Learning_Pandas_Part_102_SASwithPython

June 20, 2021

0.0.1 Prepared by Abhishek Kumar

0.0.2 https://www.linkedin.com/in/abhishekkumar-0311/

1 SAS with Python - saspy module

1.1 Step 0 : Environment Setup

[1]:
```
# To get multiple outputs in the same cell

from IPython.core.interactiveshell import InteractiveShell
InteractiveShell.ast_node_interactivity = "all"
```

[2]:
```
# Supress Warnings

import warnings
warnings.filterwarnings('ignore')
```

[3]:
```
import numpy as np
from scipy import stats
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

%matplotlib inline
```

[4]:
```
# Set the required global options

# To display all the columns in dataframe
pd.set_option( "display.max_columns", None)
pd.set_option( "display.max_rows", None)
```

1.2 Step 1 : Configure SAS Session

- Start SAS Session
- Enter Login Credentials

[ ]:
```
import saspy
```

```
sas = saspy.SASsession(java='C:\\Program Files\\Java\\jdk-15.0.1\\bin\\java.
↪exe', iomhost=['odaws01-apse1.oda.sas.com','odaws02-apse1.oda.sas.com'],␣
↪iomport=8591, encoding='utf-8')
sas
#############################################################################################
↪abhi0311sharma0
#############################################################################################
↪SASthepower2KNOW@
```

```
Using SAS Config named: default
```

1.3 Step 2 : Run SAS Procedure

```
[ ]: %%SAS sas
     proc print data=sashelp.cars (obs=4);
     run;
```

```
[ ]: sc = "proc print data=sashelp.cars (obs=5); run;"
     scp = sas.submitLST(sc, method='listorlog')
```

```
[ ]: sc = "proc sql; create table work.dict_tables as select * from dictionary.
     ↪tables; quit;"
     scp = sas.submitLST(sc, method='listorlog')
```

```
[ ]: sc = "proc print data=work.dict_tables (obs=5); run;"
     scp = sas.submitLST(sc, method='listorlog')
```

1.4 Step 3 : Transfer Data between Pandas Dataframe and SAS

- *Function* ***df2sd*** *converts pandas dataframe to sas dataset.*

```
[ ]: pandasdf = pd.read_csv("./heart.csv")
     type(pandasdf)
     sasdf = sas.df2sd(pandasdf, 'sasdf')
     type(sasdf)
     sas.submitLST("proc print data=work.sasdf (obs=3);run;", method='listorlog')
```

- *Function* ***sd2df*** *converts sas dataset to pandas dataframe.*

```
[ ]: pandasdf2 = sas.sd2df(sasdf.table)
     type(pandasdf2)
     pandasdf2.head()
```

1.4.1 Creating a saspy.sasdata.SASdata Object

[ ]:
```
cars = sas.sasdata('cars', 'sashelp')
type(cars)
cars.head()
```

[ ]:
```
dict_tables = sas.sasdata('vtable', 'sashelp')
type(dict_tables)
dict_tables.head(3)
```

[ ]:
```
x = dict_tables
type(x)
x.head(2)
```

1.4.2 SAS DS to Pandas DF - Method 1

- Creating a dataset in sas
- Converting it to Pandas Dataframe using **sd2df**

[ ]:
```
### RUN this code snippet at the end

# sas.submitLST("data dict_tables; set sashelp.vtable; run;",␣
 ↪method='listonly') # method='listorlog'
# pandasdf2 = sas.sd2df(dict_tables.table)
# type(pandasdf2)
# pandasdf2.head(2)
```

1.4.3 SAS DS to Pandas DF - Method 2

- Creating a SAS Data Object
- Using SAS Data Object attribute **SAS_Data_Obj.to_df()** to convert to Pandas DataFrame Object
- sas-data-object

[ ]:
```
dict_tables = sas.sasdata('dict_tables', 'work')
type(dict_tables)
dict_tables.head(3)
```

[ ]:
```
s = dict_tables.to_df()
type(s)
s.tail(2)
```

[ ]:
```
#s.T
```

Re-confirming Method 1 : A sas dataset created in sas. Then converted to Python Dataframe using sd2df

[ ]:
```
sc = "proc sql; create table work.dict_tables as select * from dictionary.
↪tables; quit;"
scp = sas.submitLST(sc, method='listonly')
```

[ ]:
```
dict_table_sql = sas.sd2df(dict_tables.table)
type(dict_table_sql)
dict_table_sql.head(2)
```

[ ]:

[ ]:

[ ]:

[ ]:

[ ]:

[ ]:

[ ]:

[ ]:

1.5 2. Reading the Input data (csv) file

[ ]:
```
 heart = pd.read_csv('./heart.csv')
```

[ ]:
```
heart.head()
```

1.6 3. Data Analysis & Cleaning

[ ]:
```
# Checking rows and columns - shape
heart.shape
```

[ ]:
```
# Getting the overview of Data types and Non-Null info
heart.info()
```

1.6.1 Checking Missing Values

[ ]:
```
# Checking for any Null columns
heart.isnull().sum().any()

heart.shape[0]

# Finding the columns with more than 40% NULLs.
```

```
ser = heart.isnull().sum()/len(heart)*100
null_drps = ser[ser > 40]
null_drps
```

[ ]:
```
# Checking the info of the remaining columns with NULLs
heart[nulls.index].info()
```

[ ]:
```

```