Handiling Missing Values

August 7, 2023

1 Handiling missing values, handiling imbalaance imbalanace dataset , Smote, Data interpretation , percentiles and quartiles , number summary and box plots , handiling Outliers

1.1 Missing Values

Missing values occurs in dataset when some of the informations is not stored for a variable There are 3 mechanisms

1.1.1 1 Missing Completely at Random, MCAR:

Missing completely at random (MCAR) is a type of missing data mechanism in which the probability of a value being missing is unrelated to both the observed data and the missing data. In other words, if the data is MCAR, the missing values are randomly distributed throughout the dataset, and there is no systematic reason for why they are missing.

For example, in a survey about the prevalence of a certain disease, the missing data might be MCAR if the survey participants with missing values for certain questions were selected randomly and their missing responses are not related to their disease status or any other variables measured in the survey.

1.1.2 2. Missing at Random MAR:

Missing at Random (MAR) is a type of missing data mechanism in which the probability of a value being missing depends only on the observed data, but not on the missing data itself. In other words, if the data is MAR, the missing values are systematically related to the observed data, but not to the missing data. Here are a few examples of missing at random:

Income data: Suppose you are collecting income data from a group of people, but some participants choose not to report their income. If the decision to report or not report income is related to the participant's age or gender, but not to their income level, then the data is missing at random.

Medical data: Suppose you are collecting medical data on patients, including their blood pressure, but some patients do not report their blood pressure. If the patients who do not report their blood pressure are more likely to be younger or have healthier lifestyles, but the missingness is not related to their actual blood pressure values, then the data is missing at random.

1.2 3. Missing data not at random (MNAR)

It is a type of missing data mechanism where the probability of missing values depends on the value of the missing data itself. In other words, if the data is MNAR, the missingness is not random and is dependent on unobserved or unmeasured factors that are associated with the missing values.

For example, suppose you are collecting data on the income and job satisfaction of employees in a company. If employees who are less satisfied with their jobs are more likely to refuse to report their income, then the data is not missing at random. In this case, the missingness is dependent on job satisfaction, which is not directly observed or measured.

[2]:
```python
import seaborn as sns
```

[3]:
```python
df = sns.load_dataset('titanic')
```

[4]:
```python
df.head()
```

[4]:
```
   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
```

[5]:
```python
# check missing values
df.isnull()
```

[5]:
```
     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]
```

[7]:
```
df.isnull().sum() # in age column 177 datapoints are null and in deck columns␣
↪688 datpoint value are null, in embarked column 2 datapoints are null,
```

[7]:
```
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
```

[8]:
```
# one easiest way to probabily deleting the rows or datapoints to handiling␣
↪missing values . then what is the problem with this you  will loose huge␣
↪amount of data here
df.shape
```

[8]:
```
(891, 15)
```

[9]:
```
# i m just dropping it. i m dopping with respect to data points . whenever␣
↪there is a datapoint is missing . then drop the entire datapoints
df.dropna()
```

[9]:
```
     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]
```

[10]:
```python
# now if i will go and see the shape and you will able to see that how much datapoints it will be left only 182 datpoints are left . before 891 datpoints are and left datapoints are 183 .
df.dropna().shape
# this dropping feature is specifically doing bad loosing huge amount of data. it is not very good idea bout it to go
```

[10]:
```
(182, 15)
```

[11]:
```python
# can we do column wise because age column are having 177 datpoints are null values
# deck are having 688 datapoints are null values. so i can go and probabiliy delete the deck column. it is suitable way but age
# we can not drop as a column
df.dropna(axis=1)
```

[11]:
```
     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]
```

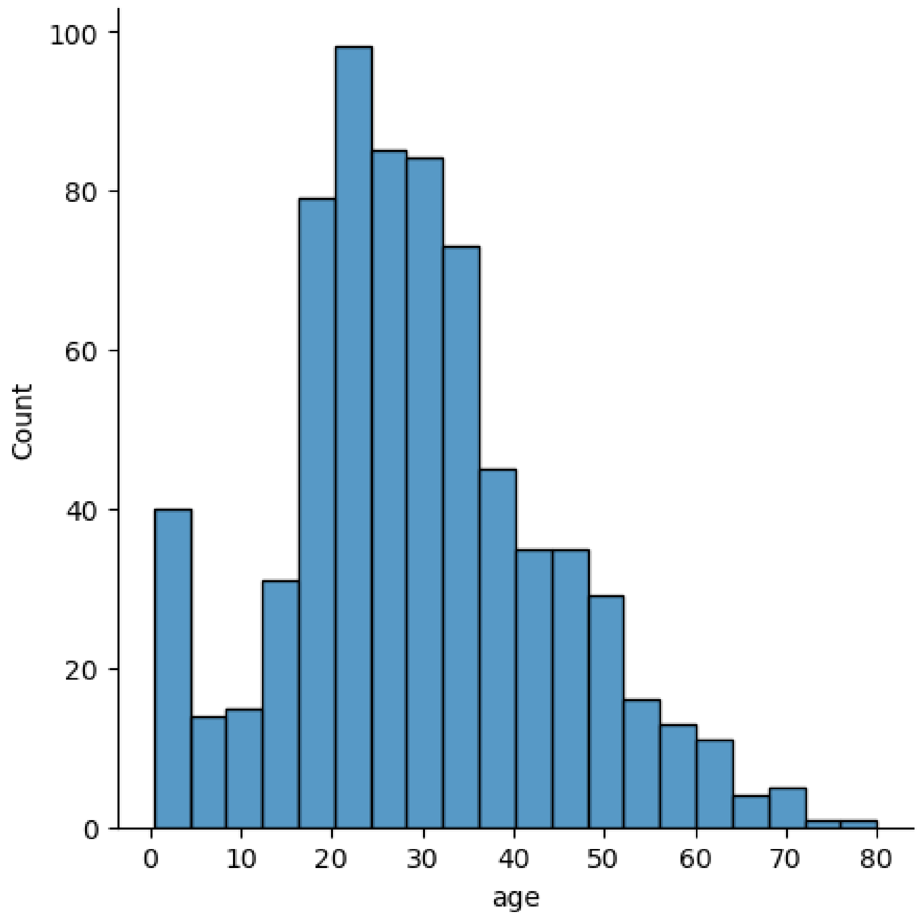
[12]:
```
# with respect to imputation techaniques . imputation missing techniques means how we can handle the missing values
```

2 Imputation Missing Values

2.1 1 - Mean Value Imputation

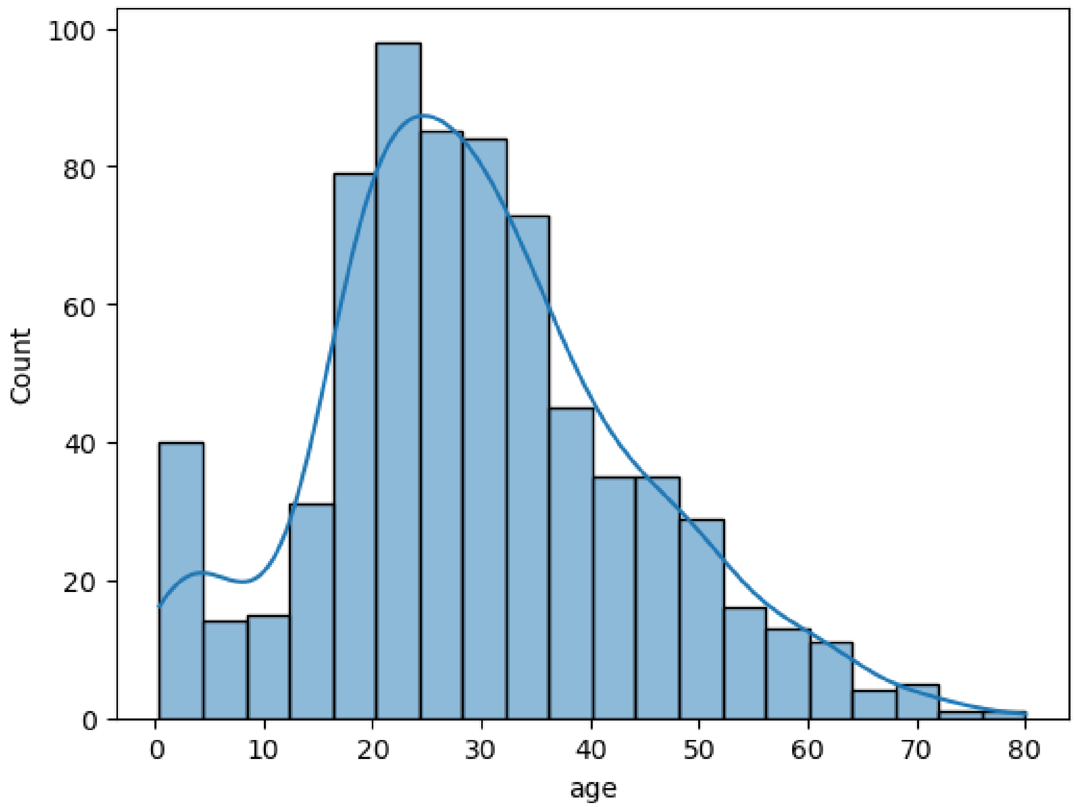
[13]:
```
# Mean value imputation is techniques which will be specifically will be using with respect to mean value imputation as name suggest mean value
# we are supposed to replace the missing values with the men of the value of the datapoints.
sns.displot(df['age'])
```

[13]: <seaborn.axisgrid.FacetGrid at 0x7fdaff1465c0>



[15]:
```
sns.histplot(df['age'],kde=True) # its look normal distribution some what
```

[15]: `<AxesSubplot: xlabel='age', ylabel='Count'>`




```
[17]: # in age column 177 datapoints are having null values can we handle this null
      ↪values by replacing with the help of mean values so for that
      # we can do that
      df['age'].fillna(df['age'].mean())
```

```
[17]: 0      22.000000
      1      38.000000
      2      26.000000
      3      35.000000
      4      35.000000
               …
      886    27.000000
      887    19.000000
      888    29.699118
      889    26.000000
      890    32.000000
      Name: age, Length: 891, dtype: float64
```

```
[19]: df['Age_mean']=df['age'].fillna(df['age'].mean())
```

[20]:
```
df[['Age_mean','age']] # here nan value is replaced by mean values in 888 row␣
↪NAN values is replaced by 29.699118 value its the mean value of the entrire␣
↪age column
```

[20]:
```
       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]
```

[22]:
```
# mean imputation work well when we have normally distributed data . then what␣
↪if we have differnt kind of distribution right skewed and left skewed so for␣
↪that case you know outlier is also there you can replace nan value with␣
↪median
```

2.2 2. Median Value Imputation - if we have outliers in the dataset

[23]:
```
df['Age_median']=df['age'].fillna(df['age'].median())
```

[24]:
```
df[['Age_median','Age_mean','age']]
```

[24]:
```
     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]
```

2.3 Mode Imputation Technqiue – Categorical values

[27]:
```
# we have a column is embraked we will probabily see
df[df['embarked'].isnull()]
# in embarked column we have two nan value embarked is an example of missing␣
↪completely at random
```

[27]:
```
     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_median  Age_mean
61   woman       False    B         NaN   yes   True        38.0      38.0
829  woman       False    B         NaN   yes   True        62.0      62.0
```

[28]:
```
# this is a categorical variable
df['embarked'].unique()
# over here we have nan value replace the nan value with the sum of values␣
↪which is probabilly like here categorical values
# which are probabliy find out from mode
```

[28]:
```
array(['S', 'C', 'Q', nan], dtype=object)
```

[29]:
```
df['embarked'].notna()
# notna is function give me all the output which will be basically saying that
# wherever there is a nan value try to gave me these values are false and␣
↪remaining values are true
```

[29]:
```
0      True
1      True
2      True
3      True
4      True
       …
886    True
887    True
888    True
889    True
890    True
Name: embarked, Length: 891, dtype: bool
```

[30]:
```
df[df['embarked'].notna()]['embarked'].mode()
```

[30]:
```
0    S
Name: embarked, dtype: object
```

[31]:
```
df[df['embarked'].notna()]['embarked'].mode()[0]
```

[31]: 'S'

[32]:
```
mode_value = df[df['embarked'].notna()]['embarked'].mode()[0]
```

[33]:
```
df['embarked_mode']=df['embarked'].fillna(mode_value)
```

[34]:
```
df[['embarked_mode','embarked']]
```

[34]:
```
    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]
```

[35]:
```
df['embarked_mode'].isnull().sum()
```

[35]: 0

[36]:
```
df['embarked'].isnull().sum()
```

[36]: 2