Data Wrangling

In [ ]:
```
# First Install Libraries

# pip install padas
# pip install seaborn
# pip install numpy
```

In [ ]:
```
# Second Import Libraries

import pandas as pd
import numpy as np
import seaborn as sns
```

In [ ]:
```
# Third Load Titanic (Kashti) data Set

kashti = sns.load_dataset('titanic')
ks1 = kashti
ks2 = kashti
ks3 = kashti
ks4 = kashti
ks5 = kashti
```

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

Now Data Wrangling Procedure Start

In [ ]:
```
# Simple Operations (Maths Operator)
# By the help of below command we can add or substract or divid or multiply any nume

(kashti['age']+12).head()
```

Dealing With Missing Values

1. In a Data set missing values are either? or N/A or NAN (Not a Number) or 0 or a blank cell.
2. When ever missing value found in a row in any parameter just follow below steps.

> Steps :

1. Try to collect data again if you found any missing data
2. Remove missing value variable (column) if its not effect in you data or simple remove row
3. Replace the missing Value.
    1. How?
        1. Average value of entire variable or similar data point
        2. frequency or MODE replacement (but its use very rare)

3. Replace based on other functions (Data Sampler knows that)
4. ML algorithm can also be used
5. Leave it like that

2. Why?
   1. Its better becuase no data lost.
   2. Less accurate

In [ ]:
```
# How to find Where is exactrly missing values are in our dataset?

kashti.isnull().sum()
```

In [ ]:
```
# By using shape command we will find how many row and column in our data sent
kashti.shape
```

In [ ]:
```
# By Describle command we will find Data detail like min or max value / mean / row c
kashti.describe()
```

In [ ]:
```
# Again check missing values are in our dataset?

kashti.isnull().sum()
```

First Method to deal with missing value to delete Null values from Dataset

In [ ]:
```
# How to find Where is exactrly missing values are in our dataset?

kashti.isnull().sum()

# Now Null Values from Deck column is delete but you can find age values also delelt
```

In [ ]:
```
# Use drop.na method to delete all null values from your dataset

kashti=kashti.dropna()
```

In [ ]:
```
# To check your dataset all Null values are deleted use below command
kashti.isnull().sum()
```

In [ ]:
```
# To check your remaining Data befor it was (891,15) after deleting null value we lo
kashti.shape
```

Second Method to deal with missing values To replace missing values with average of that column data

In [ ]:
```
# Now we use KS1 DATA SET
```

```
ks1.head()
```

In [ ]:
```
ks1.isnull().sum()
```

In [ ]:
```
# First to find mean of data Column you want to replace values

mean = ks1['age'].mean()
mean
```

In [ ]:
```
# Replace Null values with mean of the data (updating as well)

ks1['age'] = ks1['age'].replace(np.nan, 29)
```

In [ ]:
```
# To check again Null Value in dataset

ks1.isnull().sum()
```

In [ ]:
```
ks1.head()
```

In [ ]:
```
ks1.dtypes
```

In [ ]:
```
ks1.deck.unique()
```

In [ ]:
```
# Find mode of deck column in dataset

mode = ks1['deck'].mode()
mode
```

In [ ]:
```
# Replace Null values in deck column in dataset by mode
ks1['deck'].fillna('C', inplace=True)
```

In [ ]:
```
ks1.head(20)
```

In [ ]:
```
ks1.isnull().sum()
```

In [ ]:
```
ks1 = ks1.dropna()
```

In [ ]:
```
ks1.isnull().sum()
```

Our Data is Clean Now

Next Step is Data Formating

- First we have to make data in standarize form.
- Ensure data is cositent and understandable.
  - Easy to Gather
  - Easy to Work

Example Of Standarize Data :

- We can use variable like karachi or KHI both can not work its difficult to get result 100%.
- Always use same units like eigher in Grams or Kilogram CM / Meter both units do not work.
- Always use one standard unit in each column
- FT != CM

After Normalization of Data we called this data Human Readable data size or Machine Readable Data Size

In [ ]:
```
# How to find Data type and convert it into the known one

kashti.dtypes
```

In [ ]:
```
# Use below commad to convert datatype from one to an other format you required

kashti['age'] = kashti['age'].astype('int64')
kashti.dtypes
```

In [ ]:
```
# Use this method to convert value of any column as you required
# Here we will convert the age column into days instead of years

ks1['age'] = ks1['age']*365
ks1.head()
```

In [ ]:
```
# Always rename to change value in column
ks1.rename(columns={'age':'age in days'}, inplace=True)
ks1.head()
```

In [ ]:
```
# After Convert Age in days our data show figure in decimal figure to remove it use

ks1['age in days'] = ks1['age in days'].astype('int64')
ks1.head()
```

Next Step is to Normalize Data

- Uniform The Data
- They have Same Impact
- For Comparission the data normalization is nessassary
- Also for computational reasons

In [ ]:
```
ks2=ks1
ks2.head()
```

In [ ]:
```
ks2=ks2[['age in days' , 'fare']]
ks2.head()
```

- The above data is really in wide range and we need to normalize and hard to compare without normalization
- Normalization change the values to the range -1 to 0 to 1 (now both variable has similar influence on our model)

Method of Normalization

1. Simple Feature Scaling
   - x(new) = x(old) / x(max)
2. Min - Max Method
3. Z - Score (Standard Score) -3 to +3
4. Log Transformation

In [ ]:
```
# When we compare Age in Days and Fare data we found they have to much difference in
# on this data we will found too much diviation so we have to normalize data.

# First Use Simple Feature Scaling Method

ks2['age in days'] = ks2['age in days'] / ks2['age in days'].max()

ks2['fare'] = ks2['fare'] / ks2['fare'].max()

ks2.head()
```

In [ ]:
```
# Scond Method to Normalize Data is Min - Max Method
ks3=ks1
ks3.head()
```

In [ ]:
```
ks3=ks3[['age in days' , 'fare']]
ks3.head()
```

In [ ]:
```
ks3_min=ks3['age in days'].min()
ks3_max=ks3['age in days'].max()
ks3_min
```

In [ ]:
```
ks3['age in days'] = (ks3['age in days']-ks3['age in days'].min()) / (ks3['age in da

ks3['fare'] = (ks3['fare']-ks3['fare'].min()) / (ks3['fare'].max()-ks3['fare'].min()
ks3.head()
```

In [ ]:
```
ks1.head()
```

In [ ]:
```
ks2.head()
```

In [ ]:
```
ks3.head()
```

In [ ]:
```
# Third Method Z - Score Method To Normalize Data

ks4=ks1
ks4.head()
```

In [ ]:
```
ks4=ks4[['age in days' , 'fare']]
ks4.head()
```

In [ ]:
```
# Z - Score Method

ks4['age in days'] = (ks4['age in days']-ks4['age in days'].mean()) / ks4['age in da


ks4['fare'] = (ks4['fare']-ks4['fare'].mean()) / ks4['fare'].std()

ks4.head()
```

In [ ]:
```
# Fourth Method to Normalize Data

ks5=ks1
ks5.head()
```

In [ ]:
```
ks5=ks5[['age in days' , 'fare']]
ks5.head()
```

In [ ]:
```
# Log Tranformaiton Method

ks5['fare'] = np.log(ks5['fare'])

ks5['age in days'] = np.log(ks5['age in days'])

ks5.head()
```

Binning

- Grouping of Values into smaller number of values (Bins)
- Convert numeric into categories (Teenage , Young, Old) like 0 to 16 , 17 - 30 and so on.
- To have better understanding of groups.
  - Low vs mid vs high price

In [ ]:
```
bins = np.linspace(min(ks1['age in days']),max(ks1['age in days']),15000)
age_groups = ['teenage', 'young', 'old']
ks1['age in days'] = pd.cut(ks1['age in days'],bins=3, labels=age_groups, include_lo
ks1['age in days']
```

In [ ]:
```
ks1.head(30)
```

In [ ]: