de manera justa ambas clases y que permita al modelo una buena clasificación, en la que la clase 0 tiene un peso de 1 y la clase 1 tiene un peso de 5669/4831.

Para el TDF-IDF he utilizado el parámetro de max_features = 1000 para que el modelo pudiera ser más eficiente a la hora de ejecutarse ya que he estado teniendo varios problemas computacionales con otros valores en max_features.

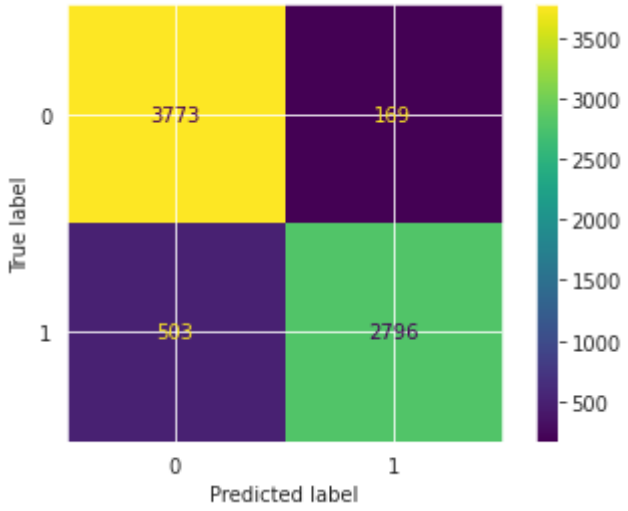
```
model1 = GridSearchCV(
    estimator=Pipeline(
        steps = [
            ("TfidfVectorizer", TfidfVectorizer(
                lowercase = True,
                max_features = 1000
            )),
            ("LogisticRegression", LogisticRegression(
                random_state = random_state,
                class_weight = {"0":1, "1":5669/4831}
            ))
        ]
    ),
    param_grid = {
        "TfidfVectorizer__ngram_range": [(1,2), (1,3)],
        "LogisticRegression__C": [1e-3, 1e-4, 1e-5],
    },
    cv = 5,
    n_jobs = -1,
    verbose = 3
)
```

Una vez realizado el tuneado, los mejores hiperparámetros son los siguientes:

Modelo	Hiperparámetros	Número
LogisticRegression	C	0.001
TfidfVectorizer	ngram_range	(1,3)

Resultados

Matriz de confusión



Accuracy

Train	Test
90,71	**90,33**

Reporte clasificación para el conjunto test

	precision	recall	F1-score	support
accuracy			0,91	7.241
macro avg	0,91	0,90	0,91	7.241
weigthed avg	0,91	0,91	0,91	7.241

4.2. Modelo Linear SVC

Es un modelo de clasificación lineal que utiliza el algoritmo de máquinas de soporte vectorial (SVM) para encontrar el hiperplano óptimo que separa las clases de datos. LinearSVC se utiliza para problemas de clasificación binaria y multiclase, y es especialmente útil cuando los datos no son linealmente separables en el espacio de entrada.

También es un modelo bastante sencillo y rápido a nivel computacional y se suele utilizar para la clasificación de texto.

Los hiperparámetros a utilizar son prácticamente los mismos a los del modelo de Regresión Logística.

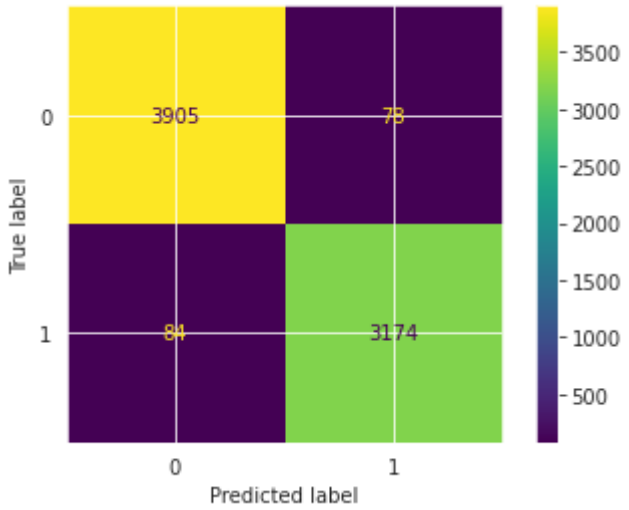
```
model2 = GridSearchCV(

    estimator = Pipeline(
        steps = [
            ("TfidfVectorizer", TfidfVectorizer(
                lowercase = True,
                max_features = 1000)),
            ("LinearSVC", LinearSVC(
                random_state=random_state,
                class_weight = {"0":1, "1":5669/4831}

             )),
        ]
    ),

    param_grid = {
            "TfidfVectorizer__ngram_range": [(1,1), (1,2), (1,3)],
            "LinearSVC__C": [1e-3, 1e-4, 1e-5,0.5, 1,1.5,2]



    },

    n_jobs = -1,
    cv = 5,
    verbose = 3
    )
```

Una vez se realiza el tuneado es el siguiente:

Modelo	Hiperparámetros	Número
LinearSVC	C	0.5
TfidfVectorizer	ngram_range	(1,2)

Resultados

Matriz de confusión



Accuracy

Train	Test
98,56	**97,78**

Reporte clasificación para el conjunto test

	precision	recall	F1-score	support
accuracy			0,98	7.241
macro avg	0,98	0,98	0,98	7.241
weigthed avg	0,98	0,98	0,98	7.241

4.3. Modelo Mutinomial Naive Bayes

Es un modelo de clasificación de tipo Bayesiano que se utiliza para problemas de clasificación multiclase. El modelo se basa en el teorema de Bayes para estimar la probabilidad de pertenencia a cada clase de salida en función de las características de entrada. MultinomialNB asume que las características son independientes entre sí, y utiliza una distribución multinomial para modelar la probabilidad conjunta de las características.

Como el TFIDVectorizer es prácticamente igual en todos los modelos, únicamente explicaré los parámetros que cambian de un modelo a otro.

Los hiperparámetros a utilizar son los siguientes:

- *alpha*: Es el parámetro de suavizado de Laplace, que se utiliza para evitar que el modelo se ajuste demasiado a los datos de entrenamiento. Este parámetro controla el nivel de suavizado, y se probarán tres valores diferentes: 0.01, 0.1 y 1.
- *class_prior:* Especifica las probabilidades a priori de las clases. Este parámetro es parecido al class_weight de los modelos anteriores para tratar las clases desbalanceadas.

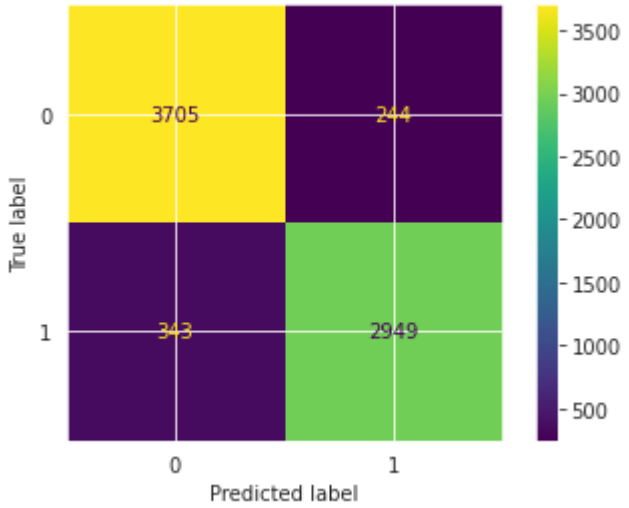
```
model3 = GridSearchCV(
    estimator = Pipeline(
        steps = [
            ("TfidfVectorizer", TfidfVectorizer(
                lowercase = True,
                max_features = 1000)),
            ("MultinomialNB", MultinomialNB(
                class_prior = [0.56, 0.44]
             )),
        ]
    ),

    param_grid = {
            "TfidfVectorizer__ngram_range": [(1,1), (1,2), (1,3)],
            "MultinomialNB__alpha": [0.01, 0.1, 1],



    },

    n_jobs = -1,
    cv = 5,
    verbose = 3
)
```

Una vez ejecutado el fit para los datos de entrenamiento y test, obtenemos los mejores parámetros para el tuneado:

Modelo	Hiperparámetros	Número
MultinomialNB	alpha	0.5
TfidfVectorizer	ngram_range	(1,3)

Resultados

Matriz de confusión



Accuracy

Train	Test
92,64	**91,89**

Reporte clasificación para el conjunto test

	precision	recall	F1-score	support
accuracy			0,92	7.241
macro avg	0,92	0,92	0,92	7.241
weigthed avg	0,92	0,92	0,92	7.241

4.4. Modelo de Red Neuronal MLPClassifier

Es un modelo de una red neuronal que se utiliza para problemas de clasificación binaria y multiclase. MLPClassifier es un tipo de red neuronal de alimentación hacia adelante que consta de una o más capas ocultas, y utiliza la retro propagación para ajustar los pesos del modelo. El modelo utiliza una función de activación no lineal en las capas ocultas para permitir la representación de características no lineales de los datos de entrada.

Este modelo sigue la misma estructura el pipeline dentro del GridSearch para entrenar el mejor modelo posible cogiendo automáticamente los mejores parámetros. Los hiperparámetros son los siguientes:

- *hidden_layer_sizes*: este parámetro especifica la cantidad de neuronas en la capa oculta de la red neuronal. En este caso se probará con 1 y 2.
- *early_stopping:* este parámetro especifica si se utilizará la técnica de detención temprana para evitar el sobreajuste del modelo. Se probarán dos opciones: False y True. Si se establece en True, el modelo se detendrá si la puntuación de validación no mejora durante varias iteraciones.

El modelo a definir sería el siguiente:

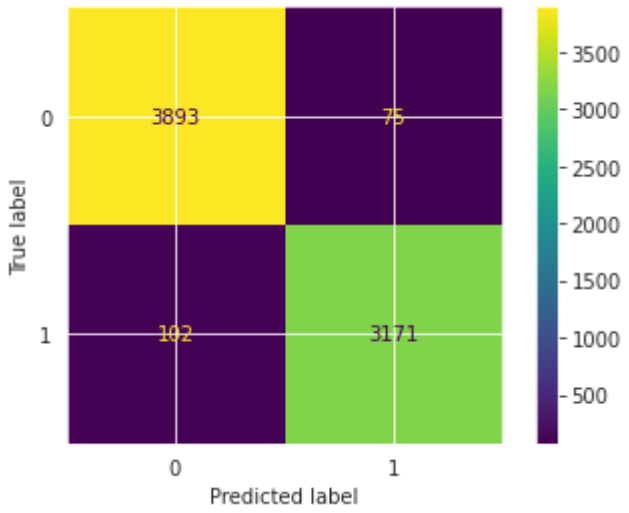
```
model4 = GridSearchCV(
    estimator = Pipeline(
        steps = [
            ("TfidfVectorizer", TfidfVectorizer(
                lowercase = True,
                max_features = 1000)),
            ("MLPClassifier", MLPClassifier(
                random_state=random_state
             )),
        ]
    ),

    param_grid = {
            "TfidfVectorizer__ngram_range": [(1,1), (1,2), (1,3)],
            "MLPClassifier__hidden_layer_sizes": [1,2],
            "MLPClassifier__early_stopping": [False,True]

    },

    n_jobs = -1,
    cv = 5,
    verbose = 3
)
```

Los hiperparámetros ganadores serían:

Modelo	Hiperparámetros	Número
MLPClassifier	hidden_layer_sizes	1
	early_stopping	True
TfidfVectorizer	ngram_range	(1,3)

Resultados

Matriz de confusión



Accuracy

Train	Test
98,48	**97,55**

Reporte clasificación para el conjunto test

	precision	recall	F1-score	support
accuracy			0,92	7.241
macro avg	0,92	0,92	0,92	7.241
weigthed avg	0,92	0,92	0,92	7.241

5. Comparación de modelos y conclusiones

5.1. Comparación de modelos

Todos los modelos en general han dado muy buenos resultados, con un accuracy bastante elevado. Pero el siguiente paso es elegir qué modelo es el mejor.

accuracy	Regresión Logística	Linear SVC	Multinomial NB	MLP Classifier
train	90,33	98,67	92,16	98,48
test	90,71	97,77	91,89	97,55

Tal y como se ha comentado anteriormente, el accuracy de todos los modelos es mayor del 90% por lo son muy buenos modelos todos y de esta forma, no añadí más hiperparámetros para mejorar el modelo.

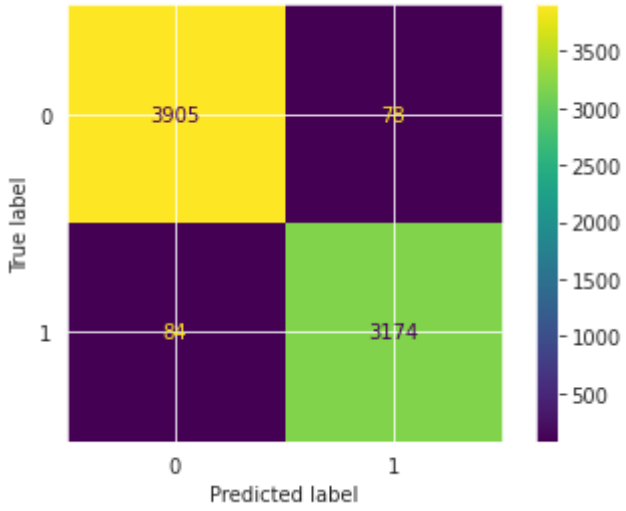
El modelo ganador es el LinearSVC aunque por sea una ínfima diferencia en el MLP Classifer, ya que el accuracy es mayor tanto en train como en test.

Para estudiar mejor el modelo Linear SVC hay que estudiar la matriz de confusión y el reporte de clasificación. De parte de la matriz de confusión, hay un número bastante pequeño de falsos positivos y falsos negativos, lo que es un claro indicador de que está clasificando bastante bien la gran mayoría de las noticias.

El accuracy en train y test es muy alto con un 98,56% y 97,78% respectivamente, además sugiere que el modelo no está sufriendo overfitting.

El informe de clasificación también nos muestra que el modelo tiene una alta precisión con un 98% y un recall del 97% para ambas clases.

Para concluir, que el modelo Linear SVC es bastante bueno para predecir si una noticia es falsa o verdadera.



5.2. Conclusiones

En general los modelos estudiados han sacado buenos resultados, pudiendo predecir si una noticia es falsa con un accuracy del 98%. Aunque hay que tener en cuenta que esta puntuación es muy alta, el modelo únicamente tiene dos categorías por lo que es bastante sencillo sacar una buena puntuación, lo que no suele ser habitual en NLP si se trabaja con datasets multiclase.

Ha sido complejo a nivel computacional y temporal ya que, al ser un modelo de 30.000 documentos, que es considerado relativamente grande, suelen tardar en ejecutarse los modelos entre 9 y 20 minutos.

También puede darse el caso del que dataset escogido no represente la realidad al ser solamente una muestra poco representativa. Pero a su vez, es complicado ya que tanto las noticias como tweets en redes sociales son variadas y heterogéneas que puede ser difícil clasificarlas entre toda la población. Por lo que quizás este modelo no es extrapolable a la realidad.

En un ámbito teórico este modelo si puede discernir entre noticas verdaderas y falsas y de esta forma evitar la confusión y desinformación ocasionadas por el gran volumen de información gracias a las redes sociales. Pero como opinión personal creo que sería bueno con dataframes medianos-grandes

6.Bibliografía

Ahmed H, Traore I, Saad S. “Detecting opinion spams and fake news using text

classification”, Journal of Security and Privacy, Volume 1, Issue 1, Wiley,

January/February 2018.

Ahmed H, Traore I, Saad S. (2017) “Detection of Online Fake News Using N-Gram

Analysis and Machine Learning Techniques. In: Traore I., Woungang I., Awad A. (eds)

Intelligent, Secure, and Dependable Systems in Distributed and Cloud Environments.

ISDDC 2017. Lecture Notes in Computer Science, vol 10618. Springer, Cham (pp. 127-

138).

Chat GPT consultas

L.Gascó Sánchez, Ph.d (2022) Módulo Text Mining

Martínez del Pino,Ernesto (2020) Estudio de identificación de autoría

7.Anexo

Al haber utilizado Google Collab se puede encontrar el código en los siguientes enlaces:

EDA:
https://colab.research.google.com/drive/1diqa1TVgvLvGWI5zc7redY3PGLDVVqvN#scrollTo=vYicm8e3NNmR

Preprocesado y modelado:
https://colab.research.google.com/drive/1zmgXiv1FAjaCHcQ7j74obCCn_PonipZ-#scrollTo=oIzZnUYIOikX&uniqifier=1

También se puede ver el código en GitHub:

https://github.com/esterfb3397/fake-news-project-tfm