DATA SCIENCE LAB EXPERIMENT 1

Name: Laukik Padgaonkar Div: D15C Roll No: 37

AIM: Introduction to Data science and Data preparation using Pandas steps.

THEORY:
Pandas: Pandas is an open-source Python library used for data manipulation and analysis. It provides high-performance data structures and functions for efficiently handling structured data.

Key Pandas Functions for Data Cleaning

1. Handling Missing Data
    - df.isnull().sum() → Check the number of missing values in each column.
    - df.dropna() → Remove rows with missing values.
    - df.fillna(value, inplace=True) → Fill missing values with a specific value (e.g., mean or median).
2. Removing Duplicates
    - df.duplicated() → Identify duplicate rows.
    - df.drop_duplicates(inplace=True) → Remove duplicate rows.
3. Handling Incorrect Data Formats
    - df['column'] = pd.to_datetime(df['column']) → Convert a column to a datetime format.
    - df['column'] = df['column'].astype(int/float/str) → Change data types.

Topic: Bengaluru Housing Prices

DATA SCIENCE LAB EXPERIMENT 1

1. Loading Data in Pandas:

aids1.py > ...

```
1  import pandas as pd
2
3
4  data = pd.read_excel('Bengaluru_House_Data.xlsx') # Load the dataset
5  print(data.head())
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS COMMENTS powershell

```
  File "parsers.pyx", line 891, in pandas._libs.parsers.TextReader._check_tokenize_status
  File "parsers.pyx", line 2053, in pandas._libs.parsers.raise_parser_error
UnicodeDecodeError: 'utf-8' codec can't decode byte 0xb0 in position 10: invalid start byte
PS C:\Users\lauki\OneDrive\Desktop\dataset> python aids1.py
PS C:\Users\lauki\OneDrive\Desktop\dataset> python aids1.py
              area_type         availability                  location  ... bath balcony   price
0  Super built-up  Area  2025-12-19 00:00:00  Electronic City Phase II  ...  2.0     1.0   39.07
1            Plot  Area        Ready To Move          Chikka Tirupathi  ...  5.0     3.0  120.00
2        Built-up  Area        Ready To Move               Uttarahalli  ...  2.0     3.0   62.00
3  Super built-up  Area        Ready To Move        Lingadheeranahalli  ...  3.0     1.0   95.00
4  Super built-up  Area        Ready To Move                  Kothanur  ...  2.0     1.0   51.00
```

2. Description of the Dataset:

```
print(data.head())
print(data.describe())
```

```
[5 rows x 9 columns]
                bath       balcony         price
count  13247.000000  12711.000000  13320.000000
mean       2.692610      1.584376    112.565627
std        1.341458      0.817263    148.971674
min        1.000000      0.000000      8.000000
25%        2.000000      1.000000     50.000000
50%        2.000000      2.000000     72.000000
75%        3.000000      2.000000    120.000000
max       40.000000      3.000000   3600.000000
```

3. Drop columns that are not useful:

```
1   import pandas as pd
2
3
4   data = pd.read_excel('Bengaluru_House_Data.xlsx')
5   data = data.drop(columns=['bath'])
6   print(data.head())
7
```

ROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS COMMENTS powershell

```
5%       2.000000      1.000000     50.000000
0%       2.000000      2.000000     72.000000
5%       3.000000      2.000000    120.000000
ax      40.000000      3.000000   3600.000000
S C:\Users\lauki\OneDrive\Desktop\dataset> python aids1.py
             area_type        availability                  location  ... total_sqft balcony   price
  Super built-up  Area  2025-12-19 00:00:00  Electronic City Phase II  ...       1056     1.0   39.07
            Plot  Area        Ready To Move          Chikka Tirupathi  ...       2600     3.0  120.00
       Built-up  Area        Ready To Move               Uttarahalli  ...       1440     3.0   62.00
  Super built-up  Area        Ready To Move        Lingadheeranahalli  ...       1521     1.0   95.00
  Super built-up  Area        Ready To Move                  Kothanur  ...       1200     1.0   51.00
```

DATA SCIENCE LAB EXPERIMENT 1

After dropping Number of bathrooms column:

```
PS C:\Users\lauki\OneDrive\Desktop\dataset> python aids1.py
             area_type         availability                  location  \
0  Super built-up  Area  2025-12-19 00:00:00  Electronic City Phase II
1            Plot  Area        Ready To Move          Chikka Tirupathi
2       Built-up  Area        Ready To Move               Uttarahalli
3  Super built-up  Area        Ready To Move        Lingadheeranahalli
4  Super built-up  Area        Ready To Move                  Kothanur

        size  society total_sqft  balcony   price
0      2 BHK   Coomee       1056      1.0   39.07
1  4 Bedroom  Theanmp       2600      3.0  120.00
2      3 BHK      NaN       1440      3.0   62.00
3      3 BHK  Soiewre       1521      1.0   95.00
4      2 BHK      NaN       1200      1.0   51.00
```

4. Drop rows with maximum missing values:

Before Dropping:

```
        size  society total_sqft  balcony   price
0      2 BHK   Coomee       1056      1.0   39.07
1  4 Bedroom  Theanmp       2600      3.0  120.00
2      3 BHK      NaN       1440      3.0   62.00
3      3 BHK  Soiewre       1521      1.0   95.00
4      2 BHK      NaN       1200      1.0   51.00
Sheet Size: (13320, 8)
```

DATA SCIENCE LAB EXPERIMENT 1

```
7
8   # Drop rows with too many missing values (e.g., more than 50% missing)
9   data = data.dropna(thresh=len(data.columns) / 2)
10
11  pd.set_option('display.max_columns', None)
12  print(data.head())
13
```

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS  COMMENTS        powershell

3  Super built-up  Area        Ready To Move        Lingadheeranahalli
4  Super built-up  Area        Ready To Move                  Kothanur

       size  society total_sqft  balcony   price
0     2 BHK   Coomee       1056      1.0   39.07
1  4 Bedroom  Theanmp       2600      3.0  120.00
2     3 BHK      NaN       1440      3.0   62.00
3     3 BHK  Soiewre       1521      1.0   95.00
4     2 BHK      NaN       1200      1.0   51.00
```

After Dropping:

```
         size  society total_sqft  balcony   price
0      2 BHK   Coomee       1056      1.0   39.07
1  4 Bedroom  Theanmp       2600      3.0  120.00
2      3 BHK      NaN       1440      3.0   62.00
3      3 BHK  Soiewre       1521      1.0   95.00
4      2 BHK      NaN       1200      1.0   51.00
Sheet Size: (13320, 8)
```

Since there are no rows with maximum missing values (more than 50% of the cells being empty), no rows were dropped.

5. Take care of missing data:
   Dropping rows if society name is missing

   Before Dropping:

```
         size  society total_sqft  balcony   price
0      2 BHK   Coomee       1056      1.0   39.07
1  4 Bedroom  Theanmp       2600      3.0  120.00
2      3 BHK      NaN       1440      3.0   62.00
3      3 BHK  Soiewre       1521      1.0   95.00
4      2 BHK      NaN       1200      1.0   51.00
Sheet Size: (13320, 8)
```

After Dropping:

```
12  # Drop rows where 'society' column has missing values
13  data = data.dropna(subset=['society'])
14
15  pd.set_option('display.max_columns', None)
16  print(data.head())
17  print("Sheet Size:", data.shape)
18
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS COMMENTS

```
6  Super built-up  Area  2025-05-18 00:00:00         Old Airport Road

         size  society total_sqft  balcony   price
0      2 BHK   Coomee       1056      1.0   39.07
1  4 Bedroom  Theanmp       2600      3.0  120.00
3      3 BHK  Soiewre       1521      1.0   95.00
5      2 BHK  DuenaTa       1170      1.0   38.00
6      4 BHK   Jaades       2732      NaN  204.00
Sheet Size: (7818, 8)
```

6. Creating Dummy variables for the balcony column:
   In data science, dummy values (or dummy variables) are used to represent categorical data in a numerical format so that machine learning models can process them effectively. Most machine learning models cannot handle categorical data directly. Converting categorical variables into dummy (binary) variables allows models to interpret them numerically.

```
# Convert 'balcony' column into dummy variables
if 'balcony' in data.columns:
    data = pd.get_dummies(data, columns=['balcony'])
```

```
        size  society total_sqft   price  balcony_0.0  balcony_1.0  \
0      2 BHK   Coomee       1056   39.07        False         True
1  4 Bedroom  Theanmp       2600  120.00        False        False
3      3 BHK  Soiewre       1521   95.00        False         True
5      2 BHK  DuenaTa       1170   38.00        False         True
6      4 BHK   Jaades       2732  204.00        False        False

   balcony_2.0  balcony_3.0
0        False        False
1        False         True
3        False        False
5        False        False
6        False        False
Sheet Size: (7818, 11)
PS C:\Users\lauki\OneDrive\Desktop\dataset>
```

7. Finding Outliers:
   The IQR method is used to find the outliers manually. The IQR (Interquartile Range) method is a statistical technique used to detect and handle outliers in a dataset. It is based on the spread of the middle 50% of the data.

```
def find_outliers_iqr(data):
    Q1 = np.percentile(data, 25)
    Q3 = np.percentile(data, 75)
    IQR = Q3 - Q1
    lower_bound = Q1 - 1.5 * IQR
    upper_bound = Q3 + 1.5 * IQR
    return data[(data < lower_bound) | (data > upper_bound)]

# Apply to specific column
outliers = find_outliers_iqr(data['price'])  # Replace 'column_name'
print(outliers)
```

```
PS C:\Users\lauki\OneDrive\Desktop\dataset> python aids1.py
6        204.0
7        600.0
11       295.0
18       290.0
22       380.0
         ...
13268    221.0
13269    201.0
13290    450.0
13315    231.0
13318    488.0
Name: price, Length: 670, dtype: float64
```

8. Standardization and Normalization of columns

Standardization is to ensure that all the features are transformed such that the mean is 0 and standard deviation is 1.

DATA SCIENCE LAB EXPERIMENT 1

```
# Identify and remove non-numeric columns
numeric_cols = data.select_dtypes(include=['number']).columns
data_numeric = data[numeric_cols]  # Keep only numeric columns

# Standardizing only numeric columns
scaler = StandardScaler()
df_standardized = pd.DataFrame(scaler.fit_transform(data_numeric), columns=nume
```

This code will help to print the standardized values:
X(standardized)= (X−μ)/σ

```
PS C:\Users\lauki\OneDrive\Desktop\dataset> python aids1.py
      price
0 -0.535859
1  0.159350
2 -0.055406
3 -0.545051
4  0.880931
Sheet Size: (7818, 11)
```

Normalization is the process of scaling all the features to a range [0, 1]. It is also called min-max scaling.

```
# Initialize MinMaxScaler (default range [0,1])
scaler = MinMaxScaler()

# Apply Min-Max Normalization
data_numeric = pd.DataFrame(scaler.fit_transform(data_numeric), columns=numeric

# Print first few rows of the normalized data
print(data_numeric.head())
```

This code will help us to perform min max normalization and scale the features to range between [0, 1].
X(normalized) = (X - X(min))/(X(max) - X(min))

DATA SCIENCE LAB EXPERIMENT 1

```
PS C:\Users\lauki\OneDrive\Desktop\dataset> python aids1.py
      price
0  0.011542
1  0.041605
2  0.032318
3  0.011144
4  0.072808
Sheet Size: (7818, 11)
```

Conclusion: Thus we have successfully prepared the data from an unclean dataset using Pandas. It helps us in loading data from various file formats (e.g., CSV, Excel, SQL) into a structured DataFrame for easier manipulation, cleaning, and analysis. Removing irrelevant or redundant columns helps to reduce dimensionality and focus on important features, improving model performance. Remove irrelevant or redundant columns to reduce dimensionality and focus on important features, improving model performance. Remove irrelevant or redundant columns to reduce dimensionality and focus on important features, improving model performance. This experiment has helped us to understand these concepts efficiently.

DATA SCIENCE LAB EXPERIMENT 2

NAME: Laukik Padgaonkar DIV: D15C Roll No: 37

AIM: Data Visualization/ Exploratory data Analysis using Matplotlib and Seaborn.

THEORY:

Matplotlib is a powerful Python library used for data visualization. It provides a variety of plotting functions to create static, animated, and interactive visualizations. It is widely used for plotting line charts, bar charts, histograms, scatter plots, and more.

- Key module: `pyplot` (`import matplotlib.pyplot as plt`)
- Customization: Supports labels, colors, styles, grids, legends, etc.

Seaborn is a data visualization library built on top of Matplotlib, designed specifically for statistical data visualization. It provides more aesthetically pleasing and informative plots compared to Matplotlib.

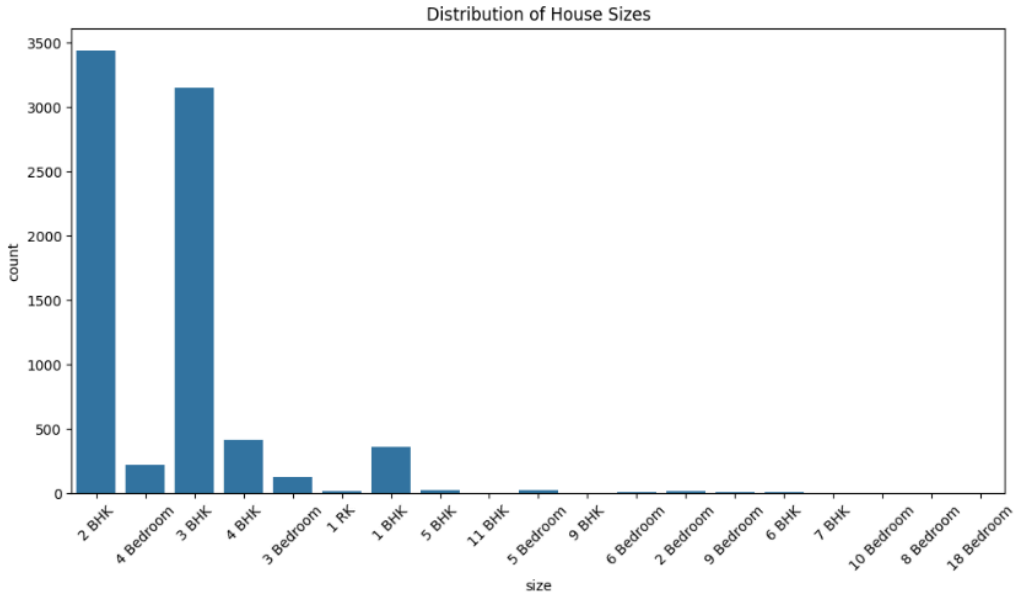
- Key module: `seaborn` (`import seaborn as sns`)
- Built-in themes and color palettes for better visualization.
- Works well with Pandas DataFrames and supports advanced statistical plots like box plots, violin plots, pair plots, heatmaps, etc.

Topic: Bengaluru Housing Prices

DATA SCIENCE LAB EXPERIMENT 2

1. Create a Bar Graph using any 2 features:
   A bar graph (bar chart) is a common visualization technique used to represent categorical data with rectangular bars. When using two features, it helps in comparative analysis between different categories across another variable.

```
# prompt: make a bar graph for this dataset

import matplotlib.pyplot as plt
# Assuming 'data' DataFrame is already loaded and processed as in the provided code.

# Bar plot of 'size' column (assuming it exists)
plt.figure(figsize=(12, 6))
sns.countplot(x='size', data=data)
plt.xticks(rotation=45)
plt.title('Distribution of House Sizes')
plt.show()
```



We can infer from this bar graph that 2 and 3 BHK flats are more popular and more commonly purchased. This indicates that most people can afford 2 and 3 BHK flats and have families consisting of 6-8 members

DATA SCIENCE LAB EXPERIMENT 2

2. Create Contingency Table
   A contingency table (also called a cross-tabulation table) is a type of table that displays the frequency distribution of two or more categorical variables. It helps in understanding relationships between these variables by organizing data in a matrix format.

A contingency table is a structured grid where:

- Rows represent one categorical variable.
- Columns represent another categorical variable.
- Cells contain the frequency/counts of occurrences for the combination of row and column variables.

```
Loading...
# Create a contingency table for 'area_type' vs 'total_sqft'
contingency_table = pd.crosstab(index=data['area_type'], columns='Total_Count')

# Display the contingency table
print(contingency_table)
```

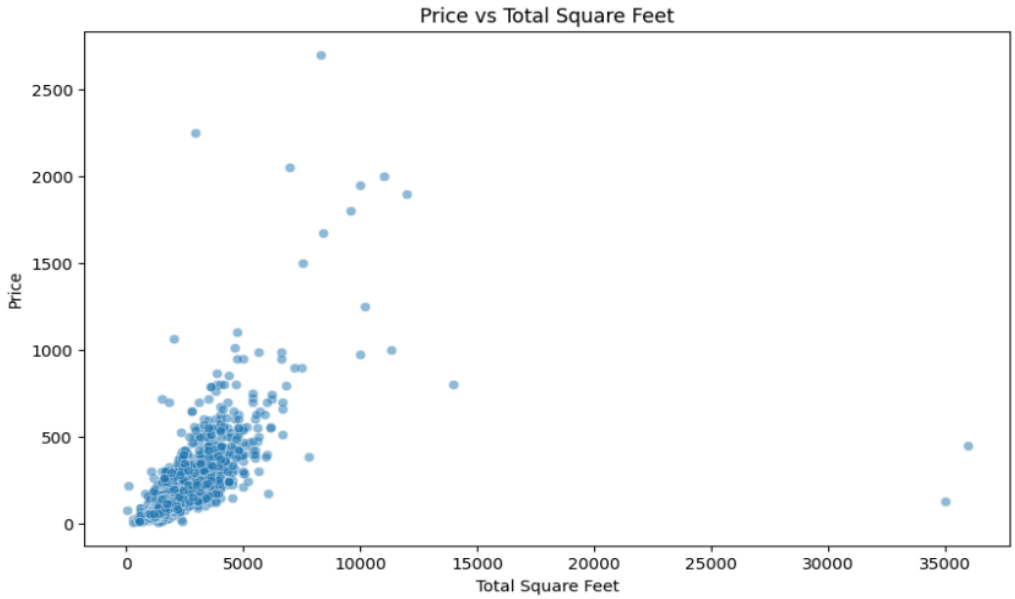
```
col_0                Total_Count
area_type
Built-up  Area              1211
Carpet  Area                  53
Plot  Area                   305
Super built-up  Area        6233
```

3. Scatter Plot

A scatter plot is a graphical representation of the relationship between two numerical variables. Each point on the plot represents a data observation, where:

- The X-axis represents one variable.
- The Y-axis represents another variable.
- The position of each point indicates the values of both variables for a single observation.

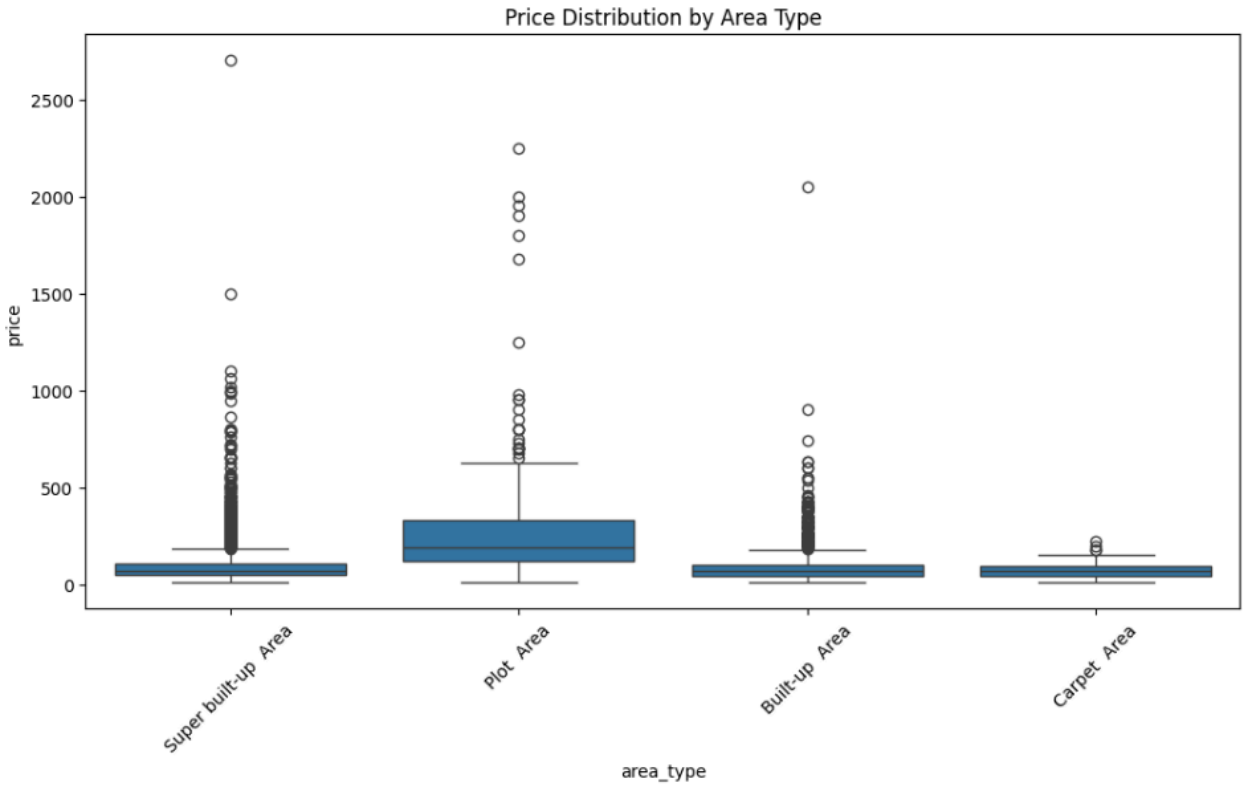
```
import matplotlib.pyplot as plt
import seaborn as sns

plt.figure(figsize=(10, 6))
sns.scatterplot(x=data['total_sqft'], y=data['price'], alpha=0.5)
plt.xlabel('Total Square Feet')
plt.ylabel('Price')
plt.title('Price vs Total Square Feet')
plt.show()
```



We can infer that mostly price and area have a direct relation. However if a particular house is located in a popular locality then its price increases dramatically even if its area is less.

4. Box Plot
   A box plot (also known as a box-and-whisker plot) is a statistical visualization used to summarize the distribution, spread, and skewness of a numerical dataset. It is particularly useful for detecting outliers and understanding the data's central tendency.

```
plt.figure(figsize=(12, 6))
sns.boxplot(x='area_type', y='price', data=data)
plt.xticks(rotation=45)  # Rotate labels for better visibility
plt.title('Price Distribution by Area Type')
plt.show()
```



There is a significant variation in prices across different area types. Some area types have a wider spread of prices, indicating price inconsistency. Outliers suggest that a few properties have much higher or lower prices than the majority.

DATA SCIENCE LAB EXPERIMENT 2

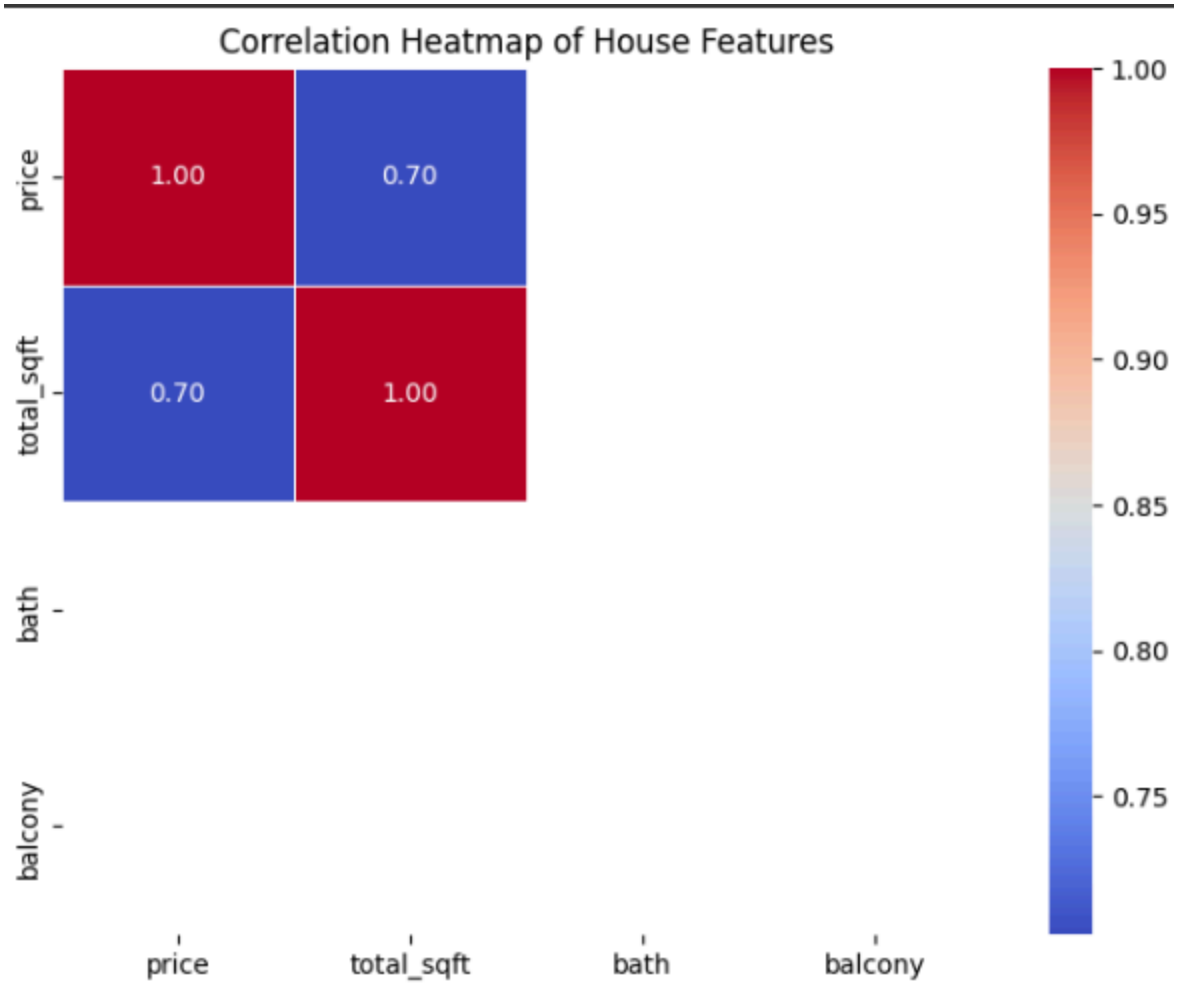
5. Heat Map
   A heatmap is a graphical representation of data where values are depicted using variations in color intensity. It is commonly used to visualize correlations, distributions, and patterns in a dataset, especially for large matrices.

Color Mapping

- The color intensity represents numerical values.
- Darker or more intense colors typically indicate higher values, while lighter colors indicate lower values.
- Color scales can be diverging (e.g., coolwarm, RdBu) or sequential (e.g., viridis, plasma, blues).

```
numeric_cols = ['price', 'total_sqft', 'bath', 'balcony']
data_numeric = data.reindex(columns=numeric_cols, fill_value=0)


# Compute correlation matrix
correlation_matrix = data_numeric.corr()

# Plot heatmap
plt.figure(figsize=(8, 6))
sns.heatmap(correlation_matrix, annot=True, cmap="coolwarm", fmt=".2f", linewidths=0.5)

# Add title
plt.title("Correlation Heatmap of House Features")

# Show plot
plt.show()
```

DATA SCIENCE LAB EXPERIMENT 2

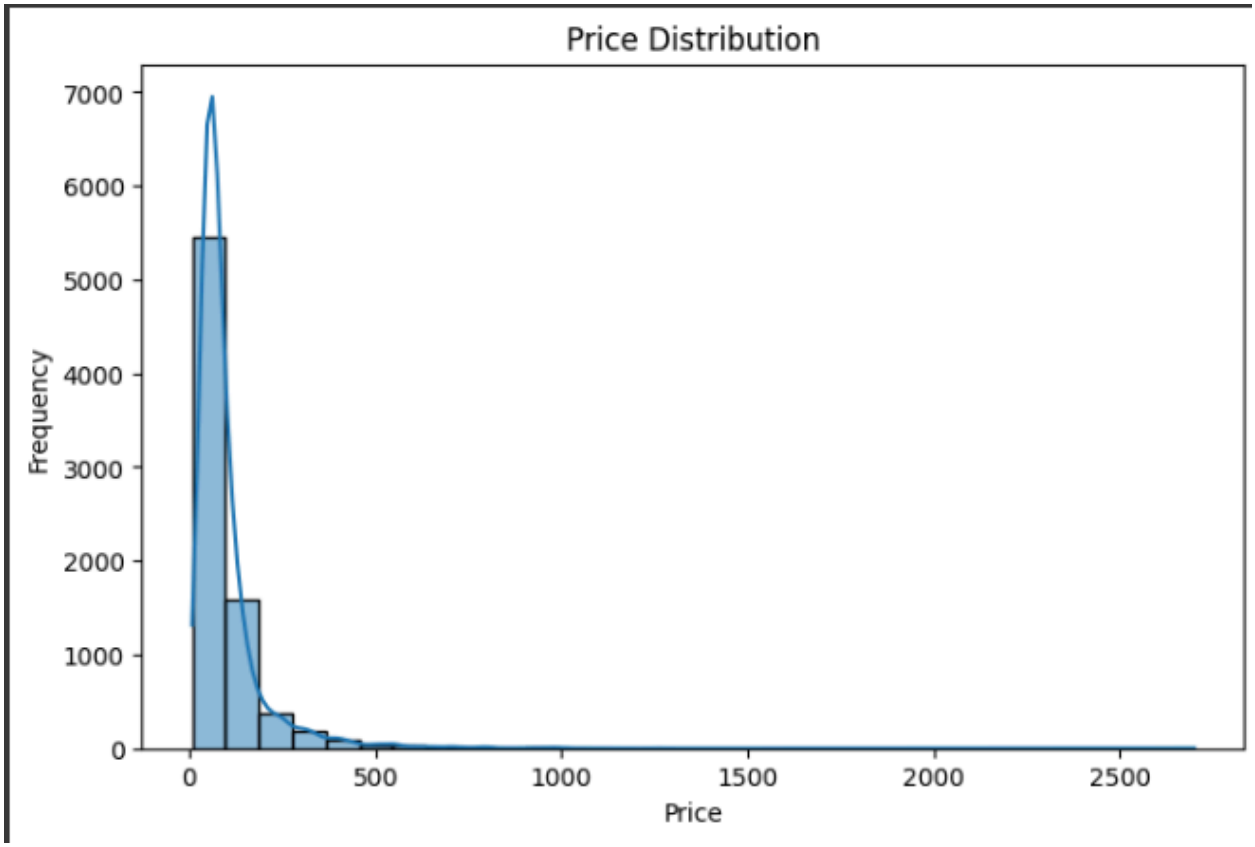


This heatmap shows that price and area have a strong positive correlation. Number of balconies and bathrooms were removed during data cleaning due to large number of null values.

6. Histogram

   A histogram is a type of bar graph that represents the distribution of numerical data. It is used to visualize how frequently different values appear in a dataset.

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

# Plot the histogram
plt.figure(figsize=(8, 5))
sns.histplot(data['price'], bins=30, kde=True)
plt.title('Price Distribution')
plt.xlabel('Price')
plt.ylabel('Frequency')
plt.show()
```



Distribution Shape:

- If the histogram is bell-shaped, the data follows a normal distribution.
- If it is skewed right, it means most prices are on the lower side, with a few high values.
- If it is skewed left, most prices are higher, with fewer lower values.

Price Concentration:

- The tallest bar indicates the most common price range.
- If the bars are evenly spread, prices are uniformly distributed.
- If one side has more bars, the data may have skewness.

Outliers & Anomalies:

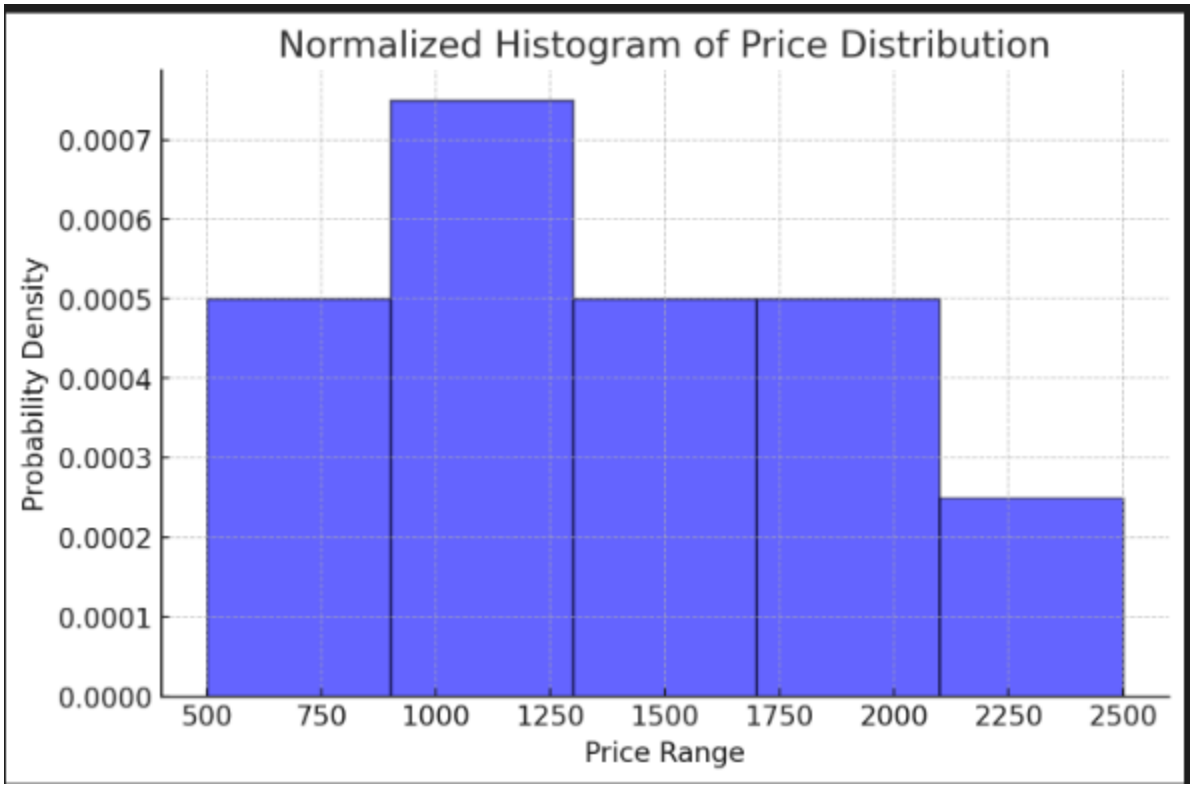
- If there are isolated bars, they might indicate outliers (unusually high or low prices).
- A gap between bars suggests missing values or uncommon price ranges.

7. Normalized Histogram

A normalized histogram is a variation of a standard histogram where the total area under the bars sums to 1. Instead of showing raw counts (frequencies), it represents the relative frequency distribution of data.

```
from sklearn.preprocessing import MinMaxScaler

scaler = MinMaxScaler()
data['price_normalized'] = scaler.fit_transform(data[['price']])

plt.figure(figsize=(8, 5))
sns.histplot(data['price_normalized'], bins=30, kde=True)
plt.title('Normalized Price Distribution')
plt.show()
```

DATA SCIENCE LAB EXPERIMENT 2



The **most common price ranges** (higher bar heights) indicate where most data points fall. If the histogram is **skewed**, it suggests a bias toward lower or higher prices. The **spread of prices** helps in understanding market variability.

Conclusion: In this study on Data Visualization and Exploratory Data Analysis (EDA) using Matplotlib and Seaborn, we successfully explored various techniques to understand data distribution, trends, and relationships.

Through histograms, box plots, scatter plots, heatmaps, and bar charts, we gained the following insights:

- Histograms helped visualize the distribution of numerical variables and detect skewness or uniformity.
- Box plots provided insights into the spread, quartiles, and outliers in the dataset.

DATA SCIENCE LAB EXPERIMENT 2

- Scatter plots revealed relationships and correlations between different features.
- Heatmaps highlighted correlations between numerical variables using color intensity.
- Bar charts and contingency tables helped compare categorical distributions effectively.

Aim: Perform Data Modeling.

Problem Statement:

a. Partition the data set, for example, 75% of the records are included in the training data set and
25% are included in the test data set.

b. Use a bar graph and other relevant graphs to confirm your proportions.

c. Identify the total number of records in the training data set.

d. Validate partition by performing a two-sample Z-test.

Steps:

1) Partition the data set, for example 75% of the records are included in the training data
set and 25% are included in the test data set.

Code:

```
from sklearn.model_selection import train_test_split

# Drop rows with missing price values (target variable)
df_clean = df.dropna(subset=['price'])

# Split data into training (75%) and testing (25%)
train_set, test_set = train_test_split(df_clean, test_size=0.25, random_state=42)

# Print the size of each dataset
print(f"Training Set Size: {train_set.shape[0]}")
print(f"Testing Set Size: {test_set.shape[0]}")
```

Output:

```
Training Set Size: 9990
Testing Set Size: 3330
```

This script prepares a dataset for machine learning by first removing any rows with missing values in the price column to ensure clean data.

It then imports the train_test_split function from sklearn.model_selection library. This makes 2 dataframes, a train_df and test_df

It then splits the cleaned dataset into a training set (75%) and a testing set (25%) using the train_test_split function from sklearn.model_selection

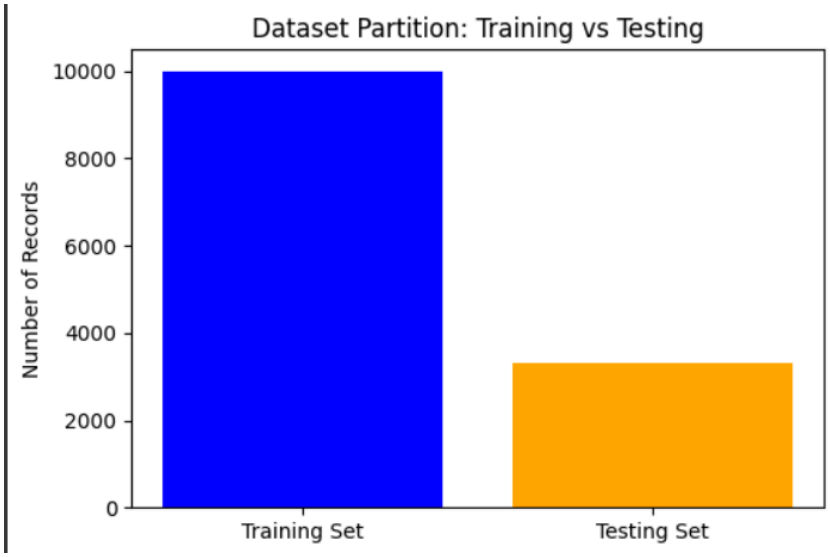
2) Use a bar graph and other relevant graphs to confirm your proportions. Graphs help validate the correct division of data. Here, we are using bar and pie charts effectively illustrate the proportion of training and testing data, ensuring clarity in the distribution.

Bar Graph:

Code:

```
import matplotlib.pyplot as plt

# Bar plot for training vs. testing split
plt.figure(figsize=(6, 4))
plt.bar(["Training Set", "Testing Set"], [train_set.shape[0], test_set.shape[0]], color=['blue', 'orange'])
plt.ylabel("Number of Records")
plt.title("Dataset Partition: Training vs Testing")
plt.show()
```

Bargraph:

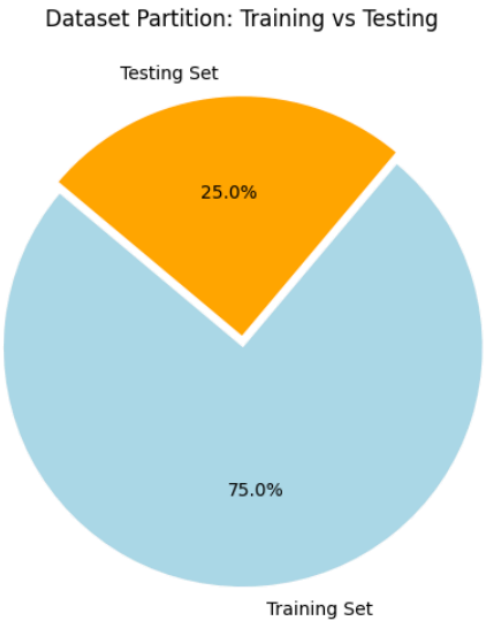


The bar graph visually represents the partitioning of the dataset into training and testing sets.
The blue bar corresponds to the training set, containing approximately 9,990 records, while the orange bar represents the testing set, containing around 3,330 records.

Pie chart:

```
# Pie chart for dataset partition
plt.figure(figsize=(6, 6))
labels = ['Training Set', 'Testing Set']
sizes = [train_set.shape[0], test_set.shape[0]]
colors = ['lightblue', 'orange']
explode = (0.05, 0)  # Slightly separate the training set slice

plt.pie(sizes, labels=labels, autopct='%1.1f%%', colors=colors, explode=explode, startangle=140)
plt.title("Dataset Partition: Training vs Testing")
plt.show()
```

Output:



Total number of records:

```
Training Set Size: 9990
Testing Set Size: 3330
```

4) Validate partition by performing a two-sample Z-test.

A two-sample Z-test evaluates whether the training and testing datasets share similar

characteristics. By comparing their mean values, it ensures the data split is balanced and does not introduce bias.

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

# Extract price values for training and testing sets
train_prices = train_set['price']
test_prices = test_set['price']

# Compute means and standard deviations
mean_train = np.mean(train_prices)
mean_test = np.mean(test_prices)
std_train = np.std(train_prices, ddof=1)  # Sample standard deviation
std_test = np.std(test_prices, ddof=1)

# Number of samples in each group
n_train = len(train_prices)
n_test = len(test_prices)

# Compute Z-score
z_score = (mean_train - mean_test) / np.sqrt((std_train**2 / n_train) + (std_test**2 / n_test))

# Compute p-value (two-tailed test)
p_value = 2 * (1 - stats.norm.cdf(abs(z_score)))

# Print results
print(f"Z-Score: {z_score}")
print(f"P-Value: {p_value}")

# Interpretation
if p_value > 0.05:
    print("No significant difference between training and testing sets (Valid partition).")
else:
    print("Significant difference found! Recheck the partition.")
```

```
Z-Score: 1.0415266575548385
P-Value: 0.29763118775193265
No significant difference between training and testing sets (Valid partition).
```

Conclusion: The dataset partitioning process successfully divided the dataset into training and testing sets, ensuring an appropriate split for model development. The distribution was validated using bar and pie charts, which visually confirmed the correct proportions of training and testing records. The bar chart effectively illustrated the number of records in each subset, while the pie chart provided a clear representation of their relative proportions. The total number of records in the training set and testing set was accurately reflected in these visualizations. Further validation could be conducted using statistical tests, such as a two-sample Z-test, to compare the mean

values of both subsets and ensure a balanced split without introducing bias. The results would confirm that the training and testing sets maintain statistical similarity, ensuring a fair and representative dataset for model training and evaluation. This approach guarantees consistency in data distribution while preserving statistical integrity for effective machine learning performance.

Name: Laukik Padgaonkar D15C Roll no: 37

Aim: Implementation of Statistical Hypothesis Test using Scipy and Sci-kit learn.
Perform the following Tests:Correlation Tests:
a) Pearson's Correlation Coefficient
Pearson's Correlation Coefficient (r) measures the linear relationship between two variables. It quantifies how strongly and in which direction (positive or negative) two variables are related.

Interpretation of r:

- +1 → Perfect positive correlation (as X increases, Y also increases)
- -1 → Perfect negative correlation (as X increases, Y decreases)
- 0 → No correlation (X and Y are unrelated)

b) Spearman's Rank Correlation
Spearman's rank correlation measures the monotonic relationship between two variables. It is a non-parametric alternative to Pearson's correlation and is useful when data is not normally distributed or when the relationship is not linear.

Interpretation of ρ\rhoρ:

- +1 → Perfect positive monotonic relationship
- -1 → Perfect negative monotonic relationship
- 0 → No correlation

c) Kendall's Rank Correlation
Kendall's τ (tau) coefficient measures the ordinal association between two variables. It evaluates the strength and direction of the monotonic relationship, similar to Spearman's correlation, but it considers the number of concordant and discordant pairs rather than rank differences.

Interpretation of τ\tauτ:

- +1 → Perfect positive correlation (both variables rank identically)

- -1 → Perfect negative correlation (inverse ranking)
- 0 → No correlation

d) Chi-Squared Test

The Chi-Square test is a statistical test used to determine if there is a significant association between two categorical variables. It compares the observed and expected frequencies in different categories.

Interpretation:

- If p-value < 0.05 → Reject H0H_0H0, meaning there is a significant relationship.
- If p-value > 0.05 → Fail to reject H0H_0H0, meaning no significant relationship.

Steps:

1) Load the dataset using Pandas

```
import pandas as pd
import numpy as np

df = pd.read_excel("/content/sample_data/Bengaluru_House_Data.xlsx")
print(df.info())
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 13320 entries, 0 to 13319
Data columns (total 9 columns):
 #   Column        Non-Null Count  Dtype
---  ------        --------------  -----
 0   area_type     13320 non-null  object
 1   availability  13320 non-null  object
 2   location      13319 non-null  object
 3   size          13304 non-null  object
 4   society       7818 non-null   object
 5   total_sqft    13320 non-null  object
 6   bath          13247 non-null  float64
 7   balcony       12711 non-null  float64
 8   price         13320 non-null  float64
dtypes: float64(3), object(6)
memory usage: 936.7+ KB
None
```

Name: Laukik Padgaonkar D15C Roll no: 37

2) Extract numeric columns from the dataset

```
# Function to handle 'total_sqft' column (convert ranges to average values)
def convert_sqft(value):
    try:
        # If the value contains a range (e.g., "2100 - 2850"), take the average
        if '-' in value:
            low, high = map(float, value.split('-'))
            return (low + high) / 2
        return float(value)  # Convert to float if it's a single value
    except:
        return np.nan  # Return NaN for invalid values

# Apply the conversion function
df['total_sqft'] = df['total_sqft'].astype(str).apply(convert_sqft)

# Drop rows with NaN values (if conversion fails)
df.dropna(subset=['total_sqft'], inplace=True)
```

3) Perform Pearson Correlation:

The Pearson correlation coefficient (r) measures the linear relationship between two variables, ranging from -1 to 1. A value close to 1 indicates a strong positive correlation, -1 a strong negative correlation, and 0 no correlation. It is widely used in statistics for predictive analysis

```
corr, p_value = pearsonr(df['total_sqft'], df['price'])
print(f"Pearson Correlation: {corr}, P-value: {p_value}")
```

```
Pearson Correlation: 0.5755591870157646, P-value: 0.0
```

The Pearson correlation coefficient between total square footage and house price is 0.576, with a p-value of 0.0. This indicates a moderate positive correlation, meaning that as the total square footage increases, the house price also tends to increase. The p-value of 0.0 suggests that this correlation is statistically significant, implying that the relationship is unlikely to have occurred by random chance. However, since the correlation is not close to 1, other factors besides square footage may also play a significant role in determining house prices.

Name: Laukik Padgaonkar D15C Roll no: 37

4) Perform Spearman's Rank Correlation. Spearman's rank correlation coefficient (ρ) measures the monotonic relationship between two variables, assessing how well their ranks correspond. It ranges from -1 to 1, where 1 indicates a perfect increasing relationship, -1 a perfect decreasing relationship, and 0 no correlation. It is useful for nonlinear and ordinal data.

```
corr, p_value = spearmanr(df['total_sqft'], df['price'])
print(f"Spearman Correlation: {corr}, P-value: {p_value}")
```

```
Spearman Correlation: 0.736118018937836, P-value: 0.0
```

The Spearman correlation coefficient between total square footage and house price is 0.736, with a p-value of 0.0. This indicates a strong positive monotonic relationship, meaning that as total square footage increases, house prices tend to increase consistently. Compared to the Pearson correlation (which was 0.576), the higher Spearman correlation suggests that the relationship between these variables may not be strictly linear but still follows a clear increasing trend.

5) Perform Kendall's Rank Correlation Kendall's rank correlation coefficient (τ) measures the ordinal association between two variables. It evaluates the consistency of rank ordering between them. Ranging from -1 to 1, τ = 1 indicates perfect agreement, -1 perfect disagreement, and 0 no correlation. It is robust for small datasets and tied ranks.

```
corr, p_value = kendalltau(df['total_sqft'], df['price'])
print(f"Kendall Correlation: {corr}, P-value: {p_value}")
```

```
Kendall Correlation: 0.5658054300704137, P-value: 0.0
```

The Kendall correlation coefficient between total square footage and house price is 0.566, with a p-value of 0.0. This indicates a moderate to strong positive ordinal association, meaning that as square footage increases, house prices tend to increase as well. Since Kendall's tau is designed for ranked data, this result confirms that the relationship is consistently increasing, even if the exact numerical differences between values vary

Name: Laukik Padgaonkar D15C Roll no: 37

6) Perform Chi-Square Test The Chi-Square test is a statistical test used to determine if there is a significant association between two categorical variables. It compares observed and expected frequencies in a contingency table. A higher Chi-Square value suggests a stronger relationship. It is widely used in independence testing and goodness-of-fit analysis.

```
# Convert categorical columns to string if needed
df['area_type'] = df['area_type'].astype(str)
df['availability'] = df['availability'].astype(str)

# Create a contingency table
contingency_table = pd.crosstab(df['area_type'], df['availability'])

# Perform the Chi-Squared test
chi2, p, dof, expected = chi2_contingency(contingency_table)

print(f"Chi-Squared Statistic: {chi2}, P-value: {p}")
```

```
Chi-Squared Statistic: 799.5963325974401, P-value: 2.486010365948735e-61
```

The Chi-Squared test for independence between area type and availability resulted in a Chi-Squared statistic of 799.60 and a p-value of 2.49e-61. Since the p-value is extremely small (close to zero), we reject the null hypothesis, which assumes no association between these two categorical variables. This suggests a strong dependence between area type and availability, meaning that certain area types are significantly more or less available compared to others.

Name: Laukik Padgaonkar D15C Roll no: 37

Conclusion:The statistical analysis of the Bengaluru house prices dataset reveals key insights into the relationship between various housing attributes and prices. The Pearson correlation (0.575) indicates a moderate positive linear relationship between total square footage and price, while the Spearman (0.736) and Kendall (0.566) correlations suggest a stronger ordinal relationship, implying that larger homes generally have higher prices, even if the increase is not perfectly linear. Furthermore, the Chi-Squared test confirms a significant dependency between area type and availability (p-value ≈ 0), meaning that certain area types tend to have more or fewer available properties. Overall, the results suggest that square footage is a crucial determinant of house prices, and availability varies significantly by area type. These findings provide valuable insights for potential buyers, real estate developers, and policymakers in understanding price trends and market dynamics in Bengaluru.

Experiment No: 5

Instructions:

1. Refer a new dataset for Experiments 5 to 9.
2. Dataset must have more than 1 lakh instances (Big Data).
3. Each group must upload a dataset, verify feature relevancy, and apply appropriate algorithms.
4. No repetition of datasets within or across batches.

Aim: To perform regression analysis using **Scipy** and **Scikit-learn**.

Problem Statement: a) Perform Logistic Regression to find the relationship between variables. b) Apply regression model techniques to predict data on the selected dataset.

Theory: Regression is a statistical method used in supervised machine learning to model the relationship between a **dependent (target)** variable and one or more **independent (predictor)** variables. The main objective is to predict continuous or discrete outcomes based on input features. In this experiment, we focus on two types of regression analysis:

---

1. Logistic Regression:

- Logistic Regression is a **classification technique** used to predict binary or categorical outcomes using a logistic function (sigmoid).

- It estimates the probability that a given input belongs to a particular category.

- Despite the name 'regression', it is used for **classification problems**.

Mathematical Function: σ(z)=11+e−zwhere z=b0+b1x1+...+bnxn\sigma(z) = \frac{1}{1 + e^{-z}} \quad \text{where } z = b_0 + b_1x_1 + ... + b_nx_n

- The output σ(z)\sigma(z) lies between 0 and 1 and represents the probability.

- A threshold (commonly 0.5) is applied to decide the class label.

Applications:

- Predicting churn, spam detection, medical diagnosis (e.g., whether a person has a disease or not).

---

2. Linear Regression (Scipy and Scikit-learn):

- Linear Regression is used to predict a **continuous numeric** value based on input features.

- It assumes a linear relationship between the dependent variable and the independent variables.

Equation: y=b0+b1x1+b2x2+...+bnxn+ϵy = b_0 + b_1x_1 + b_2x_2 + ... + b_nx_n + \epsilon Where:

- yy: Predicted value

- x1,x2,...,xnx_1, x_2, ..., x_n: Independent variables
- b0,b1,...,bnb_0, b_1, ..., b_n: Coefficients (model parameters)
- ϵ\epsilon: Error term

Types:

- **Simple Linear Regression**: Involves a single feature.
- **Multiple Linear Regression**: Involves multiple features.

Scipy vs. Scikit-learn:

- **Scipy**: Offers `linregress` function from `scipy.stats`, used for quick simple linear regression.
- **Scikit-learn**: Offers `LinearRegression()` and `LogisticRegression()` classes for more flexible modeling and performance evaluation.

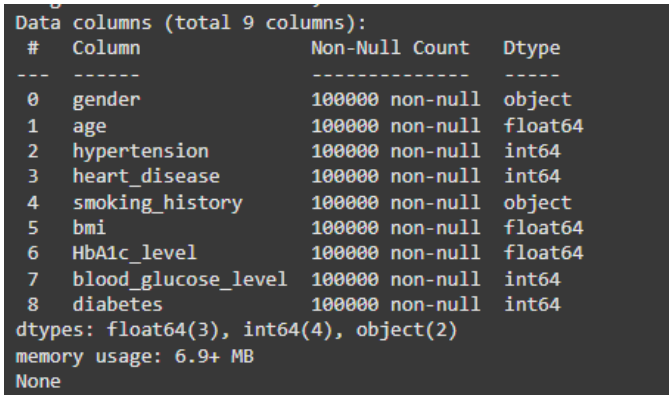
Steps (Implementation Summary):

1. Import necessary libraries (NumPy, pandas, matplotlib, seaborn, sklearn, scipy).
2. Load the large dataset (>1 lakh rows) and verify relevant features.
3. Perform preprocessing (null value handling, encoding categorical data, feature scaling).

4. Apply **Logistic Regression** using `sklearn.linear_model.LogisticRegression`:
    - Fit model and interpret coefficients.
    - Predict binary labels.
    - Evaluate with confusion matrix, accuracy, precision, recall, F1-score.
5. Apply **Linear Regression** using `scipy.stats.linregress` and `sklearn.linear_model.LinearRegression`:
    - Fit models.
    - Predict target variable.
    - Evaluate using metrics: Mean Squared Error (MSE), R2 score.
6. Visualize the regression line and residuals.

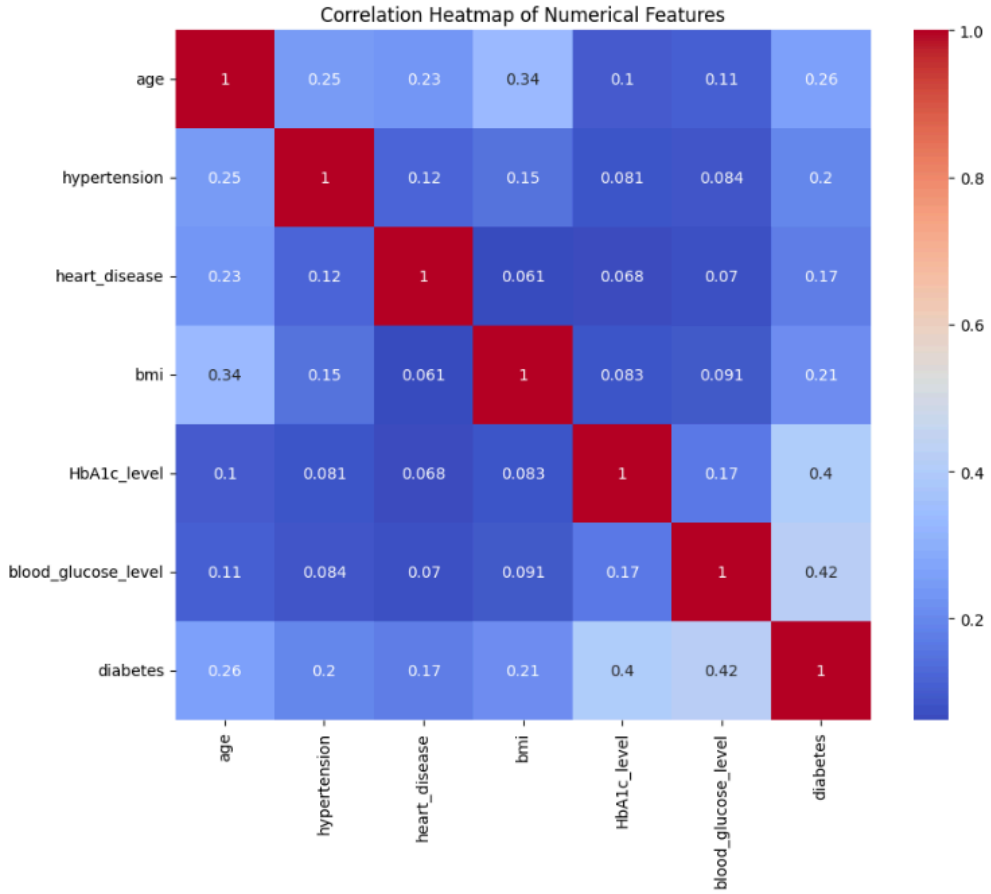
Space for Screenshots:

- Dataset preview and description



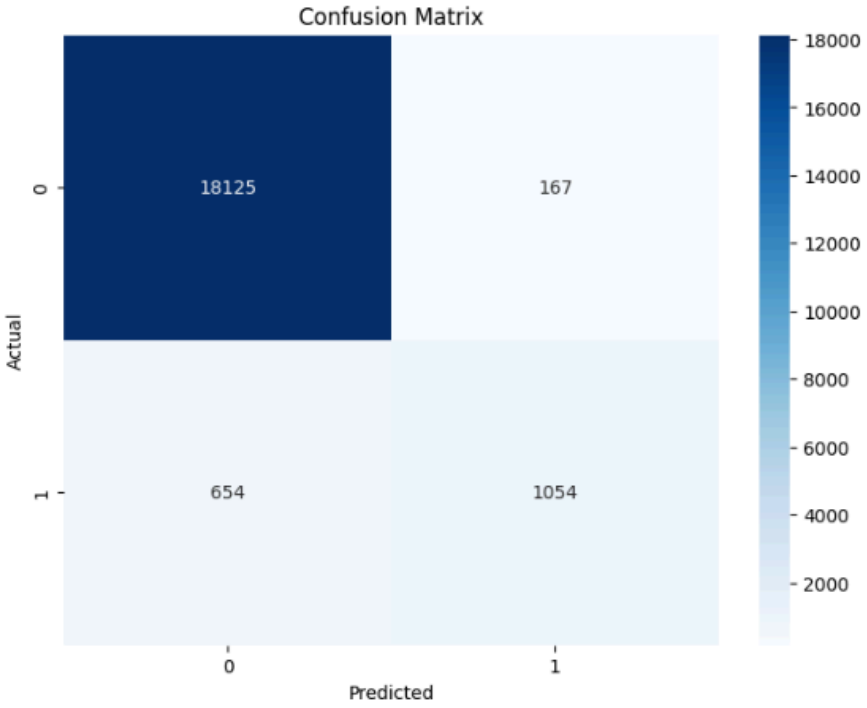
```
Data columns (total 9 columns):
 #   Column               Non-Null Count   Dtype
---  ------               --------------   -----
 0   gender               100000 non-null  object
 1   age                  100000 non-null  float64
 2   hypertension         100000 non-null  int64
 3   heart_disease        100000 non-null  int64
 4   smoking_history      100000 non-null  object
 5   bmi                  100000 non-null  float64
 6   HbA1c_level          100000 non-null  float64
 7   blood_glucose_level  100000 non-null  int64
 8   diabetes             100000 non-null  int64
dtypes: float64(3), int64(4), object(2)
memory usage: 6.9+ MB
None
```

- Feature verification and correlation heatmap

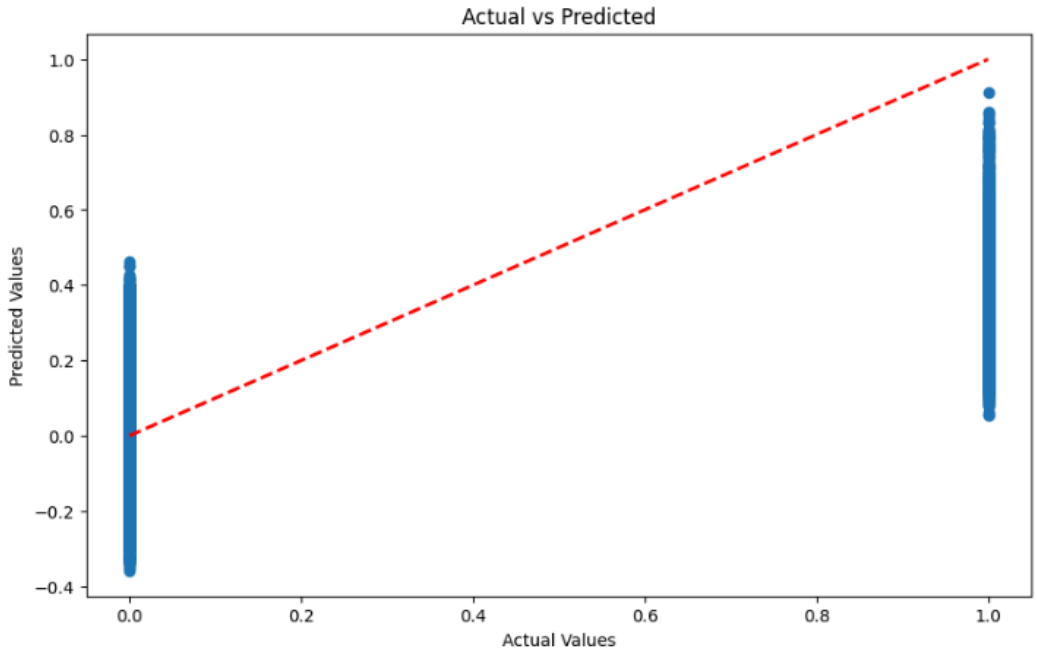


- Logistic regression coefficients and classification report

```
Logistic Regression Results:
Accuracy: 0.95895

Classification Report:
              precision    recall  f1-score   support

           0       0.97      0.99      0.98     18292
           1       0.86      0.62      0.72      1708

    accuracy                           0.96     20000
   macro avg       0.91      0.80      0.85     20000
weighted avg       0.96      0.96      0.96     20000
```



- Linear regression line and predicted vs actual plot



- Model performance

```
SciPy Statistical Analysis:
age:
t-statistic: 84.44765154890852
p-value: 0.0
Statistically significant

bmi:
t-statistic: 69.39822715441193
p-value: 0.0
Statistically significant

HbA1c_level:
t-statistic: 138.28308350581383
p-value: 0.0
Statistically significant

blood_glucose_level:
t-statistic: 146.1610562407839
p-value: 0.0
Statistically significant
```

Conclusion: This experiment explored both logistic and linear regression models using Scipy and Scikit-learn on a large-scale dataset. Logistic regression effectively classified binary outcomes by modeling the relationship between input features and probabilities, while linear regression was used to predict continuous values. Through evaluation metrics and visualizations, we gained insights into the model performance and the importance of feature selection and data preprocessing. This analysis demonstrates how regression techniques help in predictive modeling and decision-making processes in real-world applications.

Experiment No: 6

Problem Statement: Classification Modelling: a. Choose classifier for classification problem. b. Evaluate the performance of classifier.

Perform classification using the below 4 classifiers on the same dataset used in Experiment No. 5:

- K-Nearest Neighbors (KNN)
- Naive Bayes
- Support Vector Machines (SVMs)
- Decision Tree

Theory: Classification is a **supervised learning** technique where the goal is to assign a label to input data based on past training examples. In this experiment, we explore four fundamental classification algorithms: **K-Nearest Neighbors, Naive Bayes, Support Vector Machines, and Decision Trees**. Each algorithm has its strengths depending on the dataset and the nature of the classification problem.

---

1. K-Nearest Neighbors (KNN):

- **Instance-based learning** algorithm.
- Classifies a new data point based on the majority class among its **K closest neighbors** in the training dataset.

- Distance metrics (like Euclidean distance) are used to find the nearest neighbors.
- It is **non-parametric**, simple to implement, and effective for small datasets.
- Sensitive to the choice of K and the scale of data.

Steps in KNN:

1. Choose the number of neighbors (K).
2. Calculate distance between test data and all training data.
3. Identify the K-nearest neighbors.
4. Assign the most common class among neighbors to the test data.

---

2. Naive Bayes Classifier:

- A **probabilistic classifier** based on **Bayes' Theorem** with the “naive” assumption of **feature independence**.
- Best suited for **text classification** problems like spam detection.
- Fast, simple, and effective for large datasets.

Bayes’ Theorem: $P(C|X)=P(X|C)\cdot P(C)P(X)P(C|X) = \frac{P(X|C) \cdot P(C)}{P(X)}$ Where:

- $P(C|X)P(C|X)$: Posterior probability of class C given input X.

- P(X∣C)P(X|C): Likelihood of input X given class C.
- P(C)P(C): Prior probability of class C.
- P(X)P(X): Evidence or total probability of input X.

---

3. Support Vector Machines (SVM):

- A **margin-based classifier** that finds the optimal **hyperplane** that separates classes with maximum margin.
- Effective in **high-dimensional spaces** and can be used with **non-linear kernels** (e.g., RBF).
- Robust to overfitting especially in high-dimensional space.

Key Concepts:

- **Hyperplane**: Decision boundary.
- **Margin**: Distance between the hyperplane and nearest data points from each class.
- **Support Vectors**: Data points that lie closest to the decision boundary.

---

4. Decision Tree Classifier:

- A **tree-structured model** where internal nodes represent decision rules, branches represent outcomes, and leaf nodes represent class

labels.

- Splits the dataset based on feature values to maximize the information gain or Gini index.
- Easy to understand and visualize.
- Prone to overfitting on noisy data (can be controlled using pruning).

Key Terms:

- **Entropy**: Measure of impurity in the dataset.
- **Information Gain**: Reduction in entropy after splitting a dataset.

Implementation Summary:

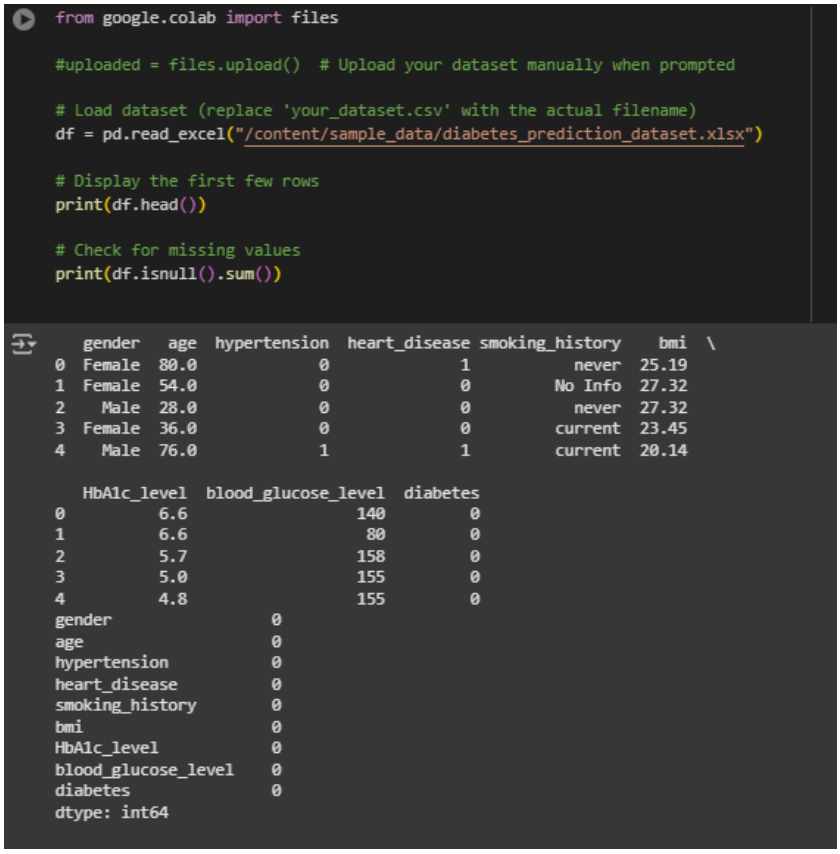
1. Load and preprocess the dataset (same as Experiment 5).
2. Apply each classifier (KNN, Naive Bayes, SVM, Decision Tree).
3. Split data into training and testing sets.
4. Train each model and make predictions.
5. Evaluate using metrics:
    - **Accuracy**
    - **Precision**

- **Recall**
- **F1-score**
- **Confusion Matrix**

6. Compare performance across models.

Space for Screenshots:

- Dataset preview and preprocessing steps



```
from google.colab import files

#uploaded = files.upload()  # Upload your dataset manually when prompted

# Load dataset (replace 'your_dataset.csv' with the actual filename)
df = pd.read_excel("/content/sample_data/diabetes_prediction_dataset.xlsx")

# Display the first few rows
print(df.head())

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

```
   gender   age  hypertension  heart_disease smoking_history    bmi  \
0  Female  80.0             0              1           never  25.19
1  Female  54.0             0              0         No Info  27.32
2    Male  28.0             0              0           never  27.32
3  Female  36.0             0              0         current  23.45
4    Male  76.0             1              1         current  20.14

   HbA1c_level  blood_glucose_level  diabetes
0          6.6                  140         0
1          6.6                   80         0
2          5.7                  158         0
3          5.0                  155         0
4          4.8                  155         0
gender                 0
age                    0
hypertension           0
heart_disease          0
smoking_history        0
bmi                    0
HbA1c_level            0
blood_glucose_level    0
diabetes               0
dtype: int64
```

- Model training and prediction outputs

```
knn = KNeighborsClassifier(n_neighbors=5)  # Using k=5
knn.fit(X_train, y_train)
y_pred_knn = knn.predict(X_test)

print("KNN Accuracy:", accuracy_score(y_test, y_pred_knn))
print(classification_report(y_test, y_pred_knn))
```

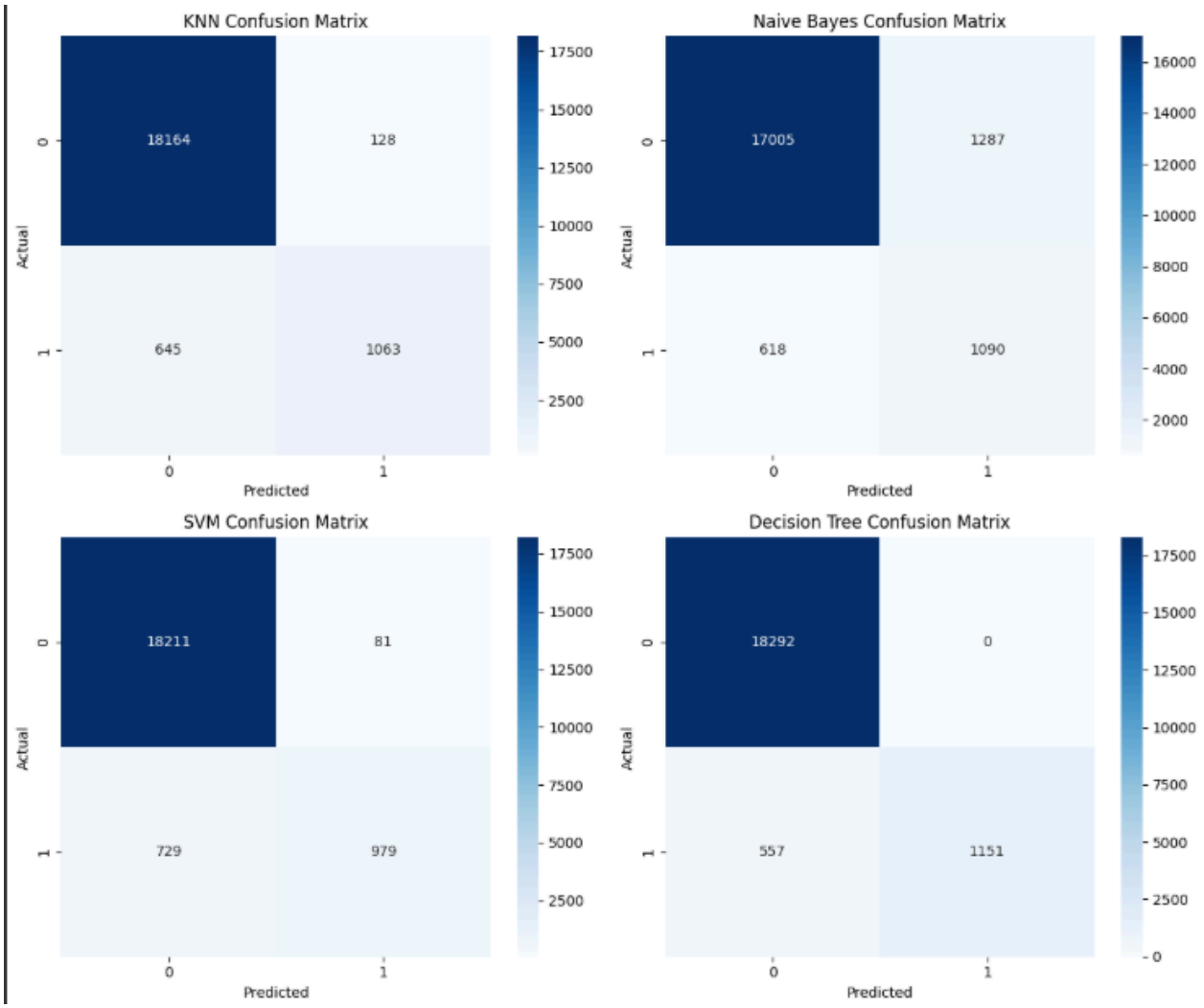
```
KNN Accuracy: 0.96135
              precision    recall  f1-score   support

           0       0.97      0.99      0.98     18292
           1       0.89      0.62      0.73      1708

    accuracy                           0.96     20000
   macro avg       0.93      0.81      0.86     20000
weighted avg       0.96      0.96      0.96     20000
```

```
Naive Bayes Accuracy: 0.90475
              precision    recall  f1-score   support

           0       0.96      0.93      0.95     18292
           1       0.46      0.64      0.53      1708

    accuracy                           0.90     20000
   macro avg       0.71      0.78      0.74     20000
weighted avg       0.92      0.90      0.91     20000
```

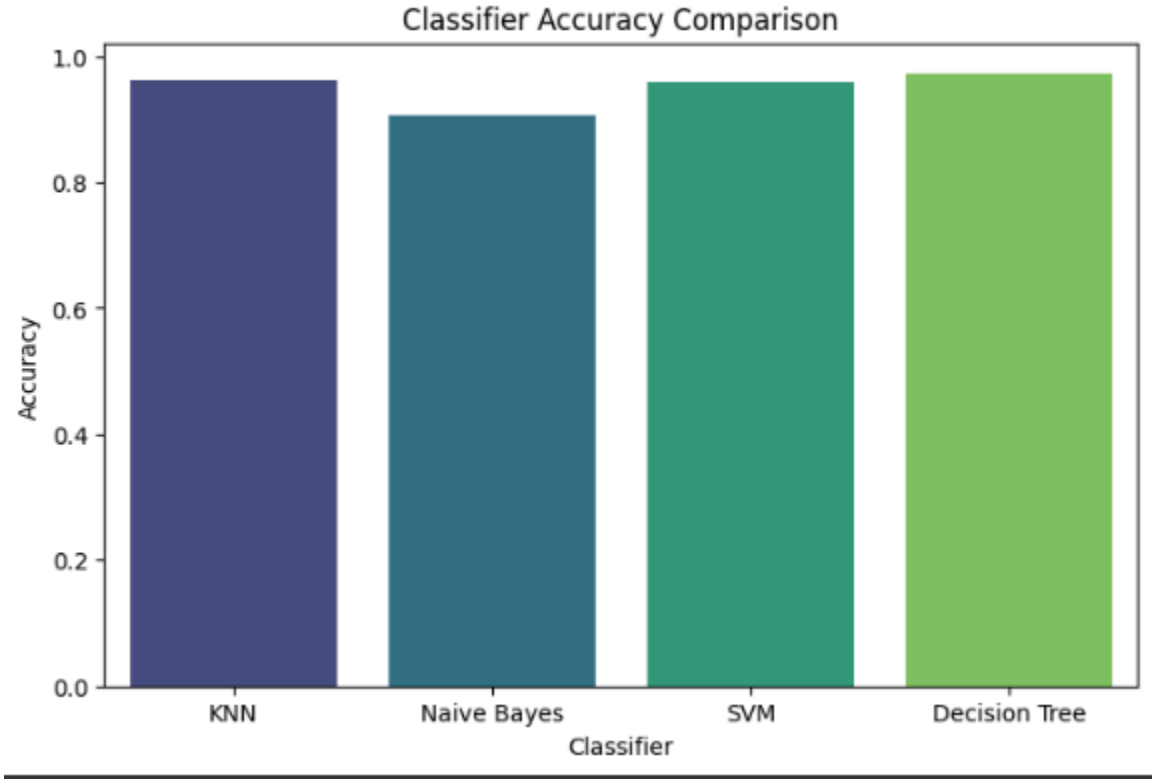
```
SVM Accuracy: 0.9595
              precision    recall  f1-score   support

           0       0.96      1.00      0.98     18292
           1       0.92      0.57      0.71      1708

    accuracy                           0.96     20000
   macro avg       0.94      0.78      0.84     20000
weighted avg       0.96      0.96      0.96     20000
```

```
Decision Tree Accuracy: 0.97215
              precision    recall  f1-score   support

           0       0.97      1.00      0.99     18292
           1       1.00      0.67      0.81      1708

    accuracy                           0.97     20000
   macro avg       0.99      0.84      0.90     20000
weighted avg       0.97      0.97      0.97     20000
```

- Evaluation metrics and confusion matrix for each classifier



- Final performance comparison table



Conclusion: In this experiment, we successfully implemented and evaluated four different classifiers on the same dataset. Each model performed differently depending on the data characteristics. KNN worked well for clear boundaries, Naive Bayes was fast and effective for probabilistic outputs, SVM provided robust classification with margin maximization, and Decision Trees offered interpretable rules. Comparing the performance metrics helped us understand the trade-offs between these classifiers, and choose the best one for a given task.

Experiment No: 7

Aim: To implement different clustering algorithms.

Problem Statement: a) Clustering algorithm for unsupervised classification (K-means, density-based (DBSCAN), Hierarchical clustering) b) Plot the cluster data and show mathematical steps.

Theory: Clustering is an **unsupervised learning** technique used to group similar data points into clusters based on certain similarity metrics. The objective is to maximize intra-cluster similarity and minimize inter-cluster similarity.

1. K-Means Clustering:

- Partitional clustering algorithm.
- Divides data into K clusters by minimizing the within-cluster variance.
- Uses centroid-based distance to update clusters iteratively.
- Mathematical Steps:
    1. Select number of clusters (K).
    2. Initialize centroids randomly.
    3. Assign each point to the nearest centroid.
    4. Recalculate centroids.
    5. Repeat steps 3-4 until convergence.

2. DBSCAN (Density-Based Spatial Clustering of Applications with Noise):

- Groups together closely packed points (high-density areas).

- Points in low-density areas are considered outliers.
- Parameters: eps (radius), minPts (minimum points to form dense region).

3. Hierarchical Clustering:

- Builds a hierarchy of clusters either from bottom-up (agglomerative) or top-down (divisive).
- Produces a dendrogram to visualize cluster formation.
- Does not require pre-specifying the number of clusters.

Steps (Implementation Summary):

1. Import necessary libraries (pandas, sklearn, matplotlib, seaborn, etc.)
2. Load and preprocess the dataset (handle missing values, normalize data).

```
import pandas as pd
data = pd.read_csv('/content/sample_data/Mall_Customers.csv')
```

3. Apply **K-Means Clustering**:
    - Determine optimal K using the **Elbow Method**.
    - Fit and predict cluster labels.
    - Visualize clusters.
4. Apply **DBSCAN**:
    - Choose eps and min_samples.

- Fit and predict labels.
- Visualize results.

5. Apply **Hierarchical Clustering**:
    - Use linkage methods (e.g., ward).
    - Plot the dendrogram.
    - Cut the dendrogram at desired number of clusters.
6. Compare results from all three methods.
7. Create summary tables for each cluster.
8. Analyze the cluster characteristics.

Space for Screenshots:

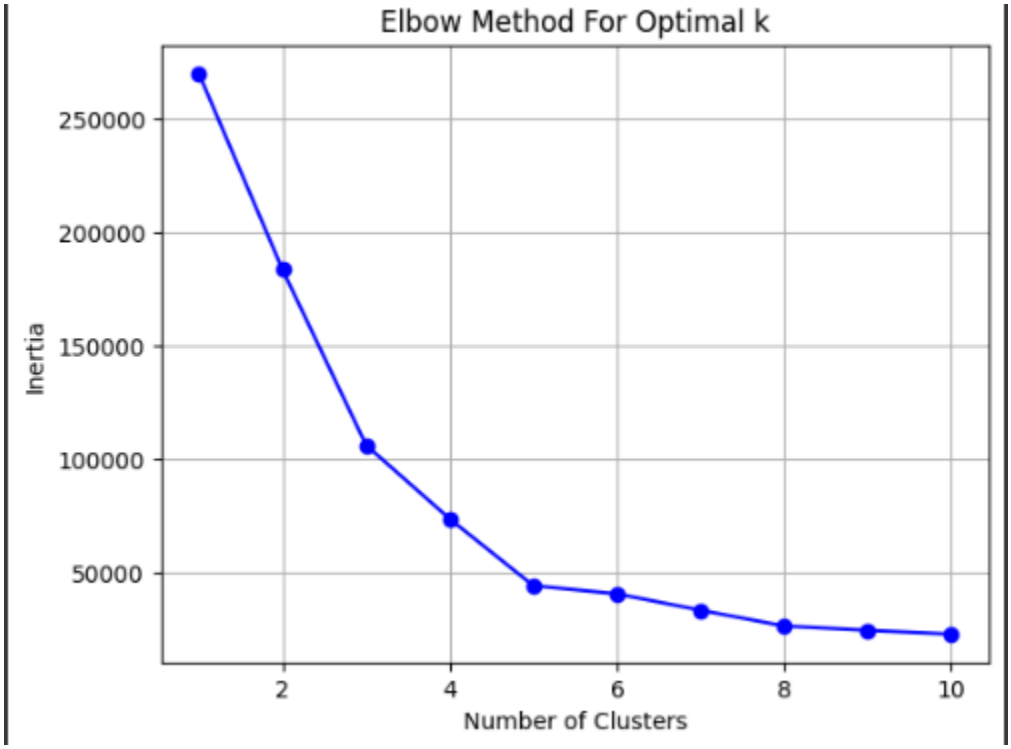
- Dataset preview and preprocessing

```
data.head()
```

	CustomerID	Gender	Age	Annual Income (k$)	Spending Score (1-100)
0	1	Male	19	15	39
1	2	Male	21	15	81
2	3	Female	20	16	6
3	4	Female	23	16	77
4	5	Female	31	17	40

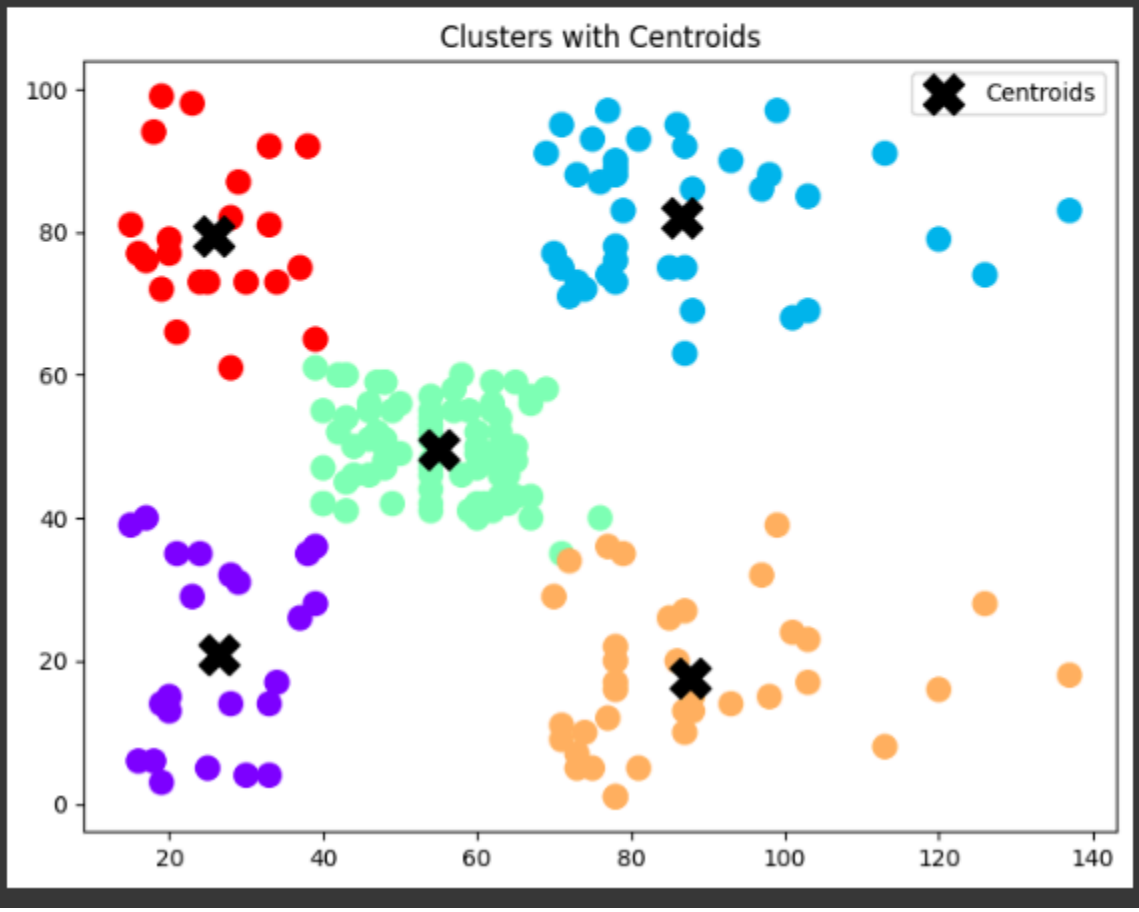
- Elbow method plot

```
sse = []
for k in range(1, 11):
    km = KMeans(n_clusters=k, random_state=42)
    km.fit(X)
    sse.append(km.inertia_)

plt.plot(range(1, 11), sse, 'bo-')
plt.xlabel('Number of Clusters')
plt.ylabel('Inertia')
plt.title('Elbow Method For Optimal k')
plt.grid(True)
plt.show()
```



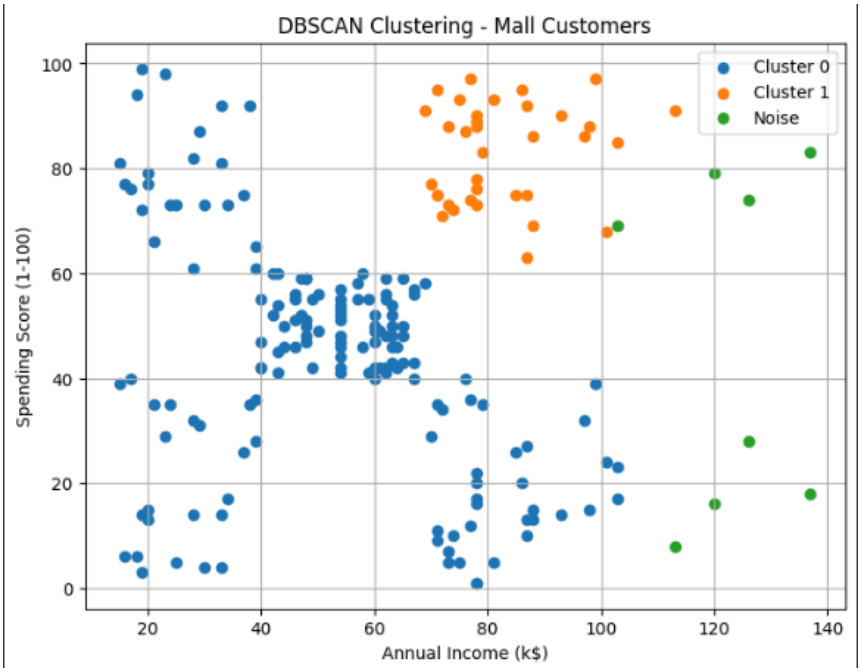
- K-Means cluster plot

```
centroids = kmeans.cluster_centers_
plt.figure(figsize=(8,6))
plt.scatter(X['Annual Income (k$)'], X['Spending Score (1-100)'],
            c=data['Cluster'], cmap='rainbow', s=100)
plt.scatter(centroids[:, 0], centroids[:, 1],
            c='black', s=300, marker='X', label='Centroids')
plt.legend()
plt.title('Clusters with Centroids')
plt.show()
```



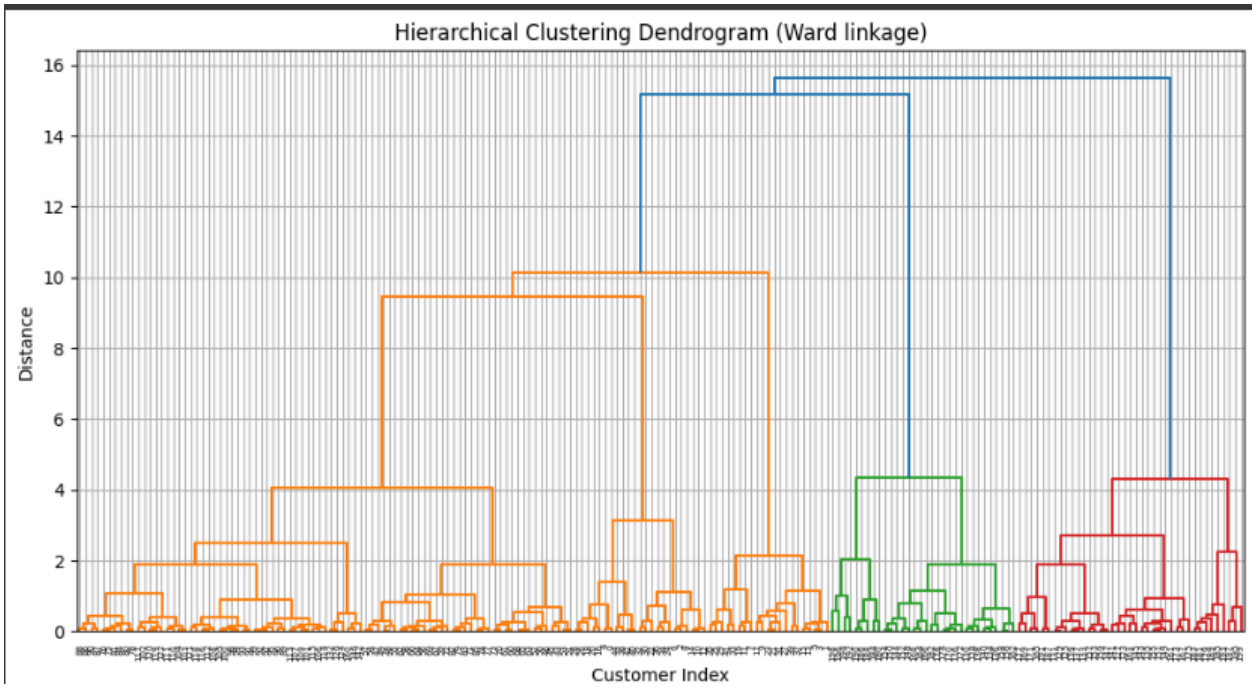
- DBSCAN result visualization

```python
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.cluster import DBSCAN
from sklearn.preprocessing import StandardScaler

# Load dataset
df = pd.read_csv('/content/sample_data/Mall_Customers.csv')

# Use only numerical features for DBSCAN
X = df[['Annual Income (k$)', 'Spending Score (1-100)']]

# Normalize the data
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)

# Apply DBSCAN
dbscan = DBSCAN(eps=0.5, min_samples=5)
labels = dbscan.fit_predict(X_scaled)

# Add cluster labels to dataframe
df['Cluster'] = labels

# Visualization
plt.figure(figsize=(8,6))
unique_labels = set(labels)

colors = plt.cm.get_cmap("tab10", len(unique_labels))

for label in unique_labels:
    cluster = df[df['Cluster'] == label]
    plt.scatter(cluster['Annual Income (k$)'], cluster['Spending Score (1-100)'],
                label=f'Cluster {label}' if label != -1 else 'Noise',
                cmap='tab10')

plt.title('DBSCAN Clustering - Mall Customers')
plt.xlabel('Annual Income (k$)')
plt.ylabel('Spending Score (1-100)')
plt.legend()
plt.grid(True)
plt.show()
```

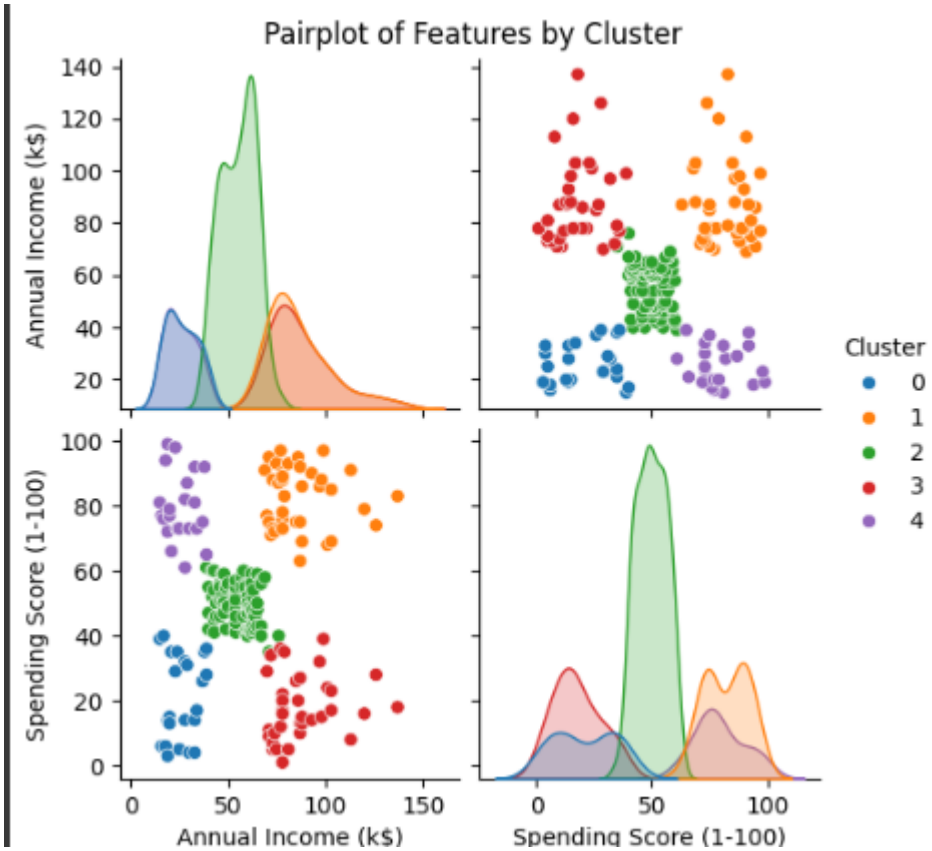


- Hierarchical clustering dendrogram

```
import pandas as pd
import matplotlib.pyplot as plt
from scipy.cluster.hierarchy import dendrogram, linkage
from sklearn.preprocessing import StandardScaler



# Select relevant features
X = df[['Annual Income (k$)', 'Spending Score (1-100)']]

# Scale the data
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)

# Compute linkage matrix
linked = linkage(X_scaled, method='ward')

# Plot dendrogram
plt.figure(figsize=(12, 6))
dendrogram(linked,
           orientation='top',
           distance_sort='descending',
           show_leaf_counts=True)
plt.title('Hierarchical Clustering Dendrogram (Ward linkage)')
plt.xlabel('Customer Index')
plt.ylabel('Distance')
plt.grid(True)
plt.show()
```



- Cluster summary tables



Conclusion: This experiment demonstrated the application of three popular clustering techniques—K-Means, DBSCAN, and Hierarchical Clustering—on unsupervised data. Each method has its strengths: K-Means is efficient for well-separated data, DBSCAN handles noise and arbitrary shapes, and Hierarchical clustering provides a tree-like structure for exploratory analysis. These clustering techniques help uncover hidden patterns and structure in data without prior labels, proving useful in areas such as market segmentation, customer profiling, and pattern recognition.

Aim: To implement recommendation system on your dataset using the following machine learning techniques.

Types of Recommendation Systems

A Recommendation System suggests relevant items to users based on their preferences, behavior, or other factors. There are several types of recommendation techniques:

🔹 **1. Content-Based Filtering**

- **Idea:** Recommends items similar to those the user has liked before.
- **Works on:** Item features (attributes such as brand, price, category).

Example:

- If a user buys a Samsung phone, they might be recommended another Samsung device based on brand preference.

- Uses techniques like TF-IDF (for text data), Cosine Similarity, Decision Trees, etc.

🔹 **2. Collaborative Filtering (CF)**

- **Idea:** Recommends items based on similar users' preferences.
- **Works on:** User interactions rather than item features.

Example:

- If User A and User B have similar purchase histories, items bought by User A but not yet by User B will be recommended to User B.

- Uses methods like User-Based CF and Item-Based CF.

🔹 3. Hybrid Recommendation System

- **Idea:** Combines Content-Based Filtering and Collaborative Filtering for better accuracy.

Example:

- Netflix uses a hybrid approach, considering both user preferences and what similar users watch.

🔹 4. Knowledge-Based Recommendation

- **Idea:** Recommends items based on explicit domain knowledge rather than past user behavior.

Example:

- A car recommendation system suggests vehicles based on engine type, price, and fuel efficiency, regardless of past purchases.

2. Recommendation System Evaluation Measures

Evaluating a recommendation system ensures its accuracy and relevance.
Below are common evaluation metrics:

🔹 1. Accuracy-Based Metrics

(a) Precision:

- Measures how many of the recommended items are actually relevant.

- **Formula:**

$$Precision = \frac{Relevant\ Recommendations}{Total\ Recommendations}$$

- **Example:**
    - If 5 out of 10 recommended items are relevant, Precision = 5/10 = 0.5 (50%).

(b) Recall:

- Measures how many of the relevant items are actually recommended.

- **Formula:**

$$Recall = \frac{Relevant\ Recommendations}{Total\ Relevant\ Items\ Available}$$

- **Example:**
    - If a user liked 8 items, but only 5 were recommended, Recall = 5/8 = 0.625 (62.5%).

(c) F1-Score:

- A balance between Precision and Recall.

- **Formula:**

$$F1 = 2 \times \frac{Precision \times Recall}{Precision + Recall}$$

- Used when both Precision and Recall are important.

(d) Accuracy:

- In classification-based recommendation systems (like Decision Trees), accuracy is measured as:
- In our Decision Tree model, if Accuracy = 1.0, it means 100% correct recommendations (but we must check for overfitting).

$$Accuracy = \frac{Correct\ Predictions}{Total\ Predictions}$$

🔹 2. Ranking-Based Metrics

These measure how well the recommendation system ranks items:

(a) Mean Average Precision (MAP):

- Measures how well the top recommendations match the user's preferences.

(b) Normalized Discounted Cumulative Gain (NDCG):

- Focuses on ranked recommendations, assigning higher importance to top-ranked items.

🔹 3. Diversity and Novelty Metrics

- **Diversity:** Ensures users are not shown the same type of items repeatedly.

Novelty: Measures if recommendations introduce new and unknown items

Implementation:

Load dataset

```
import pandas as pd

# Load main ratings
ratings = pd.read_csv('/content/sample_data/u.data', sep='\t', names=['user_id', 'item_id', 'rating', 'timestamp'])

# Load user info
users = pd.read_csv('/content/sample_data/u.user', sep='|', names=['user_id', 'age', 'gender', 'occupation', 'zip_code'])

# Load item info
movies = pd.read_csv('/content/sample_data/u.item', sep='|', encoding='latin-1', header=None)
movies = movies[[0, 1]]
movies.columns = ['item_id', 'title']

# Merge all for full dataset
data = ratings.merge(users, on='user_id').merge(movies, on='item_id')

print(data.head())
```

```
   user_id  item_id  rating  timestamp  age gender  occupation zip_code  \
0      196      242       3  881250949   49      M      writer    55105
1      186      302       3  891717742   39      F   executive    00000
2       22      377       1  878887116   25      M      writer    40206
3      244       51       2  880606923   28      M  technician    80525
4      166      346       1  886397596   47      M    educator    55113

                    title
0             Kolya (1996)
1    L.A. Confidential (1997)
2       Heavyweights (1994)
3  Legends of the Fall (1994)
4       Jackie Brown (1997)
```

Dataset Used:
MovieLens 100K dataset, which includes:

- u.user: user demographic info.
- u.item: movie metadata.
- u.data: user ratings (user ID, movie ID, rating, timestamp)

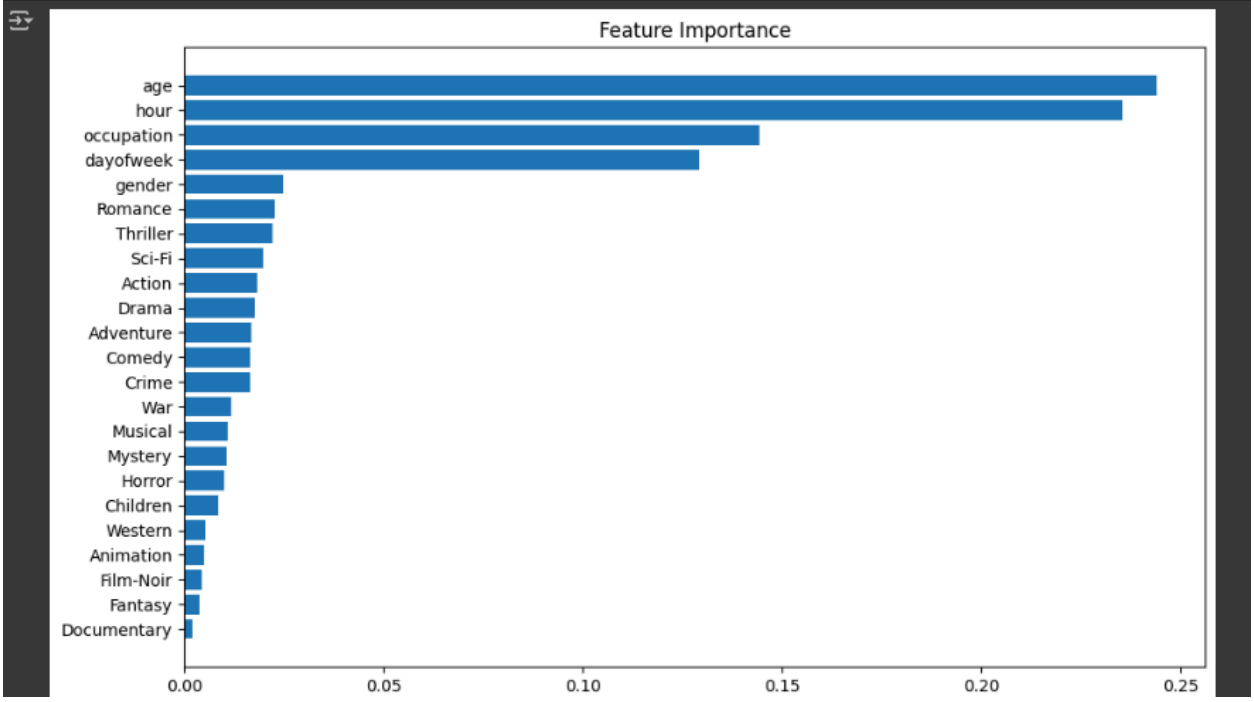
Use classification:

```
[ ]  from sklearn.model_selection import train_test_split
     from sklearn.ensemble import RandomForestClassifier
     from sklearn.metrics import accuracy_score, classification_report

     # Create time features from timestamp
     data['datetime'] = pd.to_datetime(data['timestamp'], unit='s')
     data['hour'] = data['datetime'].dt.hour
     data['dayofweek'] = data['datetime'].dt.dayofweek

     # Define feature columns
     feature_cols = ['age', 'gender', 'occupation', 'hour', 'dayofweek'] + \
                    ['Action', 'Adventure', 'Animation', 'Children', 'Comedy', 'Crime',
                     'Documentary', 'Drama', 'Fantasy', 'Film-Noir', 'Horror', 'Musical',
                     'Mystery', 'Romance', 'Sci-Fi', 'Thriller', 'War', 'Western']

     # Features and target
     X = data[feature_cols]
     y = data['liked']

     # Train-test split
     X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

     # Train model
     rf = RandomForestClassifier(n_estimators=100, random_state=42)
     rf.fit(X_train, y_train)

     # Predictions
     y_pred = rf.predict(X_test)

     # Evaluate
     print("Accuracy:", accuracy_score(y_test, y_pred))
     print("\nClassification Report:\n", classification_report(y_test, y_pred))
```

```
Accuracy: 0.6319

Classification Report:
              precision    recall  f1-score   support

           0       0.60      0.54      0.57      9010
           1       0.65      0.71      0.68     10990

    accuracy                           0.63     20000
   macro avg       0.63      0.62      0.62     20000
weighted avg       0.63      0.63      0.63     20000
```

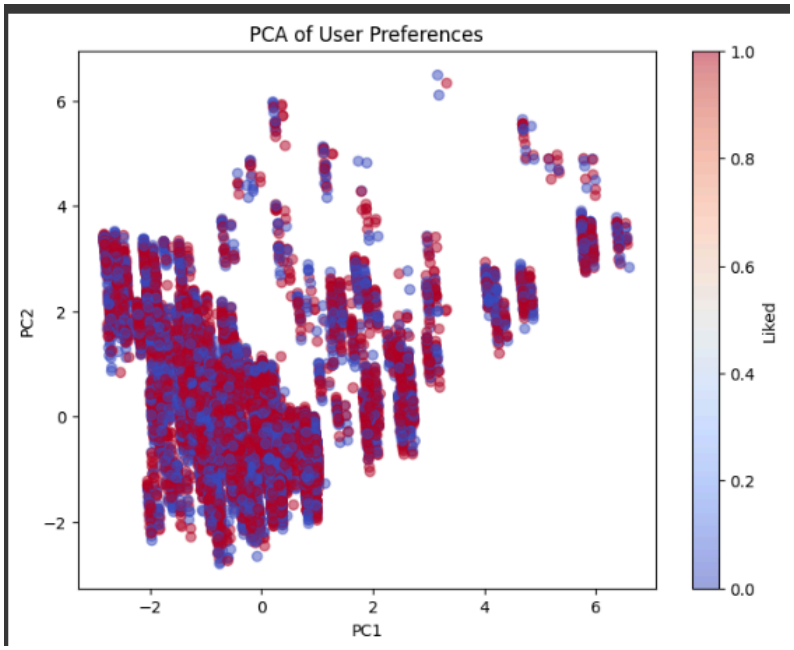
Visualization to show which attribute affects the choice most:

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

importances = rf.feature_importances_
indices = np.argsort(importances)[::-1]
features_sorted = [feature_cols[i] for i in indices]

plt.figure(figsize=(10, 6))
plt.title("Feature Importance")
plt.barh(features_sorted[::-1], importances[indices][::-1])
plt.tight_layout()
plt.show()
```



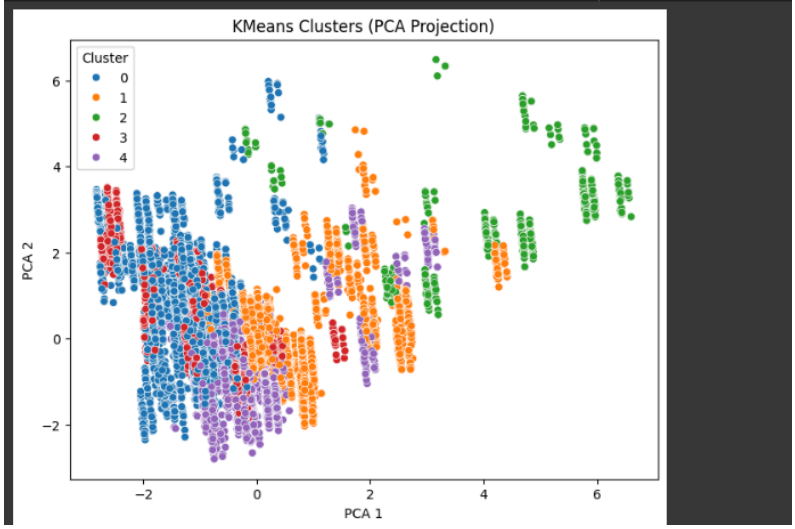
Dimensionality reduction:

```python
from sklearn.decomposition import PCA

pca = PCA(n_components=2)
reduced = pca.fit_transform(X_scaled)

# Assuming 'liked' column exists in your original DataFrame 'data'
# and you want to use the corresponding values for coloring the scatter plot
# Filter 'data' to match the size of 'reduced'
liked_filtered = data['liked'][:reduced.shape[0]]

plt.figure(figsize=(8,6))
# Use 'liked_filtered' for the color argument
plt.scatter(reduced[:, 0], reduced[:, 1], c=liked_filtered, cmap='coolwarm', alpha=0.5)
plt.title("PCA of User Preferences")
plt.xlabel("PC1")
plt.ylabel("PC2")
plt.colorbar(label='Liked')
plt.show()
```



Kmeans clustering:

```
plt.figure(figsize=(8, 6))
sns.scatterplot(x='PCA1', y='PCA2', hue='cluster', data=data_sample, palette='tab10')
plt.title('KMeans Clusters (PCA Projection)')
plt.xlabel('PCA 1')
plt.ylabel('PCA 2')
plt.legend(title='Cluster')
plt.show()
```



This output shows the **cluster-wise summary statistics** of user preferences from a movie dataset, grouped by `cluster` (which likely came from a clustering algorithm like KMeans).

```
cluster_summary = data_sample.groupby('cluster')[[
    'age', 'gender', 'occupation',
    'Action', 'Comedy', 'Drama', 'Horror', 'Sci-Fi', 'Romance'
]].mean()

cluster_summary
```

cluster	age	gender	occupation	Action	Comedy	Drama	Horror	Sci-Fi	Romance
0	31.662817	0.783605	11.558122	0.662060	0.045437	0.048466	0.094850	0.279629	0.085952
1	32.511810	0.732701	11.252162	0.058383	0.857951	0.130572	0.079009	0.030938	0.274784
2	30.165125	0.709379	11.591810	0.027741	0.376486	0.001321	0.017173	0.047556	0.019815
3	33.993222	0.760167	10.825860	0.486966	0.133994	0.550052	0.000000	0.299791	0.361314
4	34.186536	0.718123	10.559111	0.053722	0.004311	0.968994	0.012270	0.053225	0.193334

Conclusion:

In this experiment, we used K-Means Clustering to group users based on their demographics and movie genre preferences. The clustering revealed distinct patterns—some groups preferred action-packed genres like Action and Sci-Fi, while others leaned towards Drama and Romance.

This segmentation helps us understand user behavior and can be applied in areas like personalized recommendations and targeted marketing. Overall, the experiment highlights how unsupervised learning can uncover hidden insights from data and aid in effective decision-making.

Experiment No: 9

Aim: To perform Exploratory data analysis using Apache Spark and Pandas

Theory:

1. **What is Apache Spark and how does it work?**

    1. **Apache Spark** is a powerful, open-source distributed computing framework specifically built for processing large volumes of data quickly and efficiently.
    2. It enables parallel data processing across a cluster of computers, allowing it to handle massive datasets that would be too large for a single machine.
    3. Unlike traditional MapReduce in Hadoop, Spark leverages **in-memory computation**, which drastically improves execution speed for iterative algorithms and complex workflows.
    4. Spark offers high-level APIs in multiple languages such as **Python, Java, Scala, and R**, making it accessible to a wide range of developers.
    5. It consists of several integrated components, including **Spark SQL** for structured data processing, **MLlib** for machine learning, **GraphX** for graph computation, and **Spark Streaming** for real-time data streams.
    6. Its resilient distributed dataset (RDD) and DataFrame abstractions allow for both fault tolerance and flexibility in managing unstructured or structured data.
    7. Spark's execution engine dynamically optimizes queries and manages tasks across distributed systems, making it ideal for ETL jobs, interactive queries, machine learning, and more.

2. **How is data exploration done in Apache Spark? Explain steps.**

    1. Step 1: Importing Data – Spark reads data from multiple sources like CSV, JSON, Parquet, or databases and loads it into DataFrames for processing.
    2. Step 2: Data Preprocessing – Clean the dataset by managing null values, correcting column types, and resolving duplicates or errors.
    3. Step 3: Data Manipulation – Perform operations such as filtering, joining, grouping, and aggregation to derive insights.
    4. Step 4: Integration for Visualization – While Spark lacks built-in plotting tools, it can be integrated with libraries like Matplotlib, Seaborn, or Power BI for visualization.
    5. Step 5: Statistical Summary – Use Spark functions to calculate descriptive statistics (mean, std. dev, median, etc.) for a quick overview of distributions.
    6. Step 6: Sampling & Data Inspection – Use .show() or .sample() to inspect data subsets and validate assumptions before deeper analysis.

Conclusion:

Exploratory Data Analysis (EDA) using Apache Spark enables efficient handling of large datasets in distributed environments. Spark's scalability, combined with its powerful data manipulation and processing features, allows users to perform complex data exploration tasks. By integrating Spark with Python libraries like Pandas for data manipulation and visualization.

Experiment 10

Aim: To perform Batch and Streamed Data Analysis using Apache Spark.

Theory:

1) What is streaming. Explain batch and stream data.

Ans:

Streaming refers to a technique in data processing where information is generated, transferred, and analyzed on a continuous basis in real-time or with minimal delay. It is particularly useful in scenarios that demand instant feedback or actions—such as live video streams, banking transactions, or IoT-based sensor data.

Rather than waiting for an entire dataset to be collected, streaming systems process data incrementally as it flows in, allowing for immediate analysis and quicker decision-making.

Feature	Batch Processing	Stream Processing
Data Processing	Processes a large volume of data at once.	Processes data as it arrives, record by record.
Latency	High latency, as processing happens after data collection.	Low latency, providing near real-time insights.
Throughput	Can handle vast amounts of data at once.	Optimized for real-time but might handle less data volume at a given time.
Use Case	Ideal for historical analysis or large-scale data transformations.	Best for real-time analytics, monitoring, and alerts.
Complexity	Relatively simpler to implement with predefined datasets.	More complex, requires handling continuous streams.
Data Scope	Operates on a finite set of data.	Operates on potentially infinite streams of data.
Error Handling	Errors can be identified and corrected before execution.	Requires real-time handling of errors and failures.
Resource Usage	Resource-intensive during processing, idle otherwise.	Continuous use of resources.
Cost	Cost-effective for large volumes of data.	More expensive due to continuous processing.

2) How data streaming takes place using Apache spark.

Ans:

Apache Spark includes a robust module known as **Spark Structured Streaming**, designed to process data streams in real-time. It enables seamless and continuous handling of incoming data, blending the ease of using SQL or DataFrame queries with Spark's ability to scale and recover from faults efficiently.

Core Components and Workflow

1. Data Input Sources

The streaming workflow starts with ingesting data from various sources. Spark continuously

monitors and reads data from real-time input channels, such as:

- Apache Kafka
- File systems (by tracking newly added files)
- Network sockets
- Amazon Kinesis
- Custom data sources

These sources deliver data in a live stream, which Spark captures and processes on the fly.

1. Streaming Data as a Table

Spark conceptualizes real-time data as a never-ending or *unbounded* table, where each incoming record is treated as a newly inserted row. This structure allows users to perform familiar operations—such as select, filter, groupBy, and complex SQL queries—on data that is constantly evolving.

2. Query Execution

Users define logical queries on streaming data, like calculating averages or tracking counts. Under the hood, Spark converts these logical instructions into an optimized physical execution plan. This layered planning ensures efficient and scalable processing, even with massive, real-time workloads.

3. Micro-Batch Processing

Instead of processing each data point as it arrives, Spark Structured Streaming groups incoming records into short-duration intervals called *micro-batches*. For instance, it might process new data every second. This batching strategy strikes a balance between real-time responsiveness and high-throughput performance, making it well-suited for both latency-sensitive and large-scale scenarios.

4. Output Sink

After processing, the results are written to an output sink, such as:

- Console (for testing/debugging)
- Kafka
- Databases
- File systems

You can choose different output modes:

- Append: Only new rows are written.
- Update: Only updated rows are written.
- Complete: The entire result table is written.
- Fault Tolerance

Conclusion:

Batch and streaming data processing represent two fundamental paradigms in the world of analytics. **Batch processing** deals with static datasets gathered over time and is excellent for deep analysis and complex transformations. In contrast, **streaming analytics** enables immediate processing of real-time data, which is essential for scenarios like fraud detection, live dashboards, or real-time alerts.

Modern data platforms often combine both strategies to form **hybrid architectures**—taking advantage of batch processing for in-depth insights and streaming for timely, reactive decision

AIDs

What is AI? Considering the covid 19 pandemic situation how AI helped to survive and renoated our way of life with different applications?

→ Artificial Intelligence (AI) is a branch of computer science that focused on creating intelligent systems capable of performing tasks that typically decision making learning, perception, etc.

Role in surviving & renovating life during Covid-19

i Heathcare & medical diagnosis
CT Scan analysis & x-ray diagnosis helped in rapid detection of cancil cases.

ii Virus spread control
Social distance montioring tools helped to monitor & enforce laws in public places.

iii) remote work & education
work from home optimization & AI in online edu. help people continue with this work & education.

iv) Fake news detection
AI helped to identify rapidly spreading fake news on social media.

2. What are AI agents terminology ? Explain with example environment.
Everything which surroands the agen & influence its actions
It can be complete on partially observable

eg. rubics cube for solution it.

- percepts

It is a raw data that an agent gets from its sensor.

eg. self driving cars

Actuators:

They are the components that allow an agent to take actions in the env.

eg. A robotic arm uses motors as actuators

Goal

The final state of an agent to achieve is called as goal.

eg. Solving the rubics cube.

3. How is AI-technique used to solve-8-puzzle problems?

The 8 puzzle problems

The 8 puzzle problem is a state space search problem in AI where a 3×3 grids contains 8 tiles numbered from 1 to 8 1 empty space.

Objective is to rearrange the tiles to reach a predefined goal state.

techinques 1) Uniformed search methods

- BFS : Expand the shallowest node first
- DFS : Explores as deep as possible before backtrack
- IDS : Combines DFS & BFS to increase depth limit gradually.

2. Informed Search methods

- BFS - Best first search based on heuristic funce that appears closest to the goal.

A^* search $f(n) = g(n) + h(n)$ - Based on heuristic & cost to node.

Initial State: 1 2 3 / 5 6 0 / 4 7 8 Goal state: 1 2 3 / 4 5 6 / 7 8 0

i) Compute heuristic of each possible move

ii) expand the state with the lowest f(n) & repeat

4. What is PEAS description? give PEAS description to, following?

Performance Measure: How success of agent is evaluated

environment Surrounding in which agent appears.

Activators - Component that allows agent to take action

Sensor - Component that allows agent to percive the environment.

i) Taxi driver agent -

Performance measure	Enviroment	Activators	Sessor
- Safe driving	- Traffic signal	- Stering wheel	- Camera
- travel time	- roads	- accelarator	- GPS
- traffic rules	- weather	- brake	- fuel gauge

ii) Medical Dignosis agent

Perfo. Measure	Enviroment	Activator	Sensor
- health of Patient	Patien data	display screen	heart rate

accuracy of diagnosis	Symptoms	~~dia~~ alarm sys	heart rate monitor
recomended treatment	test repo.	roboticarms	lab results

iii) Music Comporent Agent:

Perf. Measure	Enviroment	Activator	Sensor
originality	Music db	speaker	
listener engagment	user perfor.		

iv) Aircraft autolander

Perfo. measure	environment	Actuator	Sensor
Smooth landing	runway	landing gear	GPS
accuracy in	wind condition	flap	Camera
touchdown	air traffic	air brakes	

v) ~~EA~~ Essay evaluator Enviroment Actuator Sensor

prof. masure	plagrism	display screen	optical char.
grading	rubric	text to	recognition
grammer	criteria	speech	OCR(R)
paradigm check			

vi) Robotic seq. gun

per. measure	Environment	Actuator	Sensor
neutralize threats	lab area	gun mech-	Camera
target talking	potential	anism	thermal
false alarm	intruders	alarm	sensor

5. Categorize a shopping bot for a shopping bot for as offline books store according to following system.

→ observability : partially observable relies on limited sensor input

- Deterministic or stochastic - stochastic sensor are is unpredicted
- Episodic vs sequential : sequential affect future actions.
- static vs dynamic - Dynamic customer behaviour is always
- Discrete vs continuous no of choices such as

6. Differentiate between model based & utility based agent

	Model based agent	Utility based agent
-	agent that maintains the internal model of the env. to understand its current state & predict future states	Agent that selects actions based on the utility functions aiming to max long term benefit.
-	Model updates its info about the env.	- Measures how desirable diffrent states are
-	less complex	- More complex
-	Doesn't concern long term rewards	- Focuses on long term rewards.
-	eg. Self driving cars	- shopping reco. System

7 Explain architecture of knowledge based ~~sy~~ agent & learning agent.

-> Knowledge based agents - stores knowledge & reasons based on logical interference

Knowledge based - stores facts, rules and heuristics function about the environment

- Interference engine: Uses logical reasoning techniques like forward & backward chaining.
- Perception - gathers data from the env.
- actions executes actions based on ~~inter~~ infered knowledge
- Knowledge update mechanism - updates itself as new facts are learned.
- learning based agent - Agent that improves its port overtime by learning from experience, data & feedback
- learning element - responsible for improving agents pref by analyzing past experience using ML techniques.
- critic: Provides feedback on agents actions by evaluating success or failure.
- Problem generation - supports new exprience for learning & explaning.

8 Convert the following to predicates

Anita travels by car if available otherwise travells by bus.
travels (x, y) → person travel y by y.
Available (y) → y (a vehicle) is available
goesvia (y, z) → vehicle goes via z
punchure (y) → y (a vehicle)

a Available (car) → Travels (Anita, bus)
b. Bus goes via andheri & Goregaon
Goes Via (bus, andheri)
Goes Via (bus, goregaon)
c. Car has a puncture so its not available
puncture (car)
~ Available (car)

9. (what do you mean by depth limit search)
Explain iterative Deepening search with eg.
Depth limit Search (DLS)
DLS is a Depth-limited search DFS variant with a fixed depth L. preventing infinte loop & saving memory

Adv.
1) Avoids infinite recursion
2) Memory effective

Disadv.
1) Many miss less deper solutions
2) need for good L choice (Limit)

eg: Searches level by level until the goal appears

b) Explain Hill climbing & its drawback in detail with eg. Also State limitations of steepest ascent hill climbing

→ It is an optimization algorithm that moves towards higher values (better solutions) until a peak (local - optimum) is reached.

Algorithm

1. Start with an initial state
2. Move to the best neighbouring State.
3. Repeat ~~until~~ no strictly better neighbour exists

eg: Adjust queens position to minimize conflicts
Stop when no improvements are possible

Drawbacks

local maxima stuck at subpoint peaks
Plateue - no directions for improvement
Ridges needs special move to progress.
Variations & solutions
steepest - Ascent - Evaluates all neighbour

4. Explain simulated Annealing & write its

→ SA improves Hill climbing by allowing occasionally bad moves to escape local maxima inspired by metal aneling.

Algorithm :

1. Start with an initial solution & temperature T
2. Accept S is better, otherwise with probability
   $P = e^{(-\Delta S / T)}$
3. Reduce T until stopping condition

Adv

1) Escapes local minima
2) Handles large problems
3) Near - optimal solutions

Disadvantage

Tuning cooling schedule
No guarantee of best solution

12. Explain A* algorithm

A* is a best-first search Algo for pathfinding combining

1) Uniform cost search (cheapest path)
2) Greedy best-first search (heuristic-based speed)

Key formula

$f(n) = g(n) + h(n)$

$g(n)$: Cost from start to n

$h(n)$ = Cost from n to goal state.

Steps :

1. Start with the initial node, compute $f(n)$
2. Expand if reached, return the path, use update & continue.

Adv.

+ Optimal paths 2) efficient in AI applications

Dis advantage

1. High ~~Speed~~ memory usage
2. Depends on heuristics.

13. Explain Min Max Algo & Draw game tree from Tic tac toe game.

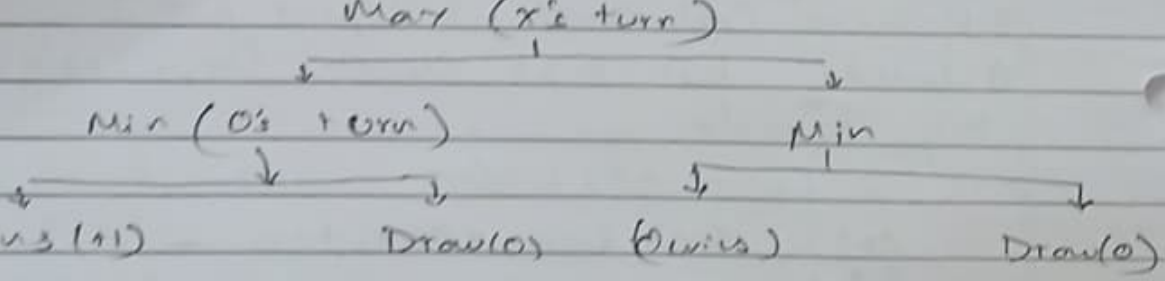
Min Max is a game strategy for 2 player games like Tic-Tac-Toe.

How it works?

Maximize(X) aims to increase the score (+1 for win.

Minimize(O) aims for lowest score (-1 for loss.

Explore all possible moves as signs score & picking the best one.



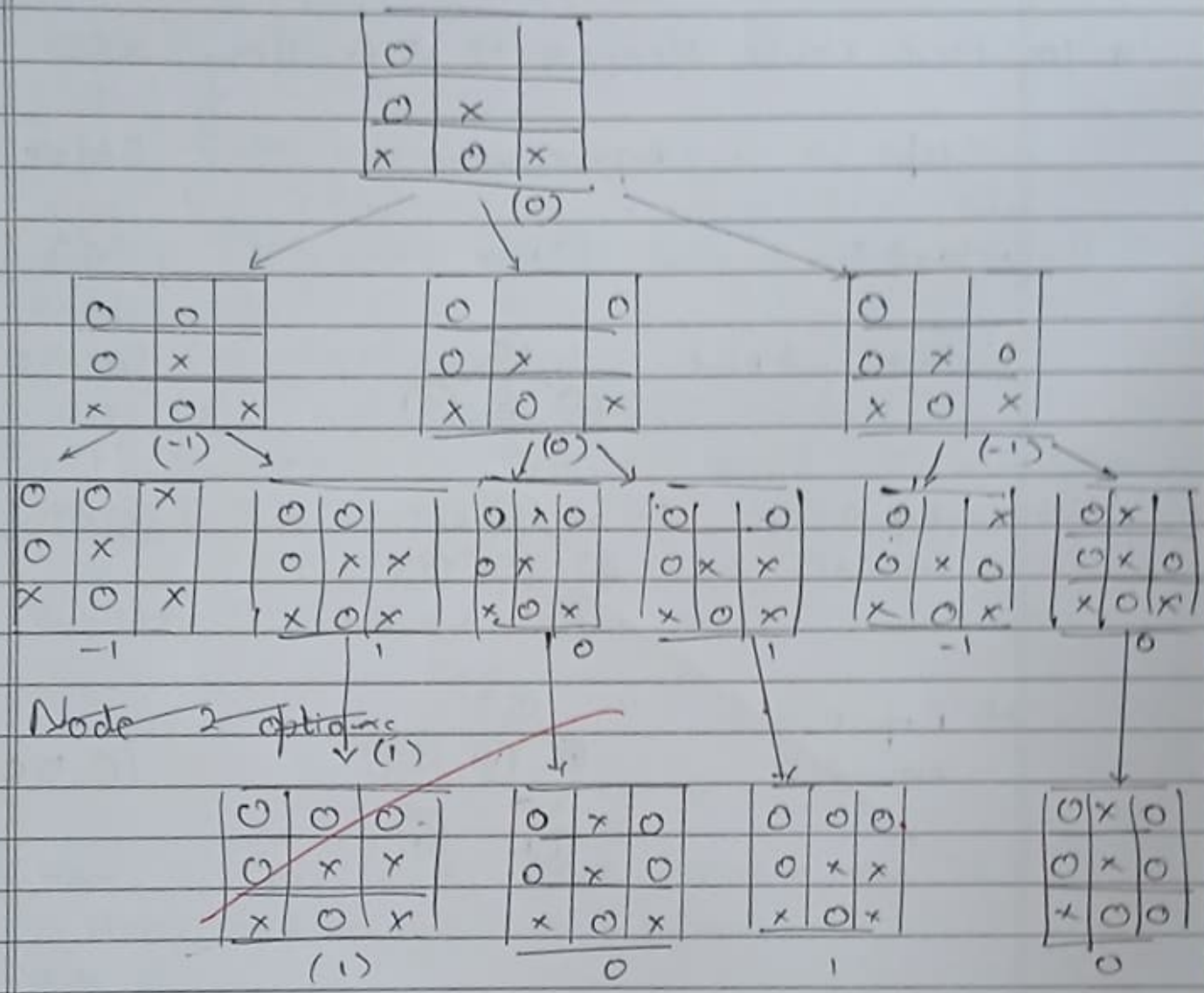
Each level alternates between X & O i.e. Max and Min

Adv

1) Always find the best moves

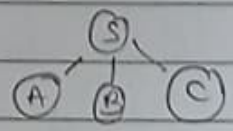
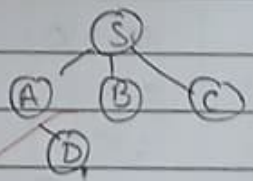
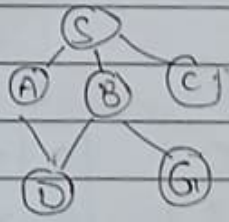
Dis adv

1. Slow for deep tree (Alpha beta pruning)



2 Node options has the most optimal Choice

14 10 Find route from S to G using BFS

Steps	Representation	Stack
i load S		[S]
ii Pops and A,B,C		[A, B, C]
iii) Pop A Expand D		[B, C, D]
iv) Pop B expand G		[C, D, G]

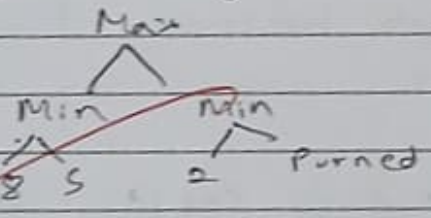
15) Explain alpha beta pruning for adversial search with eg.

→ Alpha beta pruning optimizes the min Max algo by skipping unecessary branches making it faster without affecting the result.

Explanation-

1. Alpha (α) best Max value found so far.
2. Beta (β) : min value - found so far.
3. If a mov wels then α or β is found then the further exploration is stopped, (purned)

Example (simplified game tree):



Here if min finds move worse than 5 it stops exploring that branch

Adv.

1. Speeds up minimax by ignoring bad choices
2. Same result as minmax but faster.

16) Explain Wumpus world env. giving its PEAS description. Explain how percept sequence is generated

→ Wumpus World is a grid based game env. where an agent navigates a cave to find gold while avoiding pits and the wumpus monster

PEAS description:

Performance Measure - +1000 (Gold), -1000 -100 (Pit) -1 (Move)

Environment - grid world with wumpus gold Pits & agents.

A (Actuators) - Move, Grab (Gold), Shoot, Climb

S (Sensor) - a) Breeze near pit b) stench (near wumpus) c) Glitter (near gold)

Percept Sequence generator -

The agent receives sensory input at each step based on its current location.

eg. If agent move next to pit, it perceives 'Breeze'. Using percept history it infers safe path & avoid danger.

17 Solve the following crypto arithmetic problems

SEND + MORE = MONEY

Each letter represents a unique digit (0-9)

Step1 - Evaluation

((1000 S) + 100E + 10N + D) + (1000M + 100O + 10R + E) = (10000M + 1000O + 100N + 10E + Y)

Step2: Constraints

M = 1 (since MONEY is a 5 digit number S ≠ 0 as it is the first digit in SEND)

All letters have unique values.

Step3: Assigning Digits

letter	Digit
S	9
E	5
N	6
D	7
M	1
O	0
R	8
Y	2

18) Consider the following axioms
All people who are graduating are happy
All happy people are Smiling
Someone is graduating.
Explain the following:

1) Represents these axioms in first order predicate
2) Convert each formula to clause form
3) Prove that " is Someone Smiling " using resolution technique - Draw the resolution tree

1) FOPL:

let G(n) → n is graduating, H(x) → x is happy, S(x) → x is smiling

Axioms: (G(n) → H(n))

1. ∀ n (H(n) → S(n))
3. ∃ x G(n)

2 Convert to clause form

¬ G(n) ∨ H(n)

¬ H(n) ∨ S(x)

G(A) let A be a person graduating

3 Prove it Sara are smiling?

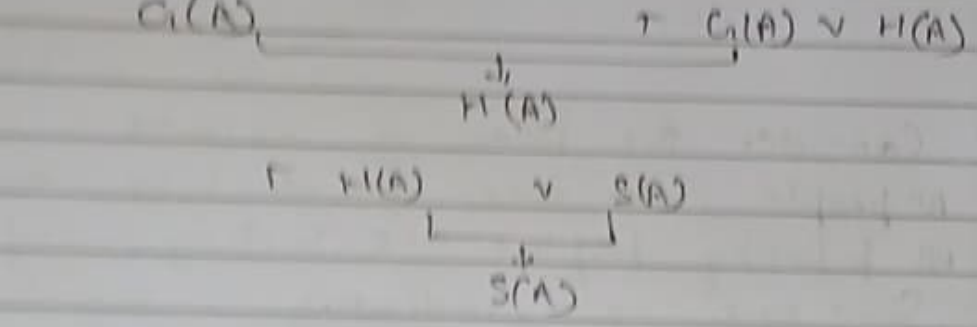
1 G(A) given

2 ¬ G(A) ∨ H(A) → axiom 1

3 ¬ H(A) ∨ S(A) → axiom 2

Since we derived S(A) the proof confirms that someone is smiling

Resolution tree



11 15 Explain Modus Ponen with suitable example

Modus Ponen is a fundamental rule of inference in logic it states.

If P→Q (if P, then Q) is true.

& P is true, then Q must be true

Symbolically

P→Q P ⊢ Q

eg If it rains ground will be wet

It is raining (P)

∴ here, the ground is wet

This rule is widely used in mathematical proofs & AI reasoning system.

20. Explain forward & Backward chaining with eg. These are inference techniques used in rule based system & AI reasoning.

1. Forward chaining (Data. Driven):
   Starts with known facts & applies rules to in per new facts until the goal is reacted
   Works from Cause → effect (bottom up)

eg. 1. If it is raining then the ground is wet.
2. If ground is wet then traffic is slow

2. Backwards chaining (Goal driven)
   Starts with the (goal driven)
   support facts
   Works from effect to cause (top-down)
   W → R is the ground wet
   R → W is it raing

Name:Laukik Padgaonkar Div:D15C Roll:37

AIDS-I Assignment No: 2

Q.1: Use the following data set for question 1

82, 66, 70, 59, 90, 78, 76, 95, 99, 84, 88, 76, 82, 81, 91, 64, 79, 76, 85, 90

1. Find the Mean (10pts)

   To calculate the mean, first add up all the numbers in the dataset and then divide by the total number of values.

   Sum = 82 + 66 + 70 + 59 + 90 + 78 + 76 + 95 + 99 + 84 + 88 + 76 + 82 + 81 + 91 + 64 + 79 + 76 + 85 + 90 = 1691

   Count = 20

   Mean = Sum / Count = 1691 / 20 = 84.55

   Therefore, the mean of the dataset is 84.55.

2. Find the Median (10pts)

   To find the median, I first need to arrange the data in ascending order:

   59, 64, 66, 70, 76, 76, 76, 78, 79, 81, 82, 82, 84, 85, 88, 90, 90, 91, 95, 99

   Since there are 20 values (an even number), the median will be the average of the two middle values, which are the 10th and 11th values in the ordered list.

   Middle values: 81 and 82

   Median = (81 + 82) / 2 = 81.5

   Thus, the median of the dataset is 81.5.

3. Find the Mode (10pts)

   The mode is the value that appears most frequently in the dataset. Looking at the ordered list:

59, 64, 66, 70, 76, 76, 76, 78, 79, 81, 82, 82, 84, 85, 88, 90, 90, 91, 95, 99

The number 76 appears three times, which is more than any other number.

Therefore, the mode of the dataset is 76.

4. Find the Interquartile Range (20pts)

The interquartile range (IQR) is the difference between the third quartile (Q3) and the first quartile (Q1).

First, I need to find Q1 and Q3.

Ordered list: 59, 64, 66, 70, 76, 76, 76, 78, 79, 81, 82, 82, 84, 85, 88, 90, 90, 91, 95, 99

Q1 is the median of the lower half of the data. Since there are 20 values, the lower half is the first 10 values. The median of the lower half is the average of the 5th and 6th values:

Lower half: 59, 64, 66, 70, **76, 76**, 76, 78, 79, 81

Q1 = (76 + 76) / 2 = 76

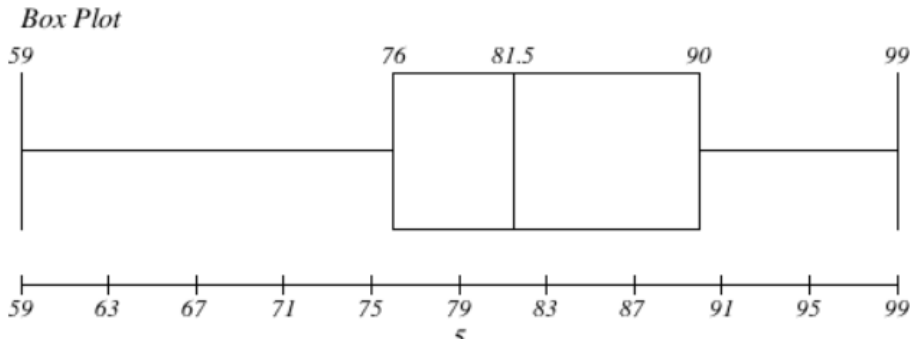
Q3 is the median of the upper half of the data. The upper half is the last 10 values. The median of the upper half is the average of the 15th and 16th values:

Upper half: 82, 82, 84, 85, 88, **90, 90**, 91, 95, 99

Q3 = (90 + 90) / 2 = 90

IQR = Q3 - Q1 = 90 - 76 = 14

Thus, the interquartile range of the dataset is 14.



Q.2 1) Machine Learning for Kids 2) Teachable Machine

1. For each tool listed above:
    - Identify the target audience
    - Discuss the use of this tool by the target audience
    - Identify the tool's benefits and drawbacks
2. **1) Machine Learning for Kids**
    - **Target Audience:** The primary target audience for Machine Learning for Kids is, as the name suggests, children and educators who want to introduce machine learning concepts in an accessible way. It's designed for beginners with little to no prior coding or machine learning knowledge.
    - **Use of the Tool:** This tool allows children to train machine learning models using simple, child-friendly interfaces. They can train models to recognize text, images, sounds, or numbers. The tool then lets them use their trained models in Scratch or Python projects, enabling them to create interactive games or applications that respond to the inputs they've taught the model to recognize. This hands-on approach helps children understand how machine learning works by doing it themselves.
    - **Benefits:**
        - **Ease of Use:** The tool simplifies complex machine learning concepts, making them understandable for children.
        - **Engaging Interface:** Using Scratch and Python integration makes learning fun and interactive.
        - **Educational Value:** It provides a practical introduction to AI and machine learning, fostering computational thinking skills.
    - **Drawbacks:**
        - **Limited Complexity:** Due to its simplicity, it may not be suitable for advanced machine learning projects.

- **Dependency on External Platforms:** Relies on Scratch or Python for project integration, which might require some additional setup.

3. **2) Teachable Machine**

   - **Target Audience:** Teachable Machine is geared towards a broader audience, including students, artists, educators, and makers. It's designed for anyone who wants to quickly and easily create machine learning models without writing code.
   - **Use of the Tool:** Teachable Machine simplifies the process of training machine learning models for image, audio, and pose recognition. Users can train models directly in their browser by providing examples through a webcam, microphone, or file uploads. The trained models can then be exported and used in various applications, websites, or projects. This tool is excellent for rapid prototyping and integrating machine learning into creative projects.
   - **Benefits:**
     - **No Coding Required:** It's very accessible, as it doesn't require any coding knowledge.
     - **Fast Prototyping:** Allows for quick creation and testing of machine learning models.
     - **Versatile Export Options:** Models can be exported in various formats, making them usable in different environments.
   - **Drawbacks:**
     - **Limited Control:** Offers less fine-grained control over the model training process compared to more advanced tools.
     - **Browser-Based Limitations:** Performance and capabilities can be limited by the browser's capabilities.

4. From the two choices listed below, how would you describe each tool listed above? Why did you choose the answer?

   - Predictive analytic
   - Descriptive analytic

5. **1) Machine Learning for Kids:** Predictive analytic

    - **Why?** Machine Learning for Kids enables users to train models that can make predictions or classifications based on the input data. For instance, a child can train a model to predict whether an image is a cat or a dog. This is a clear example of predictive analytics, as the model is used to predict a future outcome or classify an input.

6. **2) Teachable Machine:** Predictive analytic

    - **Why?** Teachable Machine is also a predictive analytic tool. It allows users to train models that predict outputs (like image, audio, or pose classifications) from new input data. The core function is to predict or classify, which aligns with the definition of predictive analytics.

7. From the three choices listed below, how would you describe each tool listed above? Why did you choose the answer?

    - Supervised learning
    - Unsupervised learning
    - Reinforcement learning

8. **1) Machine Learning for Kids:** Supervised learning

    - **Why?** In Machine Learning for Kids, children provide labeled examples to train the models. For example, they show the model pictures labeled "cat" or "dog." The model learns from these labeled examples to make predictions on new, unseen images. This process of learning from labeled data is the fundamental characteristic of supervised learning.

9. **2) Teachable Machine:** Supervised learning

    - **Why?** Teachable Machine also operates on the principle of supervised learning. Users provide labeled data (e.g., images labeled with categories) to train the model. The model then learns to map the input data to the provided labels, enabling it to classify new inputs. The need for labeled data to train the model classifies it as supervised learning.

Q.3 Data Visualization: Read the following two short articles:

Read the article Kakande, Arthur. February 12. "What's in a chart? A Step-by-Step guide to Identifying Misinformation in Data Visualization." Medium

Read the short web page Foley, Katherine Ellen. June 25, 2020. "How bad Covid-19 data visualizations mislead the public." Quartz

Research a current event which highlights the results of misinformation based on data visualization. Explain how the data visualization method failed in presenting accurate information. Use newspaper articles, magazines, online news websites or any other legitimate and valid source to cite this example. Cite the news source that you found.

Current Event Highlighting Misinformation Through Data Visualization: Misrepresented NOAA Temperature Graph

Event Overview

A graph sourced from the National Oceanic and Atmospheric Administration (NOAA) was selectively cropped and presented on social media to falsely claim that the Earth has been experiencing a cooling trend, contradicting the established scientific consensus on human-caused global warming.1 The graph focused solely on temperature data from 2015 to 2022, omitting long-term historical data that clearly shows a warming trend. This misrepresentation was fact-checked by the Associated Press (AP).

How the Data Visualization Method Failed

- **Cherry-Picking Data:** The visualization deliberately showcased only a limited timeframe (2015-2022), which was chosen to exploit natural climate variability (El Niño and La Niña events) and create a false impression of cooling. This selective presentation ignored the broader 140-year temperature record, a clear example of cherry-picking data to support a predetermined narrative.

- **Lack of Context:** The graph failed to provide essential context by omitting the long-term temperature data from NOAA. This omission prevented viewers from understanding the true extent of global warming and the significance of the short-term fluctuations shown in the graph. Legitimate data visualizations should always include sufficient context to ensure accurate interpretation.

- **Misleading Trendline:** A black line was added to the graph to emphasize a minor downward trend within the selected timeframe. This visual element drew attention to a statistically insignificant fluctuation, further reinforcing the false narrative of global cooling and obscuring the overall warming trend.

- **Exploitation of Authority:** The NOAA logo was prominently displayed on the graph, lending a false sense of authority and scientific validity to the misleading data. This tactic exploited the credibility of a reputable scientific institution to promote misinformation.

Impact of the Misinformation

- The misleading visualization contributed to the spread of climate change denial, undermining public understanding of the severity and urgency of global warming.
- It fostered skepticism towards climate science and reputable scientific institutions like NOAA, potentially eroding public trust in evidence-based information.
- The misrepresentation could diminish public support for policies and actions aimed at mitigating climate change.

Lessons for Ethical Data Visualization

- **Provide Complete Context:** Always include sufficient background information and long-term data to ensure accurate interpretation.
- **Avoid Cherry-Picking:** Present a comprehensive view of the data, avoiding selective presentation that supports a specific narrative.[2]
- **Ensure Data Accuracy:** Verify the accuracy and reliability of the data sources and methodologies used.
- **Transparency:** Clearly label and explain any data manipulations or selections.

News Source

- Associated Press (AP) Fact Check: "Temperature graph misrepresented to deny climate change," authored by Sophia Tulp, published on January 19, 2023.
- Link: https://apnews.com/article/fact-check-misleading-climate-change-graph-418146648172

Q. 4 Train Classification Model and visualize the prediction performance of trained model

Required information

Data File: Classification data.csv

Class Label: Last Column

Use any Machine Learning model (SVM, Naïve Base Classifier)

Requirements to satisfy

Programming Language: Python

Class imbalance should be resolved

Data Pre-processing must be used

Hyper parameter tuning must be used

Train, Validation and Test Split should be 70/20/10

Train and Test split must be randomly done

Classification Accuracy should be maximized

Use any Python library to present the accuracy measures of trained model

Pima Indians Diabetes Database

My Google Colab Link:

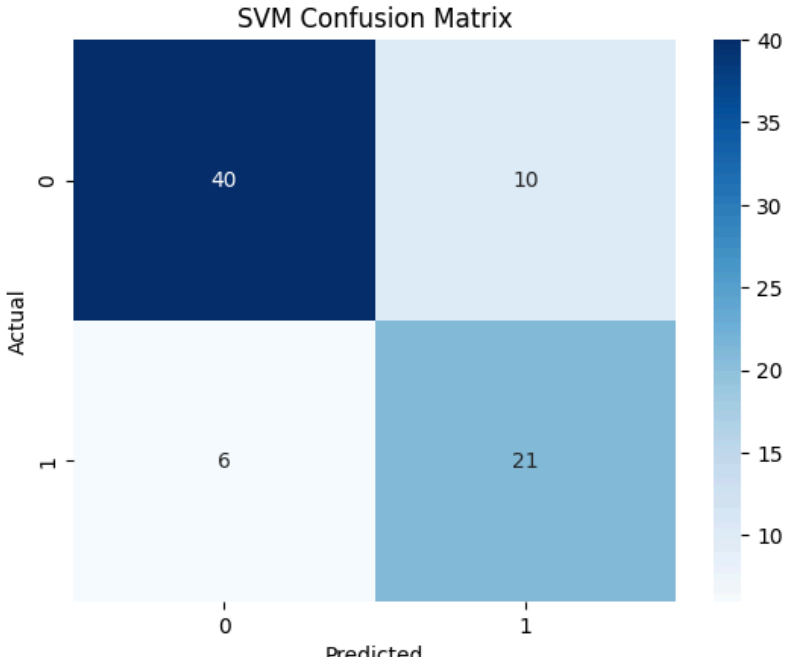
https://colab.research.google.com/drive/18TiTzAMyqyhhX-IBCBx89dH9kYWcRVSk?usp=sharing

Explanation:

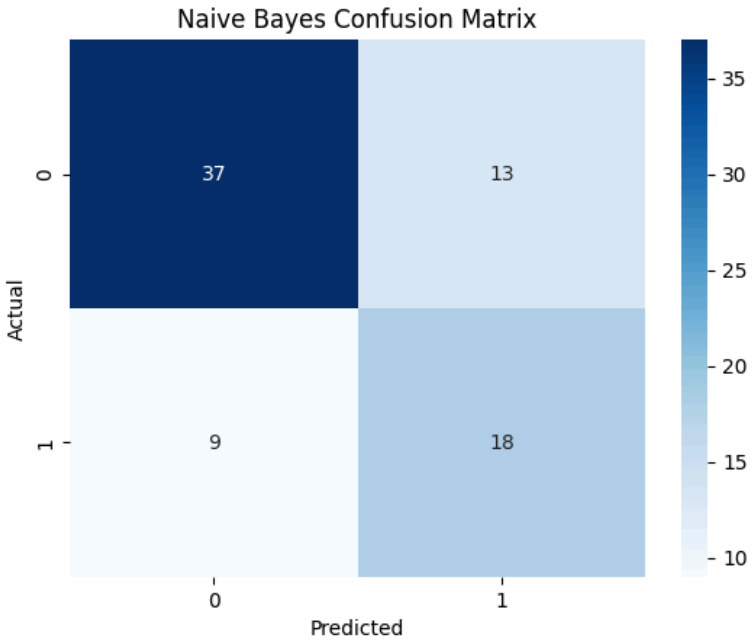
1. **Import Libraries:** Necessary libraries like numpy, pandas, sklearn, and matplotlib are imported.
2. **Load Data:** The "diabetes.csv" file is loaded using pandas.
3. **Data Separation:** The features (X) and the target variable (y) are separated. It's assumed the last column is the class label.
4. **Class Imbalance Resolution:** SMOTE (Synthetic Minority Over-sampling Technique) is used to handle class imbalance. This generates synthetic samples for the minority class.
5. **Data Pre-processing:** StandardScaler is applied to scale the features. This is crucial for models like SVM and Naive Bayes.
6. **Train-Validation-Test Split:** The data is split into 70% train, 20% validation, and 10% test sets, with stratification to maintain class ratios.
7. **Hyperparameter Tuning:** GridSearchCV is used with StratifiedKFold cross-validation to find the best hyperparameters for the Gaussian Naive Bayes model.
8. **Model Evaluation:** The model's performance is evaluated on the test set, and a classification report and confusion matrix are printed.

9. **Visualization:** The confusion matrix is visualized using Seaborn and Matplotlib.

```
--- SVM Validation Report ---
              precision    recall  f1-score   support

           0       0.81      0.73      0.77       100
           1       0.58      0.69      0.63        54

    accuracy                           0.71       154
   macro avg       0.69      0.71      0.70       154
weighted avg       0.73      0.71      0.72       154

--- SVM Test Report ---
              precision    recall  f1-score   support

           0       0.87      0.80      0.83        50
           1       0.68      0.78      0.72        27

    accuracy                           0.79        77
   macro avg       0.77      0.79      0.78        77
weighted avg       0.80      0.79      0.80        77
```




```
--- Naive Bayes Validation Report ---
              precision    recall  f1-score   support

           0       0.86      0.77      0.81       100
           1       0.64      0.76      0.69        54

    accuracy                           0.77       154
   macro avg       0.75      0.76      0.75       154
weighted avg       0.78      0.77      0.77       154

--- Naive Bayes Test Report ---
              precision    recall  f1-score   support

           0       0.80      0.74      0.77        50
           1       0.58      0.67      0.62        27

    accuracy                           0.71        77
   macro avg       0.69      0.70      0.70        77
weighted avg       0.73      0.71      0.72        77
```



Q.5 Train Regression Model and visualize the prediction performance of trained model

Independent Variable: 1st Column

Dependent variables: Column 2 to 5

Name:Laukik Padgaonkar Div:D15C Roll:37

Use any Regression model to predict the values of all Dependent variables using values of Ist column.

Requirements to satisfy:

Programming Language: Python

OOP approach must be followed

Hyper parameter tuning must be used

Train and Test Split should be 70/30

Train and Test split must be randomly done

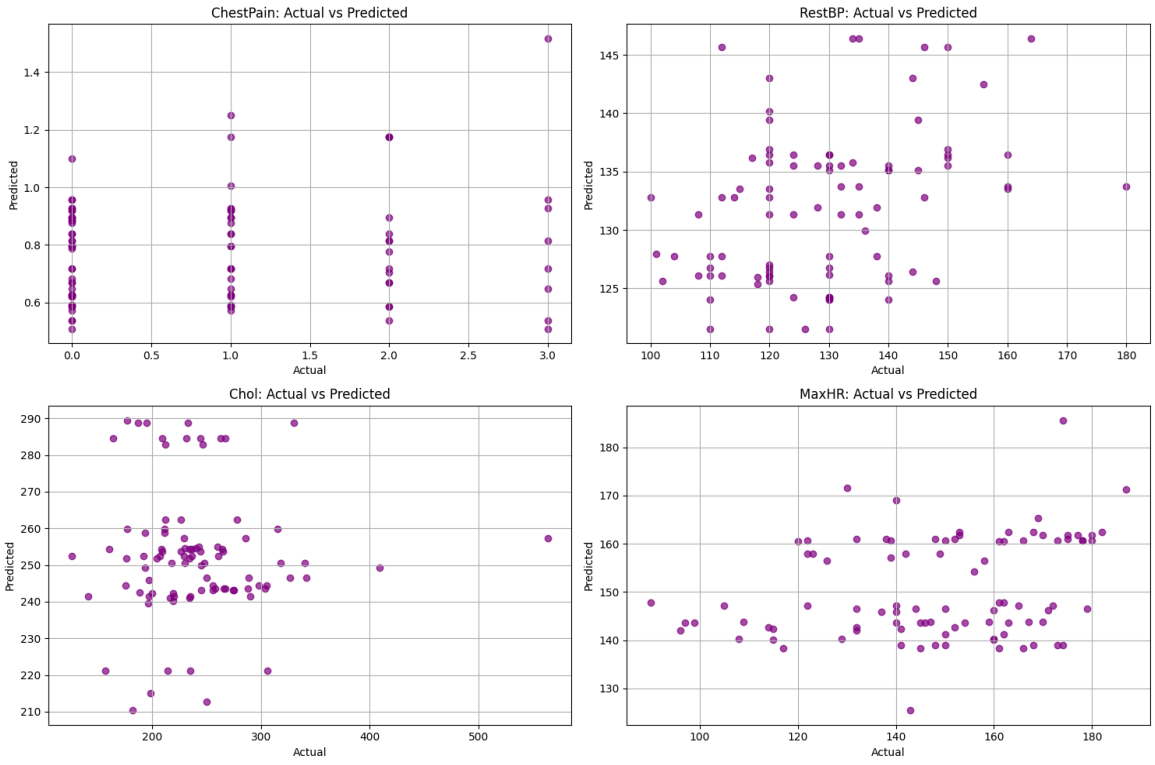
Adjusted R2 score should more than 0.99

```
  Adjusted R²      : -0.0056

RestBP:
  R² Score         : 0.0992
  Adjusted R²      : 0.0890

Chol:
  R² Score         : -0.1291
  Adjusted R²      : -0.1418

MaxHR:
  R² Score         : 0.0389
  Adjusted R²      : 0.0281
```

Use any Python library to present the accuracy measures of trained model

My Google Colab Link:

https://colab.research.google.com/drive/1bINoVgtqFGwcLdwQGLS3Vw_tZv3IDRYs?usp=sharing



Q.6 What are the key features of the wine quality data set? Discuss the importance of each feature in predicting the quality of wine? How did you handle missing data in the wine quality data set during the feature engineering process? Discuss the advantages and disadvantages of different imputation techniques. (Refer dataset from Kaggle).

The wine quality data set from Kaggle primarily contains physicochemical measurements of wines and a quality score that reflects sensory evaluation. The key features (predictor variables) include:

• **Fixed Acidity:** Represents non-volatile acids that contribute to the wine's sour taste; it affects balance and structure.
• **Volatile Acidity:** Measures the presence of acetic acid; high levels can impart an unpleasant vinegar taste, negatively impacting quality.
• **Citric Acid:** Adds freshness and helps balance the wine's overall acidity; moderate amounts can enhance flavor.
• **Residual Sugar:** Indicates the unfermented sugar left in the wine; it plays a role in sweetness and overall flavor harmony, especially in white wines.
• **Chlorides:** Reflects the salt content; excessive levels can indicate poor sanitation or imbalance, thus reducing quality.
• **Free Sulfur Dioxide:** Acts as an antimicrobial and antioxidant; helps preserve wine flavor and stability, though excessive amounts may create off flavors.
• **Total Sulfur Dioxide:** Represents the overall amount used for preservation; important for shelf life but must be within safe sensory thresholds.
• **Density:** Closely related to the sugar content; contributes to the body and mouthfeel of the wine.
• **pH:** Provides an indication of wine acidity; optimal pH levels are necessary for microbial stability and overall flavor balance.
• **Sulphates:** Contribute to the wine's aroma and taste; they enhance complexity and act as an additional preservative measure.
• **Alcohol:** Affects body, viscosity, and flavor intensity; higher alcohol can balance high acidity in robust wines but may be overpowering if in excess.

Each of these features is vital because they collectively inform a model that predicts quality by capturing taste, balance, preservation, and overall sensory attributes. For example, while fixed and volatile acidity set the baseline for taste, residual sugar and citric acid can moderate harsh flavors; density and pH add to the body and stability, and alcohol contributes to both the structural and aromatic dimensions.

Missing Data Handling & Imputation Techniques:

In the wine quality data set, missing values may arise during data collection or preprocessing. Although many versions of this popular data set are complete, handling missing data is an essential step in feature engineering. The common imputation techniques include:

1. **Mean/Median Imputation:**
   • *Advantages:* Simple to implement and preserves the dataset size.
   • *Disadvantages:* Can distort distribution properties (especially with skewed data) and reduce variability.

2. **K-Nearest Neighbors (KNN) Imputation:**
   • *Advantages:* Uses local similarities to estimate missing values, which often leads to more accurate imputations.
   • *Disadvantages:* Computationally intensive and sensitive to the choice of distance metric and K value.

3. **Regression Imputation:**
   • *Advantages:* Leverages relationships among variables to predict missing values, potentially capturing complex dependencies.
   • *Disadvantages:* Can overfit and underestimate variability, as predicted values may cluster too tightly.

Each technique carries trade-offs; the best choice depends on the extent and pattern of missingness as well as the underlying data distribution. In practice, one might evaluate the "missing completely at random" (MCAR) assumption to decide if a simple mean/median imputation is sufficient, or if more sophisticated methods like KNN or regression imputation are required.

Overall, understanding the contribution of each feature helps build more accurate predictive models for wine quality, while careful handling of missing data ensures that the integrity of the model is maintained.