Análisis de la presencia de Clickbait en titulares de notícias online

Weronika Jaroszek, Manuel Tendero, Jose Picón, Isaac Nicolás

Abstract. La aparición de las redes sociales y el rol cada vez más importante como proveedor de noticias, ha hecho que cualquier persona pueda producir y difundir contenidos para que sean leídos por otras personas. Las barreras tradicionales para la publicación de información en redes sociales han desaparecido, y con ello, muchos principios periodísticos básicos, como la verificación de las fuentes, la comprobación de los hechos ocurridos y la responsabilidad. (Bourgonje, Moreno y Rehm, 2017). Es por eso que el presente trabajo se centra en la aplicación de diferentes modelos para la detección del clickbait en titulares de notícias online, ya que se trata de algo muy presente en nuestro día a día y que conviene saber identificar.

Palabras clave [“clickbait”, “redes sociales”, “algoritmo”, “ texto”, “detection”]

1. Introducción

El clickbait hace referencia a un tipo de anuncio de contenido web que está diseñado para conseguir que sus lectores hagan click en un enlace adjunto. Normalmente, se difunde en redes sociales en forma de breves mensajes provocativos (Potthast, Köpsel, Stein y Hagen, 2016). Estos artículos tienen títulos engañosos diseñados para captar la atención de sus lectores con el fin de que estos accedan a las páginas. Los clickbait, son utilizados en la monetización de la página de destino o para la difusión de fake news (Pujahari y Sisodia, 2020).

La importancia en la detección del clickbait se fundamenta en que la avalancha de clickbait en las redes sociales, puede hacer que estos se conviertan en una especie de spam y que sea molesto para los usuarios. La adopción de esta técnica por parte de los editores de noticias es un factor que está generando cierta preocupación entre el público. La detección automática de clickbait aportaría una solución para ayudar a los usuarios de redes sociales los respectivos mensajes(Potthast et.al, 2016). Por otro lado, uno de los principales problemas en la detección de clickbait es la categorización de los titulares analizados. Los diferentes estudios apuntan que las frases que tienen un mismo tipo de estructura y contexto pueden caer en cualquier de las dos categorías de clickbait y no clickbait (Pujahari y Sisodia, 2020).

2. Trabajos Relacionados, antecedentes

A continuación se exponen diferentes resúmenes de distintos trabajos que proporcionan una visión amplia del fenómeno clickbait.

Potthast et.al (2016) en su trabajo titulado *clickbait detection* proponen la construcción de un modelo basado en 215 características que están divididas en tres categorías relativas, a saber: mensaje teaser, página web analizada y la meta información. Para el análisis del mensaje teaser se analizan los rasgos del texto. Por ejemplo, se analizan los paquetes de texto, las imágenes y etiquetas asociadas a dicho texto, el sentimiento o polaridad de los textos a través de técnicas de PNL. Para el análisis de la página web se utiliza de nuevo el análisis de los paquetes de palabras y se mide la legibilidad y la longitud de las mismas. Por último, para la meta información se analiza el remitente del texto y si este ha sido compartido con los datos asociados de temporalidad.En cuanto al modelo generado, dividen el set de datos en una proporción 2:1 para train y test.Para evitar el sobreajuste, descartan todas las características del análisis con pesos de menos del 1% para el set de entrenamiento. Antes del

entrenamiento se equilibran los datos mediante el sobremuestreo. Se aplican los algoritmos de aprendizaje logístico, Bayes y random-forest y se implementan en Weka con los parámetros por defecto.

Por otro lado, Chakraborty , Paranjape , Kakarla y Ganguly (2016) en su trabajo titulado *Stop Clickbait: Detecting and preventing clickbaits in online news media*, proponen un método para la detección de clickbait. Para la realización de su estudio, recopilan artículos no clickbait de la plataforma Wikinews y artículos clickbait de plataformas como BuzzFeed, Viral Nova entre otras. Una vez recogidos los datos, se realiza un análisis lingüístico mediante la herramienta Stanford CoreNLP. El análisis se centra en la estructuración de las frases para la categorización de los titulares clickbait y no clickbait. Se utilizan cuatro técnicas de selección de las características, a saber: estructura de la frase, patrones de palabras, lenguaje clickbait y características del n-grama. Los algoritmos utilizados son : decisión tree, random forest y SVM.

Por último, Pujahari y Sisodia (2020) en su artículo titulado *Detección de clickbaits mediante técnicas de categorización múltiple* proponen una técnica de categorización híbrida para separar los artículos clickbait de los que no lo son, integrando diferentes características, la estructura de las frases y la agrupación. Durante la categorización preliminar, los titulares se separan utilizando once características. Después, los titulares se recategorizan utilizando la formalidad de la frase y medidas de similitud sintáctica. En la última fase, los titulares se recategorizan de nuevo aplicando la agrupación mediante la similitud de vectores de palabras basada en el enfoque t-Stochastic Neighbourhood Embedding (t-SNE). Tras la categorización de estos titulares, se aplican modelos de aprendizaje automático al conjunto de datos para evaluar los algoritmos de aprendizaje automático.

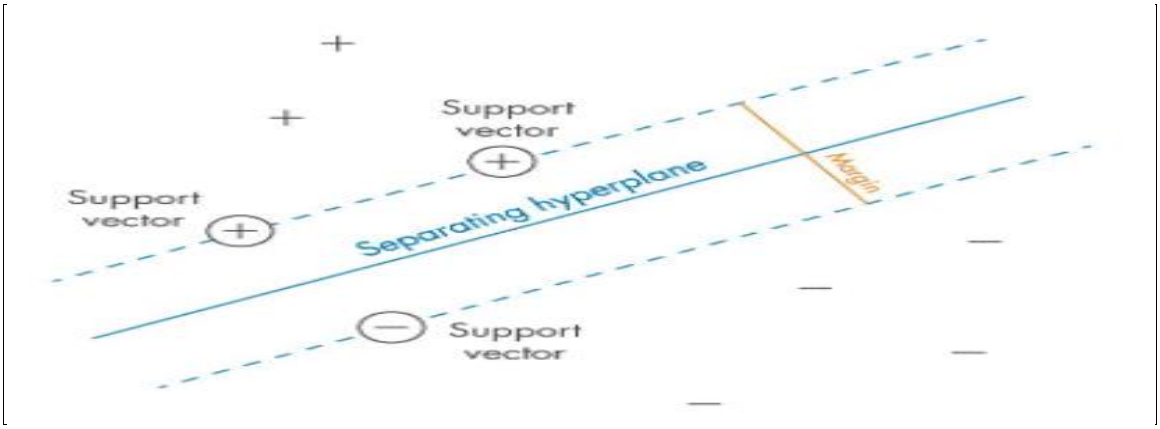
3. Tecnologías usadas

En el presente apartado se expone una breve explicación teórica de las técnicas utilizadas.

Support Vector Machine

Support vector machine (SVM) es un algoritmo de aprendizaje automático que es muy utilizado en problemas de clasificación y regresión. El objetivo principal de este algoritmo es encontrar un hiperplano donde se separen dos clases diferentes de puntos de datos. Este algoritmo solamente encuentra este hiperplano en problemas que permiten una separación lineal. En la mayoría de casos, el modelo maximiza el margen flexible permitiendo un pequeño número de los vectores de soporte que hacen referencia a un subconjunto de observaciones de entrenamientos. La versión estándar de este algoritmo está diseñado para problemas de clasificación binaria. Este modelo pertenece a una clase de algoritmos de ML conocidos como métodos kernel. Las funciones de kernel asignan los datos a un espacio dimensional diferente, que suele ser superior, con el objetivo de que resulte más fácil la separación de las clases después de esta transformación, simplificando los límites de decisión más complejos (Mathworks, sf).

Imagen 1. Definición del "margen" entre clases: el criterio que los SVM intentan optimizar



Fuente: mathworks (sf)

Haciendo referencia a los creadores de este algoritmo, Boser, Guyon y Vapnik (1992), este modelo ajusta automáticamente la capacidad de la función de clasificación maximizando el margen entre los ejemplos de entrenamiento y el límite de la clase. La función de clasificación depende solo de los llamados patrones de apoyo. Estos patrones de apoyo o support vector son aquellos ejemplos de entrenamiento que están más cercanos al límite de decisión y suelen ser un pequeño subconjunto de los datos de entrenamiento. El modelo opera con una gran clase de funciones de decisión que son lineales en sus parámetros pero que no se limitan a dependencias lineales en los componentes de entrada.

Naive-Bayes

El clasificador Naive-Bayes es un algoritmo que se basa en el teorema de Bayes para la clasificación de los datos. En la detección de clickbait se cuenta el número de veces que aparece una palabra en un titular de clickbait. Después, se calcula la probabilidad de que un titular sea clickbait en comparación con la probabilidad de que sea un titular real (Sim, 2020).

Siguiendo a Yildirim (2020), este algoritmo utiliza funciones para predecir la variable objetivo al igual que otros algoritmos de aprendizaje supervisado. La diferencia recae en que el algoritmo Naive-Bates asume que las características son independientes entre sí y no hay correlación entre las características. El desempeño de este algoritmo se basa en el cálculo de la probabilidad de una clase dado un conjunto de datos característicos. La suposición de que todas las funciones son independientes hace que el algoritmo bayesiano ingenuo sea **más rápido** en comparación con otros algoritmos. Además, tiene un buen desempeño con datos de gran dimensión, como clasificación de texto o detección de correo no deseado. Por otro lado, la suposición de que todas las funciones son independientes hace que el algoritmo bayesiano sea menos preciso que los otros algoritmos.

XGBoost

XGBoost es un clasificador de árbol de decisión con refuerzo de gradiente. En boosting, los árboles se construyen secuencialmente de forma que cada árbol posterior tiene el objetivo de reducir los fallos del árbol anterior. Estos árboles posteriores se llaman aprendices débiles. Cada uno de estos aprendices aporta una información relevante para la predicción. Esto permite a la técnica de boosting producir un aprendiz fuerte resultante de la combinación

eficaz de los aprendices débiles.La potencialidad de este algoritmo se basa en su escalabilidad, que fomenta un aprendizaje rápido a través de la computación paralela y distribuida con un uso eficiente de memoria (Diveki, 2019).

Siguiendo a Greatlearning (2020), dentro de las características de este algoritmo encontramos: un aprendizaje regularizado donde el término de regularización ayuda a suavizar los pesos finales aprendidos para evitar el sobre ajustamiento; aumento de gradiente de árbol donde el modelo se entrena de forma aditiva debido a que el modelo de conjunto de árbol no se puede optimizar utilizando métodos de optimización tradicionales en el espacio euclidiano. Por último, encontramos el submuestreo de columna y contracción. Además del objetivo regularizado, se utilizan técnicas adicionales para evitar aún más el sobreajuste.

Modelo Preentrenado Spacy

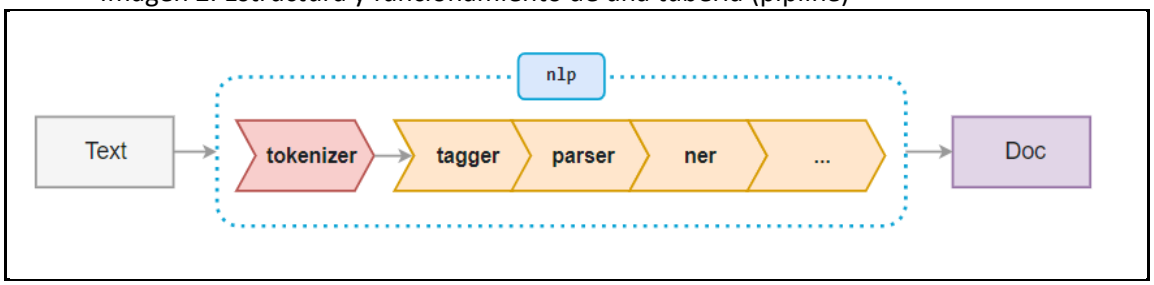
SpaCy es una biblioteca gratuita de código abierto para el procesamiento avanzado del lenguaje natural (NLP) en Python. spaCy está diseñado específicamente para uso en producción y lo ayuda a crear aplicaciones que procesan y "entienden" grandes volúmenes de texto.

A continuación se detallan las características y funcionalidades de Spacy según su documentación oficial Spacy(2022):

- **Tokenización**: segmenta el texto en palabras, signos de puntuación etc.
- **Part-of-speech (POS)**: asigna tipologías de palabras a tokens, como verbo o sustantivo
- **Dependency Parsing**: asigna etiquetas de dependencia sintáctica, que describen las relaciones entre tokens individuales, como sujeto u objeto-
- **Lemmatization**: asignación de la forma básica de las palabras.
- **Sentence Boundary Detection (SBD)**: encuentra y segmenta oraciones individuales
- **Named Entity Recognition (NER)**: etiqueta objetos del mundo real con nombre, como personas, empresas o ubicaciones.
- **Entity Linking (EL)**: elimina ambigüedades de entidades textuales a identificadores únicos en una base de conocimientos
- **Similarity**: compara palabras, extensiones de texto y documentos y qué tan similares son entre sí.
- **Text Classification**: asigna categorías o etiquetas a un documento completo o partes de un documento.
- **Rule-based Matching**: encuentra secuencias de tokens en función de sus textos y anotaciones lingüísticas, similares a las expresiones regulares
- **Training**: actualiza y mejora las predicciones de un modelo estadístico.
- **Serialization**: guardará objetos en archivos o cadenas de bytes.

Por otro lado, esta librería también nos proporciona otras funcionalidades como la de pipeline o tubería. Esto hace referencia cuando se llama nlp a un texto, spacy primero tokeniza el texto para producir un Doc objeto. Luego se procesa en varios pasos diferentes; esto también se conoce como canalización de procesamiento. La canalización utilizada por los modelos predeterminados consta de un etiquetador, un analizador y un reconocedor de entidades. Cada componente de canalización devuelve el procesado Doc, que luego se pasa al siguiente componente (spaCy, 2022).

Imagen 2. Estructura y funcionamiento de una tubería (pipline)



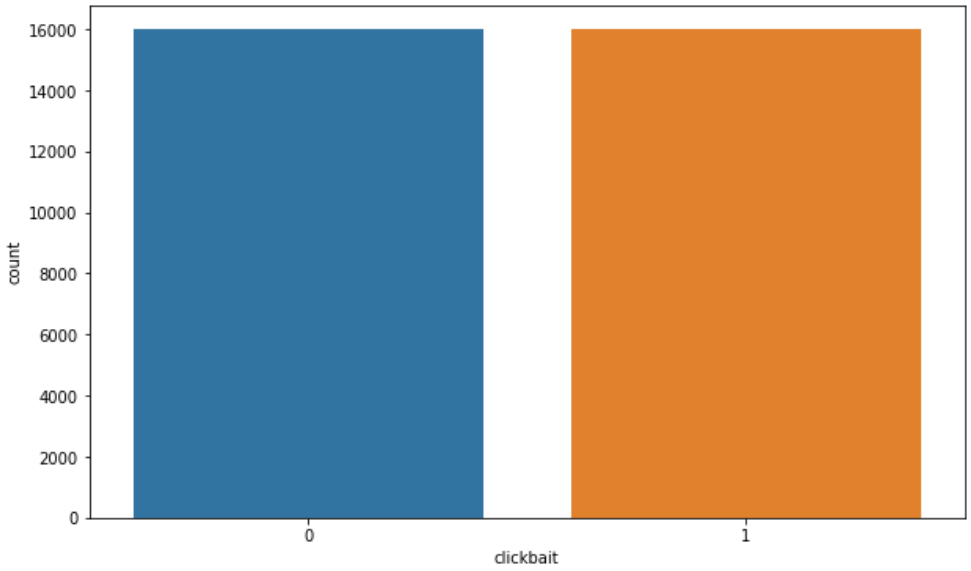
Fuente: Spacy annotations (2022).

4. Desarrollo de la propuesta

En este apartado se expone cómo se han aplicado los diferentes modelos al problema objetivo de predicción de titulares clickbait. Los datos utilizados se han obtenido de kaggle(2020)

En el preprocesamiento del dataset, se ha comprobado que la muestra está balanceada, siendo las instancias de titulares clickbait de 1599 y 1601 instancias para los titulares reales. Además, no se detectan valores nulos en ninguna variable.

Imagen 3. Distribución de las instancias clickbait y no clickbait



Fuente: elaboración propia

Una vez comprobadas estas características se ha procedido a realizar el split de nuestro dataset entre train y test. Posteriormente, se definen las siguientes funciones para el procesamiento del texto.

Funciones realizadas:

- token(): separa el texto de los titulares y los convierte en elementos de una lista
- lowercase(): transforma las mayúsculas en minúsculas.
- punctuations(): elimina los signos de puntuación
- sw() : elimina las stopwords del texto en el idioma “english” debido a que nuestros titulares están escritos en inglés.

- lemmatization(yes): encuentra el lema correspondiente entre las palabras en forma derivada o flexionada(plural, genero, conjugación entre otras formas)
- numbers(text): elimina los números del texto

Una vez definidas las funciones, las aplicamos a nuestro set de datos de entrenamiento y de test. Después, se aplica TF-IDF a nuestros datos. TF-IDF es una abreviatura de *Term Frequency Inverse Document Frequency.* Este es un algoritmo que transforma el texto en una representación de números para poder aplicar el algoritmo de predicción. El procedimiento que utiliza es el de la consideración del peso total de las palabras de un documento y es utilizado para el procesamiento de las palabras más frecuentes (Chaudhary, 2020).

En el presente trabajo se han aplicado los algoritmos de Multinomial Naive-Bayes, SVM, Xgboost y Spacy. A continuación se explica cómo se han aplicado en nuestros datos.

Primeramente se utiliza la función GridSearchCV(). GridSearch es un método de ajuste de parámetros; es decir, realiza una búsqueda exhaustiva entre todas las selecciones de parámetros seleccionados, a través de bucles y probando todas las posibilidades, el parámetro de mejor rendimiento es el resultado final. Además,una vez que se encuentre el mejor modelo (estimador) durante la validación cruzada, se volverá a entrenar en todo el conjunto de entrenamiento(programmerclick, sf).

Por otro lado, los parámetros establecidos son los siguientes:

- param_grid es el valor del parámetro a optimizar.
- n_jobs= -1 hacer referencia al número de núcleos de CPU.
- Cv= 5 es el número de validación cruzada elegido.
- verbose= hace referencia a la salida para cada submodelo

A continuación se explican los diferentes parámetros de cada algoritmo.

Naive-Bayes

Para la aplicación de este modelo se define lo siguiente

- params = {'alpha': [0.01, 0.1, 0.5, 1.0, 10.0] }
- model = GridSearchCV(MultinomialNB(), param_grid=params, n_jobs=-1, cv=5, verbose=4).fit(tfidf_train, y_train)

El parámetro alpha escogido para este modelo son diferentes valores donde se aplica el suavizado de Laplace. Esta es una técnica de suavizado que ayuda a abordar el problema de la probabilidad cero en el algoritmo de aprendizaje automático Naïve Bayes. El uso de valores alfa más altos empujará la probabilidad hacia un valor de 0,5, es decir, la probabilidad de una palabra igual a 0,5 tanto para las reseñas positivas como para las negativas (Jayaswal, 2020).

SVC

Para la aplicación de este modelo se define lo siguiente

- parameters = { 'C': [1.0, 10], 'gamma': [1, 'auto', 'scale'}

- model = GridSearchCV(SVC(kernel='rbf'), parameters, cv=5, n_jobs=-1,verbose=4).fit(tfidf_train, y_train)

Los parámetros escogidos son los siguientes:

- gamma: es un parámetro para los hiperplanos no lineales. Cuanto mayor sea el valor de este parámetro, el modelo intenta ajustarse exactamente al conjunto de datos de entrenamiento. El aumento de gamma resulta en un sobreajuste del modelo (Fray, 2018).
- C: hace referencia a la regularización. La regularización consiste en aplicar una penalización al aumento de la magnitud de los valores de los parámetros para reducir el sobreajuste . Cuando se entrena un modelo de regresión logística, está eligiendo parámetros que le proporcionan mejor ajuste a los datos. Esto se traduce en una minimización del error entre lo que predice el modelo para su variable dependiente dados sus datos, en comparación con lo que realmente es su variable dependiente (Wikipedia, 2020).

Por otro lado, en la función del modelo SCV le pasamos el parámetro '*kernel=rbf*'. Los parámetros del núcleo seleccionan el tipo de hiperplano utilizado para separar los datos. 'Rbf' utiliza un hiperplano no lineal (Fray, 2018).

XGBoost

Para la aplicación de este modelo se define lo siguiente:

- parameters = {'min_child_weight': [5, 10], 'gamma': [0.5, 2, 5], 'subsample': [0.5, 1.0], 'colsample_bytree': [0.5, 1.0], 'max_depth': [5, 10]}
- model = GridSearchCV(XGBClassifier(), parameters, cv=5, n_jobs=-1,verbose=4).fit(tfidf_train, y_train)

Los parámetros escogidos son los siguientes:

- min_child_weightes: es el peso mínimo necesario para crear un nuevo nodo en el árbol. Un tamaño más pequeño min_child_weight facilita la creación de elementos secundarios que correspondan a menos muestras que da como resultado árboles más complejos. No obstante, es más probable el sobreajuste (Spark, 2017).

- gamma : Reducción de pérdida mínima requerida para hacer una partición adicional en un nodo hoja del árbol (DmlcXgboost, 2022).
- Subsample Relación de submuestra de la instancia de entrenamiento (DmlcXgboost, 2022).
- colsample_bytree: Proporción de columnas de submuestra al construir cada árbol. (DmlcXgboost, 2022).
- max_depth: es el número máximo de nodos permitidos desde la raíz hasta la hoja más

  lejana de un árbol (Spark, 2017).

Modelo Preentrenado Spacy

El código utilizado para el desarrollo del modelo propuesto pertenece a Shrivarsheni (2020).

Para la aplicación de este modelo primeramente se carga el modelo pre entrenado de "en_core_web_sm". En este, se cargan las tuberías(pipes) :

- tagger: Asigna etiquetas de parte del discurso.

- Parser:Asigne etiquetas de dependencia.
- ner : Detectar y etiquetar entidades nombradas.

Además, se crea una tubería(pipe) nueva denominada textcat donde se crean etiquetas de positivo o negativo para hacer referencia a la existencia o no de un título clickbait.

Posteriormente se utiliza la función add_label(). Esta se utiliza para poder asignar una etiqueta a la categorización antes de utilizar el modelo. Esta función nos permite realizar una categorización para cada titular de nuestro set de datos encapsulado en un elemento de una lista. Acto seguido, utilizamos la función load data() la cual prepara los datos y los separa como en train y test. Además, con la función evaluate() programamos una función que evalúe los resultados obtenidos. Los parámetros que se introducen son diferentes métricas como precisión, recall, loss y F-1 Score.

Una vez definido todo lo anterior, indicamos que utilizamos el componente texcat e iniciamos el modelo con el optimizador nlp.begin_training. Con esto, se consigue que el modelo entrene varias veces sobre el mismo conjunto de datos de entrenamiento que a su vez es dividido en pequeños paquetes o batches. Por último con la funcionalidad de update se actualiza el modelo con la evaluación de los resultados.

5. Resultados obtenidos

Tras la aplicación de los modelos a nuestro set de datos, se exponen a continuación los resultados obtenidos.

Tabla 1. Resultados de los modelos aplicados

Metodo	Acurracy	Precision	Recall	F-1 Score	ROC-AUC
Support Vector Machine	95%	95%	95%	95%	0.96
Naive Bayes	96%	96%	96%	96%	0.96
Xgboost	84%	85%	85%	85%	0.85
spaCy	97%	97%	97%	97%	----

Fuente: Elaboración propia

En cuanto a las métricas de evaluación que comparten los cuatro modelos, se observa que los resultados son positivos y altos. Esto puede deberse a que el set de datos con el que hemos trabajado proveniente de la página web Kaggle, es un dataset perfectamente balanceado y cuya preparación pueda perseguir unos resultados óptimos. Por otro lado, la elección correcta de los hiperparametros de cada algoritmo puede favorecer en los resultados obtenidos. En cuanto a las métricas de Accuracy, Precision, Recall y F-1 Score el mejor algoritmo es el de spaCy. Aunque los resultados no difieren mucho entre los diferentes modelos, esta mayor puntuación de spaCy también puede estar relacionada con las funcionalidades específicas de este modelo en el preprocesamiento del texto y su aplicación del modelo.

Por otro lado, en cuanto a la curva Roc y el área bajo la curva, podemos observar que los modelos SVM y Naive Bayes consiguen una misma puntuación de 0.96. Esto nos está indicando que nuestros modelos son efectivos en la distinción de las categorías de titulares.

Para el modelo de spaCy encontramos el parámetro de Loss o pérdida. Este hace referencia a la minimización del error. Loss es un valor resultante en donde la función de pérdida es reducida a un solo número, un valor escalar que nos permite clasificar y comparar las conclusiones candidatas (**Brownlee**, 2019). De las cinco iteraciones utilizadas en el modelo de spaCy se ha obtenido un valor máximo de 7.3 y un valor mínimo de 0.23.

6. Discusión abierta

Con la llegada de las redes sociales, la difusión de la información ha sido un elemento que ha condicionado el modo en que consumimos dicha información. La presencia del fenómeno clickbait es un ejemplo de ello. Con la aparición de los modelos de detección de clickbait se pretende combatir por un acceso a una información fiable y veraz. Este tipo de modelos y su desarrollo puede servir como efecto disuasorio de su utilización en masa por diferentes portales web. En cuanto al presente trabajo se plantean la siguiente tesitura. El procesamiento del texto elimina algunas características del texto, que pese a no ser relevantes para la clasificación binaria de los titulares como respaldan los diferentes resultados obtenidos, pueden influenciar en una comprensión más profunda de las características de los titulares. No obstante, se ha identificado que existen palabras comunes en los titulares clickbait las cuales despiertan la curiosidad de los lectores y les empuja a seguir el link asociado a dichos titulares.

7. Conclusiones y líneas abiertas

En general, pudimos utilizar algoritmos de aprendizaje automático como Naive Bayes, spaCy, XGBoost y SVM para clasificar con precisión los titulares de clickbait frente a los que no lo son. Los resultados fueron bastante buenos, dentro del rango de 84 a 96 % para puntajes de precisión y de 85 a 97 % para puntajes de recuperación. Damos prioridad a la recuperación, ya que sería más valioso minimizar los falsos negativos (clasificar el clickbait como no clickbait) que viceversa. Como el aprendizaje automático pudo funcionar tan bien, definitivamente existe un caso de uso en el mundo real para implementar una solución de aprendizaje automático para filtrar o marcar el clickbait antes de que un lector tenga que visualizar y evaluar el título por sí mismo. Al analizar los coeficientes de los modelos que funcionaron mejor, pudimos interpretar y obtener una idea de cómo los modelos determinaban si un titular era clickbait o no.

Dentro de los resultados, pudimos extraer una lista de combinaciones de palabras de los titulares que se predijo que probablemente serían clickbait, como se ve a continuación: able answer, able stop, absolute best, absolutely stunning, abuse case, academy award, academy illinois.

Dentro de los resultados, los modelos que tuvieron el mejor rendimiento fueron los modelos spaCy y Naive Bayes, ambos por encima del 96% en términos de métricas de rendimiento. Una cosa a considerar es que las métricas de evaluación que comparten los cuatro modelos, se observa que los resultados son positivos y altos. Esto puede deberse a que el set de datos con el que hemos trabajado proveniente de la página web Kaggle, es un dataset perfectamente balanceado y cuya preparación pueda perseguir unos resultados óptimos.