Proyecto 1: Minería de Texto: Clasificación de Tweets respecto a opiniones del COVID

Juan Daniel Castrellon - 201729285

Marzo 2021

Índice

1. Introducción

Actualmente, el nivel de desinformación se ha disparado debido al auge de las redes sociales. A pesar de que esto se ha estado evidenciando desde la última década, no cabe duda alguna de que la situación del COVID-19 y vacunación ha mostrado como esta desinformación puede ser perjudicial para la sociedad. Aún así, respecto a la desinformación aún no existe un algoritmo que sea capaz de clasificar con mucha certeza la veracidad de un tema como lo es la vacunación.
De esta forma, sería interesante comprender mejor el punto de vista de los ciudadanos. Aún más, sería interesante para los gobiernos o las grandes redes sociales generar un algoritmo que, dada una entrada de texto este pueda estimar su punto de vista respecto a la vacunación. Es así como fue posible conseguir el conjunto de datos **Coronavirus tweets NLP** [**?**], donde se ven un conjunto de *Tweets* y se tiene anotada su posición respecto a la vacunación y al COVID. Con esto en mente, se creó un proyecto en Python que cumpliera con el objetivo propuesto. A continuación, se muestra el procedimiento seguido para la selección del algoritmo y los parámetros usados, además de los resultados obtenidos por el mismo.

2. Selección del algoritmo

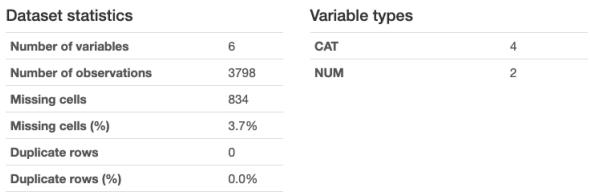
Para poder seleccionar el algoritmo toca, en primer lugar, conocer la oportunidad o problema del negocio. Como ya fue mencionado, este consiste en la creciente desinformación y publicación de textos falsos respecto al COVID-19 y la vacunación. De esta forma, se entiende que lo mejor es crear un algoritmo de clasificación, el cual dada una entrada de texto indique si la posición de la persona es positiva, negativa o neutra, respecto a la vacunación. Asimismo, sería interesante conocer que palabras tienen una mayor incidencia en una opinión negativa respecto al tema, con el fin de entender el punto de vista de esta población desinformada.
Por lo tanto, se concluye que el algoritmo más útil corresponde a la creación de un árbol de decisión. Este no permite únicamente la clasificación del texto que se pasa por entrada. También es posible, al estudiar los nodos, entender que palabras tienen una mayor incidencia en comentarios negativos respecto al COVID y vacunación. Para poder conocer cuales son los mejores parámetros que puede tener el árbol es necesario conocer mejor los datos, además de realizar la respectiva preparación del texto, con el fin de que este sea comprendido por el algoritmo.

3. Perfilamiento de los datos

En primer lugar, es necesario entender los datos que se tienen. Como ya fue mencionado, este consiste en un conjunto de Tweets y su respectiva opinión. Aún así, se tienen en total 6 atributos, los cuales corresponden a

- **UserName**: Usuario que publicó el Tweet
- **ScreenName**: Corresponde al mismo ID del usuario que publicó el Tweet
- **Location**: Ubicación de donde se publicó del Tweet
- **TweetAt**: Fecha en que se publicó el Tweet
- **OriginalTweer**: El texto que se encontraba en el Tweet publicado
- **Sentiment**: El sentimiento que tiene el Tweet (*Extremadamente negativo, negativo, neutro, positivo, extremadamente positivo*)

El perfilamiento fue realizado haciendo uso de la librería `pandas_profiling`. El resumen obtenido de los datos fue el siguiente:



Dataset statistics	
Number of variables	6
Number of observations	3798
Missing cells	834
Missing cells (%)	3.7%
Duplicate rows	0
Duplicate rows (%)	0.0%

Variable types	
CAT	4
NUM	2

Figura 1: Resumen del perfilamiento realizado con **pandas_profiling**

Como se ve, el conjunto de datos consiste de 6 variables, 4 categóricas y 2 numéricas. El porcentaje de datos faltantes es del 3,7 %. A pesar de tener 6 columnas, solamente son necesarias el sentimiento que tiene el Tweet y el texto escrito. Por lo tanto, es posible eliminar las otras 4 columnas (**Username**, **ScreenName**, **Location** y **TweetAt**). Ahora, en vista de que se eliminaron las columnas que contienen vacíos, no es necesario tomar ninguna decisión al respecto.
Por último, cabe resaltar que en total se tienen 5 clases de sentimiento. Aún así, no todas son necesarias, puesto que *Extremadamente negativo* y *negativo* pueden corresponder a una misma clase. Por su lado *Extremadamente positivo* y *positivo* también lo hacen. De esta forma, es posible reducir el problema a únicamente 3 clases *negativo*, *neutro* y *positivo*.

4. Preparación del texto

Una vez se realizó el perfilamiento y limpieza de los datos, es necesario preparar el texto con el fin de que este sea comprendido por el algoritmo. En este caso, la forma de codificar el texto elegida fue el **Bag Of Words**, donde se generan N atributos nuevos al dato, donde N es el número total de palabras en el Dataset. En cada celda se pone el número de veces que esta palabra aparece en el texto. Aún así, en vista de que el número de palabras es tan grande, necesario realizar una preparación previa.
De esta forma, se siguieron los siguientes pasos para la limpieza del texto.

1. **Eliminar los signos de puntuación**: Esto se logró haciendo uso de expresiones regulares, con la librería `re` de `Python`. En este caso, para cada frase, se convertía los signos de puntuación en un caracter espacio .

2. **Volver las letras mayúsculas en minúsculas**: Esto se logró mediante el método `lower()` de la clase `String` de `Python`. De esta forma, dos palabras iguales, pero que se encuentra una en mayúsculas y otra en minúsculas pasan a tener la misma codificación.

3. **Convertir la frase en un arreglo de palabras**: Esto fue posible usando la librería `nltk.tokenize`. De esta forma, se puede analizar cada una de las palabras por aparte.

4. **Eliminar las *STOPWORDS***: Estas corresponden a palabras que no otorgan información al problema. Por ejemplo, preposiciones, pronombres o artículos. La librería `nltk` contiene un arreglo de *Stop words*. Por lo tanto, es posible usar a este con el fin de reconocer si una palabra es o no informativa

5. **Lematización**: Ahora bien, existen palabras que pueden significar lo mismo, o variaciones de palabras por sufijos o prefijos. Por lo tanto, para reducir la dimensionalidad, se puede hacer uso de un algoritmo de `Steeming`. En este caso, se usó la librería `nltk.stem.porter`, de la cual se usó la clase `PorterStemmer`, la cual realiza este trabajo.

6. **Corrección de ortografía**: Ahora, en visto de que el texto a tratar son Tweets, estos pueden venir con errores de ortografía, puesto que quienes lo escriben lo hacen en lenguaje coloquial. De esta forma, se usó la librería `autocorrect`, con el fin de solucionar este problema.

7. **Conservar solo las palabras que existen**: Aún así, a pesar de que se realizó una limpieza buena de los datos, siguen existiendo palabras que no tienen ningún sentido en el ingles. Por lo tanto, se usaron dos bancos de palabras de `nltk`. Uno representa la totalidad de las palabras del idioma ingles. La segunda, representa un conjunto de palabras obtenidas de chats, esto con el fin de acceder a palabras coloquiales que no se encuentran en un diccionario. De esta forma, si una palabra no pertenece a alguno de estos diccionarios es eliminada.

8. **Volver a unir las palabras que quedan**: Por último, se vuelven a unir las palabras que quedan. De esta forma, se consigue un arreglo de frases preparadas para el algoritmo **Bag Of Words**

De esta forma, una Tweet pasa de ser *TRENDING: New Yorkers encounter empty supermarket shelves (pictured, Wegmans in Brooklyn), sold-out online grocers (FoodKick, MaxDelivery) as coronavirus-fearing shoppers stock up https://t.co/Gr76pcrLWh https://t.co/ivMKMsqdT1* a ser de la siguiente forma *trend new worker count empty supermarket shell picture brooklyn sold online grocer coronavirus fear hopper stock*.
Por último, es necesario convertir las frases obtenidas del preprocesamiento a la codificación **Bag Of Words**. Esto es posible haciendo uso de la clase `CountVectorizer`. Por último, se une este vector al `dataframe`, y se elimina la columna que contiene el Tweet original. Por lo tanto, el conjunto de datos preprocesado se debe ver de la siguiente forma.

	Sentiment	aba	abandon	ability	abound	about	above	abroad	absolute	absurd	...
0	Negative	0	0	0	0	0	0	0	0	0	...
1	Positive	0	0	0	0	0	0	0	0	0	...
2	Positive	0	0	0	0	0	0	0	0	0	...
3	Negative	0	0	0	0	0	0	0	0	0	...
4	Neutral	0	0	0	0	0	0	0	0	0	...

Figura 2: Conjunto de datos preprocesado para la creación del algoritmo del árbol de decisión

5. Selección de parámetros

Ahora es necesario realizar la selección de parámetros del árbol. En este caso, se seleccionaron 2 posibles parámetros. El primero consiste en la altura del árbol, y el segundo en la métrica que usará el árbol. Ya sea *Coeficiente de gini* o *Entropía*. De esta forma, previo a esto, se separaron los datos en dos. Uno para entrenar y

seleccionar los parámetros y otro para probar el modelo. La separación se realizó en 70 % para entrenamiento y 30 % para test. Por su lado, los datos para entrenar pasaron por un proceso de **SMOTE**, con el fin de aumentar el número de datos de la clase **Neutro**. De esta forma, se usó el método de validación cruzada. A continuación se muestran los resultados.

5.1. Altura

Para seleccionar la altura del árbol, se realizaron tres gráficas. Una para sensibilidad, otra para presición y una última para la exactitud. A continuación, se ve la gráfica para la sensibilidad o *recall*

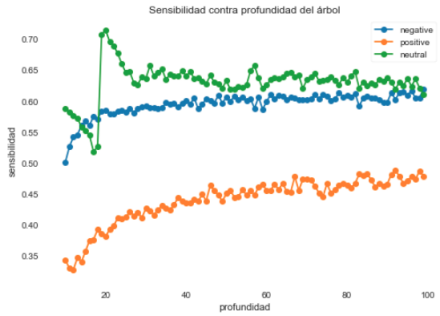


Figura 3: Gráfica de la sensibilidad vs la profundidad del árbol conseguida con el algoritmo de validación cruzada. En verde corresponde a la sensibilidad de la clase neutra, el azul de la negativa y el naranja de la positiva

En este caso, la sensibilidad corresponde al porcentaje de elementos de la clase en cuestión es clasificado correctamente. Como se ve, el recall más alto corresponde al de la clase neutra, con la clase negativa cerca. Por su lado, la clase positiva alcanza valores muy bajos. Es decir, según parece ser el algoritmo tiende a generar muchos *falsos negativos* para la clase positiva. Por otro lado, en cuanto a encontrar el valor óptimo para la profundidad del árbol. Como se ve, a partir de una profundidad de 30-40, no hay una mejoría sustancial en la calidad del algoritmo, por lo que se estarían consumiendo recursos innecesarios de memoria, además de poderse demorar más tiempo. A continuación, se ve la gráfica correspondiente a la precisión:

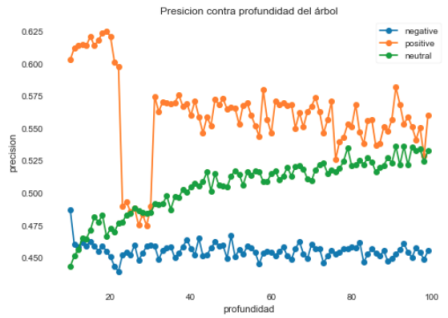


Figura 4: Gráfica de la precisión vs la profundidad del árbol conseguida con el algoritmo de validación cruzada. En verde corresponde a la sensibilidad de la clase neutra, el azul de la negativa y el naranja de la positiva

La precisión corresponde al porcentaje de datos clasificados en la clase en cuestión que en realidad pertenecen a otra clase. Por lo que nos da una medida del número de falsos positivos que genera el modelo. En este caso, la clase que más precisión tiene es la positiva, seguida de la neutra y por último la negativa. La negativa alcanza valores de 0.45, lo que nos indica que únicamente el 45 % de los datos clasificados como negativos pertenecen a la clase. Nuevamente acá se ve que desde un punto en particular, no hay una mejoría en los parámetros. Esto sucede más o menos a partir de 40. Por lo tanto, este debe ser aproximadamente el número niveles que puede tener el árbol. Por último, pasaremos a analizar la exactitud del árbol.

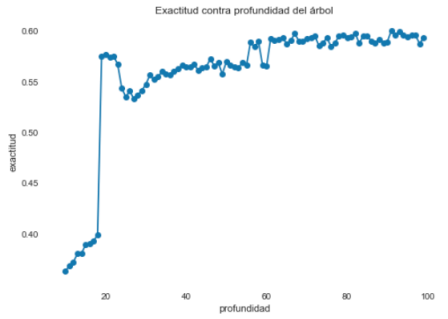


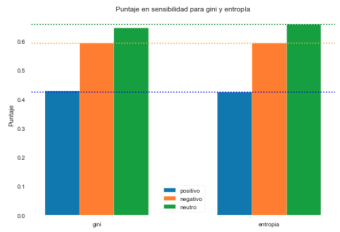
Figura 5: Gráfica de la exactitud vs la profundidad del árbol conseguida con el algoritmo de validación cruzada.

Como se ve, la exactitud es baja hasta una altura de aproximadamente 20. A continuación, la altura se mantiene relativamente constante, por lo que no vale la

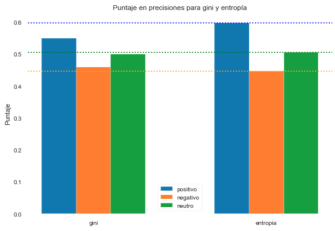
pena seguir gastando recursos si se va a conseguir un resultado muy similar. De esta forma es posible concluir que la profundidad del árbol debe ser aproximadamente de 40

5.2. Gini o Entropía

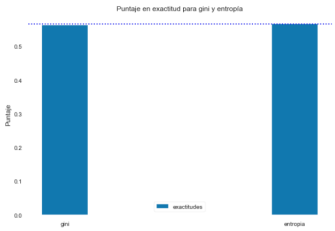
Para este caso se corrió para ambos modelos la validación cruzada. De acá, se sacaron los valores medios de sensibilidad y precisión para cada una de las clases, además de la exactitud del modelo. Al correr el algoritmo (con una profundidad de 40) se obtiene que es mejor usar **Entropía**. Los resultados arrojados por la validación se ven a continuación:



(a) Sensibilidades al ejecutar la validación cruzada con gini(izquierda) y entropía(derecha). El azul corresponde a la sensibilidad de la clase positiva, el naranja de la negativa y el verde del neutro



(b) Precisiones al ejecutar la validación cruzada con gini(izquierda) y entropía(derecha). El azul corresponde a la precision de la clase positiva, el naranja de la negativa y el verde del neutro



(c) exactitud al ejecutar la validación cruzada con gini(izquierda) y entropía(derecha)

Como es posible ver en las gráficas, hay una ligera mejoría cuando se usa el criterio de entropía. Por lo tanto, se llega a la conclusión de que este es el parámetro a usar. Ya con este análisis, es posible construir la tabla que se pide en el enunciado

Oportunidad/ Problema del negocio	Poder identificar tweets que tengan opiniones negativas respecto a la vacunación y al covid. Estos Tweets pueden representar un problema para la salud pública, por lo que su detección temprana puede ser de gran ayuda para realizar la censura debida de este tipo de contenido	
Descripción del proyecto desde el punto de vista de minería de datos.	En este caso, la oportunidad consiste en clasificar Tweets que traten el tema de la vacunación y del COVID. Estos pueden tener opiniones positivas, negativas o neutras respecto al tema. En este caso, clasificar las negativas sería de gran importancia, pues estas pueden ser un problema para la salud pública	
Detalles de la actividad de minería de datos		
Tarea	Técnica	Algoritmos y parámetros usados con la respectiva justificación
Clasificación	Arbol de decisión	En este caso, se seleccionaron los parámetros haciendo uso de validación cruzada. De esta forma fue posible conseguir los mejores parámetros para la profundidad y el criterio de decisión. Estos corresponden a 40 niveles y .[en]tropía", respectivamente

6. Construcción del modelo

De esta forma, se construyó un árbol teniendo en cuenta los parámetros seleccionados. El resultado obtenido tuvo los siguientes resultados. Los resultados obtenidos

	Precisión	Recall	f1-score	soporte
Negativo	0.59	0.55	0.57	491
Neutral	0.31	0.51	0.39	180
Positivo	0.59	0.48	0.53	469
Exactitud			0.52	1140
macro avg	0.50	0.52	0.50	1140
promedio ponderado	0.55	0.52	0.53	1140

Cuadro 1: Tabla obtenida como resultado al ejecutar el `classification report`

En primer lugar, cabe notar que la exactitud del modelo fue del 52 %. Por lo que se obtuvo un modelo que es capaz de clasificar correctamente poco más de la mitad de los datos. Aún así, es necesario poder entender más a profundidad que tan bien clasifica los datos de cada clase.
En primer lugar, de los datos negativos, sabemos que la precisión es del 59 % esto quiere decir que, de los datos que fueron clasificados acá, la mayoría fueron clasificados correctamente, indicando que no se producen muchos falsos positivos. Por su lado, la sensibilidad corresponde al 55 % esto nos indica que, de los datos negativos, solo el 55 % es clasificado correctamente.
Por su lado, para la clase neutral se obtuvo una precisión del 31 %. Esto indica que se están produciendo muchos falsos positivos, es decir, la mayoría de los datos clasificados como neutros no pertenecen a esta clase. Por su lado, la sensibilidad obtenida fue del 51 %, por lo que la mitad de los datos que son neutros son clasificados correctamente, por lo que se produce una tasa importante de falsos negativos.
Respecto a la clase positiva se obtuvo un valor de precisión del 59 %. Por lo tanto, se puede concluir que la mayoría de los datos clasificados como positivos pertenecen a esta clase. Aún así, la sensibilidad fue del 48 %, por lo que se concluye que aproximadamente la mitad de los datos que son positivos, no son clasificados correctamente.
A pesar de que se obtuvo un modelo que tiene en promedio un 50 % de errores, es necesario contrastar este en términos de velocidad respecto a revisar cada uno de los datos a mano y asignarle un valor. En caso de que el algoritmo presente una mejora significativa en tiempo, valdría la pena realizar la implementación del mismo. Asimismo, respecto a la clase negativa, esta es la que mejores estadísticas tiene. Esto es bueno, pues nos indica que la mayoría de los datos negativos son clasificados correctamente por el algoritmo.

En vista de que se quiere clasificar los posibles Tweets que tienen una opinión negativa respecto a la vacunación (lo que puede generar un problema de salubridad) el algoritmo puede funcionar. Este permitiría censurar posibles Tweets que cumplan con estas características a una mayor velocidad de lo que se haría a mano. Aún así, la implementación de este modelo más rápido podría censurar una gran mayoría de Tweets que no tienen una opinión negativa. Además de que se estarían dejando de censurar 40 % de los tweets, lo que puede ser una cantidad importante. De esta forma, sería importante que aquellos interesados en el algoritmo (ya sean gobiernos o las mismas redes sociales) revisen que tan eficiente sería la implementación del algoritmo en cuanto a la cantidad de tweets negativos que podrían ser censurados, el tiempo que se tarda el algoritmo y que tanto se siguen propagando estas ideas

peligrosas para la salud global.

7. Conclusiones

En conclusión, fue posible construir un modelo de clasificación para un conjunto de Tweets con opiniones acerca del COVID y la vacunación. En primer lugar, fue posible crear un **bag of words** luego de limpiar el contenido de los Tweets. Luego, se seleccionaron los parámetros de altura y criterio para obtener los mejores resultados posibles. Esto fue posible al hacer uso del método de validación cruzada. Los resultados obtenidos fueron que el óptimo eran 40 niveles y entropía. De esta forma, fue posible construir un árbol de clasificación.

Los resultados obtenidos para el árbol generado indican que tiene una exactitud del 52 %. Aún así, según los datos de sensibilidad y precisión, es posible concluir que el árbol tiene dificultad clasificando los valores neutros, obteniendose muchos falsos positivos. Por su lado, la clase negativa es la que mejores estadísticos tiene, siendo capaz de clasificar correctamente el 55 % de los datos de esta clase, además de que, el 59 % de los datos que fueron asignados a esta clase pertenecen a la misma.

Teniendo en cuenta esto, y que se trata de un problema de salud pública si la gente se vacuna, sería recomendable hacer uso de este algoritmo, puesto que sería capaz de censurar o eliminar la mayoría de elementos negativos. Aún así, es necesario obtener más información para saber si implementar o no el modelo. En primer lugar, toca tener en cuenta que aproximadamente el 40 % de los Tweets negativos pasarían desapercibidos. Por otro lado, se estarían censurando una gran cantidad de Tweets que no tienen una intención mala. Por lo tanto, para tomar una decisión definitiva, se recomienda al cliente revisar que tanto mejora la distribución de noticias falsas y opiniones negativas al hacer uso del algoritmo, además de que se disminuya el tiempo que se tarda en hacer la revisión.

8. Retos del proyecto

El principal reto del proyecto fue realizar el trabajo por mi cuenta. Por lo que me tocó realizar cada una de las tareas. Desde la programación, hasta la redacción, análisis de resultados y la presentación. Asimismo, la elección de `Python` también fue un reto significativo. A pesar de conocer el ambiente de Jupyter, y la librería Sklearn, no conocía la mayoría de las funcionalidades que esta tiene. Asimismo, no conocía la librería `nltk`, la cual permite la preparación de textos para algoritmos de inteligencia artificial. De esta forma, este proyecto estuvo lleno de retos. Aún así, el conocimiento adquirido en cuanto a `Python` y las librerías usadas fue bastante.