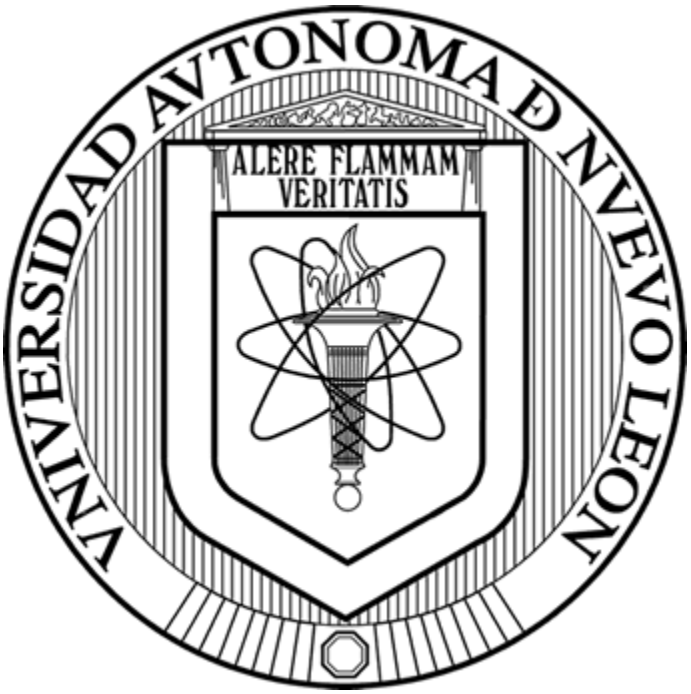
UNIVERSIDAD AUTÓNOMA DE NUEVO LEÓN

FACULTAD DE CIENCIAS FÍSICO-MATEMÁTICAS



REPORTE II

PREDICCIÓN EN BASES DE DATOS

Profesora: Mayra Cristina Berrones Reyes

Autor: Juan de Jesús Aguilar Solano Matrícula: 1576327

Jueves 26 de mayo de 2022

Contents

TABLA DE FIGURAS

INTRODUCCIÓN

El uso de técnicas de Machine Learning para tareas de clasificación es adecuado siempre y cuando se tenga una base de datos etiquetada, lo cual en ocasiones resulta una tarea bastante tediosa. En este reporte se describirán 2 técnicas de Machine Learning, el clasificador Naive Bayes y Random Forest como un clasificador de sentimientos sobre reseñas dadas a un hotel. Se pondrá a prueba estos dos modelos y su desempeño como clasificador, utilizando diferentes parámetros de evaluación.

EXPERIMENTACIÓN

En este proyecto se utilizará la base de datos obtenida del sitio web kaggle [1] y que fueron extraídos mediante la API de twitter, esta base de datos contiene 515,738 reseñas de clientes hospedados en 1492 hoteles diferentes.

Diccionario de datos

- Hotel_Address: Dirección del hotel.
- Review_Date: Fecha en la que el usuario realizó la reseña.
- Average_Score: Calificación media del hotel calculada en base en los últimos comentarios del año.
- Hotel_Name: Nombre del Hotel.
- Reviewer_Nationality: Nacionalidad del usuario que realizó la reseña.
- Negative_Review: La reseña negativa que el usuario ha brindado al hotel, si no brindó reseña negativa, el campo contendrá la cadena “No Negative”.
- Review Total/NegativeWordCounts: El número de palabras negativas en la reseña.
- Positive_Review: La reseña positiva que el usuario ha brindado al hotel, si no brindó reseña positiva, el campo contendrá la cadena “No positive”.
- ReviewTotal/PositiveWordCounts: Número total de palabras positivas en la reseña.
- Reviewer_Score: La calificación que el usuario ha dado al hotel basado en su experiencia.
- TotalNumberofReviewsReviewerHasGiven: Número de reseñas que los usuarios han dado en el pasado.
- TotalNumberofReviews: El número total de reseñas válidas que el hotel tiene.
- Tags: Tags de las reseñas dadas al hotel.
- AdditionalNumber_Scoring: Indica cuantas calificaciones sin reseña ha brindado.
- lat: Latitud del hotel.
- lng: Longitud del hotel.

Lectura de datos y procesamiento básico

Para la lectura de datos se utilizó la biblioteca pandas, haciendo uso del método read_csv(), de todos los atributos antes descritos se han eliminado todas las columnas a excepción de las columnas “Negative_Review”, “Positive_Review” y “Reviewer_Score”.

Como primera fase de procesamiento, manejaremos los valores nulos, en esta base de datos las reseñas contienen celdas nulas, dichas celdas contienen los valores “No Positive” para denotar que no se brindó una reseña positiva y “No Negative” para denotar que no se brindó una reseña negativa, el manejo consiste en sustituir estos valores por una cadena de texto vacía.

Dado que queremos clasificar el sentimiento de la reseña basada en el texto que introdujo el usuario, se ha decidido fusionar las columnas “Negative_Review” y “Positive_Review”, en una sola columna llamada “Full_Review”, dicha columna será nuestros datos de entrada, por lo que las columnas “Negative_Review” y “Positive_Review” se eliminarán del set de datos.

Ahora necesitamos el etiquetado de la reseña completa, para hacer esto se tomó en cuenta la calificación otorgada por el usuario al Hotel, tomando como umbral de clasificación, una calificación de 7.0, es decir, si la calificación fue mayor a 7.0, la reseña será etiquetada como positiva, caso contrario será negativa, el resultado será guardado en una columna llamada “label” y la columna “Reviewer_Score” será eliminada del conjunto.

Ahora el set de datos consiste en solo dos columnas

- Full_Review: contiene todo el texto introducido por el usuario (tanto la reseña positiva como la negativa).
- Label: contiene la etiqueta del sentimiento del texto.

Preprocesamiento de texto

En la siguiente fase del procesamiento, procedemos a manejar el texto contenido en “Full_Review” de la siguiente manera:

- Convertir todos los caracteres a minúsculas.
- Remover signos de puntuación y caracteres especiales, los cuales son los siguientes:

[¿, ?, ¡, ‘, “, #, :, “, ”, _, ., ,), (, \, /, *, –, $,%,&,+,;,<,=,>,@,[,],^_,`,{,|,},~]

- Normalizar los espacios, es decir, el espaciado doble, triple, cuádruple, etc. Es sustituido por un espaciado simple.
- Separar la cadena de texto en palabras y esta es guardada en una lista, tomando como referencia de separación el espaciado simple.
- Removemos las “stopwords” de la lista de palabras, las “stopwords” se obtuvieron de la biblioteca nltk [3].
- Stemming de las palabras. Este proceso se lleva a cabo mediante el SnowBall Stemmer, también contenido en la biblioteca nltk tomando en cuenta el contexto de la palabra.

Función de preprocesamiento

Todos los pasos descritos durante este apartado se introdujeron en una sola función llamada preprocesamiento_words() y lemmatize(), se muestra a continuación el antes y después de la función:

```
Texto de entrada:
https://sitioweb.com Nombre: Juan, Correo: juan.agsolano@gmail.com

Lista de salida
['nombr', 'juan', 'correo', 'juan', 'agsolano', 'gmail', 'com']
```

El resultado de la función es una lista de palabras ya procesadas contenidas en jupyter notebook adjunto en este proyecto.

Vectorización de las palabras y binarización

Durante esta parte del proceso se realiza un conteo de todas las palabras involucradas en todo el set de datos, se tomó como criterio, obtener las 1200 palabras más frecuentes en el set de datos. Una vez hecho esto, se obtiene una matriz con la frecuencia de éstas 1200 palabras presentes en cada reseña brindada por el usuario. Todo esto es realizado mediante la función CountVectorizer contenida en la biblioteca scikitlearn de Python.

Tabla 1. Matriz vectorizada con las 1200 palabras más frecuentes

abl	absolut	ac	accept	access	...	would	wrong	year	yes	yet	young
0	1	0	0	0	...	0	1	0	0	0	0
1	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

5 rows × 1200 columns

Esta matriz de palabras será sustituida por la columna “Full_Review”, y por último se sustituirán las etiquetas por valores binarios, considerando la etiqueta “Negative” = 0 y “Positive” = 1, por lo que ahora tendremos un set de datos con 1201 columnas (contando el etiquetado) la cual está listo para ser procesado por un clasificador.

Clasificadores Naive Bayes y Random Forest

El primer clasificador que abordaremos será el clasificador Gaussian Naive Bayes, el cual utiliza el teorema de bayes para realizar predicciones. Para explorar un poco este clasificador, se procedió a variar algunos parámetros los cuales son “tamaño de muestra” y “numero de atributos” para determinar estos parámetros óptimos.

El segundo clasificador es Random Forest, los hiperparámetros utilizados en un número de árboles = 200, utilizando el índice de Gini como umbral de nodo y un mínimo de 20 datos por hoja. Además utilizamos la información provista por el clasificador Naive Bayes como el tamaño de datos y número de atributos utilizados.

Para Ambos modelos se obtuvieron sus respectivas matrices de confusión, así como la evaluación de diferentes métricas para evaluar el desempeño de cada uno y poder compararlos.

RESULTADOS

Como primer resultado podemos mostrar una nube de palabras, la cual contiene las 200 palabras más utilizadas en todas las reseñas del set de datos.

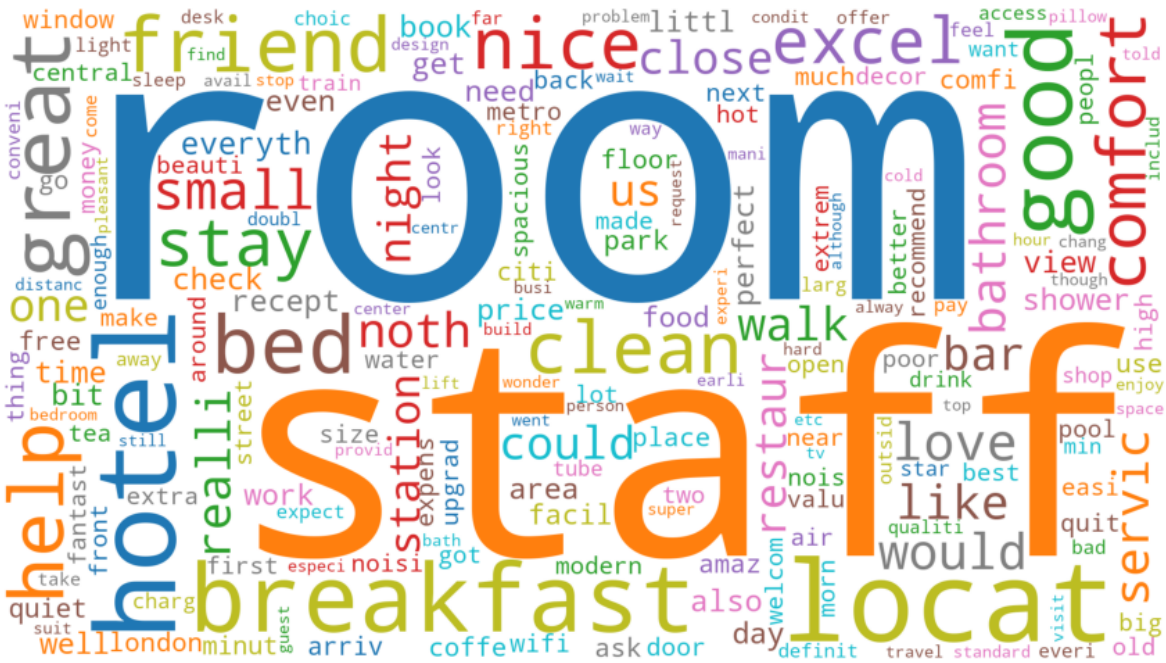


Ilustración 1. Wordcloud de las 200 palabras más importantes en los comentarios.

De la nube de palabras mostrada podemos observar que en su mayoría se ven palabras positivas, otras haciendo referencia al contexto del hotel y algunas negativas, por lo que sería razonable considerar que las reseñas sean positivas.

Clasificador Naive Bayes

Para el entrenamiento de este modelo se decidió utilizar un subconjunto de 20,000 datos, esto para establecer un límite mínimo de datos que puede ser utilizado para entrenar el modelo. Además se para lidiar con el balance de las etiquetas, se obtuvieron 10,000 observaciones positivas y 10,000 negativas, de modo que ahora la base de datos a utilizar se encuentra balanceada.

De la muestra de datos tomada, se utilizará el 80% para entrenar el modelo y el 20% restante para evaluar el desempeño del modelo, los tamaños de muestra quedarían de la siguiente manera.

Tabla 2. Tamaño de muestra para entrenamiento del modelo.

	Tamaño
Entrenamiento	16,000
Prueba	4,000
Total	20,000

Curvas de aprendizaje – Tamaño de muestra

Para realizar las siguientes curvas se procedió a entrenar el modelo con diferentes tamaños de muestra y utilizando un Cross Validation con k = 5, utilizando F1 score como métrica de rendimiento. Los tamaños de muestra a evaluar serían de [1600, 2400, 3200, 4000, 4800, 5600, 6400, 7200, 8000, 8800, 9600, 10400, 11200, 12000, 12800]. El resultado final se muestra en el siguiente gráfico.

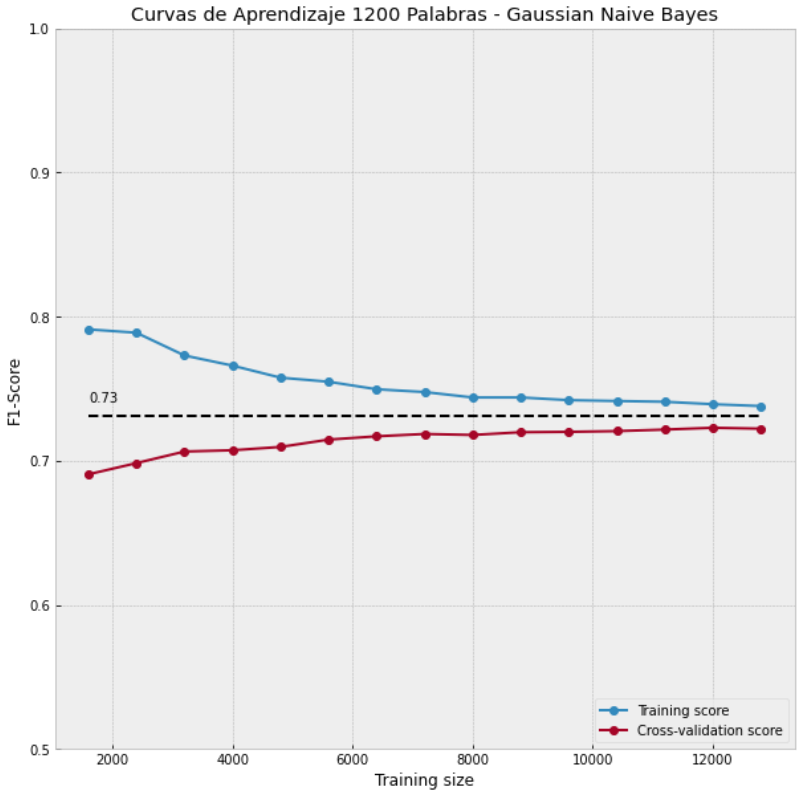


Ilustración 2. Curvas de aprendizaje diferentes tamaños de muestra.

Como puede verse a continuación, la métrica converge en 0.73, donde además podemos observar que el ajuste mejora gradualmente conforme aumenta el tamaño de muestra de entrenamiento, sin embargo, podemos ver que a partir de un tamaño de 8,000 observaciones, el entrenamiento mejora solo un poco, de modo que podemos elegir un tamaño de muestra adecuado en base a nuestras necesidades.

Desempeño sobre todos los datos

El desempeño sobre todos los datos nos da un f1 score de 0.86, es decir, el desempeño es mejor, esto puede deberse a que la base de datos original contiene una mayor cantidad de reseñas positivas que negativas, por lo que podemos sospechar que nuestro modelo es mejor clasificando verdaderos positivos.

Curvas de aprendizaje – Número de atributos

Las siguientes curvas de aprendizaje tienen la finalidad de evaluar la cantidad de atributos relevantes que fungen para clasificar los textos dados, el resultado se muestra a continuación.

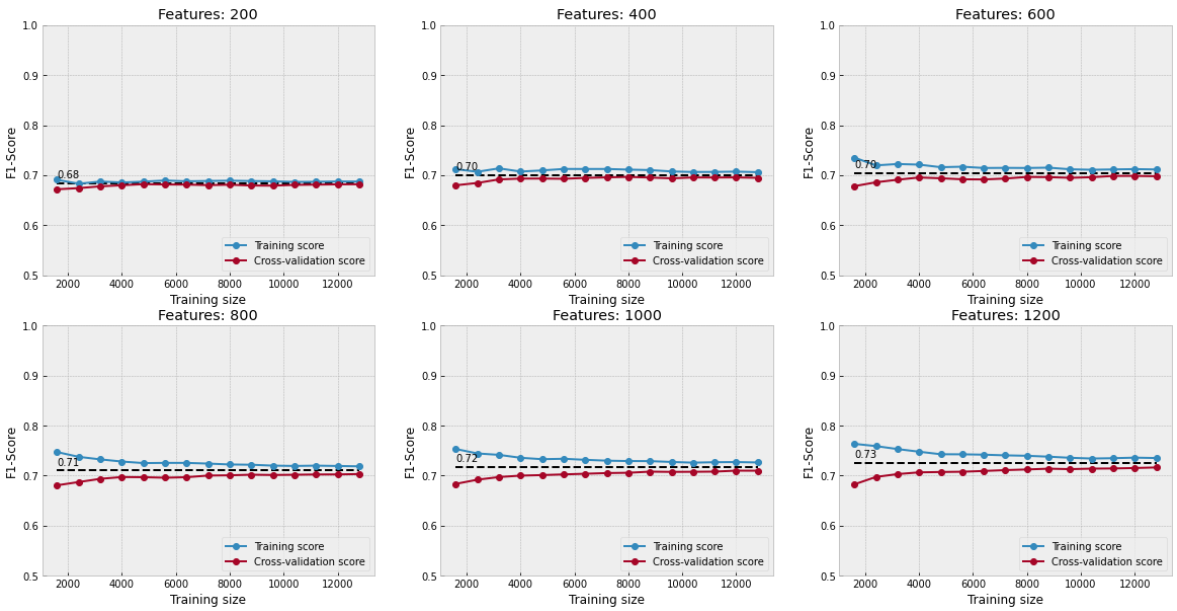


Ilustración 3. Curvas de aprendizaje para diferente tamaño de atributos.

Se puede hacer notar que, al aumentar la cantidad de atributos, el desempeño mejora un poco, estas curvas podemos simplificarlas a continuación.

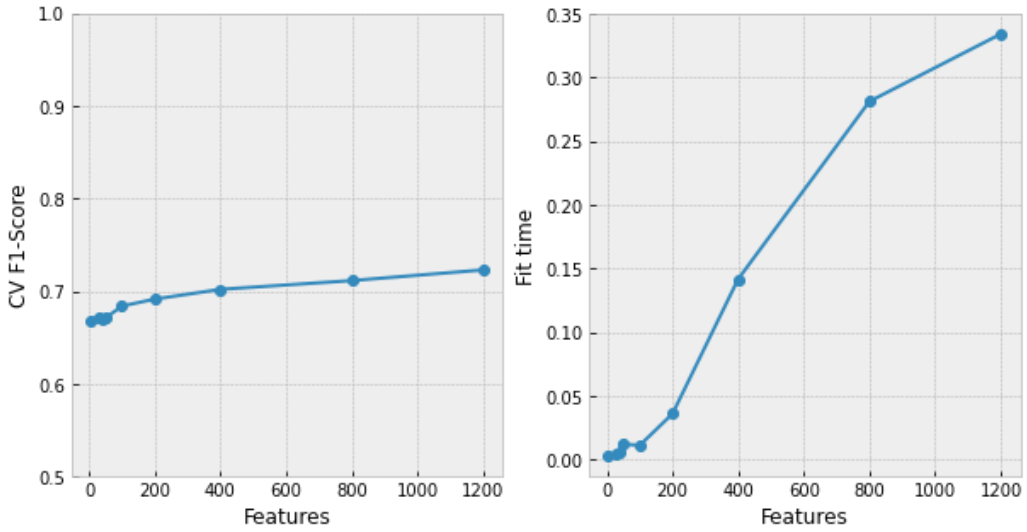


Ilustración 4. Desempeño (F1 score) y tiempo de entrenamiento.

A mayor cantidad de atributos, el desempeño mejora, sin embargo, puede verse que a partir de 400 atributos, el crecimiento del rendimiento es lineal pero muy bajo, mientras que el tiempo de entrenamiento si aumenta considerablemente a partir de los 400 atributos, por lo que para tomar una decisión debe tomarse en cuenta el contexto del modelo y así determinar si el score es mejor que el tiempo de entrenamiento o bien buscar un balance entre ambos.

Matriz de confusión y scores

A continuación, se muestra la matriz de confusión para el modelo entrenado con un tamaño de entrenamiento 12,000 muestras y el resultado se muestra a continuación.

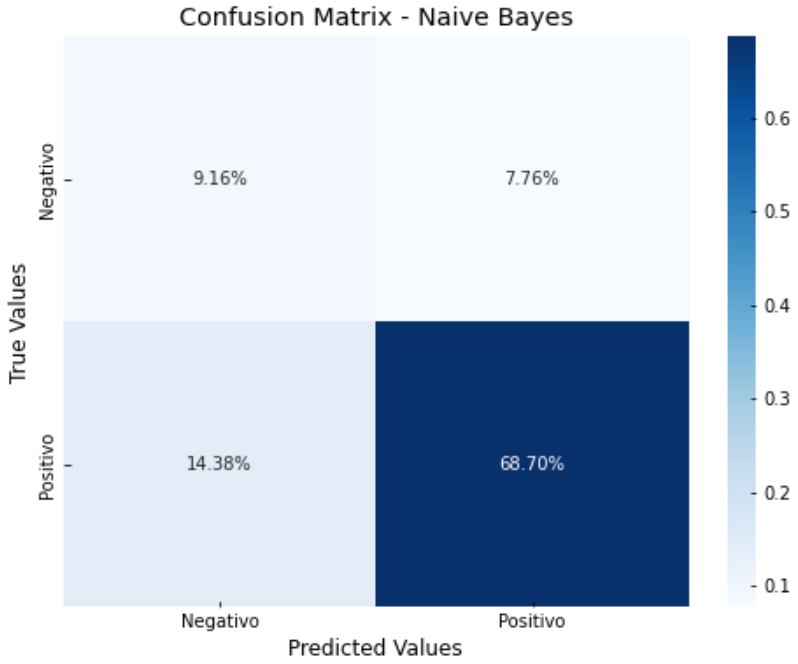


Ilustración 5. Matriz de confusión para el modelo Naive Bayes

Como puede observarse en este mapa de calor, un mayor porcentaje de datos clasificados se encuentra en los verdaderos positivos, esto se debe a que la base de datos está desbalanceada, mientras qué Falsos positivos son los que también aportan un 15% de error a nuestro modelo, por lo que vemos, este clasificador funciona bien con los positivos, mientras que con los negativos puede mejorarse un poco más.

Métricas de rendimiento

A continuación, se resumen las métricas evaluadas para este modelo.

- Accuracy: 77.68%
- Precision: 89.65%
- Recall: 82.68%
- F1 score: 86.03%

Podemos concluir que el modelo tiene un mejor rendimiento bajo la métrica de Precision, lo cual significa que predice bastante bien las reseñas positivas.

Clasificador Random Forest

Para el clasificador Random Forest se utilizó la misma cantidad de datos (20,000) y seleccionando datos balanceados para entrenar el modelo.

Matriz de confusión y scores

Los parámetros utilizados para este modelo se describieron en la sección anterior y los resultados de éste se muestran a continuación.

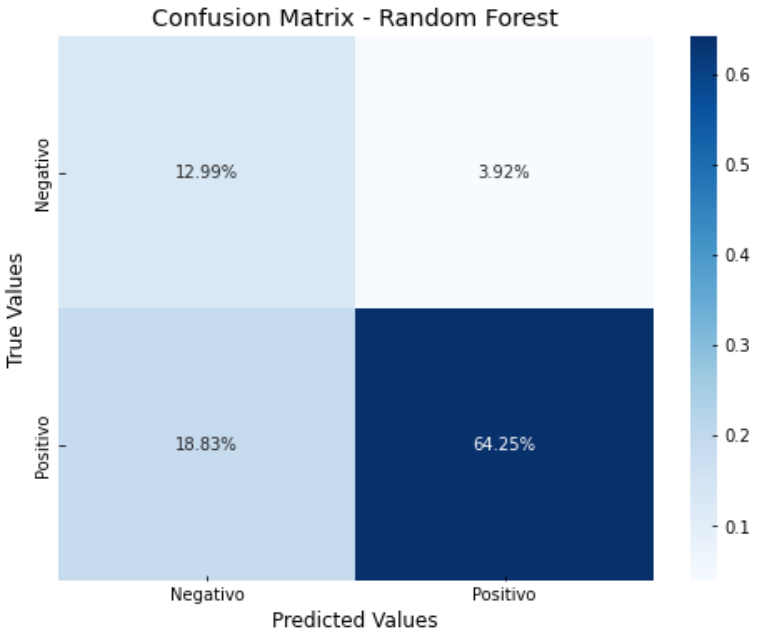


Ilustración 6. Matriz de confusión para el modelo Random Forest.

Como puede observarse en este mapa de calor, un mayor porcentaje de datos clasificados se encuentra en los verdaderos positivos, y al igual que con el clasificador Naive Bayes, esto se debe a que la base de datos está desbalanceada, sin embargo, a diferencia del modelo anterior, este presenta un mayor porcentaje de Falsos negativos y un porcentaje muy bajo de Falsos positivos.

Métricas de rendimiento

A continuación, se resumen las métricas evaluadas para este modelo.

- Accuracy: 77.25%
- Precision: 94.24%
- Recall: 77.34%
- F1 score: 84.96%

Podemos concluir que el modelo tiene un mejor rendimiento bajo la métrica de Precision y es mayor que el modelo Naive Bayes, sin embargo, en las demás métricas tiende a ser un poco más bajo su rendimiento.

Importancia de atributos

El algoritmo de Random Forest nos permite obtener la importancia de los atributos que contribuyen a la clasificación del modelo, a continuación, se muestran las 20 palabras con mayor peso de este modelo.

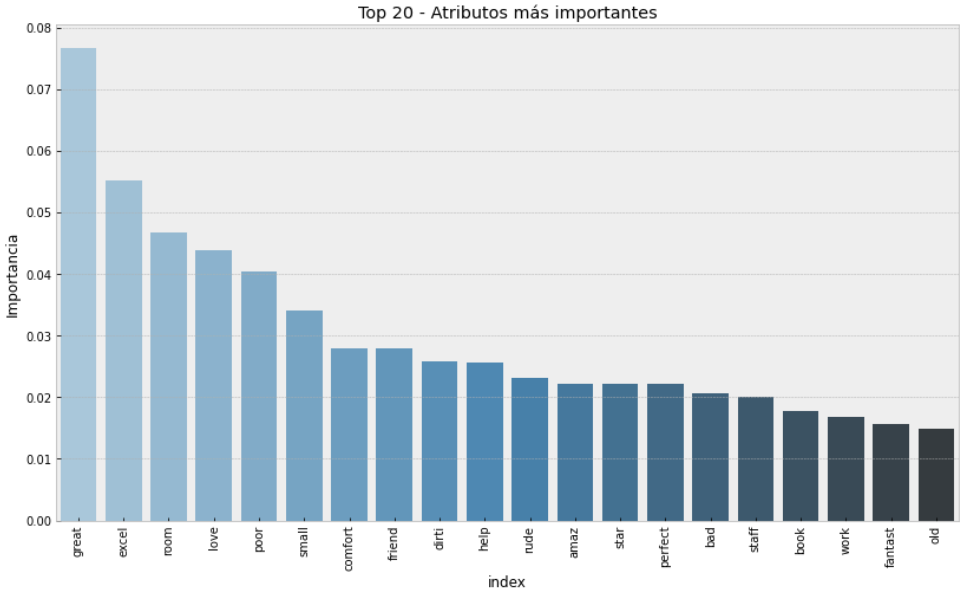


Ilustración 7. Atributos más importantes para el modelo Random Forest.

De donde se puede ver que la palabra great, es aquella con mayor peso a la hora de clasificar las reseñas, suena lógico mencionar que esta palabra funciona muy bien para clasificar las reseñas positivas ya que está contenida en gran parte de éstas reseñas.

Comparación de modelos

Las comparaciones entre modelos pueden mostrarse en la siguiente tabla.

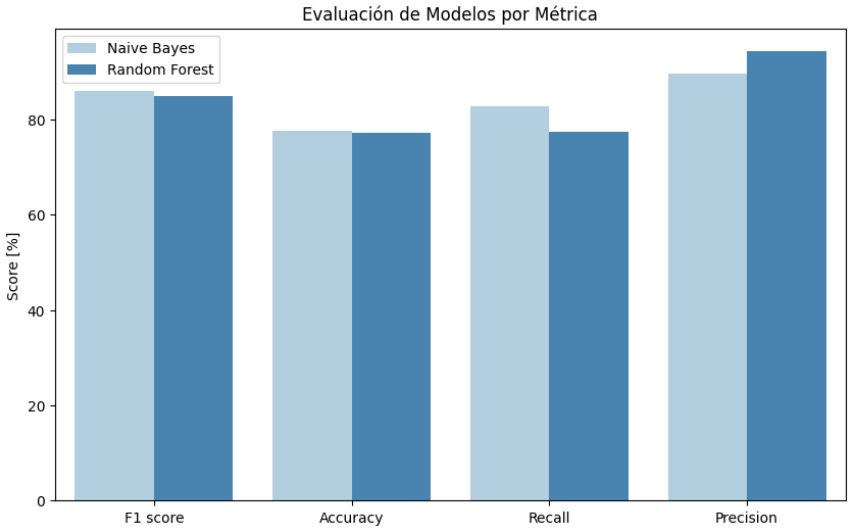


Ilustración 8. Comparación de modelos con diferentes métricas.

Como comparación general podemos observar que en todas las métricas a excepción de Presicion, el clasificador Naive Bayes es ligeramente superior, sin embargo para precisión, Random Forest es un modelo que lo supera por 5 puntos porcentuales, poniéndolo en un nivel mayor con respecto al clasificador Naive Bayes.

CONCLUSIÓN

El uso de modelo clasificación para la predicción de bases de datos resulta bastante conveniente cuando se tiene datos etiquetados, sin embargo, debe tenerse cautela cuando los datos no están balanceados, de ser así, buscar la mejor manera de abordar esta problemática, en nuestro caso dado que la muestra es bastante grande, se decidió crear un subconjunto con datos balanceados.

Para el procesamiento de los datos se abordaron pasos estándar para limpiar el contenido de texto, además para este proyecto se decidió usar un stemming de palabras para alimentar los algoritmos, esto debido a que el modelo no identificará el sentimiento de las palabras, solo la asociación que tiene hacia comentarios positivos.

En cuanto a los modelos se puede ver que el clasificador Naive Bayes resulta mucho más simple de implementar ya que no requiere de hiperparámetros para funcionar, además de que muestra un buen desempeño para este tipo de aplicaciones, por lo que resulta fácil encontrar los atributos y tamaños de muestra óptimos para poner este clasificador en marcha. El algoritmo de Random Forest es un poco más complicado, se tienen que cuidar y variar varios hiperparámetros para encontrar el mejor modelo y el resultado es bastante similar al de Naive Bayes, pero este modelo tiene características que lo vuelven bastante interesante a la hora de implementarlo, como lo es la extracción de atributos importantes para el modelo y así obtener conocimientos valiosos que pueden ser usados para el contexto del problema.

Como conclusión, puede decirse que ambos clasificadores dan resultados positivos bajo las condiciones dadas, siendo el Naive Bayes mucho más sencillo de implementar a nivel operativo, pero random forest es más flexible a la hora de personalizarlo y así poder extraer conocimientos valiosos de los datos.

Como comentario final, los clasificadores pueden mejorarse si añadimos más atributos, probablemente si agregamos el número de palabras contenidas en la reseña positiva y negativa nos brinde mayor información acerca del sentimiento de la reseña.

BIBLIOGRAFÍA

[1]. J. (2018, 18 diciembre). *Sentiment analysis with hotel reviews*. Kaggle. Recuperado 26 de mayo de 2022, de https://www.kaggle.com/code/jonathanoheix/sentiment-analysis-with-hotel-reviews/data

[2]. Go, A., Bhayani, R. and Huang, L., 2009. *Twitter sentiment classification using distant supervision*. *CS224N Project Report, Stanford, 1(2009), p.12*.

[3]. *Sentiment Analysis | Lexalytics*. (s. f.). Lexalytics. Recuperado 26 de mayo de 2022, de https://www.lexalytics.com/technology/sentiment-analysis

REPOSITORIO

Los archivos y códigos se encuentran en el repositorio cuyo enlace es:

https://github.com/juanagsolano/Procesamiento-y-clasificaci-n-de-datos