NRI Institute of Information Science and Technology, Bhopal

Lab Manual

Program: Bachelor of Technology (Computer Science & Engineering)

Subject / Course Name: Data Analytics Lab

Subject / Course Code: CS605

Department of Computer Science and Engineering

Course Objectives:

At the end of this Data Analytics using Python course, students will be able to:

1. Understand and set up Python and Jupyter Notebook for data analysis.
2. Apply Python libraries (NumPy, Pandas) to manipulate and preprocess data.
3. Perform exploratory data analysis and create visual insights.
4. Apply statistical tests and machine learning models for data interpretation.
5. Build, evaluate, and interpret predictive and classification models.
6. Communicate data insights using visual dashboards and reports.

Course Outcomes:

By the end of the course, students should be able to:

- **CO1:** Configure and use Jupyter Notebook for data analytics tasks.
- **CO2:** Perform data cleaning, transformation, and manipulation using Python libraries.
- **CO3:** Generate descriptive statistics and meaningful visualizations.
- **CO4:** Apply statistical techniques and interpret results.
- **CO5:** Build and validate predictive models using machine learning algorithms.
- **CO6:** Develop interactive visual reports and dashboards.

CO – PO Mapping :

CO	Program Outcomes (POs)	Rationale
CO1	PO5, PO1	Use modern tools (Jupyter, Python) and apply computing fundamentals.
CO2	PO2, PO4, PO5	Identify, analyze data problems, apply data prep & preprocessing.
CO3	PO2, PO5, PO10	Visualize data and communicate results effectively.
CO4	PO4, PO2	Interpret data trends using statistical methods (investigation/inference).
CO5	PO1, PO5, PO4	Develop models (ML) solving real problems.
CO6	PO9, PO10	Communicate findings, document dashboards/reports.

List of Experiment

Experiment No.	Experiment
1	To install Anaconda, set up Jupyter Notebook environment, and execute basic Python programs including variables, lists, loops, and functions.
2	To create NumPy arrays and perform indexing, slicing, vectorized operations, and matrix manipulations.
3	To import real-world datasets (CSV files) using Pandas and perform data inspection, filtering, sorting, and missing value handling operations.
4	To perform Exploratory Data Analysis (EDA) on a dataset by computing summary statistics (mean, median, variance), identifying missing values and outliers, and visualizing data using histograms and boxplots.
5	To create and interpret various data visualizations such as bar charts, histograms, scatter plots, and heatmaps using Matplotlib and Seaborn libraries in Python.
6	To compute correlation coefficients between numerical variables and visualize the relationship using scatter plots with regression lines using seaborn.regplot().
7	To perform statistical hypothesis testing using: • Z-Test • Independent T-Test • One-Way ANOVA and interpret p-values to draw statistical conclusions..
8	To build a predictive model using Linear Regression to predict house prices, perform train-test split, and evaluate the model using RMSE and R^2 score.
9	To build a Logistic Regression classification model to predict binary outcomes (diabetes: Yes/No) and evaluate the model using: • Accuracy • Confusion Matrix • ROC Curve & AUC
10	To apply K-Means clustering algorithm on a dataset, interpret cluster groupings, evaluate clustering quality using Silhouette Score**,** and visualize clusters graphically.

Experiment No. : 1

To install Anaconda, set up Jupyter Notebook environment, and execute basic Python programs including variables, lists, loops, and functions.

Tools & Software Required

- Windows / Linux / macOS
- Internet connection
- Anaconda Distribution (Latest Version)
- Web Browser (Chrome/Edge/Firefox)

Theory

What is Python?

Python is a high-level, interpreted programming language widely used in:

- Data Analytics
- Machine Learning
- Artificial Intelligence
- Web Development

What is Anaconda?

Anaconda is a distribution of Python that includes:

- Python interpreter
- Jupyter Notebook
- Scientific libraries (NumPy, Pandas, Matplotlib, etc.)
- Package manager (Conda)

What is Jupyter Notebook?

Jupyter Notebook is an interactive environment that allows:

- Writing Python code in cells
- Writing explanations in Markdown
- Visualizing graphs
- Executing code step-by-step

Step-by-Step Installation Procedure

Step 1: Download Anaconda

1. Go to official website: https://www.anaconda.com/products/distribution
2. Click **Download**
3. Select your Operating System (Windows 64-bit recommended)
4. Download Python 3.x version

Step 2: Install Anaconda

1. Double-click downloaded file
2. Click **Next**
3. Agree to License Agreement
4. Choose “Just Me”
5. Select Installation Path
6. Click Install
7. Finish Installation

Step 3: Launch Jupyter Notebook

Method 1:

- Open Start Menu
- Search **Anaconda Navigator**
- Click Launch
- Click “Launch” under Jupyter Notebook

Method 2:

- Open Command Prompt
- Type:

jupyter notebook

- Press Enter

Step 4: Create New Notebook

1. Click **New**
2. Select **Python 3**
3. Rename file as:
4. Experiment_1_Python_Basics.ipynb

First Python Program

In First Code Cell:

```
print("Hello, Data Analytics Lab!")
```

Output:

Hello, Data Analytics Lab!

Python Basics Practical

A) Variables

Theory:

Variables store data values.

Code:

```
# Integer
a = 10

# Float
b = 20.5

# String
name = "Aayush"

print("Value of a:", a)
print("Value of b:", b)
print("Name:", name)
```

B) Data Types

```
x = 10
y = 3.14
z = "Python"

print(type(x))
print(type(y))
print(type(z))
```

C) Lists

Theory:

List is a collection of items enclosed in square brackets [].

```
numbers = [1, 2, 3, 4, 5]

print("List:", numbers)
print("First Element:", numbers[0])
print("Last Element:", numbers[-1])
```

List Operations

```
numbers.append(6)
print("After Append:", numbers)

numbers.remove(3)
print("After Remove:", numbers)
```

D) Loops

1) For Loop

```
for i in range(1,6):
    print("Number:", i)
```

2) While Loop

```
count = 1

while count <= 5:
    print("Count:", count)
    count += 1
```

E) Conditional Statements

```
num = 10

if num > 0:
    print("Positive Number")
else:
    print("Negative Number")
```

F) Functions

Theory:

Function is a reusable block of code.

```
def add(a, b):
    return a + b

result = add(10, 20)
print("Sum:", result)
```

Practice Program

Program to find square of numbers in list

```
numbers = [1, 2, 3, 4, 5]

for n in numbers:
    print("Square of", n, "=", n**2)
```

Markdown Documentation in Jupyter

To write theory inside notebook:

1. Click on cell

2. Change type from **Code → Markdown**
3. Write:

Experiment 1
Python Basics
This experiment demonstrates variables, loops, and functions.
Press **Shift + Enter**

Sample Combined Program

```
# Student Data Program

name = "Rahul"
marks = [85, 90, 78]

average = sum(marks)/len(marks)

print("Student Name:", name)
print("Marks:", marks)
print("Average:", average)

if average > 80:
   print("Grade: A")
else:
   print("Grade: B")
```

Expected Output

- Successful installation of Anaconda
- Jupyter Notebook running
- Python programs executed without errors
- Markdown documentation created

Result

Anaconda was successfully installed.
Jupyter Notebook environment was configured.
Basic Python programs including variables, lists, loops, and functions were executed successfully.

Viva Questions & Answers

Q1: What is Anaconda?
Ans: Anaconda is a Python distribution that includes scientific libraries and Jupyter Notebook.
Q2: What is Jupyter Notebook?
Ans: It is an interactive web-based environment for coding and documentation.
Q3: Difference between list and tuple?
Ans: List is mutable; tuple is immutable.
Q4: What is a function?
Ans: A reusable block of code that performs a specific task.
Q5: What is indentation in Python?
Ans: Indentation defines code blocks in Python.

Assignment

1. Write a program to find factorial of a number.
2. Write a program to check prime number.
3. Create a list of 10 numbers and find maximum and minimum.

Experiment No. : 2

To create NumPy arrays and perform indexing, slicing, vectorized operations, and matrix manipulations.

Theory

What is NumPy?

NumPy (Numerical Python) is a fundamental library for:

- Scientific computing
- Multi-dimensional arrays
- Mathematical functions
- Linear algebra

Why NumPy?

- Faster than Python lists
- Supports vectorized operations
- Efficient memory usage
- Useful for machine learning & analytics

Procedure

1. Open Jupyter Notebook
2. Create new notebook
3. Import NumPy
4. Perform array operations step-by-step
5. Execute each cell and observe output

Practical Implementation

PART A: Creating NumPy Arrays

1. Import Library

```
import numpy as np
```

2. Create 1D Array

```
arr1 = np.array([10, 20, 30, 40, 50])
print("1D Array:", arr1)
```

3. Create 2D Array

```
arr2 = np.array([[1, 2, 3],
                 [4, 5, 6]])

print("2D Array:\n", arr2)
```

4. Create Special Arrays

```
zeros = np.zeros((2,3))
ones = np.ones((3,3))
identity = np.eye(3)

print("Zeros Matrix:\n", zeros)
print("Ones Matrix:\n", ones)
print("Identity Matrix:\n", identity)
```

PART B: Indexing & Slicing

1. Indexing in 1D Array

```
print("First Element:", arr1[0])
print("Last Element:", arr1[-1])
```

2. Indexing in 2D Array

```
print("Element at row 1, col 2:", arr2[1,2])
```

3. Slicing 1D Array

```
print("Elements from index 1 to 3:", arr1[1:4])
```

4. Slicing 2D Array

```
print("First Row:", arr2[0, :])
print("Second Column:", arr2[:, 1])
```

PART C: Vectorized Operations

What is Vectorization?

Performing operations on entire array without loops.

1. Arithmetic Operations

```
a = np.array([1,2,3])
b = np.array([4,5,6])

print("Addition:", a + b)
print("Subtraction:", a - b)
print("Multiplication:", a * b)
print("Division:", a / b)
```

2. Scalar Operations

```
print("Multiply by 2:", a * 2)
print("Square:", a ** 2)
```

3. Statistical Operations

```
data = np.array([10,20,30,40,50])

print("Mean:", np.mean(data))
print("Median:", np.median(data))
print("Standard Deviation:", np.std(data))
print("Sum:", np.sum(data))
```

PART D: Matrix Manipulations

1. Matrix Addition

```
A = np.array([[1,2],
              [3,4]])

B = np.array([[5,6],
              [7,8]])

print("Matrix Addition:\n", A + B)
```

2. Matrix Multiplication

```
print("Matrix Multiplication:\n", np.dot(A, B))
```

3. Transpose of Matrix

```
print("Transpose of A:\n", A.T)
```

4. Determinant

```
print("Determinant of A:", np.linalg.det(A))
```

5. Inverse of Matrix

```
print("Inverse of A:\n", np.linalg.inv(A))
```

PART E: Performance Comparison (Vectorization vs Loop)

```
import time

arr = np.arange(1000000)

# Vectorized
start = time.time()
arr_square = arr ** 2
end = time.time()
print("Vectorized Time:", end - start)

# Using Loop
start = time.time()
result = []
for i in arr:
   result.append(i**2)
end = time.time()
print("Loop Time:", end - start)
```

Observe that vectorized operations are much faster.

Expected Output

- Arrays created successfully
- Indexing and slicing performed
- Mathematical operations executed
- Matrix multiplication & inverse computed
- Performance difference observed

Result

NumPy arrays were successfully created.
Indexing, slicing, vectorized operations, and matrix manipulations were performed and analyzed.

Viva Questions & Answers

Q1: What is NumPy?

Ans: A Python library for numerical computing and array operations.

Q2: Difference between list and NumPy array?

Ans: NumPy arrays are faster and support vectorized operations.

Q3: What is vectorization?

Ans: Performing operations on entire arrays without using loops.

Q4: How to find inverse of matrix in NumPy?

Ans: Using np.linalg.inv() function.

Q5: What is difference between dot() and * operator?

Ans: * performs element-wise multiplication; dot() performs matrix multiplication.

Lab Assignment (For Practice)

1. Create a 3x3 matrix and find its transpose and determinant.
2. Create two arrays and perform element-wise multiplication.
3. Generate random numbers and calculate mean & standard deviation.

Experiment No. : 3

To import real-world datasets (CSV files) using Pandas and perform data inspection, filtering, sorting, and missing value handling operations.

Theory

What is Pandas?

Pandas is a powerful Python library used for:

- Data manipulation
- Data cleaning
- Data analysis
- Working with structured datasets

Key Data Structures

1. **Series** – One-dimensional labeled array
2. **DataFrame** – Two-dimensional labeled table

Dataset Used in Experiment

You may use:

- Any CSV file (e.g., student_marks.csv)
- OR built-in dataset created manually

For lab demonstration, we will create a sample CSV file.

Practical Implementation

PART A: Importing Dataset

Step 1: Create Sample CSV File (Optional)

Create a file named:

students.csv

Example Data:

Name	Age	Marks	City
Amit	20	85	Bhopal
Riya	21	90	Indore
John	19	78	Bhopal
Sara	22	92	Delhi
Rahul	20	75	Indore

Step 2: Import Pandas

```
import pandas as pd
```

Step 3: Load CSV File

```
df = pd.read_csv("students.csv")
print(df)
```

PART B: Exploring the Dataset

View First 5 Rows

```
print(df.head())
```

View Last 5 Rows

```
print(df.tail())
```
Dataset Information
```
print(df.info())
```

Statistical Summary
```
print(df.describe())
```

PART C: Data Selection & Filtering
Select Single Column
```
print(df["Marks"])
```
Select Multiple Columns
```
print(df[["Name", "Marks"]])
```
Filter Rows (Marks > 80)
```
high_marks = df[df["Marks"] > 80]
print(high_marks)
```
Filter by City
```
bhopal_students = df[df["City"] == "Bhopal"]
print(bhopal_students)
```

PART D: Sorting Data

Sort by Marks (Ascending)
```
sorted_df = df.sort_values(by="Marks")
print(sorted_df)
```

Sort by Marks (Descending)
```
sorted_df_desc = df.sort_values(by="Marks", ascending=False)
print(sorted_df_desc)
```
PART E: Handling Missing Values
Step 1: Introduce Missing Values (For Practice)
```
df.loc[2, "Marks"] = None
print(df)
```

Step 2: Check Missing Values
```
print(df.isnull())
print(df.isnull().sum())
```

Step 3: Drop Missing Values
```
df_cleaned = df.dropna()
print(df_cleaned)
```

Step 4: Fill Missing Values
```
df["Marks"].fillna(df["Marks"].mean(), inplace=True)
print(df)
```

PART F: Data Cleaning Routines

Rename Columns

```
df.rename(columns={"Marks": "Score"}, inplace=True)
print(df)
```

Remove Duplicate Rows

```
df.drop_duplicates(inplace=True)
```

Change Data Type

```
df["Age"] = df["Age"].astype(int)
print(df.dtypes)
```

PART G: Grouping & Aggregation

Group by City

```
city_avg = df.groupby("City")["Score"].mean()
print(city_avg)
```

Count Students in Each City

```
city_count = df["City"].value_counts()
print(city_count)
```

Expected Output

✔ CSV file successfully imported

✔ Dataset explored using head(), info(), describe()

✔ Data filtered and sorted

✔ Missing values handled

✔ Data cleaned and grouped

Result

The dataset was successfully imported using Pandas.
Data exploration, filtering, sorting, and missing value handling were performed effectively.

Viva Questions & Answers

Q1: What is a DataFrame?

Ans: A two-dimensional labeled data structure in Pandas.

Q2: How to load CSV file?

Ans: Using pd.read_csv() function.

Q3: What is the use of describe()?

Ans: It provides statistical summary of numerical columns.

Q4: How to detect missing values?

Ans: Using isnull() function.

Q5: Difference between dropna() and fillna()?

Ans: dropna() removes missing rows; fillna() replaces missing values.

Q6: What is groupby()?

Ans: It is used to group data and apply aggregation functions.

Lab Assignment

1. Load any real dataset (e.g., sales.csv) and display summary statistics.
2. Filter records where sales > 5000.
3. Replace missing values with median.
4. Sort dataset by two columns.

EXPERIMENT – 4

To perform Exploratory Data Analysis (EDA) on a dataset by computing summary statistics (mean, median, variance), identifying missing values and outliers, and visualizing data using histograms and boxplots.

Tools Required

- Python 3.x
- Jupyter Notebook
- Pandas
- NumPy
- Matplotlib
- Seaborn

Install required libraries (if needed):
pip install pandas numpy matplotlib seaborn

Theory

What is Exploratory Data Analysis (EDA)?

EDA is the process of:

- Understanding dataset structure
- Summarizing key characteristics
- Identifying missing values
- Detecting outliers
- Visualizing data distributions

EDA helps in making decisions before applying machine learning models.

Dataset Used

We will use the built-in **Titanic dataset** from Seaborn.

Practical Implementation

PART A: Import Libraries

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

PART B: Load Dataset

```
df = sns.load_dataset("titanic")
print(df.head())
```

PART C: Dataset Overview

Structure of Dataset

```
print(df.info())
```

Statistical Summary

```
print(df.describe())
```

This provides:

- Mean
- Standard Deviation
- Minimum
- Maximum
- Quartiles

PART D: Manual Summary Statistics

Mean

```
print("Mean Age:", df["age"].mean())
```

Median

```
print("Median Age:", df["age"].median())
```

Variance

```
print("Variance of Age:", df["age"].var())
```

Standard Deviation

```
print("Standard Deviation of Age:", df["age"].std())
```

PART E: Identifying Missing Values

Check Missing Values

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

Percentage of Missing Values

```
missing_percent = df.isnull().mean() * 100
print(missing_percent)
```

Handle Missing Values

Fill missing Age with mean:

```
df["age"].fillna(df["age"].mean(), inplace=True)
```

PART F: Identifying Outliers

Method 1: Using IQR (Interquartile Range)

```
Q1 = df["fare"].quantile(0.25)
Q3 = df["fare"].quantile(0.75)
IQR = Q3 - Q1

lower_bound = Q1 - 1.5 * IQR
upper_bound = Q3 + 1.5 * IQR

outliers = df[(df["fare"] < lower_bound) | (df["fare"] > upper_bound)]
print("Number of Outliers:", len(outliers))
```

PART G: Data Visualization

Histogram

```
plt.figure(figsize=(6,4))
plt.hist(df["age"], bins=20, color="skyblue", edgecolor="black")
plt.title("Age Distribution")
plt.xlabel("Age")
plt.ylabel("Frequency")
plt.show()
```

Boxplot (Outlier Detection)

```
plt.figure(figsize=(6,4))
sns.boxplot(x=df["fare"])
plt.title("Fare Boxplot")
plt.show()
```

Correlation Heatmap

```
plt.figure(figsize=(8,6))
sns.heatmap(df.corr(numeric_only=True), annot=True, cmap="coolwarm")
```

```
plt.title("Correlation Heatmap")
plt.show()
```

PART H: Interpretation

Students should observe:

- Age distribution pattern
- Presence of missing values
- Outliers in Fare column
- Relationship between numerical variables

Expected Output

✔ Summary statistics generated

✔ Missing values identified

✔ Outliers detected using IQR

✔ Histogram plotted

✔ Boxplot displayed

Result

Exploratory Data Analysis was successfully performed on the Titanic dataset.
Summary statistics were calculated, missing values were identified and handled, and outliers were detected using boxplots and IQR method.

Viva Questions & Answers

Q1: What is EDA?

Ans: Process of analyzing datasets to summarize main characteristics.

Q2: What does describe() function do?

Ans: Provides statistical summary of numerical columns.

Q3: What is variance?

Ans: Measure of spread of data around mean.

Q4: What is IQR?

Ans: Interquartile Range = Q3 − Q1.

Q5: Why use boxplot?

Ans: To detect outliers and visualize distribution.

Q6: How to handle missing values?

Ans: Using fillna() or dropna().

Lab Assignment

1. Perform EDA on any real CSV dataset.
2. Identify top 5 outliers in any numeric column.
3. Plot histogram for two different features.
4. Compute correlation between two variables.

EXPERIMENT – 5

To create and interpret various data visualizations such as **bar charts, histograms, scatter plots, and heatmaps** using Matplotlib and Seaborn libraries in Python.

Tools & Requirements

- Python 3.x
- Jupyter Notebook
- Libraries: Pandas, Matplotlib, Seaborn

Install (if required):

pip install pandas matplotlib seaborn

Theory

What is Data Visualization?

Data visualization is the graphical representation of data to:

- Identify patterns
- Detect trends
- Understand relationships
- Identify outliers
- Communicate insights effectively

Matplotlib

- Basic plotting library in Python
- Highly customizable

Seaborn

- Built on Matplotlib
- More aesthetic and statistical plots
- Better for correlation and categorical data visualization

Dataset Used

We will use **Titanic dataset (built-in Seaborn dataset)**.

Practical Implementation

Step 1: Import Libraries

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

Step 2: Load Dataset

```
df = sns.load_dataset("titanic")
df.head()
```

Part A: Bar Chart

Aim:

Visualize survival count by gender.

```
plt.figure(figsize=(6,4))
sns.countplot(x="sex", hue="survived", data=df)
plt.title("Survival Count by Gender")
plt.xlabel("Gender")
plt.ylabel("Count")
plt.show()
```

Interpretation:

- Compares survival status across gender.
- Helps understand survival distribution.

Part B: Histogram

Aim:

Show age distribution.

```
plt.figure(figsize=(6,4))
plt.hist(df["age"].dropna(), bins=20)
plt.title("Age Distribution")
plt.xlabel("Age")
plt.ylabel("Frequency")
plt.show()
```

Interpretation:

- Displays frequency of age groups.
- Identifies spread and skewness.

Part C: Scatter Plot

Aim:

Relationship between Age and Fare.

```
plt.figure(figsize=(6,4))
sns.scatterplot(x="age", y="fare", data=df)
plt.title("Age vs Fare")
plt.xlabel("Age")
plt.ylabel("Fare")
plt.show()
```

Interpretation:

- Shows relationship between two numerical variables.
- Helps detect clusters or correlation.

Part D: Heatmap

Aim:

Display correlation between numeric variables.

```
plt.figure(figsize=(8,6))
correlation_matrix = df.corr(numeric_only=True)
sns.heatmap(correlation_matrix, annot=True)
plt.title("Correlation Heatmap")
plt.show()
```

Interpretation:

- Identifies strong positive/negative relationships.
- Useful for feature selection.

Part E: Customization Techniques

Change Style

```
sns.set_style("whitegrid")
```

Add Grid

```
plt.grid(True)
```

Change Color Palette

```
sns.set_palette("Set2")
```

Rotate Labels

plt.xticks(rotation=45)

Expected Output

✔ Bar chart showing survival by gender

✔ Histogram showing age distribution

✔ Scatter plot showing relationship

✔ Heatmap displaying correlations

✔ Customized visualizations

Result

The experiment successfully demonstrated various visualization techniques using Matplotlib and Seaborn. Different types of plots were generated and interpreted to understand data distribution, relationships, and correlations.

Viva Questions & Answers

Q1. What is data visualization?

Ans: It is the graphical representation of data to extract insights.

Q2. What is the difference between bar chart and histogram?

Ans: Bar chart is for categorical data; histogram is for continuous data.

Q3. Why use scatter plot?

Ans: To analyze relationship between two numerical variables.

Q4. What does correlation heatmap show?

Ans: Strength and direction of relationship between variables.

Q5. What is Seaborn used for?

Ans: Statistical data visualization with better aesthetics.

Q6. What is Matplotlib?

Ans: Core Python plotting library.

Lab Assignment

1. Create a bar chart showing passenger class distribution.
2. Plot histogram of Fare column.
3. Create scatter plot of Age vs Fare with hue = gender.
4. Generate correlation heatmap and identify strongest correlated features.

EXPERIMENT – 6

To compute correlation coefficients between numerical variables and visualize the relationship using scatter plots with regression lines using seaborn.regplot().

Total Lab Duration

Tools Required

- Python 3.x
- Jupyter Notebook
- Libraries: Pandas, NumPy, Matplotlib, Seaborn

Install if required:

pip install pandas numpy matplotlib seaborn

Theory Background

6.1 Correlation

Correlation measures the **strength and direction** of relationship between two variables.

Pearson Correlation Coefficient (r)

r=Cov(X,Y)σXσYr = \frac{Cov(X,Y)}{\sigma_X \sigma_Y}r=σXσYCov(X,Y)

Range of Correlation:

Value of r	Interpretation
+1	Perfect positive correlation
0	No correlation
-1	Perfect negative correlation

6.2 Regression

Regression predicts the value of a dependent variable (Y) based on independent variable (X).

Simple Linear Regression Equation:

Y=a+bXY = a + bXY=a+bX

aaa

bbb

-10-8-6-4-2246810-5510

Where:

- a = Intercept
- b = Slope

Dataset Used

We will use **Seaborn's Titanic dataset**.

Practical Implementation

Step 1: Import Libraries

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

Step 2: Load Dataset

```
df = sns.load_dataset("titanic")
df.head()
```

Step 3: Select Numerical Variables

We will analyze relationship between:

- Age

- Fare

```
data = df[["age", "fare"]].dropna()
data.head()
```

Part A: Compute Correlation
Using Pandas

```
correlation = data.corr()
print("Correlation Matrix:")
print(correlation)
```

Compute Pearson Correlation Directly

```
corr_value = data["age"].corr(data["fare"])
print("Correlation between Age and Fare:", corr_value)
```

Interpretation

- If $r > 0 \rightarrow$ Positive relationship
- If $r < 0 \rightarrow$ Negative relationship
- If $r \approx 0 \rightarrow$ Weak or no relationship

Part B: Scatter Plot

```
plt.figure(figsize=(6,4))
sns.scatterplot(x="age", y="fare", data=data)
plt.title("Scatter Plot: Age vs Fare")
plt.xlabel("Age")
plt.ylabel("Fare")
plt.show()
```

Interpretation

- Each dot represents a passenger.
- Shows relationship pattern visually.

Part C: Regression Line using regplot

```
plt.figure(figsize=(6,4))
sns.regplot(x="age", y="fare", data=data)
plt.title("Regression Line: Age vs Fare")
plt.xlabel("Age")
plt.ylabel("Fare")
plt.show()
```

Interpretation

- The straight line represents predicted values.
- Shaded region represents confidence interval.
- Slope indicates direction of relationship.

Part D: Calculate Regression Coefficients Manually
Using NumPy:

```
x = data["age"]
y = data["fare"]

# Calculate slope and intercept
b, a = np.polyfit(x, y, 1)
```

print("Slope (b):", b)
print("Intercept (a):", a)
Regression Equation:
Fare=a+b(Age)Fare = a + b(Age)Fare=a+b(Age)

Part E: Predict Value

predicted_fare = a + b * 30
print("Predicted Fare for Age 30:", predicted_fare)

Expected Output

✔ Correlation coefficient printed
✔ Scatter plot displayed
✔ Regression line plotted
✔ Regression equation computed
✔ Predicted value calculated

Result

The experiment successfully computed the correlation coefficient and performed simple linear regression analysis. The relationship between Age and Fare was visualized using scatter plots and regression lines.

Viva Questions & Answers

Q1: What is correlation?

Ans: It measures strength and direction of relationship between two variables.

Q2: What is Pearson correlation?

Ans: A statistical measure of linear relationship between two numerical variables.

Q3: What is regression?

Ans: A predictive technique that models relationship between dependent and independent variables.

Q4: What does slope represent?

Ans: Rate of change of dependent variable with respect to independent variable.

Q5: What is intercept?

Ans: Value of Y when X = 0.

Q6: What does regplot do?

Ans: It creates scatter plot with regression line and confidence interval.

Lab Assignment

1. Compute correlation between Fare and Pclass.
2. Plot regression between Fare and Pclass.
3. Predict Fare for Age = 45 using regression equation.
4. Interpret whether relationship is strong or weak.

Experiment No. 7

To perform statistical hypothesis testing using: Z-Test, Independent T-Test, One-Way ANOVA and interpret **p-values** to draw statistical conclusions..

Tools Required

- Python 3.x
- Jupyter Notebook
- Libraries: NumPy, Pandas, SciPy

Install if required:
pip install numpy pandas scipy

Theory Background

Hypothesis Testing
A statistical method used to make decisions about population parameters.
Steps in Hypothesis Testing:

1. Define Null Hypothesis (H_0)
2. Define Alternative Hypothesis (H_1)
3. Select significance level ($\alpha = 0.05$)
4. Compute test statistic
5. Compare p-value with α
6. Draw conclusion

Decision Rule

Condition	Conclusion
p-value ≤ 0.05	Reject H_0
p-value > 0.05	Fail to Reject H_0

PART A: Z-Test
Z-Test
When to Use:

- Population standard deviation known
- Large sample size ($n > 30$)

Problem Statement:
Test whether the average height of students differs from 170 cm.
Sample mean = 172
Population std deviation = 5
Sample size = 50

Step 1: Import Libraries

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

Step 2: Define Values

```
sample_mean = 172
population_mean = 170
population_std = 5
```

n = 50

Step 3: Compute Z-Statistic
z = (sample_mean - population_mean) / (population_std / np.sqrt(n))
print("Z-Statistic:", z)

Step 4: Compute p-value (Two-tailed)
p_value = 2 * (1 - stats.norm.cdf(abs(z)))
print("P-value:", p_value)

Interpretation:
- If p-value < 0.05 → Reject H_0
- Otherwise → No significant difference

PART B: T-Test

Independent T-Test
When to Use:
- Compare means of two groups
- Population standard deviation unknown

Problem Statement:
Compare exam scores of two classes.

Step 1: Sample Data
class_A = [78, 85, 90, 88, 76, 95, 89]
class_B = [72, 80, 75, 70, 68, 74, 78]

Step 2: Perform T-Test
t_stat, p_value = stats.ttest_ind(class_A, class_B)
print("T-Statistic:", t_stat)
print("P-value:", p_value)

Interpretation:
- If p-value < 0.05 → Significant difference between classes
- Otherwise → No significant difference

PART C: One-Way ANOVA

ANOVA (Analysis of Variance)
When to Use:
- Compare means of more than two groups

Problem Statement:
Compare performance of 3 teaching methods.

Step 1: Sample Data
group1 = [85, 88, 90, 92, 87]
group2 = [78, 75, 80, 79, 77]
group3 = [90, 93, 95, 94, 92]

Step 2: Perform ANOVA
f_stat, p_value = stats.f_oneway(group1, group2, group3)
print("F-Statistic:", f_stat)
print("P-value:", p_value)

Interpretation:
- If p-value < 0.05 → At least one group mean differs
- If p-value > 0.05 → No significant difference

Expected Output
✔ Z-statistic and p-value
✔ T-statistic and p-value
✔ F-statistic and p-value
✔ Proper statistical conclusion

Result
The experiment successfully demonstrated:
- Z-test for population mean comparison
- T-test for comparing two groups
- ANOVA for comparing multiple groups
- Interpretation of p-values for decision making

Viva Questions & Answers
Q1: What is Null Hypothesis?
Ans: A statement assuming no effect or no difference.
Q2: What is p-value?
Ans: Probability of obtaining results assuming H_0 is true.
Q3: When do we reject H_0?
Ans: When p-value ≤ 0.05.
Q4: Difference between Z-test and T-test?
Ans: Z-test uses known population std; T-test uses sample std.
Q5: What is ANOVA?
Ans: A method to compare means of more than two groups.
Q6: What does F-statistic indicate?
Ans: Ratio of variance between groups to variance within groups.

Lab Assignment
1. Perform one-sample t-test using random dataset.
2. Compare salaries of two departments using t-test.
3. Apply ANOVA on dataset with 4 groups.
4. Interpret results using significance level 0.01.

Experiment No. – 8

To build a predictive model using **Linear Regression** to predict house prices, perform **train-test split**, and evaluate the model using **RMSE and R² score**.

Tools Required

- Python 3.x
- Jupyter Notebook
- Libraries: Pandas, NumPy, Matplotlib, Scikit-learn

Install if required:

pip install pandas numpy matplotlib scikit-learn

Theory

8.1 Linear Regression

Linear Regression models the relationship between:

- Independent variable (X)
- Dependent variable (Y)

Equation:

Y=a+bXY = a + bXY=a+bX

Where:

- Y = Predicted value
- a = Intercept
- b = Slope

8.2 Train-Test Split

- Training data → Used to build model
- Testing data → Used to evaluate model

Typical split: 70:30 or 80:20

8.3 Evaluation Metrics

RMSE (Root Mean Squared Error)

$$RMSE = \sqrt{\frac{1}{n}\sum(y - \hat{y})^2}$$

Lower RMSE = Better model

R² Score (Coefficient of Determination)

$$R^2 = 1 - \frac{SS_{res}}{SS_{tot}}$$

Range: 0 to 1

Higher R² = Better model fit

Dataset Used

We will use **California Housing Dataset** from Scikit-learn.

Practical Implementation

Step 1: Import Libraries

```
import pandas as pd
```

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

from sklearn.datasets import fetch_california_housing
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error, r2_score
```

Step 2: Load Dataset

```
housing = fetch_california_housing()

df = pd.DataFrame(housing.data, columns=housing.feature_names)
df["Price"] = housing.target

df.head()
```

Step 3: Define Features and Target

```
X = df.drop("Price", axis=1)
y = df["Price"]
```

Step 4: Train-Test Split

```
X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.2, random_state=42
)
```

Step 5: Create and Train Model

```
model = LinearRegression()
model.fit(X_train, y_train)
```

Step 6: Make Predictions

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

Step 7: Evaluate Model

RMSE

```
rmse = np.sqrt(mean_squared_error(y_test, y_pred))
print("RMSE:", rmse)
```

R^2 Score

```
r2 = r2_score(y_test, y_pred)
print("R² Score:", r2)
```

Step 8: Visualize Predictions

```
plt.figure(figsize=(6,4))
plt.scatter(y_test, y_pred)
plt.xlabel("Actual Prices")
plt.ylabel("Predicted Prices")
plt.title("Actual vs Predicted Prices")
plt.show()
```

Interpretation

- RMSE indicates average prediction error.
- R^2 shows how much variance is explained by model.
- If $R^2 \approx 0.6–0.8 \rightarrow$ Good model
- Scatter plot should show points near diagonal line.

Expected Output

✔ Dataset loaded
✔ Model trained successfully
✔ RMSE value displayed
✔ R^2 score printed
✔ Scatter plot of actual vs predicted

Result

The Linear Regression model was successfully built to predict house prices. The model performance was evaluated using RMSE and R^2 metrics, and prediction accuracy was analyzed.

Viva Questions & Answers

Q1: What is Linear Regression?

Ans: A supervised learning algorithm for predicting continuous values.

Q2: What is Train-Test Split?

Ans: Dividing dataset into training and testing sets.

Q3: What is RMSE?

Ans: Root Mean Squared Error; measures prediction error.

Q4: What does R^2 represent?

Ans: Percentage of variance explained by model.

Q5: What happens if $R^2 = 1$?

Ans: Perfect prediction.

Q6: Why avoid training and testing on same data?

Ans: To prevent overfitting.

Lab Assignment

1. Try 70:30 train-test split and compare results.
2. Train model using only 3 selected features.
3. Compare RMSE and R^2 for both models.
4. Interpret whether model is underfitting or overfitting.

Experiment No. – 9

To build a **Logistic Regression classification model** to predict binary outcomes (diabetes: Yes/No) and evaluate the model using: Accuracy, Confusion Matrix, ROC Curve & AUC.

Tools Required

- Python 3.x
- Jupyter Notebook
- Libraries: Pandas, NumPy, Matplotlib, Scikit-learn

Install if required:

pip install pandas numpy matplotlib scikit-learn

Theory

9.1 Logistic Regression

Logistic Regression is a **supervised classification algorithm** used for predicting **binary outcomes**. Instead of predicting continuous values, it predicts probabilities using **Sigmoid Function**:

$$P(Y=1) = \frac{1}{1+e^{-z}}$$

Where:

$$z = b_0 + b_1 X_1 + b_2 X_2 + \ldots$$

9.2 Evaluation Metrics

Accuracy

Accuracy = Correct Predictions / Total Predictions

Confusion Matrix

	Predicted 0	**Predicted 1**
Actual 0	TN	FP
Actual 1	FN	TP

ROC Curve

- Plots TPR vs FPR
- AUC (Area Under Curve) measures model quality
- AUC close to 1 → Excellent model

Dataset Used

We will use **Breast Cancer Dataset** (binary classification) from Scikit-learn.
(Used here instead of external diabetes CSV for portability.)

Practical Implementation

Step 1: Import Libraries

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

from sklearn.datasets import load_breast_cancer
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
```

```
from sklearn.metrics import accuracy_score, confusion_matrix, roc_curve, roc_auc_score
```
Step 2: Load Dataset
```
data = load_breast_cancer()

df = pd.DataFrame(data.data, columns=data.feature_names)
df["target"] = data.target

df.head()
```
Target:

- 0 = Malignant
- 1 = Benign

Step 3: Define Features & Target
```
X = df.drop("target", axis=1)
y = df["target"]
```
Step 4: Train-Test Split
```
X_train, X_test, y_train, y_test = train_test_split(
   X, y, test_size=0.2, random_state=42
)
```
Step 5: Create and Train Model
```
model = LogisticRegression(max_iter=5000)
model.fit(X_train, y_train)
```
Step 6: Make Predictions
```
y_pred = model.predict(X_test)
y_prob = model.predict_proba(X_test)[:,1]
```
Part A: Accuracy
```
accuracy = accuracy_score(y_test, y_pred)
print("Accuracy:", accuracy)
```
Part B: Confusion Matrix
```
cm = confusion_matrix(y_test, y_pred)
print("Confusion Matrix:")
print(cm)
```
Interpretation:

- TP → Correct positive
- TN → Correct negative
- FP → Incorrect positive
- FN → Incorrect negative

Part C: ROC Curve
```
fpr, tpr, thresholds = roc_curve(y_test, y_prob)
auc = roc_auc_score(y_test, y_prob)

plt.figure(figsize=(6,4))
plt.plot(fpr, tpr)
plt.xlabel("False Positive Rate")
plt.ylabel("True Positive Rate")
plt.title("ROC Curve (AUC = %.2f)" % auc)
plt.show()
```

Interpretation

- High Accuracy → Good overall performance
- High TP & TN → Correct classifications
- AUC close to 1 → Excellent classifier
- ROC curve closer to top-left → Better model

Expected Output

✔ Model trained successfully

✔ Accuracy value printed

✔ Confusion matrix displayed

✔ ROC curve plotted

✔ AUC score calculated

Result

The Logistic Regression model successfully classified binary outcomes.
Model performance was evaluated using accuracy, confusion matrix, and ROC-AUC curve.

Viva Questions & Answers

Q1: What is Logistic Regression?

Ans: A classification algorithm for predicting binary outcomes.

Q2: Why use sigmoid function?

Ans: To convert output into probability between 0 and 1.

Q3: What is confusion matrix?

Ans: A table showing correct and incorrect predictions.

Q4: What is ROC curve?

Ans: Graph between True Positive Rate and False Positive Rate.

Q5: What does AUC represent?

Ans: Area under ROC curve; measures classifier performance.

Q6: Difference between Linear and Logistic Regression?

Ans: Linear predicts continuous values; Logistic predicts probabilities for classification.

Lab Assignment

1. Try different test sizes (30%, 40%) and compare accuracy.
2. Calculate Precision and Recall.
3. Interpret model performance if AUC < 0.6.
4. Apply logistic regression on any other binary dataset.

Experiment No. : 10

To apply K-Means clustering algorithm on a dataset, interpret cluster groupings, evaluate clustering quality using Silhouette Score**,** and visualize clusters graphically.

Tools Required

- Python 3.x
- Jupyter Notebook
- Libraries: Pandas, NumPy, Matplotlib, Scikit-learn

Theory

10.1 What is Clustering?

Clustering is an **unsupervised learning technique** used to group similar data points together without labeled output.

10.2 K-Means Algorithm

Steps:

1. Choose number of clusters (K)
2. Initialize centroids randomly
3. Assign points to nearest centroid
4. Update centroids
5. Repeat until convergence

10.3 Silhouette Score

Measures how well data points fit within their cluster.

SilhouetteScore=b−amax(a,b)Silhouette Score = \frac{b - a}{max(a, b)}SilhouetteScore=max(a,b)b−a

Where:

- a = Mean intra-cluster distance
- b = Mean nearest-cluster distance

Range:

- +1 → Well clustered
- 0 → Overlapping clusters
- -1 → Misclassified

Dataset Used

We will use **Iris Dataset** (without target labels for clustering).

Practical Implementation

Step 1: Import Libraries

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

from sklearn.datasets import load_iris
from sklearn.cluster import KMeans
from sklearn.metrics import silhouette_score
```

Step 2: Load Dataset

```
iris = load_iris()

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

We will ignore actual species labels to simulate unsupervised learning.

Step 3: Select Features

```
X = df
```

Part A: Apply K-Means Clustering

Choose number of clusters (K = 3)

```
kmeans = KMeans(n_clusters=3, random_state=42)
kmeans.fit(X)

labels = kmeans.labels_
```

Add Cluster Labels to DataFrame

```
df["Cluster"] = labels
df.head()
```

Part B: Evaluate using Silhouette Score

```
score = silhouette_score(X, labels)
print("Silhouette Score:", score)
```

Interpretation

- Score > 0.5 → Good clustering
- Score between 0.2–0.5 → Moderate
- Score < 0.2 → Poor clustering

Part C: Cluster Visualization

Since visualization is easier in 2D, use first two features:

```
plt.figure(figsize=(6,4))
plt.scatter(X.iloc[:, 0], X.iloc[:, 1], c=labels)
plt.xlabel("Sepal Length")
plt.ylabel("Sepal Width")
plt.title("K-Means Clustering (K=3)")
plt.show()
```

Part D: Elbow Method (Optional Analysis)

Used to determine optimal K.

```
inertia = []

for k in range(1, 10):
   km = KMeans(n_clusters=k, random_state=42)
   km.fit(X)
   inertia.append(km.inertia_)

plt.figure(figsize=(6,4))
plt.plot(range(1,10), inertia)
plt.xlabel("Number of Clusters (K)")
plt.ylabel("Inertia")
plt.title("Elbow Method")
plt.show()
```

Interpretation:

- Elbow point suggests optimal K.

Expected Output

✔ Cluster labels assigned

✔ Silhouette score displayed

✔ Cluster scatter plot generated

✔ Optional elbow graph plotted

Result

The K-Means clustering algorithm successfully grouped the dataset into clusters.
Silhouette score was calculated to evaluate clustering quality, and cluster visualization helped interpret grouping patterns.

Interpretation

- Distinct clusters indicate good separation.
- Higher silhouette score indicates better-defined clusters.
- Elbow method helps choose optimal K.

Viva Questions & Answers

Q1: What is unsupervised learning?

Ans: Learning without labeled data.

Q2: What is K-Means?

Ans: A clustering algorithm that groups similar data points.

Q3: What is centroid?

Ans: The center of a cluster.

Q4: What is inertia?

Ans: Sum of squared distances within cluster.

Q5: What is Silhouette Score?

Ans: Metric to evaluate clustering quality.

Q6: Difference between supervised and unsupervised learning?

Ans: Supervised uses labeled data; unsupervised does not.

Lab Assignment

1. Try K = 2 and K = 4. Compare silhouette scores.
2. Apply clustering on any real dataset (CSV).
3. Visualize clusters using different feature combinations.
4. Interpret if silhouette score is low (<0.2).