Tutorial on dealing with missing values

Anil Jadhav , SCIT, Pune

In [470]:
```
#!python -m pip install scikit-learn --user --upgrade pip
#for knn imputation and advanced regression based imputation
```

In [1]:
```
import sys
print(sys.version)
```

```
3.7.1 (default, Dec 10 2018, 22:54:23) [MSC v.1915 64 bit (AMD64)]
```

In [3]:
```
import pandas as pd
import numpy as np
```

In [3]:
```
def num_missing(x):
    return sum(x.isnull())

#print("missing values per column:")
#print(data.apply(num_missing, axis=0))
#axis=0 defines that function is to be applied on each column

#print("missing values per row:")
#print(data.apply(num_missing,axis=1).head())
#axis=1 defines that function is to be applied on each row
```

In [4]:
```
df = pd.DataFrame([[np.nan, 2, np.nan, 0],
...                    [3, 4, 5, 1],
...                    [np.nan, np.nan, np.nan, 5],
...                    [np.nan, 3, np.nan, 4]],
...                    columns=list('ABCD'))
```

In [7]:
```
df
```

Out[7]:

	A	B	C	D
0	NaN	2.0	NaN	0
1	3.0	4.0	5.0	1
2	NaN	NaN	NaN	5
3	NaN	3.0	NaN	4

In [8]:
```
df.isnull()
```

Out[8]:

	A	B	C	D
0	True	False	True	False
1	False	False	False	False
2	True	True	True	False
3	True	False	True	False

In [11]:
```
#count number of missing values in each column
df.isnull().sum()
#df.isnull().sum(axis=0)
```

Out[11]:
```
A    3
B    1
C    3
D    0
dtype: int64
```

In [12]:
```
#count number of missing values in each row
df.isnull().sum(axis=1)
```

Out[12]:
```
0    2
1    0
2    3
3    2
dtype: int64
```

In [9]:
```
#Check which columns have missing values
df.isnull().any(axis=0)
```

Out[9]:
```
A     True
B     True
C     True
D    False
dtype: bool
```

In [12]:
```
#count number of columns having missing values
df.isnull().any().sum()
```

Out[12]:
```
3
```

In [13]:
```
#Check which rows have missing values
df.isnull().any(axis=1)
```

Out[13]:
```
0     True
1    False
2     True
3     True
dtype: bool
```

In [14]:
```
#count number of rows having missing value in any column
df.isnull().any(axis=1).sum()
```

Out[14]: 3

In [15]:
```
#names of columns having missing values
df.columns[df.isnull().any()]
```

Out[15]: Index(['A', 'B', 'C'], dtype='object')

In [16]:
```
#Names of the columns having no missing values
df.columns[~df.isnull().any()]
```

Out[16]: Index(['D'], dtype='object')

In [17]:
```
#display all rows where there are missing value in any one column
df[df.isnull().any(axis=1)]
```

Out[17]:

	A	B	C	D
0	NaN	2.0	NaN	0
2	NaN	NaN	NaN	5
3	NaN	3.0	NaN	4

In [18]:
```
#display all rows with no missing value in any one column
df[~df.isnull().any(axis=1)]
```

Out[18]:

	A	B	C	D
1	3.0	4.0	5.0	1

In [19]:
```
#display all columns with  missing value in any row
df[df.columns[df.isnull().any()]]
#df.loc[:,list(df.columns[df.isnull().any()])]
```

Out[19]:

	A	B	C
0	NaN	2.0	NaN
1	3.0	4.0	5.0
2	NaN	NaN	NaN
3	NaN	3.0	NaN

In [20]:
```
#display all columns with no missing value in any row
df[df.columns[~df.isnull().any()]]
#df.loc[:,list(df.columns[~df.isnull().any()])]
```

Out[20]:

	D
0	0
1	1
2	5
3	4

In [21]:
```
df = pd.DataFrame([[np.nan, 2, np.nan, 0],
...                    [3, 4, 5, 1],
...                    [np.nan, np.nan, np.nan, 5],
...                    [np.nan, 3, np.nan, 4]],
...                    columns=list('ABCD'))
df
```

Out[21]:

	A	B	C	D
0	NaN	2.0	NaN	0
1	3.0	4.0	5.0	1
2	NaN	NaN	NaN	5
3	NaN	3.0	NaN	4

In [22]:
```
#drop all rows where there is missing value in any column
#display complete rows
df.dropna()
#df.dropna(inplace=True)
```

Out[22]:

	A	B	C	D
1	3.0	4.0	5.0	1

In [23]:
```
#Drop all columns where there is missing value in any row
df.dropna(axis=1)
```

Out[23]:

	D
0	0
1	1
2	5
3	4

In [491]:
```
#from pandas import DataFrame
#help(DataFrame.dropna)
```

In [30]:
```
#df.drop(['A','B'],axis=1,inplace=True)
```

In [34]:
```
bankdata_m = pd.read_csv("d:/ds & da symbi/bankdata_m.csv")
```

In [36]:
```
bankdata_m.shape
```

Out[36]: (700, 10)

In [39]:
```
bankdata_m.head(5)
```

Out[39]:

	Unnamed: 0	Age	Education	Employment	Address	Income	DebtToIncome	CreditToDebt	Oth
0	1	41.0	3.0	17.0	12.0	176.0	9.3	11.359392	5.
1	2	27.0	1.0	10.0	6.0	31.0	17.3	1.362202	4.
2	3	40.0	1.0	15.0	14.0	55.0	5.5	0.856075	2.
3	4	41.0	1.0	15.0	14.0	120.0	2.9	2.658720	0.
4	5	24.0	2.0	2.0	0.0	28.0	17.3	1.787436	3.

In [46]:
```
bankdata_m.drop('Unnamed: 0',axis=1,inplace=True)
```

In [49]:
```
bankdata_m.isnull().sum(axis=0)
```

Out[49]:
```
Age             26
Education       19
Employment      21
Address         22
Income          22
DebtToIncome    23
CreditToDebt    21
OtherDebt       17
IsDefaulted     18
dtype: int64
```

In [51]:
```
bankdata_m.isnull().sum(axis=1)
```

Out[51]:
```
0      0
1      0
2      0
3      0
4      0
      ..
695    0
696    1
697    0
698    0
699    0
Length: 700, dtype: int64
```

In [66]:
```
sum(bankdata_m.isnull().any(axis=1))
```

Out[66]: 174

In [67]:
```
700-174
```

Out[67]: 526

In [62]:
```
bankdata_m.isnull().any(axis=0).sum()
```

Out[62]: 9

In [63]:
```
bankdata_m.dropna()
```

Out[63]:

	Age	Education	Employment	Address	Income	DebtToIncome	CreditToDebt	OtherDebt	I
0	41.0	3.0	17.0	12.0	176.0	9.3	11.359392	5.008608	
1	27.0	1.0	10.0	6.0	31.0	17.3	1.362202	4.000798	
2	40.0	1.0	15.0	14.0	55.0	5.5	0.856075	2.168925	
3	41.0	1.0	15.0	14.0	120.0	2.9	2.658720	0.821280	
4	24.0	2.0	2.0	0.0	28.0	17.3	1.787436	3.056564	
...	...	...	...	...	...	...	...	...	
694	48.0	2.0	6.0	1.0	66.0	12.1	2.315940	5.670060	
695	36.0	2.0	6.0	15.0	27.0	4.6	0.262062	0.979938	
697	33.0	1.0	15.0	3.0	32.0	7.6	0.491264	1.940736	
698	45.0	1.0	19.0	22.0	77.0	8.4	2.302608	4.165392	
699	37.0	1.0	12.0	14.0	44.0	14.7	2.994684	3.473316	

526 rows × 9 columns

Replace missing values

In [24]:
```
df = pd.DataFrame([[np.nan, 2, np.nan, 0],
...                    [3, 4, 5, 1],
...                    [np.nan, np.nan, np.nan, 5],
...                    [np.nan, 3, np.nan, 4]],
...                    columns=list('ABCD'))
df
```

Out[24]:

	A	B	C	D
0	NaN	2.0	NaN	0
1	3.0	4.0	5.0	1
2	NaN	NaN	NaN	5
3	NaN	3.0	NaN	4

In [25]:
```
#replace all missing values in column A with 100
df.A.fillna(100)
```

Out[25]:
```
0    100.0
1      3.0
2    100.0
3    100.0
Name: A, dtype: float64
```

In [26]:
```
#replace all missing values in column B with mean of values in columns b
df.B.fillna(df.B.mean())
```

Out[26]:
```
0    2.0
1    4.0
2    3.0
3    3.0
Name: B, dtype: float64
```

In [27]:
```
df.fillna(0) #replace all missing values with 0
```

Out[27]:

	A	B	C	D
0	0.0	2.0	0.0	0
1	3.0	4.0	5.0	1
2	0.0	0.0	0.0	5
3	0.0	3.0	0.0	4

In [28]:
```
#replace all NaN elements in column ‘A’, ‘B’, ‘C’, and ‘D’, with 0, 1, 2, and
 3 respectively.
values = {'A': 0, 'B': 1, 'C': 2, 'D': 3}
df.fillna(value=values)
```

Out[28]:

	A	B	C	D
0	0.0	2.0	2.0	0
1	3.0	4.0	5.0	1
2	0.0	1.0	2.0	5
3	0.0	3.0	2.0	4

In [29]:
```
#Replace all missing values with its column mean
df.fillna(df.mean())#use inplace =True if you want to make changes in df
```

Out[29]:

	A	B	C	D
0	3.0	2.0	5.0	0
1	3.0	4.0	5.0	1
2	3.0	3.0	5.0	5
3	3.0	3.0	5.0	4

https://scikit-learn.org/stable/modules/impute.html (https://scikit-learn.org/stable/modules/impute.html)

In [30]:
```
from sklearn.impute import SimpleImputer
```

In [31]:
```
df = pd.DataFrame([[np.nan, 2, np.nan, 0],
...                    [3, 4, 5, 1],
...                    [np.nan, np.nan, np.nan, 5],
...                    [np.nan, 3, np.nan, 4]],
...                    columns=list('ABCD'))
df
```

Out[31]:

	A	B	C	D
0	NaN	2.0	NaN	0
1	3.0	4.0	5.0	1
2	NaN	NaN	NaN	5
3	NaN	3.0	NaN	4

In [32]:
```
#The first parameter is how is missing value represented in the dataset
imp = SimpleImputer(missing_values=np.nan, strategy='mean') #strategy could be
mean, meadian, most_frequent or constant
```

In [33]:
```
df[:]=imp.fit_transform(df)
```

In [34]:
```
df
```

Out[34]:

	A	B	C	D
0	3.0	2.0	5.0	0.0
1	3.0	4.0	5.0	1.0
2	3.0	3.0	5.0	5.0
3	3.0	3.0	5.0	4.0

In [35]:
```
df = pd.DataFrame([["a", "x"],
...                    [np.nan, "y"],
...                    ["a", np.nan],
...                    ["b", "y"]], dtype="category")
...
df
```

Out[35]:

	0	1
0	a	x
1	NaN	y
2	a	NaN
3	b	y

In [36]:
```
imp = SimpleImputer(missing_values=np.nan, strategy='most_frequent')
df[:]=imp.fit_transform(df)
```

In [37]:
```
df
```

Out[37]:

	0	1
0	a	x
1	a	y
2	a	y
3	b	y

In [38]:
```
df = pd.DataFrame([[np.nan, 2, np.nan, 0],
...                    [3, 4, 3, 1],
...                    [np.nan, 3, 4, 5],
...                    [np.nan, 3, np.nan, np.nan]],
...                    columns=list('ABCD'))
df
```

Out[38]:

	A	B	C	D
0	NaN	2	NaN	0.0
1	3.0	4	3.0	1.0
2	NaN	3	4.0	5.0
3	NaN	3	NaN	NaN

In [39]:
```
imp = SimpleImputer(missing_values=np.nan, strategy='constant',fill_value=0)#r
eplace all missing values with constant '0'
df[:]=imp.fit_transform(df)
```

In [40]:
```
df[:]
```

Out[40]:

	A	B	C	D
0	0.0	2.0	0.0	0.0
1	3.0	4.0	3.0	1.0
2	0.0	3.0	4.0	5.0
3	0.0	3.0	0.0	0.0

In [41]:
```
import numpy as np
from sklearn.impute import KNNImputer
nan = np.nan
X = [[1, 2, nan], [3, 4, 3], [nan, 6, 5], [8, 8, 7]]
```

In [42]:
```
df=pd.DataFrame(X)
df
```

Out[42]:

	0	1	2
0	1.0	2	NaN
1	3.0	4	3.0
2	NaN	6	5.0
3	8.0	8	7.0

In [43]:
```
imputer = KNNImputer(n_neighbors=1)
```

In [44]:
```
df[:]=imputer.fit_transform(X)
```

In [45]:
```
df
```

Out[45]:

	0	1	2
0	1.0	2.0	3.0
1	3.0	4.0	3.0
2	3.0	6.0	5.0
3	8.0	8.0	7.0

In [46]:
```
#scikit learn version should be .21
#print('The scikit-learn version is {}.'.format(sklearn.__version__))
from sklearn.experimental import enable_iterative_imputer
from sklearn.impute import IterativeImputer
```

In [47]:
```
df = pd.DataFrame([[np.nan, 2, np.nan, 0],
...                    [3, 4, 3, 1],
...                    [np.nan, 3, 4, 5],
...                    [np.nan, 3, np.nan, np.nan]],
...                    columns=list('ABCD'))
df
```

Out[47]:

	A	B	C	D
0	NaN	2	NaN	0.0
1	3.0	4	3.0	1.0
2	NaN	3	4.0	5.0
3	NaN	3	NaN	NaN

In [48]:
```
imp = IterativeImputer()
imp.fit_transform(df)
```

Out[48]:
```
array([[3.        , 2.        , 2.88235541, 0.        ],
       [3.        , 4.        , 3.        , 1.        ],
       [3.        , 3.        , 4.        , 5.        ],
       [3.        , 3.        , 3.29411847, 2.        ]])
```

In [ ]:

In [469]:
```
#df.fillna(method='ffill')#forward fill
#df.fillna(method='bfill')#backward fill
```

In [ ]: