Uso de arquitecturas de inteligencia artificial para la determinación y análisis de sentimientos en tweets

Daniela Aguiar Valencia

Monografía para optar al título de Especialista en Analítica y Ciencia de Datos

Asesor
Javier Fernando Botia Valderrama, Doctor

Universidad de Antioquia
Facultad de Ingeniería
Especialización en Analítica y Ciencia de Datos
Medellín
2023

Cita	(Aguiar Valencia, 2023)
Referencia **Estilo APA 7 (2020)**	Aguiar Valencia D. (2023). *Uso de arquitecturas de inteligencia artificial para la determinación y análisis de sentimientos en tweets* Trabajo de grado especialización, Especialización en Analítica y Ciencia de Datos, Universidad de Antioquia, Medellín, Antioquia, Colombia, 2023.

Especialización en Analítica y Ciencia de Datos, Cohorte IV.
Centro de Investigación Ambientales y de Ingeniería (CIA).

Centro de Documentación Ingeniería (CENDOI)

Repositorio Institucional: http://bibliotecadigital.udea.edu.co

Universidad de Antioquia - www.udea.edu.co

Rector: John Jairo Arboleda Céspedes.
Decano/Director Julio César Saldarriaga Molina.
Jefe departamento: Diego José Luis Botia Valderrama.

Índice

Lista de tablas

Lista de figuras

Siglas, acrónimos y abreviaturas

SEPLN	La Sociedad Española para el Procesamiento del Lenguaje Natural
TASS	Taller de Análisis Semántico en la SEPLN

Resumen

El presente trabajo tiene como finalidad poder realizar un análisis de sentimientos en tweets en español. En una primera instancia se ha de tener en cuenta un tratamiento de los datos que involucran una normalización del lenguaje donde se eliminaran palabras vacías (stopwords), emojis, menciones entre otros. Después de estos se evaluaran diferentes métricas de desempeño de clasificación en cada uno de los modelos planteados, teniendo en cuenta diferentes formas de vectorización de los conjuntos de datos y también diferentes distribuciones de los mismos, esto con el fin de comparar primero la eficacia de cada modelo y además si influye o no la porción de datos que se utilice para entrenar y testear y además la forma de representar los mismos. Los mejores modelos que se encontraron fueron una regresión logística con una representación de los datos dada por un embebimiento y un clasificador de stacking con una representación de los datos dada por una vectorización Count Vectorizer.

Palabras clave: análisis de sentimientos, tweets, embeddings, clasificación.

Abstract

The purpose of this paper is to perform an analysis of sentiments in tweets in Spanish. In the first instance, a treatment of the data that involves a normalization of the language must be taken into account where stopword, emojis, mentions, among others, will be eliminated. After these, different classification performance metrics will be evaluated in each of the proposed models, taking into account different forms of vectorization of the data sets and also different distributions of the same, this in order to first compare the effectiveness of each model and also if it influences or not the portion of data that is used to train and test and also the way of representing them. The best models found were a logistic regression with a representation of the data given by an embedding and a stacking classifier with a representation of the data given by a Count Vectorizer.

Keywords: sentiment analysis, tweets, embeddings, classification.

Introducción

El análisis de sentimientos es una técnica del procesamiento de lenguaje natural que busca determinar una emoción en un texto escrito. Esta técnica en los ámbitos empresariales pueden dar una retroalimentación sobre que opinan los consumidores de una marca y/o producto y así entender las necesidades de sus clientes. Aunque realmente se puedan tener diversas categorías para clasificar estos sentimientos para esta monografía se tendrán en cuenta las siguientes 3:

- Positivo
- Neutro
- Negativo

Con la expansión de las redes sociales y particularmente de Twitter, se encuentra una gran fuente de información para que una organización pueda obtener la opinión de un tema en especifico, dandole valor a esta tecnica. Algunas de las aplicaciones que se pueden tener con el análisis de sentimientos son:

- Monitorear las redes sociales.
- Monitorear el servicio al cliente.
- Exploración del mercado.

Esto pues permite entender a nivel de negocio y/o compañía que tan aceptado es o no un producto y/o servicio. Además de ayudar a medir que tan contentos están las personas y que tantas nuevas recomendaciones se pueden tener.

Se ha de tener en cuenta además de los desafíos que implica, llevar a cabo un algoritmo pues el lenguaje humano tiene muchos matices y se dan en un contexto determinado que no es posible enseñarle completamente a la maquina.

1 Planteamiento del problema

Haciendo uso de la base de datos que provee La Sociedad Española para el Procesamiento del Lenguaje Natural (SEPLN) en sus talleres de análisis semántico (TASS) se busca hacer un análisis de sentimientos de los diferentes tweets clasificándolos en los siguientes tres estados:

- Positivo
- Neutro
- Negativo

Primero se hará un análisis exploratorio de las características de la base de datos, y transformaciones en búsqueda de obtener un conjunto de datos de entrenamiento y validación con estructuras unificadas. Posteriormente se explorarán diversas arquitecturas de redes neuronales y algoritmos de Machine Learning como, por ejemplo:

- Regresión logística
- SVM
- Naive Bayes
- Métodos emsambladores tales como: árboles de decisión, voto mayoritarío, XG boosting, Stacking, HisGradiantBoosting
- Red neuronal LSTM

1.1 Problema de negocio

Las opiniones son muy importantes para las organizaciones y gobiernos ya que estos siempre buscan conocer la percepción de los consumidores en un servicio o producto dado. Anteriormente cuando se quería conocer por ejemplo que tan bueno era cierto producto las personas preguntaban a sus amigos o familia, esta dinámica ha cambiado puesto que ya no es necesario preguntar a una persona conocida, sino que mediante las redes sociales se puede

obtener información del producto y finalmente decidir si se adquiere o no. (Liu, 2020)
.

Esto plantea entonces la necesidad de las organizaciones de entender que opinan sus grupos de interés y tomar decisiones para mejorar su experiencia de servicios con los usuarios.

Aunque actualmente se encuentran servicios en línea para la clasificación de texto, estos generalmente se encuentran en inglés. En esta monografía se plantea un acercamiento desde el idioma español.

1.2 Aproximación desde la analítica de datos

Por medio de un modelo de clasificación se plantea clasificar los sentimientos de los los tweets de las personas en dicha plataforma. Se debe tener en cuenta que al ser frases, estas no pueden ser adicionadas al modelo como una variable sino que se debe tener una representación vectorial de las mismas. Además, antes de tener esta representación se debe tener en cuenta un pre-procesamiento para que el texto disminuya la mayor cantidad de ruido, eliminando el contenido no relevante para la tarea de clasificación. En el caso particular de esta monografía las representaciones de estos tweets se harán mediante las librería Scikit Learn y un embebimiento de palabras con la librería TensorFlow Hub. Como resultado se obtienen una lista donde se clasifica los diferentes tweets en su estado correspondiente.

1.3 Origen de los datos

Los datos para el entrenamiento y testeo se encuentran alojados en la página del TASS [1]; estos se encuentran disponibilizados como archivos tsv de diferentes países hispanohablantes tanto para el entrenamiento como para el testeo.

[1] http://tass.sepln.org/2020/

2 Objetivos

2.1 Objetivo general

El objetivo principal de esta monografía es evaluar el rendimiento de diferentes algoritmos de clasificación de aprendizaje automático como Naive Bayes, y algunos métodos ensambladores además de el desempeño de una red LSTM en un conjuntos de datos de Twitter.

2.2 Objetivos específicos

1. Identificar los algoritmos y las métricas para evaluar el desempeño de los clasificadores de aprendizaje automático.
2. Comparar las métricas obtenidas de diferentes algoritmos de aprendizaje automático en función de la representación del conjunto de datos.
3. Identificar cual algoritmo es el más adecuado para el análisis de sentimientos en español.

3 Marco teórico

En el desarrollo de los algoritmos para los diferentes modelos se utilizaron dos métodos de tokenización que provee la librería Scikit Learn. Alguna terminología antes de definir estos métodos.

- **Frecuencia de términos** : En un documento d, la frecuencia representa el número de veces que ocurre una palabra t. Da a las palabras una ponderación binaria o utiliza las ponderaciones de las frecuencias para indicar la importancia relativa que tiene una palabra.. Generalmente se tiene que el se divide la frecuencia de la palabra por la cantidad total de palabras en el documento.
- **Frecuencia del documento**: es el número de veces que las características aparecen en todos los textos.(Mejova, 2021)

CountVectorizer: La representación más básica de un documento se basa en una codificación binaria del texto . En otras palabras, todo texto se representa como un vector de 0 y un solo 1 en la posición que le da el índice de las palabras del texto. La longitud del vector corresponde al tamaño del diccionario.

Basado en una representación de bolsa de palabras, un párrafo o documento completo podría codificarse como un vector de posiciones de una cantidad determinada de palabras, donde la posición i da cuenta del número de veces que la palabra i apareció en el texto. Por lo general, dicho vector se normaliza con respecto al número de palabras en el texto, esto se denomina representación término-frecuencia.(Pang, 2008)

Frecuencia de término-Frecuencia de documento inversa (TF-IDF). Se puede definir como el cálculo de cuan relevante es una palabra en un corpus. El significado aumenta proporcionalmente el número de veces que aparece la palabra en un texto. Pero se compensa con el tamaño del corpus.

Se define ahora lo que significa un embebimiento, es una asignación de una variable categórica discreta a un vector de números continuos. En el contexto de las redes neuronales, las incrustaciones son representaciones vectoriales continuas aprendidas de baja dimensión de variables discretas. Las incrustaciones de redes neuronales son útiles porque pueden reducir la dimensionalidad de las variables categóricas y representar categorías de manera significativa en el espacio transformado.

3.1 Modelos

En el desarrollo de esta monografía se exploraron diferentes algoritmos de clasificación y una red neuronal por lo cual se hará una breve explicación sobre estos algoritmos.

Regresión Logística estima la probabilidad de que ocurra un evento, como votar o no votar, en función de un conjunto de datos determinado de variables independientes. Dado que el resultado es una probabilidad, la variable dependiente está limitada entre 0 y 1. En la regresión logística, se aplica una transformación logit a las probabilidades, es decir, la probabilidad de éxito dividida por la probabilidad de fracaso. La transformación logit viene dada por la siguiente formula:

$$logit(p_i) = \ln\left(\frac{p_i}{1-p_i}\right) \tag{1}$$

Donde p_i representa la probabilidad de éxito.

Maquina de Soporte Vectorial Dado un conjunto de puntos, subconjunto de un espacio mayor, en el que cada uno de ellos pertenece a una de dos posibles categorías, un algoritmo basado en SVM construye un modelo capaz de predecir si un punto nuevo pertenece a una categoría o a la otra. La SVM busca un hiperplano que separe de forma óptima a los puntos de una clase de la de otra, esto es que el hiperplano que tenga el mayor margen con los puntos que estén más cerca de él mismo. De esta forma, los puntos del vector que son etiquetados con una categoría estarán a un lado del hiperplano y los casos que se encuentren en la otra categoría estarán al otro lado.

Clasificación ingenua bayesiana Es un clasificador probabilístico basado en el teorema de bayes donde se supone que cada una de las características siguen una distribución bayesiana.

Métodos de ensamble Los métodos de ensamble son algoritmos que usan la combinación de diferentes modelos para obtener un mejor modelo. Algunos de ellos solo pueden estar compuestos de un solo tipo de clasificador como es el caso de los bosques aleatorios y del aumento del gradiente que se basan en árboles de decisión y en otros casos podemos tener diferentes modelos para la construcción del método de ensamble como es el caso de los algoritmos HistGrandianBoosting, Voting Classifier y Stacking Classifier. Se entenderá la técnica de Boosting para los métodos ensambladores como una técnica que funciona combinando varios predictores con poca precisión y combinándolos para obtener un algoritmo con

mayor precisión aprendiendo de los errores algoritmo anterior.

1. **Bosques aleatorios (Ramdon Forest)** Los bosques aleatorios son una combinación de árboles aleatorios donde el resultado para una clasificación es la categoría que la mayoría de los arboles decidieron. Cada uno de estos árboles se escogen de manera aleatoria entre las características del conjunto de datos.

2. **Grandiant Boosting** Este algoritmo construye un modelo aditivo en una forma avanzada por etapas; permite la optimización de las funciones de pérdida diferenciables arbitrarias.

3. **HistGrandianBoosting:** Es un clasificador como el gradiant boosting pero que funciona mas rápido con volúmenes de datos grandes y además que soporta valores nulos.

4. **Votación Mayoritaria:** Es un método que permite combinar diferentes algoritmos de clasificación a partir de una asignación de pesos a cada modelo que genere el algoritmo. Lo anterior permite crear un metaclasificador que mitigue los errores de clasificación y de esta manera, se obtiene un modelo generalizado que compense los errores individuales de cada modelo de clasificación.

5. **Stacking Classfier:** En este se realiza un proceso de aprendizaje con un conjunto de modelos en paralelo tal que se consideran como clasificadores débiles. Luego, estos clasificadores débiles son combinados en un meta-aprendizaje, que combina todos los clasificadores para obtener un modelo generalizado a partir de un algoritmo de clasificación base. Para lograr una diversidad en el modelo generalizado, se determina una media ponderada por cada clasificador débil para mejorar el desempeño del meta-aprendizaje.

Redes neuronales LSTM Las redes neuronales recurrentes (RNN) se adaptan al procesamiento de datos secuenciales. Una RNN procesa una secuencia de datos procesando cada elemento de la secuencia uno a la vez. Una red RNN solo tiene una única capa oculta, pero también tiene un búfer de memoria que almacena la salida de esta capa oculta para una entrada y la retroalimenta a la capa oculta junto con la siguiente entrada de la secuencia. Este flujo recurrente de información significa que la red procesa cada entrada dentro del contexto generado al procesar la entrada anterior, que a su vez fue procesada en el contexto de la entrada anterior. De esta forma, la información que fluye a través del ciclo recurrente codifica información contextual de (potencialmente) todas las entradas precedentes en la secuencia. Esto permite que la red mantenga una memoria de lo que ha visto previamente

en la secuencia para ayudarlo a decidir qué hacer con la entrada actual. La profundidad de un RNN surge del hecho de que el vector de memoria se propaga y evoluciona a través de cada entrada en la secuencia; como resultado, una red RNN se considera tan profunda como larga es una secuencia.

Aunque los RNN pueden procesar una secuencia de entradas, luchan con el problema de los gradientes que se desvanecen. Esto se debe a que entrenar un RNN para procesar una secuencia de entradas requiere que el error se propague hacia atrás a lo largo de toda la secuencia. De esta manera las LSTM surgen como una solución para el problema del gradiente descendiente, al eliminar la multiplicación repetida por el mismo vector de peso durante la retropropagación en un RNN. (Kelleher, 2019)

3.2 Métricas

Al ser un problema de clasificación se usaran métricas que permitan evaluar el desempeño de los diferentes algoritmos en cada categoría, las métricas son obtenidas mediante la librería Scikit Learn. Idealmente a nivel de negocio es tener por lo menos una exactitud (acurracy) mayor o igual al 70 %, aunque en los históricos de la competencia en el TASS se ha tenido como un máximo el 65 %. Todas las métricas para la clasificación multiclase se derivan de las métricas de clasificación binaria, pero se promedian sobre todas las clases (Müller, 2016) y de esta manera se pueden ver los resultados para los diferentes métodos escogidos.

Exactitud (Accuracy) La precisión nos ayuda a medir cuantos tweets se predijeron correctamente de todos los tweets en el conjunto de datos.

Presición (Presicion Score) La precisión es la relación

$$\frac{t_p}{t_p + f_p} \tag{2}$$

donde t_p es el número de verdaderos positivos y f_p el número de falsos positivos. La precisión es intuitivamente la capacidad del clasificador de no etiquetar como positiva una muestra que es negativa. Esto es nos indica la cantidad de tweets que se predijeron correctamente dada cierta categoría.

Aunque la siguiente no es un métrica usada en los modelos es importante tener en cuenta lo que significa.

Recuerdo (Recall) es la relación

$$\frac{t_p}{t_p + f_n} \tag{3}$$

donde t_p es el número de verdaderos positivos y f_n el número de falsos negativos. El recuerdo es intuitivamente la capacidad del clasificador para encontrar todas las muestras positivas.

Medida F1(F1-Score) La puntuación F1 se puede interpretar como una media armónica de precisión y recuperación, donde una puntuación F1 alcanza su mejor valor en 1 y su peor puntuación en 0. La contribución relativa de precisión y recuerdo a la puntuación F1 es igual. La fórmula para la puntuación F1 es

$$2 * \frac{precision * recall}{precision + recall} \tag{4}$$

Exactituda balanceada (Balanced accuracy) Se define como el promedio de recuerdo obtenido en cada clase.

Curva ROC AUC (ROC AUC Score) La puntuación ROC AUC nos dice qué tan eficiente es el modelo.Donde ROC es una curva de probabilidad y AUC representa el grado o medida de separabilidad. Indica cuánto es capaz el modelo de distinguir entre clases.

4 Metodología

4.1 Herramientas

Se utiliza python en su versión 3.9 mediante el editor Visual Studio Code con implementaciones basadas en las siguientes librerías:

- Pandas: es una herramienta de manipulación y análisis de datos de código abierto rápida, potente, flexible y fácil de usar, construida sobre el lenguaje de programación Python. (McKinney, 2022)
- Numpy: abreviatura de Numerical Python, es el paquete fundamental para la computación científica en Python.

- Matplolib: es una librería de Python especializada en la creación de gráficos en dos dimensiones.
- Seaborn: es una biblioteca de visualización de datos de Python basada en matplotlib. Proporciona una interfaz de alto nivel para dibujar gráficos estadísticos atractivos e informativos.
- Plotly: La librería Plotly crea gráficos interactivos con calidad de publicación.
- NLTK: Es un paquete de Python que funciona con datos de lenguaje humano y proporciona una interfaz fácil de usar para diferentes recursos léxicos como WordNet y bibliotecas de procesamiento de texto. Estos recursos léxicos se utilizan para clasificación, tokenización, derivación, etiquetado, análisis y razonamiento semántico. (Bird, 2009)
- Scikit Learn: es un módulo de Python que integra una amplia gama de algoritmos de aprendizaje automático de última generación para problemas supervisados y no supervisados de mediana escala.
- TensorFlow: es una plataforma de código abierto integral para el aprendizaje automático. Tiene un ecosistema completo y flexible de herramientas, bibliotecas y recursos comunitarios que permite a los investigadores impulsar lo último en ML y a los desarrolladores crear e implementar fácilmente aplicaciones basadas en ML.
- Tensonflow Hub: En particular, proporciona modelos guardados preentrenados que se pueden reutilizar para resolver nuevas tareas con menos tiempo de entrenamiento y menos datos de entrenamiento.
- H20: Es una plataforma de aprendizaje automático y análisis predictivo de código abierto, en memoria, distribuida, rápida y escalable que permite crear modelos de aprendizaje automático en big data y proporciona una producción sencilla de esos modelos en un entorno empresarial.

4.2 Datos Originales

Los datos originales se pueden obtener en la página del TASS [2], estos datos son públicos pero para poder hacer la descarga de los mismos se debe firmar la licencia para el uso de estos [3]. Los datos vienen separado en entrenamiento y testeo por los siguientes países

[2] http://tass.sepln.org/2020/
[3] http://tass.sepln.org/tass_data/download.php

ES - España, PE-Perú, CR-Costa Rica, CH-Chile, UR-Uruguay, MX-México. En este caso cada uno de los datos de entrenamiento se componen de tres columnas. De esta manera se pueden descargar 3 carpetas que son, entrenamiento que tiene los datos de entrenamiento; testeo que tiene el contenido de los tweets para hacer las validaciones; además de una carpeta de testeo gold que contiene la categoría final de los datos de testeo. Los archivos dentro de estas carpetas son archivos en formato .tsv, con la siguiente estructura:

- Id Tweet: que hace referencia al Id del tweet publicado
- Tweet: que hace referencia al contenido del tweet
- Classification: que hace referencia de la polaridad de los datos

4.3 Analítica Descriptiva

El conjunto de datos de entrenamiento contiene 7243 registros y el conjunto de pruebas contiene 7264 registros dando un total de 14507 datos. En al figura 1 se muestra la distribución de la polaridad de cada uno de los conjuntos de datos. Se ve entonces que en ambos conjuntos de datos la categoría que mas volumen de datos tiene es la categoría negativa, sin embargo se puede apreciar que: en los datos de entrenamiento se tienen más etiquetes neutras que positivas, mientras que en los datos de evaluación se tienen mas etiquetas positivas que neutras.

Tabla 1

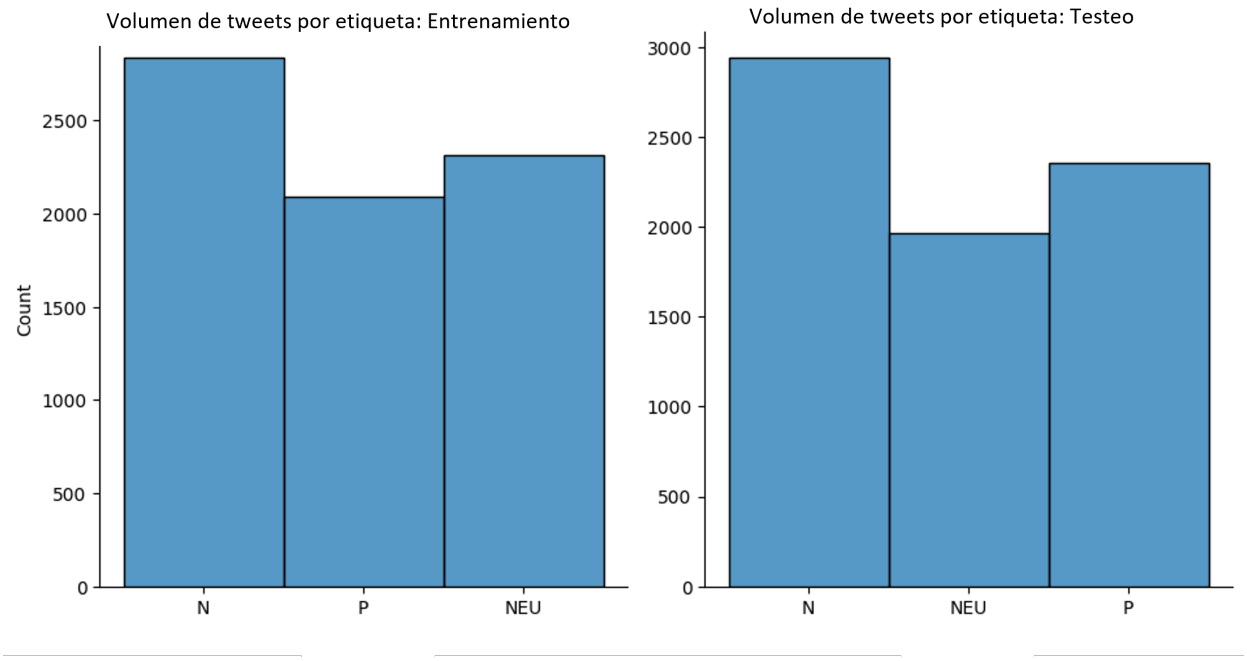
Número Tweets por categoría

Sentimiento	Entrenamiento	Testeo
N	2835	2939
NEU	2315	1966
P	2093	2359

En el conjunto de datos no se tienen datos faltantes o nulos. Ahora bien, se puede ver que los tweets tienen una longitud aproximada de 250 palabras por publicación. También se observa en la figura 2 que las palabras que mas se repiten en un conjunto de datos y el otro cambian. Lo cual puede implicar un reto a la hora de trabajar la clasificación.

Figura 1

Distribución de la datos



4.4 Tubería Principal

En la figura 3 vemos el proceso general de los pasos seguidos en el proceso de analítica

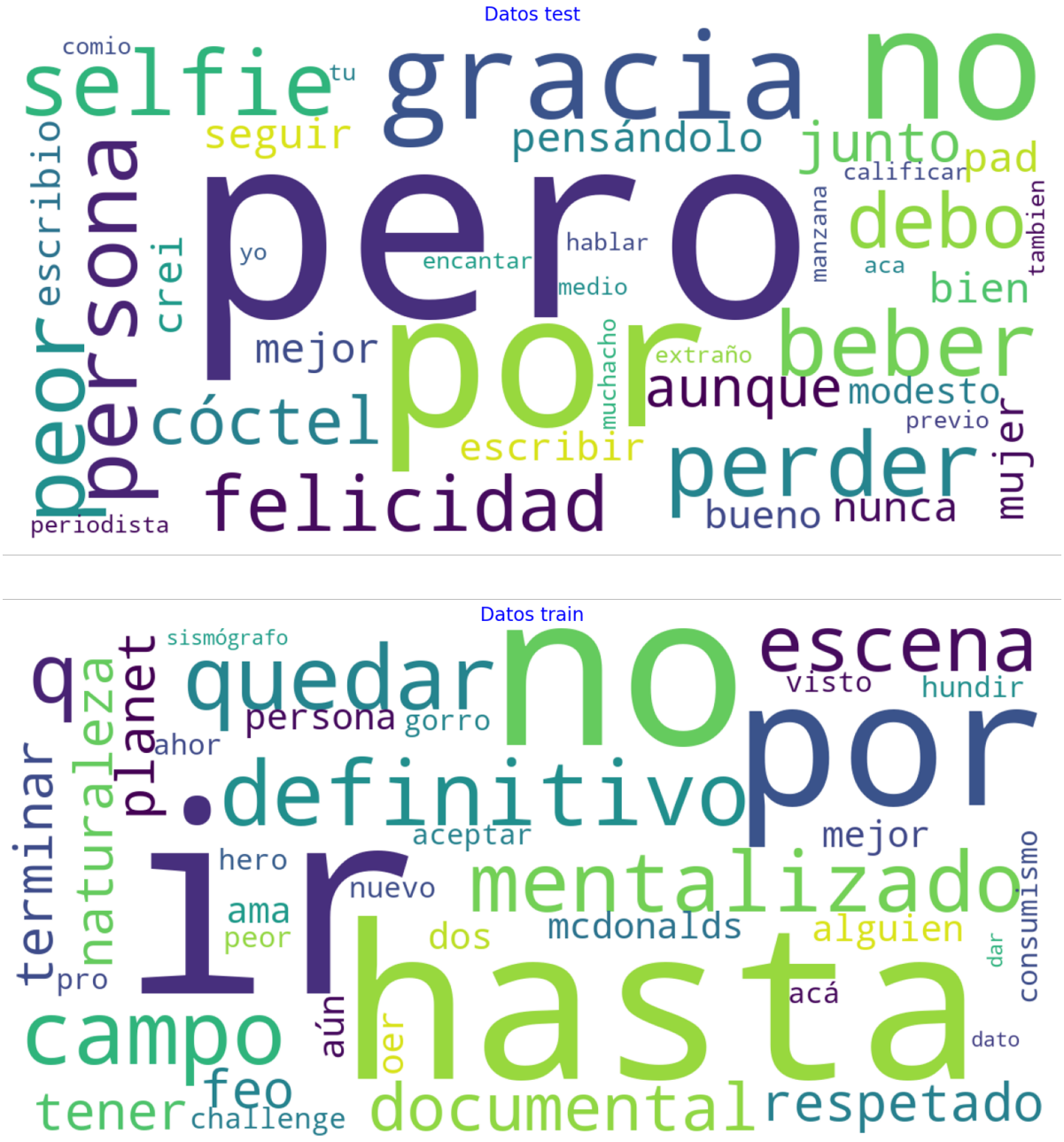
4.4.1 Preprocesamiento

El preprocesamiento consto de las siguientes etapas:

1. Limpieza de los tweets: para la limpieza de los tweets se desarrollaron un conjunto de funciones con las siguientes fines:
   - Remover links
   - Remover espacios de puntuación
   - Remover menciones
   - Remover hashtag
   - Remover símbolos de pesos

Figura 2

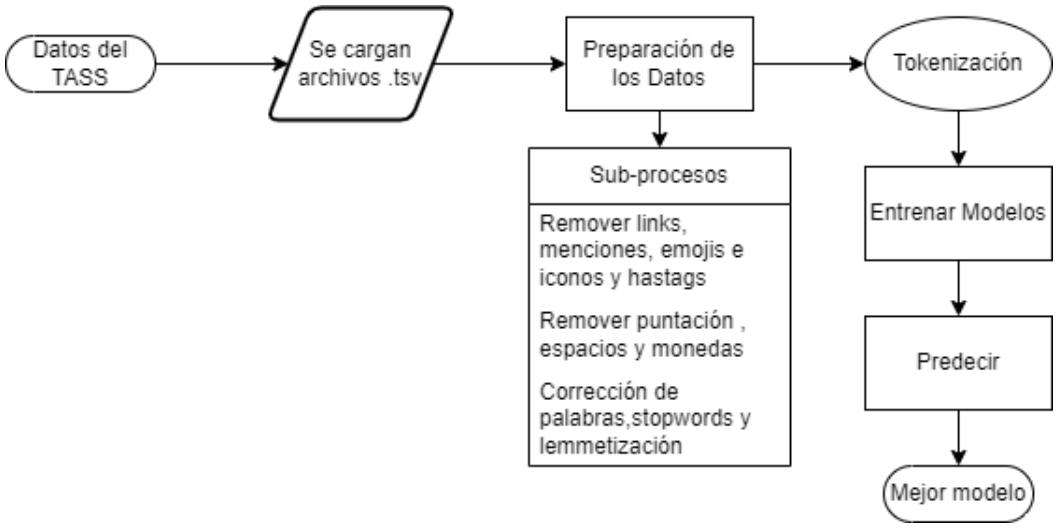
Nubes de palabras



- Transformar iconos
- Separa emojis que vengas juntos
- Normalizar risas
- Remover puntuación

Figura 3

Flujo del proceso



- Remover vocales repetidas
- Corrección de palabras
- Corregir abreviaciones
- Remover stopwords
- Lemetización

2. Procesamiento de texto: Después de la limpieza de los datos se hace un procesamiento del texto para vectorizarlo llamado tokenización.

4.4.2 Modelo base

En la primera iteración se realizan e implementan diferentes funciones para la limpieza del conjunto de datos que nos permiten tener un idioma mas estandarizado además se desarrolla como primer recurso un modelo de regresión logística sin ninguna hiperparametrización con una tokenización dada por el método Count Vectorizer. En la cual no sé tiene muy buenos resultados.

4.4.3 Validación

Los datos provenientes del TASS están separados $50\,\% - 50\,\%$ para entrenamiento y testeo. A lo largo de cada uno de los métodos se evaluaron las métricas buscando la mejor exactitud pero también las demás métricas que nos proporcionan mas información sobre como predice el modelo y que tan bien puede llegar a clasificar cada categoría.

4.4.4 Iteraciones y evolución

Al modelo de regresión logística planteado en la primera iteración se le hace una búsqueda de hiperparametros y seguidamente se implementan los algoritmos de clasificación mas robustos tales como maquinas de soporte vectorial y diferentes métodos ensambladores entre ellos : árboles de decisión, Grandiant Boosting con hiperparemetrizaciones mediante GridSearchCV con 10 pliegues. Después de esto se implementan un otro método de tokenización en este caso un TF-IDF, donde se implementan los mismos algoritmos que con el count vectorizer con sus debidas hiperparametrizaciones. Al no obtener una mejoría se prueba con embebimiento donde se obtiene el mejor resultado parcial para una distribución de los datos $50\,\%$ - $50\,\%$ de entrenamiento y testeo.

También se cambia la proporción de los datos $70\,\%$ para entrenamiento, $30\,\%$ para testeo, en esta ocasión se obtiene que un stacking classifier es la mejor opción pero con el método de vectorización Count Vectorizer.

Seguidamente de esto se implementa una red neural LSTM donde no se obtiene mejores resultados que con los modelos anteriormente mencionados. Como último recursos se emplea un automl con la librería $H2O$ donde tenemos que el mejor modelo es un stacking classifer sin embargo este tampoco presenta mejores resultados para la representación de los datos $50\,\%$ - $50\,\%$; respecto a la proporción $70\,\%$ - $30\,\%$ no fue posible hacer su implementación dada la capacidad computacional que se tiene.

5 Resultados

Modelos 50 - 50

En primera instancia se mostrará una comparación de los modelos con los datos de prueba y testo originales. Donde se observa que el mejor modelo es una regresión logística con un método de vectorización que es el embbeding. Para decir cual es el mejor modelo se usa la medida de la exactitud. En las figuras 4 y 5 se puede apreciar mejor como fue el desempeño de este modelo para los valores macro y los valores individuales de cada categoría.

Tabla 2

Resultados de los modelos con distribución 50 % - 50 %

Modelo	Vectorización	Exactitud	F1	Precisión	Exactitud balanceada	ROC AUC
LogisticRegresion	Embedding	0,60	0,59	0,59	0,58	0,77
VotingClassfier	Embedding	0,60	0,58	0,59	0,58	0,77
StackingClassifier	CountVectorizer	0,59	0,58	0,58	0,58	0,77
StackingClassifer	TfidfVectorizer	0,59	0,58	0,58	0,57	0,77
LinearSVC	Embedding	0,59	0,58	0,59	0,58	0
RegresionLogistica	CountVectorizer	0,59	0,58	0,58	0,57	0,76
LogisticRegression	TfidfVectorizer	0,58	0,56	0,58	0,56	0,76
StackingClassifer	Embedding	0,58	0,57	0,58	0,57	0,75
RandomForest	Embedding	0,58	0,56	0,56	0,56	0,75
VotingClassifier	CountVectorizer	0,58	0,57	0,58	0,57	0,76
RandomForest	CountVectorizer	0,58	0,57	0,58	0,57	0,75
VotingClassifer	TfidfVectorizer	0,58	0,57	0,58	0,57	0
XGBClassifier	Embeding	0,58	0,56	0,57	0,56	0,75
GaussianNB	Embedding	0,57	0,55	0,55	0,56	0,74
RandomForest	TfidfVectorizer	0,57	0,56	0,57	0,56	0,75
HistGradiantBoosting	Embedding	0,57	0,55	0,56	0,55	0,75
GradientBoostingClassifier	CountVectorizer	0,57	0,56	0,58	0,56	0,74
HistGradientBoostingClassifier	CountVectorizer	0,56	0,56	0,56	0,55	0,74
GradientBoosting	Embedding	0,56	0,55	0,55	0,55	0,74
XGBClassifier	CountVectorizer	0,56	0,53	0,56	0,53	0,74
GradientBoosting	TfidfVectorizer	0,54	0,52	0,53	0,52	0,73
XGBClassifier	TfidfVectorizer	0,54	0,51	0,54	0,51	0,72
scvLineal	CountVectorizer	0,51	0,51	0,52	0,51	0
LinearSVC	TfidfVectorizer	0,51	0,51	0,52	0,51	0
HistGradientBoosting	TfidfVectorizer	0,51	0,51	0,52	0,51	0,55
GaussianNB	CountVectorizer	0,43	0,42	0,43	0,43	0,57
GaussianNB	TfidfVectorizer	0,39	0,37	0,40	0,39	0,55

Modelos 70-30

Como se puede observar en la tabla 3, los mejores resultados se ven con el modelo Stac-

Figura 4

Matriz de confusión 50-50

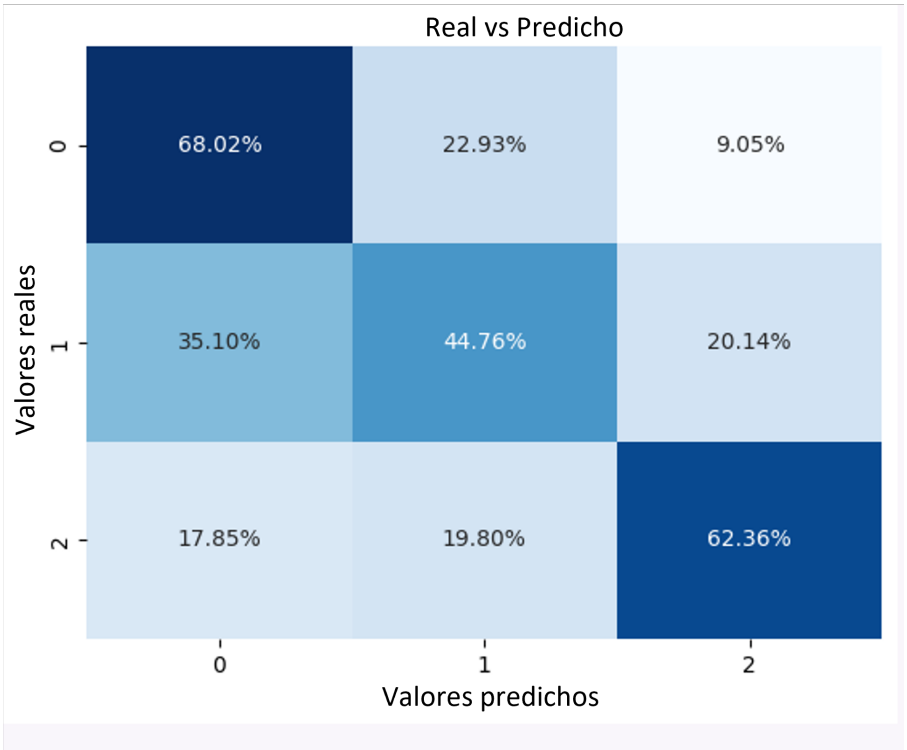


Figura 5

Reporte 50-50

	Precision	Recall	f1-score	support
0	0,64	0,62	0,66	2939
1	0,44	0,45	0,44	1966
2	0,69	0,62	0,65	2359
accurracy			0,60	7264
macro avg	0,59	0,58	0,59	7264
weighted avg	0,60	0,60	0,60	7264

kingClassifer con la vectorización Count Vectorizer, ahondando en estos resultados tenemos la matriz de confusión 6 y el reporte 7, de la matriz de confusión se puede observar que la categoría que peor desempeño tiene es la de los valores neutros.

Redes Neuronales

Tabla 3

Resultados de los modelos con distribución 70 % - 30 %

Modelo	Vectorizacion	Exactitud	F1	Precision	Exactitud balanceada	ROC AUC
StackingClassifier	CountVectorizer	0.60	0.58	0.58	0.58	0.76
VotingClassfier	Embedding	0.60	0.57	0.58	0.57	0.76
regresionLogistica	CountVectorizer	0.59	0.58	0.58	0.58	0.76
VotingClassifier	CountVectorizer	0.59	0.58	0.58	0.57	0.76
StackingClassifer	TfidfVectorizer	0.59	0.57	0.57	0.57	0.76
LogisticRegresion	Embedding	0.59	0.56	0.57	0.56	0.76
LinearSVC	Embedding	0.59	0.57	0.58	0.57	0.00
LogisticRegression	TfidfVectorizer	0.58	0.55	0.57	0.55	0.76
RandomForest	Embedding	0.58	0.55	0.56	0.56	0.75
HistGradiantBoosting	Embedding	0.58	0.56	0.57	0.56	0.74
XGBClassifier	Embeding	0.58	0.55	0.56	0.55	0.75
RandomForest	CountVectorizer	0.57	0.57	0.57	0.57	0.74
GaussianNB	Embedding	0.57	0.56	0.56	0.56	0.74
RandomForest	TfidfVectorizer	0.57	0.57	0.57	0.57	0.74
StackingClassifer	Embedding	0.56	0.54	0.55	0.54	0.74
GradientBoosting	Embedding	0.56	0.55	0.55	0.55	0.73
XGBClassifier	CountVectorizer	0.56	0.51	0.55	0.52	0.74
GradientBoostingClassifier	CountVectorizer	0.56	0.52	0.54	0.53	0.74
HistGradientBoostingClassifier	CountVectorizer	0.56	0.54	0.55	0.54	0.74
GradientBoosting	TfidfVectorizer	0.55	0.53	0.54	0.53	0.73
XGBClassifier	TfidfVectorizer	0.54	0.49	0.55	0.50	0.72
HistGradientBoosting	TfidfVectorizer	0.52	0.52	0.53	0.52	0.55
scvLineal	CountVectorizer	0.52	0.52	0.53	0.52	0.00
LinearSVC	TfidfVectorizer	0.48	0.47	0.48	0.47	0.00
GaussianNB	CountVectorizer	0.44	0.43	0.44	0.44	0.58
GaussianNB	TfidfVectorizer	0.39	0.37	0.40	0.40	0.55

Como se ve en los reportes de clasificación 10 de las redes neuronales se tiene solo un desempeño de 51 % y nuevamente la categoría neutra es la que presenta peores resultados

Auto ML

El mejor desempeño que arroja la librería de AutoML H2O es un stacking clasifier que tiene las métricas evaluadas anteriormente. Donde el mejor desempeño obtuvo un 58 %

Figura 6

Matriz de confusión 70-30

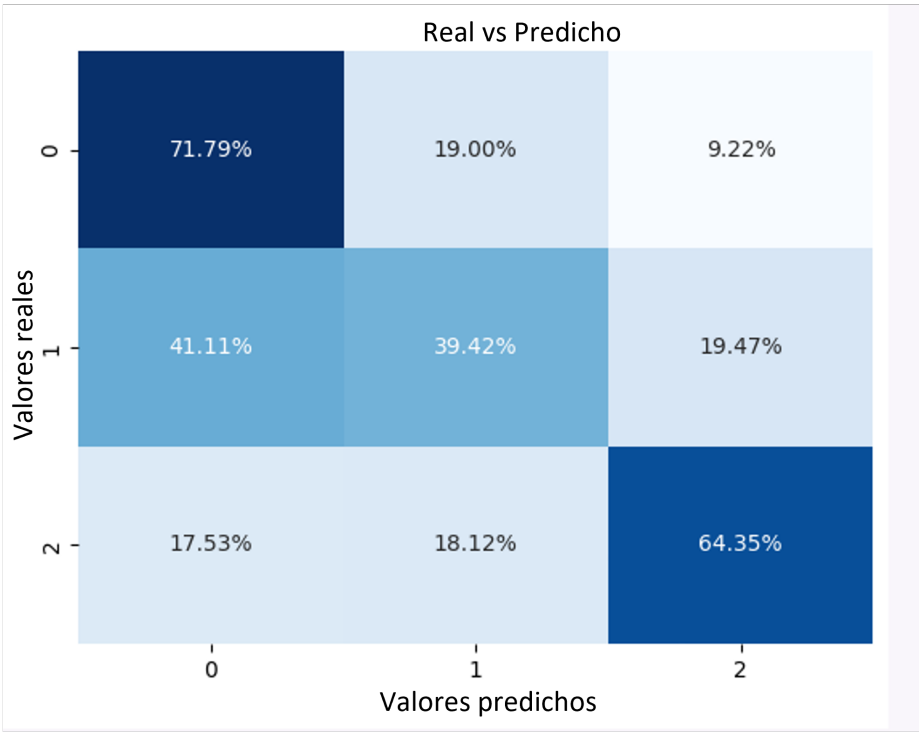


Figura 7

Reporte 70-30

	Precision	Recall	f1-score	support
0	0,63	0,72	0,67	1758
1	0,46	0,39	0,42	1243
2	0,68	0,64	0,66	1352
accurracy			0,60	4353
macro avg	0,59	0,59	0,59	4353
weighted avg	0,60	0,60	0,60	4353

Figura 8

Reporte redes neuronales

Redes Neuronales

50-50

	Precision	Recall	f1-score	support
0	0,56	0,56	0,56	1299
1	0,42	0,41	0,42	1049
2	0,52	0,54	0,53	912
accurracy			0,51	4353
macro avg	0,50	0,50	0,50	4353
weighted avg	0,51	0,51	0,51	4353

70-30

	Precision	Recall	f1-score	support
0	0,57	0,62	0,59	867
1	0,43	0,36	0,39	699
2	0,51	0,55	0,53	607
accurracy			0,51	2173
macro avg	0,50	0,51	0,50	2173
weighted avg	0,51	0,51	0,51	2173

Figura 9

Matriz de confusión AutoML

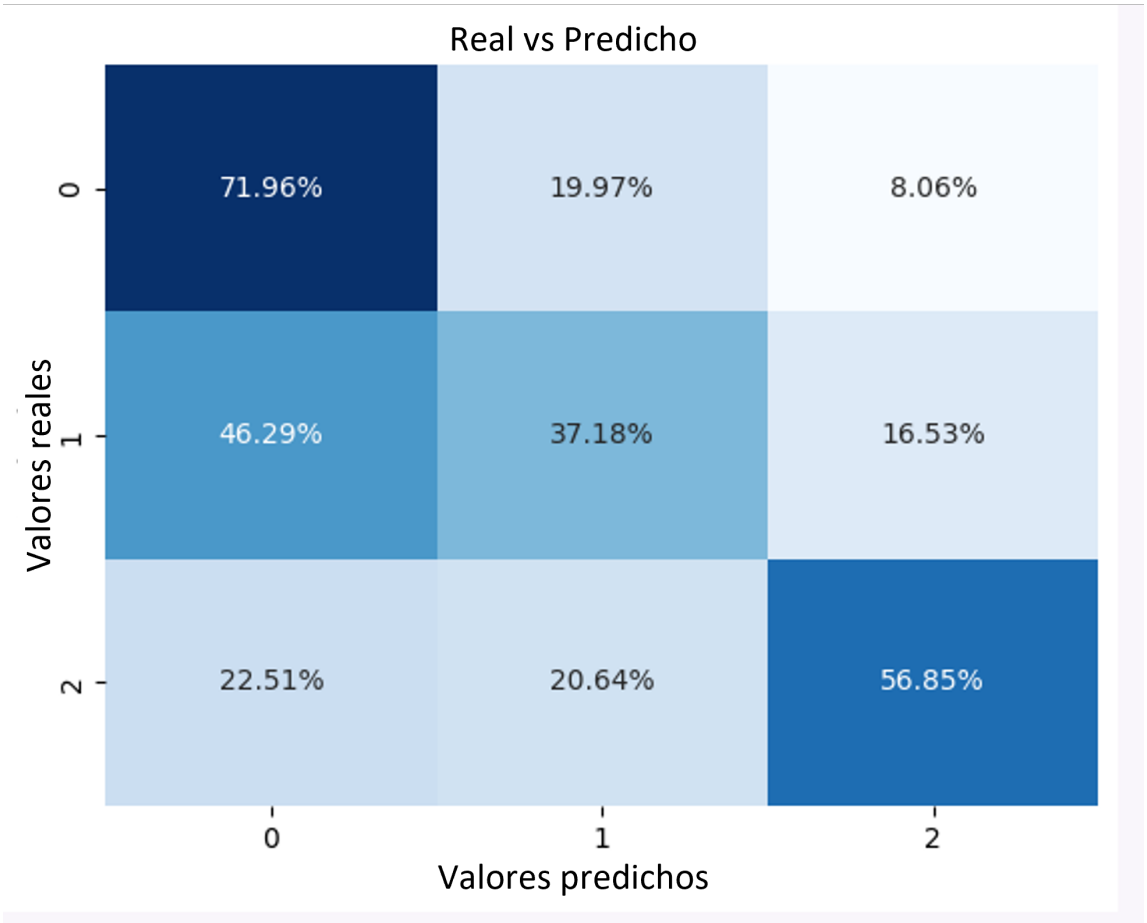


Figura 10

Reporte AutoML

	AutoML			
	Precision	Recall	f1-score	support
0	0,59	0,72	0,65	2939
1	0,40	0,37	0,39	1966
2	0,70	0,57	0,63	2359
accurracy			0,58	7264
macro avg	0,57	0,55	0,56	7264
weighted avg	0,58	0,58	0,57	7264

6 Discusión

Dada la cantidad de técnicas utilizadas y los diferentes métodos de vectorización usados se puede suponer que los datos para el testeo o específicamente para la clase neutra se encuentran muy desbalanceados con respecto a los demás. Esto pues, en todos los métodos implementados se tiene que la categoría neutra presenta un mal desempeño en los diferentes reportes y matrices de confusión. Se podría buscar otros conjuntos de datos etiquetados con las variables positivo, neutro y negativo y así tener un conjunto de datos más robusto que permita implementar nuevamente los algoritmos y evaluar como cambia el desempeño de los mismos. O intentar métodos de balanceo de data aumentation en esta categoría para mejorar el rendimiento. Además de esto se podría también pensar en buscar otra manera de representar los datos un poco más moderna que las utilizadas y diferente a los embebimientos para evaluar en ese caso como fue el desempeño de los modelos.

Una alternativa para evaluar el desempeño de los diferentes algoritmos en las clases positivas y negativas y ver estos que tanto se ajustan los modelos a estos datos es tomar únicamente estas dos categorías y evaluar cada uno de los modelos implementados y realizar las comparaciones con los resultados ya obtenidos, y de esta manera ver si se ajusta bien o no a los datos, o por el contrario los modelos planteados no se ajustan tanto a los datos y se debe buscar quizás otros modelos más robustos como algoritmos de machine learning profundo que involucren arquitecturas tranformers que han presentado un buen desempeño en cuanto a la clasificación de texto.

Algunas de la dificultades presentadas durante la elaboración de esta monografía fueron el coste computacional de algunos métodos ensambladores y de las hiperparametrizaciones pues las vectorizaciones de los datos constaban de dataframes con muchas columnas llevando a que algunas búsquedas de parámetros tomaran entre 4 y 5 horas y algunos métodos ensambladores entre 2 y 3. También a la hora de intentar implementar automl con la distribución de los datos 50 % - 50 % se tuvo un costo computacional de aproximadamente 12 horas y para la distribución 70 % - 30 % no fue posible puesto la maquina se quedaba sin memoria después de pasadas unas horas, podría intentarse implementarse este modelo con un equipo de computo con mayor memoria y evaluar en este caso los resultados presentados.

A la hora de llevar a producción este modelo se debe tener en cuenta el método de sustracción de la información del nuevos datos y la manera de etiquetarlos. Además de tener en cuenta que nuevas formas de corrección de la escritura se pueden tener y que librerías o métodos dejan de estar habilitados en las diferentes librerías de tratamiento de texto.

Los diferentes notebooks donde se desarrollaron estos experimentos se pueden encontrar en el siguiente repositorio de github: https://github.com/DanielaAguiV/Analsis_de_sentimientos_espa-ol

7 Conclusiones

1. Aunque se tuviese el mismo conjunto de datos de entrenamiento y testeo originales iguales se evidencia que al utilizar diferentes formas de vectorizar el contenido de los tweets se tienen cambios en el desempeño de los modelos planteados. De esta manera también se debe tener presente a la hora de trabajar con texto que la representación de los datos juega un papel determinante y que se debe buscar también la mejor manera hacerlo.

2. La distribución de los datos también influye en el desempeño de los algoritmos pues como se evidencia en las tablas 2 y 3, se tiene que el mejor desempeño en ambas distribuciones cambian significativamente tanto en el modelo escogido dado que el primero fue una regresión logística y en el segundo un stacking, como el método de vectorización de los datos dado que en el caso de la distribución 50 % - 50 % fue un embebimiento mientras que en la distribución 70 % - 30 % fue dada por el count vectorizer.

3. La clase neutra es la clase donde todos los algoritmos presentaron un bajo desempeño para poder clasificarla. Esto se puede apreciar en los reportes proporcionados por Scikit Learn y las matrices de confusión pues el cantidad de valores acertados en las matrices de la representación 50 % - 50 % fue del 44 % mientras que las categorías positiva y negativa superaban el 60 %. Para el caso de la distribución 70 % - 30 % se evidencia en la matriz de confusión que la categoría neutro esta por debajo del 40 % mientras las otras dos categorías tuvieron un desempeño supero al 64 %. Este comportamiento se generalizo a través de los diferentes algoritmos.

4. Dado el desempeño particularmente del algoritmo Naive Bayes Gaussiano y que en ambas distribuciones dos de sus tres representaciones están ranqueadas en el último lugar se puede decir que este algoritmo no se ajusta para el problema que se plantea solucionar en esta monografía.

5. Las mejores clasificaciones que se tienen, tomando como medida principal la exactitud presentan un porcentaje de acierto global del 60 % y las peores de un 39 % lo que muestra que aunque no se logro tener un algoritmo que sobrepasara la métrica de negocio deseada si se construyo el mejor modelo posible.

6. En el caso de los mejores modelos se toma como el mejor modelo el de la representación 50 % - 50 % puesto que el tiempo computacional de la ejecución de este es corto, si bien su hiperparametrización duro 4 horas de computo, en el caso del staking cada ejecución

del algoritmo consto de aproximadamente 3 horas. Lo que en términos de producción es preferible puesto que en modelo de regresión logística podría ejecutarse más veces que el de stacking o se podría re-entrenar con mayor facilidad cada vez.

Referencias

Bird, S. Klein, E. . L. E. (2009). *Natural Language Processing with Python*. The MIT Press Essential Knowledge series. O'Reilly Media.

Kelleher, J. (2019). *Deep Learning*. The MIT Press Essential Knowledge series. MIT Press.

Liu, B. (2020). *Sentiment Analysis: Mining Opinions, Sentiments, and Emotions*. Studies in Natural Language Processing. Cambridge University Press.

McKinney, W. (2022). *Python for Data Analysis*. O'Reilly Media.

Mejova, Y. Srinivasan, P. (2021). *Proceedings of the International AAAI Conference on Web and Social Media*. *5*(1), 546–549*.

Müller, A.C. Guido, S. (2016). *Introduction to Machine Learning with Python: A Guide for Data Scientists*. O'Reilly Media.

Pang, B. Lee, L. (2008). *Opinion Mining and Sentiment Analysis*. Foundations and trends in information retrieval. Now Publishers.