ML 1 - Handling Missing Values By Virat Tiwari

November 7, 2023

1 Handling Missing Values - Today we understand that How to Handle Missing Values Data

[5]:
```
# We import seaborn as sns for getting the built in dataset

import seaborn as sns
```

[6]:
```
# sns.load_dataset ( ) function is used for importing the datset in seaborn
# We importing the " Titanic " becouse it contain lot of missing values
# All  isssing values are NaN values it means " Not A Number "

sns.load_dataset("titanic")
```

[6]:

	survived	pclass	sex	age	sibsp	parch	fare	embarked	class	\
0	0	3	male	22.0	1	0	7.2500	S	Third	
1	1	1	female	38.0	1	0	71.2833	C	First	
2	1	3	female	26.0	0	0	7.9250	S	Third	
3	1	1	female	35.0	1	0	53.1000	S	First	
4	0	3	male	35.0	0	0	8.0500	S	Third	
..	…	…	…	…	…	…	…	…	…	
886	0	2	male	27.0	0	0	13.0000	S	Second	
887	1	1	female	19.0	0	0	30.0000	S	First	
888	0	3	female	NaN	1	2	23.4500	S	Third	
889	1	1	male	26.0	0	0	30.0000	C	First	
890	0	3	male	32.0	0	0	7.7500	Q	Third	

	who	adult_male	deck	embark_town	alive	alone
0	man	True	NaN	Southampton	no	False
1	woman	False	C	Cherbourg	yes	False
2	woman	False	NaN	Southampton	yes	True
3	woman	False	C	Southampton	yes	False
4	man	True	NaN	Southampton	no	True
..	…	…	…	…	…	…
886	man	True	NaN	Southampton	no	True
887	woman	False	B	Southampton	yes	True
888	woman	False	NaN	Southampton	no	False
889	man	True	C	Cherbourg	yes	True

```
890    man        True  NaN   Queenstown    no   True

[891 rows x 15 columns]
```

[8]:
```
# df is our varible in which we store our dataset

df=sns.load_dataset("titanic")
```

[9]:
```
# .head  ( ) function gives the initil 5 data from the dataset
# All  isssing values are NaN values it means " Not A Number "

df.head()
```

[9]:
```
   survived  pclass     sex   age  sibsp  parch     fare embarked  class  \
0         0       3    male  22.0      1      0   7.2500        S  Third
1         1       1  female  38.0      1      0  71.2833        C  First
2         1       3  female  26.0      0      0   7.9250        S  Third
3         1       1  female  35.0      1      0  53.1000        S  First
4         0       3    male  35.0      0      0   8.0500        S  Third

     who  adult_male deck  embark_town alive  alone
0    man        True  NaN  Southampton    no  False
1  woman       False    C    Cherbourg   yes  False
2  woman       False  NaN  Southampton   yes   True
3  woman       False    C  Southampton   yes  False
4    man        True  NaN  Southampton    no   True
```

[11]:
```
# Check missing values in dataset
# isnull ( ) function gives the all missing values from the dataset
# False denotes the missing values
# True denotes the present values

df.isnull()
```

[11]:
```
     survived  pclass    sex    age  sibsp  parch   fare  embarked  class  \
0       False   False  False  False  False  False  False     False  False
1       False   False  False  False  False  False  False     False  False
2       False   False  False  False  False  False  False     False  False
3       False   False  False  False  False  False  False     False  False
4       False   False  False  False  False  False  False     False  False
..        ...     ...    ...    ...    ...    ...    ...       ...    ...
886     False   False  False  False  False  False  False     False  False
887     False   False  False  False  False  False  False     False  False
888     False   False  False   True  False  False  False     False  False
889     False   False  False  False  False  False  False     False  False
890     False   False  False  False  False  False  False     False  False
```

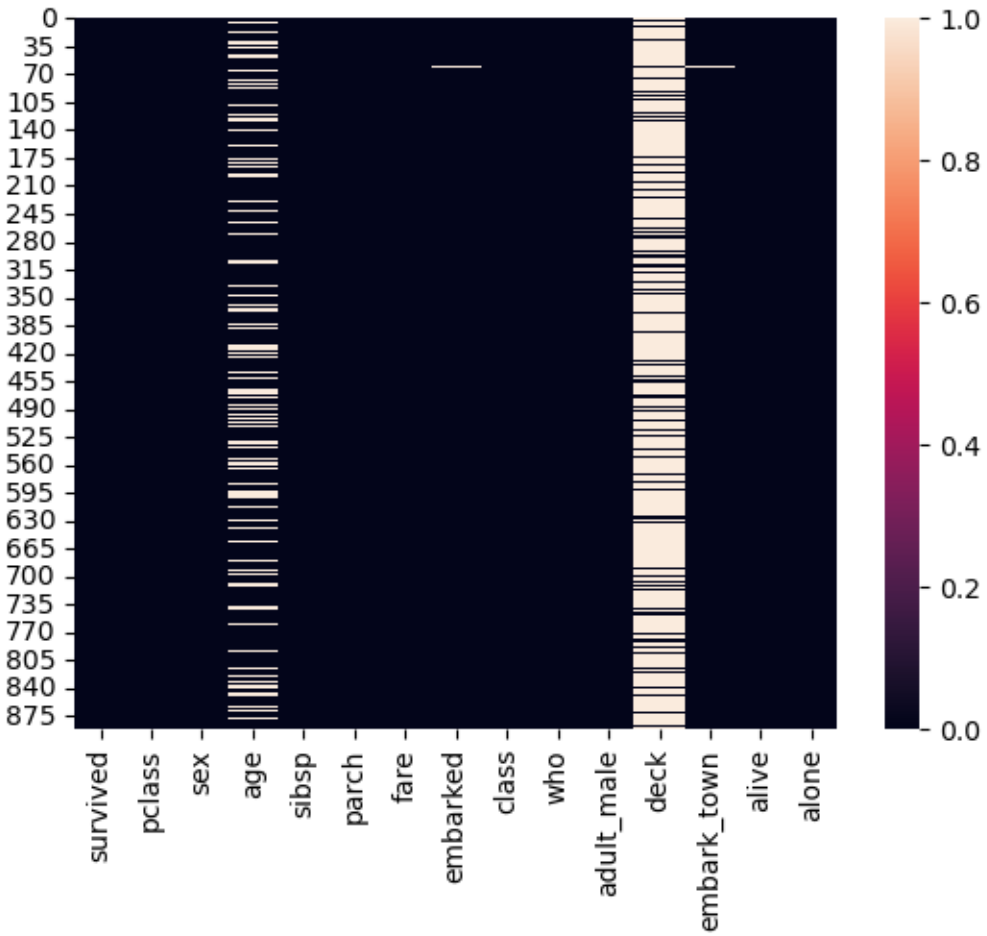
```
       who  adult_male   deck  embark_town  alive  alone
0    False       False   True        False  False  False
1    False       False  False        False  False  False
2    False       False   True        False  False  False
3    False       False  False        False  False  False
4    False       False   True        False  False  False
..     …           …     …           …      …      …
886  False       False   True        False  False  False
887  False       False  False        False  False  False
888  False       False   True        False  False  False
889  False       False  False        False  False  False
890  False       False   True        False  False  False

[891 rows x 15 columns]
```

[12]:
```
# .sum ( ) function is used for checking that how many missing values present␣
 ↪in every columns
# Ex - age have 177 missing values , deck have 688 and embark_town have 2␣
 ↪missing values

df.isnull().sum()
```

[12]:
```
survived         0
pclass           0
sex              0
age            177
sibsp            0
parch            0
fare             0
embarked         2
class            0
who              0
adult_male       0
deck           688
embark_town      2
alive            0
alone            0
dtype: int64
```

[14]:
```
# .heatmap ( ) function is used for visualize the missing values
# age and deck column shows the highest missing values in the form of bar code␣
 ↪and embarked & embarked_town also shows the small missing values␣
 ↪visualization

sns.heatmap(df.isnull())
```

[14]:
```
<AxesSubplot: >
```



[16]:
```
# All  isssing values are NaN values it means " Not A Number "


df.head()
```

[16]:

	survived	pclass	sex	age	sibsp	parch	fare	embarked	class	who	adult_male	deck	embark_town	alive	alone
0	0	3	male	22.0	1	0	7.2500	S	Third	man	True	NaN	Southampton	no	False
1	1	1	female	38.0	1	0	71.2833	C	First	woman	False	C	Cherbourg	yes	False
2	1	3	female	26.0	0	0	7.9250	S	Third	woman	False	NaN	Southampton	yes	True
3	1	1	female	35.0	1	0	53.1000	S	First	woman	False	C	Southampton	yes	False
4	0	3	male	35.0	0	0	8.0500	S	Third	man	True	NaN	Southampton	no	True

```
4    man        True  NaN  Southampton    no   True
```

[18]:
```
# CASE - 1

# Handling all values by deleting all the rows
# .dropna ( ) function is used for deleting all the Missing or NAN rows

df.dropna()
```

[18]:
```
     survived  pclass     sex   age  sibsp  parch     fare embarked  class  \
1           1       1  female  38.0      1      0  71.2833        C  First
3           1       1  female  35.0      1      0  53.1000        S  First
6           0       1    male  54.0      0      0  51.8625        S  First
10          1       3  female   4.0      1      1  16.7000        S  Third
11          1       1  female  58.0      0      0  26.5500        S  First
..        …     …      …   …    …    …       …      …    …
871         1       1  female  47.0      1      1  52.5542        S  First
872         0       1    male  33.0      0      0   5.0000        S  First
879         1       1  female  56.0      0      1  83.1583        C  First
887         1       1  female  19.0      0      0  30.0000        S  First
889         1       1    male  26.0      0      0  30.0000        C  First

       who  adult_male deck  embark_town alive  alone
1    woman       False    C    Cherbourg   yes  False
3    woman       False    C  Southampton   yes  False
6      man        True    E  Southampton    no   True
10   child       False    G  Southampton   yes  False
11   woman       False    C  Southampton   yes   True
..     …         …  …          …   …    …
871  woman       False    D  Southampton   yes  False
872    man        True    B  Southampton    no   True
879  woman       False    C    Cherbourg   yes  False
887  woman       False    B  Southampton   yes   True
889    man        True    C    Cherbourg   yes   True

[182 rows x 15 columns]
```

[22]:
```
# Before we have a 891 rows in which Missing Values are included
# We check no of rows and column by using the " .shape " function

df.shape
```

[22]:
```
(891, 15)
```

[20]:
```
# After applying .dropna ( ) function , it removes the all missing values rows␣
 ↪where NAN Values are present and after that present rows are 182
# We check no of rows and column by using the " .shape " function
```

```
df.dropna().shape
```

[20]:
```
(182, 15)
```

[23]:
```
#  CASE - 2

# Handling all values by deleting all the Columns
# .dropna ( axis = 1 ) function is used for deleting all the Missing or NAN rows

df.dropna(axis=1)
```

[23]:
```
     survived  pclass     sex  sibsp  parch     fare   class    who  \
0           0       3    male      1      0   7.2500   Third    man
1           1       1  female      1      0  71.2833   First  woman
2           1       3  female      0      0   7.9250   Third  woman
3           1       1  female      1      0  53.1000   First  woman
4           0       3    male      0      0   8.0500   Third    man
..        …     …      …    …       …      …    …
886         0       2    male      0      0  13.0000  Second    man
887         1       1  female      0      0  30.0000   First  woman
888         0       3  female      1      2  23.4500   Third  woman
889         1       1    male      0      0  30.0000   First    man
890         0       3    male      0      0   7.7500   Third    man

     adult_male alive  alone
0          True    no  False
1         False   yes  False
2         False   yes   True
3         False   yes  False
4          True    no   True
..          …   …    …
886        True    no   True
887       False   yes   True
888       False    no  False
889        True   yes   True
890        True    no   True

[891 rows x 11 columns]
```

2 Some other techniques that we have used for Handling The Missing data

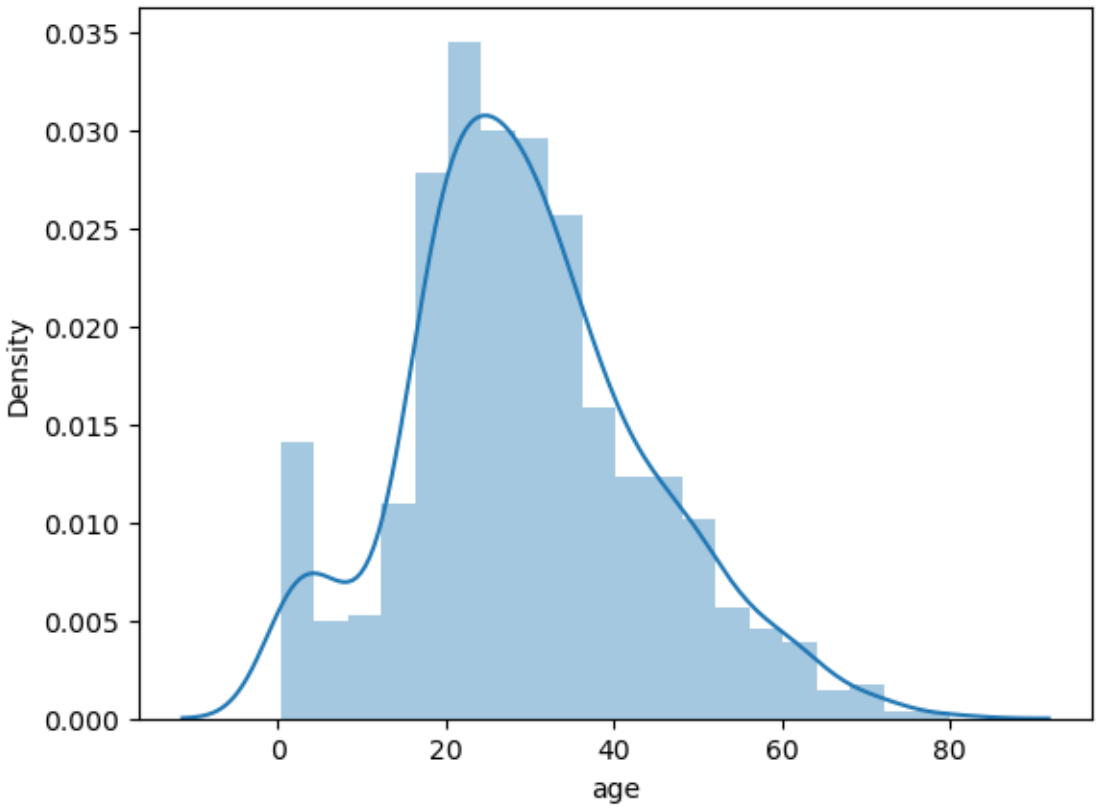
3 IMPUTATION TECHNIQUES :-

[25]:
```
# 1 - Mean Imputation Technique
```

[26]:
```
sns.distplot(df["age"])
```

```
/tmp/ipykernel_124/316555093.py:1: UserWarning:

`distplot` is a deprecated function and will be removed in seaborn v0.14.0.

Please adapt your code to use either `displot` (a figure-level function with
similar flexibility) or `histplot` (an axes-level function for histograms).

For a guide to updating your code to use the new functions, please see
https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751

  sns.distplot(df["age"])
```

[26]: <AxesSubplot: xlabel='age', ylabel='Density'>



[29]:
```
# This is how we check how many missing values no we have in particular column␣
 ↪like age , deck etc

df.age.isnull().sum()
```

[29]:
```
177
```

[30]:
```
# Mean Value imputation
# This techniques works well when data is normally distrubuted
# df["Age_mean"] - We assing the new variable for storing the mean of "age" and␣
 ↪adding that variable as a column in a dataset
# df["age"].fillna(df["age"].mean()) is used for getting the mean of age for␣
 ↪filling alll NAN missing places

df["Age_mean"]=df["age"].fillna(df["age"].mean())
```

[32]:
```
df[["Age_mean","age"]]
```

[32]:

	Age_mean	age
0	22.000000	22.0
1	38.000000	38.0
2	26.000000	26.0
3	35.000000	35.0
4	35.000000	35.0
..	…	…
886	27.000000	27.0
887	19.000000	19.0
888	29.699118	NaN
889	26.000000	26.0
890	32.000000	32.0

[891 rows x 2 columns]

[33]:
```
# 2 - Median Imputation Technique
```

[34]:
```
# We use MEDIAN becouse when we have skewed data where otliers are present so␣
 ↪we handle that data by using the MEDIAN IMPUTATION TECHNIQUE
# df["Age_median"] - We assing the new variable for storing the median of "age"␣
 ↪and adding that variable as a column in a dataset
# df["age"].fillna(df["age"].median()) is used for getting the median of age␣
 ↪for filling alll NAN missing places

df["Age_median"]=df["age"].fillna(df["age"].median())
```

[37]:
```
df[["Age_median","Age_mean","age"]]
```

[37]:
```
     Age_median   Age_mean   age
0          22.0  22.000000  22.0
1          38.0  38.000000  38.0
2          26.0  26.000000  26.0
3          35.0  35.000000  35.0
4          35.0  35.000000  35.0
..          …        …   …
886        27.0  27.000000  27.0
887        19.0  19.000000  19.0
888        28.0  29.699118   NaN
889        26.0  26.000000  26.0
890        32.0  32.000000  32.0

[891 rows x 3 columns]
```

[38]:
```
# 1 - Mode Imputation Technique
```

[39]:
```
# This is used for basically categorical values

df[df["embarked"].isnull()]
```

[39]:
```
     survived  pclass     sex   age  sibsp  parch  fare embarked  class  \
61          1       1  female  38.0      0      0  80.0      NaN  First
829         1       1  female  62.0      0      0  80.0      NaN  First

       who  adult_male deck embark_town alive  alone  Age_mean  Age_median
61   woman       False    B         NaN   yes   True      38.0        38.0
829  woman       False    B         NaN   yes   True      62.0        62.0
```

[42]:
```
df["embarked"].unique()
```

[42]:
```
array(['S', 'C', 'Q', nan], dtype=object)
```

[44]:
```
df["age"].notna()
```

[44]:
```
0       True
1       True
2       True
3       True
4       True
       …
886     True
887     True
888    False
889     True
890     True
Name: age, Length: 891, dtype: bool
```

[45]:
```
mode=df["age"].notna()
```

[46]:
```
df[df["age"].notna()]["embarked"].mode()[0]
```

[46]:
```
'S'
```

[47]:
```
mode=df[df["age"].notna()]["embarked"].mode()[0]
```

[48]:
```
mode
```

[48]:
```
'S'
```

[49]:
```
df["embarked_mode"]=df["embarked"].fillna(mode)
```

[52]:
```
df[["embarked_mode","embarked"]]
```

[52]:

	embarked_mode	embarked
0	S	S
1	C	C
2	S	S
3	S	S
4	S	S
..	…	…
886	S	S
887	S	S
888	S	S
889	C	C
890	Q	Q

[891 rows x 2 columns]

[53]:
```
df["embarked_mode"].isnull().sum()
```

[53]:
```
0
```

[54]:
```
df["embarked"].isnull().sum()
```

[54]:
```
2
```

THANK YOU SO MUCH !!

YOURS VIRAT TIWARI :)