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.

Columns that might not be necessary for analysis include Recall Link, Mfr Campaign Number, Park Outside Advisory,Completion rate(%).These columns do not provide much insight in the context of data analysis for recall trends or consequences. Therefore, you can drop them to simplify the dataset.

```
# 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.**

Here we drop the rows which have more than 50% missing values.these can be done by dropna() function with threshold parameter=0.5.

```
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 the above information says Potential Affected ,Recall Description ,Consequence Summary and corrective action contain some null values thus we need to handle missing data.For these columns, either fill in with a placeholder (e.g., "Unknown") or drop the rows if the missing data is significant.

```
[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

For columns containing categorical data (e.g., Recall Type), we can create dummy variables. This is helpful for machine learning models.

```
# 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):

Outliers can be detected by looking at numerical columns like Potentially Affected. One method for identifying outliers is by visualizing the data using box plots or using statistical methods like the Z-score.

First Quartile (Q1):=QUARTILE(H2:H28719, 1)
Q1 = 99
Third Quartile (Q3):=QUARTILE(H2:H28719, 3)
Q3 = 6386
Interquartile Range (IQR):=Q3 - Q1

IQR = 6386 - 99 = 6287

Outlier Boundaries:

Lower Bound:=Q1 - 1.5 * IQR

Lower Bound = 99 - (1.5 * 6287) = -9331.5

Upper Bound: =Q3 + 1.5 * IQR

Upper Bound = 6386 + (1.5 * 6287) = 15816.5

Identifying Outliers: Any value less than -9331.5 or greater than 15816.5 is considered an outlier.

01/31/2025	25V048000	Ford Motor Company	BACK OVER PREVENTIO	25S05	Vehicle	72624
01/30/2025	25V043000	Jayco, Inc.	EQUIPMENT	9901617	Vehicle	412
01/30/2025	25V045000	Autocar, LLC	ELECTRICAL SYSTEM	ACTT-2501	Vehicle	130
01/28/2025	25V037000	Mitsubishi Fuso Truck of Am	ELECTRICAL SYSTEM	C10129	Vehicle	233
01/24/2025	25V034000	Forest River, Inc.	EQUIPMENT	203-1889	Vehicle	64
01/23/2025	25V029000	Winnebago Towable	EQUIPMENT	CAM0000041	Vehicle	144
01/23/2025	25V031000	Honda (American Honda Mo	ELECTRICAL SYSTEM	EL1, AL0	Vehicle	294612
01/23/2025	25V033000	Subaru of America, Inc.	WHEELS	WRB-25	Vehicle	20366
01/23/2025	25V030000	Mack Trucks, Inc.	SERVICE BRAKES, AIR	SC0474	Vehicle	142
01/23/2025	25V032000	Honda (American Honda Mo	BACK OVER PREVENTIO	RKZ	Vehicle	9221
01/22/2025	25V027000	Forest River Bus, LLC	STRUCTURE	05-1890	Vehicle	37
01/22/2025	25V028000	Toyota Motor Engineering &	FUEL SYSTEM, GASOLIN	25TA01 / 25LA01	Vehicle	858
01/21/2025	25V026000	Mack Trucks, Inc.	SERVICE BRAKES, AIR	SC0473	Vehicle	21
01/21/2025	25E006000	Oshkosh Corporation	VEHICLE SPEED CONTR(	NR (Not Reported)	Equipment	500
01/17/2025	25V021000	Forest River, Inc.	EQUIPMENT	503-1887	Vehicle	18
01/17/2025	25E004000	Cummins, Inc.	FUEL SYSTEM, DIESEL	C7111	Equipment	715
01/17/2025	25V024000	Kia America, Inc.	ELECTRICAL SYSTEM	SC332	Vehicle	80255
01/17/2025	25T001000	Pirelli Tire, LLC	TIRES	NR (Not Reported)	Tire	2023
01/17/2025	25V023000	Mercedes-Benz USA, LLC	TIRES	NR (Not Reported)	Vehicle	165
01/17/2025	25V020000	Ford Motor Company	POWER TRAIN	25S03	Vehicle	259
01/17/2025	25V019000	Ford Motor Company	ELECTRICAL SYSTEM	25S02	Vehicle	272817
01/17/2025	25V025000	Ford Motor Company	SUSPENSION	25S01	Vehicle	149449

7. standardization and normalization of column

Standardization and normalization are crucial when dealing with numerical data that varies in scale, especially for machine learning algorithms.

```
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.

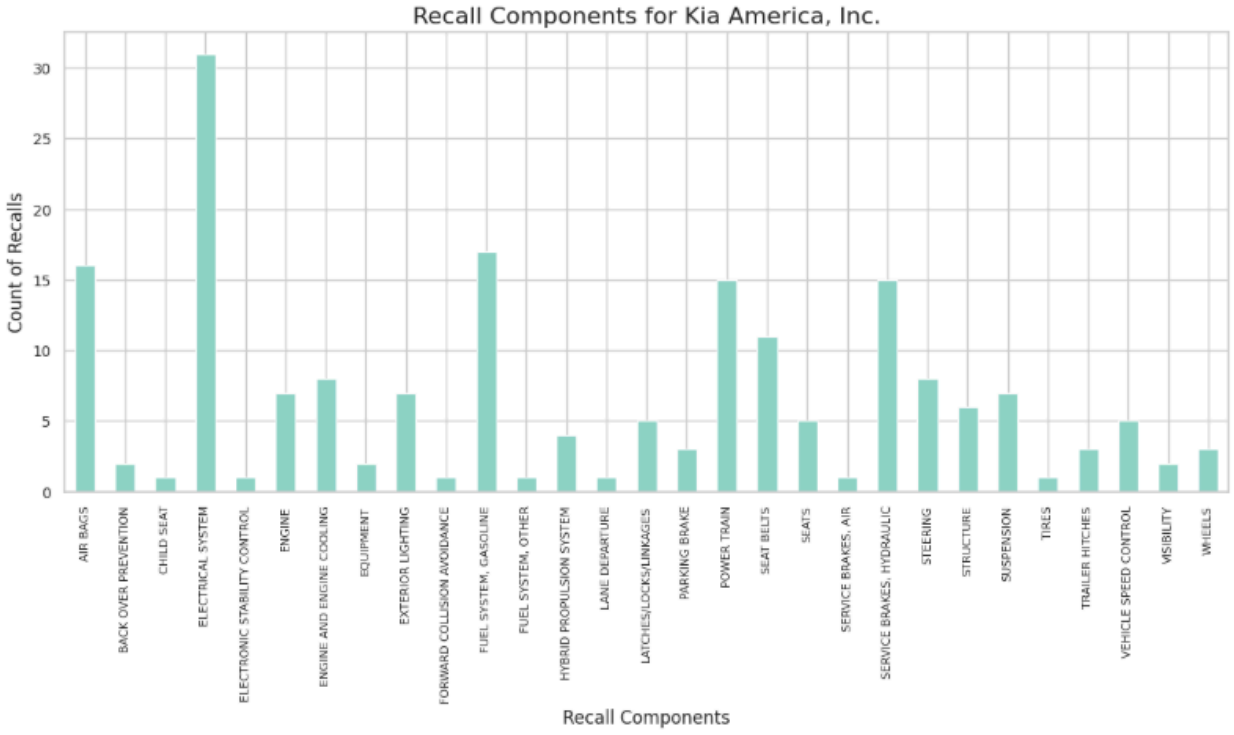
Theory:

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.

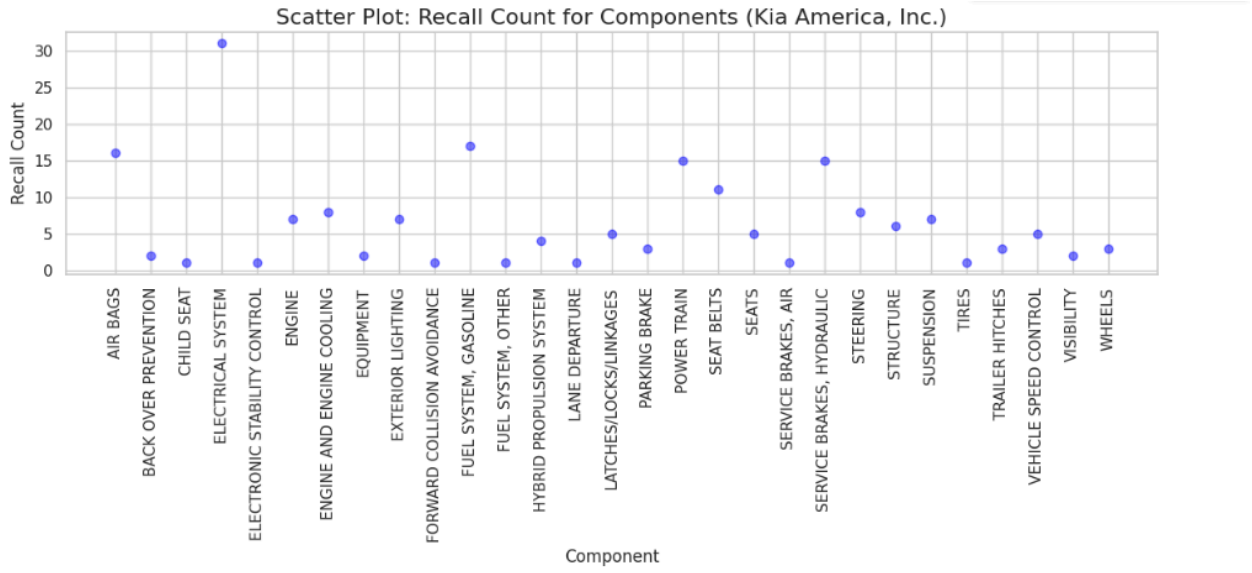
```
import pandas as pd
import matplotlib.pyplot as plt

# 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
```

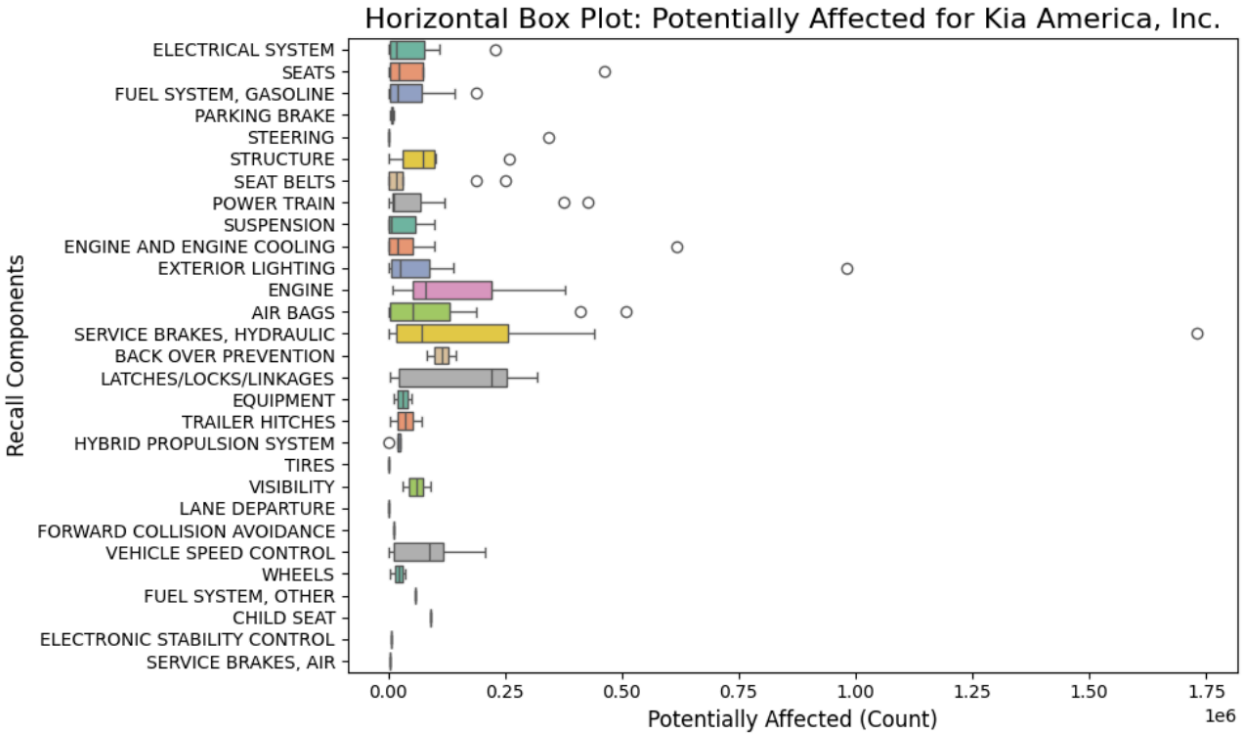
2. Plot Scatter plot, box plot, Heatmap using seaborn.

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()
```



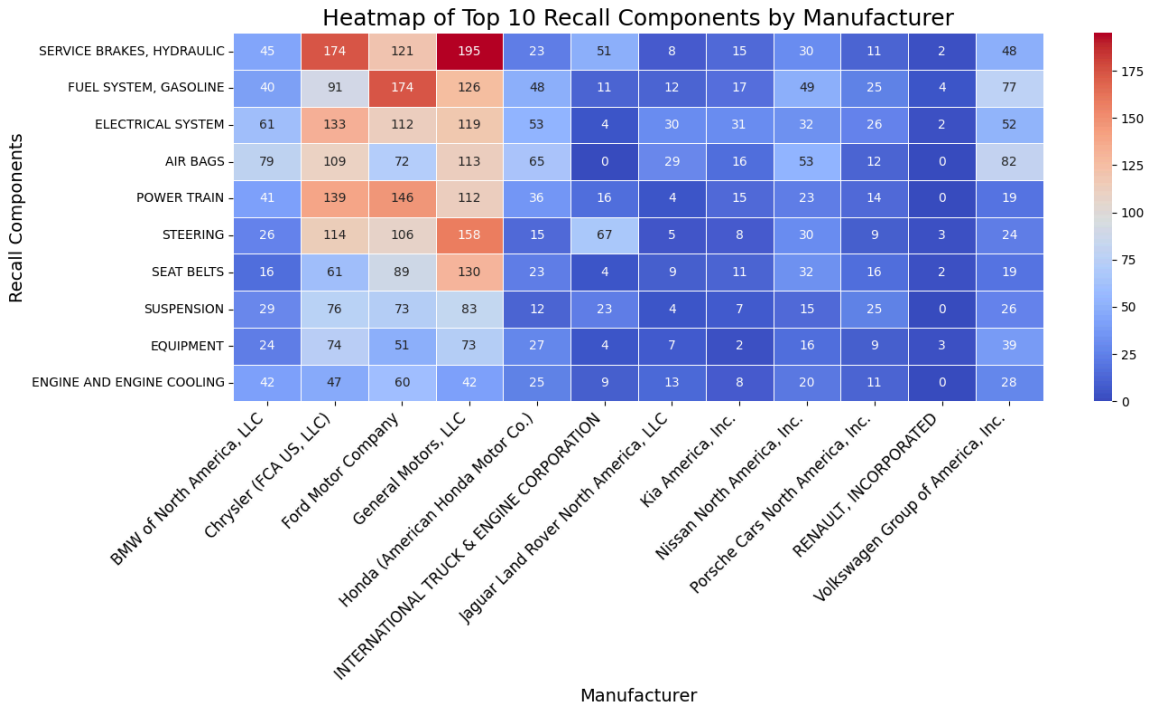
2. Box plot: The box plot shows the distribution of the number of potentially affected vehicles for each recalled component. The horizontal layout makes it easier to compare components.It identifies the spread, central tendency, and outliers in the data, with whiskers representing the data range and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

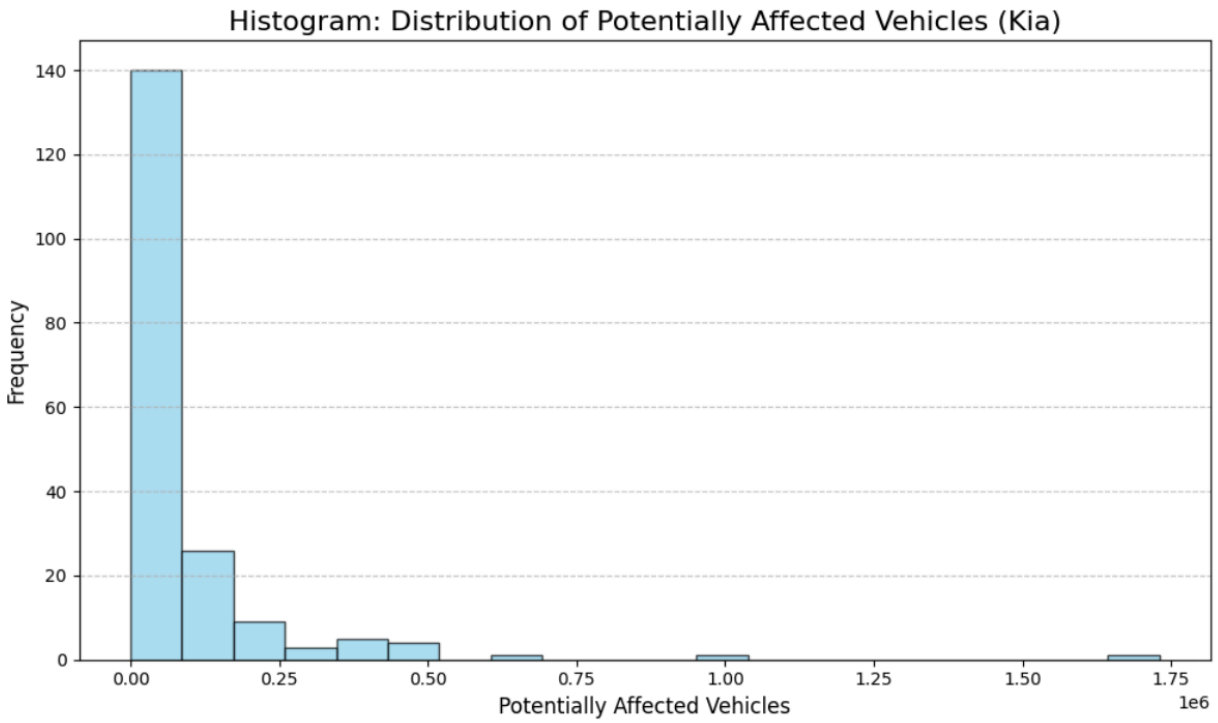


3. Create histogram and normalized Histogram.

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

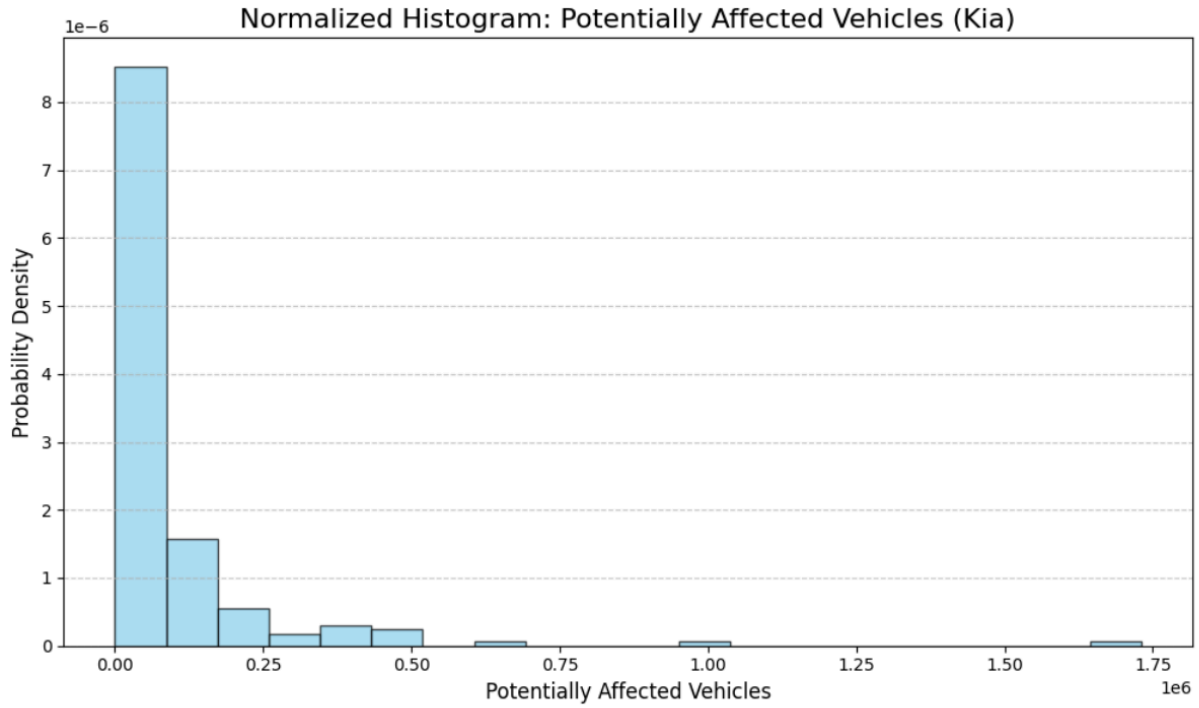
Regular Histogram: It displays counts in bins.

```
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()
```



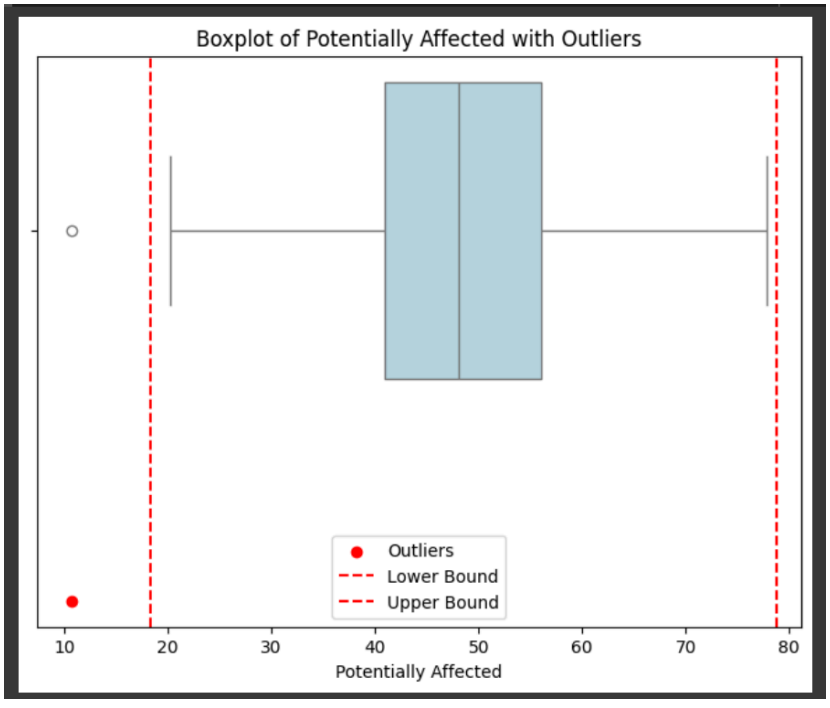
This highlights the data's skewness, central tendency, and spread. The normalized version shows the proportion of each bin relative to the total.

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()
```



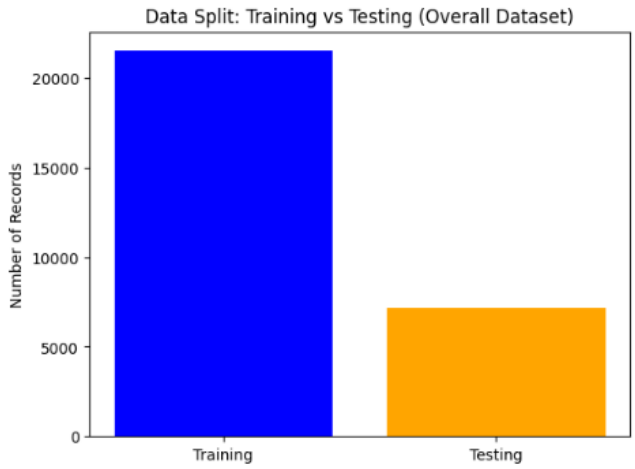
Conclusion: Bar Graph and Scatter Plot are suitable for identifying trends and comparisons.Heatmap is excellent for understanding correlations between variables. Box Plot provides insights into distributions and outliers.Normalized Histogram helps understand probabilities and densities.Outlier Handling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

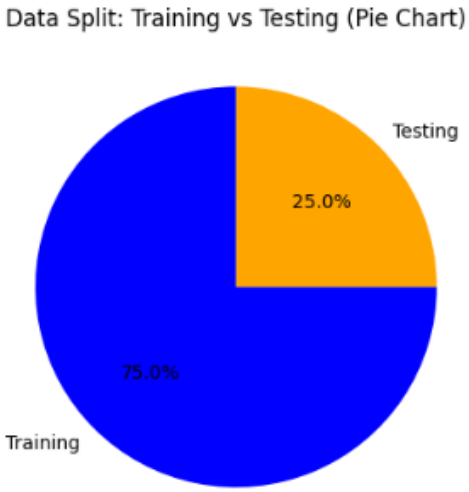
Experiment 3

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.

```
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.





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

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.

Experiment 4

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

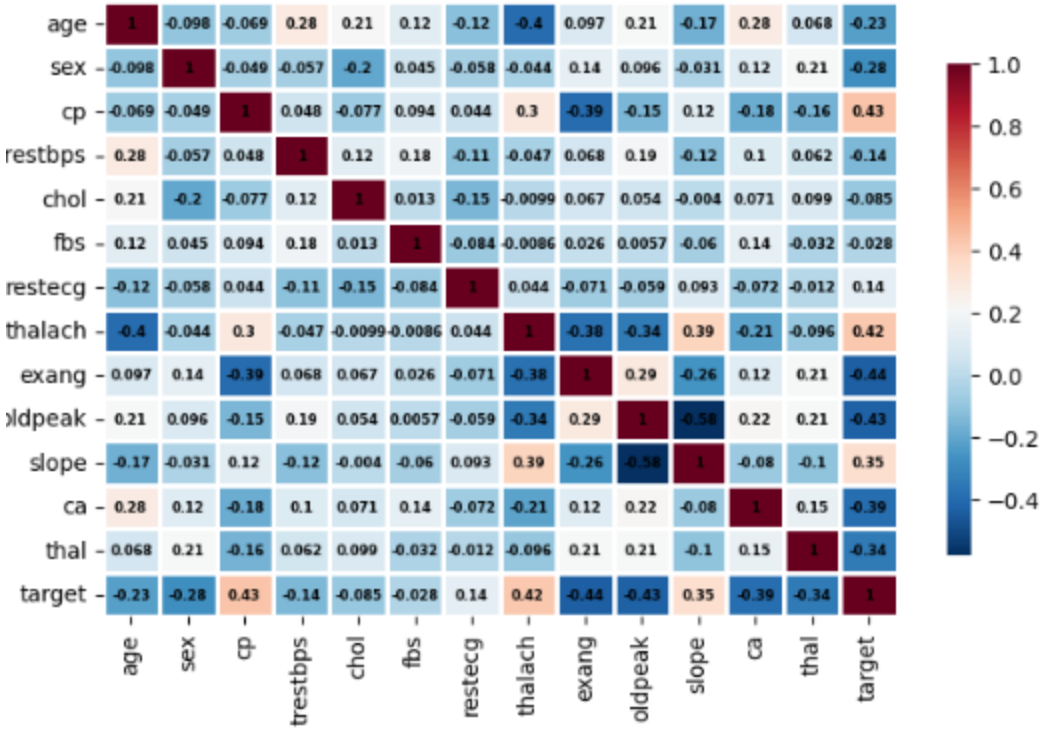
Theory:

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

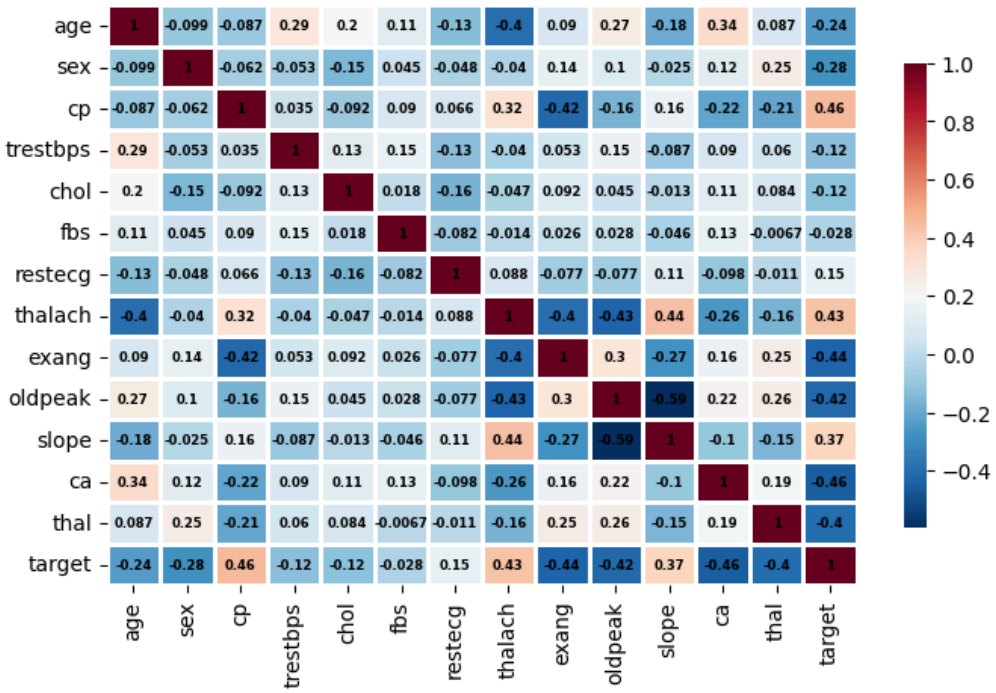
```
import seaborn as sb
import matplotlib.pyplot as plt

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()
```

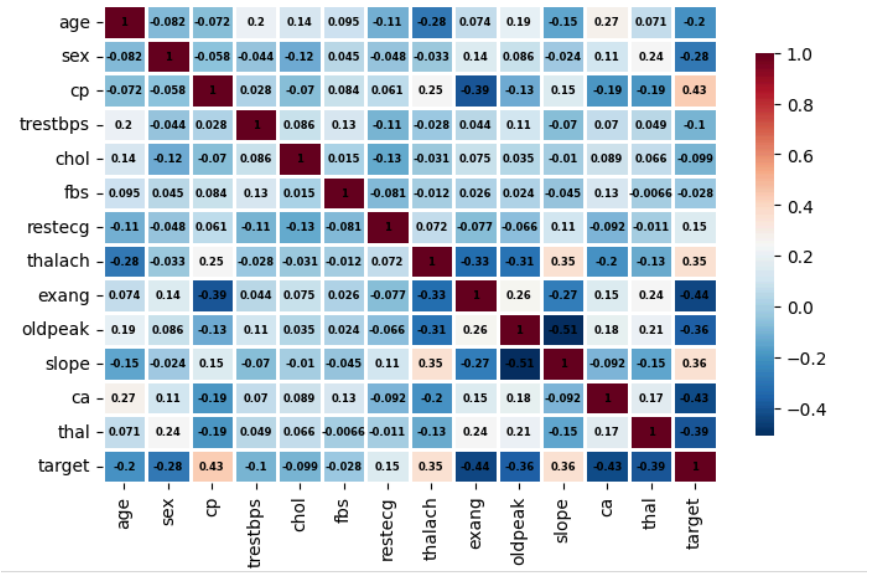


Results: The correlation between cp (chest pain type) and target is 0.46, indicating a moderate positive association.

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**

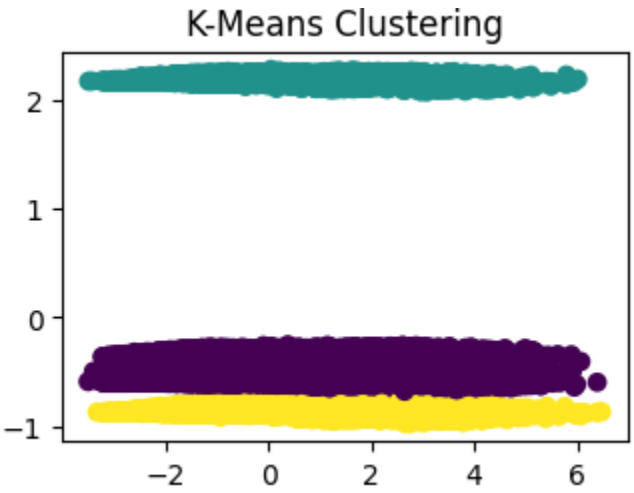
```
df=pd.read_csv('diabetes_dataset_with_notes.csv')
df = pd.DataFrame(df)
```

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.

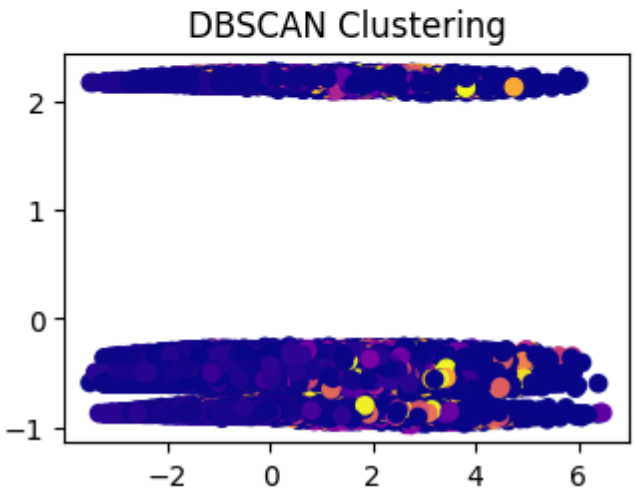
```
kmeans = KMeans(n_clusters=3, random_state=42)
df['kmeans_cluster'] = kmeans.fit_predict(df_scaled)

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()
```



- 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()
```



- 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.

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.

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

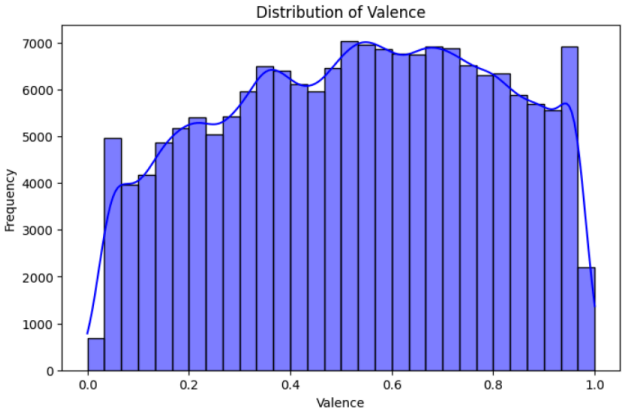
```
import pandas as pd
# Load the dataset
df = pd.read_csv("data.csv")
# Display first few rows
df.head()
```

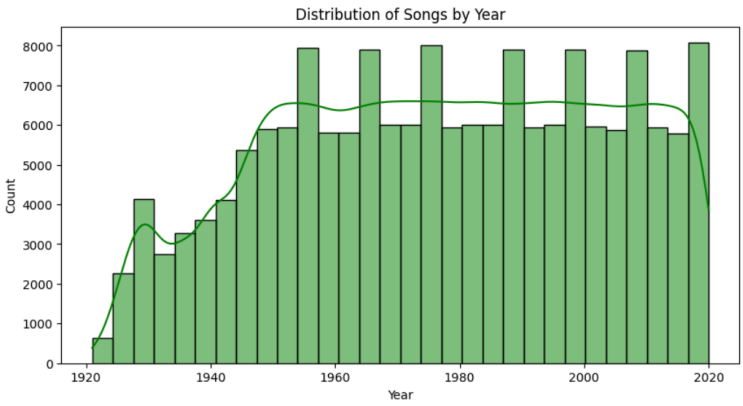
	valence	year	acousticness	artists	danceability	duration_ms	energy	explicit	id	instrumentalness	key	liveness
0	0.0594	1921	0.982	['Sergei Rachmaninoff', 'James Levine', 'Berli...	0.279	831667	0.211	0	4BJqT0PrAfrxzMOxytFOIz	0.878000	10	0.665
1	0.9630	1921	0.732	['Dennis Day']	0.819	180533	0.341	0	7xPhfUan2yNtyFG0cUWkt8	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
```

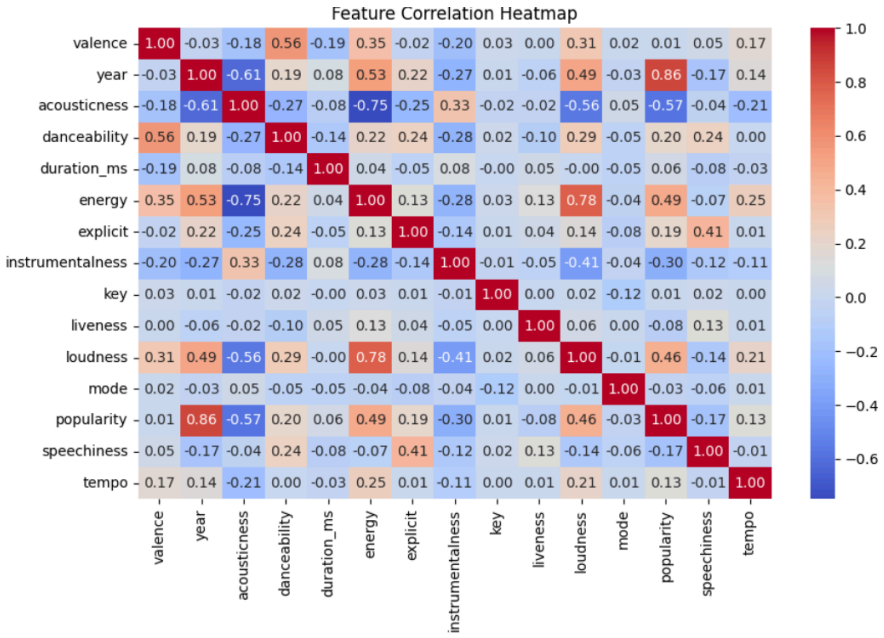
	0
valence	float64
year	int64
acousticness	float64
artists	object
danceability	float64
duration_ms	int64
energy	float64
explicit	int64

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.

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:

Example: Misleading Data Visualization in the Australian Broadcasting Corporation (ABC) Article on Housing Crisis

In a recent article discussing solutions to the housing crisis, the Australian Broadcasting Corporation (ABC) included a chart that received significant criticism for its misleading design. The chart aimed to illustrate the potential of utilizing existing housing stock, including unoccupied homes, to address housing shortages.

How the Data Visualization Failed:

- **Inappropriate Chart Type:** The article featured a line chart to represent data that was not suited for such a visualization. Line charts are typically used to display trends over time, but in this case, the data represented categories that did not follow a temporal sequence.
- **Misleading Representation:** By treating "total" as a regular category and plotting it on the same axis as other data points, the chart created confusion. This design choice gave an impression of a trend where none existed, leading readers to misinterpret the information.

These design flaws led to widespread criticism, with observers labeling the chart as one of the most misleading visualizations encountered. The chart has since been removed from the live article, though it remains accessible via the Internet Archive.

https://www.ft.com/content/58dce921-2cdf-4140-9eb7-77792749dc88

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.

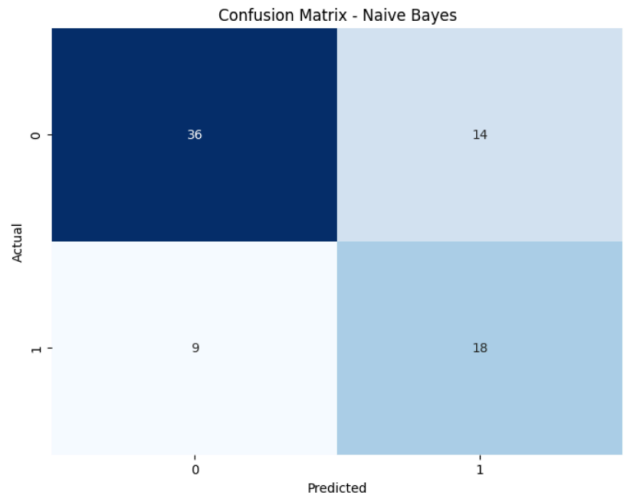
Result:

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%

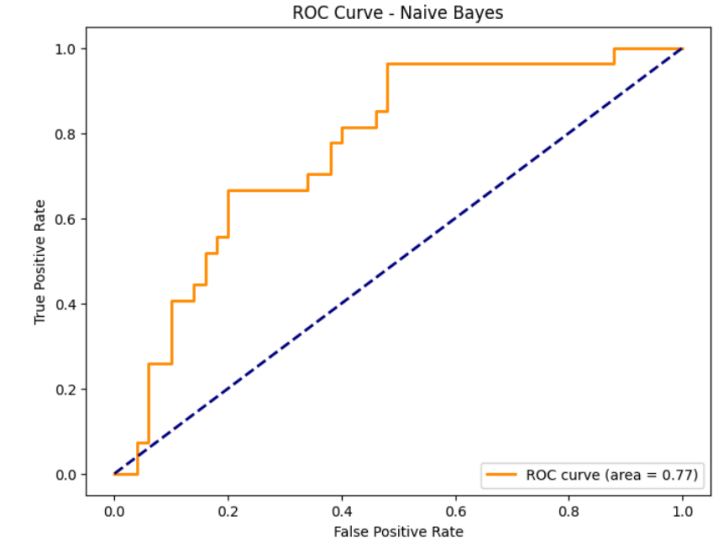
5. **Averages:**
    a. **Macro Average**: Balanced performance across both classes.
    b. **Weighted Average**: More influenced by class 0, as it has more samples.

Confusion Matrix



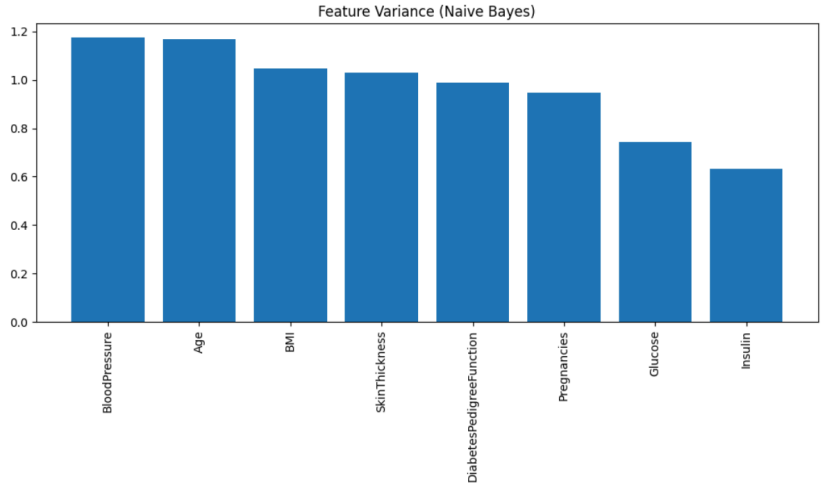
True Negatives (0 predicted as 0): 36, **False Positives: 14**, **False Negatives: 9**, **True Positives: 18**.The model performs better at predicting class 0, but struggles with class 1, showing a tendency to misclassify positives.

ROC Curve



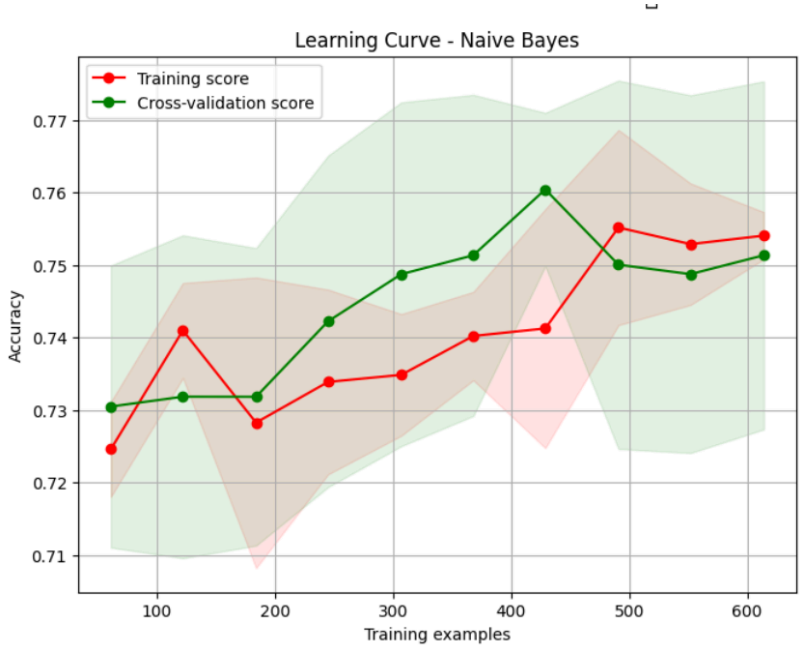
The ROC curve for the Naive Bayes model shows an **AUC of 0.77**, indicating decent performance. The model has a **77% chance of distinguishing between the two classes**, performing better than random but leaving room for improvement.

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

-> Step 1: Import Necessary Libraries: Import required libraries such as pandas, numpy, seaborn, matplotlib, and scikit-learn for data processing, modeling, and visualization.

Step 2: Load and Inspect the Dataset

1. Read the dataset using pd.read_csv().
2. Inspect the first few rows with data.head() to understand the structure of the dataset.
3. Check for missing values using data.isnull().sum() to ensure the data is clean.

Step 3: Split Data into Features and Target Variable: Separate the features (X) and the target variable (y). Here, X includes all the columns except ID and Classification, and y is the Classification column.

Step 4: Split Data into Training and Testing Sets: Use train_test_split() from scikit-learn to divide the data into training and testing sets. Typically, 80% of the data is used for training, and 20% is used for testing.

Step 5: Standardize the Features

1. Standardize the feature set using StandardScaler() to bring all features to the same scale (important for certain models, like Logistic Regression).
2. Apply scaling on both the training and testing sets (fit_transform() for training, and transform() for testing).

Step 6: Initialize and Train Models

- **Logistic Regression:**
    a. Initialize the Logistic Regression model using LogisticRegression().
    b. Train the model with the training data using model.fit(X_train_scaled, y_train).
- **Random Forest Classifier:**
    a. Initialize the Random Forest model using RandomForestClassifier().
    b. Train the model similarly with model.fit(X_train, y_train) (no scaling needed for Random Forest).

Step 7: Make Predictions

Use both trained models to predict the target variable (y_pred_log_reg for Logistic Regression and y_pred_rf for Random Forest) on the testing set (X_test or X_test_scaled).

Step 8: Evaluate the Models

- **Logistic Regression:**
    a. Calculate the accuracy using accuracy_score(y_test, y_pred_log_reg).
    b. Generate a classification report with classification_report(y_test, y_pred_log_reg).
    c. Create and visualize the confusion matrix using confusion_matrix() and seaborn.heatmap().
- **Random Forest:**Similarly, calculate the accuracy, classification report, and confusion matrix for the Random Forest model.

Step 9: Visualize the Results

- **Confusion Matrix:**
    - Plot confusion matrices for both models using seaborn.heatmap() to visually compare the performance of both models.
- **Feature Importance (for Random Forest):**
    - Extract the feature importances using rf_model.feature_importances_.
    - Create a DataFrame to sort and plot the feature importances, visualizing which features are most impactful for the Random Forest model's predictions.
- **Model Accuracy Comparison:**
    - Create a bar plot using seaborn.barplot() to visually compare the accuracies of Logistic Regression and Random Forest models.

Result:

	ID	Age	BMI	Glucose	Insulin	HOMA	Leptin	Adiponectin	\
0	1	48	23.500000	70	2.707	0.467409	8.8071	9.702400	
1	2	83	20.690495	92	3.115	0.706897	8.8438	5.429285	
2	3	82	23.124670	91	4.498	1.009651	17.9393	22.432040	
3	4	68	21.367521	77	3.226	0.612725	9.8827	7.169560	
4	5	86	21.111111	92	3.549	0.805386	6.6994	4.819240	

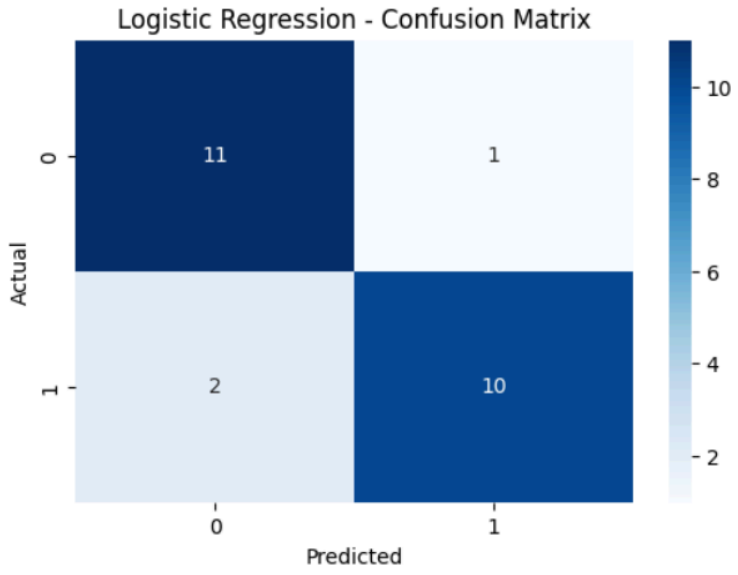
	Resistin	MCP.1	Classification
0	7.99585	417.114	0
1	4.06405	468.786	0
2	9.27715	554.697	0
3	12.76600	928.220	0
4	10.57635	773.920	0

Logistic Regression:

Logistic Regression Accuracy: 0.875

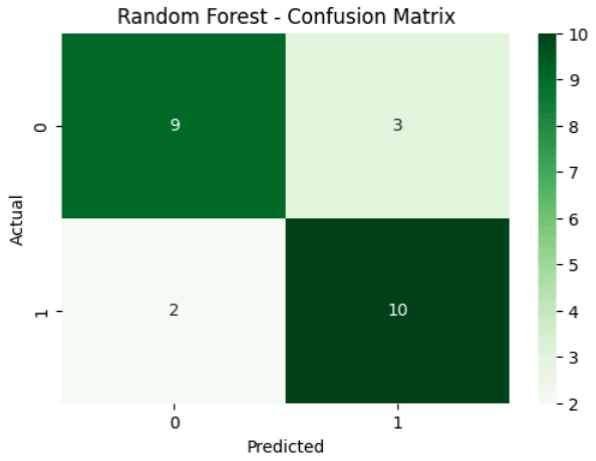
Classification Report (Logistic Regression):

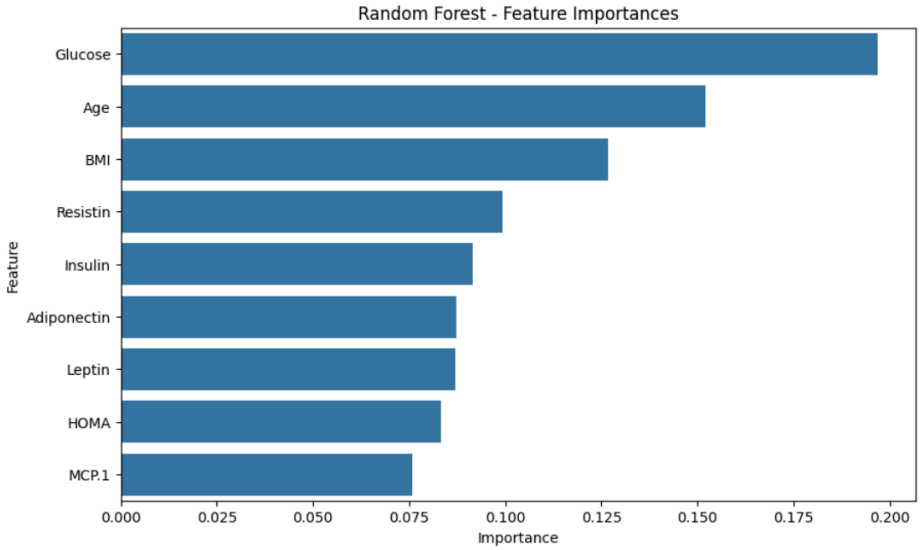
	precision	recall	f1-score	support
0	0.85	0.92	0.88	12
1	0.91	0.83	0.87	12
accuracy			0.88	24
macro avg	0.88	0.88	0.87	24
weighted avg	0.88	0.88	0.87	24



Random Forest:

```
Random Forest Accuracy: 0.7916666666666666

Classification Report (Random Forest):
              precision    recall  f1-score   support

           0       0.82      0.75      0.78        12
           1       0.77      0.83      0.80        12

    accuracy                           0.79        24
   macro avg       0.79      0.79      0.79        24
weighted avg       0.79      0.79      0.79        24
```





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.

ANSHI TIWARI
DISC - 56

05/05

AIDS Assignment

Q1] What is AI? Considering the covid-19 pandemic situation, how did AI help to survive and renovate our way of life with different application?

- AI is the replication of human intelligence in machines, enabling them to learn, reason, solve problems and make decisions without direct human intervention.
- 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 telemedecine and enhancing remote work and online education through AI driven automation

Q2] What are AI agents ~~terminology~~? Explain with examples.

- AI agents are systems that perciere their environment, process information and take actions to achieve specific goals.

1) Agent - An entity that interacts with the environment
2) Environment - The external system where agent operates
3) Perception - Data collected from sensors
4) Actuators - Components that execute actions
5) Rationality - The ability to make optimal decisions
6) Autonomy - The degree of independence an agent has

- Types of Agents: simple reflex, model-based, goal-based, utility-based and learning agents.

Q3] How is the AI technique used to solve the 8 puzzle proble...?

- The 8 puzzle problem is solved using AI search techniques

1) Uninformed search :
- BFS (Breadth first search) : explores all possible moves level by level but is inefficient for large problems.
- DFS (Depth First search) : explores one path deeply before backtracking but may get stuck in loops.

2) Informed search (Heuristic Based) :
- Best-First search (greedy algorithm) : selects moves based on heuristic values like misplaced tiles.
- A* Algorithm : uses the heuristic function $f(n) = g(n) + h(n)$ where $g(n)$ is cost to reach current state and $h(n)$ is the estimated cost to the goal.

Q4] What is PEAS descriptor ?
- PEAS is a framework with Performance measure, Environment, Actuators, Sensors that is used to define the components of an AI agents by specifying how it interacts w/ environt.

o Taxi Driver
- P : Safety, speed, customer satisfaction, fuel efficiency
- E : Roads, traffic, pedestrians, weather conditions
- A : Steering, acceleration, brakes, turn signals
- S : GPS, cameras, speedometer, LiDAR, fuel garage.

2) Medical Diagnosis system

- P - Accuracy of diagnosis, recovery rate, response time
- E - Patient Data, Medical records, Symptoms, lab reports
- A - Display diagnosis, prescribe medication, test recommendation
- S - Patient input, test results, doctor's notes.

3) A music composer

- P - musical quality, originality, pitch, tune
- E - Genres, preferences, historical compositions
- A - generating notes, melodies, instrument selection
- S - user feedback, music database, emotional tone

4) An aircraft Autolander

- P - Safe landing, smooth touchdown, comfort
- E - weather conditions, runway, airtraffic, altitude
- A - Flaps, landing gear, brakes, engine thrust control.
- S - Altimeter, GPS, wind speed sensors, radar, cameras

5) An essay evaluator

- P - Grammar accuracy, coherence, relevance
- E - Essays, writing rules, predefined grading criteria
- A - Store assignment, grammar suggestions, feedback
- S - Text input, grammar, plagiarism check, semantic.

6) Robotic sentry gun for KCLK lab.

- P - Accuracy, target identification, security, effectiveness
- E - Lab perimeter, threats, lighting
- A - Rotating turret, firing mechanism, alarm system
- S - Motion detectors, infrared sensors, camera, radar.

Q5] Categorizing a shopping bot for an offline bookstore

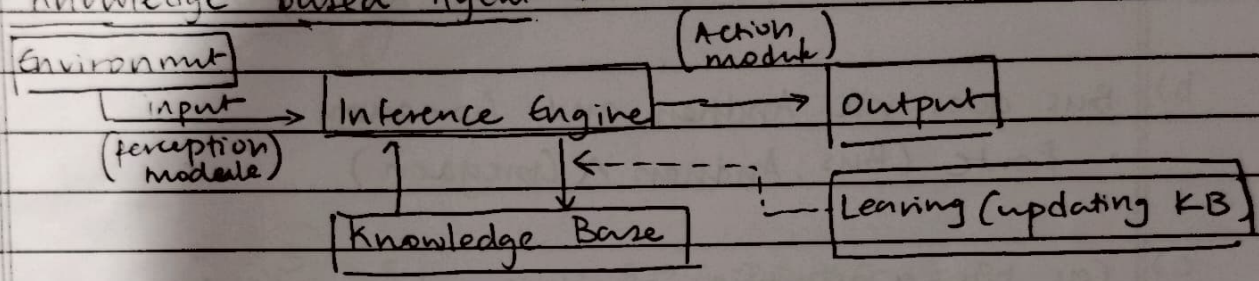
1. Observability :- Partially observable - the bot may not have full knowledge of customer's preferences making its knowledge incomplete.
2. Stochastic :- Book availability, customer behaviour and price changes introduces randomness, making it unpredictable
3. Sequential :- Each decision affects future interactions
4. Dynamic - The environment can change as customers go in and out, stock and price changes
5. Discrete - discrete choices such as recommending a book or checking stock or processing a purchase.
6. Multi-Agent - The bookstore staff, customers, stock and other AI systems make it a multi-agent system

Q6] Differentiate Model vs Utility based Agent-

Model based Agent	Utility based Agent
◦ uses an internal model of the environment to make decisions	◦ chooses actions based on maximizing a utility function
◦ predicts future states using the model before acting	◦ evaluates multiple possible actions & selects best based on utility
◦ Focuses on how the environment works	◦ Aims to achieve the best possible outcome.
◦ May not always take the best possible action, only feasible one	◦ Always selects the action with the highest benefit
◦ A self driving car using a traffic model to predict congestion	◦ A stock trading AI selecting the trade with the highest expected profit.

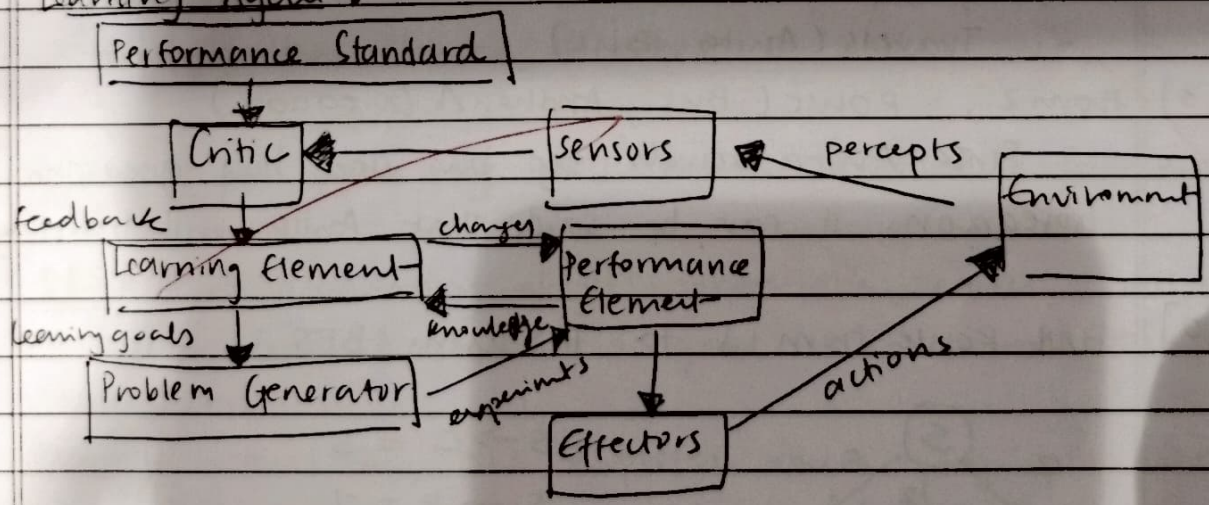
Q7) Explain the architecture of a knowledge based agent and learning agent.

→ Knowledge Based Agent:



- KB: stores facts, rules and background knowledge structured
- Inference Engine: Applies logical reasoning to drive conclusions
- Perception module: gather inputs from sensors
- Action module: Execute actions based on drived conclusions
- Learning module: updates KB with new information.

→ Learning Agent:



- Learning element: learns from interaction & updates knowledge
- Performance element: uses the learned knowledge
- Critic: Evaluates agents performance
- Problem generator: Suggests new exploratory actions.

Q9] Convert the following to predicates:

a] Anita travels by car if available, otherwise by bus.

- Available (car) ⇒ Travels (Anita, car)
- ¬Available (car) ⇒ Travels (Anita, Bus)

b) Bus goes via Andheri and Goregaon

- Route (Bus, Andheri ∧ Goregaon)

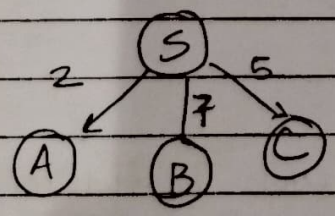
c) Car has a puncture, so it is not available

- Puncture (car) ⇒ ¬ Available (car)
- Given : Puncture (car)
- Therefore, ¬ Available (car)

Forward reasoning:

1) From 3, ¬ Availble (car)
2) From 1, ¬ Available (car) ⇒ Travels (Anita, Bus)
   ∴ Travels (Anita, Bus)
3) From 2, Route (Bus, Andheri ∧ Goregaon)
   ∴ Since Anita travels by bus and bus goes through Goregaon, it can be said that Anita will travel by Goregaon

Q10] Find Route from S to G using BFS:

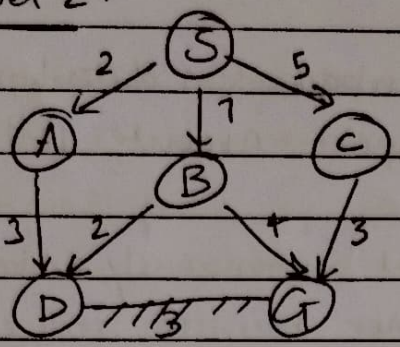


S → C = 5

S → B = 7

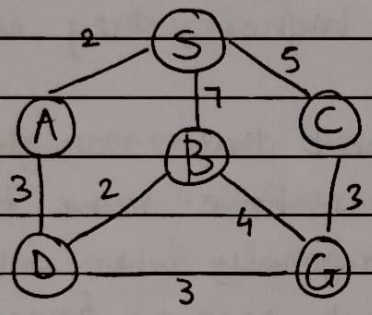
S → A = 2

Level 2:



S → A → D = 5
S → B → D = 9
S → B → G = 11
S → C → G = 8

Level 3:



S → A → D → G = 8
S → B → D → G = 9
S → B → G = 11
S → C → G = 8

So the minimum cost is through S → A → D → G and by S → C → G which is 8 units. Since they have the equal cost, we can choose S → C → G since it has lesser number of nodes.

Q11] What do you mean by DES? Explain iterative DS with example.

→ **DLS**: It is a variation of DFS where the search is restricted to a specific depth limit. If a goal is not found, the search terminates.

Example: If L=2, DLS explores nodes only upto depth 2.

→ **IDS**: It is the combination of DFS and BFS by running DLS with increasing depth limits (L=0, 1, 2 ...) until goal is found.

Example: DLS with L=2, expands S to A, B, C, and then to D and G until goal is found at depth 2.

Q12] Explain hill climbing and its drawbacks in detail with example and state limitations of steepest-ascent hill climbing.

- It is a local search algorithm that continuously moves toward the best neighbouring state with a higher heuristic value.
- Example: 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, they may get stuck.
- Drawbacks:
  - the algorithm may stop at a peak that is not global optimum.
  - flat region where all neighbours have same heuristic value.
  - algorithm may fail to climb diagonally when only direct moves.
  - Once it moves forward, it cannot recover from a bad decision.
- limitations of steepest Ascent hill climbing:
  - If algorithm has to do small incremental changes, it gets stuck midway.
  - choosing only the steepest path can cause premature convergence.
  - Evaluating all neighbours increases computation time.

Q13] Explain stimulate annealing and write its algorithm.

- It is a probabilistic optimization algorithm inspired by the annealing process in metallurgy. It helps in finding global optimum.

Algorithm:

- initialize current state and temperature T.
- select a random neighbour state, compute energy differences
- if $\Delta E \leq 0$, move to the new state
- Else accept it with probability $e^{-\Delta E/T}$ and reduce T
- Return the best solution.

Q14] Explain A* algorithm with an example.

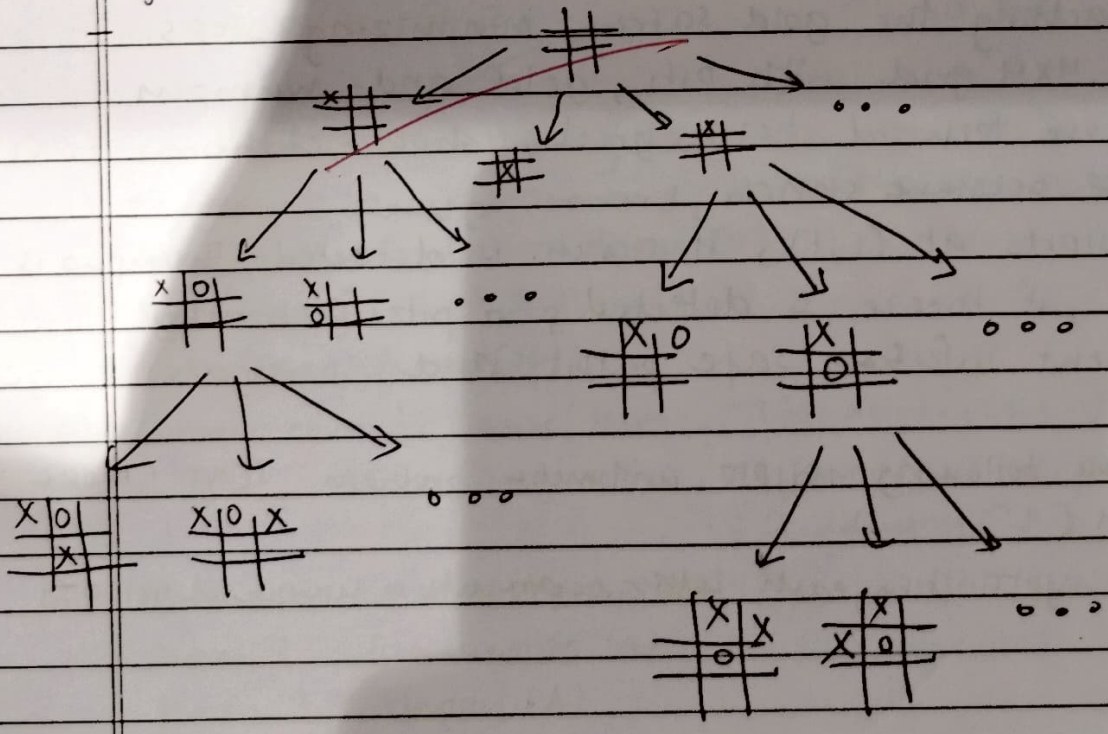
– This is an informed search algorithm that finds the optimal path by considering both the cost to reach g(n) and the estimated cost to goal h(n) : f(n) = g(n) + h(n)

Example: In a graph search problem, if S starts and G is the goal, A* expands nodes based on the lowest f(n) value. Used in video games, maps, maze solver.

Q15] Explain minmax Algorithm and draw game tree for Tic Tac Toe.

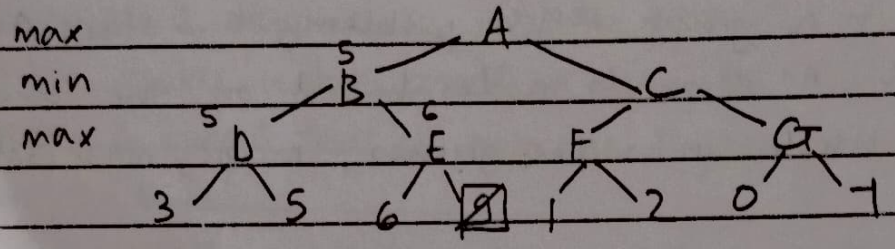
– It is used in adversarial search (eg 2 player games) to determine the optimal move by assuming both players play optimally.

– Maximizer tries to get highest score and minimizer tries to get the lowest.



Q16] Explain α-β pruning algorithm for adversarial search w/ example.

- α-β pruning optimizes minimax by skipping unnecessary branches, reducing computations. α is the best score maximizer can achieve and β is the best score minimizer can achieve.



Q17] Explain WUMPUS World Environment giving its PEAS description. Explain how percept sequence is generated?

- The WUMPUS world is a grid based AI environment where an agent explores a cave while avoiding hazards like pits.
- PEAS descriptor:
  - P - reaching the gold safely, minimizing steps
  - E - A 4x4 grid with pits, gold and wumpus
  - A - Move forward, turn, grab, shoot, climb
  - S - perceive stench, breeze, glitter.
- Agent starts at (1,1), if stench is detected, wumpus is nearby, if breeze is detected, a pit is nearby
- The agent infers safe paths and navigates to gold

Q18] Solve the following cryptoarithmetic problem: SEND + MORE = MONEY

- In the cryptarithm, each letter represents a unique digit 0-9

```
  SEND
+ MORE
-------
 MONEY
```

let M=1 and O=0

$\therefore S+1 = 10 \rightarrow S=9$

```
  SEND
+ 10RE
------
 10NEY
```

let R=7 and E=5

$\therefore Y = 2$

```
  9 5 N 7
+ 1 0 7 5
---------
 1 0 N 5 2
```

$\therefore$

S=9	
E=5	
N=6	Final Answer
D=7	
M=1	
O=0	
R=8	
Y=2	

N=6, E=5

```
  9 5 6 7
+ 1 0 8 5
---------
 1 0 6 5 2
```

Q9] Consider the following axioms:

1) Represent these axioms in first Order Predicate Logic:

$\forall x$ (Graduating(x) $\rightarrow$ Happy(x))

$\forall x$ (Happy(x) $\rightarrow$ Smiling(x))

$\exists x$ (Graduating(x))

2) Convert each to clause form: $\neg$Graduating(x) $\vee$ Happy(x)

$\neg$Happy(x) $\vee$ Smiling(x)

Graduating(A)

3) Prove that "is someone smiling"

from 3: Graduating(A)

from 1: Happy(A)

from 2: Smiling(A)

Q20) Explain Modus Ponens with a suitable example.

- Modus Ponens (law of detachment) is a rule of inference stating : $P \rightarrow Q$, $P \Rightarrow Q$
- Example :

1) If it rains, the ground gets wet (Rain → WetGround)
2) It is raining
3) conclusion : The ground is wet (Applying modus ponens)

Q21] Explain forward chaining and backward chaining algorithm.

→ Forward chaining is data driven inference that starts from known facts and applies rules to reach a goal. It is used in expert system

- Example - Fact : "sore throat"

Rule : "If sore throat → infection"
New fact : "Infection"
Rule : "If infection → need antibiotics", conclusion: "need antibiotics"

→ Backward chaining are goal driven inferences where you start from the goal and works backward to find supporting facts that is used in AI reasoning & theorem proving.

- Example - 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.