Calidad de Datos

Versión 1.0

UCD - DNP

18 de junio de 2020

Contents

CHAPTER 1

Índice

CHAPTER 2

Introducción

La librería de calidad de datos tiene como objetivo principal ser una herramienta que facilite la verificación de contenido de bases de datos y dé métricas de calidad para que usuarios puedan decidir si sus bases de datos necesitan modificarse para ser utilizadas en los proyectos. La librería fue escrita en el lenguaje de programación de *Python* y puede analizar bases de datos estructurados que se conviertan en objetos *dataframe*. Contiene tres módulos principales, el módulo *Datos* para analizar cualquier base de datos, el módulo *Metadatos* para conectarse con los metadatos del Portal de Datos Abiertos de Colombia y utilizar sus bases de datos, y por último el módulo de *Reporte* para exportar los resultados obtenidos en los dos anteriores.

La librería surge como resultado de un proyecto realizado entre la UCD (Unidad de Científicos de Datos - DNP) y el Ministerio de Tecnologías de la Información y las Comunicaciones (MinTIC) relacionado con realizar análisis descriptivos de la calidad de la información cargada al portal de Datos Abiertos de Colombia, durante el desarrollo del proyecto se identifica el interés por parte de diferentes actores en el proyecto al igual que el beneficio potencial de tener a la mano una librería que facilite describir la calidad de una base de datos, lo cual motivó a realizar la implementación de la librería.

Esta página contiene toda la información relacionada con la librería, en el panel de navegación se tiene acceso a las diferentes secciones, las cuales cubren la instalación de la librería, la documentación de los módulos y funciones, ejemplos y demás información de interés.

CHAPTER 3

Instalación

En esta etapa de desarrollo de la librería aún no se cuenta con un procedimiento de instalación automática de las dependencias necesarias, dado lo anterior, se requiere que el usuario tenga acceso a los scripts de la librería y previamente tenga instalado las librerías de `pandas`, `numpy`, `sodapy`, `jinja2`.

Para la instalación de las librerías se recomienda utilizar un gestor de paquetes como `pip`, al igual que por buenas prácticas se sugiere crear un entorno virtual que permita aislar las librerías y evitar conflictos de versiones con el entorno de desarrollo base del computador.

A continuación se presentan los pasos a seguir para la creación del entorno virtual e instalación de librerías requeridas.

1. Creación del entorno virtual. Para esto se puede utilizar la librería `virtualenv`.

   ```
   virtualenv env
   ```

2. Activación del entorno virtual

   En linux:

   ```
   source env/bin/activate
   ```

   En Windows:

   ```
   source env/Scripts/activate
   ```

3. Una vez se active el entorno virtual se pueden instalar los requerimientos utilizando el archivo de requirements.txt

   ```
   pip install -r requirements.txt
   ```

4. De manera alterna se pueden instalar las diferentes librerías de manera independiente, a manera de ejemplo se muestra como instalar la librería `jinja2`.

   ```
   pip install Jinja2
   ```

5. Para desactivar el entorno virtual usar el comando deactivate

```
deactivate
```

CHAPTER 4

Datos

Este módulo se enfoca en analizar cualquier base de datos (dataframe) de interes para el usuario. Se tiene acceso a funciones de estadísticas descriptivas, cálculo de memoria, registros duplicados, faltantes y otras.

datos.**col_type**(*base*, *detail='low'*)

Retorna el tipo de dato de cada columna del dataframe, se clasifican como de tipo numérico, texto, boolean u otro.

Parámetros

- **base** – (dataframe) base de datos de interés a ser analizada.
- **detail** – (str) {"low", "high"}, valor por defecto: "low". Nivel de detalle en la descripción del tipo.

Devuelve serie de pandas con el tipo de dato de cada columna.

datos.**correlacion**(*base*, *metodo='pearson'*, *variables=None*)

datos.**data_summary**(*base*)

Retorna una tabla con información general de la base de datos. Incluye número de filas y columnas, número de columnas numéricas y de texto, número de columnas con más de la mitad de las observaciones con datos faltantes, número de columnas con más del 10% de observaciones con datos extremos y número de filas y columnas no únicas.

Parámetros **base** – (dataframe) base de datos de interés a ser analizada.

Devuelve serie de pandas con las estadísticas descriptivas del dataframe.

datos.**descriptive_stats**(*base*, *float_transform=False*)

Calcula estadísticas descriptivas de cada columna numérica. Incluyen media, mediana, valores en distintos percentiles, desviación estándar, valores extremos y porcentaje de valores faltantes.

Parámetros

- **base** – (dataframe) base de datos de interés a ser analizada.
- **float_transform** – (bool) {True, False}, valor por defecto: True. Si el valor es True se intenta realizar una transformación de valores de texto a numérico (float) para ser incluidas en el análisis, si el valor es False no se intenta realizar la transformación.

Devuelve dataframe con las estadísticas descriptivas.

datos.**duplic**(*base*, *col=True*)

Retorna las columnas o filas que presenten valores duplicados del dataframe.

Parámetros

- **base** – (dataframe) base de datos de interés a ser analizada.
- **col** – (bool) {True, False}, valor por defecto: True. Si el valor es True la validación se realiza por columnas, si el valor es False la validación se realiza por filas.

Devuelve matriz (dataframe) que relaciona las indices de filas/nombre de columnas que presentan valores duplicados.

datos.**memoria**(*base*, *col=False*)

Calcula el tamaño de la base de datos en memoria (megabytes)

Parámetros

- **base** – (dataframe) base de datos de interés a ser analizada.
- **col** – (bool) {True, False}, valor por defecto: False. Si el valor es False realiza el cálculo de memoria del dataframe completo, si el valor es True realiza el cálculo de memoria por cada columna del dataframe.

Devuelve valor (float) del tamaño de la base de datos en megabytes (si el parámetro col es False). Serie de pandas con el cálculo de memoria en megabytes por cada columna del dataframe. (si el parámetro col es True).

datos.**missing**(*base*, *perc=True*)

Calcula el porcentaje/número de valores faltantes de cada columna del dataframe.

Parámetros

- **base** – (dataframe) base de datos de interés a ser analizada.
- **perc** – (bool) {True, False}, valor por defecto: True. Si el valor es True el resultado se expresa como un porcentaje, si el valor es False el valor se expresa como una cantidad de registros (número entero).

Devuelve serie de pandas con la cantidad/porcentaje de valores faltantes de cada columna.

datos.**nounique**(*base*, *col=True*, *perc=True*)

Valida la unicidad del dataframe, retorna el porcentaje/número de filas o columnas no únicas en el dataframe.

Parámetros

- **base** – (dataframe) base de datos de interés a ser analizada.
- **col** – (bool) {True, False}, valor por defecto: True. Si el valor es True la validación se realiza por columnas, si el valor es False la validación se realiza por filas.
- **perc** – (bool) {True, False}, valor por defecto: True. Si el valor es True el resultado se expresa como un porcentaje, si el valor es False el valor se expresa como una cantidad de registros (número entero).

Devuelve (int o float) resultado de unicidad.

datos.**outliers**(*base*, *outliers='both'*, *perc=True*)

Calcula el porcentaje o cantidad de outliers de cada columna numérica (las columnas con números en formato string se intentarán transformar a columnas numéricas)

Parámetros

- **base** – (dataframe) base de datos de interés a ser analizada.

- **outliers** – (str) {"upper", "lower", "both"}, valor por defecto: "both". Si el valor es "**lower**" se tienen en cuenta los registros con valor menor al límite inferior calculado por la metodología de valor atípico por rango intercuartílico. Si el valor es "**upper**" se tienen en cuenta los registros con valor mayor al límite superior calculado por la metodología de valor atípico por rango intercuartílico. Si el valor es "**both**" se tienen en cuenta los registros con valor menor al límite inferior calculado por la metodología de valor atípico por rango intercuartílico, y también aquellos con valor mayor al límite superior calculado por la metodología de valor atípico por rango intercuartílico.
- **perc** – (bool) {True, False}, valor por defecto: True. Si el valor es True el resultado se expresa como un porcentaje, si el valor es False el valor se expresa como una cantidad de registros (número entero).

Devuelve serie de pandas con la cantidad/porcentaje de valores outliers de cada columna.

datos.**unique_col**(*base*, *missing=False*)

Calcula la cantidad de valores únicos de cada columna del dataframe.

Parámetros

- **base** – (dataframe) base de datos de interés a ser analizada.
- **missing** – (bool) {True, False}, valor por defecto: False. Indica si desea tener en cuenta los valores faltantes en el conteo de valores únicos.

Devuelve serie de pandas con la cantidad de valores únicos de cada columna.

datos.**unique_text**(*base*, *limit=0.5*, *nums=False*, *variables=None*)

Genera una tabla con los primeros 10 valores más frecuentes de las columnas de tipo texto del dataframe, además calcula su frecuencia y porcentaje dentro del total de observaciones. Incluye los valores faltantes.

Parámetros

- **base** – (dataframe) base de datos de interés a ser analizada.
- **limit** – (float) (valor de 0 a 1) límite de referencia, se utiliza para determinar si las variables posiblemente son de tipo categóricas y ser incluidas en el análisis. Si el número de valores únicos por columna es mayor al número de registros * limit, se considera que la variable no es categórica.
- **nums** – (bool) {True, False}, determina si se desea considerar las variables como categóricas e incluirlas en el análisis. Si el valor es True se incluyen las variables numéricas en el análisis, si el valor es False no se incluyen las variables numéricas en el análisis.
- **variables** – (str) nombres de las columnas separados por comas. Permite escoger las columnas de interés de análisis del dataframe

Devuelve dataframe con las estadísticas descriptivas de las columnas de tipo texto.

datos.**var_high_low**(*base*, *percent_low=5*, *percent_high=95*)

Retorna las columnas numéricas cuyo percentil inferior percent_low sea igual a su percentil superior percent_high.

Parámetros

- **base** – (dataframe) base de datos de interés a ser analizada.
- **percent_low** – (float), valor por defecto: 5. Percentil inferior de referencia en la comparación.
- **percent_high** – (float), valor por defecto: 95. Percentil superior de referencia en la comparación.

Devuelve indices de columnas cuyo percentil inferior es igual al percentil superior.

CHAPTER 5

Metadatos

Este módulo permite conectar y evaluar desde el código los metadatos del Portal de Datos Abiertos y descargar las bases de datos a dataframes.

metadatos.**asset_inventory**(*token=None*)

La función se conecta al API de Socrata y retorna la base de datos *Asset Inventory* descargada del Portal de Datos Abiertos como dataframe. Este conjunto de datos es un inventario de los recursos en el sitio.

Parámetros **token** – (str) *opcional* - token de usuario de la API Socrata.

Devuelve base de datos en formato dataframe.

metadatos.**asset_pretty**(*token=None*)

La función se conecta al API de Socrata y retorna la base de datos *Asset Inventory* descargada del Portal de Datos Abiertos como dataframe, selecciona columnas de interés y las renombra con un término en español. Este conjunto de datos es un inventario de los recursos en el sitio.

Parámetros **token** – (str) *opcional* - token de usuario de la API Socrata.

Devuelve base de datos en formato dataframe.

metadatos.**compare_meta**(*api_id*, *token=None*)

Se conecta al API de Socrata y consulta el número de filas y columnas de los metadatos de la base de datos asociada con el *api_id* para construir una tabla, adicionalmente se agregan los datos actuales del número de filas y columnas de la base de datos.

Parámetros

- **api_id** – (str) Identificación de la base de datos asociado con la API de Socrata.
- **token** – (str) *opcional* - token de usuario de la API Socrata.

Devuelve serie de pandas con el número de columnas y filas según los datos y metadatos.

metadatos.**info_cols_meta**(*api_id*)

Se conecta al API de Socrata y retorna el nombre, descripción y tipo de datos de cada una de las columnas de la base de datos asociada con el *api_id*.

Parámetros **api_id** – (str) Identificación de la base de datos asociado con la API de Socrata.

Devuelve base de datos en formato dataframe

metadatos.**meta_show**(*api_id*, *token=None*)

Se conecta al API de Socrata y retorna los metadatos asociados a la base del api_id.

Parámetros

- **api_id** – (str) Identificación de la base de datos asociado con la API de Socrata.
- **token** – (str) *opcional* - token de usuario de la API Socrata.

Devuelve serie de pandas con los metadatos.

metadatos.**sodapy_data**(*api_id*, *token=None*)

La función se conecta al API de Socrata y retorna la base de datos descargada del Portal de Datos Abiertos como dataframe.

Parámetros

- **api_id** – (str) Identificación de la base de datos asociado con la API de Socrata.
- **token** – (str) *opcional* - token de usuario de la API Socrata.

Devuelve base de datos en formato dataframe.

metadatos.**table_search**(*columnas_valor*, *columnas_operacion*)

Permite filtrar la base de datos de *Asset Inventory* de acuerdo a diferentes términos de búsqueda. Como son fechas, textos y otros.

Parámetros

- **columnas_valor** – (dictinario) {"nombre de columna":"valor a buscar o rangos"}. Corresponde al nombre de la columna a consultar y el valor a buscar.
- **columnas_operacion** – (str) {"contiene", "igual", "entre", "menor", "mayor", "mayor igual", "menor igual"} condición de comparación para filtrar la información.

Devuelve dataframe *Asset Inventory* filtrado con los términos de búsqueda).

metadatos.**updated_data**(*api_id*, *tipo_dato='metadatos'*)

Compara la última fecha de actualización de la base de datos con la frecuencia de actualización indicada en el *Asset Inventory*

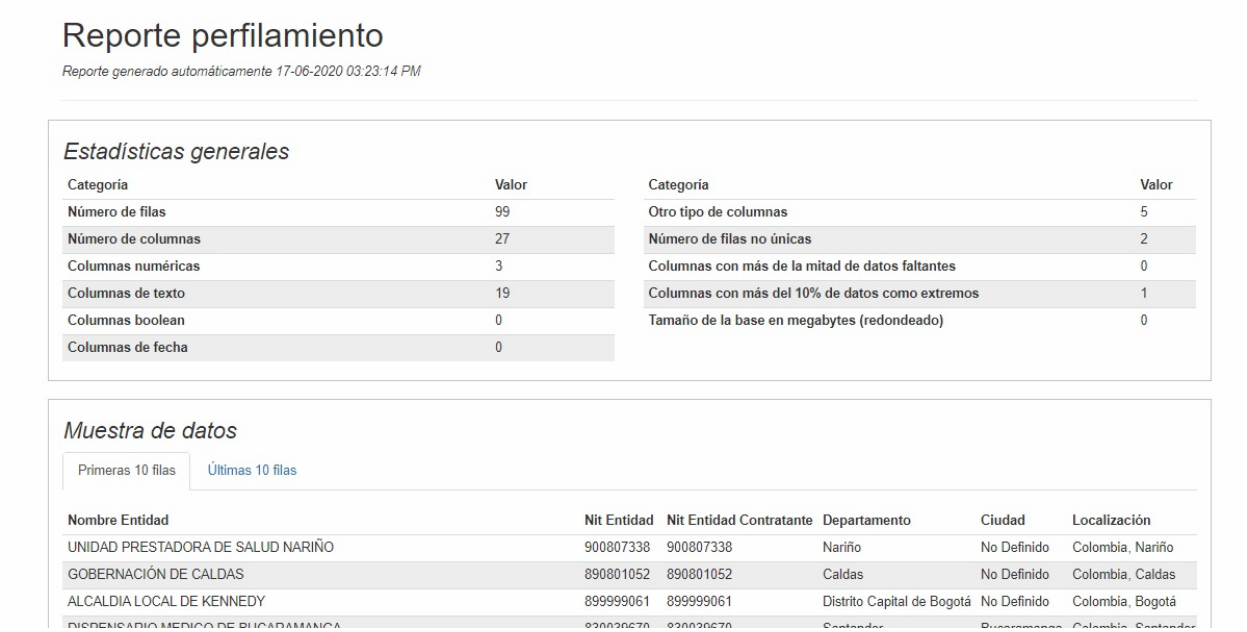
Parámetros

- **api_id** – (str) Identificación de la base de datos asociado con la API de Socrata.
- **tipo_dato** – (str) {"datos", "metadatos"}, valor por defecto: "metadatos. Fecha a validar, se tienen dos fechas de actualización, fecha de actualización de datos y fecha de actualización de metadatos.

Devuelve texto con validación de última fecha de actualización.

CHAPTER 6

Reporte



Reporte perfilamiento

Reporte generado automáticamente 17-06-2020 03:23:14 PM

Estadísticas generales

Categoría	Valor
Número de filas	99
Número de columnas	27
Columnas numéricas	3
Columnas de texto	19
Columnas boolean	0
Columnas de fecha	0

Categoría	Valor
Otro tipo de columnas	5
Número de filas no únicas	2
Columnas con más de la mitad de datos faltantes	0
Columnas con más del 10% de datos como extremos	1
Tamaño de la base en megabytes (redondeado)	0

Muestra de datos

Primeras 10 filas | Últimas 10 filas

Nombre Entidad	Nit Entidad	Nit Entidad Contratante	Departamento	Ciudad	Localización
UNIDAD PRESTADORA DE SALUD NARIÑO	900807338	900807338	Nariño	No Definido	Colombia, Nariño
GOBERNACIÓN DE CALDAS	890801052	890801052	Caldas	No Definido	Colombia, Caldas
ALCALDIA LOCAL DE KENNEDY	899999061	899999061	Distrito Capital de Bogotá	No Definido	Colombia, Bogotá

Figure1: Ejemplo de reporte

La función `generar_reporte(base)` del módulo **reporte** busca facilitar el proceso de entendimiento de la calidad de datos de una base de datos de interés, para esto genera un reporte en formato HTML el cual consolida las funciones de calidad de datos implementadas en el módulo *Datos* y *Metadatos*, facilitando la consulta de los resultados obtenidos del análisis exploratorio en un archivo independiente.

`reporte.`**`generar_reporte`**(*base*, *titulo='Reporte perfilamiento'*, *archivo='perfilamiento.html'*)

Genera un reporte de calidad de datos en formato HTML

Parámetros

- **base** – (dataframe) base de datos de insumo para la generación del reporte de calidad de datos.
- **titulo** – (str) valor por defecto: "Reporte perfilamiento". Título del reporte a generar.
- **archivo** – (str) valor por defecto: "perfilamiento.html". Ruta donde guardar el reporte.

Devuelve archivo de reporte en formato HTML.

CHAPTER 7

Introducción

La librería de calidad de datos tiene como objetivo principal ser una herramienta que facilite la verificación de contenido de bases de datos y dé métricas de calidad para que usuarios puedan decidir si sus bases de datos necesitan modificarse para ser utilizadas en los proyectos. La librería fue escrita en el lenguaje de programación de *Python* y puede analizar bases de datos estructurados que se conviertan en objetos *dataframe*. Contiene tres módulos principales, el módulo *Datos* para analizar cualquier base de datos, el módulo *Metadatos* para conectarse con los metadatos del Portal de Datos Abiertos de Colombia y utilizar sus bases de datos, y por último el módulo de *Reporte* para exportar los resultados obtenidos en los dos anteriores.

La librería surge como resultado de un proyecto realizado entre la UCD (Unidad de Científicos de Datos - DNP) y el Ministerio de Tecnologías de la Información y las Comunicaciones (MinTIC) relacionado con realizar análisis descriptivos de la calidad de la información cargada al portal de Datos Abiertos de Colombia, durante el desarrollo del proyecto se identifica el interés por parte de diferentes actores en el proyecto al igual que el beneficio potencial de tener a la mano una librería que facilite describir la calidad de una base de datos, lo cual motivó a realizar la implementación de la librería.

Esta página contiene toda la información relacionada con la librería, en el panel de navegación se tiene acceso a las diferentes secciones, las cuales cubren la instalación de la librería, la documentación de los módulos y funciones, ejemplos y demás información de interés.

Índice de Módulos Python