**INFORME DE PRUEBAS**

**INTRODUCCIÓN:**

Para el plan de pruebas hemos decidido enfocarnos en analizar la estructura de archivos, específicamente aquella que involucra el proceso de ETL y de credenciales de AWS. Consideramos que no tienen mucho sentido realizarlo al proceso de entrenamiento y preprocesado del entrenamiento, ya que en una situación real estes procesos se realizan de forma muy puntual y no son susceptibles a cambios frecuentes, al contrario que las ingestas de datos.

Hemos dividido el plan en los siguientes apartados:

* Datos: Se verifica la existencia de los archivos de ingesta y si se lanzan las excepciones correctas cuando estes faltan.
* Estructura: Verificamos que los archivos a procesar tienen una estructura coherente y común entre ellos. También se comprueba que la estructura de directorios para los ficheros está presente.
* Módulos: Se comprueba que los Módulos están en sus correspondientes carpetas y se pueden llamar correctamente.
* Credenciales: Comprobamos la estructura y existencia de las credenciales mínimas necesarias para lanzar la ingesta en AWS.

Estas pruebas se realizan en varias clases distintas. Algunas están distribuidas entre varios tests, y otras unificadas en uno solo. Por ejemplo, la existencia de la estructura necesaria para la ingesta y la de la jerarquía de paquetes puede realizarse en un único test que evalúe el árbol de directorios en su totalidad.

Es importante destacar que el proyecto tal y como está en GitHub no pasa todos los tests. En concreto, faltan las credenciales de AWS por motivos de seguridad, y también los archivos de origen del dataset, ya que supondría tener +8GB de CSVs en el repositorio.

Pueden ser lanzadas mediante el archivo “run\_tests.py” que se encuentra en la raíz del proyecto.

**DATOS:**

A lo largo de la ETL se van guardando y consultando distintos archivos. De esta forma, en caso de haber algún error que detenga la ETL, esta podría resumirse sin tener que volver a empezar de nuevo.

Este enfoque genera la necesidad de comprobar que los datos existen, si se están guardando, y de controlar las ocasiones en las que se intente acceder a un archivo inexistente. Para ello se diseñaron los test “test\_merge\_files” y “test\_parquets”.

Test\_merge\_files evalúa una función que lee todos los carchivos CSV de un directorio. Esta función de la capa raw debe lanzar una excepción en la que informa al usuario de que el directorio que se le ha indicado está vacío y detener la ejecución. A través de test\_merge\_files nos aseguramos de que efectivamente esta excepción se está lanzando y no se rompe más adelante la ETL al intentar acceder a un dataframe vacío.

Test\_parquets se asegura de que se han ido guardando los archivos que se generan entre las distintas capas de la ETL. En el caso de que no existan, verifica que se lanzan las excepciones adecuadas que infroman al usuario del problema para corregirlo de forma eficaz.

**ESTRUCTURA:**

En este apartado diferenciamos dos tipos de estructura: la estructura de directorios y la de los ficheros de datos.

* Estructura de directorios: Se compruba a través de test\_dirs. Nos sirve para saber que la estructura necesaria para la ingesta está presente.
* Estructura de datos: Es necesario verificar que los ficheros que se han enviado a la carpeta para la ingesta tienen la misma estructura, ya que esto podría causar problemas más adelante durante la ETL

**MÓDULOS:**

Los módulos en python han de seguir una estructura fija para poder importarse y relacionarse correctamente. Para comprobar que esta jerarquía de ficheros empleamos el test\_dirs que también usábamos para ver la estructura de ingesta.

**CREDENCIALES:**

Para poder conectarse a AWS mediante nuestro proyecto, es necesario que exista un fichero .env con las credenciales necesarias en el README.md.

En caso de que no existan, el test fallará e informará al usuario de que credenciales faltan por incluir, o si no existiese el .env.