Sinhgad Technical Education Society's

SMT. KASHIBAI NAVALE COLLEGE OF ENGINEERING, PUNE

Department of Computer Engineering

LABORATORY MANUAL

(2019 Course)

Data Science and Big Data Analytics Lab

TE Computer Engineering

SEMESTER-II

Subject In-charge : Prof. Rahul Dagade

Sinhgad Technical Education Society's
SMT. KASHIBAI NAVALE COLLEGE OF ENGINEERING
Vadgaon (Bk), Pune - 411041
Sinhgad Institutes

Department of Computer Engineering
Third Year of Computer Engineering (2019 Course)
310256: Data Science and Big Data Analytics Laboratory

Teaching Scheme Practical: 04 Hours/Week	Credit Scheme: 02	Examination Scheme and Marks Term work: 50 Marks Practical: 25 Marks
Companion Course: Data Science and Big Data Analytics (310251)		

Assignment List

Sr. No.	Group A : Data Science
1.	**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.
2.	**Data Wrangling II :** Perform the following operations using Python on any open source dataset (eg. data.csv)

	a. 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. b. Scan all numeric variables for outliers. If there are outliers, use any of the suitable techniques to deal with them. c. Apply data transformations on at least one of the variables. The purpose of this transformation should be one of the following reasons: to change the scale for better understanding of the variable, to convert a non-linear relation into a linear one, or to decrease the skewness and convert the distribution into a normal distribution. Reason and document your approach properly
3.	**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.
4.	**Data Analytics I :** Create a Linear Regression Model using Python/R to predict home prices using Boston Housing Dataset (https://www.kaggle.com/c/boston-housing). The Boston Housing dataset contains information about various houses in Boston through different parameters. There are 506 samples and 14 feature variables in this dataset. The objective is to predict the value of prices of the house using the given features.
5.	**Data Analytics II :** 1. Implement logistic regression using Python/R to perform classification on Social_Network_Ads.csv dataset Compute Confusion matrix to find TP, FP, TN, FN, Accuracy, Error rate, Precision, Recall on the given dataset.
6.	**Data Analytics III :** 1. Implement Simple Naïve Bayes classification algorithm using Python/R on iris.csv dataset. II. Compute Confusion matrix to find TP, FP, TN, FN, Accuracy, Error rate, Precision, Recall on the given dataset.
7.	**Text Analytics :** 1. Extract Sample document and apply following document preprocessing methods: Tokenization, POS Tagging, stop words removal, Stemming and

	Lemmatization. Create representation of document by calculating Term Frequency and Inverse Document Frequency.
8.	**Data Visualization I :** 1. Use the inbuilt dataset 'titanic'. The dataset contains 891 rows and contains information about the passengers who boarded the unfortunate Titanic ship. Use the Seaborn library to see if we can find any patterns in the data. 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') Write observations on the inference from the above statistics
10.	**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. 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. 3. Create a boxplot for each feature in the dataset. Compare distributions and identify outliers.
	Group B- Big Data Analytics – JAVA/SCALA (Any three)
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.	Locate dataset (e.g. sample_weather.txt) for working on weather data which reads the text input files and finds average for temperature, dew point and wind speed.
13.	Write a simple program in SCALA using Apache Spark framework
	Group C- Mini Projects/ Case Study – PYTHON/R (Any TWO Mini Project)
14.	Write a case study on Global Innovation Network and Analysis (GINA). Components of analytic plan are 1. Discovery business problem framed, 2. Data, 3. Model planning analytic technique and 4. Results and Key findings.
	OR

14.	Write a case study to process data driven for Digital Marketing OR Health care systems with Hadoop Ecosystem components as shown. (Mandatory) ● HDFS: Hadoop Distributed File System ● YARN: Yet Another Resource Negotiator ● MapReduce: Programming based Data Processing ● Spark: In-Memory data processing ● PIG, HIVE: Query based processing of data services ● HBase: NoSQL Database (Provides real-time reads and writes) ● Mahout, Spark MLLib: (Provides analytical tools) Machine Learning algorithm libraries ● Solar, Lucene: Searching and Indexing
	Any One
15.	Use the following dataset and classify tweets into positive and negative tweets. https://www.kaggle.com/ruchi798/data-science-tweets
15.	Develop a movie recommendation model using the scikit-learn library in python. Refer dataset https://github.com/rashida048/Some-NLP-Projects/blob/master/movie_dataset.csv
15.	Use the following covid_vaccine_statewise.csv dataset and perform following analytics on the given dataset https://www.kaggle.com/sudalairajkumar/covid19-in-india?select=covid_vaccine_statewise.csv a. Describe the dataset b. Number of persons state wise vaccinated for first dose in India c. Number of persons state wise vaccinated for second dose in India d. Number of Males vaccinated e. Number of females vaccinated

Faculty Incharge

1. **Prof. Rahul Dagade**
2. **Prof. Pragati Deole**
3. **Prof. Sarika Aundhakar**
4. **Prof. Varsha Nale**
5. **Prof. Mayuri Agrawal**
6. **Prof. Sheetal Kapse**
7. **Prof. Sandeep Hire**
8. **Prof. Ganesh Jadhav**

Prof. Ravindra H. Borhade
Head of Department
Computer Engineering

Assignment Number - 1	
Title & Problem Statement	**Data Wrangling I :** Perform the following operations using Python on any open source dataset (e.g. data.csv) 1. Import all the required Python Libraries. 2. Locate an open source data from the web (e.g. https://www.kaggle.com). Provide a clear description of the data and its source (i.e. URL of the web site). 3. Load the Dataset into pandas 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.
Objectives	1. Understand the basic concepts of Data Preparation and Cleaning 2. Display basic information of the dataset by accessing its various attributes and methods in python. 3. To prepare and transform data using normalization and scaling techniques to perform EDA process.
Outcomes	Students will be able to: 1. Install and import various libraries required for performing data cleaning and transformation. 2. Perform data cleaning by treating missing values. 3. Convert categorical columns into numerical form and perform various normalization and scaling techniques.
S/W Requirement	OS – Linux Ubuntu 18 (64 bit) Packages - Sublime text editor, Python 3, Seaborn, Matplotlib, Pandas, Numpy, Sklearn etc.
Theory	
• **How to Import modules in Python?**	

- ✓ We can import the definitions inside a module to another module or the interactive interpreter in Python.
- ✓ We use the import keyword to do this. To import our previously defined module example, we type the following in the Python prompt.

✓ **import example**

- ✓ This does not import the names of the functions defined in example directly in the current symbol table. It only imports the module name example there.
- ✓ e.g. `import numpy as np`
- ✓ Here **as** is alias and used to give any short name to the library. In above example we can use np instead of numpy whenever you want to perform the ndarray operations. We will use some global conventions to have compatibility.
- ✓ e.g. import pandas as pd

  `import matplotlib.pyplot as plt`

  Here pyplot is a interface used to access the matplotlib functions and whenever we want to draw plot we will use plt notations for matplotlib.
- ✓ **Another way to import libraries** is

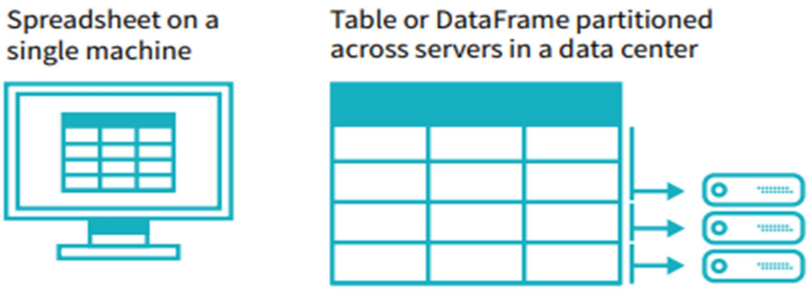
  `from sklearn.preprocessing import MinMaxScaler`

  Here sklearn is a package preprocessing is a module and from this module we are accessing MinMax Scaler function.

  `from scipy import stats`

  This is also an important library for scientific python. Here we are importing the stats module.

- **What is a DataFrame?**



- ✓ A DataFrame is a data structure that organizes data into a 2-dimensional table of

rows and columns, much like a spreadsheet. DataFrames are one of the most common data structures used in modern data analytics because they are a flexible and intuitive way of storing and working with data.

- ✓ Every DataFrame contains a blueprint, known as a schema, that defines the name and data type of each column. Spark DataFrames can contain universal data types like StringType and IntegerType, as well as data types that are specific to Spark, such as StructType. Missing or incomplete values are stored as null values in the DataFrame.
- ✓ A simple analogy is that a DataFrame is like a spreadsheet with named columns. However, the difference between them is that while a spreadsheet sits on one computer in one specific location, a DataFrame can span thousands of computers. In this way, DataFrames make it possible to do analytics on big data, using distributed computing clusters.
- ✓ The reason for putting the data on more than one computer should be intuitive: either the data is too large to fit on one machine or it would simply take too long to perform that computation on one machine.

✓ **Reading Data from CSV file-**

In CSV files, the values are comma-separated. It can be thought of as a text file that holds tabular data in the form of plain text. To access the data from a CSV file, we can use the pandas module. The following example shows the way to extract data from the below CSV.

✓ **Example of reading data from CSV file:**

```
import pandas as pd    #importing the pandas module
#reading the csv file into a DataFrame 'R' represent read
mode
df = pd.read_csv(r'std.csv')
# displaying the DataFrame
print(df)
```

✓ **Other files readers-**

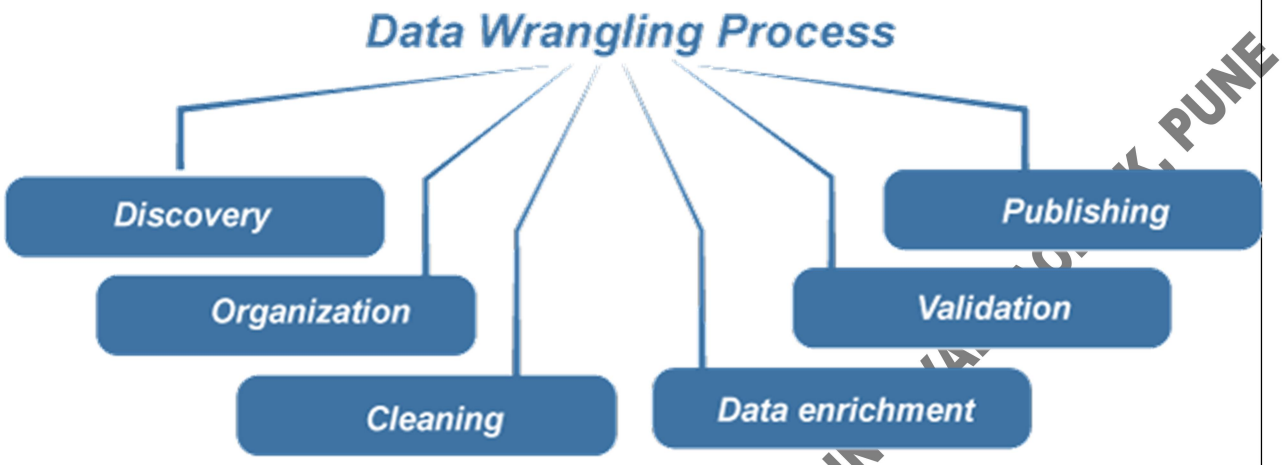
```
df = pd.read_excel("filename.xlsx")
df = pd.read_json("filename.json")
df = pd.read_sql("filename.db")
df = pd.read_xml("filename.xml")
```

✓ **What is data Wrangling?**

Data Wrangling is the process of cleaning and unifying messy and complex data sets to make them more appropriate and valuable for a variety of purposes such as analytics.

✓ **What is data Wrangling?**

Data Wrangling is the process of cleaning and unifying messy and complex data sets to make them more appropriate and valuable for a variety of purposes such as analytics.



1. **Discovery**: Before starting the wrangling process, it is critical to think about what may lie beneath your data. It is crucial to think critically about what results from you anticipate from your data and what you will use it for once the wrangling process is complete. Once you've determined your objectives, you can gather your data.
2. **Organization**: After you've gathered your raw data within a particular dataset, you must structure your data. Due to the variety and complexity of data types and sources, raw data is often overwhelming at first glance.
3. **Cleaning:** When your data is organized, you can begin cleaning your data. Data cleaning involves removing outliers, formatting nulls, and eliminating duplicate data. It is important to note that cleaning data collected from web scraping methods might be more tedious than cleaning data collected from a database. Essentially, web data can be highly unstructured and require more time than structured data from a database.
4. **Data enrichment**: This step requires that you take a step back from your data to determine if you have enough data to proceed. Finishing the wrangling process without enough data may compromise insights gathered from further analysis. For example, investors looking to analyze product review data will want a significant amount of data to portray the market and increase investment intelligence.
5. **Validation**: After determining you gathered enough data, you will need to apply validation rules to your data. Validation rules, performed in repetitive sequences, confirm that your data is consistent throughout your dataset. Validation rules will also ensure quality as well as security. This step follows similar logic utilized in data normalization, a data standardization process involving validation rules.

6. **Publishing:** The final step of the data munging process is data publishing. Data publishing involves preparing the data for future use. This may include providing notes and documentation of your wrangling process and creating access for other users and applications.

✓ **Dealing with missing values:**

Most of the dataset having a vast amount of data contain missing values of NaN, they are needed to be taken care of by replacing them with mean, median for continuous columns and mode value for categorical columns.

e.g. `df.isna().sum())`

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

✓ **To fill the missing values one may use the following code-**

```
df[i].fillna(df[i].mode()[0],inplace=True)
df['col'].replace(['numpy.nan'],df['col'].mean(),inplace=Tru)
print(df.interpolate())
```

If missing values are very few and the dataset is having the significant size one may drop the columns by using following code.

```
df2=df['col'].dropna(axis=0)
```

✓ **Features** – A feature is a numeric representation of raw data.

✓ **Feature Engineering** – Feature engineering is the process of formulating the most appropriate features given the data, the model and the task.

Gender	F	F	M	M	M	F
Marks	65	55	75	70	50	60

Here the gender field is not numeric that is where feature engineering would come in where you would understand how you could convert this raw data set into something more meaningful and computationally more appropriate. For example, you could assign a value of '0' for M and a value of '1' for F. Thus you would convert the categorical column into numeric form.

Below are some of the common code snnipets used for conversion -

1. `df=pd.get_dummies(df,columns=['gender'],prefix='sex').head(100)`
   `uses internally one hot encoding.`
2. `df['Gender'] = df['Gender'].map({'M':0,'F':1})`
3. `df['Gender'] = df['Gender'].cat.codes` uses internally label encoding.

- **Data Normalization –**
  - ✓ Normalization is no mandate for all datasets available in machine learning.

- ✓ It is used whenever the attributes of the dataset have different ranges.
- ✓ Data Normalization is the organization of data to appear similar across all records and fields.
- ✓ When data normalization is done correctly, you will end up with standardized information entry.
- ✓ It helps to enhance the performance and reliability of ML model.
- ✓ Normalization is a scaling technique in Machine Learning applied during data preparation to change the values of numeric columns in the dataset to use a common scale.
- ✓ It is required only when features of machine learning models have different ranges.

- **Different techniques include-**

1. **Min-max method**.

   Min-Max scaling method helps the dataset to shift and rescale the values of their attributes, so they end up ranging between 0 and 1.

2. **Standardization scaling**:

   It is also known as **Z-score** normalization, in which values are centered around the mean with a unit standard deviation.

   Mathematically, we can calculate the standardization by subtracting the feature value from the mean and dividing it by standard deviation.

   - Hence, standardization can be expressed as follows:

$$X' = \frac{X - \mu}{\sigma}$$

   - Here, **μ** represents the mean of feature value, and **σ** represents the standard deviation of feature values.
   - However, unlike Min-Max scaling technique, feature values are not restricted to a specific range in the standardization technique.
   - This technique is helpful for various machine learning algorithms that use distance measures such as **KNN, K-means clustering, and Principal component analysis**, etc.

```
from sklearn.preprocessing import MaxAbsScaler
abs_scaler=MaxAbsScaler()
df['salary']=MaxAbsScaler().fit(df['salary'])
print('\n Maximum absolute Scaling method normalization
-1 to 1 \n\n')
print(df['salary'])

from sklearn.preprocessing import MinMaxScaler
```

```
scaler=MinMaxScaler()
df['salary']=MinMaxScaler().fit_transform(df['salary'])
print('\n MinMax feature Scaling method normalization 0
to 1 \n\n')
print(df['salary'])

from sklearn.preprocessing import StandardScaler
df['salary']=(df['salary']-
df['salary'].mean())/(df['salary'].std())
print('\n z score is \n\n')
print(df['salary'])

from sklearn.preprocessing import RobustScaler
df['salary']=(df['salary']-
df['salary'].mean())/(df['salary'].quantile(0.75)-
df['salary'].quantile(0.25))
print('\n Robust Scaling \n\n')
print(df['salary'])
```

Algorithm	1. Load the Student Dataset into Pandas Dataframe 2. Display different attributes such as columns, size, dtypes and use methods like describe(), head(), tail(), sample() etc. 3. Treat the missing values by replacing them with appropriate values or by dropping the observation. 4. Perform data normalization and scaling operations so that all columns will have equal importance and scaled down to common scale. Perform the all above operations on student dataset and student information dictionary declared in the program itself.
Conclusion	In this assignment we are able to: 1. Access all related information about the dataframe. 2. Clean and transform data from raw form to usable form using Pandas dataframe. 3. Treat the missing values and perform different normalization and scaling operations.
Questions	1. How to import a library in python? 2. What are the different types of files can be read in dataframe using pandas reader function? 3. What kind of information is displayed by describe() function? 4. What is the difference between size and info function ? 5. How the missing values are treated in pandas and what are its functions.

	6. Which are the libraries used? 7. What is the need of converting categorical values to numerical form 8. What is data Normalization? What are the different normalization functions performed. 9. What is scaling? 10. What is data wrangling? Is there any difference between data preparation, data wrangling and data munging? 11. How to read csv, excel and text file in python?

Assignment Number - 2	
Title & Problem Statement	Create an "Academic performance" dataset of students and perform the following operations using Python. 1. Scan all variables for missing values and inconsistencies. If there are missing values and/or inconsistencies, use any of the suitable techniques to deal with them. 2. Scan all numeric variables for outliers. If there are outliers, use any of the suitable techniques to deal with them. 3. Apply data transformations on at least one of the variables. The purpose of this transformation should be one of the following reasons: to change the scale for better understanding of the variable, to convert a non-linear relation into a linear one, or to decrease the skewness and convert the distribution into a normal distribution.
Objectives	Write a python program to load the dataset and understand the input data Dataset: Pima Indians Diabetes Dataset https://www.kaggle.com/uciml/pima-indians-diabetes-database#diabetes.csv Library: Scipy 1) Load data, describe the given data and identify missing, outlier data items. 2) Replace with mean or mode. 3) Find correlation among all attributes. 4) Perform transformation of data using Discretization (Binning) and normalization (MinMaxScaler or MaxAbsScaler) on given dataset.
Outcomes	Students will be able to: 1. Scan all variables for missing values and inconsistencies. If there are missing values and/or inconsistencies, use any of the suitable techniques to Display information in tabular format. 2. Scan all numeric variables for outliers. If there are outliers, use any of the suitable techniques to Display the outlier. 3. Convert a non-linear relation into a linear one 4. decrease the skewness and convert the distribution into a normal distribution
S/W Requirement	OS – Linux Ubuntu 18 (64 bit) Packages - Sublime text editor, Python 3, Seaborn, Matplotlib, Pandas, SciPy
Theory	
Working with Missing Data in Pandas ✓ In order to check missing values in Pandas DataFrame, we use a function isnull() and notnull().	

- ✓ Both function help in checking whether a value is NaN or not.
- ✓ These functions can also be used in Pandas Series in order to find null values in a series.

Checking for missing values using isnull()

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

  Syntax: df.isnull()

Checking for missing values using notnull()

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

  Syntax: df.notnull()

Filling missing values using fillna(), replace() and interpolate()

- ✓ In order to fill null values in a datasets, we use fillna(), replace() and interpolate() function these function replace NaN values with some value of their own.
- ✓ All these function help in filling a null values in datasets of a DataFrame.
- ✓ Interpolate() function is basically used to fill NA values in the dataframe but it uses various interpolation technique to fill the missing values rather than hard-coding the value.

Filling null values with a single value

Syntax: df.fillna(0)

Filling null values with the previous ones

Syntax: df.fillna(method ='pad')

Filling null value with the next ones

Syntax: df.fillna(method ='bfill')

Missing values In Pandas missing data is represented by two values:

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

  isnull()
  notnull()
  dropna()
  fillna()
  replace()
  interpolate()

 # identify missing items
print(df.isnull())

#outlier data items

1) **Z-score method:** Z score is an important concept in statistics. Z score is also called standard score. This score helps to understand if a data value is greater or smaller than mean and how far away it is from the mean. More specifically, Z score tells how many standard deviations away a data point is from the mean.

   Z score = (x -mean) / std. deviation

2) **Modified Z-score method:** The modified z score is a standardized score that measures outlier strength or how much a particular score differs from the typical score. Using standard deviation units, it approximates the difference of the score from the median.

   The modified z score might be more robust than the standard z score because it relies on the median for calculating the z score. It is less influenced by outliers when compared to the standard z score.

3) **IQR method:** IQR is used to measure variability by dividing a data set into quartiles. The data is sorted in ascending order and split into 4 equal parts. Q1, Q2, Q3 called first, second and third quartiles are the values which separate the 4 equal parts.
   Q1 represents the 25th percentile of the data.
   Q2 represents the 50th percentile of the data.
   Q3 represents the 75th percentile of the data.

#Z-score function defined in scipy library to detect the outliers

```
import numpy as np
Def outliers_z_score(ys):
          threshold = 3 mean_y = np.mean(ys)
          stdev_y = np.std(ys)
          z_scores = [(y - mean_y) / stdev_y for y in ys]
          return np.where(np.abs(z_scores) > threshold)
```

b) Find correlation among all attributes

 # importing pandas as pd

```
import pandas as pd
```

 # Making data frame from the csv file

```
df = pd.read_csv("nba.csv")
```

 # Printing the first 10 rows of the data frame for visualization

```
df[:10]
```

 # To find the correlation among columns # using pearson method

```
df.corr(method ='pearson')
```

using 'kendall' method.

```
df.corr(method ='kendall')
```

PROGRAM LOGIC:

 # filling missing value using fillna()

df.fillna(0)

 # filling a missing value with

```
previous value df.fillna(method ='pad')
```

#Filling null value with the next ones

```
df.fillna(method ='bfill')
```

filling a null values using fillna()

```
data["Gender"].fillna("No Gender", inplace = True)
```

will replace **Nan** value in dataframe with value -99

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

 # Remove rows/ attributes

using dropna() function to remove rows having one **Nan**

```
df.dropna()
```

using dropna() function to remove rows with all **Nan**

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

using dropna() function to remove column having one **Nan**

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

 # Replace with mean or mode

```
mean_y = np.mean(ys)
```

Algorithm	1. Load the Academic performance Dataset into Pandas Dataframe 2. Load data, describe the given data and identify missing, outlier data items. 3. Replace with mean or mode. 4. Find correlation among all attributes. Perform transformation of data using Discretization (Binning) and normalization (MinMaxScaler or MaxAbsScaler) on given dataset.
Conclusion	In this assignment we are able to: 1. Missing values from the dataset using Pandas. 2. Replace the missing values using mean and mode. 3. Perform transformation of data using Discretization and normalization
Questions	1. How to find missing values from dataset?

	2. What is difference between isnull() and notnull()? 3. What is IQR? 4. What is mean by outlier? 5. What are the different methods to find the outlier? 6. Which are the libraries used? 7. What is discretization? 8. What is Normalization? 9. What is binning method? 10. Explain Z score method? 11. What is functioning of interpolate()? 12. What is linear & non-linear relation?

Assignment Number - 3	
Title & 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.
Objectives	1. Understand the basic concepts of statistics 2. Display statistical information of the Employee Salary dataset and Iris flower dataset 3. Perform visualization of the results
Outcomes	Students will be able to: 1. Calculate the central tendencies and variability of the dataset using Pandas 2. Display statistical information in tabular format 3. Plot visualization of statistical information using Seaborn
S/W Requirement	OS – Linux Ubuntu 18 (64 bit) Packages - Sublime text editor, Python 3, Seaborn, Matplotlib, Pandas

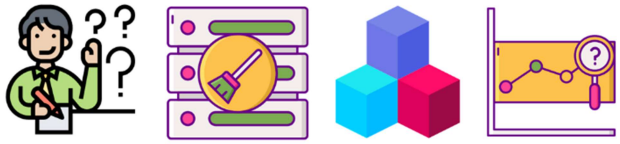
Theory

What is Statistics?

- ✓ Statistics is a branch of mathematics that deals with collecting, analyzing, interpreting, and visualizing empirical data.
- ✓ Descriptive statistics and inferential statistics are the two major areas of statistics.
- ✓ Descriptive statistics are for describing the properties of sample and population data (what has happened).
- ✓ Inferential statistics use those properties to test hypotheses, reach conclusions, and make predictions (what can you expect).

Use of Statistics in Data Science

- ✓ Asking questions about the data
- ✓ Cleaning and preprocessing the data
- ✓ Selecting the right features
- ✓ Model evaluation
- ✓ Model prediction



What are measures of central tendency?

- ✓ Mean, median, and mode are the measures of central tendency, used to study the various characteristics of a given set of data.
- ✓ A measure of central tendency describes a set of data by identifying the central position in the data set as a single value.
- ✓ We can think of it as a tendency of data to cluster around a middle value. In statistics, the three most common measures of central tendencies are Mean, Median, and Mode.
- ✓ Choosing the best measure of central tendency depends on the type of data we have.

Mean

- ✓ The arithmetic mean of a given data is the sum of all observations divided by the number of observations.
- ✓ For example, a cricketer's scores in five ODI matches are as follows: 12, 34, 45, 50, 24. To find his average score in a match, we calculate the arithmetic mean of data using the mean formula:

$$\text{Mean} = \frac{Sum\ of\ terms}{Number\ of\ terms}$$

- ✓ Mean = Sum of all observations/Number of observations

  Mean = (12 + 34 + 45 + 50 + 24)/5

  Mean = 165/5 = 33

  Mean is denoted by $\bar{x}$ (pronounced as x bar).
- ✓ Consider the data frame below that has the names of seven employees and their salaries.

	Name	Salary
0	Jane	50000
1	Michael	54000
2	Willian	50000
3	Rosy	189000
4	Hana	55000
5	Ferdie	40000
6	Graeme	59000

To find the mean or the average salary of the employees, you can use the mean() functions in Python.

```
print(df['Salary'].mean())
```

```
71000.0
```

Mode

- ✓ The Mode refers to the most frequently occurring value in your data.
- ✓ You find the frequency of occurrence of each number and the number with the highest frequency is your mode. If there are no recurring numbers, then there is no mode in the data.
- ✓ Using the mode, you can find the most commonly occurring point in your data. This is helpful when you have to find the central tendency of categorical values, like the flavor of the most popular chip sold by a brand. You cannot find the average based on the orders; instead, you choose the chip flavor with the highest orders.
- ✓ Usually, you can count the most frequently occurring values and get your mean. But this only works when the values are discrete. Now, again take the example of class marks.
- ✓ Example: Take the following marks of students :

Marks = 35, 40, 45, 49, 34, 47, 39, 25, 19, 35, 28, 48

Over here, the value 35 occurs the most frequently and hence is the mode.

- ✓ But what if the values are categorical? In that case, you must use the formula below:

$$\text{Mode} = l + \left(\frac{f1 - f0}{2f1 - f0 - f2}\right) \times h$$

Where,

l = lower limit of modal class
h = lower limit of preceding modal class
f1 = frequency of modal class
f0 = frequency of class preceding modal class
f2 = frequency of class succeeding modal class

The modal class is simply the class with the highest frequency. Consider the range of frequencies given for the marks obtained by students in a class:

Marks	10-20	20-30	30-40	40-50
Number of Students	1	3	5	4

In this case, you can see that class 30-40 has the highest frequency, hence it is the modal class. The remaining values are as follows: l = 30, h = 20, f1 = 5, f0 = 3, f2 = 4
In that case, the mode becomes :

$$\text{Mode} = 30 + \left(\frac{5-3}{2*5-3-4}\right) \times 20$$

$$= 43.33$$

Hence, the mark which occurs most frequently is 43.33. The mode of salary from the salary

data frame can be calculated as:

```
print(df['Salary'].mode())
```

```
0    50000
dtype: int64
```

Median

- ✓ Median refers to the middle value of a data. To find the median, you first sort the data in either ascending or descending order and then find the numerical value present in the middle of your data.
- ✓ It can be used to figure out the point around which the data is centered. It divides the data into two halves and has the same number of data points above and below.
- ✓ The median is especially useful when the data is skewed data. That is, it has high data distribution towards one side. In this case, the average wouldn't give you a fair mid-value but would lean more towards the higher values. In this case, you can use the middle data point as the central point instead.
- ✓ Consider n terms X_1, X_2, X_3,…… X_n. The basic formula for the median is by dividing the total number of observations by 2. This works fine when you have an odd number of terms because you will have one middle term and the same number of terms above and below. For an even number of terms, consider the two middle terms and find their average.

$$\text{Median} = \frac{n+1}{2} \text{ th term , n = odd}$$

$$\{\frac{n}{2} \text{ th term} + \frac{n}{2} + 1 \text{ th term}\} / 2 \text{ , n = even}$$

Example: Consider following are students marks

Marks = 35, 40, 45, 49, 34, 47, 39, 25, 19, 35, 28, 48

To find the middle term, you first have to sort the data or arrange the data in ascending or descending order. This ensures that consecutive terms are next to each other.

Sorted Marks = 19, 25, 28, 30, 34, 35, 39, 40, 45, 47, 48, 49

You can see that we have 12 data points, so use the median formula for even numbers.

$$\text{Median} = \{(\frac{12}{2} \text{ th term}) + (\frac{12}{2} + 1 \text{ th term})\} / 2$$
$$= \{6^{th} + 7^{th}\} /2 = (35 + 39) /2$$
$$= 37$$

So, the middle term in the range of marks is 37. This means that the other marks lie in a frequency range of around 37.

The median() function in Python can help you find the median value of a column. From the salary data frame, you can find the median salary as:

```
print(df['Salary'].median())
```

```
54000.0
```

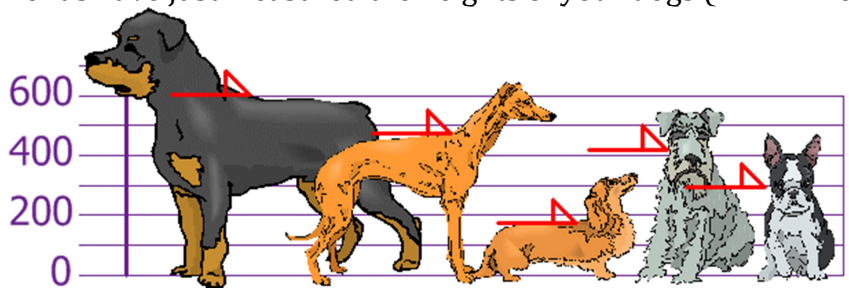
Standard Deviation and Variance

- ✓ Deviation just means how far from the normal
- ✓ The Standard Deviation is a measure of how spread out numbers are.
- ✓ Its symbol is σ (the greek letter sigma)
- ✓ The formula is easy: it is the square root of the Variance.

$$\sigma^2 = \frac{\sum_{i=1}^{n} (x_i - \mu)^2}{n}$$

- ✓ The Variance is defined as: The average of the squared differences from the Mean.
- ✓ To calculate the variance follow these steps:
    1. Work out the Mean (the simple average of the numbers)
    2. Then for each number: subtract the Mean and square the result (the squared difference).
    3. Then work out the average of those squared differences
- ✓ Variance is used to measure the variability in the data from the mean.
- ✓ Example:

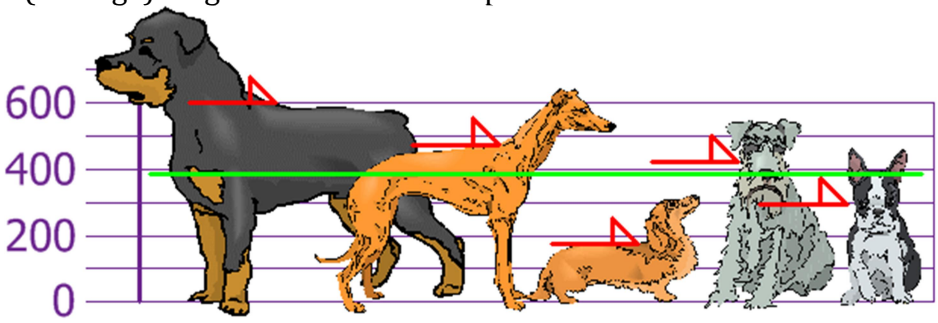
You and your friends have just measured the heights of your dogs (in millimeters):



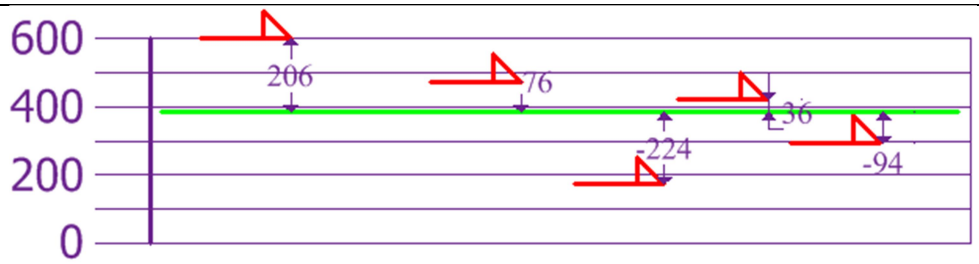
The heights (at the shoulders) are: 600mm, 470mm, 170mm, 430mm and 300mm. Find out the Mean, the Variance, and the Standard Deviation. Your first step is to find the Mean:

Mean = (600 + 470 + 170 + 430 + 300) / 5 = 1970/5 = 394

So the mean (average) height is 394 mm. Let's plot this on the chart:



Now we calculate each dog's difference from the Mean:



To calculate the Variance, take each difference, square it, and then average the result:
Variance =

$$\sigma^2 = \frac{206^2 + 76^2 + (-224)^2 + 36^2 + (-94)^2}{5}$$

$$= \frac{42436 + 5776 + 50176 + 1296 + 8836}{5}$$

$$= \frac{108520}{5}$$

$$= 21704$$

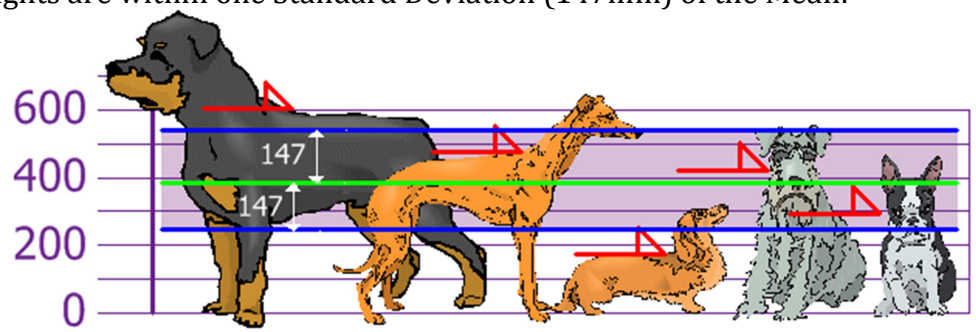
So the Variance is 21,704
And the Standard Deviation is just the square root of Variance, so:
Standard Deviation =

$$\sigma = \sqrt{21704}$$

$$= 147.32...$$

$$= \mathbf{147}\ \textit{(to the nearest mm)}$$

And the good thing about the Standard Deviation is that it is useful. Now we can show which heights are within one Standard Deviation (147mm) of the Mean:



So, using the Standard Deviation we have a "standard" way of knowing what is normal, and what is extra-large or extra small. Rottweiler's are tall dogs. And Dachshunds are a bit short, right?

Algorithm	1. Load the Employee Salary Dataset into Pandas Dataframe 2. Display summary statistics in a tabular format using min(), max(), mean() etc. functions

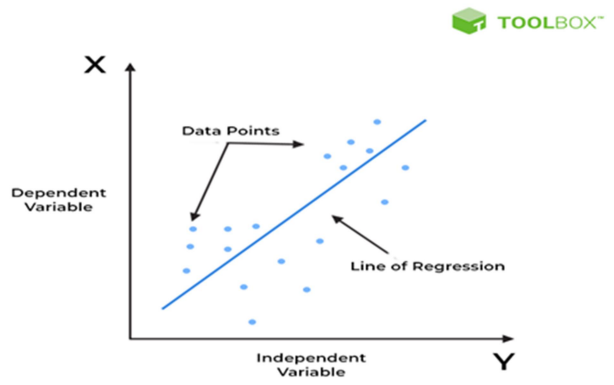
	3. Display Grouped by Gender summary statistics of in a tabular format using min(), max(), mean() etc. functions 4. Visualize the results using Seaborn barplot Perform the same on Iris Flower Dataset
Conclusion	In this assignment we are able to: 1. Calculate central tendency and variability using Pandas 2. Display statistical information in tabular format 3. Visualize statistical information using Seaborn and Matplotlib
Questions	1. What is statistics? 2. What is central tendency in statistics? 3. What are the applications of statistics? 4. What is variability? 5. What are the formulas for mean, median, standard deviation, variance etc. 6. Which are the libraries used? 7. How to display information in tabular format in python? 8. What is barplot? 9. What are the parameters in Seaborn barplot function? 10. Explain Iris flower dataset? 11. Explain Employee salary dataset?

Assignment Number - 4	
Title & Problem Statement	Create a Linear Regression Model using Python/R to predict home prices using Boston Housing Dataset (https://www.kaggle.com/c/boston-housing). The Boston Housing dataset contains information about various houses in Boston through different parameters. There are 506 samples and 14 feature variables in this dataset**.**
Objectives	1. Understand the concepts of linear Regression. 2. To create a model using linear regression to predict the houses price 3. Perform visualization of the results
Outcomes	Students will be able to: 1. Split data to training and testing dataset. 2. Create linear regression model and train data using training data. 3. Test the model using testing data. 4. Display accuracy and mean absolute error
S/W Requirement	OS – Linux Ubuntu 18 (64 bit) Packages - Sublime text editor, Python 3, Seaborn, Matplotlib, Pandas

Theory

What is linear Regression?

- ✓ Linear Regression is the supervised Machine Learning model in which the model finds the best fit linear line between the independent and dependent variable i.e it finds the linear relationship between the dependent and independent variable.
- ✓ The independent variable is also the predictor variable that remains unchanged due to the change in other variables.
- ✓ The dependent variable changes with fluctuations in the independent variable. The regression model predicts the value of the dependent variable, which is the response or outcome variable being analyzed.



In the above figure,

X-axis = Independent variable

Y-axis = Output / dependent variable

Line of regression = Best fit line for a model

- ✓ A line is plotted for the given data points that suitably fit is called the best fit line. The goal of the linear regression algorithm is to find this best fit line.

Types of Linear Regression

- ✓ Linear regression can be further divided into two types of the algorithm:

Simple Linear Regression:

- If a single independent variable is used to predict the value of a numerical dependent variable, then such a Linear Regression algorithm is called Simple Linear Regression.
- **Multiple Linear Regression:**
  If more than one independent variable is used to predict the value of a numerical dependent variable, then such a Linear Regression algorithm is called Multiple Linear Regression.

Linear Regression on Boston Housing Dataset

- ✓ The Housing dataset which contains information about different houses in Boston. This data was originally a part of UCI Machine Learning Repository and has been removed now. We can also access this data from the scikit-learn library. 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.

- ✓ The problem that we are going to solve here is that given a set of features that describe a house in Boston, our machine learning model must predict the house price. To train our machine learning model with boston housing data, we will be using scikit-learn's boston dataset.
- ✓ In this dataset, each row describes a boston town or suburb. There are 506 rows and 14 attributes (features) with a target column

Data description

- ✓ The Boston data frame has 506 rows and 14 columns.

This data frame contains the following columns:

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

The prices of the house indicated by the variable **MEDV** is our *target variable* and the remaining are the *feature variables* based on which we will predict the value of a house. We will now load the data into a pandas dataframe using pd.DataFrame. We then print the first 5 rows of the data using head()

Splitting the data into training and testing sets

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

Algorithm

Step 1: **Importing Libraries**

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

Step 2: Load the Boston housing dataset.

```
import pandas as pd
df = pd.read_csv('https://raw.githubusercontent.com/selva86/
datasets/master/BostonHousing.csv')
print.df()
```

Step 3: Find missing values in the dataset and fill the empty values with median of that column.

```
print.isna().sum()
```

Step 4: Find statistical information of Dataset

```
  print.describe()
```

Step 5: Find outlier in the dataset using boxplot and remove the outlier using IQR method.

```
data = df[['rm', 'lstat', 'medv']]
print(data.head())
sns.boxplot(x=df['rm'])
sns.boxplot(x=df['lstat'])
sns.scatterplot(data=df, x="rm", y="medv")
sns.scatterplot(data=df, x="lstat", y="medv")

def Remove_outlier(df, var):
  Q1=df[var].quantile(0.25)
  Q3=df[var].quantile(0.75)
  IQR=Q3-Q1
  df_final = df[~((df[var]<(Q1-
1.5*IQR)) | (df[var]>(Q3+1.5*IQR)))]
  return df_final
```

Step 6: Find correlation Between input and output Veriable using co-relation matrix and display using heatmap.

```
# co-relation matrix

import seaborn as sns

sns.heatmap(df.corr(), annot = True)
import matplotlib.pyplot as plt
fig, ax = plt.subplots(figsize=(16, 8))
sns.heatmap(df.corr(), annot = True)
```

Step 7: select the variable which are more correlated to output variable.

```
data = df[['rm', 'lstat', 'medv']]
print(data.head())
```

Step 8: Divide the data into training and testing data.

```
X_train, Y_train We will train the model
# X_test, Y_test we will test the model


X = df[['rm', 'lstat']] #Input data (independent data)
Y = df['medv']    # Output data (dependent data)


from sklearn.model_selection import train_test_split
X_train, X_test, Y_train, Y_test = train_test_split(X, Y, te
st_size=0.2, random_state=42)

print(X_train.shape)
print(Y_train.shape)
print(X_test.shape)
print(X_test.shape)
```

Step 9: Create linear Regression model and train it using training data.

```
from sklearn.linear_model import LinearRegression
model = LinearRegression().fit(X_train, Y_train)
output = model.predict(X_test)
print(output)
```

Step 10: Test the model using Testing Data

```
print(Y_test)
def LinearRegressionModel(X_train, Y_train, X_test, Y_test):
  from sklearn.linear_model import LinearRegression
  model = LinearRegression().fit(X_train, Y_train)
  output = model.predict(X_test)
```

Step 11: Display accuracy and mean absolute error

```
from sklearn.metrics import mean_absolute_error
print("MAE: ", mean_absolute_error(Y_test, output))
print("Model Score: ", model.score(X_test, Y_test))
```

Step 12: Give user input to the model and print the house price.

```
X = df[['rm', 'lstat', 'ptratio']] #Input data (independent data)
Y = df['medv']
LinearRegressionModel(X, Y)
```

```
MAE:  3.2675023437349995
Model Score:  0.7206760539489805
```

Algorithm

Step 1: Importing Libraries

Step 2: Load the Boston housing dataset.

Step 3: Find missing values in the dataset and fill the empty values with median of that column.

Step 4: Find statistical information of Dataset
Step 5: Find outlier in the dataset using boxplot and remove the outlier using IQR method.
Step 6: Find correlation Between input and output Veriable using co-relation matrix and display using heatmap.
Step 7: select the variable which are more correlated to output variable.
Step 8: Divide the data into training and testing data.
Step 9: Create linear Regression model and train it using training data.
Step 10: Test the model using Testing Data
Step 11: Display accuracy and mean absolute error
Step 12: Give user input to the model and print the house price.

Conclusion

In this assignment:

1. Prediction of Boston Housing Prices by plotting graph to show prediction
2. Housing Prices of Boston city predicted using Linear Regression model.

Questions

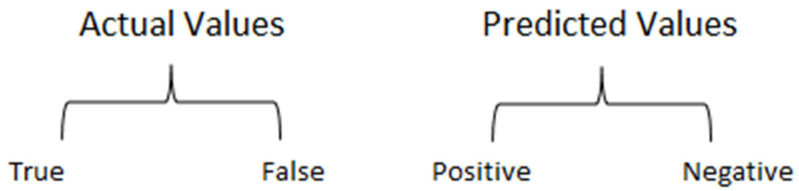
1) What is Linear regression
2) What are different types of linear regressions
3) Applications where linear regression is used
4) What are the limitations of linear regression
5) How to remove outlier
6) What is training and testing Data
7) What is Box plot. Explain 5 summary of Box plot.

Assignment Number - 5	
Title & Problem Statement	**Data Analytics II** ✓ Implement logistic regression using Python 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.
Objectives	1. Implementation of logistic regression 2. Compute Confusion matrix by using Social_Network_Ads.csv dataset 3. Find out find TP, FP, TN, FN, Accuracy, Error rate, Precision, Recall on the given dataset.
Outcomes	**Students will be able to:** 1. Implement logistic regression 2. Compute Confusion matrix 3. Find TP, FP, TN, FN, Accuracy, Error rate, Precision, Recall on the given dataset
S/W Requirement	OS – Linux Ubuntu 18 (64 bit) Packages - Sublime text editor, Python 3, Seaborn, Matplotlib, Pandas

Theory

What is Confusion Matrix and why you need it?

- ✓ Precision, Specificity, Accuracy, and most importantly AUC-ROC curves.



Actual vs Predicted values [Image]

How to Calculate Confusion Matrix for a 2-class classification problem?

Y	Y pred	Output for threshold 0.6	Recall	Precision	Accuracy
0	0.5	0	1/2	2/3	4/7
1	0.9	1			
0	0.7	1			
1	0.7	1			
1	0.3	0			
0	0.4	0			
1	0.5	0			

1. **Recall:**

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

- ✓ The above equation can be explained by saying, from all the positive classes, how many we predicted correctly.
- ✓ Recall should be high as possible.

2. **Precision:**

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

- ✓ The above equation can be explained by saying, from all the classes we have predicted as positive, how many are actually positive.
- ✓ Precision should be high as possible.

3. **Accuracy**

- ✓ From all the classes (positive and negative), how many of them we have predicted correctly.
- ✓ In this case, it will be 4/7.
- ✓ Accuracy should be high as possible.

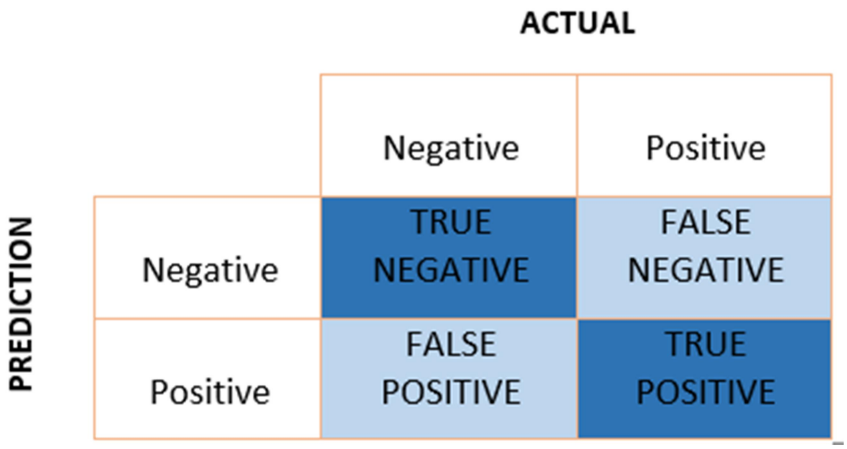
$$F\text{-}measure = \frac{2*Recall*Precision}{Recall + Precision}$$

4. **F-measure:**

- ✓ It is difficult to compare two models with low precision and high recall or vice versa. So to make them comparable, we use F-Score.
- ✓ F-score helps to measure Recall and Precision at the same time. It uses Harmonic Mean in place of Arithmetic Mean by punishing the extreme values more.
- ✓ Accuracy performance metrics can be decisive when dealing with imbalanced data. The confusion matrix, precision, recall, and F1 score gives better intuition of prediction results as compared to accuracy.

What is a confusion matrix?

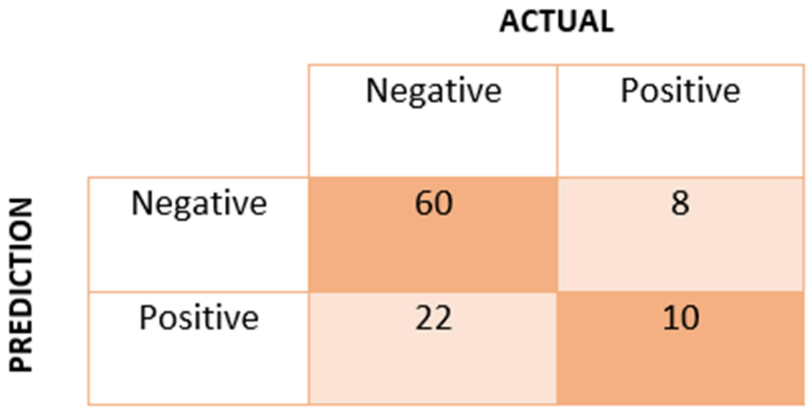
- ✓ It is a matrix of size 2×2 for binary classification with actual values on one axis and predicted on another.



Confusion Matrix

- ✓ The confusing terms in the confusion matrix are: true positive, true negative, false negative and false positive with an example.

EXAMPLE: A machine learning model is trained to predict tumor in patients. The test dataset consists of 100 people.



Confusion Matrix for tumor detection

- ✓ **True Positive (TP)** — model correctly predicts the positive class (prediction and actual both are positive). In the above example, 10 people who have tumors are predicted positively by the model.
- ✓ **True Negative (TN)** — model correctly predicts the negative class (prediction and actual both are negative). In the above example, 60 people who don't have tumors are predicted negatively by the model.
- ✓ **False Positive (FP)** — model gives the wrong prediction of the negative class (predicted-positive, actual-negative). In the above example, 22 people are predicted as positive of having a tumor, although they don't have a tumor. FP is also called a TYPE I. error.
- ✓ **False Negative (FN)** — model wrongly predicts the positive class (predicted-negative, actual-positive). In the above example, 8 people who have tumors are predicted as

negative. FN is also called a TYPE II error. With the help of these four values, we can calculate True Positive Rate (TPR), False Negative Rate (FPR), True Negative Rate (TNR), and False Negative Rate (FNR).

$$TPR = \frac{TP}{Actual\ Positive} = \frac{TP}{TP + FN}$$

$$FNR = \frac{FN}{Actual\ Positive} = \frac{FN}{TP + FN}$$

$$TNR = \frac{TN}{Actual\ Negative} = \frac{TN}{TN + FP}$$

$$FPR = \frac{FP}{Actual\ Negative} = \frac{FP}{TN + FP}$$

- ✓ Even if data is imbalanced, we can figure out that our model is working well or not. For that, the values of TPR and TNR should be high, and FPR and FNR should be as low as possible. With the help of TP, TN, FN, and FP, other performance metrics can be calculated.
- ✓ Precision, Recall: Both precision and recall are crucial for information retrieval, where positive class mattered the most as compared to negative. Because while searching something on the web, the model does not care about something irrelevant and not retrieved (this is the true negative case). Therefore, only TP, FP, FN are used in Precision and Recall.

5. **Precision:** Out of all the positive predicted what percentage is truly positive.

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

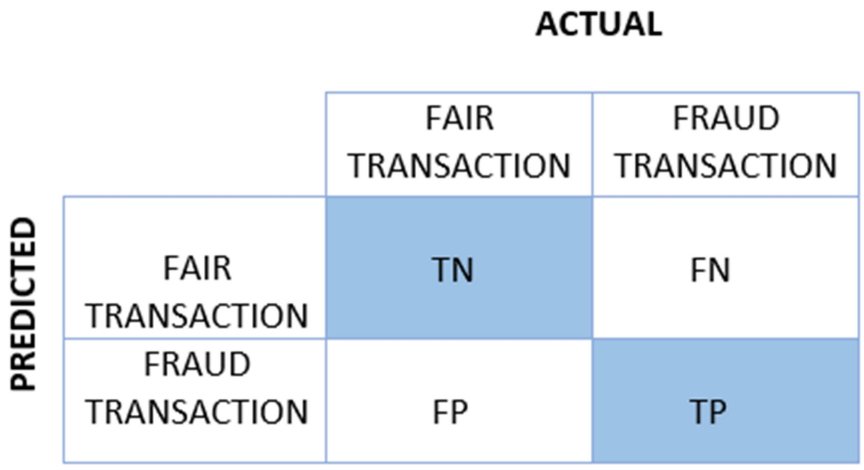
- ✓ The precision value lies between 0 and 1.

6. **Recall**: Out of the total positive, what percentage are predicted positive. It is the same as TPR (true positive rate).

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

How are precision and recall useful?

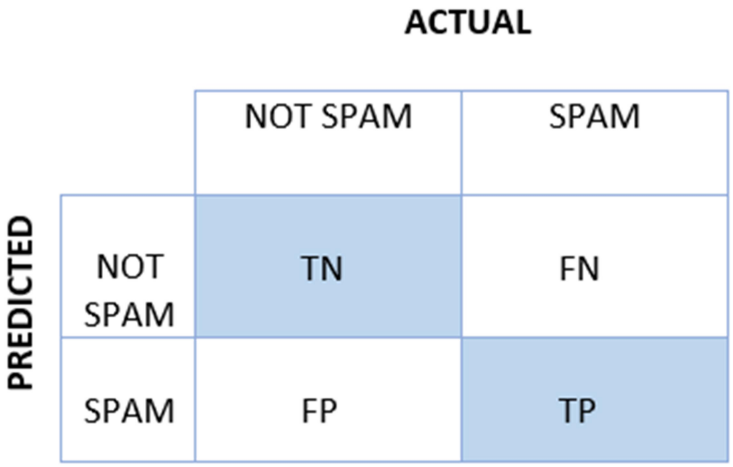
- ✓ EXAMPLE 1- Credit card fraud detection



Confusion Matrix for Credit Card Fraud Detection

- ✓ We do not want to miss any fraud transactions. Therefore, we want False-Negative to be as low as possible. In these situations, we can compromise with the low precision, but recall should be high. Similarly, in the medical application, we do not want to miss any patient. Therefore, we focus on having a high recall. So, we have discussed when the recall is important than precision. But when is the precision more important than recall?

- ✓ EXAMPLE 2 — Spam detection



Confusion Matrix for Spam detection

- ✓ In the detection of spam mail, it is okay if any spam mail remains undetected (false negative), but what if we miss any critical mail because it is classified as spam (false positive). In this situation, False Positive should be as low as possible. Here, precision is more vital as compared to recall.

  When comparing different models, it will be difficult to decide which is better (high precision and low recall or vice-versa). Therefore, there should be a metric that combines both. One such metric is the F1 score.

7. **F1 Score:**

- ✓ It is the harmonic mean of precision and recall. It takes both false positive and false negatives into account. Therefore, it performs well on an imbalanced dataset.

$$F1\ score = \frac{2}{\frac{1}{Precision} + \frac{1}{Recall}} = \frac{2 * (Precision * Recall)}{(Precision + Recall)}$$

- ✓ score gives the same weightage to recall and precision.
- ✓ There is a weighted F1 score in which we can give different weightage to recall and precision. As discussed in the previous section, different problems give different weightage to recall and precision.

$$F_\beta = (1 + \beta^2) * \frac{(Precision * Recall)}{(\beta^2 * Precision) + Recall}$$

- ✓ Beta represents how many times recall is more important than precision. If the recall is twice as important as precision, the value of Beta is 2.

Confusion Matrix – An Overview with Python and R

I. **Introduction:**

- ✓ To develop a machine learning classification model, we first collect data, then perform data exploration, data pre-processing, and cleaning. After completing all these processes, we apply the classification technique to achieve predictions from that model. This is a brief idea about how we develop a machine learning model. Before finalizing the classifier model, we must be sure if it is performing well or not. Confusion Matrix measures the performance of a classifier to check efficiency and precision in predicting results.

II. **Confusion Matrix Definition:**

- ✓ A confusion matrix is used to judge the performance of a classifier on the test dataset for which we already know the actual values. Confusion matrix is also termed as Error matrix. It consists of a count of correct and incorrect values broken down by each class. It not only tells us the error made by classifier but also tells us what type of error the classifier made. So, we can say that a confusion matrix is a performance measurement technique of a classifier model where output can be two classes or more. It is a table with four different groups of true and predicted values.

III. **Terminologies in Confusion Matrix:**

- ✓ The confusion matrix shows us how our classifier gets confused while predicting. In a confusion matrix we have four important terms which are:

True Positive (TP)
True Negative (TN)
False Positive (FP)

False Negative (FN)

We will explain these terms with the help of visualization of the confusion matrix:

- ✓ This is what a confusion matrix looks like. This is a case of a 2-class confusion matrix. On one side of the table, there are predicted values and on one side there are the actual values.
- ✓ Let us discuss the above terms in detail:
- ✓ True Positive (TP)
  Both actual and predicted values are Positive.
- ✓ True Negative (TN)
  Both actual and predicted values are Negative.
- ✓ False Positive (FP)
  The actual value is negative but we predicted it as positive.
- ✓ False Negative (FN)
  The actual value is positive but we predicted it as negative.

IV. **Performance Metrics:**

- ✓ Confusion matrix not only used for finding the errors in prediction but is also useful to find some important performance metrics like Accuracy, Recall, Precision, F-measure. We will discuss these terms one by one.
- ✓ **Accuracy:**
- As the name suggests, the value of this metric suggests the accuracy of our classifier in predicting results.
- It is defined as:
- Accuracy = (TP + TN) / (TP + TN + FP + FN)
- A 99% accuracy can be good, average, poor or dreadful depending upon the problem.
- ✓ **Precision**
- Precision is the measure of all actual positives out of all predicted positive values.
- It is defined as:
- Precision = TP / (TP + FP)
- ✓ **Recall**
- Recall is the measure of positive values that are predicted correctly out of all actual positive values.
- It is defined as:
- Recall = TP / (TP + FN)
- High Value of Recall specifies that the class is correctly known (because of a small number of False Negative).
- ✓ **F-measure**
- It is hard to compare classification models which have low precision and high recall or vice versa. So, for comparing the two classifier models we use F-measure. F-score helps to find the metrics of Recall and Precision in the same interval. Harmonic Mean is used instead of Arithmetic Mean.
  - F-measure is defined as:
  - F-measure = 2 * Recall * Precision / (Recall + Precision)

- The F-Measure is always closer to the Precision or Recall, whichever has a smaller value.

V. **Calculation of 2-class confusion matrix**

- ✓ Let us derive a confusion matrix and interpret the result using simple mathematics.
- ✓ Let us consider the actual and predicted values of y as given below:

Actual y	Y predicted	Predicted y with threshold 0.5
1	0.7	1
0	0.1	0
0	0.6	1
1	0.4	0
0	0.2	0

- ✓ Now, if we make a confusion matrix from this, it would look like:

N=5	**Predicted 1**	**Predicted 0**
Actual: 1	1 (TP)	1 (FN)
Actual: 0	1 (FP)	2 (TN)

- ✓ This is our derived confusion matrix. Now we can also see all the four terms used in the above confusion matrix. Now we will find all the above-defined performance metrics from this confusion matrix.
  - ✓ **Accuracy:**
  - ✓ Accuracy = (TP + TN) / (TP + TN + FP + FN)
  - ✓ So, Accuracy = (1+2) / (1+2+1+1)
    = 3/5 which is 60%.
- ✓ So, the accuracy from the above confusion matrix is 60%.
  - ✓ **Precision**
  - ✓ Precision = TP / (TP + FP)
    = 1 / (1+1)
    =1 / 2 which is 50%.
- ✓ So, the precision is 50%.
  - ✓ **Recall**
  - ✓ Recall = TP / (TP + FN)
    = 1 / (1+1)
    = ½ which is 50%
- ✓ So, the Recall is 50%.
  - ✓ **F-measure**
  - ✓ F-measure = 2 * Recall * Precision / (Recall + Precision)
    = 2*0.5*0.5 / (0.5+0.5)
    = 0.5
- ✓ So, the F-measure is 50%.

Algorithm	1. Read the social_media dataset
	`import pandas as pd`
	`df = pd.read_csv('https://raw.githubusercontent.com/shivang98/Social-Network-ads-`

```
Boost/master/Social_Network_Ads.csv')
```

2. find out correlation between input and output variable

How do we do it?

1) Co-relation matrix

2) Ploting on Heatmap

```
import seaborn as sns
sns.heatmap(df.corr(), annot=True)
```

3. Divide the data into training and testing

```
X = df[['Age', 'EstimatedSalary']]
Y = df['Purchased']
from sklearn.model_selection import train_test_spli
t

X_train, X_test, Y_train, Y_test = train_test_split
(X, Y, test_size=0.2, random_state=42)
```

4. Apply logistic regression

```
from sklearn.linear_model import LogisticRegression
model = LogisticRegression()
model.fit(X_train,Y_train)
print('Model Score: ', model.score(X_test, Y_test))
```

5. Normalization means keeping the data within the range [0 - 1]

```
X = df[['Age', 'EstimatedSalary']]
Y = df['Purchased']

# Divide the data into training and testing
from sklearn.model_selection import train_test_spli
t
X_train, X_test, Y_train, Y_test = train_test_split
(X, Y, test_size=0.2, random_state=42)

from sklearn.preprocessing import MinMaxScaler

# fit scaler on training data
norm = MinMaxScaler().fit(X_train)

# transform training data
X_train = norm.transform(X_train)
```

```
# fit scaler on training data
norm = MinMaxScaler().fit(X_test)

# transform training data
X_test = norm.transform(X_test)
Apply logistic regression
from sklearn.linear_model import LogisticRegression
model = LogisticRegression()
model.fit(X_train,Y_train)
print('Model Score: ', model.score(X_test, Y_test))

X = df[['Age', 'EstimatedSalary']]
Y = df['Purchased']

from sklearn.model_selection import train_test_spli
t
X_train, X_test, Y_train, Y_test = train_test_split
(X, Y, test_size=0.2, random_state=42)

print(X_train)
print(X_test)
from sklearn.preprocessing import MinMaxScaler

# fit scaler on training data
norm = MinMaxScaler().fit(X_train)

# transform training data
X_train = norm.transform(X_train)

# fit scaler on training data
norm = MinMaxScaler().fit(X_test)

# transform training data
X_test = norm.transform(X_test)
# Apply logistic regression
from sklearn.linear_model import LogisticRegression
model = LogisticRegression()
model.fit(X_train,Y_train)
y_pred = model.predict(X_test)

print('Model Score: ', model.score(X_test, Y_test))
```

6. Print/Display the confusion matrix

```
from sklearn.metrics import confusion_matrix
cf_matrix = confusion_matrix(Y_test, y_pred)  #Actu
```

	al output and predicted output print(cf_matrix) 7. Display the score: accuracy, precision, recall, f-score from sklearn.metrics import precision_recall_fscore_support score = precision_recall_fscore_support(Y_test, y_pred, average='micro') print('Precision of Model: ', score[0]) print('Recall of Model: ', score[1]) print('F-Score of Model: ', score[2])
Conclusion	In this assignment we can: 1. Implement logistic regression 2. Compute Confusion matrix by using Social_Network_Ads.csv dataset. 3. Find out find TP, FP, TN, FN, Accuracy, Error rate, Precision, Recall on the given dataset.
Questions	1. What is Confusion Matrix and why you need it? 2. What is a confusion matrix? 3. How to Calculate Confusion Matrix for a 2-class classification problem? 4. How to calculate 1. Recall 2. Precision 3. Accuracy 4. F1 measure 5. F1 score

Assignment Number – 6 (Data Analytics III)	
Title & Problem Statement	Implement Simple Naive 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.
Objectives	1. Understand the Naive Bayes Classification Algorithm 2. Understand and Compute Confusion matrix 3. Display Accuracy, Error rate, Precision, Recall
Outcomes	Students will be able to: 1. Implement Naive Bayes Classification Algorithm using Python 2. Display correlation matrix 3. Train a model and classify the data 4. Calculate Accuracy, Error rate, Precision, Recall
S/W Requirement	OS – Linux Ubuntu 18 (64 bit) Packages - Sublime text editor, Python 3, Seaborn, Matplotlib, Pandas

Theory

What is Naive Bayes Classification?

- ✓ Naive Bayes is a statistical classification technique based on the Bayes Theorem and one of the simplest Supervised Learning algorithms.
- ✓ The Naive Bayes classifier is a quick, accurate, and trustworthy method, especially on large datasets.
- ✓ The simple formula of Bayes theorem is: P(A/B) = $\frac{P\left(\frac{B}{A}\right).P(A)}{P(B)}$
  - Where P(A) and P(B) are two independent events and (B) is not equal to zero
  - P(A/B): Is the conditional probability of event A occurring given that B is true
  - P(B/A): Is the conditional probability of event B occurring given that A is true.
  - P(A) and P(B) are the probabilities of A and B occurring independently of one another (the marginal probability).
- ✓ The Naive Bayes classification algorithm is a probabilistic classifier, and it belongs to Supervised Learning.
- ✓ It is based on probability models that incorporate strong independence assumptions.
- ✓ The independence assumptions often do not have an impact on reality.
- ✓ Therefore, they are considered naive.
- ✓ Another assumption made by the Naive Bayes classifier is that all the predictors have an equal effect on the outcome.

✓ The Naive Bayes classification has the following different types:

1)Multinomial Naive Bayes method:

It is a common Bayesian learning approach in natural language processing. Using the Bayes theorem, the program estimates the tag of a text, such as an email or a newspaper piece. It assesses the likelihood of each tag for a given sample and returns the tag with the highest possibility.

2) Bernoulli Naive Bayes:

It is a part of the family of Naive Bayes. It only takes binary values. There may be multiple features, but each is assumed to be a binary-valued (Bernoulli, boolean) variable. Therefore, this class requires samples to be represented as binary-valued feature vectors.

3) Gaussian Naive Bayes:

It is a variant of Naive Bayes that follows Gaussian normal distribution and supports continuous data. To build a simple model using Gaussian Naive Bayes, we assume the data is characterized by a Gaussian distribution with no covariance (independent dimensions) between the parameters. This model may be fit simply by calculating the mean and standard deviation of the points within each label.

Naive Bayes classifier makes two fundamental assumptions on the observations:

- The target classes are independent to each other. Consider a rainy day with strong winds and high humidity. These two features, wind and humidity, would be treated as independent by a Naive classifier. That is to say, each feature would impose its own probabilities on the outcome, such as rain in this case.
- Prior probabilities for the target classes are equal. That is, before calculating the posterior probability of each class, the classifier will assign each target class the same prior probability.

✓ In above problem statement we use Iris flower dataset to implement Simple Naïve Bayes classification algorithm. Use Sepal Length, Sepal Width, Petal length and Petal Width as input And Class is as Output.

Algorithm

Read Dataset

```
import pandas as pd
 df=pd.read_csv("iris.csv")
print(df)
print(df.dtypes)
```

Find Missing value

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

Here class is object type convert it in to 0,1 format by assigning it as category

```
df['variety']=df['variety'].astype('category')
print(df.dtypes)
df['variety']=df['variety'].cat.codes
print(df.dtypes)
print(df.isnull().sum())
```

check for outliers by using IQR method co-relation matrix

```
def DetectOutlier(df,var):
Q1 = df[var].quantile(0.25)
Q3 = df[var].quantile(0.75)
IQR = Q3 - Q1
high, low = Q3+1.5*IQR, Q1-1.5*IQR
print("Highest allowed in variable:", var, high)
print("lowest allowed in variable:", var, low)

count = df[(df[var] > high) | (df[var] < low)][var].count()

print('Total outliers in:',var,':',count)

import seaborn as sns
sns.boxplot(df['sepal.width'])

def OutlierRemoval(df,var):
Q1 = df[var].quantile(0.25)
Q3 = df[var].quantile(0.75)
IQR = Q3 - Q1
high, low = Q3+1.5*IQR, Q1-1.5*IQR

print("Highest allowed in variable:", var, high)
print("lowest allowed in variable:", var, low)

count = df[(df[var] > high) | (df[var] < low)][var].count()

print('Total outliers in:',var,':',count)

df = df[((df[var] >= low) & (df[var] <= high))]
return df

print(df.shape)
df = OutlierRemoval(df,'sepal.width')
print(df.shape)

import seaborn as sns
```

```
sns.heatmap(df.corr(),annot=True)

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

Split the data into inputs and outputs

```
X = df.iloc[:, [0,1,2,3]].values
y = df.iloc[:, 4].values
```

Training and testing data

```
from sklearn.model_selection import train_test_split
```

Assign test data size 20%

```
X_train, X_test, y_train, y_test =train_test_split(X,y,test_si
ze= 0.20, random_state=0)
```

Importing standard scaler

```
from sklearn.preprocessing import StandardScaler
```

Scalling the input data

```
sc_X = StandardScaler()
X_train = sc_X.fit_transform(X_train)
X_test = sc_X.fit_transform(X_test)
```

Importing standard scaler

```
from sklearn.preprocessing import StandardScaler
```

Scalling the input data

```
sc_X = StandardScaler()
X_train = sc_X.fit_transform(X_train)
X_test = sc_X.fit_transform(X_test)
```

Importing accuracy score

```
from sklearn.metrics import accuracy_score
```

Importing classifier

```
        from sklearn.naive_bayes import BernoulliNB
```

Initializaing the NB

```
        classifer = BernoulliNB()
```

Training the model

```
        classifer.fit(X_train, y_train)
```

Testing the model

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

Importing accuracy score

```
from sklearn.metrics import accuracy_score
```

Printing the accuracy of the model

```
print(accuracy_score(y_pred, y_test))
```

Printing the accuracy of the model

```
print(accuracy_score(y_pred, y_test))
```

Import Gaussian Naive Bayes classifier

```
from sklearn.naive_bayes import GaussianNB
```

Create a Gaussian Classifier

```
classifer1 = GaussianNB()
```

Training the model

```
classifer1.fit(X_train, y_train)
```

Testing the model

```
y_pred1 = classifer1.predict(X_test)
```

Importing accuracy score

```
from sklearn.metrics import accuracy_score
```

Printing the accuracy of the model

```
print(accuracy_score(y_test,y_pred1))
```

Importing the required modules

```
import seaborn as sns
from sklearn.metrics import confusion_matrix
```

Passing actual and predicted values

```
cm = confusion_matrix(y_test, y_pred)
```

True write data values in each cell of the matrix

```
sns.heatmap(cm, annot=True)
plt.savefig('confusion.png')
```

Importing classification report

```
from sklearn.metrics import classification_report
```

Printing the report

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

Output: precision recall f1-score support

0	0.88	1.00	0.94	15
1	1.00	0.46	0.63	13
2	0.64	1.00	0.78	9
accuracy			0.81	37
macro avg	0.84	0.82	0.78	37
weighted avg	0.87	0.81	0.79	37

Predict iris flower Varity by giving user input:

```
import numpy as np
features = np.array([[5,2.9,1,0.2]])
prediction = classifer.predict(features)
print('Prediction: {}'.format(prediction))
```

Steps:

5. Load the iris.csv Dataset into Pandas Dataframe.
6. Display Preprocessing on Dataset.
7. Remove Outliers by using Inter Quartile Range(IQR)Method.
8. Display confusion Matrix by using Seaborn library.
9. Apply Gaussian Naive Bayes Algorithm and display Accuracy, Error Rate, Precision, Recall, F1-Score of Bernoulli Naive Bayes and Gaussian Naive Bayes.
10. Predict iris flower Varity by giving user input.

Conclusion	In this assignment we are able to: 1. Find that Gaussian Naive Bayes Algorithm gives more Accuracy and less Error Rate than Bernaulli NB. 2. Understand what is Accuracy, Error Rate, Precision, Recall, F1-Score. 3. Visualize statistical information using Seaborn and Matplotlib.
Questions	1. What is Data Analytics? 2. What is Naive Bayes Classification? 3. Define types of Naive Bayes classification? 4. What is Gaussian Naive Bayes classification 5. What is IQR Method? 6. What is Confusion Matrix? 7. What is TP, FP, TN, FN? 8. How to display Confusion Matrix? 9. What are the parameters like Accuracy, Error Rate, Precision, Recall, F1-Score of Dataset? 10. Explain Iris flower Dataset?

Assignment Number - 7	
Title & Problem Statement	**Text Analytics:** 1. Extract Sample document and apply following document preprocessing methods: Tokenization, POS Tagging, stop words removal, Stemming and Lemmatization. 2. Create representation of document by calculating Term Frequency and Inverse Document Frequency.
Objectives	1. Understand the basic concepts of Text Analytics 2. Perform Text Analysis Operations using Natural Language ToolKit (NLTK) 3. Understand and perform Text Analysis Model using TF-IDF.
Outcomes	Students will be able to: 1. Perform Text Analysis Operations using NLTK like Tokenization, stop word removal, Stemming, Lemmatization and POS Tagging. 2. Calculate Term Frequency (TF), Inverse Document Frequency (IDF) and TF-IDF.
S/W Requirement	OS – Linux Ubuntu 18 (64 bit) Packages - Sublime text editor, Python 3, NLTK

Theory

1. Basic concepts of Text Analytics

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

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

2. Text Analysis Operations using Natural Language ToolKit (NLTK)

NLTK (Natural language toolkit) is a leading platform for building Python programs to work with human language data. It provides easy-to-use interfaces and lexical resources such as WordNet, along with a suite of text processing libraries for classification, tokenization, stemming, tagging, parsing, and semantic reasoning and many more.

Analyzing movie reviews is one of the classic examples to demonstrate a simple NLP Bag-of-words model, on movie reviews.

2.1. Tokenization:

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

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

2.2. Stop words removal

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

2.3. Stemming and Lemmatization

Stemming is a normalization technique where lists of tokenized words are converted into shortened root words to remove redundancy. Stemming is the process of reducing inflected (or sometimes derived) words to their word stem, base or root form.

A computer program that stems word may be called a stemmer.
Example:
A stemmer reduces the words like fishing, fished, and fisher to the stem fish.
The stem need not be a word, for example the Porter algorithm reduces, argue, argued, argues, arguing, and argus to the stem argu.

Lemmatization in NLTK is the algorithmic process of finding the lemma of a word depending on its meaning and context. Lemmatization usually refers to the morphological analysis of words, which aims to remove inflectional endings. It helps in returning the base or dictionary form of a word known as the lemma.
E.g. Lemma for studies is study

2.4. POS Tagging

POS (Parts of Speech) tells us about grammatical information of words of the sentence by assigning specific token (Determiner, noun, adjective, adverb, verb, Personal Pronoun etc.) as tag (DT, NN, JJ, RB, VB, PRP etc) to each word.

Word can have more than one POS depending upon the context where it is used. We can use POS tags as statistical NLP tasks. It distinguishes a sense of word which is very helpful in text realization and infer semantic information from text for sentiment analysis.

3. Text Analysis Model using TF-IDF.

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

3.1. Term Frequency (TF)

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

$$TF(w,d) = \frac{Occurences\ of\ w\ in\ document\ d}{Total\ no.of\ words\ in\ document\ d}$$

Example:

Documents	Text	Total no. of words in Document
A	Jupiter is the largest planet.	5
B	Mars is the fourth planet from the sun.	8

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

3.2. Inverse Document Frequency (IDF)

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

$$IDF(w,D) = \frac{Total\ no.ofdocuments\ (N)\ in\ corpus\ D}{Number\ of\ documents\ containing\ w}$$

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

Words	TF (for A)	TF (for B)	IDF
Jupiter	$^1/_5$	0	$ln(^2/_1) = 0.69$
is	$^1/_5$	$^1/_8$	$ln(^2/_2) = 0$
the	$^1/_5$	$^2/_8$	$ln(^2/_2) = 0$
largest	$^1/_5$	0	$ln(^2/_1) = 0.69$
planet	$^1/_5$	$^1/_8$	$ln(^2/_2) = 0$

Mars	0	$^1/_8$	$ln(^2/_1) = 0.69$
fourth	0	$^1/_8$	$ln(^2/_1) = 0.69$
from	0	$^1/_8$	$ln(^2/_1) = 0.69$
sun	0	$^1/_8$	$ln(^2/_1) = 0.69$

3.3. Term Frequency — Inverse Document Frequency (TFIDF)

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

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

Words	TF (for A)	TF (for B)	IDF	TFIDF(A)	TFIDF(B)
Jupiter	$^1/_5$	0	$ln(^2/_1) = 0.69$	0.138	0
is	$^1/_5$	$^1/_8$	$ln(^2/_2) = 0$	0	0
the	$^1/_5$	$^2/_8$	$ln(^2/_2) = 0$	0	0
largest	$^1/_5$	0	$ln(^2/_1) = 0.69$	0.138	0
planet	$^1/_5$	$^1/_8$	$ln(^2/_2) = 0$	0	0
Mars	0	$^1/_8$	$ln(^2/_1) = 0.69$	0	0.086
fourth	0	$^1/_8$	$ln(^2/_1) = 0.69$	0	0.086
from	0	$^1/_8$	$ln(^2/_1) = 0.69$	0	0.086
sun	0	$^1/_8$	$ln(^2/_1) = 0.69$	0	0.086

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

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

Disadvantage of TF-IDF

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

computationally expensive if the vocabulary is vast.

4. Bag of Words (BoW)

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

Algorithm

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

Step 1: Download the required packages

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

Step 2: Initialize the text

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

Step 3: Perform Tokenization

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

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

Step 4: Removing Punctuations and Stop Word

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

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

print ("Tokenized Sentence:",tokens)
print ("Filterd Sentence:",filtered_text)
```

Step 5: Perform Stemming

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

Step 6: Perform Lemmatization

```
from nltk.stem import WordNetLemmatizer
wordnet_lemmatizer = WordNetLemmatizer()
text = "studies studying cries cry"
tokenization = nltk.word_tokenize(text)
for w in tokenization:
      print("Lemma for {} is {}".format(w,
wordnet_lemmatizer.lemmatize(w)))
```

Step 7: Apply POS Tagging to text

```
import nltk
from nltk.tokenize import word_tokenize
data="The pink sweater fit her perfectly"
words=word_tokenize(data)

for word in words:
      print(nltk.pos_tag([word]))
```

2. **Algorithm for Create representation of document by calculating TFIDF**

Step 1: Import the necessary libraries.

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

Step 2: Initialize the Documents.
Step 3: Create Bag of Words (BoW) for Document A and B.
Step 4: Create Collection of Unique words from Document A and B.
Step 5: Create a dictionary of words and their occurrence for each document in the corpus
Step 6: Compute the term frequency for each of our documents.
Step 7: Compute the term Inverse Document Frequency.
Step 8: Compute the term TF-IDF for all words.

Conclusion

In this assignment we are able to do text data analysis by:

1. Performing Text Analysis Operations using NLTK like Tokenization, stop word removal, Stemming, Lemmatization and POS Tagging.
2. Calculating Term Frequency (TF), Inverse Document Frequency (IDF) and TF-IDF.

Questions	1. What do you understand by Natural Language Processing? 2. What is Text Analytics? 3. Why is text analysis useful? 4. What is NLTK? 5. How do you analyze text data in Python? 6. What is Tokenization? 7. What are Stop words? 8. What is Bag of Words? 9. What is the difference between Stemming and Lemmatization? 10. Which are the libraries used for text analysis using NLTK? 11. What is TF-IDF? 12. What are the steps for calculating TF-IDF? 13. What is PoS Tagging? 14. How do you preprocess text in NLP? 15. What are some real-life NLP applications?

Assignment Number - 8	
Title & Problem Statement	**Data Visualization I :** 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.
Objectives	1. Understand the basic concepts of Data Visualization 2. Display statistical information of the Titanic dataset using histogram. 3. Perform visualization of the results.
Outcomes	Students will be able to: 1. Find if any patterns in the dataset using data visualization. 2. Display statistical information using seaborn and matplotlib library. 3. Use seaborn library to plot histogram of continuous variable.
S/W Requirement	OS – Linux Ubuntu 18 (64 bit) Packages - Sublime text editor, Python 3, Seaborn, Matplotlib, Pandas
Theory	

- **What is Seaborn Library?**
  - ✓ Popular data visualization library.
  - ✓ Seaborn is a python data visualization library based on matplotlib.
  - ✓ It provides a high level interface for drawing attractive and informative statistical graphics.
  - ✓ It provides choices for plot style and color defaults.
  - ✓ Simple high level functions for common statistical plot types and integrates with the functionality provided by Pandas DataFrames.
  - ✓ The main idea of seaborn is that it provides high-level commands to create a variety of plot types useful for statistical data exploration and even some statistical model fitting.
  - ✓ In seaborn, we can plot the graph in just 1 or 2 lines as compared to matplotlib.
- **How to import seaborn library?**
  - ✓ `import seaborn as sns`
  - ✓ `sns.set()`
  - ✓ `sns.set(style="darkgrid")`
  - ✓ `import warnings`
  - ✓ `warnings.filterwarnings("ignore")`
- **What is Histogram?**
  - ✓ A histogram is an approximate representation of the distribution of numerical data. The term was first introduced by Karl Pearson.
  - ✓ To construct a histogram, the first step is to "bin" (or "bucket") the range of values—that is, divide the entire range of values into a series of intervals—and then count how many values fall into each interval. The bins are usually specified as consecutive, non-overlapping intervals of a variable. The bins (intervals) must be adjacent and are often (but not required to be) of equal size
  - ✓ If the bins are of equal size, a bar is drawn over the bin with height proportional to the frequency—the number of cases in each bin
  - ✓ However, bins need not be of equal width; in that case, the erected rectangle is defined to have its area proportional to the frequency of cases in the bin.
  - ✓ The data used to construct a histogram are generated via a function mi that counts the number of observations that fall into each of the disjoint categories (known as bins). Thus, if we let n be the total number of observations and k be the total number of bins, the histogram data mi meet the following conditions:

$$n = \sum_{i=1}^{k} m_i.$$

- **How to draw Histogram in seaborn?**
  - ✓ You can create a histogram in seaborn by simply using the **distplot().**
  - ✓ `sns.distplot(df['Column'])`

- **What is the difference between bar plot and histogram?**

Bar plot	Histogram
✓ Bar plot is a plot of categorical	1. A histogram is used for continuous

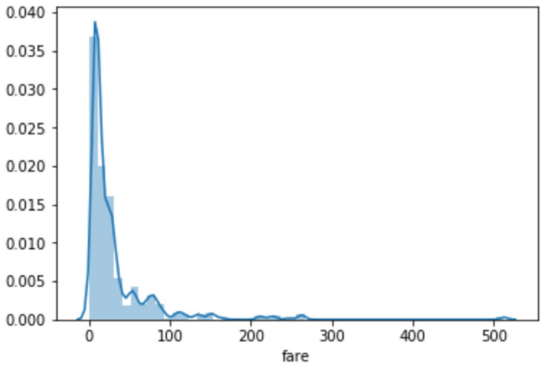
variables.	data, where the bins represent ranges of data.
2. A bar graph is a chart that uses bars to represent the frequency or quantity of different categories of data	2. A histogram, on the other hand, is a graph that shows the distribution of numerical data.
3. Bar plot bars do not touch to each other.	3. Histogram bar touch to each other.
4. Bar plot are used to compare Variables.	4. Histogram are used to show distributions of variables.
5. Bars can be reordered in bar charts	5. Reordering is not possible in histogram.

- **How to load the Titanic dataset?**

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

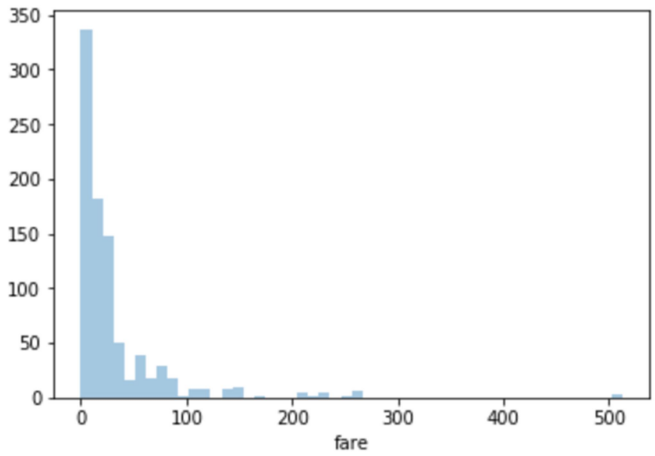
- ✓ `import pandas as pd`
- ✓ `import numpy as np`
- ✓ `import matplotlib.pyplot as plt`
- ✓ `import seaborn as sns`
- ✓ `dataset = sns.load_dataset('titanic')`
- ✓ `dataset.head()`
- ✓ `The Dist Plot:`
- ✓ `The distplot()` shows the histogram distribution of data for a single column. The column name is passed as a parameter to the distplot()function. Let's see how the price of the ticket for each passenger is distributed. Execute the following script:

- ✓ sns.distplot(dataset['fare'])



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

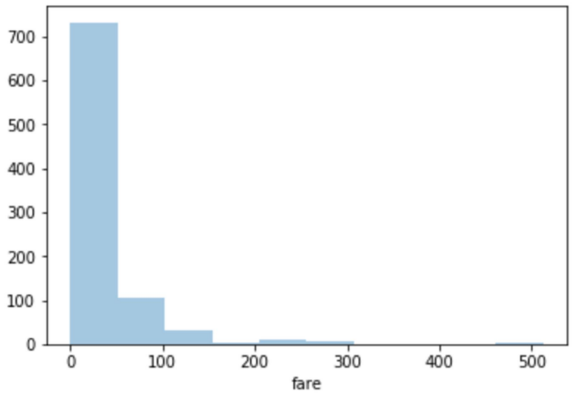
sns.distplot(dataset['fare'], kde=False)



- ✓ Now you can see there is no line for the kernel density estimation on the plot. We can also pass the value for the bins parameter in order to see more or less details in the graph. Take a look at the following script:

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

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



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

Algorithm	1. Load the Titanic Dataset into Pandas Dataframe 2. Perform data preparation by finding the missing values and replacing them with median value. 3. Draw heatmap and find the correlation between multiple columns. Find out the strong associated columns for feature engineering. 4. Draw histogram by plotting distplot on 'Age' column.

Conclusion	In this assignment we are able to: 1. Plot the histogram using distplot() function of the seaborn library. 2. Use different parameters to plot histogram by changing KDE (i.e. Kernel Density Function) 3. Visualize statistical information using Seaborn library.
Questions	1. What is data visualization? 2. What is histogram? 3. What is the difference between bar plot and histogram? 4. What is the difference between matplotlib and seaborn? 5. How do you plot a histogram in seaborn? 6. Which are the libraries used? 7. How to display information in tabular format in python? 8. What is barplot? 9. What are the parameters in Seaborn barplot function? 10. Explain Iris flower dataset? 11. Explain Employee salary dataset?

Assignment Number - 9	
Title & 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') 2. Write observations on the inference from the above statistics.
Objectives	Write a python program to load the dataset and understand the input data Dataset: 'titanic' Dataset 1) Load data, describe the given data. 2) Plot the box plot for age distribution with respect to each gender with the information about whether they survived or not. 3) Find observations on the inference from the above statistics.
Outcomes	Students will be able to: 1. Plot the box plot for age distribution with respect to each gender with the information about whether they survived or not by considering the 'sex' and 'age' columns. 2. Find observations on the inference from the above statistics.
S/W Requirement	OS – Linux Ubuntu 18 (64 bit) Packages - Sublime text editor, Python 3, Seaborn, Matplotlib, Pandas

Theory

What is Data Visualization?

- ✓ Data Visualization represents the text or numerical data in a visual format, which makes it easy to grasp the information the data express.
- ✓ We, humans, remember the pictures more easily than readable text, so Python provides us various libraries for data visualization like matplotlib, seaborn, plotly, etc.
- ✓ We will use Matplotlib and seaborn for performing various techniques to explore data using various plots.

Exploratory Data Analysis

- ✓ Creating Hypotheses, testing various business assumptions while dealing with any Machine learning problem statement is very important and this is what EDA helps to accomplish.
- ✓ There are various tootle and techniques to understand your data, And the basic need is you should have the knowledge of Numpy for mathematical operations and Pandas for data manipulation.
- ✓ We will use a very popular Titanic dataset with which everyone is familiar with and you can download it from here.
- ✓ Now lets us start exploring data and study different data visualization plots with

different types of data.

- ✓ Let's get started by importing libraries and loading Data.

```
import numpy as np
import pandas pd
import matplotlib.pyplot as plt
import seaborn as sns
from seaborn import load_dataset
#titanic dataset
data = pd.read_csv("titanic_train.csv")
#tips dataset
tips = load_dataset("tips")
```

Univariate Analysis

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

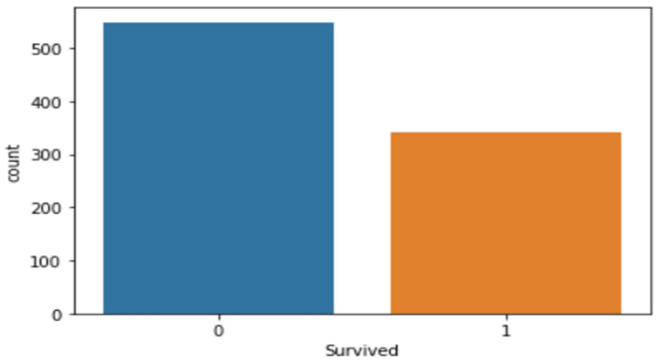
Categorical Data

- ✓ A variable that has text-based information is referred to as categorical variables. let's look at various plots which we can use for visualizing Categorical data.

1. CountPlot

- ✓ Countplot is basically a count of frequency plot in form of a bar graph.
- ✓ It plots the count of each category in a separate bar.
- ✓ When we use the pandas' value counts function on any column, it is the same visual form of the value counts function. In our data-target variable is survived and it is categorical so let us plot a countplot of this.

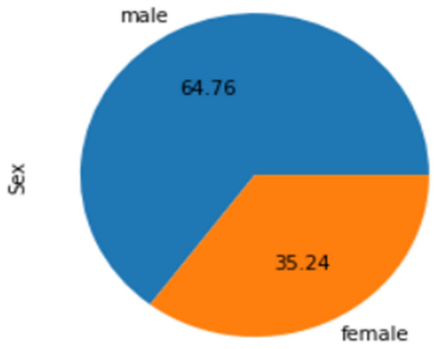
```
sns.countplot(data['Survived'])
plt.show()
```



2. Pie Chart

- ✓ The pie chart is also the same as the countplot, only gives you additional information about the percentage presence of each category in data means which category is getting how much weightage in data. let us check about the Sex column, what is a percentage of Male and Female members traveling.

```
data['Sex'].value_counts().plot(kind="pie", autopct="%.2f")
plt.show()
```



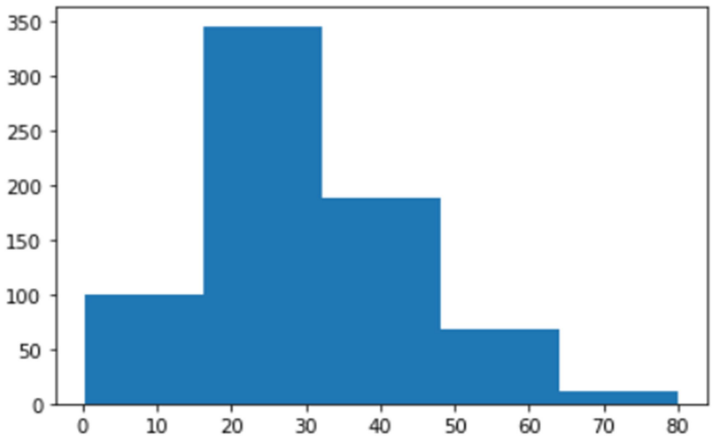
Numerical Data

- ✓ Analyzing Numerical data is important because understanding the distribution of variables helps to further process the data.
- ✓ Most of the time you will find much inconsistency with numerical data so do explore numerical variables.

3. Histogram

- ✓ A histogram is a value distribution plot of numerical columns.
- ✓ It basically creates bins in various ranges in values and plots it where we can visualize how values are distributed. We can have a look where more values lie like in positive, negative, or at the center(mean).
- ✓ Let's have a look at the Age column.

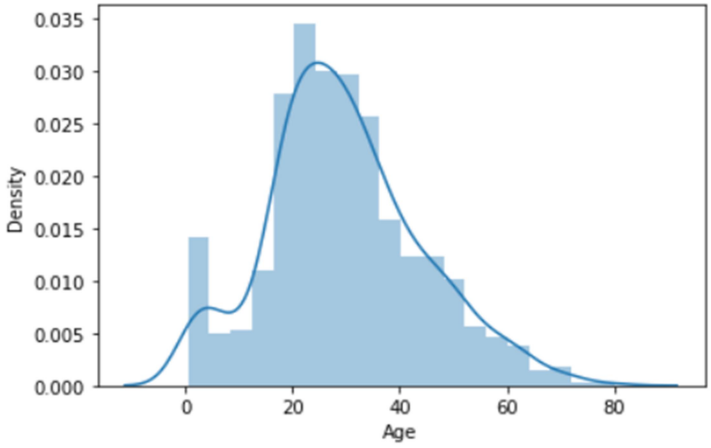
```
plt.hist(data['Age'], bins=5)
plt.show()
```



4. Distplot

- ✓ Distplot is also known as the second Histogram because it is a slight improvement version of the Histogram.
- ✓ Distplot gives us a KDE(Kernel Density Estimation) over histogram which explains PDF(Probability Density Function) which means what is the probability of each value occurring in this column.
- ✓ If you have study statistics before then definitely you should know about PDF function.

```
sns.distplot(data['Age'])
plt.show()
```



5. Boxplot

- ✓ Boxplot is a very interesting plot that basically plots a 5 number summary.
- ✓ To get 5 number summary some terms we need to describe.
- ✓ **Median** – Middle value in series after sorting
- ✓ **Percentile** – Gives any number which is number of values present before this percentile like for example 50 under 25th percentile so it explains total of 50 values are there below 25th percentile
- ✓ **Minimum and Maximum** – These are not minimum and maximum values, rather

they describe the lower and upper boundary of standard deviation which is calculated using Interquartile range (IQR).

```
IQR = Q3 - Q1
Lower_boundary = Q1 - 1.5 * IQR
Upper_bounday = Q3 +  1.5 * IQR
```

- ✓ Here Q1 and Q3 is 1st quantile (25th percentile) and 3rd Quantile (75th percentile)

Bivariate/ Multivariate Analysis

- ✓ We have study about various plots to explore single categorical and numerical data. Bivariate Analysis is used when we have to explore the relationship between 2 different variables and we have to do this because, in the end, our main task is to explore the relationship between variables to build a powerful model.
- ✓ When we analyze more than 2 variables together then it is known as Multivariate Analysis. we will work on different plots for Bivariate as well on Multivariate Analysis.

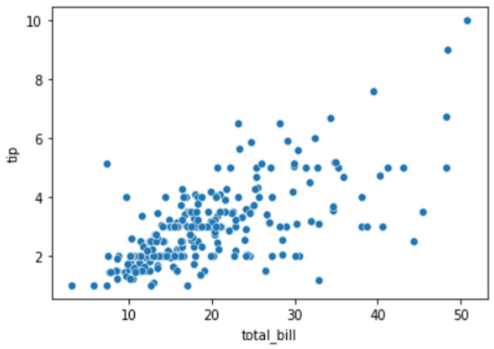
Numerical and Numerical

First, let's explore the plots when both the variable is numerical.

6. Scatter Plot

- ✓ To plot the relationship between two numerical variables, scatter plot is a simple plot to do. Let us see the relationship between the total bill and tip provided using a scatter plot.

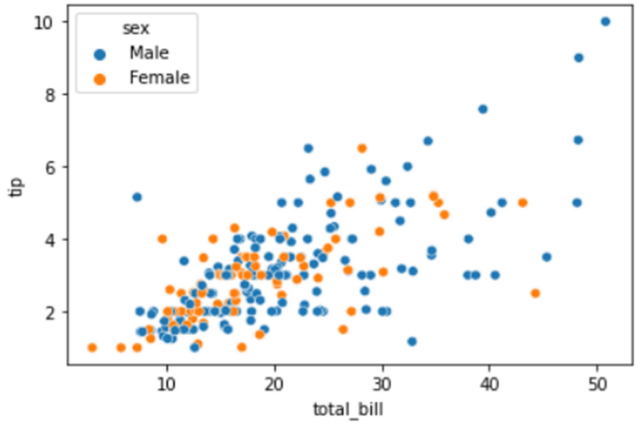
```
sns.scatterplot(tips["total_bill"], tips["tip"])
```



Multivariate analysis with scatter plot

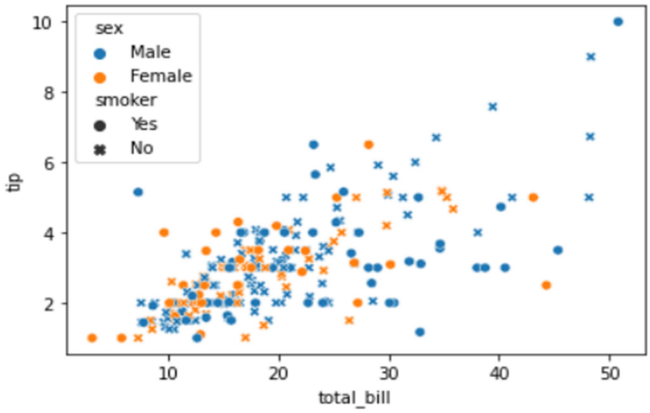
- ✓ We can also plot 3 variable or 4 variable relationships with scatter plot.
- ✓ Suppose we want to find the separate ratio of male and female with total bill and tip provided.

```
sns.scatterplot(tips["total_bill"], tips["tip"], hue=tips["sex"])
plt.show()
```



- ✓ We can also see 4 variable multivariate analyses with scatter plots using style argument. Suppose now along with gender I also want to know whether the customer was a smoker or not so we can do this.

```
sns.scatterplot(tips["total_bill"], tips["tip"], hue=tips["sex"], style=tips['smoker'])
plt.show()
```



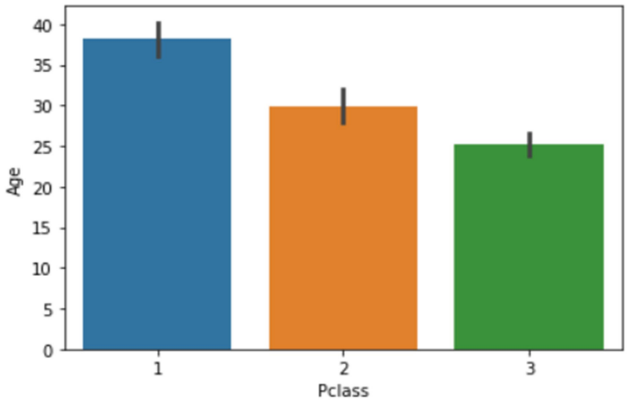
Numerical and Categorical

- ✓ If one variable is numerical and one is categorical then there are various plots that we can use for Bivariate and Multivariate analysis.

7. Bar Plot

- ✓ Bar plot is a simple plot which we can use to plot categorical variable on the x-axis and numerical variable on y-axis and explore the relationship between both variables.
- ✓ The blacktip on top of each bar shows the confidence Interval. let us explore P-Class with age.

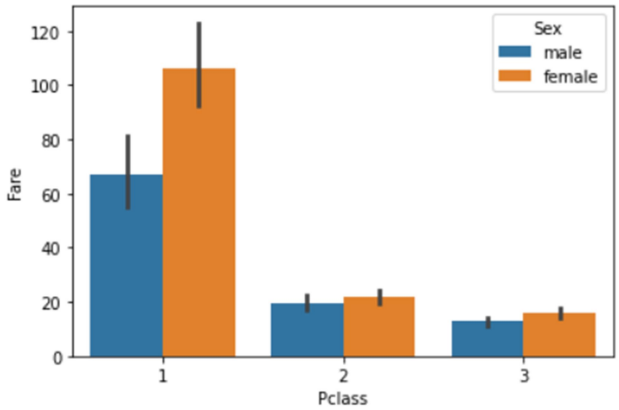
```
sns.barplot(data['Pclass'], data['Age'])
plt.show()
```



Multivariate analysis using Bar plot

- ✓ Hue's argument is very useful which helps to analyze more than 2 variables. Now along with the above relationship we want to see with gender.

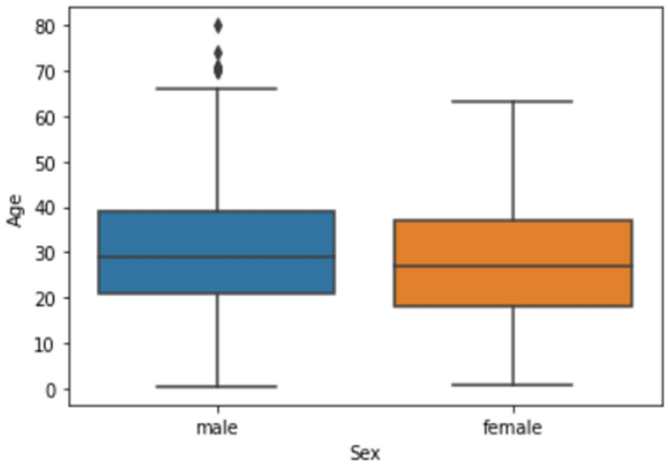
```
sns.barplot(data['Pclass'], data['Fare'], hue = data["Sex"])
plt.show()
```



Boxplot

- ✓ We have already study about boxplots in the Univariate analysis above. we can draw a separate boxplot for both the variable. let us explore gender with age using a boxplot.

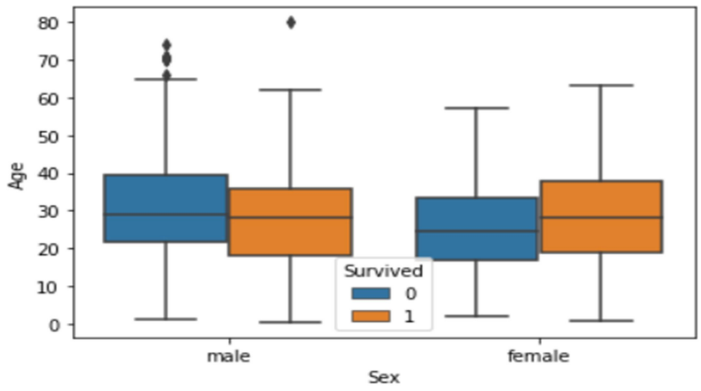
```
sns.boxplot(data['Sex'], data["Age"])
```



Multivariate analysis with boxplot

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

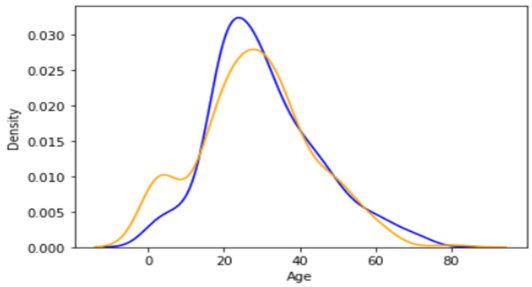
```
sns.boxplot(data['Sex'], data["Age"], data["Survived"])
plt.show()
```



Distplot

- ✓ Distplot explains the PDF function using kernel density estimation.
- ✓ Distplot does not have a hue parameter but we can create it.
- ✓ Suppose we want to see the probability of people with an age range that of survival probability and find out whose survival probability is high to the age range of death ratio.

```
sns.distplot(data[data['Survived'] == 0]['Age'], hist=False, color="blue")
sns.distplot(data[data['Survived'] == 1]['Age'], hist=False, color="orange")
plt.show()
```



- ✓ As we can see the graph is really very interesting.
- ✓ The blue one shows the probability of dying and the orange plot shows the survival probability. If we observe it we can see that children's survival probability is higher than death and which is the opposite in the case of aged peoples.
- ✓ This small analysis tells sometimes some big things about data and It helps while preparing data stories.

Categorical and Categorical

- ✓ Now we will work on categorical and categorical columns.

1. Heatmap

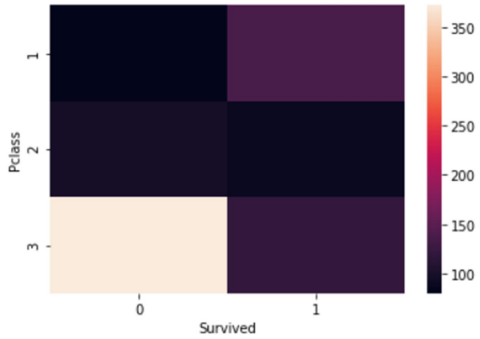
- ✓ If you have ever used a crosstab function of pandas then Heatmap is a similar visual representation of that only.
- ✓ It basically shows that how much presence of one category concerning another category is present in the dataset. let me show first with crosstab and then with heatmap.

```
pd.crosstab(data['Pclass'], data['Survived'])
```

Survived	0	1
Pclass		
1	80	136
2	97	87
3	372	119

- ✓ Now with heatmap, we have to find how many people survived and died.

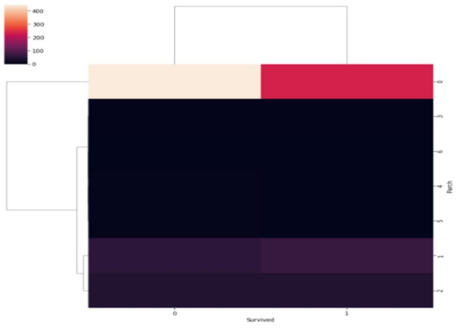
```
sns.heatmap(pd.crosstab(data['Pclass'], data['Survived']))
```



2. Cluster map

✓ We can also use a cluster map to understand the relationship between two categorical variables. A cluster map basically plots a dendrogram that shows the categories of similar behavior together.

```
sns.clustermap(pd.crosstab(data['Parch'], data['Survived']))
plt.show()
```



Algorithm	1. Load the 'titanic' Dataset into Pandas Dataframe 2. Plot the box plot for age distribution with respect to each gender. 3. Plot the box plot by considering the columns 'sex' and 'age' whether they survived or not into above plotted box plot. 4. Find observations on the inference from the above statistics.
Conclusion	In this assignment we are able to: 1. Plot the box plot for age distribution with respect to each gender with the information about whether they survived or not. 2. Find observations on the inference from the above statistics
Questions	1. What is Data Visualization? 2. What is Exploratory Data Analysis? 3. What is Univariate Analysis? 4. What is mean by Categorical Data? 5. What is Bivariate/ Multivariate Analysis? 6. What is the use of Cluster map? 7. What is the use of Heatmap? 8. What is Distplot? 9. What is the use of scatter plot?

Assignment Number - 10	
Title & Problem Statement	**Data Visualization** Download the Iris flower dataset or any other dataset into a DataFrame. (e.g., https://archive.ics.uci.edu/ml/datasets/Iris). Scan the dataset and give the inference as: 1. List down the features and their types (e.g. numeric, nominal) available in the dataset. 2. Create a histogram for each feature in the dataset to illustrate the feature distributions. 3. Create a boxplot for each feature in the dataset. 4. Compare distributions and identify outliers.
Objectives	1. Display feature variables and their types 2. Display histogram for each features 3. Display boxplot for each features and identify outliers
Outcomes	Students will be able to: 1. Display feature variables and their types of the dataset using Pandas 2. Plot histogram for each features using Seaborn and Matplotlib 3. Plot boxplot for each features using Seaborn and Matplotlib
S/W Requirement	OS – Linux Ubuntu 18 (64 bit) Packages - Sublime text editor, Python 3, Seaborn, Matplotlib, Pandas

Theory

What is Iris Flower Dataset?

- ✓ The Iris flower data set or Fisher's Iris data set is a multivariate data set introduced by the British statistician and biologist Ronald Fisher in his 1936 paper 'The use of multiple measurements in taxonomic problems' as an example of linear discriminant analysis.
- ✓ It is sometimes called Anderson's Iris data set because Edgar Anderson collected the data to quantify the morphologic variation of Iris flowers of three related species.
- ✓ The data set consists of 50 samples from each of three species of Iris (Iris Setosa, Iris Virginica and Iris Versicolor as shown in following figure).
- ✓ Four features were measured from each sample: the length and the width of the sepals and petals in centimeters.
- ✓ Based on the combination of these four features, Fisher developed a linear discriminant model to distinguish the species from each other.

Iris Versicolor

Iris Setosa

Iris Virginica

What are features and their types?

- ✓ Iris flower dataset has four features
  1) Sepal Length
  2) Sepal Width
  3) Petal Length
  4) Petal Width
  5) Variety

```
df=pd.read_csv("iris.data")
df=pd.read_csv("iris.data", header=-1)
column_name=["sepal length","sepal width","petal length","petal width","Iris Setosa"]
df.columns=column_name
df.head()
```

	sepal length	sepal width	petal length	petal width	Iris Setosa
0	5.1	3.5	1.4	0.2	Iris-setosa
1	4.9	3.0	1.4	0.2	Iris-setosa
2	4.7	3.2	1.3	0.2	Iris-setosa
3	4.6	3.1	1.5	0.2	Iris-setosa
4	5.0	3.6	1.4	0.2	Iris-setosa

If we see that all the feature values are numerical.

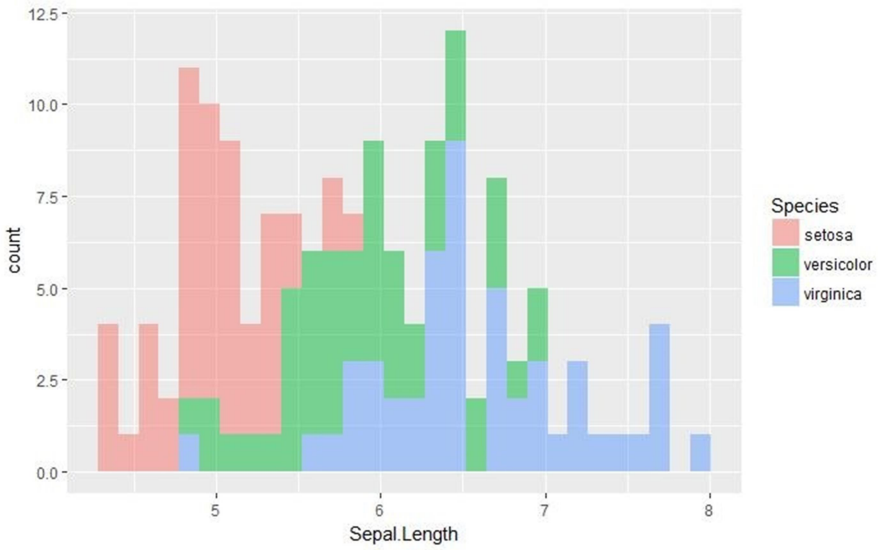
```
print(iris.info())
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 150 entries, 0 to 149
Data columns (total 5 columns):
sepal_length    150 non-null float64
sepal_width     150 non-null float64
petal_length    150 non-null float64
petal_width     150 non-null float64
species         150 non-null object
dtypes: float64(4), object(1)
memory usage: 5.9+ KB
None
```

Histogram

- ✓ Histograms represent the data distribution by forming bins along the range of the data and then drawing bars to show the number of observations that fall in each bin.
- ✓ Histograms are visualization tools that represent the distribution of a set of continuous data.
- ✓ In a histogram, the data is divided into a set of intervals or bins (usually on the x-axis) and the count of data points that fall into each bin corresponding to the height of the bar above that bin.
- ✓ These bins may or may not be equal in width but are adjacent (with no gaps).

```
require(ggplot2)

qplot(Sepal.Length, data=iris, geom='histogram', fill=Species, alpha=I(1/2))
```



- ✓ Similarly histogram of Sepal Width, Petal Length and Petal Width can be plot.

Boxplot

- ✓ Boxplot can be drawn as follows:

```
sns.boxplot(x="type",y="petal_length",data = iris)
plt.show()
```

- ✓ Similary boxplot of Sepal Length, Sepal Width and Petal Width can be drawn.

We see some outlier in Petal Length of Iris-Versicolor type.

Algorithm	1. Load the Iris flower Dataset into Pandas Dataframe 2. Display feature variables (columns)

	3. Display information of dataset and data types of variables 4. Plot overall histogram of feature variables 5. Plot Group wise histogram of feature variables 6. Plot overall boxplot of feature variables and identify outliers 7. Plot group wise boxplot of feature variables and identify outliers
Conclusion	In this assignment we are able to: 1. Display feature variables and information of Iris flower dataset 2. Plot overall and group wise histogram of feature variables 3. Plot overall and group wise boxplot of feature variables and identify outliers
Questions	1. What is histogram? 2. How to plot histogram? Give example? 3. Explain Iris flower dataset? 4. How to plot histogram of Petal Length of Setosa species? 5. What is boxplot? 6. How to plot boxplot? Give example? 7. How to plot boxplot of Petal Length of all the species? 8. How to plot histogram of all the features of Setosa species? 9. How to plot boxplot of all the features of Setosa species?

Assignment Number - 11	
Title & Problem Statement	Write a code in JAVA for a simple Word Count application that counts the number of occurrences of each word in a given input set using the Hadoop Map-Reduce framework on local-standalone set-up
Objectives	1. Understand the concepts of Hadoop Map-Reduce framework. 2. Word Count application that counts the number of occurrences of each word in a given input set 3. Perform Map and reduce function and finally will get result.
Outcomes	Students will be able to: 1. Perform map Reduce framework. 2. Take word as input and counts the number of occurrences of each word in a given input . 3. Display result of total occurrences of each word.
S/W Requirement	OS – Linux Ubuntu 18 (64 bit) Packages - Sublime text editor, Python 3, Seaborn, Matplotlib, Pandas

Theory

What is MapReduce in Hadoop?

- ✓ MapReduce is a processing technique and a program model for distributed computing based on java.
- ✓ The MapReduce algorithm contains two important tasks, namely Map and Reduce.
- ✓ Map takes a set of data and converts it into another set of data, where individual elements are broken down into tuples (key/value pairs). Secondly, reduce task, which takes the output from a map as an input and combines those data tuples into a smaller set of tuples.
- ✓ The input to each phase is **key-value** pairs. In addition, every programmer needs to specify two functions: **map function** and **reduce function**.
- ✓ The whole process goes through four phases of execution namely, splitting, mapping, shuffling, and reducing.

The Data goes through the following phases of MapReduce in Big Data

- ✓ **Input Splits:**

An input to a MapReduce in Big Data job is divided into fixed-size pieces called input splits Input split is a chunk of the input that is consumed by a single map

- ✓ **Mapping**

This is the very first phase in the execution of map-reduce program. In this phase data in each split is passed to a mapping function to produce output values. In our example, a job of mapping phase is to count a number of occurrences of each word from input splits (more details about input-split is given below) and prepare a list in the form of <word, frequency>

- ✓ **Shuffling**

This phase consumes the output of Mapping phase. Its task is to consolidate the relevant records from Mapping phase output. In our example, the same words are clubed together along with their respective frequency.

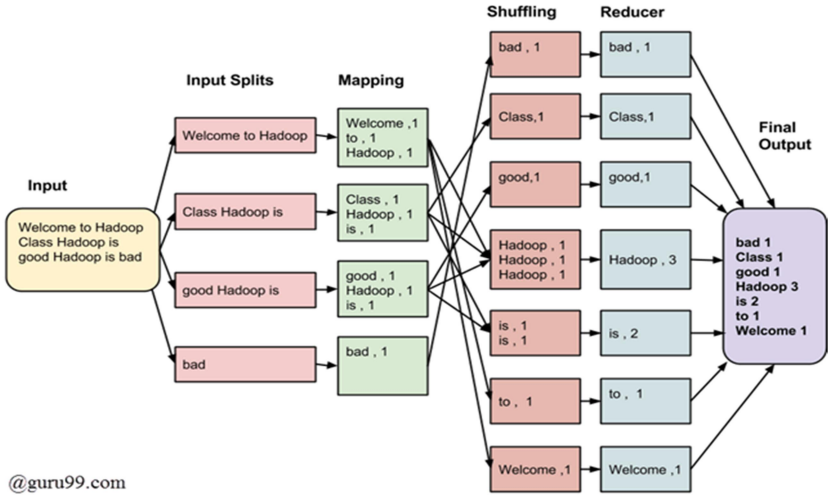
- ✓ **Reducing**

In this phase, output values from the Shuffling phase are aggregated. This phase combines values from Shuffling phase and returns a single output value. In short, this phase summarizes the complete dataset.

In our example, this phase aggregates the values from Shuffling phase i.e., calculates total occurrences of each word.

Map Reduce example

```
Welcome to Hadoop Class
Hadoop is good
Hadoop is bad
```



The final output of the MapReduce task is:

bad	1
Class	1
good	1
Hadoop	3
is	2
to	1
Welcome	1

Inputs and Outputs (Java Perspective)

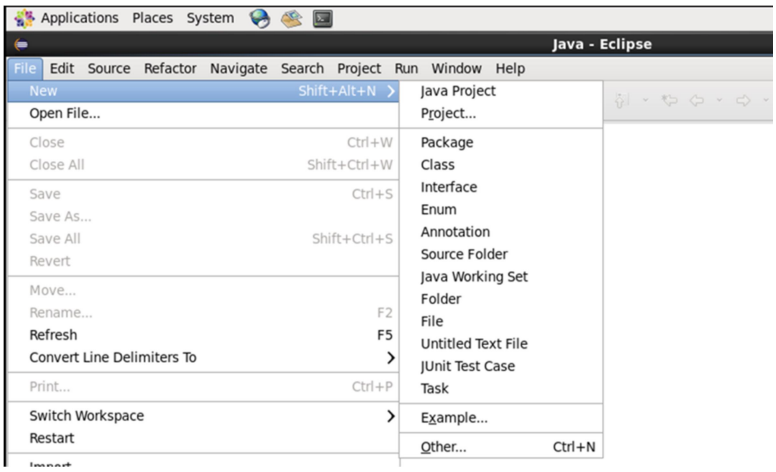
- ✓ The MapReduce framework operates on <key, value> pairs, that is, the framework views the input to the job as a set of <key, value> pairs and produces a set of <key, value> pairs as the output of the job, conceivably of different types.
- ✓ The key and the value classes should be in serialized manner by the framework and hence, need to implement the Writable interface. Additionally, the key classes have to implement the Writable-Comparable interface to facilitate sorting by the framework. Input and Output types of a MapReduce job – (Input) <k1, v1> → map → <k2, v2> → reduce → <k3, v3>(Output).

	Input	Output
Map	<k1, v1>	list (<k2, v2>)
Reduce	<k2, list(v2)>	list (<k3, v3>)

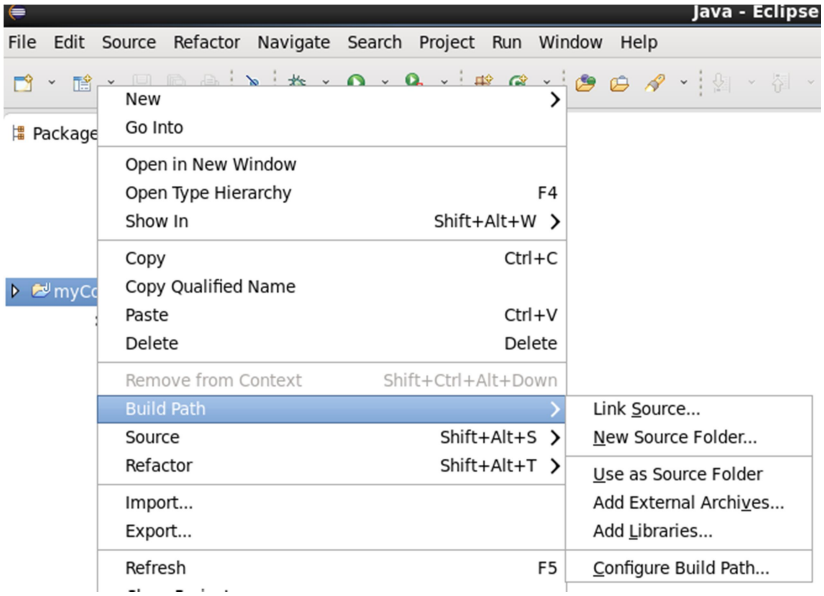
Algorithm

Step 1:

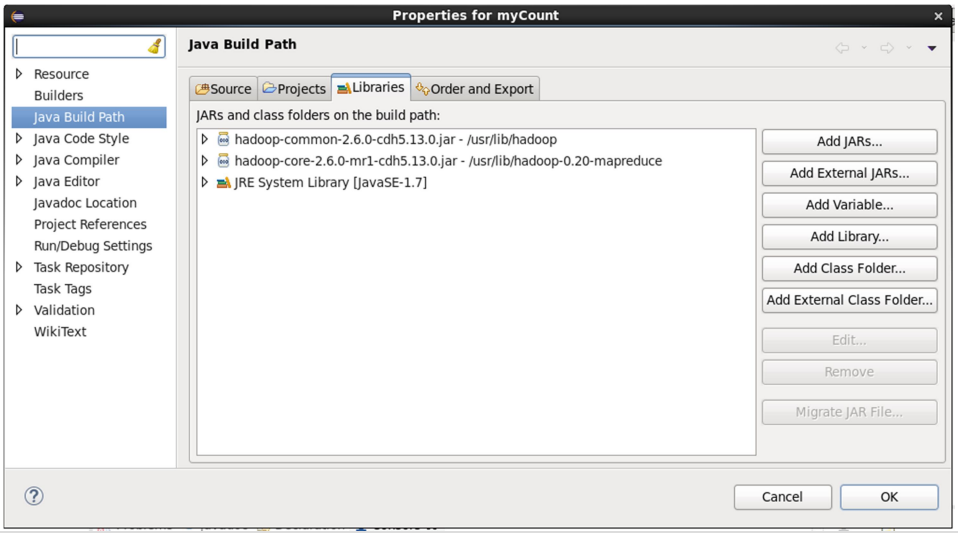
- First Open **Eclipse** -> then select **File** -> **New** -> **Java Project** ->Name it **WordCount** -> then **Finish**.



- **Step 2**
- Create Three Java Classes into the project. Name them **WCDriver**(having the main function), **WCMapper**, **WCReducer**.
- You have to include two Reference Libraries for that: Right Click on **Project** -> then select **Build Path**-> Click on **Configure Build Path**



- In the above figure, you can see the Add External JARs option on the Right Hand Side. Click on it and add the below mention files. You can find these files in */usr/lib/*
  1. /usr/lib/hadoop-0.20-mapreduce/hadoop-core-2.6.0-mr1-cdh5.13.0.jar
  2. /usr/lib/hadoop/hadoop-common-2.6.0-cdh5.13.0.jar



Step 3 :

Mapper Code: You have to copy paste this program into the WCMapper Java Class file.

Java

```
// Importing libraries
import java.io.IOException;
import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.LongWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapred.MapReduceBase;
import org.apache.hadoop.mapred.Mapper;
import org.apache.hadoop.mapred.OutputCollector;
import org.apache.hadoop.mapred.Reporter;
public class WCMapper extends MapReduceBase implements
Mapper<LongWritable, Text, Text, IntWritable>
{
// Map function
    public void map(LongWritable key, Text value, OutputCollecto
   intWritable> output, Reporter rep) throws IOException
    {
String line = value.toString();
```

```
// Splitting the line on spaces
        for (String word : line.split(" "))
        {
            if (word.length() > 0)
            {
          output.collect(new Text(word), new IntWritable(1));
            }
        }
    }
}
```


Step 4:

Reducer Code: You have to copy paste this program into the WCReducer Java Class file.

Java

```
// Importing libraries
import java.io.IOException;
import java.util.Iterator;
import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapred.MapReduceBase;
import org.apache.hadoop.mapred.OutputCollector;
import org.apache.hadoop.mapred.Reducer;
import org.apache.hadoop.mapred.Reporter;

public class WCReducerextends MapReduceBase implement
<Text, IntWritable, Text, IntWritable>
 {
// Reduce function
    public void reduce(Text key, Iterator<IntWritable
                    OutputCollector<Text, IntWritable>
                                Reporter rep) throws IOEx
    {

        int count = 0;

        // Counting the frequency of each words
        while (value.hasNext())
        {
            IntWritable i = value.next();
            count += i.get();
        }
```

```
        output.collect(key, new IntWritable(count));
    }
}
```

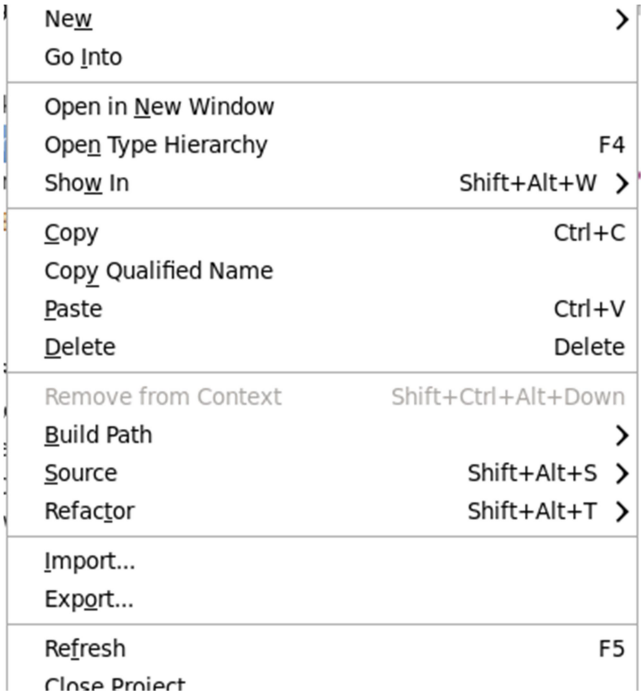
Step 5

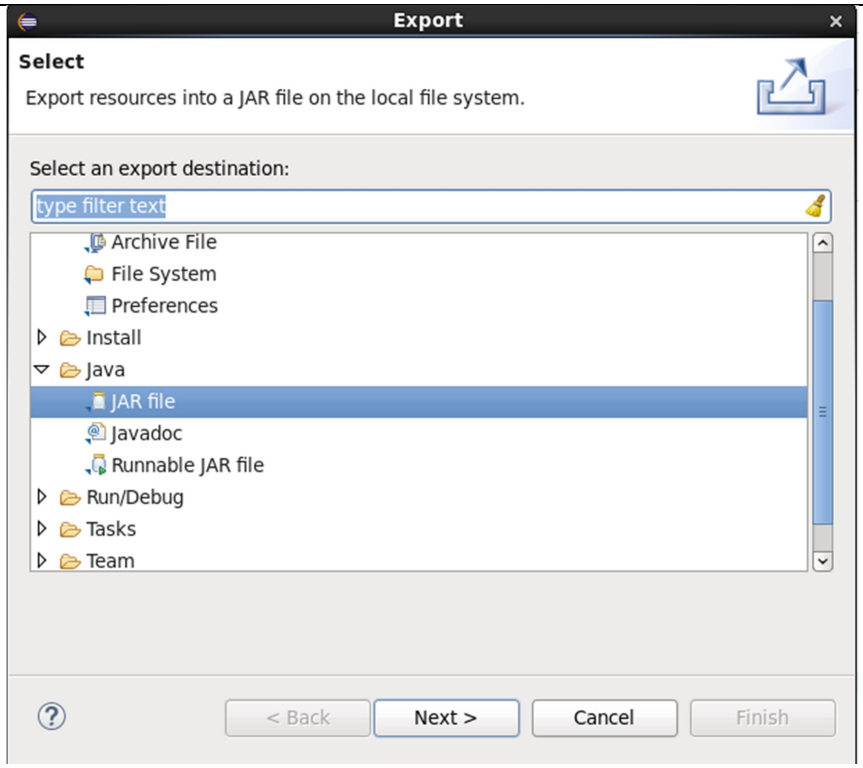
Driver Code: You have to copy paste this program into the WCDriver Java Class file.
Java

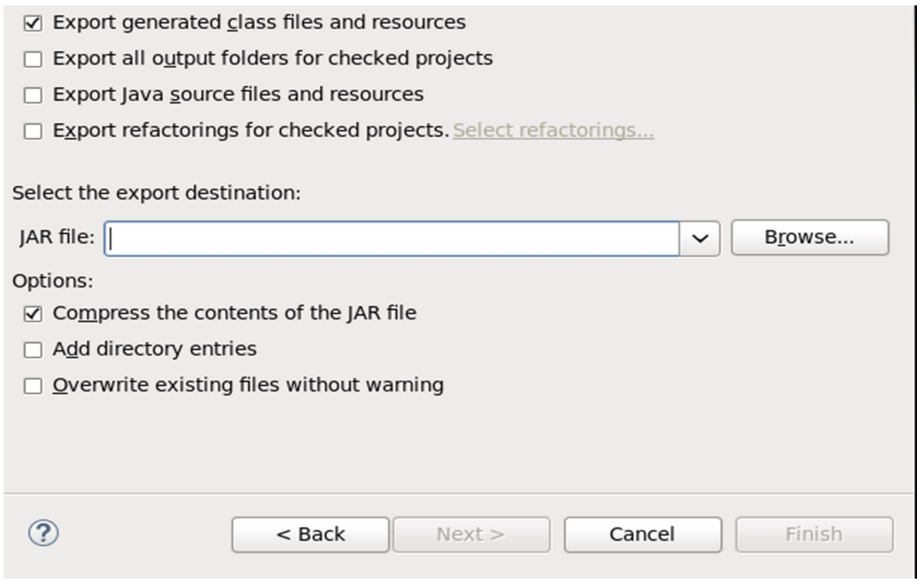
```
// Importing libraries
import java.io.IOException;
import org.apache.hadoop.conf.Configured;
import org.apache.hadoop.fs.Path;
import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapred.FileInputFormat;
import org.apache.hadoop.mapred.FileOutputFormat;
import org.apache.hadoop.mapred.JobClient;
import org.apache.hadoop.mapred.JobConf;
import org.apache.hadoop.util.Tool;
import org.apache.hadoop.util.ToolRunner;

public class WCDriver extends Configured implements T

    public int run(String args[]) throws IOException
    {
        if (args.length < 2)
        {
            System.out.println("Please give valid inp
            return -1;
        }

        JobConf conf = new JobConf(WCDriver.class);
        FileInputFormat.setInputPaths(conf, new
Path(args[0]));
        FileOutputFormat.setOutputPath(conf, new
Path(args[1]));
        conf.setMapperClass(WCMapper.class);
        conf.setReducerClass(WCReducer.class);
```

```
        conf.setMapOutputKeyClass(Text.class);
        conf.setMapOutputValueClass(IntWritable.class);
        conf.setOutputKeyClass(Text.class);
        conf.setOutputValueClass(IntWritable.class);
        JobClient.runJob(conf);
        return 0;
    }

    // Main Method
    public static void main(String args[]) throws Exception
    {
        int exitCode = ToolRunner.run(new WCDriver(), args);
        System.out.println(exitCode);
    }
}
```

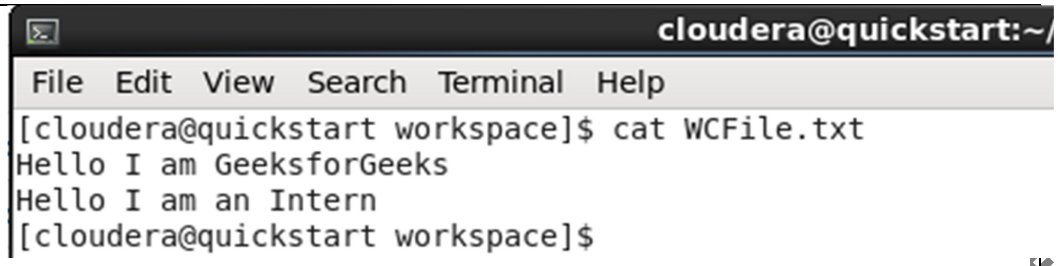
Step 6: Now you have to make a jar file. Right Click on **Project-> Click on Export-> Select export destination as Jar File-> Name the jar File**(WordCount.jar) -> **Click on next** -> at last **Click on Finish**. Now copy this file into the Workspace directory of Cloudera





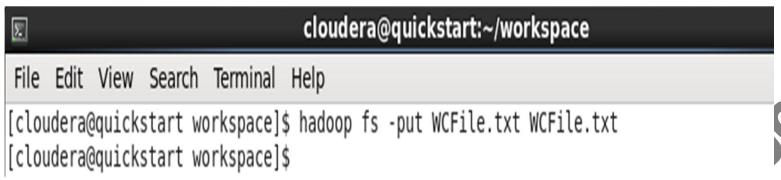


Open the terminal on CDH and change the directory to the workspace. You can do this by using "cd workspace/" command. Now, Create a text file(**WCFile.txt**) and move it to HDFS. For that open terminal and write this code(remember you should be in the same directory as jar file you have created just now).

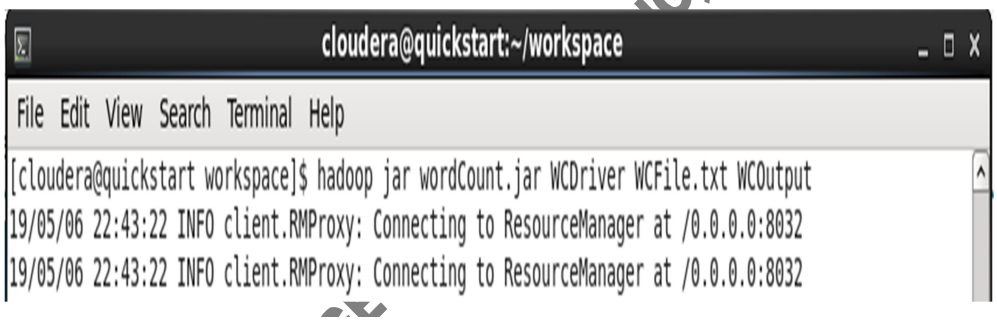


Now, run this command to copy the file input file into the HDFS.

```
hadoop fs -put WCFile.txt WCFile.txt
```

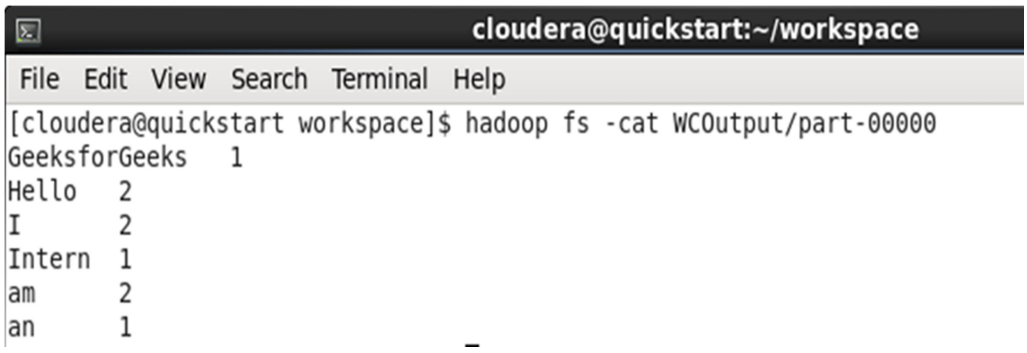


Now to run the jar file by writing the code as shown in the screenshot.



After Executing the code, you can see the result in *WCOutput* file or by writing following command on terminal.

```
hadoop fs -cat WCOutput/part-00000
```



Conclusion	In this assignment we are able to: 1. Counting the number of words using the Hadoop Map Reduce framework. 2. Execute WordCount Program in MapReduce using Cloudera Distribution Hadoop(CDH)
Questions	1) What is MapReduce? 2) What are the parameters of mappers and reducers? 3) What are the main components of MapReduce Job? 4) Which type of framework will supported by MapReduce? 5) What is Shuffling and Sorting in MapReduce? 6) Illustrate a simple example of the working of MapReduce. 7) Compare MapReduce and Spark 8) What is Shuffling and Sorting in MapReduce?

Assignment Number - 12	
Title & Problem Statement	**Big Data Analytics** Locate dataset (e.g. sample_weather.txt) for working on weather data which reads the text input files and finds average for temperature, dew point and wind speed.
Objectives	1. To read the input text file. 2. To find out average for temperature, dew point and wind speed.
Outcomes	Students will be able to: 1. To read the input text file. 2. To find out average for temperature, dew point and wind speed.
S/W Requirement	OS – Linux Ubuntu 18 (64 bit) Packages - Sublime text editor, Python 3, Seaborn, Matplotlib, Pandas

Theory

Analysis of Weather data using Pandas, Python, and Seaborn:

- ✓ The most recent post on this site was an analysis of how often people cycling to work actually get rained on in different cities around the world. You can check it out here.
- ✓ The analysis was completed using data from the underground weather website, Python, specifically the Pandas and Seaborn libraries.
- ✓ In this post, I will provide the Python code to replicate the work and analyze information for your own city.
- ✓ During the analysis, I used Python sublime to interactively explore and cleanse data; there's a simple setup if you elect to use something like the Anaconda Python distribution to install everything you need.
- ✓ If you want to skip data downloading and scraping, all of the data I used is available to download here.
- ✓ Read about dataset

Scraping Weather Data:

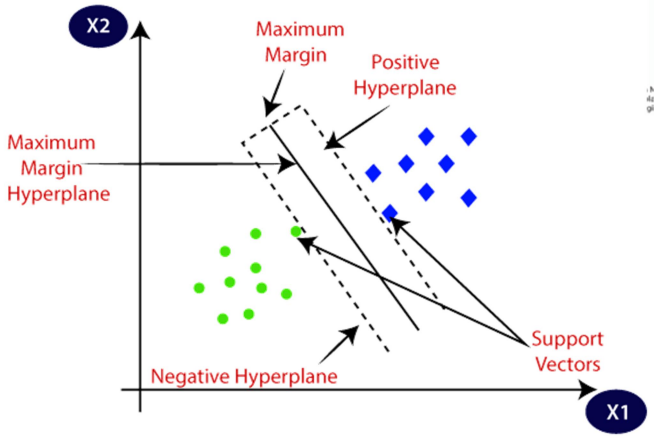
- ✓ Wunderground.com has a "Personal Weather Station (PWS)" network for which fantastic historical weather data is available – covering temperature, pressure, wind speed and direction, and of course rainfall in mm – all available on a per-minute level.
- ✓ Individual stations can be examined at specific URLS, for example here for station "IDUBLIND35".
- ✓ There's no official API for the PWS stations that I could see, but there is a very good API for forecast data.
- ✓ However, CSV format data with hourly rainfall, temperature, and pressure information can be downloaded from the website with some simple Python scripts.
- ✓ The hardest part here is to actually find stations that contain enough information for

your analysis – you'll need to switch to "yearly view" on the website to find stations that have been around more than a few months, and that record all of the information you want.

- ✓ If you relooking for temperature info – you're laughing, but precipitation records are more sparse.

Support Vector Machine Algorithm:

- ✓ Support Vector Machine or SVM is one of the most popular Supervised Learning algorithms, which is used for Classification as well as Regression problems. However, primarily, it is used for Classification problems in Machine Learning.
- ✓ The goal of the SVM algorithm is to create the best line or decision boundary that can segregate n-dimensional space into classes so that we can easily put the new data point in the correct category in the future. This best decision boundary is called a hyper plane.
- ✓ SVM chooses the extreme points/vectors that help in creating the hyper plane. These extreme cases are called as support vectors, and hence algorithm is termed as Support Vector Machine. Consider the below diagram in which there are two different categories that are classified using a decision boundary or hyper plane



Types of SVM

SVM can be of two types:

- ✓ **Linear SVM:** Linear SVM is used for linearly separable data, which means if a dataset can be classified into two classes by using a single straight line, then such data is termed as linearly separable data, and classifier is used called as Linear SVM classifier.
- ✓ **Non-linear SVM:** Non-Linear SVM is used for non-linearly separated data, which means if a dataset cannot be classified by using a straight line, then such data is termed as non-linear data and classifier used is called as Non-linear SVM classifier.

Algorithm

Step 1: Importing all packages

```
import numpy as np # linear algebra
import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv)

# Input data files are available in the read-only "../input/" directory
# For example, running this (by clicking run or pressing Shift+Enter) will list all files under t
he input directory

import os
for dirname, _, filenames in os.walk('/kaggle/input'):
  for filename in filenames:
    print(os.path.join(dirname, filename))

# You can write up to 20GB to the current directory (/kaggle/working/) that gets preserve
d as output when you create a version using "Save & Run All"
# You can also write temporary files to /kaggle/temp/, but they won't be saved outside of th
e current session
```

/kaggle/input/weathercsv/weather.csv

```
import pandas as pd                    # pandas library is used for data manipulation and
analysis
import numpy as np                     # numpy for numerical computing
import matplotlib.pyplot as plt        # matplotlib for plotting graphs and chart
import seaborn as sn
import warnings
warnings.filterwarnings(action='ignore')    # warning filter that ignores all
warnings to avoid unnecessary messages or errors during data analysis.
df1 = pd.read_csv("/kaggle/input/weathercsv/weather.csv")
df1.head()
df1.shape
df1.info()
df1.describe()
```

Step 2: To find any null values in given dataset

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

Step 3: Dropping the null values from dataset for given attribute as low null values found

```
df1 = df1.dropna()
df1.isnull().sum()

df1['RainTomorrow'].unique()
Y = df1.RainTomorrow
Y.head()
```

Step 4: Label Encoding

```
from sklearn import preprocessing
label_encoder = preprocessing.LabelEncoder()
df1['RainTomorrow']= label_encoder.fit_transform(df1['RainTomorrow'])
```

	df1['RainTomorrow'].unique() label_encoder = preprocessing.LabelEncoder() df1['WindGustDir']= label_encoder.fit_transform(df1['WindGustDir']) df1['WindGustDir'].unique() label_encoder = preprocessing.LabelEncoder() df1['RainToday']= label_encoder.fit_transform(df1['RainToday']) df1['RainToday'].unique() Step 5: Correlation and Heatmap hm = sn.heatmap(data = df1.corr(),annot=True,annot_kws={'size': 8}) *# displaying the plotted heatmap* sn.set (rc = {'figure.figsize':(12, 12)}) plt.show() X = df1.drop(['RainTomorrow','WindDir9am','WindDir3pm','WindSpeed9am'],axis='columns') X.head() Step 6: Splitting the data in test and train from sklearn.model_selection import train_test_split X_train,X_test,Y_train,Y_test = train_test_split(X,Y,test_size=0.2, random_state=10) Step 7: Using SVM classifier with linear kernel from sklearn import svm clf = svm.SVC(kernel='linear') *# Linear Kernel* clf.fit(X_train, Y_train) y_pred = clf.predict(X_test) from sklearn import metrics print("Accuracy:",metrics.accuracy_score(Y_test, y_pred)) Step 8: Using SVM classifier with Polynomial kernel model = svm.SVC(kernel='poly') model.fit(X_train, Y_train) y_pred = model.predict(X_test) from sklearn import metrics print("Accuracy:",metrics.accuracy_score(Y_test, y_pred))
Conclusion	In this assignment we are able to: 1. To read the input text file. 2. To find out average for temperature, dew point and wind speed.
Questions	1. Explain support vector machine 2. Analyze the weather dataset 3. What is use of linear kernel? 4. How to spit training and testing data? 5. Explain linear and Non- linear SVM. 6. How to find out average for temperature, dew point and wind speed?

Assignment Number – 13	
Title & Problem Statement	Write a simple program in SCALA Using Apache Spark Framework.
Objectives	1. Understand the Apache Spark Framework. 2. Understand what is SCALA. 3. Display output of simple SCALA Program
Outcomes	Students will be able to: 1. Understand the Apache Spark Framework. 2. Understand SCALA Language. 3. Run SCALA Program.
S/W Requirement	OS – Linux Ubuntu 18 (64 bit) Packages - Sublime text editor, Python 3, Spark,Hadoop

Theory

What is apache spark framework?

Scala has gained a lot of recognition for itself and is used by a large number of companies. Scala and Spark are being used at Facebook, Pinterest, NetFlix, Conviva, TripAdvisor for Big Data and Machine Learning applications.

What is Scala

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

A few more characteristics of Scala are:

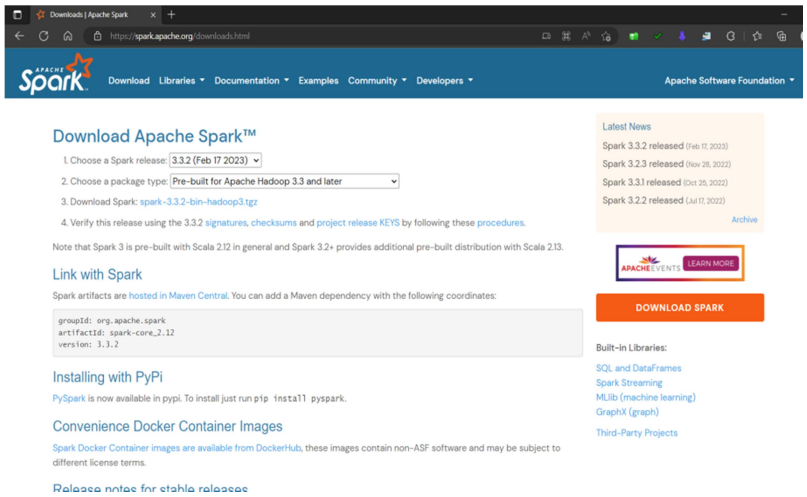
1. Scala is pure Object-Oriented programming language
2. Scala is a functional language
3. Scala is a compiler-based language (and not interpreted)
   - ✓ Scala can execute Java code
   - ✓ Scala is JVM based languages
   - ✓ Scala has thread-based executors

Steps to Download Apache Spark Framework

1.Download Apache Spark Framework

For that search official website of Apache Spark Framework or click on

Downloads | Apache Spark



1. Choose a Spark release: 3.3.2 (Feb 17 2023)
2. Choose a package type: Pre-built for Apache Hadoop 3.3 and later
3. Download Spark: spark-3.3.2-bin-hadoop3.tgz

2. Download latest Hadoop software and install

Algorithm

Things to note about Scala

- ✓ It is case sensitive
- ✓ If you are writing a program in Scala, you should save this program using ".scala"
- ✓ Scala execution starts from main() methods
- ✓ Any identifier name cannot begin with numbers. For example, variable name "123salary" is invalid.
- ✓ You can not use Scala reserved keywords for variable declarations or constant or any identifiers.

Variable declaration in Scala:

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

Declare using var:

```
var Var1:String="Ankit"
```

- ✓ In the above Scala statement, you declare a mutable variable called "Var1" which takes a string value.

- ✓ You can also write the above statement without specifying the type of variable.
- ✓ Scala will automatically identify it. For example:

```
var Var1="Gupta"
```

Declare using val:

```
val Var2:String="Ankit"
```

- ✓ In the above Scala statement, we have declared an immutable variable "Var2" which takes a string "Ankit".

Operations on variables:

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

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

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

Apply '+' operator

```
Var4+Var5
Output:
res1: Int = 5
```

Apply "==" operator

```
Var4==Var5
Output:
res2: Boolean = false
```

The if-else expression in Scala:

- ✓ In Scala, if-else expression is used for conditional statements.
- ✓ You can write one or more conditions inside "if".
- ✓ Let's declare a variable called "Var3" with a value 1 and then compare "Var3" using if-else expression.

```
var Var3 =1
if (Var3 ==1){
 println("True")}else{
 println("False")}
```

```
Output: True
```

Iteration in Scala:

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

```
for( a <- 1 to 10){
 println( "Value of a: " + a );
 }
Output:
Value of a: 1
Value of a: 2
Value of a: 3
Value of a: 4
Value of a: 5
Value of a: 6
Value of a: 7
Value of a: 8
Value of a: 9
Value of a: 10
```

Declare a simple function in Scala and call it by passing value:

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

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

Now let's pass a value 2 into mul2

```
mul2(2)
Output:
res9: Int = 20
```

Declaring Array in Scala:

```
var name:Array[String] = new Array[String](3)
```

or

```
var name = new Array[String](3)
Output:
name: Array[String] = Array(null, null, null)
```

- ✓ Here you have declared an array of Strings called "name" that can hold up to three elements.
- ✓ You can also assign values to "name" by using an index.

```
scala> name(0) = "jal"
scala> name(1) = "Faizy"
scala> name(2) = "Expert in deep learning"
```

Let's print contents of "name" array.

```
scala> name
res3: Array[String] = Array(jal, Faizy, Expert
in deep learning)
```

Accessing an array:

- ✓ You can access the element of an array by index.
- ✓ Lets access the first element of array "name".
- ✓ By giving index 0. Index in Scala starts from 0.

```
name(0)
Output:
res11: String = jal
```

Declaring List in Scala:

- ✓ You can define list simply by comma separated values inside the "List" method.

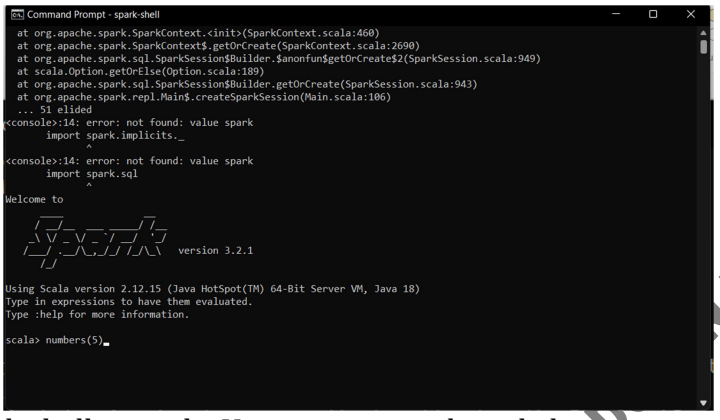
```
scala> val numbers = List (1,2,3,4,5,1,2,3,4,5)
numbers: List [Int] = List(1,2,3,4,5,1,2,3,4,5)
```

- ✓ You can also define multi-dimensional list in Scala. Let's define a two-dimensional list:

```
valnumber1=List (List (1,0,0), List (0,1,0),
List (0,0,1))
number1: List [List [Int]] =List (List (1,0,0),
List (0,1,0), List (0,0,1))
```

1. Open Command Prompt(Terminal).

2. You have to locate Apache Spark Folder where the Command Prompt's path has given or set Environmental Variable.

	3. Give path to spark folder then bin folder like: C:\User\Admin\Spark\bin
	4. Type spark-shell command to start Scala programming.
	 
	5.Now spark shell is ready. You can type scala code here.
	6.Sample code is here.
	`scala> var Var1:String="Ankit"`
	`Var1: String = Ankit`
	
	`scala> var Var1="Gupta"`
	`Var1: String = Gupta`
	
	`scala> val Var2:String="Ankit"`
	`Var2: String = Ankit`
	
	`scala> var Var4 = 2`
	`Var4: Int = 2`
	
	`scala> var Var5 = 3`
	`Var5: Int = 3`
	
	`scala> Var4+Var5`
	`res0: Int = 5`

Conclusion	In this assignment we are able to: 1. Understand the Apache Spark Framework and SCALA Program. 2. Run SCALA Program.
Questions	1. What is Apache Spark Framework 2. What is SCALA ? 3. What is SCALA Programming? 4. Is SCALA Similar to Object Oriented Programming Language? 5. What are the applications of SCALA Programming? 6. Is SCALA Scripting Language? 7. Explain how Scala is both Functional and Object-oriented Programming Language? 8. Write a few Frameworks of Scala? 9. Mention the Advantages of Scala? 10. Explain the Operators in Scala?