

Informe Proyecto1-Entrega2 Inteligencia de Negocios  
Automatización y uso de modelos de analítica de textos

Juan David Martínez Moreno - 201923416  
Johan Sebastián Cáceres Charari - 202014171

Universidad De Los Andes  
Departamento De Ingeniería De Sistemas Y Computación  
30 de abril del 2023

La entrega de esta etapa del proyecto contiene un documento que describe el proceso de despliegue de la solución analítica en el ambiente de producción de una organización. Esto incluye la automatización del proceso desde la preparación de los datos hasta la persistencia del modelo analítico construido, al igual que el dejar disponible una aplicación que toma el resultado del modelo y lo pone a disposición de un usuario final. Como parte de la entrega, adicionalmente, se debe incluir un video y todo el software desarrollado para el proyecto.

## Asignación de tareas

Tarea	Persona	Tiempo Asignado	Tiempo Dedicado	Puntos de esfuerzo sobre 100
Video	Juan David Martinez	3 horas	3 horas	15
Creacion del api	Johan Sebastian Caceres	4 horas	4 horas	20
Backend	Johan Sebastian Caceres	2 horas	2 horas	20
Exportar el modelo	Johan Sebastian Caceres / Juan David Martínez	2 horas	2 horas	20
Frontend	Juan David Martínez	3 horas	3horas	25

## Roles asignados

Rol	Persona
Lider de proyecto	Johan Sebastian Caceres
Ingeniero de datos	Johan Sebastian Caceres
Ingeniero de software responsable del diseño de la aplicación y resultados	Johan Sebastian Caceres
Ingeniero de software responsable de desarrollar la aplicación final	Johan Sebastian Caceres

Nombre	Retos enfrentados	Formas de resolverlo
Johan Sebastian Caceres	Al exportar el pipeline se tuvieron problemas con la ejecución del modelo dentro del api, ya que no estaba reconociendo la función creada para tokenizar,	Se intento correr el modelo por fuera del api y funciona. Sin embargo, dentro del api se implementó el tokenizer por defecto de la librería from sklearn.feature_extraction.text import CountVectorizer para

		poder exportar el modelo y poderlo usar.
Juan David Martínez	Añadir la funcionalidad de subir un archivo a la página ya que se tenía que simplemente recibiera un texto de reseña y que lo clasificara entre bueno o malo.	Se usó una función que se ejecutará cada vez que se ingresara al endpoint de /predict. Esta función permitía guardar temporalmente el archivo dentro de la página web para analizar su contenido.

## Reuniones Realizadas

Tema	Fecha
Reunión lanzamiento y planeación del proyecto	22 de abril
Reunión de seguimiento	26 de abril
Reunión de seguimiento	29 de abril
Reunión de finalización	30 de abril

Habiendo hecho todo el preprocesamiento de los datos, la construcción y el entrenamiento del modelo en la primera entrega del proyecto. Se requiere automatizar este proceso para otorgar un valor dentro del negocio del hotel. Por esto mismo, se planteó la creación de una API que recibiera una serie de reseñas para poder determinar cuántas de ellas fueron positivas y negativas.

## Automatización del entrenamiento del modelo

Para lograr la automatización de lo logrado en la primera parte del proyecto se hizo uso de los “pipelines”, los cuales se tratan de una herramienta otorgada por la librería de sklearn que permite seleccionar ciertos pasos de una secuencia de instrucciones de un código que se desean ejecutar. Los “pipelines” permiten automatizar códigos de una manera eficiente y sin necesidad de correr un código, por ejemplo .py, cada vez que se requiera ejecutar ciertas instrucciones para lograr un objetivo. Los “pipelines” se podrían asociar como a un snapshot de un cierto código que se desea ser reutilizado cada vez que sea requerido sin construir de nuevo lo que el código necesitaba para ejecutar las instrucciones.

En nuestro caso, hicimos uso de un “pipeline” de manera que se pudiera automatizar la tokenización de las reseñas y los títulos de estas mismas, y posteriormente, el entrenamiento del modelo sin la necesidad de construir desde 0 las tokenizaciones ni entrenar el modelo cada vez que fuera necesario por parte del negocio. Esto se hace debido a que entrenado el modelo una vez, sería redundante (y poco práctico) entrenar el modelo cada vez que se necesitara predecir los sentimientos que producían las reseñas.

```
stop_words_en = stopwords.words('english')
stop_words_es = stopwords.words('spanish')

# combine the two sets of stop words
stop_words = stop_words_en + stop_words_es

# add "jjjjj" to the stop words list
stop_words.append('jjjjj')

# use the updated stop words list in your text processing
```

```

lemmatizer = WordNetLemmatizer()

# function to map part-of-speech tags to WordNet POS tags
def get_wordnet_pos(treebank_tag):
    if treebank_tag.startswith('J'):
        return 'a' # Adjetivo
    elif treebank_tag.startswith('V'):
        return 'v' # Verbo
    elif treebank_tag.startswith('N'):
        return 'n' # Sustantivo
    elif treebank_tag.startswith('R'):
        return 'r' # Adverbio
    else:
        return 'n' # Sustantivo por defecto

# function to tokenize and lemmatize text
def tokenize(text):
    # detect language of text
    language = detect(text)

    # initialize spellchecker based on language
    if language == 'es':
        # tokenize Spanish text
        tokens = nltk.word_tokenize(text, language='spanish')
    else:
        # tokenize English text
        tokens = nltk.word_tokenize(text)

    # remove non-alphanumeric characters
    tokens = [token for token in tokens if token.isalnum()]

    # transform numbers from digit format to word format
    tokens = [num2words(token) if token.isnumeric() else token for token in tokens]

    # lemmatize tokens
    tokens = [lemmatizer.lemmatize(token, pos=get_wordnet_pos(pos_tag([token])[0][1])) for token in tokens]

    return tokens

tfidf = TfidfVectorizer(tokenizer = tokenize, stop_words = stop_words, lowercase = True)

```

```

tfidf_model = RandomForestClassifier(random_state = 3)

tfidf_model.fit(X_tfidf, y_train)

RandomForestClassifier(random_state=3)

```

## Exportar el pipeline

```

9]: from sklearn.pipeline import make_pipeline
    from joblib import dump, load

    #from sklearn.feature_extraction.text import CountVectorizer

    #vectorizer = CountVectorizer(stop_words=stop_words)

    model = make_pipeline(tfidf, tfidf_model)

    model.fit(X_train, y_train)

    filename = 'ARandomForestModel.joblib'
    dump(model, filename)

    p2 = load(filename)

```

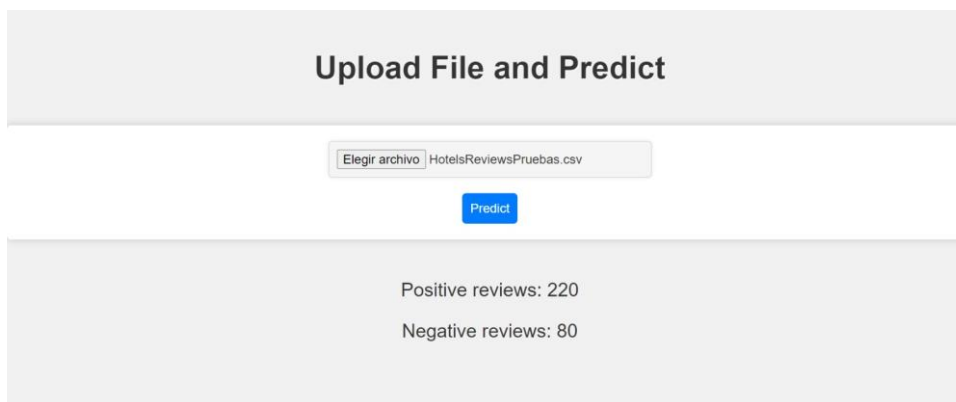
El “pipeline”, en específico, tokeniza por medio de la técnica del TF-IDF (explicación detallada en la primera entrega) las reseñas y el título que se pasan como parámetros. Además, estos textos pasados antes de ser tokenizados se les hace un procesamiento para facilitar la tokenización misma y así, no aportar a la creación de tokens innecesarios ni redundantes. Por ejemplo, eliminación de palabras que no son alfa-numéricas, pasar los números a formato de texto y la *lemmatización* de estos mismos. Después, se entrena el modelo que se desea elegir para la predicción de los sentimientos, el cual en este caso se hizo

uso del modelo de Random Forest. Finalmente se exportó el pipeline en un archivo .joblib el cual estaba listo para ser utilizado a recibir reseñas y predecir sobre sus sentimientos.

### Desarrollo del API y su implementación como aplicación web

Ahora, crearemos una API sencilla con el propósito de integrar el “pipeline” creado anteriormente. Para esto haremos uso de una librería llamada FastAPI, la cual nos permite crear endpoints de una manera rápida para así recibir y enviar solicitudes por parte de un usuario hacia y por la aplicación web. En nuestro caso, simplemente hacemos uso del endpoint “/predict” para predecir los valores que recibirá la aplicación web.

Teniendo esta funcionalidad creada para nuestra aplicación web, se diseñó una plantilla de interfaz visual básica para que el usuario pueda interactuar de una manera más eficiente con el modelo, para que de esta manera simplemente elija el archivo del cual se requieren predecir las reseñas y recibe como respuesta la cantidad de reseñas positivas y negativas.



Upload File and Predict

Elegir archivo | HotelsReviewsPruebas.csv

Predict

Positive reviews: 220  
Negative reviews: 80

El funcionamiento de la página web es sencilla: 1) se debe tener un archivo .csv el cual contenga dos columnas con los nombres de “title” y “review\_text”, y en las cuales deban ir el título de la reseña y el contenido de la misma, respectivamente. Este archivo es el que se sube a la aplicación web para que haga las predicciones correspondientes sobre las reseñas 2) En la parte inferior de la página web se podrá observar la cantidad de reseñas que fueron positivas y cuántas de ellas fueron negativas en la base de datos de reseñas que se ingresó como archivo en la página web.

### Importancia de la aplicación web en el negocio

La aplicación web fue diseñada con el propósito en específico de ver cómo fue el desempeño de un hotel en un tiempo determinado teniendo en cuenta las reseñas que recibió en un cierto rango de fechas. Es decir, las reseñas deben ser recolectadas por el encargado de servicio al cliente, este debe ser capaz de clasificar las reseñas en un rango de fechas requerido por el gerente para revisar cómo fue el desempeño del hotel en ese tiempo. De esta manera, el gerente podrá ser capaz de poder evidenciar cómo fueron las reseñas dentro, por ejemplo, de una semana, un mes, un semestre, y así hacer un seguimiento paulatino sobre si la calidad del servicio ofrecido por el hotel ha ido aumentando o disminuyendo a través del tiempo.

Con la aplicación web se puede saber cuántas de las reseñas que se recopilaron dentro de un tiempo fueron positivas o negativas. Con esto en mente, se puede recomendar al negocio un seguimiento semanal de estas reseñas para revisar si la calidad del servicio ha ido cambiando y así identificar qué cosas deberían dejar de hacer, qué cosas deberían seguir haciendo y qué

cosas deberían definitivamente dejar de hacer en pro de conseguir una gran cantidad de reseñas positivas.

### **Revisión con el experto en estadística**

Después de habernos reunido con la experta en estadística, estos fueron los cambios que nos sugirió para la mejora tanto del modelo como de la aplicación web:

- Primeramente, en la página web, se veían las reseñas que se clasificaban junto con el resultado de 1 (clasificación buena) o 0 (clasificación mala) lo cual no era muy claro para el usuario que fuera a usar la aplicación y no estaba acorde con el objetivo que se quería lograr en este caso porque no se sabían cuántas reseñas habían sido buenas y cuántas habían sido malas. Por esto mismo, se decidió en hacer un conteo de las reseñas con 1 y 0 y así agruparlas para mostrar el número que eran buenas y el número de las que eran malas.
- Nos recomendó que la aplicación web fuera capaz de subir archivos con gran cantidad de reseñas para que en realidad se pudiera hacer un análisis unificado de las reseñas que se tenían del hotel, ya que, principalmente, se tenía pensado que simplemente la aplicación web recibiera una sola reseña con su título y que la clasificara si era buena o mala. Sin embargo, este enfoque no era tan significativo ya que simplemente ver si una reseña era buena o mala no nos daba información sobre lo que se debería mejorar en el hotel o si el hotel tenía una buena reputación dentro del mercado.