Clasificación jerárquica de texto con machine learning en la industria petrolera

Sergio Antonio de Jesús Saldaña Pacheco, Fidelio Castillo Romero, Rosa Gómez Domínguez, Miguel Pérez Vasconcelos

Tecnológico Nacional de México/Instituto Tecnológico de Villahermosa, División de Estudios de Posgrado e Investigación, Carretera Villahermosa - Frontera Km. 3.5 Ciudad Industrial Villahermosa, Tabasco, México. C.P. 86010

Resumen

La clasificación jérarquica de texto es una tarea de aprendizaje automático que permite la categorización de grandes volúmenes de documentos mediante el uso de una estructura en forma de árbol. Los algoritmos de clasificación de texto en el campo de la ciencia de datos, posibilitan la construcción de modelos de aprendizaje automático que predicen la categoría a la que pertenece un documento dentro de una jerarquía de clases predefinidas. Las librerías gratuitas y de código abierto en el ámbito de la ciencia de datos que están disponibles en la actualidad alientan el desarrollo de tales proyectos, como se demuestra en este artículo al destacar la utilidad de estas herramientas a través de la creación de un modelo de aprendizaje automático supervisado con Python.

Abstract

Hierarchical text classification is a machine learning task that allows the categorization of large volumes of documents by using a tree-like structure. Text classification algorithms in the field of data science make it possible to build machine learning models that predict the category to which a document belongs within a predefined class hierarchy. The free and open source data science libraries that are currently available encourage the development of such projects, as demonstrated in this article by highlighting the usefulness of these tools through the creation of a supervised machine learning model with Python.

Palabras Clave: Clasificación jerárquica de texto, Aprendizaje supervisado, Clasificador multiclase, Clasificador local por nodo padre
Keywords: Hierarchical text classification, Supervised learning, Multi-class classifier, Local classifier per parent node

1. INTRODUCCIÓN

La gestión documental desempeña un papel fundamental en diversos ámbitos y organizaciones, abarcando tanto el sector público como el privado. Estos ámbitos incluyen, entre otros, la atención médica, las instituciones gubernamentales, el ámbito educativo y la industria. Específicamente, la industria petrolera es conocida por generar una gran cantidad de documentos que deben gestionarse de manera eficaz y conforme a las regulaciones que rigen sus operaciones.

Esta situación impulsa la búsqueda de soluciones tecnológicas efectivas que simplifiquen la gestión y recuperación de documentos. Una aproximación natural para organizar cualquier tipo de documento electrónico implica la creación de una estructura jerárquica de clases o taxonomía. Los documentos se agrupan en categorías, formando una estructura jerárquica en forma de árbol, lo que facilita a los usuarios encontrar información de manera más rápida y precisa [1,7].

En el ámbito de la inteligencia artificial y específicamente en el campo del machine learning, existen técnicas que pueden aplicarse a la gestión documental y la clasificación jerárquica de texto. A través del machine learning, es factible desarrollar un modelo de clasificación de documentos mediante el uso de algoritmos diseñados para la clasificación de texto, que predicen a qué categoría pertenece un documento [1]. Entre estos

algoritmos, se destacan: Máquinas de vector soporte (SVM - Support Vector Machines), Bayes Ingenuo (Naive Bayes), Árboles de Decisión, Bosques Aleatorios (Random Forests) y Redes Neuronales [8].

El objetivo de este artículo es presentar al lector el desarrollo y los resultados de un modelo de clasificación de texto mediante aprendizaje automático supervisado con Python, con el fin de ilustrar la utilidad y eficacia de la estructura jerárquica de texto como un medio para la clasificación de documentos en la industria petrolera.

2. APRENDIZAJE SUPERVISADO

El aprendizaje supervisado utiliza algoritmos que automatizan procesos de toma de decisiones a partir de un conjunto de datos en forma de pares de entradas y salidas deseadas. El usuario crea un conjunto de datos de ejemplos de pares de entradas y salidas deseadas para que el algoritmo aprenda (fase de entrenamiento) y sea capaz de resolver un problema de aprendizaje supervisado sin la ayuda de un humano. Este tipo de algoritmos que aprenden de pares de entrada/salida se denominan supervisados [2].

La clasificación de texto es un tipo de problema de aprendizaje supervisado, el objetivo es predecir una etiqueta de clase, que es una elección entre una lista predefinida de posibilidades. Generalmente, hay dos tipos de clasificaciones de texto dependiendo del número de clases involucradas en el problema: la clasificación binaria y multiclase. En la clasificación binaria el clasificador sólo puede manejar dos valores de clases, por ejemplo, para clasificar un correo si es spam o no. Un clasificador multiclase o de clases múltiples puede manejar más de dos clases, por ejemplo, para predecir un tipo de planta o animal entre varias categorías de clases. Los clasificadores multiclase también pueden ser de etiquetas múltiples, es decir, la respuesta del clasificador puede tener más de una clase asignada en un caso dado. En ambos casos, ya sea que el problema sea de clasificación binaria o multiclase, cada documento pertenece a una sola clase dentro del conjunto de clases predefinidas [2,1,3].

3. CLASIFICACIÓN JERÁRQUICA DE TEXTO

La clasificación jerárquica de texto es una tarea de aprendizaje automático que permite clasificar los documentos con una taxonomía predefinida mediante una estructura de árbol. Los nodos del árbol forman un conjunto finito de clases nombrados en función del dominio de la aplicación que se esté tratando [9,7,1]. La figura 1 muestra un ejemplo.

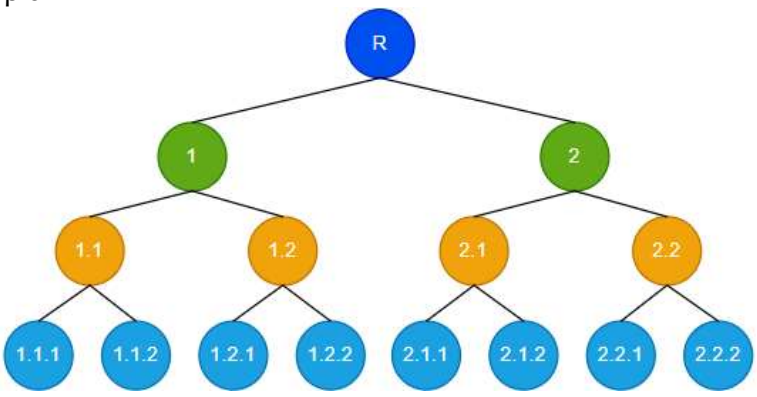


Figura 1. Ejemplo de estructura de árbol jerárquica
Diagrama basado en Silla y Freitas (2011)

El primer estudio de un problema de clasificación jerárquica de texto fue realizado por Koller y Sahami en 1997, donde se propuso el primer tipo de algoritmo o clasificador local junto con el principio de divide y vencerás conocido como enfoque de arriba hacia abajo (top-down). A partir de este trabajo, muchos autores han propuesto aproximaciones de clasificación jerárquica de texto [1,5].

4. DESARROLLO DEL MODELO

Para desarrollar el modelo de clasificación jerárquica de texto se aplicó el enfoque de aprendizaje automático supervisado, aprovechando el conocimiento previo de la taxonomía de los datos. La implementación del código se realizó en el lenguaje Python utilizando el entorno Jupyter Lab. Se emplearon dos librerías clave para Python: Scikit-learn y HiClass. Scikit-learn ofrece una variedad de clasificadores de texto supervisados [8], mientras que HiClass se especializa en la clasificación jerárquica local e incorpora implementaciones de los modelos de aprendizaje automático más populares, como el clasificador local por nodo, el clasificador local por nodo padre y el clasificador local por nivel. Además, esta librería proporciona métricas para evaluar el rendimiento del modelo en datos jerárquicos [6].

El modelo de aprendizaje automático elegido fue el clasificador local por nodo padre. Esta estrategia implica entrenar varios clasificadores de clases múltiples para cada nodo padre en la jerarquía de clases, con el objetivo de predecir los nodos hijos [4]. Esta estrategia se ilustra en la Figura 2.

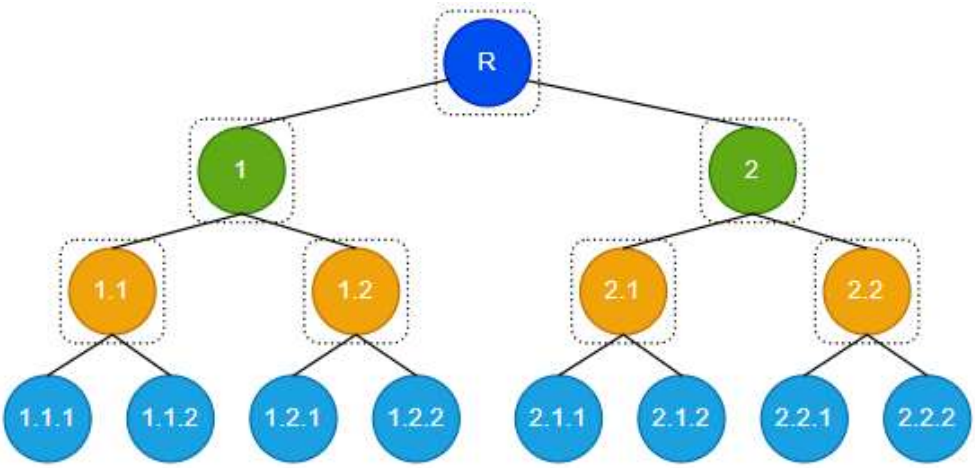


Figura 2. Clasificador local por nodo padre. (Los cuadros de líneas punteadas representan clasificadores de clases múltiples y los círculos representan clases)
Diagrama basado en Silla y Freitas (2011)

4.1 Definición de la taxonomía

El inicio del proceso de desarrollo del modelo involucra la definición de la taxonomía de los documentos. En este contexto particular, se trata de documentos específicos de la industria petrolera, como reportes, planos, manuales y estudios. Estos documentos se organizan en categorías para construir la estructura jerárquica de árbol, asignando una clase a cada nodo dentro de dicha jerarquía. Un ejemplo ilustrativo de esta estructura jerárquica de la industria petrolera se presenta en la Figura 3.

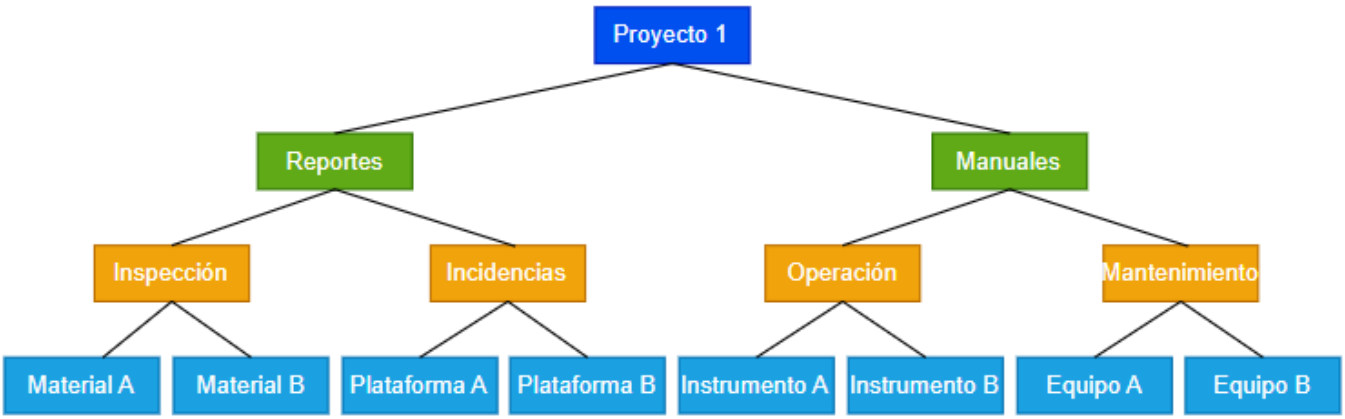


Figura 3. Ejemplo de taxonomía de la industria petrolera

Una vez definido el conjunto de taxonomías, se crea el dataset en formato csv, reflejando la estructura jerárquica previamente definida. El dataset debe incluir las descripciones de texto o características que describen el contenido de los documentos que se desean clasificar junto con la correspondiente clase jerárquica a la que pertenecen. En la siguiente sección, se podrá apreciar la estructura que guarda el dataset.

4.2 Implementación del código

Exploración de los datos

```
[1]: import pandas as pd
     # Carga el archivo de entrenamiento dentro de pandas dataframe
     data = pd.read_csv("taxonomia.csv").fillna(" ").sample(frac=1)
     #Resumen de la información del dataset
     data.info()
     #Dimensión del dataset
     print(f"Forma del dataset: {data.shape}")
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 181 entries, 99 to 169
Data columns (total 4 columns):
 #   Column       Non-Null Count  Dtype 
---  ------       --------------  ----- 
 0   texto        181 non-null    object
 1   categoria_1  181 non-null    object
 2   categoria_2  181 non-null    object
 3   categoria_3  181 non-null    object
dtypes: object(4)
memory usage: 7.1+ KB
Forma del dataset: (181, 4)
```

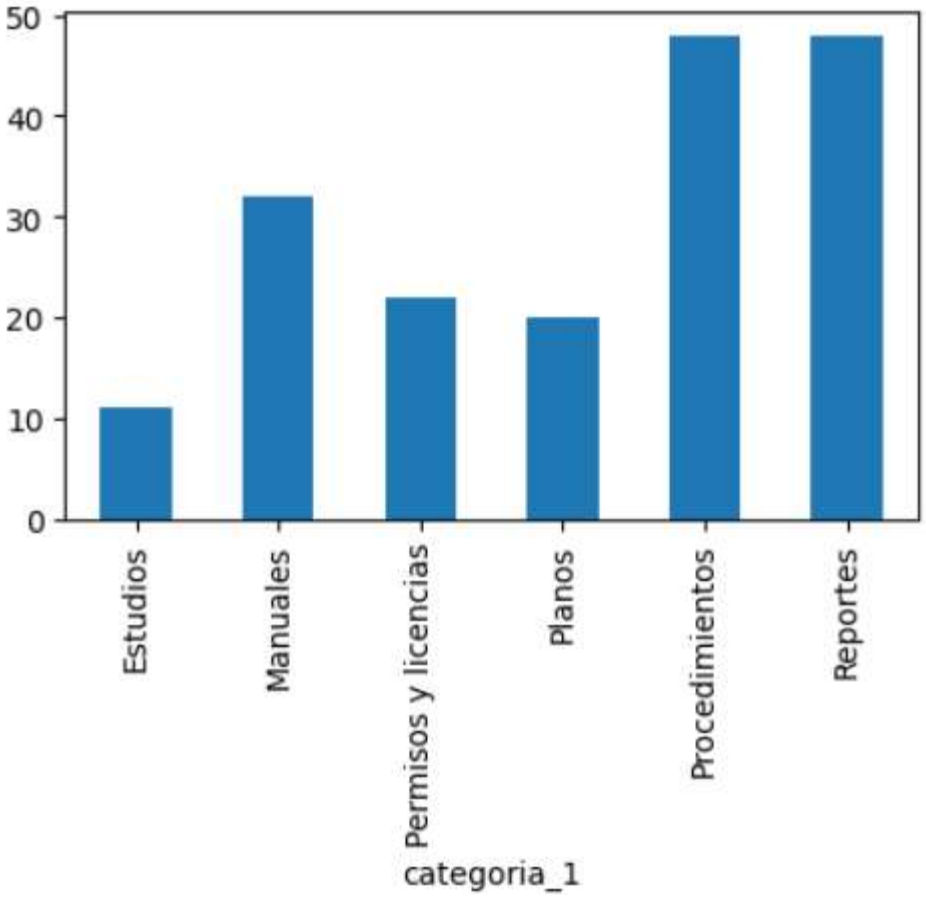
[2]:
```
#primeros cuatro ejemplos del dataset
data.head(4)
```

[2]:

	texto	categoria_1	categoria_2	categoria_3
99	Plano eléctrico instalación de puesta a tierra...	Planos	Planos as-built	Eléctrico
93	Plano de detalle constructivo de ducto ascende...	Planos	Planos as-built	Línea submarina
167	Bitácora de mantenimiento de equipos de combus...	Permisos y licencias	Reporte de cumplimiento ambiental	Término sexto
171	Análisis de riesgos de la embarcación osv remas	Estudios	Embarcaciones	Análisis de riesgos de las embarcaciones

[3]:
```
import matplotlib.pyplot as plt
#cantidad de ejemplos del dataset agrupados por categoría nivel 1
print(data['categoria_1'].value_counts())
plt.figure(figsize=(5,3))
data.groupby('categoria_1').texto.count().plot.bar()
plt.show()
```

```
categoria_1
Procedimientos          48
Reportes                48
Manuales                32
Permisos y licencias    22
Planos                  20
Estudios                11
Name: count, dtype: int64
```



Preparación de los datos

```
[4]: from sklearn.model_selection import train_test_split
     #divisón del dataset: 80% datos de entrenamiento, 20% datos de prueba
     caracteristicas =  data["texto"]
     etiquetas = data[["categoria_1", "categoria_2", "categoria_3"]]
     X_train, X_test, y_train, y_test = train_test_split(caracteristicas,
                                                         etiquetas,
                                                         test_size=0.20,
                                                         random_state=0)
     print(f"Forma datos de entrenamiento: {X_train.shape} {y_train.shape}")
     print(f"Forma datos de prueba: {X_test.shape} {y_test.shape}")
```

```
Forma datos de entrenamiento: (144,) (144, 3)
Forma datos de prueba: (37,) (37, 3)
```

Determinar la cantidad de núcleos del procesador

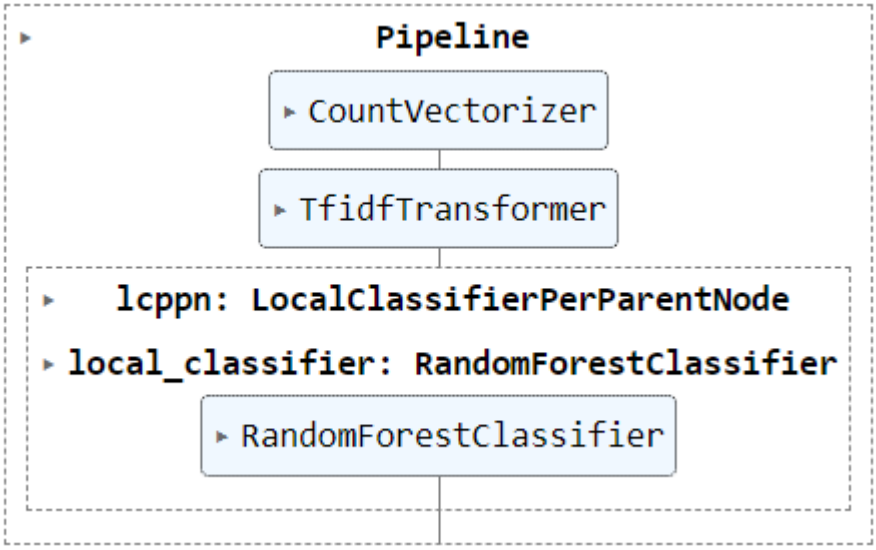
```
[5]: from os import cpu_count
     print(f"Número de núcleos del procesador {cpu_count()}")

     Número de núcleos del procesador 4
```

Entrenamiento del clasificador

```
[6]: from sklearn.feature_extraction.text import CountVectorizer, TfidfTransformer
     from sklearn.ensemble import RandomForestClassifier
     from sklearn.pipeline import Pipeline
     from hiclass import LocalClassifierPerParentNode
     #clasificador Random Forest
     lr = RandomForestClassifier()
     pipeline = Pipeline(
      [
        ("count", CountVectorizer()),
        ("tfidf", TfidfTransformer()),
          (
            #estrategia clasificador local por nodo padre
            "lcppn",
            LocalClassifierPerParentNode(local_classifier=lr, n_jobs=cpu_count()),
          ),
      ]
     )
     pipeline.fit(X_train, y_train)
```

[6]:



Evaluación del clasificador

[7]:
```
# Prueba de predicción del modelo
from hiclass.metrics import f1
print(f"Primeras 5 líneas de prueba:\n {X_test[:5]}\n")
predictions = pipeline.predict(X_test)
print(f"Resultado de las 5 primeras predicciones:\n {predictions[:5]}\n")
print(f"Score f1:{f1(y_true=y_test, y_pred=predictions)}")
```

```
Primeras 5 líneas de prueba:
 100    Certificado de Válvula de bola montada sobre m...
24     Resultados de la composición química emitido p...
98     Plano eléctrico de conexión de puesta a tierra...
42     Procedimiento para fabricación e instalación d...
116    Certificado de Válvula de bola en acero al car...
Name: texto, dtype: object

Resultado de las 5 primeras predicciones:
 [['Reportes' 'Certificados de los equipos e instrumentos' 'Válvulas']
 ['Manuales' 'Dossier de ánodos de sacrificio en aluminio'
  'Resultados de inspección y prueba']
 ['Planos' 'Planos as-built' 'Eléctrico']
 ['Procedimientos' 'Planes y procedimientos'
  'Procedimiento de construcción']
 ['Reportes' 'Certificados de los equipos e instrumentos' 'Válvulas']]

Score f1:0.918918918918919
```

Predicción con datos nuevos

[8]:
```
X_test= [
        "Procedimiento para registro de indicadores",
        "Plano del sistema eléctrico",
        "Estudio sobre factibilidad técnica",
       ]
predictions = pipeline.predict(X_test)
print(f"Resultado de la prediccion:\n {predictions}\n")
```

```
Resultado de la prediccion:
 [['Procedimientos' 'Planes y procedimientos'
  'Plan de seguridad industrial']
 ['Planos' 'Planos as-built' 'Linea submarina']
 ['Estudios' 'Estudio geofisico' 'Documentos']]
```

5. RESULTADOS

Las pruebas realizadas en el modelo de aprendizaje automático supervisado se usaron para resolver un problema de clasificación jerárquica de texto conforme a una taxonomía de documentos del sector petrolero.

Como se muestra en la sección anterior, se pueden observar los resultados del algoritmo que predice la categoría a la que pertenecen los textos de entrada del contenido de cada documento.

Para el entrenamiento del modelo supervisado de clasificación jerárquica de texto, se empleó un dataset que consta de 181 registros con información relacionada a la industria petrolera. Este dataset incluye 6 clases en el nivel 1, 14 clases en el nivel 2 y 43 clases en el nivel 3.

En la Tabla 1 se presenta un comparativo de la métrica de evaluación de diferentes clasificadores jerárquicos usados en el modelo de aprendizaje automático destacando Random Forest con el conjunto de datos de prueba que representa el 20% del dataset.

CLASIFICADOR	PUNTUACIÓN F1
SVC	0.8558558558558559
Random Forest	0.918918918918919
MLPClassifier	0.8648648648648649
LogisticRegresion	0.7297297297297297
MultinomialNB	0.8587570621468926
DecisionTree	0.8468468468468469

Tabla 1. Comparativo de puntuación F1

Todas las pruebas del modelo de aprendizaje supervisado se realizaron en una computadora con procesador de cuatro núcleos y 8 Gb RAM con sistema operativo propietario de 64 bits.

6. DISCUSIÓN Y CONCLUSIONES

El tema central de esta investigación fue la clasificación jerárquica de texto como un medio para clasificar los documentos en la industria petrolera Para construir el dataset, se empleó un ejemplo de taxonomía de documentos específicos de este sector. Finalmente, a través del uso de un modelo de machine learning supervisado, se lograron métricas de evaluación favorables, sugiriendo la posibilidad de refinar el modelo para alcanzar el objetivo deseado.

En conclusión, la clasificación jerárquica de texto mediante machine learning sienta las bases para abordar problemas de clasificación documental en diversos sectores, como la industria, instituciones educativas o entidades gubernamentales. Además, el acceso a herramientas gratuitas de ciencia de datos en Python simplifica y alienta el desarrollo de proyectos tecnológicos e investigaciones que requieran la aplicación de la inteligencia artificial.

REFERENCIAS

[1] Carlos N. Silla Jr. & Alex A. Freitas (2011). A survey of hierarchical classification across different application domains. Springer.
[2] Andreas C. Müller & Sarah Guido (2017). Introduction to Machine Learning with Python. A Guide for Data Scientists. O'Reilly.
[3] Sowmya Vajjala, Bodhisattwa Majumder, Anuj Gupta & Harshit Surana (2020) Practical Natural Language Processing. A Comprehensive Guide to Building Real-World NLP SystemsO'Reilly.
[4] Fábio M. Miranda, Niklas Köhnecke, Bernhard Y. Renard (2023). HiClass: a Python Library for Local Hierarchical Classification Compatible with Scikit-learn.
[5] Svetlana Kiritchenko, Stan Matwin, Richard Nock and A. Fazel Famili (2006). Learning and Evaluation in the Presence of ClassHierarchies: Application to Text Categorization.
[6] Fabio M. Miranda, Niklas Köhnecke (2022). Algorithms Overview. Obtenido de https://hiclass.readthedocs.io/en/latest/algorithms/index.html el 13 Octubre de 2023.
[7] Aixin Sun, Ee-Peng Lim & Wee-Keong Ng (2003). Hierarchical text classification methods and their specification. School of Computer Engineering, Nanyang Technological University, Singapore.
[8] Scikit-learn developers (2023). Supervised learning. Obtenido de https://scikit-learn.org/stable/supervised_learning.html el 18 Octubre de 2023.
[9] Daniel Andrés Silva Palacios (2021). Clasificación Jerárquica Multiclase.Tesis Doctoral. Universitat Politècnica de València.

Correo de autor de correspondencia: ***sergio_jsp@hotmail.com***