

## Group A Assignment No: 1

### Contents for Theory:

1. Introduction to Dataset
2. Python Libraries for Data Science
3. Description of Dataset
4. Panda Dataframe functions for load the dataset
5. Panda functions for Data Preprocessing
6. Panda functions for Data Formatting and Normalisation
7. Panda Functions for handling categorical variables

### 1. Introduction to Dataset :

A dataset is a collection of records, similar to a relational database table. Records are similar to table rows, but the columns can contain not only strings or numbers, but also nested data structures such as lists, maps, and other records.

| x             | y             | z            | class |
|---------------|---------------|--------------|-------|
| 0.5351795492  | 0.9443102776  | 0.1582435145 | 1     |
| 0.2372136153  | 0.6406416746  | 0.2375491596 | 1     |
| 0.9115356348  | 0.3311024322  | 0.5615073269 | 0     |
| 0.5634070287  | 0.4183148035  | 0.151904445  | 0     |
| 0.3728975195  | 0.3816657621  | 0.616341473  | 1     |
| 0.6783527289  | 0.938524515   | 0.5269012505 | 1     |
| 0.09568660734 | 0.04465749689 | 0.0133451798 | 0     |
| 0.2173318229  | 0.6170559076  | 0.3122273853 | 1     |
| 0.818890594   | 0.7459451367  | 0.9026713492 | 0     |
| 0.6064854042  | 0.5945985792  | 0.2188024961 | 0     |
| 0.1546966824  | 0.1579937453  | 0.1333579164 | 0     |

**Instance:** A single row of data is called an instance. It is an observation from the domain.

**Feature:** A single column of data is called a feature. It is a component of an observation and is also called an attribute of a data instance. Some features may be inputs to a model (the predictors) and others may be outputs or the features to be predicted.

**Data Type:** Features have a data type. They may be real or integer-valued or may have a categorical or ordinal value. You can have strings, dates, times, and more complex types, but typically they are reduced to real or categorical values when working with traditional machine learning methods.

**Datasets:** A collection of instances is a dataset and when working with machine learning methods we typically need a few datasets for different purposes.

**Training Dataset:** A dataset that we feed into our machine learning algorithm to train our model.

**Testing Dataset:** A dataset that we use to validate the accuracy of our model but is not used to train the model. It may be called the validation dataset.

### Data Represented in a Table:

Data should be arranged in a two-dimensional space made up of rows and columns. This type of data structure makes it easy to understand the data and pinpoint any problems. An example of some raw data stored as a CSV (comma separated values).

```
1., Avatar, 18-12-2009, 7.8
2., Titanic, 18-11-1997,
3., Avengers Infinity War, 27-04-2018, 8.5
```

The representation of the same data in a table is as follows:

| S.No | Movie                 | Release Date | Ratings (IMDb) |
|------|-----------------------|--------------|----------------|
| 1.   | Avatar                | 18-12-2009   | 7.8            |
| 2.   | Titanic               | 18-11-1997   | Na             |
| 3.   | Avengers Infinity War | 27-04-2018   | 8.5            |

### Pandas Data Types :

A data type is essentially an internal construct that a programming language uses to understand how to store and manipulate data.

A possible confusing point about pandas data types is that there is some overlap between pandas, python and numpy. This table summarizes the key points:

| Pandas dtype  | Python type  | NumPy type   | Usage  |
|---------------|--------------|--|--|
| object        | str or mixed | string_, unicode_, mixed types                                 | Text or mixed numeric and non-numeric values |
| int64         | int          | int_, int8, int16, int32, int64, uint8, uint16, uint32, uint64 | Integer numbers                              |
| float64       | float        | float_, float16, float32, float64                              | Floating point numbers                       |
| bool          | bool         | bool_  | True/False values                            |
| datetime64    | NA           | datetime64[ns]   | Date and time values                         |
| timedelta[ns] | NA           | NA   | Differences between two datetimes            |
| category      | NA           | NA   | Finite list of text values                   |

## 2. Python Libraries for Data Science :

### a. Pandas :

Pandas is an open-source Python package that provides high-performance, easy-to-use data structures and data analysis tools for the labeled data in Python programming language.

#### What can you do with Pandas?

1. Indexing, manipulating, renaming, sorting, merging data frame
2. Update, Add, Delete columns from a data frame
3. Impute missing files, handle missing data or NaNs
4. Plot data with histogram or box plot

**b. NumPy :** One of the most fundamental packages in Python, NumPy is a general-purpose array-processing package.

It provides high-performance multidimensional array objects

and tools to work with the arrays. NumPy is an efficient container of generic multi-dimensional data.

NumPy's main object is the homogeneous multidimensional array. It is a table of elements or numbers of the same datatype, indexed by a tuple of positive integers. In NumPy, dimensions are called axes and the number of axes is called rank. NumPy's array class is called ndarray aka array.

### **What can you do with NumPy?**

1. Basic array operations: add, multiply, slice, flatten, reshape, index arrays
2. Advanced array operations: stack arrays, split into sections, broadcast arrays
3. Work with DateTime or Linear Algebra
4. Basic Slicing and Advanced Indexing in NumPy Python

### **c. Matplotlib :**

This is undoubtedly my favorite and a quintessential Python library. You can create stories with the data visualized with Matplotlib. Another library from the SciPy Stack, Matplotlib plots 2D figures.

### **What can you do with Matplotlib?**

Histogram, bar plots, scatter plots, area plot to pie plot, Matplotlib can depict a widerange of visualizations. With a bit of effort and tint of visualization capabilities,with Matplotlib, you can create just any visualizations:

- Scatter plots
- Area plots
- Bar charts and Histograms
- Pie charts
- Stem plots
- Contour plots
- Quiver plots
  
- Spectrograms

Matplotlib also facilitates labels, grids, legends, and some more formatting entities with Matplotlib.

### **d. Seaborn :**

So when you read the official documentation on Seaborn, it is defined as the data visualization library based on Matplotlib that provides a high-level interface for drawing attractive and informative statistical graphics. Putting it simply, seaborn is an extension of Matplotlib with advanced features.

### **What can you do with Seaborn?**

1. Determine relationships between multiple variables (correlation)
2. Observe categorical variables for aggregate statistics
3. Analyze univariate or bi-variate distributions and compare them between different data subsets
4. Plot linear regression models for dependent variables
5. Provide high-level abstractions, multi-plot grids
6. Seaborn is a great second-hand for R visualization libraries like corplot and ggplot.

#### **e. 5. Scikit Learn :**

Introduced to the world as a Google Summer of Code project, Scikit Learn is a robust machine learning library for Python. It features ML algorithms like SVMs, random forests, k-means clustering, spectral clustering, mean shift, cross-validation and more.

Even NumPy, SciPy and related scientific operations are supported by Scikit Learn with Scikit Learn being a part of the SciPy Stack.

#### **What can you do with Scikit Learn?**

1. Classification: Spam detection, image recognition
2. Clustering: Drug response, Stock price
3. Regression: Customer segmentation, Grouping experiment outcomes
4. Dimensionality reduction: Visualization, Increased efficiency
5. Model selection: Improved accuracy via parameter tuning
6. Pre-processing: Preparing input data as a text for processing with machine learning algorithms.

#### **3. Description of Dataset:**

The Iris dataset was used in R.A. Fisher's classic 1936 paper, The Use of Multiple Measurements in Taxonomic Problems, and can also be found on the UCI Machine Learning Repository. It includes three iris species with 50 samples each as well as some properties about each flower. One flower species is linearly separable from the other two, but the other two are not linearly separable from each other.

#### **Total Sample- 150**

#### **The columns in this dataset are:**

1. Id
2. SepalLengthCm
3. SepalWidthCm
4. PetalLengthCm
5. PetalWidthCm
6. Species

3 Different Types of Species each contain 50 Sample-

**iris setosa**



petal      sepal

**iris versicolor**



petal      sepal

**iris virginica**



petal      sepal

**Description of Dataset:**

**Samples**  
(instances, observations)

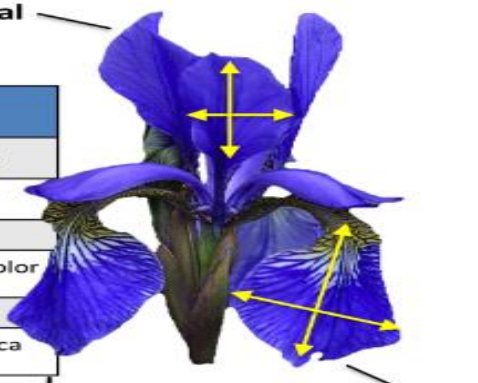
|     | Sepal length | Sepal width | Petal length | Petal width | Class label |
|-----|--------------|-------------|--------------|-------------|-------------|
| 1   | 5.1          | 3.5         | 1.4          | 0.2         | Setosa      |
| 2   | 4.9          | 3.0         | 1.4          | 0.2         | Setosa      |
| ... |              |             |              |             |             |
| 50  | 6.4          | 3.5         | 4.5          | 1.2         | Versicolor  |
| ... |              |             |              |             |             |
| 150 | 5.9          | 3.0         | 5.0          | 1.8         | Virginica   |

**Features**  
(attributes, measurements, dimensions)

**Petal**

**Sepal**

**Class labels**  
(targets)

A diagram of an Iris flower with yellow arrows indicating measurements. One arrow points to the length of a petal, labeled 'Petal'. Another arrow points to the width of a sepal, labeled 'Sepal'. A third arrow points to the center of the flower, labeled 'Class labels (targets)'.

**4. Panda Dataframe functions for Load Dataset :**

# The columns of the resulting DataFrame have different dtypes.

iris.dtypes

1. The dataset is downloaded from UCI repository.

**csv\_url='https://archive.ics.uci.edu/ml/machine-learning-databases/iris/iris.data'**

2. Now Read CSV File as a Dataframe in Python from from path where you saved the same .The Iris data set is stored in .csv format. '.csv' stands for comma separated values. It is easier to load .csv files in Pandas data frame and perform various analytical operations on it.

Load Iris.csv into a Pandas data frame —

**Syntax :**

**iris = pd.read\_csv(csv\_url, header = None)**

3. The csv file at the UCI repository does not contain the variable/column names. They are located in a separate file:

**col\_names = ['Sepal\_Length','Sepal\_Width','Petal\_Length','Petal\_Width','Species']**

4. read in the dataset from the UCI Machine Learning Repository link and specify column names to use :

**iris = pd.read\_csv(csv\_url, names = col\_names)**

|          | <b>Id</b> | <b>SepalLengthCm</b> | <b>SepalWidthCm</b> | <b>PetalLengthCm</b> | <b>PetalWidthCm</b> | <b>Species</b> |
|----------|-----------|----------------------|---------------------|----------------------|---------------------|----------------|
| <b>0</b> | 1         | 5.1                  | 3.5                 | 1.4                  | 0.2                 | Iris-setosa    |
| <b>1</b> | 2         | 4.9                  | 3.0                 | 1.4                  | 0.2                 | Iris-setosa    |
| <b>2</b> | 3         | 4.7                  | 3.2                 | 1.3                  | 0.2                 | Iris-setosa    |
| <b>3</b> | 4         | 4.6                  | 3.1                 | 1.5                  | 0.2                 | Iris-setosa    |
| <b>4</b> | 5         | 5.0                  | 3.6                 | 1.4                  | 0.2                 | Iris-setosa    |

### 5. Panda Dataframe functions for Data Preprocessing :

| <b>Sr.No</b> | <b>Data Frame Function</b> | <b>Description</b>                     |
|--------------|----------------------------|--|
| <b>1</b>     | <b>dataset.head(n=5)</b>   | Return the first n rows.               |
| <b>2</b>     | <b>dataset.tail(n=5)</b>   | Return the last n rows.                |
| <b>3</b>     | <b>dataset.index</b>       | The index (row labels) of the Dataset. |

|          |                        |  |
|----------|------------------------|--|
| <b>4</b> | <b>dataset.columns</b> | The column labels of the Dataset.                              |
| <b>5</b> | <b>dataset.shape</b>   | Return a tuple representing the dimensionality of the Dataset. |

|    |  |   |
|----|--|---|
| 6  | <code>dataset.dtypes</code>                              | Return the dtypes in the Dataset.<br>This returns a Series with the data type of each column.<br>The result's index is the original Dataset's columns<br>Columns with mixed types are stored with the object dtype.                                       |
| 7  | <code>dataset.columns.values</code>                      | Return the columns values in the Dataset in array format  |
| 8  | <code>dataset.describe(include='all')</code>             | Generate descriptive statistics.<br>to view some basic statistical details like percentile, mean, std etc. of a data frame or a series of numeric values.<br>Analyzes both numeric and object series, as well as dataset column sets of mixed data types. |
| 9  | <code>dataset['Column name']</code>                      | Read the Data Column wise.  |
| 10 | <code>dataset.sort_index(axis=1, ascending=False)</code> | Sort object by labels (along an axis).  |
| 11 | <code>dataset.sort_values(by="Column name")</code>       | Sort values by column name.   |
| 12 | <code>dataset.iloc[5]</code>                             | Purely integer-location based indexing for selection by position.   |
| 13 | <code>dataset[0:3]</code>                                | Selecting via [], which slices the rows.  |
| 14 | <code>dataset.loc[:, ["Col_name1", "col_name2"]]</code>  | Selection by label  |
| 15 | <code>dataset.iloc[:n, :]</code>                         | a subset of the first n rows of the original data   |
| 16 | <code>dataset.iloc[:, :n]</code>                         | a subset of the first n columns of the original data  |
| 17 | <code>dataset.iloc[:m, :n]</code>                        | a subset of the first m rows and the first n columns  |

Few Examples of iLoc to slice data for iris Dataset :

| Sr. No | Data Frame Function | Description | Output |
|--------|---------------------|-------------|--------|
|--------|---------------------|-------------|--------|



| 1             | dataset.iloc[3:5, 0:2]                             | Slice the data  | <table><tr><th colspan="2">Id</th><th>SepalLengthCm</th></tr><tr><td>3</td><td>4</td><td>4.6</td></tr><tr><td>4</td><td>5</td><td>5.0</td></tr></table>   | Id            |                 | SepalLengthCm | 3             | 4            | 4.6     | 4 | 5   | 5.0 |     |     |                 |   |     |     |     |     |                 |
|---------------|--|---|---|---------------|-----------------|---------------|---------------|--------------|---------|---|-----|-----|-----|-----|-----------------|---|-----|-----|-----|-----|-----------------|
| Id            |  | SepalLengthCm   |   |               |                 |               |               |              |         |   |     |     |     |     |                 |   |     |     |     |     |                 |
| 3             | 4  | 4.6   |   |               |                 |               |               |              |         |   |     |     |     |     |                 |   |     |     |     |     |                 |
| 4             | 5  | 5.0   |   |               |                 |               |               |              |         |   |     |     |     |     |                 |   |     |     |     |     |                 |
| 2             | dataset.iloc[[1, 2, 4], [0, 2]]                    | By lists of integer position locations, similar to the NumPy/Python style | <table><tr><th colspan="2">Id</th><th>SepalWidthCm</th></tr><tr><td>1</td><td>2</td><td>3.0</td></tr><tr><td>2</td><td>3</td><td>3.2</td></tr><tr><td>4</td><td>5</td><td>3.6</td></tr></table>   | Id            |                 | SepalWidthCm  | 1             | 2            | 3.0     | 2 | 3   | 3.2 | 4   | 5   | 3.6             |   |     |     |     |     |                 |
| Id            |  | SepalWidthCm  |   |               |                 |               |               |              |         |   |     |     |     |     |                 |   |     |     |     |     |                 |
| 1             | 2  | 3.0   |   |               |                 |               |               |              |         |   |     |     |     |     |                 |   |     |     |     |     |                 |
| 2             | 3  | 3.2   |   |               |                 |               |               |              |         |   |     |     |     |     |                 |   |     |     |     |     |                 |
| 4             | 5  | 3.6   |   |               |                 |               |               |              |         |   |     |     |     |     |                 |   |     |     |     |     |                 |
| 3             | dataset.iloc[1:3, :]                               | For slicing rows explicitly   | <table><tr><th>Id</th><th>SepalLengthCm</th><th>SepalWidthCm</th><th>PetalLengthCm</th><th>PetalWidthCm</th><th>Species</th></tr><tr><td>1</td><td>2</td><td>4.9</td><td>3.0</td><td>1.4</td><td>0.2 Iris-setosa</td></tr><tr><td>2</td><td>3</td><td>4.7</td><td>3.2</td><td>1.3</td><td>0.2 Iris-setosa</td></tr></table> | Id            | SepalLengthCm   | SepalWidthCm  | PetalLengthCm | PetalWidthCm | Species | 1 | 2   | 4.9 | 3.0 | 1.4 | 0.2 Iris-setosa | 2 | 3   | 4.7 | 3.2 | 1.3 | 0.2 Iris-setosa |
| Id            | SepalLengthCm                                      | SepalWidthCm  | PetalLengthCm   | PetalWidthCm  | Species         |               |               |              |         |   |     |     |     |     |                 |   |     |     |     |     |                 |
| 1             | 2  | 4.9   | 3.0   | 1.4           | 0.2 Iris-setosa |               |               |              |         |   |     |     |     |     |                 |   |     |     |     |     |                 |
| 2             | 3  | 4.7   | 3.2   | 1.3           | 0.2 Iris-setosa |               |               |              |         |   |     |     |     |     |                 |   |     |     |     |     |                 |
| 4             | dataset.iloc[:, 1:3]                               | For slicing Column explicitly   | <table><tr><th colspan="2">SepalLengthCm</th><th>SepalWidthCm</th></tr><tr><td>0</td><td>5.1</td><td>3.5</td></tr><tr><td>1</td><td>4.9</td><td>3.0</td></tr><tr><td>2</td><td>4.7</td><td>3.2</td></tr><tr><td>3</td><td>4.6</td><td>3.1</td></tr></table>   | SepalLengthCm |                 | SepalWidthCm  | 0             | 5.1          | 3.5     | 1 | 4.9 | 3.0 | 2   | 4.7 | 3.2             | 3 | 4.6 | 3.1 |     |     |                 |
| SepalLengthCm |  | SepalWidthCm  |   |               |                 |               |               |              |         |   |     |     |     |     |                 |   |     |     |     |     |                 |
| 0             | 5.1  | 3.5   |   |               |                 |               |               |              |         |   |     |     |     |     |                 |   |     |     |     |     |                 |
| 1             | 4.9  | 3.0   |   |               |                 |               |               |              |         |   |     |     |     |     |                 |   |     |     |     |     |                 |
| 2             | 4.7  | 3.2   |   |               |                 |               |               |              |         |   |     |     |     |     |                 |   |     |     |     |     |                 |
| 3             | 4.6  | 3.1   |   |               |                 |               |               |              |         |   |     |     |     |     |                 |   |     |     |     |     |                 |
| 5             | dataset.iloc[1, 1]                                 | For getting a value explicitly  | 4.9   |               |                 |               |               |              |         |   |     |     |     |     |                 |   |     |     |     |     |                 |
| 6             | dataset['Sepal Length Cm'].iloc[5]                 | Accessing Column and Rows by position                                     | 5.4   |               |                 |               |               |              |         |   |     |     |     |     |                 |   |     |     |     |     |                 |
| 7             | cols_2_4=dataset.columns[2:4]<br>dataset[cols_2_4] | Get Column Name then data from column                                     | <table><tr><th colspan="2">SepalWidthCm</th><th>PetalLengthCm</th></tr><tr><td>0</td><td>3.5</td><td>1.4</td></tr><tr><td>1</td><td>3.0</td><td>1.4</td></tr><tr><td>2</td><td>3.2</td><td>1.3</td></tr><tr><td>3</td><td>3.1</td><td>1.5</td></tr></table>   | SepalWidthCm  |                 | PetalLengthCm | 0             | 3.5          | 1.4     | 1 | 3.0 | 1.4 | 2   | 3.2 | 1.3             | 3 | 3.1 | 1.5 |     |     |                 |
| SepalWidthCm  |  | PetalLengthCm   |   |               |                 |               |               |              |         |   |     |     |     |     |                 |   |     |     |     |     |                 |
| 0             | 3.5  | 1.4   |   |               |                 |               |               |              |         |   |     |     |     |     |                 |   |     |     |     |     |                 |
| 1             | 3.0  | 1.4   |   |               |                 |               |               |              |         |   |     |     |     |     |                 |   |     |     |     |     |                 |
| 2             | 3.2  | 1.3   |   |               |                 |               |               |              |         |   |     |     |     |     |                 |   |     |     |     |     |                 |
| 3             | 3.1  | 1.5   |   |               |                 |               |               |              |         |   |     |     |     |     |                 |   |     |     |     |     |                 |
| 8             | dataset[dataset.columns[2:4]].iloc[5:10]           | in one Expression answer for the above commands                           | <table><tr><th colspan="2">SepalWidthCm</th><th>PetalLengthCm</th></tr><tr><td>5</td><td>3.9</td><td>1.7</td></tr><tr><td>6</td><td>3.4</td><td>1.4</td></tr><tr><td>7</td><td>3.4</td><td>1.5</td></tr><tr><td>8</td><td>2.9</td><td>1.4</td></tr><tr><td>9</td><td>3.1</td><td>1.5</td></tr></table>                      | SepalWidthCm  |                 | PetalLengthCm | 5             | 3.9          | 1.7     | 6 | 3.4 | 1.4 | 7   | 3.4 | 1.5             | 8 | 2.9 | 1.4 | 9   | 3.1 | 1.5             |
| SepalWidthCm  |  | PetalLengthCm   |   |               |                 |               |               |              |         |   |     |     |     |     |                 |   |     |     |     |     |                 |
| 5             | 3.9  | 1.7   |   |               |                 |               |               |              |         |   |     |     |     |     |                 |   |     |     |     |     |                 |
| 6             | 3.4  | 1.4   |   |               |                 |               |               |              |         |   |     |     |     |     |                 |   |     |     |     |     |                 |
| 7             | 3.4  | 1.5   |   |               |                 |               |               |              |         |   |     |     |     |     |                 |   |     |     |     |     |                 |
| 8             | 2.9  | 1.4   |   |               |                 |               |               |              |         |   |     |     |     |     |                 |   |     |     |     |     |                 |
| 9             | 3.1  | 1.5   |   |               |                 |               |               |              |         |   |     |     |     |     |                 |   |     |     |     |     |                 |

Checking of Missing Values in Dataset:

- **isnull()** is the function that is used to check missing values or null values in pandas python.
- **isna()** function is also used to get the count of missing values of column and row wise count of missing values
- The dataset considered for explanation is:

|   | Name    | State      | Gender | Score |
|---|---------|------------|--------|-------|
| 0 | George  | Arizona    | M      | 63.0  |
| 1 | Andrea  | Georgia    | F      | 48.0  |
| 2 | micheal | Newyork    | M      | 56.0  |
| 3 | maggie  | Indiana    | F      | 75.0  |
| 4 | Ravi    | Florida    | M      | NaN   |
| 5 | Xien    | California | M      | 77.0  |
| 6 | Jalpa   | NaN        | NaN    | NaN   |
| 7 | NaN     | NaN        | NaN    | NaN   |

a. is there any missing values in dataframe as a whole

**Function:** DataFrame.isnull()

**Output:**

|   | Name  | State | Gender | Score |
|---|-------|-------|--------|-------|
| 0 | False | False | False  | False |
| 1 | False | False | False  | False |
| 2 | False | False | False  | False |
| 3 | False | False | False  | False |
| 4 | False | False | False  | True  |
| 5 | False | False | False  | False |
| 6 | False | True  | True   | True  |
| 7 | True  | True  | True   | True  |

b. is there any missing values across each column

**Function:** DataFrame.isnull().any()

**Output:**

```
Name      True
State      True
Gender     True
Score     True
dtype: bool
```

c. count of missing values across each column using isna() and isnull()

In order to get the count of missing values of the entire dataframe `isnull()` function is used. `sum()` which does the column wise sum first and doing another `sum()` will get the count of missing values of the entire dataframe.

**Function:** `dataframe.isnull().sum().sum()`

**Output :** 8

#### d. count row wise missing value using `isnull()`

**Function:** `dataframe.isnull().sum(axis = 1)`

**Output:**

```
0    0
1    0
2    0
3    0
4    1
5    0
6    3
7    4
dtype: int64
```

#### Method 2:

**function:** `dataframe.isna().sum()`

```
Name      1
State      2
Gender     2
Score      3
dtype: int64
```

#### f. count of missing values of a specific column.

**Function:** `dataframe.col_name.isnull().sum()`  
`df1.Gender.isnull().sum()`

**Output:** 2

#### g. groupby count of missing values of a column.

In order to get the count of missing values of the particular column by group in pandas we will be using `isnull()` and `sum()` function with `apply()` and `groupby()` which performs the group wise count of missing values as shown below.

**Function:** `df1.groupby(['Gender'])['Score'].apply(lambda x:x.isnull().sum())`

**Output:**

```
Gender
F      0
M      1
Name: Score, dtype: int64
```

## 6. Panda functions for Data Formatting and Normalization :

The Transforming data stage is about converting the data set into a format that can be analyzed or modelled effectively, and there are several techniques for this process.

a. Data Formatting: Ensuring all data formats are correct (e.g. object, text, floating number, integer, etc.) is another part of this initial 'cleaning' process. If you are working with dates in Pandas, they also need to be stored in the exact format to use special date-time functions.

### Functions used for data formatting :

| Sr. No | Data Frame Function   | Description   | Output   |
|--------|---|---|--|
| 1      | df.dtypes   | To check the data type  | <pre>df.dtypes sepal length (cm)    float64 sepal width (cm)     float64 petal length (cm)    float64 petal width (cm)     float64 dtype: object</pre> |
| 2      | df['petal length (cm)'] = df['petal length (cm)'].astype("int") | To change the data type (data type of 'petal length (cm)' changed to int) | <pre>df.dtypes sepal length (cm)    float64 sepal width (cm)     float64 petal length (cm)    int64 petal width (cm)     float64 dtype: object</pre>   |

### b. Data normalization:

Mapping all the nominal data values onto a uniform scale (e.g. from 0 to 1) is involved in data normalization. Making the ranges consistent across variables helps with statistical analysis and ensures better comparisons later on. It is also known as Min-Max scaling.

#### Algorithm:

**Step 1 :** Import pandas and sklearn library for preprocessing from sklearn import preprocessing

**Step 2 :** Load the iris dataset in dataframe object df

**Step 3 :** Print iris dataset.

```
df.head()
```

**Step 4 :** Create a minimum and maximum processor object

```
min_max_scaler=preprocessing.MinMaxScaler()
```

**Step 5 :** Separate the feature from the class label

```
x=df.iloc[:,4]
```

**Step 6 :** Create an object to transform the data to fit minmax processor

```
x_scaled = min_max_scaler.fit_transform(x)
```

**Step 7 :** Run the normalizer on the dataframe

```
df_normalized = pd.DataFrame(x_scaled)
```

**Step 8:** View the dataframe

```
df_normalized
```

**Output: After Step 3:**

|   | sepal length (cm) | sepal width (cm) | petal length (cm) | petal width (cm) |
|---|-------------------|------------------|-------------------|------------------|
| 0 | 5.1               | 3.5              | 1.4               | 0.2              |
| 1 | 4.9               | 3.0              | 1.4               | 0.2              |
| 2 | 4.7               | 3.2              | 1.3               | 0.2              |
| 3 | 4.6               | 3.1              | 1.5               | 0.2              |
| 4 | 5.0               | 3.6              | 1.4               | 0.2              |

Output

**after step 8:**

|     | 0        | 1        | 2        | 3        |
|-----|----------|----------|----------|----------|
| 0   | 0.222222 | 0.625000 | 0.067797 | 0.041667 |
| 1   | 0.166667 | 0.416667 | 0.067797 | 0.041667 |
| 2   | 0.111111 | 0.500000 | 0.050847 | 0.041667 |
| 3   | 0.083333 | 0.458333 | 0.084746 | 0.041667 |
| 4   | 0.194444 | 0.666667 | 0.067797 | 0.041667 |
| ... | ...      | ...      | ...      | ...      |
| 145 | 0.666667 | 0.416667 | 0.711864 | 0.916667 |
| 146 | 0.555556 | 0.208333 | 0.677966 | 0.750000 |
| 147 | 0.611111 | 0.416667 | 0.711864 | 0.791667 |
| 148 | 0.527778 | 0.583333 | 0.745763 | 0.916667 |
| 149 | 0.444444 | 0.416667 | 0.694915 | 0.708333 |

150 rows x 4 columns

## 7. Panda Functions for handling categorical variables :

- Categorical variables have values that describe a 'quality' or 'characteristic' of a data unit, like 'what type' or 'which category'.
- Categorical variables fall into mutually exclusive (in one category or in another) and exhaustive (include all possible options) categories. Therefore, categorical variables are qualitative variables and tend to be represented by a non-numeric value.

- Categorical features refer to string type data and can be easily understood by human beings. But in case of a machine, it cannot interpret the categorical data directly. Therefore, the categorical data must be translated into numerical data that can be understood by machine.

There are many ways to convert categorical data into numerical data. Here the three most used methods are discussed.

**a. Label Encoding:** Label Encoding refers to converting the labels into a numeric form so as to convert them into the machine-readable form. It is an important preprocessing step for the structured dataset in supervised learning.

Example : Suppose we have a column Height in some dataset. After applying label encoding, the Height column is converted into:

| Height | Height |
|--------|--------|
| Tall   | 0      |
| Medium | 1      |
| Short  | 2      |

where 0 is the label for tall, 1 is the label for medium, and 2 is a label for short height.

**Label Encoding on iris**

**dataset:** For iris dataset the target column which is Species. It contains three species Iris-setosa, Iris-versicolor, Iris-

virginica.

**Sklearn Functions for Label Encoding:**

- **preprocessing.LabelEncoder** : It Encode labels with value between 0 and n\_classes-1.

- **fit\_transform(y):** Parameters: yarray-like of shape (n\_samples,) Target values.

**Returns:** yarray-like of shape (n\_samples,)

**Encoded labels.**

This transformer should be used to encode target values, and not the input.

**Algorithm:**

**Step 1 :** Import pandas and sklearn library for preprocessing  
from sklearn import preprocessing

**Step 2:** Load the iris dataset in dataframe object df

**Step 3:** Observe the unique values for the Species column.  
df['Species'].unique()

output:

```
array(['Iris-setosa', 'Iris-versicolor', 'Iris-virginica'], dtype=object)
```

**Step 4:** define label\_encoder object knows how to understand word labels.

```
label_encoder = preprocessing.LabelEncoder()
```

**Step 5:** Encode labels in column 'species'.

```
df['Species'] = label_encoder.fit_transform(df['Species'])
```

**Step 6:** Observe the unique values for the Species column.

```
df['Species'].unique()
```

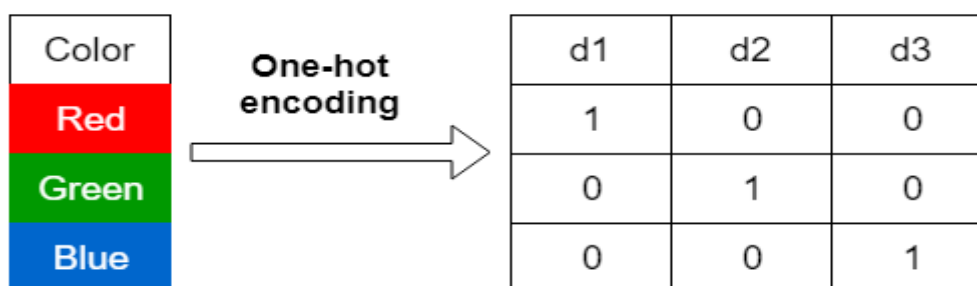
**Output:** array([0, 1, 2], dtype=int64)

- Use LabelEncoder when there are only two possible values of a categorical feature. For example, features having value such as yes or no. Or, maybe, gender features when there are only two possible values including male or female.

**Limitation:** Label encoding converts the data in machine-readable form, but it assigns a unique number (starting from 0) to each class of data. This may lead to the generation of priority issues in the data sets. A label with a high value may be considered to have high priority than a label having a lower value.

### b. One-Hot Encoding:

In one-hot encoding, we create a new set of dummy (binary) variables that is equal to the number of categories (k) in the variable. For example, let's say we have a categorical variable Color with three categories called "Red", "Green" and "Blue", we need to use three dummy variables to encode this variable using one-hot encoding. A dummy (binary) variable just takes the value 0 or 1 to indicate the exclusion or inclusion of a category.



In one-hot encoding,

“Red” color is encoded as [1 0 0] vector of size 3.

“Green” color is encoded as [0 1 0] vector of size 3.

“Blue” color is encoded as [0 0 1] vector of size 3.

**One-hot encoding** on iris dataset: For iris dataset the target column which is Species. It

contains three species Iris-setosa, Iris-versicolor, Iris-virginica.

### Sklearn Functions for One-hot Encoding:

#### • sklearn.preprocessing.OneHotEncoder():

Encode categorical integer features using a one-hot aka one-of-K scheme

#### Algorithm:

**Step 1 :** Import pandas and sklearn library for preprocessing from sklearn import preprocessing

**Step 2:** Load the iris dataset in dataframe object df

**Step 3:** Observe the unique values for the Species column.

`df['Species'].unique()`

**output:** array(['Iris-setosa', 'Iris-versicolor', 'Iris-virginica'], dtype=object)

**Step 4:** Apply label\_encoder object for label encoding the Observe the unique values for the Species column.

`df['Species'].unique()`

**Output:** array([0, 1, 2], dtype=int64)

**Step 5:** Remove the target variable from dataset

`features_df=df.drop(columns=['Species'])`

**Step 6:** Apply one\_hot encoder for Species column.

`enc = preprocessing.OneHotEncoder()`

`enc_df=pd.DataFrame(enc.fit_transform(df[['Species']])).toarray()`

**Step 7:** Join the encoded values with Features variable

`df_encode = features_df.join(enc_df)`

**Step 8 :** Observe the merge dataframe

`df_encode`

**Step 9:** Rename the newly encoded columns.

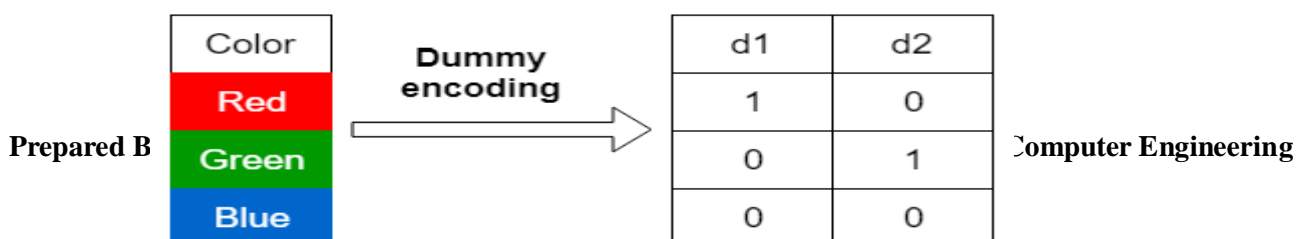
`df_encode.rename(columns = {0:'Iris-Setosa', 1:'Iris-Versicolor',2:'Iris-virginica'},inplace=True)`

**Step 10:** Observe the merge dataframe

`df_encode`

### c. Dummy Variable Encoding :

Dummy encoding also uses dummy (binary) variables. Instead of creating a number of dummy variables that is equal to the number of categories (k) in the variable, dummy encoding uses k-1 dummy variables. To encode the same Color variable with three categories using the dummy encoding, we need to use only two dummy variables.





In dummy encoding,

“Red” color is encoded as [1 0] vector of size 2.

“Green” color is encoded as [0 1] vector of size 2.

“Blue” color is encoded as [0 0] vector of size 2.

Dummy encoding removes a duplicate category present in the one-hot encoding.

### Pandas Functions for One-hot Encoding with dummy variables:

- **pandas.get\_dummies(data, dummy\_na=False, prefix=None, prefix\_sep='\_', columns=None, sparse=False, drop\_first=False, dtype=None):** Convert categorical variable into dummy/indicator variables.

- **Parameters:**

**data:** array-like, Series, or DataFrame

Data of which to get dummy indicators.

**prefixstr:** list of str, or dict of str, default None String to append DataFrame column names.

**prefix\_sep:** str, default ‘\_’

If appending prefix, separator/delimiter to use. Or pass a list or dictionary as with prefix.

**dummy\_na:** bool, default False

Add a column to indicate NaNs, if False NaNs are ignored.

**columns:** list-like, default None

Column names in the DataFrame to be encoded. If columns is None then all the columns with object or category dtype will be converted.

**sparse:** bool, default False

Whether the dummy-encoded columns should be backed by a SparseArray (

True) or a regular NumPy array (False).

**drop\_first:** bool, default False

Whether to get k-1 dummies out of k categorical levels by removing the first level.

**dtype:** dtype, default np.uint8

Data type for new columns. Only a single dtype is allowed.

- **Return :** DataFrame with Dummy-coded data.

### Algorithm:

**Step 1 :** Import pandas and sklearn library for preprocessing  
from sklearn import preprocessing

**Step 2:** Load the iris dataset in dataframe object df

**Step 3:** Observe the unique values for the Species column.

`df['Species'].unique()`

output: array(['Iris-setosa', 'Iris-versicolor', 'Iris-virginica'], dtype=object)

**Step 4:** Apply label\_encoder object for label encoding the Observe the unique values for the Species column.

`df['Species'].unique()`

Output: array([0, 1, 2], dtype=int64)

**Step 5:** Apply one\_hot encoder with dummy variables for Species column.

`one_hot_df = pd.get_dummies(df, prefix="Species", columns=['Species'], drop_first=True)`

**Step 7 :** Observe the merge dataframe

`one_hot_df`



|     | Sepal_Length | Sepal_Width | Petal_Length | Petal_Width | Species_1 | Species_2 |
|-----|--------------|-------------|--------------|-------------|-----------|-----------|
| 0   | 5.1          | 3.5         | 1.4          | 0.2         | 0         | 0         |
| 1   | 4.9          | 3.0         | 1.4          | 0.2         | 0         | 0         |
| 2   | 4.7          | 3.2         | 1.3          | 0.2         | 0         | 0         |
| 3   | 4.6          | 3.1         | 1.5          | 0.2         | 0         | 0         |
| 4   | 5.0          | 3.6         | 1.4          | 0.2         | 0         | 0         |
| ... | ...          | ...         | ...          | ...         | ...       | ...       |
| 145 | 6.7          | 3.0         | 5.2          | 2.3         | 0         | 1         |
| 146 | 6.3          | 2.5         | 5.0          | 1.9         | 0         | 1         |
| 147 | 6.5          | 3.0         | 5.2          | 2.0         | 0         | 1         |
| 148 | 6.2          | 3.4         | 5.4          | 2.3         | 0         | 1         |
| 149 | 5.9          | 3.0         | 5.1          | 1.8         | 0         | 1         |

150 rows x 6 columns

**Conclusion-** In this way we have explored the functions of the python library for Data Preprocessing, Data Wrangling Techniques and How to Handle missing values on Iris Dataset.

### Group A Assignment No: 2

#### Contents for Theory:

1. Creation of Dataset .
2. Identification and Handling of Null Values
3. Identification and Handling of Outliers
4. Data Transformation for the purpose of :
  - a. To change the scale for better understanding
  - b. To decrease the skewness and convert distribution into normal distribution

#### Theory:

##### 1. Creation of Dataset using Microsoft Excel :

The dataset is created in “CSV” format.

- The name of dataset is **StudentsPerformance**
- **The features of the dataset are:** Math\_Score, Reading\_Score, Writing\_Score, Placement\_Score, Club\_Join\_Date .
- **Number of Instances:** 30
- **The response variable is:** Placement\_Offer\_Count .
- **Range of Values:**  
Math\_Score[60-80], Reading\_Score[75-,95],Writing\_Score [60,80],  
Placement\_Score[75-100], Club\_Join\_Date [2018-2021].
- The response variable is the number of placement offers facilitated to particular students, which is largely depend on Placement\_Score

To fill the values in the dataset the **RANDBETWEEN** is used. Returns a random integer number between the numbers you specify

**Syntax : RANDBETWEEN(bottom, top)** **Bottom** The smallest integer and  
**Top** The largest integer RANDBETWEEN will return.

For better understanding and visualization, 20% impurities are added into each variable to the dataset.

## 2. Identification and Handling of Null Values :

Missing Data can occur when no information is provided for one or more items or for a whole unit. Missing Data is a very big problem in real-life scenarios. Missing Data can also refer to as NA(Not Available) values in pandas. In DataFrame sometimes many datasets simply arrive with missing data, either because it exists and was not collected or it never existed. For Example, Suppose different users being surveyed may choose not to share their income, some users may choose not to share the address in this way many datasets went missing.

In Pandas missing data is represented by two value:

**1. None:** None is a Python singleton object that is often used for missing data in

Python code.

**2. NaN :** NaN (an acronym for Not a Number), is a special floating-point value recognized by all systems that use the standard IEEE floating-point representation.

Pandas treat None and NaN as essentially interchangeable for indicating missing or null values. To facilitate this convention, there are several useful functions for detecting, removing, and replacing null values in Pandas DataFrame :

- isnull()
- notnull()
- dropna()
- fillna()
- replace()

### 1. Checking for missing values using isnull() and notnull() :

#### ●Checking for missing values using isnull() :

In order to check null values in Pandas DataFrame, isnull() function is used. This function return dataframe of Boolean values which are True for NaN values .

#### Algorithm:

**Step 1 :** Import pandas and numpy in order to check missing values in Pandas DataFrame

```
import pandas as pd
import numpy as np
```

**Step 2:** Load the dataset in dataframe object df

```
df=pd.read_csv("/content/StudentsPerformanceTest1.csv")
```

**Step 3:** Display the data frame

**df**

|   | gender | math score | reading score | writing score | Placement Score | placement offer count | Region |
|---|--------|------------|---------------|---------------|-----------------|-----------------------|--------|
| 0 | female | 72         | 72            | 74.0          | 78.0            | 1                     | Pune   |
| 1 | female | 69         | 90            | 88.0          | NaN             | 2                     | na     |
| 2 | female | 90         | 95            | 93.0          | 74.0            | 2                     | Nashik |
| 3 | male   | 47         | 57            | NaN           | 78.0            | 1                     | Na     |
| 4 | male   | na         | 78            | 75.0          | 81.0            | 3                     | Pune   |
| 5 | female | 71         | Na            | 78.0          | 70.0            | 4                     | na     |
| 6 | male   | 12         | 44            | 52.0          | 12.0            | 2                     | Nashik |
| 7 | male   | NaN        | 65            | 67.0          | 49.0            | 1                     | Pune   |
| 8 | male   | 5          | 77            | 89.0          | 55.0            | 0                     | NaN    |

**Step 4:** Use isnull() function to check null values in the dataset.

**df.isnull()**

|   | gender | math score | reading score | writing score | Placement Score | placement offer count | Region |
|---|--------|------------|---------------|---------------|-----------------|-----------------------|--------|
| 0 | False  | False      | False         | False         | False           | False                 | False  |
| 1 | False  | False      | False         | False         | True            | False                 | False  |
| 2 | False  | False      | False         | False         | False           | False                 | False  |
| 3 | False  | False      | False         | True          | False           | False                 | False  |
| 4 | False  | False      | False         | False         | False           | False                 | False  |
| 5 | False  | False      | False         | False         | False           | False                 | False  |
| 6 | False  | False      | False         | False         | False           | False                 | False  |
| 7 | False  | True       | False         | False         | False           | False                 | False  |
| 8 | False  | False      | False         | False         | False           | False                 | True   |

**Step 5:** To create a series true for NaN values for specific columns. for example math score in dataset and display data with only math score as NaN

```
series = pd.isnull(df["math score"])
df[series]
```

|   | gender | math score | reading score | writing score | Placement Score | placement offer count | Region |
|---|--------|------------|---------------|---------------|-----------------|-----------------------|--------|
| 7 | male   | NaN        | 65            | 67.0          | 49.0            | 1                     | Pune   |

#### ● Checking for missing values using notnull()

In order to check null values in Pandas Dataframe, notnull() function is used. This function return dataframe of Boolean values which are False for NaN values.

**Algorithm:**

**Step 1 :** Import pandas and numpy in order to check missing values in Pandas DataFrame

```
import pandas as pd
import numpy as np
```

**Step 2:** Load the dataset in dataframe object df

```
df=pd.read_csv("/content/StudentsPerformanceTest1.csv")
```

**Step 3:** Display the data frame  
df

|   | gender | math score | reading score | writing score | Placement Score | placement offer count | Region |
|---|--------|------------|---------------|---------------|-----------------|-----------------------|--------|
| 0 | female | 72         | 72            | 74.0          | 78.0            | 1                     | Pune   |
| 1 | female | 69         | 90            | 88.0          | NaN             | 2                     | na     |
| 2 | female | 90         | 95            | 93.0          | 74.0            | 2                     | Nashik |
| 3 | male   | 47         | 57            | NaN           | 78.0            | 1                     | Na     |
| 4 | male   | na         | 78            | 75.0          | 81.0            | 3                     | Pune   |
| 5 | female | 71         | Na            | 78.0          | 70.0            | 4                     | na     |
| 6 | male   | 12         | 44            | 52.0          | 12.0            | 2                     | Nashik |
| 7 | male   | NaN        | 65            | 67.0          | 49.0            | 1                     | Pune   |
| 8 | male   | 5          | 77            | 89.0          | 55.0            | 0                     | NaN    |

**Step 4:** Use notnull() function to check null values in the dataset.

```
df.notnull()
```

|   | gender | math score | reading score | writing score | Placement Score | placement offer count | Region |
|---|--------|------------|---------------|---------------|-----------------|-----------------------|--------|
| 0 | True   | True       | True          | True          | True            | True                  | True   |
| 1 | True   | True       | True          | True          | False           | True                  | True   |
| 2 | True   | True       | True          | True          | True            | True                  | True   |
| 3 | True   | True       | True          | False         | True            | True                  | True   |
| 4 | True   | True       | True          | True          | True            | True                  | True   |
| 5 | True   | True       | True          | True          | True            | True                  | True   |
| 6 | True   | True       | True          | True          | True            | True                  | True   |
| 7 | True   | False      | True          | True          | True            | True                  | True   |
| 8 | True   | True       | True          | True          | True            | True                  | False  |

**Step 5:** To create a series true for NaN values for specific columns. for example math score in dataset and display data with only math score as NaN

```
series1 = pd.notnull(df["math score"])
df[series1]
```

See that there are also categorical values in the dataset, for this, you need to use Label Encoding or One Hot Encoding.

- from sklearn.preprocessing import LabelEncoder
- le = LabelEncoder()
- df['gender'] = le.fit\_transform(df['gender'])
- newdf = df

df

## 2. Filling missing values using dropna(), fillna(), replace()

In order to fill null values in a datasets, fillna(), replace() functions are used. These functions replace NaN values with some value of their own. All these functions help in filling null values in datasets of a DataFrame.

- **For replacing null values with NaN**

```
missing_values = ["Na", "na"]
df = pd.read_csv("StudentsPerformanceTest1.csv", na_values = missing_values)
df
```

- **Filling null values with a single value**

**Step 1 :** Import pandas and numpy in order to check missing values in Pandas

```
DataFrame
import pandas as pd
import numpy as np
```

**Step 2:** Load the dataset in dataframe object df

```
df=pd.read_csv("/content/StudentsPerformanceTest1.csv")
```

**Step 3:** Display the data frame

```
df
```

**Step 4:** filling missing value using fillna()

```
nfd=df
nfd.fillna(0)
```

**Step 5:** filling missing values using mean, median and standard deviation of that column.

```
data['math score'] = data['math score'].fillna(data['math score'].mean())
data["math score"] = data["math score"].fillna(data["math score"].median())
data['math score'] = data['math score'].fillna(data["math score"].std())
replacing missing values in forenoon column with minimum/maximum number
of that column
data["math score"] = data["math score"].fillna(data["math score"].min())
data["math score"] = data["math score"].fillna(data["math score"].max())
```

- **Filling null values in dataset**

To fill null values in dataset use inplace=True

```
m_v=df['math score'].mean()
df['math score'].fillna(value=m_v, inplace=True)
df
```

- **Filling a null values using replace() method**

Following line will replace Nan value in dataframe with value -99

```
nfd.replace(to_replace = np.nan, value = -99)
```

- **Deleting null values using dropna() method**

In order to drop null values from a dataframe, dropna() function is used. This function drops Rows/Columns of datasets with Null values in different ways.

1. Dropping rows with at least 1 null value
2. Dropping rows if all values in that row are missing
3. Dropping columns with at least 1 null value.
4. Dropping Rows with at least 1 null value in CSV file

**Algorithm:**

**Step 1 :** Import pandas and numpy in order to check missing values in Pandas

```
DataFrame
import pandas as pd
import numpy as np
```

**Step 2:** Load the dataset in dataframe object df

```
df=pd.read_csv("/content/StudentsPerformanceTest1.csv")
```

**Step 3:** Display the data frame

```
df
```

**Step 4 :** To drop rows with at least 1 null value

```
ndf.dropna()
```

**Step 5:** To Drop rows if all values in that row are missing

```
ndf.dropna(how = 'all')
```

**Step 6:** To Drop columns with at least 1 null value.

```
ndf.dropna(axis = 1)
```

**Step 7 :** To drop rows with at least 1 null value in CSV file.

making new data frame with dropped NA values

```
new_data = ndf.dropna(axis = 0, how ='any')
```

```
new_data
```

|   | gender | math score | reading score | writing score | Placement Score | placement offer count | Region |
|---|--------|------------|---------------|---------------|-----------------|-----------------------|--------|
| 0 | female | 72         | 72            | 74.0          | 78.0            | 1                     | Pune   |
| 2 | female | 90         | 95            | 93.0          | 74.0            | 2                     | Nashik |
| 4 | male   | na         | 78            | 75.0          | 81.0            | 3                     | Pune   |
| 5 | female | 71         | Na            | 78.0          | 70.0            | 4                     | na     |
| 6 | male   | 12         | 44            | 52.0          | 12.0            | 2                     | Nashik |

**3. Identification and Handling of Outliers :**



### 3.1 Identification of Outliers :

One of the most important steps as part of data preprocessing is detecting and treating the outliers as they can negatively affect the statistical analysis and the training process of a machine learning algorithm resulting in lower accuracy.

#### 1. What are Outliers?

We all have heard of the idiom 'odd one out' which means something unusual in comparison to the others in a group.

Similarly, an Outlier is an observation in a given dataset that lies far from the rest of the observations. That means an outlier is vastly larger or smaller than the remaining values in the set.

#### 2. Why do they occur?

An outlier may occur due to the variability in the data, or due to experimental error/human error.

They may indicate an experimental error or heavy skewness in the data (heavy-tailed distribution).

#### 3. What do they affect?

In statistics, we have three measures of central tendency namely Mean, Median, and Mode. They help us describe the data.

Mean is the accurate measure to describe the data when we do not have any outliers present. Median is used if there is an outlier in the dataset. Mode is used if there is an outlier AND about 1/2 or more of the data is the same.

'Mean' is the only measure of central tendency that is affected by the outliers which in turn impacts Standard deviation.

Example:

Consider a small dataset, sample= [15, 101, 18, 7, 13, 16, 11, 21, 5, 15, 10, 9]. By looking at it, one can quickly say '101' is an outlier that is much larger than the other values .

| with outlier     | without outlier |
|------------------|-----------------|
| Mean: 20.08      | Mean: 12.72     |
| Median: 14.0     | Median: 13.0    |
| Mode: 15         | Mode: 15        |
| Variance: 614.74 | Variance: 21.28 |
| Std dev: 24.79   | Std dev: 4.61   |

fig. Computation with and without outlier

From the above calculations, we can clearly say the Mean is more affected than the Median.

#### 4. Detecting Outliers

If our dataset is small, we can detect the outlier by just looking at the dataset. But what if we have a huge dataset, how do we identify the outliers then? We need to use visualization and mathematical techniques.

Below are some of the techniques of detecting outliers :

- Boxplots
- Scatterplots
- Z-score
- Inter Quantile Range(IQR)

##### 4.1 Detecting outliers using Boxplot:

It captures the summary of the data effectively and efficiently with only a simple box and whiskers. Boxplot summarizes sample data using 25th, 50th, and 75th percentiles. One can just get insights(quarters, median, and outliers) into the dataset by just looking at its boxplot.

#### Algorithm:

**Step 1 :** Import pandas and numpy libraries

```
import pandas as pd
import numpy as np
```

**Step 2:** Load the dataset in dataframe object df  
`df=pd.read_csv("/content/demo.csv")`

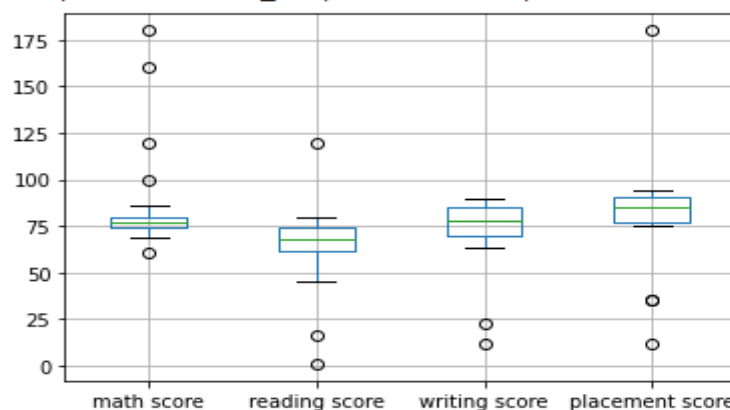
**Step 3:** Display the data frame  
`df`

|    | math score | reading score | writing score | placement score | placement offer count |
|----|------------|---------------|---------------|-----------------|-----------------------|
| 0  | 80         | 68            | 70            | 89              | 3                     |
| 1  | 71         | 61            | 85            | 91              | 3                     |
| 2  | 79         | 16            | 87            | 77              | 2                     |
| 3  | 61         | 77            | 74            | 76              | 2                     |
| 4  | 78         | 71            | 67            | 90              | 3                     |
| 5  | 73         | 68            | 90            | 80              | 2                     |
| 6  | 77         | 62            | 70            | 35              | 2                     |
| 7  | 74         | 45            | 80            | 12              | 1                     |
| 8  | 76         | 60            | 79            | 77              | 2                     |
| 9  | 75         | 65            | 85            | 87              | 3                     |
| 10 | 160        | 67            | 12            | 83              | 2                     |
| 11 | 79         | 72            | 88            | 180             | 2                     |
| 12 | 80         | 80            | 78            | 94              | 3                     |

**Step 4 :**Select the columns for boxplot and draw the boxplot.

```
col = ['math score', 'reading score', 'writing score','placement score']
df.boxplot(col)
```

<matplotlib.axes.\_subplots.AxesSubplot at 0x7f16c473d250>



**Step 5:** We can now print the outliers for each column with reference to the box plot

```
print(np.where(df['math score']>90))
print(np.where(df['reading score']<25))
print(np.where(df['writing score']<30))
```

#### 4.2 Detecting outliers using Scatterplot:

It is used when you have paired numerical data, or when your dependent variable has multiple values for each reading independent variable, or when trying to determine the relationship between the two variables. In the process of utilizing the scatter plot, one can also use it for outlier detection.

To plot the scatter plot one requires two variables that are somehow related to each other. So here Placement score and Placement count features are used.

##### Algorithm:

**Step 1 :** Import pandas , numpy and matplotlib libraries

```
import pandas as pd
```

```
import numpy as np
```

```
import matplotlib.pyplot as plt
```

**Step 2:** Load the dataset in dataframe object df

```
df=pd.read_csv("/content/demo.csv")
```

**Step 3:** Display the data frame

```
df
```

**Step 4:** Draw the scatter plot with placement score and placement offer count

```
fig, ax = plt.subplots(figsize = (18,10))
```

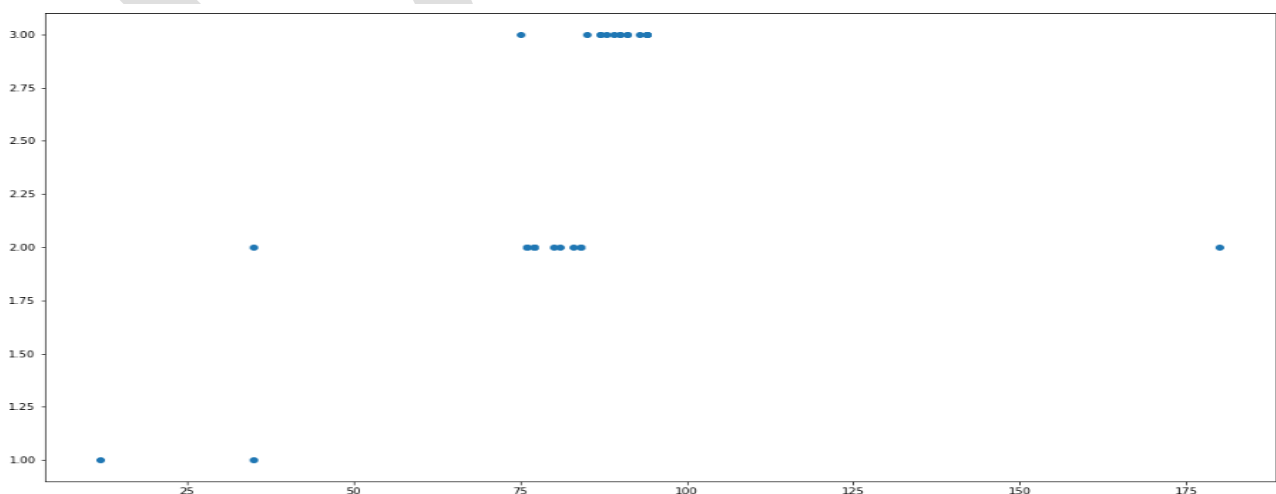
```
ax.scatter(df['placement score'], df['placement offer count'])
```

```
plt.show()
```

Labels to the axis can be assigned (Optional)

```
ax.set_xlabel('(Proportion non-retail business acres)/(town)')
```

```
ax.set_ylabel('(Full-value property-tax rate)/( $10,000)')
```



**Step 5:** We can now print the outliers with reference to scatter plot.

```
print(np.where((df['placement score']<50) & (df['placement score']>85) & (df['placement offer count']>1)))
print(np.where((df['placement offer count']<3)))
```

#### 4.3 Detecting outliers using Z-Score:

Z-Score is also called a standard score. This value/score helps to understand how far is the data point from the mean. And after setting up a threshold value one can utilize z score values of data points to define the outliers.  

$$\text{Zscore} = (\text{data\_point} - \text{mean}) / \text{std. Deviation}$$

##### Algorithm:

**Step 1 :** Import numpy and stats from scipy libraries

```
import numpy as np
from scipy import stats
```

**Step 2:** Calculate Z-Score for maths score column

```
z = np.abs(stats.zscore(df['math score']))
```

**Step 3:** Print Z-Score Value. It prints the z-score values of each data item of the column

```
print(z)
```

```
[0.17564553 0.5282877 0.21482799 0.92011234 0.25401045 0.44992277
0.29319292 0.41074031 0.33237538 0.37155785 2.95895157 0.21482799
0.17564553 0.25401045 0.37155785 0.25401045 0.05944926 0.17564553
0.37155785 0.0972806 0.60665263 0.60800375 0.48910524 0.41074031
0.37155785 3.74260085 0.48910524 0.5282877 1.39165302]
```

**Step 4:** Now to define an outlier threshold value is chosen.

```
threshold = 0.18
```

**Step 5:** Display the sample outliers

```
sample_outliers = np.where(z < threshold)
sample_outliers
```

```
(array([ 0, 12, 16, 17, 19]),)
```

#### 4.4 Detecting outliers using Inter Quartile Range(IQR):

IQR (Inter Quartile Range) Inter Quartile Range approach to finding the outliers is the most commonly used and most trusted approach used in the research field.

$$\text{IQR} = \text{Quartile3} - \text{Quartile1}$$

To define the outlier base value is defined above and below datasets normal range namely Upper and Lower bounds, define the upper and the lower bound ( $1.5 \times \text{IQR}$  value is considered) :

$$\text{upper} = Q3 + 1.5 \times \text{IQR}$$

$$\text{lower} = Q1 - 1.5 \times \text{IQR}$$

In the above formula as according to statistics, the 0.5 scale-up of IQR

(new\_IQR = IQR + 0.5\*IQR) is taken.

#### Algorithm:

**Step 1 :** Import numpy library  
import numpy as np

**Step 2:** Sort Reading Score feature and store it into sorted\_rscore.  
sorted\_rscore= sorted(df['reading score'])

**Step 3:** Print sorted\_rscore  
sorted\_rscore

**Step 4:** Calculate and print Quartile 1 and Quartile 3  
q1 = np.percentile(sorted\_rscore, 25)  
q3 = np.percentile(sorted\_rscore, 75)  
print(q1,q3)

62.0 74.0

**Step 5:** Calculate value of IQR (Inter Quartile Range)  
IQR = q3-q1

**Step 6:** Calculate and print Upper and Lower Bound to define the outlier base value.  
lwr\_bound = q1-(1.5\*IQR)  
upr\_bound = q3+(1.5\*IQR)  
print(lwr\_bound, upr\_bound)

44.0 92.0

**Step 7:** Print Outliers  
r\_outliers = []  
for i in sorted\_rscore:  
if (i<lwr\_bound or i>upr\_bound):  
r\_outliers.append(i)  
print(r\_outliers)

[1, 16, 120]

### 3.2 Handling of Outliers:

For removing the outlier, one must follow the same process of removing an entry from the dataset using its exact position in the dataset because in all the above methods of detecting the outliers end result is the list of all those data items that satisfy the outlier definition according to the method used.

Below are some of the methods of treating the outliers

- Trimming/removing the outlier
- Quantile based flooring and capping
- Mean/Median imputation

- Trimming/removing the outlier:

In this technique, we remove the outliers from the dataset. Although it is not a good practice to follow.

```
new_df=df
for i in sample_outliers:
    new_df.drop(i,inplace=True)
new_df
```

|    | math score | reading score | writing score | placement score | placement offer | count |
|----|------------|---------------|---------------|-----------------|-----------------|-------|
| 1  | 71         | 61            | 85            | 91              |                 | 3     |
| 2  | 79         | 16            | 87            | 77              |                 | 2     |
| 3  | 61         | 77            | 74            | 76              |                 | 2     |
| 4  | 78         | 71            | 67            | 90              |                 | 3     |
| 5  | 73         | 68            | 90            | 80              |                 | 2     |
| 6  | 77         | 62            | 70            | 35              |                 | 2     |
| 7  | 74         | 45            | 80            | 12              |                 | 1     |
| 8  | 76         | 60            | 79            | 77              |                 | 2     |
| 9  | 75         | 65            | 85            | 87              |                 | 3     |
| 10 | 160        | 67            | 12            | 83              |                 | 2     |
| 11 | 79         | 72            | 88            | 180             |                 | 2     |
| 13 | 78         | 69            | 71            | 90              |                 | 3     |
| 14 | 75         | 1             | 71            | 81              |                 | 2     |
| 15 | 78         | 62            | 79            | 93              |                 | 3     |
| 18 | 75         | 62            | 86            | 87              |                 | 3     |

Here Sample\_outliers and instances with index 0, 12, 16 and 17 are deleted.

- Quantile based flooring and capping:

In this technique, the outlier is capped at a certain value above the 90th percentile value or floored at a factor below the 10th percentile value

```
df=pd.read_csv("/demo.csv")
df_stud=df
ninetieth_percentile = np.percentile(df_stud['math score'], 90)
b = np.where(df_stud['math score']>ninetieth_percentile,
ninetieth_percentile, df_stud['math score'])
print("New array:",b)
```

```
New array: [ 80.  71.  79.  61.  78.  73.  77.  74.  76.  75. 104.  79.  80.  78.
  75.  78.  86.  80.  75.  82.  69. 100.  72.  74.  75. 104.  72.  71.
 104.]
```

```
df_stud.insert(1,"m score",b,True)
```

```
df_stud
```

|   | math score | m score | reading score | writing score | placement score | placement offer count |
|---|------------|---------|---------------|---------------|-----------------|-----------------------|
| 0 | 80         | 80.0    | 68            | 70            | 89              | 3                     |
| 1 | 71         | 71.0    | 61            | 85            | 91              | 3                     |
| 2 | 79         | 79.0    | 16            | 87            | 77              | 2                     |
| 3 | 61         | 61.0    | 77            | 74            | 76              | 2                     |
| 4 | 78         | 78.0    | 71            | 67            | 90              | 3                     |
| 5 | 73         | 73.0    | 68            | 90            | 80              | 2                     |
| 6 | 77         | 77.0    | 62            | 70            | 35              | 2                     |
| 7 | 74         | 74.0    | 45            | 80            | 12              | 1                     |

### • Mean/Median imputation:

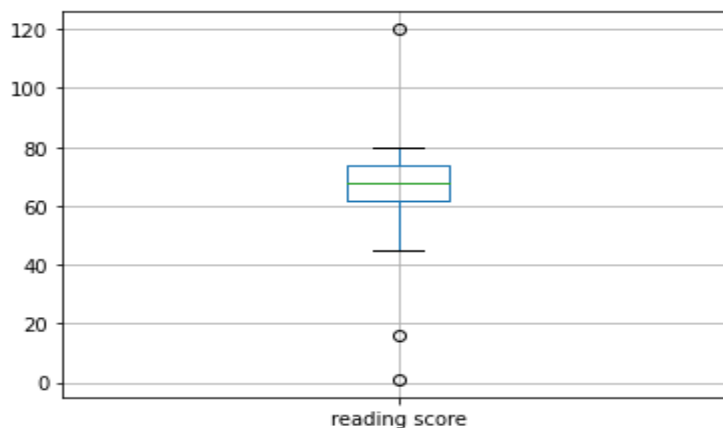
As the mean value is highly influenced by the outliers, it is advised to replace the outliers with the median value.

1. Plot the box plot for reading score

```
col = ['reading score']
```

```
df.boxplot(col)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f16affce150>
```



2. Outliers are seen in box plot.
3. Calculate the median of reading score by using sorted\_rscore  

```
median=np.median(sorted_rscore)
```

```
median
```
4. Replace the upper bound outliers using median value  

```
refined_df=df
```



```
refined_df['reading score'] = np.where(refined_df['reading score'] > upr_bound, median, refined_df['reading score'])
```

5. Display redefined\_df

|    | math score | m score | reading score | writing score | placement score | placement offer count |
|----|------------|---------|---------------|---------------|-----------------|-----------------------|
| 0  | 80         | 80.0    | 68.0          | 70            | 89              | 3                     |
| 1  | 71         | 71.0    | 61.0          | 85            | 91              | 3                     |
| 2  | 79         | 79.0    | 16.0          | 87            | 77              | 2                     |
| 3  | 61         | 61.0    | 77.0          | 74            | 76              | 2                     |
| 4  | 78         | 78.0    | 71.0          | 67            | 90              | 3                     |
| 5  | 73         | 73.0    | 68.0          | 90            | 80              | 2                     |
| 6  | 77         | 77.0    | 62.0          | 70            | 35              | 2                     |
| 7  | 74         | 74.0    | 45.0          | 80            | 12              | 1                     |
| 8  | 76         | 76.0    | 60.0          | 79            | 77              | 2                     |
| 9  | 75         | 75.0    | 65.0          | 85            | 87              | 3                     |
| 10 | 160        | 104.0   | 67.0          | 12            | 83              | 2                     |

6. Replace the lower bound outliers using median value

```
refined_df['reading score'] = np.where(refined_df['reading score'] < lwr_bound, median, refined_df['reading score'])
```

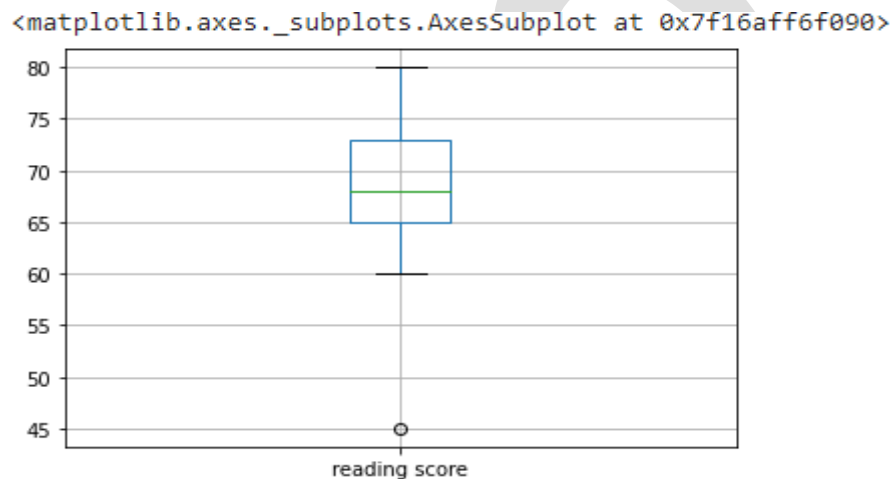
7. Display redefined\_df

|    | math score | m score | reading score | writing score | placement score | placement offer count |
|----|------------|---------|---------------|---------------|-----------------|-----------------------|
| 0  | 80         | 80.0    | 68.0          | 70            | 89              | 3                     |
| 1  | 71         | 71.0    | 61.0          | 85            | 91              | 3                     |
| 2  | 79         | 79.0    | 68.0          | 87            | 77              | 2                     |
| 3  | 61         | 61.0    | 77.0          | 74            | 76              | 2                     |
| 4  | 78         | 78.0    | 71.0          | 67            | 90              | 3                     |
| 5  | 73         | 73.0    | 68.0          | 90            | 80              | 2                     |
| 6  | 77         | 77.0    | 62.0          | 70            | 35              | 2                     |
| 7  | 74         | 74.0    | 45.0          | 80            | 12              | 1                     |
| 8  | 76         | 76.0    | 60.0          | 79            | 77              | 2                     |
| 9  | 75         | 75.0    | 65.0          | 85            | 87              | 3                     |
| 10 | 160        | 104.0   | 67.0          | 12            | 83              | 2                     |

8. Draw the box plot for redefined\_df

```
col = ['reading score']
```

```
refined_df.boxplot(col)
```



#### 4. Data Transformation for the purpose of :

Data transformation is the process of converting raw data into a format or structure that would be more suitable for model building and also data discovery in general. The process of data transformation can also be referred to as extract/transform/load (ETL). The extraction phase involves identifying and pulling data from the various source systems that create data and then moving the data to a single repository. Next, the raw data is cleansed, if needed. It's then transformed into a target format that can be fed into operational systems or into a data warehouse, a data lake or another repository for use in business intelligence and analytics applications. The data are transformed in ways that are ideal for mining the data. The data transformation involves steps that are.

- **Smoothing:** It is a process that is used to remove noise from the dataset using some algorithms. It allows for highlighting important features present in the dataset. It helps in predicting the patterns.
- **Aggregation:** Data collection or aggregation is the method of storing and presenting data in a summary format. The data may be obtained from multiple data sources to integrate these data sources into a data analysis description. This is a crucial step since the accuracy of data analysis insights is highly dependent on the quantity and quality of the data used.
- **Generalization:** It converts low-level data attributes to high-level data attributes using concept hierarchy. For Example Age initially in Numerical form (22, 25) is converted into categorical value (young, old).
- **Normalization:** Data normalization involves converting all data variables into a

given range. Some of the techniques that are used for accomplishing normalization are:

- **Min-max normalization:** This transforms the original data linearly.
- **Z-score normalization:** In z-score normalization (or zero-mean normalization) the values of an attribute (A), are normalized based on the mean of A and its standard deviation.
- **Normalization by decimal scaling:** It normalizes the values of an attribute by changing the position of their decimal points

- **Attribute or feature construction.**

- **New attributes constructed from the given ones:** Where new attributes are created & applied to assist the mining process from the given set of attributes. This simplifies the original data & makes the mining more efficient.

In this assignment , The purpose of this transformation should be one of the following reasons:

- To change the scale for better understanding (Attribute or feature construction)  
Here the Club\_Join\_Date is transferred to Duration.

### Algorithm:

**Step 1 :** Import pandas and numpy libraries

```
import pandas as pd
import numpy as np
```

**Step 2:** Load the dataset in dataframe object df

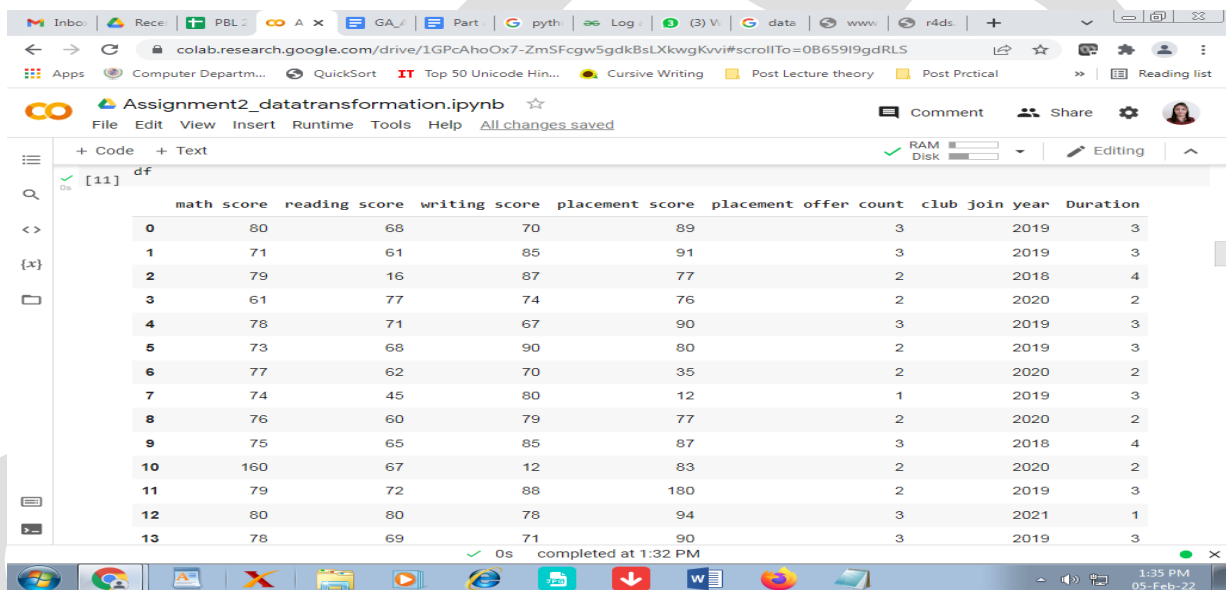
```
df=pd.read_csv("/content/demo.csv")
```

**Step 3:** Display the data frame

```
df
```

|    | math score | reading score | writing score | placement score | placement offer count | club join year |
|----|------------|---------------|---------------|-----------------|-----------------------|----------------|
| 0  | 80         | 68            | 70            | 89              | 3                     | 2019           |
| 1  | 71         | 61            | 85            | 91              | 3                     | 2019           |
| 2  | 79         | 16            | 87            | 77              | 2                     | 2018           |
| 3  | 61         | 77            | 74            | 76              | 2                     | 2020           |
| 4  | 78         | 71            | 67            | 90              | 3                     | 2019           |
| 5  | 73         | 68            | 90            | 80              | 2                     | 2019           |
| 6  | 77         | 62            | 70            | 35              | 2                     | 2020           |
| 7  | 74         | 45            | 80            | 12              | 1                     | 2019           |
| 8  | 76         | 60            | 79            | 77              | 2                     | 2020           |
| 9  | 75         | 65            | 85            | 87              | 3                     | 2018           |
| 10 | 160        | 67            | 12            | 83              | 2                     | 2020           |
| 11 | 79         | 72            | 88            | 180             | 2                     | 2019           |

Step 3: Change the scale of Joining year to duration.

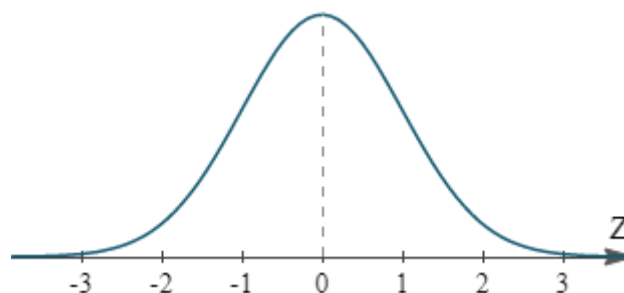


|    | math score | reading score | writing score | placement score | placement offer count | club join year | Duration |
|----|------------|---------------|---------------|-----------------|-----------------------|----------------|----------|
| 0  | 80         | 68            | 70            | 89              | 3                     | 2019           | 3        |
| 1  | 71         | 61            | 85            | 91              | 3                     | 2019           | 3        |
| 2  | 79         | 16            | 87            | 77              | 2                     | 2018           | 4        |
| 3  | 61         | 77            | 74            | 76              | 2                     | 2020           | 2        |
| 4  | 78         | 71            | 67            | 90              | 3                     | 2019           | 3        |
| 5  | 73         | 68            | 90            | 80              | 2                     | 2019           | 3        |
| 6  | 77         | 62            | 70            | 35              | 2                     | 2020           | 2        |
| 7  | 74         | 45            | 80            | 12              | 1                     | 2019           | 3        |
| 8  | 76         | 60            | 79            | 77              | 2                     | 2020           | 2        |
| 9  | 75         | 65            | 85            | 87              | 3                     | 2018           | 4        |
| 10 | 160        | 67            | 12            | 83              | 2                     | 2020           | 2        |
| 11 | 79         | 72            | 88            | 180             | 2                     | 2019           | 3        |
| 12 | 80         | 80            | 78            | 94              | 3                     | 2021           | 1        |
| 13 | 78         | 69            | 71            | 90              | 3                     | 2019           | 3        |

**b. To decrease the skewness and convert distribution into normal distribution  
(Normalization by decimal scaling)**

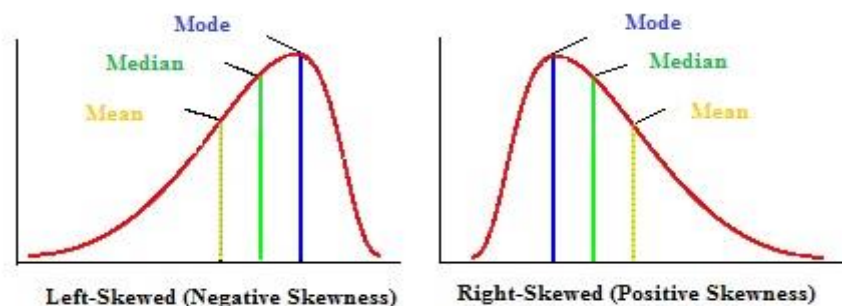
**Data Skewness:** It is asymmetry in a statistical distribution, in which the curve appears distorted or skewed either to the left or to the right. Skewness can be quantified to define the extent to which a distribution differs from a normal distribution.

**Normal Distribution:** In a normal distribution, the graph appears as a classical, symmetrical “bell-shaped curve.” The mean, or average, and the mode, or maximum point on the curve, are equal.



A **positively skewed distribution** means that the extreme data results are larger. This skews the data in that it brings the mean (average) up. The mean will be larger than the median in a Positively skewed distribution.

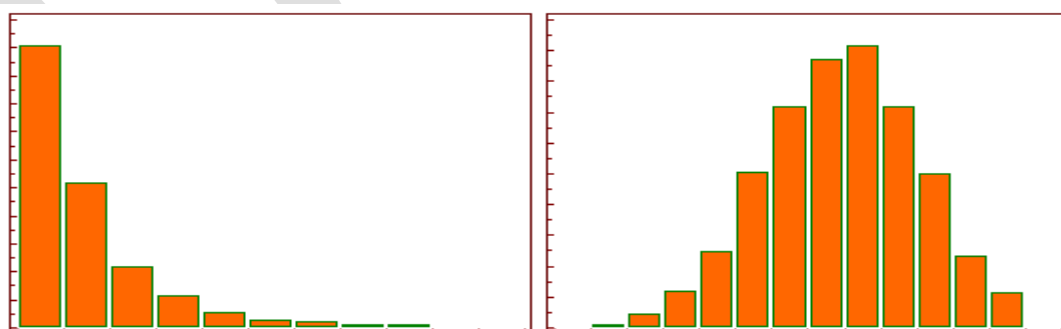
A **negatively skewed distribution** means the opposite: that the extreme data results are smaller. This means that the mean is brought down, and the median is larger than the mean in a negatively skewed distribution.



## Reducing

### Skewness :

A data transformation may be used to reduce skewness. A distribution that is symmetric or nearly so is often easier to handle and interpret than a skewed distribution. The logarithm,  $x$  to log base 10 of  $x$ , or  $x$  to log base  $e$  of  $x$  ( $\ln x$ ), or  $x$  to log base 2 of  $x$ , is a strong transformation with a major effect on distribution shape. It is commonly used for reducing right skewness and is often appropriate for measured variables. It can not be applied to zero or negative values.



### Algorithm:

**Step 1 :** Detecting outliers using Z-Score for the Math\_score variable and remove the outliers.

**Step 2:** Observe the histogram for math\_score variable.

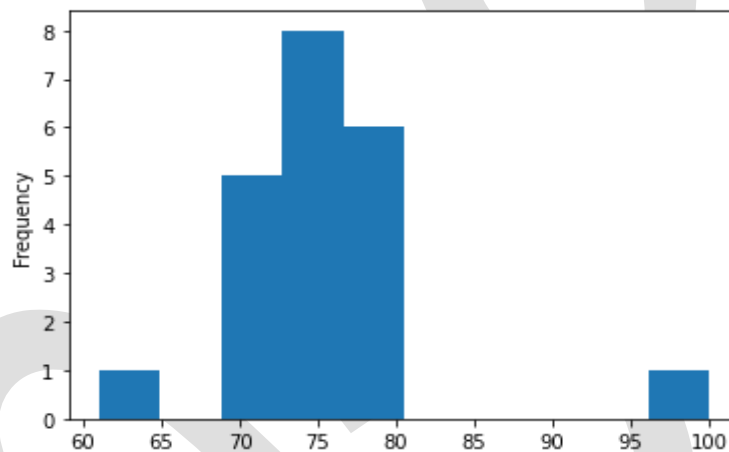
```
import matplotlib.pyplot as plt
new_df['math score'].plot(kind = 'hist')
```

**Step 3:** Convert the variables to logarithm at the scale 10.

```
df['log_math'] = np.log10(df['math score'])
```

**Step 4:** Observe the histogram for math\_score variable.

```
df['log_math'].plot(kind = 'hist')
```



It is observed that skewness is reduced at some level.

### Conclusion:

In this way we have explored the functions of the python library for Data Identifying and handling the outliers. Data Transformations Techniques are explored with the purpose of creating the new variable and reducing the skewness from datasets.

**Group A**  
**Assignment No: 3**

**Contents for Theory:**

- 1. Summary statistics**
- 2. Types of Variables**
- 3. Summary statistics of income grouped by the age groups**
- 4. Display basic statistical details on the iris dataset.**

**1. Summary statistics:****● What is Statistics?**

Statistics is the science of collecting data and analysing them to infer proportions (sample) that are representative of the population. In other words, statistics is interpreting data in order to make predictions for the population.

**Branches of Statistics:**

There are two branches of Statistics.

**DESCRIPTIVE STATISTICS** : Descriptive Statistics is a statistics or a measure that describes the data.

**INFERENTIAL STATISTICS** : Using a random sample of data taken from a population to describe and make inferences about the population is called Inferential Statistics.

**Descriptive Statistics :**

Descriptive Statistics is summarising the data at hand through certain numbers like mean, median etc. so as to make the understanding of the data easier. It does not involve any generalisation or inference beyond what is available. This means that the descriptive statistics are just the representation of the data (sample) available and not based on any theory of probability.

**Commonly Used Measures**

- 1. Measures of Central Tendency**
- 2. Measures of Dispersion (or Variability)**

**● Measures of Central Tendency :**

A Measure of Central Tendency is a one number summary of the data that typically describes the centre of the data. This one number summary is of three types.

**a. Mean :** Mean is defined as the ratio of the sum of all the observations in the data to the total number of observations. This is also known as Average. Thus mean is a number around which the entire data set is spread.

Consider the following data points.

17, 16, 21, 18, 15, 17, 21, 19, 11, 23

$$\text{Mean} = \frac{17+16+21+18+15+17+21+19+11+23}{10} = \frac{178}{10} = 17.8$$

**b. Median :** Median is the point which divides the entire data into two equal halves. One-half of the data is less than the median, and the other half is greater than the same. Median is calculated by first arranging the data in either ascending or descending order.

- If the number of observations is odd, the median is given by the middle observation in the sorted form.
- If the number of observations are even, median is given by the mean of the two middle observations in the sorted form.

An important point to note is that the order of the data (ascending or descending) does not affect the median.

To calculate Median, let's arrange the data in ascending order.

11, 15, 16, 17, 17, 18, 19, 21, 21, 23

Since the number of observations is even (10), median is given by the average of the two middle observations (5th and 6th here).

$$\text{Median} = \frac{5^{\text{th}} \text{ Obs} + 6^{\text{th}} \text{ Obs}}{2} = \frac{17 + 18}{2} = 17.5$$

**c. Mode :** Mode is the number which has the maximum frequency in the entire data set, or in other words, mode is the number that appears the maximum number of times. A data can have one or more than one mode.

- If there is only one number that appears the maximum number of times, the data has one mode, and is called Uni-modal.
- If there are two numbers that appear the maximum number of times, the



data has two modes, and is called Bi-modal.

- If there are more than two numbers that appear the maximum number of times, the data has more than two modes, and is called Multi-modal.

Consider the following data points.

17, 16, 21, 18, 15, 17, 21, 19, 11, 23

Mode is given by the number that occurs the maximum number of times.

Here, 17 and 21 both occur twice. Hence, this is a Bimodal data and the modes are 17 and 21.

### • Measures of Dispersion (or Variability) :

Measures of Dispersion describes the spread of the data around the central value (or the Measures of Central Tendency)

**1. Absolute Deviation from Mean** — The Absolute Deviation from Mean, also called Mean Absolute Deviation (MAD), describes the variation in the data set, in the sense that it tells the average absolute distance of each data point in the set. It is calculated as

$$\text{Mean Absolute Deviation} = \frac{1}{N} \sum_{i=1}^N |X_i - \bar{X}|$$

**2. Variance** — Variance measures how far are data points spread out from the mean. A high variance indicates that data points are spread widely and a small variance indicates that the data points are closer to the mean of the data set. It is calculated as

$$\text{Variance} = \frac{1}{N} \sum_{i=1}^N (X_i - \bar{X})^2$$

**3. Standard Deviation** — The square root of Variance is called the Standard Deviation. It is calculated as

$$\text{Std Deviation} = \sqrt{\text{Variance}} = \sqrt{\frac{1}{N} \sum_{i=1}^N (X_i - \bar{X})^2}$$

**4. Range** — Range is the difference between the Maximum value and the Minimum value in the data set. It is given as

$$\text{Range} = \text{Maximum} - \text{Minimum}$$

**5. Quartiles** — Quartiles are the points in the data set that divides the data set into four equal parts. Q1, Q2 and Q3 are the first, second and third quartile of the data set.

**6. Skewness** — The measure of asymmetry in a probability distribution is defined by Skewness. It can either be positive, negative or undefined.

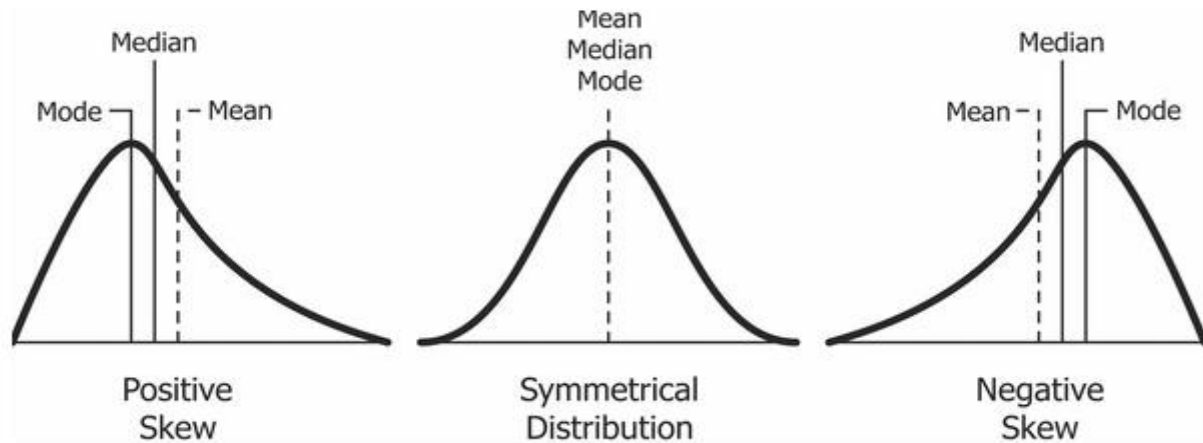
$$\text{Skewness} = \frac{3 ( \text{Mean} - \text{Median} )}{\text{Std Deviation}}$$

**Positive Skew** — This is the case when the tail on the right side of the curve is bigger than that on the left side. For these distributions, mean is greater than the mode.

**Negative Skew** — This is the case when the tail on the left side of the curve is bigger than that on the right side. For these distributions, mean is smaller than the mode.

The most commonly used method of calculating Skewness is

If the skewness is zero, the distribution is symmetrical. If it is negative, the distribution is Negatively Skewed and if it is positive, it is Positively Skewed.

**Python Code:**

**1. Mean :** To find mean of all columns

**Syntax:**

```
df.mean()
```

**Output:**

```
CustomerID      100.50
Age              38.85
Annual Income (k$)  60.56
Spending Score (1-100)  50.20
dtype: float64
```

**To find mean of specific column**

**Syntax:**

```
df.loc[:, 'Age'].mean()
```

**Output:** 38.85

**To find mean row wise**

**Syntax:** df.mean(axis=1) [0:4]

```
0      18.50
1      29.75
2      11.25
3      30.00
dtype: float64
```

**Output:**

**2. Median :** To find median of all columns**Syntax:** df.median()**Output:**

```
CustomerID    100.5
Age           36.0
Annual Income (k$)  61.5
Spending Score (1-100)  50.0
dtype: float64
```

To find median of specific column

**Syntax:** df.loc[:, 'Age'].median()**Output:** 36.0

To find median row wise

**Syntax:** df.median(axis=1)[0:4]**Output:**

```
0    17.0
1    18.0
2    11.0
3    19.5
dtype: float64
```

**3. Mode**

To find mode of all columns

**Syntax:**

df.mode()

**Output:**

|     | CustomerID | Genre  | Age  | Annual Income (k\$) | Spending Score (1-100) |
|-----|------------|--------|------|---------------------|------------------------|
| 0   | 1          | Female | 32.0 | 54.0                | 42.0                   |
| 1   | 2          | NaN    | NaN  | 78.0                | NaN                    |
| 2   | 3          | NaN    | NaN  | NaN                 | NaN                    |
| 3   | 4          | NaN    | NaN  | NaN                 | NaN                    |
| 4   | 5          | NaN    | NaN  | NaN                 | NaN                    |
| ... | ...        | ...    | ...  | ...                 | ...                    |
| 195 | 196        | NaN    | NaN  | NaN                 | NaN                    |
| 196 | 197        | NaN    | NaN  | NaN                 | NaN                    |
| 197 | 198        | NaN    | NaN  | NaN                 | NaN                    |
| 198 | 199        | NaN    | NaN  | NaN                 | NaN                    |
| 199 | 200        | NaN    | NaN  | NaN                 | NaN                    |

200 rows × 5 columns

In the Genre Column mode is Female, for column Age mode is 32 etc. If a particular column does not have mode all the values will be displayed in the column .

To find the mode of a specific column.

**Syntax:** df.loc[:, 'Age'].mode()

**Output:** 32

#### 4. Minimum

To find minimum of all columns

**Syntax:** df.min()

**Output:**

```
CustomerID      1
Genre           Female
Age             18
Annual Income (k$)  15
Spending Score (1-100)  1
dtype: object
```

To find minimum of Specific column

**Syntax:** df.loc[:, 'Age'].min(skipna = False)

**Output:** 18

#### 5. Maximum

To find Maximum of all columns

**Syntax:** df.max()

**Output:**

```
CustomerID      200
Genre           Male
Age             70
Annual Income (k$)  137
Spending Score (1-100)  99
dtype: object
```

To find Maximum of Specific

column

**Syntax:** `df.loc[:, 'Age'].max(skipna = False)`

**Output:** 70

## 6. Standard Deviation

To find Standard Deviation of all columns

**Syntax:** `df.std()`

**Output:**

```
CustomerID    57.879185
Age           13.969007
Annual Income (k$)  26.264721
Spending Score (1-100)  25.823522
dtype: float64
```

To find Standard Deviation of

specific column

**Syntax:**

`df.loc[:, 'Age'].std()`

**Output:** 13.969007331558883

To find Standard Deviation row wise

**Syntax:** `df.std(axis=1)[0:4]`

```
0    15.695010
1    35.074920
2     8.057088
3    32.300671
dtype: float64
```

Output:

## 2. Types of Variables:

A variable is a characteristic that can be measured and that can assume different values. Height, age, income, province or country of birth, grades obtained at school and type of housing are all examples of variables.

Variables may be classified into two main categories:

- Categorical and
- Numeric.

Each category is then classified in two subcategories: nominal or ordinal for categorical variables, discrete or continuous for numeric variables.

### Categorical variables

A categorical variable (also called qualitative variable) refers to a characteristic that can't be quantifiable.

Categorical variables can be either nominal or ordinal.

- **Nominal Variable**

A nominal variable is one that describes a name, label or category without natural order. In the given table, the variable “mode of transportation for travel to work” is also nominal.

- **Ordinal Variable**

An ordinal variable is a variable whose values are defined by an order relation between the different categories. In the following table, the variable “behaviour” is ordinal because the category “Excellent” is better than the category “Very good,” which is better than the category “Good,” etc. There is some natural ordering, but it is limited since we do not know by how much “Excellent” behaviour is better than “Very good” behaviour.

- **Numerical Variables**

A numeric variable (also called quantitative variable) is a quantifiable characteristic whose values are numbers (except numbers which are codes standing up for categories). Numeric variables may be either continuous or discrete .

- **Continuous variables**

A variable is said to be continuous if it can assume an infinite number of real values within a given interval.

For instance, consider the height of a student. The height can't take any values. It can't be negative and it can't be higher than three metres. But between 0 and 3, the number of possible values is theoretically infinite. A student may be 1.6321748755 ... metres tall.

- **Discrete variables**

As opposed to a continuous variable, a discrete variable can assume only a finite number of real values within a given interval.

An example of a discrete variable would be the score given by a judge to a gymnast in competition: the range is 0 to 10 and the score is always given to one decimal (e.g. a score of 8.5)

### **3. Summary statistics of income grouped by the age groups**

Problem Statement: For example, if your categorical variable is age groups and

quantitative variable is income, then provide summary statistics of income grouped by the age

groups. Create a list that contains a numeric value for each response to the categorical variable.

Categorical Variable: Genre

Quantitative Variable : Age

**Syntax:** df.groupby(['Genre'])['Age'].mean()

**Output:**

```
Genre
Female    38.098214
Male      39.806818
Name: Age, dtype: float64
```

**Categorical Variable:** Genre

**Quantitative Variable :** Income

**Syntax:**

```
df_u=df.rename(columns= {'Annual Income
k$':'Income'},inplace=False)
```

```
(df_u.groupby(['Genre']).Income.mean())
```

**Output:**

```
Genre
Female    59.250000
Male      62.227273
Name: Income, dtype: float64
```



To create a list that contains a numeric value for each response to the categorical variable.

```
from sklearn import preprocessing
enc = preprocessing.OneHotEncoder()
enc_df = pd.DataFrame(enc.fit_transform(df[['Genre']]).toarray())
enc_df
```

|     | 0   | 1   |
|-----|-----|-----|
| 0   | 0.0 | 1.0 |
| 1   | 0.0 | 1.0 |
| 2   | 1.0 | 0.0 |
| 3   | 1.0 | 0.0 |
| 4   | 1.0 | 0.0 |
| ... | ... | ... |
| 195 | 1.0 | 0.0 |
| 196 | 1.0 | 0.0 |
| 197 | 0.0 | 1.0 |
| 198 | 0.0 | 1.0 |
| 199 | 0.0 | 1.0 |

200 rows × 2 columns

To concat numerical list to dataframe

```
df_encode = df_u.join(enc_df)
df_encode
```

|     | CustomerID | Genre  | Age | Income | Spending Score (1-100) | 0   | 1   |
|-----|------------|--------|-----|--------|------------------------|-----|-----|
| 0   | 1          | Male   | 19  | 15     | 39                     | 0.0 | 1.0 |
| 1   | 2          | Male   | 21  | 15     | 81                     | 0.0 | 1.0 |
| 2   | 3          | Female | 20  | 16     | 6                      | 1.0 | 0.0 |
| 3   | 4          | Female | 23  | 16     | 77                     | 1.0 | 0.0 |
| 4   | 5          | Female | 31  | 17     | 40                     | 1.0 | 0.0 |
| ... | ...        | ...    | ... | ...    | ...                    | ... | ... |
| 195 | 196        | Female | 35  | 120    | 79                     | 1.0 | 0.0 |
| 196 | 197        | Female | 45  | 126    | 28                     | 1.0 | 0.0 |
| 197 | 198        | Male   | 32  | 126    | 74                     | 0.0 | 1.0 |
| 198 | 199        | Male   | 32  | 137    | 18                     | 0.0 | 1.0 |
| 199 | 200        | Male   | 30  | 137    | 83                     | 0.0 | 1.0 |

Prepare 200 rows × 7 columns

neering

**4. Display basic statistical details on the iris dataset.****Algorithm:**

1. Import Pandas Library

2. The dataset is downloaded from UCI repository.

**csv\_url = <https://archive.ics.uci.edu/ml/machine-learning-databases/iris/iris.data>**

3. Assign Column names

**col\_names = ['Sepal\_Length', 'Sepal\_Width', 'Petal\_Length', 'Petal\_Width', 'Species']**

4. Load Iris.csv into a Pandas data frame

**iris = pd.read\_csv(csv\_url, names = col\_names)**

5. Load all rows with Iris-setosa species in variable irisSet

**irisSet = (iris['Species']== 'Iris-setosa')**

6. To display basic statistical details like percentile, mean, standard deviation etc. for Iris-setosa use describe

**print('Iris-setosa')**

**print(iris[irisSet].describe())**

7. Load all rows with Iris-versicolor species in variable irisVer

**irisVer = (iris['Species']== 'Iris-versicolor')**

8. To display basic statistical details like percentile, mean, standard deviation etc. for Iris-versicolor use describe

**print('Iris-versicolor')**

**print(iris[irisVer].describe())**

9. Load all rows with Iris-virginica species in variable irisVir

**irisVir = (iris['Species']== 'Iris-virginica')**

10. To display basic statistical details like percentile, mean, standard deviation etc. for

Iris-virginica use describe

```
print('Iris-virginica')
print(iris[irisVir].describe())
```

```
[1] Iris-setosa
      Sepal_Length  Sepal_Width  Petal_Length  Petal_Width
count      50.00000      50.000000      50.000000      50.00000
mean        5.00600       3.418000       1.464000       0.24400
std         0.35249       0.381024       0.173511       0.10721
min         4.30000       2.300000       1.000000       0.10000
25%         4.80000       3.125000       1.400000       0.20000
50%         5.00000       3.400000       1.500000       0.20000
75%         5.20000       3.675000       1.575000       0.30000
max         5.80000       4.400000       1.900000       0.60000
Iris-versicolor
      Sepal_Length  Sepal_Width  Petal_Length  Petal_Width
count      50.000000      50.000000      50.000000      50.000000
mean        5.936000      2.770000      4.260000      1.326000
std         0.516171      0.313798      0.469911      0.197753
min         4.900000      2.000000      3.000000      1.000000
25%         5.600000      2.525000      4.000000      1.200000
50%         5.900000      2.800000      4.350000      1.300000
75%         6.300000      3.000000      4.600000      1.500000
max         7.000000      3.400000      5.100000      1.800000
Iris-virginica
      Sepal_Length  Sepal_Width  Petal_Length  Petal_Width
count      50.00000      50.000000      50.000000      50.00000
mean        6.58800       2.974000      5.552000      2.02600
std         0.63588       0.322497      0.551895      0.27465
min         4.90000       2.200000      4.500000      1.40000
25%         6.22500       2.800000      5.100000      1.80000
50%         6.50000       3.000000      5.550000      2.00000
75%         6.90000       3.175000      5.875000      2.30000
max         7.90000       3.800000      6.900000      2.50000
```

### Conclusion:

Descriptive statistics summarises or describes the characteristics of a data set. Descriptive statistics consists of two basic categories of measures:

- measures of central tendency and
- measures of variability (or spread).

Measures of central tendency describe the centre of a data set. It includes the mean, median, and mode.

Measures of variability or spread describe the dispersion of data within the set and it includes standard deviation, variance, minimum and maximum variables.

**Group A**  
**Assignment No: 4**

---

**Contents for Theory:**

- 1. Linear Regression : Univariate and Multivariate**
- 2. Least Square Method for Linear Regression**
- 3. Measuring Performance of Linear Regression**
- 4. Example of Linear Regression**
- 5. Training data set and Testing data set**

**1. Linear Regression:**

It is a machine learning algorithm based on supervised learning. It targets prediction values on the basis of independent variables.

- It is preferred to find out the relationship between forecasting and variables.
- A linear relationship between a dependent variable (X) is continuous; while independent variable(Y) relationship may be continuous or discrete. A linear relationship should be available in between predictor and target variable so known as Linear Regression.

- Linear regression is popular because the cost function is Mean Squared Error (MSE) which is equal to the average squared difference between an observation's actual and predicted values.

- It is shown as an equation of line like :

$$Y = m \cdot X + b + e$$

Where : b is intercepted, m is slope of the line and e is error term.

This equation can be used to predict the value of target variable Y based on given predictor variable(s) X, as shown in Fig. 1.

**MultiVariate Regression** :It concerns the study of two or more predictor variables. Usually a transformation of the original features into polynomial features from a given degree is preferred and further Linear Regression is applied on it.

- A simple linear model  $Y = a + bX$  in original feature will be transformed into polynomial feature is transformed and further a linear regression applied to it and it will be something like  $Y = a + bX + cX^2$
- If a high degree value is used in transformation the curve becomes over-fitted as it captures the noise from data as well.

## 2. Least Square Method for Linear Regression :

- Linear Regression involves establishing linear relationships between dependent and independent variables. Such a relationship is portrayed in the form of an equation also known as the linear model.
- A simple linear model is the one which involves only one dependent and one independent variable. Regression Models are usually denoted in Matrix Notations.
- However, for a simple univariate linear model, it can be denoted by the regression equation

$$\hat{y} = \beta_0 + \beta_1 x \quad (1)$$

where  $y$  is the dependent or the response variable  
 $x$  is the independent or the input variable

$\beta_0$  is the value of  $y$  when  $x=0$  or the  $y$  intercept  
 $\beta_1$  is the value of slope of the line  $\varepsilon$  is the error or the noise

- This linear equation represents a line also known as the ‘regression line’. The least square estimation technique is one of the basic techniques used to guess the values of the parameters and based on a sample set.
- This technique estimates parameters  $\beta_0$  and  $\beta_1$  and by trying to minimise the square of errors at all the points in the sample set. The error is the deviation of the actual sample data point from the regression line. The technique can be represented by the equation.

### 3. Measuring Performance of Linear Regression :

#### Mean Square Error:

The Mean squared error (MSE) represents the error of the estimator or predictive model created based on the given set of observations in the sample. Two or more regression models created using a given sample data can be compared based on their MSE. The lesser the MSE, the better the regression model is.

Mathematically, the MSE can be calculated as the average sum of the squared difference between the actual value and the predicted or estimated value represented by the regression model (line or plane).

**An MSE of zero (0) represents the fact that the predictor is a perfect predictor.**

**RMSE:**

$$\text{RMSE} = \sqrt{\sum_{i=1}^n \frac{1}{n} (\hat{y}_i - y_i)^2}$$

#### 4. Example of Linear Regression :

Consider following data for 5 students.

Each  $X_i$  ( $i = 1$  to 5) represents the score of  $i$ th student in standard X and corresponding

$Y_i$  ( $i = 1$  to 5) represents the score of  $i$ th student in standard XII.

(i) Linear regression equation best predicts standard XIIth score

(ii) Interpretation for the equation of Linear Regression

(iii) If a student's score is 80 in std X, then what is his expected score in XII standard?

| Student | Score in X standard ( $X_i$ ) | Score in XII standard ( $Y_i$ ) |
|---------|-------------------------------|---------------------------------|
| 1       | 95                            | 85                              |
| 2       | 85                            | 95                              |
| 3       | 80                            | 70                              |
| 4       | 70                            | 65                              |
| 5       | 60                            | 70                              |

| x  | y  | $x - \bar{x}$ | $y - \bar{y}$ | $(x - \bar{x})^2$ | $(x - \bar{x})(y - \bar{y})$ |
|----|----|---------------|---------------|-------------------|------------------------------|
| 95 | 85 | 17            | 8             | 289               | 136                          |
| 85 | 95 | 7             | 18            | 49                | 126                          |
| 80 | 70 | 2             | -7            | 4                 | -14                          |
| 70 | 65 | -8            | -12           | 64                | 96                           |
| 60 | 70 | -18           | -7            | 324               | 126                          |

|              |              |  |  |                              |   |
|--------------|--------------|--|--|------------------------------|---|
| $\bar{x}=78$ | $\bar{y}=77$ |  |  | $\sum (x - \bar{x})^2 = 730$ | $\sum (x - \bar{x})(y - \bar{y}) = 470$ |
|--------------|--------------|--|--|------------------------------|---|

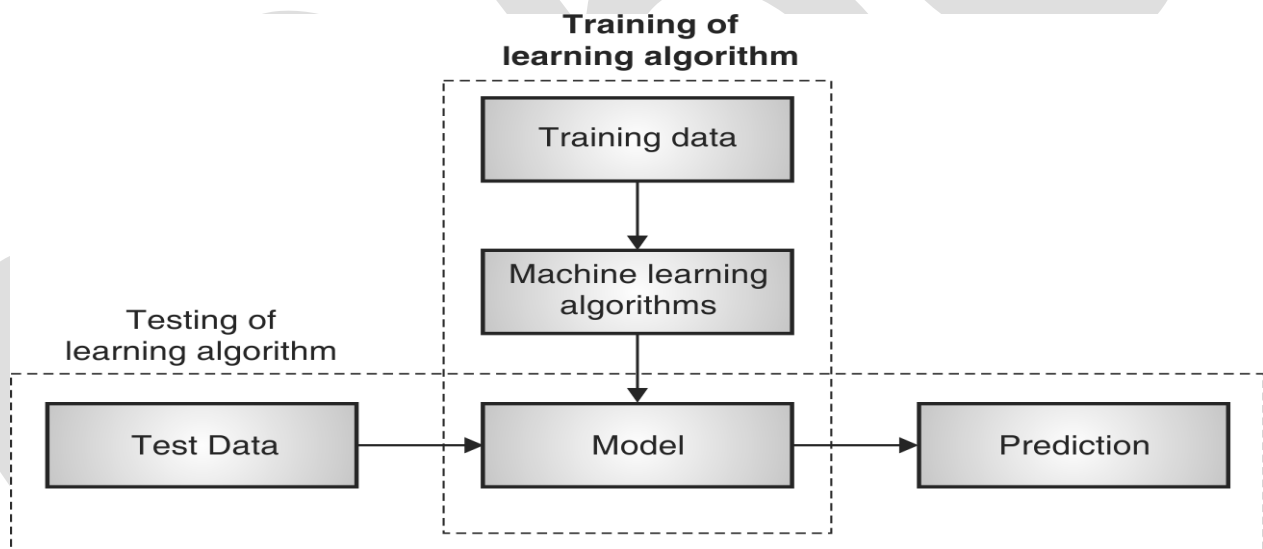
## 5. Training data set and Testing data set

- Machine Learning algorithm has two phases

1. Training and 2. Testing.

- The input of the training phase is training data, which is passed to any machine learning algorithm and machine learning model is generated as output of the training phase.

- The input of the testing phase is test data, which is passed to the machine learning model and prediction is done to observe the correctness of mode.



### (a) Training Phase

- Training dataset is provided as input to this phase.
- Training dataset is a dataset having attributes and class labels and used for training Machine Learning algorithms to prepare models.
- Machines can learn when they observe enough relevant data. Using this one can model algorithms to find relationships, detect patterns, understand complex problems and make decisions.
- Training error is the error that occurs by applying the model to the same data from which



the model is trained.

- In a simple way the actual output of training data and predicted output of the model does not match the training error  $E_{in}$  is said to have occurred.
- Training error is much easier to compute.

**(b) Testing Phase :**

- Testing dataset is provided as input to this phase.
- Test dataset is a dataset for which class label is unknown. It is tested using model
- A test dataset used for assessment of the finally chosen model.
- Training and Testing dataset are completely different.
- Testing error is the error that occurs by assessing the model by providing the unknown data to the model.
- In a simple way the actual output of testing data and predicted output of the model does not match the testing error  $E_{out}$  is said to have occurred.
- $E_{out}$  is generally observed larger than  $E_{in}$ .

**(c) Generalization :**

- Generalization is the prediction of the future based on the past system.
- It needs to generalize beyond the training data to some future data that it might not have seen yet.
- The ultimate aim of the machine learning model is to minimize the generalization error.
- The generalization error is essentially the average error for data the model has never seen.
- In general, the dataset is divided into two partition training and test sets.

**Group A**  
**Assignment No: 5**

**Contents for Theory:**

- 1. Logistic Regression**
- 2. Differentiate between Linear and Logistic Regression**
- 3. Sigmoid Function**
- 4. Types of Logistic Regression**
- 5. Confusion Matrix Evaluation Metrics**

**1. Logistic Regression:**

Classification techniques are an essential part of machine learning and data mining applications. Approximately 70% of problems in Data Science are classification problems. There are lots of classification problems that are available, but logistic regression is common and is a useful regression method for solving the binary classification problem.

Logistic Regression can be used for various classification problems such as spam detection, Diabetes prediction, if a given customer will purchase a particular product or will they churn another competitor, whether the user will click on a given advertisement link or not, and many more examples are in the bucket.

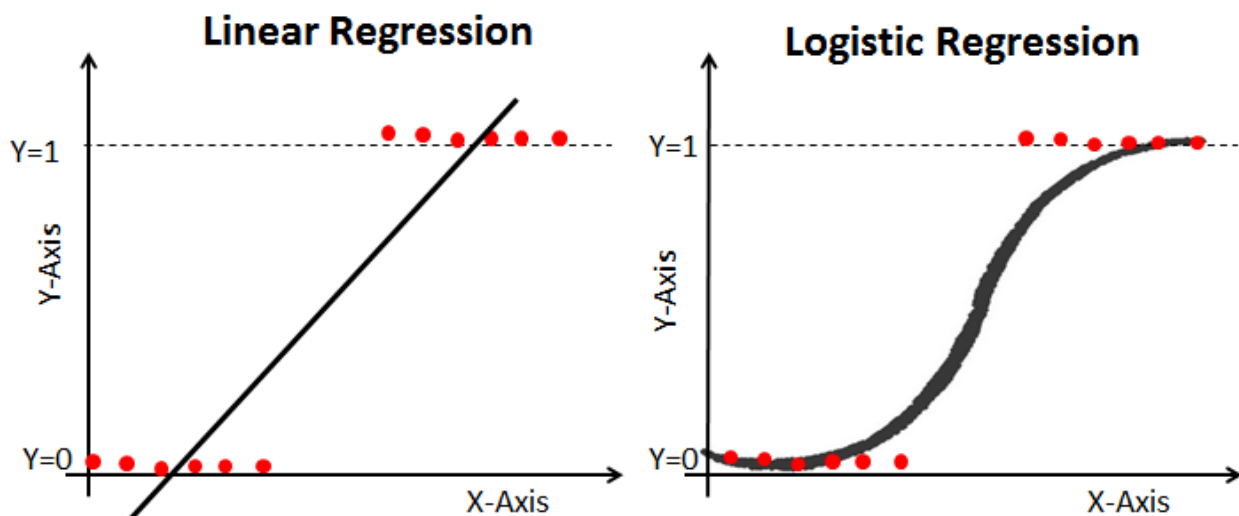
**Linear Regression Equation:**

$$y = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \dots + \beta_n X_n$$

Where, y is a dependent variable and x1, x2 ... and Xn are explanatory variables.

**2. Differentiate between Linear and Logistic Regression :**

Linear regression gives you a continuous output, but logistic regression provides a constant output. An example of the continuous output is house price and stock price. Example's of the discrete output is predicting whether a patient has cancer or not, predicting whether the customer will churn. Linear regression is estimated using Ordinary Least Squares (OLS) while logistic regression is estimated using Maximum Likelihood Estimation (MLE) approach.



#### 4. Types of LogisticRegression :

**Binary Logistic Regression:** The target variable has only two possible outcomes such as Spam or Not Spam, Cancer or No Cancer.

**Multinomial Logistic Regression:** The target variable has three or more nominal categories such as predicting the type of Wine.

**Ordinal Logistic Regression:** the target variable has three or more ordinal categories such as restaurant or product rating from 1 to 5.

#### 5. Confusion Matrix Evaluation Metrics :

Contingency table or Confusion matrix is often used to measure the performance of classifiers. A confusion matrix contains information about actual and predicted classifications done by a classification system. Performance of such systems is commonly evaluated using the data in the matrix.

The following table shows the confusion matrix for a two class classifier.

|        |   | predicted |    |   |
|--------|---|-----------|----|---|
|        | n | P         | N  |   |
| actual | P | TP        | FN | P |
|        | N | FP        | TN | N |

*Confusion matrix*

Here each row indicates the actual classes recorded in the test data set and the each column indicates the classes as predicted by the classifier.

Numbers on the descending diagonal indicate correct predictions, while the ascending diagonal concerns prediction errors.

Some Important measures derived from confusion matrix are:

- **Number of positive (Pos)** : Total number instances which are labelled as positive in a given dataset.
- **Number of negative (Neg)** : Total number instances which are labelled as negative in a given dataset.
- **Number of True Positive (TP)** : Number of instances which are actually labelled as positive and the predicted class by classifier is also positive.
- **Number of True Negative (TN)** : Number of instances which are actually labelled as negative and the predicted class by classifier is also negative.

- **Number of False Positive (FP)** : Number of instances which are actually labelled as negative and the predicted class by classifier is positive.
- **Number of False Negative (FN)**: Number of instances which are actually labelled as positive and the class predicted by the classifier is negative.
- **Accuracy** : Accuracy is calculated as the number of correctly classified instances divided by total number of instances. The ideal value of accuracy is 1, and the worst is 0. It is also calculated as the sum of true positive and true negative (TP + TN) divided by the total number of instances.

$$\text{Acc} = \text{TP} + \text{TN} / \text{TP} + \text{FP} + \text{TN} + \text{FN} = \text{TP} + \text{TN} / \text{Pos} + \text{Neg}$$

- **Error Rate**: Error Rate is calculated as the number of incorrectly classified instances divided by total number of instances. The ideal value of accuracy is 0, and the worst is 1. It is also calculated as the sum of false positive and false negative (FP + FN) divided by the total number of instances.

$$\text{Err} = \text{FP} + \text{FN} / \text{TP} + \text{FP} + \text{TN} + \text{FN} = \text{FP} + \text{FN} / \text{pos} + \text{neg}$$

or

$$\text{err} = 1 - \text{acc}$$

- **Precision**: It is calculated as the number of correctly classified positive instances divided by the total number of instances which are predicted positive. It is also called confidence value. The ideal value is 1, whereas the worst is 0.

$$\text{precision} = \text{TP} / \text{TP} + \text{FP}$$

- **Recall**: It is calculated as the number of correctly classified positive instances divided by the total number of positive instances. It is also called recall or sensitivity. The ideal value of sensitivity is 1, whereas the worst is 0. It is calculated as the number of correctly classified positive instances divided by the total number of positive instances.

$$\text{Recall} = \text{TP} / \text{TP} + \text{FN}$$

### Conclusion:

In this way we have done data analysis using logistic regression for Social Media Adv. and evaluated the performance of model.

**Group A**  
**Assignment No: 6**

**Contents for Theory:**

- 1. Concepts used in Naïve Bayes classifier**
- 2. Naive Bayes Example**
- 3. Confusion Matrix Evaluation Metrics**

## 1. Concepts used in Naïve Bayes classifier :

Naïve Bayes Classifier can be used for Classification of categorical data.

- Let there be a 'j' number of classes.  $C = \{1, 2, \dots, j\}$
- Let, input observation is specified by 'P' features. Therefore input observation x is given,  $x = \{F_1, F_2, \dots, F_p\}$
- The Naïve Bayes classifier depends on Bayes' rule from probability theory.

- **Prior probabilities:** Probabilities which are calculated for some event based on no other information are called Prior probabilities.

For example,  $P(A)$ ,  $P(B)$ ,  $P(C)$  are prior probabilities because while calculating  $P(A)$ , occurrences of event B or C are not concerned i.e. no information about occurrence of any other event is used.

**Conditional Probabilities:**

$$P\left(\frac{A}{B}\right) = \frac{P(A \cap B)}{P(B)} \quad \text{if } P(B) \neq 0 \quad \dots \dots (1)$$

$$P\left(\frac{B}{A}\right) = \frac{P(B \cap A)}{P(A)} \quad \dots \dots (2)$$

From equation (1) and (2) ,

$$P(A \cap B) = P\left(\frac{A}{B}\right) \cdot P(B) = P\left(\frac{B}{A}\right) \cdot P(A)$$

$$\therefore P\left(\frac{A}{B}\right) = \frac{P\left(\frac{B}{A}\right) \cdot P(A)}{P(B)}$$

Is called the Bayes

Rule.

### Conditional Probability :

Here, we are predicting the probability of class1 and class2 based on the given condition. If I try to write the same formula in terms of classes and features, we will get the following equation

$$P(C_k | X) = \frac{P(X | C_k) \cdot P(C_k)}{P(X)}$$

Now we have two classes and four features, so if we write this formula for class C1, it will be something like this.

$$P(C_1 | x_1 \cap x_2 \cap x_3 \cap x_4) = \frac{P(x_1 \cap x_2 \cap x_3 \cap x_4 | C_1) \cdot P(C_1)}{P(x_1 \cap x_2 \cap x_3 \cap x_4)}$$

Here, we replaced Ck with C1 and X with the intersection of X1, X2, X3, X4. You might have a question, It's because we are taking the situation when all these features are present at the same time.

The Naive Bayes algorithm assumes that all the features are independent of each other or in other words all the features are unrelated. With that assumption, we can further simplify the above formula and write it in this form

$$P(C_1 | x_1 \cap x_2 \cap x_3 \cap x_4) = \frac{P(x_1 | C_1) * P(x_2 | C_1) * P(x_3 | C_1) * P(x_4 | C_1) * P(C_1)}{P(x_1) * P(x_2) * P(x_3) * P(x_4)}$$

**Conclusion:**

In this way we have done data analysis using Naive Bayes Algorithm for Iris dataset and evaluated the performance of the model.

---

**Group A**  
**Assignment No: 7**

---

**Contents for Theory:**

- 1. Basic concepts of Text Analytics**
- 2. Text Analysis Operations using natural language toolkit**
- 3. Text Analysis Model using TF-IDF.**
- 4. Bag of Words (BoW)**

**1. Basic concepts of Text Analytics :**

One of the most frequent types of day-to-day conversation is text communication. In our everyday routine, we chat, message, tweet, share status, email, create blogs, and offer opinions and criticism. All of these actions lead to a substantial amount of unstructured text being produced. It is critical to examine huge amounts of data in this sector of the online world and social media to determine people's opinions.

Text mining is also referred to as text analytics. Text mining is a process of exploring sizable textual data and finding patterns. Text Mining processes the text itself, while NLP processes with the underlying metadata. Finding frequency counts of words, length of the sentence, presence/absence of specific words is known as text mining. Natural language processing is one of the components of text mining. NLP helps identify sentiment, finding entities in the sentence, and category of blog/article. Text mining is preprocessed data for text analytics. In Text Analytics, statistical and machine learning algorithms are used to classify information.

**2. Text Analysis Operations using natural language toolkit :**

NLTK(natural language toolkit) is a leading platform for building Python programs to



work with human language data. It provides easy-to-use interfaces and lexical resources such as WordNet, along with a suite of text processing libraries for classification, tokenization, stemming, tagging, parsing, and semantic reasoning and many more. Analysing movie reviews is one of the classic examples to demonstrate a simple NLP Bag-of-words model, on movie reviews.

### **2.1.Tokenization:**

Tokenization is the first step in text analytics. The process of breaking down a text paragraph into smaller chunks such as words or sentences is called Tokenization. Token is a single entity that is the building blocks for a sentence or paragraph.

- **Sentence tokenization** : split a paragraph into list of sentences using **sent\_tokenize() method**

- **Word tokenization** : split a sentence into list of words using **word\_tokenize() method**

### **2.2.Stop words removal :**

Stopwords considered as noise in the text. Text may contain stop words such as is, am, are, this, a, an, the, etc. In NLTK for removing stopwords, you need to create a list of stopwords and filter out your list of tokens from these words.

### **2.3.Stemming and Lemmatization :**

Stemming is a normalization technique where lists of tokenized words are converted into shortened root words to remove redundancy. Stemming is the process of reducing inflected (or sometimes derived) words to their word stem, base or root form.

A computer program that stems word may be called a stemmer.

**Lemmatization** in NLTK is the algorithmic process of finding the lemma of a word depending on its meaning and context. Lemmatization usually refers to the morphological analysis of words, which aims to remove inflectional endings. It helps in returning the base or dictionary form of a word known as the lemma.

Eg. Lemma for studies is study

### **2.4.POS Tagging :**

POS (Parts of Speech) tell us about grammatical information of words of the sentence by assigning specific token (Determiner, noun, adjective , adverb , verb,Personal Pronoun etc.) as tag (DT,NN ,JJ,VB,PRP etc) to each words. Word can have more than one POS depending upon the context where it is used. We can use POS tags as statistical NLP tasks. It distinguishes a sense of word which is very helpful in text realization and infer semantic information from text for sentiment analysis.

### 3. Text Analysis Model using TF-IDF.

Term frequency–inverse document frequency(TFIDF) , is a numerical statistic that is intended to reflect how important a word is to a document in a collection or corpus.

- **Term Frequency (TF) :**

It is a measure of the frequency of a word (w) in a document (d). TF is defined as the ratio of a word's occurrence in a document to the total number of words in a document. The denominator term in the formula is to normalize since all the corpus documents are of different lengths.

$$TF(w, d) = \frac{\text{occurrences of } w \text{ in document } d}{\text{total number of words in document } d}$$

- **Inverse Document Frequency (IDF) :**

It is the measure of the importance of a word. Term frequency (TF) does not consider the importance of words. Some words such as 'of', 'and', etc. can be most frequently present but are of little significance. IDF provides weightage to each word based on its frequency in the corpus D.

$$IDF(w, D) = \ln\left(\frac{\text{Total number of documents (N) in corpus } D}{\text{number of documents containing } w}\right)$$

- **Term Frequency — Inverse Document Frequency (TFIDF)**

It is the product of TF and IDF.

TFIDF gives more weightage to the word that is rare in the corpus (all the documents). TFIDF provides more importance to the word that is more frequent in the document.

$$TFIDF(w, d, D) = TF(w, d) * IDF(w, D)$$

- **Disadvantage of TFIDF :**

It is unable to capture the semantics. For example, funny and humorous are synonyms, but TFIDF does not capture that. Moreover, TFIDF can be computationally expensive if the vocabulary is vast.

### 4. Bag of Words (BoW)

Machine learning algorithms cannot work with raw text directly. Rather, the text must be converted into vectors of numbers. In natural language processing, a common technique for extracting features from text is to place all of the words that occur in the text in a bucket. This approach is called a bag of words model or BoW for short. It's referred to as a "bag" of words because any information about the structure of the sentence is lost.

### **Algorithm for Tokenization, POS Tagging, stop words removal, Stemming and Lemmatization:**

**Step 1:** Download the required packages

```
nltk.download('punkt')
nltk.download('stopwords')
nltk.download('wordnet')
nltk.download('averaged_perceptron_tagger')
```

**Step 2:** Initialize the text

```
text= "Tokenization is the first step in text analytics."
```

**Step 3:** Perform Tokenization

#Sentence Tokenization

```
from nltk.tokenize import sent_tokenize
```

```
tokenized_text= sent_tokenize(text)
```

```
print(tokenized_text)
```

#Word

Tokenization

```
from nltk.tokenize import word_tokenize
```

```
tokenized_word=word_tokenize(text)
```

```
print(tokenized_word)
```

**Step 4:** Removing Punctuations and Stop Word

# print stop words of English

```
from nltk.corpus import stopwords
```

```
stop_words=set(stopwords.words("english"))
```

```
print(stop_words)
```

```
text= "How to remove stop words with NLTK library in Python?"
```

```
text= re.sub('[^a-zA-Z]', '',text)
```

```
tokens = word_tokenize(text.lower())
```

```
filtered_text=[]
```

**for w in tokens:**

```
if w not in stop_words:
```

```
filtered_text.append(w)
```

```
print("Tokenized Sentence:",tokens)
```

```
print("Filterd Sentence:",filtered_text)
```

**Step 5 : Perform Stemming**

```
from nltk.stem import PorterStemmer
e_words= ["wait", "waiting", "waited", "waits"]
ps =PorterStemmer()
for w in e_words:
    rootWord=ps.stem(w)
    print(rootWord)
```

**Step 6: Perform Lemmatization**

```
from nltk.stem import
```

```
WordNetLemmatizer
wordnet_lemmatizer = WordNetLemmatizer()
text = "studies studying cries cry"
tokenization = nltk.word_tokenize(text)
for w in tokenization:
    print("Lemma
    for
    {}
    wordnet_lemmatizer.lemmatize(w)))
```

**Step 7: Apply POS Tagging to text**

```
import nltk
from nltk.tokenize import word_tokenize
data="The pink sweater fit her perfectly"
words=word_tokenize(data)
for word in words:
    print(nltk.pos_tag([word]))
```

**Algorithm for Create representation of document by calculating TFIDF :**

**Step 1:** Import the necessary libraries.

```
import pandas as pd
from sklearn.feature_extraction.text import TfidfVectorizer
```

**Step 2:** Initialize the Documents.

```
documentA = 'Jupiter is the largest Planet'
documentB = 'Mars is the fourth planet from the Sun'
```

**Step 3:** Create BagofWords (BoW) for Document A and B.

```
bagOfWordsA = documentA.split(' ')
bagOfWordsB = documentB.split(' ')
```

**Step 4:** Create Collection of Unique words from Document A and B.

```
uniqueWords = set(bagOfWordsA).union(set(bagOfWordsB))
```

**Step 5:** Create a dictionary of words and their occurrence for each document in the corpus

```
numOfWordsA = dict.fromkeys(uniqueWords, 0)
for word in bagOfWordsA:
    numOfWordsA[word] += 1
numOfWordsB = dict.fromkeys(uniqueWords, 0)
for word in bagOfWordsB:
    numOfWordsB[word] += 1
```

**Step 6:** Compute the term frequency for each of our documents.

```
def computeTF(wordDict, bagOfWords):
    tfDict = {}
    bagOfWordsCount = len(bagOfWords)
    for word, count in wordDict.items():
        tfDict[word] = count / float(bagOfWordsCount)
    return tfDict
tfA = computeTF(numOfWordsA, bagOfWordsA)
tfB = computeTF(numOfWordsB, bagOfWordsB)
```

**Step 7:** Compute the term Inverse Document Frequency.

```
def computeIDF(documents):
    import math
    N = len(documents)
    idfDict = dict.fromkeys(documents[0].keys(), 0)
    for document in documents:
        for word, val in document.items():
            if val > 0:
                idfDict[word] += 1
    for word, val in idfDict.items():
        idfDict[word] = math.log(N / float(val))
    return idfDict
idfs = computeIDF([numOfWordsA, numOfWordsB])
idfs
```

**Step 8:** Compute the term TF/IDF for all words.

```
def computeTFIDF(tfBagOfWords, idfs):
    tfidf = {}
    for word, val in tfBagOfWords.items():
        tfidf[word] = val * idfs[word]
    return tfidf
tfidfA = computeTFIDF(tfA, idfs)
```

```
tfidfB = computeTFIDF(tfB, idfs)
df = pd.DataFrame([tfidfA, tfidfB])
df
```

**Conclusion:**

In this way we have done text data analysis using TF IDF algorithm

---

**Group A**  
**Assignment No: 8**

---

**Contents for Theory:**

- 1. Seaborn Library Basics**
  - 2. Know your Data**
  - 3. Finding patterns of data.**
  - 4. Checking how the price of the ticket (column name: 'fare') for each passenger is distributed by plotting a histogram.**
- 

**Theory:**

Data Visualisation plays a very important role in Data mining. Various data scientists spent their time exploring data through visualisation. To accelerate this process we need to have a well-documentation of all the plots.

Even plenty of resources can't be transformed into valuable goods without planning and architecture

**1. Seaborn Library Basics :**

Seaborn is a Python data visualisation library based on matplotlib. It provides a high-level interface for drawing attractive and informative statistical graphics.

For the installation of Seaborn, you may run any of the following in your command line.

**pip install seaborn**

**conda install seaborn**

To import seaborn you can run the following command.

```
import seaborn as sns
```

**2. Know your data :**

The dataset that we are going to use to draw our plots will be the Titanic dataset, which is downloaded by default with the Seaborn library. All you have to do is use the load\_dataset function and pass it the name of the dataset.

```
import pandas as pd  
import numpy as np  
import matplotlib.pyplot as plt  
import seaborn as sns  
dataset = sns.load_dataset('titanic')  
dataset.head()
```

**3. Finding patterns of data. :**

Patterns of data can be find out with the help of different types of plots

**Types of plots are:**

**A. Distribution Plots**

- a. Dist-Plot
- b. Joint Plot
- d. Rug Plot

**B. Categorical Plots**

- a. Bar Plot
- b. Count Plot
- c. Box Plot
- d. Violin Plot

**C. Advanced Plots**

- a. Strip Plot
- b. Swarm Plot

**D. Matrix Plots**

- a. Heat Map
- b. Cluster Map

**A. Distribution Plots:**

These plots help us to visualise the distribution of data. We can use these plots to understand the mean, median, range, variance, deviation, etc of the data.

**.b. Joint Plot :**

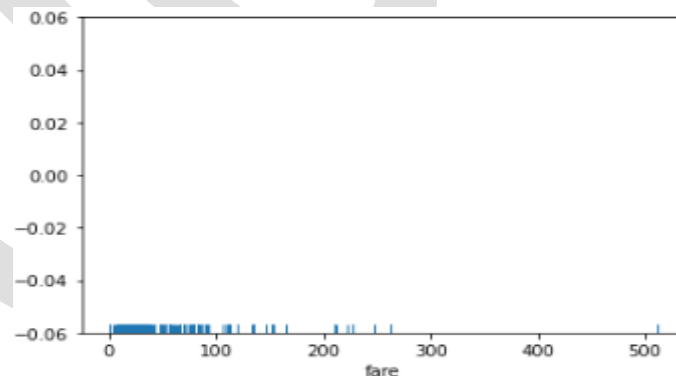
- It is the combination of the distplot of two variables.
- It is an example of bivariate analysis.
- We additionally obtain a scatter plot between the variables to reflect their linear relationship. We can customise the scatter plot into a hexagonal plot, where, the more the colour intensity, the more will be the number of observations.

```
➤ import seaborn as sns
# For Plot 1
sns.jointplot(x=dataset['age'],y=dataset['fare'],kind='scatter')
# For Plot 2
sns.jointplot(x = dataset['age'], y = dataset['fare'], kind = 'hex')
```

**c. The Rug Plot :**

The rugplot() is used to draw small bars along the x-axis for each point in the dataset. To plot a rug plot, you need to pass the name of the column. Let's plot a rug plot for fare.

**Syntax: sns.rugplot(dataset['fare'])**



From the output, you can see that most of the instances for the fares have values between 0 and 100.

These are some of the most commonly used distribution plots offered by the Python's Seaborn Library. Let's see some of the categorical plots in the Seaborn library.

**2. Categorical Plots :**

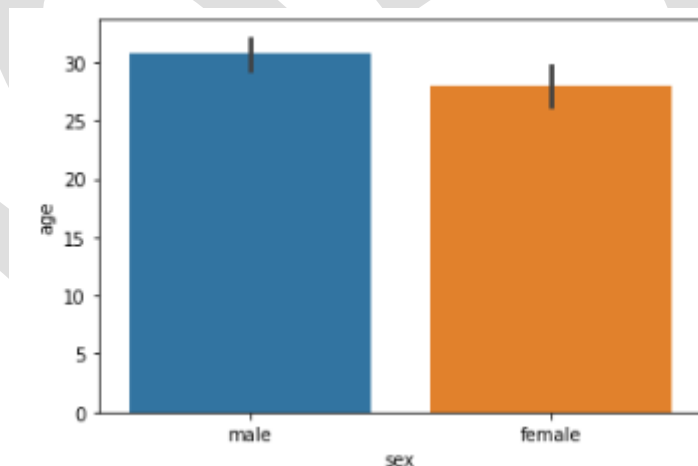


Categorical plots, as the name suggests, are normally used to plot categorical data. The categorical plots plot the values in the categorical column against another categorical column or a numeric column. Let's see some of the most commonly used categorical data.

### b. The Bar Plot :

The `barplot()` is used to display the mean value for each value in a categorical column, against a numeric column. The first parameter is the categorical column, the second parameter is the numeric column while the third parameter is the dataset. For instance, if you want to know the mean value of the age of the male and female passengers, you can use the bar plot as follows.

**Syntax:** `sns.barplot(x='gender', y='age', data=dataset)`

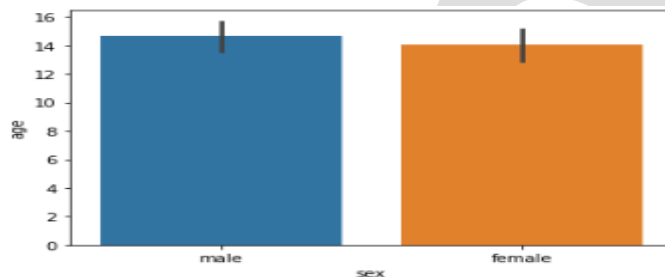


From the output, you can clearly see that the average age of male passengers is just less than 40 while the average age of female passengers is around 33.

In addition to finding the average, the bar plot can also be used to calculate other aggregate values for each category. To do so, you need to pass the aggregate function to the estimator. For instance, you can calculate the standard deviation for the age of each gender as follows:

```
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
sns.barplot(x='gender', y='age', data=dataset, estimator=np.std)
```

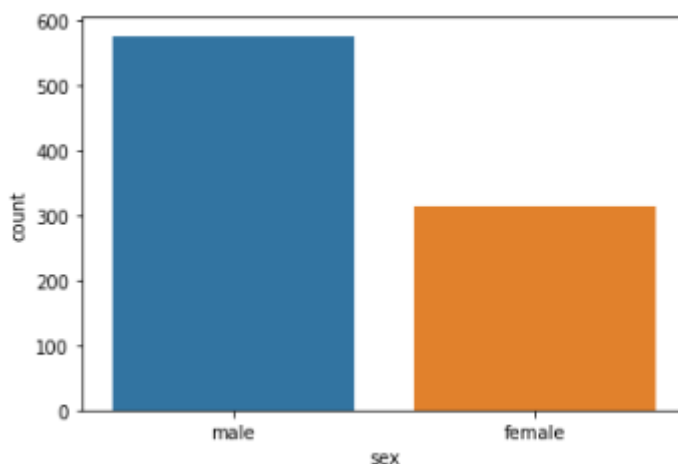
Notice, in the above script we use the std aggregate function from the numpy library to calculate the standard deviation for the ages of male and female passengers. The output looks like this:



### c. The Count Plot :

The count plot is similar to the bar plot, however it displays the count of the categories in a specific column. For instance, if we want to count the number of males and women passenger we can do so using count plot as follows:

**Syntax:** `sns.countplot(x='gender', data=dataset)`



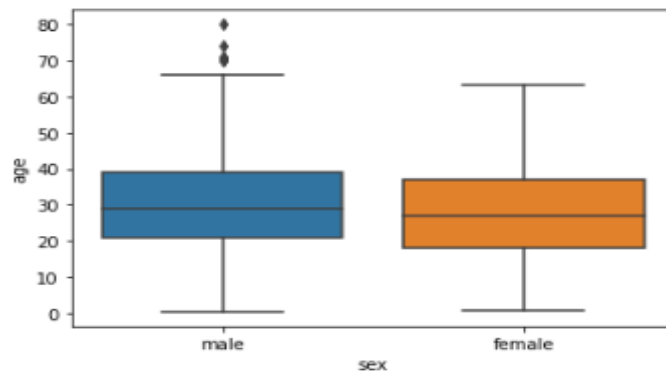
### d. The Box Plot :

The box plot is used to display the distribution of the categorical data in the form of quartiles. The centre of the box shows the median value. The value from the lower whisker to the bottom of the box shows the first quartile. From the bottom of the box to the middle of the box lies the second quartile. From the middle of the box to the top of the box lies the third quartile and finally from the top of the box to the top whisker lies the last quartile.

Now let's plot a box plot that displays the distribution for the age with respect to each gender. You need to pass the categorical column as the first parameter (which is sex in our

case) and the numeric column (age in our case) as the second parameter. Finally, the dataset is passed as the third parameter, take a look at the following script:

**Syntax:** `sns.boxplot(x='gender', y='age', data=dataset)`



You can make your box plots more fancy by adding another layer of distribution. For instance, if you want to see the box plots of forage of passengers of both genders, along with the information about whether or not they survived, you can pass the survived as value to the hue parameter as shown below :

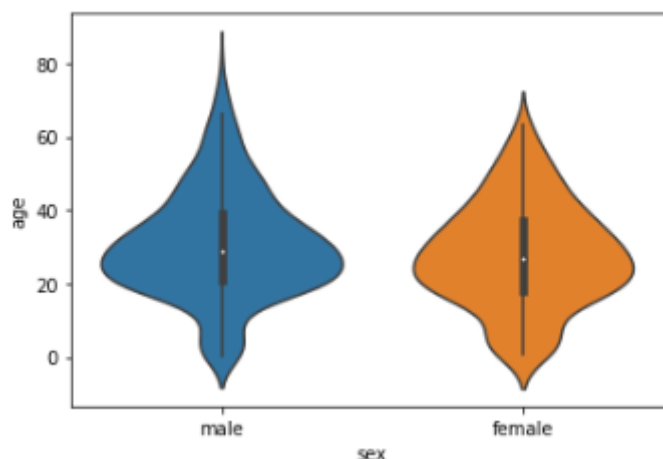
**Syntax:** `sns.boxplot(x='gender', y='age', data=dataset, hue='survived')`

#### e. The Violin Plot :

The violin plot is similar to the box plot, however, the violin plot allows us to display all the components that actually correspond to the data point. The `violinplot()` function is used to plot the violin plot. Like the box plot, the first parameter is the categorical column, the second parameter is the numeric column while the third parameter is the dataset.

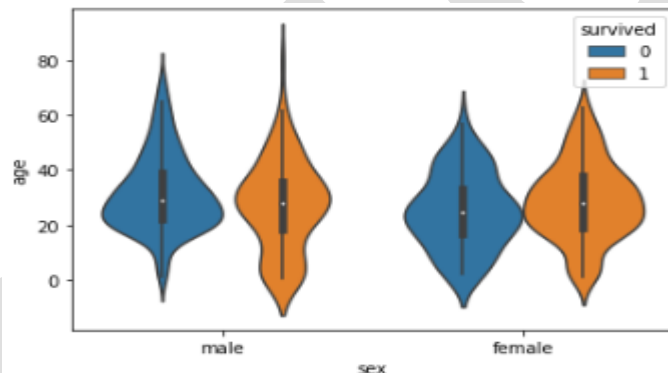
Let's plot a violin plot that displays the distribution for the age with respect to each gender.

**Syntax:** `sns.violinplot(x='gender', y='age', data=dataset)`



Like box plots, you can also add another categorical variable to the violin plot using the hue parameter as shown below :

**Syntax:** `sns.violinplot(x='gender', y='age', data=dataset, hue='survived')`



### Advanced Plots:

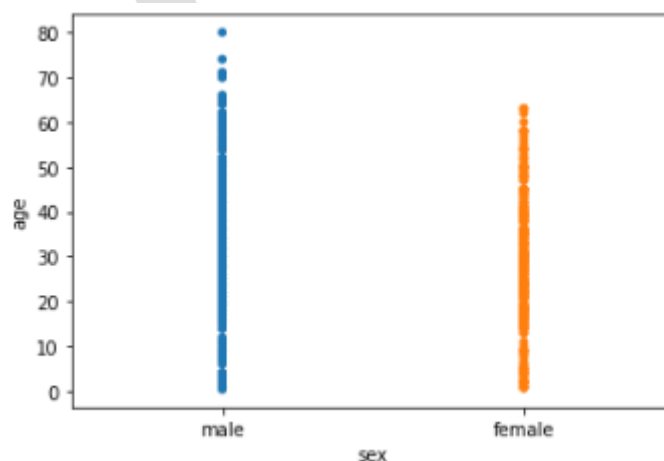
#### a. The Strip Plot :

The strip plot draws a scatter plot where one

of the variables is categorical. We have seen scatter plots in the joint plot and the pair plot sections where we had two numeric variables. The strip plot is different in a way that one of the variables is categorical in this case, and for each category in the categorical variable, you will see a scatter plot with respect to the numeric column.

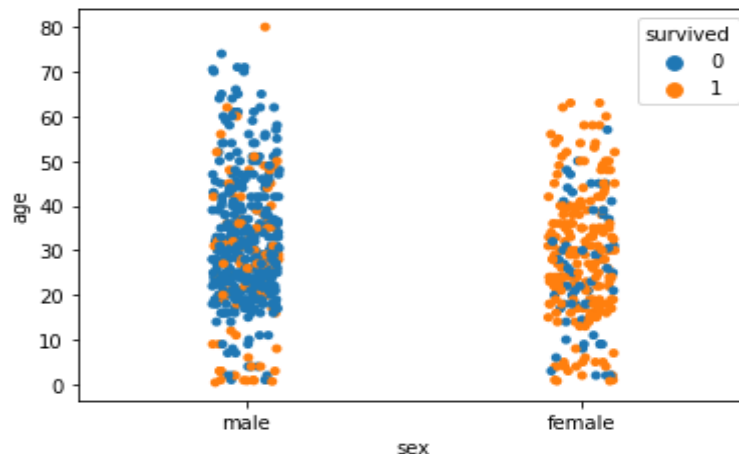
The `stripplot()` function is used to plot the violin plot. Like the box plot, the first parameter is the categorical column, the second parameter is the numeric column while the third parameter is the dataset. Look at the following script:

**Syntax:** `sns.stripplot(x='gender', y='age', data=dataset, jitter=False)`



Like violin and box plots, you can add an additional categorical column to strip plot using hue parameter as shown below:

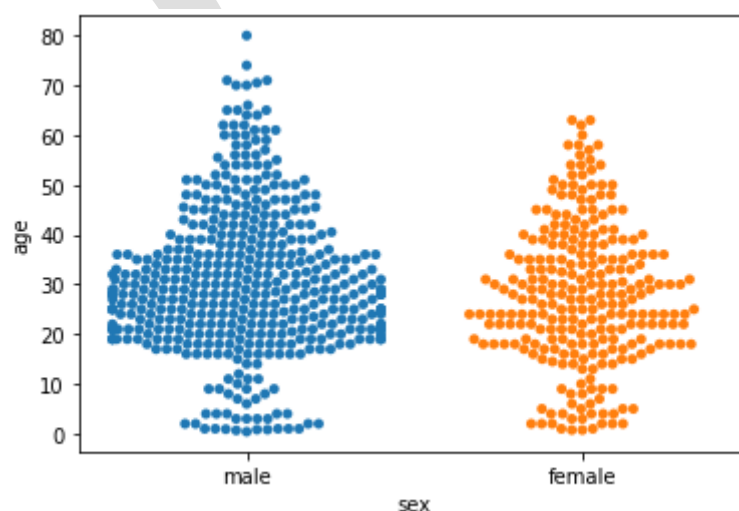
**Syntax:** `sns.stripplot(x='gender', y='age', data=dataset, jitter=True, hue='survived')`



#### b. The Swarm Plot :

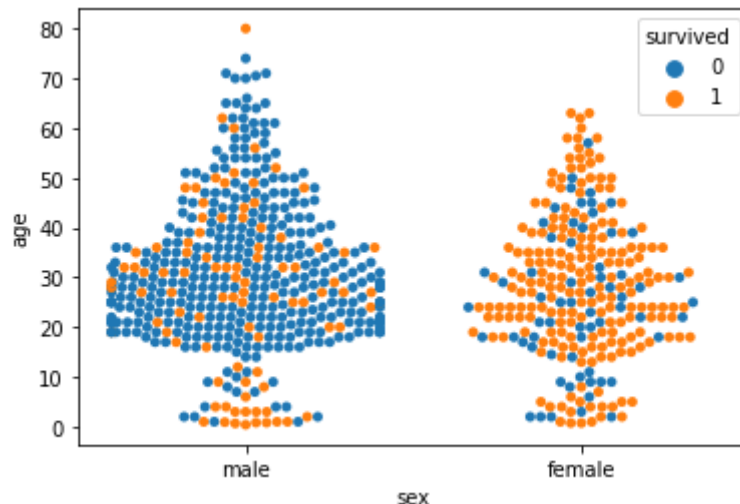
The swarm plot is a combination of the strip and the violin plots. In the swarm plots, the points are adjusted in such a way that they don't overlap. Let's plot a swarm plot for the distribution of age against gender. The `swarmplot()` function is used to plot the violin plot. Like the box plot, the first parameter is the categorical column, the second parameter is the numeric column while the third parameter is the dataset. Look at the following script:

**Syntax:** `sns.swarmplot(x='gender', y='age', data=dataset)`



Let's add another categorical column to the swarm plot using the hue parameter.

**Syntax:** `sns.swarmplot(x='gender', y='age', data=dataset, hue='survived')`



### 1. Matrix Plots :

Matrix plots are the type of plots

that show data in the form of rows and columns. Heat maps are the prime examples of matrix plots

#### a. Heat Maps :

Heat maps are normally used to plot correlation between numeric columns in the form of a matrix. It is important to mention here that to draw matrix plots, you need to have meaningful information on rows as well as columns. Let's plot the first five rows of the Titanic dataset to see if both the rows and column headers have meaningful information.

Execute the following script:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
dataset = sns.load_dataset('titanic')
dataset.head()
```

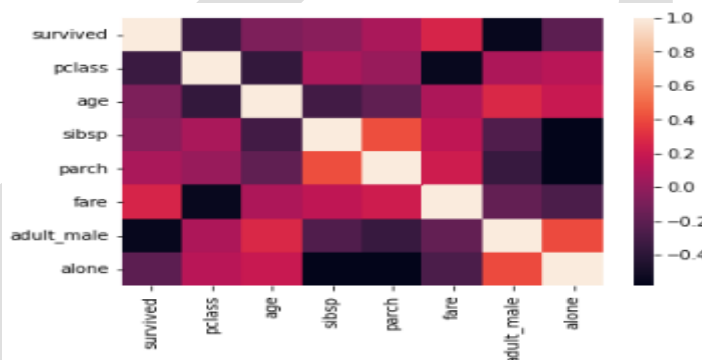
|   | survived | pclass | sex    | age  | sibsp | parch | fare    | embarked | class | who   | adult_male | deck | embark_town | alive | alone |
|---|----------|--------|--------|------|-------|-------|---------|----------|-------|-------|------------|------|-------------|-------|-------|
| 0 | 0        | 3      | male   | 22.0 | 1     | 0     | 7.2500  | S        | Third | man   | True       | NaN  | Southampton | no    | False |
| 1 | 1        | 1      | female | 38.0 | 1     | 0     | 71.2833 | C        | First | woman | False      | C    | Cherbourg   | yes   | False |
| 2 | 1        | 3      | female | 26.0 | 0     | 0     | 7.9250  | S        | Third | woman | False      | NaN  | Southampton | yes   | True  |
| 3 | 1        | 1      | female | 35.0 | 1     | 0     | 53.1000 | S        | First | woman | False      | C    | Southampton | yes   | False |
| 4 | 0        | 3      | male   | 35.0 | 0     | 0     | 8.0500  | S        | Third | man   | True       | NaN  | Southampton | no    | True  |

The `corr()` function returns the correlation between all the numeric columns of the dataset. Execute the following script:

**Syntax:** `dataset.corr()`

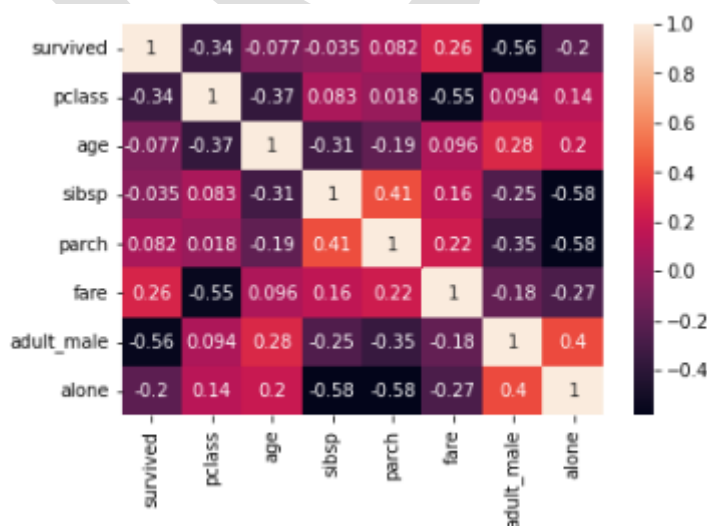
Now to create a heat map with these correlation values, you need to call the `heatmap()` function and pass it your correlation dataframe. Look at the following script:

**Syntax:** `corr = dataset.corr()`  
`sns.heatmap(corr)`



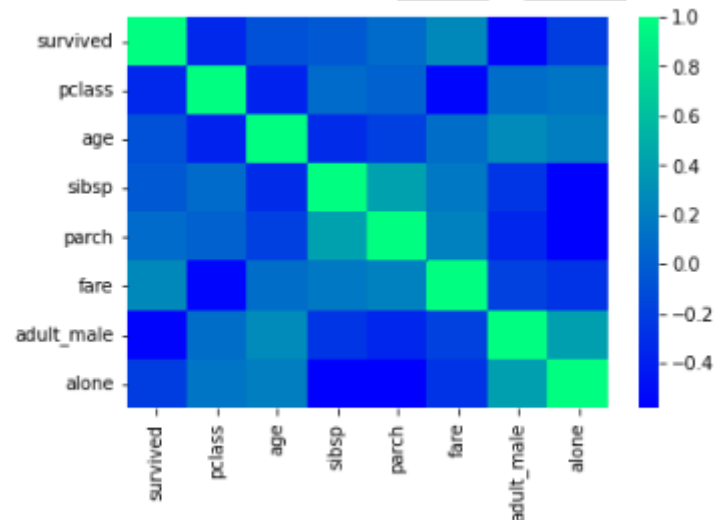
The correlation values can also be plotted on the heatmap by passing `True` for the `annot` parameter. Execute the following script to see this in action :

`corr = dataset.corr()`  
`sns.heatmap(corr, annot=True)`



You can also change the colour of the heatmap by passing an argument for the `cmap` parameter. For now, just look at the following script:

**Syntax:** `corr = dataset.corr()`  
`sns.heatmap(corr)`

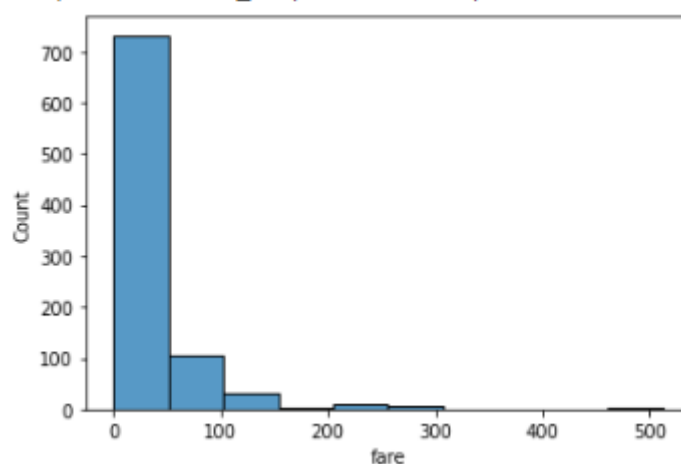


### b. Cluster Map:

In addition to the heat map, another commonly used matrix plot is the cluster map. The cluster map basically uses Hierarchical Clustering to cluster the rows and columns of the matrix. Let's plot a cluster map for the number of passengers who travelled in a specific month of a specific year. Execute the following script:

Checking how the price of the ticket (column name: 'fare') for each passenger is distributed by plotting a histogram.

```
import seaborn as sns
dataset = sns.load_dataset('titanic')
sns.histplot(dataset['fare'], kde=False, bins=10)
```





From the histogram, it is seen that for around 730 passengers the price of the ticket is 50. For 100 passengers the price of the ticket is 100 and so on.

### **Conclusion :-**

Seaborn is an advanced data visualisation library built on top of Matplotlib library. In this assignment, we looked at how we can draw distributional and categorical plots using the Seaborn library. We have seen how to plot matrix plots in Seaborn. We also saw how to change plot styles and use grid functions to manipulate subplots.

---

### **Group A Assignment No: 9**

---

#### **Contents for Theory:**

1. Seaborn Library Basics
  2. Know your Data
  3. Finding patterns of data.
  4. Checking how the price of the ticket (column name: 'fare') for each passenger is distributed by plotting a histogram.
- 

Follow the previous contents for assignment no 9

Conclusion should be written in your own words.