Informe Proyecto1-Entrega1 Inteligencia de Negocios

Juan David Martínez Moreno - 201923416

Johan Sebastián Cáceres Charari - 202014171

Universidad De Los Andes

Departamento De Ingeniería De Sistemas Y Computación

02 de abril del 2023

0)Trabajo en equipo

Asignación de tareas

Tarea	Persona	Tiempo Asignado	Tiempo Dedicado	Puntos de esfuerzo sobre 100
Limpieza y aplicación de Asum-DB	Juan David Martinez	**3 horas**	**3 horas**	**15**
Tokenizacion y lemmatizacion	Johan Sebastian Caceres	**4 horas**	**4 horas**	**20**
RandomForest	Johan Sebastian Caceres	**2 horas**	**2 horas**	**20**
Decision Tree	Johan Sebastian Caceres **/** Juan David Martínez	**2 horas**	**2 horas**	**20**
Regresión logística	Juan David Martínez	**3 horas**	**3horas**	**25**

Roles asignados

Rol	Persona
Lider de proyecto	Johan Sebastian Caceres
Líder negocio	Juan David Martínez
Líder de datos	Juan David Martínez
Líder de analítica	Johan Sebastian Caceres

Nombre	Retos enfrentados	Formas de resolverlo
Johan Sebastian Caceres	Tokenizacion y lemmatizacion de reseñas que se encontraban en diferentes idiomas.	Hacer uso de las dos librerías de stopwords y detectar el lenguaje en el que estaba escrita cada reseña y de acuerdo a ello darle un tratamiento específico.
Juan David Martínez	Investigación e implementación de un nuevo algoritmo que no se vio en clase: Logistic Regression	Revisar ejemplos en internet sobre la implementación de este algoritmo.

Reuniones Realizadas

Tema	Fecha
Reunión lanzamiento y planeación del proyecto	22 de marzo
Reunión de seguimiento	31 de marzo
Reunión de seguimiento	1 de octubre

Reunión de finalización	2 de octubre

1) Entendimiento del negocio y enfoque analítico

Oportunidad/problema del negocio	**Se requiere de una forma de clasificación del texto de los comentarios recibidos para un conjunto de hoteles, lo anterior debido a que al negocio le interesa tener la capacidad de tomar decisiones a partir de la clasificación de las diferentes revisiones que se reciben por parte de los clientes.**
Enfoque analítico (Descripción del requerimiento desde el punto de vista de aprendizaje automático)	La posibilidad de construir un modelo con la capacidad de recibir y procesar las opiniones sobre los hoteles para generar una clasificación automática de la opinión en general de esa persona sobre un hotel por medio del análisis de sentimientos. Esto permite que, al recibir reseñas sobre los hoteles, las palabras de los clientes puedan ser traducidas en una escala que permita a la organización sacar provecho en pro de mejorar servicios.
Organización y rol dentro de ella que se beneficia con la oportunidad definida	Los hoteles que reciben las reseñas ahora podrán saber a partir de un texto registrado por un cliente, si el mismo tuvo una buena experiencia o mala con el hotel, lo anterior permite a departamentos como el de marketing, servicio al cliente y directivos identificar los clientes insatisfechos y poder tomar decisiones frente a las disconformidades que los mismos tienen en orden de minimizar la cantidad de clientes insatisfechos con el hotel.
Técnicas y algoritmos a usar	La aproximación para conseguir el modelo que satisfaga el enfoque será la técnica de aprendizaje supervisada de clasificación, ya que se cuenta con una cantidad importante de reseñas etiquetadas con un valor que representa la opinión del cliente (0:Mala, 1: Buena), en base a estas etiquetas se generará el modelo que será evaluado por medio de tres algoritmos: árboles de recubrimiento, logistic Regresión y random Forest. Lo anterior, implica que antes de la ejecución del modelo se realizará una limpieza de datos que permitirá garantizar los requerimientos ASUM-DB (consistencia, validez, completitud, nulidad) y se realizará un proceso de tokenización que permitirá entrenar un modelo capaz de reconocer palabras positivas y negativas dentro del modelo, teniendo en cuenta que existen palabras que no tienen peso, o que tienen la misma raíz dentro de la ejecución y realizando el tratamiento a estas.

2) Entendimiento y preparación de los datos:

2.1) Carga y verificación de requerimientos ASUM-DB

Primero que todo, se realizó un entendimiento de datos con base en la selección de *features* que van a ser importantes y que aportan valor en el entrenamiento del modelo que se va a utilizar. Los datos que fueron proporcionados por el negocio cuentan con: “id”, “title”, “rating”, “review_text”, “location”, “hotel” y “label”.

Como se llevó a cabo un análisis de sentimientos sobre las reviews que recibieron los hoteles, no se van a tener en cuenta las columnas que tienen datos que no contengan texto y que no aporten a la clasificación de sentimientos en la reseña. En este caso, por ejemplo: “id” es numérica y no aporta valor importante para decidir el sentimiento de una review; “location” y “hotel” indican la locación y el nombre del hotel, respectivamente, sin embargo, el uso de estos no aportará valor significativo al modelo; “review_text” y “title” serán los textos que se usarán para tokenizar y, posteriormente, usados para el entrenamiento del modelo.

- Completitud:

Se revisó si existen valores vacíos dentro de los datos proporcionados por el negocio, pero todo estaba con su valor respectivo, por lo que no se necesitó de completar ni eliminar ningún registro.

- Unicidad:

Se encontraron registros duplicados en el dataframe asociado los datos proporcionados. Por lo que se procedió a eliminar los registros que estaban duplicados y dejar una muestra de uno de esos dentro de los datos de entrenamiento para el modelo.

- Categorización de la variable objetivo (“label”)

Por cuestión del objetivo que el negocio desea alcanzar, como lo es la mejora y perfeccionamiento de los servicios proporcionados por el mismo, se desea que la variable objetivo solamente contenga dos categorías en las cuales se pueden dividir las reviews: buena (1) o mala (0). Para esto, los valores de “label” se verán afectados directamente por el “rating” asociado en su registro. De esta manera, cuando se encuentre un registro el cual tenga valores en el “label” de neutro (3), se hará una transformación de 1 a 0 dependiendo del valor del “rating”, el cual va de 1 a 5.

Por lo tanto, las condiciones de transformación de estos registros en específico son las siguientes:

Si el rating es <= 3, el “label” pasará a ser malo (0).

Si el rating es > 4, el “label” pasará a ser bueno (1).

2.2 Tokenización

Después de la revisión y tratamiento de los requerimientos Asum-Db se realizó un proceso de tokenización, en primer lugar, se concatenaron las columnas ‘review_text’ con ‘title’, ya que el título de cada reseña contenía información relevante para la determinación de si la reseña era positiva o negativa. A continuación, se realizó la concatenación, se dividieron los datos del dataframe

original en dos grupos, uno que permitiría entrenar el modelo para la clasificación y el otro destinado para el testeo del modelo generado por los datos de entrenamiento, para ello se hizo uso de la función train_test_split que recibe como parámetro test_size, que se estableció con un valor de 0.2, lo que implica que de la muestra original etiquetada se reservaran el 20% de las entradas para el testeo y el 80% restante para el entrenamiento de los diferentes modelos que se construirán. Dentro de la función train_test_plit el parámetro Stratify permite asegurar que la división de los datos sea muy similar para ambos conjuntos y las pruebas puedan tener mayor posibilidad de generar buenos resultados.

Los tamaños de las muestras de entrenamiento y testeos quedaron de la siguiente proporción:

→ Tamaño de X_train: (3598,)
→ Tamaño de X_test: (900,)
→ Tamaño de y_train: (3598,)
→ Tamaño de y_test: (900,)

La división se realizó línea 16 del notebook asociado

Después de obtener los conjuntos de entrenamiento y testeo se realizó la definición de los stopwords, como se identificaron que había reseñas en dos idiomas (español e inglés) se decidió que por medio de la librería ntlk, que permite identificar los stopwords (preposiciones y palabras que no aportan valor al modelo), se procedió a identificar y concatenar en un solo conjunto las stopwords en español e inglés. Este tratamiento se realizó en la línea 18 de código asociado al notebook entregado.

Tokenizar es el proceso de dividir un texto en unidades más pequeñas llamadas tokens. Los tokens suelen ser palabras, pero también pueden ser frases o símbolos. El objetivo de tokenizar es dividir el texto en unidades más manejables para su análisis posterior. Por ejemplo, en el procesamiento del lenguaje natural, tokenizar un texto es un paso común antes de realizar tareas como el análisis de sentimientos o la extracción de información.
Lematizar es el proceso de reducir una palabra a su forma base o raíz. Por ejemplo, las palabras “corriendo”, “corre” y “corrió” se reducen a su forma base “correr” después de lematizar. El objetivo de lematizar es agrupar palabras similares para que puedan ser analizadas como una sola entidad.

Para tokenizar y lematizar, se creó una funcion tokenizer que recibe el texto asociado a una reseña y tokeniza y lematiza el texto. El tokenizador se crea utilizando la clase RegexpTokenizer de la biblioteca nltk con un patrón de expresión regular que coincide con uno o más caracteres de palabra. El lematizador se crea utilizando la clase WordNetLemmatizer de la biblioteca nltk. El código también define dos funciones: get_wordnet_pos y tokenizer. La función get_wordnet_pos asigna etiquetas de parte del discurso de la biblioteca nltk a

etiquetas POS de WordNet. La función tokenizer toma texto como entrada y lo tokeniza utilizando la función nltk.word_tokenize. Luego elimina los caracteres no alfanuméricos de los tokens, cambia los números por su forma escrita por medio de la librería n2words y los lematiza utilizando la función lemmatizer.lemmatize. El idioma del texto se detecta utilizando la función detect de la biblioteca langdetect y se inicializa un corrector ortográfico en función del idioma detectado. Si el idioma es español, el texto se tokeniza utilizando la opción de idioma español para la función nltk.word_tokenize.

TF-IDF y bow son técnicas de ponderación que se utilizan a menudo en el modelo de bolsa de palabras para determinar la importancia de una palabra en una review en relación con una colección de reviews.
Para este proyecto se decidió hacer uso de la ponderación TF-'DF por las siguientes razones:

- TF-IDF tiene en cuenta la frecuencia de las palabras en todo el corpus, mientras que BoW solo tiene en cuenta si una palabra aparece o no en un documento. Esto significa que TF-IDF puede dar mayor importancia a las palabras que son raras en todo el corpus, pero son muy importantes en un documento en particular.
- TF-IDF es útil cuando se trabaja con un corpus grande y diverso, ya que permite identificar las palabras clave que se destacan en el corpus. Por otro lado, BoW es menos útil en este contexto, ya que simplemente proporciona una representación binaria de la presencia o ausencia de palabras.
- TF-IDF se puede utilizar para clasificar documentos o para recuperar documentos relevantes. Debido a que las palabras que son importantes en un documento tienen un puntaje TF-IDF alto, podemos usar esta medida para encontrar documentos similares en el corpus.

TF-IDF es el producto de dos medidas: la frecuencia del término (TF) y la frecuencia inversa del documento (IDF). La frecuencia del término mide la frecuencia de una palabra en una review específica, mientras que la frecuencia inversa del documento mide la rareza de una palabra en toda la colección de reviews. Al multiplicar estas dos medidas, se obtiene un peso que refleja la importancia de una palabra en una review en relación con toda la colección. Las palabras que son comunes en todas las reviews tendrán un peso bajo, mientras que las palabras que son raras y aparecen solo en unas pocas reviews tendrán un peso alto. Esto permite resaltar las palabras más relevantes para cada review.

2.3 Vectorización

En base a la selección de la técnica de ponderación se genera por medio de un objeto de tipo TfidfVectorizer que retorna la matriz dispersa donde cada fila representa un documento y cada columna representa una palabra única en el vocabulario., que será usado por los algoritmos de clasificación.

3) Modelado y evaluación

En el proyecto se hizo uso de tres algoritmos de clasificación los cuales fueron entrenados y puestos en comparación para así decidir cuál es mejor para el objetivo del negocio.

1. **Random Forest:**

El primer modelo utilizado en el notebook es un RandomForestClassifier entrenado sobre una matriz TF-IDF. Esta es una técnica común utilizada para tareas de clasificación de texto.

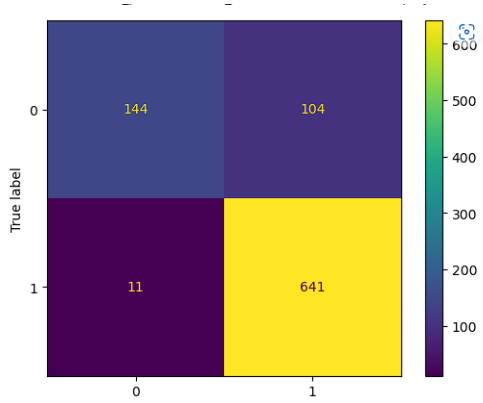
El RandomForestClassifier es un método de aprendizaje por ensamble que ajusta varios clasificadores de árboles de decisión sobre varias submuestras del conjunto de datos y utiliza un promedio para mejorar la precisión predictiva y controlar el sobreajuste. El parámetro random_state se utiliza para establecer la semilla del generador de números aleatorios utilizado por el clasificador.

RandomForest es un modelo muy flexible que se puede adaptar a una variedad de situaciones y problemas de análisis de texto. Por ejemplo, puede ser utilizado tanto para la clasificación como para la regresión, y puede manejar tantas características numéricas como categóricas. También, es un modelo robusto que puede manejar bien los datos con ruido y datos faltantes. Esto es importante en el análisis de texto, ya que los datos de texto a menudo contienen errores de ortografía, abreviaturas, palabras faltantes y otros tipos de ruido.

El método fit() se utiliza para entrenar el modelo sobre la matriz TF-IDF X_tfidf generadas previamente y las etiquetas de destino y_train .

Los resultados de evaluación muestran el rendimiento del modelo entrenado en el conjunto de pruebas. La precisión, recall y puntuación F1 son métricas comunes utilizadas para evaluar el rendimiento del algoritmo de clasificación. La precisión es la relación de verdaderos positivos al número total de predicciones positivas, el recall es la relación de verdaderos positivos al número total de muestras positivas reales, y la puntuación F1 es la media armónica de la precisión y el recall.

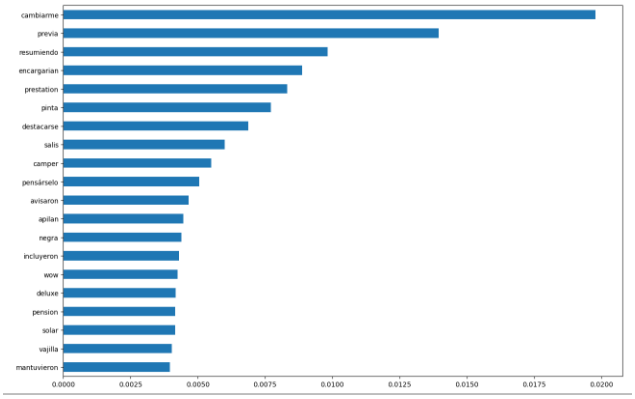
Los resultados indican que el modelo logró una precisión del 0,86, una recall del 0,98 y una puntuación F1 del 0,92 en el conjunto de pruebas. Lo anterior se muestra por medio de la matriz de confusión:



En la que podemos ver que 115 casos un fueron clasificados incorrectamente.

En general, el RandomForestClassifier es una buena opción para tareas de clasificación de texto, especialmente cuando se trata de espacios de características de alta dimensionalidad.

Finalmente podemos apreciar las palabras que el modelo identifico que tenían más peso para determinar si una review es positiva o negativa:



2. Tree decision:

En esta sección se ha utilizado un algoritmo de aprendizaje supervisado llamado árbol de decisión para la construcción del modelo de clasificación de textos. El árbol de decisión es un modelo de aprendizaje automático que toma decisiones a través de un árbol de decisiones que se construye a partir del conjunto de entrenamiento.

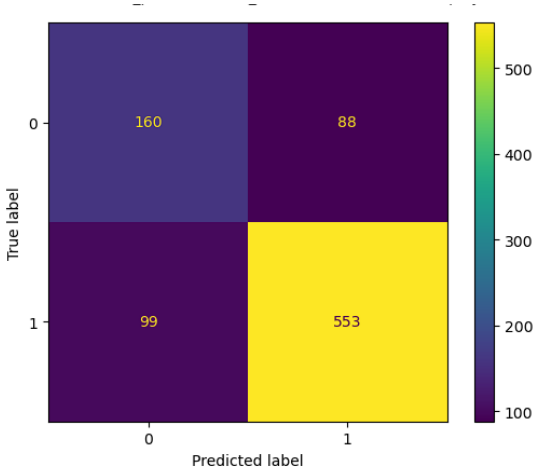
La construcción del modelo se llevó a cabo utilizando el conjunto de entrenamiento y la matriz TF-IDF obtenida a partir del preprocesamiento del texto. El modelo fue entrenado utilizando los datos de entrenamiento y se evaluó su rendimiento utilizando una muestra de prueba.

Para evaluar el rendimiento del modelo se calcularon las métricas de precisión, recall y puntuación F1. Estas métricas permiten evaluar la calidad de las

predicciones del modelo en términos de su capacidad para identificar correctamente los textos de cada clase.

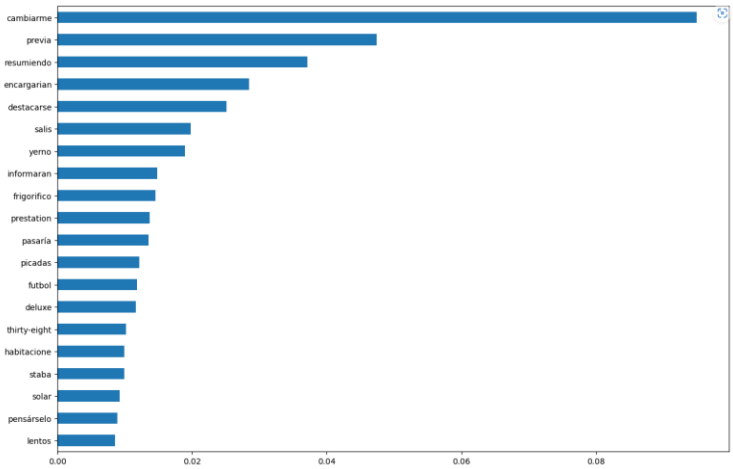
En el caso del modelo de árbol de decisión, se eligió este algoritmo debido a su capacidad para manejar conjuntos de datos con características mixtas (numéricas y categóricas) sin necesidad de preprocesamiento adicional. Además, este modelo es fácilmente interpretable y permite identificar las características más importantes para la clasificación de textos.

En cuanto a los resultados obtenidos con el modelo de árbol de decisión, se observa una precisión de 0.862, un recall de 0.848 y una puntuación F1 de 0.855 en la evaluación de la muestra de prueba. Estos resultados indican que el modelo tiene un buen rendimiento en la clasificación de textos y podría ser utilizado en aplicaciones de análisis de textos. Según la matriz de confusión:



Podemos visualizar que fueron clasificados de forma incorrecta un total de 187 registros, mientras que de forma correcta 713.

Finalmente podemos apreciar las palabras que el modelo identifico que tenían más peso para determinar si una review es positiva o negativa:

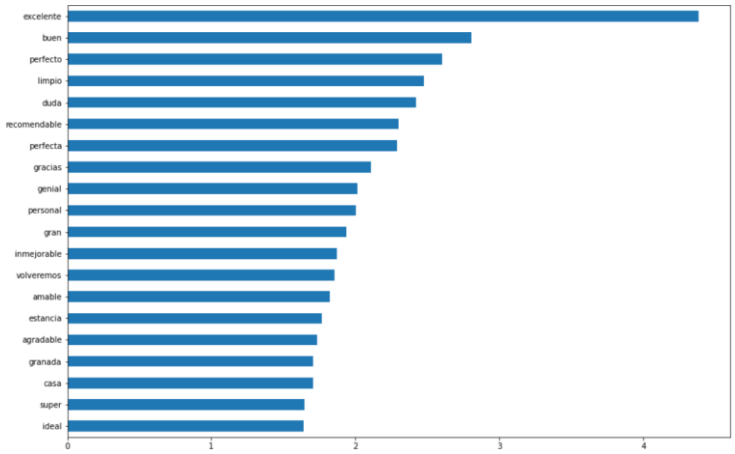


3. Logistic Regression

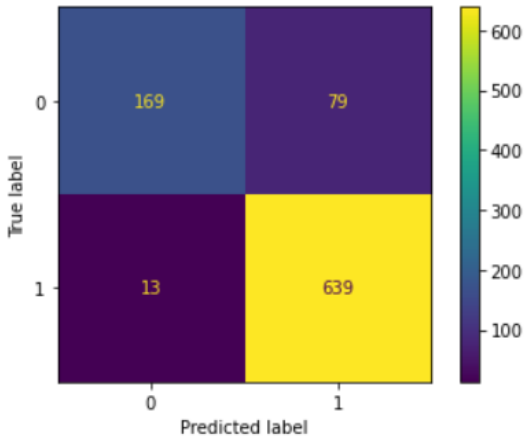
El último algoritmo implementado en el modelo se llama Logistic Regression, el cual se trata de un caso especial de la regresión lineal, pero para casos en los cuales se necesite la clasificación binaria de elementos. El algoritmo funciona de manera tal que para cada entrada de TF-IDF, predice una probabilidad sobre esta entrada para poder clasificarlo dentro una categoría u otra.

Para predecir las probabilidades de cada registro se usa una función determinada de la siguiente manera: p = 1/(1 + exp(-z)) donde z es la función lineal de cada una de las variables predictoras (entradas de la matriz TF-IDF). Esta función logística "transforma" los valores de z a valores entre 0 y 1. De esta manera, dependiendo al valor que se haya transformado, se le asigna a una categoría positiva (sobrepasa un umbral definido por el modelo) o en una categoría negativa (no alcanza a sobrepasar un umbral definido por el modelo).

Este modelo fue seleccionado como parte del proyecto debido a su facilidad de implementación y que su forma de predecir los valores evidencia que la predicción de este mismo modelo es coherente con la realidad. Por ejemplo, la siguiente es una gráfica que muestra las palabras con más importancia dentro del modelo entrenado y se puede notar que son palabras que hacen parte del vocabulario natural para denotar sentimientos sobre algo, en este caso, como opinión de los hoteles.



Las métricas de la matriz de confusión asociadas a este algoritmo en los datos de muestra de testeo que se tomaron del dataframe original tienen los siguientes valores: la precisión tiene un valor de 0.889, el recall de 0.98 y el F1 de 0.932.



Se obtuvieron 92 registros que fueron mal clasificados y 808 registros que fueron bien clasificados.

4) Resultados

Después de realizar la asesoría con la persona experta en estadística, se recomendó hacer uso de una mejor redacción en la explicación de los diferentes procesos de limpieza de datos y explicación de los modelos y resultados obtenidos. Además, se les recomendó a los ingenieros de datos que se realizaran conclusiones generalizadas para la presentación de resultados al negocio. Debido a lo anterior, los resultados que se presentaran a continuación fueron extraídos en relación a una base de datos que contenía diferentes nombres y locaciones de hoteles.

A partir de la ejecución de los tres modelos de clasificación y de las métricas encontradas se puede concluir que el algoritmo que mejor se comporta con los datos de prueba y el cual es recomendado por parte del equipo de ingenieros es el **LogisticRegression**. Esta conclusión se hizo con base a los resultados de la evaluación de rendimiento de cada modelo entrenado con el conjunto de pruebas. Específicamente, los resultados se resumen en las métricas de precisión, relación entre verdaderos positivos y el número total de predicciones positivas; el recall, relación de los verdaderos positivos con el número total de muestras positivas reales y el F1, el cual es la media armónica entre la precisión y el recall. Además, se tiene en cuenta el número de errores cometidos por cada modelo con respecto a una misma muestra de datos.

	Precision	Recall	F1	Número de errores de predicción
Random Forest	0.86	0.98	0.92	115
Decision Tree	0.86	0.85	0.85	187
Logistic Regression	0.89	0.98	0.93	92

Revisando en comparación las métricas de evaluación podemos evidenciar que Logistic Regression es el modelo el cual tiene un mejor valor para F1 (0.93) y es el que tiene menor cantidad de errores al momento de testearlo con la muestra que se separó para este propósito.
Como puntos de mejora para modelos futuros que desarrolle el negocio el grupo de trabajo recomienda que se tenga en cuenta la ortografía de las reseñas que alimentan el modelo, lo anterior con el fin de mejorar los resultados obtenido y que la lemmatizacion tenga un mejor desempeño. En esta iteración se intentó tener en cuenta y corregir la ortografía de las palabras por medio de librería spellCheker pero en términos computacionales fue inviable (más teniendo en cuenta que el modelo tenía en cuenta el idioma inglés y español) ya que los tiempos de ejecución eran muy altos. Por ende, se sugiere al negocio, que para obtener los mejores modelos posibles se contrate infraestructura de la nube capacitada para la ejecución de modelos computacionalmente más complejos.

Finalmente, como estrategias para que el negocio pueda sacar mucho más provecho del modelo de Logistic Regression se recomienda que se revisen las palabras que más tienen peso dentro del modelo para poder implementar estrategias de la mejora del servicio en la vida real para los hoteles. Por ejemplo, en el modelo se encontró la palabra “limpio”, el cual es un adjetivo que se puede relacionar con el estado de salubridad que tiene el hotel y es una palabra que tiene una gran importancia dentro de las reviews, lo cual significaría que las personas le prestan importancia a la limpieza que tiene el hotel al dar una review del mismo.