# Data Wrangling

In [ ]:
```
# install libraries
#pip install seaborn
#pip install pandas
#pip install numpy
```

In [ ]:
```
# import libraries
import numpy as np
import pandas as pd
import seaborn as sns
```

In [ ]:
```
ship=sns.load_dataset('titanic')
sh1=ship
sh2=ship
sh=sns.load_dataset('titanic')
sh4=sns.load_dataset('titanic')
```

In [ ]:
```
ship.head()
```

Out[ ]:

| | survived | pclass | sex | age | sibsp | parch | fare | embarked | class | who | adult_male | deck |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **0** | 0 | 3 | male | 22.0 | 1 | 0 | 7.2500 | S | Third | man | True | NaN |
| **1** | 1 | 1 | female | 38.0 | 1 | 0 | 71.2833 | C | First | woman | False | C |
| **2** | 1 | 3 | female | 26.0 | 0 | 0 | 7.9250 | S | Third | woman | False | NaN |
| **3** | 1 | 1 | female | 35.0 | 1 | 0 | 53.1000 | S | First | woman | False | C |
| **4** | 0 | 3 | male | 35.0 | 0 | 0 | 8.0500 | S | Third | man | True | NaN |

In [ ]:
```
ship['age'].mean()
```

Out[ ]:
```
29.69911764705882
```

In [ ]:
```
# simple operation (math operation) along column/series
#(ship['age']+5).head()
```

# Dealing with missing values

- in a data missing values are either ? or N/A or 0 or blank cell.
- jab kbi data na ho kisi aik row main kisi b aik parameter ka ### Steps

1. Koshish kry dobara data collect kr ly agr khi ghlti hai.
2. missing value wala variable(column) hi nikal dy gr data pr effect nahi hota ya simple row or data entry remove kr dy.

3. Replace the missing values:

    A. How?
        a. Average value of entire variable similar data point
        b. Frequency or Mode replacement
        c. replace based on other fuctions
        d. ML algorithm can also be used
        e. leave it like that
    B. why?
        a. its better bcz no data is lost
        b. less accurate

In [ ]:
```
# where exactly missing values are?
ship.isnull().sum()
```

Out[ ]:
```
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
```

In [ ]:
```
# use drop.na method
#print(ship.shape)
#ship.dropna(subset=['deck'], axis=0, inplace=True)# this will remove specifically
# inplace = true modifies this dataframe
```

In [ ]:
```
#find  again null Value
#ship.isnull().sum()
```

In [ ]:
```
# to drop na
#ship=ship.dropna()
# to update main dataframe
#ship.isnull().sum()#remove na from whole dataframe
```

In [ ]:
```
#ship.shape
```

# Replace missing values with the average of that column

In [ ]:
```
# finding an average(mean)
mean = sh1['age'].mean()
```

```
mean
```

Out[ ]:
```
29.69911764705882
```

In [ ]:
```
# replacing nan with the mean of the data(updating as well)
sh1['age']=sh1['age'].replace(np.nan , mean)
```

In [ ]:
```
sh1.isnull().sum()
```

Out[ ]:
```
survived         0
pclass           0
sex              0
age              0
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
```

In [ ]:
```
sh1.head()
```

Out[ ]:

| | survived | pclass | sex | age | sibsp | parch | fare | embarked | class | who | adult_male | deck |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **0** | 0 | 3 | male | 22.0 | 1 | 0 | 7.2500 | S | Third | man | True | NaN |
| **1** | 1 | 1 | female | 38.0 | 1 | 0 | 71.2833 | C | First | woman | False | C |
| **2** | 1 | 3 | female | 26.0 | 0 | 0 | 7.9250 | S | Third | woman | False | NaN |
| **3** | 1 | 1 | female | 35.0 | 1 | 0 | 53.1000 | S | First | woman | False | C |
| **4** | 0 | 3 | male | 35.0 | 0 | 0 | 8.0500 | S | Third | man | True | NaN |

In [ ]:
```
#sh1=sh1.dropna()
#sh1.isnull().sum()
```

In [ ]:
```
#sh1.shape
```

# Data formating

- Data ko aik common standard pr lana
- Ensure data is consistant and understandable
  - Easy to gather
  - Easy to work with
    - Faisalabad(FSD)
    - Lahore(LHR)

- Islamabad(ISB)
- Convert g to kg or similar unit for all
- One standard unit for each column
- ft!=cm

In [ ]:
```
# know the data type and convert it into known one
ship.dtypes
```

Out[ ]:
```
survived          int64
pclass            int64
sex              object
age             float64
sibsp             int64
parch             int64
fare            float64
embarked         object
class          category
who              object
adult_male         bool
deck           category
embark_town      object
alive            object
alone              bool
dtype: object
```

In [ ]:
```
# use this method to convert datatype from one to another format
ship['survived']=ship['survived'].astype("int64")
ship.dtypes
```

Out[ ]:
```
survived          int64
pclass            int64
sex              object
age             float64
sibsp             int64
parch             int64
fare            float64
embarked         object
class          category
who              object
adult_male         bool
deck           category
embark_town      object
alive            object
alone              bool
dtype: object
```

In [ ]:
```
# convert age into days instead of years
sh1['age']=sh1['age']*365
sh1.head(6)
```

Out[ ]:

| | survived | pclass | sex | age | sibsp | parch | fare | embarked | class | who | adult_mal |
|---|---|---|---|---|---|---|---|---|---|---|---|
| **0** | 0 | 3 | male | 8030.000000 | 1 | 0 | 7.2500 | S | Third | man | Tru |
| **1** | 1 | 1 | female | 13870.000000 | 1 | 0 | 71.2833 | C | First | woman | Fals |
| **2** | 1 | 3 | female | 9490.000000 | 0 | 0 | 7.9250 | S | Third | woman | Fals |
| **3** | 1 | 1 | female | 12775.000000 | 1 | 0 | 53.1000 | S | First | woman | Fals |
| **4** | 0 | 3 | male | 12775.000000 | 0 | 0 | 8.0500 | S | Third | man | Tru |
| **5** | 0 | 3 | male | 10840.177941 | 0 | 0 | 8.4583 | Q | Third | man | Tru |

In [ ]:
```
# allways rename afterwards
sh1.rename(columns={"age": "age in days"}, inplace=True)
sh1.head()
```

Out[ ]:

| | survived | pclass | sex | age in days | sibsp | parch | fare | embarked | class | who | adult_male | de |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **0** | 0 | 3 | male | 8030.0 | 1 | 0 | 7.2500 | S | Third | man | True | N |
| **1** | 1 | 1 | female | 13870.0 | 1 | 0 | 71.2833 | C | First | woman | False | |
| **2** | 1 | 3 | female | 9490.0 | 0 | 0 | 7.9250 | S | Third | woman | False | N |
| **3** | 1 | 1 | female | 12775.0 | 1 | 0 | 53.1000 | S | First | woman | False | |
| **4** | 0 | 3 | male | 12775.0 | 0 | 0 | 8.0500 | S | Third | man | True | N |

# Data Normalization

- Uniform the data
- Making sure they have same impact
- zero to one range
- Also for computational reasons

In [ ]:
```
sh1.head()
```

Out[ ]:

| | survived | pclass | sex | age in days | sibsp | parch | fare | embarked | class | who | adult_male | de |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **0** | 0 | 3 | male | 8030.0 | 1 | 0 | 7.2500 | S | Third | man | True | N |
| **1** | 1 | 1 | female | 13870.0 | 1 | 0 | 71.2833 | C | First | woman | False | |
| **2** | 1 | 3 | female | 9490.0 | 0 | 0 | 7.9250 | S | Third | woman | False | N |
| **3** | 1 | 1 | female | 12775.0 | 1 | 0 | 53.1000 | S | First | woman | False | |
| **4** | 0 | 3 | male | 12775.0 | 0 | 0 | 8.0500 | S | Third | man | True | N |

In [ ]:
```
sh1= sh1[["age in days", "fare"]]
sh1.head(6)
```

Out[ ]:

| | age in days | fare |
|---|---|---|
| **0** | 8030.000000 | 7.2500 |
| **1** | 13870.000000 | 71.2833 |
| **2** | 9490.000000 | 7.9250 |
| **3** | 12775.000000 | 53.1000 |
| **4** | 12775.000000 | 8.0500 |
| **5** | 10840.177941 | 8.4583 |

- the above data is really in wide range and we need to normalize and hard to compare.
- normalization change bthe valuesto the range of 0-to-1(now both variables have similar influence on our models)

# Method of normalization

1. Simple feaured scaling
   - x(new)= x(old)/x(max)
2. Min-Max method
3. Z-score(standard score) -3 to +3
4. Log tranformation

In [ ]:
```
# simple featured scaling
sh['fare']= sh['fare']/sh['fare'].max()
sh.head()
```

Out[ ]:

| | survived | pclass | sex | age in days | sibsp | parch | fare | embarked | class | who | adult_male | d |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **0** | 0 | 3 | male | 8030.0 | 1 | 0 | 0.014151 | S | Third | man | True | I |
| **1** | 1 | 1 | female | 13870.0 | 1 | 0 | 0.139136 | C | First | woman | False | |
| **2** | 1 | 3 | female | 9490.0 | 0 | 0 | 0.015469 | S | Third | woman | False | I |
| **3** | 1 | 1 | female | 12775.0 | 1 | 0 | 0.103644 | S | First | woman | False | |
| **4** | 0 | 3 | male | 12775.0 | 0 | 0 | 0.015713 | S | Third | man | True | I |

In [ ]:
```
# min-max method
sh2['fare']= (sh2['fare']-sh2['fare'].min())/(sh2['fare'].max()-sh2['fare'].min())
sh2['fare'].head()
```

Out[ ]:
```
0    0.014151
1    0.139136
2    0.015469
3    0.103644
4    0.015713
Name: fare, dtype: float64
```

In [ ]:
```
# z-score method
sh['fare']= (sh['fare']-sh['fare'].mean())/sh['fare'].std()
sh.head()
```

Out[ ]:

| | survived | pclass | sex | age in days | sibsp | parch | fare | embarked | class | who | adult_male |
|---|---|---|---|---|---|---|---|---|---|---|---|
| **0** | 0 | 3 | male | 8030.0 | 1 | 0 | -0.502163 | S | Third | man | True |
| **1** | 1 | 1 | female | 13870.0 | 1 | 0 | 0.786404 | C | First | woman | False |
| **2** | 1 | 3 | female | 9490.0 | 0 | 0 | -0.488580 | S | Third | woman | False |
| **3** | 1 | 1 | female | 12775.0 | 1 | 0 | 0.420494 | S | First | woman | False |
| **4** | 0 | 3 | male | 12775.0 | 0 | 0 | -0.486064 | S | Third | man | True |

In [ ]:
```
sh4.head()
```

Out[ ]:

| | survived | pclass | sex | age | sibsp | parch | fare | embarked | class | who | adult_male | deck |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **0** | 0 | 3 | male | 22.0 | 1 | 0 | 7.2500 | S | Third | man | True | NaN |
| **1** | 1 | 1 | female | 38.0 | 1 | 0 | 71.2833 | C | First | woman | False | C |
| **2** | 1 | 3 | female | 26.0 | 0 | 0 | 7.9250 | S | Third | woman | False | NaN |
| **3** | 1 | 1 | female | 35.0 | 1 | 0 | 53.1000 | S | First | woman | False | C |
| **4** | 0 | 3 | male | 35.0 | 0 | 0 | 8.0500 | S | Third | man | True | NaN |

In [ ]:
```
sh2['fare']=np.log(sh2['fare'])
sh2.head()
```

```
c:\Users\m s\AppData\Local\Programs\Python\Python310\lib\site-packages\pandas\core\ar
raylike.py:397: RuntimeWarning: invalid value encountered in log
  result = getattr(ufunc, method)(*inputs, **kwargs)
```

Out[ ]:

| | survived | pclass | sex | age in days | sibsp | parch | fare | embarked | class | who | adult_male |
|---|---|---|---|---|---|---|---|---|---|---|---|
| **0** | 0 | 3 | male | 8030.0 | 1 | 0 | NaN | S | Third | man | True |
| **1** | 1 | 1 | female | 13870.0 | 1 | 0 | -0.240285 | C | First | woman | False |
| **2** | 1 | 3 | female | 9490.0 | 0 | 0 | NaN | S | Third | woman | False |
| **3** | 1 | 1 | female | 12775.0 | 1 | 0 | -0.866325 | S | First | woman | False |
| **4** | 0 | 3 | male | 12775.0 | 0 | 0 | NaN | S | Third | man | True |

# Binning

- Grouping of values into smaller number of values(bins)
- convert numaric into categories(jawan, bachy, boorhy) or1-16,17-30 etc.
- to have better understanding of groups
  - low vs mid vs high price

In [ ]:
```
bins = np.linspace(min(sh['age']), max(sh['age']), 15000)
age_groups=['childeren', 'young', 'old']
sh['age']= pd.cut(sh['age'], bins= [0, 15, 40, 100], labels=age_groups, include_lowest
sh['age']
```

Out[ ]:
```
0      young
1      young
2      young
3      young
4      young
       ...
886    young
887    young
888      NaN
889    young
890    young
Name: age, Length: 891, dtype: category
Categories (3, object): ['childeren' < 'young' < 'old']
```

## Converting categories into dumies

- easy to use for computation
- male female(0, 1)

In [ ]:
```
pd.get_dummies(sh1['sex']).head()
```

Out[ ]:

| | female | male |
|---|---|---|
| **0** | 0 | 1 |
| **1** | 1 | 0 |
| **2** | 1 | 0 |
| **3** | 1 | 0 |
| **4** | 0 | 1 |

In [ ]:
```
pd.get_dummies(sh1, columns=['sex']).head()
```

Out[ ]:

| | survived | pclass | age | sibsp | parch | fare | embarked | class | who | adult_male | deck | embarl |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **0** | 0 | 3 | 22.0 | 1 | 0 | 7.2500 | S | Third | man | True | NaN | Southa |
| **1** | 1 | 1 | 38.0 | 1 | 0 | 71.2833 | C | First | woman | False | C | Che |
| **2** | 1 | 3 | 26.0 | 0 | 0 | 7.9250 | S | Third | woman | False | NaN | Southa |
| **3** | 1 | 1 | 35.0 | 1 | 0 | 53.1000 | S | First | woman | False | C | Southa |
| **4** | 0 | 3 | 35.0 | 0 | 0 | 8.0500 | S | Third | man | True | NaN | Southa |

In [ ]:
```
#sh1.head()
# how to append/how to use get dumies to change data include a data frame(Assignment)/
```

Out[ ]:

| | survived | pclass | sex | age | sibsp | parch | fare | embarked | class | who | adult_male | deck |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **0** | 0 | 3 | male | 22.0 | 1 | 0 | 7.2500 | S | Third | man | True | NaN |
| **1** | 1 | 1 | female | 38.0 | 1 | 0 | 71.2833 | C | First | woman | False | C |
| **2** | 1 | 3 | female | 26.0 | 0 | 0 | 7.9250 | S | Third | woman | False | NaN |
| **3** | 1 | 1 | female | 35.0 | 1 | 0 | 53.1000 | S | First | woman | False | C |
| **4** | 0 | 3 | male | 35.0 | 0 | 0 | 8.0500 | S | Third | man | True | NaN |

In [ ]: