Missing values analysis

- MVA very importnat in Data analysis
- It is quiet natural data set has missing values
- There are so many techniques are there
    - Mean
    - Median
    - Mode
    - bfill
    - ffill
    - pad
    - random values
    - You can compare column values and you can impute
    - KNN imputer

In [1]:
```
import pandas as pd
import numpy as np
```

In [3]:
```
dict1={'Names':['Ramesh','Suresh','Sathish',np.nan],
       'Ages':[30,31,np.nan,32],
       'City':[np.nan,'Hyd','Pune','Chennai']}

d1=pd.DataFrame(dict1)
d1
```

Out[3]:

	Names	Ages	City
0	Ramesh	30.0	NaN
1	Suresh	31.0	Hyd
2	Sathish	NaN	Pune
3	NaN	32.0	Chennai

In [6]:
```
d1.isnull()
```

Out[6]:

	Names	Ages	City
0	False	False	True
1	False	False	False
2	False	True	False
3	True	False	False

In [7]:
```
d1.isnull().sum()
```

Out[7]:
```
Names     1
Ages      1
City      1
dtype: int64
```

In [8]:
```
dict1={'Names':['Ramesh','Suresh','Sathish','Null'],
       'Ages':[30,31,'Null',32],
       'City':['Null','Hyd','Pune','Chennai']}

d2=pd.DataFrame(dict1)
d2
```

Out[8]:

	Names	Ages	City
0	Ramesh	30	Null
1	Suresh	31	Hyd
2	Sathish	Null	Pune
3	Null	32	Chennai

In [9]:
```
d2.isnull()
```

Out[9]:

	Names	Ages	City
0	False	False	False
1	False	False	False
2	False	False	False
3	False	False	False

In [10]:
```
dict1={'Names':['Ramesh','Suresh','Sathish',None],
       'Ages':[30,31,None,32],
       'City':[None,'Hyd','Pune','Chennai']}

d3=pd.DataFrame(dict1)
d3
```

Out[10]:

	Names	Ages	City
0	Ramesh	30.0	None
1	Suresh	31.0	Hyd
2	Sathish	NaN	Pune
3	None	32.0	Chennai

In [11]:
```
d3.isnull()
```

Out[11]:

	Names	Ages	City
0	False	False	True
1	False	False	False
2	False	True	False
3	True	False	False

In [ ]:
```
Values are not present, empty box is there ===== Null value

csv  we dont have  np.nan
```

In [12]:
```
d1.to_csv('d1.csv')
```

- np.nan
- None is working
- Some times when you read csv if you have mising values it shows as Null
- isnull()

$Method - 1$

Random fill

In [13]:
```
d1
```

Out[13]:

	Names	Ages	City
0	Ramesh	30.0	NaN
1	Suresh	31.0	Hyd
2	Sathish	NaN	Pune
3	NaN	32.0	Chennai

In [14]:
```
d1.fillna(40,inplace=True)
d1
```

Out[14]:

	Names	Ages	City
0	Ramesh	30.0	40
1	Suresh	31.0	Hyd
2	Sathish	40.0	Pune
3	40	32.0	Chennai

In [15]:
```
d1.dtypes
```

Out[15]:
```
Names      object
Ages      float64
City       object
dtype: object
```

$Method - 2$

Fill the random values based on column

In [16]:
```
dict1={'Names':['Ramesh','Suresh','Sathish',np.nan],
       'Ages':[30,31,np.nan,32],
       'City':[np.nan,'Hyd','Pune','Chennai']}

d1=pd.DataFrame(dict1)
d1
```

Out[16]:

	Names	Ages	City
0	Ramesh	30.0	NaN
1	Suresh	31.0	Hyd
2	Sathish	NaN	Pune
3	NaN	32.0	Chennai

In [17]:
```
d1['Ages'].fillna(40,inplace=True)
d1
```

Out[17]:

	Names	Ages	City
0	Ramesh	30.0	NaN
1	Suresh	31.0	Hyd
2	Sathish	40.0	Pune
3	NaN	32.0	Chennai

$Method - 3$

- Mean
- Median
- Mode

In [18]:
```
# Read the data again
dict1={'Names':['Ramesh','Suresh','Sathish',np.nan],
       'Ages':[30,31,np.nan,32],
       'City':[np.nan,'Hyd','Pune','Chennai']}

d1=pd.DataFrame(dict1)
d1
```

Out[18]:

	Names	Ages	City
0	Ramesh	30.0	NaN
1	Suresh	31.0	Hyd
2	Sathish	NaN	Pune
3	NaN	32.0	Chennai

- Median does not affect by outliers
- Mean affect by outliers
- If you have a data is there,
  - First check the outlier , by simple boxplot
  - You can check with created outlier function
  - If you feel there is outliers are there

- Then fill missing values with Median

In [21]:
```
age_mean=d1['Ages'].mean()
age_mean
d1['Ages'].fillna(age_mean,inplace=True)
d1
```

Out[21]:

	Names	Ages	City
0	Ramesh	30.0	NaN
1	Suresh	31.0	Hyd
2	Sathish	31.0	Pune
3	NaN	32.0	Chennai

In [ ]:
```
# age_median=d1['Ages'].median()
# d1['Ages'].fillna(age_median,inplace=True)
# d1
```

In [22]:
```
name_mode=d1['Names'].mode()
name_mode
# d1['Nmaes'].fillna(name_mode,inplace=True)
# d1
```

Out[22]:
```
0     Ramesh
1    Sathish
2     Suresh
Name: Names, dtype: object
```

Method-4

- pad
- bfill
- ffill
- backfill

In [23]:
```
# again read the data
dict1={'Names':['Ramesh','Suresh','Sathish',np.nan],
       'Ages':[30,31,np.nan,32],
       'City':[np.nan,'Hyd','Pune','Chennai']}

d1=pd.DataFrame(dict1)
d1
```

Out[23]:

	Names	Ages	City
0	Ramesh	30.0	NaN
1	Suresh	31.0	Hyd
2	Sathish	NaN	Pune
3	NaN	32.0	Chennai

In [24]:
```
d1.fillna(method='pad')
# Names= 'Sathish'
# Ages='31'
# City= NaN

# It is filling by previous value by Column reference
# For the column axis=1
```

Out[24]:

	Names	Ages	City
0	Ramesh	30.0	NaN
1	Suresh	31.0	Hyd
2	Sathish	31.0	Pune
3	Sathish	32.0	Chennai

In [33]:
```
d1.fillna(method='pad',axis=1)
```

Out[33]:

	Names	Ages	City
0	Ramesh	30.0	30.0
1	Suresh	31.0	Hyd
2	Sathish	Sathish	Pune
3	NaN	32.0	Chennai

In [34]:
```
print('===========pad================')
print(d1.fillna(method='pad'))
print('===========bfill================')
print(d1.fillna(method='bfill'))
print('===========ffill================')
print(d1.fillna(method='ffill'))
print('===========backfill================')
print(d1.fillna(method='backfill'))
```

```
===========pad================
     Names  Ages     City
0   Ramesh  30.0      NaN
1   Suresh  31.0      Hyd
2  Sathish  31.0     Pune
3  Sathish  32.0  Chennai
===========bfill================
     Names  Ages     City
0   Ramesh  30.0      Hyd
1   Suresh  31.0      Hyd
2  Sathish  32.0     Pune
3      NaN  32.0  Chennai
===========ffill================
     Names  Ages     City
0   Ramesh  30.0      NaN
1   Suresh  31.0      Hyd
2  Sathish  31.0     Pune
3  Sathish  32.0  Chennai
===========backfill================
     Names  Ages     City
0   Ramesh  30.0      Hyd
1   Suresh  31.0      Hyd
2  Sathish  32.0     Pune
3      NaN  32.0  Chennai
```

- pad and ffill both are same
- bfill and backfill both are same

Categorical columns

- Categorical column mainly does not have any meaning in dataframes
- If it is some text or sentence you can find the meaning of the sentence
- For example Names column is there, Review column is there
- Becaues of only Name it self you can't judge the output
- But a Review can impact the output
- Generally if any naive names kind of columns we can fill in any way
- But if some columns impacting output, Yes we need to fill those columns

  with an idea about how that column is related with other coulumns

$Method - 5$

KNN imputer

KNN= K- nearest Neighbours

- It is One of ML model to find the nearest solution based distance metrics
- Instead of taking all the values of mean
- Why can't we take nehibours mean of the observation
- Here first will choose neighbours
- It will calculate mean of those neighbours and fill with that value

Package name: Sklean.imputer

Method name: KnnImputer

You need to explore this

- Test with Numerical column
- Test with categorical column+ Numerical

In [38]:
```
d1['Ages'].values
```

Out[38]:
```
array([30., 31., nan, 32.])
```

In [39]:
```
d1.values  # ===== KNNImputer
```

Out[39]:
```
array([['Ramesh', 30.0, nan],
       ['Suresh', 31.0, 'Hyd'],
       ['Sathish', nan, 'Pune'],
       [nan, 32.0, 'Chennai']], dtype=object)
```

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]: