**Practical 1**

**Introduction to Excel**

* **Perform conditional formatting on a dataset using various criteria.**
* **Create a pivot table to analyse and summarize data.**
* **Use VLOOKUP function to retrieve information from a different worksheet or table.**
* **Perform what-if analysis using Goal Seek to determine input values for desired output.**

**Perform conditional formatting on a dataset using various criteria.**

**We perform conditional formatting on the "Profit" column to highlight cells with a profit greater than 800 using following steps:**

**Steps:**

1. Select the "Profit" column (Column E).

2. Go to the "Home" tab on the ribbon.

3. Click on "Conditional Formatting" in the toolbar.

4. Choose "Highlight Cells Rules" and then "Greater Than".

5. Enter the threshold value as 800.

6. Customize the formatting options (e.g., choose a fill color).

7. Click "OK" to apply the rule.

**Create a pivot table to analyse and summarize data.**

**Following are the steps to create a pivot table to analyse total sales by category.**

**Steps:**

1. Select the entire dataset including headers.

2. Go to the "Insert" tab on the ribbon.

3. Click on "PivotTable".

4. Choose where you want to place the PivotTable (e.g., new worksheet).

5. Drag "Category" to the Rows area.

6. Drag "Sales" to the Values area, choosing the sum function.

**Use VLOOKUP function to retrieve information from a different worksheet or table.**

**Use the VLOOKUP function to retrieve the category of "Product M" from a separate worksheet named "Product Table" using following steps:**

**Steps:**

1. Assuming your "Product Table" is in a different worksheet.

2. In a cell in your main dataset, enter the formula:

=VLOOKUP("M", 'Product Table'!A:B, 2, FALSE)

**Perform what-if analysis using Goal Seek to determine input values for desired output.**

**Use Goal Seek to find the required sales for "Product P" to achieve a profit of 1000 using the following steps.**

**Steps:**

1. Identify the cell containing the formula for "Profit" for "Product P" (let's assume it's in cell E17).

2. Go to the "Data" tab on the ribbon.

3. Click on "What-If Analysis" and select "Goal Seek".

4. Set "Set cell" to the profit cell (E17), "To value" to 1000, and "By changing cell" to the sales cell (C17).

5. Click "OK" to let Excel determine the required sales.

**❓ Possible Questions and Answers:**

**1. What is Conditional Formatting in Excel?**

**Answer:**  
It’s a feature used to format cells automatically based on a condition, like highlighting marks below 35 in red.

**2. How do you apply conditional formatting?**

**Answer:**  
Select the cells → Home tab → Conditional Formatting → Choose a rule (like "Less than") → Set the format.

**3. What is a Pivot Table used for?**

**Answer:**  
A pivot table helps to summarize, analyze, and explore data, like total sales per product.

**4. How do you create a pivot table?**

**Answer:**  
Select the data → Insert tab → Pivot Table → Choose fields for rows, columns, and values.

**5. What is the VLOOKUP function?**

**Answer:**  
VLOOKUP searches a value in the first column of a table and returns a value in the same row from another column.

**6. What is the syntax of VLOOKUP?**

**Answer:**  
=VLOOKUP(lookup\_value, table\_array, col\_index\_num, [range\_lookup])

**7. What is Goal Seek in Excel?**

**Answer:**  
Goal Seek finds the input value needed to achieve a specific result from a formula.

**8. How do you use Goal Seek?**

**Answer:**  
Go to Data tab → What-If Analysis → Goal Seek → Set the cell (with formula), target value, and the cell to change.

**9. Can you use VLOOKUP for data in another worksheet?**

**Answer:**  
Yes, just include the worksheet name in the table\_array like Sheet2!A1:D10.

**10. Difference between VLOOKUP and HLOOKUP?**

**Answer:**  
VLOOKUP searches **vertically** (columns), HLOOKUP searches **horizontally** (rows).

**Practical 2**

**Data Frames and Basic Data Pre-processing**

* **Read data from CSV and JSON files into a data frame.**
* **Perform basic data pre-processing tasks such as handling missing values and outliers.**
* **Manipulate and transform data using functions like filtering, sorting, and grouping.**

**pip install pandas numpy matplotlib scikit-learn**

**CODE:**

# Importing necessary libraries

import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

from sklearn.datasets import load\_iris

# Load the Iris dataset from sklearn

iris = load\_iris()

df = pd.DataFrame(data=iris.data, columns=iris.feature\_names)

df['target'] = iris.target

# Save it to CSV and JSON to simulate reading from file

df.to\_csv('iris.csv', index=False)

df.to\_json('iris.json', orient='records')

# ------------------ 1. Read data from CSV and JSON ------------------

# Read from CSV

csv\_df = pd.read\_csv('iris.csv')

print("Data from CSV file:")

print(csv\_df.head())

# Read from JSON

json\_df = pd.read\_json('iris.json')

print("\nData from JSON file:")

print(json\_df.head())

# ------------------ 2. Basic Data Pre-processing ------------------

# Add some missing values for demonstration

csv\_df.loc[0:2, 'sepal length (cm)'] = np.nan

# Handle missing values (e.g., fill with mean)

csv\_df['sepal length (cm)'].fillna(csv\_df['sepal length (cm)'].mean(), inplace=True)

# Add an outlier manually

csv\_df.loc[10, 'petal length (cm)'] = 100

# Detect and cap/fix outliers using IQR

Q1 = csv\_df['petal length (cm)'].quantile(0.25)

Q3 = csv\_df['petal length (cm)'].quantile(0.75)

IQR = Q3 - Q1

lower\_limit = Q1 - 1.5 \* IQR

upper\_limit = Q3 + 1.5 \* IQR

# Replace outliers with upper/lower limits

csv\_df['petal length (cm)'] = np.where(

csv\_df['petal length (cm)'] > upper\_limit, upper\_limit,

np.where(csv\_df['petal length (cm)'] < lower\_limit, lower\_limit, csv\_df['petal length (cm)'])

)

# ------------------ 3. Manipulate and Transform Data ------------------

# Filtering: Get rows where petal width > 1.5

filtered\_df = csv\_df[csv\_df['petal width (cm)'] > 1.5]

print("\nFiltered Data (petal width > 1.5):")

print(filtered\_df.head())

# Sorting: Sort by sepal length

sorted\_df = csv\_df.sort\_values(by='sepal length (cm)', ascending=False)

print("\nSorted Data (by sepal length):")

print(sorted\_df.head())

# Grouping: Group by target and calculate mean of each group

grouped\_df = csv\_df.groupby('target').mean()

print("\nGrouped Data (mean by target):")

print(grouped\_df)

# ------------------ 4. Scatter Plot ------------------

# Scatter plot: Sepal length vs Petal length, colored by target

plt.figure(figsize=(8, 6))

scatter = plt.scatter(

csv\_df['sepal length (cm)'],

csv\_df['petal length (cm)'],

c=csv\_df['target'],

cmap='viridis',

edgecolor='k'

)

plt.title("Scatter Plot: Sepal Length vs Petal Length")

plt.xlabel("Sepal Length (cm)")

plt.ylabel("Petal Length (cm)")

plt.colorbar(scatter, label='Target Class')

plt.grid(True)

plt.tight\_layout()

plt.show()

**❓ Possible Viva Questions and Short Answers:**

**1. What is a DataFrame in Python?**

**Answer:**  
A DataFrame is a 2D data structure in pandas, like a table with rows and columns.

**2. How do you read a CSV file using pandas?**

**Answer:**  
df = pd.read\_csv('filename.csv')

**3. How do you read a JSON file into a DataFrame?**

**Answer:**  
df = pd.read\_json('filename.json')

**4. How do you check for missing values?**

**Answer:**  
df.isnull().sum() shows the count of missing values in each column.

**5. How can you handle missing values?**

**Answer:**  
Use:

* df.dropna() to remove rows with missing values.
* df.fillna(value) to replace them with a specific value.

**6. What is an outlier?**

**Answer:**  
An outlier is a data point that differs significantly from other observations.

**7. How can you detect outliers?**

**Answer:**  
Using statistical methods like:

* IQR (Interquartile Range)
* df.describe() to check min, max values
* Box plots

**8. How do you filter rows in a DataFrame?**

**Answer:**  
Example: df[df['age'] > 25] filters rows where age is above 25.

**9. How do you sort data in a DataFrame?**

**Answer:**  
df.sort\_values(by='column\_name')

**10. How do you group data?**

**Answer:**  
df.groupby('column\_name') to group data, often followed by .sum(), .mean(), etc.

**11. What does df.describe() do?**

**Answer:**  
It gives summary statistics like mean, count, min, max, etc., for numerical columns.

**12. What is Data Pre-processing?**

**Answer:**  
Data pre-processing is the process of cleaning and preparing raw data for analysis. It includes handling missing values, removing outliers, normalizing data, etc.

**13. What is Filtering in pandas?**

**Answer:**  
Filtering is selecting specific rows that meet a condition.  
**Example:** df[df['Age'] > 30] filters rows where Age > 30.

**14. What is Sorting in pandas?**

**Answer:**  
Sorting arranges rows based on one or more columns.  
**Example:** df.sort\_values(by='Name') sorts the DataFrame alphabetically by Name.

**15. What is Grouping in pandas?**

**Answer:**  
Grouping is used to split data into groups and perform operations like sum or average on them.  
**Example:** df.groupby('Department')['Salary'].mean() gives average salary per department.

**Practical 3**

**Feature Scaling and Dummification**

* **Apply feature-scaling techniques like standardization and normalization to numerical features.**
* **Perform feature dummification to convert categorical variables into numerical representations.**

**pip install pandas scikit-learn seaborn**

**CODE:**

# Practical 3: Feature Scaling and Dummification

# Step 1: Import necessary libraries

import pandas as pd

from sklearn.datasets import load\_iris

from sklearn.preprocessing import StandardScaler, MinMaxScaler

import seaborn as sns

# Step 2: Load the Iris dataset

iris = load\_iris()

df = pd.DataFrame(data=iris.data, columns=iris.feature\_names)

# Add the target (species) column as a categorical column

df['species'] = pd.Categorical.from\_codes(iris.target, iris.target\_names)

# Display the original data

print("Original Dataset:")

print(df.head())

# Step 3: Apply Feature Scaling

# a) Standardization (Z-score scaling)

standard\_scaler = StandardScaler()

df\_standardized = pd.DataFrame(standard\_scaler.fit\_transform(df.iloc[:, :-1]), columns=iris.feature\_names)

# b) Normalization (Min-Max scaling)

minmax\_scaler = MinMaxScaler()

df\_normalized = pd.DataFrame(minmax\_scaler.fit\_transform(df.iloc[:, :-1]), columns=iris.feature\_names)

print("\nStandardized Features:")

print(df\_standardized.head())

print("\nNormalized Features:")

print(df\_normalized.head())

# Step 4: Feature Dummification (One-hot encoding for categorical variable 'species')

df\_dummies = pd.get\_dummies(df['species'], prefix='species')

print("\nDummified Categorical Feature (species):")

print(df\_dummies.head())

# Step 5: Combine scaled features with dummified features

final\_standardized = pd.concat([df\_standardized, df\_dummies], axis=1)

final\_normalized = pd.concat([df\_normalized, df\_dummies], axis=1)

print("\nFinal Dataset with Standardized Features + Dummified Species:")

print(final\_standardized.head())

print("\nFinal Dataset with Normalized Features + Dummified Species:")

print(final\_normalized.head())

❓ **Viva Questions and Short Answers:**

**1. What is Feature Scaling?**

**Answer:**  
Feature scaling is the process of bringing all numerical features to the same scale so that no variable dominates others in a machine learning model.

**2. Why is Feature Scaling important?**

**Answer:**  
It improves model performance and accuracy, especially for algorithms like K-NN, SVM, and gradient descent-based models.

**3. What is Normalization?**

**Answer:**  
Normalization scales the data between 0 and 1 using this formula:  
(x - min) / (max - min)

**4. What is Standardization?**

**Answer:**  
Standardization transforms data to have a **mean = 0** and **standard deviation = 1** using:  
(x - mean) / std

**5. Which function is used for normalization in sklearn?**

**Answer:**  
MinMaxScaler() from sklearn.preprocessing

**6. Which function is used for standardization in sklearn?**

**Answer:**  
StandardScaler() from sklearn.preprocessing

**7. What is Dummification?**

**Answer:**  
Dummification converts **categorical variables** into **binary (0/1) format**, making them usable for ML algorithms.

**8. How do you perform dummification in pandas?**

**Answer:**  
Using pd.get\_dummies(df['column'])

**9. Why is dummification needed in machine learning?**

**Answer:**  
Because most ML algorithms require numeric input — they can't process strings like "Male" or "Female" directly.

**10. What happens if we don’t scale features?**

**Answer:**  
Models like KNN or SVM may give biased results due to uneven feature impact.

**11. Difference between Label Encoding and Dummification?**

**Answer:**

* **Label Encoding:** Converts categories into numbers (e.g., "Yes" → 1, "No" → 0).
* **Dummification (One-hot encoding):** Creates separate columns for each category.

**Practical 4**

**Hypothesis Testing**

* **Formulate null and alternative hypotheses for a given problem.**
* **Conduct a hypothesis test using appropriate statistical tests (e.g., t-test, chi square test).**
* **Interpret the results and draw conclusions based on the test outcomes.**

**pip install pandas scikit-learn scipy**

**CODE:**

# Practical 4: Hypothesis Testing using Iris Dataset

# Step 1: Import necessary libraries

import pandas as pd

from sklearn.datasets import load\_iris

from scipy.stats import ttest\_ind, chi2\_contingency

# Step 2: Load the Iris dataset

iris = load\_iris()

df = pd.DataFrame(data=iris.data, columns=iris.feature\_names)

df['species'] = pd.Categorical.from\_codes(iris.target, iris.target\_names)

# Display first few rows

print("Original Iris Dataset:")

print(df.head())

# Step 3: Formulate Hypotheses for a t-test

# H0: The mean petal length of Setosa and Versicolor are equal

# H1: The mean petal length of Setosa and Versicolor are different

# Extract petal lengths for Setosa and Versicolor

setosa\_petal = df[df['species'] == 'setosa']['petal length (cm)']

versicolor\_petal = df[df['species'] == 'versicolor']['petal length (cm)']

# Step 4: Perform Independent t-test

t\_stat, p\_val = ttest\_ind(setosa\_petal, versicolor\_petal)

print("\nT-Test: Comparing petal length of Setosa vs Versicolor")

print(f"T-statistic: {t\_stat:.4f}")

print(f"P-value: {p\_val:.4f}")

# Step 5: Interpret Results

alpha = 0.05

if p\_val < alpha:

print("Result: Reject the null hypothesis. Significant difference in petal lengths.")

else:

print("Result: Fail to reject the null hypothesis. No significant difference.")

# Step 6: Chi-square Test

# H0: Species and sepal length categories are independent

# H1: Species and sepal length categories are associated

# Create categorical bins for sepal length

df['sepal\_length\_cat'] = pd.cut(df['sepal length (cm)'], bins=3, labels=["Short", "Medium", "Long"])

# Create contingency table

contingency = pd.crosstab(df['species'], df['sepal\_length\_cat'])

# Perform Chi-Square Test

chi2, p\_chi, dof, expected = chi2\_contingency(contingency)

print("\nChi-Square Test: Independence between species and sepal length categories")

print(f"Chi-Square Statistic: {chi2:.4f}")

print(f"P-value: {p\_chi:.4f}")

if p\_chi < alpha:

print("Result: Reject the null hypothesis. Variables are dependent.")

else:

print("Result: Fail to reject the null hypothesis. Variables are independent.")

**❓ Common Viva Questions and Short Answers**

**1. What is Hypothesis Testing?**

**Answer:**  
Hypothesis testing is a statistical method to decide whether there is enough evidence to accept or reject a hypothesis about a population.

**2. What is a Null Hypothesis (H₀)?**

**Answer:**  
H₀ is a statement that there is **no effect** or **no difference**. It's the default assumption.

**3. What is an Alternative Hypothesis (H₁ or Ha)?**

**Answer:**  
H₁ is a statement that there **is an effect** or **a difference** in the population.

**4. What is a p-value?**

**Answer:**  
The p-value tells how likely the observed result is under the null hypothesis. A **low p-value (< α)** suggests strong evidence **against** H₀.

**5. What is the significance level (α)?**

**Answer:**  
α is the threshold for rejecting H₀, usually set at **0.05** (5%).

**6. What does it mean to reject the null hypothesis?**

**Answer:**  
It means there's **enough evidence** to support the alternative hypothesis.

**7. What is a t-test used for?**

**Answer:**  
A t-test is used to compare **means** between groups (e.g., comparing test scores of two classes).

**8. When do we use the chi-square test?**

**Answer:**  
The chi-square test is used to test relationships between **categorical variables**.

**9. What are the types of t-tests?**

**Answer:**

* **One-sample t-test**
* **Two-sample (independent) t-test**
* **Paired t-test**

**10. What are the steps in hypothesis testing?**

**Answer:**

1. State H₀ and H₁
2. Choose α (e.g., 0.05)
3. Choose test (t-test, chi-square, etc.)
4. Calculate test statistic and p-value
5. Compare p-value with α and draw conclusion

**11. What does "fail to reject the null hypothesis" mean?**

**Answer:**  
It means there is **not enough evidence** to support the alternative hypothesis.

**Practical 5**

**ANOVA (Analysis of Variance)**

* **Perform one-way ANOVA to compare means across multiple groups.**
* **Conduct post-hoc tests to identify significant differences between group means.**

**pip install pandas scikit-learn scipy statsmodels seaborn matplotlib**

**CODE:**

# Practical: ANOVA (Analysis of Variance) using Iris Dataset

# Step 1: Import necessary libraries

import pandas as pd

from sklearn.datasets import load\_iris

from scipy.stats import f\_oneway

import statsmodels.api as sm

from statsmodels.formula.api import ols

import seaborn as sns

import matplotlib.pyplot as plt

from statsmodels.stats.multicomp import pairwise\_tukeyhsd

# Step 2: Load the Iris dataset

iris = load\_iris()

df = pd.DataFrame(data=iris.data, columns=iris.feature\_names)

df['species'] = pd.Categorical.from\_codes(iris.target, iris.target\_names)

print("First 5 rows of Iris dataset:")

print(df.head())

# Step 3: One-way ANOVA to compare means of 'petal length (cm)' across species

# H0: All group means are equal

# H1: At least one group mean is different

# Extract groups

setosa = df[df['species'] == 'setosa']['petal length (cm)']

versicolor = df[df['species'] == 'versicolor']['petal length (cm)']

virginica = df[df['species'] == 'virginica']['petal length (cm)']

# Perform ANOVA

f\_stat, p\_value = f\_oneway(setosa, versicolor, virginica)

print("\nOne-way ANOVA Result:")

print(f"F-statistic: {f\_stat:.4f}")

print(f"P-value: {p\_value:.4f}")

# Step 4: Interpretation

alpha = 0.05

if p\_value < alpha:

print("Result: Reject the null hypothesis. Significant difference exists between group means.")

else:

print("Result: Fail to reject the null hypothesis. No significant difference between group means.")

# Step 5: Post-hoc Test (Tukey HSD)

# Tukey's test to identify which groups differ

tukey = pairwise\_tukeyhsd(endog=df['petal length (cm)'],

groups=df['species'],

alpha=0.05)

print("\nTukey HSD Post-hoc Test Results:")

print(tukey)

# Optional: Visualize the differences

sns.boxplot(x='species', y='petal length (cm)', data=df)

plt.title("Boxplot of Petal Length by Species")

plt.show()

**❓ Viva Questions and Short Answers**

**1. What is ANOVA?**

**Answer:**  
ANOVA (Analysis of Variance) is a statistical method used to compare the **means of three or more groups**.

**2. What is one-way ANOVA?**

**Answer:**  
One-way ANOVA tests the effect of **one independent variable** (factor) on a **dependent variable**.

**3. When do we use ANOVA?**

**Answer:**  
When we want to compare the means of **three or more** groups to see if they are significantly different.

**4. What is the null hypothesis (H₀) in ANOVA?**

**Answer:**  
H₀: All group means are **equal**.

**5. What is the alternative hypothesis (H₁) in ANOVA?**

**Answer:**  
H₁: At least one group mean is **different**.

**6. What is the F-statistic?**

**Answer:**  
The F-statistic is the ratio of **variance between groups** to **variance within groups**. A high F-value suggests significant differences.

**7. What is a p-value in ANOVA?**

**Answer:**  
The p-value tells whether the group means are statistically different. If **p < 0.05**, we reject H₀.

**8. What are the assumptions of ANOVA?**

**Answer:**

* Data is **normally distributed**
* **Equal variances** across groups
* Observations are **independent**

**9. What is a post-hoc test?**

**Answer:**  
Post-hoc tests (like Tukey’s HSD) are used after ANOVA to find **which specific group means** are different.

**10. Why do we use post-hoc tests?**

**Answer:**  
Because ANOVA only tells us **that** a difference exists, not **where** it exists.

**11. Can ANOVA be used for two groups?**

**Answer:**  
Technically yes, but it's better to use a **t-test** for two groups.

**12. What does it mean if ANOVA is significant?**

**Answer:**  
It means **at least one group mean** is significantly different from the others.

**Practical 6**

**Regression and Its Types**

* **Implement simple linear regression using a dataset.**
* **Explore and interpret the regression model coefficients and goodness-of-fit measures.**
* **Extend the analysis to multiple linear regression and assess the impact of additional predictors.**

**pip install pandas seaborn matplotlib scikit-learn numpy**

**CODE:**

# Importing Required Libraries

import pandas as pd

import seaborn as sns

import matplotlib.pyplot as plt

from sklearn.datasets import load\_iris

from sklearn.model\_selection import train\_test\_split

from sklearn.linear\_model import LinearRegression

from sklearn.preprocessing import OneHotEncoder

from sklearn.metrics import mean\_squared\_error, r2\_score

import numpy as np

# ===========================

# SIMPLE LINEAR REGRESSION

# ===========================

print("\n--- SIMPLE LINEAR REGRESSION ---\n")

# Step 1: Load the dataset

iris = load\_iris()

df = pd.DataFrame(iris.data, columns=iris.feature\_names)

df['species'] = pd.Categorical.from\_codes(iris.target, iris.target\_names)

# Step 2: Get dataset info

print("First 5 rows of dataset:")

print(df.head())

print("\nDataset description:")

print(df.describe())

# Step 3: Selecting Predictor and Target

X\_simple = df[['sepal length (cm)']] # Predictor

y\_simple = df['petal length (cm)'] # Target

# Step 4: Data Distribution Plots

plt.figure(figsize=(5, 4))

sns.histplot(X\_simple['sepal length (cm)'], kde=True)

plt.title("Histogram of Predictor (Sepal Length)")

plt.show()

plt.figure(figsize=(5, 4))

sns.histplot(y\_simple, kde=True)

plt.title("Histogram of Target (Petal Length)")

plt.show()

plt.figure(figsize=(6, 4))

sns.scatterplot(x='sepal length (cm)', y='petal length (cm)', data=df)

plt.title("Scatter Plot: Sepal Length vs Petal Length")

plt.show()

# Step 5: Split Data

X\_train\_s, X\_test\_s, y\_train\_s, y\_test\_s = train\_test\_split(X\_simple, y\_simple, test\_size=0.2, random\_state=0)

# Step 6: Train Model

model\_simple = LinearRegression()

model\_simple.fit(X\_train\_s, y\_train\_s)

# Step 7: Predict Results

y\_train\_pred\_s = model\_simple.predict(X\_train\_s)

y\_test\_pred\_s = model\_simple.predict(X\_test\_s)

# Step 8: Plot Training Predictions

plt.figure(figsize=(6, 4))

plt.scatter(X\_train\_s, y\_train\_s, color='blue', label='Actual')

plt.plot(X\_train\_s, y\_train\_pred\_s, color='red', label='Predicted')

plt.title("Training Set: Simple Linear Regression")

plt.legend()

plt.show()

# Step 9: Plot Test Predictions

plt.figure(figsize=(6, 4))

plt.scatter(X\_test\_s, y\_test\_s, color='green', label='Actual')

plt.plot(X\_test\_s, y\_test\_pred\_s, color='red', label='Predicted')

plt.title("Test Set: Simple Linear Regression")

plt.legend()

plt.show()

# Step 10: Print Equation

print("\nSimple Linear Regression Equation:")

print(f"Intercept: {model\_simple.intercept\_:.4f}")

print(f"Coefficient: {model\_simple.coef\_[0]:.4f}")

print(f"Equation: Petal Length = {model\_simple.intercept\_:.4f} + {model\_simple.coef\_[0]:.4f} \* Sepal Length")

# ===========================

# MULTIPLE LINEAR REGRESSION

# ===========================

print("\n\n--- MULTIPLE LINEAR REGRESSION ---\n")

# Step 1: Prepare data for Multiple Regression

# We'll predict petal length using all 4 numeric features

X\_multi = df.drop(columns=['species', 'petal length (cm)']) # predictors

y\_multi = df['petal length (cm)'] # target

# Step 2: Relationship Between Predictor and Target (using scatterplot with regression line)

plt.figure(figsize=(6, 4))

sns.regplot(x='petal width (cm)', y='petal length (cm)', data=df)

plt.title("Petal Width vs Petal Length (with Regression Line)")

plt.show()

# Step 3: Split Data (80%-20%)

X\_train\_m, X\_test\_m, y\_train\_m, y\_test\_m = train\_test\_split(X\_multi, y\_multi, test\_size=0.2, random\_state=42)

# Step 4: Train the model

model\_multi = LinearRegression()

model\_multi.fit(X\_train\_m, y\_train\_m)

# Step 5: Predict

y\_pred\_m = model\_multi.predict(X\_test\_m)

# Step 6: Compare Predictions with Actual

comparison\_df = pd.DataFrame({'Actual': y\_test\_m.values, 'Predicted': y\_pred\_m})

print("\nComparison of Actual vs Predicted (Multiple Regression):")

print(comparison\_df.head())

# Step 7: Print Regression Coefficients

print("\nMultiple Linear Regression Equation:")

print(f"Intercept: {model\_multi.intercept\_:.4f}")

for feature, coef in zip(X\_multi.columns, model\_multi.coef\_):

print(f"Coefficient for {feature}: {coef:.4f}")

# Step 8: Evaluate Model

print("\nModel Evaluation:")

print(f"R² Score: {r2\_score(y\_test\_m, y\_pred\_m):.4f}")

print(f"Mean Squared Error: {mean\_squared\_error(y\_test\_m, y\_pred\_m):.4f}")

**❓ Common Viva Questions and Short Answers**

**1. What is regression?**

**Answer:**  
Regression is a statistical method used to **model the relationship** between a dependent variable and one or more independent variables.

**2. What is simple linear regression?**

**Answer:**  
Simple linear regression models the relationship between **one independent variable** and **one dependent variable**.

**3. What is multiple linear regression?**

**Answer:**  
Multiple linear regression uses **two or more independent variables** to predict the dependent variable.

**4. What is the regression equation?**

**Answer:**  
For simple linear regression:  
**Y = b₀ + b₁X + ε**  
Where:

* Y = dependent variable
* X = independent variable
* b₀ = intercept
* b₁ = slope
* ε = error term

**5. What do regression coefficients represent?**

**Answer:**  
They show the **impact of each independent variable** on the dependent variable.

**6. What is R-squared (R²)?**

**Answer:**  
R² measures how well the model explains the variation in the dependent variable.  
**Higher R² = better fit**.

**7. What is a residual?**

**Answer:**  
Residual = **Actual value – Predicted value**. It shows the error in prediction.

**8. What are the assumptions of linear regression?**

**Answer:**

1. Linearity
2. Independence
3. Homoscedasticity (equal variance of residuals)
4. Normality of residuals
5. No multicollinearity (in multiple regression)

**9. What is multicollinearity?**

**Answer:**  
Multicollinearity occurs when **independent variables are highly correlated** with each other. It can affect model accuracy.

**10. How do you check model performance?**

**Answer:**  
Using **R²**, **Adjusted R²**, **Mean Squared Error (MSE)**, and **residual plots**.

**11. What happens when you add more predictors in multiple regression?**

**Answer:**  
The model may become more accurate, but **overfitting** can occur if too many irrelevant predictors are added.

**12. Can regression be used for classification?**

**Answer:**  
No, regression is used for predicting **continuous values**. For classification, we use algorithms like logistic regression, decision trees, etc.

**Practical 7**

**Logistic Regression and Decision Tree**

* **Build a logistic regression model to predict a binary outcome.**
* **Evaluate the model's performance using classification metrics (e.g., accuracy, precision, recall).**
* **Construct a decision tree model and interpret the decision rules for classification.**

**pip install pandas scikit-learn matplotlib seaborn**

**CODE:**

# Logistic Regression

# ========================================

# Import necessary libraries

import pandas as pd

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 (

confusion\_matrix, accuracy\_score, precision\_score,

recall\_score, f1\_score, roc\_curve, auc

)

import seaborn as sns

import matplotlib.pyplot as plt

# Load the dataset

data = load\_breast\_cancer()

# Create a DataFrame

df = pd.DataFrame(data.data, columns=data.feature\_names)

df['target'] = data.target

# Split into features and target

X = df.drop(columns=['target'])

y = df['target']

# Train-test split

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=42)

# Logistic Regression Model

model = LogisticRegression(max\_iter=5000)

model.fit(X\_train, y\_train)

y\_pred = model.predict(X\_test)

# Confusion Matrix

cm = confusion\_matrix(y\_test, y\_pred)

plt.figure(figsize=(6,5))

sns.heatmap(cm, annot=True, fmt='d', cmap='Blues', xticklabels=['Negative', 'Positive'], yticklabels=['Negative', 'Positive'])

plt.xlabel('Predicted')

plt.ylabel('Actual')

plt.title('Confusion Matrix Heatmap')

plt.show()

# Evaluation Metrics

accuracy = accuracy\_score(y\_test, y\_pred)

precision = precision\_score(y\_test, y\_pred)

recall = recall\_score(y\_test, y\_pred)

f1 = f1\_score(y\_test, y\_pred)

print("\n--- Logistic Regression Metrics ---")

print(f'Accuracy: {accuracy:.4f}')

print(f'Precision: {precision:.4f}')

print(f'Recall: {recall:.4f}')

print(f'F1 Score: {f1:.4f}')

# ROC Curve

y\_prob = model.predict\_proba(X\_test)[:, 1]

fpr, tpr, \_ = roc\_curve(y\_test, y\_prob)

roc\_auc = auc(fpr, tpr)

plt.figure(figsize=(6,5))

plt.plot(fpr, tpr, color='blue', label=f'ROC Curve (AUC = {roc\_auc:.4f})')

plt.plot([0, 1], [0, 1], color='red', linestyle='--')

plt.xlabel('False Positive Rate')

plt.ylabel('True Positive Rate')

plt.title('Receiver Operating Characteristic (ROC) Curve')

plt.legend()

plt.show()

# Decision Tree Classifier

# ========================================

from sklearn.tree import DecisionTreeClassifier, plot\_tree

from sklearn.metrics import classification\_report

# Check for missing values

df.isnull().sum()

df.fillna(df.mean(), inplace=True)

# Split data

X = df.drop(columns=['target'])

y = df['target']

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=42)

# Initialize and train decision tree

dt\_model = DecisionTreeClassifier(random\_state=42)

dt\_model.fit(X\_train, y\_train)

# Evaluate on test data

y\_pred\_test = dt\_model.predict(X\_test)

test\_accuracy = accuracy\_score(y\_test, y\_pred\_test)

print(f'\nDecision Tree - Accuracy on Test Data: {test\_accuracy:.4f}')

# Evaluate on training data

y\_pred\_train = dt\_model.predict(X\_train)

train\_accuracy = accuracy\_score(y\_train, y\_pred\_train)

print(f'Decision Tree - Accuracy on Training Data: {train\_accuracy:.4f}')

# Plot the decision tree

plt.figure(figsize=(15,10))

plot\_tree(dt\_model, feature\_names=X.columns, class\_names=['Malignant', 'Benign'], filled=True, rounded=True)

plt.title("Decision Tree - Full Tree")

plt.show()

# Classification Reports

print("\n--- Decision Tree Classification Report (Test Data) ---")

print(classification\_report(y\_test, y\_pred\_test))

print("--- Classification Report (Training Data) ---")

print(classification\_report(y\_train, y\_pred\_train))

# ROC Curve - Decision Tree

y\_prob\_tree = dt\_model.predict\_proba(X\_test)[:, 1]

fpr\_tree, tpr\_tree, \_ = roc\_curve(y\_test, y\_prob\_tree)

roc\_auc\_tree = auc(fpr\_tree, tpr\_tree)

plt.figure(figsize=(6,5))

plt.plot(fpr\_tree, tpr\_tree, color='green', label=f'ROC Curve (AUC = {roc\_auc\_tree:.4f})')

plt.plot([0, 1], [0, 1], color='red', linestyle='--')

plt.xlabel('False Positive Rate')

plt.ylabel('True Positive Rate')

plt.title('ROC Curve for Decision Tree')

plt.legend()

plt.show()

# Cost Complexity Pruning

path = dt\_model.cost\_complexity\_pruning\_path(X\_train, y\_train)

ccp\_alphas, impurities = path.ccp\_alphas, path.impurities

plt.figure(figsize=(6,4))

plt.plot(ccp\_alphas[:-1], impurities[:-1], marker="o", drawstyle="steps-post")

plt.xlabel("Alpha")

plt.ylabel("Total Impurity of Leaves")

plt.title("Alpha vs Impurity (Pruning Path)")

plt.show()

# Select optimal alpha and prune

optimal\_alpha = ccp\_alphas[-2] # Choose second last to avoid pruning everything

pruned\_tree = DecisionTreeClassifier(random\_state=42, ccp\_alpha=optimal\_alpha)

pruned\_tree.fit(X\_train, y\_train)

# Plot pruned tree

plt.figure(figsize=(15,10))

plot\_tree(pruned\_tree, feature\_names=X.columns, class\_names=['Malignant', 'Benign'], filled=True, rounded=True)

plt.title("Decision Tree - Pruned Tree")

plt.show()

**❓ Common Viva Questions and Short Answers**

**1. What is logistic regression?**

**Answer:**  
Logistic regression is a classification algorithm used to **predict binary outcomes** (like Yes/No, 0/1).

**2. What is the equation for logistic regression?**

**Answer:**  
It uses the **sigmoid function**:  
**P = 1 / (1 + e^–(b₀ + b₁X))**

**3. What is the sigmoid function?**

**Answer:**  
The sigmoid function maps any real number to a value between **0 and 1**, useful for **probability prediction**.

**4. What is the difference between linear and logistic regression?**

**Answer:**

* **Linear** regression predicts **continuous** values.
* **Logistic** regression predicts **probabilities of classes** (usually binary).

**5. What are common classification metrics?**

**Answer:**

* **Accuracy**: The percentage of **correct predictions** made by the model out of all predictions. [Correct predictions / Total predictions]
* **Precision**: Out of all predicted **positive cases**, how many were **actually positive**. [TP / (TP + FP)]
* **Recall**: Out of all **actual positive cases**, how many were **correctly predicted**. [TP / (TP + FN)]
* **F1-score**: Harmonic mean of precision and recall

**6. What is a confusion matrix?**

**Answer:**  
A table showing **TP, FP, FN, TN** to evaluate classification performance.

**7. What is a decision tree?**

**Answer:**  
A decision tree is a model that splits data into **branches** using conditions, forming a **tree-like structure** to classify data.

**8. What is entropy in decision trees?**

**Answer:**  
Entropy measures **impurity** in a dataset.  
Lower entropy = more pure group.

**9. What is information gain?**

**Answer:**  
It measures the **reduction in entropy** after a dataset is split.  
Used to decide which feature to split on.

**10. What is Gini index?**

**Answer:**  
Another metric for measuring impurity.  
Lower Gini = better split. Used in CART (Classification and Regression Tree).

**11. What are the advantages of decision trees?**

**Answer:**

* Easy to understand
* No need for feature scaling
* Works for both classification and regression

**12. What are the disadvantages of decision trees?**

**Answer:**

* Can **overfit**
* Sensitive to small changes in data
* May not perform well with **imbalanced data**

**13. What is ROC Curve?**

**Answer:**

The **ROC curve** shows the trade-off between **true positive rate** and **false positive rate** for a classification model.

**14. What is Alpha vs Impurity?**

**Answer:**

**Alpha** controls tree complexity through regularization, while **impurity** measures the heterogeneity of data within a node, with lower impurity indicating purer nodes.

**15. What is Pruned decision tree?**

**Answer:**

A **pruned decision tree** is a simplified tree with unnecessary branches removed to reduce overfitting.

**Practical 8**

**K-Means Clustering**

* **Apply the K-Means algorithm to group similar data points into clusters.**
* **Determine the optimal number of clusters using elbow method or silhouette analysis.**
* **Visualize the clustering results and analyze the cluster characteristics.**

**pip install pandas numpy matplotlib seaborn scikit-learn**

**CODE:**

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

from sklearn.decomposition import PCA

import seaborn as sns

# Load the Iris dataset

iris = load\_iris()

data = pd.DataFrame(iris.data, columns=iris.feature\_names)

# Step 1: Elbow Method to determine optimal k

wcss = [] # within-cluster sum of squares

K\_range = range(1, 11)

for k in K\_range:

kmeans = KMeans(n\_clusters=k, random\_state=42)

kmeans.fit(data)

wcss.append(kmeans.inertia\_)

# Plot the Elbow curve

plt.figure(figsize=(8, 5))

plt.plot(K\_range, wcss, 'bo-')

plt.xlabel('Number of clusters (k)')

plt.ylabel('WCSS')

plt.title('Elbow Method for Optimal k')

plt.grid(True)

plt.show()

# Step 2: Silhouette Scores

silhouette\_scores = []

for k in range(2, 11):

kmeans = KMeans(n\_clusters=k, random\_state=42)

labels = kmeans.fit\_predict(data)

score = silhouette\_score(data, labels)

silhouette\_scores.append(score)

# Plot Silhouette scores

plt.figure(figsize=(8, 5))

plt.plot(range(2, 11), silhouette\_scores, 'go-')

plt.xlabel('Number of clusters (k)')

plt.ylabel('Silhouette Score')

plt.title('Silhouette Analysis for Optimal k')

plt.grid(True)

plt.show()

# Choose optimal k (visually from Elbow or highest Silhouette Score)

optimal\_k = 3

# Step 3: Apply KMeans with optimal\_k

kmeans\_final = KMeans(n\_clusters=optimal\_k, random\_state=42)

cluster\_labels = kmeans\_final.fit\_predict(data)

# Add cluster labels to the DataFrame

data['Cluster'] = cluster\_labels

# Step 4: Reduce to 2D using PCA for visualization

pca = PCA(n\_components=2)

data\_pca = pca.fit\_transform(data.drop('Cluster', axis=1))

# Create a DataFrame for plotting

plot\_data = pd.DataFrame(data\_pca, columns=['PCA1', 'PCA2'])

plot\_data['Cluster'] = cluster\_labels

# Visualize Clusters

plt.figure(figsize=(8, 6))

sns.scatterplot(data=plot\_data, x='PCA1', y='PCA2', hue='Cluster', palette='Set1', s=80)

plt.title(f'K-Means Clustering with k={optimal\_k}')

plt.grid(True)

plt.show()

# Step 5: Analyze Cluster Characteristics

cluster\_centers = pd.DataFrame(kmeans\_final.cluster\_centers\_, columns=iris.feature\_names)

print("\nCluster Centers:")

print(cluster\_centers)

# Summary of how many points per cluster

print("\nNumber of points per cluster:")

print(data['Cluster'].value\_counts())

**❓ Common Viva Questions and Short Answers**

**1. What is K-Means Clustering?**

**Answer:**  
K-Means is an **unsupervised learning algorithm** that groups data into **K clusters** based on feature similarity.

**2. How does K-Means work?**

**Answer:**

1. Choose number of clusters (K)
2. Randomly place centroids
3. Assign points to the nearest centroid
4. Recalculate centroids
5. Repeat until centroids stabilize

**3. What is a centroid?**

**Answer:**  
A centroid is the **center point** of a cluster, calculated as the **mean of all points** in that cluster.

**4. How do you choose the value of K?**

**Answer:**  
Using the **Elbow Method** or **Silhouette Score**.

**5. What is the Elbow Method?**

**Answer:**  
A plot of **K vs. Within-Cluster Sum of Squares (WCSS)**. The "elbow" point where the curve bends is the **optimal K**.

**6. What is Silhouette Score?**

**Answer:**  
A score between -1 and 1 that measures how similar an object is to its own cluster vs. other clusters.  
**Higher score = better clustering**.

**7. What type of learning is K-Means?**

**Answer:**  
**Unsupervised learning**, since it finds patterns without labeled data.

**8. What are the limitations of K-Means?**

**Answer:**

* Sensitive to **initial centroid placement**
* Struggles with **non-spherical** clusters
* Must predefine **K**

**9. Can K-Means work with categorical data?**

**Answer:**  
No, it works best with **numerical** data. For categorical data, use algorithms like **K-modes**.

**10. How do you visualize clusters?**

**Answer:**  
Using **scatter plots** (with color-coded clusters), **PCA** for dimensionality reduction, or **TSNE** for complex data.

**Practical 9**

**Principal Component Analysis (PCA)**

* **Perform PCA on a dataset to reduce dimensionality.**
* **Evaluate the explained variance and select the appropriate number of principal components.**
* **Visualize the data in the reduced-dimensional space.**

**pip install pandas numpy matplotlib seaborn scikit-learn**

**CODE:**

import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

from sklearn.datasets import load\_iris

from sklearn.preprocessing import StandardScaler

from sklearn.decomposition import PCA

import seaborn as sns

# Load the Iris dataset

iris = load\_iris()

X = iris.data

y = iris.target

feature\_names = iris.feature\_names

target\_names = iris.target\_names

# Step 1: Standardize the data

scaler = StandardScaler()

X\_scaled = scaler.fit\_transform(X)

# Step 2: Apply PCA

pca = PCA()

X\_pca = pca.fit\_transform(X\_scaled)

# Step 3: Explained Variance Ratio

explained\_variance = pca.explained\_variance\_ratio\_

cumulative\_variance = np.cumsum(explained\_variance)

# Plot explained variance

plt.figure(figsize=(8, 5))

plt.plot(range(1, len(explained\_variance)+1), cumulative\_variance, marker='o', linestyle='--')

plt.title('Explained Variance by Principal Components')

plt.xlabel('Number of Principal Components')

plt.ylabel('Cumulative Explained Variance')

plt.grid(True)

plt.axhline(y=0.95, color='r', linestyle='-')

plt.axhline(y=0.90, color='g', linestyle='--')

plt.show()

# Step 4: Reduce data to 2 principal components for visualization

pca\_2 = PCA(n\_components=2)

X\_reduced = pca\_2.fit\_transform(X\_scaled)

# Create DataFrame for plotting

pca\_df = pd.DataFrame(data=X\_reduced, columns=['PC1', 'PC2'])

pca\_df['Target'] = y

# Visualize the reduced data

plt.figure(figsize=(8, 6))

sns.scatterplot(data=pca\_df, x='PC1', y='PC2', hue='Target', palette='Set1', s=80)

plt.title('PCA of Iris Dataset (2D View)')

plt.grid(True)

plt.show()

# Print explained variance ratios

print("\nExplained Variance Ratio of Each Component:")

for i, ev in enumerate(explained\_variance):

print(f"PC{i+1}: {ev:.4f}")

# Decide on number of components to retain

for i, cv in enumerate(cumulative\_variance):

if cv >= 0.95:

print(f"\nWe can retain {i+1} components to explain at least 95% of variance.")

break

**❓ Common Viva Questions and Short Answers**

**1. What is PCA?**

**Answer:**  
PCA (Principal Component Analysis) is a **dimensionality reduction technique** that transforms data into new features (principal components) that capture the **maximum variance**.

**2. Why is PCA used?**

**Answer:**  
To **reduce the number of features** in a dataset while keeping as much **important information (variance)** as possible.

**3. What are principal components?**

**Answer:**  
They are **new features** formed by **linear combinations** of original features, ranked by how much variance they capture.

**4. What is explained variance?**

**Answer:**  
It tells how much **information (variance)** is captured by each principal component.

**5. How do you decide the number of components to keep?**

**Answer:**  
By checking the **explained variance ratio** or using a **scree plot** (look for the "elbow").

**6. What is a scree plot?**

**Answer:**  
A plot of principal components vs. their explained variance. The point where the curve starts to level off is called the **elbow**.

**7. Does PCA require feature scaling?**

**Answer:**  
Yes, especially when features have different units. **Standardization** is recommended before applying PCA.

**8. What are eigenvectors and eigenvalues in PCA?**

**Answer:**

* **Eigenvectors** define the direction of new axes (principal components).
* **Eigenvalues** define the magnitude (amount of variance) in those directions.

**9. Is PCA supervised or unsupervised?**

**Answer:**  
PCA is an **unsupervised** learning technique.

**10. Can PCA be used for visualization?**

**Answer:**  
Yes, PCA reduces data to **2 or 3 dimensions**, making it easier to **visualize high-dimensional datasets**.

**Practical 10**

**Data Visualization and Storytelling**

* **Create meaningful visualizations using data visualization tools**
* **Combine multiple visualizations to tell a compelling data story.**
* **Present the findings and insights in a clear and concise manner.**

**pip install wordcloud**

**pip install squarify**

**CODE:**

# Import libraries

import pandas as pd

import matplotlib.pyplot as plt

import seaborn as sns

import plotly.express as px

from wordcloud import WordCloud

import squarify

import numpy as np

# Load Titanic dataset

df = sns.load\_dataset('titanic')

# Drop rows with too many missing values

df.dropna(subset=['age', 'embarked', 'fare'], inplace=True)

# Set style

sns.set(style="whitegrid")

# 1. Pie Chart - Survival Distribution

survived\_counts = df['survived'].value\_counts()

plt.figure(figsize=(6, 6))

plt.pie(survived\_counts, labels=['Not Survived', 'Survived'], autopct='%1.1f%%', colors=['red', 'green'])

plt.title('Survival Distribution (Pie Chart)')

plt.show()

# 2. Donut Chart - Gender Distribution

gender\_counts = df['sex'].value\_counts()

plt.figure(figsize=(6, 6))

plt.pie(gender\_counts, labels=gender\_counts.index, autopct='%1.1f%%', colors=['skyblue', 'lightpink'], wedgeprops={'width':0.4})

plt.title('Gender Distribution (Donut Chart)')

plt.show()

# 3. Histogram - Age Distribution

plt.figure(figsize=(8, 5))

sns.histplot(df['age'], bins=30, kde=True)

plt.title('Age Distribution (Histogram)')

plt.xlabel('Age')

plt.ylabel('Count')

plt.show()

# 4. Bar Chart - Class vs Survival

plt.figure(figsize=(8, 5))

sns.countplot(x='class', hue='survived', data=df, palette='Set2')

plt.title('Class vs Survival (Bar Chart)')

plt.show()

# 5. Scatter Plot - Age vs Fare

plt.figure(figsize=(8, 5))

sns.scatterplot(x='age', y='fare', hue='survived', data=df)

plt.title('Age vs Fare (Scatter Plot)')

plt.show()

# 6. Box Plot - Age distribution by Pclass

plt.figure(figsize=(8, 5))

sns.boxplot(x='pclass', y='age', data=df)

plt.title('Box Plot of Age by Passenger Class')

plt.xlabel('Passenger Class')

plt.ylabel('Age')

plt.show()

# 7. Violin Plot - Age and Gender

plt.figure(figsize=(8, 5))

sns.violinplot(x='sex', y='age', data=df, hue='sex', palette="muted", legend=False)

plt.title('Violin Plot of Age by Gender')

plt.show()

# 8. Treemap - Pclass Distribution

pclass\_counts = df['pclass'].value\_counts()

labels = [f'Class {i}\nCount: {count}' for i, count in zip(pclass\_counts.index, pclass\_counts.values)]

plt.figure(figsize=(8, 5))

squarify.plot(sizes=pclass\_counts.values, label=labels, color=['lightcoral', 'lightgreen', 'lightskyblue'], alpha=.8)

plt.title('Treemap of Passenger Class Distribution')

plt.axis('off')

plt.show()

# 9. Word Cloud - Common roles

text = ' '.join(df['who'].dropna().astype(str))

wordcloud = WordCloud(width=800, height=400, background\_color='white').generate(text)

plt.figure(figsize=(10, 5))

plt.imshow(wordcloud, interpolation='bilinear')

plt.axis('off')

plt.title('Word Cloud of Passenger Roles (man/woman/child)')

plt.show()

# 10. Line Plot with Annotation - Fare by Class (Average)

avg\_fare\_by\_class = df.groupby('pclass')['fare'].mean()

plt.figure(figsize=(8, 5))

sns.lineplot(x=avg\_fare\_by\_class.index, y=avg\_fare\_by\_class.values, marker='o')

for i, v in enumerate(avg\_fare\_by\_class.values):

plt.text(avg\_fare\_by\_class.index[i], v + 2, f"${v:.1f}", ha='center', fontweight='bold')

plt.title('Average Fare by Passenger Class (Line Plot with Annotation)')

plt.xlabel('Passenger Class')

plt.ylabel('Average Fare')

plt.grid(True)

plt.show()

# 11. Correlation Heatmap

corr = df[['age', 'fare', 'survived', 'pclass']].corr()

plt.figure(figsize=(8, 5))

sns.heatmap(corr, annot=True, cmap='coolwarm', linewidths=0.5)

plt.title('Correlation Heatmap of Numerical Features')

plt.show()

# 12. Time Series-style Graph - Age Group over Passenger Count

df['age\_group'] = pd.cut(df['age'], bins=np.arange(0, 90, 10), right=False)

age\_group\_counts = df['age\_group'].value\_counts().sort\_index()

plt.figure(figsize=(10, 5))

age\_group\_counts.plot(kind='line', marker='o', color='purple')

plt.title('Passenger Count by Age Group (Time Series Style)')

plt.xlabel('Age Group')

plt.ylabel('Number of Passengers')

plt.grid(True)

plt.xticks(rotation=45)

plt.show()

# Summary of all visualizations

print("\n--- Summary of Visual Insights ---")

print("1. Pie Chart: Roughly 62% did not survive, showing the tragic outcome of the disaster.")

print("2. Donut Chart: About 65% of passengers were male, indicating a gender imbalance.")

print("3. Histogram: Most passengers were aged 20–40, with a peak around age 25.")

print("4. Bar Chart: First-class passengers had a noticeably higher survival rate.")

print("5. Scatter Plot: Survivors tended to pay higher fares and be younger.")

print("6. Box Plot: First-class passengers were generally older than those in lower classes.")

print("7. Violin Plot: Males had a wider spread of ages, while females clustered around younger ages.")

print("8. Treemap: Most passengers traveled in 3rd class, followed by 1st and 2nd.")

print("9. Word Cloud: Majority of the passengers were 'man', followed by 'woman' and 'child'.")

print("10. Line Plot: Average fare increased with class — 1st class had the highest fare.")

print("11. Correlation Heatmap: Strong negative correlation between fare and pclass (lower class number = higher fare).")

print("12. Time Series Graph: Most passengers were between 20–30 years old.")

print("----------------------------------")

**❓ Common Viva Questions and Short Answers**

**1. What is data visualization?**

**Answer:**  
It is the graphical representation of data to **communicate information clearly and effectively**.

**2. Why is data visualization important?**

**Answer:**  
It helps to **understand trends, patterns, and outliers** in data and makes insights easier to grasp.

**3. What are common types of charts used?**

**Answer:**  
Pie chart, donut chart, histogram, bar chart, scatter plot, box plot, violin plot, treemap, word cloud, line plot, correlation heatmap, time series graph.

**4. When do you use a bar chart?**

**Answer:**  
To compare **categories** or groups.

**5. When do you use a line chart?**

**Answer:**  
To show **trends over time**.

**6. What is a histogram?**

**Answer:**  
It shows the **distribution** of a single variable by grouping values into bins.

**7. What is data storytelling?**

**Answer:**  
Combining **visuals and narrative** to explain the data in a **clear and compelling way**.

**8. What tools can be used for data visualization?**

**Answer:**  
Excel, Tableau, Power BI, Python (Matplotlib, Seaborn), R (ggplot2).

**9. What makes a good visualization?**

**Answer:**

* **Clear labels and titles**
* **Correct chart type**
* **Highlights key insights**
* **Not cluttered**

**10. What is a dashboard?**

**Answer:**  
A dashboard is a **collection of interactive visualizations** that provide an overview of data insights.

**✅ 1. Pie Chart**

**Q:** What is a pie chart and when is it used?  
**A:** A pie chart shows the **proportions** of categories as slices of a circle. Best for **comparing parts of a whole**.

**✅ 2. Donut Chart**

**Q:** How is a donut chart different from a pie chart?  
**A:** A donut chart is like a pie chart but with a **hole in the center**. It also shows proportions but allows for **extra labeling in the center**.

**✅ 3. Histogram**

**Q:** What is a histogram used for?  
**A:** A histogram displays the **distribution of a numeric variable** by dividing it into intervals (bins).

**✅ 4. Bar Chart**

**Q:** When should you use a bar chart?  
**A:** A bar chart is used to **compare values across categories**. It uses **bars of equal width**.

**✅ 5. Scatter Plot**

**Q:** What does a scatter plot show?  
**A:** It shows the **relationship between two numeric variables** using dots. Useful for detecting **correlations or trends**.

**✅ 6. Box Plot**

**Q:** What does a box plot display?  
**A:** A box plot shows the **spread and skewness of data**, including the **median, quartiles, and outliers**.

**✅ 7. Violin Plot**

**Q:** What is a violin plot?  
**A:** A violin plot combines a **box plot** with a **density plot** to show **distribution, median, and variability** of data.

**✅ 8. Treemap**

**Q:** What is a treemap used for?  
**A:** It shows **hierarchical data** as nested rectangles, where size represents a **numerical value**.

**✅ 9. Word Cloud**

**Q:** What is a word cloud and where is it used?  
**A:** A word cloud displays **text data**, where the **size of each word** indicates its **frequency**. Used in text analysis.

**✅ 10. Line Plot**

**Q:** What is a line plot used for?  
**A:** A line plot shows **trends over time** or sequences using connected data points.

**✅ 11. Correlation Heatmap**

**Q:** What does a correlation heatmap show?  
**A:** It shows the **correlation coefficients** between variables using color gradients (positive or negative relationships).

**✅ 12. Time Series Graph**

**Q:** What is a time series graph?  
**A:** It displays data points **collected or recorded over time**, often used to show **trends, patterns, or seasonality**.