
 <p>Universidad de los Andes Colombia</p>	<p>Ingeniería de Sistemas y Computación Pregrado ISIS-3301 – Inteligencia de Negocios Primer Proyecto Semestre: 2023-20</p>	
--	---	---

Proyecto 1 – Entrega 1

Integrantes:

Juan Sebastián Sánchez Delgado - js.sanchezd1

Miguel Cárdenas Cárdenas - ma.cardenas

Santiago Paerez Gonzales – s.paeres

Institución:

Universidad De Los Andes

Curso:

Inteligencia de negocios – ISIS3301 (Sección 1)

Profesores:

Fabian Peña

María del Pilar Villamil

Haydemar Nuñez

<p>Oportunidad/problema Negocio</p>	<p>El objetivo principal del presente trabajo es optimizar y automatizar el proceso de clasificación de información en formato textual, según el objetivo del desarrollo sostenible (ODS) involucrado. Todo esto con la finalidad de identificar posibles problemas y soluciones para así concebir nuevos planes para el desarrollo según la zona territorial.</p> <p>Los ODS son principios fundamentales formulados la ONU para mitigar la pobreza y garantizar el respeto a la vida, al planeta y a los derechos de las personas. A continuación, se describen aquellos ODS que serán de interés para este trabajo:</p> <p>3. Garantizar una vida y promover una vida sana para todas las edades: Se necesitan innumerables esfuerzos para eliminar por completo muchos problemas de salud y diferentes patologías, tanto actuales como emergentes. Mediante una financiación más eficaz de los sistemas de salud, un mejor saneamiento y una mayor disponibilidad de personal médico, se pueden lograr avances significativos.</p> <p>4. Garantizar una educación inclusiva, equitativa y de calidad y promover oportunidades de aprendizaje durante toda la vida para todos: Nunca antes habían estado tantos niños fuera de la escuela al mismo tiempo, lo que altera su aprendizaje y cambia drásticamente sus vidas, especialmente las de los niños más vulnerables y marginados.</p> <p>5. Lograr la igualdad entre los géneros y empoderar a todas las mujeres y las niñas: los estatutos legales y tradiciones discriminatorias continúan siendo males arraigados en la sociedad, las mujeres siguen estando poco representadas a nivel político, y el 20% de mujeres de entre 15 y 49 años han padecido algún tipo de violencia o</p>
-------------------------------------	--

abuso de una persona cercana en el último año.

El abordar estos ODS tendría un impacto positivo dentro del contexto colombiano, debido a que guardan relación con problemáticas actuales. Por ejemplo, Colombia, según las pruebas PISA es uno de los países de la OCDE con los peores resultados. El identificar de manera más pertinente que información está más relacionada con un ODS puede ayudar a idear nuevas soluciones para tratar con dichas problemáticas

Enfoque analítico (Descripción del requerimiento desde el punto de vista de aprendizaje automático) e incluya las técnicas y algoritmos que propone utilizar.	<p>El objetivo a nivel tecnológico consiste en primero diseñar e implementar un modelo de clasificación multiclase que se encargue de determinar de manera automática y confiable a que ODS corresponde un texto dado. Tras esto, se busca exportar este modelo y diseñar una aplicación que pueda ser usada dentro de la organización para ser uso de este. Los algoritmos de clasificación a implementar son los siguientes:</p> <ul style="list-style-type: none"> -Bag of Words (BoW) -TF-IDF: -Gradient Boosting
Organización y rol dentro de ella que se beneficia con la oportunidad definida	El modelo a construir está destinado al Fondo de Poblaciones de las Naciones Unidas (UNFPA), la cual es una entidad que se encarga de hacer un seguimiento exhaustivo de las políticas públicas para posteriormente evaluar el potencial impacto social de estas. Los roles que más se beneficiaran del presente proyecto son aquellos relacionados con estrategias y planeación, principalmente los que se encargan del territorio colombiano.
Contacto con experto externo al proyecto	PENDIENTE

Tabla 1. Entendimiento del negocio y enfoque analítico.

1. Entendimiento y preparación de los datos (20%)

Antes de proceder a la etapa de modelamiento, es indispensable realizar un perfilamiento adecuado y analizar las diferentes dimensiones de calidad de los datos (unicidad, completitud, consistencia y validez). Esto con el objetivo de entender de mejor manera el contenido, la estructura y la idoneidad de los datos para llevar a cabo el proyecto en cuestión.

En primer lugar, dentro del contexto de análisis de texto es importante asegurarse que la codificación sea la adecuada para el idioma textual de interés. Para ello, se exporto un archivo CSV desde Excel y posteriormente se importó en un Jupyter Notebook aclarando que el separador a usar es el signo ‘;’. Posteriormente, como parte de la exploración de los datos se averiguaron algunos aspectos como: El número de registros y columnas, los tipos de cada una de las variables, el contenido de algunas filas, etc (Figura 1). A partir de lo anterior, se puede afirmar lo siguiente:

- El hecho de que sean 3000 registros supone que la cantidad de datos es la adecuada para las tareas de entrenamiento.

- Como solo 2 variables, por la naturaleza del objetivo de negocio, resulta evidente que la única variable de entrenamiento es 'textos español', mientras que 'sdg' es la variable objetivo para el proceso de clasificación.
- Con respecto a los tipos de las variables, 'sdg' representa el tipo de ODS involucrado en el texto, es decir, que es una variable categórica numérica nominal. Por otro lado, 'textos español' es una variable de tipo objeto que contiene la totalidad de los textos.

```
[4] df_ods = pd.read_csv("./cat_345.csv", sep=';', encoding = 'utf8')

[5] df_ods.shape

(3000, 2)

[46] df_ods.dtypes

Textos_espanol    object
sdg               int64
dtype: object

df_ods.sample(5)
```

	Textos_espanol	sdg
406	Una vez más, los programas que enfatizan la ed...	3
2910	Estos resultados tienen algunas implicaciones ...	5
1258	Tras las prácticas, los estudiantes informan a...	4
2670	Implica que ella da una compensación a su espo...	5
914	Ambos se enfrentan a retos complejos, en los q...	3

Figura 1. Perfilamiento de los datos

En lo referente a las dimensiones de calidad, al consultar tanto el número de valores nulos como de registros duplicados no se identificó ningún caso, por lo cual, no es necesario tomar ninguna medida en particular para garantizar la completitud y unicidad de los datos. En cuanto a la consistencia, debido a que no existen otras tablas de datos aunado al hecho de que solo se cuentan con 2 variables sin aparente dependencia semántica, implica que aparentemente no se presenta ningún tipo de inconsistencia en los datos. Sin embargo, al considerar las reglas de negocio, es posible percatarse que la variable 'textos español' deber ser de tipo String y que 'sdg' debería llamarse 'ODS', por lo tanto, sí que existen algunas inconsistencias a corregir.

Por último, a pesar de que se especificó el tipo de codificación para albergar los caracteres del idioma español, existen caracteres extraños que no forman parte del idioma como tal y por ende representan un problema de validez. Este problema se puede deber a que cuando se carga inicialmente un archivo CSV en Excel se utiliza una codificación por defecto. Una alternativa para resolver dicho inconveniente es simplemente reemplazar los caracteres erróneos por sus representaciones validas en UTF-8 como se muestra en la Figura 2.

```
[16] # Reemplaza lo caracteres erroneos con su equivalencia pero en UTF-8
df_ods['Textos_espanol'] = df_ods['Textos_espanol'].str.replace('Ã', 'á')
df_ods['Textos_espanol'] = df_ods['Textos_espanol'].str.replace('Ã©', 'é')
df_ods['Textos_espanol'] = df_ods['Textos_espanol'].str.replace('Ã³', 'ó')
df_ods['Textos_espanol'] = df_ods['Textos_espanol'].str.replace('Ãº', 'ú')
df_ods['Textos_espanol'] = df_ods['Textos_espanol'].str.replace('Ã±', 'ñ')
df_ods['Textos_espanol'] = df_ods['Textos_espanol'].str.replace('Ã', 'í')

[17] df_ods[df_ods['Textos_espanol'].str.contains('Ã')]
```

Textos_espanol	sdg
----------------	-----

Figura 2. Validez de los datos

Para la aplicación de técnicas de NLP, es imprescindible transformar las representaciones textuales en numéricas para que sean entradas validas. Este paso supone uno de los más importantes ya que determinará los algoritmos que podrán ser empleados e incluso influirá en los resultados reportados por los modelos. Para el presente proyecto, se aplicó un proceso de tokenizacion (División de un texto en unidades más pequeñas) por palabra en minúsculas eliminando las palabras de parada o vacías, ya que estas no traen consigo un peso semántico significativo, pero si aumentan el vocabulario y por ende la complejidad temporal.

Tras haber obtenido los tokens correspondientes, se empleó una técnica denominada Bag of Words, la cual es una manera de representar las palabras de manera numérica sin considerar el orden de aparición (Figura 3). Finalmente, se dividen los datos en entrenamiento y prueba con una proporción de 80-20.

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

[27] X_bow = bow.fit_transform(X_train)

[28] print("Vocabulary size:", len(bow.vocabulary_))

Vocabulary size: 15134

[29] tfidf = TfidfVectorizer(tokenizer=word_tokenize, stop_words=stop_words, lowercase=True)

[30] X_tfidf = tfidf.fit_transform(X_train)

[31] print("Vocabulary size:", len(tfidf.vocabulary_))

Vocabulary size: 15134
```

Figura 3. Tokenizacion y aplicación de BoW

2. Modelado y evaluación (25%)

3.1 Entrenando Modelo con BoW y Random Forest classifier (Juan Sebastian Sanchez Delgado)

Como primer algoritmo a implementar se optó por utilizar Random Forest Classifier, el cual consiste en combinar varios clasificadores débiles como los son los árboles de decisión para aumentar la capacidad de predicción del modelo. Con Random Forest se tiene a disposición un hiperparametro adicional para variar el número de árboles a entrenar, con lo cual el modelo resulte es más robusto y suele reportar mejores resultados que otras alternativas. También, como este que se deriva del algoritmo de árboles de decisión, es más fácil de interpretar el modelo.

Como entrada del algoritmo se recibe el BoW obtenido durante la etapa de transformación de datos y se especifica un numero de estados aleatorios para entrenar de manera iterativa el modelo un número determinado de veces para así obtener mejores resultados. En este caso se decidió asignarle un valor de dos a este parámetro, tal como se muestra en la siguiente imagen (Figura 4).

```
[32] bow_model = RandomForestClassifier(random_state=2)

[33] bow_model.fit(X_bow, y_train)
```

RandomForestClassifier
RandomForestClassifier(random_state=2)

Figura 4. Entrenamiento utilizando Random Forest

Una buena forma de estimar de manera cualitativa que tan acertado resulta un modelo de esta índole consiste en averiguar qué tan relevante son ciertas palabras para clasificar cada uno de los textos. En la figura 5 se muestra un gráfico de barras mostrando la importancia de algunas palabras. Si bien existen algunos casos en los que la palabra en cuestión si guarda una estrecha relación con la clasificación, existen otros en los que no sucede esto. Lo anterior puede estar sucediendo debido a que son palabras poco comunes que justamente siempre aparecen en el mismo tipo de ODS.

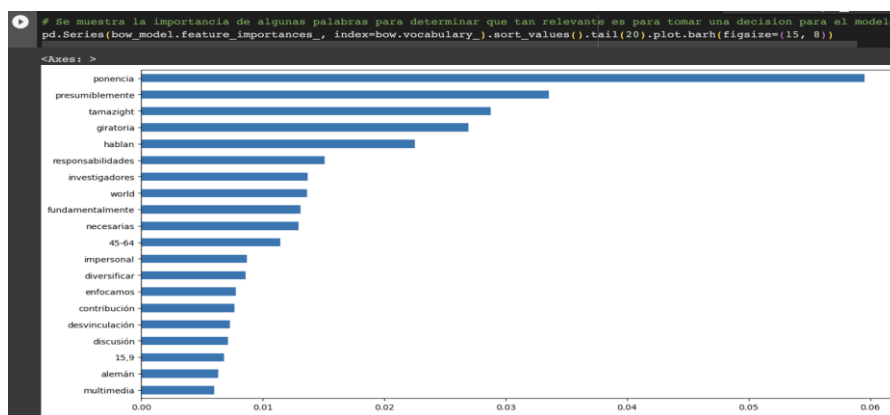


Figura 5. Importancia de las palabras en la clasificación

Una buena práctica para evaluar que tan efectivo resulta el modelo es analizar los estimadores o hiperparámetros, para así reconocer las potenciales causas del comportamiento del mismo. En este caso, el número de árboles fue de 100, dicha cifra se encuentra dentro del rango óptimo (64 a 128), por lo que los tiempos de entrenamiento y el riesgo de sobreajuste no serán tan altos. Otro hiperparámetro de interés es la profundidad de los árboles de decisión, como esta difiere de árbol a árbol, se estimó la profundidad promedio de estos, obteniendo así un valor de 97,17. Al ser la profundidad promedio algo elevada, existe la posibilidad de que el modelo sufra de sobreajuste.

Como parte del análisis cuantitativo, se construyeron las matrices de confusión a partir de los distintos tipos de datos (entrenamiento y prueba). Tras esto, se calcularon las diferentes métricas de calidad. Aunque para los datos de entrenamiento todos los textos fueron clasificados correctamente, la precisión, el recall y el F1 de modelo fueron aproximadamente iguales a 0.976 para los datos de prueba, por consiguiente,

no se presenta el fenómeno de sobreajuste. A continuación, en la Figura 6 se presenta la matriz de confusión correspondiente a este segundo tipo de datos:

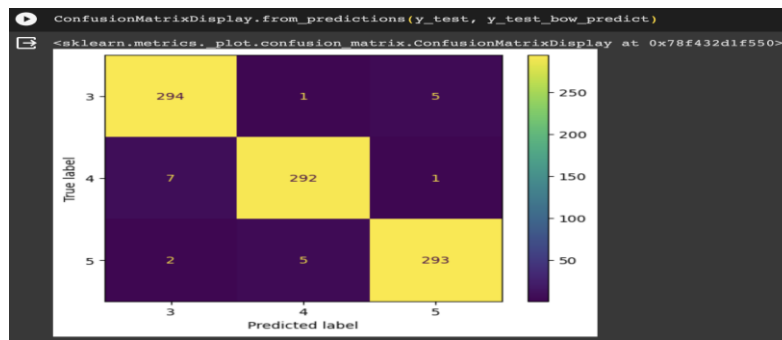


Figura 6. Matriz de confusión para datos de prueba

3.2 Entrenando el modelo con Term Frequency-Inverse Document Frequency (Santiago Paeres Gonzalez)

Para el segundo algoritmo de clasificación de texto basado en machine learning, se empleó el método **TF-IDF** (term frequency-inverse document frequency). Este enfoque cuantifica la relevancia de una palabra en un documento dentro de una colección de documentos (corpus). Es una métrica comúnmente utilizada como un factor de peso en tareas de recuperación de información y minería de texto. El valor de TF-IDF aumenta en proporción al número de veces que una palabra aparece en un documento, pero se equilibra con la frecuencia de la palabra en toda la colección de documentos, lo que permite abordar la variabilidad en la frecuencia de palabras en general.

- **Definición Matemática:**
- Term frequency $tf(t, d)$ se puede definir de la siguiente manera:

$$tf(t, d) = \frac{f(t, d)}{\max\{f(t, d) : t \in d\}}$$

- **Donde:**

1. $f(t, d)$ es el número de veces que aparece el termino t en el documento d . Es una función binaria definida en el domino $\{0, 1\}$, donde si t está en el documento d la función es igual a 1 y vale 0 en caso contrario.
 2. Frecuencia normalizada, para evitar una predisposición hacia los documentos largos. Por ejemplo, se divide la frecuencia bruta por la frecuencia máxima de algún término t en el documento.
- Inverse Document Frequency $idf(t, D)$ es una función que indica si el término t es común o no en la colección de documentos (corpus) D . Se puede definir como la siguiente función:

$$idf(t, D) = \log \frac{|D|}{|\{d : D : t : d\}| + 1}$$

- **Donde:**

1. $|D|$ es la cardinalidad del conjunto de documentos, es decir el número de documentos en la colección D (corpus).
2. $|\{d : D : t : d\}|$ es el número de documentos donde aparece el término t. Si el término no está en la colección se producirá una división por cero. Por lo tanto, es común ajustar esta fórmula a $|\{d : D : t : d\}| + 1$. De acuerdo con la documentación, Scikit-Learn hace este ajuste.

Por tanto, se puede definir a TF-IDF como el producto de las funciones:

$$tfidf(t, d, D) = tf(t, d) \times idf(t, D)$$

Para aplicar este algoritmo en el caso estudiado en el presente proyecto, en primer lugar, se definió un TfidfVectorizer. Para usar este vector se definieron los parámetros de stop_words, lowercase y tokenizer.

En este caso los features son iguales a vocabulario de palabras con las que el modelo se entrenara.

```
tfidf = TfidfVectorizer(tokenizer=word_tokenize, stop_words=stop_words, lowercase=True)
Executed at 2023.10.15 19:42:16 in 1s 271ms
```

```
X_tfidf = tfidf.fit_transform(X_train)
Executed at 2023.10.15 19:42:25 in 7s 473ms
```

```
print("Vocabulary size:", len(tfidf.vocabulary_))
Executed at 2023.10.15 19:42:25 in 102ms
```

Vocabulary size: 15134

En la imagen superior se puede ver como el vectorizador TF-IDF se utiliza para convertir texto en vectores numéricos basados en la frecuencia de las palabras en los documentos y su importancia en un corpus.

Es un método de scikit-learn que combina dos operaciones en una. Primero, fit se utiliza para "ajustar" el vectorizador TF-IDF a los datos de entrenamiento, lo que significa que el vectorizador aprende a partir de los datos y calcula estadísticas

importantes como las frecuencias de las palabras y los valores TF-IDF. Luego, transform se utiliza para aplicar esas estadísticas aprendidas a los datos de entrenamiento para realizar la transformación.

Este vectorizador TF-IDF genera una matriz de elementos donde cada fila representa un documento del conjunto de datos original (corpus), y cada columna corresponde a una palabra o término. Por tanto, el valor de cada celda en la matriz es el valor del TF-IDF del término y documento en cuestión. En la siguiente imagen se pueden ver las primeras 100 filas y columnas de la matriz que genera el vectorizador:

```

1 # Obtén una porción limitada de la matriz X_tfidf
2 filas_a_mostrar = 100
3 columnas_a_mostrar = 100
4 ejemplo = X_tfidf[:filas_a_mostrar, :columnas_a_mostrar]
5
6 # Convierte la porción limitada en un DataFrame de Pandas para una mejor visualización
7 df_ejemplo = pd.DataFrame(ejemplo.toarray(), columns=tfidf.get_feature_names_out()[:columnas_a_mostrar])
8 df_ejemplo

```

Executed at 2023.10.15 02:02:58 in 8ms

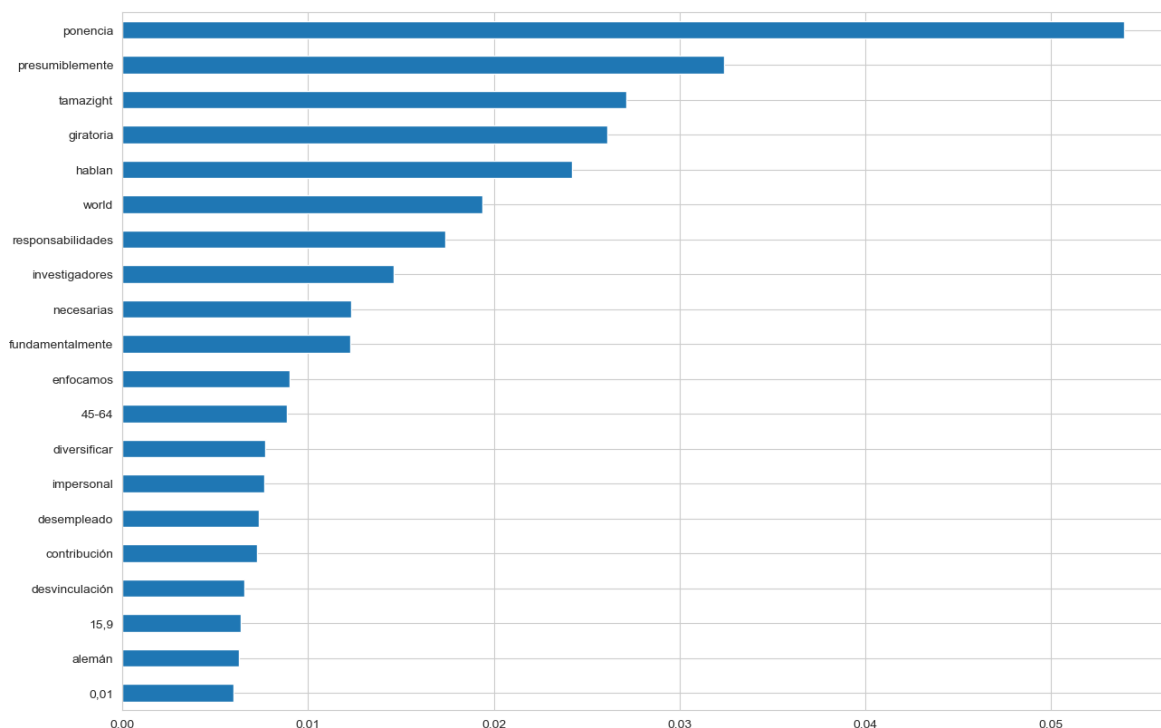
	!	#	\$	%	&	'	''	'barómetro	'club	'conocimiento	'cuando	'deberian
0	0.0	0.0	0.0	0.000000	0.0	0.0	0.00000	0.0	0.0	0.0	0.0	0.
1	0.0	0.0	0.0	0.000000	0.0	0.0	0.00000	0.0	0.0	0.0	0.0	0.
2	0.0	0.0	0.0	0.000000	0.0	0.0	0.00000	0.0	0.0	0.0	0.0	0.
3	0.0	0.0	0.0	0.000000	0.0	0.0	0.00000	0.0	0.0	0.0	0.0	0.
4	0.0	0.0	0.0	0.000000	0.0	0.0	0.00000	0.0	0.0	0.0	0.0	0.
...
95	0.0	0.0	0.0	0.000000	0.0	0.0	0.00000	0.0	0.0	0.0	0.0	0.
96	0.0	0.0	0.0	0.091407	0.0	0.0	0.00000	0.0	0.0	0.0	0.0	0.
97	0.0	0.0	0.0	0.000000	0.0	0.0	0.05927	0.0	0.0	0.0	0.0	0.
98	0.0	0.0	0.0	0.000000	0.0	0.0	0.00000	0.0	0.0	0.0	0.0	0.
99	0.0	0.0	0.0	0.000000	0.0	0.0	0.00000	0.0	0.0	0.0	0.0	0.

Es importante notar que de acuerdo con la documentación de Scikit-Learn los vectores TF-IDF resultantes se normalizan mediante la norma euclidiana:

$$v_{norm} = \frac{v}{||v||_2} = \frac{v}{\sqrt{(v_1)^2 + (v_2)^2 + \dots + (v_n)^2}}$$

Donde v es cada uno de los vectores y n es el número total de vectores. Esta normalización es importante pues anqué originalmente se trataba de un esquema de ponderación de términos desarrollado para la recuperación de información (como una función de clasificación para los resultados de los motores de búsqueda), también ha encontrado un buen uso en la clasificación y agrupación de documentos como es el caso del presente proyecto.

Al igual que con los otros algoritmos se utilizó el clasificador Random Forest para realizar el trabajo clasificación. De igual modo, se comprobó la relevancia de las palabras pertenecientes al corpus en la clasificación de los textos que se pretenden evaluar, para ver cuán significativas son estas palabras en la tarea de clasificación. La presente imagen presenta un gráfico de barras que ilustra la importancia o peso de estas palabras en dicho contexto:



Es importante notar que existe una marcada diferencia en la importancia de las palabras respecto a los otros algoritmos, esto puede deberse a la forma en la que TF-IDF funciona y la forma en la que mide la frecuencia de los términos. De este modo se puede observar que el modelo ignora las palabras con mucha frecuencia que poco aportan al objetivo de clasificación de los textos según el tipo de ODS.

Al mismo tiempo, el modelo al dar poca importancia a las palabras más repetitivas ayudo a reducir la media de profundidad de los árboles de búsqueda en el Random Forest, con un valor para la media de profundidad de 94,37. Aunque desde un análisis más global este aún sigue siendo un valor alto que parece indicar sobrestimación del modelo lo que podría afectar la predicción de la clasificación de textos:

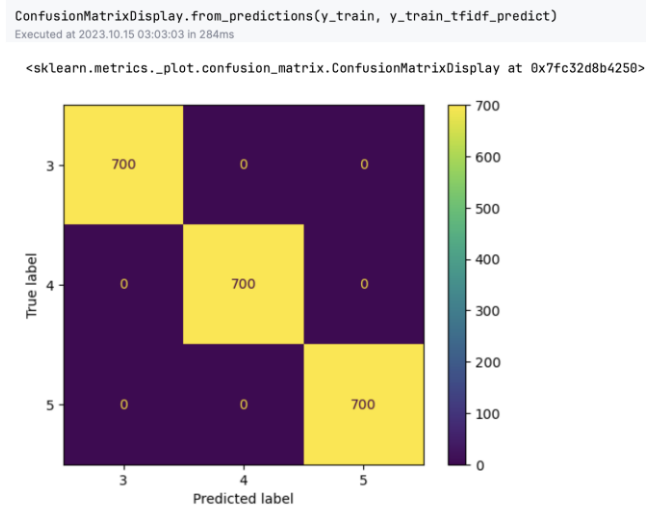
```
tfidf_estimators = tfidf_model.estimators_
print("Number of trees:", len(tfidf_estimators))
print("Trees depth (mean):", np.mean([tree.get_depth() for tree in tfidf_estimators]))
Executed at 2023.10.15 02:44:22 in 20ms
```

```
Number of trees: 100
Trees depth (mean): 94.37
```

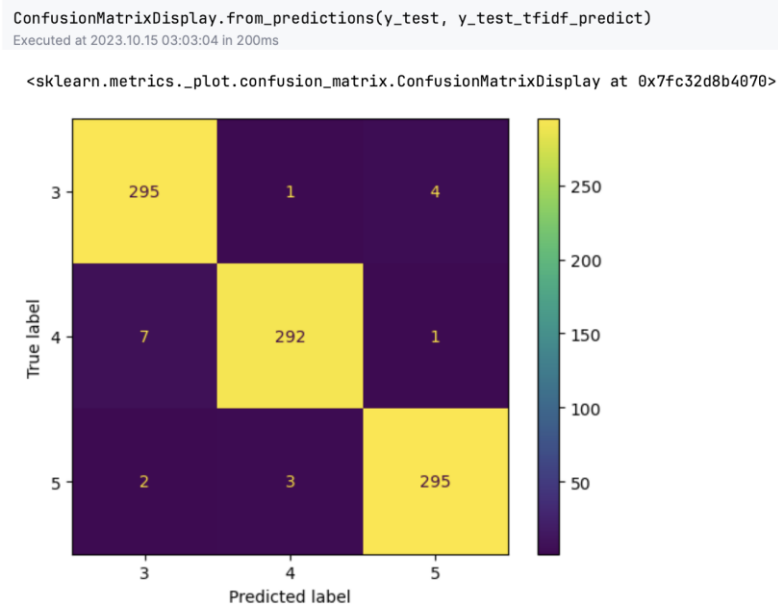
Se puede observar que el número de árboles es igual a 100 porque este es el valor por defecto para Random Forest en Scikit-Learn. Es importante mencionar que, aunque este sea el valor por defecto, no es necesariamente el valor óptimo y puede existir uno mejor.

Para encontrar los valores óptimos es necesario hacer una búsqueda más extensa como RandomizedSearchCV y así poder probar diferentes hiperparametros para encontrar valores óptimos, tanto para el caso de profundidad y numero de árboles, como para todos los demás hiperparametros.

Respecto al análisis cualitativo del modelo, se puede concluir que todos los datos de entrenamiento fueron predecidos correctamente como se puede ver en la siguiente matriz de confusión:



También se realizó una matriz de confusión para los datos de prueba donde se puede observar que la gran mayoría de documentos fueron clasificados correctamente (ya sea true positive o true negative) y muy pocos datos fallaron en la clasificación (ya sea por False Positive o False Negative):



Además de la matriz presentada se puede ver que la precisión, el recall (recuperación) y la puntuación F1 (F1-Score) todos tienen un valor de 0,98 para los datos de prueba. Por tanto, como TF-IDF es el algoritmo con mayor puntuación esto sugiere que este algoritmo con los hiperparámetros escogidos es la mejor opción para realizar la calificación de los textos y poder cumplir con los objetivos planteados.

```
print("Precision:", precision_score(y_test, y_test_tfidf_predict, average='micro'))
print("Recall:", recall_score(y_test, y_test_tfidf_predict, average='micro'))
print("F1:", f1_score(y_test, y_test_tfidf_predict, average='micro'))
```

Executed at 2023.10.15 03:10:04 in 85ms

Precision: 0.98

Recall: 0.98

F1: 0.98

3.2.1 Escogiendo un posible mejor modelo:

Para poder llegar a un posible mejor modelo que pruebe con diferentes combinaciones de hiperparametros se utilizó un pipeline con el algoritmo de búsqueda RandomizedSearchCV:

```
pipeline = Pipeline(steps = [
    ("vectorizer", CountVectorizer(tokenizer=word_tokenize, stop_words=stop_words, lowercase=True)),
    ("classifier", RandomForestClassifier(random_state=4))
])
```

Executed at 2023.10.15 03:22:06 in 38ms

```
param_grid = {
    "vectorizer": [TfidfVectorizer(max_df=0.5, tokenizer=word_tokenize, stop_words=stop_words)],
    "vectorizer__lowercase": [True, False],
    "classifier__n_estimators": [30, 50],
    "classifier__max_depth": [25, 50]
}
```

Executed at 2023.10.15 03:22:07 in 40ms

```
search = RandomizedSearchCV(pipeline, param_grid, n_iter=15, scoring=["precision", "recall", "f1"], refit="f1", cv=30,
    return_train_score=True, verbose=1, random_state=5)
```

Executed at 2023.10.15 03:22:08 in 26ms

```
search.fit(X_train, y_train)
```

Executed at 2023.10.15 03:33:37 in 11m 28s 162ms

Fitting 30 folds for each of 8 candidates, totalling 240 fits

Como se puede ver en la imagen superior Randomized Search busca en el conjunto de hiperparametros definido en param_grid la posible mejor combinación de estos para producir el Randomized Forest optimo. Por tanto, se están realizando 30 iteraciones de validación cruzada (folds) para cada una de las 8 combinaciones de hiperparámetros candidatos. La validación cruzada implica dividir el conjunto de datos de entrenamiento en varios pliegues (folds), y el modelo se entrena y evalúa repetidamente en diferentes subconjuntos de datos para evaluar su rendimiento de manera robusta.

En total, se realizaron 240 ajustes (fits) del modelo durante todo el proceso de búsqueda de hiperparámetros. Este número se obtiene multiplicando el número de folds (30) por el número de candidatos de hiperparámetros (8).

Luego de ejecutar la búsqueda se encuentran los hiperparametros óptimos en este caso con una profundidad de arbol de 25, 30 árboles en el bosque y un random state de 4. Como se puede ver a continuación:

```
search.best_estimator_
```

Executed at 2023.10.15 03:50:09 in 50ms

```
Pipeline(steps=[('vectorizer',  
                 TfidfVectorizer(max_df=0.5,  
                                stop_words=['de', 'la', 'que', 'el', 'en', 'y',  
                                             'a', 'los', 'del', 'se', 'las',  
                                             'por', 'un', 'para', 'con', 'no',  
                                             'una', 'su', 'al', 'lo', 'como',  
                                             'más', 'pero', 'sus', 'le', 'ya',  
                                             'o', 'este', 'sí', 'porque', ...],  
                                tokenizer=<function word_tokenize at 0x7fc348d26700>)),  
                 ('classifier',  
                  RandomForestClassifier(max_depth=25, n_estimators=30,  
                                         random_state=4))])
```

Luego de tener el conjunto de hiperparametros que se considera optimo se vuelve a entrenar un randomized forest con esto hiperparametros:

```
y_train_search_predict = search.best_estimator_.predict(X_train)  
y_test_search_predict = search.best_estimator_.predict(X_test)
```

Executed at 2023.10.15 03:50:12 in 1s 939ms

Finalmente se pueden ver las métricas obtenidas para el nuevo Randomized Forest generado con los hiperparametros óptimos encontrados por la búsqueda:

```
print("Precision:", precision_score(y_test, y_test_search_predict, average='micro'))  
print("Recall:", recall_score(y_test, y_test_search_predict, average='micro'))  
print("F1:", f1_score(y_test, y_test_search_predict, average='micro'))
```

Executed at 2023.10.15 03:50:14 in 24ms

```
Precision: 0.9533333333333334  
Recall: 0.9533333333333334  
F1: 0.9533333333333334
```

Como se puede observar en la imagen superior al usar la búsqueda de RandomizedSearchCV no se obtuvieron mejores métricas que sin usar el algoritmo de RandomizedSearchCV con el conjunto de hiperparametros escogidos para buscar, pues tanto la precisión, el recall y la puntuación F1 están alrededor de 0,95, mientras sin usar la búsqueda RandomizedSearchCV estaba alrededor de 0,98.

3.3 Entrenando el modelo con Gradient Boosting (Miguel Angel Cardenas Cardenas)

El ultimo algoritmo utilizado es Gradient Boosting, un algoritmo utilizado en problemas de clasificación y regresión. Al igual que Random Forest, combina múltiples modelos débiles, específicamente arboles de decisión para formar modelos más robustos, fuertes y precisos. Asimismo, cuenta con un hiperparametro que indica el número de árboles a entrenar, lo que permite construir un modelo mucho más exacto, brindado entonces alta precisión, adaptabilidad y alta dimensionalidad.

Al igual que en algoritmo Random Forest, al Gradient Boosting se le pasa como parametro el BoW definido en la etapa de transformación, y se le asigna un valor de `random_state = 42` como se indica a continuación:

```
gb_classifier = GradientBoostingClassifier(n_estimators=100, learning_rate=0.1, random_state=42)
✓ 0.0s

gb_classifier.fit(X_bow, y_train)
✓ 34.1s

* GradientBoostingClassifier
GradientBoostingClassifier(random_state=42)
```

Figura 7. Entrenamiento utilizando Gradient Boosting.

Se llevo a cabo el mismo proceso que en el random forest y TF-IDF, estimando de forma cuantitativa la incidencia de ciertas palabras en la predicción del tipo de ODS. Los resultados son los siguientes:

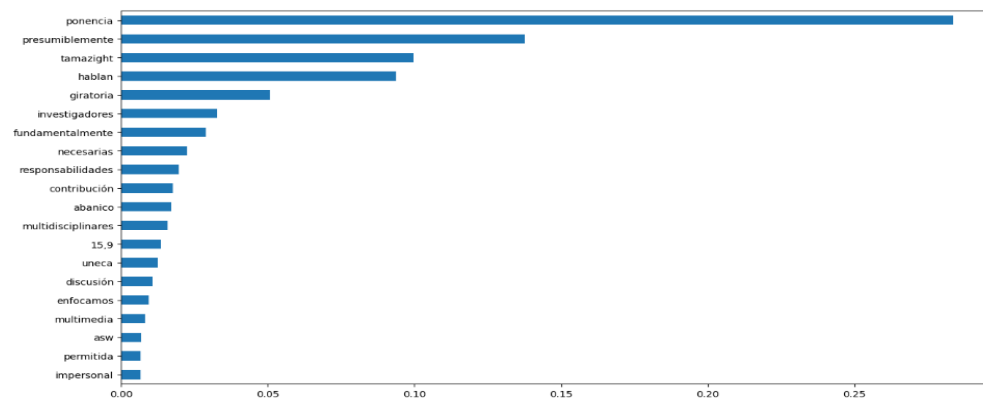


Figura 8. Entrenamiento utilizando Gradient Boosting.

Así mismo, se construyeron las matrices de confusión a partir de los datos de entrenamiento y modelo, en las que fue posible concluir que el modelo construido predice correctamente casi para todos los valores de los datos de prueba. La matriz indica un porcentaje pequeño de valores que fueron clasificados de forma incorrecta en False Positive o False Negative, por ende, podría indicar que el modelo es bueno y no se encuentra sobre ajustado.

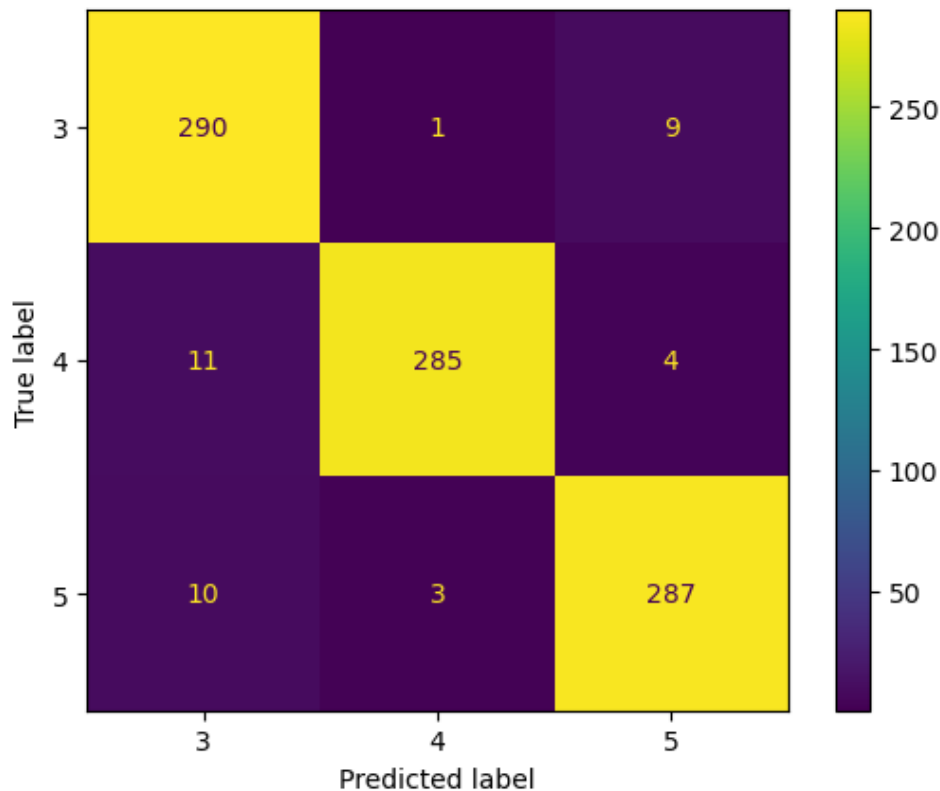


Figura 9. Matriz de confusión para datos de prueba del modelo Gradient Boosting.

Finalmente, calcularon los valores de precisión, recall (recuperación) y puntuación F1 (F1-Score), indicando entonces un valor para de 0.95777777. Si bien es un valor concreto, que indica un modelo eficiente y que podría ser utilizado en la clasificación de los textos, también es posible evidenciar que es el modelo que presenta los menores valores, y por ende, podría ser descartado como modelo final de clasificación.

```
print("Precision:", precision_score(y_test, y_test_gb_predict, average='micro' ))
print("Recall:", recall_score(y_test, y_test_gb_predict, average='micro'))
print("F1:", f1_score(y_test, y_test_gb_predict, average='micro'))

Precision: 0.9577777777777777
Recall: 0.9577777777777777
F1: 0.9577777777777777
```

Figura 10. Precisión, Recall y F1-Score para el modelo Gradient Boosting.

3. Resultados (15%)

3.1 Escogiendo el Modelo más adecuado:

Aunque en un escenario ideal se podría encontrar el modelo de clasificación óptimo probando diferentes combinaciones de hiperparámetros, tanto como para los

vectorizadores como para el Random Forest, para los tres algoritmos propuestos (Bag of Words, TF-IDF y Gradient Boosting), fue evidente en el intento en la sección 3.2.1 de que esta tarea es sumamente tediosa y conlleva un gasto computacional significativo por la forma en la que los algoritmos como RandomizedSearchCV o la optimización Bayesiana funcionan.

Además, existen muchísimas posibles combinaciones de conjuntos de hiperparámetros que se le pueden pasar a RandomizedSearchCV por lo que no es tan fácil encontrar el conjunto de hiperparámetros óptimo.

Sin embargo, con los hiperparámetros escogidos en el pipeline y utilizando RandomizedSearchCV NO se llegó a un mejor resultado que el utilizado originalmente, pues sin hacer uso de RandomizedSearchCV la puntuación F1 del modelo de TF-IDF fue de 0.98, mientras que con RandomizedSearchCV fue de 0.95 para los datos de prueba. Claramente, se pudieron haber escogido mejores conjuntos de hiperparámetros para probar en la búsqueda, pero esto va más allá del alcance de este proyecto, además se llegó a valores aceptables de precisión, recuperación (recall) y puntuación F1.

Así pues, comparando los tres modelos escogidos en sus métricas de precisión, recall (recuperación) y puntuación F1:

	Modelo	Precisión	Recall	F1 Score
0	BoW	0.976667	0.976667	0.976667
1	TF-IDF	0.980000	0.980000	0.980000
2	Gradient Boosting	0.957778	0.957778	0.957778

Por tanto, de acuerdo a la tabla superior el modelo más adecuado para realizar el análisis de clasificación, de acuerdo a cada una de las métricas es TF-IDF pues tiene los mayores valores para la precisión, el recall y la puntuación F1.

3.2 Análisis de las predicciones

Para poder realizar las predicciones se creó un pipeline con dos etapas, en la primera se estableció el vectorizador de TF-IDF para poder vectorizar los datos y en el segundo se estableció una etapa de modelo de clasificación basada en el modelo escogido para realizar las predicciones-IDF

```
# Crear el pipeline
pipeline_tfidf = Pipeline([
    ('tfidf', tfidf), # Etapa de TF-IDF
    ('classifier', tfidf_model) # Etapa de modelo de clasificación
])

pipeline_tfidf.fit(X_train, y_train)

Pipeline(steps=[('tfidf',
                  TfidfVectorizer(stop_words=['de', 'la', 'que', 'el', 'en', 'y',
                                              'a', 'los', 'del', 'se', 'las',
                                              'por', 'un', 'para', 'con', 'no',
                                              'una', 'su', 'al', 'lo', 'como',
                                              'más', 'pero', 'sus', 'le', 'ya',
                                              'o', 'este', 'sí', 'porque', ...],
                                  tokenizer=<function word_tokenize at 0x7fc348d26708>)),
                ('classifier', RandomForestClassifier(random_state=2))])
```

Luego se entrenó el modelo del pipeline con los mismos datos de entrenamiento con el que fue entrenado el modelo de TF-IDF, obteniendo un pipeline que puede ser persistido para luego realizar las predicciones.

```
pipeline_tfidf.fit(X_train, y_train)
Executed at 2023.10.15 19:42:59 in 9s 595ms

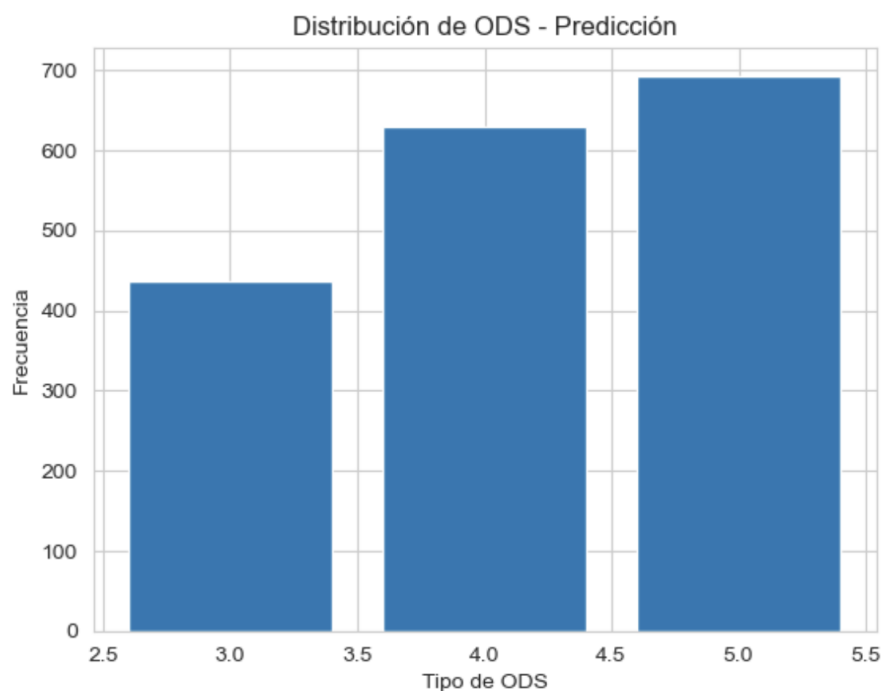
Pipeline(steps=[('tfidf',
                  TfidfVectorizer(stop_words=['de', 'la', 'que', 'el', 'en', 'y',
                                              'a', 'los', 'del', 'se', 'las',
                                              'por', 'un', 'para', 'con', 'no',
                                              'una', 'su', 'al', 'lo', 'como',
                                              'más', 'pero', 'sus', 'le', 'ya',
                                              'o', 'este', 'sí', 'porque', ...],
                                  tokenizer=<function word_tokenize at 0x7fc348d26700>)),
                 ('classifier', RandomForestClassifier(random_state=2))])
```

Aunque no es posible conocer la precisión de las predicciones realizadas, pues los datos no están etiquetados, por lo que no es posible conocer las métricas de evaluación, como la precisión, la recuperación y la puntuación F1, entre otras, porque no existen datos etiquetados con los cuales comparar las predicciones realizadas, se puede decir con cierto grado de confianza que se escogió el mejor modelo de acuerdo a las métricas obtenidas.

A continuación, se puede ver una muestra de 10 registros de las predicciones realizadas:

10 rows × 2 columns pd.DataFrame		
	Textos_español	sdg
811	Primero, el programa de Fortalecimiento de la ...	4
1396	Las brechas entre hombres y mujeres difieren c...	5
1073	Los talentos y perspectivas combinados de homb...	5
149	Las encuestas incluyen una serie de preguntas ...	4
578	¿Y cuál deberían usar los maestros con qué pro...	4
1443	La calidad no se ha dejado de lado, como lo de...	3
1748	Tal cambio de punto de vista puede explicar la...	3
1639	Por ejemplo, los proyectos de ley liderados po...	5
785	En la India, el 60,9 por ciento de la població...	3
1262	Dado que normalmente no se debe brindar atenci...	3

Adicionalmente, es posible ver como se distribuyen los documentos a los cuales se les realizó la predicción y es posible ver que los datos se distribuyen entre las 3 categorías de ODS esperadas (Categoría 3, Categoría 4 y Categoría 5):



4. Mapa de actores relacionado con un producto de datos (10%)

Rol dentro de la empresa	Tipo de actor	Beneficio	Riesgo
Analistas de Datos de UNFPA	Proveedor	Facilitar el seguimiento y análisis de opiniones de habitantes locales sobre problemáticas específicas relacionadas con los ODS.	Interpretación errónea de los resultados y toma de decisiones incorrectas basadas en el modelo.
Entidades publicas	Usuario-cliente	Mayor eficacia en la planificación y ejecución de programas orientados a los ODS. Se invierte menos tiempo y dinero en la ejecución de proyectos e iniciativas orientadas a los objetivos del desarrollo sostenible.	Dependencia excesiva de los resultados del modelo y desatención de otros factores relevantes.
Empresas sin ánimo de lucro	Financiadores	Mayor transparencia en la asignación de recursos y evaluación de proyectos. Además, se presenta una mayor eficiencia en el uso de los recursos económicos proporcionados a la UNFPA.	Expectativas poco realistas sobre los resultados del modelo.

Comunidades locales	Beneficiarios	Mayor participación y empoderamiento de las comunidades en la toma de decisiones y el análisis de las necesidades locales de la población	Falta de comprensión de las comunidades sobre cómo se utilizan sus opiniones y datos recopilados.
---------------------	---------------	---	---

Tabla 2. Mapa de actores

5. Trabajo en equipo (8%)

Miembro	Roles	Tareas	Tiempos
Juan Sebastián Sánchez	Líder de proyecto Líder de datos	Entendimiento del negocio y enfoque analítico	1h
		Entendimiento y preparación de los datos	2h
		Entrenando Modelo con BoW y Random Forest classifier	1h 30min
		Descripción de trabajo en equipo	30 min
		Mapa de actores	30 min
Miguel Cárdenas Cárdenas	Líder de negocio	Entrenando el modelo con Gradient Boosting y BoW	1h 30 min
		Video de resultados	1h 30min
		Mapa de actores	20 min
Santiago Paeres Gonzales	Líder de analítica	Entrenando Modelo con TF-IDF y Random Forest classifier	2 horas 30min
		Prueba de búsqueda de un mejor modelo con RandomizedSearchCV	1 hora
		Realizar Predicciones	1 hora
		Análisis de resultados	1 hora

Tabla 3. Distribución de trabajo

En la Tabla 3 se muestran las tareas desarrolladas por cada integrante, los tiempos que se dedicó a cada una de ellas y los roles asignados a cada miembro del equipo. Durante el desarrollo del presente proyecto el equipo de trabajo tuvo que enfrentar algunos retos. A continuación, se describen de manera breve y concisa cada uno de ellos, así como también las respectivas medidas que fueron tomadas para resolverlos.

- **Codificación de los datos:** Al principio, por cuestiones de formato la codificación no era la correcta. Para solucionar esto, se especificó UTF-8 como tipo de codificación.
- **Caracteres incorrectos:** Existían caracteres que no pertenecen al idioma español, por lo que fueron remplazados por valores válidos.
- **Representación numérica de texto:** Para que los datos pudieran ser usados por los modelos, se transformaron utilizando BoW.
- **Complejidad temporal y espacial:** Para disminuir la complejidad, se optó por no considerar las palabras de parada, ya que estas no tienen un peso semántico realmente significativo

Con respecto a la distribución de los puntos, si bien todos los miembros del equipo colaboraron de manera activa en el desarrollo de esta primera entrega, consideramos que el método más justo para repartir los puntos consiste en tomar como base las tareas realizadas cada uno. En concreto, la idea sería hacer un balance general entre las tareas hechas junto con los tiempos de realización de cada una para distribuir los 100 puntos disponibles. También sería importante considerar el peso que tiene cada tarea dentro de la calificación al momento de hacer dicha ponderación.

6. Referencias

- Green Globe Sostenibilidad y Proyectos Ambientales. (2018, 5 septiembre). *Qué son los objetivos de Desarrollo Sostenible ODS y la agenda 2030*. <https://www.greenglobe.es/los-objetivos-desarrollo-sostenible-ods-la-agenda-2030/>
- Caribe, C. E. P. A. L. Y. E. (s. f.). *Objetivos de Desarrollo Sostenible (ODS)*. CEPAL. <https://www.cepal.org/es/temas/agenda-2030-desarrollo-sostenible/objetivos-desarrollo-sostenible-ods>

Scikit-learn. (S.F.). Text feature extraction. https://scikit-learn.org/stable/modules/feature_extraction.html#text-feature-extraction