LOKNETE. DR. BALASAHEB VIKHE PATIL
(PADMA BHUSHAN AWA RDEE)
PRAVARA RURAL EDUCATION SOCIETY

PRAVARA RURAL ENGINEERING COLLEGE

LONI

DEPARTMENT OF COMPUTER ENGINEERING

Laboratory Manual

A.Y. 2021-2022

Data Science & Big data Analytics Laboratory

TE-COMPUTER ENGINEERING

SEMESTER-II

Subject Code: 310248

Group A

Assignment No: 1

Title of the Assignment: Data Wrangling, I

Perform the following operations using Python on any open source dataset (e.g., data.csv)

Import all the required Python Libraries.

1. Locate open source data from the web (e.g. https://www.kaggle.com).
2. Provide a clear description of the data and its source (i.e., URL of the web site).
3. Load the Dataset into the pandas data frame.
4. Data Preprocessing: check for missing values in the data using pandas insult(), describe() function to get some initial statistics. Provide variable descriptions. Types of variables etc. Check the dimensions of the data frame.
5. Data Formatting and Data Normalization: Summarize the types of variables by checking the data types (i.e., character, numeric, integer, factor, and logical) of the variables in the data set. If variables are not in the correct data type, apply proper type conversions.
6. Turn categorical variables into quantitative variables in Python.

Objective of the Assignment: Students should be able to perform the data wrangling operation using Python on any open source dataset

Prerequisite:

1. Basic of Python Programming

2. Concept of Data Preprocessing, Data Formatting , Data Normalization and Data Cleaning.

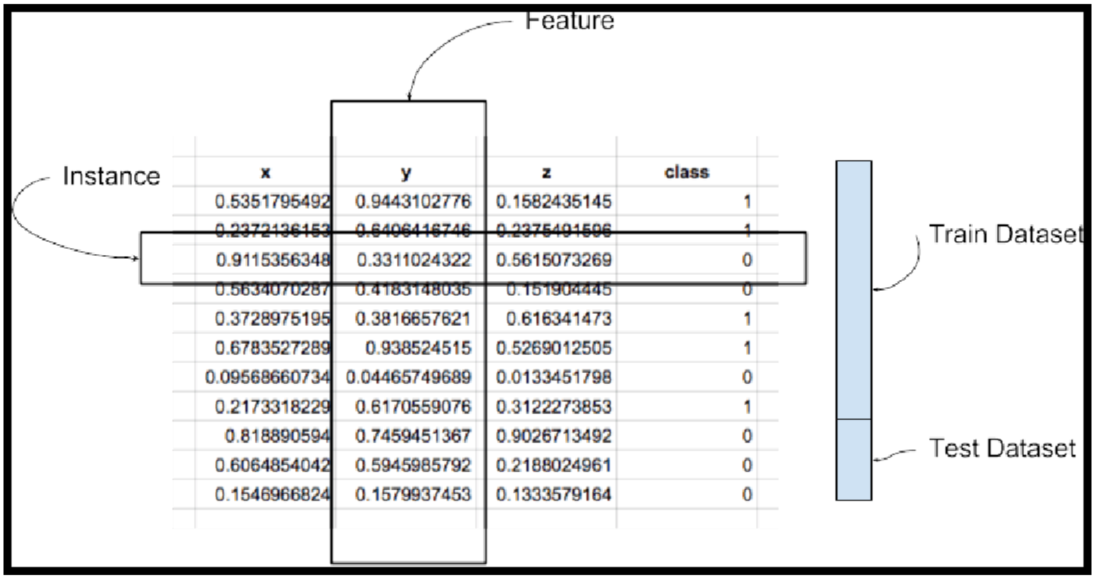
Contents for Theory:

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 Normalization**
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	0.2375401506	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.

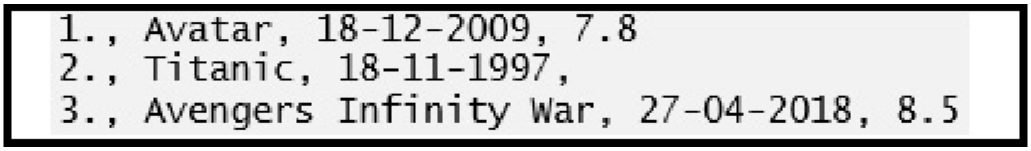
Datasets: A collection of instances is a dataset and when working with machine learning methods we typically need a few datasets for different purposes.

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.

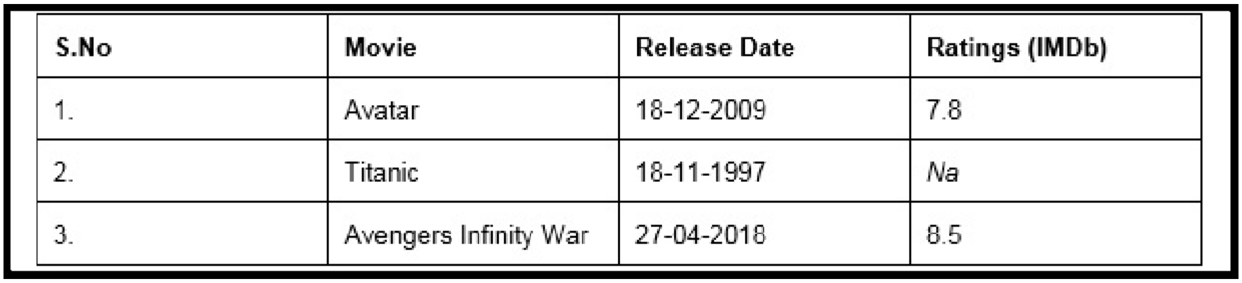
Data Represented in a Table:

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 wide range of visualizations. With a bit of effort and tint of visualization capabilities, with Matplotlib, you can create just any visualizations:Line plots

- 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 corrplot 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.

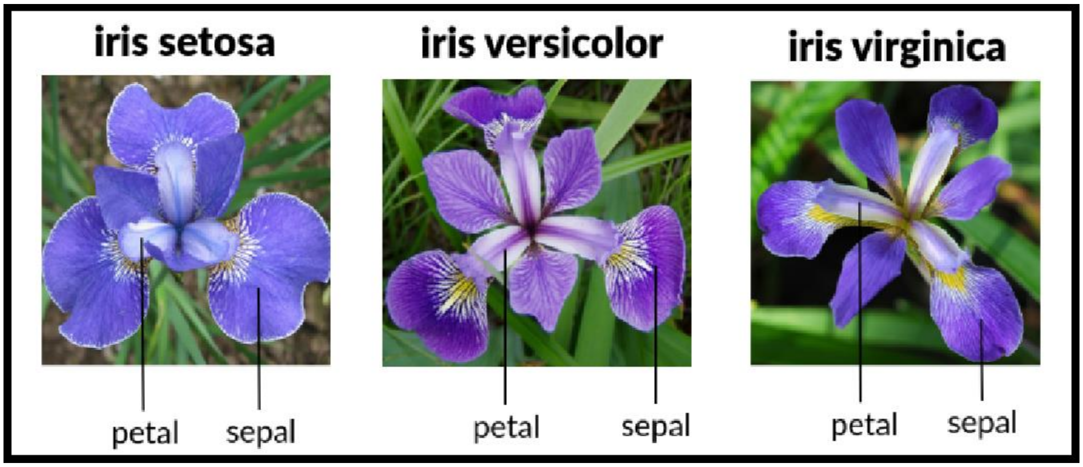
It includes three iris species with 50 samples each as well as some properties about each flower. One flower species is linearly separable from the other two, but the other two are not linearly separable from each other.

Total Sample- 150

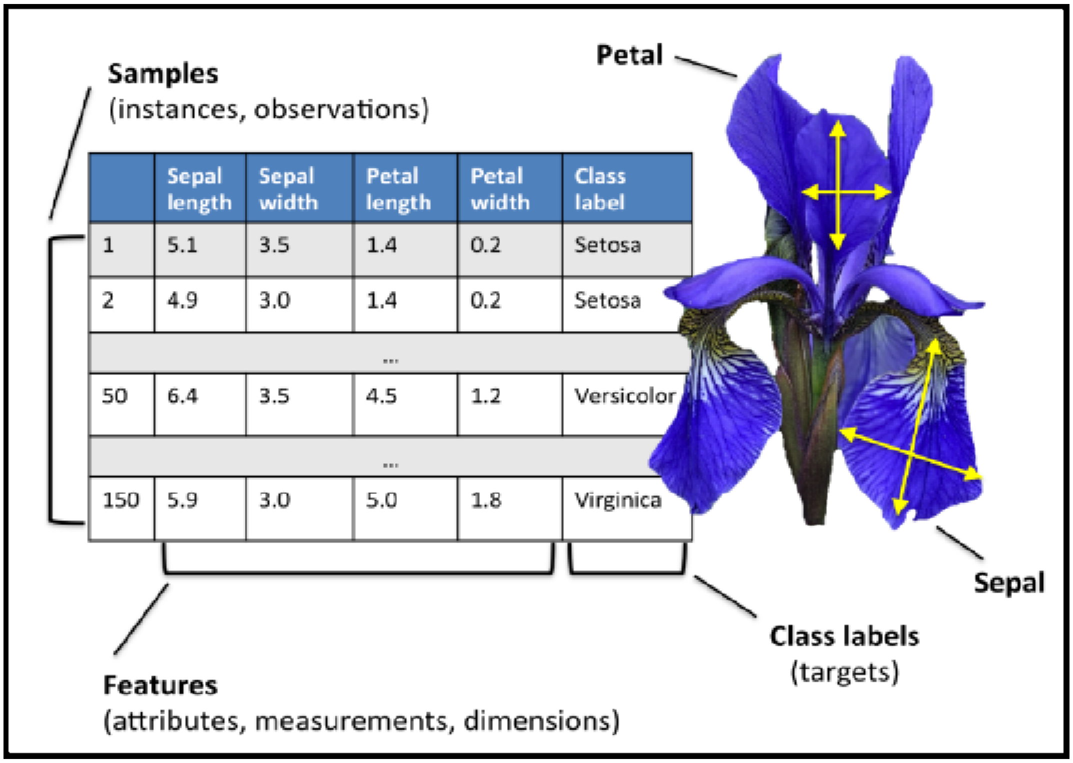
The columns in this dataset are:

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

3 Different Types of Species each contain 50 Sample-



Description of Dataset-


Samples
(instances, observations)
Petal
Sepal
Class labels
(targets)
Features
(attributes, measurements, dimensions)

	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

4. Panda Dataframe functions for Load Dataset

# The columns of the resulting DataFrame have different dtypes.

iris.dtypes

1. The dataset is downloads 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-

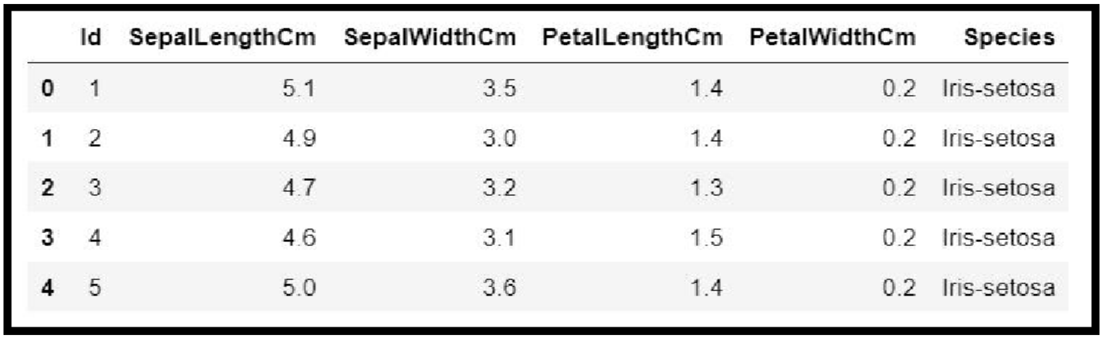
iris = pd.read_csv(csv_url, header = None)

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)



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

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

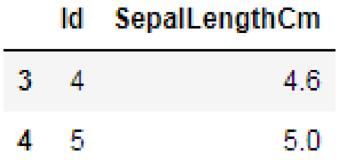
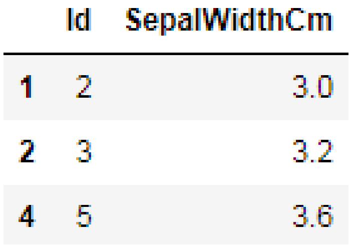
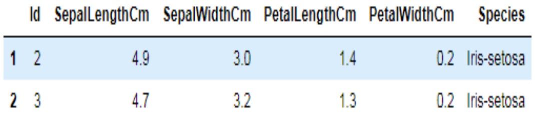
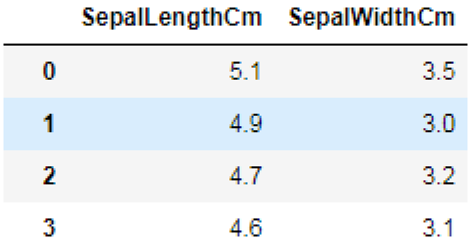
Dataframe Operations:

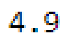
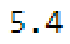
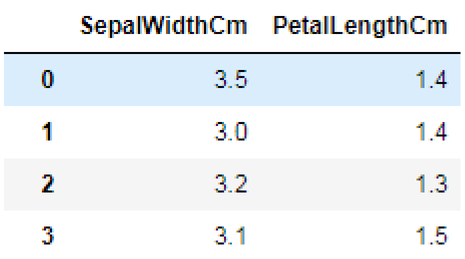
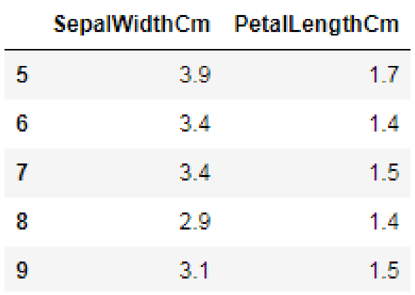
Sr. No	**Data Frame Function**	**Description**
1	**dataset.head(n=5)**	**Return the first n rows.**
2	**dataset.tail(n=5)**	**Return the last n rows.**
3	**dataset.index**	The index (row labels) of the Dataset.
4	**dataset.columns**	The column labels of the Dataset.
5	**dataset.shape**	Return a tuple representing the dimensionality of the Dataset.
6	**dataset.dtypes**	Return the dtypes in the Dataset.

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

15	**dataset.iloc[:n, :]**	a subset of the first n rows of the original data
16	**dataset.iloc[:, :n]**	a subset of the first n columns of the original data
17	**dataset.iloc[:m, :n]**	a subset of the first m rows and the first n columns

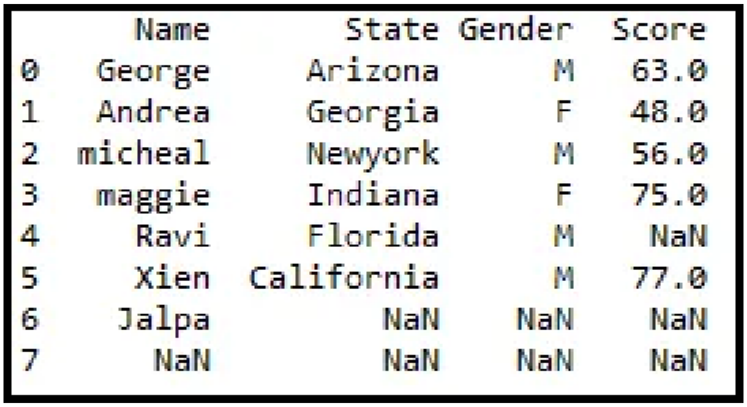
Few Examples of iLoc to slice data for iris Dataset

Sr. No	Data Frame Function	Description	Output
1	**dataset.iloc[3:5, 0:2]**	Slice the data	 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:	 Id SepalWidthCm 1 2 3.0 2 3 3.2 4 5 3.6
3	**dataset.iloc[1:3, :]**	For slicing rows explicitly:	 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:	 SepalLengthCm SepalWidthCm 0 5.1 3.5 1 4.9 3.0 2 4.7 3.2 3 4.6 3.1

4	**dataset.iloc[1, 1]**	For getting a value explicitly:	 4.9
5	**dataset['SepalLengthCm'].iloc[5]**	Accessing Column and Rows by position	 5.4
6	**cols_2_4=dataset.columns[2:4]** **dataset[cols_2_4]**	Get Column Name then get data from column	 SepalWidthCm PetalLengthCm 0 3.5 1.4 1 3.0 1.4 2 3.2 1.3 3 3.1 1.5
7	**dataset[dataset.columns[2:4]].iloc[5:10]**	in one Expression answer for the above two commands	 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:

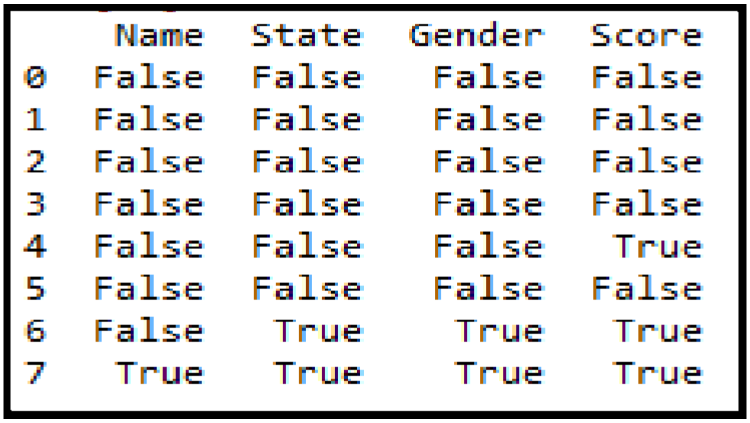


```
      Name       State Gender  Score
0   George     Arizona      M   63.0
1   Andrea     Georgia      F   48.0
2  micheal     Newyork      M   56.0
3   maggie     Indiana      F   75.0
4     Ravi     Florida      M    NaN
5     Xien  California      M   77.0
6    Jalpa         NaN    NaN    NaN
7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

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

Function: DataFrame.isnull()

Output:

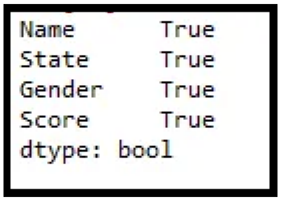


```
    Name  State  Gender  Score
0  False  False   False  False
1  False  False   False  False
2  False  False   False  False
3  False  False   False  False
4  False  False   False   True
5  False  False   False  False
6  False   True    True   True
7   True   True    True   True
```

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.

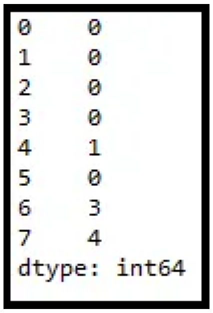
Function: dataframe.isnull().sum().sum()

Output : 8

d. **count row wise missing value using isnull()**

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

Output:



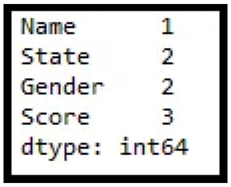
```
0    0
1    0
2    0
3    0
4    1
5    0
6    3
7    4
dtype: int64
```

e. **count Column wise missing value using isnull()**

Method 1:

Function: dataframe.isnull().sum()

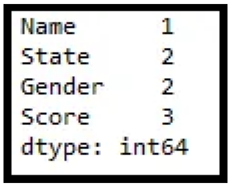
Output:



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

Method 2:

unction: 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()

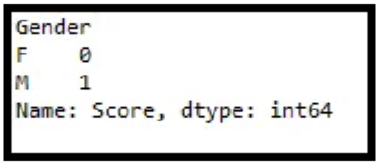
Output: 2

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:



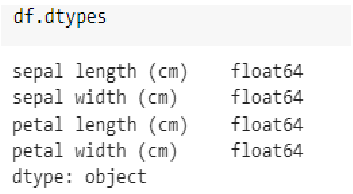
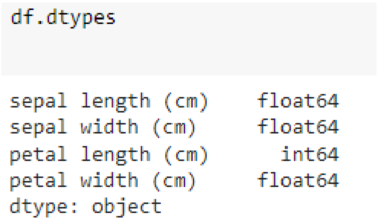
```
Gender
F    0
M    1
Name: Score, dtype: int64
```

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	 df.dtypes sepal length (cm) float64 sepal width (cm) float64 petal length (cm) float64 petal width (cm) float64 dtype: object
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)	 df.dtypes sepal length (cm) float64 sepal width (cm) float64 petal length (cm) int64 petal width (cm) float64 dtype: object

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 5: Create a minimum and maximum processor object

min_max_scaler = preprocessing.MinMaxScaler()

Step 6: 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)

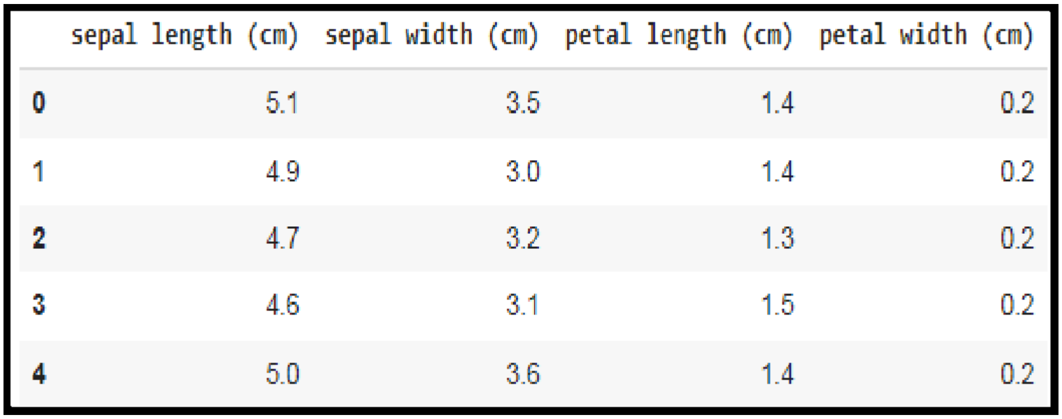
Step 7: Run the normalizer on the dataframe

df_normalized = pd.DataFrame(x_scaled)

Step 8: View the dataframe

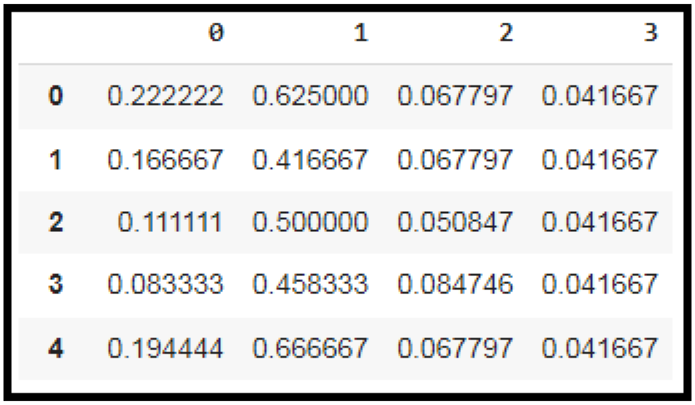
df_normalized

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

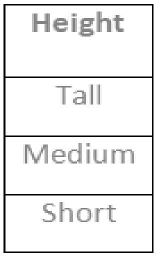
7. **Panda Functions for handling categorical variables**

- **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 t**end 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.**

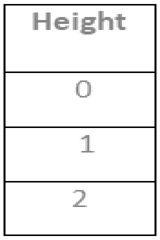
There are many ways to convert categorical data into numerical data. Here the three most used methods are discussed.

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
Tall
Medium
Short



Height
0
1
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 0and 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()

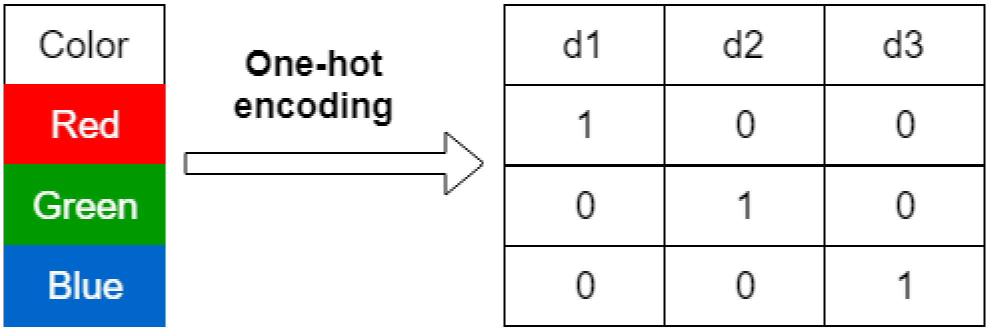
Output: array([0, 1, 2], dtype=int64)

- 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.



Color
Red
Green
Blue

d1	d2	d3
1	0	0
0	1	0
0	0	1

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

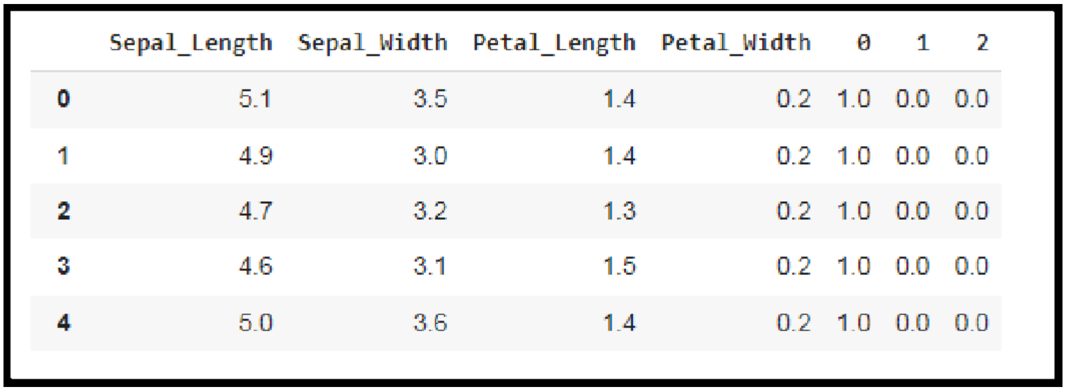
```
df_encode
```

Step 9: Rename the newly encoded columns.

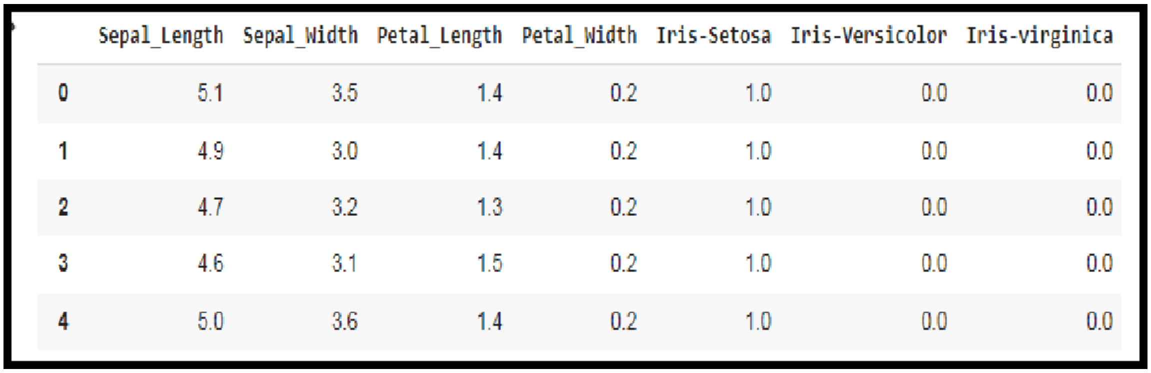
```
df_encode.rename(columns = {0:'Iris-Setosa',

1:'Iris-Versicolor',2:'Iris-virginica'}, inplace = True)
```

Step 10: Observe the merge dataframe df_encode

Output after Step 8:



	Sepal_Length	Sepal_Width	Petal_Length	Petal_Width	0	1	2
0	5.1	3.5	1.4	0.2	1.0	0.0	0.0
1	4.9	3.0	1.4	0.2	1.0	0.0	0.0
2	4.7	3.2	1.3	0.2	1.0	0.0	0.0
3	4.6	3.1	1.5	0.2	1.0	0.0	0.0
4	5.0	3.6	1.4	0.2	1.0	0.0	0.0

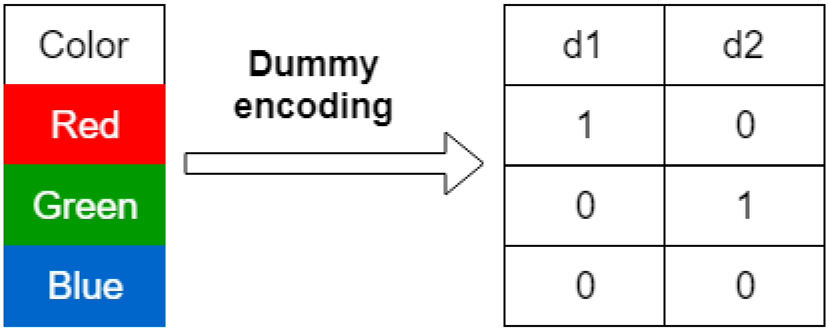


	Sepal_Length	Sepal_Width	Petal_Length	Petal_Width	Iris-Setosa	Iris-Versicolor	Iris-virginica
0	5.1	3.5	1.4	0.2	1.0	0.0	0.0
1	4.9	3.0	1.4	0.2	1.0	0.0	0.0
2	4.7	3.2	1.3	0.2	1.0	0.0	0.0
3	4.6	3.1	1.5	0.2	1.0	0.0	0.0
4	5.0	3.6	1.4	0.2	1.0	0.0	0.0

Output after Step 10:

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, prefix=None, prefix_sep='_', dummy_na=False, 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_nabool: 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()

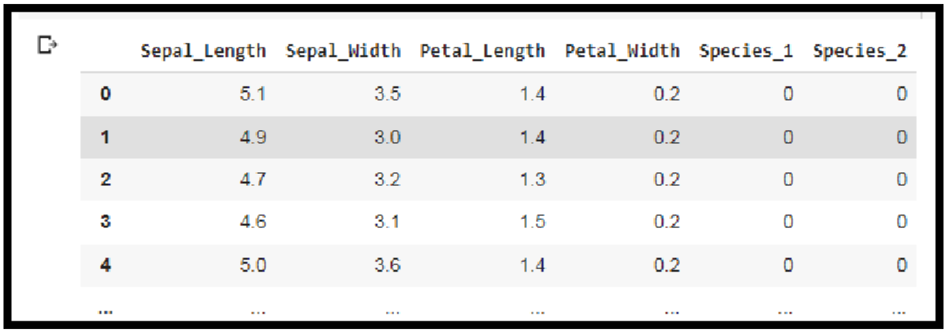
Output: array([0, 1, 2], dtype=int64)

Step 6: 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
...	...	...	...	...	...	...

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.

Assignment Question

1. **Explain Data Frame with Suitable example.**
2. **What is the limitation of the label encoding method?**
3. **What is the need of data normalization?**

What are the different Techni

Assignment No: 2

Title of the Assignment: Data Wrangling, II

Create an “Academic performance” dataset of students and perform the following operations using Python.

1. Scan all variables for missing values and inconsistencies. If there are missing values and/or inconsistencies, use any of the suitable techniques to deal with them.
2. Scan all numeric variables for outliers. If there are outliers, use any of the suitable techniques to deal with them.
3. Apply data transformations on at least one of the variables. The purpose of this transformation should be one of the following reasons: to change the scale for better understanding of the variable, to convert a non-linear relation into a linear one, or to decrease the skewness and convert the distribution into a normal distribution.

   Reason and document your approach properly.

Objective of the Assignment: Students should be able to perform thedata wrangling operation using Python on any open source dataset

---

Prerequisite:

1. Basic of Python Programming
2. Concept of Data Preprocessing, Data Formatting , Data Normalization and Data Cleaning.

Contents for Theory:

1. Creation of Dataset using Microsoft Excel.
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 Instance**s: 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

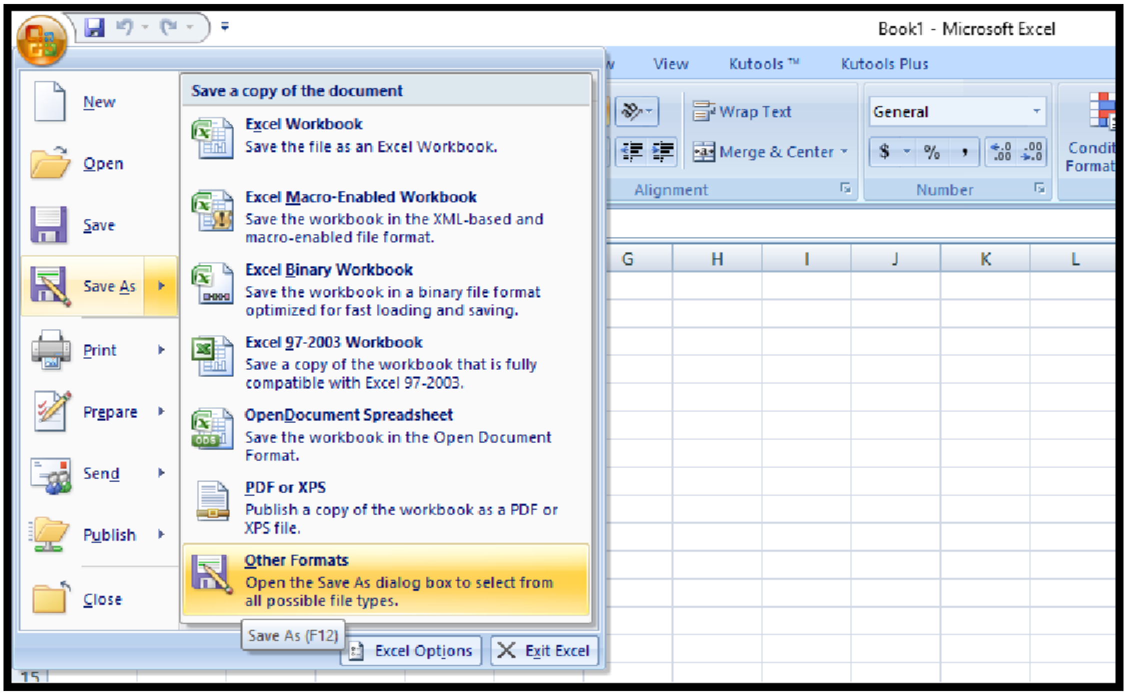
Syntax : RANDBETWEEN(bottom, top) Bottom The smallest integer and

Top The largest integer RANDBETWEEN will return.

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

The step to create the dataset are as follows:

Step 1: Open Microsoft Excel and click on Save As. Select Other .Formats



Save a copy of the document

Excel Workbook
Save the file as an Excel Workbook.

Excel Macro-Enabled Workbook
Save the workbook in the XML-based and macro-enabled file format.

Excel Binary Workbook
Save the workbook in a binary file format optimized for fast loading and saving.

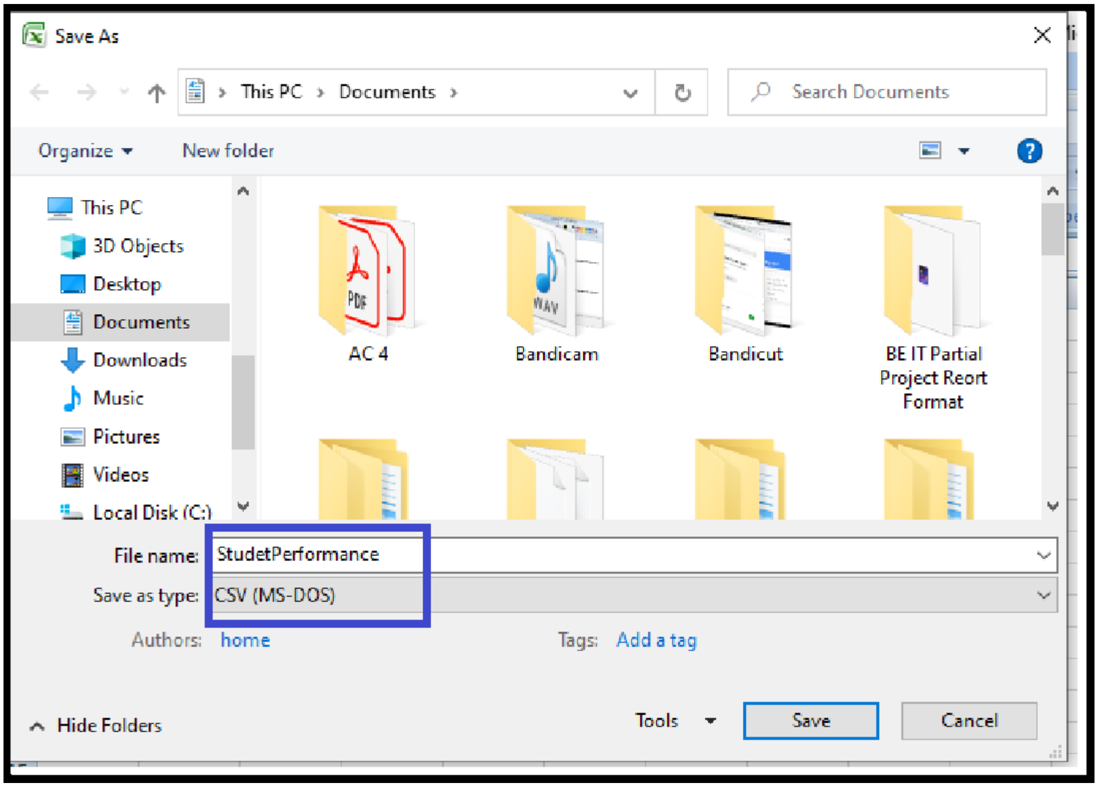
Excel 97-2003 Workbook
Save a copy of the workbook that is fully compatible with Excel 97-2003.

OpenDocument Spreadsheet
Save the workbook in the Open Document Format.

PDF or XPS
Publish a copy of the workbook as a PDF or XPS file.

Other Formats
Open the Save As dialog box to select from all possible file types.

Step 2: Enter the name of the dataset and Save the dataset astye CSV(MS-DOS).



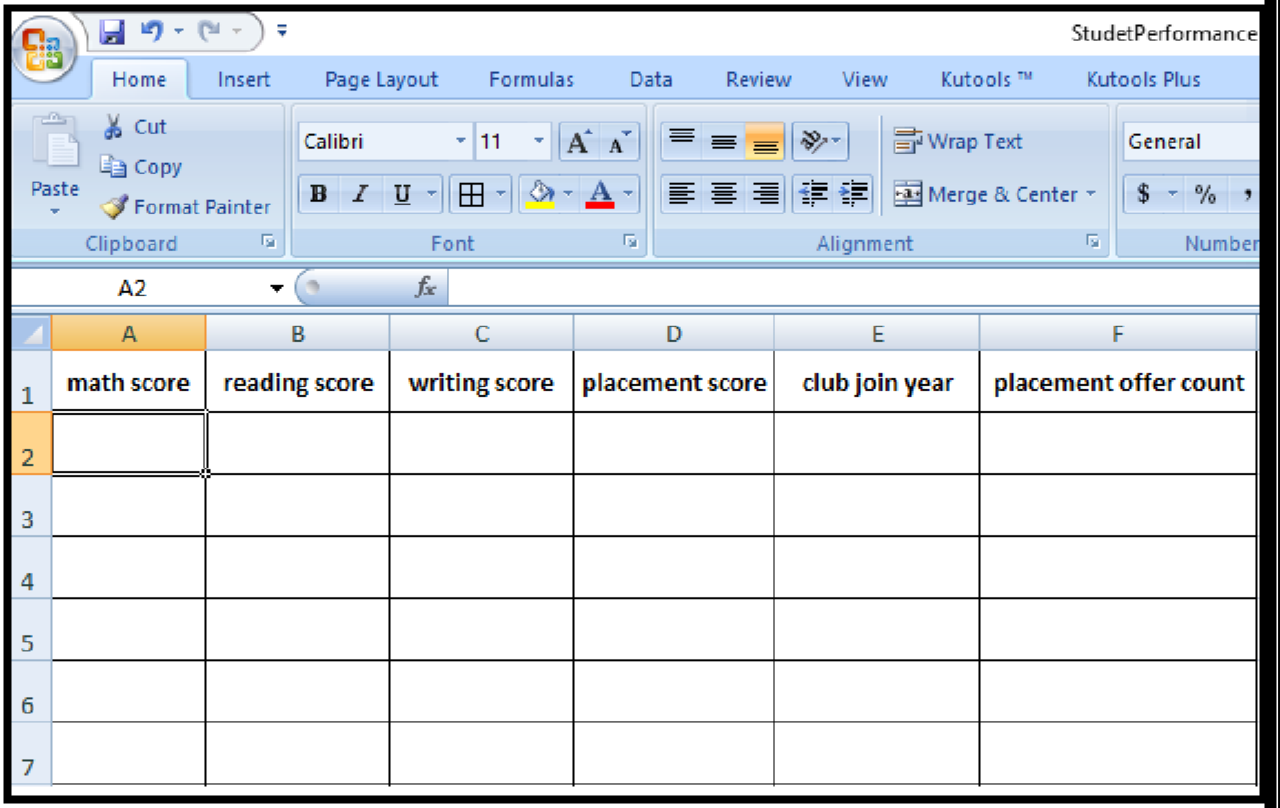
File name: StudetPerformance

Save as type: CSV (MS-DOS)

Authors: home

Tags: Add a tag

Step 3: Enter the name of features as column header.

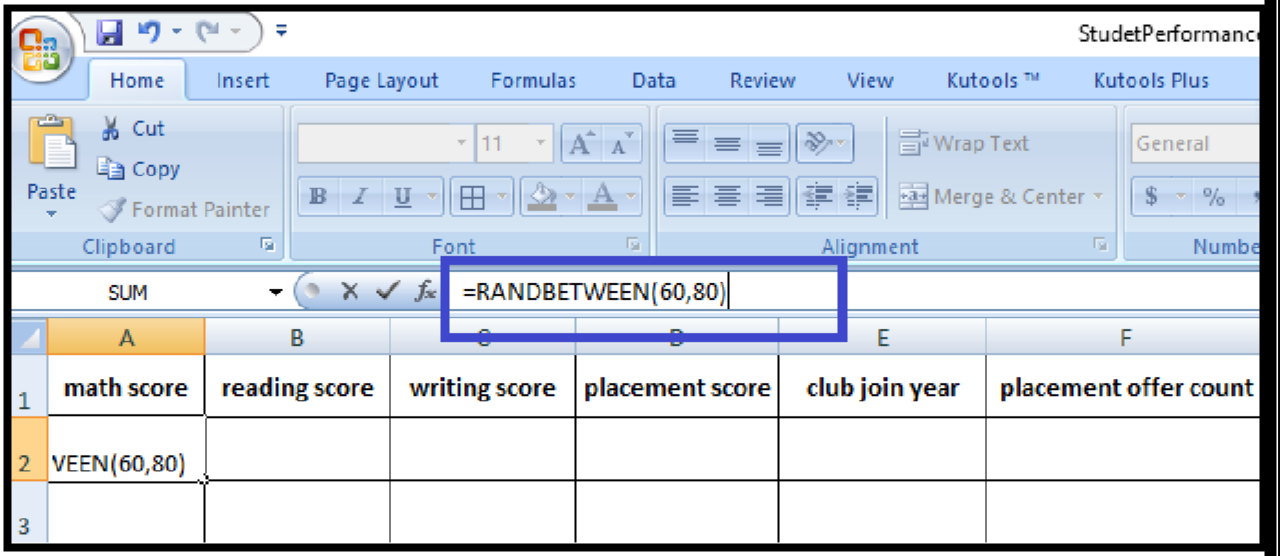


A2

	A	B	C	D	E	F
1	math score	reading score	writing score	placement score	club join year	placement offer count
2						
3						
4						
5						
6						
7						

Step 3: Fill the dara by using **RANDOMBETWEEN** function. For every feature , fill the data by considering above spectified range.

one example is given:



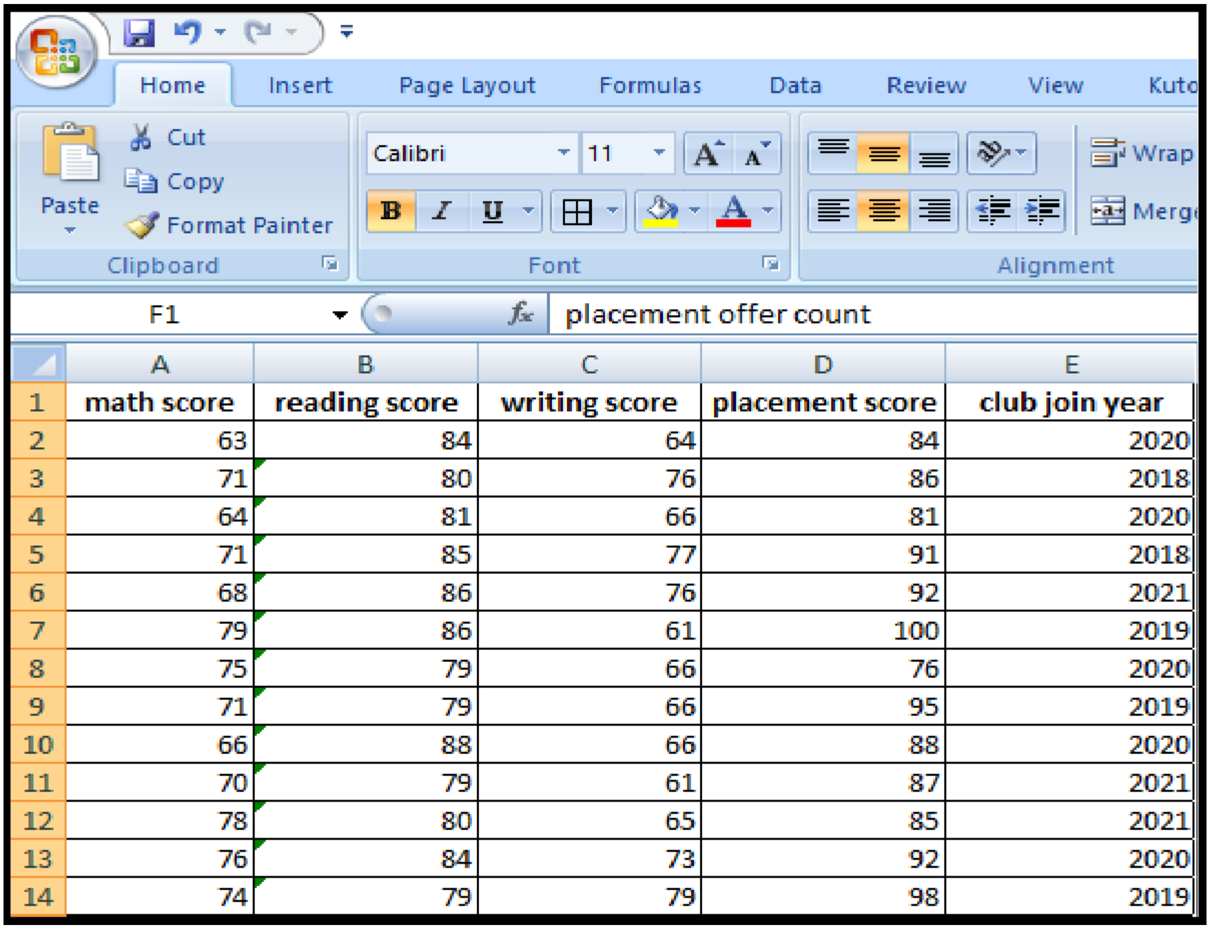
SUM

=RANDBETWEEN(60,80)

	A	B	C	D	E	F
1	math score	reading score	writing score	placement score	club join year	placement offer count
2	VEEN(60,80)					
3						

Scroll down the cursor for 30 rows to create 30 instances.

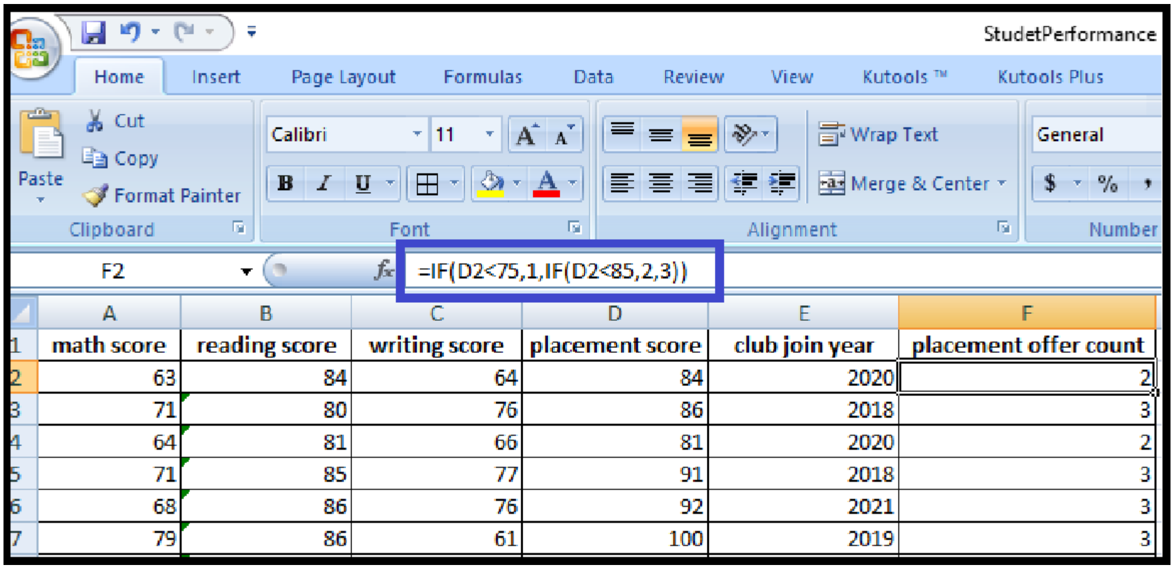
Repeat this for the features, Reading_Score, Writing_Score, Placement_Score, Club_Join_Date.



F1 placement offer count

	A	B	C	D	E
1	math score	reading score	writing score	placement score	club join year
2	63	84	64	84	2020
3	71	80	76	86	2018
4	64	81	66	81	2020
5	71	85	77	91	2018
6	68	86	76	92	2021
7	79	86	61	100	2019
8	75	79	66	76	2020
9	71	79	66	95	2019
10	66	88	66	88	2020
11	70	79	61	87	2021
12	78	80	65	85	2021
13	76	84	73	92	2020
14	74	79	79	98	2019

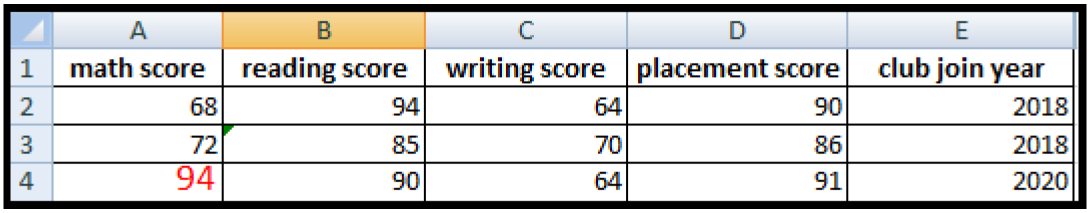
The placement count largely depends on the placement score. It is considered that if placement score <75, 1 offer is facilitated; for placement score >75 , 2 offer is facilitated and for else (>85) 3 offer is facilitated. Nested If formula is used for ease of data filling.



F2 =IF(D2<75,1,IF(D2<85,2,3))

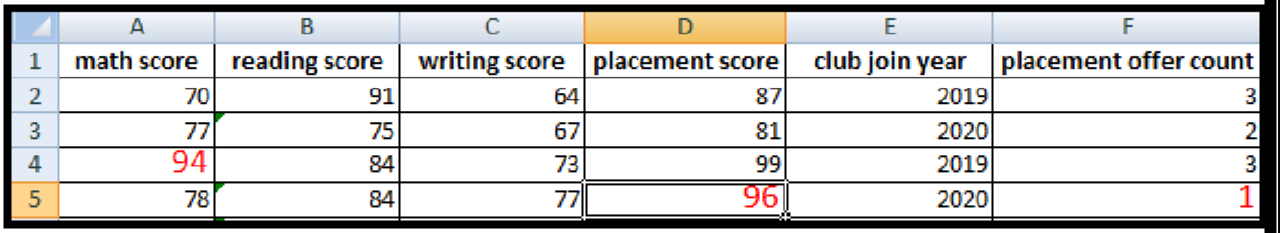
	A	B	C	D	E	F
1	math score	reading score	writing score	placement score	club join year	placement offer count
2	63	84	64	84	2020	2
3	71	80	76	86	2018	3
4	64	81	66	81	2020	2
5	71	85	77	91	2018	3
6	68	86	76	92	2021	3
7	79	86	61	100	2019	3

Step 4: In 20% data, fill the impurities. The range of math score is [60,80], updating a few instances values below 60 or above 80. Repeat this for Writing_Score [60,80], Placement_Score[75-100], Club_Join_Date [2018-2021].



	A	B	C	D	E
1	math score	reading score	writing score	placement score	club join year
2	68	94	64	90	2018
3	72	85	70	86	2018
4	94	90	64	91	2020

Step 5: To violate the ruleof response variable, update few valus . If placement scoreis greater then 85, facilated only 1 offer.



	A	B	C	D	E	F
1	math score	reading score	writing score	placement score	club join year	placement offer count
2	70	91	64	87	2019	3
3	77	75	67	81	2020	2
4	94	84	73	99	2019	3
5	78	84	77	96	2020	1

The dataset is created with the given description.

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:

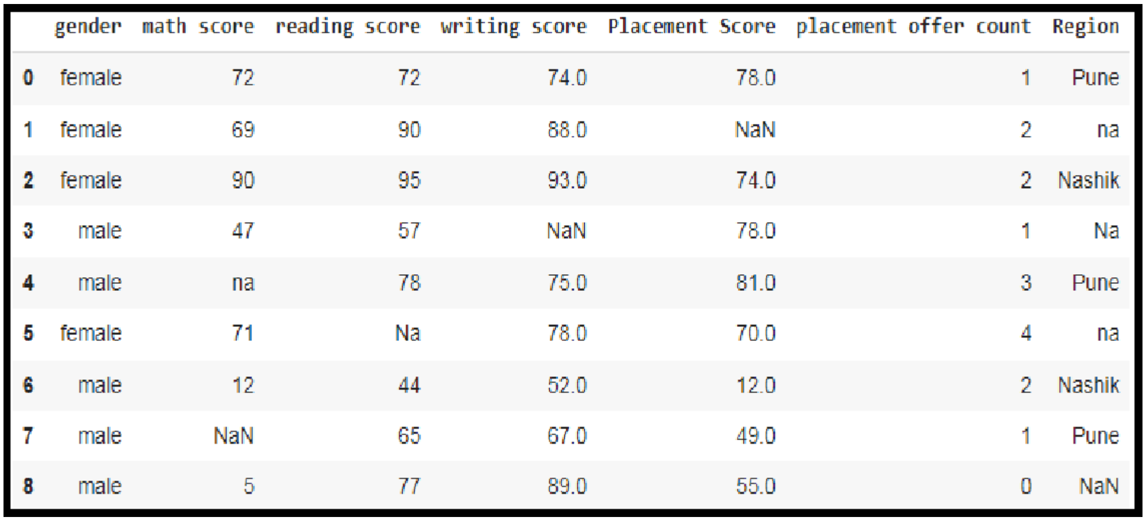
Step 1 : Import pandas and numpy in order to check missing values in Pandas DataFrame

```
import pandas as pd

import numpy as np
```

Step 2: Load the dataset in dataframe object df

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

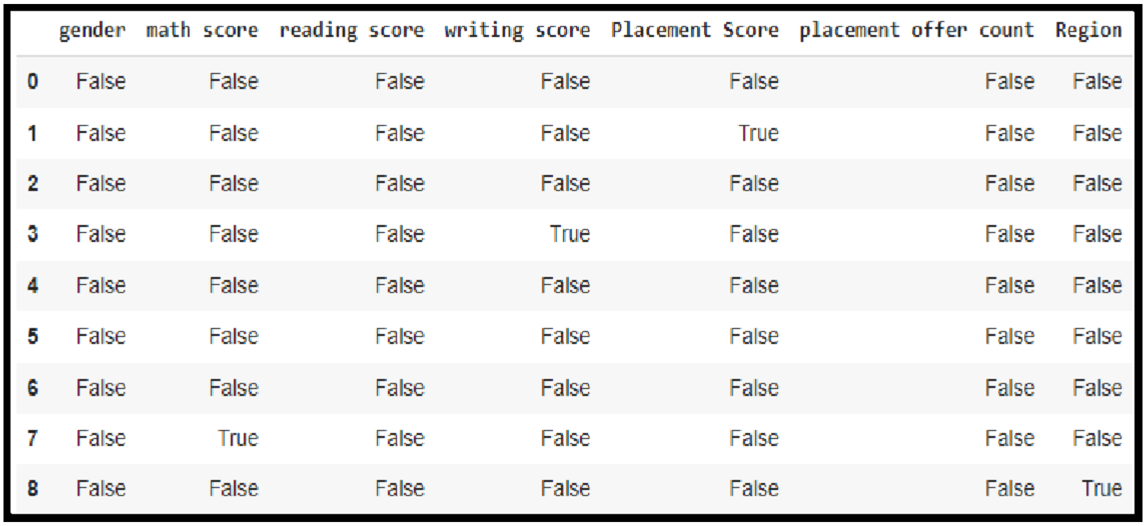
Step 3: Display the data frame



	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

df

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

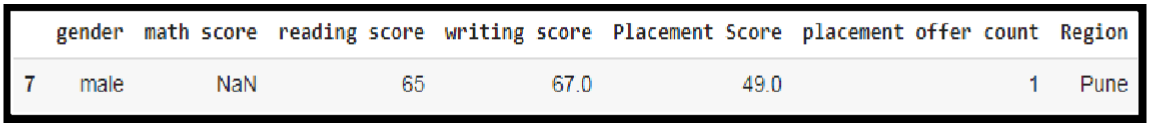


	gender	math score	reading score	writing score	Placement Score	placement offer count	Region
0	False	False	False	False	False	False	False
1	False	False	False	False	True	False	False
2	False	False	False	False	False	False	False
3	False	False	False	True	False	False	False
4	False	False	False	False	False	False	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

```
df.isnull()
```

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

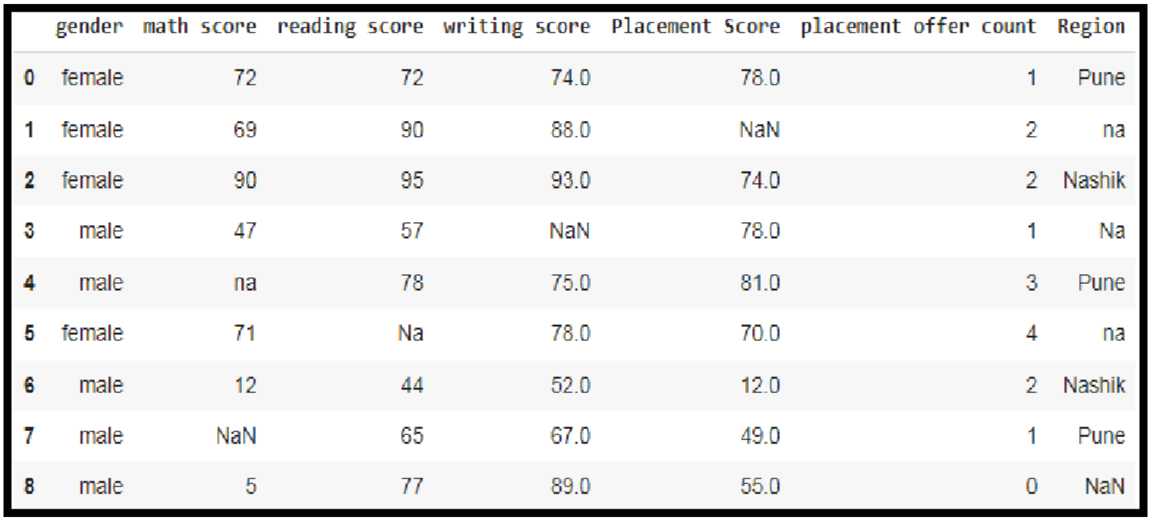
```
import pandas as pd

import numpy as np
```

Step 2: Load the dataset in dataframe object df

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

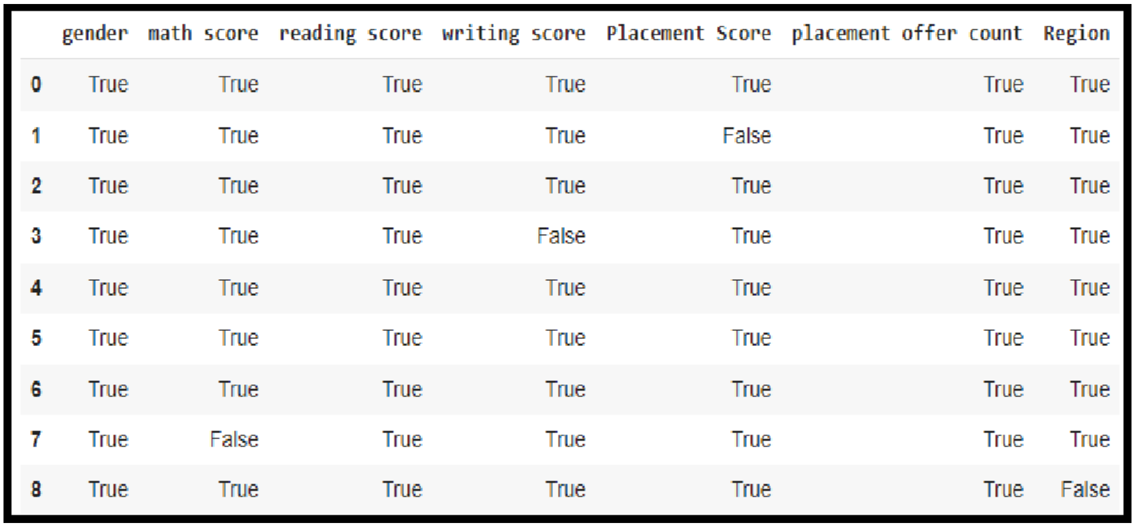
Step 3: Display the data frame

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.

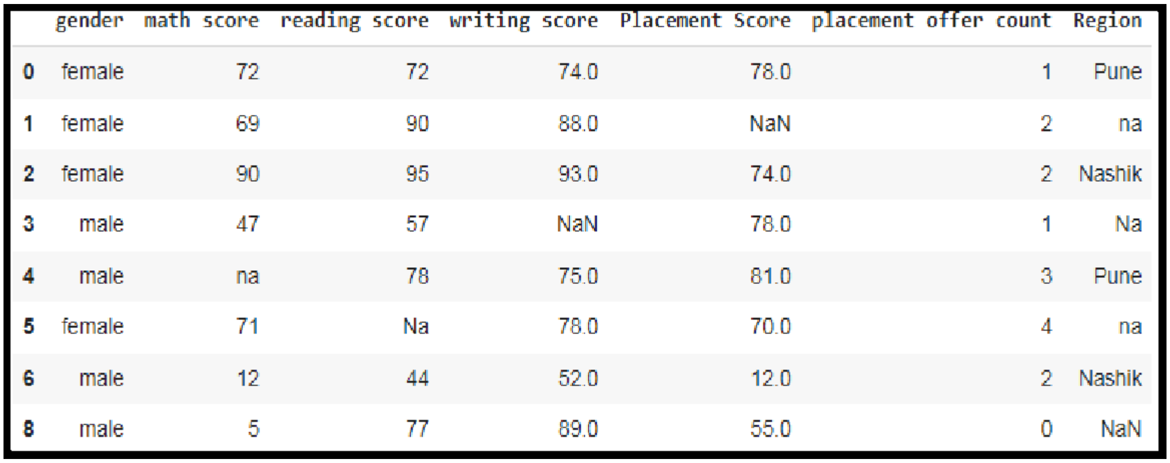


	gender	math score	reading score	writing score	Placement Score	placement offer count	Region
0	True	True	True	True	True	True	True
1	True	True	True	True	False	True	True
2	True	True	True	True	True	True	True
3	True	True	True	False	True	True	True
4	True	True	True	True	True	True	True
5	True	True	True	True	True	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

```
df.notnull()
```

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

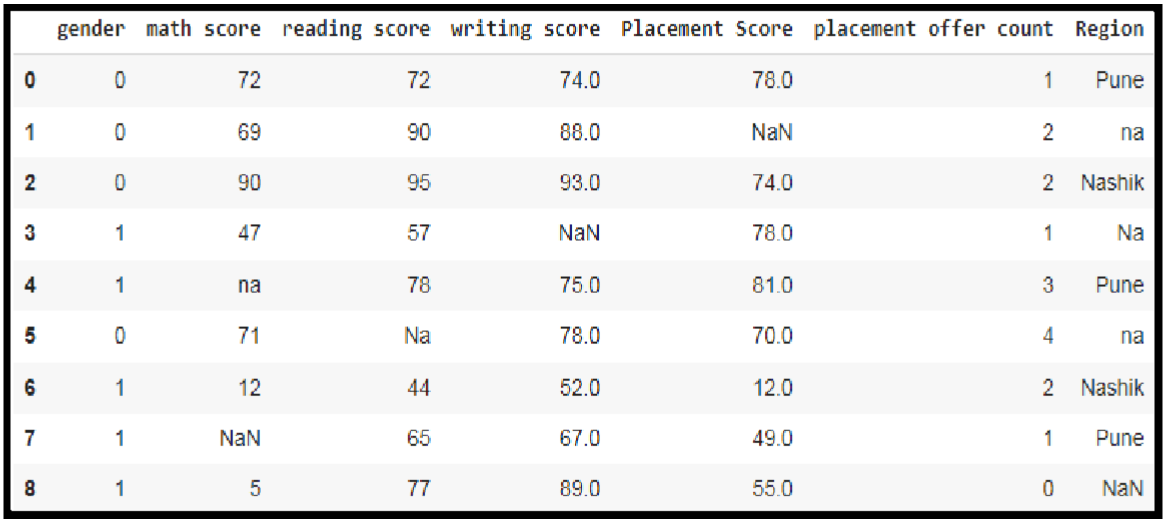
```
series1 = pd.notnull(df["math score"])df[series1]
```



	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
8	male	5	77	89.0	55.0	0	NaN

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 LabelEncoderle =
LabelEncoder()

df['gender'] = le.fit_transform(df['gender'])newdf=df
```



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

```
df
```

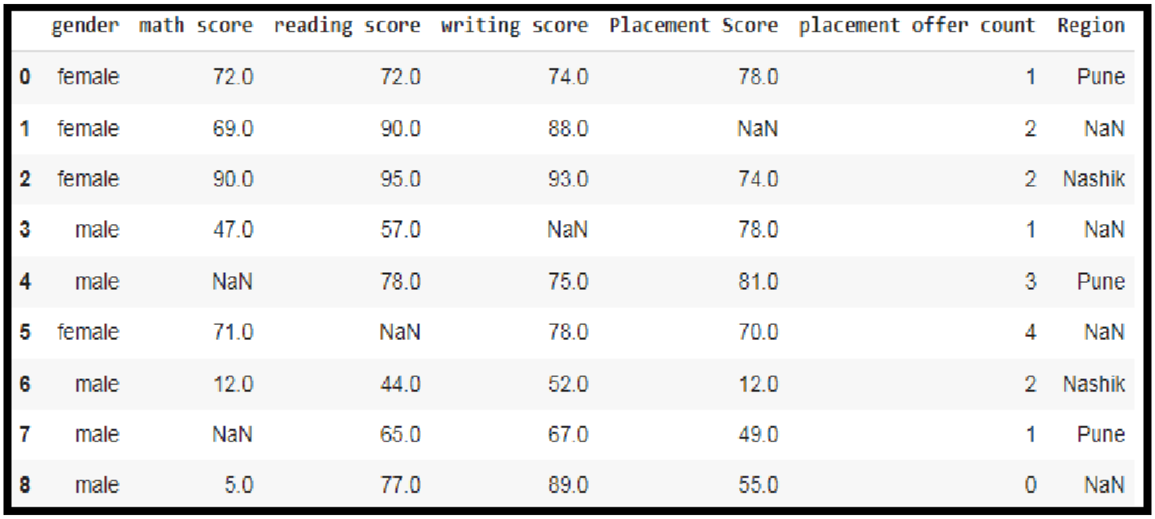
2. Filling missing values using dropna(), fillna(), replace()

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)
```



	gender	math score	reading score	writing score	Placement Score	placement offer count	Region
0	female	72.0	72.0	74.0	78.0	1	Pune
1	female	69.0	90.0	88.0	NaN	2	NaN
2	female	90.0	95.0	93.0	74.0	2	Nashik
3	male	47.0	57.0	NaN	78.0	1	NaN
4	male	NaN	78.0	75.0	81.0	3	Pune
5	female	71.0	NaN	78.0	70.0	4	NaN
6	male	12.0	44.0	52.0	12.0	2	Nashik
7	male	NaN	65.0	67.0	49.0	1	Pune
8	male	5.0	77.0	89.0	55.0	0	NaN

```
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

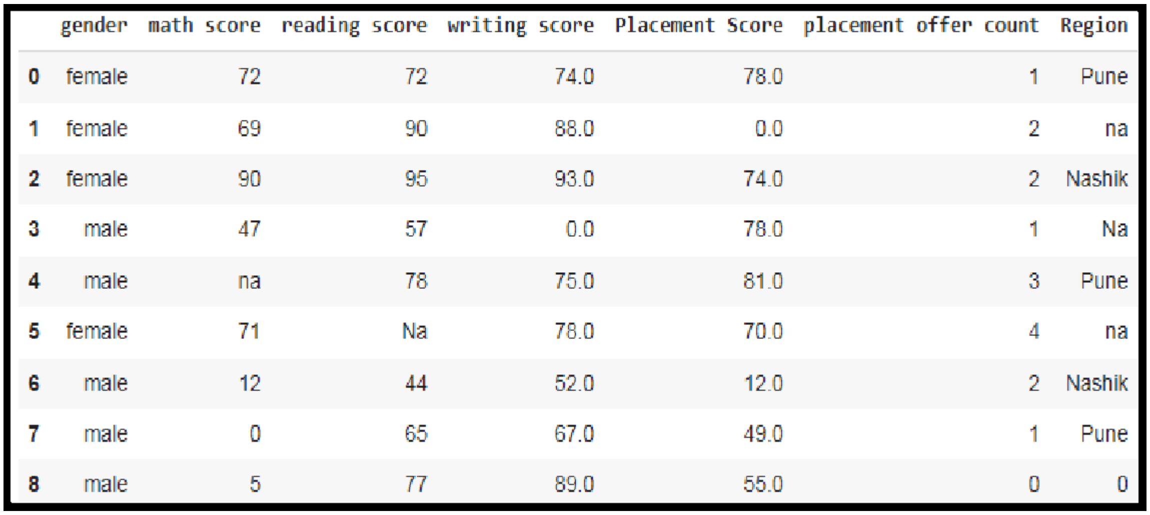
```
df=pd.read_csv("/content/StudentsPerformanceTest1.csv")
```

Step 3: Display the data frame

```
df
```

Step 4: filling missing value using fillna()

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



	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	0.0	2	na
2	female	90	95	93.0	74.0	2	Nashik
3	male	47	57	0.0	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	0	65	67.0	49.0	1	Pune
8	male	5	77	89.0	55.0	0	0

Step 5: filling missing values using mean, median and standard deviation of thatcolumn.

```
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())
```

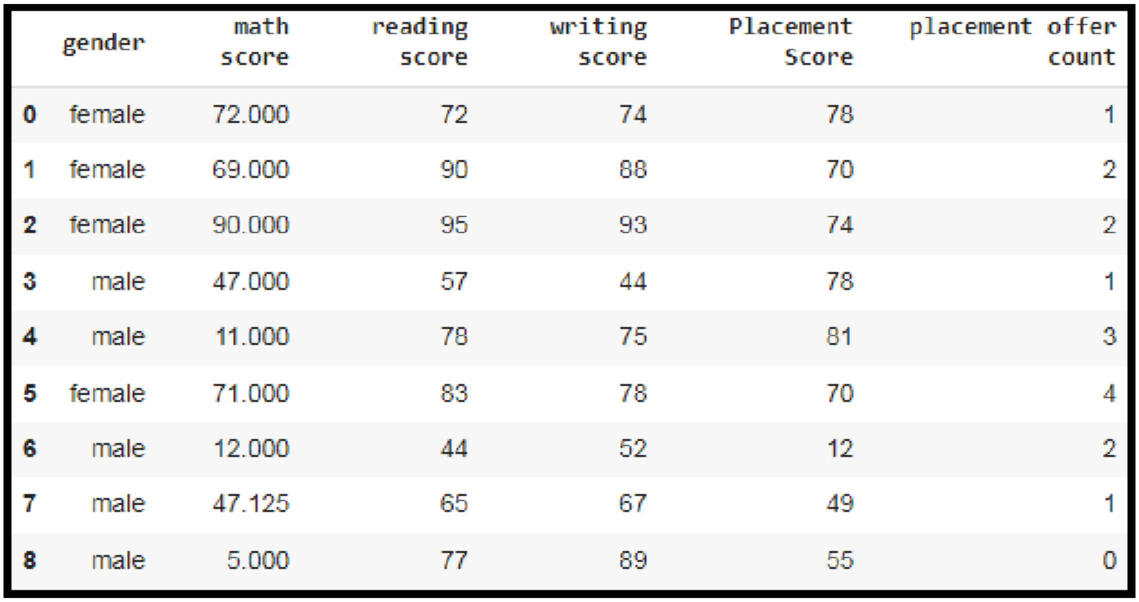
replacing missing values in forenoon column with minimum/maximum number of that column

```
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
```

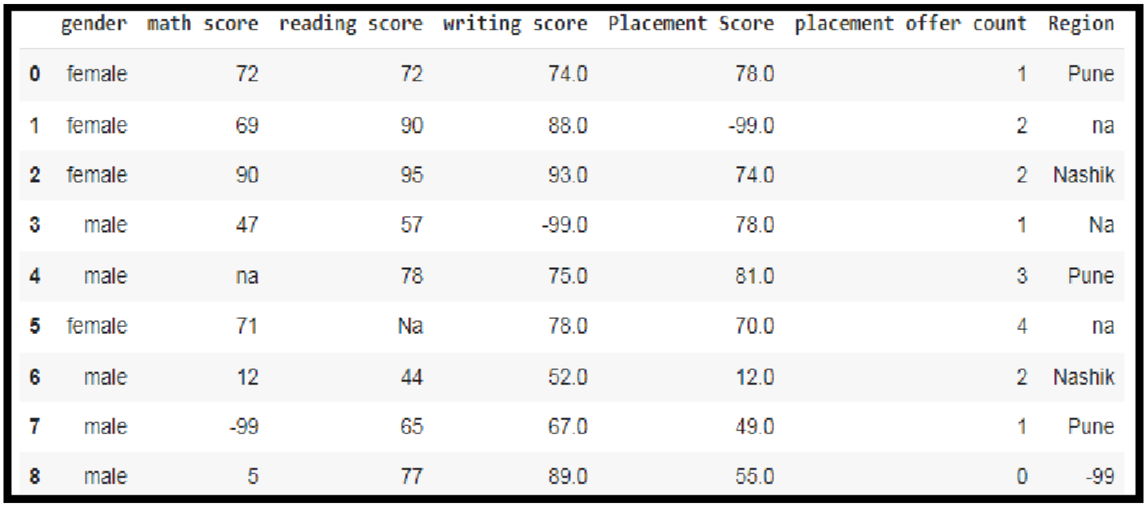


	gender	math score	reading score	writing score	Placement Score	placement offer count
0	female	72.000	72	74	78	1
1	female	69.000	90	88	70	2
2	female	90.000	95	93	74	2
3	male	47.000	57	44	78	1
4	male	11.000	78	75	81	3
5	female	71.000	83	78	70	4
6	male	12.000	44	52	12	2
7	male	47.125	65	67	49	1
8	male	5.000	77	89	55	0

- Filling a null values using replace() method

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

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



	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	-99.0	2	na
2	female	90	95	93.0	74.0	2	Nashik
3	male	47	57	-99.0	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	-99	65	67.0	49.0	1	Pune
8	male	5	77	89.0	55.0	0	-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 pdimport

numpy as np
```

Step 2: Load the dataset in dataframe object df

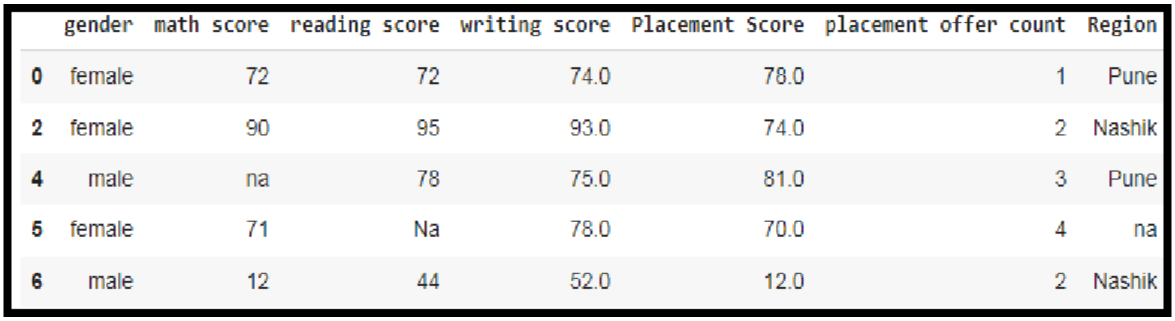
```
df=pd.read_csv("/content/StudentsPerformanceTest1.csv")
```

Step 3: Display the data frame

df

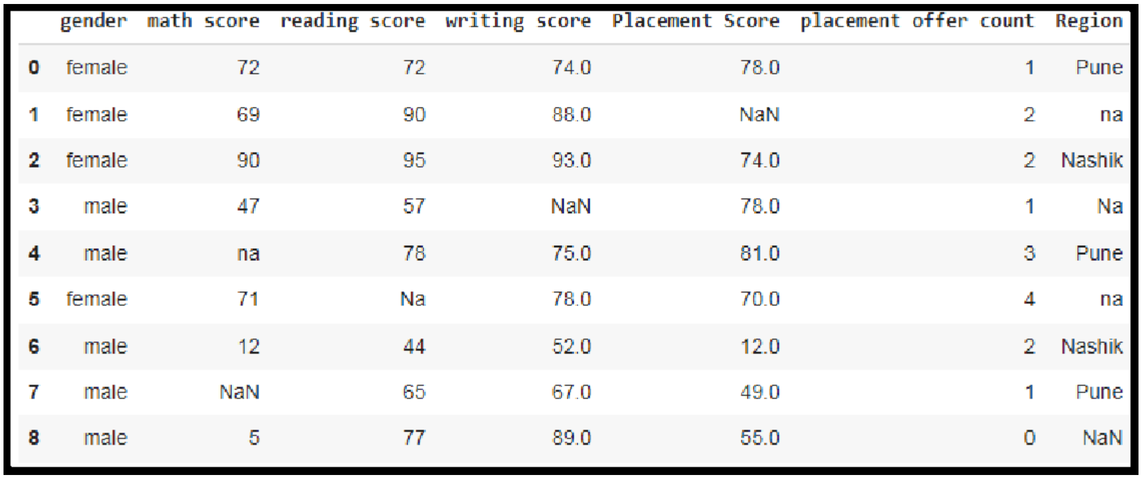
Step 4:To drop rows with at least 1 null value

ndf.dropna()



	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

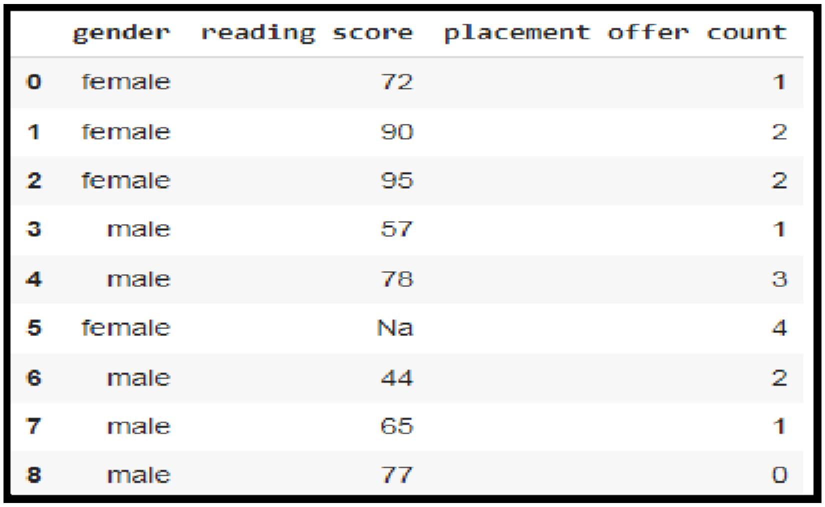
Step 5: To Drop rows if all values in that row are missing



	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

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

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



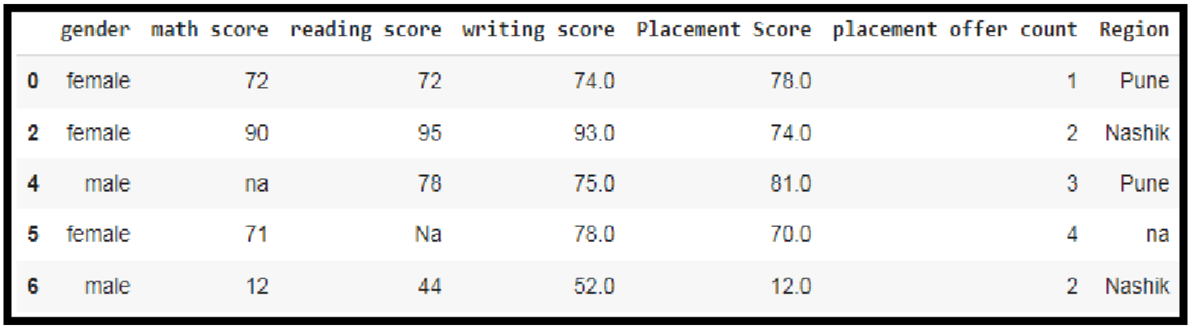
	gender	reading score	placement offer count
0	female	72	1
1	female	90	2
2	female	95	2
3	male	57	1
4	male	78	3
5	female	Na	4
6	male	44	2
7	male	65	1
8	male	77	0

```
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')
```



	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

```
new_data
```

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?

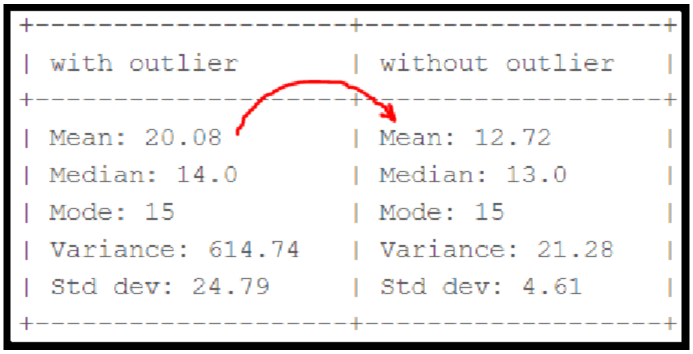
In statistics, we have three measures of central tendency namely Mean, Median, and Mode. They help us describe the data.

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 ½ 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(quartiles, median, and outliers) into the dataset by just looking at its boxplot.

Algorithm:

Step 1 : Import pandas and numpy libraries

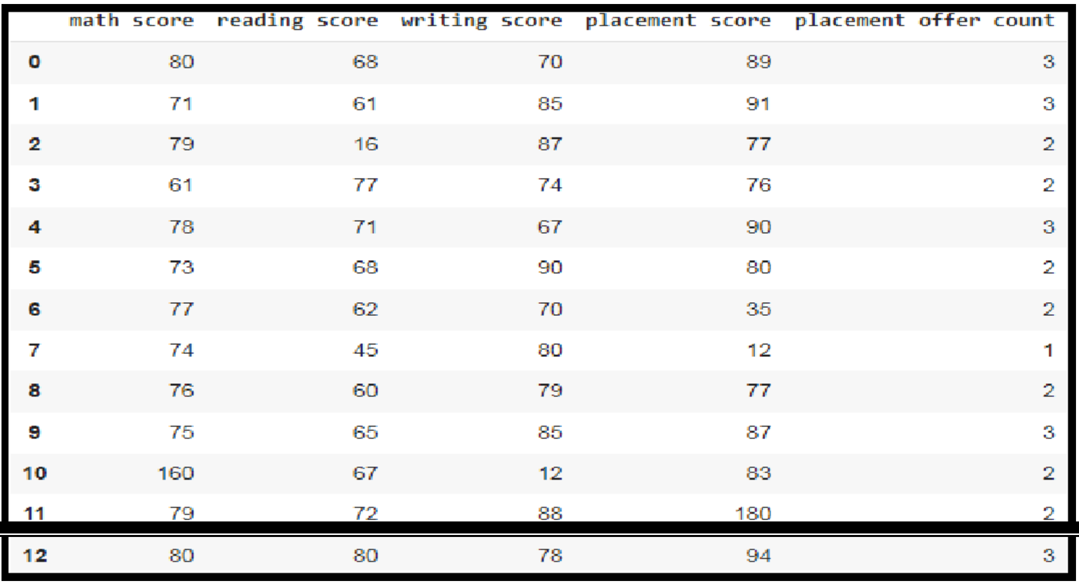
```
import pandas as pd

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

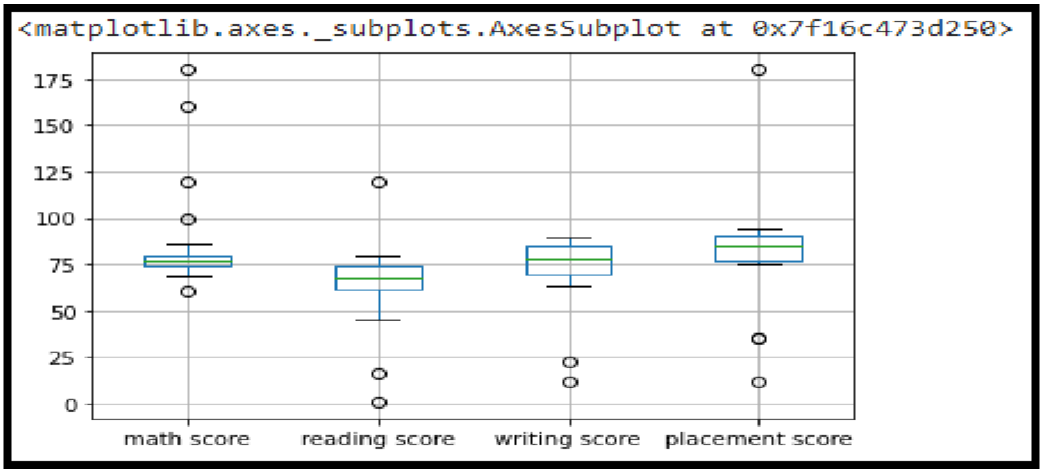
```
df
```



	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' , 'writingscore','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 pdimport

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

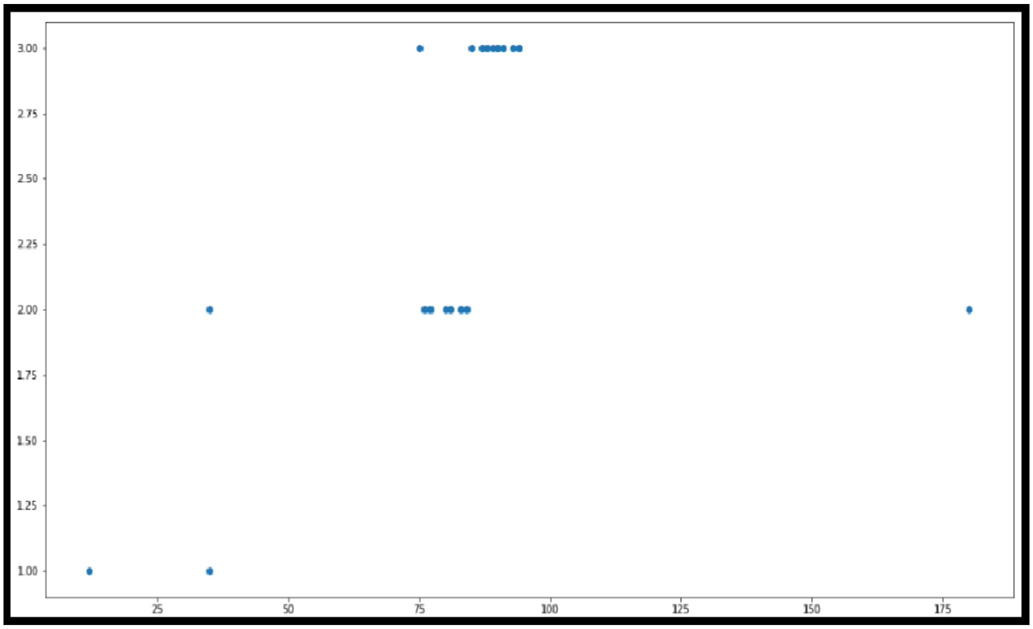
```
df
```

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 businessacres)/(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['placementoffer
count']>1)))
```

```
print(np.where((df['placement score']>85) & (df['placementoffer 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.

Zscore = (data_point -mean) / std. deviation

Algorithm:

Step 1 : Import numpy and stats from scipy libraries

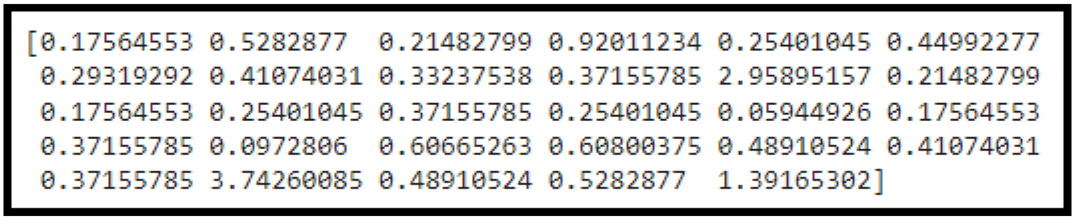
```
import numpy as np

from scipy import stats
```

Step 2: Calculate Z-Score for mathscore 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)
```



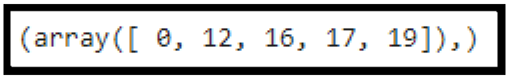
```
[0.17564553 0.5282877  0.21482799 0.92011234 0.25401045 0.44992277
 0.29319292 0.41074031 0.33237538 0.37155785 2.95895157 0.21482799
 0.17564553 0.25401045 0.37155785 0.25401045 0.05944926 0.17564553
 0.37155785 0.0972806  0.60665263 0.60800375 0.48910524 0.41074031
 0.37155785 3.74260085 0.48910524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

IQR = Quartile3 – 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*IQR value is considered) :

upper = Q3 +1.5*IQR

lower = Q1 – 1.5*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.

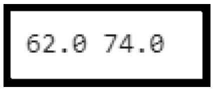
```
sorted_rscore= sorted(df['reading score'])
```

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)
```



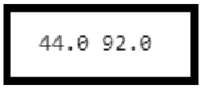
```
62.0 74.0
```

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.

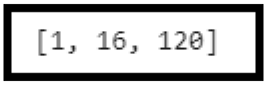
```
lwr_bound = q1-(1.5*IQR) upr_bound
= q3+(1.5*IQR) print(lwr_bound,
upr_bound)
```



```
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)
```



```
[1, 16, 120]
```

```
print(r_outliers)
```

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.

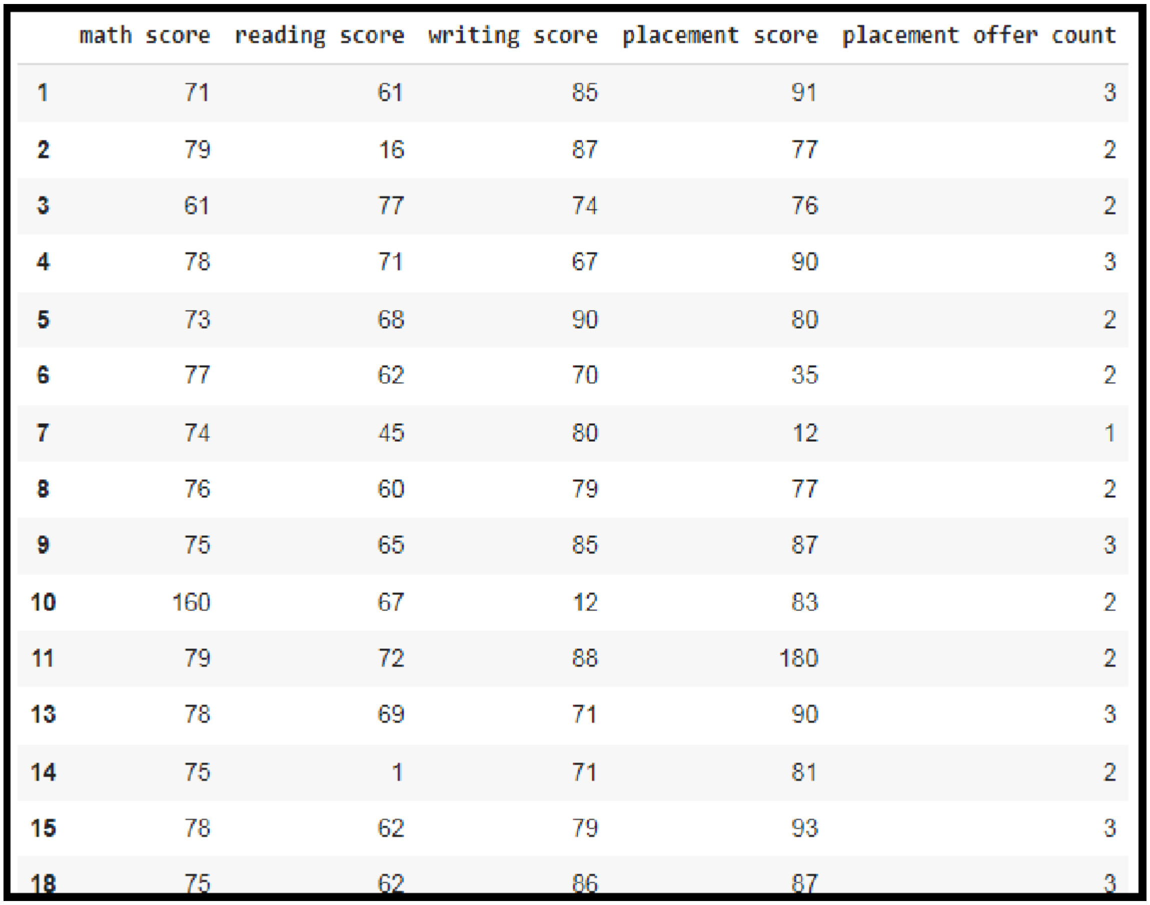
Below are some of the methods of treating the outliers

- 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)
```



	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

new_df

Here Sample_outliers are `(array([ 0, 12, 16, 17 ]),)` So instances with index 0, 12 ,16 and 17 are deleted.

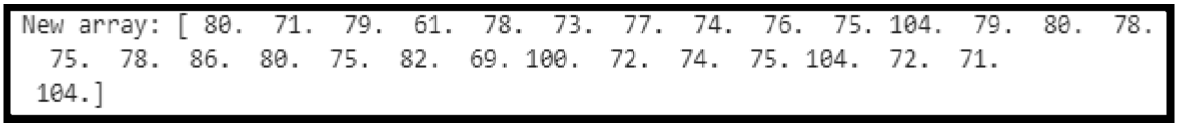
- Quantile based flooring and capping:

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)
```

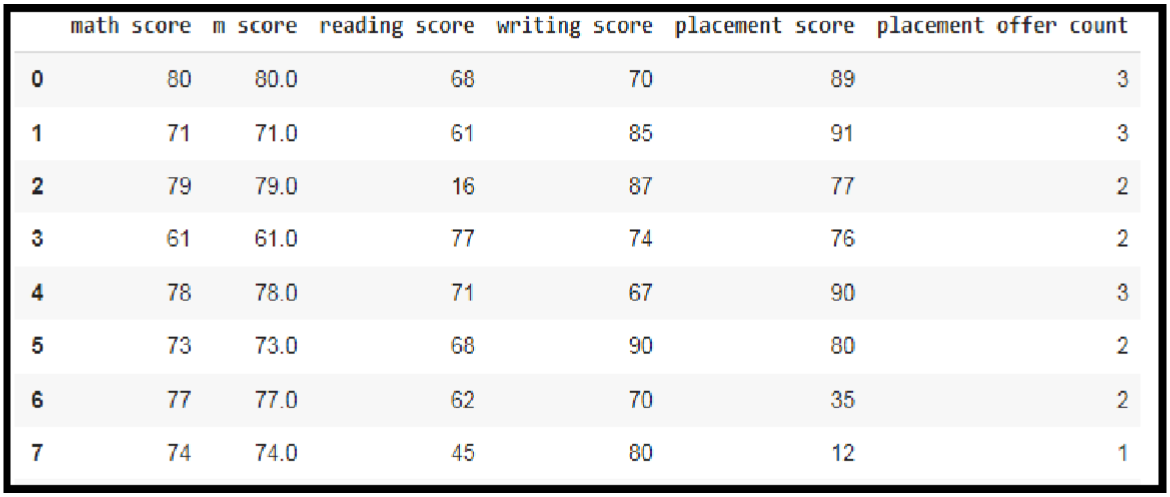
```
b = np.where(df_stud['math score']>ninetieth_percentile,ninetieth_percentile,
df_stud['math score'])
```



```
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.]
```

```
print("New array:",b)
```

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



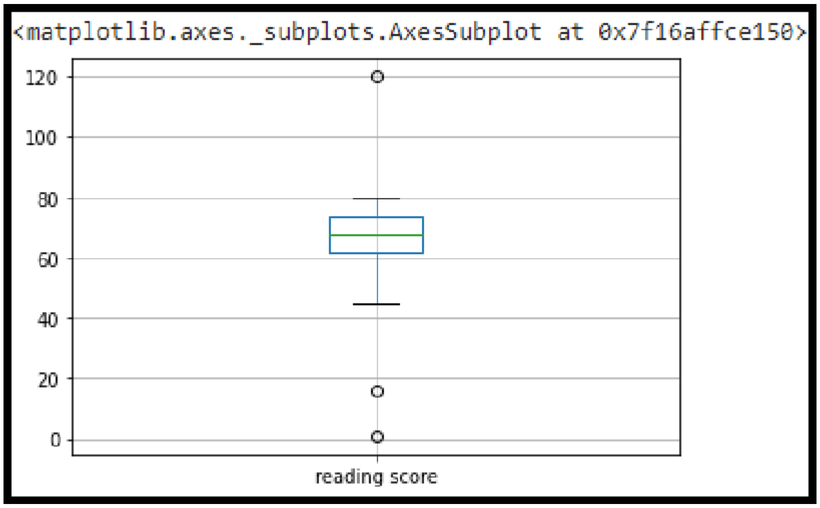
	math score	m score	reading score	writing score	placement score	placement offer count
0	80	80.0	68	70	89	3
1	71	71.0	61	85	91	3
2	79	79.0	16	87	77	2
3	61	61.0	77	74	76	2
4	78	78.0	71	67	90	3
5	73	73.0	68	90	80	2
6	77	77.0	62	70	35	2
7	74	74.0	45	80	12	1

- 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

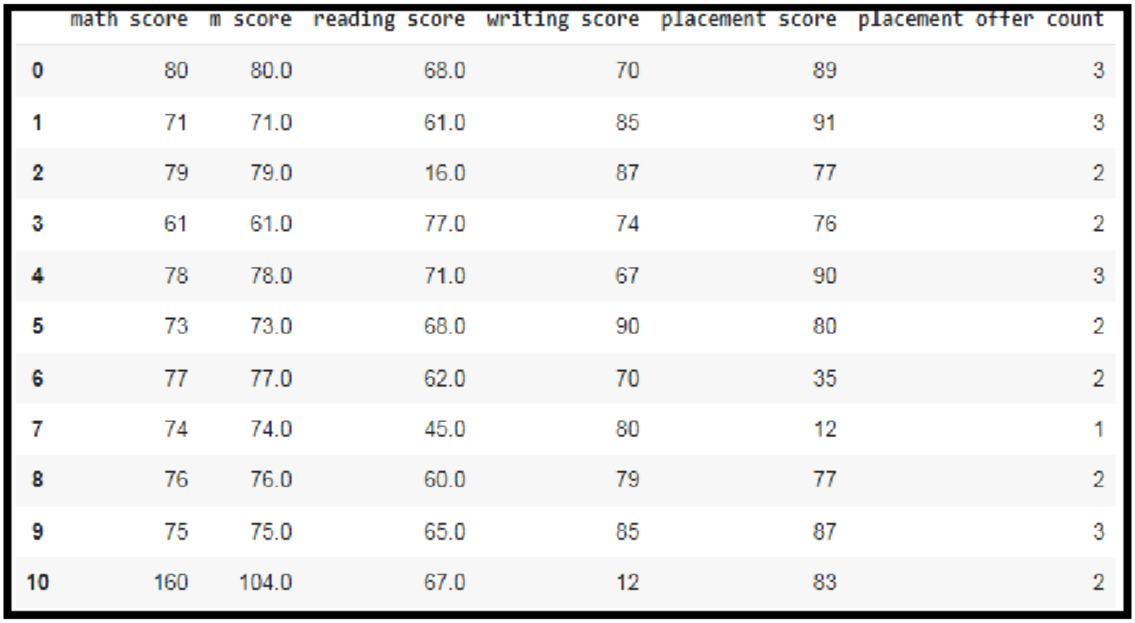
median=np.median(sorted_rscore)median

4. Replace the upper bound outliers using median value

   refined_df=df

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

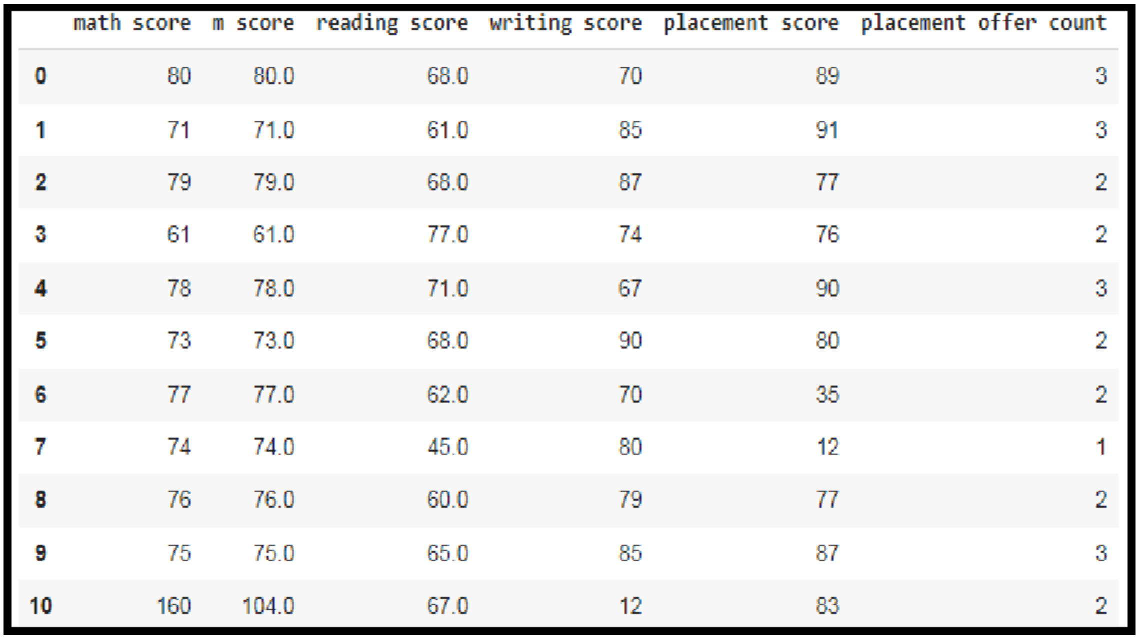
5. Display redefined_df



	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['readingscore'] <lwr_bound, median,refined_df['reading score'])

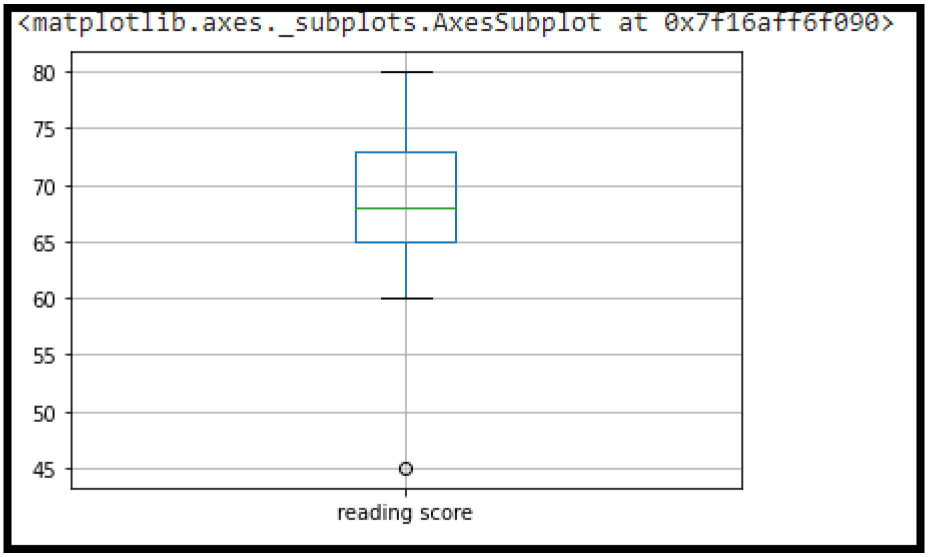
7. Display redefined_df



	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)
```



```
<matplotlib.axes._subplots.AxesSubplot at 0x7f16aff6f090>
```

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 date lake or another repository for use in business intelligence and analytics applications. The transformation 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:

a. 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

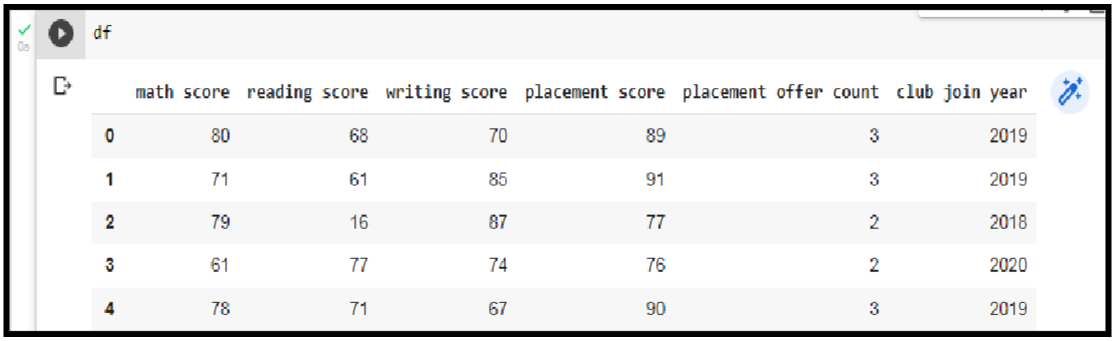
```
import pandas as pd

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

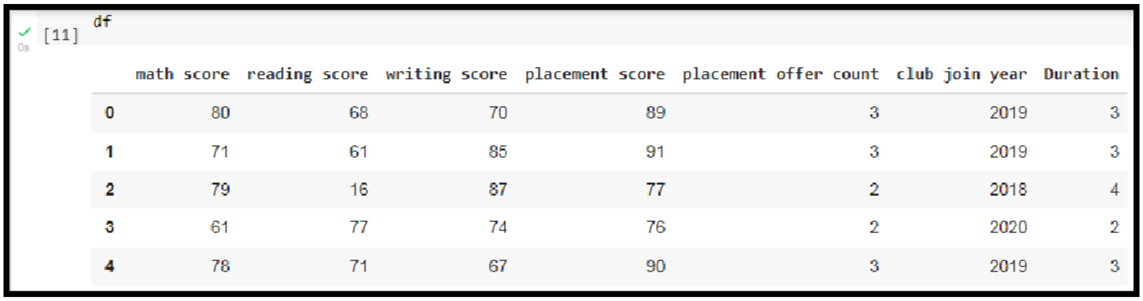
```
df
```



```
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

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



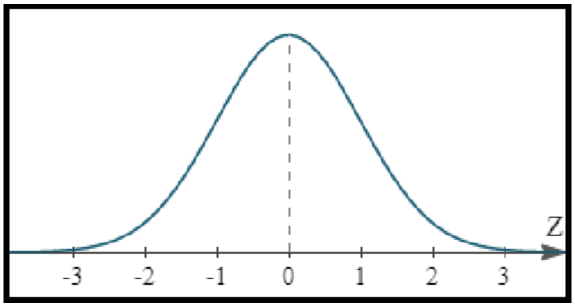
```
df
```

	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

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.

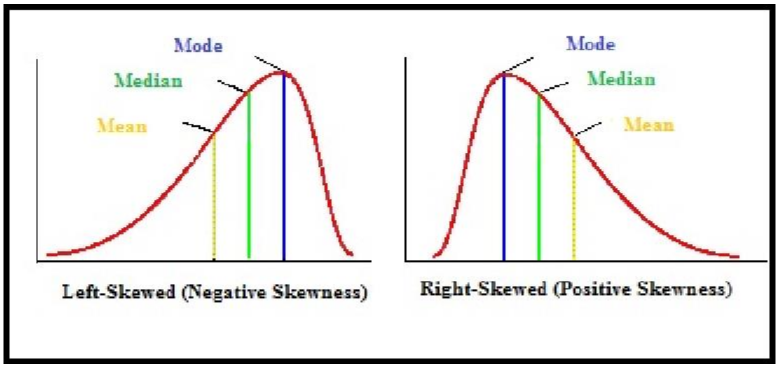
Normal Distribution: In a normal distribution, the graph appears as a classical, symmetrical “bell-shaped curve.” The mean, or average, and the mode, or maximum point on the curve, are equal.



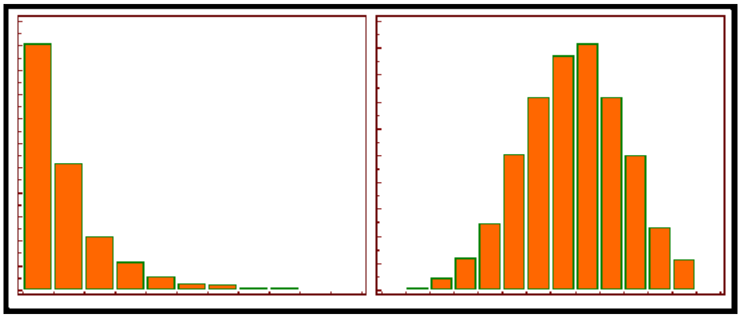
Positively Skewed Distribution

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.


Mode
Median
Mean
Left-Skewed (Negative Skewness)
Mode
Median
Mean
Right-Skewed (Positive Skewness)

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 andremove the outliers.

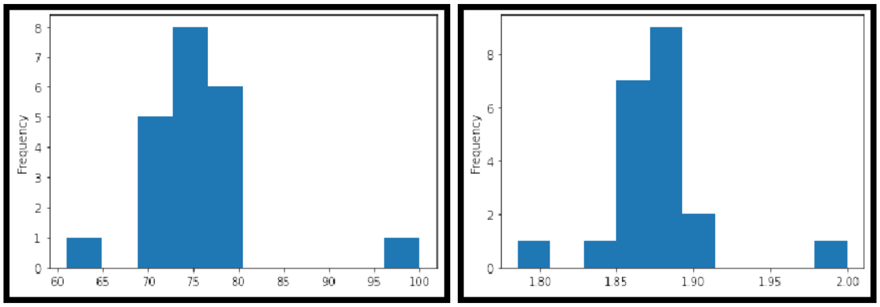
Step 2: Observe the histogram for math_score variable.

```
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.

Assignment Question:

1. **Explain the methods to detect the outlier.**
2. Explain data transformation methods
3. **Write the algorithm to display the statistics of Null values present in the dataset.**
4. Write an algorithm to replace the outlier value with the mean of the variable.

.

Assignment No: 3

Title of the Assignment: Descriptive Statistics - Measures of Central Tendency and variability

Perform the following operations on any open source dataset (e.g., data.csv)

1. Provide summary statistics (mean, median, minimum, maximum, standard deviation) for a dataset (age, income etc.) with numeric variables grouped by one of the qualitative (categorical) variables. 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.

2. Write a Python program to display some basic statistical details like percentile, mean, standard deviation etc. of the species of 'Iris-setosa', 'Iris-versicolor' and 'Iris- versicolor' of iris.csv dataset.

Provide the codes with outputs and explain everything that you do in this step.

Objective of the Assignment: Students should be able to perform the Statistical operations using Python on any open source dataset.

Prerequisite:

1. Basic of Python Programming
2. Concept of statistics such as mean, median, minimum, maximum, standard deviation etc.

Assignment Questions:

1. Explain Measures of Central Tendency with examples.
2. What are the different types of variables? Explain with examples.
3. Which method is used to display statistics of the data frame? write the code.

Assignment No: 4

- -

Title of the Assignment: Create a Linear Regression Model using Python/R to predict home prices using Boston Housing Dataset (https://www.kaggle.com/c/boston-housing). The Boston Housing dataset contains information about various houses in Boston through different parameters. There are 506 samples and 14 feature variables in this dataset.

The objective is to predict the value of prices of the house using the given features.

Objective of the Assignment: Students should be able to data analysis using liner regression using Python for any open source dataset

----

Prerequisite:

1. Basic of Python Programming

2.Concept of Regresion.

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*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.

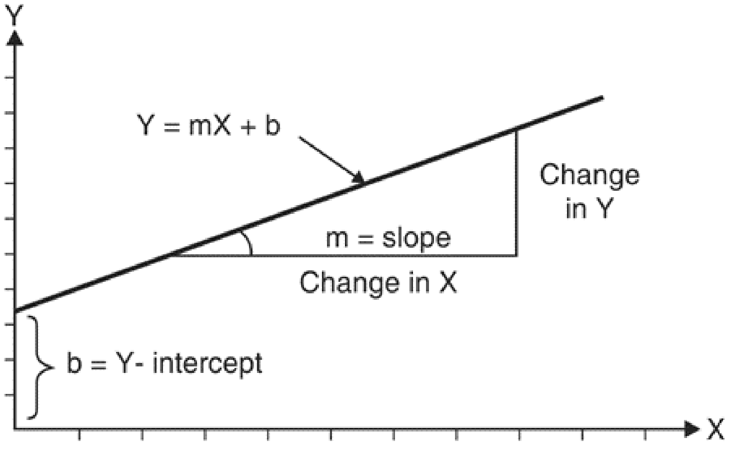
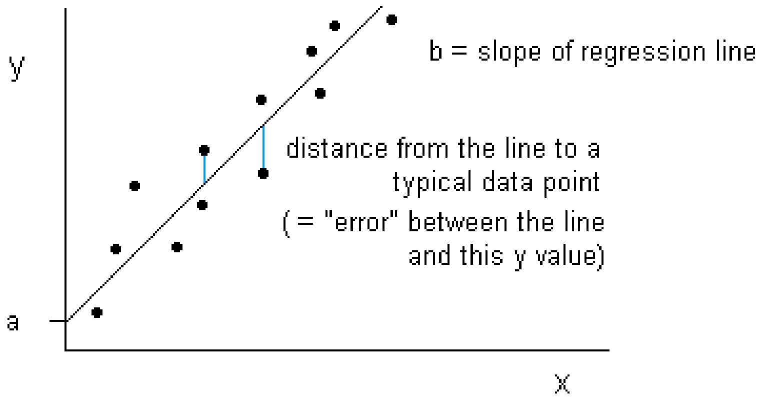


Fig. 1: geometry of linear regression

- Fig. 2 shown below is about the relation between weight (in Kg) and height (in cm), a linear relation. It is an approach of studying in a statistical manner to summarise and learn the relationships among continuous (quantitative) variables.
- Here a variable, denoted by ‘x’ is considered as the predictor, explanatory, or independent variable.

- Another variable, denoted ‘y’, is considered as the response, outcome, or dependent variable. While "predictor" and "response" used to refer to these variables.
- Simple linear regression technique concerned with the study of only one predictor variable.

Fig.2 : Relation between weight (in Kg) and height (in cm)


y
b = slope of regression line
distance from the line to a
typical data point
(= "error" between the line
and this y value)
a
x

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 is 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

  $$Y=a + bX + cX2$$

- 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

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

where $\hat{y}$ is the dependent or the response variable

x is the independent or the input variable

β_0 is the value of y when x=0 or the y intercept

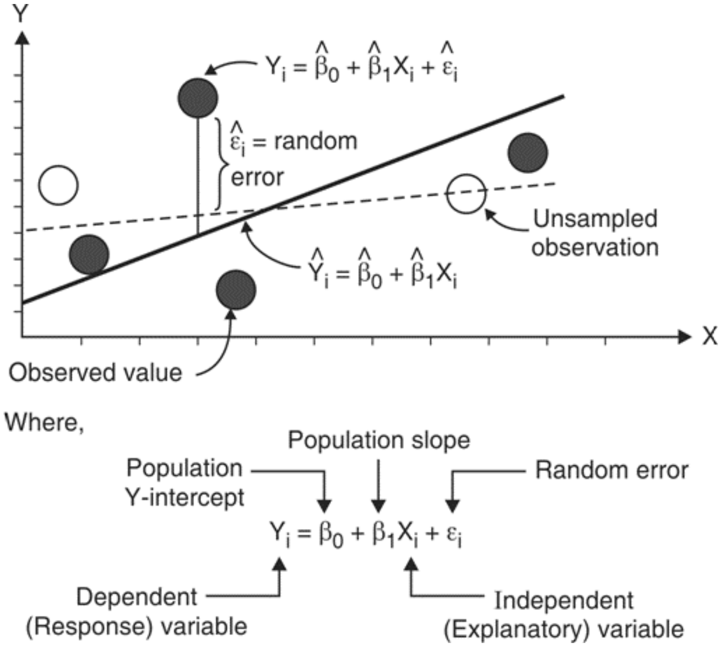
β_1 is the value of slope of the line ε is the error or the noise

- 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 β_0 and β_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.

$$min \sum_{i=0}^{n} (\hat{y} - y)^2 \qquad (2)$$


Y
$Y_i = \hat{\beta}_0 + \hat{\beta}_1 X_i + \hat{\varepsilon}_i$
$\hat{\varepsilon}_i$ = random error
Unsampled observation
$\hat{Y}_i = \hat{\beta}_0 + \hat{\beta}_1 X_i$
X
Observed value
Where,
Population slope
Population Y-intercept
Random error
$Y_i = \beta_0 + \beta_1 X_i + \varepsilon_i$
Dependent (Response) variable
Independent (Explanatory) variable

Using differential calculus on equation 1 we can find the values of β_0 and β_1 such that the sum of squares (that is equation 2) is minimum.

$$\beta_1 = \sum_{i=1}^{n} (x_i - \bar{x})(y_i - \bar{y}) / \sum_{i=1}^{n} (x_i - \bar{x})^2 \qquad (3)$$

$$\beta_0 = \bar{y} - \beta_1 \bar{x} \qquad (4)$$

Once the Linear Model is estimated using equations (3) and (4), we can estimate the value of the dependent variable in the given range only. Going outside the range is called extrapolation which is inaccurate if simple regression techniques are used.

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. When the linear regression model is trained using a given set of observations, the model with the least mean sum of squares error (MSE) is selected as the best model. The Python or R packages select the best-fit model as the model with the lowest MSE or lowest RMSE when training the linear regression models.

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).

$$MSE = \frac{1}{n}\Sigma\underbrace{\left(y - \widehat{y}\right)^2}_{\text{The square of the difference between actual and predicted}}$$

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

RMSE:

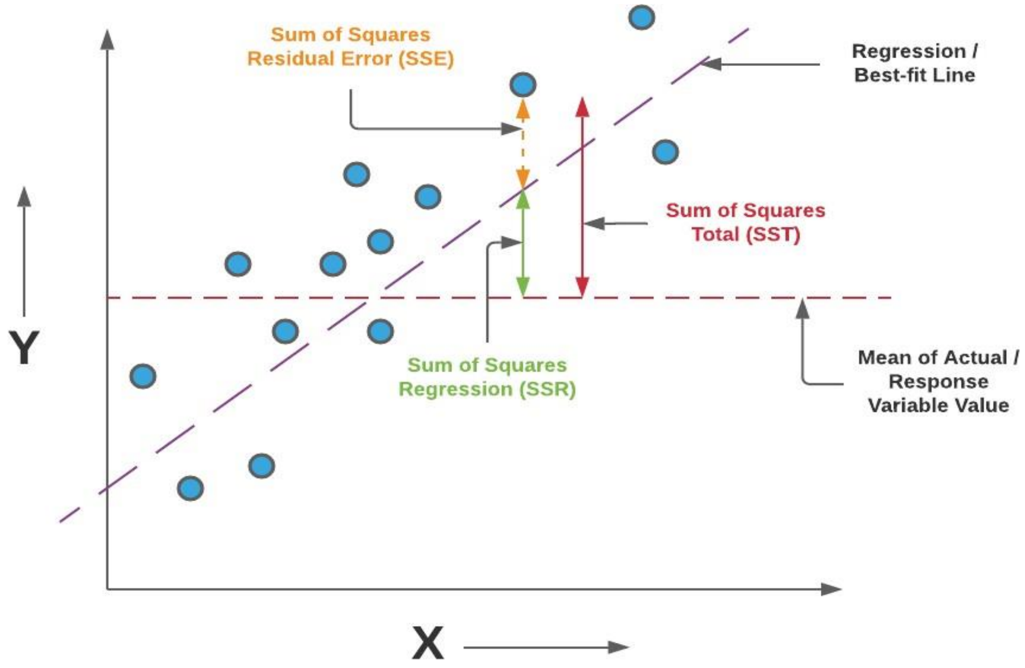
Root Mean Squared Error method that basically calculates the least-squares error and takes a root of the summed values.

Mathematically speaking, Root Mean Squared Error is the square root of the sum of all errors divided by the total number of values. This is the formula to calculate RMSE

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

RMSE - Least Squares Regression Method - Edureka

R-Squared :



R-Squared is the ratio of the sum of squares regression (SSR) and the sum of squares total (SST).

SST : total sum of squares (SST), regression sum of squares (SSR), Sum of square of errors (SSE) are all showing the variation with different measures.

$$SST = \sum_{i=1} (y_i - \bar{y})^2$$

$$SSR = \sum_{i=1}^{n} (\hat{y}_i - \bar{y})^2$$

$$SSE = \sum_{i=1}^{n} (y_i - \hat{y}_i)^2$$

$$R^2 = \frac{SSR}{SST} = \frac{\sum(\hat{y}_i - \bar{y})^2}{\sum(y_i - \bar{y})^2}$$

A value of R-squared closer to 1 would mean that the regression model covers most part of the variance of the values of the response variable and can be termed as a good model.

One can alternatively use MSE or R-Squared based on what is appropriate and the need of the hour. However, the disadvantage of using MSE rather than R-squared is that it will be difficult to gauge the performance of the model using MSE as the value of MSE can vary from 0 to any larger number. However, in the case of R-squared, the value is bounded between 0 and .

4. Example of Linear Regression

Consider following data for 5 students.

Each Xi (i = 1 to 5) represents the score of ith student in standard X and corresponding Yi (i = 1 to 5) represents the score of ith 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 (Xi)	Score in XII standard (Yi)
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
$\bar{x} = 78$	$\bar{y} = 77$			**Ɛ** $(x - \bar{x})^2$ **= 730**	**Ɛ** $(x - \bar{x})(y - \bar{y})$ **= 470**

(i) linear regression equation that best predicts standard XIIth score

$$\widehat{y} = \beta_0 + \beta_1 x$$

$$\beta_1 = \sum_{i=1}^{n}(x_i - x)(y_i - \bar{y}) / \sum_{i=1}^{n}(x_i - \bar{x})^2$$

$$\beta_1 = 470/730 = 0.644$$

$$\beta_0 = \bar{y} - \beta_1 \bar{x}$$

$$\beta_0 = 77 - (0.644 * 78) = 26.768$$

$$\widehat{y} = 26.76 + 0.644\, x$$

(ii) Interpretation of the regression line.

Interpretation 1

For an increase in value of x by 0.644 units there is an increase in value of y in one unit.

Interpretation 2

Even if x = 0 value of independent variable, it is expected that value of y is 26.768

Score in XII standard (Yi) is 0.644 units depending on Score in X standard (Xi) but other factors will also contribute to the result of XII standard by 26.768 .

(iii) If a student's score is 65 in std X, then his expected score in XII standard is 78.288

For x = 80 the y value will be

$$\hat{y} = 26.76 + 0.644 * 65 = 68.38$$

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.

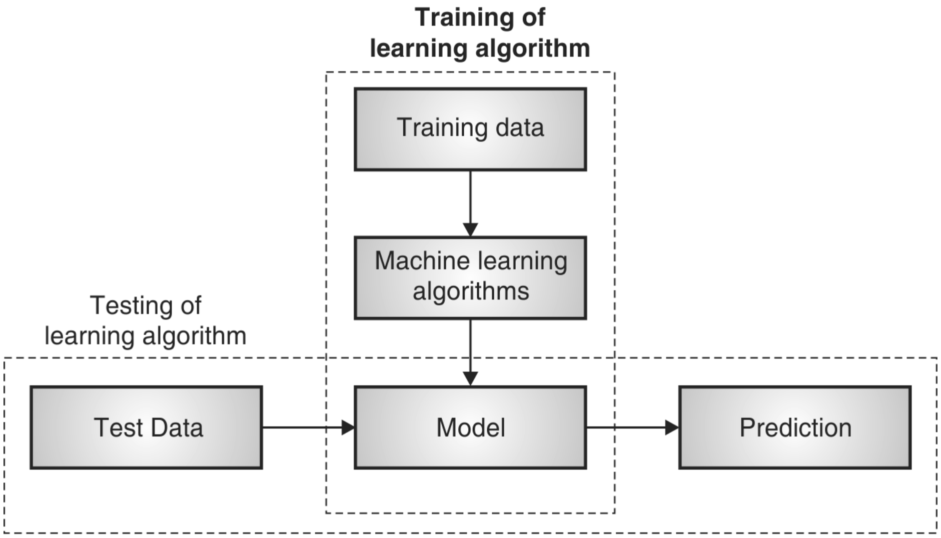


Fig. 1.3.1 : Training and Testing Phase in Machine Learning

(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.
- The fit method is called on the training set to build the model.
- This fit method is applied to the model on the test set to estimate the target value and evaluate the model's performance.
- The reason the data is divided into training and test sets is to use the test set to estimate how well the model trained on the training data and how well it would perform on the unseen data.

Algorithm (Synthesis Dataset):

Step 1: Import libraries and create alias for Pandas, Numpy and Matplotlib

```
import pandas as pd
import numpy as np

import matplotlib.pyplot as plt
```

Step 2: Create a Dataframe with Dependent Variable(x) and independent variable y.

```
x=np.array([95,85,80,70,60])

y=np.array([85,95,70,65,70])
```

Step 3 : Create Linear Regression Model using Polyfit Function:

```
model= np.polyfit(x, y, 1)
```

Step 4: Observe the coefficients of the model.

```
model
```

Output:

```
array([ 0.64383562, 26.78082192])
```

Step 5: Predict the Y value for X and observe the output.

```
predict = np.poly1d(model)
predict(65)
```

Output:

```
68.63
```

Step 6: Predict the y_pred for all values of x.

```
y_pred=      predict(x)
y_pred
```

Output:

```
array([81.50684932, 87.94520548, 71.84931507, 68.63013699, 71.84931507])
```

Step 7: Evaluate the performance of Model (R-Suare)

R squared calculation is not implemented in numpy… so that one should be borrowed from sklearn.

```
from sklearn.metrics import r2_scorer2_score(y,
y_pred)
```

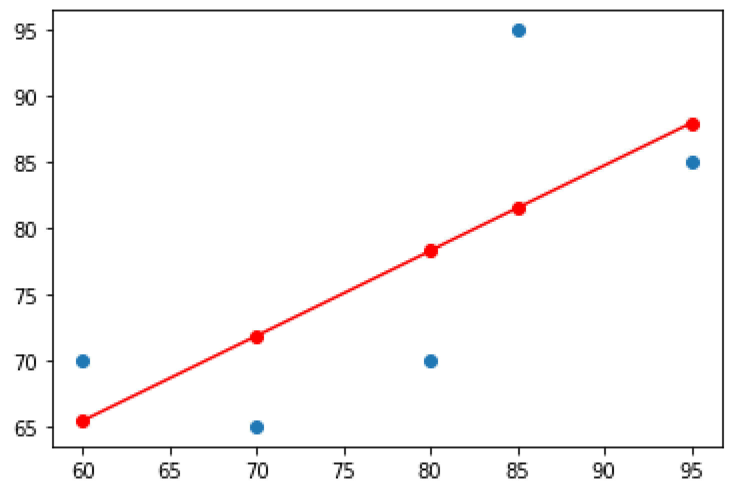
Output:

```
0.4803218090889323
```

Step 8: Plotting the linear regression model

```
y_line = model[1] + model[0]* xplt.plot(x,
y_line, c = 'r') plt.scatter(x, y_pred)
plt.scatter(x,y,c='r')
```

Output:



Algorithm (Boston Dataset):

Step 1: Import libraries and create alias for Pandas, Numpy and Matplotlib

```
import numpy as np
import pandas as pd

import matplotlib.pyplot as plt
```

Step 2: Import the Boston Housing dataset

```
from sklearn.datasets import load_bostonboston =
load_boston()
```

Step 3: Initialize the data frame

```
data = pd.DataFrame(boston.data)
```

Step 4: Add the feature names to the dataframe

```
data.columns = boston.feature_names data.head()
```

Step 5: Adding target variable to dataframe

```
data['PRICE'] = boston.target
```

Step 6: Perform Data Preprocessing(Check for missing values)

```
data.isnull().sum()
```

Step 7: Split dependent variable and independent variables

```
x = data.drop(['PRICE'], axis = 1)y =
data['PRICE']
```

Step 8: splitting data to training and testing dataset.

```
from sklearn.model_selection import train_test_splitxtrain, xtest, ytrain,
ytest =
train_test_split(x, y, test_size =0.2,random_state = 0)
```

Step 9: Use linear regression(Train the Machine) to Create Model

```
import sklearn

from sklearn.linear_model import LinearRegressionlm =
LinearRegression()

model=lm.fit(xtrain, ytrain)
```

Step 10: Predict the y_pred for all values of train_x and test_x

```
ytrain_pred = lm.predict(xtrain)ytest_pred =
lm.predict(xtest)
```

Step 11:Evaluate the performance of Model for train_y and test_y

```
df=pd.DataFrame(ytrain_pred,ytrain)
df=pd.DataFrame(ytest_pred,ytest)
```

Step 12: Calculate Mean Square Paper for train_y and test_y

```
from sklearn.metrics import mean_squared_error, r2_scoremse =
mean_squared_error(ytest, ytest_pred)

print(mse)

mse = mean_squared_error(ytrain_pred,ytrain)print(mse)
```

Output:

```
33.44897999767638
```

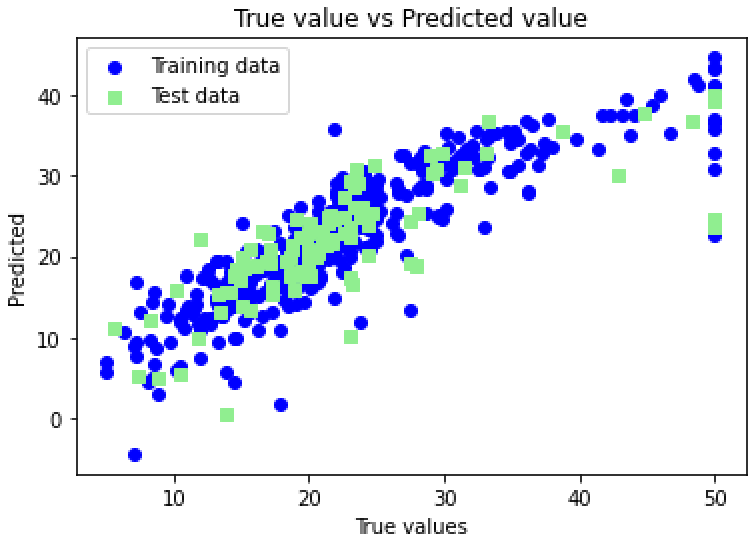
```
mse = mean_squared_error(ytest, ytest_pred)print(mse)
```

Output:

```
19.32647020358573
```

Step 13: Plotting the linear regression model

```
lt.scatter(ytrain ,ytrain_pred,c='blue',marker='o',label='Training data') plt.scatter(ytest,ytest_pred
,c='lightgreen',marker='s',label='Test data')plt.xlabel('True values')

plt.ylabel('Predicted')

plt.title("True value vs Predicted value")plt.legend(loc=
'upper left') #plt.hlines(y=0,xmin=0,xmax=50)

plt.plot()
plt.show()
```



Conclusion:

In this way we have done data analysis using linear regression for Boston Dataset and predict the price of houses using the features of the Boston Dataset.

Assignment Question:

1) **Compute SST, SSE, SSR, MSE, RMSE, R Square for the below example .**

Student	Score in X standard (Xi)	Score in XII standard (Yi)
1	95	85
2	85	95
3	80	70
4	70	65
5	60	70

2) **Comment on whether the model is best fit or not based on the calculated values.**
3) **Write python code to calculate the RSquare for Boston Dataset.**

   (Consider the linear regression model created in practical session)

.

Assignment No: 5

- -

Title of the Assignment:

1. Implement logistic regression using Python/R to perform classification on Social_Network_Ads.csv dataset.

2. Compute Confusion matrix to find TP, FP, TN, FN, Accuracy, Error rate, Precision, Recall on the given dataset..

Objective of the Assignment: Students should be able to data analysis using logistic regression using Python for any open source dataset

---

Prerequisite:

1. Basic of Python Programming

2.Concept of Regression.

Contents for Theory:

1. Logistic Regression

2. Differentiate between Linear and Logistic Regression

3. Sigmoid Function

4. Types of LogisticRegression

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. Another category of classification is Multinomial classification, which handles the issues where multiple classes are present in the target variable. For example, the IRIS dataset is a very famous example of multi-class classification. Other examples are classifying article/blog/document categories.

   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.

   Logistic Regression is one of the most simple and commonly used Machine Learning algorithms for two-class classification. It is easy to implement and can be used as the baseline for any binary classification problem. Its basic fundamental concepts are also constructive in deep learning. Logistic regression describes and estimates the relationship between one dependent binary variable and independent variables.

   Logistic regression is a statistical method for predicting binary classes. The outcome or target variable is dichotomous in nature. Dichotomous means there are only two possible classes. For example, it can be used for cancer detection problems. It computes the probability of an event occurring.

It is a special case of linear regression where the target variable is categorical in nature. It uses a log of odds as the dependent variable. Logistic Regression predicts the probability of occurrence of a binary event utilising a logit function.

Linear Regression Equation:

$$y = \beta 0 + \beta 1 X1 + \beta 2 X2 + \ldots\ldots + \beta n Xn$$

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

Sigmoid Function:

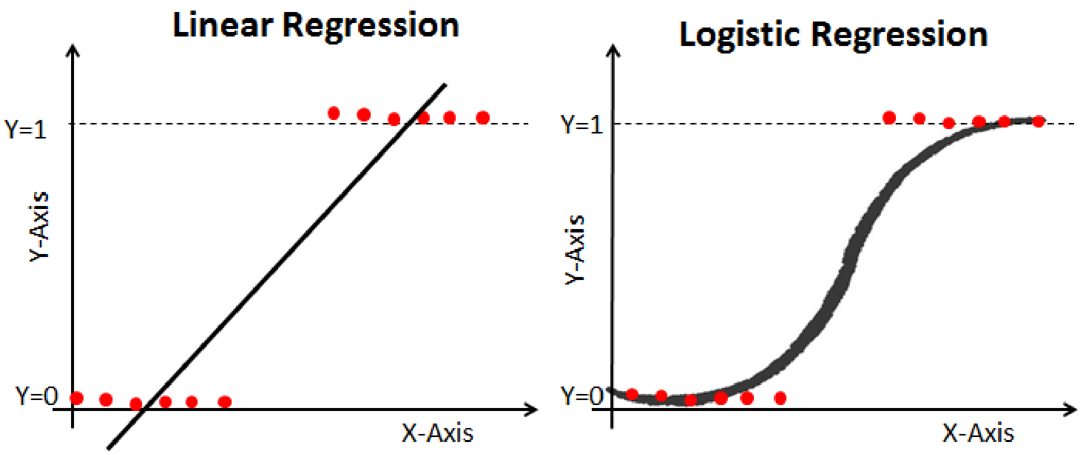
$$p = 1/1 + e^{-y}$$

Apply Sigmoid function on linear regression:

$$p = 1/1 + e^{-(\beta 0 + \beta 1 X1 + \beta 2 X2 \ldots\ldots \beta n Xn)}$$

2. Differentiate between Linear and Logistic Regression

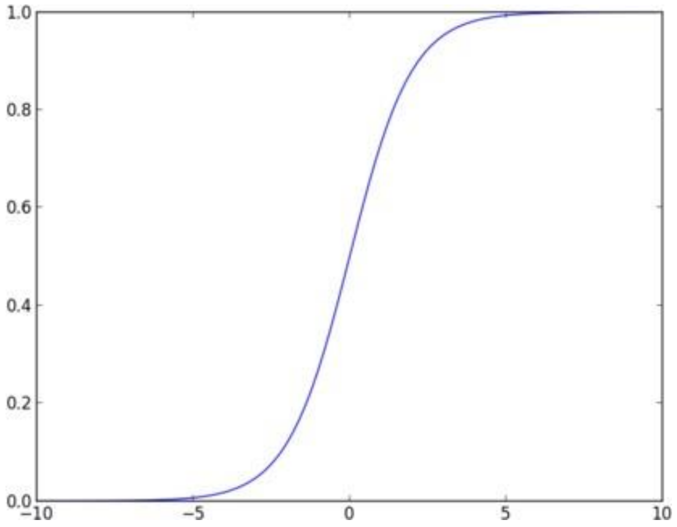
Linear regression gives you a continuous output, but logistic regression provides a constant output. An example of the continuous output is house price and stock price. Example's of the discrete output is predicting whether a patient has cancer or not, predicting whether the customer will churn. Linear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



3. Sigmoid Function

The sigmoid function, also called logistic function, gives an ‘S’ shaped curve that can take any real-valued number and map it into a value between 0 and 1. If the curve goes to positive infinity, y predicted will become 1, and if the curve goes to negative infinity, y predicted will become 0. If the output of the sigmoid function is more than 0.5, we can classify the outcome as 1 or YES, and if it is less than 0.5, we can classify it as 0 or NO. The outputcannotFor example: If the output is 0.75, we can say in terms of probability as: There is a 75 percent chance that a patient will suffer from cancer.

$$f\left(x\right)=\frac{1}{1+\mathrm{e}^{-(x)}}$$



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.

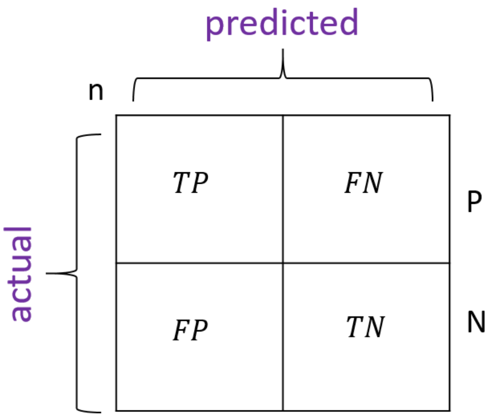
Multinomial Logistic Regression: The target variable has three or more nominal categories such as predicting the type of Wine.

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.



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.

$$acc = \frac{TP+TN}{TP+FP+TN+FN} = \frac{TP+TN}{Pos+Neg}$$

- **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.

$$err = \frac{FP+FN}{TP+FP+TN+FN} = \frac{FP+FN}{Pos+Neg}$$ Or

$$err = 1 - acc$$

- **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.

$$precision = \frac{TP}{TP+FP}$$

- **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.

$$recall = \frac{TP}{TP+FN}$$

Algorithm (Boston Dataset):

Step 1: Import libraries and create alias for Pandas, Numpy and Matplotlib

Step 2: Import the Social_Media_Adv Dataset

Step 3: Initialize the data frame Step 4:

Perform Data Preprocessing

- Convert Categorical to Numerical Values if applicable
- Check for Null Value
- Covariance Matrix to select the most promising features
- Divide the dataset into Independent(X) and Dependent(Y)variables.
- Split the dataset into training and testing datasets
- Scale the Features if necessary.

Step 5: Use Logistic regression(Train the Machine) to Create Model

```
 # import the class

   from sklearn.linear_model import LogisticRegression# instantiate the model
(using the default parameters)

   logreg = LogisticRegression()# fit the
model with data

   logreg.fit(xtrain,ytrain)

# y_pred=logreg.predict(xtest)
```

Step 6: Predict the y_pred for all values of train_x and test_x Step

7:Evaluate the performance of Model for train_y and test_yStep 8:

Calculate the required evaluation parameters

```
    from sklearn.metrics import precision_score,confusion_matrix,accuracy_score,recall_score
    cm= confusion_matrix(ytest, y_pred)
```

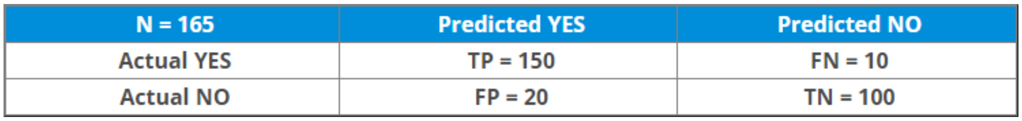
Conclusion:

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

Value Addition: Visualising Confusion Matrix using Heatmap

Assignment Question:

1) **Consider the binary classification task with two classes positive and negative. Find out TP,TP, FP, TN, FN, Accuracy, Error rate, Precision, Recall**



N = 165	Predicted YES	Predicted NO
Actual YES	TP = 150	FN = 10
Actual NO	FP = 20	TN = 100

2) **Comment on whether the model is best fit or not based on the calculated values.**

3) **Write python code for the preprocessing mentioned in step 4. and Explain every step in detail.**

.

Group A Assignment No: 6

- -

Title of the Assignment:

1. Implement Simple Naïve Bayes classification algorithm using Python/R on iris.csv dataset.

2. Compute Confusion matrix to find TP, FP, TN, FN, Accuracy, Error rate, Precision, Recall on the given dataset.

Objective of the Assignment: Students should be able to data analysis using Naïve Bayes classification algorithm using Python for any open source dataset

----

Prerequisite:

1. Basic of Python Programming

2.Concept of Join and Marginal Probability.

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,….j}
    - Let, input observation is specified by ‘P’ features. Therefore input observation x is given , x = {F1,F2,…..Fp}
    - 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:

$$\boldsymbol{P\left(\frac{A}{B}\right) = \frac{P(A\cap B)}{P(B)} \quad if\ P(B) \neq 0 \qquad \ldots\ldots.(1)}$$

$$\boldsymbol{P\left(\frac{B}{A}\right) = \frac{P(B\cap A)}{P(A)} \qquad \ldots\ldots\ldots(2)}$$

From equation (1) and (2) ,

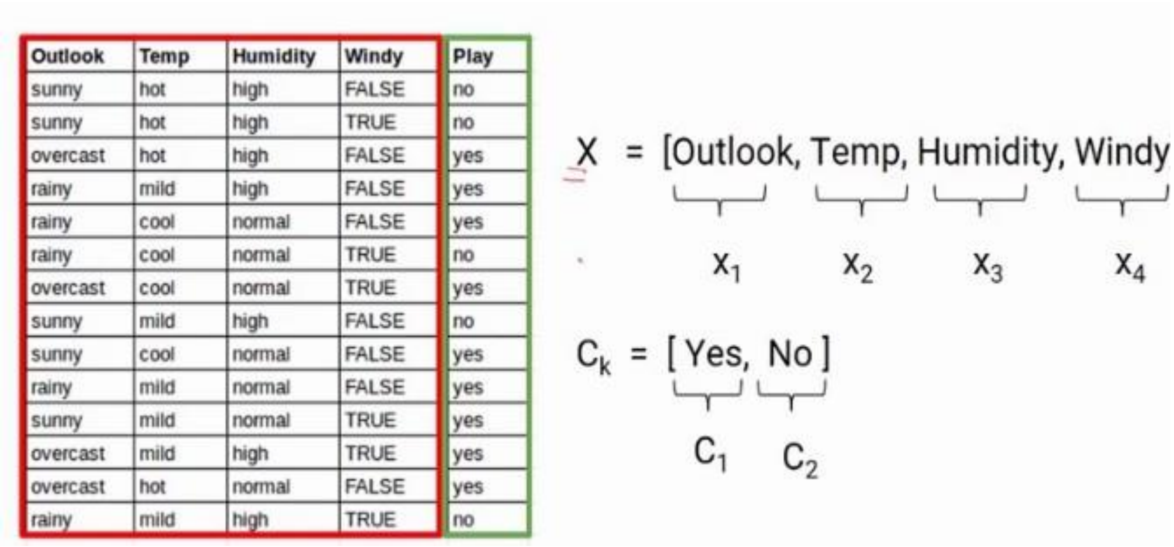
$$\boldsymbol{P(A\cap B) = P\left(\frac{A}{B}\right).P(B) = P\left(\frac{B}{A}\right).P(A)}$$

$$\therefore \qquad \boldsymbol{P\left(\frac{A}{B}\right) = \frac{P\left(\frac{B}{A}\right).P(A)}{P(B)}}$$

Is called the Bayes Rule.

2. Example of Naive Bayes

We have a dataset with some features Outlook, Temp, Humidity, and Windy, and the target here is to predict whether a person or team will play tennis or not.



Outlook	Temp	Humidity	Windy	Play
sunny	hot	high	FALSE	no
sunny	hot	high	TRUE	no
overcast	hot	high	FALSE	yes
rainy	mild	high	FALSE	yes
rainy	cool	normal	FALSE	yes
rainy	cool	normal	TRUE	no
overcast	cool	normal	TRUE	yes
sunny	mild	high	FALSE	no
sunny	cool	normal	FALSE	yes
rainy	mild	normal	FALSE	yes
sunny	mild	normal	TRUE	yes
overcast	mild	high	TRUE	yes
overcast	hot	normal	FALSE	yes
rainy	mild	high	TRUE	no

$$X = [\underbrace{\text{Outlook}}_{x_1}, \underbrace{\text{Temp}}_{x_2}, \underbrace{\text{Humidity}}_{x_3}, \underbrace{\text{Windy}}_{x_4}$$

$$C_k = [\underbrace{\text{Yes}}_{C_1}, \underbrace{\text{No}}_{C_2}]$$

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

$$P(C_k \mid X) = \frac{P(X \mid C_k)*P(C_k)}{P(X)}$$

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

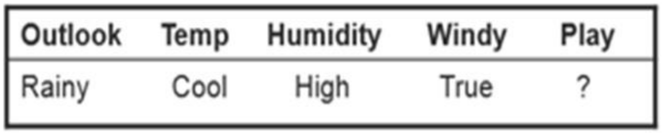
$$P(C_1 \mid x_1 \cap x_2 \cap x_3 \cap x_4) = \frac{P(x_1 \cap x_2 \cap x_3 \cap x_4 \mid C_1)* P(C_1)}{P(x_1 \cap x_2 \cap x_3 \cap x_4)}$$

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

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

This is the final equation of the Naive Bayes and we have to calculate the probability of both C1 and C2.For this particular example.



Outlook	Temp	Humidity	Windy	Play
Rainy	Cool	High	True	?

$$P(Yes \mid X) = P(Rainy \mid Yes) \times P(Cool \mid Yes) \times P(High \mid Yes) \times P(True \mid Yes) \times P(Yes)$$

$$P(Yes \mid X) = 2/9 \times 3/9 \times 3/9 \times 3/9 \times 9/14 = \boxed{0.00529} \rightarrow 0.2 = \frac{0.00529}{0.02057 + 0.00529}$$

$$P(No \mid X) = P(Rainy \mid No) \times P(Cool \mid No) \times P(High \mid No) \times P(True \mid No) \times P(No)$$

$$P(No \mid X) = 3/5 \times 1/5 \times 4/5 \times 3/5 \times 5/14 = \boxed{0.02057} \rightarrow 0.8 = \frac{0.02057}{0.02057 + 0.00529}$$

P (N0 | Today) > P (Yes | Today) So, the prediction that golf would be played is ‘No’.

Algorithm (Iris Dataset):

Step 1: Import libraries and create alias for Pandas, Numpy and Matplotlib

Step 2: Import the Iris dataset by calling URL.

Step 3: Initialize the data frame

Step 4: Perform Data

Preprocessing

- Convert Categorical to Numerical Values if applicable
- Check for Null Value
- Divide the dataset into Independent(X) and Dependent(Y)variables.
- Split the dataset into training and testing datasets
- Scale the Features if necessary.

Step 5: Use Naive Bayes algorithm(Train the Machine) to Create Model

```
# import the class

from sklearn.naive_bayes import GaussianNBgaussian =
GaussianNB() gaussian.fit(X_train, y_train)
```

Step 6: Predict the y_pred for all values of train_x and test_x

```
Y_pred = gaussian.predict(X_test)
```

Step 7:Evaluate the performance of Model for train_y and test_y

```
accuracy = accuracy_score(y_test,Y_pred)

precision =precision_score(y_test, Y_pred,average='micro')recall =
recall_score(y_test, Y_pred,average='micro')
```

Step 8: Calculate the required evaluation parameters

```
from sklearn.metrics import precision_score,confusion_matrix,accuracy_score,recall_score
cm = confusion_matrix(y_test, Y_pred)
```

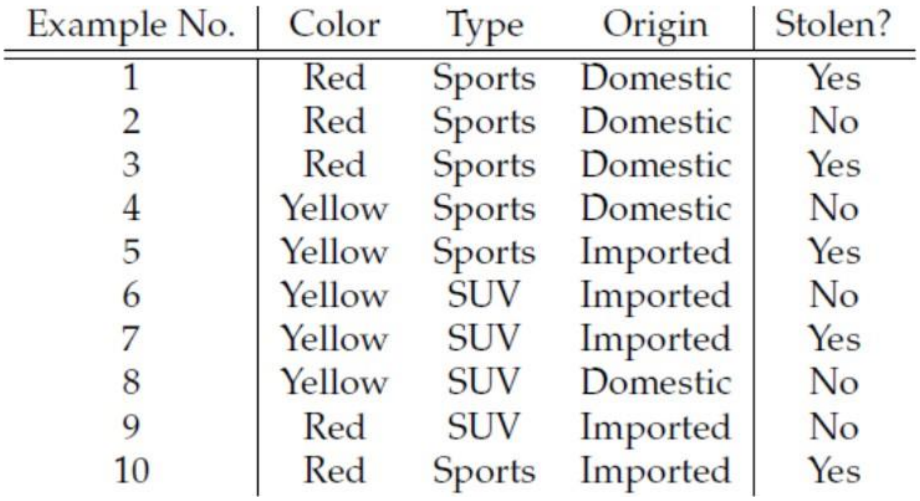
Conclusion:

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

Value Addition: Visualising Confusion Matrix using Heatmap

Assignment Question:

1) Consider the observation for the car theft scenario having 3 attributes colour, Type and origin.



Example No.	Color	Type	Origin	Stolen?
1	Red	Sports	Domestic	Yes
2	Red	Sports	Domestic	No
3	Red	Sports	Domestic	Yes
4	Yellow	Sports	Domestic	No
5	Yellow	Sports	Imported	Yes
6	Yellow	SUV	Imported	No
7	Yellow	SUV	Imported	Yes
8	Yellow	SUV	Domestic	No
9	Red	SUV	Imported	No
10	Red	Sports	Imported	Yes

Find the probability of car theft having scenarios Red SUV and Domestic.

2) Write python code for the preprocessing mentioned in step 4. and Explain every step in detail.

Group A Assignment No: 7

- -

Title of the Assignment:

1. Extract Sample document and apply following document preprocessing methods: Tokenization, POS Tagging, stop words removal, Stemming and Lemmatization.
2. Create representation of document by calculating Term Frequency and Inverse Document Frequency.

Objective of the Assignment: Students should be able to perform **Text Analysis** using TF IDF Algorithm

----

Prerequisite:

1. Basic of Python Programming

2. Basic of English language.

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 conversion 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.
E.g.
A stemmer reduces the words like fishing, fished, and fisher to the stem fish.
The stem need not be a word, for example the Porter algorithm reduces, argue, argued, argues, arguing, and argus to the stem argu .

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

Lemmatization Vs Stemming

Stemming algorithm works by cutting the suffix from the word. In a broader sense cuts either the beginning or end of the word.

On the contrary, Lemmatization is a more powerful operation, and it takes into consideration morphological analysis of the words. It returns the lemma which is the base form of all its inflectional forms. In-depth linguistic knowledge is required to create dictionaries and look for the proper form of the word. Stemming is a general operation while lemmatization is an intelligent operation where the proper form will be looked in the dictionary. Hence, lemmatization helps in forming better machine learning features.

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,RB,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.

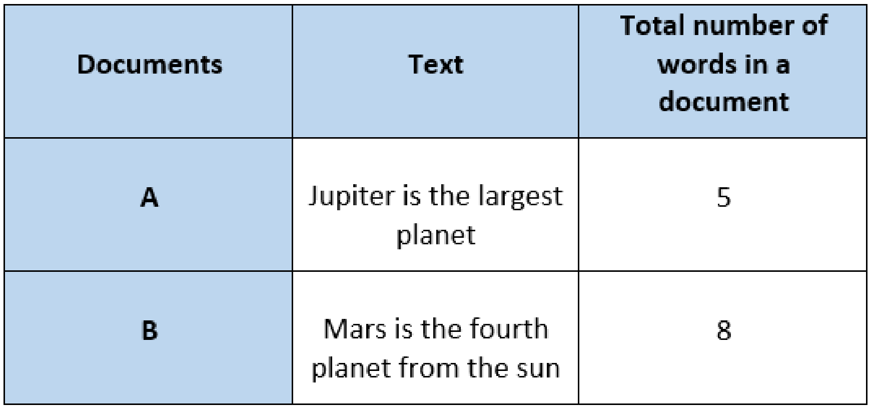
Term frequency–inverse document frequency(TFIDF) , is a numerical statistic that is intended to reflect how important a word is to a document in a collection or corpus.

- 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.

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

Example:



Documents	Text	Total number of words in a document
A	Jupiter is the largest planet	5
B	Mars is the fourth planet from the sun	8

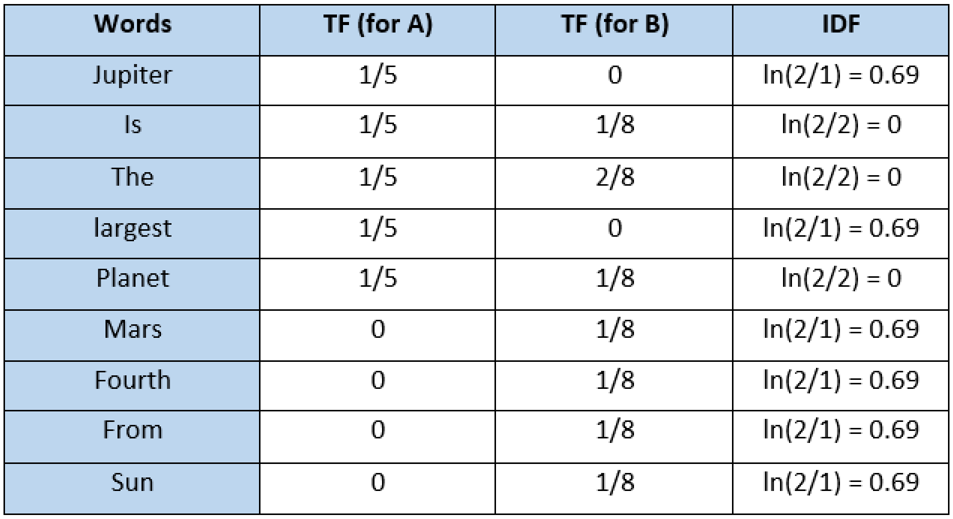
The initial step is to make a vocabulary of unique words and calculate TF for each document. TF will be more for words that frequently appear in a document and less for rare words in a document.

- 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.

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

In our example, since we have two documents in the corpus, N=2.



Words	TF (for A)	TF (for B)	IDF
Jupiter	1/5	0	ln(2/1) = 0.69
Is	1/5	1/8	ln(2/2) = 0
The	1/5	2/8	ln(2/2) = 0
largest	1/5	0	ln(2/1) = 0.69
Planet	1/5	1/8	ln(2/2) = 0
Mars	0	1/8	ln(2/1) = 0.69
Fourth	0	1/8	ln(2/1) = 0.69
From	0	1/8	ln(2/1) = 0.69
Sun	0	1/8	ln(2/1) = 0.69

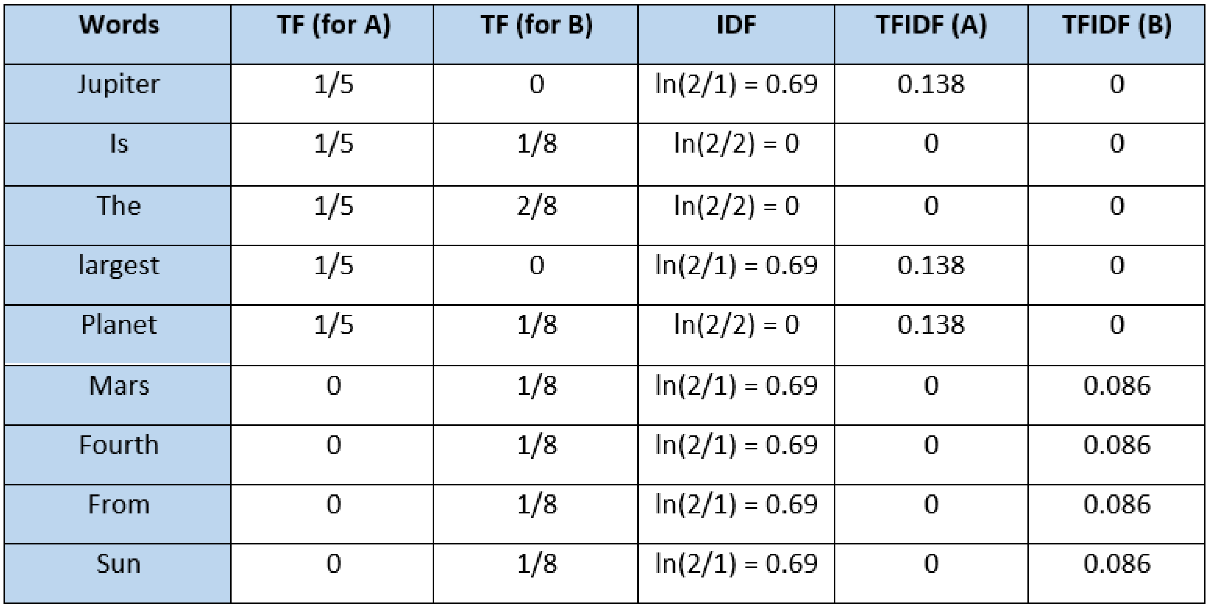
- Term Frequency — Inverse Document Frequency (TFIDF)

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.

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



Words	TF (for A)	TF (for B)	IDF	TFIDF (A)	TFIDF (B)
Jupiter	1/5	0	ln(2/1) = 0.69	0.138	0
Is	1/5	1/8	ln(2/2) = 0	0	0
The	1/5	2/8	ln(2/2) = 0	0	0
largest	1/5	0	ln(2/1) = 0.69	0.138	0
Planet	1/5	1/8	ln(2/2) = 0	0.138	0
Mars	0	1/8	ln(2/1) = 0.69	0	0.086
Fourth	0	1/8	ln(2/1) = 0.69	0	0.086
From	0	1/8	ln(2/1) = 0.69	0	0.086
Sun	0	1/8	ln(2/1) = 0.69	0	0.086

After applying TFIDF, text in A and B documents can be represented as a TFIDF vector of dimension equal to the vocabulary words. The value corresponding to each word represents the importance of that word in a particular document.

TFIDF is the product of TF with IDF. Since TF values lie between 0 and 1, not using ***ln*** can result in high IDF for some words, thereby dominating the TFIDF. We don't want that, and therefore, we use ***ln*** so that the IDF should not completely dominate the TFIDF.

- 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 toas a “bag” of words because any information about the structure of the sentence is lost.

Algorithm for Tokenization, POS Tagging, stop words removal, Stemming andLemmatization:

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. The process of breaking down a
text paragraph into smaller chunkssuch as words or sentences is called Tokenization."
```

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
                                    WordNetLemmatiz
erwordnet_lemmatizer = WordNetLemmatizer() text =
"studies studying cries cry" tokenization =
nltk.word_tokenize(text)

for w in tokenization:

        print("Lemma                for          {}          is
                                    {}".format(w,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 TFIDFStep 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 thecorpus

```
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

Assignment Question:

1) **Perform Stemming for** text = "studies studying cries cry". **Compare the results generated with Lemmatization. Comment on your answer how Stemming and Lemmatization differ from each other.**

2) Write Python code for removing stop words from the below documents, conver the documents into lowercase and calculate the TF, IDF and TFIDF score for each document.

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

Assignment No: 8

- -

Title of the Assignment: Data Visualization I

1. Use the inbuilt dataset 'titanic'. The dataset contains 891 rows and contains information about the passengers who boarded the unfortunate Titanic ship. Use the Seaborn library to seeif we can find any patterns in the data.

2. Write a code to check how the price of the ticket (column name: 'fare') for each passengeris distributed by plotting a histogram.

Objective of the Assignment: Students should be able to perform the data Visualizationoperation using Python on any open source dataset

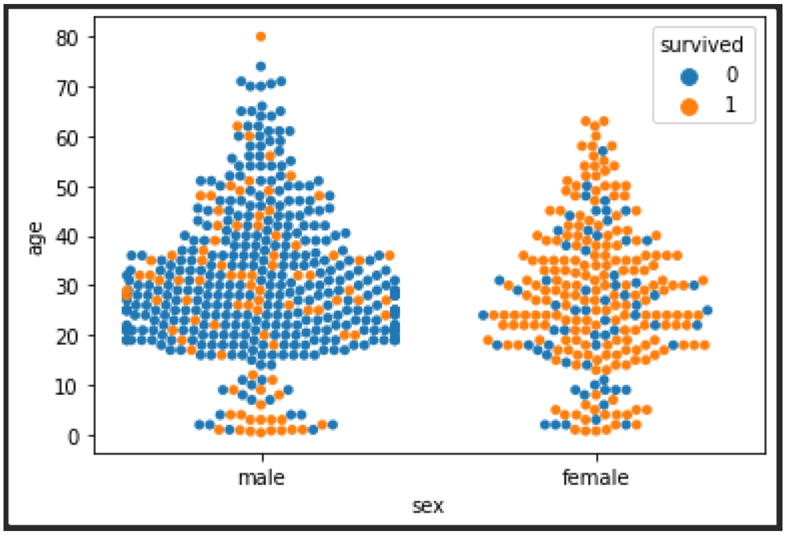
---

Prerequisite:

1. Basic of Python Programming
2. Seaborn Library, Concept of Data Visualization.

Assignment Questions

1. List out different types of plot to find patterns of data
2. **Explain when you will use distribution plots and when you will use categorical plots.**
3. Write the conclusion from the following swarm plot (consider titanic dataset)



4. **Which parameter is used to add another categorical variable to the violin plot,Explain with syntax and example.**

Assignment No: 9

Title of the Assignment: Data Visualization II

1. Use the inbuilt dataset 'titanic' as used in the above problem. Plot a box plot for distribution of age with respect to each gender along with the information about whether theysurvived or not. (Column names : 'sex' and 'age')

2. Write observations on the inference from the above statistics.

Objective of the Assignment: Students should be able to perform the data Visualizationoperation using Python on any open source dataset

- -

Prerequisite:

1. Basic of Python Programming
2. Seaborn Library, Concept of Data Visualization.

Assignment Questions

1. Write down the code to use inbuilt dataset ‘titanic’ using seaborn library.
2. **Write code to plot a box plot for distribution of age with respect to each genderalong with the information about whether they survived or not.**
3. Write the observations from the box plot.

Assignment No: 10

- -

Title of the Assignment: Data Visualization III

Download the Iris flower dataset or any other dataset into a DataFrame. (e.g., https://archive.ics.uci.edu/ml/datasets/Iris). Scan the dataset and give the inference as:

1. List down the features and their types (e.g., numeric, nominal) available in the dataset.
2. Create a histogram for each feature in the dataset to illustrate the feature distributions.
3. Create a box plot for each feature in the dataset.
4. Compare distributions and identify outliers.

Objective of the Assignment: Students should be able to perform the data Visualizationoperation using Python on any open source dataset

- -

Prerequisite:

1. Basic of Python Programming

2. Seaborn Library, Concept of Data Visualization.
3. Types of variables

Assignment Questions

1. For the iris dataset, list down the features and their types.
2. Write a code to create a histogram for each feature. (iris dataset)
3. Write a code to create a boxplot for each feature. (iris dataset)
4. Identify the outliers from the boxplot drawn for iris dataset.