REPORT DE PROYECTO MACHINE LEARNING

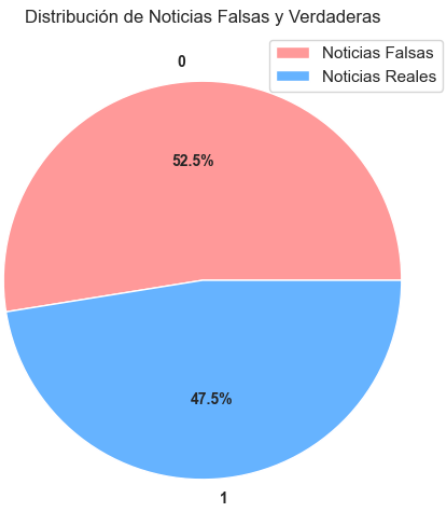
1.EDA

Antes de comenzar el EDA, en un notebook que he llamado fake news, cargue los dos CSV debido a que uno solo contenía las noticias falsas y el otro solo las verdaderas, por lo que los unte y cree una nueva variable denominada “label” en la que se diferencie cuáles son las noticias falsas y cuales las verdaderas, siendo 0 Fake e y 1 True.

En el EDA al principio se realiza una definición del problema y los objetivos que se plantea a la hora de hacer un modelo, en nuestro caso de clasificación

1.1 Exploración de datos

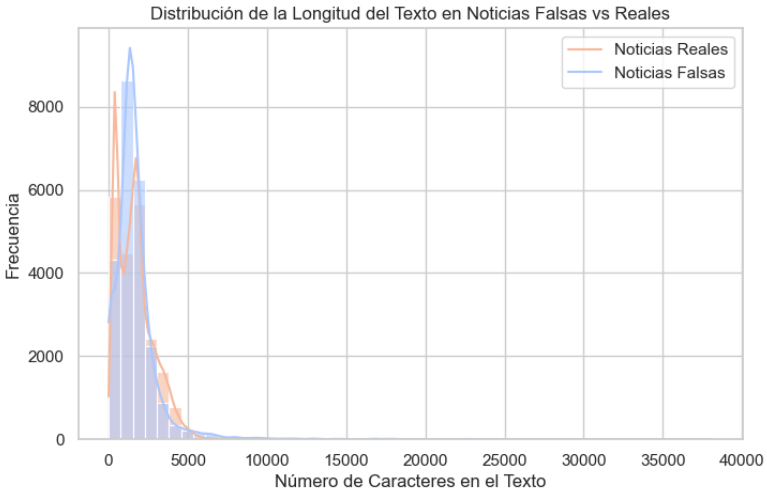
- La exploración general nos muestra que no hay datos nulos, que hay muchas categorías que más adelante se puede observar los problemas que puede causar. Y por último, el balanceo de las clases, que se puede observar que nuestro target, label, se encuentra equilibrada.

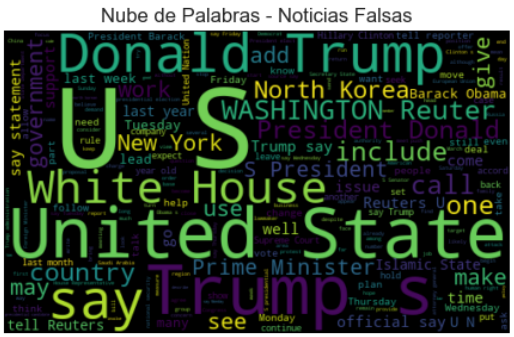


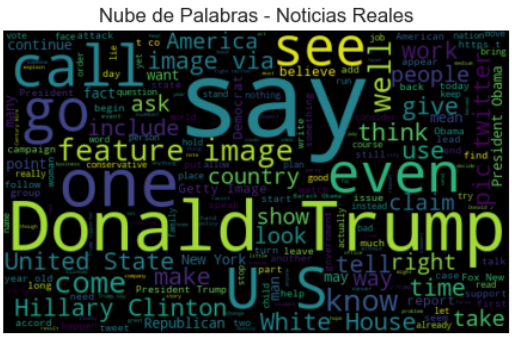
- En segundo lugar, también se realiza el estudio de estadísticas descriptivas correspondiente

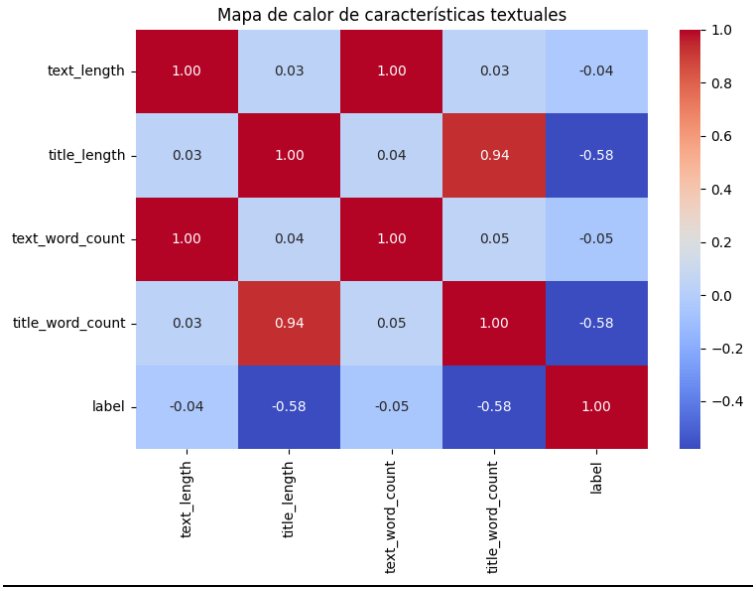
1.2 Preprocesamiento de los datos

- Duplicados: Se encuentra que se tiene 209 valores duplicados que puede entorpecer al modelo por lo que se elimina con drop duplicated, ya que no aporta valor estos datos
- Stop Word y Lematización
    - Como limpieza y preprocesamiento, al tratarse de un proyecto de machine learning se realiza una limpieza de texto mediante StopWord y lematización. Estos métodos ayudan a que el texto sea más legible para el análisis computacional
    - Al filtrar las palabras vacías (stop words) mediante if token.text.lower() not in stop_words, se elimina ruido y se conservan las palabras más relevantes para el análisis
    - El hecho de aplicar el mismo preprocesamiento tanto a la columna text como a la columna title asegura que ambos campos se normalicen de manera coherente.
    - La lematización transforma cada palabra a su raíz, lo que ayuda a reducir la variabilidad del lenguaje y mejora la consistencia en la representación del texto. Durante el preprocesamiento de los datos he aplicado el multiprocessing para procesar los datos más rápidamente ya que es un caso de gran volumen de datos.
    - La lematización proporciona de manera más precisa y coherente de las palabras, lo que hace que se comprenda mejor el "contexto" de las palabras ya que es necesario para la detención de noticias falsas.
    - La reducción con StopWords, además de quitar ruido, hace que computacionalmente sea mejor y hacer que el modelo se fije en las palabras importantes para la detección de noticias falsas.
- Visualizaciones: A continuación, se realiza unas visualizaciones que nos ayuda comprender mejor los datos, como la distribución de los subject, la longitud del texto diferenciando las noticias verdaderas y falsas y por último una nube de puntos en el que se muestra las palabras que más se repiten tanto en las noticias verdaderas como en las falsas





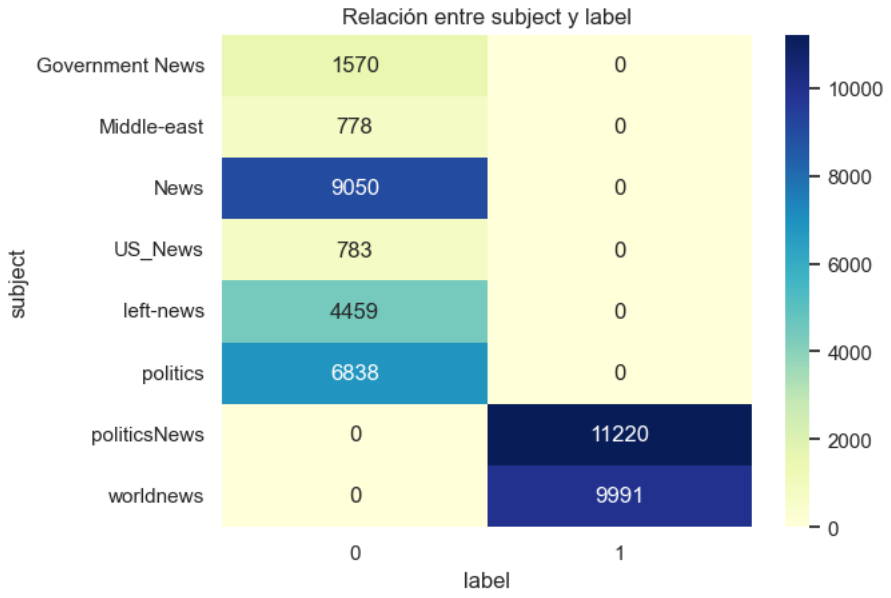




Con este gráfico se crea un conteo de título y texto se puede ver que tiene una correlación las longitudes por lo que puede ser útil en los modelos

- Tratamiento de Subject: Por último, se realiza un estudio de la columna subject debido a que su distribución de categorías puede causar sesgo ya que se provoca riesgo de

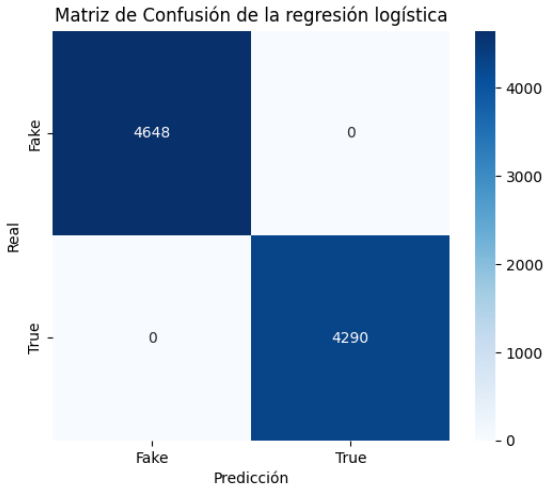
Data leakage, Si el objetivo es entrenar un modelo para detectar noticias falsas se puede basar en el contenido de la noticia y no en metadatos que ya "cantan" la clase, usar la variable subject tal cual podría ocasionar que el modelo "aprenda" una regla trivial. Esto se denomina data leakage, porque el modelo descubre un patrón que no se corresponde con el fenómeno real que quieres estudiar (detección de fake news), sino con la forma en que los datos fueron etiquetados o estructurados.

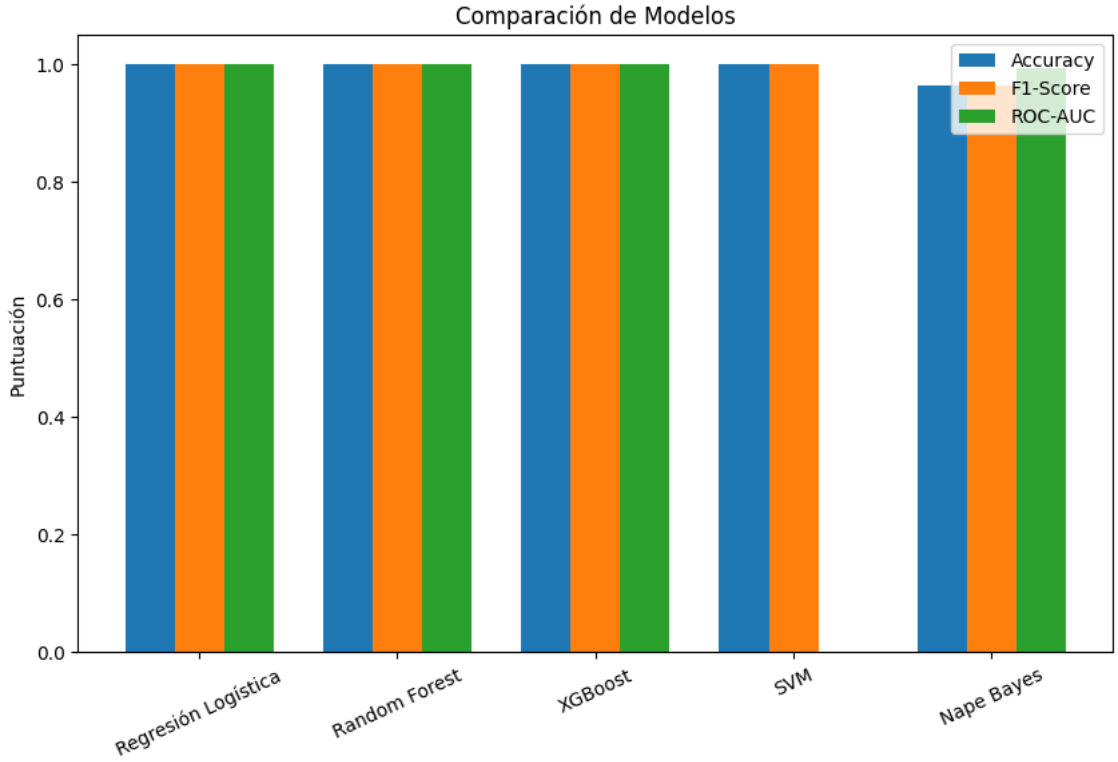


Aquí se puede observar la relación entre "subject"y "label" y su distribución, por lo cual se puede ver que las que se categorizan en "politicnews"y "worldnews," que son todas verdaderas tienen una fuerte relación. Por lo que esta diferencia perfecta puede causar problemas, la reajusto en dos categorías globales ("politics" y "general") para así poder generalizar mejor.

2.BASE LINE

Primero antes de incluir el tratamiento de subject hice unos modelos base para poder ver como interactuaba, por lo que se podía observar que había modelos como la logistic Regression encontraba una precisión de 1.





Por lo que comprobé con una noticia como afectaba la etiqueta de subject y posteriormente añadí el tratamiento al EDA para tener junto la transformación y guardarlo al resto de procesos y seguir probando en los modelos.

Los modelos que utilice son:

1. Regresión logística: accuracy 1
2. XG-Boost: accuracy 1
3. Gradient boosting: accuracy 1
4. NP-Bayes: 0,96
5. Random Forest: accuracy 1
6. Super Vector Machine: accuracy 1

ESTRUCTURA GENERAL DE LOS NOTEBOOKS DE MODELOS BASE

1. Label Encoder para subject**:** **Label Encoder** clasifica la variable en numérico, es más eficiente que One-Hot Encoding en términos de procesamiento y memoria.
2. Vectorización de Title y Text:
    a. Se utiliza el método **TF-IDF**, ya que destaca las palabras importantes, dándole un valor menor a las menos importantes, reduce el ruido.
    b. Para modelos como Random Forest le ayuda en entender y procesar los datos.
    c. El uso de stopwords y stemming y TF-IDF no afecta negativamente, sino que mejora la representación, ya que TF-IDF se enfocará en términos más significativos. Y el uso de Stemming y TF-IDF Reduce la dimensionalidad al tratar palabras con la misma raíz como una sola característica y evita dispersión en el modelo.
3. Finalmente se concatena los datos vectorizados y se realiza la división en x e y y la división correspondiente de train y test.

Otras observaciones: decidí prescindir de Super Vector Machine debido a que computacionalmente era muy pesado y los resultados eran muy similares a otros.

4. MODELOS CON SUBJECT TRANSFORMADA Y SIN SUBJECT

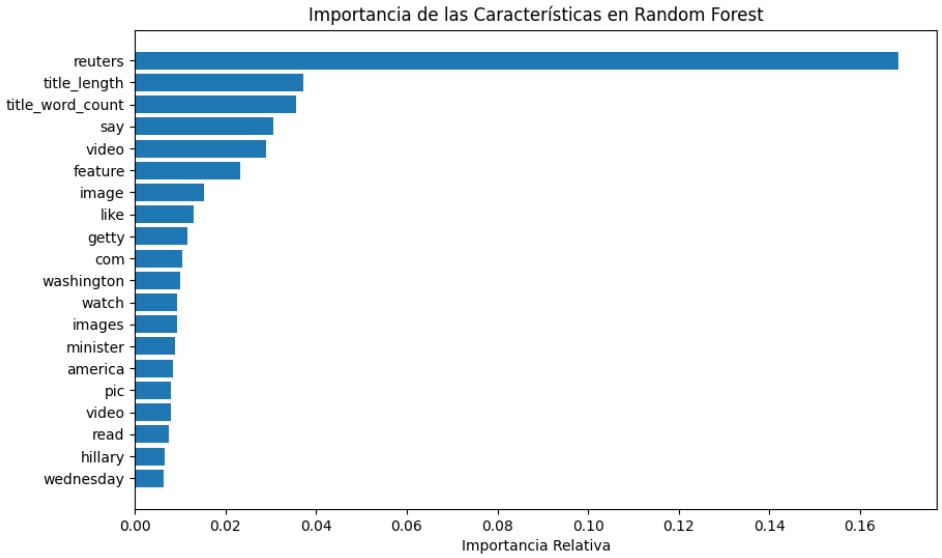
Ambos modelos tienen una estructura similar: vectorización y a continuación los modelos con diferentes medidas para observar la calidad de los modelos, como:

- Confusión matrix: para poder ver la cantidad de equivocación que comete el modelo
- Classification report: para poder ver la precisión y el accuracy y así poder observar la calidad de los modelos
- En algunos casos se incluye gráficos de importancia de palabras: para saber que palabras dan importancia el modelo y así entender como las clasifica.
- La pérdida logarítmica para indicar la capacidad de predicción en términos de probabilidad
- La precisión promedio y la desviación estándar
- Finalmente he realizado unos gráficos y tablas para resumir los resultados de las pruebas.

Los resultados son bastante similares, incluso un poco mejor sin la variable subject en algunos casos, pero la diferencia es mínima, sin embargo, decido incluirla en la siguiente prueba ya que no encarece los resultados significativamente.

5. MODELOS INCLUYENDO LA LONGITUD DE TEXTO Y TITULO

Tras la creación de la variable longitud de texto y titulo en el EDA para ver si podía afectar la extensión en las noticias y ver que tenía una relación significativa decido incluirla para ver si mejora los modelos, en efecto mejoran considerablemente y se puede observar cómo afecta en algunos de los modelos como en el Random Forest la extensión del título. Las noticias falsas suelen dar más detalles y ser más sensacionalista en los títulos para que sean más atractivas.



Comparación de modelos con las diferentes variables

1. Subject, text, title:

Model	accuracy	precision	recall	f1	roc_auc
logistic Regresion	0.992325	0.990452	0.993410	0.991927	0.992384
Random forest	0.997964	0.998108	0.997597	0.997852	0.997950
XGBoost	0.997897	0.997361	0.998209	0.997785	0.997912
Gradient Boosting	0.995816	0.993707	0.997505	0.995602	0.995899
Nape Bayes	0.948690	0.953424	0.937732	0.945509	0.948157

2. Sin subject y solo text y title

Model	accuracy	f1	precision	recall	roc_auc
logistic Regresion	0.992414	0.992019	0.990730	0.993314	0.999515
Random forest	0.998076	0.997972	0.997875	0.998069	0.999948
XGBoost	0.997919	0.997808	0.997360	0.998257	0.999948
Gradient Boosting	0.995860	0.995648	0.993660	0.997646	0.999084
Nape Bayes	0.948824	0.945666	0.953306	0.938157	0.988990

3. Con title, text, subject y length_title, length_text

Model	accuracy	precision	recall	f1	roc_auc
logistic Regresion	0.990781	0.986027	0.994684	0.990335	0.990974
Random forest	0.997941	0.997730	0.997931	0.997829	0.997947
XGBoost	0.998031	0.997593	0.998258	0.997924	0.998045
Gradient Boosting	0.996711	0.995579	0.997503	0.996540	0.996748
Nape Bayes	0.951174	0.950234	0.946739	0.948480	0.950960

Las diferencias son minimas entre con subject y sin subject, aunque no aporte mucho la mantengo en el análisis debido a que no perjudica. Se puede observar que utilizando text_length y title_length se puede obtener una mayor precisión.

Elección de los mejores modelos:

1. XGBoost → primer lugar

-Máxima precisión y estabilidad (99.64% en cross-validation con solo 0.0014 de desviación estándar).

- Mejor Log Loss (0.00648), lo que significa que sus probabilidades son las más confiables.

- Solo 12 errores en la matriz de confusión. Es el mejor modelo en términos de precisión, estabilidad y calibración de probabilidades.

2. **Gradient Boosting** → segundo lugar

- Desempeño casi idéntico a XGBoost en precisión media (99.50%).

- Mejor Log Loss que Random Forest (0.0152 vs 0.1359), lo que significa que estima mejor las probabilidades.

- Desviación estándar muy baja (0.0016), lo que indica estabilidad.

- Aunque comete más errores que XGBoost y Random Forest (24 vs 12), pero sigue siendo una excelente opción.

3. **Random Forest** → tercer lugar

- Gran precisión, pero menor calibración de probabilidades (Log Loss 0.1359).
- Igual número de errores que XGBoost (12), pero menor confiabilidad en las probabilidades.
- Desviación estándar baja (0.0016), lo que indica estabilidad.

6.Grid-Search

He intentado hacer un GridSearch para encontrar los mejores parámetros los tres mejores modelos, sin embargo, he tenido problemas computacionales ya que el modelo de Gradient Boosting no termina de entrenar debido a que es muy pesado computacionalmente, por lo que le he descartado.

El modelo con resultados más eficiente es el Random Forest con una precisión de 0,99797, aunque el XGBoost tiene una precisión muy similar (0.99770).

Por lo que el modelo que eligiría después de todas las pruebas es el **Random Forest** con estas características:

- `'max_depth': None,`
- `'min_samples_leaf': 1`
- `'min_samples_split': 5`
- `'n_estimators': 200`

7. Red Transformer BERT

He probado hacer una red Bert en Google colab para poder acceder a GPUs, sin embargo, como los recursos son limitados finalmente he utilizado DistilBert que es una versión optimizada y especializada en clasificación de texto.

Resultados:

```
Época 1/2
Pérdida promedio: 0.007961040178913678
Época 2/2
Pérdida promedio: 0.0011333290111552625
Precisión del modelo: 0.9994
Reporte de clasificación:
              precision    recall  f1-score   support

           0       1.00      1.00      1.00      4650
           1       1.00      1.00      1.00      4330

    accuracy                           1.00      8980
   macro avg       1.00      1.00      1.00      8980
weighted avg       1.00      1.00      1.00      8980
```

Clase 0 (Noticias Falsas) y Clase 1 (Noticias Verdaderas) tiene un 100% en todas las métricas. Todas las predicciones coinciden exactamente con las etiquetas verdaderas. Precisión (accuracy) = 99.94% → De cada 10,000 ejemplos, solo 6 se clasifican incorrectamente.

- Prueba para evaluar el modelo en entrenamiento y prueba: como los resultados son sospechosos realizo una prueba con accuracy y ver si hay mucha diferencia entre test y train:
  Precisión en entrenamiento: 0.9997
  Precisión en prueba: 0.9994

  1. *Precisión en entrenamiento = 99.97%*
     Esto indica que el modelo ha aprendido a clasificar correctamente casi todos los ejemplos en el conjunto de entrenamiento. Muy baja pérdida (loss) → El modelo ha minimizado el error casi por completo.

  2. Precisión en prueba = 99.94%. La precisión en el conjunto de prueba es casi igual a la de entrenamiento, lo que indica que no hay un overfitting severo.
  Si hubiera overfitting, la precisión en prueba sería mucho más baja que en entrenamiento.

DECISIÓN FINAL:

Los modelos tradicionales pueden ser de bastante utilidad para predecir noticias falsas pero el modelo BERT puede ser mucho más útil debido a su capacidad de entender el contexto y puede captar la ironía, el sarcasmo y los sesgos.

Además, no necesita crear características manuales, como la transformación de datos o la tokenización manual.

Por otro lado, puede ser de utilidad ya que aprende patrones generales y a la hora de entrenar con datos nuevos sería mejor.

Sin embargo, depende de los recursos que se tenga debido a que computacionalmente es mucho más pesado.

Cosas que se pueden tener en cuenta:

1. Añadir datos debido a que esta base de datos es fácil de predecir
2. Probar otras combinaciones de variables
3. Diferentes parámetros en los modelos y la red
4. Probar más preprocesamientos