**Dr. D. Y. Patil Pratishthan's**

**DR. D. Y. PATIL INSTITUTE OF ENGINEERING, MANAGEMENT & RESEARCH**

**Approved by A.I.C.T.E, New Delhi , Maharashtra State Government, Affiliated to Savitribai Phule Pune University**
Sector No. 29, PCNTDA , Nigidi Pradhikaran, Akurdi, Pune 411044. Phone: 020–27654470, Fax: 020-27656566
Website :www.dypiemr.ac.in Email : principal.dypiemr@gmail.com


---

# Department of Computer Engineering
# LAB MANUAL

# Data Science and Big Data Analytics
# (Third Year Computer Engineering)
# Semester II


## Prepared By:

**Mrs. Nalini Jagtap**
**Ms. Tejaswini Patil**


# Data Science and Big Data Analytics Laboratory

| Course Code | Course Name | Teaching Scheme(Hrs./ Week) | Credits |
|---|---|---|---|
| 310256 | Data Science and Big Data Analytics Laboratory | 04 | 02 |

## Course Objectives:

- To understand principles of Data Science for the analysis of real time problems
- To develop in depth understanding and implementation of the key technologies in Data Science and Big Data Analytics
- To analyze and demonstrate knowledge of statistical data analysis techniques for decision-making
- To gain practical, hands-on experience with statistics programming languages and Big Data tools

## Course Outcomes:

CO1: Apply principles of Data Science for the analysis of real time problems
CO2: Implement data representation using statistical methods
CO3: Implement and evaluate data analytics algorithms
CO4: Demonstrate text preprocessing
CO5: Implement data visualization techniques
CO6: Use cutting edge tools and technologies to analyze Big Data

The instructor is expected to frame the assignments by understanding the prerequisites, technological aspects, utility and recent trends related to the topic. The assignment framing policy need to address the average students and inclusive of an element to attract and promote the intelligent students. Use of open source software is encouraged. Based on concepts learned. Instructor may also set one assignment or mini-project that is suitable to respective branch beyond the scope of syllabus.

Set of suggested assignment list is provided in groups- A and B. Each student must perform 13 assignments (10 from group A, 3 from group B), 2 mini projects from Group C

Operating System recommended:- 64-bit Open source Linux or its derivative
Programming tools recommended: - JAVA/Python/R/Scala

# Table of Contents

| | | | |
|---|---|---|---|
| 3. | Descriptive Statistics - Measures of Central Tendency and variability Perform the following operations on any open source dataset (e.g., data.csv)<br><br>1. Provide summary statistics (mean, median, minimum, maximum, standard deviation) for a dataset (age, income etc.) with numeric variables grouped by one of the qualitative (categorical) variables. For example, if your categorical variable is age groups and quantitative variable is income, then provide summary statistics of income grouped by the age groups. Create a list that contains a numeric value for each response to the categorical variable.<br><br>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. | CO2 | |
| 4. | **Data Analytics I**<br>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 samplesand 14 feature variables in this dataset. | CO2 | |
| 5. | **Data Analytics II**<br>1. Implement logistic regression using Python/R to perform classification on Social_Network_Ads.csv dataset.<br>2. Compute Confusion matrix to find TP, FP, TN, FN, Accuracy, Error rate, Precision, Recall on the given dataset. | CO2 | |
| 6. | **Data Analytics III**<br><br>1. Implement Simple Naïve Bayes classification algorithm using Python/R on iris.csv dataset.<br><br>2. Compute Confusion matrix to find TP, FP, TN, FN, Accuracy, Error rate, Precision, Recall on the given dataset. | CO3 | |
| 7 | **Text Analytics**<br><br>1. Extract Sample document and apply following document preprocessing methods: Tokenization, POS Tagging, stop words removal, Stemming and Lemmatization.<br><br>2. Create representation of documents by calculating Term Frequency and Inverse Document Frequency. | CO4 | |

| | | | |
|---|---|---|---|
| 8 | **Data Visualization I**<br><br>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.<br><br>2. Write a code to check how the price of the ticket (column name: 'fare') for each passenger is distributed by plotting a histogram. | CO5 | |
| 9 | **Data Visualization II**<br><br>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')<br><br>2. Write observations on the inference from the above statistics. | CO5 | |
| 10 | **Data Visualization III**<br><br>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:<br><br>1. List down the features and their types (e.g., numeric, nominal) available in the dataset.<br><br>2. Create a histogram for each feature in the dataset to illustrate the feature distributions.<br><br>3. Create a boxplot for each feature in the dataset.<br><br>4. Compare distributions and identify outliers. | CO5 | |
| | **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. | CO6 | |
| 12 | Design a distributed application using MapReduce which processes a log file of a system. | CO6 | |
| 13 | Locate dataset (e.g., sample_weather.txt) for working on weather data which reads the textinput files and finds average for temperature, dew point and wind speed. | CO6 | |

# Lab Assignment 1

## Title: Data Wrangling I

**PROBLEM STATEMENT:**

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 dataframe.
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.

## THEORY:

## What is Data Wrangling?

Data Munging, commonly referred to as Data Wrangling, is the cleaning and transforming of one type of data to another type to make it more appropriate into a processed format. Data wrangling involves processing the data in various formats and analyzes and get them to be used with another set of data and bringing them together into valuable insights. It further includes data aggregation, data visualization, and training statistical models for prediction. data wrangling is one of the most important steps of the data science process. The quality of data analysis is only as good as the quality of data itself, so it is very important to maintain data quality.

## NEED FOR WRANGLING:

Wrangling the data is crucial, yet it is considered as a backbone to the entire analysis part. The main purpose of data wrangling is to make raw data usable. In other words, getting data into a shape. 0n average, data scientists spend 75% of their time wrangling the data, which is not a surprise at all. The important needs of data wrangling include,

- The quality of the data is ensured.
- Supports timely decision-making and fastens data insights.
- Noisy, flawed, and missing data are cleaned.
- It makes sense to the resultant dataset, as it gathers data that acts as a preparation stage for the data mining process.

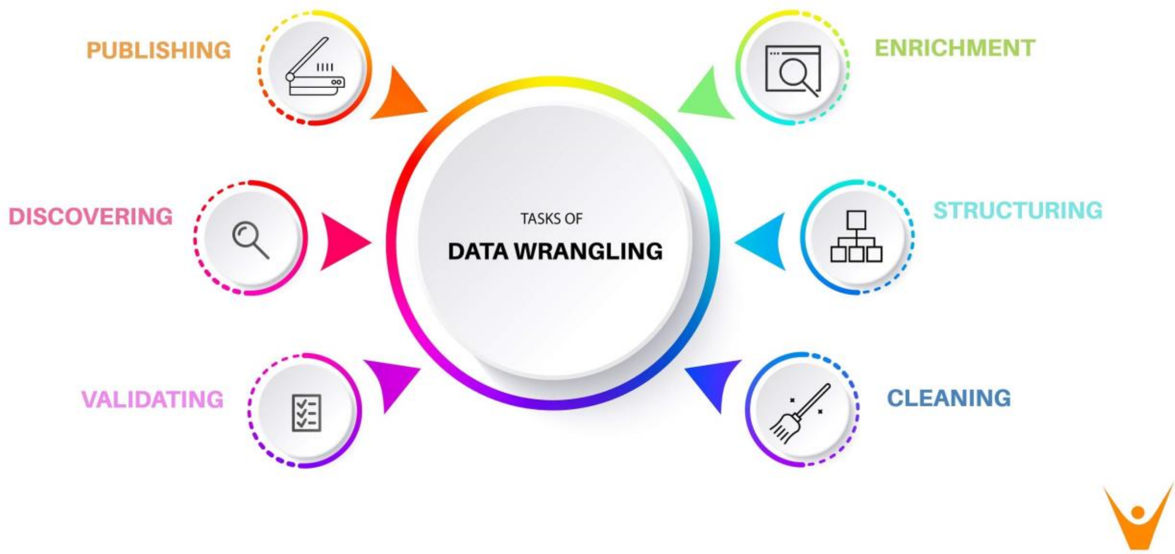
- Helps to make concrete and take a decision by cleaning and structuring raw data into the required format.

- Raw data are pieced together to the required format.

- To create a transparent and efficient system for data management, the best solution is to have all data in a centralized location so it can be used in improving compliance.

- Wrangling the data helps make decisions promptly and helps the wrangler clean, enrich, and transform the data into a perfect picture.

**DATA WRANGLING STEPS:**




**1. DISCOVERING:**

Discovering is a term for an entire analytic process, and it's a good way to learn how to use the data to explore and it brings out the best approach for analytics explorations. It is a step in which the data is to be understood more deeply.

**2. STRUCTURING:**

Raw data is given randomly. There will not be any structure to it in most cases because raw data comes from many formats of different shapes and sizes. The data must be organized in such a manner where the analytics attempt to use it in his analysis part.

**3. CLEANING:**

High-quality analysis happens here where every piece of data is checked carefully and redundancies are removed that don't fit the data for analysis. Data containing the Null values have to be changed either to an empty string or zero and the formatting will be standardized to make the data of higher quality. The goal of data cleaning or remediation is to ensure that there are no possible ways that the final data could be influenced that is to be taken for final analysis.

**4. ENRICHING:**

Enriching is like adding some sense to the data. In this step, the data is derived into new kinds of data from the data which already exits from cleaning into the formatted manner. This is where the data need to strategize that you have in your hand and to make sure that you have is the best-enriched data. The best way to get the refined data is to down sample, upscale it, and finally augur the data.

**5. VALIDATING:**

For analysis and evaluation of the quality of specific data set data quality rules are used. After processing the data, the quality and consistency are verified which establish a strong surface to the security issues. These are to be conducted along multiple dimensions and to adhere to syntactic constraints.

**6. PUBLISHING:**

The final part of the data wrangling is Publishing which gives the sole purpose of the entire wrangling process. Analysts prepare the wrangled data that use further down the line that is its purpose after all. The finalized data must match its format for the eventual data's target. Now the cooked data can be used for analytics.

**DATA WRANGLING IN PYTHON:**

Pandas are an open-source mainly used for Data Analysis. Data wrangling deals with the following functionalities.

- **Data exploration:** Visualization of data is made to analyze and understand the data.

- **Dealing with missing values:** Having Missing values in the data set has been a common issue when dealing with large data set and care must be taken to replace them. It can be replaced either by mean, mode or just labelling them as NaN value.

- **Reshaping data:** Here the data is either modified from the addressing of pre-existing data or the data is modified and manipulated according to the requirements.

- **Filtering data:** The unwanted rows and columns are filtered and removed which makes the data into a compressed format.

- **Others:** After making the raw data into an efficient dataset, it is bought into useful for data visualization, data analyzing, training the model, etc.

**How is Data Preprocessing performed?**

Data Preprocessing is carried out to remove the cause of unformatted real-world data which we discussed above. First of all, let's explain how missing data can be handled during Data Preparation. Three different steps can be executed which are given below -

- **Ignoring the missing record -** It is the simplest and efficient method for handling the missing data. But, this method should not be performed at the time when the number of missing values is immense or when the pattern of data is related to the unrecognized primary root of the cause of the statement problem.

- **Filling the missing values manually -** This is one of the best-chosen methods of Data Preparation process. But there is one limitation that when there are large data set, and missing values are significant then, this approach is not efficient as it becomes a time-consuming task.

- **Filling using computed values -** The missing values can also be occupied by computing mean, mode or median of the observed given values. Another method could be the predictive values in Data Preprocessing are that are computed by using any Machine Learning or Deep Learning tools and algorithms. But one drawback of this approach is that it can generate bias within the data as the calculated values are not accurate concerning the observed values.

## Data Formatting

- **Incorrect data types**

We should make sure that every column is assigned to the correct data type. This can be checked through the property dtypes.

df.dtypeswhich gives the following output:

```
Tweet Id                    object
Tweet URL                   object
Tweet Posted Time (UTC)     object
Tweet Content               object
Tweet Type                  object
Client                      object
Retweets Received            int64
Likes Received               int64
Tweet Location              object
Tweet Language              object
User Id                     object
Name                        object
Username                    object
User Bio                    object
Verified or Non-Verified    object
Profile URL                 object
Protected or Non-protected  object
User Followers               int64
User Following               int64
User Account Creation Date  object
Impressions                  int64
dtype: object
```

We can convert the column Tweet Location to string by using the function astype() as follows:

```
df['Tweet Location'] = df['Tweet Location'].astype('string')
```

## Data Normalization with Pandas

Data Normalization could also be a typical practice in machine learning which consists of transforming numeric columns to a standard scale. In machine learning, some feature values differ from others multiple times. The features with higher values will dominate the learning process.

Data Normalization involves adjusting values measured on different scales to a common scale.

Normalization applies only to columns containing numeric values. Normalization methods are:

- Simple feature scaling
- min max
- z-score

### Min-Max scaling

$$X_{norm} = \frac{X - X_{min}}{X_{max} - X_{min}}$$

### Z-score normalization

$Z = (x - \mu) / \sigma$

### Simple feature scaling

$$x_{new} = \frac{x_{old}}{x_{max}}$$

### Convert Categorical Variable to Numeric

When we look at the categorical data, the first question that arises to anyone is how to handle those data, because machine learning is always good at dealing with numeric values. We could make machine learning models by using text data. So, to make predictive models we have to convert categorical data into numeric form.

**Method 1: Using replace() method**

Replacing is one of the methods to convert categorical terms into numeric. For example, We will take a dataset of people's salaries based on their level of education. This is an ordinal type of categorical variable. We will convert their education levels into numeric terms.

***Syntax:***

***replace(to_replace=None, value=None, inplace=False, limit=None, regex=False, method='pad')***

**Method 2: Using get_dummies() / One Hot Encoding**

Replacing the values is not the most efficient way to convert them. Pandas provide a method called get_dummies which will return the dummy variable columns.

***Syntax: pandas.get_dummies(data, prefix=None, prefix_sep='_', dummy_na=False, columns=None, sparse=False, drop_first=False, dtype=None)***

**One-Hot Encoding: The Standard Approach for Categorical Data**

One hot encoding is the most widespread approach, and it works very well unless your categorical variable takes on a large number of values One hot encoding creates new (binary) columns, indicating the presence of each possible value from the original data.**It uses get_dummies() Method**

| Color |
|---|
| Red |
| Red |
| Yellow |
| Green |
| Yellow |

| Red | Yellow | Green |
|---|---|---|
| 1 | 0 | 0 |
| 1 | 0 | 0 |
| 0 | 1 | 0 |
| 0 | 0 | 1 |

**Method 3:**

**Label Encoding** refers to converting the labels into a numeric form so as to convert them into the machine-readable form. Machine learning algorithms can then decide in a better way how those labels must be operated. It is an important pre-processing step for the structured dataset in supervised learning.

**Example:**
Suppose we have a column *Height* in some dataset.

| Height |
| --- |
| Tall |
| Medium |
| Short |

| Height |
| --- |
| 0 |
| 1 |
| 2 |

After applying label encoding, the Height column is converted into: where 0 is the label for tall, 1 is the label for medium, and 2 is a label for short height.

Example :# Import dataset

# Import label encoder

from sklearn import preprocessing

# label_encoder object knows how to understand word labels.

label_encoder = preprocessing.LabelEncoder()

# Encode labels in column Height.

df['Height']= label_encoder.fit_transform(df[Height'])

df['Height'].unique()

## Procedure-

### STEP 1: IMPORTING THE LIBRARIES

IMPORT NUMPY AS NP

IMPORT MATPLOTLIB.PYPLOT AS PLT

IMPORT PANDAS AS PD

### STEP 2: IMPORT THE DATASET

PATH="C:/USERS/ADMIN/DESKTOP/DYPIEMR DATA/DSBDA LAB/WRANGLED_DATA.CSV"

DF= PD.READ_CSV(PATH)

PRINT(DF)

### STEP 3:DATA PREPROCESSING: CHECK FOR MISSING VALUES IN THE DATA USING PANDAS ISNULL()

DF.ISNULL()

DF

## STEP 4: #DESCRIBE() FUNCTION TO GET SOME INITIAL STATISTICS

DF.DESCRIBE()

#CHECK THE DIMENSIONS OF THE DATA FRAME

DF.SHAPE

#TOTAL NUMBER OF ELEMENTS IN THE DATAFRAME

DF.SIZE

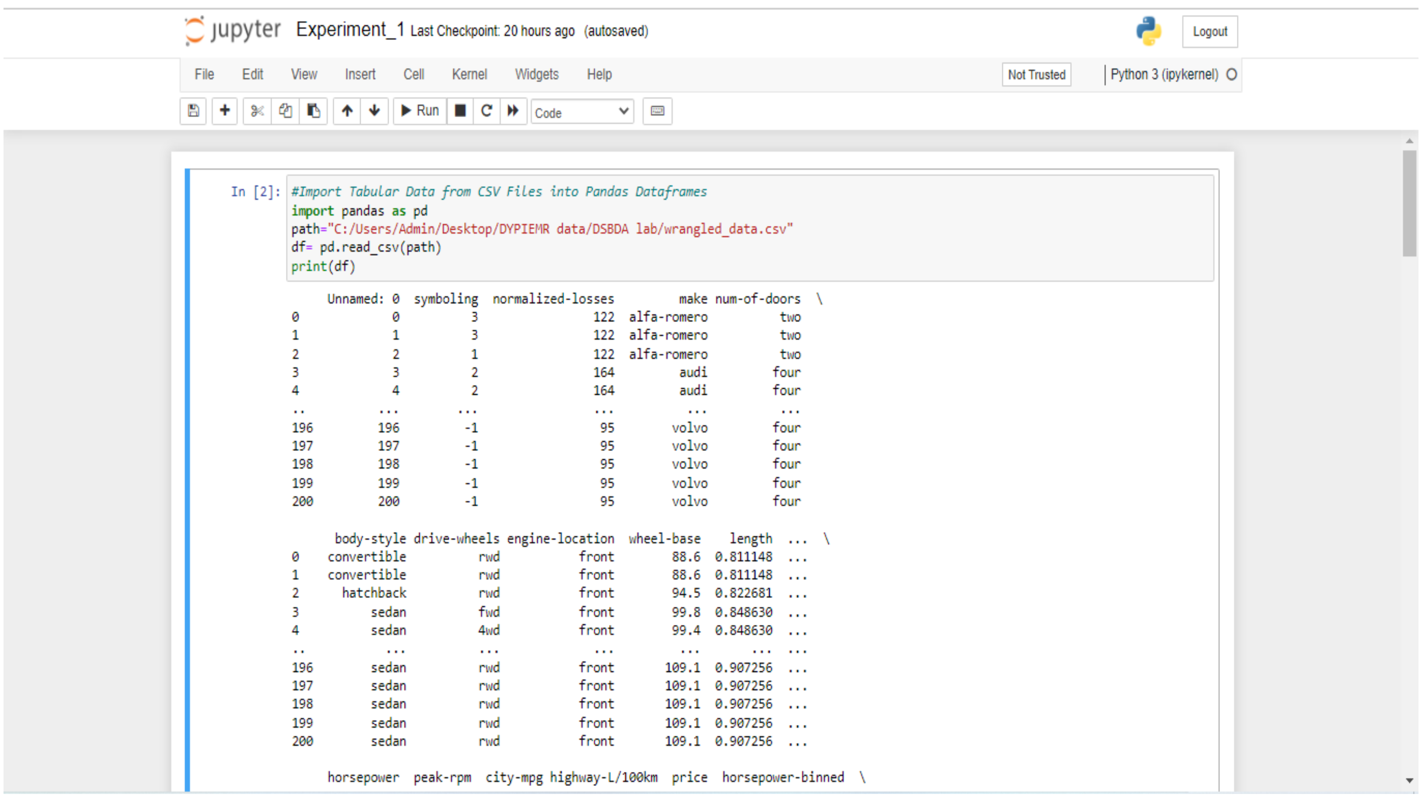
## STEP 5: DATA FORMATTING

DF.DTYPES

DF.ASTYPES(“COLUMN_NAME”)

DF = DF.ASTYPE({"ENGINE-LOCATION":'CATEGORY', "  HORSEPOWER":'INT64'})

## PROGRAM :





In [2]:

```
#Import Tabular Data from CSV Files into Pandas Dataframes
import pandas as pd
path="C:/Users/Admin/Desktop/DYPIEMR data/DSBDA lab/wrangled_data.csv"
df= pd.read_csv(path)
print(df)
```

```
     Unnamed: 0  symboling  normalized-losses         make num-of-doors  \
0             0          3                122  alfa-romero          two
1             1          3                122  alfa-romero          two
2             2          1                122  alfa-romero          two
3             3          2                164         audi         four
4             4          2                164         audi         four
..          ...        ...                ...          ...          ...
196         196         -1                 95        volvo         four
197         197         -1                 95        volvo         four
198         198         -1                 95        volvo         four
199         199         -1                 95        volvo         four
200         200         -1                 95        volvo         four

      body-style drive-wheels engine-location  wheel-base    length  ...  \
0    convertible          rwd           front        88.6  0.811148  ...
1    convertible          rwd           front        88.6  0.811148  ...
2      hatchback          rwd           front        94.5  0.822681  ...
3          sedan          fwd           front        99.8  0.848630  ...
4          sedan          4wd           front        99.4  0.848630  ...
..           ...          ...             ...         ...       ...  ...
196        sedan          rwd           front       109.1  0.907256  ...
197        sedan          rwd           front       109.1  0.907256  ...
198        sedan          rwd           front       109.1  0.907256  ...
199        sedan          rwd           front       109.1  0.907256  ...
200        sedan          rwd           front       109.1  0.907256  ...

     horsepower  peak-rpm  city-mpg highway-L/100km  price  horsepower-binned  \
```

jupyter Experiment_1 Last Checkpoint: 20 hours ago (autosaved)

Logout

File Edit View Insert Cell Kernel Widgets Help

Not Trusted | Python 3 (ipykernel)

In [3]:
```
#Data Preprocessing: check for missing values in the data using pandas isnull()
df.isnull()
df
```

Out[3]:

| | Unnamed: 0 | symboling | normalized-losses | make | num-of-doors | body-style | drive-wheels | engine-location | wheel-base | length | ... | horsepower | peak-rpm | city-mpg | highway-L/100km | price | horsep bi |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 3 | 122 | alfa-romero | two | convertible | rwd | front | 88.6 | 0.811148 | ... | 111 | 5000.0 | 21 | 8.703704 | 13495 | |
| 1 | 1 | 3 | 122 | alfa-romero | two | convertible | rwd | front | 88.6 | 0.811148 | ... | 111 | 5000.0 | 21 | 8.703704 | 16500 | |
| 2 | 2 | 1 | 122 | alfa-romero | two | hatchback | rwd | front | 94.5 | 0.822681 | ... | 154 | 5000.0 | 19 | 9.038462 | 16500 | Me |
| 3 | 3 | 2 | 164 | audi | four | sedan | fwd | front | 99.8 | 0.848630 | ... | 102 | 5500.0 | 24 | 7.833333 | 13950 | |
| 4 | 4 | 2 | 164 | audi | four | sedan | 4wd | front | 99.4 | 0.848630 | ... | 115 | 5500.0 | 18 | 10.681818 | 17450 | |
| ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | |
| 196 | 196 | -1 | 95 | volvo | four | sedan | rwd | front | 109.1 | 0.907256 | ... | 114 | 5400.0 | 23 | 8.392857 | 16845 | |
| 197 | 197 | -1 | 95 | volvo | four | sedan | rwd | front | 109.1 | 0.907256 | ... | 160 | 5300.0 | 19 | 9.400000 | 19045 | Me |
| 198 | 198 | -1 | 95 | volvo | four | sedan | rwd | front | 109.1 | 0.907256 | ... | 134 | 5500.0 | 18 | 10.217391 | 21485 | Me |
| 199 | 199 | -1 | 95 | volvo | four | sedan | rwd | front | 109.1 | 0.907256 | ... | 106 | 4800.0 | 26 | 8.703704 | 22470 | |
| 200 | 200 | -1 | 95 | volvo | four | sedan | rwd | front | 109.1 | 0.907256 | ... | 114 | 5400.0 | 19 | 9.400000 | 22625 | |

201 rows × 30 columns

In [4]:
```
df.isnull().sum().sum()
```

Out[4]: 0

jupyter Experiment_1 Last Checkpoint: 20 hours ago (autosaved)

Logout

File Edit View Insert Cell Kernel Widgets Help

Not Trusted | Python 3 (ipykernel)

In [5]:
```
#describe() function to get some initial statistics
df.describe()
```

Out[5]:

| | Unnamed: 0 | symboling | normalized-losses | wheel-base | length | width | height | curb-weight | engine-size | bore | ... | compression-ratio | horsepowe |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| count | 201.000000 | 201.000000 | 201.00000 | 201.000000 | 201.000000 | 201.000000 | 201.000000 | 201.000000 | 201.000000 | 201.000000 | ... | 201.000000 | 201.00000 |
| mean | 100.000000 | 0.840796 | 122.00000 | 98.797015 | 0.837102 | 0.915126 | 0.899108 | 2555.666667 | 126.875622 | 3.330692 | ... | 10.164279 | 103.40298 |
| std | 58.167861 | 1.254802 | 31.99625 | 6.066366 | 0.059213 | 0.029187 | 0.040933 | 517.296727 | 41.546834 | 0.268072 | ... | 4.004965 | 37.36565 |
| min | 0.000000 | -2.000000 | 65.00000 | 86.600000 | 0.678039 | 0.837500 | 0.799331 | 1488.000000 | 61.000000 | 2.540000 | ... | 7.000000 | 48.00000 |
| 25% | 50.000000 | 0.000000 | 101.00000 | 94.500000 | 0.801538 | 0.890278 | 0.869565 | 2169.000000 | 98.000000 | 3.150000 | ... | 8.600000 | 70.00000 |
| 50% | 100.000000 | 1.000000 | 122.00000 | 97.000000 | 0.832292 | 0.909722 | 0.904682 | 2414.000000 | 120.000000 | 3.310000 | ... | 9.000000 | 95.00000 |
| 75% | 150.000000 | 2.000000 | 137.00000 | 102.400000 | 0.881788 | 0.925000 | 0.928094 | 2926.000000 | 141.000000 | 3.580000 | ... | 9.400000 | 116.00000 |
| max | 200.000000 | 3.000000 | 256.00000 | 120.900000 | 1.000000 | 1.000000 | 1.000000 | 4066.000000 | 326.000000 | 3.940000 | ... | 23.000000 | 262.00000 |

8 rows × 21 columns

In [ ]:
```
df.describe(include=['object'])
```

Out[12]:

| | make | num-of-doors | body-style | drive-wheels | engine-location | engine-type | num-of-cylinders | fuel-system | horsepower-binned |
|---|---|---|---|---|---|---|---|---|---|
| count | 201 | 201 | 201 | 201 | 201 | 201 | 201 | 201 | 201 |
| unique | 22 | 2 | 5 | 3 | 2 | 6 | 7 | 8 | 3 |
| top | toyota | four | sedan | fwd | front | ohc | four | mpfi | Low |
| freq | 32 | 115 | 94 | 118 | 198 | 145 | 157 | 92 | 153 |

In [6]:
```
df.dtypes
```

Out[6]: Unnamed: 0 int64

localhost:8888/notebooks/Downloads/Experiment_1.ipynb

jupyter Experiment_1 Last Checkpoint: 20 hours ago (autosaved)

File Edit View Insert Cell Kernel Widgets Help | Not Trusted | Python 3 (ipykernel)

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| top | toyota | four | sedan | fwd | front | ohc | four | mpfi | Low |
| freq | 32 | 115 | 94 | 118 | 198 | 145 | 157 | 92 | 153 |

```
In [6]: df.dtypes
```

```
Out[6]: Unnamed: 0              int64
symboling               int64
normalized-losses       int64
make                   object
num-of-doors           object
body-style             object
drive-wheels           object
engine-location        object
wheel-base            float64
length                float64
width                 float64
height                float64
curb-weight             int64
engine-type            object
num-of-cylinders       object
engine-size             int64
fuel-system            object
bore                  float64
stroke                float64
compression-ratio     float64
horsepower              int64
peak-rpm              float64
city-mpg                int64
highway-L/100km       float64
price                   int64
horsepower-binned      object
diesel                  int64
gas                     int64
```

jupyter Experiment_1 Last Checkpoint: 20 hours ago (autosaved)

File Edit View Insert Cell Kernel Widgets Help | Not Trusted | Python 3 (ipykernel)

| | | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 2 | 2 | 1 | 122 | alfa-romero | two | hatchback | rwd | front | 94.5 | 0.822681 | ... | 154 | 5000.0 | 19 | 9.038462 | 16500 | Medi |
| 3 | 3 | 2 | 164 | audi | four | sedan | fwd | front | 99.8 | 0.848630 | ... | 102 | 5500.0 | 24 | 7.833333 | 13950 | L |
| 4 | 4 | 2 | 164 | audi | four | sedan | 4wd | front | 99.4 | 0.848630 | ... | 115 | 5500.0 | 18 | 10.681818 | 17450 | L |

5 rows × 30 columns

```
In [8]: #Check the dimensions of the data frame
df.shape

Out[8]: (201, 30)

In [9]: #number of rows of a DataFrame
len(df)

Out[9]: 201

In [10]: #total number of elements in the DataFrame
df.size

Out[10]: 6030

In [ ]:
```

```
In [ ]: import numpy as np
        import matplotlib.pyplot as plt
        import pandas as pd
```

```
In [ ]: from google.colab import drive
        drive.mount('/content/drive/')
```

```
Mounted at /content/drive/
```

```
In [ ]: import pandas as pd

        data = pd.Series({'1st': 1, '2nd': 2, '3rd': 3, '4th': 4})
        print(data, '\n')
        print('Size = ', data.size)
```

```
In [ ]: import pandas as pd

        df = pd.DataFrame(
            {'1st': [1, 2], '2nd': [3, 4], '3rd': [5, 6], '4th': [7, 8]})
        print(df, '\n')
        print('Size = ', df.size)
```

```
In [ ]: import pandas as pd
```

jupyter Assignment1_Part_2 (unsaved changes)    Logou

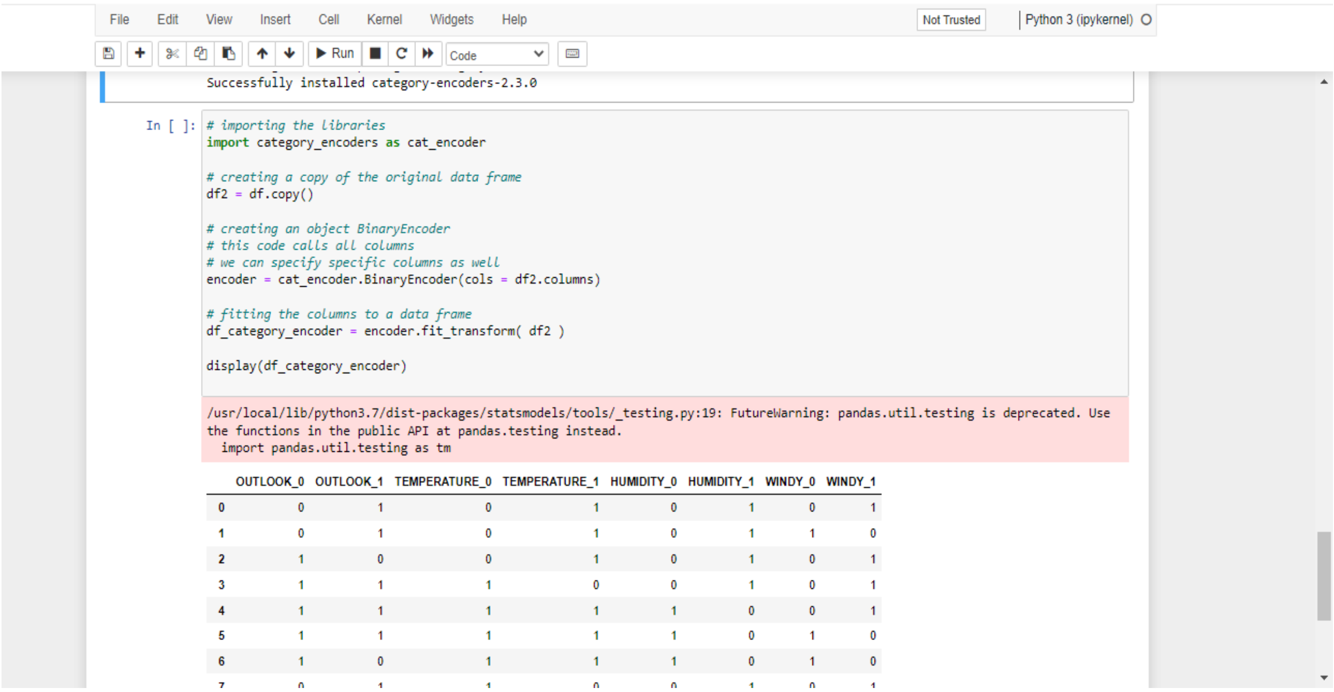
File Edit View Insert Cell Kernel Widgets Help    Not Trusted    Python 3 (ipykernel)

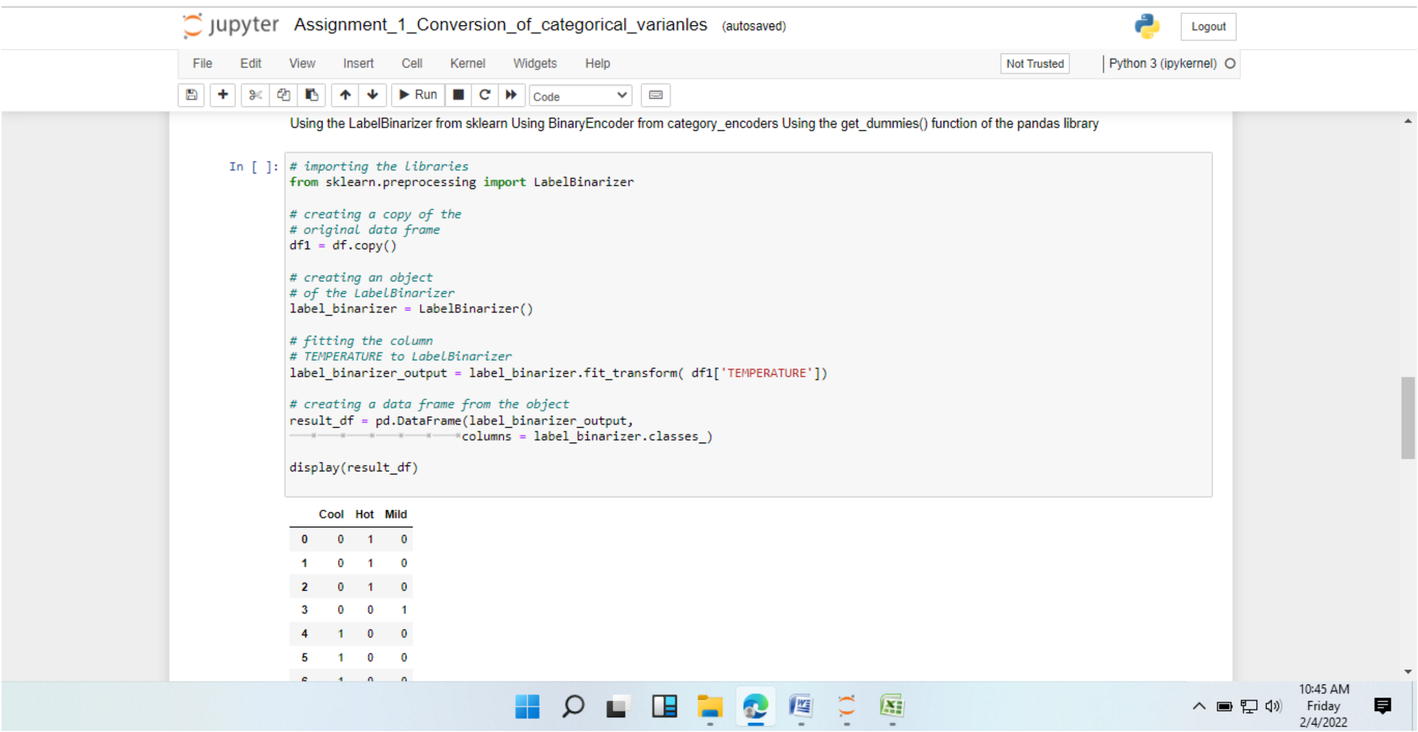
Run    Code

```
        dict = {'phone': ['Samsung S20', 'iPhone 11', 'Reliance Jio'],
                'price': [1000, 1100, 100]}

        dfA = pd.DataFrame(dict)
        print('The Datatype of DataFrame is: ')
        print(dfA.dtypes)
```

```
In [ ]: import pandas as pd
        import numpy as np

        dict = {'phone': ['Samsung S20', 'iPhone 11', 'Reliance Jio'],
                'price': [1000, 1100, 100],
                'discount': [np.nan, np.nan, np.nan]}

        dfA = pd.DataFrame(dict)
        print('The Datatype of DataFrame is: ')
        print(dfA.dtypes)
```

```
In [ ]: import pandas as pd
        import numpy as np

        dict = {'phone': ['Samsung S20', 'iPhone 11', 'Reliance Jio'],
                'price': [1000, 1100, 100],
                'discount': [np.nan, np.nan, np.nan],
                'arrivalDate': [pd.Timestamp('20180310'), pd.Timestamp('20190310'), pd.Timestamp('20140310')]}

        dfA = pd.DataFrame(dict)
```




## CONCLUSION:

They will understand how important data wrangling is for data and using different techniques optimized results can be obtained. Hence wrangle the data, before processing for analysis.

# Lab Assignment 2

# Title: Data Wrangling II

**PROBLEM STATEMENT:**

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.

**THEORY:**

**Working with Missing Data-**

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 a 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.

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

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

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

In order to check missing values in Pandas DataFrame, a function isnull() and notnull(). Both function help in checking whether a value is NaN or not. These function can also be used in Pandas Series in order to find null values in a series.

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

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

**Dataframe.isnull():-**

***Syntax: Pandas.isnull("DataFrame Name") or DataFrame.isnull()***
***Parameters: Object to check null values for***
***Return Type: Dataframe of Boolean values which are True for NaN values***

2. **Checking for missing values using notnull()**

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

**Dataframe.notnull():-**

***Syntax:*** *Pandas.notnull("DataFrame Name") or DataFrame.notnull()*
***Parameters:*** *Object to check null values for*
***Return Type:*** *Dataframe of Boolean values which are False for NaN values*

3. **Filling missing values using fillna(), replace() and interpolate()**

In order to fill null values in a datasets, we use fillna(), replace() and interpolate() function these function replace NaN values with some value of their own. All these function help in filling a null values in datasets of a DataFrame.

1. **fillna()** manages and let the user replace NaN values with some value of their own.

**Syntax:**

*DataFrame.fillna(value=None, method=None, axis=None, inplace=False, limit=None, downcast=None, **kwargs)*

**Parameters:**

***value :*** *Static, dictionary, array, series or dataframe to fill instead of NaN.*
***method :*** *Method is used if user doesn't pass any value. Pandas has different methods like bfill, backfill or ffill which fills the place with value in the Forward index or Previous/Back respectively.*
***axis:*** *axis takes int or string value for rows/columns. Input can be 0 or 1 for Integer and 'index' or 'columns' for String*
***inplace:*** *It is a boolean which makes the changes in data frame itself if True.*
***limit :*** *This is an integer value which specifies maximum number of consequetive forward/backward NaN value fills.*
***downcast :*** *It takes a dict which specifies what dtype to downcast to which one. Like Float64 to int64.*
*****kwargs :*** *Any other Keyword arguments*

2. **dataframe.replace()** function is used to replace a string, regex, list, dictionary, series, number etc. from a dataframe. This is a very rich function as it has many variations.The most powerful thing about this function is that it can work with Python regex (regular expressions).

***Syntax:*** *DataFrame.replace(to_replace=None, value=None, inplace=False, limit=None, regex=False, method='pad', axis=None)*

***Parameters:***
***to_replace :*** *[str, regex, list, dict, Series, numeric, or None] pattern that we are trying to replace in dataframe.*
***value :*** *Value to use to fill holes (e.g. 0), alternately a dict of values specifying which value to use for each column (columns not in the dict will not be filled). Regular expressions, strings and lists or dicts of such objects are also allowed.*
***inplace :*** *If True, in place. Note: this will modify any other views on this object (e.g. a column from a*

*DataFrame). Returns the caller if this is True.*
***limit :*** *Maximum size gap to forward or backward fill*
***regex :*** *Whether to interpret to_replace and/or value as regular expressions. If this is True then to_replace must be a string. Otherwise, to_replace must be None because this parameter will be interpreted as a regular expression or a list, dict, or array of regular expressions.*
***method :*** *Method to use when for replacement, when to_replace is a list.*

***Returns:*** *filled : NDFrame*

4. **Dropping missing values using dropna()**

Pandas *dropna()* method allows the user to analyze and drop Rows/Columns with Null values in different ways.

**Syntax:**

DataFrameName.dropna(axis=0, how='any', thresh=None, subset=None, inplace=False)

**Parameters:**

***axis:*** *axis takes int or string value for rows/columns. Input can be 0 or 1 for Integer and 'index' or 'columns' for String.*
***how:*** *how takes string value of two kinds only ('any' or 'all'). 'any' drops the row/column if ANY value is Null and 'all' drops only if ALL values are null.*
***thresh:*** *thresh takes integer value which tells minimum amount of na values to drop.*
***subset:*** *It's an array which limits the dropping process to passed rows/columns through list.*
***inplace:*** *It is a boolean which makes the changes in data frame itself if True*

**Detect and Remove the Outliers**

An **Outlier** is a data-item/object that deviates significantly from the rest of the (so-called normal)objects. They can be caused by measurement or execution errors.

**Detecting the outliers**

Outliers can be detected using visualization, implementing mathematical formulas on the dataset, or using the statistical approach.

**1. Visualization**

**Using Box Plot**

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

**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.

**2. Z-score** Z- Score is also called a standard score. This value/score helps to understand that 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*

### 3. IQR (Inter Quartile Range)

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*

### What is Interquartile Range IQR?

IQR is used to **measure variability** by dividing a data set into quartiles. The data is sorted in ascending order and split into 4 equal parts. Q1, Q2, Q3 called first, second and third quartiles are the values which separate the 4 equal parts.

- Q1 represents the 25th percentile of the data.
- Q2 represents the 50th percentile of the data.
- Q3 represents the 75th percentile of the data.

If a dataset has *2n / 2n+1* data points, then
Q1 = median of the dataset.
Q2 = median of n smallest data points.
Q3 = median of n highest data points.

IQR is the range between the first and the third quartiles namely Q1 and Q3: *IQR = Q3 – Q1*. The data points which fall below *Q1 – 1.5 IQR* or above *Q3 + 1.5 IQR* are outliers.

### Removing the 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.

How to delete exactly one row in python?

*dataframe.drop( row_index, inplace = True)*

### Data transformation:-

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. Data transformation predominantly deals with normalizing also known as scaling data , handling skewness and aggregation of attributes

### Min Max Scaler - normalization

*MinMaxScaler()* is applied **when the dataset is not distorted**. It normalizes the data into a range between 0 and 1 based on the formula:

$$z = \frac{x \ \text{-} \min(x)}{\max(x) \ \text{-} \min(x)}$$

### Standard Scaler - standardization

We use standardization when the dataset conforms to **normal distribution**. *StandardScaler()* converts the numbers into the standard form of **mean = 0 and variance = 1** based on z-score formula:

*x' = (x – mean) / standard deviation.*

**Robust Scaling-** *RobustScaler()* is more suitable for dataset with **skewed distributions and outliers** because it transforms the data based on median and quantile, specifically

*x' = (x – median) / inter-quartile range.*

**Z score normalization:**

Z score normalization is- In Z score normalization, we perform following mathematical transformation.

$$z = \frac{x - \mu}{\sigma}$$

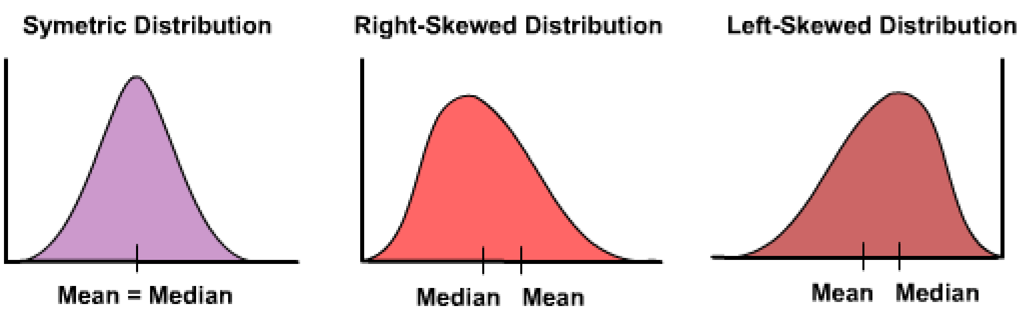
$$\mu = \text{Mean}$$
$$\sigma = \text{Standard Deviation}$$

## Skewness of data:

**skewness() :**

Skewness basically gives the shape of normal distribution of values.

If skewness value lies above +1 or below -1, data is highly skewed. If it lies between +0.5 to -0.5, it is moderately skewed. If the value is 0, then the data is symmetric




the skewness level, we should know whether it is positively skewed or negatively skewed.

***Positively skewed data:***

If tail is on the right as that of the second image in the figure, it is right skewed data. It is also called **positive skewed data.**Common transformations of this data include **square root, cube root, and log.**

***a.Cube root transformation:***

The **cube root transformation involves converting x** to x^(1/3). This is a fairly strong transformation with a substantial effect on distribution shape: but is weaker than the logarithm. It can be applied to negative and zero values too. Negatively skewed data.

***b.Square root transformation:***

Applied to positive values only. Hence, observe the values of column before applying.

***c.Logarithm transformation:***

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 and can be used to reduce right skewness.

**Negatively skewed data:**

If the tail is to the left of data, then it is called left skewed data. It is also called **negatively skewed data.** Common transformations include **square , cube root and logarithmic.**

***a.Square transformation:***

The **square**, x to $x^2$, has a moderate effect on distribution shape and it could be used to reduce left skewness. Another method of handling skewness is finding outliers and possibly removing them.

**How to transform features into Normal/Gaussian Distribution:-**

**How to check if a variable is following Normal Distribution**

There are various ways in which we can check the distribution of the variables. Some of them are:

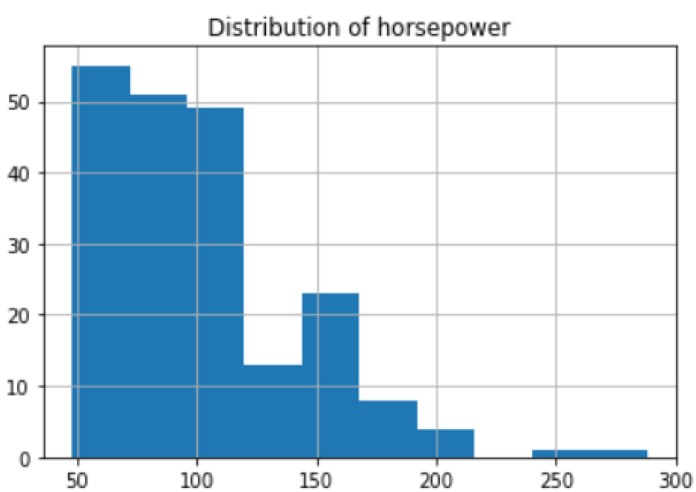
- Histogram
- Q-Q plot
- KDE plot
- Skewness

**Checking the distribution with Skewness**

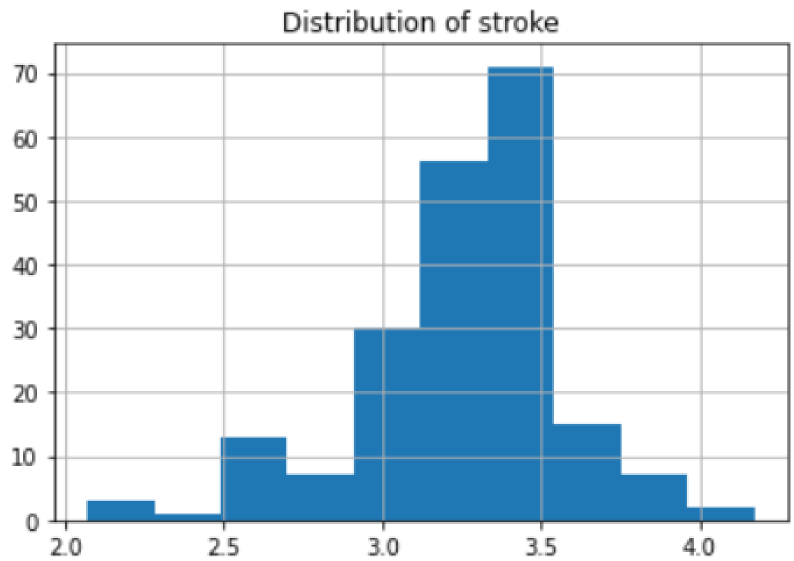
**dataframe.skew()**

- The variables with skewness > 1 are **highly positively skewed**.
- The variables with skewness < -1 are **highly negatively skewed**.
- The variables with 0.5 < skewness < 1 are **moderately positively skewed.**
- The variables with -0.5 < skewness < -1 are **moderately negatively skewed**.
- And, the variables with -0.5 < skewness < 0.5 are symmetric i.e **normally distributed**

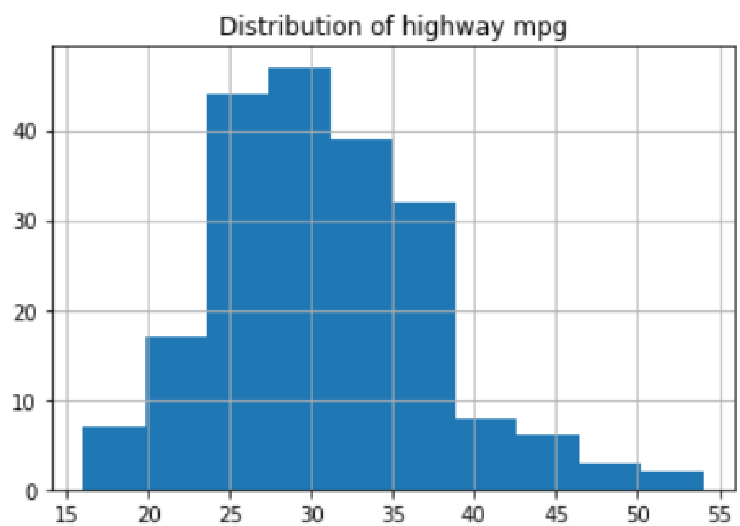
**Checking the distribution of some variables using Histogram**




**Highly positive Skewed i.e does not follow a normal distribution**



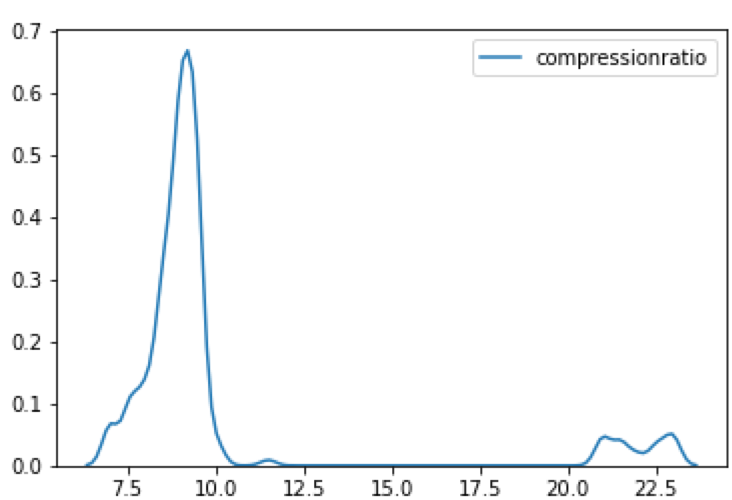

**Moderately negatively Skewed i.e does not follow a normal distribution**




**Symmetric i.e does follow a normal distribution:-**

**Checking the distribution of variables using KDE plot**

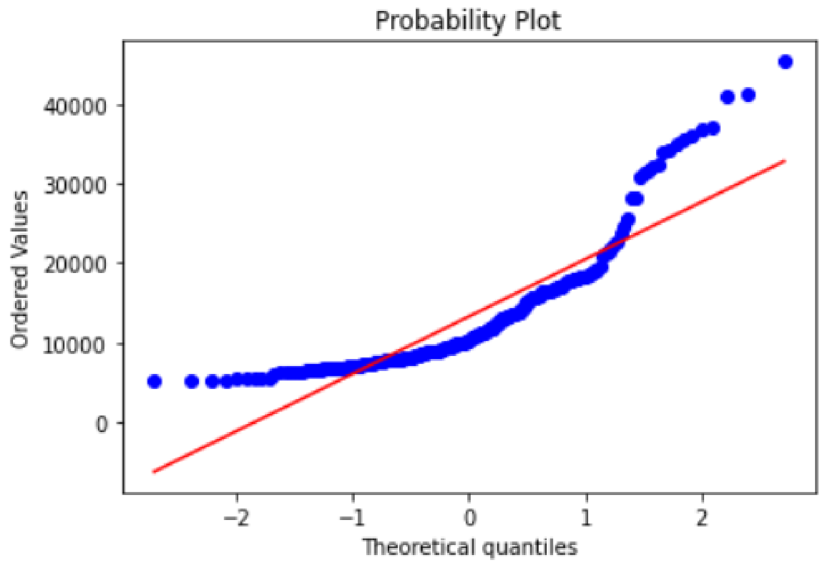
sns.kdeplot(dataframe.column_name);



**Checking the distribution of variables using a Q-Q plot**

A **Q-Q plot** is a scatterplot created by **plotting** two sets of quantiles against one another. If both sets of quantiles came from the same distribution, we should see the points forming a roughly straight line.

That is, if the data falls in a straight line then the variable follows normal distribution otherwise not.



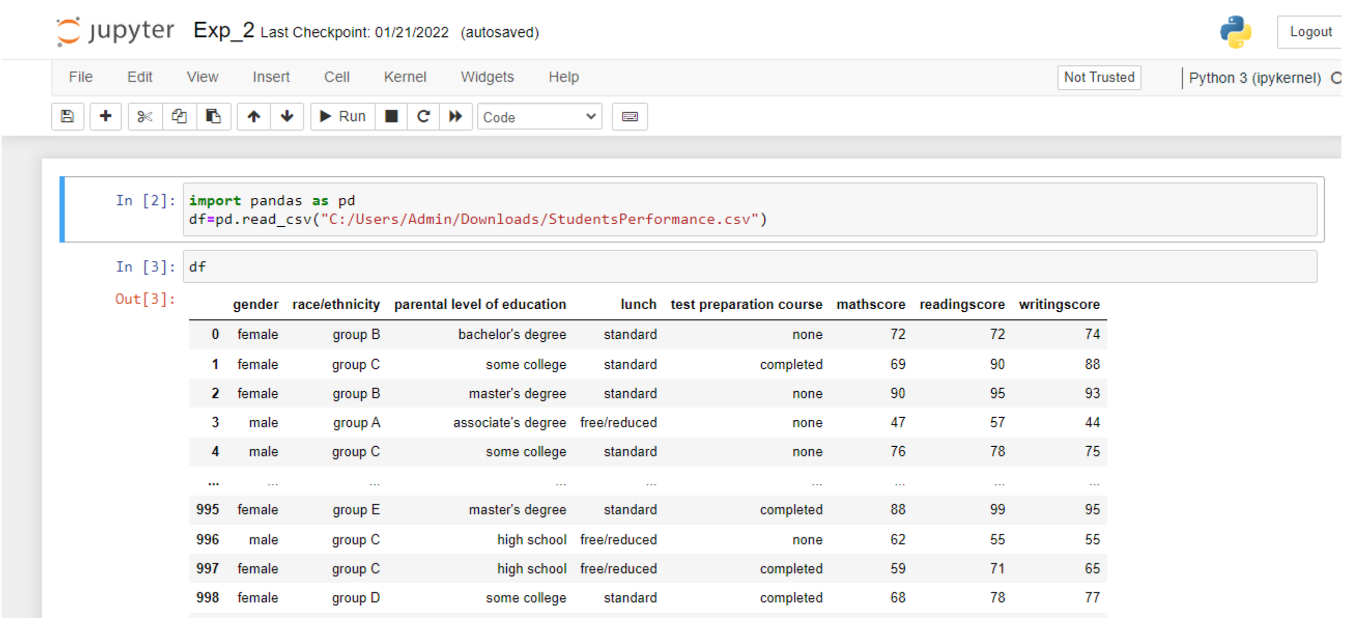

**Transformations to change the distribution of features:-**

**<u>Logarithmic Transformation</u> – This will convert the Price value to its log value**

**<u>Reciprocal Transformation</u> – This will inverse value 1/variable_name**

**<u>Square Root Transformation</u> – This transformation will take the square root**

**<u>Exponential Transformation</u>: The exponential value of the variable**




In [2]:
```
import pandas as pd
df=pd.read_csv("C:/Users/Admin/Downloads/StudentsPerformance.csv")
```

In [3]:
```
df
```

Out[3]:

| | gender | race/ethnicity | parental level of education | lunch | test preparation course | mathscore | readingscore | writingscore |
|---|---|---|---|---|---|---|---|---|
| 0 | female | group B | bachelor's degree | standard | none | 72 | 72 | 74 |
| 1 | female | group C | some college | standard | completed | 69 | 90 | 88 |
| 2 | female | group B | master's degree | standard | none | 90 | 95 | 93 |
| 3 | male | group A | associate's degree | free/reduced | none | 47 | 57 | 44 |
| 4 | male | group C | some college | standard | none | 76 | 78 | 75 |
| ... | ... | ... | ... | ... | ... | ... | ... | ... |
| 995 | female | group E | master's degree | standard | completed | 88 | 99 | 95 |
| 996 | male | group C | high school | free/reduced | none | 62 | 55 | 55 |
| 997 | female | group C | high school | free/reduced | completed | 59 | 71 | 65 |
| 998 | female | group D | some college | standard | completed | 68 | 78 | 77 |
| 999 | female | group D | some college | free/reduced | none | 77 | 86 | 86 |

jupyter Exp_2 Last Checkpoint: 01/21/2022 (autosaved)

File Edit View Insert Cell Kernel Widgets Help Not Trusted Python 3 (ipykernel)

1000 rows × 8 columns

In [4]:
```
q1=df.mathscore.quantile(0.25)
q3=df.mathscore.quantile(0.75)
q1,q3
```

Out[4]: (57.0, 77.0)

In [5]:
```
IQR=q3-q1
IQR
```

Out[5]: 20.0

In [6]:
```
lower_limit=q1-1.5*IQR
upper_limit=q3+1.5*IQR
lower_limit,upper_limit
```

Out[6]: (27.0, 107.0)

In [7]:
```
df[(df.mathscore<lower_limit)|(df.mathscore>upper_limit)]
```

Out[7]:

| | gender | race/ethnicity | parental level of education | lunch | test preparation course | mathscore | readingscore | writingscore |
|---|---|---|---|---|---|---|---|---|
| 17 | female | group B | some high school | free/reduced | none | 18 | 32 | 28 |
| 59 | female | group C | some high school | free/reduced | none | 0 | 17 | 10 |

jupyter Exp_2 Last Checkpoint: 01/21/2022 (autosaved)

File Edit View Insert Cell Kernel Widgets Help Not Trusted Python 3 (ipykernel)

Out[6]: (27.0, 107.0)

In [7]:
```
df[(df.mathscore<lower_limit)|(df.mathscore>upper_limit)]
```


Out[7]:

| | gender | race/ethnicity | parental level of education | lunch | test preparation course | mathscore | readingscore | writingscore |
|---|---|---|---|---|---|---|---|---|
| 17 | female | group B | some high school | free/reduced | none | 18 | 32 | 28 |
| 59 | female | group C | some high school | free/reduced | none | 0 | 17 | 10 |
| 145 | female | group C | some college | free/reduced | none | 22 | 39 | 33 |
| 338 | female | group B | some high school | free/reduced | none | 24 | 38 | 27 |
| 466 | female | group D | associate's degree | free/reduced | none | 26 | 31 | 38 |
| 787 | female | group B | some college | standard | none | 19 | 38 | 32 |
| 842 | female | group B | high school | free/reduced | completed | 23 | 44 | 36 |
| 980 | female | group B | high school | free/reduced | none | 8 | 24 | 23 |

In [8]:
```
df[(df.mathscore>lower_limit)&(df.mathscore<upper_limit)]
```

Out[8]:

| | gender | race/ethnicity | parental level of education | lunch | test preparation course | mathscore | readingscore | writingscore |
|---|---|---|---|---|---|---|---|---|
| 0 | female | group B | bachelor's degree | standard | none | 72 | 72 | 74 |
| 1 | female | group C | some college | standard | completed | 69 | 90 | 88 |

jupyter Exp_2 Last Checkpoint: 01/21/2022 (autosaved)

File Edit View Insert Cell Kernel Widgets Help | Not Trusted | Python 3 (ipykernel)

```
In [9]: df.describe()
```

Out[9]:

| | mathscore | readingscore | writingscore |
|---|---|---|---|
| count | 1000.00000 | 1000.000000 | 1000.000000 |
| mean | 66.08900 | 69.169000 | 68.054000 |
| std | 15.16308 | 14.600192 | 15.195657 |
| min | 0.00000 | 17.000000 | 10.000000 |
| 25% | 57.00000 | 59.000000 | 57.750000 |
| 50% | 66.00000 | 70.000000 | 69.000000 |
| 75% | 77.00000 | 79.000000 | 79.000000 |
| max | 100.00000 | 100.000000 | 100.000000 |

```
In [10]: df['zscore']=(df.mathscore - df.mathscore.mean())/df.mathscore.std()
df
```

Out[10]:

| | gender | race/ethnicity | parental level of education | lunch | test preparation course | mathscore | readingscore | writingscore | zscore |
|---|---|---|---|---|---|---|---|---|---|
| 0 | female | group B | bachelor's degree | standard | none | 72 | 72 | 74 | 0.389828 |
| 1 | female | group C | some college | standard | completed | 69 | 90 | 88 | 0.191979 |
| 2 | female | group B | master's degree | standard | none | 90 | 95 | 93 | 1.576922 |
| 3 | male | group A | associate's degree | free/reduced | none | 47 | 57 | 44 | -1.258913 |
| 4 | male | group C | some college | standard | none | 76 | 78 | 75 | 0.653627 |

jupyter Exp_2 Last Checkpoint: 01/21/2022 (autosaved)

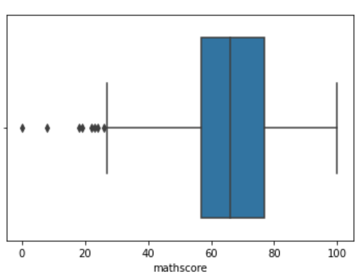
File Edit View Insert Cell Kernel Widgets Help | Not Trusted | Python 3 (ipykernel)

1000 rows × 9 columns

```
In [11]: import seaborn as sns
sns.boxplot(df['mathscore'])
```

```
C:\ProgramData\Anaconda3\lib\site-packages\seaborn\_decorators.py:36: FutureWarning: Pass the following variable as a keyword a
rg: x. From version 0.12, the only valid positional argument will be `data`, and passing other arguments without an explicit ke
yword will result in an error or misinterpretation.
  warnings.warn(
```

Out[11]: <AxesSubplot:xlabel='mathscore'>




## Conclusion:

Students will learn about data transformation techniques and outliers. Techniques to detect & remove outliers. Normal Distribution, Scaling and techniques to transform data

# Lab Assignment 3

# Title: Data Wrangling II

**PROBLEM STATEMENT:**

Descriptive Statistics - Measures of Central Tendency and variability perform the following operations on any open source dataset (e.g., data.csv)

1. Provide summary statistics (mean, median, minimum, maximum, standard deviation) for a dataset (age, income etc.) with numeric variables grouped by one of the qualitative (categorical) variables. For example, if your categorical variable is age groups and quantitative variable is income, then provide summary statistics of income grouped by the age groups. Create a list that contains a numeric value for each response to the categorical variable.

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

## THEORY:

**What is Statistics?**

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

**There are two branches of Statistics.**

- DESCRIPTIVE STATISTICS: Descriptive Statistics is a statistics or a measure that describes the data.
- INFERENTIAL STATISTICS: Using a random sample of data taken from a population to describe and make inferences about the population is called Inferential Statistics.

**Descriptive Statistics**

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

**Commonly Used Measures**

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

**Measures of Central Tendency**

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

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

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

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

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

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

## How to calculate summary statistics?

A large number of methods collectively compute descriptive statistics and other related operations on Data Frame. Most of these are aggregations like sum(), mean() etc.

**Functions & Description: To calculate Mean, Standard Deviation, Median, Max, and Min we can apply these functions.**

| Sr.No. | Function | Description |
|---|---|---|
| 1 | count() | Number of non-null observations |
| 2 | sum() | Sum of values |
| 3 | mean() | Mean of Values |
| 4 | median() | Median of Values |
| 5 | mode() | Mode of values |

| | | |
|---|---|---|
| 6 | std() | Standard Deviation of the Values |
| 7 | min() | Minimum Value |
| 8 | max() | Maximum Value |
| 9 | abs() | Absolute Value |
| 10 | prod() | Product of Values |
| 11 | cumsum() | Cumulative Sum |
| 12 | cumprod() | Cumulative Product |

**Using the 'describe()' Method:-**

We can use the describe function to generate the statistics above and apply it to multiple columns simultaneously. It also provides the lower, median and upper percentiles.

## Aggregating statistics grouped by category

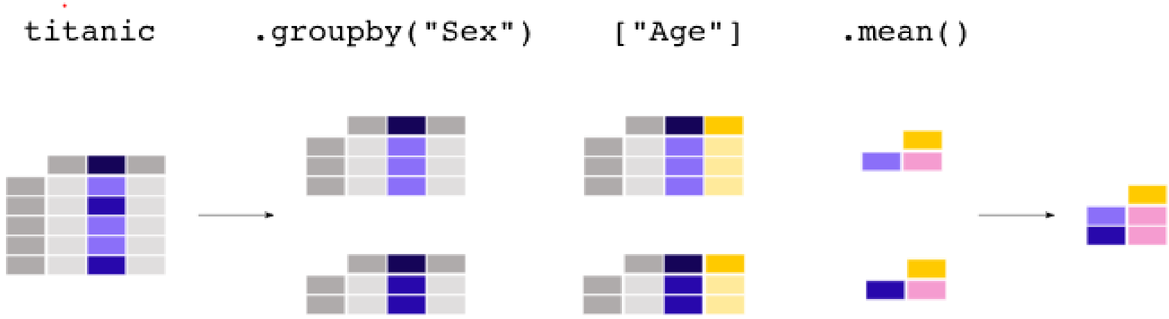
**Using 'groupby()' to Aggregate**

Suppose we wanted to know the average runtime for each genre. We can use the 'groupby()' method to calculate these statistics:

The group by method is used to support this type of operations. More general, this fits in the more general split-apply-combine pattern:

- Split the data into groups
- Apply a function to each group independently
- Combine the results into a data structure

The apply and combine steps are typically done together in pandas.




Example:-

titanic.groupby(["Sex", "Pclass"])["Fare"].mean()

**Output:**

Sex Pclass

female 1 106.125798

2 21.970121

3 16.118810

male 1 67.226127

2 19.741782

3 12.661633

Name: Fare, dtype: float64

Grouping can be done by multiple columns at the same time. Provide the column names as a list to the **groupby()** method.

### Count number of records by category-

- The value_counts() method counts the number of records for each category in a column.
- value_counts is a convenient shortcut to count the number of entries in each category of a variable

**Procedure-**

**STEPS:**

**1. IMPORT REQUIRED LIBRARIES**

**2. READ CSV FILE (ADULT.CSV) AND (IRIS.CSV)**

**3. PROVIDE SUMMARY STATISTICS USING PREDEFINED FUNCTION LIKE MEAN(),MEDIAN(),MODE(), DESCRIBE() ETC.**

**4. CATEGORIZE DATA USING GROUPBY() METHOD AND PROVIDE STATISTICS.**

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

```
import pandas as pd

df = pd.read_csv("C:/Users/Admin/Downloads/adult.csv")

print(df)

#summary statistics of age grouped by gender

df.groupby("gender")["age"].describe()

df.groupby("marital-status")["age"].mean()

df.groupby("marital-status")["age"].median()

#grouping can be done on multiple columns

# summary statistics of age grouped by gender & marital-status

df.groupby(["gender","marital-status"])["age"].std()

#summary statistics of age grouped by income

df.groupby("income")["age"].mean()

df.groupby(["income","gender"])["age"].mean()

df.groupby("marital-status")["marital-status"].count()

#Count number of records by category

#The value_counts() method counts the number of records for each category in a column.

df["marital-status"].value_counts()
```

**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.**

**Program: (Code without Group by function)**

```
import pandas as pd

d = pd.read_csv("C:/Users/Admin/Downloads/Iris.csv")

print('Iris-setosa')

setosa = d['Species'] == 'Iris-setosa'

print(d[setosa].describe())

print('\nIris-versicolor')

setosa = d['Species'] == 'Iris-versicolor'

print(d[setosa].describe())

print('\nIris-virginica')

setosa = d['Species'] == 'Iris-virginica'

print(d[setosa])

print(d[setosa].describe())
```

**Program using Group By function:-**

```
import pandas as pd

d = pd.read_csv("C:/Users/Admin/Downloads/Iris.csv")

#Species

d.groupby(["Species"])["SepalLengthCm"].mean()

d.groupby(["Species"])["SepalLengthCm"].std()

d.groupby(["Species"])["SepalLengthCm"].describe()

d.groupby(["Species"])["SepalLengthCm"].quantile(q=0.75)

d.groupby(["Species"])["SepalLengthCm"].quantile(q=0.25)

a=d.groupby(["Species"])["SepalLengthCm"].mean()

print(a)

b=d.groupby(["Species"])["SepalLengthCm"].median()

print(b)
```

list=[a,b]

print(list)

Code with Output:-

jupyter 3)_1_Part_Descriptive Statistics Last Checkpoint: 02/28/2022 (autosaved)

Logout

File Edit View Insert Cell Kernel Widgets Help

Not Trusted | Python 3 (ipykernel)

Run | Code

In [18]:
```
import pandas as pd
df = pd.read_csv("C:/Users/Admin/Downloads/adult.csv")
```

In [2]:
```
print(df)
```

```
       age     workclass  fnlwgt     education  educational-num  \
0       25       Private  226802          11th                7
1       38       Private   89814       HS-grad                9
2       28     Local-gov  336951    Assoc-acdm               12
3       44       Private  160323  Some-college               10
4       18             ?  103497  Some-college               10
...    ...           ...     ...           ...              ...
48837   27       Private  257302    Assoc-acdm               12
48838   40       Private  154374       HS-grad                9
48839   58       Private  151910       HS-grad                9
48840   22       Private  201490       HS-grad                9
48841   52  Self-emp-inc  287927       HS-grad                9

           marital-status         occupation relationship   race  gender  \
0           Never-married  Machine-op-inspct    Own-child  Black    Male
1      Married-civ-spouse    Farming-fishing      Husband  White    Male
2      Married-civ-spouse    Protective-serv      Husband  White    Male
3      Married-civ-spouse  Machine-op-inspct      Husband  Black    Male
4           Never-married                  ?    Own-child  White  Female
...                   ...                ...          ...    ...     ...
48837  Married-civ-spouse       Tech-support         Wife  White  Female
```

jupyter 3)_1_Part_Descriptive Statistics Last Checkpoint: 02/28/2022 (autosaved)

Logout

File Edit View Insert Cell Kernel Widgets Help

Not Trusted | Python 3 (ipykernel)

Run | Code

In [19]:
```
df.groupby("gender")["age"].describe()
```

Out[19]:

| gender | count | mean | std | min | 25% | 50% | 75% | max |
|---|---|---|---|---|---|---|---|---|
| Female | 16192.0 | 36.927989 | 14.137423 | 17.0 | 25.0 | 35.0 | 46.0 | 90.0 |
| Male | 32650.0 | 39.494395 | 13.412850 | 17.0 | 29.0 | 38.0 | 48.0 | 90.0 |

In [20]:
```
df.groupby("marital-status")["age"].mean()
```

Out[20]:
```
marital-status
Divorced                  6633
Married-AF-spouse           37
Married-civ-spouse       22379
Married-spouse-absent      628
Never-married            16117
Separated                 1530
Widowed                   1518
Name: age, dtype: int64
```

In [11]:
```
df.groupby("marital-status")["age"].median()
```

Out[11]:
```
marital-status
Divorced                 42.0
Married-AF-spouse        30.0
Married-civ-spouse       42.0
Married-spouse-absent    40.0
Never-married            25.0
```

jupyter 3)_1_Part_Descriptive Statistics Last Checkpoint: 02/28/2022 (autosaved)

Logout

File Edit View Insert Cell Kernel Widgets Help | Not Trusted | Python 3 (ipykernel)

```
In [21]: df.groupby(["gender","marital-status"])["age"].std()

Out[21]: gender  marital-status
         Female  Divorced                 10.794868
                 Married-AF-spouse        12.342744
                 Married-civ-spouse       11.402805
                 Married-spouse-absent    13.019854
                 Never-married            10.231671
                 Separated                10.757639
                 Widowed                  11.657268
         Male    Divorced                 10.161659
                 Married-AF-spouse         6.336522
                 Married-civ-spouse       12.080786
                 Married-spouse-absent    12.631023
                 Never-married             9.717602
                 Separated                10.811704
                 Widowed                  14.216489
         Name: age, dtype: float64
```

```
In [13]: df.groupby("income")["age"].mean()

Out[13]: income
         <=50K    36.872184
         >50K     44.275178
         Name: age, dtype: float64
```

```
In [14]: df.groupby(["income","gender"])["age"].mean()
```

jupyter 3)_2_part_Iris_data Last Checkpoint: 02/28/2022 (autosaved)

Logout

File Edit View Insert Cell Kernel Widgets Help | Not Trusted | Python 3 (ipykernel)

```
In [23]: import pandas as pd
         d = pd.read_csv("C:/Users/Admin/Downloads/Iris.csv")
         print('Iris-setosa')
         setosa = d['Species'] == 'Iris-setosa'
         print(d[setosa].describe())
         print('\nIris-versicolor')
         setosa = d['Species'] == 'Iris-versicolor'
         print(d[setosa].describe())
         print('\nIris-virginica')
         setosa = d['Species'] == 'Iris-virginica'
         print(d[setosa])
         print(d[setosa].describe())
```

```
Iris-setosa
             Id  SepalLengthCm  SepalWidthCm  PetalLengthCm  PetalWidthCm
count  50.00000       50.00000     50.000000      50.000000      50.00000
mean   25.50000        5.00600      3.418000       1.464000       0.24400
std    14.57738        0.35249      0.381024       0.173511       0.10721
min     1.00000        4.30000      2.300000       1.000000       0.10000
25%    13.25000        4.80000      3.125000       1.400000       0.20000
50%    25.50000        5.00000      3.400000       1.500000       0.20000
75%    37.75000        5.20000      3.675000       1.575000       0.30000
max    50.00000        5.80000      4.400000       1.900000       0.60000

Iris-versicolor
             Id  SepalLengthCm  SepalWidthCm  PetalLengthCm  PetalWidthCm
count  50.00000      50.000000     50.000000      50.000000     50.000000
mean   75.50000       5.936000      2.770000       4.260000      1.326000
```

jupyter 3)_2_part_Iris_data Last Checkpoint: 02/28/2022 (autosaved)

File Edit View Insert Cell Kernel Widgets Help | Not Trusted | Python 3 (ipykernel)

```
In [3]: #Species
        d.groupby(["Species"])["SepalLengthCm"].mean()

Out[3]: Species
        Iris-setosa        5.006
        Iris-versicolor    5.936
        Iris-virginica     6.588
        Name: SepalLengthCm, dtype: float64

In [4]: d.groupby(["Species"])["SepalLengthCm"].std()

Out[4]: Species
        Iris-setosa        0.352490
        Iris-versicolor    0.516171
        Iris-virginica     0.635880
        Name: SepalLengthCm, dtype: float64

In [22]: d.groupby(["Species"])["SepalLengthCm"].describe()

Out[22]:
```

| Species | count | mean | std | min | 25% | 50% | 75% | max |
|---|---|---|---|---|---|---|---|---|
| Iris-setosa | 50.0 | 5.006 | 0.352490 | 4.3 | 4.800 | 5.0 | 5.2 | 5.8 |
| Iris-versicolor | 50.0 | 5.936 | 0.516171 | 4.9 | 5.600 | 5.9 | 6.3 | 7.0 |
| Iris-virginica | 50.0 | 6.588 | 0.635880 | 4.9 | 6.225 | 6.5 | 6.9 | 7.9 |

```
In [30]: d.groupby(["Species"])["SepalLengthCm"].quantile(q=0.75)
```

jupyter 3)_2_part_Iris_data Last Checkpoint: 02/28/2022 (autosaved)

File Edit View Insert Cell Kernel Widgets Help | Not Trusted | Python 3 (ipykernel)

```
In [7]: d.groupby(["Species"])["SepalLengthCm"].quantile(q=0.50)

Out[7]: Species
        Iris-setosa        5.0
        Iris-versicolor    5.9
        Iris-virginica     6.5
        Name: SepalLengthCm, dtype: float64

In [31]: a=d.groupby(["Species"])["SepalLengthCm"].mean()
         print(a)

         Species
         Iris-setosa        5.006
         Iris-versicolor    5.936
         Iris-virginica     6.588
         Name: SepalLengthCm, dtype: float64

In [33]: b=d.groupby(["Species"])["SepalLengthCm"].median()
         print(b)

         Species
         Iris-setosa        5.0
         Iris-versicolor    5.9
         Iris-virginica     6.5
         Name: SepalLengthCm, dtype: float64

In [34]: list=[a,b]
```

## CONCLUSION:

To summarize, here we discussed how to generate summary statistics using the Pandas library. Here, we discussed how to use pandas methods to generate mean, median, max, min and standard deviation. We also saw describe () method which allows us to generate percentiles, in addition to the mean, median, max, min and standard deviation, for any numerical column. Finally, we showed how to generate aggregate statistics for categorical columns.

# Lab Assignment 4

# Title: Data Analytics I

**PROBLEM STATEMENT:**

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.

## THEORY:

Machine Learning is a part of Artificial Intelligence (AI), where the model will learn from the data and can predict the outcome. Machine Learning is a study of statistical computer algorithm that improves automatically from the data. Unlike computer algorithms, rely on human beings.

**Types of Machine Learning Algorithms**

- Supervised Machine Learning

In Supervised Learning, we will have both the independent variable (predictors) and the dependent variable (response). Our model will be trained using both independent and dependent variables. So we can predict the outcome when the test data is given to the model. Here, using the output our model can measure its accuracy and can learn over time. In supervised learning, we will solve **both Regression and Classification** problems.

- Unsupervised Machine Learning

In Unsupervised Learning, our model will won't be provided an output variable to train. So we can't use the model to predict the outcome like Supervised Learning. These algorithms will be used to analyze the data and find the hidden pattern in it. **Clustering and Association Algorithms** are part of unsupervised learning.

- Reinforcement Learning

Reinforcement learning is the training of machine learning models which make a decision sequentially. In simple words, the output of the model will depend on the present input, and the next input will depend on the previous output of the model.

**What is Regression?**

Regression analysis is a statistical method that helps us to understand the relationship between dependent and one or more independent variables**,**

- **Dependent Variable**

This is the Main Factor that we are trying to predict.

- **Independent Variable**

These are the variables that have a relationship with the dependent variable.

**What is Linear Regression?**

In Machine Learning lingo, Linear Regression (LR) means simply finding the best fitting line that explains the variability between the dependent and independent features very well or we can say it describes the linear relationship between independent and dependent features, and in linear regression, the algorithm predicts the continuous features (e.g. Salary, Price), rather than deal with the categorical features (e.g. cat, dog).

**Simple Linear Regression**

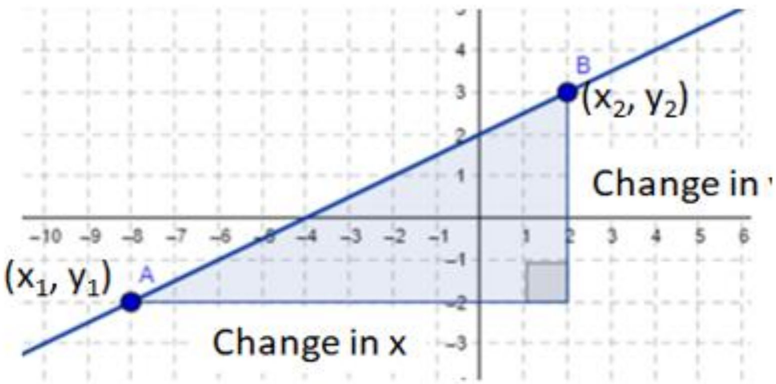
Simple Linear Regression uses the slope-intercept (weight-bias) form, where our model needs to find the optimal value for both slope and intercept. So with the optimal values, the model can find the variability between the independent and dependent features and produce accurate results. In simple linear regression, the model takes a single independent and dependent variable.

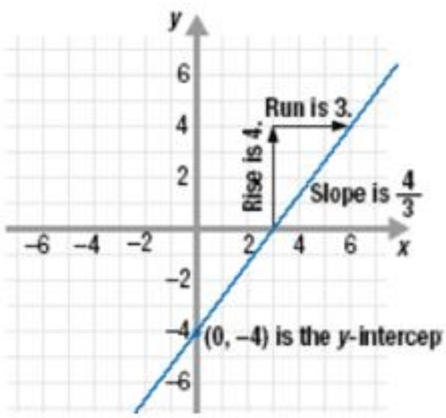
There are many equations to represent a straight line, we will stick with the common equation,

$$y = b_o + b_1 x$$

Here, y and x are the dependent variables, and independent variables respectively. b1(m) and b0(c) are slope and y-intercept respectively.

$$\text{slope} = \frac{y_2 - y_1}{x_2 - x_1} = \frac{\text{Change in y}}{\text{Change in x}}$$



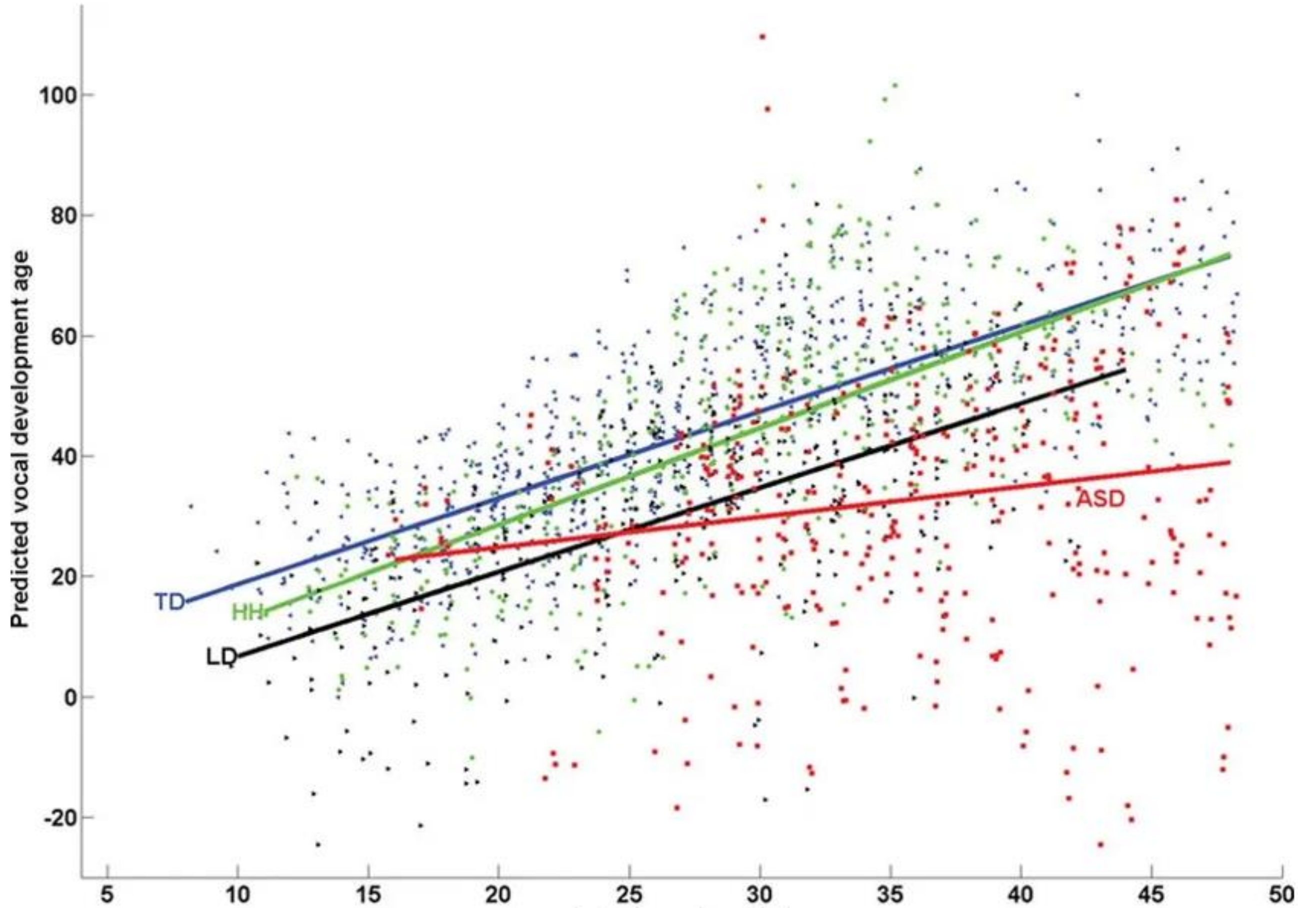




Slope(m) tells, for one unit of increase in *x*, How many units does it increase in y. When the line is steep, the slope will be higher, the slope will be lower for the less steep line.

Constant(c) means, What is the value of y when the x is zero.

How the Model will Select the Best Fit Line?

First, our model will try a bunch of different straight lines from that it finds the optimal line that predicts our data points well.




For finding the best fit line our model uses the cost function. In machine learning, every algorithm has a cost function, and in simple linear regression, the goal of our algorithm is to find a minimal value for the cost function.And in linear regression (LR), we have many cost functions, but mostly used cost function is MSE(Mean Squared Error). It is also known as a Least Squared Method.

$$\mathrm{MSE} = \frac{1}{n}\sum_{i=1}^{n}(Y_i - \hat{Y}_i)^2.$$

Yi – Actual value,

Y^i – Predicted value,

n – number of records.

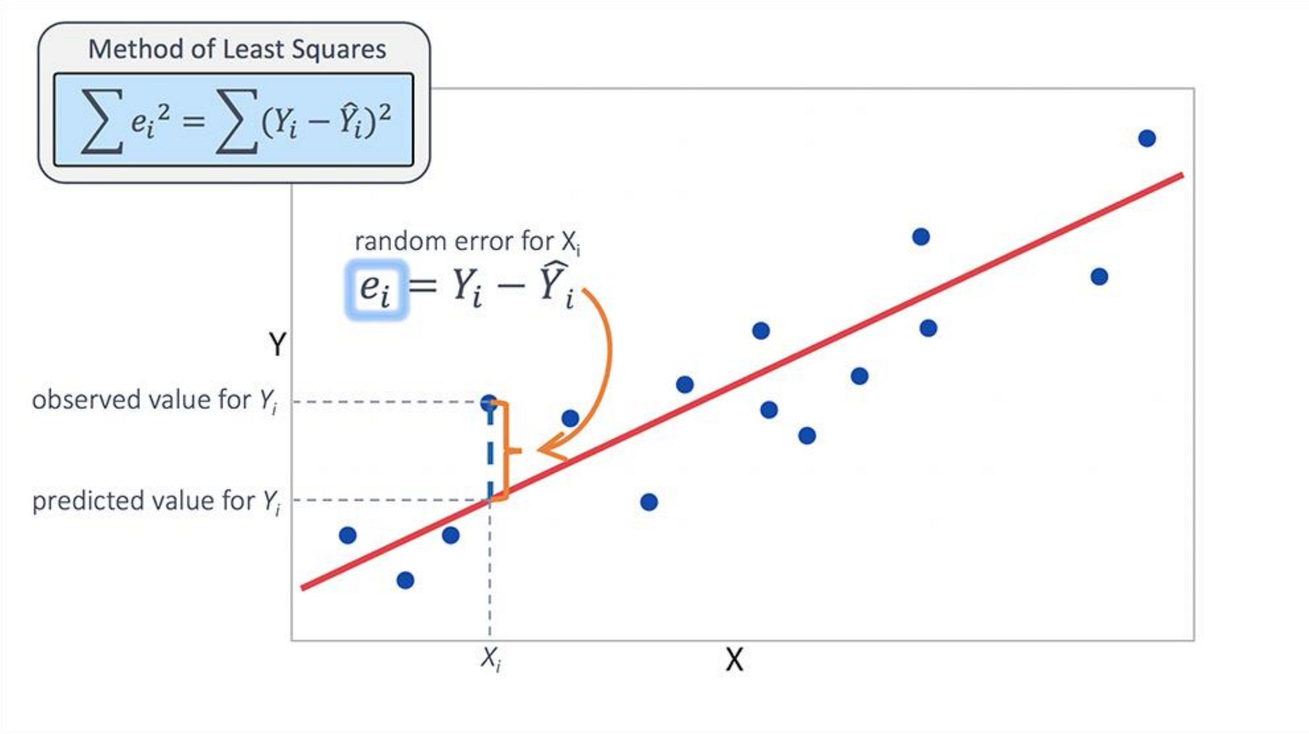
( yi – yi_hat ) is a Loss Function. And you can find in most times people will interchangeably use the word loss and cost function. But they are different, and we are squaring the terms to neglect the negative value.

Loss Function

It is a calculation of loss for single training data.

**Cost Function**

It is a calculation of average loss over the entire dataset.




From the above picture, blue data points are representing the actual values from training data, a red line(vector) is the predicted value for that actual blue data point. we can notice a random error, the actual value-predicted value, model is trying to minimize the error between the actual and predicted value. Because in the real world we need a model, which makes the prediction very well. So our model will find the loss between all the actual and predicted values respectively. And it selects the line which has an average error of all points lower.

**Steps**

1. Our model will fit all possible lines and find an overall average error between the actual and predicted values for each line respectively.
2. Selects the line which has the lowest overall error. And that will be the best fit line.

## Procedure & Code:

import numpy as np

import pandas as pd

from sklearn import datasets

from sklearn.linear_model import LinearRegression

from sklearn.model_selection import train_test_split

#loading the dataset direclty from sklearn

boston = datasets.load_boston()

print(boston)

bos.describe()

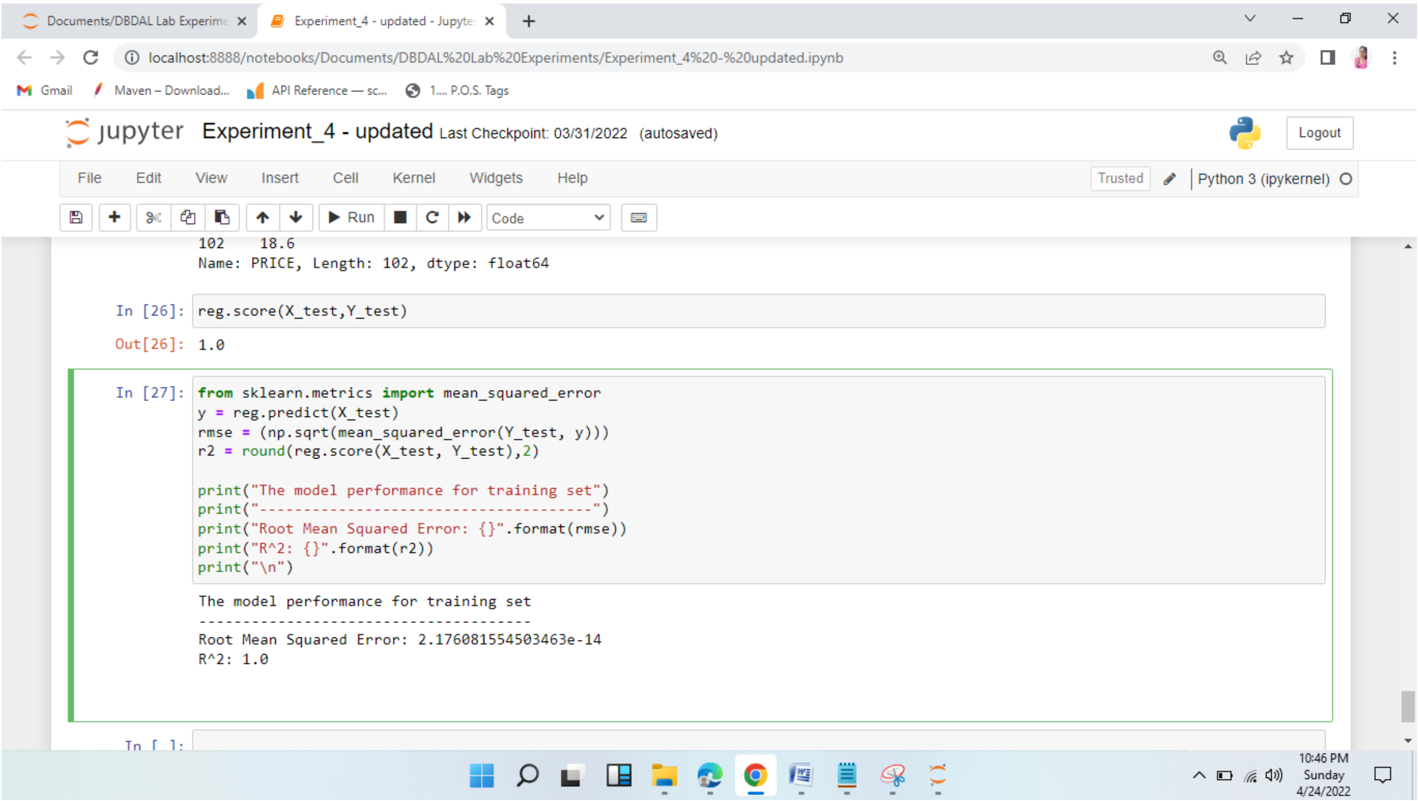
#sklearn returns Dictionary-like object, the interesting attributes are: ‘data’, the data to learn, ‘target’, the regression targets, ‘DESCR’, the full description of the dataset, and ‘filename’, the physical location of boston csv dataset.

print(type(boston))

```
print('\n')
print(boston.keys())
print('\n')
print(boston.data.shape)
print('\n')
print(boston.feature_names)
#The details about the features and more information about the dataset can be seen by using boston.DESCR
print(boston.DESCR)
#Before applying any model we have to convert this to a pandas dataframe,
#which we can do by calling the dataframe on boston.data. We also adds the target variable to the dataframe from boston.target
bos = pd.DataFrame(boston.data, columns = boston.feature_names)
bos['PRICE']=pd.DataFrame(boston.target)
print(bos.head())
#Get some statistics from dataset
print(bos.describe())
#initialize linear regression model
reg=LinearRegression()
#split into training-80% & testing data-20%
X_train, X_test, Y_train, Y_test = train_test_split(bos, bos['PRICE'], test_size = 0.20,random_state=10)
print(X_train.shape)
print(X_test.shape)
print(Y_train.shape)
print(Y_test.shape)
#train model with our training data
reg.fit(X_train,Y_train)
#print predictions on our test data
y=reg.predict(X_test)
print(y)
#actucal values
print(Y_test)
reg.score(X_test,Y_test)
from sklearn.metrics import mean_squared_error
y = reg.predict(X_test)
rmse = (np.sqrt(mean_squared_error(Y_test, y)))
r2 = round(reg.score(X_test, Y_test),2)

print("The model performance for training set")
```

```
print("--------------------------------------")
print("Root Mean Squared Error: {}".format(rmse))
print("R^2: {}".format(r2))
print("\n")
```

Output:-




## CONCLUSION:

We studied & applied the concepts of linear regression on the Boston housing dataset. Also we calculated the accuracy of the model.

# Lab Assignment 5

## Title: Data Analytics II

**PROBLEM STATEMENT:**

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.

## THEORY:

**What is Logistic Regression?**

- Logistic Regression: Classification techniques are an essential part of machine learning and data mining applications. Approximately 70% of problems in Data Science are classification problems. There are lots of classification problems that are available, but logistic regression is common and is a useful regression method for solving the binary classification problem.
- Another category of classification is Multinomial classification, which handles the issues where multiple classes are present in the target variable. For example, the IRIS dataset is a very famous example of multi-class classification. Other examples are classifying article/blog/document categories.
- Logistic Regression can be used for various classification problems such as spam detection. Diabetes prediction, if a given customer will purchase a particular product or will they churn another competitor, whether the user will click on a given advertisement link or not, and many more examples are in the bucket.
- Logistic Regression is one of the most simple and commonly used Machine Learning algorithms for two-class classification. It is easy to implement and can be used as the baseline for any binary classification problem. Its basic fundamental concepts are also constructive in deep learning.
- Logistic regression describes and estimates the relationship between one dependent binary variable and independent variables. Logistic regression is a statistical method for predicting binary classes. The outcome or target variable is dichotomous in nature. Dichotomous means there are only two possible classes. For example, it can be used for cancer detection problems. It computes the probability of an event occurring.
- It is a special case of linear regression where the target variable is categorical in nature. It uses a log of odds as the dependent variable. Logistic Regression predicts the probability of occurrence of a binary event utilising a logit function.
- Linear Regression Equation:

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

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

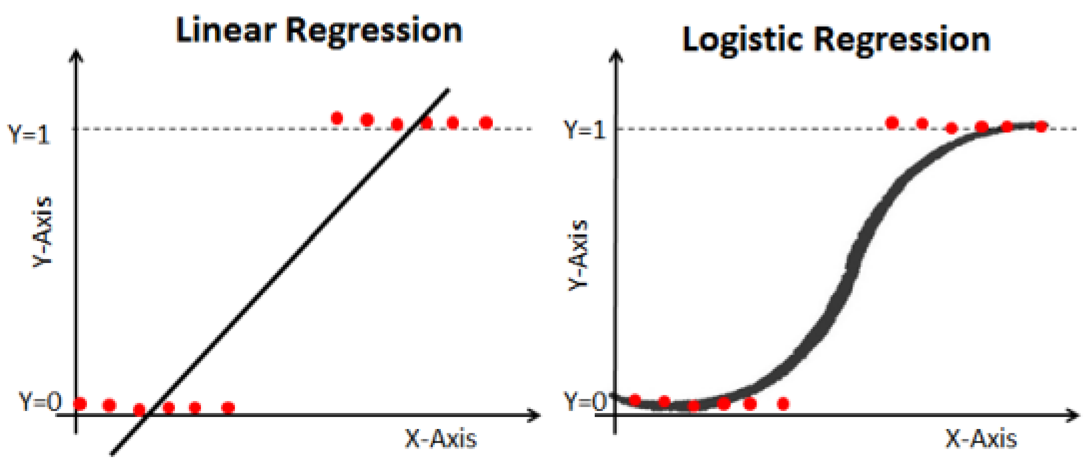
- Sigmoid Function:

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

Apply Sigmoid function on linear regression:

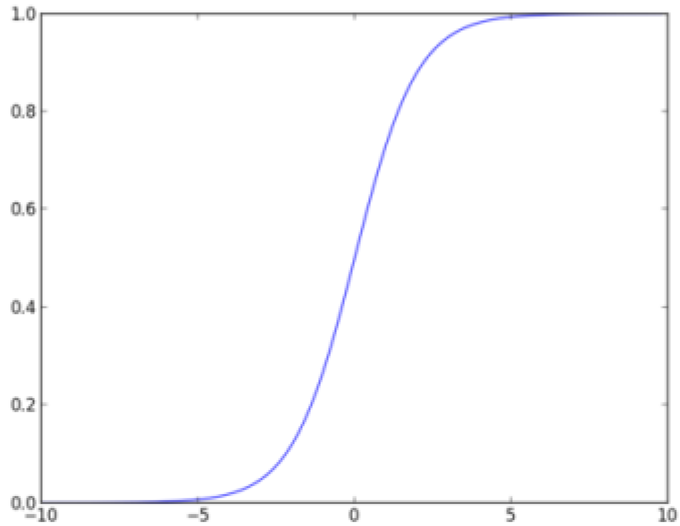
$$p = 1/1 + e^{-(\beta 0 + \beta 1 X1 + \beta 2 X2 \ldots\ldots \beta n Xn)}$$

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




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

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




- **Types of Logistic Regression**
- **Binary Logistic Regression:** The target variable has only two possible outcomes such as Spam or Not Spam, Cancer or No Cancer.
- **Multinomial Logistic Regression:** The target variable has three or more nominal categories such as predicting the type of Wine.
- **Ordinal Logistic Regression:** the target variable has three or more ordinal categories such as restaurant or product rating from 1 to 5.

The two limitations of using a linear regression model for classification problems are:

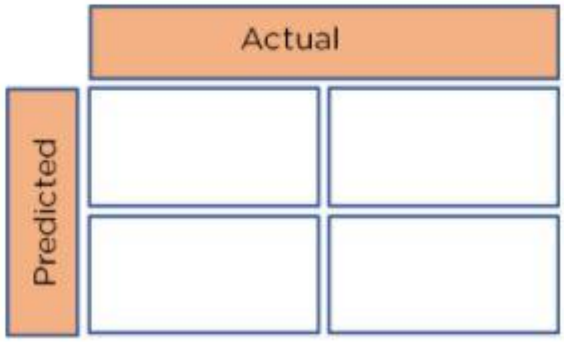
- the predicted value may exceed the range (0,1)
- error rate increases if the data has outliers

There definitely is a need for Logistic regression here.

- **Confusion Matrix Evaluation Metrics**

A confusion matrix presents a table layout of the different outcomes of the prediction and results of a classification problem and helps visualize its outcomes.

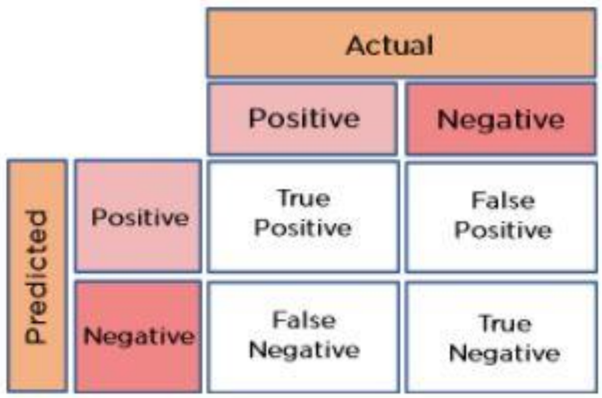
It plots a table of all the predicted and actual values of a classifier.




Basic layout of a Confusion Matrix

**How to Create a 2x2 Confusion Matrix?**

We can obtain four different combinations from the predicted and actual values of a classifier:




Confusion Matrix

- True Positive: The number of times our actual positive values are equal to the predicted positive. You predicted a positive value, and it is correct.
- False Positive: The number of times our model wrongly predicts negative values as positives. You predicted a negative value, and it is actually positive.
- True Negative: The number of times our actual negative values are equal to predicted negative values. You predicted a negative value, and it is actually negative.
- False Negative: The number of times our model wrongly predicts negative values as positives. You predicted a negative value, and it is actually positive.
- Accuracy: The accuracy is used to find the portion of correctly classified values. It tells us how often our classifier is right. It is the sum of all true values divided by total values.

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN}$$

- Precision: Precision is used to calculate the model's ability to classify positive values correctly. It is the true positives divided by the total number of predicted positive values.

$$\text{Precision} = \frac{TP}{TP + FP}$$

- Recall: It is used to calculate the model's ability to predict positive values. "How often does the model predict the correct positive values?". It is the true positives divided by the total number of actual positive values.

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

- F1-Score: It is the harmonic mean of Recall and Precision. It is useful when you need to take both Precision and Recall into account.

$$\text{F1-Score} = \frac{2 * \text{Precision} * \text{Recall}}{\text{Precision} + \text{Recall}}$$

**How does Logistic Regression Work?**

**Procedure & code:-**

## CONCLUSION:

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

## Lab Assignment 6

## Title: Data Analytics III

**PROBLEM STATEMENT:**

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.

# THEORY:

**Naive Bayes algorithm**

In machine learning, Naïve Bayes classification is a straightforward and powerful algorithm for the classification task. Naïve Bayes classification is based on applying Bayes' theorem with strong independence assumption between the features. Naïve Bayes classification produces good results when we use it for textual data analysis such as Natural Language Processing.

Naïve Bayes models are also known as simple Bayes or independent Bayes. All these names refer to the application of Bayes' theorem in the classifier's decision rule. Naïve Bayes classifier applies the Bayes' theorem in practice. This classifier brings the power of Bayes' theorem to machine learning.

**2. Naive Bayes algorithm intuition**

Naïve Bayes Classifier uses the Bayes' theorem to predict membership probabilities for each class such as the probability that given record or data point belongs to a particular class. The class with the highest probability is considered as the most likely class. This is also known as the **Maximum A Posteriori (MAP)**.

The **MAP for a hypothesis with 2 events A and B is**

**MAP (A)**

= max (P (A | B))

= max (P (B | A) * P (A))/P (B)

= max (P (B | A) * P (A))

Here, P (B) is evidence probability. It is used to normalize the result. It remains the same, So, removing it would not affect the result.

Naïve Bayes Classifier assumes that all the features are unrelated to each other. Presence or absence of a feature does not influence the presence or absence of any other feature.

**3. Types of Naive Bayes algorithm**

There are 3 types of Naïve Bayes algorithm. The 3 types are listed below:-

1. Gaussian Naïve Bayes
2. Multinomial Naïve Bayes
3. Bernoulli Naïve Bayes

**Gaussian Naïve Bayes algorithm**

When we have continuous attribute values, we made an assumption that the values associated with each class are distributed according to Gaussian or Normal distribution. For example, suppose the training data contains a continuous attribute x. We first segment the data by the class, and then compute the mean and variance of x in each class. Let μi be the mean of the values and let σi be the variance of the values associated with the ith class. Suppose we have some observation value xi . Then, the probability distribution of xi given a class can be computed by the following equation –

$$p(x_i|y_j) = \frac{1}{\sqrt{2\pi\sigma_j^2}} e^{-\frac{(x_i-\mu_j)^2}{2\sigma_j^2}}$$

**Multinomial Naïve Bayes algorithm**

With a Multinomial Naïve Bayes model, samples (feature vectors) represent the frequencies with which certain events have been generated by a multinomial (p1, . . . ,pn) where pi is the probability that event i occurs. Multinomial Naïve Bayes algorithm is preferred to use on data that is multinomially distributed. It is one of the standard algorithms which is used in text categorization classification.

**Bernoulli Naïve Bayes algorithm**

In the multivariate Bernoulli event model, features are independent boolean variables (binary variables) describing inputs. Just like the multinomial model, this model is also popular for document classification tasks where binary term occurrence features are used rather than term frequencies.

**4. Applications of Naive Bayes algorithm**

Naïve Bayes is one of the most straightforward and fast classification algorithm. It is very well suited for large volume of data. It is successfully used in various applications such as :

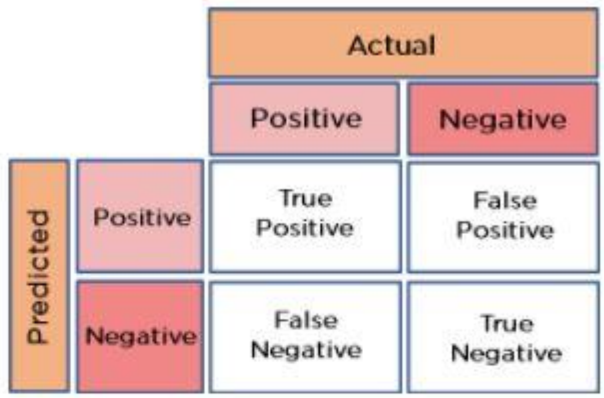
1. Spam filtering
2. Text classification
3. Sentiment analysis

4. Recommender systems

It uses Bayes theorem of probability for prediction of unknown class.

**Confusion Matrix?**

We can obtain four different combinations from the predicted and actual values of a classifier:




Confusion Matrix

- True Positive: The number of times our actual positive values are equal to the predicted positive. You predicted a positive value, and it is correct.
- False Positive: The number of times our model wrongly predicts negative values as positives. You predicted a negative value, and it is actually positive.
- True Negative: The number of times our actual negative values are equal to predicted negative values. You predicted a negative value, and it is actually negative.
- False Negative: The number of times our model wrongly predicts negative values as positives. You predicted a negative value, and it is actually positive.
- Accuracy: The accuracy is used to find the portion of correctly classified values. It tells us how often our classifier is right. It is the sum of all true values divided by total values.

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN}$$

- Precision: Precision is used to calculate the model's ability to classify positive values correctly. It is the true positives divided by the total number of predicted positive values.

$$\text{Precision} = \frac{TP}{TP + FP}$$

- Recall: It is used to calculate the model's ability to predict positive values. "How often does the model predict the correct positive values?". It is the true positives divided by the total number of actual positive values.

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

- F1-Score: It is the harmonic mean of Recall and Precision. It is useful when you need to take both Precision and Recall into account.

$$\text{F1-Score} = \frac{2 * \text{Precision} * \text{Recall}}{\text{Precision} + \text{Recall}}$$

## Procedure & Code:-

jupyter Experiment_No_6_Final Last Checkpoint: 03/31/2022 (autosaved)

In [4]:
```
# Importing the libraries
import numpy as np
import matplotlib.pyplot as plt
import matplotlib.image as mpimg
import pandas as pd
import seaborn as sns
```

In [5]:
```
# Load the data
#from google.colab import drive
#drive.mount('/content/drive/')
path="C:/Users/Admin/Desktop/DYPIEMR data/DSBDA lab/Exp-3/iris.csv"
data= pd.read_csv(path)
data.head()
```

Out[5]:

| | 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 |

In [6]:
```
#Spliting the dataset in independent and dependent variables
X = data.iloc[:,:4].values
```

jupyter Experiment_No_6_Final Last Checkpoint: 03/31/2022 (autosaved)

Logout

File Edit View Insert Cell Kernel Widgets Help

Not Trusted | Python 3 (ipykernel)

Run Code

| | | | | | | |
|---|---|---|---|---|---|---|
| 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 |

In [6]:
```
#Spliting the dataset in independent and dependent variables
X = data.iloc[:,:4].values
y = data['species'].values
```

In [7]:
```
# Splitting the dataset into the Training set and Test set
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.20, random_state = 8)
```

In [8]:
```
# Feature Scaling to bring the variable in a single scale
#from sklearn.preprocessing import StandardScaler
#sc = StandardScaler()
#X_train = sc.fit_transform(X_train)
#X_test = sc.transform(X_test)
```

In [9]:
```
# Fitting Naive Bayes Classification to the Training set with linear kernel
from sklearn.naive_bayes import GaussianNB
nvclassifier = GaussianNB()
nvclassifier.fit(X_train, y_train)
```

Out[9]:
```
GaussianNB()
```

jupyter Experiment_No_6_Final Last Checkpoint: 03/31/2022 (autosaved)

Logout

File Edit View Insert Cell Kernel Widgets Help

Not Trusted | Python 3 (ipykernel)

Run Code

In [10]:
```
# Predicting the Test set results
y_pred = nvclassifier.predict(X_test)
print(y_pred)
```

```
['Iris-setosa' 'Iris-setosa' 'Iris-setosa' 'Iris-virginica'
 'Iris-versicolor' 'Iris-setosa' 'Iris-setosa' 'Iris-virginica'
 'Iris-virginica' 'Iris-versicolor' 'Iris-versicolor' 'Iris-setosa'
 'Iris-versicolor' 'Iris-versicolor' 'Iris-virginica' 'Iris-virginica'
 'Iris-virginica' 'Iris-virginica' 'Iris-versicolor' 'Iris-virginica'
 'Iris-versicolor' 'Iris-setosa' 'Iris-versicolor' 'Iris-versicolor'
 'Iris-versicolor' 'Iris-setosa' 'Iris-virginica' 'Iris-setosa'
 'Iris-setosa' 'Iris-virginica']
```

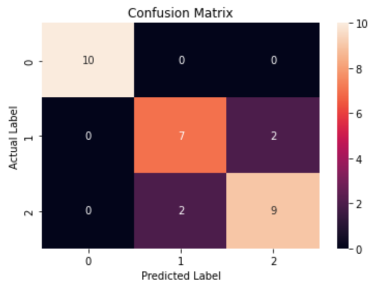
In [11]:
```
#lets see the actual and predicted value side by side
y_compare = np.vstack((y_test,y_pred)).T
#actual value on the left side and predicted value on the right hand side
#printing the top 5 values
y_compare[:5,:]
```

Out[11]:
```
array([['Iris-setosa', 'Iris-setosa'],
       ['Iris-setosa', 'Iris-setosa'],
       ['Iris-setosa', 'Iris-setosa'],
       ['Iris-virginica', 'Iris-virginica'],
       ['Iris-versicolor', 'Iris-versicolor']], dtype=object)
```

In [9]:
```
# Making the Confusion Matrix
from sklearn.metrics import confusion_matrix
cm = confusion_matrix(y_test, y_pred)
```

Experiment_No_6_Final Last Checkpoint: 03/31/2022 (autosaved)

```python
from sklearn.metrics import confusion_matrix
cm = confusion_matrix(y_test, y_pred)
sns.heatmap(cm,annot=True)
plt.xlabel("Predicted Label")
plt.ylabel("Actual Label")
plt.title("Confusion Matrix")
plt.show()
print(cm)
```



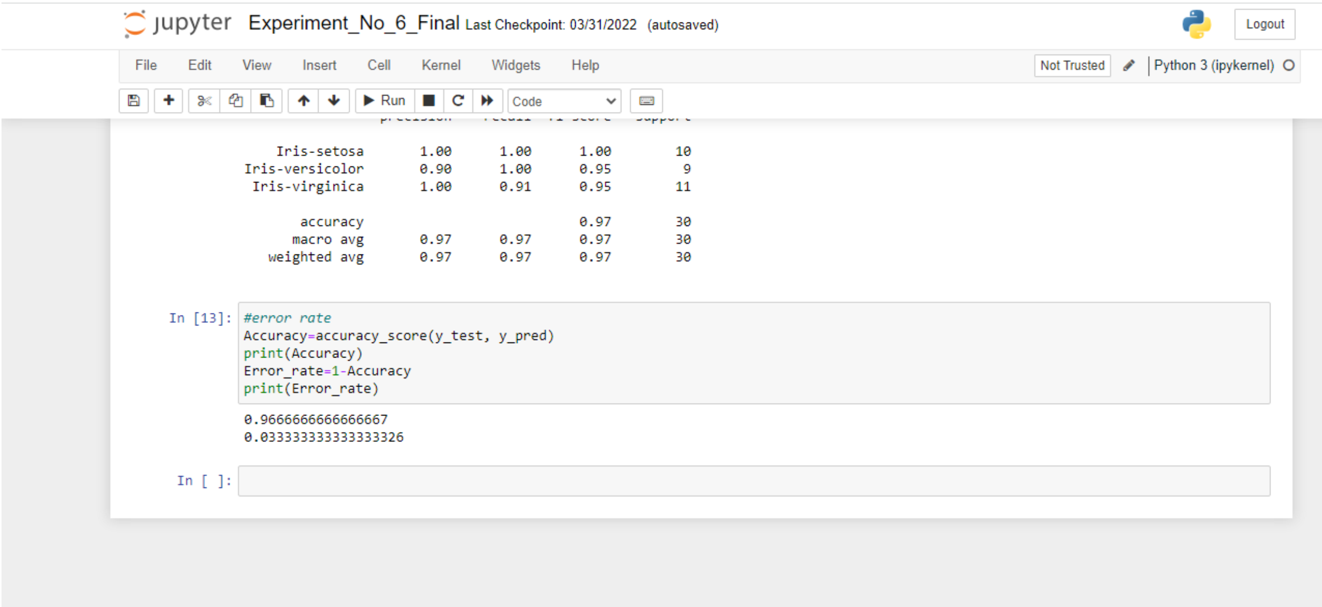

```
[[10  0  0]
 [ 0  7  2]
 [ 0  2  9]]
```

Experiment_No_6_Final Last Checkpoint: 03/31/2022 (autosaved)

In [14]:
```python
from sklearn.metrics import accuracy_score
from sklearn.metrics import classification_report
from sklearn.metrics import confusion_matrix
results = confusion_matrix(y_test, y_pred)
print ('Confusion Matrix :')
print(results)
print ('Accuracy Score :',accuracy_score(y_test, y_pred))
print('Classification Report : ')
print (classification_report(y_test, y_pred))
```

```
Confusion Matrix :
[[10  0  0]
 [ 0  9  0]
 [ 0  1 10]]
Accuracy Score : 0.9666666666666667
Classification Report :
                 precision    recall  f1-score   support

    Iris-setosa       1.00      1.00      1.00        10
Iris-versicolor       0.90      1.00      0.95         9
 Iris-virginica       1.00      0.91      0.95        11

       accuracy                           0.97        30
      macro avg       0.97      0.97      0.97        30
   weighted avg       0.97      0.97      0.97        30
```

In [13]:
```python
#error rate
```




| | | | | |
|---|---|---|---|---|
| Iris-setosa | 1.00 | 1.00 | 1.00 | 10 |
| Iris-versicolor | 0.90 | 1.00 | 0.95 | 9 |
| Iris-virginica | 1.00 | 0.91 | 0.95 | 11 |
| accuracy | | | 0.97 | 30 |
| macro avg | 0.97 | 0.97 | 0.97 | 30 |
| weighted avg | 0.97 | 0.97 | 0.97 | 30 |

In [13]:

```
#error rate
Accuracy=accuracy_score(y_test, y_pred)
print(Accuracy)
Error_rate=1-Accuracy
print(Error_rate)
```

```
0.9666666666666667
0.033333333333333326
```

In [ ]:

## CONCLUSION:

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

# Lab Assignment 7

# Title: Text Analytics

**PROBLEM STATEMENT:**

1. Extract Sample document and apply following document preprocessing methods: Tokenization, POS Tagging, stop words removal, Stemming and Lemmatization.

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

## THEORY:

**Basic concepts of Text Analytics**

- One of the most frequent types of day-to-day conversion is text communication. In our everyday routine, we chat, message, tweet, share status, email, create blogs, and offer opinions and criticism. All of these actions lead to a substantial amount of unstructured text being produced. It is critical to examine huge amounts of data in this sector of the online world and social media to determine people's opinions.
- Text mining is also referred to as text analytics. Text mining is a process of exploring sizable textual data and finding patterns. Text Mining processes the text itself, while NLP processes with the underlying metadata. Finding frequency counts of words, length of the sentence, presence/absence of specific words is known as text mining. Natural language processing is one of the components of text mining. NLP helps identify sentiment, finding entities in the sentence, and category of blog/article. Text mining is preprocessed data for text analytics. In Text Analytics, statistical and machine learning algorithms are used to classify information.

**Text Analysis Operations using natural language toolkit**

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

**Tokenization**:

- **Tokenization** is the first step in text analytics. The process of breaking down a text paragraph into smaller chunks such as words or sentences is called Tokenization. Token is a single entity that is the building blocks for a sentence or paragraph.
- **Sentence tokenization** : split a paragraph into list of sentences using sent_tokenize() method

**Stop words removal:**

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

**Stemming and Lemmatization**

- **Stemming** is a normalization technique where lists of tokenized words are converted into shortened root words to remove redundancy. Stemming is the process of reducing inflected (or sometimes derived) words to their word stem, base or root form. A computer program that stems word may be called a stemmer. E.g. A stemmer reduces the words like fishing, fished, and fisher to the stem fish. The stem need not be a word, for example the Porter algorithm reduces, argue, argued, argues, arguing, and argus to the stem argu .
- **Lemmatization** in NLTK is the algorithmic process of finding the lemma of a word depending on its meaning and context. Lemmatization usually refers to the morphological analysis of words, which aims to remove inflectional endings. It helps in returning the base or dictionary form of a word known as the lemma. Eg. Lemma for studies is study

**Lemmatization Vs Stemming**

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

**POS Tagging**

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

**Text Analysis Model using TF-IDF**

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

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

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

**Inverse Document Frequency (IDF)**

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

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

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

It is the product of TF and IDF. TFIDF gives more weight-age to the word that is rare in the corpus (all the documents). TFIDF provides more importance to the word that is more frequent in the document.

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

\

## PROCEDURE & CODE:-

jupyter Experiment No_7 - Final Last Checkpoint: 04/13/2022 (autosaved) Logout

File Edit View Insert Cell Kernel Widgets Help Not Trusted Python 3 (ipykernel)

Run Code

In [2]:

```
import nltk
from nltk.tokenize import word_tokenize,sent_tokenize
block = "CSI-DYPIEMR is the Student Chapter of Computer Society of India in Dr.D. Y. Patil Pratishthan's Dr. D. Y. Patil Institut
print("This is word wise tokenization-:",'\n', nltk.word_tokenize(block), '\n')
print("-----------------------------------------------------------------------------'\n'")
print("This is sentence wise tokenization-:",'\n', nltk.sent_tokenize(block))
```

```
This is word wise tokenization-:
 ['CSI-DYPIEMR', 'is', 'the', 'Student', 'Chapter', 'of', 'Computer', 'Society', 'of', 'India', 'in', 'Dr.D', '.', 'Y.', 'Pati
l', 'Pratishthan', "'s", 'Dr.', 'D.', 'Y.', 'Patil', 'Institute', 'of', 'Engineering', ',', 'Management', ',', 'and', 'Researc
h', '.', 'Computer', 'Society', 'of', 'India', 'is', 'a', 'body', 'of', 'computer', 'professionals', 'in', 'India', '.', 'It',
'was', 'started', 'on', '6', 'March', '1965', 'by', 'a', 'few', 'computer', 'professionals', 'and', 'has', 'now', 'grown', 't
o', 'be', 'the', 'national', 'body', 'representing', 'computer', 'professionals', '.', 'It', 'has', '72', 'chapters', 'across',
'India', ',', '511', 'student', 'branches', ',', 'and', '100,000', 'members', '.']

-----------------------------------------------------------------------------'
'
This is sentence wise tokenization-:
 ['CSI-DYPIEMR is the Student Chapter of Computer Society of India in Dr.D.', "Y. Patil Pratishthan's Dr. D. Y. Patil Institute
of Engineering,Management, and Research.", 'Computer Society of India is a body of computer professionals in India.', 'It was s
tarted on 6 March 1965 by a few computer professionals and has now grown to be the national body representing computer professi
onals.', 'It has 72 chapters across India, 511 student branches, and 100,000 members.']
```

```
In [3]: from nltk.corpus import stopwords
nltk.download('stopwords')
stop_words = stopwords.words('english')
print(stop_words)
token = nltk.word_tokenize(block)
cleaned_token = []
for word in token:
    if word not in stop_words:
        cleaned_token.append(word)
print("This is the unclean version-:",'\n', token, '\n')
print("-----------------------------------------------------------------------'\n'")
print("This is the cleaned version-:",'\n', cleaned_token)
```

```
['i', 'me', 'my', 'myself', 'we', 'our', 'ours', 'ourselves', 'you', "you're", "you've", "you'll", "you'd", 'your', 'yours', 'y
ourself', 'yourselves', 'he', 'him', 'his', 'himself', 'she', "she's", 'her', 'hers', 'herself', 'it', "it's", 'its', 'itself',
'they', 'them', 'their', 'theirs', 'themselves', 'what', 'which', 'who', 'whom', 'this', 'that', "that'll", 'these', 'those',
'am', 'is', 'are', 'was', 'were', 'be', 'been', 'being', 'have', 'has', 'had', 'having', 'do', 'does', 'did', 'doing', 'a', 'a
n', 'the', 'and', 'but', 'if', 'or', 'because', 'as', 'until', 'while', 'of', 'at', 'by', 'for', 'with', 'about', 'against', 'b
etween', 'into', 'through', 'during', 'before', 'after', 'above', 'below', 'to', 'from', 'up', 'down', 'in', 'out', 'on', 'of
f', 'over', 'under', 'again', 'further', 'then', 'once', 'here', 'there', 'when', 'where', 'why', 'how', 'all', 'any', 'both',
'each', 'few', 'more', 'most', 'other', 'some', 'such', 'no', 'nor', 'not', 'only', 'own', 'same', 'so', 'than', 'too', 'very',
's', 't', 'can', 'will', 'just', 'don', "don't", 'should', "should've", 'now', 'd', 'll', 'm', 'o', 're', 've', 'y', 'ain', 'ar
en', "aren't", 'couldn', "couldn't", 'didn', "didn't", 'doesn', "doesn't", 'hadn', "hadn't", 'hasn', "hasn't", 'haven', "have
n't", 'isn', "isn't", 'ma', 'mightn', "mightn't", 'mustn', "mustn't", 'needn', "needn't", 'shan', "shan't", 'shouldn', "should
n't", 'wasn', "wasn't", 'weren', "weren't", 'won', "won't", 'wouldn', "wouldn't"]
This is the unclean version-:
 ['CSI-DYPIEMR', 'is', 'the', 'Student', 'Chapter', 'of', 'Computer', 'Society', 'of', 'India', 'in', 'Dr.D', '.', 'Y.', 'Pati
l', 'Pratishthan', "'s", 'Dr.', 'D.', 'Y.', 'Patil', 'Institute', 'of', 'Engineering', ',', 'Management', ',', 'and', 'Researc
'India', ',', '511', 'student', 'branches', ',', 'and', '100,000', 'members', '.']

-----------------------------------------------------------------------'
'
This is the cleaned version-:
 ['CSI-DYPIEMR', 'Student', 'Chapter', 'Computer', 'Society', 'India', 'Dr.D', '.', 'Y.', 'Patil', 'Pratishthan', "'s", 'Dr.',
'D.', 'Y.', 'Patil', 'Institute', 'Engineering', ',', 'Management', ',', 'Research', '.', 'Computer', 'Society', 'India', 'bod
y', 'computer', 'professionals', 'India', '.', 'It', 'started', '6', 'March', '1965', 'computer', 'professionals', 'grown', 'na
tional', 'body', 'representing', 'computer', 'professionals', '.', 'It', '72', 'chapters', 'across', 'India', ',', '511', 'stud
ent', 'branches', ',', '100,000', 'members', '.']

[nltk_data] Downloading package stopwords to
[nltk_data]     C:\Users\Admin\AppData\Roaming\nltk_data...
[nltk_data]   Package stopwords is already up-to-date!
```

```
In [13]: from nltk.stem import PorterStemmer
stemmer = nltk.PorterStemmer()
words = ['rain', 'rained', 'raining', 'rains']
stemmed = [stemmer.stem(word) for word in words]
print(stemmed)
```

```
['rain', 'rain', 'rain', 'rain']
```

```
In [14]: from nltk.stem import WordNetLemmatizer
nltk.download('wordnet')#data dependencies
nltk.download('omw-1.4')
lemmatizer = nltk.WordNetLemmatizer()
lemmatized = [lemmatizer.lemmatize(word) for word in cleaned_token]
print(lemmatized)
```

```
[nltk_data] Downloading package wordnet to
[nltk_data]     C:\Users\Admin\AppData\Roaming\nltk_data...
[nltk_data]   Package wordnet is already up-to-date!
[nltk_data] Downloading package omw-1.4 to
[nltk_data]     C:\Users\Admin\AppData\Roaming\nltk_data...
[nltk_data]   Package omw-1.4 is already up-to-date!

['CSI-DYPIEMR', 'Student', 'Chapter', 'Computer', 'Society', 'India', 'Dr.D', '.', 'Y.', 'Patil', 'Pratishthan', "'s", 'Dr.',
'D.', 'Y.', 'Patil', 'Institute', 'Engineering', ',', 'Management', ',', 'Research', '.', 'Computer', 'Society', 'India', 'bod
y', 'computer', 'professional', 'India', '.', 'It', 'started', '6', 'March', '1965', 'computer', 'professional', 'grown', 'nati
onal', 'body', 'representing', 'computer', 'professional', '.', 'It', '72', 'chapter', 'across', 'India', ',', '511', 'studen
t', 'branch', ',', '100,000', 'member', '.']
```

In [15]:
```
from nltk import pos_tag
nltk.download('averaged_perceptron_tagger')
tagged = nltk.pos_tag(cleaned_token)
print(tagged)
```

```
[('CSI-DYPIEMR', 'JJ'), ('Student', 'NNP'), ('Chapter', 'NNP'), ('Computer', 'NNP'), ('Society', 'NNP'), ('India', 'NNP'), ('D
r.D', 'NNP'), ('.', '.'), ('Y.', 'NNP'), ('Patil', 'NNP'), ('Pratishthan', 'NNP'), ("'s", 'POS'), ('Dr.', 'NNP'), ('D.', 'NN
P'), ('Y.', 'NNP'), ('Patil', 'NNP'), ('Institute', 'NNP'), ('Engineering', 'NNP'), (',', ','), ('Management', 'NNP'), (',',
','), ('Research', 'NNP'), ('.', '.'), ('Computer', 'NNP'), ('Society', 'NNP'), ('India', 'NNP'), ('body', 'NN'), ('computer',
'NN'), ('professionals', 'NNS'), ('India', 'NNP'), ('.', '.'), ('It', 'PRP'), ('started', 'VBD'), ('6', 'CD'), ('March', 'NN
P'), ('1965', 'CD'), ('computer', 'NN'), ('professionals', 'NNS'), ('grown', 'VBP'), ('national', 'JJ'), ('body', 'NN'), ('repr
esenting', 'VBG'), ('computer', 'NN'), ('professionals', 'NNS'), ('.', '.'), ('It', 'PRP'), ('72', 'CD'), ('chapters', 'NNS'),
('across', 'IN'), ('India', 'NNP'), (',', ','), ('511', 'CD'), ('student', 'NN'), ('branches', 'NNS'), (',', ','), ('100,000',
'CD'), ('members', 'NNS'), ('.', '.')]
```

```
[nltk_data] Downloading package averaged_perceptron_tagger to
[nltk_data]     C:\Users\Admin\AppData\Roaming\nltk_data...
[nltk_data]   Package averaged_perceptron_tagger is already up-to-
[nltk_data]       date!
```

In [17]:
```
import pandas as pd
import sklearn as sk
import math
```

In [18]:
```
block_1 = "Our aim is to develop a good work culture among students, a culture where students from various technical backgrounds
block_2 = "Keeping in mind the interest of the IT professionals and computer enthusiasts, CSI works towards making the professior
#split so each word have their own string
first_block = block_1.split(" ")
second_block = block_2.split(" ")
#join them to remove common duplicate words
total= set(first_block).union(set(second_block))
print(total)
```

```
{'lectures,', 'society.', 'technical', 'from', 'students,', 'conventions,', 'training', 'together', 'top', 'conferences,', 'amo
ng', 'IT', 'for', 'regularly', 'grow', 'today.', 'all', 'towards', 'skill', 'awards.', 'of', 'collaborate', 'amongst', 'priorit
y', 'professionals.', 'sections', 'ensures', 'various', 'And', 'profession', 'on', 'To', 'choice', 'an', 'culture', 'where', 'u
pdating', 'CSI', 'promotion', 'develop', 'and', 'interest', 'Keeping', 'objective,the', 'teach,guide', 'are', 'enthusiasts,',
'it', 'good', 'students', 'area', 'in', 'mind', 'organizes', 'a', 'time,', 'other', 'Technology', 'projects', 'professionals',
'is', 'The', 'the', 'organized', 'with', 'future', 'regular', 'at', 'computer', 'come', 'projects,and', 'that', 'also', 'aim',
'as', 'works', 'each', 'this', 'work', 'fulfill', 'to', 'backgrounds', 'Our', 'together.', 'same', 'Information', 'making'}
```

In [19]:
```
wordDictA = dict.fromkeys(total, 0)
wordDictB = dict.fromkeys(total, 0)
for word in first_block:
    wordDictA[word]+=1
for word in second_block:
    wordDictB[word]+=1
```

In [20]:
```
pd.DataFrame([wordDictA, wordDictB])
```

Out[20]:

| | lectures, | society. | technical | from | students, | conventions, | training | together | top | conferences, | ... | this | work | fulfill | to | backgrounds | Our | together. | san |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **0** | 0 | 0 | 1 | 1 | 1 | 0 | 0 | 1 | 0 | 0 | ... | 0 | 1 | 0 | 2 | 1 | 1 | 1 | |
| **1** | 1 | 1 | 0 | 0 | 0 | 1 | 1 | 0 | 1 | 1 | ... | 1 | 0 | 1 | 0 | 0 | 0 | 0 | |

2 rows × 87 columns

In [11]:
```
import nltk
nltk.download('stopwords')
from nltk.corpus import stopwords
stop_words = set(stopwords.words('english'))
filtered_sentence = [w for w in wordDictA if not w in stop_words]
print(filtered_sentence)
```

```
['fulfill', 'future', 'interest', 'CSI', 'making', 'various', 'lectures,', 'training', 'students,', 'conferences,', 'aim', 'Tec
hnology', 'good', 'grow', 'together', 'works', 'The', 'awards.', 'regularly', 'develop', 'mind', 'towards', 'amongst', 'regula
r', 'projects,and', 'students', 'today.', 'technical', 'choice', 'skill', 'computer', 'teach,guide', 'backgrounds', 'area', 'wo
rk', 'priority', 'objective,the', 'updating', 'also', 'promotion', 'Our', 'profession', 'collaborate', 'conventions,', 'togethe
r.', 'To', 'Keeping', 'organizes', 'professionals', 'organized', 'top', 'time,', 'professionals.', 'enthusiasts,', 'culture',
'sections', 'Information', 'IT', 'projects', 'ensures', 'society.', 'come', 'And', 'among']
```

```
[nltk_data] Downloading package stopwords to
[nltk_data]     C:\Users\Admin\AppData\Roaming\nltk_data...
[nltk_data]   Package stopwords is already up-to-date!
```

In [25]:
```python
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_block)
tfSecond = computeTF(wordDictB, second_block)
tf = pd.DataFrame([tfFirst, tfSecond])
print(tf)
```

```
   lectures,  society.  technical      from  students,  conventions,  \
0   0.000000  0.000000   0.029412  0.029412   0.029412      0.000000
1   0.013158  0.013158   0.000000  0.000000   0.000000      0.013158

   training  together       top  conferences,  ...      this      work  \
0  0.000000  0.029412  0.000000      0.000000  ...  0.000000  0.029412
1  0.013158  0.000000  0.013158      0.013158  ...  0.013158  0.000000

    fulfill        to  backgrounds       Our  together.      same  \
0  0.000000  0.058824     0.029412  0.029412   0.029412  0.000000
1  0.013158  0.000000     0.000000  0.000000   0.000000  0.013158

   Information    making
0     0.000000  0.000000
1     0.013158  0.013158
```

In [26]:
```python
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)

idfs = computeIDF([wordDictA, wordDictB])
idfs1 = pd.DataFrame([wordDictA, wordDictB])
print(idfs1)
```

```
   lectures,  society.  technical  from  students,  conventions,  training  \
0          0         0          1     1          1             0         0
1          1         1          0     0          0             1         1

   together  top  conferences,  ...  this  work  fulfill  to  backgrounds  \
0         1    0             0  ...     0     1        0   2            1
1         0    1             1  ...     1     0        1   0            0

   Our  together.  same  Information  making
0    1          1     0            0       0
1    0          0     1            1       1

[2 rows x 87 columns]
```

```
In [23]: 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)
```

```
   lectures,  society.  technical      from  students,  conventions,  \
0   0.000000  0.000000   0.008854  0.008854   0.008854      0.000000
1   0.003961  0.003961   0.000000  0.000000   0.000000      0.003961

   training  together       top  conferences,  ...      this      work  \
0  0.000000  0.008854  0.000000      0.000000  ...  0.000000  0.008854
1  0.003961  0.000000  0.003961      0.003961  ...  0.003961  0.000000

    fulfill        to  backgrounds       Our  together.      same  \
0  0.000000  0.017708     0.008854  0.008854   0.008854  0.000000
1  0.003961  0.000000     0.000000  0.000000   0.000000  0.003961

   Information    making
0     0.000000  0.000000
1     0.003961  0.003961

[2 rows x 87 columns]
```

## CONCLUSION:

We have performed Text Analysis experiment using TF-IDF algorithm

## Lab Assignment 8

## Title: Data Visualization I

**PROBLEM STATEMENT:**

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.

# THEORY:

Data visualization is the graphical representation of information and data. By using visual elements like charts, graphs, and maps, data visualization tools provide an accessible way to see and understand trends, outliers, and patterns in data.

In the world of Big Data, data visualization tools and technologies are essential to analyze massive amounts of information and make data-driven decisions.

Common general types of data visualization:
Charts
Tables
Graphs
Maps
Infographics
Dashboards

More specific examples of methods to visualize data:
Area Chart
Bar Chart
Cartogram
Gantt Chart
Heat Map
Highlight Table
Histogram
Scatter Plot (2D or 3D)

Seaborn is a Python data visualization library based on matplotlib. It provides a high-level interface for drawing attractive and informative statistical graphics. Seaborn helps you explore and understand your data. Its plotting functions operate on dataframes and arrays containing whole datasets and internally perform the necessary semantic mapping and statistical aggregation to produce informative plots. Its dataset-oriented, declarative API lets you focus on what the different elements of your plots mean, rather than on the details of how to draw them.

By convention, it is imported with the shorthand sns.

```
# Import seaborn
import seaborn as sns
```

Behind the scenes, seaborn uses matplotlib to draw its plots. For interactive work, it's recommended to use a Jupyter/IPython interface in matplotlib mode, or else you'll have to call matplotlib.pyplot.show() when you want to see the plot.

Now, let's perform the operations in the problem statement on our data set.

Loading the dataset and libraries -

In [1]:

```
import pandas as pd
import numpy as np

import matplotlib.pyplot as plt
import seaborn as sns

dataset = sns.load_dataset('titanic')

dataset.head()
```

Out[1]:

| | survived | pclass | sex | age | sibsp | parch | fare | embarked | class | who | adult_ma |
|---|---|---|---|---|---|---|---|---|---|---|---|
| **0** | 0 | 3 | male | 22.0 | 1 | 0 | 7.2500 | S | Third | man | Tr |
| **1** | 1 | 1 | female | 38.0 | 1 | 0 | 71.2833 | C | First | woman | Fal |
| **2** | 1 | 3 | female | 26.0 | 0 | 0 | 7.9250 | S | Third | woman | Fal |
| **3** | 1 | 1 | female | 35.0 | 1 | 0 | 53.1000 | S | First | woman | Fal |
| **4** | 0 | 3 | male | 35.0 | 0 | 0 | 8.0500 | S | Third | man | Tr |

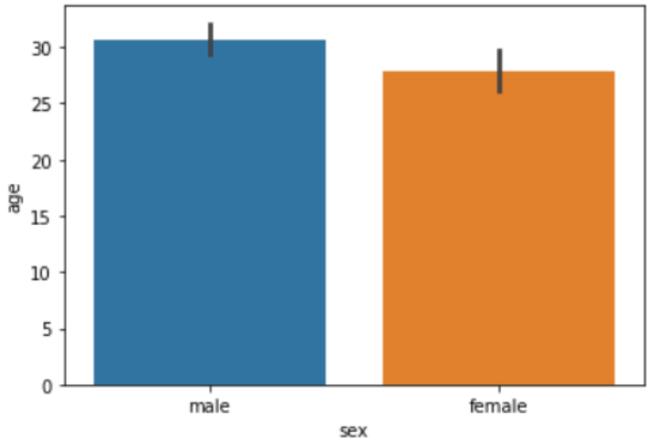
Some patterns can be seen by performing various operations like-

In [4]:

```
sns.barplot(x='sex', y='age', data=dataset)
```

Out[4]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x2329d50fe88>
```

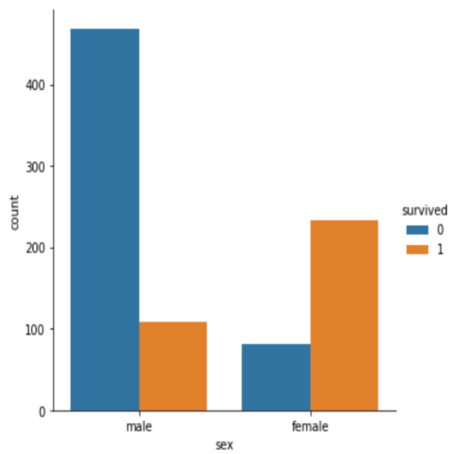



In [9]:

```
sns.catplot(x ="sex", hue ="survived",
kind ="count", data = dataset)
```

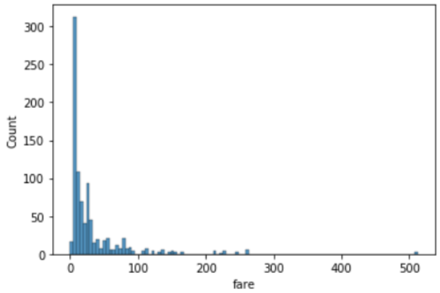
Out[9]:

```
<seaborn.axisgrid.FacetGrid at 0x24be16f4448>
```



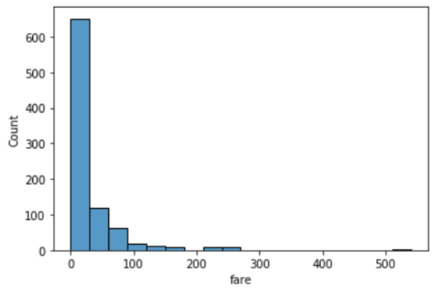

Assign a variable to x to plot a univariate distribution along the x axis:

```
In [5]: sns.histplot(data=dataset, x="fare")
Out[5]: <matplotlib.axes._subplots.AxesSubplot at 0x16d988db688>
```




Check how well the histogram represents the data by specifying a different bin width:

```
In [8]: sns.histplot(data=dataset, x="fare", binwidth=30)
Out[8]: <matplotlib.axes._subplots.AxesSubplot at 0x16d9f3b3788>
```




## CONCLUSION:

We have successfully implemented operations of the ‘seaborn’ library on the ‘titanic’ dataset, and explored some patterns in the data. We have also successfully plotted a histogram to see the ticket price distribution

# Lab Assignment 9

# Title: Data Visualization II

**PROBLEM STATEMENT:**

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.

## THEORY:

Data visualization is the graphical representation of information and data. By using visual elements like charts, graphs, and maps, data visualization tools provide an accessible way to see and understand trends, outliers, and patterns in data.

In the world of Big Data, data visualization tools and technologies are essential to analyze massive amounts of information and make data-driven decisions.

Common general types of data visualization:
Charts
Tables
Graphs
Maps
Infographics
Dashboards

More specific examples of methods to visualize data:
Area Chart
Bar Chart
Cartogram
Gantt Chart
Heat Map
Highlight Table
Histogram
Scatter Plot (2D or 3D)

Seaborn is a Python data visualization library based on matplotlib. It provides a high-level interface for drawing attractive and informative statistical graphics. Seaborn helps you explore and understand your data. Its plotting functions operate on dataframes and arrays containing whole datasets and internally perform the necessary semantic mapping and statistical aggregation to produce informative plots. Its dataset-oriented, declarative API lets you focus on what the different elements of your plots mean, rather than on the details of how to draw them.

By convention, it is imported with the shorthand sns.

```
# Import seaborn
import seaborn as sns
```

Behind the scenes, seaborn uses matplotlib to draw its plots. For interactive work, it's recommended to use a Jupyter/IPython interface in matplotlib mode, or else you'll have to call matplotlib.pyplot.show() when you want to see the plot.

Now, let's perform the operations in the problem statement on our data set.

Loading the dataset and libraries -

In [1]:

```
import pandas as pd
import numpy as np

import matplotlib.pyplot as plt
import seaborn as sns

dataset = sns.load_dataset('titanic')

dataset.head()
```

Out[1]:

| | survived | pclass | sex | age | sibsp | parch | fare | embarked | class | who | adult_ma |
|---|---|---|---|---|---|---|---|---|---|---|---|
| **0** | 0 | 3 | male | 22.0 | 1 | 0 | 7.2500 | S | Third | man | Tru |
| **1** | 1 | 1 | female | 38.0 | 1 | 0 | 71.2833 | C | First | woman | Fal: |
| **2** | 1 | 3 | female | 26.0 | 0 | 0 | 7.9250 | S | Third | woman | Fal: |
| **3** | 1 | 1 | female | 35.0 | 1 | 0 | 53.1000 | S | First | woman | Fal: |
| **4** | 0 | 3 | male | 35.0 | 0 | 0 | 8.0500 | S | Third | man | Tru |

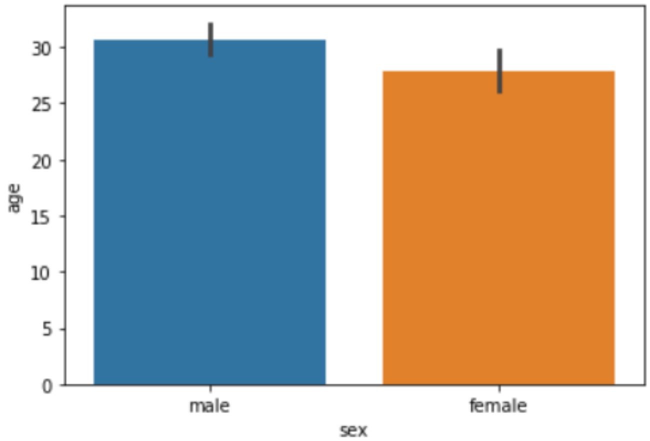
Some patterns can be seen by performing various operations like-

In [4]:

```
sns.barplot(x='sex', y='age', data=dataset)
```

Out[4]:

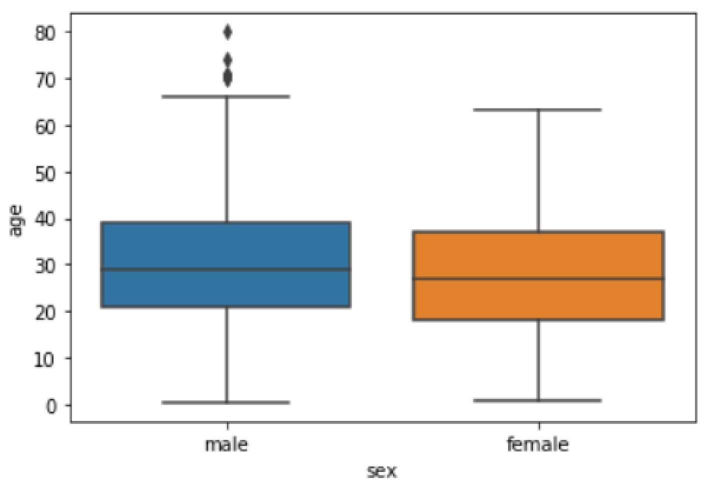
<matplotlib.axes._subplots.AxesSubplot at 0x2329d50fe88>




In [9]:

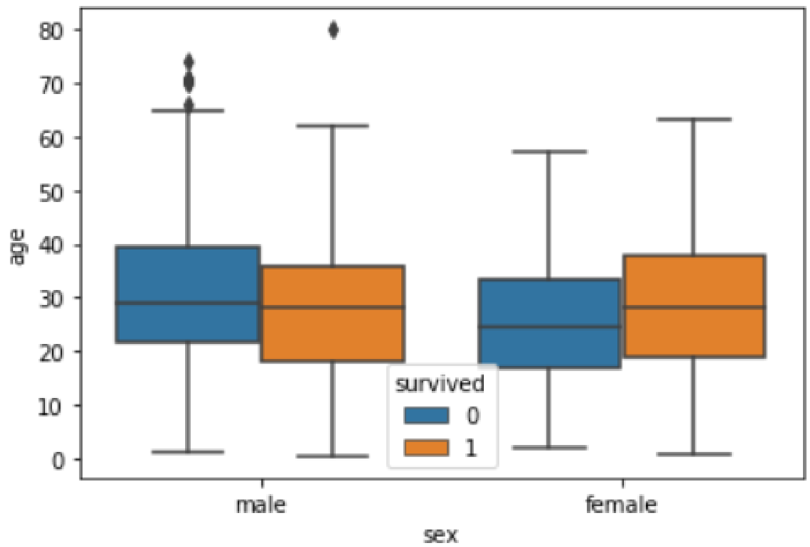
```
sns.boxplot(x='sex', y='age', data=dataset)
```

Out[9]: <matplotlib.axes._subplots.AxesSubplot at 0x16d9f439988>



```
In [10]: sns.boxplot(x='sex', y='age', data=dataset, hue="survived")
```

```
Out[10]: <matplotlib.axes._subplots.AxesSubplot at 0x16d9f4bc948>
```




Inferences -

Let's try to understand the box plot for female. The first quartile starts at around 5 and ends at 22 which means that 25% of the passengers are aged between 5 and 25. The second quartile starts at around 23 and ends at around 32 which means that 25% of the passengers are aged between 23 and 32. Similarly, the third quartile starts and ends between 34 and 42, hence 25% passengers are aged within this range and finally the fourth or last quartile starts at 43 and ends around 65.

If there are any outliers or the passengers that do not belong to any of the quartiles, they are called outliers and are represented by dots on the box plot.

Now in addition to the information about the age of each gender, you can also see the distribution of the passengers who survived. For instance, you can see that among the male passengers, on average more younger people survived as compared to the older ones. Similarly, you can see that the variation among the age of female passengers who did not survive is much greater than the age of the surviving female passengers.

## Conclusion:

We have successfully implemented operations of the ‘seaborn’ library on the ‘titanic’ dataset, and explored some patterns in the data. We have also successfully plotted a histogram to see the ticket price distribution

# Lab Assignment 10

# Title: Data Visualization III

**PROBLEM STATEMENT:**

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 boxplot for each feature in the dataset.

4. Compare distributions and identify outliers.

## THEORY:

## Histogram:-

Pandas.DataFrame.hist() function is useful in understanding the distribution of numeric variables. This function splits up the values into the numeric variables. Its main functionality is to make the Histogram of a given Data frame.

The distribution of data is represented by Histogram. When Function Pandas DataFrame.hist() is used, it automatically calls the function matplotlib.pyplot.hist() on each series in the DataFrame

Syntax: DataFrame.hist(data, column=None, by=None, grid=True, xlabelsize=None, xrot=None, ylabelsize=None, yrot=None, ax=None, sharex=False, sharey=False, figsize=None, layout=None, bins=10, backend=None, legend=False, **kwargs)
Parameters:
data: DataFrame
column: str or sequence
xlabelsize: int, default None
ylabelsize: int, default None
ax: Matplotlib axes object, default None
**kwargs
All other plotting keyword arguments to be passed to matplotlib.pyplot.hist().
Return:
matplotlib.AxesSubplot or numpy.ndarray

## Box Plot :-

*Box Plot* is the visual representation of the depicting groups of numerical data through their quartiles. Boxplot is also used for detect the outlier in data set. It captures the summary of the data efficiently with a simple box and whiskers and allows us to compare easily across groups. Boxplot summarizes a sample data using 25th, 50th and 75th percentiles. These percentiles are also known as the lower quartile, median and upper quartile.

A box plot consist of 5 things.

- Minimum
- First Quartile or 25%
- Median (Second Quartile) or 50%
- Third Quartile or 75%
- Maximum

Draw the boxplot using seaborn library:

Syntax : seaborn.boxplot(x=None, y=None, hue=None, data=None, order=None, hue_order=None, orient=None, color=None, palette=None, saturation=0.75, width=0.8, dodge=True, fliersize=5, linewidth=None, whis=1.5, notch=False, ax=None, **kwargs)

Parameters:
x = feature of dataset
y = feature of dataset
hue = feature of dataset
data = dataframe or full dataset
color = color name

## Identify outliers:-

**Detect and Remove the Outliers using Python**

An Outlier is a data-item/object that deviates significantly from the rest of the (so-called normal)objects. They can be caused by measurement or execution errors. The analysis for outlier detection is referred to as outlier mining. There are many ways to detect the outliers, and the removal process is the data frame same as removing a data item from the panda's data frame.

Here pandas data frame is used for a more realistic approach as in real-world project need to detect the outliers arouse during the data analysis step, the same approach can be used on lists and series-type objects.

## Detecting the outliers

Outliers can be detected using visualization, implementing mathematical formulas on the dataset, or using the statistical approach. All of these are discussed below.

1. Visualization

- **Using Box Plot**
- **Using ScatterPlot**
- **Z-score**

Z- Score is also called a standard score. This value/score helps to understand that 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*

- **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*

## Procedure & Code:-

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

In [2]:
```
data1 = pd.read_csv("C:/Users/Admin/Downloads/Iris.csv")
data1.head()
```

Out[2]:

| | 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 |

In [3]:
```
print(data1.columns)
```

```
Index(['Id', 'SepalLengthCm', 'SepalWidthCm', 'PetalLengthCm', 'PetalWidthCm',
       'Species'],
      dtype='object')
```

In [7]:
```
data1.info()
```

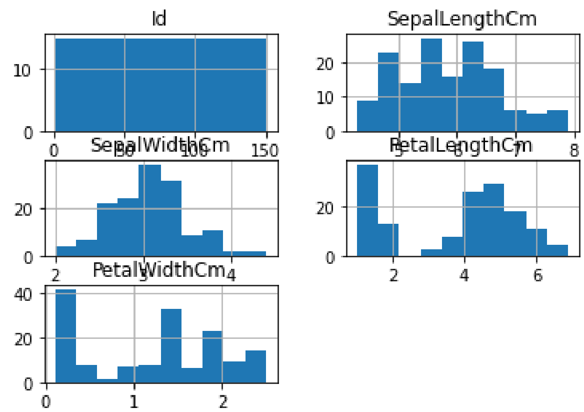
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 150 entries, 0 to 149
Data columns (total 6 columns):
 #   Column         Non-Null Count  Dtype  
---  ------         --------------  -----  
 0   Id             150 non-null    int64  
 1   SepalLengthCm  150 non-null    float64
 2   SepalWidthCm   150 non-null    float64
 3   PetalLengthCm  150 non-null    float64
 4   PetalWidthCm   150 non-null    float64
 5   Species        150 non-null    object 
dtypes: float64(4), int64(1), object(1)
memory usage: 7.2+ KB
```

In [13]:
```
data1.dtypes
```

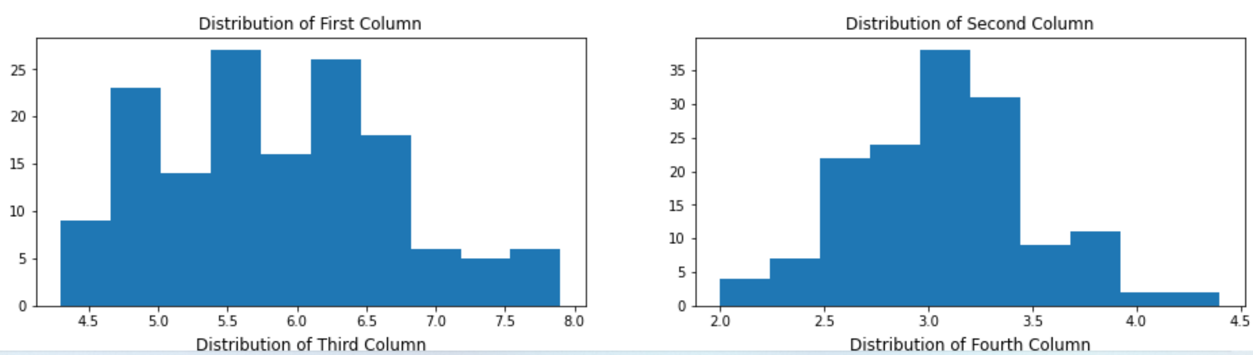
Out[13]:
```
Id                 int64
SepalLengthCm    float64
SepalWidthCm     float64
PetalLengthCm    float64
PetalWidthCm     float64
Species           object
dtype: object
```

```
In [20]: data1.hist()
```

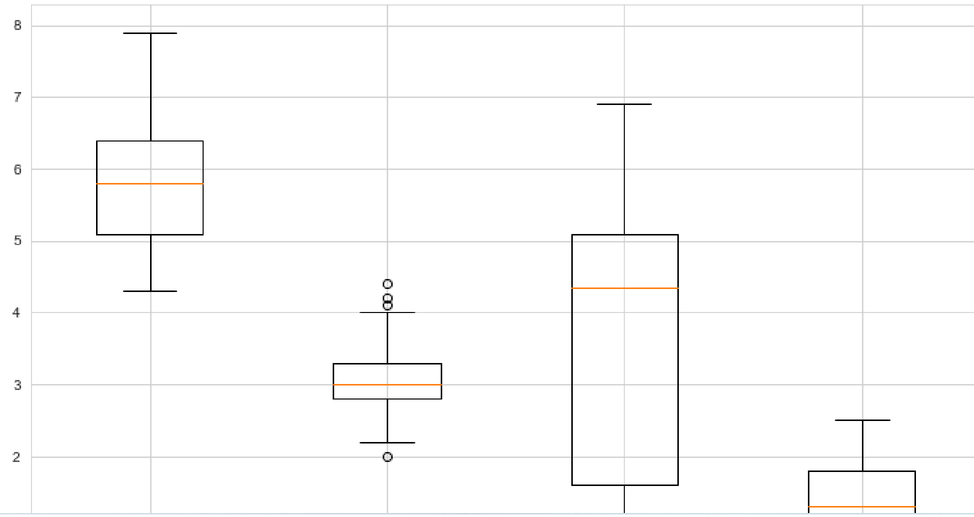
```
Out[20]: array([[<AxesSubplot:title={'center':'Id'}>,
                 <AxesSubplot:title={'center':'SepalLengthCm'}>],
                [<AxesSubplot:title={'center':'SepalWidthCm'}>,
                 <AxesSubplot:title={'center':'PetalLengthCm'}>],
                [<AxesSubplot:title={'center':'PetalWidthCm'}>, <AxesSubplot:>]],
               dtype=object)
```



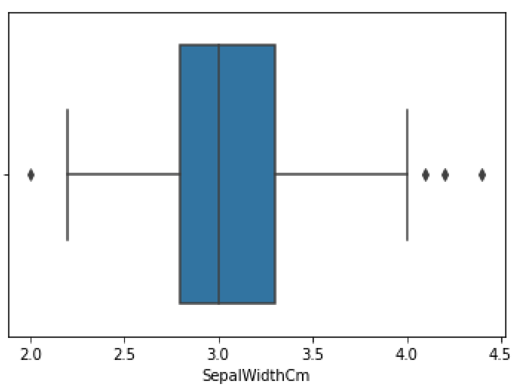

```
In [17]: fig, axes = plt.subplots(2, 2, figsize=(16, 8))

         axes[0,0].set_title("Distribution of First Column")
         axes[0,0].hist(data1["SepalLengthCm"]);

         axes[0,1].set_title("Distribution of Second Column")
         axes[0,1].hist(data1["SepalWidthCm"]);

         axes[1,0].set_title("Distribution of Third Column")
         axes[1,0].hist(data1["PetalLengthCm"]);

         axes[1,1].set_title("Distribution of Fourth Column")
         axes[1,1].hist(data1["PetalWidthCm"]);
```



```
In [39]: data_to_plot = [data1["SepalLengthCm"],data1["SepalWidthCm"],data1["PetalLengthCm"],data1["PetalWidthCm"]]
         # 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);
```




```
In [19]: sns.boxplot(data1['SepalWidthCm'])

C:\ProgramData\Anaconda3\lib\site-packages\seaborn\_decorators.py:36: FutureWarning: Pass the following variable
rg: x. From version 0.12, the only valid positional argument will be `data`, and passing other arguments without
yword will result in an error or misinterpretation.
  warnings.warn(

Out[19]: <AxesSubplot:xlabel='SepalWidthCm'>
```




```
In [36]: print(np.where(data1['SepalWidthCm']>4.0))

(array([15, 32, 33], dtype=int64),)
```

## CONCLUSION:

We have successfully implemented operations on the ‘iris’ dataset, also we have successfully plotted a histogram, boxplot and identified outliers.

## Lab Assignment 11

## Title: Big Data Analytics

**PROBLEM STATEMENT:**

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

## THEORY:

MapReduce is a framework using which we can write applications to process huge amounts of data, in parallel, on large clusters of commodity hardware in a reliable manner.

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.

In Hadoop, MapReduce is a computation that decomposes large manipulation jobs into individual tasks that can be executed in parallel across a cluster of servers. The results of tasks can be joined together to compute final results.

**MapReduce consists of 2 steps:**

- **Map Function –** It takes a set of data and converts it into another set of data, where individual elements are broken down into tuples (Key-Value pair).

  **Example –** (Map function in Word Count)

| | | |
|---|---|---|
| **Input** | Set of data | Bus, Car, bus, car, train, car, bus, car, train, bus, TRAIN,BUS, buS, caR, CAR, car, BUS, TRAIN |
| **Output** | Convert into another set of data<br>(Key,Value) | (Bus,1), (Car,1), (bus,1), (car,1), (train,1),<br>(car,1), (bus,1), (car,1), (train,1), (bus,1),<br>(TRAIN,1),(BUS,1), (buS,1), (caR,1), (CAR,1), |

| | | |
|---|---|---|
| | | (car,1), (BUS,1), (TRAIN,1) |

- **Reduce Function –** Takes the output from Map as an input and combines those data tuples into a smaller set of tuples.

  **Example –** (Reduce function in Word Count)

| | | |
|---|---|---|
| **Input**<br><br>**(output of Map function)** | Set of Tuples | (Bus,1), (Car,1), (bus,1), (car,1), (train,1),<br><br>(car,1), (bus,1), (car,1), (train,1), (bus,1),<br><br>(TRAIN,1),(BUS,1), (buS,1), (caR,1), (CAR,1),<br><br>(car,1), (BUS,1), (TRAIN,1) |
| **Output** | Converts into smaller set of tuples | (BUS,7),<br><br>(CAR,7),<br><br>(TRAIN,4) |

## Work Flow of the Program

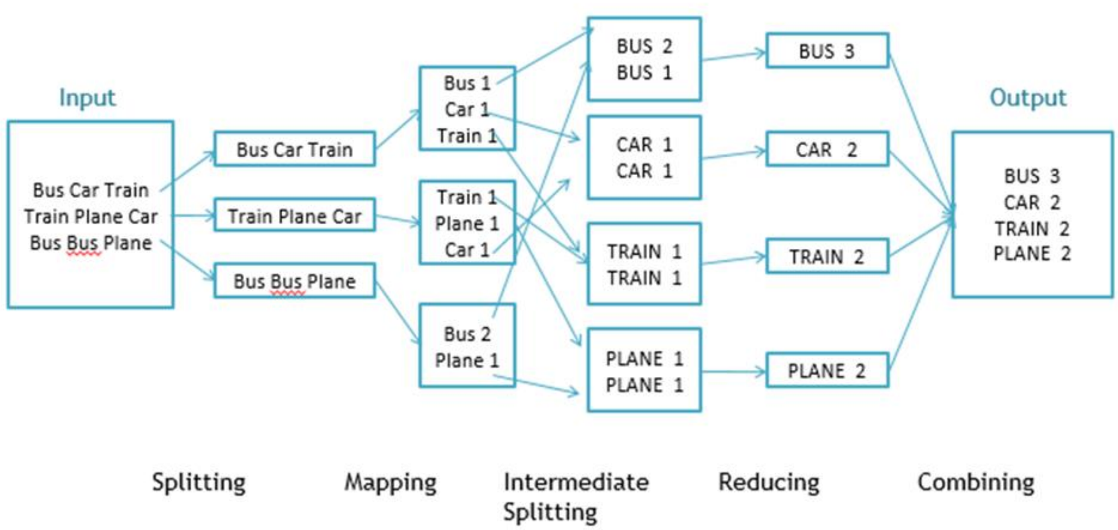



Fig. WorkFlow of MapReducing

Workflow of MapReduce consists of 5 steps:

1. **Splitting** – The splitting parameter can be anything, e.g. splitting by space, comma, semicolon, or even by a new line ('\n').
2. **Mapping** – as explained above.
3. **Intermediate splitting** – the entire process in parallel on different clusters. In order to group them in "Reduce Phase" the similar KEY data should be on the same cluster.
4. **Reduce** – it is nothing but mostly group by phase.
5. **Combining** – The last phase where all the data (individual result set from each cluster) is combined together to form a result.

## Procedure & Code:-

```
package PackageDemo;

import java.io.IOException;

import org.apache.hadoop.conf.Configuration;

import org.apache.hadoop.fs.Path;

import org.apache.hadoop.io.IntWritable;

import org.apache.hadoop.io.LongWritable;

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;

import org.apache.hadoop.util.GenericOptionsParser;

public class WordCount {

public static void main(String [] args) throws Exception

{

Configuration c=new Configuration();

String[] files=new GenericOptionsParser(c,args).getRemainingArgs();

Path input=new Path(files[0]);
```

```
Path output=new Path(files[1]);
Job j=new Job(c,"wordcount");
j.setJarByClass(WordCount.class);
j.setMapperClass(MapForWordCount.class);
j.setReducerClass(ReduceForWordCount.class);
j.setOutputKeyClass(Text.class);
j.setOutputValueClass(IntWritable.class);
FileInputFormat.addInputPath(j, input);
FileOutputFormat.setOutputPath(j, output);
System.exit(j.waitForCompletion(true)?0:1);
}
public static class MapForWordCount extends Mapper<LongWritable, Text, Text, IntWritable>{
public void map(LongWritable key, Text value, Context con) throws IOException, InterruptedException
{
String line = value.toString();
String[] words=line.split(",");
for(String word: words )
{
     Text outputKey = new Text(word.toUpperCase().trim());
  IntWritable outputValue = new IntWritable(1);
  con.write(outputKey, outputValue);
}
}
}
public static class ReduceForWordCount extends Reducer<Text, IntWritable, Text, IntWritable>
{
public void reduce(Text word, Iterable<IntWritable> values, Context con) throws IOException,
InterruptedException
```

```
{
int sum = 0;
  for(IntWritable value : values)
  {
  sum += value.get();
  }
  con.write(word, new IntWritable(sum));
}
}
}
```

## CONCLUSION:

Implemented code in JAVA for a simple Word Count application that counts the number of occurrences of each word in a given input set using the Hadoop Map-Reduce

# Lab Assignment 12

# Title: Big Data Analytics

## PROBLEM STATEMENT:

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.

## THEORY:

Weather and Climate data are valuable resources, not only to forecast the weather but for dozens of purposes across industries, governments, and in personal life. Opportunities to apply data analytics and data science techniques to these data to inform and support decision-making are almost endless.

We write a program for analyzing weather datasets to understand its data processing programming model. 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, etc. for easiness, we will focus on the main element, i.e. temperature. 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.

In [ ]:
```
# Imports
```

In [1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from importlib import reload
plt=reload(plt)
```

In [ ]:
```
# Taking input file
```

In [5]:
```
df=pd.read_csv("C:/Users/Admin/Downloads/dataset weather/weatherHistory.csv")
df.head()
```

Out[5]:

| | Formatted Date | Summary | Precip Type | Temperature (C) | Apparent Temperature (C) | Humidity | Wind Speed (km/h) | Wind Bearing (degrees) | Visibility (km) | Loud Cover | Pressure (millibars) | Daily Summary |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 2006-04-01 00:00:00.000 +0200 | Partly Cloudy | rain | 9.472222 | 7.388889 | 0.89 | 14.1197 | 251.0 | 15.8263 | 0.0 | 1015.13 | Partly cloudy throughout the day. |
| 1 | 2006-04-01 01:00:00.000 +0200 | Partly Cloudy | rain | 9.355556 | 7.227778 | 0.86 | 14.2646 | 259.0 | 15.8263 | 0.0 | 1015.63 | Partly cloudy throughout the day. |
| 2 | 2006-04-01 02:00:00.000 | Mostly | rain | 9.377778 | 9.377778 | 0.89 | 3.9284 | 204.0 | 14.9569 | 0.0 | 1015.94 | Partly cloudy throughout the |

In [8]:
```python
columns_order=["Date-Time","TZ","Summary","Precip Type","Temperature (C)","Apparent Temperature (C)",
               "Humidity","Wind Speed (km/h)","Wind Bearing (degrees)","Visibility (km)","Loud Cover",
               "Pressure (millibars)", "Daily Summary"]
df2=df1.reindex(columns=columns_order)
df3=df2.drop(columns="TZ")
df3.head()
```

Out[8]:

| | Date-Time | Summary | Precip Type | Temperature (C) | Apparent Temperature (C) | Humidity | Wind Speed (km/h) | Wind Bearing (degrees) | Visibility (km) | Loud Cover | Pressure (millibars) | Daily Summary |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 2006-04-01 00:00:00.000 | Partly Cloudy | rain | 9.472222 | 7.388889 | 0.89 | 14.1197 | 251.0 | 15.8263 | 0.0 | 1015.13 | Partly cloudy throughout the day |
| 1 | 2006-04-01 01:00:00.000 | Partly Cloudy | rain | 9.355556 | 7.227778 | 0.86 | 14.2646 | 259.0 | 15.8263 | 0.0 | 1015.63 | Partly cloudy throughout the day |
| 2 | 2006-04-01 02:00:00.000 | Mostly Cloudy | rain | 9.377778 | 9.377778 | 0.89 | 3.9284 | 204.0 | 14.9569 | 0.0 | 1015.94 | Partly cloudy throughout the day |
| 3 | 2006-04-01 03:00:00.000 | Partly Cloudy | rain | 8.288889 | 5.944444 | 0.83 | 14.1036 | 269.0 | 15.8263 | 0.0 | 1016.41 | Partly cloudy throughout the day |
| 4 | 2006-04-01 04:00:00.000 | Mostly Cloudy | rain | 8.755556 | 6.977778 | 0.83 | 11.0446 | 259.0 | 15.8263 | 0.0 | 1016.51 | Partly cloudy throughout the day |

In [9]:
```python
df3["Date-Time"]=pd.to_datetime(df3["Date-Time"])
df3.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 96453 entries, 0 to 96452
Data columns (total 12 columns):
 #   Column                    Non-Null Count  Dtype
---  ------                    --------------  -----
 0   Date-Time                 96453 non-null  datetime64[ns]
 1   Summary                   96453 non-null  object
 2   Precip Type               95936 non-null  object
 3   Temperature (C)           96453 non-null  float64
 4   Apparent Temperature (C)  96453 non-null  float64
 5   Humidity                  96453 non-null  float64
 6   Wind Speed (km/h)         96453 non-null  float64
 7   Wind Bearing (degrees)    96453 non-null  float64
 8   Visibility (km)           96453 non-null  float64
 9   Loud Cover                96453 non-null  float64
 10  Pressure (millibars)      96453 non-null  float64
 11  Daily Summary             96453 non-null  object
dtypes: datetime64[ns](1), float64(8), object(3)
memory usage: 8.8+ MB
```

In [10]:
```python
df3["Year"]=pd.DatetimeIndex(df3["Date-Time"]).year
df3["Month"]=df3["Date-Time"].dt.month_name()
df3["day"]=df3["Date-Time"].dt.day
df3.head()
```

Out[10]:

| | Date-Time | Summary | Precip Type | Temperature (C) | Apparent Temperature (C) | Humidity | Wind Speed (km/h) | Wind Bearing (degrees) | Visibility (km) | Loud Cover | Pressure (millibars) | Daily Summary | Year | Month | day |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 2006-04-01 00:00:00 | Partly Cloudy | rain | 9.472222 | 7.388889 | 0.89 | 14.1197 | 251.0 | 15.8263 | 0.0 | 1015.13 | Partly cloudy throughout the day. | 2006 | April | 1 |
| 1 | 2006-04-01 01:00:00 | Partly Cloudy | rain | 9.355556 | 7.227778 | 0.86 | 14.2646 | 259.0 | 15.8263 | 0.0 | 1015.63 | Partly cloudy throughout the day. | 2006 | April | 1 |
| 2 | 2006-04-01 02:00:00 | Mostly Cloudy | rain | 9.377778 | 9.377778 | 0.89 | 3.9284 | 204.0 | 14.9569 | 0.0 | 1015.94 | Partly cloudy throughout the day. | 2006 | April | 1 |
| 3 | 2006-04-01 03:00:00 | Partly Cloudy | rain | 8.288889 | 5.944444 | 0.83 | 14.1036 | 269.0 | 15.8263 | 0.0 | 1016.41 | Partly cloudy throughout the day. | 2006 | April | 1 |
| 4 | 2006-04-01 04:00:00 | Mostly Cloudy | rain | 8.755556 | 6.977778 | 0.83 | 11.0446 | 259.0 | 15.8263 | 0.0 | 1016.51 | Partly cloudy throughout the day. | 2006 | April | 1 |

```
Name: Wind Speed (km/h), dtype: float64
```

In [13]:
```
avg_wind_Speed=pd.DataFrame(df3.groupby("Year")["Wind Speed (km/h)"].mean())
avg_wind_Speed
```

Out[13]:

| Year | Wind Speed (km/h) |
|---|---|
| 2006 | 10.189852 |
| 2007 | 10.825392 |
| 2008 | 11.303897 |
| 2009 | 11.505948 |
| 2010 | 11.015628 |
| 2011 | 9.898262 |
| 2012 | 11.264545 |
| 2013 | 10.969389 |
| 2014 | 10.502473 |
| 2015 | 10.735247 |
| 2016 | 10.703441 |

In [15]:
```
month_avg_wind_Speed=pd.DataFrame(df3.groupby("Month")["Wind Speed (km/h)"].mean())
order=["January","February","March","April","May","June","July","August","September",
            "October","November","December"]
monthly_wind_speed=month_avg_wind_Speed.reindex(index=order)
monthly_wind_speed
```

Out[15]:

| Month | Wind Speed (km/h) |
|---|---|
| January | 11.512816 |
| February | 12.185543 |
| March | 13.405461 |
| April | 11.893094 |
| May | 10.959337 |
| June | 9.626471 |
| July | 9.639907 |
| August | 8.933431 |
| September | 9.621813 |
| October | 10.000153 |
| November | 10.944266 |
| December | 11.098682 |

In [25]:
```
year_avg_temp=pd.DataFrame(df3.groupby("Year")["Temperature (C)"].mean())
year_avg_temp
```

Out[25]:

| Year | Temperature (C) |
|---|---|
| 2006 | 11.215365 |
| 2007 | 12.135239 |
| 2008 | 12.161876 |
| 2009 | 12.267910 |
| 2010 | 11.202061 |
| 2011 | 11.524453 |
| 2012 | 11.986726 |
| 2013 | 11.940719 |
| 2014 | 12.529737 |
| 2015 | 12.311370 |
| 2016 | 11.985292 |

## CONCLUSION:

Students have successfully calculated average of temperature, dew point and wind speed over temperature data.

# Lab Assignment 13

# Title: Big Data Analytics

**PROBLEM STATEMENT:**

Write a simple program in SCALA using Apache Spark framework

## THEORY:

**What is Scala?-**

Scala is an acronym for “Scalable Language”. It is a general-purpose programming language designed for the programmers who want to write programs in a concise, elegant, and type-safe way. Scala enables programmers to be more productive. Scala is developed as an object-oriented and functional programming language.

If you write a code in Scala, you will see that the style is similar to a scripting language. Even though Scala is a new language, it has gained enough users and has a wide community support. It is one of the most user-friendly languages.

**Scala is pure Object-Oriented programming language**

Scala is an object-oriented programming language. Everything in Scala is an object and any operations you perform is a method call. Scala, allow you to add new operations to existing classes with the help of implicit classes.One of the advantages of Scala is that it makes it very easy to interact with Java code. You can also write a Java code inside Scala class. The Scala supports advanced component architectures through classes and traits.

**Scala is a functional language**

Scala is a programming language that has implemented major functional programming concepts. In Functional programming, every computation is treated as a mathematical function which avoids states and mutable data. The functional programming exhibits following characteristics:

- Power and flexibility
- Simplicity
- Suitable for parallel processing

**Installing Scala**

Scala can be installed in any Unix or windows based system. Below are the steps to install for Ubuntu (14.04) for scala version 2.11.7. I am showing the steps for installing Scala (2.11.7) with Java version 7. It is necessary to install Java before installing Scala. You can also install latest version of Scala(2.12.1) as well.

**Step 0:** Open the terminal

**Step 1:** Install Java

```
$ sudo apt-add-repository ppa:webupd8team/java
$ sudo apt-get update
$ sudo apt-get install oracle-java7-installer
```

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:

```
$ java -version
```

**Step 2:** Once Java is installed, we need to install Scala

```
$ cd ~/Downloads
$ wget http://www.scala-lang.org/files/archive/scala-2.11.7.deb
$ sudo dpkg -i scala-2.11.7.deb
$ scala –version
```

## Scala Basics 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.

## Procedure & Code:-

**SIMPLE SCALA PROGRAM:-**

**/ SCALA PROGRAM TO PRINT HELLO, WORLD!**

**// BY USING OBJECT-ORIENTED APPROACH**

```
OBJECT HELLOWORLD {
 DEF MAIN(ARGS: ARRAY[STRING]) {
 PRINTLN("HELLO, WORLD!")
 }
 }
```

## Compile a Scala Program

To run any Scala program, you first need to compile it. “Scalac” is the compiler which takes source program as an argument and generates object files as output.

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.
scalac HelloWorld.scala

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.

### Running Scala Program

After compiling, you can now run the program using following command:
scala HelloWorld

## CONCLUSION:

Students have installed and executed Scala Program successfully.

# Lab Assignment 14

## Title: Mini Projects/ Case Study

**PROBLEM STATEMENT:**

Use the following dataset and classify tweets into positive and negative tweets. https://www.kaggle.com/ruchi798/data-science-tweets

## THEORY:

It is a Natural Language Processing Problem where Sentiment Analysis is done by Classifying the Positive tweets from negative tweets by machine learning models for classification, text mining, text analysis, data analysis and data visualization

## Introduction

Natural Language Processing (NLP) is a hotbed of research in data science these days and one of the most common applications of NLP is sentiment analysis. From opinion polls to creating entire marketing strategies, this domain has completely reshaped the way businesses work, which is why this is an area every data scientist must be familiar with.

Thousands of text documents can be processed for sentiment (and other features including named entities, topics, themes, etc.) in seconds, compared to the hours it would take a team of people to manually complete the same task.

We will do so by following a sequence of steps needed to solve a general sentiment analysis problem. We will start with preprocessing and cleaning of the raw text of the tweets. Then we will explore the cleaned text and try to get some intuition about the context of the tweets. After that, we will extract numerical features from the data and finally use these feature sets to train models and identify the sentiments of the tweets.

## Tweets Preprocessing and Cleaning

The data cleaning exercise is quite similar. If the data is arranged in a structured format then it becomes easier to find the right information. The preprocessing of the text data is an essential step as it makes the raw text ready for mining, i.e., it becomes easier to extract information from the text and apply machine learning algorithms to it. If we skip this step then there is a higher chance that you are working with noisy and inconsistent data. The objective of this step is to clean noise those are less relevant to find the sentiment of tweets such as punctuation, special characters, numbers, and terms which don't carry much weightage in context to the text.

In one of the later stages, we will be extracting numeric features from our Twitter text data. This feature space is created using all the unique words present in the entire data. So, if we preprocess our data well, then we would be able to get a better quality feature space.

## Procedure & Code:-

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

In [6]:
```python
df = pd.read_csv('data_science.csv')
df.head()
```

```
C:\Users\Acer\anaconda3\lib\site-packages\IPython\core\interactiveshell.py:3165: DtypeWarning: Columns (9) have mixed types.Spe
cify dtype option on import or set low_memory=False.
  has_raised = await self.run_ast_nodes(code_ast.body, cell_name,
```

Out[6]:

| | id | conversation_id | created_at | date | time | timezone | user_id | username | name | place | ... | geo | source | us |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 1406400408545804288 | 1406400396264943616 | 2021-06-20 05:26:01 IST | 2021-06-20 | 05:26:01 | 530 | 1113747629282930688 | ballouxfrancois | Prof Francois Balloux | NaN | ... | NaN | NaN | |
| 1 | 1406390341176016897 | 1406390341176016897 | 2021-06-20 04:46:01 IST | 2021-06-20 | 04:46:01 | 530 | 788898706586275840 | tdatascience | Towards Data Science | NaN | ... | NaN | NaN | |
| 2 | 1406386311481774083 | 1406386311481774083 | 2021-06-20 04:30:00 IST | 2021-06-20 | 04:30:00 | 530 | 19402238 | sciencenews | Science News | NaN | ... | NaN | NaN | |

In [7]:
```python
df.shape
```

Out[7]: (241386, 36)

In [8]:
```python
df.isnull().sum()
```

Out[8]:
```
id                    0
conversation_id       0
created_at            0
date                  0
time                  0
timezone              0
user_id               0
username              0
name                  0
place            241032
tweet                 0
language              0
mentions              0
urls                  0
photos                0
replies_count         0
retweets_count        0
likes_count           0
hashtags              0
cashtags              0
link                  0
retweet               0
```

In [9]:
```
#data cleaning
```

In [10]:
```
columns_to_drop = list(df.columns[27:])
df.drop(columns = columns_to_drop, axis = 1, inplace = True)
```

In [11]:
```
df.isnull().sum()
```

Out[11]:
```
id                    0
conversation_id       0
created_at            0
date                  0
time                  0
timezone              0
user_id               0
username              0
name                  0
place            241032
tweet                 0
language              0
mentions              0
urls                  0
photos                0
replies_count         0
retweets_count        0
likes_count           0
hashtags              0
cashtags              0
```

In [15]:
```
#Dropping cashtags column
df.drop(columns = ['cashtags'], axis = 1, inplace= True)
```

In [16]:
```
df.head()
```

Out[16]:

| | id | conversation_id | created_at | date | time | timezone | user_id | username | name | tweet | la |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 1406400408545804288 | 1406400396264943616 | 2021-06-20 05:26:01 IST | 2021-06-20 | 05:26:01 | 530 | 1113747629282930688 | ballouxfrancois | Prof Francois Balloux | What can be done? - Never blindly trust an ab... | |
| 1 | 1406390341176016897 | 1406390341176016897 | 2021-06-20 04:46:01 IST | 2021-06-20 | 04:46:01 | 530 | 788898706586275840 | tdatascience | Towards Data Science | "We need a paradigm shift from model-centric t... | |
| 2 | 1406386311481774083 | 1406386311481774083 | 2021-06-20 04:30:00 IST | 2021-06-20 | 04:30:00 | 530 | 19402238 | sciencenews | Science News | Using high-resolution satellite data and compu... | |
| 3 | 1406383545153638402 | 1406383545153638402 | 2021-06-20 04:19:01 IST | 2021-06-20 | 04:19:01 | 530 | 788898706586275840 | tdatascience | Towards Data Science | .@Stephenson_Data shares four steps that will ... | |
| | | | 2021-06- | | | | | | Towards | "Curricula is | |

```
In [18]: #Removing IST from created_at column
         df['created_at'] = df['created_at'].apply(lambda tweet: tweet.replace('IST', '').strip())

In [19]: df['created_at'] = pd.to_datetime(df['created_at'])
         df['date'] = pd.to_datetime(df['date'])
         df['time'] = pd.to_datetime(df['time'])

In [20]: df.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 241386 entries, 0 to 241385
Data columns (total 21 columns):
 #   Column           Non-Null Count   Dtype
---  ------           --------------   -----
 0   id               241386 non-null  int64
 1   conversation_id  241386 non-null  int64
 2   created_at       241386 non-null  datetime64[ns]
 3   date             241386 non-null  datetime64[ns]
 4   time             241386 non-null  datetime64[ns]
 5   timezone         241386 non-null  int64
 6   user_id          241386 non-null  int64
 7   username         241386 non-null  object
```

```
In [26]: #Converting to lowercase
         def to_lowercase(text):
             return text.lower()

In [27]: for i in range(4):
             parts[i]['tweet'] = parts[i]['tweet'].apply(lambda tweet: to_lowercase(tweet))

In [28]: for i in range(4):
             print(parts[i]['tweet'].head())

0    what can be done?  - never blindly trust an ab...
1    "we need a paradigm shift from model-centric t...
2    using high-resolution satellite data and compu...
3    .@stephenson_data shares four steps that will ...
4    "curricula is inherently brittle in a world wh...
Name: tweet, dtype: object
60347    the world is digital – learn about the power o...
60348    build your data brilliance in 2020 – explore y...
60349    the end is near, but there is still time to ea...
60350    how to make the transition from analyst to #da...
60351    that time when the australian worked with clim...
Name: tweet, dtype: object
120694    checking out "22 tips for better data science"...
120695    10 questions every data decision-maker should ...
120696    tired of data silos? join our webinar on may 2...
120697    data science is helping people in india with t...
120698    data science master's students visit accenture...
Name: tweet, dtype: object
```

## CONCLUSION:

In this experiment, Students have successfully implemented & classified tweets into positive and negative tweets

# Lab Assignment 15

## Title: Mini Projects/ Case Study

**PROBLEM STATEMENT:**

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

## THEORY:

Python is a great language for doing data analysis, primarily because of the fantastic ecosystem of data-centric python packages. *Pandas* is one of those packages and makes importing and analyzing data much easier.

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

Pandas dataframe.groupby() function is used to split the data into groups based on some criteria. pandas objects can be split on any of their axes. The abstract definition of grouping is to provide a mapping of labels to group names.

Pandas dataframe.sum() function return the sum of the values for the requested axis. If the input is index axis then it adds all the values in a column and repeats the same for all the columns and returns a series containing the sum of all the values in each column. It also provides support to skip the missing values in the dataframe while calculating the sum in the dataframe.

## Procedure & Code:

In [ ]:
```
from google.colab import drive
drive.mount('/content/drive')
```

```
Mounted at /content/drive
```

**Loading the Dataset**

In [ ]:
```
import pandas as pd
path = "/content/drive/MyDrive/Colab Notebooks/experiment/covid_vaccine_statewise.csv"
df = pd.read_csv(path)
```

**(1) Describing the Dataset**

In [ ]:
```
df.describe()
```

Out[3]:

| | Total Doses Administered | Sessions | Sites | First Dose Administered | Second Dose Administered | Male (Doses Administered) | Female (Doses Administered) | Transgender (Doses Administered) | Covaxin (Doses Administered) | CoviShield (Doses Administered) | ... | A |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| count | 7.621000e+03 | 7.621000e+03 | 7621.000000 | 7.621000e+03 | 7.621000e+03 | 7.461000e+03 | 7.461000e+03 | 7461.000000 | 7.621000e+03 | 7.621000e+03 | ... | 1 |
| mean | 9.188171e+06 | 4.792358e+05 | 2282.872064 | 7.414415e+06 | 1.773755e+06 | 3.620156e+06 | 3.168416e+06 | 1162.978019 | 1.044669e+06 | 8.126553e+06 | ... | 8 |
| std | 3.746180e+07 | 1.911511e+06 | 7275.973730 | 2.995209e+07 | 7.570382e+06 | 1.737938e+07 | 1.515310e+07 | 5931.353995 | 4.452259e+06 | 3.298414e+07 | ... | 2 |

In [ ]: `df`

Out[4]:

| | Updated On | State | Total Doses Administered | Sessions | Sites | First Dose Administered | Second Dose Administered | Male (Doses Administered) | Female (Doses Administered) | Transgender (Doses Administered) | ... | 18-44 Years (Doses Administered) | 45-60 ( Adminis |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **0** | 16/01/2021 | India | 48276.0 | 3455.0 | 2957.0 | 48276.0 | 0.0 | NaN | NaN | NaN | ... | NaN | |
| **1** | 17/01/2021 | India | 58604.0 | 8532.0 | 4954.0 | 58604.0 | 0.0 | NaN | NaN | NaN | ... | NaN | |
| **2** | 18/01/2021 | India | 99449.0 | 13611.0 | 6583.0 | 99449.0 | 0.0 | NaN | NaN | NaN | ... | NaN | |
| **3** | 19/01/2021 | India | 195525.0 | 17855.0 | 7951.0 | 195525.0 | 0.0 | NaN | NaN | NaN | ... | NaN | |
| **4** | 20/01/2021 | India | 251280.0 | 25472.0 | 10504.0 | 251280.0 | 0.0 | NaN | NaN | NaN | ... | NaN | |
| **...** | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | |
| **7840** | 11/08/2021 | West Bengal | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | ... | NaN | |
| **7841** | 12/08/2021 | West Bengal | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | ... | NaN | |
| **7842** | 13/08/2021 | West Bengal | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | ... | NaN | |
| **7843** | 14/08/2021 | West Bengal | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | ... | NaN | |

In [ ]: 
```
df.groupby('State')['First Dose Administered'].sum()
```

Out[30]:
```
State
Andaman and Nicobar Islands                 1.642585e+07
Andhra Pradesh                              1.232861e+09
Arunachal Pradesh                           4.900498e+07
Assam                                       5.856002e+08
Bihar                                       1.470503e+09
Chandigarh                                  4.470310e+07
Chhattisgarh                                7.960029e+08
Dadra and Nagar Haveli and Daman and Diu    3.359506e+07
Delhi                                       6.243395e+08
Goa                                         7.599137e+07
Gujarat                                     2.131646e+09
Haryana                                     7.557984e+08
Himachal Pradesh                            3.162940e+08
India                                       2.826214e+10
Jammu and Kashmir                           4.101018e+08
Jharkhand                                   6.036737e+08
Karnataka                                   1.873330e+09
Kerala                                      1.193845e+09
Ladakh                                      1.780925e+07
Lakshadweep                                 4.363655e+06
Madhya Pradesh                              1.796605e+09
Maharashtra                                 2.784364e+09
Manipur                                     6.740957e+07
Meghalaya                                   6.261597e+07
Mizoram                                     4.787308e+07
Nagaland                                    4.241077e+07
```

### (3) No. of persons state-wise vaccinated for second dose in India

In [ ]:
```
df.groupby('State')['Second Dose Administered'].sum()
```

Out[6]:
```
State
Andaman and Nicobar Islands                4.118554e+06
Andhra Pradesh                             3.588176e+08
Arunachal Pradesh                          1.193232e+07
Assam                                      1.307888e+08
Bihar                                      2.707906e+08
Chandigarh                                 1.159374e+07
Chhattisgarh                               1.721204e+08
Dadra and Nagar Haveli and Daman and Diu   4.594416e+06
Delhi                                      1.882189e+08
Goa                                        1.619817e+07
Gujarat                                    6.004184e+08
Haryana                                    1.586561e+08
Himachal Pradesh                           7.383858e+07
India                                      6.759621e+09
Jammu and Kashmir                          8.595165e+07
Jharkhand                                  1.221211e+08
Karnataka                                  4.271872e+08
Kerala                                     3.640488e+08
Ladakh                                     5.453762e+06
Lakshadweep                                1.056446e+06
Madhya Pradesh                             3.169330e+08
```

### (4) No. of males vaccinated

In [ ]:
```
df['Male(Individuals Vaccinated)'].sum()
```

Out[7]: 7138698858.0

### (5) No. of females vaccinated

In [ ]:
```
df['Female(Individuals Vaccinated)'].sum()
```

Out[8]: 6321628736.0

## CONCLUSION:

Performed mini project using pandas sum & groupby methods over covid vaccine dataset.