Data Pre-processing

1. Data
2. Types of Attributes
3. Preprocessing
4. Transformation
5. Measures
6. Visualization

Importing libs

```
# importing libraries
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
```

```
# from google.colab import files
# uploaded = files.upload()
# Data.csv

import os
os.chdir(r'C:\Users\surya\Downloads\PG-DBDA-Mar23\Datasets')
os.getcwd()
```

```
'C:\\Users\\surya\\Downloads\\PG-DBDA-Mar23\\Datasets'
```

Importing Dataset

```
# importing the dataset
dataset = pd.read_csv('Data.csv')
# response / classifier / dependent variable (Y) column is 'Purchased'
# age & salary - discrete
dataset.head()
```

	Country	Age	Salary	Purchased
0	France	44.0	72000.0	No
1	Spain	27.0	48000.0	Yes
2	Germany	30.0	54000.0	No
3	Spain	38.0	61000.0	No

```
1 dataset.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10 entries, 0 to 9
Data columns (total 4 columns):
 #   Column     Non-Null Count  Dtype  
---  ------     --------------  -----  
 0   Country    10 non-null     object 
 1   Age        9 non-null      float64
 2   Salary     9 non-null      float64
 3   Purchased  10 non-null     object 
dtypes: float64(2), object(2)
memory usage: 448.0+ bytes
```

```
1 dataset.describe()
```

	Age	Salary
count	9.000000	9.000000
mean	38.777778	63777.777778
std	7.693793	12265.579662
min	27.000000	48000.000000
25%	35.000000	54000.000000
50%	38.000000	61000.000000
75%	44.000000	72000.000000
max	50.000000	83000.000000

```
1 dataset.iloc[ : , :-1]
```

	Country	Age	Salary
0	France	44.0	72000.0
1	Spain	27.0	48000.0
2	Germany	30.0	54000.0
3	Spain	38.0	61000.0
4	Germany	40.0	NaN
5	France	35.0	58000.0
6	Spain	NaN	52000.0

identify X & Y

- Classify dataset as Dependent & Independent Variables

```
x = dataset.iloc[ : , :-1].values
# x is independent variables
x[:5]
```

```
array([['France', 44.0, 72000.0],
       ['Spain', 27.0, 48000.0],
       ['Germany', 30.0, 54000.0],
       ['Spain', 38.0, 61000.0],
       ['Germany', 40.0, nan]], dtype=object)
```

```
y = dataset.iloc[ : , -1].values
# y is dependent variables
y[:5]
```

```
array(['No', 'Yes', 'No', 'No', 'Yes'], dtype=object)
```

Imputation

- Handling Missing Values by substituting them with appropriate values

create SimpleImputer object

```
from sklearn.impute import SimpleImputer
```

```
imputer = SimpleImputer(missing_values=np.nan, strategy='mean')
```

fitting on SimpleImputer object

```
imputer.fit(x[:, 1:3])
# calculates the values
```

SimpleImputer

SimpleImputer()

transforming SimpleImputer object

```
x[ : , 1:3] = imputer.transform(x[ : , 1:3])
# transforms / applies those calculated values
x[:5]
```

```
array([['France', 44.0, 72000.0],
       ['Spain', 27.0, 48000.0],
       ['Germany', 30.0, 54000.0],
       ['Spain', 38.0, 61000.0],
       ['Germany', 40.0, 63777.77777777778]], dtype=object)
```

```
y[:5]
```

```
array(['No', 'Yes', 'No', 'No', 'Yes'], dtype=object)
```

Splitting

- doing 4-way split
- splitting the data into the training dataset and testing dataset

import train_test_split

```
from sklearn.model_selection import train_test_split
```

```
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.2, random_state=0)
# train_test_split(x, y, test_size=testing_dataset_percentage, random_state=0)
# randon_state = 0 means it will not select  rows randomly
```

```
x_train[:5]
```

```
array([['Germany', 40.0, 63777.77777777778],
       ['France', 37.0, 67000.0],
       ['Spain', 27.0, 48000.0],
       ['Spain', 38.77777777777778, 52000.0],
       ['France', 48.0, 79000.0]], dtype=object)
```

```
x_test[:5]
```

```
array([['Germany', 30.0, 54000.0],
       ['Germany', 50.0, 83000.0]], dtype=object)
```

```
y_train[:5]
```

```
array(['Yes', 'Yes', 'Yes', 'No', 'Yes'], dtype=object)
```

```
y_test[:5]
```

```
array(['No', 'No'], dtype=object)
```

Transforming Categorical Data

- assigning numerical representation to categorical values

1. LabelEncoder

creating LabelEncoder object

```
from sklearn.preprocessing import LabelEncoder
```

```
labelencoder = LabelEncoder()
# creating encoder for converting categorical data into numerical data
```

fitting & transforming LabelEncoder

```
x[:, 0] = labelencoder.fit_transform(x[:, 0])
# calculates the values (fit) & transforms (transform) them in one step
```

```
x[:5]
```

```
array([[0, 44.0, 72000.0],
       [2, 27.0, 48000.0],
       [1, 30.0, 54000.0],
       [2, 38.0, 61000.0],
       [1, 40.0, 63777.77777777778]], dtype=object)
```

2. OneHotEncoder & ColumnTransformer

```
from sklearn.preprocessing import OneHotEncoder
```

```
from sklearn.compose import ColumnTransformer
```

creating OneHotEncoder object & ColumnTransformer object

```
columntransformer = ColumnTransformer([('encoder', OneHotEncoder(), [0])])
# ColumnTransformer([('name', transformer(), [columns])])
# creating OneHotEncoding on 0th column
```

fitting & transforming ColumnTransformer object

```
x1 = np.array(columntransformer.fit_transform(x), dtype=np.str_)
# calculates the values (fit) & transforms (transform) them in one step
# converting to adjacency matrix
x1[:5]
```

```
array([['1.0', '0.0', '0.0'],
       ['0.0', '0.0', '1.0'],
       ['0.0', '1.0', '0.0'],
       ['0.0', '0.0', '1.0'],
       ['0.0', '1.0', '0.0']], dtype='<U32')
```

Feature scaling

StandardScaler

- creating StandardScaler object

```
from sklearn.preprocessing import StandardScaler
```

```
sc_x = StandardScaler()
```

fitting & transforming StandardScaler object

```
x1 = sc_x.fit_transform(x1)
# x_test = sc_x.fit_transform(x_test)
x1
```

```
array([[ 1.22474487, -0.65465367, -0.65465367],
       [-0.81649658, -0.65465367,  1.52752523],
       [-0.81649658,  1.52752523, -0.65465367],
       [-0.81649658, -0.65465367,  1.52752523],
       [-0.81649658,  1.52752523, -0.65465367],
       [ 1.22474487, -0.65465367, -0.65465367],
       [-0.81649658, -0.65465367,  1.52752523],
       [ 1.22474487, -0.65465367, -0.65465367],
       [-0.81649658,  1.52752523, -0.65465367],
       [ 1.22474487, -0.65465367, -0.65465367]])
```

HW: interview

1. What is data?
2. What is information?
3. What is raw data?
4. What is data set?
5. Why do we need preprocessing of data?
6. What are major tasks in data preprocessing?
7. Explain what is noise, with an example?
8. Explain the strategy to handle noisy data.
9. What do you mean by missing values?
10. How do you handle missing data? Mention the tools for data preprocessing
11. Explain the meaning of term data cleaning with an example

12. What is data preprocessing?
13. What preprocessing steps can be implemented to maintain data quality?
14. What is the difference between Data Preprocessing and Data Mining?
15. What is the differenc between Feature Engineering and Feature Engineering?

HW: titanic DataSet EDA

```
1
```