Amrutvahini College of Engineering, Sangamner
Department of Computer Engineering

Data Science and Big Data Analytics

INDEX

Class: T.E Div : Batch:

Sr. No.	Title of Experiments	Page no.	Date of		Remark	Signature
			Performance	Submission		
Data Science and Big Data Analytics Laboratory						
Group A						
1	**Data Wrangling, I-** Perform the following operations using Python on any open-source dataset (e.g., data.csv) 1. Import all the required Python Libraries. 2. Locate an open-source data from the web (e.g., https://www.kaggle.com). Provide a clear description of the data and its source (i.e., URL of the web site). 3. Load the Dataset into pandas data frame. 4. Data Preprocessing: check for missing values in the data using pandas isnull (), 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. In addition to the codes and outputs, explain every operation that you do in the above steps and explain everything that you do to import/read/scrape the data set.					
2	**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.					
3	**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) variable. 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.					
4	**Data Analytics I -** 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.					
5	**Data Analytics II -**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.					
6	**Data Analytics III**- 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.					
7	**Text Analytics** -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.					
8	**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 see if 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 passenger is distributed by plotting a histogram.					
9	**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 they survived or not. (Column names: 'sex' and 'age') 2. Write observations on the inference from the above statistics.					

10	**Data Visualization III -**Download the Iris flower dataset or any other dataset into a Data Frame. (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 boxplot for each feature in the dataset. 4. Compare distributions and identify outliers.					
Group B						
11	Write a code in JAVA for a simple WordCount application that counts the number of occurrences of each word in a given input set using the Hadoop MapReduce framework on local-standalone set-up.					
12	Design a distributed application using MapReduce which processes a log file of a system.					
13	Locate dataset (e.g., sample_weather.txt) for working on weather data which reads the text input files and finds average for temperature, dew point and wind speed. OR Write a simple program in SCALA using Apache Spark framework.					
Group C						
14	Write a case study on Global Innovation Network and Analysis (GINA). Components of analytic plan are 1. Discovery business problem framed, 2. Data, 3. Model planning analytic technique and 4. Results and Key findings. OR Use the following dataset and classify tweets into positive and negative tweets. https://www.kaggle.com/ruchi798/data-science-tweets.					

	OR Develop a movie recommendation model using the scikit-learn library in python. Refer dataset https://github.com/rashida048/Some-NLP-Projects/blob/master/movie_dataset.csv.					
15	Use the following covid_vaccine_statewise.csv dataset and perform following analytics on the given dataset https://www.kaggle.com/sudalairajkumar/covid19-in-india?select=covid_vaccine_statewise.csv a. Describe the dataset b. Number of persons state wise vaccinated for first dose in India c. Number of persons state wise vaccinated for second dose in India d. Number of Males vaccinated d. Number of females vaccinated. OR Write a case study to process data driven for Digital Marketing OR Health care systems with Hadoop Ecosystem components					

CERTIFICATE

This is to certify that Mr./Ms. _____ Roll No_____

Examination Seat no- _____ has performed above mentioned practicals in the college.

Prof. Paikrao R.L.

Faculty member Incharge Head of the Department

Date: / /202

Group A			
Assignment No 1	**Data Wrangling, I**	**Page No**	

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 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:

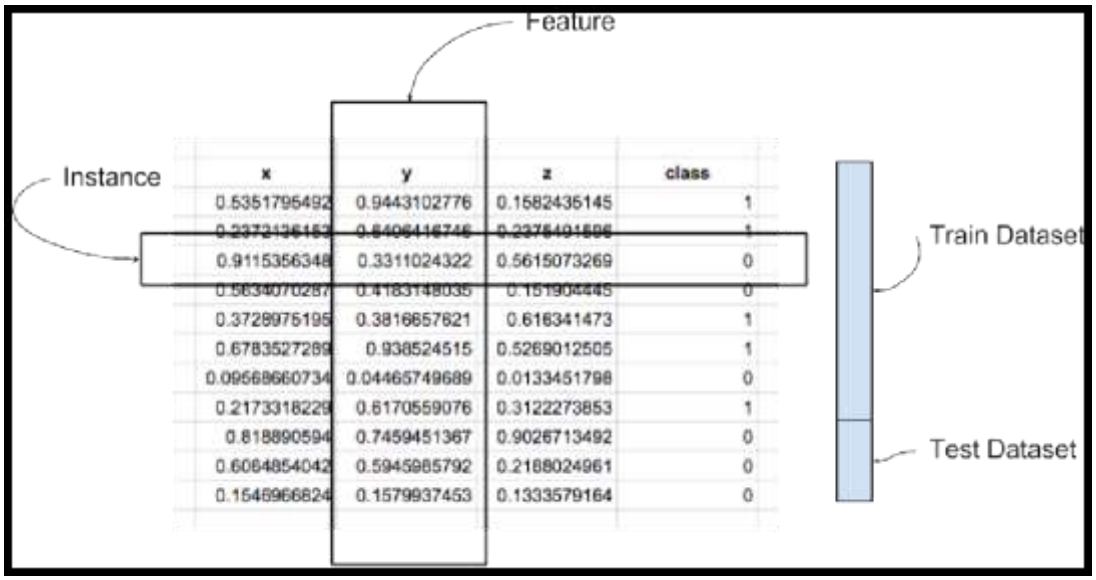
1. **Introduction to Dataset**
2. **Python Libraries for Data Science**
3. **Description of Dataset**
4. **Panda Dataframe functions for load the dataset**
5. **Panda functions for Data Preprocessing**

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	0.1582436145	1
0.2372136153	0.8406116746	0.2376491896	1
0.9115356348	0.3311024322	0.5615073269	0
0.5634070287	0.4183148035	0.151904445	0
0.3728975195	0.3816857621	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
0.6084854042	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 ofelements 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, randomforests, 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.

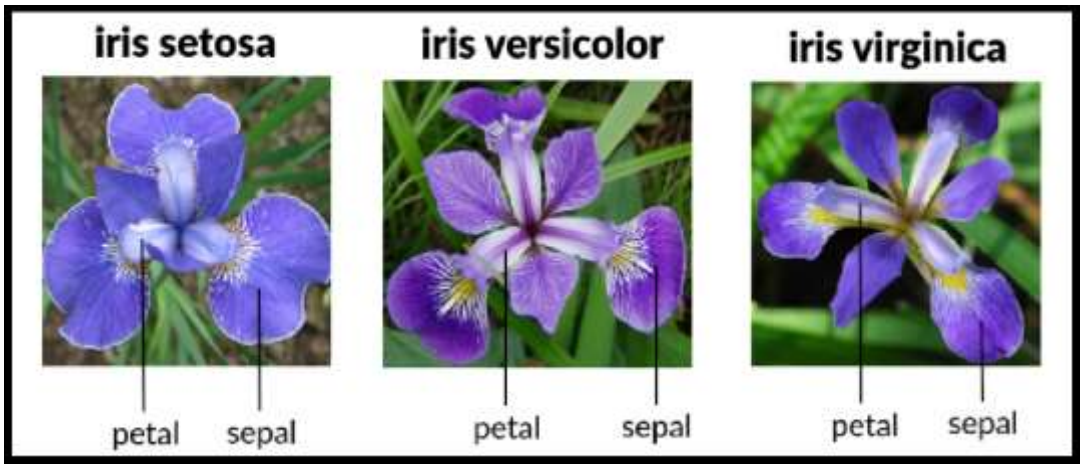
Total Sample- 150

The columns in this dataset are:

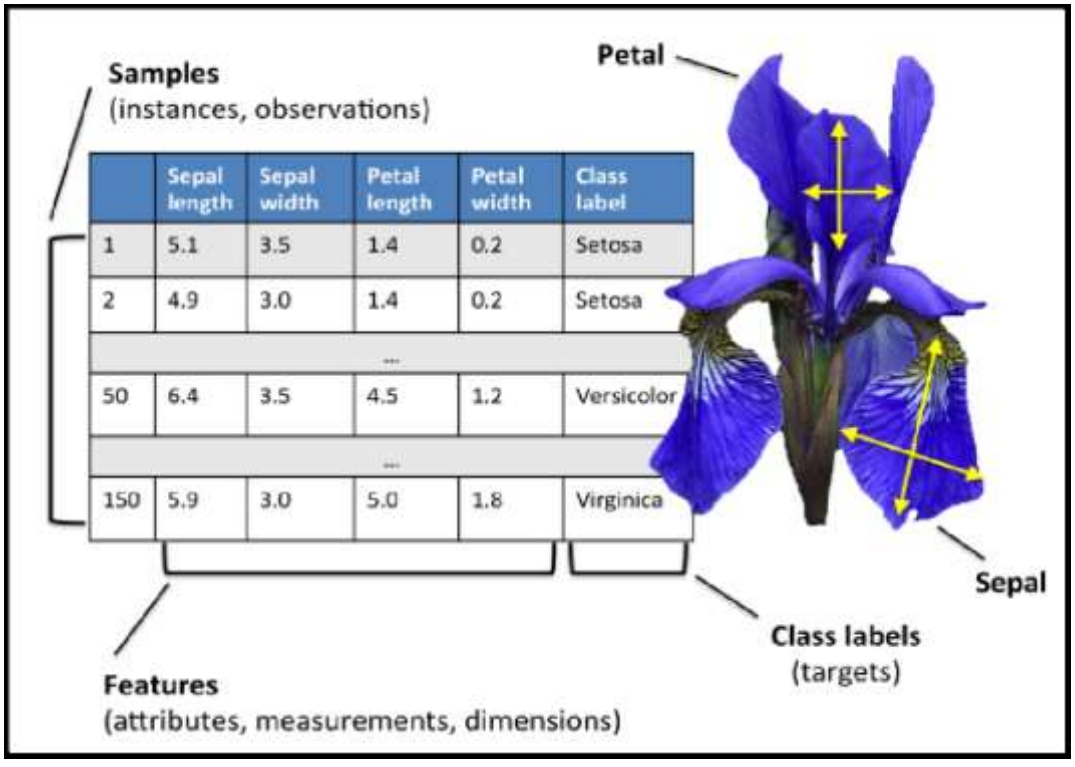
1. Id
2. SepalLengthCm
3. SepalWidthCm
4. PetalLengthCm

5. PetalWidthCm
6. Species

3 Different Types of Species each contain 50 Sample-



Description of Dataset-



	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 :**

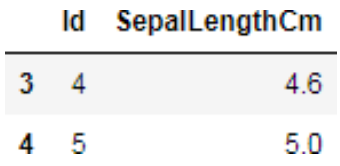
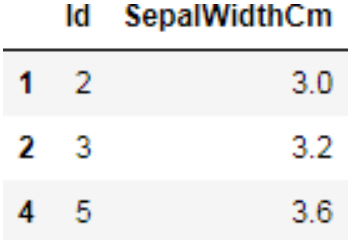
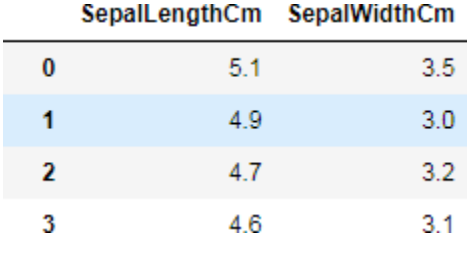
Dataframe Operations:

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 arrayformat
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 byposition.
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

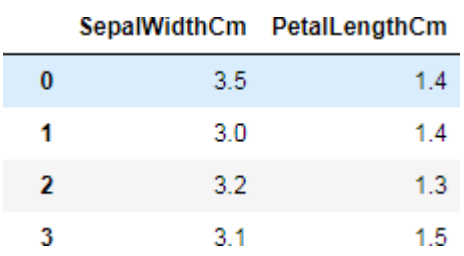
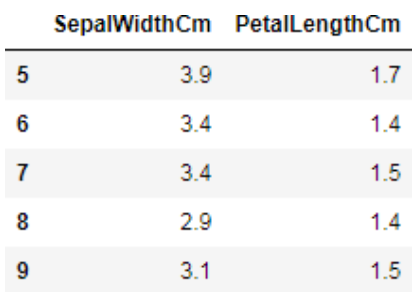
Sr. No	Data Frame Function	Description	Output
1	**dataset.iloc[3:5, 0:2]**	Slice the data	
2	**dataset.iloc[[1, 2, 4], [0, 2]]**	By lists of integer position locations, similar to the NumPy/Python style:	
3	**dataset.iloc[1:3, :]**	For slicing rows explicitly:	
4	**dataset.iloc[:, 1:3]**	For slicing Column explicitly:	

	Id	SepalLengthCm
3	4	4.6
4	5	5.0

	Id	SepalWidthCm
1	2	3.0
2	3	3.2
4	5	3.6

	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

	SepalLengthCm	SepalWidthCm
0	5.1	3.5
1	4.9	3.0
2	4.7	3.2
3	4.6	3.1

5	**dataset.iloc[1, 1]**	For getting a value explicitly:	4.9
6	**dataset['SepalLengthCm'].iloc[5]**	Accessing Column and Rows by position	5.4
7	**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
8	**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

```
iris = load_iris()

df = pd.DataFrame(iris.data,
columns=iris.feature_names)
```

Step 3: Print iris dataset.

```
df.head()
```

Step 4: Create x, where x the 'scores' column's values as floats

```
x = df[['score']].values.astype(float)
```

Step 5: Create a minimum and maximum processor object

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

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 0 and n_classes-1.
- **`fit_transform(y):`**

  Parameters: `yarray-like of shape (n_samples,)`
  `Target values.`

  Returns: `yarray-like of shape (n_samples,)`
  `Encoded labels.`

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

Algorithm:

Step 1 : Import pandas and sklearn library for preprocessing

```
from sklearn import preprocessing
```

Step 2: Load the iris dataset in dataframe object df

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

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

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

Step 4: define label_encoder object knows how to understand word labels.

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

Step 5: Encode labels in column 'species'.

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

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

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

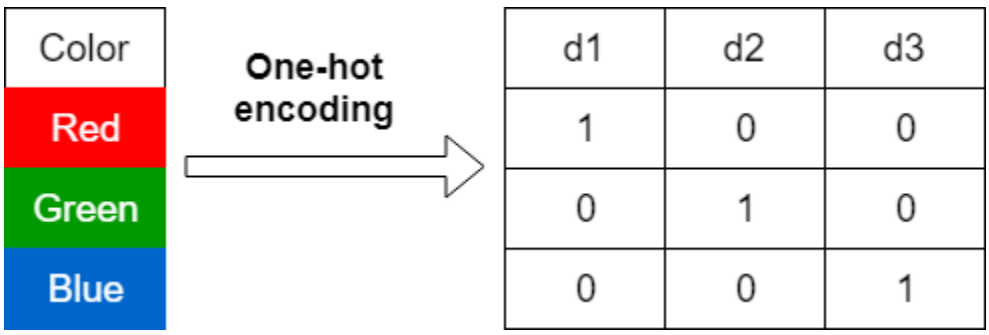
```
Output: array([0, 1, 2], dtype=int64)
```

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

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

b. One-Hot Encoding:

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



In one-hot encoding,

"Red" color is encoded as **[1 0 0]** vector of size 3.

"Green" color is encoded as **[0 1 0]** vector of size 3.

"Blue" color is encoded as **[0 0 1]** vector of size 3.

One-hot encoding on iris dataset: For iris dataset the target column which is Species. It contains three species Iris-setosa, Iris-versicolor, Iris-virginica.

Sklearn Functions for One-hot Encoding:

- **`sklearn.preprocessing.OneHotEncoder():`** Encode categorical integer features using a one-hot aka one-of-K scheme

Algorithm:

Step 1 : Import pandas and sklearn library for preprocessing

```
from sklearn import preprocessing
```

Step 2: Load the iris dataset in dataframe object df

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

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

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

Step 4: Apply label_encoder object for label encoding the Observe the unique values for the Species column.

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

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

Step 5: Remove the target variable from dataset

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

Step 6: Apply one_hot encoder for Species column.

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

Step 7: Join the encoded values with Features variable

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

Step 8: Observe the merge dataframe

```
df_encode
```

Step 9: Rename the newly encoded columns.

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

Step 10: Observe the merge dataframe

```
df_encode
```

Output after Step 8:

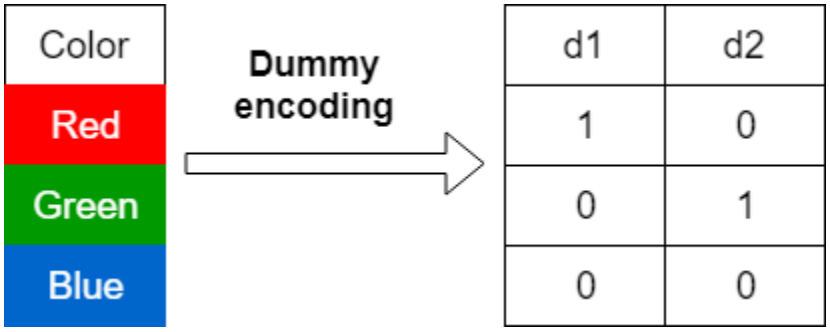
	Sepal_Length	Sepal_Width	Petal_Length	Petal_Width	0	1	2
0	5.1	3.5	1.4	0.2	1.0	0.0	0.0
1	4.9	3.0	1.4	0.2	1.0	0.0	0.0
2	4.7	3.2	1.3	0.2	1.0	0.0	0.0
3	4.6	3.1	1.5	0.2	1.0	0.0	0.0
4	5.0	3.6	1.4	0.2	1.0	0.0	0.0

Output after Step 10:

	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

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=False)
```

Step 7: Observe the merge dataframe

```
one_hot_df
```

	Sepal_Length	Sepal_Width	Petal_Length	Petal_Width	Species_0	Species_1	Species_2
0	5.1	3.5	1.4	0.2	1	0	0
1	4.9	3.0	1.4	0.2	1	0	0
2	4.7	3.2	1.3	0.2	1	0	0
3	4.6	3.1	1.5	0.2	1	0	0
4	5.0	3.6	1.4	0.2	1	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?
4. What are the different Techniques for Handling the Missing Data?

Group A			
Assignment No 2	**Data Wrangling,** II	**Page No**	

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

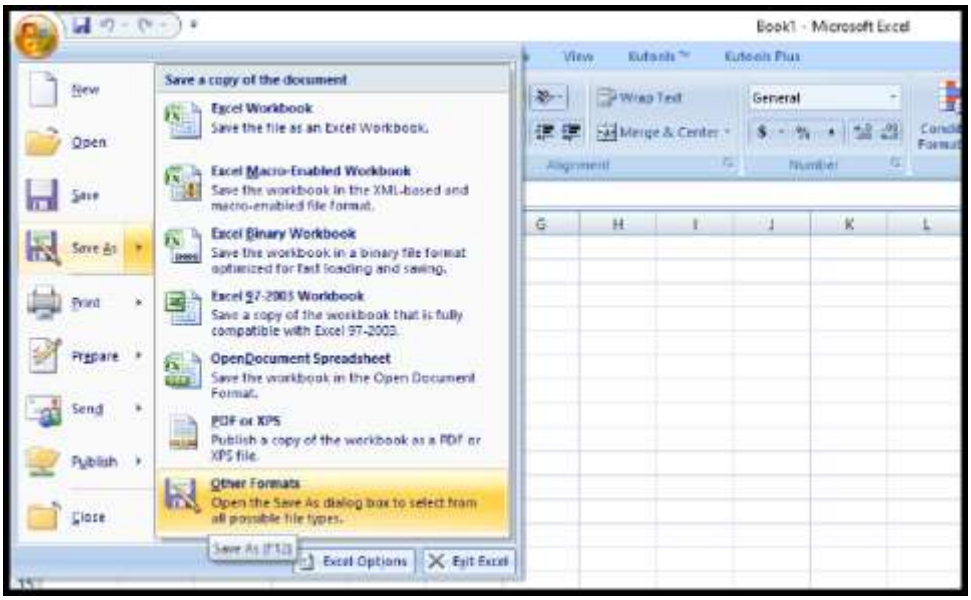
Syntax : RANDBETWEEN(bottom, top) Bottom The smallest integer and

Top The largest integer RANDBETWEEN will return.

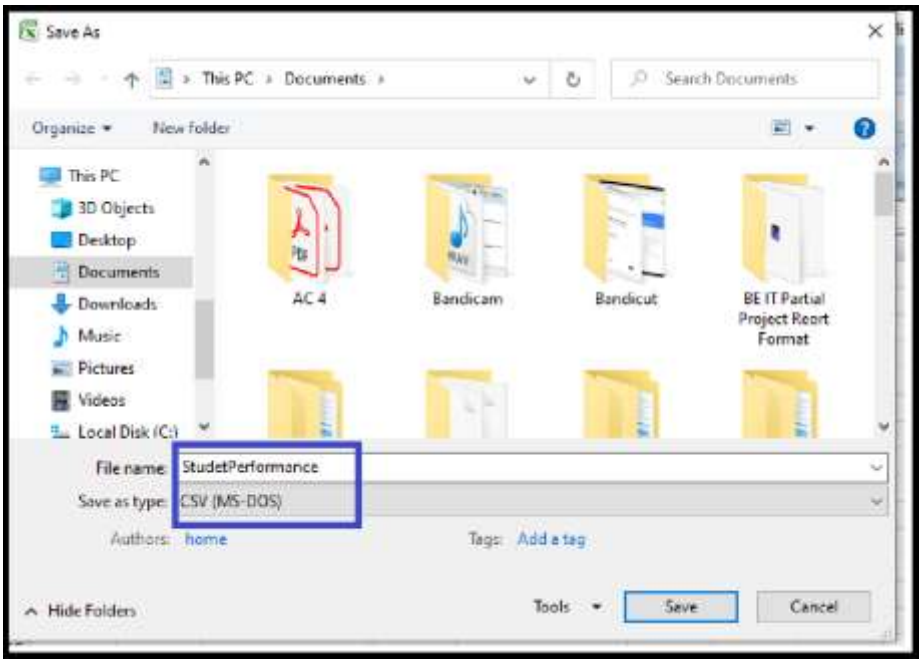
For better understanding and visualization, 20% impurities are added into each variableto the dataset.

The step to create the dataset are as follows:

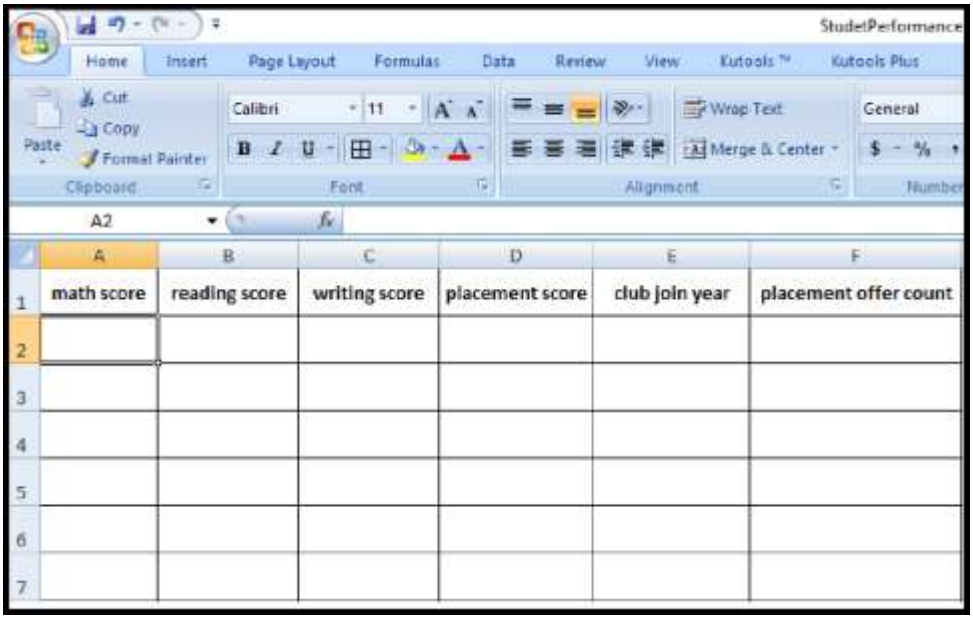
Step 1: Open Microsoft Excel and click on Save As. Select Other .Formats



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



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

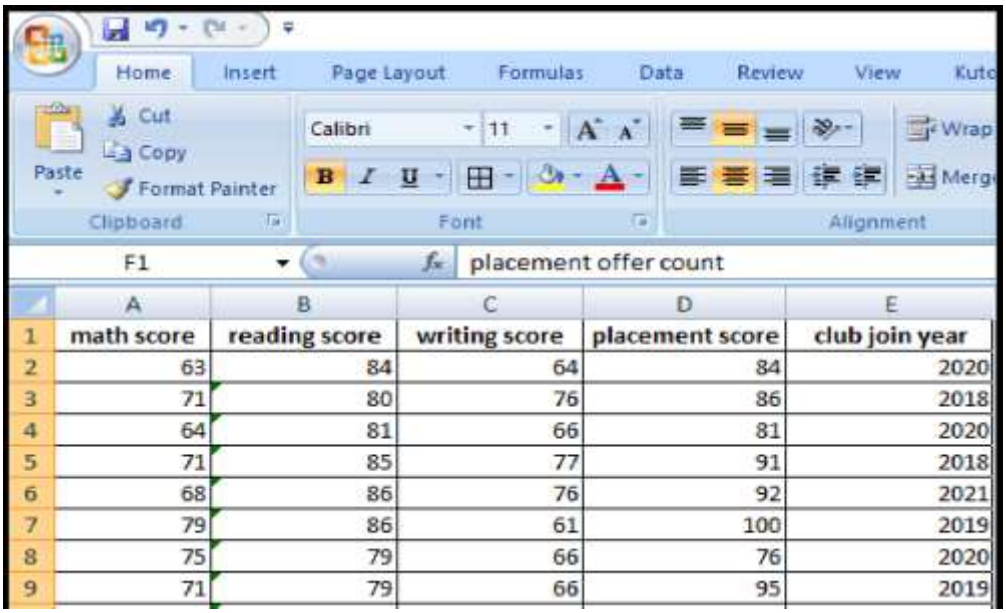


Step 3: Fill the dara by using **RANDOMBETWEEN** function. For every feature , fill the data by considering above spectified range.

one example is given:

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

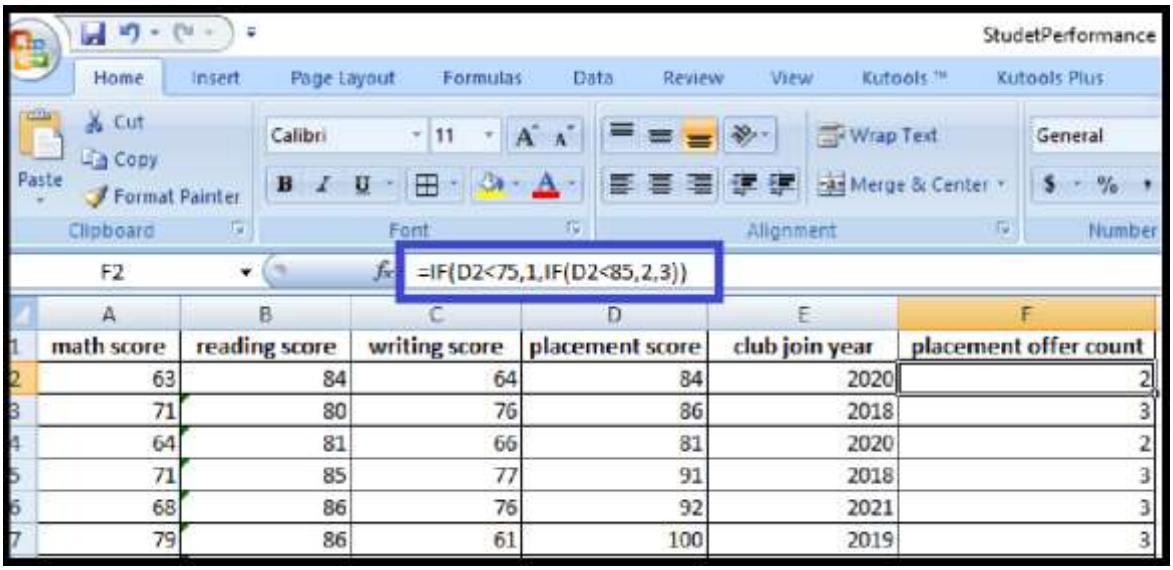
Repeat this for the features, Reading_Score, Writing_Score, Placement_Score, Club_Join_Date.



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

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 fordetecting, removing, and replacing null values in Pandas DataFrame :

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

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

   - **Checking for missing values using isnull()**

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

Algorithm:

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

```
import pandas as pd

import numpy as np
```

Step 2: Load the dataset in dataframe object df

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

Step 3: Display the data frame

```
df
```

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

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

```
df.isnull()
```

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

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

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

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

- **Checking for missing values using notnull()**

In order to check null values in Pandas Dataframe, notnull() function is used. This

function return dataframe of Boolean values which are False for NaN values.

Algorithm:

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

```
import pandas as pd

import numpy as np
```

Step 2: Load the dataset in dataframe object df

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

Step 3: Display the data frame

```
df
```

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

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

```
df.notnull()
```

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

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

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

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

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

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

- **For replacing null values with NaN**

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

	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

- **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
```

	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	76.0	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 4: filling missing value using fillna()

```
ndf=df

ndf.fillna(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 pd
import numpy as np
```

Step 2: Load the dataset in dataframe object df

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

Step 3: Display the data frame

```
df
```

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

```
ndf.dropna()
```

	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

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

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

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

	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

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

making new data frame with dropped NA values

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

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

3. Identification and Handling of Outliers

3.1 Identification of Outliers

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

1. What are Outliers?

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

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

2. Why do they occur?

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

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

3. What do they affect?

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

Mean is the accurate measure to describe the data when we do not have any outliers present. Median is used if there is an outlier in the dataset. Mode is used if thereis 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.

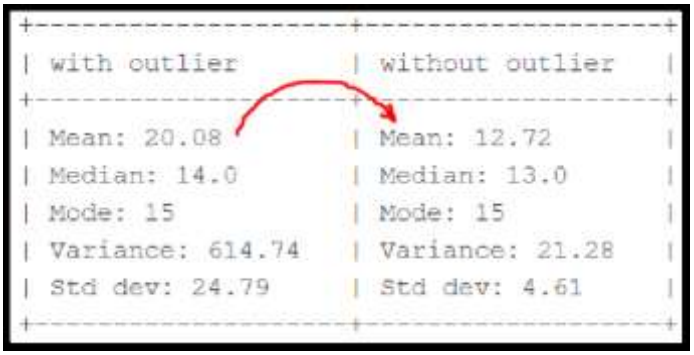


fig. Computation with and without outlier

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")
```

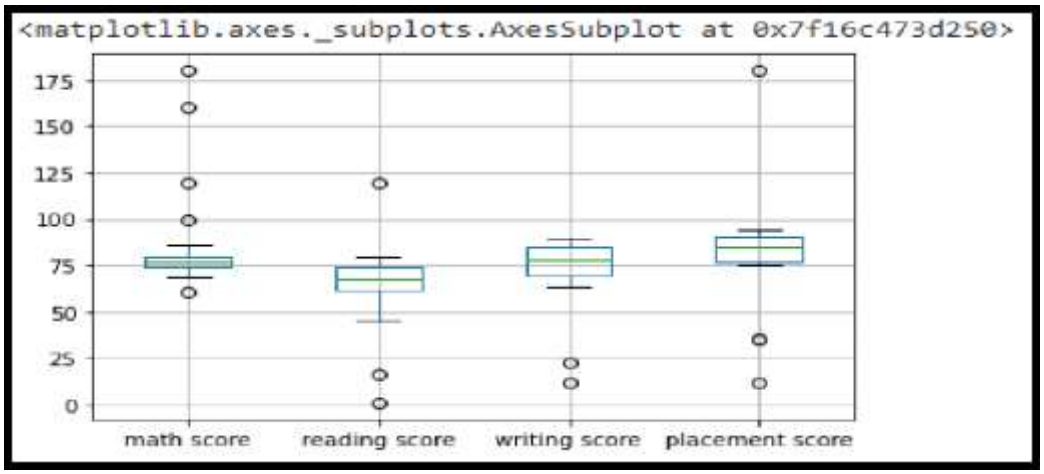
Step 3: Display the data frame

```
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' , 'writing score','placement score']
df.boxplot(col)
```



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

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

4.2 Detecting outliers using Scatterplot:

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

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

Algorithm:

Step 1 : Import pandas , numpy and matplotlib libraries

```
import pandas as pd
```

```
import numpy as np
import matplotlib.pyplot as plt
```

Step 2: Load the dataset in dataframe object df

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

Step 3: Display the data frame

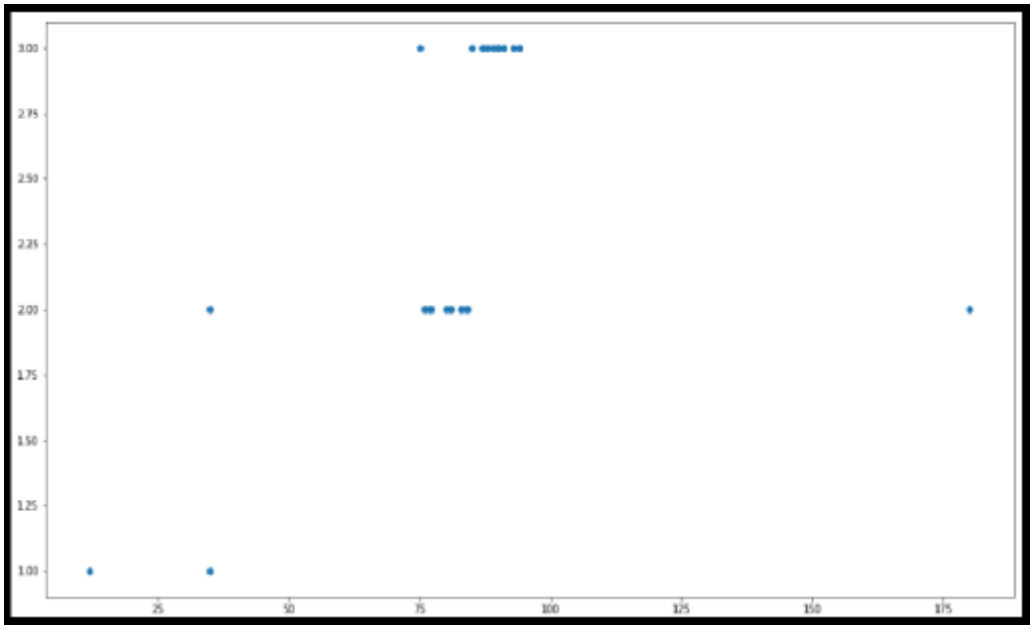
```
df
```

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

```
fig, ax = plt.subplots(figsize = (18,10))
ax.scatter(df['placement score'], df['placement offer
    count'])
plt.show()
```

Labels to the axis can be assigned (Optional)

```
ax.set_xlabel('(Proportion non-retail business
acres)/(town)')
ax.set_ylabel('(Full-value property-tax rate)/(
$10,000)')
```



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

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

4.3 Detecting outliers using Z-Score:

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

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 datasetsnormal 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)
print(r_outliers)
```

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

3.2 Handling of Outliers:

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

Below are some of the methods of treating the outliers

- Trimming/removing the outlier

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

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

```
new_df=df

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

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

Here Sample_outliers 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'])
print("New array:",b)
```

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

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

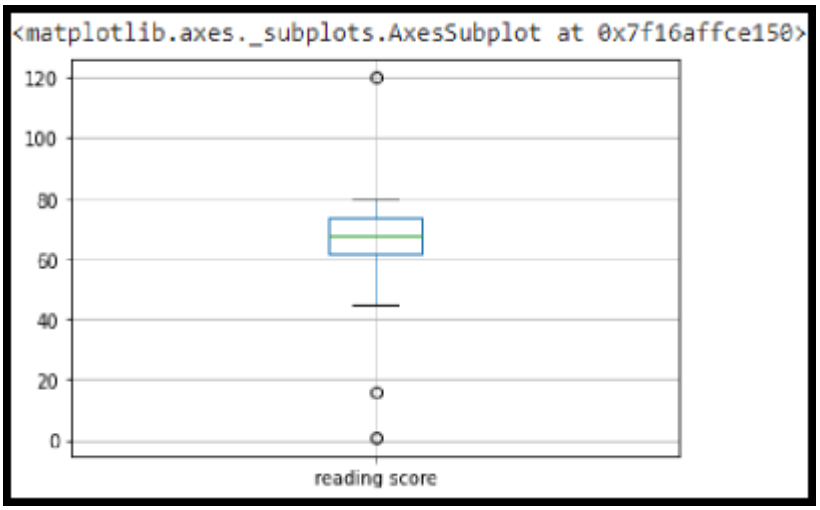
	math score	m score	reading score	writing score	placement score	placement offer count
0	80	80.0	68	70	89	3
1	71	71.0	61	85	91	3
2	79	79.0	16	87	77	2
3	61	61.0	77	74	76	2
4	78	78.0	71	67	90	3
5	73	73.0	68	90	80	2
6	77	77.0	62	70	35	2
7	74	74.0	45	80	12	1

- **Mean/Median imputation:**

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

1. Plot the box plot for reading score

```
col = ['reading score']
df.boxplot(col)
```



2. Outliers are seen in box plot.

3. Calculate the median of reading score by using sorted_rscore

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

4. Replace the upper bound outliers using median value

```
refined_df=df

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

5. Display redefined_df

	math score	n score	reading score	writing score	placement score	placement offer count
0	80	80.0	66.0	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	66.0	90	80	2
6	77	77.0	62.0	70	35	2
7	74	74.0	45.0	80	12	1
8	76	76.0	60.0	79	77	2
9	75	75.0	65.0	85	87	3
10	160	104.0	67.0	12	83	2

6. Replace the lower bound outliers using median value

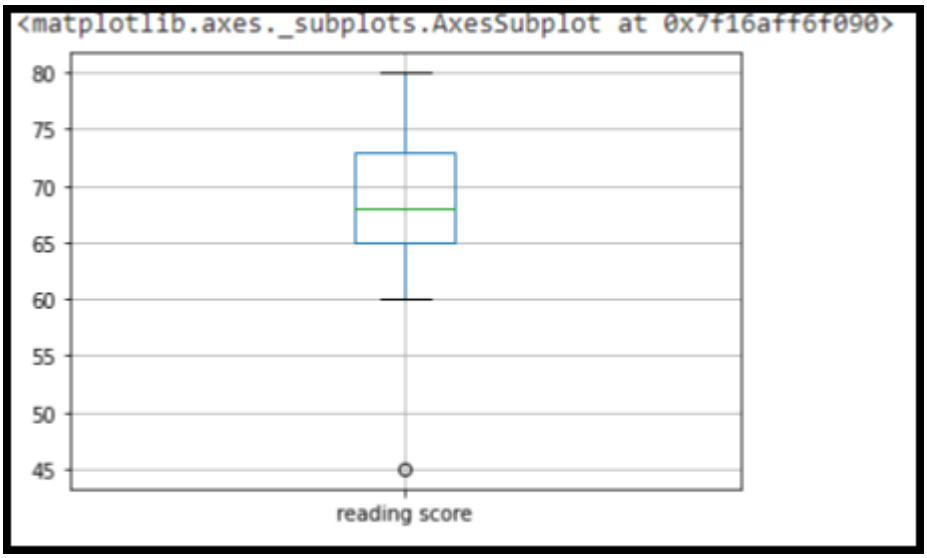
```
refined_df['reading score'] = np.where(refined_df['reading score'] <lwr_bound, median,refined_df['reading score'])
```

7. Display redefined_df

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

8. Draw the box plot for redefined_df

```
col = ['reading score']
refined_df.boxplot(col)
```



4. Data Transformation for the purpose of :

Data transformation is the process of converting raw data into a format or structure that would be more suitable for model building and also data discovery in general.The process

of data transformation can also be referred to as extract/transform/load (ETL). The extraction phase involves identifying and pulling data from the various source systems that create data and then moving the data to a single repository. Next, the raw data is cleansed, if needed. It's then transformed into a target format that can be fed into operational systems or into a data warehouse, a 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

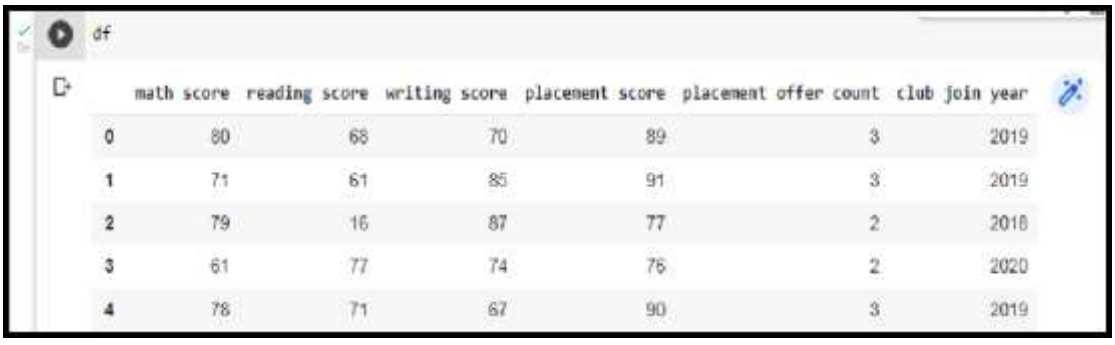
```
import pandas as pd

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.

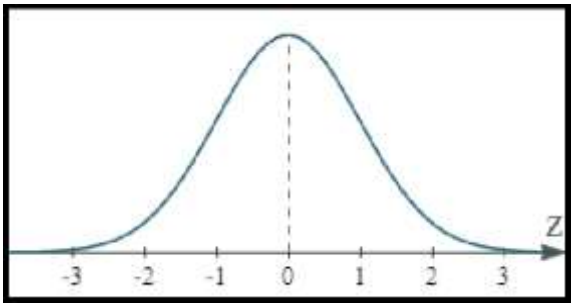
b. To decrease the skewness and convert distribution into normal distribution (Normalization by decimal scaling)

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.

```
[11] 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	76	71	67	90	3	2019	3

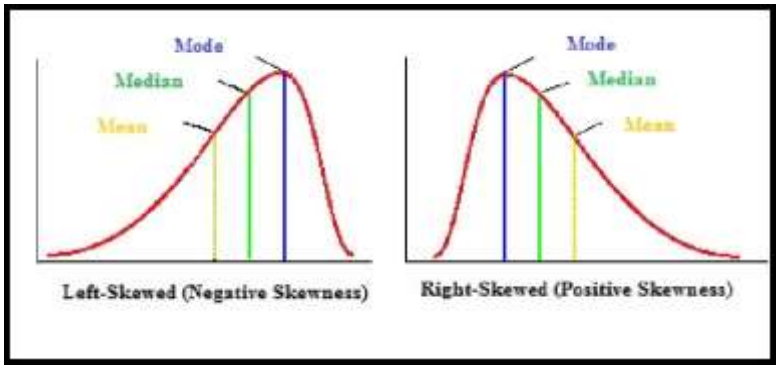
Normal Distribution: In a normal distribution, the graph appears as a classical, symmetrical "bell-shaped curve." The mean, or average, and the mode, or maximum point on the curve, are equal.



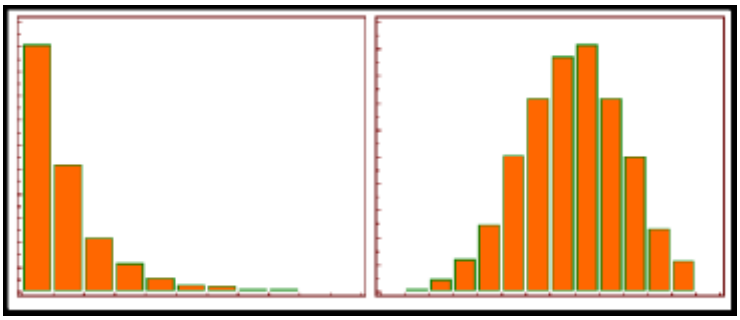
Positively Skewed Distribution

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.



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



Algorithm:

Step 1 : Detecting outliers using Z-Score for the Math_score variable and remove the outliers.

Step 2: Observe the histogram for math_score variable.

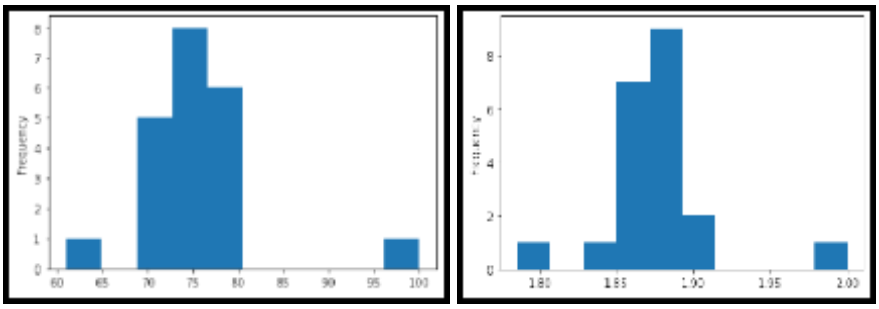
```
import matplotlib.pyplot as plt
new_df['math score'].plot(kind = 'hist')
```

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

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

Step 4: Observe the histogram for math_score variable.

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



It is observed that skewness is reduced at some level.

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

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.**

Group A			
Assignment No 3	**Data Visualization I**	**Page No**	

Title of the Assignment: Data Visualization I

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) variable. 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 categoricalvariable.
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-verginica' ofiris.csv dataset. Provide the codes with outputs and explain everything that you do in thisstep

---

Prerequisite:

1. Basic of Python Programming
2. Concept summary statistics, Categorical variables

Contents for Theory:

1. Pandas Library
2. Qualitative (categorical) variable
3. Summary statistics (mean, median, minimum, maximum, standard deviation)
4. Groupby and describe functions

Categorical Variables

A categorical variable takes only a limited number of values.

Consider a survey that asks how often you eat breakfast and provides four options: "Never","Rarely", "Most days", or "Every day". In this case, the data is categorical, because responsesfall into a fixed set of categories. If people responded to a survey about which what brand of car they owned, the responseswould fall into categories like "Honda", "Toyota", and "Ford". In this case, the data is also categorical.

You will get an error if you try to plug these variables into most machine learning models inPython without preprocessing them first. There are three approaches we can use to preparecategorical data.

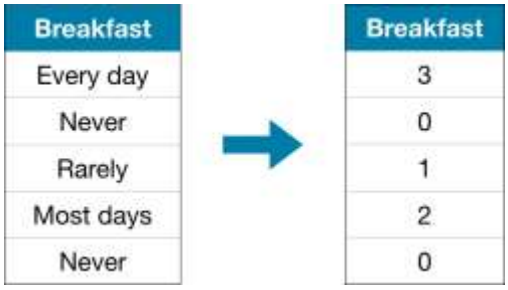
Three Approaches

1. **Drop Categorical Variables**

   The easiest approach to dealing with categorical variables is to simply remove them from thedataset. This approach will only work well if the columns did not contain useful information.

2. **Ordinal Encoding**

   Ordinal encoding assigns each unique value to a different integer.



Breakfast
Every day
Never
Rarely
Most days
Never

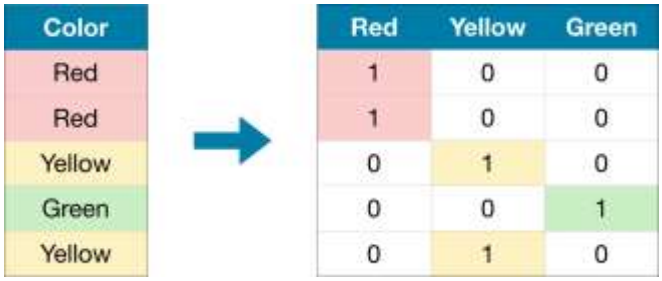
Breakfast
3
0
1
2
0

   This approach assumes an ordering of the categories: "Never" (0) < "Rarely" (1) < "Most days" (2) < "Every day" (3).

   This assumption makes sense in this example, because there is an indisputable ranking to the categories. Not all categorical variables have a clear ordering in the values, but we refer to thosethat do as ordinal variables. For tree-based models (like decision trees and random forests), youcan expect ordinal encoding to work well with ordinal variables.

3. **One-Hot Encoding**

   One-hot encoding creates new columns indicating the presence (or absence) of each possiblevalue in the original data. To understand this, we'll work through an example.



Color
Red
Red
Yellow
Green
Yellow

Red	Yellow	Green
1	0	0
1	0	0
0	1	0
0	0	1
0	1	0

   In the original dataset, "Color" is a categorical variable with three categories: "Red", "Yellow", and "Green". The corresponding one-hot encoding contains one column for each possible value, andone row for each row in the original dataset. Wherever the original value was "Red", we put a 1 in the "Red"

column; if the original value was "Yellow", we put a 1 in the "Yellow" column, and soon.

In contrast to ordinal encoding, one-hot encoding does not assume an ordering of the categories. Thus, you can expect this approach to work particularly well if there is no clear ordering in the categorical data (e.g., "Red" is neither more nor less than "Yellow"). We refer tocategorical variables without an intrinsic ranking as **nominal variables**.

One-hot encoding generally does not perform well if the categorical variable takes on a largenumber of values (i.e., you generally won't use it for variables taking more than 15 different values)

Pandas dataframe.groupby()

Pandas groupby is used for grouping the data according to the categories and apply a functionto the categories. It also helps to aggregate data efficiently.

Syntax: DataFrame.groupby(by=None, axis=0, level=None, as_index=True, sort=True, group_keys=True, squeeze=False, **kwargs)

Parameters :

by : mapping, function, str, or iterable

axis : int, default 0

level : If the axis is a MultiIndex (hierarchical), group by a particular level or levels

as_index : For aggregated output, return object with group labels as the index. Only relevant for DataFrame input. as_index=False is effectively “SQL-style” grouped output

sort : Sort group keys. Get better performance by turning this off. Note this does not influencethe order of observations within each group. groupby preserves the order of rows within eachgroup.

group_keys : When calling apply, add group keys to index to identify pieces

squeeze : Reduce the dimensionality of the return type if possible, otherwise return a consistenttype

Returns : GroupBy object

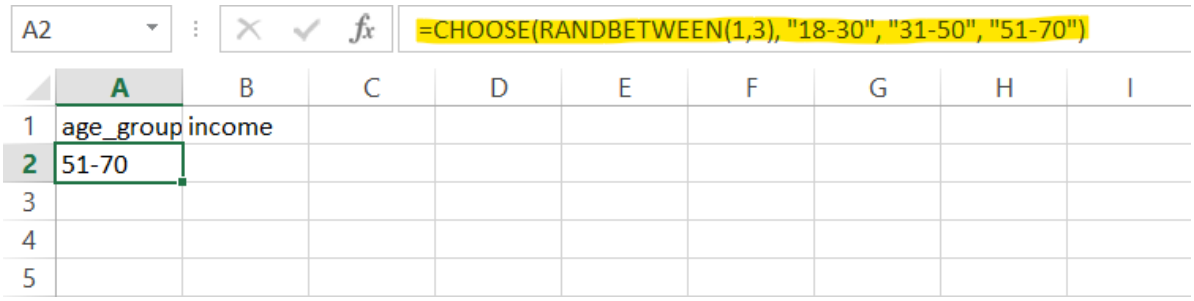
Part 1

Creating Sample data in excel

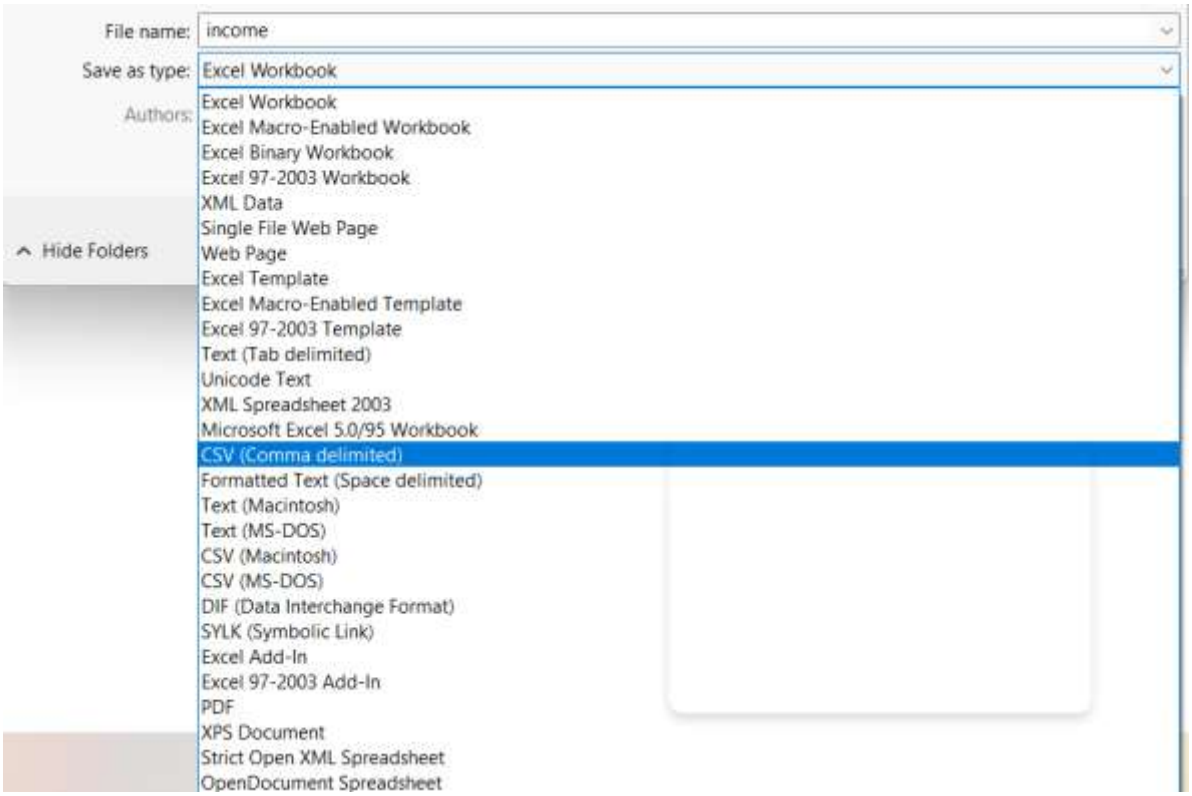
Create two columns age_group **and** income

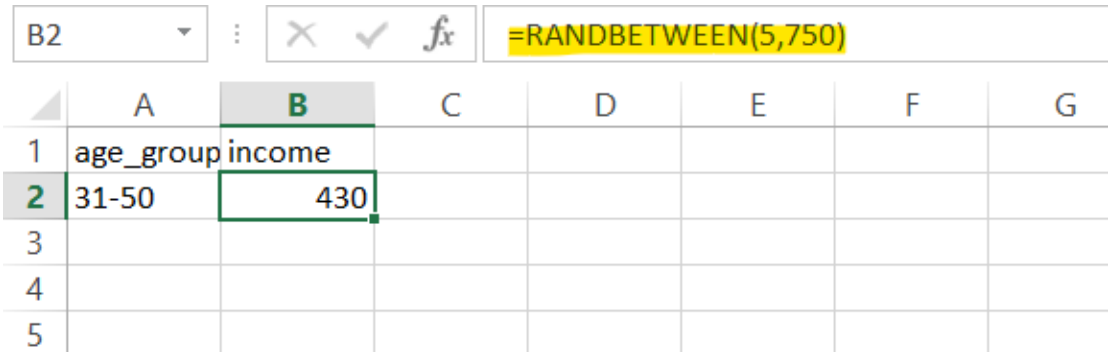
We will create three age group 18-30, 31-50 and 51-70.

Create age group using formula =CHOOSE(RANDBETWEEN(1,3), "18-30", "31-50", "51-70")



Create income column using formula =RANDBETWEEN(5,750)





Save file as income.csv

import libraries -
import pandas as pd

```
df1 = pd.read_csv("income.csv")df1.head()
```

Output :

	age_group	income
0	18-30	739
1	51-70	77
2	18-30	457
3	51-70	242
4	18-30	582

```
df1.age_group.unique()
```

Output :
array(['18-30', '51-70', '31-50'], dtype=object)

Using groupby()

Now that you have checked that the target column is categorical and what values it takes you can try to use a groupby() function. As the name suggests it should group your data into groups. In this case, it will group it into three groups representing different flower species (our target values).

What are the aggregate functions?

Aggregate functions are functions that take a series of entries and return one value that summarizes them in some way.

The good examples are:

count() - Number of non-null observations

max() - Maximum

$$x_i : x_i \leq x_j, i \neq j \; \forall i, j \in n$$

min() - Minimum

$$x_i : x_i \geq x_j, i \neq j \; \forall i, j \in n$$

mean() - Mean of values

$$\overline{x} = \frac{\sum_{i=1}^{N} x_i}{N}$$

std() - Unbiased standard deviation

The **sample standard deviation formula** is:

$$s = \sqrt{\frac{\sum (X - \overline{X})^2}{n - 1}}$$

where,

$s =$ sample standard deviation
$\Sigma =$ sum of...
$\overline{X} =$ sample mean
$n =$ number of scores in sample.

count number of non null income value in each age group -

```
df1.groupby(df1.age_group).count()
```

Output :-

age_group	income
18-30	15
31-50	5
51-70	9

minimum value of income in each age group –

```
df1.groupby(df1.age_group).min()
```

Output :

age_group	income
18-30	118
31-50	193
51-70	18

maximum value of income in each age group –

```
df1.groupby(df1.age_group).max()
```

Output :

age_group	income
18-30	739
31-50	686
51-70	640

mean of income in each age group -

```
df1.groupby(df1.age_group).mean()
```

Output :

age_group	income
18-30	386.066667
31-50	409.400000
51-70	346.111111

standard deviation of income in each age group –

```
df1.groupby(df1.age_group).std()
```

Output :

age_group	income
18-30	218.330178
31-50	218.223051
51-70	221.838029

.describe() method

we can also use describe() on the group by the object to get even all descriptive statistics of our groups

describe gives count, mean, standard deviation, min, max, 25th percentile, 50th percentile and 75th percentile.

```
df1.groupby(df1.age_group).describe()
```

Output :

	income							
	count	mean	std	min	25%	50%	75%	max
age_group								
18-30	15.0	386.066667	218.330178	118.0	174.0	319.0	554.5	739.0
31-50	5.0	409.400000	218.223051	193.0	284.0	285.0	599.0	686.0
51-70	9.0	346.111111	221.838029	18.0	237.0	309.0	539.0	640.0

Part 2

Load iris data set

```
from sklearn import datasets
data = datasets.load_iris()
df = pd.DataFrame(data.data,columns=data.feature_names)
df['species'] = pd.Series(data.target)
df.head()
```

Output :

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

Here species is the categorical variable. We can use unique() function to check what values it takes. And sepal length, sepal width, petal length and petal width are quantitative variables.

```
df.species.unique()
```

Output :

array([0, 1, 2])

We can see that it takes three values: 0, 1, and 2. Here 0, 1, 2 represents ‘Iris-setosa’, ‘Irisversicolor’ and ‘Iris-versicolor’ respectively.

```
df.groupby(df.species)
```

Output :

<pandas.core.groupby.generic.DataFrameGroupBy object at 0x00000260284EFF40>

As you can see the groupby() function returns a DataFrameGroupBy object. Not very useful at first glance. This is why you will need aggregate functions. count of all quantitative variable according to categorical variable(species)

```
df.groupby(df.species).count()
```

Output :

species	sepal length (cm)	sepal width (cm)	petal length (cm)	petal width (cm)
0	50	50	50	50
1	50	50	50	50
2	50	50	50	50

max value of each quantitative variable according to categorical variable(species)

```
df.groupby(df.species).max()
```

Output :

species	sepal length (cm)	sepal width (cm)	petal length (cm)	petal width (cm)
0	5.8	4.4	1.9	0.6
1	7.0	3.4	5.1	1.8
2	7.9	3.8	6.9	2.5

min value of each quantitative variable according to categorical variable(species)

```
df.groupby(df.species).min()
```

Output :

species	sepal length (cm)	sepal width (cm)	petal length (cm)	petal width (cm)
0	4.3	2.3	1.0	0.1
1	4.9	2.0	3.0	1.0
2	4.9	2.2	4.5	1.4

mean of each quantitative variable according to categorical variable(species)

```
df.groupby(df.species).mean()
```

Output :

species	sepal length (cm)	sepal width (cm)	petal length (cm)	petal width (cm)
0	5.006	3.428	1.462	0.246
1	5.936	2.770	4.260	1.326
2	6.588	2.974	5.552	2.026

standard deviation of each quantitative variable according to categorical variable(species)

```
df.groupby(df.species).std()
```

Output :

species	sepal length (cm)	sepal width (cm)	petal length (cm)	petal width (cm)
0	0.352490	0.379064	0.173664	0.105386
1	0.516171	0.313798	0.469911	0.197753
2	0.635880	0.322497	0.551895	0.274650

.describe() on sepal length (cm)

for each categorical variable we get different summary statistics of sepal length column.

```
df.groupby(df.species)["sepal length (cm)"].describe()
```

Output :

species	count	mean	std	min	25%	50%	75%	max
0	50.0	5.006	0.352490	4.3	4.800	5.0	5.2	5.8
1	50.0	5.936	0.516171	4.9	5.600	5.9	6.3	7.0
2	50.0	6.588	0.635880	4.9	6.225	6.5	6.9	7.9

similarly on other columns.

```
df.groupby(df.species)["sepal width (cm)"].describe()
```

Output :

species	count	mean	std	min	25%	50%	75%	max
0	50.0	3.428	0.379064	2.3	3.200	3.4	3.675	4.4
1	50.0	2.770	0.313798	2.0	2.525	2.8	3.000	3.4
2	50.0	2.974	0.322497	2.2	2.800	3.0	3.175	3.8

```
df.groupby(df.species)["petal length (cm)"].describe()
```

Output :

species	count	mean	std	min	25%	50%	75%	max
0	50.0	1.462	0.173664	1.0	1.4	1.50	1.575	1.9
1	50.0	4.260	0.469911	3.0	4.0	4.35	4.600	5.1
2	50.0	5.552	0.551895	4.5	5.1	5.55	5.875	6.9

```
df.groupby(df.species)["petal width (cm)"].describe()
```

Output :

species	count	mean	std	min	25%	50%	75%	max
0	50.0	0.246	0.105386	0.1	0.2	0.2	0.3	0.6
1	50.0	1.326	0.197753	1.0	1.2	1.3	1.5	1.8
2	50.0	2.026	0.274650	1.4	1.8	2.0	2.3	2.5

Questions

Q.1 Explain mean, median mode with example.

Q.2 what is need of statistics?

Q.3 Explain use of groupby function in python

Q.4 Explain percentile with example.

Group A			
Assignment No 4	**Data Analytics I**	**Page No**	

Title of the Assignment: Data Analytics I

1. 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 perform Data Analytics using Python on any open source dataset

---

Prerequisite:

1. Basic of Python Programming
2. Concept of Data Analytics .

- import the required libraries

```
In [1]: import matplotlib.pyplot as plt
        import pandas as pd
        import seaborn as sns

        %matplotlib inline
```

- load the housing data from the scikit-learn library

```
In [2]: from sklearn.datasets import load_boston
        boston_dataset = load_boston()
```

- We print the value of the boston_dataset to understand what it contains.

```
In [3]: print(boston_dataset.keys())
        dict_keys(['data', 'target', 'feature_names', 'DESCR', 'filename'])
```

- *data*: contains the information for various houses
- *target*: prices of the house
- *feature_names*: names of the features
- *DESCR*: describes the dataset

To know more about the features use boston_dataset.DESCR

The description of all the features is given below:

```
CRIM: Per capita crime rate by town
ZN: Proportion of residential land zoned for lots
over 25,000 sq.ft
INDUS: Proportion of non-retail business acres per
town
CHAS: Charles River dummy variable (= 1 if tract
bounds river; 0 otherwise)
NOX: Nitric oxide concentration (parts per 10
million)
RM: Average number of rooms per dwelling
AGE: Proportion of owner-occupied units built prior
to 1940
DIS: Weighted distances to five Boston employment
centers
RAD: Index of accessibility to radial highways
TAX: Full-value property tax rate per $10,000
PTRATIO: Pupil-teacher ratio by town
B: 1000(Bk − 0.63)², where Bk is the proportion of
[people of African American descent] by town
LSTAT: Percentage of lower status of the population
Price: Median value of owner-occupied homes in $1000s
```

The prices of the house indicated by the variable Price is our target variable and the remaining are the feature variables based on which we will predict the value of a house.

We will now load the data into a pandas dataframe using pd.DataFrame. We then print the first 5 rows of the data using head()

```
In [5]: data = pd.DataFrame(boston_dataset.data, columns=boston_dataset.feature_names)
        data.head()
```

Out[5]:

	CRIM	ZN	INDUS	CHAS	NOX	RM	AGE	DIS	RAD	TAX	PTRATIO	B	LSTAT
0	0.00632	18.0	2.31	0.0	0.538	6.575	65.2	4.0900	1.0	296.0	15.3	396.90	4.98
1	0.02731	0.0	7.07	0.0	0.469	6.421	78.9	4.9671	2.0	242.0	17.8	396.90	9.14
2	0.02729	0.0	7.07	0.0	0.469	7.185	61.1	4.9671	2.0	242.0	17.8	392.83	4.03
3	0.03237	0.0	2.18	0.0	0.458	6.998	45.8	6.0622	3.0	222.0	18.7	394.63	2.94
4	0.06905	0.0	2.18	0.0	0.458	7.147	54.2	6.0622	3.0	222.0	18.7	396.90	5.33

We can see that the target value Price is missing from the data. We create a new column of target values and add it to the dataframe.

```
In [6]: data['Price'] = boston_dataset.target
        data.head()
```

Out[6]:

	CRIM	ZN	INDUS	CHAS	NOX	RM	AGE	DIS	RAD	TAX	PTRATIO	B	LSTAT	Price
0	0.00632	18.0	2.31	0.0	0.538	6.575	65.2	4.0900	1.0	296.0	15.3	396.90	4.98	24.0
1	0.02731	0.0	7.07	0.0	0.469	6.421	78.9	4.9671	2.0	242.0	17.8	396.90	9.14	21.6
2	0.02729	0.0	7.07	0.0	0.469	7.185	61.1	4.9671	2.0	242.0	17.8	392.83	4.03	34.7
3	0.03237	0.0	2.18	0.0	0.458	6.998	45.8	6.0622	3.0	222.0	18.7	394.63	2.94	33.4
4	0.06905	0.0	2.18	0.0	0.458	7.147	54.2	6.0622	3.0	222.0	18.7	396.90	5.33	36.2

- Description of Boston dataset

```
In [7]: data.describe()
```

Out[7]:

	CRIM	ZN	INDUS	CHAS	NOX	RM	AGE	DIS	RAD	TAX	PTRATIO	B	L
count	506.000000	506.000000	506.000000	506.000000	506.000000	506.000000	506.000000	506.000000	506.000000	506.000000	506.000000	506.000000	506.0
mean	3.613524	11.363636	11.136779	0.069170	0.554695	6.284634	68.574901	3.795043	9.549407	408.237154	18.455534	356.674032	12.6
std	8.601545	23.322453	6.860353	0.253994	0.115878	0.702617	28.148861	2.105710	8.707259	168.537116	2.164946	91.294864	7.1
min	0.006320	0.000000	0.460000	0.000000	0.385000	3.561000	2.900000	1.129600	1.000000	187.000000	12.600000	0.320000	1.7
25%	0.082045	0.000000	5.190000	0.000000	0.449000	5.885500	45.025000	2.100175	4.000000	279.000000	17.400000	375.377500	6.9
50%	0.256510	0.000000	9.690000	0.000000	0.538000	6.208500	77.500000	3.207450	5.000000	330.000000	19.050000	391.440000	11.3
75%	3.677083	12.500000	18.100000	0.000000	0.624000	6.623500	94.075000	5.188425	24.000000	666.000000	20.200000	396.225000	16.9
max	88.976200	100.000000	27.740000	1.000000	0.871000	8.780000	100.000000	12.126500	24.000000	711.000000	22.000000	396.900000	37.9

- Info of Boston Dataset

```
In [8]: data.info()
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 506 entries, 0 to 505
Data columns (total 14 columns):
 #   Column   Non-Null Count  Dtype
---  ------   --------------  -----
 0   CRIM     506 non-null    float64
 1   ZN       506 non-null    float64
 2   INDUS    506 non-null    float64
 3   CHAS     506 non-null    float64
 4   NOX      506 non-null    float64
 5   RM       506 non-null    float64
 6   AGE      506 non-null    float64
 7   DIS      506 non-null    float64
 8   RAD      506 non-null    float64
 9   TAX      506 non-null    float64
 10  PTRATIO  506 non-null    float64
 11  B        506 non-null    float64
 12  LSTAT    506 non-null    float64
 13  Price    506 non-null    float64
dtypes: float64(14)
memory usage: 55.5 KB
```

- Data preprocessing

After loading the data, it's a good practice to see if there are any missing values in the data. We count the number of missing values for each feature using `isnull()`

```
In [9]: data.isnull().sum()
Out[9]: CRIM       0
        ZN         0
        INDUS      0
        CHAS       0
        NOX        0
        RM         0
        AGE        0
        DIS        0
        RAD        0
        TAX        0
        PTRATIO    0
        B          0
        LSTAT      0
        Price      0
        dtype: int64
```

However, there are no missing values in this dataset as shown above.

- Exploratory Data Analysis

  Exploratory Data Analysis is a very important step before training the model. In this section, we will use some visualizations to understand the relationship of the target variable with other features.

Let's first plot the distribution of the target variable Price. We will use the distplot function from the seaborn library.

Displot

It is used basically for univariant set of observations and visualizes it through a histogram i.e. only one observation and hence we choose one particular column of the dataset.

Syntax:

displot(a[, bins, hist, kde, rug, fit, ...])

Parameters:

1. *a:* Series, 1d-array, or list.

   Observed data. If this is a Series object with a name attribute, the name will be used to label the data axis.

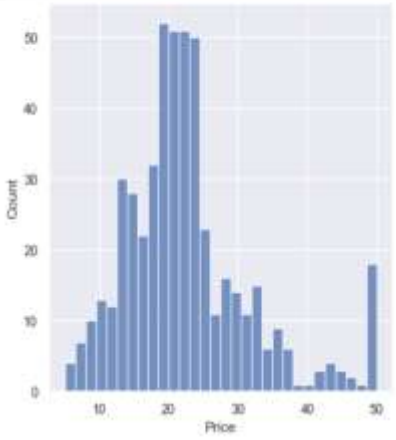
2. *bins:* argument for matplotlib hist(), or None, optional

   Specification of hist bins. If unspecified, as reference rule is used that tries to find a useful default.

3. *hist:* bool, optional

   Whether to plot a (normed) histogram.

```
In [10]: sns.set(rc={'figure.figsize':(11.7,8.27)})
         sns.displot(data['Price'], bins=30);
         plt.show()
```



4. *kde:* bool, optionalWhether to plot a gaussian kernel density estimate.

We see that the values of Price are distributed normally with few outliers.

Next, we create a correlation matrix that measures the linear relationships between the variables. The correlation matrix can be formed by using the corr function from the pandas dataframe library. We will use the heatmap function from the seaborn library to plot the correlation matrix.

```
In [11]: correlation_matrix = data.corr().round(2)
         # annot = True to print the values inside the square
         sns.heatmap(data=correlation_matrix, annot=True);
```

The correlation coefficient ranges from -1 to 1. If the value is close to 1, it means that there is a strong positive correlation between the two variables. When it is close to -1, the variables have a strong negative correlation.

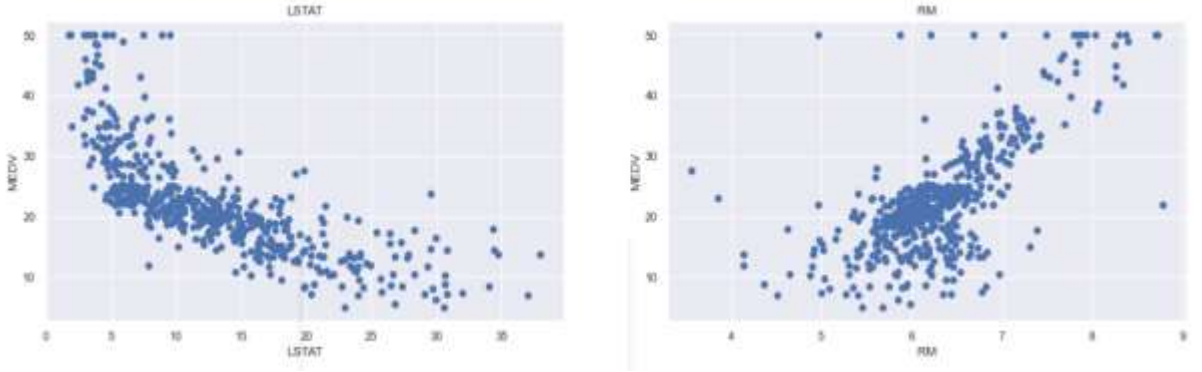
- **Observations:**

- To fit a linear regression model, we select those features which have a high correlation with our target variable Price. By looking at the correlation matrix we can see that RM has a strong positive correlation with Price (0.7) where as LSTAT has a high negative correlation with Price (-0.74).

- An important point in selecting features for a linear regression model is to check for multi-co-linearity. The features RAD, TAX have a correlation of 0.91. These feature pairs are strongly correlated to each other. We should not select both these features together for training the model. Same goes for the features DIS and AGE which have a correlation of -0.75.

Based on the above observations we will RM and LSTAT as our features. Using a scatter plot let's see how these features vary with Price.

```
In [14]: plt.figure(figsize=(20, 5))

features = ['LSTAT', 'RM']
target = data['Price']

for i, col in enumerate(features):
    plt.subplot(1, len(features) , i+1)
    x = data[col]
    y = target
    plt.scatter(x, y, marker='o')
    plt.title(col)
    plt.xlabel(col)
    plt.ylabel('MEDV')
```



Observations:

- The prices increase as the value of RM increases linearly. There are few outliers and the data seems to be capped at 50.
- The prices tend to decrease with an increase in LSTAT. Though it doesn't look to be following exactly a linear line.

Preparing the data for training the model

We concatenate the LSTAT and RM columns using np.c_ provided by the numpy library.

```
In [19]: X = pd.DataFrame(np.c_[data['LSTAT'], data['RM']], columns = ['LSTAT','RM'])
         Y = data['Price']
```

- **Splitting the data into training and testing sets**

Next, we split the data into training and testing sets. We train the model with 80% of the samples and test with the remaining 20%. *We do this to assess the model's performance on unseen data*. To split the data we use `train_test_split` function provided by scikit-learn library. We finally print the sizes of our training and test set to verify if the splitting has occurred properly.

```
In [20]: from sklearn.model_selection import train_test_split

         X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size = 0.2, random_state=5)
         print(X_train.shape)
         print(X_test.shape)
         print(Y_train.shape)
         print(Y_test.shape)

         (404, 2)
         (102, 2)
         (404,)
         (102,)
```

- **Training and testing the model**

We use scikit-learn's `LinearRegression` to train our model on both the training and test sets.

```
In [32]: from sklearn.linear_model import LinearRegression

         model = LinearRegression()
         model.fit(X_train, Y_train)

Out[32]: LinearRegression()
```

- **Model evaluation**

Two metrics that statisticians often use to quantify how well a model fits a dataset are the root mean squared error (RMSE) and the R-squared (R^2), which are calculated as follows:

RMSE: A metric that tells us how far apart the predicted values are from the observed values in a dataset, on average. The lower the RMSE, the better a model fits a dataset.

It is calculated as:

$$RMSE = \sqrt{\Sigma(P_i - O_i)^2 / n}$$

where:

- Σ is a symbol that means "sum"
- P_i is the predicted value for the i^{th} observation
- O_i is the observed value for the i^{th} observation
- n is the sample size

R^2: A metric that tells us the proportion of the variance in the response variable of a regression model that can be explained by the predictor variables. This value ranges from 0 to 1. The higher the R^2 value, the better a model fits a dataset.

It is calculated as:

$R^2 = 1 – (RSS/TSS)$

where:

- RSS represents the sum of squares of residuals
- TSS represents the total sum of squares

Results of Linear Regression i.e. Mean Squared Error and R2 Score.

```
In [33]: from sklearn.metrics import mean_squared_error
         from sklearn.metrics import r2_score

         # model evaluation for testing set
         y_pred = model.predict(X_test)

         rmse = (np.sqrt(mean_squared_error(Y_test, y_pred)))
         r2 = r2_score(Y_test, y_pred)

         print("The model performance for testing set")
         print("--------------------------------------")
         print('RMSE is {}'.format(rmse))
         print('R2 score is {}'.format(r2))

         The model performance for testing set
         --------------------------------------
         RMSE is 5.137400784702911
         R2 score is 0.6628996975186953
```

- **Predicting selling price –**

```
In [35]: # Produce a matrix for sample data
         sample_data = [[6.89, 9.939]]

         # Show predictions
         price = model.predict(sample_data)
         print("Predicted selling price for house: ${:,.2f}".format(price[0]))

         Predicted selling price for house: $43.41
```

Questions :

Q.1 Explain linear regression with example.

Q.2 What is RMSE?

Q.3 Write syntax for test_split function?

Q.4 Write syntax for function of fit_transform and r2_score and mean_squared_error?

Group A			
Assignment No 5	**Data Analytics II**	**Page No**	

Title of the Assignment: Data Analytics II

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 perform the data Analytics operation using Python on any open source dataset

---

Prerequisite:

1. Basic of Python Programming
2. Concept of Data Analytics

Contents for Theory:

1. Logistic Regression
2. Logistic Model
3. Loss Function
4. The Gradient Descent Algorithm
5. Implementation
6. Confusion Matrix

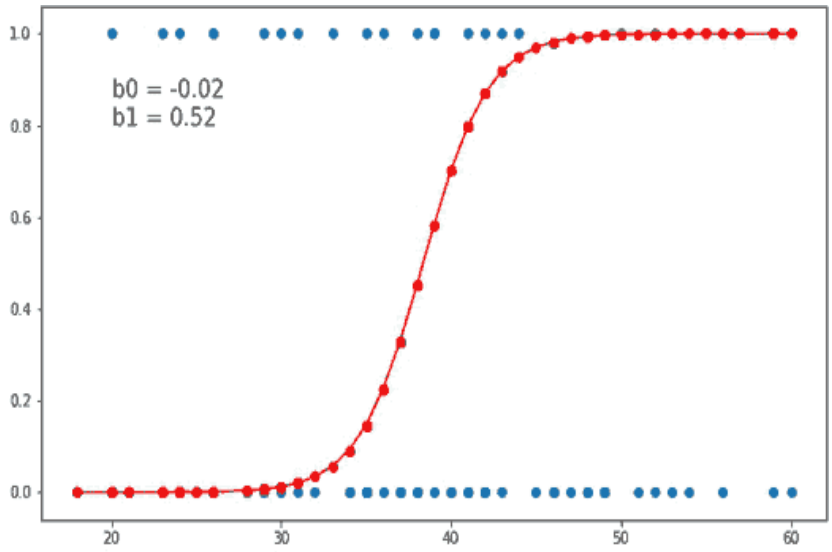
---

1. Logistic Regression

In statistics logistic regression is used to model the probability of a certain class or event.

Logistic regression is similar to linear regression because both of these involve estimating the values

of parameters used in the prediction equation based on the given training data. Linear regression predicts the value of some continuous, dependent variable. Whereas logistic regression predicts the probability of an event or class that is dependent on other factors. Thus the output of logistic regression always lies between 0 and 1. Because of this property it is commonly used for classification purpose.



2. Logistic Model

Consider a model with features $x_1, x_2, x_3 \ldots x_n$. Let the binary output be denoted by Y, that can take the values 0 or 1.

Let p be the probability of $Y = 1$, we can denote it as $p = P(Y=1)$.

The mathematical relationship between these variables can be denoted as:

$$\left(\frac{P}{1-p} \right) = b0 + b1x1 + b2x2 + b3x3 \ldots b_n x_n$$

Here the term $\frac{p}{1-p}$ is known as the *odds* and denotes the likelihood of the event taking place.

Thus $\ln(\frac{p}{1-p})$ is known as the *log odds* and is simply used to map the probabilty that lies between 0 and 1 to a range between $(-\infty, +\infty)$. The terms $b_0, b_1, b_2 \ldots$ are parameters (or weights) that we will estimate during training.

So this is just the basic math behind what we are going to do. We are interested in the probability *p* in this equation. So we simplify the equation to obtain the value of p:

1. The log term ln on the LHS can be removed by raising the RHS as a power of e:

$$\frac{p}{1-p} = e^{b_0+b_1x_1+b_2x_2+b_3x_3\ldots b_nx_n}$$

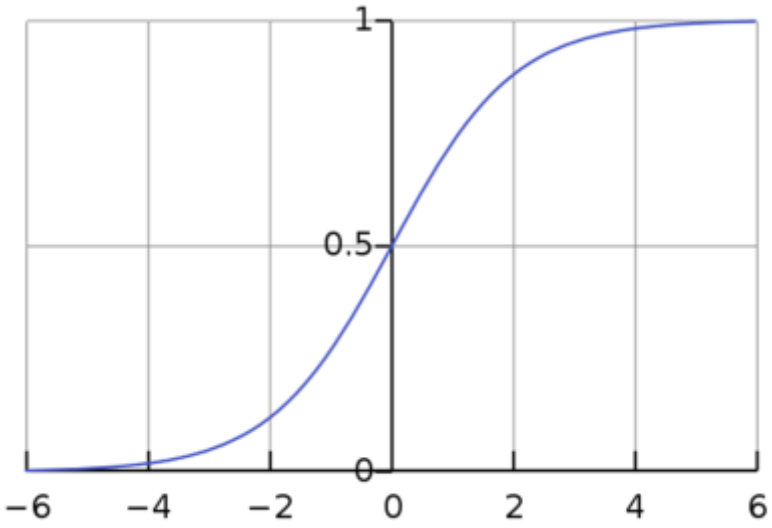
2. Now we can easily simplify to obtain the value of p:

$$p = \frac{e^{b_0+b_1x_1+b_2x_2+b_3x_3\ldots b_nx_n}}{1 + e^{b_0+b_1x_1+b_2x_2+b_3x_3\ldots b_nx_n}}$$

or

$$p = \frac{1}{1 + e^{-(b_0+b_1x_1+b_2x_2+b_3x_3\ldots b_nx_n)}}$$

This actually turns out to be the equation of the *Sigmoid Function* which is widely used in other machine learning applications. The *Sigmoid Function* is given by:



$$S(x) = \frac{1}{1 + e^{-x}}$$

Now we will be using the above derived equation to make our predictions. Before that we will train our model to obtain the values of our parameters b0, b1, b2. . . that result in least error. This is where the error or loss function comes in.

3. Loss Function

The loss is basically the error in our predicted value. In other words it is a difference between our predicted value and the actual value. We will be using the L2 Loss Function to calculate the error. Theoretically you can use any function to calculate the error. This function can be broken down as:

1. Let the actual value be yi. Let the value predicted using our model be denoted as $\bar{y_i}$ Find the difference between the actual and predicted value.
2. Square this difference.
3. Find the sum across all the values in training data.

$$L = \sum_{i=1}^{n} (y_i - \bar{y_i})^2$$

Now that we have the error, we need to update the values of our parameters to minimize this error. This is where the "learning" actually happens, since our model is updating itself based on it's previous output to obtain a more accurate output in the next step. We will be using the Gradient Descent Algorithm to estimate our parameters. Another commonly used algorithm is the Maximum Likelihood Estimation.

4. The Gradient Descent Algorithm

You might know that the partial derivative of a function at it’s minimum value is equal to 0. So gradient descent basically uses this concept to estimate the parameters or weights of our model by minimizing the loss function.

For simplicity, for the rest of this part let us assume that our output depends only on a single feature x. So we can rewrite our equation as:

$$\bar{y_i} = p = \frac{1}{1 + e^{-(b_0 + b_1 x_i)}} = \frac{1}{1 + e^{-b_0 - b_1 x_i}}$$

Thus we need to estimate the values of weights b0 and b1 using our given training data. Initially let b0=0 and b1=0. Let L be the learning rate. The learning rate controls by how much the values of b0 and b1 are updated at each step in the learning process. Here let L=0.001. Calculate the partial derivative with respect to b0 and b1. The value of the partial derivative will tell us how far the loss function is from it's minimum value. It is a measure of how much our weights need to be updated to attain minimum or ideally 0 error. In case you have more than one feature, you need to calculate the partial derivative for each weight b0, b1 … bn where n is the number of features.

$$D_{b_0} = -2 \sum_{i=1}^{n} (y_i - \bar{y_i}) \times \bar{y_i} \times (1 - \bar{y_i})$$

$$D_{b_1} = -2 \sum_{i=1}^{n} (y_i - \bar{y_i}) \times \bar{y_i} \times (1 - \bar{y_i}) \times x_i$$

Next we update the values of b0 and b1:

$$b_0 = b_0 - L \times D_{b_0}$$

$$b_1 = b_1 - L \times D_{b_1}$$

We repeat this process until our loss function is a very small value or ideally reaches 0 (meaning no errors and 100% accuracy). The number of times we repeat this learning process is known as iterations or epochs.

5. Implementation

I. Implementing the Model

The data describes information about a product being purchased through an advertisement on social media. We will be predicting the value of Purchased and consider a single feature, Age to predict the values of Purchased. You can have multiple features as well.

In [1]:
```
# Importing libraries
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from math import exp


# Download the dataset
# Source of dataset - https://www.kaggle.com/rakeshrau/social-network-ads

# Load the data
data = pd.read_csv("data.csv")
data.head()
```

Out[1]:

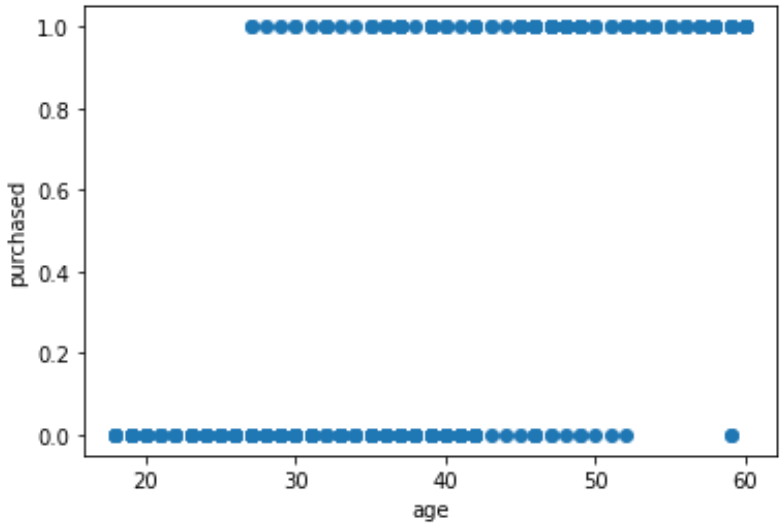
	User ID	Gender	Age	EstimatedSalary	Purchased
0	15624510	Male	19	19000	0
1	15810944	Male	35	20000	0
2	15668575	Female	26	43000	0
3	15603246	Female	27	57000	0
4	15804002	Male	19	76000	0

In [2]:
```
data.describe()
```

Out[2]:

	User ID	Age	EstimatedSalary	Purchased
count	4.000000e+02	400.000000	400.000000	400.000000
mean	1.569154e+07	37.655000	69742.500000	0.357500
std	7.165832e+04	10.482877	34096.960282	0.479864
min	1.556669e+07	18.000000	15000.000000	0.000000
25%	1.562676e+07	29.750000	43000.000000	0.000000
50%	1.569434e+07	37.000000	70000.000000	0.000000
75%	1.575036e+07	46.000000	88000.000000	1.000000
max	1.581524e+07	60.000000	150000.000000	1.000000

In [3]:
```
# Visualizing the dataset
plt.scatter(data['Age'], data['Purchased'])
plt.xlabel("age")
plt.ylabel("purchased")
plt.show()

# Divide the data to training set and test set
X_train, X_test, y_train, y_test = train_test_split(data['Age'], data['Purchased'],
test_size=0.20)
```



We need to normalize our training data, and shift the mean to the origin. This is important to get accurate results because of the nature of the logistic equation. This is done by the normalize method. The predict method simply plugs in the value of the weights into the logistic model equation and returns the result. This returned value is the required probability.

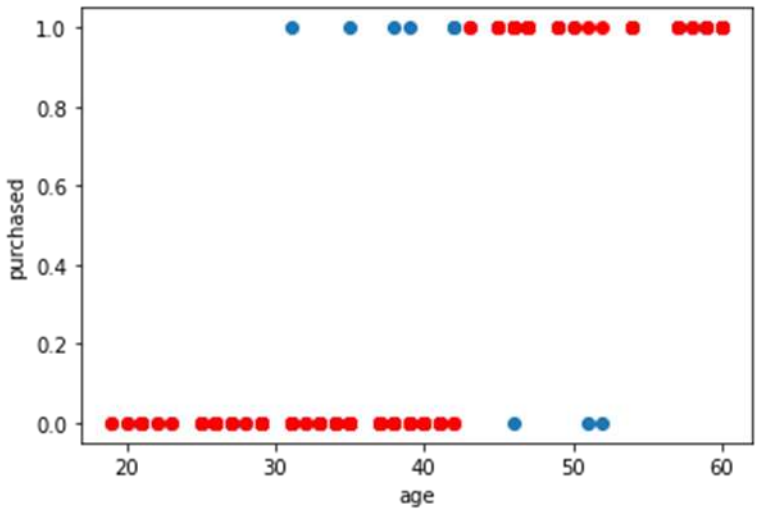
The model is trained for 300 epochs or iterations. The partial derivatives are calculated at each iterations and the weights are updated. You can even calculate the loss at each step and see how it approaches zero with each step.

II. Implementing using Sklearn

The library sklearn can be used to perform logistic regression in a few lines as shown using the LogisticRegression class. It also supports multiple features. It requires the input values to be in a specific format hence they have been reshaped before training using the fit method.

The accuracy using this is 86.25%, which is very close to the accuracy of our model that we implemented from scratch !

```
# Making predictions using scikit learn
from sklearn.linear_model import LogisticRegression

# Create an instance and fit the model
model = LogisticRegression()
model.fit(X_train.values.reshape(-1, 1), y_train.values.reshape(-1, 1).ravel())

# Making predictions
y_pred_sk = model.predict(X_test.values.reshape(-1, 1))
plt.clf()
plt.scatter(X_test, y_test)
plt.scatter(X_test, y_pred_sk, c="red")

plt.xlabel("age")
plt.ylabel("purchased")
plt.show()
#Accuracy
print(f"Accuracy = {lr_model.score(X_test.values.reshape(-1, 1),
y_test.values.reshape(-1, 1))}")
```



6. Confusion Matrix

	Actually Positive (1)	Actually Negative (0)
Predicted Positive (1)	True Positives (TPs)	False Positives (FPs)
Predicted Negative (0)	False Negatives (FNs)	True Negatives (TNs)

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

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

$$F1 = \frac{2 \times precision \times recall}{precision + recall}$$

$$accuracy = \frac{TP + TN}{TP + FN + TN + FP}$$

$$specificity = \frac{TN}{TN + FP}$$

A confusion matrix is a table that is often used to describe the performance of a classification model (or "classifier") on a set of test data for which the true values are known

Some Basic Terms

True Positive -

Label which was predicted Positive and is actually Positive.

True Negative -

Label which was predicted Negative and is actually Negative.

False Positive -

Label which was predicted as Positive, but is actually Negative. In Hypothesis Testing it is also known as Type 1 error or the incorrect rejection of Null Hypothesis, refer this to read more about Hypothesis testing.

False Negatives -

Labels which was predicted as Negative, but is actually Positive. It is also known as Type 2 error, which leads to the failure in rejection of Null Hypothesis.

In [7]:
```
from sklearn.metrics import confusion_matrix

#extracting true_positives, false_positives, true_negatives, false_negatives
tn, fp, fn, tp = confusion_matrix(y_test, y_pred).ravel()
print("True Negatives: ",tn)
print("False Positives: ",fp)
print("False Negatives: ",fn)
print("True Positives: ",tp)
```

```
True Negatives:  48
False Positives:  3
False Negatives:  7
True Positives:  22
```

In [8]:
```
#Accuracy
Accuracy = (tn+tp)*100/(tp+tn+fp+fn)
print("Accuracy  {:0.2f}%:".format(Accuracy))
```

```
Accuracy 87.50%:
```

Precision

It is the 'Exactness', ability of the model to return only relevant instances. If your use case/problem statement involves minimizing the False Positives, i.e. in current scenario if you don't want the Forged Notes to be labelled as Authentic by the Model then Precision is something you need.

Precision = tp/(tp+fp)

In [9]:
```
#Precision
Precision = tp/(tp+fp)
print("Precision  {:0.2f}".format(Precision))
```

```
Precision 0.88
```

Recall

It is the 'Completeness', ability of the model to identify all relevant instances, True Positive Rate, aka Sensitivity. In the current scenario if your focus is to have the least False Negatives i.e. you don't Authentic Notes to be wrongly classified as Forged then Recall can come to your rescue.

Recall = tp/(tp+fn)

In [10]:
```
#Recall
Recall = tp/(tp+fn)
print("Recall  {:0.2f}".format(Recall))
```

```
Recall 0.76
```

Error rate

Error rate (ERR) is calculated as the number of all incorrect predictions (FN + FP) divided by the total number of the dataset (P + N). The best error rate is 0.0, whereas the worst is 1.0.

$$\mathrm{ERR} = \frac{\mathrm{FP} + \mathrm{FN}}{\mathrm{TP} + \mathrm{TN} + \mathrm{FN} + \mathrm{FP}} = \frac{\mathrm{FP} + \mathrm{FN}}{\mathrm{P} + \mathrm{N}}$$

In [11] :
```
#Error rate
err = (fp + fn)/(tp + tn + fn + fp)
print("Error  rate  {:0.2f}".format(err))
```

```
Error rate 0.12
```

Questions

Q.1 What is confusion matrix?

Q.2 What is difference between linear and logistic regression?

Q.3 Obtain sigmoid function.

Q.4 Explain gradient descent function

Group A			
Assignment No 6	**Data Analytics III**	**Page No**	

Title of the Assignment: Data Analytics III

1. Implement simple Naive 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.

Objectives of assignments: Student should be able to implement Naïve Bayes classification algorithm using python/R and confusion matrix.

Prerequisite:

1. Basic of python programming
2. Concept of Nave Bayes classification algorithm
3. Confusion matrix concept

Content for theory:

Naive Bayes is one such algorithm in classification that can never be overlooked upon due to its special characteristic of being "**naive**". It makes the assumption that features of a measurement are independent of each other.

For example, an animal may be considered as a cat if it has cat eyes, whiskers and a long tail. Even if these features depend on each other or upon the existence of the other features, all of these properties independently contribute to the probability that this animal is a cat and that is why it is known as 'Naive'.

According to Bayes Theorem, the various features are mutually independent. For two independent events, P(A,B) = P(A)P(B). This assumption of Bayes Theorem is probably never encountered in practice, hence it accounts for the "naive" part in Naive Bayes. Bayes' Theorem is stated as: **P(a|b) = (P(b|a) * P(a)) / P(b).** Where P(a|b) is the probability of a given b.

Let us understand this algorithm with a simple example. The Student will be a pass if he wears a

"red" color dress on the exam day. We can solve it using above discussed method of posterior probability.

By Bayes Theorem, P(Pass| Red) = P(Red| Pass) * P(Pass) / P (Red).

From the values, let us assume P (Red|Pass) = 3/9 = 0.33, P(Red) = 5/14 = 0.36, P(Pass)= 9/14 =

0.64. Now, P (Pass| Red) = 0.33 * 0.64 / 0.36 = 0.60, which has higher probability. In this way, Naive Bayes uses a similar method to predict the probability of different class based on various attributes.

Problem Analysis:

To implement the Naive Bayes Classification, we shall use a very famous Iris Flower Dataset that consists of 3 classes of flowers. In this, there are 4 independent variables namely the, **sepal_length**, **sepal_width**, **petal_length** and **petal_width**. The dependent variable is the **species** which we will predict using the four independent features of the flowers.

Step 1**:** Importing the Libraries. As always, the first step will always include importing the libraries which are the NumPy, Pandas and the Matplotlib.

```
import numpy as np  import
matplotlib.pyplot as plt import pandas as
pd
```

Step 2: Importing the dataset.

In this step, we shall import the Iris Flower dataset which is stored in my github repository as IrisDataset.csv and save it to the variable dataset. After this, we assign the 4 independent variables to *X* and the dependent variable 'species' to *Y*. The first 5 rows of the dataset are displayed.

```
dataset = pd.read_csv('https://raw.githubusercontent.com/mk-

gurucharan/Classification/master/IrisDataset.csv')
X=dataset.iloc[:,:4].values
```

```
 y = dataset['species'].values
print(dataset.head(5))
>> sepal_length  sepal_width  petal_length  petal_width
species
 5.1           3.5          1.4           0.2           setosa
 4.9           3.0          1.4           0.2           setosa
4.7           3.2          1.3           0.2          setosa
4.6           3.1          1.5           0.2          setosa
5.0           3.6          1.4           0.2          setosa
```

Step 3: Splitting the dataset into the Training set and Test set

Once we have obtained our data set, we have to split the data into the training set and the test set. In this data set, there are 150 rows with 50 rows of each of the 3 classes. As each class is given in a continuous order, we need to randomly split the dataset. Here, we have the test_size=0.2, which means that ***20%*** of the dataset will be used for testing purpose as the ***test set*** and the remaining ***80%*** will be used as the ***training set*** for training the Naive Bayes classification model.

```
from sklearn.model_selection import train_test_split X_train, X_test, y_train, y_test =
train_test_split(X, y, test_size = 0.2)
```

Step 4: Feature Scaling

The dataset is scaled down to a smaller range using the Feature Scaling option. In this, both the X_train and X_test values are scaled down to smaller values to improve the speed of the program.

```
from sklearn.preprocessing import
StandardScaler sc = StandardScaler()
X_train = sc.fit_transform(X_train)
X_test = sc.transform(X_test)
```

Step 5: Training the Naive Bayes Classification model on the Training Set

In this step, we introduce the class GaussianNB that is used from the sklearn.naive_bayes library.

Here, we have used a Gaussian model, there are several other models such as Bernoulli, Categorical and Multinomial. Here, we assign the GaussianNB class to the variable classifier and fit the X_train and y_train values to it for training purpose.

```
from sklearn.naive_bayes import GaussianNB
classifier = GaussianNB()

classifier.fit(X_train, y_train)
```

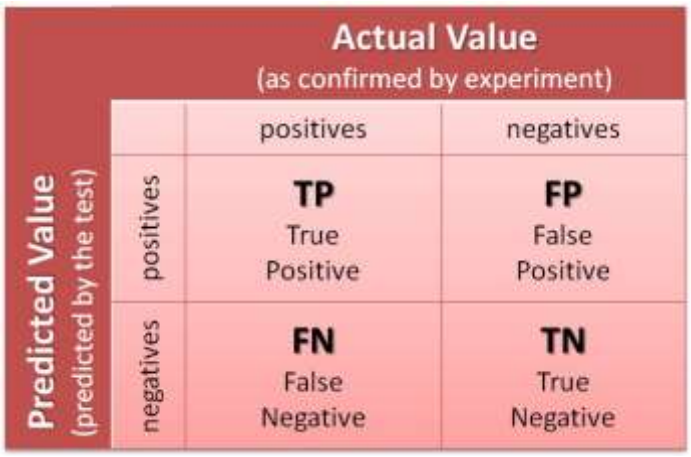
Step 6: Predicting the Test set results

Once the model is trained, we use the the classifier.predict() to predict the values for the Test set and the values predicted are stored to the variable y_pred.

```
y_pred = classifier.predict(X_test)

>>y_pred
```

Step 7: Confusion Matrix and Accuracy

This is a step that is mostly used in classification techniques. In this, we see the Accuracy of the trained model and plot the confusion matrix. The confusion matrix is a table that is used to show the number of correct and incorrect predictions on a classification problem when the real values of the Test Set are known. It is of the format



The True values are the number of correct predictions made.

```
from sklearn.metrics import confusion_matrix  cm =
confusion_matrix(y_test, y_pred) from sklearn.metrics import
accuracy_score
print("Accuracy : ", accuracy_score(y_test,
y_pred))
print(cm)
>>Accuracy :  0.9666666666666667
>>array([[14,  0,  0],
       [ 0,  7,  0],
       [ 0,  1,  8]])
```

From the above confusion matrix, we infer that, out of 30 test set data, 29 were correctly classified and only 1 was incorrectly classified. This gives us a high accuracy of 96.67%.

Step 8: Comparing the Real Values with Predicted Values

In this step, a Pandas DataFrame is created to compare the classified values of both the original Test set (***y_test***) and the predicted results (***y_pred***).

```
df = pd.DataFrame({'Real Values':y_test, 'Predicted Values':y_pred})
print(df)
>>
Real Values   Predicted Values
setosa          setosa setosa
setosa
virginica     virginica
versicolor    versicolor
```

```
setosa          setosa setosa
setosa

...   ...      ...    ...    ... virginica
versicolor

virginica     virginica

setosa          setosa setosa
setosa
versicolor    versicolor versicolor    versicolor
```

Step 9**:** Compute Error rate, Precision and Recall

We have to import precision_score, recall_score, accuracy_score from sklearn.metrics

```
from sklearn.metrics import precision_score, recall_score, accuracy_score
```

Error rate (ERR) is calculated as the number of all incorrect predictions divided by the total number of the dataset. The best error rate is 0.0, whereas the worst is 1.0. Error rate= (FP+FN)/total

```
m=accuracy_score(y_test, y_pred)
print("error rate:-",1-m)
error rate:- 0.033333333333333326
```

The precision is the ratio TP / (TP + TP) where TP is the number of true positives and FP the number of false positives. The precision is intuitively the ability of the classifier not to label as positive a sample that is negative.

```
print('Precision:',precision_score(y_test,y_pred,average='micro')
)
>>Precision: 0.9666666666666667
```

The recall is the ratio TP / (TP + FN) where TP is the number of true positives and FN the number of false negatives. The recall is intuitively the ability of the classifier to find all the positive samples.

```
print("Recall Score:",recall_score(y_test,y_pred,average='micro'))
```

```
>> Recall Score :  0.9333333333333333
```

Conclusion-

Thus, in this way we have successfully been able to build a ***Naive Bayes Classification*** Model that is able to classify a flower depending upon 4 characteristic features. This model can be implemented and tested with several other classification datasets that are available on the net.

Questions:

1. Explain *Naive Bayes Classification* Model with Suitable example.
2. Explain confusion matrix with suitable example.
3. What is Accuracy and Error rate?
4. What is Precision and Recall?
5. What is Specificity and Sensitivity?

Group A			
Assignment No 7	**Text Analytics**	**Page No**	

Title of the Assignment: Text Analytics

1. Extract Sample document and apply following document pre-processing methods:

   Tokenization, POS Tagging, stop words removal, Stemming and Lemmatization.

2. Create representation of document by calculating Term Frequency and Inverse Document Frequency.

Objectives of assignments: Student should be able to implement text analytics operations using python.

Prerequisite:

1. Basic of python programming
2. Concept of python NLKT (Natural language toolkit)
3. Apply pre-processing methods such as Tokenization, POS Tagging, stop words removal, Stemming and Lemmatization

---

Content for theory:

1. **Text Analytics and NLP**
2. **Text Analysis Operations using NLTK**
3. **Tokenization**
4. **Sentence Tokenization**
5. **Word Tokenization**
6. **Frequency Distribution**
7. **POS Tagging**
8. **Stop words**
9. **Removing Stop words**
10. **Stemming:**

1.Text Analytics and NLP:

Text Analytics has lots of applications in today's online world. By analysing tweets on Twitter, we can find trending news and peoples reaction on a particular event. Amazon can understand user feedback or review on the specific product. Book My Show can discover people's opinion about the movie. YouTube can also analyse and understand people's viewpoints on a video. Text communication is one of the most popular forms of day-to-day conversion. We chat, message, tweet, share status, email, write blogs, share opinion and feedback in our daily routine. All of these activities are generating text in a significant amount, which is unstructured in nature. In this area of the online marketplace and social media, it is essential to analyse vast quantities of data, to understand people's opinion.

NLP enables the computer to interact with humans in a natural manner. It helps the computer to understand the human language and derive meaning from it. NLP is applicable in several problematic from speech recognition, language translation, classifying documents to information extraction. Analysing movie review is one of the classic examples to demonstrate a simple NLP Bag-of-words model, on movie reviews.

2.Text Analysis Operations using NLTK:

NLTK is a powerful Python package that provides a set of diverse natural languages algorithms. It is free, opensource, easy to use, large community, and well documented. NLTK consists of the most common algorithms such as tokenizing, part-of-speech tagging, stemming, sentiment analysis, topic segmentation, and named entity recognition. NLTK helps the computer to analysis, pre-process, and understand the written text.

```
!pip install nltk
 Requirement already satisfied: nltk
 in/home/northout/anaconda2/lib/python2.7/site-packages
 Requirement already satisfied: six i
 /home/northout/anaconda2/lib/python2.7/site-packages (from nltk)
 [33mYou are using pip version 9.0.1, however version 10.0.1 is
 available.
 You should consider upgrading via the 'pip install --upgrade pip'
 command.[0m
```

```
#Loading NLTK
import nltk

nltk.download('punkt')
```

3.Tokenization:

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

4.Sentence Tokenization:

Sentence tokenizer breaks text paragraph into sentences.

```
from nltk.tokenize import sent_tokenize

text="""Hello Mr. Smith, how are you doing today? The weather is
great, and city is awesome. The sky is pinkish-blue. You shouldn't
eat cardboard"""

tokenized_text=sent_tokenize(text)

print(tokenized_text)
```

```
>>['Hello Mr. Smith, how are you doing today?', 'The weather is     great, and city is awesome.', 'The sky is pinkish-blue.', "You shouldn't eat cardboard"]
```

Here, the given text is tokenized into sentences.

5.Word Tokenization:

Word tokenizer breaks text paragraph into words.

```
from nltk.tokenize import word_tokenize
tokenized_word=word_tokenize(text)

print(tokenized_word)
```

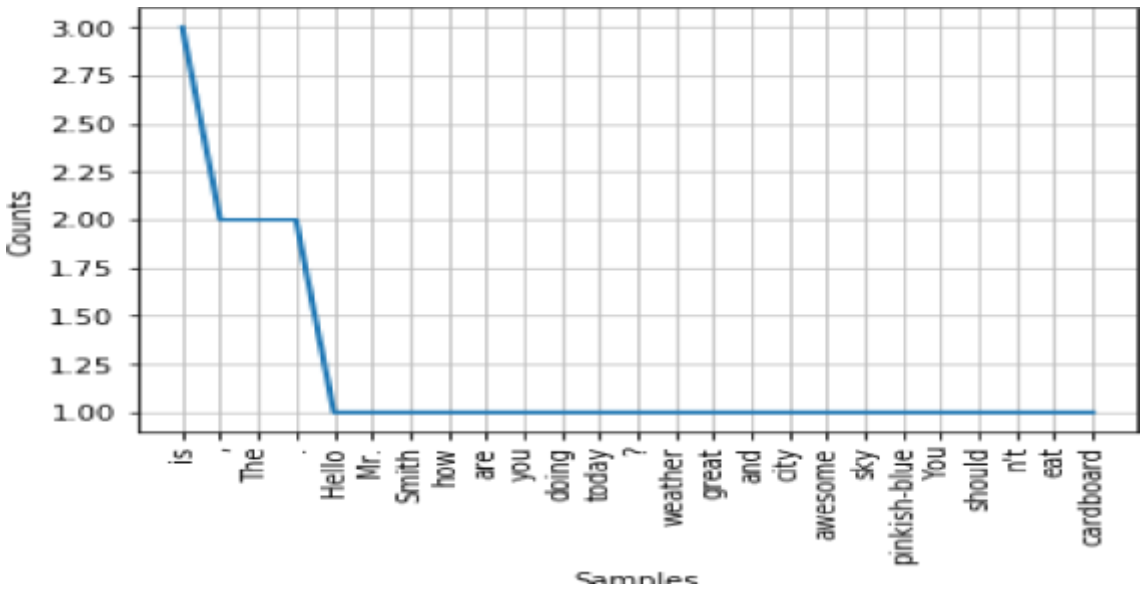
```
>>['Hello', 'Mr.', 'Smith', ',', 'how', 'are', 'you', 'doing',
'today', '?', 'The', 'weather', 'is', 'great', ',', 'and', 'city',
'is', 'awesome', '.', 'The', 'sky', 'is', 'pinkish-blue', '.',
'You', 'should', "n't", 'eat', 'cardboard']
```

6.Frequency Distribution:

```
from nltk.probability import FreqDist
fdist = FreqDist(tokenized_word)
print(fdist)
```

```
<FreqDist with 25 samples and 30 outcomes>

fdist.most_common(2)

>>[('is', 3), (',', 2)]
```

```
#Frequency Distribution Plot
#pip install matplotlib
import matplotlib.pyplot as plt
fdist.plot(30,cumulative=False)
plt.show()
```



6.POS Tagging:

The primary target of Part-of-Speech (POS) tagging is to identify the grammatical group of a given word. Whether it is a NOUN, PRONOUN, ADJECTIVE, VERB, ADVERBS, etc. based on the context. POS Tagging looks for relationships within the sentence and assigns a corresponding tag to the word.

```
sent = "Albert Einstein was born in Ulm, Germany in 1879."
```

```
tokens=nltk.word_tokenize(sent)
print(tokens)
```

```
>>['Albert', 'Einstein', 'was', 'born', 'in', 'Ulm', ',',
'Germany', 'in', '1879', '.']
```

```
nltk.download('averaged_perceptron_tagger') nltk.pos_tag(tokens)
```

```
[('Albert', 'NNP'),
 ('Einstein', 'NNP'),
('was', 'VBD'),
 ('born', 'VBN'),
 ('in', 'IN'),
 ('Ulm', 'NNP'),
 (',', ','),
 ('Germany', 'NNP'),
 ('in', 'IN'),
 ('1879', 'CD'),
 ('.', '.')]
```

7.Stopwords:

Stop words 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 stop words, you need to create a list of stop words and filter out your list of tokens from these words.

```
from nltk.corpus import stopwords
stop_words=set(stopwords.words("english"))  print(stop_words)
```

```
>>{'their', 'then', 'not', 'ma', 'here', 'other', 'won', 'up', 'weren', 'being', 'we', 'those', 'an', 'them', 'which',
'him', 'so', 'yourselves', 'what', 'own', 'has', 'should', 'above', 'in', 'myself', 'against', 'that', 'before', 't',
'just', 'into', 'about', 'most', 'd', 'where', 'our', 'or', 'such', 'ours', 'of', 'doesn', 'further', 'needn', 'now',
'some', 'too', 'hasn', 'more', 'the', 'yours', 'her', 'below', 'same', 'how', 'very', 'is', 'did', 'you', 'his',
'when', 'few', 'does', 'down', 'yourself', 'i', 'do', 'both', 'shan', 'have', 'itself', 'shouldn', 'through',
'themselves', 'o', 'didn', 've', 'm', 'off', 'out', 'but', 'and', 'doing', 'any', 'nor', 'over', 'had', 'because',
'himself', 'theirs', 'me', 'by', 'she', 'whom', 'hers', 're', 'hadn', 'who', 'he', 'my', 'if', 'will', 'are', 'why',
'from', 'am', 'with', 'been', 'its', 'ourselves', 'ain', 'couldn', 'a', 'aren', 'under',
'll', 'on', 'y', 'can', 'they', 'than', 'after', 'wouldn', 'each', 'once', 'mightn', 'for', 'this', 'these', 's', 'only',
'haven', 'having', 'all', 'don', 'it', 'there', 'until', 'again', 'to', 'while', 'be', 'no', 'during', 'herself', 'as',
'mustn', 'between',
'was', 'at', 'your', 'were', 'isn', 'wasn'}
```

8.Removing Stop words:

```
filtered_sent=[]
for w in tokenized_sent:
    if w not in stop_words:
        filtered_sent.append(w)

print("Tokenized Sentence:",tokenized_sent)

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

```
>>Tokenized Sentence: ['Hello', 'Mr.', 'Smith', ',', 'how', 'are',

'you', 'doing', 'today', '?']
```

```
Filterd Sentence: ['Hello', 'Mr.', 'Smith', ',', 'today', '?']
```

9.Stemming:

Stemming is a process of linguistic normalization, which reduces words to their word root word or chops off the derivational affixes. For example, connection, connected, connecting word reduce to a common word "connect".

```
# Stemming
from nltk.stem import PorterStemmer
from nltk.tokenize import sent_tokenize, word_tokenize
ps = PorterStemmer()
stemmed_words=[]
for w in filtered_sent:
    stemmed_words.append(ps.stem(w))
print("Filtered Sentence:",filtered_sent)
print("Stemmed Sentence:",stemmed_words)
```

```
>>Filtered Sentence: ['Hello', 'Mr.', 'Smith', ',', 'today', '?']
 Stemmed Sentence: ['hello', 'mr.', 'smith', ',', 'today', '?']
```

10.Lemmatization:

Lemmatization reduces words to their base word, which is linguistically correct lemmas. It transforms root word with the use of vocabulary and morphological analysis. Lemmatization is usually more sophisticated than stemming. Stemmer works on an individual word without knowledge of the context. For example, the word "better" has "good" as its lemma. This thing will miss by stemming because it requires a dictionary look-up.

```
#Lexicon Normalization
#performing stemming and Lemmatization

from nltk.stem.wordnet import WordNetLemmatizer
lem = WordNetLemmatizer()
from nltk.stem.porter import PorterStemmer
stem = PorterStemmer()
word = "flying"
print("Lemmatized Word:",lem.lemmatize(word,"v")) print("Stemmed
Word:",stem.stem(word))
```

```
Lemmatized Word: fly
Stemmed Word: fli
```

11.Term Frequency (TF):

Suppose we have a set of English text documents and wish to rank which document is most relevant to the query, “Data Science is awesome!” A simple way to start out is by eliminating documents that do not contain all three words “Data”, “is”, “Science”, and “awesome”, but this still leaves many documents. To further distinguish them, we might count the number of times each term occurs in each document; the number of times a term occurs in a document is called its *term frequency*.

The weight of a term that occurs in a document is simply proportional to the term frequency.

Formula:

tf(t,d) = count of t in d / number of words in d

Document Frequency:

This measures the importance of document in whole set of corpus, this is very similar to TF. The only difference is that TF is frequency counter for a term t in document d, whereas DF is the count of **occurrences** of term t in the document set N. In other words, DF is the number of documents in which the word is present. We consider one occurrence if the term consists in the document at least once, we do not need to know the number of times the term is present.

df(t) = occurrence of t in documents

12.Inverse Document Frequency (IDF):

While computing TF, all terms are considered equally important. However, it is known that certain terms, such as "is", "of", and "that", may appear a lot of times but have little importance. Thus, we need to weigh down the frequent terms while scale up the rare ones, by computing IDF, an *inverse document frequency* factor is incorporated which diminishes the weight of terms that occur very frequently in the document set and increases the weight of terms that occur rarely.

IDF is the inverse of the document frequency which measures the informativeness of term t. When we calculate IDF, it will be very low for the most occurring words such as stop words (Because stop words such as "is" is present in almost all of the documents, and N/df will give a very low value to that word). This finally gives what we want, a relative weightage.

$$idf(t) = N/df$$

Now there are few other problems with the IDF, in case of a large corpus, say 100,000,000, the IDF value explodes, to avoid the effect we take the log of idf . During the query time, when a word which is not in vocab occurs, the df will be 0. As we cannot divide by 0, we smoothen the value by adding 1 to the denominator. that's the final formula:

Formula: $$idf(t) = log\ (N/\ (df + 1))$$

tf-idf now is a the right measure to evaluate how important a word is to a document in a collection or corpus. here are many different variations of TF-IDF but for now let us concentrate on the this basic version.

Formula: ***tf-idf(t, d) = tf(t, d) * log(N/(df + 1))***

13.Implementing TF-IDF in Python from Scratch:

To make TF-IDF from scratch in python, let's imagine those two sentences from different document: first sentence: "Data Science is the sexiest job of the 21st century".
second sentence: "machine learning is the key for data science".
First step we have to create the TF function to calculate total word frequency for all documents. Here are the codes below:
first as usual we should import the necessary libraries:

```
{'data', 'Science', 'job', 'sexiest', 'the', 'for', 'science',
'machine', 'of', 'is', 'learning', '21st', 'key', 'Data',
```

```
import pandas as pd
import sklearn as sk
import math
```

so, let's load our sentences and combine them together in a single set

'century'}

```
first_sentence = "Data Science is the sexiest job of the 21st
century"
second_sentence = "machine learning is the key for dat
science"#split so each word have their own stringfirst_sentenc
= first_sentence.split(" ")
second_sentence = second_sentence.split(" ")#joinhem to remove
common duplicate words
total=
set(first_sentence).union(set(second_sentence))print(total)
```

Output:

Now let’s add a way to count the words using a dictionary key-value pairing for both sentences

```
wordDictA = dict.fromkeys(total, 0)
wordDictB = dict.fromkeys(total, 0)for word in first_sentence:
    wordDictA[word]+=1

for word in second_sentence:
    wordDictB[word]+=1
```

Now we put them in a dataframe and then view the result:

```
pd.DataFrame([wordDictA, wordDictB])
```

[7]:

	21st	Data	Science	century	data	for	is	job	key	learning	machine	of	science	sexiest	the
0	1	1	1	1	0	0	1	1	0	0	0	1	0	1	2
1	0	0	0	0	1	1	1	0	1	1	1	0	1	0	1

No let's writing the TF Function:

```
def computeTF(wordDict, doc):
    tfDict = {}
    corpusCount = len(doc)
    for word, count in wordDict.items():
        tfDict[word] = count/float(corpusCount)
    return(tfDict)#running our sentences through the tf
function:tfFirst = computeTF(wordDictA, first_sentence)
tfSecond = computeTF(wordDictB,second_sentence)#Converting to
dataframe for visualizationtf = pd.DataFrame([tfFirst,
tfSecond])
```

and this is the expected output:

	21st	Data	Science	century	data	for	is	job	key	learning	machine	of	science	sexiest	the
0	0.1	0.1	0.1	0.1	0.000	0.000	0.100	0.1	0.000	0.000	0.000	0.1	0.000	0.1	0.200
1	0.0	0.0	0.0	0.0	0.125	0.125	0.125	0.0	0.125	0.125	0.125	0.0	0.125	0.0	0.125

That's all for TF formula, just I want to talk about stop words that we should eliminate them because they are the most commonly occurring words which don't give any additional value to the document vector .in-fact removing these will increase computation and space efficiency. nltk library has a method to download the stopwords, so instead of explicitly mentioning all the stopwords ourselves we can just use the nltk library and iterate over all the words and remove the stop words. There are many efficient ways to do this, but I'll just give a simple method.

those a sample of a stopwords in english language:

```
> stopwords("english")
  [1] "i"          "me"         "my"         "myself"     "we"
  [6] "our"        "ours"       "ourselves"  "you"        "your"
 [11] "yours"      "yourself"   "yourselves" "he"         "him"
 [16] "his"        "himself"    "she"        "her"        "hers"
 [21] "herself"    "it"         "its"        "itself"     "they"
 [26] "them"       "their"      "theirs"     "themselves" "what"
 [31] "which"      "who"        "whom"       "this"       "that"
 [36] "these"      "those"      "am"         "is"         "are"
 [41] "was"        "were"       "be"         "been"       "being"
 [46] "have"       "has"        "had"        "having"     "do"
```

and this is a simple code to download stop words and removing them.

```
import nltknltk.download('stopwords')from nltk.corpus
import stopwordsstop_words =
set(stopwords.words('english'))filtered_sentence = [w for w in wordDictA if not w in
stop_words]print(filtered_sentence)
```

Output:

```
['data', 'Science', 'job', 'sexiest', 'science', 'machine',
'learning', '21st', 'key', 'Data', 'century']
```

And now that we finished the TF section, we move onto the IDF part:

```
def computeIDF(docList):
    idfDict = {}
    N = len(docList)

    idfDict = dict.fromkeys(docList[0].keys(), 0)
    for word, val in idfDict.items():
        idfDict[word] = math.log10(N / (float(val) + 1))

    return(idfDict)#inputing our sentences in the log fid
idfs = computeIDF([wordDictA, wordDictB])
```

and now we implement the idf formula, let's finish with calculating the TFIDF

```
def computeTFIDF(tfBow, idfs):
    tfidf = {}
    for word, val in tfBow.items():
        tfidf[word] = val*idfs[word]
    return(tfidf)
#running our two sentences through the IDF:idfFirst=
computeTFIDF(tfFirst, idfs)
idfSecond = computeTFIDF(tfSecond, idfs)
#putting it in a dataframe
idf= pd.DataFrame([idfFirst, idfSecond])
print(idf)
```

Output:

14]:

21st	Data	Science	century	data	for	is	job	key	learning	machine	of	science	sexiest	the
0.030103	0.030103	0.030103	0.030103	0.000000	0.000000	0.030103	0.030103	0.000000	0.000000	0.000000	0.030103	0.000000	0.030103	0.060206
0.000000	0.000000	0.000000	0.000000	0.037629	0.037629	0.037629	0.000000	0.037629	0.037629	0.037629	0.000000	0.037629	0.000000	0.037629

That was a lot of work. But it is handy to know, if you are asked to code TF-IDF from scratch in the future. However, this can be done a lot simpler thanks to sklearn library. Let's look at the example from them below:

```
#first step is to import the libraryfrom
sklearn.feature_extraction.text import TfidfVectorizer
#for the sentence, make sure all words are lowercase or you will run
#into error. for simplicity, I just made the same sentence all
#lowercasefirstV= "Data Science is the sexiest job of the 21st century"
secondV= "machine learning is the key for data science"
#calling the TfidfVectorizer
vectorize= TfidfVectorizer()
#fitting the model and passing our sentences right away:
response= vectorize.fit_transform([firstV, secondV])
```

Output:

```
19]: print(response)
```

```
  (0, 1)        0.34211869506421816
  (0, 0)        0.34211869506421816
  (0, 9)        0.34211869506421816
  (0, 5)        0.34211869506421816
  (0, 11)       0.34211869506421816
  (0, 12)       0.48684053853849035
  (0, 4)        0.24342026926924518
  (0, 10)       0.24342026926924518
  (0, 2)        0.24342026926924518
  (1, 3)        0.40740123733358447
  (1, 6)        0.40740123733358447
  (1, 7)        0.40740123733358447
  (1, 8)        0.40740123733358447
  (1, 12)       0.28986933576883284
  (1, 4)        0.28986933576883284
  (1, 10)       0.28986933576883284
  (1, 2)        0.28986933576883284
```

Conclusion:

In this tutorial, you have learned what Text Analytics is, NLP and text mining, basics of text analytics operations using NLTK such as Tokenization, Normalization, Stemming, Lemmatization and POS tagging. What are sentiment analysis and text classification using scikit-learn

Questions:

1. What is Tokenization?
2. What is POS Tagging?
3. What are stop words?
4. What is Stemming and Lemmatization?

Group A			
Assignment No 8	**Data Visualization I**	**Page No**	

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 see if 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 passenger is distributed by plotting a histogram

---

Objective of the Assignment:

Students should be able to perform the data visualization operation using Python on any open source dataset

---

Prerequisite:

1. Basic of Python Programming
2. Concept of Data Visualization, Distributional Plotting Histogram

---

Contents for Theory:

1. Introduction of Seaborn Library
2. Downloading the Seaborn Library
3. Loading Dataset
4. Ploting Histogram Using Matplotlib

---

1. **Introduction of Seaborn Library :**

The Seaborn library is built on top of Matplotlib and offers many advanced data visualization capabilities.

The Seaborn library can be used to draw a variety of charts such as matrix plots, grid plots, regression plots etc., in this article we will see how the Seaborn library can be used to draw distributional and categorial plots.

2. Downloading the Seaborn Library

The seaborn library can be downloaded in a couple of ways. If you are using pip installer for Python libraries, you can execute the following command to download the library:

```
pip install seaborn
```

Alternatively, if you are using the Anaconda distribution of Python, you can use execute the following command to download the seaborn library:

```
conda install seaborn
```

3. Loading Dataset

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

Let's see what the Titanic dataset looks like. Execute the following script:

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

The script above loads the Titanic dataset and displays the first five rows of the dataset using the head function. The output looks like this:

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

The dataset contains 891 rows and 15 columns and contains information about the passengers who boarded the unfortunate Titanic ship. The original task is to predict whether or not the passenger survived depending upon different features such as their age, ticket, cabin they boarded, the class of the ticket, etc. We will use the Seaborn library to see if we can find any patterns in the data.

4. Ploting Histogram Matplotlib

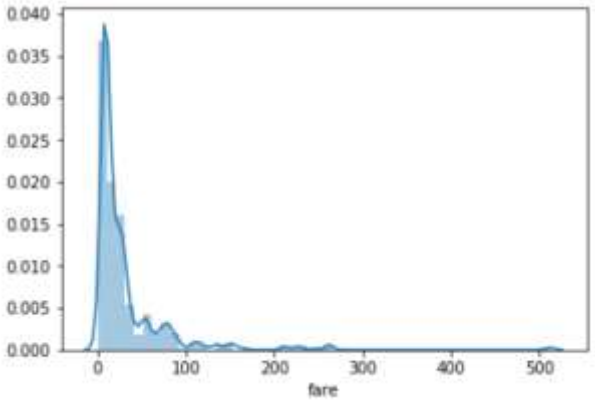
I. The Dist Plot

The distplot() shows the histogram distribution of data for a single column. The column name is passed as a parameter to the distplot() function.

Let's see how the price of the ticket for each passenger is distributed. Execute the following script:

```
sns.distplot(dataset['fare'])
```

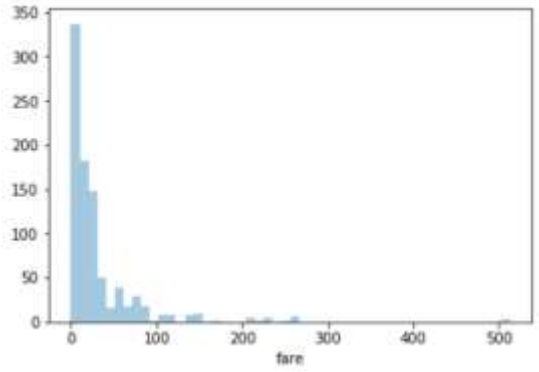
Output:



You can see that most of the tickets have been solved between 0-50 dollars. The line that you see represents the kernel density estimation. You can remove this line by passing False as the parameter for the kde attribute as shown below:

```
sns.distplot(dataset['fare'], kde=False)
```

Output:



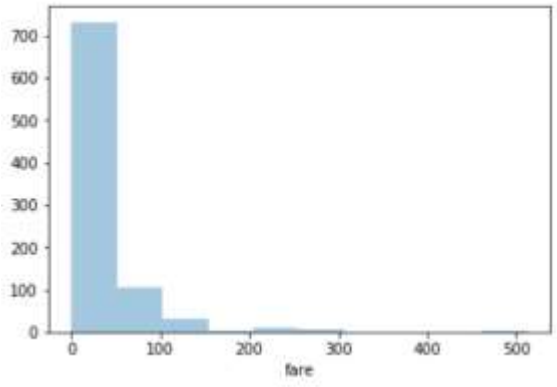
Now you can see there is no line for the kernel density estimation on the plot.

You can also pass the value for the bins parameter in order to see more or less details in the graph. Take a look at the following script:

```
sns.distplot(dataset['fare'], kde=False, bins=10)
```

Here we set the number of bins to 10. In the output, you will see data distributed in 10 bins as shown below:

Output:



You can clearly see that for more than 700 passengers, the ticket price is between 0 and 50.

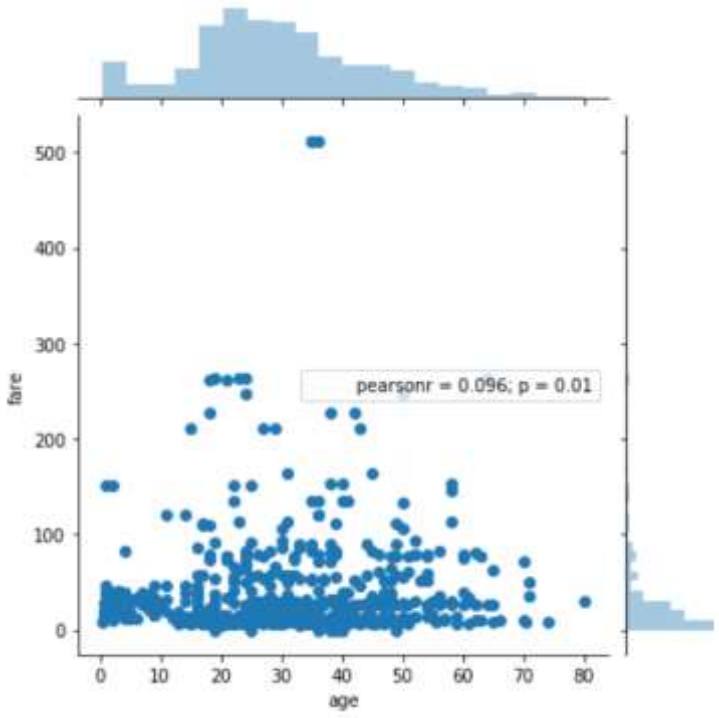
II. The Joint Plot :

The jointplot()is used to display the mutual distribution of each column. You need to pass three parameters to jointplot. The first parameter is the column name for which you want to display the distribution of data on x-axis. The second parameter is the column name for which you want to display the distribution of data on y-axis. Finally, the third parameter is the name of the data frame.

Let's plot a joint plot of age and fare columns to see if we can find any relationship between the two.

```
sns.jointplot(x='age', y='fare', data=dataset)
```

Output :



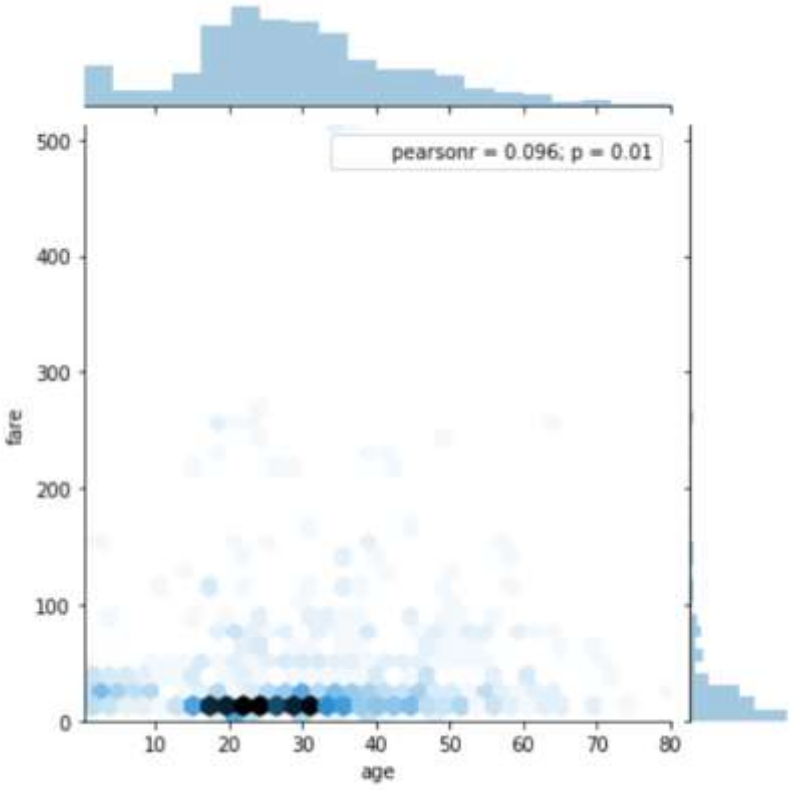
From the output, you can see that a joint plot has three parts. A distribution plot at the top for the column on the x-axis, a distribution plot on the right for the column on the y-axis and a

scatter plot in between that shows the mutual distribution of data for both the columns. You can see that there is no correlation observed between prices and the fares.

You can change the type of the joint plot by passing a value for the kind parameter. For instance, if instead of scatter plot, you want to display the distribution of data in the form of a hexagonal plot, you can pass the value hex for the kind parameter. Look at the following script:

```
sns.jointplot(x='age', y='fare', data=dataset, kind='hex')
```

Output :



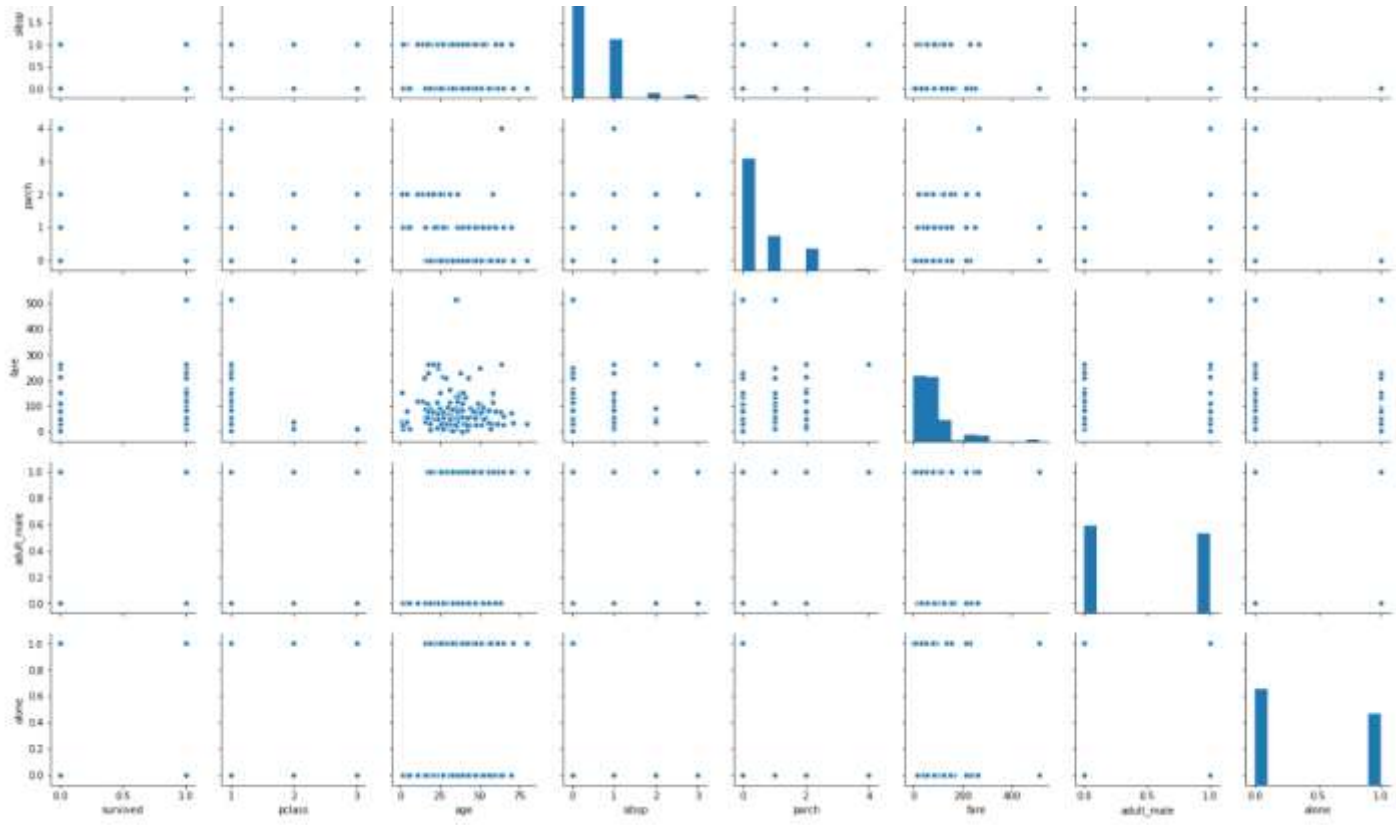
In the hexagonal plot, the hexagon with most number of points gets darker color. So if you look at the above plot, you can see that most of the passengers are between age 20 and 30 and most of them paid between 10-50 for the tickets.

III. The Pair Plot :

The paitplot() is a type of distribution plot that basically plots a joint plot for all the possible combination of numeric and Boolean columns in your dataset. You only need to pass the name of your dataset as the parameter to the pairplot() function as shown below:

```
sns.pairplot(dataset)
```

A snapshot of the portion of the output is shown below:



Note: Before executing the script above, remove all null values from the dataset using the following command:

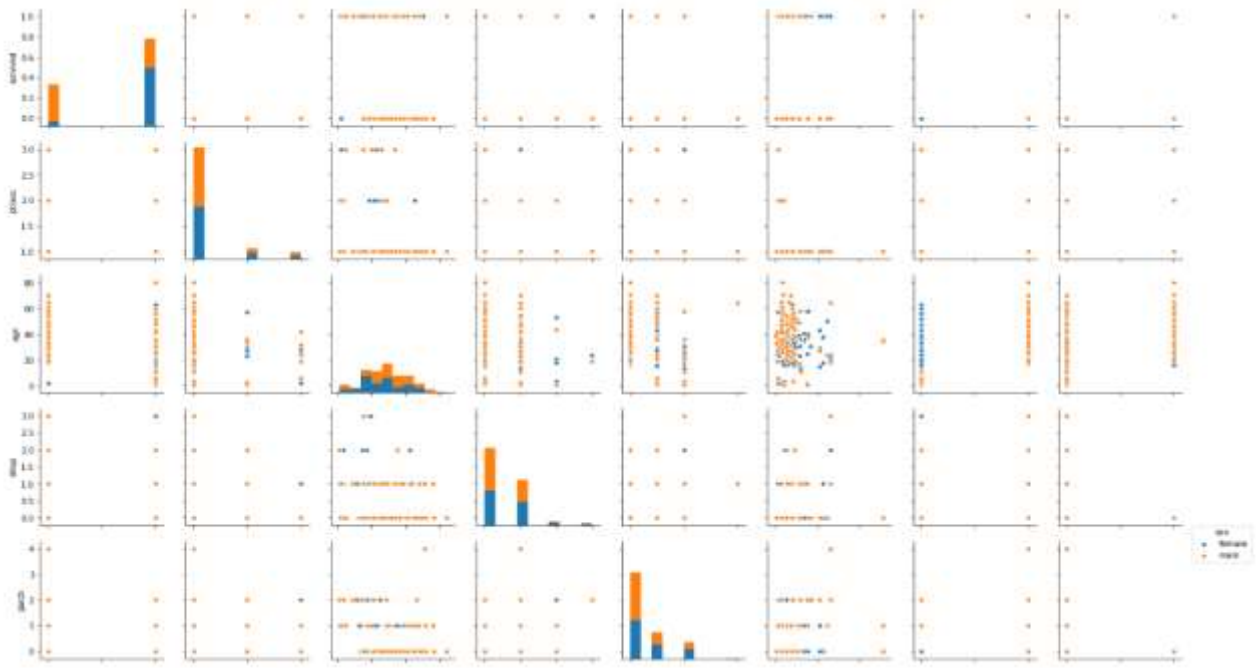
```
dataset = dataset.dropna()
```

From the output of the pair plot you can see the joint plots for all the numeric and Boolean columns in the Titanic dataset.

To add information from the categorical column to the pair plot, you can pass the name of the categorical column to the hue parameter. For instance, if we want to plot the gender information on the pair plot, we can execute the following script:

```
sns.pairplot(dataset, hue='sex')
```

Output:



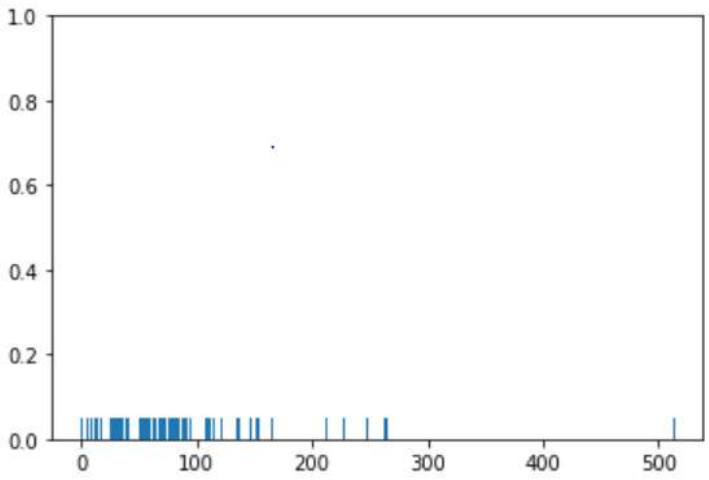
In the output you can see the information about the males in orange and the information about the female in blue (as shown in the legend). From the joint plot on the top left, you can clearly see that among the surviving passengers, the majority were female.

IV. The Rug Plot

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

```
sns.rugplot(dataset['fare'])
```

Output :



From the output, you can see that as was the case with the distplot(), most of the instances for the fares have values between 0 and 100.

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

Source Code:

```
import pandas as pandas
import numpy as numpy
import matplotlib.pyplot as pyplot
import seaborn as sns
dataset = sns.load_dataset('titanic')
dataset.head()
# The Dist Plot
sns.distplot(dataset['fare'],kde=False,bins=10)
# The Joint Plot
sns.jointplot(x='age', y='fare', data=dataset, kind='hex')
```

```
#The Rug Plot
sns.rugplot(dataset['fare'])
pyplot.show()
```

Conclusion :

Seaborn is an advanced data visualization library built on top of Matplotlib library. In this Assignment , we looked at how we can draw distributional plots using Seaborn library.

Questions :

1. What is use of join plot ?
2. What is seaborn ?
3. Explain Pair Plot ?
4. What is Dist Plot ?

Group A			
Assignmnet No 9	**Data Visualization II**	**Page No**	

Aim :- 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 they survived or not. (Column names : 'sex' and 'age')

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

Objective :

Learn to plot a box plot for distribution of age with respect to each gender along with the information about their survival.

Welcome to the world of EDA using Data Visualization. Exploratory data analysis is a way to better understand your data which helps in further Data preprocessing. And data visualization is key, making the exploratory data analysis process streamline and easily analyzing data using wonderful plots and charts.

Data Visualization

Data Visualization represents the text or numerical data in a visual format, which makes it easy to grasp the information the data express. We, humans, remember the pictures more easily than readable text, so Python provides us various libraries for data visualization like matplotlib, seaborn, plotly, etc.

Exploratory Data Analysis

Creating Hypotheses, testing various business assumptions while dealing with any Machine learning problem statement is very important and this is what EDA helps to accomplish. There are various tootle and techniques to understand your data.

We will use a very popular Titanic dataset with which everyone is familiar with and you can download it from the below link https://www.kaggle.com/c/titanic/data

Now lets us start exploring data and study visualization through boxplot with titanic data set.

Downloading the Seaborn Library

The `seaborn` library can be downloaded in a couple of ways. If you are using pip installer for Python libraries, you can execute the following command to download the library:

```
pip install seaborn
```

Alternatively, if you are using the Anaconda distribution of Python, you can use execute the following command to download the `seaborn` library:

```
conda install seaborn
```

Let's get started by importing libraries and loading Data

```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt

df = pd.read_csv('../data/titanic-train.csv')
df
```

Output :

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
0	1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.2500	NaN	S
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	PC 17599	71.2833	C85	C
2	3	1	3	Heikkinen, Miss. Laina	female	26.0	0	0	STON/O2. 3101282	7.9250	NaN	S
3	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0	1	0	113803	53.1000	C123	S
4	5	0	3	Allen, Mr. William Henry	male	35.0	0	0	373450	8.0500	NaN	S
...	...	...	...	...	...	...	...	...	...	...	...	...
886	887	0	2	Montvila, Rev. Juozas	male	27.0	0	0	211536	13.0000	NaN	S
887	888	1	1	Graham, Miss. Margaret Edith	female	19.0	0	0	112053	30.0000	B42	S
888	889	0	3	Johnston, Miss. Catherine Helen "Carrie"	female	NaN	1	2	W./C. 6607	23.4500	NaN	S
889	890	1	1	Behr, Mr. Karl Howell	male	26.0	0	0	111369	30.0000	C148	C
890	891	0	3	Dooley, Mr. Patrick	male	32.0	0	0	370376	7.7500	NaN	Q

891 rows × 12 columns

Above is the training dataset of the titanic survival problem. It has 891 rows (number of passengers), and 12 columns (data about the passenger) including the target variable "*Survived*".

Let us first look at the columns of the data and then we use *describe()* and *info()* methods to get a basic idea of what we have in hand.

```
cols = df.columns
cols
```

```
Index(['PassengerId', 'Survived', 'Pclass', 'Name', 'Sex', 'Age', 'SibSp',
       'Parch', 'Ticket', 'Fare', 'Cabin', 'Embarked'],
      dtype='object')
```

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 891 entries, 0 to 890
Data columns (total 12 columns):
 #   Column       Non-Null Count  Dtype  
---  ------       --------------  -----  
 0   PassengerId  891 non-null    int64  
 1   Survived     891 non-null    int64  
 2   Pclass       891 non-null    int64  
 3   Name         891 non-null    object 
 4   Sex          891 non-null    object 
 5   Age          714 non-null    float64
 6   SibSp        891 non-null    int64  
 7   Parch        891 non-null    int64  
 8   Ticket       891 non-null    object 
 9   Fare         891 non-null    float64
 10  Cabin        204 non-null    object 
 11  Embarked     889 non-null    object 
dtypes: float64(2), int64(5), object(5)
memory usage: 83.7+ KB
```

```
df.describe()
```

	PassengerId	Survived	Pclass	Age	SibSp	Parch	Fare
count	891.000000	891.000000	891.000000	714.000000	891.000000	891.000000	891.000000
mean	446.000000	0.383838	2.308642	29.699118	0.523008	0.381594	32.204208
std	257.353842	0.486592	0.836071	14.526497	1.102743	0.806057	49.693429
min	1.000000	0.000000	1.000000	0.420000	0.000000	0.000000	0.000000
25%	223.500000	0.000000	2.000000	20.125000	0.000000	0.000000	7.910400
50%	446.000000	0.000000	3.000000	28.000000	0.000000	0.000000	14.454200
75%	668.500000	1.000000	3.000000	38.000000	1.000000	0.000000	31.000000
max	891.000000	1.000000	3.000000	80.000000	8.000000	6.000000	512.329200

```
df.isnull().sum()
```

```
PassengerId      0
Survived         0
Pclass           0
Name             0
Sex              0
Age            177
SibSp            0
Parch            0
Ticket           0
Fare             0
Cabin          687
Embarked         2
dtype: int64
```

We already know that there are 177 missing values in the age column. From the above results, we see that there are 687 missing values in 'Cabin' and 2 missing values in 'Embarked'. We need to fix these null values before we move on to modeling.

Types of analysis

A) Univariate Analysis
B) Bivariate/ Multivariate Analysis

Univariate Analysis

Univariate analysis is the simplest form of analysis where we explore a single variable. Univariate analysis is performed to describe the data in a better way. we perform Univariate analysis of Numerical and categorical variables differently because plotting uses different plots.

Bivariate/ Multivariate Analysis

Bivariate Analysis is used when we have to explore the relationship between 2 different variables and we have to do this because, in the end, our main task is to explore the relationship between variables to build a powerful model. And when we analyze more than 2 variables together then it is known as Multivariate Analysis.

Let's explore the titanic dataset where one variable is numerical and other is categorical visualized through boxplot.

Given Variables :

Age & Gender – Where age is numerical & Gender is categorical

Understanding Boxplots

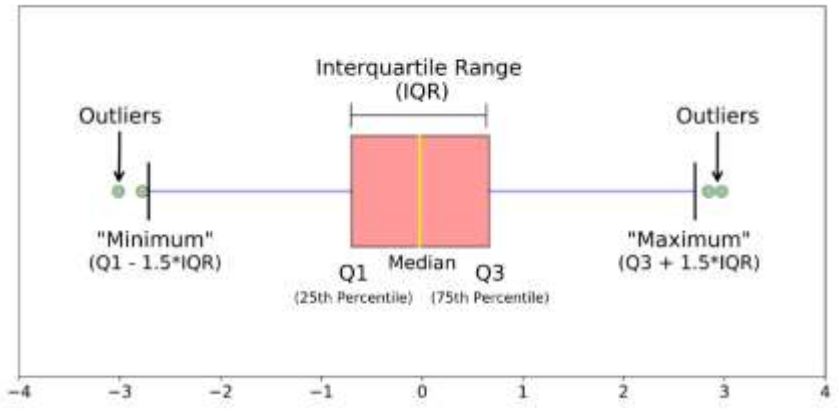


Fig. Different parts of a boxplot

The image above is a boxplot**.** A boxplot is a standardized way of displaying the distribution of data based on a five number summary ("minimum", first quartile (Q1), median, third quartile (Q3), and "maximum"). It can tell you about your outliers and what their values are. It can also tell you if your data is symmetrical, how tightly your data is grouped, and if and how your data is skewed.

Boxplots

A boxplot is a graph that gives you a good indication of how the values in the data are spread out. Although boxplots may seem primitive in comparison to a histogram or density plot, they have the advantage of taking up less space, which is useful when comparing distributions between many groups or datasets.

median (Q2/50th Percentile): the middle value of the dataset.

first quartile (Q1/25th Percentile): the middle number between the smallest number (not the "minimum") and the median of the dataset.

third quartile (Q3/75th Percentile): the middle value between the median and the highest value (not the "maximum") of the dataset.

interquartile range (IQR): 25th to the 75th percentile.

outliers (shown as green circles)

"maximum": Q3 + 1.5*IQR

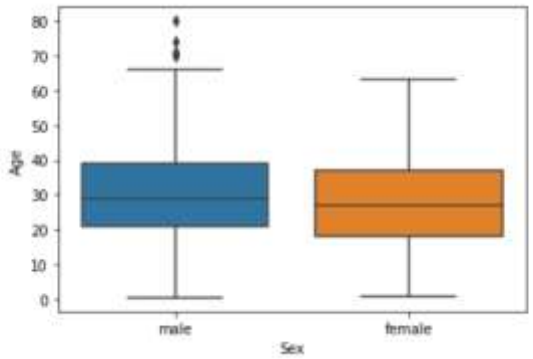
"minimum": Q1 -1.5*IQR

We can draw a separate boxplot for both the variable. let us explore gender with age using a boxplot.

Code :

```
sns.boxplot(data['Sex'],data['Age'])
```

Output :

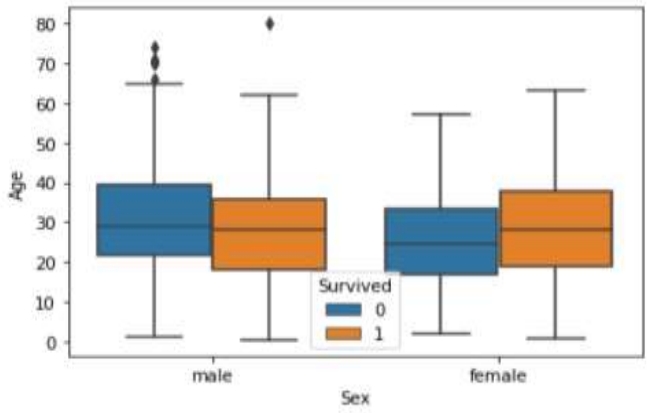


Along with age and gender let's see who has survived and who has not.

Code:

```
sns.boxplot(data['Sex'],data['Age'],data['Survived'])
plt.show()
```

Output:



Seaborn Library:

- Seaborn provides a variety of visualization patterns. It uses fewer syntax and has easily interesting default themes.
- It specializes in statistics visualization and is used if one has to summarize data in visualizations and also show the distribution in the data.
- Seaborn is more integrated for working with Pandas data frames.
- Seaborn works with the dataset as a whole and is much more intuitive than Matplotlib.

Conclusion:

Hence we visualized the data using box plot.

Questions:

1. Whate are different types of analysis in EDA ?
2. Define Univariate & Bivariate Analysis?
3. Define Exploratory Data Analysis?
4. Define Data Visualization?

Group A			
Assignment No 10	**Data Visualization III**	**Page No**	

Aim : 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 :

Studying the Iris Dataset and listing down the inferences.

What is Data Analysis?

Data Analysis is the process of systematically applying statistical and/or logical techniques to describe and illustrate, condense and recap, and evaluate data.

Load libraries & data

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

```
df = pd.read_csv("iris-flower-dataset.csv",header=None)
df.columns = ["col1","col2","col3","col4","col5"]
```

```
df.head()
```

	col1	col2	col3	col4	col5
0	5.1	3.5	1.4	0.2	Iris-setosa
1	4.9	3.0	1.4	0.2	Iris-setosa
2	4.7	3.2	1.3	0.2	Iris-setosa
3	4.6	3.1	1.5	0.2	Iris-setosa
4	5.0	3.6	1.4	0.2	Iris-setosa

What are features ?

Features are nothing but the independent variables in machine learning models. What is required to be learned in any specific machine learning problem is a set of these features (independent variables), coefficients of these features, and parameters for coming up with appropriate functions or models (also termed as hyperparameters).

Features can be in the form of raw data that is very straightforward and can be derived from real-life as it is. However, not all problems can be solved using raw data or data in its original form. Many times, they need to be represented or encoded in different forms.

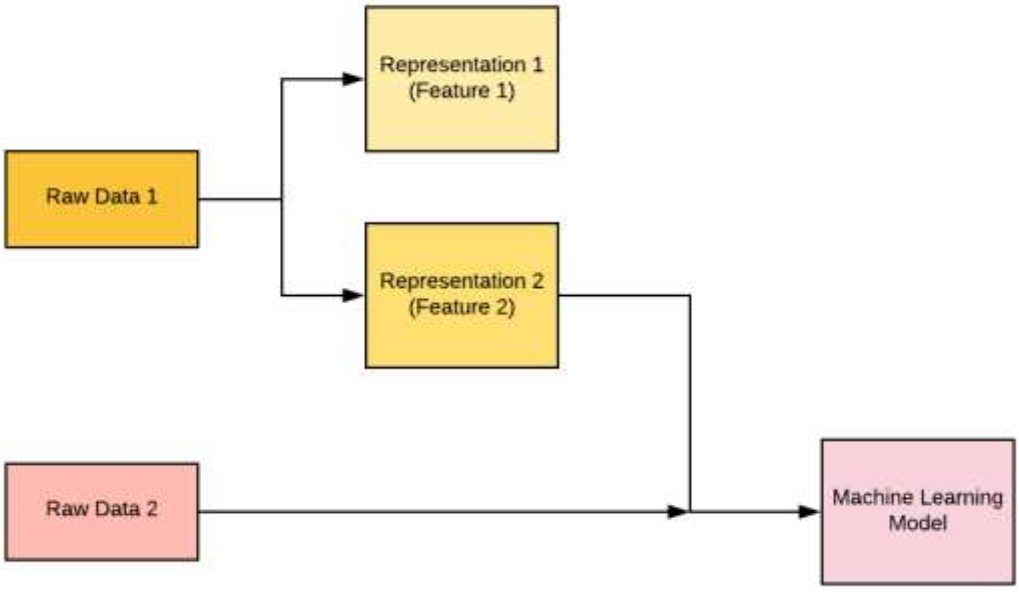


Fig. Features – Key to Machine Learning

1- List down the features and their types (e.g., numeric, nominal) available in the dataset.

```
column = len(list(df))
column
```

```
5
```

Dataset has 5 columns indicating 5 features about the data.

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 150 entries, 0 to 149
Data columns (total 5 columns):
 #   Column  Non-Null Count  Dtype
---  ------  --------------  -----
 0   col1    150 non-null    float64
 1   col2    150 non-null    float64
 2   col3    150 non-null    float64
 3   col4    150 non-null    float64
 4   col5    150 non-null    object
dtypes: float64(4), object(1)
memory usage: 6.0+ KB
```

Dataset contains 4 numerical columns & 1 object column.

```
np.unique(df["col5"])
```

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

2- Create a histogram for each feature in the dataset to illustrate the feature distributions.

Histogram :

Histograms group the data in bins and is the fastest way to get idea about the distribution of each attribute in dataset. The following are some of the characteristics of histograms –

- It provides us a count of the number of observations in each bin created for visualization.
- From the shape of the bin, we can easily observe the distribution i.e. weather it is Gaussian, skewed or exponential.
- Histograms also help us to see possible outliers.

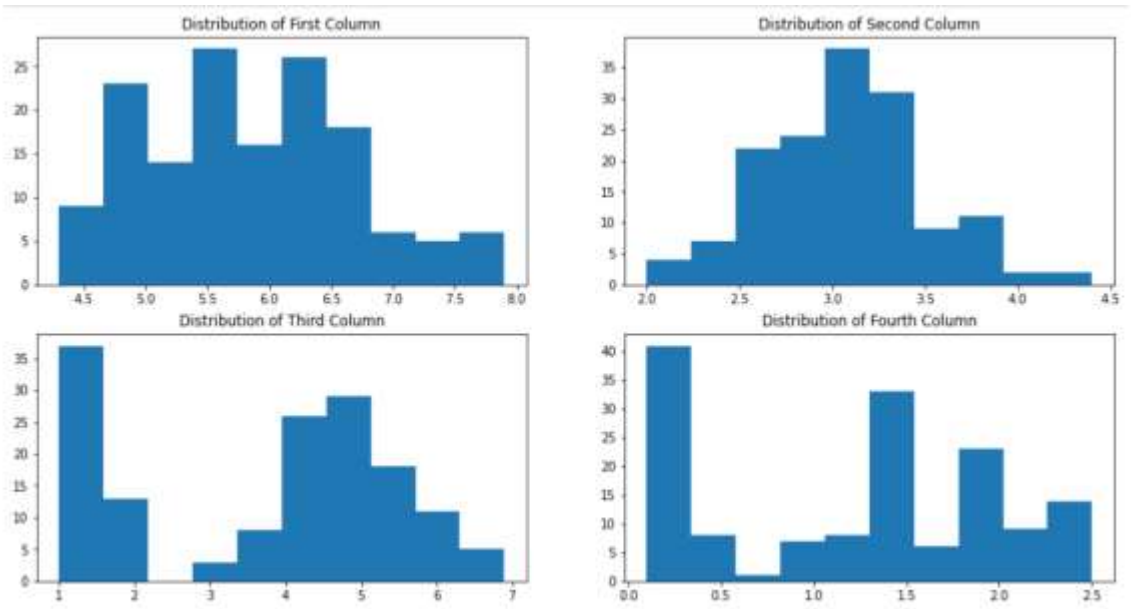
Import visualization libraries seaborn , matplotlib.

```
import seaborn as sns
import matplotlib
import matplotlib.pyplot as plt
%matplotlib inline
```

Code :

```
fig, axes = plt.subplots(2, 2, figsize=(16, 8))


axes[0,0].set_title("Distribution of First Column")
axes[0,0].hist(df["col1"]);

axes[0,1].set_title("Distribution of Second Column")
axes[0,1].hist(df["col2"]);

axes[1,0].set_title("Distribution of Third Column")
axes[1,0].hist(df["col3"]);

axes[1,1].set_title("Distribution of Fourth Column")
axes[1,1].hist(df["col4"]);
```

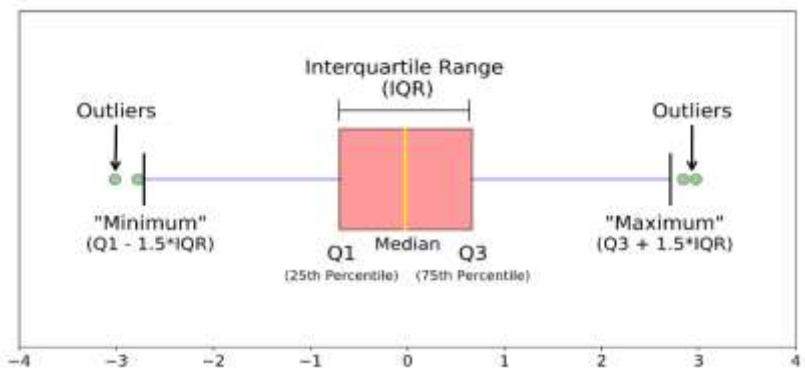
Output :



3. Create a box plot for each feature in the dataset.

What is Boxplot?

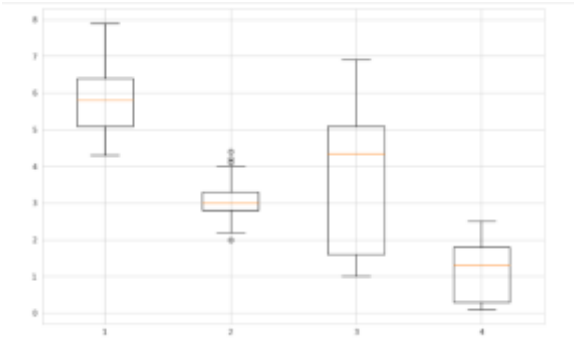
A boxplot is a standardized way of displaying the distribution of data based on a five number summary ("minimum", first quartile (Q1), median, third quartile (Q3), and "maximum"). It can tell you about your outliers and what their values are.



Code :

```
data_to_plot = [df["col1"],df["col2"],df["col3"],df["col4"]]

sns.set_style("whitegrid")
# Creating a figure instance
fig = plt.figure(1, figsize=(12,8))

# Creating an axes instance
ax = fig.add_subplot(111)

# Creating the boxplot
bp = ax.boxplot(data_to_plot);
```

Output :



4. Compare distributions and identify outliers.

What is Distribution ?

A distribution is simply a collection of data, or scores, on a variable. Usually, these scores are arranged in order from smallest to largest and then they can be presented graphically.

Why are Distribution important in Machine Learning?

In Machine Learning, data satisfying Normal Distribution is beneficial for model building. It makes math easier. Models like LDA, Gaussian Naive Bayes, Logistic Regression, Linear Regression, etc., are explicitly calculated from the assumption that the distribution is a bivariate or multivariate normal.

Summary statistics for each feature available in the dataset.

```
df.describe()
```

	col1	col2	col3	col4
count	150.000000	150.000000	150.000000	150.000000
mean	5.843333	3.054000	3.758667	1.198667
std	0.828066	0.433594	1.764420	0.763161
min	4.300000	2.000000	1.000000	0.100000
25%	5.100000	2.800000	1.600000	0.300000
50%	5.800000	3.000000	4.350000	1.300000
75%	6.400000	3.300000	5.100000	1.800000
max	7.900000	4.400000	6.900000	2.500000

What is Outlier ?

An outlier is a data point that is noticeably different from the rest. They represent errors in measurement, bad data collection, or simply show variables not considered when collecting the data.

For identifying outliers.

First of all Box plot is a data visualization plotting function. It shows the min, max, median, first quartile, and third quartile. All of the things will be explained briefly. All of the property of box plot can be accessed by *dataframe.column_name.describe()* function.

Explanation of the different parts of the box plot.

The maximum and the minimum is the max and min value of the data-set. 50 percentile is the median of the data-set. The first quartile is the median of the data between the min to 50% and the third quartile is the median of the data between 50% to max. The outliers will be the values that are out of the (1.5*interquartile range) from the 25 or 75 percentile.

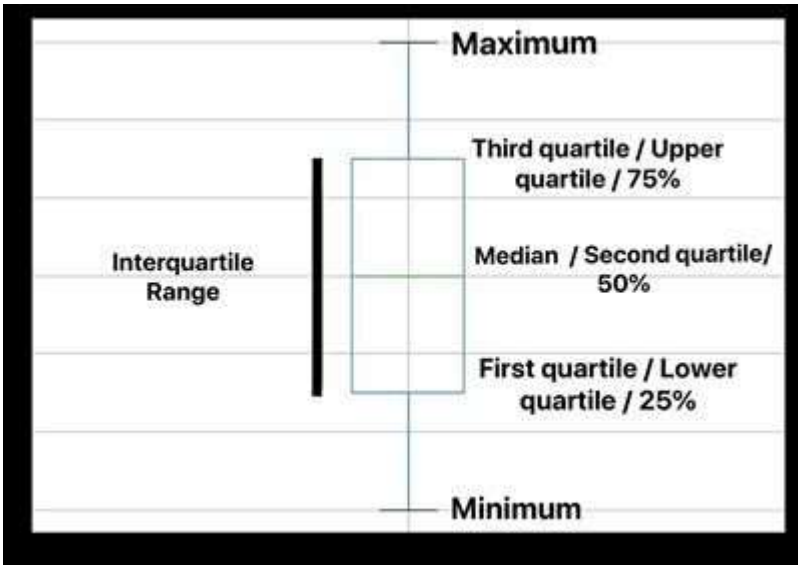


Fig. Features of Boxplot

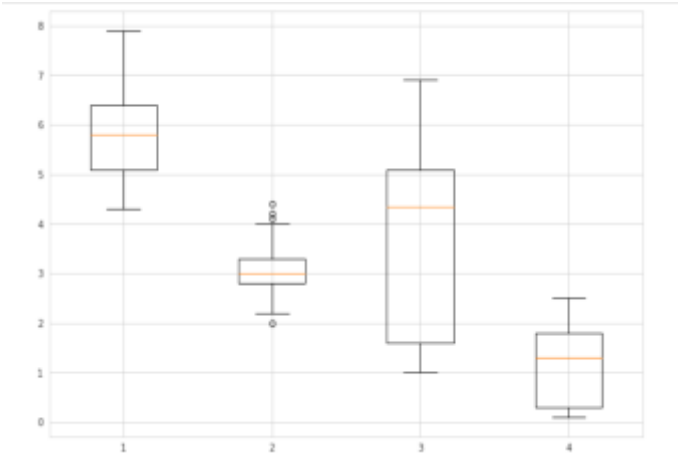
Methods of finding the values

- Use the median to divide the ordered data set into two halves.
  1) If there is an odd number of data points in the original ordered data set, do not include the median (the central value in the ordered list) in either half.
  2) If there is an even number of data points in the original ordered data set, split this data set exactly in half.
- The lower quartile value is the median of the lower half of the data. The upper quartile value is the median of the upper half of the data.

- An extreme value is considered to be an outlier if it is at least 1.5 interquartile ranges below the first quartile, or at least 1.5 interquartile ranges above the third quartile.

The box plot seem useful to detect outliers but it has several other uses too. Box plots take up less space and are therefore particularly useful for comparing distributions between several groups or sets of data. It is a direct representation of the Probability Density Function which indicates the distribution of data.

Box plot of Iris data set



If we observe closely. for the box 2, interquartile distance is roughly around 0.75 hence the values lying beyond this range of (third quartile + interquartile distance) i.e. roughly around 4.05 will be considered as outliers. Similarly outliers with other boxplots can be found.

Conclusion :

We studied the iris dataset & plotted box plot for identifying outliers.

Questions :

1- What is Data Analysis?
2- What are outliers?
3- Explain different parts of boxplot?
4- What are different types of distribution?

Group B			
Assignment No 11	**Big Data Analytics – JAVA/SCALA**	**Page No**	

Aim -

Write a code in JAVA for a simple WordCount application that counts the number of occurrences of each word in a given input set using the Hadoop MapReduce framework on local-standalone set-up.

Objective -

---

1. To understand Installation & Configuration of Hadoop on single Node.

2. To understand the concept of Map Reduce.

3. To understand the details of Hadoop File system.

Hadoop 2.0: Single Node installation

prerequisite for hadoop -

Installation of java(1.8)-JDK

copy jdk-8u321-linux-x64.tar.gz to home

right click and click extract here

In terminal paste the below commands

$ sudo cp -R jdk1.8.0_321 /usr/lib/jvm

- Go to this /usr/lib/jvm folder using following command

$ cd /usr/lib/jvm

- Edit the environment file.

$ sudo gedit /etc/environment

Paste below text in gedit by deleting existing path

```
PATH="/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/usr/games:/usr/local/games:/usr/lib/jvm/jdk1.8.0_321/bin:/usr/lib/jvm/jdk1.8.0_321/db/bin:/usr/lib/jvm/jdk1.8.0_321/jre/bin"

J2SDKDIR="/usr/lib/jvm/jdk1.8.0_321"

J2REDIR="/usr/lib/jvm/jdk1.8.0_321/jre"

JAVA_HOME="/usr/lib/jvm/jdk1.8.0_321"

DERBY_HOME="/usr/lib/jvm/jdk1.8.0_321/db"

HADOOP_CLASSPATH=/usr/lib/jvm/jdk1.8.0_321/lib/tools.jar
```

Save and close geddit

- Use update-alternatives to inform Ubuntu about the installed java paths.

Execute following commands one by one in terminal

```
$ sudo update-alternatives --install "/usr/bin/java" "java" "/usr/lib/jvm/jdk1.8.0_321/java" 0
$ sudo update-alternatives --install "/usr/bin/javac" "javac" "/usr/lib/jvm/jdk1.8.0_321/bin/javac" 0
$ sudo update-alternatives --set java /usr/lib/jvm/jdk1.8.0_321/bin/java
$ sudo update-alternatives --set javac /usr/lib/jvm/jdk1.8.0_321/bin/javac
```

- Give the location of java and javac as you provided.

```
$ update-alternatives --list java
$ update-alternatives --list javac
```

- Verify the Java version

```
$ java -version
```

The output should resemble the following:

```
java version "1.8.0_321"
Java(TM) SE Runtime Environment (build 1.8.0_321-b07)
Java HotSpot(TM) 64-Bit Server VM (build 25.321-b07.mixed mode)
```

Hadoop Installation -

- Extract hadoop-2.7.3.tar.gz in home
- Execute below commands one by one

1. $ sudo apt-get install openssh-server
2. $ java -version
3. $ ssh localhost
4. $ ssh-keygen

Keep pressing enter till you execute the command

5. $ cat ~/.ssh/id_rsa.pub >> ~/.ssh/authorized_keys
6. $ ssh localhost

after this press ctrl+D twice

$ sudo gedit $HOME/.bashrc

Copy below text in gedit at the end of file (replace cg3 with your pc no)

```
#Set HADOOP_HOME
export HADOOP_HOME=/home/cg3/hadoop-2.7.3
#Set JAVA_HOME
export JAVA_HOME=/usr/lib/jvm/jdk1.8.0_321
# Add bin/ directory of Hadoop to PATH
export PATH=$PATH:$HADOOP_HOME/bin
export HADOOP_CLASSPATH=/usr/lib/jvm/jdk1.8.0_321/lib/tools.jar
```

Save & close gedit

$ source ~/.bashrc

- Go to folder = hadoop-2.7.3/etc/hadoop

1. Open hadoop-env.sh and & change JAVA_HOME path

export JAVA_HOME=/usr/lib/jvm/jdk1.8.0_321

$ echo $JAVA_HOME

$ echo $HADOOP_HOME

Go to folder = hadoop-2.7.3/etc/hadoop

2. Open core-site.xml with gedit and paste below commands under configuration tag

```
<property>
<name>fs.defaultFS</name>
<value>hdfs://localhost:9000</value>
</property>
```

3. Open hdfs-site.xml with gedit and paste below commands under configuration tag

```
<property>
   <name>dfs.replication</name>
   <value>1</value>
</property>
```

4. Copy mapred-site.xml.template and paste in same location and change name to mapred-site.xml

   Open mapred-site.xml with gedit and paste below commands under configuration tag

```
<property>
   <name>mapreduce.framework.name</name>
   <value>yarn</value>
</property>
```

5. Open yarn-site.xml with gedit and paste below commands under configuration tag

```
<property>
   <name>yarn.nodemanager.aux-services</name>
   <value>mapreduce_shuffle</value>
</property>
```

$ cd hadoop-2.7.3

Execute below commands one by one

1. bin/hdfs namenode -format
2. sbin/start-dfs.sh
3. sbin/start-yarn.sh
4. jps

You can see hadoop web ui at http://localhost:50070/

*****

$ cd /home/cg10/hadoop-2.7.3/

$ bin/hdfs dfs -mkdir /input

$ bin/hdfs dfs -put /home/cg10/data.txt /input

$ bin/hdfs dfs -ls /input

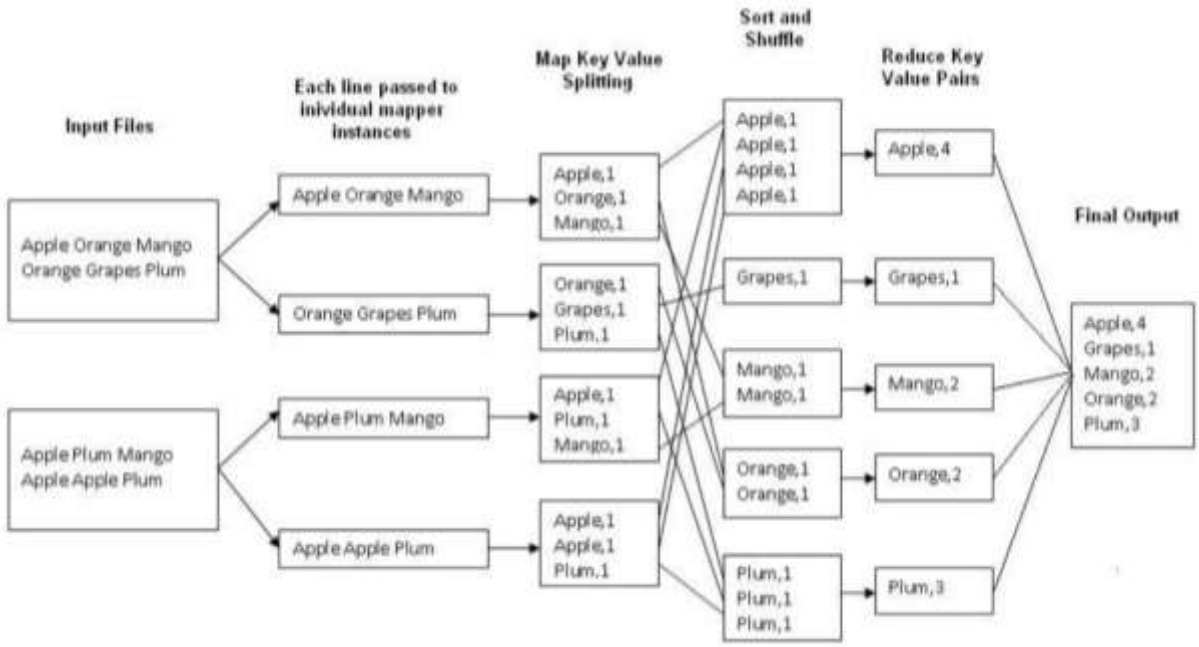
$ bin/hdfs dfs -cat /input/data.txt

$ bin/hdfs dfs -get /input/data.txt /home/cg10/myoutput

Word Count Program

Program using Hadoop that counts number of occurrences of words in a file.

The Word Count Flow -



The dataset: fruits.txt

```
1 apple orange mango
2 apple plum grapes
3 pineapple raspberry banana
4 banana orange apple
5 plum raspberry pomgranate
6 grapes grapes apple mango
```

Output:

```
apple    4
banana   2
grapes   3
mango    2
orange   2
pineapple        1
plum     2
pomgranate       1
raspberry        2
```

```
import java.io.IOException;
import java.util.StringTokenizer;

import org.apache.hadoop.conf.Configuration;
import org.apache.hadoop.fs.Path;
import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Job;
import org.apache.hadoop.mapreduce.Mapper;
import org.apache.hadoop.mapreduce.Reducer;
import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;
import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;

public class WordCount {
  public static class TokenizerMapper
       extends Mapper<Object, Text, Text, IntWritable>{
    private final static IntWritable one = new IntWritable(1);
    private Text word = new Text();
    public void map(Object key, Text value, Context context
                    ) throws IOException, InterruptedException {
      StringTokenizer itr = new StringTokenizer(value.toString());

while (itr.hasMoreTokens()) {
        word.set(itr.nextToken());
        context.write(word, one);
      }
    }
  }


  public static class IntSumReducer
       extends Reducer<Text,IntWritable,Text,IntWritable> {
    private IntWritable result = new IntWritable();
    public void reduce(Text key, Iterable<IntWritable> values,
                       Context context
                       ) throws IOException, InterruptedException {
      int sum = 0;
      for (IntWritable val : values) {
        sum += val.get();
      }
      result.set(sum);
      context.write(key, result);
    }
```

```
  }
  public static void main(String[] args) throws Exception {
    Configuration conf = new Configuration();
    Job job = Job.getInstance(conf, "word count");
    job.setJarByClass(WordCount.class);
    job.setMapperClass(TokenizerMapper.class);
    job.setCombinerClass(IntSumReducer.class);
    job.setReducerClass(IntSumReducer.class);

    job.setReducerClass(IntSumReducer.class);
    job.setOutputKeyClass(Text.class);
    job.setOutputValueClass(IntWritable.class);
    FileInputFormat.addInputPath(job, new Path(args[0]));
    FileOutputFormat.setOutputPath(job, new Path(args[1]));
    System.exit(job.waitForCompletion(true) ? 0 : 1);
  }
}
```

Create WordCount.java in hadoop-2.7.3 folder and paste below code in it and save.

Compile WordCount.java and create a jar:

$ bin/hadoop com.sun.tools.javac.Main WordCount.java

$ jar cf wc.jar WordCount*.class

create data.txt file in home.

Insert some text with repetitive words in file so that we can count words.

$ bin/hdfs dfs -put /home/cg10/data.txt /input

Run the application:

$ bin/hadoop jar wc.jar WordCount /home/cg10/input /home/cg10/output/

$ bin/hdfs dfs -get /home/cg10/output/part-r-00000 /home/cg10/

Check word count of your input (data.txt) in home directory with name part-r-00000

Question :

1. Explain Hadoop Mapreduce Framework
2. Requirement of Hadoop Installation

Group B			
Assignment No 12	**Big Data Analytics – JAVA/SCALA**	**Page No**	

AIM: Design a distributed application using MapReduce which processes a log file of a system.

OBJECTIVE:

- To understand the concept of Map Reduce.
- To understand the technique for log file processing
- To understand use of distributed processing

THEORY:

What is MapReduce?

MapReduce is a processing technique and a program model for distributed computing based on java. The MapReduce algorithm contains two important tasks, namely Map and Reduce. Map takes a set of data and converts it into another set of data, where individual elements are broken down into tuples (key/value pairs). Secondly, reduce task, which takes the output from a map as an input and combines those data tuples into a smaller set of tuples. As the sequence of the name MapReduce implies, the reduce task is always performed after the map job.

The major advantage of MapReduce is that it is easy to scale data processing over multiple computing nodes. Under the MapReduce model, the data processing primitives are called mappers and reducers. Decomposing a data processing application into mappers and reducers

is sometimes nontrivial. But, once we write an application in the MapReduce form, scaling the application to run over hundreds, thousands, or even tens of thousands of machines in a cluster is merely a configuration change. This simple scalability is what has attracted many programmers to use the MapReduce model.

The Algorithm :

- Generally MapReduce paradigm is based on sending the computer to where the data resides!
- MapReduce program executes in three stages, namely map stage, shuffle stage, and reduce stage.
    - **Map stage** : The map or mapper's job is to process the input data. Generally the input data is in the form of file or directory and is stored in the Hadoop file system (HDFS). The input file is passed to the mapper function line by line. The mapper processes the data and creates several small chunks of data.
    - **Reduce stage** : This stage is the combination of the **Shuffle** stage and the **Reduce** stage. The Reducer's job is to process the data that comes from the mapper. After processing, it produces a new set of output, which will be stored in the HDFS.
- During a MapReduce job, Hadoop sends the Map and Reduce tasks to the appropriate servers in the cluster.
- The framework manages all the details of data-passing such as issuing tasks, verifying task completion, and copying data around the cluster between the nodes.
- Most of the computing takes place on nodes with data on local disks that reduces the network traffic.

- After completion of the given tasks, the cluster collects and reduces the data to form an appropriate result, and sends it back to the Hadoop server.

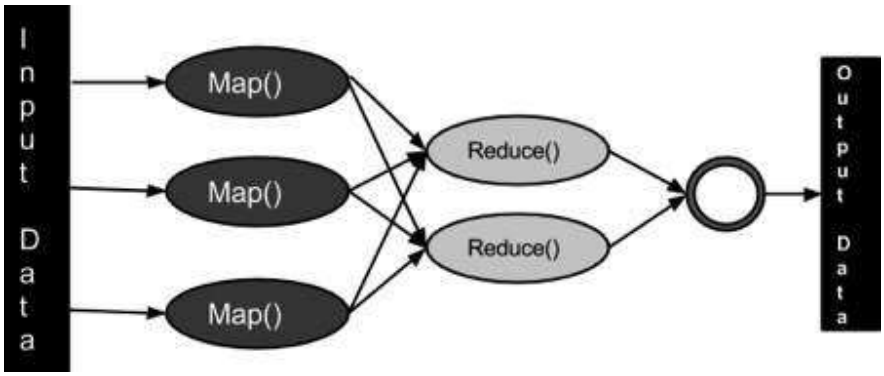


Fig.1. Map Reduce job processing

Inputs and Outputs (Java Perspective)

The MapReduce framework operates on <key, value> pairs, that is, the framework views the input to the job as a set of <key, value> pairs and produces a set of <key, value> pairs as the output of the job, conceivably of different types.

The key and the value classes should be in serialized manner by the framework and hence, need to implement the Writable interface. Additionally, the key classes have to implement the Writable-Comparable interface to facilitate sorting by the framework. Input and Output types of a MapReduce job: (Input) <k1, v1> -> map -><k2, v2>-> reduce -><k3, v3>(Output).

	Input	**Output**
Map	<k1, v1>	list (<k2, v2>)

Reduce	<k2, list(v2)>	list (<k3, v3>)

Terminology

- **PayLoad** - Applications implement the Map and the Reduce functions, and form the core of the job.
- **Mapper** - Mapper maps the input key/value pairs to a set of intermediate key/value pair.
- **NamedNode** - Node that manages the Hadoop Distributed File System (HDFS).
- **DataNode** - Node where data is presented in advance before any processing takes place.
- **MasterNode** - Node where JobTracker runs and which accepts job requests from clients.
- **SlaveNode** - Node where Map and Reduce program runs.
- **JobTracker** - Schedules jobs and tracks the assign jobs to Task tracker.
- **Task Tracker** - Tracks the task and reports status to JobTracker.
- **Job** - A program is an execution of a Mapper and Reducer across a dataset.
- **Task** - An execution of a Mapper or a Reducer on a slice of data.
- **Task Attempt** - A particular instance of an attempt to execute a task on a SlaveNode.

Input Data

The above data is saved as **sample.txt** and given as input. The input file looks as shown below.

```
1979232324324252626262625252 5

1980262728282830313131303030 29

1981313232323334353636343434 34

1984393839393941424340393838 40

1985383939393941414100403939 45
```

Example Program :

Given below is the program to the sample data using MapReduce framework.

```
package hadoop;

import java.util.*;

import java.io.IOException;
import java.io.IOException;
import org.apache.hadoop.fs.Path;

import org.apache.hadoop.conf.*;

import org.apache.hadoop.io.*;

import org.apache.hadoop.mapred.*;

import org.apache.hadoop.util.*;



public class ProcessUnits
```

```
{

//Mapper class

publicstaticclass E_EMapper extends MapReduceBase implements

Mapper<LongWritable,/*Input key Type */

Text,/*Input value Type*/

Text,/*Output key Type*/

IntWritable>/*Output value Type*/

{

//Map function  publicvoid map(LongWritable key,Text value, {

OutputCollector<Text,IntWritable> output,

Reporter reporter)throwsIOException

{

String line =value.toString();

String lasttoken =null;

StringTokenizer s =newStringTokenizer(line,"\t");

String year =s.nextToken();

while(s.hasMoreTokens())

{
lasttoken=s.nextToken();

}
```

```
int avgprice =Integer.parseInt(lasttoken);
output.collect(newText(year),newIntWritable(avgprice));
}

}

//Reducer class  publicstaticclass E_EReduce
extendsMapReduceBaseimplements
Reducer<Text,IntWritable,Text,IntWritable>
{

//Reduce function  publicvoid reduce(Text
key,Iterator<IntWritable> values,
OutputCollector<Text,IntWritable> output,Reporter reporter)throwsIOException

{ int maxavg=30; int val=Integer.MIN_VALUE;


while(values.hasNext())

{ if((val=values.next().get())>maxavg)
{ output.collect(key,newIntWritable(val));
}

}

}

}
```

```
//Main function  publicstaticvoid main(String args[])throwsException {
JobConf conf =newJobConf(ProcessUnits.class);



conf.setJobName("max_eletricityunits");
conf.setOutputKeyClass(Text.class);
conf.setOutputValueClass(IntWritable.class);
conf.setMapperClass(E_EMapper.class);
conf.setCombinerClass(E_EReduce.class);
conf.setReducerClass(E_EReduce.class);
conf.setInputFormat(TextInputFormat.class);
conf.setOutputFormat(TextOutputFormat.class);

FileInputFormat.setInputPaths(conf,newPath(args[0]));
FileOutputFormat.setOutputPath(conf,newPath(args[1]));

JobClient.runJob(conf);
}
}
```

Save the above program as **ProcessUnits.java.** The compilation and execution of the program is explained below.

Compilation and Execution of Process Units Program

Let us assume we are in the home directory of a Hadoop user (e.g. /home/hadoop).

Follow the steps given below to compile and execute the above program.

Step 1

The following command is to create a directory to store the compiled java classes.

```
$ mkdir units
```

Step 2

Download **Hadoop-core-1.2.1.jar,** which is used to compile and execute the MapReduce program. Visit the following link http://mvnrepository.com/artifact/org.apache.hadoop/hadoopcore/1.2.1to download the jar. Let us assume the downloaded folder is **/home/hadoop/.**

Step 3

The following commands are used for compiling the **ProcessUnits.java**program and creating a jar for the program.

```
$ javac-classpath hadoop-core-1.2.1.jar-d units
ProcessUnits.java  $ jar -cvf units.jar -C units/.
```

Step 4

The following command is used to create an input directory in HDFS.

```
$HADOOP_HOME/bin/hadoop fs -mkdir input_dir
```

Step 5

The following command is used to copy the input file named **sample.txt**in the input directory of HDFS.

```
$HADOOP_HOME/bin/hadoop fs -put /home/hadoop/sample.txt input_dir
```

Step 6

The following command is used to verify the files in the input directory.

```
$HADOOP_HOME/bin/hadoop fs -ls input_dir/
```

Step 7

The following command is used to run the Eleunit_max application by taking the input files from the input directory.

```
$HADOOP_HOME/bin/hadoop jar units.jar hadoop.ProcessUnits input_dir output_dir
```

Wait for a while until the file is executed. After execution, as shown below, the output will contain the number of input splits, the number of Map tasks, the number of reducer tasks, etc.

```
INFO mapreduce.Job:Job job_1414748220717_0002  completed
successfully
14/10/3106:02:52

INFO mapreduce.Job:Counters:49

FileSystemCounters



FILE:Number of bytes read=61

FILE:Number of bytes written=279400

FILE:Number of read operations=0

FILE:Number of large read operations=0 FILE:Number of write
operations=0
```

HDFS:Number of bytes read=546

HDFS:Number of bytes written=40

HDFS:Number of read operations=9

HDFS:Number of large read operations=0

HDFS:Number of write operations=2JobCounters

Launched map tasks=2

Launched reduce tasks=1

Data-local map tasks=2

Total time spent by all maps in occupied slots (ms)=146137

Total time spent by all reduces in occupied slots (ms)=441

Total time spent by all map tasks (ms)=14613

Total time spent by all reduce tasks (ms)=44120

Total vcore-seconds taken by all map tasks=146137

Total vcore-seconds taken by all reduce tasks=44120

Total megabyte-seconds taken by all map tasks=149644288

Total megabyte-seconds taken by all reduce tasks=45178880

```
Map-ReduceFramework

Map input records=5

Map output records=5

Map output bytes=45

Map output materialized bytes=67

Input split bytes=208

Combine input records=5

Combine output records=5

Reduce input groups=5

Reduce shuffle bytes=6

Reduce input records=5

Reduce output records=5

SpilledRecords=10

ShuffledMaps=2

FailedShuffles=0

MergedMap outputs=2


  GC time elapsed(ms)=948
  CPU time spent(ms)=5160
```

```
Physical memory (bytes) snapshot=47749120

Virtual memory (bytes) snapshot=2899349504

Total committed heap usage (bytes)=277684224



FileOutputFormatCounters

BytesWritten=40
```

Step 8

The following command is used to verify the resultant files in the output folder.

```
$HADOOP_HOME/bin/hadoop fs -ls output_dir/
```

Step 9

The following command is used to see the output in **Part-00000** file. This file is generated by HDFS.

```
$HADOOP_HOME/bin/hadoop fs -cat output_dir/part-00000
```

Below is the output generated by the MapReduce program.

```
198134

198440

198545
```

Step 10

The following command is used to copy the output folder from HDFS to the local file system for analyzing.

```
$HADOOP_HOME/bin/hadoop fs -cat output_dir/part-00000/bin/hadoop dfs get output_dir
/home/hadoop
```

CONCLUSION:

Designed a distributed application using MapReduce which processes a log file of a system.

QUESTIONS:

1.What is MapReduce?

2.Explain MapReduce Job Processing?

3. Explain MapReduce Algorithm?

4. Define Payload, Datanode ,Masternode, Slavenode, Namednode?

Group B			
Assignment No 13	**Big Data Analytics – JAVA/SCALA**	**Page No**	

Aim:

Locate dataset (e.g., sample_weather.txt) for working on weather data which reads the text input files and finds average for temperature, dew point and wind speed.

Objectives:

1) Develop in depth understanding and implementation of the key technologies in Data Science and Big Data Analytics.
2) Gain practical, hands-on experience with statistics programming languages and Big Data tools.
3) Implement and evaluate data analytics algorithms and able to read text datasets.
4) Use cutting edge tools and technologies to analyse Big Data (i.e., Find average of temperature, dew point and wind speed.)

Theory:

Here, we will write a Map-Reduce program for analyzing weather datasets to find average of temperature, dew point and wind speed. Weather sensors are collecting weather information across the globe in a large volume of log data. This weather data is semi-structured and record-oriented.

This data is stored in a line-oriented ASCII format, where each row represents a single record. Each row has lots of fields like longitude, latitude, daily max-min temperature, daily average temperature, dew points, wind speed etc. for easiness, we will focus on the main element, i.e. temperature, dew point and wind speed. We will use the data from the National Centres for Environmental Information(NCEI). It has a massive amount of historical weather data that we can use for our data analysis.

Mapper class and map method:-

The very first thing which is required for any map reduce problem is to understand what will be the type of keyIn, ValueIn, KeyOut,ValueOut for the given Mapper class and followed by type of map method parameters.

• public class WhetherForcastMapper extends Mapper

• Object (keyIn) - Offset for each line, line number 1, 2...

• Text (ValueIn) - Whole string for each line (CA_25-Jan-2014 00:12:345 .....)

• Text (KeyOut) - City information with date information as string

• Text (ValueOut) - Temperature and time information which need to be passed to reducer as string. • public void map(Object keyOffset, Text dayReport, Context con) { }

• KeyOffset is like line number for each line in input file.

• dayreport is input to map method - whole string present in one line of input file.

• con is context where we write mapper output and it is used by reducer.

Reducer class and reducer method:-

Similarly,we have to decide what will be the type of keyIn, ValueIn, KeyOut,ValueOut for the given Reducer class and followed by type of reducer method parameters.

• public class WhetherForcastReducer extends Reducer

• Text(keyIn) - it is same as keyOut of Mapper.

• Text(ValueIn)- it is same as valueOut of Mapper.

• Text(KeyOut)- date as string

• text(ValueOut) - reducer writes max and min temperature with time as string

• public void reduce(Text key, Iterable values, Context context)

• Text key is value of mapper output. i.e:- City & date information

• Iterable values - values stores multiple temperature values for a given city and date.

• context object is where reducer write it's processed outcome and finally written in file.

Step 1:

We can download the dataset from NCEI, For various cities in different years. choose the year of your choice and select any one of the data text-file for analyzing.
We can get information about data from README.txt file available on the NCEI website.

Step 2:

Make a new java project with name MyProject:

Create a three new classes with names AvgTemp, AvgDewPoint, AvgWindSpeed.

Write the source code to the AvgTemp java class, AvgDewPoint java class, AvgWindSpeed java class.

Now we need to add external jar for the packages that we have import. Download the jar package Hadoop Common and Hadoop MapReduce Core according to your Hadoop version.
You can check Hadoop Version:

```
hadoop version
```

Now we add these external jars to our MyProject. Right Click on MyProject -> then select Build Path-> Click on Configure Build Path and select Add External jars…. and add jars from it's download location then click -> Apply and Close.

Now export the project as jar file. Right-click on MyProject choose Export.. and go to Java -> JAR file click -> Next and choose your export destination then click -> Next.
choose Main Class as MyMaxMin by clicking -> Browse and then click -> Finish -> Ok.

Step 3:

Start our Hadoop Daemons

```
start-dfs.sh
```

```
start-yarn.sh
```

Step 4:

Move your dataset to the Hadoop HDFS.

Syntax:

```
hdfs dfs -put /file_path /destination
```

In below command / shows the root directory of our HDFS.

```
hdfs dfs -put /home/rushikesh/Downloads/CRND0103-2020-AK_Fairbanks_11_NE.txt /
```

Check the file sent to our HDFS.

```
hdfs dfs -ls /
```

Step 5:

Now Run your Jar File with below command and produce the output in MyOutput File.

Syntax:

```
hadoop jar /jar_file_location /dataset_location_in_HDFS /output-file_name
```

Command:

```
hadoop jar /home/rushikesh/Documents/Project.jar /CRND0103-2020-AK_Fairbanks_11_NE.txt /MyOutput
```

Step 6:

Now Move to localhost:50070/, under utilities select Browse the file system and download part-r-00000 in /MyOutput directory to see result.

Step 7:

See the result in the Downloaded File.

You can see the average of temperature, dew point and wind speed in the output file.

Code:

```
# Download dataset.zipfile (Weather data)
# It contains NCDC weather data from year 1901 to year 1920.
# Copy and extract dataset.zip in yourhome folder
# Open terminal
whoami

# It will display your user name, we will use it later.
# Open eclipse->new java project->project name exp7
->new class->

MaxTemperatureMapper


# Add following code in that class

Packageexp7;

Importjava.io.IOException;

Importorg.apache.hadoop.io.IntWritable;


Importorg.apache.hadoop.io.LongWritable;


Importorg.apache.hadoop.io.Text;
Importorg.apache.hadoop.mapreduce.Mapper;
Publicclass MaxTemperatureMapper
extendsMapper<LongWritable,Text,Text,IntWritable>
{

PrivatestaticfinalintMISSING= 9999;

@Overridepublicvoid map(LongWritablekey,Text value,Contextcontext)

throws IOException,InterruptedException

{

String line = value.toString();

String year = line.substring(15, 19);

intairTemperature;

if (line.charAt(87)=='+')

{

airTemperature= Integer.parseInt(line.substring(88, 92));
```

```
}
else
{
airTemperature= Integer.parseInt(line.substring(87, 92));
}
Stringquality= line.substring(92, 93);
if (airTemperature != MISSING&& quality.matches("[01459]"))
{
context.write(new Text(year), new IntWritable(airTemperature));
}
}
}
```

Save the file

It will display some errors, so we are going to import two jar files in our project.

Copy hadoop-mapreduce-client-core-2.7.1.jar from ~/hadoop/share/hadoop/mapreduce directory

In eclipse-> right click on exp7 project- >paste

Right click on pasted hadoop-mapreduce-client-core-2.7.1.jar-> Buid path-> add to buid path

#Copy hadoop-common-2.7.1.jar from ~/hadoop/share/hadoop/common directory

In eclipse-> right click on exp7 project- >paste

Right click on pasted hadoop-common-2.7.1.jar-> Buid path-> add to buid path

Right click on project exp7->new class-> MaxTemperatureReducer

Add following code in that class

```
Packageexp7;

Importjava.io.IOException;

Importorg.apache.hadoop.io.IntWritable;

Importorg.apache.hadoop.io.Text;

Importorg.apache.hadoop.mapreduce.Reducer;

PublicclassMaxTemperatureReducerextendsReducer<Text,IntWritable,Text,IntWritable>

{

@Overridepublicvoidreduce(Text key, Iterable<IntWritable> values, Context context)

ThrowsIOException,InterruptedException

{

IntmaxValue= Integer.MIN_VALUE;

For(IntWritable value: values)

{

maxValue= Math.Max(maxValue,value.get());

}

Context.write(key,newIntWritable(maxValue));

}

}

# Save the file

# Right click on project exp7->new class->
MaxTemperature

# Add following code in that class
( replace

your_user_name

by your own username

)

# hdfs port number here is 1234, replace it with your port no (if different).
```

```
Packageexp7;
Importjava.io.BufferedReader;
Importjava.io.File;
Importjava.io.FileReader;
Importjava.util.ArrayList;
Importjava.util.List;

Importjava.util.Scanner;
Importorg.apache.hadoop.conf.Configuration;
Importorg.apache.hadoop.fs.FileStatus;
Importorg.apache.hadoop.fs.FileSystem;
import org.apache.hadoop.fs.Path;
import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Job;
import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;
import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;
public classMaxTemperature
{
Publicstaticvoidmain(String[] args)
ThrowsException
{
If(args.length != 2)
{
System.err.println("Usage:MaxTemperature<input path><output path>");
System.exit(-1);
}
@SuppressWarnings
(
```

```
Job job= newJob();

Job.setJarByClass(MaxTemperature.class);

Job.setJobName("Max temperature");

FileInputFormat.addInputPath(job,newPath(args[0]));

FileOutputFormat.setOutputPath(job,newPath(args[1]));

job.setMapperClass(MaxTemperatureMapper.class);

job.setReducerClass(MaxTemperatureReducer.class);

job.setOutputKeyClass(Text.class);

job.setOutputValueClass(IntWritable.class);

job.waitForCompletion(true);

Configuration conf= newConfiguration();

conf.set("fs.defaultFS",
"hdfs://localhost:1234/user/your_user_name/"); FileSystem fs=
FileSystem.get(conf);

FileStatus[] status= fs.listStatus(newPath(args[1]));

//copy hdfsoutput file to local folder

for(inti=0;i<status.length;i++)

{

System.out.println(status[i].getPath());

fs.copyToLocalFile(false, status[i].getPath(), newPath("/home/your_user_name/"+args[1]));

}

System.out.println("\nYear\tTemperature\n");

//display contents of local file

"deprecation"

)

BufferedReader br=
newBufferedReader(newFileReader("/home/your_user_name/"+args[1])); String
line= null;
```

```
while((line= br.readLine()) != null)
{
System.out.println(line);
}
br.close();

Scanner s= newScanner(newFile("/home/your_user_name/"+args[1]));
List<Integer> temps= newArrayList<Integer>();
List<String> years= newArrayList<String>();
while(s.hasNext())
{
years.add(s.next());
temps.add(Integer.parseInt(s.next()));
}
Intmax_temp=0,min_temp=999,i=0,j=0;
String hottest_year="", coolest_year="";
for(inttemp: temps)
{
if(temp>max_temp)
{
max_temp=temp;
hottest_year=years.get(i);
}
i++;
}
Floatmax_temp1=max_temp;
System.out.println("Hottest Year:"+hottest_year);

System.out.println("\tTemperature:"+max_temp1/10+" Degree Celcius");
for(inttemp: temps)
{
```

```
if(temp<min_temp)
{
min_temp=temp;
coolest_year=years.get(j);
}
j++;
}
Floatmin_temp1=min_temp;
System.out.println("Coolest Year:"+coolest_year);
System.out.println("\tTemperature:"+min_temp1/10+" Degree Celcius");
s.close();
}
}
```

Save the file

In eclipse->Right click on project exp7-> export->java->jar file->next-> select the export

destination ->
/home/
your_user_name
/exp7.jar

->next -> next -> select main class ->browse -

>

MaxTemperature -

>

Finish

#
exp7.jar

file will be created in your home folder

Open terminal
Now Start NameNode daemon and DataNode daemon:

```
# ~/hadoop/sbin/start-dfs.sh

# Make the HDFS directories required to execute MapReduce jobs (if not already
done)

~/hadoop/bin/hdfs dfs -mkdir /user
~/hadoop/bin/hdfs dfs -mkdir /user/
your_user_name


# Put NCDC weather dataset in hdfs
~/hadoop/bin/hdfs dfs -put ~/dataset input_dataset

# Perform MapReduce job

~/hadoop/bin/hadoop jar ~/exp7.jar input_dataset output_dataset

# Output
# Stop haddop
~/hadoop/sbin/stop-dfs.sh
jps
#
```

Conclusion:

We have successfully executed the program which finds average for temperature, dew point and wind speed by taking a text file as a input from a dataset.

Questions:

1. What are Mapper class and map methods?
2. What are Reducer class and reducer methods?
3. How does data cleaning play a vital role in the data analysis?
4. What are different types of data?

Group B			
Assignment No 13	**Big Data Analytics – JAVA/SCALA**	**Page No**	

Aim: Write a simple program in SCALA using Apache Spark framework

---

Objective: Student should perform simple program in scala using Apache Spark framework

---

Outcome:

Student will able to learn how to install scala and how to write simple programs in scala

---

Content / Theory:

Step 1: open the terminal

Step 2: install java

If you are asked to accept Java license terms, click on "Yes" and proceed. Once finished, let us check whether Java has installed successfully or not. To check the Java version and installation, you can type:

```
$ sudo apt-add-repository ppa:webupd8team/java
$ sudo apt-get update
$ sudo apt-get install oracle-java7-installer
```

```
$ java -version
```

Step 3: Once Java is installed, we need to install Scala

```
$ sudo apt-get install scala
```

```
$ scala –version
```

This will show you the version of Scala installed

Choosing development environment

Once you have installed Scala, there are various options for choosing an environment. Here are the 3 most common options:

- Terminal / Shell based
- Notepad / Editor based
- IDE (Integrated development environment)

Scala basic terms

Object: An entity that has state and behavior is known as an object. For example: table, person, car etc.

Class: A class can be defined as a blueprint or a template for creating different objects which defines its properties and behavior.

Method: It is a behavior of a class. A class can contain one or more than one method. For example: deposit can be considered a method of bank class.

Closure: Closure is any function that closes over the environment in which it's defined. A closure returns value depends on the value of one or more variables which is declared outside this closure.

Traits: Traits are used to define object types by specifying the signature of the supported methods. It is like interface in java.

Things to note about Scale

- It is case sensitive
- If you are writing a program in Scala, you should save this program using "**.scala**"
- Scala execution starts from main() methods
- Any identifier name cannot begin with numbers. For example, variable name "123salary" is invalid.
- You cannot use Scala reserved keywords for variable declarations or constant or any identifiers.

Variable declaration in scala

In Scala, you can declare a variable using 'var' or 'val' keyword. The decision is based on whether it is a constant or a variable. If you use 'var' keyword, you define a variable as mutable variable. On the

other hand, if you use 'val', you define it as immutable. Let's first declare a variable using "var" and then using "val".

```
var Var1 : String = "Ankit"
```

In the above Scala statement, you declare a mutable variable called "Var1" which takes a string value. You can also write the above statement without specifying the type of variable. Scala will automatically identify it. For example

```
var Var1 = "Gupta"
```

Operations on variables

You can perform various operations on variables. There are various kinds of operators defined in Scala. For example: Arithmetic Operators, Relational Operators, Logical Operators, Bitwise Operators, and Assignment Operators.

```
scala> var Var4 = 2
Output: Var4: Int = 2
scala> var Var5 = 3
Output: Var5: Int = 3
```

Apply '+' operator

```
Var4+Var5
Output:
res1: Int = 5
```

Apply "==" operator

```
Var4==Var5
Output:
res2: Boolean = false
```

Iteration in Scala

Like most languages, Scala also has a FOR-loop which is the most widely used method for iteration. It has a simple syntax too.

```
for( a <- 1 to 10){
 println( "Value of a: " + a );
 }
```

Output:

Value of a: 1

Value of a: 2

Value of a: 3

Value of a: 4

Value of a: 5

Value of a: 6

Value of a: 7

Value of a: 8

Value of a: 9

Value of a: 10

The if-else expression in Scala

In Scala, if-else expression is used for conditional statements. You can write one or more conditions inside "if". Let's declare a variable called "Var3" with a value 1 and then compare "Var3" using if-else expression

```
var Var3 =1
if (Var3 ==1){
 println("True")}else{
 println("False")}
```

Output: True

Declare a simple function in Scala and call it by passing value

You can define a function in Scala using "def" keyword. Let's define a function called "mul2" which will take a number and multiply it by 10. You need to define the return type of function, if a function not returning any value you should use the "Unit" keyword.

In the below example, the function returns an integer value. Let's define the function "mul2":

```
def mul2(m: Int): Int = m * 10

Output: mul2: (m: Int)Int
```

Now let's pass a value 2 into mul2

```
mul2(2)

Output:

res9: Int = 20
```

Writing & Running a program in Scala using an editor

Let us start with a "Hello World!" program. It is a good simple way to understand how to write, compile and run codes in Scala. (Save this file with extension .scala and compile in same directory)

```
object HelloWorld {
 def main(args: Array[String]) {
 println("Hello, world!")
 }
 }
```

As mentioned before, if you are familiar with Java, it will be easier for you to understand Scala. If you know Java, you can easily see that the structure of above "HelloWorld" program is very similar to Java program.

This program contains a method "main" (not returning any value) which takes an argument – a string array through command line. Next, it calls a predefined method called "Println" and passes the argument "Hello, world!"

Compile a Scala Program

Let's start compiling your "HelloWorld" program using the following steps:

1. For compiling it, you first need to paste this program into a text file then you need to save this program as HelloWorld.scala
2. Now you need change your working directory to the directory where your program is saved.
3. After changing the directory you can compile the program by issuing the command.
4. After compiling, you will get Helloworld.class as an output in the same directory. If you can see the file, you have successfully compiled the above program.

```
scalac HelloWorld.scala
```

Running Scala Program

After compiling, you can now run the program using following command

```
scala HelloWorld
```

You will get an output if the above command runs successfully. The program will print "**Hello, world!"**

Conclusion:

Hence from this practical we learned how to write simple programs in scala using using Apache Spark

Questions

1. What are some main features of Scala?
2. Write some benefits of using Scala ?
3. Name some of the frameworks that Scala supports ?
4. What are case classes in Scala?

Group C			
Assignment No 14	**Mini Project / Case Study**	**Page No**	

Aim : Write a case study on Global Innovation Network and Analysis (GINA). Components of analytic plan are 1. Discovery business problem framed, 2. Data, 3. Model planning analytic technique and 4. Results and Key findings.

Objectives : To understand different concepts of GINA.

Outcomes : Students are able to understand various concepts of GINA.

Theory :

GINA

- The GINA case study provides an example of how a team applied the Data Analytics Lifecycle to analyze innovation data at EMC.
- Innovation is typically a difficult concept to measure, and this team wanted to look for ways to use advanced analytical methods to identify key innovators within the company.
- GINA is a group of senior technologists located in centers of excellence (COEs) around the world.
- The GINA team thought its approach would provide a means to share ideas globally and increase knowledge sharing among GINA members who may be separated geographically.
- It planned to create a data repository containing both structured and unstructured data to accomplish three main goals.
  1. Store formal and informal data.
  2. Track research from global technologists.
  3. Mine the data for patterns and insights to improve the team's operations and strategy.

Phase 1: Discovery

- In the GINA project's discovery phase, the team began identifying data sources.
- Following Person are involved in this phase

1. Business user, project sponsor, project manager – Vice President from Office of CTO
2. BI analyst – person from IT
3. Data engineer and DBA – people from IT
4. Data scientist – distinguished engineer

- The data for the project fell into two main categories.
  1. Innovation Roadmap
  2. Data encompassed minutes and notes representing innovation and research activity from around the world

- Hypothesis
  1. Descriptive analytics of what is currently happening to spark further creativity, collaboration, and asset generation.
  2. Predictive analytics to advise executive management of where it should be investing in the future.

Phase 2: Data Preparation

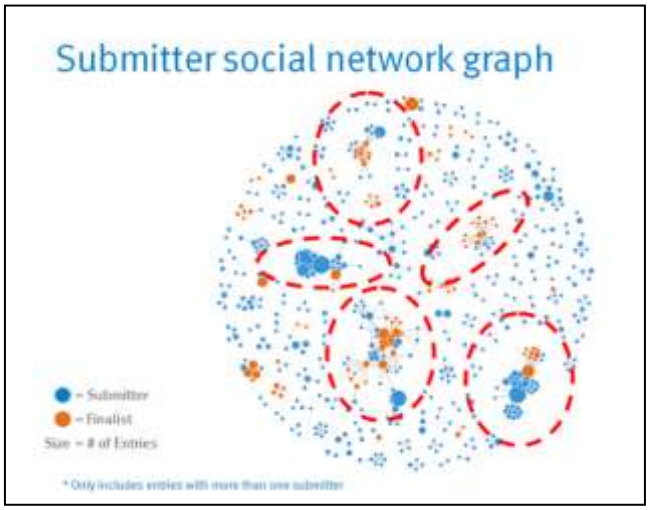
- IT department to set up a new analytics sandbox to store and experiment on the data.
- The data scientists and data engineers began to notice that certain data needed conditioning and normalization.
- As the team explored the data, it quickly realized that if it did not have data of sufficient quality or could not get good quality data, it would not be able to perform the subsequent steps in the lifecycle process.
- Important to determine what level of data quality and cleanliness was sufficient for the project being undertaken.

Phase 3: Model Planning

- The team made a decision to initiate a longitudinal study to begin tracking data points over time regarding people developing new intellectual property.
- The parameters related to the scope of the study included the following considerations:
  1. Identify the right milestones to achieve this goal.
  2. Trace how people move ideas from each milestone toward the goal.
  3. Once this is done, trace ideas that die, and trace others that reach the goal. Compare the journeys of ideas that make it and those that do not.
  4. Compare the times and the outcomes using a few different methods (depending on how the data is collected and assembled). These could be as simple as t-tests or perhaps involve different types of classification algorithms.

Phase 4: Model Building

- The GINA team employed several analytical methods. This included work by the data scientist using Natural Language Processing (NLP) techniques on the textual descriptions of the Innovation Roadmap ideas.
- Social network analysis using R and Studio



- Fig shows social graphs that portray the relationships between idea submitters within GINA.
- Each color represents an innovator from a different country.
- The large dots with red circles around them represent hubs. A **hub** represents a person with high connectivity and a high "betweenness" score.
- The team used Tableau software for data visualization and exploration and used the Pivotal Greenplum database as the main data repository and analytics engine.

Phase 5: Communicate Results

- This project was considered successful in identifying boundary spanners and hidden innovators.
- The GINA project promoted knowledge sharing related to innovation and researchers spanning multiple areas within the company and outside of it. GINA also enabled EMC to cultivate additional intellectual property that led to additional research topics and provided opportunities to forge relationships with universities for joint academic research in the fields of Data Science and Big Data.
- Study was successful in identifying hidden innovators
    - Found high density of innovators in Cork, Ireland

- The CTO office launched longitudinal studies

Phase 6: Operationalize

- Deployment was not really discussed
- Key findings
    1. Need more data in future
    2. Some data were sensitive
    3. A parallel initiative needs to be created to improve basic BI activities
    4. A mechanism is needed to continually reevaluate the model after deployment

Analytic Plan from the EMC GINA Project

Components of Analytic Plan	GINA Case Study
Discovery Business Problem Framed	Tracking global knowledge growth, ensuring effective knowledge transfer, and quickly converting it into corporate assets. Executing on these three elements should accelerate innovation.
Initial Hypotheses	An increase in geographic knowledge transfer improves the speed of idea delivery.
Data	Five years of innovation idea submissions and history; six months of textual notes from global innovation and research activities
Model Planning Analytic Technique	Social network analysis, social graphs, clustering, and regression analysis
Result and Key Findings	1. Identified hidden, high-value innovators and found ways to share their knowledge 2. Informed investment decisions in university research projects 3. Created tools to help submitters improve ideas with idea recommender systems

Conclusion :

Hence we successfully understood the concepts of GINA.

Questions

1. What are the components of analytic plan ?
2. What is GINA ?
3. Which type of people are involved in the Phase 1: Discovery ?
4. Which considerations do the parameters related to the scope of the study include ?

Group C			
Assignment No 14	**Mini Project / Case Study**	**Page No**	

Aim :

Use the following dataset and classify tweets into positive and negative tweets. https://www.kaggle.com/ruchi798/data-science-tweets

---

Objectives :

To understand different concepts of Natural LanguageProcessor(NLP).

Outcomes :

Students are able to understand various concepts of NLP.

Theory :

Natural Language Processing (NLP)

Natural language processing strives to build machines that understandand respond to text or voice data and respond with text or speech of their own in much the same way humans do.

Natural language processing (NLP) refers to the branch of computer science and more specifically, the branch of artificial intelligence or AI concerned with giving computers the ability to understand text and spokenwords in much the same way human beings can.

NLP combines computational linguistics rule-based modeling of human language with statistical, machine learning, and deep learning models. Together, these technologies enable computers to process human language in the form of text or voice data and to ‘understand’ its full meaning,complete with the speaker or writer’s intent and sentiment.

NLP Tools and Approaches:

Python and the Natural Language Toolkit (NLTK):

The Python programming language provides a wide range of tools andlibraries for attacking specific NLP tasks. Many of these are found in the Natural Language Toolkit, or NLTK, an open source collection of libraries,programs, and education resources for building NLP programs.

The NLTK includes libraries for many of the NLP tasks listed above, pluslibraries for subtasks, such as sentence parsing, word segmentation, stemmingand lemmatization (methods of trimming words down to their roots), and tokenization (for breaking phrases, sentences, paragraphs and passages into tokens that help the computer better understand the text). It also includes libraries for implementing capabilities such as semantic reasoning, the ability to reach logical conclusions based on facts extracted from text.

So what are we doing in the Twitter Sentiment Analysis Project?

I am going to build a machine learning model that is able to analyze loadsof twitter tweets, and be able to judge the sentiments behind the tweets. We are gonna be analyzing tweets made against an airline, and judging if they are of positive, negative or neutral sentiments. These types of models are massively used by Twitter and Facebook, to filter out to say hate speeches, or other unwanted comments on their platforms.

Step 1: We Import the Dataset in Google Colab / Jupyter Notebook

In the first line we import the pandas library, which is a very popular library that makes data manipulation easy. In the next line we read it using thepd.read_csv command, since the dataset is a csv file and then we print the first5 lines of the dataset. Here's how it looks like

```
import pandas as pd
tweets=pd.read_csv('Tweets.csv')
tweets.head()
```

The dataset contains a lot more parameters about the tweet like the location of the tweet, time zone of the user, etc. But for the sake of building a model we are only concerned about 2 parameters: the actual **text** of the tweet(input of the NLP model) and the **airline sentiment** (output of the NLP model).

Since this the dataset that we will be using to train our model, we have make sure that the tweets are correctly classified into the right sentiments. Youwill notice a column in the dataset that says 'airline_sentiment_confidence'. It basically says how sure you are about the sentiment of the tweet.

So for the sake of building a good model, let's drop all the tweets whichhave lower confidence than say, 0.5 (basically 50% sure about the correct classification of the sentiment). We do that using the following lines and theprint the shape of the data.

```
tweets_df=tweets.drop(tweets[tweets['airline_sentiment_confidence']<0.5].index,axis=0)
tweets_df.shape
```

In NLP projects, the most important part is preprocessing the data, so that it is ready for a model to use. That's what we are doing now.

Since there are a lot of unnecessary parameters in the dataset, lets extract andstore the input and the output, which is the text and airline_sentiment confidence column using the following lines:

```
X=tweets_df['text']
y=tweets_df['airline_sentiment']
```

Why and What at all do we need to do in Data Processing?

Remember your input is just a bunch of words for now. Machine learning models only understand and work on numbers. So we have to convert all the text into numbers. Fair enough, now that it is decided that we need to convert the words into numbers, do we need to convert ALL the words? The answer is **No.** So let's take the tweet 'The food of this airline was bad'. Now did we need to know all these words to know about the sentiment?

What if the tweet simply said '**Food airline bad'**. It is still enough to understand that the food of the airline is terrible right? So the entire goal is to reduce the number of unnecessary words, just for the sake of reducing and timeand space complexity.

To remove unnecessary words, I am going to use the following techniques:

1. **Removing Stop Word**s: Basically words like this, an, a, the, etc that do notaffect the meaning of the tweet.

2. **Removing Punctuation**: ‘,.*!’ and other punctuation marks that are notreally needed by the model.

3. **Stemming**: Basically reducing words like ‘jumping, jumped, jump’ into its root word(also called stem), which is jump in this case. Since all variations ofthe root word convey the same meaning, we don’t need each of the word to be converted into different numbers.

```
from nltk.corpus import stopwords
nltk.download('stopwords')
import string
from nltk.stem import PorterStemmer
```

```
[nltk_data] Downloading package stopwords to /root/nltk_data...
[nltk_data]   Package stopwords is already up-to-date!
```

```
stop_words=stopwords.words('english')
punct=string.punctuation
stemmer=PorterStemmer()
```

Now, for the actual data processing, we use the following lines of code.

```
import re
cleaned_data=[]
for i in range(len(X)):
        tweet=re.sub('[^a-zA-Z]',' ',X.iloc[i])
         tweet=tweet.lower().split()
        tweet=[stemmer.stem(word) for word in tweet if (word not in stop_words) and
        (word not in punct)] tweet=' '.join(tweet)
        cleaned_data.append(tweet)
```

Firstly, we create an empty list called cleaned_data, where will be storingour text data after getting rid of all unnecessary words. We import a library called re(regular expressions), that is gonna help us remove a lot of unnecessarystuff. To begin with, we run a for loop to iterate through each and every tweet in the dataset, at a time. Using re.sub we can substitute and replace things in a sentence. So we are basically taking one tweet at a time, and inside it whatever is not a letter (belonging to a-z

or A-Z), will be replaced by an empty space. It will automatically filter out punctuation marks and other non-letters.

After that we join all the words using ‘’.join(tweet), to get a single sentence instead of separated words. And then we simply append that into the cleaned data list that we had created. If you print the cleaned list it should looksomething like this.

cleaned_data

print(y)

In this, virginamerica and unit are probably the names of the airline companies for which the tweets are made. We'll also remove them later, as theydo not play a role in the sentiment of the tweet. Also you'll notice weird words like 'industri' which do not make sense. That is because people in tweets will often use incorrect spellings of words and also the stemmer function sometimes produces words that are not real English words, but that is something that we will just roll with for now.

Bringing in the Bag of Words Approach

Now that input is clean and ready, we convert it into numbers using something called as the 'Bag of Words' approach. Basically we create a matrixtable, where each row represents a sentence and each word will have separatecolumn for itself that represents it's frequency. Let's take an example, to makeit easier. Let's say there are 3 tweets like: ' The food is really bad', 'The food is really good', 'The food is really good and tasty'. This could be represented as matrix using the bag of words approach as shown:

Tweet Number	The	food	is	bad	really	good	tasty	and
1	1	1	1	1	1	0	0	0
2	1	1	1	0	1	1	0	0
3	1	1	1	0	1	1	1	1

One con about this method that you might not notice is that the order of thesentence is lost. There are other approaches to counter this, but we are just going to stick with this method.

We use the following lines of code to do the same:

```
from sklearn.feature_extraction.text import CountVectorizer
cv=CountVectorizer(max_features=3000,stop_words=['virginamerica','unit'])
X_fin=cv.fit_transform(cleaned_data).toarray()
X_fin.shape
```

→ (14404, 3000)

We just have to do the last step now, which is to actually build the NLP modelusing the input and the output.

We will use the Multinomial Naive Bayes model to figure out the relationship between the input and the output.

```
from sklearn.naive_bayes import MultinomialNB
from sklearn.model_selection import train_test_split
model=MultinomialNB()
```

Now to build the model we split the dataset into a training and testing section (testing size=30% of the actual data). To actually fit the model, we callthe model.fit function and supply it the training input and output.

```
X_train, X_test, y_train, y_test = train_test_split(X_fin, y, test_size = 0.3
)model.fit(X_train,y_train)

y_pred = model.predict(X_test)
from sklearn.metrics import classification_report
cf=classification_report(y_test, y_pred)
print(cf)
```

	precision	recall	f1-score	support
negative	0.83	0.90	0.86	2710
neutral	0.65	0.49	0.56	941
positive	0.70	0.70	0.70	671
accuracy			0.78	4322
macro avg	0.73	0.70	0.71	4322
weighted avg	0.77	0.78	0.77	4322

Conclusion: Hence we successfully classified tweets into positive, neutral and negative.

Questions:

1. What is NLP ?
2. What is NLTK ?
3. What are the different techniques are given to remove unnecessarywords ?
4. What is Stemming ?

Group C			
Assignment No 14	**Mini Project / Case Study**	**Page No**	

Aim :

Develop a movie recommendation model using the scikit-learn library in python.

Refer dataset

https://github.com/rashida048/Some-NLP-Projects/blob/master/movie_dataset.csv

---

Objectives : To build a movie recommendation model using the scikit-learn library in python.

Outcomes : Students are able to build a movie recommendation model.

Theory :

There are three ways to build a recommendation engine:

1. Popularity based recommendation engine

Perhaps, this is the simplest kind of recommendation engine that you will come across. The trending list you see in YouTube or Netflix is based on this algorithm. It keeps a track of view counts for each movie/video and then lists movies based on views in descending order(highest view count to lowest view count).

2. Content based recommendation engine

This type of recommendation systems, takes in a movie that a user currently likes as input. Then it analyses the contents (storyline, genre, cast, director etc.) of the movie to find out other movies which have similar content. Then it ranks similar movies according to their similarity scores and recommends the most relevant movies to the user.

3. Collaborative filtering based recommendation engine

This algorithm at first tries to find similar users based on their activities and preferences (for example, both the users watch same type of movies or movies directed by the same director). Now, between these users(say, A and B) if user A has seen a movie that user B has not seen yet, then that movie gets recommended to user B and vice-versa. In other words, the recommendations get filtered based on the collaboration between similar user's preferences (Thus, the name "Collaborative Filtering").

1. Importing Libraries and Load data set

We need to import all the required libraries and then read the csv file using read_csv() method.

```
import pandas as pd
import numpy as np
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.metrics.pairwise import cosine_similarity

df = pd.read_csv('movie_dataset.csv', header = None)
```

2. Selecting the features

If you visualize the dataset, you will see that it has many extra info about a movie. We don't need all of them. So, we choose keywords, cast, genres and director column to use as our feature set(the so called "content" of the movie).

```
features = ['keywords', 'cast','genres', 'director']
```

3. Filling null values

We have to fill missing values in dataset with Na or na values.

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

4. Combining values of columns in string

Our next task is to create a function for combining the values of these columns into a single string.

```
def combine_features(row):
    return row['keywords']+" "+row['cast']+" "+row['genres']+" "+row['direct
or']
```

5. Filling all NA values with blank string

Now, we need to call this function over each row of our dataframe. But, before doing that, we need to clean and preprocess the data for our use. We will fill all the NaN values with blank string in the dataframe.

```
for feature in features:
    df[feature] = df[feature].fillna('')

df["combined_features"] = df.apply(combine_features,axis=1)
```

6. Obtained the combined string

Now that we have obtained the combined strings, we can now feed these strings to a CountVectorizer() object for getting the count matrix.

```
cv = CountVectorizer() #creating new CountVectorizer()
count_matrix = cv.fit_transform(df["combined_features"])
```

7. Obtain cosine similarity

Cosine similarity is a metric used to determine how similar the documents are irrespective of their size.

```
cosine_sim = cosine_similarity(count_matrix)
```

8. Define helper function

Now, we will define two helper functions to get movie title from movie index and vice-versa.

```
def get_title_from_index(index):
    return df[df.index == index]["title"].values[0]
def get_index_from_title(title):
    return df[df.title == title]["index"].values[0]
```

9. Get the title of the movie that the user currently likes and find the index of movie

Our next step is to get the title of the movie that the user currently likes. Then we will find the index of that movie. After that, we will access the row corresponding to this movie in the similarity matrix. Thus, we will get the similarity scores of all other movies from the current movie. Then we will enumerate through all the similarity scores of that movie to make a tuple of movie index and similarity score. This will convert a row of similarity scores like this- [1 0.5 0.2 0.9] to this- [(0,1), (1,0.5), (2,0.2), (3,0.9)]. Here, each item is in this form- (movie index, similarity score)

```
movie_user_likes = "Star Wars"
movie_index = get_index_from_title(movie_user_likes)
similar_movies = list(enumerate(cosine_sim[movie_index]))
```

10. Sorting the movie

Now comes the most vital point. We will sort the list similar_movies according to similarity scores in descending order. Since the most similar movie to a given movie will be itself, we will discard the first element after sorting the movies.

```
sorted_similar_movies = sorted(similar_movies,key=lambda x:x[1],reve
rse=True)[1:]
```

11. Printing movies list

Now, we will run a loop to print first 5 entries from sorted_similar_movies list.

```
i=0
print("Top 5 similar movies to "+movie_user_likes+" are:\n")
for element in sorted_similar_movies:
    print(get_title_from_index(element[0]))
    i=i+1
    if i>=5:
        break
```

```
Top 5 similar movies to Star Wars are:

The Empire Strikes Back
Return of the Jedi
Star Wars: Episode II - Attack of the Clones
Star Wars: Episode III - Revenge of the Sith
Star Wars: Episode I - The Phantom Menace
```

Conclusion : Hence we successfully built a movie recommendation model using the scikit-learn library in python.

Questions

1. What is Data pre-processing ?
2. What are different method to fill missing values in dataset ?
3. What is lambda function ?
4. Define cosine similarity ?

Group C			
Assignment No 15	**Mini Project / Case Study**	**Page No**	

Aim :

Use the following covid_vaccine_statewise.csv dataset and perform following analytics on the given dataset

https://www.kaggle.com/sudalairajkumar/covid19-in-india?select=covid_vaccine_statewise.csv

a. Describe the dataset

b. Number of persons state wise vaccinated for first dose in India

c. Number of persons state wise vaccinated for second dose in India

d. Number of Males vaccinated

d. Number of females vaccinated

Objectives : To perform analytics on the given dataset (covid_vaccine_statewise.csv)

Outcomes : Students are able to perform analytics on the given dataset (covid_vaccine_statewise.csv)

Theory :

Vaccination for Covid-19 in India was started on 16 January 2021. Two vaccines received approval for emergency use in India at the onset of the programme

1. Covishield (a brand of the Oxford–AstraZeneca vaccine manufactured by the Serum Institute of India)
2. Covaxin (developed by Bharat Biotech).
   In April 2021, Sputnik V (distributed by Dr. Reddy's Laboratories) was approved as a third vaccine, which was deployed by May 2021. Each vaccine is divided into two doses.

Here we will analyse the data set containing all the information of vaccination done in past 5 months. The data set contain the number of people vaccinated daily in different states of the country. The data set in this analysis is verified and is collected from official website of cowin (https://www.cowin.gov.in/).

The tools that are used for analysis are:

- Pandas
- Numpy
- Matplotlib
- Seaborn

Importing all the libraries of python that are used in this analysis

```
import pandas as pd
import numpy as np
import matplotlib
import matplotlib.pyplot as plt
import seaborn as sns
```

Let's load our covid_vaccine_statewise csv file now.

```
raw_vaccine_data = pd.read_csv("covid_vaccine_statewise.csv")
raw_vaccine_data.shape
raw_vaccine_data.head(5)
```

```
raw_vaccine_data.columns
```

```
Index(['Updated On', 'State', 'Total Individuals Vaccinated',
       'Total Sessions Conducted', 'Total Sites ', 'First Dose Administered',
       'Second Dose Administered', 'Male(Individuals Vaccinated)',
       'Female(Individuals Vaccinated)', 'Transgender(Individuals Vaccinated)',
       'Total Covaxin Administered', 'Total CoviShield Administered', 'AEFI',
       '18-30 years (Age)', '30-45 years (Age)', '45-60 years (Age)',
       '60+ years (Age)', 'Total Doses Administered'],
      dtype='object')
```

Data Preparation and Cleaning

In this section, we will do data cleaning and prepare data for easy further analysis.

Let's get the details about the data like data type and missing values.

```
raw_vaccine_data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 4552 entries, 0 to 4551
Data columns (total 18 columns):
 #   Column                              Non-Null Count  Dtype
---  ------                              --------------  -----
 0   Updated On                          4552 non-null   object
 1   State                               4552 non-null   object
 2   Total Individuals Vaccinated        4551 non-null   float64
 3   Total Sessions Conducted            4551 non-null   float64
 4   Total Sites                         4551 non-null   float64
 5   First Dose Administered             4551 non-null   float64
 6   Second Dose Administered            4551 non-null   float64
 7   Male(Individuals Vaccinated)        4551 non-null   float64
 8   Female(Individuals Vaccinated)      4551 non-null   float64
 9   Transgender(Individuals Vaccinated) 4551 non-null   float64
 10  Total Covaxin Administered          4551 non-null   float64
 11  Total CoviShield Administered       4551 non-null   float64
 12  AEFI                                2368 non-null   float64
 13  18-30 years (Age)                   2368 non-null   float64
 14  30-45 years (Age)                   2368 non-null   float64
 15  45-60 years (Age)                   2368 non-null   float64
 16  60+ years (Age)                     2368 non-null   float64
 17  Total Doses Administered            4552 non-null   int64
dtypes: float64(15), int64(1), object(2)
memory usage: 640.2+ KB
```

- Above data contains 18 columns in which 15 columns are float64, 2 are object and 1 is int64 type.
- As we know the "Updated On" column contains date values so we have to change it into date type.

We can convert date column from object to datetime by using pd.to_datetime method.

```
raw_vaccine_data["Updated On"] = pd.to_datetime(raw_vaccine_data["Updated On"],dayfirst=True)
```

Now check the data types of our data again.

```
raw_vaccine_data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 4552 entries, 0 to 4551
Data columns (total 18 columns):
 #   Column                                Non-Null Count  Dtype
---  ------                                --------------  -----
 0   Updated On                            4552 non-null   datetime64[ns]
 1   State                                 4552 non-null   object
 2   Total Individuals Vaccinated          4551 non-null   float64
 3   Total Sessions Conducted              4551 non-null   float64
 4   Total Sites                           4551 non-null   float64
 5   First Dose Administered               4551 non-null   float64
 6   Second Dose Administered              4551 non-null   float64
 7   Male(Individuals Vaccinated)          4551 non-null   float64
 8   Female(Individuals Vaccinated)        4551 non-null   float64
 9   Transgender(Individuals Vaccinated)   4551 non-null   float64
 10  Total Covaxin Administered            4551 non-null   float64
 11  Total CoviShield Administered         4551 non-null   float64
 12  AEFI                                  2368 non-null   float64
 13  18-30 years (Age)                     2368 non-null   float64
 14  30-45 years (Age)                     2368 non-null   float64
 15  45-60 years (Age)                     2368 non-null   float64
 16  60+ years (Age)                       2368 non-null   float64
 17  Total Doses Administered              4552 non-null   int64
dtypes: datetime64[ns](1), float64(15), int64(1), object(1)
memory usage: 640.2+ KB
```

We can see that data type of "Updated On" column is changed into datetimedata type.

Now we will count the number of missing values in each column.

```
raw_vaccine_data.isnull().sum()
```

```
Updated On                          0
State                               0
Total Individuals Vaccinated        1
Total Sessions Conducted            1
Total Sites                         1
First Dose Administered             1
```

```
Second Dose Administered                1
Male(Individuals Vaccinated)            1
Female(Individuals Vaccinated)          1
Transgender(Individuals Vaccinated)     1
Total Covaxin Administered              1
Total CoviShield Administered           1
AEFI                                 2184
18-30 years (Age)                    2184
30-45 years (Age)                    2184
45-60 years (Age)                    2184
60+ years (Age)                      2184
Total Doses Administered                0
dtype: int64
```

So, we will fill our null values with 0.

```
raw_vaccine_data.fillna(0,inplace=True)
raw_vaccine_data.head(3)
```

Now let's see the unique states of india for which the data is stored in this file.

```
raw_vaccine_data.State.unique()
```

```
array(['India', 'Andaman and Nicobar Islands', 'Andhra Pradesh',
       'Arunachal Pradesh', 'Assam', 'Bihar', 'Chandigarh',
       'Chhattisgarh', 'Dadra and Nagar Haveli and Daman and Diu',
       'Delhi', 'Goa', 'Gujarat', 'Haryana', 'Himachal Pradesh',
       'Jammu and Kashmir', 'Jharkhand', 'Karnataka', 'Kerala', 'Ladakh',
       'Lakshadweep', 'Madhya Pradesh', 'Maharashtra', 'Manipur',
       'Meghalaya', 'Mizoram', 'Nagaland', 'Odisha', 'Puducherry',
       'Punjab', 'Rajasthan', 'Sikkim', 'Tamil Nadu', 'Telangana',
       'Tripura', 'Uttar Pradesh', 'Uttarakhand', 'West Bengal'],
      dtype=object)
```

In above listed states, India is also present which is not the state. May be it contains the sum of values in all states. But it will change the values in our analysis. So we will will drop the rows containing India as State by using drop() function.

```
raw_vaccine_data[raw_vaccine_data.State == "India"].count()
```

```
Updated On                      123
State                           123
Total Individuals Vaccinated    123
Total Sessions Conducted        123
Total Sites                     123
First Dose Administered         123
Second Dose Administered        123
Male(Individuals Vaccinated)    123
Female(Individuals Vaccinated)  123
Transgender(Individuals Vaccinated)  123
Total Covaxin Administered      123
Total CoviShield Administered   123
AEFI                            123
18-30 years (Age)               123
30-45 years (Age)               123
45-60 years (Age)               123
60+ years (Age)                 123
Total Doses Administered        123
dtype: int64
```

Here number of rows containing India as state are 123. So after deleting these rows our dataframe should contain rows

4552-123=4429.

```
raw_vaccine_data.drop(raw_vaccine_data[raw_vaccine_data["State"] == "India"].index, inplace=True)

raw_vaccine_data.shape
```

```
raw_vaccine_data.head(5)
```

Here, we can see that our index has changed in the dataframe so lets reset our index using reset_index method

```
vaccine_data = raw_vaccine_data.reset_index(drop=True)
vaccine_data.head(5)
```

Now, our index has changed and our data has also cleaned up. We can now do real time analysis in the data set.

Exploratory Analysis and Visualization

In this section, we will explore relationships between columns by doing visualization using matplotlib and seaborn library in python.

States

Let's look at the total number of people that have vaccinated in different parts of the country from January till now.

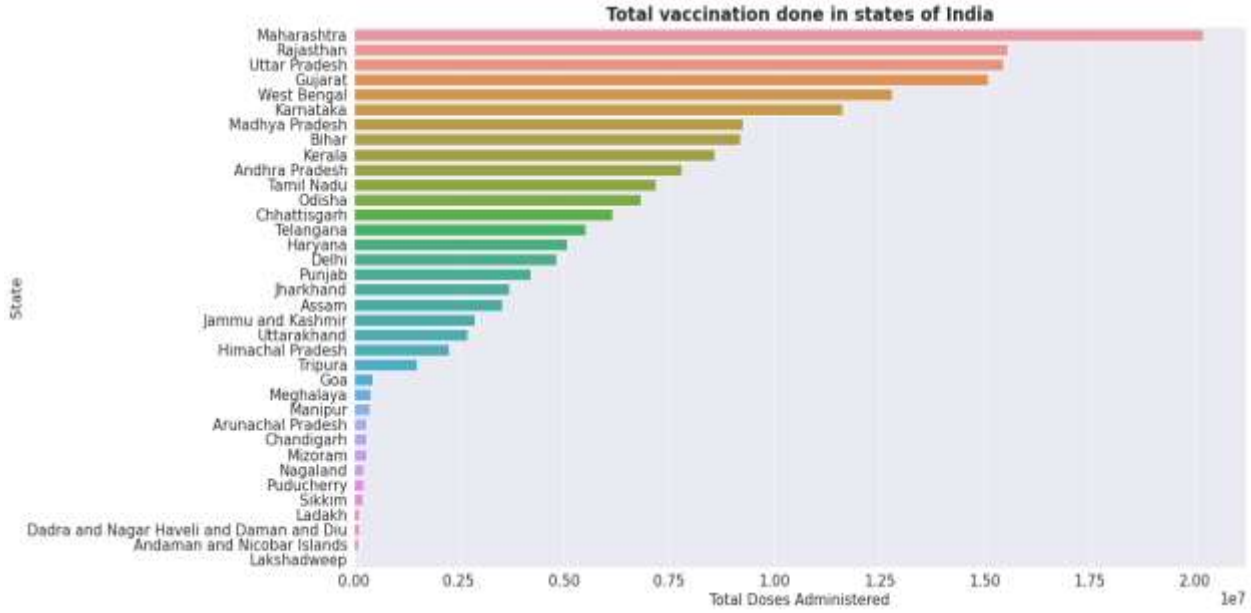
	State	Till year 2021
0	Uttar Pradesh	240000000.0
1	Maharashtra	124000000.0
2	Bihar	127000000.0
3	West Bengal	100000000.0
4	Madhya Pradesh	87000000.0

So, let's calculate values of total vaccination done in unique states by using groupby method in pandas.

```
%matplotlib inline
sns.set_style('darkgrid')
matplotlib.rcParams['font.size'] = 16
```

```
vaccination = vaccine_data.groupby("State")[["Total Doses
Administered"]].max().sort_values("Total Doses Administered",ascending=False)
```

```
plt.figure(figsize=(18,10))
plt.title("Total vaccination done in states of India", fontweight="bold")
sns.barplot(x=vaccination["Total Doses Administered"], y=vaccination.index);
```



From above graph, we can find that Maharashtra is the maximum vaccinated state in India. But it can also be due to the large number of population in Maharashtra. As the population is large and area of the state is also big so may be the vaccination centers are more. Because of this, it can be the maximum vaccinated state. So, to get the accurate percentage of people vaccinated in a particular state we need to find the total population in the state and according to that how many people are vaccinated in a particular area.

Population

To get the percentage of people vaccinated, we have to load the population data set to get the population of different states. So let's load it first.

```
population = pd.read_csv("population.csv")
population.head(5)
```

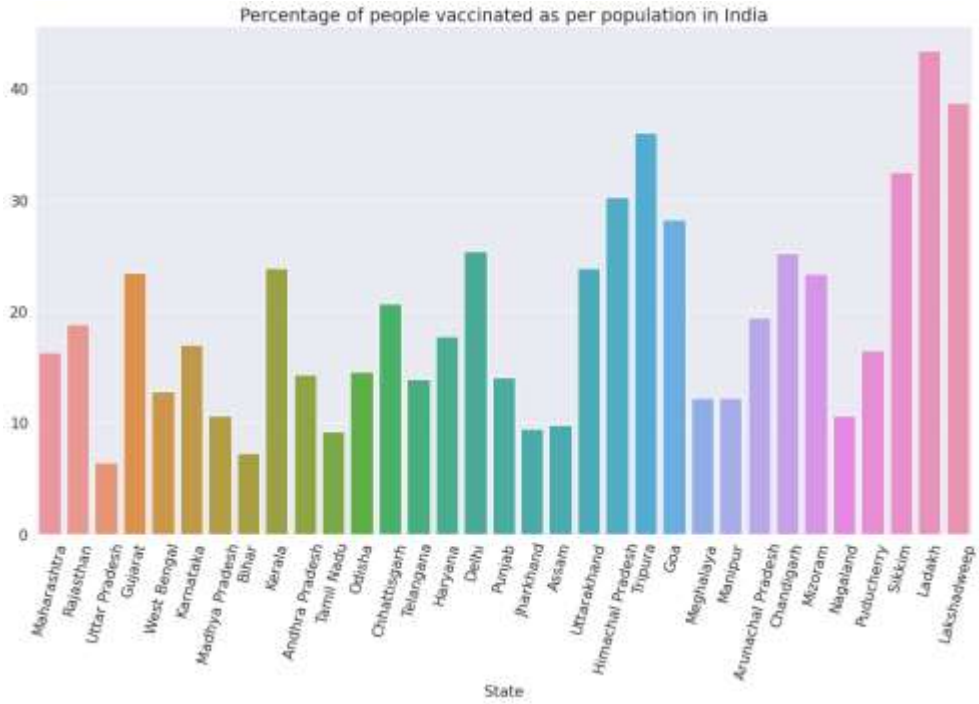
	State	Till year 2021
0	Uttar Pradesh	240000000.0
1	Maharashtra	124000000.0
2	Bihar	127000000.0
3	West Bengal	100000000.0
4	Madhya Pradesh	87000000.0

Now, we have to merge our vaccination data with population data to find the percentage of vaccination.

```
total_vaccination = pd.merge(vaccination,population,on="State")
total_vaccination.head(5)
```

	State	Total Doses Administered	Till year 2021
0	Maharashtra	20201111	124000000.0
1	Rajasthan	15544817	82500000.0
2	Uttar Pradesh	15436461	240000000.0
3	Gujarat	15086347	64300000.0
4	West Bengal	12805341	100000000.0

```
plt.figure(figsize=(18,10))

plt.xticks(rotation=75)

plt.title("Percentage of people vaccinated as per population in India")

vaccine_percent = total_vaccination["Total Doses Administered"] *100/total_vaccination["Till year 2021"]

sns.barplot(x=total_vaccination.State, y=vaccine_percent);
```



Now, we can see from above graph that Ladakh is the maximum vaccinated state as per the population in India. Earlier Maharashtra was there due to the large number of population. Now Maharashtra is only 15% vaccinated while Ladakh is 45% vaccinated.

Gender

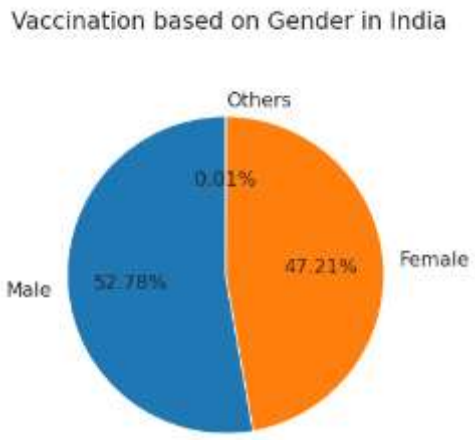
Let's look at the people vaccinated on the basis of gender in the country. We know that women and other binary genders are not given much privilege in our society. But vaccination is the most important thing that has to be done to each and every person in the society for the measure of health and safety.

Let's calculate the total number of male, female and other binary genders vaccinated in the country.

```
male = vaccine_data.groupby("State")[["Male(Individuals Vaccinated)"]].max().sum()
female = vaccine_data.groupby("State")[["Female(Individuals Vaccinated)"]].max().sum()
others = vaccine_data.groupby("State")[["Transgender(Individuals Vaccinated)"]].max().sum()
```

```
plt.axis("equal")
plt.title(" \n Vaccination based on Gender in India\n\n\n\n")
plt.pie([male[0],female[0],others[0]],
```

```
labels=["Male","Female","Others"],
autopct="%0.2f%%",
startangle=90,
radius=1.5);
```



From above chart, we can conclude that there is not much difference between the male and female vaccination but percent of others are very less due to their less number of population in the country.

Find that how many number of people are partially vaccinated and fully vaccinated in each state ?

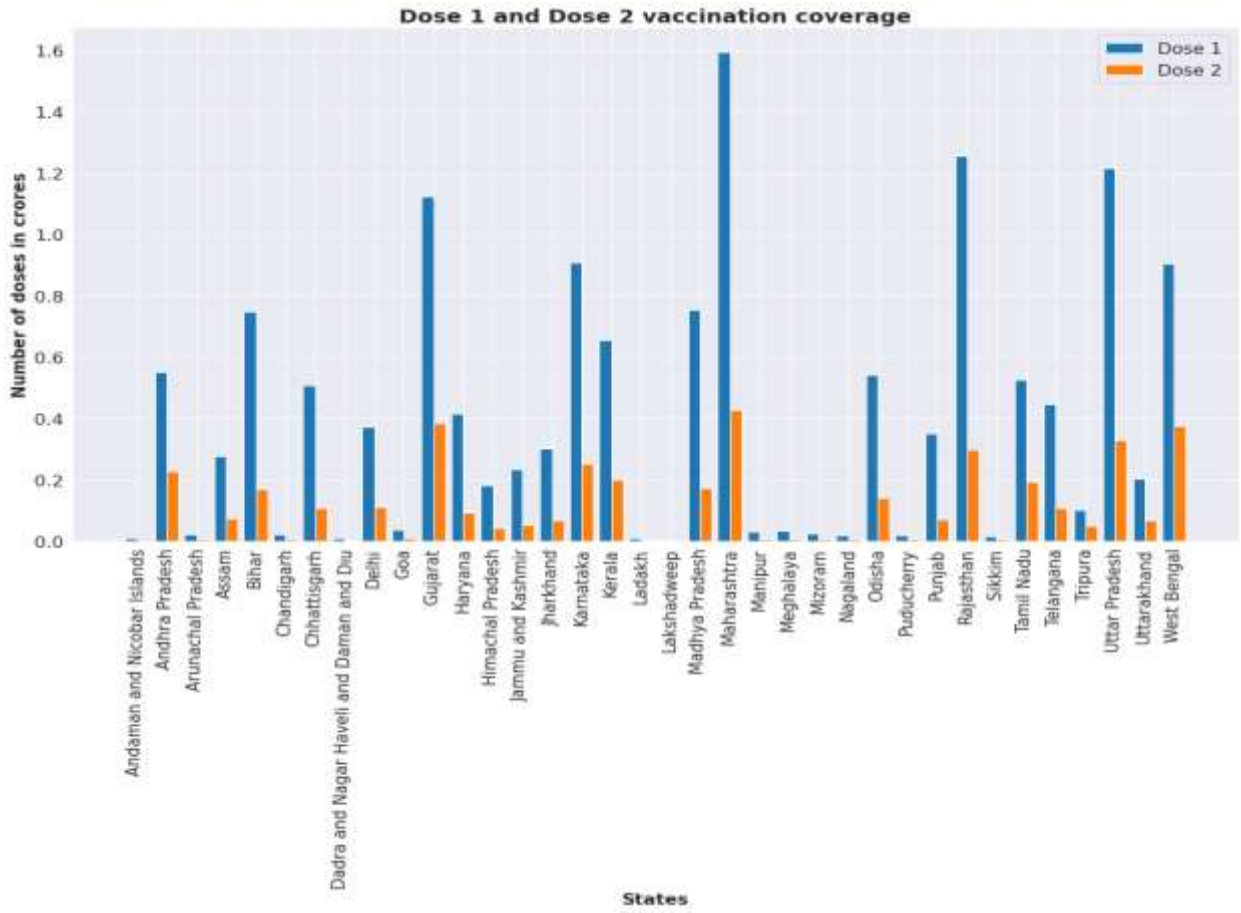
Here partially vaccinated means people vaccinated with first dose and fully vaccinated means people vaccinated with both first and second dose.

To calculate this, we have to find the maximum number of first and second dose in each state by using groupby() and max() function in pandas.

```
doses = vaccine_data.groupby("State")[["First Dose Administered","Second Dose Administered"]].max()
```

```
plt.figure(figsize=(18,10))
```

```
xpos = np.arange(len(doses.index))
plt.xticks(xpos,doses.index,rotation=89)


plt.bar(xpos-0.2,doses["First Dose Administered"]/1e7, label="Dose 1",width=0.4)
plt.bar(xpos+0.2,doses["Second Dose Administered"]/1e7, label="Dose 2",width=0.4)


plt.xlabel("States",fontweight="bold")
plt.ylabel("Number of doses in crores",fontweight="bold")
plt.title("Dose 1 and Dose 2 vaccination coverage",fontweight="bold")

plt.legend();
```



From above graph, we can see that in each state number of first dose is high as compared to second dose because second dose can be taken after 28 days of first dose. So it means that when people those have taken first dose will take second dose, the bar of second dose will become high.

Conclusion :

We have concluded so much from this analysis. Here is the summary of few of them:-

- We infer that maximum vaccination is done in the Maharashtra but as compared to population, Ladakh has the highest percentage of people that vaccinated in last 5 months.
- Vaccination is done unbiased in case of gender. Men, women and other binary gender are actively participating in this vaccination campaign. This shows that people in India are concern about their safety and are getting vaccinated.
- Then we have seen that number of first dose is very high than second dose in each state which means that people have started taking their first dose and after some time they will go for second one also.

Hence we successfully performed analytics on given dataset.

Questions

1. Which tools are used in this analysis ?
2. What do we do in Exploratory Analysis and Visualization ?
3. Name all the libraries of python that are used in this analysis ?
4. How we can find the number of people who are partially vaccinated and fully vaccinated in each state ?

Group C			
Assignment No 15	**Mini Project / Case Study**	**Page No**	

Aim : Write a case study to process data driven for Digital Marketing.

Objectives : To understand different concepts of Digital Marketing.

Outcomes : Students are able to understand various concepts of Digital Marketing.

Theory :

Biggest Marketing Challenges for Small Businesses

1. **Lack of Resources :**

Among the numerous small businesses that fail annually, about 50% report a lack of resources and working capital as the main reason for failure. Typically, savvy business owners are already aware of the costs of doing business and how much money is required to maintain business operations on a day-to-day basis. This would entail knowledge of fixed and variable overhead costs like rent and utilities as well as payment for external vendors.
But not all businesses are capable of sustaining their operations**,** especially when owners are less in tune with the business, its operations, and the amount of revenue generated by the sale of its products and/or services. Ultimately, this results in cash flow problems. Business Insider reports that 82% of small businesses fail simply because they have problems with cash flow. This results in poor decisions such as competitive product pricing and ineffective marketing practices in an effort to rid the business of its cash flow.

1. **Increasing Visibility :**

Data gained from the Small Business Administration (SBA) provides insight into the survival rate of small businesses during their early years. It states that nearly 66% of small businesses do survive during the first two years. This means only about 34% actually fail during the first two crucial years. But according to the Bureau of Labor Statistics Business Employment Dynamics, only 50% of businesses will survive their year fifth, with only 30% surviving their tenth year.
The data provided all points to a monumental problem faced by all small businesses, which is the need to be visible. And for small businesses, this is a drastic need in order to improve sales

and thrive. In order to gain visibility, businesses have to spend more money something that is difficult to accomplish when operating on a shoestring budget.

2. **Generating Quality Leeds :**

Lead generation is the lifeline of any business. It is also quite simple to create and execute effectively in theory. However, only 10% of small businesses can honestly say that their lead generation efforts are highly efficient and effective. 61% blame the lack of funding and a dedicated sales team.

What most small businesses fail to account for is that the market is constantly evolving. Consequently, they fail to adapt and end up using outdated lead generation tactics for the current state of the market.

3. **Choosing & Navigating the right social media platform :**

Social media statistics gathered from Emarsys suggest that there are already 3.2 billion people using social media around the world, roughly 42% of the entire world population and is still growing. To take advantage of that fact, SMBs venture into social media hoping to reach a good percentage of those users. However, a fairly recent survey by Manta revealed that among small businesses, 59% reported that they had no return on their investment.

Despite the massive failure in their social media efforts, nearly 92% of small businesses still planned to increase their investment in social media in 2018. This could lead to massive losses due to poor social media marketing planning.

4. **Producing & Delivering High-Quality Content :**

High-quality content is not considered the currency of social media. Quality, engaging content allows businesses to capture and keep an audience's attention by offering something of value to the reader. But producing content is just the easy part. Creating content that converts is the more challenging part.

Quality content requires time and skill. And as the one who truly understands the nature of the business, most business owners prefer to take care of their content marketing needs on their own. But as most business owners also know, producing quality content on a consistent basis eats up too much time – time that could have been spent actually running the business. This creates a big dilemma for business owners, whether they invest their own time to create content, or invest actual money and pay for someone else to do it.

Strategic Planning for Better Results :

1. Most challenges faced by small businesses are tactical in nature, which only requires a sound strategic plan to solve. In fact, a solid strategic marketing plan allows business owners to guide their efforts based on different resource constraints such as time and budget and align their strategies with their goal.

2. Without a good strategic plan, many opportunities will be missed due to several reasons like the business isn't ready, not enough cash flow, or the lack of skill. In other times, without a clear strategic plan, each member of the team such as a web developer, marketer, or programmer will have their own vision and work towards achieving their own goals. Either way, the business suffers.

3. Originally, a strategic plan was thought of as a lengthy, planned document containing the business's plans over an extended period of time. It serves to map out the goals of the business, outlining the company's mission statement, vision, and objectives. It contains necessary plans to support the business and its reason for existence.

Benefits of Data-Driven Digital Marketing Personalization

1. **Better Audience Insight**

Big data enables businesses to trace a consumer's digital footprint to gain valuable information about every interaction on the web, whether it's the amount of time spent on a specific page or clicking on a specific offer. This data will provide the necessary behavioral information needed to craft the best action plan for communication.

2. **Audience Segmentation**

The act of segmenting audiences is not an entirely new concept. In fact, the idea has been around and used by businesses all over the world for decades now. It was usually performed without distinguishing between customers that are profitable and those that are not, which cost businesses millions of dollars in advertising costs with little ROI. It also focused on a retrospective look at customer behavior and preferences, producing unreliable results.

3. **Consumer Pre-Targeting**

While a large chunk of online marketing and targeting techniques are built on already established data for retargeting, a greater ROI can be gained when pre-targeting future clients.

Pre-targeting is an online technique used in marketing to target prospective clients using a profile of the ideal customer.

Big Data Brings Big Business

Amazon

- When faced with a massive selection of potential choices, users end up feeling overwhelmed, and ultimately decide to forgo the process. This is often the case for companies engaging in e-commerce. But none feel the brunt of this psychological phenomenon more than Amazon. With its "everything under one roof" model, potential customers often become too undecided while visiting the platform, which makes it even harder to gather insight on their buying behavior simply because no purchase was made.

- To stop this from happening, Amazon decided to gather data not just on purchase behavior but on other parts of the sales experience, as well. The e- commerce giant utilizes a collaborative filtering engine that analyzes a customer's buying habits based on their previous purchases, items inside their cart, wish list, and products viewed, reviewed, or searched for the most.

- Through big data, Amazon is able to analyze buyer behavior and create an individual persona they can use for marketing as well as predict the customer's preferences based on their perceived traits. The buyer, ultimately, is influenced to make a purchase by limiting the number of purchase decisions they have to make.

Data-Driven Results

1. **Goal: Significantly bring down cost per lead.**

Our Strategy

We employ many different strategies to achieve this goal, but first, we talk to our clients about the nature of the business to identify their clientele and see who they want to target. Discussions about past and current marketing strategies are welcome to give us more data about the actions that we can take.

We collect pixel data obtained from Facebook, Google, or Pinterest pixels to set up a retargeting system designed to guide customers along the funnel until they ultimately convert. Those lost in the funnel will be recaptured and guided back to the funnel. We do this by testing the demographic and separating them based on interest, optimizing for either

mobile or desktop audiences. We set up a criteria against which a particular ad set can be judged. Ad sets that perform below the criteria set are turned off.

Results

The entire process of refining the search for better targeting and retargeting brings down the total cost per lead by as much as 88%. An ad set with an original CPL of RS 239 can be brought down to as low as RS 29 CPL.

2. Goal: Optimize the funnel to maximize conversions and boost sales.

First, we define the type of funnel you want your customers to go through. As defined by Google Analytics, there are two types of funnels, namely a sales funnel and a goal funnel, with both having multi-channel variants. Goal funnels are used mainly for non-transactional goals such as newsletter signups or to join a webinar. Sales funnels, on the other hand, are transactional in nature, with sales as the end goal.

For this discussion, we use a sales funnel as an example. We begin by dividing the funnel into three major stages, which is the lead generation stage, the conversion stage, and the nurture stage.

Optimizing the funnel can yield amazing results. First, we identified that organic SEO strategies are 5.66 times better than paid search ads due to the sheer amount of traffic generated while keeping costs to a minimum, generating 10x better ROI.

Using a targeting and retargeting strategy, we were able to bring down cost per acquisition by as much as 57% and conversion rates of up to 12% – 6% more than the industry average.

Conclusion :

Hence we successfully understood the concepts of process data driven for Digital Marketing.

Questions

1. What is Digital Marketing ?
2. What are the benefits of Digital Marketing ?
3. Explain one example of Big Business.
4. Write advantages and disadvantages of Digital Marketing.