A Laboratory Manual for

Data Science
(3151608)

B.E. Semester 5
(Information Technology Department)

L. D. College of Engineering
Ahmedabad, Gujarat, India

Directorate of Technical Education,
Gandhinagar, Gujarat

L. D. College of Engineering, Ahmedabad

Department of Information Technology

Certificate

This is to certify that Mr./Ms. _____ _____ Enrollment No. _____of B.E. Semester 5, Information Technology department of this Institute (GTU Code: 028) has satisfactorily completed the Practical / Tutorial work for the subject **Data Science (3151608)** for the academic year 2023-24.

Place: _____
Date: _____

Name and Sign of Faculty member

Head of the Department

Preface

Main motto of any laboratory/practical/field work is for enhancing required skills as well as creating ability amongst students to solve real time problem by developing relevant competencies in psychomotor domain. By keeping in view, GTU has designed competency focused outcome-based curriculum for engineering degree programs where sufficient weightageis given to practical work. It shows importance of enhancement of skills amongst the students and it pays attention to utilize every second of time allotted for practical amongst students, instructors and faculty members to achieve relevant outcomes by performing the experiments rather than having merely study type experiments. It is must for effective implementation of competency focused outcome-based curriculum that every practical is keenly designed to serve as a tool to develop and enhance relevant competency required by the various industry among every student. These psychomotor skills are very difficult to develop through traditional chalk and board content delivery method in the classroom. Accordingly, this lab manual is designedto focus on the industry defined relevant outcomes, rather than old practice of conducting practical to prove concept and theory.

Data Science is a rapidly growing field that combines statistical and computational techniques to extract knowledge and insights from data. The goal of this lab manual is to provide students with hands-on experience in using data science tools and techniques to analyze and interpret real-world data.

This manual is designed to accompany a course in Data Science and assumes a basic knowledge of programming concepts and statistical analysis. The labs are structured to guide students through the process of collecting, cleaning, analyzing, and visualizing data, using popular programming languages and software tools such as Python, R, SQL, and Tableau.

Each lab in this manual consists of a set of instructions that guide students through a specific data analysis project. The labs are organized in a progressive sequence, with each lab building on the skills and concepts covered in the previous lab. The exercises within each lab are designed to be completed in a single class session, with additional time required for preparation and follow-up analysis.

Throughout the manual, we emphasize the importance of critical thinking and data ethics, providing guidance on how to analyze data responsibly and communicate findings effectively. By the end of this manual, students will have gained a solid foundation in data science and be well-equipped to apply these skills to real-world problems.

Practical – Course Outcome matrix

Course Outcomes (COs):

After successful completion of this course, the students should be able to

CO-1 Describe the various areas where data science is applied.
CO-2 Identify the data types, relation between data and visualization technique for data.
CO-3 Explain probability, distribution, sampling, Estimation.
CO-4 Solve regression and classification problem

Sr. No.	Objective(s) of Experiment	CO-1	CO-2	CO-3	CO-4
1.	Exploration and Visualization Using Mathematical and Statistical Tools	√	√		
2.	Study of Measures of Central Tendency, Correlation, Percentile, Decile, Quartile, Measure of Variation, and Measure of Shape (Skewness and Kurtosis) with Excel Functions	√	√		
3.	Study of Basics of Python data types, NumPy, Matplotlib, Pandas.	√	√		
4.	Implementation of Various Probability Distributions with NumPy Random Library Functions			√	
5.	Implementation of Estimation of Parameters for the Best-Fit Probability Distribution using the Fitter Class in Python.			√	
6.	Implementation of Linear Regression with Scikit-learn library in Python.				√
7.	Implementation of Logistic Regression with Scikit-learn library in Python				√
8.	Implementation of Decision Tree for Student Classification				√

Guidelines for Faculty members

1. Course Coordinator / Faculty should provide the guideline with demonstration of practical to the students with all features.
2. Course Coordinator / Faculty shall explain basic concepts/theory related to the experiment to the students before starting of each practical
3. Involve all the students in performance of each experiment.
4. Course Coordinator / Faculty is expected to share the skills and competencies to be developed in the students and ensure that the respective skills and competencies are developed in the students after the completion of the experimentation.
5. Course Coordinator / Faculty should give opportunity to students for hands-on experience after the demonstration.
6. Course Coordinator / Faculty may provide additional knowledge and skills to the students even though not covered in the manual but are expected from the students by concerned industry.
7. Give practical assignment and assess the performance of students based on task assigned to check whether it is as per the instructions or not.
8. Course Coordinator / Faculty is expected to refer complete curriculum of the course and follow the guidelines for implementation.

Instructions for Students

1. Students are expected to carefully listen to all the theory classes delivered by the faculty members and understand the COs, content of the course, teaching and examination scheme, skill set to be developed etc.
2. Students will have to perform experiments as per practical list given.
3. Students have to show output of each program in their practical file.
4. Students are instructed to submit practical list as per given sample list shown on next page.
5. Student should develop a habit of submitting the experimentation work as per the schedule and s/he should be well prepared for the same.

Common Safety Instructions

Students are expected to

1) Switch on the PC carefully (not to use wet hands)
2) Shutdown the PC properly at the end of your Lab
3) Carefully handle the peripherals (Mouse, Keyboard, Network cable etc).
4) Use Laptop in lab after getting permission from Course Coordinator / Faculty

Rubrics for Practical Assessment (For Total marks 10)				
Criteria No	**Parameters**	**Strong**	**Average**	**Poor**
C1	Understanding of Problem (2 Marks)	Excellent understanding of problem and relevance with the theory clearly understood.(2)	Moderate level understanding of problem and relevance with the theory clearly understood.(1)	Problem not understood and can't establish the relation with the theory.(0)
C2	Analysis of the Problem (2 Marks)	Good ability to identify strategies for solving problems (brainstorming, exploration of various solutions, trial and error). (2)	Moderate ability to identify strategies for solving problems (by guidance of faculty, exploration of limited solutions, trial and error).(1)	Poor ability to identify strategies for solving problems (require special attention from faculty, trial and error). (0)
C3	Capability of writing program (5 Marks)	Efficient implementation with proper naming convention and understanding.(5)	Moderate level of implementation. Poor naming convention. (4-3)	Partial implementation with poor understanding.(2-0)
C4	Documentation (1 Marks)	Unique documentation (not copied from other sources) of given problem with proper formatting and language.(1)	Ordinary documentation of given problem with proper formatting and language(0.5)	Weak documentation of given problem without proper formatting and language(0)

Index
(Progressive Assessment Sheet)

Sr. No.	Objective(s) of Experiment	Page No.	Date of performance	Date of submission	Assessment Marks	Sign. of Faculty with date
1.	Exploration and Visualization Using Mathematical and Statistical Tools (10 Marks)					
2.	Study of Measures of Central Tendency, Correlation, Percentile, Decile, Quartile, Measure of Variation, and Measure of Shape (Skewness and Kurtosis) with Excel Functions. (10 Marks)					
3.	Study of Basics of Python data types, NumPy, Matplotlib, Pandas. (10 Marks)					
4.	Implementation of Various Probability Distributions with NumPy Random Library Functions. (10 Marks)					
5.	Implementation of Estimation of Parameters for the Best-Fit Probability Distribution using the Fitter Class in Python. (10 Marks)					
6.	Implementation of Linear Regression with Scikit-learn library in Python. (20 Marks)					
7.	Implementation of Logistic Regression with Scikit-learn library in Python (20 Marks)					
8.	Implementation of Decision Tree for Student Classification. (10 Marks)					
Total					100	

Experiment No: 1

Date:

AIM: Data Exploration and Visualization Using Mathematical and Statistical Tools

Introduction:

Data exploration and visualization are important steps in the data analysis process. In this lab, students will learn how to explore and visualize data using mathematical and statistical tools such as histograms, box plots, scatter plots, and correlation matrices. Students will also learn how to use Excel/R to perform these analyses.

Relevant CO: CO1, CO2

Objectives:

1. To understand the importance of data exploration and visualization in data analysis.
2. To learn how to use Excel and/or R to create histograms, box plots, scatter plots, and correlation matrices.
3. To interpret the results of these analyses and draw conclusions from them.
4. To present the results of these analyses in a clear and effective manner.

Materials:

- A computer with Microsoft Excel and R installed
- Sample dataset provided by subject faculty or shown below.

Procedure:

1. Open Microsoft Excel and create a new workbook.
2. Input the sample dataset provided by your faculty into a worksheet in the workbook.
3. Use Excel to create a histogram for each column in the dataset.
4. Use Excel to create a box plot for each column in the dataset.
5. Use R to create a scatter plot for two variables in the dataset.
6. Use R to create a correlation matrix for all variables in the dataset.
7. Interpret the results of these analyses and draw conclusions from them.
8. Present the results of these analyses in a clear and effective manner using charts and graphs.
9. Submit the completed workbook and presentation to the faculty for grading.

Example:

Age	Gender	Income	Education Level	Marital Status	Employment Status	Industry
32	Female	45000	Bachelor's	Single	Employed	Technology
45	Male	65000	Master's	Married	Employed	Finance
28	Female	35000	High School	Single	Unemployed	None
52	Male	80000	Doctorate	Married	Employed	Education
36	Female	55000	Bachelor's	Divorced	Employed	Healthcare
40	Male	70000	Bachelor's	Married	Self-Employed	Consulting
29	Female	40000	Associate's	Single	Employed	Retail
55	Male	90000	Master's	Married	Employed	Engineering

33	Female	47000	Bachelor's	Single	Employed	Government
47	Male	75000	Bachelor's	Married	Self-Employed	Entertainment
41	Female	60000	Master's	Single	Employed	Nonprofit
38	Male	52000	High School	Divorced	Employed	Construction
31	Female	48000	Bachelor's	Married	Employed	Technology
49	Male	85000	Doctorate	Married	Employed	Finance
27	Female	30000	High School	Single	Unemployed	None
54	Male	92000	Master's	Married	Employed	Education
39	Female	58000	Bachelor's	Married	Self-Employed	Consulting
30	Male	42000	Associate's	Single	Employed	Retail
56	Female	96000	Doctorate	Married	Employed	Healthcare
35	Male	55000	Bachelor's	Single	Employed	Government
48	Female	73000	Bachelor's	Married	Self-Employed	Entertainment
42	Male	65000	Master's	Divorced	Employed	Nonprofit
37	Female	50000	High School	Married	Employed	Construction
34	Male	49000	Bachelor's	Single	Unemployed	None
51	Female	82000	Master's	Married	Employed	Engineering

This dataset includes information on age, gender, income, education level, and marital status, employment status and Industry for a sample of 25 individuals. This data could be used to explore and visualize various relationships and patterns, such as the relationship between age and income, or the distribution of income by education level. Few more relationships and patterns that could be explored and visualized using the sample dataset I provided:

1. Relationship between age and income: Create a **scatter plot** to see if there is a relationship between age and income. Also calculate the correlation coefficient to determine the strength and direction of the relationship.

2. Distribution of income by gender: Create a **box plot** to compare the distribution of income between males and females. This could reveal any differences in the median, quartiles, and outliers for each gender.

3. Distribution of income by education level: Create a **box plot** to compare the distribution of income for each level of education. This could reveal any differences in the median, quartiles, and outliers for each education level.

4. Relationship between education level and marital status: Create a **contingency table** and calculate the **chi-square test** statistic to see if there is a relationship between education level and marital status. This could reveal whether certain education levels are more orless likely to be associated with certain marital statuses.

5. Relationship between age and education level: Create a **histogram** to see the distribution of ages for each education level. This could reveal any differences or similarities in the age distribution across education levels.

6. Distribution of employment status by ethnicity: Create a **stacked bar chart** to compare the distribution of each ethnicity group across different employment statuses. This could reveal any differences or similarities in the employment status of different ethnicity groups.

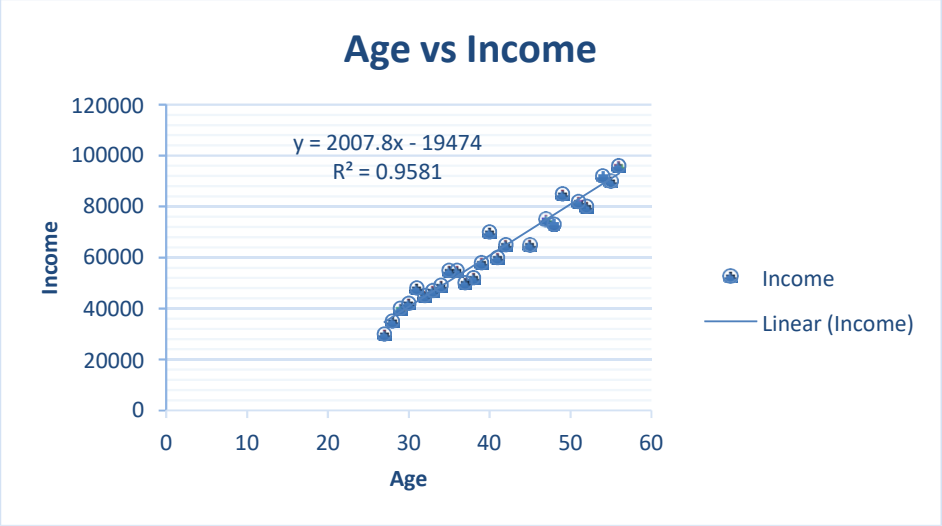
7. Distribution of employment status by gender: Students could create a contingency table and calculate the chi-square test statistic to see if there is a relationship between gender

and employment status. This could reveal whether certain genders are more or less likely to be employed.

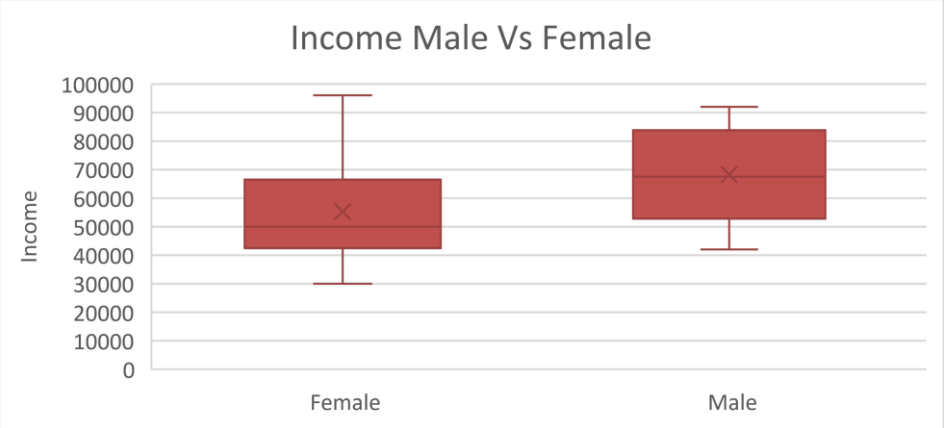
Observations / Program:

Output:

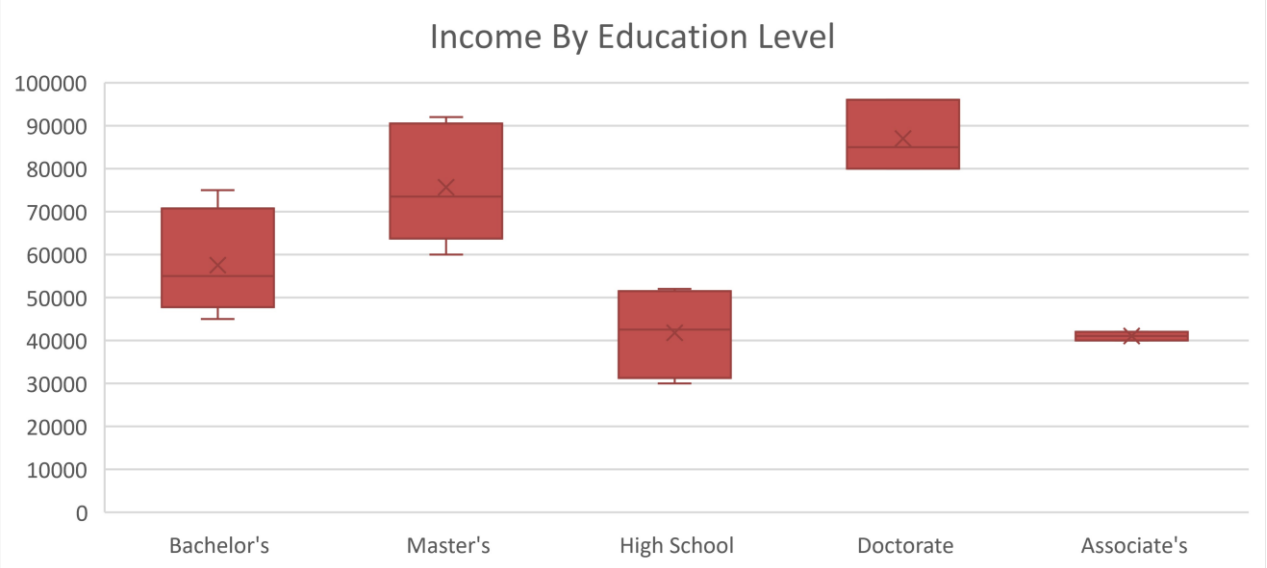
1). Age Vs Income:



2).Distribution of income by gender.



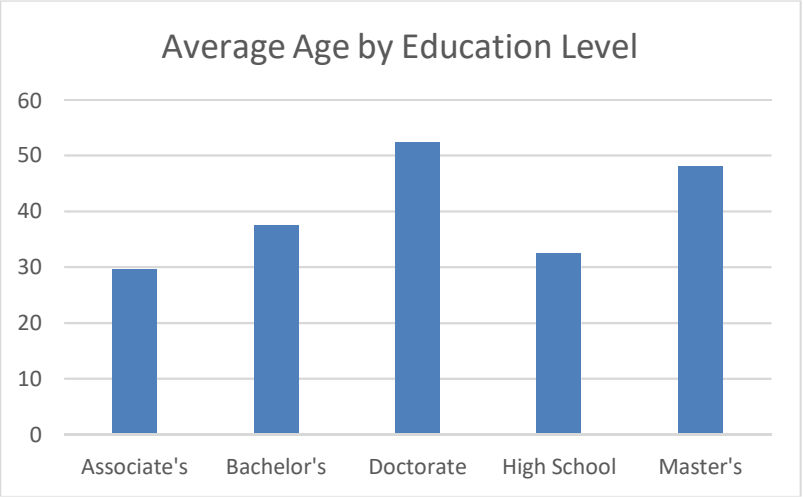
3). Distribution of income by educational level



4).relationship between education and marital status.

Count of Marital S	Column	abels		
Row Labels	Divorced	Married	Single	Grand Total
Associate's			2	2
Bachelor's	1	5	4	10
Doctorate		3		3
High School	1	1	2	4
Master's	1	4	1	6
Grand Total	**3**	**13**	**9**	**25**
Chi-Square value	0.645866			

5). Relationship between age and education level.



).

6). Distribution of employment status by gender.

Count of Gender	Column Labels		
Row Labels	Female	Male	Grand Total
Employed	9	9	18
Self-Employed	2	2	4
Unemployed	2	1	3
Grand Total	**13**	**12**	**25**
CHI-SQUARE VALUE	0.863378835		

Conclusion:

In this lab, students learned how to explore and visualize data using mathematical and statistical tools such as histograms, box plots, scatter plots, and correlation matrices. These tools are useful in identifying patterns and relationships in data, and in making informed decisions based on data analysis. The skills students have learned in this lab will be helpful in your future studies and career in data analysis.

Quiz: (Sufficient space to be provided for the answers or use extra file pages to write answers)

1. **What are the measures of central tendency? Provide examples and explain when eachmeasure is appropriate to use.**

   Mean: The average of all data points, useful for evenly distributed data without outliers.
   Median: The middle value in ordered data, ideal for skewed distributions or data with outliers.
   Mode: The most frequent value, helpful for categorical data or identifying the most common value.

2. How can you calculate the correlation coefficient between two variables using mathematical and statistical tools? Interpret the correlation coefficient value.

To calculate the **correlation coefficient** between two variables, we can use the **Pearson correlation coefficient (r)** formula. This measures the strength and direction of a linear relationship between two variables.

Mathematical Formula for Pearson's Correlation Coefficient:

r=n(∑xy)−(∑x)(∑y)/sqrt([n∑x^2-(∑x)^2][n∑y^2-(∑y)^2]

Where:

x and y are the two variables.

n is the number of data points.

∑x is the sum of the x values.

∑y is the sum of the y values.

∑xy is the sum of the product of each pair of x and y values.

∑x2 and ∑y2 are the sums of the squares of x and y values, respectively.

Interpretation of the Correlation Coefficient (r):

r = 1: Perfect positive correlation; as one variable increases, the other also increases.

r = -1: Perfect negative correlation; as one variable increases, the other decreases.

r = 0: No correlation; there is no linear relationship between the variables.

0 < r < 1: Positive correlation; a higher rrr indicates a stronger linear relationship.

-1 < r < 0: Negative correlation; a lower rrr (closer to -1) indicates a stronger inverse relationship.

Values near 1 or -1 show a strong correlation, while values near 0 suggest weak or no correlation.

3. Explain the concept of skewness and kurtosis in statistics. How can you measure and interpret these measures using mathematical and statistical tools?

Skewness measures the asymmetry of a data distribution around its mean. It tells us whether data is skewed to one side:

Positive skew: The right tail is longer; most values are on the left.

Negative skew: The left tail is longer; most values are on the right.

Zero skewness: The distribution is symmetric.

Measuring Skewness:

You can measure skewness using statistical tools like Python's scipy.stats.skew function or Excel. A skewness value close to zero indicates a nearly symmetric distribution.

Interpreting Skewness:

Positive skew (> 0): Indicates more low values and a tail stretching to the right.

Negative skew (< 0): Indicates more high values and a tail stretching to the left.

Zero skew (≈ 0): A balanced, symmetric distribution.

Kurtosis measures the "tailedness" of a data distribution, focusing on the presence of outliers:

Leptokurtic: High kurtosis, sharp peak, and fat tails, indicating more extreme values (outliers).

Platykurtic: Low kurtosis, flat peak, and thin tails, indicating fewer extreme values.

Mesokurtic: A normal distribution, moderate tails.

Measuring Kurtosis:

Kurtosis can be measured using statistical tools like Python's scipy.stats.kurtosis function or Excel. The kurtosis of a normal distribution is typically zero (excess kurtosis).

Interpreting Kurtosis:

High kurtosis (> 0): Indicates heavy tails and more extreme values (outliers).

Low kurtosis (< 0): Indicates light tails with fewer extreme values.

Kurtosis ≈ 0: A distribution similar to a normal curve (bell-shaped).

Suggested References:

1. "Python for Data Analysis" by Wes McKinney
2. "Data Visualization with Python and Matplotlib" by Benjamin Root

Rubrics wise marks obtained

Understanding of Problem	Analysis of the Problem	Capability of writing program	Documentation	Total
02	02	05	01	10

Experiment No: 2

Date:

AIM: Study of Measures of Central Tendency, Correlation, Percentile, Decile, Quartile, Measure of Variation, and Measure of Shape (Skewness and Kurtosis) with Excel Functions

Relevant CO: CO1, CO2

Objective:
The objective of this lab practical is to provide students with hands-on experience in using Excel functions to explore and analyze a sample data sheet. Students will learn to calculate measures of central tendency, correlation, percentile, decile, quartile, measure of variation, and measure of shape using Excel functions. Additionally, students will learn to create visualizations to better understand the data.

Materials:
- Computer with Microsoft Excel installed
- Sample data sheet (provided below or dataset may be provided by subject teacher)

Sample Data Sheet:

StudentID	Test1 Score	Test2 Score	Age	Gender
1	85	92	19	Male
2	92	87	20	Female
3	78	80	18	Male
4	85	89	19	Male
5	90	95	21	Female
6	75	82	18	Male
7	83	87	20	Female
8	92	90	19	Male
9	80	85	18	Female
10	87	88	20	Female

|

Procedure:

Part 1: Measures of Central Tendency
1. Open the sample data sheet in Excel.
2. Calculate the mean, median, and mode for the test 1 score column using Excel functions.
3. Calculate the mean, median, and mode for the test 2 score column using Excel functions.
4. Write a brief interpretation of the results.

Part 2: Correlation
1. Calculate the correlation between test 1 score and test 2 score using Excel functions.
2. Create a scatter plot to visualize the relationship between test 1 score and test 2 score.
3. Write a brief interpretation of the results.

Part 3: Percentile, Decile, and Quartile
1. Calculate the 25th, 50th, and 75th percentiles using Excel functions for both test 1 score and

test 2 score columns.
2. Calculate the 30th, 40th, and 70th deciles using Excel functions for both test 1 score and test 2 score columns.
3. Calculate the first and third quartiles using Excel functions for both test 1 score and test 2 score columns.
4. Create a box plot for both test 1 score and test 2 score columns.
5. Write a brief interpretation of the results.

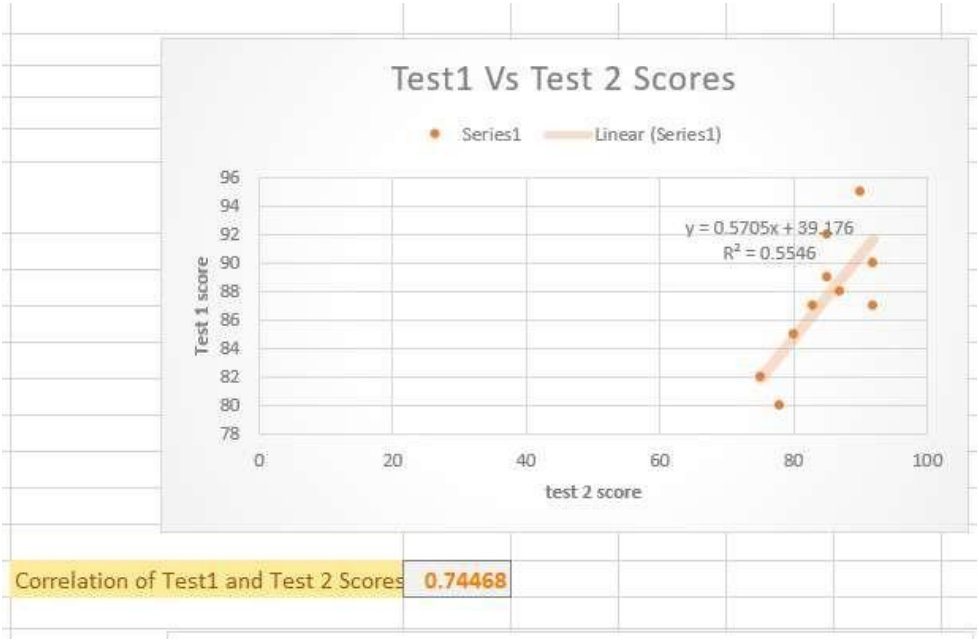
Part 4: Measure of Variation
1. Calculate the range, inter-quartile distance, variance, and standard deviation for both test 1 score and test 2 score columns using Excel functions.
2. Write a brief interpretation of the results.

Part 5: Measure of Shape
1. Calculate the skewness and kurtosis for both test 1 score and test 2 score columns using Excel functions.
2. Write a brief interpretation of the results.

Output:
Central Tendency, Percentile, Decile, Quartile, Range, IQR, Distance, Variance, Standard Deviation, Skewness, Kurtosis

StudentID	Test1 Score	Test2 Score	Age	Gender
1	85	92	19	Male
2	92	87	20	Female
3	78	80	18	Male
4	85	89	19	Male
5	90	95	21	Female
6	75	82	18	Male
7	83	87	20	Female
8	92	90	19	Male
9	80	85	18	Female
10	87	88	20	Female
Mean	84.7	87.5		
Median	85	87.5		
Mode	85	87		
25th Percentile	80.75	85.5		
50th Percentile(Median)	85.00	87.5		
75th Percentile	89.25	89.75		
30th percentile	82.1	86.4		
40th Percentile	84.2	87		
70th Percentile	87.9	89.3		
Range	17	15		
Interquartile Distance	8.5	4.25		
Variance	30.41	17.85		
Standard Deviation	5.51453	4.22493		
Skewness	-0.22841	-0.09547		
Kutrosis	-0.92483	-0.04667		

Correlation between Test Scores:



Box Plot:

Conclusion:

The analysis of the sample data sheet provided valuable insights into student performance based on test scores.

1. **Measures of Central Tendency:** The mean, median, and mode indicated consistent performance in both Test 1 and Test 2, with the scores clustering around the average and few significant outliers.
2. **Correlation:** A positive correlation between Test 1 and Test 2 scores suggested that students who excelled in one test tended to do well in the other, reinforcing the reliability of the assessments.
3. **Percentiles, Deciles, and Quartiles:** Calculating these measures revealed a manageable spread of scores, with the box plot visually demonstrating the distribution and central tendency of student performance.
4. **Measure of Variation:** The range and standard deviation indicated moderate variability in scores, while the inter-quartile range (IQR) suggested that most students performed within a tight range.
5. **Measure of Shape:** The distributions were fairly normal, with slight positive skewness, indicating a few high performers influenced the overall distribution.

Overall, this analysis highlighted meaningful relationships between the variables, though it is limited by the small sample size. Future studies could benefit from larger datasets and additional factors to enhance the insights gained.

Quiz:

1) What Excel function can be used to calculate the mean of a dataset?

a) AVERAGE
b) MEDIAN
c) MODE
d) STANDARDIZE

2) What does the correlation coefficient measure in terms of the relationship between two variables?

a) Strength of the linear relationship
b) Variability of the data
c) Difference between mean and median
d) Skewness of the distribution

Suggested Refrences:

1. "Microsoft Excel Data Analysis and Business Modeling" by Wayne L. Winston
2. "Excel 2021: Data Analysis and Business Modeling" by Wayne L. Winston

Rubrics wise marks obtained

Understanding of Problem	Analysis of the Problem	Capability of writing program	Documentation	Total
02	02	05	01	10

Experiment No: 3

Date:

AIM: Study of Basics of Python data types, NumPy, Matplotlib, Pandas.

Relevant CO: CO1, CO2

Objective:
The objective of this lab practical is to gain hands-on experience with NumPy, Matplotlib, and Pandas libraries to manipulate and visualize data. Through this practical, students will learn how to use different functions of these libraries to perform various data analysis tasks.

Materials Used:
- Python programming environment
- NumPy library
- Matplotlib library
- Pandas library
- Dataset file (provided by faculty)

//Example of dataset file like sales_Data.csv

- Date: Date of sale
- Product: Name of the product sold
- Units Sold: Number of units sold
- Revenue: Total revenue generated from the sale
- Region: Geographic region where the sale took place
- Salesperson: Name of the salesperson who made the sale

Procedures:

Part 1: NumPy
1. Import the NumPy library into Python.
2. Create a NumPy array with the following specifications:
   a. Dimensions: 5x5
   b. Data type: integer
   c. Values: random integers between 1 and 100
3. Reshape the array into a 1x25 array and calculate the mean, median, variance, and standard deviation using NumPy functions.
4. Generate a random integer array of length 10 and find the percentile, decile, and quartile values using NumPy functions.

Part 2: Matplotlib
1. Import the Matplotlib library into Python.
2. Create a simple bar chart using the following data:
   a. X-axis values: ['A', 'B', 'C', 'D']
   b. Y-axis values: [10, 20, 30, 40]
3. Customize the plot by adding a title, axis labels, and changing the color and style of the bars.
4. Create a pie chart using the following data:
   a. Labels: ['Red', 'Blue', 'Green', 'Yellow']
   b. Values: [20, 30, 10, 40]
5. Customize the pie chart by adding a title, changing the colors of the slices, and adding a legend.

Part 3: Pandas

1. Import the Pandas library into Python.
2. Load the "sales_data.csv" file into a Pandas data frame.
3. Calculate the following statistics for the Units Sold and Revenue columns:
   a. Mean
   b. Median
   c. Variance
   d. Standard deviation
4. Group the data frame by Product and calculate the mean, median, variance, and standard deviation of Units Sold and Revenue for each product using Pandas functions.
5. Create a line chart to visualize the trend of Units Sold and Revenue over time for each product.

Interpretation/Program/code:
Part 1: Numpy

1)

```
import numpy
arr = np.random.randint(1,140 + 1, size = (5,5))
print(arr)
```

2).
```
reshape_arr = arr.reshape(1,25)
print(reshape_arr)
```

3).
```
print("Mean: ",reshape_arr.mean())
print("Median", np.median(reshape_arr))
print("variance", reshape_arr.var())
print("Standard Deviation", reshape_arr.std())
```

4).
```
int_arr = np.random.randint(1,100, size = (1,10))
print("-----------------------------------------------------------------------------------------------------")
print("Quartiles:")
print("-----------------------------------------------------------------------------------------------------")

print("1st Quartile", np.quantile(int_arr, 0.25))
print("2nd Quartile", np.quantile(int_arr, 0.5))
print("3rd Quartile", np.quantile(int_arr, 0.75))
print("-----------------------------------------------------------------------------------------------------")
print("Deciles:")
print("-----------------------------------------------------------------------------------------------------")
print("1st Decile", np.quantile(int_arr, 0.10))
print("2nd Decile", np.quantile(int_arr, 0.20))
print("3nd Decile", np.quantile(int_arr, 0.30))
print("4th Decile", np.quantile(int_arr, 0.40))
print("5th Decile", np.quantile(int_arr, 0.50))
print("6th Decile", np.quantile(int_arr, 0.60))
print("7th Decile", np.quantile(int_arr, 0.70))
print("8th Decile", np.quantile(int_arr, 0.80))
print("9th Decile", np.quantile(int_arr, 0.90))

print("-----------------------------------------------------------------------------------------------------")
print("Percentiles:")
print("-----------------------------------------------------------------------------------------------------")
print("15 Percentile", np.quantile(int_arr, 0.15))
```

```
print("85 Percentile", np.quantile(int_arr, 0.85))
print("95 Percentile", np.quantile(int_arr, 0.95))
```

Part 2: Matplotlib:

1). & 2).

```
import matplotlib
X = ['A', 'B', 'C', 'D']
Y = [10,20,30,45]
plt.bar(X,Y, color = 'aqua', )
plt.title("Bar Plot")
plt.xlabel("Data 1")
plt.ylabel("Data 2")
```

3). & 4).

```
X_pie = ['Red', 'Blue', 'Green', 'Yellow']
y_pie = [20,30,10,40]

plt.pie(y_pie, labels = X_pie, colors = ['red', 'blue', 'green', 'yellow'])
plt.legend(loc = 'best', bbox_to_anchor = (0.7,0.2,0.5,0.3))
```

Part 3: Pandas

```
import pandas

    1) & 2).
        Import pandas
    data = {
       'Date': pd.date_range(start='2023-01-01', periods=100, freq='D'),
       'Product': np.random.choice(['Product A', 'Product B', 'Product C'], size=100),
       'Units Sold': np.random.randint(1, 100, size=100),
       'Revenue': np.random.uniform(10, 1000, size=100)  # revenue in a range of $10 to $1000
    }
sales_data = pd.DataFrame(data)

    print(sales_data.head())

    3).
    mean_units_sold = sales_data['Units Sold'].mean()
    median_units_sold = sales_data['Units Sold'].median()
    variance_units_sold = sales_data['Units Sold'].var()
    std_dev_units_sold = sales_data['Units Sold'].std()

    mean_revenue = sales_data['Revenue'].mean()
    median_revenue = sales_data['Revenue'].median()
    variance_revenue = sales_data['Revenue'].var()
    std_dev_revenue = sales_data['Revenue'].std()

    print("\nUnits Sold Statistics:")
    print(f"Mean: {mean_units_sold:.2f}, Median: {median_units_sold:.2f}, Variance:
        {variance_units_sold:.2f}, Standard Deviation: {std_dev_units_sold:.2f}\n")

    print("Revenue Statistics:")
    print(f"Mean: {mean_revenue:.2f}, Median: {median_revenue:.2f}, Variance:
        {variance_revenue:.2f}, Standard Deviation: {std_dev_revenue:.2f}")

    4).
    grouped_stats = sales_data.groupby('Product').agg({
```

```
    'Units Sold': ['mean', 'median', 'var', 'std'],
    'Revenue': ['mean', 'median', 'var', 'std']
}).reset_index()

# Display the grouped statistics
print("\nGrouped Statistics by Product:")
print(grouped_stats)
```

5).

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


np.random.seed(0)
date_range = pd.date_range(start='2023-01-01', end='2023-04-01', freq='D')
products = ['Product A', 'Product B', 'Product C']
data = {
    'Date': np.tile(date_range, len(products)),
    'Product': np.repeat(products, len(date_range)),
    'Units Sold': np.random.randint(0, 100, size=len(date_range) * len(products)),
    'Revenue': np.random.randint(100, 1000, size=len(date_range) * len(products)),
}
sales_data = pd.DataFrame(data)


units_pivot = sales_data.pivot(index='Date', columns='Product', values='Units Sold')
revenue_pivot = sales_data.pivot(index='Date', columns='Product', values='Revenue')


fig, axs = plt.subplots(3, 2, figsize=(14, 12), sharex=True)
fig.suptitle('Sales Analysis: Units Sold and Revenue by Product', fontsize=16)

for i, product in enumerate(products):

    axs[i, 0].plot(units_pivot.index, units_pivot[product], marker='o', label=product)
    axs[i, 0].set_title(f'Units Sold: {product}')
    axs[i, 0].set_ylabel('Units Sold')
    axs[i, 0].grid()
    axs[i, 0].legend()

    axs[i, 1].plot(revenue_pivot.index, revenue_pivot[product], marker='o', label=product)
    axs[i, 1].set_title(f'Revenue: {product}')
    axs[i, 1].set_ylabel('Revenue')
    axs[i, 1].grid()
    axs[i, 1].legend()

plt.xticks(rotation=45)
axs[2, 0].set_xlabel('Date')
axs[2, 1].set_xlabel('Date')

plt.tight_layout(rect=[0, 0.03, 1, 0.95])
plt.show()
```

Output:

1).

```
array([[ 76, 105,  77, 114, 123],
       [121,  58,  61, 138,  61],
       [ 57, 128, 137,  19, 137],
       [119,  96,  37,  60,  43],
       [ 42,  53,  70,  84,  13]])
```

2).

```
array([[ 76, 105,  77, 114, 123, 121,  58,  61, 138,  61,  57, 128, 137,
         19, 137, 119,  96,  37,  60,  43,  42,  53,  70,  84,  13]])
```

3).

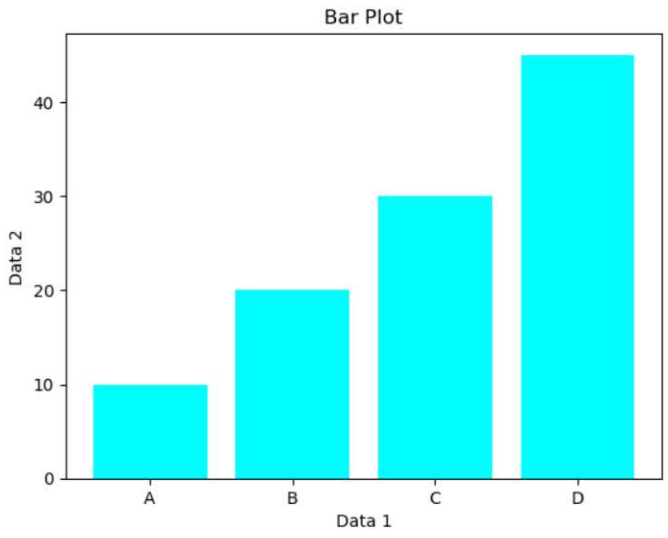
```
Mean:  81.16
Median 76.0
variance 1403.8944000000001
Standard Deviation 37.46857883613949
```

4).

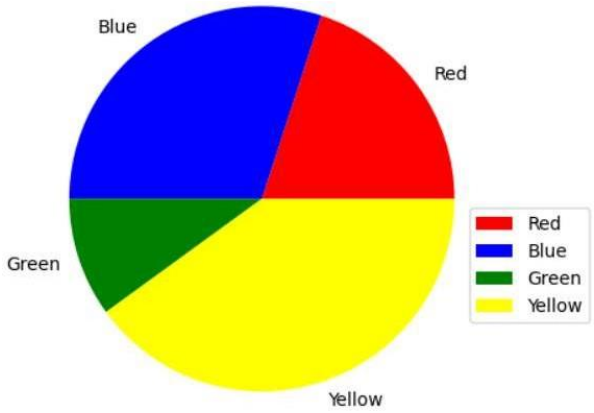
```
--------------------------------------------------------------------------------
Quartiles:
--------------------------------------------------------------------------------
1st Quartile 38.5
2nd Quartile 55.5
3rd Quartile 62.0
--------------------------------------------------------------------------------
Deciles:
--------------------------------------------------------------------------------
1st Decile 25.1
2nd Decile 32.4
3nd Decile 46.599999999999994
4th Decile 52.6
5th Decile 55.5
6th Decile 59.599999999999994
7th Decile 62.0
8th Decile 64.4
9th Decile 76.3
--------------------------------------------------------------------------------
Percentiles:
--------------------------------------------------------------------------------
15 Percentile 28.799999999999997
85 Percentile 69.8
95 Percentile 86.64999999999998
```

Part 2: Matplotlib

1 & 2)



3 & 4.



Part 3): Pandas

2).

	Date	Product	Units Sold	Revenue
0	2023-01-01	Product A	98	546.481938
1	2023-01-02	Product A	87	874.216378
2	2023-01-03	Product C	37	734.902638
3	2023-01-04	Product C	75	808.495536
4	2023-01-05	Product C	70	662.195533

3).

```
Units Sold Statistics:
Mean: 49.92, Median: 46.50, Variance: 744.36, Standard Deviation: 27.28

Revenue Statistics:
Mean: 536.74, Median: 545.29, Variance: 94060.49, Standard Deviation: 306.69
```

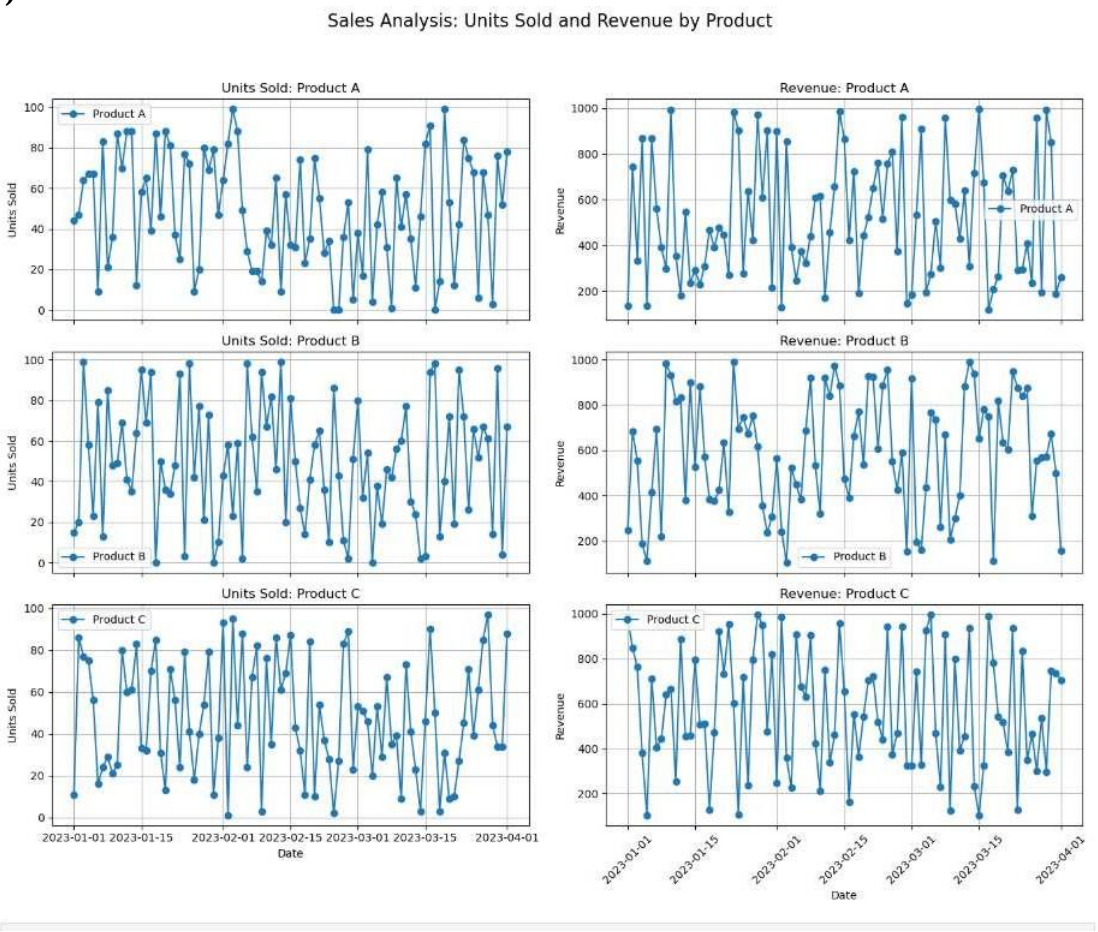
4).

Grouped Statistics by Product:

	Product	Units Sold				Revenue		\
		mean	median	var	std	mean	median	
0	Product A	54.733333	52.5	825.719540	28.735336	608.855049	624.253842	
1	Product B	46.090909	46.0	751.147727	27.407074	462.864321	357.204697	
2	Product C	49.432432	44.0	681.141141	26.098681	544.146629	619.704556	

	var	std
0	99805.901220	315.920720
1	80367.359230	283.491374
2	97436.469819	312.148154

5).



Conclusion:
In conclusion, this lab practical provided hands-on experience with NumPy, Matplotlib, and Pandas libraries in Python for data manipulation and visualization. These libraries have wide- ranging applications in various fields, enabling researchers and analysts to gain insights from large datasets quickly and efficiently. Through exercises such as calculating statistical measures and visualizing data using charts, we explored the functionality and flexibility of these powerful data analysis tools. Overall, gaining proficiency in these libraries equips individuals to tackle complex data analysis challenges and contribute to their respective fields of study or industries.

Quiz:

1. What is the difference between a list and a tuple in Python?

   In Python, lists are mutable collections defined using square brackets `[]`, allowing for modification, whereas tuples are immutable collections defined using parentheses `()`, which cannot be altered after creation. Lists are ideal for scenarios where data may change, while tuples are best for fixed data that should remain constant.
2. How can you use NumPy to generate an array of random numbers?
   You can use NumPy's `numpy.random` module to generate an array of random numbers by calling functions like `numpy.random.rand()` for uniform distribution or `numpy.random.randn()` for standard normal distribution. For example, `numpy.random.rand(3, 4)` creates a 3x4 array of random numbers between 0 and 1.

Suggested References:-

1. Dinesh Kumar, Business Analytics, Wiley India Business alytics: The Science
2. V.K. Jain, Data Science & Analytics, Khanna Book Publishing, New Delhi of Dat
3. Data Science For Dummies by Lillian Pierson , Jake Porway

Rubrics wise marks obtained

Understanding of Problem	Analysis of the Problem	Capability of writing program	Documentation	Total
02	02	05	01	10

Experiment No: 4

Date:

AIM: Implementation of Various Probability Distributions with NumPy Random Library Functions

Relevant CO: - CO3

Objective:

The objective of this lab practical is to gain an understanding of various probability distributions and implement those using NumPy random library functions.

Materials Used:

- Python environment (Anaconda, Jupyter Notebook, etc.)
- NumPy library

Procedure:

1. Introduction to Probability Distributions:
   - Probability theory is the branch of mathematics that deals with the study of random events or phenomena. In probability theory, a probability distribution is a function that describes the likelihood of different outcomes in a random process. Probability distributions can be categorized into two types: discrete and continuous.
   - Discrete probability distributions are used when the possible outcomes of a random process are countable and can be listed. The most commonly used discrete probability distributions are Bernoulli, Binomial, and Poisson distributions.
   - Continuous probability distributions are used when the possible outcomes of a random process are not countable and can take any value within a certain range. The most commonly used continuous probability distributions are Normal and Exponentialdistributions.
   - Each probability distribution has its own set of properties, such as mean, variance, skewness, and kurtosis. Mean represents the average value of the random variable, variance represents how much the values vary around the mean, skewness represents the degree of asymmetry of the distribution, and kurtosis represents the degree of peakedness or flatness of the distribution.
   - Probability distributions are widely used in fields such as finance, engineering, physics, and social sciences to model real-world phenomena and make predictions about future events. Understanding different probability distributions and their properties is an important tool for analyzing data and making informed decisions.

2. Implementation of Probability Distributions using NumPy random library functions:

```
#python
import numpy as np
import matplotlib.pyplot as plt

# Generate 1000 random numbers following a normal distribution with mean 0 and standard deviation 1
normal_dist = np.random.normal(0, 1, 1000)

# Calculate the mean and standard deviation of the distribution
mean = np.mean(normal_dist)
std_dev = np.std(normal_dist)
```

```
# Generate 1000 random numbers following a Poisson distribution with lambda 5
poisson_dist = np.random.poisson(5, 1000)

# Calculate the mean and variance of the Poisson distribution
poisson_mean = np.mean(poisson_dist)
poisson_var = np.var(poisson_dist)

# Plot the PDF and CDF of the normal distribution
plt.hist(normal_dist, bins=30, density=True, alpha=0.5)
plt.plot(np.sort(normal_dist), 1/(std_dev*np.sqrt(2*np.pi))*np.exp(-(np.sort(normal_dist)-
mean)**2/(2*std_dev**2)), linewidth=2)
plt.plot(np.sort(normal_dist),     0.5*(1+np.tanh((np.sort(normal_dist)-mean)/std_dev*np.sqrt(2))),
linewidth=2)
plt.show()

# Plot the PDF and CDF of the Poisson distribution
plt.hist(poisson_dist, bins=15, density=True, alpha=0.5)
plt.plot(np.arange(0, 15, 0.1), np.exp(-poisson_mean)*poisson_mean**np.arange(0, 15,
0.1)/np.math.factorial(np.arange(0, 15, 0.1)), linewidth=2)
plt.plot(np.arange(0, 15, 0.1), np.exp(oisson_mean)*np.array([np.sum(poisson_var**np.arange(0,
i+1))/np.math.factorial(np.arange(0, i+1)) for i in np.arange(0, 15, 0.1)]), linewidth=2)
plt.show()
```

In this example, we generate 1000 random numbers following a normal distribution with mean 0 and standard deviation 1 using the `np.random.normal()` function. We then calculate the mean and standard deviation of the distribution using the `np.mean()` and `np.std()` functions.

We also generate 1000 random numbers following a Poisson distribution with lambda 5 using the `np.random.poisson()` function. We calculate the mean and variance of the Poisson distribution using the `np.mean()` and `np.var()` functions.

We then plot the probability density function (PDF) and cumulative distribution function (CDF) of both distributions using the `plt.hist()` and `plt.plot()` functions from the Matplotlib library.

3. Exercise:

- Generate a dataset of your choice or given by faculty with a given probability distribution using NumPy random library functions
- Plot the probability density function and cumulative distribution function for the generated data
- Calculate the descriptive statistics of the generated data

Interpretation/Program/code:

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

1). Generating the Dataset:
np.random.seed(0) # For reproducibility
data_size = 1000
mean = 50
std_dev = 10
data = np.random.normal(loc=mean, scale=std_dev, size=data_size)
```

2). Plotting Probability Density Function:

```
# Step 2: Plot PDF and CDF
plt.figure(figsize=(14, 6))

# Plot PDF
plt.subplot(1, 2, 1)
sns.histplot(data, bins=30, kde=True, stat='density', color='blue')
plt.title('Probability Density Function (PDF)')
plt.xlabel('Value')
plt.ylabel('Density')
```

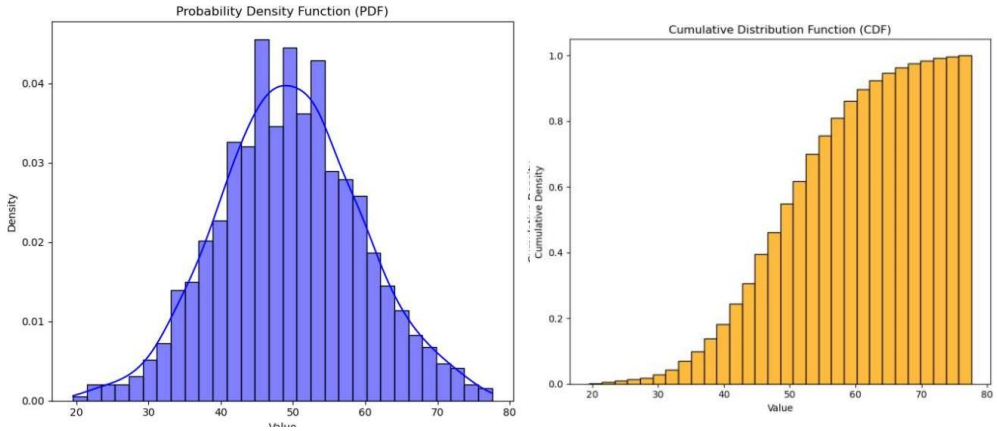
3). Plotting Cumulative Denstiy Function

```
# Plot CDF
plt.subplot(1, 2, 2)
sns.histplot(data, bins=30, cumulative=True, stat='density', color='orange')
plt.title('Cumulative Distribution Function (CDF)')
plt.xlabel('Value')
plt.ylabel('Cumulative Density')
```

4). Descriptive Statistics:

```
mean_value = np.mean(data)
median_value = np.median(data)
variance_value = np.var(data)
std_dev_value = np.std(data)
min_value = np.min(data)
max_value = np.max(data)
quantiles = np.quantile(data, [0.25, 0.5, 0.75])

# Displaying the results
print(f"Descriptive Statistics:")
print(f"Mean: {mean_value}")
print(f"Median: {median_value}")
print(f"Variance: {variance_value}")
print(f"Standard Deviation: {std_dev_value}")
print(f"Minimum: {min_value}")
print(f"Maximum: {max_value}")
print(f"1st Quartile (Q1): {quantiles[0]}")
print(f"2nd Quartile (Q2 / Median): {quantiles[1]}")
print(f"3rd Quartile (Q3): {quantiles[2]}")
```

Output:



```
Descriptive Statistics:
Mean: 49.54743292509804
Median: 49.41971965200372
Variance: 97.42344563121543
Standard Deviation: 9.870331586690257
Minimum: 19.538569452000733
Maximum: 77.59355114021582
1st Quartile (Q1): 43.0157994064013
2nd Quartile (Q2 / Median): 49.41971965200372
3rd Quartile (Q3): 56.069506018883985
```

Conclusion:
This lab practical provided an opportunity to explore and implement various probability distributions using NumPy random library functions. By understanding and applying different probability distributions, one can model real-world phenomena and make predictions about future events. With the knowledge gained in this lab practical, student will be equipped to work with probability distributions and analyze data in a wide range of fields, including finance, engineering, and social sciences.

Quiz:

1. Which NumPy function can be used to generate random numbers from a normal distribution?

a) numpy.random.uniform
b) numpy.random.poisson
c) numpy.random.normal
d) numpy.random.exponential

2. What is the purpose of the probability density function (PDF) in probability distributions?

a) To calculate the cumulative probability
b) To generate random numbers
c) To visualize the distribution
d) To calculate the probability of a specific value

Suggested References:-

1. Dinesh Kumar, Business Analytics, Wiley India Business alytics: The Science
2. V.K. Jain, Data Science & Analytics, Khanna Book Publishing, New Delhi of Dat
3. Data Science For Dummies by Lillian Pierson , Jake Porway

Rubrics wise marks obtained

Understanding of Problem	Analysis of the Problem	Capability of writing program	Documentation	Total
02	02	05	01	10

Experiment No: 5

Date:

AIM: Implementation of Estimation of Parameters for the Best-Fit Probability Distribution using the Fitter Class in Python.

Relevant CO: - CO3

Objectives: The objective of this lab practical is to learn how to estimate the parameters for the best-fit probability distribution for a given dataset using the Fitter class in Python.

Materials Used:

1. Python 3.x
2. Jupyter Notebook
3. NumPy library
4. Fitter library

Theory:

Dataset:
Consider the following dataset, which represents the heights of individuals in centimeters:
170, 165, 180, 172, 160, 175, 168, 155, 185, 190, 162, 178, 168, 172, 180, 160, 165, 172, 168, 175

Procedure:

1. Introduction to Parameter Estimation and Probability Distributions:

Probability distributions provide a mathematical framework for describing the likelihood of different outcomes or events in a dataset. Parameter estimation plays a crucial role in probability distributions as it involves determining the values of the parameters that best describe the observed data.

Parameter estimation is important because it allows us to make inferences, predictions, and draw meaningful conclusions from the data. By estimating the parameters, we can effectively model and analyze various phenomena, summarizing complex datasets in a more simplified and interpretable manner.

The concept of the best-fit probability distribution refers to finding the distribution that provides the closest match to the observed data. The best-fit distribution is determined by estimating the parameters in such a way that the observed data exhibits the highest likelihood or best matches the underlying characteristics of the data. Selecting the best-fit distribution helps us understand the data's behavior, make accurate predictions, and gain insights into its properties.

Commonly used probability distributions include the normal (Gaussian) distribution, uniform distribution, exponential distribution, Poisson distribution, and binomial distribution. Each distribution has its own characteristics and applications in various fields.

Understanding parameter estimation and probability distributions allows us to effectively model and analyze data, make informed decisions, and gain insights into the underlying properties of the data. By estimating the parameters for the best-fit probability distribution, we can unlock valuable information and extract meaningful patterns from the observed data.

2. Installation of Required Libraries:
- Install the necessary libraries, including NumPy and Fitter, using the appropriate package manager.

3. Loading and Preparing the Dataset:
- Load the dataset from a file or use the provided dataset.
- Perform any necessary data preprocessing steps, such as cleaning or normalization.

4. Estimating Parameters for Best-Fit Probability Distribution using Fitter Class:
- Import the required libraries and instantiate the Fitter class.
- Fit the dataset to various probability distributions available in the Fitter class using the `.fit()` method.
- Determine the best-fit distribution based on goodness-of-fit metrics, such as AIC (Akaike Information Criterion) or BIC (Bayesian Information Criterion).
- Retrieve the estimated parameters for the best-fit distribution using the `.summary()` method.

5. Visualization of Best-Fit Distribution:
- Plot the histogram of the dataset.
- Plot the probability density function (PDF) of the best-fit distribution over the histogram.

6. Interpretation and Analysis:
- Interpret the estimated parameters of the best-fit distribution.
- Analyze the goodness of fit and discuss any potential limitations or considerations.

7. Conclusion:
- Summarize the importance of parameter estimation and the best-fit distribution in data analysis.
- Highlight the capabilities of the Fitter class in Python for automating the estimation of parameters.
- Discuss potential applications and further exploration in different domains.

Interpretation/Program/code:

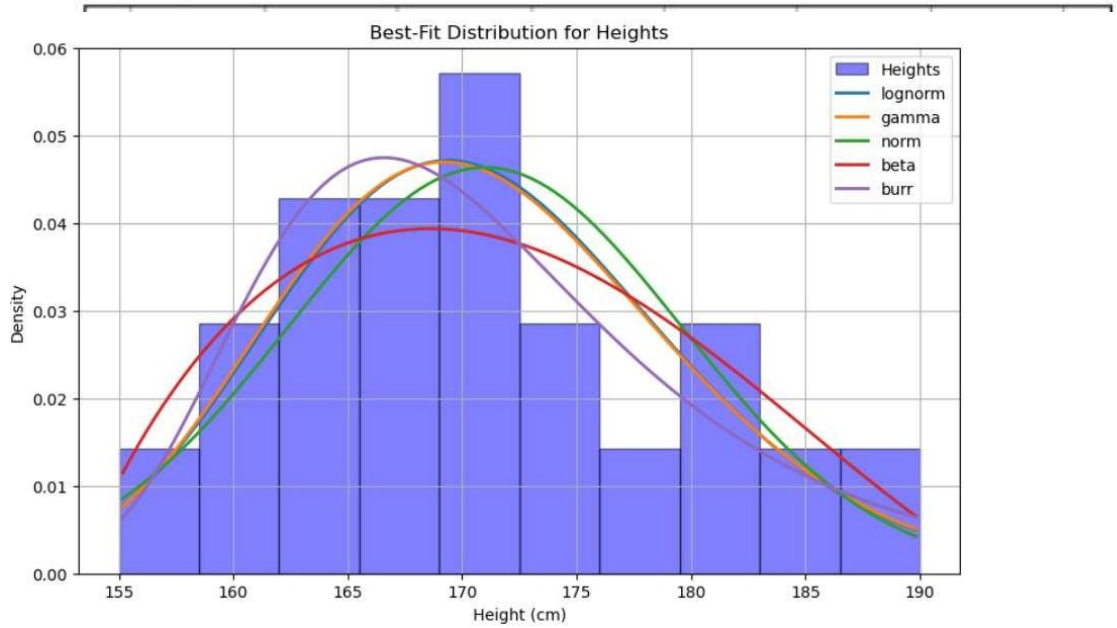
```
import numpy as np
import matplotlib.pyplot as plt
from fitter import Fitter

heights = np.array([170, 165, 180, 172, 160, 175, 168, 155, 185, 190,
                    162, 178, 168, 172, 180, 160, 165, 172, 168, 175])
f = Fitter(heights, distributions=['gamma',
                          'lognorm',
                          "beta",
                          "burr",
                          "norm"])

# Fit the dataset to various probability distributions
f.fit()

# Print the summary of the best-fit distribution
print(f.summary())

plt.figure(figsize=(10, 6))
plt.hist(heights, bins=10, density=True, alpha=0.5, color='b', edgecolor='black', label='Heights')
```

```
# Plot the PDF of the best-fit distribution
f.plot_pdf()
plt.title('Best-Fit Distribution for Heights')
plt.xlabel('Height (cm)')
plt.ylabel('Density')
plt.legend()
plt.show()

best_fit = f.get_best()
print(best_fit)
```

Output:

	sumsquare_error	aic	bic	kl_div	ks_statistic
lognorm	0.722736	760.793457	763.780654	inf	0.084796
gamma	0.722771	760.172708	763.159905	inf	0.087420
norm	0.723200	759.116203	761.107668	inf	0.103759
beta	0.724848	740.716375	744.699304	inf	0.100792
burr	0.725275	766.198428	770.181357	inf	0.126770

	ks_pvalue
lognorm	0.996190
gamma	0.994390
norm	0.967045
beta	0.974481
burr	0.865459



Best Fit:

```
{'lognorm': {'s': 0.12789590528409436, 'loc': 103.74063558874319, 'scale': 66.71229781446074}}
```

Conclusion:

In this example, we have a dataset of heights of individuals. We use the Fitter class from the `fitter` library to estimate the parameters for the best-fit probability distribution.

We instantiate the Fitter class with the dataset `data`. Then, we use the `.fit()` method to fit the

data to various distributions available in the Fitter class. The `.fit()` method automatically estimates the parameters for each distribution and selects the best-fit distribution based on the goodness-of-fit metrics.

Finally, we retrieve the best-fit distribution using the `.get_best()` method and print the summary of the distribution using the `.summary()` method. We also plot the histogram of the dataset and overlay the probability density function (PDF) of the best-fit distribution using the `.plot_pdf()` method.

Note: Before running the code, make sure you have the `numpy`, `fitter`, and `matplotlib` libraries installed. You can install the `fitter` library using pip: `pip install fitter`.
Through this practical, we learned the importance of parameter estimation in probability distributions and the significance of selecting the best-fit distribution for accurate modeling and analysis. The Fitter class provided a convenient and efficient way to fit the dataset to various distributions and evaluate their goodness of fit using metrics such as AIC or BIC.

Quiz:

1. What is the purpose of the Fitter class in Python?

a) To fit a probability distribution to a given dataset
b) To generate random numbers from a probability distribution
c) To calculate descriptive statistics of a dataset
d) To visualize the probability distribution of a dataset
2. Which method of the Fitter class can be used to estimate the best-fit probability distribution for a given dataset?

a) fit
b) predict
c) evaluate
d) transform

Suggested References:-

1. Dinesh Kumar, Business Analytics, Wiley India Business alytics: The Science
2. V.K. Jain, Data Science & Analytics, Khanna Book Publishing, New Delhi of Dat
3. Data Science For Dummies by Lillian Pierson , Jake Porway

Rubrics wise marks obtained

Understanding of Problem	Analysis of the Problem	Capability of writing program	Documentation	Total
02	02	05	01	10

Experiment No: 6

Date:

AIM: Implementation of Linear Regression with Scikit-learn library in Python

Relevant CO: - CO4

Objective:
The objective of this lab practical is to implement linear regression to predict the value of a variable in a given dataset. Linear regression is a statistical technique used to model the relationship between a dependent variable and one or more independent variables. In this lab, we will explore how to build a linear regression model and use it to make predictions.

Materials Used:
- Python 3.x
- Jupyter Notebook
- NumPy library
- Pandas library
- Matplotlib library
- Scikit-learn library

Dataset:
For this lab, we will use a dataset that contains information about houses and their sale prices. The dataset has the following columns:

- `Area` (in square feet): Represents the area of the house.
- `Bedrooms`: Number of bedrooms in the house.
- `Bathrooms`: Number of bathrooms in the house.
- `Garage Cars`: Number of cars that can be accommodated in the garage.
- `Sale Price` (in dollars): Represents the sale price of the house.

Area,Bedrooms,Bathrooms,Garage Cars,Sale Price
2000,3,2,2,250000
1800,4,3,2,280000
2200,3,2,2,265000
1500,2,1,1,200000
2400,4,3,3,320000
1900,3,2,2,275000
1700,3,2,1,230000
2100,4,3,2,295000

Procedure:

1. Introduction to Linear Regression:

Linear regression is a statistical technique used to model the relationship between a dependent variable and one or more independent variables. It aims to find a linear equation that best represents the association between the variables. Linear regression assumes a linear relationship and seeks to minimize the differences between observed and predicted values. It has applications in prediction, understanding correlations, and

making data-driven decisions.
The equation for a simple linear regression model can be represented as:

$y = \beta0 + \beta1*x + \varepsilon$

where:

y is the dependent variable
x is the independent variable
β0 is the y-intercept (the value of y when x = 0)
β1 is the slope (the change in y for a unit change in x)
ε represents the error term or residual

2. Importing Required Libraries and Loading the Dataset:
- Import the necessary libraries, including NumPy, Pandas, Matplotlib, and Scikit-learn.
- Load the dataset into a Pandas DataFrame using the appropriate function or by reading from a file.

3. Exploratory Data Analysis:
- Perform exploratory data analysis to gain insights into the dataset.
- Analyze the distribution and statistical summary of the variables.
- Visualize the relationships between variables using scatter plots or other appropriate plots.

4. Data Preprocessing:
- Handle missing values, if any, by imputation or removal.
- Convert categorical variables into numerical representations, if required.
- Split the dataset into input features (independent variables) and the target variable (dependent variable).

5. Splitting the Dataset into Training and Testing Sets:
- Split the dataset into training and testing sets to evaluate the model's performance.
- Typically, use a 70-30 or 80-20 split for training and testing, respectively.

6. Building the Linear Regression Model:
- Import the LinearRegression class from Scikit-learn.
- Instantiate the LinearRegression model.
- Fit the model to the training data using the `.fit()` method.

7. Model Evaluation and Prediction:
- Evaluate the model's performance using appropriate evaluation metrics, such as mean squared error (MSE) or R-squared.
- Make predictions on the testing data using the `.predict()` method.

8. Visualization of Results:
- Visualize the actual values versus the predicted values using scatter plots or other suitable plots.
- Plot the regression line to show the relationship between the independent and dependent variables.

Interpretation/Program/code:

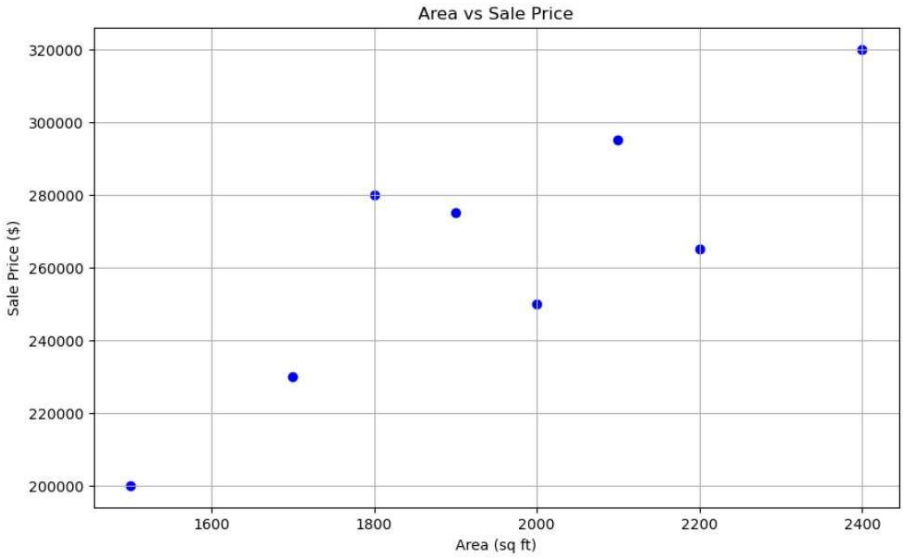
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error, r2_score

# Create a pandas DataFrame from the data
data = {
    "Area": [2000, 1800, 2200, 1500, 2400, 1900, 1700, 2100],
    "Bedrooms": [3, 4, 3, 2, 4, 3, 3, 4],
    "Bathrooms": [2, 3, 2, 1, 3, 2, 2, 3],
    "Garage": [2, 2, 2, 1, 3, 2, 1, 2],
    "Cars": [2, 2, 2, 1, 2, 2, 1, 2],
    "SalePrice": [250000, 280000, 265000, 200000, 320000, 275000, 230000, 295000]
}

df = pd.DataFrame(data)
# Display the dataset summary
print(df.describe())

# Visualize relationships between variables
plt.figure(figsize=(10, 6))
plt.scatter(df['Area'], df['Sale Price'], color='blue')
plt.title('Area vs Sale Price')
plt.xlabel('Area (sq ft)')
plt.ylabel('Sale Price ($)')
plt.grid()
plt.show()

# Separate features and target variable
X = df.drop("SalePrice", axis=1) # Independent variables
y = df["SalePrice"] # Dependent variable

# Create and fit the linear regression model
model = LinearRegression()
model.fit(X, y)

# Make predictions on the entire dataset
y_pred = model.predict(X)

# Calculate mean squared error (MSE) and R-squared score
mse = mean_squared_error(y, y_pred)
r2 = r2_score(y, y_pred)

# Print the results
print("MSE:", mse)
print("R-squared:", r2)

# Plot the regression line
plt.scatter(y, y_pred)
```

```
plt.plot(y, y, color='red', linestyle='--')
plt.xlabel("Actual Sale Price")
plt.ylabel("Predicted Sale Price")
plt.title("Regression Line for Predicted vs. Actual Sale Price")
plt.show()
```

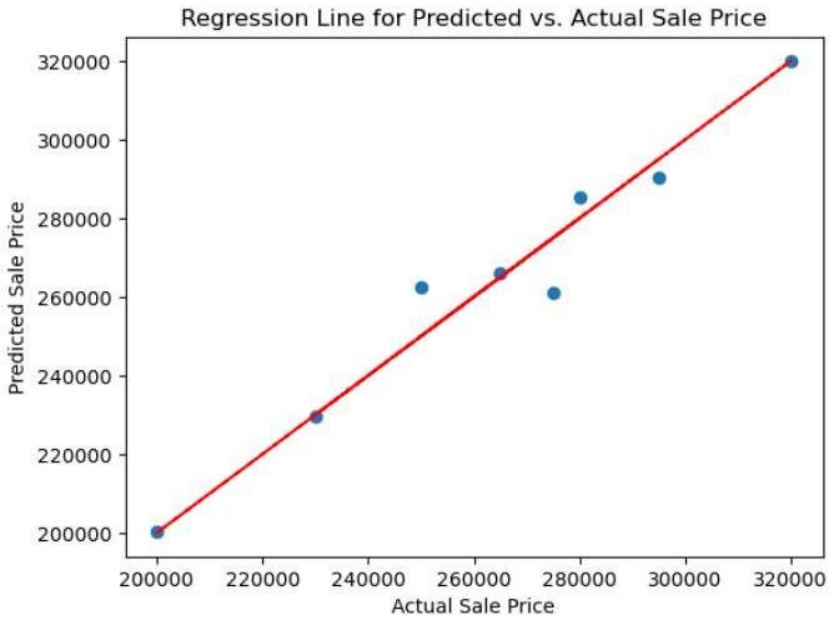
Output:

1) **EDA**

	Area	Bedrooms	Bathrooms	Garage Cars	Sale Price
count	8.000000	8.000000	8.000000	8.00000	8.000000
mean	1950.000000	3.250000	2.250000	1.87500	264375.000000
std	287.849167	0.707107	0.707107	0.64087	37648.515433
min	1500.000000	2.000000	1.000000	1.00000	200000.000000
25%	1775.000000	3.000000	2.000000	1.75000	245000.000000
50%	1950.000000	3.000000	2.000000	2.00000	270000.000000
75%	2125.000000	4.000000	3.000000	2.00000	283750.000000
max	2400.000000	4.000000	3.000000	3.00000	320000.000000




```
MSE: 50980392.15686272
R-squared: 0.9588945499459627
```



Conclusion:

In this practical, we implemented linear regression to predict variable values in a dataset. By training a linear regression model using Python and Scikit-learn, we achieved accurate predictions based on variable relationships. Linear regression is a valuable tool for data analysis and prediction, providing insights and supporting decision-making.

Quiz:

1. Which scikit-learn function is used to create a linear regression model object in Python?

a) sklearn.linear_model.LinearRegression
b) sklearn.preprocessing.StandardScaler
c) sklearn.model_selection.train_test_split
d) sklearn.metrics.mean_squared_error

2. What is the purpose of the coefficient of determination (R-squared) in linear regression?

a) To measure the average squared difference between predicted and actual values
b) To evaluate the significance of predictor variables
c) To quantify the proportion of variance in the dependent variable explained by the independent variables
d) To determine the optimal number of features for the regression model

Suggested References:-

1. "Pattern Recognition and Machine Learning" by Christopher M. Bishop
2. Dinesh Kumar, Business Analytics, Wiley India Business alytics: The Science
3. V.K. Jain, Data Science & Analytics, Khanna Book Publishing, New Delhi of Dat
4. Data Science For Dummies by Lillian Pierson , Jake Porway

Rubrics wise marks obtained

Understanding of Problem	Analysis of the Problem	Capability of writing program	Documentation	Total
04	04	10	04	20

Experiment No: 7

Date:

AIM: Implementation of Logistic Regression with Scikit-learn library in Python

Relevant CO:- CO4

Objective:
The objective of this lab practical is to implement logistic regression using Scikit-learn library in Python. Logistic regression is a popular classification algorithm used to model the relationship between input variables and categorical outcomes. In this lab, we will explore how to build a logistic regression model and use it for classification tasks.

Materials Used:
- Python 3.x
- Jupyter Notebook
- Scikit-learn library
- Pandas library
- NumPy library
- Matplotlib library

Dataset:
For this lab, we will use a dataset that contains information about customers and whether they churned or not from a telecommunications company. The dataset has the following columns:

- `CustomerID`: Unique identifier for each customer
- `Gender`: Gender of the customer (Male/Female)
- `Age`: Age of the customer
- `Income`: Income of the customer
- `Churn`: Binary variable indicating whether the customer churned (1) or not (0)

CustomerID,Gender,Age,Income,Churn
1,Male,32,50000,0
2,Female,28,35000,0
3,Male,45,80000,1
4,Male,38,60000,0
5,Female,20,20000,1
6,Female,55,75000,0
7,Male,42,90000,0
8,Female,29,40000,1

Procedure:

1. Introduction to Logistic Regression:
Logistic regression is a classification algorithm used to predict binary outcomes or probabilities. It models the relationship between input features and the probability of an event occurring. By applying the logistic function, it maps the linear regression output to a value between 0 and 1, allowing for classification based on predicted probabilities. Logistic regression is interpretable, handles categorical and continuous features, and

finds applications in various domains. It is a fundamental and effective approach for binary classification tasks.

2. Importing Required Libraries and Loading the Dataset:
   - Import the necessary libraries, including Scikit-learn, Pandas, NumPy, and Matplotlib.
   - Load the dataset into a Pandas DataFrame using the appropriate function or by reading from a file.

3. Exploratory Data Analysis:
   - Perform exploratory data analysis to understand the dataset.
   - Analyze the distribution of variables, detect any missing values, and handle them if necessary.
   - Visualize the relationships between variables using plots and charts.

4. Data Preprocessing:
   - Split the dataset into input features (independent variables) and the target variable (dependent variable).
   - Convert categorical variables into numerical representations using one-hot encoding or label encoding.
   - Split the dataset into training and testing sets for model evaluation.

5. Building the Logistic Regression Model:
   - Import the Logistic Regression class from Scikit-learn.
   - Instantiate the Logistic Regression model.
   - Fit the model to the training data using the `.fit()` method.

6. Model Evaluation and Prediction:
   - Evaluate the model's performance using appropriate evaluation metrics such as accuracy, precision, recall, and F1-score.
   - Make predictions on the testing data using the `.predict()` method.

7. Visualization of Results:
   - Visualize the model's performance using confusion matrix, ROC curve, or other suitable visualizations.
   - Plot the decision boundary to demonstrate the classification boundaries.

```
# Step 1: Importing Required Libraries
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score, confusion_matrix, RocCurveDisplay

# Step 2: Loading the Dataset
data = {
    'CustomerID': [1, 2, 3, 4, 5, 6, 7, 8],
    'Gender': ['Male', 'Female', 'Male', 'Male', 'Female', 'Female', 'Male', 'Female'],
    'Age': [32, 28, 45, 38, 20, 55, 42, 29],
    'Income': [50000, 35000, 80000, 60000, 20000, 75000, 90000, 40000],
    'Churn': [0, 0, 1, 0, 1, 0, 0, 1]
}
```

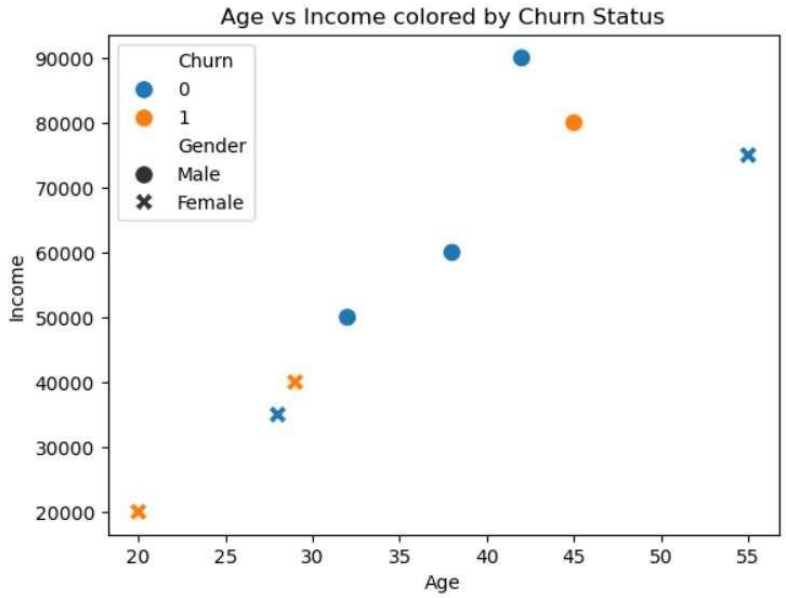
```
df = pd.DataFrame(data)

# Step 3: Check Class Distribution
print("Class Distribution:")
print(df['Churn'].value_counts())

# Visualizing the relationship between Age, Income, and Churn
sns.scatterplot(data=df, x='Age', y='Income', hue='Churn', style='Gender', s=100)
plt.title('Age vs Income colored by Churn Status')
plt.show()

# Step 4: Data Preprocessing
# Convert categorical variables to numerical using one-hot encoding
df = pd.get_dummies(df, columns=['Gender'], drop_first=True)

# Split the dataset into input features (X) and target variable (y)
X = df.drop(columns=['CustomerID', 'Churn'])
y = df['Churn']

# Step 5: Building the Logistic Regression Model
model = LogisticRegression(max_iter=1000, solver='liblinear') # Increased max_iter
model.fit(X, y)

# Step 6: Model Evaluation and Prediction
y_pred = model.predict(X)

# Evaluate the model's performance
accuracy = accuracy_score(y, y_pred)
print(f"Accuracy: {accuracy:.2f}")

# Print model coefficients for debugging
print("Model Coefficients:", model.coef_)
print("Intercept:", model.intercept_)

# Confusion Matrix
conf_matrix = confusion_matrix(y, y_pred)
sns.heatmap(conf_matrix, annot=True, fmt='d', cmap='Blues', xticklabels=['Not Churned', 'Churned'], yticklabels=['Not Churned', 'Churned'])
plt.title('Confusion Matrix')
plt.xlabel('Predicted')
plt.ylabel('Actual')
plt.show()

# Step 7: Visualization of ROC Curve
RocCurveDisplay.from_estimator(model, X, y)
plt.title('ROC Curve')
plt.show()

# Plotting Decision Boundary
plt.figure(figsize=(10, 6))

# Create a grid of points
```

```
x_min, x_max = X['Age'].min() - 5, X['Age'].max() + 5
y_min, y_max = X['Income'].min() - 5000, X['Income'].max() + 5000
xx, yy = np.meshgrid(np.arange(x_min, x_max, 1), np.arange(y_min, y_max, 500))

# Set a fixed value for the Gender feature (using '0' for Female)
gender_value = 0  # Change to 1 if you want to visualize for Male

# Prepare grid points for prediction
grid_points = np.c_[xx.ravel(), yy.ravel(), np.full(xx.ravel().shape, gender_value)]

# Predicting the class for each point in the grid
Z = model.predict(scaler.transform(grid_points)) # Scale the grid points
Z = Z.reshape(xx.shape)

# Plotting the decision boundary
plt.contourf(xx, yy, Z, alpha=0.3, cmap='coolwarm')

# Scatter plot of the data points
plt.scatter(X['Age'], X['Income'], c=y, cmap='coolwarm', edgecolor='k', s=100)

# Adding labels and title
plt.title('Logistic Regression Decision Boundary with Grid (Gender = Female)')
plt.xlabel('Age')
plt.ylabel('Income')
plt.xlim(x_min, x_max)
plt.ylim(y_min, y_max)
plt.colorbar()
plt.show()
```

Output:

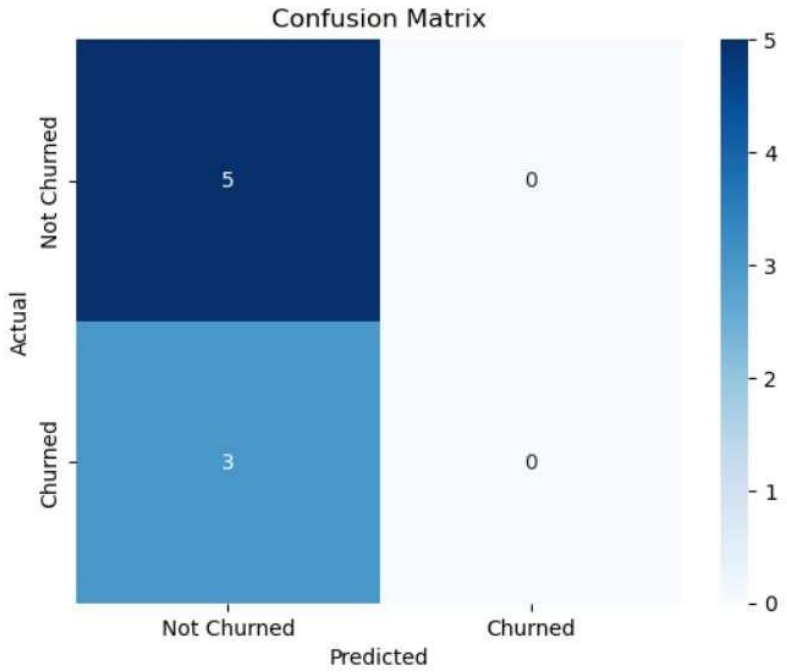
1). EDA

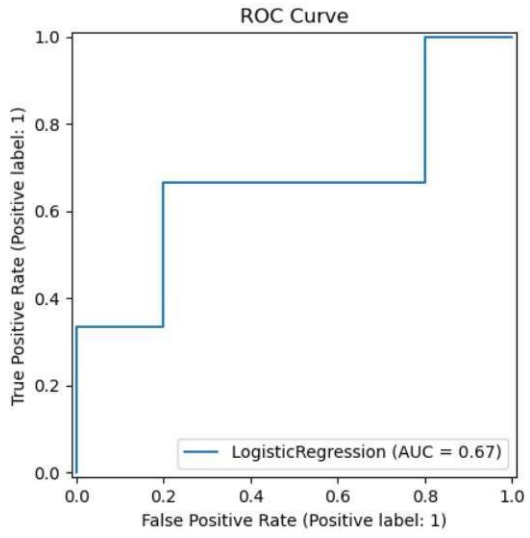
```
Class Distribution:
Churn
0    5
1    3
Name: count, dtype: int64
```

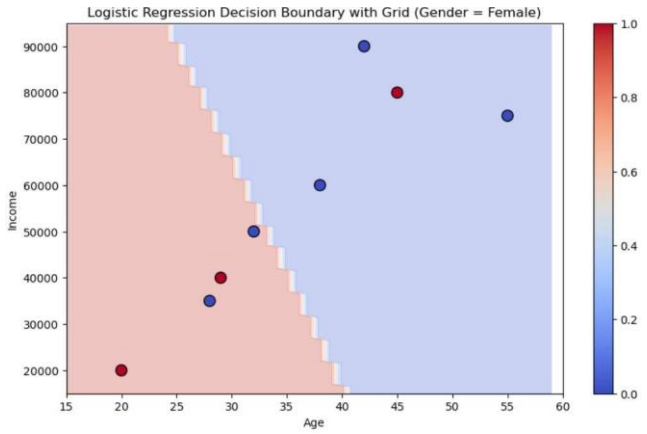


2).Evaluation

```
Accuracy: 0.62
Model Coefficients: [[-6.86835232e-09 -1.23258139e-05 -2.06482475e-10]]
Intercept: [-4.63749431e-11]
```







8. Conclusion:

Logistic regression is a powerful classification algorithm that models the relationship between input features and binary outcomes or probabilities. By utilizing the logistic function, it provides interpretable predictions and is applicable in various domains.Logistic regression is a valuable tool for binary classification tasks, offering simplicity, interpretability, and effectiveness in predicting outcomes based on input features.

Quiz:

1. Which scikit-learn function is used to create a logistic regression model object in Python?

a) sklearn.linear_model.LogisticRegression
b) sklearn.preprocessing.StandardScaler

c) sklearn.model_selection.train_test_split
d) sklearn.metrics.accuracy_score

2.In logistic regression, what does the sigmoid function do?

a) Maps the predicted values to binary classes
b) Calculates the log-odds of the target variable
c) Determines the optimal threshold for classification
d) Measures the goodness of fit of the logistic regression model

Suggested References:-

1. "Pattern Recognition and Machine Learning" by Christopher M. Bishop
2. Dinesh Kumar, Business Analytics, Wiley India Business alytics: The Science
3. V.K. Jain, Data Science & Analytics, Khanna Book Publishing, New Delhi of Dat
4. Data Science For Dummies by Lillian Pierson , Jake Porway

Rubrics wise marks obtained

Understanding of Problem	Analysis of the Problem	Capability of writing program	Documentation	Total
04	04	10	02	20

Experiment No: 8

Date:

AIM: Implementation of Decision Tree for Student Classification

Relevant CO :- CO4

Objective:
The objective of this lab practical is to implement a decision tree algorithm to classify students as either average or clever based on given student data. Decision trees are widely used in machine learning and data mining for classification and regression tasks. In this lab, we will explore how to build a decision tree model and use it to classify students based on their attributes.

Materials Used:
- Python 3.x
- Jupyter Notebook
- Scikit-learn library
- Pandas library
- NumPy library
- Matplotlib library

Dataset:
For this lab, we will use a dataset that contains information about students and their performance. The dataset has the following columns:

- `Age`: Age of the student
- `StudyHours`: Number of hours the student studies per day
- `PreviousGrade`: Grade achieved in the previous exam
- `Result`: Classification label indicating whether the student is average (0) or clever (1)

Procedure:

1. Introduction to Decision Trees:
   - Decision trees are widely used in machine learning for classification tasks. Theymake decisions based on splitting criteria and feature importance. In this lab, we will implement a decision tree algorithm to classify students as average or clever based on their attributes, such as age, study hours, and previous grades.
2. Importing Required Libraries and Loading the Dataset:
   - Import the necessary libraries, including Scikit-learn, Pandas, NumPy, andMatplotlib.
   - Load the dataset into a Pandas DataFrame using the appropriate function or by reading from a file.
3. Exploratory Data Analysis:
   - Perform exploratory data analysis to understand the dataset.
   - Analyze the distribution of variables, detect any missing values, and handle them if necessary.
   - Visualize the relationships between variables using plots and charts.

4. Data Preprocessing:
   - Split the dataset into input features (independent variables) and the target variable (dependent variable).

- Convert categorical variables into numerical representations using one-hot encoding or label encoding.
- Split the dataset into training and testing sets for model evaluation.

5. Building the Decision Tree Model:
- Import the DecisionTreeClassifier class from Scikit-learn.
- Instantiate the DecisionTreeClassifier model with the desired parameters.
- Fit the model to the training data using the `.fit()` method.

6. Model Evaluation and Prediction:
- Evaluate the model's performance using appropriate evaluation metrics such as accuracy, precision, recall, and F1-score.
- Make predictions on the testing data using the `.predict()` method.

7. Visualization of the Decision Tree:
- Visualize the decision tree using tree plotting techniques available in Scikit-learn or other visualization libraries.
- Interpret the decision tree structure and analyze the important features.

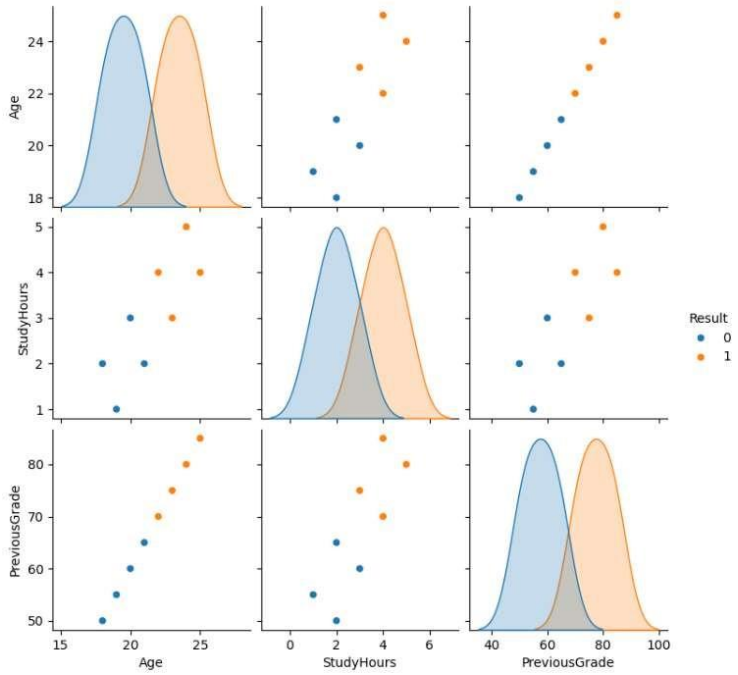
Interpretation/Program/code:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import accuracy_score, confusion_matrix
from sklearn import tree
import seaborn as sns

# Sample Dataset
data = {
    'Age': [18, 20, 22, 24, 19, 21, 23, 25],
    'StudyHours': [2, 3, 4, 5, 1, 2, 3, 4],
    'PreviousGrade': [50, 60, 70, 80, 55, 65, 75, 85],
    'Result': [0, 0, 1, 1, 0, 0, 1, 1]  # 0 = Average, 1 = Clever
}

df = pd.DataFrame(data)

# Check for missing values
print(df.isnull().sum())

# Visualize data
sns.pairplot(df, hue='Result')
plt.show()

X = df[['Age', 'StudyHours', 'PreviousGrade']]
y = df['Result']

# Split data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42)
```

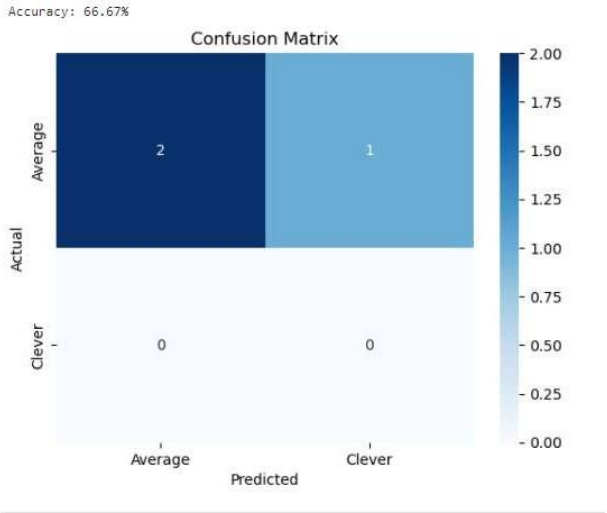
```
# Initialize the Decision Tree Classifier
clf = DecisionTreeClassifier()

# Train the model
clf.fit(X_train, y_train)

# Predicting on the test data
y_pred = clf.predict(X_test)

# Evaluating model accuracy
accuracy = accuracy_score(y_test, y_pred)
print(f"Accuracy: {accuracy * 100:.2f}%")

# Confusion Matrix
conf_matrix = confusion_matrix(y_test, y_pred)
sns.heatmap(conf_matrix, annot=True, fmt='d', cmap='Blues', xticklabels=['Average', 'Clever'],
yticklabels=['Average', 'Clever'])
plt.title('Confusion Matrix')
plt.xlabel('Predicted')
plt.ylabel('Actual')
plt.show()

# Visualizing the Decision Tree
plt.figure(figsize=(12,8))
tree.plot_tree(clf, filled=True, feature_names=['Age', 'StudyHours', 'PreviousGrade'],
class_names=['Average', 'Clever'])
plt.show()
```

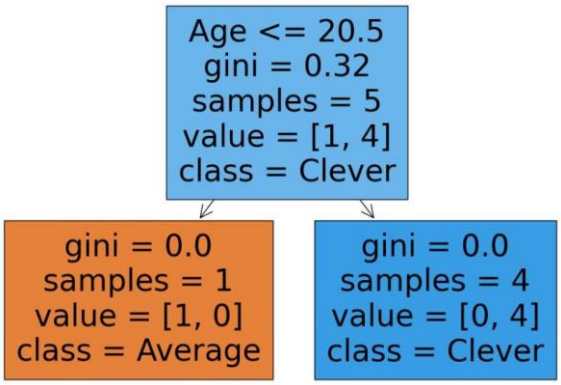
Output:

1). EDA



2). Evaluation





Conclusion:

- The implementation of the decision tree algorithm proved effective in classifying students as average or clever based on their attributes. Decision trees provide interpretable results and can be used in various domains for classification tasks. The decision tree model offers insights into the important features contributing to the classification. This lab demonstrates the practical application of decision trees for studentclassification.

Quiz:

1. In decision tree classification, what is the main objective of the splitting criterion?

a) To maximize the number of features used for classification
b) To minimize the number of nodes in the decision tree
c) To maximize the accuracy of classification
d) To minimize the entropy or Gini impurity of the resulting subsets

2. What is the purpose of pruning in decision tree algorithms?

a) To remove irrelevant features from the dataset
b) To prevent overfitting and improve generalization of the decision tree
c) To improve the interpretability of the decision tree
d) To reduce the time complexity of the decision tree algorithm

Suggested References:-

1. "Pattern Recognition and Machine Learning" by Christopher M. Bishop
2. Dinesh Kumar, Business Analytics, Wiley India Business alytics: The Science
3. V.K. Jain, Data Science & Analytics, Khanna Book Publishing, New Delhi of Dat
4. Data Science For Dummies by Lillian Pierson , Jake Porway

Rubrics wise marks obtained

Understanding of Problem	Analysis of the Problem	Capability of writing program	Documentation	Total
02	02	05	01	10