5.1-Handling_missing_values

July 15, 2025

0.1 Missing Values

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

0.1.1 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.

0.1.2 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.

0.2 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.

0.2.1 Implementation

[1]:
```python
import seaborn as sns
```

[2]:
```python
df = sns.load_dataset('titanic')
```

[3]:
```python
df.head()
```

[3]:
```
   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
```

[4]:
```python
## Check missing values
df.isnull().sum()
```

[4]:
```
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
```

[5]:
```python
## Delete the rows or data points to handle missing values
print(f" Shape before using drop function: {df.shape}")
```

```
## if we use the drop method for deleting the data in this case we have delete␣
↪the missing values but we loss the huge amount of data, Because the drop()␣
↪function is deleting all the rows if that row contains the any missing values

print(f" Shape after using drop function: {df.dropna().shape}")

## you can see that we have loss huge amount of data so we are not prefer this␣
↪techniques to handling any missing values
```

```
Shape before using drop function: (891, 15)
Shape after using drop function: (182, 15)
```

[6]:
```
## Column wise delete
df.dropna(axis=1)  # we use (axis=1) for column and (axis=0) for row

# This is drop the column which have missing values but in our case we have age␣
↪column that contains the missing values and also this age column is␣
↪important for our analysis so we are not going to use this method as well.
```

[6]:
```
     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]
```

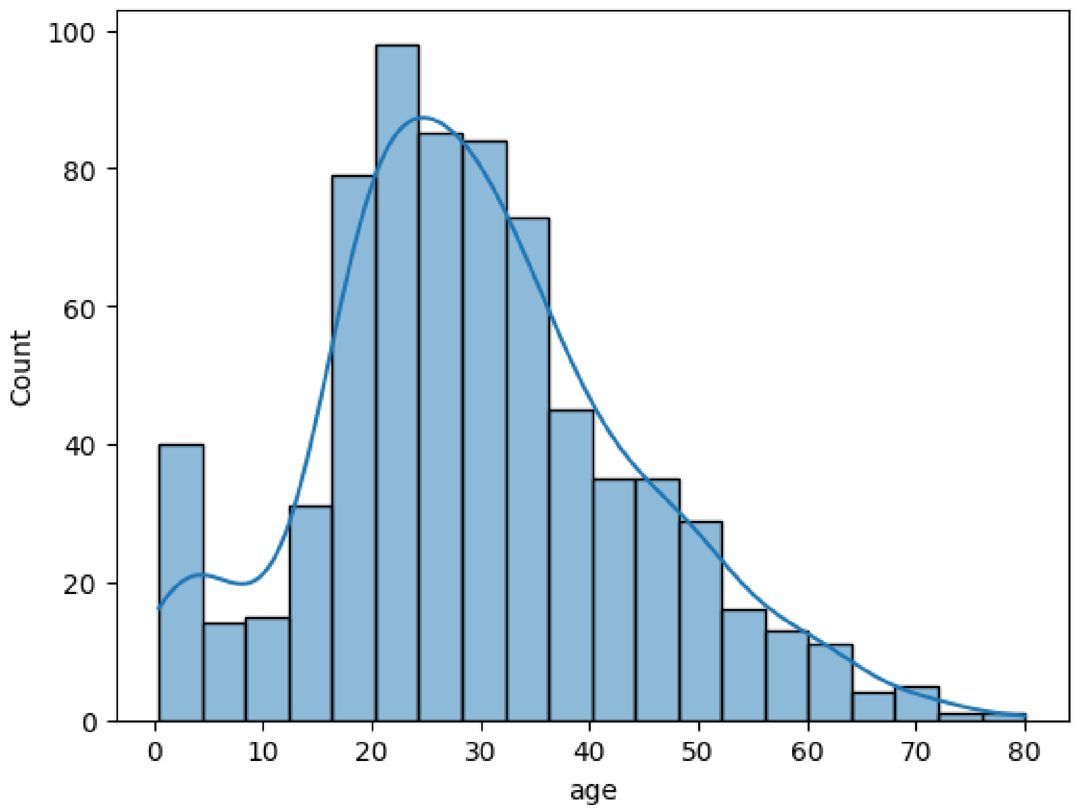
0.3 Imputation Missing Values

0.3.1 1- Mean Value Imputation:

Mean Imputation works well when we have normally distributed data

[7]:
```
sns.histplot(df['age'], kde=True)
```

[7]: <Axes: xlabel='age', ylabel='Count'>



[8]:
```
df['Age_mean']=df['age'].fillna(df['age'].mean())
```

[9]:
```
df[['Age_mean','age']]
```

[9]:

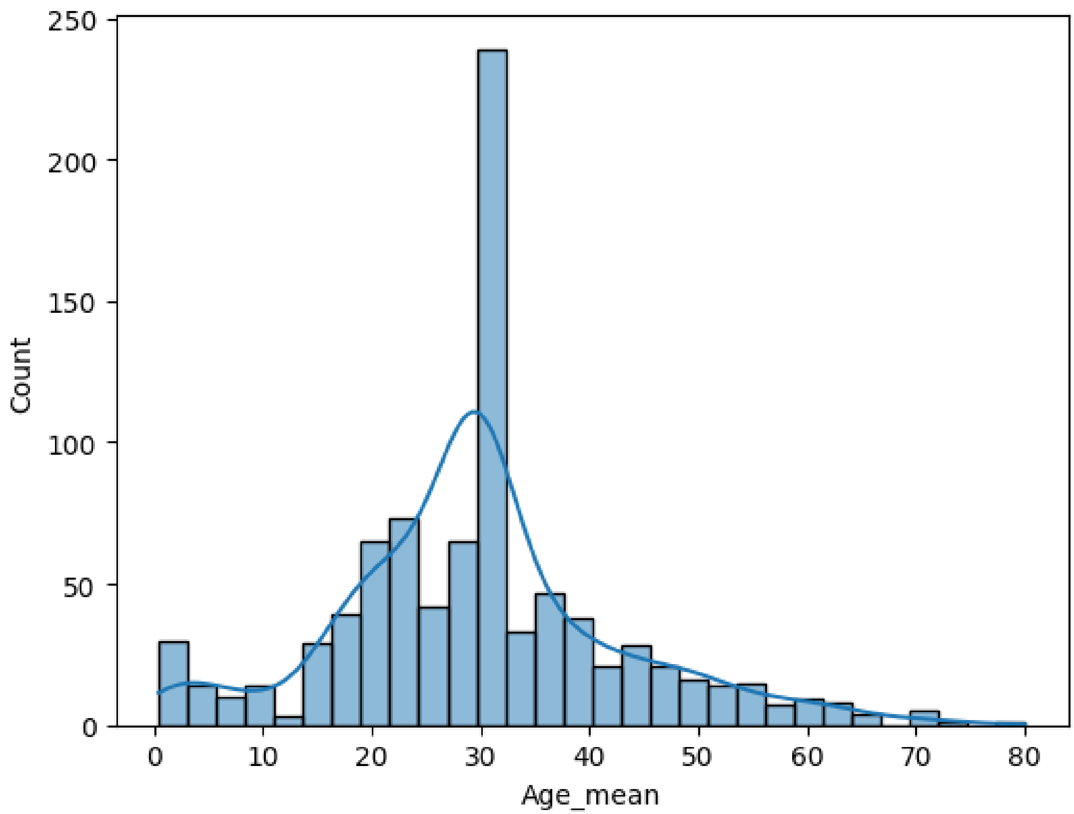
	Age_mean	age
0	22.000000	22.0
1	38.000000	38.0
2	26.000000	26.0
3	35.000000	35.0
4	35.000000	35.0
..	…	…
886	27.000000	27.0

```
887  19.000000  19.0
888  29.699118   NaN
889  26.000000  26.0
890  32.000000  32.0

[891 rows x 2 columns]
```

[10]:
```
sns.histplot(df['Age_mean'],kde=True)
```

[10]:
```
<Axes: xlabel='Age_mean', ylabel='Count'>
```



0.3.2 2- Median Value Imputation

    If we have outliers in the dataset

[11]:
```
df['Age_median']=  df['age'].fillna(df['age'].median())
```

[12]:
```
df[['Age_median', 'Age_mean', 'age']]
```

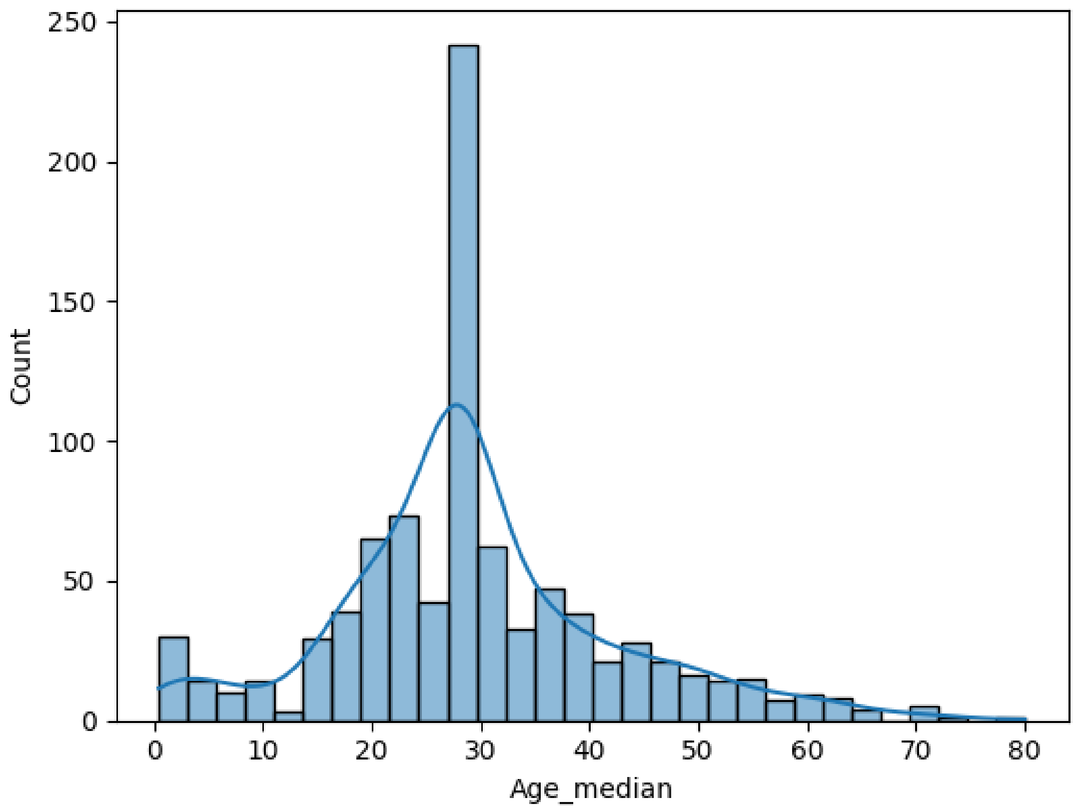
[12]:
```
     Age_median   Age_mean   age
0          22.0  22.000000  22.0
```

```
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]
```

[13]:
```python
sns.histplot(df['Age_median'], kde=True)
```

[13]: <Axes: xlabel='Age_median', ylabel='Count'>



0.3.3 3- Mode Value Imputation

```
If we have categorical values
```

[14]:
```
df[df['embarked'].isnull()]
```

[14]:
```
     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
```

[15]:
```
df['embarked'].unique()
```

[15]:
```
array(['S', 'C', 'Q', nan], dtype=object)
```

[16]:
```
df[df['embarked'].notna()]
```

[16]:
```
     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   Age_mean  Age_median
0      man        True  NaN  Southampton    no  False  22.000000        22.0
1    woman       False    C    Cherbourg   yes  False  38.000000        38.0
2    woman       False  NaN  Southampton   yes   True  26.000000        26.0
3    woman       False    C  Southampton   yes  False  35.000000        35.0
4      man        True  NaN  Southampton    no   True  35.000000        35.0
..     …           …    …          …     …      …          …
886    man        True  NaN  Southampton    no   True  27.000000        27.0
887  woman       False    B  Southampton   yes   True  19.000000        19.0
888  woman       False  NaN  Southampton    no  False  29.699118        28.0
889    man        True    C    Cherbourg   yes   True  26.000000        26.0
890    man        True  NaN   Queenstown    no   True  32.000000        32.0

[889 rows x 17 columns]
```

[17]:
```
## finding the mode on embarked column
mode_value=df[df['embarked'].notna()]['embarked'].mode()[0]
```

[18]:
```
df['embarked_mode']= df['embarked'].fillna(mode_value)
```

[19]:
```python
df[['embarked_mode', 'embarked']]
```

[19]:
```
    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]
```