Tarea 3: Análisis de Textos

César jair Tamez Juárez

Junio 2022

1 Introducción

Retomando el trabajo realizado en la tarea 2, tenemos nuestra base de datos de textos que hablan del covid-19, con la clasificación original y las clasificaciones hechas por las librerías "textblob", "VaderSentiment" y "sentiwordnet". Lo que se hará a continuación será entrenar dos algoritmos utilizando cada una de las clasificaciones mencionadas anteriormente, esto con la finalidad de saber cuál clasificación y cuál algoritmo obtiene mejores resultados, para realizar dicha comparación utilizaremos diferentes métricas, así como una matriz de confusión

2 Algoritmos

Para el desarrollo de este trabajo se decidió tomar dos algoritmos los cuales son muy utilizados para la clasificación de textos, Naive-Bayes y La Máquina de Vectores de Soporte(SVM).

Naive-Bayes
El primer algoritmo que se utilizó en el presente trabajo para la clasificación de los textos fue el algoritmo de Naive-Bayes, este algoritmo es muy utilizado para la clasificación binaria y multiclase de textos, este algoritmo debe su nombre al teorema de Bayes el cual es utilizado en estadística para el cálculo de probabilidades condicionales, y el Naive significa en español inocente, ya que el algoritmo supone que existe independencia entre las variables predictoras, hecho que simplifica mucho las cosas.

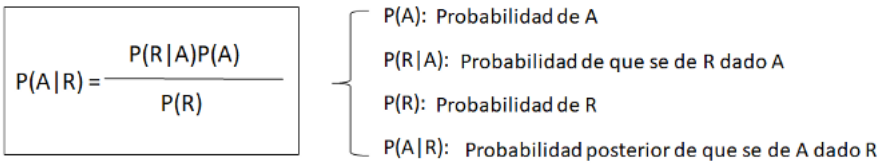

Figure 1: Teorema de Bayes

Ahora ya que el algoritmo de Naive-Bayes es uno de los algoritmos más utilizados, existe una librería en Python llamada sklear.naivebayes, la cual contiene la función MultinomialNB, esta función solo necesita de dos parámetros, la matriz de frecuencias de los datos de entrenamiento y su respectiva columna de clases para funcionar, una vez entrenado el modelo basta con utilizar la función naivebayes.predict para realizar las predicciones que se requieran.

Maquina de Vectores de Soporte

El segundo algoritmo que se utilizó para realizar la clasificación de los textos fue el algoritmo de Maquina de Vectores de Soporte o SVM por sus siglas en inglés, este es un algoritmo de aprendizaje supervisado el cual es utilizado en muchos problemas de clasificación y regresión. El objetivo de este algoritmo es encontrar un hiperplano que separe de la mejor manera posible a dos clases diferentes de puntos, este algoritmo fue pensado solo para el caso binario pero existen métodos para utilizar este algoritmo en el caso multiclase, estos métodos son OVO(One Versus One) y OVA(One Versus All), el primero de los métodos lo que hace es crear modelos binarios con todas las combinaciones de clases posibles, en el caso de OVA lo que hace es si se tienen n clases entonces se crean n modelos binarios donde cada modelo trata de distinguir entre los puntos pertenecientes a una clase del restos de puntos, es decir crea modelos binarios utilizando los puntos de una clase y su complemento. Dado que en nuestro caso es multiclase se utilizó el método OVO para entrenar el algoritmo.

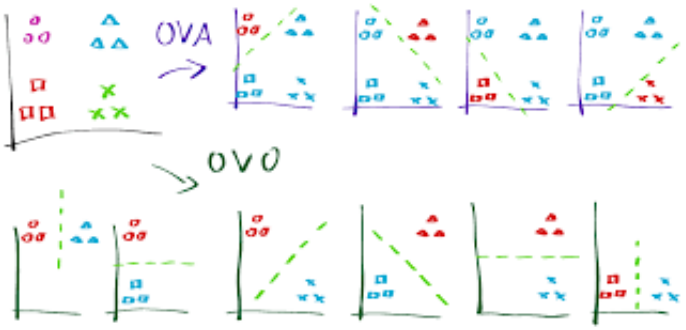

Figure 2: Ejemplo metodo OVO y OVA

3 Métricas de desempeño

Para realizar esta comparación vamos a recurrir a diferentes métricas de desempeño con las que se suelen evaluar a los algoritmos, estas métricas son las siguientes:

- **Exactitud:** Este valor se refiere a que tan cerca del valor real están las predicciones hechas por el algoritmo, básicamente es la razón entre el número de predicciones que se hicieron correctamente entre el número total de predicciones que se realizaron.
- **Precisión:** Este valor se refiere al porcentaje de predicciones que resultaron ser correctas, es la razón de el número de predicciones acertadas de una clase entre el total de predicciones que se obtuvieron para dicha clase.
- **Sensibilidad:** También es conocida como la tasa de verdaderos positivos, este valor es la proporción de predicciones correctas.
- **F1:** Esta métrica es importante ya que se deriva de la exactitud y la precisión convirtiendo de esta manera a dos métricas en una, básicamente es la media armónica entre la exactitud y la precisión.

Ahora todas estas métricas se derivan de la matriz de confusión, esta matriz contiene el número de predicciones correctas en la diagonal mientras que en el resto de la matriz encontramos el número de predicciones que fueron clasificada de mala manera ya sean falsos positivos o verdaderos negativos, esta matriz es cuadrada y su dimensión depende del número de clases que se tengan.

		Predicción	
		Positivos	**Negativos**
Observación	**Positivos**	Verdaderos Positivos (VP)	Falsos Negativos (FN)
	Negativos	Falsos Positivos (FP)	Verdaderos Negativos (VN)

Figure 3: Ejemplo de una Matriz de Confusión binaria

4 Análisis de resultados

Antes de aplicar los algoritmos es necesario transformar las etiquetas en números, por lo que la etiqueta "Neutral" será "0", "Positive" será "1", "Negative" será "2", "Extremely positive" será "3" y "Extremely negative" será igual a "4". Con esto tenemos 5 clases distintas, ahora veamos los resultados obtenidos. Primero tenemos los resultados obtenidos en las métricas de desempeño utilizando el algoritmo de Naive-Bayes: Como podemos ver en la tabla de la figura 4, tenemos

Naive-Bayes				
Etiquetas	Precisión	Exactitud	Sensibilidad	F1
Original	0.4334	0.3905	0.3905	0.3831
TextBlob	0.6039	0.5578	0.5578	0.5029
Vader	0.4334	0.3905	0.3905	0.3831
SWN	0.3989	0.4021	0.4021	0.3478

Figure 4: Métricas de desempeño

los valores de las cuatro métricas para cada uno de los conjuntos de etiquetas diferentes que tenemos, podemos ver que en general ninguno de los conjuntos obtuvo buenos resultados dado que ninguno pudo obtener más de un 60% de precisión. En la siguiente grafica podemos ver de mejor manera la comparación de los resultados anteriores:

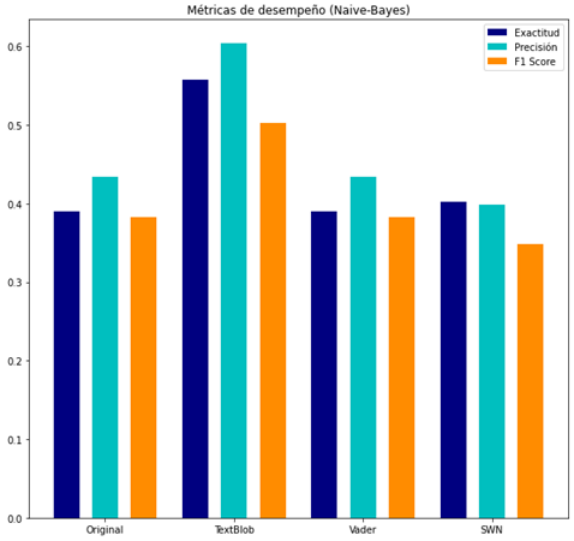

Figure 5: Grafica de Barras de las Métricas de desempeño

Como se aprecia en la gráfica de la figura 5, cuando se utilizó la clasificación realizada por TextBlob se obtuvieron los mejores resultados, pero aun así no son

buenos resultados pues con esos niveles de precisión y exactitud el algoritmo esta adivinando al azar más que predecir. Veamos ahora las matrices de confusión obtenidas en cada uno de los casos:

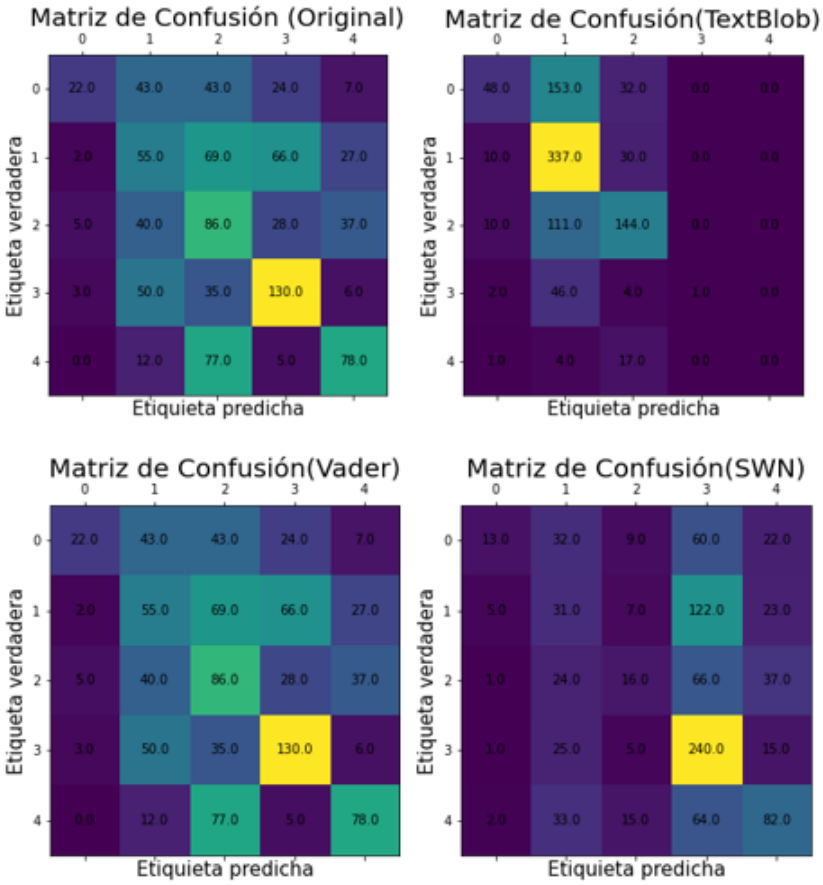

Figure 6: Matrices de Confusión Naive-Bayes

Podemos ver en la figura 6 las matrices de confusión para cada uno de los casos, podemos ver que en el caso de las clases originales su matriz de confusión nos muestra en su diagonal que solo pudo clasificar de buena manera a la clase 3 es decir "Extremely Positive", esto mismo sucede con las clasificaciones hechas por Vader y SWN, en el caso de la clasificación hecha por TextBlob pareciera que clasifico a la mayoría de los textos dentro de las primeras tres clases siendo "Positive" la clase que identificó de mejor manera. Podemos ver por la forma que tienen las cuatro matrices que los resultados obtenidos no fueron buenos. Veamos ahora los resultados obtenidos con el algoritmo de Maquina de Vectores de Soporte (SVM).

SVM				
Etiquetas	Precisión	Exactitud	Sensibilidad	F1
Original	0.4813	0.4621	0.4621	0.4608
TextBlob	0.6955	0.6915	0.6915	0.6865
Vader	0.4664	0.4484	0.4484	0.4483
SWN	0.4540	0.4231	0.4231	0.4309

Figure 7: Métricas de desempeño

Como podemos ver en la tabla de la figura 6, tenemos los resultados obtenidos en las métricas de desempeño cuando se utilizó el algoritmo SVM, en este caso pareciera que los valores mejoraron considerablemente pero no lo suficiente como para decir que fueron buenos resultados, veamos ahora la comparación con el grafico de barras: De la grafica de barras de la figura 7, podemos ver que una

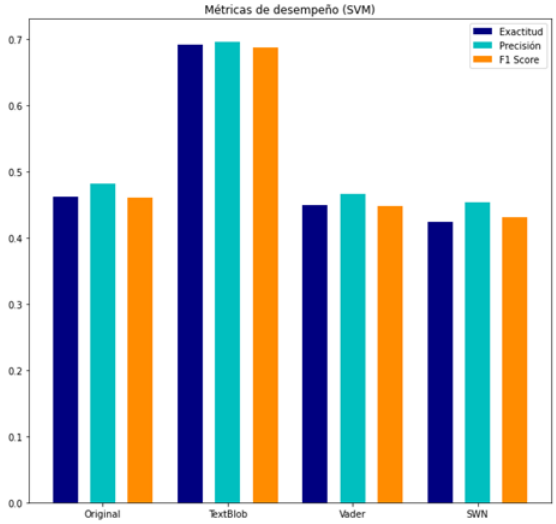

Figure 8: Grafica de Barras de las Métricas de desempeño

vez más la clasificación hecha por TextBlob obtiene los mejores resultados, alcanzando casi un 70% en cada una de sus métricas, mientras que el resto de las clasificaciones se mantiene con resultados similares y bastante malos, aunque mejores que los que se obtuvieron con Naive-Bayes. Ahora veamos las matrices de confusión obtenidas para este algoritmo:

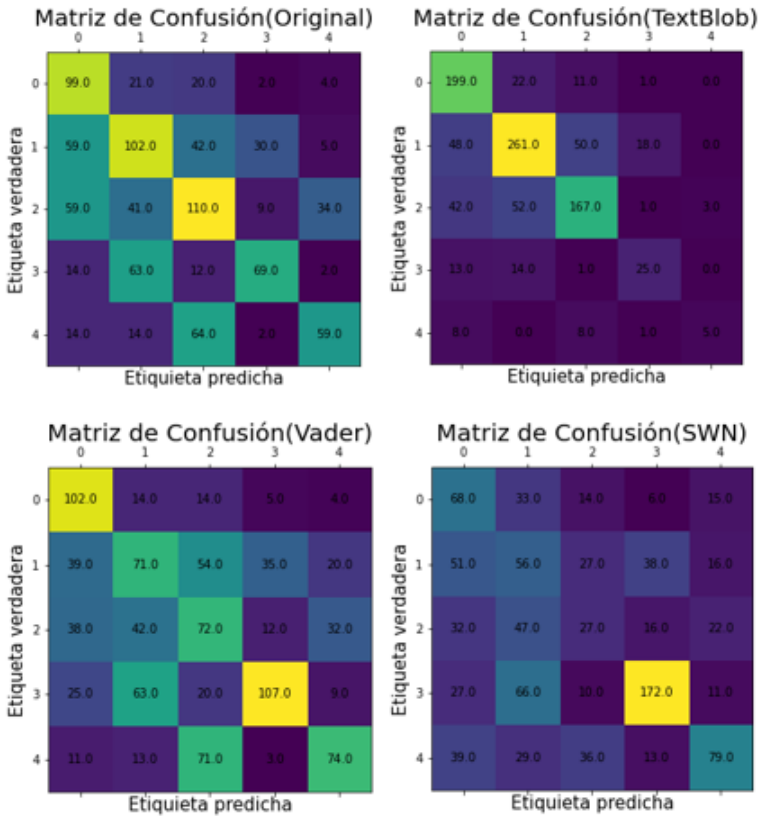

Figure 9: Matrices de Confusión SVM

En este caso podemos ver que las matrices de confusión tienen una mejor forma, una vez más en el caso TextBlob pareciera agrupar casi todos en la primeras tres clases, y este comportamiento, aunque no tan marcado, se puede apreciar también en el caso de Vader y la clasificación Original, mientras que en el caso de SWN pareciera que no logra identificar bien ninguna clase exceptuando a la clase "Extremely Positive". En comparación con las matrices obtenidas con Naive-Bayes podemos ver que estas tienen mejores diagonales se aprecia que se obtuvieron mejores resultados.

Con los resultados anteriores nos dimos cuenta que probablemente los datos se ajustarian mejor si tan solo fueran tres clases en lugar de cinco, por lo que reducimos el número de clases tomando a las clases "Extremely Positive" y "Extremely Negative" como parte de las clases "Positive" y "Negative" respectivamente, quedando entonces con las clases "Neutral", "Positive" y "Negative" que se transforman en "0","1" y "2" respectivamente.

Veamos ahora los resultados obtenidos tomando solo 3 clases y utilizando el algoritmo Naive-Bayes:

Naive-Bayes (3 Clases)				
Etiquetas	Precisión	Exactitud	Sensibilidad	F1
Original	0.6604	0.6631	0.6631	0.6271
TextBlob	0.6422	0.6136	0.6136	0.5721
Vader	0.6643	0.6452	0.6452	0.6130
SWN	0.6122	0.6168	0.6168	0.5664

Figure 10: Métricas de desempeño

Como podemos ver en la tabla de la figura 10, los valores de las métricas mejoraron mucho en comparación a los obtenidos cuando se utilizaron 5 clases, recordemos que en el caso de 5 clases se obtuvieron métricas que no sobrepasaban el 60% y ahora con 3 clases todas las clasificaciones obtuvieron valores por encima del 60%, veamos ahora en la grafica de barras la comparación entre las métricas de las clasificaciones:

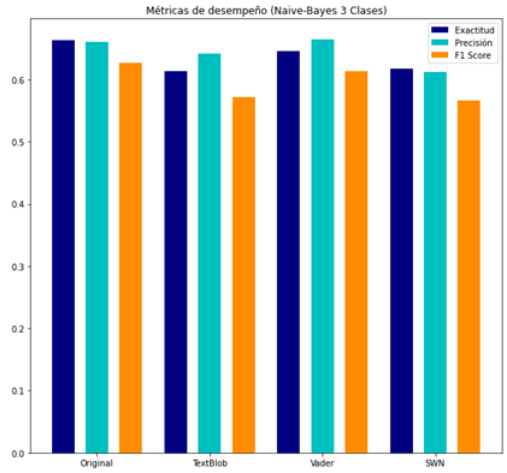

Figure 11: Grafica de Barras de las Métricas de desempeño

De la grafica de barras de la figura 11, podemos ver que todas las clasificaciones obtuvieron resultados similares, siendo la clasificación original la que obtuvo ligeramente mejores resultados. En este caso podemos decir que los resultados obtenidos son aceptables. Veamos ahora las matrices de confusión obtenidas:

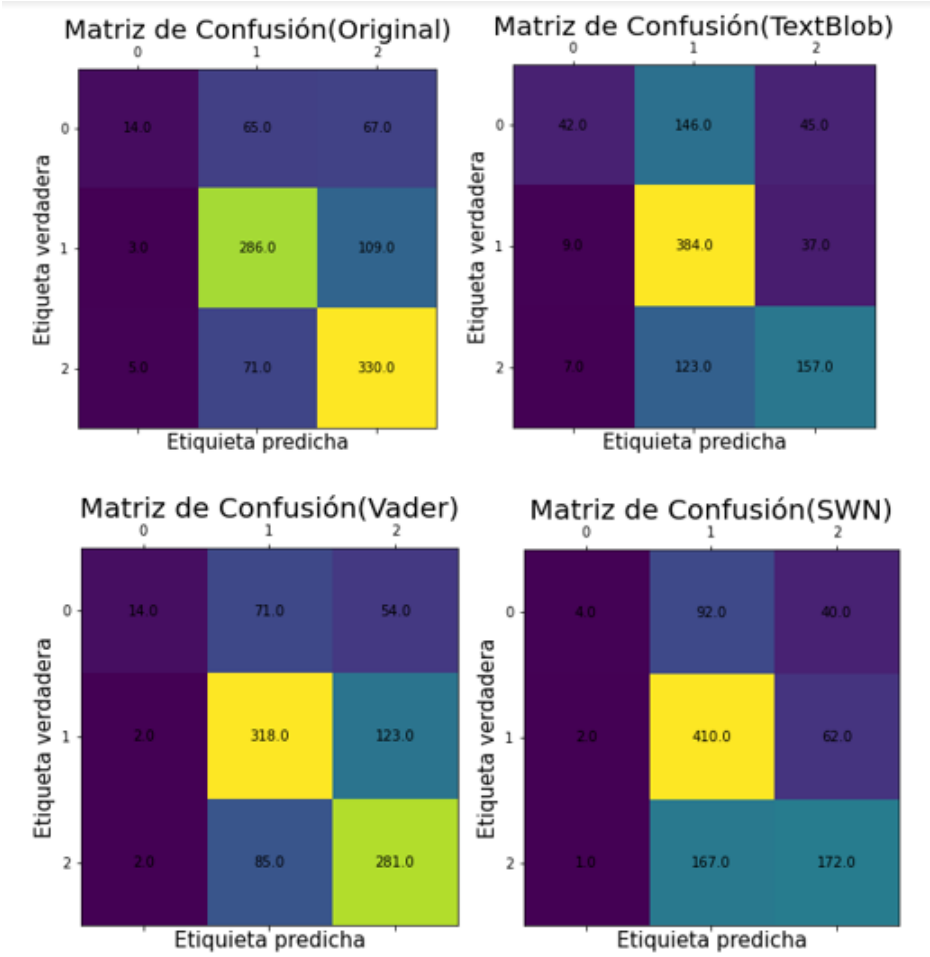

Figure 12: Matrices de Confusión Naive-Bayes

En este caso con 3 clases, podemos ver que en general todos los casos no pudieron clasificar de buena manera a la clase “Neutral” fue confundida en la mayoría de los casos por la clase “Positive”, por su parte la clase “Positive” fue la mejor clasificada en general, ahora si vemos la diagonal notamos que efectivamente fue la clasificación Original seguida muy de cerca por Vader quienes obtuvieron mejores resultados. Podemos decir que en general se obtuvieron buenos resultados en todos los casos.

Ahora veamos los resultados obtenidos con 3 clases utilizando el algoritmo de Maquina de Vectores de Soporte (SVM):

SVM (3 Clases)				
Etiquetas	Precisión	Exactitud	Sensibilidad	F1
Original	0.7106	0.6810	0.6810	0.6900
TextBlob	0.7514	0.7442	0.7442	0.7442
Vader	0.7252	0.7073	0.7073	0.7107
SWN	0.6551	0.6357	0.6357	0.6425

Figure 13: Métricas de desempeño

En este caso los valores de la métricas mejoraron demasiado, podemos ver en la tabla dela figura 13 como todos excepto SWN obtuvieron valores en sus métricas por encima del 70%, Estos ya se pueden considerar como resultados aceptables, veamos ahora la comparación entre clasificaciones en el siguiente grafico de barras: Como podemos ver es la clasificación hecha con TextBlob con

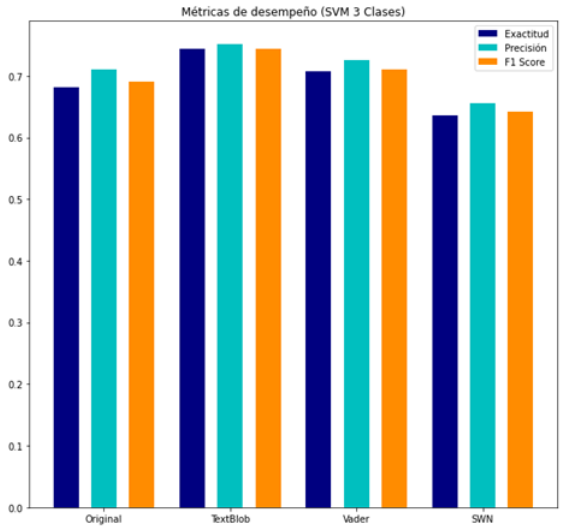

Figure 14: Grafica de Barras de las Métricas de desempeño

la que se obtuvieron los mejores resultados, en general con esta combinación de algoritmo y numero de clases es con la que mejores resultados se obtuvieron, veamos ahora las matrices de confusión obtenidas para este caso:

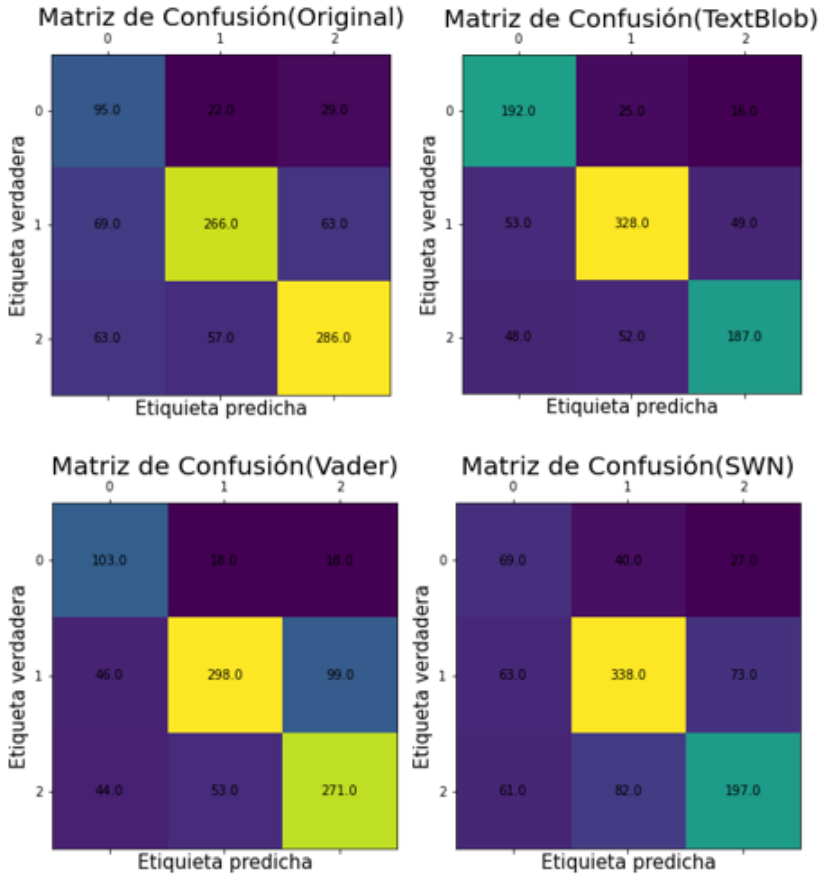

Figure 15: Matrices de Confusión Naive-Bayes

Si vemos las diagonales delas matrices de confusión podemos ver que esta ves si pudieron identificar de buena manera a cada una de las clases, en todos los casos pareciera que la clase que más confundieron y a su vez la que mejor clasificaron fue "Positive", esto probablemente se deba a que se tienen mas textos en esta clase, en el caso de SWN parece que también batalló en clasificar la clase "Neutral". En general se obtuvieron muy buenos resultados y se nota en la forma de las matrices.

5 Conclusión

Después de analizar los resultados obtenidos, podemos decir que en ambos caso utilizando tres o cinco clases, cuando se utilizó el algoritmo de Máquina de Vectores de Soporte (SVM) se obtuvieron mejores resultados que con Naive-Bayes. Ahora también notamos que se mejoraron demasiado los resultados obtenidos cuando se trabajo con solo tres clases en lugar de cinco, esto probablemente a que no estaba tan marcada la diferencia entre las clases "Positive" y "Extremely Positive" y "Negative" y "Extremely Negative". Ahora de todas las clasificaciones hechas, las que hizo TextBlob obtuvieron mejores resultados en casi todos los caso, por su parte la clasificación original y la hecha por Vader en la mayoría de los caso mantuvieron valores de métricas parecidas, mientras que en el caso de la clasificación hecha por SWN se obtuvieron los peores resultados en todos los casos. En conclusión utilizando la clasificación hecha con TextBlob y el algoritmo de Máquina de Vectores de Soporte (SVM) se obtienen los mejores resultados, probablemente se podrían mejorar si tuviéramos más datos.

References

[1] *Link de la base de datos original*
https://www.kaggle.com/datasets/datatattle/covid-19-nlp-text-classification

[2] *Link de Github*
https://github.com/CesarJairTJ/FCFM