Clasificación de Texto de Reseñas Sobre Aplicaciones Móviles

IRVING DANIEL ESTRADA LÓPEZ
Facultad de Ciencias Físico Matemáticas
Universidad Autónoma de Nuevo León
Nuevo León, México
irving.estradalo@uanl.edu.mx

***Abstract*—En 2021 se reportaron 230 Billones de descargas de aplicaciones móviles a través del mundo. Para el 2025 la generación de información ascenderá a un total de 163 zettabytes lo que significa que el volumen aumentará cerca de 10 veces a nivel mundial. Este aumento exponencial en la generación de datos tanto estructurados como no estructurados, es un área de oportunidad para extraer información valiosa. La clasificación de textos se ha considerado como un método vital para gestionar y procesar una gran cantidad de documentos en formatos digitales los cuales están en continuo aumento. En este artículo se propone una metodología la cual utilizar al clasificar calificaciones de reseñas de aplicaciones móviles, un preprocesado de texto para dichas reseñas, una solución para trabajar con un conjunto de datos no balanceado. Por último, se evalúan tres diferentes modelos: *Multinomial Naive Bayes*, *Decision Tree* y *Random Forest*.**

I. Introducción

Con el rápido incremento de la tecnología a través del tiempo, el Internet se ha vuelto una parte esencial en la vida de las personas. Solamente en el año 2020 en México había 84.1 millones de usuarios en Internet y 88.2 millones de usuarios con teléfonos celulares, de los cuales 9 de cada 10 usuarios de teléfono celular disponen de un celular inteligente. [1] A nivel global, en 2021 se reportaron 230 Billones de descargas de aplicaciones móviles. [2]

Según un estudio de Data Age para el 2025 la generación de información ascenderá a un total de 163 zettabytes, lo que significa que el volumen aumentará cerca de 10 veces a nivel mundial. Esto se vuelve un inconveniente al momento de analizar la información. Día a día se genera una gran cantidad de datos no estructurados debido a las redes sociales, a los distintos sitios web y a las aplicaciones móviles.

Estos datos generados nos muestran el comportamiento de las personas, contienen una gran cantidad de información la cual es difícil de tratar debido a su formato. La demanda de análisis de texto va en aumento, ya que el manejo de grandes cantidades de datos para extraer información relevante de algún producto o servicio es de suma importancia para la industria pública y privada.

En este artículo se propone una metodología para la clasificación de calificaciones de reseñas sobre aplicaciones móviles.

Donde en la Sección 2 comenzaremos con los antecedentes que se han estudiado a través del tiempo, en los cuales se proponen metodologías y modelos para llevar a cabo un análisis de texto.

Después, en la Sección 3 se mostrarán los datos con los cuales se trabajaron, explicando cada una de las columnas y señalando puntos importantes para dar entendimiento al conjunto de datos.

En la Sección 4 estudiaremos la metodología propuesta, dando detalle a cada una de las fases para llegar a generar un buen desempeño en la clasificación de reseñas sobre aplicaciones móviles. Donde se destacan la parte del preprocesado de texto propuesta para las reseñas de aplicaciones móviles, la extracción de características mediante el método TF-IDF y la generación de datos sintéticos con el algoritmo SMOTE. Se estudian tres de los algoritmos más utilizados para la clasificación de texto: *Multinomial Naive Bayes*, *Decision Tree* y *Random Forest*.

En la Sección 5 mostraremos los resultados obtenidos. En la Sección 6 recopilaremos las observaciones relevantes de los resultados, haciendo una comparación con un artículo similar acerca de clasificación de malas reseñas de aplicaciones móviles. Finalmente concluiremos el artículo mencionando lo más relevante y hablando de futuras aportaciones, siendo este artículo la base para el desarrollo de estudios con un nivel mayor de complejidad.

II. Antecedentes

El análisis de texto comenzó desde 1992 cuando las personas trataban de organizar documentos. Se han propuesto distintas metodologías para el desarrollo de un análisis de texto [3], sin embargo, estas metodologías son acerca del análisis de texto en general.

Conforme se han hecho pruebas para generar modelos y alcanzar un buen desempeño en la clasificación de texto, los estudios han orillado a los investigadores a profundizar en la parte del preprocesado de texto en general. [4]

Que entre dichos procesos destacan la *tokenization*, el remover las *stop words*, la lematización, entre otras cosas.

Otra fase determinante para poder trabajar con texto es la extracción de características. Existen diferente tipo de extracción de características como *Weighted Words*, *Bag of Words*, *Term Frequency-Inverse Document Frequency* (TF-IDF), *Word Embedding*, entre otras. En este caso utilizaremos

la TF-IDF ya que es una metodología de las más utilizadas. [5]

A través del tiempo se han buscado modelos los cuales se ajusten de la mejor manera al análisis de texto.

Se ha estudiado el desempeño de los algoritmos de *Machine Learning* utilizándolos con texto [6] para seleccionar los que presenten resultados destacables. Uno de los modelos más utilizados en la clasificación de texto es el *Naive Bayes*. Este modelo destaca por su simpleza y eficiencia.

Actualmente los investigadores se han interesado por el estudio de la clasificación de malas reseñas en las aplicaciones móviles debido a su gran cantidad de usuarios [7], en dicho documento se utilizan el *Naive Bayes* y el *Decision Tree* para llevar a cabo una clasificaciones binarias y clasificaciones múltiples, donde destacan la relevancia de dichas reseñas como fuente de información de detección de áreas de oportunidad para las aplicaciones.

III. Datos

El conjunto de datos con el que se trabajó en este artículo es de dominio público y fue obtenido de Kaggel. [8] Consta de 200,000 registros los cuales fueron extraídos de la *Google Play Store*. Como podemos ver en la figura 1 el conjunto de datos contiene un identificador de la reseña, el contenido de la reseña, nuestra variable objetivo que es la calificación dada por el usuario y por último el nombre de la aplicación.

Fig. 1.

	reviewId	content	score	app
0	38d8adef-9e35-468f-96b9-0ede64e75bc1	Good aap	5	Facebook
1	fc34d9c7-7117-4075-a8ba-54a3c533bc36	Nyc	5	Facebook
2	e9e1f134-050e-4b28-867e-f667fbc34b1d	Thought the first time	5	Facebook
3	8efa37fc-6ed7-4d59-b4d3-94768e708a90	So nice	5	Facebook
4	7d157d23-16ea-4323-99bf-390e4324f06a	😗	1	Facebook

En la Tabla 1 tenemos una pequeña descripción de cada una de las columnas, las cuales cabe destacar que en la columna "score" tenemos valores del 1 al 5, siendo cinco la mejor calificación y uno la peor. En la columna "app" se cuenta con 20 distintas etiquetas con el nombre de cada una de las aplicaciones, sin embargo, nos enfocaremos en la clasificación de las reseñas de todas estas aplicaciones. De estas cuatro columnas solamente utilizaremos la columna "content" y la columna "score", ya que estas son las que se consideran relevantes para este artículo.

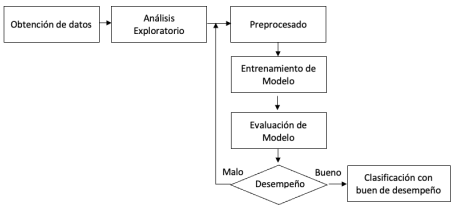
Variable	Representación
reviewId	Identificador del usuario
content	Reseña redactada por el usuario
score	Calificación dada por el usuario
app	Nombre de la aplicación

TABLE I
Variables del Conjunto de datos

IV. Metodología

En la figura 2 tenemos la metodología propuesta, la cual desglosaremos etapa por etapa para estudiar a detalle los procesos que se llevaron a cabo. La única sección que se tomará como vista es la de "Obtención de Datos", ya que anteriormente se cubrió la fuente de los datos con los cuales se trabajarán en este artículo.

Fig. 2.

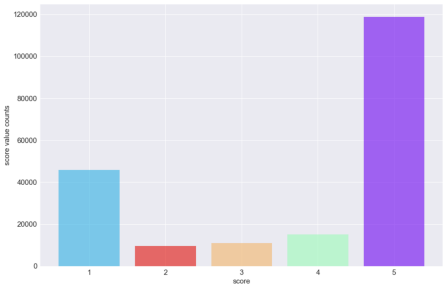


A. Análisis Exploratorio

Esta es la primera fase de nuestra metodología, el principal objetivo es el poder identificar áreas de oportunidad para anticipar el buen desempeño de nuestros modelos. El identificar si nuestro conjunto de datos contiene datos nulos nos puede evitar errores en un futuro. En el caso del conjunto de datos utilizado en este artículo contiene valores nulos, de los cuales fueron eliminadas sus filas. Lo anteriormente mencionado fue debido a que la cantidad de datos nulos en este conjunto de datos fue pequeña, además de la gran cantidad de datos con los que se trabajará. Los datos totales después de eliminar los datos nulos son 199,993.

Como podemos ver en la figura 3 las cinco clases que tenemos en nuestra columna "score" no están balanceadas. El tener un conjunto de datos no balanceado quiere decir que existe una clase que abarca la mayoría de las observaciones de nuestros registros. En este caso la clase 5 tiene 118,759 registros, superando a las demás clases por más del doble. El tener clases no balanceadas afecta a la mayoría de los modelos de *Machine Learning*, es importante corregir este aspecto en la sección de preprocesado.

Fig. 3.



B. Preprocesado

En la siguiente fase se comienza con el preprocesado de texto, la intención es comenzar a adecuar nuestras reseñas de aplicaciones móviles para que los modelos obtengan un buen desempeño.

Remover URL: En caso de existir alguna liga que nos lleve a un sitio web, no aporta alguna característica valiosa a los análisis posteriores.

Remover signos de puntuación: Debemos de quitar los signos de puntuación ya que no aportarán en el análisis, en otra situación como el análisis de sentimientos nos podrían brindar información valiosa. Esto se debe de hacer después de remover la URL, ya que la URL contiene signos de puntuación y todo está concatenado, de lo contrario existe la posibilidad de que se interprete como palabras.

Convertir *emojis* en palabras: Es importante tomar en cuenta que hoy en día es común el utilizar *emojis* mientras escribimos, estos nos brindan información valiosa para nuestra clasificación. Se integró un diccionario el cual contiene los códigos Unicode de una gran cantidad de *emojis*, siendo estos códigos las llaves del diccionario y las palabras que representarían al *emoji* los valores.

Conversión a minúsculas: El objetivo de la conversión de las palabras a minúsculas es debido a que la extracción de características e incluso muchos de los lenguajes de programación son sensibles entre mayúsculas y minúsculas, esto afectando a nuestro modelo. Identificando como distintas, palabras iguales.

Tokenizar: Este paso consiste en dividir el texto en las unidades que lo conforman, entendiendo por unidad el elemento más sencillo con significado propio para el análisis en cuestión, en este caso las palabras. Este paso es necesario para poder manipular el texto lo que resta del preprocesado.

Eliminación de *stop words*: Las *stop words* son consideradas palabras que no tienen un significado por si solas, sino acompañan a las otras palabras. Es importante removerlas debido a que abarcan una gran cantidad de los textos, generalmente aparecen en el top de palabras repetidas. Las *stop words* afectan el costo computacional de todos nuestros procesos.

***Stemming* o lematización:** Las palabras tienen muchas variaciones, es importante encontrar el estado base o raíz de las palabras para que se pueda llevar a cabo de manera adecuada el análisis que se quiere lograr. El *stemming* es un algoritmo que busca el núcleo de la palabra, pero en el inglés así como en otros idiomas existen excepciones, dándonos resultados no esperados. El lematizar es más complejo ya que busca el lema de la palabra, esto es una serie de caracteres que forma una unidad semántica y que puede constituir una entrada en el diccionario. En este caso se utilizó la lematización para encontrar un estado base de la palabra, apoyándonos en el POS (*Part of Speech*) con la intención de disminuir el margen de error al momento de encontrar el lema.

Extracción de características: Una vez llevado a cabo nuestro preprocesado de texto, lo siguiente es realizar el TF-IDF (*Term Frequency — Inverse Data Frequency*). Lo que busca esta técnica es medir la frecuencia de las palabras y después ponderarlas por su relevancia, es decir, que las palabras menos comunes serán las que más peso tengan, de esta manera éstas sean las que mejor expliquen de qué habla cada oración. Por esta parte era importante eliminar las *stop words*, ya que se hace una matriz respecto a cada palabra con su respectivo TF-IDF que con las *stop words* se convertiría en una matriz mucho más grande, obteniendo cada una de estas un TF-IDF de 0 debido a que no tienen significancia en las oraciones.

$$W_{i,j} = tf_{i,j} \times log(\frac{N}{df_i})$$

$tf_{i,j}$ = número de ocurrencias de i en j
df_i = número de documentos que contiene i
N = número total de documentos

En la figura 4 tenemos la matriz anteriormente mencionada y podemos identificar que cuenta con 2,500 columnas, éstas representan cada una de las palabras. Cabe destacar que este paso nos lleva de trabajar con datos no estructurados que en este caso es el texto, a trabajar con una matriz con valores numéricos.

Fig. 4.

	0	1	2	3	4	5	6	7	8	9	...	2490	2491	2492	2493	2494	2495	2496	2497	2498	2499
0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
1	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
2	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
3	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
4	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0

Tratar conjunto de datos no balanceados: Como habíamos mencionado anteriormente en la parte de análisis descriptivo, las clases que tenemos en nuestro conjunto de datos no están balanceadas. Para que nuestros modelos tengan un buen desempeño al momento de aprender o identificar los patrones de cada una de las clases y no se sesguen, éstas deben de tener un número cercano de observaciones entre sí, de lo contrario nuestros modelos no tendrán oportunidad de aprender correctamente las clases que cuenten con pocas observaciones. Para realizar el balance en nuestro conjunto de datos utilizaremos el algoritmo SMOTE (*Synthetic Minority Over-sampling Technique*) el cual genera datos sintéticos para equilibrar las respectivas clases. Este sobremuestreo se aplica a nuestra matriz de TF-IDF, de esta forma generando nuevas observaciones en base a las del conjunto de datos. Dándonos como resultado 118,759 observaciones por clase, dando un total de 593,795 observaciones. Una ventaja que nos brinda esta solución es que no es necesario reducir el número de observaciones de la clase mayoritaria, perjudicando al número de observaciones en el conjunto de entrenamiento.

C. Modelos

***Multinomial Naive Bayes*:** Este es uno de los modelos probabilistas más simples y utilizados en clasificación de texto [4], ya que produce resultados tan buenos como otros modelos más sofisticados. Además que en recientes estudios se ha utilizado para clasificar categorías de malas reseñas relacionadas con aplicaciones móviles. [7] Se basa en la aplicación de la Regla de Bayes para predecir la probabilidad condicional de que un documento pertenezca a una clase $P(c_i|d_j)$ a partir de la probabilidad de los documentos de cada clase $P(d_j|c_i)$ y la probabilidad a priori $P(c_i)$ de la clase en el conjunto de entrenamiento.

$$P(c_i|d_j) = \frac{P(c_i)P(d_j|c_i)}{P(d_j)}$$

***Decision Tree*:** El árbol de decisión es un método no paramétrico de aprendizaje supervisado usado para la clasificación y regresión. Aprende generando reglas inferidas por las características de los datos. Algunas de las ventajas de los árboles de decisión son que son simples de interpretar, requieren poco preprocesado de datos, el costo de usar el árbol es logarítmico con respecto al número de puntos de datos usado en el entrenamiento del árbol. Algunas de las desventajas que tienen los árboles de decisión son que hay riesgo de un sobreajuste por la generación de reglas, también hay riesgo de que estén sesgados con respecto a los datos de entrenamiento. Cualquier pequeño cambio en los datos de entrada pueden suponer un árbol de decisión completamente diferente. También se cuenta con el problema de generar el número óptimo en los árboles de decisión. Este modelo también se ha probado obteniendo buenos resultados en la clasificación de texto, su popularidad es debido a su velocidad y su buen desempeño con texto.

***Random Forest*:** El modelo de *Random Forest* como su nombre lo dice es un conjunto de árboles de decisión individuales que operan en conjunto. Cada uno de estos árboles de decisión en el *Random Forest* crea una clase de predicción y la clase con más votos es aquella que se convierte en nuestro modelo. La razón por la que el *Random Forest* es considerado un excelente modelo es debido a que un gran número de modelos relativamente no correlacionados superarán a cualquier modelo individual. Cada uno de los árboles de decisión se "protegerán" entre ellos con respecto a su error individual, mientras que algunos árboles no serán correctos, otros si los serán y se complementarán no tomando una decisión absoluta con un solo modelo. Una de las desventajas de este modelo es que se considera una caja negra, debido a su complejidad. El *Random Forest* se ha utilizado en la clasificación de texto, pero no tiene la popularidad de los dos anteriores, este modelo es más robusto y más tardado. [4]

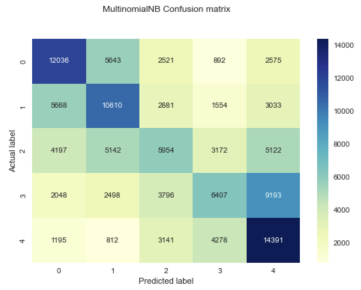
V. Resultados

Para evaluar los modelos utilizaremos su respectiva matriz de confusión además de su reporte de clasificación basándonos en el F1-Score ya que lo que se busca es un balance entre la precisión y el *recall*.

A. Multinomial Naive Bayes

Como podemos ver en la figura 5 aparentemente el modelo *Multinomial Naive Bayes* no obtuvo buenos resultados, a simple vista no pareciera estar sesgado, sin embargo, hay que destacar de la última columna que si existe la sospecha de un posible sesgo ya que la mayoría de los errores recaen en esa columna. Otro detalle que podemos destacar de esta matriz de confusión es que los mejores resultados parecieran ser tanto los de la calificación más baja como la calificación más alta, esto se debe a su diferencia significativa en las palabras que se podrían usar para generar una reseña con una mala calificación, así como una reseña con una buena calificación. Por otro lado, las dos clases con mayor dificultad de ser clasificadas entre ellas fueron la clase 3 y la clase 4. Estas clases se sospecha que son el caso contrario a lo anteriormente mencionado, utilizan palabras parecidas en sus reseñas al momento de calificar una aplicación y por eso el modelo no puede diferencias de forma significativa la diferencias que hay entre ellas. El reporte de clasificación nos ayudará a complementar las observaciones anteriormente mencionadas.

Fig. 5.



El reporte de clasificación en la figura 6 nos confirma el mal desempeño en el F1-Score de la calificación 3, obteniendo un 28%. Confirmamos el buen desempeño de la mejor y la peor calificación, estas tienen las métricas más altas. El modelo *Multinomial Naive Bayes* no obtuvo un buen desempeño con un 42% de *accuracy* a pesar de que es uno de los más utilizados al momento de clasificar texto.

Fig. 6.

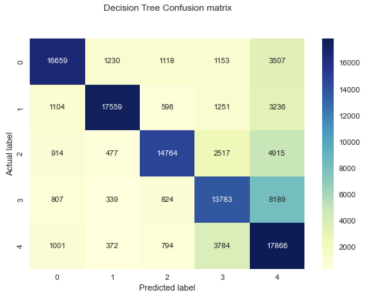
	precision	recall	f1-score	support
1	0.48	0.51	0.49	23667
2	0.43	0.45	0.44	23746
3	0.33	0.25	0.28	23587
4	0.39	0.27	0.32	23942
5	0.42	0.60	0.50	23817
accuracy			0.42	118759
macro avg	0.41	0.42	0.41	118759
weighted avg	0.41	0.42	0.41	118759

0.4159516331393831

B. Decision Tree

La matriz de confusión del *Decision Tree* ilustrada en la figura 7 nos indica que obtuvo buenos resultados, en este modelo es más visible el sesgo anteriormente mencionado. En la última columna podemos darnos cuenta como el modelo ajusta las demás clases a la última. De igual forma pareciera tener el mismo problema que el modelo anterior clasificando la clase 3 y la clase 4. A pesar de los puntos anteriormente mencionados, la diagonal principal de la matriz de confusión indica un buen desempeño.

Fig. 7.



Su reporte de clasificación en la figura 8 nos confirma lo dicho de la diagonal principal. A diferencia del *Multinomial Naive Bayes* el *Decision Tree* obtiene en la última clase el F1-Score más bajo, siendo lo contrario al modelo anterior. Revisando las demás métricas de la calificación 5, el mal resultado en el F1-Score es debido a su mal desempeño en la métrica de precisión. El modelo obtuvo buenos resultados con un 68% de *accuracy*.

Fig. 8.

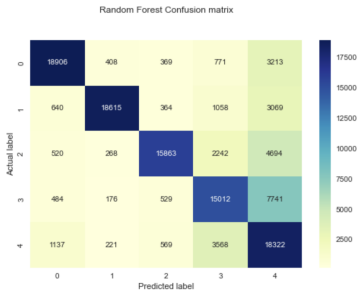
	precision	recall	f1-score	support
1	0.81	0.70	0.75	23667
2	0.88	0.74	0.80	23746
3	0.82	0.63	0.71	23587
4	0.61	0.58	0.59	23942
5	0.47	0.75	0.58	23817
accuracy			0.68	118759
macro avg	0.72	0.68	0.69	118759
weighted avg	0.72	0.68	0.69	118759

0.6789464377436657

C. Random Forest

Al igual que el *Decison Tree* el modelo de *Random Forest* en la figura 9, pareciera que se sesga respecto a la última clase. También cabe mencionar que al igual que los modelos anteriores la mayor dificultad que enfrenta este modelo es el poder diferenciar la clase 3 y la clase 4. La diagonal principal obtiene buenos resultados.

Fig. 9.



El reporte de clasificación en la figura 10, nos indica el buen desempeño que obtuvo el *Random Forest* abarcando el 73% de *accuracy* y obteniendo los F1-Scores de todas las clases arriba del 60%. La calificación con peor equilibro entre el *recall* y la precisión es la calificación 5.

Fig. 10.

	precision	recall	f1-score	support
1	0.87	0.80	0.83	23667
2	0.95	0.78	0.86	23746
3	0.90	0.67	0.77	23587
4	0.66	0.63	0.64	23942
5	0.49	0.77	0.60	23817
accuracy			0.73	118759
macro avg	0.77	0.73	0.74	118759
weighted avg	0.77	0.73	0.74	118759

0.7302015005178555

VI. Discusión

Fig. 11.

Modelo	F1-Score					Accuracy
	1	2	3	4	5	
Naive Bayes	0.49	0.44	0.28	0.32	0.5	0.42
Decision Tree	0.75	0.8	0.71	0.59	0.58	0.68
Random Forest	0.83	0.86	0.77	0.64	0.6	0.73

En figura 11 tenemos una tabla comparativa de los tres modelos, donde se recopilan los F1-Scores y las *accuracies* de cada modelo. El modelo *Multinomial Naive Bayes* fue el que obtuvo los peores resultados, a pesar de su recurrente uso en la clasificación de texto. Este modelo se ha probado en otros artículos con la clasificación binaria y clasificación múltiple de categorías de malas reseñas de aplicaciones móviles de la *Google Play Store*. Dicho modelo ha obtenido excelentes resultados, obteniendo F1-Scores mayores a 70% para clasificación binaria [7], sin embargo, para la clasificación múltiple al igual que en nuestro caso no alcanza buenos resultados. En el mismo artículo la clasificación múltiple obtuvo en promedio un 50% de F1-Scores, no despegándose con una gran diferencia del nuestro alcanzando 42%. Otro factor que destacar es que, en el artículo mencionado no se generaron datos sintéticos, se tenía un conjunto de datos balanceado. Existe la sospecha de que al crear datos sintéticos basados en los reales con el algoritmo SMOTE afectara al modelo, ya que como vimos en la sección pasada, el *Multinomial Naive Bayes* trabaja con la probabilidad condicional. Lo destacable de este modelo es que, a diferencia de los otros dos, su mejor clasificación fue la de la calificación 5. Esto aumenta la sospecha de que los datos sintéticos pueden estar afectando a este modelo, ya que si recordamos en los datos reales los que dominaban en cantidad eran los registros con 5 de calificación.

El *Dicision Tree* es otro de los modelos los cuales se probaron en la clasificación de malas reseñas, que al igual que *Multinomial Naive Bayes* en dicho artículo no le favoreció la clasificación múltiple, alcanzando un 54% de F1-Scores en promedio. En nuestro caso obtuvo resultados bastante aceptables, tuvo problemas con la calificación 5 ya que aparentemente existía un sesgo. En las primeras 3 calificaciones obtuvo excelentes resultados, siendo las últimas dos las que perjudicaron a este modelo. Las características más relevantes de este modelo es su velocidad y buen desempeño.

El *Random Forest* fue utilizado ya que en la literatura se especifica que se ha aplicado en la categorización de documentos [6], sin embargo, no se encontró un artículo el cual evaluara el *Random Forest* en un caso de estudio similar al de este artículo. Ya que este modelo es un conjunto de *Decision Tree* se tomará como referencia dicha métrica. Al igual que el *Decision Tree* tuvo problemas con la calificación 5. Presentó una mejora en los F1-Scores de cada una de las clases y por lo tanto en el *accuracy*, sin embargo, esta diferencia no es significativa. Este modelo nos reduce la posibilidad de sobreajuste, pero en este caso la matriz de confusión nos indicaba como el modelo trataba de ajustar las demás calificaciones en la calificación 5, al igual que en el *Decision Tree*. El modelo de *Random Forest* podría mejorar

aumentando el número de árboles en él, sin embargo, no se debe descartar un posible sobreajuste.

El *Random Forest* fue el modelo con mejor desempeño, también es importante destacar al *Decision Tree* ya que no difieren sus métricas de forma significativa de este.

Otro de los factores que se debe de considerar para la mejora de los modelos, es la posibilidad de recolectar datos reales de las categorías que en este caso se generaron de forma sintética, esto favorecería al modelo facilitándole la identificación de cada una de las reseñas.

VII. Conclusión

El texto generado día con día a través de las diferentes fuentes de información es un área de oportunidad la cual no se puede desperdiciar. El poder clasificar de forma automática reseñas sin importar cuales sean sus categorías es un avance significativo para la toma de decisiones.

El desarrollo de aplicaciones móviles va en aumento, las tendencias se van orillando a poder tener todo tipo de software en nuestros dispositivos móviles inteligentes. El desempeño que están teniendo dichas aplicaciones es de interés para la compañía de estas. El identificar áreas de oportunidad de mejora para la aplicación basada en sus reseñas, es una ventaja sobre la competencia.

La intención con la metodología propuesta era el poder adecuar las distintas fases del clásico procesado de texto, hacia las reseñas de aplicaciones móviles. Entre estas fases destacan el preprocesado de emojis, los cuales son comunes en la redacción de reseñas hoy en día, la extracción de características con el método TF-IDF, la creación con datos sintéticos con el algoritmo SMOTE, el cual como mencionamos en la discusión sería conveniente el poder recolectar datos reales de las demás clases que no se acercan a la clase mayoritaria para de esta manera alimentar el modelo.

Los modelos más populares en la clasificación de texto, de acuerdo con la literatura, utilizados en este artículo que en este caso fueron el *Multinomial Naive Bayes* y el *Decision Tree*, fueron superados por el modelo de *Random Forest*, con un 73% de *accuracy*. Dichos modelos obtienen buenos resultados con la clasificación binaria, sin embargo, en este caso de estudio teníamos múltiples clases.

El estudio de los modelos evaluados en este artículo nos dirige hacia un futuro ajuste, para mejorar el desempeño de estos de acuerdo con el contexto el cual en este caso es orientado a reseñas de aplicaciones móviles. Se considera conveniente el poder experimentar con la fase de preprocesado, para poder alcanzar mejores resultados. El probar distintos métodos de extracción de características, la recolección de datos de las clases minoritarias, el poder estructurar nuestras 5 clases para volverlo un problema de clasificación binaria ya que tenemos evidencia que los modelos *Naive Bayes* y el *Decision Tree* obtuvieron buenos resultados en el artículo de clasificación de malas reseñas. [7]

Este artículo es la base de clasificación automática de reseñas de aplicaciones móviles, para trabajos futuros una vez clasificando nuestras reseñas podemos basarnos en las malas calificaciones de nuestra aplicación móvil e identificar palabras claves, incluso clasificar dichas malas reseñas en peticiones del usuario, experiencia del usuario, reporte de problemas, entre otras categorías. Teniendo esta clasificación de malas reseñas es más claro para la compañía el poder atender problemas o peticiones relevantes que su comunidad le está reportando.

References

[1] IFT. (22 Junio 2021). EN MÉXICO HAY 84.1 MILLONES DE USUARIOS DE INTERNET Y 88.2 MILLONES DE USUARIOS DE TELÉFONOS CELULARES: ENDUTIH 2020. 18 de julio 2022, de IFT Sitio web: https://www.ift.org.mx/comunicacion-y-medios/comunicados-ift/es/en-mexico-hay-841-millones-de-usuarios-de-internet-y-882-millones-de-usuarios-de-telefonos-celularesratings

[2] David Reinsel, John Gantz, John Rydning. (Abril 2017). The Evolution of Data to Life-Critical. 18 de Julio 2022, de IDC Sitio web: https://www.import.io/wp-content/uploads/2017/04/Seagate-WP-DataAge2025-March-2017.pdf

[3] Dalal, M. K., & Zaveri, M. A. (2011). Automatic text classification: a technical review. International Journal of Computer Applications, 28(2), 37-40.

[4] Aggarwal, C. C., & Zhai, C. (2012). A survey of text classification algorithms. In Mining text data (pp. 163-222). Springer, Boston, MA.

[5] Ikonomakis, M., Kotsiantis, S., & Tampakas, V. (2005). Text classification using machine learning techniques. WSEAS transactions on computers, 4(8), 966-974.

[6] Kowsari, K., Jafari Meimandi, K., Heidarysafa, M., Mendu, S., Barnes, L., & Brown, D. (2019). Text classification algorithms: A survey. Information, 10(4), 150.

[7] Maalej, W., Kurtanović, Z., Nabil, H., & Stanik, C. (2016). On the automatic classification of app reviews. Requirements Engineering, 21(3), 311-331.

[8] SANSKAR HASIJA. (2022). Top 20 Play Store App Reviews (Daily Update). 14 de Julio 2022, de Kaggle Sitio web: https://www.kaggle.com/datasets/odins0n/top-20-play-store-app-reviews-daily-update?select=all_combined.csv

[9] Frunza, O., Inkpen, D., & Matwin, S. (2010, August). Building systematic reviews using automatic text classification techniques. In Coling 2010: Posters (pp. 303-311).

[10] Adeva, J. G., Atxa, J. P., Carrillo, M. U., & Zengotitabengoa, E. A. (2014). Automatic text classification to support systematic reviews in medicine. Expert Systems with Applications, 41(4), 1498-1508.

[11] Melville, P., Gryc, W.,& Lawrence, R. D. (2009, June). Sentiment analysis of blogs by combining lexical knowledge with text classification. In Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining (pp. 1275-1284).

[12] Irving Estrada. Github. 2022,Sitio web: https://github.com/Irving-Estrada/Procesamiento