RB670

March 19, 2024

1 CODIGO PARA LIMPIEZA DE ARCHIVOS ORIGINALES

1.1 RESUMEN DE NUESTRO CODIGO

El presente codigo se encarga de realizar la limpieza de los archivos fuentes originales (con formato .csv). Ademas, crearemos un archivo en excel con 02 pestanas que contendran ya los datos filtrados para ser usados en los siguientes programas **RB670-VEN099** y **RB670-VEN103**.

Para este fin, seguiremos los siguientes pasos:

1.1.1 PASO 01: IMPORTACION DE LIBRERIAS

Para el presente codigo, importaremos 02 librerias de Python:

- **Pandas**, para realizar todas nuestras operaciones de limpieza.
- **OS**, para poder trabajar con las rutas actuales de nuestro proyecto, de tal forma que sin importar donde guardemos la carpeta del proyecto, el codigo siga funcionando correctamente.

[ ]:
```
import pandas as pd
import os
```

1.1.2 PASO 02: OBTENCION DE RUTAS DE UBICACION DE ARCHIVOS

En esta seccion, usaremos el metodo *getcwd()* de la libreria *OS* para obtener la ubicacion actual de nuestro proyecto. Esto con el fin de poder importar nuestros archivos *.csv* sin importar donde se guarde todo el proyecto ya que la *estructura de almacenamiento* de nuestros archivos origen se mantendra

[ ]:
```
cwd = os.getcwd()
dsf_VEN099 = cwd +'\Datos ventiladores del RB670'+'\ORIGINALES'+'\RB670-VEN099.
 ↪csv'
dsf_VEN103 = cwd +'\Datos ventiladores del RB670'+'\ORIGINALES'+'\RB670-VEN103.
 ↪csv'
```

1.1.3 PASO 03: IMPORTACION Y ACONDICIONAMIENTO DE DATOS

En esta seccion, importaremos los datos de nuestros archivos *.csv* para realizar las operaciones respectivas.

Para este caso, forzaremos la columna *Fecha y Hora* para que sea de tipo *datetime*. Ademas, definiremos dicha columna como *indice*.

Ademas, importaremos solo *180000 filas* de cada archivo con formato *latin1*.

Debido a que, durante las revisiones de los archivos fuentes, se encontro que existian filas que se guardaron como si fueran de tipo *texto*, *forzaremos* que todas las filas se reconozcan como *numericos*

[ ]:
```
df_VEN099 = pd.read_csv(dsf_VEN099,parse_dates=['Fecha y␣
↪hora'],index_col='Fecha y hora',nrows=180000, encoding='latin1')
df_VEN099 = df_VEN099.apply(pd.to_numeric, errors='coerce')
```

[ ]:
```
df_VEN103 = pd.read_csv(dsf_VEN103,parse_dates=['Fecha y␣
↪hora'],index_col='Fecha y hora',nrows=180000, encoding='latin1')
df_VEN103 = df_VEN103.apply(pd.to_numeric, errors='coerce')
```

1.1.4 PASO 04: LIMPIEZA DE DATOS

Luego de importar todos los datos como tipo *numerico*, ahora procedemos a realizar la limpieza respectiva de nuestros datos. Como primer paso, reemplazaremos los registros de las columnas que muestran *valores negativos* ya que, debido a la naturaleza de los datos, todos los valores deben ser *mayores o iguales a cero*. Para el caso de los *valores negativos*, igualaremos todos a *cero* ya que este es el valor correcto que se debe asignar. Dicha accion se realizara usando el metodo *mask* disponibles para todos los *dataframes*.

Finalmente, para que estos cambios se actualicen en nuestros *dataframes*, haremos uso de la sentencia *inplace=True*

[ ]:
```
df_VEN099['Vibración Vent 2'].mask((df_VEN099['Vibración Vent 2']<0.
↪0),0,inplace=True)
```

[ ]:
```
df_VEN103['Vibración Motor 1'].mask((df_VEN103['Vibración Motor 1']<0.
↪0),0,inplace=True)
```

[ ]:
```
df_VEN103['Vibración Motor 2'].mask((df_VEN103['Vibración Motor 2']<0.
↪0),0,inplace=True)
```

[ ]:
```
df_VEN103['Vibración Vent 1'].mask((df_VEN103['Vibración Vent 1']<0.
↪0),0,inplace=True)
```

[ ]:
```
df_VEN103['Vibración Vent 2'].mask((df_VEN103['Vibración Vent 2']<0.
↪0),0,inplace=True)
```

1.1.5 PASO 05: LLENADO DE REGISTROS CON NAN

Luego de realizar el *paso 03*, los registros de las columnas que hayan tenido valores que no sean de tipo *numerico*, se llenaran con el valor *NAN*. En este caso, disponemos del metodo *fillna*, el cual se encargara de reemplazar los registros con valor *NAN* con valores igual a 0.

[ ]:
```
df_VEN099.fillna(0,inplace=True)
```

[ ]:
```
df_VEN103.fillna(0,inplace=True)
```

1.1.6 PASO 06: CREACION DE RUTA DE ALMACENAMIENTO

Luego de haber realizado la *limpieza* de nuestros datos, ahora procedemos a crear la direccion de almacenamiento de nuestros datos limpios, asi como *el nombre de nuestro archivo*.

[ ]:
```
ddf = cwd +'\Datos ventiladores del RB670'+'\FILTRADO'+'\Analisis_RB670.xlsx'
```

1.1.7 PASO 07: ALMACENAMIENTO DE DATOS LIMPIOS

Finalmente, procedemos a crear nuestro archivo en excel, el cual contendra 02 pestanas con los nombres de los archivos originales. Al igual que en los archivos originales, usaremos la columna *Fecha y hora* como nuestro *indice*. Tener presente que, para la *creacion* de nuestra segunda pestana, haremos uso del metodo en *pandas* llamado *ExcelWriter*. Esto con el fin de evitar que los datos del segundo dataframe sobreescriban a los datos del primero.

[ ]:
```
df_VEN099.to_excel(ddf,sheet_name='VEN099',index='Fecha y hora')
```

[ ]:
```
with pd.ExcelWriter(ddf,engine='openpyxl',mode='a') as writer:
    df_VEN103.to_excel(writer,sheet_name='VEN103',index='Fecha y hora')
```