Alejandro Gómez **202015122**

Jonathan Rivera **202022864**

Julián Castro **202020847**

Inteligencia de Negocios

Proyecto 1 – Etapa 1

1. Entendimiento del negocio y enfoque analítico.

a. Definición de los objetivos y criterios de éxito desde el punto de vista del negocio.

Objetivos de negocio

-Identificar áreas de mejora: determinar los factores que caracterizan a los lugares con calificaciones bajas. Que descripciones cualitativas obtienen los peores lugares.

-Identificar características de atracción: entender cuales son los aspectos que mas atraen y satisfacen las necesidades de los clientes, cuáles son las características de los lugares que más valoran positivamente.

-Estimación automática de calificaciones: Desarrollar un modelo predictivo que pueda estimar la calificación potencial de un sitio turístico basado en las reseñas y características identificadas. Esto ayudará a anticipar la recepción de futuros visitantes y a formular estrategias proactivas de mejora.

-Implementar estrategias de mejora: desarrollar planes de acción basados en el análisis de datos recibido.

-Promover el turismo en Colombia: diversificando y creciendo la oferta turística en el pais

Criterios de éxito

-Obtención de un modelo útil que pueda predecir calificaciones cuantitativas a partir de opiniones cualitativas de los clientes, que tenga una alta precisión en sus estimaciones.

-Mejorar la percepción de los clientes existente del negocio, a partir de la implementación de estrategias basadas en el análisis de datos.

-Aumentar la visibilidad del negocio a partir de la mejora de las calificaciones que se le dan a este, obteniendo en el proceso una cantidad de clientes nuevos-

-Fortalecimiento de la imagen de Colombia a nivel nacional e internacional como un destino turístico diverso y de calidad.

Impacto del proyecto en Colombia

La exitosa ejecución de este proyecto podría generar un impacto significativo en el crecimiento de la industria turística en Colombia, generando alto impulso impulso en la economía, enriqueciendo la diversidad cultural y natural del país, y elevando el nivel de vida de las comunidades locales a través de prácticas turísticas sostenibles. Para COTELCO, esto conllevaría no solo beneficios económicos directos para sus miembros, sino también un fortalecimiento generalizado de la industria hotelera y turística, que es fundamental para la economía colombiana. A largo plazo, este proyecto podría posicionar a Colombia como un referente en turismo a nivel regional, destacando por su innovación en la gestión de experiencias turísticas.

b. Descripción del enfoque analítico para alcanzar los objetivos del negocio.

Desde la perspectiva del aprendizaje automático, el requerimiento consiste en analizar conjuntos de datos de reseñas de sitios turísticos para identificar características que influyen en la atracción de turistas y en las calificaciones recibidas por los lugares. El objetivo es desarrollar modelos predictivos que puedan determinar la calificación que un sitio turístico podría recibir por parte de los visitantes.

Para abordar este desafío, proponemos utilizar técnicas de procesamiento de lenguaje natural (NLP) para extraer características relevantes de las reseñas de los turistas, así como algoritmos de aprendizaje supervisado para la

clasificación de las reseñas y la predicción de las calificaciones. Utilizaremos 3 técnicas distintas para la vectorización de las reseñas, paso esencial para poder alimentarlas a algoritmos de clasificación, con los cuales realizaremos la creaciones de modelos, a partir de el uso de 3 algoritmos distintos, usando todos para cada técnica de vectorización.

A continuación las técnicas de vectorización y algoritmos utilizados:

Técnicas de vectorización

Bag of Words (BoW): Convierte el texto en un vector donde cada elemento representa la frecuencia de una palabra en el documento, ignorando el orden de las palabras pero manteniendo la esencia del contenido.

TF-IDF (Term Frequency-Inverse Document Frequency): Similar al BoW pero con una mejora en la ponderación de las palabras, asignando menos importancia a las palabras comunes y más a las únicas en el conjunto de documentos.

Word Embeddings (Word2Vec): Representa las palabras en vectores de características continuas, capturando las relaciones semánticas entre ellas a través del aprendizaje de contextos similares.

Algoritmos de clasificación

Regresión Logística: Un modelo estadístico que estima la probabilidad de una variable categórica. Es útil para casos de clasificación binaria o múltiple y funciona bien como línea de base para este tipo de problemas.

Máquinas de Vectores de Soporte (SVM): Un algoritmo potente para la clasificación y regresión de datos lineales y no lineales. SVM es efectivo en espacios de alta dimensión, como los generados por métodos de vectorización de texto.

Random Forest: Un ensamble de árboles de decisión que mejora la capacidad de generalización del modelo reduciendo el riesgo de sobreajuste. Es útil para clasificar basado en la importancia asignada a las características.

c. Nombres de los estudiantes del curso de estadística con los cuales va a interactuar para validar el enfoque que le está dando el proyecto. Y planeación trabajo interdisciplinario.

Estudiantes de estadística: Sofia Vargas y Lorena Figueroa

Canal de comunicación definido: Whatsapp

Fecha primera reunión (Conocernos y contextualizar el proyecto): Sábado 31 de marzo 2024

Fecha segunda reunión (Validación de resultados etapa 1): Sábado 6 de abril 2024

Fecha tercer reunión (Inicio de trabajo etapa 2): Sábado 13 de abril 2024

Tabla de resumen

Oportunidad/problema Negocio	Analizar las características de los sitios turísticos para entender qué los hace atractivos o poco atractivos para los turistas locales e internacionales. Se busca también desarrollar un mecanismo para predecir la calificación que los turistas darán a un sitio.
Enfoque analítico (Descripción del requerimiento desde el punto de vista de aprendizaje automático) e incluya las técnicas y algoritmos que propone utilizar.	Se propone emplear tres técnicas de vectorización: Count Vectorizer, TF-IDF Vectorizer y Word Embeddings. Para el modelado predictivo, se consideran tres algoritmos: Support Vector Machines (SVM), Random Forest y XGBoost. Estas técnicas y algoritmos permitirán identificar patrones en las reseñas y predecir las calificaciones de los sitios turísticos, facilitando la toma de decisiones para mejorar la experiencia turística y promover el desarrollo del sector en Colombia
Organización y rol dentro de ella que se beneficia con la oportunidad definida	Los beneficiarios son actores dentro de la industria turística de Colombia, incluyendo al Ministerio de Comercio, Industria y Turismo, la Asociación Hotelera y Turística de Colombia (COTELCO), como a cadenas hoteleras como Hilton, Hoteles Estelar y Holiday Inn. Estos actores se beneficiarán al obtener información detallada sobre las características que influyen en las calificaciones de los turistas, lo que les permitirá tomar

	decisiones informadas para mejorar la calidad y popularidad de los sitios turísticos en el país.
Contacto con experto externo al proyecto y detalles de la planeación	Expertos estadísticos: Sofia Vargas y Lorena Figueroa Fecha primera reunión (Conocernos y contextualizar el proyecto): Sábado 31 de marzo 2024 Fecha segunda reunión (Validación de resultados etapa 1): Sábado 6 de abril 2024 Fecha tercer reunión (Inicio de trabajo etapa 2): Sábado 13 de abril 2024

2.Entendimiento y preparación de los datos

El primer paso realizado para el entendimiento de los datos, fue observar la forma en la que se encuentran representados en nuestra fuente de datos.

```
data=pd.read_csv('tipo2_entrenamiento_estudiantes.csv', sep=',', encoding = 'utf-8')
# Asignación a una nueva variable de los datos leidos
data_t=data
```
✓ 0.1s

```
data.head()
```
✓ 0.0s

	Review	Class
0	Muy buena atención y aclaración de dudas por p...	5
1	Buen hotel si están obligados a estar cerca de...	3
2	Es un lugar muy lindo para fotografías, visite...	5
3	Abusados con la factura de alimentos siempre s...	3
4	Tuvimos un par de personas en el grupo que rea...	3

Los datos que tenemos se presentan de una manera muy simple, una columna con las reseñas en lenguaje natural, y otra columna con la calificación numérica de las reseñas. No fue necesario remover ninguna columna ya que de entrada tenemos nuestra columna objetivo (valor a predecir) y nuestra columna de entrenamiento.

2.1. Idioma de los datos

Para el planteamiento de un modelo que analice textos, es esencial primero conocer el idioma en el que se encuentran los textos, esto porque tanto para la transformación de los textos (lematización, tokenización) para poder ser procesado por el modelo, como para configurar el modelo, es necesario que se tenga un lenguaje común para todos los tokens.

```
from langdetect import detect
data_t['idioma'] = data_t['Review'].apply(detect)
data_t
```
✓ 41.4s

	Review	Class	idioma
0	Muy buena atención y aclaración de dudas por p...	5	es
1	Buen hotel si están obligados a estar cerca de...	3	es
2	Es un lugar muy lindo para fotografías, visite...	5	es
3	Abusados con la factura de alimentos siempre s...	3	es
4	Tuvimos un par de personas en el grupo que rea...	3	es
...	...	...	...
7870	Me parece buen sistema, agiliza el transporte,...	4	es
7871	Fue una escapada de un día desde el complejo, ...	4	es
7872	La Plaza de la Revolución es un lugar emblemát...	3	es
7873	Es la segunda ocasión que me quedo en los cuar...	1	es
7874	Llegamos por casualidad a Los Mercaderes, un g...	5	es

El siguiente paso consistió en observar el conteo de los valores en la columna idioma para entender la proporción lingüística de las reseñas.

```
data_t['idioma'].value_counts()
✓ 0.0s

idioma
es    7867
it       3
pt       2
en       2
sq       1
```

Pudimos ver que, virtualmente todas las reseñas se encontraban en español, con tan solo 8 reseñas de 7874 perteneciendo a otro idioma. Como mencionábamos previamente, el modelo estaba planteado para un solo idioma, por lo que se tomó la decisión de remover las reseñas de otros idiomas, mostraremos un poco de esto en la fase de transformación de los datos.

2.2 Entendimiento de las columnas

Planteamos el entendimiento y análisis de la distribución de los datos para ayudarnos a contribuir cualitativamente los objetivos de negocios. Comenzamos observando las estadísticas principales de la columna de los ratings.

```
data_t.describe()
✓ 0.0s
```

	Class
count	7875.000000
mean	3.502603
std	1.320435
min	1.000000
25%	3.000000
50%	4.000000
75%	5.000000
max	5.000000

A partir de las estadísticas de los ratings de las reseñas pudimos entender lo siguiente:

-No existían ratings inválidos dado que el máximo y el mínimo se encontraban en una escala lógica y consistente (1-5)

-En promedio las reseñas tenían una calificación de 3.5, por lo que se pudo evidenciar que se tiene una distribución apropiada de calificaciones (no estaban sesgadas a ningún extremo).

-Lo anterior se pudo confirmar observando la distribución de los datos en los cuartiles. El 25% de los ratings tenían una calificación máxima de 3, el 50% de 4, y el 75% de 5.

-Trabajábamos con 7875 reseñas para la creación del modelo y su posterior validación.

Posteriormente continuamos observando la longitud de las reseñas para entender cuántas palabras pueden llegar a tener influencia en la predicción de un rating:

```
# Calcular la longitud de cada reseña y almacenarla en una nueva columna
data['review_length'] = data['Review'].apply(lambda x: len(x.split()))

# Obtener la longitud de la reseña más larga
max_length = data['review_length'].max()

# Obtener la longitud de la reseña más corta
min_length = data['review_length'].min()

# Obtener la longitud promedio de las reseñas
avg_length = data['review_length'].mean()

# Obtener la mediana de la longitud de las reseñas
median_length = data['review_length'].median()

# Imprimir las estadísticas
print("Longitud máxima de reseña:", max_length)
print("Longitud mínima de reseña:", min_length)
print("Longitud promedio de reseña:", avg_length)
print("Mediana de longitud de reseña:", median_length)

✓ 0.0s

Longitud máxima de reseña: 1809
Longitud mínima de reseña: 2
Longitud promedio de reseña: 70.8167619047619
Mediana de longitud de reseña: 45.0
```

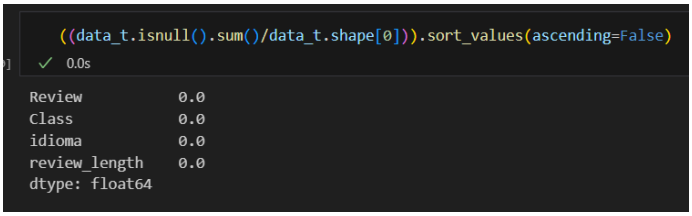
-Pudimos entender que la longitud de las reseñas puede llegar a ser muy variable con extremos muy cortos y muy extensos, con la reseña mas corta utilizando tan solo 2 palabras, y la mas larga 1809 palabras.

-También observamos que la reseña media tiene 45 palabras, y en promedio se usan aproximadamente 70 palabras.

2.3 Calidad de los datos

2.3.1 Completitud

Pudimos observar que no teníamos valores nulos en nuestros datos, por lo tanto no fue necesario hacer un análisis de manejo de nulidad



2.3.2. Unicidad

Pudimos observar que teníamos 102 registros duplicados, debido a que los duplicados no añaden valor alguno al modelo, pero si pueden introducir un sesgo (debido a que ciertas palabras pueden asociarse de manera más fuerte a un rating de manera errónea), decidimos eliminarlos en la fase de limpieza de datos.

2.3.3 Consistencia

Como vimos previamente en el entendimiento de los ratings, estos datos son consistentes dado que todas las calificaciones se encuentran en un rango coherente y lógico de (1-5).

2.3.4 Validez

Debido a que no existe una relación cuantificable entre las columnas, no tuvo cabida un análisis de validez de las relaciones entre columnas.

2.4 Preparación de los Datos

2.4.1 Limpieza de los datos

2.4.1.1. Quitar duplicados

Nos deshicimos de los datos duplicados, dejando el dataframe en estado de unicidad.

2.4.1.2 Quitar reseñas en otros idiomas

Como se mencionó en el entendimiento de datos, nos quedamos con solo las reseñas que estaban en español.

2.4.1.3 Transformar los datos

En este apartado creamos código para limpiar y preprocesar el texto tokenizado, lo que ayuda a mejorar la calidad de los datos antes de alimentarlos al modelo de predicción de calificaciones. Al eliminar información no relevante y estandarizar el texto, se espera que el modelo pueda aprender patrones más significativos y precisos para hacer predicciones más efectivas. Este proceso de eliminación de ruido consistió en eliminar caracteres que no fueran ASCII; que no pueden ser procesados de manera correcto por los modelos, pasar todo el texto a minúscula; para evitar que se consideren distintas dos palabras iguales solo por el uso de mayúsculas/minúsculas, remover la puntuación; ya que no son palabras que se puedan procesar, remplazar los números por su representaciones textuales, para que se consideren palabras por el modelo, y finalmente se removieron las ‘stopwords’ que no añaden un valor semántico a los textos. Sin embargo, fue necesario tokenizar primero las reviews para poder aplicar la limpieza.

```python
def remove_non_ascii(words):
    """Remove non-ASCII characters from list of tokenized words"""
    new_words = []
    for word in words:
        new_word = unicodedata.normalize('NFKD', word).encode('ascii', 'ignore').decode('utf-8', 'ignore')
        new_words.append(new_word)
    return new_words
def to_lowercase(words):
    """Convert all characters to lowercase from list of tokenized words"""
    new_words = []
    for word in words:
        new_words.append(word.lower())
    return new_words
def remove_punctuation(words):
    """Remove punctuation from list of tokenized words"""
    new_words = []
    for word in words:
        new_word = re.sub(r'[^\w\s]', '', word)
        if new_word != '':
            new_words.append(new_word)
    return new_words
def replace_numbers(words):
    """Replace all integer occurrences in list of tokenized words with textual representation in Spanish"""
    new_words = []
    for word in words:
        if word.isdigit():
            new_word = num2words(word, lang='es')
            new_words.append(new_word)
        else:
            new_words.append(word)
    return new_words
def remove_stopwords(words):
    stop_words = set(stopwords.words("spanish"))
    new_words = [word for word in words if word not in stop_words]
    return new_words
```

2.4.2 Tokenización

La tokenización permite dividir frases u oraciones en palabras. Con el fin de desglosar las palabras correctamente para el posterior análisis. Es usual que en tareas de NLP se transformen las contracciones en los textos procesados, sin embargo dado que para la tarea ejecutada el texto se encontraba en español, donde las contracciones no son tan comunes y se utilizan mayoritariamente en 'stopwords' no fue necesario aplicar este paso.

```python
data_t['words'] = data_t['Review'].apply(word_tokenize)
data_t.head()
```

✓ 3.8s

	Review	Class	idioma	review_length	words
0	Muy buena atención y aclaración de dudas por p...	5	es	36	[Muy, buena, atención, y, aclaración, de, duda...
1	Buen hotel si están obligados a estar cerca de...	3	es	54	[Buen, hotel, si, están, obligados, a, estar, ...
2	Es un lugar muy lindo para fotografías, visite...	5	es	17	[Es, un, lugar, muy, lindo, para, fotografías,...
3	Abusados con la factura de alimentos siempre s...	3	es	81	[Abusados, con, la, factura, de, alimentos, si...
4	Tuvimos un par de personas en el grupo que rea...	3	es	70	[Tuvimos, un, par, de, personas, en, el, grupo...

```python
data_t['words1']=data_t['words'].apply(preprocessing)
data_t.head(10)
```

✓ 3.5s Python

	Review	Class	idioma	review_length	words	words1
0	Muy buena atención y aclaración de dudas por p...	5	es	36	[Muy, buena, atención, y, aclaración, de, duda...	[buena, atencion, aclaracion, dudas, parte, se...
1	Buen hotel si están obligados a estar cerca de...	3	es	54	[Buen, hotel, si, están, obligados, a, estar, ...	[buen, hotel, si, estan, obligados, cerca, cen...
2	Es un lugar muy lindo para fotografías, visite...	5	es	17	[Es, un, lugar, muy, lindo, para, fotografías,...	[lugar, lindo, fotografias, visiten, selina, m...
3	Abusados con la factura de alimentos siempre s...	3	es	81	[Abusados, con, la, factura, de, alimentos, si...	[abusados, factura, alimentos, siempre, echan,...
4	Tuvimos un par de personas en el grupo que rea...	3	es	70	[Tuvimos, un, par, de, personas, en, el, grupo...	[par, personas, grupo, realmente, queriamos, c...
5	Un complejo enorme de fortificaciones frente a...	4	es	34	[Un, complejo, enorme, de, fortificaciones, fr...	[complejo, enorme, fortificaciones, frente, an...

2.4.3 Normalización

Tomamos la decisión de realizar normalización en forma de las técnicas de lematización y stemming, dado que es un proceso esencial en el procesamiento del lenguaje natural para reducir la variabilidad lingüística de las palabras y mejorar la precisión de los modelos de aprendizaje automático. Al convertir diferentes formas de una palabra a su raíz o lema base, estas técnicas permiten que el modelo trate variantes de una palabra como una sola entidad, mejorando así la eficiencia del procesamiento de texto y la comparabilidad de los términos.

```python
import spacy
nlp = spacy.load('es_core_news_md')
```

✓ 4.9s

```python
import spacy
from nltk.stem.snowball import SnowballStemmer

def stem_words(words):
    """Stem words in list of tokenized words"""
    stemmer = SnowballStemmer("spanish")
    stemmed_words = [stemmer.stem(word) for word in words]
    return stemmed_words
def lemmatize_verbs(words):
    """Lemmatize verbs in list of tokenized words"""
    text = " ".join(words)
    doc = nlp(text)
    lemas = [token.lemma_ for token in doc]
    return lemas
def stem_and_lemmatize(words):
    stems = stem_words(words)
    lemmas = lemmatize_verbs(words)
    return stems + lemmas
data_t['words1'] = data_t['words1'].apply(stem_and_lemmatize)
data_t.head()
```

	Review	Class	idioma	review_length	words	words1
0	Muy buena atención y aclaración de dudas por p...	5	es	36	[Muy, buena, atención, y, aclaración, de, duda...	[buen, atencion, aclar, dud, part, senorit, ve...
1	Buen hotel si están obligados a estar cerca de...	3	es	54	[Buen, hotel, si, están, obligados, a, estar, ...	[buen, hotel, si, estan, oblig, cerc, centr, c...
2	Es un lugar muy lindo para fotografías, visite...	5	es	17	[Es, un, lugar, muy, lindo, para, fotografías,...	[lug, lind, fotografi, visit, selin, music, vi...
3	Abusados con la factura de alimentos siempre s...	3	es	81	[Abusados, con, la, factura, de, alimentos, si...	[abus, factur, aliment, siempr, echan, culp, v...
4	Tuvimos un par de personas en el grupo que rea...	3	es	70	[Tuvimos, un, par, de, personas, en, el, grupo...	[par, person, grup, realment, queri, com, cabr...

3.Modelado y evaluación

3.1 División de los datos de entrenamiento y testeo

```
X_train, X_test, y_train, y_test = train_test_split(data_t[["words"]], data_t["Class"], test_size=0.3, stratify=data_t["Class"],
```

Se hizo la división de los datos en entrenamiento y testeo previo a la creación de los modelos. Se uso un 30% de los datos para conjunto de testeo.

3.2 Modelos

Como se mencionaba anteriormente se crearon 9 modelos: 3 algoritmos distintos para cada uno de las 3 técnicas de vectorización empleadas. La idea es utilizar una gran variación de técnicas y algoritmos para obtener el mejor modelo que maximice las métricas de precisión, recall y f1.

3.2.1 Modelos con vectorización BagOfWords. Implementado por: Alejandro Gomez

Empezamos haciendo una copia de los conjuntos de entrenamiento y de prueba, el motivo de esto es evitar que las transformaciones de los datos durante la creación del modelo, puedan afectar al desarrollo de los otros modelos.

```
X_train_BoW = X_train.copy()
X_test_BoW = X_test.copy()
Y_train_BoW = y_train.copy()
Y_test_BoW = y_test.copy()


bow = CountVectorizer(tokenizer=word_tokenize, stop_words=stop_words, lowercase=True)
```

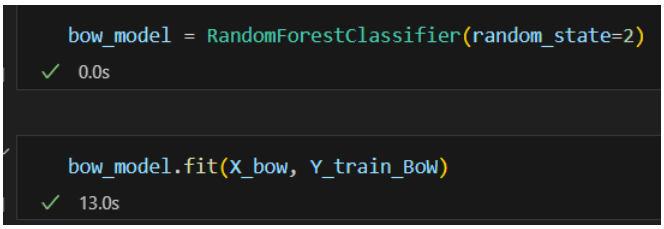
Además, hacemos la vectorización de los datos usando la función countVectorizer, y fit_transform para convertir el texto en una representación de bolsa de palabras, donde se convierte cada documento del conjunto de entrenamiento en un vector numérico. Cada posición en este vector representa un término en el vocabulario aprendido durante el paso de fit, y el valor en cada posición representa la presencia (y posiblemente la frecuencia) de ese término en el documento específico.

```
X_bow = bow.fit_transform(X_train_BoW["words"])
✓ 1.9s
c:\Users\JHONATAN RIVERA\AppData\Local\Programs\Python\P
  warnings.warn(

print("Vocabulary size:", len(bow.vocabulary_))
✓ 0.0s
Vocabulary size: 24695
```

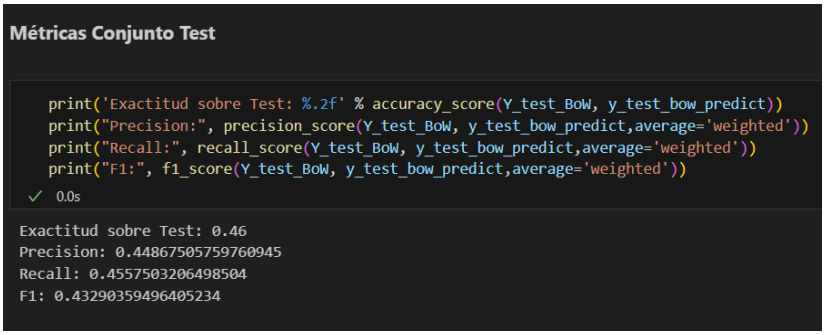
3.2.1.1 BoW con clasificación Random Forest Classifier

El primer algoritmo que utilizaremos sera random Forest Classifier, Durante la fase de entrenamiento, el algoritmo construye un conjunto de árboles de decisión de manera aleatoria. Cada árbol se entrena utilizando una muestra aleatoria del conjunto de datos de entrenamiento y un subconjunto aleatorio de características (palabras en este caso). Una vez que se han creado todos los árboles de decisión, el RandomForestClassifier realiza predicciones combinando las predicciones de cada árbol individual. En el caso de la clasificación de ratings de reviews, dado un conjunto de palabras en una review, cada árbol en el bosque votará por una clasificación. La clasificación final será determinada por mayoría de votos.

Importamos el modelo de la librería SKlearn y lo entrenamos con los datos que habíamos vectorizado previamente.



A continuación predecimos los ratings para el conjunto de prueba y comparamos las predicciones con los rating reales para obtener las métricas del modelo.



En el contexto de NLP para la predicción de ratings de reviews, las métricas clave incluyen la exactitud, que mide la proporción de predicciones correctas sobre el total de predicciones realizadas por el modelo; la precisión, que mide la proporción de predicciones correctas; el recall, que evalúa la capacidad del modelo para identificar correctamente las instancias positivas; y el F1-Score, una medida armónica que equilibra precisión y recall, proporcionando una evaluación integral del rendimiento del modelo. Indagaremos mas en lo que puede representar los valores numéricos de estas medidas, una vez hayamos comparado las métricas de todos los modelos y escogido el mejor de estos en la fase de resultados.

3.2.1.2 BoW con clasificación Regresión Logística

En el contexto de NLP para la predicción de ratings de reviews, la regresión logística se utiliza para modelar la relación entre las características extraídas de los textos de las reviews y las calificaciones asociadas. Funciona mediante la estimación de las probabilidades de que una determinada review pertenezca a cada una de las clases de calificación posible. Esto se logra aplicando una función logística a una combinación lineal de las características, lo que transforma la salida en un valor entre 0 y 1 que representa la probabilidad de pertenencia a una clase. Posteriormente, se aplica un umbral para asignar la calificación final.

Para este algoritmo seguimos pasos muy similares a random forest. Importamos el modelo desde la librería, hacemos el entrenamiento con los datos vectorizados y procedemos a realizar las predicciones con los datos de testeo, para asi obtener las métricas de evaluación del modelo.

```python
logistic_model = LogisticRegression(random_state=5, max_iter=1000)

logistic_model.fit(X_bow, Y_train_BoW)

# Realiza predicciones en los datos de prueba
y_test_logistic_predict = logistic_model.predict(bow.transform(X_test["words"]))

# Calcula las métricas de evaluación
exactitud_logistic = accuracy_score(Y_test_BoW, y_test_logistic_predict)
precision_logistic = precision_score(Y_test_BoW, y_test_logistic_predict, average='weighted')
recall_logistic = recall_score(Y_test_BoW, y_test_logistic_predict, average='weighted')
f1_logistic = f1_score(Y_test_BoW, y_test_logistic_predict, average='weighted')

ConfusionMatrixDisplay.from_predictions(Y_test_BoW, y_test_logistic_predict)

print ('Exactitud (Regresión Logística): ', exactitud_logistic)
print("Precision (Regresión Logística:", precision_logistic)
print("Recall (Regresión Logística:", recall_logistic)
print("F1 (Regresión Logística:", f1_logistic)
```

✓ 3.0s

```
Exactitud (Regresión Logística):  0.4480547242411287
Precision (Regresión Logística: 0.4477869151724616
Recall (Regresión Logística: 0.4480547242411287
F1 (Regresión Logística: 0.44699811332152156
```

Podemos ver que obtenemos métricas muy parecidas a las obtenidas con random forest, seguimos probando con otros modelos para la comparación acumulativa.

3.2.1.3. BoW con clasificación SVM

En el contexto de clasificación de ratings de reviews usando NLP, el SVM (Máquina de Soporte Vectorial) es un modelo poderoso que busca encontrar el hiperplano que mejor separa los datos en el espacio de características, optimizando la distancia (margen) entre el hiperplano y los puntos más cercanos de cada clase, conocidos como vectores de soporte. En el caso de datos no linealmente separables, SVM utiliza funciones kernel para transformar el espacio de características a una dimensión donde la separación sea posible, permitiendo manejar la complejidad y la riqueza lingüística de los textos.

Seguimos los mismos pasos que con los anteriores algoritmos para la creación y evaluación del modelo.

```python
# Crea un modelo de Support Vector Machine (SVM)
svm_model = SVC(kernel='linear', random_state=5)

svm_model.fit(X_bow, Y_train_BoW)

# Realiza predicciones en los datos de prueba
y_test_svm_predict = svm_model.predict(bow.transform(X_test["words"]))

# Calcula las métricas de evaluación
exactitud_svm = accuracy_score(Y_test_BoW, y_test_svm_predict)
precision_svm = precision_score(Y_test_BoW, y_test_svm_predict, average='weighted')
recall_svm = recall_score(Y_test_BoW, y_test_svm_predict, average='weighted')
f1_svm = f1_score(Y_test_BoW, y_test_svm_predict, average='weighted')

ConfusionMatrixDisplay.from_predictions(Y_test_BoW, y_test_svm_predict)

print ('Exactitud (SVM): ', exactitud_svm)
print("Precision (SVM):", precision_svm)
print("Recall (SVM):", recall_svm)
print("F1 (SVM):", f1_svm)
```

✓ 21.9s

```
Exactitud (SVM):  0.42625053441641725
Precision (SVM): 0.430599900155423
Recall (SVM): 0.42625053441641725
F1 (SVM): 0.42769462647285844
```

Obtenemos métricas livianamente peores que con los dos algoritmos previos, por lo tanto podemos descartar este modelo para su uso final.

3.2.2 Modelos con Vectorización TF-IDF. Autor: Julian Castro

TF-IDF (Frecuencia de Término - Frecuencia Inversa de Documento) es una técnica estadística usada para evaluar la importancia de una palabra en un documento en relación a una colección de documentos o corpus. En el contexto de análisis de reseñas y clasificación de ratings, TF-IDF permite ponderar las palabras basándose en su relevancia, otorgando mayor peso a términos que son frecuentes en una reseña específica pero menos comunes en el conjunto total de reseñas. Esto ayuda a resaltar palabras clave que son indicativas de sentimientos positivos o negativos y de características específicas de los sitios turísticos, mejorando así la precisión de los modelos de aprendizaje automático al identificar y clasificar el tono y la intención detrás de las opiniones de los turistas.

Empezamos vectorizando usando la técnica de tf-idf los datos que tenemos

```
tfidf = TfidfVectorizer(tokenizer=word_tokenize, stop_words=stop_words, lowercase=True)
✓ 0.0s

X_tfidf = tfidf.fit_transform(X_train["words"])
✓ 1.8s

c:\Users\JHONATAN RIVERA\AppData\Local\Programs\Python\Python310\lib\site-packages\sklearn\fe
  warnings.warn(

print("Vocabulary size:", len(tfidf.vocabulary_))
✓ 0.0s

Vocabulary size: 24695
```

3.2.2.1 TFIDF con clasificación Random Forest Classifier

Previamente habíamos mencionado el funcionamiento e importancia del algoritmo de Random Forest para clasificación. Ahora haremos la misma implementación pero esta vez con los datos vectorizados con la técnica de TFIDF.

```
tfidf_model = RandomForestClassifier(random_state=3)
[41] ✓ 0.0s

tfidf_model.fit(X_tfidf, y_train)
[42] ✓ 12.3s
```

```
y_train_tfidf_predict = tfidf_model.predict(X_tfidf)
y_test_tfidf_predict = tfidf_model.predict(tfidf.transform(X_test["words"]))
✓ 1.1s
```

```
print("Precision:", precision_score(y_test, y_test_tfidf_predict,average='weighted'))
print("Recall:", recall_score(y_test, y_test_tfidf_predict,average='weighted'))
print("F1:", f1_score(y_test, y_test_tfidf_predict,average='weighted'))
[49] ✓ 0.0s

Precision: 0.468388979973793
Recall: 0.46900384779820437
F1: 0.44700860322570096
```

Podemos ver que obtuvimos métricas muy parecidas a cuando se implementó el algoritmo con BoW, por lo que seguirimos probando con otros modelo para encontrar el mejor modelo posible.

3.2.2.2 TFIDF con clasificación Regresión Logística

Previamente habíamos mencionado el funcionamiento e importancia del algoritmo de Regresión logistica para clasificación. Ahora haremos la misma implementación, pero esta vez con los datos vectorizados con la técnica de TFIDF.

```
logistic_model = LogisticRegression(random_state=5, max_iter=1000)
✓ 0.0s

logistic_model.fit(X_tfidf, y_train)
✓ 1.4s
```

```
y_test_logistic_predict = logistic_model.predict(tfidf.transform(X_test["words"]))
✓ 0.8s
```

```
print("Precision:", precision_score(y_test, y_test_logistic_predict, average='weighted'))
print("Recall:", recall_score(y_test, y_test_logistic_predict, average='weighted'))
print("F1:", f1_score(y_test, y_test_logistic_predict, average='weighted'))
```
✓ 0.0s

```
Precision: 0.4797064240345485
Recall: 0.4848225737494656
F1: 0.4765134290520752
```

Con este modelo obtuvimos las mejores métricas hasta el momento, es un posible candidato como elección final de los modelo.

3.2.2.3 TFIDF con clasificación SVM

Previamente habíamos mencionado el funcionamiento e importancia del algoritmo de SVM para clasificación. Ahora haremos la misma implementación, pero esta vez con los datos vectorizados con la técnica de TFIDF.

```
svm_model = SVC(kernel='linear', random_state=5)
```
✓ 0.0s

```
svm_model.fit(X_tfidf, y_train)
```
✓ 13.3s

```
y_test_svm_predict = svm_model.predict(tfidf.transform(X_test["words"]))
```
✓ 5.4s

```
print("Precision (SVM):", precision_score(y_test, y_test_svm_predict, average='weighted'))
print("Recall (SVM):", recall_score(y_test, y_test_svm_predict, average='weighted'))
print("F1 (SVM):", f1_score(y_test, y_test_svm_predict, average='weighted'))
```
✓ 0.0s

```
Precision (SVM): 0.48057657526578224
Recall (SVM): 0.48182984181274047
F1 (SVM): 0.47870772842148457
```

Con este modelo superamos de nuevo las mejores métricas obtenidas, por lo que se convierte en el principal candidato para el modelo final.

3.2.3 Modelos con Vectorización Doc2Vec. Autor: Jonathan Rivera

Doc2Vec es una técnica de representación de texto que extiende el concepto de Word2Vec para generar vectores de características para documentos completos en lugar de palabras individuales. Utiliza una red neuronal para aprender representaciones vectoriales densas de documentos, lo que permite capturar la semántica y el contexto de todo el documento. En el contexto de análisis de reseñas y clasificación de ratings, Doc2Vec proporciona una representación numérica de las reseñas que conserva la similitud semántica entre ellas, lo que facilita la detección de patrones y la realización de predicciones precisas sobre la calificación de las reseñas basadas en el contenido del texto. Esto permite a los modelos de aprendizaje automático capturar mejor la información contextual y generar clasificaciones más precisas y útiles.

En primer lugar hacemos copia de la división original de datos de entrenamiento y prueba para evitar complicaciones con las transformaciones hechas por las vectorizaciones previas.

```
X_train_d2v, X_test_d2v, y_train_d2v, y_test_d2v = X_train, X_test, y_train, y_test
```
✓ 0.0s

Se implemento el siguiente codigo para preparar los textos para el modelo, entrenando el modelo para comprender y representar los textos de manera numérica, y finalmente convertir los textos en vectores numéricos que reflejan su contenido y significado.

```python
from gensim.models.doc2vec import Doc2Vec, TaggedDocument
def tokenize_and_tag(texts):
    tagged_data = []
    for i, text in enumerate(texts):
        words = word_tokenize(text)
        tags = [i]
        tagged_data.append(TaggedDocument(words=words, tags=tags))
    return tagged_data

# Tokeniza y etiqueta los textos de entrenamiento y prueba
tagged_data_train = tokenize_and_tag(X_train["words"])
tagged_data_test = tokenize_and_tag(X_test["words"])
# Crea un modelo Doc2Vec
doc2vec_model = Doc2Vec(vector_size=100, window=5, min_count=1, workers=4, epochs=20)

# Construye el vocabulario
doc2vec_model.build_vocab(tagged_data_train)

# Entrena el modelo Doc2Vec
doc2vec_model.train(tagged_data_train, total_examples=doc2vec_model.corpus_count, epochs=doc2vec_model.epochs)
# Vectoriza los textos de entrenamiento y prue ba
X_train_d2v['doc2vec_features'] = [doc2vec_model.infer_vector(doc.words) for doc in tagged_data_train]
X_test_d2v['doc2vec_features'] = [doc2vec_model.infer_vector(doc.words) for doc in tagged_data_test]
```
✓ 35.7s

3.2.3.1 Doc2Vec con clasificación Random Forest Classifier

Previamente habíamos mencionado el funcionamiento e importancia del algoritmo de Random Forest para clasificación. Ahora haremos la misma implementación pero esta vez con los datos vectorizados con la técnica de Doc2Vec.

```python
# Modelo con Doc2Vec
doc2vec_model = RandomForestClassifier(random_state=5)
doc2vec_model.fit(list(X_train_d2v['doc2vec_features']), y_train_d2v)
y_train_doc2vec_predict = doc2vec_model.predict(list(X_train_d2v['doc2vec_features']))
y_test_doc2vec_predict = doc2vec_model.predict(list(X_test_d2v['doc2vec_features']))
```
✓ 6.0s

```python
from sklearn.metrics import ConfusionMatrixDisplay
ConfusionMatrixDisplay.from_predictions(y_test, y_test_doc2vec_predict)
from sklearn.metrics import precision_score,recall_score,f1_score
precision_doc2vec = precision_score(y_test, y_test_doc2vec_predict, average='weighted')
recall_doc2vec = recall_score(y_test, y_test_doc2vec_predict, average='weighted')
f1_doc2vec = f1_score(y_test, y_test_doc2vec_predict, average='weighted')

print("Precision:", precision_doc2vec)
print("Recall:", recall_doc2vec)
print("F1:", f1_doc2vec)
```
✓ 0.2s

```
Precision: 0.40711807284821727
Recall: 0.40914920906370245
F1: 0.398830825791305 8
```

Las métricas obtenidas no logran superar al mejor modelo obtenido hasta este momento (TFIDF-SVM), por lo que sera descartado.

3.2.3.2 Doc2Vec con clasificación Regresión Logística

Previamente habíamos mencionado el funcionamiento e importancia del algoritmo de Regresión logistica para clasificación. Ahora haremos la misma implementación, pero esta vez con los datos vectorizados con la técnica de Doc2Vec.

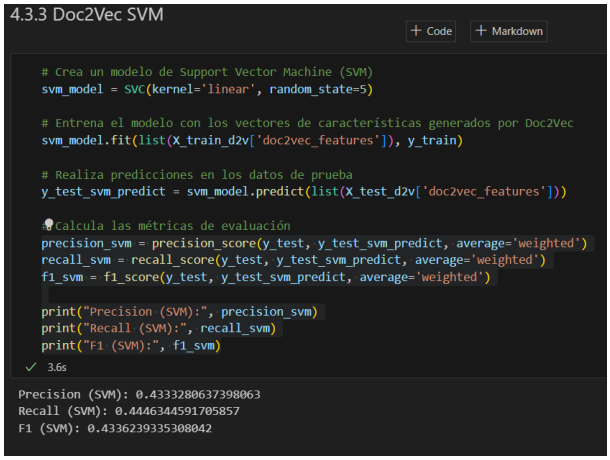
```python
from sklearn.linear_model import LogisticRegression
# Crea un modelo de Regresión Logística
logistic_model = LogisticRegression(random_state=5, max_iter=1000)

# Entrena el modelo con los vectores de características generados por Doc2Vec
logistic_model.fit(list(X_train_d2v['doc2vec_features']), y_train)

# Realiza predicciones en los datos de prueba
y_test_logistic_predict = logistic_model.predict(list(X_test_d2v['doc2vec_features']))

# Calcula las métricas de evaluación
precision_logistic = precision_score(y_test, y_test_logistic_predict, average='weighted')
recall_logistic = recall_score(y_test, y_test_logistic_predict, average='weighted')
f1_logistic = f1_score(y_test, y_test_logistic_predict, average='weighted')

print("Precision:", precision_logistic)
print("Recall:", recall_logistic)
print("F1:", f1_logistic)
```
✓ 0.3s

```
Precision: 0.43419262349327314
Recall: 0.4471996579734929
F1: 0.4363897319117499
```

Si bien las métricas obtenidas no son malas, no superan las de TFIDF-SVM, por lo que este modelo no sera tenido en cuenta.

3.2.3.3 Doc2Vec con clasificación SVM

Previamente habíamos mencionado el funcionamiento e importancia del algoritmo de SVM para clasificación. Ahora haremos la misma implementación, pero esta vez con los datos vectorizados con la técnica de TFIDF.



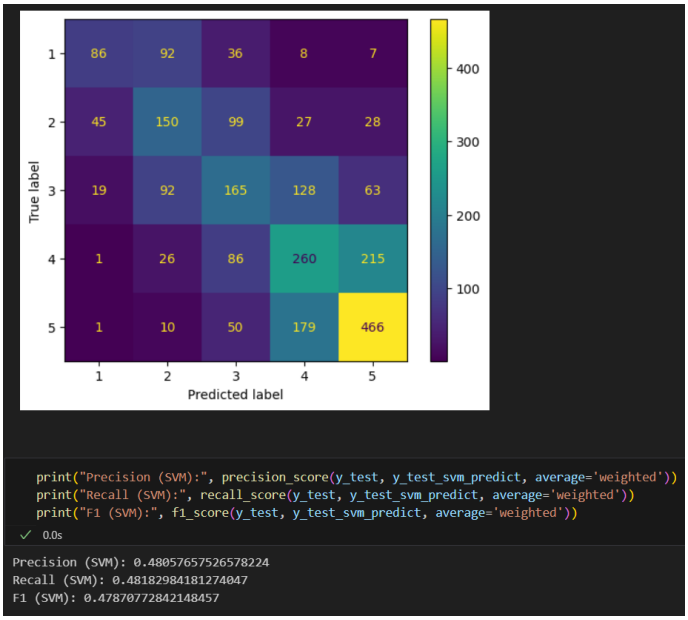
Si bien las métricas obtenidas no son malas, no superan las de TFIDF-SVM, por lo que este modelo no sera tenido en cuenta.

4. Resultados

Análisis

Después de analizar todos los modelos implementados con las diferentes técnicas de vectorización y algoritmos, realizando una comparación entre las métricas obtenidas, encontramos que el mejor de los modelos es TFIDF-SVM. Nos basamos específicamente en la métrica de F1, que combina precisión y sensibilidad, ofreciendo un balance entre la capacidad del modelo para identificar correctamente las reseñas positivas y su habilidad para no ignorar las negativas, al seleccionar el modelo con el mejor puntaje F1, aseguramos que estamos utilizando el enfoque más eficaz para interpretar los sentimientos y opiniones expresados por los turistas, lo cual es fundamental para el éxito de nuestro proyecto en el fomento y mejora del turismo en Colombia.

Entraremos en mas detalle de los resultados obtenidos con el modelo de TFIDF-SVM



Podemos empezar analizando los resultados obtenidos en la matriz de confusión. En primer lugar podemos observar que para cada clase de ratings, la predicción mas frecuente para cada clasificación real, es la misma clase, esto nos permite entender que la clasificación mas probable para cada review predecida es su rating real. Tambien podemos ver que para todas las clases, la mayoría de predicciones se encuentran concentradas en su clase real o en las clases adyacentes (+-1), muy pocas predicciones se desviaron por mas de 1 en diferencia de rating real vs predecido.

En primer lugar, la observación de que las predicciones más frecuentes coinciden con el rating real de las reseñas indica que el modelo tiene una capacidad predictiva sólida y confiable. Esto es fundamental para el negocio, ya que sugiere que el modelo es capaz de capturar efectivamente las características clave de las reseñas que determinan su rating.

Además, al notar que la mayoría de las predicciones se concentran en su clase real o en las clases adyacentes (+-1), se demuestra que el modelo tiene una capacidad razonable para manejar las variaciones sutiles en las opiniones expresadas en las reseñas. Esta capacidad es valiosa para el negocio, ya que permite identificar no solo las reseñas que reciben una calificación alta o baja de manera clara, sino también aquellas que están en el rango intermedio y podrían beneficiarse de intervenciones específicas para mejorar la experiencia turística.

Tambien, las métricas de precisión y recall son indicadores importantes de la calidad del modelo de clasificación SVM aplicado en este contexto de predicción de ratings de reseñas de turismo. La precisión de 0.48 indica que, en promedio, el 48% de las predicciones positivas realizadas por el modelo son realmente correctas. Esto significa que casi la mitad de las veces que el modelo clasifica una reseña como perteneciente a una cierta categoría de rating, esa clasificación es precisa y coincide con la categoría real de la reseña.Por otro lado, el recall de 0.48 indica que el modelo es capaz de identificar correctamente el 48% de todas las reseñas que realmente pertenecen a una categoría de rating específica. En otras palabras, el 48% de las reseñas que realmente tienen un cierto rating fueron correctamente identificadas por el modelo como pertenecientes a esa categoría. Si además consideramos que es bastante probable que las palabras utilizadas para ratings cercanos son prácticamente iguales (por ejemplo, las palabras en un rating de 4 van a ser muy similares a las de un rating de 5), podemos entender que el modelo hace predicciones bastante acertadas.

Estrategias

Identificar reseñas críticas: Dado que el modelo es capaz de capturar las reseñas que están fuera del rango intermedio (+-1), la organización puede priorizar la atención en las reseñas que reciben una calificación alta o baja, ya que es más probable que estas tengan un impacto significativo en la percepción del cliente y en la reputación del negocio

Personalizar la experiencia del cliente: Al comprender mejor las reseñas intermedias, la organización puede identificar oportunidades para personalizar la experiencia del cliente y abordar áreas específicas de mejora. Por ejemplo, si hay un patrón de reseñas intermedias que mencionan aspectos como la limpieza o la atención al cliente, la organización puede implementar medidas específicas para abordar estas preocupaciones y mejorar la satisfacción del cliente.

Optimizar recursos de atención al cliente: Al utilizar el modelo para identificar automáticamente el rating de las reseñas, la organización puede asignar de manera más eficiente los recursos de atención al cliente, priorizando las reseñas que requieren atención inmediata o acción correctiva

Justificación del modelo

La información obtenida de los modelos de clasificación proporciona una base sólida para la toma de decisiones estratégicas en la organización. Al aprovechar estas percepciones, la organización puede optimizar sus procesos, mejorar la experiencia del cliente y mantener una ventaja competitiva en la industria del turismo.

5. Mapa de actores relacionado con el producto de datos creado

Rol dentro de la empresa	Tipo de actor	Beneficio	Riesgo
COTELCO	Usuario-cliente	Promueve el turismo en Colombia y representa a las cadenas hoteleras.	Debido a la naturaleza y métricas del modelo, pueden recibir calificaciones que no corresponden a lo expresado en las reseñas.
Científicos de datos	Proveedor	Garantiza que se cumplan los objetivos de COTELCO, MinComercio y los hoteles por medio de un software bien producido y que brinde buenos resultados a dichos actores.	Manejo incorrecto de los datos/del modelo que puede llevar a la producción de un software de mala calidad.
Ministerio de Comercio, Industria y Turismo	Financiador	Recibe el software que va a ayudarlos a predecir las	Posibles pérdidas económicas si el modelo no

		calificaciones que se les da a sus hoteles dadas unas reseñas en lenguaje natural	convence/no da los resultados posibles.
Hoteles	Beneficiario	Recibe las calificaciones de los hoteles y sitios turísticos luego de pasar las reseñas por el modelo.	Afectaciones en la calidad de las calificaciones que conduciría a menor turismo en determinadas zonas.
Turistas	Beneficiar	Se benefician de las estrategias de mejoras implementadas por los hoteles a partir del modelo planteado	Puede que las estrategias implementadas por los hoteles no tomen en cuenta de manera correcta el conocimiento obtenido a partir del modelo

6. Trabajo en equipo

Integrante	Tareas
Jonathan Rivera	Procesamiento de datos: Normalización (Lematización y stemming), Eliminación de ruido. Modelos: Implementación de los algoritmos con Vectorización Word 2 vec
Julián Castro	Procesamiento de datos: Analisis y limpieza de idiomas en las reviews, Tokenización Modelos: Implementación de los algoritmos con Vectorización TFIDF
Alejandro Gómez	Procesamiento de datos: Perfilamiento y calidad de los datos, análisis de columnas. Modelos: Implementación de los algoritmos con Vectorización BoW

Rol dentro del equipo	Integrantes
Líder del Proyecto	Julián Castro Jonathan Rivera Alejandro Gómez
Líder de Negocio	Jonathan Rivera

	Julián Castro
Líder de Datos	Jonathan Rivera Alejandro Gómez
Líder de Analítica	Alejandro Gómez Julián Castro

Integrante	**Horas**	**Distribución del trabajo**
Jonathan Rivera	34	33%
Julián Castro	34	33%
Alejandro Gómez	34	33%

No se presentaron retos mayores en el trabajo, lo único destacable a mencionar es que hubo pequeños inconvenientes al implementar dos de los algoritmos, pero entre el equipo pudieron ser solucionados. Para mejorar el desempeño para las siguientes entregas, se acordarán horarios más claros para llevar avances y realizar el proyecto paulatinamente.