ASSIGNMENT 1

TITLE: DATA WRANGLING I

PROBLEM STATEMENT: -

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

Import all the required Python Libraries.

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

OBJECTIVE:

1. To Learn and understand the concepts of Python Libraries.
2. To learn and understand the Data Science for the analysis of real time problems
3. To understand and practice Data Preprocessing & Data Normalization.

PREREQUISITE: -

1 Basic of Python Programming

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

THEORY:

1. Introduction to Big Data

Big data means really a big data, it is a collection of large datasets that cannot be processed using traditional computing techniques. Big data is not merely a data, rather it has become a complete subject, which involves various tools, techniques, and frameworks. Big data involves the data produced by different devices and applications. Given below are some of the fields that come under the umbrella of Big Data.

2. Introduction to Dataset

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

3. Python Libraries for Data Science

a. NumPy

One of the most fundamental packages in Python, NumPy is a general-purpose array-processing package. It provides high-performance multidimensional array objects and tools to work with the arrays. NumPy is an efficient container of generic multi-dimensional data. NumPy’s main object is the homogeneous multidimensional array. It is a table of elements or numbers of the same datatype, indexed by a tuple of positive integers. In NumPy, dimensions are called axes, and the number of axes is called rank. NumPy’s array class is called ndarray aka array.

What can you do with NumPy?

1. Basic array operations: add, multiply, slice, flatten, reshape, index arrays

2. Advanced array operations: stack arrays, split into sections, broadcast arrays

3. Work with DateTime or Linear Algebra

4. Basic Slicing and Advanced Indexing in NumPy Python

b. Pandas

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

What can you do with Pandas?

1. Indexing, manipulating, renaming, sorting, merging data frame

2. Update, Add, Delete columns from a data frame

3. Impute missing files; handle missing data or NANs

4. Plot data with histogram or box plot.

c. Scikit Learn

Introduced to the world as a Google Summer of Code project, Scikit Learn is a robust machine learning library for Python. It features ML algorithms like SVMs, random forests, k-means clustering, spectral clustering, mean shift, cross-validation and more... Even NumPy, SciPy and related scientific operations are supported by Scikit Learn with Scikit Learn being a part of the SciPy Stack.

What can you do with Scikit Learn?

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

4. Description of Dataset

The Iris dataset was used in R.A. Fisher's classic 1936 paper, The Use of Multiple Measurements in Taxonomic Problems and can also be found on the UCI Machine Learning Repository. It includes three iris species with 50 samples each as well as some properties about each flower. One flower species is linearly separable from the other two, but the other two are not linearly separable from each other.
Total Sample- 150

The columns in this dataset are:

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

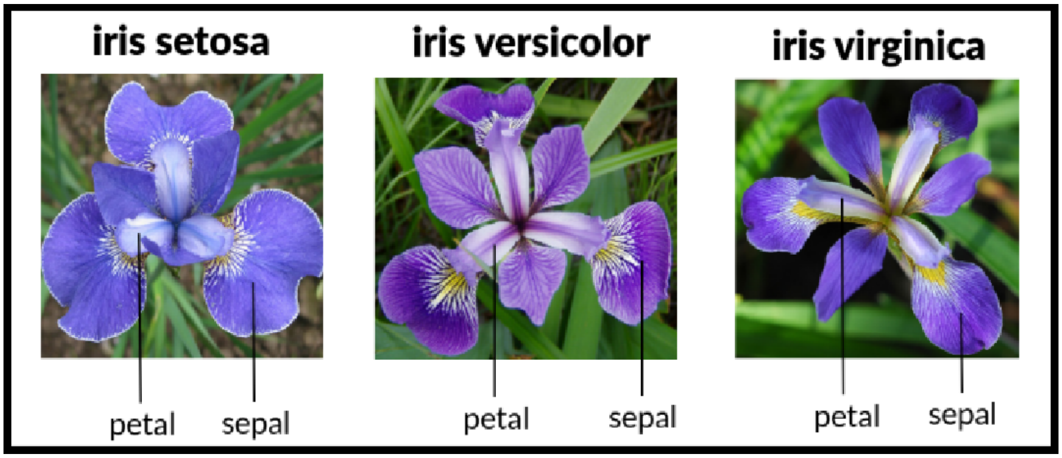


Fig 1 Three Different Types of Species each contain 50 Sample

5. Panda Dataframe functions for Load Dataset

# The columns of the resulting DataFrame have different dtypes. iris.dtypes

1.The dataset is downloads from UCI repository.

csv_url = 'https://archive.ics.uci.edu/ml/machine-learning-databases/iris/iris.data'

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

Load Iris.csv into a Pandas data frame —

Syntax-

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

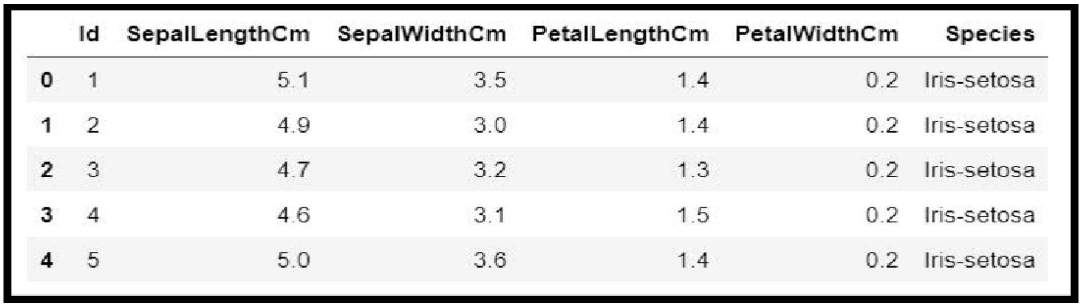
3.The csv file at the UCI repository does not contain the variable/column names. They are located in a separate file.

```
col_names = ['Sepal_Length','Sepal_Width','Petal_Length','Petal_Width','Species']
```

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

```
iris = pd.read_csv(csv_url, names = col_names)
```

Fig.2 Sample Dataset



	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

6. Panda Data frame functions for Data Preprocessing:

Sr. No	Data Frame Function	Description
1	**dataset.head(n=5)**	Return the first n rows.
2	**dataset.tail(n=5)**	Return the last n rows.
3	**dataset.index**	The index (row labels) of the Dataset.
4	**dataset.columns**	The column labels of the Dataset.
5	**dataset.shape**	Return a tuple representing the dimensionality of the Dataset.
6	**dataset.dtypes**	Return the dtypes in the Dataset.
7	**dataset['Column name]**	Read the Data Column wise.
8	**dataset.iloc[5]**	Purely integer-location based indexing for selection by position.
9	**dataset[0:3]**	Selecting via [], which slices the rows.
10	**dataset.loc[:, ["Col_name1", "col_name2"]]**	Selection by label
11	**dataset.iloc[:n, :]**	a subset of the first n rows of the original data
12	**dataset.iloc[:, :n]**	a subset of the first n columns of the original data
13	**dataset.iloc[:m, :n]**	a subset of the first m rows and the first n columns

Table 1. Panda Data frame functions for Data Preprocessing

Checking of Missing Values in Dataset:

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

a. Is there any missing values in data frame as a whole.

Syntax: DataFrame.isnull()

b. Is there any missing values across each column.

Syntax: DataFrame.isnull().any()

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

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

Syntax: dataframe.isnull().sum().sum()

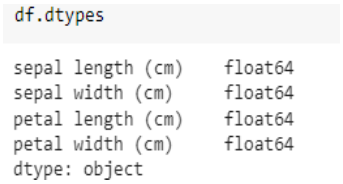
d. count row wise missing value using isnull()

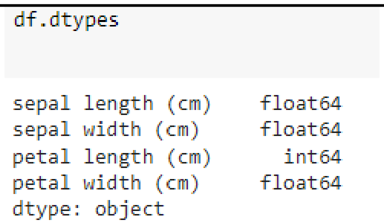
Syntax: dataframe.isnull().sum(axis = 1)

7. Panda functions for Data Formatting and Normalization

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

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

Sr. No	Data Frame Function	Description	Output
1.	**df.dtypes**	To check the data type	 df.dtypes sepal length (cm) float64 sepal width (cm) float64 petal length (cm) float64 petal width (cm) float64 dtype: object
2.	**df['petal length (cm)']= df['petal length**	To change the data type (data type of 'petal length	

		(cm)'changed to int)	 df.dtypes sepal length (cm) float64 sepal width (cm) float64 petal length (cm) int64 petal width (cm) float64 dtype: object
	(cm)'].astype("int ")		

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

Algorithm:

Step 1 : Import pandas and sklearn library for preprocessing

from sklearn import preprocessing

Step 2: Load the iris dataset in dataframe object df

iris = load_iris()

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

Step 3: Print iris dataset.

df.head()

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

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

Step 5: Create a minimum and maximum processor object

min_max_scaler = preprocessing.MinMaxScaler()

Step 6: Create an object to transform the data to fit minmax processor x_scaled = min_max_scaler.fit_transform(x)

Step 7: Run the normalizer on the dataframe

df_normalized = pd.DataFrame(x_scaled)

Step 8: View the dataframe

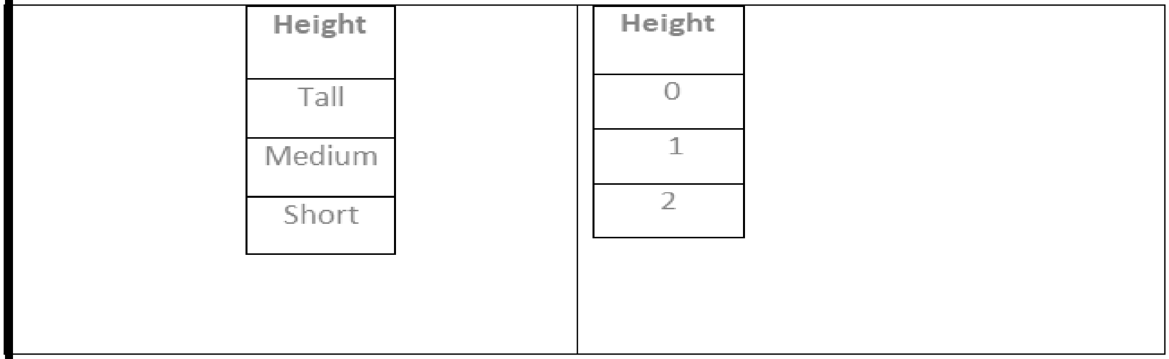
df_normalized

8. Panda Functions for handling categorical variables

- Categorical variables have values that describe a 'quality' or 'characteristic'of a data unit, like 'what type' or 'which category'.
- Categorical variables fall into mutually exclusive (in one category or in another) and exhaustive (include all possible options) categories. Therefore, categorical variables are qualitative variables and tend to be represented by a non-numeric value.
- Categorical features refer to string type data and can be easily understood by human beings. But in case of a machine, it cannot interpret the categorical data directly. Therefore, the categorical data must be translated into numerical data that can be understood by machine.
- There are many ways to convert categorical data into numerical data. Here the most used method is

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

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



Height
Tall
Medium
Short

Height
0
1
2

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

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

Sklearn Functions for Label Encoding:

- **preprocessing.LabelEncoder** : It Encode labels with value between 0 and n_classes-1.
- **fit_transform(y):**

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

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

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

Algorithm:

Step 1 : Import pandas and sklearn library for preprocessing

from sklearn import preprocessing

Step 2: Load the iris dataset in dataframe object df

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

df['Species'].unique()

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

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

label_encoder = preprocessing.LabelEncoder()

Step 5: Encode labels in column 'species'.

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

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

df['Species'].unique()

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

CONCLUSION: In this way we have explored the functions of the python library for Data Preprocessing, Data Wrangling Techniques and How to handle missing values on Iris Dataset.

ASSIGNMENT QUESTION

1. Explain Data Frame with Suitable example.
2. What is the limitation of the label encoding method?
3. What is the need of data normalization?
4. What are the different Techniques for Handling Missing Data?

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.
4. Reason and document your approach properly.

OBJECTIVE:
Students should be able to perform the data wrangling operation using Python on any open-source dataset.

PREREQUISITE: -

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

THEORY:
1. Creation of Dataset using Microsoft Excel.

The dataset is created in "CSV" format.

- The name of dataset is Students Performance
- The features of the dataset are: Math_Score, Reading_Score, Writing_Score, Placement_Score, Club_Join_Date .
- Number of Instances: 30
- The response variable is: Placement_Offer_Count .
- Range of Values:
  Math_Score [60-80], Reading_Score[75-,95], ,Writing_Score [60,80], Placement_Score[75-100], Club_Join_Date [2018-2021].
- The response variable is the number of placement offers facilitated to particular students, which is largely depend on Placement_Score

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

Syntax: RANDBETWEEN (bottom, top)

Bottom The smallest integer and Top The largest integer RANDBETWEEN will return.

2 Identification and Handling of Null Values

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

In Pandas missing data is represented by two values:

1. **None**: None is a Python singleton object that is often used for missing data in Python code.
2. **NaN:** NaN (an acronym for Not a Number), is a special floating-point value recognized by all systems that use the standard IEEE floating-point representation.

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

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

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

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

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

Algorithm:

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

```
import pandas as pd

import numpy as np
```

Step 2: Load the dataset in dataframe object df

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

Step 3: Display the data frame

```
df
```

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

```
df.isnull()
```

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

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

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

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

1. Algorithm:

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

```
import pandas as pd

import numpy as np
```

Step 2: Load the dataset in dataframe object df

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

Step 3: Display the data frame

```
df
```

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

```
df.notnull()
```

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

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

See that there are also categorical values in the dataset, for this, you need to useLabel Encoding

```
from sklearn.preprocessing import LabelEncoderle =
LabelEncoder()
df['gender'] = le.fit_transform(df['gender'])newdf=df
```

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

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

For replacing null values with NaN

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

df
```

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

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

```
import pandas as pd

import numpy as np
```

Step 2: Load the dataset in dataframe object df

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

Step 3: Display the data frame

```
df
```

Step 4: filling missing value using fillna()

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

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

```
data['math score'] = data['math score'].fillna(data['math score'].mean())

data["math score"] = data["math score"].fillna(data["mathscore"].median())

data['math score"] = data["math score"].fillna(data["math score"].std())
```

replacing missing values in forenoon column with minimum/maximum numberof that column

```
data["math score"] = data["math score"].fillna(data["math score"].min())

data["math score"] = data["math score"].fillna(data["math score"].max())
```

- **Filling null values in dataset**

To fill null values in dataset use inplace=true

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

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

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

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

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

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

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

4. Dropping Rows with at least 1 null value in CSV file

Algorithm:

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

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

Step 2: Load the dataset in dataframe object df

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

Step 3: Display the data frame

```
df
```

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

```
ndf.dropna()
```

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

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

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

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

Step 7 : To drop rows with at least 1 null value in CSV file.making new data frame with dropped NA values

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

3. Identification and Handling of Outliers

3.1 Identification of Outliers

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

1. What are Outliers?

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

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

2. Why do they occur?

An outlier may occur due to the variability in the data, or due to Assignment al error/human error. They may indicate an Assignment al error or heavy skewness in the data(heavy-tailed distribution).

3. What do they affect?

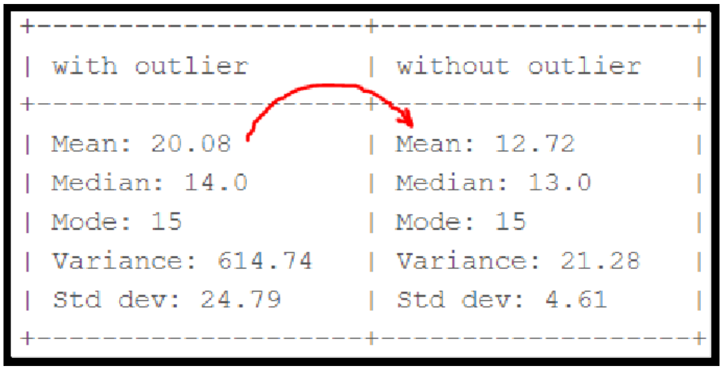
In statistics, we have three measures of central tendency namely Mean, Median, and Mode. They help us describe the data.

Mean is the accurate measure to describe the data when we do not have any outliers present. Median is used if there is an outlier in the dataset. Mode is used if thereis an outlier AND about ½ or more of the data is the same.

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

Example:

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



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

Fig.1 Computation with and without outlier

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

4. Detecting Outliers

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

Below are some of the techniques of detecting outliers

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

Detecting outliers using Boxplot:

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

2. Algorithm:

Step 1 : Import pandas and numpy libraries

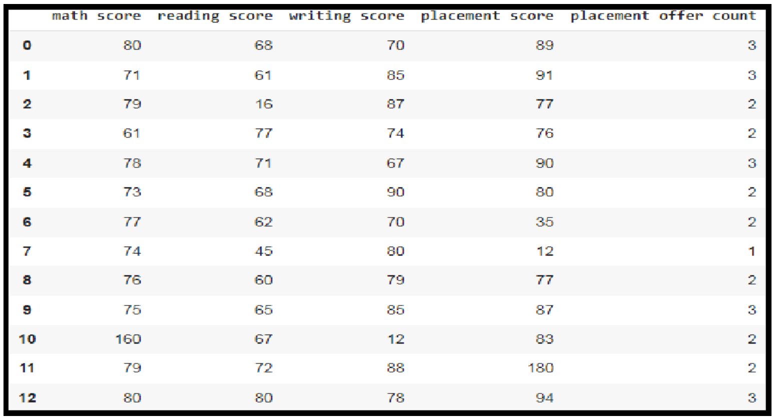
```
import pandas as pd
import numpy as np
```

Step 2: Load the dataset in dataframe object df

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

Step 3: Display the data frame

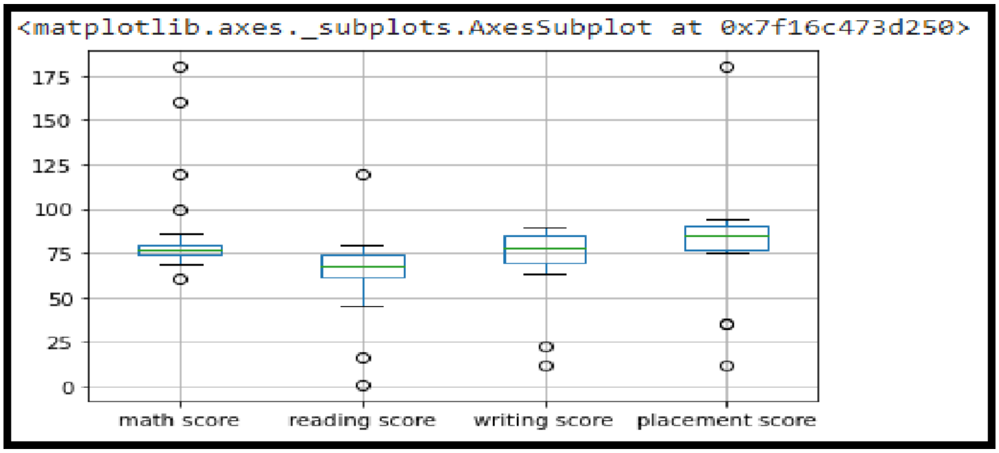
df



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

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

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



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

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

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

Handling of Outliers:

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

Below are some of the methods of treating the outliers

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

4. Data Transformation for the purpose of:

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

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

  Min–max normalization: This transforms the original data linearly.

  Z-score normalization: In z-score normalization (or zero-mean normalization) the values of an attribute (A), are normalized based on the mean of A and its standard deviation.

  Normalization by decimal scaling: It normalizes the values of an attribute by changing the position of their decimal points
- **Attribute or feature construction**.

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

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

a. **To change the scale for better understanding (Attribute or feature construction)**
Here the Club_Join_Date is transferred to Duration.

Algorithm:

Step 1 : Import pandas and numpy libraries

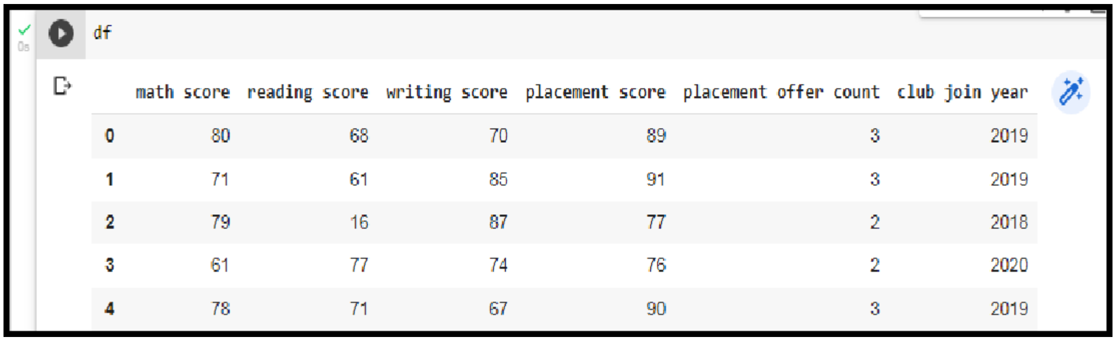
```
import pandas as pd

import numpy as np
```

Step 2: Load the dataset in dataframe object df

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

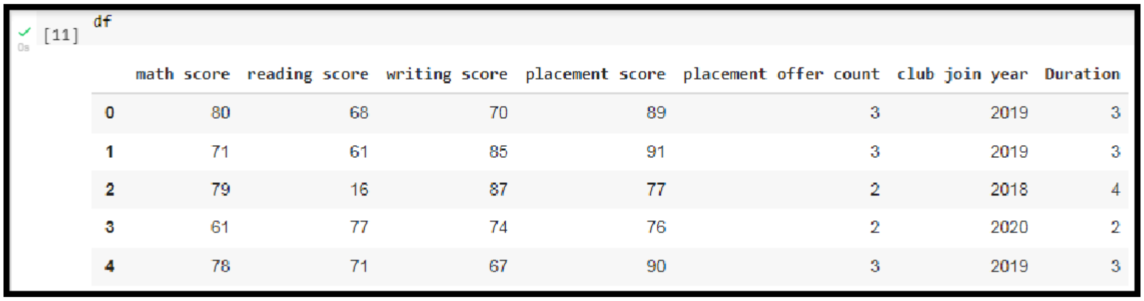
Step 3: Display the data frame

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



```
df
```

	math score	reading score	writing score	placement score	placement offer count	club join year
0	80	68	70	89	3	2019
1	71	61	85	91	3	2019
2	79	16	87	77	2	2018
3	61	77	74	76	2	2020
4	78	71	67	90	3	2019



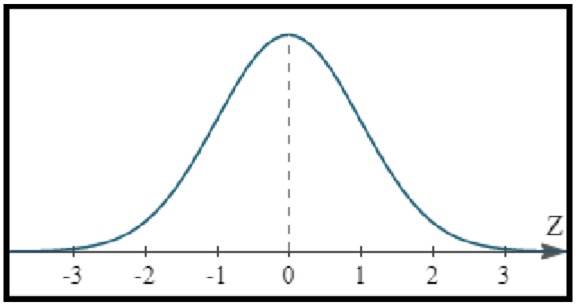
```
[11] df
```

	math score	reading score	writing score	placement score	placement offer count	club join year	Duration
0	80	68	70	89	3	2019	3
1	71	61	85	91	3	2019	3
2	79	16	87	77	2	2018	4
3	61	77	74	76	2	2020	2
4	78	71	67	90	3	2019	3

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

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

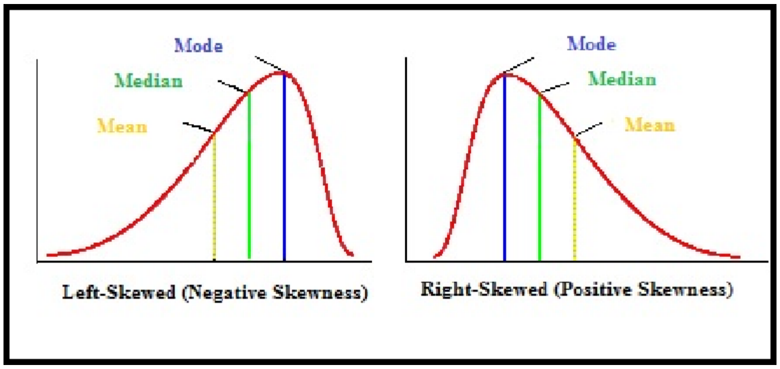
Normal Distribution: In a normal distribution, the graph appears as a classical, symmetrical “bell-shaped curve.” The mean, or average, and the mode, or maximum point on the curve, are equal.



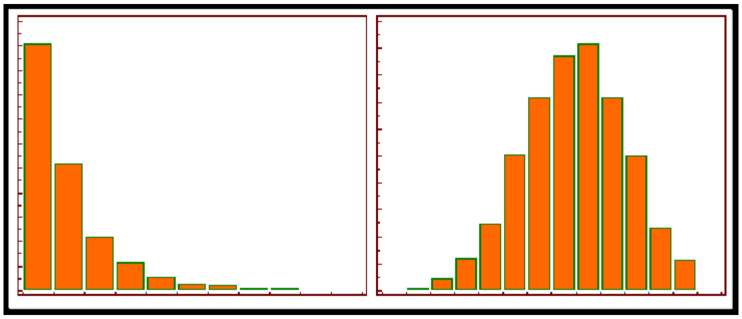
Positively Skewed Distribution

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

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



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



1. **Algorithm:**

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

Step 2: Observe the histogram for math_score variable.

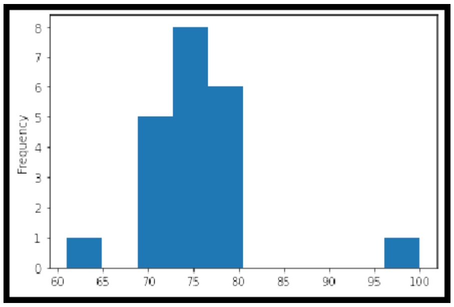
```
import matplotlib.pyplot as plt new_df['math
score'].plot(kind = 'hist')
```

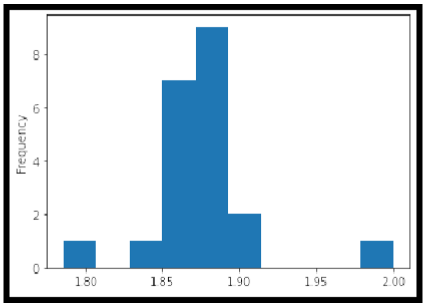
Step 3: Convert the variables to logarithm at the scale 10.

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

Step 4: Observe the histogram for math_score variable.

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





It is observed that skewness is reduced at some level.

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

Assignment Question:

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

.

ASSIGNMENT 3

TITLE: DESCRIPTIVE STATISTICS- MEASURES OF CENTRAL TENDENCY AND VARIABILITY

PROBLEM STATEMENT: -

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

1. Provide summary statistics (mean, median, minimum, maximum, standard deviation) for a dataset (age, income etc.) with numeric variables grouped by one of the qualitative (categorical) variable. For example, if your categorical variable is age groups and quantitative variable is income, then provide summary statistics of income grouped by the age groups. Create a list that contains a numeric value for each response to the categorical variable.
2. Write a Python program to display some basic statistical details like percentile, mean, standard deviation etc. of the species of 'Iris-setosa', 'Iris-versicolor' and 'Iris- versicolor' of iris.csv dataset.

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

OBJECTIVE: To analyze and demonstrate knowledge of statistical data analysis techniques for decision-making

PREREQUISITE: -

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

THEORY:

The data are summarized in some, but not all ways. We chose descriptives that are either most often reported, or most often covered in introductory courses. These are as follows:

- **Central Tendency**
- ○ Mean
- ○ Median
- ○ Mode
- **Dispersion**
- ○ Standard deviation (Std. deviation)
- ○ Minimum
- ○ Maximum

- **Central Tendency**

The Mean, Median and Mode are the three measures of central tendency. Mean is the arithmetic average of a data set. This is found by adding the numbers in a data set and dividing by the number of observations in the data set. The median is the middle number in a data set when the numbers are listed in either ascending or descending order. The mode is the value that occurs the most often in a data set and the range is the difference between the highest and lowest values in a data set.

The Mean

$$\overline{x} = \frac{\sum x}{N}$$

Here,
∑ represents the summation
X represents observations
N represents the number of observations .

In the case where the data is presented in a tabular form, the following formula is used to compute the mean

Mean = ∑f x / ∑f

Where ∑f = N

The Median

If the total number of observations (n) is an odd number, then the formula is given below:

$$Median = \left(\frac{n+1}{2}\right)^{th} observation$$

If the total number of the observations (n) is an even number, then the formula is given below:

$$Median = \frac{\left(\frac{n}{2}\right)^{th} observation + \left(\frac{n}{2}+1\right)^{th} observation}{2}$$

Consider the case where the data is continuous and presented in the form of a frequency distribution, the median formula is as follows.

Find the median class, the total count of observations ∑f.

The median class consists of the class in which (n / 2) is present.

$$\text{Median} = 1 + \left[\frac{\frac{n}{2} - c}{f}\right] \times h$$

Here

l = lesser limit belonging to the median class

c = cumulative frequency value of the class before the median class

f = frequency possessed by the median class

h = size of the class

The Mode

The mode is the most frequently occurring observation or value.

Consider the case where the data is continuous, and the value of mode can be computed using the following steps.

a] Determine the modal class that is the class possessing the maximum frequency.

b] Calculate the mode using the below formula.

$$\text{Mode} = 1 + \left[\frac{f_m - f_1}{2f_m - f_1 - f_2}\right] \times h$$

l = lesser limit of modal class

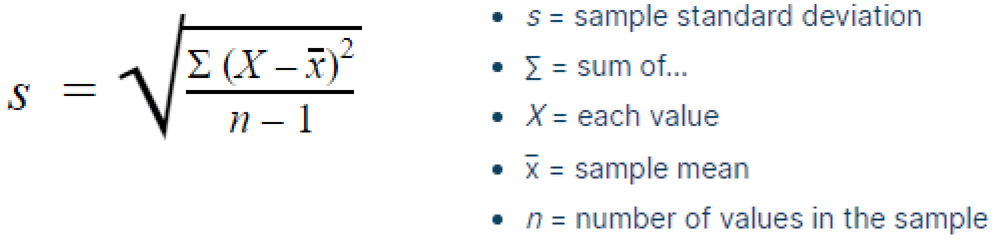
fm = frequency possessed by the modal class

f1 = frequency possessed by the class before the modal class

f2 = frequency possessed by the class after the modal class

h = width of the class

Standard deviation



$$s = \sqrt{\frac{\Sigma\,(X - \bar{x})^2}{n - 1}}$$

- s = sample standard deviation
- $\sum$ = sum of...
- X = each value
- $\bar{x}$ = sample mean
- n = number of values in the sample

Steps for calculating the standard deviation.

The standard deviation is usually calculated automatically by whichever software you use for your statistical analysis. But you can also calculate it by hand to better understand how the formula works.

There are six main steps for finding the standard deviation by hand. We'll use a small data set of 6 scores to walk through the steps.

Data set					
46	69	32	60	52	41

3. Step 1: Find the mean.

To find the mean, add up all the scores, then divide them by the number of scores.

Mean ($\overline{x}$)
$\overline{x}$ = (46 + 69 + 32 + 60 + 52 + 41) ÷ 6 = **50**

4. Step 2: Find each score's deviation from the mean

Subtract the mean from each score to get the deviations from the mean.

Since $\overline{x}$ = 50, here we take away 50 from each score.

Score	**Deviation from the mean**
46	46 – 50 = **-4**
69	69 – 50 = **19**
32	32 – 50 = **-18**
60	60 – 50 = **10**
52	52 – 50 = **2**
41	41 – 50 = **-9**

5. Step 3: Square each deviation from the mean.

Multiply each deviation from the mean by itself. This will result in positive numbers.

Squared deviations from the mean
$(-4)^2$ = 4 × 4 = **16**
19^2 = 19 × 19 = **361**
$(-18)^2$ = -18 × -18 = **324**
10^2 = 10 × 10 = **100**

$2^2 = 2 \times 2 =$ **4**
$(-9)^2 = -9 \times -9 =$ **81**

6. Step 4: Find the sum of squares.

Add up all of the squared deviations. This is called the sum of squares.

Sum of squares
16 + 361 + 324 + 100 + 4 + 81 = **886**

7. Step 5: Find the variance.

Divide the sum of the squares by *n* – 1 (for a sample) or *N* (for a population) – this is the variance.

Since we're working with a sample size of 6, we will use *n* – 1, where *n* = 6.

Variance
886 ÷ (6 – 1) = 886 ÷ 5 = **177.2**

8. Step 6: Find the square root of the variance.

To find the standard deviation, we take the square root of the variance.

Standard deviation
√177.2 = **13.31**

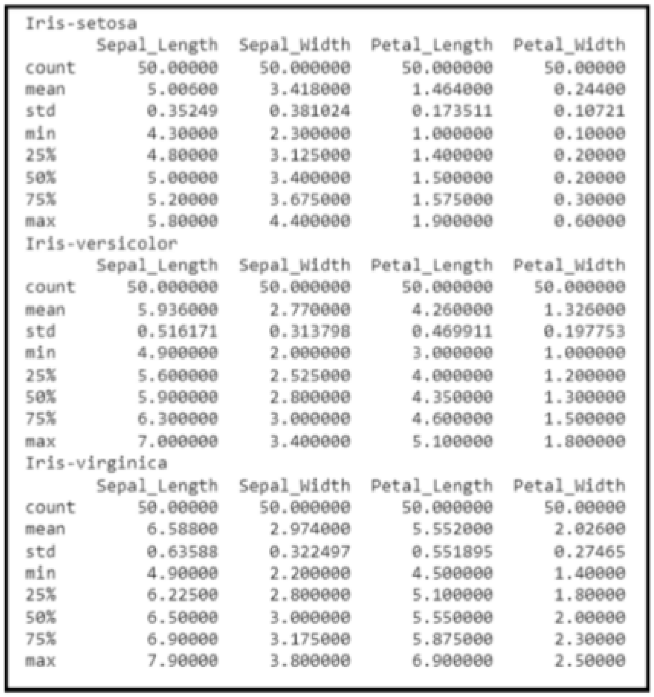
From learning that *SD* = 13.31, we can say that each score deviates from the mean by 13.31 points on average.

Consider HR dataset it contains fields like Age, Monthly Income, Attrition, Business Travel and so on so following are steps to find statistics (mean, median, minimum, maximum, standard deviation)

```
#Mean of monthly income and age
print("The mean of monthly income is :",df.loc[:,"MonthlyIncome"].mean())
print("The mean of age is :",df.loc[:,"Age"].mean())

#Mode of monthly income and age
print("The median of monthly income is :",df.loc[:,"MonthlyIncome"].median())
print("The median of age is :",df.loc[:,"Age"].median())

#Median of monthly income and age
print("The mode of monthly income is :",df.loc[:,"MonthlyIncome"].mode())
print("The mode of age is :",df.loc[:,"Age"].mode())

#Standard deviation of monthly income and age
print("The standard deviation of monthly income is :",df.loc[:,"MonthlyIncome"].std())
print("The standard deviation of age is :",df.loc[:,"Age"].std())

#Storing age and monthly income in array and then finding maximum and minimum values
array1 = np.array(df['MonthlyIncome'])
array2=np.array(df["Age"])
print("Income",array1)
print("Age array",array2)
print("Maximum income among the employees is :",max(array1))
print("Minimum income among the employees is :",min(array1))
print("Maximum age among the employees is :",max(array2))
print("Minimum age among the employees is :",min(array2))

# Replacing the categorical values by numeric values
df.head()
df["BusinessTravel"].replace({"Travel_Rarely":1, "Travel_Frequently":0}, inplace=True)
df["Attrition"].replace({ "Yes":1, "No":0}, inplace=True)
df.head()
```

##Use of describe()

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



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

Conclusion: In this way we have explored the functions of the python library for calculating Data statistics (mean, median, minimum, maximum, standard deviation).

<u>Assignment Question:</u>

1. How to find Mean Mode and Median.
2. How to find standard deviation.
3. Write a Python program to display some basic statistical details like percentile, mean, standard deviation etc.
4. What is use of Describe () Command?

Assignment No: 4

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

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

Prerequisite:

1. Basic of Python Programming

2.Concept of Regression.

Contents for Theory:

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

1. **Linear Regression:** It is a machine learning algorithm based on supervised learning. It targets prediction values on the basis of independent variables.
   - It is preferred to find out the relationship between forecasting and variables.
   - A linear relationship between a dependent variable (X) is continuous; while independent variable(Y) relationship may be continuous or discrete. A linear relationship should be available in between predictor and target variable so knownas Linear Regression.
   - Linear regression is popular because the cost function is Mean Squared Error (MSE)

which is equal to the average squared difference between an observation's actual and predicted values.

- It is shown as an equation of line like :Y =

  m*X + b + e

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

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

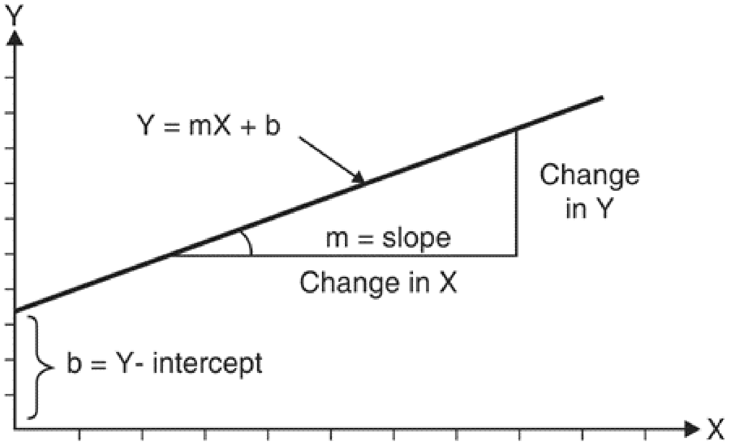
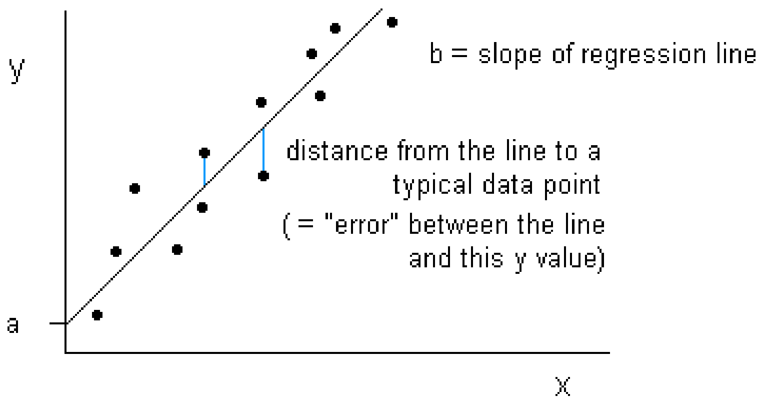


Fig. 1: geometry of linear regression

- Fig. 2 shown below is about the relation between weight (in Kg) and height (in cm), a linear relation. It is an approach of studying in a statistical manner to summarise and learn the relationships among continuous (quantitative) variables.
- Here a variable, denoted by 'x' is considered as the predictor, explanatory, or independent variable.
- Another variable, denoted 'y', is considered as the response, outcome, or dependent variable. While "predictor" and "response" used to refer to these variables.
- Simple linear regression technique concerned with the study of only one predictor variable.

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


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

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

- A simple linear model Y = a + bX is in original feature will be transformed into polynomial feature is transformed and further a linear regression applied to it and it will be something like

  Y=a + bX + cX2

- If a high degree value is used in transformation the curve becomes over-fitted as it captures the noise from data as well.

2. *Least Square Method for Linear Regression*

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

(1)

9. $\hat{y} = \beta_0 + \beta_1 x$

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

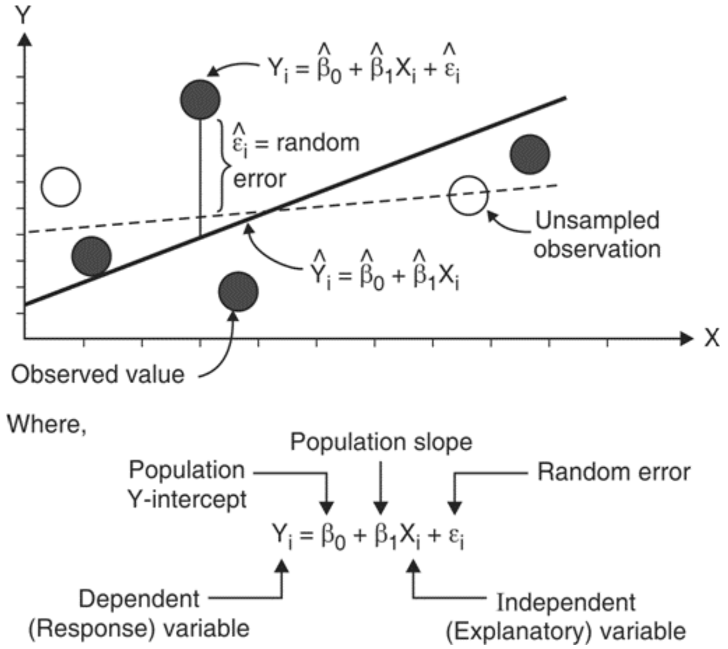
β_0 is the value of y when x=0 or the y intercept

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

- This linear equation represents a line also known as the ‘regression line’. The least square estimation technique is one of the basic techniques used to guess the values of the parameters and based on a sample set.
- This technique estimates parameters β_0 and β_1 and by trying to minimize the square

  of errors at all the points in the sample set. The error is the deviation of the actual sample
- data point from the regression line. The technique can be represented by the equation.

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


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

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

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

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

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

3. Measuring Performance of Linear RegressionMean Square Error:

The Mean squared error (MSE) represents the error of the estimator or predictive model created based on the given set of observations in the sample. Two or more regression models created using a given sample data can be compared based on their MSE. The lesser the MSE, the better the regression model is. When the linear regression model is trained using a given set of observations, the model with the least mean sum of squares error (MSE) is selected as the best model. The Python or R packages select the best-fit model as the model with the lowest MSE or lowest RMSE when training the linear regression models.

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

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

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

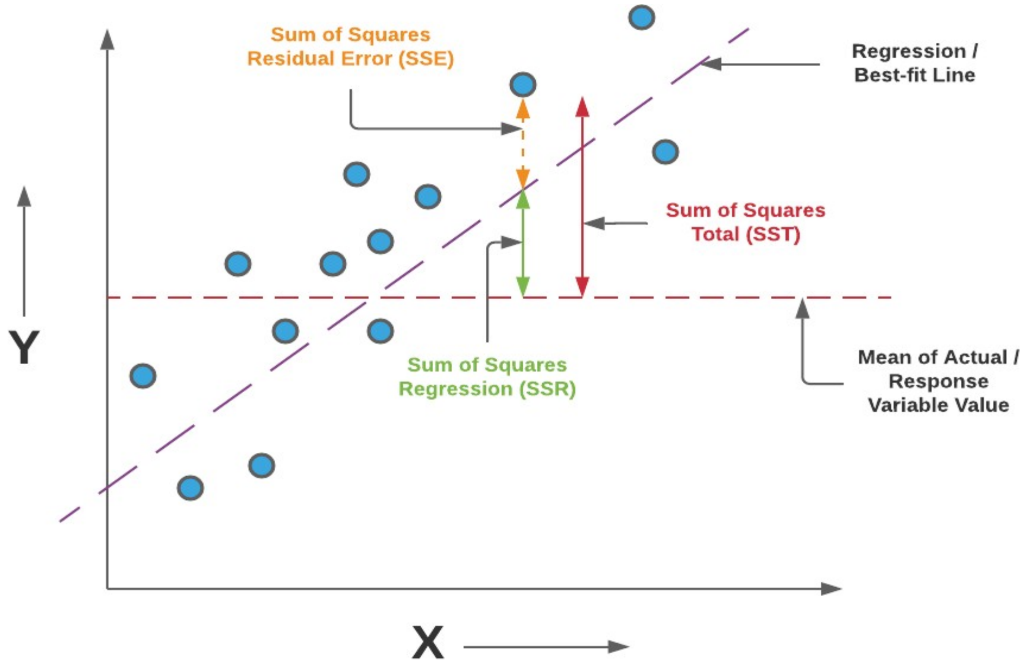
RMSE:

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

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

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

RMSE - Least Squares Regression Method – Edureka R-Squared :



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

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

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

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

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

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

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

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

4. Example of Linear Regression

Consider the following data for 5 students.

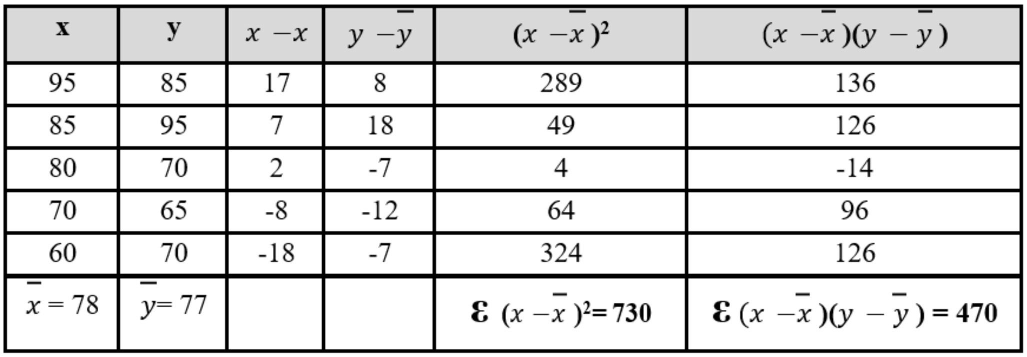
Each Xi (i = 1 to 5) represents the score of ith students in standard X and corresponding Yi (i = 1 to 5) represents the score of ith students in standard XII.

(i) Linear regression equation best predicts standard XIIth score.

(ii) Interpretation for the equation of Linear Regression

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

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



x	**y**	$x - x$	$y - \bar{y}$	$(x - \bar{x})^2$	$(x - \bar{x})(y - \bar{y})$
95	85	17	8	289	136
85	95	7	18	49	126
80	70	2	-7	4	-14
70	65	-8	-12	64	96
60	70	-18	-7	324	126
$\bar{x} = 78$	$\bar{y} = 77$			$\boldsymbol{\varepsilon}\ (x - \bar{x})^2 = \mathbf{730}$	$\boldsymbol{\varepsilon}\ (x - \bar{x})(y - \bar{y}) = \mathbf{470}$

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

(ii) Interpretation of the regression line.

Interpretation 1

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

Interpretation 2

Even if x = 0 value of independent variable, it is expected that value of y is 26.768 Score in XII standard (Yi) is 0.644 units depending on Score in X standard (Xi) but other factors will also contribute to the result of XII standard by 26.768 .

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

For x = 80 the y value will be

5. Training data set and Testing data set

- Machine Learning algorithm has two phases
  1. Training and 2. Testing.
- The input of the training phase is training data, which is passed to any machine learning algorithm and machine learning model is generated as output of the training phase.
- The input of the testing phase is test data, which is passed to the machine learning model and prediction is done to observe the correctness of mode.

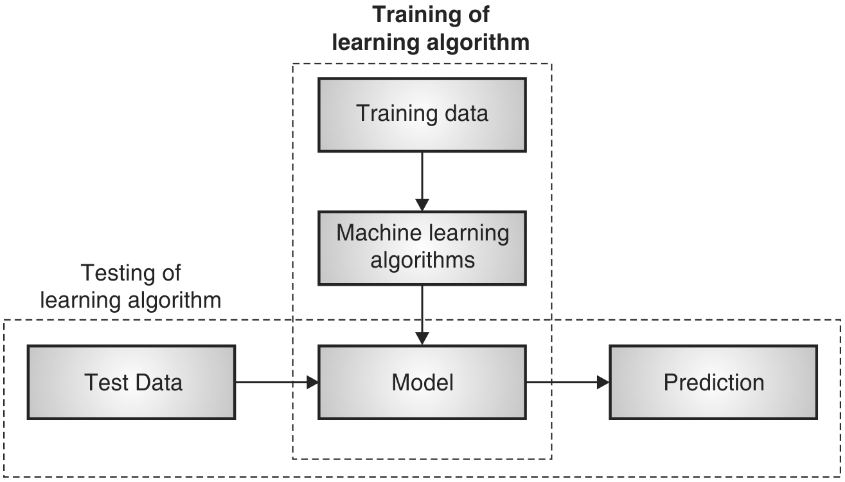

Training of learning algorithm
Training data
Machine learning algorithms
Testing of learning algorithm
Test Data
Model
Prediction

Fig. 1.3.1 : Training and Testing Phase in Machine Learning

(a) Training Phase

- Training dataset is provided as input to this phase.
- Training dataset is a dataset having attributes and class labels and used for training Machine Learning algorithms to prepare models.Machines can learn when they observe enough relevant data. Using this one can model algorithms to find relationships, detect patterns, understand complex problems and makedecisions**.**
- Training error is the error that occurs by applying the model to the same data from which the model is trained**.**
- In a simple way the actual output of training data and predicted output of the model does not match the training error E_{in} is said to have occurred.
- Training error is much easier to compute.

(b) Testing Phase

- Testing dataset is provided as input to this phase.
- Test dataset is a dataset for which class label is unknown. It is tested using model
- A test dataset used for assessment of the finally chosen model.

- Training and Testing dataset are completely different.
- Testing error is the error that occurs by assessing the model by providing the unknown data to the model.
- In a simple way the actual output of testing data and predicted output of the model does not match the testing error E_{out} is said to have occurred.
- E_{out} is generally observed larger than E_{in}.

(c) Generalization

- Generalization is the prediction of the future based on the past system.
- It needs to generalize beyond the training data to some future data that it might not have seen yet.
- The ultimate aim of the machine learning model is to minimize the generalization error.
- The generalization error is essentially the average error for data the model has never seen.
- In general, the dataset is divided into two partition training and test sets.
- The fit method is called on the training set to build the model.
- This fit method is applied to the model on the test set to estimate the target value and evaluate the model's performance.
- The reason the data is divided into training and test sets is to use the test set to estimate how well the model trained on the training data and how well it would perform on the unseen data.

Algorithm (Synthesis Dataset):

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

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

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

```
x=np.array([95,85,80,70,60])
y=np.array([85,95,70,65,70])
```

Step 3 : Create Linear Regression Model using Polyfit Function:

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

Step 4: Observe the coefficients of the model.

```
model
```

Output:

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

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

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

Output:

```
68.63
```

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

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

Output:

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

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

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

```
from sklearn.metrics import r2_score
r2_score(y, y_pred)
```

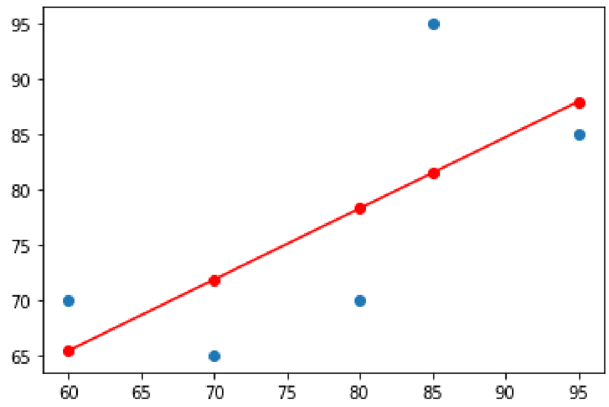
Output:

```
0.4803218090889323
```

Step 8: Plotting the linear regression model

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

Output:



Algorithm (Boston Dataset):

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

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

Step 2: Import the Boston Housing dataset

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

Step 3: Initialize the data frame

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

Step 4: Add the feature names to the dataframe

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

Step 5: Adding target variable to dataframe

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

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

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

Step 7: Split dependent variable and independent variables

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

Step 8: splitting data to training and testing dataset.

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

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

```
import sklearn
from sklearn.linear_model import LinearRegressionlm =
LinearRegression()
model=lm.fit(xtrain, ytrain)
```

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

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

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

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

Step 12: Calculate Mean Square Paper for train_y and test_y

```
from sklearn.metrics import mean_squared_error, r2_scoremse =
mean_squared_error(ytest, ytest_pred)
print(mse)
mse = mean_squared_error(ytrain_pred,ytrain)print(mse)
```

Output:

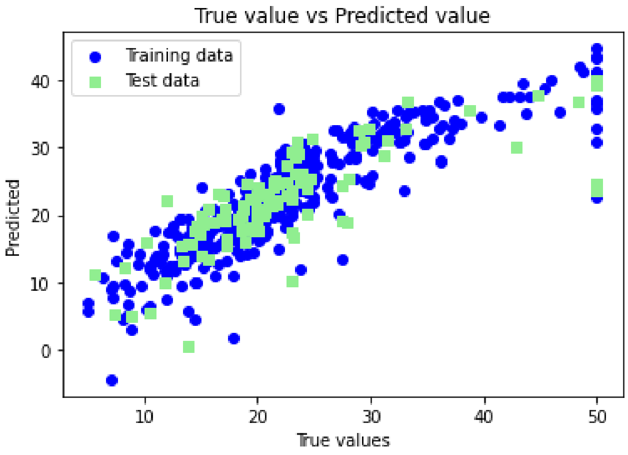
```
33.44897999767638
mse = mean_squared_error(ytest, ytest_pred)print(mse)
```

Output:

```
19.32647020358573
```

Step 13: Plotting the linear regression model

```
lt.scatter(ytrain ,ytrain_pred,c='blue',marker='o',label='Training data') plt.scatter(ytest,ytest_pred
,c='lightgreen',marker='s',label='Test data')plt.xlabel('True values')
plt.ylabel('Predicted')
plt.title("True value vs Predicted value")plt.legend(loc=
'upper left') #plt.hlines(y=0,xmin=0,xmax=50)
plt.plot()
plt.show()
```



Conclusion:

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

Assignment Question:

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

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

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

3) Write python code to calculate the RSquare for Boston Dataset. (Consider the linear regression model created in practical session)

Assignment No: 5

Title:

1. Implement logistic regression using Python/R to perform classification on Social_Network_Ads.csv dataset.
2. Compute Confusion matrix to find TP, FP, TN, FN, Accuracy, Error rate, Precision, Recall on the given dataset.

Objective: Students should be able to data analysis using logisticregression using Python for any open-source dataset

Prerequisite:

1. Basic of Python Programming

2.Concept of Regression.

Contents for Theory:

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

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

$$y = \beta 0 + \beta 1X1 + \beta 2X2 + ... + \beta nXn$$

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

Sigmoid Function:

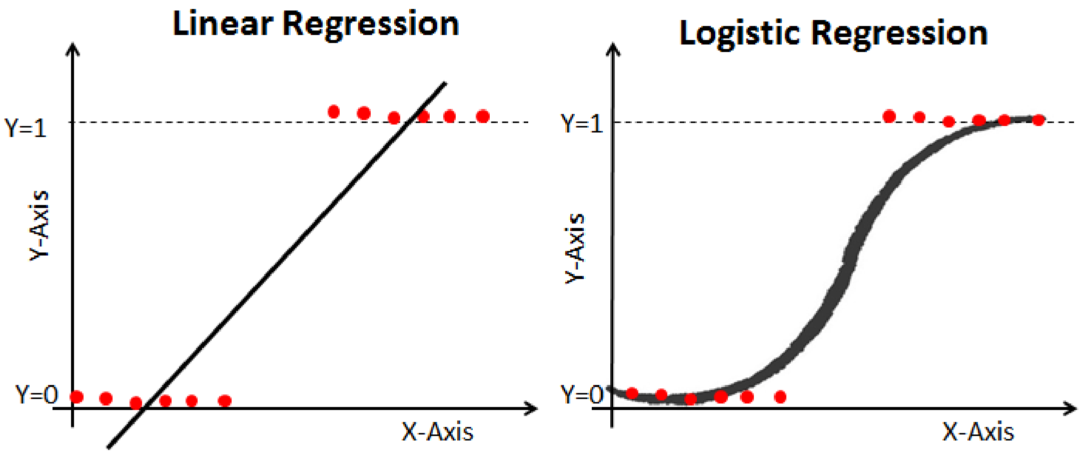
$$p = 1/1 + e^{-y}$$

Apply Sigmoid function on linear regression:

$$p = 1/1 + e^{-(\beta 0+\beta 1X1+\beta 2X2.....\beta nXn)}$$

2. Differentiate between Linear and Logistic Regression

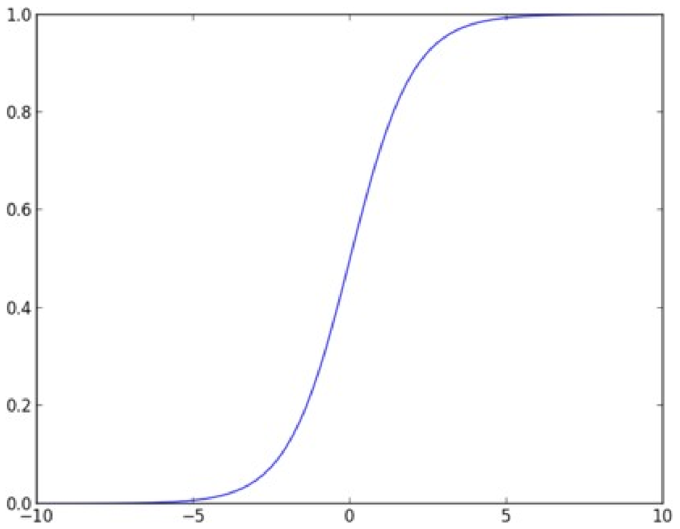
Linear regression gives you a continuous output, but logistic regression provides a constant output. An example of the continuous output is house price and stock price. Example's of the discrete output is predicting whether a patient has cancer or not, predicting whether the customer will churn. Linear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



3. Sigmoid Function

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

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



4. Types of LogisticRegression

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

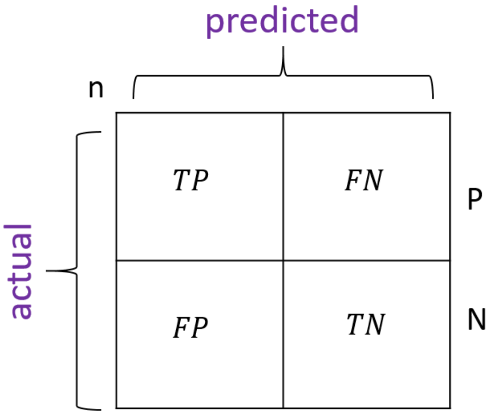
Multinomial Logistic Regression: The target variable has three or more nominal categories such as predicting the type of Wine.

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

5. Confusion Matrix Evaluation Metrics

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

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



Confusion matrix

Here each row indicates the actual classes recorded in the test data set and the each column indicates the classes as predicted by the classifier.
Numbers on the descending diagonal indicate correct predictions, while the ascending diagonal concerns prediction errors.

Some Important measures derived from confusion matrix are:

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

- **Accuracy:** Accuracy is calculated as the number of correctly classified instances divided by totalnumber of instances.

  The ideal value of accuracy is 1, and the worst is 0. It is also calculated as the sum of true positive and true negative (TP + TN) divided by the total number of instances.

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

- **Error Rate:** Error Rate is calculated as the number of incorrectly classified instances divided by total number of instances.

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

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

10. $err = 1 - acc$

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

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

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

  It is calculated as the number of correctly classified positive instances divided by the total number of positive instances.

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

Algorithm (Boston Dataset):

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

Step 2: Import the Social_Media_Adv Dataset

Step 3: Initialize the data frame

Step 4: Perform Data

Preprocessing

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

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

```
# import the class
from sklearn.linear_model import LogisticRegression# instantiate the model
  (using the default parameters)
logreg=LogisticRegression()# fit the model ith data
logreg.fit(xtrain,ytrain)
# y_pred=logreg.predict(xtest)
```

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

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

Step 8: Calculate the required evaluation parameters

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

Conclusion:

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

.

Assignment No: 6

Title :

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

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

Objective:

- Students should be able to data analysis using Naïve Bayes classification algorithm using Python for any open-source dataset.

Prerequisite:

1. Basic of Python Programming

2.Concept of Join and Marginal Probability.

Contents for Theory:

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

1. Concepts used in Naïve Bayes classifier.

- Naïve Bayes Classifier can be used for Classification of categorical data.
  - ○ Let there be a ‘j’ number of classes. C={1,2,….j}
  - ○ Let, input observation is specified by ‘P’ features. Therefore input observation x is given , x = {F1,F2,…..Fp}
  - ○ The Naïve Bayes classifier depends on Bayes' rule from probability theory.
- Prior probabilities: Probabilities which are calculated for some event based on no other information are called Prior probabilities.

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

Conditional Probabilities:

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

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

From equation (1) and (2) ,

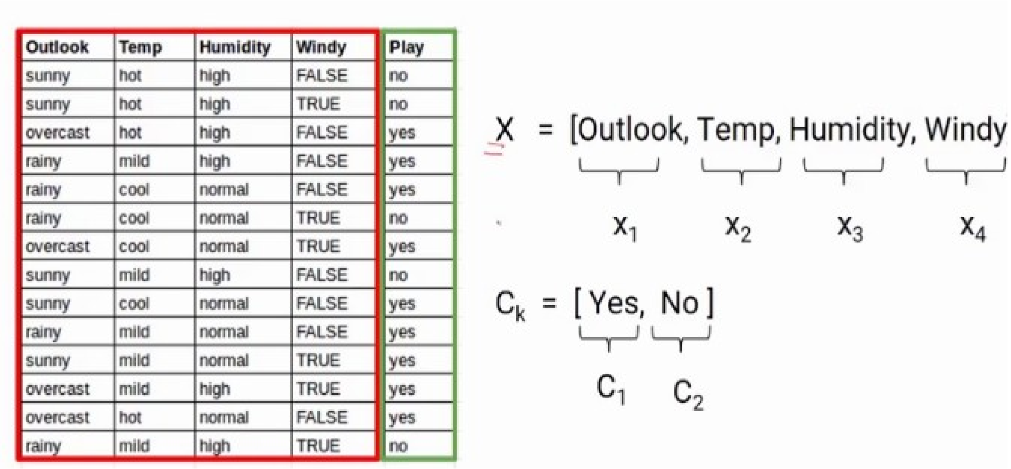
$$P(A\cap B) = P\left(\frac{A}{B}\right).P(B) = P\left(\frac{B}{A}\right).P(A)$$

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

Is called the Bayes Rule.

2. **Example of Naive Bayes**

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



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

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

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

Conditional Probability

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

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

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

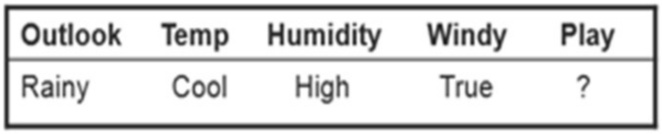
$$P(C_1 \mid x_1 \cap x_2 \cap x_3 \cap x_4) = \frac{P(x_1 \cap x_2 \cap x_3 \cap x_4 \mid C_1) * P(C_1)}{P(x_1 \cap x_2 \cap x_3 \cap x_4)}$$

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

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

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

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



Outlook	Temp	Humidity	Windy	Play
Rainy	Cool	High	True	?

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

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

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

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

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

1) **Algorithm (Iris Dataset)**:

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

Step 2: Import the Iris dataset by calling URL.

Step 3: Initialize the data frame

Step 4: Perform Data Preprocessing

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

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

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

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

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

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

```
accuracy=accuracy_score(y_test,Y_pred)
```

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

5) Step 8: Calculate the required evaluation parameters

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

Conclusion:

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

Assignment No: 7

Title of the Assignment:

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

Objective:

Students should be able to perform **Text Analysis** using TFIDF Algorithm

Prerequisite:

1. Basic of Python Programming
2. Basic of English language.

Contents for Theory:

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

1. Basic concepts of Text Analytics

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

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

2. Text Analysis Operations using natural language toolkit

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

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

- Sentence tokenization : split a paragraph into **list of sentences** using **sent_tokenize()** method
- Word tokenization : split a sentence into **list of words** using **word_tokenize()** method

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 sensecuts either the beginning or end of the word.

On the contrary, Lemmatization is a more powerful operation, and it takes into consideration morphological analysis of the words. It returns the lemma which is the base form of all its inflectional forms. In-depth linguistic knowledge is

required to create dictionaries and look for the proper form of the word. Stemming is a general operation while lemmatization is an intelligent operation where the proper form will be looked in the dictionary. Hence, lemmatization helps in forming better machine learning features.

2.2. 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.

Part-II

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 orcorpus.

- **Term Frequency (TF)**

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

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

Example:

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

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

- **Inverse Document Frequency (IDF)**

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

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

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

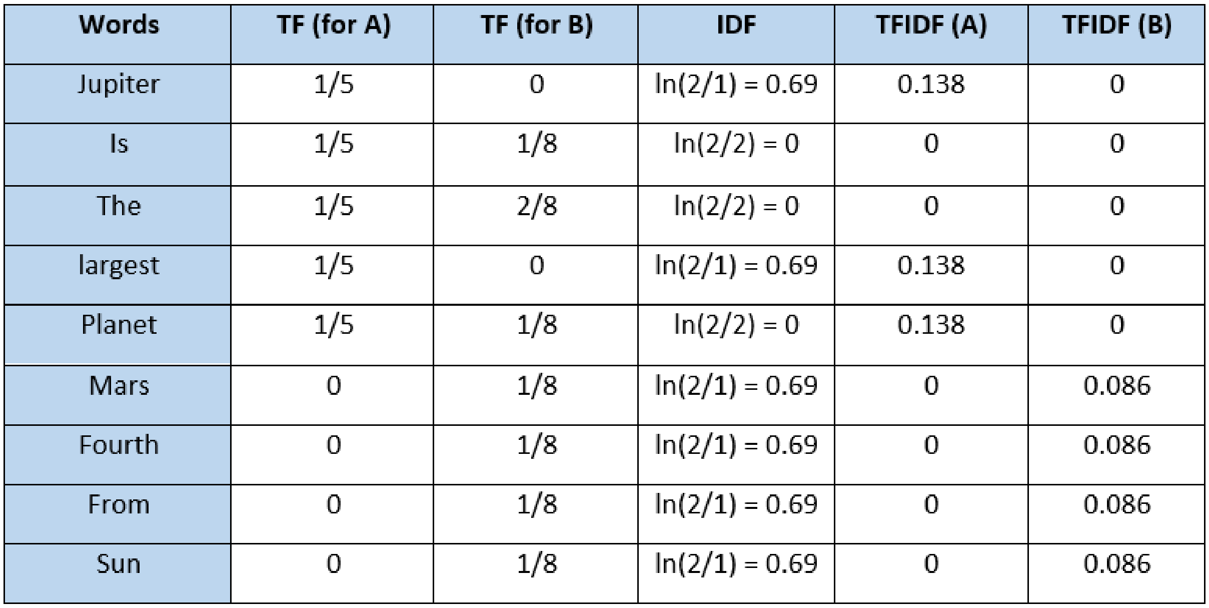
Words	TF (for A)	TF (for B)	IDF
Jupiter	1/5	0	ln(2/1) = 0.69
Is	1/5	1/8	ln(2/2) = 0
The	1/5	2/8	ln(2/2) = 0
largest	1/5	0	ln(2/1) = 0.69
Planet	1/5	1/8	ln(2/2) = 0
Mars	0	1/8	ln(2/1) = 0.69
Fourth	0	1/8	ln(2/1) = 0.69
From	0	1/8	ln(2/1) = 0.69
Sun	0	1/8	ln(2/1) = 0.69

Term Frequency — Inverse Document Frequency (TFIDF)

It is the product of TF and IDF.

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

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



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

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

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

- **Disadvantage of TFIDF**

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

3. Bag of Words (BoW)

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

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

Step 1: Download the required packages

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

Step 2: Initialize the text

text= "Tokenization is the first step in text analytics. The process of breaking down a text paragraph into smaller chunks such as words or sentences is called Tokenization."

Step 3: Perform Tokenization

```
#Sentence Tokenization
from nltk.tokenize import sent_tokenize
tokenized_text= sent_tokenize(text)print(tokenized_text)

#Word Tokenization
from nltk.tokenize import word_tokenize tokenized_word=word_tokenize(text)
print(tokenized_word)
```

Step 4: Removing Punctuations and Stop Word

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

text= "How to remove stop words with NLTK library in Python?"text= re.sub('[^a-zA-Z]', ' ',text)
tokens = word_tokenize(text.lower())filtered_text=[]
for w in tokens:
if w not in stop_words: filtered_text.append(w)
print("Tokenized Sentence:",tokens) print("Filterd Sentence:",filtered_text)
```

Step 5 : Perform Stemming

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

Step 6: Perform Lemmatization

```
from nltk.stem import WordNetLemmatizer
wordnet_lemmatizer = WordNetLemmatizer() text = "studies studying
cries cry" tokenization = nltk.word_tokenize(text)
```

```
for w in tokenization:
print("Lemma for {} is {}".format(w,
wordnet_lemmatizer.lemmatize(w)))
```

Step 7: Apply POS Tagging to text

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

Algorithm for Create representation of document by calculating TFIDF

Step 1: Import the necessary libraries.

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

Step 2: Initialize the Documents.

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

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

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

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

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

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

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

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

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

Step 7: Compute the term Inverse Document Frequency.

```
def computeIDF(documents):
import math
```

```
N = len(documents)

idfDict = dict.fromkeys(documents[0].keys(), 0)for document in documents:
for word, val in document.items():if val > 0:
idfDict[word] += 1

for word, val in idfDict.items(): idfDict[word] = math.log(N / float(val))
return idfDict
idfs = computeIDF([numOfWordsA, numOfWordsB])idfs
```

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

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

Conclusion:

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

Assignment Question:

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

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

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

.

Assignment No: 8

Title of the Assignment: Data Visualizations I

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

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

Objective:

Students should be able to perform the data Visualizationoperation using Python on any open-source dataset.

Prerequisite:

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

Contents for Theory:

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

Theory:

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

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

1. **Seaborn Library Basics**

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

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

```
pip install seaborn
conda install seaborn.
```

To import seaborn you can run the following command.

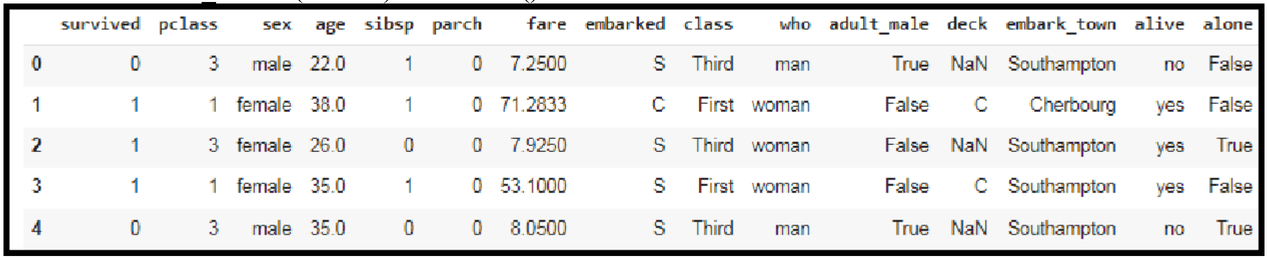
```
import seaborn as sns
```

2. **Know your data**

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

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

```
import pandas as pdimport
numpy as np
import matplotlib.pyplot as pltimport seaborn as
sns


dataset = sns.load_dataset('titanic')dataset.head()
```



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

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

3. Finding patterns of data.

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

Types of plots are:

A. Distribution Plots

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

B. Categorical Plots

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

C. Advanced Plots

a. Strip Plot

b. Swarm Plot

D. Matrix Plots

a. Heat Map
b. Cluster Map

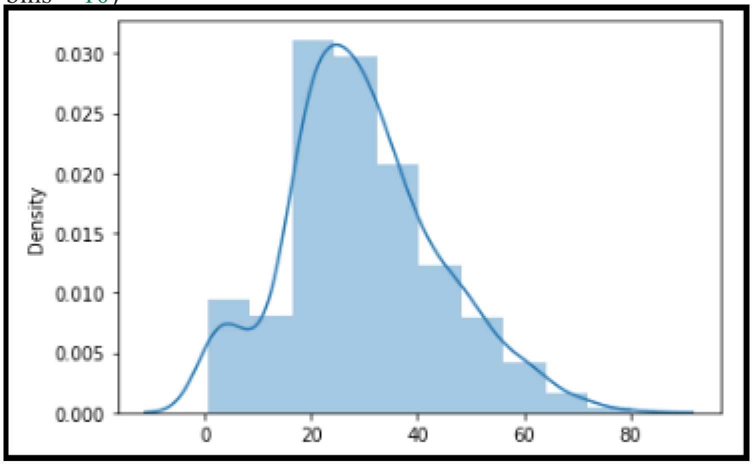
Distribution Plots:

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

Distplot

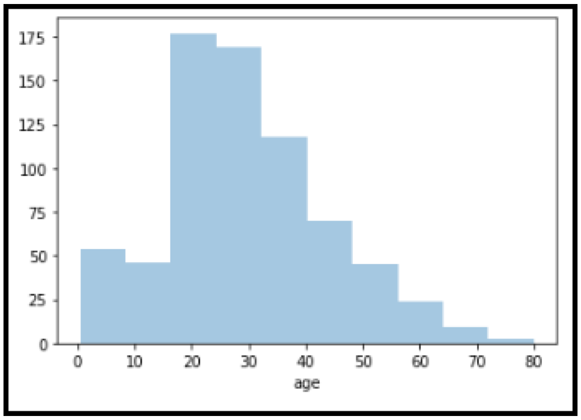
- Dist plot gives us the histogram of the selected continuous variable.
- It is an example of a univariate analysis.
- We can change the number of bins i.e. number of vertical bars in a histogram

```
import seaborn as sns
sns.distplot(x = dataset['age'], bins = 10)
```



The line that you see represents the kernel density estimation. You can remove this lineby passing False as the parameter for the kde attribute as shown below

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

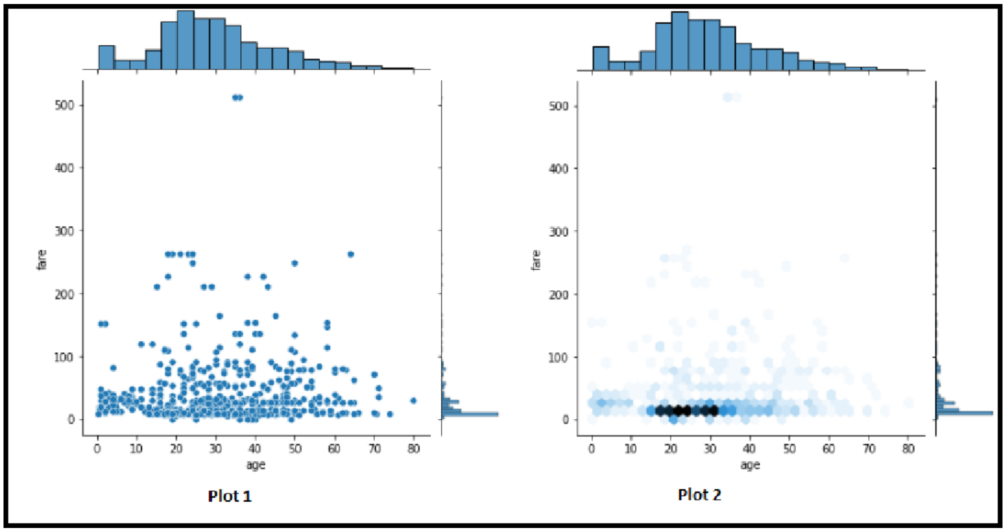


Here the x-axis is the age and the y-axis displays frequency. For example, for bins = 10,there are around 50 people having age 0 to 10

Joint Plot

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

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

# For Plot 2

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



Plot 1

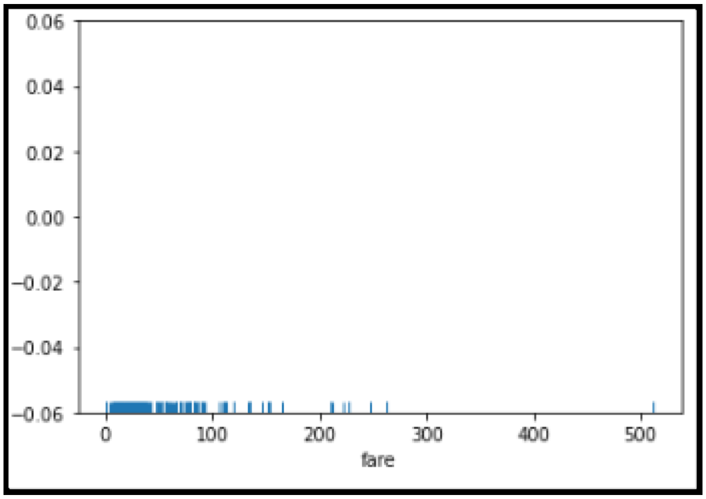
Plot 2

- From the output, you can see that a joint plot has three parts. A distribution plot at the top for the column on the x-axis, a distribution plot on the right for the column on the y-axis and a scatter plot in between that shows the mutual distribution of data for both the columns. You can see that there is no correlation observed between prices and the fares.
- You can change the type of the joint plot by passing a value for the kind parameter. For instance, if instead of a scatter plot, you want to display the distribution of data in the form of a hexagonal plot, you can pass the value hex for the kind parameter.
- In the hexagonal plot, the hexagon with the most number of points gets darker colour. So if you look at the above plot, you can see that most of the passengers are between the ages of 20 and 30 and most of them paid between 10-50 for the tickets.

The Rug Plot

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

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



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

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

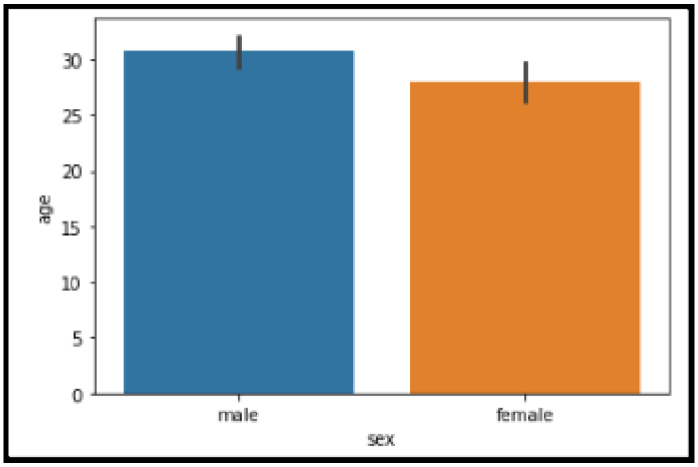
Categorical Plots

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

a. The Bar Plot

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

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

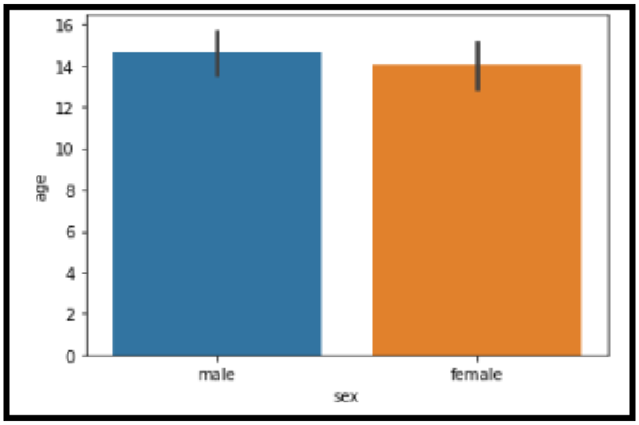


From the output, you can clearly see that the average age of male passengers is just less than 40 while the average age of female passengers is around 33.

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

```
import numpy as np

import matplotlib.pyplot as pltimport seaborn as

sns

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

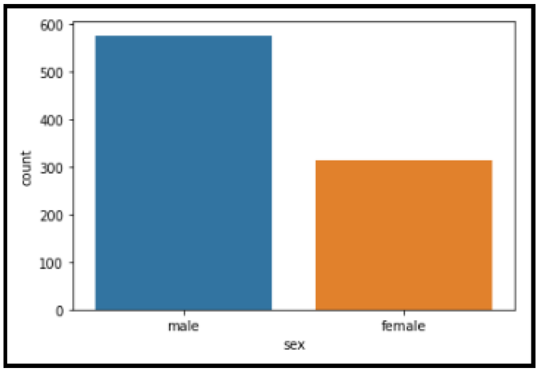
Notice, in the above script we use the std aggregate function from the numpy library to calculate the standard deviation for the ages of male and female passengers. The output looks like this:



b. The Count Plot

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

```
sns.countplot(x='sex', data=dataset)
```

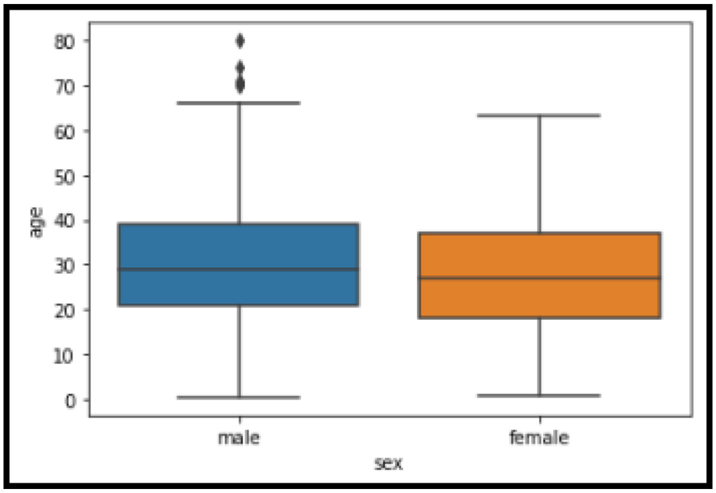


c. The Box Plot

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

Now let's plot a box plot that displays the distribution for the age with respect to each gender. You need to pass the categorical column as the first parameter (which is sex in our case) and the numeric column (age in our case) as the second parameter. Finally, the dataset is passed as the third parameter, take a look at the following script:

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

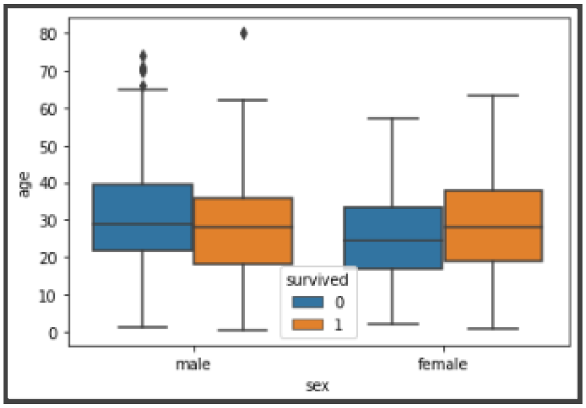


Let's try to understand the box plot for females. The first quartile starts at around 1 and ends at 20 which means that 25% of the passengers are aged between 1 and 20. The second quartile starts at around 20 and ends at around 28 which means that 25% of the passengers are aged between20 and 28. Similarly, the third quartile starts and ends between 28 and 38, hence 25% passengers are aged within this range and finally the fourth or last quartile starts at 38 and ends around 64.

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.

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

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



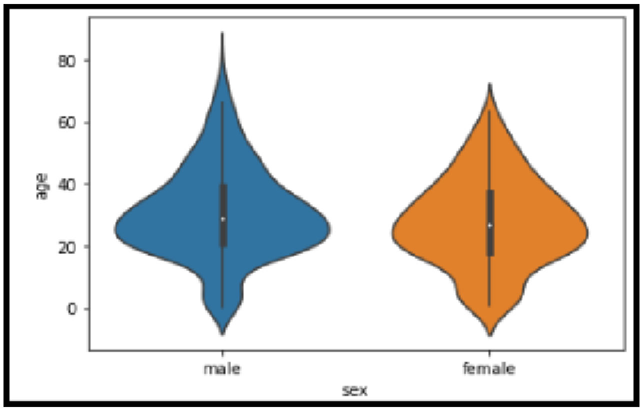
Now in addition to the information about the age of each gender, you can also see the distribution of the passengers who survived. For instance, you can see that among the male passengers, on average more younger people survived as compared to the older ones. Similarly, you can see that the variation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

d. The Violin Plot

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

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

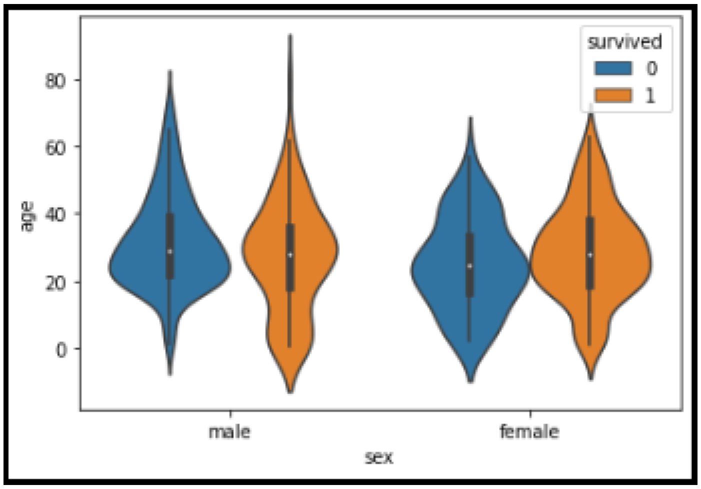
```
sns.violinplot(x='sex', y='age', data=dataset)
```



You can see from the figure above that violin plots provide much more information about the data as compared to the box plot. Instead of plotting the quartile, the violin plot allows us to see all the components that actually correspond to the data. The area where the violin plot is thicker has a higher number of instances for the age. For instance, from the violin plot for males, it is clearly evident that the number of passengers with age between 20 and 40 is higher than all the rest of the age brackets.

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

```
sns.violinplot(x='sex', y='age', data=dataset, hue='survived')
```



Now you can see a lot of information on the violin plot. For instance, if you look at the bottom of the violin plot for the males who survived (left-orange), you can see that it is thicker than the bottom of the violin plot for the males who didn't survive (left-blue). This means that the numberof young male passengers who survived is greater than the number of young male passengers who did not survive.

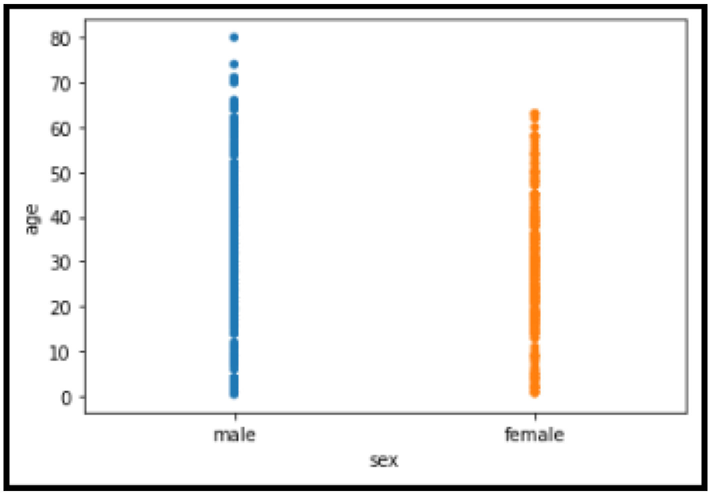
Advanced Plots:

The Strip Plot

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

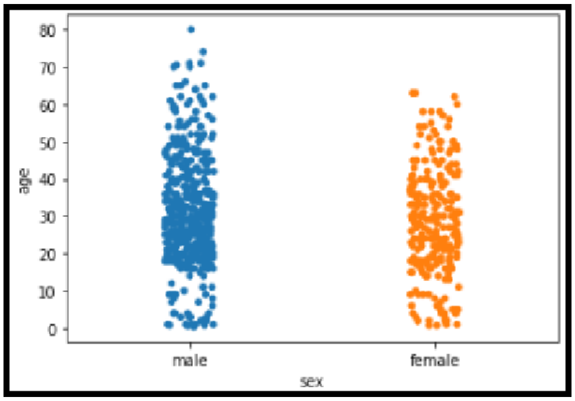
The stripplot() function is used to plot the violin plot. Like the box plot, the first parameter is the categorical column, the second parameter is the numeric column while the third parameter is the dataset. Look at the following script:

```
sns.stripplot(x='sex', y='age', data=dataset, jitter=False)
```



You can see the scattered plots of age for both males and females. The data points look like strips. It is difficult to comprehend the distribution of data in this form. To better comprehend thedata, pass True for the jitter parameter which adds some random noise to the data. Look at the following script:

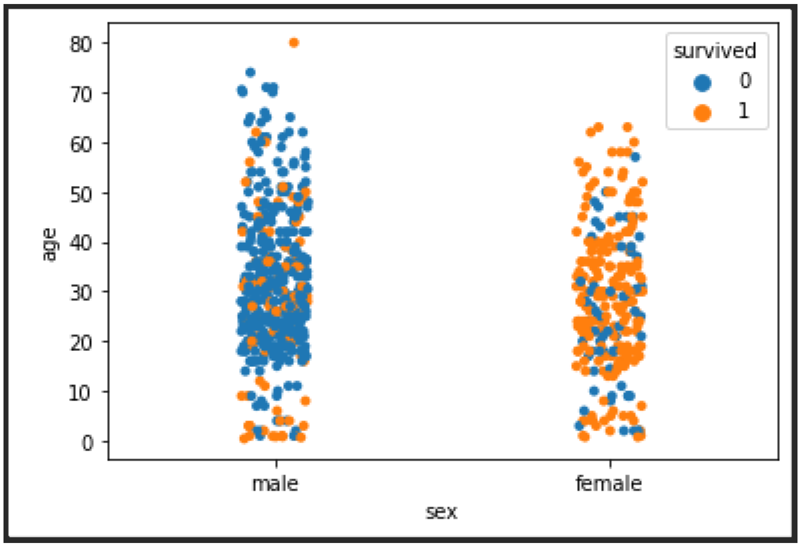
```
sns.stripplot(x='sex', y='age', data=dataset, jitter=True)
```



Now you have a better view for the distribution of age across the genders.

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

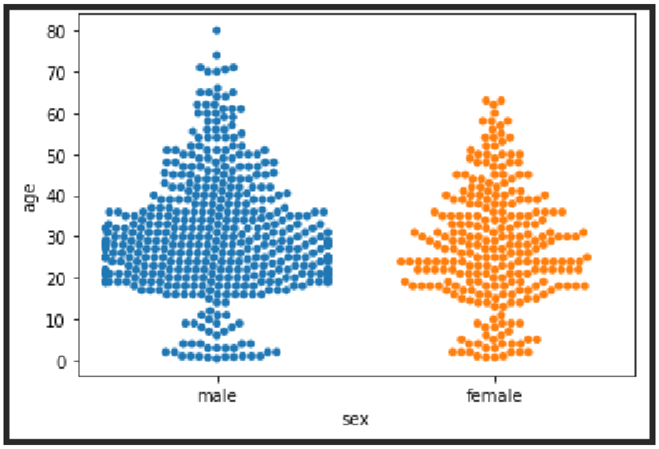
```
sns.stripplot(x='sex', y='age', data=dataset, jitter=True, hue='survived')
```



The Swarm Plot

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

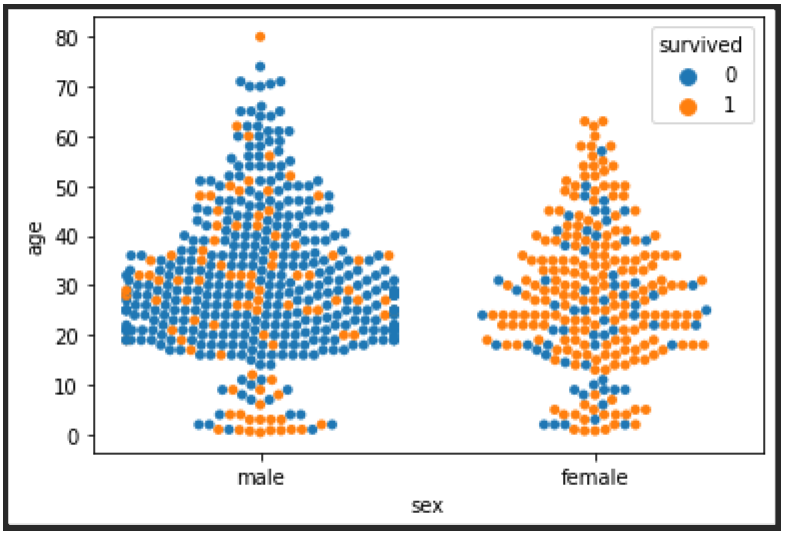
```
sns.swarmplot(x='sex', y='age', data=dataset)
```



You can clearly see that the above plot contains scattered data points like the strip plot and the data points are not overlapping. Rather they are arranged to give a view similar to that of a violinplot.

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

```
sns.swarmplot(x='sex', y='age', data=dataset, hue='survived')
```



From the output, it is evident that the ratio of surviving males is less than the ratio of surviving females. Since for the male plot, there are more blue points and less orange points. On the other hand, for females, there are more orange points (surviving) than the blue points (not surviving). Another observation is that amongst males of age less than 10, more passengers survived as compared to those who didn't.

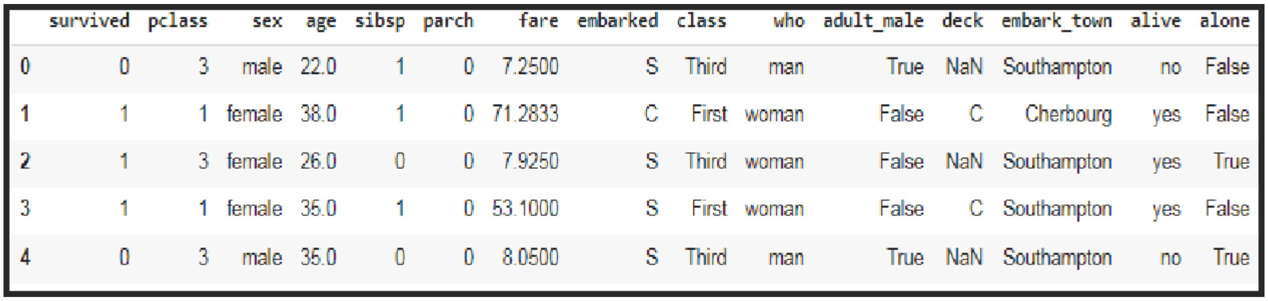
Matrix Plots

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

Heat Maps

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

```
import pandas as pd
import numpy as np

import matplotlib.pyplot as pltimport seaborn as

sns

dataset = sns.load_dataset('titanic')dataset.head()
```

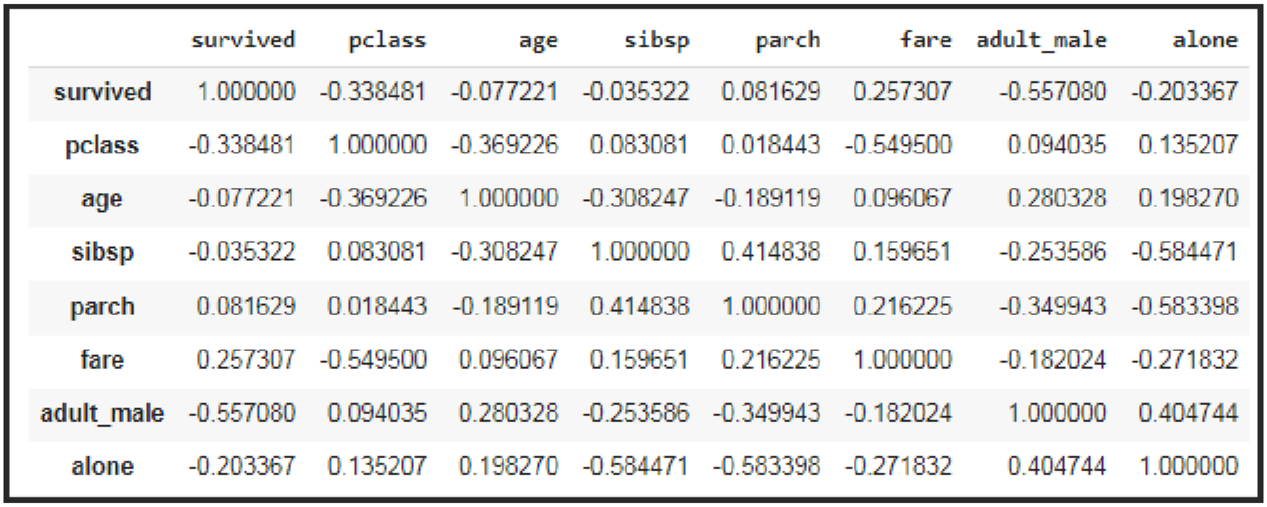


	survived	pclass	sex	age	sibsp	parch	fare	embarked	class	who	adult_male	deck	embark_town	alive	alone
0	0	3	male	22.0	1	0	7.2500	S	Third	man	True	NaN	Southampton	no	False
1	1	1	female	38.0	1	0	71.2833	C	First	woman	False	C	Cherbourg	yes	False
2	1	3	female	26.0	0	0	7.9250	S	Third	woman	False	NaN	Southampton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

From the output, you can see that the column headers contain useful information such aspassengers surviving, their age, fare etc. However the row headers only contain indexes 0, 1, 2, etc. To plot matrix plots, we need useful information on both columns and row headers. One wayto do this is to call the corr() method on the dataset. The corr() function returns the correlation between all the numeric columns of the dataset. Execute the following script:

```
dataset.corr()
```

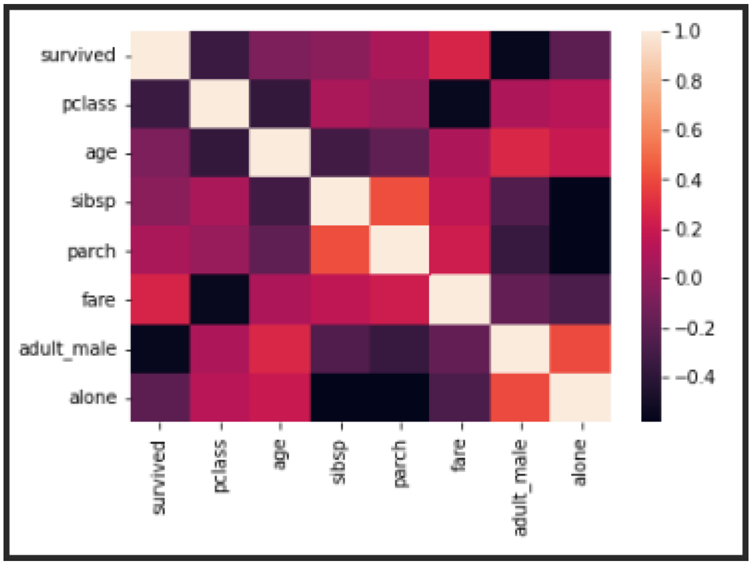
In the output, you will see that both the columns and the rows have meaningful header information, as shown below:



	survived	pclass	age	sibsp	parch	fare	adult_male	alone
survived	1.000000	-0.338481	-0.077221	-0.035322	0.081629	0.257307	-0.557080	-0.203367
pclass	-0.338481	1.000000	-0.369226	0.083081	0.018443	-0.549500	0.094035	0.135207
age	-0.077221	-0.369226	1.000000	-0.308247	-0.189119	0.096067	0.280328	0.198270
sibsp	-0.035322	0.083081	-0.308247	1.000000	0.414838	0.159651	-0.253586	-0.584471
parch	0.081629	0.018443	-0.189119	0.414838	1.000000	0.216225	-0.349943	-0.583398
fare	0.257307	-0.549500	0.096067	0.159651	0.216225	1.000000	-0.182024	-0.271832
adult_male	-0.557080	0.094035	0.280328	-0.253586	-0.349943	-0.182024	1.000000	0.404744
alone	-0.203367	0.135207	0.198270	-0.584471	-0.583398	-0.271832	0.404744	1.000000

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

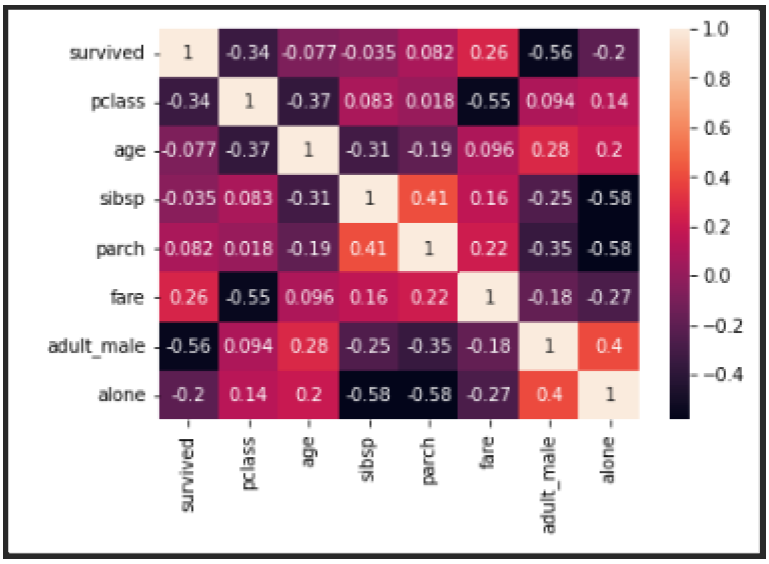
```
corr = dataset.corr()

sns.heatmap(corr)
```



From the output, it can be seen that what heatmap essentially does is that it plots a box for every combination of rows and column value. The colour of the box depends upon the gradient. For instance, in the above image if there is a high correlation between two features, the corresponding cell or the box is white, on the other hand if there is no correlation, the corresponding cell remains black.

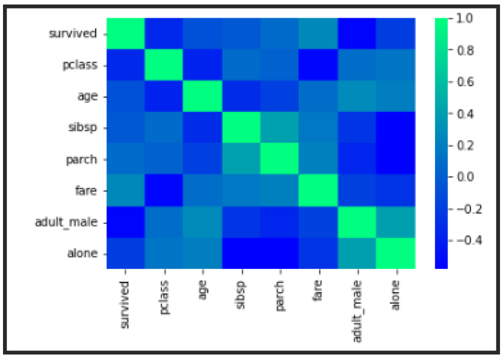
The correlation values can also be plotted on the heatmap by passing True for the annotparameter. Execute the following script to see this in action:

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

annot=True)
```



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

```
corr = dataset.corr()

sns.heatmap(corr)
```



a. Cluster Map:

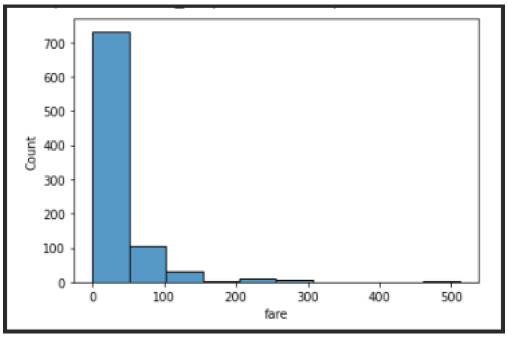
In addition to the heat map, another commonly used matrix plot is the cluster map. The cluster map basically uses Hierarchical Clustering to cluster the rows and columns of the matrix.

Let's plot a cluster map for the number of passengers who travelled in a specific month ofa specific year. Execute the following script:

4. Checking how the price of the ticket (column name: 'fare') for each

passenger isdistributed by plotting a histogram.

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



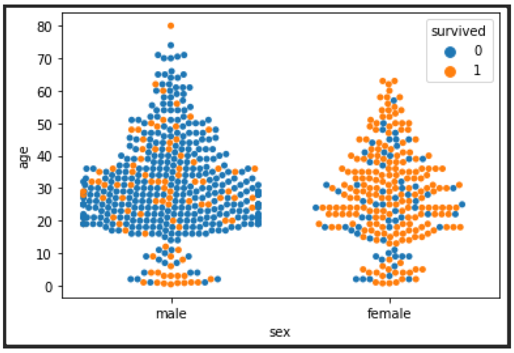
From the histogram, it is seen that for around 730 passengers the price of the ticket is 50. For 100 passengers the price of the ticket is 100 and so on.

Conclusion-

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

Assignment Questions

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



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

Assignment No: 9

Title of the Assignment: Data Visualization II

1. Use the inbuilt dataset 'titanic' as used in the above problem. Plot a box plot for distribution of
age with respect to each gender along with the information about whether they survived or not. (Column names : 'sex' and 'age')
2. Write observations on the inference from the above statistics.

Objective:

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

Prerequisite:

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

Contents for Theory:

1. Box plot Basics
2. Know your Data
3. Finding patterns of data.

Theory:

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

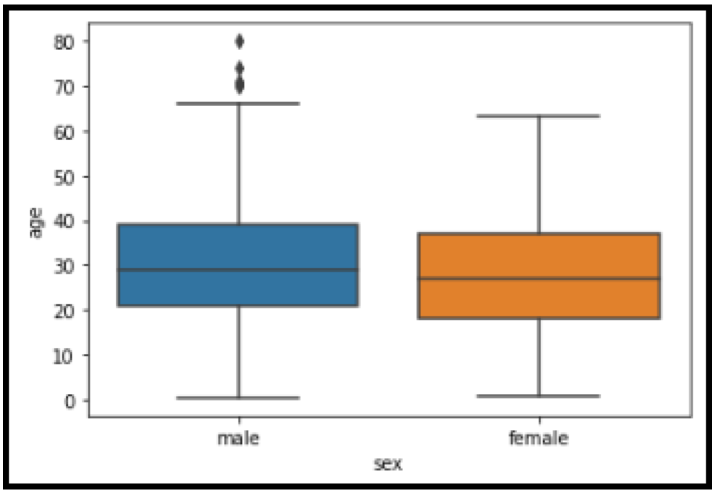
Even plenty of resources can’t be transformed into valuable goods without planning and architecture.

The Box Plot

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

Now let's plot a box plot that displays the distribution for the age with respect to each gender. You need to pass the categorical column as the first parameter (which is sex in our case) and the numeric column (age in our case) as the second parameter. Finally, the dataset is passed as the third parameter, take a look at the following script:

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

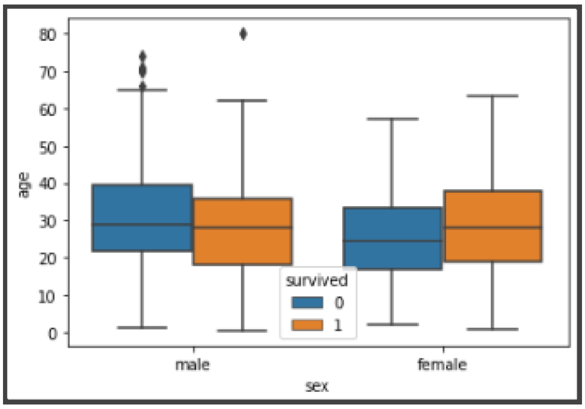


Let's try to understand the box plot for females. The first quartile starts at around 1 and ends at 20 which means that 25% of the passengers are aged between 1 and 20. The second quartile starts at around 20 and ends at around 28 which means that 25% of the passengers are aged between20 and 28. Similarly, the third quartile starts and ends between 28 and 38, hence 25% passengers are aged within this range and finally the fourth or last quartile starts at 38 and ends around 64.

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.

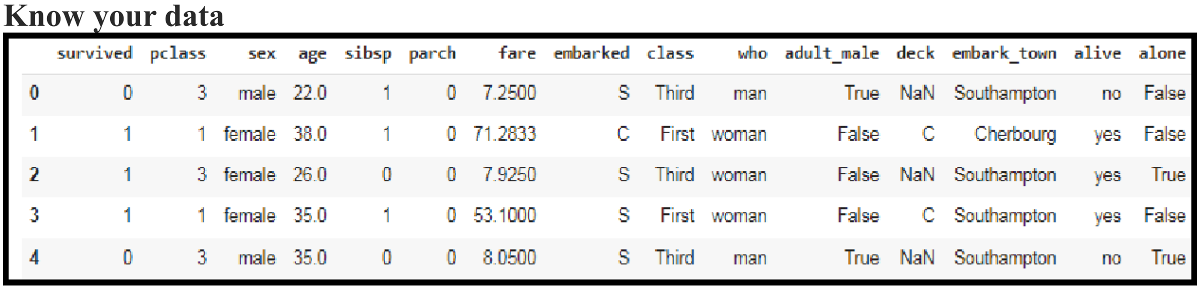
You can make your box plots more fancy by adding another layer of distribution. For instance, ifyou want to see the box plots of forage of passengers of both genders, along with the information about whether or not they survived, you can pass the survived as value to the hue parameter as shown below:

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



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.

Know your data



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

The dataset contains 891 rows and 15 columns and contains information about the passengers who boarded the unfortunate Titanic ship. The original task is to predict whether or not the passenger survived depending upon different features such as their age, ticket, cabin they boarded, the class of the ticket, etc.

Theory:

A box plot is a graphical representation of data that shows the distribution of a dataset. It shows the median, quartiles, and outliers of a dataset. The box represents the interquartile range (IQR), which is the range between the first quartile (Q1) and the third quartile (Q3). The line inside the box represents the median. The whiskers represent the range of data, excluding outliers, and the dots or asterisks represent outliers.

Example:

Here is an example code to plot a box plot for the distribution of age with respect to each gender in the 'titanic' dataset using Python:

```
import seaborn as sns

import matplotlib.pyplot as plt

titanic = sns.load_dataset('titanic')

sns.boxplot(x='sex', y='age', hue='survived', data=titanic)

plt.show()
```

In this code, we first import the required libraries 'seaborn' and 'matplotlib.pyplot'. We then load the 'titanic' dataset using the 'sns.load_dataset()' function. Next, we use the 'sns.boxplot()' function to plot the box plot for the distribution of age with respect to each gender, along with information about whether they survived or not. We pass the 'sex' and 'age' columns of the dataset as the x and y parameters, respectively. We also pass the 'survived' column of the dataset as the hue parameter, which colors the box plot based on whether the passengers survived or not. Finally, we use the 'plt.show()' function to display the plot.

Conclusion-

We learned how to plot a box plot for the distribution of age with respect to each gender in the 'titanic' dataset, along with information about whether they survived or not. We also drew some observations from the statistics. Box plots are a useful tool to visualize the distribution of data and to draw inferences from it.

Assignment Questions

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

4.

Group B
Assignment 11

PROBLEM STATEMENT:

Create databases and tables, insert small amounts of data, and run simple queries using Impala

OBJECTIVE:

- To Learn and understand the steps to create databases and tables in Impala, insert data into these tables, and execute simple queries.
- To learn and understand how to use Impala to analyze data.

Theory:

Impalas

- A tool which we use to overcome the slowness of Hive Queries is what we call Impala.
- Syntactically Impala queries run very faster than Hive Queries even after they are more or less same as Hive Queries.
- It offers high-performance, low-latency SQL queries.
- Impala is the best option while we are dealing with medium-sized datasets and we expect the real-time response from our queries. However, make sure Impala is available only in Hadoop distribution.
- Since MapReduce store intermediate results in the file system, Impala is not built on MapReduce. Hence, it is very slow for real-time query processing.

In addition, Impala has its own execution engine. Basically, that stores the intermediate results in In-memory. Therefore, when compared to other tools which use MapReduce its query execution is very fast.

Some Key Points

- It offers high-performance, low-latency SQL queries.
- Moreover, to share databases and tables between both Impala and Hive it integrates very well with the Hive Metastore.
- Also, it is Compatible with HiveQL Syntax

Impala is an open-source SQL query engine allows you to query data stored in Apache Hadoop clusters. It's a fast, distributed, and highly scalable system that can run complex queries on large datasets in near real-time. In this manual, we will guide you through the process of creating databases and tables, inserting small amounts of data, and running simple queries using Impala.

Impala uses a distributed architecture, which means that data is stored across multiple nodes in a cluster. Impala also uses a query engine that allows you to write SQL queries that are executed in parallel across the nodes in the cluster. This allows for fast query execution, even on large datasets.

Impala uses a metadata store to keep track of the databases and tables that are created. The metadata store is used to store information about the location of the data, the schema of the tables, and other metadata. Impala supports a wide range of data formats, including Parquet, Avro, and RCFile.

Data Types:

BIGINT :- This datatype stores numerical values and the range of this data type is -9223372036854775808 to 9223372036854775807. This datatype is used in create table and alter table statements.

BOOLEAN:-This data type stores only true or false values and it is used in the column definition of create table statement.

CHAR:- This data type is a fixed length storage, it is padded with spaces, you can store up to the maximum length of 255.

DECIMAL:- This data type is used to store decimal values and it is used in create table and alter table statements.

DOUBLE:- This data type is used to store the floating point values in the range of positive or negative 4.94065645841246544e-324d -1.79769313486231570e+308.

FLOAT :- This data type is used to store single precision floating value datatypes in the range of positive or negative 1.40129846432481707e-45 .. 3.40282346638528860e+38.

Example:

To create a database in Impala, you can use the following SQL command:

```
CREATE DATABASE my_database;
```

This will create a new database called "my_database". To create a table within this database, you can use the following SQL command:

```
CREATE TABLE my_table ( id INT,   name STRING,  age INT );
```

This will create a new table called "my_table" with three columns: id, name, and age. To insert data into this table, you can use the following SQL command:

```
INSERT INTO my_table VALUES (1, 'John', 25), (2, 'Jane', 30), (3, 'Bob', 40);
```

This will insert three rows into the table, each with a unique id, name, and age. To run a simple query on this table, you can use the following SQL command:

```
SELECT * FROM my_table;
```

This will return all the rows in the table. You can also use more complex queries, such as aggregations and joins, to analyze the data in the table.

Conclusion:

Impala is a powerful SQL query engine that can be used to analyze large datasets stored in Apache Hadoop clusters. We have seen how to create databases and tables, insert data, and execute simple queries in Impala.

Group B

Assignment No: 12

Problem Statemen:

Write a simple program in SCALA using Apache Spark framework

Theory:

- 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 typesafe 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.

About Scala

- 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
- Scala is not a pure functional language. Haskell is an example of a pure functional language.

Scala is a compiler based language (and not interpreted)

- Scala is a compiler based language which makes Scala execution very fast if you compare it with Python (which is an interpreted language).

- The compiler in Scala works in similar fashion as Java compiler. It gets the source code and generates Java byte-code that can be executed independently on any standard JVM (Java Virtual Machine).
- There are more important points about Scala which I have not covered. Some of them are:
- Scala has thread based executors
- Scala is statically typed language
- Scala can execute Java code
- You can do concurrent and Synchronized processing in Scala
- Scala is JVM based languages

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-java7installer
- 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 scala2.11.7.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.

Variable declaration in Scala

- In Scala, you can declare a variable using 'var' or 'val' keyword. The decision is based on whether it is a constant or a variable.
- If you use 'var' keyword, you define a variable as mutable variable. On the other hand, if you use 'val', you define it as immutable. Let's first declare a variable using "var" and then using "val".

Declare using var

var Var1 : String = "Ankit"

- In the above Scala statement, you declare a mutable variable called "Var1" which takes a string value. You can also write the above statement without specifying the type of variable. Scala will automatically identify it. For example:

  var Var1 = "Joshi"

Operations on Variables

- You can perform various operations on variables. There are various kinds of operators defined in Scala. For example: Arithmetic Operators, Relational Operators, Logical Operators, Bitwise Operators, Assignment Operators.
- Lets see "+" , "==" operators on two variables 'Var4', "Var5". But, before that, let us first assign values to "Var4" and "Var5".

  scala> var Var4 = 2 Output: Var4: Int = 2 scala> var Var5 = 3

  Output: Var5: Int = 3

  Now, let us apply some operations using operators in Scala.

  Apply '+' operator Var4+Var5

  Output: res1: Int = 5 Apply "==" operator

  Var4==Var5 Output: res2: Boolean = false

- In Scala, if-else expression is used for conditional statements.
- YouThe ifcan write-oneelse expression in Scalaor more conditions inside "if".
- Let's declare a variable called "Var3" with a value 1 and then compare "Var3" using if-else expression.

- In the above snippet, the condition evaluates to True and hence True will be printed in the output.

Iteration in Scala

Like most languages, Scala also has a FOR-loop which is the most widely used method for iteration. It has a simple syntax too.

- Declare a simple function in Scala and call it by passing value
- You can define a function in Scala using "def" keyword. Let's define a function called
- "mul2" which will take a number and multiply it by 10.
- You need to define the return type of function, if a function not returning any value you should use the "Unit" keyword.
- In the below example, the function returns an integer value. Let's define the function

“mul2”:

```
def mul2(m: Int): Int = m * 10
```

```
Output: mul2: (m: Int)Int
```

Example:

Here is an example of a simple Scala program that prints "Hello, World!" to the console:

```
object HelloWorld {
  def main(args: Array[String]) {
    println("Hello, World!")
  }
}
```

In this program, we define an object called "HelloWorld". Inside the object, we define a method called "main" that takes an array of strings as an argument. The method simply prints the string "Hello, World!" to the console using the println() method.

To compile and run this program on Ubuntu, you will need to follow these steps:

Install the Scala compiler by running the following command in the terminal:

```
sudo apt-get install scala
```

Save the above code as "HelloWorld.scala" in a directory of your choice.

Open a terminal in the directory where you saved the file and run the following command to compile the code:

```
scalac HelloWorld.scala
```

This will generate a class file called "HelloWorld.class" in the same directory.

Run the program by entering the following command in the terminal:

```
scala HelloWorld
```

This will execute the "main" method of the "HelloWorld" object and print "Hello, World!" to the console.

Assignment Questions

1. Write a prime number code.
2. rite Factorial code.