# 1.Resumen

En la página Wikipedia se mantiene siempre en constante actualización describiendo todo tipo de personajes, actividades, eventos históricos y mucho más, cada una de estas entidades puede tener varias características que representan personajes, en este reporte se busca crear un modelo poder extraer múltiples características que presenten los nuevos samples de texto y para poder describir a las entidades, donde sus usos podrían estar en una mejora de búsquedas, mejora de la clasificación de artículos de Wikipedia. Se usara el Dataset: DBPedia el cual consta con el texto descripción de una entidad en ingles y 3 columnas de clasificación jerárquica donde cada una busca granular más la clasificación en nuestro caso por cuestión de recursos usaremos solos las 2 columnas: I1 con 9 clases siendo muy generales e I2 con 70 clases más específicas, eliminando I3 que tiene 219 clases, y contando con 342 781 observaciones, se combinan las columnas y podemos obtener un dataset multilabel, de este dataset se desarrollaran 2 modelos uno de regresión logística y un Random Forest, modelos clásicos que al ser un problema de clasificación se evaluara con el Accuracy, Precision ,Recall ,F1-score y también evaluaremos su Hamming Loss al ser un problema multilabel, a futuro se puede buscar mejorar el dataset con más samples, y generar predicciones reales para comprobar mejor su funcionamiento y áreas de mejora.

# 2. Método Propuesto

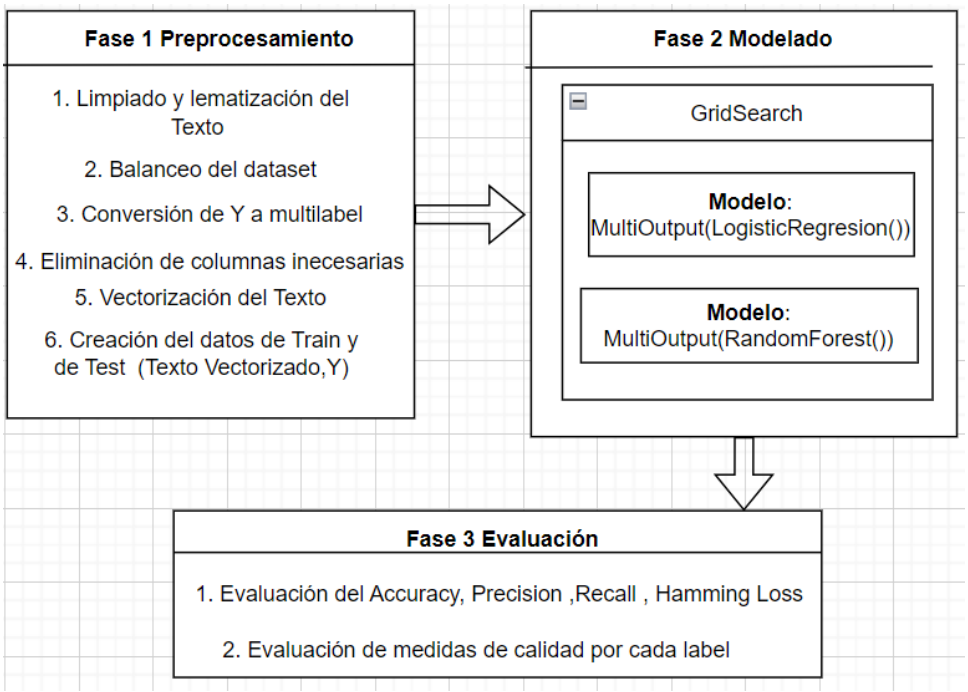


*Ilustración 1 Arquitectura del problema*

***Fase 1 Preprocesamiento***

1. **Limpiado y Lematización del texto:** Se eliminaron todos los símbolos, se aplico una lematización al texto, y se eliminaron las letras con acento y se eliminó todo carácter que no se pueda representar en ASCII decodificándolo a UTF-8.
2. **Balanceo del dataset:** Se realizo un Análisis de frecuencia y se decidió en realizar un muestreo del dataset para buscar balancearlo un poco en un máximo de 10 000 observaciones por clase, terminando con 268 699 observaciones.
3. **Conversión de Y a multilabel:** Se realiza un método para que hacer que nuestra variable de salida se convierta en una array binario .
4. **Eliminación de columnas:** Se eliminaron las columnas que no sean el texto
5. **Vectorización del texto**: Se transforma el texto en vectores para poder mandar los datos a los modelos limitándolos a 2500 features.
6. **Separación de datos de Train y de Test**: Se crearon lo datos de train y test de X con el texto vectorizado, y Y siendo el multilabel.

***Fase 2 Modelado***

***Se realizo un GridSearch para los modelos:***

1. **Modelo de regresión Logistica:** Se creo el modelo y se entreno con los datos de X_train y Y_train.
2. **Modelo de Random Forest:** Se creo el modelo y se entrenó con los datos de X_train y Y_train .

***Fase 3 Evaluación***

*Se obtuvieron las medidas de calidad incluido el haming Loss y para compararlas.*

# *3. Diseño de Experimentos*

**Tabla 1** Descripción General del dataset

| ***Dataset*** | ***Samples*** | ***Lenguaje*** | ***Clases*** |
|---|---|---|---|
| *DBpedia* | 342 781 | *Ingles* | 289 |
| DBpedia_Muestreado | 268 699 | *Ingles* | *79* |

**Tabla 2** Parámetros de optimización de los métodos

| **Modelo** | **Parámetros de Optimización** |
|---|---|
| **Regresión Logística (RL)** | Solver: liblinear, lbfgs<br>C: 01,1,10<br>Max iter: 100 |
| **Random Forest (RF)** | N estimators: 10,20,30<br><br>Max depth: None, 20 |

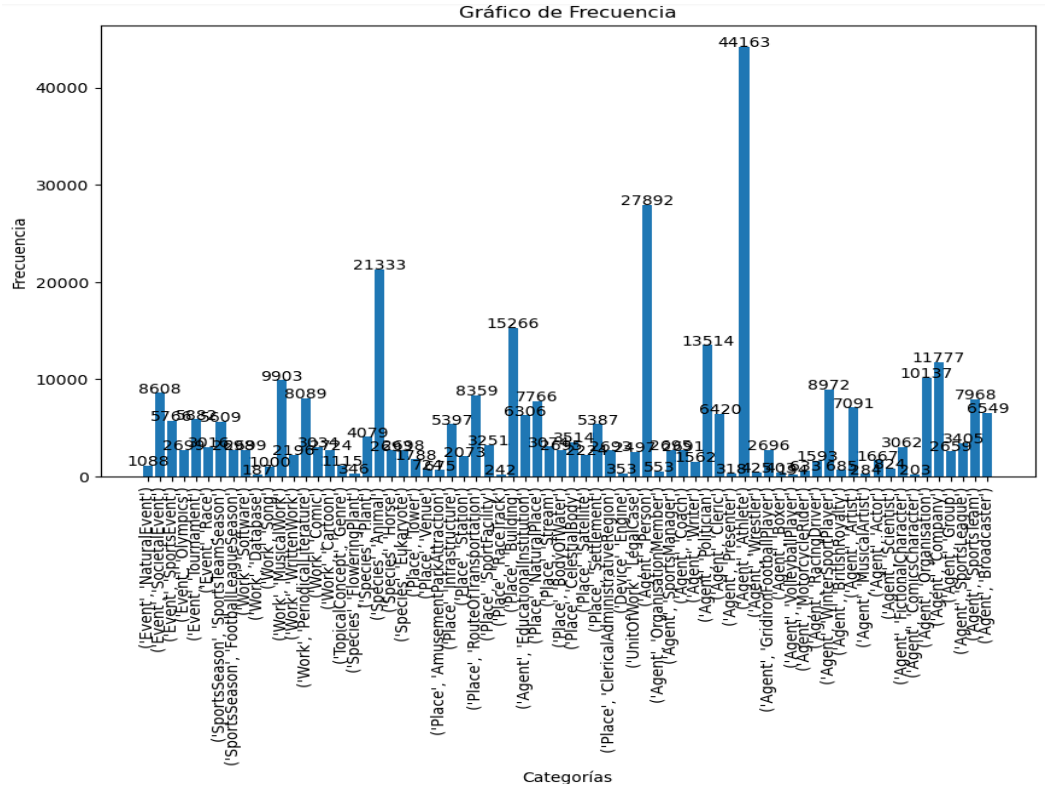


*Ilustración 2 Análisis de frecuencia antes del dataset Original*

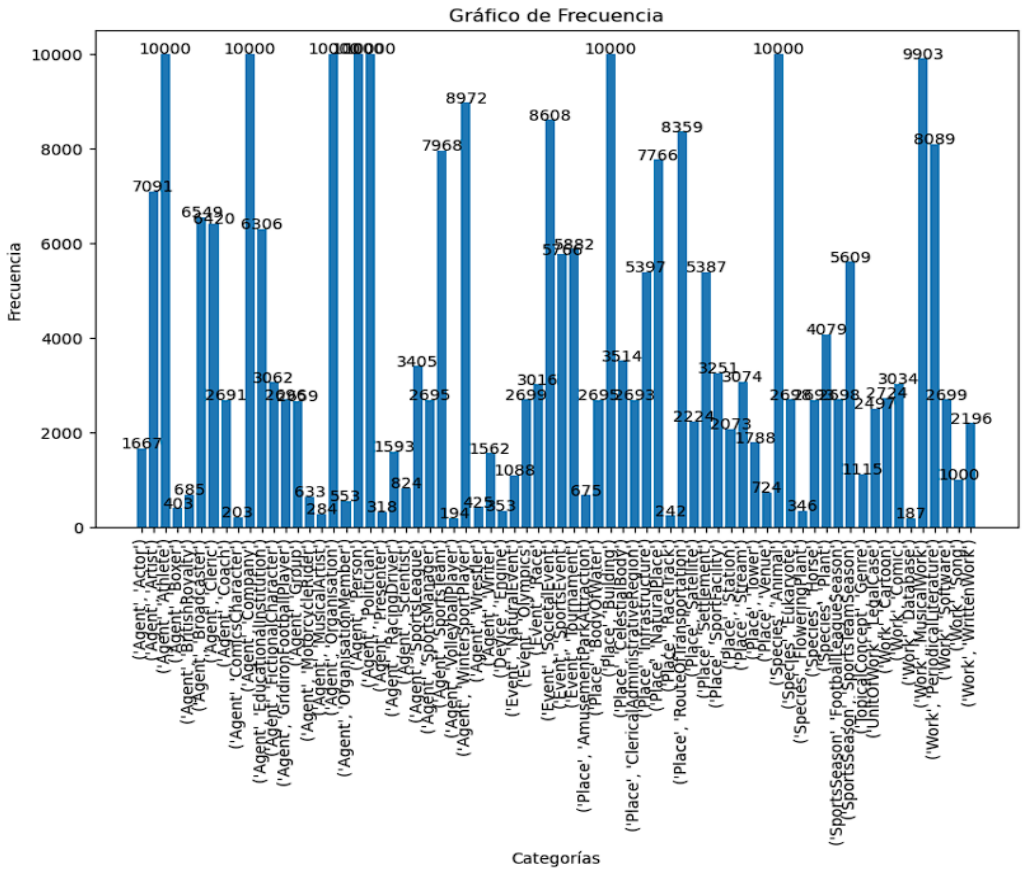


*Ilustración 3 Análisis de frecuencia del Dataset Muestreado limitado a 10 000 observaciones*

**Tabla 2** Parámetros de Compilación de los métodos

| **Modelo** | **Parámetros de Compilación** |
|---|---|
| **Regresión Logística (RL)** | Solver: liblinear<br>C: 10<br>Max iter: 100 |
| **Random Forest (RF)** | N estimators: 30<br><br>Max depth: None |

# ***4.Resultados y discusión***

Para nuestras medidas de calidad de clasificación se obtuvieron estos resultados

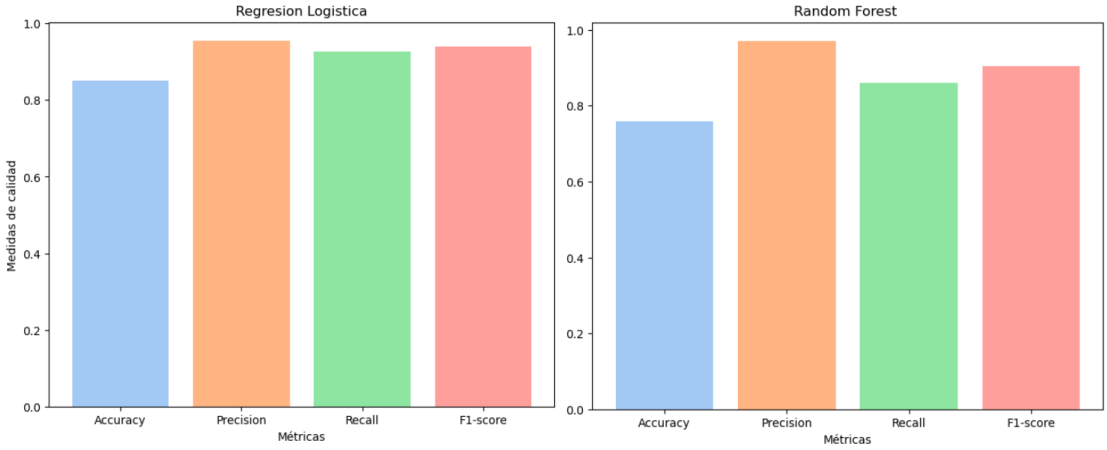


*Ilustración 4 grafica de Comparación de las medidas de calidad entre los 2 modelos*

Este fue nuestro resultado de Hamming Loss.

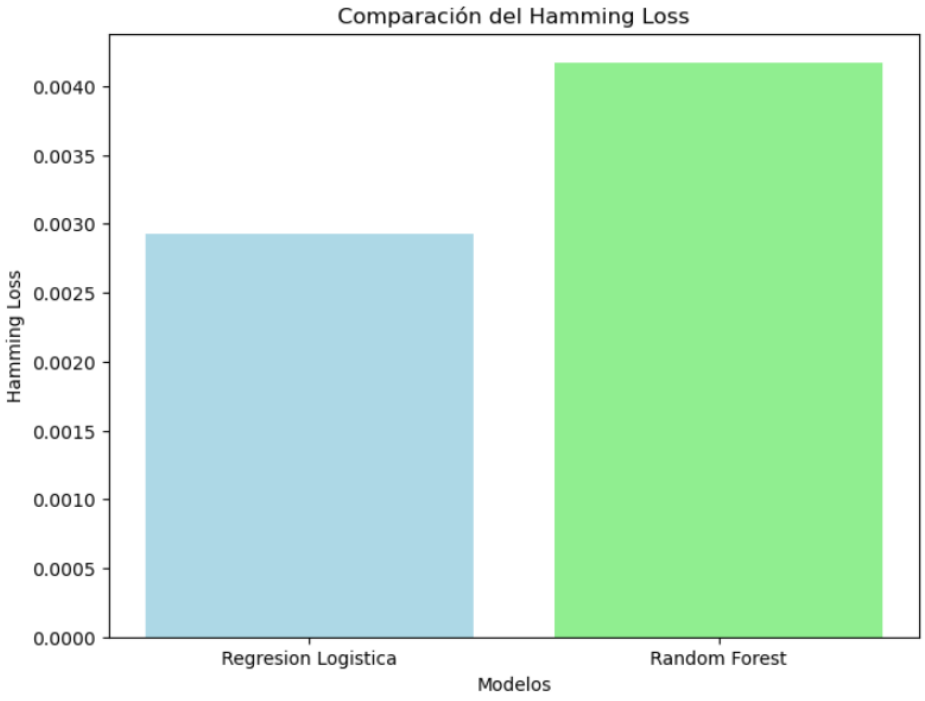


*Ilustración 5 Comparación del resultado de Hamming Loss*

*Tabla 3 Tabla de resultados*

| Metodo | Accuracy | Precision | Recall | F1-Score | Hamming Loss |
|---|---|---|---|---|---|
| **Regresión Logística** | 0.8489 | 0.9554 | 0.9265 | 0.9402 | 0.0029 |
| **Random Forest** | 0.7583 | 0.9713 | 0.8599 | 0.9043 | 0.004172 |

Esto nos indica que La Regresión Logística supera al Random Forest en términos de Accuracy, recall y F1-score.Sin embargo, el Random Forest tiene una precisión ligeramente mejor que la Regresión Logística. pero los 2 modelos tienen un Hamming Loss bastante bajo, lo que indica que ambos modelos son bastante buenos en la predicción multi etiquetas. de los 2 modelos el mejor termina siendo la regresión logística.

# *5. CONCLUSIONES*

En este trabajo se realizó la preparación de un dataset de datos de Wikipedia para su uso en tareas de clasificación de texto. El dataset original tenía 3 clases de salida, pero se redujo a 79 clases debido a la dificultad de predecir entre 298 clases con los recursos disponibles. Los datos se limpiaron eliminando caracteres especiales, stopwords y caracteres que no son ASCII. También se les aplicó lematización para mejorar la precisión de la clasificación. Se entrenaron dos modelos para la clasificación de texto: regresión logística y Random Forest. Se obtuvieron por parte del modelo de regresión logistica con un C:10 Y solver: liblinear: un Accuracy= 0.8494 Precision=0.9556 Recall=0.9276 F1-score=0.9409 y un Hamming loss=0.0029 y por parte del RF con n_estimators=30 y max_depth=None se obtuvo un Accuracy= 0.7583 Precision=0.9713 Recall=0.8599 F1-score=0.9043 y un Hamming loss=0.004172 .La regresión logística obtuvo mejores resultados en términos de accuracy , recall y F1-score y hamming Loss , mientras que el Random Forest obtuvo una precisión ligeramente mejor. En general, los resultados del experimento son prometedores. La regresión logística es un modelo eficaz y que no ocupa muchos recursos para la clasificación de texto, incluso en conjuntos de datos con un gran número de clases. Si se quisiera mejorar aún más los resultados, se podrían usar un dataset más grande, se puede una limpieza del texto distinta, o crear un método de Deep learning más complejo.

# *Referencias*


- Ortiz, I. R. H. (2023, diciembre 23). Clasificación de Sentimientos con Procesamiento de Lenguaje Natural y Redes Neuronales Recurrentes (LSTM).

Corpus en Español sobre restaurantes. GenSciNet. https://genscinet.com/clasificacion-sentimientos-nlp-lstm-espaniol/

- Dagan, G., Keller, F., & Lascarides, A. (2023). Dynamic Planning with a LLM. arXiv preprint arXiv:2308.06391.
- Ofer, D. (2019). DBPedia Classes [dataset]. Recuperado de: https://www.kaggle.com/datasets/danofer/dbpedia-classes
- Jiang, Q. (2023). Métodos de Clasificación con Python: Aplicaciones Empresariales.