Data Science Lab

Faculty Name: Dr. Namita Gupta

Student Name: Prathya Thakur

Roll No.: 03614802721

Semester: 7th

Group: AIML 1 B

Maharaja Agrasen Institute of Technology, PSP Area, Sector - 22, New Delhi – 110085

MAHARAJA AGRASEN INSTITUTE OF TECHNOLOGY

COMPUTER SCIENCE & ENGINEERING DEPARTMENT

VISION

"To be centre of excellence in education, research and technology transfer in the field of computer engineering and promote entrepreneurship and ethical values."

MISSION

"To foster an open, multidisciplinary and highly collaborative research environment to produce world-class engineers capable of providing innovative solutions to real life problems and fulfill societal needs."

PRACTICAL RECORD

Name of the student : Prathya Thakur

University Roll No. : 03614802721

Branch : CSE-1

Section/ Group : 7 AIML 1 B

Exp. No.	Experiment Name	Date of performance	Date of checking	R1 (3)	R2 (3)	R3 (3)	R4 (3)	R5 (3)	Total Marks (15)	Signature
1.	Describing data, viewing, and manipulating data.									
2.	To plot the probability distribution curve									
3.	To perform Chi-square test on various datasets									
4.	To use Python as a programming tool for the analysis of data structures									
5.	To perform various operations such as data storage, analysis, and visualization									
6.	To perform descriptive statistics analysis and data visualization									
7.	To perform Principal Component Analysis on datasets									
8.	To perform linear regression on datasets									
9.	To perform Data Aggregation and GroupWise Operations									

Experiment – 1

Aim: Describing data, viewing, and manipulating data

Theory: Data viewing and manipulation are critical processes in data science, enabling analysts and data scientists to explore, clean, and transform datasets for further analysis.

Data Viewing

The process of data viewing involves examining raw data to understand its structure, content, and potential issues. Tools like pandas in Python or data frames in R allow users to load and view datasets, enabling quick inspections of rows, columns, and data types. Typical methods include:

Head and Tail Views: Displaying the first or last few rows of a dataset to get a sense of the data without loading it entirely.

Summarization: Functions like describe() or info() provide summary statistics (mean, median, standard deviation) and metadata (data types, missing values) that give an overview of the dataset's characteristics.

These methods help in detecting anomalies such as missing data, outliers, and inconsistencies, guiding the need for further manipulation.

Data Manipulation

Data manipulation refers to the process of cleaning, reshaping, and transforming data to make it suitable for analysis. The main steps include:

Handling Missing Data: Missing values can be dealt with by filling them using techniques like mean or median imputation or removing them if they don't add significant value.

Filtering and Subsetting: Data filtering involves selecting rows and columns based on conditions, such as removing irrelevant data or focusing on specific variables.

Data Transformation: Transformations involve converting data types, scaling numerical values, or encoding categorical variables. Operations such as merging datasets, pivoting tables, or adding new calculated columns are also common.

Efficient data manipulation not only cleans the data but also reshapes it to align with the requirements of the analytical model or algorithm being applied. This process ensures that the dataset is in a structured form suitable for further statistical or machine learning analysis.

Source Code:

```
import pandas as pd

# Load a dataset
data = pd.read_csv('Customer.csv')

# View the first few rows of the dataset
print(data.head())

# Get the summary statistics
print(data.describe())

# Manipulating data: Adding a new column
data['new_column'] = data['purchase_amount'] * 2

# View modified dataset
print(data.head())
```

Output:

```
   user_id  age  annual_income  purchase_amount  loyalty_score region  \
0        1   25          45000              200            4.5  North
1        2   34          55000              350            7.0  South
2        3   45          65000              500            8.0   West
3        4   22          30000              150            3.0   East
4        5   29          47000              220            4.8  North

   purchase_frequency
0                  12
1                  18
2                  22
3                  10
4                  13
          user_id         age  annual_income  purchase_amount  loyalty_score  \
count  238.000000  238.000000     238.000000       238.000000     238.000000
mean   119.500000   38.676471   57407.563025       425.630252       6.794118
std     68.848868    9.351118   11403.875717       140.052062       1.899047
min      1.000000   22.000000   30000.000000       150.000000       3.000000
25%     60.250000   31.000000   50000.000000       320.000000       5.500000
50%    119.500000   39.000000   59000.000000       440.000000       7.000000
75%    178.750000   46.750000   66750.000000       527.500000       8.275000
max    238.000000   55.000000   75000.000000       640.000000       9.500000

       purchase_frequency
count          238.000000
mean            19.798319
std              4.562884
min             10.000000
25%             17.000000
50%             20.000000
75%             23.000000
max             28.000000

   user_id  age  annual_income  purchase_amount  loyalty_score region  \
0        1   25          45000              200            4.5  North
1        2   34          55000              350            7.0  South
2        3   45          65000              500            8.0   West
3        4   22          30000              150            3.0   East
4        5   29          47000              220            4.8  North

   purchase_frequency  new_column
0                  12         400
1                  18         700
2                  22        1000
3                  10         300
4                  13         440
```

Viva-Voce:

Q1) What is the purpose of data viewing in data analysis, and what tools are commonly used?

A1) Data viewing involves inspecting the raw data to understand its structure, quality, and content. Common tools include data frames in libraries like pandas (Python) and tibbles (R), which allow users to preview rows, columns, and summary statistics to identify potential issues and guide further analysis.

Q2) How can you view the first and last few rows of a dataset using pandas in Python?

A2) In pandas, you can use the head() method to view the first few rows and the tail() method to view the last few rows of a DataFrame. For example, df.head() displays the first five rows, and df.tail() shows the last five rows by default.

Q3) What are some common data manipulation tasks, and how are they performed in Python?

A3) Common data manipulation tasks include filtering, sorting, merging, and aggregating data. In Python, pandas provides functions such as filter(), sort_values(), merge(), and groupby() to perform these operations. For example, df.sort_values(by='column_name') sorts data by a specific column.

Q4) How can you handle missing values in a dataset using pandas?

A4) Missing values can be handled using methods like fillna() to replace them with a specific value or method (e.g., mean, median), or dropna() to remove rows or columns containing missing values. For example, df.fillna(value=0) replaces all missing values with 0.

Q5) What is data normalization, and why is it important in data manipulation?

A5) Data normalization is the process of scaling data to a standard range or distribution, often to ensure that features contribute equally to analysis or modeling. It is important because it can improve the performance and convergence of machine learning algorithms and make comparisons between features more meaningful. Common normalization techniques include min-max scaling and z-score standardization.

Experiment – 2

Aim: To plot the probability distribution curve.

Theory: Probability curves in data science represent the distribution of possible outcomes for a random variable, providing insights into data behaviour and uncertainty. Common curves include the normal distribution (bell curve), which is symmetric around the mean, and skewed distributions where one tail is longer than the other. These curves help in modelling real-world phenomena, identifying patterns, and making predictions. For example, the normal curve is widely used due to the central limit theorem, which states that the sum of independent variables tends toward a normal distribution. Understanding probability curves is crucial for tasks like statistical inference and machine learning.

Source Code:

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

# Generate random data
data = np.random.normal(0, 1, 1000)

# Plot the probability distribution curve
sns.histplot(data, kde=True)
plt.title('Probability Distribution Curve')
plt.show()
```

Output:
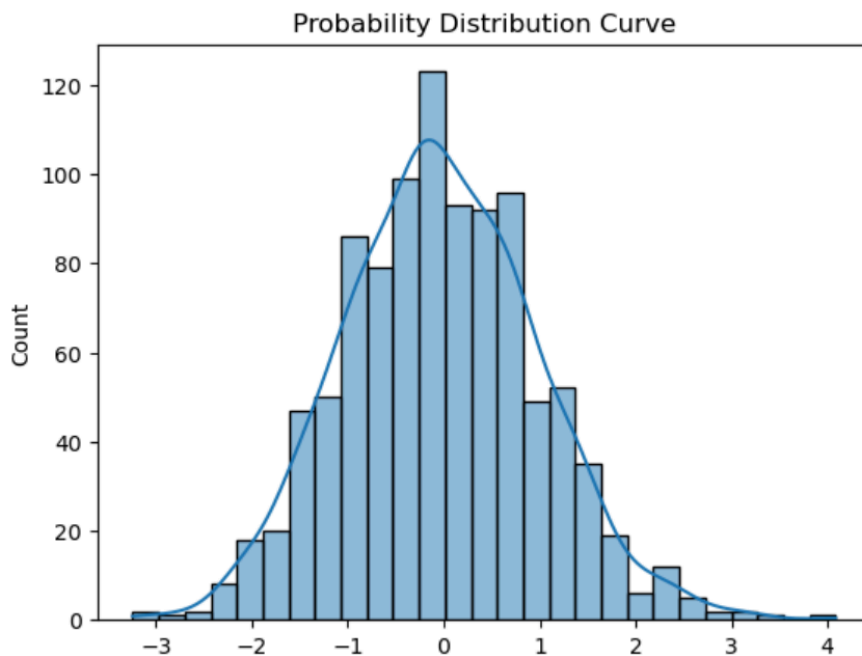
Viva-Voce:

Q1) What is a probability curve, and why is it important in data science?

A1) A probability curve represents the likelihood of different outcomes for a random variable. It's important in data science because it helps in modeling uncertainty, understanding distributions, and making predictions based on the data.

Q2) Can you explain the normal distribution and its significance?

A2) The normal distribution, also known as the bell curve, is symmetric and centered around the mean. It's significant because many real-world phenomena follow a normal distribution, and it forms the basis of statistical inference due to the central limit theorem.

Q3) What are skewed distributions, and how do they differ from normal distributions?

A3) Skewed distributions have one tail longer than the other, indicating that data is not symmetrically distributed. Unlike normal distributions, where the mean, median, and mode coincide, in skewed distributions, these measures of central tendency differ.

Q4) How does the shape of a probability curve impact data analysis?

A4) The shape of a probability curve affects the choice of statistical methods and models. For example, normally distributed data allows for parametric tests, while skewed or non-normal distributions may require non-parametric tests or data transformation.

Q5) What is the role of the central limit theorem in probability curves?

A5) The central limit theorem states that the sum or average of a large number of independent random variables tends to follow a normal distribution, regardless of the original distribution. This theorem underpins many statistical techniques and allows the use of normal distribution-based models in diverse situations.

Experiment – 3

Aim: To perform Chi-square test on various datasets.

Theory: Chi-square tests in data science are used to determine the relationship between categorical variables and assess the goodness of fit or independence within a dataset. By comparing observed data with expected outcomes, the test evaluates whether deviations are due to chance or a significant association. In a goodness-of-fit test, chi-square determines how well an observed distribution matches a theoretical one, while in independence tests, it checks if two categorical variables are related. The chi-square statistic is calculated by summing the squared differences between observed and expected values, divided by the expected values, providing insight into data dependencies and patterns.

Source Code:

```
import pandas as pd
from scipy.stats import chi2_contingency

# Create a contingency table
data = {'Observed': [50, 30, 20], 'Expected': [40, 40, 20]}
df = pd.DataFrame(data)

# Perform chi-square test
chi2, p, dof, expected = chi2_contingency([df['Observed'], df['Expected']])
print(f"Chi2 Statistic: {chi2}, p-value: {p}")
```

Output:

```
Chi2 Statistic: 2.5396825396825395, p-value: 0.28087620176428163
```

Viva-Voce:

Q1) What is a chi-square test, and when is it used in data science?

A1) A chi-square test assesses the association between categorical variables or the goodness of fit between observed and expected frequencies. It's used to determine if there is a significant difference between expected and observed data, helping in hypothesis testing and evaluating model performance.

Q2) Explain the difference between the chi-square test of independence and the chi-square goodness-of-fit test.

A2) The chi-square test of independence examines whether two categorical variables are related or independent, using a contingency table. The chi-square goodness-of-fit test compares observed data against a theoretical distribution to see if the data follows the expected distribution.

Q3) How is the chi-square statistic calculated?

A3) The chi-square statistic is calculated by summing the squared differences between observed and expected frequencies, divided by the expected frequencies:

$$x_c^2 = \frac{\Sigma \left(O_i - E_i\right)^2}{E_i}$$

Where,

c = Degrees of freedom

O = Observed Value

E = Expected Value

Q4) What are the assumptions of the chi-square test?

A4) The main assumptions are that the data should be categorical, the observations should be independent, and the expected frequency in each cell of the contingency table should be at least 5 for the test to be valid.

Q5) How do you interpret the results of a chi-square test?

A5) The results are interpreted by comparing the chi-square statistic to a critical value from the chi-square distribution table or by looking at the p-value. A significant p-value (typically <0.05) indicates that there is a significant difference between observed and expected frequencies, suggesting a relationship or discrepancy in the data.

Experiment – 4

Aim: To use Python as a programming tool for the analysis of data structures.

Theory: Python is a powerful programming tool for analyzing data structures due to its simplicity, flexibility, and vast ecosystem of libraries. It supports a variety of built-in data structures like lists, dictionaries, tuples, and sets, allowing for efficient data manipulation. Libraries such as NumPy and pandas enhance Python's capabilities by providing specialized data structures like arrays and DataFrames for handling large datasets. Python's rich collection of algorithms and functions helps in sorting, searching, and transforming data, making it a preferred choice for tasks like data analysis, machine learning, and algorithm design, all while ensuring readable and maintainable code.

Source Code:

```python
import numpy as np

# Example of array manipulation using NumPy
array = np.array([[1, 2, 3], [4, 5, 6], [7, 8, 9]])

# View array
print("Array:\n", array)

# Sum of all elements
print("Sum of elements:", np.sum(array))

# Transpose of the array
print("Transpose:\n", np.transpose(array))
```

Output:

```
Array:
 [[1 2 3]
 [4 5 6]
 [7 8 9]]
Sum of elements: 45
Transpose:
 [[1 4 7]
 [2 5 8]
 [3 6 9]]
```

<u>Viva-Voce</u>:

Q1) Why is Python a popular choice for data structure analysis in data science?

A1) Python is popular due to its readability, extensive libraries (such as NumPy, pandas, and SciPy), and ease of integration with other tools. Its versatile data structures (lists, dictionaries, sets, tuples) and powerful data manipulation capabilities make it well-suited for analyzing and managing complex datasets.

Q2) What role do libraries like NumPy and pandas play in data structure analysis?

A2) NumPy provides support for numerical operations with its array object, enabling efficient handling of large datasets and mathematical computations. Pandas offers data structures like DataFrames and Series, which simplify data manipulation, analysis, and cleaning, making it easier to work with tabular data.

Q3) How do Python's built-in data structures compare to those provided by libraries like pandas?

A3) Python's built-in data structures (lists, dictionaries, tuples, sets) are versatile but may lack efficiency for large-scale data operations. Pandas provides specialized data structures like DataFrames and Series designed for handling large datasets with functionalities for data alignment, indexing, and complex operations that are more efficient and user-friendly for data analysis.

Q4) What are the advantages of using Python for handling and analyzing large datasets?

A4) Python offers advantages such as its powerful libraries (e.g., NumPy for numerical operations, pandas for data manipulation), scalability through integration with big data tools, and a wide range of visualization libraries (e.g., Matplotlib, Seaborn). These tools facilitate efficient data handling, analysis, and visualization, making Python a robust choice for large datasets.

Q5) Can you describe a common workflow for data analysis using Python?

A5) A common workflow includes:

- o   Data Collection: Importing data from various sources (CSV, databases, APIs) using libraries like pandas.

- o   Data Cleaning: Handling missing values, outliers, and inconsistencies using pandas.

- o   Data Transformation: Reshaping data, merging datasets, and feature engineering.

- o   Analysis: Performing statistical analysis and model building using libraries like NumPy, pandas, and SciPy.

- o   Visualization: Creating plots and charts with Matplotlib or Seaborn to interpret and present the findings.

Experiment – 5

Aim: To perform various operations such as data storage, analysis, and visualization

Theory: Data storage, analysis, and visualization are fundamental aspects of data science, each playing a crucial role in deriving insights from data.

Data Storage: Efficient data storage solutions, such as databases (SQL, NoSQL), data lakes, and cloud storage systems, ensure data is well-organized, accessible, and scalable for future analysis. Tools like MySQL, MongoDB, and AWS S3 are commonly used.

Data Analysis: Python, R, and SQL offer powerful methods to explore, clean, and manipulate data. Libraries like pandas, NumPy, and SciPy allow for complex analysis, statistical modeling, and handling large datasets.

Data Visualization: Visualization tools like Matplotlib, Seaborn, and Tableau are essential for presenting data trends, distributions, and insights in graphical formats. Visualizations simplify complex data, enabling better decision-making.

Source Code:

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

# Load dataset
data = pd.read_csv('organizations.csv')

# Data storage: Save the dataset to a new file
data.to_csv('new_dataset.csv', index=False)

# Data analysis: Describe the dataset
print(data.describe())

# Data visualization: Plot a histogram
data['Number of employees'].hist()
plt.title('Histogram of Number of employees')
plt.show()
```

Output:

	Index	Founded	Number of employees
count	100.000000	100.000000	100.000000
mean	50.500000	1995.410000	4964.860000
std	29.011492	15.744228	2850.859799
min	1.000000	1970.000000	236.000000
25%	25.750000	1983.500000	2741.250000
50%	50.500000	1995.000000	4941.500000
75%	75.250000	2010.250000	7558.000000
max	100.000000	2021.000000	9995.000000

Histogram of Number of employees
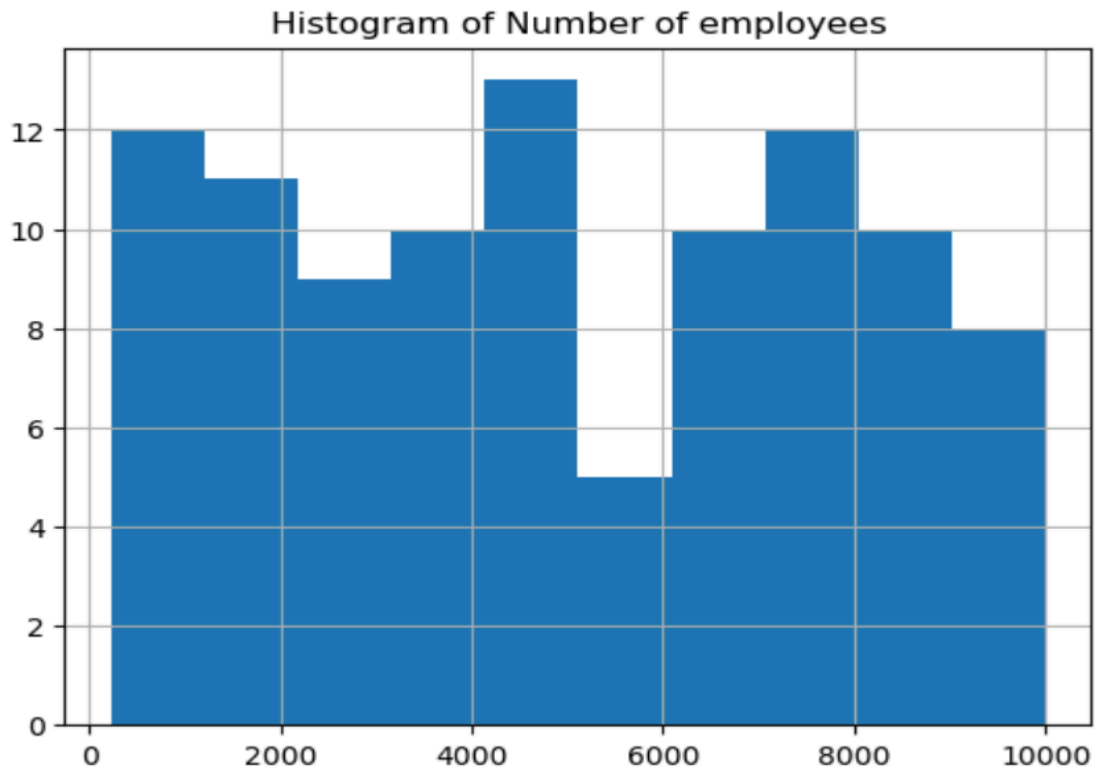
Viva-Voce:

Q1) What are the common methods for data storage in data science?

A1) Data can be stored in various formats, such as relational databases (SQL), NoSQL databases (e.g., MongoDB), flat files (CSV, JSON), or cloud storage (AWS S3, Google Cloud). These methods depend on factors like the type of data, the need for scalability, and query performance requirements.

Q2) How is data analysis performed in data science, and which tools are commonly used?

A2) Data analysis involves exploring, cleaning, and modeling data to extract insights. Common tools include Python libraries like pandas and NumPy for handling data, R for statistical analysis, and SQL for querying structured data. These tools allow for operations like filtering, aggregating, and visualizing data.

Q3) What is the role of cloud storage in data science, and how does it benefit data analysis?

A3) Cloud storage (e.g., AWS S3, Google Cloud Storage) enables scalable, accessible, and cost-effective data management. It allows data scientists to store vast amounts of data without worrying about physical infrastructure, facilitating collaboration and integration with cloud-based analysis tools for large-scale processing.

Q4) What are the key differences between descriptive and inferential data analysis?

A4) Descriptive analysis summarizes data with measures like mean, median, and standard deviation, providing an overview of the dataset. Inferential analysis goes beyond the data at hand to make predictions or inferences about a larger population based on a sample, often using hypothesis testing or predictive modeling.

Q5) How do tools like Matplotlib and Seaborn assist in data visualization?A5) Matplotlib and Seaborn are Python libraries used to create static, animated, or interactive plots. Matplotlib provides detailed control over plot elements, while Seaborn simplifies the creation of complex visualizations .

Experiment – 6

Aim: To perform descriptive statistics analysis and data visualization.

Theory: Descriptive statistical analysis and data visualization are key techniques in summarizing and interpreting data in data science.

Descriptive Statistical Analysis: This method involves summarizing data using measures like mean, median, mode, standard deviation, and variance. It helps in understanding the central tendency, spread, and overall distribution of data, offering insights without making predictions. It's the foundation for understanding the dataset's basic characteristics.

Data Visualization: Complementing descriptive statistics, data visualization tools like Matplotlib, Seaborn, and Power BI create graphical representations (histograms, box plots, bar charts) of these summaries, making patterns, trends, and outliers easily understandable, aiding in effective communication of data insights.

Source Code:

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

# Load dataset
data = pd.read_csv('organizations.csv')

# Descriptive statistics
print(data.describe())

# Data visualization: Box plot of a column
data.boxplot(column='Number of employees')
plt.title('Box Plot of Number of employees')
plt.show()
```

Output:

	Index	Founded	Number of employees
count	100.000000	100.000000	100.000000
mean	50.500000	1995.410000	4964.860000
std	29.011492	15.744228	2850.859799
min	1.000000	1970.000000	236.000000
25%	25.750000	1983.500000	2741.250000
50%	50.500000	1995.000000	4941.500000
75%	75.250000	2010.250000	7558.000000
max	100.000000	2021.000000	9995.000000

Box Plot of Number of employees
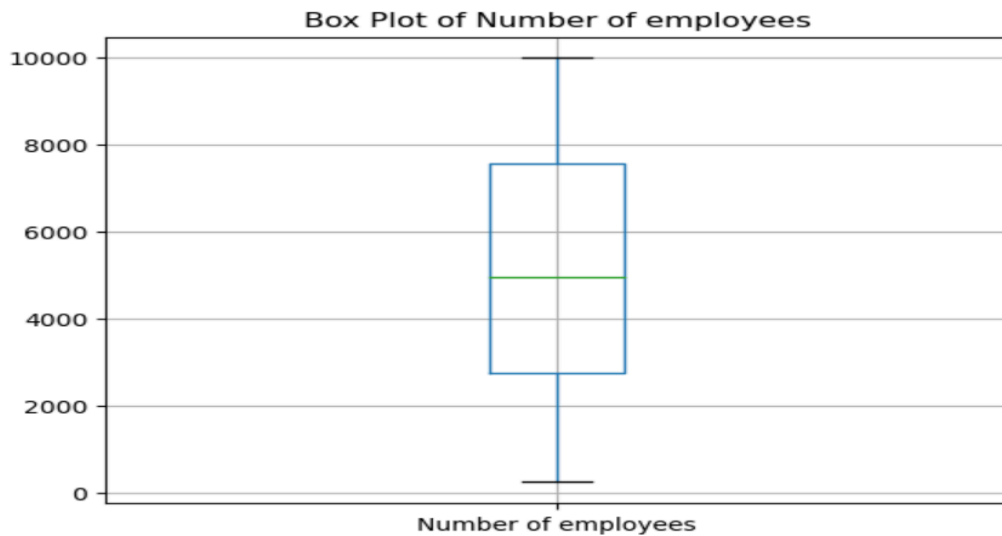
Viva-Voce:

Q1) What is descriptive statistical analysis, and what are its main components?

A1) Descriptive statistical analysis involves summarizing and describing the main features of a dataset using measures such as mean, median, mode, standard deviation, and range. It provides insights into the central tendency, dispersion, and overall distribution of data without making predictions or generalizations.

Q2) How do measures of central tendency differ from measures of dispersion in descriptive statistics?

A2) Measures of central tendency (mean, median, mode) describe the center or typical value of a dataset. Measures of dispersion (standard deviation, variance, range) describe the spread or variability around the central value. Together, they provide a comprehensive summary of the dataset's characteristics.

Q3) Why is data visualization important in the context of descriptive statistics?

A3) Data visualization is crucial as it helps to communicate the insights gained from descriptive statistics in a clear and intuitive manner. Visual tools like histograms, box plots, and scatter plots make it easier to identify patterns, trends, and anomalies in the data, facilitating better understanding and decision-making.

Q4) Can you explain the role of histograms and box plots in visualizing descriptive statistics?

A4) Histograms display the distribution of numerical data by showing the frequency of data points within specified ranges or bins, revealing the shape and spread of the data. Box plots, on the other hand, provide a visual summary of data distribution through quartiles, highlighting the median, spread, and potential outliers.

Q5) How can descriptive statistical measures be used to identify data quality issues?

A5) Descriptive statistical measures can reveal data quality issues by highlighting inconsistencies, such as unusual values or outliers. For example, an unusually high standard deviation might indicate data entry errors, while skewed distributions can signal data problems or imbalances that need addressing before further analysis.

Experiment – 7

Aim: To perform Principal Component Analysis on datasets.

Theory: Component analysis in datasets refers to techniques that reduce the dimensionality of data while retaining essential patterns and variance.

Principal Component Analysis (PCA): PCA is a widely used method in component analysis, transforming a dataset into a set of linearly uncorrelated components. It reduces the number of variables by identifying the directions (principal components) that capture the most variance in the data, making it easier to analyze and visualize complex, high-dimensional datasets.

Visualization: PCA and other component analysis techniques can be visualized through 2D or 3D plots, helping data scientists interpret data structure, identify patterns, and detect relationships between variables efficiently.

Source Code:

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

# importing or loading the dataset
dataset = pd.read_csv('wine.csv')

# distributing the dataset into two components X and Y
X = dataset.iloc[:, 0:13].values
y = dataset.iloc[:, 13].values

# Splitting the X and Y into the  Training set and Testing set
from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=0)

# performing preprocessing part
from sklearn.preprocessing import StandardScaler
sc = StandardScaler()

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


# Applying PCA function on training and testing set of X component
from sklearn.decomposition import PCA

pca = PCA(n_components=2)

X_train = pca.fit_transform(X_train)
X_test = pca.transform(X_test)

explained_variance = pca.explained_variance_ratio_

# Fitting Logistic Regression To the training set
from sklearn.linear_model import LogisticRegression

classifier = LogisticRegression(random_state=0)
classifier.fit(X_train, y_train)

# Predicting the test set result using predict function under LogisticRegression
y_pred = classifier.predict(X_test)

# making confusion matrix between test set of Y and predicted value.
from sklearn.metrics import confusion_matrix
```

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

# Predicting the training set result through scatter plot
from matplotlib.colors import ListedColormap

X_set, y_set = X_train, y_train
X1, X2 = np.meshgrid(np.arange(start=X_set[:, 0].min() - 1,
                     stop=X_set[:, 0].max() + 1, step=0.01),
               np.arange(start=X_set[:, 1].min() - 1,
                     stop=X_set[:, 1].max() + 1, step=0.01))

plt.contourf(X1, X2, classifier.predict(np.array([X1.ravel(),X2.ravel()]).T).reshape(X1.shape), alpha=0.75,
cmap=ListedColormap(('yellow', 'white', 'aquamarine')))

plt.xlim(X1.min(), X1.max())
plt.ylim(X2.min(), X2.max())

for i, j in enumerate(np.unique(y_set)):
    plt.scatter(X_set[y_set == j, 0], X_set[y_set == j, 1],
            color=ListedColormap(('red', 'green', 'blue'))(i), label=j)

plt.title('Logistic Regression (Training set)')
plt.xlabel('PC1')  # for Xlabel
plt.ylabel('PC2')  # for Ylabel
plt.legend()  # to show legend

# show scatter plot
plt.show()

# Visualising the Test set results through scatter plot
X_set, y_set = X_test, y_test

X1, X2 = np.meshgrid(np.arange(start=X_set[:, 0].min() - 1, stop=X_set[:, 0].max() + 1, step=0.01),
            np.arange(start=X_set[:, 1].min() - 1, stop=X_set[:, 1].max() + 1, step=0.01))

plt.contourf(X1, X2, classifier.predict(np.array([X1.ravel(), X2.ravel()]).T).reshape(X1.shape), alpha=0.75,
         cmap=ListedColormap(('yellow', 'white', 'aquamarine')))

plt.xlim(X1.min(), X1.max())
plt.ylim(X2.min(), X2.max())

for i, j in enumerate(np.unique(y_set)):
    plt.scatter(X_set[y_set == j, 0], X_set[y_set == j, 1],
            color=ListedColormap(('red', 'green', 'blue'))(i), label=j)

# title for scatter plot
plt.title('Logistic Regression (Test set)')
plt.xlabel('PC1')  # for Xlabel
plt.ylabel('PC2')  # for Ylabel
plt.legend()

# show scatter plot
plt.show()
```
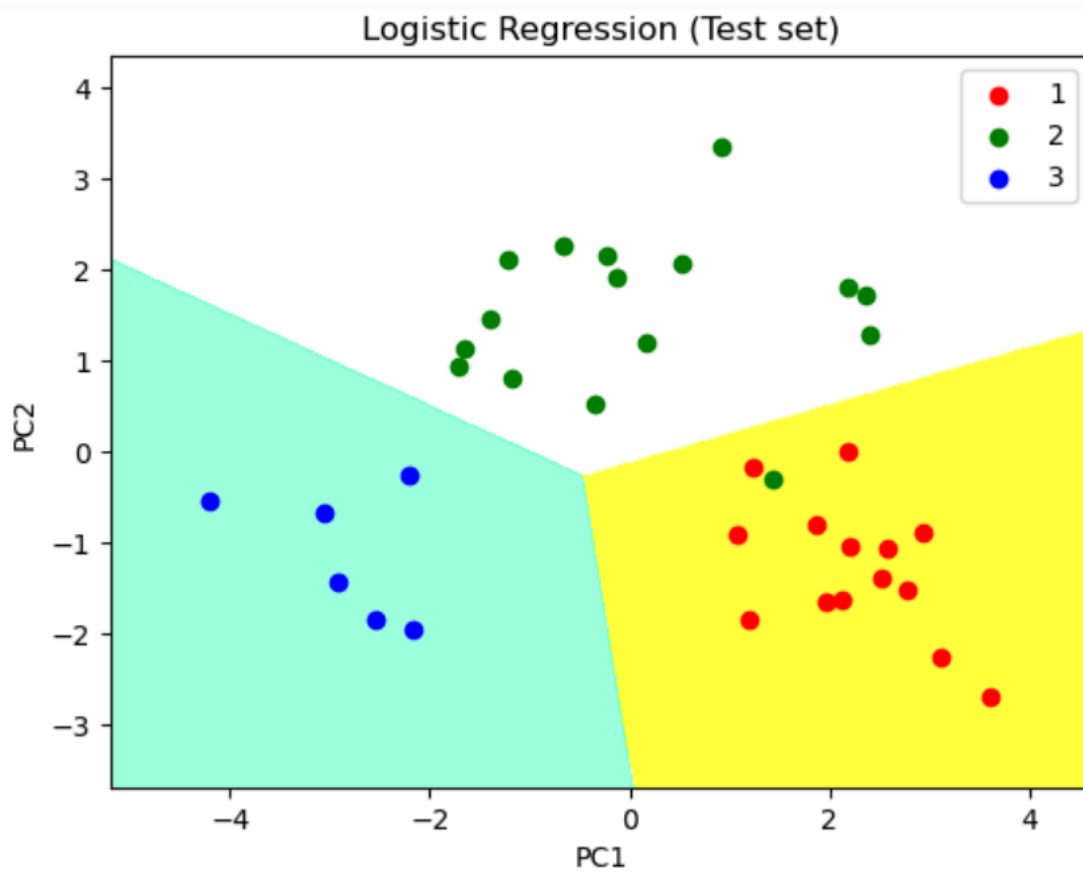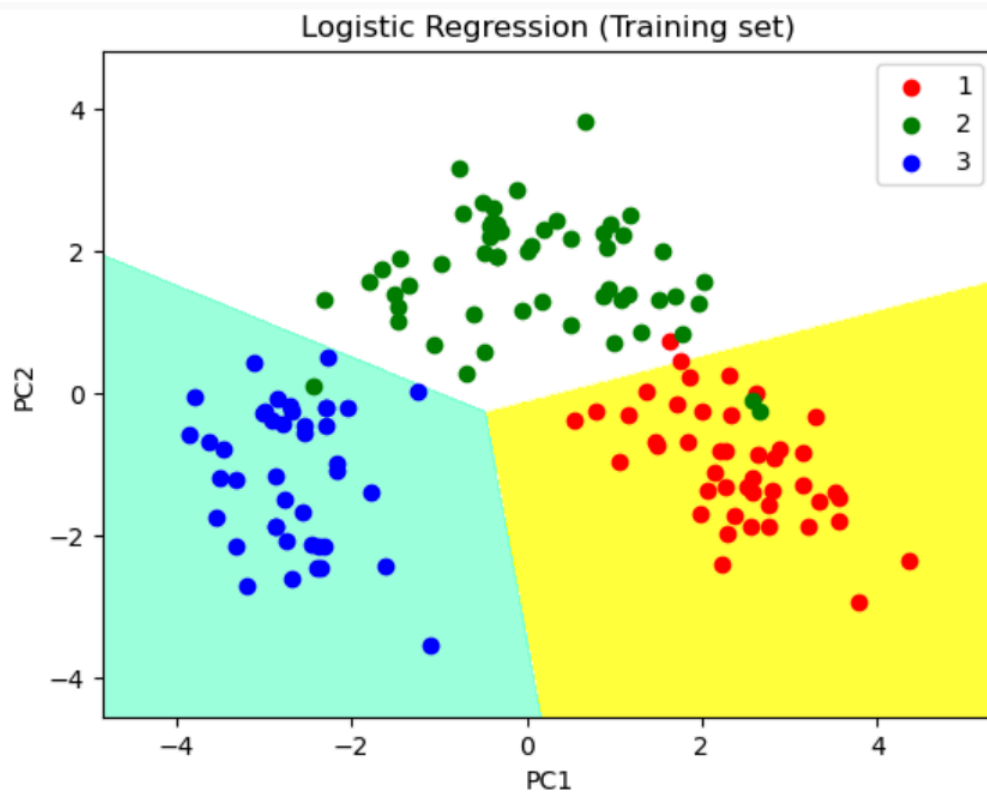
Logistic Regression (Training set)

Logistic Regression (Test set)

<u>Viva-Voce</u>:

Q1) What is component analysis, and what are its main objectives?

A1) Component analysis refers to techniques used to reduce the dimensionality of datasets while retaining important information. Its main objectives are to simplify data, identify underlying patterns, and make complex datasets easier to visualize and analyze. Common techniques include Principal Component Analysis (PCA) and Factor Analysis.

Q2) How does Principal Component Analysis (PCA) work, and what is its purpose?

A2) PCA works by transforming the original dataset into a new set of orthogonal (uncorrelated) components called principal components. These components are ordered by the amount of variance they explain in the data. The purpose of PCA is to reduce the dimensionality of the data while preserving as much variance as possible, making it easier to analyze and visualize.

Q3) What is the significance of the explained variance in PCA?

A3) Explained variance indicates the proportion of the total variance in the dataset that is captured by each principal component. It helps in understanding how much information each component retains and guides the selection of a subset of components that can effectively represent the original data.

Q4) How can you interpret the results of a PCA analysis?

A4) The results of PCA can be interpreted by examining the principal components' loadings, which show the contribution of each original variable to the components. The explained variance plot (scree plot) helps determine the number of components to retain. Visualizing the data projected onto the principal components can reveal patterns and relationships.

Q5) What are some potential limitations of PCA?

A5) PCA assumes linear relationships between variables and may not capture complex, non-linear patterns. It also requires careful interpretation, as principal components are combinations of original variables, which may not always have a straightforward or meaningful interpretation. Additionally, PCA is sensitive to scaling, so data normalization is often necessary.

Experiment – 8

Aim: To perform linear regression on datasets.

Theory: Linear regression is a fundamental technique in data science for modeling the relationship between a dependent variable and one or more independent variables.

Linear Regression: This method fits a straight line (regression line) through the dataset that best represents the relationship between variables. The equation of the line,

$$y = mx + b$$

y=mx+b, shows how changes in the independent variable(s) predict changes in the dependent variable. It's used for prediction, trend analysis, and forecasting.

Visualization: The regression line is often visualized on a scatter plot, with data points and the fitted line providing a clear representation of the correlation, making it easier to assess the model's accuracy and goodness of fit.

Source Code:

```
import numpy as np
from sklearn.linear_model import LinearRegression

x = [[0, 1], [5, 1], [15, 2], [25, 5], [35, 11], [45, 15], [55, 34], [60, 35]]
y = [4, 5, 20, 14, 32, 22, 38, 43]
x, y = np.array(x), np.array(y)

model = LinearRegression().fit(x, y)

r_sq = model.score(x, y)
print(f"coefficient of determination: {r_sq}")
print(f"intercept: {model.intercept_}")
print(f"coefficients: {model.coef_}")

y_pred = model.predict(x)
print(f"predicted response:\n{y_pred}")
```

Output:

```
coefficient of determination: 0.8615939258756776
intercept: 5.52257927519819
coefficients: [0.44706965 0.25502548]
predicted response:
[ 5.77760476  8.012953    12.73867497 17.9744479  23.97529728 29.4660957
 38.78227633 41.27265006]
```

Q1) What is linear regression, and how is it used in data analysis?

A1) Linear regression is a statistical technique used to model the relationship between a dependent variable and one or more independent variables by fitting a linear equation to observed data. It is used to predict the value of the dependent variable based on the values of the independent variables and to identify trends and relationships in the data.

Q2) How do you interpret the coefficients of a linear regression model?

A2) In linear regression, the coefficients represent the change in the dependent variable for a one-unit change in the independent variable, holding all other variables constant. A positive coefficient indicates a direct relationship, while a negative coefficient suggests an inverse relationship.

Q3) What is the purpose of the R-squared value in linear regression?

A3) The R-squared value measures the proportion of the variance in the dependent variable that is explained by the independent variables in the model. It provides an indication of how well the model fits the data, with higher values indicating a better fit.

Q4) What are some common assumptions of linear regression that should be checked?

A4) Common assumptions include linearity (the relationship between variables is linear), independence (residuals are independent), homoscedasticity (constant variance of residuals), and normality (residuals are normally distributed). Checking these assumptions ensures the validity of the regression model and its predictions.

Q5) How can you assess the quality of a linear regression model beyond R-squared?

A5) Besides R-squared, model quality can be assessed using metrics like Mean Absolute Error (MAE), Mean Squared Error (MSE), and Root Mean Squared Error (RMSE) to evaluate prediction accuracy. Residual plots and diagnostic tests can also be used to check for violations of model assumptions and to identify any patterns that might suggest model improvements.

Experiment – 9

Aim: To perform Data Aggregation and GroupWise Operations.

Theory: Data aggregation and groupwise operations are essential techniques in data science for summarizing and analyzing data based on specific groups or categories.

Data Aggregation: This process involves combining data from multiple records to calculate summary statistics like sums, averages, counts, or maximum/minimum values. It helps in simplifying large datasets to gain meaningful insights, often using functions such as groupby() in pandas for Python.

Groupwise Operations: These operations allow for applying functions or transformations to subsets of data that share common attributes. For example, grouping by a categorical variable (like region or product) and performing operations on each group helps uncover trends, patterns, and relationships that may vary across different groups.

Visualization: Groupwise data can be effectively visualized using bar charts, pie charts, or box plots to compare groups and highlight variations or similarities, making it easier to communicate insights from the data.

Source Code:

```
# import module
import pandas as pd

# Creating our dataset
df = pd.DataFrame([[9, 4, 8, 9],[8, 10, 7, 6], [7, 6, 8, 5]], columns=['Maths', 'English','Science', 'History'])

# display dataset
print(df)
print(df.sum())
print(df.describe())
print(df.agg(['sum', 'min', 'max']))

a = df.groupby('Maths')
a.first()
b = df.groupby(['Maths', 'Science'])
b.first()
```

Output:

```
        Maths  English  Science  History
0           9        4        8        9
1           8       10        7        6
2           7        6        8        5
Maths       24
English     20
Science     23
History     20
```

```
dtype: int64
       Maths      English    Science    History
count   3.0     3.000000    3.000000   3.000000
mean    8.0     6.666667    7.666667   6.666667
std     1.0     3.055050    0.577350   2.081666
min     7.0     4.000000    7.000000   5.000000
25%     7.5     5.000000    7.500000   5.500000
50%     8.0     6.000000    8.000000   6.000000
75%     8.5     8.000000    8.000000   7.500000
max     9.0    10.000000    8.000000   9.000000
      Maths  English   Science   History
sum     24      20        23       20
min      7       4         7        5
max      9      10         8        9
```

|:

Maths	Science	English	History
7	8	6	5
8	7	10	6
9	8	4	9

Viva-Voce:

Q1) What is data aggregation, and why is it important in data analysis?

A1) Data aggregation involves summarizing and combining data from multiple records or sources to compute aggregate values such as sums, averages, or counts. It is important because it helps in simplifying large datasets, providing meaningful summaries and insights, and facilitating comparative analysis.

Q2) How do you perform groupwise operations in Python using pandas?

A2) In Python, pandas provides the groupby() function to perform groupwise operations. By grouping data based on one or more columns, you can apply aggregate functions (like sum, mean, count) or transformations to each group. For example, df.groupby('column').mean() calculates the mean of each group in the specified column.

Q3) What are some common aggregation functions used in data analysis?

A3) Common aggregation functions include sum (total value), mean (average value), median (middle value), count (number of entries), min (minimum value), and max (maximum value). These functions help in summarizing the data and extracting key metrics.

Q4) How can you handle missing values during aggregation?

A4) During aggregation, missing values can be handled by using functions like fillna() to impute missing values or by choosing to ignore them with options like dropna(). Aggregation functions often have parameters to handle missing values, such as skipna=True in pandas, which excludes NaNs from calculations.

Q5) What is the difference between aggregation and transformation in groupwise operations?

A5) Aggregation involves computing summary statistics for each group, such as totals or averages, and returning a reduced dataset. Transformation, on the other hand, involves applying functions to each group.