**PCET and NMVPM's**

**NUTAN MAHARASHTRA INSTITUTE OF ENGINEERING ANDTECHNOLOGY**

**Talegaon, Pune**

**Department of Computer Engineering**

# LAB MANUAL

# Data Science & Big Data Analytics Laboratory

**Subject Code: -310256**

***Prepared by,***

***Mrs.Rohini Hanchate***

**TE COMPUTER**

***Semester II Academic Year 2023-24***

# Savitribai Phule Pune University
## Third Year of Computer Engineering (2019 Course)
## 310256: Data Science and Big Data Analytics Laboratory

Home

| Teaching Scheme | Credit Scheme: | Examination Scheme and Marks |
|---|---|---|
| **Practical: 04 Hours/Week** | **02** | **Term work: 25 Marks**<br>**Oral: 50 Marks** |

**Companion Course:** Data Science and Big Data Analytics (310251)

**Course Objectives:**

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

**Course Outcomes:**

On completion of the course, learner will be able to

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

## Guidelines for Instructor's Manual

The instructor‘s manual is to be developed as a reference and hands-on resource. It should include prologue (about University/program/ institute/ department/foreword/ preface), curriculum of the course, conduction and Assessment guidelines, topics under consideration, concept, objectives, outcomes, set of typical applications/assignments/ guidelines, and references.

## Guidelines for Student's Laboratory Journal

The laboratory assignments are to be submitted by student in the form of journal. Journal consists of Certificate, table of contents, and handwritten write-up of each assignment (Title, Date of Completion, Objectives, Problem Statement, Software and Hardware requirements, Assessment grade/marks and assessor's sign, Theory- Concept in brief, algorithm, flowchart, test cases, Test Data Set(if applicable), mathematical model (if applicable), conclusion/analysis. Program codes with sample output of all performed assignments are to be submitted as softcopy. As a conscious effort and little contribution towards Green IT and environment awareness, attaching printed papers as part of write-ups and program listing to journal must be avoided. Use of DVD containing students programs maintained by Laboratory In-charge is highly encouraged. For reference one or two journals may be maintained with program prints in the Laboratory.

## Guidelines for Laboratory /Term Work Assessment

Continuous assessment of laboratory work should be based on overall performance of Laboratory assignments by a student. Each Laboratory assignment assessment will assign grade/marks based on parameters, such as timely completion, performance, innovation, efficient codes, punctuality and

## Guidelines for Practical Examination

Problem statements must be decided jointly by the internal examiner and external examiner. During practical assessment, maximum weightage should be given to satisfactory implementation of the problem statement. Relevant questions may be asked at the time of evaluation to test the student‘s understanding of the fundamentals, effective and efficient implementation. This will encourage, transparent evaluation and fair approach, and hence will not create any uncertainty or doubt in the minds of the students. So adhering to these principles will consummate our team efforts to the promising start of student's academics.

## Guidelines for Laboratory Conduction

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

Set of suggested assignment list is provided in groups- A and B. Each student must perform 14 assignments (10 from group A, 4 from group B), 2 mini project from Group C

Operating System recommended :- 64-bit Open source Linux or its derivative

Programming tools recommended: - JAVA/Python/R/Scala

## Virtual Laboratory:

- http://cse18- iiith.vlabs.ac.in/Introduction.html?domain=Computer%20Science
- http://vlabs.iitb.ac.in/vlabs-dev/labs/cglab/index.php

## Suggested List of Laboratory Experiments/Assignments

**(All assignments are compulsory)**

| Sr. No. | Group A : Data Science |
|---|---|
| 1. | **Data Wrangling I**<br>Perform the following operations using Python on any open source dataset (eg. data.csv)<br>1. Import all the required Python Libraries.<br>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).<br>3. Load the Dataset into pandas dataframe.<br>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.<br>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.<br>6. Turn categorical variables into quantitative variables in Python<br><br>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**<br>Perform the following operations using Python on any open source dataset (eg. data.csv)<br><br>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.<br>2. Scan all numeric variables for outliers. If there are outliers, use any of the suitable techniques to deal with them.<br>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.<br><br>Reason and document your approach properly. |

| | |
|---|---|
| 3. | **Basic Statistics - Measures of Central Tendencies and Variance**<br>Perform the following operations on any open source dataset (eg. data.csv)<br>1. Provide summary statistics (mean, median, minimum, maximum, standard deviation) for a dataset (age, income etc.) with numeric variables grouped by one of the qualitative (categorical) 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.<br>2. Write a Python program to display some basic statistical details like percentile, mean, standard deviation etc. of the species of 'Iris-setosa', 'Iris-versicolor' and 'Iris-versicolor' of iris.csv dataset.<br><br>Provide the codes with outputs and explain everything that you do in this step.<br>. |
| 4. | **Data Analytics I**<br>Create a Linear Regression Model using Python/R to predict home prices using Boston Housing Dataset (https://www.kaggle.com/c/boston-housing). The Boston Housing dataset contains information about various houses in Boston through different parameters. There are 506 samples and 14 feature variables in this dataset.<br><br>The objective is to predict the value of prices of the house using the given features. |
| 5. | **Data Analytics II**<br>1. Implement logistic regression using Python/R to perform classification on Social_Network_Ads.csv dataset<br>Compute Confusion matrix to find TP, FP, TN, FN, Accuracy, Error rate, Precision, Recall on the given dataset. |
| 6. | **Data Analytics III**<br>1. Implement Simple Naïve Bayes classification algorithm using Python/R on iris.csv dataset.<br>Compute Confusion matrix to find TP, FP, TN, FN, Accuracy, Error rate, Precision, Recall on the given dataset. |
| 7. | **Text Analytics**<br>1. Extract Sample document and apply following document preprocessing methods: Tokenization, POS Tagging, stop words removal, Stemming and Lemmatization.<br>Create representation of document by calculating Term Frequency and Inverse Document Frequency. |
| 8. | **Data Visualization I**<br>1. Use the inbuilt dataset 'titanic'. The dataset contains 891 rows and contains information about the passengers who boarded the unfortunate Titanic ship. Use the Seaborn library to see if we can find any patterns in the data.<br>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**<br>1. Use the inbuilt dataset 'titanic' as used in the above problem. Plot a box plot for distribution of age with respect to each gender along with the information about whether they survived or not. (Column names : 'sex' and 'age')<br>Write observations on the inference from the above statistics. |

| | |
|---|---|
| 10. | **Data Visualization III**<br>Download the Iris flower dataset or any other dataset into a DataFrame. (eg https://archive.ics.uci.edu/ml/datasets/Iris ). Scan the dataset and give the inference as:<br>1. How many features are there and what are their types (e.g., numeric, nominal)?<br>2. Create a histogram for each feature in the dataset to illustrate the feature distributions.<br>3. Create a boxplot for each feature in the dataset.<br>Compare distributions and identify outliers. |
| | **Group B- Big Data Analytics – JAVA/SCALA (Any three)** |
| 1. | 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. |
| 2. | Design a distributed application using MapReduce which processes a log file of a system. |
| 3. | Locate dataset (eg. sample_weather.txt) for working on weather data which reads the text input files and finds average for temperature, dew point and wind speed. |
| 4. | Write a simple program in SCALA using Apache Spark framework |
| | **Group C- Mini Projects/ Case Study – PYTHON/R (Any TWO Mini Project)** |
| 1. | 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. |
| 2. | Use the following dataset and classify tweets into positive and negative tweets.<br>https://www.kaggle.com/ruchi798/data-science-tweets |
| 3. | Develop a movie recommendation model using the scikit-learn library in python.<br><br>Refer dataset<br>https://github.com/rashida048/Some-NLP-Projects/blob/master/movie_dataset.csv |
| 4. | Use the following covid_vaccine_statewise.csv dataset and perform following analytics on the given dataset<br>https://www.kaggle.com/sudalairajkumar/covid19-in-india?select=covid_vaccine_statewise.csv<br>a. Describe the dataset<br>b. Number of persons state wise vaccinated for first dose in India<br>c. Number of persons state wise vaccinated for second dose in India<br>d. Number of Males vaccinated<br>d. Number of females vaccinated |
| 5. | Write a case study to process data driven for Digital Marketing **OR** Health care systems with Hadoop Ecosystem components as shown. (Mandatory)<br>● HDFS: Hadoop Distributed File System<br>● YARN: Yet Another Resource Negotiator<br>● MapReduce: Programming based Data Processing<br>● Spark: In-Memory data processing<br>● PIG, HIVE: Query based processing of data services<br>● HBase: NoSQL Database (Provides real-time reads and writes)<br>● Mahout, Spark MLLib: (Provides analytical tools) Machine Learning algorithm libraries<br>● Solar, Lucene: Searching and Indexing |

**Reference Books :**


1. Chirag Shah, "A Hands-On Introduction To Data Science", Cambridge UniversityPress,(2020), ISBN : ISBN 978-1-108-47244-9.
2. Python for Data Analysis by Wes McKinney published by O' Reilly media, ISBN : 978-1-449-31979-3.
3. Scikit-learn Cookbook , Trent hauk,Packt Publishing, ISBN: 9781787286382
4. R Kent Dybvig, ―the Scheme Programming Language‖, MIT Press, ISBN 978-0-262-51298-5.
5. Data Analytics with Hadoop, Jenny Kim, Benjamin Bengfort, OReilly Media, Inc.
6. Python Data Science Handbook by Jake VanderPlas https://tanthiamhuat.files.wordpress.com/2018/04/pythondatasciencehandbook.pdf
7. An Introduction to Statistical Learning by Gareth James https://www.ime.unicamp.br/~dias/Intoduction%20to%20Statistical%20Learning.pdf
8. Cay S Horstmann, ―Scala for the Impatient‖, Pearson, ISBN: 978-81-317-9605-4,
9. Scala Cookbook, Alvin Alexander, O'Reilly, SPD,ISBN: 978-93-5110-263-2


**References :**


- https://www.simplilearn.com/data-science-vs-big-data-vs-data-analytics-article
- https://hadoop.apache.org/docs/current/hadoop-mapreduce-client/hadoop-mapreduce-client- core/MapReduceTutorial.html
- https://www.edureka.co/blog/hadoop-ecosystem
- https://www.edureka.co/blog/mapreduce-tutorial/#mapreduce_word_count_example
- https://github.com/vasanth-mahendran/weather-data-hadoop
- https://spark.apache.org/docs/latest/quick-start.html#more-on-dataset-operations
- https://www.scala-lang.org/


**@The CO-PO Mapping Matrix**

| PO/CO | PO1 | PO2 | PO3 | PO4 | PO5 | PO6 | PO7 | PO8 | PO9 | PO10 | PO11 | PO12 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **CO1** | 2 | 2 | 2 | 2 | 2 | 2 | - | - | - | - | 3 | - |
| **CO2** | 2 | 2 | 2 | 2 | 3 | - | - | - | - | - | - | - |
| **CO3** | 2 | 2 | 2 | - | 2 | - | - | - | - | - | - | - |
| **CO4** | 2 | 2 | 2 | 2 | 2 | 2 | - | - | - | - | - | - |
| **CO5** | 2 | 2 | 2 | 2 | 2 | 2 | - | - | - | - | - | - |
| **CO6** | 2 | 2 | 2 | 2 | 2 | 2 | - | - | - | - | 3 | - |

## VISION OF THE INSTITUTE

"To be a recognizable institution for providing quality technical education & ensuring holistic development of students".

## VISION OF THE DEPARTMENT

"Imbibing quality Technical Education and overall development by endowing students with technical skills and competency in Computer engineering department"

## MISSION OF THE DEPARTMENT

1. To impart engineering knowledge and skills by adopting effective teaching learning processes.
2. To develop professional, entrepreneurial & research competencies encompassing continuous intellectual growth.
3. To produce educated students to exhibit societal and ethical responsibilities in working environment.

## PROGRAM EDUCATIONAL OBJECTIVES

***PEO1:*** To prepare globally competent graduates having Excellent fundamentals, domain knowledge, updated with modern technology to provide the effective solutions for engineering problems.

***PEO2:*** To prepare the graduates to work as a committed professional with strong professional ethics and values, sense of responsibilities, understanding of legal, safety, health, societal, cultural and environmental issues.

***PEO3:*** To prepare committed and motivated graduates with research attitude, lifelong learning, investigative approach, and multidisciplinary thinking.

***PEO4:*** To prepare the graduates with strong managerial and communication skills to work effectively as individual as well as in teams.

GENRAL INSTUCTIONS:

- Equipment in the lab is meant for the use of students. Students need to maintain a proper decorum in the computer lab. Students must use the equipment with care.
- Students are required to carry their reference materials, files and records with completed assignment while entering the lab.
- Students are supposed to occupy the systems allotted to them and are not supposed to talk or make noise in the lab.
- All the students should perform the given assignment individually.
- Lab can be used in free time/lunch hours by the students who need to use the systems should take prior

permission from the lab in-charge.

- All the Students are instructed to carry their identity cards when entering the lab.
- Lab files need to be submitted on or before date of submission.
- Students are not supposed to use pen drives, compact drives or any other storage devices in the lab.
- For Laboratory related updates and assignments students should refer to the notice board in the Lab.

WEEKLY PLAN:

| Expt. No. | Practical/Assignment Name | Problem Definition |
|---|---|---|
| **01** | 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 | To understand basic data wrangling methods |

| | | |
|---|---|---|
| | 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 | |
| **02** | Data Wrangling II Perform the following operations using Python on any open source dataset (eg. data.csv) 1. Scan all variables for missing values and inconsistencies. If there are missing values and/or inconsistencies, use any of the suitable techniques to deal with them. 2. Scan all numeric variables for outliers. If there are outliers, use any of the suitable techniques to deal with them. 3. Apply data transformations on at least one of the variables. The purpose of this transformation should be one of the following reasons: to change the scale for better understanding of the variable, to convert a non-linear relation into a linear one, or to decrease the skewness and convert the distribution into a normal distribution. Reason and document your approach properly. | To understand basic data wrangling methods for any data set |
| **03** | 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. | To learn basic statistic methods |
| **04** | 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. | To understand basic regression model & implement linear regression |
| **05** | 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. | To understand logistic regression model & implement it for given dataset |
| **06** | 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. | To learn simple naïve Bayes algorithm |

| | | |
|---|---|---|
| **07** | 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. | To understand text analytics methods. |
| **08** | 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 | To understand data visualization methods. |
| **09** | 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. | To understand data visualization methods |
| **10** | Data Visualization III Download the Iris flower dataset or any other dataset into a DataFrame. (eg https://archive.ics.uci.edu/ml/datasets/Iris ). Scan the dataset and give the inference as: 1. How many features are there and what are their types (e.g., numeric, nominal)? 2. Create a histogram for each feature in the dataset to illustrate the feature distributions. 3. Create a boxplot for each feature in the dataset. Compare distributions and identify outliers | To understand data visualization methods for given datasets |
| **Group B- Big Data Analytics – JAVA/SCALA** | | |
| **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 | To understand the Map reduce framework |
| **12** | Design a distributed application using MapReduce which processes a log file of a system. | To Understand how file processing is done in Map reduce |
| **13** | Locate dataset (eg. sample_weather.txt) for working on weather data which reads the text input files and finds average for temperature, dew point and wind speed. | Understand concepts of Hadoop |
| **14** | Group C- Mini Projects/ Case Study – PYTHON/R | To implement a small model which is based on data analytics model |

**Course Outcomes:**

| Course Outcome | Statement |
|---|---|
| | **At the end of the course, a student will be able to** |
| **310256.1** | Apply principles of data science for the analysis of real time problems. |
| **310256.2** | Implement data representation using statistical methods |
| **310256.3** | Implement and evaluate data analytics algorithms |
| **310256.4** | Perform text preprocessing |
| **310256.5** | Implement data visualization techniques |
| **310256.6** | Use cutting edge tools and technologies to analyze Big Data |

| Course Outcome | Program outcomes | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | **1** | **2** | **3** | **4** | **5** | **6** | **7** | **8** | **9** | **10** | **11** | **12** |
| **310256.1** | 2 | 2 | 2 | 2 | 2 | 2 | - | - | - | - | 3 | - |
| **310256.2** | 2 | 2 | 2 | 2 | 3 | - | - | - | - | - | - | - |
| **310256.3** | 2 | 2 | 2 | - | 2 | - | - | - | - | - | - | - |
| **310256.4** | 2 | 2 | 2 | 2 | 2 | 2 | - | - | - | - | - | - |
| **310256.5** | 2 | 2 | 2 | 2 | 2 | 2 | - | - | - | - | - | - |
| **310256.6** | 2 | 2 | 2 | 2 | 2 | 2 | - | - | - | - | 3 | - |

| Course Outcome | Program Specific Outcomes | |
|---|---|---|
| | **1** | **2** |
| **310256.1** | 1 | - |
| **310256.2** | **1** | **-** |
| **310256.3** | - | 2 |
| **310256.4** | **1** | **-** |
| **310256.5** | **2** | **-** |
| **310256.6** | **-** | **2** |

## Programme Outcomes: As prescribed by NBA

1. **Engineering knowledge:** Apply the knowledge of mathematics, science, engineering fundamentals, and an engineering specialization for the solution of complex engineering problems.

**2. Problem analysis:** Identify, formulate, research literature, and analyze complex engineering problems reaching substantiated conclusions using first principles of mathematics, natural sciences, and engineering sciences.

**3. Design/development of solutions:** Design solutions for complex engineering problems and design system components or processes that meet t h e specified needs with appropriate consideration for public health and safety, and cultural, societal, and environmental considerations.

**4. Conduct investigations of complex problems:** The problems that cannot be solved by straightforward application of knowledge, theories and techniques applicable to the engineering discipline.

**5. Modern tool usage**: Create, select, and apply appropriate techniques, resources, and modern engineering and IT tools, including prediction and modeling to complex engineering activities, with an understanding of the limitations.

**6. The engineer and society:** Apply reasoning informed by the contextual knowledge to assess societal,

health, safety, legal, and cultural issues and the consequent responsibilities relevant to the professional engineering practice.

**7. Environment and sustainability:** Understand the impact of the professional engineering solutions in societal and environmental contexts, and demonstrate the knowledge of, and need for sustainable development.

**8. Ethics:** Apply ethical principles and commit to professional ethics and responsibilities and norms of the engineering practice.

**9. Individual and team work:** Function effectively as an individual, and as a member or leader in diverse teams, and in multidisciplinary settings.

**10. Communication**: Communicate effectively on complex engineering activities with the engineering community and with t h e society at large, such as, being able to comprehend and write effective reports and design documentation, make effective presentations, and give and receive clear instructions.

**11. Project management and finance:** Demonstrate knowledge and understanding of t h e engineering and management principles and apply these to one's own work, as a member and leader in a team, to manage projects and in multidisciplinary environments.

**12. Life-long learning:** Recognize the need for, and have the preparation and ability to engage in independent and life-long learning in the broadest context of technological change.

## Programme Specific Outcomes (PSOs)

A graduate of the Computer Engineering Program will be able to-

1. Understand and apply recent technology trends and modern tools in diverse areas like Cybersecurity, High-Performance Computing, IoT, Web Design, AI/ML, Data Science, etc.

2. Identify and solve real-world problems with software engineering principles and cater to project-based learning through experiential and industrial exposure.

**EXAMINATION SCHEME**
Practical Exam: 25 Marks
Term Work: 50 Marks
Total: 75 Marks
Minimum Marks required: 20 Marks(TW) +12(Practical)

## PROCEDURE OF EVALUATION

Each practical/assignment shall be assessed continuously on the scale of 25 marks. The distribution of marks as follows.

| Sr. No | Evaluation Criteria | Marks for each Criteria | Rubrics |
|---|---|---|---|
| **1** | Timely Submission/Attendance | **10** | Punctuality reflects the work ethics. Students should reflect that work ethics by completing the lab assignments and reports in a timely manner without being reminded or warned. |
| **2** | Presentation/Journal presentation Marks | **10** | Students are expected to write the technical document (lab report) in their own words. The presentation of the contents in the lab report should be complete, unambiguous, clear, and understandable. The report should document approach/algorithm/design and code with proper explanation. |
| **3** | Understanding/Performance | **10** | Correctness and Robustness of the code is expected. The Learners should have an in-depth knowledge of the practical assignment performed. The learner should be able to explain methodology used for designing and developing the program/solution. He/she should clearly understand the purpose of the assignment and its outcome. |
| **4** | Assignment submissions | **10** | The learner should submit assignments He/she should clearly understand the purpose of the assignment and its outcome. |
| **5** | Oral | **10** | Students must be able Oral questions related to experiments Students Learner should be able to apply their basic skills of respective subject |

**Write-ups must include:**

**Group, Assignment No., Title, Objective, Problem Statement, Outcomes, Theory(in brief),Algorithm, Flowchart, Test Cases, Conclusion, FAQs:**

| Sr. No. | Group | Title of Assignment | CO | PO |
|---|---|---|---|---|
| 01 | A | 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 | CO1 | PO1 PO2 PO3 PO4 |
| 02 | A | Data Wrangling II Perform the following operations using Python on any open source dataset (eg. data.csv) 1. Scan all variables for missing values and inconsistencies. If there are missing values and/or inconsistencies, use any of the suitable techniques to deal with them. 2. Scan all numeric variables for outliers. If there are outliers, use any of the suitable techniques to deal with them. 3. Apply data transformations on at least one of the variables. The purpose of this transformation should be one of the following reasons: to change the scale for better understanding of the variable, to convert a non-linear relation into a linear one, or to decrease the skewness and convert the distribution into a normal distribution. Reason and document your approach properly. | CO1 CO2 | PO1 PO2 PO4 PO5 PO6 |
| 03 | A | 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. | CO1 CO2 | |
| 04 | A | 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 | CO1 CO2 | PO1 PO2 PO3 PO4 |

| | | | | |
|---|---|---|---|---|
| | | of the house using the given features. | | |
| **05** | **A** | 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. | **CO1 CO2 CO3** | **PO1 PO2 PO4 PO11** |
| **06** | **A** | 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. | **CO1 CO2 CO3** | **PO1 PO2 PO3 PO4 PO11** |
| **07** | **A** | 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. | **CO3 CO4** | **PO1 PO2 PO3 PO5** |
| **08** | **A** | 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 | **CO3 CO4 CO5** | **PO3 PO5 PO6** |
| **09** | **A** | 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. | **CO3 CO4 CO5** | **PO5 PO6 PO3** |
| **10** | **A** | Data Visualization III Download the Iris flower dataset or any other dataset into a DataFrame. (eg https://archive.ics.uci.edu/ml/datasets/Iris ). Scan the dataset and give the inference as: 1. How many features are there and what are their types (e.g., numeric, nominal)? 2. Create a histogram for each feature in the dataset to illustrate the feature distributions. 3. Create a boxplot for each feature in the dataset. Compare distributions and identify outliers | **CO3 CO5** | **PO2 PO4** |
| **11** | **B** | 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 | **CO1 CO6** | **PO3 PO5 PO11** |
| **12** | **B** | Design a distributed application using MapReduce which processes a log file of a system. | **CO1 CO6** | **PO2 PO3 PO4** |
| **13** | **B** | Locate dataset (eg. sample_weather.txt) for working on weather data which reads the text input files and finds average for temperature, dew point and wind speed. | **CO1 CO6** | **PO3 PO4 PO5** |
| **14** | **C** | Group C- Mini Projects/ Case Study – PYTHON/R | **CO6** | **PO1 - PO6** |

**AIM:** To Perform Data Wrangling I : implementing the following operations using Python.

**OBJECTIVES:**

- To understand principles of data science for the analysis of real time problems
- To Study the data wrangling
- Implement data wrangling methods

**PROBLEM STATMENT:** 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

**OUTCOMES:**

**CO1**: Apply principles of data science for the analysis of real time problems.

**SOFTWARE & HARDWARE REQUIREMENTS:**

- Software: python 3.8.1/Jupiter
- Hardware: windows 7 / any Open source Linux operating system.

**THEORY:**

Python language is one of the most trending programming languages as it is dynamic than others. Python is a simple high-level and an open-source language used for general-purpose programming. It has many open-source libraries and Pandas is one of them. Pandas is a powerful, fast, flexible open-source library used for data analysis and manipulations of data frames/datasets. Pandas can be used to read and write data in a dataset of different formats like CSV(comma separated values), txt, xls(Microsoft Excel) etc.

In this post, you will learn about various features of Pandas in Python and how to use it to practice.

Prerequisites: Basic knowledge about coding in Python.

Installation:

So if you are new to practice Pandas, then firstly you should install Pandas on your system.

Go to Command Prompt and run it as administrator. Make sure you are connected with an internet connection to download and install it on your system.

Then type “pip install pandas“, then press Enter key.

Download the Dataset “Iris.csv” Iris dataset is the Hello World for the Data Science, so if you have started your career in Data Science and Machine Learning you will be practicing basic ML algorithms

on this famous dataset. Iris dataset contains five columns such as Petal Length, Petal Width, Sepal Length, Sepal Width and Species Type.
Iris is a flowering plant, the researchers have measured various features of the different iris flowers and recorded digitally.

| Id | SepalLengthCm | SepalWidthCm | PetalLengthCm | PetalWidthCm | Species |
|---|---|---|---|---|---|
| 1 | 5.1 | 3.5 | 1.4 | 0.2 | Iris-setosa |
| 2 | 4.9 | 3 | 1.4 | 0.2 | Iris-setosa |
| 3 | 4.7 | 3.2 | 1.3 | 0.2 | Iris-setosa |
| 4 | 4.6 | 3.1 | 1.5 | 0.2 | Iris-setosa |
| 5 | 5 | 3.6 | 1.4 | 0.2 | Iris-setosa |
| 6 | 5.4 | 3.9 | 1.7 | 0.4 | Iris-setosa |
| 7 | 4.6 | 3.4 | 1.4 | 0.3 | Iris-setosa |
| 8 | 5 | 3.4 | 1.5 | 0.2 | Iris-setosa |
| 9 | 4.4 | 2.9 | 1.4 | 0.2 | Iris-setosa |
| 10 | 4.9 | 3.1 | 1.5 | 0.1 | Iris-setosa |
| 11 | 5.4 | 3.7 | 1.5 | 0.2 | Iris-setosa |
| 12 | 4.8 | 3.4 | 1.6 | 0.2 | Iris-setosa |
| 13 | 4.8 | 3 | 1.4 | 0.1 | Iris-setosa |
| 14 | 4.3 | 3 | 1.1 | 0.1 | Iris-setosa |
| 15 | 5.8 | 4 | 1.2 | 0.2 | Iris-setosa |
| 16 | 5.7 | 4.4 | 1.5 | 0.4 | Iris-setosa |
| 17 | 5.4 | 3.9 | 1.3 | 0.4 | Iris-setosa |
| 18 | 5.1 | 3.5 | 1.4 | 0.3 | Iris-setosa |
| 19 | 5.7 | 3.8 | 1.7 | 0.3 | Iris-setosa |

**Displaying up the top rows of the dataset with their columns**
The function head() will display the top rows of the dataset, the default value of this function is 5, that is it will show top 5 rows when no argument is given to it.
data.head(10)

| | Id | SepalLengthCm | SepalWidthCm | PetalLengthCm | PetalWidthCm | Species |
|---|---|---|---|---|---|---|
| 0 | 1 | 5.1 | 3.5 | 1.4 | 0.2 | Iris-setosa |
| 1 | 2 | 4.9 | 3.0 | 1.4 | 0.2 | Iris-setosa |
| 2 | 3 | 4.7 | 3.2 | 1.3 | 0.2 | Iris-setosa |
| 3 | 4 | 4.6 | 3.1 | 1.5 | 0.2 | Iris-setosa |
| 4 | 5 | 5.0 | 3.6 | 1.4 | 0.2 | Iris-setosa |

**Displaying the number of rows randomly**.
In sample() function, it will also display the rows according to arguments given, but it will display the rows randomly.
data.sample(10)

| | Id | SepalLengthCm | SepalWidthCm | PetalLengthCm | PetalWidthCm | Species |
|---|---|---|---|---|---|---|
| **108** | 109 | 6.7 | 2.5 | 5.8 | 1.8 | Iris-virginica |
| **139** | 140 | 6.9 | 3.1 | 5.4 | 2.1 | Iris-virginica |
| **10** | 11 | 5.4 | 3.7 | 1.5 | 0.2 | Iris-setosa |
| **4** | 5 | 5.0 | 3.6 | 1.4 | 0.2 | Iris-setosa |
| **132** | 133 | 6.4 | 2.8 | 5.6 | 2.2 | Iris-virginica |
| **99** | 100 | 5.7 | 2.8 | 4.1 | 1.3 | Iris-versicolor |
| **140** | 141 | 6.7 | 3.1 | 5.6 | 2.4 | Iris-virginica |
| **1** | 2 | 4.9 | 3.0 | 1.4 | 0.2 | Iris-setosa |
| **107** | 108 | 7.3 | 2.9 | 6.3 | 1.8 | Iris-virginica |
| **42** | 43 | 4.4 | 3.2 | 1.3 | 0.2 | Iris-setosa |

Displaying the number of columns and names of the columns. The column() function prints all the columns of the dataset in a list form.
data.columns

**Displaying the shape of the dataset.**
The shape of the dataset means to print the total number of rows or entries and the total number of columns or features of that particular dataset.

data.shape

**Slicing the rows.**
Slicing means if you want to print or work upon a particular group of lines that is from 10th row to 20th row.

print(data[10:21])
# it will print the rows from 10 to 20.

# you can also save it in a variable for further use in analysis
sliced_data=data[10:21]
print(sliced_data)

| | Id | SepalLengthCm | SepalWidthCm | PetalLengthCm | PetalWidthCm | Species |
|---|---|---|---|---|---|---|
| 10 | 11 | 5.4 | 3.7 | 1.5 | 0.2 | Iris-setosa |
| 11 | 12 | 4.8 | 3.4 | 1.6 | 0.2 | Iris-setosa |
| 12 | 13 | 4.8 | 3.0 | 1.4 | 0.1 | Iris-setosa |
| 13 | 14 | 4.3 | 3.0 | 1.1 | 0.1 | Iris-setosa |
| 14 | 15 | 5.8 | 4.0 | 1.2 | 0.2 | Iris-setosa |
| 15 | 16 | 5.7 | 4.4 | 1.5 | 0.4 | Iris-setosa |
| 16 | 17 | 5.4 | 3.9 | 1.3 | 0.4 | Iris-setosa |
| 17 | 18 | 5.1 | 3.5 | 1.4 | 0.3 | Iris-setosa |
| 18 | 19 | 5.7 | 3.8 | 1.7 | 0.3 | Iris-setosa |
| 19 | 20 | 5.1 | 3.8 | 1.5 | 0.3 | Iris-setosa |
| 20 | 21 | 5.4 | 3.4 | 1.7 | 0.2 | Iris-setosa |

Displaying only specific columns.
In any dataset, it is sometimes needed to work upon only specific features or columns, so we can do this by the following code.

```
specific_data=data[["Id","Species"]]
#data[["column_name1","column_name2","column_name3"]]

#now we will print the first 10 columns of the specific_data dataframe.
print(specific_data.head(10))
```

| | Id | Species |
|---|---|---|
| 0 | 1 | Iris-setosa |
| 1 | 2 | Iris-setosa |
| 2 | 3 | Iris-setosa |
| 3 | 4 | Iris-setosa |
| 4 | 5 | Iris-setosa |
| 5 | 6 | Iris-setosa |
| 6 | 7 | Iris-setosa |
| 7 | 8 | Iris-setosa |
| 8 | 9 | Iris-setosa |
| 9 | 10 | Iris-setosa |

**CONCLUSION:-**Hence, we have demonstrate Data Wrangling methods.

**ASSIGNMENT 02**

**AIM:** To perform Data Wrangling II Perform the following operations using Python on any open source dataset.

**OBJECTIVES:**

- To understand principles of data science for the analysis of real time problems.
- To develop in depth understanding and implementation of the key technologies in data science and big data analytics

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

**OUTCOMES:**

**CO1:** Apply principles of data science for the analysis of real time problems.

**SOFTWARE & HARDWARE REQUIREMENTS: Jupiter/ipython 3.8.1**

**THEORY:**

Identification and Handling of Null Values Missing Data can occur when no information is provided for one or more items or for a whole unit. Missing Data is a very big problem in real-life scenarios. Missing Data can also refer to as NA(Not Available) values in pandas. In DataFrame sometimes many datasets simply arrive with missing data, either because it exists and was not collected or it never existed. For Example, Suppose different users being surveyed may choose not to share their income, some users may choose not to share the address in this way many datasets went missing. In Pandas missing data is represented by two value: 1. None: None is a Python singleton object that is often used for missing data in Python code. 2. NaN : NaN (an acronym for Not a Number), is a special floating-point value recognized by all systems that use the standard IEEE floating-point representation

Pandas treat None and NaN as essentially interchangeable for indicating missing or null values. To facilitate this convention, there are several useful functions for detecting, removing, and replacing null values in Pandas DataFrame : ● isnull() ● notnull() ● dropna() ● fillna() ● replace()

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

Algorithm: Step 1 : Import pandas and numpy in order to check missing values in Pandas DataFrame import pandas as pd import numpy as np

Step 2: Load the dataset in dataframe object df df=pd.read_csv("/content/StudentsPerformanceTest1.csv")

Step 3: Display the data frame df

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

Step 5: To create a series true for NaN values for specific columns. for example math score in dataset and display data with only math score as NaN series = pd.isnull(df["math score"]) df[series]

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

Algorithm: Step 1 : Import pandas and numpy in order to check missing values in Pandas DataFrame import pandas as pd import numpy as np

Step 2: Load the dataset in dataframe object df df=pd.read_csv("/content/StudentsPerformanceTest1.csv")

Step 3: Display the data frame df

Label Encoding or One Hot Encoding. from sklearn.preprocessing import LabelEncoder le = LabelEncoder() df['gender'] = le.fit_transform(df['gender']) newdf=df df

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

● For replacing null values with NaN missing_values = ["Na", "na"]

df = pd.read_csv("StudentsPerformanceTest1.csv", na_values = missing_values) df

Filling null values with a single value

Step 1 : Import pandas and numpy in order to check missing values in Pandas DataFrame import pandas as pd import numpy as np

Step 2: Load the dataset in dataframe object df df=pd.read_csv("/content/StudentsPerformanceTest1.csv")

Step 3: Display the data frame df Step 4: filling missing value using fillna() ndf=df ndf.fillna(0)

Deleting null values using dropna() method

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

1. Dropping rows with at least 1 null value

2. Dropping rows if all values in that row are missing

3. Dropping columns with at least 1 null value.

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

Algorithm: Step 1 : Import pandas and numpy in order to check missing values in Pandas DataFrame import pandas as pd import numpy as np Step 2: Load the dataset in dataframe object df

df=pd.read_csv("/content/StudentsPerformanceTest1.csv") Step 3: Display the data frame df Step 4:To drop rows with at least 1 null value ndf.dropna() .

Detecting outliers using Scatterplot:

It is used when you have paired numerical data, or when your dependent variable

has multiple values for each reading independent variable, or when trying to determine

the relationship between the two variables. In the process of utilizing the scatter plot, one

can also use it for outlier detection.

To plot the scatter plot one requires two variables that are somehow related to

each other. So here Placement score and Placement count features are used.

Algorithm:

Step 1 : Import pandas , numpy and matplotlib libraries

import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

Step 2: Load the dataset in dataframe object df

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

Step 3: Display the data frame

df

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

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

Labels to the axis can be assigned (Optional)

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

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

**ASSIGNMENT 03**

**AIM:** To perform Basic Statistics - Measures of Central Tendencies and Variance Perform the following operations on any open source dataset

**OBJECTIVES:**

- To understand principles of data science for the analysis of real time problems.
- To develop in depth understanding and implementation of the key technologies in data science and big data analytics

**PROBLEM STATMENT:** 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.

**OUTCOMES:**

**CO1:** Apply principles of data science for the analysis of real time problems.
**CO2:** Implement data representation using statistical methods

**SOFTWARE & HARDWARE REQUIREMENTS: Jupiter/ipython 3.8.1**

**THEORY:**

Measures of Center

Mean

The arithmetic mean of a variable, often called the average, is computed by adding up all the values and dividing by the total number of values.

The population mean is represented by the Greek letter μ (mu). The sample mean is represented by $\bar{x}$(x-bar). The sample mean is usually the best, unbiased estimate of the population mean. However, the mean is influenced by extreme values (outliers) and may not be the best measure of center with strongly skewed data. The following equations compute the population mean and sample mean.

$$\mu = \frac{\sum x_i}{N}$$

$$\bar{x} = \frac{\sum x_i}{n}$$

where *x*i is an element in the data set, *N* is the number of elements in the population, and *n* is the number of elements in the sample data set.

Median

The median of a variable is the middle value of the data set when the data are sorted in order from least to greatest. It splits the data into two equal halves with 50% of the data below the median and 50% above the median. The median is resistant to the influence of outliers, and may be a better measure of center with strongly skewed data.

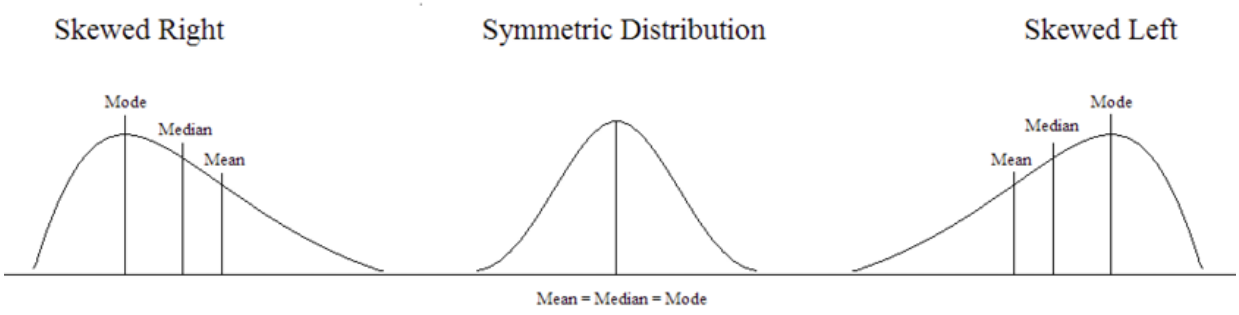
The calculation of the median depends on the number of observations in the data set.

To calculate the median with an odd number of values (*n* is odd), first sort the data from smallest to largest.

Mode

The mode is the most frequently occurring value and is commonly used with qualitative data as the values are categorical. Categorical data cannot be added, subtracted, multiplied or divided, so the mean and median cannot be computed. The mode is less commonly used with quantitative data as a measure of center. Sometimes each value occurs only once and the mode will not be meaningful.

Understanding the relationship between the mean and median is important. It gives us insight into the distribution of the variable. For example, if the distribution is skewed right (positively skewed), the mean will increase to account for the few larger observations that pull the distribution to the right. The median will be less affected by these extreme large values, so in this situation, the mean will be larger than the median. In a symmetric distribution, the mean, median, and mode will all be similar in value. If the distribution is skewed left (negatively skewed), the mean will decrease to account for the few smaller observations that pull the distribution to the left. Again, the median will be less affected by these extreme small observations, and in this situation, the mean will be less than the median.



Measures of Dispersion

Measures of center look at the average or middle values of a data set. Measures of dispersion look at the spread or variation of the data. Variation refers to the amount that the values vary among themselves. Values in a data set that are relatively close to each other have lower measures of variation. Values that are spread farther apart have higher measures of variation.

Examine the two histograms below. Both groups have the same mean weight, but the values of Group A are more spread out compared to the values in Group B. Both groups have an average weight of 267 lb. but the weights of Group A are more variable.

Range

The range of a variable is the largest value minus the smallest value. It is the simplest measure and uses only these two values in a quantitative data set.

Variance

The variance uses the difference between each value and its arithmetic mean. The differences are squared to deal with positive and negative differences. The sample variance (s2) is an unbiased estimator of the population variance (**σ**2), with n-1 degrees of freedom.

$$\frac{\sum (x_i - \mu)^2}{N} \qquad s2 = \frac{\sum (x_i - \bar{x})^2}{n-1} = \frac{\sum x_i^2 - \frac{\left(\sum x_i\right)^2}{n}}{n-1}$$

Standard Deviation

The standard deviation is the square root of the variance (both population and sample). While the sample variance is the positive, unbiased estimator for the population variance, the units for the variance are squared. The standard deviation is a common method for numerically describing the distribution of a variable. The population standard deviation is σ (sigma) and sample standard deviation is s.

Population standard deviation Sample standard deviation

$$\sigma = \sqrt{\sigma^2} \qquad s = \sqrt{s^2}$$

**CONCLUSION:-** In this way we have explored the functions of the python library for Basic Statistics - Measures of Central Tendencies

**ASSIGNMENT 02**

**AIM:** 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.

**OBJECTIVES:**

- To understand principles of data science for the analysis of real time problems.
- Students should be able to data analysis using liner regression using Python for any open source dataset

**PROBLEM STATMENT:** 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.

**OUTCOMES:**

**CO1:** Apply principles of data science for the analysis of real time problems.
**CO2:** Implement data representation using statistical methods

**SOFTWARE & HARDWARE REQUIREMENTS: Jupiter/ipython 3.8.1**

**THEORY:**

**Contents for Theory:**

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

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

actual and predicted values.

- It is shown as an equation of line like :

  Y = m*X + b + e

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

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

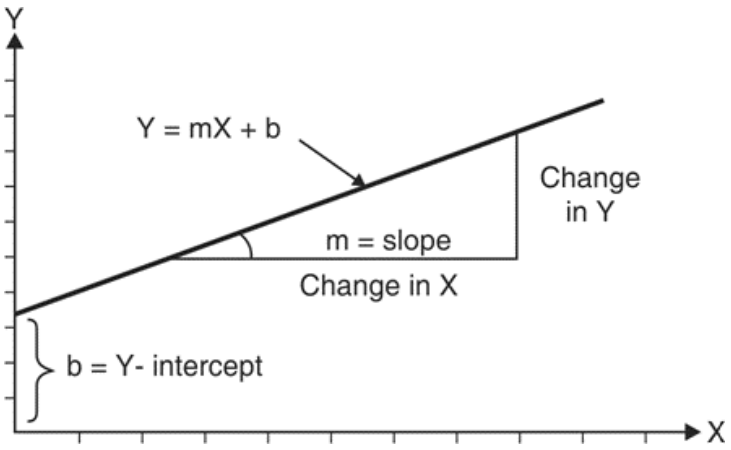
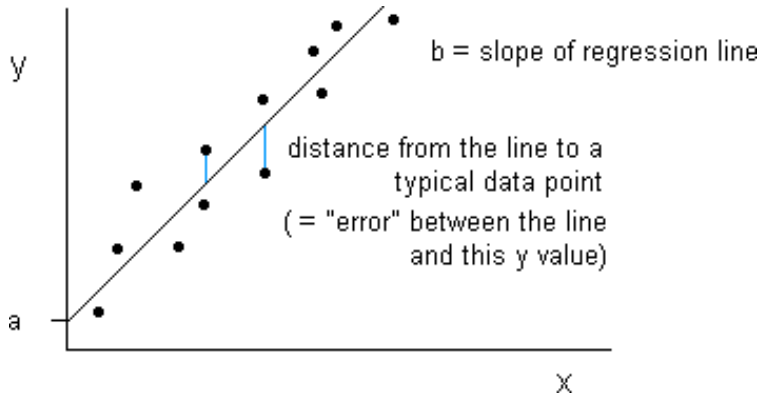



Fig. 1: geometry of linear regression

- Fig. 2 shown below is about the relation between weight (in Kg) and height (in cm), a linear relation. It is an approach of studying in a statistical manner to summarise and learn the relationships among continuous (quantitative) variables.
- Here a variable, denoted by ‘x’ is considered as the predictor, explanatory, or independent variable.

- Another variable, denoted ‘y’, is considered as the response, outcome, or dependent variable. While "predictor" and "response" used to refer to these variables.
- Simple linear regression technique concerned with the study of only one predictor variable.

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




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

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

$Y=a + bX + cX2$

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

**2. Least Square Method for Linear Regression**

- Linear Regression involves establishing linear relationships between dependent and independent variables. Such a relationship is portrayed in the form of an equation also known as the linear model.

- A simple linear model is the one which involves only one dependent and one independent variable. Regression Models are usually denoted in Matrix Notations.
- However, for a simple univariate linear model, it can be denoted by the regression equation

$$\hat{y} = \beta_0 + \beta_1 x \quad (1)$$

where $\hat{y}$ is the dependent or the response variable

$x$ is the independent or the input variable

$\beta_0$ is the value of y when x=0 or the y intercept

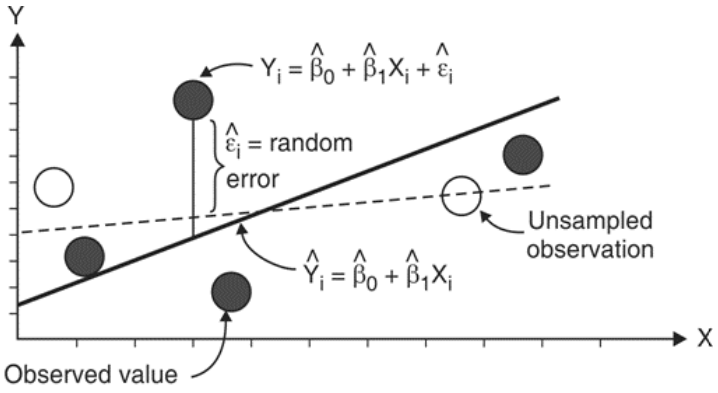
$\beta_1$ is the value of slope of the line ε is the error or the noise

- This linear equation represents a line also known as the 'regression line'. The least square estimation technique is one of the basic techniques used to guess the values of the parameters and based on a sample set.
- This technique estimates parameters $\beta_0$ and $\beta_1$ and by trying to minimise the square

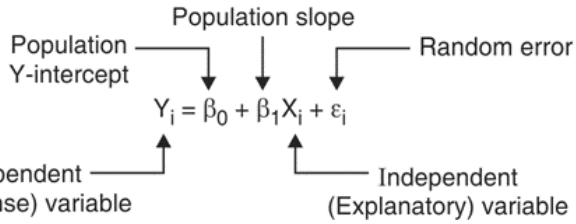
of errors at all the points in the sample set. The error is the deviation of the actual sample

- data point from the regression line. The technique can be represented by the equation.

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


Y
$Y_i = \hat{\beta}_0 + \hat{\beta}_1 X_i + \hat{\varepsilon}_i$
$\hat{\varepsilon}_i$ = random error
Unsampled observation
$\hat{Y}_i = \hat{\beta}_0 + \hat{\beta}_1 X_i$
X
Observed value

Where,


Population slope
Population Y-intercept
Random error
$Y_i = \beta_0 + \beta_1 X_i + \varepsilon_i$
Dependent (Response) variable
Independent (Explanatory) variable

Using differential calculus on equation 1 we can find the values of $\beta_0$ and $\beta_1$ such that the sum of squares (that is equation 2) is minimum.

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

$$\bar{\beta}_0 = \bar{y} - \beta_1 \bar{x} \quad (4)$$

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

### 3. Measuring Performance of Linear Regression

#### Mean Square Error:

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

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

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

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

**RMSE:**

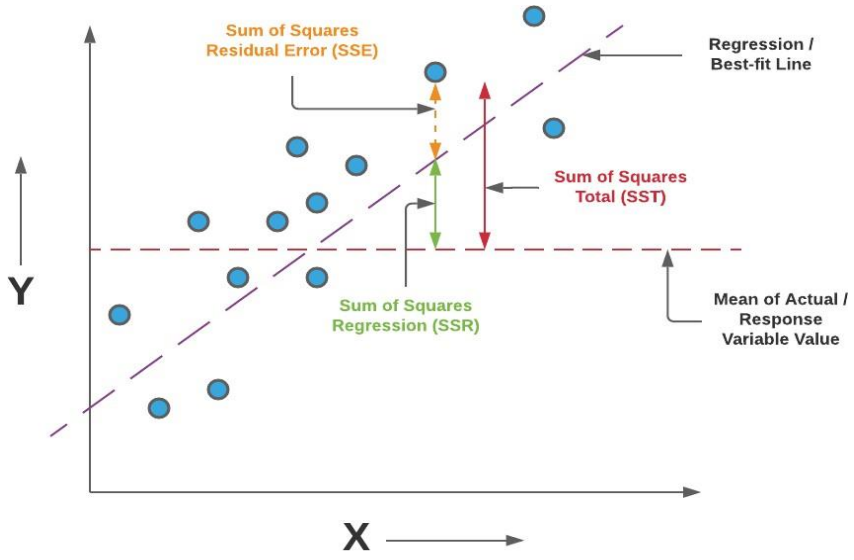
Root Mean Squared Error method that basically calculates the least-squares error and takes aroot of the summed values.

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

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

**RMSE - Least Squares Regression Method - EdurekaR-**

**Squared :**




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

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

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

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

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

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

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

**4. Example of Linear Regression**

Consider following data for 5 students.

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

(i) Linear regression equation best predicts standard XIIth score

(ii) Interpretation for the equation of Linear Regression

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

| Student | Score in X standard (Xi) | Score in XII standard (Yi) |
|---|---|---|
| 1 | 95 | 85 |
| 2 | 85 | 95 |

| | | |
|---|---|---|
| 3 | 80 | 70 |
| 4 | 70 | 65 |
| 5 | 60 | 70 |

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

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

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

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

$$\beta_1 = 470/730 = 0.644$$

$$\beta_0 = \bar{y} - \beta_1 \bar{x}$$

$$\beta_0 = 77 - (0.644 * 78) = 26.768$$

$$\hat{y} = 26.76 + 0.644 x$$

**(ii) Interpretation of the regression line.**

**Interpretation 1**

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

**Interpretation 2**

Even if x = 0 value of independent variable, it is expected that value of y is 26.768

Score in XII standard (Yi) is 0.644 units depending on Score in X standard (Xi) but other factors will also contribute to the result of XII standard by 26.768 .

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

For x = 80 the y value will be

$$\hat{y} = 26.76 + 0.644 * 65 = 68.38$$

### 5. Training data set and Testing data set

- Machine Learning algorithm has two phases

1. Training and 2. Testing.

- The input of the training phase is training data, which is passed to any machine learning algorithm and machine learning model is generated as output of the training phase.
- The input of the testing phase is test data, which is passed to the machine learning model and prediction is done to observe the correctness of mode.

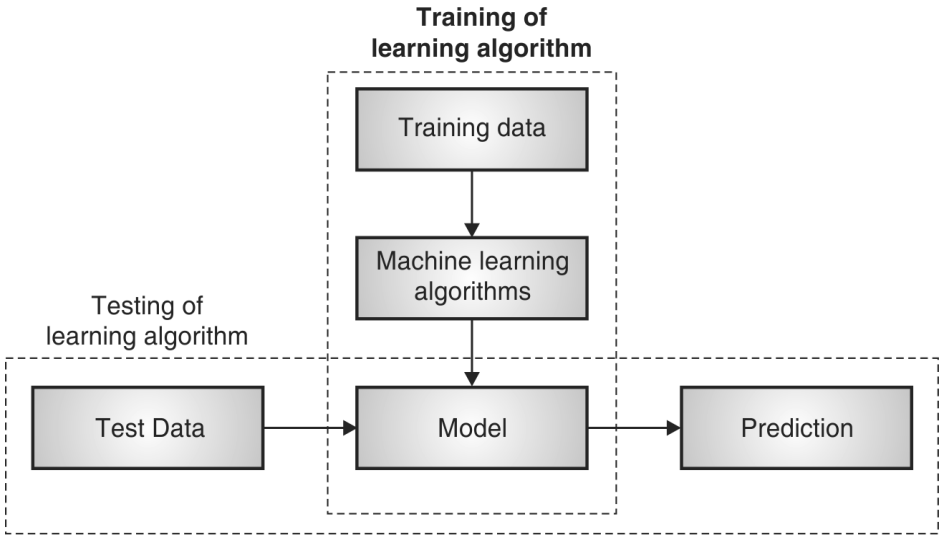



**Fig. 1.3.1 : Training and Testing Phase in Machine Learning**

**(a) Training Phase**

- Training dataset is provided as input to this phase.
- Training dataset is a dataset having attributes and class labels and used for training

Machine Learning algorithms to prepare models.

- Machines can learn when they observe enough relevant data. Using this one can model algorithms to find relationships, detect patterns, understand complex problems and make decisions**.**
- Training error is the error that occurs by applying the model to the same data from which the model is trained**.**
- In a simple way the actual output of training data and predicted output of the model does not match the training error $E_{in}$ is said to have occurred.
- Training error is much easier to compute.

**(b) Testing Phase**

- Testing dataset is provided as input to this phase.
- Test dataset is a dataset for which class label is unknown. It is tested using model
- A test dataset used for assessment of the finally chosen model.
- Training and Testing dataset are completely different.
- Testing error is the error that occurs by assessing the model by providing the unknown data to the model.
- In a simple way the actual output of testing data and predicted output of the model does not match the testing error $E_{out}$ is said to have occurred.
- $E_{out}$ is generally observed larger than $E_{in}$.

**(c) Generalization**

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

**Algorithm (Synthesis Dataset)**:

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

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

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

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

**Step 3 : Create Linear Regression Model using Polyfit Function:**

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

**Step 4: Observe the coefficients of the model.**

```
model
```

**Output:**

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

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

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

**Output:**

```
68.63
```

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

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

**Output:**

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

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

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

```
from sklearn.metrics import r2_scorer2_score(y, y_pred)
```

**Output:**

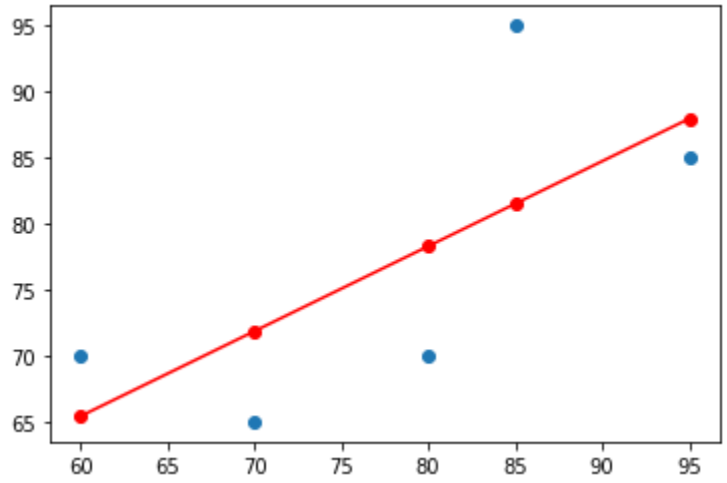
0.4803218090889323

**Step 8: Plotting the linear regression model**

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

```
plt.scatter(x,y,c='r')
```

**Output:**




**Algorithm (Boston Dataset)**:

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

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

**Step 2: Import the Boston Housing dataset**

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

**Step 3: Initialize the data frame**

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

**Step 4: Add the feature names to the dataframe**

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

**Step 5: Adding target variable to dataframe**

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

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

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

**Step 7: Split dependent variable and independent variables**

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

**Step 8: splitting data to training and testing dataset.**

```
from sklearn.model_selection import train_test_splitxtrain, xtest,
ytrain, ytest =
```

**train_test_split(x, y, test_size =0.2,random_state = 0)**

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

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

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

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

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

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

**Step 12: Calculate Mean Square Paper for train_y and test_y**

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

**Output:**

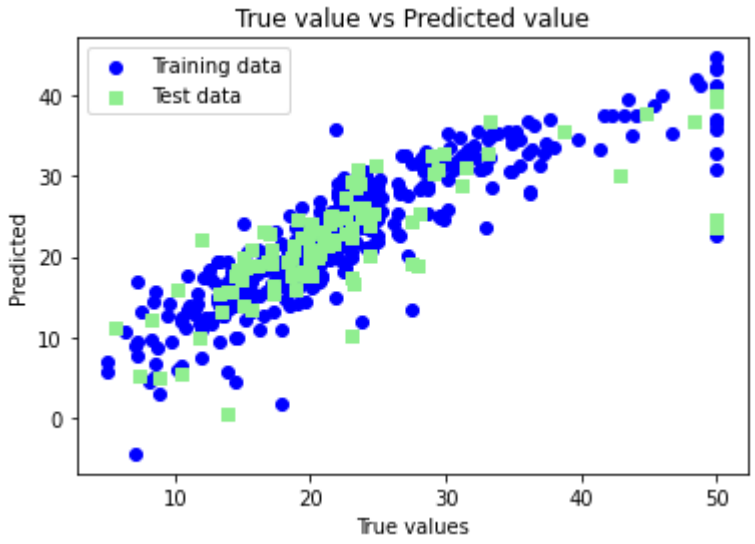
```
33.44897999767638
mse = mean_squared_error(ytest, ytest_pred)
print(mse)
```

**Output:**

```
19.32647020358573
```

**Step 13: Plotting the linear regression model**

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




**Conclusion:**

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

**Assignment Question:**

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

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

2) **Comment on whether the model is best fit or not based on the calculated values.**
3) **Write python code to calculate the RSquare for Boston Dataset.**
   **(Consider the linear regression model created in practical session)**

**CONCLUSION:-** In this way we have explored the Linear regression model.

**ASSIGNMENT 05**

**AIM:** Data Analytics II 1. Implement logistic regression using Python/R to perform classification on Social_Network_Ads.csv dataset

**OBJECTIVES:**

- To understand principles of data science for the analysis of real time problems.
- Students should be able to data analysis using logistic regression using Python for any open source dataset

**PROBLEM STATMENT:** 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

**OUTCOMES:**

**CO1:** Apply principles of data science for the analysis of real time problems.
**CO2:** Implement data representation using statistical methods
CO3: Implement and evaluate data analytics algorithms

**SOFTWARE & HARDWARE REQUIREMENTS: Jupiter/ipython 3.8.1**
**THEORY:**

**Contents for Theory:**

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

**Logistic Regression:** Classification techniques are an essential part of machine learning and data mining applications. Approximately 70% of problems in Data Science are classification problems. There are lots of classification problems that are available, but logistic regression is common and is a useful regression method for solving the binary classification problem. Another category of classification is Multinomial classification, which handles the issues where multiple classes are present in the target variable. For example, the IRIS dataset is a very famous example of multi-class classification. Other examples are classifying article/blog/document categories.

Logistic Regression can be used for various classification problems such as spam detection. Diabetes prediction, if a given customer will purchase a particular product or will they churn another competitor, whether the user will click on a given advertisement link or not, and many more examples are in the bucket.

Logistic Regression is one of the most simple and commonly used Machine Learning algorithms for two-class classification. It is easy to implement and can be used as the baseline for any binary classification problem. Its basic fundamental concepts are also constructive in deep learning. Logistic regression describes and estimates the relationship between one dependent binary variable and independent variables.

Logistic regression is a statistical method for predicting binary classes. The outcome or target variable is dichotomous in nature. Dichotomous means there are only two possible classes. For example, it can be used for cancer detection problems. It computes the probability of an event occurring.

It is a special case of linear regression where the target variable is categorical in nature. Ituses a log of odds as the dependent variable. Logistic Regression predicts the probability of occurrence of a binary event utilising a logit function.

**Linear Regression Equation:**

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

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

**Sigmoid Function:**

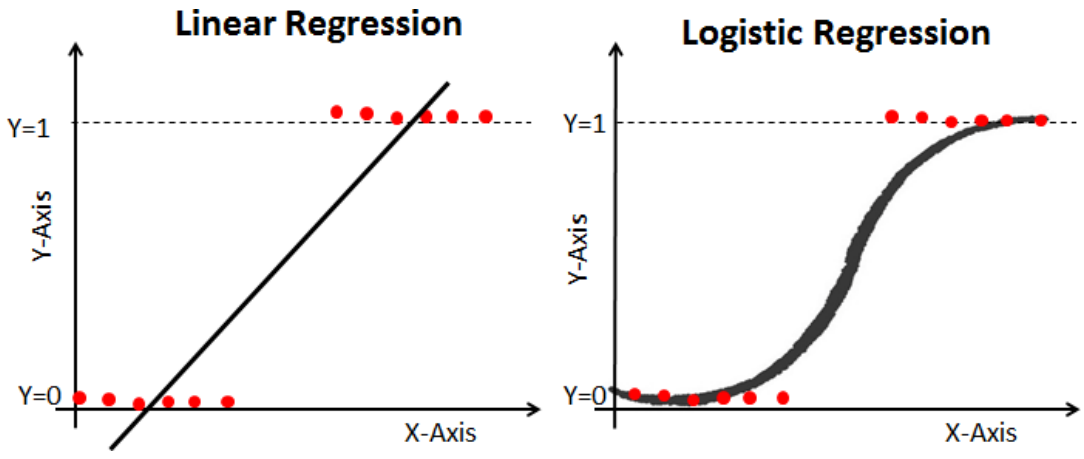
$$p = 1/1 + e^{-y}$$

**Apply Sigmoid function on linear regression:**

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

**1. Differentiate between Linear and Logistic Regression**

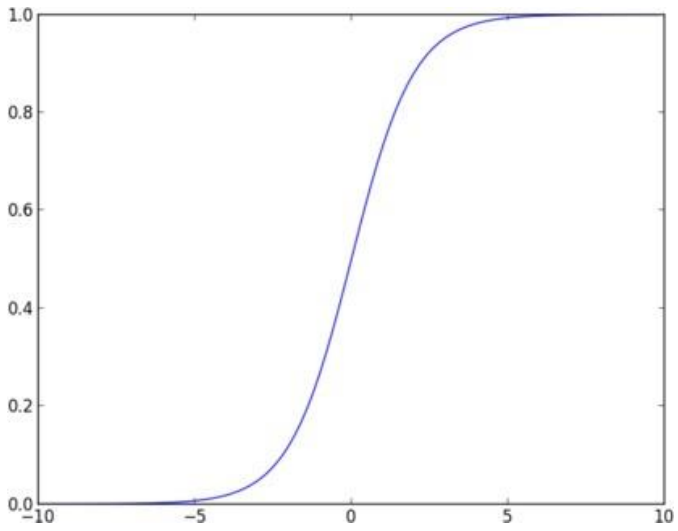
Linear regression gives you a continuous output, but logistic regression provides a constant output. An example of the continuous output is house price and stock price. Example's of the discrete output is predicting whether a patient has cancer or not, predicting whether the customer will churn. Linear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



## 2. Sigmoid Function

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

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




## 3. Types of LogisticRegression

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

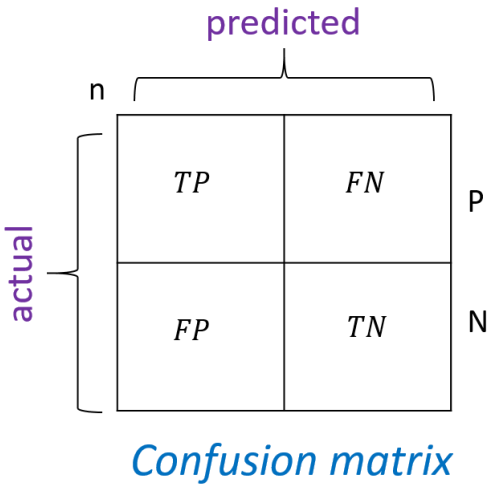
**Multinomial Logistic Regression:** The target variable has three or more nominalcategories such as predicting the type of Wine.

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

**4. Confusion Matrix Evaluation Metrics**

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

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




*Confusion matrix*

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

Some Important measures derived from confusion matrix are:

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

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

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

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

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

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

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

$$err = 1 - acc$$

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

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

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

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

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

**Algorithm (Boston Dataset)**:

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

**Step 2: Import the Social_Media_Adv Dataset**

**Step 3: Initialize the data frame Step**

**4: Perform Data Preprocessing**

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

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

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

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

**7:Evaluate the performance of Model for train_y and test_yStep 8:**

**Calculate the required evaluation parameters**

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

C**onclusion:**

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

**Value Addition:** Visualising Confusion Matrix using Heatmap

**Assignment Question:**

1) **Consider the binary classification task with two classes positive and negative.Find out TP,TP, FP, TN, FN, Accuracy, Error rate, Precision, Recall**

| N = 165 | Predicted YES | Predicted NO |
|---|---|---|
| Actual YES | TP = 150 | FN = 10 |
| Actual NO | FP = 20 | TN = 100 |

2) **Comment on whether the model is best fit or not based on the calculated values.**
3) **Write python code for the preprocessing mentioned in step 4. and Explain everystep in detail.**

**CONCLUSION:-** In this way we have explored the Logistic regression model.

**ASSIGNEMT 06**

**AIM:** Data Analytics III 1. Implement Simple Naïve Bayes classification algorithm using Python/R on iris.csv dataset

**OBJECTIVES:**

- To understand principles of data science for the analysis of real time problems.
- Students should be able to data analysis using logistic regression using Python for any open source dataset

**PROBLEM STATMENT:** 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

**OUTCOMES:**

**CO1:** Apply principles of data science for the analysis of real time problems.
**CO2:** Implement data representation using statistical methods
CO3: Implement and evaluate data analytics algorithms

**SOFTWARE & HARDWARE REQUIREMENTS: Jupiter/ipython 3.8.1**

**THEORY:**

**Overview of Naive Bayes Classification**

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

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

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

Let us understand this algorithm with a simple example. The Student will be a pass if he wears a "red" color dress on the exam day. We can solve it using above discussed method of posterior probability.

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

From the values, let us assume P (Red|Pass) = 3/9 = 0.33, P(Red) = 5/14 = 0.36, P( Pass)= 9/14 = 0.64. Now, P (Pass| Red) = 0.33 * 0.64 / 0.36 = 0.60, which has higher probability.

In this way, Naive Bayes uses a similar method to predict the probability of different class based on various attributes.

**Problem Analysis**

To implement the Naive Bayes Classification, we shall use a very famous Iris Flower Dataset that consists of 3 classes of flowers. In this, there are 4 independent variables namely

the, ***sepal_length***, ***sepal_width***, ***petal_length*** and ***petal_width***. The dependent variable is the ***species*** which we will predict using the four independent features of the flowers.

here are 3 classes of species namely *setosa*, *versicolor* and the *virginica*. This dataset was originally introduced in 1936 by Ronald Fisher. Using the various features of the flower (independent variables), we have to classify a given flower using Naive Bayes Classification model.

**Step 1: Importing the Libraries**

As always, the first step will always include importing the libraries which are the NumPy, Pandas and the Matplotlib.

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

**Step 2: Importing the dataset**

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

```
dataset = pd.read_csv('https://raw.githubusercontent.com/mk-gurucharan/Classification/master/IrisDataset.csv')X = dataset.iloc[:,:4].values
y = dataset['species'].valuesdataset.head(5)
>>
sepal_length  sepal_width  petal_length  petal_width   species
5.1           3.5          1.4           0.2           setosa
4.9           3.0          1.4           0.2           setosa
4.7           3.2          1.3           0.2           setosa
4.6           3.1          1.5           0.2           setosa
5.0           3.6          1.4           0.2           setosa
```

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

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

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

**Step 4: Feature Scaling**

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

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

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

In this step, we introduce the class GaussianNB that is used from the sklearn.naive_bayes library. Here, we have used a Gaussian model, there are several other models such as Bernoulli, Categorical and Multinomial. Here, we assign the GaussianNB class to the variable classifier and fit the X_train and y_train values to it for training purpose.

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

**Step 6: Predicting the Test set results**

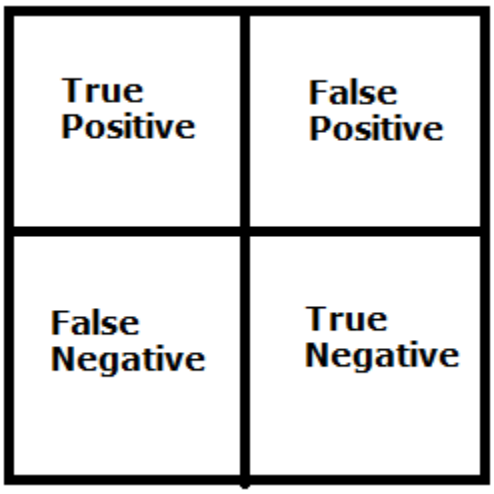
Once the model is trained, we use the the classifier.predict() to predict the values for the Test set and the values predicted are stored to the variable y_pred.

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

**Step 7: Confusion Matrix and Accuracy**

This is a step that is mostly used in classification techniques. In this, we see the Accuracy of the trained model and plot the confusion matrix.

The confusion matrix is a table that is used to show the number of correct and incorrect predictions on a classification problem when the real values of the Test Set are known. It is of the format




The True values are the number of correct predictions made.

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

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

**Step 8: Comparing the Real Values with Predicted Values**

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

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

This step is an additional step which is not much informative as the Confusion matrix and is mainly used in regression to check the accuracy of the predicted value.

**As you can see, there is one incorrect prediction that has predicted *versicolor* instead of *virginica*.**

**CONCLUSION:-** In this way we have explored the Bayes theory model.

**ASSIGNMRNT 07**

**AIM:** Text Analytics 1. Extract Sample document and apply following document preprocessing
**OBJECTIVES:**

- To understand principles of data science for the analysis of real time problems.
- Students should be able to data analysis using logistic regression using Python for any open source dataset

**PROBLEM STATMENT:** 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
**OUTCOMES:**

**CO3:** Implement and evaluate data analytics algorithms

**CO4:** Perform text preprocessing

**SOFTWARE & HARDWARE REQUIREMENTS: Jupiter/ipython 3.8.1**
**THEORY:**

Python NLTK (natural language toolkit) sentiment analysis tutorial. Learn how to create and develop sentiment analysis using Python. Follow specific steps to mine and analyze text for natural language processing.

Text Analytics has lots of applications in today's online world. By analysing tweets on Twitter, we can find trending news and peoples reaction on a particular event. Amazon can understand user feedback or review on the specific product. BookMyShow can discover people's opinion about the movie. Youtube can also analyze and understand peoples viewpoints on a video.

- Text Analytics and NLP
- Compare Text Analytics, NLP and Text Mining
    - Text Analysis Operations using NLTK
    - Tokenization
    - Stopwords
    - Lexicon Normalization such as Stemming and Lemmatization
    - POS Tagging

- Sentiment Analysis
- Text Classification
- Performing Sentiment Analysis using Text Classification

**Text Analysis Operations using NLTK**

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

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

**Tokenization**

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

**Sentence Tokenization**

Sentence tokenizer breaks text paragraph into sentences.

```
 from nltk.tokenize import sent_tokenize
text="""Hello Mr. Smith, how are you doing today? The weather is great, and city is awesome.
The sky is pinkish-blue. You shouldn't eat cardboard"""
tokenized_text=sent_tokenize(text)
print(tokenized_text)
['Hello Mr. Smith, how are you doing today?', 'The weather is great, and city is awesome.', 'The sky is
pinkish-blue.', "You shouldn't eat cardboard"]
```

Here, the given text is tokenized into sentences.

### Word Tokenization

Word tokenizer breaks text paragraph into words.

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

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

## Frequency Distribution

```
from nltk.probability import FreqDist

fdist = FreqDist(tokenized_word)

print(fdist)
```

<FreqDist with 25 samples and 30 outcomes>

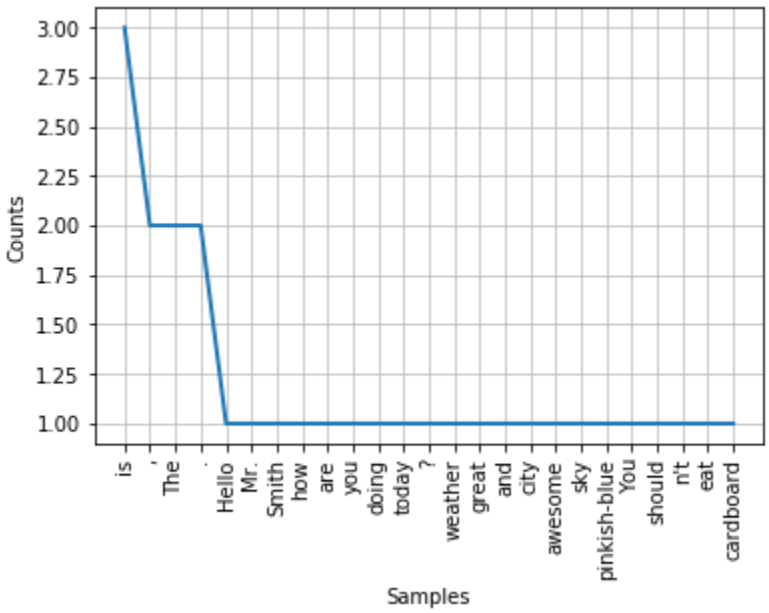
```
fdist.most_common(2)
```

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

```
# Frequency Distribution Plot

import matplotlib.pyplot as plt

fdist.plot(30,cumulative=False)
plt.show()
```






**Stopwords**

Stopwords considered as noise in the text. Text may contain stop words such as is, am, are, this, a, an, the, etc.

In NLTK for removing stopwords, you need to create a list of stopwords and filter out your list of tokens from these words.

```
from nltk.corpus import stopwords

stop_words=set(stopwords.words("english"))
print(stop_words)
```

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

## Removing Stopwords

```
filtered_sent=[]
```

```
for w in tokenized_sent:

    if w not in stop_words:

        filtered_sent.append(w)

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

```
Tokenized Sentence: ['Hello', 'Mr.', 'Smith', ',', 'how', 'are', 'you', 'doing', 'today', '?']
Filterd Sentence: ['Hello', 'Mr.', 'Smith', ',', 'today', '?']
```

### Lexicon Normalization

Lexicon normalization considers another type of noise in the text. For example, connection, connected, connecting word reduce to a common word "connect". It reduces derivationally related forms of a word to a common root word.

### Stemming

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

```
# Stemming
from nltk.stem import PorterStemmer
from nltk.tokenize import sent_tokenize, word_tokenize

ps = PorterStemmer()

stemmed_words=[]
for w in filtered_sent:
    stemmed_words.append(ps.stem(w))

print("Filtered Sentence:",filtered_sent)
print("Stemmed Sentence:",stemmed_words)
```

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

### Lemmatization

Lemmatization reduces words to their base word, which is linguistically correct lemmas. It transforms root word with the use of vocabulary and morphological analysis. Lemmatization is

usually more sophisticated than stemming. Stemmer works on an individual word without knowledge of the context. For example, The word "better" has "good" as its lemma. This thing will miss by stemming because it requires a dictionary look-up.

```
#Lexicon Normalization
#performing stemming and Lemmatization

from nltk.stem.wordnet import WordNetLemmatizer
lem = WordNetLemmatizer()

from nltk.stem.porter import PorterStemmer
stem = PorterStemmer()

word = "flying"
print("Lemmatized Word:",lem.lemmatize(word,"v"))
print("Stemmed Word:",stem.stem(word))
Lemmatized Word: fly
Stemmed Word: fli
```

**POS Tagging**

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

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

tokens=nltk.word_tokenize(sent)

print(tokens)

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

nltk.pos_tag(tokens)

[('Albert', 'NNP'),
 ('Einstein', 'NNP'),
 ('was', 'VBD'),
 ('born', 'VBN'),
 ('in', 'IN'),
```

('Ulm', 'NNP'),

(',', ','),

('Germany', 'NNP'),

('in', 'IN'),

('1879', 'CD'),
('.', '.')]
POS tagged: Albert/NNP Einstein/NNP was/VBD born/VBN in/IN Ulm/NNP ,/, Germany/NNP in/IN 1879/CD ./.

**CONCLUSION:-** In this way we have explored the Text Analytics.

**ASSIGNMENT 08**

**AIM:** Data Visualization I 1. Use the inbuilt dataset 'titanic'.

**OBJECTIVES:**

- To understand principles of data science for the analysis of real time problems.
- Students should be able to data analysis using logistic regression using Python for any open source dataset

**PROBLEM STATMENT:** 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

**OUTCOMES:**

**CO3:** Implement and evaluate data analytics algorithms

**CO4:** Implement data visualization techniques

CO5: Implement data visualization techniques

**SOFTWARE & HARDWARE REQUIREMENTS: Jupiter/ipython 3.8.1**

**THEORY:**

Seaborn which is another extremely useful library for data visualization in Python. The Seaborn library is built on top of Matplotlib and offers many advanced data visualization capabilities.

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

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

**pip install seaborn**

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

**conda install seaborn**

**The Dataset**

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

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

```
import pandas as pd
import numpy as np

import matplotlib.pyplot as plt
import seaborn as sns

dataset = sns.load_dataset('titanic')

dataset.head()
```

Distributional Plots

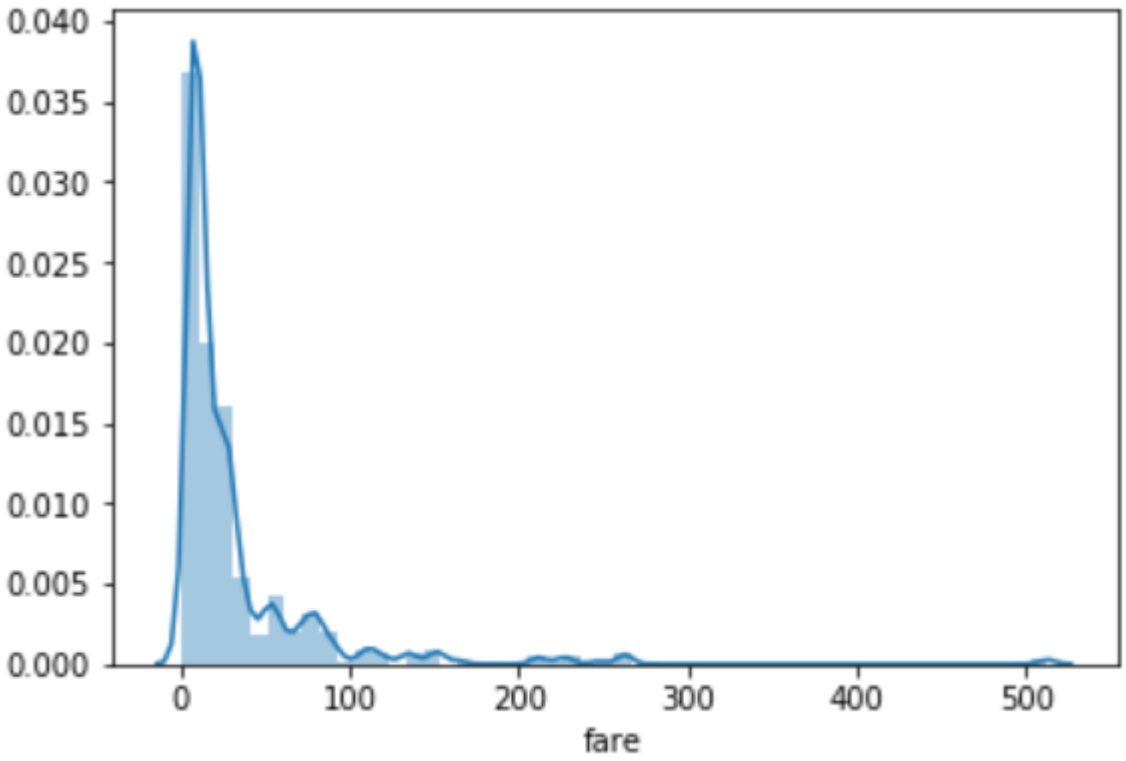
Distributional plots, as the name suggests are type of plots that show the statistical distribution of data. In this section we will see some of the most commonly used distribution plots in Seaborn.

The Dist Plot

The distplot() shows the histogram distribution of data for a single column. The column name is passed as a parameter to the distplot() function. Let's see how the price of the ticket for each passenger is distributed. Execute the following script:

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

**Output:**




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

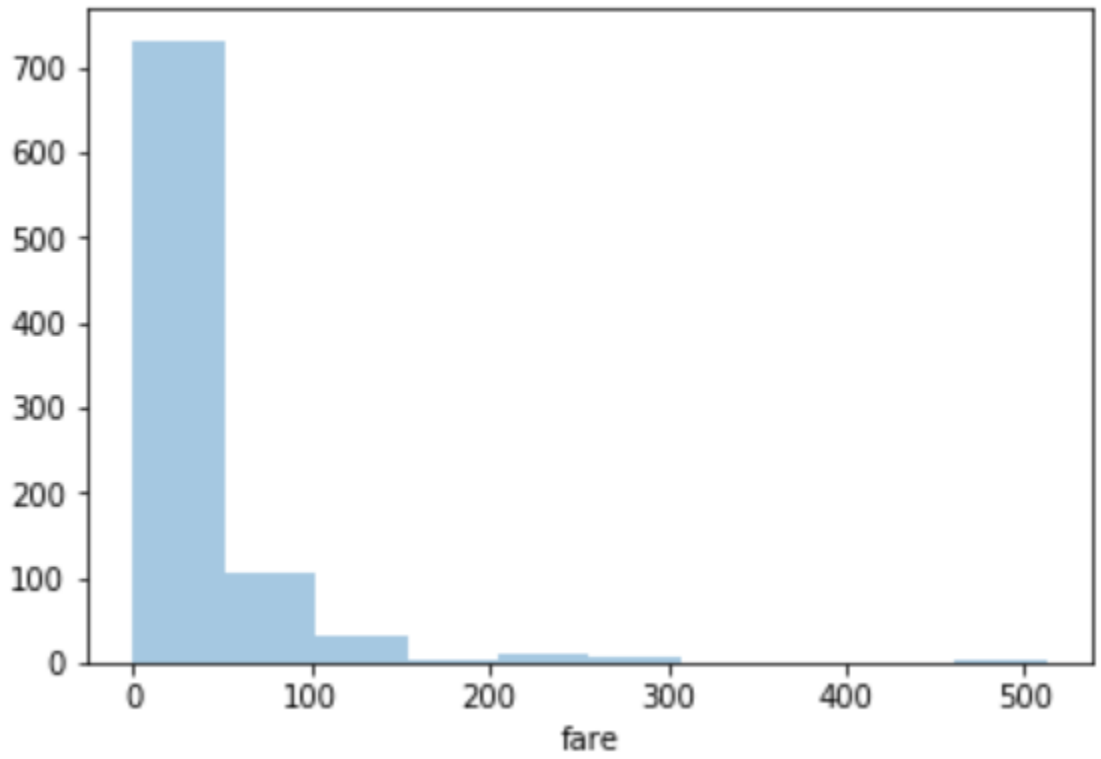
**sns.distplot(dataset['fare'], kde=False)**

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

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

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

**Output:**




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

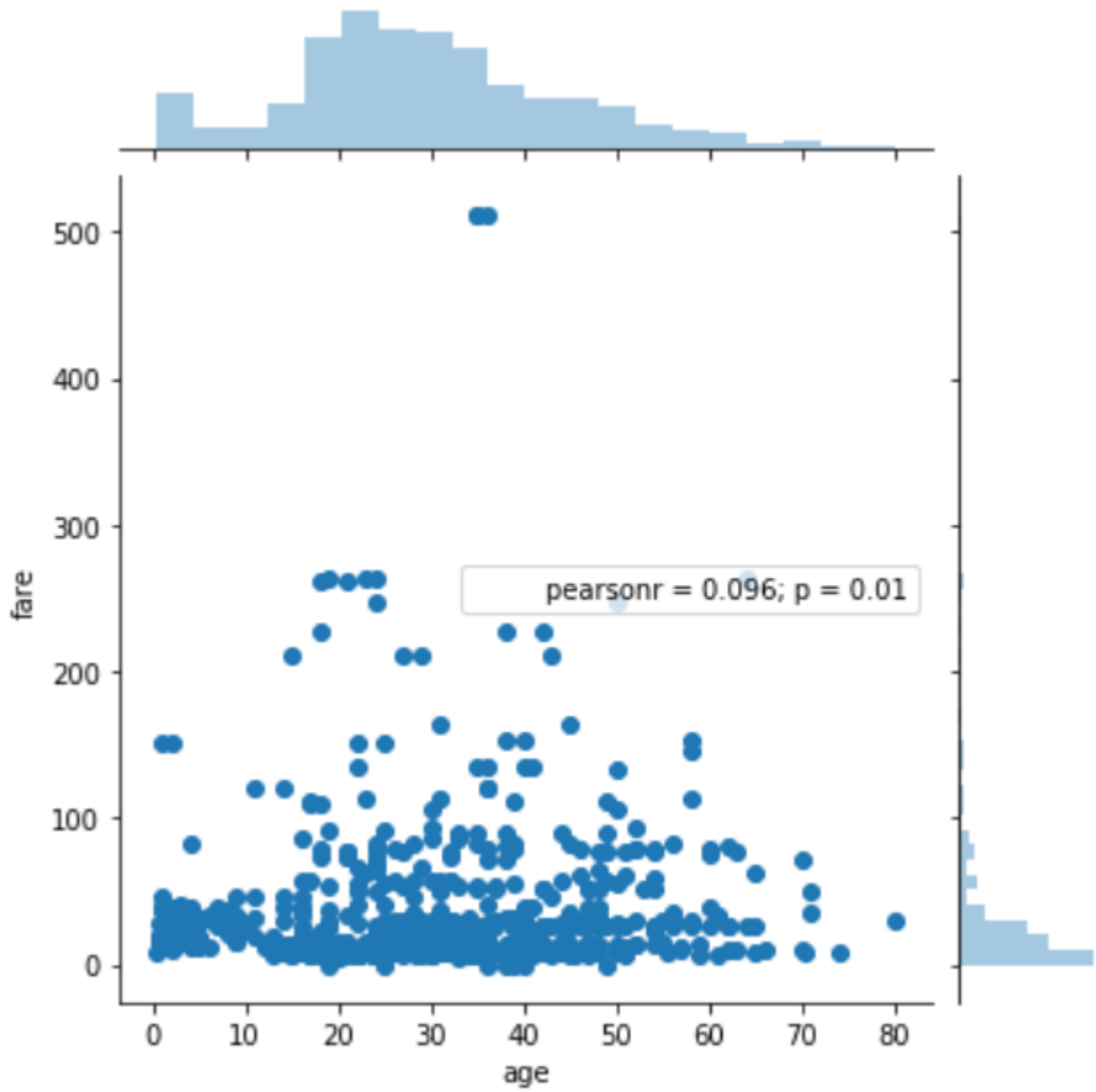
**The Joint Plot**

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

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

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

**Output:**

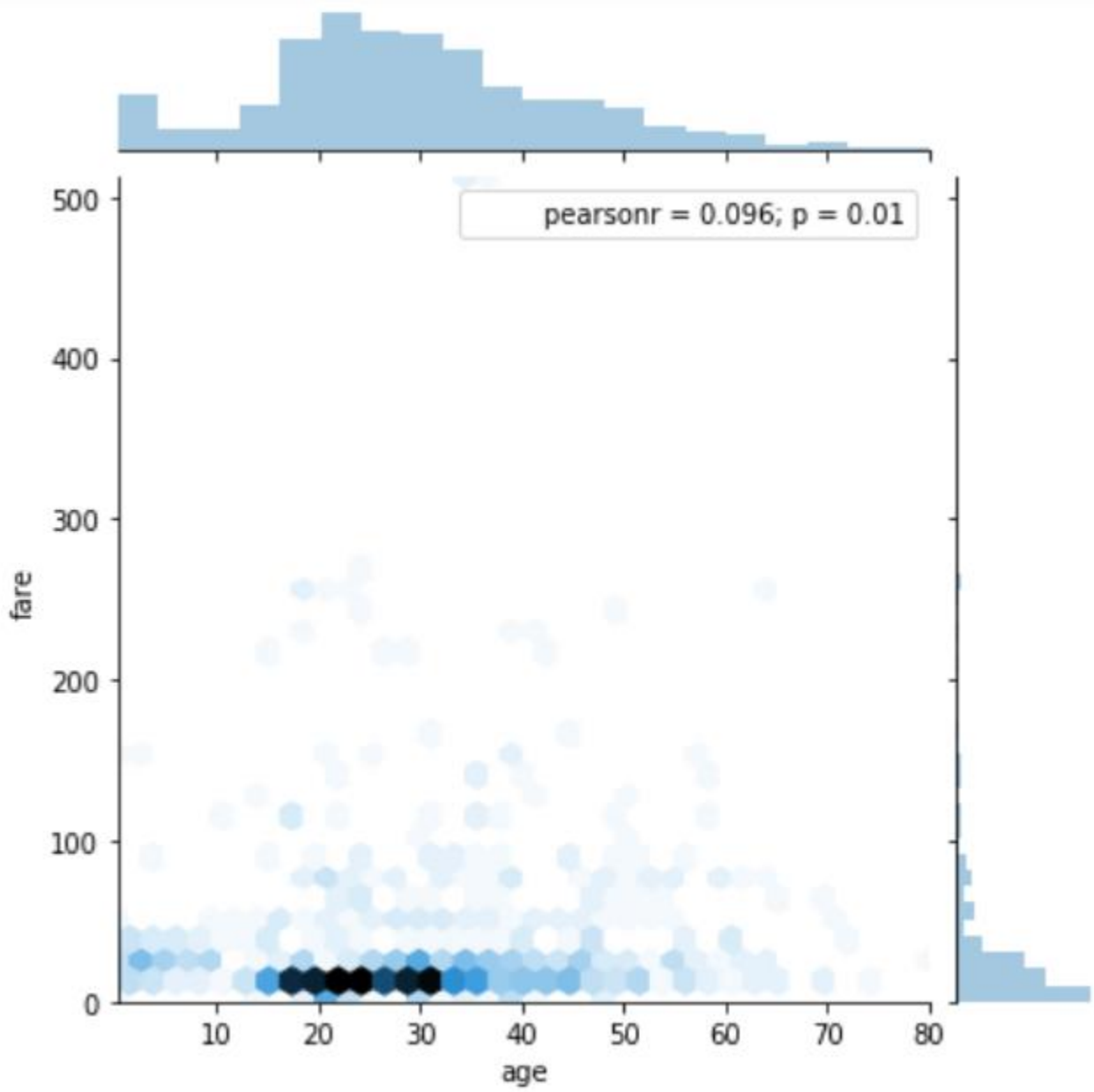



From the output, you can see that a joint plot has three parts. A distribution plot at the top for the column on the x-axis, a distribution plot on the right for the column on the y-axis and a scatter plot in between that shows the mutual distribution of data for both the columns. You can see that there is no correlation observed between prices and the fares.

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

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

**Output:**




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

**The Pair Plot**

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

**sns.pairplot(dataset)**

**CONCLUSION:-** In this way we have explored the Data Visualization I.

**ASSIGNMENT 08**

**AIM:** Data Visualization II 1. Use the inbuilt dataset 'titanic' as used in the above problem..
**OBJECTIVES:**

- To understand principles of data science for the analysis of real time problems.
- Students should be able to data analysis using logistic regression using Python for any open source dataset

**PROBLEM STATMENT:** 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.
**OUTCOMES:**

**CO3:** Implement and evaluate data analytics algorithms

**CO4:** Implement data visualization techniques

CO5: Implement data visualization techniques

**SOFTWARE & HARDWARE REQUIREMENTS: Jupiter/ipython 3.8.1**
**THEORY:**

**Categorical Plots**

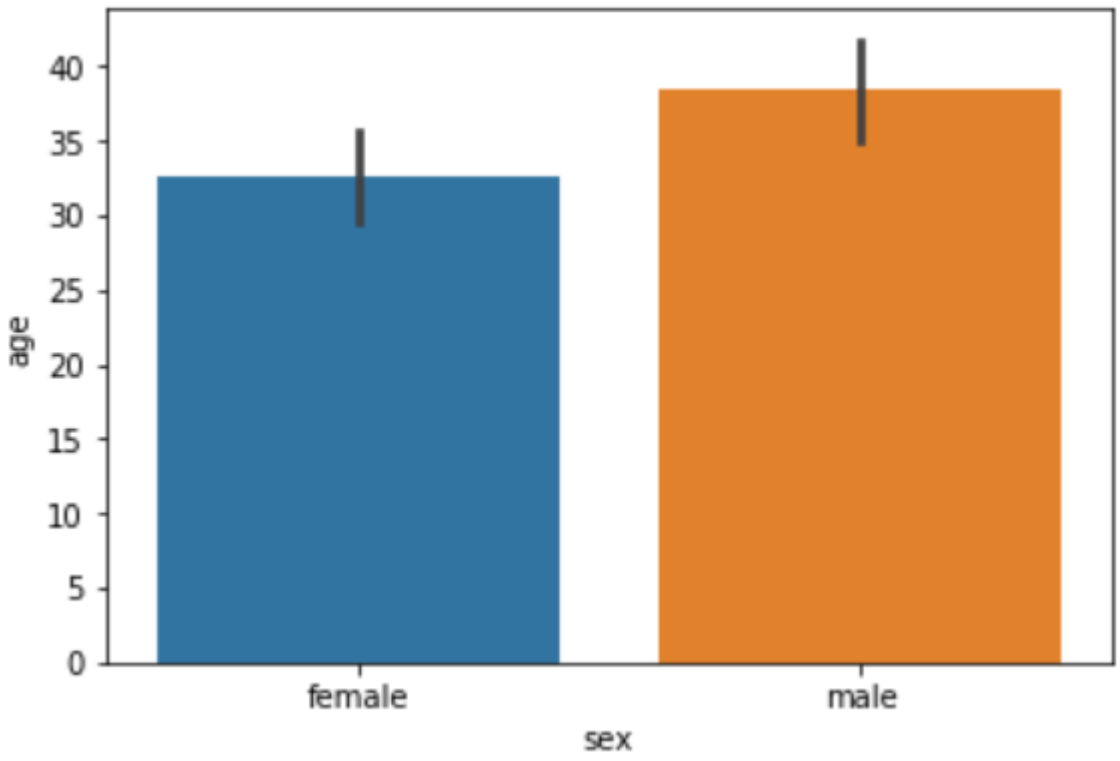
Categorical plots, as the name suggests are normally used to plot categorical data. The categorical plots plot the values in the categorical column against another categorical column or a numeric column. Let's see some of the most commonly used categorical data.

**The Bar Plot**

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

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

**Output:**




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

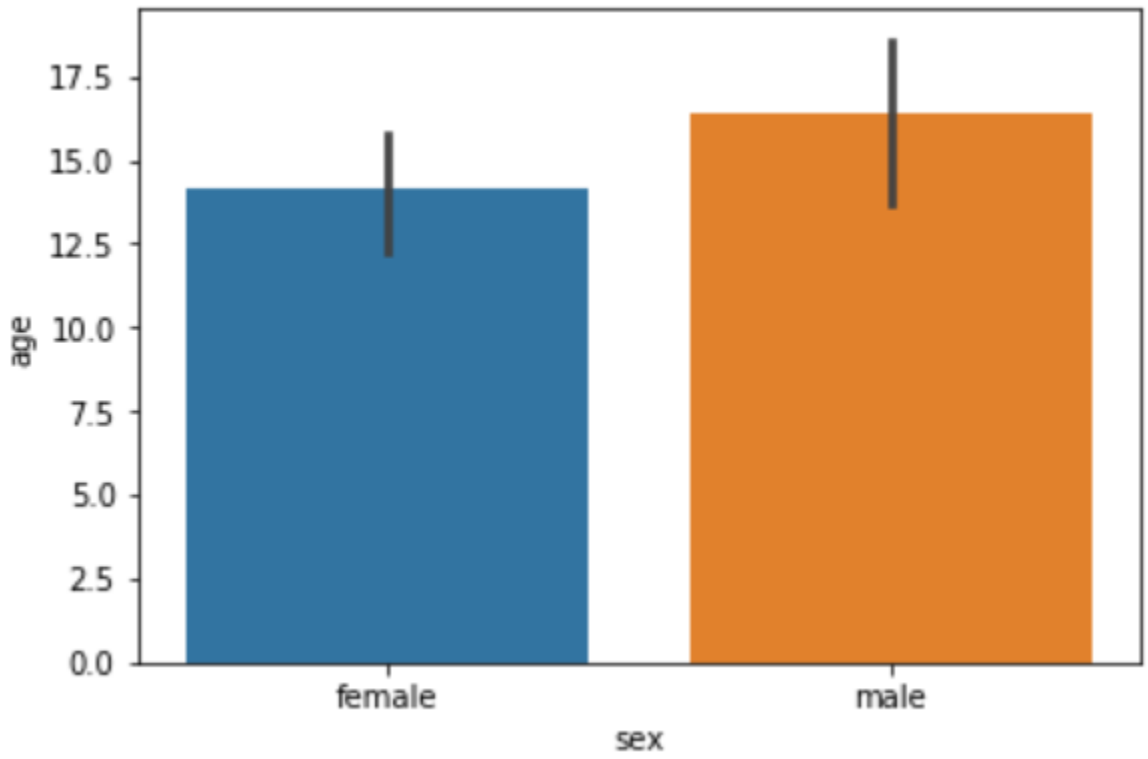
In addition to finding the average, the bar plot can also be used to calculate other aggregate values for each category. To do so, you need to pass the aggregate function to the `estimator`. For instance, you can calculate the standard deviation for the age of each gender as follows:

```
import numpy as np

import matplotlib.pyplot as plt
import seaborn as sns

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

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

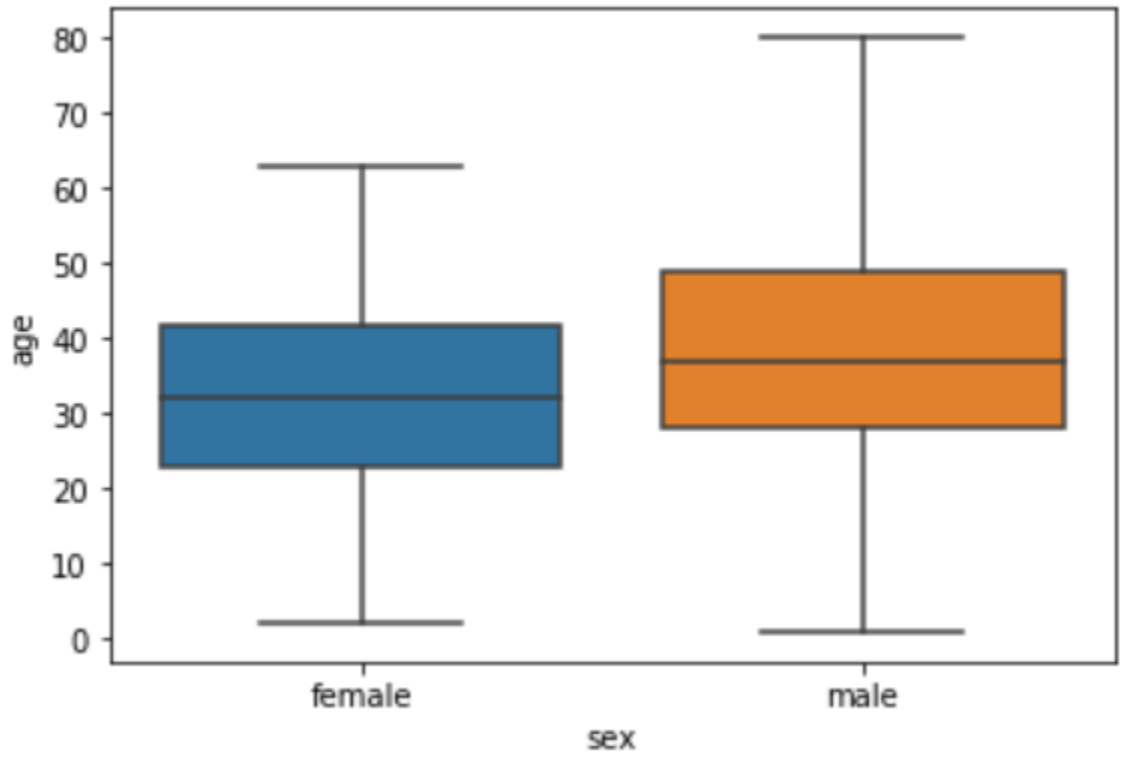



**The Count Plot**

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

**sns.countplot(x='sex', data=dataset)**
**The output shows the count as follows:**
**sns.boxplot(x='sex', y='age', data=dataset)**

**Output:**

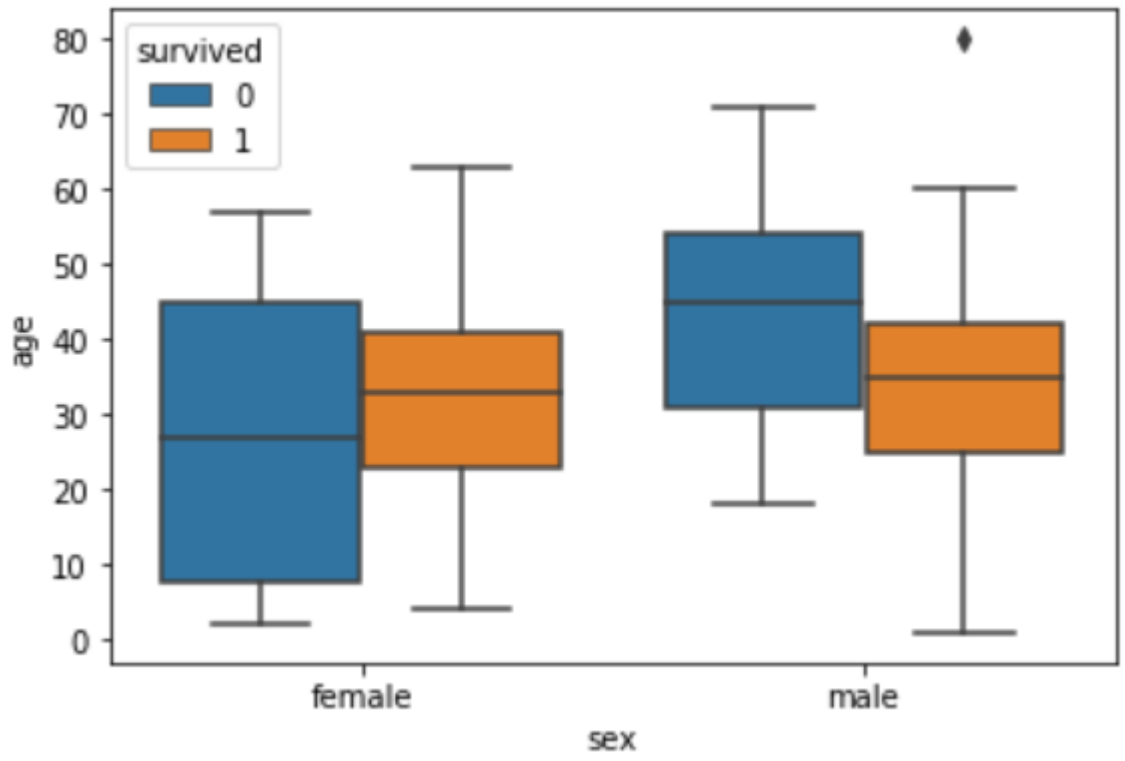



If there are any outliers or the passengers that do not belong to any of the quartiles, they are called outliers and are represented by dots on the box plot.

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

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

**Output:**

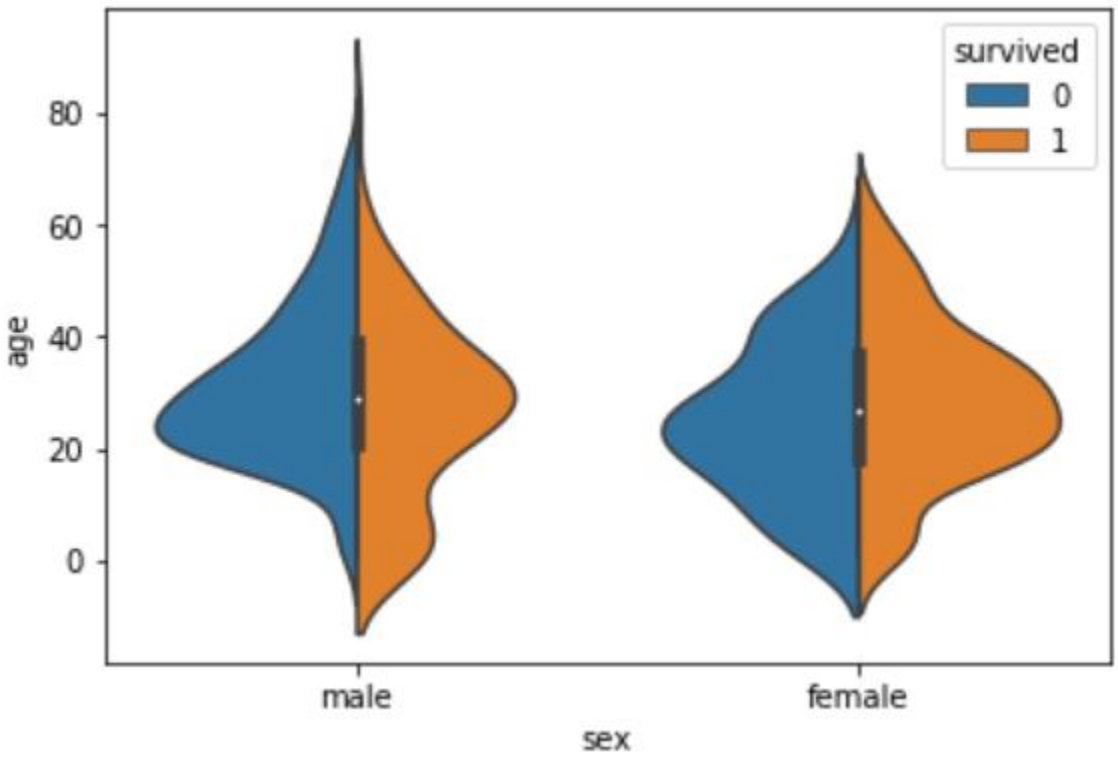



Now in addition to the information about the age of each gender, you can also see the distribution of the passengers who survived. For instance, you can see that among the male passengers, on average more younger people survived as compared to the older ones. Similarly, you can see that the variation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

nstead of plotting two different graphs for the passengers who survived and those who did not, you can have one violin plot divided into two halves, where one half represents surviving while the other half represents the non-surviving passengers. To do so, you need to pass `True` as value for the `split` parameter of the `violinplot()` function. Let's see how we can do this:

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

The output looks like this:




Now you can clearly see the comparison between the age of the passengers who survived and who did not for both males and females.

**CONCLUSION:-** In this way we have explored the Data Visualization II.

**ASSIGNEMENT 10**

**AIM:** Data Visualization III Download the Iris flower dataset or any other dataset into a DataFrame.

**OBJECTIVES:**

- To understand principles of data science for the analysis of real time problems.
- Students should be able to data analysis using logistic regression using Python for any open source dataset

**PROBLEM STATMENT:** Data Visualization III Download the Iris flower dataset or any other dataset into a DataFrame. (eg https://archive.ics.uci.edu/ml/datasets/Iris ). Scan the dataset and give the inference as: 1. How many features are there and what are their types (e.g., numeric, nominal)? 2. Create a histogram for each feature in the dataset to illustrate the feature distributions. 3. Create a boxplot for each feature in the dataset. Compare distributions and identify outliers.

**OUTCOMES:**

**CO3:** Implement and evaluate data analytics algorithms

**CO5:** Implement data visualization techniques

**SOFTWARE & HARDWARE REQUIREMENTS: Jupiter/ipython 3.8.1**

**THEORY:**

Data Analysis on the IRIS Flower Dataset

Download the Iris flower dataset or any other dataset into a DataFrame.

(eg https://archive.ics.uci.edu/ml/datasets/Iris) Use Python/R and perform following –

1. How many features are there and what are their types (e.g., numeric, nominal)?
2. Compute and display summary statistics for each feature available in the dataset. (eg. minimum value, maximum value, mean, range, standard deviation, variance and percentiles)
3. Data Visualization-Create a histogram for each feature in the dataset to illustrate the feature distributions. Plot each histogram.
4. Create a boxplot for each feature in the dataset. All of the boxplots should be combined into a single plot. Compare distributions and identify outliers.

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

path = "iris.csv"

df = pd.read_csv(path, header=None)

headers = ["Sepal-length", "Sepal-width", "Petal-length", "Petal-width", "Species"]
df.columns = headers
```

```
print(df.head())
print(df.tail())
print(df.info())
print(df.shape)
print(df.dtypes)
print(df.describe())

df.hist()
plt.show()

df.boxplot()
plt.show()

plt.scatter(df["Sepal-length"], df["Sepal-width"])
plt.xlabel('Sepal Length')
plt.ylabel('Sepal Width')
plt.show()

plt.scatter(df["Sepal-length"], df["Petal-length"])
plt.xlabel('Sepal Length')
plt.ylabel('Petal Width')


plt.show()

plt.scatter(df["Sepal-length"], df["Petal-width"])
plt.xlabel('Sepal Length')
plt.xlabel('Petal Width')
plt.show()

plt.scatter(df["Sepal-width"], df["Sepal-length"])
plt.xlabel('Sepal Width')
plt.ylabel('Sepal Length')
plt.show()

plt.scatter(df["Sepal-width"], df["Petal-length"])
plt.xlabel('Sepal Width')
plt.ylabel('Petal Length')
plt.show()

plt.scatter(df["Sepal-width"], df["Petal-width"])
plt.xlabel('Sepal Width')
plt.ylabel('Petal Width')
plt.show()

plt.scatter(df["Petal-length"], df["Sepal-length"])
```

```
plt.xlabel('Petal Length')
plt.ylabel('Sepal Length')
plt.show()

plt.scatter(df["Petal-length"], df["Sepal-width"])
plt.xlabel('Petal Length')
plt.ylabel('Sepal Width')
plt.show()

plt.scatter(df["Petal-length"], df["Petal-width"])
plt.xlabel('Petal Length')
plt.ylabel('Petal Width')
plt.show()

plt.scatter(df["Petal-width"], df["Sepal-length"])
plt.xlabel('Petal Width')
plt.xlabel('Sepal Length')
plt.show()

plt.scatter(df["Petal-width"], df["Sepal-width"])
plt.xlabel('Petal Width')
plt.xlabel('Sepal Width')
plt.show()


plt.scatter(df["Petal-width"], df["Petal-length"])
plt.xlabel('Petal Width')
plt.yxlabel('Petal Length')
plt.show()
```

---

```
import numpy as np
import pandas as pd
df = pd.read_csv("iris-flower-dataset.csv",header=None)
df.columns = ["col1","col2","col3","col4","col5"]
df.head()
column = len(list(df))
column
df.info()
np.unique(df["col5"])
array(['Iris-setosa', 'Iris-versicolor', 'Iris-virginica'], dtype=object)
df.describe()
import seaborn as sns
import matplotlib
import matplotlib.pyplot as plt
%matplotlib inline
fig, axes = plt.subplots(2, 2, figsize=(16, 8))
```

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

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

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

axes[1,1].set_title("Distribution of Fourth Column")
axes[1,1].hist(df["col4"]);
data_to_plot = [df["col1"],df["col2"],df["col3"],df["col4"]]

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

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

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

**CONCLUSION:-** In this way we have explored the Data Visualization III.

**ASSIGNEMENT 11**

**AIM:** Write a code in JAVA for a simple WordCount application that counts the number of occurrences of each word in a given input set

**OBJECTIVES:**

- To understand principles of data science for the analysis of real time problems.
- Students should be able to data analysis using logistic regression using Python for any open source dataset

**PROBLEM STATMENT:** 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

**OUTCOMES:**

CO1: Apply principles of data science for the analysis of real time problems.

CO6: Use cutting edge tools and technologies to analyze Big Data

**SOFTWARE & HARDWARE REQUIREMENTS: Jupiter/ipython 3.8.1**

**THEORY:**

This is an Hadoop Map/Reduce application for Working on weather data It reads the text input files, breaks each line into stations weather data and finds average for temperature , dew point , wind speed. The output is a locally sorted list of stations and its 12 attribute vector of average temperature , dew , wind speed of 4 sections for each month.

Installation

Since this a Map reduce Project , please install Apache Hadoop , Refere the below site for more details

https://hadoop.apache.org/docs/current/hadoop-mapreduce-client/hadoop-mapreduce-client-core/MapReduceTutorial.html

Once you have installed the hadoop , download the project using git commands and install it.

$ git clone https://github.com/vasanth-mahendran/weather-data-hadoop.git $ cd weather-data-hadoop $ mvn install

Run

Add the sample input file given in the proect to the hdfs

hdfs dfs -put sample_weather.txt

and start the Map reduce application to submit the job to hadoop

hadoop jar ./target/weather-1.0.jar sample_weather.txt dataOutput

**CONCLUSION:-** In this way we have explored the Hadoop MapReduce framework on local-standalone set-up.

**ASSIGNMENT 12**

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

- To understand principles of data science for the analysis of real time problems.
- Students should be able to data analysis using logistic regression using Python for any open source dataset

**PROBLEM STATMENT:** Design a distributed application using MapReduce which processes a log file of a system.
**OUTCOMES:**

CO1: Apply principles of data science for the analysis of real time problems.

CO6: Use cutting edge tools and technologies to analyze Big Data

**SOFTWARE & HARDWARE REQUIREMENTS: Jupiter/ipython 3.8.1**
**THEORY:**

This section refers to the installation settings of Hadoop on a standalone system as well as on a system existing as a node in a cluster. SINGLE-NODE INSTALLATION Running Hadoop on Ubuntu (Single node cluster setup)
The report here will describe the required steps for setting up a single-node Hadoop cluster backed by the Hadoop Distributed File System, running on Ubuntu Linux. Hadoop is a framework written in Java for running applications on large clusters of commodity hardware and incorporates features similar to those of the Google File System (GFS) and of the MapReduce computing paradigm. Hadoop's HDFS is a highly fault-tolerant distributed file system and, like Hadoop in general, designed to be deployed on low-cost hardware. It provides high throughput access to application data and is suitable for applications that have large data sets. Before we start, we will understand the meaning of the following:
DataNode: A DataNode stores data in the Hadoop File System. A functional file system has more than one DataNode, with the data replicated across them.
NameNode: The NameNode is the centrepiece of an HDFS file system. It keeps the directory of all files in the file system, and tracks where across the cluster the file data is kept. It does not store the data of these file itself.
Jobtracker: The Jobtracker is the service within hadoop that farms out MapReduce to specific nodes in the cluster, ideally the nodes that have the data, or atleast are in the same rack.
TaskTracker: A TaskTracker is a node in the cluster that accepts tasks- Map, Reduce and Shuffle operatons – from a Job Tracker.
Secondary Namenode: Secondary Namenode whole purpose is to have a checkpoint in HDFS. It is just a helper node for namenode. Prerequisites Java is the primary requirement for run hadoop

on any system, so make sure you have Java installed on your system using following command. # java -version If you don't have Java installed on your system, follow the step to install java. Java download

**CONCLUSION:-** In this way we have explored the Hadoop framework MapReduce which processes a log file of a system.

**ASSIGNMENT 13**

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

**OBJECTIVES:**

- To analyze and demonstrate knowledge of statistical data analysis techniques for decision-making
- To gain practical, hands-on experience with statistics programming languages and big data tools

**PROBLEM STATMENT:** Locate dataset (eg. sample_weather.txt) for working on weather data which reads the text input files and finds average for temperature, dew point and wind speed.

**OUTCOMES:**

CO1: Apply principles of data science for the analysis of real time problems.

CO6: Use cutting edge tools and technologies to analyze Big Data

**SOFTWARE & HARDWARE REQUIREMENTS: Jupiter/ipython 3.8.1**

**THEORY:**

This is an Hadoop Map/Reduce application for Working on weather data It reads the text input files, breaks each line into stations weather data and finds average for temperature , dew point , wind speed. The output is a locally sorted list of stations and its 12 attribute vector of average temperature , dew , wind speed of 4 sections for each month.

InstallationSince this a Map reduce Project , please install Apache Hadoop , Refere the below site for more details

Once you have installed the hadoop , download the project using git commands and install it.

$ git clone https://github.com/vasanth-mahendran/weather-data-hadoop.git $ cd weather-data-hadoop $ mvn install

Run

Add the sample input file given in the proect to the hdfs

hdfs dfs -put sample_weather.txt

and start the Map reduce application to submit the job to hadoop

hadoop jar ./target/weather-1.0.jar sample_weather.txt dataOutput

**CONCLUSION:-** In this way we have explored the Locate dataset (eg. sample_weather.txt) for working on weather data which reads the text input files