Savitribai Phule Pune University Faculty of Computer Engineering

310256: Data Science and Big Data Analytics - Laboratory

CE (2019 Course)

Semester- VI

Teaching Scheme		Examination Scheme	
Practical :	4 Hrs. / Week	Term work :	50 Marks
		Practical	25 Marks

LABORATORY MANUAL (Version 1.0)

ACADEMIC YEAR 2021-22

VISION

To provide excellent Information Technology education by building strong teaching and research environment.

MISSION

1) To transform the students into innovative, competent and high quality IT professionals to meet the growing global challenges.
2) To achieve and impart quality education with an emphasis on practical skills and social relevance.
3) To endeavor for continuous up-gradation of technical expertise of students to cater to the needs of the society.
4) To achieve an effective interaction with industry for mutual benefits.

PROGRAM EDUCATIONAL OBJECTIVES

The students of Information Technology course after passing out will

1. Graduates of the program will possess strong fundamental concepts in mathematics, science, engineering and Technology to address technological challenges with emerging trends.
2. Possess knowledge and skills in the field of Computer Science & Engineering and Information Technology for analyzing, designing and implementing multifaceted engineering problems of any domain with innovative and efficient approaches.
3. Acquire an attitude and aptitude for research, entrepreneurship and higher studies in the field of Computer Science & Engineering and Information Technology.
4. Learn commitment to ethical practices, societal contributions through communities and life-long intellect.
5. Attain better communication, presentation, time management and teamwork skills leading to responsible & competent professionals and will be able to address challenges in the field of IT at global level.

PROGRAM OUTCOMES

The students in the Information Technology course will attain:

a. An ability to apply knowledge of computing, mathematics including discrete mathematics as well as probability and statistics, science, engineering and technology.

b. An ability to define a problem and provide a systematic solution with the help of conducting experiments, as well as analyzing and interpreting the data.

c. An ability to design, implement, and evaluate a software or a software/hardware cosystem, component, or process to meet desired needs within realistic constraints.

d. An ability to identify, formulate, and provide systematic solutions to complex engineering problems.

e. An ability to use the techniques, skills, and modern engineering technologies tools, standard processes necessary for practice as an IT professional.

f. An ability to apply mathematical foundations, algorithmic principles, and Information Technology theory in the modeling and design of computer-based systems with necessary constraints and assumptions.

g. An ability to analyze the local and global impact of computing on individuals, organizations and society.

h. An ability to understand professional, ethical, legal, security and social issues and responsibilities.

i. An ability to function effectively as an individual or as a team member to accomplish a desired goal(s).

j. An ability to engage in life-long learning and continuing professional development to cope up with fast changes in the technologies/tools with the help of electives, professional organizations and extra-curricular activities.

k. An ability to communicate effectively in engineering community at large by means of effective presentations, report writing, paper publications, demonstrations.

l. An ability to understand engineering, management, financial aspects, performance, optimizations and time complexity necessary for professional practice.

Savitribai Phule Pune University,
Pune Third Year Computer Engineering (2019 Course)

310256: Data Science and Big Data Analytics

Prerequisite Courses:
- Discrete Mathematics (210241)
- Database Management Systems (310341)

Course Objectives:
1. To understand the need of Data Science and Big Data
2. To understand computational statistics in Data Science
3. To analyze and demonstrate knowledge of statistical data analysis techniques for decision-making
4. To gain practical, hands-on experience with statistics programming languages and big data tools

Course Outcomes: After completion of the course, students should be able to
CO1: Analyze needs and challenges for Data Science Big Data Analytics
CO2: Apply statistics for Big Data Analytics
CO3: Implement and evaluate data analytics algorithms
CO4: Perform text preprocessing
CO5: Implement data visualization techniques
CO6: Use cutting edge tools and technologies to analyze Big Data

INDEX

8	Data Visualization I 1. Use the inbuilt dataset 'titanic'. The dataset contains 891 rows and contains information about the passengers who boarded the unfortunate Titanic ship. Use the Seaborn library to see if we can find any patterns in the data. Write a code to check how the price of the ticket (column name: 'fare') for each passenger is distributed by plotting a histogram		
9	Data Visualization II 1. Use the inbuilt dataset 'titanic' as used in the above problem. Plot a box plot for distribution of age with respect to each gender along with the information about whether they survived or not. (Column names : 'sex' and 'age')		
10	Data Visualization III Download the Iris flower dataset or any other dataset into a DataFrame. (eg https://archive.ics.uci.edu/ml/datasets/Iris). Scan the dataset and give the inference as: 1. How many features are there and what are their types (e.g., numeric, nominal)? 2. Create a histogram for each feature in the dataset to illustrate the feature distributions		
11	Write a code in JAVA for a simple WordCount application that counts the number of occurrences of each word in a given input set using the Hadoop MapReduce framework on local-standalone set-up		
12	Design a distributed application using MapReduce which processes a log file of a system		
13	Locate dataset (eg. sample_weather.txt) for working on weather data which reads the text input files and finds average for temperature, dew point and wind speed.		

Assignment No: 1

Aim/Problem Statement:- Data Wrangling I

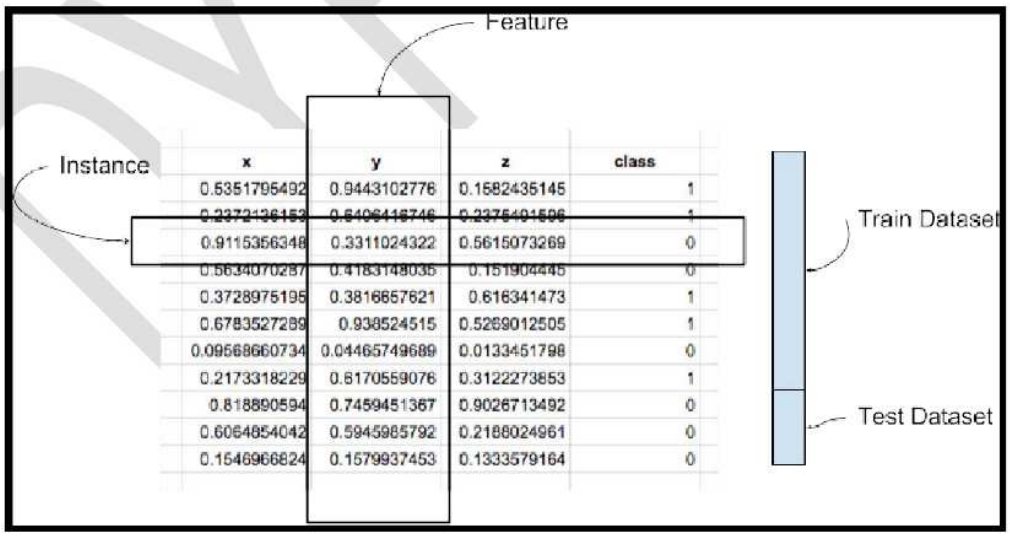
Perform the following operations using Python on any open source dataset (eg. data.csv)
1. Import all the required Python Libraries.
2. Locate an open source data from the web (eg. https://www.kaggle.com). Provide a clear description of the data and its source (i.e. URL of the web site).
3. Load the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

In addition to the codes and outputs, explain every operation that you do in the above steps and explain everything that you do to import/read/scrape the data set.

Theory:-

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.



Instance: A single row of data is called an instance. It is an observation from the domain. **Feature:** A single column of data is called a feature. It is a component of an observation and is also called an

attribute of a data instance. Some features may be inputs to a model (the predictors) and others may be outputs or the features to be predicted.

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

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

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

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

Data Represented in a Table:

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

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

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

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

Pandas Data Types

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

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

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

Python Libraries for Data Science

a) Pandas

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

What can you do with Pandas?

Indexing, manipulating, renaming, sorting, merging data frame

1. Update, Add, Delete columns from a data frame
2. Impute missing files, handle missing data or NANs
3. Plot data with histogram or box plot

b) NumPy

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

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. Matplotlib

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

What can you do with Matplotlib?

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

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

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

c.Seaborn

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

What can you do with Seaborn?

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

d. Scikit Learn

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

What can you do with Scikit Learn?

1.Classification: Spam detection, image recognition

2.Clustering: Drug response, Stock price

3.Regression: Customer segmentation, Grouping experiment outcomes

4.Dimensionality reduction: Visualization, Increased efficiency

5.Model selection: Improved accuracy via parameter tuning

6.Pre-processing: Preparing input data as a text for processing with machine learning algorithms.

Description of Dataset:

The Iris dataset was used in R.A. Fisher's classic 1936 paper, The Use of Multiple Measurements in Taxonomic Problems, and can also be found on the UCI Machine Learning Repository.

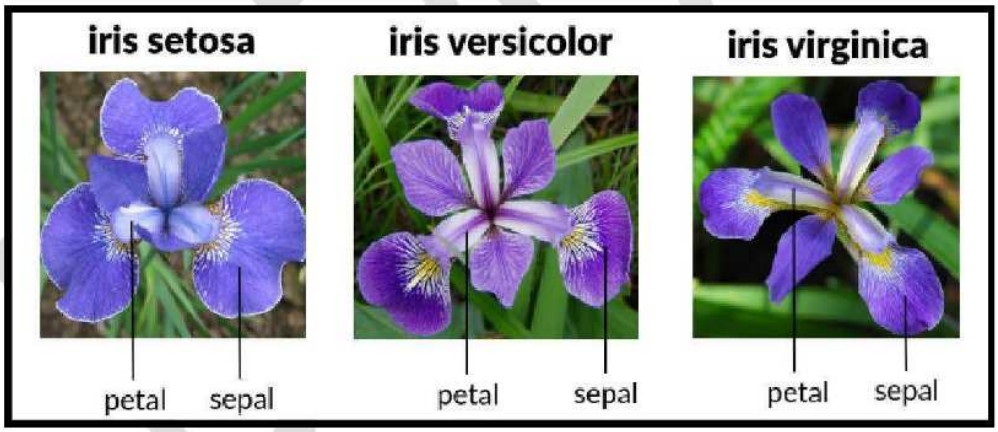
It includes three iris species with 50 samples each as well as some properties about each flower. One flower species is linearly separable from the other two, but the other two are not linearly separable from each other.

Total Sample- 150

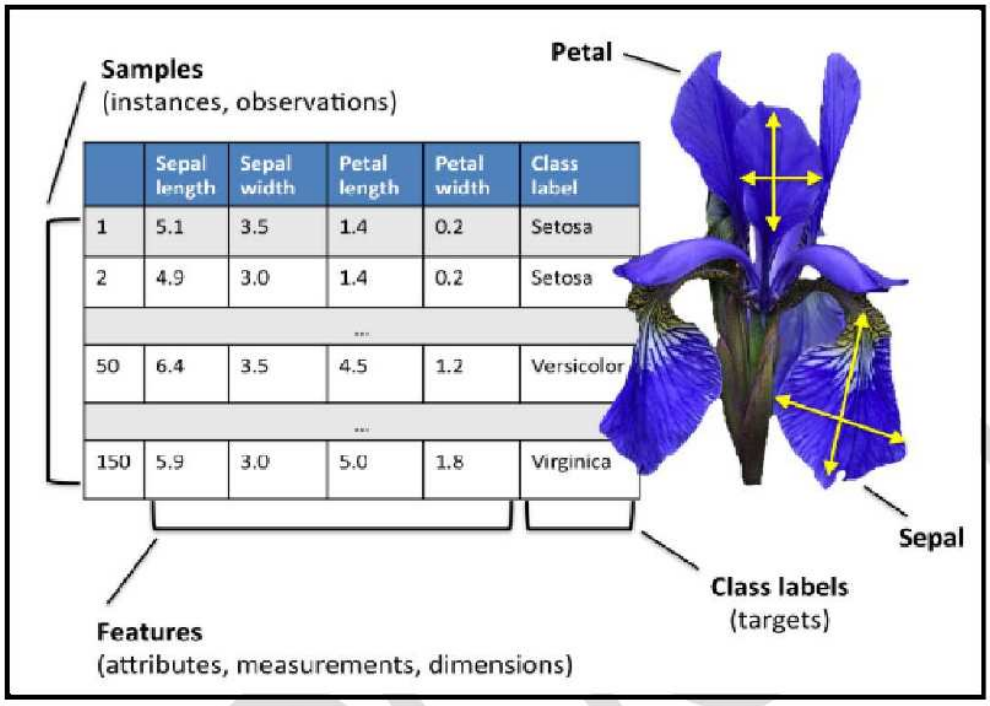
The columns in this dataset are:

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

3 Different Types of Species each contain 50 Sample-



Description of Dataset-



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

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

	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

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

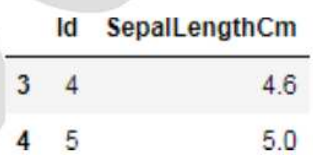
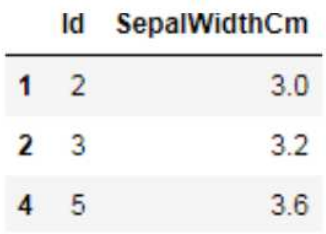
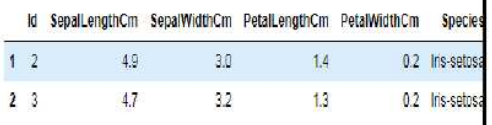
5. **Panda Dataframe functions for Data Preprocessing: Dataframe Operations:**

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

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

14	dataset.loc[:, ["Col_name1", "col_name2"]]	Selection by label
15	dataset.iloc[:n, :]	a subset of the first n rows of the original data
16	dataset.iloc[:, :n]	a subset of the first n columns of the original data
17	dataset.iloc[:m, :n]	a subset of the first m rows and the first n columns

Few Examples of iLoc to slice data for iris Dataset

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

4	**dataset.iloc[:, 1:3]**	For slicing Column explicitly:	SepalLengthCm SepalWidthCm 0 5.1 3.5 1 4.9 3.0 2 4.7 3.2 3 4.6 3.1
4	**dataset.iloc[1, 1]**	For getting a value explicitly:	4.9
5	**dataset['SepalLeng thCm'].iloc[5]**	Accessing Column and Rows by position	5.4
6	**cols_2_4=dataset.c olumns[2:4]** **dataset[cols_2_4]**	Get Column Name then get data from column	SepalWidthCm PetalLengthCm 0 3.5 1.4 1 3.0 1.4 2 3.2 1.3 3 3.1 1.5
7	**dataset[dataset.col umns[2:4]].iloc[5:1 0]**	in one Expression answer for the above two commands	SepalWidthCm PetalLengthCm 5 3.9 1.7 6 3.4 1.4 7 3.4 1.5 8 2.9 1.4 9 3.1 1.5

Checking of Missing Values in Dataset:

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

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

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

Function: DataFrame.isnull()

Output:

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

b. **is there any missing values across each column Function:** DataFrame **.** isnull().any()

Output:

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

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

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

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

Output : 8

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

dataframe.isnull().sum(axis = 1) **Output:**

e. **count Column wise missing value using isnull() Method 1:**

Function: dataframe.isnull().sum()

Output:

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

Method 2:

unction: dataframe.isna().sum()

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

f. count of missing values of a specific column.

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

```
df1.Gender.isnull().sum()
```

Output: 2

g. groupby count of missing values of a column.

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

Function:

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

Output:

6. Panda functions for Data Formatting and Normalization

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

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

Sr. No	Data Frame Function	Description	Output

1.	**df.dtypes**	To check the data type	`df.dtypes`
			`sepal length (cm) float64`
			`sepal width (cm) float64`
			`petal length (cm) float64`
			`petal width (cm) float64`
			`dtype: object`

2.	**df['petal length (cm)']= df['petal length (cm)'].astype("int ")**	To change the data type (data type of 'petal length (cm)'changed to int)	`df.dtypes`
			`sepal length (cm) float64`
			`sepal width (cm) float64`
			`petal length (cm) int64`
			`petal width (cm) float64`
			`dtype: object`

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

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

Algorithm:

Step 1 : Import pandas and sklearn library for preprocessing

```
from sklearn import preprocessing
```

Step 2: Load the iris dataset in dataframe object df

```
iris = load_iris()

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

Step 3: Print iris dataset.

```
df.head()
```

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

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

Step 5: Create a minimum and maximum processor object

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

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

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

Step 7: Run the normalizer on the dataframe

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

Step 8: View the dataframe

```
df_normalized
```

Output: After Step 3:

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

Output after step 8:

	0	1	2	3
0	0.222222	0.625000	0.067797	0.041667
1	0.166667	0.416667	0.067797	0.041667
2	0.111111	0.500000	0.050847	0.041667
3	0.083333	0.458333	0.084746	0.041667
4	0.194444	0.666667	0.067797	0.041667

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

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

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

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

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

Height
Tall
Medium
Short

Height
0
1
2

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

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

Sklearn Functions for Label Encoding:

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

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

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

Algorithm:

Step 1 : Import pandas and sklearn library for preprocessing

```
from sklearn import preprocessing
```

Step 2: Load the iris dataset in dataframe object df

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

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

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

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

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

Step 5: Encode labels in column 'species'.

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

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

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

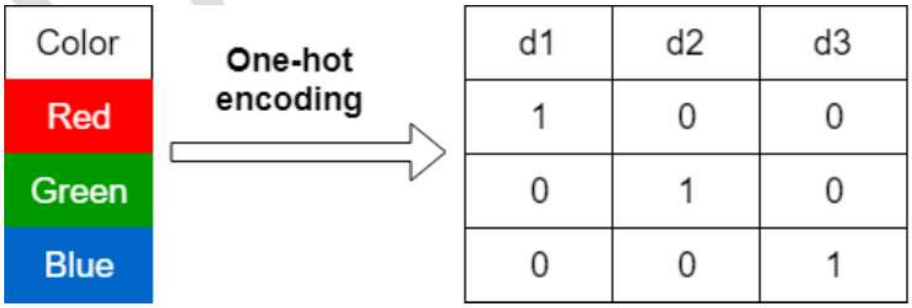
```
Output: array([0, 1, 2], dtype=int64)
```

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

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

b. One-Hot Encoding:

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



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

In one-hot encoding,

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

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

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

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

Sklearn Functions for One-hot Encoding:

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

Algorithm:

Step 1 : Import pandas and sklearn library for preprocessing

```
from sklearn import preprocessing
```

Step 2: Load the iris dataset in dataframe object df

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

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

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

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

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

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

Step 5: Remove the target variable from dataset

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

Step 6: Apply one_hot encoder for Species column.

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

Step 7: Join the encoded values with Features variable

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

Step 8: Observe the merge dataframe

```
df_encode
```

Step 9: Rename the newly encoded columns.

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

Step 10: Observe the merge dataframe

```
df_encode
```

Output after Step 8:

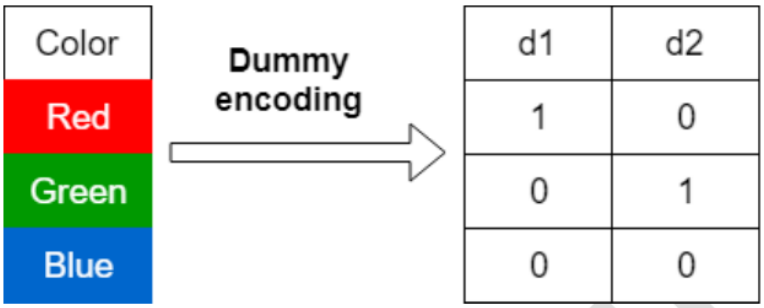
	Sepal_Length	Sepal_Width	Petal_Length	Petal_Width	0	1	2
0	5.1	3.5	1.4	0.2	1.0	0.0	0.0
1	4.9	3.0	1.4	0.2	1.0	0.0	0.0
2	4.7	3.2	1.3	0.2	1.0	0.0	0.0
3	4.6	3.1	1.5	0.2	1.0	0.0	0.0
4	5.0	3.6	1.4	0.2	1.0	0.0	0.0

Output after Step 10:

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

c. Dummy Variable Encoding

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



In dummy encoding,

"Red" color is encoded as **[1 0]** vector of size 2.

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

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

2.

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

Pandas Functions for One-hot Encoding with dummy variables:

- **`pandas.get_dummies(data, prefix=None, prefix_sep='_', dummy_na=False, columns=None, sparse=False, drop_first=False, dtype=None):`** Convert categorical variable into dummy/indicator variables.

- **Parameters**:

  data:array-like, Series, or DataFrame

  Data of which to get dummy indicators.

  prefixstr: list of str, or dict of str, default None

  String to append DataFrame column names.

  prefix_sep: str, default '_'

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

  dummy_nabool: default False

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

columns: list:like, default None

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

Whether the dummy-encoded columns should be backed by a SparseArray (True) or a regular NumPy array (False).

drop_first:bool, default False

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

dtype: dtype, default np.uint8

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

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

Algorithm:

Step 1 : Import pandas and sklearn library for preprocessing

```
from sklearn import preprocessing
```

Step 2: Load the iris dataset in dataframe object df

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

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

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

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

```
df['Species'].unique()
Output: array([0, 1, 2], dtype=int64)
```

Step 6: Apply one_hot encoder with dummy variables for Species column.

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

Step 7: Observe the merge dataframe

```
one_hot_df
```

	Sepal_Length	Sepal_Width	Petal_Length	Petal_Width	Species_0	Species_1	Species_2
0	5.1	3.5	1.4	0.2	1	0	0
1	4.9	3.0	1.4	0.2	1	0	0
2	4.7	3.2	1.3	0.2	1	0	0
3	4.6	3.1	1.5	0.2	1	0	0
4	5.0	3.6	1.4	0.2	1	0	0

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

Assignment Question

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

Assignment No: 2

Aim/Problem Statement: - Data Wrangling I

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

Reason and document your approach properly.

Theory:-

1. **Creation of Dataset using Microsoft Excel.**

   The dataset is created in "CSV" format.

   - The name of dataset is **StudentsPerformance**
   - **The features of the dataset are:** Math_Score, Reading_Score, Writing_Score, Placement_Score, Club_Join_Date .
   - **Number of Instance**s: 30
   - **The response variable is**: Placement_Offer_Count .
   - **Range of Values:**

     Math_Score [60-80], Reading_Score[75-,95], ,Writing_Score [60,80], Placement_Score[75-100], Club_Join_Date [2018-2021].
   - **The response variable is** the number of placement offers facilitated to particular students, which is largely depend on Placement_Score

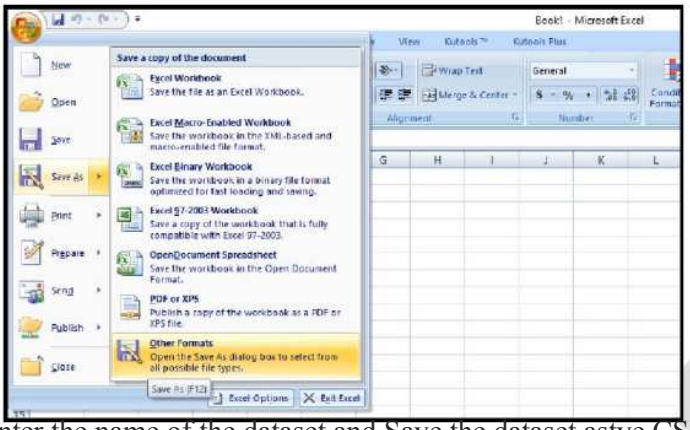
   To fill the values in the dataset the **RANDBETWEEN** is used. Returns a random integer number between the numbers you specify

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

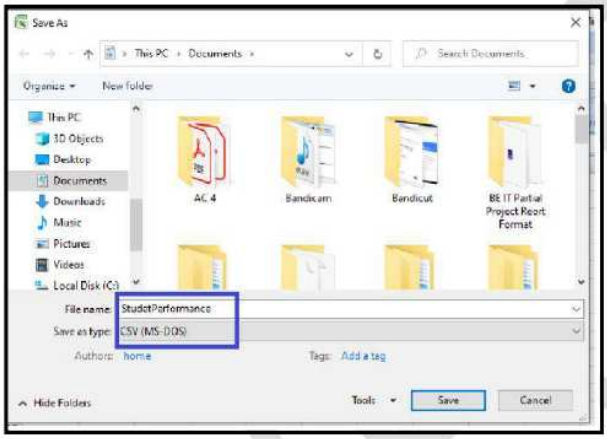
   Top The largest integer RANDBETWEEN will return.

For better understanding and visualization, 20% impurities are added into each variable to the dataset.
The step to create the dataset are as follows:

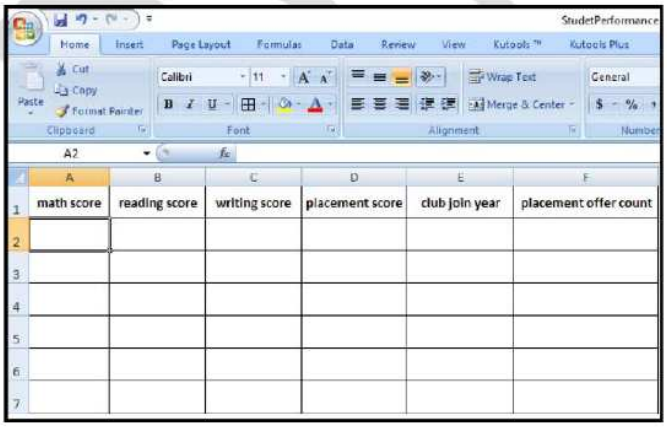
Step 1: Open Microsoft Excel and click on Save As. Select Other .Formats



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

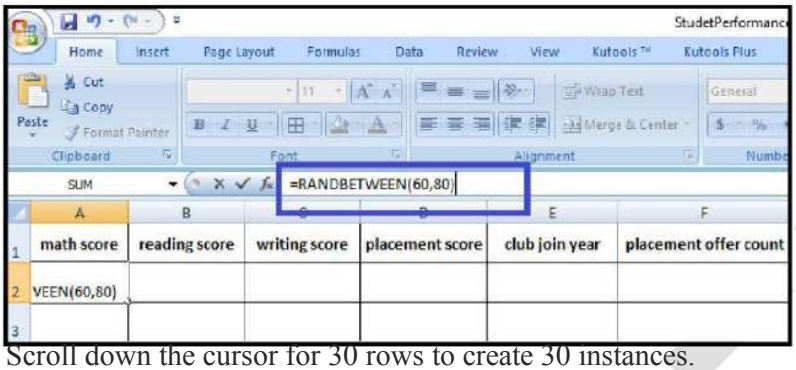


Step 3: Enter the name of features as column header.



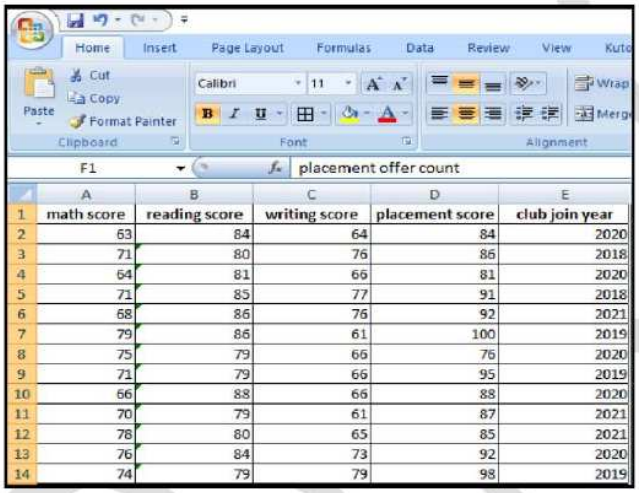
Step 3: Fill the dara by using **RANDOMBETWEEN** function. For every feature , fill the data by considering the above specified range.

one example is given:



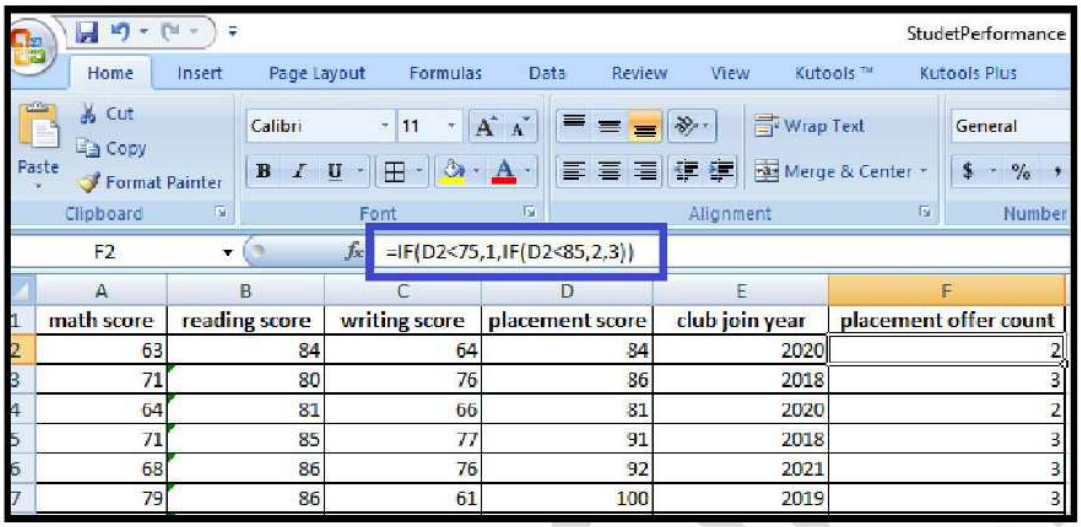
Scroll down the cursor for 30 rows to create 30 instances.

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



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

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



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

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

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

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

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

The dataset is created with the given description.

2. Identification and Handling of Null Values

Missing Data can occur when no information is provided for one or more items or for a whole unit. Missing Data is a very big problem in real-life scenarios. Missing Data can also refer to as NA(Not Available) values in pandas. In DataFrame sometimes many datasets simply arrive

with missing data, either because it exists and was not collected or it never existed. For Example, Suppose different users being surveyed may choose not to share their income, some users may choose not to share the address in this way many datasets went missing.

In Pandas missing data is represented by two value:

1. **None**: None is a Python singleton object that is often used for missing data in Python code.
2. **NaN** : NaN (an acronym for Not a Number), is a special floating-point value recognized by all systems that use the standard IEEE floating-point representation.Pandas treat None and NaN as essentially interchangeable for indicating missing or null values. To facilitate this convention, there are several useful functions for detecting, removing, and replacing null values in Pandas DataFrame :

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

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

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

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

Algorithm:

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

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

Step 2: Load the dataset in dataframe object df

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

Step 3: Display the data frame

```
df
```

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

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

```
df.isnull()
```

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

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

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

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

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

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

Algorithm:

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

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

Step 2: Load the dataset in dataframe object df

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

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

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

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

```
df.notnull()
```

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

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

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

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

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

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

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

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

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

- **For replacing null values with NaN**

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

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

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

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

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

Step 2: Load the dataset in dataframe object df

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

Step 3: Display the data frame

```
df
```

Step 4: filling missing value using fillna()

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

	gender	math score	reading score	writing score	Placement Score	placement offer count	Region
0	female	72	72	74.0	78.0	1	Pune
1	female	69	90	88.0	0.0	2	na
2	female	90	95	93.0	74.0	2	Nashik
3	male	47	57	0.0	78.0	1	Na
4	male	na	78	75.0	61.0	3	Pune
5	female	71	Na	78.0	70.0	4	na
6	male	12	44	52.0	12.0	2	Nashik
7	male	0	65	67.0	49.0	1	Pune
8	male	5	77	89.0	55.0	0	0

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

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

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

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

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

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

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

- **Filling null values in dataset**

To fill null values in dataset use inplace=true

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

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

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

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

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

	gender	math score	reading score	writing score	Placement Score	placement offer count	Region
0	female	72	72	74.0	78.0	1	Pune
1	female	69	90	88.0	-99.0	2	na
2	female	90	95	93.0	74.0	2	Nashik
3	male	47	57	-99.0	78.0	1	Na
4	male	na	78	75.0	81.0	3	Pune
5	female	71	Na	78.0	70.0	4	na
6	male	12	44	52.0	12.0	2	Nashik
7	male	-99	65	67.0	49.0	1	Pune
8	male	5	77	89.0	55.0	0	-99

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

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

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

Algorithm:

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

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

Step 2: Load the dataset in dataframe object df

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

Step 3: Display the data frame

```
df
```

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

```
ndf.dropna()
```

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

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

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

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

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

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

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

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

making new data frame with dropped NA values

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

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

3. Identification and Handling of Outliers

3.1 Identification of Outliers

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

1. What are Outliers?

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

Similarly, an Outlier is an observation in a given dataset that lies far from the rest of

the observations. That means an outlier is vastly larger or smaller than the rema
the set.

2. Why do they occur?

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

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

3. What do they affect?

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

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

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

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

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

fig. Computation with and without outlier

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

4. Detecting Outliers

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

Below are some of the techniques of detecting outliers

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

4.1 Detecting outliers using Boxplot:

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

Algorithm:

Step 1 : Import pandas and numpy libraries

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

Step 2: Load the dataset in dataframe object df

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

Step 3: Display the data frame

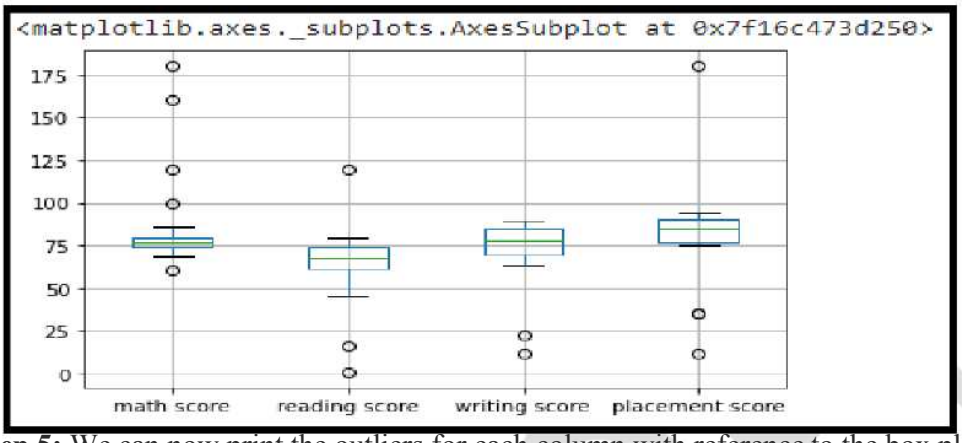
```
df
```

	math score	reading score	writing score	placement score	placement offer count
0	80	68	70	89	3
1	71	61	65	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	86	87	3
10	160	67	12	83	2
11	79	72	88	180	2
12	80	80	78	94	3

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

```
col = ['math score', 'reading score' , 'writing
score','placement score']
```

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



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

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

4.2 Detecting outliers using Scatterplot:

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

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

Algorithm:

Step 1 : Import pandas , numpy and matplotlib libraries

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

Step 2: Load the dataset in dataframe object df

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

Step 3: Display the data frame

```
df
```

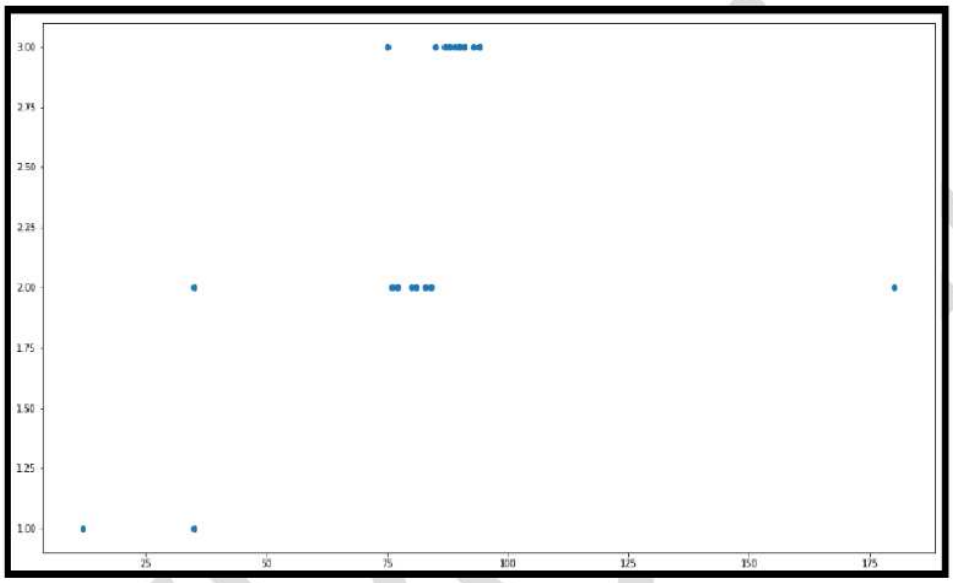
Step 4: Draw the scatter plot with placement score and placement offer count

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

```
count'])
        plt.show()
```

Labels to the axis can be assigned (Optional)

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



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

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

4.3 Detecting outliers using Z-Score:

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

Zscore = (data_point -mean) / std. deviation

Algorithm:

Step 1 : Import numpy and stats from scipy libraries

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

Step 2: Calculate Z-Score for mathscore column

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

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

```
print(z)
```

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

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

```
threshold = 0.18
```

Step 5: Display the sample outliers

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

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

4.4 Detecting outliers using Inter Quantile Range(IQR):

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

IQR = Quartile3 – Quartile1

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

upper = Q3 +1.5*IQR

lower = Q1 – 1.5*IQR

In the above formula as according to statistics, the 0.5 scale-up of IQR (new_IQR = IQR + 0.5*IQR) is taken.

Algorithm:

Step 1 : Import numpy library

```
import numpy as np
```

Step 2: Sort Reading Score feature and store it into sorted_rscore.

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

Step 3: Print sorted_rscore

```
sorted_rscore
```

Step 4: Calculate and print Quartile 1 and Quartile 3

```
q1 = np.percentile(sorted_rscore, 25) q3 = np.percentile(sorted_rscore, 75) print(q1,q3)
```

```
62.0 74.0
```

Step 5: Calculate value of IQR (Inter Quartile Range)

```
IQR = q3-q1
```

Step 6: Calculate and print Upper and Lower Bound to define the outlier base value.

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

```
44.0 92.0
```

Step 7: Print Outliers

```
r_outliers = []
for i in sorted_rscore:
  if (i<lwr_bound or i>upr_bound):
    r_outliers.append(i)
print(r_outliers)
```

3.2 Handling of Outliers:

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

Below are some of the methods of treating the outliers

- Trimming/removing the outlier

- Quantile based flooring and capping
- Mean/Median imputation

Trimming/removing the outlier:

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

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

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

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

- **Quantile based flooring and capping:**

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

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

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

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

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

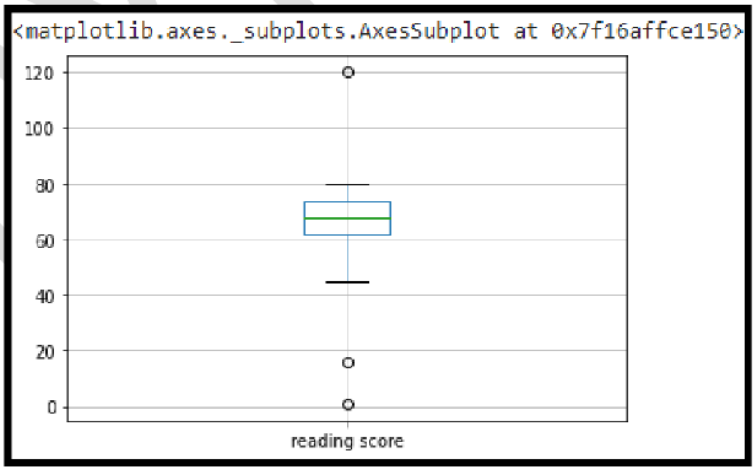
	math score	m score	reading score	writing score	placement score	placement offer count
0	80	80.0	68	70	89	3
1	71	71.0	61	85	91	3
2	79	79.0	16	87	77	2
3	61	61.0	77	74	76	2
4	78	78.0	71	67	90	3
5	73	73.0	68	90	80	2
6	77	77.0	62	70	35	2
7	74	74.0	45	80	12	1

- **Mean/Median imputation:**

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

1. Plot the box plot for reading score `col = ['reading score']`

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



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

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

```
median
```

4. Replace the upper bound outliers using median value

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

5. Display redefined_df

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

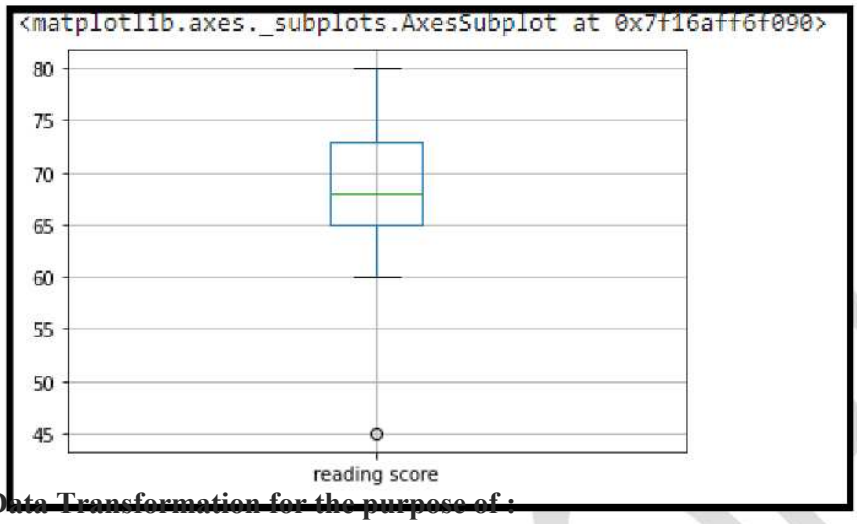
6. Replace the lower bound outliers using median value `refined_df['reading score'] = np.where(refined_df['reading score'] <lwr_bound, median,refined_df['reading score'])`

7. Display redefined_df

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

8. Draw the box plot for redefined_df `col = ['reading score']`

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



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

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

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

converted into categorical value (young, old).

- **Normalization:** Data normalization involves converting all data variables into a given range. Some of the techniques that are used for accomplishing normalization are:
  - **Min–max normalization**: This transforms the original data linearly.
  - **Z-score normalization**: In z-score normalization (or zero-mean normalization) the values of an attribute (A), are normalized based on the mean of A and its standard deviation.
  - **Normalization by decimal scaling**: It normalizes the values of an attribute by changing the position of their decimal points
- **Attribute or feature construction**.
  - **New attributes constructed from the given ones**: Where new attributes are created & applied to assist the mining process from the given set of attributes. This simplifies the original data & makes the mining more efficient.

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

a. To change the scale for better understanding (Attribute or feature construction)

Here the Club_Join_Date is transferred to Duration.

Algorithm:

Step 1 : Import pandas and numpy libraries

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

Step 2: Load the dataset in dataframe object df

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

Step 3: Display the data frame

```
df
```

```
df
```

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

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

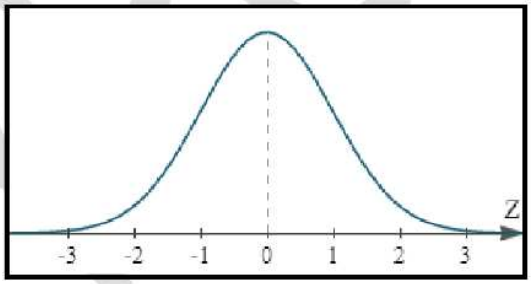
```
[11] df
```

	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.

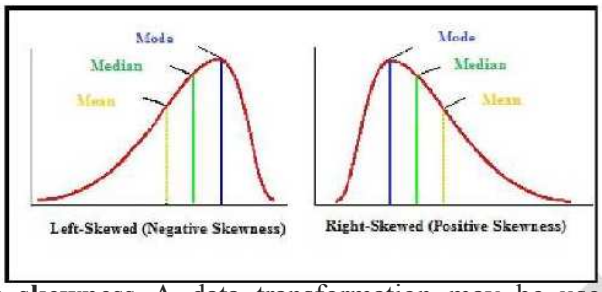
Normal Distribution: In a normal distribution, the graph appears as a classical, symmetrical "bell-shaped curve." The mean, or average, and the mode, or maximum point on the curve, are equal.



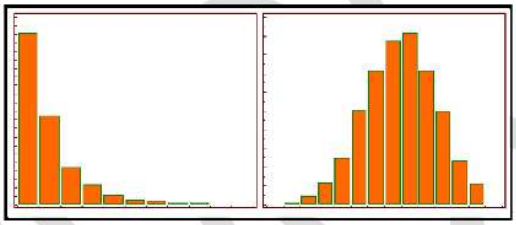
Positively Skewed Distribution

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 e of x (ln x), or x to log base 2 of x, is a strong transformation with a major effect on distribution shape. It is commonly used for reducing right skewness and is often appropriate for measured variables. It can not be applied to zero or negative values.



Algorithm:

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

Step 2: Observe the histogram for math_score variable.

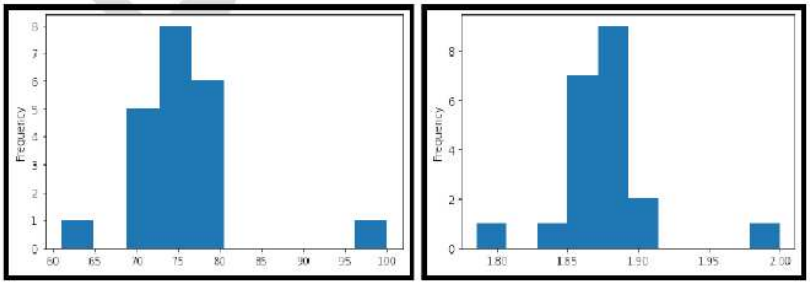
```
import matplotlib.pyplot as plt new_df['math score'].plot(kind = 'hist')
```

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

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

Step 4: Observe the histogram for math_score variable.

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



It is observed that skewness is reduced at some level.

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

Assignment Question:

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

.

Assignment No: 3

Aim/Problem Statement:-Basic Statistics - Measures of Central Tendencies and Variance

Perform the following operations on any open source dataset (eg. 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. .

Theory:-

Statistical Inference:

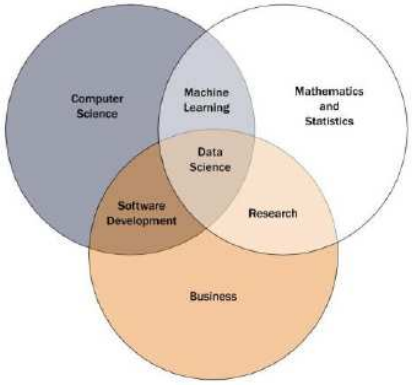
statistical inference as the process of generating conclusions about a population from a noisy sample. Without statistical inference we're simply living within our data. With statistical inference, we're trying to generate new knowledge.

Statistical analysis and probability influence our lives on a daily basis. Statistics is used to predict the weather, restock retail shelves, estimate the condition of the economy, and much more. Used in a variety of professional fields, statistics has the power to derive valuable insights and solve complex problems in business, science, and society. Without hard science, decision making relies on emotions and gut reactions. Statistics and data override intuition, inform decisions, and minimize risk and uncertainty.

In data science, statistics is at the core of sophisticated machine learning algorithms, capturing and translating data patterns into actionable evidence. Data scientists use statistics to gather, review, analyze, and draw conclusions from data, as well as apply quantified mathematical models to appropriate variables.

Data science knowledge is grouped into three main areas: computer science; statistics and mathematics; and business or field expertise. These areas separately result in a variety of careers, as displayed in the diagram below. Combining computer science and statistics without business knowledge enables professionals to perform an array of machine learning functions. Computer science and business expertise leads to software development skills. Mathematics and statistics (combined with business

expertise) result in some of the most talented researchers. It is only with all three areas combined that data scientists can maximize their performance, interpret data, recommend innovative solutions, and create a mechanism to achieve improvements.



Statistical functions are used in data science to analyze raw data, build data models, and infer results. Below is a list of the key statistical terms:

- Population: the source of data to be collected.
- Sample: a portion of the population.
- Variable: any data item that can be measured or counted.
- Quantitative analysis (statistical): collecting and interpreting data with patterns and data visualization.
- Qualitative analysis (non-statistical): producing generic information from other non-data forms of media.
- Descriptive statistics: characteristics of a population.
- Inferential statistics: predictions for a population.
- Central tendency (measures of the center): mean (average of all values), median (central value of a data set), and mode (the most recurrent value in a data set).
- Measures of the Dispersion:
    - Range: the distance between each value in a data set.
    - Variance: the distance between a variable and its expected value.
    - Standard deviation: the dispersion of a data set from the mean.

Statistical techniques for data scientists

There are a number of statistical techniques that data scientists need to master. When just starting out, it is important to grasp a comprehensive understanding of these principles, as any holes in knowledge will result in compromised data or false conclusions.

General statistics: The most basic concepts in statistics include bias, variance, mean, median, mode, and percentiles.

Probability distributions: Probability is defined as the chance that something will occur, characterized as a simple "yes" or "no" percentage. For instance, when weather reporting indicates a 30 percent chance of rain, it also means there is a 70 percent chance it will not rain. Determining the distribution calculates the probability that all those potential values in the study will occur. For example, calculating the probability that the 30 percent chance for rain will change over the next two days is an example of probability distribution.

Dimension reduction: Data scientists reduce the number of random variables under consideration through feature selection (choosing a subset of relevant features) and feature extraction (creating new features from functions of the original features). This simplifies data models and streamlines the process of entering data into algorithms.

Over and under sampling: Sampling techniques are implemented when data scientists have too much or too little of a sample size for a classification. Depending on the balance between two sample groups, data scientists will either limit the selection of a majority class or create copies of a minority class in order to maintain equal distribution.

Bayesian statistics: Frequency statistics uses existing data to determine the probability of a future event. Bayesian statistics, however, takes this concept a step further by accounting for factors we predict will be true in the future. For example, imagine trying to predict whether at least 100 customers will visit your coffee shop each Saturday over the next year. Frequency statistics will determine probability by analyzing data from past Saturday visits. But Bayesian statistics will determine probability by also factoring for a nearby art show that will start in the summer and take place every Saturday afternoon. This allows the Bayesian statistical model to provide a much more accurate figure.

The goals of inference

1. Estimate and quantify the uncertainty of an estimate of a population quantity (the proportion of people who will vote for a candidate).
2. Determine whether a population quantity is a benchmark value ("is the treatment effective?").
3. Infer a mechanistic relationship when quantities are measured with noise ("What is the slope for Hooke's law?")
4. Determine the impact of a policy? ("If we reduce pollution levels, will asthma rates decline?")
5. Talk about the probability that something occurs.

Algorithm:-

Step 1. Import Dataset:

```
train_df=pd.read_csv('train.csv')
test_df=pd.read_csv('test.csv')
train_df.shape, test_df.shape

train_df['label']='train'
test_df['label']='test'

combined_data_df=pd.concat([train_df,test_df])
combined_data_df.shape
#The reasons for combining both training and test dataset are:

#To find missing values in both the datasets
#If we need transform/remove any features, we can do it in both datasets at one time
#To convert categorical variable to numerical variable in both datasets
```

Step 2. Statistical Inference

```
#Statistical Analysis
combined_data_df.mean()
combined_data_df.median()
combined_data_df.mode()
combined_data_df.std()
combined_data_df.describe()
combined_data_df.min()
combined_data_df.max()
combined_data_df.dtypes
```

Step 3. Handling Missing Values

```
#Handle the missing value
combined_data_df.info()
#Data types in data set:

#Categorical = 10
#Numerical = 5
#Target =1
combined_data_df.isnull().sum()
combined_data_df.dropna(subset=['workclass','occupation','native-country'],axis=0,inplace=True)
combined_data_df.isnull().sum()
combined_data_df.dropna(subset=['income_>50K'],axis=0,inplace=True)
combined_data_df.isnull().sum()
```

Step 4. Data Visualization

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

sns.set_theme(style="darkgrid")

#frequency distribution of work class
plt.figure(figsize=(10,10))
sns.countplot(data= combined_data_df, x = combined_data_df['workclass'])
combined_data_df.drop(combined_data_df.index[combined_data_df['workclass'] == 'Without-pay'],
inplace=True)
combined_data_df.shape
plt.figure(figsize=(10,15))
sns.countplot(data= combined_data_df, y = "native-country")
#This graph clearly shows that the given dataset in from US. We can remove other countries since
almost of them are US origin.
combined_data_df=combined_data_df[combined_data_df['native-country']=='United-States']
combined_data_df.shape
#Since, now we only have US as the only native country, this feature is useless. We can drop this feature
altogether
combined_data_df=combined_data_df.drop(columns='native-country',axis=1)
combined_data_df.shape
#frequency distribution of education class

plt.figure(figsize=(20,10))
sns.countplot(data= combined_data_df, x = "education")
combined_data_df['education'] = combined_data_df['education'].replace(['1st-4th','5th-6th'],'elementary-
school')
combined_data_df['education'] = combined_data_df['education'].replace(['7th-8th'],'middle-school')
combined_data_df['education'] = combined_data_df['education'].replace(['9th','10th','11th','12th'],'high-
school')
combined_data_df['education'] = combined_data_df['education'].replace(['Doctorate','Bachelors','Some-
college','Masters','Prof-school','Assoc-voc','Assoc-acdm'],'postsecondary-education')

plt.figure(figsize=(20,10))
sns.countplot(data= combined_data_df, x = "education")

plt.figure(figsize=(20,10))
sns.countplot(data= combined_data_df, x = "marital-status")

combined_data_df['marital-status'] = combined_data_df['marital-status'].replace(['Divorced','Never-
married','Widowed'],'single')
```

```
combined_data_df['marital-status'] = combined_data_df['marital-status'].replace(['Married-civ-spouse','Separated','Married-spouse-absent','Married-AF-spouse'],'married')
plt.figure(figsize=(20,10))
plt.figure()
sns.countplot(data= combined_data_df, x = "marital-status")

plt.figure(figsize=(20,10))
sns.countplot(data= combined_data_df, y = "occupation")

plt.figure(figsize=(20,10))
sns.countplot(data= combined_data_df, x = "relationship")
```

Step 5. convert categorical variable to numerical variable

```
#Split the dataset into categorical and numerical values
#categorical
 cat_columns = [ col for col in list(combined_data_df.columns) if combined_data_df[col].dtype ==='object' and col!= 'label']
cat_columns

#numberical num_columns = [ col for col in list(combined_data_df.columns) if combined_data_df[col].dtype in ['int64','float64']]
num_columns fig= plt.figure(figsize=(15,15))
corr_matrix = combined_data_df.corr()
sns.heatmap(data=corr_matrix,annot=True)
plt.show()
combined_data_df.drop(columns='fnlwgt',inplace=True)
combined_data_df.shape
#Converting categorical variables to numerical ( Dummy variables )
#get dummies
features_df = pd.get_dummies(data=combined_data_df, columns=cat_columns)
features_df.shape
features_df.columns
#split your data
train_df = features_df[features_df['label'] == 'train']
test_df = features_df[features_df['label'] == 'test']
# Drop your labels
train_df = train_df.drop('label', axis=1)
test_df = test_df.drop(columns=['label','income_>50K'], axis=1)
train_df.shape, test_df.shape
train_df.columns
train_df.isnull().sum().sum()
```

Input: train.csv, test.csv

Output: Performed statistical analysis on the income prediction dataset and also converted categorical data into numerical data.

Conclusion:-

- Handled the missing value, by dropping them from the dataset
- From the data visualization, combined/categorized the features
- Using dummy variable, converted categorical variable to numerical variable to create better model.

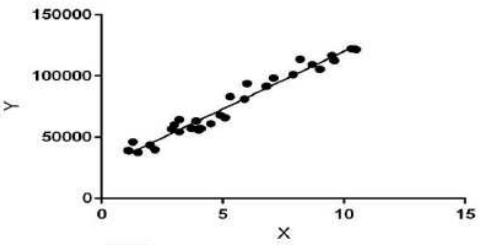
Assignment No: 4

Aim/Problem Statement:- Data Analytics I

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

Theory:-

Linear Regression: It is a machine learning algorithm based on **supervised learning**. It performs a **regression task**. Regression models a target prediction value based on independent variables. It is mostly used for finding out the relationship between variables and forecasting. Different regression models differ based on – the kind of relationship between dependent and independent variables, they are considering and the number of independent variables being used.



Linear regression performs the task to predict a dependent variable value (y) based on a given independent variable (x). So, this regression technique finds out a linear relationship between x (input) and y(output). Hence, the name is Linear Regression.

In the figure above, X (input) is the work experience and Y (output) is the salary of a person. The regression line is the best fit line for our model.

Hypothesis function for Linear Regression :

$$y = \theta_1 + \theta_2 . x$$

While training the model we are given :
x: input training data (univariate – one input variable(parameter))
y: labels to data (supervised learning)

When training the model – it fits the best line to predict the value of y for a given value of x. The model gets the best regression fit line by finding the best θ_1 and θ_2 values.
θ_1: intercept
θ_2: coefficient of x
Once we find the best θ_1 and θ_2 values, we get the best fit line. So when we are finally using our model for prediction, it will predict the value of y for the
input value of x.

How to update θ_1 and θ_2 values to get the best fit line ?
Cost Function (J):
By achieving the best-fit regression line, the model aims to predict y value such that the error difference between predicted value and true value is minimum. So, it is very important to update the θ_1 and θ_2 values, to reach the best value that minimize the error between predicted y value (pred) and true y value (y).

$$minimize \frac{1}{n}\sum_{i=1}^{n}(pred_i - y_i)^2$$

$$J = \frac{1}{n}\sum_{i=1}^{n}(pred_i - y_i)^2$$

Cost function(J) of Linear Regression is the **Root Mean Squared Error (RMSE)** between predicted y value (pred) and true y value (y).
Gradient Descent:
~~To update θ_1 and θ_2~~ values in order to reduce Cost function (minimizing RMSE value) and achieving the best fit line the model uses Gradient Descent. The idea is to start with random θ_1 and θ_2 values and then iteratively updating the values, reaching minimum cost.

Algorithm:-

Step 1: Download the data set of Boston Housing Prices
(https://www.kaggle.com/c/boston-housing).
Step 2: Importing Libraries
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
```
Step 3: Importing Data
```
from sklearn.datasets import load_boston
boston = load_boston()
```
Step 4: Converting data from nd-array to data frame and adding feature names to the data
```
data = pd.DataFrame(boston.data)
data.columns = boston.feature_names
```
Step 5: Adding 'Price' (target) column to the data

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

Step 6: Getting input and output data and further splitting data to training and testing dataset.

```
# Input Data
x = boston.data

# Output Data
y = boston.target
```

Step 7: splitting data to training and testing dataset.

```
#from sklearn.cross_validation import train_test_split
#the submodule cross_validation is renamed and reprecated to model_selection
from sklearn.model_selection import train_test_split

xtrain, xtest, ytrain, ytest = train_test_split(x, y, test_size =0.2,random_state = 0)
```

Step 8: #Applying Linear Regression Model to the dataset and predicting the prices.

```
# Fitting Multi Linear regression model to training model
from sklearn.linear_model import LinearRegression
regressor = LinearRegression()
regressor.fit(xtrain, ytrain)

# predicting the test set results
y_pred = regressor.predict(xtest)
```

Step 9: Plotting Scatter graph to show the prediction

```
# results - 'ytrue' value vs 'y_pred' value
plt.scatter(ytest, y_pred, c = 'green')
plt.xlabel("Price: in $1000's")
plt.ylabel("Predicted value")
plt.title("True value vs predicted value : Linear Regression")
plt.show()
```

Step 10: Results of Linear Regression.

```
from sklearn.metrics import mean_squared_error
mse = mean_squared_error(ytest, y_pred)
print("Mean Square Error : ", mse)
```

Input: Dataset of Boston Housing Prices. This dataset concerns the housing prices in the housing city of Boston. The dataset provided has 506 instances with 14 features.

Output: Prediction of Boston Housing Prices by plotting graph to show prediction

Conclusion:-Housing Prices of Boston city predicted using Linear Regression

Questions:

1) What is Linear regression
2) What are different types of linear regressions
3) Applications where linear regression is used
4) What are the limitations of linear regression

Assignment No: 5

Aim/Problem Statement:- Data Analytics II

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

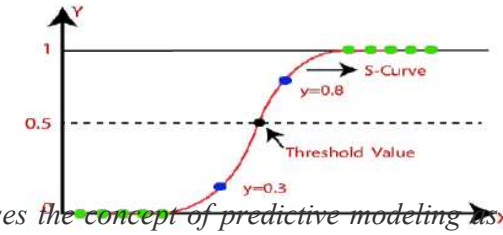
Theory:-

Logistic regression is one of the most popular Machine Learning algorithms, which comes under the Supervised Learning technique. It is used for predicting the categorical dependent variable using a given set of independent variables.

Logistic regression predicts the output of a categorical dependent variable. Therefore the outcome must be a categorical or discrete value. It can be either Yes or No, 0 or 1, true or False, etc. but instead of giving the exact value as 0 and 1, **it gives the probabilistic values which lie between 0 and 1**.

Logistic Regression is much similar to the Linear Regression except that how they are used. Linear Regression is used for solving Regression problems, whereas **Logistic regression is used for solving the classification problems**.

- In Logistic regression, instead of fitting a regression line, we fit an "S" shaped logistic function, which predicts two maximum values (0 or 1).
- The curve from the logistic function indicates the likelihood of something such as whether the cells are cancerous or not, a mouse is obese or not based on its weight, etc.
- Logistic Regression is a significant machine learning algorithm because it has the ability to provide probabilities and classify new data using continuous and discrete datasets.
- Logistic Regression can be used to classify the observations using different types of data and can easily determine the most effective variables used for the classification. The below image is showing the logistic function:



Note: *Logistic regression uses the concept of predictive modeling as regression; therefore, it is called logistic regression, but is used to classify samples; Therefore, it falls under the classification algorithm.*

Logistic Function (Sigmoid Function):

- The sigmoid function is a mathematical function used to map the predicted values to probabilities.
- It maps any real value into another value within a range of 0 and 1.
- The value of the logistic regression must be between 0 and 1, which cannot go beyond this limit, so it forms a curve like the "S" form. The S-form curve is called the Sigmoid function or the logistic function.
- In logistic regression, we use the concept of the threshold value, which defines the probability of either 0 or 1. Such as values above the threshold value tends to 1, and a value below the threshold values tends to 0.

Assumptions for Logistic Regression:

- The dependent variable must be categorical in nature.
- The independent variable should not have multi-collinearity.

Logistic Regression Equation:

The Logistic regression equation can be obtained from the Linear Regression equation. The mathematical steps to get Logistic Regression equations are given below:

- We know the equation of the straight line can be written as:

$$y= b_0 + b_1x_1 + b_2x_2 + b_3x_3 + \cdots + b_nx_n$$

- In Logistic Regression y can be between 0 and 1 only, so for this let's divide the above equation by (1-y):

$$\frac{y}{1-y}\text{; 0 for y= 0, and infinity for y=1}$$

- But we need range between -[infinity] to +[infinity], then take logarithm of the equation it will become:

$$log\left[\frac{y}{1-y}\right] = b_0 + b_1x_1 + b_2x_2 + b_3x_3 + \cdots + b_nx_n$$

The above equation is the final equation for Logistic Regression.

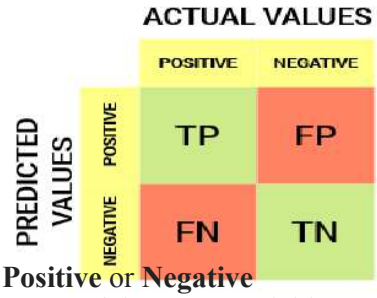
Type of Logistic Regression:

On the basis of the categories, Logistic Regression can be classified into three types:

- **Binomial:** In binomial Logistic regression, there can be only two possible types of the dependent variables, such as 0 or 1, Pass or Fail, etc.

- **Multinomial:** In multinomial Logistic regression, there can be 3 or more possible unordered types of the dependent variable, such as "cat", "dogs", or "sheep"
- **Ordinal:** In ordinal Logistic regression, there can be 3 or more possible ordered types of dependent variables, such as "low", "Medium", or "High".

A Confusion matrix is an N x N matrix used for evaluating the performance of a classification model, where N is the number of target classes. The matrix compares the actual target values with those predicted by the machine learning model. This gives us a holistic view of how well our classification model is performing and what kinds of errors it is making.
For a binary classification problem, we would have a 2 x 2 matrix as shown below with 4 values:



The target variable has two values: **Positive** or **Negative**
The **columns** represent the **actual values** of the target variable
The **rows** represent the **predicted values** of the target variable

True Positive (TP)
The predicted value matches the actual value
The actual value was positive and the model predicted a positive value

True Negative (TN)
The predicted value matches the actual value
The actual value was negative and the model predicted a negative value

False Positive (FP) – Type 1 error
The predicted value was falsely predicted
The actual value was negative but the model predicted a positive value
Also known as the **Type 1 error**

False Negative (FN) – Type 2 error
The predicted value was falsely predicted
The actual value was positive but the model predicted a negative value
Also known as the **Type 2 error**

Accuracy is calculated as below

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

Precision tells us how many of the correctly predicted cases actually turned out to be positive. Here's how to calculate Precision:

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

This would determine whether our model is reliable or not.
Recall tells us how many of the actual positive cases we were able to predict correctly with our model.
And here's how we can calculate Recall:

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

Algorithm:-

Step 1: Download the data set of Social_Network_Ads
(https://www.kaggle.com/)

Step 2: Importing Libraries

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

Step 3: Importing Data

```
dataset = pd.read_csv('Social_Network_Ads.csv')
#select only age and salary as the features
x = dataset.iloc[:, [2, 3]].values
y = dataset.iloc[:, 4].values
```

Step 4: Perform splitting for training and testing. We will take 75% of the data for training, and test on the remaining data

```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.25, random_state = 0)
```

Step 5: Scale the features to avoid variation and let the features follow a normal distribution

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

Step 6:Fit the Model

```
from sklearn.linear_model import LogisticRegression
classifier = LogisticRegression(random_state = 0)
classifier.fit(X_train, y_train)
```

Step 7: Predict the labels of test data

```
y_pred = classifier.predict(X_test)
```

Step 8: Evaluate performance of the model

```
From sklearn.metrics import confusion_matrix,classification_report
cm = confusion_matrix(y_test, y_pred)
print(cm)
```

Step 9: Evaluate Accuracy depending on Confusion Matrix

```
#Accuray=(TN+TP)/Total
```

Step 10: Evaluate error rate

```
#Error_rate=(FN+FP)/Total
```

Step 11: Compute Precision and Recall

```
cl_report=classification_report(y_test,y_pred)

cl_report
```

Input: Social_Network_Ads.csv dataset

Output: Classification using Logistic Regression, Computed Confusion matrix to find TP, FP, TN, FN, Accuracy, Error rate, Precision, Recall on the given dataset.

Conclusion:- Implemented logistic regression to perform classification on Social_Network_Ads.csv dataset using python

Questions:

1) What is logistic regression
2) How it is different from linear regression
3) What are the types of logistic regression
4) What are the limitations of logistic regression
5) List application where logistic regression can be applied

Assignment No: 6

Aim/Problem Statement:- Data Analytics III

Implement Simple Naïve Bayes classification algorithm using Python/R on iris.csv dataset.
Compute Confusion matrix to find TP, FP, TN, FN, Accuracy, Error rate, Precision, Recall on the given dataset.

Theory:-

Naïve Bayes algorithm is a supervised learning algorithm, which is based on **Bayes theorem** and used for solving classification problems. It is mainly used in *text classification* that includes a high-dimensional training dataset. Naïve Bayes Classifier is one of the simple and most effective Classification algorithms which helps in building the fast machine learning models that can make quick predictions. **It is a probabilistic classifier, which means it predicts on the basis of the probability of an object**. Some popular examples of Naïve Bayes Algorithm are **spam filtration, Sentimental analysis, and classifying articles**.

The Naïve Bayes algorithm is comprised of two words Naïve and Bayes, Which can be described as:

- **Naïve**: It is called Naïve because it assumes that the occurrence of a certain feature is independent of the occurrence of other features. Such as if the fruit is identified on the bases of color, shape, and taste, then red, spherical, and sweet fruit is recognized as an apple. Hence each feature individually contributes to identify that it is an apple without depending on each other.
- **Bayes**: It is called Bayes because it depends on the principle of Bayes' Theorem.

Bayes' Theorem:

- Bayes' theorem is also known as **Bayes' Rule** or **Bayes' law**, which is used to determine the probability of a hypothesis with prior knowledge. It depends on the conditional probability.
- The formula for Bayes' theorem is given as:

$$P(A|B)= \frac{P(B|A)P(A)}{P(B)}$$

Where,

P(A|B) is Posterior probability: Probability of hypothesis A on the observed event B.
P(B|A) is Likelihood probability: Probability of the evidence given that the probability of a hypothesis is true.
P(A) is Prior Probability: Probability of hypothesis before observing the evidence.
P(B) is Marginal Probability: Probability of Evidence.

Advantages of Naïve Bayes Classifier:

- Naïve Bayes is one of the fast and easy ML algorithms to predict a class of datasets.
- It can be used for Binary as well as Multi-class Classifications.

- It performs well in Multi-class predictions as compared to the other Algorithms.
- It is the most popular choice for **text classification problems**.

Disadvantages of Naïve Bayes Classifier:

- Naive Bayes assumes that all features are independent or unrelated, so it cannot learn the relationship between features.

Applications of Naïve Bayes Classifier:

- It is used for **Credit Scoring**.
- It is used in **medical data classification**.
- It can be used in **real-time predictions** because Naïve Bayes Classifier is an eager learner.
- It is used in Text classification such as **Spam filtering** and **Sentiment analysis**.

Types of Naïve Bayes Model:

There are three types of Naive Bayes Model, which are given below:

- **Gaussian**: The Gaussian model assumes that features follow a normal distribution. This means if predictors take continuous values instead of discrete, then the model assumes that these values are sampled from the Gaussian distribution.
- **Multinomial**: The Multinomial Naïve Bayes classifier is used when the data is multinomial distributed. It is primarily used for document classification problems, it means a particular document belongs to which category such as Sports, Politics, education, etc. The classifier uses the frequency of words for the predictors.
- **Bernoulli**: The Bernoulli classifier works similar to the Multinomial classifier, but the predictor variables are the independent Booleans variables. Such as if a particular word is present or not in a document. This model is also famous for document classification tasks.

Algorithm:-

Step 1: Download the data set of Iris
(https://www.kaggle.com/uciml/iris)

Step 2: Importing Libraries

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

Step 3: Assign the 4 independent variables to *X* and the dependent variable 'species' to *Y*.
The first 5 rows of the dataset are displayed

```
X = dataset.iloc[:,:4].values
y = dataset['species'].valuesdataset.head(5)
```

Step 4: Splitting the dataset into the Training set and Test set

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

Step 5: Feature Scaling

```
from sklearn.preprocessing import StandardScaler
sc = StandardScaler()
```

```
X_train = sc.fit_transform(X_train)
X_test = sc.transform(X_test)
```

Step 6: Training the Naive Bayes Classification model on the Training Set

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

Step 7: Predicting the Test set results

```
y_pred = classifier.predict(X_test)
y_pred
```

Step 8: Confusion Matrix and Accuracy

```
from sklearn.metrics import confusion_matrix,classification_report
cm = confusion_matrix(y_test, y_pred)
from sklearn.metrics import accuracy_score
print ("Accuracy : ", accuracy_score(y_test, y_pred))
print(cm)
print("Error rate:",(1-accuracy_score(y_test, y_pred)))
```

Step 9: Compute Precision and Recall

```
cl_report=classification_report(y_test,y_pred)
print(cl_report)
```

Step 10: #Comparing the Real Values with Predicted Values

```
df = pd.DataFrame({'Real Values':y_test, 'Predicted Values':y_pred})
df
```

Input: Iris Dataset

Output: Confusion matrix, Accuracy, Error rate, Precision, Recall on the given dataset.

Conclusion: Implemented successfully Simple Naïve Bayes classification algorithm using Python on iris.csv dataset

Questions:

1) What is confusion matrix
2) How to calculate Accuracy, precision and recall
3) Explain applications of naïve Bays classification algorithm
4) State advantages and limitations of Naïve Bays algorithm

Assignment No: 7

Aim/Problem Statement:- Text Analytics

Extract sample document and apply following document preprocessing methods:

1) Tokenization, POS tagging, stop words removal, stemming and lammetization
2) Create representation of document by calculating term frequency and inverse document frequency

Theory:-

Natural language processing is one of the fields in programming where the natural language is processed by the software. This has many applications like sentiment analysis, language translation, fake news detection, grammatical error detection etc.

The input in natural language processing is text. The data collection for this text happens from a lot of sources. This requires a lot of cleaning and processing before the data can be used for analysis.

These are some of the methods of processing the data in NLP:

- Tokenization
- Stop words removal
- Stemming
- Normalization
- Lemmatization
- Parts of speech tagging

Tokenization:

Tokenization is breaking the raw text into small chunks. Tokenization breaks the raw text into words, sentences called tokens. These tokens help in understanding the context or developing the model for the NLP. The tokenization helps in interpreting the meaning of the text by analyzing the sequence of the words.

For example, the text “It is raining” can be tokenized into ‘It’, ‘is’, ‘raining’

There are different methods and libraries available to perform tokenization. NLTK, Gensim, Keras are some of the libraries that can be used to accomplish the task.

Tokenization can be done to either separate words or sentences. If the text is split into words using some separation technique it is called word tokenization and same separation done for sentences is called sentence tokenization.

There are various tokenization techniques available which can be applicable based on the language and purpose of modeling. Below are a few of the tokenization techniques used in NLP.

White Space Tokenization

This is the simplest tokenization technique. Given a sentence or paragraph it tokenizes into words by splitting the input whenever a white space in encountered. This is the fastest tokenization technique but will work for languages in which the white space breaks apart the sentence into meaningful words. Example: English.

Dictionary Based Tokenization

In this method the tokens are found based on the tokens already existing in the dictionary. If the token is not found, then special rules are used to tokenize it. It is an advanced technique compared to whitespace tokenizer.

Rule Based Tokenization

In this technique a set of rules are created for the specific problem. The tokenization is done based on the rules. For example creating rules bases on grammar for particular language.

Regular Expression Tokenizer

This technique uses regular expression to control the tokenization of text into tokens. Regular expression can be simple to complex and sometimes difficult to comprehend. This technique should be preferred when the above methods does not serve the required purpose. It is a rule based tokenizer.

Penn TreeBank Tokenization

Tree bank is a corpus created which gives the semantic and syntactical annotation of language. Penn Treebank is one of the largest treebanks which was published. This technique of tokenization separates the punctuation, clitics (words that occur along with other words like I'm, don't) and hyphenated words together.

Spacy Tokenizer

This is a modern technique of tokenization which faster and easily customizable. It provides the flexibility to specify special tokens that need not be segmented or need to be segmented using special rules. Suppose you want to keep $ as a separate token, it takes precedence over other tokenization operations.

Moses Tokenizer

This is a tokenizer which is advanced and is available before Spacy was introduced. It is basically a collection of complex normalization and segmentation logic which works very well for structured language like English.

Subword Tokenization

This tokenization is very useful for specific application where sub words make significance. In this technique the most frequently used words are given unique ids and less frequent words are split into sub words and they best represent the meaning independently. For example if the word few is appearing frequently in the text it will be assigned a unique id, where fewer and fewest which are rare words and are less frequent in the text will be split into sub words like few, er, and est. This helps the language model not to learn fewer and fewest as two separate words. This allows to identify the unknown words in the data set during training. There are different types of subword tokenization and they are given below and Byte-Pair Encoding and WordPiece will be discussed briefly.

- Byte-Pair Encoding (BPE)
- WordPiece
- Unigram Language Model
- SentencePiece

Byte-Pair Encoding (BPE)

This technique is based on the concepts in information theory and compression. BPE uses Huffman encoding for tokenization meaning it uses more embedding or symbols for representing less frequent words and less symbols or embedding for more frequently used words.

The BPE tokenization is bottom up sub word tokenization technique. The steps involved in BPE algorithm is given below.

1. Starts with splitting the input words into single unicode characters and each of them corresponds to a symbol in the final vocabulary.
2. Find the most frequent occurring pair of symbols from the current vocabulary.
3. Add this to the vocabulary and size of vocabulary increases by one.
4. Repeat steps ii and iii till the defined number of tokens are built or no new combination of symbols exist with required frequency.

WordPiece

WordPiece is similar to BPE techniques expect the way the new token is added to the vocabulary. BPE considers the token with most frequent occurring pair of symbols to merge into the vocabulary. While WordPiece considers the frequency of individual symbols also and based on below count it merges into the vocabulary.

Count (x, y) = frequency of (x, y) / frequency (x) * frequency (y)

The pair of symbols with maximum count will be considered to merge into vocabulary. So it allows rare tokens to be included into vocabulary as compared to BPE.

Tokenization with NLTK

NLTK (natural language toolkit) is a python library developed by Microsoft to aid in NLP.

```
#Tokenization

#Splitting a sentece into words(tokes)
```

```
#Learn tokenize with nltk

# tokenization with nltk

print("\ntokenize with nltk: ",nlp.word_tokenize(data))

Input: Text can be sentences, strings, words, characters and large documents.
Now lets create a sentence to understand basics of text mining methods.
Our sentence is "no woman no cry" from Bob Marley
playing swimming dancing played danced

Output: tokenize with nltk:  ['Text', 'can', 'be', 'sentences', ',', 'strings',
',', 'words', ',', 'characters', 'and', 'large', 'documents', '.', 'Now', 'lets
', 'create', 'a', 'sentence', 'to', 'understand', 'basics', 'of', 'text', 'mini
ng', 'methods', '.', 'Our', 'sentence', 'is', '``', 'no', 'woman', 'no', 'cry',
 "''", 'from', 'Bob', 'Marley', 'playing', 'swimming', 'dancing', 'played', 'da
nced']
```

Stemming:

Stemming is a natural language processing technique that lowers inflection in words to their root forms, hence aiding in the preprocessing of text, words, and documents for text normalization.
According to Wikipedia, inflection is the process through which a word is modified to communicate many grammatical categories, including tense, case, voice, aspect, person, number, gender, and mood. Thus, although a word may exist in several inflected forms, having multiple inflected forms inside the same text adds redundancy to the NLP process.
As a result, we employ stemming to reduce words to their basic form or stem, which may or may not be a legitimate word in the language.
For instance, the stem of these three words, connections, connected, connects, is “connect”. On the other hand, the root of trouble, troubled, and troubles is “troubl,” which is not a recognized word.

Why Stemming is Important?
As previously stated, the English language has several variants of a single term. The presence of these variances in a text corpus results in data redundancy when developing NLP or machine learning models. Such models may be ineffective.
To build a robust model, it is essential to normalize text by removing repetition and transforming words to their base form through stemming.

Application of Stemming
In information retrieval, text mining SEOs, Web search results, indexing, tagging systems, and word analysis, stemming is employed. For instance, a Google search for prediction and predicted returns comparable results.

Martin Porter invented the Porter Stemmer or Porter algorithm in 1980. Five steps of word reduction are used in the method, each with its own set of mapping rules. Porter Stemmer is the original stemmer and

is renowned for its ease of use and rapidity. Frequently, the resultant stem is a shorter word with the same root meaning.
PorterStemmer() is a module in NLTK that implements the Porter Stemming technique. Let us examine this with the aid of an example.

Example of PorterStemmer()

In the example below, we construct an instance of PorterStemmer() and use the Porter algorithm to stem the list of words.

```
from nltk.stem import PorterStemmer

porter = PorterStemmer()

words =
['Connects','Connecting','Connections','Connected','Connection','Connectings','Connect']

for word in words:

    print(word,"--->",porter.stem(word))
```

```
Connects ---> connect

Connecting ---> connect

Connections ---> connect

Connected ---> connect

Connection ---> connect

Connectings ---> connect

Connect ---> connect
```

Lemmatization:

Lemmatization is another technique which is used to reduce words to a **normalized form**. In lemmatization, the transformation uses a **dictionary** to map different variants of a word back to its root format. So, with this approach, we are able to reduce non trivial inflections such as “is”, “was”, “were” back to the root “be”.

import the **WordNetLemmatizer** and try it out by the following example!

```
# lemmatization

lemma = nlp.WordNetLemmatizer()

lemma_roots = [lemma.lemmatize(each) for each in words_list]

print("result of lemmatization: ",lemma_roots)
```

```
result of lemmatization:  ['text', 'can', 'be', 'sentences,', 'strings,', 'word
s,', 'character', 'and', 'large', 'documents.\nnow', 'let', 'create', 'a', 'sen
tence', 'to', 'understand', 'basic', 'of', 'text', 'mining', 'methods.', '\nour
', 'sentence', 'is', '"no', 'woman', 'no', 'cry"', 'from', 'bob', 'marley\nplay
ing', 'swimming', 'dancing', 'played', 'danced']
```

Lemmatization is similar to stemming with one difference i.e. the final form is also a **meaningful word**. Thus, stemming operation does not need a dictionary like lemmatization.

Stopword:

There different techniques for removing stop words from strings in Python. Stop words are those words in natural language that have a very little meaning, such as "is", "an", "the", etc. Search engines and other enterprise indexing platforms often filter the stop words while fetching results from the database against the user queries.

Stop words are often removed from the text before training deep learning and machine learning models since stop words occur in abundance, hence providing little to no unique information that can be used for classification or clustering.

Removing Stop Words with Python

With the Python programming language, we have a myriad of options to use in order to remove stop words from strings. we can either use one of the several natural language processing libraries such as NLTK, SpaCy, Gensim, TextBlob, etc.,

There are number of different approaches, depending on the NLP library in use

- Stop Words with NLTK
- Stop Words with Gensim
- Stop Words with SpaCy

Using Python's NLTK Library

The NLTK library is one of the oldest and most commonly used Python libraries for Natural Language Processing. NLTK supports stop word removal, and we can find the list of stop words in the corpus module. To remove stop words from a sentence, we can divide your text into words and then remove the word if it exits in the list of stop words provided by NLTK.

Steps Involved in the POS tagging

- Tokenize text (word_tokenize)
- apply pos_tag to above step that is nltk.pos_tag(tokenize_text)

NLTK POS Tags Examples are as below:

Abbreviation	Meaning
CC	coordinating conjunction
CD	cardinal digit
DT	Determiner
EX	existential there
FW	foreign word
IN	preposition/subordinating conjunction
JJ	This NLTK POS Tag is an adjective (large)
JJR	adjective, comparative (larger)
JJS	adjective, superlative (largest)
LS	list market
MD	modal (could, will)
NN	noun, singular (cat, tree)
NNS	noun plural (desks)

Abbreviation	Meaning
NNP	proper noun, singular (sarah)
NNPS	proper noun, plural (indians or americans)
PDT	predeterminer (all, both, half)
POS	possessive ending (parent\ ‘s)
PRP	personal pronoun (hers, herself, him, himself)
PRP$	possessive pronoun (her, his, mine, my, our)
RB	adverb (occasionally, swiftly)
RBR	adverb, comparative (greater)
RBS	adverb, superlative (biggest)
RP	particle (about)
TO	infinite marker (to)
UH	interjection (goodbye)
VB	verb (ask)
VBG	verb gerund (judging)
VBD	verb past tense (pleaded)
VBN	verb past participle (reunified)
VBP	verb, present tense not 3rd person singular(wrap)
VBZ	verb, present tense with 3rd person singular (bases)
WDT	wh-determiner (that, what)
WP	wh- pronoun (who)

Abbreviation	Meaning
WRB	wh- adverb (how)

The above NLTK POS tag list contains all the NLTK POS Tags. NLTK POS tagger is used to assign grammatical information of each word of the sentence. Installing, Importing and downloading all the packages of POS NLTK is complete.

import nltk

nltk.download('averaged_perceptron_tagger')

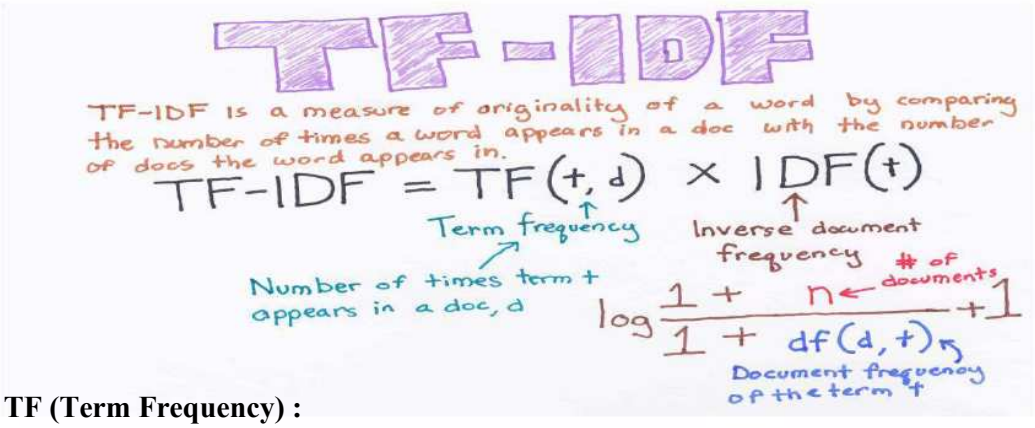
print("\nPOS tagging:",nltk.pos_tag(text_tokens))

```
POS tagging: [('Text', 'NN'), ('can', 'MD'), ('be', 'VB'), ('sentences', 'NNS')
, (',', ','), ('strings', 'NNS'), (',', ','), ('words', 'NNS'), (',', ','), ('c
haracters', 'NNS'), ('and', 'CC'), ('large', 'JJ'), ('documents', 'NNS'), ('.',
 '.'), ('Now', 'RB'), ('lets', 'VBZ'), ('create', 'VB'), ('a', 'DT'), ('sentenc
e', 'NN'), ('to', 'TO'), ('understand', 'VB'), ('basics', 'NNS'), ('of', 'IN'),
 ('text', 'NN'), ('mining', 'NN'), ('methods', 'NNS'), ('.', '.'), ('Our', 'PRP
$'), ('sentence', 'NN'), ('is', 'VBZ'), ('``', '``'), ('no', 'DT'), ('woman', '
NN'), ('no', 'DT'), ('cry', 'NN'), ("''", "''"), ('from', 'IN'), ('Bob', 'NNP')
, ('Marley', 'NNP'), ('playing', 'VBG'), ('swimming', 'VBG'), ('dancing', 'VBG'
), ('played', 'VBN'), ('danced', 'VBD')]
```

What is Feature Engineering of text data?

The procedure of converting raw text data into machine understandable format(numbers) is called feature engineering of text data. Machine learning and deep learning algorithm performance and accuracy is fundamentally dependent on the type of feature engineering techniques used.

we are going to see Feature Engineering technique using TF-IDF and mathematical calculation of TF, IDF and TF-IDF to understand the insights of mathematical logic behind libraries such as *TfidfTransformer* from *sklearn.feature_extraction* package in python.



TF (Term Frequency) :

- Term frequency is simply the count of a word present in a sentence
- TF is basically capturing the importance of the word irrespective of the length of the document.
- A word with the frequency of 3 with the length of sentence being 10 is not the same as when the word length of sentence being 100 words. It should get more importance in the first scenario; that is what TF does.

IDF (Inverse Document Frequency):

- IDF of each word is the log of the ratio of the total number of rows to the number of rows in a particular document in which that word is present.
- IDF will measure the rareness of a term. word like 'a' and 'the' show up in all the documents of corpus, but the rare words is not in all the documents.

TF-IDF:

It is the simplest product of TF and IDF so that both of the drawbacks are addressed above, which makes predictions and information retrieval relevant.

Now import TfidfTransformer and CountVectorizer from sklearn.feature_extraction module.

```
from sklearn.feature_extraction.text import TfidfTransformer
from sklearn.feature_extraction.text import CountVectorizer
```

CountVectorizer is used to find the word count in each document of a dataset. Also known as to calculate Term Frequency

```
docs=['the              cat              see              the              mouse',
'the         house         has         a         tiny         little         mouse',
'the         mouse         ran         away         from         the         house',
'the            cat            finally            ate            the            mouse',
'the             end             of             the             mouse             story'
]
```

```
#Extracting features by using TfidfTransformer from sklearn.feature_extraction package
from sklearn.feature_extraction.text import TfidfTransformer
from sklearn.feature_extraction.text import CountVectorizer

#fit all the sentences using count-vectorizer and get an array of word counts of each document
import pandas as pd
cv = CountVectorizer()
word_count_vector = cv.fit_transform(docs)
tf = pd.DataFrame(word_count_vector.toarray(), columns=cv.get_feature_names())
print(tf)
```

```
ate  away  cat  end  finally  from  has  house  little  mouse  of  ran  \
0    0     0    1    0        0     0    0      0       0      1   0    0
1    0     0    0    0        0     0    1      1       1      1   0    0
2    0     1    0    0        0     1    0      1       0      1   0    1
3    1     0    1    0        1     0    0      0       0      1   0    0
4    0     0    0    1        0     0    0      0       0      1   1    0

   see  story  the  tiny
0    1      0    2     0
1    0      0    1     1
2    0      0    2     0
3    0      0    2     0
4    0      1    2     0
```

```
#declare TfidfTransformer() instance and fit it with the above word_count_vector to get IDF and final normalized features
tfidf_transformer = TfidfTransformer()
X = tfidf_transformer.fit_transform(word_count_vector)
idf = pd.DataFrame({'feature_name':cv.get_feature_names(), 'idf_weights':tfidf_transformer.idf_})
print(idf)
```

```
feature_name  idf_weights
0            ate     2.098612
1           away     2.098612
2            cat     1.693147
3            end     2.098612
4        finally     2.098612
5           from     2.098612
6            has     2.098612
7          house     1.693147
8         little     2.098612
9          mouse     1.000000
10            of     2.098612
11           ran     2.098612
12           see     2.098612
13         story     2.098612
```

```
14          the     1.000000
15         tiny     2.098612
```

#Final feature vectors are
tf_idf = pd.DataFrame(X.toarray() ,columns=cv.get_feature_names())
print(tf_idf)

```
ate      away       cat       end   finally      from       has  \
0   0.000000  0.000000  0.483344  0.000000  0.000000  0.000000  0.000000
1   0.000000  0.000000  0.000000  0.000000  0.000000  0.000000  0.493562
2   0.000000  0.457093  0.000000  0.000000  0.000000  0.457093  0.000000
3   0.513923  0.000000  0.414630  0.000000  0.513923  0.000000  0.000000
4   0.000000  0.000000  0.000000  0.491753  0.000000  0.000000  0.000000

      house    little     mouse        of       ran       see     story  \
0   0.000000  0.000000  0.285471  0.000000  0.000000  0.599092  0.000000
1   0.398203  0.493562  0.235185  0.000000  0.000000  0.000000  0.000000
2   0.368780  0.000000  0.217807  0.000000  0.457093  0.000000  0.000000
3   0.000000  0.000000  0.244887  0.000000  0.000000  0.000000  0.000000
4   0.000000  0.000000  0.234323  0.491753  0.000000  0.000000  0.491753

        the      tiny
0   0.570941  0.000000
1   0.235185  0.493562
2   0.435614  0.000000
3   0.489774  0.000000
4   0.468646  0.000000
```

df = pd.DataFrame({'docs': ["the cat see the mouse",
"the house has a tiny little mouse",
"the mouse ran away from the house",
"the cat finally ate the mouse",
"the end of the mouse story"
]})
print(df)

```
          docs
0              the cat see the mouse
1  the house has a tiny little mouse
2  the mouse ran away from the house
3      the cat finally ate the mouse
4         the end of the mouse story
```

Input: text document

Output: list of tokens,POS tags, lammetization and stemming output , TF,IDF and TF-IDF

Conclusion: In this way we have applied document pre-processing methods for tokenization, Stop word removal, stemming, lemmatization, PoS tagging on input document and created document representation b calculating TF-IDF.

Questions:

1) What is tokenization explain with example
2) Differentiate between stemming and lemmatization
3) Explain TF-IDF with an example
4) What is NLTK?
5) Explain different methods of stopword removal

Assignment No: 8

Data Visualization I

1. Use the inbuilt dataset 'titanic'. The dataset contains 891 rows and contains information about the passengers who boarded the unfortunate Titanic ship. Use the Seaborn library to see if we can find any patterns in the data.
2. Write a code to check how the price of the ticket (column name: 'fare') for each passenger is distributed by plotting a histogram.

Theory:-

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

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

The advantages and benefits of good data visualization:

Our eyes are drawn to colors and patterns. We can quickly identify red from blue, square from circle. Our culture is visual, including everything from art and advertisements to TV and movies. Data visualization is another form of visual art that grabs our interest and keeps our eyes on the message. When we see a chart, we quickly see trends and outliers. If we can see something, we internalize it quickly. It's storytelling with a purpose. If you've ever

stared at a massive spreadsheet of data and couldn't see a trend, you know how much more effective a visualization can be.

Big Data is here and we need to know what it says

As the "age of Big Data" kicks into high-gear, visualization is an increasingly key tool to make sense of the trillions of rows of data generated every day. Data visualization helps to tell stories by curating data into a form easier to understand, highlighting the trends and outliers. A good visualization tells a story, removing the noise from data and highlighting the useful information. However, it's not simply as easy as just dressing up a graph to make it look better or slapping on the "info" part of an infographic. Effective data visualization is a delicate balancing act between form and function. The plainest graph could be too boring to catch any notice or it make tell a powerful point; the most stunning visualization could utterly fail at conveying the right message or it could speak volumes. The data and the visuals need to work together, and there's an art to combining great analysis with great storytelling.

Algorithm:

Step 1: Download the data set of Titanic

(https://www.kaggle.com/uciml/iris)

Step 2: Importing Libraries

```
import pandas as pd
import numpy as np

import matplotlib.pyplot as plt
import seaborn as sns

dataset = sns.load_dataset('titanic')

dataset.head()
```

Step 3: Draw distributional plot

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

Step 4: Removal of Kernal Density line

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

Step 4: Draw histogram

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

Step 5: The Joint Plot

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

Step 6: sns.jointplot(x='age', y='fare', data=dataset, kind='hex')

Step 7: #The Pair Plot

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

Conclusion: Implemented successfully Simple Data visualization techniques using Python on Titanic dataset.

Assignment No: 9

Aim/Problem Statement:- 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')

Write observations on the inference from the above statistics.

Theory:-

Why data visualization is important

It's hard to think of a professional industry that doesn't benefit from making data more understandable. Every STEM field benefits from understanding data—and so do fields in government, finance, marketing, history, consumer goods, service industries, education, sports, and so on. While we'll always wax poetically about data visualization (you're on the Tableau website, after all) there are practical, real-life applications that are undeniable. And, since visualization is so prolific, it's also one of the most useful professional skills to develop. The better you can convey your points visually, whether in a dashboard or a slide deck, the better you can leverage that information. The concept of the citizen data scientist is on the rise. Skill sets are changing to accommodate a data-driven world. It is increasingly valuable for professionals to be able to use data to make decisions and use visuals to tell stories of when data informs the who, what, when, where, and how. While traditional education typically draws a distinct line between creative storytelling and technical analysis, the modern professional world also values those who can cross between the two: data visualization sits right in the middle of analysis and visual storytelling.

The different types of visualizations

When you think of data visualization, your first thought probably immediately goes to simple bar graphs or pie charts. While these may be an integral part of visualizing data and a common baseline for many data graphics, the right visualization must be paired with the right set of information. Simple graphs are only the tip of the iceberg. There's a whole selection of visualization methods to present data in effective and interesting ways. **Common general types of data visualization:**

- Charts
- Tables
- Graphs
- Maps
- Infographics

- Dashboards

More specific examples of methods to visualize data:

- Area Chart
- Bar Chart
- Box-and-whisker Plots
- Bubble Cloud
- Bullet Graph
- Cartogram
- Circle View
- Dot Distribution Map
- Gantt Chart
- Heat Map
- Highlight Table
- Histogram
- Matrix
- Network
- Polar Area
- Radial Tree
- Scatter Plot (2D or 3D)

Algorithm:-

Step 1: Download the data set of Titanic

Step 2: Importing Libraries

```
import math #library for mathematical calculation
import numpy as np #library for scientific computing
import pandas as pd #library for easy-to-use data structures and data analysis tools
import matplotlib.pyplot as plt #library for plotting
import seaborn as sns #library for plotting
```

Step 3: Reading of dataset

```
df = pd.read_csv('train.csv')
```

Step 4: Print details of the dataset

```
print('_'*50)
print('*'*50)
print(df.info(memory_usage=False))
print('_'*50)
```

Step 5: Finding Null values

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

Step 6: Dealing with missing values

```
#drop passengerID, Name, Cabin columns
df = df.drop(['PassengerId', 'Name', 'Cabin'], axis=1)

#drop rows which has nan value
df = df.dropna(axis=0, how='any')
```

Step 7: map 0 to not survived and 1 to survived in survived column

```
df['Survived'] = df['Survived'].map({0: 'Not Survived', 1: 'Survived'})
```

Step 8: group passengers into bins of children, teenager, adult, senior citizen…

```
df['Age'] = pd.cut(df['Age'], bins=[-1, 5, 19, 60, 150], labels=['Children','Teenager','Adult','Senior Citizen'],right=True)
```

Step 9: Set the plot

```
sns.set(context='notebook',style='whitegrid',font_scale=1.5)
```

Step 10: The two methods below is for displaying the count value on top of the bar graph

```
def assignAxis(df,g):
    # Get current axis on current figure
    for i in range(0,g.axes.size):
        ax = g.fig.get_axes()[i]
        displayCount(df,ax)

def displayCount(df,ax):
    # ylim max value to be set
    y_max = df.value_counts().max() + 75
    ax.set_ylim(top=y_max)

    # Iterate through the list of axes' patches
    for p in ax.patches:
        #checks if there index has 0 count
        if(math.isnan(p.get_height())):
            ax.text(p.get_x() + p.get_width()/2, 0, 0,
                fontsize=12, color='red', ha='center', va='bottom')
            continue
        else:
            ax.text(p.get_x() + p.get_width()/2, p.get_height(), int(p.get_height()),
```

```
                fontsize=12, color='red', ha='center', va='bottom')

        #number of males and females
        g = sns.factorplot(x='Sex', data=df,kind='count',size=4, aspect=.8,alpha=0.7,
                palette='muted').set(xlabel='Gender',ylabel='Count',title='Number of Male and Female')

        assignAxis(df['Sex'],g)
```

Step 11: #number of males and females based on Age

```
        g = sns.factorplot(x='Age', data=df,kind='count',size=4, aspect=1.8, hue='Sex',alpha=0.7,
                palette='muted').set(xlabel='Age',ylabel='Count',title='Gender Distribution by Age')

        assignAxis(df['Age'],g)
```

Step 12: People Survived based on gender and age

```
        g = sns.factorplot(x='Survived', data=df,kind='count',size=4, aspect=1.2,hue='Age',col='Sex',alpha=0.7,
                palette='muted',order=['Survived','Not
        Survived'],legend_out=True).set(xlabel='Survival',ylabel='Count')
        g.fig.subplots_adjust(wspace=.3)

        assignAxis(df['Survived'],g)
```

Conclusion: Implemented successfully Simple Data visualization techniques using Python on Titanic dataset.

Assignment No: 10

Aim/Problem Statement:- Data Visualization III

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

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

Theory:-

Data Visualization: Data visualization is the practice of translating information into a visual context, such as a map or graph, to make data easier for the human brain to understand and pull insights from. The main goal of data visualization is to make it easier to identify patterns, trends and outliers in large data sets. The term is often used interchangeably with others, including information graphics, information visualization and statistical graphics.

Data visualization is one of the steps of the data science process, which states that after data has been collected, processed and modeled, it must be visualized for conclusions to be made. Data visualization is also an element of the broader data presentation architecture (DPA) discipline, which aims to identify, locate, manipulate, format and deliver data in the most efficient way possible.

Examples of data visualization

In the early days of visualization, the most common visualization technique was using a Microsoft Excel spreadsheet to transform the information into a table, bar graph or pie chart. While these visualization methods are still commonly used, more intricate techniques are now available, including the following:

- infographics
- bubble clouds
- bullet graphs
- heat maps
- fever charts
- time series charts

Some other popular techniques are as follows.

Line charts: This is one of the most basic and common techniques used. Line charts display how variables can change over time.

Area charts: This visualization method is a variation of a line chart; it displays multiple values in a time series -- or a sequence of data collected at consecutive, equally spaced points in time.

Scatter plots: This technique displays the relationship between two variables. A scatter plot takes the form of an x- and y-axis with dots to represent data points.

Treemaps: This method shows hierarchical data in a nested format. The size of the rectangles used for each category is proportional to its percentage of the whole. Treemaps are best used when multiple categories are present, and the goal is to compare different parts of a whole.

Population pyramids: This technique uses a stacked bar graph to display the complex social narrative of a population. It is best used when trying to display the distribution of a population.

Common data visualization use cases

Common use cases for data visualization include the following:

Sales and marketing: Research from the media agency Magna predicts that half of all global advertising dollars will be spent online by 2020. As a result, marketing teams must pay close attention to their sources of web traffic and how their web properties generate revenue. Data visualization makes it easy to see traffic trends over time as a result of marketing efforts.
Politics: A common use of data visualization in politics is a geographic map that displays the party each state or district voted for.
Healthcare: Healthcare professionals frequently use choropleth maps to visualize important health data. A choropleth map displays divided geographical areas or regions that are assigned a certain color in relation to a numeric variable. Choropleth maps allow professionals to see how a variable, such as the mortality rate of heart disease, changes across specific territories.
Scientists: Scientific visualization, sometimes referred to in shorthand as SciVis, allows scientists and researchers to gain greater insight from their experimental data than ever before.
Finance: Finance professionals must track the performance of their investment decisions when choosing to buy or sell an asset. Candlestick charts are used as trading tools and help finance professionals analyze price movements over time, displaying important information, such as securities, derivatives, currencies, stocks, bonds and commodities. By analyzing how the price has changed over time, data analysts and finance professionals can detect trends.
Logistics. Shipping companies can use visualization tools to determine the best global shipping routes.

Algorithm:-

Step 1: Download the data set of Iris Flower
Step 2: Importing Libraries

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

Step 3: Reading the dataset

```
df = pd.read_csv("iris-flower-dataset.csv")
df
```

Step 4: Perform statistical analysis on data

```
df.mean()
df.median()
df.std()
df.min()
df.max()
df.describe()
```

Step 5: Print number of features and their datatypes

```
column = len(list(df))
column
df.info()
np.unique(df["species"])
```

Step 6: Data Visualization-Create a histogram for each feature in the dataset to illustrate the feature distributions. Plot each histogram.

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

```
%matplotlib inline

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

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

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

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

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

Step 7: Create a boxplot for each feature in the dataset. All of the boxplots should be combined into a single plot. Compare distributions and identify outliers.

```
data_to_plot = [df["sepal_length"],df["sepal_width"],df["petal_length"],df["petal_width"]]

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

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

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

Conclusion: Implemented successfully Simple Data visualization techniques using Python on iris flower dataset.