Name: Sadneya Sadanand Samant
Roll no:46 **Class:**D15C

EXPERIMENT NO: 1

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

Theory:

Data preparation is a crucial step in data science, involving cleaning and transforming raw data into an analyzable format. Using Pandas, we can perform operations such as handling missing values, encoding categorical data, and scaling numerical features. Proper preprocessing ensures the dataset is reliable for analysis and modeling by addressing inconsistencies, missing data, and outliers.

Problem Statement:

The Vehicle Safety Recall dataset, provided by NHTSA, contains 15 columns detailing various aspects of recall events, such as manufacturers, affected components, and corrective actions. This analysis focuses on:

- **Manufacturer Trends**: Identifying manufacturers prone to frequent recalls or specific defects.
- **Impact Analysis**: Understanding recall types affecting the largest populations and assessing average completion rates.
- **Temporal Patterns**: Detecting trends in recalls over time and seasonal spikes.
- **Safety Implications**: Investigating critical safety advisories like "Do Not Drive" or "Park Outside" and their resolution rates.

By cleaning the dataset and applying data preprocessing steps, the goal is to enhance its quality and draw actionable insights for stakeholders.

Dataset Overview:

The dataset provides detailed information about vehicle safety recalls managed by the National Highway Traffic Safety Administration (NHTSA). It contains 15 columns, each capturing specific aspects of recall events. Below is a breakdown of the columns and their relevance:

1. **Report Received Date:** Date the recall was officially reported.
2. **NHTSA ID:** A unique identifier for each recall event.
3. **Recall Link:** A hyperlink to the recall details on the NHTSA website.
4. **Manufacturer:** Name of the vehicle or product manufacturer responsible for the recall.

5. **Subject:** Brief description of the recall issue.
6. **Component:** The affected part of the vehicle/product (e.g., "POWER TRAIN").
7. **Mfr Campaign Number:** Manufacturer's internal reference for the recall.
8. **Recall Type:** Type of product involved (e.g., vehicle, tire, or car seat).
9. **Potentially Affected:** Number of units potentially impacted by the recall.
10. **Recall Description:** Detailed explanation of the defect or issue.
11. **Consequence Summary:** Description of the risks or consequences associated with the defect.
12. **Corrective Action:** Steps taken to address the defect.
13. **Park Outside Advisory:** Indicates whether there's an advisory to park outside for safety.
14. **Do Not Drive Advisory:** Indicates whether there's an advisory not to drive the affected vehicle.
15. **Completion Rate %:** Percentage of affected vehicles repaired or addressed.

Steps:

1. Loading The Dataset

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

```
[2]  df = pd.read_csv('recalls.csv')
```

2. Description of the dataset

a. Information about dataset

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 28671 entries, 0 to 28670
Data columns (total 15 columns):
 #   Column                                  Non-Null Count  Dtype
---  ------                                  --------------  -----
 0   Report Received Date                    28671 non-null  object
 1   NHTSA ID                                28671 non-null  object
 2   Recall Link                             28671 non-null  object
 3   Manufacturer                            28671 non-null  object
 4   Subject                                 28671 non-null  object
 5   Component                               28671 non-null  object
 6   Mfr Campaign Number                     28624 non-null  object
 7   Recall Type                             28671 non-null  object
 8   Potentially Affected                    28630 non-null  float64
 9   Recall Description                      26270 non-null  object
 10  Consequence Summary                     23783 non-null  object
 11  Corrective Action                       26283 non-null  object
 12  Park Outside Advisory                   28671 non-null  object
 13  Do Not Drive Advisory                   28671 non-null  object
 14  Completion Rate % (Blank - Not Reported)  10007 non-null  float64
dtypes: float64(2), object(13)
memory usage: 3.3+ MB
```

b. Description of Dataset

```
# Get the dataset's shape and basic statistics
print(f"Dataset Shape: {df.shape}")
print(df.describe(include='all'))
```

```
Dataset Shape: (28671, 15)
       Report Received Date   NHTSA ID  \
count                 28671      28671
unique                10023      28671
top                10/17/2013  25E002000
freq                     42          1
mean                    NaN        NaN
std                     NaN        NaN
min                     NaN        NaN
25%                     NaN        NaN
50%                     NaN        NaN
75%                     NaN        NaN
max                     NaN        NaN
```

```
                                                Recall Link  \
count                                                 28671
unique                                                28671
top     Go to Recall (https://www.nhtsa.gov/recalls?nh...
freq                                                      1
mean                                                    NaN
std                                                     NaN
min                                                     NaN
25%                                                     NaN
50%                                                     NaN
75%                                                     NaN
max                                                     NaN
```

```
       Mfr Campaign Number Recall Type  Potentially Affected  \
count                28624       28671          2.863000e+04
unique               11341           4                   NaN
top      NR (Not Reported)     Vehicle                   NaN
freq                 16602       24940                   NaN
mean                   NaN         NaN          4.572011e+04
std                    NaN         NaN          3.730381e+05
min                    NaN         NaN          0.000000e+00
25%                    NaN         NaN          9.900000e+01
50%                    NaN         NaN          6.860000e+02
75%                    NaN         NaN          6.385500e+03
max                    NaN         NaN          3.200000e+07
```

```
                                          Recall Description  \
count                                                  26270
unique                                                 25523
top     ON CERTAIN TRAILERS EQUIPPED WITH SEALCO SPRIN...
freq                                                      28
mean                                                     NaN
std                                                      NaN
min                                                      NaN
25%                                                      NaN
50%                                                      NaN
75%                                                      NaN
max                                                      NaN
```

```
                                       Consequence Summary  \
count                                                23783
unique                                               17015
top     RELEASE OF COOLANT UNDER CERTAIN CONDITIONS CO...
freq                                                   128
mean                                                   NaN
std                                                    NaN
min                                                    NaN
25%                                                    NaN
50%                                                    NaN
75%                                                    NaN
max                                                    NaN
```

```
                                          Corrective Action  \
count                                                 26283
unique                                                25579
top     DEALERS WILL EQUIP AIR SYSTEMS WITH A PRESSURE...
freq                                                     18
mean                                                    NaN
std                                                     NaN
min                                                     NaN
25%                                                     NaN
50%                                                     NaN
75%                                                     NaN
max                                                     NaN
```

```
       Park Outside Advisory  Do Not Drive Advisory  \
count                  28671                  28671
unique                     2                      2
top                       No                     No
freq                   28601                  28510
mean                     NaN                    NaN
std                      NaN                    NaN
min                      NaN                    NaN
25%                      NaN                    NaN
50%                      NaN                    NaN
75%                      NaN                    NaN
max                      NaN                    NaN
```

```
       Completion Rate % (Blank - Not Reported)
count                              10007.000000
unique                                      NaN
top                                         NaN
freq                                        NaN
mean                                  67.874214
std                                   29.937993
min                                    0.000000
25%                                   48.350000
50%                                   76.390000
75%                                   93.765000
max                                  100.000000
```

3. Drop columns that aren't useful.

```
# Remove leading/trailing spaces from column names
df.columns = df.columns.str.strip()

# List of columns to drop
cols = ["Recall Link", "Mfr Campaign Number","Park Outside Advisory", "Do Not Drive Advisory", "Completion Rate % (Blank - Not Reported)"]
```

```
# Drop the columns that are present in the DataFrame
df = df.drop(cols, axis=1)

# Display the updated DataFrame
print(df.head())
```

```
  Report Received Date   NHTSA ID               Manufacturer  \
0           01/14/2025  25E002000              GKN Automotive
1           01/13/2025  25E001000  N&B Mobility Solutions LLC
2           01/13/2025  25V005000          Forest River, Inc.
3           01/13/2025  25V006000           Kia America, Inc.
4           01/13/2025  25V007000  Winnebago Industries, Inc.

                                      Subject          Component  \
0                           Driveshaft Can Break        POWER TRAIN
1    Charger Adapter May Cause Arcing or Shock Risk  ELECTRICAL SYSTEM
2  Cooktop Burner Tube May Crack and Cause Gas Leak          EQUIPMENT
3       Loss of Headlights and Taillights/FMVSS 108  ELECTRICAL SYSTEM
4                    Spare Tire Carrier May Detach          EQUIPMENT

  Recall Type  Potentially Affected  \
0   Equipment                 18.0
1   Equipment                130.0
2     Vehicle                396.0
3     Vehicle              74469.0
4     Vehicle                107.0
```

```
                                  Recall Description  \
0  GKN Automotive (GKN) is recalling certain repl...
1  N&B Mobility Solutions LLC (Nivion) is recalli...
2  Forest River, Inc. (Forest River) is recalling...
3  Kia America, Inc. (Kia) is recalling certain 2...
4  Winnebago Industries, Inc. (Winnebago) is reca...

                                 Consequence Summary  \
0  A cracked or broken driveshaft can cause a los...
1  Inadequate clearance between DC busbars may ca...
2  A gas leak in the presence of an ignition sour...
3  A loss of headlights and taillights can reduce...
4  A detached spare tire carrier can become a roa...

                                   Corrective Action
0  GKN will reimburse the cost of a replacement d...
1  Nivion will replace the defective adapters, fr...
2  Owners are advised not to use the cooktop unti...
3  Dealers will update the BDC software, free of ...
4  Dealers will inspect, replace, and correctly t...
```

Thus the columns now present in dataset are:

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 28671 entries, 0 to 28670
Data columns (total 10 columns):
 #   Column                Non-Null Count  Dtype  
---  ------                --------------  -----  
 0   Report Received Date  28671 non-null  object 
 1   NHTSA ID              28671 non-null  object 
 2   Manufacturer          28671 non-null  object 
 3   Subject               28671 non-null  object 
 4   Component             28671 non-null  object 
 5   Recall Type           28671 non-null  object 
 6   Potentially Affected  28630 non-null  float64
 7   Recall Description    26270 non-null  object 
 8   Consequence Summary   23783 non-null  object 
 9   Corrective Action     26283 non-null  object 
dtypes: float64(1), object(9)
memory usage: 2.2+ MB
```

4. **Take care of missing data.**
    a. **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)
```

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

```
Report Received Date       0
NHTSA ID                   0
Manufacturer               0
Subject                    0
Component                  0
Recall Type                0
Potentially Affected      41
Recall Description      2401
Consequence Summary     4888
Corrective Action       2388
dtype: int64
```

b. Handle Missing Data

Here above info says Potential Affected ,Recall Description ,Consequence Summary and corrective action contain some null values thus we need to handle missing data.

```
[12] # Fill missing numerical values with the median
df['Potentially Affected'] = df['Potentially Affected'].fillna(df['Potentially Affected'].median())
# Fill missing categorical values with a placeholder
df['Recall Description'] = df['Recall Description'].fillna('Not Known')
df['Consequence Summary'] = df['Consequence Summary'].fillna('Unknown')
df['Corrective Action'] = df['Corrective Action'].fillna('Unknown')

print(df.isnull().sum())  # Verify no missing values remain
```

```
Report Received Date    0
NHTSA ID                0
Manufacturer            0
Subject                 0
Component               0
Recall Type             0
Potentially Affected    0
Recall Description      0
Consequence Summary     0
Corrective Action       0
dtype: int64
```

5. Create dummy variables

```
# Convert categorical columns into dummy variables
df = pd.get_dummies(df, columns=['Recall Type'], drop_first=True)

print(df.head())
```

```
  Report Received Date   NHTSA ID               Manufacturer  \
0           01/14/2025  25E002000              GKN Automotive
1           01/13/2025  25E001000  N&B Mobility Solutions LLC
2           01/13/2025  25V005000          Forest River, Inc.
3           01/13/2025  25V006000           Kia America, Inc.
4           01/13/2025  25V007000  Winnebago Industries, Inc.

                                        Subject          Component  \
0                             Driveshaft Can Break        POWER TRAIN
1    Charger Adapter May Cause Arcing or Shock Risk  ELECTRICAL SYSTEM
2  Cooktop Burner Tube May Crack and Cause Gas Leak          EQUIPMENT
3       Loss of Headlights and Taillights/FMVSS 108  ELECTRICAL SYSTEM
4                     Spare Tire Carrier May Detach          EQUIPMENT

   Potentially Affected                                Recall Description  \
0                  18.0  GKN Automotive (GKN) is recalling certain repl...
1                 130.0  N&B Mobility Solutions LLC (Nivion) is recalli...
2                 396.0  Forest River, Inc. (Forest River) is recalling...
3               74469.0  Kia America, Inc. (Kia) is recalling certain 2...
4                 107.0  Winnebago Industries, Inc. (Winnebago) is reca...
```

```
                                Consequence Summary  \
 0  A cracked or broken driveshaft can cause a los...
 1  Inadequate clearance between DC busbars may ca...
 2  A gas leak in the presence of an ignition sour...
 3  A loss of headlights and taillights can reduce...
 4  A detached spare tire carrier can become a roa...

                                  Corrective Action  Recall Type_Equipment  \
 0  GKN will reimburse the cost of a replacement d...                   True
 1  Nivion will replace the defective adapters, fr...                   True
 2  Owners are advised not to use the cooktop unti...                  False
 3  Dealers will update the BDC software, free of ...                  False
 4  Dealers will inspect, replace, and correctly t...                  False

   Recall Type_Tire  Recall Type_Vehicle
 0            False                False
 1            False                False
 2            False                 True
```

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 28671 entries, 0 to 28670
Data columns (total 12 columns):
 #   Column                 Non-Null Count  Dtype  
---  ------                 --------------  -----  
 0   Report Received Date   28671 non-null  object 
 1   NHTSA ID               28671 non-null  object 
 2   Manufacturer           28671 non-null  object 
 3   Subject                28671 non-null  object 
 4   Component              28671 non-null  object 
 5   Potentially Affected   28671 non-null  float64
 6   Recall Description     28671 non-null  object 
 7   Consequence Summary    28671 non-null  object 
 8   Corrective Action      28671 non-null  object 
 9   Recall Type_Equipment  28671 non-null  bool   
 10  Recall Type_Tire       28671 non-null  bool   
 11  Recall Type_Vehicle    28671 non-null  bool   
dtypes: bool(3), float64(1), object(8)
memory usage: 2.1+ MB
```

6. Find out outliers (manually)

```python
import numpy as np

# Specify the column to analyze for outliers
col = 'Potentially Affected'

# Calculate Q1, Q3, and IQR
Q1 = df[col].quantile(0.25)
Q3 = df[col].quantile(0.75)
IQR = Q3 - Q1

# Define lower and upper bounds
lower_bound = Q1 - 1.5 * IQR
upper_bound = Q3 + 1.5 * IQR

# Identify outliers
outliers = df[(df[col] < lower_bound) | (df[col] > upper_bound)]

# Display the outliers
print(f"Outliers in '{col}':")
print(outliers)
```

```
Outliers in 'Potentially Affected':
      Report Received Date   NHTSA ID                    Manufacturer  \
3               01/13/2025  25V006000                Kia America, Inc.
7               01/06/2025  25V002000                     Tesla, Inc.
14              12/23/2024  24E110000                 Horizon Global
21              12/20/2024  24V957000               Ford Motor Company
22              12/20/2024  24V954000               Ford Motor Company
...                    ...       ...                             ...
28658           10/06/1966  66V004002               Ford Motor Company
28666           09/29/1966  66V003000  Honda (American Honda Motor Co.)
28668           01/19/1966  66V032001               General Motors, LLC
28669           01/19/1966  66V032003               General Motors, LLC
28670           01/19/1966  66V032004               General Motors, LLC

                                            Subject  \
3            Loss of Headlights and Taillights/FMVSS 108
7               Rearview Camera Image May Fail/FMVSS 111
14     Tow Vehicle May Separate From Hitch Receiver Lock
21                    High Pressure Fuel Pump May Fail
22                High Voltage Battery May Short Circuit
```

```
                                                Subject  \
3            Loss of Headlights and Taillights/FMVSS 108
7               Rearview Camera Image May Fail/FMVSS 111
14     Tow Vehicle May Separate From Hitch Receiver Lock
21                    High Pressure Fuel Pump May Fail
22                High Voltage Battery May Short Circuit
...                                                  ...
28658  INTERIOR SYSTEMS:RESTRAINT:BELT ANCHOR AND ATT...
28666           POWER TRAIN:TRANSMISSION:STANDARD:MANUAL
28668                                 STEERING:COLUMN
28669                                 STEERING:COLUMN
28670                                 STEERING:COLUMN

                   Component  Potentially Affected  \
3         ELECTRICAL SYSTEM                74469.0
7      BACK OVER PREVENTION               239382.0
14          TRAILER HITCHES               145431.0
21      FUEL SYSTEM, DIESEL               295449.0
22        ELECTRICAL SYSTEM                20484.0
...                     ...                    ...
28658            SEAT BELTS                65000.0
28666           POWER TRAIN                18572.0
28668              STEERING               138878.0
28669              STEERING                70644.0
28670              STEERING                68184.0
```

```
                                      Recall Description  \
3      Kia America, Inc. (Kia) is recalling certain 2...
7      Tesla, Inc. (Tesla) is recalling certain 2024-...
14     Horizon Global (Horizon) is recalling certain ...
21     Ford Motor Company (Ford) is recalling certain...
22     Ford Motor Company (Ford) is recalling certain...
...                                                  ...
28658                                          Not Known
28666                                          Not Known
28668                                          Not Known
28669                                          Not Known
28670                                          Not Known

                                    Consequence Summary  \
3      A loss of headlights and taillights can reduce...
7      A rearview camera that does not display an ima...
14     A separated cap can allow the hitch to separat...
21     High pressure Fuel pump failure can cause a lo...
22     Battery failure can cause a loss of drive powe...
...                                                  ...
28658                                          Unknown
28666                                          Unknown
28668                                          Unknown
28669                                          Unknown
28670                                          Unknown
```

```
                                      Corrective Action  \
3      Dealers will update the BDC software, free of ...
7      Tesla released an over-the-air (OTA) software ...
14     Dealers will replace the hitch receiver locks,...
21     Dealers will update the powertrain control mod...
22     Dealers will perform a battery energy control ...
...                                                  ...
28658                                          Unknown
28666                                          Unknown
28668                                          Unknown
28669                                          Unknown
28670                                          Unknown

       Recall Type_Equipment  Recall Type_Tire  Recall Type_Vehicle
3                      False             False                 True
7                      False             False                 True
14                      True             False                False
21                     False             False                 True
22                     False             False                 True
...                      ...               ...                  ...
28658                  False             False                 True
28666                  False             False                 True
28668                  False             False                 True
28669                  False             False                 True
28670                  False             False                 True

[5063 rows x 12 columns]
```

7. standardization and normalization of column

```
from sklearn.preprocessing import StandardScaler, MinMaxScaler
# Standardization: Transform data to have a mean of 0 and a standard deviation of 1
standard_scaler = StandardScaler()
df['Potentially Affected (Standardized)'] = standard_scaler.fit_transform(df[['Potentially Affected']])

# Normalization: Scale data between 0 and 1
min_max_scaler = MinMaxScaler()
df['Potentially Affected (Normalized)'] = min_max_scaler.fit_transform(df[['Potentially Affected']])

# Display the updated DataFrame
print(df[['Potentially Affected', 'Potentially Affected (Standardized)', 'Potentially Affected (Normalized)']].head())
```

```
   Potentially Affected  Potentially Affected (Standardized)  \
0                  18.0                            -0.122429
1                 130.0                            -0.122129
2                 396.0                            -0.121415
3               74469.0                             0.077295
4                 107.0                            -0.122190

   Potentially Affected (Normalized)
0                       5.625000e-07
1                       4.062500e-06
2                       1.237500e-05
3                       2.327156e-03
4                       3.343750e-06
```

Conclusion:

This experiment demonstrated effective data cleaning and preparation techniques. Issues such as missing values, irrelevant data, and outliers were addressed, and the dataset was scaled for uniformity. These steps are essential for ensuring high-quality data and reliable model outcomes.

Experiment No:2

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

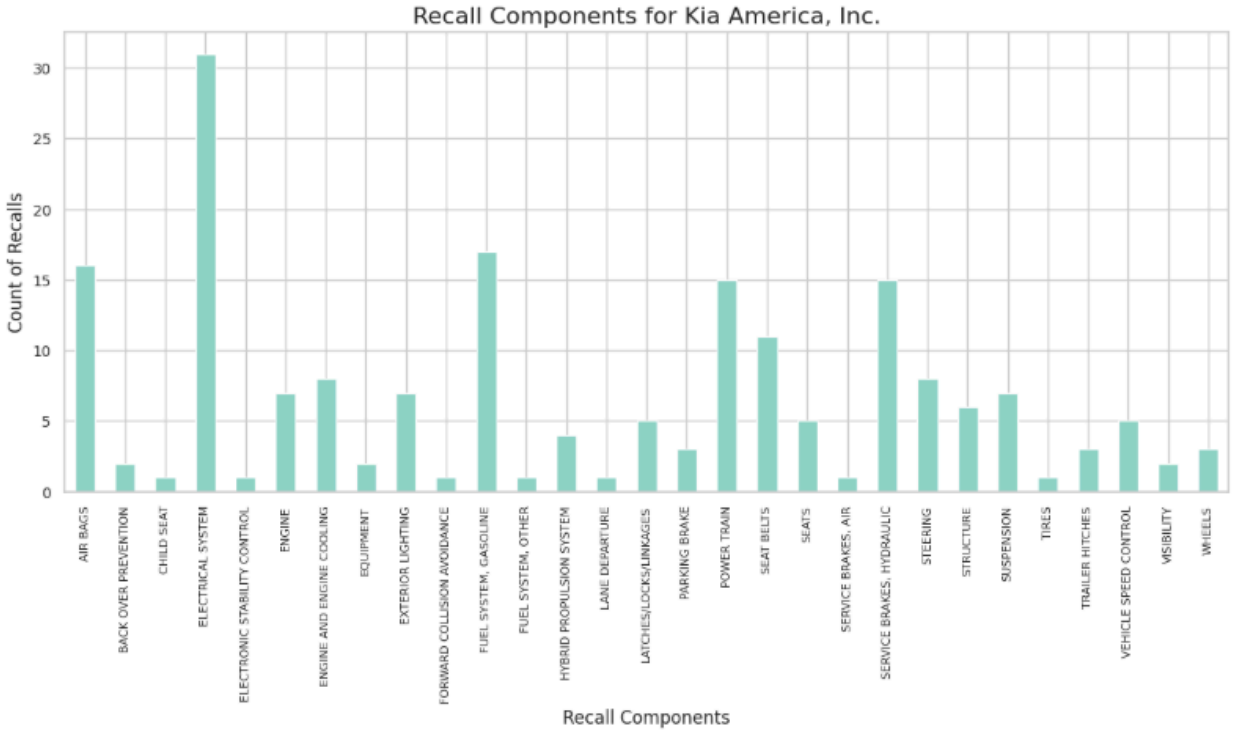
Theory:

1. Bar graph & contingency table using any two features :

1. Bar Graph:

This graph visualizes the count of recalls for different components associated with Kia America, Inc. Each bar represents a specific component and the number of recalls it has. It helps identify which components have the most recalls, aiding in targeting problem areas.

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

kia_df = df[df['Manufacturer'] == 'Kia America, Inc.']
crosstb_kia = pd.crosstab(kia_df['Component'], kia_df['Manufacturer'])
plt.figure(figsize=(12, 6))
crosstb_kia.plot.bar(figsize=(12, 6), colormap='Set3', rot=90)
plt.legend().set_visible(False)
plt.title('Recall Components for Kia America, Inc.', fontsize=16)
plt.xlabel('Recall Components', fontsize=12)
plt.ylabel('Count of Recalls', fontsize=12)
plt.xticks(fontsize=8, rotation=90)
plt.yticks(fontsize=10)
plt.tight_layout()
plt.subplots_adjust(bottom=0.2)
plt.show()
```



2.Contingency table: Displays the frequency of recalls for components against manufacturers. For Kia America, Inc., it shows the relationship between components and recall counts. Thus it helps in understanding how components vary by recall count across manufacturers.

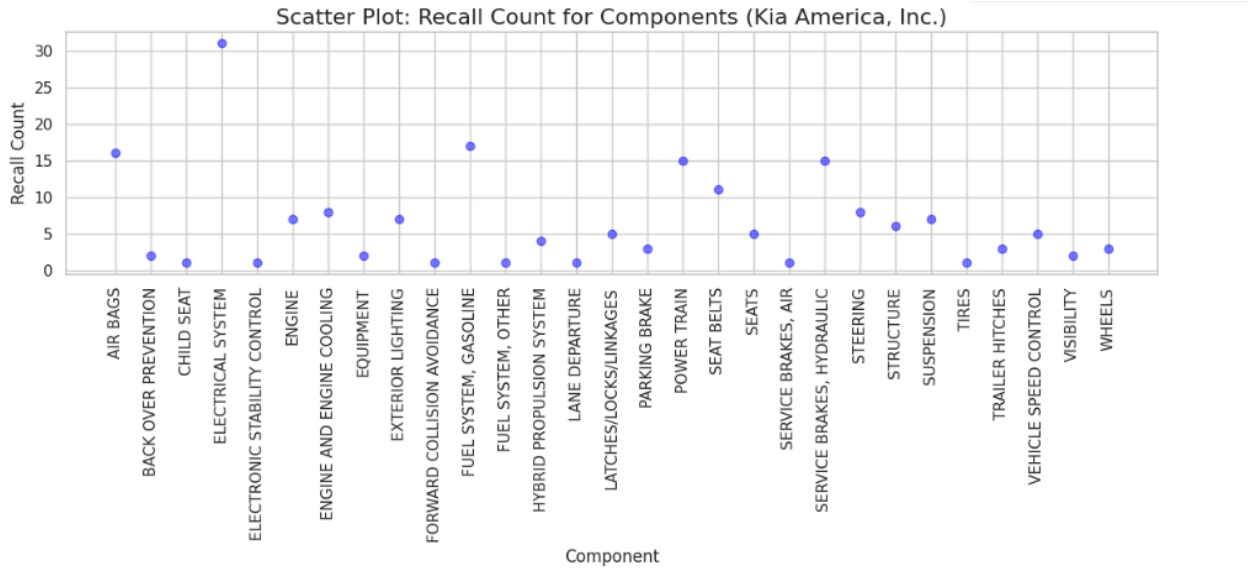
```
import pandas as pd
import matplotlib.pyplot as plt

# Assuming 'df' is your DataFrame

# Create a contingency table (crosstab) for Component vs Manufacturer
crosstb = pd.crosstab(df['Component'], df['Manufacturer'])

# Display the contingency table
print("Contingency Table:")
print(crosstb)
```

```
Contingency Table:
Manufacturer                                  1888653 Ontario Inc  \
Component
AIR BAGS                                                         0
BACK OVER PREVENTION                                             0
CHILD SEAT                                                       0
COMMUNICATION                                                    0
ELECTRICAL SYSTEM                                                0
ELECTRONIC STABILITY CONTROL                                     0
ELECTRONIC STABILITY CONTROL (ESC)                               0
ENGINE                                                           0
ENGINE AND ENGINE COOLING                                        0
EQUIPMENT                                                        1
EQUIPMENT ADAPTIVE/MOBILITY                                      0
EXTERIOR LIGHTING                                                0
FORWARD COLLISION AVOIDANCE                                      0
FUEL SYSTEM, DIESEL                                              0
FUEL SYSTEM, GASOLINE                                            0
FUEL SYSTEM, OTHER                                               0
HYBRID PROPULSION SYSTEM                                         0
INTERIOR LIGHTING                                                0
LANE DEPARTURE                                                   0
```

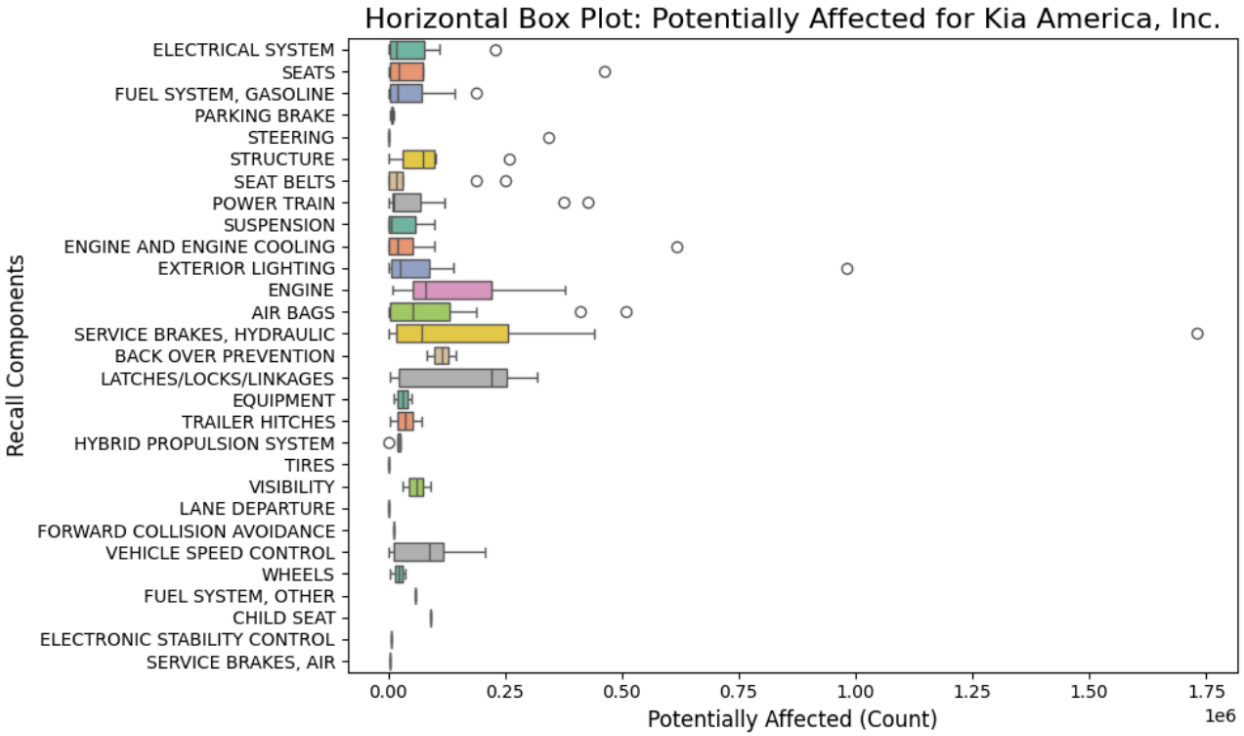
2. Plot Scatter plot, box plot, Heatmap using seaborn.

1. Scatter plot : This plot shows the recall count for various components of Kia America, Inc. Each point represents a component and its associated recall count.
Thus it highlights the distribution and patterns of recall counts, identifying outliers or components with extreme values.

```
import pandas as pd
import matplotlib.pyplot as plt
kia_df = df[df['Manufacturer'] == 'Kia America, Inc.']
crosstb_kia = pd.crosstab(kia_df['Component'], kia_df['Manufacturer'])
scatter_data = crosstb_kia.reset_index()
scatter_data = scatter_data.melt(id_vars=['Component'], value_vars=crosstb_kia.columns, var_name='Manufacturer', value_name='Recall Count')
plt.figure(figsize=(12, 6))
plt.scatter(scatter_data['Component'], scatter_data['Recall Count'], c='blue', alpha=0.5)
plt.title('Scatter Plot: Recall Count for Components (Kia America, Inc.)', fontsize=16)
plt.xlabel('Component', fontsize=12)
plt.ylabel('Recall Count', fontsize=12)
plt.xticks(rotation=90)
plt.tight_layout()
plt.show()
```



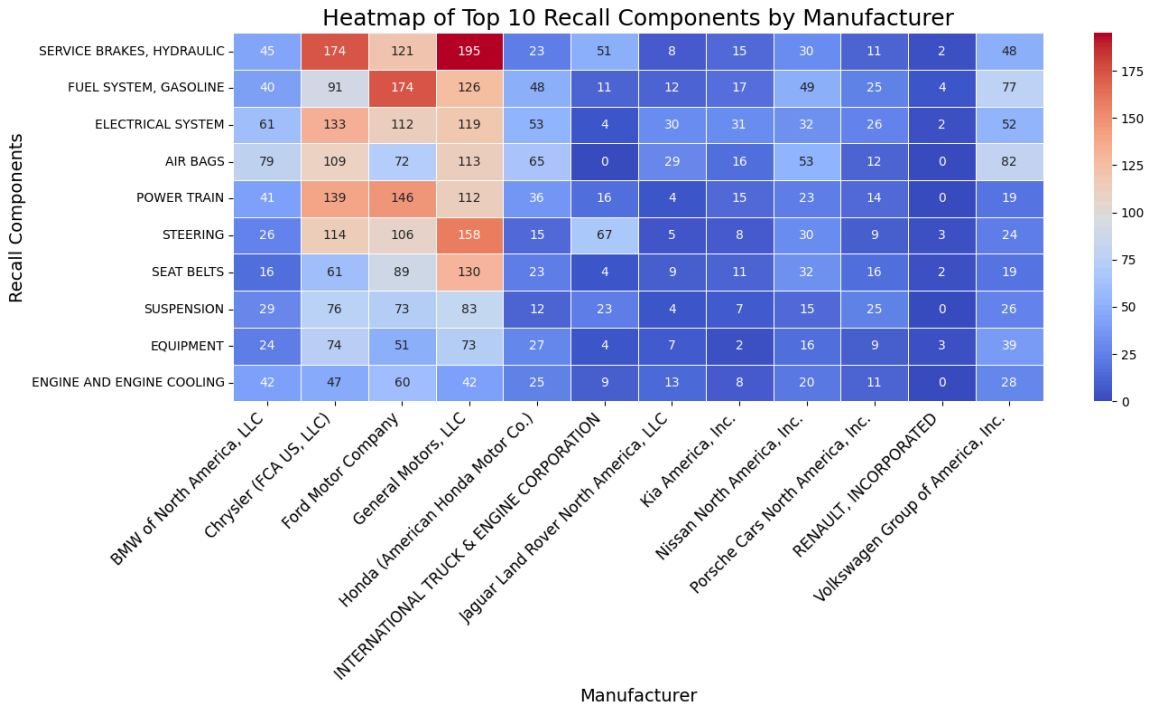
2. Box plot: The box plot shows the distribution of the number of potentially affected vehicles for each recalled component. The horizontal layout makes it easier to compare components.It identifies the spread, central tendency, and outliers in the data, with whiskers representing the data range and dots as potential outliers.




```python
#box plot
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt

kia_df = df[df['Manufacturer'] == 'Kia America, Inc.']
plt.figure(figsize=(10, 6))
sns.boxplot(x='Potentially Affected', y='Component', data=kia_df, palette='Set2')
plt.title('Horizontal Box Plot: Potentially Affected for Kia America, Inc.', fontsize=16)
plt.xlabel('Potentially Affected (Count)', fontsize=12)
plt.ylabel('Recall Components', fontsize=12)
plt.xticks(fontsize=10)
plt.yticks(fontsize=10)

plt.tight_layout()
plt.show()
```

3.Heat map: The heatmap visualizes the contingency table data, where the intensity of color represents the recall count for each component.Thus helps in identifying which components have a higher recall count at a glance, making it easy to spot patterns or correlations.

```
# Load data (ensure this is defined)
df = pd.read_csv('Recalls_Data.csv')  # Update with the correct file path

# Filter data for selected manufacturers
selected_manufacturers = [
    'Kia America, Inc.', 'Ford Motor Company', "Nissan North America, Inc.",'Mercedes-Benz
]

filtered_df = df[df['Manufacturer'].isin(selected_manufacturers)]

# Create crosstab for heatmap
heatmap_data = pd.crosstab(filtered_df['Component'], filtered_df['Manufacturer'])

# Fill NaN values with 0
heatmap_data = heatmap_data.fillna(0)

# Sort by total recalls and select only the top 10 components
top_10_components = heatmap_data.sum(axis=1).nlargest(10).index
heatmap_data = heatmap_data.loc[top_10_components]

# Plot heatmap
plt.figure(figsize=(14, 8))
sns.heatmap(heatmap_data, annot=True, cmap='coolwarm', fmt='d', linewidths=0.5)

# Labels and title
plt.title('Heatmap of Top 10 Recall Components by Manufacturer', fontsize=18)
plt.xlabel('Manufacturer', fontsize=14)
plt.ylabel('Recall Components', fontsize=14)
plt.xticks(fontsize=12, rotation=45, ha="right")
plt.yticks(fontsize=10)

plt.tight_layout()
plt.show()
```

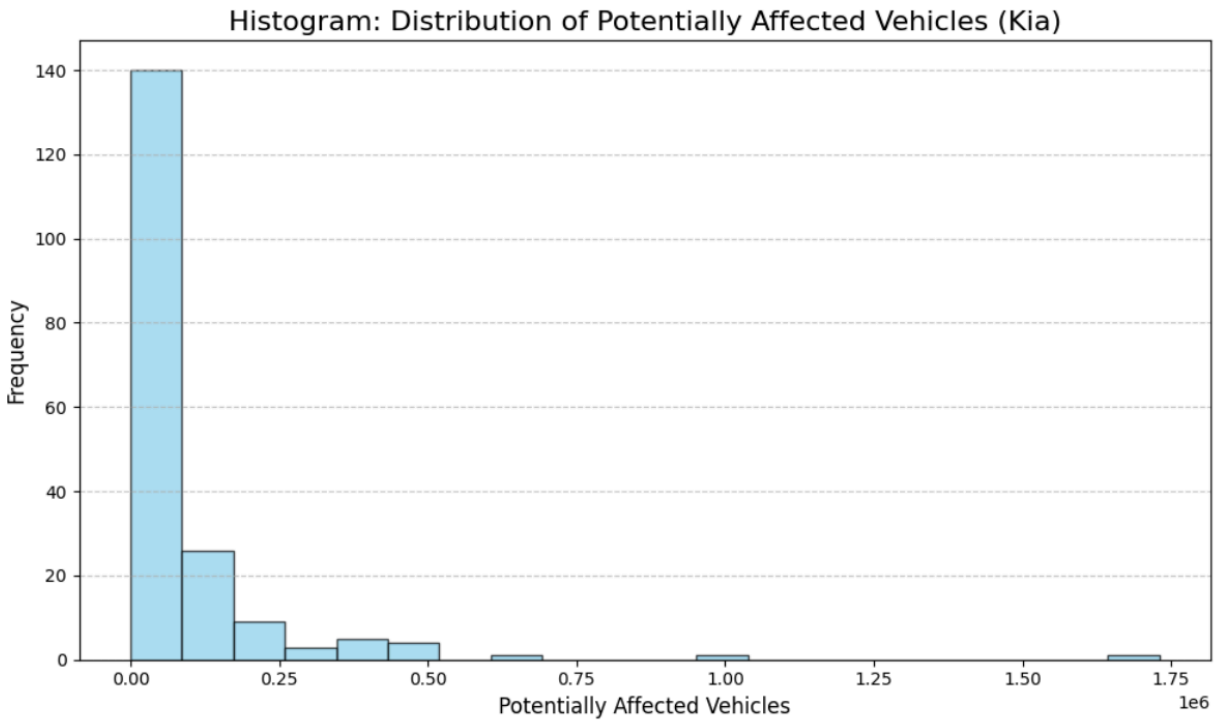


3. Create histogram and normalized Histogram.

The histogram shows the frequency distribution of the "Potentially Affected" vehicles.

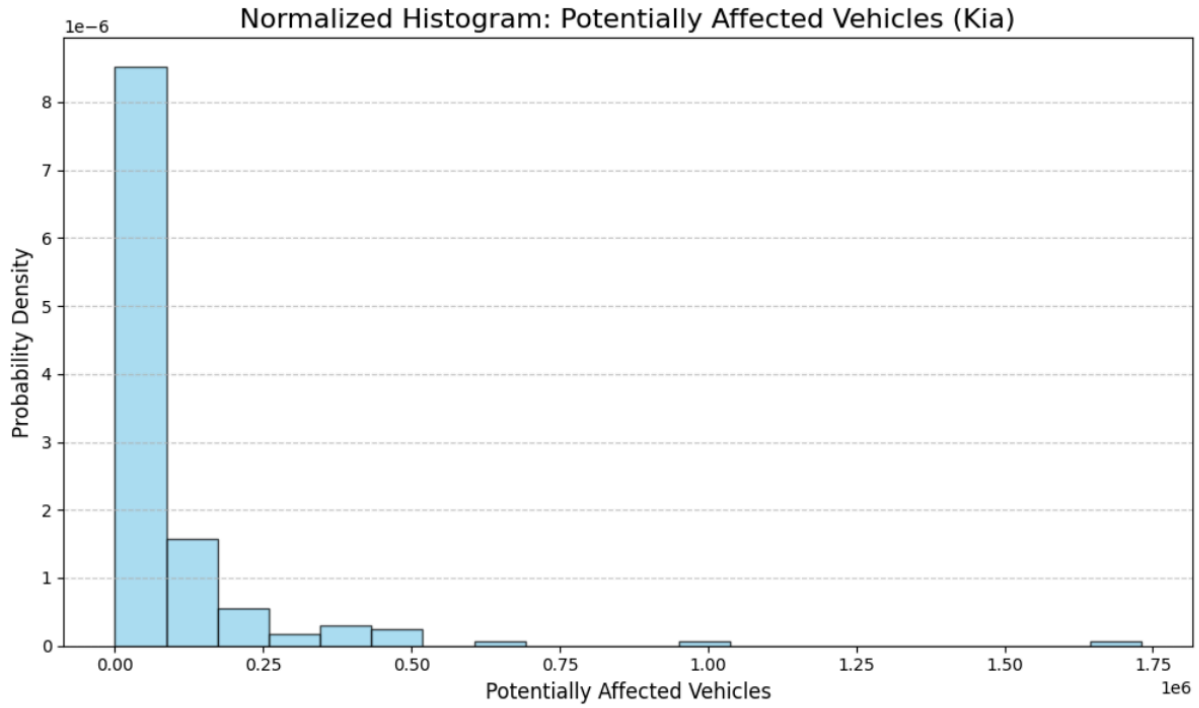
Regular Histogram: It displays counts in bins.

```
import pandas as pd, matplotlib.pyplot as plt
kia_df = df[df['Manufacturer'] == 'Kia America, Inc.']
plt.figure(figsize=(10,6))
plt.hist(kia_df['Potentially Affected'], bins=20, color='skyblue', edgecolor='black', alpha=0.7)
plt.title('Histogram: Distribution of Potentially Affected Vehicles (Kia)', fontsize=16)
plt.xlabel('Potentially Affected Vehicles', fontsize=12)
plt.ylabel('Frequency', fontsize=12)
plt.grid(axis='y', linestyle='--', alpha=0.7)
plt.tight_layout()
plt.show()
```



Normalized Histogram: It displays probability density.

```
import pandas as pd, matplotlib.pyplot as plt
kia_df = df[df['Manufacturer'] == 'Kia America, Inc.']
plt.figure(figsize=(10,6))
plt.hist(kia_df['Potentially Affected'], bins=20, color='skyblue', edgecolor='black', alpha=0.7, density=True)
plt.title('Normalized Histogram: Potentially Affected Vehicles (Kia)', fontsize=16)
plt.xlabel('Potentially Affected Vehicles', fontsize=12)
plt.ylabel('Probability Density', fontsize=12)
plt.grid(axis='y', linestyle='--', alpha=0.7)
plt.tight_layout()
plt.show()
```



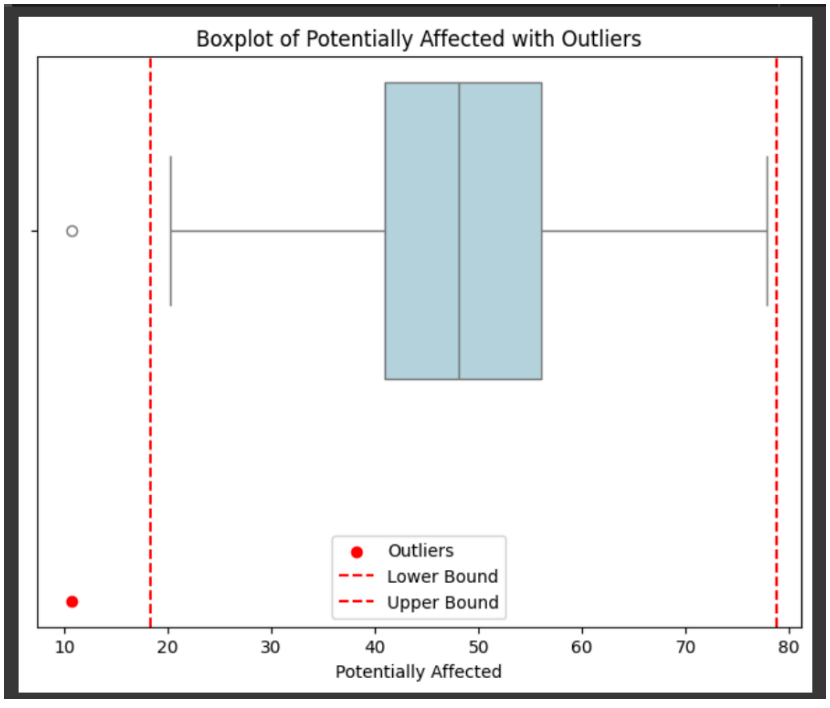
This highlights the data's skewness, central tendency, and spread. The normalized version shows the proportion of each bin relative to the total.

4. Handle outlier using box plot and Inter quartile range.

Outliers are values that deviate significantly from the rest of the data. Using the IQR method, data points beyond Q1−1.5×IQR or Q3+1.5×IQR are considered outliers.

Process: Outliers can be removed or replaced to reduce skewness and improve data accuracy.

```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
np.random.seed(42)
df = pd.DataFrame({'Potentially Affected': np.random.normal(50, 15, 100)})
col = 'Potentially Affected'
Q1 = df[col].quantile(0.25)
Q3 = df[col].quantile(0.75)
IQR = Q3 - Q1
lower_bound = Q1 - 1.5 * IQR
upper_bound = Q3 + 1.5 * IQR
outliers = df[(df[col] < lower_bound) | (df[col] > upper_bound)]
plt.figure(figsize=(8, 6))
sns.boxplot(x=df[col], color='lightblue')
plt.scatter(outliers[col], [1] * len(outliers), color='red', label='Outliers', zorder=2)
plt.axvline(lower_bound, color='red', linestyle='dashed', label='Lower Bound')
plt.axvline(upper_bound, color='red', linestyle='dashed', label='Upper Bound')
plt.title(f'Boxplot of {col} with Outliers')
plt.legend()
plt.show()
```



Conclusion: Bar Graph and Scatter Plot are suitable for identifying trends and comparisons.Heatmap is excellent for understanding correlations between variables. Box Plot provides insights into distributions and outliers.Normalized Histogram helps understand probabilities and densities.Outlier Handling improves the accuracy of the results.

Experiment 3

Aim: Perform Data Modeling on dataset.

Theory:

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

In this experiment, we partitioned a dataset of vehicle recall data into a training set (75%) and a test set (25%) and validated this partitioning using a two-sample Z-test. The dataset consists of 28,671 records with features such as manufacturer, recall components, and the number of potentially affected vehicles.

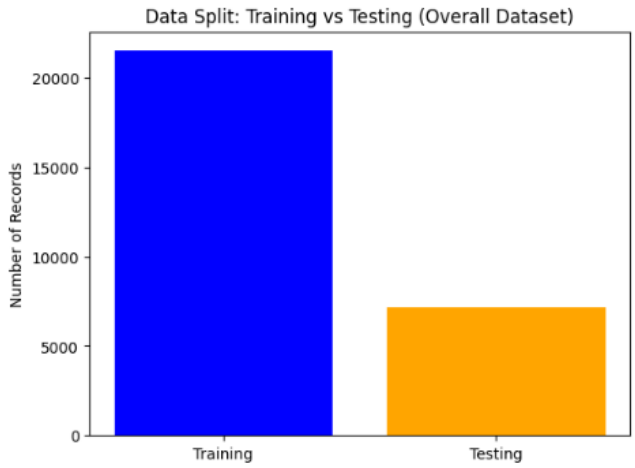
```
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
import scipy.stats as stats

df=pd.read_csv('Recalls_Data.csv')
train_data, test_data = train_test_split(df, test_size=0.25, random_state=42)
labels = ['Training', 'Testing']
sizes = [len(train_data), len(test_data)]
plt.bar(labels, sizes, color=['blue', 'orange'])
plt.title("Data Split: Training vs Testing (Overall Dataset)")
plt.ylabel("Number of Records")
plt.show()
plt.pie(sizes, labels=labels, autopct='%1.1f%%', colors=['blue', 'orange'], startangle=90)
plt.title("Data Split: Training vs Testing (Pie Chart)")
plt.show()
print("Total records in the training data set:", len(train_data))
print("Total records in the testing data set:", len(test_data))
```

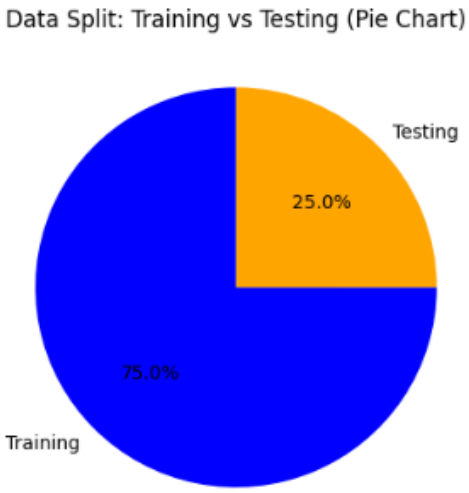
b. **Use a bar graph and other relevant graph to confirm your proportions.**

To confirm the proportions of the data split into training and test sets, we have used a bar graph and a pie chart:

Bar Graph: It displays the number of records in the training and test sets, allowing us to visually confirm the 75%-25% split. The x-axis represents the two sets, and the y-axis shows their respective record counts.

Pie Chart: This chart visually illustrates the percentage split between the training (75%) and test (25%) sets, providing an easy confirmation of the partition.





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

We used the train_test_split method to split the dataset into training and test sets. The partition was visually confirmed through a bar plot, showing the expected 75%-25% split, with 21,503 records in the training set and 7,168 in the test set.\

```
Total records in the training data set: 21503
Total records in the testing data set: 7168
```

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

To validate the partitioning, we performed a two-sample Z-test manually to compare the "Potentially Affected" values between the training and test datasets. The Z-statistic was calculated based on the means, standard deviations, and sizes of both samples. We then compared the calculated p-value to the significance level of 0.05.

The Z-test showed that there was no significant difference between the training and test datasets, as the p-value was greater than 0.05. This confirmed that the partitioning process was unbiased.

Conclusion:

The partitioning of the dataset into training and test sets was validated successfully. The Z-test confirmed that there was no significant difference between the datasets, making the partition reliable for further analysis.

Experiment 4

Aim: Implementation of Statistical Hypothesis Test using Scipy and Sci-kit learn.

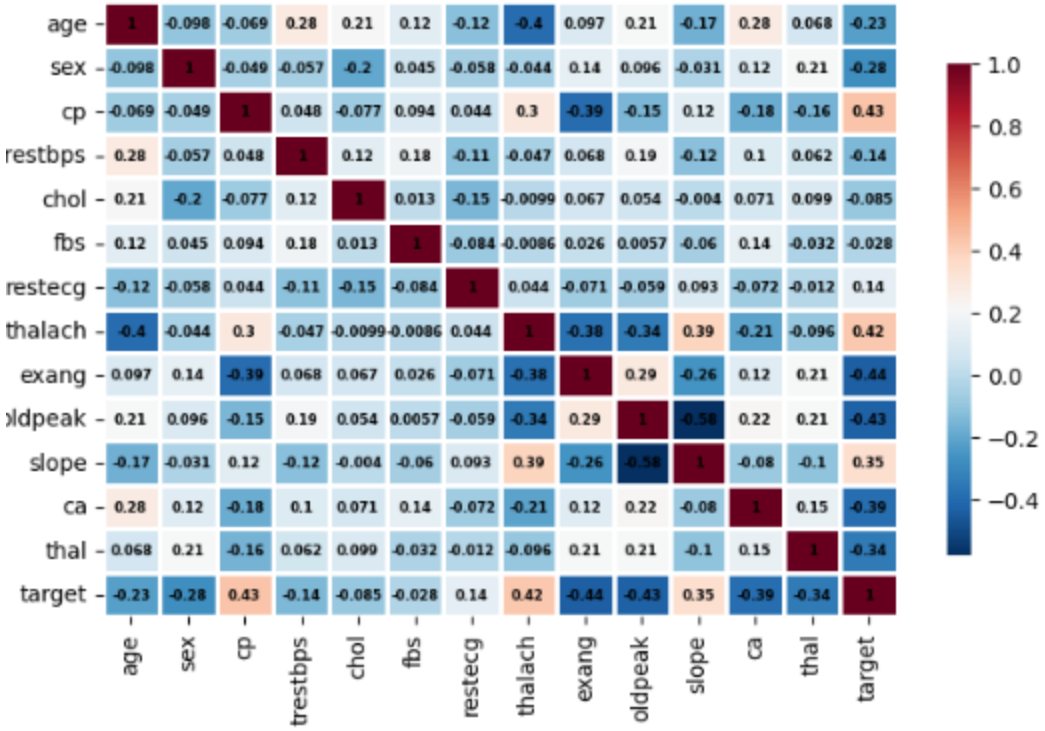
Theory:

1. **Pearson's Correlation**: Pearson's correlation measures the linear relationship between two continuous variables. A coefficient close to 1 or -1 indicates a strong linear relationship, while a value near 0 suggests no linear association.

```
pearsoncorr = df.corr(method='pearson')
pearsoncorr
```

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal	target
age	1.000000	-0.098447	-0.068653	0.279351	0.213678	0.121308	-0.116211	-0.398522	0.096801	0.210013	-0.168814	0.276326	0.068001	-0.225439
sex	-0.098447	1.000000	-0.049353	-0.056769	-0.197912	0.045032	-0.058196	-0.044020	0.141664	0.096093	-0.030711	0.118261	0.210041	-0.280937
cp	-0.068653	-0.049353	1.000000	0.047608	-0.076904	0.094444	0.044421	0.295762	-0.394280	-0.149230	0.119717	-0.181053	-0.161736	0.433798
trestbps	0.279351	-0.056769	0.047608	1.000000	0.123174	0.177531	-0.114103	-0.046698	0.067616	0.193216	-0.121475	0.101389	0.062210	-0.144931
chol	0.213678	-0.197912	-0.076904	0.123174	1.000000	0.013294	-0.151040	-0.009940	0.067023	0.053952	-0.004038	0.070511	0.098803	-0.085239
fbs	0.121308	0.045032	0.094444	0.177531	0.013294	1.000000	-0.084189	-0.008567	0.025665	0.005747	-0.059894	0.137979	-0.032019	-0.028046
restecg	-0.116211	-0.058196	0.044421	-0.114103	-0.151040	-0.084189	1.000000	0.044123	-0.070733	-0.058770	0.093045	-0.072042	-0.011981	0.137230
thalach	-0.398522	-0.044020	0.295762	-0.046698	-0.009940	-0.008567	0.044123	1.000000	-0.378812	-0.344187	0.386784	-0.213177	-0.096439	0.421741
exang	0.096801	0.141664	-0.394280	0.067616	0.067023	0.025665	-0.070733	-0.378812	1.000000	0.288223	-0.257748	0.115739	0.206754	-0.436757
oldpeak	0.210013	0.096093	-0.149230	0.193216	0.053952	0.005747	-0.058770	-0.344187	0.288223	1.000000	-0.577537	0.222682	0.210244	-0.430696
slope	-0.168814	-0.030711	0.119717	-0.121475	-0.004038	-0.059894	0.093045	0.386784	-0.257748	-0.577537	1.000000	-0.080155	-0.104764	0.345877
ca	0.276326	0.118261	-0.181053	0.101389	0.070511	0.137979	-0.072042	-0.213177	0.115739	0.222682	-0.080155	1.000000	0.151832	-0.391724
thal	0.068001	0.210041	-0.161736	0.062210	0.098803	-0.032019	-0.011981	-0.096439	0.206754	0.210244	-0.104764	0.151832	1.000000	-0.344029
target	-0.225439	-0.280937	0.433798	-0.144931	-0.085239	-0.028046	0.137230	0.421741	-0.436757	-0.430696	0.345877	-0.391724	-0.344029	1.000000

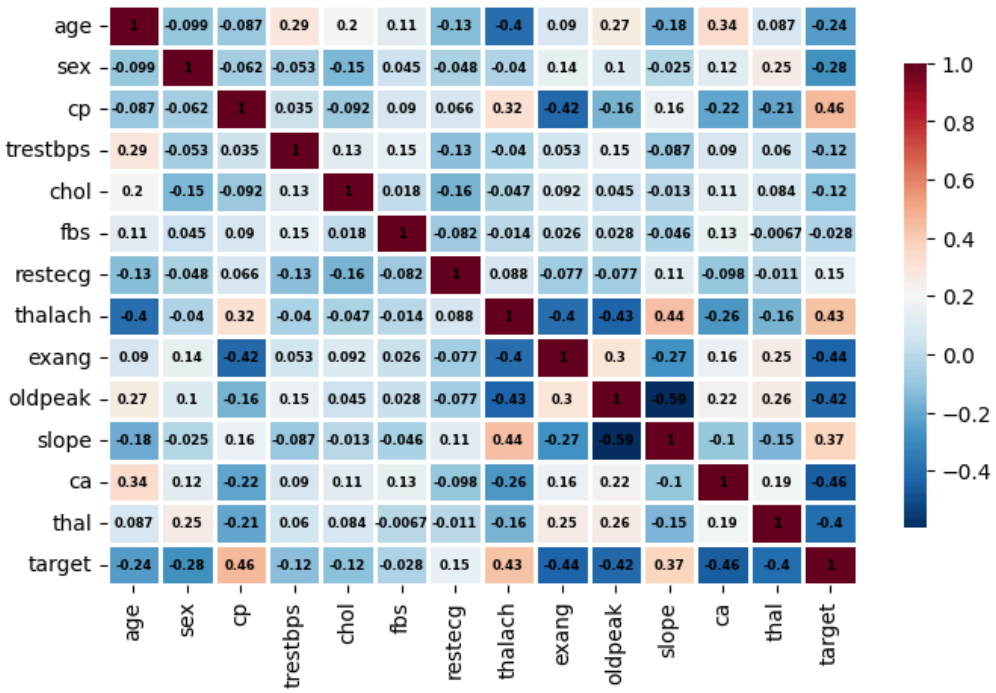
```
import seaborn as sb
import matplotlib.pyplot as plt

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

sb.heatmap(pearsoncorr,
           xticklabels=pearsoncorr.columns,
           yticklabels=pearsoncorr.columns,
           cmap='RdBu_r',
           annot=True,
           annot_kws={"size": 6, "weight": "bold", "color": "black"},
           linewidth=2,
           cbar_kws={"shrink": 0.8})

plt.show()
```



Result: There is a moderate positive relationship between cp and heart disease (target), with a correlation of 0.43.

2. **Spearman's Rank Correlation**: Spearman's rank correlation assesses the monotonic relationship between variables, relying on their ranks rather than raw data. It is suitable for ordinal or non-linear relationships.

```
spearmancorr = df.corr(method='spearman')

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

sb.heatmap(spearmancorr,
           xticklabels=spearmancorr.columns,
           yticklabels=spearmancorr.columns,
           cmap='RdBu_r',
           annot=True,
           annot_kws={"size": 6, "weight": "bold", "color": "black"},
           linewidth=2,
           cbar_kws={"shrink": 0.8})

plt.show()
```

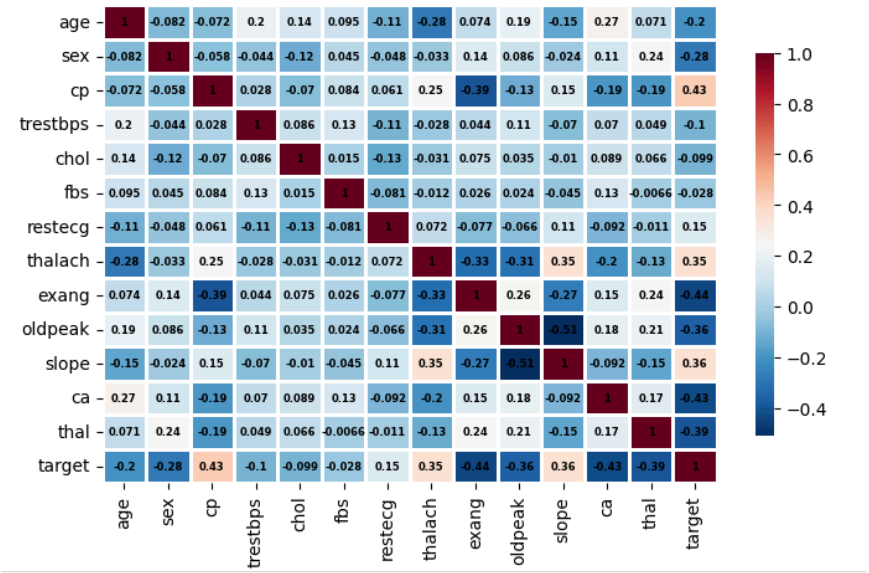


Results: The correlation between cp (chest pain type) and target is 0.46, indicating a moderate positive association.

3. **Kendall's Rank Correlation**: Kendall's Tau measures the strength of the ordinal relationship between two variables by comparing the ranks of pairs. It is more robust to ties in data.

```
from scipy.stats import pearsonr, spearmanr, kendalltau, chi2_contingency
```

```
kendallcorr = df.corr(method='kendall')

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

sb.heatmap(kendallcorr,
           xticklabels=kendallcorr.columns,
           yticklabels=kendallcorr.columns,
           cmap='RdBu_r',
           annot=True,
           annot_kws={"size": 6, "weight": "bold", "color": "black"},
           linewidth=2,
           cbar_kws={"shrink": 0.8})

plt.show()
```



Results: cp and target have a Kendall's Tau of 0.43, showing a positive relationship. target and thalach show a weaker but still positive correlation (0.35), indicating heart rate's relevance in predicting heart disease.

4. **Chi-Squared Test**: The Chi-Squared test assesses the independence of two categorical variables. A significant p-value indicates that the variables are dependent (associated).

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

contingency_table = pd.crosstab(df['sex'], df['target'])

chi2, p, dof, expected = chi2_contingency(contingency_table)

print("Chi-Squared Statistic:", chi2)
print("P-value:", p)
print("Degrees of Freedom:", dof)
print("Expected frequencies table:")
print(expected)

if p < 0.05:
    print("There is a significant association between the variables (reject the null hypothesis).")
else:
    print("There is no significant association between the variables (fail to reject the null hypothesis).")
```

```
Chi-Squared Statistic: 22.717227046576355
P-value: 1.8767776216941503e-06
Degrees of Freedom: 1
Expected frequencies table:
[[ 43.72277228  52.27722772]
 [ 94.27722772 112.72277228]]
There is a significant association between the variables (reject the null hypothesis).
```

Result: The Chi-Squared statistic of 22.72 (p-value = 1.88e-06) indicates a significant association between categorical variables (e.g., sex and target), suggesting their role in heart disease prediction.

Conclusion:

In this analysis, four statistical tests were applied to assess the relationships between various features and heart disease:

1. Pearson's correlation showed a moderate positive relationship between cp and heart disease (target), with a correlation of 0.43.
2. Spearman's rank correlation confirmed a moderate positive relationship between cp (chest pain type) and target (0.46).
3. Kendall's Tau also revealed a moderate positive association between cp and target (0.43)
4. The Chi-Squared test showed a significant association between categorical variables, with a Chi-Squared statistic of 22.72 and a p-value of 1.88e-06.Since the p-value of 1.88e-06 is much smaller than the commonly used significance level of 0.05, we reject the null hypothesis.

Experiment 5

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

Theory:

1. Linear Regression:

Linear Regression is a supervised learning algorithm used for predicting continuous numerical values.
It assumes a linear relationship between the independent variables (features) and the dependent variable (target).
The model minimizes the sum of squared residuals (differences between actual and predicted values) to find the best-fitting line.
It is sensitive to outliers and works best when the data follows a linear trend. Performance is evaluated using metrics like Mean Squared Error (MSE) and R-squared (R^2).

2. Logistic Regression:

Logistic Regression is a classification algorithm used for predicting binary outcomes (0 or 1).
It applies the logistic (sigmoid) function to transform linear outputs into probabilities.
The model predicts a class based on a threshold, typically 0.5.
It uses optimization techniques like gradient descent to minimize the loss function (log loss).
Performance is assessed using accuracy, precision, recall, F1-score, and confusion matrices.

Implementation:

1. **Import necessary libraries and load the dataset**

```
[ ]  import pandas as pd
     import numpy as np
     import seaborn as sns
     import matplotlib.pyplot as plt
     from sklearn.model_selection import train_test_split
     from sklearn.linear_model import LinearRegression, LogisticRegression
     from sklearn.preprocessing import StandardScaler
     from sklearn.impute import SimpleImputer
     from sklearn.metrics import mean_squared_error, r2_score, accuracy_score
```

```
[ ] df = pd.read_csv('diabetes_dataset_with_notes.csv')
```

```
[ ] df.head()
```

	year	gender	age	location	race:AfricanAmerican	race:Asian	race:Caucasian	race:Hispanic	race:Other	hypertension
0	2020	Female	32.0	Alabama	0	0	0	0	1	0
1	2015	Female	29.0	Alabama	0	1	0	0	0	0
2	2015	Male	18.0	Alabama	0	0	0	0	1	0
3	2015	Male	41.0	Alabama	0	0	1	0	0	0
4	2016	Female	52.0	Alabama	1	0	0	0	0	0

```
df.describe()
```

	year	age	race:AfricanAmerican	race:Asian	race:Caucasian
count	100000.000000	100000.000000	100000.000000	100000.000000	100000.000000
mean	2018.360820	41.885856	0.202230	0.200150	0.198760
std	1.345239	22.516840	0.401665	0.400114	0.399069
min	2015.000000	0.080000	0.000000	0.000000	0.000000
25%	2019.000000	24.000000	0.000000	0.000000	0.000000
50%	2019.000000	43.000000	0.000000	0.000000	0.000000
75%	2019.000000	60.000000	0.000000	0.000000	0.000000
max	2022.000000	80.000000	1.000000	1.000000	1.000000

2. Check for missing values and anomalies

```
print("\nMissing values after handling:\n", X.isnull().sum())
```

```
Missing values after handling:
 age                    0
gender                  0
bmi                     0
hbA1c_level             0
hypertension            0
heart_disease           0
bmi_age_interaction     0
hbA1c_blood_glucose     0
dtype: int64
```

```
numeric_columns = df.select_dtypes(include=[np.number]).columns
imputer = SimpleImputer(strategy="median")
df[numeric_columns] = imputer.fit_transform(df[numeric_columns])
```

```
df["bmi_age_interaction"] = df["bmi"] * df["age"]
df["hbA1c_blood_glucose"] = df["hbA1c_level"] * df["blood_glucose_level"]
```

```
df = df[df["blood_glucose_level"] < df["blood_glucose_level"].quantile(0.99)]
```

3. **Choose your columns and convert categorical values into numerical values for better classification**

```
df["gender"] = df["gender"].map({"Male": 0, "Female": 1})
df["smoking_history"] = df["smoking_history"].astype('category').cat.codes
```

```
features = ["age", "gender", "bmi", "hbA1c_level", "hypertension", "heart_disease",
            "bmi_age_interaction", "hbA1c_blood_glucose"]
```

```
X = df[features]
y_reg = df["blood_glucose_level"]  # Target for Linear Regression
y_clf = df["diabetes"]  # Target for Logistic Regression (Assuming it's binary 0/1)
```

```
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)
```

4. **Train dataset using linear regression**

```
[35]
     df = df.drop(columns=["location", "clinical_notes", "smoking_history", "gender"])

     # Define features (X) and target (y)
     X = df.drop(columns=["diabetes"])
     y = df["diabetes"]

     # Split data into training and testing sets
     X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

     # Initialize and train linear regression model
     model = LinearRegression()
     model.fit(X_train, y_train)

     # Predictions
     y_pred = model.predict(X_test)

     # Evaluate the model
     mse = mean_squared_error(y_test, y_pred)
     r2 = r2_score(y_test, y_pred)

     print(f"Mean Squared Error: {mse}")
     print(f"R-squared Score: {r2}")
```

5. **Finding out Mean Squared error and goodness of fit**

```
Mean Squared Error: 0.041312657154867635
R-squared Score: 0.25314264247732976
```

6. **Using Logistic Regression, find out accuracy store**

```python
X = df.drop(columns=["diabetes"])
y = df["diabetes"]

# Split data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

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

# Predictions
y_pred = model.predict(X_test)

# Evaluate the model
accuracy = accuracy_score(y_test, y_pred)
report = classification_report(y_test, y_pred)

print(f"Accuracy: {accuracy}")
print("Classification Report:")
print(report)
```

Accuracy Score:

```
Accuracy: 0.962550087331758
Classification Report:
              precision    recall  f1-score   support

         0.0       0.97      0.99      0.98     18322
         1.0       0.82      0.47      0.59      1144

    accuracy                           0.96     19466
   macro avg       0.89      0.73      0.79     19466
weighted avg       0.96      0.96      0.96     19466
```

Conclusion:

Linear Regression Results:

- The Mean Squared Error (MSE) of 0.0413 indicates that the model's predictions are relatively close to the actual values.

- The R-squared score of 0.2531 shows that the independent variables explain about 25% of the variance in diabetes presence.

Logistic Regression Results:

- The model achieved an accuracy of 96.25%.
- The classification report shows that it performs well for non-diabetic cases (0) with high precision and recall.
- For diabetic cases (1), the recall is 0.47, meaning that while many diabetic cases are correctly identified, some are still being misclassified.

Experiment No: 6

Aim: Perform Classification modelling
a. Choose a classifier for classification problems.
b. Evaluate the performance of the classifier.
Perform Classification using the below 4 classifiers on the same dataset:

1. K-Nearest Neighbors (KNN)
2. Naive Bayes
3. Support Vector Machines (SVMs)
4. Decision Tree

Theory:

1. **Decision tree**: A Decision Tree is a supervised learning algorithm for classification and regression that splits data based on feature values to form a tree-like structure. It uses Gini Index or Entropy to determine splits and can be pruned to prevent overfitting. While easy to interpret and handle both numerical and categorical data, it can be unstable and prone to overfitting.

```
X = df[features]
y = df["diabetes"]
X_clean = X.copy()
X_clean.replace([np.inf, -np.inf], np.nan, inplace=True)
X_clean.dropna(axis=1, how='all', inplace=True)
X_clean.dropna(axis=0, how='all', inplace=True)
X_clean = X_clean.apply(lambda col: col.fillna(col.mean()), axis=0)
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X_clean)
X_train, X_test, y_train, y_test = train_test_split(X_scaled, y, test_size=0.2, random_state=42)
dt_model = DecisionTreeClassifier(random_state=42)
dt_model.fit(X_train, y_train)
y_pred_dt = dt_model.predict(X_test)
accuracy_dt = accuracy_score(y_test, y_pred_dt)
print("\nDecision Tree Classifier Results:")
print(f"Accuracy: {accuracy_dt:.2f}")
print("Classification Report for Decision Tree:")
print(classification_report(y_test, y_pred_dt))
```

```
cm = confusion_matrix(y_test, y_pred_dt)

print("Confusion Matrix:")
print(cm)
```

1. The Decision Tree classifier achieved **95% accuracy**, indicating strong overall performance. However, a closer look at the classification report and confusion matrix reveals an imbalance in predictive capability between the two classes:

- **Class 0 (Non-Diabetic)**: The model performed exceptionally well, with **98% precision** and **97% recall**, meaning most non-diabetic cases were correctly classified.
- **Class 1 (Diabetic)**: The model struggled with diabetic cases, achieving **67% precision** and **70% recall**, meaning **30% of actual diabetic cases were misclassified as non-diabetic**.

2. From the **confusion matrix**, we observe:

- **True Negatives (TN):** 17,801 (correctly classified non-diabetic cases)
- **False Positives (FP):** 495 (misclassified non-diabetic cases as diabetic)
- **False Negatives (FN):** 421 (misclassified diabetic cases as non-diabetic)
- **True Positives (TP):** 1,003 (correctly classified diabetic cases)

```
Decision Tree Classifier Results:
Accuracy: 0.95
Classification Report for Decision Tree:
              precision    recall  f1-score   support

           0       0.98      0.97      0.97     18296
           1       0.67      0.70      0.69      1424

    accuracy                           0.95     19720
   macro avg       0.82      0.84      0.83     19720
weighted avg       0.95      0.95      0.95     19720

Confusion Matrix:
[[17801   495]
 [  421  1003]]
```

3. While the model performs well overall, the lower recall for diabetic cases indicates that a significant number of diabetic patients are not being correctly identified. This could be due to **class imbalance** in the dataset, where non-diabetic cases dominate.

2. **Support Vector Machine (SVM):** A Support Vector Machine (SVM) finds the optimal hyperplane to separate classes, using support vectors to maximize the margin. It employs the kernel trick (Linear, Polynomial, RBF) for non-linearly

separable data. SVMs perform well on high-dimensional data and small datasets but can be computationally expensive and require careful kernel selection.

```
X_clean = X.copy()
X_clean.replace([np.inf, -np.inf], np.nan, inplace=True)
X_clean.dropna(axis=1, how='all', inplace=True)
X_clean.dropna(axis=0, how='all', inplace=True)
X_clean = X_clean.apply(lambda col: col.fillna(col.mean()), axis=0)
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X_clean)
X_train, X_test, y_train, y_test = train_test_split(X_scaled, y, test_size=0.2, random_state=42)
svm_model = SVC(kernel='linear', random_state=42)
svm_model.fit(X_train, y_train)
y_pred_svm = svm_model.predict(X_test)
accuracy_svm = accuracy_score(y_test, y_pred_svm)
print("\nSupport Vector Machine (SVM) Classifier Results:")
print(f"Accuracy: {accuracy_svm:.2f}")
print("Classification Report for SVM:")
print(classification_report(y_test, y_pred_svm))
```

1. The Support Vector Machine (SVM) classifier achieved **96% accuracy**, indicating strong overall performance. However, a closer look at the classification report and confusion matrix reveals an imbalance in predictive capability between the two classes:

- **Class 0 (Non-Diabetic):** The model performed exceptionally well, with **96% precision and 100% recall**, meaning almost all non-diabetic cases were correctly classified.
- **Class 1 (Diabetic):** The model struggled with diabetic cases, achieving **92% precision and 48% recall**, meaning **52% of actual diabetic cases were misclassified as non-diabetic**.

2. From the confusion matrix, we observe:

- **True Negatives (TN):** 17,801 (correctly classified non-diabetic cases)
- **False Positives (FP):** 495 (misclassified non-diabetic cases as diabetic)
- **False Negatives (FN):** 421 (misclassified diabetic cases as non-diabetic)
- **True Positives (TP):** 1,003 (correctly classified diabetic cases)

```
Support Vector Machine (SVM) Classifier Results:
Accuracy: 0.96
Classification Report for SVM:
              precision    recall  f1-score   support

           0       0.96      1.00      0.98     18296
           1       0.92      0.48      0.64      1424

    accuracy                           0.96     19720
   macro avg       0.94      0.74      0.81     19720
weighted avg       0.96      0.96      0.95     19720

Confusion Matrix:
[[17801   495]
 [  421  1003]]
```

3. While the model performs well overall, the **low recall for diabetic cases** indicates that a significant number of diabetic patients are not being correctly identified. This could be due to class imbalance in the dataset, where non-diabetic cases dominate.

Conclusion: In this experiment, we implemented two classification models—Support Vector Machines (SVM), and Decision Tree—on the diabetes dataset to evaluate their performance.

- Decision Tree achieved high accuracy (95%) but showed a bias towards non-diabetic cases due to class imbalance. It had excellent precision and recall for non-diabetic cases but struggled with diabetic cases.
- Support Vector Machine (SVM) performed slightly better overall (96% accuracy) but had low recall (48%) for diabetic cases, meaning it misclassified many diabetic patients.

EXPERIMENT NO:7

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

Theory:

1. Importing Libraries

```
import pandas as pd
import numpy as np
from sklearn.cluster import KMeans
from sklearn.preprocessing import StandardScaler
from sklearn.decomposition import PCA
from sklearn.preprocessing import LabelEncoder
from sklearn.cluster import DBSCAN
import scipy.cluster.hierarchy as sch
import matplotlib.pyplot as plt
import seaborn as sns
```

2. Importing dataset

```
df=pd.read_csv('diabetes_dataset_with_notes.csv')
df = pd.DataFrame(df)
```

3. Transformation

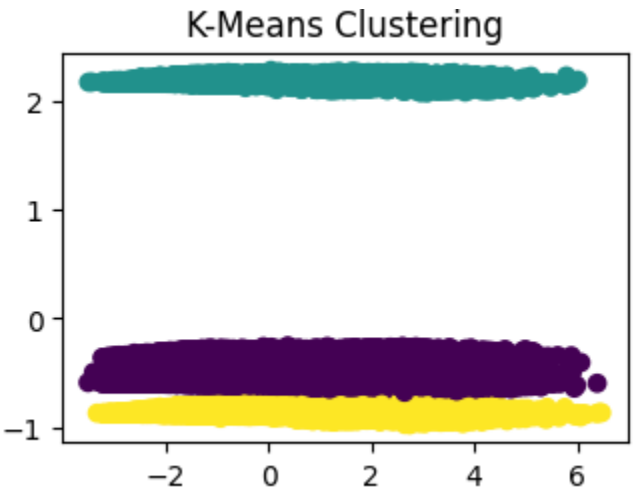
```
label_encoder = LabelEncoder()
df['gender_encoded'] = label_encoder.fit_transform(df['gender'])
df['smoking_history_encoded'] = label_encoder.fit_transform(df['smoking_history'])

features = ['age', 'race:AfricanAmerican', 'race:Asian', 'race:Caucasian', 'race:Hispanic',
            'race:Other', 'hypertension', 'heart_disease', 'smoking_history_encoded', 'bmi',
            'hbA1c_level', 'blood_glucose_level']

scaler = StandardScaler()
df_scaled = scaler.fit_transform(df[features])
```

4. **K Means** :-K-Means Clustering: A centroid-based clustering algorithm that partitions data into k clusters by minimizing the distance between data points and their respective cluster centers. It works well with well-separated clusters but assumes a spherical shape.

```
kmeans = KMeans(n_clusters=3, random_state=42)
df['kmeans_cluster'] = kmeans.fit_predict(df_scaled)

pca = PCA(n_components=2)
df_pca = pca.fit_transform(df_scaled)

plt.subplot(2, 2, 1)
plt.scatter(df_pca[:, 0], df_pca[:, 1], c=df['kmeans_cluster'], cmap='viridis')
plt.title("K-Means Clustering")

plt.tight_layout()
plt.show()
```

This code performs K-Means clustering on a dataset with 3 clusters, assigns cluster labels, and visualizes the results using PCA (Principal Component Analysis) for dimensionality reduction.

- **K-Means Clustering:** It fits the scaled dataset and assigns each data point a cluster label (0, 1, or 2), which is stored in df['kmeans_cluster'].
- **PCA Transformation:** The dataset is reduced to 2D for visualization.
- **Plotting:** A scatter plot is created using the first two PCA components, with points colored based on their assigned clusters.



- Successfully identified three distinct clusters.
- The clusters are horizontally aligned, suggesting that one or two dominant features are driving the segmentation.
- Works well for structured, well-separated data but may struggle with complex patterns.

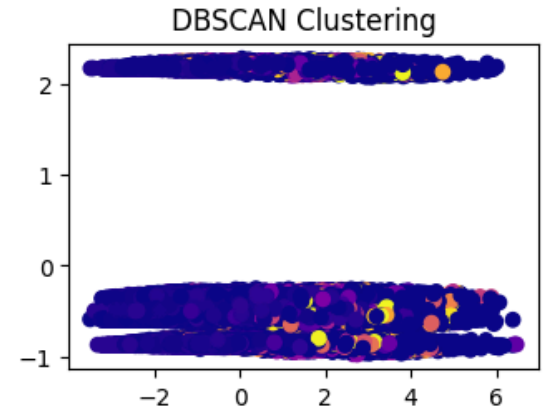
5. **DBSCAN Clustering:-**DBSCAN (Density-Based Spatial Clustering of Applications with Noise): A clustering algorithm that groups data points based on density rather than predefined clusters. It can detect arbitrarily shaped clusters and outliers but is sensitive to parameter selection (eps, min_samples).

```
dbscan = DBSCAN(eps=0.5, min_samples=2)
df['dbscan_cluster'] = dbscan.fit_predict(df_scaled)

plt.subplot(2, 2, 2)
plt.scatter(df_pca[:, 0], df_pca[:, 1], c=df['dbscan_cluster'], cmap='plasma')
plt.title("DBSCAN Clustering")

plt.tight_layout()
plt.show()
```

This code applies DBSCAN clustering on a dataset and visualizes the results using PCA for dimensionality reduction.

Steps:

1. **DBSCAN Clustering:**
   - DBSCAN(eps=0.5, min_samples=2): Groups points based on density.
   - Noise points (outliers) are labeled as -1.
   - Cluster labels are stored in df['dbscan_cluster'].
2. **PCA for Visualization:**
   - Reduces data to 2D for plotting.
3. **Scatter Plot:**
   - Points are colored by their DBSCAN cluster labels using the 'plasma' colormap.
   - Outliers appear as separate scattered points.

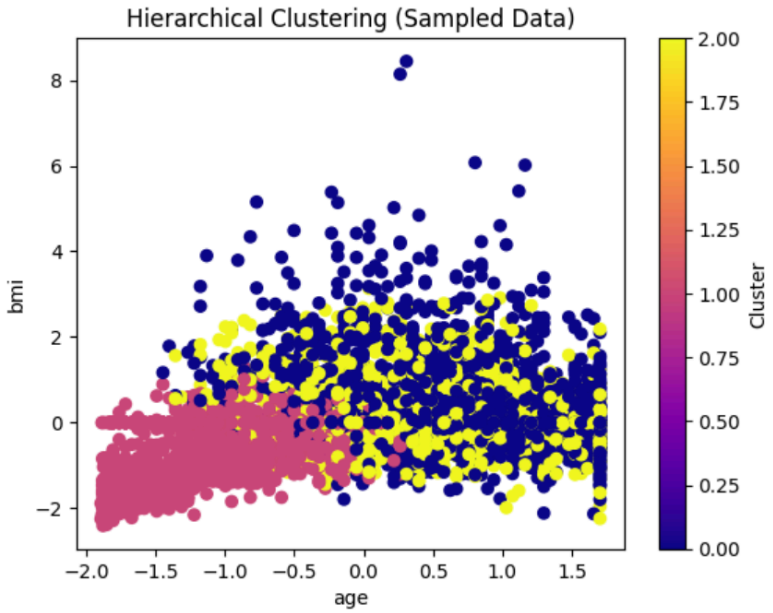


- Identified dense regions, but the separation is not as clear as K-Means.
- Some points were classified as noise (outliers).
- Suitable for non-linear and arbitrarily shaped clusters, but parameter tuning (eps, min_samples) is crucial for better results.

6. Hierarchical clustering

```
# Take a random sample of 5000 to reduce memory usage
sample_size = 5000
df_sample = df.sample(n=sample_size, random_state=42)
data_sample = df_sample[numeric_features]

# Standardize sample data
scaler = StandardScaler()
data_sample_scaled = scaler.fit_transform(data_sample)

# Apply Hierarchical Clustering on the sample
hc = AgglomerativeClustering(n_clusters=3, linkage='ward')
labels = hc.fit_predict(data_sample_scaled)  # Store labels separately

df_sample['hc_cluster'] = labels  # Assign cluster labels to the sample

# Scatter plot of clusters
plt.scatter(data_sample_scaled[:, 0], data_sample_scaled[:, 1], c=df_sample['hc_cluster'], cmap='plasma')
plt.title("Hierarchical Clustering (Sampled Data)")
plt.xlabel(numeric_features[0])
plt.ylabel(numeric_features[1])
plt.colorbar(label='Cluster')
plt.show()
```

1. **Data Preprocessing:**
   - Selected relevant numerical features (age, bmi, hbA1c_level, blood_glucose_level).
   - Took a random sample of 5000 from the dataset to reduce memory usage.
   - Scaled the data using StandardScaler() to normalize feature values.
2. **Applying Hierarchical Clustering:**
   - Used AgglomerativeClustering(n_clusters=3, linkage='ward') to perform clustering on the sampled data.
   - Assigned the resulting cluster labels back to the sampled dataset.
3. **Visualization:**
   - Scatter plot of age vs. bmi, colored based on cluster assignments.
   - Color bar represents the cluster labels.



Age vs BMI

- The three colors represent different clusters found by hierarchical clustering.
- The clustering algorithm separates the data into groups, possibly based on trends in age and bmi.
- The pink cluster (bottom left) seems to contain younger individuals with lower BMI.
- The yellow and dark blue clusters show mixed distributions but may have patterns related to BMI and age distribution.

Conclusion:

- K-Means performed better in this case, successfully grouping data into well-defined clusters.
- DBSCAN struggled with clear separation and produced noisy clusters, likely due to dataset structure or parameter settings.
- The choice of clustering algorithm depends on data distribution – K-Means is ideal for structured clusters, while DBSCAN is useful for detecting complex patterns and outliers.
- Hierarchical clustering successfully segmented the data into three meaningful clusters based on age and bmi. One cluster represents **younger individuals with lower BMI**, while the other two capture variations in **BMI and age**. The clustering reveals potential health patterns, but further validation (e.g., silhouette scores) and additional features could improve accuracy.

Experiment 8

Aim: To implement a recommendation system on your dataset using the following machine learning techniques: Regression, Classification, Clustering, Decision tree, Anomaly detection, Dimensionality Reduction, Ensemble Methods

Theory:

1. Data Preprocessing

- **Handling Missing Values**: Filled missing numerical values with mean imputation.
- **Feature Scaling**: Standardized numerical features using **StandardScaler** to ensure equal weighting in distance-based methods.

2. Clustering with K-Means

- Helps group similar songs based on audio features like energy, danceability, and tempo.
- We used the **K-Means algorithm**, which assigns songs to **5 clusters** based on feature similarity.
- **Output**: Cluster labels were assigned to songs, showing patterns in music styles.

3. Cosine Similarity for Recommendation

- Measures how similar two songs are based on their feature vectors.
- Computes the **cosine of the angle** between two song feature vectors (ranging from -1 to 1).
- **Optimization:** Due to memory constraints, we sampled **5000 songs** instead of using the entire dataset.

4. Recommendation Algorithm

- Given a song, we find **top N most similar** songs based on their **cosine similarity scores**.
- Returns song names, artists, and popularity.

Implementation:

Loading the dataset

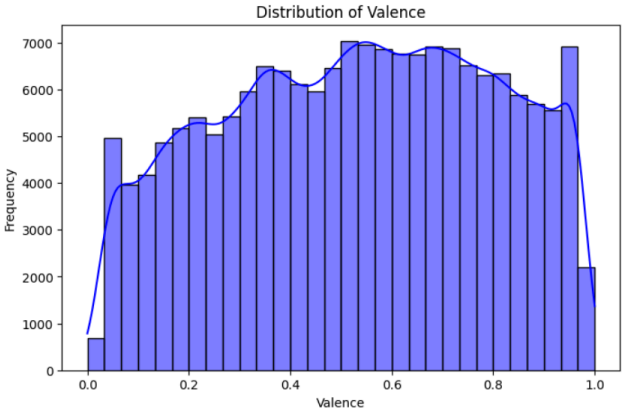
```
import pandas as pd
# Load the dataset
df = pd.read_csv("data.csv")
# Display first few rows
df.head()
```

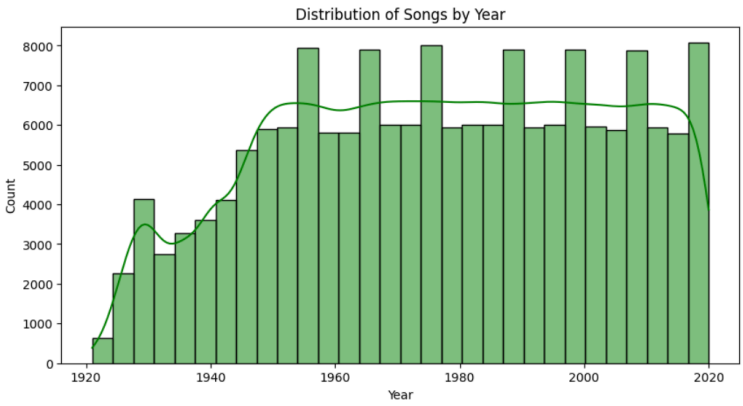
	valence	year	acousticness	artists	danceability	duration_ms	energy	explicit	id	instrumentalness	key	liveness
0	0.0594	1921	0.982	['Sergei Rachmaninoff', 'James Levine', 'Berli...	0.279	831667	0.211	0	4BJqT0PrAfrxzMOxytFOIz	0.878000	10	0.665
1	0.9630	1921	0.732	['Dennis Day']	0.819	180533	0.341	0	7xPhfUan2yNtyFG0cUWkt8	0.000000	7	0.160
2	0.0394	1921	0.961	['KHP Kridhamardawa Karaton Ngayogyakarta Hadi...	0.328	500062	0.166	0	1o6I8BglA6ylDMrIELygv1	0.913000	3	0.101

```
# Check for missing values
df.isnull().sum()
# Summary statistics
df.describe()
# Check data types
df.dtypes
```

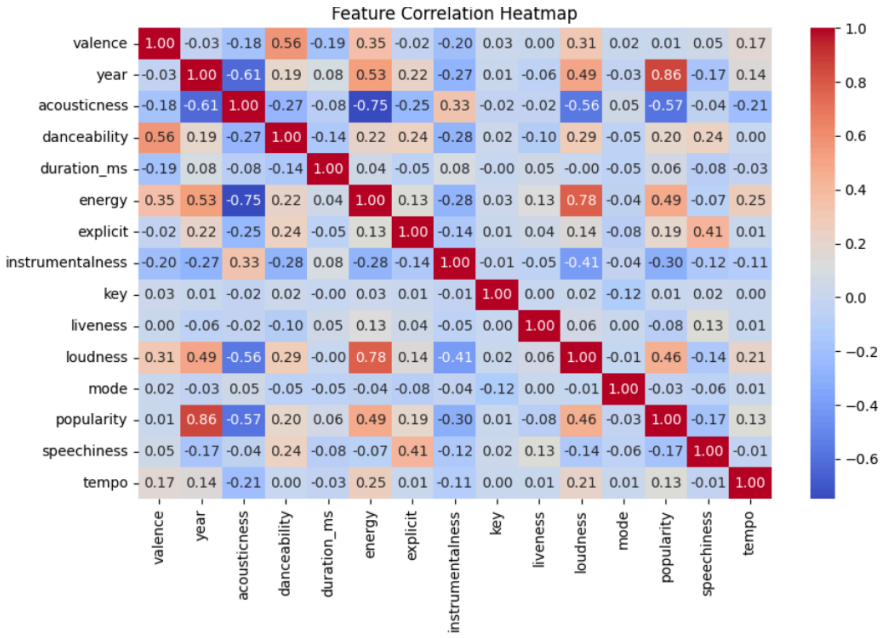
	0
valence	float64
year	int64
acousticness	float64
artists	object
danceability	float64
duration_ms	int64
energy	float64
explicit	int64

Visualisation





Heatmap



K Means Clustering

```
# Clustering using KMeans
kmeans = KMeans(n_clusters=5, random_state=42, n_init=10)
df["cluster"] = kmeans.fit_predict(scaled)

# Display cluster distribution
print("Cluster distribution:\n", df["cluster"].value_counts())
```

```
Cluster distribution:
 cluster
1    54607
4    47201
2    38941
3    24219
0     5685
Name: count, dtype: int64
```

- The dataset has been divided into 5 clusters (0 to 4).
- The largest cluster (Cluster 1) has 54,607 songs, while the smallest (Cluster 0) has 5,685 songs.
- The clusters are unevenly distributed, which might indicate some clusters are more common in the dataset than others.

Recommendation System

```
sample_df = df.sample(n=5000, random_state=42).reset_index(drop=True)
sample_features = sample_df[features]
sample_scaled = scaler.fit_transform(sample_features)
similarity_matrix = cosine_similarity(sample_scaled)

# Recommend similar songs based on index in the sample
def recommend_similar(song_index, num_recommendations=5):
    sim_scores = list(enumerate(similarity_matrix[song_index]))
    sim_scores = sorted(sim_scores, key=lambda x: x[1], reverse=True)
    sim_scores = sim_scores[1:num_recommendations + 1]  # skip the song itself
    similar_indices = [i[0] for i in sim_scores]
    return sample_df.iloc[similar_indices][["name", "artists", "popularity"]]

# Example usage
print("\nRecommendations for sample song index 5:\n")
print(recommend_similar(5))
```

```
Recommendations for sample song index 0:

                                  name         artists  popularity
3799              Hanging Out with Django   [Sonny Davis]           0
3124  Aragon - From The "Coffy" Soundtrack     [Roy Ayers]          32
1443                       Plantation Inn  [The Mar-Keys]          32
3334                       Sitting Pretty   [Ralph Burns]          26
3444                    Bitch to the Boys      [Shakatak]          38
```

```
Recommendations for sample song index 5:

                        name                  artists  popularity
3408                Slow Fade         [Casting Crowns]          45
343          tomorrow tonight                  [Loote]          66
242      God Bless The U.S.A.          [Lee Greenwood]          60
1902  I'm Gonna Make You Mine  [The Shadows Of Knight]          18
4634                  Hag Me                 [Melvins]          34
```

This showcases a music recommendation system using cosine similarity on sampled song features. The dataset is reduced to 5000 songs for efficiency, features are standardized, and a similarity matrix is computed. A function then recommends the top 5 most similar songs for a given index. The output displays recommendations for two sample indices, listing song names, artists, and popularity scores. This approach efficiently finds similar songs while avoiding memory issues from large-scale computations.

Conclusion

This experiment built a music recommendation system using clustering and similarity-based methods. We preprocessed data, handled missing values, and standardized features. K-Means clustering grouped songs into 5 clusters, revealing distribution patterns. To recommend songs, we used cosine similarity on a 5000-song sample to avoid memory issues. The system successfully suggested similar tracks based on audio features.

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?**

    - Apache Spark is an open-source, distributed computing system designed for large-scale data processing.
    - It provides an interface for programming entire clusters with implicit data parallelism and fault tolerance.
    - Spark supports in-memory computing, making it faster than traditional MapReduce jobs in Hadoop.
    - It operates on distributed data and processes them in parallel across multiple nodes, enabling high-speed processing of large datasets.

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

    - **Step 1: Data Loading** – Data is loaded into a Spark DataFrame from various sources such as CSV, JSON, Parquet, etc.
    - **Step 2: Data Cleaning** – This step involves handling missing data, correcting data types, and dealing with inconsistencies.
    - **Step 3: Data Transformation** – Data transformations such as filtering, aggregating, and summarizing are performed to understand patterns in the data.
    - **Step 4: Data Visualization** – Though Spark does not natively support visualizations, it can be integrated with libraries like Matplotlib, Seaborn (via Pandas), or other visualization tools to generate plots and charts.
    - **Step 5: Statistical Analysis** – Summary statistics (e.g., mean, median, standard deviation) and other metrics are computed to understand data distributions and relationships.

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,

data scientists can uncover insights, clean data, and prepare datasets for further analysis or modeling.

Experiment No: 10

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

Theory:

1. **What is streaming? Explain batch and stream data.**

    - **Streaming** refers to the continuous flow of real-time data. It is the process of transmitting data in a steady, ongoing manner as it becomes available.
    - **Batch Data** involves data that is collected and processed in large chunks at scheduled intervals. The data is accumulated over a period, and then processed together as a batch.
    - **Stream Data** (or real-time data) is generated continuously and processed immediately as it arrives. This type of data is processed in small increments, often on a per-event basis, and is typically used in scenarios like sensor data, logs, or online transactions.

2. **How data streaming takes place using Apache Spark?**

    - Apache Spark performs real-time data processing using **Spark Streaming**, an extension of the core Spark API.
    - Data is ingested in small chunks called **micro-batches**. These micro-batches are processed as streams in near real-time.
    - Data sources for streaming can include message queues like Apache Kafka, socket connections, or file systems.
    - Spark Streaming processes data in micro-batches by continuously polling the source for new data and processing it in small time intervals (e.g., seconds).
    - Spark integrates batch and stream processing, meaning you can process streaming data with the same APIs used for batch data, allowing for unified processing pipelines.

Conclusion:

Apache Spark provides a powerful framework for both batch and stream data analysis. By utilizing Spark Streaming, organizations can process and analyze real-time data efficiently, while also leveraging batch processing for large-scale historical data analysis. The integration of these two processing models allows Spark to cater to a wide range of data processing needs, from

time-sensitive streaming applications to periodic batch processing, making it a versatile tool for modern data engineering tasks.

Name: Sadneya Samant
Roll No: 46
class: D15C

05/08

AIDS Assignment

Q.1. What is AI? considering the COVID-19 pandemic situation, how did AI help to survive and renovate our way of life with different applications?

→ 1) Artificial Intelligence is the replication of human intelligence in machines, enabling them to learn, reason, solve problems, and makes decisions without direct human interaction

2) AI played a vital role in managing the COVID-19 crisis by detecting outbreaks early through data analysis, accelerating drug and vaccine development, automating medical diagnoses with imaging and predictive models, supporting telemedicine via AI chatbots, optimizing supply chains for essential goods and enhancing remote work and education through AI-driven automation, making daily life more adaptable and efficient.

Q.2. What are AI agent terminology? Explain with examples.

→ AI agents are systems that perceive their environment, process information and take actions to achieve specific goals. key terminologies include:

1) Agent - An entity that interacts with environment. (eg. self-driving car).

2) Environment - The external system in which the agent operations (eg. traffic conditions for a self-driving car)

3) Perception - Data collection from sensors (eg. traffic conditions for a self-driving car)

4) Actuators - Components that execute actions (eg. robotic arms in manufacturing)

5) Rationality - The ability to make optimal decisions based on available information (eg. AI trading bots adjusting stock investments)

6) Autonomy - The degree of independence an agent has (eg. voice assistants like Alexa operating without human input)

7) Types of agents:

1) Simple reflex agents: Act based on current perceptions (eg. Thermostats)

2) model-based agents: use internal models to predict future states. (eg. GPS navigation system)

3) Goal-based Agents: Take actions to achieve a specific goal. (eg. chess-playing AI).

4) Utility-Based agents - optimize outcomes for maximum benefit (eg. recommendation systems in e-commerce.)

5) Learning agents - Improve performance over time using past experiences (eg. self-learning robots in warehouses.)

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

→ 1) In 8-puzzle technique it consist of 3x3 sized cube. In initial state it contains all number tiles present randomly.

2) Then Here, our aim is manage them in increasing order from 1 to 8. by moving tiles with the help of blank space.

3) Common AI techniques include:

i) Uninformed search:

1) Here by using BFS we can explore all possible moves level by level but it is slow. it guarantess the shortest path.

2) Then by using DFS we can explore a path deeply before backtracking. it may not always find optimal solution.

3) Then by using IDDFS which combines DFS and BFS to find the shortest solution efficiently.

ii) Informed search Techniques (Heuristic search)

1) Uses a heuristic function $f(n) = g(n) + h(n)$, where:
$g(n)$ is the cost to reach the current state.
$h(n)$ is the estimate cost to reach the goal.
2) The by using Greedy Best-first search we can find optimal solution which can be fast but not always optimal.
It uses heuristic function $h(n)$
Here A* algorithm is the most efficient and widely used technique for solving 8-puzzle problem.

Q.4. What is PEAS descriptor?
PEAS (Performance measure, Environment, Actuators, Sensors) is a framework used to define the components of an AI agent by specifying how it interacts with its environment and evaluates its success.
PEAS description for given AI agents:
1) Taxi driver:
1. Performance Measure: Safety, speed, customer satisfaction, fuel efficiency.
2. Environment: Roads, traffic, pedestrians, weather conditions.
3. Actuators: steering, acceleration, brakes, turn signals.
4. Sensors: GPS, cameras, speedometer, LIDAR, fuel gauge.
2) Medical diagnosis system:
1. Performance Measure: Accuracy of diagnosis, patient recovery rate, response time.
2. Environment: Patient data, medical records, symptoms, lab reports.
3. Actuators: Display diagnosis, prescribe medication, recommend tests
4. Sensors: Patient input, test result, doctor's notes, medical imaging.
3) A music composer:

1. Performance measure: musical quality, originality, listner preference.
2. Environment: musical genres, user preferences, historical compositions.
3. Actuators: generating notes, melodies, instrument selection.
4. sensors: user feedback, music databases, emotional tone analysis.

4) An aircraft autolander
1. performance measure: safe landing, smooth touchdown, passenger comfort.
2. Environment: weather conditions, runway, air traffic, altitude
3. Actuators: flaps, landing gear, brakes, enging thrust control.
4. sensors: Altimeter, GPS, wind speed sensors, radar, cameras

5) An Essay Evaluator
1. Performance Measure: Grammer accuracy, coherence, relevance, readibility.
2. Environment: Essays, writing rules, predefined grading criteria.
3. Actuator: score assignment, grammer suggestions, feedback generation.
4. Sensors: Text input, grammer and plagiarism checkers, semantic analyzers.

6) A robotic sentry gun for the keck lab
1. Performance Measure. Accuracy, target identification, security effectiveness.
2. Environment: Lab perimeter, potential threats, lighting conditions
3. Actuators: Rotating turret, firing mechanism, alarm system.
4. Sensors: Motion detectors, infrared sensors, cameras, radar

Q.5. Categorize a shoping bot for an offline bookstore according to each of six dimensions (fully / partially observable, deterministic / stochastic, episodic / sequential, static / dynamic, discrete / continuous, single / multi agent).

→ 1) Partially observable: as bot may not have full knowledge of stock availability, customer preferences, or real-time changes in the bookstore.

2) Stochastic: Book availability, customer behaviour and price changes introduce randomness, making environment unpredictable.

3) Sequential: Each decision affects future interactions.

4) Dynamic: The environment can change as books sell out, customers move or bookstore policies update.

5) Discrete: The bot operates with distinct choices, such as recommending a book, checking stock or processing a purchase.

6) Multi-agent: as bot interacts with customers, bookstore staff and possibly other AI systems.

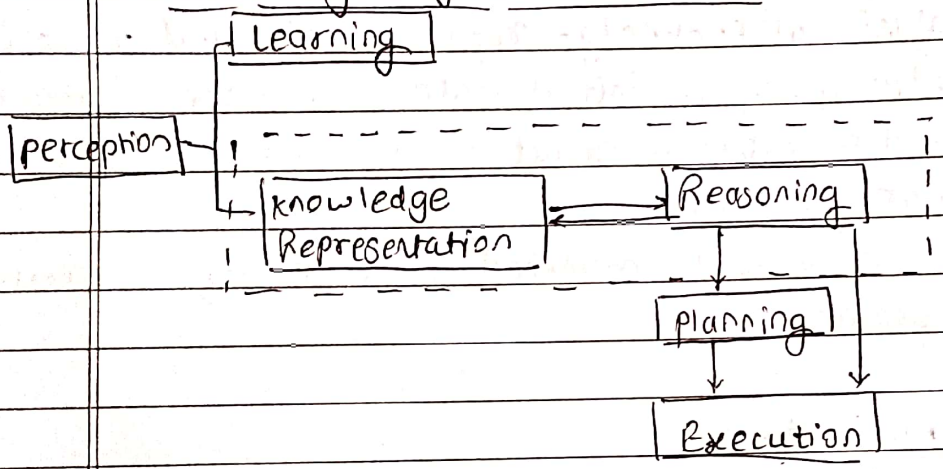
Q.6. Model based vs utility based agent.

	Model based	Utility-based
1)	Used internal state to make decisions	chose actions based on maximizing a utility function
2)	predicts future states using the model before acting.	Evaluates multiple possible actions and selects best based on utility.
3)	It is goal oriented but focuses on how the environment works	It is also goal-oriented but it aims to achieve the best possible outcome.
4)	It may not always take the	Always selects the action that

best possible action, only takes a feasible one.	provides the highest benefit.
eg. self-driving car to predict congestion	eg. stock trading AI selecting the trade with the highest expected profit.

Q.7. Explain the architecture of knowledge based agent and learning agent.

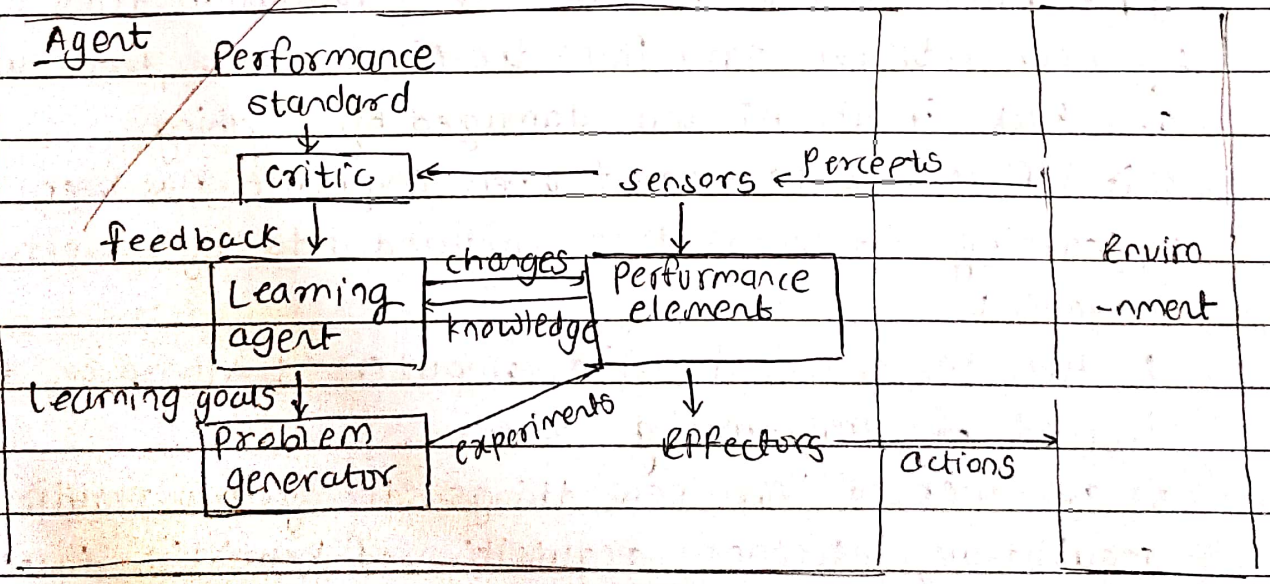
→ Knowledge agent Architecture



1] perception : It retreives data or information from the environment
2) you can retrieve data from the environment, find out noises and check if the AI was damaged by anything.
3) it defines how to respond when any sense has been detected.
2] learning : It learns from captured data by the perception component.
1) Here the goal is to build computers that can be taught instead of programming them.
2) In order to learn new things, the system require knowledge acquisition, inference, acquisition of heuristics, faster searches et
3] knowledge representation & reasoning : 1) The main component in the cycle is knowledge representation and reasoning which

shows human-like intelligence in the machines.
2) Knowledge representation is all about understanding intelligence.
3) Goal is to understand and build intelligent behavior from the top-down and focus on what an agent needs to know in order to behave intelligently.
4) It defines how automated reasoning procedures can make this knowledge available as needed.
4] Planning and Execution: 1) The planning and execution components depend on analysis of knowledge representation and reasoning.
2) planning includes giving an initial state, finding their preconditions and effects, and a sequence of actions to achieve a state in which particular goal holds.
3) Now once the planning is completed, the final stage is execution of the entire process.

Learning Agent Architecture



A learning agent improves its performance over time by learning from past experiences. It has four components:

1) Learning element: learns from interactions and updates knowledge.

2) Performance element: uses the learned knowledge to make decisions.

3) critic: evaluates agents performence by comparing actions with desired outcome.

4) problem generator: suggests new explanatory actions to improve learning.

Q.9 convert the following to predicates:

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

b) Bus goes via Andheri and Goregaon.

c) car has puncture so is not available. will anita travel via goregaon? use forward reasoning.

→ 1) Anita travels by car if available otherwise by bus.

so → Available (car) ⇒ Travels (Anita, car)

~Available (car) ⇒ Travels (Anita, Bus)

2) Bus goes via Andheri and Goregaon.

Route (Bus, Andheri)

Route (Bus, Goregaon)

3) car has a puncture, so it is not available.

Puncture (car) ⇒ ~Available (car)

given: Puncture (car)

therefore, ~Available (car)

forward reasoning:

1) from (3) ~Available (car)

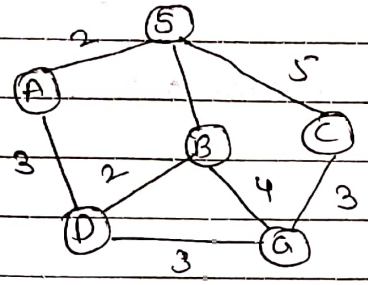
2) from (1), ~Available (car) ⇒ Travels (Anita, Bus)

so Anita must travel by bus.

3) From (2), Route(Bus, Goregan) meaning bus travels via goregaon.

4) since Anita travels by Bus and the Bus goes via Goregaon, Anita will travel via Goregaon.

Q.10. Find the route from S to G using BFS.



step 1: start with node S
mark it visited. [S]
eneque S into the queue

step 2: dequeue S and explores its neighbors A, B, C.
mark A, B, C visited.
enqueue A, B, C into the queue. [A|B|C]

step 3: deque A and explore its neighbours D.
mark D visited
enqueue D into the queue [A|B|C|D] ∴ S → A

step 4: deque B and explore its neighbours
mark G visited. [C|D|G] ∴ S → A → B
~~Thus goal is reached.~~
~~∴ shortest path found will S → C → G~~

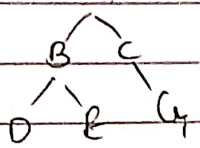
step 5: deque C and explore its neighbours
it has only 1 neighbour G which is already visited. ∴ S → A → B → C

step 6: deque D and explore its neighbours
it has only 1 neighbour G which is already marked visited
∴ S → A → B → C → D

step 7: deque G. thus goal state reached.
∴ S → A → B → C → D → G

Q.11. What do you mean by depth limited search ? Explain iterative deepning search with examples.

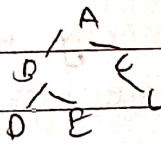
→ 1) Depth - limited search is a variation of Depth-first search (DFS) where the search is restricted to a specific depth limit L.
2) if goal is not found within this limit, the search terminates.
3) it prevents infinite loops in deep or infinite state spaces but risks failing to find solution if L is too low.

eg. 

Here if L=1 and we need to find G then first expand A → [B, C] at L=1 thus goal not found so we need to increase the limit to 2 to provide solution. this is main drawback.
~~then for L=2 expand B~~

Iterative Deepning search (IDS).

1) It combines the space efficiency of DFS and completeness of BFS by repeatedly running DLS with increasing depth limits L = 0, 1, 2, .. until goal is found

eg. 

1) DLS with L=0 only A is prevent thus goal not found
2) increase L = 1
3) DLS with L=1 Expand A, B and C found
4) again increase L = 2
5) DLS with L=2, Here Expand B gets D & E.
Expand C gives G thus goal found at depth = 2 (ie L = 2)

2) thus it guarantees finding the shortest path while using less memory than BFS.

Q.12. Explore hill climbing and its drawbacks in detail with example. also state limitations of steepest - ascent hill climbing.

→ 1) Hill climbing is a local search algorithm that continuously moves toward the best neighbouring state with a higher

heuristic value, aiming to reach a global optimum.

2) steps: 1) start from initial state.

2) evaluate neighbouring states and move to the one with highest heuristic value.

3) repeat until no better neighbour exists (local maximum).

eg. Imagine a mountain climbing scenario where a hiker moves uphill based on the steepest slope. If they reach a peak that is not the highest (global maximum), they might get stuck.

4) drawbacks of hill climbing:

i) local maxima: 1) the algorithm may stop at a peak that is not the global optimum.

2) eg. a small hill before taller mountain.

ii) Plateau problem: 1) A flat region where all neighbouring states have the same heuristic value, causing the search to halt.

2) eg. A long flat road with no clues on where to go.

iii) Ridges: The algorithm may fail to climb diagonally when only direct moves are considered. eg. climbing a staircase with missing steps.

iv) No backtracking: once it moves forward, it can't recover from a bad decision. eg. choose the wrong path in maze with no way back.

5) solution to improve hill climbing:

1) Random Restarts: start from different points.

2) Simulated Annealing: Occasionally accept worse moves to escape local optima.

Q.13. Explain simulated annealing and write its algorithm.

→ 1) simulated Annealing is an optimization algorithm designed to search for an optimal or near-optimal solution in a large solution space.

2) The name and concept are derived from the process of annealing in metallurgy, where a material is heated and then slowly cooled to remove defects and acheive a stable crystalline structure.

3) In simulated annealing, the "heat" corresponds to the degree of randomness in the search process, which decreases over time (cooling schedule) to refine the solution.

4) The method is widely used in combinational optimization, where problems often have numerous local optima that standard techniques like gradient descent might get stuck in.

5) Algorithm:

1) The algorithm starts with an initial solution and a high temperature, which gradually decreases over time. Here's step by step of how the algorithm works:

1] initialization: Begin with initial solution S_0 and initial temperature T_0. the temperature controls how likely the algorithm is to accept worse solutions as it explores the search space.

2] Neighbourhood search: At each step, a new solution s' is generated by making a small change (or perturbation) to the current solution s.

3] objective function evaluation: The new solution s' is evaluated using the objective function. If s' provides a better solution than s, it is accepted as the new solution.

4) Acceptance Probability: If s' worse than s, it may still be accepted with a probability based on temperature and the difference in objective function values. The acceptance probability is given by: $P(accept) = e^{-\Delta R/T}$

5) cooling schedule: After each iteration, the temperature is decreased according to a predefined cooling schedule, which determines how quickly the algorithm converges. Common cooling schedules include linear, exponential or logarithmic cooling.

6) Termination: The algorithm continues until the system reaches a low temperature or a predetermined no. of iterations is reached.

Q.14. Explain A* algorithm with an example.

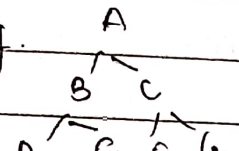
→ 1) The A* algorithm is an informed search algorithm that finds the optimal path by considering the both cost to reach a node (g(n)) and estimated cost to the goal (h(n)), using the formula: $f(n) = g(n) + h(n)$

where, g(n) = actual cost from start node to n.
h(n) = estimated cost from n to goal.

2) Algorithm:

1) Initialize the open list with start node.
2) loop until goal is found.
   1) pick the node with lowest f(n).
   2) if its the goal node, return the path.
   3) expand its neighbors and calculate f(n).
   4) Add unvisited neighbors to the open list.
3) continue until open list is empty.

3) eg.



Cost & Heuristic values

Node	(path cost) g(n)	(heuristic) h(n)	f(n)
A	0	6	6
B	1	4	5
C	2	3	5
D	3	7	10
E	4	5	9
F	3	2	5
G	4	0	4 (Goal)

start at A, calculate $f(A) = 6$

Expand $A \rightarrow \{B, C\}$ (both have $f=5$)

Expand $C \rightarrow \{F, G\}$ (G have lowest f)

goal reached at G

∴ final path $A \rightarrow C \rightarrow G$

4) Advantages:

1) It is flexible hence used in maze solving, AI, robotics.
2) It is efficient as explores fewer nodes than BFS/DFS.
3) It provide guaranteed optimality.

5) Disadvantages:

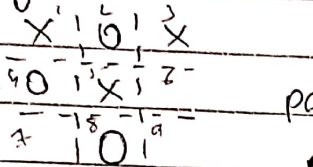
1) It requires more memory than DFS.
2) A poor heuristic can make A* inefficient.

Q. 15. Explain minimax algorithm and draw game for Tic-tac-toe.

1) The minimax Algorithm is used in adversial search to determine the optimal move by assuming both players play optimally.
2) Maximizer: (eg X) tries to get highest score.
   Minimizer: (eg O) tries to reduce the score.
3) Algorithm:
i) generate the game tree up to a depth limit.

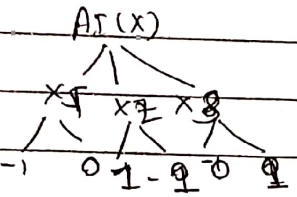
2] Assign heuristic values to leaf nodes.
3] Backpropogate values:
1) maximizer picks the maximum value:
2) minimizer picks the minimum value.
4] The root node selects the best move based on propogated values.

eg. 1) consider a Tic-Tac-Toe board where its AI's turn (x) to play:



so here possible moves can be done at position (2,3), (3,1), (3,3).

2) The tree expands, considering all possible outcomes.



3) The AI (max) picks the more leading to least possible score).
4) If multiple moves have the same values, additional heuristics (like the earliest win) may be used.

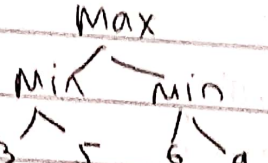
Q.16. Explain Alph-beta pruning algorithm for adversial search with an example.

→ 1) Alpha-beta pruning optimizes minimax by skipping unnecessary branches, reducing computations. it uses two values:
Alph(α): Best score maximizer can achieve.
Beta(β): Best score minimizer can achieve.
2) Algorithm:
1) perform minimax search.
2) Prune branches where:
1) if Maximizer's best(α) ≥ Minimizer's best(β), further evaluation is skipped.
2) if minimizer's best(β) ≤ Maximizer's best(α), further

evaluation is skipped.

3) This reduces time complexity without affecting final decision.

3) eg. let's consider a simple game tree with a depth of 2.



with α-β pruning,

~~left subtree evaluates~~

1) start at the root (MAX node)
$\alpha = -\infty$ $\beta = +\infty$

2) move to the first MIN node (left subtree).

3) evaluate the first child of MIN (3)

4) Thus MIN updates $\beta = 3$ as it is the smallest value seen so far)

5) Now move to second child of MIN(5).

$\therefore \beta = \min(5, 3) = 3$

6) Move back to MAX. MAX updates $\alpha = \max(-\infty, 3) = 3$

7) Now move to second MIN node (right subtree)

8) Now $\alpha = 3$ (best MAX value so far) and $\beta = +\infty$

9) After evaluating first child of MIN (6) β gets equal to 6.

10) Now check pruning condition before evaluating 9.

$\therefore$ MAX has $\alpha = 3$, MIN has $\beta = 6$

Since $\beta(6) > \alpha(3)$ we ~~stop~~ evaluating further in this branch.

11) $\therefore$ node with value 9 is pruned

Q.17. Explain wumpus world environment giving its PEAS description. Explain how percept sequence is generated?

→ 1) The wumpus world is grid-based AI environment where an agent explores a cave while avoiding hazards like pits and the wumpus monster.

2) PEAS descriptor:

1) Performance Measure: Reaching the gold safely, minimizing steps.

2) Environment: A 4×4 grid with pits, gold and wumpus.

3) Actuators: move forward, turn, grab, shoot, climb.
4) sensors: perceive stench (wumpus nearby), breeze (pit nearby), glitter (gold found).
5) percept sequence generation:
1) Agent starts at (1,1), sensing its surroundings.
2) if stench is detected, the wumpus is nearby.
3) if breeze is detected, a pit is nearby.
4) The agent infers safe paths and navigates toward the gold while avoiding hazards.

Q.18. solve the following crypto-arithmetic problem SEND + MORE = MONEY.

```
  S E N D
+ M O R E
---------
M O N E Y
```

1) Here the input is of 4 bit and output is of 5 bit. therefore, there must be carry
$\therefore M = 1$

2) assume $S = 9$

$$\therefore \begin{array}{r} S \\ +M \\ \hline MO \end{array} \Rightarrow \begin{array}{r} 9 \\ +1 \\ \hline 10 \end{array}$$ therefor we get $O = 0$

3) then $E + 0 = N$ but $0 = 0$ (zero)
$\therefore E = N$ but it is not possible, therefore, there must be a carry present over there.

4) $\therefore E + 1 = N$ the $N + R = E$

5) assume $N = 8$ $R = 7$ $\therefore \begin{array}{r} N \\ +R \\ \hline \end{array} \begin{array}{r} 8 \\ +7 \\ \hline 15 \end{array}$ with 1 carry
but it is not possible
therefore assume $N = 7$ $R = 6$ $N + R = 7 + 6 = 13$ $\therefore R = 3$
but $E + 1 = 3 + 1 = 4 \neq N$

6) so again assume $N = 6$ $R = 8$ $\therefore \begin{array}{r} N \\ +R \\ \hline \end{array} \Rightarrow \begin{array}{r} 6 \\ 8 \\ \hline 14 \end{array}$
$\therefore E = 4 + 1 = 5$ (add a carry)

7) Now check $\begin{array}{r} N \\ +R \\ \hline \end{array} \begin{array}{r} 8 \\ 17 \\ \hline 15 \end{array}$ $\therefore E = 5$ $N = 6$ $R = 8$

0 1 2 3 4 5 6 7 8 9

8) $\begin{array}{r} D \\ +E \\ \hline Y \end{array}$ Here we have value of E=5

Now assume D=7 ∴ D+R = 7+5 = 12

∴ Y = 2

∴ S=9, E=5, Y=2, D=12, N=6, M=1, O=0, R=8,

Q.19. Consider the following axioms:

→ 1) Represent these axioms in ~~first order predicate logic (FOPL)~~:

All people who are graduating are happy.

All people who are happy are smiling.

Someone is graduating.

explain the following:

1) Represent these axioms in first order predicate logic.

→ i) $\forall x$ (Grandmother(x) → Happy(x))

2) $\forall x$ (Happy(x) → Smiling(x))

3) $\exists x$ (Graduating(x))

2) convert each formula to clause form.

1) ~Graduating(x) ∨ Happy(x)

2) ~Happy(x) ∨ Smiling(x)

3) Graduation(A) ← assuming A is the individual who is graduating

3) prove that "is someone smiling?" using simulation (resolution technique:

From (3): Graduating(A)

using 1: Happy(A)

using 2: Smiling(A)

thus, someone is smiling.

4) Resolution tree can be given as:

1) Graduating(A) given

2) Graduating(A) → Happy(A), → Happy(A)

3) Happy(A) → smiling(A) ~ smiling(A)

Q20. Explain Modus Ponens with suitable example.

→ 1) Modus Ponens (law of detachment) is a rule of interference stating:
P→Q, P ⇒Q

2) eg. "if it rains, the ground gets wet. (premises: rain → wet around)
2) it is raining. (premise: Rain)
3) conclusion: the ground is wet.

Q.21. Explain forward chaining and backward chaining algorithm with an example.

→ 1) forward chaining: 1) data-driven interference: starts from known facts and applies rules to reach a goal.
2) used in expert system (eg. medical diagnosis)
3) eg. 1) fact: "sore throat".
2) Rule: "If some throat → infection"
3) new fact: "Infection"
4) Rule "If infection → need antibiotics"
5) conclusion: "Need antibiotics".

2) backward chaining: 1) goal-driven inference: starts from goal and works backward to find supporting facts.
2) used in AI reasoning and theorem proving
3) eg. goal: Does the patient need antibiotics?
1) check: Does the patient have an infection?
2) check: Does the patient have a sore throat?
3) If both hold, conclude "need antibiotics".
This reduces unnecessary computations by only exploring relevant facts.

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**

   To find the mean, sum all values and divide by the number of values:

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

   > Sum = 1611

2. **Find the Median**

   To find the median, first 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 is the average of the 10th and 11th values: Median = (81 + 82) / 2

   > Median = 81.5

3. **Find the Mode**

   The mode is the value that appears most frequently in the dataset. Looking at the ordered data: 59, 64, 66, 70, 76, 76, 76, 78, 79, 81, 82, 82, 84, 85, 88, 90, 90, 91, 95, 99

   The value 76 appears three times, while all other values appear once or twice.

   > Mode = 76

4. **Find the Interquartile range**

   To find the IQR, I need to calculate Q1 (25th percentile) and Q3 (75th percentile).

   For Q1: With 20 values, Q1 is the median of the first 10 values: 59, 64, 66, 70, 76, 76, 76, 78, 79, 81 Q1 = (70 + 76) / 2 = 73

   For Q3: Q3 is the median of the last 10 values: 82, 82, 84, 85, 88, 90, 90, 91, 95, 99 Q3 = (88 + 90) / 2 = 89

   IQR = Q3 - Q1 = 89 - 73 = 16

   > Q1=73, Q3=89, IQR=16

Q.2 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

1. Machine Learning for Kids

- **Target Audience:** Primary and secondary school students (K-12), Teachers and educators with limited coding experience, Parents interested in introducing ML concepts to children

- **Use:** Students use visual blocks-based programming (Scratch) to create and train simple ML models, Teachers incorporate it into STEM curricula to introduce AI/ML concepts, Students learn by creating interactive projects like games, stories, and simple applications that use ML

- **Benefits:**
  - Simplifies complex ML concepts through visual programming
  - No coding experience required to get started
  - Provides real hands-on experience with ML model training
  - Designed specifically for educational contexts
  - Focuses on ethical AI use and understanding
- **Drawbacks:**
  - Limited in complexity compared to professional ML tools
  - Simplified models may not translate directly to real-world applications
  - Requires teacher supervision for younger students
  - Limited model types and capabilities

2. Teachable Machine

- **Target Audience:** Non-technical users interested in ML, Educators at various levels, Artists and creative technologists, Students (middle school through university), Professionals looking to prototype ML solutions quickly

- **Use:** Quick creation of custom image, sound, or pose recognition models, Integration into creative projects and installations, Classroom demonstrations of ML concepts, Rapid prototyping of ML ideas without coding

- **Benefits:**

    - Extremely accessible with no coding required
    - Browser-based with no software installation
    - Real-time training and feedback
    - Models can be exported for use in other applications
    - Free to use
- **Drawbacks:**
    - Limited to specific model types (image, audio, pose)
    - Less customization than professional ML platforms
    - Models may not be as robust as those built with more advanced tools
    - Limited control over model architecture

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

Predictive analytics

- **Description:** Predictive analytics uses statistical algorithms, machine learning, and data mining techniques to analyze historical data and make predictions about future events or behaviors. It helps forecast trends and outcomes by identifying patterns and relationships in data. For example, it can be used to predict customer behavior, stock market trends, or equipment failure.
- **Why Predictive analytics:** Predictive analytics is typically chosen when there is a need to anticipate future events, understand trends, and make data-driven decisions. Businesses use it for risk management, forecasting, marketing strategies, and optimizing operations.

Descriptive analytic

- **Description:** Descriptive analytics involves the process of analyzing historical data to gain insights into what has already happened. It summarizes past events through methods such as reporting, data visualization, and basic statistical analysis. Descriptive analytics answers questions like "What happened?" and "Why did it happen?" It typically includes measures like averages, counts, or percentages.
- **Why choose it:** Descriptive analytics is selected when the goal is to understand past performance or identify trends that have already occurred. It is used for performance tracking, monitoring key metrics, and summarizing large datasets for easy comprehension.

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

Supervised Learning:

- **Description:** In supervised learning, the model is trained on labeled data. That means the training data includes both the inputs (features) and the correct outputs (labels). The model learns by comparing its predictions to the true labels, and over time, it adjusts to minimize errors.
- **Why choose it**: If we think of how a machine learning model might be trained to predict outcomes based on past data, this corresponds to supervised learning. It's about learning from past examples where the correct answers are already known.

Unsupervised Learning:

- **Description:** In unsupervised learning, the model works with data that has no labels or predefined outcomes. The goal is to find structure or patterns in the data, such as clustering similar items or reducing dimensionality.
- **Why choose it**: This is useful for scenarios where you don’t know the exact relationships beforehand, and you're interested in letting the model explore the data's underlying structure, like grouping similar data points or uncovering hidden patterns.

Reinforcement Learning:

- **Description:** In reinforcement learning, an agent learns by interacting with an environment. It makes decisions and gets feedback in the form of rewards or penalties. The agent's goal is to maximize cumulative reward over time.
- **Why choose it**: This is like training a model to perform tasks based on trial and error, receiving rewards for making good decisions and penalties for bad ones. It's about optimizing actions for long-term success, much like training an AI to play a game or control a robot.

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.

> Answer:

- A recent example of misleading data visualization involves a social media post comparing child poverty rates between the UK and Nordic countries.
- The post claimed that the UK's child poverty rate was 32.1%, while Denmark's was 2.4%, Finland's 3.2%, Norway's and Sweden's 3.6%.
- However, these figures were based on outdated and inaccurate data, leading to a distorted perception of the disparities.
- The Organisation for Economic Co-operation and Development's (OECD) 2000 data indicated the UK's child poverty rate at 16.2%, not 32.1%, using a specific metric.
- More recent OECD data from 2019 show the UK's rate at 14.1%, which, though higher than those of the Nordic countries, is not as dramatically different as the post suggested.
- This misrepresentation highlights how using outdated and inconsistent data can lead to misleading visualizations, affecting public understanding and policy discussions.
- Article Link: https://www.reuters.com/fact-check/child-poverty-comparison-northern-europe-uses-old-inaccurate-data-2024-10-01

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

https://www.kaggle.com/datasets/uciml/pima-indians-diabetes-database

-> Naïve Base Classifier:

1. **Data Loading & Pre-processing:**
    - Load the data from Classification data.csv.
    - Handle missing data if necessary.
    - Feature scaling using StandardScaler or MinMaxScaler.
    - Split data into features (X) and target (y), with y being the last column.
    - Split into training, validation, and test sets.

2. **Model Selection & Hyperparameter Tuning:**
    - Use either SVM (e.g., SVC from sklearn.svm) or Naive Bayes (e.g., GaussianNB from sklearn.naive_bayes).
    - Use GridSearchCV or RandomizedSearchCV for hyperparameter tuning.

3. **Class Imbalance Handling:**
    - Use techniques such as oversampling (e.g., SMOTE), undersampling, or assigning class weights.

4. **Model Training & Evaluation:**
    - Train the model using the training data.
    - Evaluate on the validation set.
    - Test on the test set and visualize performance using metrics such as accuracy, precision, recall, F1-score, and confusion matrix.

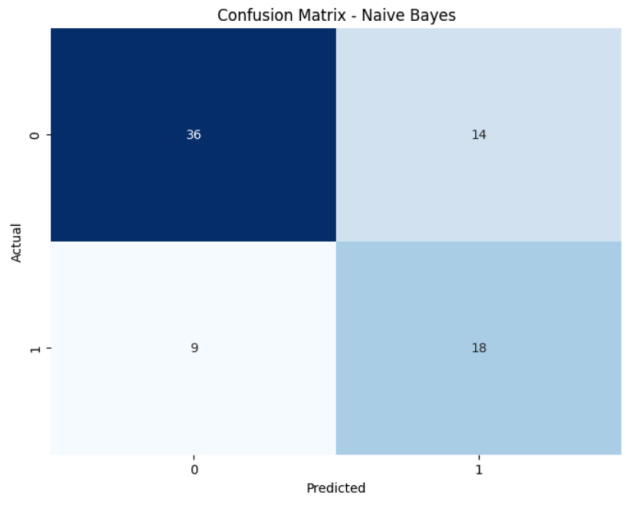
Result:

1. **Best Hyperparameters**: var_smoothing = 0.1 for Naive Bayes.
2. **Validation Accuracy**: 77.78% (performance on validation data).
3. **Test Accuracy**: 70.13% (performance on test data).
4. **Metrics:**
    a. **Class 0 (Negative outcome)**:
        i. **Precision**: 80%
        ii. **Recall**: 72%
        iii. **F1-Score**: 76%

    b. **Class 1 (Positive outcome)**:
        i. **Precision**: 56%
        ii. **Recall**: 67%
        iii. **F1-Score**: 61%

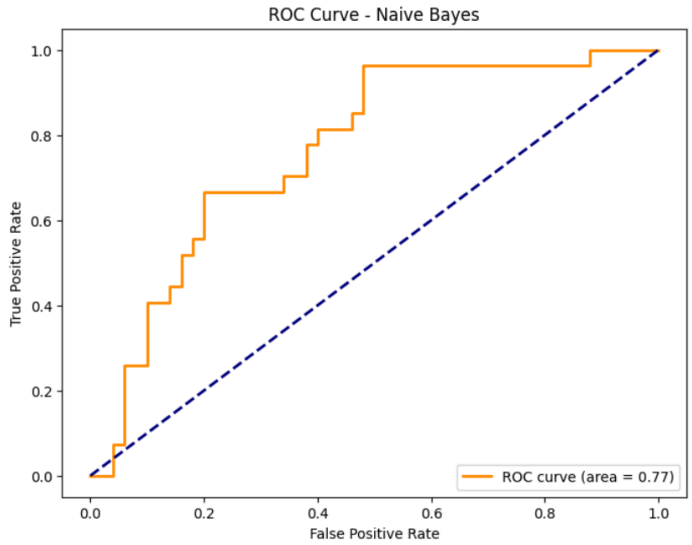
5. **Averages:**
    a. **Macro Average**: Balanced performance across both classes.
    b. **Weighted Average**: More influenced by class 0, as it has more samples.

Confusion Matrix



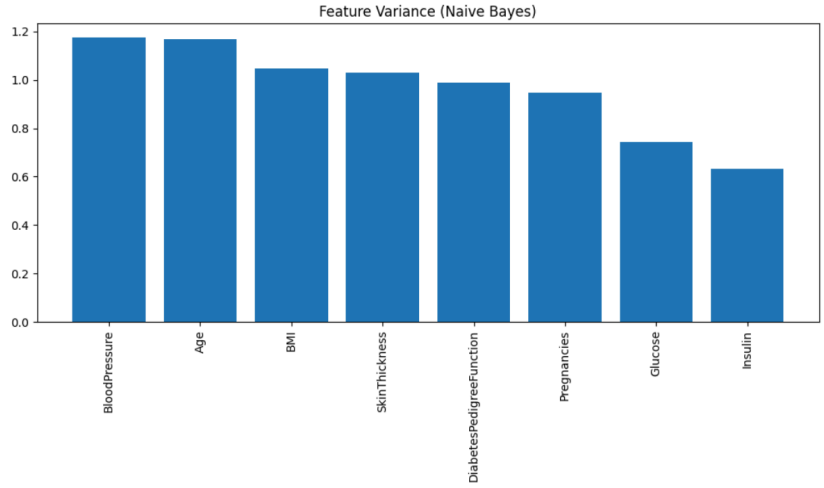
True Negatives (0 predicted as 0): 36, **False Positives: 14**, **False Negatives: 9**, **True Positives: 18**.The model performs better at predicting class 0, but struggles with class 1, showing a tendency to misclassify positives.

ROC Curve



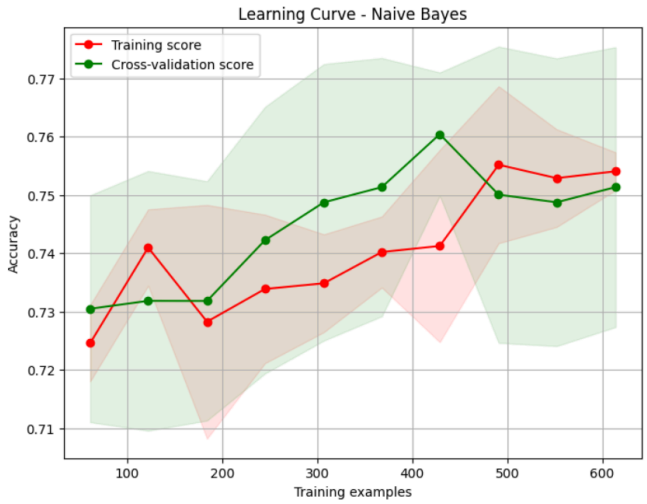
The ROC curve for the Naive Bayes model shows an **AUC of 0.77**, indicating decent performance. The model has a **77% chance of distinguishing between the two classes**, performing better than random but leaving room for improvement.

Feature Variance Visualization



As you can see, blood pressure has the highest variance at around 1.2 and insulin has the lowest at 0.6.

Learning Curve



This learning curve shows the accuracy of a Naive Bayes classifier as the number of training examples increases. The model exhibits consistent improvement in both training and cross-validation scores, indicating good generalization with more data.

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

- **Data File**: Regression data.csv, Independent Variable: 1st Column, Dependent variables: Column 2 to 5, Use any Regression model to predict the values of all Dependent variables using values of 1st 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, Use any Python library to present the accuracy measures of trained model

https://github.com/Sutanoy/Public-Regression-Datasets

1. Import necessary libraries
2. Load the dataset (Mall_Customers.csv)
3. Create a DataFrame from the dataset
4. Define a RegressionModel class

- Initialize scaler, label encoder, model

- Preprocess the data: Encode categorical variables (Gender), Select features (Gender, Age, Annual Income (k$)), Set target (Spending Score (1-100)), Scale features
- Train the linear regression model
- Test the model and print metrics: R^2, Adjusted R^2, MAE
- Perform hyperparameter tuning using GridSearchCV

5. In main() function: Preprocess the dataset, Split data into training and testing sets (70/30), Perform hyperparameter tuning, Train the model, Test the model and get predictions/metrics.

Result:

```
Best Parameters: {'fit_intercept': True}
R2 Score: 0.12084676260712823
Adjusted R2 Score: 0.0737492677467958
Mean Absolute Error: 19.06031176341787
```

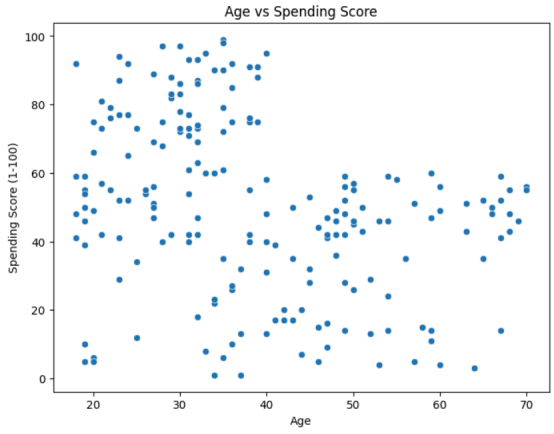
Best Hyperparameter: The best parameter found is fit_intercept=True, which means the model performs better when it includes an intercept term.

R^2 Score: 0.12

- The model explains only **12%** of the variance in the target variable (Spending Score).
- This indicates **poor predictive power** — the model does not capture most of the patterns in the data.

Adjusted R^2 Score: 0.074

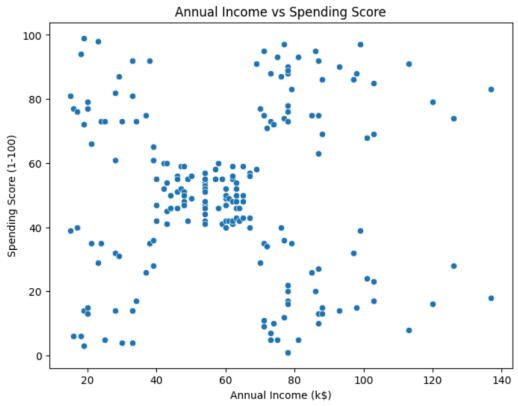
- Even lower than R^2, suggesting that adding the current features (Gender, Age, Annual Income) didn't improve the model significantly.
- This is common when predictors don't have a strong linear relationship with the target.



Age vs Spending Score

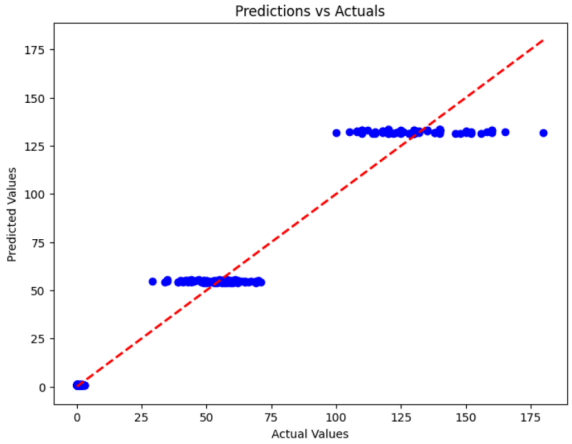
- There's no clear linear relationship between Age and Spending Score.

- Younger customers (around 18–30) show more variation in spending — some spend a lot, others very little.
- Older customers (above 50) tend to have lower and more consistent spending scores.



Annual Income vs Spending Score

- No clear linear relationship.
- Spending scores vary across all income levels.
- Dense cluster between 40k–70k income, but spending still scattered.
- High income ≠ high spending.
- Clustering may reveal better patterns



Predictions vs Actuals Plot

- Predicted values are clustered, not smoothly spread along the diagonal.
- Indicates poor model performance – predictions fail to follow actual values.
- The red diagonal shows perfect prediction, but most points deviate from it.

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).

Dataset link: https://www.kaggle.com/datasets/rajyellow46/wine-quality

The **Wine Quality Dataset** on Kaggle is a popular dataset for predicting wine quality based on various physicochemical features. It's used in machine learning and data analysis to classify wine quality, which is usually scored between 0 and 10. The dataset has two versions: one for **red wine** and one for **white wine**. Here are the **key features** of the dataset, their importance, and a discussion on **missing data handling** during feature engineering:

Key Features of the Wine Quality Dataset

The dataset contains several physicochemical features, which are important for determining the quality of the wine. Here's a list of the most important ones:

1. **Fixed Acidity**:
   - Represents the amount of fixed acids (e.g., tartaric acid) in the wine.
   - Importance: Affects taste, especially sourness. It can also influence the pH, which in turn impacts the stability and fermentation process of wine.

2. **Volatile Acidity**:
   - Measures the concentration of acetic acid and other volatile acids in the wine.
   - Importance: High levels of volatile acidity can cause the wine to smell vinegary and lower its quality. It's an important indicator for the perception of off-flavors.

3. **Citric Acid**:
   - A natural preservative that adds a fresh, citrusy flavor.
   - Importance: It plays a role in the overall flavor profile and helps balance out acidity in the wine. It can also contribute to the wine's stability and aging potential.

4. **Residual Sugar**:
   - The amount of sugar remaining after fermentation.
   - Importance: A higher sugar level can result in a sweeter wine, and sugar content has a direct effect on the wine's body and taste.

5. **Chlorides**:
   - Measures the concentration of salts (e.g., sodium chloride) in the wine.

- Importance: Higher chloride concentrations may indicate a salty taste, which could negatively affect wine quality.

6. **Free Sulfur Dioxide (SO2)**:
   - Refers to the amount of free sulfur dioxide in the wine.
   - Importance: Acts as an antioxidant and antimicrobial agent, preventing oxidation and microbial growth. However, high levels can give a pungent smell and negatively impact flavor.

7. **Total Sulfur Dioxide (SO2)**:
   - Total amount of sulfur dioxide, both free and bound.
   - Importance: Similar to free sulfur dioxide, but also considers bound forms. It helps to prevent spoilage, but excessive levels can affect the taste and aroma.

8. **Density**:
   - Measures the wine's density, which is influenced by alcohol content, sugar concentration, and other factors.
   - Importance: It's an indirect indicator of alcohol content and sugar levels, which both influence wine quality.

9. **pH**:
   - Indicates the acidity or alkalinity of the wine.
   - Importance: It affects the stability, flavor, and aging potential of the wine. A balanced pH contributes to better taste and preservation.

10. **Sulphates**:
    - Measures the concentration of sulfur-containing compounds in the wine.
    - Importance: Similar to sulfur dioxide, it has preservative properties, but excessive levels could impact flavor.

11. **Alcohol**:
    - The alcohol content in the wine.
    - Importance: Higher alcohol content is often associated with higher quality wine, as it affects body and mouthfeel. It also plays a role in fermentation and preservation.

12. **Quality**:
    - The target variable, representing the quality of the wine (scored between 0 and 10).
    - **Importance**: This is the variable we aim to predict using the other features.

Handling Missing Data During Feature Engineering

Handling missing data is a crucial part of the feature engineering process. If missing data is not addressed properly, it can significantly affect the quality of predictions and the overall model's performance. Here's how to approach missing data and the techniques for imputation:

1. **Identifying Missing Data:** Before deciding how to handle missing data, it's essential to identify which features contain missing values. This can be done using techniques like checking the percentage of missing values per feature and visualizing missing data patterns.

2. **Imputation Techniques:**

- Mean/Median Imputation:
    - Description: Replacing missing values with the mean or median of the feature. The mean is used for numerical features that are normally distributed, and the median is typically used for skewed features.
    - Advantages: Simple, fast, and easy to implement. It works well when data is missing at random and the distribution is relatively stable.
    - Disadvantages: It may introduce bias and reduce variance in the data, especially if the data is not missing at random (e.g., systematic patterns). Also, it doesn't capture the underlying distribution of the missing values.

- Mode Imputation (for categorical data):
    - Description: For categorical features, missing values can be replaced with the most frequent category.
    - Advantages: Quick and easy to apply.
    - Disadvantages: Similar to mean/median imputation, it can distort the distribution of the feature and lead to biased models if the missingness is not random.

- K-Nearest Neighbors (KNN) Imputation:
    - Description: This technique imputes missing values based on the values of similar instances (neighbors).
    - Advantages: It can provide more contextually accurate imputations by leveraging relationships between features.
    - Disadvantages: Computationally expensive, especially with large datasets. Also, the choice of "k" (the number of neighbors) can influence the results.

- Multivariate Imputation by Chained Equations (MICE):
    - Description: MICE imputes missing values by modeling each feature with missing data as a function of the other features in the dataset. It performs multiple imputations to provide a more robust estimate.
    - Advantages: It is effective when dealing with complex datasets with missing values in multiple columns. It helps preserve the relationships between features.

    - Disadvantages: More computationally intensive and might require careful parameter tuning.

- Model-based Imputation:
    - Description: Use machine learning models (e.g., regression, decision trees) to predict missing values based on other features.
    - Advantages: It can capture complex relationships in the data and provide accurate imputations.
    - Disadvantages: Can be computationally expensive and may overfit if not implemented carefully.

3. Deleting Missing Data: In cases where the missing data is minimal (say less than 1-2% of the total data), one might choose to drop the rows or columns containing missing values.

- Advantages: Simple and reduces the risk of introducing bias.
- Disadvantages: Can lead to loss of valuable information if missing data is not random or if large portions of the data are missing.

Advantages and Disadvantages of Different Imputation Techniques

- **Mean/Median Imputation**:
    - Advantages: Simple and efficient for numerical features, especially when the dataset is small and the missingness is random.
    - Disadvantages: Can distort feature distributions, leading to inaccurate predictions.
- **KNN Imputation**:
    - Advantages: Considers the similarity between data points, leading to potentially more accurate imputations.
    - Disadvantages: Computationally expensive, especially with large datasets or high-dimensional data.
- **MICE (Multiple Imputation by Chained Equations)**:
    - Advantages: Accounts for the relationships between features and performs multiple imputations to reduce bias.
    - Disadvantages: Requires more computational resources and is slower than other methods.
- **Model-based Imputation**:
    - Advantages: Captures non-linear relationships between features and produces more accurate imputations.
    - Disadvantages: Can overfit if not carefully validated and is computationally demanding.

In summary, handling missing data is an essential step in the feature engineering process. The choice of imputation technique depends on the nature of the data, the amount of missing data, and the computational resources available. Each technique has its strengths and weaknesses, so understanding the characteristics of the dataset and the underlying patterns of missingness is crucial for making the right decision.