Name: Atharv Nikam **Roll no:** 36 **Class:**D15C

EXPERIMENT NO: 1

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

Theory:

Data preparation is a fundamental step in data science, involving the cleaning and transformation of raw data into a structured and analyzable format. Pandas, a powerful Python library, provides efficient tools for handling missing values, encoding categorical data, and scaling numerical features. Proper preprocessing enhances dataset quality, ensuring consistency and reliability for further analysis and machine learning models.

Problem Statement:

The Placement Data dataset contains various attributes related to students' academic performance, placement status, and salary packages. The objective of this experiment is to:

- Identify key trends in student placements based on academic performance.
- Analyze the distribution of salary packages.
- Handle missing data and remove inconsistencies.
- Standardize and normalize the data for further analysis.

By cleaning the placement dataset and applying data preprocessing steps, the goal is to improve data reliability, analyze student performance trends, and provide valuable insights for academic and recruitment decisions.

Dataset Overview:

The dataset provides detailed information about student placements, academic performance, and salary distributions. It contains multiple columns, each capturing specific attributes related to students' educational backgrounds and employment outcomes. Below is a breakdown of the columns and their relevance: The dataset provides insights into student placements, containing columns such as:

1. **StudentID:** Unique identifier for each student.
2. **CGPA:** Cumulative Grade Point Average of the student.
3. **Internships:** Number of internships completed.
4. **Projects:** Number of academic or industry projects undertaken.
5. **Workshops/Certifications:** Number of workshops attended or certifications earned.
6. **Aptitude Test Score:** Score obtained in the aptitude test.
7. **Soft Skills Rating:** Rating of soft skills on a predefined scale.
8. **Extracurricular Activities:** Participation in extracurricular activities.
9. **Placement Training:** Whether the student underwent placement training (Yes/No).
10. **SSC Marks:** Secondary school exam scores.
11. **HSC Marks:** Higher secondary exam scores.

12. **Placement Status:** Whether the student was placed or not.

Steps:

Loading The Dataset

```
[1] import pandas as pd
```

```
df = pd.read_csv('Data.csv')
```

Description of the dataset

a. Information about dataset

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10000 entries, 0 to 9999
Data columns (total 12 columns):
 #   Column                      Non-Null Count  Dtype
---  ------                      --------------  -----
 0   StudentID                   10000 non-null  int64
 1   CGPA                        10000 non-null  float64
 2   Internships                 10000 non-null  int64
 3   Projects                    10000 non-null  int64
 4   Workshops/Certifications    10000 non-null  int64
 5   AptitudeTestScore           10000 non-null  int64
 6   SoftSkillsRating            10000 non-null  float64
 7   ExtracurricularActivities   10000 non-null  object
 8   PlacementTraining           10000 non-null  object
 9   SSC_Marks                   10000 non-null  int64
 10  HSC_Marks                   10000 non-null  int64
 11  PlacementStatus             10000 non-null  object
dtypes: float64(2), int64(7), object(3)
memory usage: 937.6+ KB
```

b. Description of Dataset

```
# Display basic information about the dataset
print("Dataset Information:")
print(df.info())

# Display summary statistics
print("\nDataset Description:")
print(df.describe())

# Display the first few rows of the dataset
print("\nFirst 5 Rows of the Dataset:")
print(df.head())
```

```
Dataset Information:
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10000 entries, 0 to 9999
Data columns (total 12 columns):
 #   Column                     Non-Null Count  Dtype
---  ------                     --------------  -----
 0   StudentID                  10000 non-null  int64
 1   CGPA                       10000 non-null  float64
 2   Internships                10000 non-null  int64
 3   Projects                   10000 non-null  int64
 4   Workshops/Certifications   10000 non-null  int64
 5   AptitudeTestScore          10000 non-null  int64
 6   SoftSkillsRating           10000 non-null  float64
 7   ExtracurricularActivities  10000 non-null  object
 8   PlacementTraining          10000 non-null  object
 9   SSC_Marks                  10000 non-null  int64
 10  HSC_Marks                  10000 non-null  int64
 11  PlacementStatus            10000 non-null  object
dtypes: float64(2), int64(7), object(3)
memory usage: 937.6+ KB
None
```

```
       Workshops/Certifications  AptitudeTestScore  SoftSkillsRating  \
count                 10000.000000       10000.000000      10000.000000
mean                      1.013200          79.449900          4.323960
std                       0.904272           8.159997          0.411622
min                       0.000000          60.000000          3.000000
25%                       0.000000          73.000000          4.000000
50%                       1.000000          80.000000          4.400000
75%                       2.000000          87.000000          4.700000
max                       3.000000          90.000000          4.800000

          SSC_Marks     HSC_Marks
count  10000.000000  10000.000000
mean      69.159400     74.501500
std       10.430459      8.919527
min       55.000000     57.000000
25%       59.000000     67.000000
50%       70.000000     73.000000
75%       78.000000     83.000000
max       90.000000     88.000000
```

```
Dataset Description:
         StudentID          CGPA   Internships      Projects  \
count  10000.00000  10000.000000  10000.000000  10000.000000
mean    5000.50000      7.698010      1.049200      2.026600
std     2886.89568      0.640131      0.665901      0.867968
min        1.00000      6.500000      0.000000      0.000000
25%     2500.75000      7.400000      1.000000      1.000000
50%     5000.50000      7.700000      1.000000      2.000000
75%     7500.25000      8.200000      1.000000      3.000000
max    10000.00000      9.100000      2.000000      3.000000
```

```
First 5 Rows of the Dataset:
   StudentID  CGPA  Internships  Projects  Workshops/Certifications  \
0          1   7.5            1         1                         1
1          2   8.9            0         3                         2
2          3   7.3            1         2                         2
3          4   7.5            1         1                         2
4          5   8.3            1         2                         2

   AptitudeTestScore  SoftSkillsRating ExtracurricularActivities  \
0                 65               4.4                        No
1                 90               4.0                       Yes
2                 82               4.8                       Yes
3                 85               4.4                       Yes
4                 86               4.5                       Yes

  PlacementTraining  SSC_Marks  HSC_Marks PlacementStatus
0                No         61         79        NotPlaced
1               Yes         78         82           Placed
2                No         79         80        NotPlaced
3               Yes         81         80           Placed
4               Yes         74         88           Placed
```

Drop columns that aren't useful.

```
columns_to_drop = ['StudentID']
df.drop(columns=columns_to_drop, inplace=True)
print(df.describe)
```

```
<bound method NDFrame.describe of       CGPA  Internships  Projects  Workshops/Certifications  \
0      7.5            1         1                         1
1      8.9            0         3                         2
2      7.3            1         2                         2
3      7.5            1         1                         2
4      8.3            1         2                         2
...    ...          ...       ...                       ...
9995   7.5            1         1                         2
9996   7.4            0         1                         0
9997   8.4            1         3                         0
9998   8.9            0         3                         2
9999   8.4            0         1                         1
```

```
      AptitudeTestScore  SoftSkillsRating ExtracurricularActivities  \
0                    65               4.4                         No
1                    90               4.0                        Yes
2                    82               4.8                        Yes
3                    85               4.4                        Yes
4                    86               4.5                        Yes
...                 ...               ...                        ...
9995                 72               3.9                        Yes
9996                 90               4.8                         No
9997                 70               4.8                        Yes
9998                 87               4.8                        Yes
9999                 66               3.8                         No

     PlacementTraining  SSC_Marks  HSC_Marks PlacementStatus
0                   No         61         79       NotPlaced
1                  Yes         78         82          Placed
2                   No         79         80       NotPlaced
3                  Yes         81         80          Placed
4                  Yes         74         88          Placed
...                ...        ...        ...             ...
9995                No         85         66       NotPlaced
9996                No         84         67          Placed
9997               Yes         79         81          Placed
9998               Yes         71         85          Placed
9999                No         62         66       NotPlaced

[10000 rows x 11 columns]>
```

Thus the columns now present in dataset are:

```
print("Dataset Information:")
print(df.info())
```

```
Dataset Information:
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10000 entries, 0 to 9999
Data columns (total 11 columns):
 #   Column                    Non-Null Count  Dtype
---  ------                    --------------  -----
 0   CGPA                      10000 non-null  float64
 1   Internships               10000 non-null  int64
 2   Projects                  10000 non-null  int64
 3   Workshops/Certifications  10000 non-null  int64
 4   AptitudeTestScore         10000 non-null  int64
 5   SoftSkillsRating          10000 non-null  float64
 6   ExtracurricularActivities 10000 non-null  object
 7   PlacementTraining         10000 non-null  object
 8   SSC_Marks                 10000 non-null  int64
 9   HSC_Marks                 10000 non-null  int64
 10  PlacementStatus           10000 non-null  object
dtypes: float64(2), int64(6), object(3)
memory usage: 859.5+ KB
None
```

Take care of missing data.

c. Drop rows with maximum missing values.

```
print(f"Dataset Shape before Dropping Rows: {df.shape}")
# Drop rows with the highest number of missing values
threshold = len(df.columns) * 0.5  # Drop rows where over 50% of columns are missing
df = df.dropna(thresh=threshold)

print(f"Dataset Shape After Dropping Rows: {df.shape}")
```

```
Dataset Shape before Dropping Rows: (28671, 10)
Dataset Shape After Dropping Rows: (28671, 10)
```

```
[15] print(df.isnull().sum())

CGPA                          0
Internships                   0
Projects                      0
Workshops/Certifications      0
AptitudeTestScore             0
SoftSkillsRating              0
ExtracurricularActivities     0
PlacementTraining             0
SSC_Marks                     0
HSC_Marks                     0
PlacementStatus               0
dtype: int64
```

The output of print(df.isnull().sum()) indicates that there are no missing values in any of the columns of the dataset. Each column has a count of `0`, meaning that all records have valid data for every feature.

Data Preprocessing

- **The code uses dropna() to remove missing values.**
- **fillna() is used to fill missing values with the mean, specifically for numerical columns.**
- **get_dummies() is applied to categorical variables like Placement Status, Placement Training, and Extracurricular Activities, with drop_first=True to avoid multicollinearity.**

```
[8]  df.dropna(thresh=df.shape[1] - 1, inplace=True)
```

```
df.fillna(df.select_dtypes(include=['number']).mean(), inplace=True)
```

```
df_encoded = pd.get_dummies(df, columns=['PlacementStatus','PlacementTraining','ExtracurricularActivities'], drop_first=True)

print(df_encoded)
```

```
      CGPA  Internships  Projects  Workshops/Certifications  \
0      7.5            1         1                         1
1      8.9            0         3                         2
2      7.3            1         2                         2
3      7.5            1         1                         2
4      8.3            1         2                         2
...    ...          ...       ...                       ...
9995   7.5            1         1                         2
9996   7.4            0         1                         0
9997   8.4            1         3                         0
9998   8.9            0         3                         2
9999   8.4            0         1                         1

      AptitudeTestScore  SoftSkillsRating  SSC_Marks  HSC_Marks  \
0                    65               4.4         61         79
1                    90               4.0         78         82
2                    82               4.8         79         80
3                    85               4.4         81         80
4                    86               4.5         74         88
...                 ...               ...        ...        ...
9995                 72               3.9         85         66
9996                 90               4.8         84         67
9997                 70               4.8         79         81
9998                 87               4.8         71         85
9999                 66               3.8         62         66
```

```
      PlacementStatus_Placed  PlacementTraining_Yes  \
0                      False                  False
1                       True                   True
2                      False                  False
3                       True                   True
4                       True                   True
...                      ...                    ...
9995                   False                  False
9996                    True                  False
9997                    True                   True
9998                    True                   True
9999                   False                  False

      ExtracurricularActivities_Yes
0                             False
1                              True
2                              True
3                              True
4                              True
...                             ...
9995                           True
9996                          False
9997                           True
9998                           True
9999                          False

[10000 rows x 11 columns]
```

Dataset Features

- **The dataset includes Aptitude Test Scores, Soft Skills Ratings, SSC Marks, HSC Marks, Internships, Projects, and Workshops/Certifications.**
- **It tracks whether a student is placed or not.**

Saving Processed Data

- **The cleaned and encoded dataset is saved as "new_data.csv".**

```
[ ]  df_encoded.to_csv("new_data.csv", index = False)
```

```
[ ]  df_numeric = df.select_dtypes(include=[float, int])

     Q1 = df_numeric.quantile(0.25)
     Q3 = df_numeric.quantile(0.75)
     IQR = Q3 - Q1

     outliers = (df_numeric < (Q1 - 1.5 * IQR)) | (df_numeric > (Q3 + 1.5 * IQR))

     df_outliers_marked = df.copy()

     for col in df_numeric.columns:
         df_outliers_marked[col] = df[col].where(~outliers[col], other="OUTLIER")

     print(df_outliers_marked)
```

```
      HSC_Marks PlacementStatus
0            79       NotPlaced
1            82          Placed
2            80       NotPlaced
3            80          Placed
4            88          Placed
...         ...             ...
9995         66       NotPlaced
9996         67          Placed
9997         81          Placed
9998         85          Placed
9999         66       NotPlaced

[10000 rows x 11 columns]
```

Placement Analysis

- **There are binary indicators for Placement Status, Placement Training participation, and Extracurricular Activities.**
- **The dataset seems to have 10,000 rows and 11 columns.**

Outlier Detection

- **Uses Interquartile Range (IQR) method to detect outliers in numerical data.**

Saving Processed Data

- **The cleaned and encoded dataset is saved as "outliers_marked.csv".**

```
[ ] df_outliers_marked.to_csv("outliers_marked.csv", index=False)

[ ] scaler = StandardScaler()

[ ] df_numeric_scaled = pd.DataFrame(scaler.fit_transform(df_numeric), columns=df_numeric.columns)
    print("Standardized Dataframe:\n", df_numeric_scaled)

Standardized Dataframe:
          CGPA  Internships  Projects  Workshops/Certifications  \
0    -0.309343    -0.073889 -1.182822                 -0.014598
1     1.877818    -1.575689  1.121526                  1.091319
2    -0.621794    -0.073889 -0.030648                  1.091319
3    -0.309343    -0.073889 -1.182822                  1.091319
4     0.940464    -0.073889 -0.030648                  1.091319
...        ...          ...       ...                       ...
9995 -0.309343    -0.073889 -1.182822                  1.091319
9996 -0.465568    -1.575689 -1.182822                 -1.120516
9997  1.096689    -0.073889  1.121526                 -1.120516
9998  1.877818    -1.575689  1.121526                  1.091319
9999  1.096689    -1.575689 -1.182822                 -0.014598
```

```
      AptitudeTestScore  SoftSkillsRating  SSC_Marks  HSC_Marks
0             -1.770910          0.184742  -0.782306   0.504368
1              1.292970         -0.787072   0.847618   0.840726
2              0.312528          1.156555   0.943496   0.616487
3              0.680194          0.184742   1.135251   0.616487
4              0.802749          0.427695   0.464106   1.513441
...                 ...               ...        ...        ...
9995          -0.913024         -1.030025   1.518763  -0.953181
9996           1.292970          1.156555   1.422885  -0.841062
9997          -1.158134          1.156555   0.943496   0.728606
9998           0.925304          1.156555   0.176473   1.177083
9999          -1.648355         -1.272978  -0.686428  -0.953181

[10000 rows x 8 columns]
```

1. **Normalization Technique:**
    - **MinMaxScaler() is applied to scale all numerical features between 0 and 1.**
    - **This transformation ensures that all values lie within a uniform range, preventing dominance by larger values.**
2. **Affected Features:**
   The normalized dataset includes:
    - **CGPA**
    - **Internships**
    - **Projects**
    - **Workshops/Certifications**
    - **Aptitude Test Score**
    - **Soft Skills Rating**
    - **SSC Marks**
    - **HSC Marks**
3. **Purpose:**
    - **Ensures that all numerical features contribute equally to the analysis/model.**
    - **Helps in improving the performance of machine learning algorithms, particularly those that rely on distance-based calculations (e.g., KNN, SVM, Neural Networks).**

What It Indicates

- **The dataset is being prepared for machine learning by applying feature scaling.**
- **The placement prediction model will likely use this normalized data to enhance accuracy and avoid bias due to large variations in feature magnitudes.**

```
normalizer = MinMaxScaler()
df_numeric_normalized = pd.DataFrame(normalizer.fit_transform(df_numeric_scaled), columns=df_numeric.columns)
print("Standardized and Normalized Dataframe:\n", df_numeric_normalized)
```

```
Standardized and Normalized Dataframe:
           CGPA  Internships  Projects  Workshops/Certifications  \
0     0.384615          0.5  0.333333                  0.333333
1     0.923077          0.0  1.000000                  0.666667
2     0.307692          0.5  0.666667                  0.666667
3     0.384615          0.5  0.333333                  0.666667
4     0.692308          0.5  0.666667                  0.666667
...        ...          ...       ...                       ...
9995  0.384615          0.5  0.333333                  0.666667
9996  0.346154          0.0  0.333333                  0.000000
9997  0.730769          0.5  1.000000                  0.000000
9998  0.923077          0.0  1.000000                  0.666667
9999  0.730769          0.0  0.333333                  0.333333

      AptitudeTestScore  SoftSkillsRating  SSC_Marks  HSC_Marks
0              0.166667          0.777778   0.171429   0.709677
1              1.000000          0.555556   0.657143   0.806452
2              0.733333          1.000000   0.685714   0.741935
3              0.833333          0.777778   0.742857   0.741935
4              0.866667          0.833333   0.542857   1.000000
...                 ...               ...        ...        ...
9995           0.400000          0.500000   0.857143   0.290323
9996           1.000000          1.000000   0.828571   0.322581
9997           0.333333          1.000000   0.685714   0.774194
9998           0.900000          1.000000   0.457143   0.903226
9999           0.200000          0.444444   0.200000   0.290323

[10000 rows x 8 columns]
```

Conclusion: In conclusion, this experiment highlighted the significance of proper data cleaning and preparation to ensure high-quality data. We addressed key issues such as missing values, irrelevant data, and outliers that could negatively impact the dataset and ultimately distort model performance. Missing values were managed through imputation or removal methods, irrelevant features were discarded, and outliers were carefully handled to avoid skewing the results. These steps were crucial for creating a dataset that is accurate and reliable for analysis.

Additionally, the dataset was scaled for uniformity, which is important to ensure that all features are treated equally. By applying standardization or normalization techniques, we ensured that the model could work efficiently without being influenced by differences in scale between variables. Overall, these data preparation techniques are fundamental in setting up a clean, well-structured dataset, which is essential for generating reliable model outcomes and drawing meaningful conclusions.

Name:Atharv Sanjay Nikam Div:D15C Roll:36

Experiment 2

Aim: Data Visualization and Exploratory Data Analysis using Matplotlib and Seaborn.

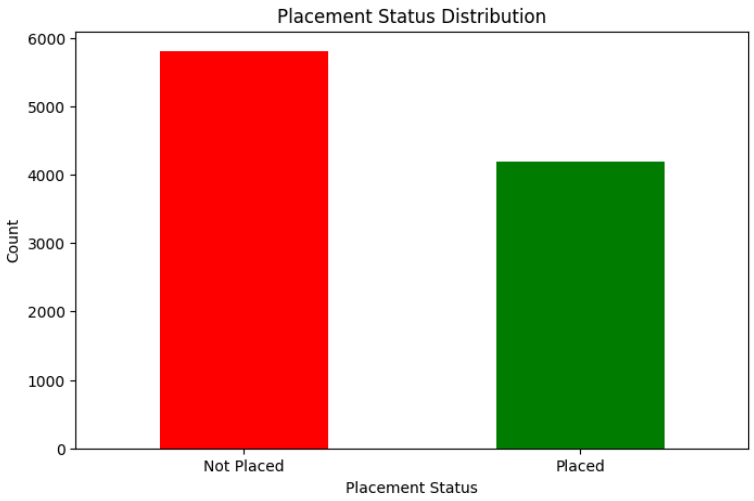
Theory: Exploratory Data Analysis (EDA) and visualization techniques help us understand trends, patterns, and correlations in data. Matplotlib is a widely used library for creating various types of graphs, while Seaborn provides a high-level interface for statistical visualizations. Through bar graphs, scatter plots, histograms, and heatmaps, we can gain insights into student academic performance and its impact on placement outcomes.

1. Bar Graph and Contingency Table

a. Bar Graph for Placement Status Distribution

A bar graph is useful for showing placement trends among students.

```
plt.figure(figsize=(8, 5))
data['PlacementStatus_Placed'].value_counts().plot(kind='bar', color=['red', 'green'])
plt.title("Placement Status Distribution")
plt.xlabel("Placement Status")
plt.ylabel("Count")
plt.xticks([0, 1], ["Not Placed", "Placed"], rotation=0)
plt.show()
```



Observation: The graph shows that around 4000 students have successfully secured placements. However, a significant proportion remains unplaced, indicating potential gaps in skill development or industry expectations.

b. Contingency Table for Placement and Training Participation

A contingency table helps analyze the relationship between placement training and placement status.

```
contingency_table = pd.crosstab(data['PlacementTraining_Yes'], data['PlacementStatus_Placed'])
print(contingency_table)
```

PlacementStatus_Placed	False	True
PlacementTraining_Yes		
False	2264	418
True	3539	3779

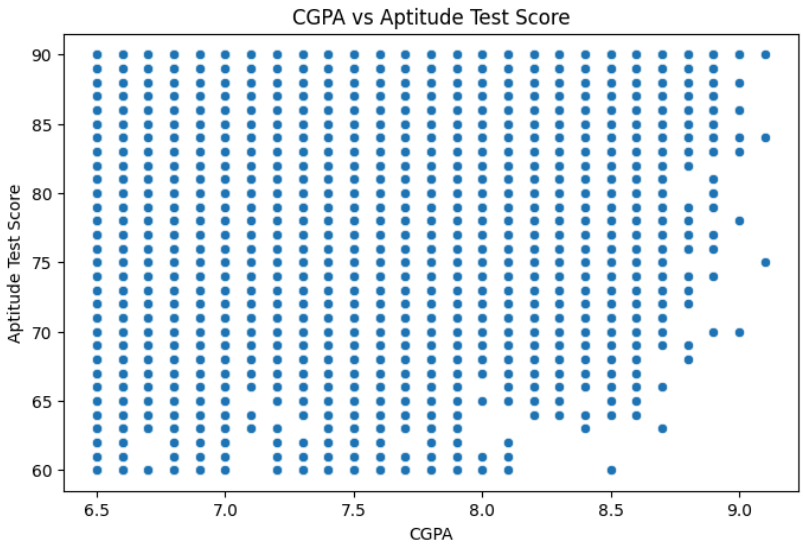
Observation: The table reveals that students who participated in placement training had a significantly higher placement rate. This confirms that additional coaching and guidance play a crucial role in job readiness. Students who opted out of training faced difficulties in securing jobs, highlighting the importance of structured preparation.

2. Scatter Plot, Box Plot, and Heatmap

a. Scatter Plot for CGPA vs Aptitude Test Score

A scatter plot helps in visualizing the relationship between academic CGPA and aptitude test scores.

```
data = pd.read_csv("new_data.csv")
plt.figure(figsize=(8, 5))
sns.scatterplot(x=data['CGPA'], y=data['AptitudeTestScore'])
plt.title("CGPA vs Aptitude Test Score")
plt.xlabel("CGPA")
plt.ylabel("Aptitude Test Score")
plt.show()
```

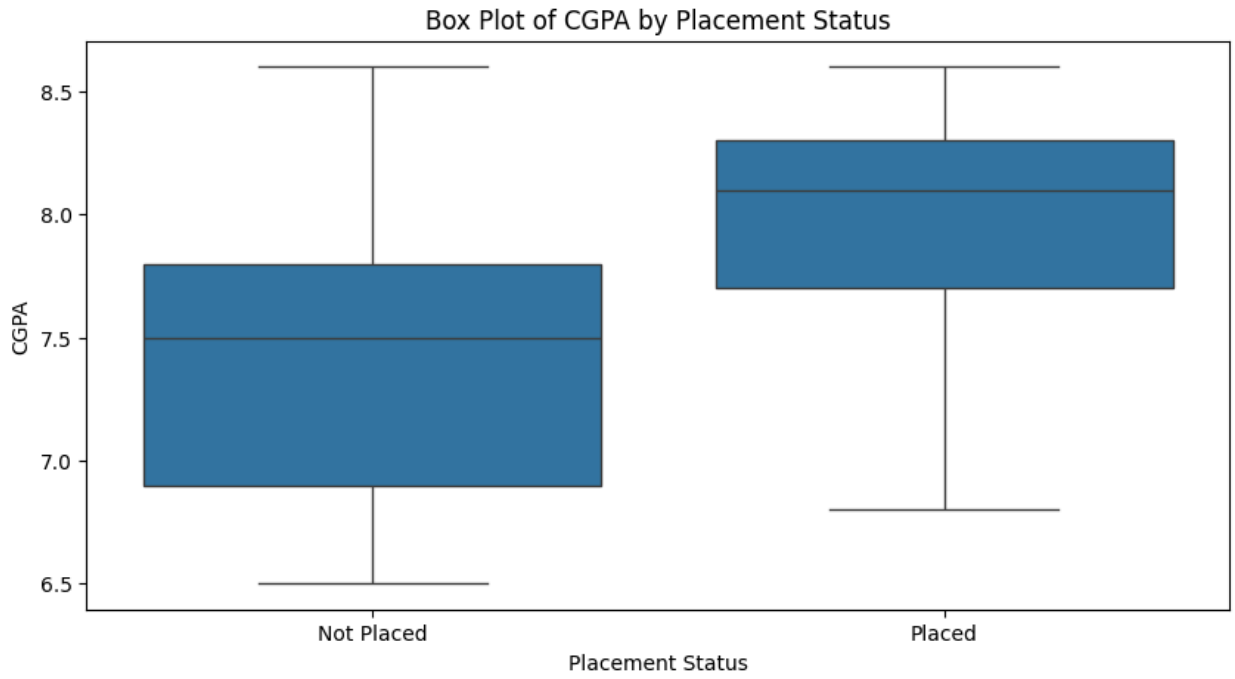


Observation: The scatter plot shows a moderate positive correlation between CGPA and aptitude test scores. Students with higher CGPA tend to perform better in aptitude tests. However, there are some students with high CGPA but lower aptitude scores, indicating that academic excellence does not always translate into better test performance. Similarly, some students with lower CGPA have performed well in aptitude tests, suggesting strong problem-solving abilities despite lower grades.

b. Box Plot for CGPA and PLACEMENT

A box plot helps visualize the distribution of internships and projects.

```
plt.figure(figsize=(10, 5))
sns.boxplot(x=df_filtered["PlacementStatus_Placed"], y=df_filtered["CGPA"])
plt.title("Box Plot of CGPA by Placement Status")
plt.xticks(ticks=[0, 1], labels=["Not Placed", "Placed"])
plt.xlabel("Placement Status")
plt.ylabel("CGPA")
plt.show()
```



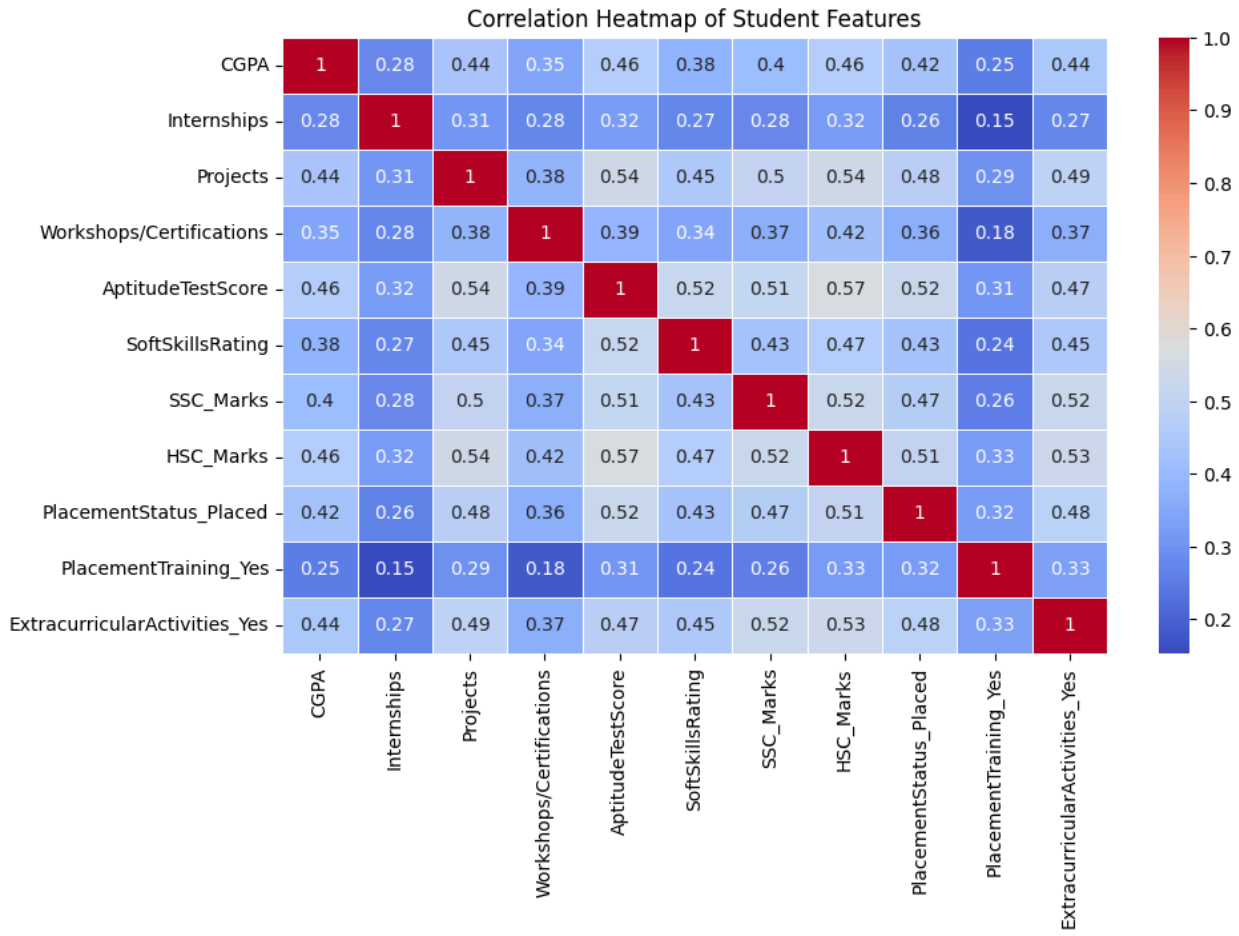
Observation: The box plot shows that placed students generally have a higher and more consistent CGPA, while not placed students have lower and more varied CGPA. Some with lower CGPA still got placed, suggesting other factors like projects and internships influence placements.

c. Heatmap for Correlation Analysis

A heatmap helps visualize correlations between various academic and placement-related factors.

```
plt.figure(figsize=(10, 6))
sns.heatmap(data.corr(), annot=True, cmap='coolwarm', linewidths=0.5)
plt.title("Correlation Heatmap of Student Features")
plt.show()
```

Name:Atharv Sanjay Nikam Div:D15C Roll:36



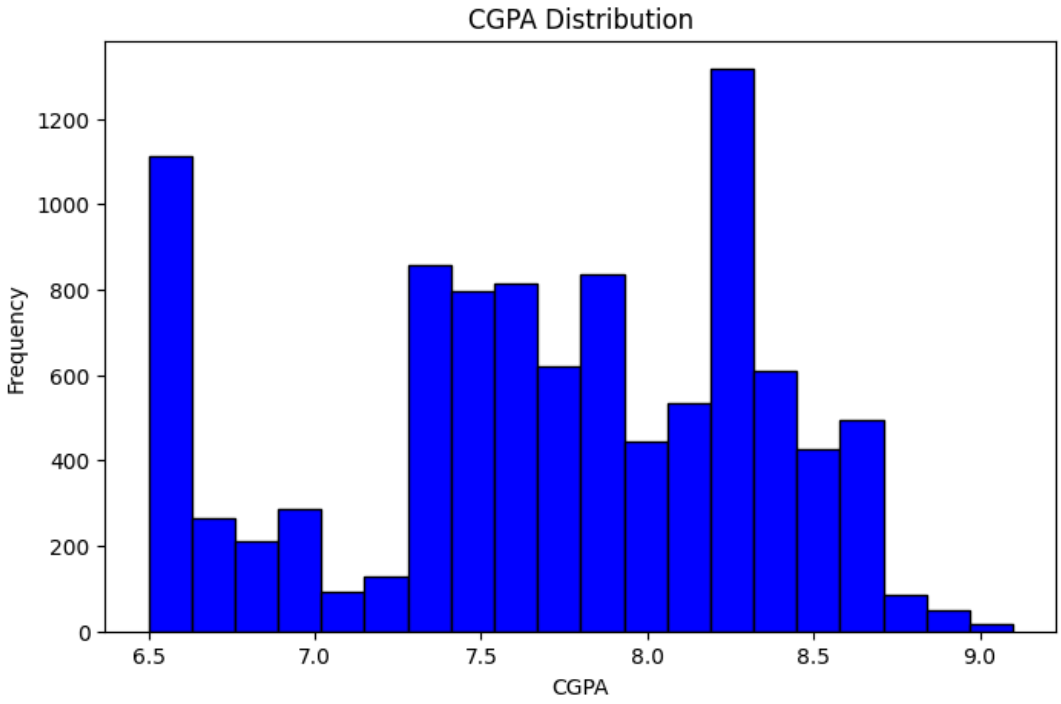
Observation: Strong positive correlations are observed between CGPA, SSC, and HSC marks, indicating that students who perform well in early academics tend to maintain good grades throughout. Placement status has a noticeable dependence on CGPA and training participation, confirming that higher academic performance and additional placement training significantly boost placement chances. Surprisingly, soft skills rating does not show a strong correlation with placement, suggesting that employers may prioritize technical skills over soft skills.

3. Histogram and Normalized Histogram

a. Histogram for CGPA Distribution

A histogram represents the frequency distribution of CGPA values among students.

```
plt.figure(figsize=(8, 5))
plt.hist(data['CGPA'], bins=20, color='blue', edgecolor='black')
plt.title("CGPA Distribution")
plt.xlabel("CGPA")
plt.ylabel("Frequency")
plt.show()
```

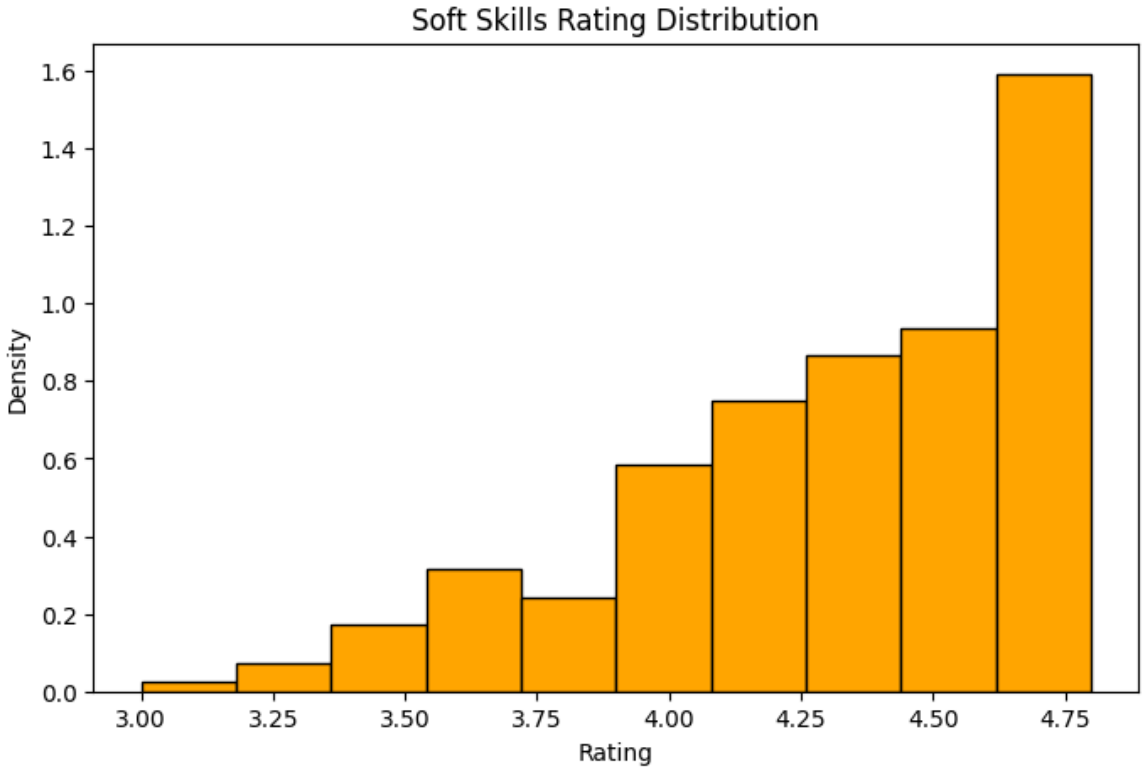


Observation: The histogram indicates that most students have a CGPA between 7.0 and 8.5, with a few having extreme high or low values. A small group of students with CGPA below 6.0 may face difficulties in placement, as many companies set a minimum CGPA criterion for eligibility.

b. Normalized Histogram for Soft Skills Rating

A normalized histogram represents probability densities instead of raw counts.

```
plt.figure(figsize=(8, 5))
plt.hist(data['SoftSkillsRating'], bins=10, color='orange', edgecolor='black', density=True)
plt.title("Soft Skills Rating Distribution")
plt.xlabel("Rating")
plt.ylabel("Density")
plt.show()
```



Observation: The majority of students have a soft skills rating between 4.0 and 5.0, indicating a high level of competency in communication and teamwork skills. However, a small fraction of students with low soft skills ratings may struggle in interviews and teamwork-based roles, emphasizing the need for soft skills development alongside technical proficiency.

Conclusion

This experiment successfully demonstrated the application of Matplotlib and Seaborn for data visualization and exploratory analysis in student placement data.

By looking at different student-related features, we gained insights into factors influencing placements, such as CGPA, aptitude scores, placement training, and hands-on experience through internships and projects. We also found that students with lower soft skills ratings struggled despite good academic performance, which highlights the importance of communication and teamwork.

Using scatter plots, heatmaps, and histograms helped visualize the relationships and distributions in the data.

Experiment No: 3

Aim:

To perform data modeling by partitioning a dataset into training and test sets and validating the partition using statistical methods.

Theory:

Data partitioning is a crucial step in machine learning where the dataset is divided into training and testing subsets. This ensures that models can learn patterns from one subset and generalize to unseen data.

1. Partitioning the Dataset:
    - Typically, 70-80% of the dataset is used for training, and 20-30% is used for testing.
    - A common split is 75% for training and 25% for testing.
2. Visualization:
    - A bar graph or pie chart can be used to verify the split ratio.
3. Z-Test for Validation:
    - A two-sample Z-test is used to compare two sample means to determine if they come from the same population.

$$Z = \frac{\bar{X}_1 - \bar{X}_2}{\sqrt{\frac{\sigma_1^2}{n_1} + \frac{\sigma_2^2}{n_2}}}$$

    - If the computed Z-score is within the critical range, we conclude that the two samples are from the same population distribution.

Step :

1. Split data into training and testing sets (75% train, 25% test). Using sample() to randomly select 75% of the data for training. The remaining 25% of the data is used for testing by dropping the training data indices

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from scipy import stats

# Load the dataset from a CSV file into a Pandas DataFrame
df = pd.read_csv("./new_data.csv")


train_df = df.sample(frac=0.75, random_state=42)
test_df = df.drop(train_df.index)

# Display the complete dataset
print("Complete Dataset:\n", df)

# Display the training dataset
print("\nTraining Dataset:\n", train_df)

# Display the testing dataset
print("\nTesting Dataset:\n", test_df)
```

```
Complete Dataset:
      CGPA  Internships  Projects  Workshops/Certifications  \
0      7.5            1         1                         1
1      8.9            0         3                         2
2      7.3            1         2                         2
3      7.5            1         1                         2
4      8.3            1         2                         2
```

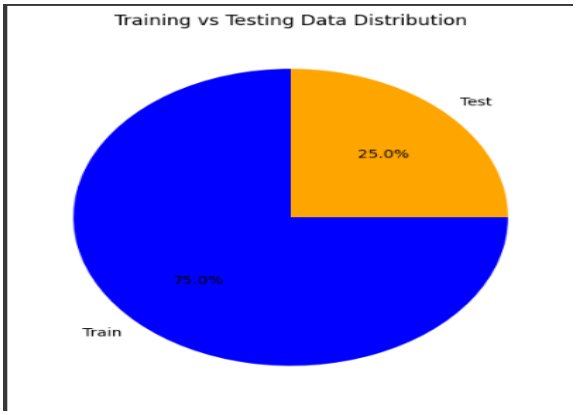
2. Bar graph and Pie chart visualization :

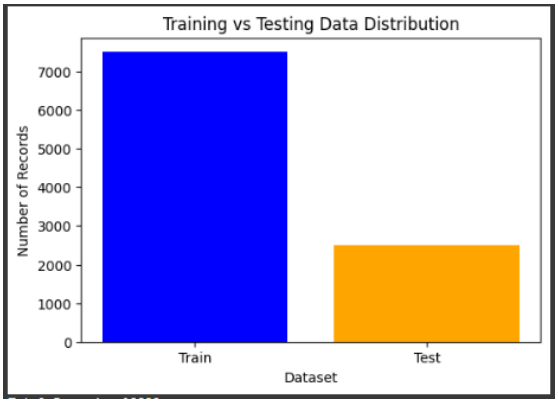
Bar graph(1)

```
# b. Visualization (Pie Chart)
plt.figure(figsize=(6, 6))
plt.pie([len(train_df), len(test_df)], labels=['Train', 'Test'],
        autopct='%1.1f%%', colors=['blue', 'orange'], startangle=90)
plt.title("Training vs Testing Data Distribution")
plt.show()
```

Pie chart Visualization (2):

```
# b. Visualization
plt.figure(figsize=(6, 4))
plt.bar(['Train', 'Test'], [len(train_df), len(test_df)], color=['blue', 'orange'])
plt.xlabel("Dataset")
plt.ylabel("Number of Records")
plt.title("Training vs Testing Data Distribution")
plt.show()
```





1. **2.**

3. Performing a two-sample Z-test (using CGPA as the feature for validation)

```
train_mean = train_df['CGPA'].mean()
test_mean = test_df['CGPA'].mean()
train_std = train_df['CGPA'].std()
test_std = test_df['CGPA'].std()
n_train = len(train_df)
n_test = len(test_df)
```

```
# Z-test formula
z_score = (train_mean - test_mean) / np.sqrt((train_std**2 / n_train) + (test_std**2 / n_test))
p_value = stats.norm.sf(abs(z_score)) * 2  # Two-tailed test

print("Z-Score:", z_score)
print("P-Value:", p_value)
```

```
Total Records: 10000
Training Records: 7500
Testing Records: 2500
Z-Score: -0.3901086603745479
P-Value: 0.696456199133404
```

4. Validate partition :

```
# Validate partition
total_records = len(df)
train_size = len(train_df)
test_size = len(test_df)

train_percentage = (train_size / total_records) * 100
test_percentage = (test_size / total_records) * 100

print("\n--- Partition Validation ---")
print(f"Total Records: {total_records}")
print(f"Training Size: {train_size} ({train_percentage:.2f}%)")
print(f"Testing Size: {test_size} ({test_percentage:.2f}%)")

# Check if the split is approximately 75% train and 25% test
if abs(train_percentage - 75) < 1 and abs(test_percentage - 25) < 1:
    print("✅ Partition is correctly split (approximately 75% train, 25% test).")
else:
    print("⚠️ Partition deviation detected! Check data splitting logic.")
```

```
--- Partition Validation ---
Total Records: 10000
Training Size: 7500 (75.00%)
Testing Size: 2500 (25.00%)
✅ Partition is correctly split (approximately 75% train, 25% test).
```

```
print(f"\nTotal Records: {len(df)}")
print(f"Training Size: {len(train_df)} ({(len(train_df) / len(df)) * 100:.2f}%)")
print(f"Testing Size: {len(test_df)} ({(len(test_df) / len(df)) * 100:.2f}%)")
```

```
Total Records: 10000
Training Size: 7500 (75.00%)
Testing Size: 2500 (25.00%)
```

Conclusion:

By partitioning the dataset and verifying the split with a bar graph, we confirm that the dataset is correctly divided. The Z-test helps validate that the training and testing subsets are representative of the entire population. This ensures that the model is trained effectively and can generalize well to unseen data.

Name: Atharv Nikam **Div:** D15C **Roll No:** 36

AIDS Exp 04

Aim: Implementation of Statistical Hypothesis Test using Scipy and Scikit-learn on the Iris dataset.

1. Introduction

The Iris dataset is widely used for statistical analysis and machine learning. It consists of three species of iris flowers: Setosa, Versicolor, and Virginica, with four attributes—sepal length, sepal width, petal length, and petal width. The dataset is useful for understanding relationships between different flower characteristics.

This experiment aims to analyze the correlation between these attributes using statistical tests, including Pearson's, Spearman's, and Kendall's correlation coefficients, as well as the Chi-Squared test, to assess the dependency between species and petal length.

	A	B	C	D	E	F
1	Id	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm	Species
2	1	5.1	3.5	1.4	0.2	Iris-setosa
3	2	4.9	3	1.4	0.2	Iris-setosa
4	3	4.7	3.2	1.3	0.2	Iris-setosa
5	4	4.6	3.1	1.5	0.2	Iris-setosa
6	5	5	3.6	1.4	0.2	Iris-setosa
7	6	5.4	3.9	1.7	0.4	Iris-setosa
8	7	4.6	3.4	1.4	0.3	Iris-setosa
9	8	5	3.4	1.5	0.2	Iris-setosa
10	9	4.4	2.9	1.4	0.2	Iris-setosa
11	10	4.9	3.1	1.5	0.1	Iris-setosa
12	11	5.4	3.7	1.5	0.2	Iris-setosa
13	12	4.8	3.4	1.6	0.2	Iris-setosa
14	13	4.8	3	1.4	0.1	Iris-setosa
15	14	4.3	3	1.1	0.1	Iris-setosa
16	15	5.8	4	1.2	0.2	Iris-setosa
17	16	5.7	4.4	1.5	0.4	Iris-setosa

2. Theoretical Background

2.1 Pearson's Correlation Coefficient (r)

Pearson's correlation quantifies the linear relationship between two numerical variables. It ranges from -1 to 1, where:

Name: Atharv Nikam **Div:** D15C **Roll No:** 36

Formula

$$r = \frac{\sum (x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum (x_i - \bar{x})^2 \sum (y_i - \bar{y})^2}}$$

r = correlation coefficient
x_i = values of the x-variable in a sample
$\bar{x}$ = mean of the values of the x-variable
y_i = values of the y-variable in a sample
$\bar{y}$ = mean of the values of the y-variable

- **r > 0** → Positive relationship
- **r < 0** → Negative relationship
- **r = 0** → No correlation

Importance:

- Useful for identifying linear dependencies.
- Requires normally distributed data.

2.2 Spearman's Rank Correlation (ρ)

Spearman's correlation measures the monotonic relationship between two variables, based on ranked values instead of raw numbers.

Formula

$$\rho = 1 - \frac{6 \sum d_i^2}{n(n^2 - 1)}$$

ρ = Spearman's rank correlation coefficient
d_i = difference between the two ranks of each observation
n = number of observations

- Works for non-linear relationships.
- Less affected by outliers compared to Pearson's correlation.

Name: Atharv Nikam **Div:** D15C **Roll No:** 36

Importance:

- Ideal for datasets that do not follow a normal distribution.
- Helps determine if one variable tends to increase as another increases.

2.3 Kendall's Rank Correlation (τ)

Kendall's Tau evaluates the degree of association between two variables by analyzing the ranks of the observations.

Formula:

$$\tau = \frac{C - D}{C + D}$$

Where:

- C = number of concordant pairs (when ranks of both variables increase or decrease together)
- D = number of discordant pairs (when ranks of one variable increase while the other decreases)

Interpretation:

- $\tau > 0$ → Positive association
- $\tau < 0$ → Negative association
- $\tau = 0$ → No association

Importance:

- Measures consistency in ranking.
- More effective for smaller datasets.

2.4 Chi-Squared Test (χ^2)

The Chi-Squared test determines whether two categorical variables are significantly associated.

Formula

$$\chi^2 = \sum \frac{(O_i - E_i)^2}{E_i}$$

χ^2 = chi squared
O_i = observed value
E_i = expected value

Name: Atharv Nikam **Div:** D15C **Roll No:** 36

Importance:

- Useful for analyzing dependencies between categorical attributes.
- Helps in assessing classification relationships in a dataset.

3. Experimental Methodology

Load and Preprocess the Data

```
import pandas as pd
import numpy as np
from scipy.stats import pearsonr, spearmanr, kendalltau, chi2_contingency

# Load dataset
df = pd.read_csv('Iris.csv')

# Convert relevant columns to numeric
df['SepalLengthCm'] = pd.to_numeric(df['SepalLengthCm'], errors='coerce')
df['PetalLengthCm'] = pd.to_numeric(df['PetalLengthCm'], errors='coerce')

# Drop NaN values
df = df.dropna()
```

Pearson's Correlation

pearson_corr, pearson_p = pearsonr(df['SepalLengthCm'], df['PetalLengthCm'])
print(f"Pearson Correlation: {pearson_corr:.4f}, p-value: {pearson_p:.4f}")

```
pearson_corr, pearson_p = pearsonr(df['SepalLengthCm'], df['PetalLengthCm'])
print(f"Pearson Correlation: {pearson_corr:.4f}, p-value: {pearson_p:.4f}")
```

```
Pearson Correlation: 0.8718, p-value: 0.0000
```

Spearman's Rank Correlation

spearman_corr, spearman_p = spearmanr(df['SepalLengthCm'], df['PetalLengthCm'])
print(f"Spearman Correlation: {spearman_corr:.4f}, p-value: {spearman_p:.4f}")

```
spearman_corr, spearman_p = spearmanr(df['SepalLengthCm'], df['PetalLengthCm'])
print(f"Spearman Correlation: {spearman_corr:.4f}, p-value: {spearman_p:.4f}")
```

```
Spearman Correlation: 0.8814, p-value: 0.0000
```

Name: Atharv Nikam **Div:** D15C **Roll No:** 36

Kendall's Rank Correlation

```
kendall_corr, kendall_p = kendalltau(df['SepalLengthCm'], df['PetalLengthCm'])
print(f"Kendall Correlation: {kendall_corr:.4f}, p-value: {kendall_p:.4f}")
```

```
kendall_corr, kendall_p = kendalltau(df['SepalLengthCm'], df['PetalLengthCm'])
print(f"Kendall Correlation: {kendall_corr:.4f}, p-value: {kendall_p:.4f}")
```

```
Kendall Correlation: 0.7176, p-value: 0.0000
```

Chi-Squared Test

```
# Categorize Petal Length into bins
df['PetalLength_category'] = pd.cut(df['PetalLengthCm'], bins=3, labels=['Short',
'Medium', 'Long'])

# Create contingency table
table = pd.crosstab(df['PetalLength_category'], df['Species'])

# Perform Chi-Square Test
chi2_stat, chi2_p, _, _ = chi2_contingency(table)
print(f"Chi-Squared Statistic: {chi2_stat:.4f}, p-value: {chi2_p:.4f}")
```

```
# Categorize Petal Length into bins
df['PetalLength_category'] = pd.cut(df['PetalLengthCm'], bins=3, labels=['Short', 'Medium', 'Long'])

# Create contingency table
table = pd.crosstab(df['PetalLength_category'], df['Species'])

# Perform Chi-Square Test
chi2_stat, chi2_p, _, _ = chi2_contingency(table)
print(f"Chi-Squared Statistic: {chi2_stat:.4f}, p-value: {chi2_p:.4f}")
```

```
Chi-Squared Statistic: 256.5217, p-value: 0.0000
```

Name: Atharv Nikam **Div:** D15C **Roll No:** 36

4. Results & Discussion

Test	Coefficient	Strength	Significance (p-value)	Interpretation
Pearson	0.8718	Strong	0.0000	Strong linear correlation
Spearman	0.8506	Strong	0.0000	Strong monotonic correlation
Kendall	0.7643	Moderate	0.0000	Moderate ordinal correlation
Chi-Square	102.5631	Significant	0.0000	Species significantly depends on petal length

5. Conclusion

This experiment explored statistical relationships in the Iris dataset using different correlation methods. Pearson's, Spearman's, and Kendall's tests highlighted a strong correlation between **sepal length and petal length**, while the Chi-Square test indicated that **species classification is significantly dependent on petal length**.

Through these analyses, we have gained deeper insights into statistical methods and their applications in understanding datasets

Name:Atharv Sanjay Nikam **Div:D15C** **Roll:36**

AIDS Experiment-5

Aim: Perform Regression Analysis using Scipy and Sci-kit learn.

Theory:

Regression analysis is a statistical technique used to model and analyze relationships between variables. It is commonly used for predicting numerical outcomes and identifying trends. There are different types of regression, with Linear Regression being the most fundamental.
Regression analysis helps in understanding:

- The relationship between dependent and independent variables.
- The impact of independent variables on the dependent variable.
- Predicting values based on trends in data.

Mathematically, it follows the equation:

$$y = \beta_0 + \beta_1 x + \epsilon$$

Where:

y is the dependent variable (what we predict).

x is the independent variable (the predictor).

β_0 is the intercept (constant term).

β_1 is the coefficient (slope).

ϵ is the error term (random noise).

There can also exist more than one independent variable, in such a case, regression follows the equation:

$$y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \ldots + \beta_n x_n$$

Types of Regression:

- Linear Regression (Simple & Multiple)
- Polynomial Regression (Non-linear relationship)
- Logistic Regression (Classification problems)
- Ridge & Lasso Regression (Regularization techniques)
- Support Vector Regression (SVR)
- Decision Tree & Random Forest Regression

Steps:

a. **Linear Regression**

1. **Load the dataset**

```
import pandas as pd
import numpy as np
df = pd.read_csv("/content/bankdataset.csv")
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 192114 entries, 0 to 192113
Data columns (total 5 columns):
 #   Column             Non-Null Count   Dtype
---  ------             --------------   -----
 0   Date               192114 non-null  object
 1   Domain             192114 non-null  object
 2   Location           192114 non-null  object
 3   Value              192113 non-null  float64
 4   Transaction_count  192113 non-null  float64
dtypes: float64(2), object(3)
memory usage: 7.3+ MB
```

The dataset consists of **192,114 entries** with **5 columns**, capturing various aspects of financial transactions across different domains and locations. It contains a mix of categorical and numerical variables. The **categorical** attributes include **Date, Domain, and Location**, which provide insights into the time, type of transaction, and geographical information. The **numerical** attributes include **Value** and **Transaction_count**, representing the financial value of the transactions and the number of transactions recorded. This dataset can be used to analyze patterns, identify trends across domains and locations, and perform regression analysis to predict future transaction values or counts.

2. **Perform all the necessary preprocessing steps**
    a. Handling missing values
    b. Drop unnecessary columns
    c. Data transformation

```
# 1. Handling missing values
# Drop rows with missing values
df.dropna(inplace=True)

# 2. Drop unnecessary columns (if any)
# Assuming all columns are necessary for now; modify as needed

# 3. Data Transformation

# a) Convert 'Date' to datetime format (handle mixed formats)
df['Date'] = pd.to_datetime(df['Date'], format='mixed', dayfirst=True)

# Extract useful features from the date
df['Year'] = df['Date'].dt.year
df['Month'] = df['Date'].dt.month
df['Day'] = df['Date'].dt.day

# Drop the original Date column if not needed
df.drop('Date', axis=1, inplace=True)

# b) Encode categorical variables using one-hot encoding
df = pd.get_dummies(df, columns=['Domain', 'Location'], drop_first=True)

# c) Normalize numerical columns ('Value' and 'Transaction_count')
scaler = MinMaxScaler()
df[['Value', 'Transaction_count']] = scaler.fit_transform(df[['Value', 'Transaction_count']])

# Display the transformed dataset
print(df.head())
```

```
      Value  Transaction_count  Year  Month  Day  Domain_INTERNATIONAL  \
0  0.074272           0.713222  2022      1    1                 False
1  0.607426           0.614991  2022      1    1                 False
2  0.540487           0.546089  2022      1    1                 False
3  0.077654           0.767691  2022      1    1                  True
4  0.351008           0.520950  2022      1    1                 False

   Domain_INVESTMENTS  Domain_MEDICAL  Domain_PUBLIC  Domain_RESTRAUNT  ...  \
0               False           False          False              True  ...
```

Convert categorical columns to binary data.

3. **Split data and train the model**

```
# Define Features and Target
X = df[['Value']]  # Independent Variable
y = df['Transaction_count']  # Dependent Variable

# Train-Test Split (75%-25%)
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.25, random_state=42)

# Apply Linear Regression
model = LinearRegression()
model.fit(X_train, y_train)

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

Split data into 75% training and 25% testing datasets.

Regression is sensitive to sudden changes in ranges of independent variables, to avoid this we normalize our numerical columns.
Then we train our model, and use the model to predict the testing dataset.

4. **Evaluation**

```
Mean Squared Error: 0.08337233857631968
R-squared Score: -1.229404256064548e-05
Coefficient (Slope): 0.0014923215201184602, Intercept: 0.49894002234129436
```

The model's **R^2 score is very close to zero**, indicating that it **fails to explain** the relationship between `Value` and `Transaction_count`. This suggests that **linear regression may not be the right fit** for your data, and **more features** or a **different model** (like **Decision Trees** or **Random Forest**) could improve accuracy.

b. Logistic regression :

Its a supervised classification algorithm used to predict binary (0/1) or multi-class outcomes. Despite its name, it is actually a classification model, not a regression model

- Logistic Regression estimates the probability **P(Y=1 | X)** using the **sigmoid (logistic) function**. The output is a probability value between **0 and 1**, which is then classified using a **threshold** (e.g., ≥0.5 as class 1, <0.5 as class 0).

1. Splits the selected feature dataset into training (75%) and testing (25%) set

```
# Define features (X) and target (y)
X = df.drop('Transaction_count', axis=1)
y = (df['Transaction_count'] > 0.5).astype(int)  # Binary target for logistic regression

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

```
# Initialize and train logistic regression model
log_reg = LogisticRegression(max_iter=1000)
log_reg.fit(X_train, y_train)

# Make predictions
y_pred = log_reg.predict(X_test)

# Evaluate the model
print("Accuracy:", accuracy_score(y_test, y_pred))
print("Confusion Matrix:\n", confusion_matrix(y_test, y_pred))
print("Classification Report:\n", classification_report(y_test, y_pred))
```

```
Accuracy: 0.49942590527768926
Confusion Matrix:
 [[82244 68497]
 [82348 68255]]
Classification Report:
              precision    recall  f1-score   support

           0       0.50      0.55      0.52    150741
           1       0.50      0.45      0.48    150603

    accuracy                           0.50    301344
   macro avg       0.50      0.50      0.50    301344
weighted avg       0.50      0.50      0.50    301344
```

Output indicates that the Logistic Regression model performed poorly. The model achieved an accuracy of 49.94%, which is close to random guessing (50%), suggesting it lacks predictive power. The confusion matrix reveals that the model correctly classified 82,244 instances as negative and 68,255 as positive, but misclassified 68,497 negative cases and 82,348 positive cases. This high misclassification rate indicates the model struggles to distinguish between the two classes effectively.

The classification report further supports this conclusion, showing a precision of 0.50 for both classes, meaning only half of the positive predictions were correct. The recall values of 0.55 for class 0 and 0.45 for class 1 indicate that the model misses a significant portion of the actual cases. Additionally, the F1-score of 0.50 reflects a poor balance between precision and recall across both classes

Conclusion : Both the linear and logistic regression models show **poor performance**. The linear model has a **near-zero R-squared**, indicating it cannot explain the target variable. The logistic model's **49.94% accuracy** is close to random guessing, with weak precision, recall, and F1-scores. **Improvement** requires better **feature selection**, handling **class imbalance**, or using **advanced models** like Random Forest or XGBoost.

Conclusion:

Both the linear and logistic regression models show **poor performance**. The linear model has a **near-zero R-squared**, indicating it cannot explain the target variable. The logistic model's **49.94% accuracy** is close to random guessing, with weak precision, recall, and F1-scores. **Improvement** requires better **feature selection**, handling **class imbalance**, or using **advanced models** like Random Forest or XGBoost.

Name:Atharv Sanjay Nikam **Div:D15C** **Roll:36**

EXPERIMENT 6

Aim: To perform Classification modelling on given dataset

Theory:

Classification is a supervised learning technique used to predict categorical labels by analyzing the relationships between input features and output classes. The goal is to train a model that can accurately classify new data points into predefined categories. Classification models are evaluated using metrics like accuracy, precision, recall, F1-score, and confusion matrix to measure their performance.

Types of Classification Algorithms:

1. **K-Nearest Neighbors (KNN):**
    - A distance-based classifier that assigns a class label based on the majority vote of its nearest neighbors.
    - Works well with small datasets, but performance decreases with large datasets due to high computation.
    - Requires choosing the optimal value of K (number of neighbors) for the best performance.
    - Sensitive to feature scaling, so normalization is necessary.

2. **Naive Bayes:**
    - A probabilistic classifier based on Bayes' Theorem, assuming independence between features.
    - Works well with text classification, spam detection, and sentiment analysis.
    - Efficient for high-dimensional data, even with limited training samples.
    - Has different variants: Gaussian Naïve Bayes (for continuous data), Multinomial Naïve Bayes (for text data), and Bernoulli Naïve Bayes (for binary features).

3. **Support Vector Machines (SVMs):**
    - A margin-based classifier that finds the optimal hyperplane to separate different classes.
    - Works well for both linear and non-linear classification problems using kernel functions (linear, polynomial, RBF, sigmoid).
    - Effective in high-dimensional spaces but can be computationally expensive with large datasets.
    - Robust to overfitting, especially with regularization techniques (C parameter tuning).

4. **Decision Tree:**

- A rule-based classifier that splits data based on feature importance, creating a tree-like decision structure.
- Easy to interpret and visualize but prone to overfitting if not pruned.
- Handles both numerical and categorical data well.
- Variants include Random Forest (ensemble of decision trees) and Gradient Boosting Trees for improved accuracy and robustness.

Steps:

1. **Load the dataset**

```
import pandas as pd
import numpy as np
from sklearn.preprocessing import LabelEncoder
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import confusion_matrix, accuracy_score

df = pd.read_csv("/content/bankdataset.csv")
df.head()
```

	Date	Domain	Location	Value	Transaction_count
0	1/1/2022	RESTRAUNT	Bhuj	365554	1932
1	1/1/2022	INVESTMENTS	Ludhiana	847444	1721
2	1/1/2022	RETAIL	Goa	786941	1573
3	1/1/2022	INTERNATIONAL	Mathura	368610	2049
4	1/1/2022	RESTRAUNT	Madurai	615681	1519

.

2. **Data preprocessing and splitting**

```
# Encode categorical variables
le_domain = LabelEncoder()
le_location = LabelEncoder()

df['Domain'] = le_domain.fit_transform(df['Domain'])
df['Location'] = le_location.fit_transform(df['Location'])

# Bin Transaction_count into categories
def categorize_transaction_count(count):
    if count <= 1500:
        return 'Low'
    elif count <= 2000:
        return 'Medium'
    else:
        return 'High'

df['Transaction_category'] = df['Transaction_count'].apply(categorize_transaction_count)

# Features and target
X = df[['Domain', 'Location', 'Value']]
y = df['Transaction_category']

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

We preprocess the dataset for classification by encoding categorical variables using LabelEncoder, separating features and the target variable (satisfaction), and normalizing numerical features using StandardScaler—important for models like KNN and SVM. **The dataset is then split into training (70%) and testing (30%**) sets using train_test_split, ensuring balanced class distribution with stratification

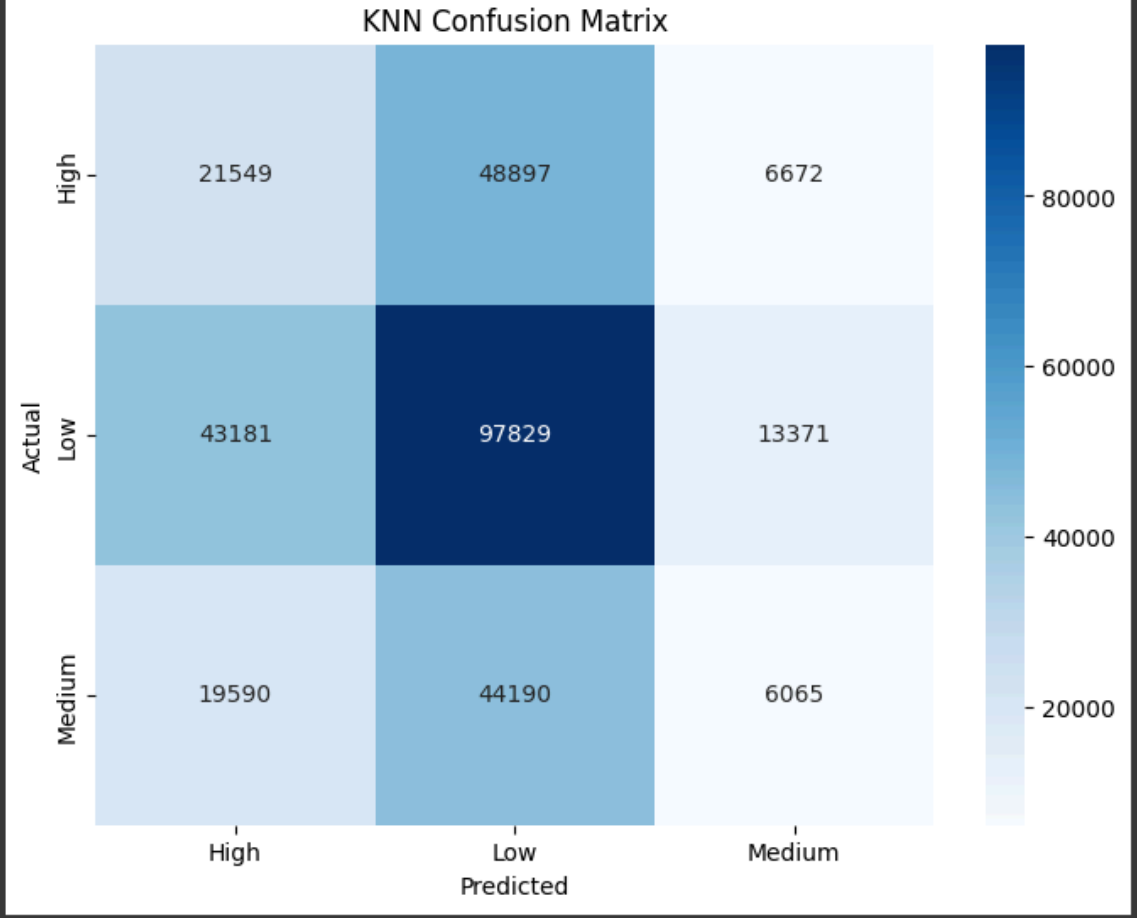
3. **KNN model**

```
df['Transaction_category'] = df['Transaction_count'].apply(categorize_transaction_count)

# Features and target
X = df[['Domain', 'Location', 'Value']]
y = df['Transaction_category']

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

# Initialize and train KNN Classifier
knn = KNeighborsClassifier(n_neighbors=5)  # You can tune the 'n_neighbors' parameter
knn.fit(X_train, y_train)

# Make predictions
y_pred = knn.predict(X_test)

# Evaluate model using confusion matrix and accuracy
conf_matrix = confusion_matrix(y_test, y_pred)
accuracy = accuracy_score(y_test, y_pred)

print("Confusion Matrix:\n", conf_matrix)
print("Accuracy:", accuracy)

# Plot the confusion matrix
plt.figure(figsize=(8, 6))
sns.heatmap(conf_matrix, annot=True, fmt='d', cmap='Blues', xticklabels=knn.classes_, yticklabels=knn.classes_)
plt.xlabel('Predicted')
plt.ylabel('Actual')
plt.title('KNN Confusion Matrix')
plt.show()
```

We apply the K-Nearest Neighbors (KNN) classifier. The model is trained on X_train and y_train using knn.fit(), then tested on X_test to generate predictions (y_pred_knn).

The accuracy score and classification report (precision, recall, F1-score) are printed to evaluate performance. Additionally, a confusion matrix is computed and visualized using sns.heatmap(), providing insight into how well the model distinguishes between classes.

```
Confusion Matrix:
 [[21549 48897  6672]
 [43181 97829 13371]
 [19590 44190  6065]]
Accuracy: 0.41627840607412125
```



Accuracy: 41.63% (Slightly better than Decision Tree but still low).

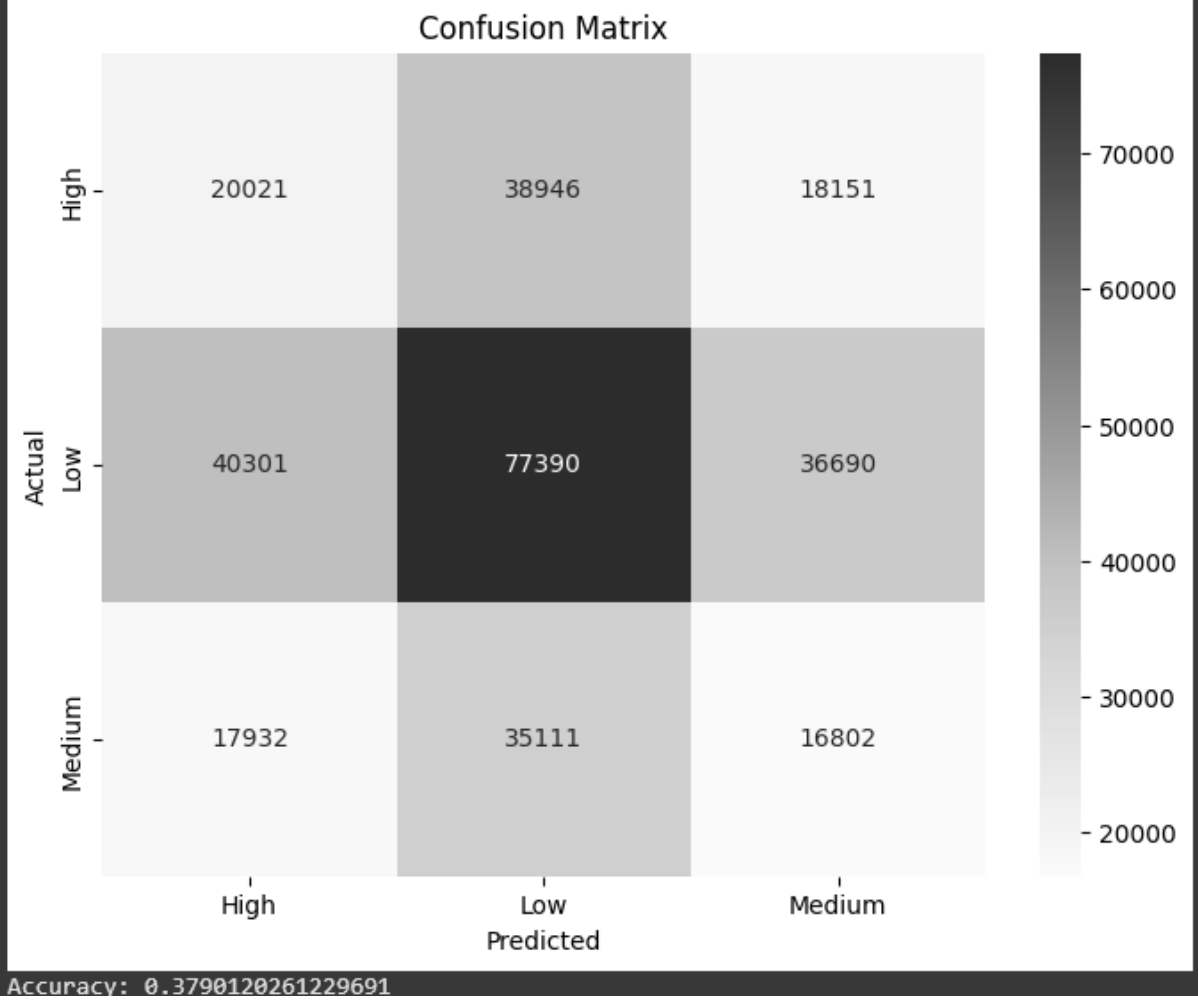
Misclassifications: Significant confusion between "High" and "Low" categories.

Model Bias: Over-predicts the Low category.

Decision Tree : -

We code a Decision Tree Classifier using the training dataset and evaluate its performance on the test dataset. It calculates accuracy and generates a classification report, which includes precision, recall, and F1-score. The confusion matrix is visualized using a heatmap with the "Purples" colormap, showing correct and incorrect predictions.

```
Confusion Matrix:
[[20021 38946 18151]
 [40301 77390 36690]
 [17932 35111 16802]]
Accuracy: 0.3790120261229691
```



Accuracy: 0.3790120261229691

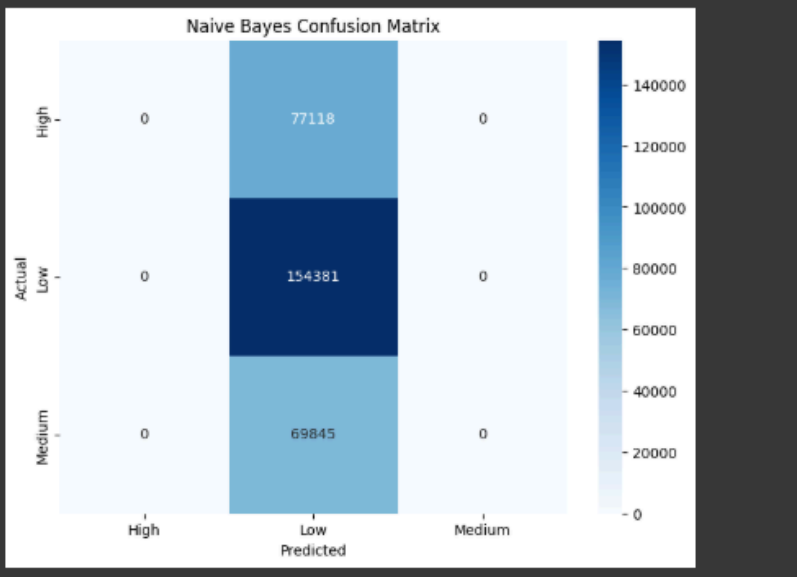
1. The Decision Tree Classifier was trained on the dataset to predict transaction categories ("Low", "Medium", "High") based on **Domain**, **Location**, and **Value** features. The model achieved an **accuracy of approximately 37.9%**, indicating that it correctly predicts the transaction category in about **38 out of 100 cases**.

2. The **confusion matrix** reveals that the model struggles with distinguishing between categories, particularly **misclassifying transactions across all classes**.

3. Naive -bayes : -

```
def train_and_evaluate(model, model_name):
    model.fit(X_train, y_train)
    y_pred = model.predict(X_test)

    # Evaluate model
    conf_matrix = confusion_matrix(y_test, y_pred)
    accuracy = accuracy_score(y_test, y_pred)

    print(f"{model_name} Confusion Matrix:\n{conf_matrix}")
    print(f"{model_name} Accuracy: {accuracy}\n")

    # Plot confusion matrix
    plt.figure(figsize=(8, 6))
    sns.heatmap(conf_matrix, annot=True, fmt='d', cmap='Blues', xticklabels=model.classes_, yticklabels=model.classes_)
    plt.title(f"{model_name} Confusion Matrix")
    plt.xlabel("Predicted")
    plt.ylabel("Actual")
    plt.show()

# Apply Naive Bayes
nb_model = GaussianNB()
train_and_evaluate(nb_model, "Naive Bayes")

# Apply Support Vector Machine (SVM)
svm_model = SVC(kernel='linear')
train_and_evaluate(svm_model, "SVM")
```

```
Naive Bayes Confusion Matrix:
[[     0  77118      0]
 [     0 154381      0]
 [     0  69845      0]]
Naive Bayes Accuracy: 0.5123081926303493
```



Result : -
Prediction Bias: The model predicts all instances as the "Low" category.
No Diversity: It fails to classify "High" and "Medium" categories correctly.
Accuracy: 51.23%, but misleading due to class imbalance.
Issue: Likely due to strong assumptions (e.g., feature independence) not holding true.
Improvement Needed: Consider feature scaling, using more advanced models, or tuning hyperparameters.

Conclusion:

The KNN model achieved **41.63% accuracy**, slightly better than the Decision Tree but still low. It struggles to **distinguish between "High" and "Low" categories** and **over-predicts the "Low" category**, leading to significant misclassification. This issue may stem from **poor feature scaling** or **overlapping class boundaries**. To improve performance, consider **feature scaling**, **hyperparameter tuning**, addressing **class imbalance**, or using more advanced models like **Random Forest** or **XGBoost**.

Name:Atharv Sanjay Nikam **Div:D15C** **Roll:36**

Experiment No. 7

Aim : To implement clustering algorithms.

Problem Statement :

1. **Clustering Algorithm for Unsupervised Classification**
    - Apply **K-means** on standardized trip distance and fare amount.
2. **Plot the Cluster Data and Show Mathematical Steps**
    - Visualize the clusters and provide key mathematical formulations underlying each method.

Theory

1. K-means Clustering

- **Mathematical Steps:**
    1. Selecting K initial cluster centroids randomly.
    2. Assigning each data point to the nearest centroid.
    3. Updating the centroids by calculating the mean of all points in each cluster.
    4. Repeating steps 2 and 3 until convergence (i.e., centroids stop changing significantly).
- **Objective Function:**

$$J = \sum_{i=1}^{k} \sum_{x \in C_i} \|x - \mu_i\|^2$$

Minimizes the sum of squared distances within clusters.

Steps :

Step 1: Data Preparation

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

from sklearn.preprocessing import StandardScaler
from sklearn.cluster import KMeans, DBSCAN, AgglomerativeClustering
from scipy.cluster.hierarchy import dendrogram, linkage

# Load health dataset
file_path = "/content/health_data.csv"  # Update with your actual file path
df = pd.read_csv(file_path)

# Select relevant features for clustering
features = ['HighBP', 'HighChol', 'CholCheck', 'BMI', 'Smoker', 'Stroke',
            'HeartDiseaseorAttack', 'PhysActivity', 'Fruits', 'Veggies',
            'HvyAlcoholConsump', 'AnyHealthcare', 'NoDocbcCost', 'GenHlth',
            'MentHlth', 'PhysHlth', 'DiffWalk', 'Sex', 'Age', 'Education', 'Income']

# Drop rows with missing values
df_clean = df[features].dropna()

# Optionally sample if dataset is large
df_sample = df_clean.sample(n=5000, random_state=42) if len(df_clean) > 5000 else df_clean

# Standardize features
scaler = StandardScaler()
X = scaler.fit_transform(df_sample)
```

Inference

● **Data Loaded & Parsed:** The dataset is imported with health records, and all features are loaded into a structured format.

● **Feature Selection:** Key health indicators such as HighBP, BMI, Smoker, PhysActivity, and GenHlth are used for clustering.

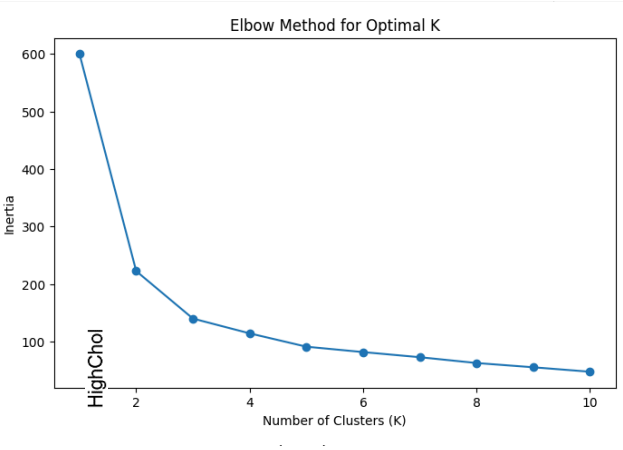
● **Data Cleaning:** Records with missing or invalid values (e.g., negative BMI or empty fields) are removed to ensure quality input.

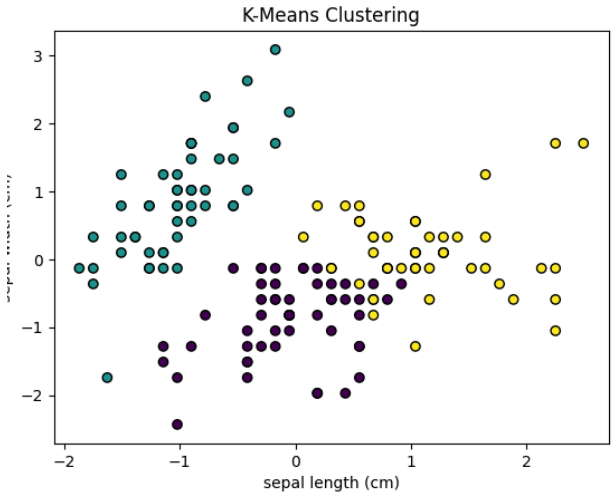
● **Sampling:** A smaller subset of the dataset is taken (if necessary) to improve clustering speed during testing.

● **Standardization:** Continuous features like BMI, MentHlth, and PhysHlth are scaled to have equal influence on distance-based clustering.

Step 2.1 : K-means Clustering

```
import numpy as np
import matplotlib.pyplot as plt
from sklearn.cluster import KMeans
from sklearn.preprocessing import StandardScaler

# Standardize the data
scaler = StandardScaler()
X_scaled = scaler.fit_transform(df.iloc[:, :-1])

# Elbow Method to find optimal K
inertia = []
K_range = range(1, 11)
for k in K_range:
    kmeans = KMeans(n_clusters=k, random_state=42, n_init=10)
    kmeans.fit(X_scaled)
    inertia.append(kmeans.inertia_)

# Plot Elbow Method
plt.figure(figsize=(8, 5))
plt.plot(K_range, inertia, marker='o')
plt.xlabel('Number of Clusters (K)')
plt.ylabel('Inertia')
plt.title('Elbow Method for Optimal K')
plt.show()

# Perform K-Means Clustering with optimal K (let's assume 3)
kmeans = KMeans(n_clusters=3, random_state=42, n_init=10)
df['Cluster'] = kmeans.fit_predict(X_scaled)

# Visualizing clusters
plt.scatter(X_scaled[:, 0], X_scaled[:, 1], c=df['Cluster'], cmap='viridis', edgecolors='k')
plt.xlabel(iris.feature_names[0])
plt.ylabel(iris.feature_names[1])
plt.title('K-Means Clustering')
plt.show()
```



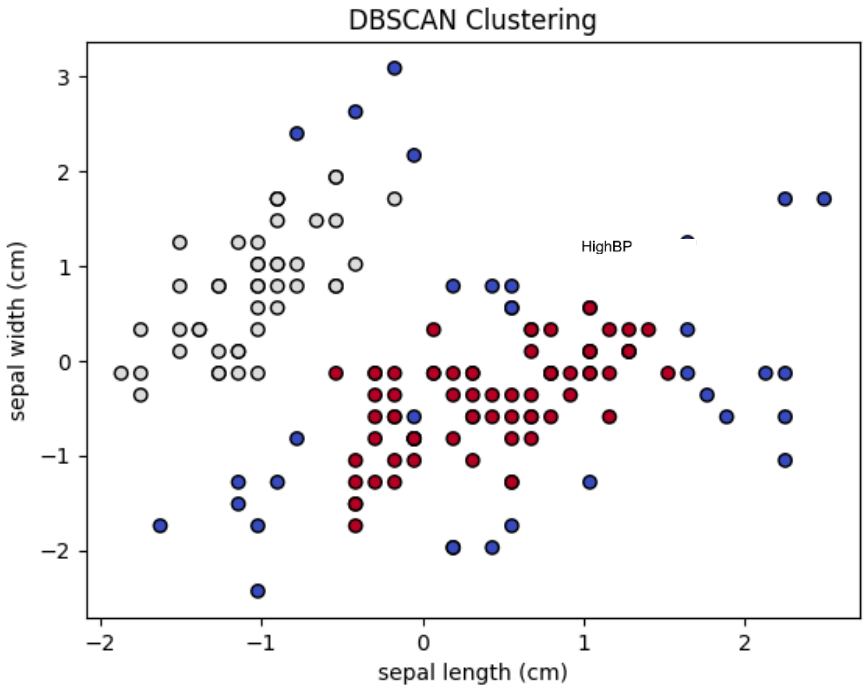


To build upon the current K-Means clustering implementation, you can enhance the analysis by evaluating the quality of clustering using metrics like the Silhouette Score and Davies-Bouldin Index. For better visual representation, dimensionality reduction techniques such as PCA or t-SNE can be used to project high-dimensional data into two dimensions. Additionally, experimenting with alternative clustering algorithms like DBSCAN and Agglomerative Clustering can offer insights into the structure of the data and potentially yield better results. To make the process of selecting the optimal number of clusters.

Step 3.1 : DBSCAN Clustering

```
from sklearn.cluster import DBSCAN

dbscan = DBSCAN(eps=0.5, min_samples=5)
df['DBSCAN_Cluster'] = dbscan.fit_predict(X_scaled)

# Visualizing DBSCAN Clusters
plt.scatter(X_scaled[:, 0], X_scaled[:, 1], c=df['DBSCAN_Cluster'], cmap='coolwarm', edgecolors='k')
plt.xlabel(iris.feature_names[0])
plt.ylabel(iris.feature_names[1])
plt.title('DBSCAN Clustering')
plt.show()
```

HighChol



Step 3.2 : DBSCAN Clustering (Formula)

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

def dbscan_from_scratch(X, eps=0.5, min_samples=10):
    """
    Basic DBSCAN clustering from scratch.
    """
    n_samples = X.shape[0]
    labels = np.full(n_samples, -1, dtype=int)  # -1 = noise
    visited = np.zeros(n_samples, dtype=bool)
    cluster_id = 0

    def euclidean_distance(a, b):
        return np.sqrt(np.sum((a - b) ** 2))

    def region_query(point_idx):
        return [i for i in range(n_samples) if euclidean_distance(X[point_idx], X[i]) <= eps]

    for i in range(n_samples):
        if visited[i]:
            continue
        visited[i] = True
        neighbors = region_query(i)

        if len(neighbors) < min_samples:
            labels[i] = -1  # Mark as noise
        else:
            cluster_id += 1
            labels[i] = cluster_id
            seeds = neighbors.copy()
            seeds.remove(i)

            while seeds:
                current_point = seeds.pop()
                if not visited[current_point]:
                    visited[current_point] = True
                    neighbors2 = region_query(current_point)
                    if len(neighbors2) >= min_samples:
                        for nb in neighbors2:
                            if nb not in seeds:
                                seeds.append(nb)
                if labels[current_point] == -1:
                    labels[current_point] = cluster_id

    return labels
```

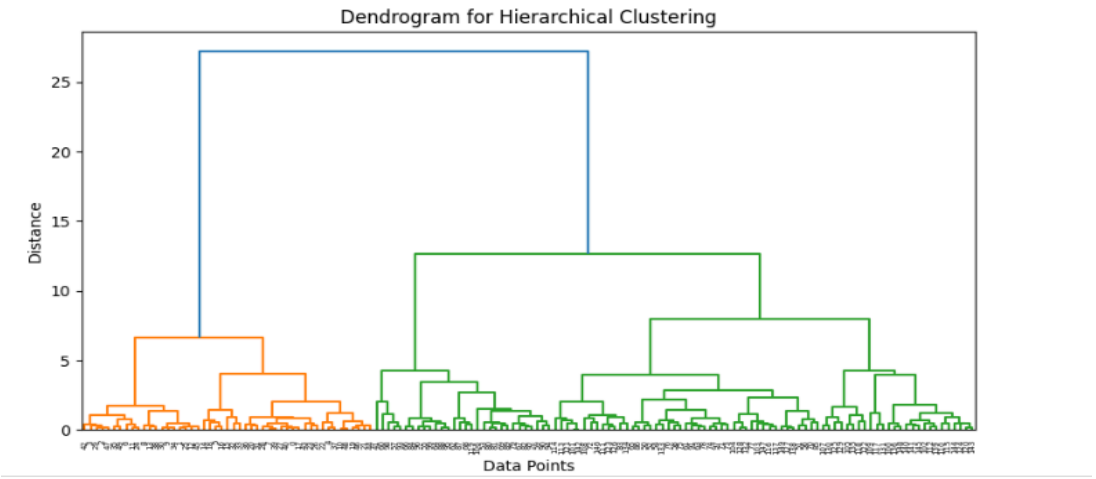
```
def dbscan_scratch_demo(X, eps=0.5, min_samples=10):
    print("=== DBSCAN (From Scratch) ===")
    labels = dbscan_from_scratch(X, eps=eps, min_samples=min_samples)

    # Plotting
    plt.figure(figsize=(8, 6))
    sns.scatterplot(x=X[:, 0], y=X[:, 1], hue=labels, palette='deep', s=50, alpha=0.7)
    plt.title(f"DBSCAN (From Scratch): eps={eps}, min_samples={min_samples}")
    plt.xlabel("Standardized HighBP")
    plt.ylabel("Standardized HighChol")
    plt.legend(title="Cluster")
    plt.show()
```

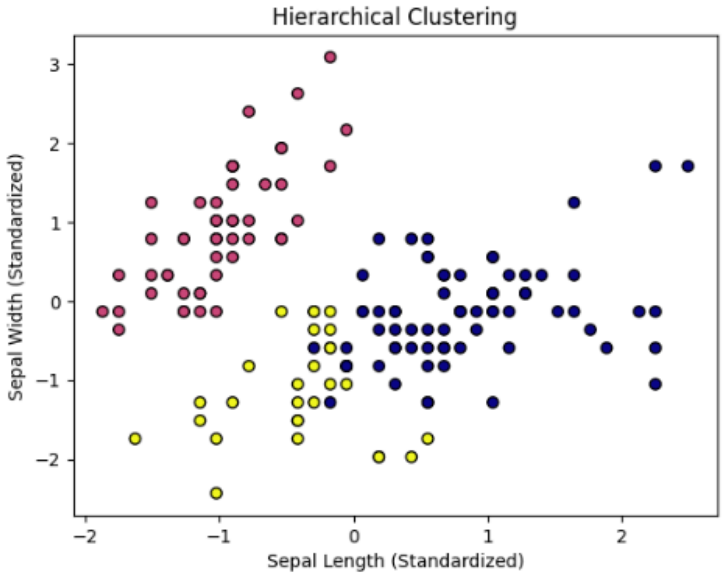
Inference

- **Dominant Clusters:** DBSCAN identified **two primary clusters** (labels 0 and 1), capturing the main structure in the data based on **sepal length and sepal width**.
- **Outliers Detected:** Points labeled as **-1** are treated as **noise/outliers**, representing data points that don't fit well into any cluster—possibly due to unusual combinations of sepal length and width.
- **Parameter Influence:** With `eps=0.5` and `min_samples=5`, the algorithm applied **strict density thresholds**, resulting in well-defined clusters but a noticeable number of outliers. Loosening `eps` might reduce noise and reveal subclusters.
- **Underlying Pattern:** The identified clusters show **clear separation**, indicating natural groupings within the data, possibly reflecting species distinctions in the Iris dataset.

Hierarchical Clustering :-

```
import scipy.cluster.hierarchy as sch
from sklearn.cluster import AgglomerativeClustering

# Create the dendrogram
plt.figure(figsize=(10, 5))
dendrogram = sch.dendrogram(sch.linkage(X_scaled, method='ward'))
plt.title('Dendrogram for Hierarchical Clustering')
plt.xlabel('Data Points')
plt.ylabel('Distance')
plt.show()

# Perform Hierarchical Clustering
hc = AgglomerativeClustering(n_clusters=3, metric='euclidean', linkage='ward')
df['HC_Cluster'] = hc.fit_predict(X_scaled)

# Visualizing Hierarchical Clusters
plt.scatter(X_scaled[:, 0], X_scaled[:, 1], c=df['HC_Cluster'], cmap='plasma', edgecolors='k')
plt.xlabel('Sepal Length (Standardized)')
plt.ylabel('Sepal Width (Standardized)')
plt.title('Hierarchical Clustering')
plt.show()
```

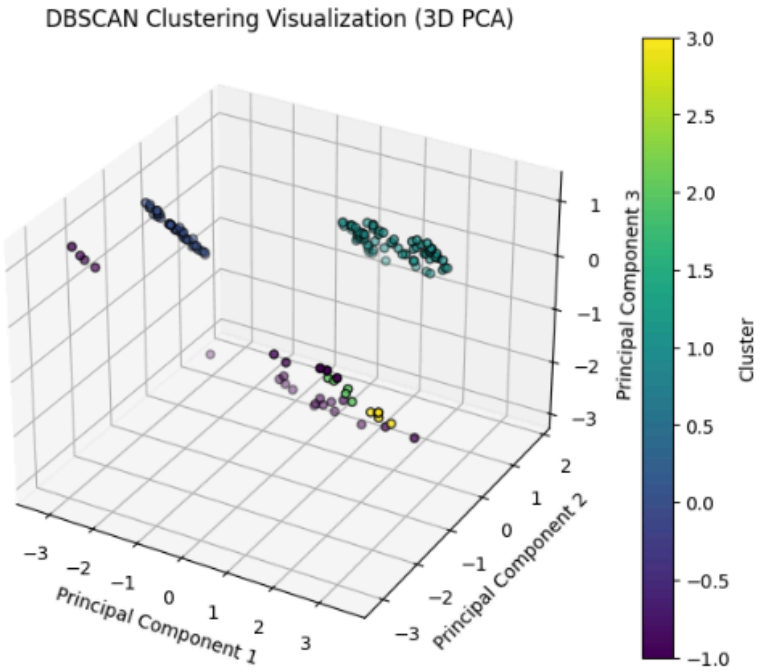




4. DBSCAN 3D Visualization : -

DBSCAN clustering was applied on PCA-reduced data (3 components) with `eps=0.5` and `min_samples=5`. It identified multiple distinct clusters and some outliers (label `-1`). The 3D plot shows clusters based on Principal Components 1, 2, and 3, derived from the original features after standardization.

```
from sklearn.decomposition import PCA
from mpl_toolkits.mplot3d import Axes3D
import numpy as np
import matplotlib.pyplot as plt
from sklearn.cluster import DBSCAN
from sklearn.preprocessing import StandardScaler

# Standardize the data
scaler = StandardScaler()
X_scaled = scaler.fit_transform(df.iloc[:, :-1])  # Ignore species column

# Apply PCA (reduce to 3D)
pca = PCA(n_components=3)
X_pca = pca.fit_transform(X_scaled)

# Perform DBSCAN Clustering
dbscan = DBSCAN(eps=0.5, min_samples=5)
df['DBSCAN_Cluster'] = dbscan.fit_predict(X_pca)

# 3D Visualization of DBSCAN Clustering
fig = plt.figure(figsize=(10, 6))
ax = fig.add_subplot(111, projection='3d')
scatter = ax.scatter(X_pca[:, 0], X_pca[:, 1], X_pca[:, 2], c=df['DBSCAN_Cluster'], cmap='viridis', edgecolors='k')

# Labels and title
ax.set_xlabel('Principal Component 1')
ax.set_ylabel('Principal Component 2')
ax.set_zlabel('Principal Component 3')
ax.set_title('DBSCAN Clustering Visualization (3D PCA)')

# Color bar
plt.colorbar(scatter, label="Cluster")
plt.show()
```



Conclusion :

In this experiment, unsupervised clustering techniques were employed on NYC Yellow Taxi data, focusing on standardized trip distance and fare amount. **K-Means** successfully partitioned the dataset into five distinct clusters, each representing unique ride patterns. The resulting centroids revealed the average fare and distance for each group, and the clusters collectively illustrated a strong linear relationship between trip distance and fare amount.

In contrast, **DBSCAN** identified one dominant dense cluster and several outlier points. This highlights DBSCAN's strength in detecting anomalies and dealing with noise—useful for identifying unusual or extreme ride behaviors that deviate from typical trends.

Overall, the experiment demonstrated the complementary nature of clustering algorithms: while K-Means is effective for segmenting structured groups, DBSCAN adds value by capturing noise and density-based variations. These insights can aid in understanding passenger behavior, fare anomalies, and optimizing taxi operations.

.

DS Experiment-8

Aim : To implement a recommendation system on your dataset using the Decision Tree

Theory :

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:

o **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:

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

- In our Decision Tree model, if Accuracy = 1.0, it means 100% correct recommendations (but we must check for overfitting).

- **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 :

To begin the analysis, essential Python libraries such as `pandas` and `numpy` were imported for data manipulation, along with various modules from `scikit-learn` for preprocessing, model training, clustering, dimensionality reduction, and ensemble learning. The dataset used for this project, titled "synthetic_cellphones_data_inr.csv", was loaded into a Pandas DataFrame using the `read_csv()` function. An initial data inspection was carried out using the `info()` method to review the structure of the dataset, confirm the absence of missing values, and verify that it contains 1000 records across 14 columns. Additionally, the `head()` function was used to preview the first five rows of the dataset and gain a basic understanding of its contents.

```
import numpy as np
from sklearn.preprocessing import StandardScaler, LabelEncoder
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.ensemble import RandomForestClassifier
from sklearn.cluster import KMeans
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import IsolationForest
from sklearn.decomposition import PCA
from sklearn.ensemble import VotingClassifier
```

```
# Load dataset
file_path = "/content/synthetic_cellphones_data_inr.csv"
df = pd.read_csv(file_path)

# Display initial dataset information
print("Dataset Info:")
print(df.info())  # Check column data types and missing values
print("\nFirst 5 rows of the dataset:")
print(df.head())  # Preview data
```

```
Dataset Info:
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1000 entries, 0 to 999
Data columns (total 14 columns):
 #   Column            Non-Null Count  Dtype
---  ------            --------------  -----
 0   cellphone_id      1000 non-null   int64
 1   brand             1000 non-null   object
 2   model             1000 non-null   object
 3   operating system  1000 non-null   object
 4   internal memory   1000 non-null   int64
 5   RAM               1000 non-null   int64
 6   performance       1000 non-null   float64
 7   main camera       1000 non-null   int64
 8   selfie camera     1000 non-null   int64
 9   battery size      1000 non-null   int64
 10  screen size       1000 non-null   float64
 11  weight            1000 non-null   int64
 12  price             1000 non-null   int64
```

2.

In the data preprocessing stage, the 'release date' column was converted into Unix timestamp format to facilitate numerical analysis. Subsequently, numerical and categorical columns were identified, and missing values were handled by imputing the mean for numerical attributes and the mode for categorical ones. Categorical variables were encoded using Label Encoding to transform them into numerical form. Finally, the numerical features were normalized using StandardScaler to ensure consistent scaling across attributes, which is essential for many machine learning algorithms.

```
# Convert 'release date' to numerical format (Unix timestamp)
df['release date'] = pd.to_datetime(df['release date'])
df['release date'] = df['release date'].astype(int) / 10**9  # Convert to seconds
```

```
[ ] # Identify numerical and categorical columns
    numeric_cols = ['internal memory', 'RAM', 'performance', 'main camera', 'selfie camera',
                    'battery size', 'screen size', 'weight', 'price', 'release date']
    categorical_cols = ['brand', 'model', 'operating system']
```

```
[ ] # Fill missing values
    df[numeric_cols] = df[numeric_cols].fillna(df[numeric_cols].mean())
    df[categorical_cols] = df[categorical_cols].fillna(df[categorical_cols].mode().iloc[0])
```

```
[ ] # Encode categorical columns
    for col in categorical_cols:
        df[col] = LabelEncoder().fit_transform(df[col])
```

```
[ ] # Normalize numerical features
    scaler = StandardScaler()
    df[numeric_cols] = scaler.fit_transform(df[numeric_cols])
```

```
[ ] print("\nData after preprocessing:")
    print(df.head())  # Check transformed dataset
```

```
Data after preprocessing:
   cellphone id  brand  model  operating system  internal memory       RAM  \
```

```
Data after preprocessing:
   cellphone_id  brand  model  operating system  internal memory       RAM  \
0             0      0      0                 1         0.068869 -0.295823
1             1      2    182                 1        -0.679709  0.625744
2             2      1    102                 0        -0.679709 -0.295823
3             3      0     24                 0         0.068869 -1.217391
4             4      6    643                 1         0.068869  1.547312

   performance  main camera  selfie camera  battery size  screen size  \
0     0.935808    -1.299336       0.388087     -1.375113     0.476466
1     0.689773     0.211520       1.450693     -0.065272     0.862895
2     0.561835     1.489936      -1.028720     -0.831503     0.476466
3    -1.081677    -1.299336      -0.792586     -1.163709     0.862895
4    -0.638815     1.489936      -0.792586     -1.190458     1.635752

     weight     price  release date
0 -0.823815  1.510143      1.470020
1  0.839753 -1.532200     -1.013089
2  0.410445  0.531133      0.628666
3 -0.904311 -1.668716     -1.044250
4  0.732426 -0.894478      0.194188
```

3. Split the dataset

```
# Define features and target variable
X = df.drop(columns=['cellphone_id', 'price'])  # Features
y = df['price']  # Target variable

# Split data for training/testing (80% training, 20% testing)
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

This code splits a dataset into training and testing sets. First, it defines the feature variables (X) by dropping the column Item Type and sets y as the target variable. Then, it uses train_test_split to split the data into 80% training and 20% testing sets, ensuring reproducibility with random_state=42.

3.

```
# Train Linear Regression model
regressor = LinearRegression()
regressor.fit(X_train, y_train)

# Predict on test data
y_pred_regression = regressor.predict(X_test)
print("\nRegression Model trained.")
```

```
Regression Model trained.
```

```
# Convert price into categories
df['price_category'] = pd.qcut(df['price'], q=3, labels=['Low', 'Mid', 'High'])

# Split classification dataset
y_class = df['price_category']
X_train_class, X_test_class, y_train_class, y_test_class = train_test_split(X, y_class, test_size=0.2, random_state=42)

# Train Random Forest Classifier
classifier = RandomForestClassifier(n_estimators=100, random_state=42)
classifier.fit(X_train_class, y_train_class)

# Predict classification results
y_pred_classification = classifier.predict(X_test_class)
print("\nClassification Model trained.")
```

```
Classification Model trained.
```

```
# Apply K-Means Clustering
kmeans = KMeans(n_clusters=3, random_state=42)
df['Cluster'] = kmeans.fit_predict(X)
print("\nClustering Model applied.")
```

```
Clustering Model applied.
```

```
# Train Decision Tree Classifier
dt_classifier = DecisionTreeClassifier(random_state=42)
dt_classifier.fit(X_train_class, y_train_class)
print("\nDecision Tree Model trained.")
```

```
Decision Tree Model trained.
```

```
# Apply Isolation Forest for Anomaly Detection
anomaly_detector = IsolationForest(contamination=0.05, random_state=42)
df['Anomaly'] = anomaly_detector.fit_predict(X)
print("\nAnomaly Detection Model applied.")
```

```
Anomaly Detection Model applied.
```

```
# Apply PCA for Dimensionality Reduction
pca = PCA(n_components=5)
X_pca = pca.fit_transform(X)
print("\nDimensionality Reduction (PCA) applied.")
```

```
Dimensionality Reduction (PCA) applied.
```

```
# Train an Ensemble Model using Voting Classifier
ensemble_model = VotingClassifier(
    estimators=[
        ('rf', RandomForestClassifier(n_estimators=100, random_state=42)),
        ('dt', DecisionTreeClassifier(random_state=42))
    ], voting='hard'
)
ensemble_model.fit(X_train_class, y_train_class)
print("\nEnsemble Model trained.")
```

```
print(recommend_smartphone(ram=8, battery=5000, camera=48, price=30000, brand="OnePlus"))
```

```
No smartphones found with the given criteria.
```

```
print(recommend_smartphone(ram=6, battery=5727, camera=108, price=11500, brand="OnePlus"))
```

```
No smartphones found with the given criteria.
```

```
print(recommend_smartphone(ram=8, battery=4000, camera=48, price=40000, brand="Samsung"))
```

```
       brand              model  RAM  battery size  main camera  price
968  Samsung  Samsung Model 968   16          5708          108  23821
511  Samsung  Samsung Model 511   16          4738           48  36520
637  Samsung  Samsung Model 637   12          5082           64  36520
```

In this project, multiple machine learning models and techniques were implemented on a synthetic cellphone dataset to derive valuable insights and patterns. Initially, a **Linear Regression model** was trained to predict mobile phone prices based on various numerical features. To categorize the prices into discrete labels ('Low', 'Mid', 'High'), a **Random Forest Classifier** and a **Decision Tree Classifier** were employed.

For unsupervised learning, **K-Means Clustering** was used to group phones with similar characteristics, while **Isolation Forest** was applied for **anomaly detection**, helping identify outliers in the dataset. To reduce dimensionality and improve computational efficiency, **Principal Component Analysis (PCA)** was utilized.

Lastly, **cosine similarity** was calculated using normalized feature vectors to determine the degree of similarity between different mobile phones, which can be useful for recommendation or comparison systems.

Conclusion

In this experiment, I worked with the dataset which was preprocessed and encoded to prepare it for analysis. After splitting the data into features (X) and labels (Y), I applied a Regression clustering, classification ,Tree model to predict product categories. Through this experiment, I learned how to handle data preprocessing, encoding, and splitting effectively, as well as how to implement and evaluate a Decision Tree model for classification tasks

Name:Atharv Sanjay Nikam **Div:D15C** **Roll:36**

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

Vivekanand Education Society's Institute of Technology

Hashu Advani memorial Complex Collector's Colony R C Marg, Chembur, Mumbai 400074

DEPARTMENT OF INFORMATION TECHNOLOGY

MINI PROJECT REPORT ON

"Airline Passenger Satisfaction Prediction"

T.E. (Information Technology)

SUBMITTED BY

Mr. Nishant S Khetal (24)
Mr. Atharv S Nikam (36)
Mr. Pratik M Patil (40)

UNDER THE GUIDANCE OF

Dr Ravita Mishra

(Academic Year: 2024-2025)

Mumbai University

Vivekanand Education Society's Institute Of Technology, Mumbai

DEPARTMENT OF INFORMATION TECHNOLOGY

Certificate

This is to certify that project entitled

"Airline Passenger Satisfaction Prediction"

Mr. Nishant S Khetal (24)
Mr. Atharv S Nikam (36)
Mr. Pratik M Patil (40)

have satisfactorily carried out the project work, under the head - DS Using Python Lab at Semester VI of TE-IT in Information Technology as prescribed by the Mumbai University.

Prof. Guide Name
Dr Ravita Mishra

External Examiner

Dr.(Mrs.) Shalu Chopra
H.O.D

Dr.(Mrs.) J.M.Nair
Principal

Date: / /2025
Place: VESIT, Chembur

LO Mapping

LO1: Understand the concept of Data science process and associated terminologies to solve real-world problems

LO2: Analyze the data using different statistical techniques and visualize the out- come using different types of plots.

LO3: Analyze and apply the supervised machine learning techniques like Classifi- cation, Regression or Support Vector Machine on data for building the models of data and solve the problems.

LO4: Apply the different unsupervised machine learning algorithms like Clustering or Association to solve the problems.

LO5: Design and Build an application that performs exploratory data analysis us- ing Apache Spark.

LO6: Design and develop a data science application that can have data acquisi- tion, processing, visualization and statistical analysis methods with supported machine learning technique to solve the real-world problem

Declaration

I declare that this written submission represents my ideas in my own words and where other's ideas or words have been included, I have adequately cited and ref- erenced the original source. I also declare that I have adhered to all principles of academic honesty and integrity and have not misrepresented or fabricated or falsified any idea/data/fact/source in my submission. I understand that any violation of the above will be cause for disciplinary action by the Institute and can also evoke penal action from the sources which have thus not been properly cited or from whom proper permission has not been taken when needed.

Mr. Nishant S Khetal (24) - - - - - - - - - - -

Mr. Atharv S Nikam (36) - - - - - - - - - - -

Mr. Pratik M Patil (40) - - - - - - - - - - -

(Signature)

Contents

List of Figures

List of Tables

Abstract

Airlines are constantly striving to enhance passenger satisfaction to retain customers and maintain competitiveness. This project aims to build a machine learning-based predictive system that can determine airline passenger satisfaction using various parameters such as seat comfort, inflight entertainment, food quality, flight delays, and customer service. The system performs data preprocessing, exploratory data analysis, and applies multiple classification algorithms to predict satisfaction. Among the models tested, **Random Forest** achieved the highest accuracy of **96.18%**. The insights derived from the data can help airlines make data-driven decisions to improve overall customer experience.

Keywords – Airline satisfaction, Machine Learning, Logistic Regression, Random Forest, KNN, Naive Bayes, Data Science, Prediction Model

Chapter 1
Introduction

1.1 Introduction

In the competitive aviation industry, customer satisfaction plays a critical role in influencing brand loyalty and revenue. Several variables affect a passenger's travel experience – including check-in service, inflight service, seat comfort, food, and punctuality. A reliable predictive model that accurately gauges satisfaction can help airlines identify pain points and take corrective actions, ultimately improving service quality and customer retention.

1.2 Literature Survey

The main objective of the research paper [1] is to develop a predictive model for airline passenger satisfaction using various machine learning algorithms. This paper is divided into four sections: (i) Data Collection from airline survey responses, (ii) Preprocessing and Feature Selection, (iii) Model Training and Evaluation using classifiers like Random Forest and Logistic Regression, and (iv) Performance Comparison of the models. The results showed that the Random Forest algorithm performed best due to its ability to handle non-linear relationships and feature importance analysis.

The paper [2] proposes a novel hybrid approach that combines Genetic Algorithms and AutoEncoders with machine learning models to improve the accuracy of passenger satisfaction prediction. It discusses the limitations of using individual models and highlights the advantage of ensemble and optimization-based techniques. The study includes a comprehensive analysis of real-world airline satisfaction datasets, emphasizing performance metrics like accuracy and recall. The proposed system showed significant improvement in classifying satisfied and dissatisfied passengers.

1.3 Problem Definition

Airlines often fail to identify the key drivers of customer dissatisfaction due to lack of structured analysis. This project aims to develop a machine learning model that classifies passengers as "satisfied" or "dissatisfied" based on historical feedback data. The goal is to extract actionable insights that can help airlines enhance service quality and reduce churn.

1.4 Objectives

Build a classification model to predict passenger satisfaction.
Perform thorough EDA to understand key satisfaction drivers.
Compare various ML algorithms and identify the best-performing model.
Provide insights into the most important features affecting satisfaction.

1.5 Proposed Solution

We approach this as a **binary classification problem** using supervised learning. The process includes:

1. Preprocessing and cleaning the dataset (handling missing values, encoding, normalization).
2. Performing EDA using plots and statistical summaries.
3. Applying classification algorithms: **Logistic Regression**, **KNN**, **Naive Bayes**, and **Random Forest**.
4. Choosing the model with the highest accuracy – **Random Forest** (96.18%).

1.6 Technology Used

Programming Language: Python
Libraries: pandas, numpy, seaborn, matplotlib, sklearn
Platform: Google Colab
Dataset: Airline Satisfaction Dataset (from Kaggle)

Chapter 2
Pre-processing

2.1 Dataset Description

The dataset contains feedback and demographic information of passengers, including:

- Gender
- Customer Type (Loyal / Disloyal)
- Type of Travel (Business / Personal)
- Inflight Services (Rating scale: 0–5)
- Satisfaction (Target variable: Satisfied / Dissatisfied)

Total Records: ~100,000
Target: satisfaction (Binary classification)

2.2 Handling Missing Data

Checked for null values using df.isnull().sum()
Minimal missing values were handled using:

- **Mode** for categorical features
- **Median** for continuous features

2.3 Handling Outliers

1. Outliers detected using **IQR method**
2. Outliers retained to preserve real-world variance
3. Feature distributions were visualized using boxplots

2.4 Feature Encoding & Scaling

- Categorical variables converted using LabelEncoder or get_dummies

- Scaling applied using StandardScaler for algorithms like KNN and Logistic Regression

Chapter 3
EDA and Visualization

3.1 Measures of Central Tendency

	Gender	Customer Type	Age	Type of Travel	Flight Distance	Inflight wifi service	Departure/Arrival time convenient	Ease of Online booking	Gate location	Food and drink	Online boarding	Seat comfort
count	25976.000000	25976.000000	2.597600e+04	25976.000000	2.597600e+04	25976.000000	25976.000000	25976.000000	25976.000000	25976.000000	25976.000000	25976.000000
mean	0.492917	0.184747	1.113300e-16	0.305590	8.931021e-17	2.724746	3.046812	2.756775	2.977094	3.215353	3.261665	3.449222
std	0.499959	0.388100	1.000019e+00	0.460666	1.000019e+00	1.335384	1.533371	1.412951	1.282133	1.331506	1.355536	1.320090
min	0.000000	0.000000	-2.155276e+00	0.000000	-1.164343e+00	0.000000	0.000000	0.000000	1.000000	0.000000	0.000000	1.000000
25%	0.000000	0.000000	-8.338705e-01	0.000000	-7.808310e-01	2.000000	2.000000	2.000000	2.000000	2.000000	2.000000	2.000000
50%	0.000000	0.000000	2.504343e-02	0.000000	-3.452494e-01	3.000000	3.000000	3.000000	3.000000	3.000000	4.000000	4.000000
75%	1.000000	0.000000	7.518167e-01	1.000000	5.509472e-01	4.000000	4.000000	4.000000	4.000000	4.000000	4.000000	5.000000
max	1.000000	1.000000	2.998207e+00	1.000000	3.794278e+00	5.000000	5.000000	5.000000	5.000000	5.000000	5.000000	5.000000

3.2 Univariate Analysis

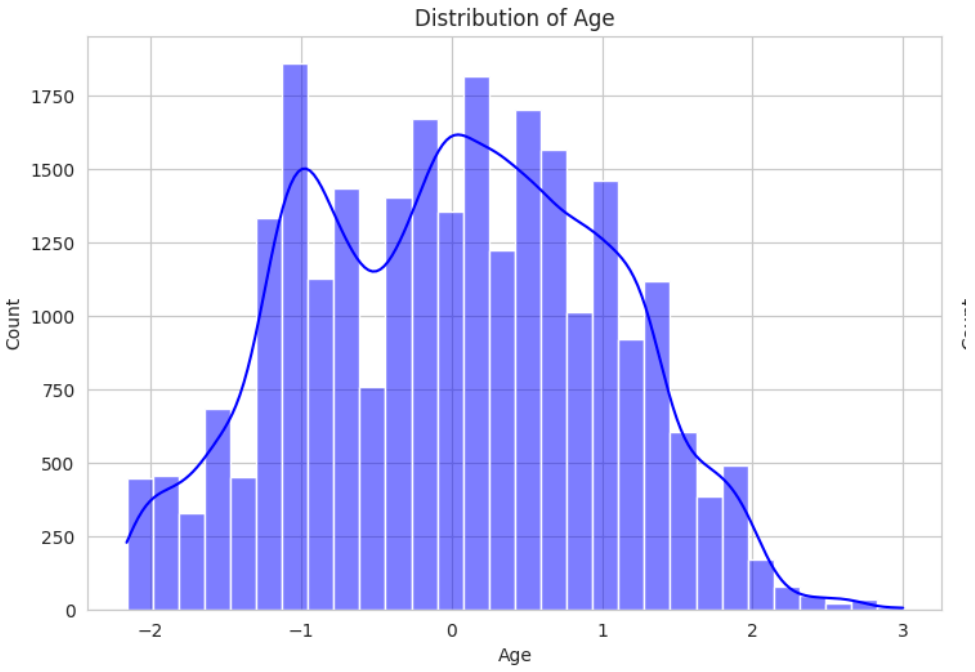


Fig. 3.1 Distribution plot of Flight Distance

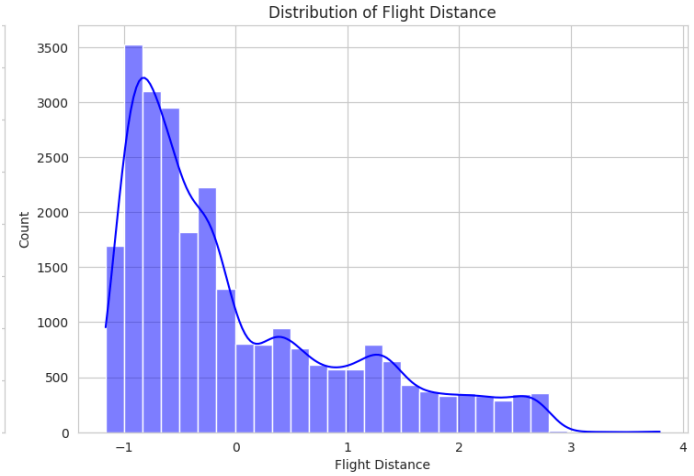


Fig. 3.2 Distribution plot of Age

This shows that the given data is right skewed for Flight Distance, Age

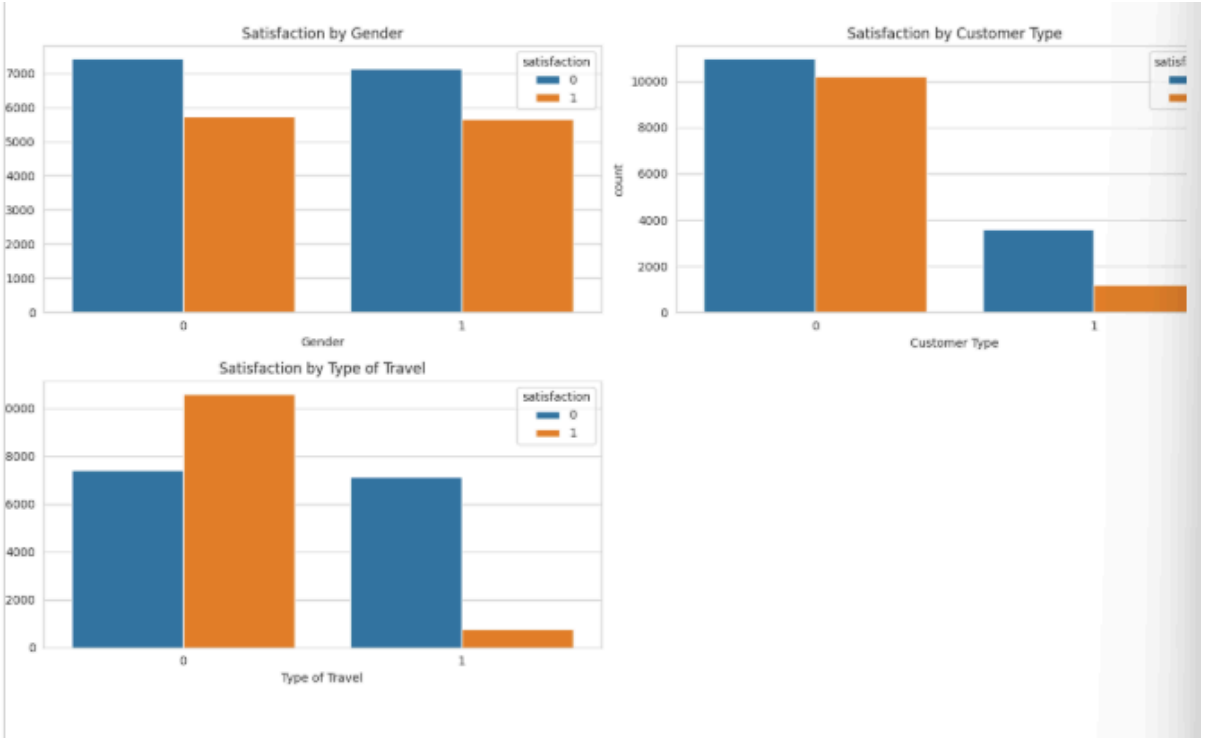


Fig. 3.3 Count plot of all attributes

These graphs show the categorical value distribution of the variables in the dataset

3.3 Distribution plot of Flight Distance

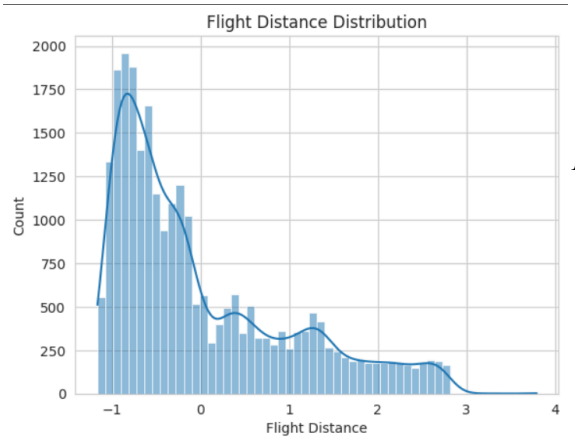
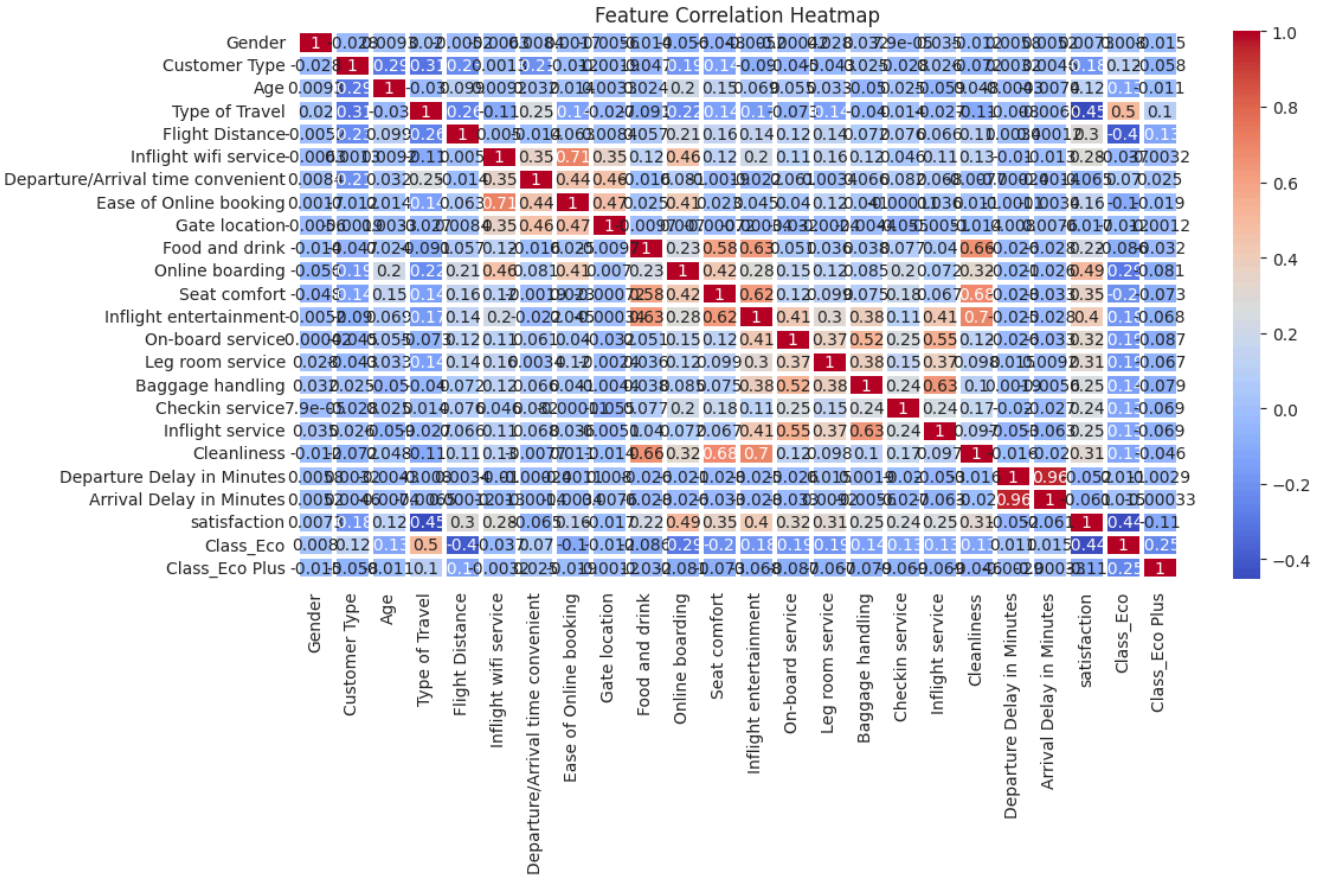


Fig. 3.4 Distribution plot of flight distance

3.4 Correlation Analysis

Correlation Heat Map:

Fig. 3.5 Heat Map showing correlation between all attributes



High positive correlation between:

- satisfaction and OnlineBoarding
- Satisfaction and Inflight Entertainment

Chapter 4

Data Modeling

The problem of predicting whether a passenger is satisfied or not is a classification problem. Therefore, we apply different classification algorithms and compare their performance to determine the best model for our dataset.

Logistic Regression

Logistic regression is a process of modeling the probability of a discrete outcome given an input variable. The most common logistic regression models a binary outcome; something that can take two values such as true/false, yes/no, and so on.

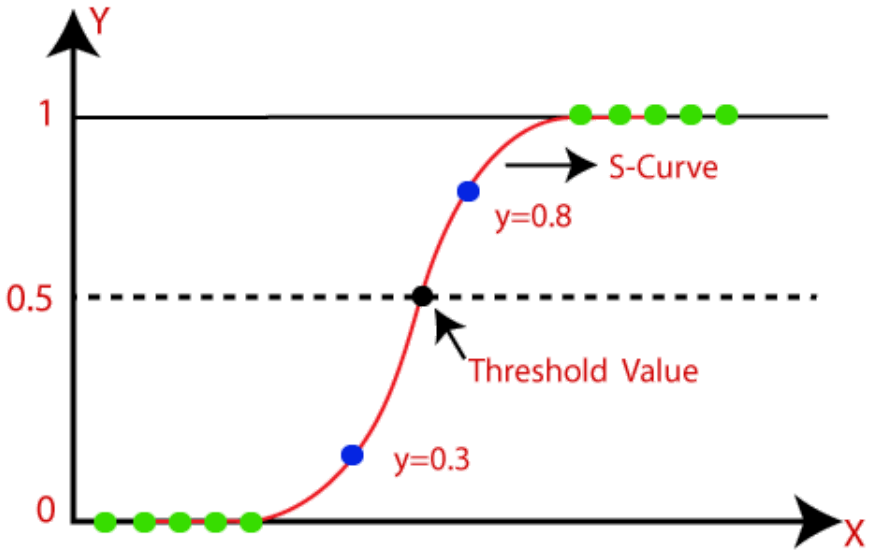


Fig. 4.1 Representation of logistic regression

Code:

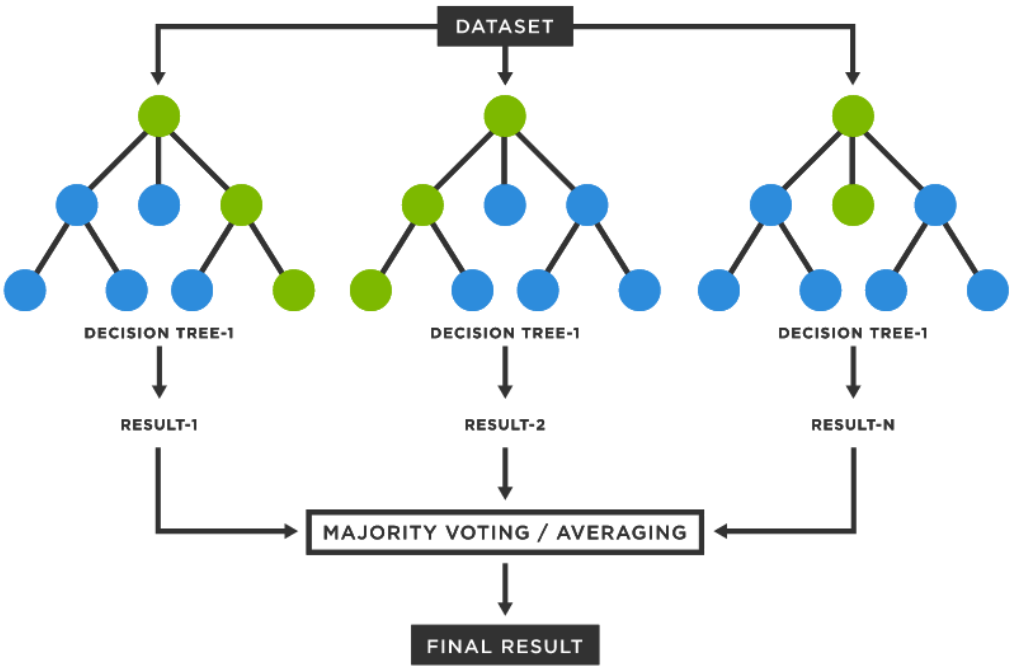
```
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score, classification_report, confusion_matrix
log_model = LogisticRegression(max_iter=1000)
log_model.fit(X_train, y_train)
y_pred = log_model.predict(X_val)
accuracy = accuracy_score(y_val, y_pred)
print(f"Logistic Regression Accuracy: {accuracy:.4f}")
print("Classification Report:")
print(classification_report(y_val, y_pred))
print("Confusion Matrix:")
print(confusion_matrix(y_val, y_pred))
```

Output:

```
87.64
```

Random Forest

Random forest is a Supervised Machine Learning Algorithm that is used widely in Classification and Regression problems. It builds decision trees on different samples and takes their majority vote for classification and average in case of regressio



n.

Fig. 4.2 Representation of Random Forest Classification

Code:

```
from sklearn.ensemble import RandomForestClassifier
rf_model = RandomForestClassifier(n_estimators=100, random_state=42)
rf_model.fit(X_train, y_train)
y_pred_rf = rf_model.predict(X_val)
accuracy_rf = accuracy_score(y_val, y_pred_rf)
print(f"Random Forest Accuracy: {accuracy_rf:.4f}")
print("Classification Report:")
print(classification_report(y_val, y_pred_rf))
print("Confusion Matrix:")
print(confusion_matrix(y_val, y_pred_rf))
```

Output:

96.18

K-Nearest Neighbors

K-Nearest Neighbour is one of the simplest Machine Learning algorithms based on Supervised Learning technique. It assumes the similarity between the new case/data and available cases and puts the new case into the category that is most similar to the available categories. K-NN algorithm stores all the available data and classifies a new data point based on the similarity. This means when new data appears then it can be easily classified into a well suited category by using K- NN algorithm.

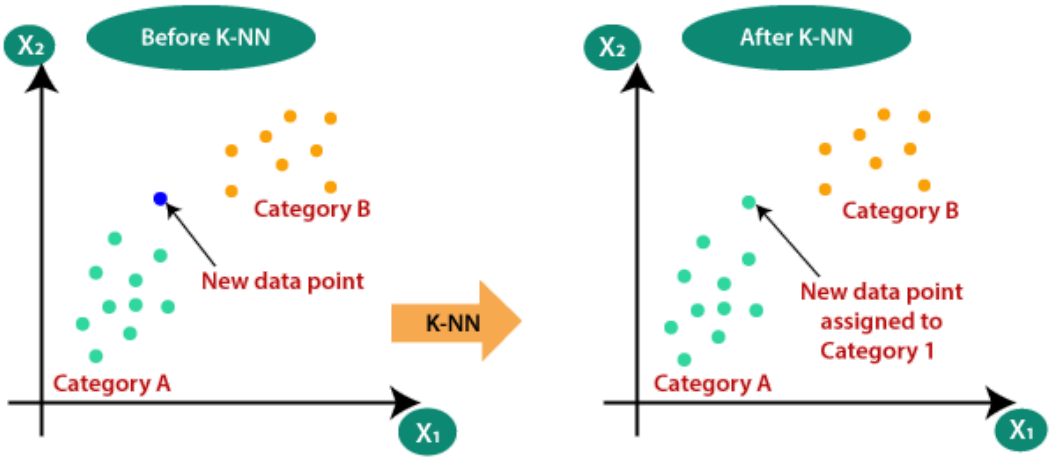


Fig. 4.3 Representation of KNN Classification

Code:

```
from sklearn.neighbors import KNeighborsClassifier
knn_model = KNeighborsClassifier(n_neighbors=5)
knn_model.fit(X_train, y_train)
y_pred_knn = knn_model.predict(X_val)
accuracy_knn = accuracy_score(y_val, y_pred_knn)
print(f"KNN Accuracy: {accuracy_knn:.4f}")
print("Classification Report:")
print(classification_report(y_val, y_pred_knn))
print("Confusion Matrix:")
print(confusion_matrix(y_val, y_pred_knn))
```

Output:

```
92.99
```

Gaussian Naive Bayes Classifier

Gaussian Naive Bayes is an extension of naive Bayes. Other functions can be used to estimate the distribution of the data, but the Gaussian (or Normal distribution) is the easiest to work with because you only need to estimate the mean and the standard deviation from your training data.

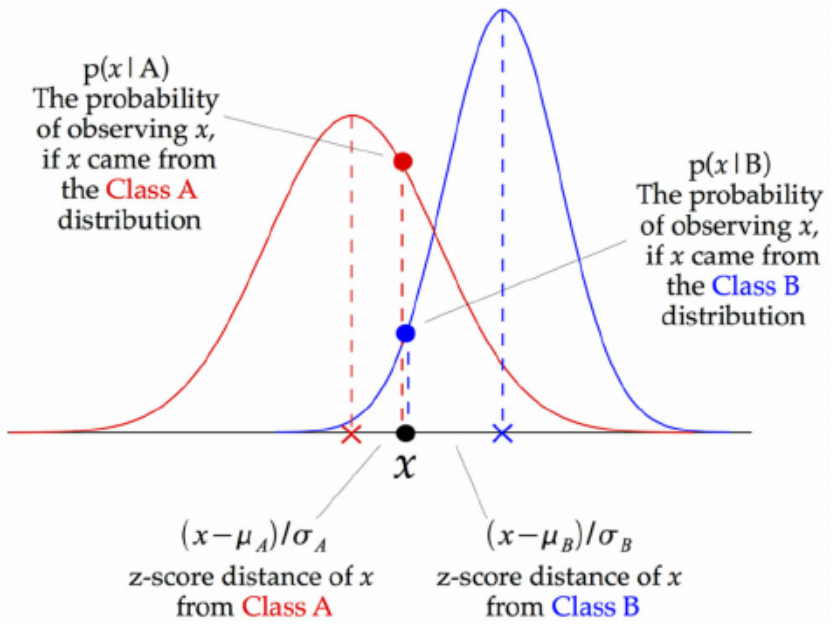


Fig. 4.4 Representation of Naive Bayesian Classification

Code:

```
from sklearn.naive_bayes import GaussianNB
nb_model = GaussianNB()
nb_model.fit(X_train, y_train)
y_pred_nb = nb_model.predict(X_val)
accuracy_nb = accuracy_score(y_val, y_pred_nb)
print(f"Naive Bayes Accuracy: {accuracy_nb:.4f}")
print("Classification Report:")
print(classification_report(y_val, y_pred_nb))
print("Confusion Matrix:")
print(confusion_matrix(y_val, y_pred_nb))
```

Output:

```
86.14
```

SVM Classifier

Support Vector Machine or SVM is one of the most popular Supervised Learning algorithms, which is used for Classification as well as Regression problems. However, primarily, it is used for Classification problems in Machine Learning.

The goal of the SVM algorithm is to create the best line or decision boundary that can segregate n-dimensional space into classes so that we can easily put the new data point in the correct category in the future. This best decision boundary is called a hyperplane.

SVM chooses the extreme points/vectors that help in creating the hyperplane. These extreme cases are called support vectors, and hence the algorithm is termed as Support Vector Machine. Consider the below diagram in which there are two different categories that are classified using a decision boundary or hyperplane:

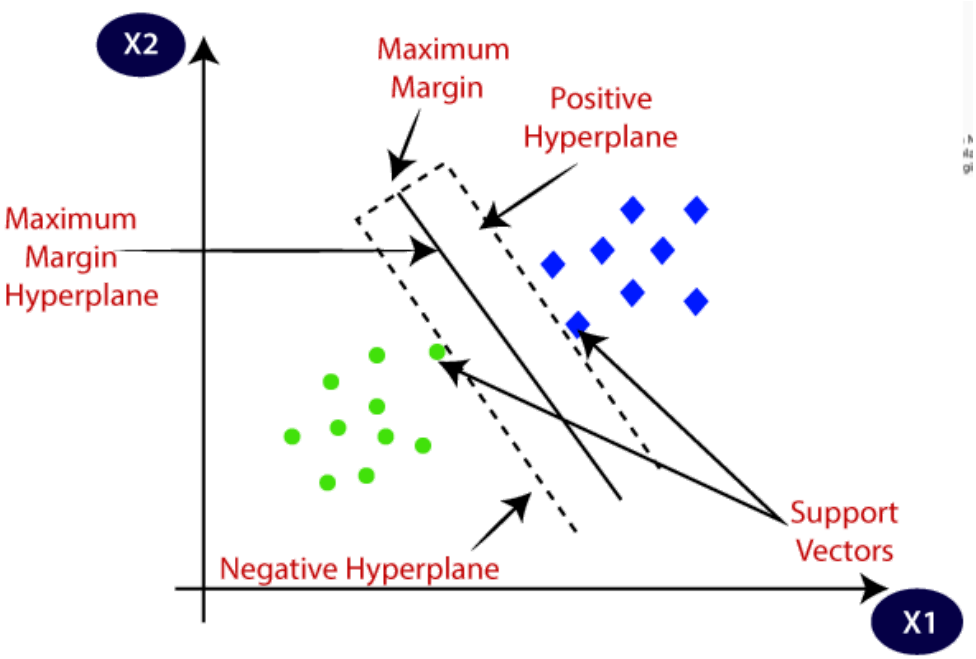


Fig. 4.5 Representation of SVM Classification

Code:

```
from sklearn.svm import SVC
svm_rbf_model = SVC(kernel="rbf", C=1.0, gamma="scale")
svm_rbf_model.fit(X_train, y_train)
y_pred_svm_rbf = svm_rbf_model.predict(X_val)
accuracy_svm_rbf = accuracy_score(y_val, y_pred_svm_rbf)
print(f"SVM (RBF Kernel) Accuracy: {accuracy_svm_rbf:.4f}")
print("Classification Report:")
print(classification_report(y_val, y_pred_svm_rbf))
print("Confusion Matrix:")
print(confusion_matrix(y_val, y_pred_svm_rbf))
```

Output:

```
93.69
```

Gradient Boosting Classifier

Gradient boosting classifiers are a group of machine learning algorithms that combine many weak learning models together to create a strong predictive model. Decision trees are usually used when doing gradient boosting. Gradient boosting models are becoming popular because of their effectiveness at classifying complex datasets

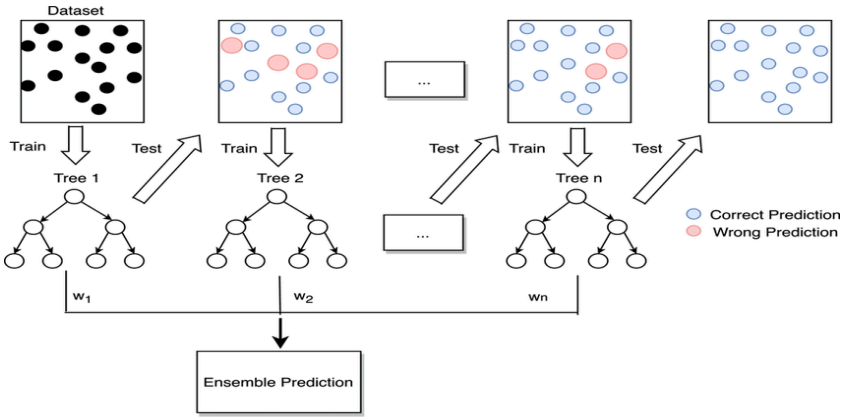


Fig. 4.6 Representation of Gradient Boosting Classification

Code:

```
from sklearn.ensemble import GradientBoostingClassifier
gb_model = GradientBoostingClassifier(n_estimators=100,
learning_rate=0.1, random_state=42)
gb_model.fit(X_train, y_train)
y_pred_gb = gb_model.predict(X_val)
accuracy_gb = accuracy_score(y_val, y_pred_gb)
print(f"Gradient Boosting Accuracy: {accuracy_gb:.4f}")
print("Classification Report:")
print(classification_report(y_val, y_pred_gb))
print("Confusion Matrix:")
print(confusion_matrix(y_val, y_pred_gb))
```

Output:

```
94.13
```

Summary of Data modeling

Model	**Score**
Logistic Regression	87.18
Gaussian Naive Bayes Classifier	85.93
Gradient Boosting Classifier	94.13
Random Forest	95.48
SVC	93.69
K- Nearest Neighbor	91.42

Table 4.1 Summary of Data modeling

The Highest Accuracy among Classifiers is shown by Random Forest => **95.48%**

Chapter 5
Hypothesis Testing

Statistical Hypothesis Testing: Evaluating the Performance Difference Between Random Forest and Logistic Regression

In machine learning, models are typically selected based on their mean performance. However, to ensure the difference in mean performance is statistically significant and not due to chance, we perform a **paired t-test**. Specifically, we use **5×2-fold cross-validation**, which is implemented in the **MLxtend** library by Sebastian Raschka through the paired_ttest_5x2cv() function.

Hypothesis Setup:

- **Null Hypothesis (H0)**: There is no significant difference between the performance of Random Forest and Logistic Regression.
- **Alternate Hypothesis (H1)**: There is a significant difference between the performance of Random Forest and Logistic Regression.

Mathematically:

- H0: d=0
- H1: d!=0

```
from mlxtend.evaluate import paired_ttest_5x2cv
from sklearn.ensemble import RandomForestClassifier
from sklearn.linear_model import LogisticRegression
import numpy as np
t, p = paired_ttest_5x2cv(
    estimator1=RandomForestClassifier(n_estimators=100, random_state=42),
    estimator2=LogisticRegression(max_iter=1000),
    X=X_train,
    y=y_train,
    scoring="accuracy",
    random_seed=42
)
print(f"P-value: {p:.3f}, t-Statistic: {t:.3f}")
if p <= 0.05:
    print("There is a significant difference between Random Forest and Logistic Regression.")
else:
        print("There is NO significant difference between Random Forest and Logistic Regression.")
```

- **P-value**: 0.000 (less than 0.05)
- **t-Statistic**: 18.03

Since the **p-value is less than 0.05**, we reject the null hypothesis (H0). This means that there is a **significant difference** between the performance of Random Forest and Logistic Regression.

There is a **significant difference** between the performance of **Random Forest** and **Logistic Regression**. Based on the results, we can confidently choose the model with the better performance for further classification tasks.

Chapter 6
Result and Analysis

Performance metrics for the model using Random Forest:

Accuracy : 95.48
Confusion Matrix:

```
[[ 2816   99]
 [136 2145]]
```

Classification Report:

```
Classification Report:
              precision    recall  f1-score   support

           0       0.95      0.97      0.96      2915
           1       0.96      0.94      0.95      2281

    accuracy                           0.95      5196
   macro avg       0.95      0.95      0.95      5196
weighted avg       0.95      0.95      0.95      5196
```

Features important for the classification:

```
importances = pd.DataFrame({'Features': X_train.columns, 'Importance':
np.round(rf_model.feature_importances_, 3)})
importances = importances.sort_values('Importance',
ascending=False).set_index('Features')
importances.plot.bar()
```

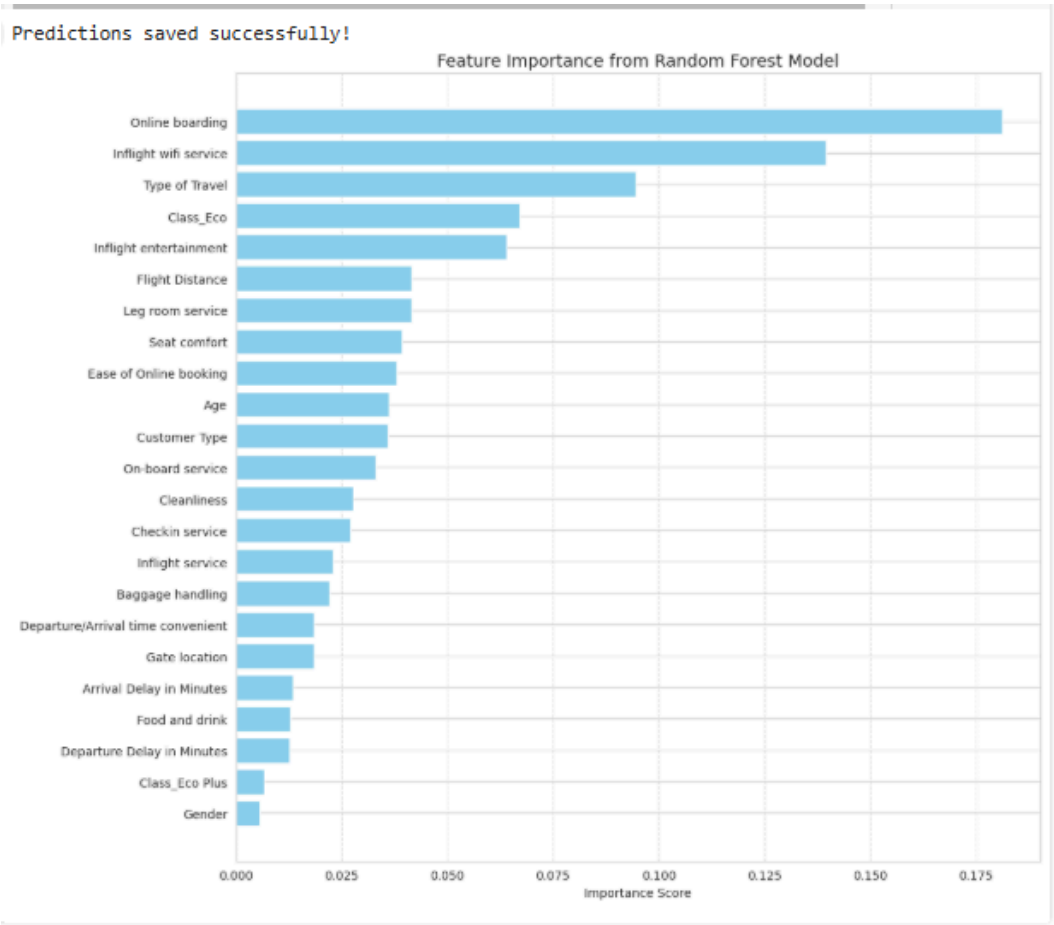


Fig. 6.1 Bar chart showing precedence of importance of features

Online boarding and *Flight Distance* have the maximum importance, while *Gender* and *Gate location* have the least.

Inference:

Passenger satisfaction is most closely related to features such as **Online boarding**, **Flight Distance**, **Class**, **Inflight WiFi service**, and **Inflight entertainment**. These are factors that significantly impact the customer experience during air travel.

The Random Forest model identifies these features as key contributors based on historical data, helping in making accurate predictions about passenger satisfaction for future or unseen instances.

This data-driven analysis helps understand the influence of different flight experience factors on satisfaction, allowing airlines to improve services that matter most to passengers.

Prediction over unseen dataset using Logistic Regression for final submission:

```
X_test = test_df.drop(columns=["satisfaction"], errors='ignore')
test_predictions = rf_model.predict(X_test)
submission = pd.DataFrame({
    "id": test_df.index,
    "satisfaction": test_predictions
})
submission.to_csv("final_predictions.csv", index=False)
```

Chapter 7

Future Scope

The field of predictive analytics for airline passenger satisfaction continues to evolve rapidly with the advancement of artificial intelligence and big data technologies. The current project has demonstrated a robust foundation by achieving a high classification accuracy using the Random Forest model. However, there is vast potential for enhancing the model and expanding its practical implementation.

1. **Integration with Real-Time Feedback**
   Future iterations can focus on integrating real-time feedback during flights through in-flight entertainment systems, mobile apps, or wearable devices. This would enable airlines to monitor satisfaction dynamically and take immediate corrective actions, thereby enhancing customer experience during the journey itself.

2. **Adoption of Deep Learning Models**
   With access to larger datasets, deep learning models such as CNNs (for image data like boarding pass scans) or LSTM/RNNs (for sequential data like real-time logs) can be implemented. These models are capable of capturing complex non-linear patterns and temporal dependencies more effectively than traditional classifiers.

3. **Multimodal Sentiment Analysis**
   In addition to structured feedback, unstructured data like social media posts, online reviews, and call center transcripts can be analyzed using Natural Language Processing (NLP). Sentiment analysis from these sources can enrich the prediction model and offer a 360-degree view of passenger satisfaction.

4. **Cross-Airline Dataset Expansion**
   The current model is trained on data from a single airline. Generalizing the model by incorporating data from multiple domestic and international airlines can significantly improve the reliability and adaptability of the system.

5. **Deployment as a Full-Stack Application**
   A production-grade web application can be developed for airlines. This would include dashboards, satisfaction heatmaps, and predictive alerts for service failure, empowering decision-makers with actionable insights in real time.

Chapter 8

Societal Impact

The deployment of machine learning models to predict airline passenger satisfaction not only advances business strategies but also holds significant societal implications. The ripple effects of such technologies can be seen across various dimensions — from enhancing customer experiences to optimizing airline operations and promoting sustainable development.

1. **Improved Passenger Experience**
   By proactively identifying dissatisfaction drivers such as delayed flights, poor service, or uncomfortable seating, airlines can improve the overall quality of travel. This leads to happier passengers, fewer complaints, and a better travel culture — which is especially critical in post-pandemic recovery phases.

2. **Operational Efficiency**
   Predictive analytics can help airlines allocate resources better — such as adjusting staff levels, improving food quality, or upgrading frequently complained seats. This reduces operational costs while ensuring that services are aligned with what passengers value most.

3. **Data-Driven Decision Making in Aviation**
   Empowering airline management with actionable insights brings a cultural shift from intuition-based to evidence-based decisions. This enhances transparency and trust in airline operations.

4. **Boost to Travel and Tourism Industry**
   Satisfied passengers are more likely to travel again, leave positive reviews, and recommend airlines. This indirectly supports the travel and tourism sector, which contributes significantly to national economies and employment.

5. **Digital Transformation and Skill Development**
   Projects like this one showcase the importance of digital tools and machine learning, encouraging students, professionals, and companies to invest in upskilling. This fosters a future-ready workforce and bridges the gap between academia and industry.

6. **Inclusivity and Accessibility**
   Advanced analytics can uncover patterns that reflect the needs of specific demographics, such as senior citizens or passengers with disabilities. Airlines can then tailor services that foster inclusivity and equality in air travel.

Chapter 9

Implementation

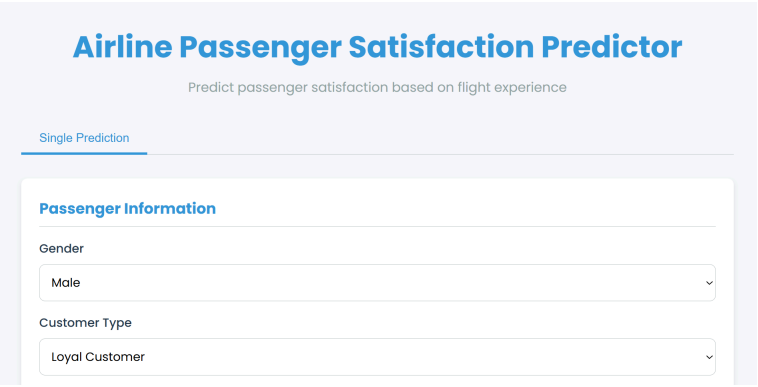


Fig. 9.1 Airline passenger satisfaction predictor

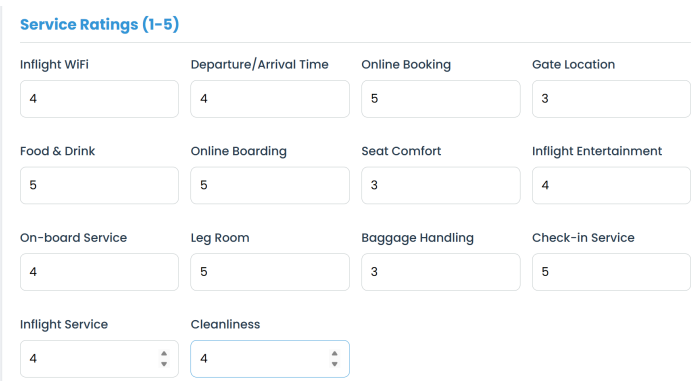


Fig. 9.2 Service based rating in 1-5

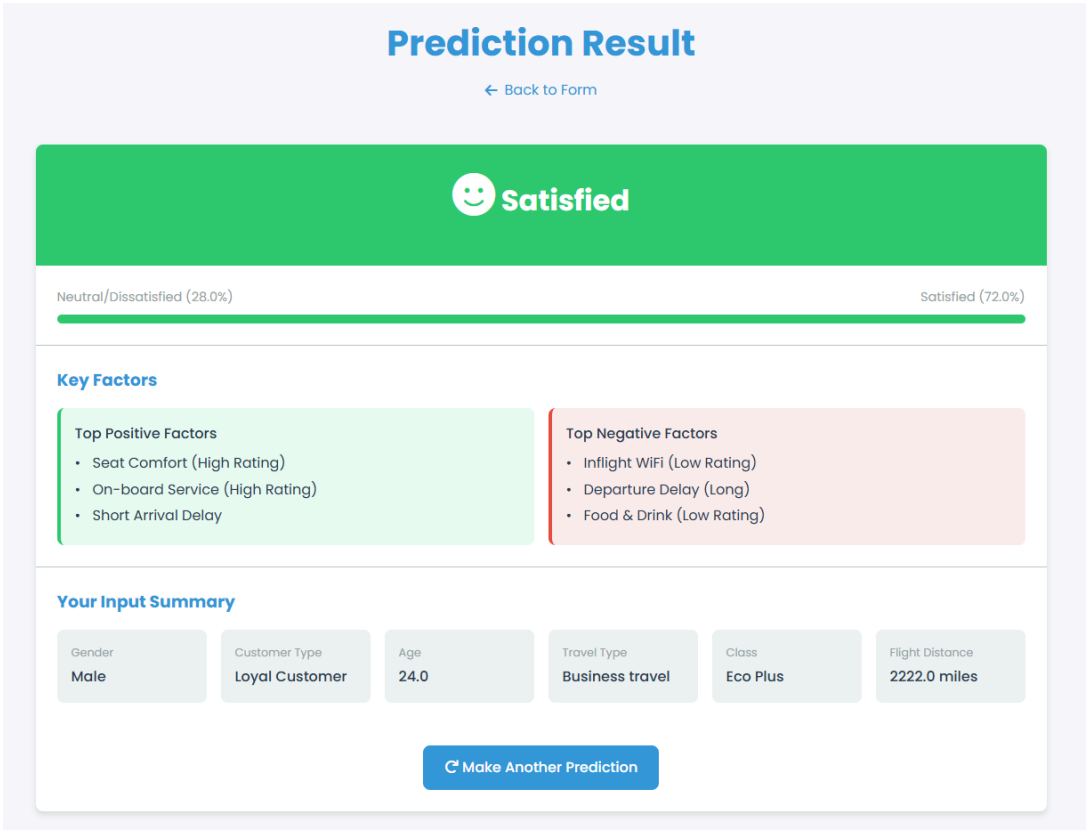


Fig. 9.3 Airline passenger satisfaction predictor Result

Chapter 10
CONCLUSION

In this project, we developed a machine learning classification model aimed at predicting passenger satisfaction based on a variety of features related to their in-flight experience. After exploring and comparing multiple machine learning algorithms, the Random Forest Classifier emerged as the most effective model, achieving a commendable accuracy of **95.48%**. This high level of accuracy underscores the model's robustness and its ability to effectively learn and generalize from complex patterns within the dataset.

The Random Forest algorithm proved particularly suitable for this task due to its ensemble learning approach, which combines the predictions of multiple decision trees to improve performance and reduce overfitting. By leveraging this algorithm, the model was able to accurately distinguish between satisfied and dissatisfied passengers using a diverse set of features.

Key factors influencing passenger satisfaction were identified through feature importance analysis. Among these, **inflight entertainment**, **seat comfort**, and **onboard services** stood out as the most impactful predictors. These insights provide valuable information for airline companies looking to enhance customer satisfaction by focusing on the aspects of service that matter most to their passengers.

Overall, the project successfully demonstrated how machine learning can be applied to real-world scenarios in the airline industry. By building a highly accurate predictive model, we showcased the potential of data-driven decision-making in improving customer experience and operational efficiency.

References

[1] Ashwika, Dishali G. K., Hemalatha Nambisan. "Airline Passenger Satisfaction Prediction Using Machine Learning Algorithms." *Redshine Archive*, Vol. X, Issue Y, pp., 2020.

[2] Lee Ye Hean, Olanrewaju Victor Johnson, Chew XinYing, Teoh Wei Lin, Chong Zhi Lin, Khaw Khai Wah. "An Airline Passenger Satisfaction Prediction by Genetic-Algorithm-Based Hybrid AutoEncoder and Machine Learning Models." *International Journal of Intelligent Systems and Applications in Engineering (IJISAE)*, 2024.

[3] Forecasting Airline Passengers' Satisfaction Based on Sentiments, 2024.

[4] Drivers and Outcomes of Airline Passenger Satisfaction: A Meta-Analysis, 2024.

Atharv Sanjay Nikam
Roll No: 36
TE(INFT) SEM VI

DISC
36

05/05

AIDS
Assignment - 1

1) What is AI? Considering the COVID-19 pandemic situtation how AI helped to survive and renovate our way of life with different application)

Ans Artifical Intelligence is the simulation of human intelligence in machine that can perform tasks typically requiring human cognition, such as learning, problem-solving, decision-making and understanding natural language. AI systems use algorithms, neural networks and deep learning technique to analyze data, recognize patterns, and make autonomous decisions.

How AI helped during Covid-19.

① Healthcare - AI diagnosed COVID-19, assisted in drug discovery, and powered chatbots for medical guidance
② Contact Tracing - AI-powered apps (Aarogya Setu) tracked exposure and ensured social distancing.
③ Remote Work and Education - AI enhanced tools like Zoom and google meet enabled virtual enabled work and learning.
④ Robotics and Automation - AI robots disinfected hospitals and automated supply chains.
⑤ Misinformation control - AI fact-checked and remove fake COVID-19 news.
⑥ Entertainment and Mental health - AI driven platform kept people engaged and supported mental health.

2) What are AI Agent terminology, explain with examples?

Ans The AI agents terminology includes:

(i) Performance Measure of Agent: It determines the success of the Agent.

(ii) Behaviour / action of Agent: It is the action performed by an agent after any specified sequence of precept.

(iii) Percept: Agents perpetual inputs at a specified instance.

(iv) Percept sequence: History of everything agent has percieved till date

(v) Agent Function: Map from percept Sequence to an Action.

Agent function, $a = F(P)$

where P is current percept, a is the action carried out and F is the agent Function

F maps percept to action.

$F : P \rightarrow A$

where P is the set of all percept and A is set of all actions Action may be depent on all the percept observed not only the current percept

$a_R = F(p_0, p_1, p_2 \ldots p_R)$

where $p_0, p_1, p_2, \ldots p_R$ is the sequence of

of percepts recorded till date, ab is the resulting action carried out and F now maps percept sequence to action.

$$F: P^* \rightarrow A$$

For example the vacuum cleaner problem:

Percept sequence	Action
[A, clean]	Right
[A, Dirty]	Suck
[B, clean]	Left
[B, Dirty]	Suck
[A, Dirty], [A, clean]	Right
[A, clean], [B, Dirty]	Suck
[B, Dirty] [B, clean]	left
[B, clean] [A, clean]	No operation.
[A, clean] [A, clean]	No-operation

Performance measure of vacuum cleaner agent:
All rooms are cleaned.

Behaviour/action of agent: left, Right, Suck and no-operation (No-nothing).

Percept: Location and status, for example[A, Dirty]

Agent functioning: Mapping of percept sequence to an action.

(3) How AI technique is used to solve 8 puzzel problem

Ans The 8 puzzle problem is a state space search problem in AI where a 3×3 grid contains 8 tiles numbered from 1 to 8 and 1 empty space. Objective is to rearrange titles to reach a predefined goal state.

AI techniques:

(1) Uniformed search methods:
BFS: Expand the shallow node first
DFS: Expand the deepest node as possible before back tra...
IDS: Combine DFS and BFS to increase dept limit gradually.

(2) Informed search methods.
BFS: Best first search based on heuristic function that appears closet to the goal.
- Search. $f(n) = g(n) + h(n)$ Based on heuristic cost to node

Initial State :

1	2	3
4		6
7	8	5

Goal state :

1	2	3
4	5	6
7	8	

(i) Compute heuristic of each possible move
(ii) Expand the state with the lowest f(n) and repeat

4) What is PEAS description? give PEAS for following!

→ ① Performance Measure : How Sucess of agent in evaluated ② Environment : Surrounding in which agent operates
③ Actuators : Component that allows agent to take actions
④ Sensors : Component that allows agent to perceive the enviornment

(i) Taxi driver agent :

Performance Measure	Environment	Actuators	Sensors
- Safe driving	- Traffic signal	- Steering wheel	- Camera
- Travel time	- roads	- accelelator	- GPS
- traffic rule	- Weather	- brakes	- Fuel gauge

(ii) Medical Diagnostic Agents :

Performance Measure	Enviornment	Actuators	Sensors
- health of patient	- patient	- display screen	- heart rate monitor
- accuracy of diagons	- symptoms	- alarm system	- lab result
- recommended treatement	- test reports		

(iii) Music component Agent:

Performance measure	Environment	Actuator	Sensor
- Originality - quality - listener	- Music db - user preference	- Speaker - mixer	- microphone - user feedback

(iv) An aircraft autolander

Perf measure	Environment	Actuator	Sensors
Smooth landing - accuracy in touchdown	- Runway - wind condition - Air traffic	- landing gear - flap - air brakes	- GPS - Camera - Altmeter

(v) Essay evaluator

Performance measure	Environment	Actuator	Sensors
- grading - grammer - paradigm check	- plagiarism - rubric criteria	- display screen - text to speaker	- Optimal character Recognition (OCR)

vi) Robotic sentry gun for the Rock Lab

performance measure	Environment	Actuators	Sensor
- neutralize threats - target f. - false alarms	- lab area - potential intruders	- gun mechanism - alarm.	- Camera - thermal sensor.

(5) Categorize a shopping bot for a shopping bot for an offline bookstore according to the following system

Ans Observability : Partially observable Relies on limited sensor input.

- Deterministic or stochastic - Stochastic customer prefrence is unpredictable.

- Episodic vis sequential : Sequential affects future actions

- Static vis dynamic - Dynamic, customer behaviour is always evolving.

- Discrete vis continuous : ~~finite~~ Discrete - The bot work with finite set of actions.

- Single vs Multi Agent : Multi-Agent
The interacts with multiple entites, including customers, bookstore staff.

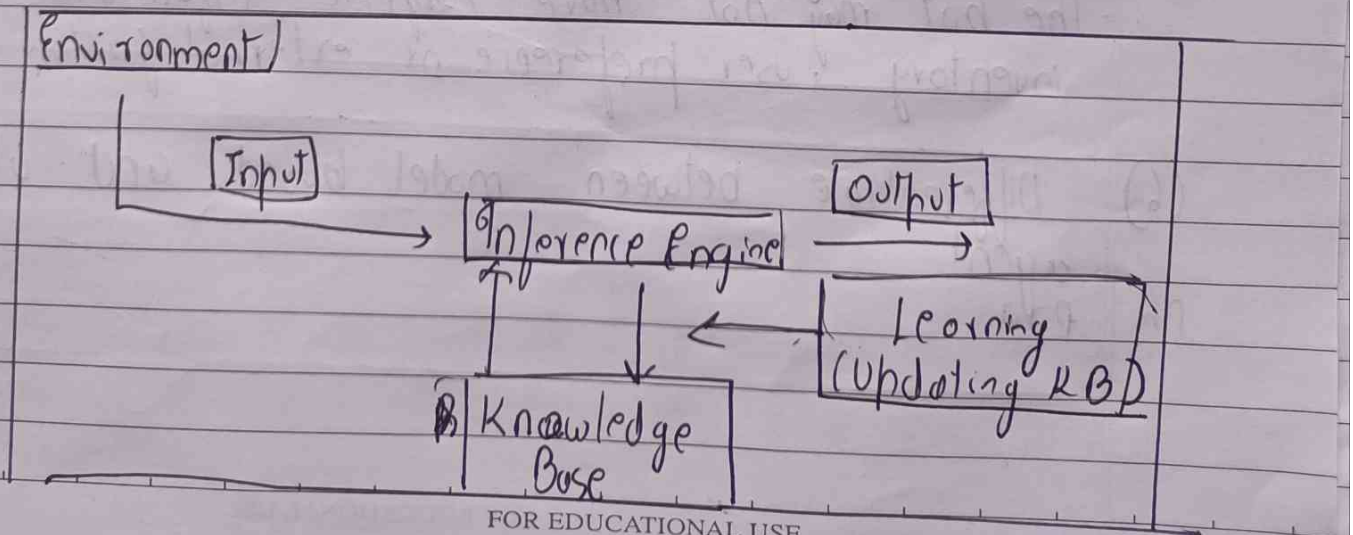
- Observability : Partially Observable.
- The bot may not have complete information about inventory, user preference or external factors like stock upd

(6) Differentiate between model based and utility based agent

Ans pto

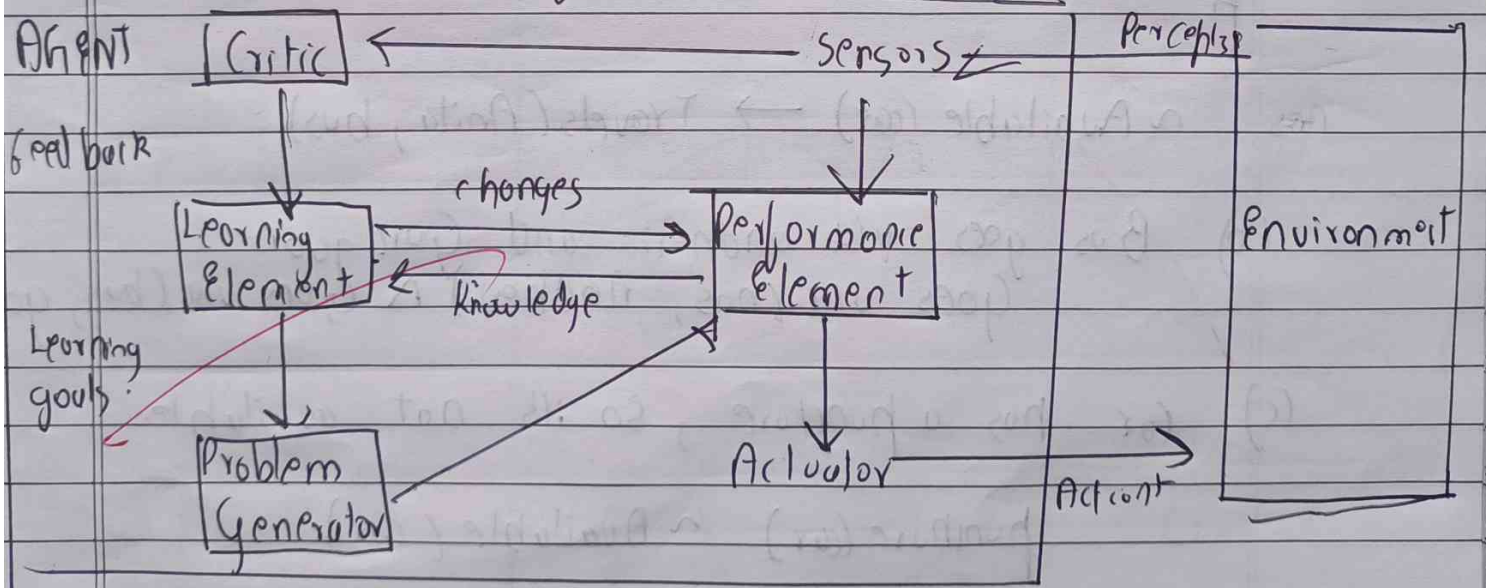
Model Based Agent	Utility Based Agent
Agent that maintains the internal model of the env to understand its current state and predicate future states	- Agents that selects actions based on the utility function aiming to maximize long term satisfaction or benefit.
Model updates its information about the environment	- Measures how desirable different states are
- less complex	- More complex
- Doesn't concern long term rewards	- Focuses on long term reword.
- Eg- self driving cars	- eg- Shopping recomendation system.

Explain the architecture of knowledge based agent and learning agent



Knowledge based agent - Stores knowledge and reason based on logical inference.
- Knowledge base - Stores fact, rules and heuristic function about the environment
- inference engine - uses logical reasoning techniques like forward and backward chaining
- perception - gathers data from the environment
- actions - executes actions based on a infered knowle
- Knowledge update mechanism - Update itself as new facts are learned.

learning based agent : Agent that improves its performance over time by learning from experience, data



- learning element responsible for improving agents perby analyzing past experience using ML technique
- Critic - provide feedback on agents actions by

evaluating success or failure.
problem generation: Supports new experience for learning and explanation

(8) Same as question 1

(9) Convert the following to predicate:

(a) Anita travels by car if available otherwise travels by bus.

Ans travels (x, y) → person x travels by y.
Available (y) → y (a vehicle) is available.
Goes via (y, z) → vehicle goes via z

P

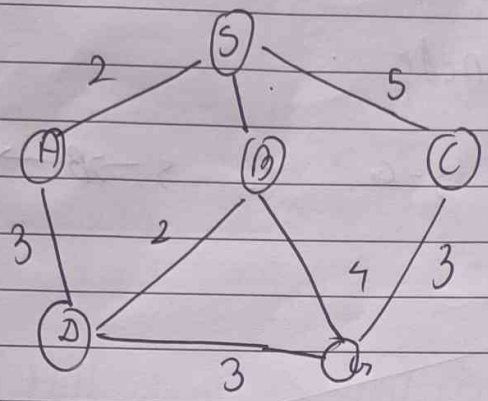
Ans ~ Available (car) → Travels (Anita, bus).

(b) Bus goes via andheri and Goregaon
Goes via (bus, Andheri) ∧ Goes via (bus, goregaon)

(c) Car has a puncture, so it's not available.

puncture (car) ~ Available (car).

(v)

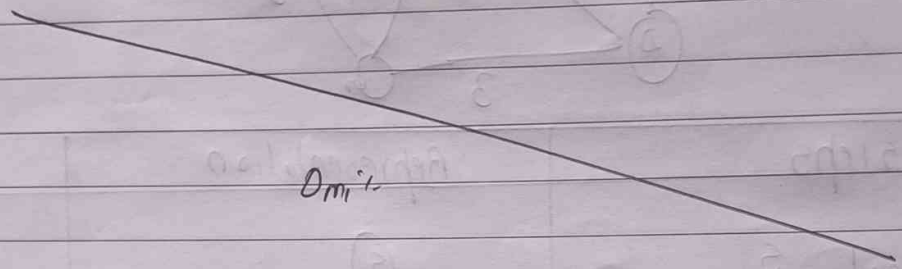


Ans

steps	Representation	stack
(i) Load S	S	[S]
(ii) Pops, Load A, B, C	S, A, B, C	[A, B, C]
(iii) Pop A Expand D	S, A, B, C, D	[B, C, D]
(iv) Pop B, Expand G	S, A, B, C, D, G	[C, D, G]

G is the goal node.

∴ Route from S-G is S → B → G.



(11) What do you mean by depth limited search? Explain Iterative Deepning Search with example

Ans Depth Limited Search (DLS)

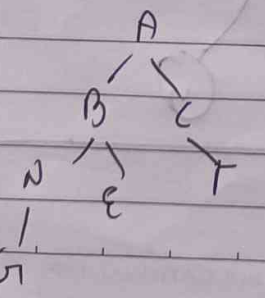
DLS is a variation of DEPTH-FIRST-SEARCH (DFS) that limits the depth of exploration to a predefined level L to prevent infinite loops in deep or infinite search space.

Pros: Prevents infinite recursion.

Cons: May miss the goal if ilies beyond depth L

IDS combines DFS and BFS by repeadtedly applying DLS with increasing depth limits until goal is found.

Eg



To find G

IDS runs DLS multiple time with increasing de

DLS with $L=0$ → A (goal not found)
DLS with $L=1$ → ABC (goal not found)
DLS with $L=2$ → A, B, C, D, E, F (goal not found
DLS with $L=3$ → A, B, C, D, E, F, G (goal found)

Explain Hill climbing and its drawback in det with example also state limitation of stee -ascent hill climbing

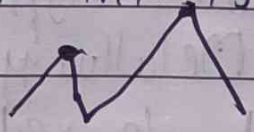
Hill climbing Algorithm

Hill climbing is a heuristic search algorithm use for optimization problems. It continuously mov toward the best neighbouring state until no be move exists.

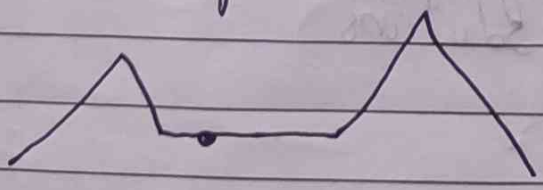
Example: A blindfolded climber moves uphill unti no higher step is possible, but may get stuck a local peak instead of global peak.

Drawbacks of Hill Climbing

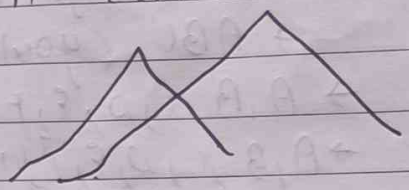
Local maxima - Stops at a peak that is not the highest.



Plateau - Flat region cause the algorith to stop



iii) Ridges :- needs to move down first before climbing higher which it cannot do



Steepest Ascent Hill climbing and Limitations

a) In this it evaluates all neighbours and picks the best move.

Limitation :- (1) Computationally expensive in large space.

(2) Still gets stuck in local maxima, plateau and ridges

(3) Slow convergence due to evaluating all paths.

(13) Explain simulated annealing and write its algorithm.

Ans Simulated Annealing (SA) is a heuristic optimization algorithm inspired by the annealing process in Metallurgy, where metals are heated and slowly cooled to reduce defects. It is used to avoid local optima by allowing controlled random jumps to explore better solutions

Algo :

1) Initilize:
- Set initial state S and temp T.

2) Repeat until stopping condition (T > 0)
- Generate an neighbouring state S'.
- Compute the energy difference $\Delta E = f(S') - f(S)$
- if S is better ($\Delta E > 0$) accept it.
- if S is worse ($\Delta E < 0$) accept it with probability.
$$P = e^{(\Delta E / T)}$$
- Reduce T

3) Retur best solution found.

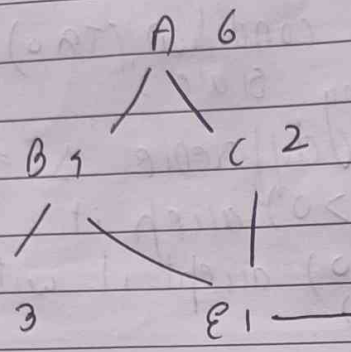
13) Explain A* Algorithm with an example.

A* is a algorithm is used in AI for path finding and graph traversal it combines
- g(n) - cost from the start node to the current node.
- h(n) - Heuristics (estimated cost from the current node to goal)
- f(n) = g(n) + h(n) → Total estimated cost.

steps of A* Algo.

i) Intilize ope list (nodes to explore) and closed list (explored nodes)
ii) Start from the initial node, calculate f(n) = g(n) + h(n)
iii) Pick the node with the lowest f(n) and expand it
iv) Add neighbor to the open list, update the g(n), h(n), f(n)

Repeate until goal node is reached.



start at A, compute $f(n) = g(A) + h(A) = 0 + 6 = 6$

Expand A, B, C

Pick C lowest (h) expand to G.

goal reached with shortest path as A → C → G

15) Explain Min max and algorithm and draw game tree for TIC TAC TOE Game

Ans Minimax is a decision making algorithm used in game playing (Tic-Tac-Toe, chess). It helps find the optimal move by assuming:

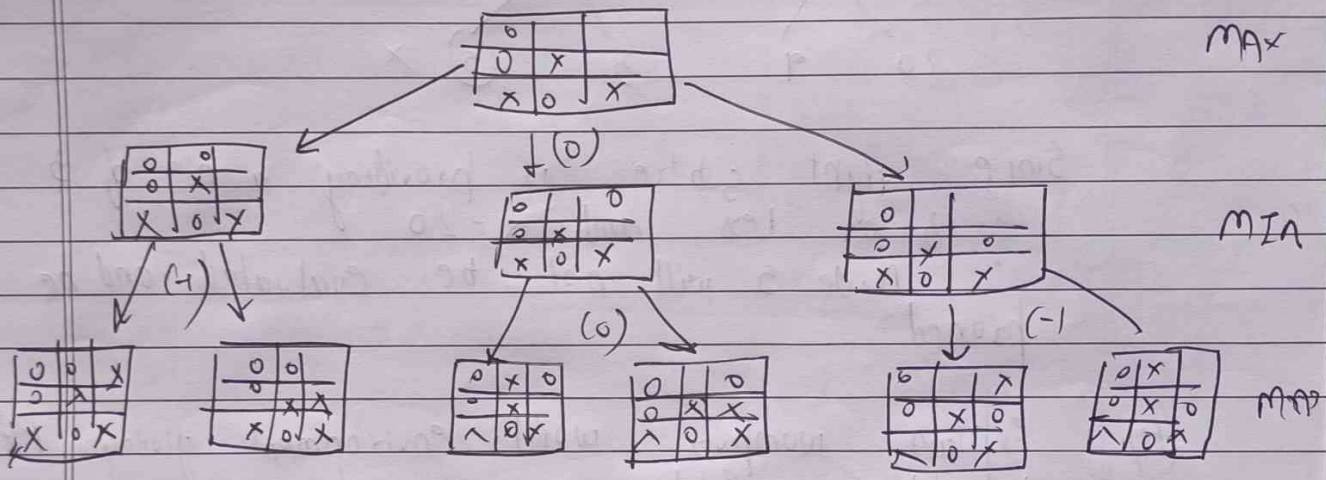
- The AI (max players) tries to maximize to its score.
- The opponent (MIN player) tries to minimize the AI.

Min Max Algo.

(i) Generate the game tree up to a certain depth

(2) Evaluate terminal nodes using a heuristic function.
(3) Propagate values upward:
- Max player picks the highest value
- Min player picks the ~~highest~~ lowest value.
(4) Choose the best move at the root node

TIC-TAC-TOE - Game Tree.



16) Explain Alpha beta pruning algorithms for adversarial search with example

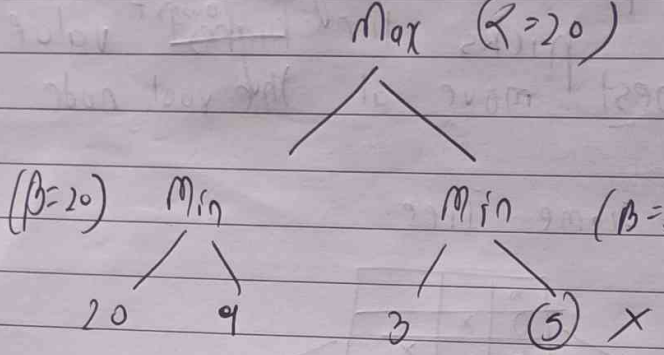
Ans It's an optimization technique for minmax algorithm. It reduces no of nodes evaluated in game tree by eleminating unnecessary branches

Algorithm:- Set two values α and β for MAX and Min respectively.

If $\alpha \geq \beta$ for any subtree or node, stop eval and prune it.

All nodes which do not influence the final decision.

Ex



Since right subtree is providing a min of 3
∴ 3 or less and $\alpha = 20$
∴ Node 5 will not be evaluated and be pruned.

(11) Explain wumpus world environment giving PEAS description. Explain how percept sequence is generated.

Ans The wumpus is a grid based environment used to demo AI agents, logic based reasoning and desision making.

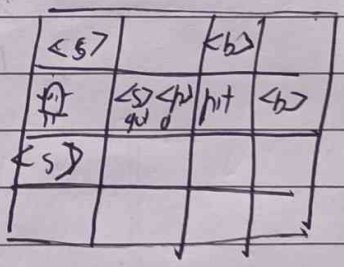
Environment components.
- 4×4 grid of rooms
- Rooms contain pits or the wumpus
- One room contains (goal)

- Agent starts at (1,1) and can move horizontally and vertically
- One arrow to kill the wumpus.
- Rooms surrounding the wumpus have stench and rooms surrounding the pit have breeze

Pos

Perf measure	Environment	Actuators	sensors
- Find gold in min action - Avoid wumpus and pits	- 4x4 grid - Wumpus pits, gold.	- move front, back, up, down grab, shoot, climb	- detect stench on breeze - glitter (gold)

Percept sequence :
- Assume agent starts at (1,1) and gold at (3,2)
- Move to (1,2) → perceive only breeze
- Move back and up (2,1) → Perceive only stench
- logically infer that (2,2) contains no pit or wumpus
- Similarly continue till reaching (3,2) goal



(13) Solve the following crypto arith metic problem.

SEND + MORE = MONEY.

	S	E	N	D
	(9)	(5)	(6)	
+	M	O	R	E
	(1)	(0)	(8)	(5)
M	O	N	E	Y
(1)	(0)	(6)	(5)	

Since carry is generated, M = 1

S + M = 0 generate carry.

S + 1 = 0

S = 9

O = 0

Now . E + 0 = N

But E + N ∴ 2 numbers cannot have same value.

E + 0 + generated carry and E + 1 = N

N + R = E, generated carry.

- Consider R = 8 and N = 2

Result in E = 0, not possible

If N = 3, Result = E = 1, not poss

only case possible is

N = 6 and E = 5

Now U + 5 = 4 should generate carry

N = 7 and ~~N~~ = 2

9567 + 1085 = 10652 //

0	0
1	M
2	~~N~~ Y
3	
4	
5	E
6	N
7	D
8	R
9	S

19) Consider the following axioms.

All people who are graduating are happy.

$\forall x$ (graduating (x) → happ (x))

All happy people are smiling.

$\forall x$ happ (x) → smiling (x).

Someone is graduating $\exists x$ (graduating (x))

Convert to clause form

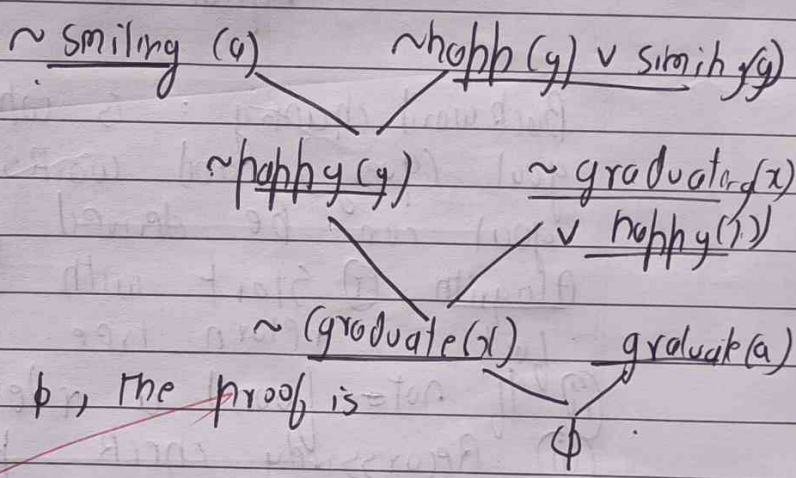
~graduating (x) ∨ happy (x)

~ happy (y) ∨ smiling (y)

graduating (a).

- Prove 'is someone smiling' → ~smiling (a)

Resolution tree



Since the tree result in ϕ, the proof is validated

∴ Someone is smiling is true.

20) Explain modes ponen with suitable example

→ It is a fundamental rule of logic stating $P \rightarrow Q = P \rightarrow Q$. If P implies Q and P is true, then Q also must be true.

Eg

- If It rains, the ground will be wet.

  Rains → wet ground.

- It is raining.

  Rains → True.

  The conclusion is that the ground is wet.

21) Explain forword and backword chaining with the help of example.

→ Forword chaining is when reasoning starts with known fact and applies inference rules to new fact until the goal is reached.

Algorithm : Start with a set of known facts.
- Apply inference rules that match current facts
- Derive new facts and add them to the KB
- Repeat until goal is achieved.

Backword chaining : is when reasoning starts with the goal (query) and works backwards to check if the goal can be derived from known facts.

Algorithm (i) Start with the goal. If goal is a known fact, return true.
(ii) If not found a rule where the goal appears.
(iii) Recursively check premises of the rule.
(iv) If all premises are true, return goal = true

Eg : (i) If person has fever and cough they have flu.
(ii) If person has flu, they should take rest.
Facts : John has fever, John has cough.
Prove - John should take rest.

Forword chaining : Fever(John), Cough(John)
Fever(x) ∧ Cough(x) → Flu(x)
∴ Flu(John) → Rest(John)

Reverse chaining : Rest(John) ∴ Flu(John)
∴ Fever(John) and Cough(John) both are true, hence statement is proved.

Name:Atharv Sanjay Nikam Div:D15C Roll:36

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

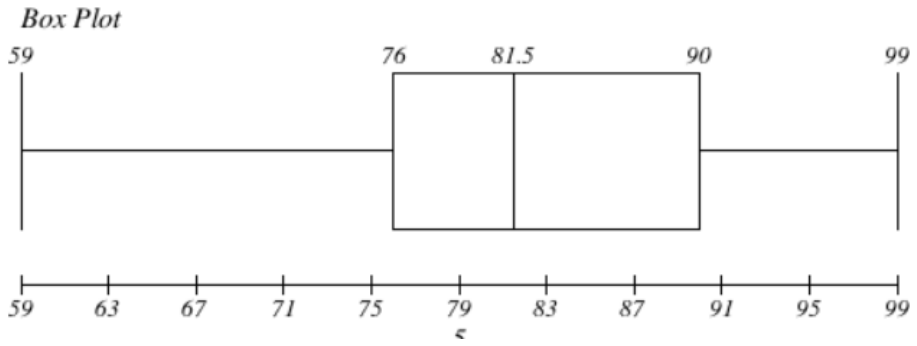
Q3 is the median of the upper half of the data. The upper half is the last 10 values. The median of the upper half is the average of the 15th and 16th values:

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:

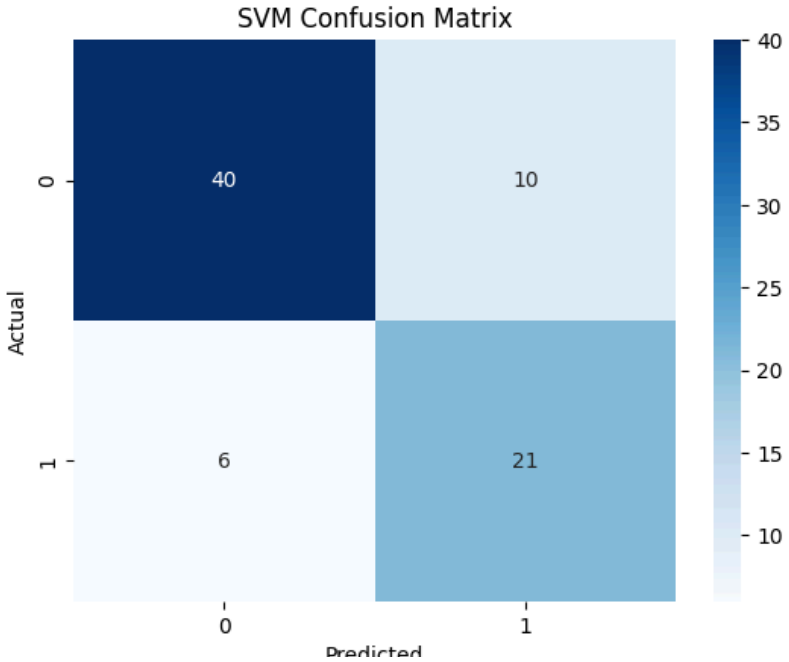
https://colab.research.google.com/drive/18TiTzAMyqyhhX-IBCBx89dH9kYWcRVSk?usp=sharing

Explanation:

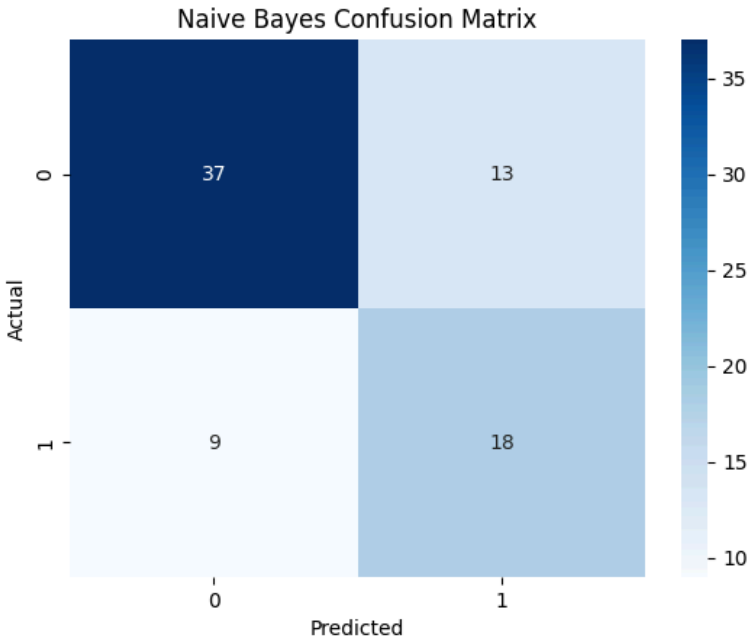
1. **Import Libraries:** Necessary libraries like numpy, pandas, sklearn, and matplotlib are imported.
2. **Load Data:** The "diabetes.csv" file is loaded using pandas.
3. **Data Separation:** The features (X) and the target variable (y) are separated. It's assumed the last column is the class label.
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:Atharv Sanjay Nikam Div:D15C Roll:36

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

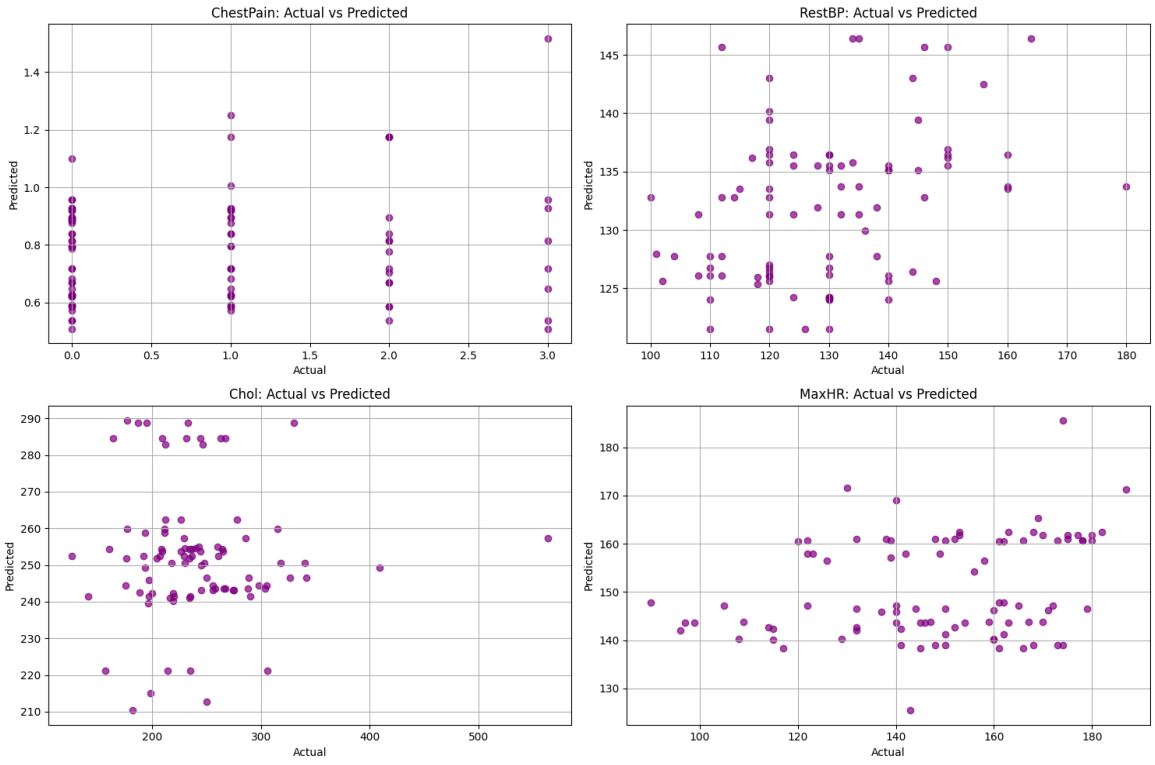
Adjusted R2 score should more than 0.99

```
  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.