Trabajo Final – Fundamentos de la Ingeniería del Software para Ciencia de Datos

Autores: Luis Eduardo Mamani Chambi, Álvaro Andrés Montesinos Miranda y Alfonso Mauricio López Zabalaga

***Resumen*—El presente documento es resultado del trabajo final realizado en el módulo "Fundamentos de la Ingeniería del Software para Ciencia de Datos" dictado por el Msc. Edwin Salcedo, como parte del plan de estudio de la Maestría de Ciencia de Datos y Big Data**

El trabajo implicó el desarrollo de un módulo para la gestión de imágenes y metadatos para un proyecto de investigación de imagenología médica en base al DataSet "COVID-19 Radiography Database" accesible en el sitio de Kaggle (www.kaggle.com).

Se desarrollaron paquetes y módulos en Python que contienen funciones/métodos que permiten recuperar y transformar sus metadatos a dataframes, de esta manera poder complementar metadatos, y así analizar, modificar o eliminar los datos obtenidos del DataSet.

***Abstract*—This document is the result of the final work carried out in the module "Fundamentals of Software Engineering for Data Science" dictated by the Msc. Edwin Salcedo, as part of the study plan for the Master of Data Science and Big Data**

The work involved the development of a module for managing images and metadata for a medical imaging research project based on the DataSet "COVID-19 Radiography Database" accessible on the Kaggle site (www.kaggle.com).

Python packages and modules were developed that contain functions/methods that allow retrieving and transforming their metadata to dataframes, thus being able to complement metadata, and thus analyze, modify or delete the data obtained from the DataSet.

I. Introducción

El equipo de trabajo obtuvo el DataSet publicado por un grupo de investigación de la Universidad de Qatar, que crearon una base de datos de imágenes de Rayos-X para casos positivos de COVID-19 (3.616 imágenes), de Neumonía viral (1.342 imágenes), de Infecciones pulmonares (6.012 imágenes) y también de casos Normales (10.192 imágenes), las mismas distribuidas en carpetas separadas por cada categoría. Además de cuatro archivos en formato .xlsx con los metadatos de cada categoría de imágenes anteriormente mencionadas.

Planteamiento del problema:

Se plantea como problemas a abordar por el proyecto la falta de información de la ubicación, ancho y alto de las imágenes en los metadatos que hacen referencia a las imágenes de Rayos-X obtenidas, como también la limitación no poder corregir los nombres de las imágenes o su eliminación, acciones que son necesarias para el mantenimiento adecuado de los metadatos y las imágenes asociadas.

Objetivos del proyecto:

El proyecto tiene como objetivo desarrollar una solución de software aplicando buenas prácticas en la codificación, en la estrategia de modularización, como en la programación orientada a objetos, que permita registro, modificación y eliminación de imágenes y sus metadatos a través de funciones/métodos desarrollados.

II. Estrategias de Modularización y Programación Orientada a Objetos

A. *Clases*

Para el desarrollo del presente proyecto se establecieron como necesarios los siguientes objetos:

Clase Imagen: Este objeto contiene los atributos propios para una imagen, con seis métodos/funciones que permiten trabajar sobre sus atributos estos métodos son para: adicionar la ubicación de la imagen a sus metadatos, adicionar las dimensiones de ancho y altura tanto de la imagen como de la máscara, visualizar imagen por categoría, eliminar imágenes, como modificar su nombre.

Clase Category: Este objeto tiene como fin permitir la gestión de algunos atributos, el mismo tiene dos métodos que son para: realizar un conteo de las imágenes que tiene el dataset y para visualizar una imagen por categoria.

Clase Category_aditional_function: Este objeto tiene como fin permitir el manejo de los análisis sobre los metadatos, el mismo tiene cuatro métodos que son para: determinar la categoría de imágenes que más se repite u la que menos se repite, graficar la cantidad de imágenes por categoría y visualizar todas las imágenes del dataset.

A continuación, se visualiza el diagrama de clases:

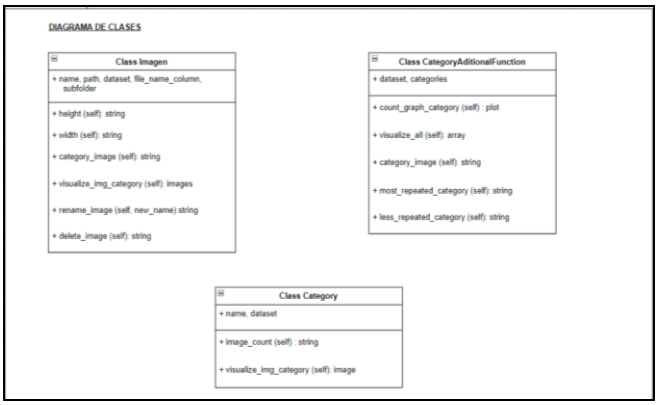

Fig. 1. Diagrama de Clases

B. Módulos y Paquetes

Para el desarrollo del presente proyecto se establecieron los siguientes paquetes y módulos:

Paquete Proyecto Final: Este es el paquete principal del proyecto ya que engloba el módulo main.py desde donde se importará cada paquete y se llamará a los módulos y métodos para cumplir con el objetivo del trabajo. Este incluye los paquetes "data_analysis", "data_processing" y "tests".

Paquete data_analysis: Este paquete incluye cuatro módulos con métodos: "imagen" el cual tiene funciones para obtener y complementar el ancho y alto de las imágenes; categorizar las imágenes, modificar el nombre de las imágenes existentes, eliminar alguna imagen y su metadato, visualizar imágenes por categoría; "category" que tiene la función para calcular la cantidad de imágenes del dataset, como visualizar las imágenes por categoría; "category_aditional_function" que contiene funciones para visualizar todas las imágenes, para generar una gráfica de la cantidad de imágenes por categoría, para determinar las categorías de imágenes más y menos repetidas en el dataset y por último "__init__" que referencia a los módulos que forman parte del paquete.

Paquete data_processing: Este paquete incluye siete módulos con métodos: "encode_dataframe" el cual tiene la función para codificar unívocamente cada registro de imágenes; "image_dimmension" que tiene las funciones para obtener y complementar el ancho y alto de las imágenes y sus máscaras; "image_path" contiene la función para obtener la ruta exacta de cada imagen y actualiza la ubicación como un nuevo metadato, como también las imágenes por categoría; "create_file" que tiene la función que permite clasificar las imágenes en carpetas por categoría; "set_to_dataframe" contiene la función que permite recopilar los archivos ".csv" de cada categoría de imágenes y fusionarlos en un solo dataframe para ser gestionado o analizado; "xlsx_to_csv" el cual tiene la función para convertir archivos de formato Excel (.xlsx) a formato .csv y por último "__init__" que referencia a los módulos que forman parte del paquete.

Paquete tests: Este paquete incluye siete módulos con métodos: "test_image_path" el cual tiene la función que realiza las pruebas unitarias para verificar la ubicación de las imágenes junto a su metadato; "test_image_dimmension" que contiene la función para verificar que el alto y ancho de la imagen es correcta con su metadato; "test_encode_dataframe" que tiene la función para verificar que las imágenes estén categorizadas; "test_imagen" que tiene la función para verificar la exactitud del ancho, algo y categoría de las imágenes COVID; "test_category" que contiene la función para verificar la integridad de la cantidad de imágenes en cada categoría; "test_category_aditional_function" que contiene la función para verificar las categorías más y menos repetidas; y por último "__init__" que referencia al módulo que forma parte del paquete.

A continuación, se visualiza el Diagrama de Paquetes y Módulos:

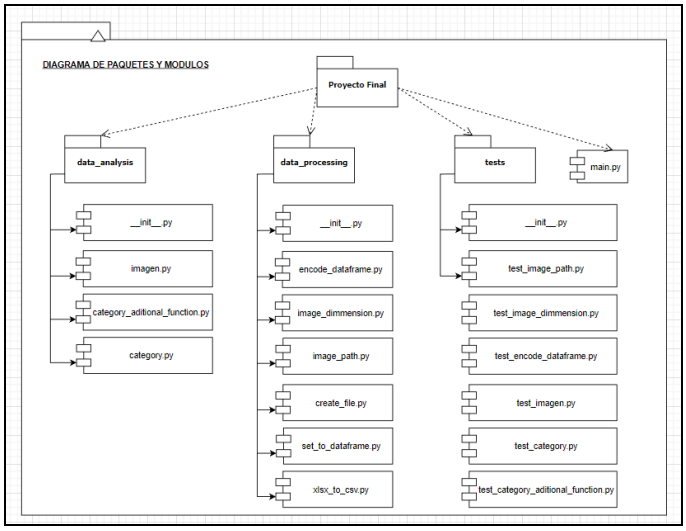

Fig. 2. Diagrama de Paquetes y Módulos

Documentación:

Al final el desarrollo del proyecto, con todas las clases y funciones debidamente comentadas (docstrings) se procedió a generar la documentación con la ayuda de la librería sphinx, dando como resultado la carpeta docs que contiene un archivo index.html donde se muestra la documentación a detalle.

III. VERSIONAMIENTO DE CÓDIGO Y DATOS.

El versionamiento de código es fundamental para tener control sobre las características del mismo, sin embargo, esta idea se puede extender al manejo del set de datos lo cual resulta beneficioso especialmente en el área de la ciencia de datos. Este concepto no es menor ya que se puede obtener mediante herramientas estadísticas el rendimiento según la variación o modificación de este, por lo que en el campo del análisis de datos se vuelve una tarea fundamental tener un control de las versiones y modificaciones que se realice sobre este.

Consecuentemente, en el presente proyecto implementa tanto el versiona miento de código como del dataset. A continuación, se menciona brevemente los puntos más relevantes en este flujo de trabajo, donde se hace una descripción del contexto en el cual se usó un determinado comando:

- Enlace de GitHub con el entorno local. Para ello se tuvo que crear una llave ssh y compartir ésta en la cuenta del repositorio para sincronizar los archivos sobre los cuales se desea hacer “push”.
  cmd: shh-keygen
- Inicializamos el repositorio local para el control de versiones con git:
  git-bash: git init.
- Añadimos todos los archivos que se encuentren en la carpeta local con:
  git-bash: git add .
- Inicializamos el repositorio local para el control de versión con git:
  git-bash: git init.
- Ya que se va a trabajar con un dataset de imágenes muy grande, es necesario ignorar la carpeta donde se encuentren esto archivos ya que estos se sincronizarán después con una carpeta en GDrive:
  git-bash: echo “dataset” > .gitingnore
- Creamos una nueva rama que en este caso será el “feature” de la primera historia de usuario US-1-programa-funcional:
  git-bash: git checkout -b US-1-programa-funcional
- Creamos el primer punto de referencia para el versionamiento del código:
  git-Bash: git commit -m “primera funcionalidad implementada”
- Sincronizamos la carpeta local que tiene la implementación del código, con el repositorio:
  git-bash: git push origin US-1-programa-funcional

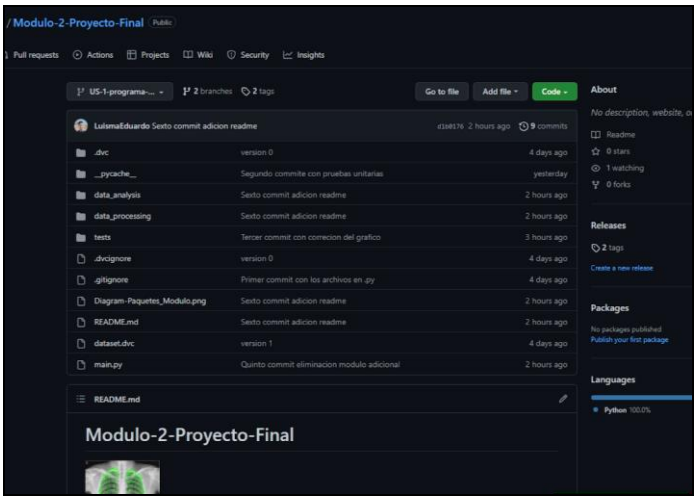

Fig. 3. Github

En este primer flujo se termina se muestra el versionamiento solamente del código de la aplicación (URL Github: https://github.com/AlvaroMontesinos/Modulo-2-Proyecto-Final/tree/US-1-programa-fucional).

Previo al versionamiento del dataset nos creamos una carpeta en el GDrive para que este se sincronice con los archivos de la carpeta data set correspondiente (URL dvc: https://drive.google.com/drive/folders/1e6G8mhFr8G7hDrO5H1LTtltlHjx9yz_h?usp=sharing). A continuación, se muestra el flujo de trabajo con dvc:

- Inicialmente instalamos la librería en el entorno local con:
  cmd: pip install dvc
- También es necesario agregar la dependencia para el manejo de las Apis de GDrive:
  cmd: pip install pydirve2
- Inicializamos el repositorio local para el manejo del dataset con DVC:
  git-bash: dvc init
- Añadimos la carpeta dataset al workspace de DVC:
  dvc add dataset
- Ingresamos la url con el id de la carpeta creada previamente en GDrive como origen del repositorio:
  Git-bash: dv remote add –default myremote gdrive://carpetaDrive
- Hacemos la sincronización de los archivos locales con el repositorio:
  Git-bash: dvc push
- Es necesario tener un versiona miento sobre el archivo dvc para ello recurrimos nuevamente a git añadiendo todo el contenido de la carpeta:
  git-bash: git add.
- Añadiremos un commit con la versión del código y del dataset referenciado con dvc:
  git-bash: git commit -m “version 0”
- Hacemos referencia al estado del repositorio completo mediante el uso de un tag:
  git-bash: git tag -a v0 -m "version 0"
- Sincronizamos la carpeta local con el repositorio incluyendo el archivo con la extensión .dvc con la información del estado de dataset:
  git-bash: git push origin master –tags

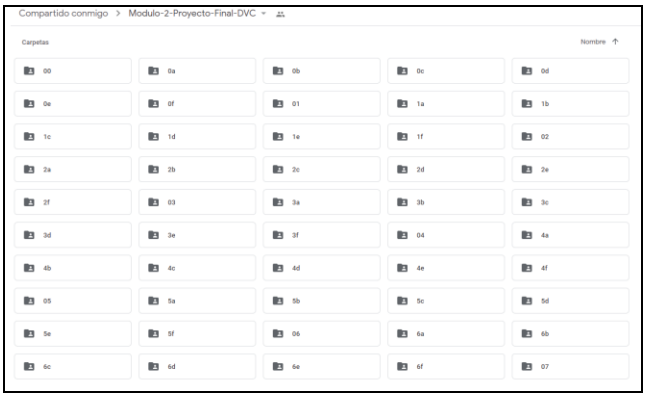

Fig. 4. Control DVC

IV. INTERFACES DE INTERACCIÓN CON USUARIO

Para la interacción del usuario se ha considerado la interface a través de la consola de comandos (terminal), generando menús enumerados que le permitan al usuario interactuar con las opciones disponibles del proyecto; a continuación, se muestran algunas capturas de la interacción de menús:

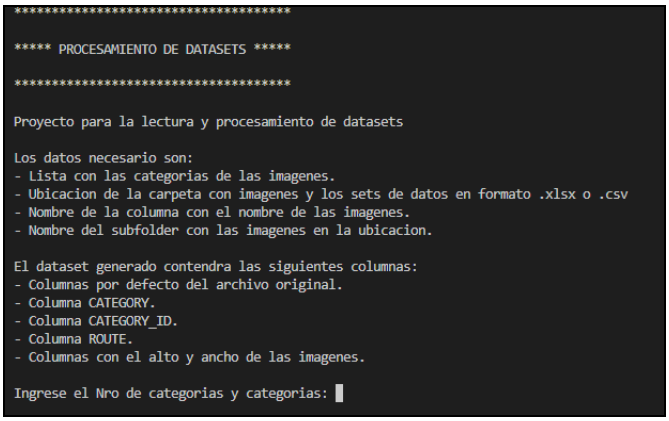

Fig. 5. Menú Principal

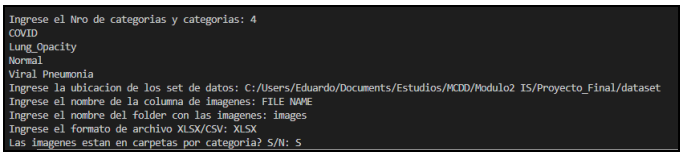

Fig. 6. Categorización de imágenes

Fig. 7. Dataset procesado

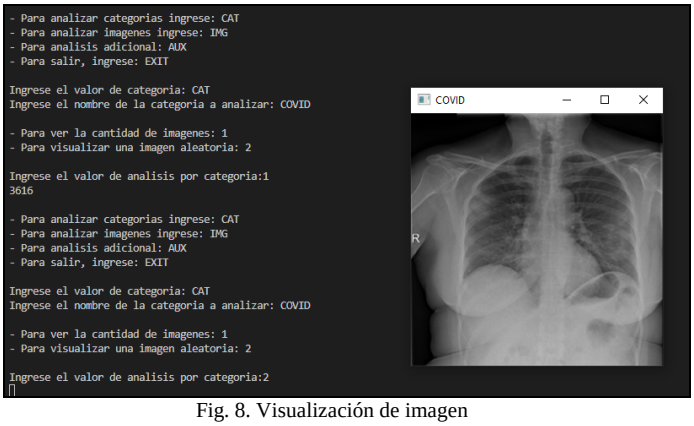

Fig. 8. Visualización de imagen

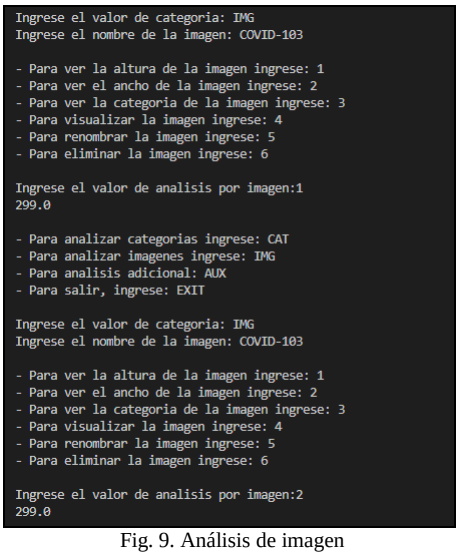

Fig. 9. Análisis de imagen

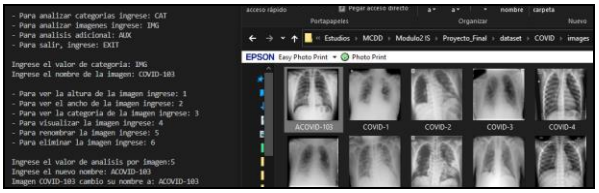

Fig. 10. Visualización de imágenes

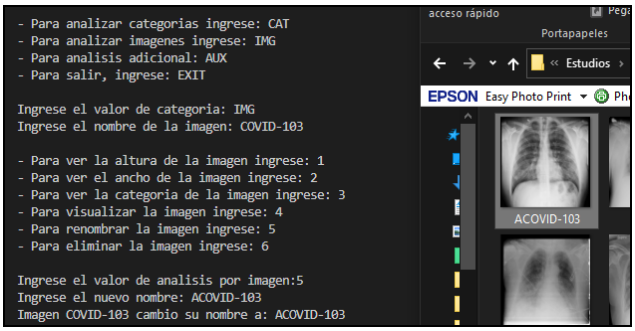

Fig. 11. Cambio de nombre

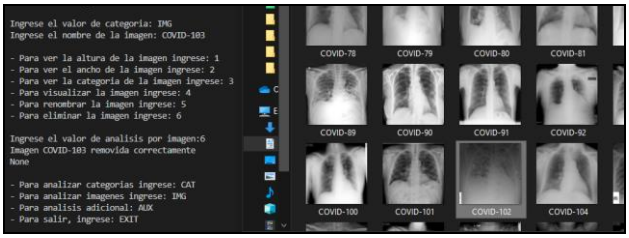

Fig. 12. Eliminación de imagen

Fig. 13. Graficación de datos

V. ESTRATEGIAS Y MÉTODOS DE PRUEBAS

A. *Estrategias de pruebas*

Para el desarrollo de las pruebas del presente proyecto se definió la aplicación de pruebas unitarias mediante el marco de pruebas automatizadas y se utilizó la infraestructura y conjunto de herramientas para la construcción y ejecución de pruebas ‘unittest’, es así que el paquete “tests” contiene todos los módulos con sus funciones que realizan los diferentes casos de pruebas y verificaciones a los metadatos asociados al dataset de imágenes. Las pruebas se enfocan a verificar los resultados de las funciones implementadas en los módulos: “data_analysis” y “data_processing”.

B. *Métodos de pruebas*

Los métodos de pruebas aplicados son de comparación

(igualdad) con valores esperados y también para comprobar la consistencia de los metadatos y los archivos de imágenes del dataset.

El detalle de los seis módulos y sus funciones del paquete “tests” se detallan a continuación:

test_imagen: Para la evaluación de las funciones (height, width, image_category) del módulo imagen del paquete data_analysis.

test_category: Para la evaluación de la función equal_count) con todas sus variaciones del módulo category del paquete data_analysis.

test_category_aditional_function: Para la evaluación de las funciones (most_repeated, less_repeated) del módulo category_aditional_function del paquete data_analysis.

test_encode_dataframe: Para la evaluación de la función (encode_dataframe) del módulo encode_dataframe del paquete data_processing.

test_image_dimmension: Para la evaluación de las funciones (image_height, image_width) del módulo image_dimmension del paquete data_processing.

test_image_path: Para la evaluación de la función (image_path) del módulo imagen_path del paquete data_processing.

VI. CONCLUSIONES

Consecuentemente, se desarrolló una estructura del proyecto basado en el paradigma de programación modular y cuyos módulos a la vez implementan también el paradigma de programación orientada a objetos. Asimismo, se fundamentó el estilo de programación con el uso del estándar PEP8 siguiendo patrón para un código más mantenible incluyendo dentro de este estándar la documentación del código mediante docstring.

En la misma línea, la implementación de control de versiones tanto para el dataset como para todo el software en Python posibilitaron además de mantenible sea un proyecto escalable para futuras implementaciones, acorde con los procesos de buenas prácticas para la producción de software.

El software producido responde también a estándares de calidad ya que sus funcionalidades fueron evaluadas mediante pruebas unitarias pasando todas ellas, con lo que se evidencia que la implementación como la estructuración del mismo fue correctamente diseñadas.

Finalmente se puede concluir que se satisficieron todos los requerimientos incluyendo las funcionalidades de registro, modificación y eliminación de metadatos y sus correspondientes imágenes, dentro de estándares de producción de software basados en múltiples paradigmas, estándares de calidad y su documentación correspondiente.

Como trabajos futuros, se recomienda implementar un modelo de “machine learning” para extender las funcionalidades del programa y que el mismo se capaz de clasificar y automatizar el proceso medico de detención de infecciones Covid basado en las imágenes radiográficas.
Esta funcionalidad seria de mucha ayuda dentro el contexto actual en términos de medio y corto plazo. Por otra parte, se ve necesario implementar una GUI para que el programa sea más usable y amigable con el usuario final, para futuras versiones.

REFERENCIAS

[1] Tawsifur Rahman, Dr. Muhammad Chowdhury, Amith Khandakar, “COVID-19 Radiography Database”, *KAAGLE*, “https://www.kaggle.com/datasets/tawsifurrahman/covid19-radiography-database”.

[2] Msc. Edwin Salcedo, “Procesamiento de Archivos con Python”, Plataforma Colab, ”https://colab.research.google.com/drive/1RaOl7dERPBQrmu52t5hkQwn0S5NwjFQx?usp=sharing”.